CS 487 SENIOR DESIGN

Team 5 Final Report

Windows Phone 7 App for Get On Board

Matthew Lam, Chris Stevenson

5/2/2011

Final report on our progress for our project for Senior Design, a Windows Phone 7 application to interface with StarTran's Get On Board system, via a dedicated server.

Motivation behind the Project

Cell phones are everywhere today. These handheld devices have replaced traditional landline phones as the most important method to reach a person. Cell phones that are smartphones form a significant and rapidly growing portion of the market. The convenience and power of these devices is compelling—the ability to have information from the Internet available at a moment's notice anywhere in cell coverage. Smartphones bring fully-realized web browsing and apps to cell phone users.

The StarTran bus service in Lincoln, NE, recently—July, 2010—completed a web application to provide real-time bus location called Get on Board. The application provides the various bus routes, bus stops, and every active bus's location in Lincoln. In this form, the system has little more utility than basic route information. Furthermore, the application takes a significant amount of screen space—making it unusable on portable devices.

An application designed specifically for cell phones can present a useful subset of the Get on Board application, as well as provide some analysis of the information for a useful result: a better experience using the city's bus system. The cell phone app can use the real-time bus location information, combined with Window Phone Location service and Bing Maps direction service, to aid users in deciding whether to ride a bus and planning the bus routes needed to get where the users wish to go.

By developing this app, we are attempting to demonstrate the feasibility for municipalities to develop mobile applications for residents and visitors alike. By building off of existing available information, the city can provide information for services offered. During our research, we found no similar apps among the thousands in the app stores of iPhone, Android, or Blackberry smartphones. Virtually all of the current cell phone apps that deal with navigation are for vehicle turn-by-turn navigation, with walking being at most an afterthought.

This application will not be designed for people who drive to work. Rather, this is for people who walk and use public transportation.

Project Description

Our proposed project is an application for Windows Phone 7 that will assist users in using Lincoln's StarTran bus system. The application will interface with StarTran's Get On-Board System via a backend server to gather information about both the bus system itself and up-to-date bus locations. Using this information, along with Bing Maps, the phone's current location and a destination, the application will present the user with one or more bus routes that will bring the user close to the destination. Note that this application will only provide bus functionality while in the Lincoln, Nebraska, area. The application will provide the user with four different methods to choose a destination: typing an address, selecting from previous searches, saved places, or by browsing on a map.

Alongside the bus routes, where the user will get on the bus, the application will present the bus route identifier along with both the expected arrival time and the expected slack time. The slack time is found by taking the difference between the arrival time and the estimated walk time. If there is not enough time to catch the bus, the application will note that as well.

The rationale for looking for more than one bus route to reach the desired destination is that one of the routes may be a further walk, but will require less time spent waiting for the bus. Rather than arbitrarily deciding which route would be ideal for the user, the application will present the relevant information to the user to enable an informed decision of which bus to take. This functionality is being implemented by the dedicated server.

Below is a table of features we have identified and the current status of each component.

Feature	Status
Search box	Complete
History—List of recent places	Complete
Bus route list	Complete
Settings page	Complete
Display Traversal	Data complete, UI Incomplete
Display Map	Complete
Display Routes	Complete
Display Directions	Data Complete, UI Incomplete
Places list—favorites, places of interest	Future work
Tombstoning	Future work

Use Cases

In order to determine behavioral requirements, it is necessary to create use cases that cover all scenarios. Below are use cases we have determined are relevant to creating a complete product.

Use case #1: User selects a destination

Scenario A:

1. User inputs destination
2. Application displays map with best bus routes to get to destination
3. User selects desired bus route
4. Application displays information and directions to bus stop
5. When user is on bus, application shows when and where to disembark from bus
6. Application provides directions from bus stop to destination

Exception:

2. If user selects a destination that is closer to walk than take a bus, application will display a message to inform the user that it will take less time to walk to destination

Alternative Scenario A1:

1a. User selects destination from favorites menu

Alternative Scenario A2:

1a. User selects destination from points of interest menu

Narrative: The use case described is the happy path. Ideally, this is the most common scenario that can occur. There is an exception we plan on catching and that is when the algorithm is computing the closest bus route to the destination. If the user can reach the destination by walking instead of taking a bus, the application will inform the user that it is more convenient to walk. We will use the “walking” option when computing the route in order to determine if the walking time will be less than the time waiting for a bus.

