Quantifying the Impact of Transit Rider Perception and Satisfaction on User Cost: An Experiential Sampling Approach Using Smartphones

by

Ian Dunlop

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Civil Engineering

Waterloo, Ontario, Canada, 2018

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Portions of this thesis have been previously published in a conference preceding or presented at conference. The following provides a listing of the sections of the thesis that have been previously published/submitted for publication in whole or in part and the citation for the publication.

Content of Chapter 3

Dunlop, I., Casello, J. and Doherty, S. T. (2015) “Tracking Transit Rider Experience: Using Smartphones to Measure Comfort and Well-Being Throughout the Trip” (15-5944) - F17 94th Annual Meeting of the Transportation Research Board, Washington, D.C., January 11-15.

Abstract

The aim of travel demand modelling is to accurately estimate trips in future time periods by understanding current travel behaviours. Accurate generalized cost modelling for determining transit mode share has proven elusive. Arbitrary weight factors are used to adjust models to reflect real world modal split. It is believed that what is missing are incorporating perceptual influences, and more generally, how the satisfaction associated with a mode of transport plays into making transit a less cost-competitive and desirable travel alternative than cost models predict.

This research uses experiential sampling method to capture a transit rider's perception of satisfaction in real-time by using their own smartphone as an observer. Supplemental data collected by the smartphone's built in technologies also enable tracking the rider's movements, linking perception to a physical point. A smartphone application was developed and tested to gather their perceptions and satisfaction of transit system performance. Data from the application were spatially matched to the transit authority's own automatic vehicle location and passenger count data to validate on-time performance and service loads.

The results demonstrated that while a smartphone application is an effective way to collect data in real-time, and a rich dataset was produced, transit rider satisfaction is not as easily quantified as expected. Each individual has their own perceptions of what it means to be satisfied in experiencing their daily activities. Satisfaction is more effectively inferred through qualitative observation, rather than asked directly. However, the results show correlations between satisfaction and transit system performance, attributes, and the rider's sensitivity to time. Participants reported significantly lower levels of satisfaction with their trip experience when their perceived wait time was longer than actual wait time.

Deploying this application to a diverse population sample may help to identify the triggers of transit rider dissatisfaction and anxiety towards determining a cost penalty against transit modal split, and where investments can be made to improve the rider experience. However, satisfaction is revealed to be a subjective concept that is difficult to measure directly.

Acknowledgements

The author thanks Dr. J. Casello (University of Waterloo) for his supervision, mentorship and generous support of this research, Kevin Yeung (University of Waterloo) for deploying the online pre-survey, Rafik Said (Wilfrid Laurier University) for coordinating survey participants and data collection, and Dr. S. Doherty (Wilfrid Laurier University) for facilitating the earlier work.

Table of Contents

List of Figures

List of Tables

1 Introduction

The aim of travel demand modelling is to accurately estimate trips in future by understanding current travel behaviours shaped by land use, demographics and the transportation network, to inform the regional planning and development process. Utility theory is the popular basis for logit models that predict the probability of selecting one mode of travel over another, such as a trip by private vehicle versus by public transit. Utility takes the form of a generalized cost comparison between the available modes. In its simplest form, generalized cost (GC) between an origin i and a destination j by mode m is comprised of the total travel time multiplied by an appropriate value of time (VOT), plus out of pocket expenses:

$$GC_{i,j}^{m} = VOT(\sum time\ components) + \ out\ of\ pocket\ expenses \quad (1)$$

Logit models are predicated on the assumption that the least expensive mode of transportation (i.e. driving by car, walking, public transit, or cycling) will also be the most likely, or the most popular mode. In most North American cities, the most popular mode tends to be the private automobile, because it offers the shortest travel time and greatest convenience. The results change in dense urban cores, where congestion and the cost of parking tip the balance more favourably towards high capacity transit or active forms of transportation, such as walking or bicycling. However, in suburban areas, transit tends to lag behind predicted demand, even when models take into account lower development density and greater distances . The logit model tends to overestimate the proportion of trips that should be taken by public transit, indicating that there are other factors to be considered in how people value and perceive their time, expenses and convenience in deciding how to travel.

Accurate generalized cost modelling for transit has proven elusive. Weight factors are empirically derived to adjust the cost model to reflect the real world split between transit and the most popular mode, automobiles. The factors are used to calibrate the model to best match the observed results. A weakness in this approach is it is not sensitive to changes in behaviour over time. One possible approach to improve models (and modeling) is to incorporate perceptual influences, and more generally, how the overall satisfaction of the transit rider influences the cost-competitiveness and desirability of travel alternatives. Unlocking perception as a barrier to modal choice could improve the accuracy of transportation modelling by making adjustment factors less arbitrary.

However, there is a problem. One, well-recognized source has made the assertion that traveler behavior cannot be modeled:

> Traveler behavior is currently represented in a highly aggregate manner. Factors influencing travel behavior (e.g., value of time or value of reliability) for different sectors of the traveling public are impossible to model with the current four-step process. This makes it difficult to represent travelers' responses to access improvements such as fare integration, changes to parking pricing, fare and parking integration, real-time customer information, and improved transit scheduling to reduce transfer time. (Transportation Cooperative Research Program, 2011).

Compounding this problem is determining whether or not travellers consciously make rational decisions about their travel behaviour on a daily basis in the first place. That rationality in travel choice depends on who are the "choosers" and the "captives." A chooser has a choice between modes that may be enabled by car ownership. A captive lacks choice, perhaps by socio-economic means, physical well-being, or other form of path dependency.

The perceptions of each type of traveller will be different, notably the perception of time.

Contemporary research efforts in the field of transit mode share modelling have focused on different weight factors for the time components of the model, based on how a transit user perceives time. The time components are: wait time at a bus stop, time in-vehicle, transferring, and service delay. The perceived value of wait time and travel time are different. Swait, Ben-Akiva and Mosche (1986) found that only 10% of workers actually make comparisons between the travel choices available to them, if there is even a choice set of alternatives available. But there is more to the real-world experience than time perception alone.

Another challenge is to understand what the barriers are to initiating behaviour change and other intrinsic biases. Anxiety and discomfort associated with travelling on transit, whether perceived or actual, contribute to the disutility in mode choice. The travel experience immediately before and during a trip on transit could be targeted for improvement, to make the choice more attractive. Therefore, understanding the rider experience could help inform where improvements to both to the modelling, and to the transit service, can be made. Travel surveys attempt to unlock this information.

An experiential sampling method (ESM) can be employed to capture a traveller's perception of their surroundings in real time, in a relatively unobtrusive way. This thesis outlines an ESM involving the use of the transit rider's own smartphone as an observer. For transportation research, using phones as an ESM offers advantages over traditional paper survey or interview methods because data are not lost in the time lag between the actual user experience and completing a post-trip survey.

ESM enables the capture of the rider's immediate reaction to their surroundings, comfort, and service performance. Supplemental data collected by the smartphone's built in technologies enable seamless tracking of the rider's movement through time and space, linking their perceptions and experience to a physical point. Smartphone applications (apps) are an emerging ESM for data collection. The main focus of the research presented in this thesis is towards the development and implementation of such an app.

This ESM is further enabled with data collected by automatic vehicle location (AVL) and automatic passenger counting (APC) systems of transit service providers. Together, these data can provide a rich assessment of the transit rider's experience from their point of view in comparison to the actual performance data of the transit system. Transit service providers need to appreciate the subtleties of comfort and satisfaction, which go beyond scheduling and route changes. The reliability of the system plays a role, of course. There are, however, other factors at play not the least of which is that the transit user must give up certain aspects of personal control, and overcome their own emotional barriers and anxieties that are shaped by broader life experiences and individual situations.

The overarching aim of this research is to lead to the development of more accurate generalized cost models for transit mode share. Rider satisfaction is a complex metric that is best measured by observing an aggregate of direct and indirect factors influencing the rider's perception of their individual transit experience. As will be revealed in this thesis, directly asking the rider to provide their level of satisfaction over the course of a trip on transit reveals a biased outcome, as individuals tend to recalibrate their perception of satisfaction based on their immediate experience. The rider's criteria for satisfaction change with their surroundings and circumstances as they move through space and time. Satisfaction becomes a compromise between various factors relating to perception meeting expectation at any given time.

1.1 Research goals

The overall aim of this research is to validate a proof of concept for using smartphones as an ESM to gather transit riders' perceptions and satisfaction of transit system performance during their trip. More specifically, the goals of this research are to:

1. Develop and test a smartphone app to track a group of transit riders through space and time to gather their perceptions and satisfaction of transit system performance.

2. Identify the triggers of transit rider dissatisfaction and anxiety that are a cost penalty against transit modal split.

3. Match the collected data to the transit authority's own automatic vehicle location and passenger count data to validate on-time performance and service loads.

4. Observe the influence of the level of experience and time sensitivity of the transit rider on the results.

With this proof of concept, the smartphone app can be used to gather experiential data from a larger representative population to quantify and qualify the impacts of satisfaction, service reliability, comfort and convenience on rider behaviour. Ultimately, such research could lead to more accurate modelling of generalized costs to predict transit mode share.

1.2 Motivation

This research – to develop a smartphone app to measure transit rider satisfaction – had its genesis in human transportation geography. A smartphone-based experiential survey was modified by the author to assess transit rider experience from a project involving the tracking of human activity and well-being in a natural environment (Doherty, Lemieux, & Canally, 2014). This previous work is discussed in detail later in this thesis. The research methodology aligned with the work of Dr. J. Casello and the Waterloo Public Transportation Initiative (WPTI) at the University of Waterloo, so a much larger smartphone-based experiential survey was developed and tested.

1.3 Thesis Organization

The next chapter is a literature review and summary of previous research. Concepts of transportation modelling are explored, along with experience sampling methodology. The work of others in measuring transit rider perceptions and behaviour, which forms the basis for this research, is discussed. Chapter 3 describes a previous research project involving the capture of transit rider experience using a smartphone, including analysis of the results and considerations for the current research. Motivated by that previous project, the method of development and testing of the new transit survey app is explained in Chapter 4. Chapter 5 analyzes the results from the data that were collected. Chapter 6 summarizes the findings, conclusions, and identifies possible improvements and expansion of the survey to a wider demographic group.

2 Literature Review & Earlier Research

This chapter presents a review of research pertaining to transportation modelling concepts, and surveying and sampling methods to study travel behaviour and user perceptions. A variety of research foci are revealed, including methodologies and outcomes to learn from for the development and deployment of a transportation survey app for experiential sampling. The first sub-section discusses transportation modelling and transit mode share.

2.1 Transportation Models

An intended outcome of research into transit rider behaviour and satisfaction is to improve the accuracy of travel demand modelling and shifts in modal split (i.e. trips by private car versus public transit). The aim of travel demand modelling is to accurately estimate trips in the future by understanding current travel behaviours shaped by land use, demographics and the transportation network. This understanding can then inform the regional planning and development process. This goal is achieved by measuring data for a "base year," and developing a model to forecast data for a "horizon year." The 4-Step Model, as shown in Figure 1, below, was until the late 1990s or early 2000s considered the most popular transport model.

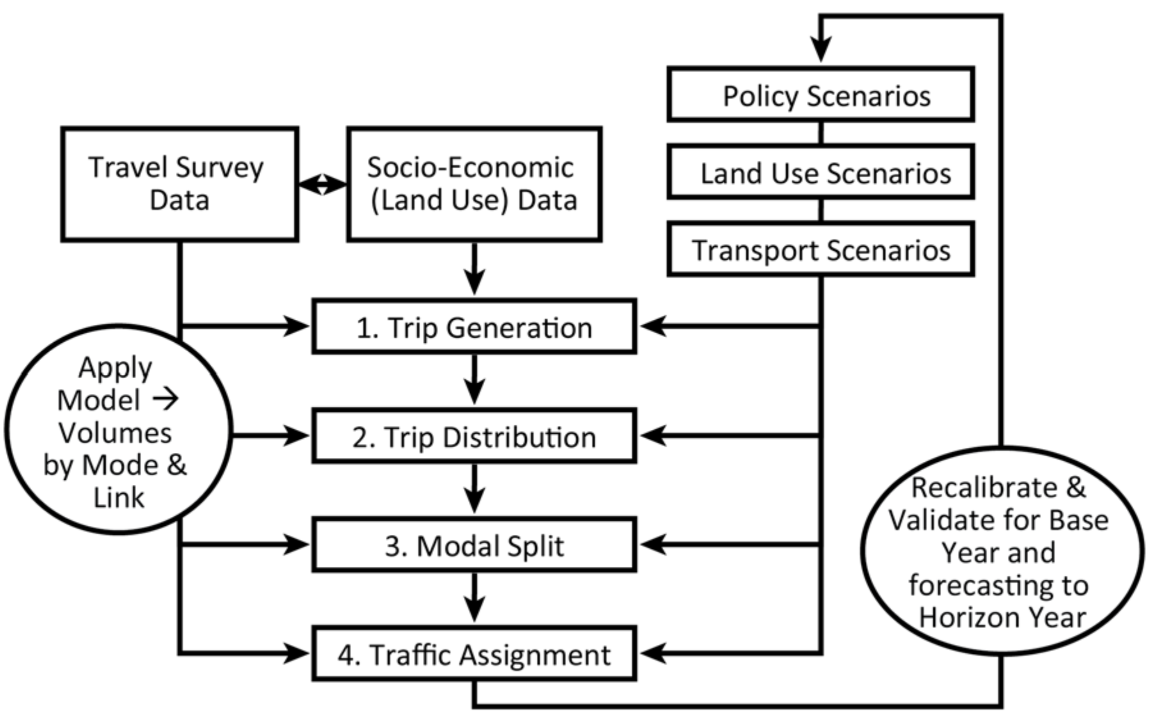


Figure 1 — Basic 4-Step Model (Adapted from Murga (2002), p. 12)

The process begins with a travel survey of a large representative sample of the population, whereby participants are asked to keep a travel diary of their daily trips over a one week period. The region being studied is divided into Travel Activity Zones (TAZ). The 4-step model is then used to analyse:

- Trip Generation – quantifying the number of trips originating in each TAZ, disaggregated by purpose – and Trip Attraction – estimating the number of trips destined for each TAZ;
- Trip Distribution – the linking of trip origins to trip destinations to create travel matrices;
- Modal Split – estimating the proportion of trips made by each available mode between all origins and destinations;
- Traffic Assignment – assigning each of the trips to a path or transit route.

Coupled with the survey results are socio-economic data, including census information, employment, and land use characteristics. To build the model based on the survey results and these data, regression compares dependent and independent variables to test the strength of the relationship between them. Multiple independent variables, such as household size, autos owned, employment and population can be used to validate trip generation or attraction. The model is recalibrated until the simulated result most strongly reflects the actual travel patterns of the survey.

A gravity model is the most commonly used method for modelling trip distribution. The basic form of gravity is an attraction force between two masses, which in this case are TAZs. The attractive force of one TAZ on another (trips originating in one TAZ, destined for another TAZ) depends largely on the distance separating them, thereby creating a deterrence factor. Deterrence is usually a function of generalized cost, which is further described below.

The goal of the travel activity model is look at past observations to project into the future. The relationship of travel between TAZs is represented by the equation below.

$$T_{ij} = T_i \frac{A_j F_{ij} K_{ij}}{\sum_j A_j F_{ij} K_{ij}} \qquad (2)$$

Trips from TAZ i to TAZ j (T_{ij}) are the product of all trips originating in zone i, (T_i) and the attractive strength of zone j (A_j) relative to all TAZs, and the cost of travel from i to j (F_{ij}) relative to the cost of travel to all zones. An adjustment factor (sometimes called the ***k*** factor) is used to calibrate the model to simulate the observed data, smoothing out the imbalances in trip generation and trip attraction between TAZs. The result is a matrix of trips from all origin TAZs to all destination TAZs.

The k factor varies depending on the size and scope of the area being observed, but is derived from a best-fit for a sum-of-means or root-mean-square error (RMSE) test. Therefore, calibration often begins with some arbitrary decisions about what parameters will be used to determine the impedance between zones represented by travel cost (Gray, 1983). This approach "is (very slightly) biased," as the model is only as good as the available input data, which often leads to determining model parameters through trial-and-error (Hollander, 2010). As such, models can only go so far in forecasting travel patterns. The variety of real-world influences contributing to k, which can be empirically observed, measured and projected, is complex. Extraneous assumptions and lack of data will skew the results away from real-world observations.

Generally, UTP (Urban Transportation Planning) recommends the cost be taken as travel time converted to a monetary value (Gray, 1983). Consideration of other attributes improves the accuracy of the model and is reflective of the more subtle influences of travel behaviour, as summarized below, but converting them to a monetary value is an abstract process.

> The cost, C, may be broadly defined as separation, which can include:
>
> - the simple spatial separation of distance;
> - the state of the travel system reflected by actual travel time;
> - extra perceived deterrence of congestion, crowding or waiting for public transport;
> - out-of-pocket costs for fares, tolls or parking;
> - or socio-economic differences, as between a white-collar worker and a blue-collar job. (Shrewsbury, 2012)

Once a 4-Step Model has been calibrated to best represent the travel behaviours of a population, different scenarios for policy, land use, and transport planning can be invoked to see how the travel patterns are impacted. The next section introduces the ideas of utility and mode choice in more detail. These concepts are then discussed in the context of activity based models in section 2.3.

2.2 Using Utility Theory to Determine Mode Choice

Utility theory attempts to understand the benefits (or costs) a consumer receives (spends) in the consumption of a good or product. Utility theory considers both positive and negative characteristics, and their relative importance, as perceived by the consumer. In transportation modeling, the net benefits, or utility derived by a particular transportation option, in comparison to the utility of the other travel options available between an origin and destination can be calculated. The Generalized Cost (GC) of travel by mode m, between TAZ i and j can be represented as follows:

$$GC_{i,j}^{m} = VOT\ (\sum time\ components) + \ out\ of\ pocket\ expenses \qquad (3)$$

GC is in dollars ($) and time Value of Time (VOT) converts time in minutes to dollars. Time components include wait time, travel time in the vehicle, and access time (i.e. time to get to or from the transit stop or vehicle). Out of pocket expenses may be the fare, in the case of transit, or parking and toll charges in the case of automobile travel. The GC for transit can be broken-down as follows:

$$GC^{transit} = VOT(T_A F_A + T_{BS} F_{BS} + T_{IV} F_{IV} + T_{TR} F_{TR} + T_E F_E) + Fare \qquad (4)$$

Where

- T_A = Access time, from trip origin to the boarding bus stop
- T_{BS} = Time waiting at the bus stop
- T_{IV} = Time in vehicle
- T_{TR} = Time in transfer
- T_E = Egress time, from egress bus stop to destination
- F_x = Weighting factor for the perception of time
- VOT = Value of time (in $ per unit of time)

The GC for automobile differs as it is generally assumed that only T_{IV} is a relevant time component because an auto can be driven directly from home to a destination. But, the cost of operating the vehicle needs to be considered, along with other out of pocket expenses such as the cost of parking at the destination or toll charges, if any. The resulting equation is:

$$GC_{Auto} = VOT(T_{IV} F_{IV}) + C * D + Parking + Tolls \qquad (5)$$

Where

D = Distance travelled
C = Cost per distance unit (in $per kilometre)

The cost to operate the automobile tends to be considered by the auto owner as the incremental cost for the trip, meaning fuel and regular maintenance. To own a vehicle is to accept the "sunk costs" of the original purchase price, insurance and licensing which accrue whether the vehicle is used or not. Hence, transit is perceived at a cost disadvantage when one or more vehicles are already owned at a household; the cost disadvantages is magnified in the cases where suburban parking is "free," and there are no road tolls.

Once the generalized costs of the different modes are determined, the probability of choosing one mode over another can be calculated using a logit model, as follows:

$$\Pr(choosing\ transit) = \frac{e^{-GC_{ij}^{Transit}}}{\Sigma_{m=1}^{M} e^{-GC_{ij}^{m}}} \qquad (6)$$

In the above equation, the probability of choosing transit for a trip from *i* to *j* is a function of the cost of the trip by transit in relation to the sum of all of the costs of the other possible modes for the same trip.

The weighting factors in the transit GC formula (3), F_x, are influenced by the perception and experience of the individual, and thus selecting appropriate values can be very subjective. Each time component has a different weight factor due to perception. In-vehicle time is considered to be perceived as the normal passage of time, and assigned a value of F=1, unless there is a delay or the vehicle is crowded. Suburban delay may be perceived as more onerous than delay in a densely urban or chronically congested area. Wait time at the bus stop is more onerous than in-vehicle wait time or delay (Iseki, Taylor, & Miller, 2006). Time components with a high disutility are assigned a weight factor of F >1. Examples of time weights from the literature are shown in Table 1 (Ryus, et al., 2013):

Table 1 - Time Weight Factors

Location	F
In-vehicle time	1.0
Walk time	2.2
Initial wait time	2.1
Transfer time	2.5

The value of F is also influenced by the expectation of the transit rider. A suburban rider expects less in-vehicle delay due to the typical traffic volumes and performance of the street transportation. A rider in a more densely urban area, or along a corridor where congestion is expected, would have a higher tolerance for being in a slow-moving vehicle. Even though the lateness as a result of the congestion may be the same, or even less, in the suburban scenario, the urban rider would feel more at ease because of the predictability of delay. The transit rider in the suburban scenario would also experience a higher level of time anxiety waiting at the bus stop for less frequent service than the urban rider who is confident that, even if he or she missed the last bus, another bus will be along shortly and their timeline shall continue to meet their service expectation.

Time perception can also be influenced by physical characteristics and environmental characteristics. Amenities at the bus stop area where a commuter waits for the bus, the comfort of the bus ride, the weather, crowding, the rider's confidence and experience in using the transit system, and the availability of schedule information all influence whether the trip will be perceived positively or negatively. Travel surveys tend to show that transit has a higher share of the modal split in areas with higher population densities, generally because transit becomes a more convenient option for those traveling there. A number of qualitative factors come into play, as the higher density makes it more cost-effective for a service provider to offer more frequent service, route options, or higher-order (rapid) transit. Encouraging higher-density density development through zoning and investing in related transit improvements makes the generalized cost of transit more competitive with private automobiles, thereby transit's share of the modal split would increase. Below is an example of how attributes influence traveler behaviour.

2.2.1 Changes in Modal Split as a Function of Land Use and Other Attributes

Using data from the Transportation Tomorrow Survey (TTS) data for the Region of Waterloo (Transportation Information Steering Committee, 2008), a multi-nomial logit model was calibrated to model different trip modes. For transit, calibrating the generalized cost along with the attributes of transit passes per household, student density, dwelling type and number of automobiles per household provided the strongest relationship to match the observed results. As shown in Figure 2, if the proportion of different dwelling types (single-family, medium and high density residential) is altered, and thereby population density is manipulated, the modelled mode share for transit increases or decreases in response.

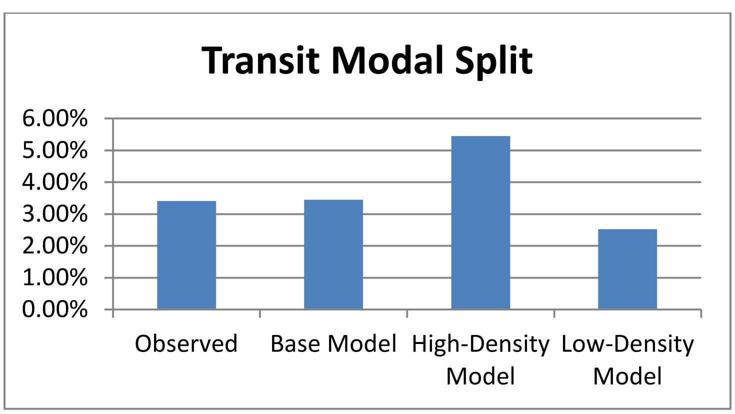


Figure 2 – An example of manipulating a transit modal split model with different variable values.

This example shows the usefulness of transportation models in testing demographic and land use scenarios to see how mode choice would be affected. Current planning trends are towards intensification of urban areas due to hard limits being set on urban boundaries. The high density scenario could have a dramatic effect on transit demand.

This practical example of applying the utility theory belies the method's inherent weaknesses, specifically for transit projections. The base models, prior to being calibrated to the observed values, tends to under- or over-estimate transit's mode share in the network. Calibration only corrects the model to match the point in time of the survey. Paradoxically, transit ridership should be more popular, if based solely on the relative utility of travelling between TAZ by different modes, but other factors come into play that make travel choices more than just cost and distance based.

Triggers of time sensitivity, satisfaction and anxiety of potential transit users influence transit's ability to capture a greater mode share, particularly in suburban areas. The utility or disutility of these attributes can be converted to a cost for modelling purposes. The utility or disutility of local physical conditions, transit service level and reliability, and related infrastructure contribute to those triggers. Instead of using a single aggregate k factor in the 4-step model at the trip generation and trip attraction stage, discrete k values for each TAZ could be modelled based on the physical characteristics (i.e. infrastructure, land use, availability of transportation mode options, distance between O-D pairs), and the socio-demographic characteristics of the zone.

Utility relates to satisfaction. What has proven to be elusive for generalized cost modelling is accurately incorporating these perceptual influences, and more generally, how the overall satisfaction of the transit rider plays into making transit a more cost-competitive and desirable travel alternative. Unlocking perception as a barrier to modal choice could improve the accuracy of transportation modelling by making adjustment factors more meaningful.

2.3 Activity Based Models

Activity-Based Models improve the 4-Step Model by incorporating qualitative factors, in addition to considering trip purposes and the sequencing of individual trips into trip chains or sub-tours. An example is a parent who travels to work and, along the way, drops their children off at daycare and stops at a coffee shop. A similar trip is generated in reverse at the end of the work shift. Additional trips may be generated for evening activities. Travel surveys are used to collect the daily transportation habits of participants, along with household attributes, land use and statistical information.

Vosha & Petersen (2009) note that a "new generation of activity-based models," based on characteristics like the number of licensed drivers in the household, availability of free parking, transit pass holders and other employer-provided travel-related incentives, show a strong interdependence between these attributes. The strongest attribute relationships, as determined through regression analysis, are combined with generalized cost to calibrate a more robust utility function to feed the logit model:

$$U_{transit} = b + F_{transit}GC_{transit} + F_1A_1 + F_2A_2 + F_3A_3 \ldots \qquad (7)$$

Where

- A_X = Attributes influencing transit mode choice
- F_x = Weighting factor for the attributes
- b = Intercept

Activity based models are generally perceived to be more behaviorally representative, because in this formulation, travelers don't make sequential decisions for every sub-component of a tour. Once a tour begins by auto, the mode choice question is no longer evaluated. Instead, the traveler only then chooses destinations and paths. Activity based models are also thought to include better representations of household interactions than conventional, four step models.

Despite these advancements, activity based models still rely on the concept of utility in determining initial mode choice and multiple other decisions made by agents in the model. As such, the introduction of AB models has not eliminated the need for improvements in utility representation.

The next section reviews methods for experiential sampling and how these methods can be incorporated into a transit user survey to capture rider perceptions in real-time and inform future research into more accurate modelling.

2.4 Experience Sampling Method

In the past, travel mode studies often involved paper surveys along with pre- and post- activity interviews. The shortcoming of the interview process is that participants may not recall the specific details of the travel, and if there were any triggers that caused changes in their perceptions or mood along the way. Although this method of surveying has been used consistently for many years, the method is not the most reliable because of the likelihood that the participants may temper their responses or not recall certain details of their activities (Clark & Doherty, 2010).

Experiential sampling methods (ESM) "capture the moment" by recording the content in a person's day-to-day activities along with the "cognitive and affective dimensions of experience" (Hektner, Schmidt, & Csikszentmihalyi, 2007). ESM can be traced to Hagerstrand's (1970) seminal work on time geography. He stresses the importance of observing the micro-situation to create linkages to macro-scale outcomes. In regard to transportation behaviour, he observed "that the car-owner, because of his [sic] random access to transport, has a much greater freedom to combine distant bundles than the person who has to walk or travel by public transportation." (p .9) His reference to bundling of travel activities recognizes how individual trip-chaining contributes to mode choice and aggregate transportation demand.

ESM-based studies have always taken advantage of the latest technology of the day. The use of telephones, pagers, or alarm clocks to cue the user to complete a questionnaire or make an entry in a diary (Fan, Chen, Liao, & Douma, 2012) were considered cutting-edge in the 1970s. Now, the availability of inexpensive, unobtrusive, tracking and recording devices, such as Global Positioning System (GPS) technology and smartphones, have made ESM a practical and rapidly growing field of research. Adaptive software can also change the questions and information presented to the user based on their unique responses and activities.

2.5 Capturing Transit Rider Perceptions

Perception – the emotional responses felt by a person based on their individual experiences and surroundings – is subjective. What perceptions and characteristics of a trip on transit are relevant to measure? Perception and experience also influence whether behavioural and emotional responses are positive or negative. A single negative experience can carry more weight as a "cost penalty" or disutility than numerous positive experiences.

2.5.1 Wait Time and Value of Time

Many factors come into play that affect the perception of wait time. The individual can feel out of control of the situation, uncertain if the bus is running on time, is late, or perhaps they just missed the bus before arriving at the bus stop. Transferring from one route to another also causes anxiety, because the complexity of the trip is compounded. The next link in the trip chain relies on the current leg of the trip performing as expected. Delay can make time seem to pass more slowly because attention to time is more acute than when things are going smoothly and focus can easily drift to other activities.

Several studies on the perception of wait times at bus stops have been conducted. Hess, Brown, and Shoup (2004) compared wait time perceptions of riders who took the first available bus, regardless of having to pay an extra fare, versus those who waited specifically for a free (fare pre-paid) bus even if one or two extra-fare buses passed by. Riders, who waited for a specific bus as other buses passed, overestimated their wait time by a factor of two. However, the value of wait time was influenced by whether the riders perceived that they chose to wait or were forced to wait for their desired bus to arrive. Stated versus revealed wait times were collected by immediately asking the rider how long they thought they waited once they are aboard the bus and comparing to the actual wait time obtained from vehicle location (AVL) data.

The value placed on time is a key to the GC model. Lisco's (1967) seminal research on the value of time revealed that the commuter's "response to trading off increasing amounts of time against a fixed background of other variables will be cumulative normal." Different types of waiting time translate to additional cost for the traveller. A delay in service, particularly in uncomfortable surroundings, increases travel cost. A study in Paris' subway revealed that a delay of 5.7 to 8.1 minutes in the uncomfortable environment of the underground "generated a considerable disutility for users," resulting in an equivalent monetary increase of 29% to 42% for the trip (Haywood & Koning, 2013).

A way to mitigate the negative effects of wait time is to provide real-time transit information at the platform. Being able to monitor when the next vehicle is expected to arrive reduces anxiety and uncertainty, thereby making wait time less onerous for the traveller (Brakewood, Barbeau, & Watakins, 2014 and Gooze, Watkins, & Borning, 2013). This information also helps passengers to adapt to unreliability in transit service performance (Carrel, Halvorsen, & Walker, 2013).

Real-time information does not necessarily need to be via electronic display; data are increasingly available online or with dedicated apps. Use of smartphones provides a distraction as the commuter's time is occupied while waiting. Fries (2011) noted a 32% lower perception of wait time for riders who enjoy "multitasking."

2.5.2 Transit Service Characteristics

Wait time perception is also influenced by the amenities available at the bus stop. The longer the wait, the more consideration riders give to their surroundings. Comfortable amenities can reduce the perceived length of wait. In Yoh, Iseki, Smart, & Taylor (2011), waiting riders were asked to complete a one-page survey that asked them to rate on a scale of one to four the importance of the stop or station attributes. Although the results showed that quality of amenities available at the bus stop improved perception, unfortunately, many people abandoned their lengthy surveys when the bus or train arrived, resulting in many incomplete surveys.

Friman, (2004) discovered riders can hold on to their prior perceptions of inconvenience and discomfort for some time after new transit service improvements have been implemented. Alternatively, some studies have approached public transit service quality from a scheduling perspective. Lui, Bunker, & Ferreira's, (2010) research on origin-destination route choice modeling showed that, when going to a common destination, passengers would select different routes based on their specific perceptions and familiarity of the service, crowding and convenience.

Rider behaviour and satisfaction are a subjective "combination of favourable conditions" (Verron & Martens, 1986) that may not always be evident in, or comparable between, different transit systems and cities. Satisfaction is subject to a wide range of attributes, including trip length, congestion, wait time, reliability and crowding (Fan, Chen, Liao, & Douma, 2012, among others). But, attitudes to these conditions can change over time, as other influencing factors come into play and travellers adjust their expectations (Benjamin, 1986).

Therefore, considering the user's point of view is fundamental in evaluating transit system performance, and must include both subjective and objective measures (Eboli & Mazzulla, 2011). Studies aimed at revealing the behavioral intentions of public transit passengers and discovering the roles of service quality, perceived value, satisfaction and involvement in measuring perception have yielded mixed results. For example, researchers collected in-situ data using a survey founded on qualitative questioning with the responses related to a five point Likert scale (Lai & Chen, 2011). This method enabled a quantitative analysis of the data, leading to the production of a lengthy statistical report, but provided little qualitative insight into the personal experiences influencing the travel behaviors and perceptions.

2.5.3 Negative perception as a cost penalty

An alternative approach to measuring the effects of transit facility and service characteristics is to consider a "cost penalty" when negative perception leads to increased rider anxiety. In terms of service reliability, the cost penalty is added to transit travel time based on the likelihood of the bus being on schedule (Nour, Casello, & Hellinga, Anxiety-based formulation to estimate generalized travel time, 2010). Particularly for risk averse travelers, anxiety about reliability comes with a high perceived cost, which influences their decision to take transit, even though ridership models might suggest they should be candidates to use the service based on demographics alone (Casello, Nour, & Hellinga, 2009).

The cost penalty can be applied to the generalized cost model for transportation mode choice, which will be discussed in more detail in a subsequent section. First, the challenges of observing and measuring rider satisfaction, anxiety, and the perceived value of time—the variables that could contribute to determining a cost penalty—are further explored through a review of related research.

An anxiety cost penalty is exacerbated by the difficulty riders have in accurately planning trips on transit based on the route and schedule information available to them. The uncertainty of arrival times is a frustration, which causes riders to weigh perceived risks of miscalculating against convenience (Bick, 2011). The increasing availability of real-time transit arrival information is expected to reduce this uncertainty, and thereby, the anxiety cost penalty.

2.6 Measuring Traveller Satisfaction

A report published by Statistics Canada (Turcotte, 2010) confirms that public transit users spend more time travelling to work than people who travel by other modes, and dissatisfaction increased with commuting time. Only 10% of commuters travelling 30 to 44 minutes by transit were satisfied with their commuting time; less than half the proportion of satisfied car users. Similar negative disparities between auto and transit satisfaction, attributed to longer commute times, are widely reported in literature. But, there many factors, other than commute time, which could influence trip satisfaction. Gender, age, height, parental background, socio-economic status, and personality traits, such as willingness to take risks, are all characteristics that will influence individual experience (Dohmen, Falk, Huffman, & Sunde, 2011).

Attitudes towards quality of service, or system performance, and thereby satisfaction are "coloured" by past experience, including those that predispose a feeling for or against, and beliefs concerning how something should behave (Forsyth & Smyth, 1986). Driving standards, schedule adherence, trip length, waiting time, appearance, cleanliness, comfort, staff attitudes and other factors are consistently cited as influences of trip satisfaction. The problem for transit users is their satisfaction is dependent upon factors that are mostly outside of their control and relies on a subjective sense of well-being (St-Louis, Manaugh, van Lierop, & El-Geneidy, 2014).

Satisfaction is a perception that is influenced by environmental surroundings, psychological variables and transit service characteristics (Eboli & Mazzulla, 2011; Yoh, Iseki, Smart, & Taylor, 2011), which include the following:

2.6.1 Environmental Conditions at the Bus Stop

Schedule information

Printed bus arrival timetables can be provided, such as an "infopost" on a bus stop sign pole, or a system map display frame in a transit shelter (Eboli & Mazzulla, 2011). Phone-based systems, whereby the rider calls the bus stop number and is dictated the arrival times of the next series of buses are becoming less popular with the advent of web-enabled smartphones. Schedule information tends to be static, or at best may invoke general messages about system delays when inclement weather or scheduled construction are known to impact the scheduled times.

Real-time information display

Digital displays showing the time to arrival of the next series of buses are becoming more commonly available at transit hubs and transfer points. These systems are connected to a central server that is monitoring bus performance in real-time via GPS or transponder. Arrival information is automatically adjusted for system delays.

Bus shelter and comfort amenities

Bus stop amenities, such as a shelter, bench, waste bin, lighting, hard-surface landing pad and staging area (other than sidewalk) provide comfort and reassurance to the waiting traveller (Yoh, Iseki, Smart, & Taylor, 2011).

Location of bus stop (i.e. on busy through arterial, land use diversity, traffic)

Suburban bus stops often lack basic amenities, and may only consist of a bus stop sign attached to a utility pole at the side of the road. Lack of amenities can increase the perception of wait time. In denser urban areas, bus stops may be located on mixed-use streets, with slower traffic, more pedestrian activity, and feel better integrated with the surroundings and provide positive distraction, instead of seeming like an anomaly.

Weather

Weather is beyond control, which is why other factors, like bus stop location and amenities, are so important. It is taken for granted that bad weather is going to influence a transit rider's, or any traveller's, perception of time, and often contributes to increased traffic congestion. As such, the influence of weather overall is neutral, but should be taken into consideration when analysing transit rider perception to, see if weather is skewing results and normalize those observations.

Distance to bus stop from origin
The physical location of the bus stop in relation to origin or destination is another influence of time perception. Although this factor can be considered more relevant to access/egress time, the actual convenience of the bus stop location does influence wait time perception. On one hand, someone with an onerous trek to a stop may find relief when actually arriving at the stop, but a long walk coupled with a bleak bus stop location and arrival time uncertainty could compound an increase in wait time perception.

2.6.2 Psychological conditions of the individual

As noted above, perceptions are subjective and can be conditioned by a traveler's unique personalities or trip objectives. Below are many of the commonly-cited influences on travelers' perceptions.

Time sensitivity
Going to work or an appointment is a more time-sensitive task than returning home. Having to be somewhere at a certain time sets in motion a mental planning process, working backwards from the required arrival time to determine a departure time that allows for a margin of delay en route.

Time of Day
Walking in the very early morning or late at night, when there is no daylight, is less comfortable than walking during the day, when you can see and be seen. This factor is part of safety perception.

Familiarity with the stop/route/system
Taking transit for the first time, or going on a new route, to a new destination, increases time awareness. This factor is a barrier to initial transit uptake with individuals – the uncertainty of trying something new. Some people embrace change, while others resist a change from entrenched behaviour and path dependency.

Vulnerability
Overall perception of safety is influenced by age, gender, psychological isolation, physical ability, and travel preparedness. Feeling less safe increases wait time perception.

Socio-demographic group
Members of certain socio-demographic groups, such as students, are considered transit "captives" because their travel options are more constricted than those of working professionals. Captives to

the lowest-cost alternative willingly or unwillingly give in to the disutility of additional time or inconvenience of that mode (Brown, Hess, & Shoup, 2004). The influence of socio-demographics on wait time is debatable.

Presence of others at the bus stop

The behavior of other riders encountered during a transit trip can have a positive or negative influence on time perception. If the presence of others is perceived as positive, such as for “safety in numbers,” the effect is positive. But if the other individuals are displaying erratic behaviour, such as being under the influence of drugs or alcohol, or play into one’s prejudices, wait time perception will be negatively influenced.

Availability of alternatives

The options available to the traveller, such as the availability of a vehicle in the household, are well documented in the transportation modelling literature. Time perception may increase with an accompanying perception that a more convenient mode of transportation is available (Brown, Hess, & Shoup, 2004).

2.6.3 Transit System Characteristics:

Frequency of service

The ability to simply show up at a bus stop and know that a bus will be along within an acceptable length of time significantly reduces time perception. A maximum of 10 minute headway is generally considered frequent service.

Density and convenience of service

The availability of different routes (serving different destinations) increases utility. If an area is served by only one bus route, the chances of having to transfer to reach a destination are much greater, which compounds the disutility of taking transit.

Connectivity

Likewise to density of service, the connectivity of the available bus route(s) to a variety of destinations, including long-range commuting, need to be considered.

Express and high-order service availability

Survey results have shown the greater propensity to take the express service instead of local bus routes, which is espoused by the Brown, Hess, & Shoup, (2004) study.

Reputation

The reputation of the transit authority may precede its perception among potential new users. Negative news articles of chronic service problems or labour disruptions can perpetuate a poor reputation.

Vehicles (cleanliness, fleet)

Survey results indicate that the cleanliness and "first impression" of the bus influences rider comfort. Comfort begins at the bus stop. If the waiting area is strewn with trash or appears run down, it is not a desirable place to be.

