CS 491 Senior Design Project I

Analysis Report

Project short-name: Pigeon's Map

Berdan Akyürek 21600904

Ömer Olkun 21100999

Tanay Toksoy 21703919

Abdullah Ayberk Görgün 21201986

Ekin Üstündağ 21602770

Supervisor: Özcan Öztürk

pigeon-s-map.github.io

1. Introduction

Pigeon’s Map is a route planning android application project, which aims to provide the best route for the people who have to handle more than one address to reach in a time-saving way. In daily life or business life, most people, mostly postal workers, may have to arrive at their final destination by visiting multiple stops. The reason for that is the lower possibility to find the most efficient path to use, waste of time and effort, and fuel consumption if any vehicle is used is inevitable. The application ensures users the shortest route for their destinations. Therefore, the waste of time, effort, and energy will be minimised by using the application.

Pigeon’s Map will have two main pages. These are the creation of the route and the map view.

To use the application, the user must create a destination to reach, indeed. The creation of the route page allows the determination of the addresses and the priority for these addresses for the algorithm which the application uses. The addresses can be added to the application in any order because the algorithm reorders the addresses to find the best path. Regardless of the order of addresses entered by the user, the algorithm finds the best order that the route becomes most efficient. Also, the user selects the circular or not, and travel way option. All options may differentiate the best path. For instance, when the circular option is on, the application creates the best route by considering the goal destination is also the beginning destination. In other words, travel ends at the start location. This scenario applies to the postal workers who need to return to the post office after delivering all packages. Another function that serves the user can do on this page is to register or login. The application can be used without a login, but in this case, rating the streets/roads and creating the warning messages attributes are not permitted for the user.

The Map page contains the view of the route which the application algorithm produces. There is a pointer that symbolises the user on the map. The pointer moves in the map simultaneously with the move of the user. There may be any number of warning boxes for the streets on the map. Thus, the user can get current information about the roads around the route. This page shows the distance between the user and the goal destination and the remaining time to the goal. Both information is changed simultaneously with the move of the pointer/user. If the user logged in, he/she could create a warning about the streets/roads where he/she passes and rate these streets/roads in range 0 to 10.

2. Current Systems

Although there are a few applications like Google Maps[1], Speedy Route[2], and MapQuest[3] which have a similar use to what we seek to achieve, they all come short in certain features.

First of all, Speedy Route does not have a mobile application. Google Maps offers the most extensive set of features including satellite imagery, traffic conditions, and route planning, while the other two are more specialised in route planning. Google Maps only calculates a route in the given order. Since it is not possible for a user to know the best route without any calculations for multiple locations to be visited, the user cannot give addresses in the most efficient order. So Google Maps is not the best choice for our case. Another application that orders the destinations most efficiently is required. Speedy Route and MapQuest apps are mostly focused on this problem. They also change the order of the locations to be visited to be able to find the best possible route as we will do in our project.

However, personalisations in the routes do not apply to any of these applications. Pigeon’s Map will provide a personalised and self-improving experience to its users. While using Pigeon’s Map, users will face a better experience with fewer problems every day. The street rating system will help people to meet against undesired situations by recalculating their route again. With this system, users will rate roads and streets positively or negatively while using the system, and the algorithm will try to offer routes that pass from desired roads instead of undesired ones. This way, users will feel safer, and they will be safer as well. However, none of the current systems provides similar functionality.

Moreover, by the warning system that will be provided by Pigeon’s Map, users of the app will communicate and warn each other. This communication will create a solidarity-based community. This community will also affect the safety of users directly. This function is also not provided by any of the current systems.

Lastly, none of the current systems considers the priorities of the destinations. Since this is a route planning application for mostly postal workers, there can be some more urgent deliveries. This urgency should be a factor while calculating the route. Pigeon’s Map will have an algorithm that will also consider this situation and create the best route.

Due to these shortcomings in these existing applications, we decided to build Pigeon’s map to solve problems of current systems and provide the best experience to the users. Also, since we will use a map API similar to Google Maps, our application will include most properties of the current systems.

