

## **Taxi Sharing Application**

Andrew Dodge – 100938015

An Honours Project submitted to  
The School of Computer Science in partial fulfillment of  
Bachelor of Computer Science Honours with  
Software Engineering Stream

Carleton University  
April 22<sup>nd</sup>, 2019

Approved by:

Honours Project Supervisor: Dr. Doron Nussbaum

School of Computer Science Undergraduate Advisor: Edina Storfer

## **Abstract**

The goal of this project was to research and design taxi sharing applications in order to develop an application which would form the basis of a future project. The project was to be evolved into an autonomous driver application thus the research was focus on what passenger information was needed as the driver would become obsolete in the future. The approach was to gather information on taxi sharing in order to create a design which could present this information in an organized manner. Following research, the application was to be implemented and tested, while the application's design actively displays all the information to the user due to time constraint the scope of the project was downgraded from a taxi sharing application to a simple taxi application.

## **Acknowledgments**

The author of this would like to acknowledge the following sources for their contributions to the project's completion. Google services, Firebase Geofire, Uber, Lyft, Bump Technologies, Waymo One, and Stack Overflow for their research contributions and respective code contributions to the final product. Furthermore, the author would like to acknowledge the following people for their support throughout the project. Dr. Doron Nussbaum, the project's supervisor for the original idea, and implementation assistance throughout the project. Mohamed Gahelrasoul, the partner for this project for assistance in development. Hana Osman, for her assistance in editing the report and never-ending love and mental support throughout the duration of the project, and finally all the friend's and family who offered their love and support across this difficult semester.

## Table of Contents

Abstract.....	i
Acknowledgements.....	ii
Table of Contents.....	iii
List of Figures.....	iv
Chapter 1: Introduction .....	1
Problem Definition.....	1
Motivation.....	1
Contributions.....	2
Chapter 2: Background .....	4
Definitions.....	4
Development Software.....	4
Previous Work.....	7
Chapter 3: Main Contribution .....	13
Main Objectives.....	13
Design Decisions.....	17
Chapter 4: Implementation .....	21
Setup.....	21
Implementaion and Error Discussion.....	24
Chapter 5: Conclusion.....	39
References.....	41
Appendix.....	44
Balsamiq Mockups.....	43
Android Studio Design.....	47
Final Application Design.....	51

## List of Figures

Figure 1: Ridesharing Market Share.....	9
Figure 2: Communication Diagram.....	14
Figure 3: Cost Calculation Equation.....	16
Figure 4: Uber Pre-Ride Screen.....	17
Figure 5: Lyft Opening Page.....	18
Figure 6: Balsamiq Mockups of Pre-Match Screen.....	19
Figure 7: Comparison between Balsamiq and Android Studio.....	25
Figure 8: On Click Function Example.....	27
Figure 9: On Click Listener Example.....	28
Figure 10: Map to store information to Firebase.....	29
Figure 11: Error Solution due to Account Creation.....	30
Figure 12: Maps Activity and XML code Example.....	32
Figure 13: Displaying a Match with a Driver.....	34
Figure 14: Ride Over Prompt.....	37

## Chapter 1: Introduction

Taxi sharing is a modern method of transportation that establishes a cross between bus and taxi travel. The central idea of which being a taxi that can pick up multiple people, either at the same location or multiple locations along the way, and then drop them off at their desired location (preferably in the same general area). To give some additional information, taxi sharing is more beneficial than traditional taxis as each patron pays a cheaper fee if the ride is shared with other passengers. Furthermore, the cab driver is paid by each individual passenger meaning that they end the day with a greater income.

### *Problem Definition*

The honours project for COMP4905, which was completed this semester, was to design and develop a taxi sharing system for the Android device. The taxi sharing system consists of three components: a passenger application, a taxi driven application, and a back-end. The focus of the project was on research and design, to determine what applications already existed within this concept and to design an application that functions on multiple different device sizes for a wide range of individuals. A back-end communication system was implemented; however, there was minimal attention paid to it as it was not the focus of the project. This project was done in tandem with another student, Mohamed Gahelrasoul, with an even division of labour. His task was to implement the driver application of the system as well as to implement the back-end communications. The other half of the project, which is the primary focus of this report, is the passenger application component of the system. It was decided at the beginning of development that the driver would not have a large variety of options associated with them, which will be further discussed later in the report. This was the critical reasoning behind the division of labour.

### *Motivation*

The motivation for this project stems from three different sources. The first source was to develop a taxi sharing system which would form the base for a future project. This system is to be

further expanded into an autonomous vehicle taxi sharing application, where driverless vehicles will be accessible to passengers to take them to and from their destinations for a fee. Especially with the introduction of self-driving cars having become more prevalent in today's society. To have an autonomous vehicle that could transport small groups of people throughout cities and to their destinations would be the next logical step once the technology becomes realized. This source was the reasoning behind making the driver side minimal, as it will eventually be eliminated entirely. To continue, the second source of motivation stems directly from the autonomous system spoken of previously. If it is possible to evolve this system to the point of automation then this will cause a decrease of vehicles on the roads. This reasoning is why this technology is so important to research. This concept would also aid to alleviate the number of vehicles on the roads, since every person who decides to share a ride with someone would ensure one less vehicle out on the road. This is crucial in modern society, where pollution from transportation has become significant and pervasive. Reducing the number of vehicles on the road would aid in decreasing the amount of pollution emitted in cities that possess this system. Further motivation for this project stems from an unrelated source to the previous two points. It is an intrinsic desire to expand on knowledge of application development and design and contribute to this development in a meaningful way. This will aid in creating a solid foundation of skills to be taken into future career paths. To continue, as stated above, the purpose of this project is to create an application in the present that will then be evolved in a future project. The problem being addressed by this project may not appear to be globally important to the present population. However, it does not make it any less important to research and begin implementation now. Especially due to the fact that there are a number of taxi sharing applications in circulation already. There is hope that the future evolution of this project will set it above the standards of current applications.

### *Contributions*

The work completed during this project was a majority of the implementation of a taxi sharing system. The project consisted of two separate applications that share a back-end database and communication system. The passenger application was the focus of this project, and consisted of two components, research and implementation. The research focused on how taxi sharing works

and how to implement a working system. The ultimate goal of the project was to determine what information was required in order to match passengers with taxis. The implementation aspect focused on the design of the application itself. Firstly, to get it to be visually appealing when using it and secondly, to ensure it can function smoothly on Android devices. The manner in which the design was implemented makes it so that the application will scale based on which Android device the application is installed on. This is especially important when it comes to the text visible on screen. The design allows for the application's text to be able to mirror the user's text-size preference. If a user has their text default set to "large" then the application will use this decision on load. With regards to the application itself, it is fully functional to an extent, able to request a ride in order to be matched with a driver. This information is displayed on the screens for all users to inspect. More detail will be given on the interactions and abilities of the users further into the report. The driver side application implemented by Mr. Gahelrasoul as mentioned above will be further expanded upon in his report, as well as the communication system.

The report follows a basic layout. Chapter One was an introduction, which included the motivation of the project as well as a small project description. Chapter Two will discuss the background of the system. This will include a description of the software used to complete the project as well as a discussion of the work which currently exists in this field and how they were used as inspiration for this application. Chapter Three will further discuss the main goal of the project, delving into an in-depth discussion of the problem attempting to be implemented and the objectives of the application itself. Chapter Four will discuss in detail the process of the application's development such as how the application was completed from start to finish, the testing which was conducted during development between Mr. Gahelrasoul's app and this application. As well as a discussion of the errors which were discovered during development and how they impacted the end result of the system. The final chapter will be a conclusion that will briefly summarize the information that was started throughout the report. Following the written portion of the report there will be a references section which will include a bibliography of all the research used during the project. Finally, there will be an appendix section which includes the design of the application as well as any other referenced images mentioned throughout the report.



## Chapter 2: Background

This chapter reviews the systems that attempt to provide taxi sharing application. It also provides preliminary information or systems used in this work. As was discussed above, this project was to design a taxi sharing application using Android Studio that would function across a number of Android devices. This chapter will define the needed information that is required to understand the concepts of the project. It will begin with a basic definition of the purpose of the application, followed by a description of the software that was used. Finally, this section will discuss the applications similar to the one designed in the project and how they were either used as research material or inspiration towards the completed project.

### *Definitions*

Taxi sharing is defined as a service that combines the simplicity of taking a taxi incorporated with the passenger-sharing ability of a bus. The main difference between this service and a regular cab is that a passenger will be matched with strangers who have similar destinations, and these additional passengers will be picked up along the route. This is opposed to a taxi, which involves the typical privacy that standard taxis provide. Furthermore, taxi sharing monetarily incentivises riders to share the ride with strangers, due to the collective fee that contributes to reduced individual fees. With respect to this, since each passenger must pay a fee to use the service, this means the driver would yield a greater income if there are a higher number of passengers. With this in mind, this was the goal of the project, to design an application that would achieve full taxi sharing capabilities. From matching similar users together, to displaying their trip on the screen, to completing the ride and displaying the proper information to the user post trip.

### *Development Software*

Before discussing the components of the application and how the system came together it is important to touch upon the program used to design the project. The application was designed using a program called Balsamiq Mockups. This program is a prototyping tool where a user can

create low-fidelity prototypes, or digital sketches, of the project being designed.<sup>1</sup> This gives the developer the ability to produce ideas, facilitate discussion, and expand upon understanding before any code is written. An important aspect of Balsamiq that sets it apart from other prototyping tools is the ability to link the digital sketches together as if it were a real application. With this feature, it is possible to mimic application flow. For example, if clicking a button opens the user's profile then it is possible to mimic that interaction in the mock-up. Once the entire design has been implemented in Balsamiq, it becomes quite simple to modify elements and view how interactions function before the development begins. Development also becomes greatly simplified with the addition of examples to help base each stage of design on. All Balsamiq Mockups for this project are visible within the Appendix section at the bottom of this report, as well as a description of how the images are linked together.

To continue, the application was implemented using the program Android Studio by Google. It is the official integrated development environment, IDE for short, for application development for Android devices. Essentially, it is a program designed specifically to create applications for Android devices, with the ability to both design the user interface and create back-end functionality. Android Studio is based off the IntelliJ IDE and incorporates the shortcuts used within that development tool to aid with programming.<sup>2</sup> There are two main languages a user can program in within Android Studio, users can choose between Java and Kotlin. Both with unique benefits and complications. For this project the language of choice was Java, for ease of implementation and back-end simplicity. With regards to development, Android Studio divides the workload into two distinct parts. The design of the screens (or activities as they are called in Android Studio), which is done in an extensible markup language, xml, file. It is possible to add elements, such as text, buttons, and images, by either writing the xml code straight into the source code, or by visually designing the application on the blueprint screen. When using the blueprint screen, it is possible to drag the elements needed onto the given activity. This is the visual method of creating each page of the application. Once an element has been added to the activity using this method, Android Studio will automatically add the xml source code to the file. This method of

---

<sup>1</sup> Balsamiq Studios. (2019). Balsamiq Mockups 3 Application Overview - Balsamiq for Desktop Documentation | Balsamiq. Retrieved March 30, 2019, from <https://balsamiq.com/wireframes/desktop/docs/overview/>

<sup>2</sup> Walter, D. (2018, October). What is Android Studio? - Definition from WhatIs.com. Retrieved March 30, 2019, from <https://searchmobilecomputing.techtarget.com/definition/Android-Studio>

design is reminiscent of the development environment Unity, where visual elements are added to the screen in this manner and the back-end of these elements are programmed afterwards in a separate file.<sup>3</sup> The separate file in this case being a Java file per activity in the application. Each respective Java file contains all the back-end functionality that the activity needs in order to accomplish the tasks assigned to it. Be it a button, referencing user information, or displaying specific information on the screen, Android Studio can also be linked with multiple different resources to aid in the production of the application. For example, Github can be linked to the Android Studio application for ease of backing up the user's data. Furthermore, since Android Studio uses a Gradle-based system for compilation, a user can connect directly to a Github project using a compilation command. This command will direct the compiler to include the source code stored in Github and add said source base to the amalgamation of files that make up the core of the application. This Gradle-based system also allows the user to specify the required permissions and Google Play Services the application will require for installation.<sup>4</sup> Finally, with regards to testing the application using Android Studio, the program has two built in functionalities where a developer can view the work they have completed thus far. Both methods will involve compiling the application and creating an Android Package, APK, that will then be installed on a device. This is where the difference lies, developers can choose to install the application on a virtual Android device, or onto a physical Android device. The virtual machines are built into Android Studio and can be downloaded at any time during development, (they are helpful tools as a developer can then test how the application will react on different Android devices), or different versions of the operating system. Furthermore, it can be used to quickly test features of the application as the developer does not have to change environments. The virtual device is stored on the same computer as the development environment. Comparatively, the developer can use a physical Android device to run the application. This is easily done by plugging an Android device into the computer at the time of compilation and selecting it as the desired location. While this method is slower than using a virtual device, it allows the developer to test how the application will function in the real world, especially if the application has network components. With the application on a physical phone

---

<sup>3</sup> Technologies, U. (2019, January). Learning the interface. Retrieved April 18, 2019, from <https://docs.unity3d.com/Manual/LearningtheInterface.html>

<sup>4</sup> Walter, D. (2018, October). What is Android Studio? - Definition from WhatIs.com. Retrieved March 30, 2019, from <https://searchmobilecomputing.techtarget.com/definition/Android-Studio>

the network components can be tested with the user's data or wireless internet. Android Studio visual examples are listed in the Appendix section of the report.

Speaking of network capabilities, this application has an online database and communication system that uses Firebase on the Google Cloud Platform.<sup>5</sup> Firebase is a Backend-as-a-Service, a model for providing web and mobile app developers to link applications to back-end cloud storage. This project's application uses Firebase for a wide array of features. The primary role that Firebase fulfills is as a database. All user information when an account is created is stored in the Firebase database and is referenced from there when called upon by the back-end code. User authentication is also controlled by Firebase, a user cannot login without a valid account stored on the database. Most importantly Firebase acts as the communication system for the application. Firebase acts a server for the driver and passenger applications to communicate between each other. While the information sharing is implemented as back-end code the communication system is done through Firebase. The majority of the back-end was implemented by Mr. Gahelrasoul, further detail will be said on this subject within his report.

### *Previous Work*

Following the description of software used throughout the project, previous work is reviewed. There are numerous applications recognized in the field of taxi sharing. Such as, Uber, Lyft, Cabify, eCab, Blueline Taxi, and the Ottawa Taxi App. However, only a select few were used as examples or research tools when designing the application. The applications that became a research focus included: Uber, Lyft, Via, and Waymo One. These applications were all used as research tools at one point in development, each for their own reasons. Most of the research was used to determine how taxi sharing works at a fundamental level. For example, how the user profiles are set up, or what components are needed to determine a proper match, as well as what is a reasonable cost for a typical ride using these applications. All applications used have a taxi sharing component to them. While Uber and Lyft are apps in their own regard, they have since implemented a taxi sharing element similar to the goal of this project's application.

