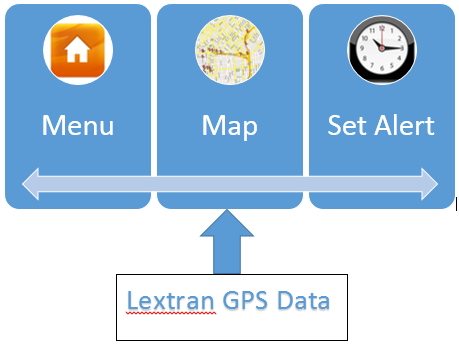
**Design**

**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.



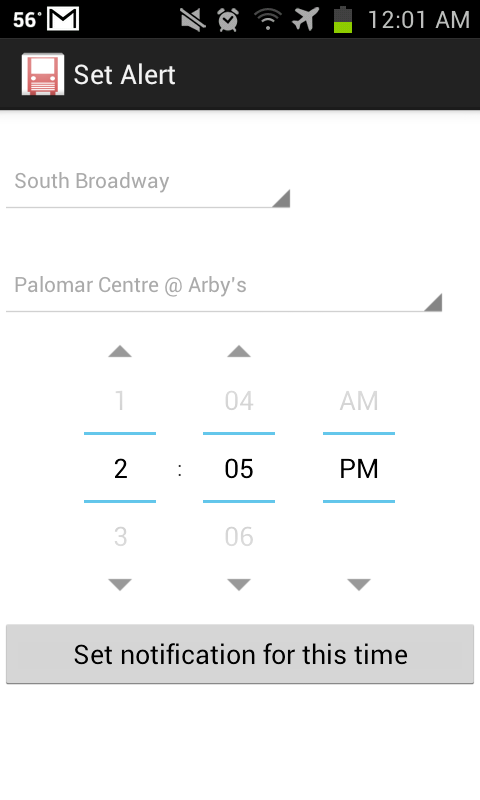
The menu screen offers two options: go to the map, or go set an alert. The intuitive nature of the application puts zero constraints on the ability to understand and use the app. We want to offer this to people regardless of technical background or even mobile app experience.

**User Screens**

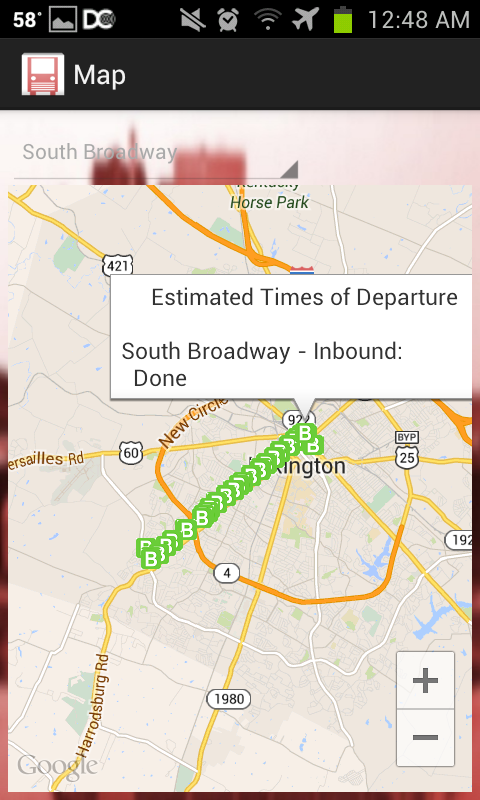
Home Screen:



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