getThere: Connecting Android users with the Cincinnati bus system

By

Emily Bradford

Submitted to
the Faculty of the Information Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

University of Cincinnati
College of Engineering & Applied Science

May 2011
getThere: Connecting Android users with the Cincinnati bus system

by

Emily Bradford

Submitted to
the Faculty of the Information Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Technology

© Copyright 2011 Emily Bradford

The author grants to the Information Technology Program permission
to reproduce and distribute copies of this document in whole or in part.

___________________________________________________ __________________
Emily Bradford       Date

___________________________________________________ __________________
Dr. Hazem Said, Faculty Advisor     Date
Acknowledgements

I would like to thank all of those involved in the process of developing my Senior Design project this year. Without the support of these people, I would have found the task overwhelming and insurmountable. I would like to thank Dr. Hazem Said for helping me break the process out into manageable, workable pieces, and for providing a sense of perspective on what is reasonable and expectable, and what is reaching beyond the scope of the project. I would also like to thank Dr. Said for being available whenever I needed to ask questions, and for being supportive when I came to him with problems.

I would like to thank Dan Pinciotti, for his assistance in helping me become accustomed to the Android platform. This project was my introduction to Android, and without Dan’s patience and willingness to help, I would have found accomplishing this task to be much more difficult than it was.

I would also like to thank Jessica Bradford, for her assistance in designing the UI components of getThere. Her input and experience as a professional graphic designer helped improve my initial user interface design.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
<td>3</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>4</td>
</tr>
<tr>
<td>List of Figures</td>
<td>5</td>
</tr>
<tr>
<td>Abstract</td>
<td>6</td>
</tr>
<tr>
<td>1. Project Description and Intended Use</td>
<td>7</td>
</tr>
<tr>
<td>1.1 Statement of the Problem</td>
<td>7</td>
</tr>
<tr>
<td>1.2 Description of the Solution</td>
<td>9</td>
</tr>
<tr>
<td>1.3 User Profile</td>
<td>12</td>
</tr>
<tr>
<td>2. Design Protocols</td>
<td>13</td>
</tr>
<tr>
<td>3. Deliverables</td>
<td>21</td>
</tr>
<tr>
<td>4. Project Planning</td>
<td>22</td>
</tr>
<tr>
<td>4.1 Budget</td>
<td>22</td>
</tr>
<tr>
<td>4.2 Timeline</td>
<td>23</td>
</tr>
<tr>
<td>4.3 Software</td>
<td>24</td>
</tr>
<tr>
<td>4.4 Hardware</td>
<td>25</td>
</tr>
<tr>
<td>5. Proof of Design</td>
<td>25</td>
</tr>
<tr>
<td>5.1 Route calculation</td>
<td>25</td>
</tr>
<tr>
<td>5.2 Direct the user to the bus stop</td>
<td>30</td>
</tr>
<tr>
<td>5.3 Save route information for re-use</td>
<td>32</td>
</tr>
<tr>
<td>5.4 Use and re-use saved locations</td>
<td>35</td>
</tr>
<tr>
<td>5.5 Support wide range of users through help tools</td>
<td>39</td>
</tr>
<tr>
<td>5.5.1 Help articles</td>
<td>39</td>
</tr>
<tr>
<td>5.5.2 Configurable help options</td>
<td>42</td>
</tr>
<tr>
<td>5.6 Support email</td>
<td>47</td>
</tr>
<tr>
<td>5.7 Navigation</td>
<td>48</td>
</tr>
<tr>
<td>6. Testing</td>
<td>49</td>
</tr>
<tr>
<td>7. Conclusions and Recommendations</td>
<td>55</td>
</tr>
<tr>
<td>7.1 Conclusion</td>
<td>55</td>
</tr>
<tr>
<td>7.2 Recommendations</td>
<td>55</td>
</tr>
<tr>
<td>8. References</td>
<td>58</td>
</tr>
<tr>
<td>9. Appendices</td>
<td>59</td>
</tr>
<tr>
<td>9.1 Appendix A – Help articles</td>
<td>59</td>
</tr>
<tr>
<td>9.2 Appendix B – Testing results</td>
<td>65</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1 Use case diagram ............................................................................................................. 14
Figure 2 Database diagram ............................................................................................................. 17
Figure 3 App Class Diagram ......................................................................................................... 19
Figure 4 Server class diagram ....................................................................................................... 20
Figure 5 Gantt chart - Condensed project timeline ....................................................................... 23
Figure 6 New route screen ............................................................................................................ 26
Figure 7 Directions screen, no transfers ....................................................................................... 27
Figure 8 Directions screen, single transfer ................................................................................... 28
Figure 9 Processing route request ............................................................................................... 29
Figure 10 Could not parse address error ..................................................................................... 30
Figure 11 Augmented Reality view .............................................................................................. 31
Figure 12 Google Map stop location ............................................................................................ 32
Figure 13 Save route screen .......................................................................................................... 33
Figure 14 Saved routes .................................................................................................................. 34
Figure 15 Saved route directions ................................................................................................. 35
Figure 16 Saved locations ............................................................................................................. 36
Figure 17 Save location screen .................................................................................................... 37
Figure 18 Location name in use error .......................................................................................... 38
Figure 19 Delete location .............................................................................................................. 39
Figure 20 Help articles screen ...................................................................................................... 40
Figure 21 Help article ................................................................................................................... 41
Figure 22 Options screen .............................................................................................................. 42
Figure 23 Saved routes screen general help .................................................................................. 44
Figure 24 Help icon ...................................................................................................................... 44
Figure 25 Input help ...................................................................................................................... 45
Figure 26 Save location used ........................................................................................................ 46
Figure 27 Use current location ..................................................................................................... 47
Figure 28 Main menu with email link .......................................................................................... 48
Figure 29 Navigation menu .......................................................................................................... 49

Table 1 Budget .............................................................................................................................. 22
Table 2 Software used ................................................................................................................... 24
Table 3 Hardware used .................................................................................................................. 25
Table 4 Testing summary .............................................................................................................. 52
Abstract

The Metro bus service provided by SORTA provides public transportation to the greater Cincinnati area. Metro provides paper maps and schedules, and an outdated trip planner, but no modern tools for using the bus system. To bring Cincinnati’s public transportation in line with tools in Chicago, New York, Boston, and London, a smartphone app was designed to provide on the go bus route calculation and management. Get There, an android application, incorporates GPS location-driven functionality, a simple interface, and Augmented Reality into one tool to navigate the Metro bus system.

Get There provides the capability to generate passenger route information as needed, save and view previously generated directions, store and re-use locations, find bus stops, and email routes and suggestions. This functionality will allow users to use public transportation in Cincinnati without stress. No more struggling to decipher a timetable, or only knowing one bus at one time. Get There will allow users to fluidly interact with Cincinnati in a way that fits their lives. This is facilitated by configurable help options, articles that assist users with the bus system, and helpful streamlining features, such as automatically using your current location, or saved locations.

