**Bus Alert**

Robert Burrus

Alex Henry

David Pennington

CS 499, University of Kentucky, 5/6/ 2014

Disclaimer:

This project has been designed and implemented as a part of the requirements for CS-499 Senior Design Project for Spring 2014 semester.

While the authors make every effort to deliver a high quality product, we do not guarantee that our products are free from defects.

Our software is provided "as is," and you use the software at your own risk.

 We make no warranties as to performance, merchantability, fitness for a particular purpose, or any other warranties whether expressed or implied.

 No oral or written communication from or information provided by the authors or the University of Kentucky shall create a warranty.

 Under no circumstances shall the authors or the University of Kentucky be liable for direct, indirect, special, incidental, or consequential damages resulting from the use, misuse, or inability to use this software, even if the authors or the University of Kentucky have been advised of the possibility of such damages.

**Abstract**

Bus Alert is a mobile app designed for the bus riders of Lexington, KY. We want to give bus riders with iOS/Android smartphones the power to see the status of a bus stop on the go. We do this by hooking into the GPS data that is present on all Lextran buses in Lexington. We offer a dynamic map that shows the bus stops on a route and the estimated times of arrival for those stops. On top of that, we have implemented a simple notification system that bus riders can use to alert them with information for a stop at a specified time.

**Introduction**

Need: Currently, there is no good method for mobile users to check on the status of the bus while on the go. Lextran provides a good web interface from a computer, but mobile users (until late in the project, when a Lextran iOS app was released without warning) are out of luck. This results in missed busses and unnecessarily long wait times at a bus stop. Knowing the estimated time of arrival of a bus based on its GPS coordinates greatly increases the user-friendliness of the bus system.

Solution: We will create Android and iOS (abandoned after release of Lextran app) application that offers live information and tracking of Lextran bus routes. We will offer an interactive map that can filter by route, then allows the user to view estimated arrival times at a particular stop on that route. We also plan to implement some kind of notification system so that users can be alerted with the estimated arrival times at a specified point in their day. This will benefit the user in that they will know if a bus is delayed 20 minutes. They can know put that time to good use instead of being stuck at the bus stop for that time.

Our customer for this project is ultimately the citizens of Lexington, but our point of contact is Nick Such of Awesome Inc. The project itself is part of the Code For America Civic Initiative that focuses on civic improvement through open source development.

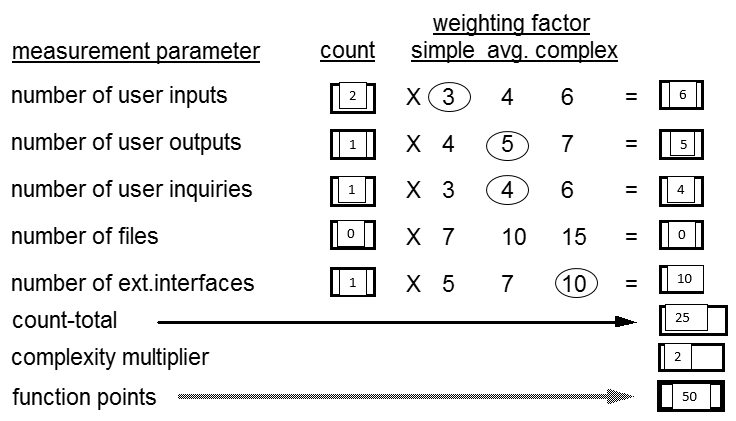
**Specifications**

* Our project will focus on the creation of a bus tracker similar to the one on http://lextran.com/bustracker that will work on mobile platforms.
* At the moment we are planning on creating a native iOS app as well as a native Android app that would allow the user to see the interactive bus map on their phone. Lextrans webapp is formatted such that the map is difficult to use on a mobile device. Our app will fill a void that will help students and Lexingtonians to not miss the bus.
* We want to implement a feature that allows the user to subscribe to a specific stop, at a specific time of the day, for a specific bus, and get notifications that tell when the estimated arrival time of the bus is.
* The success of our project will be judged on the functionality of viewing the real-time bus routes, the user friendliness of the app, and the functionality of the bus subscription feature.

**Product Planning**

**Effort and size estimation**

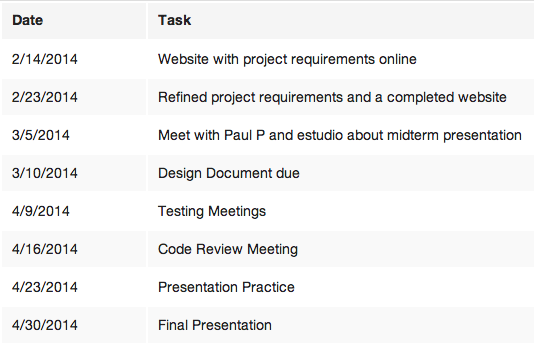
This chart is the original chart given to calculate the number of function points for our project.



