HitchHike

by

Muhammad Moiz Hussain
Adam Kunkel
Robert Kurta
Anubhav Pramanik

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Muhammad Hussain Adam Kunkel Robert Kurta Anubhav Pramanik 04/24/2022
M Hussain, A Kunkel, R Kurta, A Pramanik Date

Yahya M Gilany 04/24/2022
Yahya Gilany Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2022

Table of Contents

List of Illustrations

Abstract

A network of riders and drivers willing to carpool over long distances and share travel costs does not exist. The development of HitchHike started because of this unmet need in the current app market for a rideshare solution that is reliable, affordable, easy to use, and available to everyone. Current options, including the private market and public transportation, leave users underwhelmed with functionality and useability. The development team has integrated an app that is clean and intuitive. Users can post rides that can be used to connect them to various destinations, regardless of the distance. The application's add trip feature allows users to choose their origin and destination, whether they are a rider or a driver. The design features lead from one screen to the next without frustrating the user. Depending on users ' needs, it is easy to search for existing rides or create new rides. By introducing HitchHike into the market, the hope is to connect riders and drivers with common destinations for mutually beneficial travel experiences.

Introduction

Project Summary:

HitchHike allows travelers to search for other people willing to carpool, especially for long-distance driving. Potential users can post details about upcoming trips, including dates, times, origins, and destinations, to find others willing to split travel costs or share other expenses. Both riders and drivers can post trips for others to join. HitchHike will also provide an excellent platform to connect travelers to events where drivers can post their reasons for travel, such as sports competitions or concerts.

Problem Statement:

There is a lack of opportunity for people without a vehicle to reach out to others for long-distance carpooling. Moreover, those driving long distances do not have an option to look for someone to carpool with who could split fuel costs, drive part of the way, or even provide some company for the long drive.

Users could opt to book an Uber to take them from Cincinnati to Chicago, but it would be costly, and it is improbable for a driver to agree to take them there. Having a robust network of drivers and riders frequently carpooling long distances would make it easier for everyone to travel further by road without bearing the cost of a flight. Additionally, it would also be a much more comfortable method of travel than taking a bus. In the case of a global pandemic, it is also much safer than taking public transportation. Finally, it would be great for the environment as fewer cars would be on the road.

There is demand for a long-distance rideshare application in the US. Most of the applications in the market do not work correctly or are limited in their functionality. For example, one of the applications, "Hitchike," that is available, did not have a large enough user base because it was only available in a few select states. Comparing this to the European market, services like BlablaCar already have nearly 40 million users, and UK-based service Liftshare has more than 500,000 (Mordor Intelligence, 2020). Considering that the global ride-sharing market is expected to double by as soon as 2026 (Markets and Markets, 2021), it only makes sense that this need is also addressed in the US. If anything, there is a massive potential for growth in this area.

Solution:

HitchHike is a fully functional Android application that will allow users to connect and share rides to get to their common destinations, regardless of distance. HitchHike is much cheaper than the other ride-sharing solutions currently on the market since the riders and drivers will determine and share the costs. By allowing users to create an itinerary with multiple rides, they will be able to plan out longer trips by scheduling rides ahead of time. Having fewer people in a vehicle allows for better comfort and safety during the trip than public transport. Drivers offering rides can specify if they can offer additional services like trips for disabled riders. HitchHike also reduces the number of vehicles on the road, which will be beneficial for the environment.

Project Source:

In August, we began meeting to discuss ideas and research past expos and how some of those projects progressed. Initially, we developed a list of about fifteen different ideas among all team members; then, we started to narrow our scope by voting on our favorite projects. We all voted on the ride-sharing application that Anubhav proposed as he had trouble traveling long distances since he does not have a driver's license. We realized this was a problem that many people share. Once we had decided on HitchHike, we each began contributing ideas on how to differentiate ourselves from what is currently on the market. We came up with multiple ideas to stand out against the other predominant ride-sharing applications.

Discussion

Project Objectives/Goals:

Below is the outline of all significant features that are part of HitchHike to make the application function as intended.

- Sign-up Page
  - Users enter their email address, phone number, and password to sign up for the first time.
- Landing Page
  - Once users are logged in, they are greeted with a page containing "Rider" and "Driver" buttons to decide what kinds of trips appear on their dashboard.
- Dashboard
  - After entering the application, there is a dashboard where users can see all the available rides.
  - The dashboard contains fields which allow you to search for existing rides based on city specific needs.
  - Users have an “Add Trip” button to add rides from the dashboard.
  - Trips are organized as cards for easy viewing, including essential trip details.
- Add Trip
  - Riders and drivers can add the starting and end location of their rides and the date and time.
  - Users can also provide additional details, such as how many travelers they can accommodate, storage options, and the ability to provide disability access.
- Navigation Menu
  - There are multiple options inside the navigation menu: MyRides, MyProfile, Settings, Help, and About.
  - This allows users to move around the application seamlessly.
- MyRides
  - This page shows users the details of their upcoming/current trips.
  - Trips are organized as cards for easy viewing, including trip details such as origin, destination, date, and time.
  - Once a ride has been scheduled, users can see the details of their ride by going into the details page, displaying helpful information about the ride, and a button to navigate to Google Maps and see a projected route.

Project Scope:

The application aims to solve the problem of long-distance ride-sharing by providing a safe platform for drivers and riders to connect and hitch rides together.

After conducting market research and requirements analysis, the development of the Android application for HitchHike commenced. A web application and an iOS app are not included in the scope for the present time.

HitchHike is designed to be an intuitively easy-to-use application. After finishing the foundation for the application, development continued to include travel details and a search feature to find drivers/riders. HitchHike also conducted user interviews where participants were asked to sign up and trial the application to gather feedback on basic functionality. User feedback was a recurring theme while developing the application, as it is invaluable to make sure the application is easy to use.

Assumptions:

- Users will handle payment.
- Users will use HitchHike at their own risk.

Constraints:

- Processing valid ID (driver's license for drivers) to verify user identities.
- One lead software developer – although the other team members will also be helping with development, there is only one person on the team with formal experience.
- Because of technical limitations, we are focused only on building an Android application.