Use case #2: User does not reach bus stop before the bus departs

Preconditions: The user is en route to a bus stop, but will not reach it in time or the bus has already departed

Scenario A:

1. Application reruns algorithm and displays closest 3 routes
2. User selects desired bus route
3. Application displays information and directions to bus stop
4. When user is on bus, application shows when and where to disembark from bus
5. Application provides directions from bus stop to destination

Narrative: It is possible for the user to run into obstacles that prevent him/her from reaching the bus stop in time. At the bus departure time, the application will check if the user is on the bus. If the user was unable to reach the bus in time, then the algorithm will be rerun and a new route will be selected. It is possible that the route will stay the same and the user will have to wait for the next bus.

Use case #3: User selects a destination while on a bus

Preconditions: The user is currently on a bus for a selected route

Scenario A:

1. Application will run algorithm using the next bus stop as the current location
2. Application displays map with best bus routes to get to destination
3. User selects desired bus route
4. Application displays information to disembark from current bus
5. Application displays information and directions to correct bus stop
6. When user is on bus, application shows when and where to disembark from bus
7. Application provides directions from bus stop to destination

Alternative Scenario A1:

4a. User can stay on current bus.

5a. Omitted

Alternative Scenario A2:

5a. User can stay at current bus stop and wait for another bus

Narrative: An edge case that could occur is when a user decides to change the destination while already on a bus on the way to another destination or the user opens the application while on a bus. What we do not want to happen is to run the algorithm as if it is a normal use case. The application will use the next bus stop as the current location, and information and directions will be displayed accordingly. There is a chance that the user is already on the correct bus, and will not have to disembark at the next bus stop.

Note: We currently do not have a deterministic method for when Use Case #3 would apply. This use case will not be explored at this time. We will consider this for future exploration.

Technology and Architecture

Technology Discussion

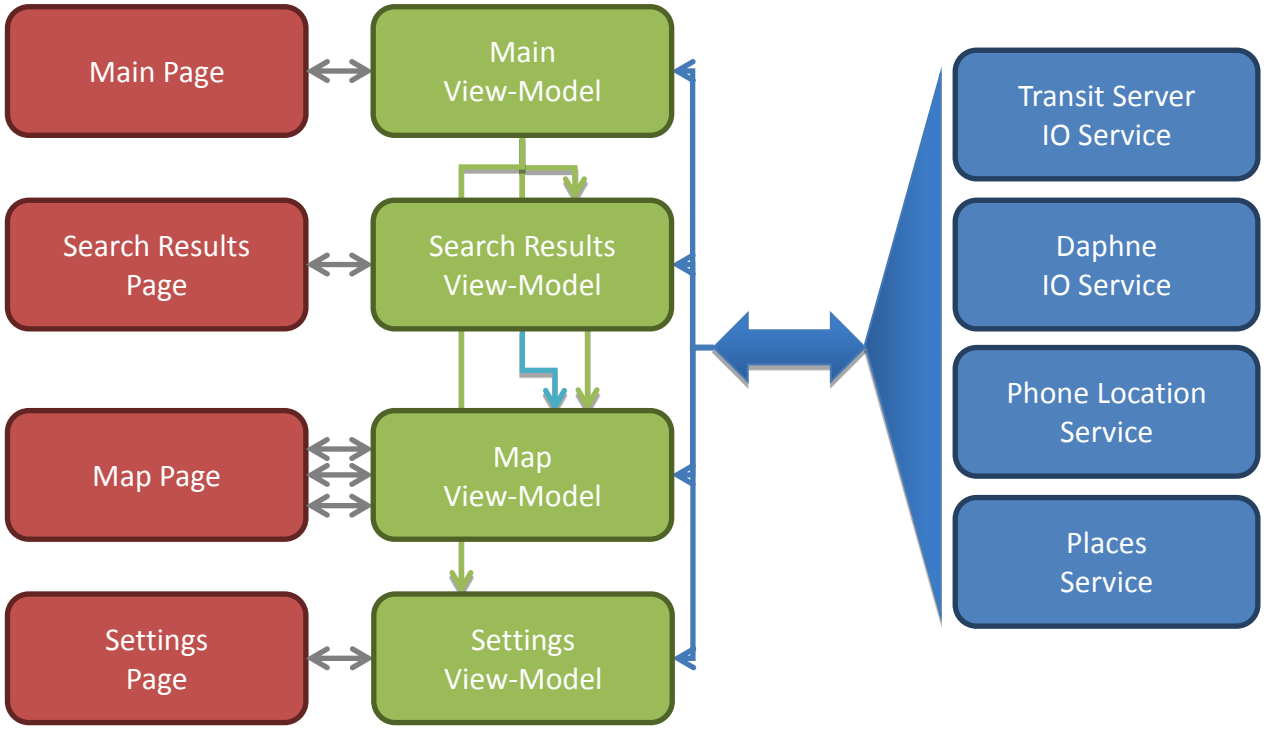
This project is being developed for the Windows Phone 7 using the Silverlight toolkit to implement the application, Bing Maps for mapping services and a server backend for information and route calculations.