50 function points for each application, approximately translates into (50\*35(for Java)) = 1750 lines of code. This was a rough approximation from our midterm report. The app now has 1206 lines of java.

Because there was no major experience with android in our group, it was hard for us to accurately count the number of lines that we could have had.

**Schedule and milestones**



**Platforms, tools, languages**

Our devices are built to work on the Android 4.4.2 (API 19).  Lextran’s API is used to overlay the bus route information and GPS data onto the map. Google Play Services is used to allow google maps to be integrated to the Android application. We used Eclipse with the android development plugins. We used the Google maps V2 API.

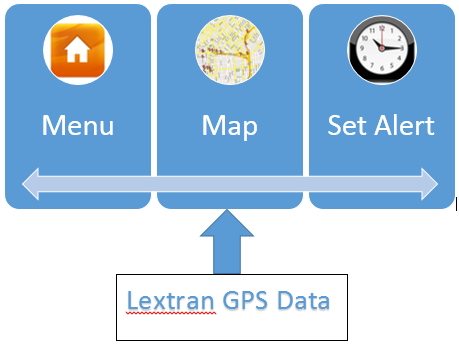
**Design**

**Environment**

The application will only be guaranteed on the Android environment 4.4.2 (API level 19). Older versions of Android could still use the app, but will not be tested for. We used Github for source control and hosting of our website. Jekyll was used for website creation. The Android app was developed through the Eclipse IDE using Java. Lextran’s API is used to overlay the bus route information and GPS data onto the map. Google Play Services is used to allow google maps to be integrated to the Android application.

**Module Descriptions and Data Flow**

The application is built with functionality and ease of use in mind. The only place you can go with the app is to a map overlaid with Lextran bus stop/route data, and a screen where you can set the time of alert. The difficult part of the project was getting updated Lextran GPS data and integrating that into our application.



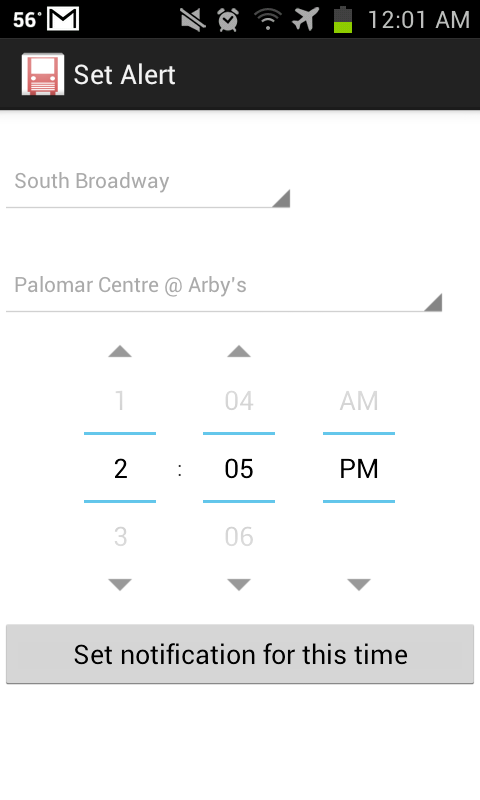
The menu screen offers two options: go to the map, or go set an alert. The intuitive nature of the application puts zero constraints on the ability to understand and use the app. We want to offer this to people regardless of technical background or even mobile app experience.

**User Screens**

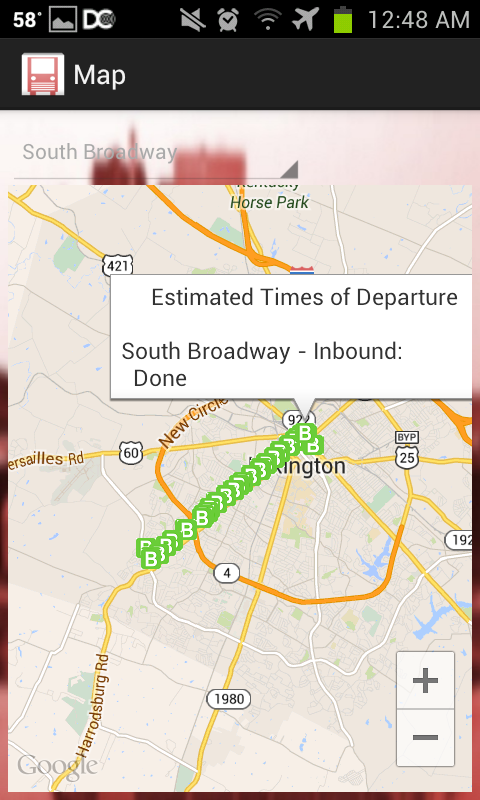
Home Screen:



Set Alert Screen: pick a route, stop and time. A notification will appear at that time telling the user how far away the buses for that stop are.