Quick Project Timeline**:**

Table 1. Project Timeline

Milestone #1	Deliverables: • Add Trip Page • My Profile Design • My Ride Design • Build Server and Database	09/27/2021	10/11/2021
Milestone #2	Deliverables: • Dashboard Page • Database Expansion & Integration • Testing & Validation	10/11/2021	10/25/2021
Milestone #3	Deliverables: • My Ride Functionality • Testing & Validation	10/25/2021	11/08/2021
Milestone #4 (Presentation Week)	Deliverables: • Finalize Prototype • Presentation Prep • Testing & Validation	11/08/2021	11/22/2021
Milestone #5	Deliverables: • My Profile Functionality • Testing & Validation	11/22/2021	12/06/2021
Milestone #6 (Winter Break: 12/10/2021 to 01/10/2022)	Deliverables: • Testing & Validation	12/06/2021	01/10/2021

Milestone #7	Deliverables: • Sign Up/Login Page • Testing & Validation • User Feedback	01/10/2022	01/24/2022
Milestone #8	Deliverables: • Testing & Validation • User Feedback	01/24/2022	02/07/2022
Milestone #9	Deliverables: • Google Maps Tie In • Testing & Validation • User Feedback	02/07/2022	02/21/2022
Milestone #10	Deliverables: • Full Product Integration • Testing & Validation • User Feedback	02/21/2022	03/07/2022
IT Expo Preparation	Deliverables: • Testing and Bug Fixing • Preparation for Final Presentation at the IT Expo	03/07/2022	IT Expo

Technologies Used**:**

There was much discussion regarding the technologies to use for the project. For example, the initial plan was to use Microsoft Azure to meet database needs; however, upon discovering Google Firebase, it was clear that Firebase was a more suitable solution for the project. Ultimately, the following technologies were implemented.

- Android Studio & Kotlin - Our application was developed using Android Studio IDE and Kotlin as the programming language.
- GitHub - GitHub was used for version control and collaboration on the project as we developed different pages and features.
- Google Firebase - Google Firebase is utilized to store user data for the application. Since the project is Android-based, it was easy to natively connect Google Firebase and its associated services with Android Studio. Initially, we proposed using a SQL server to store data; however, Google's Realtime Database was more viable for our project after further research. It also incorporates built-in Firebase Authentication through Email, Facebook, or phone numbers.

Technical Architecture Diagram:

Workflow Diagram:

The workflow diagram below represents the steps an average user completes when utilizing our application. The diagram includes the major functions that the app will perform. Hitchhike starts from a login page that asks the user to decide between "Rider" and "Driver." Based on this decision, the user is directed to the dashboard page of the application. A user can see all the upcoming rides on the dashboard and search for different rides using the filter option. If the user finds a ride, they can request to schedule that particular ride which sends a notification to the ride owner. If the user does not find any ride, they can add a ride using the "Add Trip" feature by including the origin, destination, date, time, and other details.

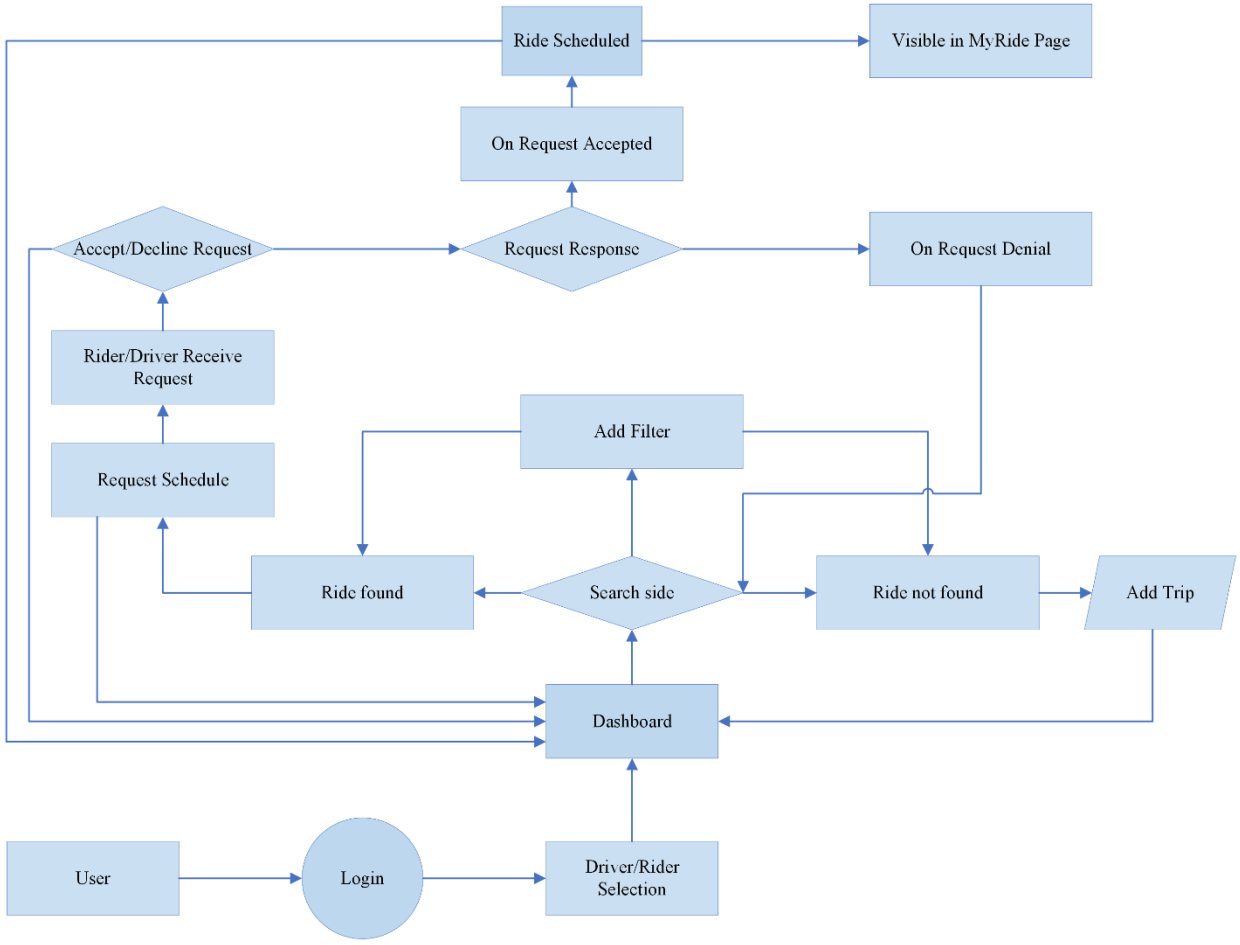

Figure 1. Workflow Diagram

Logical Diagram:

The logical diagram below depicts the data flow from a user's smartphone to the Google Firebase application platform. When information is added to the application, it is sent via cell towers or a local internet connection to the Google Firebase stores. This includes both user authentication and additions to the HitchHike database.

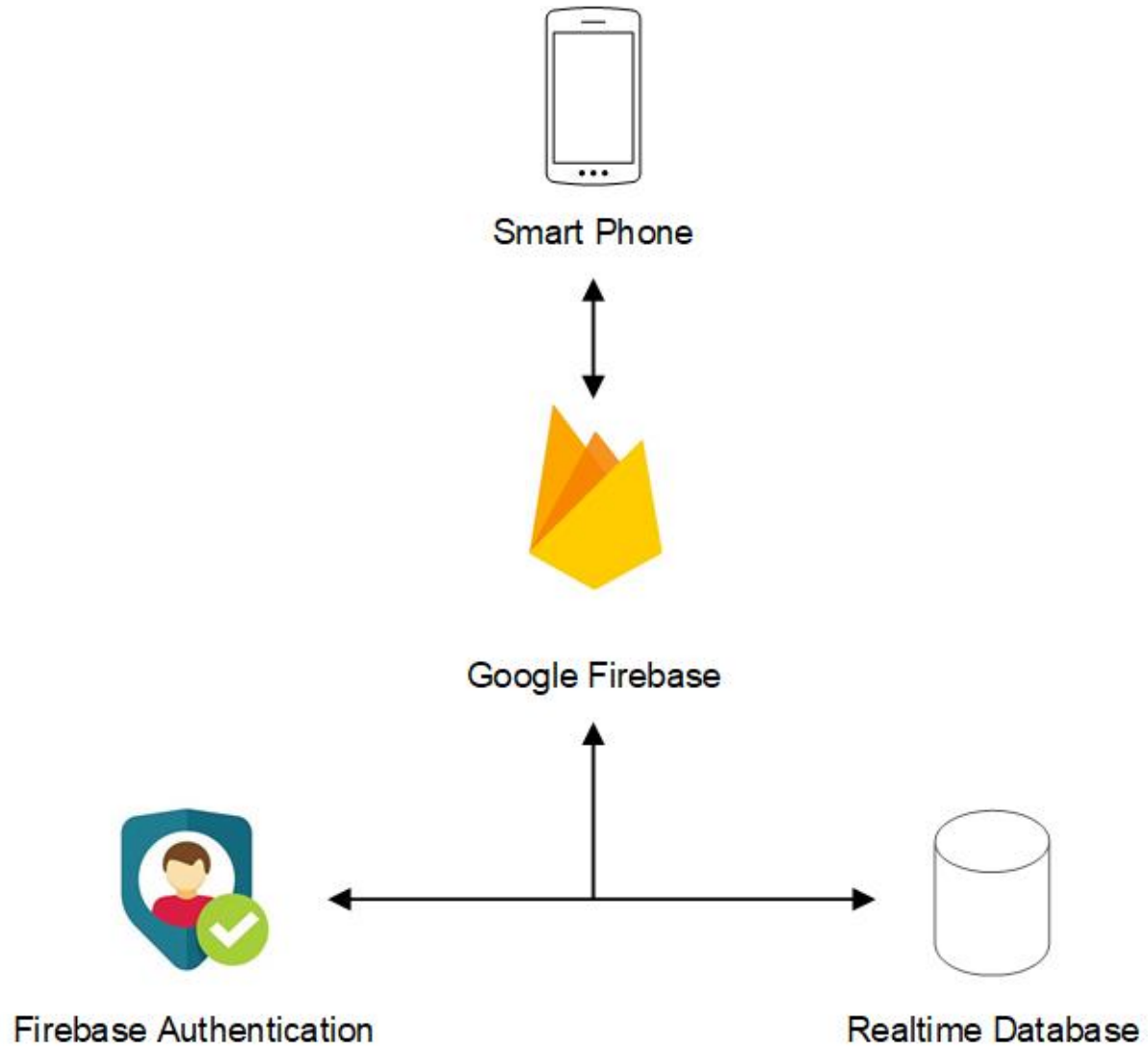

Figure 2. Logical Diagram

User Personas:

Table 2 User Persona 1

User Persona: 1 Rider	
	Undergraduate Student Bill Clinton 21 Male
Behavior	• Bill lives close to a college campus and enjoys traveling to visit family and friends.
Pain	• While attending a university, he does not have the additional income to be able to afford a personal vehicle or the maintenance that would encompass that vehicle. • He must use public transportation or request an uber, which can be unreliable, expensive, and time-consuming.
Needs & Goals	• Bill wants to visit his friends and family. • He still wants to travel outside his immediate college campus area affordably.

Table 3. User Persona 2

User Persona: 2 Driver	
	Long-Distance Driver Leslie Pickett 28 Female
Behavior	• Leslie frequently drives long distances to see her parents.
Pain	• Since Leslie travels so frequently, long distances can be tiring and boring for her. • Driving long distances increases her fuel and maintenance costs for her vehicle.
Needs & Goals	• Leslie wants to find other companions to help save on gas money. • Leslie wants to find other companions to make travel more exciting and fun.

