Union College Trolley Tracking System

Grace Yang
CpE-499: Senior Capstone 2020
Advisor: Professor Cotter
6/10/2020

Table of Contents

List of Figures

List of Tables

1 INTRODUCTION

Union College trolley service was implemented to provide students with free transportation around the campus Sunday through Thursday 6 pm - 2 am; Friday and Saturday 6 pm - 4 am. This system not only makes it more convenient for students to get around campus, but it also ensures student safety by providing a safe alternative to walking alone around campus at night. However, in an interview with Union College's Transportation Manager Rich Hewlett, he said "we did a survey this past Spring regarding student usage of the trolley. Disappointingly, the result showed that there is only an average of zero to one student each day who takes the trolley." (R. Hewlett, personal communication, Oct 2, 2019). Only a small number of students use the trolley. That led me to thinking if something could be done to increase this number.

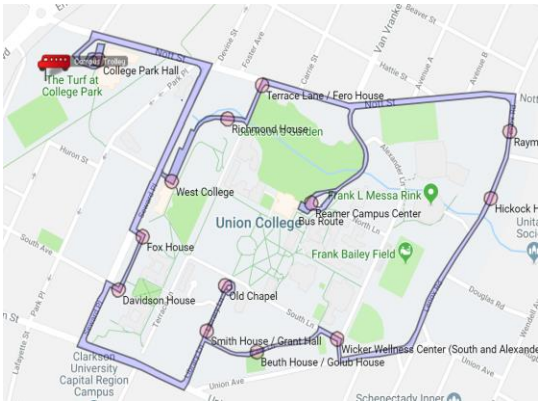


Figure 1: Union College Trolley Tracker

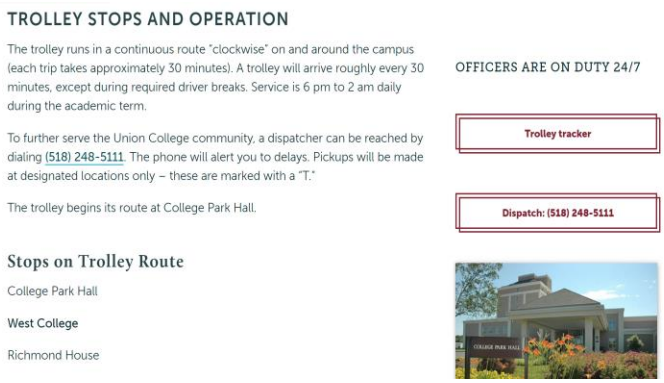


Figure 2: Trolley Stops and Operation Website [1]

I looked into the trolley services and noticed that the only way for students to track the trolley is through a trolley tracking map (Fig.1) provided on the Trolley Stops and Operation website (Fig.2) that displays the real-time location of the trolley. It does not specify the time it takes for the trolley to go from one location to another or the stopping time at each location. Therefore, although it is helpful to know the real-time location of the trolley, it is hard for the users to estimate the arrival time at the pick-up or drop-off locations. According to Union College's Transportation Manager Rich Hewlett, the trolley often takes detours to other locations other than the listed pick-up

locations, which makes it even harder for students who are waiting for the trolley to know whether the trolley will be arriving or not.

The current trolley tracker is supported by a commercial vendor who tracks the movements of the trolley via a GPS unit in the trolley and displays the current trolley location on a map. The problem with this is that the cost of service is $29 per month (R. Hewlett, personal communication, Oct 2, 2019), which is relatively high. An alternative GPS tracking system with reduced cost could be developed.

My project addresses these problems by redesigning the Trolley Stops and Operation Website and creating a user-friendly mobile application. The features of the mobile application will include the real-time location of the trolley; the arrival and departure time of the trolley at each stop; and an interface that informs the driver of the number of users at each location.

2 BACKGROUND

2.1 GPS and Tracking

GPS, also known as the Global Positioning System, is a satellite-based navigation system consisting of three segments. The space segment consists of 28 satellites that orbit around the earth. The user segment consists of receivers that you can hold in your hand or mount in your car. The control segment consists of five ground stations located around the world that make sure the satellites are working properly.

These three segments work together to ensure accurate location identification. GPS satellites simultaneously transmit synchronized time and orbital data to Earth. The Command center

transmits orbital data, time corrections, and location of other satellites in the GPS constellation. GPS receivers compute location using orbital data and the difference in arrival times of the signals of different satellites.

2.2 University Transportation Mobile Apps

Some universities provide students with public transportations to ensure student safety, reduce on-campus traffic, and provide a cost-effective way of traveling between classes and dorms that is time optimized. For example, the University of Kentucky developed a mobile app called TransLoc Rider in 2011 that offers free iPhone and Android apps for real-time GPS tracking of on-campus buses as shown in fig.3 [2]. The app includes features such as the ability to select routes or individual stops, save user's favorite stops, provide feedback on user's experience, and get notified of an estimation of the arrival and departure time to and from each stop [3]. In 2015, 21,708 users used the app to track campus buses. Other universities such as Duke University, Binghamton University, and Yale also uses similar on-campus transportation apps.

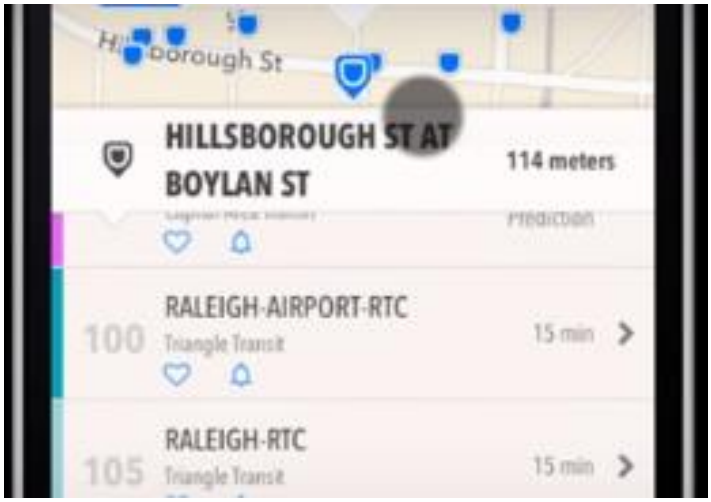


Figure 3: TransLoc Rider Mobile App[2]