Map Screen: Pick a route and all bus stops will appear. Click on any stop and a text box appears telling the user how far away the buses are for that stop. In the example below, there are no more buses coming for the day– denoted as “Done”.



**User Scenarios**

Upon opening the app, the menu appears.

Menu: select between viewing the map and setting an alert.

Map: interactive map, overlaid with Lextran data showing the bus routes and bus stops of all city buses. The location of all buses is also shown, as seen on the old web app above.

Set Alert: The user selects the bus stop and time they want to receive a notification of the ETA of the incoming bus. Multiple alerts can be set, but none can be set to repeat (potential future enhancement).

**Algorithm**

Perhaps the most difficult functionality to follow in the code is how the notification is sent. Here is a brief explanation:

When the user clicks the button to set the alert, a schedule client object passes the calendar to setAlarmForNotification. This function then calls ScheduleService (from scheduleClient.java) to run an alarmtask. The alarmTask raises an alarm when the user specified time is reached. NotifyService is activated when the alarm is raised and it creates a notification with the ETA times by querying Lextran from within NotifyService. The notification is currently not able to be deleted or set to repeat.

**Implementation**

There were only a few notable hiccups during development. The first was the unordered nature of Lextran’s API and use of differing KML files. This made parsing the files for coordinates more difficult than expected. It also meant that we couldn’t just pass the KML files to Google Maps but had to manually extract the information and draw the bus stops on the map ourselves.

A second issue that came up was that Lextran released an iOS app with essentially the same functionality as our app. We had originally planned to do both Android and iOS but dropped the iOS app when Lextran announced that it was releasing one of its own.

Limitations of the Android application are on the ability to set notifications. A user is able to set multiple notifications at one time but will not be able to view, delete, or tell them to repeat.

**Testing**

The Bus Alert system is designed for ease of use and simple functionality. Consequently the breadth of different test cases isn’t substantial. User input consists of selecting a route, bus stop, and notification time from lists of available options. There is no custom input the user can feed the application and so we do not need to consider any filtering or error handling of incorrect input.

Unit testing was our main focus. The application has 2 different views available to users, each with their own specific functionality. Each of these will be tested individually as they are completed.

1. Map Screen - This view appears upon opening the application. This is the google map that displays the route information specified. The user can select a route from a drop down box of available LexTran routes and see the relevant bus stops. If the user clicks on a bus stop, information pops up telling the user when the next bus will arrive at that stop.
2. Notification - This view allows the customer to select a bus route, bus stop, and specific time that tells the application to send the user an alert at that time telling the user when the next bus is going to arrive at that stop. Multiple notifications are allowed.

Map Testing:

|  |  |  |
| --- | --- | --- |
| Route | Stop | Result |
| South Broadway | Harrodsburg & Man O’ War | Success |
| North Broadway | E Main & N Broadway | Success |
| Trolley Blue Route | 4th & N Broadway | Success |
| Newtown Pike | Jefferson & 3rd | Success |
| Versailles Road | N Broadway & Main | Success |

Alert Testing:

|  |  |  |  |
| --- | --- | --- | --- |
| Route | Stop | Time | Result |
| South Broadway | Red Mile OB | 4:15 PM | Success |
| BCTC Southland | Regency | 4:16 PM | Success |
| Woodhill | Transit Center | 4:19 PM | Success |
| Hamburg Pavilion | Bryant@Plaudit | 4:21 PM | Success |
| Nicholasville Road | Lexington Green | 4:22 PM | Success |

\*Success designates that times were displayed or that the notification was received

**Future Enhancements/Maintenance**

**Future Enhancements**

* Recurring notification system. A system that allows for a large number of recurring alerts to be created at specific times on specific days of the week. A user could be alerted of the bus times 10 minutes before he usually leaves in the morning, every morning. This would require a screen to create, edit, and remove existing notifications.
* Streamlined Android Layout. None of us were great with UI. Maybe a layout that starts on the map and then uses menu bars to navigate to the rest of the app.
* Text alert system. Implement a webapp for non-smartphone users that would allow them to set an alert and receive text message updates to their non-smart phone.
* “You Are Here” feature of map to communicate to the user where they currently are.

**User Raised Issues During Testing**

* App does not tell the user where they currently are on the map. User noted that it would be a useful feature for those not familiar with the geography of Lexington.
* Filtering routes in the interactive map did not remove the stops from the previous route selected. As you selected different routes, the stops from the old routes would remain on the map.
* Upon opening the map, exiting, then trying to reopen the map, the application would close unexpectedly.

