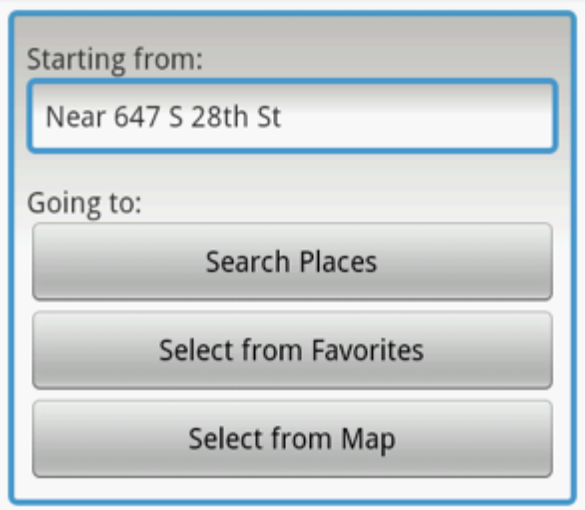
Final Progress Report 3
CSCE 487

John Miller
Josh Branchaud
Evan Chappelear

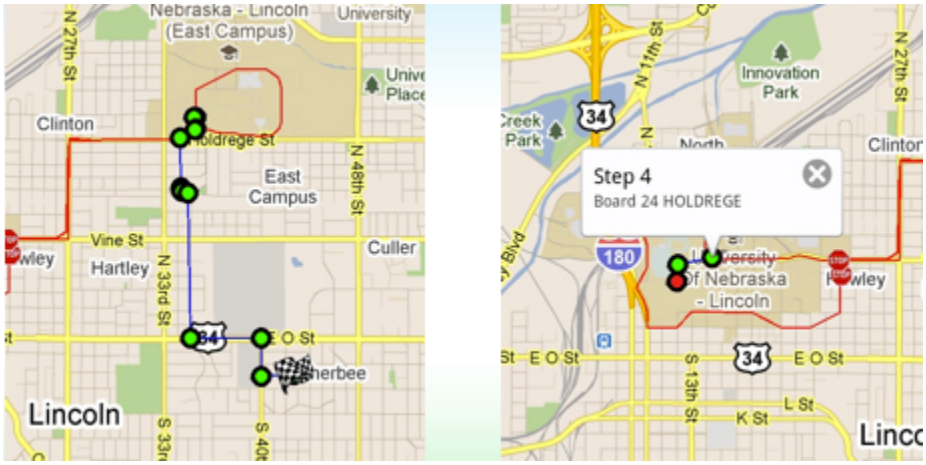
What

This section will detail the features that have been implemented up to this point. We consider this project a success because we were able to implement nearly everything that we set out to include. Though the end product has evolved since its inception, it has stayed true to what it has always been intended for.

The main feature that brought about the idea of this application in the first place was a route suggestion feature. Before this application was built, a person could get on their computer and pull up the 'Get On Board' application in their browser to get an idea of where the buses are and what the routes look like. This was clearly an improvement on simple static text schedules. However, with the prevalence of smart phones much more can be achieved than this. The first screen that appears when launching BusLinc is the 'Where To?' activity. It immediately begins using your phone's GPS capability to narrow down your coordinates. These coordinates are then sent through a Yahoo! web service which reverse geocodes them. The result of reverse geocoding coordinates is an approximate address of the users current location. This gives the user a sense of the application working and being accurate. Once the application has acquired the phone's GPS location, it will then allow the user to proceed with the route suggestion feature. At this point the user has three options:



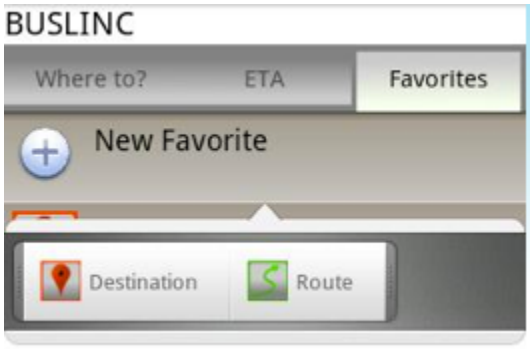
The user can search Yahoo! places with a simple keyword search. They can select destinations that already exist in their favorites. They can select a destination from Google maps by dropping a pin somewhere in the vicinity of Lincoln. By taking the route of any of these options, the user will be able to select a place to go and then BusLinc will immediately query the server to get the optimal route from their current location to the selected destination. This route is then loaded and displayed on the map view. The walking legs of the route are displayed in blue and the bus riding leg of the route is in red. Along the route are a series of green dots which are changes in direction. By selecting any one of these green dots, the user can read the directions for this portion of the route.



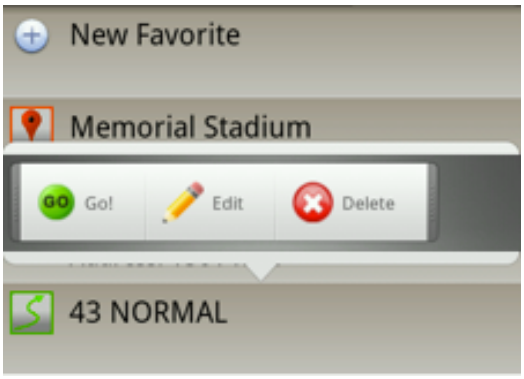
In addition to the information relevant to the route, the buses on the route and the official

bus stops on the route are displayed.

Over the course of the past two semesters we spent a lot of time researching for this project by scouring the Android Market for all sorts of related apps. Of the many apps that we found and reviewed, we noticed a feature that showed up a majority of the time. This was some sort of favorites feature. It was clear that we needed to include a feature like this in our app. It would make the app quicker and easier to use and would meet the expectations of users as set by popular apps already in the Market. We were able to successfully implement this feature. The favorites list comes preloaded with two favorites; the Lincoln Capital Building and Memorial Stadium. We figured this would be a fun way to personalize the app and give users an idea of how the favorites list works. By navigating to the favorites tab, the user can choose to create a new favorite. This can either be a favorite destination (location or bus stop) or a favorite route.



Once a favorites list contains a few items, the user may wish to interact with these in one way or another. By tapping a particular favorite, the user is given three options. They can ‘Go!’ to that favorite (or in the case of a favorite route, view that route), they can edit the information for that favorite, or they can delete that favorite if they no longer wish for it to exist in their favorites list.



The third main feature which we sought to include in BusLinc is the ETA feature. The infrastructure for this feature is in place, but it is dependent on a server call which is not quite ready for release. As soon as the server has the ETA functionality ready to go, we should be able to plug it straight into the app. We are excited to include this feature in a future release because we believe it to be the most common use case. For any person that use this app with regularity, we expect that the ETA feature will become there best friend. Each day that they seek to use the app, they will already know the route they want to take, but will be curious about the time until the bus arrive at the particular stop. That is the power of this feature.

This sums up the main features of our app to date. As can be seen, we achieved all of our specifications except for the ETA feature. We feel that this project has been a success and we look forward to the minor improvements that will take place over the coming weeks in preparation for releasing it to the Android Market.

How

The way the application was developed was by identifying what we would like the application to do, then research if it was possible and if so how. For instance, when we were working on the Favorites tab of our application the dialog boxes that popped up were unpleasant to look at and from looking at other applications we figured out there were much more elegant ways of displaying the favorite options. From there Josh research ways to implement similar features into our application and from there we discovered Quick Actions that display the same options as before but were less obtrusive and had animations that gave the

application a sleeker feel.