2.3 Real-Time GPS Location Logging Methods

There are two parts to storing GPS data: obtaining and analyzing the real-time GPS data and importing the data to a mobile application and website. To obtain the real-time GPS data, we could either achieve it using a GSM/GPRS module (Fig.5) or using a mobile phone. Some conventional ways of using mobile phones to track and log GPS data are using Android apps to extract data, using the API of Google Location Sharing to retrieve the location history of the phone, and using Find My Device service on either an iPhone or Android. There are many existing research papers on implementing real-time GPS tracking via a mobile phone. Among them, the Chittagong University's bus tracker project stood out to me. The system monitors the buses and updates the changes in their status via an Android application. The app in the vehicle receives the location from the satellite and transmits it to a software called Firebase. Firebase updates the data every few milliseconds and transmits it to the users. This method is proven to be cost-effective, easy to maintain, and provides faster location updates [4]. There are many free Android apps on the market that log the GPS data of the phone. For example, an app called GPS logger logs the phone's coordinates to GPS format files at regular intervals (Fig.6) [5]. To track the real-time location of the trolley, an Android phone with a GPS logging app running in the background could be mounted on the trolley. This method is effective as long as the phone is charged and has a stable service.

Figure 5: GDM/GPRS/GPS Module [6]

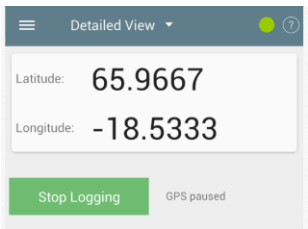


Figure 6: GPSlogger [5]

2.4 Mobile App Development

App development is developing for mobile devices such as smartphones, phablets, and tablets. Most people's mobile apps run on iOS and Android. As of August 2019, Android controls around 51.8% of the mobile device market and iOS owns 47.4% [7]. iOS is developed and supported by Apple whereas Android is developed and supported by Google. Android is an open-source operation system, which means it can be used by other device makers and it is easier to make alterations to the system. On the other hand, iOS can only be used by Apple and it is fairly restricted. Unfortunately, native iOS apps and Android apps cannot work together. They are built separately with different tools and source codes. This raises the question of how we could build an app that works for both iOS and Android? To answer this question, we need to understand the three main types of mobile apps: native app, hybrid app, and cross-platform app. The very first decision that an app developer will face is which type of app to build. There are benefits and disadvantages to these different types of mobile apps and there is no clear winner. When deciding about which type to choose, our choice depends on different factors: the budget, the size of the app, performance requirements, deadline, etc.. It is important to first have a thorough understanding of these different types of apps.

2.4.1 Native App

A native app is a software application built for a specific device platform using different programming languages and SDK. For example, Swift or Objective-C is used for iOS, and Java is used for Android. iOS uses XCode as its development tool whereas Android uses Android Studio. Therefore, a native app of iOS cannot be used on an Android device and vice versa. This introduces us to the disadvantages of developing a native app: We need different source code for iOS and

Android, it has a higher cost of development, it takes longer time to build, and we need to use specific SDK for the development process. Despite its disadvantages, it has obvious benefits as well: a native app has fast response time, wider device functionality, more robust, and matching user interface and user experience to platform conventions [8].

2.4.2 Hybrid Apps

An alternative to native apps is hybrid apps, which compensates the biggest disadvantage that native apps have: different source codes for different software applications. The codebase of a hybrid app is built with standard website development tools -- HTML, CSS, and JavaScript. The codebase is essentially a website embedded in WebView, an embeddable browser that a native application uses to render web content [9]. In short, a hybrid app is a simplified web browser embedded in a native app.

The obvious advantages of a hybrid app are: one codebase across all platform, budget and time-friendly, and has a sufficient amount of access to device features, time [10]. With advantages comes its disadvantages: it does not perform as well as native apps, it takes longer to load, it has limited capabilities due to the need to use plugins to access the built-in features of a device, and poor user experience[11]. Some examples of hybrid app platforms are PhoneGap/Cordova, Canvas, Ionic, and OnsenUI.

2.4.3 Cross-Platform Apps

Like hybrid apps, cross-platform apps also enable developers to create applications that work across platforms. Cross-platform apps' codebase is developed using an intermediate programming

language such as Javascript and Dart that connects to native components via bridges and libraries. This allows for a close-to-native user experience that hybrid apps cannot achieve while maintains hybrid apps' advantages such as allowing companies to develop time and cost-effective app that are stable and easy to maintain[12]. Some examples of cross-platform development platforms are Xamarin, React Native, Titanium, and Flutter.

After comparing these different types of apps, I decided to develop a cross-platform app due to the following reasons. The trolley mobile app needs to run on both iOS and Android devices because the main users are students on campus who use both iOS and Android devices. Considering I only had twenty weeks to complete the project, the development process for cross-platform or hybrid apps is more time-efficient compare to native apps. Since the trolley app had more complex features such as user interaction and real-time GPS tracking, I chose a cross-platform app over a hybrid app because it would deliver a better user experience that is close to that of a native app.

3 DESIGN REQUIREMENTS

Performance

1) The trolley tracking mobile app will be able to provide a real-time update of the trolley's current location within every second.
2) Efficiently notify the driver of the location of the students and the number of students waiting at each location.
3) Notify the students
    a. Arrival time of the trolley to a certain location
    b. The departure time from a certain location

c. When the trolley takes a detour off campus

4) Display daily trolley schedule
5) App startup time is less than 2 seconds

Cost

- The cost for purchasing software tools should be within $100
- The cost of purchasing an Android phone should be under $50
- The cost for publishing the app on the Apple App Store and Google Play Store should be around $125
- The app will be free for all users

Aesthetics

- The design should be creative and simple with smooth flow and pleasant color scheme

Maintainability

- The code for the app should be well documented, bugs are easy to fix
- The software is easy to understand, easy to target what needs to be changed, easy to make changes, and easy to check that the changes have not introduced additional bugs

4 DESIGN ALTERNATIVES

4.1 Real-Time GPS Tracking Module

As mentioned in the background section. There are two ways of logging real-time GPS data. One is to place an Android phone in the trolley and extract the GPS coordinates of the phone via an existing mobile application. The alternative method is to design a simple GPS tracking and data logging device. This could be accomplished by using a TinyDuino Processor Board paired with a GPS TinyShield. The GPS data could then be processed and monitored using Arduino IDE [13].

