SNAPP: Mobile SNAP Application

A Major Qualifying Project in collaboration with Worcester Polytechnic Institute

Submitted To:
   Project Advisor: George T. Heineman

Submitted By:
   Dimitri Berardi
   Nicholas Delli Carpini
   Kenneth Morton

This report represents the work of WPI undergraduate students submitted to the faculty as evidence of a degree requirement WPI routinely publishes these reports on its website without editorial or peer review. For more information about the project’s program at WPI, see https://www.wpi.edu/project-based-learning
Abstract

1.0 Introduction
1.1 Desire for a Mobile Application
1.2 Student App
1.3 Co-Driver App
1.4 Dispatcher App
1.5 Admin Portal Website

2.0 Background
2.1 SNAP System
2.2 Previous SNAP Projects and Research
2.3 Taking over from the IQP and MQP
2.4 Stakeholders
2.4.1 Police Department
2.4.2 IT Department
2.4.3 Marketing
2.4.4 Student Government Association

3.0 Methodologies
3.1 Initial Requirements
3.1.1 Meetings with the Police Department
3.1.2 Meetings with the IT Department
3.1.3 Conducting Surveys
3.2 Refining Design & Scope
3.2.1 Defining the Architecture and Technologies
3.2.2 Narrowing the Features
3.2.3 Scope Change
3.3 Application Development
3.3.1 Student Application
3.3.2 Dispatcher and Co-Driver Applications
3.3.3 Admin Portal and Additional Features
3.4 Finalizing Development

4.0 Design & Technologies
4.1 Overview
4.2 Flutter - Cross Platform App Development Framework 26
4.3 Firebase - Database Backend & Infrastructure 27
    4.3.1 Database Architecture 28
    4.3.2 Push Notification Architecture 29
4.4 Ride Request Overview 30
4.5 Student App 31
    4.5.1 Login & Home Page 32
    4.5.2 Ride Status 34
    4.5.3 Sidebar 36
4.6 Dispatcher Dashboard 36
    4.6.1 Ride Status Columns 37
    4.6.2 Additional Functions 38
4.7 Co-Driver App 40
4.8 Admin Portal 41
    4.8.1 Change SNAPP Hours 42
    4.8.2 Change SNAPP Passwords 42
4.9 SNAPP Shared Elements 43

5.0 Evaluation 44
    5.1 Overview 44
    5.2 Security Review 45
    5.3 Simulation 45
    5.4 Financial Review 47

6.0 Future Work and Ideas 49

7.0 Appendix 52
    7.1 Appendix A 52
        Survey Results 52
    7.2 Appendix B 55
        Simulation Script 56
    7.3 Appendix C 61
        Simulation Script 61
Abstract

The goal of this project is to develop a suite of applications for the WPI community and police department to streamline the SNAP service. We have developed a mobile app for students to request a SNAP ride from their smart phone, a website for a SNAP dispatcher to view current rides and assign them to a van, a tablet app for a co-driver to view a list of their rides, and an admin portal to manage the SNAP service. While we originally planned to conduct in-person trials during the final term of this project, COVID-19 prevented us from doing such. As a workaround, we completed a battery of simulations to ensure the software suite can be readily deployed in the future.
1.0 Introduction

The Student Night Assistance Patrol (SNAP) (https://www.wpi.edu/student-experience/resources/safety/campus-transportation/student-night-assistance-patrol) is a service on campus at WPI that provides free transportation for students by van. This service is used by dozens of students every evening during the academic year, who all currently call SNAP’s telephone number (508-831-6111) to request a ride within a 1-mile radius of campus.

Our project hopes to create an alternate way to request a ride with SNAP using a dedicated mobile application on a student’s phone. A mobile application would allow us to create a system that would both increase the communication between a SNAP van driver and a student requesting a ride and would reduce the need for paper logging in the SNAP department.

1.1 Desire for a Mobile Application

In order to find support for a SNAP mobile app, we put survey QR codes into the SNAP vans. The survey was designed to be short and anonymous to encourage students to fully complete the survey. From our survey, which received 45 responses, over 90% of respondents would prefer to use an application to request a ride rather than the current call-based system.

We also tried to gauge the need for a native mobile application over previously attempted web-based solutions. With a native mobile app, we can design push notifications to be sent to students based upon the current status of their ride. These notifications would inform the student when their ride is “Accepted”, “On the Way”, “Arrived”, etc. From the 90% of survey respondents who would use the app, 100% of them stated they would like to receive notifications on the status of their ride, which can most effectively be accomplished using a native mobile app.
This push notification system could also help with one of SNAP’s biggest issues, namely students missing their rides. From our survey results, over 30% of students have missed their SNAP van in the past due to being unaware that the van had arrived. By creating a push notification system that informs students when their ride has arrived, students can wait safely indoors and be confident that they will be updated on the status of their requested SNAP ride.

For the project to have a modern design and fulfill the requirements of all involved stakeholders, we decided that four separate applications needed to be developed. One app for students to request a ride, one for the SNAP dispatcher to assign rides to a van, one for the van’s co-driver to see their rides, and one for administrators to control various aspects of the SNAP program.

![Figure 1: Our survey results of what percentage of students had ever missed their SNAP ride](image)

- Yes, I was unaware my van had arrived
- Yes, but I was aware my van had arrived
- No

65%
32.5%
1.2 Student App

The goal of the student app is to allow students signed in with their WPI credentials to request a ride with SNAP. This app should be accessible on both the Apple App Store and Google Play Store, and will be free to download. The user interface will be designed similar to Uber (https://www.uber.com) or Lyft (https://www.lyft.com), providing students a user interface that they should be familiar with. After a student sends their SNAP ride request using the app, they will receive updates and push notifications based on the status of their ride, such as “Ride Assigned”, “On The Way”, and “Arrived”. Additionally, we will be implementing a variety of additional features, such as a ride history and viewable SNAP schedule, to provide a full user experience to students.

1.3 Co-Driver App

Each SNAP van has a driver and co-driver. The driver is responsible for driving to the ride requests safely without distractions, and the co-driver receives students’ ride requests from the dispatcher and logs them appropriately. The co-driver is in charge of managing the rides and the order in which they are handled.

The co-driver app will be an application on a mobile tablet to see the list of rides that the dispatcher assigned to them. The co-driver would then be able to update the ride status as they see fit, informing both the rider and the dispatcher of the new status. The co-driver should also be able to view the full details of each of their assigned rides, and they should be able to easily distinguish between rides that the van is currently serving and rides that the van has been assigned.
1.4 Dispatcher App

The SNAP dispatcher handles all incoming requests and assigns them to a SNAP van. It is presumed that the dispatcher is trained and experienced to efficiently determine which van a ride should go to, given the location of each van and their already assigned rides. Currently, a radio system is used for communication between the dispatchers and vans. With our app, the dispatcher would automatically receive new ride requests delivered from the student app. From there, they can assign a ride to a van, and the co-driver would automatically see this ride on their tablet.

Despite the implementation of this new digital system, we have no plans to remove the current phone call request and radio communications system. If a dispatcher receives a ride request from a student calling SNAP, they should be able to incorporate the phone call request into the application system. Additionally, the dispatcher and co-drivers currently communicate using a radio, and we would like to retain the radio in our system to handle any edge cases our applications cannot. The use of radio systems also increases the safety of the SNAP program by allowing quick communication between the co-driver and dispatcher.

1.5 Admin Portal Website

The admin portal will serve as a website for a SNAP admin to control differing aspects of the SNAP service such as the hours of operation and whether SNAP is currently operating. Additional features such as full SNAPP ride history view and data analytics are outside the scope of this specific project, but they could easily be developed once this project completes.
1.6 COVID-19 & Influence on Our Project

Originally, we planned on finishing all development on the four apps and completing a full testing suite by the end of the academic year. To properly test the applications and get our system implemented with SNAP, we requested funding for the co-driver tablets and all associated costs. We received approval for the funding and had planned to use this funding to purchase the tablets; however, COVID-19 drastically changed our goals for ending our project.