Likelihood of on-time performance (at bus stop, in-vehicle delay)

If a transit authority has difficulty meeting on-time performance standards, then it is seen as unreliable by its customers and confidence of new riders is diminished.

Distribution of jobs and households (and thereby typical distance and travel time)

Different communities each have unique characteristics, so model parameters are not necessarily transferable from one community to another. The distribution of housing and jobs, and thereby the distribution of trips, influences the effectiveness of the transit and transportation network. An inefficient transit system, due to its inability to serve a dispersed market, is likely to have longer wait times and lower new rider uptake.

Consideration of these transit system characteristics are increasingly important to improving the accuracy of transportation models within the community of practice, as noted in (Transportation Research Board, 2007). The factors described in this section, and their positive or negative influence on traveller satisfaction, are summarized in Table 2.

Large scale transportation surveys, like the Transportation Tomorrow Survey, and census information, can fill some of this data gap effectively on a regional or macro level. An ESM technique to capture individual perceptions of transit system characteristics and behaviour is discussed next.

Table 2 - Influences on Satisfaction

Influences on Satisfaction Summary Table		
Attribute	***Characteristic***	***Influence on Satisfaction***
Environmental Conditions at the Bus Stop		
Schedule information	Map or timetable available at bus stop	**+**
	Next bus arrival time is displayed in real time	**++**
	No information available at bus stop	**-**
	No information and low service frequency	**--**
Bus shelter and comfort amenities	Shelter, bench or other comfort amenities	**+**
	Only a sign on a post marks the stop	**-**
Physical location of bus stop	Busy location, more activities, greater perception of safety	**+**
	Remote location, lower perception of safety	**-**
Weather	Varies	**+ / -**
Distance to bus stop	Short distance, highly convenient	**+**
	Long distance, less convenient	**-**
Psychological conditions of the individual		
Time sensitivity	High sensitivity to time constraints; satisfaction influenced by schedule adherence	+ / -
	Low sensitivity to time constraints/scheduling	n/a
Time of Day	Varies	+ / -
Familiarity with the stop/route/system	High familiarity - routine user	+
	Low familiarity - less frequent user	-
Vulnerability	High perception of safety	+
	Low perception of safety	-
Socio-demographic group	Varies	+ / -
Presence of others	Varies	+ / -
Availability of alternatives	Transit is perceived as best transport choice	+
	A more convenient alternative is available	-
Transit System Characteristics		
Frequency of service	Frequent service	++
	Infrequent service	--
Density and convenience of service & Connectivity	Multiple choices are available	+
	Single choice	-
Express and high-order service availability	Express or rapid transit service available	+
	Local bus service only	-
Reputation	Varies	+ / -
Vehicles (cleanliness, fleet)	Well maintained, reliable, clean buses	+
	Older, poorly maintained fleet	-
Likelihood of on-time performance	High reliability	+
	Low reliability	-
Distribution of jobs and households	High density, mixed used development that encourages transit use	+
	Low density, auto-centric development	-

2.7 Measuring transit experience in real-time

ESM data related to personal travel activities can be correlated to actual locations using GPS. GPS also provides trip start and end times, origin and destination, duration of wait times at bus stops, and actual travel time on the bus. The rider's perceived wait and travel time can be compared to the actual durations. By combining both qualitative and quantitative information collection in real-time, a very detailed picture of transit rider perceptions and an evaluation of system and service quality can be obtained. Data quality is also improved by the capability of correlating survey responses to an actual place and time.

Anyone with a smartphone has a device capable of collecting GPS and survey data. There is little concern about hindering participants by having them wear special devices to record their travel experience, as was the case in the early days of ESM study. Participants can use a device with which they are already familiar, and does not draw undue attention to them as they go about their activities. Overcoming the challenges of devising a survey that captures relevant qualitative and quantitative data during a transit trip, and does not interfere or influence the normal perceptions of participants, is explored in the next section. The Transit Oriented Experience Survey (TOES), discussed in the next chapter, was an experiment devised with these criteria in mind, and in consideration of other similar travel activity based studies.

2.7.1 Concurrent Research

Concurrent research using smartphones to gather rider experience can be found in the work of Carrel, Mishalani, Sengupta, & Walker (2016). The goal of their research is to understand the drivers of customer satisfaction through objective (travel time) and subjective (emotional response) ratings based in a 5-point Likert scale. Their results show passenger sensitivity to in-vehicle delay is greater than out-of-vehicle delay and the impact on customer satisfaction can be predicted with an ordinal logit model based on the length of delay.

Nour (2015) developed a data-driven classification model using GPS equipped smartphones to infer transportation mode(s). Through analysis of data from phone's GPS and accelerometer, the mode of travel (i.e. walking, automobile, bicycle, or transit) can be determined with approximately 86% accuracy. The main goal of this research is to automate the identification of the different stages of transit trips. Such a system could enhance the use of smartphones as an ESM by reducing the amount of user-interaction required with the device.

3 Previous Work: Transit Oriented Experience Survey

Work on two of the research objectives for this thesis, using a smartphone app to track a group of transit users and identify triggers of dissatisfaction and anxiety, began in 2012 with the adaptation of a previously developed smartphone ESM application.

A smartphone app called XSam (short for Experience Sampler) was used in 2011 for the research of Doherty, Lemieux, & Canally (2014). The code for that app was adapted by the author to create the Transit Oriented Experience Survey (TOES) in 2012. TOES was field tested with 60 undergraduate students in 2012-2013. Participants were asked to rate their transit travel experience, before, during and after the trip, based on 10 different comfort and wellbeing indicators.

With smartphones being easy to use, portable and fairly non-intrusive, transit riders can record their thoughts and emotions in real time as they experience their public transit trip. The TOES project began as a pilot study to test this style of data collection and observe the results, not only to compare with previous survey techniques, but also to validate this method of data collection. TOES tracks changes in transit rider perceptions and their emotional state before, during and after a transit trip with the goal of collecting useful qualitative and quantitative data.

TOES demonstrated that a smartphone survey application can, more effectively than traditional survey techniques, capture transit rider perceptions, emotional wellbeing and comfort in real-time during different stages of the trip on public transit:

- Before arriving at the bus stop;
- Waiting at the bus stop;
- Travelling on the bus and;
- After departing the bus.

3.1 Overview and intent

Sixty undergraduate students from the University of Waterloo and Wilfrid Laurier University were recruited as participants for the TOES Survey. The rationale behind using students for the initial survey, aside from convenience, is that students pay for transit through their tuition fees. Therefore the transit fare is uniformly not a significant factor influencing their perceptions.

Each participating student first completed a brief pre-screening questionnaire and received some basic instructions about how the TOES app works. The questionnaire gathered the student's gender, age, and their overall familiarity with using transit (i.e. frequently, infrequently or never use local transit). Each participant was then provided a smartphone preloaded with the app.

The app prompts participants to complete a series of questions about their perceptions before, during and after their transit ride. The smartphone enables GPS and time tracking capabilities that were synchronized with the survey to provide the time and place the survey was completed. The data are collected and stored on the smartphone, and downloaded for processing and analysis after the subject returns the device.

3.2 Developing the TOES Smartphone App

The app was developed for the BlackBerry 7.0 platform using the Java programming language, the manufacturer's software development kit (SDK) and the Eclipse development environment. To collect GPS data in synchronization with the TOES app, a separate application called GATE, previously developed for Clark & Doherty (2010), was also installed on the devices. Using the companion software for GPS simplified the survey app programming, although the solution was less reliable than using a fully integrated tracking capability.

3.2.1 How TOES Works

Participants initiate the TOES survey before they are ready to begin their trip on transit. The participant can subsequently initiate the survey iterations manually, or the software will prompt them to complete the next iteration every 6 minutes during their trip. A programming challenge was to prevent the smartphone from terminating the app when the phone goes to sleep. A "persistence" module runs in the background to keep the app alive and wake-up the phone with a vibration and alarm to prompt the participant to complete the next phase of the survey.

The app also contains two sets of five perceptual queries regarding participants' surroundings and feelings of wellbeing. To keep the interface clean and readable, the 10 perceptual responses are split on two screens. Slider bars allow the participant to rate their responses to each question. Each slider records a value between 0 (extreme left-side, negative response) and 8 (extreme right-side response), with 4 being neutral (the middle, default position). In post-processing, the neutral value becomes 0, and the positive or negative emotional responses are +/- 4. At either end of the slider scale are two contrasting adjectives (Figure 3).

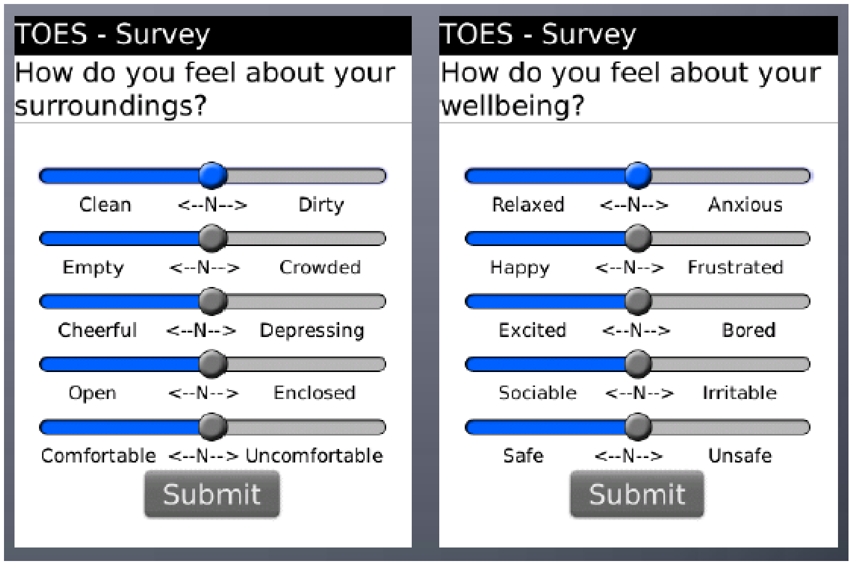


Figure 3 - Device screenshots of two survey screens that use sliders to ask the participants to assess their perceptions of their surroundings and wellbeing.

A series of three voice-recording questions are displayed after the slider responses are submitted, prompting the participant to:

- "Describe anything that has happened that changed how you feel since the last survey or the start of your trip."
- "What is the best or most enjoyable part of this trip, experience, etc.?"
- "What is the worst or least enjoyable part of this trip, experience, etc.?"

The participant can also enter a text response instead of making a recording. The intent of these voice/text questions is to qualify the answers of the slider questions, by revealing the external influences that may be affecting their perceptions and emotional responses. For example, an overcrowded bus, the bus being late, rude passengers, the weather or ride comfort is expected to influence how the participant has responded to the slider questions.

As the participant completes each stage of the survey, data files are written to the phone's memory card. Each data file includes the date, time, question sequence, unique user ID and responses in comma-separated text format. Voice recordings are saved as audio files, which are later transcribed into text by the research team.

3.3 Conducting the Survey

A breakdown of the sixty students and surveys completed is summarized in Table 1. The first survey group (Group 1) followed their regular travel patterns over 2 to 5 days without special instruction on the types or duration of trips they should take. The second survey group (Group 2), were required to complete a more structured travel pattern. Group 2 were asked to travel directly from the University to a shopping centre transit terminal in the north end of Waterloo (Conestoga Mall), and to return via one of three more circuitous routes after a stop-over (see map in Figure 4).

Table 3 - Summary of TOES survey participation and conditions

	GROUP 1	GROUP 2	COMBINED
Date	March 2012	March 2013	
Participants	11	49	**60**
Male/Female	5/6	25/24	30/30
Trips*	24*	49*	**73**
Trips with GPS	*10 (42%)*	*26 (54%)*	*36 (49%)*
Mean Trip Duration**	0:32	1:08	**0:56**
Surveys Completed	110	308	**418**
*Voice Responses****	*196 (44.5%)*	*794 (64.4%)*	
Text Responses	*n/a*	*293 (23.8%)*	
Weather	Unseasonably warm, clear	Seasonably cold, mixed clear/cloudy, wet/snowy	
Trip Type	Daily routine travel	Predetermined; no time constraint	

* Group 1 trips were one-way; Group 2 trips were all return trips.
** Trip duration was the time from the first survey response to the last, not time on the bus.
*** Total responses to the 4 descriptive questions in each survey, with mean response rate.

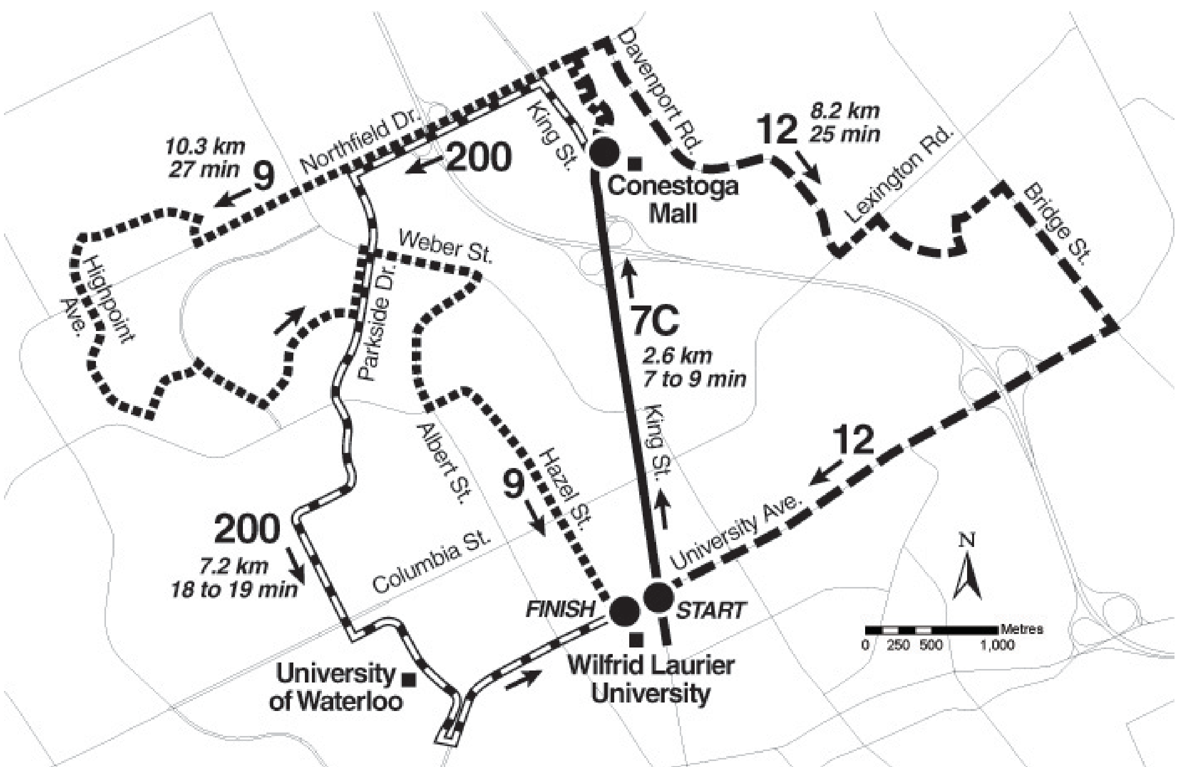


Figure 4 - Route map showing location of Group 2's start, finish and Conestoga Mall

3.4 Analysis

Once each participant completed the survey and returned the smartphone, the data were transferred from the device to PC, and compiled in a master database, along with the transcribed text recordings. A comparison of the response rates and environmental conditions of the two groups are summarized in Table 1, and explained further below.

The overall response rate improved in Group 2 with the addition of the text input option for the descriptive questions. The recorded voice and text responses provided good anecdotal information to correlate the slider question responses. It was possible to have a deeper understanding of why the participant felt a certain way in the situation, the purpose of their trip (i.e. going to class, home or shopping), and also aided in identifying errors when the subject incorrectly entered which stage of the trip they were on.

The low success rate of GPS data collection was a combination of the participants not properly enabling the GPS application when turning on the device (resulting in no data files present), or a malfunction in the device causing it to not obtain a fix on the GPS satellites (resulting in data files written with null coordinates). Data from the ten slider emotional response questions tended to be skewed to the positive side of all the slider scales, with just a few outlier values on the negative side of neutral. Very clear from these results is all participants felt safe on the local transit system, but crowding, and weather (in either good or bad contexts), are commonly on peoples' minds when they ride the bus.

These data were normalized to have the neutral response at '0', and positive and negative values to the corresponding attribute on either side of this axis. Thus, a "Comfortable" response will have a positive index value; while an "Uncomfortable" response has a negative value. The maximum and minimum values are 4 and -4 respectively, corresponding to a Likert scale of 9 possible positions along the slider interface. Although these maximum and minimum values were observed in the discrete data, the mean aggregated data values never varied beyond 2 and -2, with an average standard deviation of 2.

Mean values from the slider questions provide a meaningful pattern to the data, as shown in Figure 5**.** In this chart, the data from the first and second series of slider questions are aggregated into generalized "Comfort" and "Wellbeing" respectively. A trend from *Before* the transit trip begins, waiting at the *Bus Stop*, to *On Bus*, and *After* the trip is revealed.

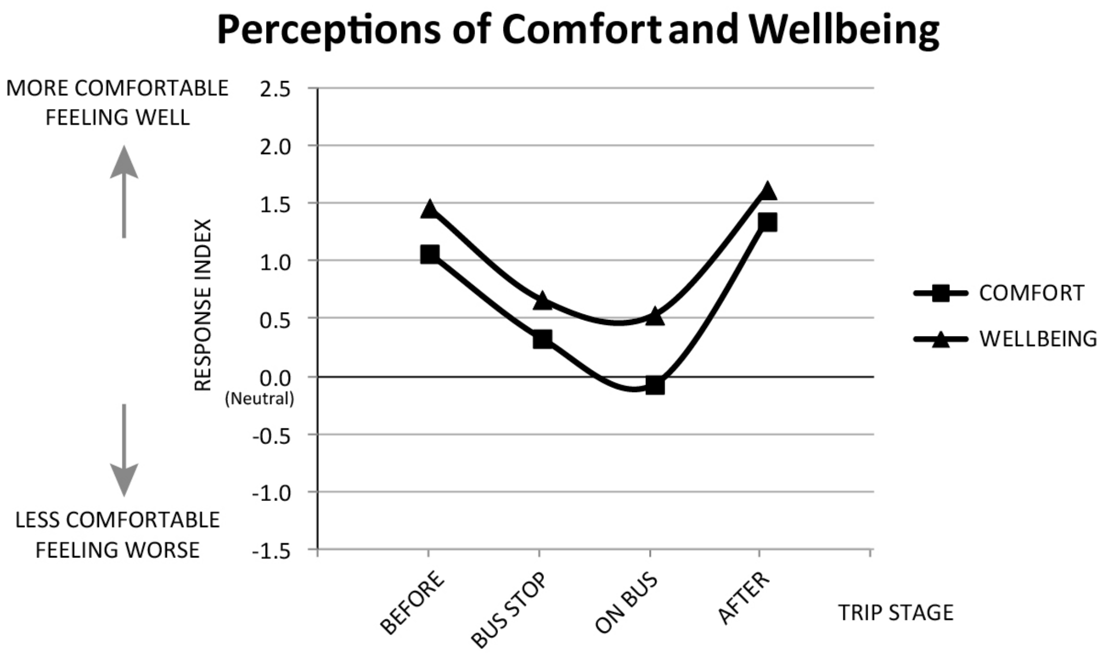


Figure 5 - Mean survey results aggregated as generalized comfort and wellbeing

The aggregated results show the decrease in generalized comfort and wellbeing levels while on the bus. It is interesting that the state of comfort and wellbeing after the trip increases to a level slightly higher than before the trip, indicating that the riders are either relieved that the trip is over or have derived some utility from their trips. The results of the two distinct groups of students, surveyed one year apart, both had the same pattern as shown in Figure 5. However, statistical analysis of the results was inconclusive due to the small sample size.

Figure 6 shows the results for the second group of participants, who had a fixed route from the university to the mall along the mainline, and a suburban route for the return trip (Figure 4). These results show decreasing levels of comfort and wellbeing while waiting at the bus stop at a busy intersection, followed by travelling on the crowded mainline bus. The indicators rebound during a layover at a transit terminal with shopping amenities, and after the trip. The participant's more pleasant experience in taking a quieter bus route from the terminal for the return trip as compared to the outbound route is also evident.

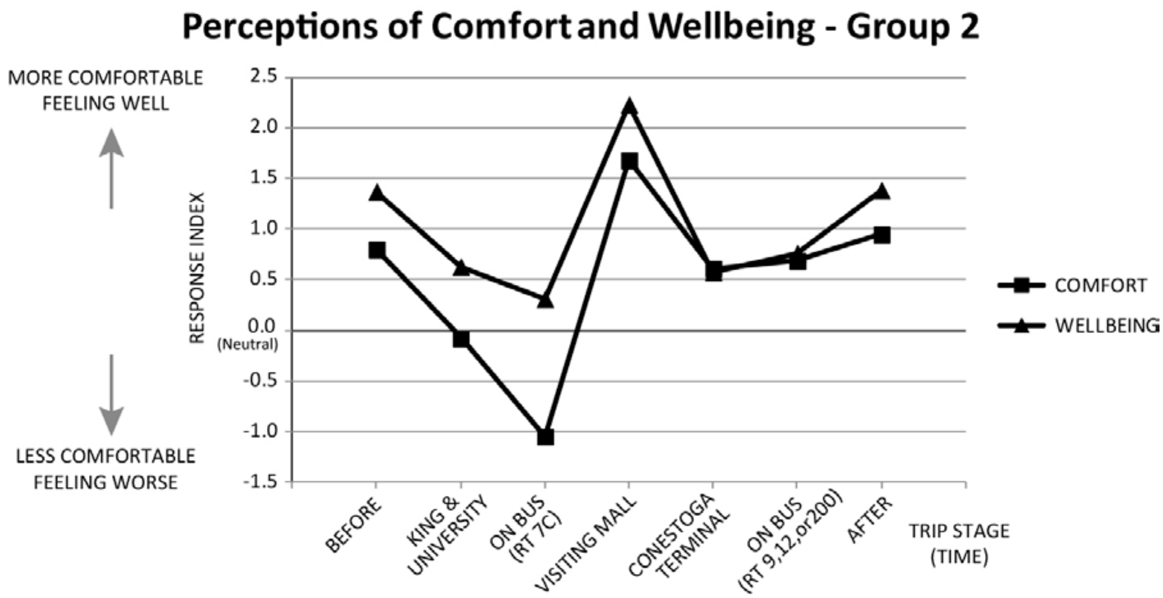


Figure 6 – Normalized Results by trip stage for Group 2 participants. These trips included a stopover visit at Conestoga Mall

Text responses during the survey when asked "What is the best (or worst) part of your experience so far?" provided some useful insights into the comfort and wellbeing data, such as not being able to get a seat on the mainline bus, a rough ride, high traffic and lack of amenities at the first bus stop, and the quieter experience of the return trip on one of the suburban routes. However, some participants felt self-conscious or awkward about speaking into the smartphone without actually talking to someone at the other end and felt as if they were drawing undue attention to their activities.

3.5 TOES Survey Conclusions

The TOES survey provided a small but interesting set of results, and successfully demonstrated the feasibility and practicality of collecting real-time data using a smartphone application.

The results of the survey confirm there is anxiety and discomfort associated with trips on transit. Anxiety and discomfort perception were highest while on a crowded bus, but levels dropped off once the trip was over, and notably improved once the participant was well away from the bus trip environment. The relief of the transit rider knowing they are at their destination, and back in control of their situation, has a significant impact on comfort and wellbeing. The results also confirm that comfort and wellbeing are better on less crowded buses, and new users are more sensitive to crowding, noises, smells, route/schedule uncertainty, and self-consciousness than experienced transit riders who are used to that environment.

As transportation planners and engineers, we can consider how to minimize negative influences through improvements to transit quality. Providing real-time schedule and bus loading information could be one such method, whereby anxiety is reduced by offering more certainty of the bus schedule, and comfort can be increased when users have the option of avoiding the most crowded buses.

3.6 Considerations for Subsequent Research

TOES provided useful results to inform a larger study, and revealed technical issues and weaknesses that can be resolved to improve the data gathering technologies' reliability. By learning from this work, a more robust survey application and methodology were implemented for the subsequent work. It was recognized that a greater diversity in subjects would make the results relevant to a broader transit audience, rather than just the student population.

To capture wait time perceptions, a time constraint needs to be imposed on the survey participants to simulate a real-world travel scenario. It is assumed that anxiety relating to a requirement to arrive at a destination within a specific time window changes rider perceptions and introduces a "cost penalty" (Nour, Casello, & Hellinga, Anxiety-based formulation to estimate generalized travel time, 2010) for arriving too early or too late.

The following recommendations from the TOES project were implemented in the development of the subsequent survey, named the "WPTI Survey", described in the next chapter:

- Integrating GPS data collection into the app, to improve reliability and better synchronization with survey results.
- Implementing an expert system, which modifies survey questions based on the participant's responses and current situation.
- Recompiling the app to support multiple smartphone platforms, allowing participants to install the app on their own phones instead of being limited to using dedicated research equipment.
- Modifying the survey logic and flow to improve consistency of the results, and taking a larger sample.

4 Methods of the WPTI Survey

Developing and testing a smartphone app to track transit riders' perceptions was a core research goal for this thesis. An ESM approach using smartphones to validate and quantify the impact of rider satisfaction could enhance the understanding of disutility and cost penalty to strengthen modelling efforts.

The Waterloo Public Transportation Initiative (WPTI) Survey project was launched in Spring 2013 to test proof-of-concept of using smartphones as an ESM to measure transit rider satisfaction over the duration of their individual trip on public transit. The research team of Dr. Jeff Casello, Kevin Yeung and the author developed the framework for the pre-survey questionnaire, which was deployed online, and determined the desired in-app functionality for to be programmed. The author wrote the programs in XCode (Objective C-based) for Apple iPhone and iPad, and in Eclipse and the Android Application Programming Interface (Java-based) to support Google Android devices. Source code is included in the Appendix. The Android app was also made compatible with BlackBerry 10 devices. This approach was much more efficient than earlier TOES survey, which used the researchers' pre-loaded devices. The participant could also use a device that they were already familiar and comfortable with using.

The rationale for how the pre-trip and survey app components were developed are explained below.

4.1 Overview and Purpose

The WPTI Survey is a component of research into understanding and quantifying how travelers perceive their experiences when traveling by public transit. The underlying hypothesis is that there are many factors that can negatively influence a traveler's experience including:

- poor on-time performance
- unclean or unpleasant surroundings
- overcrowding on the vehicle
- poor interactions with drivers or other passengers, etc.

A traveler's experience is also influenced by their personality (i.e. risk tolerance/risk aversion) and their trip purpose (i.e. work, recreational, school, shopping). The goal of the WPTI survey is to capture these experiences in real-time throughout the travelers trip on transit.

A two stage research approach to quantify travelers' perceptions was developed by the research team. The first stage involved an on-line survey in which respondents answered questions relative to the sensitivity to arriving "on-time" for meetings of various types. The respondents also provide information on their demographics, such as their age, income, and auto ownership.

After the participant completed this initial survey, they are invited to download the WPTI Survey smart phone application (app). Prior to setting out on a trip, the participant is asked for the purpose of the trip and several questions to assess their time sensitivity including the time they need to be at their destination and whether or not they know when the next bus will be arriving at the bus stop. Before, during and immediately after the in-vehicle portion of the trip, the participant was asked to rate their satisfaction with the trip up to that point. Slider bars in the app’s interface allow the respondent to choose from fully-satisfied to fully-dissatisfied. If a change was noticed in the respondent's satisfaction response, the app was coded to provide a set of supplementary questions to ascertain what may have occurred to change their satisfaction level.

The app also made use of the GPS functionality of the participant's phone. Using a combination of GPS data and time, it is possible to assess how deviations from transit schedules are perceived by users of various personality profiles. This information can be combined with the Regional transit system’s data on the vehicle's loading and on-time performance (AVL/APC).

Ultimately, the desired outcome of the research is a quantitatively assessed and improved representation of user satisfaction that informs the appropriate weight factors for the time components of transit GC modelling. However, the initial sample set was limited to student participants to test the effectiveness of the WPTI survey methodology before it gets rolled out to a larger sample set that is more representative of the population.

4.2 Methods

Participants in the survey were given a brief presentation about the project, instructions for completing the pre-survey and installing the app. They were directed to a web page which included the pre-trip survey (Figure 7), app download links, detailed instructions and additional project-related. The two-stage survey process (pre-trip and in-app) was designed to be easy to follow and intuitive. The Office of Ethics Research provided full ethics clearance (ORE #: 19639) for the project to proceed with undergraduate students as test subjects.

4.2.1 Stage 1: Pre-trip Survey

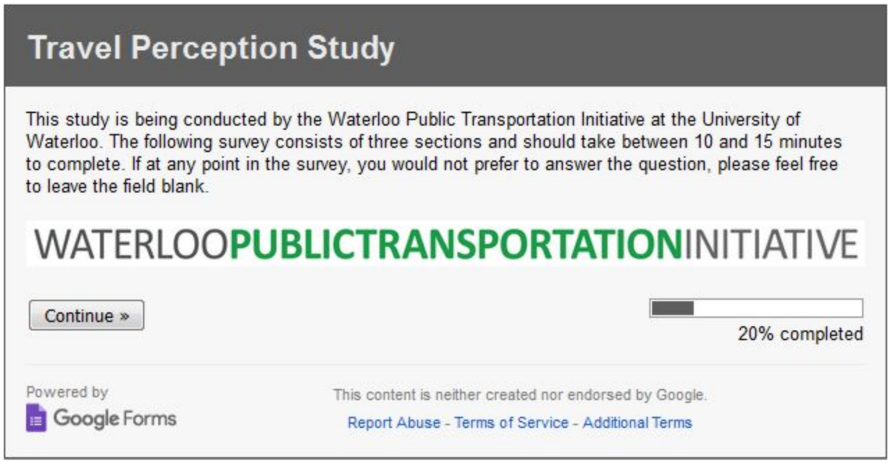


Figure 7 – PreTrip

The Pre-trip Survey captures a detailed demographic profile of the participant. The survey begins with basic traveler information, such as gender, age, income, and employment status. The survey asks a series of questions to ascertain their sensitivity to delay, being on-time, and how they would plan their schedule to arrive at an appointment. Time sensitivity is considered critical to the understanding of how the participant perceives time, and thereby how their perception of time might influence their satisfaction while taking a transit trip compared to travel by other modes. The time questions, possible responses and rationale for each, are as follows:

How important is being "on-time" to you? *(Rate on a scale of 1 to 5)*

Suppose you are travelling to an important meeting that starts at 9:00AM. At what time would you normally arrive for the meeting? *(Provide the time of arrival).* This question aims to determine if the participant would normally allow for a buffer, or window of time before their meeting start time.

Normally, this trip takes 20 minutes with no traffic. At what time would you leave for this meeting? *(Provide the time of departure).* With this question, we can see whether the participant would allow extra time in case there is traffic, or is simply aiming for their stated arrival time.

If you are travelling to the mall to meet with several friends, you are most likely to arrive:

a. *First in your group*
b. *Last in your group*
c. *Neither first or last*

This question is sociologically based, under the assumption that a person who usually arrives before others in their group is more time-sensitive than the later arrivals.

What is the MAXIMUM number of minutes BEFORE the agreed-upon time that you would arrive at the mall? *And,* ***what is the MINIMUM number of minutes BEFORE the agreed-upon time that you would arrive at the mall?*** *(Various time choices are offered).* Following on the previous question, we can see if there is any difference in time-sensitivity for arriving at a social engagement instead of an "important meeting," as posed earlier.

The participant is then asked whether or not they agree with a series of statements to further ascertain their sensitivity to time (Figure 8).

11. Rate your agreement with each of the following statements:
Being "on-time" means...

	Strongly Disagree	Disagree	Neither Disagree or Agree	Agree	Strongly Agree
Arriving at least 10 minutes before an agreed-upon time					
Arriving between 5 and 10 minutes before an agreed-upon time					
Arriving between 0 and 5 minutes before an agreed-upon time					
Arriving at least 5 minutes after an agreed-upon time					

Figure 8 - Time Sensitivity Questions

The next section of the pre-survey asks the participant about their travel experience. They are asked how often they travel by different transportation modes: Walk, Bicycle, Public Transit, Car Passenger, Car Driver, and Other (Figure 9).

12. In a typical summer month, how often do you travel on each mode?

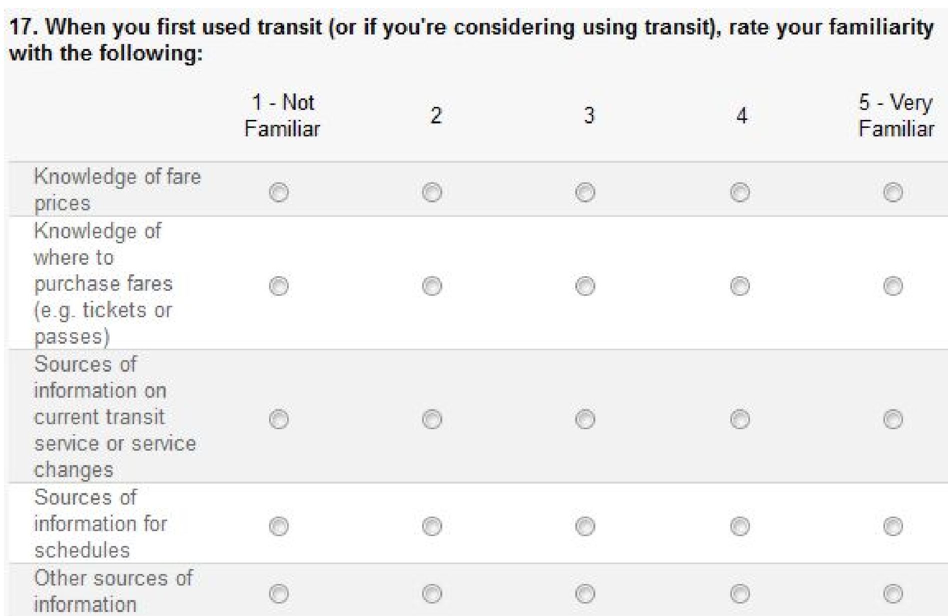
	Never	At least once a month	At least once a week	Once a day	More than once a day
Walk					
Bicycle					
Public Transit					
Car Passenger					
Car Driver					
Other					

Figure 9 - Participant Travel by Mode

Next, they are asked whether they have a driver's license and access to a car on a regular basis. The Transit experience of the participant is then queried with a series of questions relating to whether they have a transit pass, and their level of experience using the local transit system:

> ***With regard to the local transit network, how familiar are you with individual routes?***
> *The options are: Not familiar at all, My most commonly used route (i.e. to work or school), Most routes in my neighbourhood, or Many routes across the Region.*

Another a series of statements rate their familiarity with transit information, such as their knowledge of the local transit system, fares, and where to find route and schedule information (Figure 10).


17. When you first used transit (or if you're considering using transit), rate your familiarity with the following:

	1 - Not Familiar	2	3	4	5 - Very Familiar
Knowledge of fare prices					
Knowledge of where to purchase fares (e.g. tickets or passes)					
Sources of information on current transit service or service changes					
Sources of information for schedules					
Other sources of information					

Figure 10 - Participant familiarity with transit information

Upon completing the survey, the participant entered their e-mail address and received a completion acknowledgement. The confirmation message they received included a unique, sequential user identification number (User ID), which they used with the in-app portion of the WPTI Survey. The User ID ensures the subsequent anonymity of the participant's results.

The pre-survey results were downloaded from the Google Forms website in a Microsoft Excel compatible format. Although the questions in the pre-survey are qualitative in nature, the use of numerical responses, including time spans and selecting options on scales of 1 to 5, enabled quantitative analysis, which will be discussed in the Analysis section of this thesis.

After completing the pre-trip survey, obtaining their User ID via email, and installing the app on their smartphone, the participant was ready to engage in the survey while taking a trip on transit. They initiated the WPTI Survey app on their phone and entered their User ID. The survey app asked them for the purpose of their trip, how important it is for them to arrive on time, and whether or not they know the scheduled departure and arrival time of the bus. The app iteratively prompted them for their level of satisfaction of the trip up to that point, and their confidence in arriving at their destination on time. The aim of the latter question is to ascertain whether or not the participant believes they are adhering to the arrival time they stated at the beginning of the survey and how this factor may influence the level of satisfaction.

The next sub-section describes the methods of how the apps for the two software platforms were developed, followed by a step-by-step explanation of how the app works.

4.2.2 Stage 2: Smartphone Survey App

There are a variety of smartphone platforms and manufacturers currently on the market. Apple's iPhone and Google's Android compatible devices are the most widely used, so supporting both of these platforms was an easy decision. The newer generation of BlackBerry devices also support Android software natively. Apple's iPhone apps are compiled with the Objective C programming language XCode, which is unique to Apple. Android apps are compiled with the more popular Java programming language using a customized SDK (Software Development Kit) plug-in for the Eclipse Java programming interface.

The XCode app was developed using a MacBook Pro, as the iPhone app development could only be completed on an Apple computer. The iPhone app also took longer to compile than the Java app due to the author's unfamiliarity with that programming platform. The XCode interface is intuitive once you become familiar with it, but there is nonetheless a significant learning curve. Building the app took considerable trial and error before it was ready to be deployed to the student sample. The Android app development, which was also completed on the MacBook, went more quickly, as the coding for the iPhone version was followed as a guide for the programming logic and interface design. The flowchart for the development of the app is shown in Figure 11.

Once initalized, the participant enters their UserID assigned to them during the pre-survey. The app generates a unique TripID, which combines their UserID with the date and time (datestamp) of the initialization. The participant then completes the pre-trip survey, consisting of questions regarding the purpose of their trip, the time they need to be at their destination, and the anticipated pick-up and drop-off time of the bus. These data are validated to ensure the participant has entered appropriate times and information, such as ensuring that the destination time is not the same or earlier than the pick-up time. Once the participant is at the bus stop, they initiate the first satisfaction survey. Data validation ensures that the participant interacts with the app by pressing the satisfaction slider bar, even if they do not change its position from the default neutral (centre) position.

The behaviour of the survey changes depending on the stage of the trip. At the bus stop, the participant receives the full survey, including their perceptions of weather, comfort, and transit system performance. Once they are on the bus, the app only asks them basic survey questions, unless a change in satisfaction is noticed between one survey iteration and the next. In that case, the full survey is given to the participant, so try and identify triggers of satisfaction change. Once the participant has disembarked the bus, the end-of-survey questions are triggered by the participant. These app functions are further explained below.

Each screen of the app is a module in the programming language incorporating the variables, user interface, logic, and data output. The original algorithm, variables and source code for the apps are attached as an Appendix. The Android code includes comments and annotations to make it easier to follow the programming structure and logic. The author releases this original programming code under a Creative Commons Attribution 4.0 International License (see https://creativecommons.org/licenses/by/4.0/).