At the end of the day, Cincinnati has a quality bus system. Get There provides a modern, flexible, quality tool to connect users with that bus system, via their Android device.
1. Project Description and Intended Use

1.1 Statement of the Problem

Travel today is a remarkable thing, and the services made available in major metropolitan areas to assist with both local and tourism-based transportation have empowered travellers to forego a cab or a rental car, and take to public transit. In New York, tourists can be seen navigating the subway and the bus system with relative confidence and ease, smartphone firmly in hand. In Chicago and Boston, riders on the trains or buses may have paper schedules clutched in hand, but are nearly just as likely to have it on their mobile device. In London, travellers can go from Buckingham Palace to the Eye, to Speaker's Corner without trepidation, locating transit stops and changeovers with confidence thanks to being on the cutting edge of transit-focused mobile development (1) (3) (4). Major cities all over the world have specialized apps and tools to help locals and visitors get around with public transportation, but not in Cincinnati. Cincinnati is not currently being served by a public transportation app or mobile tool (12). The problem associated with this situation is two-fold: first, people in Cincinnati are not being empowered to use the public bus system with confidence and ease, and second, Cincinnati is lagging woefully behind on the national stage of public transit, by relying solely on printed schedules and a cumbersome web-based trip planner (2).

Public transportation in the Cincinnati metro area is ubiquitous. The city seemingly has bus stops on every street, and one can frequently see buses carting passengers around, through the maze of city traffic. What may not be readily apparent, however, is that most of the passengers on those buses are regular riders. They are people who use the bus as a primary mode of transportation, and have either become intimately comfortable with the intricacies of reading a bus schedule, or have simply learned their route to the point of being able to use it comfortably.
According to the U.S. Census Bureau, use of public bus transportation in Cincinnati is on the decline, and has been for several years (8). Recent surveys indicate that this can be linked to a perceived lack of convenience and accessibility on the part of potential riders (6). SORTA, the Southwestern Ohio Regional Transit Authority, operates the Metro bus system in Cincinnati. Currently, they provide their maps and schedule information via printed brochures and online pdfs, as well as an online trip planner, which require the user to know from which stop to which stop they want to travel (11). Because of this, the only way for someone to take the bus is to have planned it out ahead of time. The user would need to have plotted a route and looked up schedules, or used the trip planner, for the specific route they want to take. This is limiting, and does not provide convenience and ease of access (6). SORTA publishes facts about their performance, and has indicated that they serve 20% of the downtown Cincinnati workforce. That is a remarkable number, and speaks to the valuable service that SORTA provides. This is also directly correlated to the fact that those riders are statistically likely to only use the bus for the commute to and from work, and no other time (8).

Given the quality of the transportation service being provided, one must turn to other factors to explain the disparity between the image of public transportation in Cincinnati versus other major metropolitan areas on a national stage. The Federal Department of Transportation, through their Intelligent Transportation System initiatives, have been pushing for more sophisticated user tools to be created and released for public transportation (2). Many major cities, like New York, Chicago, San Francisco, Boston, Atlanta, and scores of others have answered this push with highly usable smartphone applications that allow users to plan public transit trips on the go, without having to print off schedules or decode maps and routes (3) (4). At the point of inception, the bus system in Cincinnati had not made its transit data publicly available to assist in
any potential development (12). Google, with its recently released transit assistance feature integrated with Google Maps, until recently, had no information for transit in Cincinnati. As other metropolitan areas move ahead by leaps and bounds, providing riders with tools to plan trips, figure out schedules, and determine the most convenient stops based on where they want to be, and not the stop at which they want to arrive, Cincinnati is using technology that was expected and still clumsy ten years ago (1).

It was in this context, with a deficit of confidence in being able to use the bus system if one has not done so on multiple previous occasions, and the way in which Cincinnati was falling behind the national transportation front-runners in terms of outside services and technologies available, that I chose to develop an application that provides users with the ability to use the bus system on the go. I wanted to bring the go * metro bus system into the 21st century, improve ridership rates, and hopefully push Cincinnati toward being a more public transportation friendly city, in line with current federal goals as well as the market in other major metropolitan areas.

1.2 Description of the Solution

In response to the lack of modern, accessible, and intuitive methods of using the SORTA go * metro bus system, the suggested solution was a smartphone application that allows users to generate instructions for using the bus system to get from point A to point B on an as-needed basis. Get There, the proposed app, meets the current needs being left unfulfilled by providing a usable platform and interface, as well as dynamic functionality to facilitate use of the public transit system in Cincinnati.

In a high level view, Get There is an Android app that utilizes the transit data of SORTA to create bus route information. The app connects with a web-service that communicates with the
Get There database, and handles the route calculation, which it then passes to the mobile device. The user uses the app to input a starting location and an ending location, and the app responds with the bus route instructions needed to use the go * metro system to travel from the first location to the second location. The app also features an Augmented Reality component, which incorporates the phone's video capabilities, compass, and GPS locator to overlay a point of interest marker onto the video feed to direct the user to the correct bus stop. The user is able to use his or her current location, or define a different location. The user is also able to save locations for easy re-use, and to review previous route information. Future iterations of Get There will also support sharing a generated route via email.

After the user submits the location information, the app returns the bus stop nearest to the user that should be used, the time at which the next appropriate bus will arrive, the number of transfers the trip will require, and the approximate time the user is expected to arrive at his or her destination. The user is also able to provide locations from a list of saved locations.

When the route is generated, the app then directs the user to the correct bus stop. Augmented Reality is a development technique that uses a video feed and adds information to it. This has been used in other applications to do everything from indicating points of interest to providing a field for video game controls. In Get There, the Augmented Reality feature uses the video feed from the mobile device to overlay a point of interest marker for the bus stop the user needs in order to begin his or her trip. The point of interest marker shifts to indicate the location of the bus stop.

The user has the option to save route directions, along with a name, so that he or she can reuse them easily at a later date. This route information is saved in on-board memory so that the user can reuse a specific route if he or she cannot connect to the internet to use the live database.
Get There is an adequate response to the current problems in bus system accessibility in Cincinnati for the following reasons:

1) Get There eliminates the need to interpret confusing bus schedules

   One of the major problems associated with using the bus system was the complicated and confusing route system that people would have to figure out how to navigate. Determining transfers, arrivals, and which stops belong to which lines have been significant barriers to people feeling confident in using public transportation. Get There responds to this problem by allowing users to indicate where they want to begin, and where they want to end, and providing simple instructions on how to use the bus system to reach their destination. As potential riders have indicated that a simpler way to use the bus system would encourage them to actually travel via bus, this has the direct potential to increase ridership (6).

2) Get There allows for "on the fly" route information generation

   Under the current system, if a potential passenger wanted to use the bus system, he or she would have to sit down with a series of bus schedules and plot out a route, or use the SORTA website to figure out a route using their cumbersome route tool, which is not optimized for mobile access, or they would need to call and have an employee plot out a route for them (11). This puts a burden of planning on the potential passenger that limits that person to a particular schedule and series of activities. Get There allows the user to generate route information wherever he or she happens to be, at any time. This allows the user the freedom to travel without the stress of being bound to a particular schedule and a particular time. Get There will empower users to travel via bus without
the stress of potentially missing a bus and being stranded.

3) Get There employs a popular platform, and current technology to provide travel information

Android phones are becoming a greater part of the market than seen in years past, and are slated to take over an even larger market share in coming years (13). Augmented Reality applications have been getting significant attention in trade magazines for a while now, and are becoming an increasingly popular feature to include in phones, particularly in displaying points of interest (7)(12). Combining these aspects into a tool that allows users to easily make use of the bus system in Cincinnati will elevate Cincinnati's public transportation services to a level which users have come to expect in London, New York, San Francisco, and other major metropolitan areas.