The COVID-19 started to affect the WPI community at the beginning of D Term 2020. Classes were moved online, and residence halls were closed. Additionally, the idea of social distancing became prevalent, and the SNAP program was cancelled when the WPI campus was closed (March 24th, 2020). There was much uncertainty in the community, and many of the involved stakeholders in our project had become much less accessible. With these factors in mind, we decided against trying to implement the SNAP application system, and we did not purchase any of the hardware for the program. While we were unable to run a proper test of the system, we have tried to produce a product in the state where it is deployable in the future should WPI want to do such. Part of this was to complete extensive documentation, including Technical Documentation for any teams working on or maintaining the system, and Instructional Documentation for SNAP employees to understand how to use the applications.
2.0 Background

2.1 SNAP System

SNAP was created in 1981 as a collection of students hired to help with basic security tasks around campus. In 1990, the program was retooled into a free transportation service offering van rides to students who felt unsafe walking through Worcester at night. Today, SNAP is operated by both campus police and student employees, and it continues to provide safe transportation around WPI, annually delivering thousands of WPI students to their destinations.

Currently, any student who wants to request a ride must call SNAP by phone. This call is directed to the dispatcher, who records the request on a physical log sheet. The dispatcher then radios the request to one of the SNAP vans, choosing based on the van’s last known position and previously assigned rides. The co-driver verifies the radio request and records it into their personal van logbook. The driver of the van is responsible for determining the best route and order to pick up students assigned to them. When the van arrives at a pick-up location, the driver uses the flashing lights on the top of the van to get the attention of the student who requested the ride. The driver waits a maximum of five minutes for the student to come to the van and continues onto the next requested location if they do not appear. Otherwise, the student’s ID is checked to ensure that they are the WPI student who requested the ride, and the driver brings the student to their requested drop-off location.

2.2 Previous SNAP Projects and Research

In the 2017-2018 academic year, a group of students completed an IQP titled “SNAPP - WPI SNAP Services Assistive Application” (https://digitalcommons.wpi.edu/iqp-all/5203). They set out to analyze the history of SNAP, how it operates, as well as evaluate crime statistics in
Worcester. They researched how other universities such as Duke and Harvard offer safe transportation around campus, which for these colleges is either done through a custom web solution or a third party paid company like Transloc. This research was then used to create a proposal and design documentation for a SNAP ride requesting website.

The IQP team worked on the logistics and requirements for the website, supporting WPI WebTech, a student group on campus, who developed the web app. Their design incorporated two major websites: a student site for requesting rides, and a dispatcher website for monitoring requests and logging ride information. There was discussion of a co-driver app to increase communication between the student and their SNAP van; however, this concept was saved for possible later iterations of the project. These web apps were not completed in time for testing during the IQP, and development was halted in May 2018.

An MQP team in the 2018-2019 academic year revisited the concept of a SNAP application, titling their project “SNAP Application” (https://digitalcommons.wpi.edu/mqp-
Similar to the IQP from the year prior, this team also designed student and dispatcher web apps. Despite these similarities in the projects, the MQP team decided to start their own codebase and not incorporate any of the previous project’s work. This team finished both the student and dispatcher web apps but were not able to incorporate a login system to validate users, so the apps were not adopted by SNAP.

2.3 Taking over from the IQP and MQP

Both the MQP team and especially the IQP team have done much of the preparatory work for a SNAP application, including establishing a need for the app and fostering relationships with WPI IT and the campus police department. The IQP team conducted two major surveys to gauge interest in a SNAP app. The first survey was to determine the most wanted features by the student body for the app. Over 80% of responders would rather use a SNAP app than the current system of calling in a request, and over 60% of responders preferred a native mobile app compared to the IQP team’s web-based design. The second survey was to determine the biggest concerns by SNAP employees for an app. Most of the employees were hesitant about adopting an app into the SNAP program, citing the difficulty to maintain an app and possible abuses of the request system as the biggest concerns.

Using these survey results, along with general recommendations from both the IQP and MQP teams, we developed our overall requirements for the project. We are keeping the name “SNAPP” for the student application from the IQP group and have migrated many of the previous teams’ application requirements and features into our own design documentation. The IQP team also created legal documentation for the app, namely a Privacy Policy and Terms of Use, which we plan to edit and incorporate into our own application.
2.4 Stakeholders

Due to the campus-wide impact of a SNAP application, there are a number of different stakeholders who control varying aspects of the proposed app. The campus police department runs SNAP through both their department and student employees. The WPI IT department can help with technological issues and might be open to helping maintain the project should it be adopted by WPI in the future. The WPI marketing department controls how the app is presented on campus and would make the student app accessible on mobile app stores. The Student Government Association (SGA) controls the budget of the SNAP program and can provide the additional funding needed for the technologies required to run and control a SNAP application.

2.4.1 Police Department

The WPI police department has provided continuous support throughout the process of our MQP, giving us a detailed understanding on how SNAP operates. Our main department contact and the head of SNAP, Lt. Karen Bueno, was quite constructive in fleshing out the requirements and most desired features for the app and informing us of any department-wide restrictions. These restrictions include no GPS tracking in the SNAP vans, a custom blacklist of locations where students cannot be picked up or dropped off, and the app shutting down alongside when SNAP closes.

The police department is a major stakeholder in this project and have requested numerous features for our application. One feature they asked for is a web-based administrator portal that can be accessed from both a computer and a mobile device. Additionally, they requested the ability to do simple data analytics on SNAP’s busiest hours, specifically the number of students missing their van and common areas of high request volume. Although some of the additional features requested by the police department are outside of the scope of
this project, the previously mentioned restrictions are required for implementation before SNAP would be willing to use our application system.

2.4.2 IT Department

The IT department has been our main point of contact regarding the architecture and technology for our applications. For example, they informed us that the WPI login system uses Microsoft’s Azure AD authentication system, which will have to be supported by our app if we want it to be considered for adoption by the student body. Our main IT contact, Dr. James Kingsley, has helped us understand how to integrate both this login technology and secure database technology into our application.

We have discussed with IT maintenance of the application should it be successful, and they have identified several things that would increase their confidence in being able to maintain our app. Firstly, they would like all payments and accounts linked to third-party APIs and frameworks to be created using a WPI administrator account rather than our own personal accounts. Additionally, they requested extensive documentation written for both the code and the general architecture of our project, ensuring future teams can easily understand and maintain our codebase. While we were not able to get official approval from IT to maintain the project due to the COVID-19 pandemic, we have mapped out their requirements should a future team work on implementation in the future.
2.4.3 Marketing

Marketing has shown great interest in this project and wanted to see it implemented by WPI in the future. They helped us in getting access to the appropriate accounts for application deployment on mobile app stores, and they also agreed to help market the app to both students and outside campus, ensuring that this app could be successful and used extensively within the community. Our main contact in marketing is Damien Arlabosse.

2.4.4 Student Government Association

SGA has been instrumental in helping us acquire funding for both one-time hardware and annual costs. For instance, we needed money to purchase mobile tablets with wireless data services for the co-drivers, which would be used to manage the rides they are assigned to. There are also some APIs that we use which are on a pay per use plan and have extenuating operating costs associated with them. We presented our project and funding request to the SGA senate and got approval for funding of the one-time purchases, namely the tablets.
3.0 Methodologies

3.1 Initial Requirements

Starting the project, we wanted to build strong relationships with any and all involved stakeholders, with the goal of developing a solid plan for developing the apps and getting them ready for adoption by WPI. Before meeting with these groups; however, we created our basic requirements and restrictions for the app system using information provided by the previous teams, formulating these details into a presentable pitch.

3.1.1 Meetings with the Police Department

Since the police department manages the SNAP program, we wanted to ensure that we could maintain a close line of contact with someone there. Our main contact was Lt. Bueno, and we discussed with her several different features that could be implemented into the apps. We decided that it was important for a student to be able to request a ride from their mobile device,

![Figure 3: A visual example of the one-mile radius in which SNAP operates](image-url)
and that the SNAP dispatcher and co-driver should be able to view the ride and update the student in real time on the status of their ride. We also discussed additional features, such as a tool to help the SNAP program with the maintenance of vans when closing at the end of the night; however, we eventually determined that this feature was outside the scope of our project.