3. Proposed System

3.1 Overview

Pigeon’s map is a mobile application in Android[4] devices that users can find efficient routes with multiple stops to specific locations. The user can enter multiple addresses, and priority levels can be set for each address. The user can leave notes and grade a street to share an opinion or have an idea about the road.

Finding an efficient route to save time can be hard when there are many stops. A postal worker should choose which address to stop by first, find an efficient path to that address and repeat it for the rest of the addresses. The postal workers usually take an educated guess to find a route, but computer-assisted calculations could always improve it. Pigeon’s Map aims to provide an efficient path to the user to save time and fuel.

3.2 Functional Requirements

- Users will be able to create an account.
- Users will be able to login with their existing accounts.
- Users will be able to enter multiple addresses.
- There will be a function to set priority for each address.
- There will be a street voting system.
- Users will be able to change/remove their grades on the street.
- Users will be able to create notes to warn others.
- Users will be able to delete their notes.
- Users will be able to report any abusive content.

3.3 Nonfunctional Requirements

- The software will be a mobile application.
- The software will work on Android devices.
- The software will be open source.
- The software will be easy to use.
- The software will be user friendly.
- The software will work fast.
- The software will be efficient.
- The software will be secure from outside attacks.
- Some functions require an internet connection.

3.4 Pseudo Requirements

- Java will be used as a programming language.
- Calculations will be operated on the client’s device.
- The application will be ad-free
- The interface language will be English only in the release.
- Android Studio[5] environment will be used for development.

3.5 System Models

3.5.1 Scenarios

Scenario 1

Register with email

Actors: User

Entry Condition: The user taps the register button on the main screen.

Exit Condition: User finishes registration process.

Main Flow:

1. User taps on the “register” button.
2. Register page opens.
3. The user chooses to register via email by tapping the button “Register with email”.
4. The user enters his/her email address, desired username, and password.
5. The user taps the “register” button.
6. The username that the user wants is already in use. So the user gets a warning, and the system asks him/her to change it.
7. The user types a new username.
8. The user taps the “register” button.
9. The user sees a page that asks for email verification.
10. The user types the verification code that he/she gets via email.
11. The user taps the “submit” button.
12. The registration process is finished.

Scenario 2

Register with a Google account

Actors: User

Entry Condition: The user taps the “register” button on the main screen.

Exit Condition: User finishes registration process.

Main Flow:

1. User taps on the “register” button.
2. Register page opens.
3. The user chooses to register via Google by tapping the button “Register with Google”.
4. The user chooses a Google account.
5. The user sees a page to enter a username.
6. The user types a new username.
7. The user taps the “register” button.
8. The registration process is finished.

Scenario 3

Login with a Google account

Actors: User

Entry Condition: The user taps the “login” button on the main screen.

Exit Condition: The user finishes the login process.

Main Flow:

1. The user chooses to login via Google by tapping the button “Login with Google”.
2. The user chooses a Google account.

3. Login process finishes.

Scenario 4

Login with email

Actors: User

Entry Condition: The user taps the “login” button on the main screen.

Exit Condition: The user finishes the login process.

Main Flow:

1. The user chooses to login via email by tapping the button “Login with email”.
2. The user types his/her email and password into required text fields.
3. The user taps the “login” button.
4. The user gets a warning that there is not an account with the information provided.
5. The user checks and retypes the information.
6. The user taps the “login” button.
7. Login process finishes.

Scenario 5

Create a new path

Actors: User

Entry Condition: The user taps the “new path” button on the main screen.

Exit Condition: The user creates the route.

Main Flow:

1. The user taps the “new path” button.
2. The user selects the travel way (walking or car).

3. The user enters an address for each location he/she will visit.
4. The user chooses priority for each location.
5. The user taps the “finish” button.
6. The user sees a map with the suggested route.

Scenario 6

Create a warning

Actors: User

Entry Condition: The user wants to warn others about something about the route.

Exit Condition: The user creates a warning.

Main Flow:

1. The user taps the “create a warning” button.
2. The user types the message.
3. The user presses the submit button.

Scenario 7

Rate a location

Actors: User

Entry Condition: The user wants to rate a street for a better experience for later use.