---

<sup>5</sup> Google. (2019). Firebase Overview. Retrieved March 30, 2019, from <https://firebase.google.com/>

To begin, the most popular modern application will be discussed. Uber currently holds the title as the most popular taxi application on the market. It is available in sixty-five countries as well as six hundred cities worldwide. There are approximately fifteen million trips completed each day using this application.<sup>6</sup> Therefore, it was a prime research tool when designing this project. However, Uber is typically just a taxi service that passengers can use to request a ride for themselves, or for small groups of passengers, which will take them to their requested destination.<sup>7</sup> The application being designed in this project is specifically a taxi sharing system. Meaning, Uber does not entirely fit the description of the project. This being said, during research it was found that Uber has a taxi sharing component available to their application. It is called UberPool, which is their variation of a taxi sharing application. It functions the way one would expect a taxi sharing application to function. A user can open the app and enter their destination and size of their group, (maximum 2 people), before requesting a ride.<sup>8</sup> The system would then match them with a driver and send them to a pick-up location. During this waiting period other users are matched to the car until the vehicle is full. The application will also display the cost of the ride before pick-up occurs as well as the estimated travel time, so that the user knows the amount the ride will cost before getting in the Uber.<sup>9</sup> During the ride the driver will either continuously pick up passengers along a similar route or drive certain passengers straight to their destination if there are no further customers. Once the destination is reached, the appropriate passenger exits the vehicle and is automatically billed via the payment method listed on the application.<sup>8</sup> This ride system greatly inspired the system put in place by the project application. The only element left out was displaying the estimated duration. The research gained from this application, other than the overall system, aided with back-end development. Analyzing Uber's system aided in determining the amount of luggage a user would be allowed to bring with them. It also aided in determining how to drop off riders, as well as determining how many passengers are allowed per group. It was determined from

---

<sup>6</sup> Iqbal, M. (2019, February 27). Uber Revenue and Usage Statistics (2018). Retrieved March 30, 2019, from <http://www.businessofapps.com/data/uber-statistics/>

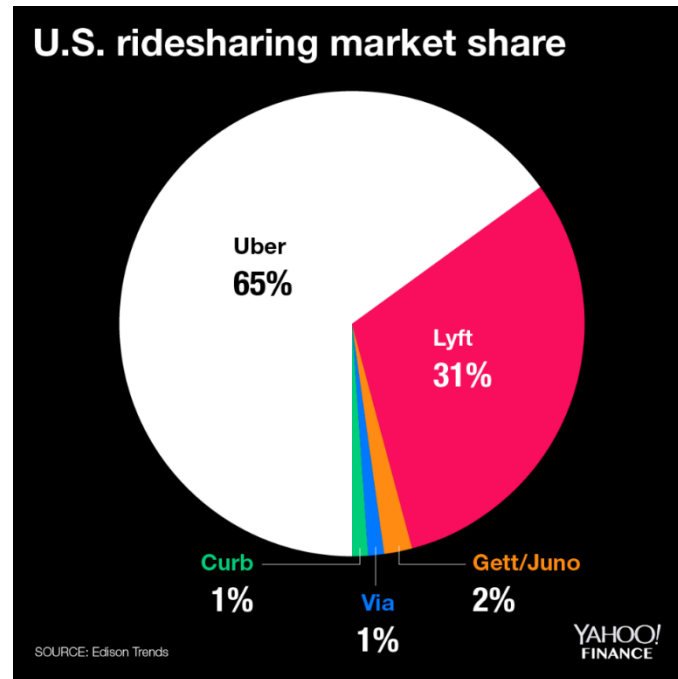
<sup>7</sup> Uber. (2019). About Uber - Our Story - Vision for Our Future. Retrieved March 30, 2019, from <https://www.uber.com/en-CA/about/>

<sup>8</sup> Uber. (2019). UberPool vs. UberX - How Does UberPool Work? Retrieved March 30, 2019, from [https://www.uber.com/en-CA/ride/uberpool/?state=G7iYn0jH75cgczSfSkAjBQ4xyOieMG7m\\_\\_FlamxG93o=&\\_csid=gAM22lbigQKqssDDOuFa4w#\\_](https://www.uber.com/en-CA/ride/uberpool/?state=G7iYn0jH75cgczSfSkAjBQ4xyOieMG7m__FlamxG93o=&_csid=gAM22lbigQKqssDDOuFa4w#_)

<sup>9</sup> Uber. (2017). UberPOOLSharing is saving. Retrieved March 30, 2019, from <https://www.uber.com/info/uberpool-nj/>

comparing the differences between Uber and UberPool that while UberPool takes slightly longer to complete a trip, it is always up to 65% cheaper than taking a regular Uber.<sup>10</sup>

To continue, Lyft was also used as a research tool for this project. Lyft is a popular North American taxi application. It is currently being utilized in 350 US cities as well as in certain cities of Ontario, Canada. While Uber dominates the market with sixty-five percent of all application-based taxis, Lyft holds onto a significant thirty-one percent of that market.<sup>11</sup>



**Figure 1: Ridesharing Market Shares<sup>12</sup>**

Therefore, they were an important source of research for this project. Furthermore, while researching the company it was found that Lyft has also introduced their own variation on taxi sharing, called Lyft Line.<sup>13</sup> Lyft Line essentially functions similarly to UberPool, however, the main difference found here was that while Uber will slowly match users with other passengers along the way (picking up and dropping off passengers as the ride progresses), Lyft will attempt

<sup>10</sup> Uber. (2017). UberPOOLSharing is saving. Retrieved March 30, 2019, from <https://www.uber.com/info/uberpool-nj/>

<sup>11</sup> Iqbal, M. (2019, March 25). Lyft Revenue and Usage Statistics (2019). Retrieved March 30, 2019, from <http://www.businessofapps.com/data/lyft-statistics/>

<sup>12</sup> Guzman, Z. (2019, March 27). Why Lyft's IPO may be more attractive than Uber's. Retrieved April 19, 2019, from <https://finance.yahoo.com/news/lyfts-ipo-may-attractive-ubers-193328062.html>

<sup>13</sup> RideShareApps. (2015, August 24). Lyft Line - What Is It And How Does It Work? Retrieved March 30, 2019, from <https://rideshareapps.com/lyft-line/>

to match the group of passengers together at the beginning of the ride and not pick up additional passengers once the drive has begun. However, if a passenger requests a ride while a trip is ongoing, and the pick-up and drop-off is optimal, then the Lyft driver will be instructed to pick them up along the way.<sup>14</sup> This feature was found to be similar to the idea that this project was trying to achieve and thus the project was implemented in a similar fashion. Similarities were also observed between the two systems. For example, there were similarities in how they displayed the approximate cost and duration of the ride before the trip begun. Another example was how the application determined the optimal route based on all the individuals' desired destinations. This means that while a user may be picked up first they may not necessarily be dropped off first, they will be dropped off in the most optimal order possible. These two apps heavily influenced the back-end system that the project was aiming for, therefore there are distinct similarities with regards to the order in which actions are carried out during a match. With regards to user interface, the design of this project was partially influenced by these apps, however many elements are original. Both Uber and Lyft have their own distinct way of calculating driver payout as well as displaying information to the user.<sup>15</sup> Both applications have a visible road map as the background with their assortment of elements organized on top of the map. This is the only similarity with regards to the user interface. The project's application once logged into, has a map as the background with elements organized on top of it. However, the design of the application is unique and not based off one of these popular applications.

Moving away from the popular taxi sharing options, another application that influenced the project will be discussed. Via is a rather unknown alternative to Uber and Lyft. It currently makes up one percent<sup>16</sup> of taxi sharing application services and is still only available in three cities: Chicago, Washington D.C., and New York City. but this was not a deterrent to using it as a research aspect. Via is not technically a taxi sharing application, but rather a shuttle service.<sup>17</sup> Users with the application will see specific pick-up and drop-off locations and will be able to ride the shuttle

---

<sup>14</sup> Lyft, Inc. (2018). About Shared rides. Retrieved March 30, 2019, from <https://help.lyft.com/hc/en-ca/articles/115013078848-About-Shared-rides>

<sup>15</sup> UberLyftDriver. (2017, October 20). Changes to Lyft Line Payment. Retrieved March 30, 2019, from <https://www.ridesharingforum.com/t/changes-to-lyft-line-payment/131>

<sup>16</sup> Iqbal, M. (2019, March 25). Lyft Revenue and Usage Statistics (2019). Retrieved March 30, 2019, from <http://www.businessofapps.com/data/lyft-statistics/>

<sup>17</sup> UberLyftDriver. (2018, March 04). Via, the Alternative to UberPOOL and Lyft Line. Retrieved March 30, 2019, from <https://www.ridesharingforum.com/t/via-the-alternative-to-uberpool-and-lyft-line/773>

for a small fee. This is similar to a city bus but without the uncertainty that the bus will never arrive. Furthermore, since the application is still relatively small, there is a large amount of customer support available to users, as the company does not have to deal with millions of requests daily. The main aspect that was taken from this service and integrated into the project was they way this application deals with pick-up. All passengers arrive at the same location and board the shuttle (or taxi, in terms of the project), at the same time. The goal of the project was to achieve this functionality; not to have to make multiple pickup stops along the route. Therefore, this element was integrated during development. To continue, since Via is closer to a shuttle service than a taxi service, it typically has the same day-to-day customers. It is a popular method for students to get to school or employees to get to work everyday. For this reason, Via attempts to develop a connection between riders and drivers, as the same driver will typically transport the same passengers each day. Via attempts to create a more enjoyable commute for all individuals involved, instead of the commute just having to be a necessary component of someone's day.<sup>18</sup> Finally, Via offers incentives to users on top of the transportation. This feature is an aspect lacking from the applications of the competitors. The incentives within their service has to do with the cost of each ride. As drivers transport more individuals over a consistent time period they begin to earn a higher income than when they first started in the employ. Furthermore, as a passenger there is a similar system in place. The more a user rides with Via the smaller the regular fare becomes. This became a noticeable feature that was under consideration to be applied to the project.

To conclude this section, there is one additional application that must be discussed. As this project's motivation is for it to evolve into an autonomous taxi service it would be unjust to ignore the autonomous taxi service currently in development. Waymo One began as Google's self-driving car project in 2009 and has since evolved into a full autonomous taxi service. The application is currently only in deployment in the Metro Phoenix area of Phoenix, Arizona with hopes that it will be rolled out further in the future.<sup>19</sup> The application works similarly to other taxi applications where a user enters their pick-up and drop-off locations before requesting a ride. The application will then display the given fee for the ride before the user confirms their ride, in order to make sure the user is comfortable paying the given amount. The vehicle will then arrive where specified and

---

<sup>18</sup> UberLyftDriver. (2018, March 04). Via, the Alternative to UberPOOL and Lyft Line. Retrieved March 30, 2019, from <https://www.ridesharingforum.com/t/via-the-alternative-to-uberpool-and-lyft-line/773>

<sup>19</sup> Waymo. (2018). Waymo – Waymo. Retrieved March 30, 2019, from <https://waymo.com/>



carry passengers to their requested destination. All of this is accomplished without a driver in the vehicle. Passengers of this service always know what to expect since every ride is identical. The vehicles are always the same and carry the same branding, and if anything were deemed unsatisfactory during the ride or if users needed to get in contact with support, there is a built-in help system on the application. Users simply need to open the support tab on the application, and they will be able to discuss the issues with their ride to a live representative.<sup>18</sup> While no features were implemented or inspired by aspects of this application it was still an interesting element found during research that deserved mentioning given the scope of this project.

To end this section, it can be concluded that there are numerous options available when it comes to designing the application for this project. A number of which had prominent influence during the design of the application, and a number of which did not have any influence whatsoever. However, regardless of the influence by outside applications, the project which will be discussed in the following two sections is entirely unique. As is the case with most elements in the world, each new age of technology will always incorporate influences from the past. However, while the back-end and flow of the application may be reminiscent of others, the design and feel of this application will be solely its own.

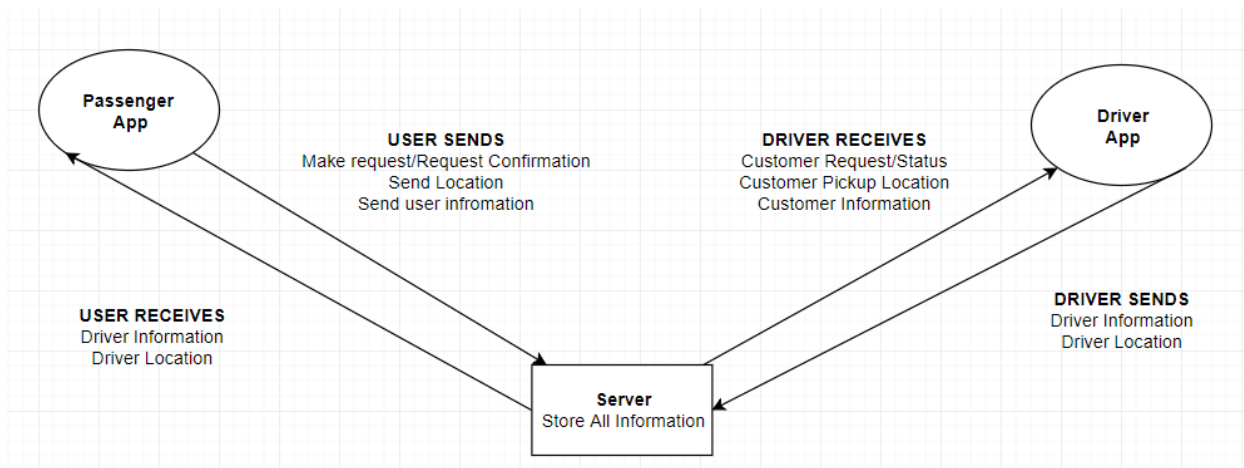
### **Chapter 3: Main Contribution and Design Decisions**

This chapter will further discuss the main goal of the project, delving into an in-depth discussion of the problem attempting to be implemented and the objectives of the application itself. This section will describe the initial plan for the system; however, it will not be an exact description of the completed software. Chapter four will discuss the implementation of the system that was designed for this project, while this chapter will discuss the initial idea and thought process leading up to the final product. Any elements that were not fully implemented will be explained later in Chapter four. Following the description of the project's objectives, this section will also discuss the design of the application. This discussion will include the research that was found to support the final design choices, influences as to why certain elements were chosen, and the design options that were determined through low fidelity prototype implementation. The goal of this section is to lay out the details of the system and the decisions leading up to the design of the application in a more complete and understandable manner. Following this section will be a detailed description of how the system was actually implemented.

#### *Main Objectives*

The goal of the project was to develop a taxi sharing application that was capable of matching users and displaying the proper information. Part of the design of the application was to navigate travel from a specific location, using the Ottawa International Airport as the pick-up area; this decision was made early on in the development stage. One of the objectives of the project was to focus on design and research and less on back-end implementation. Having a general pick-up area (Ottawa International Airport) aided in this simplicity. Another objective of the application was to mirror the functionality of the apps currently on the market, such as Uber, Lyft, and Waymo. There were no exact specifications when it came to the functionality of the application. Just minimal requirements that are typical to every taxi sharing application. The main focus of the project was research-based. This research included general application designs, essential elements that applications need, how to achieve functionality with these given elements, and how to ensure everything works together as seamlessly as possible. Before discussing these research points some attention will be given to the requirements.

The main requirement was to have an application system that consisted of three elements. There would be two separate applications, a passenger system and a driver system, which would be connected by a back-end communication system. The requirement was to have these three elements communicating seamlessly (i.e. to have the passenger system send the proper information to the driver). This information includes when a customer is requesting a ride, who the customer is, where the customer is, and where the customer requests to be dropped off. This information is saved in the server before being sent to the driver. The driver system receives it and then the customer is prompted with their matched driver's information. Furthermore, since the application is supposed to be a taxi sharing service, the idea that multiple passengers could be matched together was employed. Once a request is made and the server matches a user to a driver, the application will then display all the information it had collected on the screen in distinct areas. This was the main objective of the application. The minor details which will be discussed shortly were all research-based. The main requirement was to create and implement the triangle architecture that is quite reminiscent of a mediator design pattern. There are independent elements in this design pattern. In this case, the independent elements are the applications, which only communicate through a middle element (i.e. the server). The driver and passenger applications do not know about the other, they are only aware of the information they receive from the server. This was the main goal set out to be implemented. The rudimentary design of this system is included below:



**Figure 2:** Communication Diagram

Figure two depicts the layout of the system required for the design. To clarify, the requirements tasked for this project include implementing the passenger end of this system, as well

as minimal back-end in the server system. The driver application and the majority of the back-end was tasked for Mr. Gahelrasoul's project and thus will be discussed more in depth throughout his report.

With the main requirements and objectives of the system established, the minor objectives will now be discussed. As this project was being done in tandem with another, these requirements differ from the major system being designed and are more focused on an individual application basis. The passenger application was the focus of this project and consisted of two components; research and implementation. The research was two-fold, there was an element focused on how taxi sharing works, such as to determine what information was required in order to match passengers with taxis, and secondly, there was research based on how to implement a working system. Two of the applications which were mentioned in the background were the main focus of this research. Uber and Lyft both influenced the design of this application in multiple ways. Before design is mentioned, some attention must be paid to the requirement specified initially.

For the passenger application, research had to be completed in order to determine what was needed for a taxi sharing system. This was done through the process of downloading the typical taxi sharing apps and exploring their user creation screens. Therefore, credit must be given to Uber and Lyft for the inspiration towards the user fields. The elements under question when performing this research were: what information the user must provide initially, what must they provide before every ride, how is cost calculated for a ride, and how is the information laid out during all stages.<sup>20</sup> Both apps yielded similar results through this process. With regards to information that is provided on the sign-up page, each user had to provide their name, phone number, some form of payment (credit card or debit card), and some form of login identification (username and password). Furthermore, users had the option of providing a profile picture.<sup>21</sup> These elements were all incorporated into the design of the passenger application and will be discussed in detail shortly. To continue, a portion of the research was dedicated to discovering what elements contribute to user-matching within a taxi sharing service. Both of the applications (Uber and Lyft) keep their matching algorithms private. Therefore, the elements chosen for this

---

<sup>20</sup> Pratap, M. (2018, October 26). How to Build an App like Uber? (Complete Guide) - Uber Clone. Retrieved January 30, 2019, from <https://www.engineerbabu.com/blog/how-to-build-an-app-like-uber/>

<sup>21</sup> Lyft, Inc. (2019). Ride with Lyft – Friendly drivers and serious safety. Retrieved March 30, 2019, from <https://www.lyft.com/rider>

project were based on deductive reasoning as opposed to specific details from the literature. However, certain elements available to the public were found to be similar between these two systems. For example, both applications take into consideration the number of people within the group placing the ride request. Uber limits the number of people in a group (using UberPool) to two. This allows for additional people to be picked up during the ride.<sup>22</sup> Furthermore, since taxi sharing is all about convenient distance, it was also found to be a factor in the matching research. The application will only place a user with the closest driver, as opposed to one halfway across the city. This ensures less of a wait time and the application can then receive more requests throughout the day.<sup>23</sup> The rest of the research on these applications pertains to the design. The final element to mention with regards to the logistics research is the cost calculation. Since taxi sharing has a different cost calculation than typical taxi applications, the formula used required some thought. What was found demonstrated that typical taxi services have a base fee, a fee per kilometer traveled, and a driver waiting-time fee. Taxi sharing also includes a reduction in the total fee based on the number of passengers sharing the ride.<sup>24</sup> Therefore, the formula that was devised for this calculation was:

$$TotalCost = \frac{(Ride\ Fee + (Total\ Distance * Cost\ per\ Kilometer))}{Number\ Of\ Passengers}$$

**Figure 3:** Cost Calculation Equation

While this was the formula that was decided upon, there were difficulties when it came to implementation that will be discussed in Chapter four. This formula still remains in the implementation; however, it is not used.