We also discussed a number of restrictions placed on the application system due to the involvement of the police department. These restrictions included no GPS tracking in the SNAP vans, all data and database servers had to be located in the United States, and the student app could not allow ride requests outside of a 1-mile radius around campus. The one exception to this last rule is Union Station, a train station in Worcester slightly outside of the one-mile radius. After scoping out the project with the police department, we shadowed a SNAP dispatcher to understand how they receive and assign transportation requests. This gave us an opportunity to see exactly which aspects of a dispatcher’s job can be improved with an application.

3.1.2 Meetings with the IT Department

We also conducted a number of meetings with the IT department, namely with James Kingsley and Sia Najafi, to ensure that the school had the technical resources to support the app after our graduation. Initially, IT was open to help maintain the app after our graduation so long as we provided sufficient documentation, a robust system, and a codebase that would not require any major changes, updates, or fixes. However, due to COVID-19, we were unable to get a definite answer as to whether they would be open to maintaining the app should it be adopted by WPI.
3.1.3 Conducting Surveys

To ensure that our project had a prospective audience, we surveyed the student body’s opinion on a SNAP app. We created a short Google Form that assessed: who wanted a SNAP app, what features they would want in a SNAP app, and what aspects of the SNAP program can be improved with an application system. After this, we created a poster with a QR code that linked to the survey, and we put these posters up in the SNAP vans, getting responses over a period of a few weeks. After analyzing the results, we found that the overwhelming majority of students wanted a SNAP app. We also concluded that a SNAP app could increase the program’s efficiency and safety, as students often wait outside for the SNAP van to arrive, or miss their SNAP entirely, due to a complete lack of communication between SNAP and students. The entirety of the survey results can be found in Appendix A.
3.2 Refining Design & Scope

As the project continued, a large portion of time was spent remaining in contact with all stakeholders, creating app mockups, and deciding on the tools and technologies that we would use to develop the apps. We first wanted to visually understand what the various apps would look like before we began programming, and so we used Adobe XD (https://www.adobe.com/products/xd.html) to create user interface mockups of the student, dispatcher, and co-driver apps. While the dispatcher and co-driver mockups were static images, we created a reactive student app mockup, as the student app is both the most important and most complex part of our project. These mockups helped us, and our stakeholders understand what the apps would look like, and why they would make it easier to request a SNAP ride.

![Image of mockup design](image.png)

*Figure 4: Our initial mockup design for the student application. This is the home screen where a student would make a ride request from.*
3.2.1 Defining the Architecture and Technologies

Just as important as UI was the architecture and structure of our applications. We created diagrams describing how the different apps would interact with each other, and how a main database would essentially store all information required for the system. We wanted the apps to be interconnected, such that a change in one app would automatically be reflected in the other app. For instance, if a student requests a ride, the dispatcher should instantly see the
ride come in on their app without having to do anything. We researched a number of application frameworks and databases, and eventually decided on using Flutter (https://flutter.dev) as the framework for our apps, and Firebase (https://firebase.google.com) as a database host. Flutter and Firebase’s Firestore incorporate very well with each other and make it easy to have UI elements in the app update as something changes in the database.

3.2.2 Narrowing the Features

After settling on the architecture and design of our apps, we wanted to narrow down which features would be available in each application. In the student app, we determined it was necessary for a student to be able to request a ride and receive notifications as the status of their ride changed. The student app also had to be locked behind the WPI Azure AD login, as to only allow WPI students and staff to use the app. We also discussed additional features, such as ride history, viewing of the SNAP and shuttle schedule, and being able to cancel a ride. For the dispatcher app, we decided that they should be able to view a list of new ride requests, be able to assign each one to a van, and they should be able to manually create a ride request for someone that calls in their request rather than using the app. We also wanted the dispatcher to have the ability to reject a ride and edit the details of a ride request. For the co-driver app, we felt it was necessary for them to have the ability to view a list of rides assigned to them, and mark and update the status of any given ride.

3.2.3 Scope Change

There were also some features discussed that never actually made it into our application. First, we were unable to implement a blacklist. SNAP wanted a way for the system to automatically filter our requests to any commercial locations. With our tight time constraints and eventually the impact of COVID-19, it became much more difficult to completely implement
this feature, and instead, we gave the dispatcher the ability to reject a ride if it involves an invalid location. Another feature that never made it to our final application was to let the dispatcher send a custom notification to all students. We felt that the current email system for sending notifications worked well, and that there was no need to integrate a custom notification system into the applications. As time passed, the scope and requirements further changed and evolved. Initially, we wanted to give students the option to request a ride to/from Union Station, which is outside the one mile radius of SNAP; however, we felt that it was best for students to call in a ride for Union Station instead of using the app. Additionally, in our later discussions with Lt. Bueno, we decided that the dispatcher app should be web-based instead of our original plan to use a native app on a dedicated dispatcher tablet. Luckily, this change was not difficult to do, as Flutter is multiplatform between both native mobile operating systems and web browsers, allowing us to keep much of the same design from when the dispatcher app was conceived as a native tablet app.
3.3 Application Development

As for the actual development of the applications, we started our work with the student application. The student app was conceptualized to be easy to use for the entire student body, and we felt as though it was important to give the student app as much development time as possible to make this a reality. Additionally, the student app would have to interface with Google Maps APIs, and so we needed time to understand and estimate the costs associated with the API.

3.3.1 Student Application

The student app was developed first, and so the entire team experienced Flutter for the first time working on it. We followed the Flutter install guides and created a GitHub repository for the student app using the ‘Flutter Tutorial App’ as a base. The first part of the application we developed was the user interface, learning Flutter and its unique aspects along the way. Fortunately, the Flutter documentation is easily accessible and detailed, and there are a large number of third-party libraries to expedite feature development and implementation. These factors, along with the large community of flutter developers on sites such as Stack Overflow and Medium, meant that we were able to learn and develop in Flutter quickly. After creating the UI, we moved to implement Firebase Firestore so that ride requests could be added to an external database. Although no one on the team had much experience updating asynchronously to a network database, the integration between Firebase and Flutter made communication between the student app and Firestore efficient and simple to develop.

The first major roadblock we faced was implementing the WPI Azure AD login system to work with our application. We consulted with James Kingsley for guidance on how to incorporate this login system. Through multiple meetings, Kingsley was able to help us in not only restricting the student app to only users with a valid WPI login, but also in getting user
information back from a successful login, allowing features such as a ride history to become possible. After developing a login screen, pulling user information from login, connecting successfully with our Firebase database, and creating the general UI for the student app, we believed the app had become fully capable of its main functionality: allow a student to request a ride with SNAP.

### 3.3.2 Dispatcher and Co-Driver Applications

After finishing the main functionality of the student application, the team split to work on both the dispatcher and co-driver applications. These apps were worked on together because they share a lot of the same major functionality and were the last two integral parts to a functional SNAPP system. We created both applications’ UIs for a horizontally bigger screen, with the dispatcher app meant for a computer screen and the co-driver app designed for a tablet screen. With our experience with Flutter, we were able to implement Firebase Firestore and its appropriate connections with the UI quickly, essentially completing the main functionality of both the dispatcher and co-driver applications. After all three major applications had their main functionality implemented, we were able to simulate the SNAPP system working with a student logging in and requesting a ride, that ride being assigned to a SNAP co-driver, and that co-driver sending ride status updates to the student until their ride is complete.

### 3.3.3 Admin Portal and Additional Features

Once the full ride request lifecycle could be emulated, we began working on the auxiliary features of the student and dispatcher applications. The student app was worked on the most in this regard, as it has by far the most non-essential features. A major feature that we wanted to implement was push notifications to students when their ride status was updated. We were able to accomplish this on Android using Firebase Cloud Messaging.
(https://firebase.google.com/docs/cloud-messaging); however, we were not able to test notifications on iOS due to the need for a paid Apple Developer Account. Some of the additional features, including ‘My Location’, address entry autocomplete, and user destination validation were achieved using the Google Maps API. Using the Google Maps API was a learning experience however, as we originally designed user location to be updated in a stream while the user was in the student app. This cost us almost $800 in API calls in a single night, and luckily, we were able to both get a refund for the absurd number of API calls and figure out a more resource efficient way to get a user’s location. Outside of the additional features powered by Google Maps, we also added new screens for information not easily accessible for WPI students, such as the Gateway and Evening Shuttle schedules. As for the dispatcher app, the entirety of the co-driver app’s functionality was added so that the system could still function if the co-driver application was not in service for whatever reason. Due to the change from a tablet-native app to a website, the dispatcher app needed a login screen that the original requirements never considered and was given one using Firebase Users. Additionally, the dispatcher app was given the ability to open/close the SNAPP system, disallowing students to request new rides through the student app when the system is closed.