Exit Condition: User rates the street.

Main Flow:

1. The user taps the “rate this location” button.
2. The user chooses the rate between 1-10.
3. The user taps the “submit” button.

Scenario 8

Report a warning

Actors: User

Entry Condition: The user wants to report abusive content on user warnings.

Exit Condition: The user reports the warning.

Main Flow:

1. The user taps on the warning.
2. The user taps the report button.
3. The user presses the submit button.

3.5.2 Use Case Model

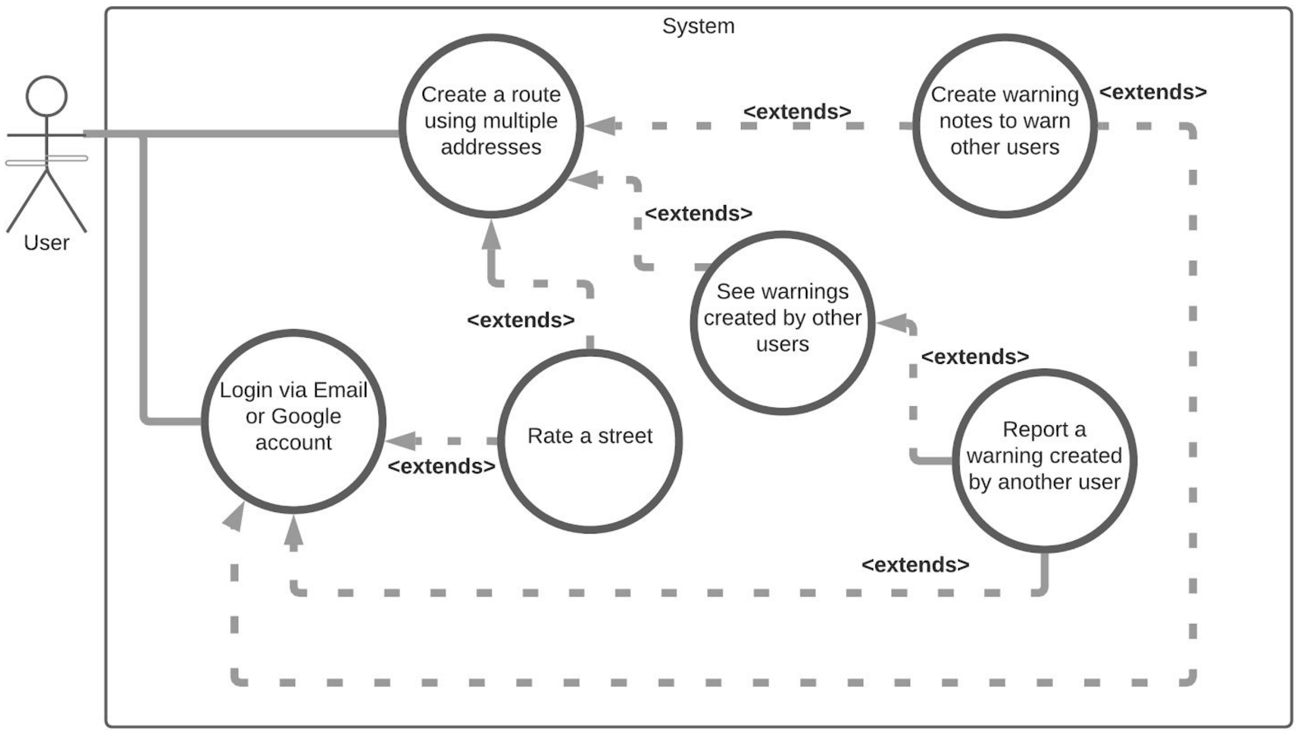

System
User
Create a route using multiple addresses
Create warning notes to warn other users
See warnings created by other users
Login via Email or Google account
Rate a street
Report a warning created by another user
<extends>