1.3 User Profile

The main users for this application fall into three general categories: End users, data providers (the SORTA Metro system), and the application maintainer. As the latter two categories of user deal with the data in the back end, and the code behind the scenes rather than interacting with the user-facing functionality, the focus herein will be largely on the end users. Although the User is the only significant actor in the Get There system, this role encompasses a variety of people within its expected demographics. The potential end users for this application are smartphone users who are interested in travelling by public transportation (specifically by bus), and are perhaps unfamiliar with Cincinnati. These users include college students, travellers and visitors, and business people who work downtown, among others (13).
The end users of Get There are the owners of smart phones who have need of public transportation in Cincinnati. Reasons for such could range from work location, school location, lack of car, one-off use to travel to a downtown event, et cetera. This user will generate a bus route via Get There from an Android phone, while connected to either a WiFi network or a data plan. They will want to reuse common locations and routes, and will need to be able to view a previously saved route when not in a network accessible location. They will want to prepare for future bus trips as well as spur of the moment trips.

The expected end users of this application will range in frequency of use from once every few months, to one period of intense use, to every few days or weeks, in an irregular pattern.

2. Design Protocols

Get There, due to the nature of smartphone apps, has one significant human actor in its end use. The actor in this case is the User. The User interacts with the application to generate bus route information, load and store location information, find the correct bus stop, and share bus routes with others. The activities for the User are illustrated in Figure 1, a use case diagram. Also depicted is the interaction between the SORTA data source and the database back end for the system. This SORTA actor is an actor with respect to the system, but interacts with the back end and is not considered, therefore, an end user.
The way in which the end user completes these tasks is informed heavily by the limitations assessed in the user profile. The user's schema for determining route information in general is likely highly influenced by web-based direction generation tools such as Google Maps and MapQuest. With this in mind, the interface is designed in such a way that it draws on the user's familiarity with those tools. The interface is kept as simple as possible, to avoid distracting from the end goal of the application and confusing the user. The interface steps through the process of providing the required information to generate the route in an intuitive way. It does not focus on

Figure 1 Use case diagram
the steps required to generate a route, but instead gets the information in an "I am here, I want to go there" pattern of information gathering (9).

The user enters the product via an icon on the touchscreen menu of the device. They move through the product via clearly labeled and intuitively placed buttons. The design implements the intuitive touch and drag input for scrolling, but otherwise interacts via buttons. Editable fields follow the user schema introduced by other apps and websites. That is, if a field is editable, it looks like a text box, and will not appear as a label. Buttons appear to be three dimensional, and do not confuse the user by appearing as flat, clickable images. The interface design models existing tools in so far as those tools represent existing schema, but deviates in order to create an intuitive, streamlined method of acquiring bus route information.

Get There is heavily customizable in terms of the amount of help provided in the application. Because this application is intended to support a frequency of use that may range from every day to once every few months or years, as well as task experience that can be described as non-existent to relatively familiar, the application needs to be flexible in allowing users to "pick up and go" in a way that best suits their abilities. A user who only uses the app once every few months may need more assistance than a user who uses it every day or so. The infrequent user may need to be prompted for how to put in the information, and how to use the resulting route information. This ranges from step by step instructions for how to maneuver a transfer, to reminding the user about how to handle transfer fares. As helpful as one user may find this information, however, another, more frequent or advanced, user may find it frustrating and an obstacle to easy use of the application.

It is with this situation in mind that the developer incorporates a highly customizable set of help options. The user decides when and if prompts display, and how much detail is incorporated
into their route information. Even with these features disabled, however, users are able to access this level of help without backing out of the route generation process entirely. With that in mind, there are intuitive question mark icons throughout the application that, when clicked, present the user with help information along the way. The user can also view help files from the main menu of the application. These help files include a tutorial for using the bus system, and information about how to handle transfer fares.

Because assuming that an application will be perfect the first time out is both arrogant and foolish, this application includes a built in method for users to provide feedback and feature requests within the app. This aspect of the design is meant to encourage users to feel ownership of the tool, and to allow the developer to adjust the tool as necessary as the needs of its users change.

The actual look and feel of the interface was considered carefully. Color choices, design layouts, and screen progression were designed via a partnership between the developer and a professional graphic designer who has agreed to assist in the project at no cost. Utilizing the assistance of a professional graphic designer helps ensure that the look and feel of the application is professional, current, and visually interesting. Colors have been chosen to mimic the go * metro system, to provide consistency in use between the app and the bus system.

The database structure used is being based largely off of the standard set forth by Google with their Transit Data Feed model. This model has been stripped of some components in order to fit the scope of the current project, and some elements are on the verge of elimination, due to algorithm processes. The modified structure is depicted in Figure 2, on the following page.
Get There: Database diagram

**calendar**

<table>
<thead>
<tr>
<th>PK</th>
<th>service_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>monday</td>
</tr>
<tr>
<td></td>
<td>tuesday</td>
</tr>
<tr>
<td></td>
<td>wednesday</td>
</tr>
<tr>
<td></td>
<td>thursday</td>
</tr>
<tr>
<td></td>
<td>friday</td>
</tr>
<tr>
<td></td>
<td>saturday</td>
</tr>
<tr>
<td></td>
<td>sunday</td>
</tr>
<tr>
<td></td>
<td>start_date</td>
</tr>
<tr>
<td></td>
<td>end_date</td>
</tr>
</tbody>
</table>

**stops**

<table>
<thead>
<tr>
<th>PK</th>
<th>stop_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>stop_name</td>
</tr>
<tr>
<td></td>
<td>stop_lat</td>
</tr>
<tr>
<td></td>
<td>stop_lon</td>
</tr>
<tr>
<td></td>
<td>zone_id</td>
</tr>
</tbody>
</table>

**routes**

<table>
<thead>
<tr>
<th>PK</th>
<th>route_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>route_short_name</td>
</tr>
<tr>
<td></td>
<td>route_long_name</td>
</tr>
<tr>
<td></td>
<td>route_desc</td>
</tr>
</tbody>
</table>

**trips**

<table>
<thead>
<tr>
<th>PK</th>
<th>trip_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>route_id</td>
</tr>
<tr>
<td></td>
<td>service_id</td>
</tr>
<tr>
<td></td>
<td>trip_headsign</td>
</tr>
<tr>
<td></td>
<td>direction_id</td>
</tr>
<tr>
<td>FK2</td>
<td>trip_id</td>
</tr>
<tr>
<td>FK1</td>
<td>route_id</td>
</tr>
</tbody>
</table>

**stop_times**

<table>
<thead>
<tr>
<th>PK</th>
<th>trip_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>stop_id</td>
</tr>
<tr>
<td></td>
<td>arrival_time</td>
</tr>
<tr>
<td></td>
<td>departure_time</td>
</tr>
<tr>
<td></td>
<td>stop_sequence</td>
</tr>
<tr>
<td></td>
<td>exception_id</td>
</tr>
</tbody>
</table>

**exceptions**

<table>
<thead>
<tr>
<th>PK</th>
<th>exception_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>route_id</td>
</tr>
<tr>
<td></td>
<td>exception_desc</td>
</tr>
</tbody>
</table>

**fare_rules**

<table>
<thead>
<tr>
<th>PK</th>
<th>fare_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>route_id</td>
</tr>
<tr>
<td></td>
<td>origin_id</td>
</tr>
<tr>
<td></td>
<td>destination_id</td>
</tr>
</tbody>
</table>

**fare_attributes**

<table>
<thead>
<tr>
<th>PK</th>
<th>fare_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>price</td>
</tr>
<tr>
<td></td>
<td>transfers</td>
</tr>
<tr>
<td></td>
<td>transfer_duration</td>
</tr>
</tbody>
</table>

**Get There: Database diagram**

Figure 2 Database diagram
The class structure of the app is based on the principle that classes should model real world items and actors. In this vein, data transfer objects which are used by the server application to model transit information in the service layer are coded to mimic bus routes, stops, stop events, trips, et cetera. Most of this calculation and modeling will be performed on the server side of the application, while the app itself will be concerned mostly with retrieving and displaying information. To that end, the app is weighted toward user interface components, and the server is weighted toward business and data logic. The exception to this is that app has the capability to store and retrieve route and location information that has been saved locally, and uses service and data layer classes to access data persistence on the device.
Figure 3 App Class Diagram
Figure 4 Server class diagram
Figures 3 and 4 illustrate the class structure for the server and mobile device aspects of the application. The basic flow of logic is a standard call and request web service design pattern. The web service is running, waiting for requests for information, and returns that information via HTTP response headers to the requesting application. The app receives this information, and processes it into a displayable form.