Application Structure

This application consists of several services: the communication to the backend server, information requests to Bing Services, and the phone's location. The phone's location is already abstracted as a service by the Silverlight toolkit. The communication to the server and the request to Bing Services will be implemented as application level services.

We are following the Model View View-Model design pattern to decouple these services from the user interface using Galasoft's MVVM Light Toolkit. Each page of the user interface has a View, which defines the user interface elements, and a View-Model, which calls upon the services within the app and phone to provide properties for data binding and to pass messages to the View for when data binding is not feasible. This also allows us to easily change out interface elements as well as providing convenient places to unit test the functionality. We are using the Silverlight toolkit to build the user interface following the Metro UI guidelines.

As currently implemented, the View-Models are statically generated, and information is added to the as it becomes available. This design choice turned out to have drawbacks when used with a navigation based application with limited resources, as view-models that did not have a currently active page are still doing work, as well as difficulty in ensuring that resources are only used while they are needed and released when not.



General App Structure

User-Interface Structure

The main page is a pivot that will allow the user to input a destination. The search handles either a street address or a more general search term. Underneath the search textbox will be a list of recent destinations. There will also be a list of places of interest the user can view. Currently, we will be creating an initial list that will be seeded with the application. The user will be able to manage favorite

locations as well. The last pivot element is a list of all the available bus routes. Part of each bus route element is a status indicator that will inform the user if the bus route data is being downloaded and another indicator for when the route is currently running.

Once the user has selected a route or destination, they will see the map page. If the user has selected a bus route, the map will display the entire route. The user will be able to zoom in and out and view the current location of the bus as well as bus stop information. If the user has selected or searched for a destination, the map will display the walking route to the appropriate bus stop. Walking directions in text format will also be available to the user.

Testing

The application will use the Silverlight Unit Test Framework for windows phone. While this is not part of the toolkit, the portions that are available provide necessary functionality for unit testing. We are unit testing as we develop to assess our design decisions as we implement, as well as validating our code. First, the services will be validated, and then the View-Models supporting the user interface. There are currently 21 automated tests, and all of them are passing.

Unit Testing Procedure

While developing the PlacesService component, tests were written for every method. A black box testing approach was used for methods that would interact with other classes, and a white box testing approach was used for internal methods. As an example, the method GetHistory(), will be called to retrieve a List<Place> and display it to the user. When the method is tested, we will add a few test places into the history list and get the list. We expect it to return a list that is sorted by the time the place was last used. We can assert the correct order of the list for the test to pass. An example of a white box test is adding places to the places list. We know that the list should not add redundant places, so we can test this by added the same place twice. We will assert that the list size is the same. The result of this procedure is that every method in the PlacesService has been tested, and all of the tests are passing. By writing tests as we add functionality, we will be able to have more confidence that the code we have written will behave in an expected manner.

LINQPad

Another tool we used for testing is a stand-alone application called LINQPad. This tool allowed us to write and test the code to parse XML data outside of the application, create the classes to represent the data needed, and easily examine the data parsed. This proved extremely valuable in decoupling the services of the application, as well as adapting to the changes to the data sources throughout the semester.