Figure 1: UML Use case diagram for Pigeon's Map

3.5.3 Object and Class Model

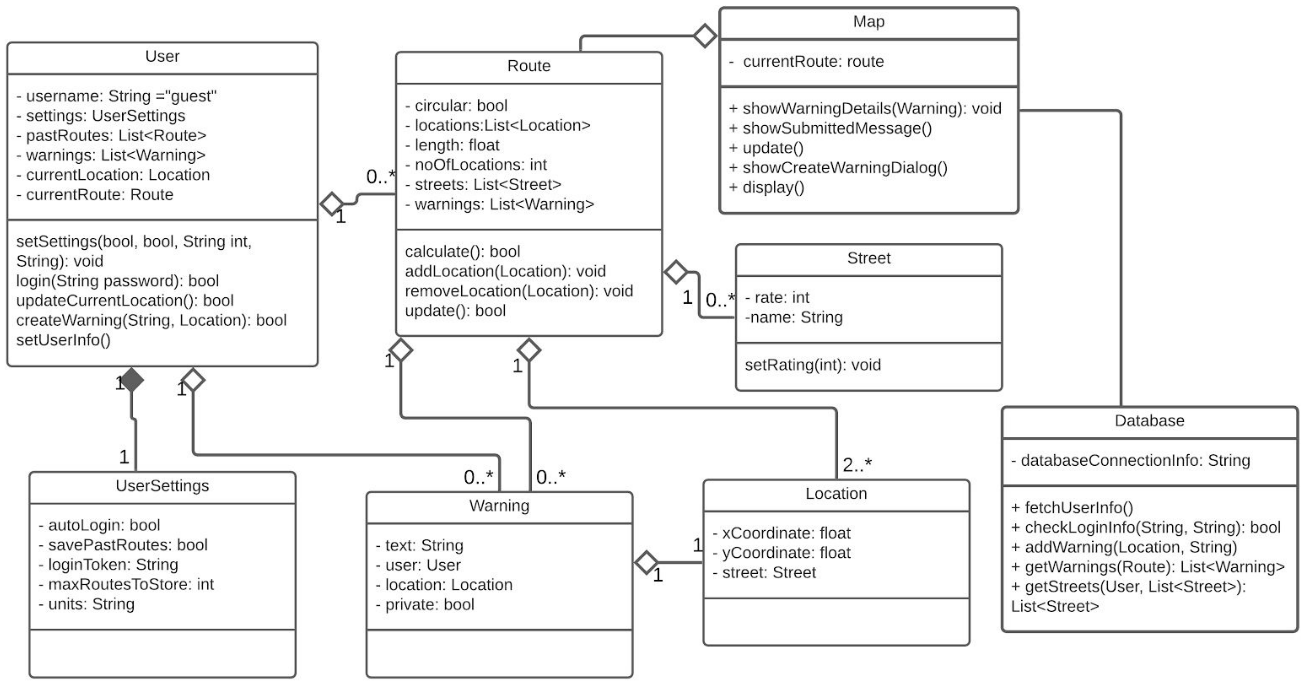

User
- username: String ="guest"
- settings: UserSettings
- pastRoutes: List<Route>
- warnings: List<Warning>
- currentLocation: Location
- currentRoute: Route
setSettings(bool, bool, String int, String): void
login(String password): bool
updateCurrentLocation(): bool
createWarning(String, Location): bool
setUserInfo()
Route
- circular: bool
- locations:List<Location>
- length: float
- noOfLocations: int
- streets: List<Street>
- warnings: List<Warning>
calculate(): bool
addLocation(Location): void
removeLocation(Location): void
update(): bool
Map
- currentRoute: route
+ showWarningDetails(Warning): void
+ showSubmittedMessage()
+ update()
+ showCreateWarningDialog()
+ display()
Street
- rate: int
-name: String
setRating(int): void
Database
- databaseConnectionInfo: String
+ fetchUserInfo()
+ checkLoginInfo(String, String): bool
+ addWarning(Location, String)
+ getWarnings(Route): List<Warning>
+ getStreets(User, List<Street>): List<Street>
UserSettings
- autoLogin: bool
- savePastRoutes: bool
- loginToken: String
- maxRoutesToStore: int
- units: String
Warning
- text: String
- user: User
- location: Location
- private: bool
Location
- xCoordinate: float
- yCoordinate: float
- street: Street
0..*
1
1
0..*
1
1
1
1
1
0..*
0..*
2..*
1
1