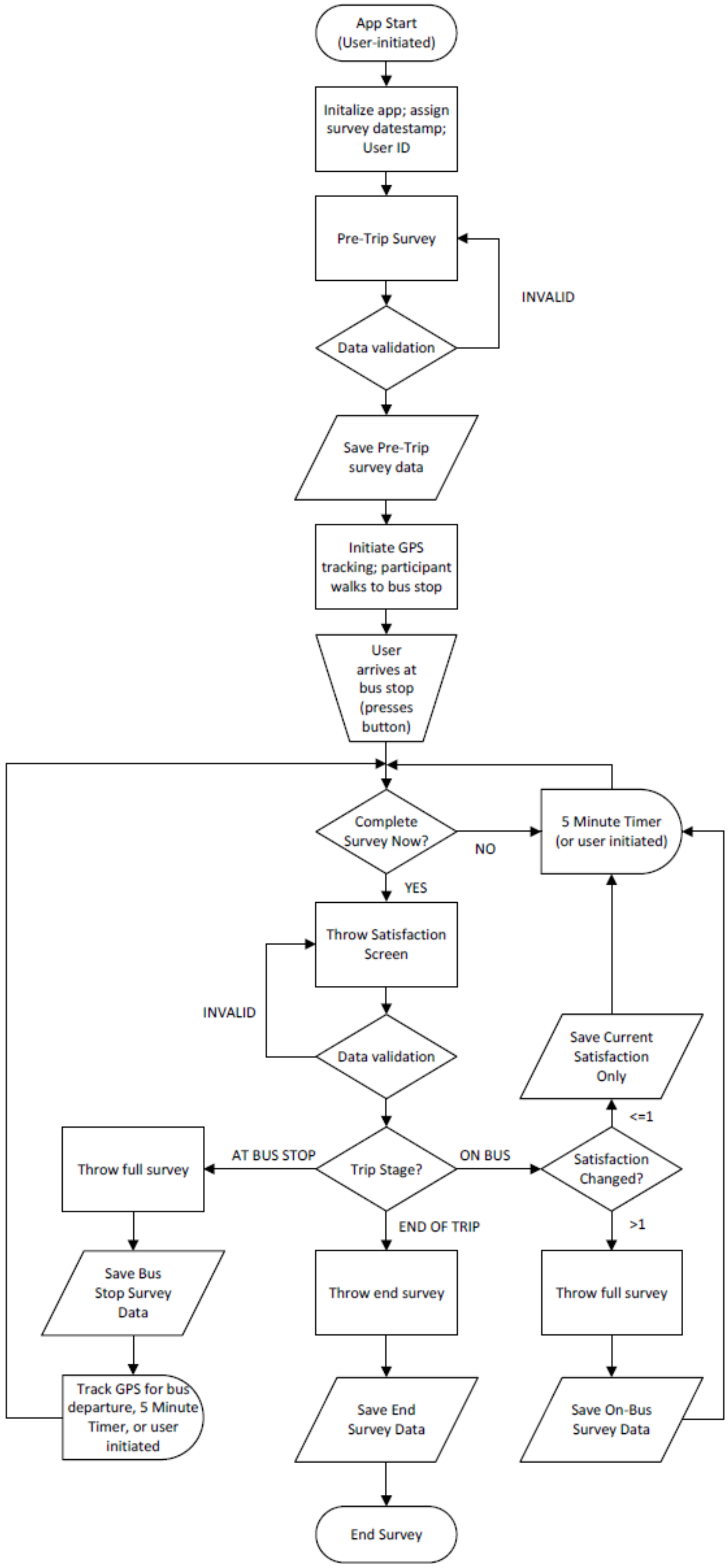


Figure 11 - App development flowchart

After testing the beta version of the app on both platforms, the apps were released and distributed through the official app "stores" for the respective devices: iTunes (iPhone), Google Play (Android) and BlackBerry World. Each of these stores required registration to enable distribution, which included providing descriptions and screen shots to appear in the front-end of the store interface. The alternative to using the stores would be manually installing the app on a participant's device by connecting it to a research computer via USB. Not only would that be a cumbersome process, it would also create privacy concerns because their personal device could be accessed. Another benefit of this additional step to distributing the software was, in the case of Apple and BlackBerry, the app went through an additional screening process required by those manufacturers to ensure the software run stably and error-free. Furthermore, using these official stores for distribution added an element of professionalism and credibility to the project, which is an important consideration for a future, expanded survey.

When the participant began their trip, they initiated the app on their smartphone by launching it via the app's icon in the phone's main menu. The first screen of the Survey asked for the User ID the participant was e-mailed after completing the online pre-survey. Examples of the Android and iPhone app screens are shown in Figure 12 and Figure 13. The two figures illustrate the differences in the appearance between the two versions of the app. For the remainder of the figures use the Android version of the screen shots.

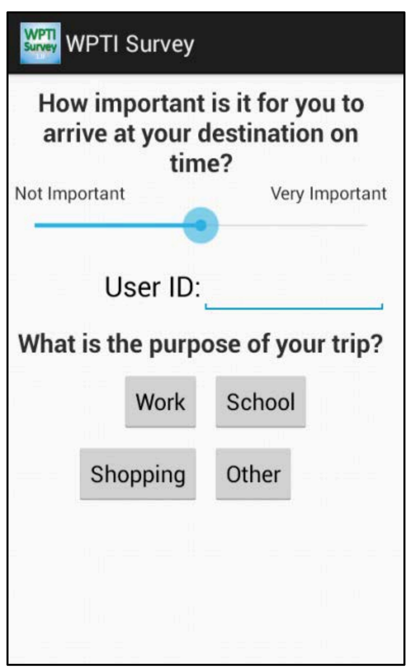


Figure 12 - First screen of the Android App

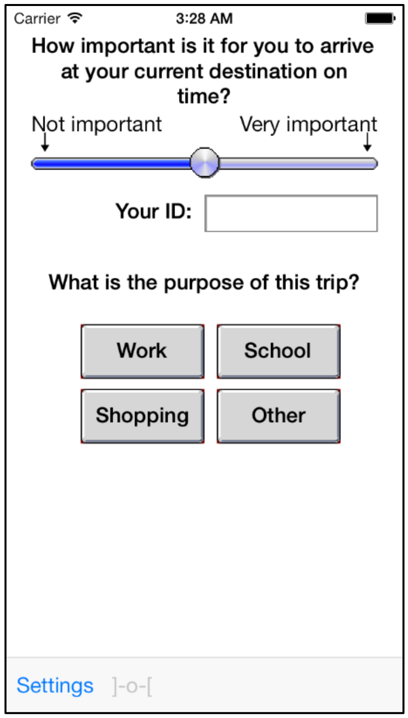


Figure 13 - First screen of the iPhone App

Some of the questions asked the user to answer by moving a slider bar on the app's interface with their finger. The slider bars record input on a Likert scale of 1 to 9, where 1 is the left-most position and 9 is the right-most position. The middle, neutral position equals 5. The first question "How important is it for you to arrive at your destination on time" asks the participant to specify if it is very important or not important at all by repositioning the slider bar selector. For example, if they were travelling to an appointment and did not want to be late, they would likely rate the importance highly, whereas if they are going home and did not have to be there at a specific time, they would likely indicate a lower importance. The middle position represents that the user feels "Neutral" about the qualitative question. Sliding to the right or left indicates a progressively more positive or negative response.

The response to this first question, in conjunction with the questions relating to how the participant perceives the importance of time and the purpose of their trip, and the pre-trip survey, provided context to the trip they are taking and their time sensitivity, which can be compared and contrasted with the results of other participants. After selecting a trip purpose, the screen will advance. There were checks built into the user interface so that if the participant does not enter their assigned User ID, or set the importance scale, they cannot proceed to the next screen.

The next screen in the survey (Figure 14) asks a series of 3 questions asking the participant to select a time for:

1) What time do you expect the bus to pick you up?
2) At what time do you need to be at your destination?
3) What time is the bus scheduled to arrive at the bus stop closest to your destination?

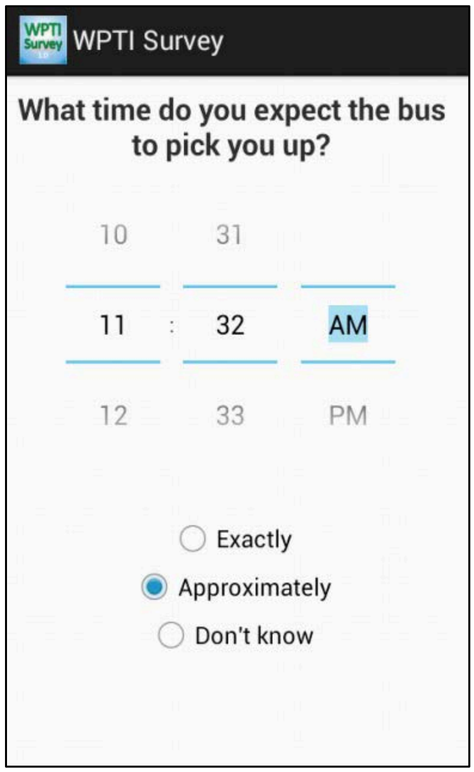


Figure 14 - Android screen for trip time selections.

For each question, the participant is asked to select "Exactly", "Approximately", or "Don't know" to indicate your level of confidence in the time they selected. For example, if the participant checked the bus schedule beforehand, they might indicate the exact time. The time the participant needed to be at their destination could be the start time of class or an appointment.

After answering these 3 time questions, the time selector disappears and the participant is asked to continue the survey once they have reached the bus stop. When they are at the bus stop, they press the "Continue" button. GPS tracking is then enabled, and the phone's GPS indicator will flash as it obtains a satellite fix. The in-trip portion of the survey is now active.

4.2.3 In-Trip (On bus) Survey

Each segment of the in-trip survey begins by asking the participant about their current level of satisfaction with their transit trip experience (Figure 15).

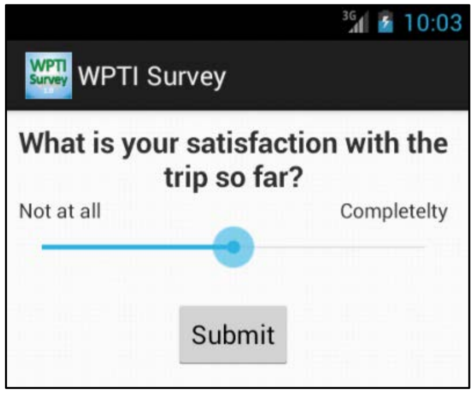


Figure 15 – What is your satisfaction with the trip so far?

The participant's satisfaction is subjective. For example, satisfaction could be based on their perception of condition of the bus stop area, or their surrounding environment, and how they are generally feeling about their trip at that point. As the trip progresses, the on-time performance of the transit system will also play a factor in their satisfaction. At each survey interval, the participant is also prompted about whether or not they think they will arrive at their destination on time and how confident they in that expectation (Figure 16).

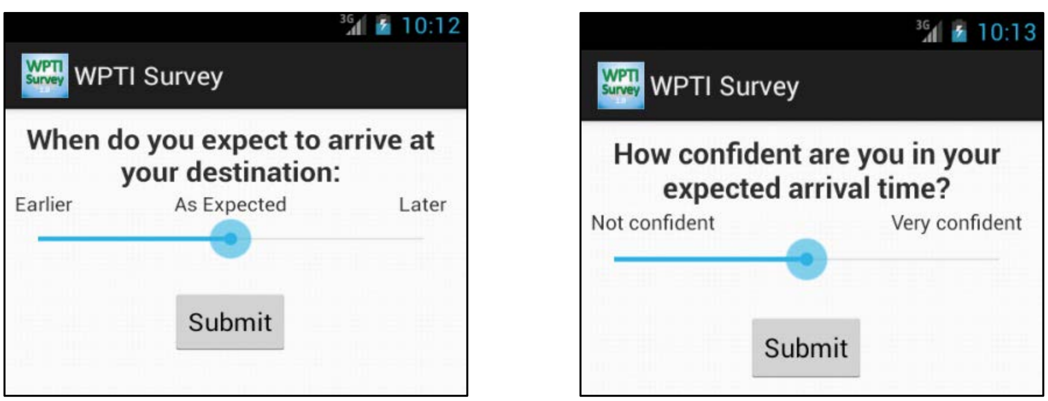


Figure 16 – These screens ask if the participant expects to arrive on time, and how confident they are in that expectation.

Participants will see a message box on the screen (Figure 17) and hear a notification reminder to repeat the survey when the bus leaves the bus stop, and every 5 minutes (300 seconds) thereafter while they are in-trip. The app uses GPS to detect when the participant has boarded the bus and left the bus stop by monitoring for a change in GPS coordinates of greater than 30 metres from the location where the first in-trip survey iteration was made.

If it is not convenient for the participant to complete subsequent survey iterations at the time, they may select “No”, and be reminded again in 2.5 minutes (150 seconds). If the participant is on a long journey and doesn’t want to complete the survey as often, they can press the Settings button (iPhone) or Settings Menu (Android/BB) and temporarily enter a new survey interval for their trip.

Figure 17 – The “nag” screen, asking the participant to complete the survey after the time interval as lapsed.

4.2.4 Supplemental In-Trip Questions

When the participant is aboard the bus, a series of supplemental questions will appear (Figure 18) asking them about the factors currently influencing their level of satisfaction. If the survey app senses a significant change in their level of satisfaction from one survey to the next (more than one slider bar position), and at the end of the trip, the supplemental questions will be repeated. The participant can also enter additional information by text or voice recording to describe their current situation and environment. Entering a message is optional.

4.2.5 End-of-Trip Questions

Once the participant has reached their destination, they can press the "End of Trip" button, which appears at the bottom of the user interface. In addition to the satisfaction and supplemental questions, they will be asked if they think they actually arrived on-time, early, or late at their destination, how long they think the trip was, and how long they think they waited for the bus (Figure 19). The participant is not expected to know the exact times or length of the trip, as we are ascertaining their perception of time for the trip, which can then be correlated to their time sensitivity and pre-trip survey submission.

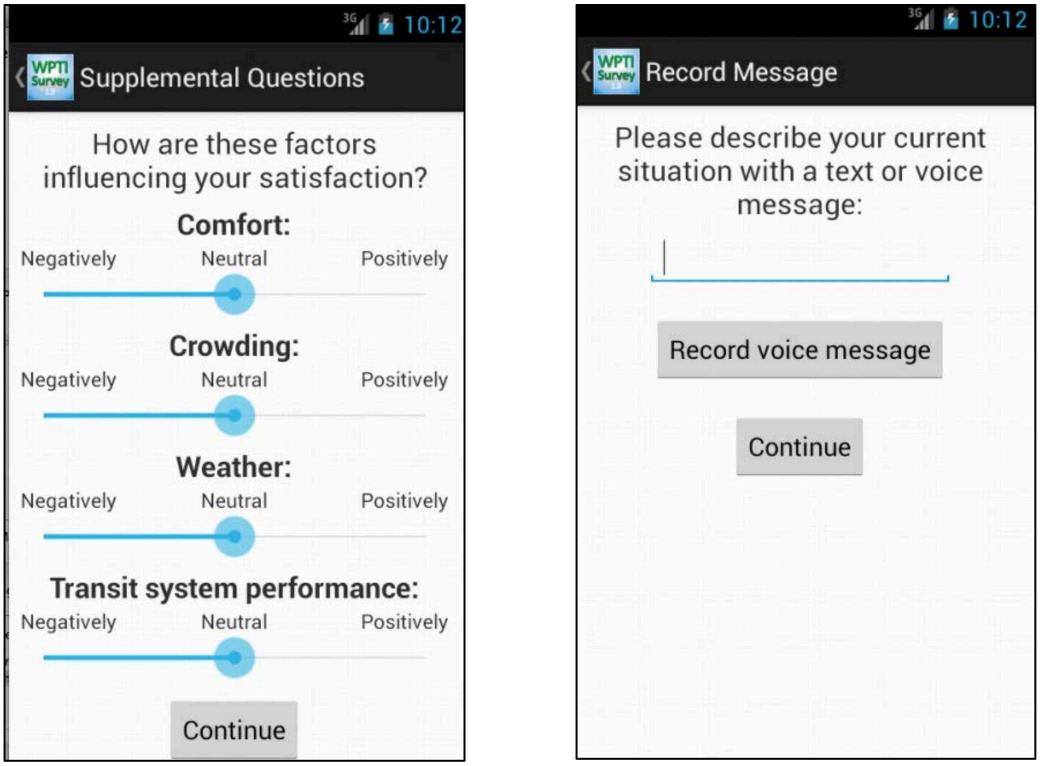


Figure 18 – Supplemental questions screens

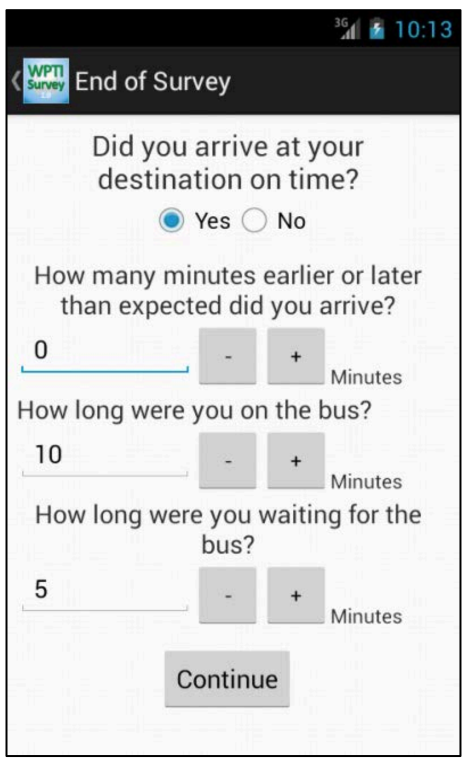


Figure 19 – End of trip questions, assesses the participant's perceptions about arriving on time and how long the trip took.

After completion of the survey, the app goes into standby mode. When the participant is ready to begin a return trip or a new trip, they can press the Start button or re-launch the app from the phone's main app menu.

4.2.6 Data Submission

Survey data are not sent automatically to the researchers. To submit their data, once the participant has completed all of their trips on transit, there is an Upload function in the app. The participant simply opens the Settings menu in the app, and presses the Upload button. The app composed an e-mail message, with the data files as attachments. Pressing the send button on the message sends the data to the research e-mail address. The data files are small, and the total upload is less than 1Mb. To avoid data charges on their phones, participants are encouraged to send their data when connected to Wi-Fi.

No further interaction between the researchers and participants is required. The participants can uninstall the app from their phone after submitting their data.

5 WPTI Survey Results

The WPTI Survey received Office of Research and Ethics approval and was conducted with 151 University of Waterloo undergraduate engineering and planning students during the Winter and Spring terms in 2014. The students initiated 688 trips, of which 457 were validated for analysis. The validation process screened out incomplete trips and data inconsistencies resulting from invalid participant input, such as departure and arrival times being the same, or trips less than 3 minutes in duration, which indicated the participant was not taking a real trip. The students can be considered captive because participating in the survey was part of a course assignment. Students not wishing to participate, or who did not have a compatible smartphone, were able to complete an alternative assignment. A summary of the participation in the survey is in Table 4.

Table 4 - Summary of WPTI Survey participation

Date	Spring-Summer 2014
Participants	151
Male / Female	91 / 60
Trips	688
Valid Trips	*457*
Trips with GPS	*141 (31%)*
Mean Trip Duration	0:19
Surveys Completed*	3,548
Voice Responses	*325*
Text Responses	*1,201*
Trip Type	At participant discretion

**Surveys completed includes all iterations for all trips.*

The students were asked to take at least 3 trips each on transit. No constraints were imposed for this pilot survey, such as time limits, specific destinations or other conditions. The students were encouraged to conduct their daily routine travel by transit. A presentation was made in-class to each participating group, which included information about the previous TOES study and instructions for installing and using the app.

As students participated, each iteration of the survey they completed was saved on the smartphone as a comma-separated text file. Files are uniquely named with a trip identification code consisting of the User ID, date/timestamp and iteration number. The volume of individual files, which numbered in the thousands, was problematic for processing. The solution was the development of a Visual Basic utility, ***WPTI Text Processor***, to parse and categorize all of the files by iteration (see Appendix). This utility created composite files that could be opened in Excel for analysis. However, the most cumbersome parsing task was transcribing all of the recorded voice responses into plain text, which meant listening to each recording and typing it in manually.

Although the actual results were not as conclusive has hoped, this initial survey was successful in confirming proof of concept for a broader study. A summary and analysis of the results is provided in the next section. It is hoped that a future survey will poll a broader sample of the population, which is expected to provide more conclusive results on the perception of time, rider satisfaction, and informing more accurate transportation modelling parameters. Suggestions for the future survey are provided in the conclusion chapter.

5.1 Analysis

Survey data were compiled into a single, multi-worksheet file in Excel. The Pre-Survey results were also included, to enable cross-referencing with the in-app results.

5.1.1 Pre-Trip Survey

Indices were added to the Pre-Survey data to combine the various aspects of time sensitivity and rider experience profiles. The new variables FamiliarityIndex, ArrSensitivityIndex, TransitRiderIndex, and ConfidenceIndex were created. The Familiarity Index assessed the participant's familiarity with the transit system, based on their response to the question about understanding the system's fare structure and how to access route and schedule information. The ArrSensitivityIndex is the participant's sensitivity to arriving on time, and what it means to arrive on time. The TransitRiderIndex combined that result with how frequently the participant chose transit over other modes of transportation. The ConfidenceIndex is a combination of all of these factors. These variables were normalized and associated with the participant's User ID for analysis with the survey app results that measure their trip.

5.1.2 In-Trip Survey

The raw data for participant trip satisfaction are measured in a Likert scale of 1 to 9, based on the user's selection on the slider bar in the user interface. A value of 1 means "Not at all satisfied", 5 is neutral (the middle position on the slider bar), and a value of 9 means "Completely satisfied" with their trip up to that point.

It was evident from the initial review of the results that people perceive satisfaction differently, and the term is subjective. Some participants defaulted to "neutral" as a starting point, while others consistently chose a higher satisfaction level, or even “completely satisfied” unless something went awry. Since the participants appeared to have different baseline standard for how they assess personal satisfaction, results were normalized for certain analyses of trip characteristics, pre-survey attributes, and satisfaction. On the Likert scale, the mean satisfaction on the scale of 1 to 9 was 5.7.

Through normalization, the mean value of the sample becomes 1. The mean satisfaction of each participant, or individual trip, can then be compared to this baseline. This mean of the means approach attempts to compensate for the observed problem of how individuals baseline their satisfaction differently, whereby one participant consistently provides values of 8 or 9 on the scale in a normal situation, while another provides a neutral response, 5 or 6 on the scale in a similar situation. The mean results for each participant are normalized to a value of 1. Once normalized, higher or lower levels of satisfaction can be observed as >1 or <1, making it easier to compare individual responses.

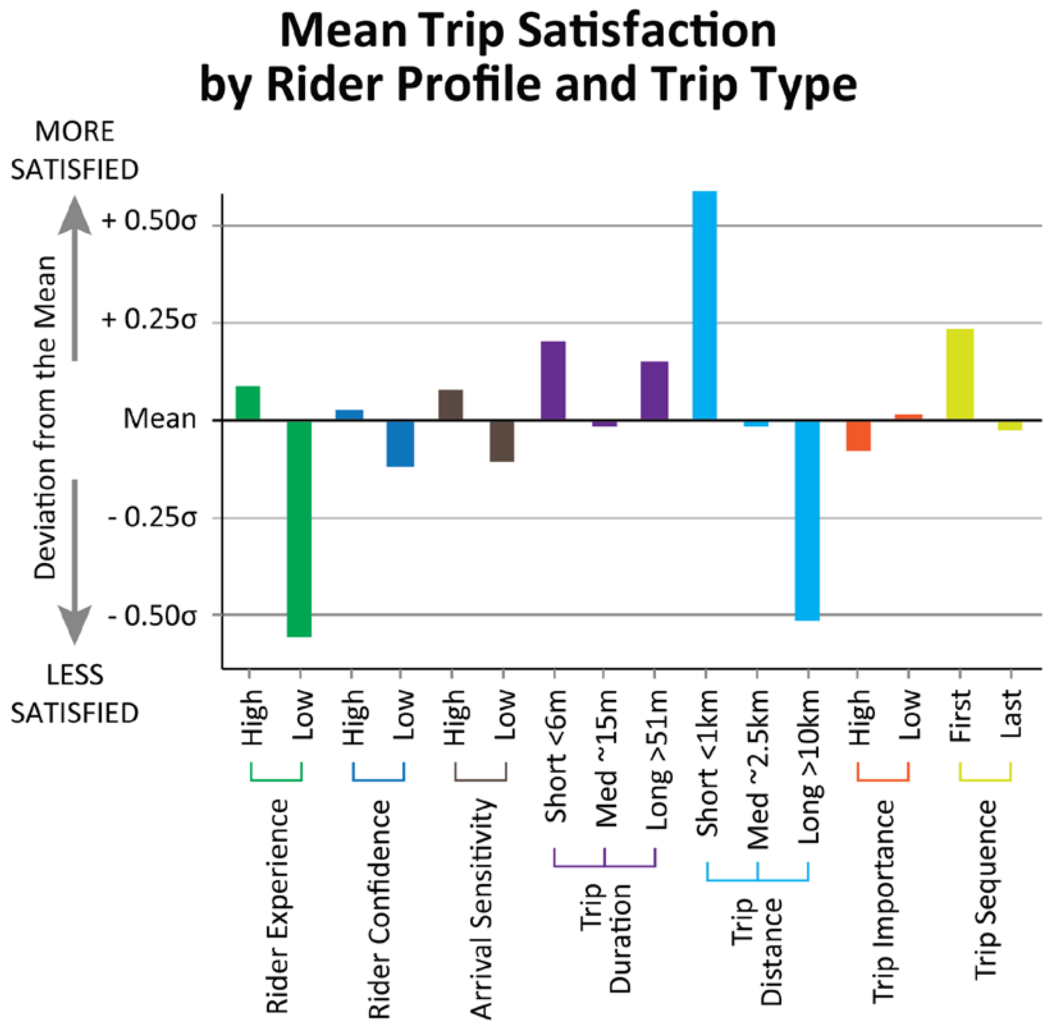


Figure 20 - Satisfaction by rider experience, trip duration, importance, and sequence.

Figure 20 shows the relative satisfaction of participants with high or low levels of pre-trip survey indices for time sensitivity, along with other trip characteristics, such as whether it was the participant's first trip, the trip length and duration, and their stated importance of the trip. The y axis is the standard deviation from the mean ($\mu = 1$, $\sigma = 0.285$). As anticipated, there are positive and negative deviations for more or less experienced or confident transit users respectively. These results are for the top and bottom 5% of the participants.

Trip length in distance, also appears to influence satisfaction, with short trips being more bearable than long ones. However, trip length in duration did not yield a conclusive result. Other attributes, such as arrival time sensitivity, trip importance, duration and the sequence of the trips taken, are slightly positive (a value >1), but less conclusive in that there little deviation from the mean satisfaction of all observations.

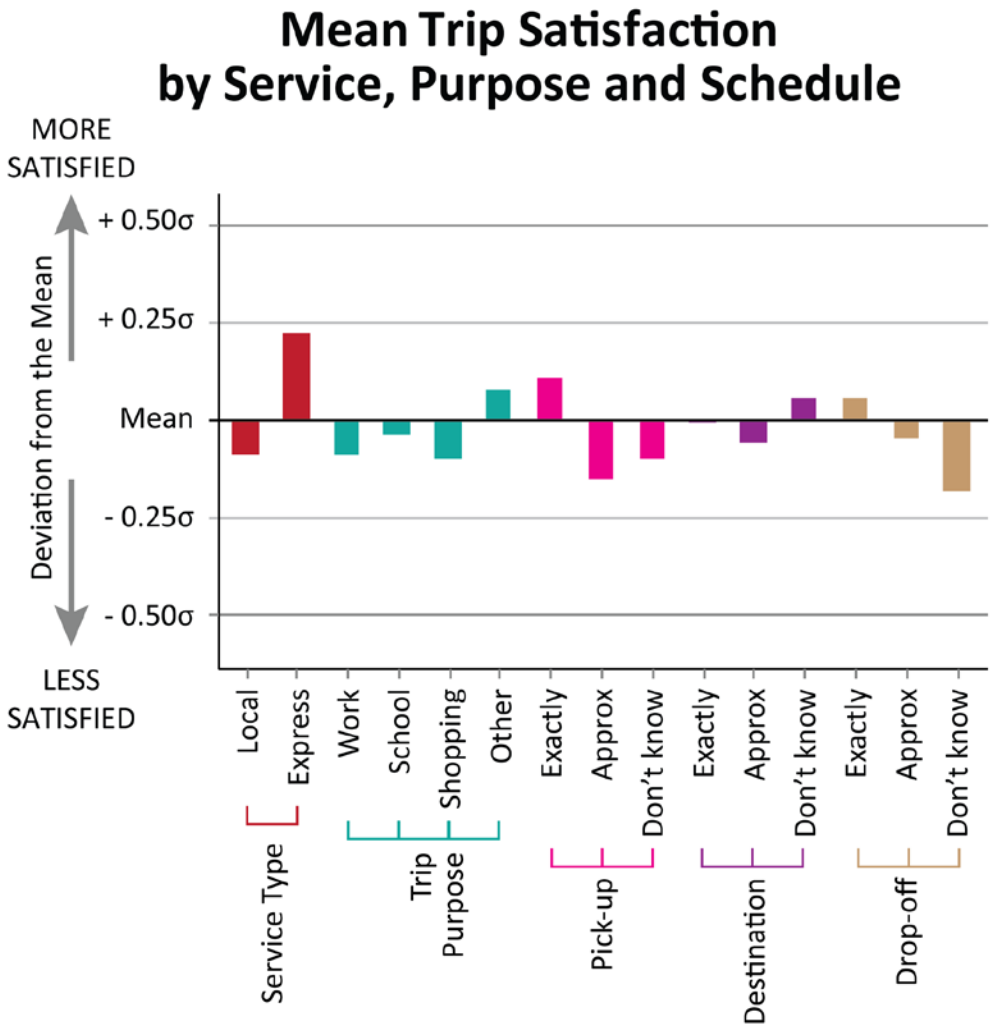


Figure 21 - Satisfaction by type of service, trip purpose and scheduled times.

Figure 21 presents mean satisfaction based on type of service, the participant's responses to the expected pick-up time, the time they need to be at their destination, and the expected time the bus will drop them off near their destination. Express bus routes produced a higher level of satisfaction than local service routes. When prompted for pick-up, destination and drop-off time information in the app, they are asked to specify whether they know the arrival time is exact (they have checked the schedule or seen a real-time display), is approximate (they have an idea of what the schedule or frequency of service is), or they don't know and have simply arrived at the bus stop to wait for the next bus.

There is little variation to mean satisfaction across these variables, and again regression testing did not find these relationships significant. However, the minor changes do indicate the trends that are anticipated, such as participants being less satisfied with local versus express service, and on-time versus not-on-time performance. An interesting observation is the lower satisfaction of participants who don't know the bus arrival time while waiting at the bus stop, confirming that the lack of such information negatively influences their perceived satisfaction with the transit system.

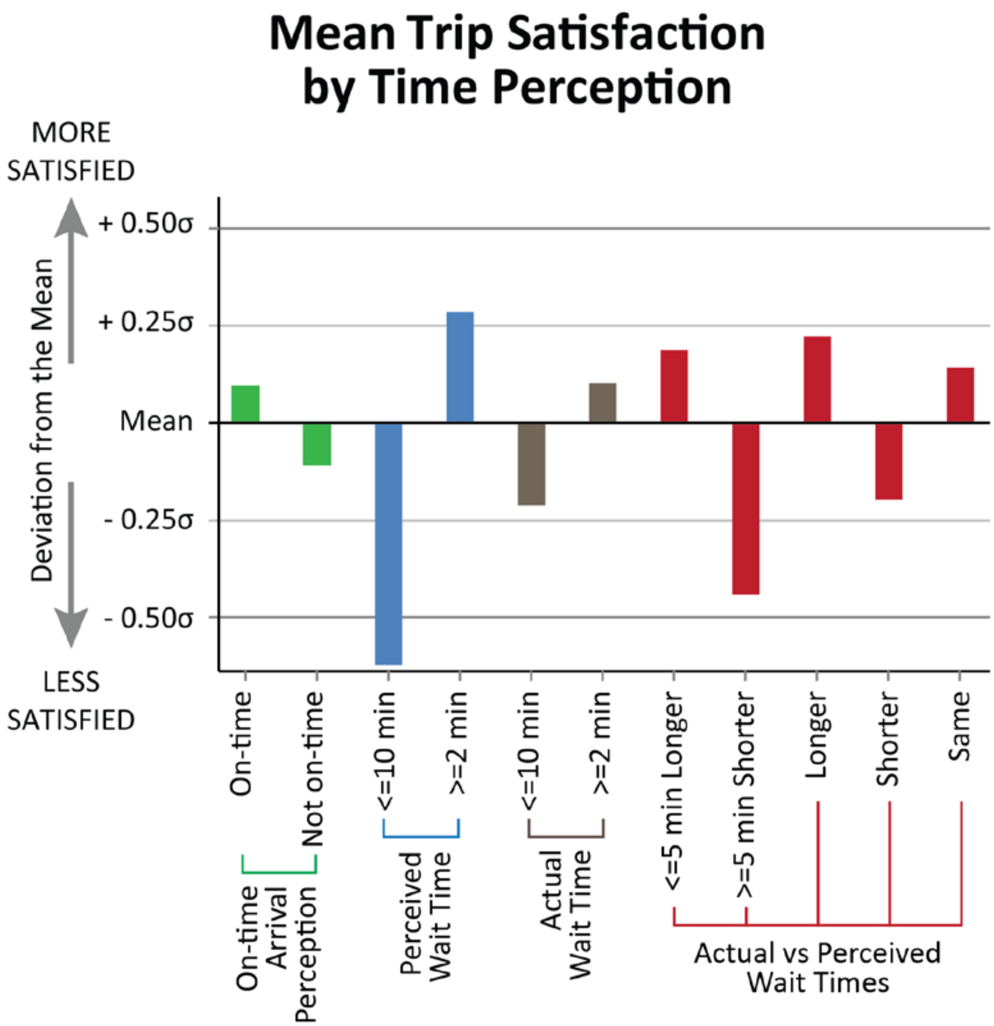


Figure 22 - Satisfaction compared to the participant's perception of on-time performance and waiting time at the bus stop.

Figure 22 compares the satisfaction responses between participants who thought the bus was on time or not on time. The first pairing is the perception of on-time arrival, from the app question "Do you think you will arrive at your destination on time?" As expected, those who perceived the bus as being on-time reported higher satisfaction with their trip experience than those who felt that the bus was not on-time, but the variation from the mean is subtle. Greater deviations were found for the perceived a wait time at the bus stop. Participants who thought they waited 10 or more minutes reported significantly lower satisfaction than those who thought they waited for 2 minutes or less.

Actual wait time at the bus stop was determined by using the difference between the time that the participant initiated the bus stop iteration of the survey and the first on-bus iteration of the survey. The accuracy and reliability of this calculated wait time depends on whether or not participants followed the guidance to initiate the app upon arriving at the bus stop and as soon as possible after boarding the bus. The average actual wait time at the bus stop was 4.7 minutes. Participants over estimated this wait time by 11.0% (5.3 minutes). The average actual time on-bus was 10.7 minutes. Participants over estimated this wait time by 7.2% (12.1 minutes). As shown in Figure 22 satisfaction levels for actual wait times of 10 minutes or more were not as low as the same perceived time.

The next columns in Figure 22 directly compare actual and perceived wait times. If the actual wait time was longer than the perceived wait time, satisfaction was higher than if the actual wait time was shorter than the perceived wait time. If the participant accurately reported the same wait time as actually observed, satisfaction with the trip was generally positive.

5.1.3 Triggers of Dissatisfaction and Anxiety

Participants were generally pleased with the performance of the transit system, and were confident of arriving on time at their destination. The app asked them at every survey iteration if they expected to arrive on-time, and how confident they were about that. Figure 23 shows that their perception of arriving on time was usually neutral at the beginning of the trip, and there was some change once they were on the bus and perhaps able to better evaluate their progress. Their confidence in arriving on time definitely improved and gradually became more confident once they were on the bus as the trip progressed, as shown in Figure 24.

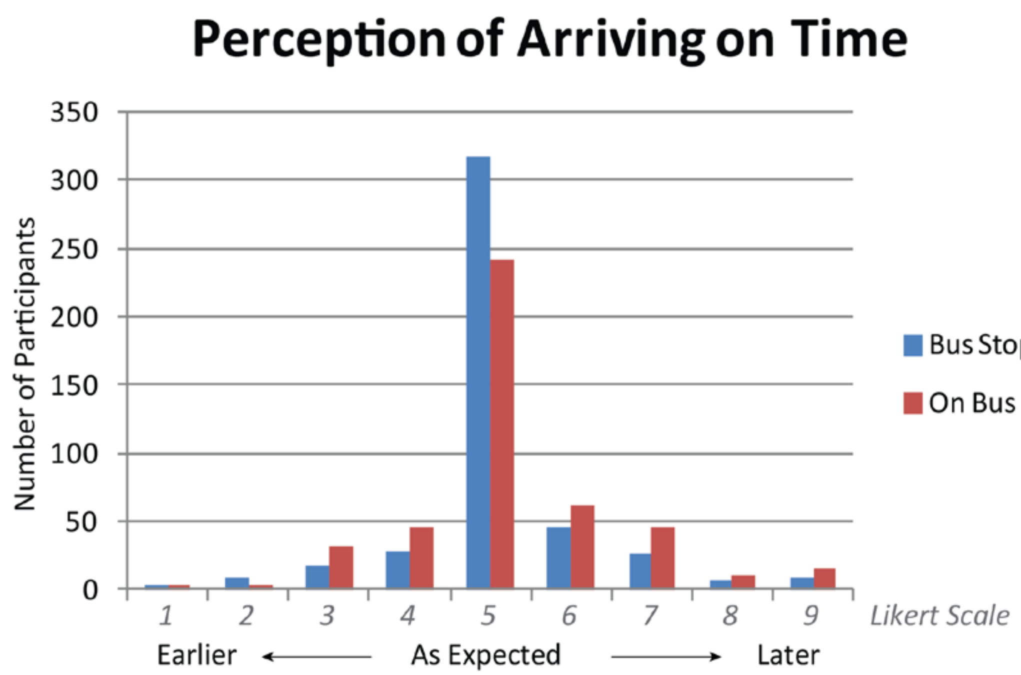


Figure 23 - The perception of whether or not the participant will arrive at their destination on time, observed at the bus stop and while on the bus.

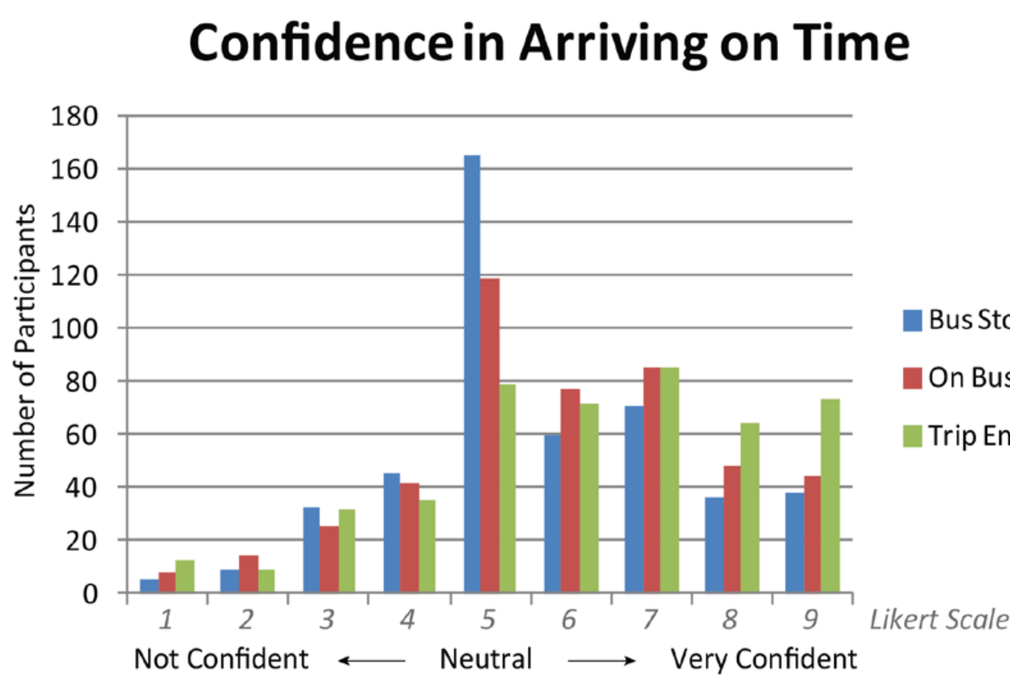


Figure 24 - The confidence of the participant in arriving on time.

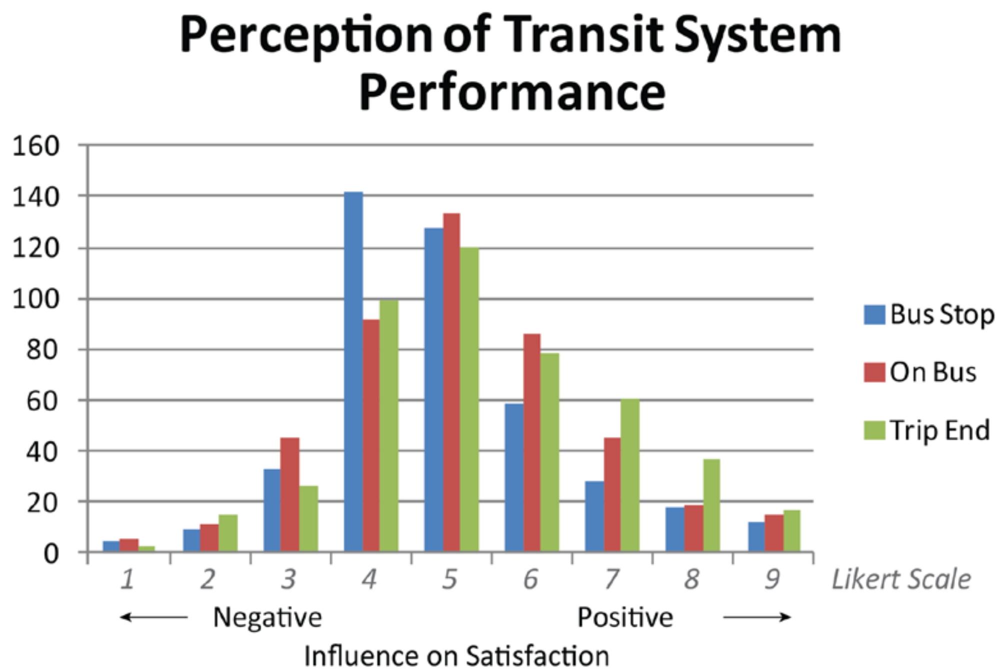


Figure 25 – The participant's perception of the overall performance of the transit system and how it is influencing their satisfaction.