All the apps were built in Flutter, and because they shared a good deal of backend functionality, we created our own library for each of the apps to incorporate. This shared library includes much of our Firebase Firestore functions, along with some general shared UI and string constants that are used multiple times throughout each app. Also, although we originally planned to create the admin web portal without using Flutter, between our experience in Flutter and the newly developed ‘SNAPP Shared Elements’ package, we began work on creating the admin portal as a Flutter Web project.
3.4 Finalizing Development

During the development of the additional features for the applications, we pitched the working SNAPP system to a number of groups on campus. First, we contacted the WPI Academic Technology Center (ATC), as we knew the co-driver application would need tablets with mobile data to be functional. With the ATC, we were able to understand the pricing and process of getting a tablet with WPI’s Verizon data account. After creating an cost breakdown for the SNAPP system including information from the ATC and an estimated number of API calls per ride request, we presented to SGA to get funding towards both the tablets needed for the co-driver applications and the annual costs associated with the mobile data for the tablets and API calls made by the student app. Once securing the funding in writing by SGA, we presented to WPI Marketing to ensure that the SNAPP student app would be known to students by the time the system is implemented.

We originally planned to also meet with the WPI legal department to create a Privacy Policy and Terms of Use for the student application, and we planned to meet with IT and marketing to finalize the hand-over of the applications; however, the influence of COVID-19 made that impossible. Without the ability to meet with these groups, we were not able to set up the appropriate channels of payment and communication that would allow us to hand off the SNAPP system to WPI. This forced us to turn down SGA’s funding as well, due to the fact that we were no longer implementing this system during our project. We retooled our final weeks’ plan: from trying to finalize and pass off the project to WPI and our stakeholders to trying to finalize and pass off the project to a future team that can hope to implement the development and designs we have done.
4.0 Design & Technologies

4.1 Overview

After talking with the associated stakeholders, we determined that there are four different applications that need to be developed to meet our objectives. First, an app for students to request a SNAP ride and find more information about the service. Second, an app for SNAP dispatchers to manage and assign requested student rides in their queue. Third, an app for co-drivers to see the rides assigned to their van and send push notifications to students, informing them of the status of their SNAP. Fourth, an administrator app to manage the SNAP service and provide data insights for admins. For each app’s codebase, we used Google’s Flutter, a platform for building cross-platform apps for development. As for our database backend, we used Google’s Firebase due to its modern security and natural integration with Flutter.

4.2 Flutter - Cross Platform App Development Framework

Flutter is an open-source software development framework built on top of Google’s Dart (https://dart.dev/) programming language. Flutter was originally developed in 2015 as a C++ rendering engine for Dart on Android and has expanded since then to allow a shared codebase to run on all major platforms. Flutter is currently supported by a strong developer base, and companies such as Tencent and The New York Times have used the framework to develop their own cross-platform application.

The main reason we chose Flutter was to expedite development. Flutter is a cross-platform framework providing developers with the tools to allow the same exact code to run on both Android and iOS, which is imperative if we want our application to be adopted by the entire student body. Flutter is very efficient compared to other cross-platform SDKs as well, as it
compiles directly into ARM instructions that can communicate with platform-specific modules. Additionally, Flutter has many libraries directly built into the framework, reducing the need to rely on third-party APIs for important functionality. Finally, as Flutter is created by Google, it integrates easily with Firebase and the Google Maps API, which are both critical to our design of the SNAPP system.

4.3 Firebase - Database Backend & Infrastructure

When considering the database framework to use in our application, there were a few main things that persuaded our decision. First and foremost, since Firebase was also developed by Google, it integrates efficiently and easily into our Flutter based projects. Additionally, Firebase has Google Analytics built in, letting us evaluate client behavior, report on broken components of the app, and analyze user interaction with the app. Firebase also uses a real time database to sync data seamlessly between all connected devices, giving users immediate updates when elements are changed within the database tables. Firebase has industry standard security and tools to define custom security rules, ensuring only authenticated users can access the database.
4.3.1 Database Architecture

The diagram above shows the different apps and tools we are developing, and the databases that they interact with. For instance, the co-driver app has access to the cancelledRides and currentRides database collection. We also developed a node.js service for notifications using firebase cloud messaging. The only database collection that it needs to access is the fcmToken table.

When designing the database architecture of our system, we wanted our apps to update seamlessly when information in the database changes. Firebase is a Backend as a Service NoSQL database, meaning that apps listen, or subscribe, to the database, updating accordingly whenever values in the database are changed. At a lower level, we implement this listener/subscriber model in Flutter using streams (https://dart.dev/tutorials/language/streams). The main point of a stream is to receive an asynchronous sequence of data by listening to some stream API. Firestore lets you subscribe to a database collection using a stream so that when something in that database changes, the apps that are subscribed through a stream will be...
notified of the change. For instance, the student app will automatically update itself when information in the currentRides, fcmTokens, rideHistory, or snapHours database collection changes.

It is important to recognize that given the nature of our database as a central hub for all our applications, it could be a major point of failure. If Firebase were to go down or there was any issue with our database, it would render the entire system unusable. Despite this, we are confident that this is a manageable risk, as Firebase has a 99.5% uptime guarantee, and if the database were to go down, SNAP could still use the call-based system they currently have in place.

4.3.2 Push Notification Architecture

As previously mentioned, students can receive push notifications on their device when the status of their ride changes. We implemented notifications using a Firebase service called Firebase Cloud Messaging (FCM). First, we wrote Node.js code that would run when a ride status was updated. This code is deployed and hosted alongside our Firestore database, and it is run automatically when a dispatcher updates the ride status, sending a notification to the corresponding student app alerting them of the update.

![Diagram visualizing the FCM pipeline used for push notifications](image)

*Figure 8: A diagram visualizing the FCM pipeline used for push notifications*
4.4 Ride Request Overview

The diagram above illustrates the basic flow of a student requesting a ride. First, a student completes a form of information regarding their ride and submits the request. This ride is added to the currentRides database, updating the dispatcher app of the new request. The dispatcher then has the choice to reject the request, and if they do, the student will be notified of the rejection. Otherwise, the dispatcher assigns the ride request to the appropriate van, adding the request to the co-driver app. The co-driver can update the ride status, and the student will receive a notification when the ride status is updated.
4.5 Student App

The student app is at the center of our system, as it addresses and solves the main issue which this project set out to. At its core, it allows any student at WPI to request a SNAP ride within a 1-mile radius of campus. It has a variety of additional functionality, including letting students receive updates on their ride status, letting students see their ride history, and a viewable schedule for SNAP and the Gateway and Evening Shuttles.

Figure 10: The login screen of our completed student application
4.5.1 Login & Home Page

When users first download the application, they are presented with the WPI Azure AD login screen. Users must log in using their WPI credentials, validating that they are a part of the WPI community in order to access any of the application’s functionality. It is impossible to bypass this screen without logging in, and therefore it is an important security measure. While our app itself does not store or interact with the login credentials, the aad_oauth (https://pub.dev/packages/aad_oauth) package we use does cache credentials using it's built in system, making it so the user doesn't have to log in each time they launch the app.