Use Cases:

Table 4 Use Case 1

Use Case ID	UC1_SignUp
Use Case Name	Sign Up
End Objective	This function allows a user to sign up.
User/Actor	Rider/Driver
Trigger	User launches the application.
Frequency of Use	On opening the application for the first time.
Preconditions	1. User downloaded the app.
Basic Flow	1. User enters name. 2. User enters Email. 3. User enters password. 4. User enters phone number. 5. User clicks submit button. 6. System confirms the account creation.
Alternate Flow	AF1 Network error. 1. Network connection has been lost. 2. Restart from basic flow step 1.
Postconditions	User successfully created an account and is redirected to the dashboard.

Table 5. Use Case 2

Use Case ID	UC2_Login
Use Case Name	Login
End Objective	This function allows a user to log into the application.
User/Actor	Rider/Driver
Trigger	User launches the application.
Frequency of Use	Every time user opens the application.
Preconditions	2. User downloaded the app and signed up. 3. User must have an account.
Basic Flow	7. User enters Email. 8. User enters password. 9. User clicks login button. 10. System confirms the user credentials.
Alternate Flow	AF1 User does not have an account. 1. User creates an account. 2. System confirms account creation. 3. User redirects to the login page. AF3 Network error. 3. Network connection has been lost. 4. Restart from basic flow step 1.
Postconditions	Successful user login and redirection to the dashboard.

Table 6. Use Case 3

Use Case ID	UC3_Search_Trip_Dashboard
Use Case Name	Search Trip
End Objective	In this case, an actor searches for an existing trip.
User/Actor	Rider/Driver.
Trigger	User launches the application.
Frequency of Use	Occurs each time a rider wants to travel with a driver or vice versa. Most likely used for longer trips but can be used for short trips.
Preconditions	User should be able to log in and search for a ride.
Basic Flow	1. User logs into the application. 2. User is redirected to the HitchHike dashboard. 3. The app automatically shows a list of available local rides. 4. The rider enters origin and destination locations in the search bar. 5. The search brings up all available rides.
Alternate Flow	AF1 User logged off the application. 1. User redirected to login screen (Basic flow step 1). 2. User adds credentials to log back in. 3. System confirms the user. 4. User redirects to the dashboard (Basic flow step 2). AF2 User does not have an account. 1. User creates an account. 2. System confirms account creation. 3. User redirects to the dashboard (Basic flow step 2). AF3 Ride has not been created. 1. User can add ride information using the add trip functionality.
Postconditions	The user found a ride and sent a schedule request.

Table 7. Use Case 4

Use Case ID	UC4_Add_Trip_1
Use Case Name	Add Trip
End Objective	This function allows users to add a trip for other users to search.
User/Actor	Rider/Driver
Trigger	User launches the application to the dashboard and clicks on the “Add Trip” button.
Frequency of Use	Potentially each time a user takes a trip. It could be a weekly occurrence or more for shorter trips.
Preconditions	1. User must have an account and be able to log in. 2. User will be making a trip and is willing to take riders.
Basic Flow	1. User logs into the application. 2. User is redirected to the HitchHike dashboard. 3. The app automatically shows a list of available local rides. 4. User clicks the "Add Trip" button. 5. User enters all the required information for their trip.

	6. Click "Submit" to add a trip to the dashboard. 7. User is redirected to the dashboard.
Alternate Flow	AF1 User logged off the application. 1. User redirected to login screen (Basic flow step 1). 2. User adds credentials to log back in. 3. System confirms the user. 4. User redirects to the dashboard (Basic flow step 2). AF2 User does not have an account. 1. User creates an account. 2. System confirms account creation. 3. User redirects to the dashboard (Basic flow step 2). AF3 Network error. 1. Network connection has been lost. 2. Restart from basic flow step 2.
Postconditions	The trip has been added to the database. Users are now able to see this trip in a dashboard.

Table 8. Use Case 5

Use Case ID	UC5_Request_Schedule
Use Case Name	Request trip
End Objective	In this case, an actor requests to schedule an already existing trip.
User/Actor	Rider/Driver.
Trigger	User launches the application and requests to schedule an existing ride.
Frequency of Use	Occurs each time a user finds a trip and needs to schedule it.
Preconditions	1. User should be able to log in and search for a ride. 2. User should be able to find a ride.
Basic Flow	1. User logs into the application. 2. User redirected to dashboard. 3. User searches and finds a suitable ride. 4. User requests to schedule that ride.
Alternate Flow	AF1 User logged off the application. 1. User redirected to login screen (Basic flow step 1). 2. User adds credentials to log back in. 3. System confirms the user. 4. User redirects to the dashboard (Basic flow step 2). AF2 User does not have an account. 1. User creates an account. 2. System confirms account creation. 3. User redirects to the dashboard (Basic flow step 2). AF3 Request denied. 1. User request has been denied.

	2. Redirected to Basic flow step 3.
Postconditions	User has requested to schedule a ride.

Table 9. Use Case 6

Use Case ID	UC6_Accept_Decline_Request
Use Case Name	Accept Decline Request
End Objective	In this case, an actor accepts or declines the requested ride.
User/Actor	Rider/Driver.
Trigger	User receives a schedule request and opens HitchHike to accept or decline a ride.
Frequency of Use	Occurs each time a user receives a ride notification.
Preconditions	1. User should be able to log in. 2. User should be able to see the requested ride.
Basic Flow	1. User logs into the application. 2. User redirected to dashboard. 3. User opens the notifications tab. 4. User accepts a ride. 5. Requesting user receives a notification of ride confirmation.
Alternate Flow	AF1 User logged off the application. 1. User redirected to login screen (Basic flow step 1). 2. User adds credentials to log back in. 3. System confirms the user. 4. User redirects to the dashboard (Basic flow step 2). AF2 User does not have an account. 1. User creates an account. 2. System confirms account creation. 3. User redirects to the dashboard (Basic flow step 2). AF3 Request denied. 1. User denies the requested ride. 2. Redirected to Basic flow step 3.
Postconditions	User accepted or declined the requested ride.