Figure 2: Class diagram for Pigeon’s Map

3.5.4 Dynamic Models

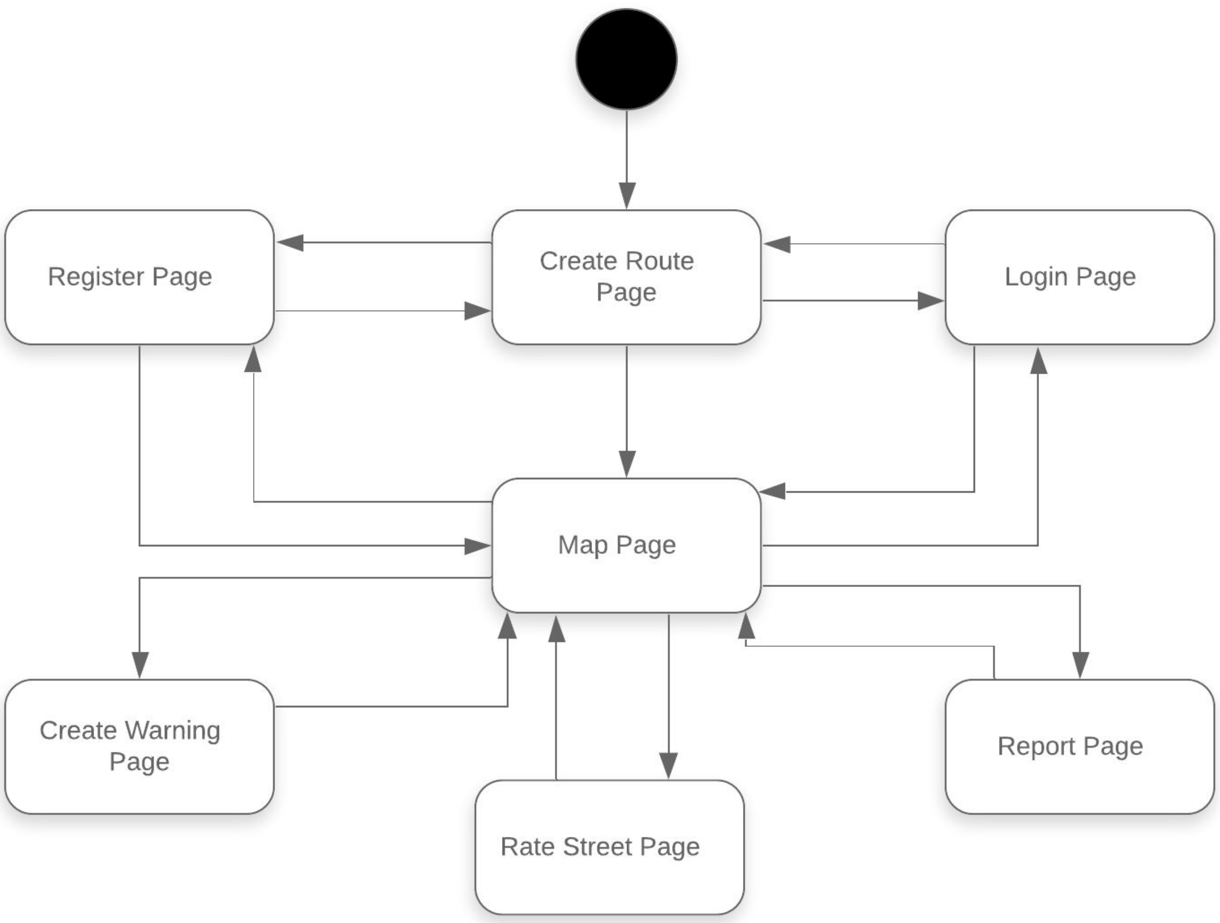