The main functionality completed since our last progress report is the human readable instructions queried from the server. Since we already had the functionality of displaying routes, all we had to do was incorporate this data on top of the instructions pulled from the server. The server returns instructions and waypoint data that will guide the user to their destination. First we had to filter out the readable instructions from the query and strip the html coding so the user could actual read where to go. Each step of the instructions was then numbered and associated with a GeoPoint where it should be displayed. With the step number, the instruction, and GeoPoint known, we then set all the data into a pinpoint on the map that had pop-up balloon containing the step number and instruction.

This information is useful, but also confusing without lines connecting them. Drawing a line directly from each step would look bad so we made use of the walking data the server provided to connect the user appropriately to each step. At first when we implemented this, we thought we had done something wrong, since the lines were still cutting through buildings and fields until we realized the data pulled from the server was off and not supplying the full walking directions. For now the walking routes are stuck this way until the server is able to provide a more robust way of providing accurate walking directions.

Quality

In order to ensure the quality of our application, we had assigned one member to be in charge of testing. We did not only need to test our own application; we also needed to test the server as well. Testing of the server was accomplished through shell scripts. The scripts made calls to the server using the wget utility in Unix and stored the output in files. The contents of these files were then inspected for their correctness. This may be a relatively simple way to do this, but it does have some advantages. First of all, this method makes it very easy to do other tasks on the files in the test script. Something that might be useful might be adding the address

into the result file or comparing the result with the result of an older test. It would also be trivial to write a program that automatically generates these test scripts, though we did not take advantage of that.

Testing of our application came in two main phases. The first phase was unit testing and the second phase was system testing. Since we did develop for the Android platform, Google has testing tools that we took advantage of. The unit testing was done with J-Unit. Being able to use J-Unit is one of the advantages of developing for the Android as opposed to the phones. System testing was used with tool called monkeyrunner. This gets a little bit confusing. There's monkeyrunner and then MonkeyRunner. Monkeyrunner is a shell script. It loads the testing script into Jython, which is basically Java with a Python syntax. MonkeyRunner is the API that the testing script uses. It allows the script to connect to and control either a running Android emulator or an Android phone that is connected to the computer.

Development and testing of our application unfortunately did not go as quickly as we had hoped, so we had to cut out some phases of testing. The first phase we cut out was for stress testing. We were planning on using a tool called Exerciser Monkey. This is similar to monkeyrunner, except it makes it makes pseudo-random inputs to the device. This phase was a regrettable loss, but not a huge loss. The loss that really hurt was field testing. It would have been very useful to get the application in the hands of people had not been working on it for months to get their feedback.

The "Where To?" tab will perhaps be the most used tab and has the most code associated with it. The "Search Places" button was something that was tested quite a bit. The first thing to test with it is if the button work. Different types of searches were tried. There was something general like "pizza", something specific like "Godfathers Pizza", there was something that had no results like "kwyjibo", and there was something with special characters that might be an SQL injection like "a'; DROP TABLE 'main';". The general search and the specific search

returned reasonable results, the nonexistent search had no results, and the SQL injection didn't break the server (which would be weird because the server doesn't use SQL). All of those are expected. With the two that had results, the first result was selected. The resulting routes were the correct routes. This was repeated after changing the emulator's GPS coordinate in Spain. All of these then showed the expected message that the phone needs to be in Lincoln for the application to work. Since all of the results were the expected results, it can be concluded that this button is working.

Risk Analysis and Mitigation

One of the issues that we have struggled with in the latter part of the semester has to do with our subversion repository. Last semester we had created a public subversion repository on Google Code. This seemed like a good approach because it did not require us to do any repository configuration or management. However, at the beginning of this semester we had uncertainties about the code being publicly available. The primary concern was that we did not know what the final direction of the app would be and so it seemed like it would be safer to have it hosted privately. The other concern was that the app was far from being in a presentable state and it seemed unfit for being displayed publicly. Because of these concerns, we determined that it would be best to host the repository privately until things had matured. We then setup a subversion repository on John's CSE account and began hosting the project there. It seemed fine at first, but somewhere down the line we ran into issues with being able to make commits. This was obviously a serious concern because it hindered our ability to work efficiently on the project while making recent changes available. We then tried recreating the repository on John's account and Josh's account, but kept running into the same issues. In that time we had determined that the app would be open sourced. Our difficulties with a self-hosted SVN repository and the decision to go OS made the idea of hosting on Google Code viable again.