3. Deliverables

The main product of this project is an Android mobile app, which communicates with a web service that queries SORTA transit information to generate passenger routes given the time, origin, and destination requirements of the user. The deliverables for this product are as follows:

- The app requests and displays bus route information to travel from a specified point A to point B. This will include the ability to use the user’s current location or a specified desired location.
- The app provides an Augmented Reality service to direct the user from their current location to the correct beginning bus stop, using the Android’s video display and a Point of Interest marker.
- The app determines the correct beginning and ending bus stops for the user, based on provided address information.
- The app allows the user to save locations on their mobile device for re-use at later times.
- The app allows the user to view and manage saved locations.
- The app allows the user to save generated route information locally, for use without an internet connection. The app allows the user to name the route and to add comments about the route for management purposes.

- The app provides help and assistance features, and allows the user to toggle these features on and off.

- The app provides basic bus riding and transfer information for new riders.

4. Project Planning

4.1 Budget

<table>
<thead>
<tr>
<th></th>
<th>Market Cost</th>
<th>Incurred Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Netbeans</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Wikitude API</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GIMP</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MySQL</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16 Gb Flash Drive</td>
<td>22.99</td>
<td>22.99</td>
</tr>
<tr>
<td>HTC Dream</td>
<td>200.00</td>
<td>70.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>222.99</strong></td>
<td><strong>92.99</strong></td>
</tr>
</tbody>
</table>

*Table 1 Budget*

The budget for the development of getThere was relatively minimal. Free development environments, graphics tools, databases, and APIs were employed. The only points of cost were the hardware involved. The development PC is not included in this table as it was the private property of the developer. The testing device, the HTC Dream or G1, as it is better known, came in well below market price.
4.2 Timeline

As shown in the Gantt chart above (Figure 5), the development of getThere took place from the middle of Fall quarter 2010 and proceeded through May 2011. The tasks for this project have been split out among several conceptual iterations, the first of which form the core of the project prototype. The project schedule focused on the more critical, high risk elements to go first in development. For Get There, this focused on getting the underlying architecture into place. The database was a critical part of the app to develop, as it governed the development of the route calculation algorithm. The database was the first piece constructed. Test data was
gathered from the SORTA website, and three routes were simplified and their data incorporated into the database.

Next, the basic route calculation algorithms were written, by first developing the single route, then one point of transfer. Because this functionality is the basis for all other functionality in the system, it was deemed business critical and placed at a high priority. Next the Augmented Reality piece was developed, as it was considerably high risk. All other functionality is based off of the aforementioned pieces, and therefore was left until after the prototype was developed. The route and location management, help system, and other components were scheduled for Spring quarter. These aspects were contingent upon the success of the earlier elements, and provide value and functionality, but built on existing capabilities.

Spring quarter involved developing the non-route functionality of the app, in addition to modifying the route calculation algorithm for improved efficiency. Attention was paid to the UI and navigational capabilities. Saved Location and Route information was implemented, along with the array of help configuration options.

4.3 Software

The software used in development of getThere is shown in the table below (Table 2).

<table>
<thead>
<tr>
<th>Name</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse</td>
<td>Development environment for the mobile app</td>
</tr>
<tr>
<td>Netbeans 6.9.1</td>
<td>Development of the web service</td>
</tr>
<tr>
<td>GIMP 2.0</td>
<td>Graphics manipulation</td>
</tr>
<tr>
<td>MySQL</td>
<td>Database management</td>
</tr>
<tr>
<td>Apache</td>
<td>Server</td>
</tr>
<tr>
<td>Google Maps API</td>
<td>Location services</td>
</tr>
<tr>
<td>Wikitude API</td>
<td>Augmented Reality</td>
</tr>
</tbody>
</table>

Table 2 Software used
4.4 Hardware

The hardware used in development of getThere is shown in the table below (Table 3).

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acer Aspire 5600 laptop</td>
<td>Development, server hosting</td>
</tr>
<tr>
<td>HTC Dream</td>
<td>Test android device</td>
</tr>
</tbody>
</table>

Table 3 Hardware used

5. Proof of Design

The following section contains a breakdown of the functionality of getThere and how this functionality indicates the successful completion of a stated deliverable.

5.1 Route calculation

getThere supports bus route calculation for passenger routes with up to one point of transfer. To generate this route, the user begins with the New Route screen, shown in Figure 6.
Figure 6 New route screen

This screen (Figure 6) displays the app’s title graphic, which is displayed on every screen except for directional screens. It also allows the user to give input for the address to travel from and the address to travel to. This layout employs high contrast, logical grouping, clear labels, and easily recognized tools. The button, clearly labeled, triggers a response from the app to make a request to geocode the addresses provided, then passes that information to the getThere web service. This service determines the closest bus stops, then calculates the route between them, then displays the directions, in a list view as shown in Figure 7, below.
Figure 7 illustrates how directions appear when the user calculates a route without any transfers. When a transfer is needed the directions are displayed as shown in Figure 8.
Figure 8 Directions screen, single transfer

This illustrates that the app is capable of providing route calculation for passenger routes with zero or one transfer. getThere also provides users with feedback during the route calculation process. While the route is calculating, the user is assured that the app has not hung up by use of a progress dialog, shown in Figure 9.
If the user provides an improperly formatted address, the user is met with a message informing the user of the proper address format, as shown in Figure 10. This is accomplished by setting a threshold for the number of requests to make of the geocoding service, as well as parsing the response from the Geocoder object.

Figure 9 Processing route request
5.2 Direct the user to the bus stop

The next deliverable involves directing the user to his or her bus stop, to begin their trip. getThere provides two mechanisms for doing this. The first mechanism involves the Wikitude Augmented Reality API. The app takes the location of the bus stop and passes it to a specialized activity from the Wikitude API, which uses the device’s camera, accelerometer, GPS, and compass and overlays a point of interest marker on the screen to indicate the direction of the bus stop, relative to the user’s location and orientation. This marker has been customized to mimic the getThere launch icon. The Augmented Reality view is shown in Figure 11.
The second mechanism for directing the user to the starting bus stop involves using the Google Maps API. This mechanism involves creating a map of the area around the starting bus stop, and placing a marker over the stop’s location. This is accomplished by building a URI request for the Google Maps API. The location of the stop is specified, the marker is coded, and the level of zoom and scope is programatically set. The request can then either be sent to the browser or the on-board Maps app, to be displayed as shown in Figure 12.
5.3 Save route information for re-use