This design is easy to implement, and the components are low cost. However, this system might be too fragile for long-term use on a trolley. A more robust design could be implemented with a GM862-GPS cellular quad-band module and a microcontroller [14]. The GM862-GPS is a satellite-based service that combines GPS and GPRS to achieve a more accurate measurement of real-time tracking. A microcontroller used together with the GM862-GPS would be used to activate and analyze the GPS data [15]. Although implementing GPS/GPRS module is doable, it is less time-efficient compared to using an app on the Android phone. It also requires more work for hardware maintenance. Therefore, using an Android app to log real-time GPS data is a more desirable choice for my project.

4.2 Mobile application design platform

As mentioned in the background, there are three types of mobile apps: native apps, hybrid apps, and cross-platform apps. After thinking through the pros and cons of each type of app, I decided that developing a cross-platform app is most suitable for my project. There are two major software development kits for cross-platform apps: React Native and Flutter.

React Native is backed by Facebook and has been used by Instagram, Facebook, Airbnb, Walmart, Testa, UberEATS, etc. [16] Flutter is backed by Google and has been used by Alibaba, AppTree, Tencent, Google Ads, etc. [16] Both software development kits share some similarities despite their differences.

Both React Native and Flutter are backed by giant tech communities: Facebook and Google. Both are open source and free SDKs. Both have strong communities keeping documentation and comprehensive resources up-to-date. Although their performance cannot be compared with that of

native apps, they have their ways of utilizing fundamental UI building elements and widgets to provide nearly native experiences. Another feature that they both share that's worth discussing is the hot reload function. This function allows any changes made to the code be reflected in the visual emulator instantaneously [16].

Although they share many similarities, there are major differences between them as well. React Native uses React library and JavaScript, whereas Flutter uses a programming language called Dart, a language developed by Google that is object-oriented and modern. React Native has stronger and more mature community support compare to Flutter because Flutter is relatively new. However, Flutter offers well-documented resources and support backed by Google to compensate for this shortcoming. In terms of front-end support, React Native uses native components to achieve attractive UIs, whereas Flutter uses built-in widgets to achieve attractive UIs. Below is a table comparing these two development kits (Table 1).

After examining the pros and cons of each software development kit, I decided to use Flutter for my project instead of React Native for the following reasons: Elegant and object-oriented programming language, excellent and complete documentation and resources by Google, and near-native user experience.

	Flutter	React Native
Language	Dart	JavaScript
Developers	Google	Facebook
Initial Release	2017	2015
Native Performances	Superior	Good
Time-to-Market of App	Faster	Slower than Flutter
Documentation	Precise, clear, and up-to-date	Up-to-date and imprecise
Who Uses	Alibaba, Reflect, Google, Greentea, Tencent, Google Ads, App Tree	Facebook, Instagram, Pinterest, Uber, Tesla, Walmart, Wix.com, Baidu Mobile, Artsy
Competitive Advantages	Simpler, faster, and elegant native apps using easy-to-use Flutter SDK	More than 3 years in the market with the ability to deliver native-like app experiences

Table 1: React Native and Flutter Comparison [16]

5 DESIGN

The mobile application is called Trill and was designed to be user friendly for both the drivers and the users. To achieve that, two versions of the mobile application were designed: one for the driver and one for the user. As shown in Fig.7, an Android phone is placed in the trolley with the driver version installed to track the current location of the trolley using the phone's internal GPS. The

data that the driver version collected is then being sent to the user version for the users to track the location of the trolley.

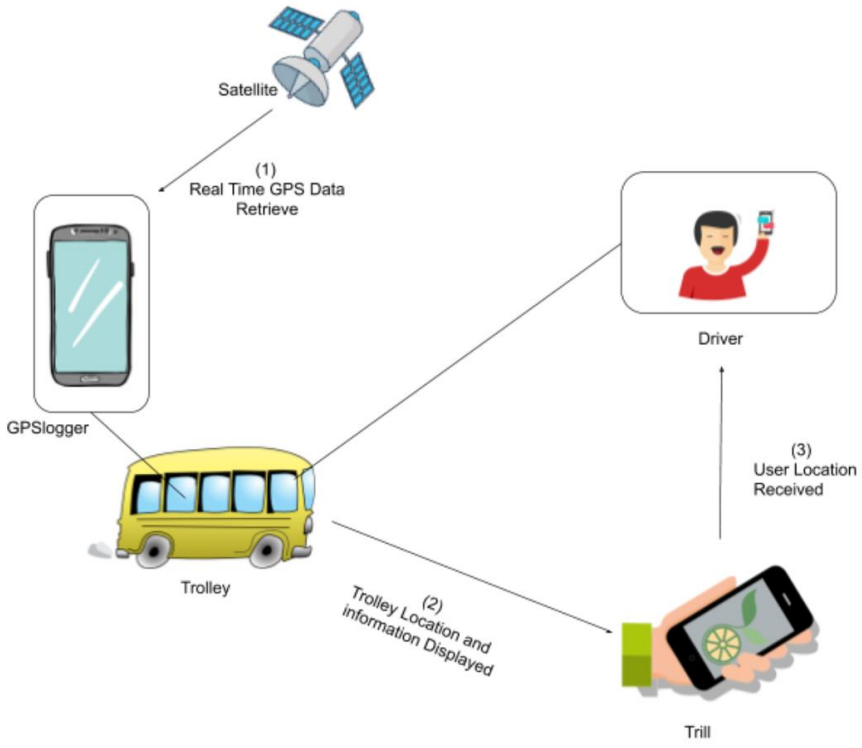


Figure 7: Block Diagram

The user version includes the following features:

- GPS tracking of the trolley's current location with a lag time that is less than 1 second
- Arrival and departure time to and from each stop displayed on the screen when the trolley icon is tapped by the user
- An integrated in-app map
- Daily trolley schedule displayed on a sliding up panel
- Notifications sent to the driver from the users via an interactive pop-up window

The driver version has additional features listed in the following:

- Allows the driver to input the arrival and departure time to and from each stop that would then be displayed on the user version
- Receiving notifications from the users
- Driver can input an alert when taking a detour off-campus

The user version has two frames as shown in Fig.8, the main screen and the sliding up panel. The main screen displays an in-app dynamic map that shows the location of the trolley and the stops. When the users click on the trolley icon, a message should pop up showing the arrival time to or the departure time from a certain location (Fig.9a). When the user clicks on a certain stop, a pop-up window would appear to ask the user whether they want to notify the driver that they are at the stop (Fig.9b). The sliding up panel can be navigated by an upward swiping motion and is a page that displays the daily trolley schedule and trolley alerts.