Once the user has successfully logged in, they will be brought to the main/home screen of the application. Since the student app revolves around requesting a SNAP ride, we decided that they should be able to do this directly from the home screen. The home screen consists of three main parts: an interactive map, a ride request form, and an expandable navigation menu. To request a ride, users must fill in at least 3 of the 4 sections of the ride request form. The required sections are “Pickup”, “Drop Off”, and “Number of Passengers”. There is also a section called “Additional Information”, which is 250 characters or less and can be used to inform the SNAP operators of any relevant circumstances regarding a ride request. The Pickup and Dropoff fields have autocomplete suggestions, powered by the Google Maps Autocomplete API, which show the valid addresses SNAP will go to based on the user’s input. If an autocomplete suggestion is not chosen by the user, then the same API is called on their fully entered address string, validating that there is exactly 1 address within the 1-mile radius autocompleting from the user’s input, otherwise rejecting the invalid input. The Pickup field also has a “My Location” option, which will use the Google Maps Geocoding API to auto fill the field with the closest address to the user’s phone’s current geographical position.
The map behind the ride request form incorporates visual flairs to make interacting with the application more enjoyable and adds a professional touch to the UI/UX. We visually show SNAP’s 1 mile radius using a translucent blue circle on the map, showing users the area of valid SNAP requests. Additionally, when a user has at least one location, either Pickup or Dropoff, a pin is dropped on the map at that location and the map smoothly zooms in on the pin with comfortable padding. When both the Pickup and Dropoff locations are entered, the map will

![Welcome to SNAPP](image)

*Figure 11: The home screen of our completed student application. This is the screen where a student makes a ride request by filling in their ride details*
zoom and center itself between pins representing both addresses. When all locations are cleared the map will reset to its initial zoom and orientation. Once a user has completed the ride request form, the "Next" button, which is always located at the very bottom of the screen, will show a summary of the ride request and ask the user to confirm the details before submitting the form. If any of the required fields are missing or invalid, the field will be outlined in red and the user will be shown a prompt to complete each field before being allowed to submit the form. Once the user taps the "Confirm" button after they have reviewed their information, they will be brought to a ride status page.

4.5.2 Ride Status

Once a user has successfully completed the ride request form and has submitted their request to SNAP, they will be brought to a ride status page. This page is much simpler and requires very little user input. The top half consists of a status message, a circular progress indicator, and an icon which represents your current status. The bottom half is a summary of your ride request and a button to cancel your ride. When you first submit a ride, the status will be set to “Ride Request Received” and the progress indicator will be completely red. As the co-driver updates the ride status, the status message will update to the following (in this order): “SNAP Ride Queued”, “SNAP On The Way”, “SNAP Has Arrived”, “Rider Picked Up”, and “SNAP Ride Complete”. With each status change, the progress indicator will increasingly fill green. If a user needs to cancel their ride for any reason, they can do this before their ride status is updated to “SNAP On The Way” by selecting the "Cancel Ride" button at the bottom of the screen. This will prompt the user to input a mandatory reason for their cancellation and confirm their selection, cancelling the request and sending the user back to the home screen. If the ride has entered the “SNAP On The Way” status, then the cancel ride button is hidden, and it will not be possible to cancel a ride. When a ride gets to “SNAP Ride Complete”, a "Return Home" button will be displayed allowing a user to return to the home page.
close the app and reopen it while they still had a ride in progress, they would be immediately taken straight to the Ride Status screen with their current status being unaffected. In addition, the dispatcher can decline a ride request, and if they were to do this, the student who made the request would be presented with a popup reading “Sorry your ride was rejected” with a rejection reason as given by the dispatcher. The only way out of this notification is to hit the “Return Home” button, bringing the user back to the home screen. Finally, if a rider does not show up within 5 minutes of their SNAP van arriving, they will be marked as a “SNAP Rider No Show” and a popup will be displayed on their student application. It will tell them why they were marked as this and ask them not to do it again. Again, the only way out of this popup is to hit the “Return Home” button which will bring the user to the home screen.
4.5.3 Sidebar

You can access additional functionality on both the home screen and the ride status screen by clicking on the hamburger menu icon in the top left-hand corner of the screen. This will pop out a side menu with the following options: “Call SNAP”, “SNAP Schedule”, “Ride History”, “Gateway Shuttle”, “Evening Shuttle”, and “Union Station”. The “Call SNAP” option simply prompts a call to the SNAP number, connecting a user to the dispatcher on duty. The “SNAP Schedule” shows the current schedule for SNAP for each term and breaks. “Ride History” will display every ride request a user has ever made, sorted by most recent. It shows all ride request information, including the date and time of the request and the final status of each ride. The “Gateway Shuttle” and “Evening Shuttle” options display the pickup location, drop off location, and time of each stop these shuttles make. Both shuttle schedules have the option to filter by a specific location and filter out all events prior to the current time. Finally, the current implementation for “Union Station” is the same as “Call SNAP”, as it prompts the user to make a call to the SNAP number. We hope that in the future Union Station requests can be integrated into the app as its own separate screen, with the “Union Station” sidebar option leading to that page. Unfortunately, at this point in time, the only way to make a request to or from Union Station is unchanged and still requires a student to call SNAP.

4.6 Dispatcher Dashboard

The Dispatcher dashboard is designed to be hosted online and acts as the control center for all ride requests, with full control over almost all aspects of a ride. When a dispatcher first visits the dispatcher webpage, they are prompted to login with the dispatcher credentials set by the admin. Any incorrect credentials result in an error message alerting the dispatcher to which part of the credentials are incorrect. Once the dispatcher has logged in, they are presented with one singular dashboard from which they can see and control everything. There are three
buttons in the top right corner labeled “Add Ride”, “Close/Open SNAP”, and “Logout”. The rest of the page is split into 5 columns labeled “Unassigned”, “Van 1”, “Van 2”, “Rejected Rides”, and “Cancelled Rides”.

4.6.1 Ride Status Columns

When a user first makes a ride request, their ride will appear as a card in the “Unassigned” column. This means that a user has requested a ride and, at this point, only the dispatcher can see it. The card displays the “From”, “To”, Current Status and “Number of Passengers” on the front, so that a dispatcher can quickly scan a card and get all the information they need to make a decision about that ride. Below this information are three buttons labeled “Unassigned”, “Van 1” and “Van 2”. These buttons allow the dispatcher to move the ride between the three columns easily. Finally, there is a pencil icon in the top right of every card, which allows the dispatcher to make edits to the ride details and status. When the dispatcher selects the pencil icon, a popup for that ride request appears with three sections. In the “Edit Ride Details” section, the dispatcher is presented with editable text boxes filled with the “Pickup” and “Dropoff” addresses, the student’s “Additional Information”, and the “Number of Passengers”. The dispatcher can change the value of any one of these text boxes to edit the ride details. Once they have made the desired changes, they can click the “Submit Changes” button, and confirm or cancel the changes. The next section is “Ride Actions” which currently consists of one red button labeled “Reject Ride”. If the dispatcher wishes to decline the selected ride, they would select this button and be presented with an additional popup. Here they must give a reason for the rejection and then either confirm or cancel their selection. If they confirm their selection, the ride will be moved to the “Rejected Rides” column and will not be editable. The card will be cleared from the “Rejected Rides” column once the student has confirmed that their ride was rejected. The third and final section is “Change Status” which has a dropdown allowing the dispatcher to edit the ride status manually. If a student cancels their ride at any
point, the card associated with that ride will be automatically moved to the “Cancelled Rides” column. Once it is in this column, the only action that can be taken on it is pressing a “Delete” button which will appear at the bottom, that will clear the card completely from the dashboard.

![Diagram of SNAPP Dispatcher](image)

*Figure 13: The final design of our completed dispatcher portal.*

4.6.2 Additional Functions

In addition to managing rides sent in through the student application, the dispatcher can manually create a ride by clicking the “Add Ride” button in the top right of the page. This will create a popup with similar fields to the student application ride request form. The dispatcher must give a “Pickup” and “Dropoff” address; however, these textboxes are completely unrestricted as we assume the dispatcher is aware of the constraints for a valid SNAP request. They also must give a number of passengers and can optionally add 250 characters of additional information. Finally, they must either assign it to the Van 1, Van 2, or the Unassigned column, depending on their specific circumstances. Once all required fields have been completed, they can save the ride and it will be shown on their dashboard. If they choose the cancel option, they will be brought back to the main screen. If they select save without having
filled in all required fields, a popup will appear, directing them to the missing field. Clicking anywhere outside of the popup has the same effect as clicking the cancel button and will bring them back to the main screen.