**Project Features**

* Interactive map that is reached from the menu by clicking on the “Map” button. Map dynamically queries Lextran’s API and allows the user to select the route (hardcoded) they wish to look at, then dynamically populates the stops that are on the route. When a user clicks on a stop, the next estimated times of bus arrivals are dynamically retrieved and then displayed.
* The notification settings page. Has a similar set up to the map with pull-down boxes to select route and stop. Then has a time selecter and a button to produce a notification at this time. The notification includes the same info that would be displayed if the user had clicked on the stop from the map.

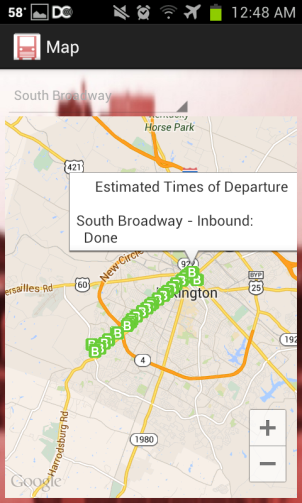
**Conclusion**

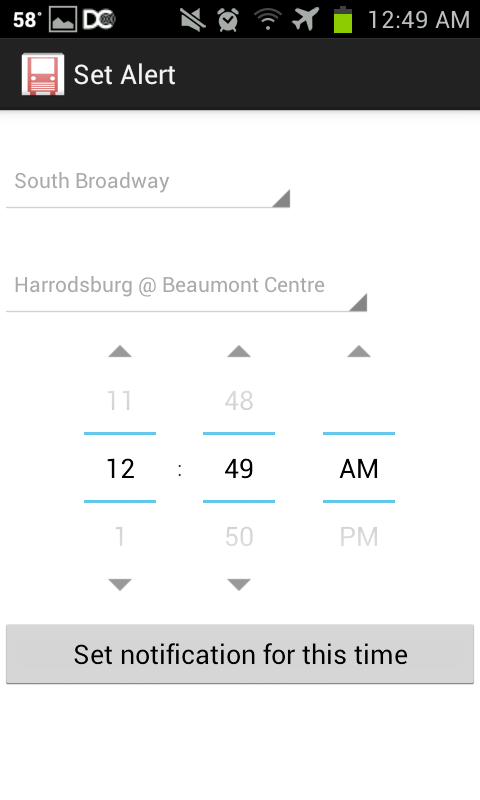
From this project we learned the basics of mobile app design in both Android and iOS. Even though the iOS app was discontinued because of Lextran’s application release, we were still able to gain the basic knowledge needed to develop an app in iOS. We also learned about the difficulties of UI, and how difficult it can be to make a simple interface for the user of an application. If we had known ahead of time that Lextran was working on an iOS application, we would have focused all of our efforts on the Android app’s development.

Nick Such is pleased with the outcome of the app. We have been in contact throughout the development of the application. He understands the development decisions we have made throughout, and is happy with their results.

**User Manual**

BusAlert is a very straightforward app, but I will explain how to use the functionality it has. The main menu of the app is what will launch the first time that you start it after installing.

The “Map” button will take you to our interactive map. Once there, you will have a drop down box that allows you to select the route that you want to view. Once you select a route, the map will query Lextran and draw the stops on that route onto the interactive map. At this point, you can click on any of the stop icons to see the estimated arrival of buses coming to the stop.

You can now continue selecting routes, or use the back button on your phone in order to go back to the main menu. If you click on the “Set Alert” button you will see the notification creation screen.

You now have another drop down for route. Once you select a route, the drop down box with populate with the stops on that route. You now select one of those stops and a time. After selecting all of those options and pressing the notification button you will be alerted that a notification has been set. Bit your back button to close the notification alert.

That covers the main features of our application. If you select the “About” button on the menu screen you will see a brief description of Bus Alert. If you select “Exit” it will close BusAlert. This concludes the guide of BusAlert!

**Installation Process**

In order to run the app you can download the APK from the app store with the name "BusAlert".

If you want to work on the code, you will need to be using the Eclipse SDK and import the project files found in this repository. Install the Google Maps API v2 tools from google using the Android Developer Console. See the link here on instructions to do that: https://developers.google.com/maps/documentation/android/start

After that you can install an Android emulator using the Android Developer Console, or you can connect a phone and run the app from there. You will have to be careful with the release key, as it is tied to my account and project. It would be good practice to use the Eclipse debug key for debugging purposes and to create a new Android Release Key for release of the app. Here is a guide on how to publish apps: http://androidistic.com/1510/how-to-publish-apps-on-google-play-store/ .