Figure 25 also shows a generally positive perception of the transit system's performance improving over the three stages of the trip. However, at the bus stop there is a slight, yet markedly, negative perception to begin with.

The detailed in-trip survey asks the participant to rate how perceptions of their environment and surroundings are positively or negatively influencing their satisfaction during the trip. Direct measurement of rider satisfaction produced inconclusive results. Research conducted by Carrel et al. (2016) also encountered problems in measuring satisfaction, because satisfaction is very subjective. One individual may declare themselves completely satisfied (8 or 9 on the Likert scale) when things are going normally, while others may select neutral by default (5 on the Likert scale). As Carrel also concludes, there needs to be a baseline assessment of satisfaction from which to measure change. In the earlier TOES survey, an impression of satisfaction could be implied from the collection of individual metrics of comfort and wellbeing that were recorded, instead of asking about comfort and wellbeing directly (Figure 3).

To compensate for individual satisfaction bias in the WPTI survey, the mean of each participant was taken so that variances from that baseline could be compared between individuals. However, mean observed satisfaction during the trip still remained relatively flat. The difference between bus stop, on bus, and trip end mean satisfaction was within 0.03 of the mean index value of 1, with a standard deviation of 0.28. The satisfaction observations were disappointing. It appears that, in addition to having different baseline measures of satisfaction, people continually reassess their satisfaction as their individual experience and circumstances change. But, there are some interesting results based on perceptual and experiential influences, as explained below.

5.1.4 Influence of Experiential Perception on Satisfaction

The next series of charts illustrates the effect of different influencing factors on the participants' satisfaction during the trip. These charts are formatted to show the frequency of pairings between the stated level of satisfaction and the significance of how their experience is having a positive or negative influence. The size of the blue bubbles indicates the relative number of responses for the attribute pair. A trend line is included if regression analysis indicates a significant correlation between the variables, under the assumption that satisfaction is the dependent variable being influenced by the participants' experience.

The first chart, Figure 26, pairs satisfaction with the perception of the overall performance of the transit system during the trip. There is a trend towards a positive perception of system performance leading to higher satisfaction. Although not surprising, the result demonstrates the capability of the app to capture this relationship. The majority of responses tend to be in the middle range of satisfaction and performance, as indicated by the size of the bubbles. The chart also illustrates the tendency of some respondents to indicate complete satisfaction as their default response, as shown by the size of the bubbles corresponding to the value of 9 (completely satisfied), although the majority appear to choose a more neutral response.

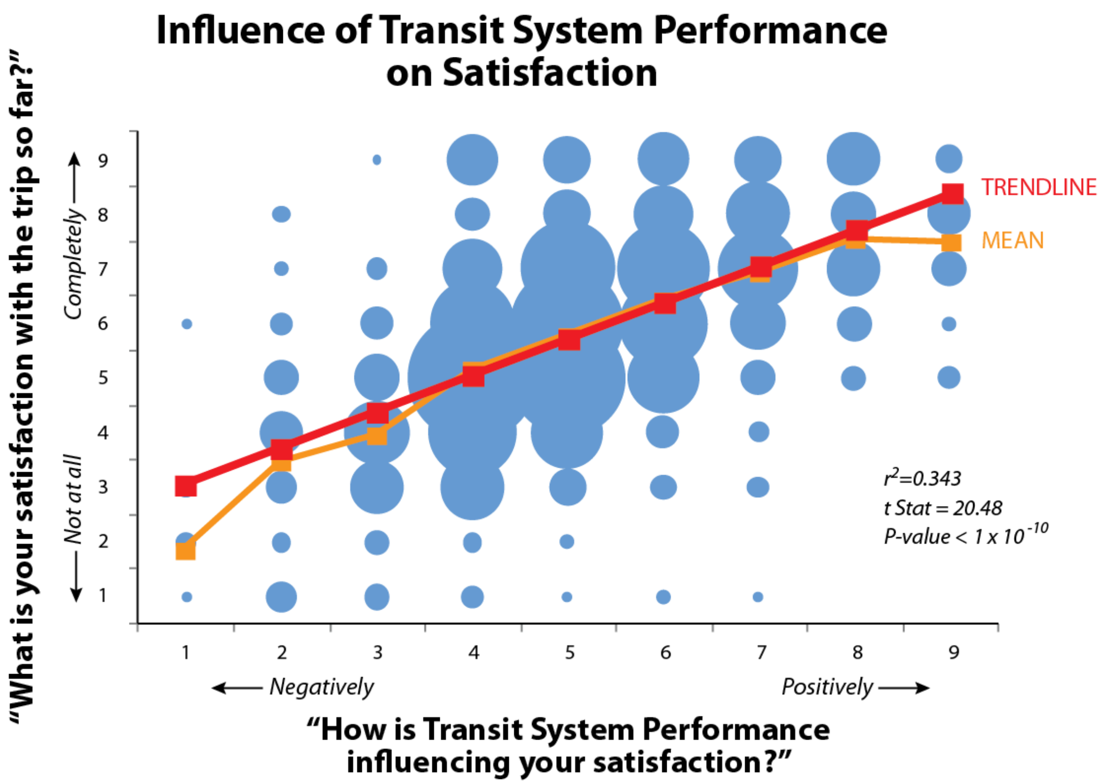


Figure 26 – Influence of perceived transit system performance on satisfaction

The earlier TOES research showed a relationship between the perceptions of comfort and wellbeing and the stages of the trip (i.e. before, at the bus stop, on the bus, and after departing the bus). The WPTI Survey results were less clear, but do indicate a trend. Figure 27 shows comfort during the three main stages of the trip. There is a slight skew (-0.09) towards greater comfort at the end of the trip. The wellbeing result is less conclusive. The bubble frequencies of both comfort and wellbeing are overlaid with each other in Figure 28 to illustrate how closely the two attributes align. The positive influence of comfort and wellbeing on satisfaction is almost identical. Perhaps participants could not differentiate between these two concepts, or they are too similar and one should be omitted from future studies.

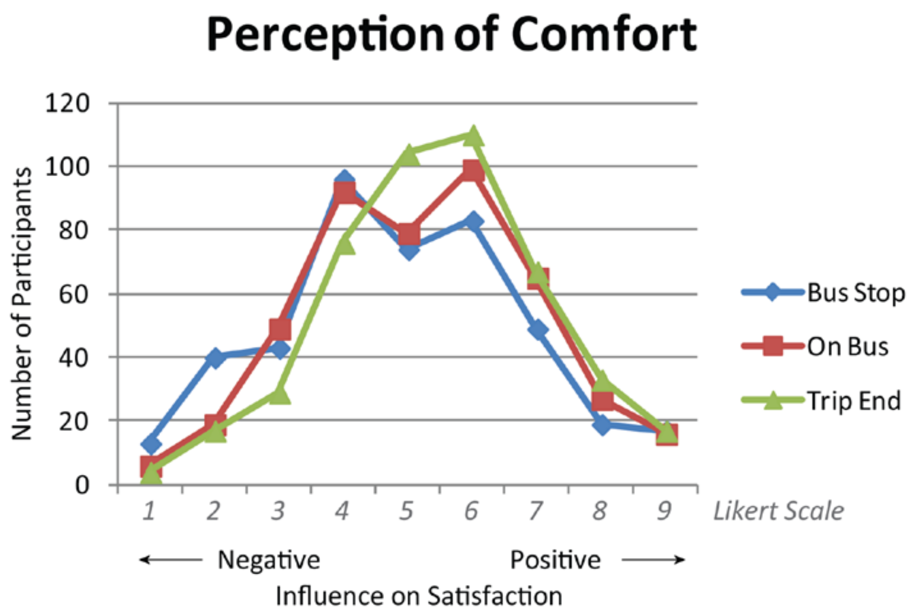


Figure 27 – Changes in the perception of comfort during the three stages of the trip and how it is influencing their satisfaction.

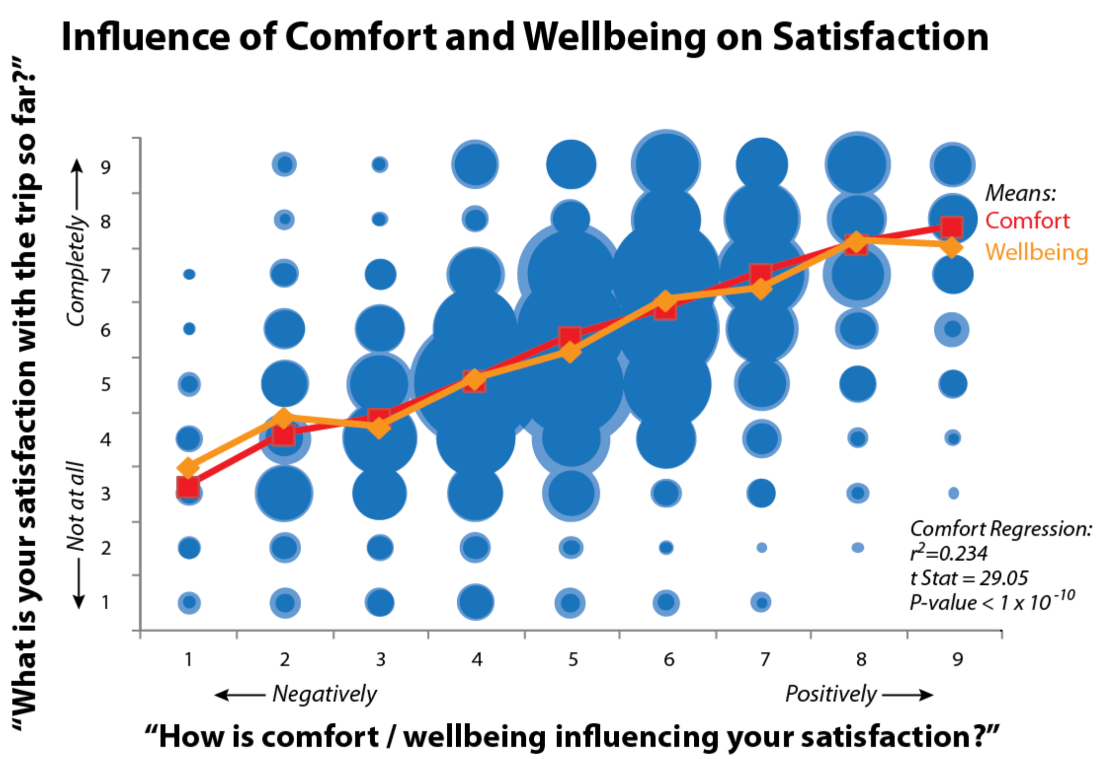


Figure 28 – Influence of perceived comfort and wellbeing on satisfaction

Another possible explanation for the less conclusive results of the WPTI Survey versus the earlier study is that, in TOES, comfort and wellbeing were each generalized from five separate attribute pairs. Thereby, the earlier results revealed the participants' comfort and wellbeing, instead of asking directly.

The perception of schedule adherence and overall transit system performance was asked as a follow up question to satisfaction each time the participant completed an iteration of the survey. Figure 29 charts the perception of whether or not the rider expects to arrive at their destination on time, earlier, or later. The majority of responses indicate that an on-time arrival is expected. A slightly earlier or later arrival expectation has a positive or negative influence on satisfaction. The lowest levels of satisfaction are clear when the participant becomes certain of a late arrival, as indicated in the lower right of the chart.

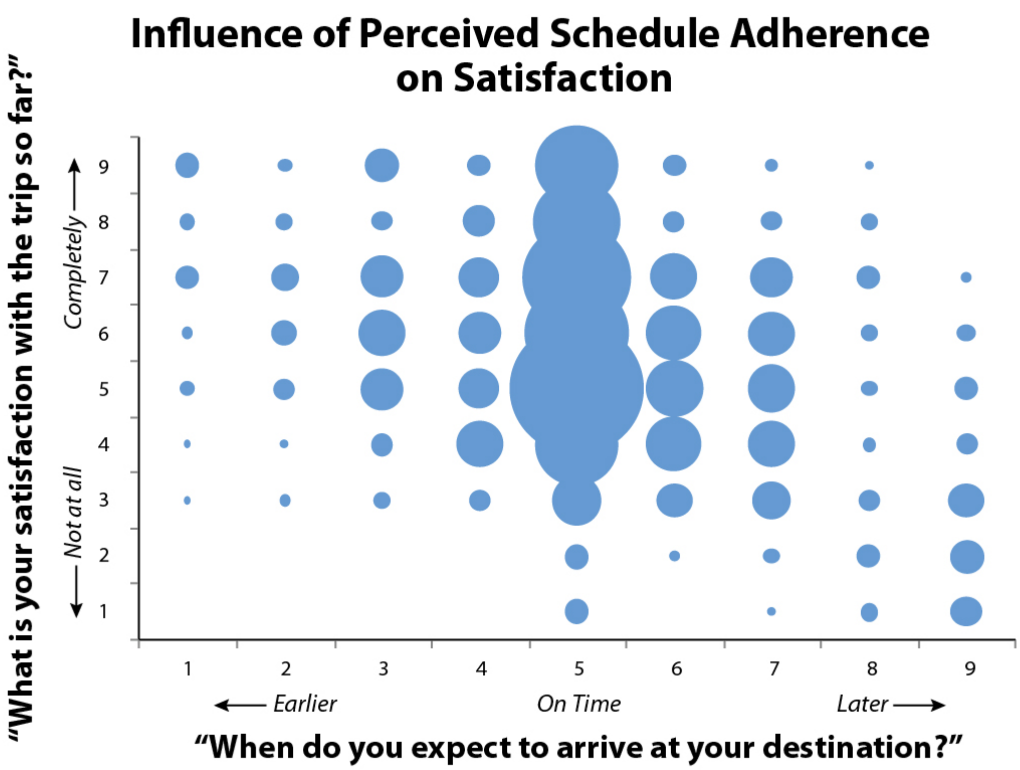


Figure 29 – Influence of perceived schedule adherence on satisfaction

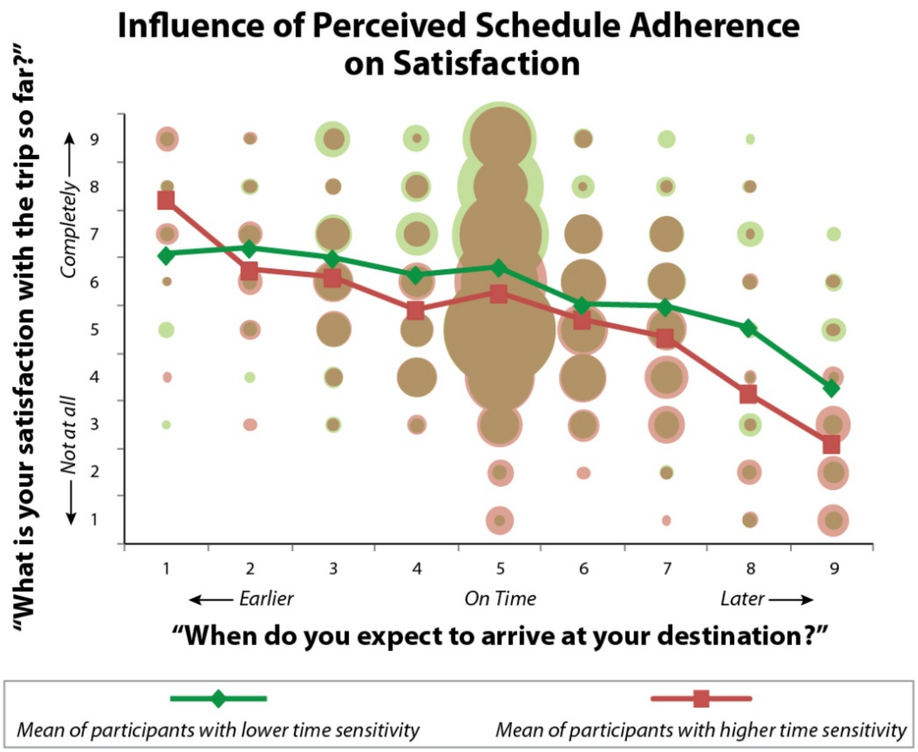


Figure 30 – Influence of perceived schedule adherence on satisfaction comparing participants with lower or higher levels of time sensitivity.

Figure 30 indicates that participants with a higher sensitivity to time, based on their Pre Survey responses, had lower levels of satisfaction if they perceived they would arrive at their destination later than expected.

The performance of the transit system influences satisfaction by affecting the overall positive or negative experience of the trip. Aside from schedule adherence, performance may relate to other perceptions such as condition or cleanliness of the bus or bus stop area, characteristics of the ride, interaction with other passengers, or availability of a seat. The relationship between satisfaction and performance is shown in Figure 26. A statistical correlation was found between satisfaction and system performance. The regression trend line and parameters are included in the chart.

Everybody likes to complain about the weather, and participants in the WPTI Survey made it be known when the weather was not amenable. However, Figure 31 shows that the effect of weather on rider satisfaction is inconclusive. Although data from the bus stop segment of the trip indicates weather is an influence on positive or negative experience, riders seem to be fairly forgiving about it. No one can control the weather, so riders are not rushing to blame transit for a miserable day. The data are well distributed, and regression analysis did not reveal a significant trend.

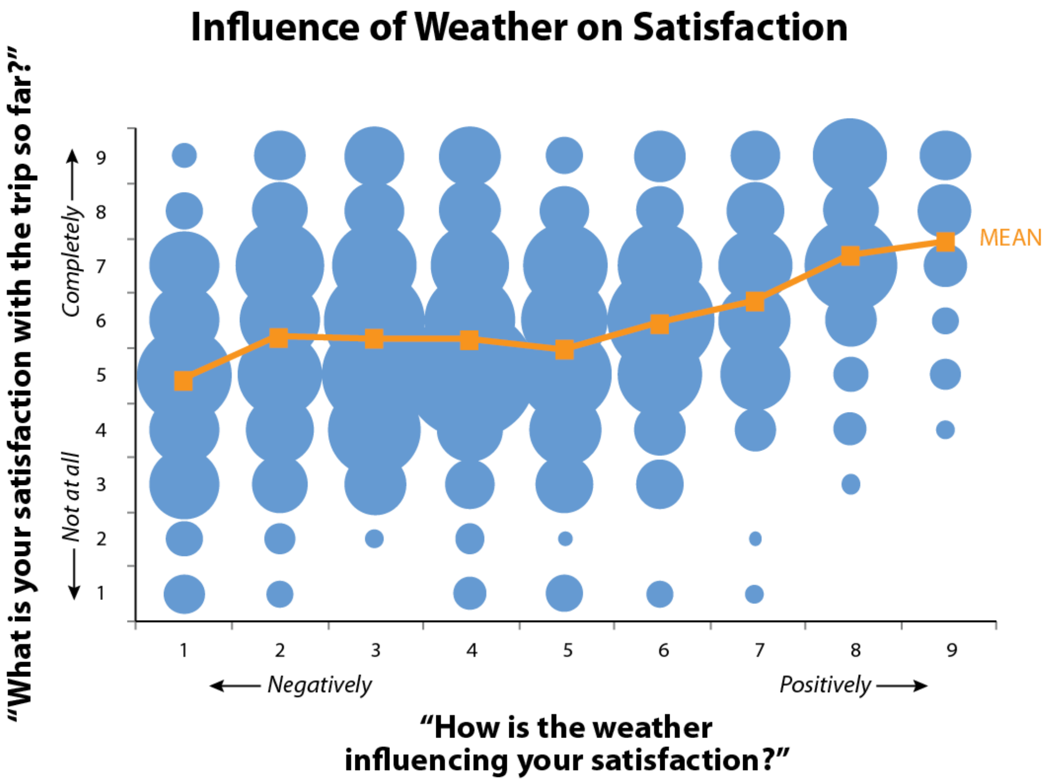


Figure 31 – Influence of weather on satisfaction

The last chart in this series, Figure 32, brings together the mean of the individual attributes, showing that the combination of factors, rather than just one aspect of the rider experience, influences rider satisfaction. Regression analysis on the combined result, and on transit system performance (Figure 26), indicates that there is a good potential for predicting rider satisfaction. An increase or decrease in the perception of how well the transit system is performing will have a corresponding influence on satisfaction.

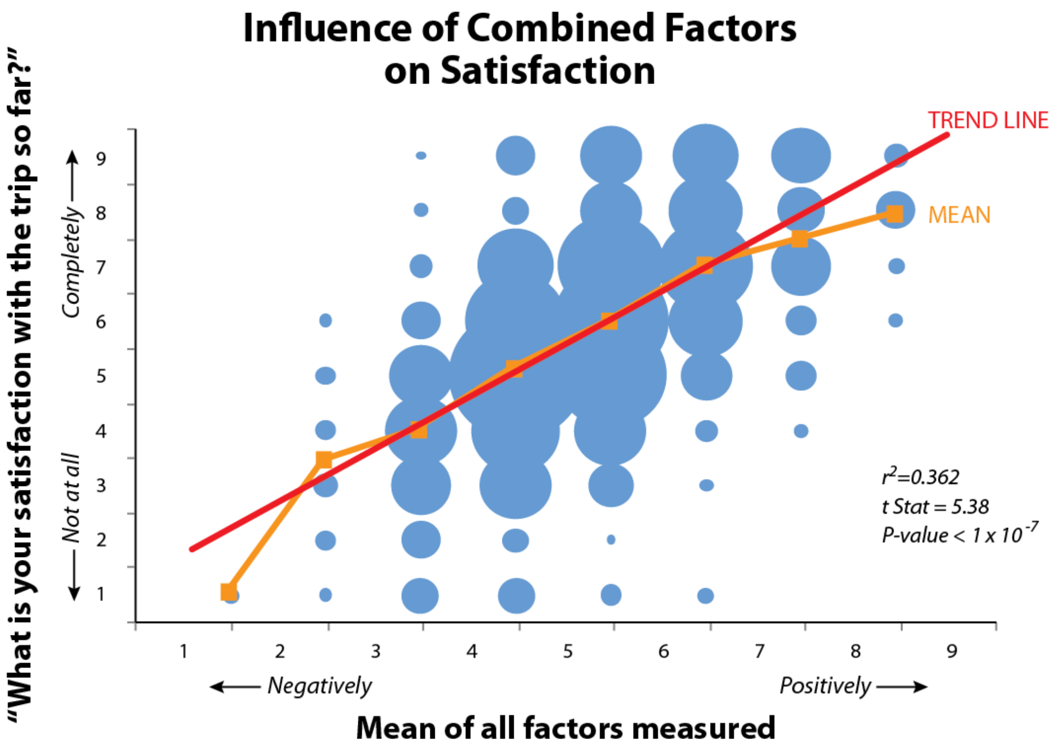


Figure 32 – Influence of combined external factors on rider satisfaction

5.1.5 Influence of Time Sensitivity

Participants were asked a series of questions in the online pre-survey to ascertain their prior experience and familiarity riding transit, and their sensitivity to arriving at an appointment on time. At the beginning of each in-app survey, participants were how important it is for them to arrive on time for the trip they were about to take.

Analysis of the pre-trip rider experience and time sensitivity indices did not reveal a correlation or pattern in the observed results. The homogeneity of the sample participants, all undergraduate students, may contribute to that lack of significance. However, a pattern emerged for arrival importance in the survey app results. Participants who indicated that it was not important at all to arrive on time indicated that comfort and transit system performance were contributing negatively to their satisfaction. The opposite is true for participants who indicated a high importance to arrive on time (Figure 33). A possible explanation is that those with low arrival importance may be captives who are only completing a trip on transit for the survey because they are required to do so.

When the bus was running early or late, the distribution of responses was impacted complimentarily. If the bus was running late, perception of arrival confidence and transit system performance was skewed towards the negative side, while the opposite is true when the bus was running early. As other research has shown, delay presents a cost penalty, which we can observe in these results Figure 34.

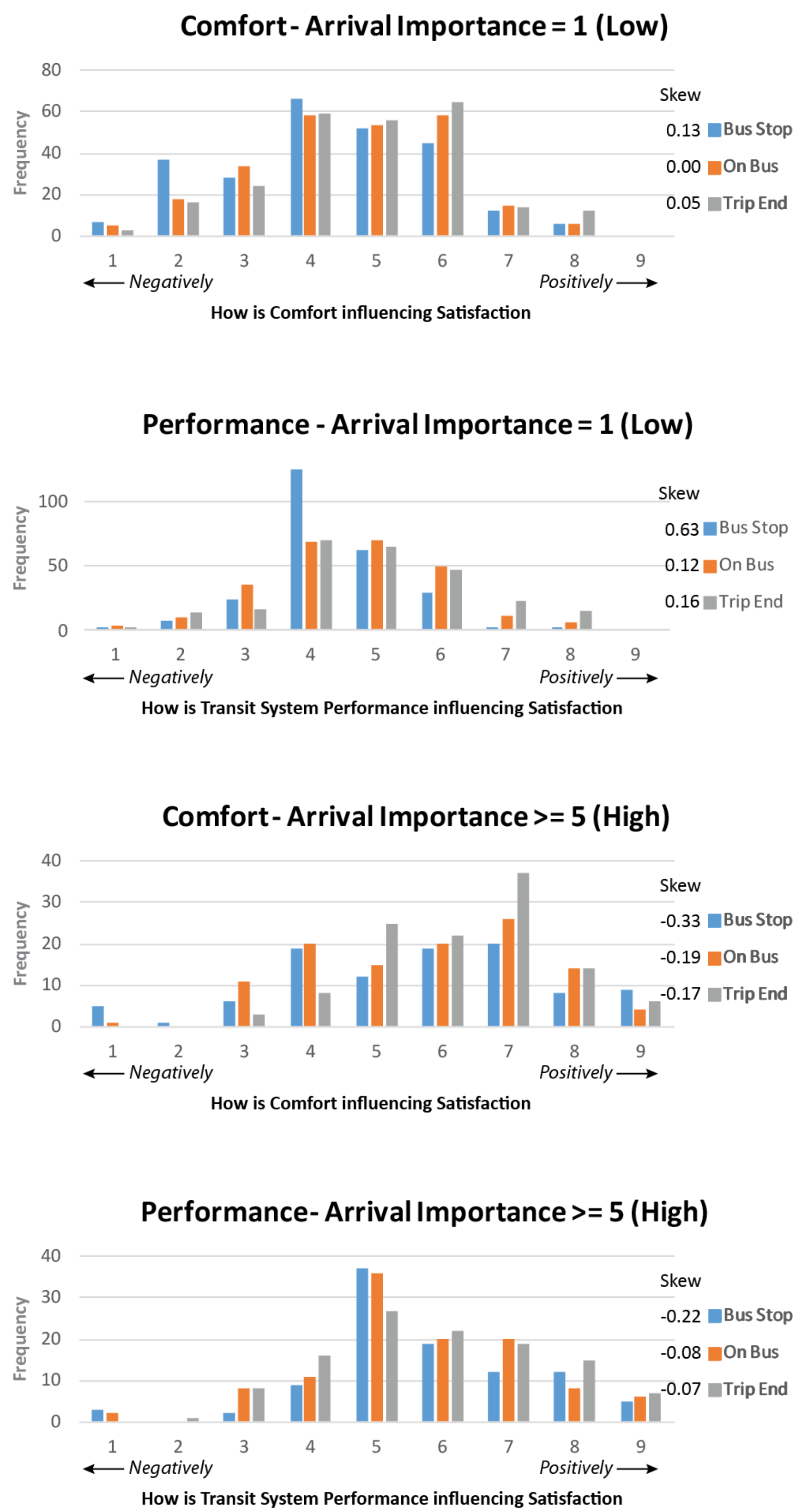


Figure 33 – Influence of comfort and performance on satisfaction for low/high arrival time importance

Figure 34 – Influences of running early or late on perceptions of on-time and system performance

5.2 Spatial Analysis

An important goal of this research, and part of the proof-of-concept to be demonstrated with the WPTI Survey app, is the collection of geospatial data in real-time along with the perceptual/satisfaction responses of the participant. Usable GPS data were gathered for 141 of the trips, which was less successful than expected due to a technical issue. Unusable GPS data, with partial routes or significant data gaps, were discarded after analysing the routes in ESRIArcMap.

A coding problem with the iPhone version of the app caused the devices to report coarse, instead of precise, location data. The coarse setting identifies location using cellular service triangulation rather than pin-pointing an exact location via satellite GPS. This setting has lower battery consumption than full GPS tracking, but otherwise is not useful for the purposes of this research. A few iPhone devices did provide accurate GPS data, likely because another app on the participant's device has this feature enabled.

Most of the successful GPS data were provided by the Android users, but not to the full extent that was hoped. Some participants may have disabled or blocked location services on their devices. A more reliable method of GPS data collection needs to be considered for the next round of surveying, whether this be ensuring that the participant's device is properly set (which may create privacy concerns), or providing them with an accompanying GPS tracking device.

The map in Figure 35 shows the 141 GPS routes taken by the participants. As expected for the student sample group, almost all of the trips are originating or destined for the University of Waterloo. GPS coordinates for these routes closely matched real-world location, as can be seen by how closely the GPS track matches the roadways on this map.

When the app is in survey mode, the app requests GPS coordinates every 3 seconds. Although frequent GPS tracking can drain the smartphone battery, the trips being studied are relatively short in duration. In addition to route tracking at 3-second intervals, the app records a waypoint when each iteration of the survey is initiated. The app also separately records the latitude and longitude of where each survey iteration was initiated, which simplifies GIS data analysis.

GPS tracking begins when the in-trip survey is initiated. The second iteration of the survey occurs when the participant is at the bus stop, prior to boarding the bus. The app records the bus stop coordinates. The app senses that the participant has boarded the bus once the distance from that coordinate is greater than 10 metres. The first in-bus iteration of the survey is then thrown.

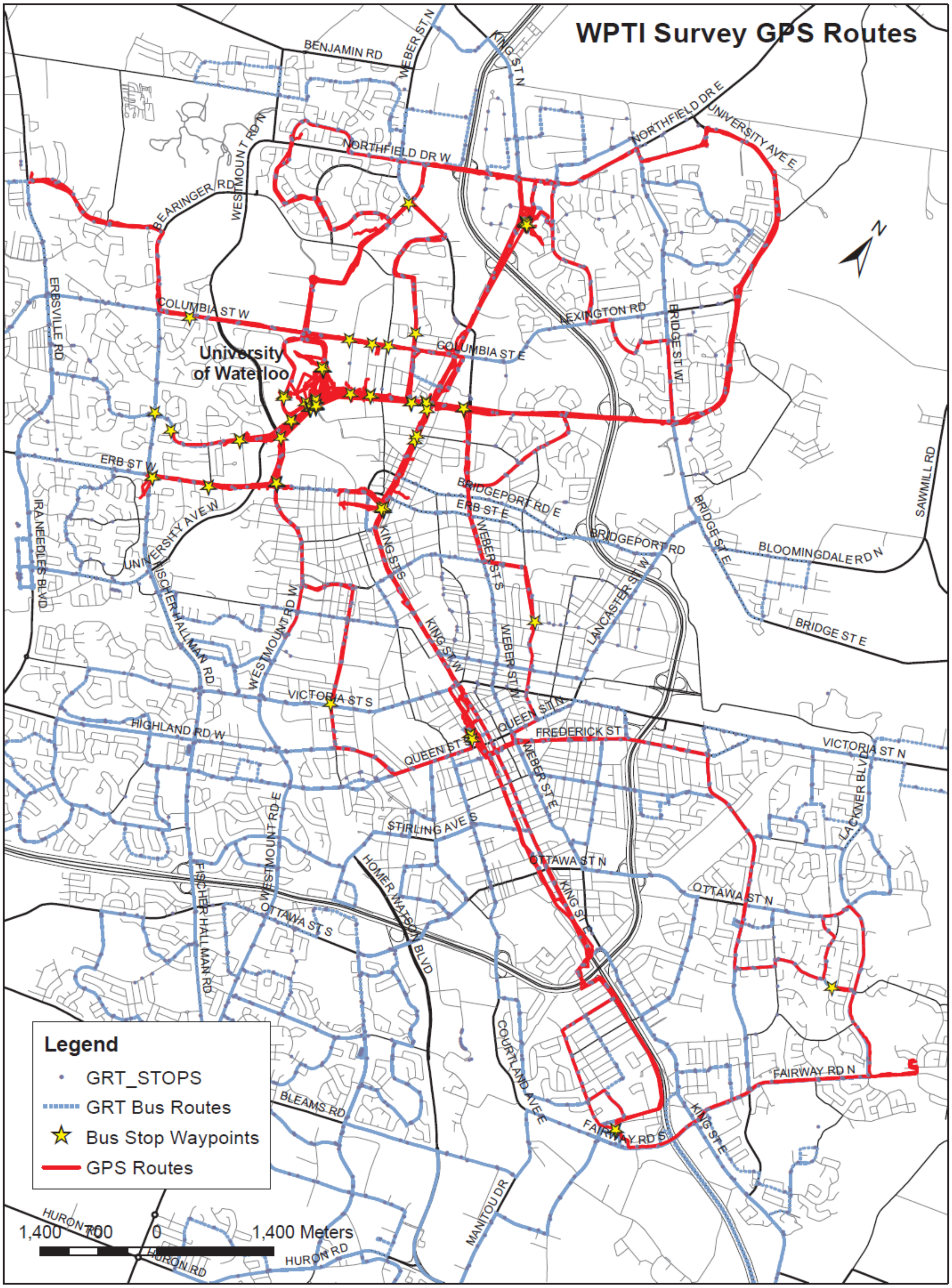


Figure 35 – Map showing the GPS data collected from the WPTI Survey, along with Grand River Transit (GRT) routes and bus stops. Bus Stop Waypoints are spatially joined data between a bus stop sensed in the app and a GRT physical bus stop location. This map was produced using ArcMap.

5.2.1 Merging Results with AVL/APC Data

The most important waypoint to be recorded is the bus stop where the participant is catching the bus. This coordinate, along with the time the participant boarded, can be spatially joined with GRT's bus stop points, Automatic Vehicle Location (AVL), and Automatic Passenger Count (APC) data, with the aim of determining the actual bus route and vehicle were boarded.

Bus stop waypoints successfully matched between the app's GPS waypoint data and Grand River Transit's bus stop data are highlighted in Figure 35. Matching of the GPS waypoints to bus stops was achieved in ArcMap by selecting the closest bus stop within a distance buffer of the waypoint. The AVL data, which identifies the time that a bus departs a bus stop, can be used to trace the bus' on-time performance during the run, and APC monitors the vehicle loading. Delays and crowded vehicles are expected to reduce rider satisfaction. By linking to the transit service provider's data, the observations of the participant can be validated.

The WPTI Survey App begins collecting GPS information at the bus stop. When the participant boards the bus, the app recognizes the change in location and then continues with the on-bus portion of the survey. The bus stop GPS waypoint is spatially joined to GRT's open source bus stop GIS data (Figure 36), enabling cross-referencing to both datasets. The route that the participant boarded can be determined by comparing GRT's open source Google Transit Feed Specification (GTFS) schedule data by matching the bus departure time closest to the boarding time recorded by the app. If the bus is off schedule, the departure time will not match the GTFS time and has to be looked up manually. Once the route is determined, the AVL and APC data for the route and date can be requested from the transit service provider. The boarding time can then be correlated to the actual bus boarded in GRT's AVL data using Excel. From the AVL match, the APC data from that vehicle can be correlated to see how many passengers were on board. Twenty-two of the trips were correlated in this way.

Matching the trip to a route and linking AVL/APC is particularly difficult in congested areas around the university, or near terminals where routes pulse on a schedule. Accurate matching requires manual verification. A simple improvement to the survey is to ask the participant if they know which route they intend to catch, which would make route verification much easier.

At the micro level, anecdotal accounts can be compiled to analyse individual trip characteristics. Once the data are set up and joined in ArcMap, selecting a bus stop brings up data for both WPTI Survey results and GRT data. It is thereby possible to track a survey participant through space and time as they make their trip, and observe changes in their satisfaction, schedule adherence, bus crowding, and other factors. However, due to the small sample size, no overall trends in reduced satisfaction were observed in these 22 successfully matched results. Five of these trips were 5 or more minutes behind schedule, but these participants did not state a reduced level of satisfaction.

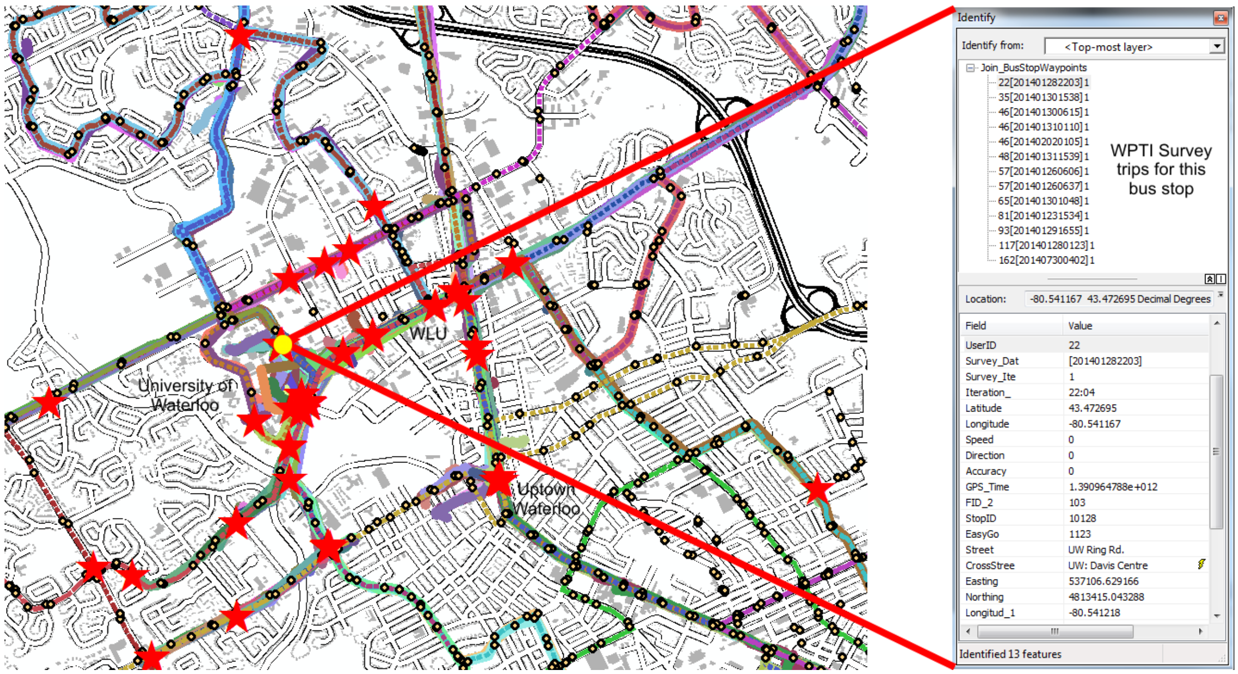


Figure 36 - WPTI Survey bus stops (red stars) correlated to GRT bus stops in ArcMap. Selecting one of these points in ArcMap's interface accesses data for both WPTI Survey results and GRT data.

The amount of manual intervention required to match GPS data to the corresponding bus run proved to be a challenge. The lack of GPS data successfully collected by the app can be overcome by improving the program, but more routes will compound the problem of matching this data to a bus. The work for this research was completed with ArcMap and Excel. The easiest matching was the bus stop GPS coordinate to the actual bus stop using the spatial-join feature in ArcMap, except at major transfer points where two or more stops were present. Congested service areas thereby proved to be the most problematic, requiring the most manual analysis to confirm the correct run was matched to the GPS tracking. In Excel, the the first match based on the closest time and bus stop was not always the correct one. A pivot table was used to find the top 3 closest matches, and the correct one manually verified. Use of more sophisticated database software could make it easier to match the bus run to the GPS stream with customized programming. Additional input from the participant (i.e. the question "Which route did you catch?") is a simpler solution.