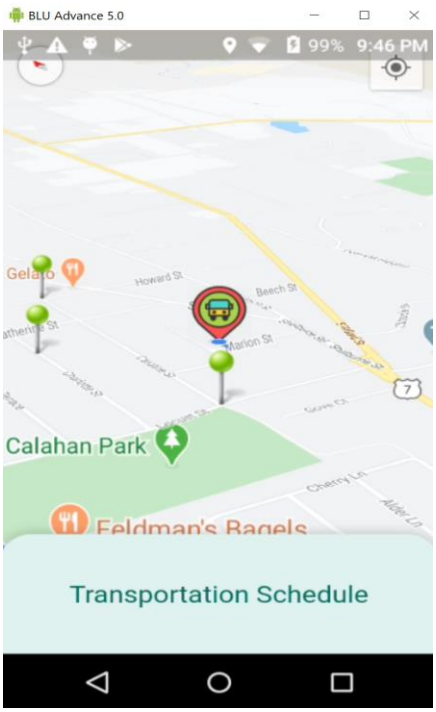


Figure 8a. Main Screen

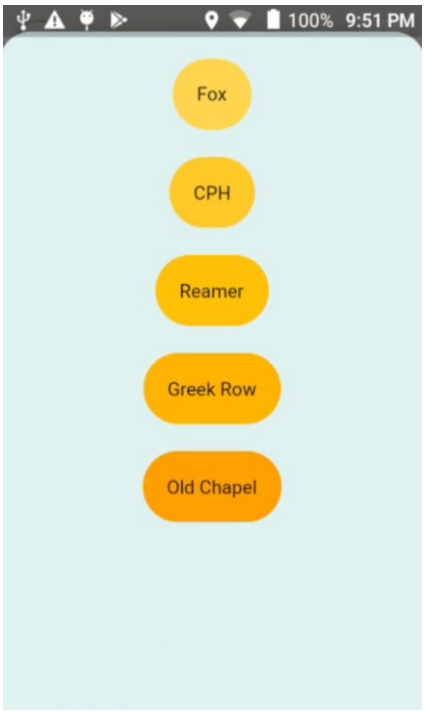


Figure 8b. Sliding Up Panel

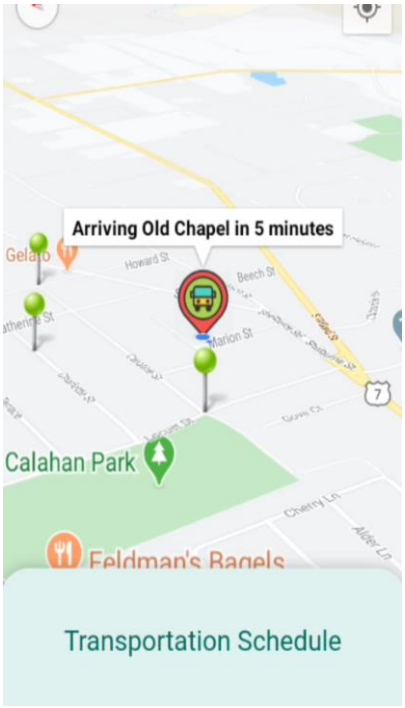


Figure 9a. Pop up Message

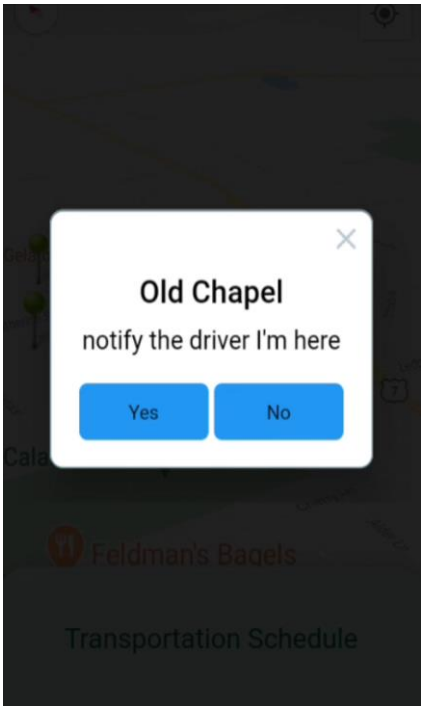


Figure 9b. User Interaction

6. IMPLEMENTATION

As mentioned in section 4.2, I chose Flutter to be the software development kit for the front-end and back-end design tool. To use Flutter, I installed Android Studio, an integrated development environment for Google's Android operating system because Flutter serves as a plugin on Android Studio. In preparation for implementing the design using Flutter, I completed several sections of an online course called "The Complete Flutter Development Bootcamp Using Dart" [17], in which I learned the basics of Dart programming language and some elementary components in Flutter. I continued to use this online course as a resource throughout the app development process.

6.1 App Logo

The first step for the implementation of the mobile app was designing a logo using Adobe Illustrator. As shown in Fig.10, the logo consists of two parts: a wheel and a sprout that embodies the concept of creating an eco-friendly and sustainable means of transportation service.