The next option presented in the top right corner is the “Close/Open SNAP” switch. At the beginning of a shift, SNAP is assumed to be closed and therefore this switch will read “Open SNAP” and will be green. The dispatcher needs to click this in order to open SNAP, therefore making the student app functional. When SNAP is closed, students cannot make ride requests from the student application. When the toggle the switch, it will read “Close SNAP” and be red. The dispatcher must yet again toggle the switch at the end of the night to close snap and disable the student applications. The final button is the logout button, which is self-explanatory. When the dispatcher’s shift is over, they should press this button to logout of the system and get sent back to the login screen.
4.7 Co-Driver App

The co-driver application is the simplest, as it requires very little user input and displays only as much information as necessary. When a co-driver shift starts, they will grab a tablet to use for their shift. Once they have logged into the device and launched the application, they will be presented with one main screen consisting of two columns and a dropdown menu in the middle of the top of the app. The dropdown menu allows the dispatcher to select which van they will be in for that specific shift. If they select Van 1, all rides assigned to the Van 1 column by the dispatcher will appear on their device, and likewise for Van 2. When the dispatcher assigns a ride to a co-driver's van number, the ride will appear as a card in the “Assigned Rides” column. This column represents all rides that have been assigned to that van that are not currently in

![Figure 14: The final design of our completed co-driver application](image)
progress; akin to a backlog. These cards display the “From” and “To” addresses, the “Number of Passengers” and any provided “Additional Information”. They also have a dropdown at the bottom of each card, which allows the co-driver to change the status of the ride. Finally, there is an information icon in the top right of each card which displays a more detailed version of that ride’s details. A ride is only in the “Assigned Rides” column if the ride status is “Ride Request Received”. Once the co-driver changes the status to “SNAP Ride Queued”, the request moves into the “In Progress” column and stays there for the duration of the ride. If a student does not show up at their “Pickup” location, then the co-driver can select the “SNAP Rider No Show” status. This will prompt them with a popup asking them if they are sure, as this action cannot be undone. If the co-driver confirms the no show, they will be brought back to the main screen and the ride will disappear. The same sequence of events occurs when the co-driver marks the status as “SNAP Ride Complete.” If a student cancels a ride that has already been assigned to a Van, a notification will pop up on that Van’s co-driver application telling them that a ride that was assigned to them has been cancelled, and the ride will disappear from the app.

4.8 Admin Portal

The admin portal is a basic application for doing tasks that only someone who runs the SNAP service should be able to do. It will be hosted online and will require a login with the admin’s credentials, which differ from the dispatchers. Once the admin has logged in, they will be brought to the Admin Home Screen. Currently, the home screen has nothing on it other than text reading “Admin Home Screen”, a sidebar menu icon and a “Close SNAP” switch. This toggle switch appears on every page of the admin portal and grants the admin complete control over opening and closing SNAP. It functions identically to the button on the dispatcher portal, enabling and disabling the student app as SNAP opens and closes. The navigation menu has two options: “Change SNAPP Hours” and “Change SNAPP Password.”
4.8.1 Change SNAPP Hours

The “Change SNAPP Hours” page consists of three sections for A/D term, B/C term and Recess/Break hours, each of which have an open and a close. Each of the open and close consists of two dropdowns, the first having the numbers 1-12 and the second having AM and PM. By changing these values in, the admin is capable of changing any of the open and close times for each of the three periods. Once the admin has made any desired changes, they must click the submit button prompting a confirmation popup to “Cancel” or “Submit”. If they submit, these changes will be saved and propagate back to the student application. If they cancel, they will be sent back to the previous screen.

4.8.2 Change SNAPP Passwords

The second page, “Change SNAPP passwords” allows the admin to change both the admin and dispatcher account passwords. On this screen they will be presented with these two options under which there will be three text boxes. The first two text boxes are for the Username and Old Password and the third is for the New Password. Once the admin has filled these sections out, they will need to press Submit in order to make the changes. If at this point any of the sections have missing or incorrect information, a label will appear below that text box telling them this. Otherwise, they will be prompted with a popup dialogue asking them to either cancel or confirm their changes.
4.9 SNAPP Shared Elements

SNAPP Shared Elements is a package we created to promote code sharing between the four different applications. It is a Flutter package that is used across all the apps to standardize various functions and constants. Included in this package, we have put all the ride status and SNAP van constants, keeping data consistent throughout both the apps and the database. We also included most of the database code, with operations such as delete, move, insert, etc. on the different tables in our database being easily callable using the package. Additionally, we included different UI elements that are used in multiple applications. Examples of these in the package include the reorderable card list for use in the co-driver and dispatcher apps, and a variety of shared buttons, decorations, and color schemes used throughout the applications.
5.0 Evaluation

5.1 Overview

We had a number of goals for each application. In order for the student app to be successful, we wanted to make a presentable app which allows a student to request a SNAP ride. Students should receive notifications when the status of their ride changes. They should also have the flexibility to cancel a SNAP ride before it has arrived. We should also restrict the student’s ability to request a ride from or to any commercial locations or any ride outside of a 1-mile radius around campus, unless the student is going to or from Union Station.

In order for the co-driver app to be successful, we wanted the co-driver to be able to view and update the statuses of any rides assigned to them. When they change the status of any ride, both the student and the dispatcher should be made aware of the change, which for the student would be accomplished through a push notification.

In order for the dispatcher app to be successful, we wanted the dispatcher to be able to organize all of the rides both unassigned and assigned. The dispatcher should be able to edit rides and change which van they are assigned to based on their own intuition. They should have the ability to cancel any ride and send the student a custom notification for anything outside of standard SNAP protocol.

In order for the admin app to be successful, we wanted the SNAP administrator to have control over the SNAP service itself. They should be able to mark SNAP as closed, which prevents students from calling in a ride. They should also be able to mark certain vans as under maintenance, as to improve usability for the dispatcher.

In addition to successfully developing these four applications, we hoped to integrate everything into the official WPI SNAP program. We had extensive conversations with IT, the
police department, and SGA on how we can make this happen. In our discussions with IT, we came to the understanding that IT could maintain the applications as long as the documentation is good, and they only have to edit the codebase for operating system updates. SGA agreed to annually fund the costs for a data plan for the co-driver mobile tablets, and the police department would be willing to integrate the applications so long as they work well.

5.2 Security Review

For the security review, we had initially planned to have someone like Craig Wills, Robert Walls or Craig Shue look at our codebase to identify any vulnerabilities that might be exploitable in our apps. However, due to COVID-19, we were unable to do this. We would recommend that a future team gets the codebase officially reviewed from a security aspect by one of these experts.

5.3 Simulation

Since we were unable to run a beta test of SNAPP at WPI due to COVID-19, we decided to conduct a simulation to mock what a normal night at SNAP would be like. We created a script that has student, dispatcher, and codriver roles. This script can be found in Appendix B. Over the course of about one hour, the different actors would take actions like requesting a ride, assigning the ride, marking the ride as completed, and cancelling the ride. We tried to cover the entire scope of what could happen during a real night of operation for the SNAP program. In the script, we each took the role of the student, dispatcher, and/or codriver. Then, we identified a time to run the simulation. Given the constraints from COVID-19, we decided to run the simulation using a mix of emulators and physical devices from our homes.

In regard to the actual simulation, things went quite smoothly. The apps did all of the functions we expected them to, with little to no issues along the way. We did encounter a couple
of bugs. First, the confirm button to cancel a ride can send duplicate cancelled rides to the database and all co-drivers are notified of a cancelled ride. Additionally, we found that there were certain features that we thought we had implemented, that we actually didn’t. For example, we want students to be required to give a reason for cancelling their ride, but never gave them a text box to do this. It became clear to us that most of our issues surrounded cancelling a ride, which is understandable given that it was one of the last features we implemented and therefore have done the least amount of testing on.

Outside of issues we found, our user interfaces were quick, easy, and intuitive to interact with. All of the applications updated in real time when they were supposed to. We believe that some of the user interfaces could use some improvement to their aesthetic, and were lacking in terms of visual design, however, during the development of this project we intentionally aimed to put functionality ahead of visual appeal.