This chapter summarised the results of the WPTI Survey and confirmed the proof of concept that a ESM-based satisfaction survey could be successful. A rich dataset was produced, although the satisfaction trends were not as clear as hoped. This first attempt did run into problems, including the aforementioned difficulty in successfully matching spatial data. Of 688 trips recorded with the app, 457 were validated, 141 had good GPS tracking, and just 22 were matched to AVL and APC. With the suggestions to be outlined in the conclusion, the success rate will be greatly improved. However, the work required to analyse the large volume of data from the survey questions and spatial analysis from a larger population sample will take considerable time.

6 Conclusion

6.1 Research Goals Achieved

The WPTI Survey app research project was a successful proof of concept for using smartphones and ESM to achieve the stated research goals.

6.1.1 Development and Testing of Smartphone App

A multi-platform smartphone app was developed to track a group of transit users through space and time to gather their perceptions of satisfaction and system performance. A rich data set was produced and analysed. Modifications can be made to the app to improve:

- the reliability of GPS tracking;
- the format of data files for ease of uploading and processing;
- the recording of satisfaction, particularly eliminating the voice response option which requires transcribing;
- the limits of the Likert scale sliders, with a reduction from 9 to 5 positions; and
- the ability to calibrate baseline satisfaction for participants.

6.1.2 Triggers of Rider Dissatisfaction and Anxiety

Triggers of transit rider dissatisfaction and anxiety that are a cost penalty against transit modal split were identified and validated by the results of the survey app. The results show correlations between satisfaction and transit system performance, other trip attributes, and time sensitivity. The homogeneity of the small sample group, comprised of all undergraduate students who had to participate in the survey for an assignment, likely contributed to the results not being as conclusive as hoped.

6.1.3 Spatial Data Joining and Technologies

Data from the survey were successfully cross-referenced to GIS mapping, transit route and bus stop data, and the transit service provider's AVL and APC data. GPS tracking within the software app can be made more reliable, or new wearable Bluetooth technologies can be employed to reliably integrate GPS data collection with a dedicated device. Other wearable technologies could further enhance the survey's capabilities, such as point-of-view video recording and monitoring the participant's heart rate and stress level to passively reveal triggers of anxiety.

6.1.4 Influence of Rider Experience and Time Sensitivity

Evidence was found that the time sensitivity of the transit rider influences anxiety and satisfaction, but the significance of rider experience was inconclusive. More detailed analysis of the pre-trip data and in-app data could reveal some correlation, but as noted earlier, the sample size and uniformity of participants is a limitation. The survey designed certainly appears capable of observing these influences, but further thought should go into ensuring pre-trip questions are worded appropriately.

6.2 Modelling Satisfaction

Collection of data on transit rider satisfaction will eventually lead to more accurate generalized cost modelling by enabling relative satisfaction to be considered in the time component weight factors. Other research has already determined that certain time components, such as in-vehicle wait time, are more onerous than other types of wait time, which increases the cost penalty for that stage of the trip. Satisfaction may provide a metric for calibrating these time components, by linking transit service characteristics and the sensitivities of the target service population. Further research using the WPTI Survey or similar application on a more diverse sample is encouraged to validate this hypothesis.

Looking back on the generalized cost and utility of transit as inputs to the Logit model, satisfaction in equation (4):

$$GC^{transit} = VOT(T_A F_A + T_{BS} F_{BS} + T_{IV} F_{IV} + T_{TR} F_{TR} + T_E F_E) + Fare \qquad (4)$$

The value for F, the weight factor for the perception of time can be adjusted for satisfaction for each of the time components. For example, F_{BS} is influenced by the availability of bus stop amenities, whereby satisfaction is higher if the bus stop has a shelter, comfortable surroundings, and bus arrival information present. Without such amenities, satisfaction is lower, increasing the value of F and thereby the associated cost penalty.

It can be derived indirectly from general observation and the anecdotal survey results that higher satisfaction would be derived from a transit system with better on-time performance, higher prevalence of real-time arrival information and comfort amenities at the bus stops, more direct route choices and service frequency, and thereby lower time-cost perception, than a transit system with poorer attributes. Such a discussion opens an entirely new area of research based on comparing transit system attributes to impute the level of rider satisfaction. That assertion goes beyond simply examining the impact of reliability on user cost towards looking at a basket of characteristics. So, for example, could a less reliable transit system overcome negative time perception by offering a high level of comfort and user experience?

6.3 Survey Improvements and Future Research

Gathering perceptions of satisfaction directly is hampered by one's individual definition of satisfaction and other mitigating factors. The flat-line mean satisfaction observed through all trip stages also suggests individuals continually reassess their level of satisfaction based on their current circumstances. A method for determining a baseline for individual satisfaction, or aggregating satisfaction based on other perceptive metrics such as comfort, system performance, weather, and time sensitivity, is necessary for correlating these data to how time and value of time are perceived as a user cost.

Another improvement to the app would be a capability to automatically detect changes transportation mode and ride comfort by using the phone's accelerometer data. GPS integration with map services, such as Google Maps, can be used to infer trip purpose by identifying land uses near origin and destination bus stops. Such technology was envisioned for the thesis research of Nour (2015). Integration of his transportation mode classification model along with life monitoring technology would significantly reduce the amount of active interaction, and thereby distraction, the survey participant would need to do with the survey app.

The homogeneity and size of the sample set for this research may have also brought a bias to the results. The results could have been more meaningful if, rather than 150 people over 2 or 3 days, the travel of and satisfaction of a smaller group of participants was tracked over 2 or 3 weeks. If this work continues, a representative population, with appropriate guidance for how to use the app, may provide results that reinforce the concept of user satisfaction as a cost penalty. Ideally, a large sample over a longer period would possibly be incorporated into a broader survey, such as the recurring Transportation Tomorrow Survey, using a smartphone app. In fact, Transportation Tomorrow 2.0 is piloting a smartphone app to log trips for the next survey in 2021 (UTTRI, 2018). Incorporating satisfaction metrics into such a survey app would further enrich the data and analysis of mode choice among participants.

References

Benjamin, J. (1986). Utilization of attitudinal measurement techniques to analyse demand for transportation methods, applications and new directions. In M. Clarke, & e. al, *Behavioural research for transport policy* (pp. 383-403). Utrecht: VNU Science Press.

Bick, C. (2011). Qualitatively understanding transit behavior from the rider's point of view. *Journal of public transportation, 14:3* , 1-20.

Brakewood, C., Barbeau, S., & Watkins, K. (2014). An experiment evaluating the impacts of real-time transit information on bus riders in Tampa, Florida. *Transportation Research Part A 69* , 409-422.

Brown, J., Hess, D. B., & Shoup, D. (2004). Unlimited Access. *Transportation 28* , 233-267.

Carrel, A., Halvorsen, A., & Walker, J. L. (2013). Passengers' Perception of and Behavioral Adaptation to Unreliability in Public Transportation. *Transportation Research Record: Journal of the Transportation Research Board. Vol. 2351 / Transit 2013, Vol.* , 152-162.

Carrel, A., Mishalani, R. G., Sengupta, R., & Walker, J. L. (2016). In Pursuit of the Happy Transit Rider: Dissecting Satisfaction Using Daily Surveys and Tracking Data. *Journal of Intelligent Transportation Systems, Vol. 20 , Iss. 4* .

Casello, J., Nour, A., & Hellinga, B. (2009). Quantifying impacts of transit reliability on user costs. *Transportation Research Record 2112* , 136-141.

Clark, A., & Doherty, S. T. (2010). A multi-instrumented approach to observing the activity rescheduling decision process. *Transportation. 2010(37)* , 165-181.

Doherty, S. T., Lemieux, C. J., & Canally, C. (2014). Tracking human activity and well-being in natural environments using wearable sensors and experience sampling. *Social Science & Medicine 106* , 83-92.

Dohmen, T., Falk, A., Huffman, D., & Sunde, U. (2011). Individual Risk Attitudes: Measurement, Determinants and Behavioral Consequences. *Journal of The European Economic Association* , Volume 9, Issue 3, pp 522-550.

Eboli, L., & Mazzulla, G. (2011). A methodology for evaluating transit service quality based on subjective and objective measures from the passenger's point of view. *Transport Policy 18* , 172-181.

Fan, Y., Chen, Q., Liao, C.-F., & Douma, F. (2012). *Smartphone-Based Travel Experience Sampling and Behavior Intervention among Young Adults.* Fan, Y., Chen, Q., Liao, C-F., & Douma, F. (2012). Smartphone-Based Travel Experience Sampling and Behavior Intervention among Young Adults. Intelligent Transportation Systems Institute. University of Minnesota. CTS 12-11.

Forsyth, E., & Smyth, A. (1986). The application of attitudinal research to the engagement of urban public transport. In M. e. Clarke, *Behavioural research for transport policy* (pp. 405-422). Utrecht: VNU Science Press.

Fries, R. N. (2011). University Traveler Value of Potential Real-Time Transit Information. *Journal of Public Transportation, Vol. 14, No. 2* .

Friman, M. (2004). Implementing quality improvements in public transport. *Journal of public transportation, 7(4)* , 49-66.

Gooze, A., Watkins, K. E., & Borning, A. (2013). Benefits of Real-Time Transit Information and Impacts of Data Accuracy on Rider Experience. *Transportation Research Record, No. 2351* , 95-103.

Gray, R. H. (1983). Estimating gravity model parameters: A simplified approach based on the odds ratio. *Transportation Research- B. Vol. 17B, No. 2* , 117-131.

Hagerstrand, T. (1970). What about people in regional science? *Papers of the regional science association, Vol XXIV* , 7-21.

Haywood, L., & Koning, M. (2013). Avoir les coudes serrés dans le métro parisien: évaluation contingente du confort des déplacements. *Revue d'Economie Industrielle. Volume 140, Issue 4* , 111-144.

Hektner, J. M., Schmidt, J. A., & Csikszentmihalyi, M. (2007). *Experience sampling method: Measuring the quality of everyday life.* New York: Sage Publications.

Hollander, Y. (2010). Multiple objectives in travel demand modelling. *Transportation Research Record, No. 2175* , 120-129.

Iseki, H., Taylor, B. D., & Miller, M. (2006). *The Effects of Out-of-Vehicle Time on Travel Behavior: Implications for Transit Transfers (Deliverable #1).* Los Angeles: California PATH Program, Institute of Transportation Studies, University of California.

Lai, W.-T., & Chen, C.-F. (2011). Behavioral intentions of public transit passengers: The roles of service quality, perceived value, satisfaction and involvement. *Transport Policy 18(2)* , 318-325.

Lisco, T. E. (1967). *The value of commuters travel time: A study in urban transportation (Thesis dissertation).* Chicago: University of Chicago, Department of Economics.

Lui, Y., Bunker, J. M., & Ferreira, L. (2010). Transit users' route-choice modelling in transit assignment: A review. *Transport Reviews 30(6)* , 753-769.

Murga, M. (2002). *The 4-step demand model.* Massachusetts Institute of Technology.

Nour, A. (2015). *Automating and optimizing a transportation mode classification model for use on smartphone data.* University of Waterloo PhD Thesis dissertation.

Nour, A., Casello, J. M., & Hellinga, B. (2010). Anxiety-based formulation to estimate generalized travel time. *Transportation research record: Journal of the transportation research board* , 108-116.

Ryus, P., Dannaher, A., Walker, M., Nichols, F., Carter, B., Ellis, E., et al. (2013). *Transit Capacity and Quality of Service Manual, Third Edition (TCRP Report 165).* Transit Cooperative Research Program.

Shrewsbury, J. (2012). *Calibration of trip distribution by generalized linear models.* Canterbury, NZ: University of Centerbury, New Zealand Transport Agency Report 473.

St-Louis, E., Manaugh, K., van Lierop, D., & El-Geneidy, A. (2014). The happy commuter: A comparison of commuter satisfaction across modes. *Transportation Research Part F 26* , 160-170.

Swait, J. D., & Ben-Akiva, M. (1986). Analysis of the effects of captivity on travel time and cost elasticities. In M. e. Clarke, *Behavioural research for transport policy* (pp. 119-134). Utrecht: VNU Science Press.

Transportation Cooperative Research Program. (2011). *Guidelines for providing access to public transportation stations, Appendix B: Assessment and evaluation tools.* TCRP.

Transportation Information Steering Committee. (2008). *Transportation Tomorrow Survey, 2006.* University of Toronto Dept. of Civil Engineering.

Transportation Research Board. (2007). *Metropolitan Travel Forecasting: Current Practice and Future Direction.* TRB Special Report 288.

Turcotte, M. (2010). Commuting to work: Results of the 2010 General Social Survey. *Component of Statistics Canada Catalogue no. 11-008-X. Canadian Social Trends* .

UTTRI. (2018). *About TTS 2.0*. Retrieved 2018, from Transportation Tomorrow 2.0: https://tts2.ca

Verron, H., & Martens, G. (1986). Users' reactions to specific combinations of policy measures. In M. e. Clarke, *Behavioural Research for Transport Policy* (pp. 135-151). Utrecht: VNU Science Press.

Yoh, A., Iseki, H., Smart, M., & Taylor, B. D. (2011). Hate to wait: Effects of wait time on public transit travelers' perceptions. *Transportation Research Record, 2216* , 116-124.

Appendices

Appendix A: App variables

PRE-TRIP SURVEY

timePickUp	What time does Subject expect to be picked up at bus stop?
timeDest	What time does Subject need to be at destination?
timeDropOff	What is the scheduled arrival time at destination bus stop?
flagPickUp	0 = exact time; 1 = approximate time; -1 = don't know
flagDest	0 = exact time; 1 = approximate time; -1 = don't know
flagDropOff	0 = exact time; 1 = approximate time; -1 = don't know
slideImportant	How important is it to arrive at the destination on time?
tripPurpose	Purpose of trip
timeStart	Time survey initiated by Subject

TRIP SURVEY

slideSatisfaction(X)	Curent satisfaction (sequential array)
txtChange(X)	Text input for slideSatisfaction(+1) ∆> 1
slideEEnv(X)	Influence of physical surroundings, Environment
slideEWeath(X)	Weather
slideECond(X)	Condition of the bus or bus stop
slideEComf(X)	Personal comfort
slidePPerf(X)	Influence of transit system performance
slidePDrv(X)	Operator's courtesy and driving
slidePArr(X)	Expected arrival performance
slidePConf(X)	Expected arrival confidence
timeBusStop	Time survey at bus stop initiated by Subject
timeSurvey	Time of current survey initiation
timeDestArr	Time Subject completes survey at destination
timeWait	Length of time Subject thinks they waited at the bus stop (asked once on bus)
timeOnBus	Length of time Subject thinks they were on the bus (asked at destination)
idTrip	Unique trip identifier
idUser	Unique user identifier
gpsCurrent(X)	Current GPS position
gpsBusStop	GPS position when survey initiated at bus stop
gpsSpeed(X)	Current GPS speed, if available
surveyOutput	String to write data to file

TEMPORARY CACHE (used for comparisons)

slideArrFirst	The first expected arrival performance value
slideArrPrev	The previous expected arrival performance value
slideConfFirst	The first arrival confidence value
slideConfPrev	The previous slider confidence value
slideSatFirst	The first satisfaction slider value
slideSatPrev	The previous satisfaction slider value
gpsPrev	The previous GPS position
timeLapsed	Time lapsed since the last survey throw

countX Counter for sequential array (0, bus stop; to (X), the number of surveys completed during the trip; dimension array to 20 to allow for variable trip length or use redimensionable array?

flagInput Tracks whether or not the Subject has provided input before enabling Submit button

countScr Screen Counter

Appendix B: Original Program Algorithm for WPTI Survey App

Subject initiates survey before their trip
Subject completes the PRE-TRIP SURVEY
App initiates GPS tracking and waits for subject to arrive at bus stop

Subject walks to bus stop

At the bus stop, subject presses "I'm at the Bus Stop" button
App records time and GPS
App throws Full Survey to Subject

Once App senses GPS distance from bus stop > 30m, throws Adaptive Survey to Subject
How long do you think were you waiting for the bus?
- In event GPS inoperable, Adaptive Survey will be thrown every 6 minutes after Bus Stop initiation
- Adaptive survey prompts user to qualify answer (with Full Survey) if Satisfaction change > 1

App prompts Subject to complete Adaptive Survey every 6 minutes
- How can we use GPS to enable throwing of Adaptive Survey at key transfer points
- Note: User may manually initiate a survey during the trip to capture changing conditions

When GPS senses Subject is walking away from the bus (how to do this?), a final Adaptive Survey will be thrown.

At final destination, subject presses "I'm at my final Destination" button
App records time and GPS
App throws Final Survey to Subject

FULL SURVEY
What is your current satisfaction with your trip so far?
What motivated this change in satisfaction level? (option for recorded or text input)
How do your physical surroundings (eg. weather, bus, comfort, etc) influence your current satisfaction? (include check boxes for attributes)
How does the transit system performance influence your current satisfaction?
When do you expect to arrive at your destination?
How confident are you in your expected arrival time?

ADAPTIVE SURVEY
What is your current satisfaction with your trip so far?
- If Delta > 1, throw:
- What motivated this change in satisfaction level? (text input)
- How do your physical surroundings (eg. weather, bus, comfort, etc) influence your current satisfaction? (include sliders for attributes)
- How does the transit system performance influence your current satisfaction? (sliders)
When do you expect to arrive at your destination? (slider)
How confident are you in your expected arrival time? (slider)

FINAL SURVEY
What was your overall satisfaction with your trip?
How did the physical surroundings influence your satisfaction?
How did the transit system performance influence your satisfaction?
Did you arrive when you expected?
How long do you think you were on the bus?

Appendix C: WPTI Survey App – Android Version Code

Android Manifest.xml

The Android Manifest file sets out the general parameters and permissions that the app needs to run. The permissions refer to system functions, such as the ability to write data to the phone's internal file system, access GPS location coordinates, record audio, wake the phone from sleep mode, and access the internet for uploading files. The various Activities listed are the user-interface components of the app and how they relate to each other.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.wptiapp1"
    android:versionCode="6"
    android:versionName="0.996" >

    <uses-sdk
        android:minSdkVersion="7"
        android:targetSdkVersion="19" />

    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name="com.wptiapp1.FirstActivity"
            android:label="@string/app_name"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.wptiapp1.SupplementalActivity"
            android:label="@string/title_activity_supplemental"
            android:parentActivityName="com.wptiapp1.FirstActivity"
             >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.sampleandroidapp.FirstActivity" />
        </activity>
        <activity
            android:name="com.wptiapp1.EndActivity"
```

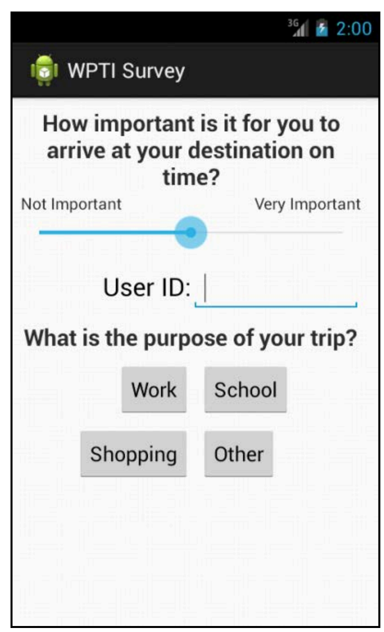
```
            android:label="@string/title_activity_end"
            android:parentActivityName="com.wptiapp1.FirstActivity"
             >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.sampleandroidapp.FirstActivity" />
        </activity>
        <activity
            android:name="com.wptiapp1.InputActivity"
            android:label="@string/title_activity_input"
            android:parentActivityName="com.wptiapp1.SupplementalActivity"
             >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.sampleandroidapp.SupplementalActivity" />
        </activity>

        <service
            android:name="com.wptiapp1.MediaPlayerService"
            android:enabled="true" />

        <receiver
            android:name="com.wptiapp1.TimeBroadcastReceiver"
            android:enabled="true" >
        </receiver>
        <receiver
            android:name="com.wptiapp1.Alarm"
            android:process=":remote" >
        </receiver>

        <activity
            android:name="com.wptiapp1.SurveyActivity"
            android:label="@string/title_activity_survey"
             >
        </activity>
        <activity
            android:name="com.wptiapp1.SettingsActivity"
            android:label="@string/title_activity_settings"
             >
        </activity>
    </application>

</manifest>
```

Strings.xml

The Strings file defines all of the user interface variables used in the app code. Some components of the user interface is dynamic, meaning that the information the user sees can change depending on what stage the user is in their trip. The Strings file sets all of the default values for the user interface variables, which can be changed through the code later on.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">WPTI Survey</string>
    <string name="action_settings">Settings</string>
    <string name="question_top">How important is it for you to arrive at your destination on time?</string>
```

```
    <string name="question_top2">Describe how you currently feel about:</string>
    <string name="question_bot">What is the purpose of your trip?</string>
    <string name="question_end">Did you arrive at your destination on time?</string>
    <string name="question_input">Briefly describe your current surroundings and why you feel that way:</string>
    <string name="question_input2">Please record a voice message by pressing this button. If you aren\'t comfortable talking right
now, you can type a text message below.</string><string name="slideVal1_low">Not Important</string>
    <string name="slideVal1_high">Very Important</string>
    <string name="slideVal1_mid"></string>
    <string name="slideVal2_low">Terrible</string>
    <string name="slideVal2_high">Tremendous</string>
    <string name="slideVal2_mid">Neutral</string>
    <string name="txtLabel_id">User ID:</string>
    <string name="butPurp1">Work</string>
    <string name="butPurp2">School</string>
    <string name="butPurp3">Shopping</string>
    <string name="butPurp4">Other</string>
    <string name="radio1">Exactly</string>
    <string name="radio2">Approximately</string>
    <string name="radio3">Don\'t know</string>
    <string name="submitButton">Continue</string>
    <string name="endButton">End of Trip</string>
    <string name="recButton">Record voice message</string>
    <string name="title_activity_supplemental">Supplemental Questions</string>
    <string name="hello_world">Hello world!</string>
    <string name="title_activity_end">End of Survey</string>
    <string name="mins">Minutes</string>
    <string name="plus">+</string>
    <string name="minus">-</string>
    <string name="title_activity_input">Record Message</string>
    <string name="title_activity_survey">WPTI Survey</string>
    <string name="title_activity_settings">Settings</string>
    <string name="settings_top">Survey Settings</string>
    <string name="txtLabel_auto">Use Automatic Interval</string>
    <string name="txtLabel_interval">Survey Interval:</string>
    <string name="butOk">OK</string>
    <string name="butCancel">Cancel</string>
    <color name="backgnd">#FFFFFF</color>
    <string name="txtLabel_upload">Upload data:</string>
    <string name="butUpload">Upload</string>

</resources>
```

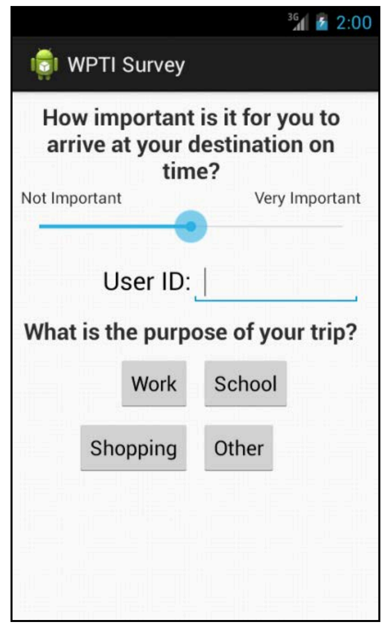


First Activity (User Interface)

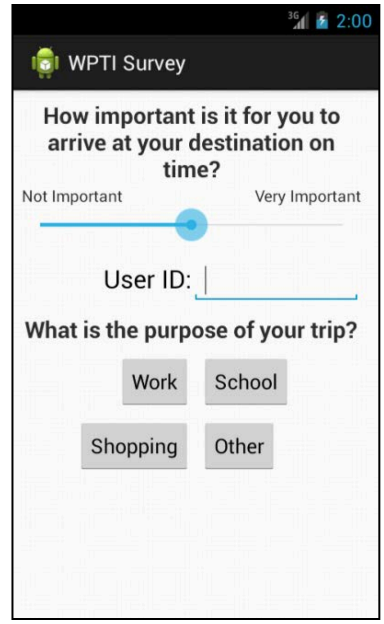
First Activity is, as the name suggests, the first app user interface component to be launched. There are two parts to the First Activity code, as there are to all of the user interface and code components of the app. The first is a .xml (Extensible Markup Language) file that defines the layout of the user interface. This dynamic screen is also the first one that the user sees when they launch the app. Once the information is completed, by selecting an importance, entering their user ID, and selecting a trip put purpose, the screen changes to display a time selector and assess if the user has checked the transit schedule prior to taking the trip. For the rest of the survey, the user is asked for their level of satisfaction for the trip so far and confidence in arriving on time.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

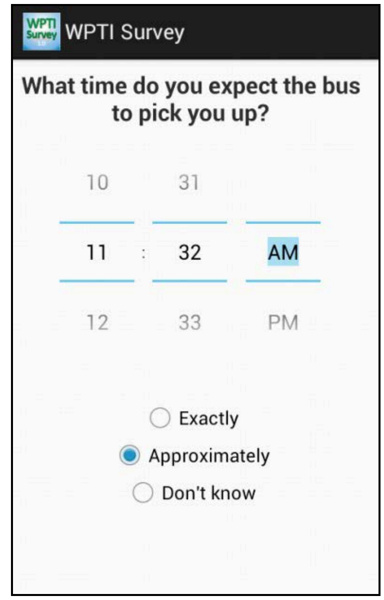
```
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/mainView1"
android:layout_width="match_parent"
android:layout_height="match_parent"

tools:context=".FirstActivity" >

<ScrollView
android:id="@+id/scrollView1"
android:layout_width="match_parent"
android:layout_height="match_parent" >

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginRight="16dip"
        android:layout_marginTop="8dip"
        android:layout_marginLeft="8dip" >

<TextView
    android:id="@+id/textQ1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:gravity="center|center_vertical"
    android:maxLines="4"
    android:text="@string/question_top"
    android:textSize="20sp"
    android:textStyle="bold" />

<TextView
    android:id="@+id/textSlide1_Min"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/textQ1"
    android:gravity="left"
    android:text="@string/slideVal1_low"
    android:textAppearance="?android:attr/textAppearanceSmall" />

<TextView
    android:id="@+id/textSlide1_High"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
             android:layout_below="@+id/textQ1"
             android:gravity="right"
    android:text="@string/slideVal1_high"
    android:textAppearance="?android:attr/textAppearanceSmall" />

<TextView
    android:id="@+id/textSlide1_Mid"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/textSlide1_Min"
    android:layout_alignBottom="@+id/textSlide1_Min"
```

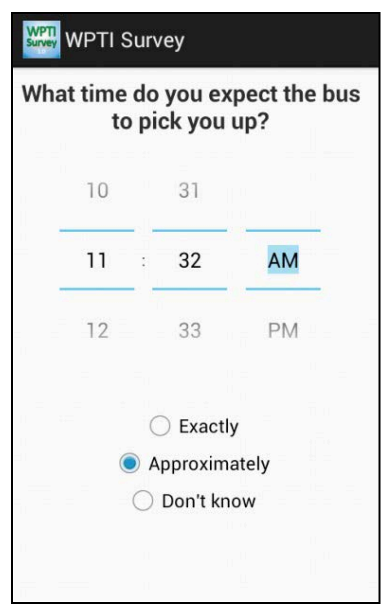


```
    android:gravity="center"
    android:layout_centerHorizontal="true"
    android:text="@string/slideVal1_mid"
    android:textAppearance="?android:attr/textAppearanceSmall" />

<SeekBar
    android:id="@+id/seekBar1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/textSlide1_Min"
    android:max="8"
    android:progress="4" />

        <requestFocus />

<TextView
    android:id="@+id/textLabel_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@+id/textSlide1_Mid"
    android:layout_below="@+id/seekBar1"
    android:layout_marginTop="16dp"
    android:text="@string/txtLabel_id"
    android:textAppearance="?android:attr/textAppearanceLarge" />

<TextView
    android:id="@+id/textQ2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_below="@+id/editText_UserID"
    android:gravity="center|center_vertical"
    android:layout_marginTop="16dp"
    android:maxLines="2"
    android:textSize="20sp"
    android:textStyle="bold"
    android:text="@string/question_bot" />

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textQ2"
    android:layout_alignRight="@+id/textSlide1_Mid"
    android:layout_marginTop="8dp"
    android:text="@string/butPurp1" />

<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@+id/button1"
    android:layout_alignTop="@+id/button1"
    android:layout_marginLeft="8dp"
    android:text="@string/butPurp2" />

<Button
```

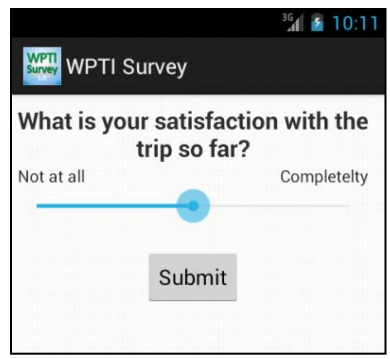


```
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/textSlide1_Mid"
        android:layout_below="@+id/button1"
        android:layout_marginTop="8dp"
        android:text="@string/butPurp3" />

    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/button3"
        android:layout_alignTop="@+id/button3"
        android:layout_marginLeft="8dp"
        android:text="@string/butPurp4" />

    <EditText
        android:id="@+id/editText_UserID"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/textLabel_id"
        android:layout_alignBottom="@+id/textLabel_id"
        android:layout_toRightOf="@+id/textLabel_id"
        android:ems="10"
        android:inputType="text" >

        <requestFocus />
    </EditText>

    <TimePicker
        android:id="@+id/timePicker1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textQ1"
        android:paddingBottom="20dp"
        android:layout_centerHorizontal="true"
        android:visibility="invisible" />

<RadioGroup
    android:id="@+id/radioGroup1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:layout_below="@+id/timePicker1"
    android:orientation="vertical"
    android:visibility="invisible"
    >

    <RadioButton
        android:id="@+id/radioButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/radio1" />

    <RadioButton
        android:id="@+id/radioButton2"
        android:layout_width="wrap_content"
```





```
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="@string/radio2" />

    <RadioButton
        android:id="@+id/radioButton3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/radio3" />
</RadioGroup>

    <Button
        android:id="@+id/butSubmit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/seekBar1"
        android:layout_marginTop="20dp"
        android:text="@string/submitButton"
        android:visibility="gone" />

    <Button
        android:id="@+id/butEnd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/butSubmit"
        android:layout_marginTop="20dp"
        android:text="@string/endButton"
        android:visibility="gone" />

        </RelativeLayout>
    </ScrollView>
</RelativeLayout>
```



First Activity (Java Code)