Figure 10. Trill Logo

6.2 Integrating Google Maps in Flutter

The first screen after users open the app is a dynamic map that shows the location of the trolley (Fig.11). To design this frame, I integrated Google Maps' API into the app. I followed the instructions on the *Google Map platform* webpage [18] and created a credential using my Gmail account on the *Google Cloud Platform* [19]. After my credential was approved, I obtained a free API key and Maps SDK for Android that gave me the access to dynamic Google Maps with optionally supported gestures for zoom and pan, and the Map's 3D camera. After obtaining the API key, I integrated it into Flutter following the instructions on the *pub.dev* webpage [20] to specify the API key in Flutter's application manifest (android/app/src/main/res/AndroidManifest.xml) shown as the following:

```
<application
    android:name="io.flutter.app.FlutterApplication"
    android:label="Trill"
    android:icon="@mipmap/ic_launcher">

    <meta-data android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyBiDNPGVe4DxMBpVu-XSEoF_XEoWQREv9A"/>
```

I then configured the "pubspec.yaml" file in Flutter to include the "google_maps_flutter 0.5.28+1" plugin. After these configurations

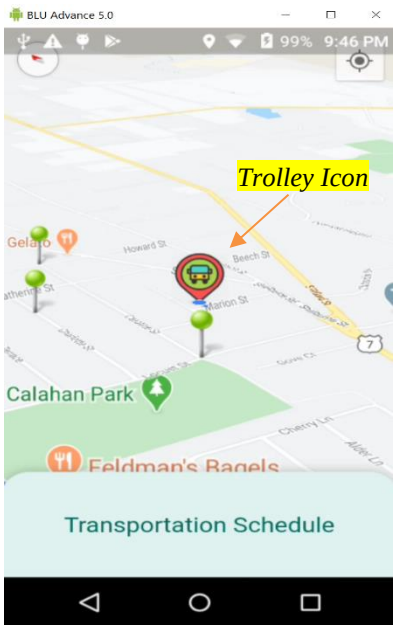


Figure 11. Main Screen

were done, I was able to integrate Google Maps into the app with some code and the "google_maps_flutter" package [21].

6.3 GPS Current Location Tracking

An important feature of this app is the GPS location tracking of the Android phone. To achieve this, I used a Flutter plugin called "location 3.0.2" [22] to import the Android device's internal location to the mobile app and get callbacks when the current location is changed. The variable used to store the location data is declared as the following:

```
location = new Location();
```

To get continuous callbacks when the position is changing and store the information to the location variable, I used the function call

```
location.onLocationChanged().listen((LocationData cLoc)
```

The initial view of the map was set to be the current location of the trolley using the following function:

```
initialCameraPosition = CameraPosition(
    target: LatLng(currentLocation.latitude,
        currentLocation.longitude), // LatLng
```

If the user's device doesn't allow the app to access its location, the initial view of the map would be set to the location of Union College to prevent null pointer issues with the following code:

```
CameraPosition initialCameraPosition = CameraPosition(target: LatLng(42.8177, -73.9296), zoom: 16.0);
```

In which "LatLng()" takes the longitude and latitude of the desired location as the parameters. Refer to Appendix A for the complete code.

To visually represent the trolley on Google Maps, I created a customized trolley icon that represents the trolley (Fig. 12). The .png file of the trolley icon was saved in the assets folder and it was integrated into the app using the function

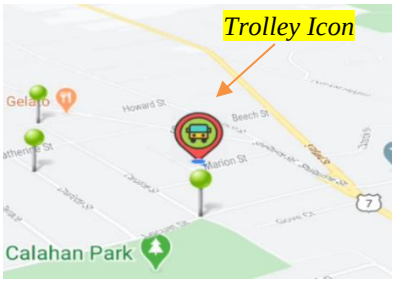


Figure 12. Trolley Icon

```
trolleyIcon  = await BitmapDescriptor.fromAssetImage
```

Whenever the location changes, the trolley icon's position on the map would change accordingly. This is achieved by removing the icon from the previous location and adding it to the new location whenever there is a location change. Refer to Appendix B for the complete code.

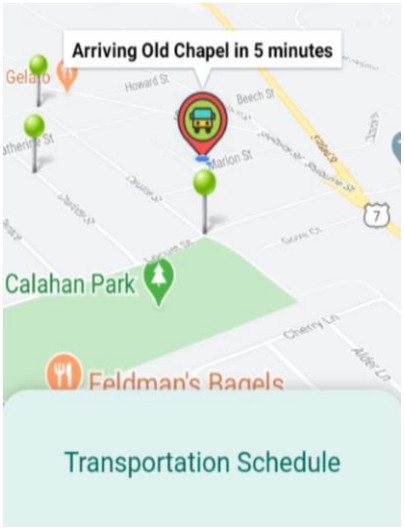


Figure 13a. Collapsed View

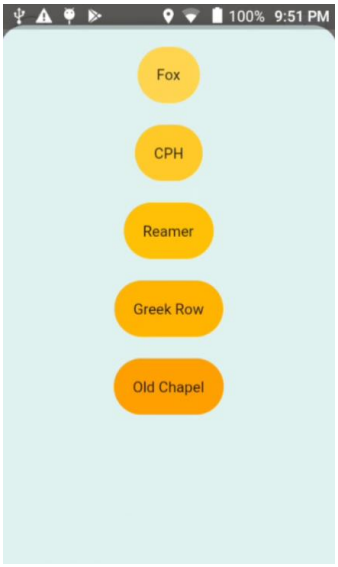


Figure 13b. Expanded View

6.4 Transportation Schedule, User Location Notification, and Customized Icons

I decided to use a sliding up panel to display the daily transportation schedule and alerts of the trolley. Fig.13a shows the view of the panel when it's collapsed, only a small portion of the panel titled "Transportation Schedule" is visible. Fig.13b is the view of the panel when it is fully expanded. The users can simply open the panel with a swipe-up motion on the phone from the

main screen. The implementation was achieved using the "sliding_up_panel 1.0.2" plugin [23]. The code for building this sliding up panel structure on top of the main screen can be found in Appendix C.

To implement the functionality of notifying the driver when the user is at a certain stop. I designed a pop-up screen shown in Fig.14. This pop-up screen would show up when the users click on a stop icon and ask the user whether they want to notify the driver that they are at the stop. The pop-up window is implemented using a "DialogButton" and the function "Navigator.pop(context)" allows for the pop-up feature.

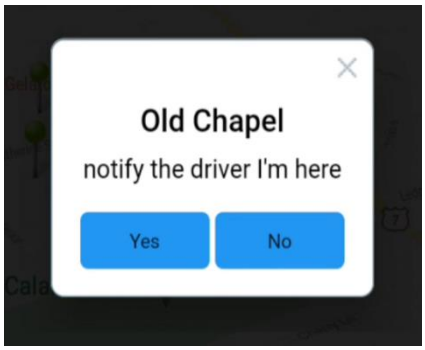


Figure 14. Pop-Up Screen

The code for the pop-up screen and setting up customized stop icons can be found in Appendix D.