Table 10. Use Case 7

Use Case ID	UC7_MyRide
Use Case Name	My Ride
End Objective	In this case, an actor can view the scheduled rides.
User/Actor	Rider/Driver.
Trigger	User checks the MyRides page.
Frequency of Use	Occurs when the user wants to view the schedule.
Preconditions	3. User should be able to log in. 4. User should be able to schedule a ride.
Basic Flow	6. User logs into the application.

	7. User redirected to dashboard. 8. User opens MyRides page. 9. User can view all scheduled rides.
Alternate Flow	AF1 User logged off the application. 5. User redirected to login screen (Basic flow step 1). 6. User adds credentials to log back in. 7. System confirms the user. 8. User redirects to the dashboard (Basic flow step 2). AF2 User does not have an account. 4. User creates an account. 5. System confirms account creation. 6. User redirects to the dashboard (Basic flow step 2).
Postconditions	Schedule confirmed

Use Case Diagram:

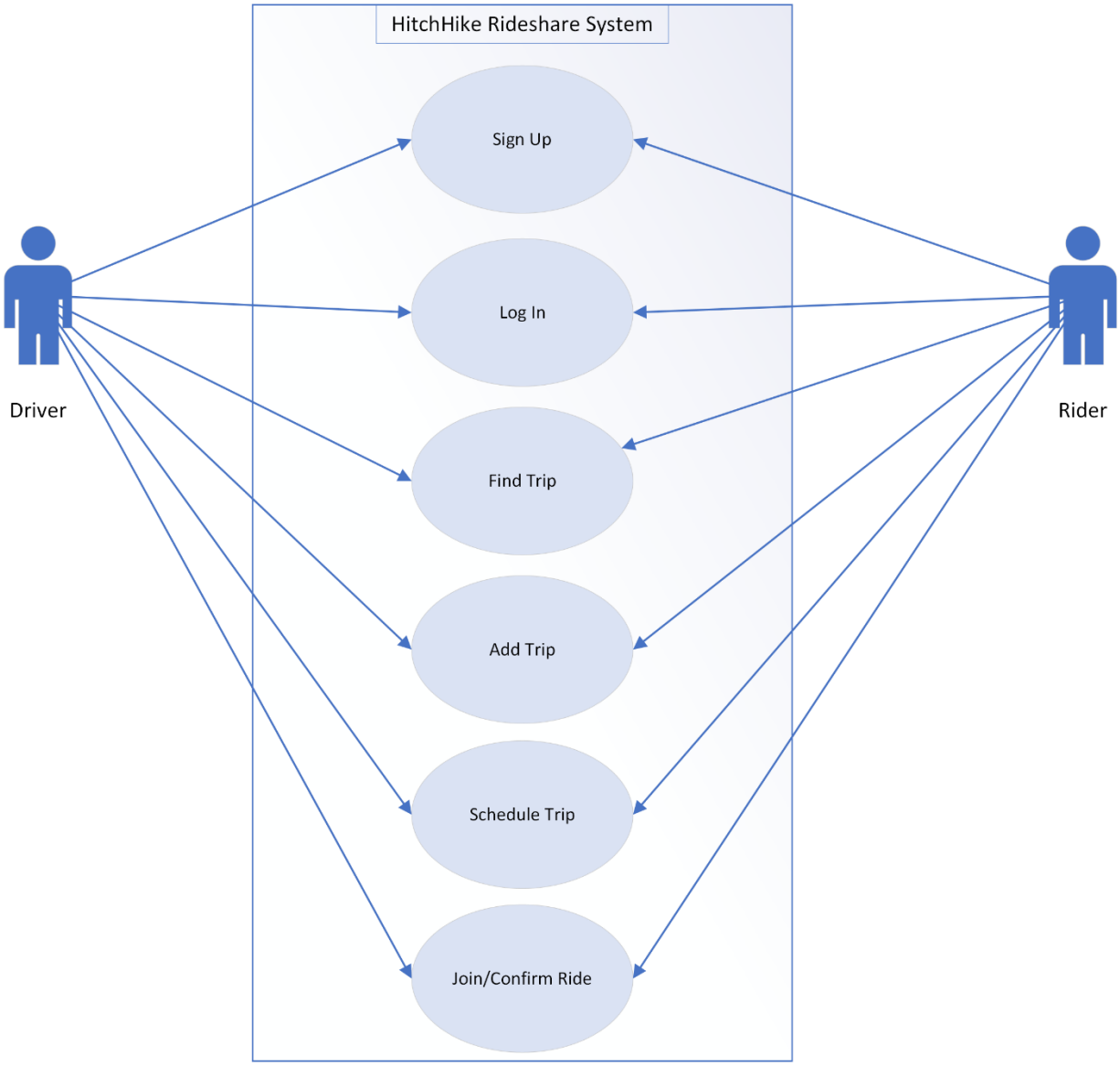

Figure 3. Use Case Diagram

Testing Plan:

Overview

The testing plan for HitchHike involves participants attempting to complete a given task on an android phone, which has the current version of the application. All tests will be based on Use Cases that can be explored in real-time by a user on a laptop. The amount of time taken to finish the required steps will be recorded along with other observations to determine the app's usability.

Methodology

Using the application on an Android phone gives a realistic look and feel of the application to the user, increasing the accuracy of results and observations. After the basic structure of HitchHike was completed, testing began at the end of each sprint. Test scenarios were pulled from the Use Cases defined earlier in this report. This allowed constant development and refining of the app's usability and overall experience. Participants performed tasks while "thinking aloud" so observers could understand both their actions and thought processes. They were then asked a series of questions to determine HitchHike's strong and weak points. Questions and results utilized both the Likert scale and open-ended questions to determine the success of current app functionality and better understand what is missing or ambiguous. As a result, functionality and design were tested simultaneously, so multiple iterations of data were available to improve the application throughout the entire development cycle.