The second component of First Activity is a Java file that works behind the scenes. The code that is run when the First Activity is initiated coordinates with the user interface. Code in the Java file is executed when the user makes a selection on the user interface. This interaction will also cause the user interface to change to display the time questions. Annotations within the code (lines beginning with //) provide further insights different functions and the way the code is set up. The first section of code initiates all of the different system and software functions that the app needs by importing them. Next, variables are defined, followed by links between variables and user interface components. Various “Listeners” are initiated, which cause code in the Main Code section to be executed through user interaction, time, and location prompts.

```
package com.wptiapp1;

import java.util.Date;
import java.text.SimpleDateFormat;
import java.util.Timer;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

import android.net.Uri;
```

```java
import android.os.Bundle;
import android.os.Environment;
import android.os.Vibrator;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.AlarmManager;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.view.Menu;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.TextView;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.EditText;
import android.widget.TimePicker;
import android.widget.Toast;
import android.app.AlertDialog;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.content.res.Configuration;
import android.util.Log;
import android.location.*;

import com.wptiapp1.R;

@SuppressLint("SimpleDateFormat") public class FirstActivity extends Activity implements OnSeekBarChangeListener, OnClickListener{

        // Static variable declarations ---------------------------------------------------------------------------------
        boolean sliderAdjusted = false, endFlag = false, onBus = false, alertDisplayed = false, preTripDone = false;
        boolean gpsFlag = true;
        static boolean alarmOn = false;
        int  countScrn = 0, timeInterval = 3, timeNext = 0;
        static int countSurvey = 0, surveyInterval = 300;
        int flagPickUp, flagDest, flagDropOff, flagButPress, timeLapsed = 0, slideImportant;
        int slideSatCurrent, slideSatPrev, slideSatFirst, slideArrCurrent, slideArrPrev,
                slideArrFirst, slideConfCurrent, slideConfPrev, slideConfFirst;
        double gpsLat, gpsLong, gpsAlt;
        float gpsDir, gpsAcc, gpsSpeed;
        long gpsTime;
        String timePickUp, timeDropOff, timeDest, dateString, butPress, tripPurpose, txtInput;
        String fileName;
        static String appFolder, userID, timePreSurvey, timeSurvey, timeCurrent;
        Timer surveyTimer;
        LocationManager locMgr;
        Location loc;
        LocationListener locListener;
        PendingIntent myPendingIntent;
```

```java
    Intent timerIntent;

    // These declarations similar to IBOutlets in xcode
    private SeekBar slideImp;
    private TextView sliderMinVal,sliderMaxVal, sliderMidVal, txtQBot, yourID;
    static TextView txtQTop;
    private Button butWork, butSchool, butShop, butOther, butEnd, butPressed;
    static Button butSubmit;
    private RadioButton butTimeApprox, butTimeExact, butTimeDuno;
    private RadioGroup radioTime;
    static EditText theID;
    private TimePicker pickTime;
    private AlarmManager alertAlarm;
    static ViewGroup mainView;
    Animation animationFadeIn;

    // Initialize things in On Create ----------------------------------------------------------------------------------
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_first);

    // Declarations similar to IBActions in xcode
    mainView = (ViewGroup)findViewById(R.id.mainView1);
    slideImp = (SeekBar)findViewById(R.id.seekBar1); // make seekbar object
    slideImp.setOnSeekBarChangeListener(this); // set seekbar listener.
    sliderMaxVal = (TextView)findViewById(R.id.textSlide1_High);
    sliderMinVal = (TextView)findViewById(R.id.textSlide1_Min);
    sliderMidVal = (TextView)findViewById(R.id.textSlide1_Mid);
    butWork = (Button)findViewById(R.id.button1);
    butWork.setOnClickListener(this);
    butSchool = (Button)findViewById(R.id.button2);
    butSchool.setOnClickListener(this);
    butShop = (Button)findViewById(R.id.button3);
    butShop.setOnClickListener(this);
    butOther = (Button)findViewById(R.id.button4);
    butOther.setOnClickListener(this);
    butSubmit = (Button)findViewById(R.id.butSubmit);
    butSubmit.setOnClickListener(this);
    butEnd = (Button)findViewById(R.id.butEnd);
    butEnd.setOnClickListener(this);
    radioTime = (RadioGroup)findViewById(R.id.radioGroup1);
    butTimeApprox = (RadioButton)findViewById(R.id.radioButton2);
    butTimeApprox.setOnClickListener(this);
    butTimeExact = (RadioButton)findViewById(R.id.radioButton1);
    butTimeExact.setOnClickListener(this);
    butTimeDuno = (RadioButton)findViewById(R.id.radioButton3);
    butTimeDuno.setOnClickListener(this);
    theID = (EditText)findViewById(R.id.editText_UserID);
    txtQTop = (TextView)findViewById(R.id.textQ1);
    txtQBot =  (TextView)findViewById(R.id.textQ2);
    yourID = (TextView)findViewById(R.id.textLabel_id);
    pickTime = (TimePicker)findViewById(R.id.timePicker1);
    animationFadeIn = AnimationUtils.loadAnimation(this, R.anim.fadein);

    // Create the output folder if it doesn't exist, check SDCard (External Storage) first
    checkExternalMedia();
```

```
    // Setup location manager and listener for GPS tracking
    setupGPSListener();

    // Get survey timestamp (not needed here?)
    setSurveyTimestamp();

    // Lock screen orientation to prevent refresh
    if(getResources().getConfiguration().orientation == Configuration.ORIENTATION_PORTRAIT) {
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    } else setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
}

public boolean onCreateOptionsMenu(Menu menu) {
     getMenuInflater().inflate(R.menu.input, menu);
     return true;
     }

public boolean onOptionsItemSelected(MenuItem item) {
     if (theID.getText().length() != 0){
             userID = theID.getText().toString();
     }
     Intent myIntentSet = new Intent(this.getApplicationContext(), SettingsActivity.class);
             startActivityForResult(myIntentSet, 0);
             return true;
     }
// Setup Overrides for listener actions - these are called in response to user interaction------------------------------
@Override
public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
}

@Override
public void onStartTrackingTouch(SeekBar seekBar) {
}

@Override
public void onStopTrackingTouch(SeekBar seekBar) {
     sliderAdjusted = true;
}

     @Override
     public void onClick(View v) {
             butPressed = (Button)v;
             // First pre-trip survey screen
             if (preTripDone == false) {
                     if (theID.getVisibility() == View.VISIBLE){
                             getPurpose();
                     }
                     else if (pickTime.getVisibility() == View.VISIBLE){
                             getTravelTime();
                             mainView.startAnimation(animationFadeIn);

                     }
             } else {
                     if (butPressed.getText().toString().equals("Continue")){
                             showSurveyAlert("WPTI Survey", "Complete the Survey now?");
                     } else if (butPressed.getText().toString().equals("Submit")) {
                             getSubmit();
                     } else if (butPressed.getText().toString().equals("Start")) {
```

```java
                        getSubmit();
                } else if (butPressed.getText().toString().equals("End of Trip")) {
                        getEnd();
                } else if (butPressed.getText().toString().equals("Quit")) {
                        endProgram();
                }
        }
}

// Main code section -------------------------------------------------------------------------------------------

public void setSurveyTimestamp() {
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyyMMddHHmmss");
        timePreSurvey = dateFormat.format(new Date());
};

public void getPurpose() {
        if (sliderAdjusted == false) {
                txtQBot.setText("Set importance first, then select a purpose.");
        }
        if (theID.getText().length() == 0){
                txtQBot.setText("Please enter the User ID you were assigned.");
        }
        if (sliderAdjusted == true && theID.getText().length() > 0){
        setSurveyTimestamp();
        tripPurpose = butPressed.getText().toString();
                        //txtQBot.setText(tripPurpose);
        userID = theID.getText().toString();

        butWork.setVisibility(View.GONE);
        butSchool.setVisibility(View.GONE);
        butShop.setVisibility(View.GONE);
        butOther.setVisibility(View.GONE);
        slideImp.setVisibility(View.INVISIBLE);
        txtQBot.setVisibility(View.INVISIBLE);
        sliderMaxVal.setVisibility(View.INVISIBLE);
        sliderMinVal.setVisibility(View.INVISIBLE);
        theID.setVisibility(View.INVISIBLE);
        yourID.setVisibility(View.INVISIBLE);
        txtQTop.setText("What time do you expect the bus to pick you up?");
        radioTime.setVisibility(View.VISIBLE);
        pickTime.setVisibility(View.VISIBLE);
        //butSubmit.setVisibility(View.VISIBLE);
        }
}

public void setupPurposeScreen() {
    butSubmit.setVisibility(View.INVISIBLE);
    txtQTop.setText("How important is it for you to arrive at your destination on time?");
    slideImp.setProgress(4);
    butWork.setVisibility(View.VISIBLE);
    butShop.setVisibility(View.VISIBLE);
    butSchool.setVisibility(View.VISIBLE);
    butOther.setVisibility(View.VISIBLE);
    slideImp.setVisibility(View.VISIBLE);
    txtQBot.setVisibility(View.VISIBLE);
    sliderMinVal.setVisibility(View.VISIBLE);
    sliderMaxVal.setVisibility(View.VISIBLE);
```

```
    sliderMidVal.setVisibility(View.INVISIBLE);
    yourID.setVisibility(View.VISIBLE);
    theID.setVisibility(View.VISIBLE);
    butEnd.setText("End of Trip");
    butEnd.setVisibility(View.GONE);
    sliderAdjusted = false;
mainView.invalidate();
    countScrn = 0;
}

public void getTravelTime() {
        SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm");
        timeSurvey = timeFormat.format(new Date()).toString();
Log.i("timeSurvey: ", timeSurvey);

        // Get the selected time
        String hour = pickTime.getCurrentHour().toString();
        String minute = pickTime.getCurrentMinute().toString();
        if (hour.length() == 1) {
                dateString = "0" + hour;
        }
        else {
                dateString = hour;
        }
        if (minute.length() == 1) {
                dateString = dateString + ":0" + minute;
        }
        else {
                dateString = dateString + ":" + minute;
        }
Log.i("Selected Time: ", dateString);
Log.i("Button Pressed: ", butPressed.getText().toString());

        countScrn = countScrn + 1;

        // Which radio button was pressed?
if (butPressed.getText().toString().equals("Exactly")) {
    flagButPress = 0;
}
else if (butPressed.getText().toString().equals("Approximately")) {
    flagButPress = 1;
}
else {
    flagButPress = 2;
}

// Record the time selected for each of the three survey screens
if (countScrn == 1){
 flagPickUp = flagButPress;
 timePickUp = dateString;
 txtQTop.setText("At what time do you need to be at your destination?");
}
else if (countScrn == 2){
 flagDest = flagButPress;
 timeDest = dateString;
 txtQTop.setText("What time is the bus scheduled to arrive at the bus stop closest to your destination?");
}
else if (countScrn == 3){
```

```
        flagDropOff = flagButPress;
        timeDropOff = dateString;

                // Write PreTripSurvey data to a file.
        // Prepare output strings
        String preTripSurvey = "UserID = " + userID + "\nSurvey_Timestamp = " + timeSurvey + "\nImportance = " +
                        slideImportant + "\nTripPurpose = " + tripPurpose + "\nPickUpTime = " + timePickUp +", " + flagPickUp +
"\nDestTime = " +
                        timeDest +", " + flagDest + "\ntimeDropOff = " + timeDropOff +", " + flagDropOff;
            Log.d("Output string: ", preTripSurvey);

            // Unlike iPhone version, setting up the output folder was already done in onCreate

            // Make a file name to write the data to in the documents directory:
            fileName = userID + "_" + timePreSurvey + "_PreTrip.txt";
            Log.i("fileName: ", fileName);

            // Save content to the file
            writeDataToFile(fileName, preTripSurvey);

                System.out.println("Done writing preTripSurvey");

        // Prepare ViewController for next stage
        txtQTop.setText("Continue the survey when you arrive at the Bus Stop.");

        radioTime.setVisibility(View.INVISIBLE);
        pickTime.setVisibility(View.GONE);
        butSubmit.setVisibility(View.VISIBLE);
        countScrn = 0;
        timeLapsed = 0;

        // Set timer to begin GPS tracking at a reduced rate and keep Survey alive if phone goes to sleep
        scheduleAlarmForDate(surveyInterval);
        // Call Location Manager to monitor position at reduced rate. Was handled by GPSActions() in iPhone
                locMgr.requestLocationUpdates(LocationManager.GPS_PROVIDER, timeInterval * 5000, 5, locListener);

                preTripDone = true;
                System.out.println("preTripSurvey Done!");

        }
                mainView.startAnimation(animationFadeIn);

        }

        public void getSubmit() {
                // Check to see if new Survey, reset screen
                if (countScrn == 100) {
                        butSubmit.setText("Continue");
                        sliderMidVal.setText("");
                        preTripDone = false;
                        sliderAdjusted = false;
                    butEnd.setVisibility(View.GONE);
                        setupPurposeScreen();
                        return;
                }
                 // Disable GPS and Alarm while current Survey iteration in progress
                locMgr.removeUpdates(locListener);
                timeLapsed = 0;
```

```
        if (alarmOn == true) {
                cancelAlarm();
                }
        countScrn = countScrn +1;

        // Push questions and process answers for each Survey screen
        if (countScrn == 1) {
                countSurvey = countSurvey +1;
                // Initialize inTrip survey screen
                txtQTop.setText("How satisfied are you with the trip so far?");
txtQTop.invalidate();
                sliderMinVal.setText("Not at all");
                sliderMinVal.setVisibility(View.VISIBLE);
                sliderMaxVal.setText("Completelty");
                sliderMaxVal.setVisibility(View.VISIBLE);
                slideImp.setVisibility(View.VISIBLE);

                SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm");
                timeCurrent = timeFormat.format(new Date()).toString();
        Log.i("Current Time: ", timeCurrent);

        // Reset slider from previous question
        if (countSurvey == 1) {
        slideImp.setProgress(4);
        } else {
        slideImp.setProgress(slideSatCurrent - 1);
        butEnd.setVisibility(View.VISIBLE);
        }
    } else if (countScrn == 2) {
        // Record the satisfaction level and test delta (>=2/9)
        slideSatPrev = slideSatCurrent;
        slideSatCurrent = slideImp.getProgress() + 1;
        int slideDelta = slideSatPrev - slideSatCurrent;
        slideDelta = (slideDelta < 0) ? -slideDelta : slideDelta; // To get absolute value of the change in satisfaction
        if (slideDelta >= 2 || endFlag == true || countSurvey <= 2) {
                // Push supplemental questions
                mainView.setVisibility(View.INVISIBLE);
        mainView.invalidate();

                pushSupplemental();
        } else {
                mainView.startAnimation(animationFadeIn);
        }
        // Push next question and reset slider
        if (endFlag == true) {
                countScrn = countScrn + 1; //Arrival question will be skipped at end of survey
                txtQTop.setText("How confident were you that you'd arrive on time?");
                sliderMinVal.setText("Not at all");
                sliderMaxVal.setText("Completely");
                sliderMidVal.setVisibility(View.INVISIBLE);
                slideArrCurrent = -1;
                slideImp.setProgress(slideConfCurrent -1);

        } else {
                sliderMinVal.setText("Earlier");
                sliderMaxVal.setText("Later");
                sliderMidVal.setText("As Expected");
                sliderMidVal.setVisibility(View.VISIBLE);
```

```
                txtQTop.setText("When do you expect to arrive at your destination:");
                if (countSurvey == 1) {
                        slideSatFirst = slideSatCurrent;
                        slideImp.setProgress(4);
                } else if (countSurvey > 1) {
                        slideImp.setProgress(slideArrCurrent - 1);
                }
        }
        butEnd.setVisibility(View.GONE);


    } else if (countScrn == 3) {
        // Record the expected arrival performance
        slideArrPrev = slideArrCurrent;
        slideArrCurrent = slideImp.getProgress() + 1;

        //Push next question and reset slider
        txtQTop.setText("How confident are you in your expected arrival time?");
        if (endFlag == true) {
                txtQTop.setText("How confident were you that you'd arrive on time?");
        }
        sliderMinVal.setText("Not at all");
        sliderMaxVal.setText("Completely");
        sliderMidVal.setVisibility(View.INVISIBLE);
        if (countSurvey == 1){
                slideArrFirst = slideArrCurrent;
                slideImp.setProgress(4);
        } else if (countSurvey > 1) {
                slideImp.setProgress(slideConfCurrent -1);
        }
                mainView.startAnimation(animationFadeIn);
    } else if (countScrn == 4) {
        slideConfPrev = slideSatCurrent;
        slideConfCurrent = slideImp.getProgress() + 1;
        gpsFlag = true;

        // Write survey data to file
        // Prepare data string to write
        String inTripHeader = "User ID, Survey_Timestamp, Survey_Iteration, Iteration_Time, Satisfaction, Arrival_Perf,
Confidence\n";

        String inTripSurvey = userID + ", " + timeSurvey + ", " + countSurvey +
                        ", " + timeCurrent + ", " + slideSatCurrent + ", " + slideArrCurrent +
                        ", " + slideConfCurrent + "\n";

    fileName = userID + "_" + timePreSurvey + "_InTrip.txt";
    Log.i("InTrip fileName: ", appFolder+"/"+fileName);
    if (countSurvey == 1) { // Write header to file
        writeDataToFile(fileName, inTripHeader);
        }
    writeDataToFile(fileName, inTripSurvey);

         System.out.println("Done writing inTripSurvey " + countSurvey);


       if (endFlag == true) {
        // Segue to End of Survey View
                        mainView.setVisibility(View.INVISIBLE);
```

```
                mainView.invalidate();
            pushEndSurvey();
            } else {
            if (countSurvey == 1) {
                slideConfFirst = slideConfCurrent;
            }
            // Reset screen counter to repeat survey at next interval
            countScrn = 0;
            txtQTop.setText("Next survey in " + surveyInterval + " seconds, or press Continue for manual initiation");
            if (countSurvey == 1){
                txtQTop.setText("Continue the survey as soon as you board the bus.");
            }
            sliderMaxVal.setVisibility(View.INVISIBLE);
            sliderMinVal.setVisibility(View.INVISIBLE);
            sliderMidVal.setVisibility(View.INVISIBLE);
            slideImp.setVisibility(View.INVISIBLE);
            butSubmit.setText("Continue");

            // Set to throw Survey at interval via AlarmTimer or GPS?
            timeLapsed = 0;
            timerActions();
                if (countSurvey == 1) {
                    // At bus stop: Detect location change greater than 10 m to throw first on-bus survey
                    locMgr.requestLocationUpdates(LocationManager.GPS_PROVIDER,    timeInterval    *    2000,    10,
locListener);
                } else {
                    // On bus: Detect small location changes of 3 m or more and update at timeInterval seconds
                    locMgr.requestLocationUpdates(LocationManager.GPS_PROVIDER,    timeInterval    *    1000,    3,
locListener);
                }
                mainView.startAnimation(animationFadeIn);
            }
        }
    }

    public void showSurveyAlert(String title, String message) {
        if (alarmOn == true) {
            cancelAlarm();
        }

        Vibrator v = (Vibrator) getBaseContext().getSystemService(Context.VIBRATOR_SERVICE);
        v.vibrate(500);

        if (alertDisplayed == false){
            alertDisplayed = true;
            AlertDialog.Builder builder = new AlertDialog.Builder(FirstActivity.this);
            builder.setTitle(title)
                .setMessage(message)
                .setCancelable(false)
                .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                    timeLapsed = 0;
                    dialog.cancel();
                        butSubmit.setText("Submit");
                        mainView.invalidate();
                    alertDisplayed = false;
                    getSubmit();
                }
```

```
                                })
                                .setNegativeButton("No",  new DialogInterface.OnClickListener() {
                                        public void onClick(DialogInterface dialog, int id) {
                                        timeLapsed = surveyInterval/2;
                                        alertDisplayed = false;
                                        scheduleAlarmForDate(surveyInterval/2);
                                        dialog.cancel();
                                        // endProgram(); // Used for debugging only!
                                        }
                                });
                        AlertDialog alertDialog = builder.create();
                        alertDialog.show();
                }
        }

        public void timerActions() {

                timeLapsed = timeLapsed + timeInterval;
                timeNext = surveyInterval - timeLapsed;

                if (alarmOn == false) {scheduleAlarmForDate(surveyInterval);}
                if (countSurvey > 1) {
                        float countdown = surveyInterval/60;
                        txtQTop.setText("Next survey in " + Float.toString(countdown) + " minutes, or press Continue for manual
initiation.");
                mainView.invalidate();
                }

                Log.i("Time Lapsed: ", String.valueOf(timeLapsed));

                if (timeLapsed >= surveyInterval) {
                        timeLapsed = 0;

                        // Prompt for Survey if user is on the bus (countSurvey>0), otherwise use GPS movement to sense when
user leaves bus stop (countSurvey==0; handle in GPSActions)

                        if (countSurvey > 0) {
                                //Prompt user to take the survey

                                // Shows AlertDialog asking user to continue Survey. This function must be called from a UI
(user interface) event, so a button click is simulated.
                                butSubmit.post(new Runnable(){
                            @Override
                            public void run() {butSubmit.performClick();
                            }
                                });
                        }
                }
        }

        public void setupGPSListener() {
        locMgr = (LocationManager)this.getSystemService(Context.LOCATION_SERVICE);
        loc = locMgr.getLastKnownLocation(LocationManager.GPS_PROVIDER);
        //List<String> providerList = locMgr.getAllProviders(); //needed?
        locListener = new LocationListener() {
                public void onLocationChanged(Location location) {
                        if (location != null) {
                                gpsLat = location.getLatitude();
```

```
                                gpsLong = location.getLongitude();
                                gpsAlt = location.getAltitude();
                                gpsDir = location.getBearing();
                                gpsSpeed = location.getSpeed();
                                gpsTime = location.getTime();

                                GPSActions();

                                if (countSurvey == 1) {
                                        // Detected possible movement away from bus stop
                                        Vibrator v = (Vibrator) getBaseContext().getSystemService(Context.VIBRATOR_SERVICE);
                                    v.vibrate(500);
                                NotificationManager                 nm                  =                   (NotificationManager)
getBaseContext().getSystemService(Context.NOTIFICATION_SERVICE);
                                Notification notification = new Notification();
                                notification.sound = Uri.parse("android.resource://com.sampleandroidapp/raw/alarm");
                                nm.notify(0, notification);
                                        butSubmit.post(new Runnable(){
                                   @Override
                                   public void run() {butSubmit.performClick();
                                   }
                                        });
                                }
                        }
                }
                public void onProviderDisabled(String provider) {
                }
                public void onProviderEnabled(String provider) {
                }
                public void onStatusChanged(String provider, int status, Bundle extras) {
                }
        };
        }

        public void GPSActions() {
                Log.i("GPSActions called: ", (String.valueOf(gpsLat) + ", " + String.valueOf(gpsLong)));

        // Write GPS data to file
        // Prepare data string to write
        String inTripGPS = userID + ", " + timeSurvey + ", " + countSurvey +
                        ", " + timeCurrent + ", " + gpsLat + ", " + gpsLong +
                        ", " + gpsSpeed + ", " + gpsDir + ", " + gpsAcc + ", " + gpsTime + "\n";

       fileName = userID + "_" + timePreSurvey + "_GPS.txt";
       Log.i("GPS fileName: ", appFolder+"/"+fileName);

           FileWriter outputStream;
           File newFile;
           try {
                   newFile = new File(appFolder, fileName);
                           outputStream = new FileWriter(newFile, true);
                           if (!newFile.exists()) {
                                        newFile.createNewFile();
                                        outputStream.write("UserID,    Survey_Timestamp,    Survey_Iteration,    Iteration_Time,
Latitude, Longitude, Speed, Direction, Altitude, Accuracy, GPS_Time_GMT\n");
                                }

                   outputStream.write(inTripGPS);
```

```java
                outputStream.flush();
                outputStream.close();
                System.out.println("Done writing GPS " + countSurvey);
              }
        catch (IOException e) {
              e.printStackTrace();
              }
    }

    public void getEnd() {
            endFlag = true;
            getSubmit();
    }
            public void pushEndSurvey() {

            Intent myIntent3 = new Intent(this.getApplicationContext(), EndActivity.class);
            startActivityForResult(myIntent3, 0);

            prepareForStandby();
    }
    public void prepareForStandby() {
    countScrn = 100;
    countSurvey = 0;
            txtQTop.setText("Thanks! Press the START button when you are ready to begin your next trip.");
    sliderMaxVal.setVisibility(View.INVISIBLE);
    sliderMinVal.setVisibility(View.INVISIBLE);
    sliderMidVal.setVisibility(View.INVISIBLE);
    slideImp.setVisibility(View.INVISIBLE);
    butEnd.setText("Quit");
    butEnd.setVisibility(View.VISIBLE);
    butSubmit.setText("Start");
    mainView.invalidate();
    endFlag = false;

    if (alarmOn == true) { // Check to see if alarm is active and shut it down so it doesn't continue nagging the user
            cancelAlarm();
                    }
    }

    public void pushSupplemental() {
            Intent myIntent2 = new Intent(this.getApplicationContext(), InputActivity.class);
            startActivityForResult(myIntent2, 0);

            Intent myIntent1 = new Intent(this.getApplicationContext(), SupplementalActivity.class);
            startActivityForResult(myIntent1, 0);
    }

    public void scheduleAlarmForDate(int alarmInterval) {

    alertAlarm = (AlarmManager) getSystemService(ALARM_SERVICE);
    timerIntent = new Intent(this, TimeBroadcastReceiver.class);
    myPendingIntent          =PendingIntent.getBroadcast(this.getApplicationContext(),          0,          timerIntent,
PendingIntent.FLAG_CANCEL_CURRENT);

                    alertAlarm.setRepeating(AlarmManager.RTC_WAKEUP,  System.currentTimeMillis()  +  alarmInterval  *  1000,
alarmInterval * 1000, myPendingIntent);

            alarmOn = true;
```

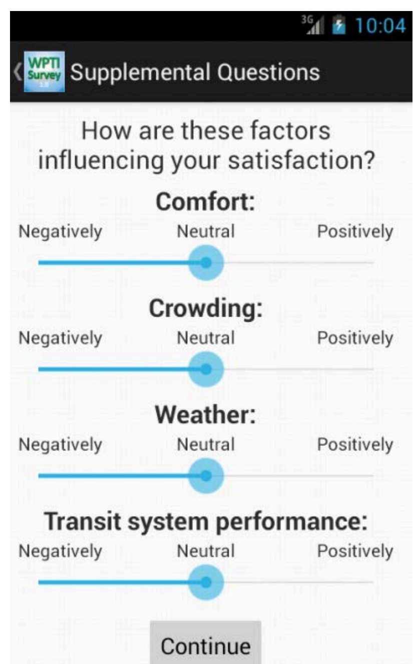
```
        }

        public void cancelAlarm() {
        alertAlarm.cancel(myPendingIntent);
        myPendingIntent.cancel();
        alarmOn = false;
        }

        private void checkExternalMedia() {
                String state = Environment.getExternalStorageState();
                if (Environment.MEDIA_MOUNTED.equals(state)) { // SD card available
               appFolder = Environment.getExternalStorageDirectory().toString() + "/WPTI";
                } else { // No external media, must write to private internal folder
               appFolder = getFilesDir().toString();
                        Toast.makeText(this, "External Storage is not available.", Toast.LENGTH_SHORT).show();
                }
       Log.d("appFolder: ", appFolder);
       new File(appFolder).mkdirs();
        }

    private void writeDataToFile(String theFile, String theData) {
                FileWriter outputStream;
           File newFile;
           try {
                  newFile = new File(appFolder, theFile);
                          outputStream = new FileWriter(newFile, true);
                          if (!newFile.exists()) {
                                         newFile.createNewFile();
                                 }
                  outputStream.write(theData);
                  outputStream.flush();
                  outputStream.close();
                }
           catch (IOException e) {
                e.printStackTrace();
                        Toast.makeText(this,   "There   was   a   problem   saving   the   survey   data   to   a   file.",
Toast.LENGTH_SHORT).show();
                }
    }

        public void endProgram() {
                FirstActivity.this.finish();
        }

    protected void onDestroy() {
       super.onDestroy();

       if (alarmOn == true) { // Check to see if alarm is active and attempt to shut it down so it doesn't continue nagging the
user
                alertAlarm.cancel(myPendingIntent);
                myPendingIntent.cancel();

                alarmOn = false;
       }
       System.exit(0);
    }
    protected void onStop() { // Another App comes into focus, such as call or SMS received, so set an Alarm to keep the Survey
alive in background
```

```
    super.onStop();

        if (countSurvey > 0 && alarmOn == false) {
                scheduleAlarmForDate(surveyInterval);
        }
    }
}
```

Supplemental Activity (User Interface)

The Supplemental survey prompts the user to rate the factors influencing their satisfaction. This screen is presented to the user at the beginning and end of the bus trip and whenever there is a significant change in satisfaction (+/- 1).

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SupplementalActivity" >

    <ScrollView
    android:id="@+id/scrollView2"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_marginRight="16dip"
            android:layout_marginTop="8dip"
            android:layout_marginLeft="8dip" >

    <TextView
        android:id="@+id/textQ3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:gravity="center|center_vertical"
        android:maxLines="3"
        android:text="@string/question_top2"
        android:textSize="20sp" />

     <TextView
        android:id="@+id/textFact1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/textQ3"
        android:layout_marginTop="5dp"
        android:text="Your Personal Comfort"
        android:textSize="20sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textSlide2_Min"
        android:layout_width="wrap_content"
```



```
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textFact1"
        android:text="@string/slideVal2_low"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <TextView
        android:id="@+id/textSlide2_High"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignBaseline="@+id/textSlide2_Min"
        android:layout_alignBottom="@+id/textSlide2_Min"
        android:text="@string/slideVal2_high"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <TextView
        android:id="@+id/textSlide2_Mid"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/textSlide2_Min"
        android:layout_alignBottom="@+id/textSlide2_Min"
        android:layout_centerHorizontal="true"
        android:text="@string/slideVal2_mid"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <SeekBar
        android:id="@+id/seekBar2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textSlide2_Min"
        android:max="8"
        android:progress="4" />

     <TextView
        android:id="@+id/textFact2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/seekBar2"
        android:layout_marginTop="5dp"
        android:text="Your Personal Wellbeing"
        android:textSize="20sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textSlide3_Min"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textFact2"
        android:text="@string/slideVal2_low"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <TextView
        android:id="@+id/textSlide3_High"
        android:layout_width="wrap_content"
```

```xml
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignBaseline="@+id/textSlide3_Min"
        android:layout_alignBottom="@+id/textSlide3_Min"
        android:text="@string/slideVal2_high"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <TextView
        android:id="@+id/textSlide3_Mid"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/textSlide3_Min"
        android:layout_alignBottom="@+id/textSlide3_Min"
        android:layout_centerHorizontal="true"
        android:text="@string/slideVal2_mid"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <SeekBar
        android:id="@+id/seekBar3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textSlide3_Min"
        android:max="8"
        android:progress="4" />

    <TextView
        android:id="@+id/textFact3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="5dp"
        android:layout_below="@+id/seekBar3"
        android:text="The Weather"
        android:textSize="20sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textSlide4_Min"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textFact3"
        android:text="@string/slideVal2_low"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <TextView
        android:id="@+id/textSlide4_High"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignBaseline="@+id/textSlide4_Min"
        android:layout_alignBottom="@+id/textSlide4_Min"
        android:text="@string/slideVal2_high"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <TextView
        android:id="@+id/textSlide4_Mid"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/textSlide4_Min"
        android:layout_alignBottom="@+id/textSlide4_Min"
        android:layout_centerHorizontal="true"
        android:text="@string/slideVal2_mid"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <SeekBar
        android:id="@+id/seekBar4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textSlide4_Min"
        android:max="8"
        android:progress="4" />

        <TextView
        android:id="@+id/textFact4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="5dp"
        android:layout_below="@+id/seekBar4"
        android:text="The Transit System"
        android:textSize="20sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textSlide5_Min"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textFact4"
        android:text="@string/slideVal2_low"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <TextView
        android:id="@+id/textSlide5_High"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignBaseline="@+id/textSlide5_Min"
        android:layout_alignBottom="@+id/textSlide5_Min"
        android:text="@string/slideVal2_high"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <TextView
        android:id="@+id/textSlide5_Mid"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/textSlide5_Min"
        android:layout_alignBottom="@+id/textSlide5_Min"
        android:layout_centerHorizontal="true"
        android:text="@string/slideVal2_mid"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <SeekBar
```

```
        android:id="@+id/seekBar5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textSlide5_Min"
        android:max="8"
        android:progress="4" />

        <Button
        android:id="@+id/butSubmitS"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/seekBar5"
        android:layout_marginTop="10dp"
        android:text="@string/submitButton" />

        </RelativeLayout>
    </ScrollView>
</RelativeLayout>
```

Supplemental Activity (Code)

The code senses changes in user input to ensure that the user has actually touched a slider instead of simply tapping the Continue button to skip the screen. Data is written to a text file along with the user ID and timestamp.

```
package com.wptiapp1;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import android.app.Activity;
import android.content.Intent;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import com.wptiapp1.R;

public class SupplementalActivity extends Activity implements OnSeekBarChangeListener{

        boolean sliderAdjusted = false;
        private SeekBar seekComfort;
        private SeekBar seekCrowding;
        private SeekBar seekWeather;
        private SeekBar seekPerformance;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_supplemental);

        Button activity2 = (Button) findViewById(R.id.butSubmitS);
        activity2.setOnClickListener(new View.OnClickListener() {
```

```
        public void onClick(View arg0) {Intent replyIntent = new Intent();

        // Write Supplemental data to file
        if (sliderAdjusted == true) {
                writeSupplementalData();

                // Close window and return to main Activity
                setResult(RESULT_OK, replyIntent);
                finish();
        } else {
                throwError();
        }
           }
           });
                seekComfort = (SeekBar) findViewById(R.id.seekBar2);
                seekCrowding = (SeekBar) findViewById(R.id.seekBar3);
                seekWeather = (SeekBar) findViewById(R.id.seekBar4);
                seekPerformance = (SeekBar) findViewById(R.id.seekBar5);

        seekComfort.setOnSeekBarChangeListener(this);
        seekCrowding.setOnSeekBarChangeListener(this);
        seekWeather.setOnSeekBarChangeListener(this);
        seekPerformance.setOnSeekBarChangeListener(this);
        }

    @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
        sliderAdjusted = true;
    }

        public void throwError() {
                Toast.makeText(this, "Please make a selection.", Toast.LENGTH_SHORT).show();
        }

        public void writeSupplementalData() {
                int sliderComfort = seekComfort.getProgress();
                int sliderCrowding = seekCrowding.getProgress();
                int sliderWeather = seekWeather.getProgress();
                int sliderPerformance = seekPerformance.getProgress();

        String supplementalData = FirstActivity.userID + ", " + FirstActivity.timeSurvey + ", " + FirstActivity.countSurvey +
                        ", " + FirstActivity.timeCurrent + ", " + sliderComfort + ", " + sliderCrowding +
                        ", " + sliderWeather + ", " + sliderPerformance + "\n";

    String fileName = FirstActivity.userID + "_" + FirstActivity.timePreSurvey + "_Supplemental.txt";
    Log.i("Supplemental fileName: ", FirstActivity.appFolder + fileName);

        FileWriter outputStream;
        File newFile;
        try {
          newFile = new File(FirstActivity.appFolder, fileName);
                          outputStream = new FileWriter(newFile, true);
                          if (!newFile.exists()) {
                                        newFile.createNewFile();
                                        outputStream.write("UserID,    Survey_Timestamp,    Survey_Iteration,    Iteration_Time,
Comfort, Wellbeing, Weather, Performance\n");
                                }
          outputStream.write(supplementalData);
```

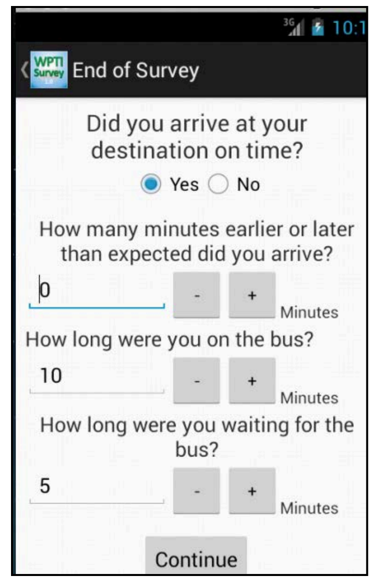
```
            outputStream.flush();
            outputStream.close();
            System.out.println("Done writing Supplemental" + FirstActivity.countSurvey);
          }
        catch (IOException e) {
         e.printStackTrace();
         }
         }

         @Override
         public void onProgressChanged(SeekBar arg0, int arg1, boolean arg2) {
                 // TODO Auto-generated method stub
         }

         @Override
         public void onStartTrackingTouch(SeekBar arg0) {
                 // TODO Auto-generated method stub
         }
}
```

End Activity (User Interface)

The End Activity is initiated after the user has tapped the End Survey button, which is part of the First Activity interface. The supplemental questions are thrown at the user, and then these final questions to assess the users' perception of how long the trip took and whether or not they actually arrived on time. The user interface is designed to accept time values, where the user can either enter a value manually or use up/down increments.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    tools:context=".EndActivity" >

    <ScrollView
    android:id="@+id/scrollView2"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_marginRight="16dip"
            android:layout_marginTop="8dip"
            android:layout_marginLeft="8dip" >

    <TextView
         android:id="@+id/textQend"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center|center_vertical"
        android:maxLines="2"
        android:textSize="20sp"
        android:text="@string/question_end" />

   <RadioGroup
```

```
    android:id="@+id/radioGroupE"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:layout_below="@+id/textQend"
    android:orientation="horizontal" >

    <RadioButton
        android:id="@+id/radioButtonE1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="Yes" />

    <RadioButton
        android:id="@+id/radioButtonE2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="No" />

</RadioGroup>

    <TextView
         android:id="@+id/textQend2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center|center_vertical"
        android:maxLines="2"
        android:layout_marginTop="10dip"
        android:textSize="18sp"
        android:layout_below="@+id/radioGroupE"
        android:text="How many minutes earlier or later than expected did you arrive?" />

    <Button
        android:id="@+id/butEQ2dn"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textQend2"
        android:layout_centerHorizontal="true"
        android:text="@string/minus" />

    <Button
        android:id="@+id/butEQ2up"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textQend2"
        android:layout_toRightOf="@+id/butEQ2dn"
        android:text="@string/plus" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/butEQ2up"
        android:layout_toRightOf="@+id/butEQ2up"
        android:text="@string/mins"
        android:textAppearance="?android:attr/textAppearanceSmall" />
```



```
<EditText
    android:id="@+id/editTxtQ2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textQend2"
    android:layout_toLeftOf="@+id/butEQ2dn"
    android:text="0"
    android:ems="10" >
</EditText>

<TextView
     android:id="@+id/textQend3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center|center_vertical"
    android:maxLines="2"
    android:layout_marginTop="10dip"
    android:textSize="18sp"
    android:layout_below="@+id/editTxtQ2"
    android:text="How long were you on the bus?" />

<Button
    android:id="@+id/butEQ3dn"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textQend3"
    android:layout_centerHorizontal="true"
    android:text="@string/minus" />

<Button
    android:id="@+id/butEQ3up"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textQend3"
    android:layout_toRightOf="@+id/butEQ3dn"
    android:text="@string/plus" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/butEQ3up"
    android:layout_toRightOf="@+id/butEQ3up"
    android:text="@string/mins"
    android:textAppearance="?android:attr/textAppearanceSmall" />

<EditText
    android:id="@+id/editTxtQ3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textQend3"
    android:layout_toLeftOf="@+id/butEQ3dn"
    android:text="10"
    android:ems="10" >
</EditText>
```

```xml
<TextView
     android:id="@+id/textQend4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center|center_vertical"
    android:maxLines="2"
    android:layout_marginTop="10dip"
    android:textSize="18sp"
    android:layout_below="@+id/editTxtQ3"
    android:text="How long were you waiting for the bus?" />

<Button
    android:id="@+id/butEQ4dn"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textQend4"
    android:layout_centerHorizontal="true"
    android:text="@string/minus" />

<Button
    android:id="@+id/butEQ4up"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textQend4"
    android:layout_toRightOf="@+id/butEQ3dn"
    android:text="@string/plus" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/butEQ4up"
    android:layout_toRightOf="@+id/butEQ4up"
    android:text="@string/mins"
    android:textAppearance="?android:attr/textAppearanceSmall" />

<EditText
    android:id="@+id/editTxtQ4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textQend4"
    android:text="5"
    android:layout_toLeftOf="@+id/butEQ4dn"
    android:ems="10" >
</EditText>

     <Button
    android:id="@+id/butSubmitE"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_below="@+id/editTxtQ4"
    android:layout_marginTop="20dp"
    android:text="@string/submitButton" />

    </RelativeLayout>
```

```
    </ScrollView>
</RelativeLayout>
```

End Activity (Code)

After the user taps the Continue button, the data is written to text files and the survey re-initiates itself so it is standing by for the next trip.

```
package com.wptiapp1;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.EditText;
import android.app.Activity;
import android.content.Intent;
import com.wptiapp1.R;

public class EndActivity extends Activity {

        String curValStr;
        int asExpected = 0;
        int curVal;
        RadioButton radioYes, radioNo;
        EditText txtQ2, txtQ3, txtQ4;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_end);

        Button activity4 = (Button) findViewById(R.id.butSubmitE);
        activity4.setOnClickListener(new View.OnClickListener() {
        public void onClick(View arg0) {Intent replyIntent = new Intent();

        // Write End of Survey data to file
        writeEndData();

        // Close window and return to main Activity
        setResult(RESULT_OK, replyIntent);
        FirstActivity.mainView.setVisibility(View.VISIBLE);
        FirstActivity.mainView.invalidate();

        finish();
            }
            });

        Button butQ2up = (Button) findViewById(R.id.butEQ2up);
        Button butQ2dn = (Button) findViewById(R.id.butEQ2dn);
        Button butQ3up = (Button) findViewById(R.id.butEQ3up);
        Button butQ3dn = (Button) findViewById(R.id.butEQ3dn);
        Button butQ4up = (Button) findViewById(R.id.butEQ4up);
```

```
Button butQ4dn = (Button) findViewById(R.id.butEQ4dn);
radioYes = (RadioButton) findViewById(R.id.radioButtonE1);
radioNo = (RadioButton) findViewById(R.id.radioButtonE2);
txtQ2 = (EditText) findViewById(R.id.editTxtQ2);
txtQ3 = (EditText) findViewById(R.id.editTxtQ3);
txtQ4 = (EditText) findViewById(R.id.editTxtQ4);

butQ2up.setOnClickListener(new View.OnClickListener() {
 public void onClick(View arg0) {
         //Button button = (Button)arg0;
        curValStr = txtQ2.getText().toString();
         //if (curValStr != "0") {curVal = Integer.parseInt(curValStr);}
         curVal = Integer.parseInt(curValStr);
         curVal = curVal + 1;
         txtQ2.setText(Integer.toString(curVal));
 }
});
butQ3up.setOnClickListener(new View.OnClickListener() {
 public void onClick(View arg0) {
         curValStr = txtQ3.getText().toString();
         curVal = Integer.parseInt(curValStr);
         curVal = curVal + 5;
         txtQ3.setText(Integer.toString(curVal));
 }
});
butQ4up.setOnClickListener(new View.OnClickListener() {
 public void onClick(View arg0) {
         curValStr = txtQ4.getText().toString();
         curVal = Integer.parseInt(curValStr);
         curVal = curVal + 1;
         txtQ4.setText(Integer.toString(curVal));
 }
});
butQ2dn.setOnClickListener(new View.OnClickListener() {
 public void onClick(View arg0) {
         curValStr = txtQ2.getText().toString();
         curVal = Integer.parseInt(curValStr);
         //if (curVal > 0) {curVal = curVal - 1;}
         curVal = curVal - 1;
         txtQ2.setText(Integer.toString(curVal));
 }
});
butQ3dn.setOnClickListener(new View.OnClickListener() {
 public void onClick(View arg0) {
         curValStr = txtQ3.getText().toString();
         curVal = Integer.parseInt(curValStr);
         if (curVal > 0) {curVal = curVal - 5;}
         txtQ3.setText(Integer.toString(curVal));
 }
});
butQ4dn.setOnClickListener(new View.OnClickListener() {
 public void onClick(View arg0) {
         curValStr = txtQ4.getText().toString();
         curVal = Integer.parseInt(curValStr);
         if (curVal > 0) {curVal = curVal - 1;}
         txtQ4.setText(Integer.toString(curVal));
 }
});
```

```
        }

        public void writeEndData() {

                if (radioYes.isChecked()) {
                        asExpected = 1;
                }

        String endData = "UserID = " + FirstActivity.userID + "\nSurvey_Timestamp = " + FirstActivity.timeSurvey + "\nAsExpected 
(N/Y 0/1) = " + asExpected +
                        "\nDestination Arrival = " + txtQ2.getText() + "\nTime On Bus = " + txtQ3.getText() + "\nTime Waiting = 
" + txtQ4.getText() + "\n";

        String fileName = FirstActivity.userID + "_" + FirstActivity.timePreSurvey + "_End.txt";
        Log.i("Supplemental fileName: ", fileName);

            FileWriter outputStream;
            File newFile;
            try {
                  newFile = new File(FirstActivity.appFolder, fileName);
                          outputStream = new FileWriter(newFile, true);
                          if (!newFile.exists()) {
                                          newFile.createNewFile();
                                          outputStream.write("UserID, Survey_Timestamp, ...\n");
                                }
                  outputStream.write(endData);
                  outputStream.flush();
                  outputStream.close();
                  System.out.println("Done writing End data" + FirstActivity.countSurvey);
                }
            catch (IOException e) {
                e.printStackTrace();
                }
        }
}
```

Settings Activity

The Settings screen allows the user to change the interval of the survey and upload their data to the researchers via an email attachment.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mainView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    tools:context=".FirstActivity" >

    <ScrollView
    android:id="@+id/scrollView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```
        android:orientation="vertical"
        android:layout_marginRight="16dip"
        android:layout_marginTop="8dip"
        android:layout_marginLeft="8dip" >

<TextView
    android:id="@+id/textQ1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:gravity="center|center_vertical"
    android:maxLines="4"
    android:text="@string/question_top"
    android:textSize="20sp"
    android:textStyle="bold" />

<TextView
    android:id="@+id/textSlide1_Min"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/textQ1"
    android:gravity="left"
    android:text="@string/slideVal1_low"
    android:textAppearance="?android:attr/textAppearanceSmall" />

<TextView
    android:id="@+id/textSlide1_High"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
             android:layout_below="@+id/textQ1"
             android:gravity="right"
    android:text="@string/slideVal1_high"
    android:textAppearance="?android:attr/textAppearanceSmall" />

<TextView
    android:id="@+id/textSlide1_Mid"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/textSlide1_Min"
    android:layout_alignBottom="@+id/textSlide1_Min"
    android:gravity="center"
    android:layout_centerHorizontal="true"
    android:text="@string/slideVal1_mid"
    android:textAppearance="?android:attr/textAppearanceSmall" />

<SeekBar
    android:id="@+id/seekBar1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/textSlide1_Min"
    android:max="8"
    android:progress="4" />

        <requestFocus />
```

```
<TextView
    android:id="@+id/textLabel_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@+id/textSlide1_Mid"
    android:layout_below="@+id/seekBar1"
    android:layout_marginTop="16dp"
    android:text="@string/txtLabel_id"
    android:textAppearance="?android:attr/textAppearanceLarge" />

<TextView
    android:id="@+id/textQ2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_below="@+id/editText_UserID"
    android:gravity="center|center_vertical"
    android:layout_marginTop="16dp"
    android:maxLines="2"
    android:textSize="20sp"
    android:textStyle="bold"
    android:text="@string/question_bot" />

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textQ2"
    android:layout_alignRight="@+id/textSlide1_Mid"
    android:layout_marginTop="8dp"
    android:text="@string/butPurp1" />

<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@+id/button1"
    android:layout_alignTop="@+id/button1"
    android:layout_marginLeft="8dp"
    android:text="@string/butPurp2" />

<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@+id/textSlide1_Mid"
    android:layout_below="@+id/button1"
    android:layout_marginTop="8dp"
    android:text="@string/butPurp3" />

<Button
    android:id="@+id/button4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@+id/button3"
    android:layout_alignTop="@+id/button3"
    android:layout_marginLeft="8dp"
```

```xml
            android:text="@string/butPurp4" />

        <EditText
            android:id="@+id/editText_UserID"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignBaseline="@+id/textLabel_id"
            android:layout_alignBottom="@+id/textLabel_id"
            android:layout_toRightOf="@+id/textLabel_id"
            android:ems="10"
            android:inputType="text" >
            <requestFocus />
        </EditText>

        <TimePicker
            android:id="@+id/timePicker1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/textQ1"
            android:paddingBottom="20dp"
            android:layout_centerHorizontal="true"
            android:visibility="invisible" />

    <RadioGroup
        android:id="@+id/radioGroup1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:layout_below="@+id/timePicker1"
        android:orientation="vertical"
        android:visibility="invisible" />

        <RadioButton
            android:id="@+id/radioButton1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/radio1" />

        <RadioButton
            android:id="@+id/radioButton2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:checked="true"
            android:text="@string/radio2" />

        <RadioButton
            android:id="@+id/radioButton3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/radio3" />
    </RadioGroup>

        <Button
            android:id="@+id/butSubmit"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_below="@+id/seekBar1"
```

```
        android:layout_marginTop="20dp"
        android:text="@string/submitButton"
        android:visibility="gone" />

    <Button
        android:id="@+id/butEnd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/butSubmit"
        android:layout_marginTop="20dp"
        android:text="@string/endButton"
        android:visibility="gone" />

        </RelativeLayout>
    </ScrollView>

</RelativeLayout>
```

Settings Activity (Code)

When the user taps the Upload button, the app collects all of the text data files and creates an email message with the files attached.