6.5 Arrival and Departure Time Display

To check the arrival and departure time of the trolley, the users can simply click on the trolley icon and a pop-up message with the arrival or departure time of the trolley would show up (Fig.15). For this version of the app, the time displayed is hardcoded and does not reflect the real arrival or departure time of the trolley.

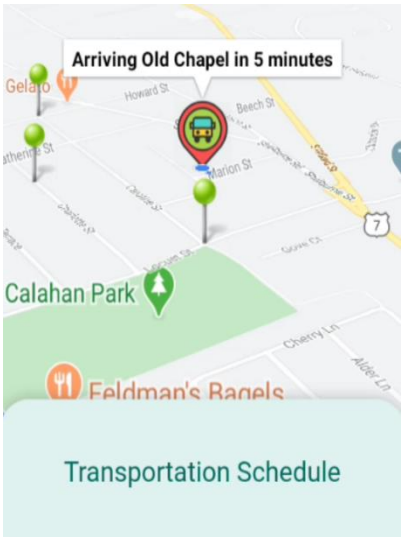


Figure 15. Pop-Up Message

7. RESULTS

The app was deployed to an Android phone (Fig.16) for functional testing to ensure that all its features are responsive and meet specifications. The testing included:

- GPS current location tracking

- Zoom functionality of the map
- Arrival or departure time display
- An interactive pop-up window for user location logging
- Sliding-up panel
- App startup time

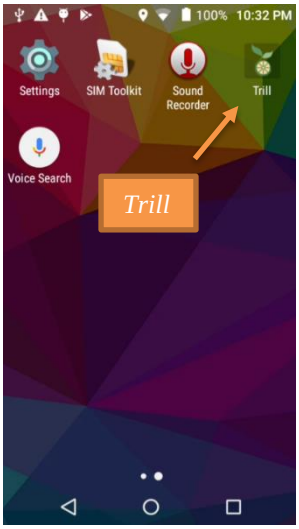


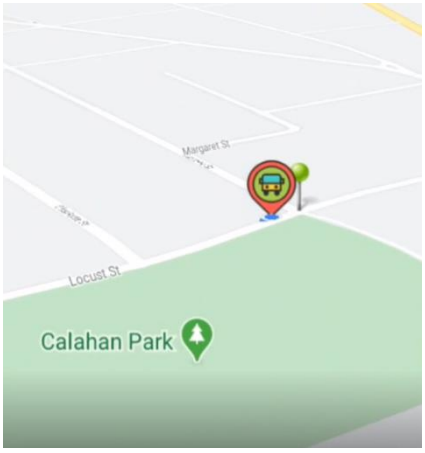
Figure 16. Trill App on Android

7.1 GPS Current Location Tracking

To test the real-time GPS tracking, I placed the Android phone in my car and drove the car around my neighborhood for five minutes while screen-recording the phone. The trolley icon correctly reflected the current location of my car and the location was updated every 2.5 seconds as the car was moving. The screen-recorded testing result could be found on the *Union College Trolley Tracking System* webpage (https://muse.union.edu/2020capstone-yangg/). This partially satisfied the first design requirement stated in section 3 as it tracked the current GPS location of the car but had a lag time 1.5 seconds slower than the required 1 second maximum lag time.

7.2 Map's Zoom Functionality

The map's zoom level could be adjusted by pinching two fingers together or spreading them apart on the screen. The zoomed-in and the zoomed-out map view is shown in Fig.17.



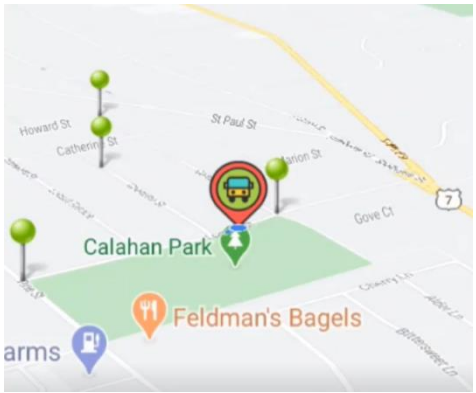


Figure 17a. Zoomed-In View *Figure 17b. Zoomed-Out View*

7.3 Arrival or Departure Time Display

As shown in Fig.18, the arrival or departure time was properly displayed on the screen with minimal lag time when the trolley icon was tapped. The message also informs users which stop the

trolley is heading toward or the stop it will depart from. This feature satisfied design requirement 3a and 3b listed in section 3.

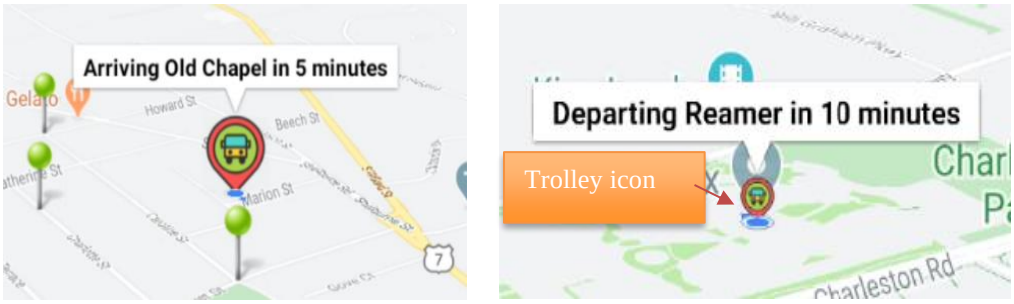


Figure 18. Arrival or Departure Time Display

7.4 Interactive Pop-Up Window

As shown in Fig.19, the interactive pop-up window that asks the user to choose whether they want to notify the driver that they are at a certain location opened with a lag time of 1.3 seconds when a stop icon was tapped. This feature satisfies the second desire requirement listed in section 3 and allows the driver to have an estimate of how many students are at each stop thus better plan the routes. If there were no students at any stops, the driver could stay at their current stop to save gas and time until one or more students are waiting at a stop.

Figure 19. Interactive Pop-Up window

7.5 Sliding Up Panel

The sliding up panel was designed to display the daily transportation schedule. I was able to easily expand and collapse the panel with an upward swiping motion and downward swiping motion shown in Fig.20. This satisfies the fourth desire requirement listed in section 3.