Finally, with all the preliminary research discussed, the focus will shift to the research on the implementation of the application. The implementation aspect focused on the design of the application itself. It focused on the visual appeal of the application as well as ensuring it functioned

---

<sup>22</sup> Uber. (2019). UberPool vs. UberX - How Does UberPool Work? Retrieved March 30, 2019, from [https://www.uber.com/en-CA/ride/uberpool/?state=G7iYn0jH75cgcZSfSkAjbQ4xyOieMG7m\\_\\_FlamxG93o=&\\_csid=gAM22lbigQKqssDDOuFa4w#\\_](https://www.uber.com/en-CA/ride/uberpool/?state=G7iYn0jH75cgcZSfSkAjbQ4xyOieMG7m__FlamxG93o=&_csid=gAM22lbigQKqssDDOuFa4w#_)

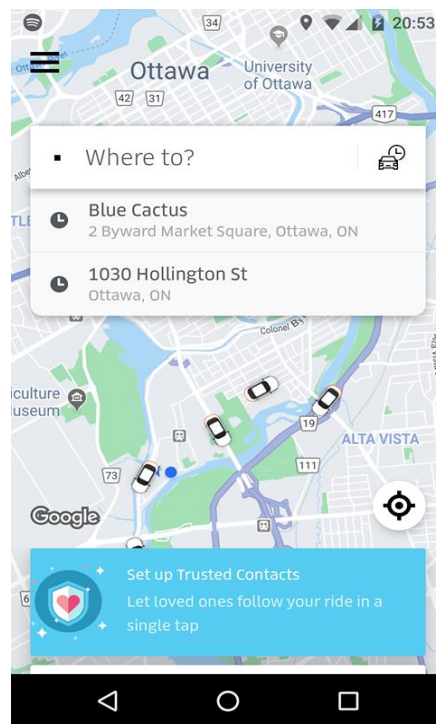
<sup>23</sup> RideShareApps. (2015, August 24). Lyft Line - What Is It And How Does It Work? Retrieved March 30, 2019, from <https://rideshareapps.com/lyft-line/>

<sup>24</sup> GmbH, S. M. (2019). Get your Taxi Fare now! Retrieved January 15, 2019, from <https://www.taxi-calculator.com/>

smoothly on Android devices. Both Uber and Lyft influence the design in their own ways with regards to this application. However, multiple unique design options were chosen in an effort to make this application distinct from others in development.

### *Design Decisions*

The main elements of interest with regards to the design included questions such as how does the information display on the screen? The main requirement was to be able to connect users together and then display this information on the screen. The question then became: how should this information be displayed? Was there a perfect method in an application that already exists? Or, would a combination of elements be the preferred decision? Through observing the Uber screen as well as the Lyft screen, it was determined that a combination would be preferential. Refer to figure four below to begin.

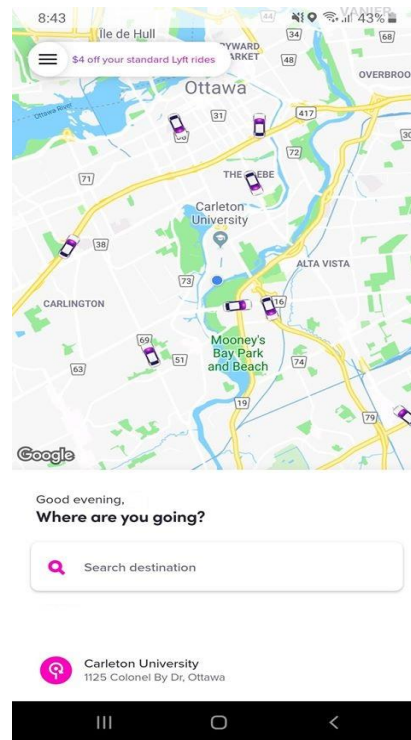


**Figure 4:** Uber Pre-Ride Screen.

Figure four is the screen a user is presented with when loading the Uber application. It features a search bar at the top of the screen as well as a user menu in the top left corner. While the design is sleek and professional, a concern stems from the fact that a large majority of the

screen displays information the user may be uninterested in. For example, the blue rectangle towards the bottom of the page covers a large portion of the map that could be helpful to the user (i.e. to see the availability of other nearby drivers). Another example would be the previous locations area under the search bar. This would be better served as a viewable option in the user menu, instead of taking up a portion of the screen. This project drew some influence from Uber. The user information placement on the app was the perfect location, given the design elements currently in place for the application.

Moving on from Uber, Lyft was the final application included in the design research. These two applications were the only two choices due to their popularity. As stated in chapter two, these applications rank first and second respectively in the taxi application community. However, Lyft influenced the design of the application far more than Uber. While Uber tends to obscure the majority of its map, Lyft displays a full view of its map. Figure five below demonstrates this.

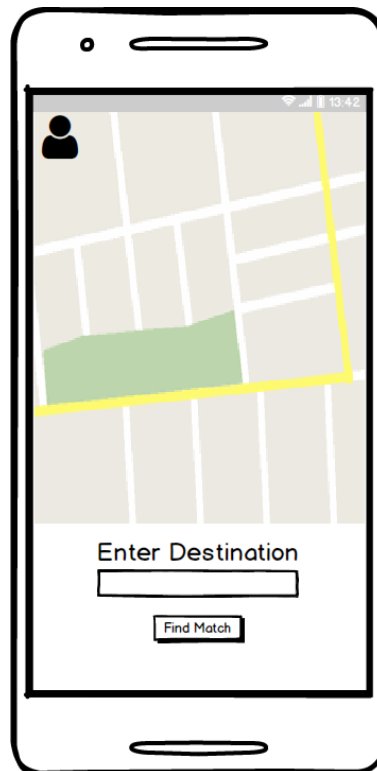


**Figure 5:** Lyft Opening Page

Figure five shows the screen a user is shown when logged into Lyft, once again it presents the user with a very professional design. A stark difference between Uber and Lyft is that the search bar in the Lyft app is located at the bottom of the screen, as well as the previous locations

listed below it. This design seemed a lot more spatially effective, and thus was used as inspiration for this project's implementation. Additionally, similar to Uber, the user profile information is placed in the top left corner of the screen, further justifying the reasoning behind the project's placement of it.

With these inspirations in mind, the design of the application began to take shape. To ease into implementation further, a Balsamiq low-fidelity prototype was developed for the application. A low-fidelity prototype is essentially a drawing, or mockup in this case, of the final design of a product. It gives the creator the ability to view the ideas before actually implementing them. Therefore, as this project was primarily focused on design-based research, it seemed to be the correct idea to test out the implementation before actually creating it.



**Figure 6:** Balsamiq Mockup of Pre-Match Screen

Figure six features the Balsamiq Mockup created for the first screen of the application once logging in. The entire Balsamiq creation is included within the appendix of the report, as well as a control flow and legend for the different elements on the images. The control flow demonstrates how each screen of the design meshes together to form what will be the final



application. When comparing figure 5 to figures 3 and 4, there are certain similarities and certain differences. Firstly, the map takes up a majority of the screen. Since it is a taxi application the map should be a main focus of the application. This is in contrast to Uber and similar to Lyft. Furthermore, the search area is at the bottom of the screen, once again similar to Lyft. However, absent from this design is the previous ride element, which was a feature that was not thought to be needed in the scope of the application. Since the application is programmed to always initiate at a set location, the history of rides was omitted. Finally, the user profile area which was similar in both Uber and Lyft was changed slightly in the project's design. As Uber and Lyft are more robust applications with a greater number of features available to the user,<sup>25</sup> their user menu needed to reflect this ability. The application designed for this project had a much simpler scope in mind for the design. Therefore, this simplicity took the form of a button used as the user menu, which would display a user profile. These elements discussed form the basis of the main similarities and differences between the applications researched, and also form the basis of the major design decisions for this application. Each of the applications follows the approach of keeping the map as the main focus of the screen and attempts to keep the information to the bottom of the page, to prevent it from impeding on the map.

With these requirements in mind, as well as the design decisions chosen for the application, all the information has been stated as to why decisions were made. Following this point, the actual implementation of the application will be discussed. This will delve into the details on how the decisions and objectives were carried out. In addition to this, this section will discuss whether they were modified or eliminated once development began.

---

<sup>25</sup> Pratap, M. (2018, October 26). How to Build an App like Uber? (Complete Guide) - Uber Clone. Retrieved January 30, 2019, from <https://www.engineerbabu.com/blog/how-to-build-an-app-like-uber/>

## Chapter 4: Implementation

This section will outline the process of implementation carried out for the project in relation to the objectives mentioned in chapter three. It will discuss in detail the process of the application's development such as: how the application was completed from start to finish, beginning with the setup which describes the components needed before the implementation could begin. This will be followed by the development process, which will describe the steps that were taken once all the materials were gathered in order to complete the application. After implementation has been discussed, the final section will explain the testing which was conducted during development between Mr. Gahelrasoul's app and this application. This section will also include a discussion on the errors which were discovered during development and how they impacted the end result of the system. By the end of this section the goal is to deliver a clear view of the development process of the project, and how the design transitioned from the research in chapter three to the final application which was delivered.

### *Setup*

To begin, once all the requirements were gathered, the project required organization. This involved gathering the required software for the application, as well as organizing the requirements into manageable sections. Therefore, the first stage of development involved determining all the required environment information and equipment necessary to complete a project of this magnitude. As this project involved the use of a map on an Android device this meant that certain Google services would be required. Through the information retrieved in the requirements gathering, the necessary equipment was determined as listed below. Details regarding each element will be discussed following.

- Environment
  - Android Studio version 3.3
    - IntelliJ IDEA 2018.2.2 (IDE base for Android Studio)
  - Balsamiq markup (Prototyping tool) version 3.5.16
  - Android version 9.0 (Target Android version)
  - Java version 1.8 (Back-end language)

- Firebase (database) version 16.0.8
  - Authentication version 16.2.0
  - Cloud Storage version 16.0.1
  - Geofire (Location Service Storage) version 2.1.1<sup>26</sup>
  - Realtime Database version 16.0.1
- Google Play Services
  - Google Maps API version 16.0.0
  - Google Location Services version 16.0.0
- Bumptech Glide (Image Display and Storage) version 4.9.0<sup>27</sup>
- Equipment
  - Razer Blade Pro
    - Windows 10 (Laptop operating system)
  - LG Nexus 5 (Phone used for testing)
    - Android Version 6.0.1

The elements described above encompass the entirety of the software and hardware used to create the application. The equipment used, Razer Blade Pro and LG Nexus 5, were chosen out of convenience, as they were the devices owned at the time of implementation. There was no specific reason for choosing these devices, the application could have been created on any computer that had the capability of running Android Studio. Furthermore, the application could have been tested on any Android device running an operating system higher than Android 5.1, as this was the minimum Android SDK version possible given the features of the application.

To continue, the software mentioned above includes the entirety of the tools required to create the application. Android Studio was the chosen environment for the project. It is the defacto standard for Android application development and as mentioned in chapter one it is based off the IntelliJ IDEA system. The version of this application was the most recent at the time of development. There has since been an update to Android Studio, however, and to mitigate errors the version was kept the same as the one chosen at the start of development. Balsamiq was chosen as the prototyping tool for the design of the system. As mentioned in chapters' one and three it

---

<sup>26</sup> Firebase. (2018, April 23). Firebase/geofire-java. GitHub repository: <https://github.com/firebase/geofire-java>

<sup>27</sup> Bump Technologies. (2019, April 02). Bumptech/glide. Github repository: <https://github.com/bumptech/glide>

allowed for a simple low-fidelity prototype to be designed for the application before implementation began. Android 9 is currently the most recent version of the Android operating system and thus was chosen as the target SDK version. This was to allow all modern applications the ability to utilize the application. Furthermore, as mentioned above Android 5.1 is the minimum SDK version due to the features implemented. Since Android Studio was the chosen IDE for the system this allowed for two languages to be available for back-end development: Java and Kotlin. Kotlin required additional research and as such was disregarded, while Java was more familiar and allowed for a simpler development. Java version 1.8 was used due to familiarity with the system as well as it being the default Java version currently.

Firebase was the chosen back-end database system for implementation. Since Firebase is a Google platform and has integration built into Android Studio, it allowed for easy back-end development and storage. The version chosen was the most recent version at the beginning of development. The four elements implemented through Firebase included: authentication, cloud storage, Geofire, and a realtime database. The authentication was used for the login component of the application, it allowed users to create an account and have it be verifiable. When an account was created, all the information would be saved to the realtime database and the username and password would be logged in the authentication area of the application. It is important to note that the password is not displayed in the database for privacy purposes. This would allow the user to simply log in with the same credentials when opening the app next. The cloud storage was used to store profile images. When a user would add a picture to their account creation it would be added to the cloud storage area of Firebase. Bumptech glide was used as a companion to this storage, as Firebase can easily store images however it is difficult to retrieve the information. Bumptech glide is a program developed by Bump Technologies stored on Github that retrieves the image stored in Firebase and will assign it to the given element of the application.<sup>28</sup> All that is required to use Bumptech glide is a compilation line in the gradle of the application. Furthermore, another Github compilation was included in this application. This one was developed by Firebase to be used as a companion to their service. Geofire is a program which allows an application to easily store a user's location, given the information being provided to it, to the Firebase database associated with

---

<sup>28</sup> Bump Technologies. (2019, April 02). Bumptech/glide. Github repository: <https://github.com/bumptech/glide>

the project.<sup>29</sup> While Firebase can store all the information provided there was still a need for additional components to use a majority of the functionality for a taxi sharing application.

Google Play Services was the final software element that was required for setup. Since it is an Android application, Google Play Services are required to utilize any of the device specific features. The two required features for the application were Google Location services, and Google Maps API. To gain access to these services the developer must include the project in the google cloud platform, enable billing on their account, and activate the required features in the API settings of the platform. The Google Maps API allows for the map to be displayed on the screen of the application, since this project is a taxi sharing application this feature was required. Following that, location services needed to be enabled. This allowed for the user's current location to be displayed on the screen and allowed for the necessary information to be saved. With these hardware and software components found and implemented into Android Studio, it was now possible to begin development of the application.

### *Implementation and Error Discussion*

As mentioned before there were two components that had to be achieved before implementation could begin. This first was gathering the required software and research, which is defined above. The second was developing the low-fidelity prototype for the application. This has been briefly touched upon above in chapter three but the usefulness of it will be expanded upon here. With the idea of the application in mind, the first stage of any development is to design the application before implementing it. This allows for an easier flow of progress, especially given that during implementation back-end code has to be created. The development of the application would take much longer to create if the design and the code were done at the same time. The prototype of the application was created using Balsamiq, and the entire design is included below in the appendix as well as a guide to understand how each screen relates to each other.

Once the low-fidelity prototype was created it was time to begin application development within Android Studio. This consisted of creating different screens, called activities, and

---

<sup>29</sup> Firebase. (2018, April 23). Firebase/geofire-java. GitHub repository: <https://github.com/firebase/geofire-java>

populating them with screen elements. Essentially, the first stage of development within Android Studio was to replicate the design of the low-fidelity prototype within Android Studio. A comparison is pictured below.