The next deliverable involves streamlining use of the app so that the user does not have to spend time generating the same information over and over. getThere allows users to save directions from a generated route, associate those directions with a name and a description, and review them when not connected to the internet. On the directions screen, there is a button marked Save Route, shown in Figure 8. When the user clicks this button, the Save Route screen is opened, shown in Figure 13.
This screen accepts user input in the form of a name and a description. The app then associates these with a PassengerRoute object, requests the save file string format from the object, and then appends this to the saved routes file, located in the app’s directory in the device’s memory. When the Saved Routes screen (Figure 14) is loaded, this file is read, parsed, and turned into an array of PassengerRoute objects, which are then fed into a list view in the activity, and displayed as a list of Route names.
When the user taps the name of one of these saved routes, the app re-loads the direction viewer screen with the route, with a few minor changes. As the idea is that this is for re-use, directions to the stop are somewhat superfluous, so those buttons are omitted, as shown in Figure 15. The only button is the Delete button. Tapping this button causes the app to strip the selected route out of the collection of routes, and re-save. getThere then re-opens the Saved Routes screen, with the array of routes freshly created, and missing the deleted route.
5.4 Use and re-use saved locations

In order to prevent users from having to enter and re-enter the same small group of addresses over and over again, getThere includes a usability feature to allow storage of addresses with friendly names. These locations can be used in lieu of street addresses when generating a route. To set up a new location, the user will select the Locations screen, shown in Figure 16. This screen will display a list of the currently saved locations, displaying both the name and the street address. If there are no currently saved locations, the list will be blank.
From the Locations screen, the user can create a new location via the Add New Location button. This will open the Location Creator screen, shown in Figure 17. This screen provides input tools for accepting a string value to associate as the name of the new location, and an input to accept a string value to associate as the address portion of the new location.
When the user clicks save, the name is checked for its relative uniqueness against the other names, to prevent duplicates and errors. If there happens to be a location with the same name in use, the app displays an error message to the user, and prompts the user for another name to associate, as seen in Figure 18.
When the location is successfully saved the app re-opens the Locations screen, and the
new named location will be added. To delete a location, for whatever reason, the user will tap the
associated name entry, and a dialog will open prompting the user to delete or cancel, as shown in
Figure 19. If the user selects delete, the Locations screen is reloaded with the updated list. The
delete functionality mimics the delete functionality for the Saved Routes screen very closely,
employing a HashMap<String, StreetLocation> construct.
5.5 Support wide range of users through help tools

5.5.1 Help articles

In order to allow an unfamiliar user to become more acclimated to the app and to using the Cincinnati bus system, getThere includes a small library of help articles. These articles are separated into subject matter, either centered around use of the app itself, or centered around generic use of the bus system. The articles display in a fashion similar to the setup of the Location and Saved Route screens, as shown in Figure 20. The Help screen associates an array of HelpArticle objects with the list adapter in the activity’s view.
In order to display one of the articles, the user will simply tap the name of the appropriate article, and the listener will build an alert dialog with the contents of the HelpArticle object displayed, as shown in Figure 21. For the full text of the included help articles, please see Appendix A. The alert dialog displaying the help article provides only one button, to close, because the help articles are static, and immutable by the end user.
The help articles are intended to provide the end user with a higher level of comfort and familiarity with the app and the Cincinnati bus system. They are not intended to be use-along guides, but rather to provide a base knowledge from which the user can expand his or her skillset. Whether wanting to save a new location or determine how one calculates a trip fare, the help articles are designed to illuminate. For help options that directly impact a user’s experience of getThere, the Options screen is the appropriate place.
5.5.2 Configurable help options

getThere is intended to support a wide variety of users, in terms of both background and skillset. To this end, getThere includes an array of configurable help options to shape the user’s experience when creating and managing routes and locations. These help options are managed in a simple construction of name, Boolean pairs. These individual options, which span every screen in the app, and control multiple facets of user experience, are grouped into five aggregate settings. These settings are displayed on the Options screen and are each associated with a checkbox, to indicate enabled or disabled, as shown in Figure 22.

![Figure 22 Options screen](image)
When the user enables the checkboxes associated with these categories, and then selects Save Settings, it ripples across the saved help configuration, flipping settings to on or off as appropriate. By default, none of these features are enabled on getThere, preferring instead for the user to define his or her own experience. If the user selects the first option, the enable all assistance option, all checkboxes are taken as enabled, and all help settings across the app are enabled.

If the user enables the second option, requesting that a brief overview of each screen be given as the user loads it, then when a user opens another screen on the app, a dialog will be opened with a general description of the page, including guidelines for use and how it fits into the overall intent of the app. The resulting message box when the user opens the Saved Routes screen with this option enabled is shown in Figure 23.
If the user enables the help with form input option, then throughout getThere, wherever there is a place for user input, the app will also place the help icon (Figure 24). This icon can be clicked, thereby displaying an instructional message about filling in that particular input field. Figure 25 shows an example of this, in the New Route screen, wherein the “from” address field is accompanied by a message about address formatting.
If the user enables the prompting for new saved locations option, then every time he or she runs a new route with a manually entered address, the app will ask the user if he or she would like to save that address, and provides a modular Save Location form. This follows the same rules and conventions of uniqueness as the other Location Creator. Figure 26 shows the Save Location dialog.
If the user enables the last option, the use current location option, then by default the app is set to use the user’s current GPS location as the from address, instead of a manually entered or saved location. The device sets up a provider, listener, and activity specifically to monitor the user’s current location, and then passes that information directly to the web service to generate the route, since the coordinates would not need geocoding. The New Route screen with the Use Current Location option set is shown in Figure 27.
These options, taken together, help users customize the way in which they interact with getThere. From basic app assistance to assistance with location services, the help configuration options lend a certain flexibility to functionality provided by the app.

5.6 Support email

Because getThere is intended to be a highly adaptive, interactive tool, there has been an effort made to facilitate communication between end users and the developer. The app makes use of the on-device email client, and provides a device-readable email address on the Main Menu screen of the app. This link, shown in Figure 28, is clearly labeled, and provides a direct link between users actually using the app, and the developer improving the tool. If a user finds a bug,
problem, or feature request, he or she can easily send an email to the developer, thereby improving the overall quality of the tool and the turnaround time between need established and need met.

![Main menu with email link](image)

**Figure 28 Main menu with email link**

### 5.7 Navigation

getThere is designed with an unpredictable user in mind, in terms of navigation. The app always opens first to the screen seen in Figure 28. This is the Main Menu screen, and employs an icon style menu layout in the interest of displaying all options at once, and providing a sense of continuity between the Android brand and the app. The user can reach all five major functional areas from this screen.
Given that it can be quite unpredictable how a user will move throughout the app, a flexible method of navigation within the app was necessary. To this end, getThere provides a navigational menu launched by the device’s menu button. When the user taps the menu button from anywhere within the app, the navigational menu appears, shown in Figure 29. From this menu the user can move between all of the functional areas with ease. This is intended to encourage natural, as needed use of the app and its different features and management tools.

![Figure 29 Navigation menu](image)

6. Testing

In any product intended for an end user, unit testing is incredibly important. The initial round of testing was done by the developer as each module of code was developed. The first
round of testing done was query testing, using the MySQL command line interface to generate queries, compare the result sets against the expected results, and then measure the length of time a query took to complete. It was determined that the threshold for the length of time route processing could be allowed to occur without incurring frustration on the part of the end user increased dramatically when the progress dialog was implemented. This told the user that there was work being done, and gave the user faith that the program had not hung up.