```
package com.wptiapp1;

import java.io.File;
import java.util.ArrayList;
import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Button;
import android.widget.Toast;

public class SettingsActivity extends Activity {

        EditText setID, setInt;
        Button setOK, setCancel, setUpload;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_settings);

                setID = (EditText)findViewById(R.id.editTextSet_UserID);
                setInt = (EditText)findViewById(R.id.editTextSet_interval);
                setOK = (Button)findViewById(R.id.butSet_ok);
                setCancel = (Button)findViewById(R.id.butSet_cancel);
                setUpload = (Button)findViewById(R.id.butSet_upload);

                setID.setText(FirstActivity.userID);
                String intervalStr = String.valueOf(FirstActivity.surveyInterval);
```

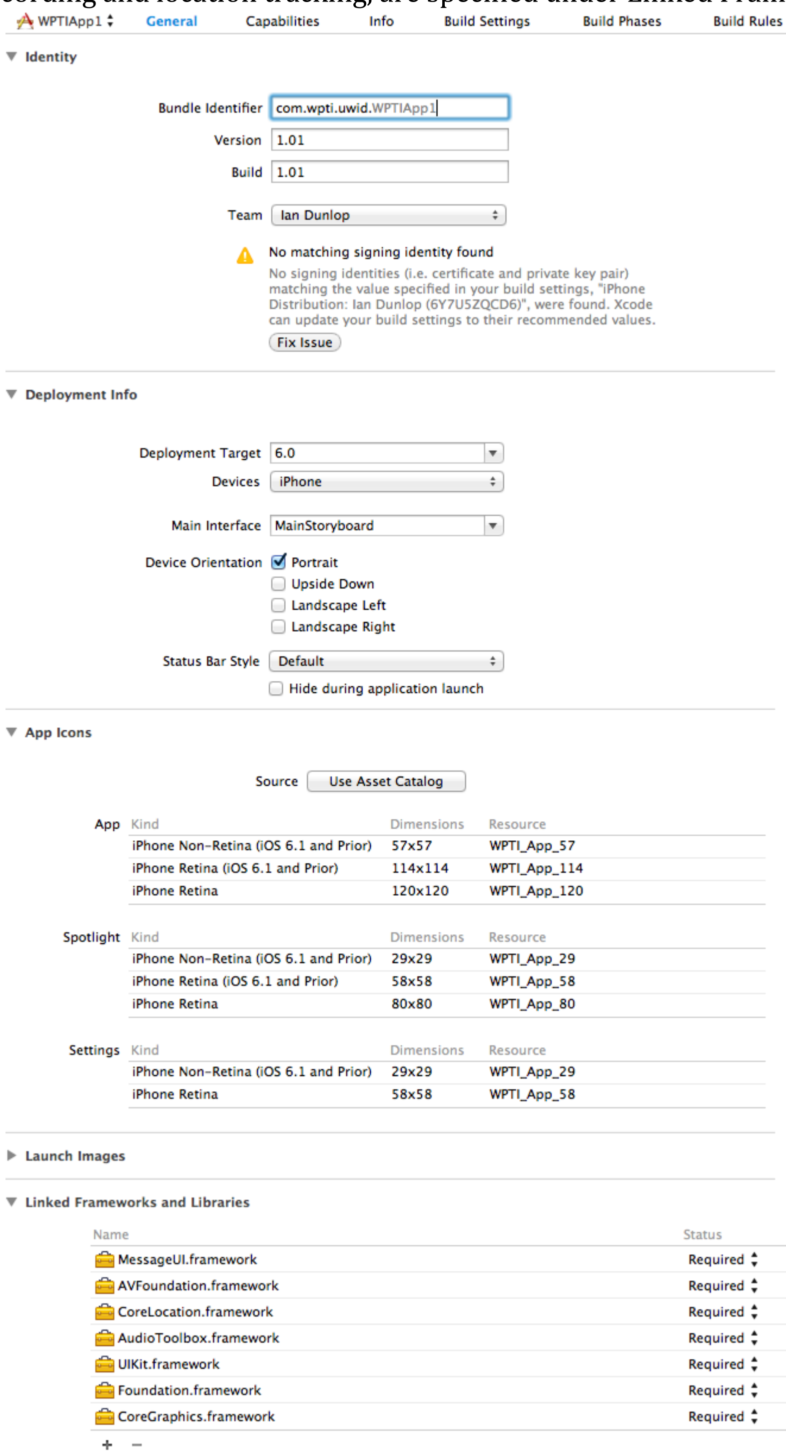
```
                float interval = Float.valueOf(intervalStr);
                //interval = interval / 60;
                setInt.setText(Float.toString(interval));

                setOK.setOnClickListener(new View.OnClickListener() {
                public void onClick(View arg0) {
                        Log.d(setID.getText().toString(), setInt.getText().toString());
                        String id2 = setID.getText().toString();
                        FirstActivity.userID = id2;
                        FirstActivity.theID.setText(id2);
                        try {
                        String intervalStr2 = setInt.getText().toString();
                        Float f = Float.valueOf(intervalStr2);
                        //f = f * 60;
                        FirstActivity.surveyInterval = f.intValue(); } catch (Exception e) {
                                Toast.makeText(arg0.getContext(),     "Please     enter     a     valid     time     interval.",
Toast.LENGTH_LONG).show();
                                return;
                        }
                        finish();
                }
                });

                setCancel.setOnClickListener(new View.OnClickListener() {
                public void onClick(View arg0) {
                  finish();
                }
                });

                setUpload.setOnClickListener(new View.OnClickListener() {
                public void onClick(View arg0) {
            try {
                 String email = "wptiapp@strategicinterchange.ca";
                 String subject = "WPTI Survey Submission";
                 String message = "The WPTI Survey App files are attached for UserID " + FirstActivity.userID;

                ArrayList<Uri> imageUris = new ArrayList<Uri>();

                 File file = new File( FirstActivity.appFolder ) ;
                 File list[] = file.listFiles();
                 if (list.length > 0) {
                                for (int i = 0; i < list.length; i++) {
                                        Log.d("File: ", list[i].getName());
                                        imageUris.add(Uri.parse("file://"     +     FirstActivity.appFolder     +     "/"     +
list[i].getName()));
                                }
                                }else {
                        Toast.makeText(arg0.getContext(), "No data files were found!", Toast.LENGTH_LONG).show();
                 }

                 final Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND_MULTIPLE);
                 emailIntent.setType("text/plain");
                 emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL, new String[] { email });
                 emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, subject);

                emailIntent.putParcelableArrayListExtra(Intent.EXTRA_STREAM, imageUris);
                emailIntent.putExtra(android.content.Intent.EXTRA_TEXT, message);
                arg0.getContext().startActivity(Intent.createChooser(emailIntent, "Send email with..."));
```

```
            } catch (Throwable t) {Toast.makeText(arg0.getContext(), "Upload failed try again: " + t.toString(),
                        Toast.LENGTH_LONG).show();}
               }
              });
        }
}
```

Appendix D: WPTI SurveyApp – i-Phone Version Code

General Parameters

The general parameters of the app's environment, similar to the Android Manifest properties, are modified through this interface. Unlike Android, the corresponding code is not available for direct editing. Access to the iPhone system functions, such as audio recording and location tracking, are specified under Linked Frameworks and Libraries.



Main and App Delegate

The Main and App Delegate are code that initialize the app. Some of this code is auto-generated, but needs to be checked and modified for the app to behave properly.

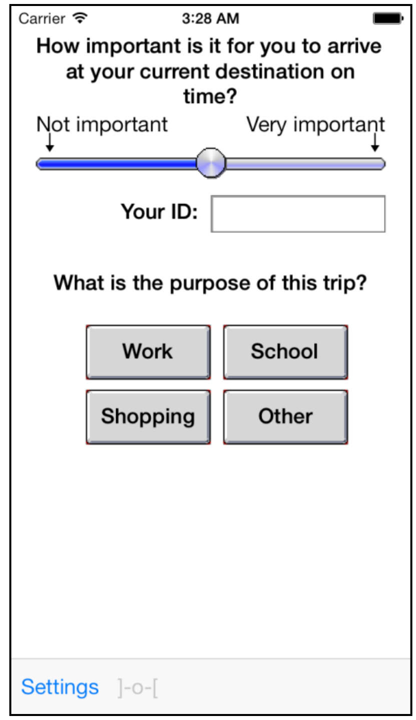
```
//  main.m
//  wptiApp
//
//  Created by Ian Dunlop on 2013-09-12.
//  Copyright (c) 2013 Ian Dunlop. All rights reserved.
//
#import <UIKit/UIKit.h>
#import "AppDelegate.h"
int main(int argc, char *argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));
    }
}
//  AppDelegate.m
//  wptiApp
//
//  Created by Ian Dunlop on 2013-09-12.
//  Copyright (c) 2013 Ian Dunlop. All rights reserved.
//
#import "AppDelegate.h"
@implementation AppDelegate
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Override point for customization after application launch.
    UIImage *minImage = [[UIImage imageNamed:@"sliderValue.png"] resizableImageWithCapInsets:UIEdgeInsetsMake(0, 4, 0, 4)];
    UIImage *maxImage = [UIImage imageNamed:@"sliderBackground.png"];
    UIImage *thumbImage = [UIImage imageNamed:@"sliderButton.png"];
    [[UISlider appearance] setMaximumTrackImage:maxImage forState:UIControlStateNormal];
    [[UISlider appearance] setMinimumTrackImage:minImage forState:UIControlStateNormal];
    [[UISlider appearance] setThumbImage:thumbImage forState:UIControlStateNormal];
    [[UISlider appearance] setThumbImage:thumbImage forState:UIControlStateHighlighted];
    [[UIApplication        sharedApplication]        registerForRemoteNotificationTypes:(UIRemoteNotificationTypeBadge         |
UIRemoteNotificationTypeSound)];
    return YES;
}
- (void)application:(UIApplication *)app didFailToRegisterForRemoteNotificationsWithError:(NSError *)err {
    NSLog(@"Error in registration. Error: %@", err);
}
@end
```

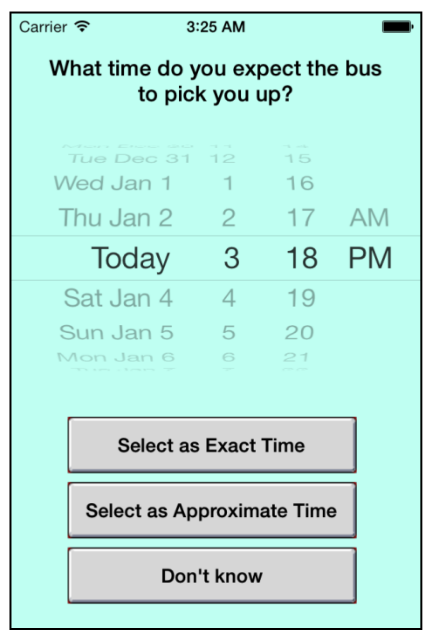
View Controller

The ViewController file is similar to the FirstActivity in Android. It is the first user interface to be initialized. ViewController.h declares the user interface properties and connections between variables and user interface components of mainView. ViewController.m is the XCode that executes behind the scenes when the user interacts with the interface or when certain events (timers, location prompts) occur. ViewController is a component of the Main Storyboard for the app, which interfaces with other view controllers, just as activities interface with each other in the Android development environment. ViewController is a dynamic interface that changes based on user input and the current stage of the survey.

ViewController.h

```
//  ViewController.h
//  wptiApp
//
//  Created by Ian Dunlop on 2013-09-12.
//  Copyright (c) 2013 Ian Dunlop. All rights reserved.

#import <UIKit/UIKit.h>
#import <CoreLocation/CoreLocation.h>
#import <AudioToolbox/AudioServices.h>
#import "ViewControllerInTrip.h"
#import "ViewControllerTextInput.h"
#import "ViewControllerEndSurvey.h"
#import "ViewControllerSetting.h"

int  countScrn = 0;
int  countSurvey = 0;
int  timeInterval = 3;
int  flagPickUp;
int  flagDest;
int  flagDropOff;
int  flagButPress;
int  timeLapsed;
int  slideImportant;
bool sliderAdjusted = FALSE;
bool endFlag = FALSE;
bool gpsFlag = TRUE;
bool onBus = FALSE;
bool alertDisplayed = FALSE;
float slideSatCurrent;
float slideSatPrev;
float slideSatFirst;
float slideArrCurrent;
float slideArrPrev;
float slideArrFirst;
float slideConfCurrent;
float slideConfPrev;
float slideConfFirst;
double gpsLat;
double gpsLong;
float gpsAlt;
float gpsDir;
float gpsAcc;
int gpsSpeed;

@interface ViewController : UIViewController <CLLocationManagerDelegate>
{
    NSTimer *timer;
}
@property int surveyInterval;
@property(nonatomic, retain) NSString *userID;
@property(nonatomic, retain) IBOutlet UIDatePicker *myDate;
@property(nonatomic, retain) IBOutlet UILabel *myMessage;
@property(nonatomic, retain) IBOutlet UILabel *yourID;
```





```
@property(nonatomic, retain) IBOutlet UITextField *theID;
@property(nonatomic, retain) IBOutlet UIButton *butTimeExact;
@property(nonatomic, retain) IBOutlet UIButton *butTimeApprox;
@property(nonatomic, retain) IBOutlet UIButton *butTimeDuno;
@property(nonatomic, retain) IBOutlet UIButton *butWork;
@property(nonatomic, retain) IBOutlet UIButton *butSchool;
@property(nonatomic, retain) IBOutlet UIButton *butShop;
@property(nonatomic, retain) IBOutlet UIButton *butOther;
@property(nonatomic, retain) IBOutlet UIButton *butSubmit;
@property(nonatomic, retain) IBOutlet UIButton *butEnd;
@property(nonatomic, retain) IBOutlet UIDatePicker *pickTime;
@property(nonatomic, retain) IBOutlet UISlider *slideImp;
@property(nonatomic, retain) IBOutlet UILabel *myPurpose;
@property(nonatomic, retain) IBOutlet UILabel *label1;
@property(nonatomic, retain) IBOutlet UILabel *label2;
@property(nonatomic, retain) IBOutlet UILabel *label3;
@property(nonatomic, retain) IBOutlet UILabel *label4;
@property(nonatomic, retain) IBOutlet UILabel *label5;
@property(nonatomic, retain) IBOutlet UIBarButtonItem *gpsStatus;
@property(nonatomic, retain) IBOutlet UIToolbar *toolbar;
@property(nonatomic, retain) CLLocation *locBusStop;
@property(nonatomic, retain) CLLocation *locCurrent;
-(IBAction)getDate:(id)sender;
-(IBAction)getImportnce:(id)sender;
-(IBAction)getPurpose:(id)sender;
-(IBAction)getSubmit:(id)sender;
-(IBAction)getEnd:(id)sender;
-(IBAction)dismissKeyboardOnTap:(id)sender;
-(void)timerActions;
-(void)GPSActions;
@end
```

ViewController.m

```
//  ViewController.m
//  wptiApp
//
//  Created by Ian Dunlop on 2013-09-12.
//  Copyright (c) 2013 Ian Dunlop. All rights reserved.

#import "ViewController.h"

@interface ViewController ()
@end
@implementation ViewController {
    CLLocationManager *locationManager;
}
@synthesize myDate, myMessage, butTimeExact, butTimeApprox, butTimeDuno, butWork, pickTime, slideImp, myPurpose, label1, label2,
label3, label4, label5, butOther, butSchool, butShop, butSubmit, butEnd, yourID, theID, locBusStop, locCurrent, gpsStatus,
toolbar, surveyInterval, userID;

NSDate *currentTime;
NSDate *gpsTime;
NSString *timePickUp;
NSString *timeDest;
NSString *timeDropOff;
```

```
NSString *timeSurvey;
NSString *timePreSurvey;
NSString *timeCurrent;
NSString *butPress;
NSString *tripPurpose;
NSString *txtInput;
NSString *fileName;
NSString *appFolder;
NSString *fileNameGPS;
SystemSoundID sound1;
- (void)viewDidLoad
{
    [super viewDidLoad];
      // Do any additional setup after loading the view, typically from a nib.
    surveyInterval = 300;
    locationManager = [[CLLocationManager alloc] init];
    locationManager.delegate = self;
    locationManager.desiredAccuracy = kCLLocationAccuracyBestForNavigation;
    [locationManager startUpdatingLocation];

    if ([[locationManager class] authorizationStatus] == kCLAuthorizationStatusDenied || [[locationManager class]
authorizationStatus] == kCLAuthorizationStatusNotDetermined) {
        NSLog(@"No location permission");
    }
    [self setSurveyTimestamp];
    NSURL *soundURL = [[NSBundle mainBundle] URLForResource:@"alarm"
                                              withExtension:@"aif"];
    AudioServicesCreateSystemSoundID((__bridge CFURLRef)soundURL, &sound1);
}
-(void)setSurveyTimestamp
{
    currentTime = [NSDate date];
    NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
    [dateFormatter setDateFormat:@"dd-MM_hh-mm"];
    timePreSurvey = [dateFormatter stringFromDate: currentTime];
    NSLog(@"Time pre-trip survey initiated %@", timePreSurvey);
}
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
}
-(IBAction)getPurpose:(id)sender
{
    if (sliderAdjusted == FALSE)
    {
        UIAlertView *alert = [[UIAlertView alloc]

                              initWithTitle:@"WPTI Survey"
                              message:@"Please select importance."
                              delegate:self
                              cancelButtonTitle:@"OK"
                              otherButtonTitles:nil];
        alert.transform = CGAffineTransformIdentity;
```

```
        alert.transform = CGAffineTransformTranslate(alert.transform, 0.0, -100.0);

        [alert show];    }
    if ([theID.text length] == 0)
    {
        UIAlertView *alert = [[UIAlertView alloc]

                              initWithTitle:@"WPTI Survey"
                              message:@"Enter your user ID."
                              delegate:self
                              cancelButtonTitle:@"OK"
                              otherButtonTitles:nil];
        alert.transform = CGAffineTransformIdentity;
        alert.transform = CGAffineTransformTranslate(alert.transform, 0.0, -100.0);

        [alert show];
    }
        if (sliderAdjusted == TRUE && [theID.text length] > 0)
    {
        [self setSurveyTimestamp];

        UIButton *button = (UIButton*)sender;
        butPress = button.titleLabel.text;

        tripPurpose = butPress;
        NSLog(@"tripPurpose = %@", tripPurpose);
        myMessage.text = tripPurpose;

        userID = theID.text;

        // Set up interface for time input screens
        butWork.hidden = true;
        butShop.hidden=true;
        butSchool.hidden=true;
        butOther.hidden=true;
        slideImp.hidden =true;
        myPurpose.hidden = true;
        label1.hidden=true;
        label2.hidden=true;
        label3.hidden=true;
        label4.hidden=true;
        yourID.hidden = true;
        theID.hidden = true;
        myMessage.text = @"What time do you expect the bus to pick you up?";
        butTimeDuno.hidden = false;
        butTimeExact.hidden = false;
        butTimeApprox.hidden = false;
        pickTime.hidden = false;
        toolbar.hidden = true;

        UIColor *colour = [[UIColor alloc]initWithRed:.75 green:1 blue:.95 alpha:1.0];
        self.view.backgroundColor = colour;
    }
}

-(void)setupPurposeScreen
{
    butSubmit.hidden = true;
```

```
    myMessage.text = @"How important is it for you to arrive at your current destination on time?";
    label1.text = @"Not Important";
    label3.text = @"Very Important";
    slideImp.value = 5;
    butWork.hidden = false;
    butShop.hidden=false;
    butSchool.hidden=false;
    butOther.hidden=false;
    slideImp.hidden =false;
    myPurpose.hidden = false;
    label1.hidden=false;
    label2.hidden=false;
    label3.hidden=false;
    label4.hidden=false;
    yourID.hidden = false;
    theID.hidden = false;
    sliderAdjusted = FALSE;
    countScrn = 0;
    toolbar.hidden = false;
}
-(IBAction)changeSettings:(id)sender
{
    [self performSegueWithIdentifier:@"segueToSettings" sender:self];
}
-(IBAction)getImportnce:(id)sender
{
    slideImportant = (slideImp.value+0.5);
    slideImp.value = slideImportant;
    sliderAdjusted = TRUE;
}
-(IBAction)getDate:(id)sender
{
    currentTime = [NSDate date];
    NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
    [dateFormatter setDateFormat:@"HH:mm"];
    timeSurvey = [dateFormatter stringFromDate: currentTime];
    NSLog(@"Current time %@", timeSurvey);
    NSDateFormatter *dateFormat = [ [NSDateFormatter alloc] init];
    [dateFormat setDateFormat:@"EEEE, MMMM, d, YYYY, HH:mm"];
    NSString *dateString = [dateFormat stringFromDate:myDate.date];
    UIButton *button = (UIButton*)sender;
    butPress = button.titleLabel.text;
    // Increment suevey screen counter
    countScrn = countScrn + 1;
    // Button selection flags
        if ([butPress isEqualToString: @"Select as Exact Time"])
        {
            flagButPress = 0;
        }
        else if ([butPress isEqualToString: @"Select as Approximate Time"])
        {
            flagButPress = 1;
        }
        else if ([butPress isEqualToString: @"Don't know"])
        {
            flagButPress = 2;
        }
    NSLog(@"Button pressed = %@", butPress);
```

```
    // Assign values based on selection and survey screen
    // Trip Purpose
    if (countScrn == 1)
    {
        timePickUp = dateString;
        flagPickUp = flagButPress;
            myMessage.text = @"At what time do you need to be at your destination?";
                NSLog(@"timePickUp = %@", timePickUp);
        UIColor *colour = [[UIColor alloc]initWithRed:.95 green:.75 blue:1 alpha:1.0];
        self.view.backgroundColor = colour;
    }
    else if (countScrn == 2)
    {
        timeDest = dateString;
        flagDest = flagButPress;
            myMessage.text = @"What time is the bus scheduled to arrive at the bus stop closest to your destination?";
                NSLog(@"timeDestin = %@", timeDest);
        UIColor *colour = [[UIColor alloc]initWithRed:.85 green:1 blue:.75 alpha:1.0];
        self.view.backgroundColor = colour;
    }
    else if (countScrn == 3)
    {
        timeDropOff = dateString;
        flagDropOff = flagButPress;
                NSLog(@"timeDropOff = %@", timeDropOff);
        UIColor *colour = [[UIColor alloc]initWithRed:1 green:1 blue:1 alpha:1.0];
        self.view.backgroundColor = colour;
    // Write the PreTripSurvey data to a file.
        // Prepare output strings
        NSString *preTripSurvey = [NSString stringWithFormat:@"UserID = %@\nSurvey_Timestamp = %@\nImportance = %d\nTripPurpose = 
%@\nPickUpTime = %@, %i\nDestTime = %@, %i\nDropOffTime = %@, %i", userID, timeSurvey, slideImportant, tripPurpose, timePickUp, 
flagPickUp, timeDest, flagDest, timeDropOff, flagDropOff];
        // Get the documents directory:
        NSArray *paths = NSSearchPathForDirectoriesInDomains
        (NSDocumentDirectory, NSUserDomainMask, YES);
        NSString *documentsDirectory = [paths objectAtIndex:0];
        // Check for WPTI app folder or create if none
        NSError *error = nil;
        NSFileManager *fileManager = [NSFileManager defaultManager];
        appFolder = [documentsDirectory stringByAppendingPathComponent:@"WPTI"];
        BOOL isDir;
        BOOL isFile = [[NSFileManager defaultManager] fileExistsAtPath:appFolder isDirectory:&isDir];
        NSLog(@"App Folder = %@", appFolder);
        if(isFile)
        {
            if (isDir) {
                NSLog(@"Folder already exists... (no action taken)");
            }
            else
            {
                NSLog(@"Problem: folder already exists as a file!");
            }
        }
        else
        {
            // WPTI Folder doesn't exist, so create it
            BOOL    success    =    [fileManager    createDirectoryAtPath:appFolder    withIntermediateDirectories:NO    attributes:nil 
error:&error];
```

```
            if (!success || error) {
                NSLog(@"Error: %@", [error localizedDescription]);
                UIAlertView *alert = [[UIAlertView alloc]

                                      initWithTitle:@"WPTI Survey"
                                      message:@"There may be a problem saving data to your phone. Contact tech support."
                                      delegate:self
                                      cancelButtonTitle:@"OK"
                                      otherButtonTitles:nil];
                alert.transform = CGAffineTransformIdentity;
                alert.transform = CGAffineTransformTranslate(alert.transform, 0.0, -100.0);

                [alert show];
            }
        }
        // Make a file name to write the data to using the documents directory:
        fileName = [NSString stringWithFormat:@"%@/%@_%@_PreTrip.txt",
                      appFolder, userID, timePreSurvey];
        NSLog(@"Pretrip Filename = %@", fileName);

        // Create content
        NSString *content = preTripSurvey;
        // Save content to the documents directory
        [content writeToFile:fileName
                  atomically:NO
                    encoding:NSStringEncodingConversionAllowLossy
                       error:nil];

    // Prepare ViewController for next stage
        myMessage.text = @"Continue the Survey when you reach the Bus Stop.";
        butTimeDuno.hidden = true;
        butTimeExact.hidden = true;
        butTimeApprox.hidden = true;
        pickTime.hidden = true;
        butSubmit.hidden = false;
        countScrn = 0;
        toolbar.hidden = false;
        timeCurrent = [dateFormatter stringFromDate: currentTime];

    // Set to start survey upon sensing departure from bus stop via GPS
        timer  =  [NSTimer  scheduledTimerWithTimeInterval:timeInterval  target:self  selector:@selector(timerActions)  userInfo:nil
repeats:YES];
        [self GPSActions];
     }
}

-(IBAction)getSubmit:(id)sender
{
    timeLapsed = 0;
    if (countScrn == 100)
    {
        butSubmit.titleLabel.text = @"Continue";
        [self setupPurposeScreen];
    }
    else
    {
        [timer invalidate];;
        //[locationManager stopUpdatingLocation];
```

```
countScrn = countScrn + 1;
    gpsStatus.enabled = false;
    toolbar.hidden = true;
if (countScrn == 1)
{
    countSurvey = countSurvey + 1;

    // Initialize inTrip survey screen
    myMessage.text = @"What is your current satisfaction with the trip so far?";
    label1.text = @"Not at all";
    label3.text = @"Completely";
    label1.hidden = false;
    label3.hidden = false;
    label2.hidden = false;
    label4.hidden = false;
    slideImp.hidden = false;

    currentTime = [NSDate date];
    NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
    [dateFormatter setDateFormat:@"HH:mm"];
    timeCurrent = [dateFormatter stringFromDate: currentTime];
    NSLog(@"Current time %@", timeCurrent);

    //Reset slider from previous question value
    if (countSurvey == 1)
    {
        slideImp.value = 5;
        slideSatCurrent = 5;
    }
    else if (countSurvey >1)
    {
        slideImp.value = slideSatCurrent;
        butEnd.hidden = false;
    }
    //AudioServicesPlayAlertSound(kSystemSoundID_Vibrate);
}
else if (countScrn == 2)
{
    // Record the satisfaction level and test delta (>=2/9)
    slideSatPrev = slideSatCurrent;
    slideSatCurrent = slideImp.value;
    int slideDelta = abs((slideSatPrev - slideSatCurrent));

    if (slideDelta >= 2 || endFlag == TRUE || countSurvey <= 2)
    {
        // Throw supplemental questions
        [self throwSupplemental];
    }

    // Throw next question and reset slider
    if (endFlag == TRUE) {
        countScrn = countScrn + 1; //Arrival question will be skipped at end of survey
        myMessage.text=@"How confident were you that you'd arrive on time?";
        label1.text=@"Not at all";
        label3.text=@"Completely";
        label5.hidden = true;
        slideArrCurrent = -1;
        slideImp.value = slideConfCurrent;
```

```
        } else {
            label1.text = @"Earlier";
            label3.text = @"Later";
            label5.hidden = false;
            myMessage.text = @"When do you expect to arrive at the destination?";
            if (countSurvey == 1)
            {
                slideSatFirst = slideSatCurrent;
                slideImp.value = 5;
            }
            else if (countSurvey >1)
            {
                slideImp.value = slideArrCurrent;
            }
        }
        butEnd.hidden = true;
    }
    else if (countScrn == 3)
    {
        // Record the expected arrival status
        slideArrPrev = slideArrCurrent;
        slideArrCurrent = slideImp.value;

        // Throw next question and reset slider
        myMessage.text = @"How confident are you in your expected arrival time?";
        if (endFlag == TRUE) {
            myMessage.text = @"How confident were you that you'd arrive on time?";
        }
        label1.text = @"Not at all";
        label3.text = @"Completely";
        label5.hidden = true;
        if (countSurvey == 1)
        {
            slideArrFirst = slideArrCurrent;
            slideImp.value = 5;
        }
        else if (countSurvey >1)
        {
            slideImp.value = slideConfCurrent;
        }
    }
    else if (countScrn == 4)
    {
        slideConfPrev = slideSatCurrent;
        slideConfCurrent = slideImp.value;

        // Write survey data to file
        NSString  *inTripHeader  =  [NSString  stringWithFormat:@"UserID,  Survey_Timestamp,  Survey_Iteration,  Iteration_Time,
Satisfaction, Arrival_Perf, Confidence\n"];
        NSString *inTripSurvey = [NSString stringWithFormat:@"%@, %@, %i, %@, %f, %f, %f\n", userID, timeSurvey, countSurvey,
timeCurrent, slideSatCurrent, slideArrCurrent, slideConfCurrent];
        gpsFlag = TRUE;
        fileName = [NSString stringWithFormat:@"%@/%@_%@_InTrip.txt", appFolder, userID, timePreSurvey];

        if (countSurvey == 1)
        {
            // Make a file name to write the data and write Header to file
```

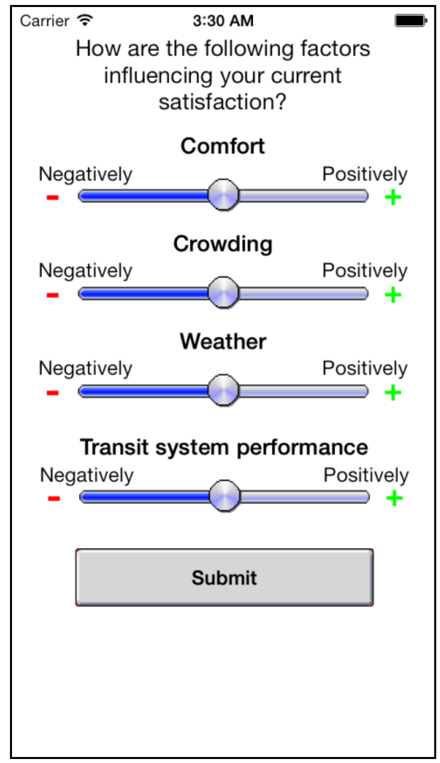
```
            NSLog(@"InTrip Filename = %@", fileName);
            [inTripHeader writeToFile:fileName atomically:YES encoding:NSStringEncodingConversionAllowLossy error:nil];
        }
        //else {
            // File exists, so append next survey record
            NSFileHandle *myHandle = [NSFileHandle fileHandleForWritingAtPath:fileName];
            [myHandle seekToEndOfFile];
            [myHandle writeData:[inTripSurvey dataUsingEncoding:NSStringEncodingConversionAllowLossy]];
        //}

        // If End of Survey, segue to ViewControllerEndSurvey
        if (endFlag == TRUE)
        {
            [self performSegueWithIdentifier:@"segueToEnd" sender:self];
        }
        else
        {
            if (countSurvey == 1)
            {
                slideConfFirst = slideConfCurrent;
                myMessage.text = @"Continue the Survey once you boarded the bus.";
            } else {
                myMessage.text = [NSString stringWithFormat:@"Next survey in %d seconds, or press Continue for manual
initiation.",surveyInterval];
            }
            // Reset screen counter to repeat survey at next interval
            countScrn = 0;
            label1.hidden = true;
            label3.hidden = true;
            label2.hidden = true;
            label4.hidden = true;
            slideImp.hidden = true;
            toolbar.hidden = false;

            // Set to throw next survey at interval
            timer = [NSTimer scheduledTimerWithTimeInterval:timeInterval target:self selector:@selector(timerActions) userInfo:nil
repeats:YES];
            [self GPSActions];
        }
    }
    }
}
-(IBAction)getEnd:(id)sender
{
    // Prepare to throw end of survey screen
    endFlag = TRUE;
    [self getSubmit:(id)sender];
}
-(void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex
{
    if (buttonIndex == 1)
    {
        // User will take the survey
        [timer invalidate];
        timeLapsed = 0;
        //[locationManager stopUpdatingLocation];
        [self getSubmit:butSubmit];
    }
```

```
    if (buttonIndex == 0)
    {
        // User cancelled survey, so remind again in half the time interval
        timeLapsed = surveyInterval/2;
    }
    alertDisplayed = FALSE;
}
-(void)showSurveyAlert
{
    //AudioServicesPlayAlertSound(kSystemSoundID_Vibrate);
    AudioServicesPlayAlertSound(sound1);
    if (alertDisplayed == FALSE)
    {
        alertDisplayed = TRUE;
        UIAlertView *alert = [[UIAlertView alloc]
                             initWithTitle:@"WPTI Survey"
                             message:@"Complete the Survey Now?"
                             delegate:self
                             cancelButtonTitle:@"Cancel"
                             otherButtonTitles:@"Yes", nil];
        alert.transform = CGAffineTransformIdentity;
        alert.transform = CGAffineTransformTranslate(alert.transform, 0.0, -100.0);

        [alert show];
    }
}
-(void)timerActions
{
    timeLapsed = timeLapsed + timeInterval;
    if (countSurvey > 1)
    {
        myMessage.text  =  [NSString  stringWithFormat:@"Next  survey  in  %d  seconds,  or  press  Continue  for  manual
initiation.",(surveyInterval-timeLapsed)];
    }
    NSLog(@" %d",timeLapsed);
    //get GPS coordinate
    [self GPSActions];
    //check for interval to throw next survey
    if (timeLapsed >=surveyInterval)
    {
        timeLapsed = 0;
        // Throw alarm in case app is running in background
        [self scheduleAlarmForDate:[NSDate date]];

        //  Prompt  for  survey  if  user  is  on  the  bus,  otherwise  use  GPS  movement  to  sense  when  user  leaves  bus  stop  (when
countSurvey == 0)
        if (countSurvey > 0)
        {
            AudioServicesPlayAlertSound(kSystemSoundID_Vibrate);
            [self showSurveyAlert];
            [self scheduleAlarmForDate:[NSDate date]];
        }
    }
}
-(void)GPSActions
{
    NSLog(@"GPSActions invoked");
    locationManager.desiredAccuracy = kCLLocationAccuracyBestForNavigation;
```

```
    [locationManager startUpdatingLocation];
    gpsLong = locationManager.location.coordinate.longitude;
    gpsLat = locationManager.location.coordinate.latitude;
    gpsSpeed = locationManager.location.speed * 3.6;
    gpsDir = locationManager.location.course;
    gpsAlt = locationManager.location.altitude;
    gpsAcc = locationManager.location.horizontalAccuracy;
    gpsTime = locationManager.location.timestamp;
    if (gpsStatus.enabled == false)
    {
        gpsStatus.enabled = true;
    }else{
        gpsStatus.enabled = false;
    }

    if (countSurvey == 1) // Monitor bus stop GPS to determine when on bus
    {
        if (gpsFlag == TRUE) // Arrived at bus stop, so get location
        {
            locBusStop = [[CLLocation alloc] initWithLatitude:gpsLat longitude:gpsLong];
        }
        else
        {
            locCurrent = [[CLLocation alloc] initWithLatitude:gpsLat longitude:gpsLong];
            CLLocationDistance distance = ([locCurrent distanceFromLocation:locBusStop]);
            NSLog(@"The distance is %f", distance);
            if (distance > 30)
            {
                onBus = TRUE;
            }
        }
    }
    // Write GPS to Logfile
    NSString *inTripGPS = [NSString stringWithFormat:@"%@, %@, %i, %@, %f, %f, %d, %f, %f, %f, %@\n", userID, timeSurvey,
countSurvey, timeCurrent, gpsLat, gpsLong, gpsSpeed, gpsDir, gpsAlt, gpsAcc, gpsTime];
    fileName = [NSString stringWithFormat:@"%@/%@_%@_%i_GPS.txt", appFolder, userID, timePreSurvey, countSurvey];
    if (gpsFlag == TRUE)
    {
        // Make a file name to write the data
        NSLog(@"InTrip Filename = %@", fileName);
        NSString *gpsHeader = @"UserID, Survey_Timestamp, Survey_Iteration, Iteration_Time, Latitude, Longitude, Speed, Direction,
Altitude, Accuracy, GPS_Time_GMT\n";
        [gpsHeader writeToFile:fileName atomically:YES encoding:NSStringEncodingConversionAllowLossy error:nil];
        gpsFlag = FALSE;
    }
    // Append next survey record to file
    NSFileHandle *gpsHandle = [NSFileHandle fileHandleForWritingAtPath:fileName];
    [gpsHandle seekToEndOfFile];
    [gpsHandle writeData:[inTripGPS dataUsingEncoding:NSStringEncodingConversionAllowLossy]];
    if (onBus == TRUE)
    {
        timeLapsed = 0; // reset timeLapsed to disable next survey alarm by timer
        onBus = FALSE; // reset variables in event of not on bus
        locBusStop = [[CLLocation alloc] initWithLatitude:gpsLat longitude:gpsLong];
        AudioServicesPlayAlertSound(kSystemSoundID_Vibrate);
        [self showSurveyAlert];
    }
    [locationManager stopUpdatingLocation];
```

```
}
-(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    if([segue.identifier isEqualToString:@"segueToSupplemental"]){
        ViewControllerInTrip *controller = (ViewControllerInTrip *)segue.destinationViewController;
        controller.tmpFileID = [NSString stringWithFormat:@"%@/%@_%@_%i", appFolder, userID, timePreSurvey, countSurvey];
        controller.supTimestamp = timeSurvey;
        controller.supIterTime = timeCurrent;
        controller.supCountSurvey = countSurvey;
    }
    if([segue.identifier isEqualToString:@"segueToEnd"]){
        ViewControllerEndSurvey *controller = (ViewControllerEndSurvey *)segue.destinationViewController;
        controller.endFileID = [NSString stringWithFormat:@"%@/%@_%@", appFolder, userID, timePreSurvey];
    }
    if([segue.identifier isEqualToString:@"segueToSettings"]){
        ViewControllerSetting *controller = (ViewControllerSetting *)segue.destinationViewController;
        controller.setUserID = userID;
        controller.setTimeInt = surveyInterval;
        controller.theAppFolder = appFolder;
    }
}
- (IBAction)unwindFromSegue:(UIStoryboardSegue *)unwindSegue
{
    UIViewController *sourceViewController = unwindSegue.sourceViewController;
    [locationManager startUpdatingLocation];
    if ([sourceViewController isKindOfClass:[ViewControllerTextInput class]])
    {
        NSLog(@"Returned from ViewControllerTextInput");
    }
    else if ([sourceViewController isKindOfClass:[ViewControllerEndSurvey class]])
    {
        NSLog(@"Returned from ViewControllerEndSurvey");

        countScrn = 100;
        countSurvey = 0;
        myMessage.text = @"Thanks! Press the Start button when you are ready to start your next trip.";
        label1.hidden = true;
        label3.hidden = true;
        label2.hidden = true;
        label4.hidden = true;
        slideImp.hidden = true;
        butSubmit.titleLabel.text = @"Start...";
        endFlag = FALSE;
        [locationManager stopUpdatingLocation];
    }
    else if ([sourceViewController isKindOfClass:[ViewControllerSetting class]])
    {
        NSLog(@"Returned from ViewControllerSetting");
        ViewControllerSetting *settingController = [unwindSegue sourceViewController];
        surveyInterval = settingController.setTimeInt;
    }
}
- (IBAction)done:(UIStoryboardSegue *)segue {
    ViewControllerSetting *settingViewController = [segue sourceViewController];
    surveyInterval = settingViewController.setTimeInt;
}
- (void)viewDidUnload
{
```

```
    [super viewDidUnload];
}
- (void)throwSupplemental
{
    [self performSegueWithIdentifier:@"segueToSupplemental" sender:self];
}
-(IBAction)dismissKeyboardOnTap:(id)sender
{
    [[self view] endEditing:YES];
}
- (void)scheduleAlarmForDate:(NSDate*)theDate
{
    NSDate* newDate = [theDate dateByAddingTimeInterval:5];
    NSLog(@"Current time: %@, Alarm time: %@", theDate, newDate);
    UIApplication* app = [UIApplication sharedApplication];
    NSArray*    oldNotifications = [app scheduledLocalNotifications];
    // Clear out the old notification before scheduling a new one.
    if ([oldNotifications count] > 0)
        [app cancelAllLocalNotifications];
    // Create a new notification.
    UILocalNotification* alarm = [[UILocalNotification alloc] init];
    if (alarm)
    {
        alarm.fireDate = newDate;
        alarm.timeZone = [NSTimeZone defaultTimeZone];
        alarm.repeatInterval = 0;
        alarm.soundName = @"alert.aif";
        alarm.alertBody = @"Please resume the survey.";
        [app scheduleLocalNotification:alarm];
    }
}
@end
```



ViewControllerInTrip

The In-Trip View Controller handles the survey slider questions.

ViewControllerInTrip.h

```
//  ViewControllerInTrip.h
//  wptiApp
//
//  Created by Ian Dunlop on 2013-10-01.
//  Copyright (c) 2013 Ian Dunlop. All rights reserved.