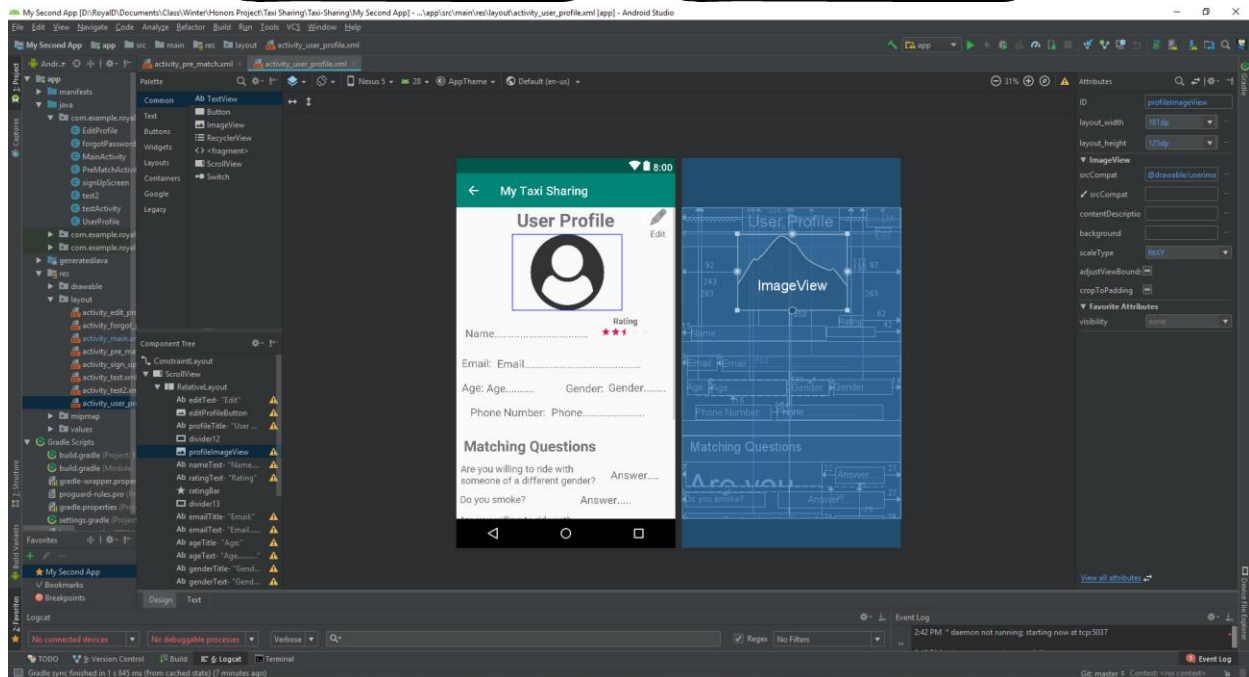
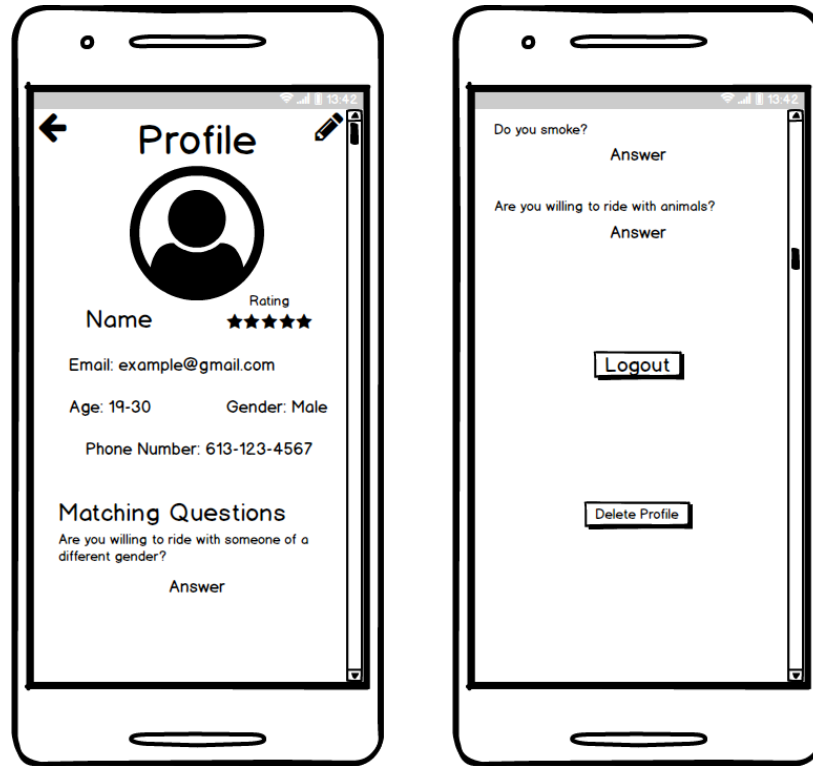


Figure 7: Comparison between Balsamiq and Android Studio

As is apparent through the images in figure seven, the design of the application is the same in both mediums. However, the Android Studio portion has far more elements involved in its creation. These elements include: the design view and the blueprint view both of which are in the center of the screen. The component tree to the left of the design, which lists all the elements on the screen. As well as the attribute window to the right which provides information on a selected component. The design and blueprint views aid the developer in creating the application and determining which elements are connected, and the area above the component tree (the palette) allows for quickly inserting components. At this stage in the process no back-end was being thought of, the goal was simply to replicate the screens that were known to be required and design them. The screens which were possible to develop this way at this stage included: the login page, the sign-up page, the user profile screen and the edit user profile screen. All other activities revolved around Google Maps being implemented and this was still too early in development. All these screens designed in Android Studio have been included within the Appendix of the report. To continue, this stage of development involved placing the elements on the activities in Android Studio. When an activity is created in this application, it creates two files. The first is the XML file which is pictured above and is the design of the screen. The second is a java file which contains the back-end functionality for that activity. These java files will be discussed momentarily.

The design within the XML file is made up of different components and containers, each of these components represents a physical element on the screen, and the containers aid in the organization of the elements. The elements on the screen range from text, to input fields, to buttons, to images. All of these elements are placed within the containers. These containers represent different types of layouts on the screen and are the main way of organizing the information. There are multiple different types of these layouts used for different situations. Most of these layouts were used at one point during development. Such as in the login activity, where a relative layout was used. This allowed for placement of elements anywhere within the layout. The elements were constrained to the edges of the screen as opposed to the layout itself. In comparison, the sign-up screen uses a vertical layout, which has a built-in constraint in the layout itself. Each element is placed one after the other vertically down the screen. This contributes to how the sign-up screen appears so linear. The final requirement for this stage of the design was to give each element on the screen a unique identification. This serves two purposes: it aids in identifying elements later on in development when things may change, and also it is what makes it possible to reference the

elements in the back-end code. When writing the back-end, the code would declare a variable that was the same as the element on the screen and assign the screen element to said variable using the `findViewById()` method. With all these elements defined it is now possible to move on to the second stage of development.

The step following the Android Studio design was to begin implementation of the back-end Java code for the simplistic UI elements and achieve the ability to create a user authentication. This involved giving functionality to the buttons which had been implemented at this point, as well as to implement Firebase into the project. The former took very little time to implement. There are two different ways to implement a button using the Android Studio development environment. The first being to create an on-click method for the button in question. This methodology is typically used when transitioning between screens within the application. The on-click functionality is built into the button within the XML file. Using the attribute editor of the selected button, it is possible to assign a method to the on-click attribute. A method implemented in this fashion is included below.

```
/Function that changes the screen to the user profile screen
public void toUserProfile(View view) {
    Intent intent = new Intent(this, UserProfile.class);
    startActivity(intent);
}
```

**Figure 8:** On Click Function Example

As the above example demonstrates, the method must take in a view variable, which represents the view that the user currently sees. The method then creates an intent, which is essentially a variable that holds the next screen the user wishes to go to. Following this, the method will start the activity placed within the intent. This is the simple functionality which is used for every screen transition within Android Studio. If the button requires more complex implementation, then an on-click listener was used for that element's implementation. An on-click listener is defined when an activity in the application begins and is constantly waiting for the element assigned to it to be pressed. An example is displayed below:



```
createAccount.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //Variables used to check for correct responses
        email = (EditText) findViewById(R.id.emailSignUp);
        password = (EditText) findViewById(R.id.passwordSignUp);
    }
});
```

**Figure 9:** On Click Listener Example

This is a brief section of the on-click listener which is used to create an account for the application. The method in question continues on for two hundred lines and thus will not be fully included. However, it is apparent that much more functionality can be added to an on-click listener. This one in particular references all necessary information on the screen and uses it to create a profile. The specifics of this will be discussed momentarily. To continue with the discussion of the development, at this stage in the process the design has been mostly completed and the transitions between screens has been implemented. At this point testing took place on a physical Android device to determine if the application was visually pleasing as well as if the transitions functioned properly. There were no errors to report at this stage and so development progressed onwards. For reference, screenshots of the entire design are included at the bottom of the appendix. The appendix itself shows the progression of development from Balsamiq images to Android Studio design and then to application functionality.

Following the implementation of the screen transition, the application was beginning to take form. However, it still did not have any required functionality. The next logical step was to implement the ability to create user profiles and save the required information. Once a profile was established, progress could be made on achieving functionality with those users. The actual creation of the profile was not overly difficult, as all the elements had already been implemented in previous stages of development. Firebase had already been implemented and thus all that was required was to call the appropriate functions. Also, the screen elements had already been designed, they just needed to be referenced. Once the information was gathered by on-click listener, the program creates a map to store the information in Firebase. This map creation is shown below:

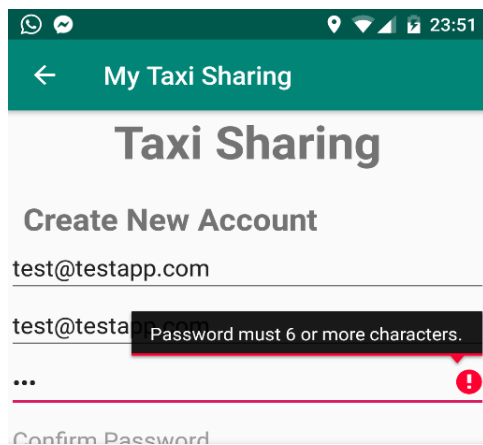
```
//Map used to store the information to the database, field name
and data
Map userData = new HashMap();
userData.put("First Name", fName);
userData.put("Last Name", lName);
userData.put("Email", userEmail);
userData.put("Age", age);
userData.put("Gender", gender);
userData.put("Phone Number", pNumber);
userData.put("Credit Card", cCard);
userData.put("Match Q1", currentQ1Answer);
userData.put("Match Q2", currentQ2Answer);
userData.put("Match Q3", currentQ3Answer);
userData.put("Rating", 0.0f);
userData.put("Searching", "No");
```

**Figure 10:** Map to store information to Firebase

The map stores the relevant information to a key that describes what the information represents. This information is all done within a function that Firebase has, which creates a user based on a unique email and password they enter. Once the information was saved to the database the program would also save the image to the database and display it using Bump Technologies' glide<sup>30</sup> functionality. Since Firebase only stores strings within the user information section of the database, the image could not be saved to the same location. This is where Firebase's cloud storage came into effect. The image would be saved in that location and a URL to this location would be stored in the user profile information area. Bump Technologies' glide method retrieves this URL from the database and converts it back to an image in order for it to be displayed on the screen. The functionality was flawless and thus was kept in the final product, and they have been referenced accordingly. Once all this implementation had been put in place, testing was to begin on this portion of the application. This was where the first error within development was discovered. Unbeknownst at the time, Firebase requires a password to be a minimum of six characters long. For the testing of this application, a password of four characters was used and therefore the account was never created as Firebase rejected the information. However, since the code was implemented properly, and it was a user error, there was limited information to deduce the problem. The solution was found by looking through the log file of the application during run time and in small font it described the authentication problem. Once the password was changed to six characters the profile was created instantly. This error inspired a design update to prevent this

<sup>30</sup> Bump Technologies. (2019, April 02). BumpTech/glide. Github repository: <https://github.com/bumpTech/glide>

situation for future users. If a password is less than 6 characters, the application will prompt them to enter a valid password of 6 or more characters, pictured below.



**Figure 11:** Error Solution due to Account Creation

Following this error, features similar to this error fix were implemented in every user field. This prevents the user from creating an account with invalid information. With a user profile now created and saved to the database, it was now possible to progress to the next stage of development. With the ability to sign in achieved, it was now time to design the rest of the application within Android Studio. The only element missing from the design at this point was the Google Maps API and the elements that belonged on the screen for the map. These elements consisted of every viewable aspect of requesting a ride: being matched with a driver and being on a ride. Both of these tasks proved to be more complicated than previously thought. Creating the maps activity was simple, as once a developer signs up for a Google API key they simply need to create a maps activity using Android Studio's built in functionality and then paste the API key inside the Android Manifest file of the application. Android Studio then takes care of the rest of the implementation for Google Maps. Therefore, once this implementation had been done, testing was attempted on the maps screen, and it was found that the application would crash on load. This was the result of Android Studio incorrectly implementing the maps activity. Since the project began in January of 2019, the version of Android Studio contained all the information from that point. However, Android Studio had updated since the project's start date, which included an updated version of Google Maps. Therefore, after a quick Google search the proper version of 16.0.0 was found and the gradle of the application was updated to reflect this change.

Following the implementation of Google Maps was the implementation of Google Play locations services. This service would be paramount in determining where the user was on the map as well as in relation to the driver. Now the driver side of the application has not been mentioned up until this point as the entire driver side was being implemented concurrently by Mr. Gahelrasoul. The entirety of the implementation of his application and errors will be discussed within his report. Driver-passenger communications will only be discussed where relevant later in this report. However, it is important to keep in mind that since these applications are intertwined, the implementation of the driver side followed a similar pattern to the passenger application. To continue with location services, the functionality proved difficult to implement in the back-end of the application for a number of reasons. Location services required a number of back-end methods to become functional as location implementation is done in two phases. There are the methods which are required to be running constantly in order to have an accurate measurement of the current location. Following this there is the necessary permission check that must be performed in order for the service to become functional on the user's device. In total there were three methods necessary to implement location services as well as error checking required to be added to multiple other methods for it to become functional. The actual location check was the simple portion to implement. There are built-in methods that are able to be accessed once the location service compilation command has been added to the applications gradle, which will return the users current location. The difficulty in implementation came with the requirement of checking for permissions, from Android 6 and on, it is necessary to ask the user for permission before accessing their current location. While the functionality was all implemented properly, it was found during testing that the application would crash when asking for permission. By observing the log file, it was found that the application was attempting to access the user's current location before permission was granted. With this information in mind the code was revisited, and it was discovered that the control flow of the application was incorrect. Two lines were out of place. The application was accessing the location before the function which requested permission was called. Once this access command was moved beneath the function call, the application worked as required and the current location appeared on the screen during testing.

After location services were added to the map, all the elements needed to be added to the maps activity that had not been designed up to this point. This proved more difficult than anticipated, as the Android Studio does not function as normal once the map has been

implemented. The map is stored within a fragment which takes up the entirety of the design screen, making it impossible to drag elements onto the blueprint similar to previous designs. This meant every element needed to be implemented using XML code. This was found to be incredibly time consuming and thus all elements designed on the map's activity page were designed in a temporary activity file instead. Once the designs were finalized, the elements were copied from the temporary activity into the text portion of the XML page on the map's activity. This saved more time for focusing on back end implementation for these elements. An example of what the map's activity looks like as well as the XML code is displayed below:

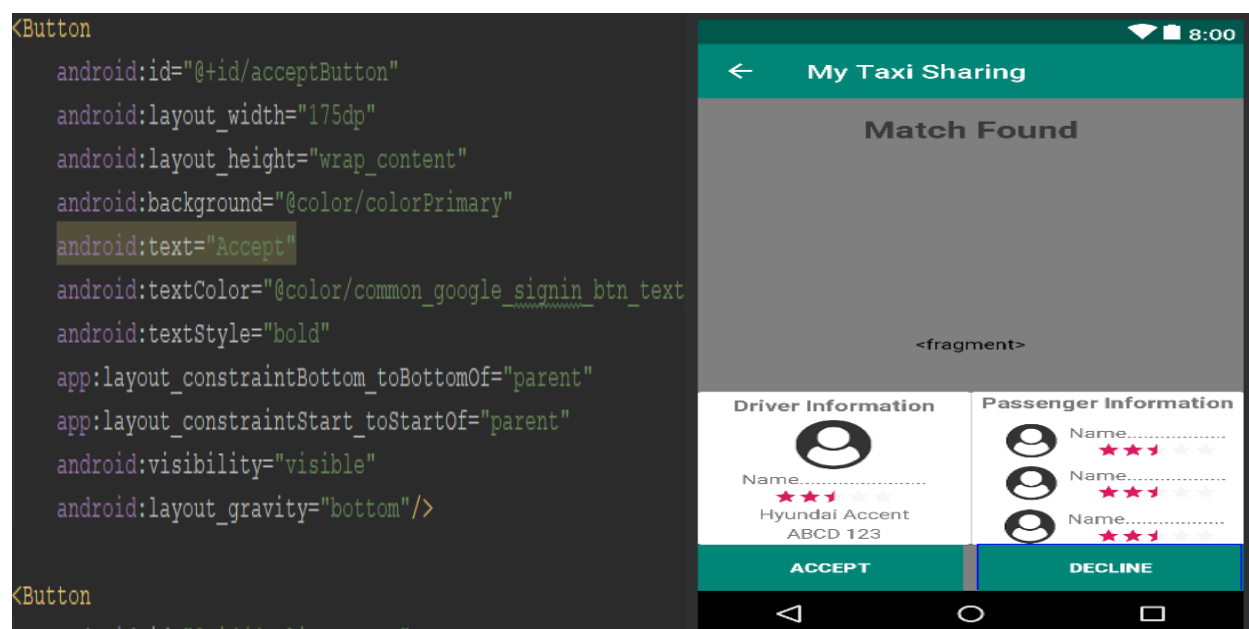


Figure 12: Maps Activity and XML code Example

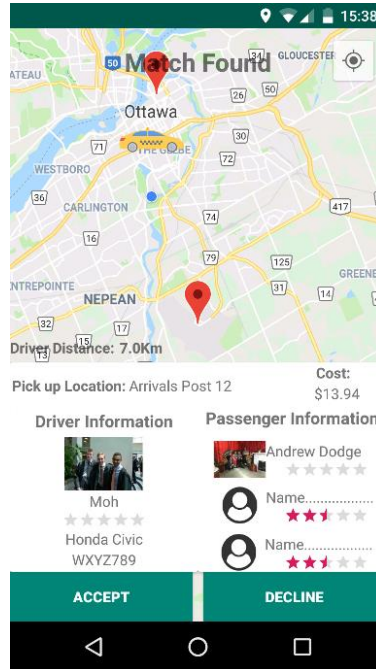
With the design of the application now entirely implemented, the only elements left to focus on were the backend methods which would control the applications behaviour. The next step in development was to implement the user profile functionality. Since the information was entirely saved to Firebase, the data simply needed to be displayed in the proper locations on the screen. This implementation was incredibly similar to how the data was saved initially and thus did not take very long to complete. However, there were two elements of interest with regard to the user profile that halted development for a moment. The first came when attempting to return to the map's activity from the user profile and the second came when attempting to sign out of the application. Both errors caused crashes during testing. The errors were simple to figure out. The first error was due to the map's activity being killed before transitioning to the user profile.