Risk Assessment

Mitigated Risks

One of the most prominent risks we will encounter is the overcoming the learning curve for the technologies needed to complete this project. The Windows Phone 7 is a new platform and uses XAML, Expression Blend, and Silverlight, which are all new to all the members of the team. The severity level of this risk is very high because we will be unable to complete the project if we are not comfortable with

these languages, and there is a moderate probability that this risk could be realized. We have averted this risk because we have read available documentation and completed tutorials that are applicable to the functionality of the project. A digital resource that has been used is *Programming Windows Phone 7* by Charles Petzold. Using these resources, we have been able to complete a majority of the desired functionality, and we feel comfortable with the architecture and concepts we have implemented.

Another high-severity risk we will need to monitor is the dependency on the server team for service and functionality. We will need to provide specific details on what features the server side must provide in order for our application to work correctly. We must also be aware of any assumptions either team may have and reconcile any differences. Our mitigation plan includes maintaining frequent communication and refining requirements to ensure that appropriate functionality will be available for us to utilize. We will also report bugs and down time by utilizing JIRA and sending email to the server team. This risk has been realized a few times this semester when the service has been down. We have reported bugs on JIRA and communicated errors to the server team. The longest period of down time we experienced has been 72 hours. The server team has also completed an acceptable feature set that was necessary to the success of our application.

Outstanding Risks

A significant risk in this project is implementing the main features. Performance tuning took an unexpected precedence twice, first when the application was slow in emulation, and again when we obtained a phone for testing. This pushed back developing the navigation functionality. To control this risk, the main effort will be finishing the implementation and testing of the traversal navigation functionality. After this is functional, we will address other needs.

Another risk is failing to meet Microsoft's Windows Phone 7 Certification requirements. The requirements that are of the most concern are the encryption of data between the phone and the server, implements a look and feel consistent with the Windows Phone 7 interface, properly handling all exceptions, and providing privacy options for the user. Since any one of these would cause the application to fail Certification, this has a high probability of occurring. Our mitigation plan is to follow a spiral development plan, taking care to anticipate exceptions, and developing a reduced functionality version if the user wishes to not have the phone send out its current location.

Unexpected Risks

An unexpected risk in this project is the performance of the application. Even after two weeks focusing on the issue, it is not completely handled. This is most evident in the map elements, which uses most of the phone's processing power. This risk was realized when we were able to deploy the application onto an actual device. Previously, we have been manually testing using an emulator for Visual Studio. We have experienced significant system slow down when viewing the map page. We are controlling this at the moment by moving as much of the work as possible into background threads. We have plans to extend this by creating a single work queue that the work intensive portions of the application can use. Additionally, we can parse the data retrieved from the server and store it in a binary format; however this is only feasible if the work can first be queued.

Conclusion

The overall development of this project was hampered by an insufficient initial design. The initial design was based upon the use cases for the application. The necessary intercommunication for the different portions of the user interface was not well understood during the design. Also, the initial design did not fully take into account the platform requirements for developing for windows phone.

We learnt about the Windows Phone platform, and developing in Silverlight during the course of the semester. The services that collected data from the Transit Server, the server teams Daphne server, and from the phones built-in Location Service were successfully kept separate from the main logic of the app and the user interface, but this was in itself enough.

As our understanding grew, we began using the Model View View-Model design pattern, incorporating a View-Model for each page, which interacted with the Services, prepared the data for the UI, and kept track of the application state, from the users prospective. This ended up being too naïve of a design. Also, using the same view—the map page—for two different uses—displaying bus routes and displaying a traversal from one location to another—proved problematic. Even though these two shared a lot of functionality, the subtleties of this design were difficult to manage.

In short, learning about a platform as you develop for it will incur a significant refactoring cost, as some design decisions work out well, and others do not.

Future Work

Revising the Map Page into two separate pages with shared behavior, finishing the UI design for displaying traversals and managing places, and implementing Tombstoning are the most pressing additions that need to be made for this to be a completed app. Switching to view-models that are created and destroyed as pages are added and removed from the navigation stack needs to be explored to determine if this is a better approach or not. If not, then a more consistent allocation and releasing of resources needs to be done.

Also, developing a Silverlight Web application using the same services is another way to expand on this project. Windows Presentation Foundation could also be used for a desktop application using the same backend services as well.