We moved the current version of the app back onto our Google Code repository and have not had problems since. When analyzing risk at the beginning of the semester, this was not something we had anticipated. We took this type of tool for granted and just expected it to work. It was not until we were emailing zip files back and forth that we realized how important versioning systems are to software development.

Another issue that presented itself within the last few weeks was that of BusLinc's ability to respond to extreme/unexpected situations. By simulating scenarios such as being in a wifi blackout zone (using the phone in airplane mode), we were able to discover how the app responded to such situations. The app did not respond in a sophisticated manner and so we had to respond to this. Instead of elegantly handling a situation in which there was no 3G and no internet, the app simply crashed. This is a sure and certain way to lose users and get a handful of 1-star ratings. In order to mitigate this issue, we did our best to find all the trouble spots and put safety nets around them. Now, if the app runs into such a situation, it will display a message alerting the user that there is no internet connection rather than just crashing.

Project Plan

Despite the fact that the semester is over, the app will live on. Both John and Josh have agreed to stay involved with the project to certain degrees to help push the application along to ready it for release on the Android Market. Although the application is working, there is still plenty of work to be done to make the app worthy of use by our fellow Lincolnites.

The main thing we would like to implement before releasing is the ETA tab which has absolutely no functionality at the moment. This feature was not implemented since the actual estimations need to be done be the server and since the functionality has yet to be implemented on the server we cannot do much with it now. In the next few weeks when the ETA goes live on the server all we will need to do is add the appropriate server calls and the functionality

will be complete. In addition to the ETA, we would like to have notification to let the user know when their bus is about to arrive and when they need to exit the bus. The notification will be able to appear on the notification bar of Android as well as a message that will appear in the application. In the case where the user is not looking at their phone, we want to the phone to vibrate or ring to alert the user.

Another main functionality that we feel is necessary to the app is being able to save favorite destinations from the map by clicking on specified points on the map. The actual implementation will be done through the balloon pop-ups that appear when clicking on an object being displayed on the map. The actual balloon pop-ups that our app is using come from an open source library obtained from github and although the library is open source we feel it would be in the app's best interests to write our own balloon pop-up functionality.

The last vital thing we would like to see added to the application before releasing is professional images and icons. Currently all of our images and icons have either been thrown together by team members or pulled directly from Google which could possibly raise some problems when releasing. Kaylei from the server team has informed the teams she knows someone who could help us give our applications a sleek and uniform feel and we are looking forward to seeing what we can get drawn for us.

The final and most important thing that needs to get done before releasing is actually field testing the application. Although the application appears to be working, it would be quite the tragedy if we released the app without actually seeing if it works only to find out that it has stranded some little old lady across town. The actual testing of the application will not be an issue since both John and Josh own Android phones and we have had several friends and relatives volunteer to help test our app as well.

Conclusion

We all agree that the app has been a success. As software engineers we approached this project with over-optimism just as we would do with any other project. We had a series of lofty goals that we figured we could get done once May rolled around. Once we actually got to programming we realized how daunting these goals actually were and the project became a bit intimidating. However, we persevered, worked hard, and came out on the other end with the product we set out to create. Along the way we learned a lot. We learned specifically about Android platform development, but more generally we learned about a lot of mobile concepts and the issues that face mobile development. This has been a tremendous learning experience and we will more than likely use at least some of what we have learned either in our future careers or in our hobby programming projects. Furthermore, it was great getting to work with other teams of programmers. As I understand it, this experience was unique to our class because the teams usually have independent projects that do not involve interaction. Our class, on the other hand, got the benefit of working with and interacting with teams on related projects and experience of the benefits and struggles that come along with that.

"Never Forget"