Because of this, when attempting to return the map's activity it was not possible due to it not existing anymore. This error was remedied by removing the finish() command within the activity transition method. The second error was slightly more complicated. It came from a mistake in the back-end control flow of the application. The sign out crash was due to the fact that the application was attempting to reference the current user even once the user had signed out. This was remedied by moving the portion of code that was causing the crash from the location within the map's activity Java file to the sign out method defined within the user profile Java file. At this point in the implementation's lifecycle, the entire application had been designed. It was possible to log in and out of the application, and all user information was displayed on the screen in the proper locations. With these components in place it was finally time to begin the taxi sharing matching element of the application. During this implementation phase, the application had the largest change in development.

Before discussing the change in development, the driver matching will be briefly discussed. The driver matching was quite simple to implement, due to a number of elements. Since the scope of the application was for the pickup location to be set at the Ottawa airport and not to have it be dynamically based on their current location, this aided in implementation. Also, all locations were easily stored to the Firebase database due to the Firebase system Geofire.<sup>31</sup> By including Geofire in the application's gradle file, the built-in database methods became available. This allowed for all determined locations to be saved to the database. This included the current location of the driver (determined in the driver application and stored to the database), and the pickup location of the passenger. Matching with the driver was achieved using a calculation that would find the closest available driver to the pickup location and automatically match this driver to the user in question. The calculation functioned by creating a radius around the customer's pickup location and each time a driver was not found, the radius would be increased by one kilometer until a driver was found. Once a driver was found, all the information for the match would be displayed on the user's screen, pictured below.

---

<sup>31</sup> Firebase. (2018, April 23). Firebase/geofire-java. GitHub repository: <https://github.com/firebase/geofire-java>



**Figure 13:** Displaying a Match with a Driver

Figure 13 displays the screen that the user is prompted with once a match has been completed. It features the driver's current location (the taxi image<sup>32</sup>), the user's current location (the blue dot), the pickup location (which is always the Ottawa airport, and is represented by the red marker to the south on the map), the customers drop-off location placeholder (the north red marker, the reason for a placeholder will be discussed momentarily), the distance the driver is from the pickup location, as well as the information of the driver and the user. The user then had the option to accept the match and be locked to that driver similar to the functionality of Uber or decline the ride and be returned to the enter destination screen. If the ride was declined, all information was removed from the screen and removed from the database. The driver matching implementation functioned flawlessly, the only error that was found during development was that even after declining a match the user would continue to see the driver location on the screen. This was due to the driver calculation having to be performed within the user location function which runs once a second. The error was remedied by adding a boolean flag which would only be set once the user was searching for a match and would be set to false if the user was not searching.

<sup>32</sup> Icon made by [<https://www.flaticon.com/authors/nikita-golubev>] from [www.flaticon.com](https://www.flaticon.com)

Once this functionality was working properly it was time to implement the passenger matching, this is where development halted.

Implementation had been progressing incredibly smoothly up until this point in development. Therefore, the system was due for a major problem. Unfortunately, this came in regard to the main aspect of the taxi sharing application. There were two functional elements which were found to be impossible to implement due to unforeseen circumstances. Furthermore, due to the time constraint of the project specification, when these problems were discovered it was too late to refactor the entire code base, especially since an entire refactoring would only result in a potential fix, not a definite one. Therefore, after deliberation with Mr. Gahelrasoul, these features were scrapped in order to focus on functionality that could still be implemented.

The first feature which was unfortunately eliminated from the final product was the ability to match the user with multiple passengers. At this stage in development matching had been achieved with the driver side of the application. It was determined that this would be the first focus of implementation before attempting to match with multiple passengers. A matching algorithm had been designed into the passenger side of the application which would search the database for users who were classified as searching. If multiple users with this classification were found, their unique customer IDs were added to an array which was to be added to the driver database. This was then to be referenced by both the driver and the customers in order to display the proper information. While it was found in testing that the matching algorithm was functional, it was determined to be impossible to store this array to the database properly. When adding the information to Firebase, only the first passenger ID would be added to the database. Countless hours were spent on attempting to get these unique IDs saved to the database. However, due to time constraints the feature was forced to be abandoned in hopes of implanting the features which still remained. The matching algorithm still exists within the passenger's codebase; however, it has been commented out. Therefore, in the future, when the project is expanded into its end goal the functionality can hopefully be reused. From this point on the development shifted from a taxi sharing application to a taxi application. As users could still match with an active driver, they would simply not be sharing a ride.

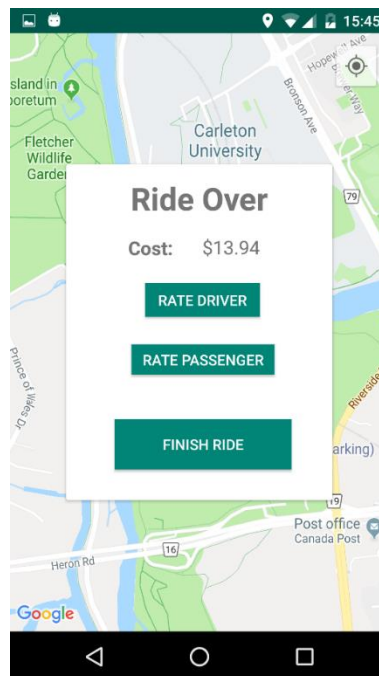
The second feature that was found to be impossible to implement was the entering of a destination and the displaying of a destination on the map's activity. Now it is important to note



that while the feature as a whole does not work, some base implementation was still kept in the final product. At this point in development it was possible to save the string of text the user entered as their destination to the database; this element was kept. Meaning, in the future as this application is evolved into its desired end goal the information will still be accessible. The problem for this feature was due the Google Places API. The goal of the API was to have users enter a destination, then Google service would reference Google's database, place the exact destination on the map and save the location. The entirety of the service had been implemented into the passenger application and during testing it was found that the user was unable to enter a destination. The screen would open for a user to enter a destination. However, each time the user would click on the field the screen would not allow for input. After researching the problem, it was discovered that the implementation, which merged perfectly with the application currently designed, was deprecated. Meaning it had been eliminated in favour of a new implementation. The integration of this new implementation was impossible given the current design of the system. Also, given the time constraint on the project the feature was forced to be scrapped. However, although the user cannot enter a valid destination, the application will still display a pickup location on the map. Furthermore, with the previous system, a calculation which was independent of the Google Place API had been implemented to determine when the driver was approaching the destination. This was able to be kept and is still in the working application. Therefore, while the user cannot enter their desired destination once this becomes feasible, when the application is evolved into its final goal this calculation can be used to determine when a ride has ended. Furthermore, since the destination and matching of passengers was unfortunately scrapped from the final application, this meant that the cost formula which was devised during research was not able to be implemented, as the calculation variables required these features in place. The calculation remains in a commented-out portion of the codebase to be used in the future of the applications development it was replaced with a different calculation. This replacement does not by any means represent the proper cost of a ride using the system. The new calculation was simply used as a means of displaying a unique number for every match the system creates. The new calculation generates a random amount between \$10 and \$20 and displays this information on the screen. This calculation is for visual purposes only and is not based upon any research. To summarize, due to time constraints in the application's development, these core features of the application had to be eliminated. Back-end elements still existed in the code but have been commented out, and from

this point on the application transitioned from a taxi sharing application to a regular taxi application.

While the application was lacking the core functionality that it had set out to achieve, it was still the goal to complete the functionality of the application so the system could still be evolved in the future. The final element that was required for development was to display that a ride had ended for the user matched with the driver. This was achieved using a simple calculation between the driver's current location and the placeholder destination location. It behaved the same as the calculation which told the user the driver was arriving. Once the driver was within 100 meters of the destination, the match information would disappear from the screen and the ride over prompt was displayed on the screen. This screen was simplistic and involved a small number of elements. It presented the user with the cost of the ride that was to be paid to the driver. Now the scope of the project assumes that the user will pay the driver within the vehicle itself. The application does not charge the user using any online payment method. The user instead pays the driver in person using either cash, or a credit/debit card presented at the end of the ride. The end ride screen is pictured below:



**Figure 14:** Ride Over Prompt

The additional elements included on the ride over prompt included a button which would allow the user to rate the driver, a button to allow the user to rate the passengers that the ride was

shared with, and a button to finish the ride. The rating system was included as a stretch goal of the project and due to the time spent on attempting to fix the core functionality, which ended up being scrapped, it was designed but not implemented. Therefore, those two rating buttons are not functional. The elements and prompts that these buttons would display were designed and are still included within the application; However, they are never used. The hope is since the designs are there they can be used in the future of the application. With reference to stretch goals, further elements which were designed but not implemented include: the delete profile button in the user profile and the forgot password screen in the login page of the application. Time became a large factor come the end of the project's life and due to this difficult implementation, decisions were forced to be made. However, the finish ride button is fully functional. It will clear all the information of the ride from the database for both the user and the driver as well as return the user to the initial map screen where they could enter a destination. With the implementation discussed, the project will now be summarized, and this report will be concluded.

## Chapter 5: Conclusion

To summarize, the goal which was set out to be achieved by this project was nearly realized. This project was to be mainly research and user interface design oriented. Functionality, (especially back-end functionality) was to be kept to a minimum. The goal was to research the fundamentals associated with taxi sharing. Determine what elements were necessary for taxi sharing based off applications which were currently in circulation, such as: what goes into the matching of users, how the cost is calculated, and how the information is displayed on the screen. This goal has been achieved. Through the countless hours of research, referenced below, and the background information discussed in chapter two, it has been demonstrated that the necessary information has been gathered and is available for when this project is evolved in the future. Furthermore, the user interface design was given an incredible amount of focus throughout the project's lifespan. Within the appendix of this report it is possible to see the evolution of the project user interface from a rudimentary low-fidelity prototype, to a blueprint design in Android Studio, and finally to the final application design. Accompanied within the appendix is also a legend, detailing the low-fidelity prototype on how design changed over the course of the project.

The secondary goal of the project was to implement these research elements into a working application. The goal of this implementation was to design a taxi sharing application with the hopes that it could evolve into an autonomous taxi service application based upon the research discovered. While the design determined at the beginning of the project projected high hopes for the application, time constraints and impossible integration with regards to design decisions, prevented full functionality from being achieved. The goal of the application was to implement taxi sharing, and while the app functions from start to finish as a taxi application, it does not include the sharing element. However, it does include the main focus of the back-end system. Communication was established as the key goal for this project. The ability to seamlessly communicate between the passenger application and the driver side application through a middle back-end system. This functionality has been established and functions properly. The user can be matched with an active driver without fail. Then the application will display all the relevant information stored in the database to both the driver and the passenger. The functionality missing however, is the ability to match with other active passengers. While multiple users can be active at the same time and can be matched with separate active drivers, the users will never be matched

together as was the initial objective of the implementation. While this takes away from the functionality of the application as a whole, it did not deter the completion of the application. The system still accomplished the task of matching a user to a driver and achieved the functionality necessary for a taxi application. Such as displaying pickup location, displaying driver information, notifying the user of the driver's arrival, and notifying the user of the ride's termination once the destination has been reached.

In conclusion, while the application did not achieve every goal laid out in the early stages of development, there is still enough functionality within the finished application to simulate a full taxi application. Furthermore, the elements which were not realized still have some base implementation within the codebase which were commented out. Therefore, while elements are lacking from the final product, there is still enough implantation realized and information gathered to progress with this project into the future. This application can still be evolved into the autonomous taxi sharing system that was theorized. Slightly more work will be required than originally anticipated. However, the underlying structure which this application hoped to provide has been realized.

## References

Balsamiq Studios. (2019). Balsamiq Mockups 3 Application Overview - Balsamiq for Desktop Documentation | Balsamiq. Retrieved March 30, 2019, from <https://balsamiq.com/wireframes/desktop/docs/overview/>

Bump Technologies. (2019, April 02). Bumptech/glide. Github repository: <https://github.com/bumptech/glide>

Firebase. (2018, April 23). Firebase/geofire-java. GitHub repository: <https://github.com/firebase/geofire-java>

GmbH, S. M. (2019). Get your Taxi Fare now! Retrieved January 15, 2019, from <https://www.taxi-calculator.com/>

Google. (2019). Firebase Overview. Retrieved March 30, 2019, from <https://firebase.google.com/>

Guzman, Z. (2019, March 27). Why Lyft's IPO may be more attractive than Uber's. Retrieved April 19, 2019, from <https://finance.yahoo.com/news/lyfts-ipo-may-attractive-ubers-193328062.html>

Iqbal, M. (2019, March 25). Lyft Revenue and Usage Statistics (2019). Retrieved March 30, 2019, from <http://www.businessofapps.com/data/lyft-statistics/>

Iqbal, M. (2019, February 27). Uber Revenue and Usage Statistics (2018). Retrieved March 30, 2019, from <http://www.businessofapps.com/data/uber-statistics/>

Lyft, Inc. (2018). About Shared rides. Retrieved March 30, 2019, from <https://help.lyft.com/hc/en-ca/articles/115013078848-About-Shared-rides>

Lyft, Inc. (2019). Ride with Lyft – Friendly drivers and serious safety. Retrieved March 30, 2019, from <https://www.lyft.com/rider>

Pratap, M. (2018, October 26). How to Build an App like Uber? (Complete Guide) - Uber Clone. Retrieved January 30, 2019, from <https://www.engineerbabu.com/blog/how-to-build-an-app-like-uber/>

RideShareApps. (2015, August 24). Lyft Line - What Is It And How Does It Work? Retrieved March 30, 2019, from <https://rideshareapps.com/lyft-line/>

Uber. (2017). UberPOOLSharing is saving. Retrieved March 30, 2019, from <https://www.uber.com/info/uberpool-nj/>

Uber. (2019). About Uber - Our Story - Vision for Our Future. Retrieved March 30, 2019, from <https://www.uber.com/en-CA/about/>

Uber. (2019). UberPool vs. UberX - How Does UberPool Work? Retrieved March 30, 2019, from [https://www.uber.com/en-CA/ride/uberpool/?state=G7iYn0jH75cgcZSfSkAjbQ4xyOieMG7m\\_\\_FlamxG93o=&\\_csid=gAM22lbigQKqssDDOuFa4w#\\_](https://www.uber.com/en-CA/ride/uberpool/?state=G7iYn0jH75cgcZSfSkAjbQ4xyOieMG7m__FlamxG93o=&_csid=gAM22lbigQKqssDDOuFa4w#_)

UberLyftDriver. (2017, October 20). Changes to Lyft Line Payment. Retrieved March 30, 2019, from <https://www.ridesharingforum.com/t/changes-to-lyft-line-payment/131>

UberLyftDriver. (2018, March 04). Via, the Alternative to UberPOOL and Lyft Line. Retrieved March 30, 2019, from <https://www.ridesharingforum.com/t/via-the-alternative-to-uberpool-and-lyft-line/773>

Technologies, U. (2019, January). Learning the interface. Retrieved April 18, 2019, from <https://docs.unity3d.com/Manual/LearningtheInterface.html>

Via. (2019). Smarter shared rides - Download the app now! Retrieved March 30, 2019, from <https://ridewithvia.com/>

Walter, D. (2018, October). What is Android Studio? - Definition from WhatIs.com. Retrieved March 30, 2019, from <https://searchmobilecomputing.techtarget.com/definition/Android-Studio>

Waymo. (2018). Waymo – Waymo. Retrieved March 30, 2019, from <https://waymo.com/>

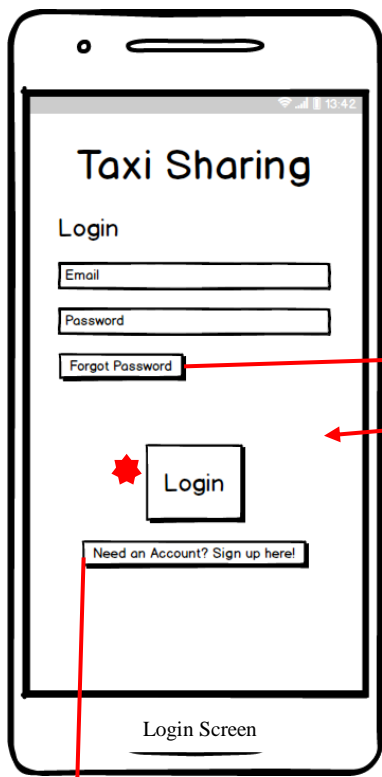
Taxi Image in Icon

Icon made by [<https://www.flaticon.com/authors/nikita-golubev>] from [www.flaticon.com](http://www.flaticon.com)



## Appendix

### Balsamiq Mockups




**Taxi Sharing**

Login

Email

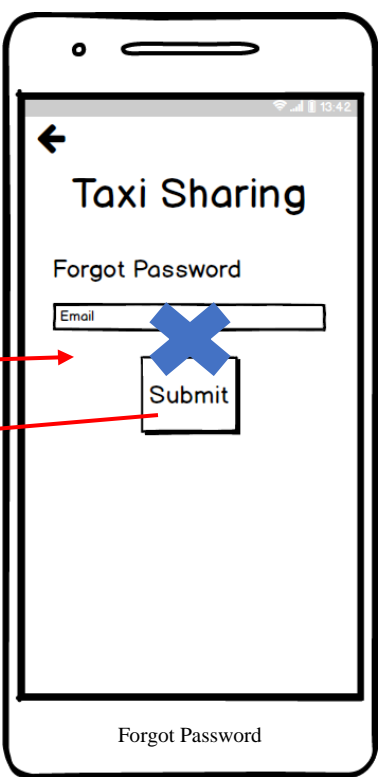
Password

Forgot Password



Need an Account? Sign up here!