#import <UIKit/UIKit.h>
#import "ViewControllerTextInput.h"
@interface ViewControllerInTrip : UIViewController
{
}
@property(nonatomic, retain) IBOutlet UIButton *butSubmit;
@property(nonatomic, retain) IBOutlet UISlider *slideComf;
@property(nonatomic, retain) IBOutlet UISlider *slideCrowd;
@property(nonatomic, retain) IBOutlet UISlider *slideWeath;
@property(nonatomic, retain) IBOutlet UISlider *slidePerf;
@property(nonatomic, retain) NSString *tmpFileID;
@property(nonatomic, retain) NSString *supTimestamp;
```

```
@property(nonatomic, retain) NSString *supIterTime;
@property int supCountSurvey;
-(IBAction)getSubmit:(id)sender;
-(IBAction)adjSlider:(id)sender;
@end
```

ViewControllerInTrip.m

```
//  ViewControllerInTrip.m
//  SampleApp
//
//  Created by Ian Dunlop on 2013-10-01.
//  Copyright (c) 2013 Ian Dunlop. All rights reserved.
#import "ViewControllerInTrip.h"
@interface ViewControllerInTrip ()
@end
@implementation ViewControllerInTrip
@synthesize slideComf, slideCrowd, slidePerf, slideWeath, tmpFileID, supCountSurvey, supIterTime, supTimestamp ;
int slideSatComf;
int slideSatCrowd;
int slideSatWeath;
int slideSatPerf;
bool sliderChanged = false;
-(IBAction)getSubmit:(id)sender
{
    if (sliderChanged == true){
    slideSatComf = slideComf.value;
    slideSatCrowd = slideCrowd.value;
    slideSatWeath = slideWeath.value;
    slideSatPerf = slidePerf.value;

    NSString *suppHeader = [NSString stringWithFormat:@"UserID, Survey_Timestamp, Survey_Iteration, Iteration_Time, Comfort, Wellbeing, Weather, Performance\n"];
    NSString *suppSurvey = [NSString stringWithFormat:@"%@, %@, %d, %@, %d, %d, %d, %d\n", tmpFileID, supTimestamp, supCountSurvey, supIterTime, slideSatComf, slideSatCrowd, slideSatWeath, slideSatPerf];

    NSString *suppFileName = [NSString stringWithFormat:@"%@_Supplemental.txt", tmpFileID];
        [suppHeader writeToFile:suppFileName atomically:YES encoding:NSStringEncodingConversionAllowLossy error:nil];
        [suppSurvey writeToFile:suppFileName atomically:YES encoding:NSStringEncodingConversionAllowLossy error:nil];

    [self performSegueWithIdentifier:@"segueToInput" sender:self];
    } else {
        UIAlertView *alert = [[UIAlertView alloc]

                              initWithTitle:@"WPTI Survey"
                              message:@"Please make a selection."
                              delegate:self
                              cancelButtonTitle:@"OK"
                              otherButtonTitles:nil];
        alert.transform = CGAffineTransformIdentity;
        alert.transform = CGAffineTransformTranslate(alert.transform, 0.0, -100.0);
        [alert show];
    }
}
-(IBAction)adjSlider:(id)sender
{
    UISlider *slider = (UISlider*)sender;
```

```
    int slideAdj = (slider.value+0.5);
    slider.value = slideAdj;
    sliderChanged = true;
}
- (id)init
{
    self = [super initWithNibName:nil bundle:nil];
    if (self != nil) {
    }
    return self;
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
    }
    return self;
}
-(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    if([segue.identifier isEqualToString:@"segueToInput"])
    {
            ViewControllerTextInput *controller = (ViewControllerTextInput *)segue.destinationViewController;
            controller.recFileID = tmpFileID;
    }
}
- (void)viewDidLoad
{
    [super viewDidLoad];
}
- (void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:(BOOL)animated];
        sliderChanged = false;
}

- (void)viewDidAppear:(BOOL)animated {
    [super viewDidAppear:(BOOL)animated];
    NSLog(@"tmpFileID = %@",tmpFileID);
}
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
}
@end
```

ViewControllerTextInput

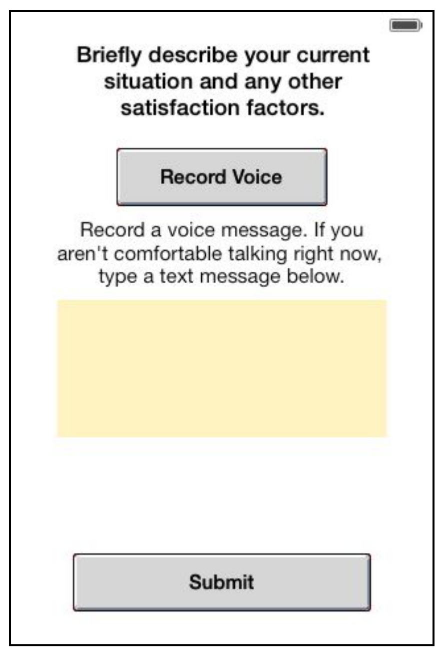
The TextInput screen handles manual input from the user to supplement the slider bar attributes. The user can type in a text message or make a voice recording. The data is saved with the userID and survey timestamp.

ViewControllerTextInput.h

```objc
//  ViewControllerTextInput.h
//  wptiApp
//
//  Created by Ian Dunlop on 2013-10-31.
//  Copyright (c) 2013 Ian Dunlop. All rights reserved.
#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>
@interface ViewControllerTextInput : UIViewController <AVAudioRecorderDelegate>
@property(nonatomic, retain) IBOutlet UIButton *butSubmitText;
@property(nonatomic, retain) IBOutlet UIButton *butRecord;
@property(nonatomic, retain) IBOutlet UITextView *txtInput;
@property(nonatomic, retain) IBOutlet UILabel *instructLabel;
@property(nonatomic, retain) NSString *recFileID;
@property(nonatomic, retain) NSString *recFilePath;
@property(nonatomic, retain) NSString *textResponse;
-(IBAction)getSubmitText:(id)sender;
-(IBAction)getRecording:(id)sender;
-(IBAction)tapText:(id)sender;
-(IBAction)dismissKeyboardOnTap:(id)sender;
@end
```

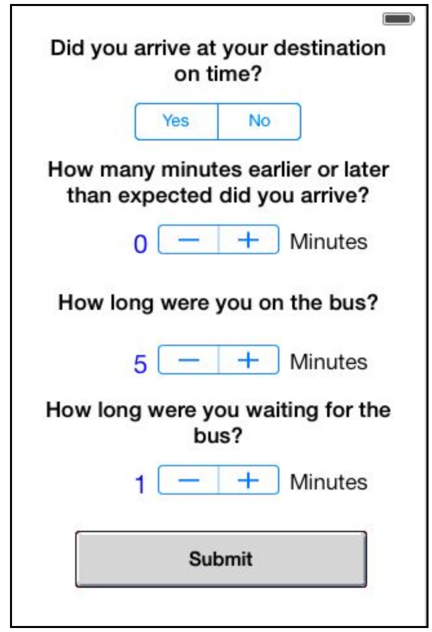
ViewControllerTextInput.m

```objc
//  ViewControllerTextInput.m
//  SampleApp
//
//  Created by Ian Dunlop on 2013-10-31.
//  Copyright (c) 2013 Ian Dunlop. All rights reserved.
#import "ViewControllerTextInput.h"
@interface ViewControllerTextInput () {
    AVAudioRecorder *recorder;
}
@end
@implementation ViewControllerTextInput
@synthesize butRecord, butSubmitText, txtInput, recFileID, recFilePath, instructLabel;

int recFlag;
-(IBAction)getSubmitText:(id)sender
{
    // Write text input to file and close window
    NSString *txtSurvey = txtInput.text;
    NSString *inFileName = [NSString stringWithFormat:@"%@_TextInput.txt", recFileID];
    [txtSurvey writeToFile:inFileName atomically:YES
encoding:NSStringEncodingConversionAllowLossy error:nil];

    [self performSegueWithIdentifier:@"UnwindInput" sender:self];
}
-(IBAction)getRecording:(id)sender
{
    if (recFlag == 1) {
        [recorder stop];

        //stop recording and write audio to file
        AVAudioSession *audioSession = [AVAudioSession sharedInstance];
        [audioSession setActive:NO error:nil];
```



```
        [butRecord setTitle:@"Record Voice" forState:UIControlStateNormal];
        recFlag = 0;
    }
    else {
        AVAudioSession *session = [AVAudioSession sharedInstance];
        [session setActive:YES error:nil];

        // Start recording
        [recorder record];
        [butRecord setTitle:@"Stop Recording" forState:UIControlStateNormal];
        recFlag = 1;
    }
}
-(IBAction)tapText:(id)sender
{
    instructLabel.text = @"Tap here when done.";
}
-(IBAction)dismissKeyboardOnTap:(id)sender
{
    [[self view] endEditing:YES];
    instructLabel.text = @"Record a voice message. If you aren't comfortable talking right now, type a text message below.";
}
- (id)init
{
    self = [super initWithNibName:nil bundle:nil];
    if (self != nil) {
        // Custom initialization
    }
    return self;
}
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}
- (void)viewDidLoad
{
    [super viewDidLoad];
      // Do any additional setup after loading the view.

    if (recFileID == NULL) {
        recFileID = @"NoID";
    }
    recFlag = 0;
    // Set the audio file
    NSString *recFileName = [NSString stringWithFormat:@"%@_Audio.m4a", recFileID];
    NSLog(@"Recording filename = %@",recFileName);
    NSArray *pathComponents = [NSArray arrayWithObjects:
                               recFileName, nil];
    NSLog(@"Recording filename = %@",pathComponents);
    NSURL *outputFileURL = [NSURL fileURLWithPathComponents:pathComponents];
    // Setup audio session
    AVAudioSession *session = [AVAudioSession sharedInstance];
    [session setCategory:AVAudioSessionCategoryPlayAndRecord error:nil];
    // Define the recorder setting
```

```
    NSMutableDictionary *recordSetting = [[NSMutableDictionary alloc] init];
    [recordSetting        setValue:[NSNumber        numberWithInt:kAudioFormatMPEG4AAC]
forKey:AVFormatIDKey];
    [recordSetting setValue:[NSNumber numberWithFloat:44100.0] forKey:AVSampleRateKey];
    [recordSetting setValue:[NSNumber numberWithInt: 2] forKey:AVNumberOfChannelsKey];
    // Initiate and prepare the recorder
    recorder = [[AVAudioRecorder alloc] initWithURL:outputFileURL settings:recordSetting
error:NULL];
    recorder.delegate = self;
    recorder.meteringEnabled = YES;
    [recorder prepareToRecord];
}
- (void)viewDidDisappear:(BOOL)animated {
    [super viewDidDisappear:(BOOL)animated];
}
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
@end
```



ViewControllerEndSurvey

The EndSurvey screen asks the final questions to assess the users' perception of how long the trip took and whether or not they actually arrived on time.

ViewControllerEndSurvey.h

```
//  ViewControllerEndSurvey.h
//  wptiApp
//
//  Created by Ian Dunlop on 2013-11-04.
//  Copyright (c) 2013 Ian Dunlop. All rights reserved.
#import <UIKit/UIKit.h>
@interface ViewControllerEndSurvey : UIViewController
@property(nonatomic, retain) NSString *endFileID;
@property(nonatomic, retain) IBOutlet UIButton *butSubmitEnd;
@property(nonatomic, retain) IBOutlet UIStepper *stepArr;
@property(nonatomic, retain) IBOutlet UIStepper *stepBus;
@property(nonatomic, retain) IBOutlet UIStepper *stepWait;
@property(nonatomic, retain) IBOutlet UISegmentedControl *segYesNo;
@property(nonatomic, retain) IBOutlet UILabel *timeArr;
@property(nonatomic, retain) IBOutlet UILabel *timeBus;
@property(nonatomic, retain) IBOutlet UILabel *timeWait;
-(IBAction)incArr:(id)sender;
-(IBAction)incBus:(id)sender;
-(IBAction)incWait:(id)sender;
-(IBAction)getSubmitEnd:(id)sender;
@end
```

ViewControllerEndSurvey.m

```
//  ViewControllerEndSurvey.m
//  wptiApp
```

```
//
//  Created by Ian Dunlop on 2013-11-04.
//  Copyright (c) 2013 Ian Dunlop. All rights reserved.
#import "ViewControllerEndSurvey.h"
@interface ViewControllerEndSurvey ()
@end
@implementation ViewControllerEndSurvey
@synthesize stepArr, stepBus, stepWait, timeArr, timeBus, timeWait, endFileID, segYesNo;
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}
- (void)viewDidLoad
{
    [super viewDidLoad];
      // Do any additional setup after loading the view.
}
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
-(IBAction)incArr:(id)sender;
{
    timeArr.text = [NSString stringWithFormat:@"%d", ((int) stepArr.value)];
}
-(IBAction)incBus:(id)sender;
{
    timeBus.text = [NSString stringWithFormat:@"%d", ((int) stepBus.value)];
}
-(IBAction)incWait:(id)sender;
{
    timeWait.text = [NSString stringWithFormat:@"%d", ((int) stepWait.value)];
}
-(IBAction)getSubmitEnd:(id)sender;
{
    // Write data to file
    NSString *endSurvey = [NSString stringWithFormat:@"As Expected (Y/N=0/1) = %i\nDestination Arrival = %@\nTime On Bus = 
%@\nTime Waiting = %@", segYesNo.selectedSegmentIndex, timeArr.text, timeBus.text, timeWait.text];

    NSString *endFileName = [NSString stringWithFormat:@"%@_EndSurvey.txt", endFileID];
    [endSurvey writeToFile:endFileName atomically:YES encoding:NSStringEncodingConversionAllowLossy error:nil];

    // Unwind segue
        [self performSegueWithIdentifier:@"UnwindEnd" sender:self];
}
@end
```

ViewControllerSettings

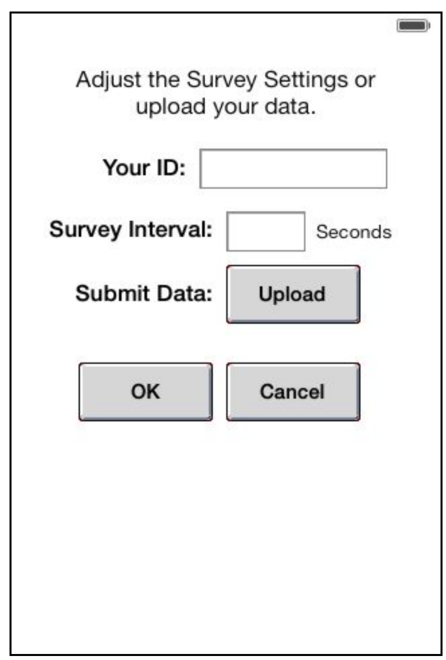
The Settings screen allows the user to upload their data via email to the researchers, and adjust the survey time interval.

ViewControllerSettings.h

```
//  ViewControllerSetting.h
//  wptiApp
//
//  Created by Ian Dunlop on 2014-01-02.
//  Copyright (c) 2014 Ian Dunlop. All rights reserved.
#import <UIKit/UIKit.h>
@interface ViewControllerSetting : UIViewController
@property(nonatomic, retain) NSString *setUserID;
@property(nonatomic, retain) NSString *theAppFolder;
@property int setTimeInt;
@property(nonatomic, retain) IBOutlet UIButton *butOk;
@property(nonatomic, retain) IBOutlet UIButton *butCan;
@property(nonatomic, retain) IBOutlet UIButton *butUpload;
@property(nonatomic, retain) IBOutlet UITextField *setTheId;
@property(nonatomic, retain) IBOutlet UITextField *setTheInt;
-(IBAction)getOK:(id)sender;
-(IBAction)getCan:(id)sender;
-(IBAction)uploadData:(id)sender;
-(IBAction)dismissKeyboardOnTap:(id)sender;
@end
```

ViewControllerSettings.m

```
//  ViewControllerSetting.m
//  wptiApp
//
//  Created by Ian Dunlop on 2014-01-02.
//  Copyright (c) 2014 Ian Dunlop. All rights reserved.
#import "ViewControllerSetting.h"
#import <MessageUI/MessageUI.h>
@interface ViewControllerSetting () <MFMailComposeViewControllerDelegate>
@property (nonatomic, strong) NSArray *filenames;
@end
@implementation ViewControllerSetting
@synthesize  butCan,  butOk,  butUpload,  setTheId,  setTheInt,  setUserID,  setTimeInt,
theAppFolder;
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}
- (void)viewDidLoad
{
    [super viewDidLoad];
      // Do any additional setup after loading the view.
    setTheId.text = setUserID;
    setTheInt.text = [NSString stringWithFormat:@"%d",setTimeInt];
    // Get the application folder and list of files
    NSArray *paths = NSSearchPathForDirectoriesInDomains
    (NSDocumentDirectory, NSUserDomainMask, YES);
    NSString *documentsDirectory = [paths objectAtIndex:0];
    NSFileManager *fileManager = [[NSFileManager alloc] init];
```



```
    theAppFolder = [documentsDirectory stringByAppendingPathComponent:@"WPTI"];
    BOOL isDir;
    BOOL isFile = [[NSFileManager defaultManager] fileExistsAtPath:theAppFolder isDirectory:&isDir];
    if(isFile)
    {
        _filenames = [fileManager contentsOfDirectoryAtPath:theAppFolder error:nil];
        }
    if(!isFile || _filenames.count == 0) {
        butUpload.enabled = false;
        NSLog(@"Upload disabled");
    }
}
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
-(IBAction)getCan:(id)sender;
{
       // [self performSegueWithIdentifier:@"UnwindCan" sender:self];
}
-(IBAction)getOK:(id)sender;
{
    // Update settings here
    int theInt = [setTheInt.text intValue];
    if (theInt < 15 || theInt > 3600) {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"WPTI Survey"
                                                        message:@"Enter a valid time interval between 15 and 3600 seconds"
                                                       delegate:nil
                                              cancelButtonTitle:@"OK"
                                              otherButtonTitles:nil];
        [alert show];

    } else {
        setTimeInt = [setTheInt.text intValue];
        //[self performSegueWithIdentifier:@"UnwindSet" sender:self];
    }
}

-(IBAction)uploadData:(id)sender;
{
    NSString *emailTitle = @"WPTI Survey Submission";
    NSString *messageBody = [NSString stringWithFormat:@"The WPTI Survey App files are attached for UserID %@", setUserID];
    NSArray *toRecipents = [NSArray arrayWithObject:@"wptiapp@strategicinterchange.ca"];
    MFMailComposeViewController *mc = [[MFMailComposeViewController alloc] init];
    [mc setMailComposeDelegate:self];
    [mc setSubject:emailTitle];
    [mc setMessageBody:messageBody isHTML:NO];
    [mc setToRecipients:toRecipents];
    NSString *file;
    NSArray *filepart;
    NSString *filename;
    NSString *extension;
    NSString *filePath;
    NSData *fileData;
    NSString *mimeType;
    for(int fc = 0; fc < _filenames.count; fc++) { //fc < _filenames.count // fc < 3 (for debug)
        // Determine the file name and extension
```

```
        file = [_filenames objectAtIndex:fc];
        filepart = [file componentsSeparatedByString:@"."];
        filename = [filepart objectAtIndex:0];
        extension = [filepart objectAtIndex:1];
        // Get the resource path and read the file using NSData
        filePath = [theAppFolder stringByAppendingPathComponent:file];
        fileData = [NSData dataWithContentsOfFile:filePath];
        // Determine the MIME type
        //NSString *mimeType;
        if ([extension isEqualToString:@"m4a"]) {
            mimeType = @"audio/mpeg";
        } else if ([extension isEqualToString:@"txt"]) {
            mimeType = @"text/plain";
        }
        // Add attachment
        [mc addAttachmentData:fileData mimeType:mimeType fileName:file];
    }
    // Present mail view controller on screen
    [self presentViewController:mc animated:YES completion:NULL];
}

-   (void)   mailComposeController:(MFMailComposeViewController   *)controller   didFinishWithResult:(MFMailComposeResult)result
error:(NSError *)error
{
    switch (result)
    {
        case MFMailComposeResultCancelled:
            NSLog(@"Mail cancelled");
            break;
        case MFMailComposeResultSaved:
            NSLog(@"Mail saved");
            break;
        case MFMailComposeResultSent:
            NSLog(@"Mail sent");
            break;
        case MFMailComposeResultFailed:
            NSLog(@"Mail sent failure: %@", [error localizedDescription]);
            break;
        default:
            break;
    }
    // Close the Mail Interface
    [self dismissViewControllerAnimated:YES completion:NULL];
}
-(IBAction)dismissKeyboardOnTap:(id)sender
{
    [[self view] endEditing:YES];
}
-(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
}
@end
```

Appendix E: WPTI Survey Text Processor – Visual Basic

This Visual Basic app parses the individual survey responses into comma-delineated value text files. The survey text files are first placed into a temporary folder. When this app is executed, the app processes each type of survey file (pre-trip, in-trip, end-trip and supplemental phases of the surveys, plus GPS feeds and text responses) by collecting the individual files and then formatting and appending the responses to master files. There were over 5,800 text files to be processed, which were parsed down to 6 and then imported to Excel for analysis.

```
Imports VB = Microsoft.VisualBasic
Imports System.IO

Public Class Form1

    Private Property values As String()
    Private Property filestrings As String()

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

        Dim f, x As Integer
        Dim fn As String
        Dim validity As Boolean
        Dim myDir As String
        Dim inputstr As String
        Dim outputstr1, outputstr2 As String
        Dim row As Integer
        Dim Platform, Version As String
        Dim tripID, oldTripID As String
        Dim UserID, Survey_Datestamp, Survey_Time, Iteration_Time As String
        Dim Survey_Iteration, old_Iteration, importance, satisfaction, arrival_Perf, confidence As Integer
        Dim accPickUp, accDest, accDropOff As Integer
        Dim comfort, crowding, wellbeing, weather, transit_Perf As Integer
        Dim Trip_Purpose, Time_PickUp, Time_Dest, Time_DropOff As String
        Dim onTime As Boolean
        Dim durArr, durOnBus, durWait As Integer
        Dim latitude, longitude, speed, direction, accuracy As Double
        Dim gpsTime As String
        Dim textInput As String

        If Button1.Text = "Close" Then End
        Button1.Text = "Processing" : Button1.Enabled = False

        Label1.Text = "Status: Processing"
        myDir = Me.TextBox1.Text
        Platform = "u"

        '------------- PROCESS _InTrip FILES --------------

        FileOpen(2, myDir & "\Aug14_InTrip.csv", OpenMode.Output)
        FileOpen(3, myDir & "\Aug14_Trips.csv", OpenMode.Output)
        'Write output file header
        WriteLine(2, "TripID", "UserID", "Survey_Datestamp", "Survey_Time", "Survey_Iteration", "Iteration_Time", "Satisfaction", 
"Arrival_Perf", "Confidence")
        WriteLine(3, "TripID", "UserID", "Survey_Datestamp", "Start_Time", "Last_Iteration", "End_Time", "Platform", "Version", 
"File_Prefix", "Completed")

        Dim filenames = Directory.EnumerateFiles(myDir, "*_InTrip.txt")
```

```
        For f = 0 To filenames.Count - 1
            fn = VB.Right(filenames(f), Len(filenames(f)) - Len(myDir) - 1)
            'Stop
            row = 1
            filestrings = Split(fn, "_")
            UserID = filestrings(0)
            If InStr(filestrings(1), "-") > 0 Then
                Survey_Datestamp = "2014" & VB.Right(filestrings(1), 2) & VB.Left(filestrings(1), 2) & VB.Left(filestrings(2), 2)
& VB.Right(filestrings(2), 2)
                Platform = "i"
            Else
                Survey_Datestamp = VB.Left(filestrings(1), Len(filestrings(1)) - 2)
                Platform = "a"
            End If
            Survey_Datestamp = "[" & Survey_Datestamp & "]"
            tripID = UserID & Survey_Datestamp
            FileOpen(1, myDir & "\" & fn, OpenMode.Input)

            Do While Not EOF(1)
                inputstr = LineInput(1)
                values = Split(inputstr, ",")
                x = values.Length - 1
                If x = 0 Then
                    'Data not formatted in columns (App v2 output format), so check for rows (App v1 output format)
                    values = Split(inputstr, "=")
                    x = values.Length - 1
                    If x = 0 Then
                        'Not a valid data string (ignore this row)
                        validity = False
                    Else
                        If values(0).ToString = "User ID " Or values(0).ToString = "UserID " Then
                            validity = True
                            Version = "1.0"

                            'process the values line by line
                            For xx = 1 To 6
                                inputstr = LineInput(1)
                                values = Split(inputstr, "=")
                                If xx = 1 Then Survey_Time = LTrim(values(1))
                                If xx = 2 Then Survey_Iteration = Val(values(1))
                                If xx = 3 Then Iteration_Time = LTrim(values(1))
                                If xx = 4 Then satisfaction = Val(values(1))
                                If xx = 5 Then arrival_Perf = Val(values(1))
                                If xx = 6 Then confidence = Val(values(1))
                            Next

                        Else
                            validity = False
                        End If
                    End If
                Else
                    'process comma-separated values
                    If values(0).ToString = "User ID" Or values(0).ToString = "UserID" Then
                        'Header line (ignore this row)
                        validity = False
                    Else
                        validity = True
```

```
                    Version = "2.0"
                    Survey_Time = LTrim(values(1))
                    Survey_Iteration = Val(values(2))
                    Iteration_Time = LTrim(values(3))
                    satisfaction = Val(values(4))
                    arrival_Perf = Val(values(5))
                    confidence = Val(values(6))
                End If

            End If
            'Write data to the output file
            If validity = True Then
                outputstr1 = UserID & ", " & Survey_Datestamp & "," & Survey_Time & ", " & Survey_Iteration & "," & Iteration_Time & ", " & satisfaction & ", " & arrival_Perf & ", " & confidence
                WriteLine(2, tripID, UserID, Survey_Datestamp, Survey_Time, Survey_Iteration, Iteration_Time, satisfaction, arrival_Perf, confidence)
                row = row + 1
            End If
        Loop
        outputstr2 = Dir(myDir & "\" & VB.Left(fn, Len(fn) - 11) & "_End*.txt")
        'Stop
        If outputstr2 = "" Then
            validity = False
        Else
            validity = True
        End If
        WriteLine(3, tripID, UserID, Survey_Datestamp, Survey_Time, Survey_Iteration, Iteration_Time, Platform, Version, VB.Left(fn, Len(fn) - 11), validity)
        FileClose(1)
        'fn = Dir()
    Next

    FileClose(2)
    FileClose(3)

    '-------------- PROCESS _PreTrip FILES ---------------

    FileOpen(4, myDir & "\Aug14_PreTrip.csv", OpenMode.Output)

    WriteLine(4, "TripID", "UserID", "Survey_Datestamp", "Survey_Time", "Importance", "Trip_Purpose", "Time_PickUp", "Acc_PickUp", "Time_Dest", "Acc_Dest", "Time_DropOff", "Acc_DropOff")
    fn = Dir(myDir & "\*_PreTrip*.txt")

    Do While fn <> ""
        filestrings = Split(fn, "_")
        UserID = filestrings(0)
        If InStr(filestrings(1), "-") > 0 Then
            Survey_Datestamp = "2014" & VB.Right(filestrings(1), 2) & VB.Left(filestrings(1), 2) & VB.Left(filestrings(2), 2) & VB.Right(filestrings(2), 2)
            'Platform = "i"
        Else
            Survey_Datestamp = VB.Left(filestrings(1), Len(filestrings(1)) - 2)
            'Platform = "a"
        End If
        Survey_Datestamp = "[" & Survey_Datestamp & "]"
        tripID = UserID & Survey_Datestamp
        FileOpen(5, myDir & "\" & fn, OpenMode.Input)
```

```
            Do While Not EOF(5)
                inputstr = LineInput(5)
                values = Split(inputstr, "=")
                x = values.Length - 1
                If x <> 1 Then
                    'Not a valid data string (ignore this row)
                    validity = False
                Else
                    validity = True
                    'UserID = values(1)
                    For xx = 1 To 6
                        inputstr = LineInput(5)
                        values = Split(inputstr, "=")
                        If xx = 1 Then Survey_Time = LTrim(values(1))
                        If xx = 2 Then importance = Val(values(1))
                        If xx = 3 Then Trip_Purpose = LTrim(values(1))
                        If xx = 4 Then Time_PickUp = VB.Right(values(1), 8)
                        If xx = 5 Then Time_Dest = VB.Right(values(1), 8)
                        If xx = 6 Then Time_DropOff = VB.Right(values(1), 8)
                    Next
                End If
                'Write data to the output file
                If validity = True Then
                    accPickUp = VB.Right(Time_PickUp, 1)
                    Time_PickUp = VB.Left(Time_PickUp, 5)
                    accDest = VB.Right(Time_PickUp, 1)
                    Time_Dest = VB.Left(Time_PickUp, 5)
                    accDropOff = VB.Right(Time_PickUp, 1)
                    Time_DropOff = VB.Left(Time_PickUp, 5)

                    outputstr1 = UserID & ", " & Survey_Datestamp & "," & Survey_Time & ", " & importance & "," & Trip_Purpose &
", " & Time_PickUp & ", " & Time_Dest & ", " & Time_DropOff
                    WriteLine(4, tripID, UserID, Survey_Datestamp, Survey_Time, importance, Trip_Purpose, Time_PickUp, accPickUp,
Time_Dest, accDest, Time_DropOff, accDropOff)
                Else
                    outputstr1 = fn & " IS INVALID!"
                    WriteLine(4, outputstr1)
                End If
            Loop
            FileClose(5)
            fn = Dir()
        Loop

        FileClose(4)

        '-------------- PROCESS _Supplemental FILES ----------------

        FileOpen(6, myDir & "\Aug14_Supplemental.csv", OpenMode.Output)

        WriteLine(6, "TripID", "UserID", "Survey_Datestamp", "Survey_Iteration", "Comfort", "Crowding", "Wellbeing", "Weather",
"Transit_System")
        fn = Dir(myDir & "\*_Supplemental*.txt")

        Do While fn <> ""
            crowding = -1
            wellbeing = -1
            Iteration_Time = " NA"
            filestrings = Split(fn, "_")
```

```
            UserID = filestrings(0)
            If InStr(filestrings(1), "-") > 0 Then
                Survey_Datestamp = "2014" & VB.Right(filestrings(1), 2) & VB.Left(filestrings(1), 2) & VB.Left(filestrings(2), 2)
& VB.Right(filestrings(2), 2)
                'Platform = "i"
            Else
                Survey_Datestamp = VB.Left(filestrings(1), Len(filestrings(1)) - 2)
                'Platform = "a"
            End If
            Survey_Datestamp = "[" & Survey_Datestamp & "]"
            tripID = UserID & Survey_Datestamp
            FileOpen(7, myDir & "\" & fn, OpenMode.Input)

            Do While Not EOF(7)
                inputstr = LineInput(7)
                'If UserID = "55" Then Stop
                values = Split(inputstr, ",")
                x = values.Length - 1
                If x < 2 Then
                    values = Split(inputstr, "=")
                    x = values.Length - 1
                    If x = 1 Then
                        validity = True
                        Survey_Iteration = filestrings(3)
                        comfort = Val(values(1))
                        'Version = "1.0"
                        'process the values line by line
                        For xx = 1 To 3
                            inputstr = LineInput(7)
                            values = Split(inputstr, "=")
                            If xx = 1 Then crowding = Val(values(1))
                            If xx = 2 Then weather = Val(values(1))
                            If xx = 3 Then transit_Perf = Val(values(1))
                        Next
                    Else
                        validity = False
                    End If
                Else
                    validity = True
                    Survey_Time = LTrim(values(1))
                    Survey_Iteration = Val(values(2))
                    Iteration_Time = LTrim(values(3))
                    comfort = Val(values(4))
                    wellbeing = Val(values(5))
                    weather = Val(values(6))
                    transit_Perf = Val(values(7))
                End If
                'Write data to the output file
                If validity = True Then
                    outputstr1 = UserID & ", " & Survey_Datestamp & "," & Survey_Iteration & ", " & comfort & ", " & crowding & ",
" & wellbeing & ", " & weather & ", " & transit_Perf
                    WriteLine(6, tripID, UserID, Survey_Datestamp, Survey_Iteration, comfort, crowding, wellbeing, weather,
transit_Perf)
                Else
                    outputstr1 = fn & " IS INVALID!"
                    WriteLine(6, outputstr1)
                End If
            Loop
```

```
            FileClose(7)
            fn = Dir()
        Loop

        FileClose(6)

        '------------------PROCESS _EndTrip FILES -------------------

        FileOpen(8, myDir & "\Aug14_EndTrip.csv", OpenMode.Output)

        WriteLine(8, "TripID", "UserID", "Survey_Datestamp", "OnTime", "Duration_Arr", "Duration_OnBus", "Duration_Wait")
        fn = Dir(myDir & "\*_End*.txt")

        Do While fn <> ""
            filestrings = Split(fn, "_")
            UserID = filestrings(0)
            If InStr(filestrings(1), "-") > 0 Then
                Survey_Datestamp = "2014" & VB.Right(filestrings(1), 2) & VB.Left(filestrings(1), 2) & VB.Left(filestrings(2), 2)
& VB.Right(filestrings(2), 2)
                'Platform = "i"
            Else
                Survey_Datestamp = VB.Left(filestrings(1), Len(filestrings(1)) - 2)
                'Platform = "a"
            End If
            Survey_Datestamp = "[" & Survey_Datestamp & "]"
            tripID = UserID & Survey_Datestamp
            FileOpen(9, myDir & "\" & fn, OpenMode.Input)
            Do While Not EOF(9)
                inputstr = LineInput(9)
                values = Split(inputstr, "=")
                x = values.Length - 1
                If x > 0 Then
                    If VB.Left(values(0), 4) = "User" Then
                        row = 2
                    Else
                        row = 4
                        If x = 1 Then
                            onTime = Val(values(1))
                        Else
                            onTime = Val(values(2))
                        End If
                    End If
                    'Version = "1.0"
                    'process the values line by line
                    'If UserID = "120" Then Stop
                    For xx = row To 6
                        inputstr = LineInput(9)
                        values = Split(inputstr, "=")
                        If xx = 3 Then
                            If values.Length = 2 Then
                                onTime = Val(values(1))
                            Else
                                onTime = VB.Right(values(0), 1)
                            End If
                        End If
                        If xx = 4 Then durArr = Val(values(1))
                        If xx = 5 Then durOnBus = Val(values(1))
```

```
                    If xx = 6 Then durWait = Val(values(1))
                Next
            Else
                validity = False
            End If
            'Write data to the output file
            If validity = True Then
                outputstr1 = UserID & ", " & Survey_Datestamp & ", " & onTime & ", " & durArr & ", " & durOnBus & ", " &
durWait
                WriteLine(8, tripID, UserID, Survey_Datestamp, onTime, durArr, durOnBus, durWait)
            Else
                outputstr1 = fn & " IS INVALID!"
                WriteLine(8, outputstr1)
            End If
        Loop
        FileClose(9)
        fn = Dir()
    Loop

    FileClose(8)

    '----------------- PROCESS _GPS FILES --------------------

    FileOpen(10, myDir & "\Aug14_GPS.csv", OpenMode.Output)
    FileOpen(20, myDir & "\Aug14_GPS_Waypoints.csv", OpenMode.Output)
    'Write output file header
    WriteLine(10, "TripID", "UserID", "Survey_Datestamp", "Survey_Iteration", "Iteration_Time", "Sequence", "Latitude",
"Longitude", "Speed", "Direction", "Accuracy", "GPS_Time")
    WriteLine(20, "TripID", "UserID", "Survey_Datestamp", "Survey_Iteration", "Iteration_Time", "Latitude", "Longitude",
"Speed", "Direction", "Accuracy", "GPS_Time")

    fn = Dir(myDir & "\*_GPS*.txt")
    old_Iteration = -1
    oldTripID = ""

    Do While fn <> ""
        filestrings = Split(fn, "_")
        UserID = filestrings(0)
        If InStr(filestrings(1), "-") > 0 Then
            Survey_Datestamp = "2014" & VB.Right(filestrings(1), 2) & VB.Left(filestrings(1), 2) & VB.Left(filestrings(2), 2)
& VB.Right(filestrings(2), 2)
            Platform = "i"
        Else
            Survey_Datestamp = VB.Left(filestrings(1), Len(filestrings(1)) - 2)
            Platform = "a"
        End If
        Survey_Datestamp = "[" & Survey_Datestamp & "]"
        tripID = UserID & Survey_Datestamp
        If oldTripID <> tripID Then row = 1
        FileOpen(11, myDir & "\" & fn, OpenMode.Input)
        Do While Not EOF(11)
            inputstr = LineInput(11)
            values = Split(inputstr, ",")
            x = values.Length - 1
            'process comma-separated values
            If VB.Left(values(0), 4) = "User" Then
                'Header line (ignore this row)
                validity = False
```

```
            Else
                validity = True
                Survey_Time = LTrim(values(1).ToString)
                Survey_Iteration = Val(values(2))
                Iteration_Time = LTrim(values(3))
                If Iteration_Time = "null" Then
                    Iteration_Time = Survey_Time
                End If
                latitude = Val(values(4))
                longitude = Val(values(5))
                speed = Val(values(6))
                direction = Val(values(7))
                If Platform = "a" Then
                    accuracy = Val(values(8))
                    gpsTime = LTrim(values(9))
                Else
                    accuracy = Val(values(9))
                    gpsTime = LTrim(values(10))
                End If
            End If
            If longitude = 0 Or latitude = 0 Then validity = False
            'Write data to the output file
            If validity = True Then
                outputstr1 = UserID & ", " & Survey_Datestamp & "," & Survey_Iteration & "," & Iteration_Time & ", " & row &
"," & latitude & "," & longitude & "," & speed & "," & direction & "," & accuracy & "," & gpsTime
                WriteLine(10, tripID, UserID, Survey_Datestamp, Survey_Iteration, Iteration_Time, row, latitude, longitude,
speed, direction, accuracy, gpsTime)

                If old_Iteration <> Survey_Iteration Then
                    WriteLine(20, tripID, UserID, Survey_Datestamp, Survey_Iteration, Iteration_Time, latitude, longitude,
speed, direction, accuracy, gpsTime)
                    old_Iteration = Survey_Iteration
                End If
                row = row + 1
            End If
        Loop
        oldTripID = tripID
        old_Iteration = -1

        FileClose(11)
        fn = Dir()
    Loop

    FileClose(10)
    FileClose(20)

    '-------------- PROCESS _TextInput FILES ----------------

    FileOpen(12, myDir & "\Aug14_TextInput.csv", OpenMode.Output)

        WriteLine(12, "TripID", "UserID", "Survey_Datestamp", "Survey_Iteration", "Text_Input")
    fn = Dir(myDir & "\*_TextInput*.txt")

        Do While fn <> ""
            textInput = ""
            filestrings = Split(fn, "_")
            UserID = filestrings(0)
            If InStr(filestrings(1), "-") > 0 Then
```

```
                    Survey_Datestamp = "2014" & VB.Right(filestrings(1), 2) & VB.Left(filestrings(1), 2) & VB.Left(filestrings(2),
2) & VB.Right(filestrings(2), 2)
                    'Platform = "i"
                Else
                    Survey_Datestamp = VB.Left(filestrings(1), Len(filestrings(1)) - 2)
                    'Platform = "a"
                End If
                Survey_Datestamp = "[" & Survey_Datestamp & "]"
                tripID = UserID & Survey_Datestamp
                FileOpen(13, myDir & "\" & fn, OpenMode.Input)

                Do While Not EOF(13)
                    inputstr = LineInput(13)
                    If inputstr <> "" Then
                        values = Split(inputstr, ",")
                        x = values.Length - 1
                        validity = True
                        'Stop
                        If LTrim(values(0)) <> UserID Then
                            textInput = inputstr
                            Survey_Iteration = filestrings(3)
                        Else
                            validity = True
                            Survey_Iteration = Val(values(2))
                            textInput = values(4)
                        End If
                    Else
                        validity = False
                    End If

                    'Write data to the output file
                    If validity = True And textInput <> "" Then
                        outputstr1 = UserID & ", " & Survey_Datestamp & "," & Survey_Iteration & ", " & inputstr
                        WriteLine(12, tripID, UserID, Survey_Datestamp, Survey_Iteration, textInput)
                    End If
                Loop
                FileClose(13)
                fn = Dir()
            Loop
            FileClose(12)
            Label1.Text = "Status: Complete"
            Button1.Text = "Close" : Button1.Enabled = True
    End Sub
End Class
```