Overall, we believe that the foundation of each of our applications is extremely solid. While we can’t say that they function perfectly 100% of the time without doing tests at scale, they appear to do their base functions without issue in our small-scale test. We hope that a future team or group will expand upon this testing. To ensure sufficient testing for this app to be adopted and used by the WPI community, it would certainly require tests involving 20 or more students making requests over several hours. Additionally, when the app does get put into production and is first starting to be used by SNAP drivers and students, it will need to be monitored closely and require slight adjustments. Getting feedback during this time is crucial, given that it is difficult for us, as the developers, to find every flaw in our system.
5.4 Financial Review

Due to the necessity of both Google Maps API calls for the student app and mobile tablets with some internet connectivity for the co-driver app, a full integration of the SNAPP system has both one-time and annual costs. The one-time costs refer to purchasing tablets for the SNAP department to use with the co-driver application, as it was designed to be used with a tablet by the co-driver in each running SNAP van. With guidance from Lt. Bueno, we decided that four tablets, three of which would have a mobile data plan, is a proper number of tablets for SNAP to ensure minimal downtime if something goes awry. We chose Apple iPads for the tablets, as they are reliable and have a variety of great accessories to ensure safety and security in the SNAP vans. The 4 iPads and their accessories total the one-time costs for implementing SNAPP to an estimated $1,323, including tax.

The annual costs include a Verizon data plan on WPI’s account for 3 of the iPads, and the Google Maps API calls. The Verizon data plan per device costs an estimated $479 annually, totaling to $1,440 for all 3 tablets. The API costs need to be calculated using estimated numbers for the average number of people using the SNAPP student app daily. Based on meetings with SNAP dispatchers, we believe that on the average day SNAP receives about 70 ride requests, with this number being higher on the weekends and much lower outside of the traditional school year. Therefore, we believe a reasonable number of requests per year to be 28,000, or 100 requests 280 days out of the year. For each ride request, we believe that on average, a student app user will call each of the Google Maps APIs 5 times. The Google Maps Places Details API costs $0.017 per call and using Google Maps Autocomplete with the Places API is an additional $0.002 per call. The Geocoding API, which is used for a user’s ‘My Location’, costs $0.005 per call. Finally, there is a baseline $200 credit monthly for all Google Maps API calls, which drastically reduces much of the annual costs for the APIs.
Although we calculated costs using 5 calls to each API per request, APIs such as Geocoding are not likely to be used more than once per ride request; however, we felt it was important to present a higher-end cost calculation as to not request insufficient funding. The annual API costs with the aforementioned estimates total to $960, bringing the total annual costs of the SNAP system to $2,656, including tax. A detailed spreadsheet of the information given above can be found in Appendix C.
6.0 Future Work and Ideas

While we were able to successfully create a functional suite of applications for the SNAP program, we wanted to detail some features that would make the application more stable and robust should it actually be deployed to the WPI community. Here are some core items that need to be implemented before the apps can be confidently released:

1. iOS Notifications - We were able to successfully implement notifications for Android; however, due to additional restrictions put in place by Apple, implementing notifications on iOS requires more steps. First, you must register an apple developer account. While the marketing department gave snapp.wpi@gmail.com access to their apple developer portal, we were not able to implement iOS notifications. Our developer account did not have access to the following page: https://developer.apple.com/account/resources/certificates/list. We recommend reaching out to Damien Arlabosse in order to give our account access to any of the pages mentioned in the section called iOS Integration on the following page: https://pub.dev/packages/firebase_messaging.

2. Firebase Database Rules (https://firebase.google.com/docs/database/security) - In order to further ensure the integrity and security of our database, Firebase offers the ability to write rules to determine who has read and write access to the database. Further investigation into the proper use of these rules is needed to determine their role and implementation in the project.

3. Terms of Use & Privacy Policy - We were not able to meet with the WPI legal department to write up the appropriate legal documentation for the student app. The student app has an unused view which would pull the legal documentation if any were available. Any future team looking to add either a Terms of Use and/or a Privacy Policy
would just need the documents, as all the layout code and database parsing should be complete.

4. Full Unit Test Suite - During our time working on this MQP, we were pretty constrained by resources and time. All four of us were learning Flutter and mobile app development and didn't have time to write extensive unit tests for the applications. While we did end up testing the apps by using them, discovering bugs, and then fixing them, we believe that a full suite of unit tests would help increase confidence in the apps robustness, both for the team and potential stakeholders like IT, marketing and the Police Department.

5. Beta Test for WPI implementation - Before full deployment, there should be a full-scale beta test of the entire system, with multiple trials of the app in a real setting. This will help bring to light hidden bugs in the application, and features that could be improved to increase usability of the apps.

6. Security Review by CS Department Security Expert - We recommend having a full security analysis of the project conducted by someone well versed in the CS department (Craig Wills, Robert Walls and Craig Shue come to mind.) This is another item that will greatly increase the confidence of our stakeholders.

In addition to the important future work discussed above, we have compiled a list of additional features that would enhance the apps, their usability, and usefulness to the SNAP program. We do not believe that these features are critical to the apps themselves, but could be nice starting points for future work on the app.

1. Blacklist - The blacklist was a feature suggested by the Police Department that would help in automating the dispatcher’s role. Essentially, they recommended we implement a filter that will automatically deny a ride if it breaks any of the rules for requesting a SNAP. While we were able to automatically filter out rides outside the one-mile SNAP radius, we were not able to filter out rides that go to/from a commercial location. While
the dispatcher does have the ability to manually reject a ride, at scale the rejection process should be more automated.

2. Union Station - Union Station is one of the edge cases for students to request a ride. It is technically outside the one-mile radius, but students are able to call in a ride from/to Union Station in advance to them needing transportation. Right now, the student is directed to call SNAP if they want transportation to/from Union Station, but it was requested by employees at SNAP to include Union Station requests in the app.

3. Shift Control using Co-Driver App - Some SNAP workers mentioned that they would like to have control of signing in and out of their shift to make tracking their hours and who worked when easier. This could be an interesting feature, but we felt that it was out of the scope of this project.

4. Van Maintenance and Inspection - Before starting a shift, the SNAP driver must manually inspect the SNAP van for any defects and to make sure it is in working condition. They record this information on paper, and the SNAP workers thought it would like to make this process integrated with the co-driver app.

5. Data Analytics - While we will be collecting basic data regarding students and their ride history / tendencies, it could prove beneficial to SNAP to collect more in-depth data for processing and analysis. This data analytics could be used to accurately calculate the highest areas of SNAP activity and could be implemented to better position the vans during busy hours.
7.0 Appendix

7.1 Appendix A

Survey Results

45 responses

<table>
<thead>
<tr>
<th>Summary</th>
<th>Question</th>
<th>Individual</th>
</tr>
</thead>
</table>

**How long do you usually wait for your SNAP van to pick you up?**

45 responses

- 48.7% 0-10 minutes
- 44.4% 10-30 minutes
- 26.7% 30-60 minutes
- 20% Over an hour

**How do you usually wait for your SNAP van to arrive?**

45 responses

- 51.1% I wait inside until I see the van
- 20% I wait outside for the van to arrive
- 26.7% It depends on the weather
- 10% Other

Have you ever missed a SNAP van?
45 responses

Would you rather use a SNAP application (similar in style to Uber/Lyft) to request a ride rather than the calling SNAP?
45 responses

Would you like to receive notifications to your phone about the status of your SNAP ride (On the Way, Arrived, etc.)
41 responses
Would you like to see a history of your previous rides?

41 responses

- 46.3% Yes
- 53.7% No

Is there anything else you’d like to see in the app?

27 responses

- Profiles of current student drivers
- DRIVER RATINGS
- Waiting Queue position should be known
- It will be amazing if you provide it
- Feedback section on app for questions & concerns about rides or timing, etc.
- Location of driver and his contact info
- Track, wait time
- How busy snap is currently
- I think it will be easier just to use a time factor. I think so far it’s effective to use the snap calling system but I think we should have an app where it says when the van has arrived or is on the way.
- ETA/how many people are ahead of me and how many of them are going far
- Use TapRide!
- GPS Tracking
- Location of the van/ ETA
How many people are in front of my call?