Scope

Participants tested the above-mentioned use cases because those are the main functions of HitchHike. "Sign_Up" was tested first, as it is the logical starting point and will allow users to create their accounts. The account is then tied to any information added in subsequent tests. The "Login" use case ensured that any registered user could successfully log in to our application. The data entered during the "Add Trip" process populates the dashboard upon login. Participants then searched for an existing trip using the "Search Trip" use case to find any available rides. Once they find a suitable ride, they can request to schedule that ride by following the "Request Schedule" use case. Riders or drivers can then receive confirmation about their schedule request once they complete the "Accept Decline Request" use case. These actions encompass the functionality of HitchHike and give developers data to analyze interactions between participants and the application.

Objectives

The following criterion helped us determine the functionality and useability of HitchHike in a real-world scenario.

- Users can create new accounts, log in to their accounts, add trips, and search for trips.
- Use cases involving data input or trip retrieval are tested thoroughly by different demographics to improve the app's usability for future tests.
- Test data is compiled and analyzed to determine necessary changes in each sprint.
- User permissions and data security are tested to ensure all participants can complete tasks while their data is protected.

- Participant testing continues through all sprints until all features of HitchHike have been implemented.
- The final iteration is for fixing bugs and adding peripheral functions.

Test Logs and Procedures

Table 11. Test Logs and Procedures

Sign_Up

Test Case #	Participant	Role	Expected Output	Actual Output	Pass/Fail	Reason for Pass/Fail	Date
1	1	Driver	Participant successfully creates a HitchHike account.	Participant successfully created a new account.	Pass	Account created without any significant issues in 1 minute and 30 seconds.	1/19/2022
2	2	Driver	Participant successfully creates a HitchHike account.	Participant successfully created a new account.	Pass	Account created in 3 minutes and 45 seconds. Participant tried to sign in with an existing Email before creating an account. Did not notice "create new account" option.	1/20/2022
3	3	Driver	Participant successfully creates a HitchHike account.	Participant successfully created a new account.	Pass	Account created in 5 minutes and 52 seconds. Participant tried to sign in with an existing Email before creating an account. Could not see "create new account" option. Had trouble with the computer interface and needed help to complete the task.	1/20/2022
4	4	Driver	Participant successfully creates a HitchHike account.	Participant successfully created a new account.	Pass	Account created without any significant issues in 1 minute and 12 seconds.	1/20/2022

Login

Test Case #	Participant	Role	Expected Output	Actual Output	Pass/Fail	Reason for Pass/Fail	Date
5	1	Driver	Participant can successfully login to account.	Participant successfully logged into account.	Pass	Logged in successfully in 15 seconds.	1/19/2022
6	2	Driver	Participant can successfully log in to account.	Participant successfully logged into account.	Pass	Logged in successfully in 42 seconds.	1/20/2022
7	3	Driver	Participant can successfully log in to account.	Participant successfully logged into account.	Pass	Logged in successfully in 47 seconds.	1/20/2022
8	4	Driver	Participant can successfully log in to account.	Participant successfully logged into account.	Pass	Logged in successfully in 12 seconds.	1/20/2022

Add_Trip

Test Case #	Participant	Role	Expected Output	Actual Output	Pass/Fail	Reason for Pass/Fail	Date
9	1	Driver	Participant can successfully add a trip.	Participant successfully added a trip.	Pass	Trip added in 1 minute and 20 seconds. Participant clicked "Rider" instead of "Driver" and asked about including the city and state.	1/19/2022
10	2	Driver	Participant can successfully add a trip.	Participant successfully added a trip.	Pass	Participant was able to add a trip in 3 minutes 42 seconds despite being confused about whether to add a state. They also just picked the current time for the ride.	1/20/2022

11	3	Driver	Participant can successfully add a trip.	Participant would not have been able to add a trip without assistance.	Fail	The participant could add a trip in 6 minutes and 5 seconds with direction. They did not understand the layout or interface of the dashboard and did not see the add trip button.	1/20/2022
12	4	Driver	Participant can successfully add a trip.	Participant successfully added a trip.	Pass	Trip added in 1 minute and 30 seconds. Participant asked about adding the city and state.	1/20/2022

Search_Dashboard

Test Case #	Participant	Role	Expected Output	Actual Output	Pass/Fail	Reason for Pass/Fail	Date
13	1	Driver	Participant can successfully find the trip they created in the previous test.	Participant found their trip.	Pass	Participant was able to find their trip in 1 minute and 5 seconds. This took a little longer than expected because the user was unaware that the search was case-sensitive.	1/19/2022
14	2	Driver	Participant can successfully find the trip they created in the previous test.	Participant would not have found their trip without assistance.	Fail	With direction, the participant was able to find their trip in 7 minutes and 10 seconds. Several issues caused this failure.	1/20/2022
15	3	Driver	Participant can successfully find the trip they created in the previous test.	Participant found their trip.	Pass	Participant was able to find their trip in 1 minute and 43 seconds.	1/20/2022

16	4	Driver	Participant can successfully find the trip they created in the previous test.	Participant found their trip.	Pass	Participant found their trip in 20 seconds. They initially found the trip by scrolling down the dashboard, so they were asked to use the search. They quickly found the trip with the search function also.	1/20/2022

Likert Scale Discussion

After testing, participants were asked to rate the overall experience of using HitchHike on a scale of 1 to 5. The questions and answers can be viewed at the end of the report in the Appendix. Responses indicated a positive overall experience with an average score of 3.5. Users were also impressed with the application's appearance, with an average score of 4.75. Reactions to the useability of the application also indicated that it was positive, but it could be improved upon, with an average score of 3.75. When participants were asked whether it was confusing to use the application, responses indicated that useability was predominantly straightforward, with an average score of 2. However, it would be ideal to address any confusion and ambiguity moving forward. Users had an above-average response when they were asked about the likelihood of them using the application, with an average score of 2.75.

Open-Ended Questions

Question	P1	P2	P3	P4
What would you change about HitchHike?	More organization and separation of trips.	Larger print. Some sort of instructions.	Larger print with more obvious instructions and buttons.	Add an option to pick a state when adding a trip and searching for a trip.
What did you like most about the app?	Straight-forward and not too busy.	Simple design.	Logging in was easy.	Page design.
What did you like least about the app?	No instructions or tips.	Searching the dashboard was unclear.	Using a computer instead of a phone made it harder to use.	The buttons were a little ambiguous.