The next round of testing was basic functionality testing for the route calculation web service. Appropriate routes were plotted, and then the previously written queries were run, through the data transfer objects and service layer objects in the web service. The results were displayed and compared against the hand-plotted routes based on the data set being used. Route calculation logic was tested, changed, and retested until there was a 100% success rate in each of the testing scenarios for route calculation, even if that success involved recognizing not being able to calculate the route.

As the app portion of the project was built, in a top-down fashion as necessitated by Android’s interesting XML and activity based architecture, each individual functional model was tested as developed. This began with basic connectivity testing as the app was first coded to consume the getThere web service. Once connectivity was established, the interface for passing route requests to the web service was tested. The HTC Dream Android test device proved invaluable during the testing of this project. Due to its reliance on location based services, the in-IDE emulator included with Eclipse could not be reliably used for testing. The physical device gave accurate representations of what the user would see, how the user would be able to manipulate the app, and what the actual connectivity issues might be in a real world scenario.
Once basic functionality testing was completed, the app was passed over to Jessica Bradford, a professional graphic designer and brand developer. Ms. Bradford reviewed the interface, styling, buttons, and flow of the navigation. She did not test for functionality, but rather for the impression that the app would give a user. Dr. Hazem Said also provided some testing in this regard. The combined recommendations by these individuals helped to shape how the interface design for getThere came together into its final iteration. The Main Menu screen had originally been composed of a single vertical column of large menu buttons. Not all options could be seen at once, and the user had to scroll down to select a new one. Dr. Said suggested changing that menu layout to an icon layout, with smaller buttons laid out near one another. This change was received well by Ms. Bradford, who reported that this was more consistent with most customer expectations. The title image logo that appears on each screen is a direct result of the review Ms. Bradford did of the interface. She insisted on consistency in design, and suggested that reusing the same elements will keep the user from feeling that the pieces of the app were disjointed.

After the initial user interface review was complete, the app was passed off to a group of people for testing and reactions. The full results of this testing can be found in Appendix B. Table 4 shows the summary of these findings.
<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Mistakes</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>~3</td>
</tr>
<tr>
<td>Most missed section</td>
<td>Options</td>
<td>N/A</td>
<td>Options</td>
<td>New Route</td>
<td>New Route</td>
<td>Options / New Route</td>
</tr>
<tr>
<td>Questions asked</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>~2</td>
</tr>
<tr>
<td>Successful Completion</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Interface rating</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>~8</td>
</tr>
<tr>
<td>Navigation rating</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>~10</td>
</tr>
<tr>
<td>Misuse rating</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>~7</td>
</tr>
</tbody>
</table>

Table 4 Testing summary

Table 4 illustrates the overall results of user testing. These users were included college aged students, with two users in their 50s, one who identified as technically savvy, one who did not, and one post-college technical professional. The college students ranged in skillset from high technical skill to basic technical skill. The T# columns indicate the data gathered from a particular tester, while the Overall column serves as an aggregate for what outcome can be most reasonably determined from the gathered data.

The total number of mistakes the user made was recorded, as this was the clearest indicator of what pieces would need redesigning for clarity, as well as what pieces would need redesigning to handle common, accidental misuse. Identifying the most missed section and the number of questions asked is also instrumental in determining where the logical user flow of the app is disrupted and the user feels he or she must guess. This information is useful in determining where to place additional help configuration options.

The successful completion metric, as well as the ratings are a high-level view of whether or not the user was able to complete the tasks, and whether or not the user was frustrated by the
process. The misuse rating does not necessarily correspond to intentional misuse, but also to how well the user felt the app responded to the way the user was trying to use it.

The process for user testing involved going through each deliverable and providing both set tasks and general “do something” tasks. Each individual was asked to complete the deliverable in a set way, and then in a freeform way. The mistakes noted correspond to a combination of both methods for each deliverable.

On the whole, user testing found that the most confusing sections were the Options and New Route screens. Users conveyed that the options did not necessarily line up with precisely what they had expected in behavior from the app. General consensus was that the options themselves were useful, but that they ought to be rephrased on the selection screen. Also, users found it best to remove the first aggregate option. Users found the New Route screen challenging as well, mostly in terms of address input and the expected format, as well as how quickly it gave error messages.

Each user was able to successfully complete route generation without interference from the tester, except for the last user. This user was self-described as having very low experience with smartphones and direction generation software. Because all other users were able to successfully complete the main objective without interference, the app appears to be largely successful. This does indicate, however, that a part of the potential userbase will need improved assistance features in order to take advantage of the main functionality of the app. While this demographic is unlikely to possess a smartphone, it is still prudent to make sure that the technically less savvy can still make use of its primary goal as a product.

Response to the user interface was largely positive. Users reported that it was intuitive, seemed logical in its progression, and was easy to read and interpret. They did suggest
integrating updated graphics and more consistent alignment. Response to navigation was wholly positive. Users liked the quick access from the menu, and reported that the order of buttons made sense. They liked being able to see all options at once. Users also liked the navigation menu attached to the menu button. They liked the convenience of not having to return to the main menu. None of the mistakes made were due to navigational errors. All of the subject areas corresponded directly to user expectations.

The most valuable lesson learned from user testing was the wide range of benevolent misuse that occurred. The New Route screen bore the brunt of it, as users attempted to use unexpected abbreviations, partial or no city names, and other manner of unexpected input. The largest complaint was that the app did not respond in a timely enough manner to incorrect user input. General consensus was that it was not such a problem that the app could not handle every variation on address that a user threw at it, but that it did not tell the user that it could not read the address until more than a minute had passed, in some cases.

User testing yielded several intended next steps:

1) Improve address parsing and respond to user input before making the Google request
2) Redesign layout with upgraded graphics and more rigid alignment
3) Rephrase options to more closely describe the result of that selection
4) Include more help icon informational spots
5) Describe the expected input before a user clicks the address text boxes
7. Conclusions and recommendations

7.1 Conclusion

At the beginning of this project, Cincinnati was suffering from a dearth of modern tools to encourage access for the Cincinnati bus system. Users had an outdated online trip planner and paper maps and schedules, but that was the limit of available tools. SORTA had not even made the Metro transit data available to outside services such as Google Maps, which provides transit data for places throughout the world. In other major metropolitan areas, smartphone apps for navigating public transit were de rigeur, and Cincinnati was visibly left behind on the global transit stage. To rectify this situation, the development of Cincinnati’s own transit app was needed. With on the go route calculation, capability to direct the user to his or her bus stop, streamlining features such as route and location reuse, and configurable help options, getThere is a modern answer to Cincinnati’s bus system access problem.

7.2 Recommendations

Since the beginning of this project, SORTA has made some advancements in their tool availability. In Winter quarter, SORTA implemented a GPS tracking system for their buses. In Spring quarter, SORTA provided their transit data to Google. These two steps have taken Cincinnati out of the last decade, and into the modern transit age. Over the course of the project this year, it has become clear that route calculation algorithms are best left to mathematicians. In that vein, the considered next step for getThere is to integrate with the steps forward that SORTA has taken. Google’s route calculation algorithm is tried, tested, and highly efficient. It still fails to deliver non-route calculating functionality, and does not supply users with a way to manage their route directions and locations externally. Integrating the functionality of the getThere app
with the route calculation capability of Google’s transit service would result in a more robust and useable product.

In terms of additional next steps, once the app has been vetted in terms of route calculation, the intention is to submit the app to the Android marketplace as a student project beta. This is intended to gain feedback on the level of real user experience. The testing done for Senior Design was helpful and on expectation for the current development level. Submitting to the marketplace would reach out and acquire a rich lode of user experience and feedback. The implementation of the support email in getThere would hopefully encourage users to provide in-use thoughts.