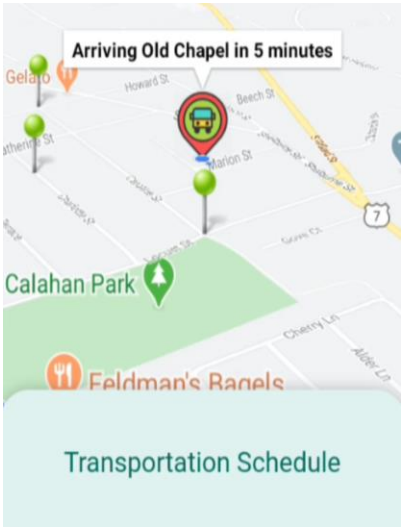


Figure 20a. Collapsed View

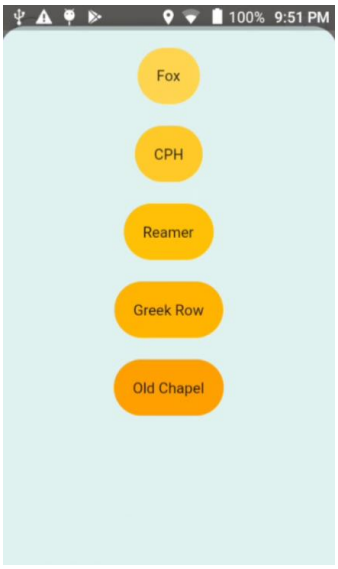


Figure 20b. Expanded View

7.6 App Startup Time

To test the app startup time design requirement listed in section 3. I initialized the app five times (Table 2) to calculate the average startup time to be 3.2 seconds. This result is 1.2 seconds slower than the time specified in the design requirement.

	Run 1	Run 2	Run 3	Run 4	Run 5
Time (s)	3.6	3.3	3	3.1	3

Table 2. App Startup Time

8. FUTURE WORK

As mentioned in the results section, the GPS location tracking had a lag time of 2.5 seconds, which exceeded the maximum of 1 second of the first design requirement listed in section 3. This would be improved to less than 1 second in the next version of the app. The app startup time should also be decreased from 3.2 seconds to less than 2 seconds as specified in the fifth design requirement. As I did not have enough time to fully implement the daily transportation schedule and alerts display, it would be finished in the next version of the app.

The driver version of the app has been fully described in section 5 but has yet to be implemented. The main features are requiring the driver to manually input the arrival time to a certain stop and the departure time from a certain stop. This information would then be displayed on the user's version. It would display notifications about the number of users at each trolley stop. It would also allow the driver to input alerts such as "the trolley is currently on a detour to Walmart".

This version of the app has only been deployed and tested on an Android device because the deployment process is easier and less time consuming compared to an iOS device. The next version of the app would be deployed and tested on an iOS device to ensure it functions properly on iOS devices as well.

9. CONCLUSION

Through the development process of the Trill app, I gathered data from the Union Transportation Department, did extensive research on which tools to use for the app development, specified the

design requirements, learned to use the Flutter development tool, implemented the design, and tested it on an Android device.

The final product successfully met most of the design requirements. It displayed the location of a moving car as mentioned in the results section with a lag time of 2.5 seconds. Although this lag time exceeded the design requirement by 1.5 seconds, it could be improved in the next version of the app. Integrating Google Maps to the app was a challenge but after doing some research and debugging, I was able to successfully integrate it into the app with pan and zoom features. To improve the visual design of the app, I placed customized icons on the map to mark the trolley location and different stops. The third design specification was satisfied by displaying the arrival or departure time to or from a certain on the screen when the trolley icon is tapped. The pop-up window that asks the user whether they want to notify the driver of their location functioned as desired and met the second design requirement. The sliding up panel for transportation schedule and alerts also functioned as desired with minimal lag time.

Building this product was a rewarding process. When faced with a challenge, I learned to ask the most pertinent questions and collaborate with my advisor and faculty of different departments, and seek out online resources, to devise a solution and work toward progress. I not only learned a new programing language and a new cross-platform app development tool, but I also gained experience in building a software product to provide a solution to an existing problem.

REFERENCE

[1] "Trolley Stops and Operation." *Union College*, www.union.edu/campus-safety/trolley-stops-and-operation.

[2] "Bus Tracking App Helps Keep Campus Moving." *UKNow*, 29 Mar. 2016, https://uknow.uky.edu/campus-news/bus-tracking-app-helps-keep-campus-moving.

[3] "Features." *TransLoc Rider*, http://translocrider.com/features#howitworks.

[4] Alamgir, Mohammad & Jahan, Israt & Aktar, Nasrin. (2019). Chittagong University Teachers' Bus Tracking System Using Smartphone Application. 10.1109/ICEEICT.2018.8628098. https://www.researchgate.net/publication/330937599_Chittagong_University_Teachers'_Bus_Tracking_System_Using_Smartphone_Application

[5] "GPSLogger for Android." *GPSLogger for Android*, https://gpslogger.app/.

[6] "SIM808 GSM/GPRS/GPS Module." *SIM808 GSM/GPRS/GPS Module - ITEAD Wiki*, https://www.itead.cc/wiki/SIM808_GSM/GPRS/GPS_Module.

[7] "Mobile OS Market Share in the U.S. 2019." *Statista*, https://www.statista.com/statistics/266572/market-share-held-by-smartphone-platforms-in-the-united-states/.

[8] Saccomani, Pietro. "Native, Web or Hybrid Apps? What's The Difference?" *MobiLoud Blog*, MobiLoud, 4 June 2019, https://www.mobiloud.com/blog/native-web-or-hybrid-apps/.

[9] Chinnathambi, Kirupa. "Understanding WebViews." *Kirupa.com*, www.kirupa.com/apps/webview.htm.

[10] Panchal, Mayur, et al. "These Hybrid App Examples Will Transform Your Perspective Towards Mobile Apps." *Excellent WebWorld*, 21 Nov. 2019, https://www.excellentwebworld.com/hybrid-app-examples/.

[11] K., Vitaly, and Vitaly K. "Native vs. Hybrid App: Considering Pros and Cons of Each Option." *Cleveroad Inc. - Web and App Development Company*, 11 July 2018, https://www.cleveroad.com/blog/native-or-hybrid-app-development-what-to-choose.

[12] "What Is Cross-Platform Development? Definition & More." *Sapho*, https://www.sapho.com/glossary/cross-platform-development/.