Testing Review

The first testing phase for HitchHike revealed more than anticipated. The application has great functionality for those with little technological experience, but our use cases proved complicated for novice users. Three out of four tasks confused participants, with two of those returning task failures. Even amongst the experienced users, there were periods with long intervals between progress toward their end goal.

In our Sign_Up use case, all users agreed that the link from our login page to the sign-up page was too subtle. The link should be more prominent and easier to read, and an underline would indicate that it is a link to the described page. These fixes, combined with a little more background regarding the app by the observer, may have improved statistics regarding the sign-up process. Participants also suggested that a more prevalent popup with password complexity would have made the process quicker and easier. Simple instructions on all of our pages would enhance usability and increase customer success and satisfaction.

The second use case is the app's Login function. This was a smooth process, and all participants completed this task quickly. There are few variables on this page, and the simplicity made the interaction easy.

HitchHike's primary function is the ability to create trips. There were multiple errors on this page for all participants. The first problem was finding the "Add Trip" button on the dashboard. The application has already been remediated to reflect more indicative color schemes for buttons. The next major problem was understanding the format for entering the origin and destination cities. Each participant was a little confused about how to address adding a state. Currently, there is no necessity for adding a state, but that does not make sense considering many states share multiple city names. The app was changed so the "State" option was available in a drop-down menu to prevent spelling and formatting issues, but that did not end all confusion. An API for autocomplete was the best solution. This also aided in the subsequent use case, searching the dashboard for trips. Other options were a little ambiguous to the user. One person registered as a rider instead of a driver, and another chose the current time instead of picking an appropriate time to schedule their ride. These errors have been mitigated by separating driver functions from rider functions to create a separate menu for them and ending the confusion. The description field was also not used according to the app's intended purpose. This field has been used throughout development to describe payment options. During testing, this field was used by participants to describe the trip, which did not provide any helpful information.

Search_Dashboard is the last use case tested in the first phase. This is where participants can search the dashboard for the trip they just created. As with the Add_Trip use case, there were multiple errors during testing. The biggest problem was formatting and case matching of the trip origin and destination fields. Ensuring that the data input is not case sensitive would mitigate this problem. The "SUBMIT" button should also be changed to "SEARCH" to describe the action that needs to take place. As with the "Add Trip" function, an autocomplete API was the best way to fix the problem.

General issues regarding familiarity with computers and smartphones were also prevalent. Participants did not realize they could scroll on the simulator, and one even minimized the app and could not continue without assistance. Using an actual phone instead of a computer simulator would aid some users in understanding the interface. Participants unanimously agreed that the general design of HitchHike was simple and easy to understand. This round of testing brought many flaws to the surface, and most of the remediation was easy. Simply changing outlining and color schemes went a long way to increase the usability of HitchHike.

Change Management Plan:

Change Request Process

Project owners and stakeholders can submit a change request. Changes can be proposed through written requests. If a user of the application wants to request a modification, a forum will be provided, which they can use to detail their concerns to the development team for consideration.

Triage Process

The change will determine the triage process. If the revision that is being proposed is a design or programming alteration, software developers will determine whether it is necessary. Similarly, the database administrators will handle adjustments to the database and data support. Lastly, security analysts will be responsible for overseeing improvements in application security. Once the team is notified of the proposed changes, time will be taken during weekly meetings to determine whether they can be implemented or not. The severity of the problem for which that change is proposed will determine the triage process in descending order of magnitude for all approved adjustments.

Approval Process

During the discussion process for a proposed change, whether the improvement is necessary will be determined. If the team decides that a change is essential for the project to move forward, it will be implemented. However, any revisions that are deemed unnecessary at that point in time can be shelved for later discussion. If the change is not approved, the requester will be notified within a reasonable amount of time about their suggestion and will be given a reason for the denial.

Levels of Criticality

The levels of criticality are low, medium, and high.

- Low: These are the changes that will not significantly impact our schedule of development. These could be modifications such as adjusting the colors or the font on a specific page we showcase during the standup meetings. Depending upon the time it takes to make those changes, they may be implemented immediately or will be placed on the backlog.

- Medium: These are the changes that will impact user experiences. There might be an interaction within the application that is confusing users. For example, the first edition of the dashboard included a "SUBMIT" button that users found confusing. When the button was changed to "SEARCH," it was more functional. New features deemed as medium impact will be handled on a scheduled basis, including security and software updates.

- High: These are the changes that will significantly impact the project. These changes will require the group to meet and discuss the problem immediately. An example of this change could be a potential loss of connection between the application and database, which locks the user out of the application. High criticality changes will be implemented as soon as a fix is developed.

Communication Plan

All changes that must deal with the application development side of matters will be notified to the stakeholders based on the criticality of the problem. If the problem is of high criticality, the stakeholders will be emailed immediately, along with the steps taken to remediate the problem. The stakeholders will be notified through bi-weekly project meetings of all other changes. All changes will be published in a weekly Email that will be provided to the stakeholders to track changes through the project's development.

Budget

The budget for HitchHike during development was very minimal because no outside personnel was hired, and the hosting platform is free up to specific data usage amounts resulting in virtually no expenses. All necessary equipment and software were already purchased and available to the app creators; however, the budget accounts for these expenses if somebody was to start this from scratch. The budget also accounts for the wages of different roles pertaining to the app's development and maintenance, such as Lead Developer, Developer, Database Administrator, and Systems Administrator. Once the app crosses the data usage threshold, the hosting platform will no longer be free, and a monthly fee will be charged for specific services. We will also budget for the marketing required to get new users for the application prior to market launch.