Login Screen









**Taxi Sharing**

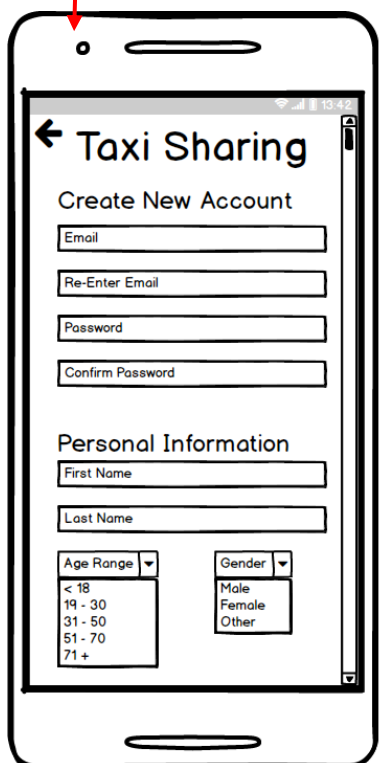
Forgot Password

Email



Forgot Password

Legend for Mockups	
Symbol	Meaning
Red Arrow 	Button clicked leads to the screen the arrow is pointing to.
Red Star 	Buttons with this symbol lead to the Pre-Match Screen.
Blue Star 	Buttons with this symbol lead to the Login Screen.
Red X 	Feature was removed during development.
Green Star 	Button with this symbol lead to Pre-Match Screen 2.
Blue X 	Feature was designed but not implemented.



**Taxi Sharing**

Create New Account

Email

Re-Enter Email

Password

Confirm Password

Personal Information

First Name

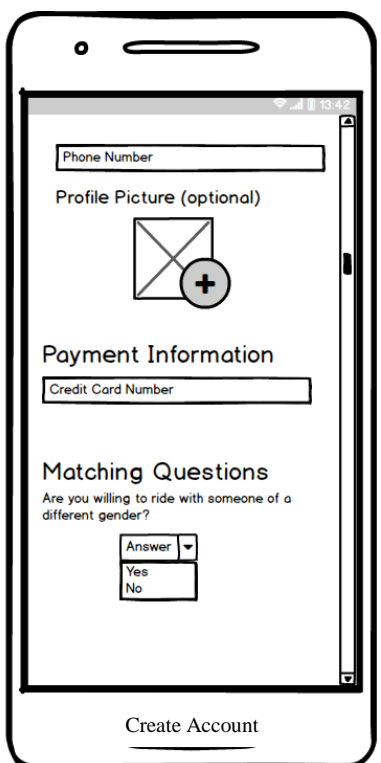
Last Name

Age Range

Gender


Age Range: < 18, 19 - 30, 31 - 50, 51 - 70, 71+

Gender: Male, Female, Other



Phone Number

Profile Picture (optional)



Payment Information

Credit Card Number

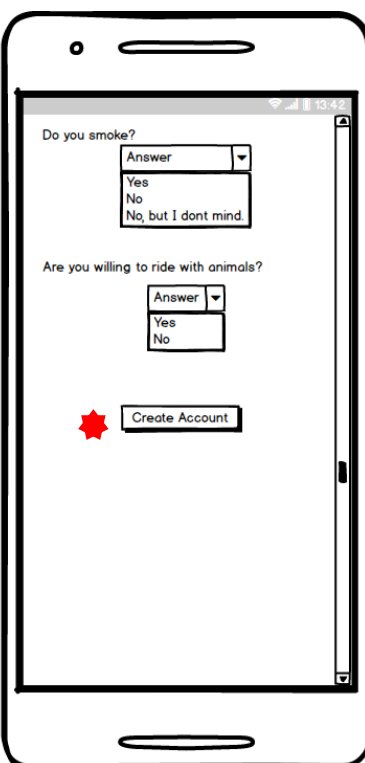
Matching Questions

Are you willing to ride with someone of a different gender?

Answer

Yes, No

Create Account



Do you smoke?


Answer

Yes, No, No, but I dont mind.

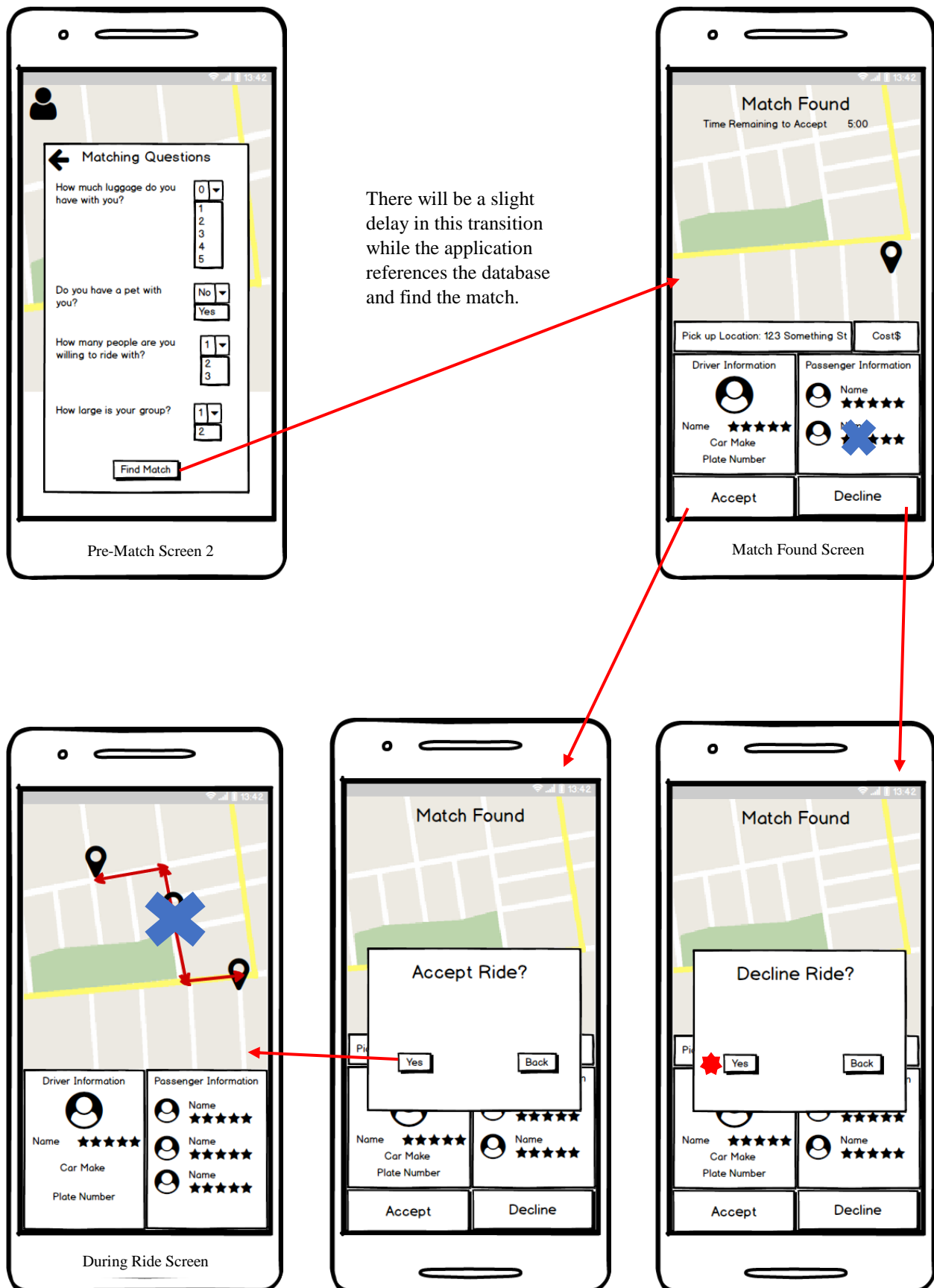
Are you willing to ride with animals?