[13] Instructables. "Tiny GPS Tracker." *Instructables*, Instructables, 19 Sept. 2017, https://www.instructables.com/id/Tiny-GPS-Tracker/.

[14] “GM862 Cellular Quad Band Module with GPS.” *CEL-07917 - SparkFun Electronics*, https://www.sparkfun.com/products/retired/7917.

[15]. W. El-Medany, A. Al-Omary, R. Al-Hakim, S. Al-Irhayim and M. Nusaif, "A Cost Effective Real-Time Tracking System Prototype Using Integrated GPS/GPRS Module," *2010 6th International Conference on Wireless and Mobile Communications*, Valencia, 2010, pp. 521-525.

[16]. Spec. “Flutter Vs. React Native: Detailed Comparison, Similarities, And Superiority.” *Medium*, Codeburst, 28 Dec. 2018, https://codeburst.io/flutter-vs-react-native-detailed-comparison-similarities-and-superiority-3e92b910fa6e.

[17]. “Creating a New Flutter Project from Scratch.” *RSS*, www.appbrewery.co/courses/548873/lectures/9986032.

[18]. “Geo-Location APIs | Google Maps Platform | Google Cloud.” *Google*, Google, cloud.google.com/maps-platform/?utm_source=google&utm_medium=cpc&utm_campaign=FY18-Q2-global-demandgen-paidsearchonnetworkhouseads-cs-maps_contactsal_saf&utm_content=text-ad-none-none-DEV_c-CRE_274433407135-ADGP_Hybrid | AW SEM | BKWS ~ Google Maps Platform-KWID_43700033921822006-kwd-454773864390-userloc_9003006&utm_term=KW_google map platform-ST_google map platform&gclid=EAIaIQobChMIpNakkOqP6gIVE2KGCh0M9gMJEAAYASAAEgIhc_D_BwE

[19]. *Google Cloud Computing, Hosting Services & APIs*, Google, cloud.google.com/gcp/?utm_source=google&utm_medium=cpc&utm_campaign=na-US-all-en-dr-bkws-all-all-trial-b-dr-1008076&utm_content=text-ad-lpsitelinkCCexp2-any-DEV_c-CRE_81884218687-ADGP_Hybrid | AW SEM | BKWS | US | en | BMM ~ Cloud Google-KWID_43700007265844938-kwd-26365363013&utm_term=KW_ cloud google-ST_ cloud google&gclid=EAIaIQobChMI-fGhuuqP6gIVDACGCh06WgkAEAAYASAAEgJijfD_BwE.

[20]. *Dart Packages*, pub.dev/.

[21]. “google_maps_flutter: Flutter Package.” *Dart Packages*, 21 May 2020, pub.dev/packages/google_maps_flutter.

[22]. “Location: Flutter Package.” *Dart Packages*, 3 Apr. 2020, pub.dev/packages/location.

[23]. “sliding_up_panel: Flutter Package.” *Dart Packages*, 13 Apr. 2020, pub.dev/packages/sliding_up_panel

APPENDIX A

```
@override
void initState() {
  super.initState();
  location = new Location();
  location.onLocationChanged().listen((LocationData cLoc) {
    currentLocation = cLoc;
    updatePinOnMap();
  });
  setSourceAndDestinationIcons();
}

CameraPosition initialCameraPosition = CameraPosition(target: LatLng(42.8177, -73.9296), zoom: 16.0);
if (currentLocation != null) {
  initialCameraPosition = CameraPosition(
      target: LatLng(currentLocation.latitude,
          currentLocation.longitude), // LatLng
      zoom: 16,
      tilt: 80,
      bearing: 30
  ); // CameraPosition
}
```

APPENDIX B

```
void setSourceAndDestinationIcons() async {
  sourceIcon = await BitmapDescriptor.fromAssetImage(
      ImageConfiguration(devicePixelRatio: 2.5),
      'assets/stop.png');

  trolleyIcon  = await BitmapDescriptor.fromAssetImage(
      ImageConfiguration(devicePixelRatio: 2.5),
      'assets/bus.png');
}

void updatePinOnMap() async {
  CameraPosition cPosition = CameraPosition(
    zoom: 16,
    tilt: 80,
    bearing: 30,
    target: LatLng(currentLocation.latitude,
        currentLocation.longitude), // LatLng
  ); // CameraPosition
  final GoogleMapController controller = await _controller.future;
  controller.animateCamera(CameraUpdate.newCameraPosition(cPosition));
  setState(() {
    var pinPosition = LatLng(currentLocation.latitude,
        currentLocation.longitude);
    _markers.removeWhere(
            (m) => m.markerId.value == 'trolley');
    _markers.add(Marker(
        markerId: MarkerId('trolley'),
        position: pinPosition, // updated position
        infoWindow: InfoWindow(
          title: 'Arriving Old Chapel in 5 minutes',
        ), // InfoWindow
        icon: trolleyIcon
    )); // Marker
  });
}
```

APPENDIX C

```
return Scaffold(
  body: SlidingUpPanel(
    backdropEnabled: true,
    color: Colors.teal[50],
    panel: Center(
      child: Column(...), // Column
    ), // Center

    collapsed: Container(
      height: MediaQuery.of(context).size.height / 12,
      decoration: BoxDecoration(...), // BoxDecoration
      child: Center(
        child: Text(...), // Text
      ), // Center
    ), // Container

    body: Center(
      child:Container(
        height: MediaQuery.of(context).size.height,
        width: MediaQuery.of(context).size.width,
        child: GoogleMap(...), // GoogleMap
      ), // Container
    ), // Center
```

APPENDIX D

```
        _markers.add(Marker(
            markerId: MarkerId('OldChapel'),
            draggable: false,
            onTap: () {
              return Alert(
                      context: context,
                      title:"Old Chapel",
                      desc:"notify the driver I'm here",
                      buttons:[
                        DialogButton(
                          child:Text("Yes"),
                          onPressed:(){
                            Navigator.pop(context);
                            },
                        ), // DialogButton
                        DialogButton(
                          child:Text("No"),
                          onPressed:(){
                            Navigator.pop(context);
                          },
                        ) // DialogButton
                       ]).show(); // Alert
            },
            icon: sourceIcon,
            position: LatLng(44.463199, -73.210465))); // Marker
```