One recommendation based on this year’s Senior Design experience is to not be afraid to seek out free development environments and tools. GIMP, although a different interface than the more popular Adobe PhotoShop, is a powerful image editing tool. It is vetted among the online community, and has been in use long enough for there to be significant resources in terms of tools and tutorials. This last consideration is important when seeking out external libraries to use as well. The longer the tool has been in the market, and the more user base it has, the more an independent developer can rely upon the tool being stable and well documented. The open-source community is a remarkable phenomenon in the development world, and users tend to want to share information rather than hoarding. The Wikitude API is likely not the best tool for the job. It was readily and freely available, but the amount of information able to be accessed about implementing the tool was minimal. The API itself was unstable and easily interfered with by GPS interference. If the device could not access its own orientation, the API did not gracefully let the user know, it simply did not do anything. This, did, however, emphasize the need for multiple tools when it is clear that one could fail. The implementation of the Google
Maps bus stop locator was due, in part, to the problems encountered trying to consistently use the augmented reality API from Wikitude. It is worthwhile to note that the backup tool may end up becoming more useful than the original implementation. Augmented reality is an appealing, attractive technology. What a user might find more immediately useful, however, is a map of their immediate area and how to find their stop. This second string tool is likely to be the primary means by which a user finds the origin bus stop.

Throughout the course of this project, iterative development has shown itself to be an excellent tool for progress and revision. This process allowed for much of the self-defeating criticism to be silenced, as well as breaking the project into manageable pieces. Developing a system is massive and overwhelming, but developing a module is a problem that can be considered on its own. For students coming into Senior Design in future years, one of the most important parts of the process is to make sure that, colloquially, you can see both the forest and the trees.
8. References

Works Cited


9. Appendices

9.1 Appendix A: Help articles

Using the App

- Saving a Passenger Route
  
  o Saving a Passenger Route can be helpful if you need to use the same bus route on a regular schedule, at the same time. This allows you to generate a passenger route, then save the directions directly to your device. You can then view these directions later, without being connected to the Internet!

  To use this feature, first generate a Passenger Route as normal. On the Directions screen, click the button that says “Save my Passenger Route”. You will then be directed to a saving screen. On this screen you will be asked to provide a name and some keywords for this Passenger Route.

  Give the Passenger Route a unique name. This is the name that will be associated with the route when you open it later. “Home from Jazz class” or “Evening Rehearsal” might be helpful names.

  Keywords are used to help you search for a route if you have many of them saved.

- Saving a Location

  o Saving a Location can speed up use of getThere significantly. Instead of typing in your addresses, just select from a list of Locations with user-friendly names! Setting up these Locations is simple and lets you plan a route without an address book!

  There are two ways to save a Location in getThere. The first way is to do so
via the Locations screen. From the main menu, click the button that says “Locations”. This will open the locations screen. On this screen you will see a list of Locations, if you have some already saved, and a couple of buttons. Click the button that says “Add new Location”. This will open the New Location screen.

Enter a unique Name for the Location. “School”, “Home”, “Work” are all useful names. Then, enter the street address for this location. The best format for street address is as follows:

1234 StreetName St, Cincinnati OH

Subbing in the actual street number for 1234 and the actual street name for StreetName. Once the information has been entered, click Save. If you receive an error message, that likely means the name is already in use. You should either choose another name, or edit the Location using that name.

The second way to save a Location is through the New Route screen. Generate a Route as you normally would, but make sure to check the box that says “Save this Location” before getting the route. You will then be prompted for a name for the location, and then brought to the directions screen as normal.

- Using a Location
  
  - Locations are one of the ways getThere makes life a little faster and a little easier. Locations let you use a pre-defined address with a simple, user-friendly name when generating a route, so that you don’t have to remember addresses or type them in when you need to get going ASAP!
To use a Location, go to the New Route screen. Next to the address bars labeled “From:” and “To:” there is an option to “Use a Saved Location”. Select this option. You will then be prompted to select a Location name from a list of names.

Simply select your Location(s), then click get There!

- Viewing a saved Passenger Route
  
  Passenger Routes are the sets of directions that you generate when you get a new Route. These directions are between two specific locations, at a particular time, on a particular day of the week. With getThere you can save your Passenger Route directions onto your mobile device, taking them with you to review and use when you don’t have an internet connection.

To view a saved Passenger Route, select “Saved Passenger Routes” from the main menu. This will open the Saved Passenger Route screen. This screen contains a list of all of the saved Passenger Routes, identified by the name you gave it upon saving, as well as a search bar. If you have many Passenger Routes saved, searching by keyword can save time, rather than scrolling through the whole list.

If you know the name of the route you are looking for, scroll through the list, and select the appropriate name. This will open the Directions screen for that route. If you do not know the name of the route you are looking for, type a relevant keyword into the search bar and click Search. This will open a results screen which will contain only Passenger Routes given the keyword you specified.
- Finding your Stop

  o Find My Stop is a getThere feature designed to help you find the right bus stop. This is helpful if you are unfamiliar with the area. To use this feature, use the New Route tool as normal, then on the Directions screen click the “Find My Bus Stop” button. This will open up your phone’s video feed, with a marker over-laid pointing you in the correct direction to find your bus stop. This marker will always indicate the direction of your bus stop as you move toward it.

    To leave the Find My Stop screen, use the back button or the menu on your mobile device.

- Configuring Options

  o Configuring Help options is one of the many ways getThere is adaptable to your needs. This lets you determine when you are prompted for information, and whether you will be asked to save automatically, or when you choose. To configure Help options, select Options from the main menu. This will open the Options configuration screen. The first set of options is the Help options configuration. There are a few settings on this screen, these settings correspond to groups of setting throughout the app. Turning on all app assistance will set all options to selected. Give me a brief overview will open a description of each page as you get to it. Help with input will turn on help icons that display input information on forms. Prompt me to save locations tells the app to prompt you when you use a manually entered address, to see if you want to save it as a Location. Always use my Current Location to begin a
route tells the app to always set your starting location as the From address on the New Route screen.

Riding the Bus

- Typical Ride

  o Typically, you will buy your ticket when you board your first bus. Fares are exact change required, no change made. It is helpful to have your money ready before you board. You pay the fare at the driver’s fare station. If you need a transfer ticket, ask for one when you board.

  It is important to pay attention to the stops as you go. There will be automatic announcements at each stop. When approaching your stop, pull the cord by the window. This will indicate to the driver that you want to be let off. This is important, because the driver will not automatically stop at every bus stop if there is no indication that a rider needs that stop.

  Once you have arrived at your stop, your trip is complete!

- Stops

  o Metro bus stops are well marked, and easy to recognize. They are frequently located next to a shelter structure, but are otherwise similar in appearance to parking signs. They are white, with the green and blue Metro logo. On the stop sign you will see at least one number. These numbers represent the bus routes that stop at this location.

  When you are directed to a bus stop, it is easiest to find them by intersection. Once you have located the appropriate intersection, look for a bus stop. If the bus stop displays the route you need, you’ve found the right place! getThere
can also assist you in this by displaying a Point of Interest marker on your phone for the correct stop.

Look for updates including Google map images of your stop!