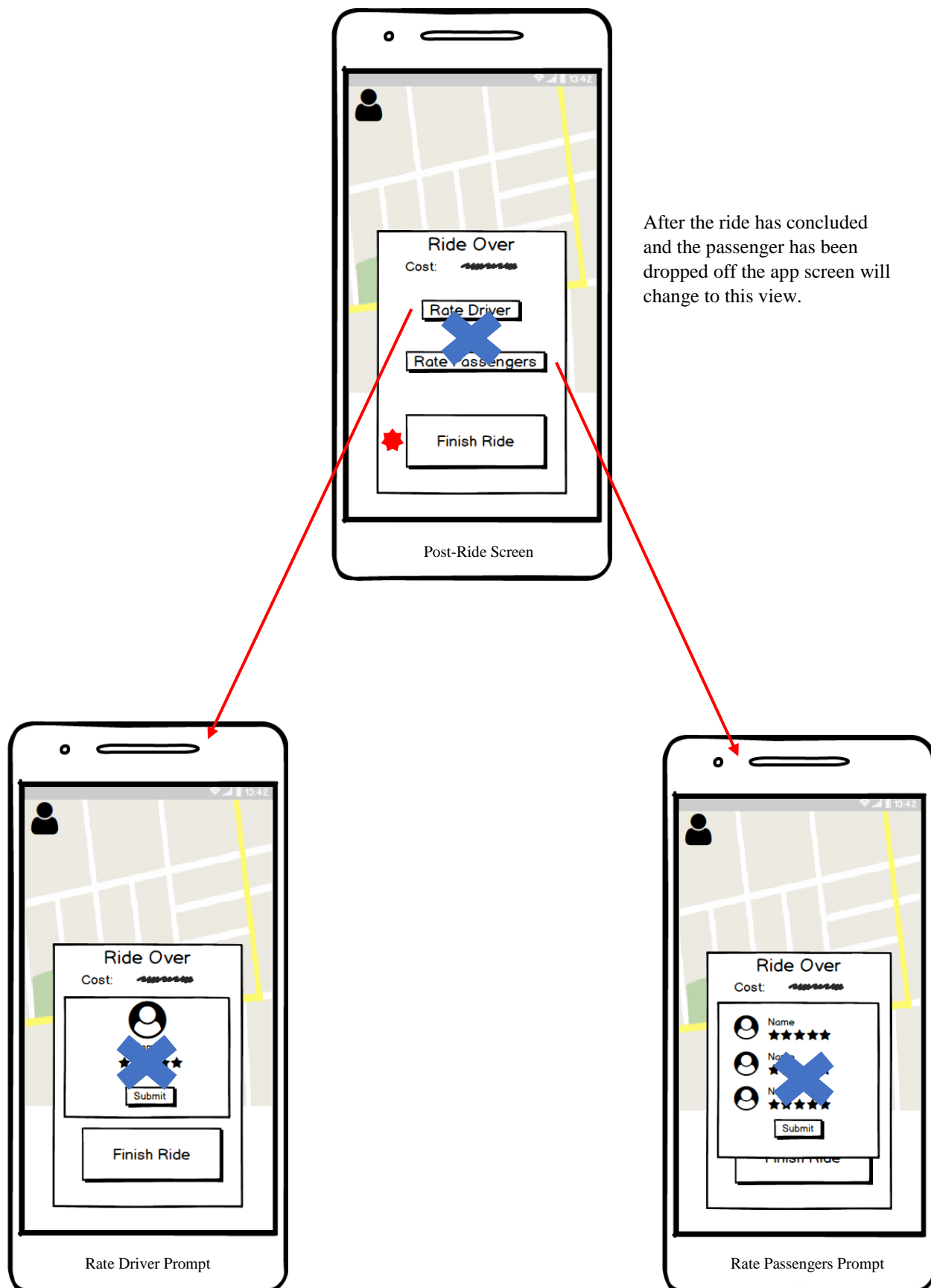
Answer

Yes, No

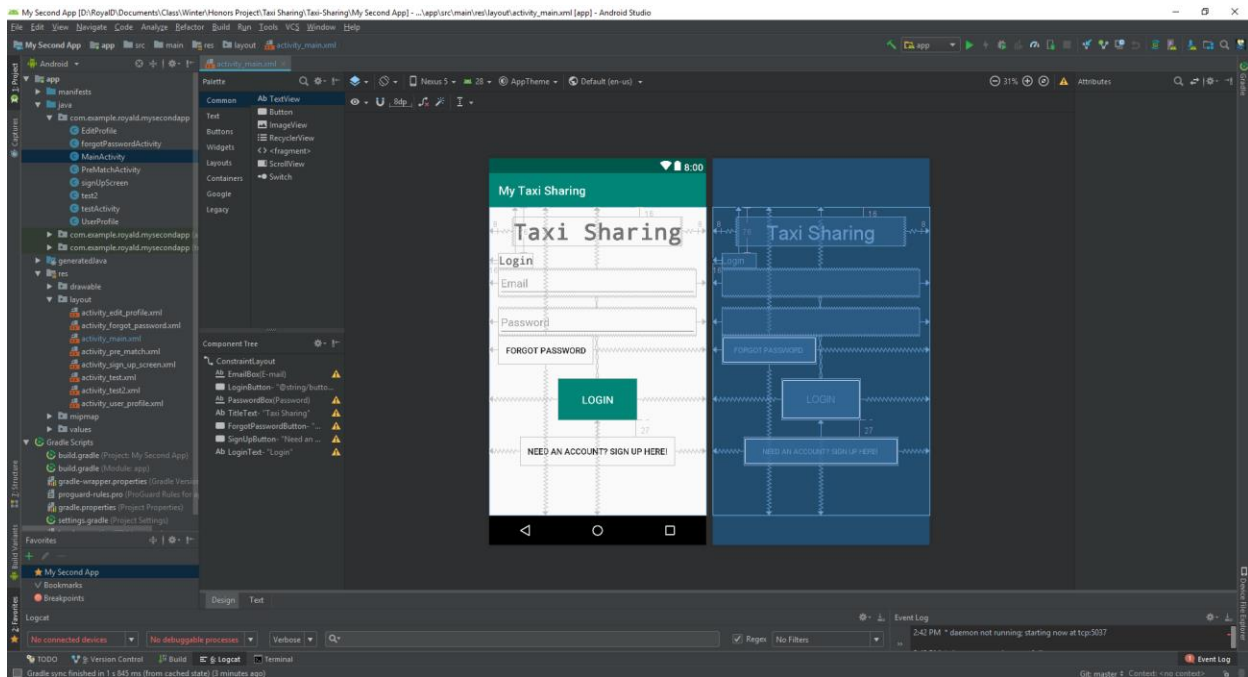




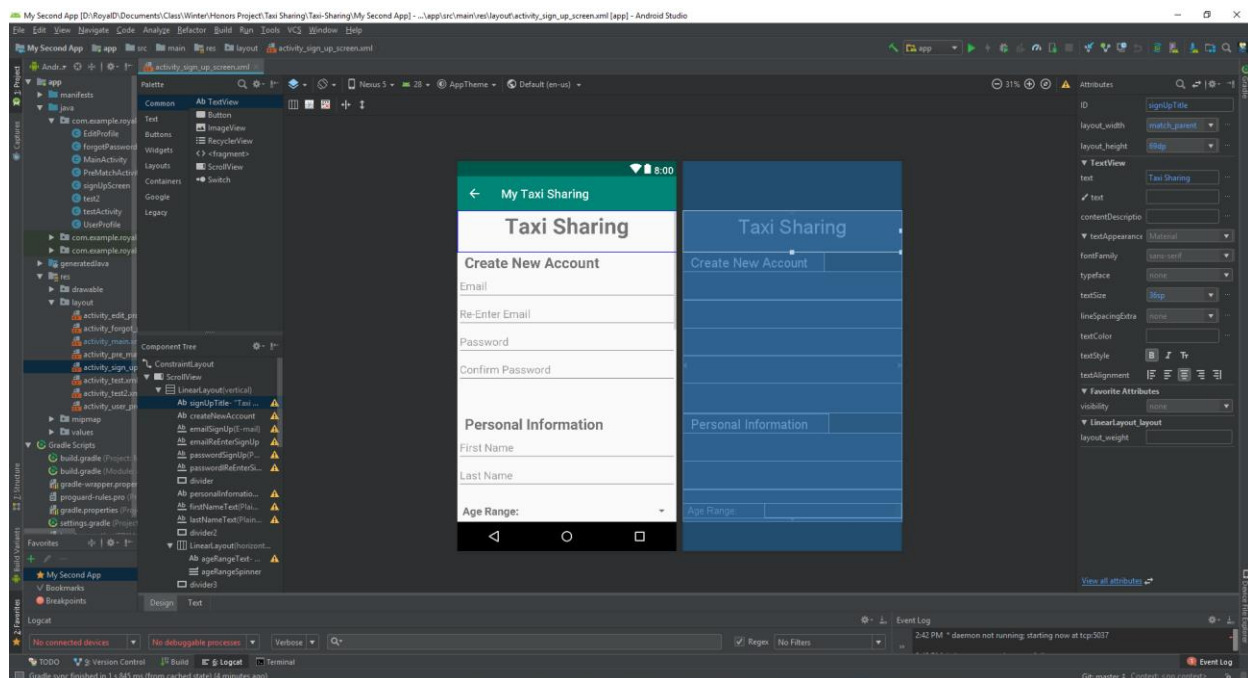




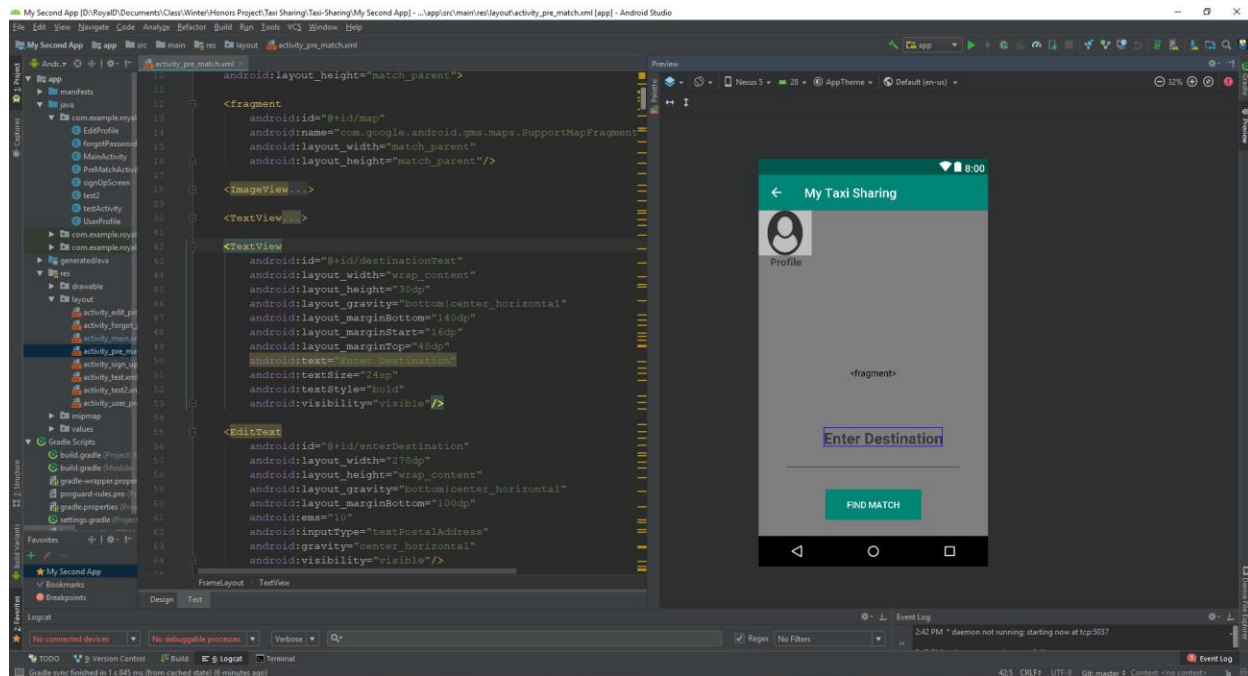
## Android Studio Design



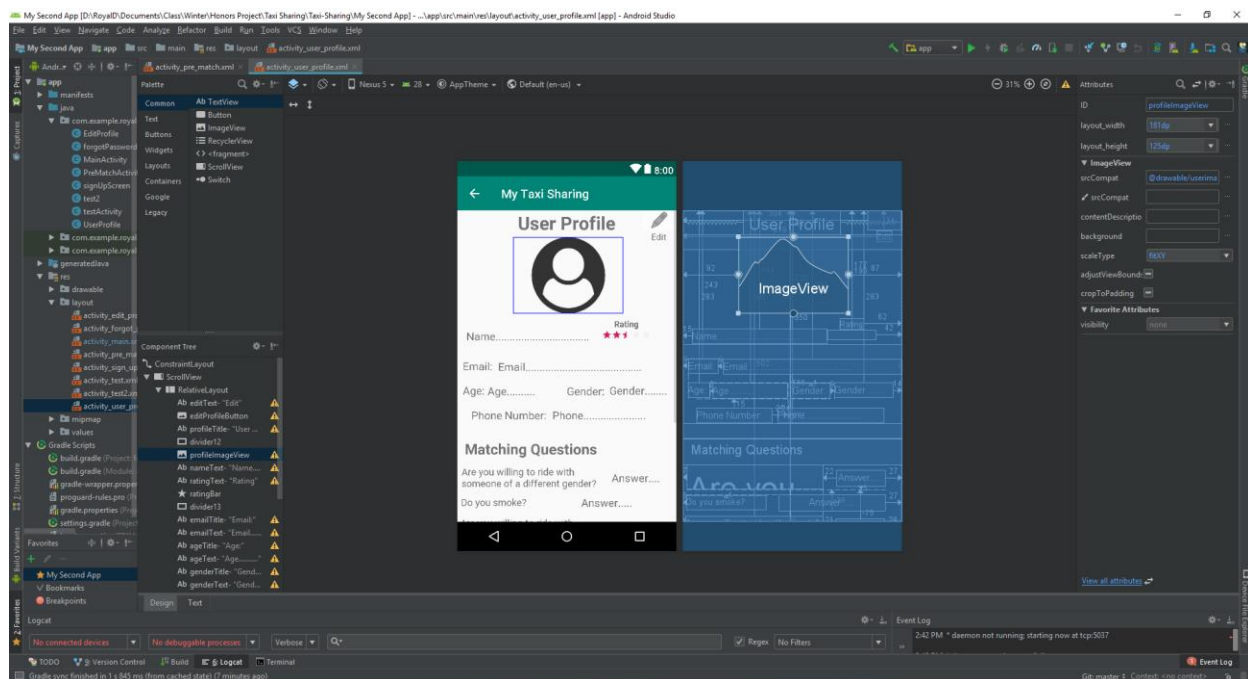
Login Page Android Studio Design



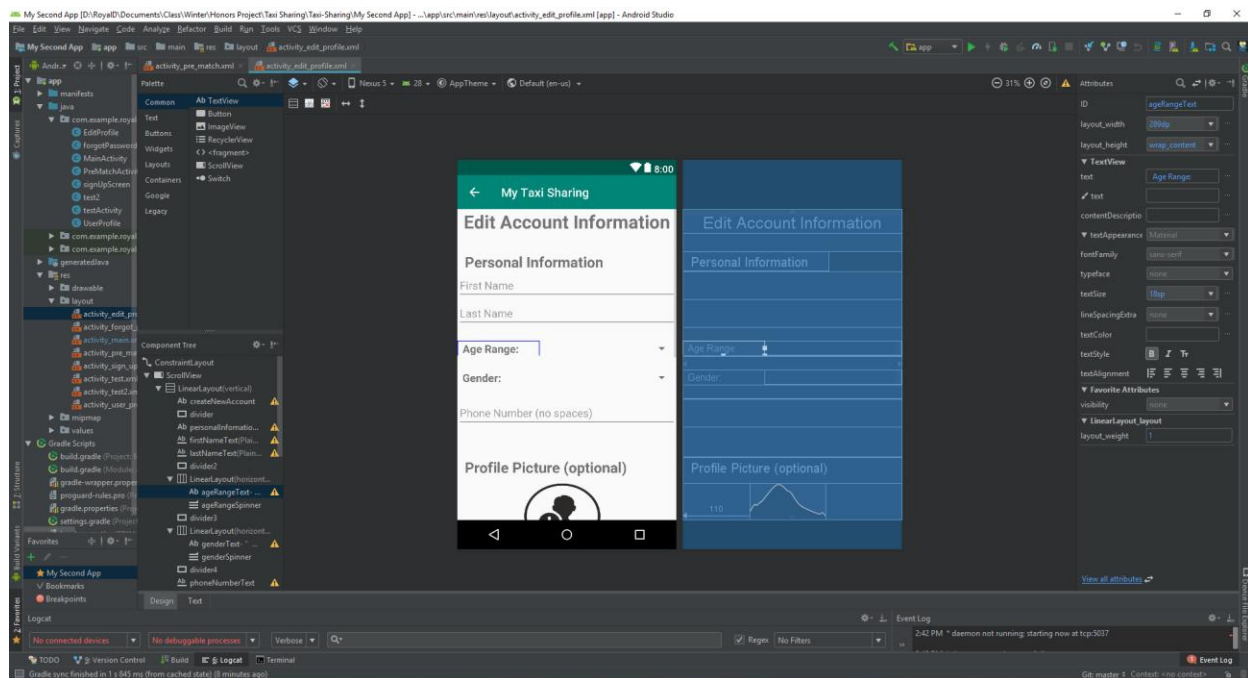
Sign Up Page Android Studio Design



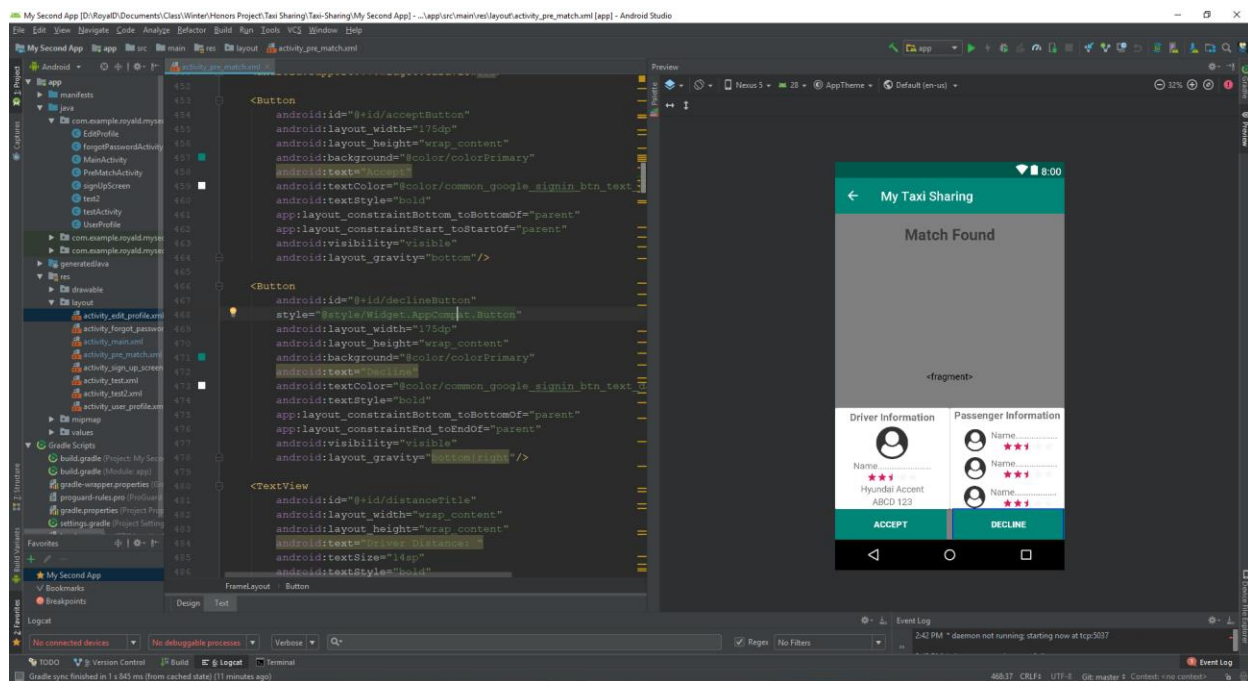
Pre-Match Android Studio Design



User Profile Android Studio Design

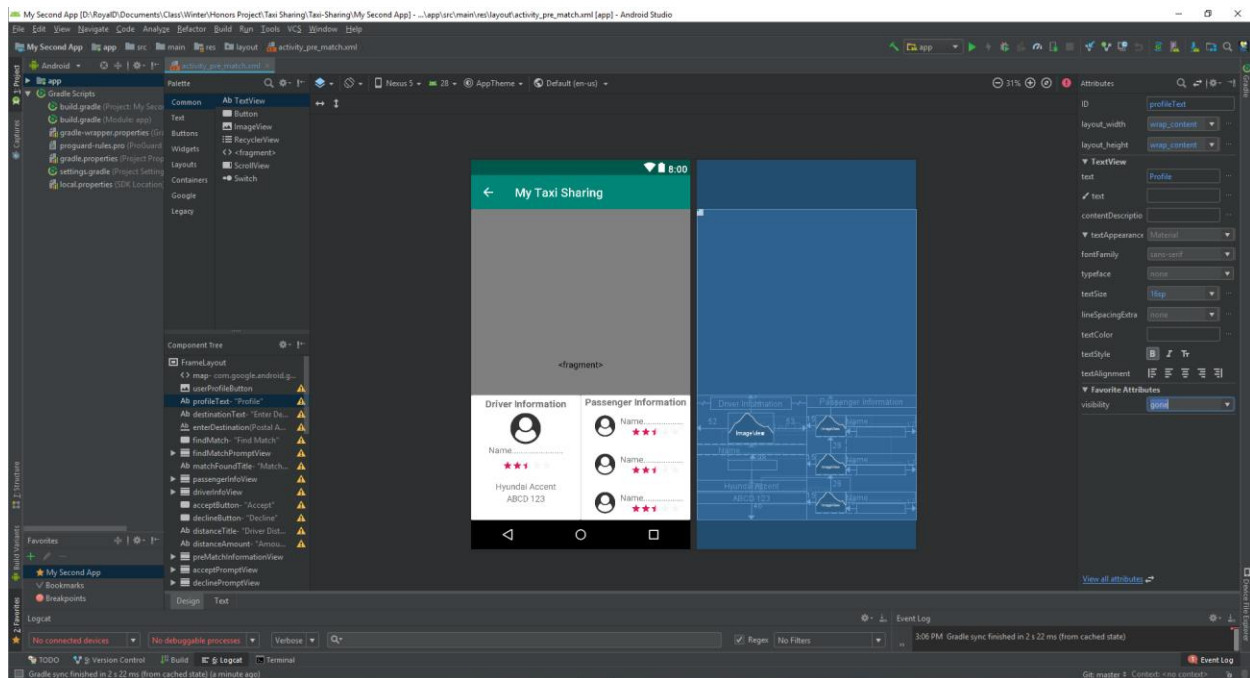


Edit Profile Android Studio Design

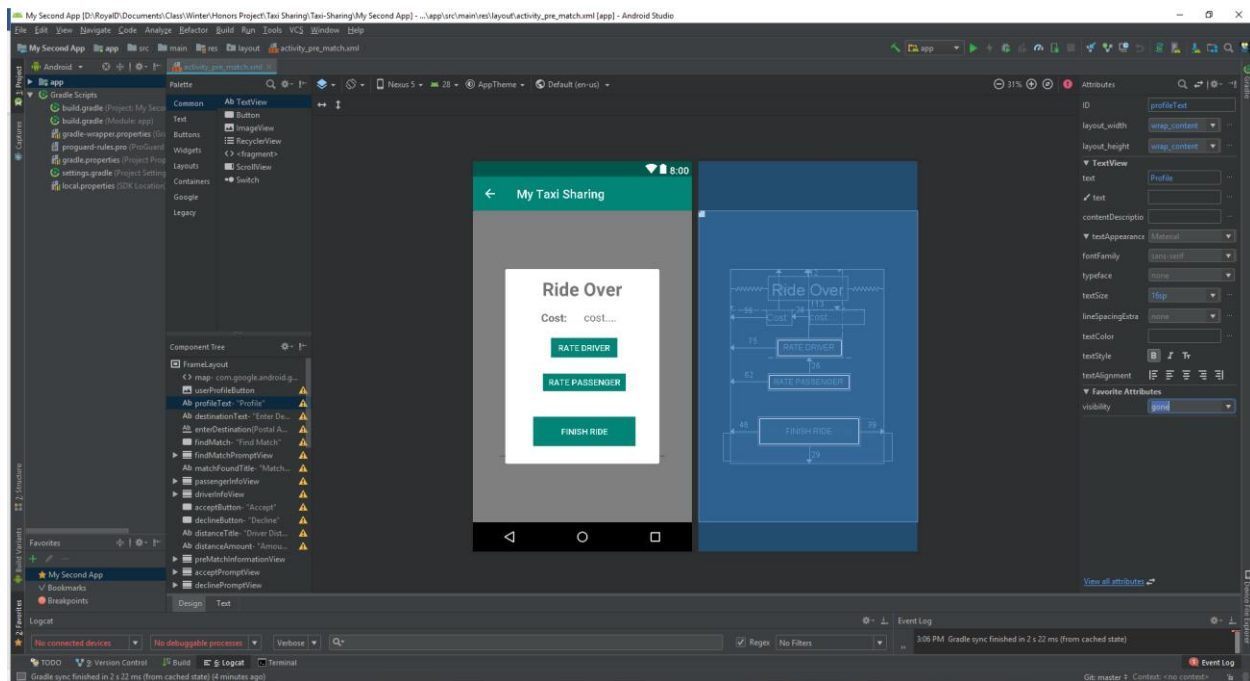


Match Found Android Studio Design





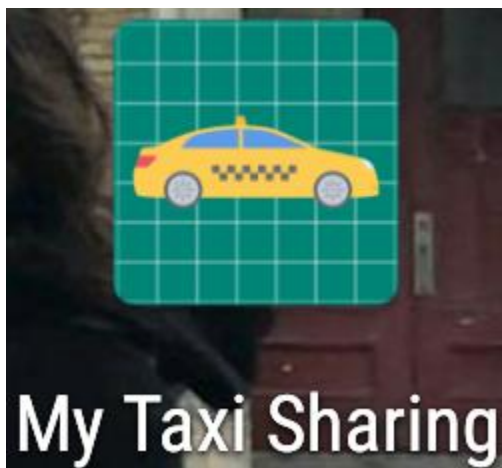
During Ride Android Studio Design



Rive Over Android Studio Design



## Final Application Design



Application Icon

 The login screen has a green header bar with the text "My Taxi Sharing" and a status bar at the top showing location, signal, and time (15:14). The main title "Taxi Sharing" is in a large, bold, grey font. Below it, the "Login" section contains an "Email" input field with a pink underline, a "Password" input field with a grey underline, and a "FORGOT PASSWORD" link. A green "LOGIN" button is positioned below these fields. At the bottom, there is a link that says "NEED AN ACCOUNT? SIGN UP HERE!". The screen is framed by a black Android navigation bar at the bottom.

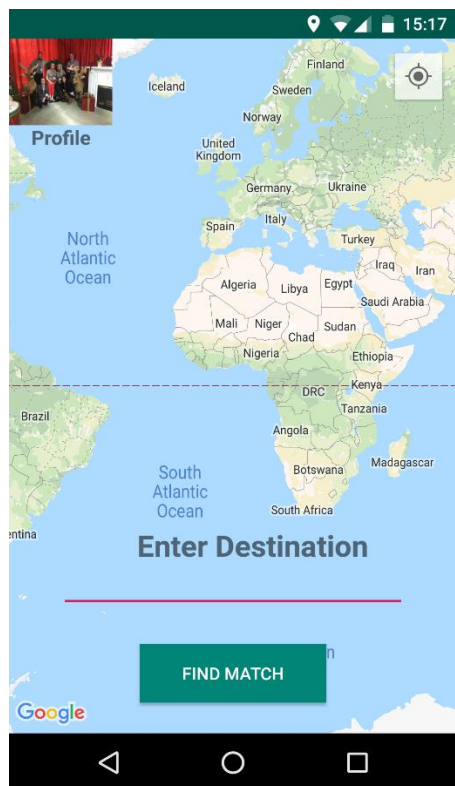
App Login Screen

 This screen shows the "Create New Account" section with two "Email" input fields, each with a pink underline, and two password fields represented by dots. Below this is the "Personal Information" section with fields for "Andrew" and "Dodge", and an "Age Range" dropdown menu set to "19 - 30". The screen has a green header bar with a back arrow and the text "My Taxi Sharing".

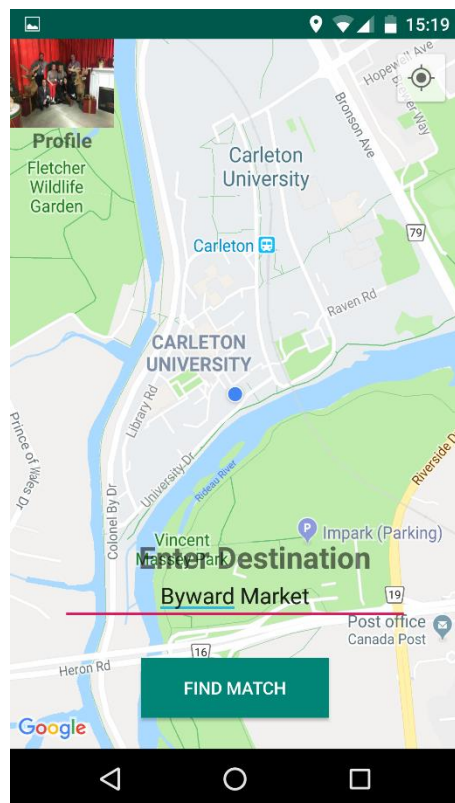