Table 12. Budget

Estimated Cost Rough Order of Magnitude					
Labor	Position	Rate/hr	Hours/wk	Project Duration (in weeks)	Total
	Lead Developer	$50	20	15	$15,000
	Developer	$45	20	15	$13,500
	Cybersecurity Specialist	$40	20	15	$12,000
	Database Administrator	$40	20	15	$12,000
					Combined Total
					$52,500

Consulting	Discipline	Rate/hr	Hours		Total
	Marketing	$100	10		$1,000
	Firebase Support	Free			$0
					Combined Total
					$1,000

Software	Name	Rate/yr	Users	Total
	Outlook 365	$150	4	$600
	Android Studio	Free		$0
	Github	Free		$0
	Google Firebase	Free		$0
				Combined Total
				$600

Hardware	Name	Cost	Users	Total
	Laptop	$1,000	4	$4,000
				Combined Total
				$4,000

Ongoing Annual Labor	Position	Rate/hr	Hours/wk	Project Duration (in weeks)	Total
	Lead Developer	$50	10	52	$26,000
	Developer	$45	10	52	$23,400
	Cybersecurity Specialist	$40	10	52	$20,800
	Database Administrator	$40	10	52	$20,800
					Combined Total
					$91,000

Ongoing Annual Software	Name	Rate/yr	Users	Total
	Outlook 365	$150	4	$600
	Android Studio	Free		0
	Github	Free		0
	Google Firebase	Free		0
				Combined Total
				$600

Total Initial Cost	$58,100
Total Annual Cost	$91,600
5 Year Total Cost	$516,100

Problems Encountered and Analysis of Problems Solved

As with any project, numerous challenges required significant troubleshooting and research to reach the level of functionality necessary for the application to meet the users' needs. The following paragraphs detail the major issues for the initial phases of the development of HitchHike.

Our first and most significant problem was one software developer on a software development project. Our team consisted of one software developer, two cybersecurity specialists, and one systems/network administrator. When developing an app, it is common practice to have multiple developers. We had to change direction from the start to keep up with programming needs. All team members contributed to the coding of the application, which meant that three of the team members had to learn how to code in Kotlin from scratch. While this was working and the app was progressing well, it took significant time to learn Kotlin and troubleshoot coding issues.

Lack of industry knowledge led to significant wasted time. As stated in prior sections, our initial setup for a database was an SQL server. This would have required an API to tie into the app. After searching for better solutions, we found Firebase. Researching how to incorporate Google Firebase into Android Studio was time-consuming. While Firebase offers a robust and full-scale solution for app development, including authentication and data storage, none of the team members had any experience on the platform prior to using it for HitchHike. This caused issues with the Firebase Realtime Database resulting in the app not correctly exchanging data with the database. Troubleshooting was done by creating multiple projects in Firebase and connecting them to the Android Studio to determine the reason behind the problem. This was solved by deleting the existing project from Firebase and creating a new one from scratch.

Different schedules and physical locations were a challenge. Our team did an excellent job of communicating through Teams and contributing when available. We were working other jobs while attending school, so meeting in person was out of the question from the beginning. Teams provided a platform for virtual meetings but finding times when all four of us were available was tricky. We often worked on parts of the project separately and met to tie the different perspectives and sections together.

Another problem that we faced this semester was that every team member worked on a different Android Studio activity to complete their respective tasks. Although it did not result in any significant issues, it took much time to merge all the projects into a single GitHub repository. Moreover, we initially had issues logging in to GitHub using Android Studio. The team members had to generate tokens added to the Android Studio project to enable the connection with GitHub to solve this issue.

We encountered several inconveniences related to Covid-19. Group members were sick from the virus early in the semester and could not participate to their full extent. The surge of Covid cases in December and January caused difficulty in international travel that ultimately led to postponed flights; team members were unable to return to the United States at their planned times due to air travel restrictions.

Conclusion

There were ample learning opportunities throughout the semester that allowed the team to learn a breadth of new software and application development skills. One of the most significant skills learned was coding in Kotlin in a short amount of time while trying to develop the application actively. We also learned that communication is vital for any group to get work done efficiently. At the beginning of the semester, we typically compartmentalized the work and split things up evenly between the group members; however, as we got deeper into the semester, we started to come together and work on things as a team. This allowed us to get assignments done quickly and brainstorm with each other. Troubleshooting was easier since subject matter experts were always available to figure out the best solutions. Documentation became simpler because we had each other's perspectives on proper wording and order regarding reports and papers.

Many of us were new to application development, so we learned many new technologies and skills, starting with Android Studio. We learned how to navigate between different sections of the application and use the emulator to test the application. We also learned how to debug our code to troubleshoot any issues properly. Secondly, using Kotlin instead of Java was a new learning experience for all of us, and we learned the commands and syntax related to developing in that language. We learned how to create classes and work according to Object-Oriented Programming. Another considerable learning experience for the group was Google Firebase. Initially, we would use a traditional SQL database; however, upon further research, we found Google Firebase, which tied into Android Studio natively and allowed for a lot of customization and compatibility. We learned how to create a real-time database and connect it to our application. Lastly, for several of our members, GitHub was a new experience. Connecting the repository to Android Studio was a little challenging. However, we solved the problem and got everyone working on the same project and have access to a shared repository.

The existing features of HitchHike will allow it to stand out among existing market applications, particularly the dashboard, which is well organized and easy to use. Users can see all available ride options immediately upon logging into the application. Supporting features were developed for seamless transitions from page to page within the application creating a visually appealing and enjoyable user experience.

References

Top Market Reports. (2021 June). *Ride Sharing Market.* **https://www.marketsandmarkets.com/Market-Reports/mobility-on-demand-market-198699113.html**

Mordor Intelligence. (2021). *Global Ridesharing Market -Growth, Trends, Covid-19 Impact, and Forecasts (2022-2027).* **https://www.mordorintelligence.com/industry-reports/ridesharing-market.**

Appendix

Likert Scale Questions

Questions	P1	P2	P3	P4	Average
How would you rate the overall experience using HitchHike?	4	3	3	4	3.5
How do you like the appearance of the app?	5	5	4	5	4.75
Was the app easy to use?	4	4	3	4	3.75
Did you find HitchHike confusing?	1	3	3	1	2
How likely are you to use this app?	3	2	3	3	2.75