- Fares
  o Fares are the money you pay to ride the bus. Determining the proper fare for a trip is pretty simple with Metro. getThere supports route calculation for routes within Zone 1. Travelling within Zone 1 has an associated fare of $1.85. This fare is for a simple, one-way trip on a single route.
  To include multiple routes in your trip, you will need to purchase a Transfer. A Transfer ticket is $0.50 and is good for up to two transfers, within a certain time period. When you purchase your ticket, ask for a Transfer ticket if you intend to travel on multiple routes in the same trip.

- Navigating a Transfer
  o Navigating a transfer may seem intimidating, but the process is the same as navigating a typical bus ride. When you purchased your ticket, you asked for a transfer ticket as well. This transfer pass is good for a specific amount of time, shown on the pass. Within that time period, when boarding your next hop bus, submit the transfer pass as you would any pass. This allows the system to know that your transfer is valid. Transfer passes are good for up to two transfers in the given time period.
  Keep your transfer pass with you until it expires or the number of transfers has been depleted.
9.2 Appendix B: Testing results

T1

1) Generate a new route
   a. Get a new route using saved locations Jen’s House and Bowling
   b. Get a new route typing in addresses
      Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
      Questions: Do I need to add the city? (to the address field)
      Comments: You should have the text box suggest the address format when
                 someone clicks into it
   2) Find your stop in two ways
      a. Open the Augmented Reality View
      b. Open the Google Map
      Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
      Questions: Is Find My Stop the Augmented Reality view?
      Comments: You should say that it is camera assistance
   3) Save the generated route
      a. Under the name “Test”
      b. Under any name
      Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
      Questions:
      Comments: The multiple name message is good
   4) View your saved route
      a. Open “Test”
      b. Open any saved route
      Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
      Questions:
      Comments:
   5) Create a new saved location
      a. Name “Test” Location “123 Test”
      b. Any name and location
      Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
      Questions:
      Comments:
   6) View Help Articles
      a. Find out how to calculate fares
      b. Find out how to review a saved route
      Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
      Questions:
      Comments: Maybe change the colors on the popups to match the rest of the
                 program
   7) Configure help options
      a. Turn on input field help
      b. Turn off input field help and turn on general screen help
      Mistakes: [X] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
      Questions:
Comments: I didn’t think describing the screen was general help
Comments: Change the option labels. The navigation menu was cool.

T2

1) Generate a new route
   a. Get a new route using saved locations Jen’s House and Bowling
   b. Get a new route typing in addresses
   Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
   Questions:
   Comments: I wasn’t sure the checkbox would show saved places

2) Find your stop in two ways
   a. Open the Augmented Reality View
   b. Open the Google Map
   Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
   Questions: Is the first one the augmented reality?
   Comments: You should put a help icon by the Find My Stop button to describe what happens when you click it

3) Save the generated route
   a. Under the name “Test”
   b. Under any name
   Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
   Questions:
   Comments:

4) View your saved route
   a. Open “Test”
   b. Open any saved route
   Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
   Questions:
   Comments: You should still let people find their stop from a saved route

5) Create a new saved location
   a. Name “Test” Location “123 Test”
   b. Any name and location
   Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
   Questions:
   Comments: Could you tell people the name has to be unique? (not a use question, classified as a comment)

6) View Help Articles
   a. Find out how to calculate fares
   b. Find out how to review a saved route
   Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
   Questions:
   Comments: These are cool

7) Configure help options
   a. Turn on input field help
   b. Turn off input field help and turn on general screen help
   Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Questions: Which one is general help?
Comments:
Comments: Need to get a cleaner logo. And change the option names.

T3
1) Generate a new route
   a. Get a new route using saved locations Jen’s House and Bowling
   b. Get a new route typing in addresses
      Mistakes: [X][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
      Questions: Do I need to put the city on the end?
      Comments: Tell people how the address should look
2) Find your stop in two ways
   a. Open the Augmented Reality View
   b. Open the Google Map
      Mistakes: [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
      Questions:
      Comments: I like that the icon looks like the rest of the app
3) Save the generated route
   a. Under the name “Test”
   b. Under any name
      Mistakes: [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
      Questions:
      Comments: I like that it opens the next page right away
4) View your saved route
   a. Open “Test”
   b. Open any saved route
      Mistakes: [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
      Questions:
      Comments: Why not show the other buttons? (posed as a comment, not a question)
5) Create a new saved location
   a. Name “Test” Location “123 Test”
   b. Any name and location
      Mistakes: [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
      Questions:
      Comments: I like that it matches saving directions
6) View Help Articles
   a. Find out how to calculate fares
   b. Find out how to review a saved route
      Mistakes: [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
      Questions:
      Comments: These should be in white
7) Configure help options
   a. Turn on input field help
   b. Turn off input field help and turn on general screen help
      Mistakes: [X] [X] [ ][ ][ ][ ][ ][ ][ ][ ][ ]
T4

1) Generate a new route
   a. Get a new route using saved locations Jen’s House and Bowling
   b. Get a new route typing in addresses
   Mistakes: [X] [X] [ ] [ ] [ ] [ ] [ ] [ ]
   Questions: Do I need the city?
   Comments: Didn’t know what address to put it

2) Find your stop in two ways
   a. Open the Augmented Reality View
   b. Open the Google Map
   Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
   Questions:
   Comments: The video thing is really interesting

3) Save the generated route
   a. Under the name “Test”
   b. Under any name
   Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
   Questions:
   Comments: Could you call this directions?

4) View your saved route
   a. Open “Test”
   b. Open any saved route
   Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
   Questions:
   Comments: You should let people cancel from this screen

5) Create a new saved location
   a. Name “Test” Location “123 Test”
   b. Any name and location
   Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
   Questions:
   Comments: Can you make sure the address is the right format here?

6) View Help Articles
   a. Find out how to calculate fares
   b. Find out how to review a saved route
   Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
   Questions:
   Comments: I like this section. New users will read through it.

7) Configure help options
   a. Turn on input field help
   b. Turn off input field help and turn on general screen help
   Mistakes: [X] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Questions:
Comments: These aren’t labeled clearly.
Comments: Follows normal driving direction sites, and I like not having to go back to the first screen to change options. Get rid of the first checkbox on the options.

T5

1) Generate a new route
   a. Get a new route using saved locations Jen’s House and Bowling
   b. Get a new route typing in addresses
      Mistakes: [X] [X] [X] [X] [ ] [ ] [ ] [ ] [ ] [ ]
      Questions: Can I use abbreviations?
      Comments: Couldn’t figure out how to get the address to work (not saved). Took a long time to say it wouldn’t work.

2) Find your stop in two ways
   a. Open the Augmented Reality View
   b. Open the Google Map
      Mistakes: [X] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
      Questions: Which button is AR?
      Comments: Didn’t know what AR was until I clicked. Should be clearer.

3) Save the generated route
   a. Under the name “Test”
   b. Under any name
      Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
      Questions:
      Comments: This works good

4) View your saved route
   a. Open “Test”
   b. Open any saved route
      Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ]
      Questions:
      Comments: Simple

5) Create a new saved location
   a. Name “Test” Location “123 Test”
   b. Any name and location
      Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
      Questions:
      Comments: I like how fast it tells you something is wrong

6) View Help Articles
   a. Find out how to calculate fares
   b. Find out how to review a saved route
      Mistakes: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
      Questions:
      Comments: These would help me with using the program. I should’ve read them first!

7) Configure help options
   a. Turn on input field help
b. Turn off input field help and turn on general screen help
Mistakes: [X] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
Questions: Where do I cancel?
Comments: These descriptions are confusing.
Comments: Couldn’t get the directions without help. Add more help icons.