ETA with a map both before and during the ride. Ability to request a ride at a later time. Ability to request outside the normal hours as long as you ask a day or two in advance.

Driver ratings:

test by LT. Bueno just to move forward to the next questions.

No

Estimated arrival

How long the snap will take to reach

The GPS of where the SNAP van is.

Estimated time of arrival

Estimated wait times (could be a VERY rough estimate)

Live location of the SNAP van and where I am in the queue to be picked up. Ppl have a problem waiting for snap because they are left in the dark on how long it’ll take for snap to come.

Estimates of arrival before pickup

Who’s driving/non-driving

---

**Why do you believe that a SNAP app would be ineffective?**

3 responses

The effort and resources that need to be put in maintaining the app is overwhelming. Making an app is one thing, making a reliable, zero downtime app is another thing.

Calling works just fine. I’m not opposed to an app or anything, I just think the current system isn’t broken.

I wouldn’t mind an app, I just don’t have much space on my phone and I don’t think calling is a hassle. I do think an app would be great for people who are nervous about phone calls though!
7.2 Appendix B

Simulation Script

K - Ken, Roles = Co-Driver for Van 1 and Student
D - Dimitri, Roles = Dispatcher and Student
N - Nick, Roles = Co-Driver for Van 2 and Student
(S) - Student Role

4:00
K & N: Logs into Student app and Co-Driver app. Once into the Co-Driver app, Ken assigns himself to Van 1 and Nick assigns himself to Van 2.
D: Logs into Student app and Dispatcher app.

4:05
K(S): Makes ride request (From: 50 Franklin St., To: 95 Lincoln St., # Passengers: 2) and confirms the request. The request is rejected due to SNAP not being currently open. Ken acknowledges the rejection.

4:08
D: Opens SNAP from the dispatcher app
4:10
D(S): Makes ride request (From: 119 Belmont St., To: 340 Main St., # Passengers: 1)
K(S): Makes ride request (From: 33 Kendall St., To: 86 Austin St., Additional Info: “We may take
a couple minutes to get to van”, # Passengers: 3)
N(S): Makes ride request (From: 6 Wachusett St., To: 54 Shelby St., # Passengers: 1)

4:12
D: Assigns Nick’s ride to Van 1. Assigns Ken and Dimitri’s ride to Van 2

4:13
K: Changes Nick’s ride status to “Snap Ride Queued”
N: Changes Ken’s ride status to “Snap Ride Queued”

4:16
K: Changes Nick’s ride status to “Snap On The Way”
N: Changes Ken’s ride status to “Snap On The Way”

4:20
K: Changes Nick’s ride status to “Snap Has Arrived”
N: Changes Ken’s ride status to “Snap Has Arrived”

4:22
K: Changes Nick’s ride status to “Rider Picked Up”
N: Changes Ken’s ride status to “Rider Picked Up”, and Dimitri’s status to “SNAP On The Way”
4:27
K: Changes Nick’s ride status to “SNAP Ride Complete”
N: Changes Dimitri’s ride status to “Snap Has Arrived”
N(S): Returns to home screen

4:30
N: Changes Dimitri’s ride status to “Rider Picked Up”
N(S): Makes ride request (From: 94 Eastern Ave., To: 73 Lancaster St., # Passengers: 1)

4:33
N: Changes Ken’s status to “SNAP Ride Complete”
D: Assigns Nick’s ride to Van 1
K(S): Returns to home screen

4:35
N: Changes Dimitri’s status to “SNAP Ride Complete”
N(S): Cancels ride
D(S): Returns to home screen
K: Confirms popup in regard to Nick’s cancelled ride
D: Delete Nick's ride from cancelled column

4:37
K(S): Makes ride request (From: Leitrim’s Pub, To: Morgan Hall, # Passengers: 4)
N(S): Makes ride request (From: 40 Belmont St., To: 28 Franklin St., # Passengers: 3)
D(S): Makes ride request (From: 68 Cedar St., To: 69 West St., # Passengers: 1)
4:38
D: Reject Ken’s ride (Reason: SNAP does not pick up from bars)
D: Assign Nick’s ride to Van 1, and Dimitri’s ride to Van 2
K(S): Confirm rejection

4:40
K: Changes Nick’s ride status to “SNAP On The Way”
N: Changes Dimitri’s ride status to “SNAP On The Way”
K(S): Call Dimitri and ask for a ride (From: 27 Windsor St., To: Faraday Hall., # Passengers: 5)
D: Dimitri manually request’s ride given ride information
D: Enter Manual Ride request with given information (From: 27 Windsor St., To: Faraday Hall., # Passengers: 5) and assign to Van 1

4:42
N(S): Call Dimitri and change From address to 40 Bowdoin St.
D: Manually changes address and confirms changes

4:45
N: Changes Dimitri’s ride status to “SNAP Has Arrived”
K: Changes Nick’s ride status to “SNAP Has Arrived”

4:48
K: Changes Nick’s ride status to “Rider Picked Up”

4:50
N: Changes Dimitri’s ride status to “SNAP Rider No Show”
4:52
D(S): Confirm popup regarding “No Show” status

K: Changes Nick’s ride status to “SNAP Ride Completed”

N(S): Return to home page

K: Changes Ken’s ride status to “SNAP On The Way”

4:55

K: Changes Nick’s ride status to “Rider Picked Up”

5:00

K: Changes Nick’s ride status to “SNAP Ride Complete”

D: Close SNAP

All: Logout of all accounts
7.3 Appendix C

Simulation Script

<table>
<thead>
<tr>
<th>One-Time Costs</th>
<th>Purchased From</th>
<th>Units</th>
<th>Price/Unit</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>4G iPad</strong></td>
<td>IT Department</td>
<td>3</td>
<td>$359.99</td>
<td>$1,079.73</td>
</tr>
<tr>
<td>iPad Credit</td>
<td></td>
<td>3</td>
<td>-$100</td>
<td>-$300</td>
</tr>
<tr>
<td><strong>Wi-Fi iPad</strong></td>
<td>IT Department</td>
<td>1</td>
<td>$200.99</td>
<td>$200.99</td>
</tr>
<tr>
<td>iPad Case</td>
<td>Amazon</td>
<td>4</td>
<td>$22.85</td>
<td>$91.40</td>
</tr>
<tr>
<td>Car Charger</td>
<td>Amazon</td>
<td>4</td>
<td>$11.99</td>
<td>$47.96</td>
</tr>
<tr>
<td>Play Store Fee</td>
<td>Google</td>
<td>1</td>
<td>$25.00</td>
<td>$25.00</td>
</tr>
<tr>
<td>MA Sales Tax</td>
<td></td>
<td></td>
<td></td>
<td>$78</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td>$1,323</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Annual Costs</th>
<th>Purchased From</th>
<th>Units</th>
<th>Price/Unit</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Google Maps API</strong></td>
<td>Google</td>
<td>140,000</td>
<td>$0.017</td>
<td>$2,380</td>
</tr>
<tr>
<td><strong>Places API</strong></td>
<td>Google</td>
<td>140,000</td>
<td>$0.002</td>
<td>$280</td>
</tr>
<tr>
<td><strong>AutoComplete</strong></td>
<td>Google</td>
<td>140,000</td>
<td>$0.005</td>
<td>$700</td>
</tr>
<tr>
<td><strong>Geocoding API</strong></td>
<td>Google</td>
<td>140,000</td>
<td>$0.005</td>
<td>$700</td>
</tr>
<tr>
<td><strong>Google Maps Monthly</strong></td>
<td>Google Maps API</td>
<td>12</td>
<td>-$200</td>
<td>-$2,400</td>
</tr>
<tr>
<td><strong>Verizon Data Plan</strong></td>
<td>IT Department</td>
<td>36</td>
<td>$38.99</td>
<td>$1,422</td>
</tr>
<tr>
<td><strong>App Store Fee</strong></td>
<td>Apple</td>
<td>1</td>
<td>$100</td>
<td>$100</td>
</tr>
<tr>
<td><strong>MA Sales Tax</strong></td>
<td></td>
<td></td>
<td></td>
<td>$158</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td>$2,656</td>
</tr>
</tbody>
</table>