 This screen continues the account creation process. It features a "Gender" dropdown menu set to "Male", a phone number input field containing "1234567890", a "Profile Picture (optional)" section with a placeholder image of a group of people, and a "Payment Information" section with a card number input field containing "1111222233334444". The screen has a green header bar with a back arrow and the text "My Taxi Sharing".


 This screen displays the "Matching Questions" section. It includes three questions with dropdown menus: "Are you willing to ride with someone of a different gender?" (set to "Yes"), "Do you smoke?" (set to "No, but I do not mind"), and "Are you willing to ride with animals?" (set to "Answer"). A green "CREATE ACCOUNT" button is at the bottom. A small dropdown menu is open next to the "Answer" option, showing "Yes" and "No". The screen has a green header bar with a back arrow and the text "My Taxi Sharing".

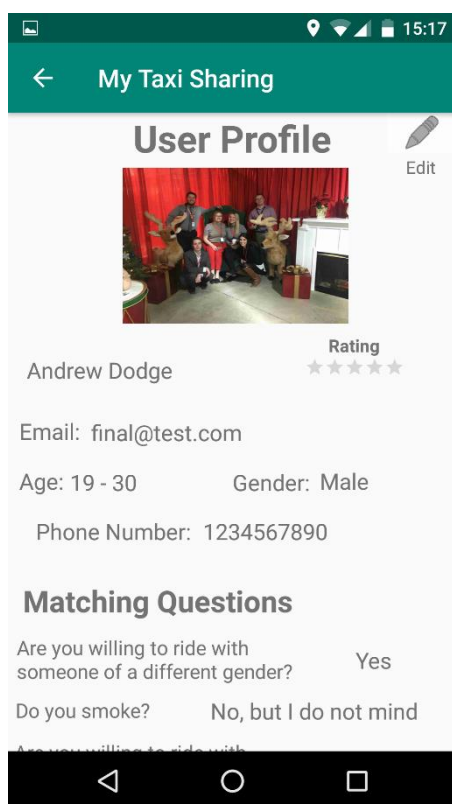
App User Creating Screen



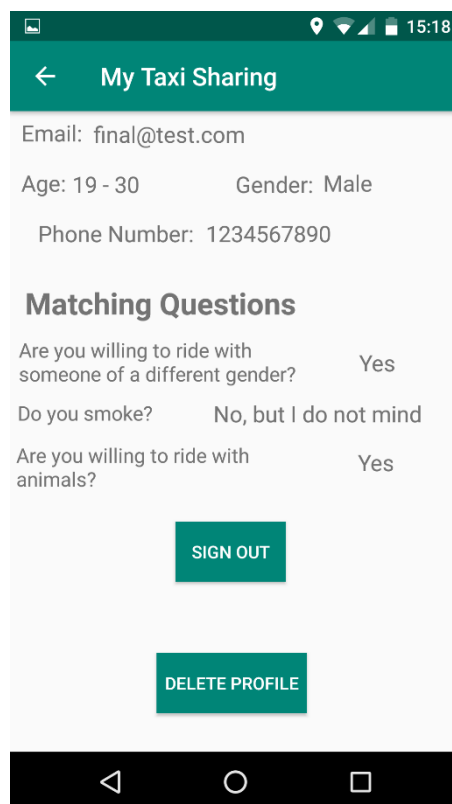
Initial Map Screen



Current Location



User Profile



**My Taxi Sharing**

## Edit Account Information

### Personal Information

Andrew

Dodge

Age Range: 19 - 30

Gender: Male

1234567890

Profile Picture (optional)

**My Taxi Sharing**

## Payment Information

1111222233334444

### Matching Questions

Are you willing to ride with someone of a different gender? Yes

Do you smoke? No, but I do not mind

Are you willing to ride with animals? Yes

RETURN TO PROFILE

Edit User Profile

**Matching Questions**

How much luggage do you have with you? 0

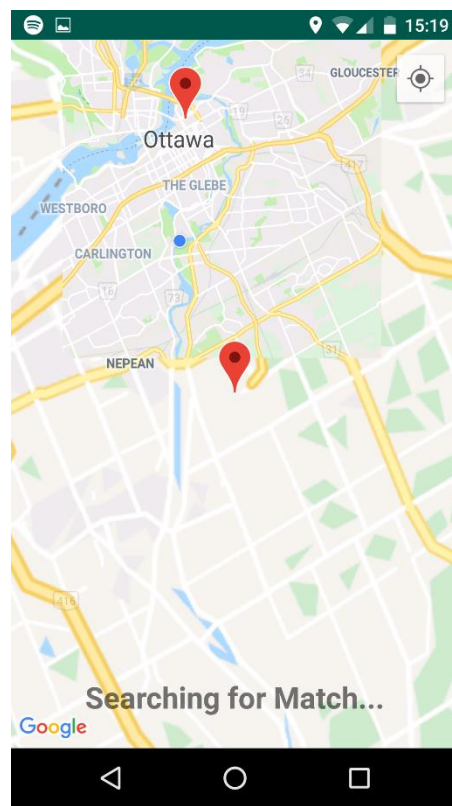
Do you have a pet with you? No

How many people are you willing to ride with? 0

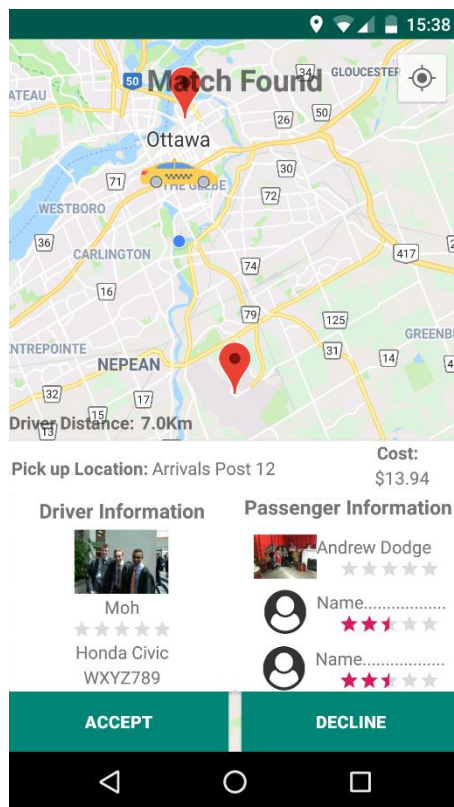
How large is your group? 1

FIND MATCH

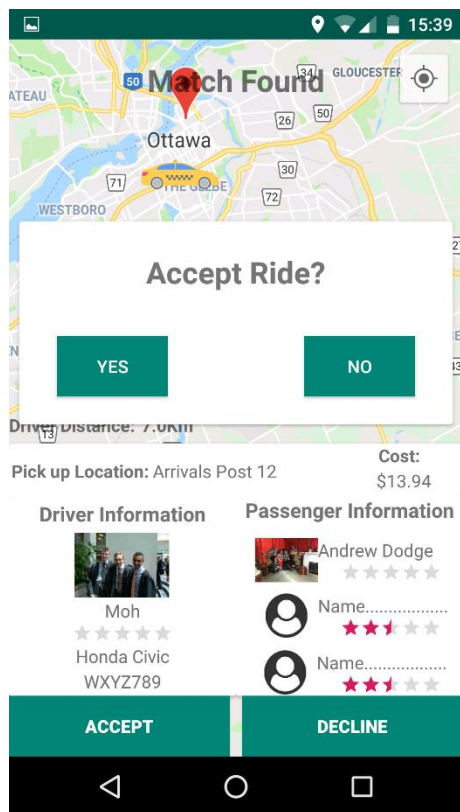
Pre-Match Prompt



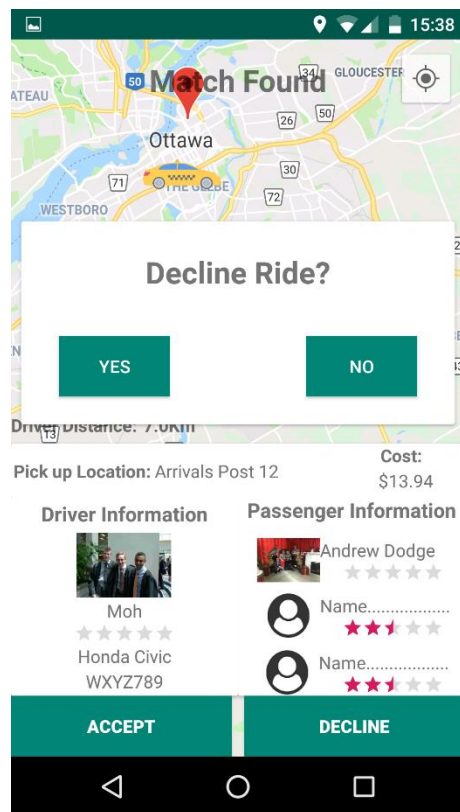
Searching for Match



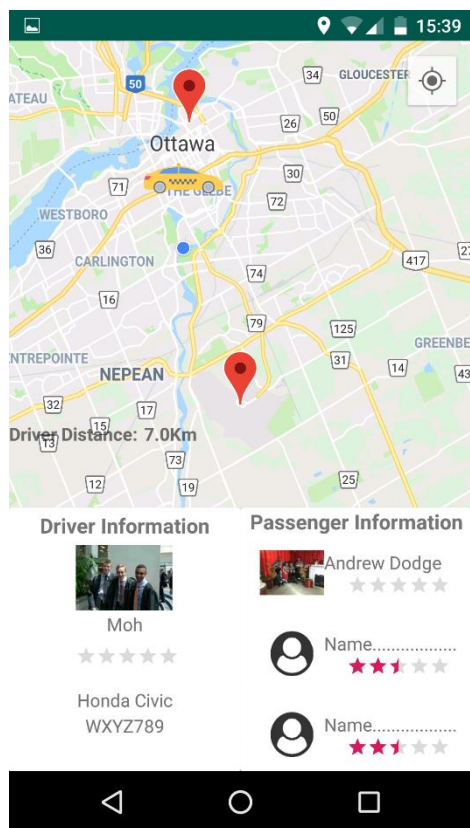
Match Found



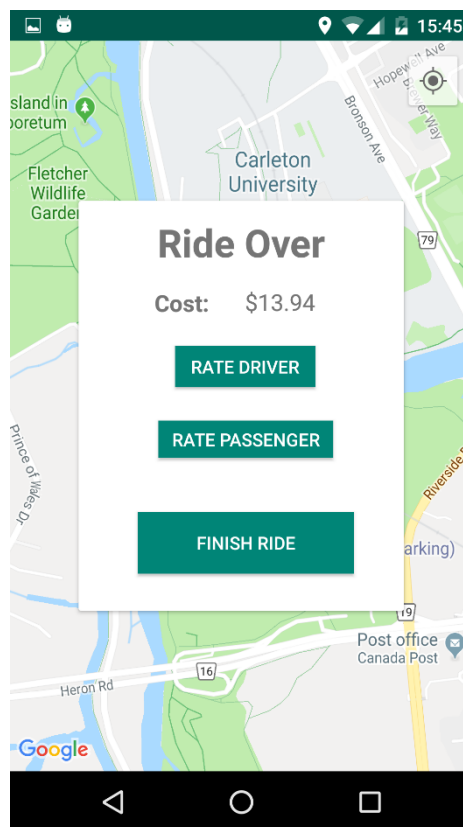
Accept Prompt



Decline Prompt



During Ride



Ride Over