Figure 3: Page State Diagram for Pigeon’s Map

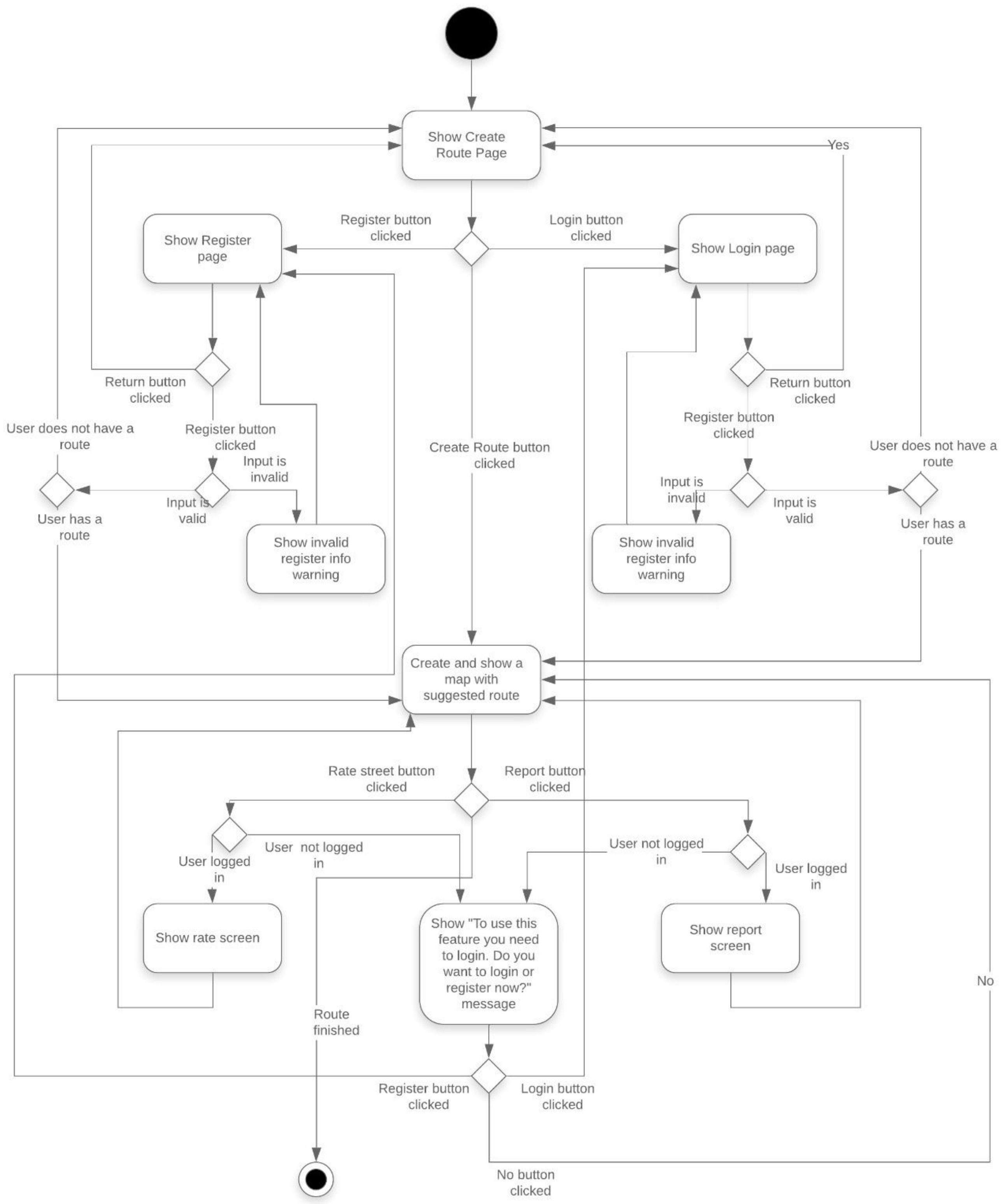


Figure 4: Activity Diagram for Pigeon’s Map

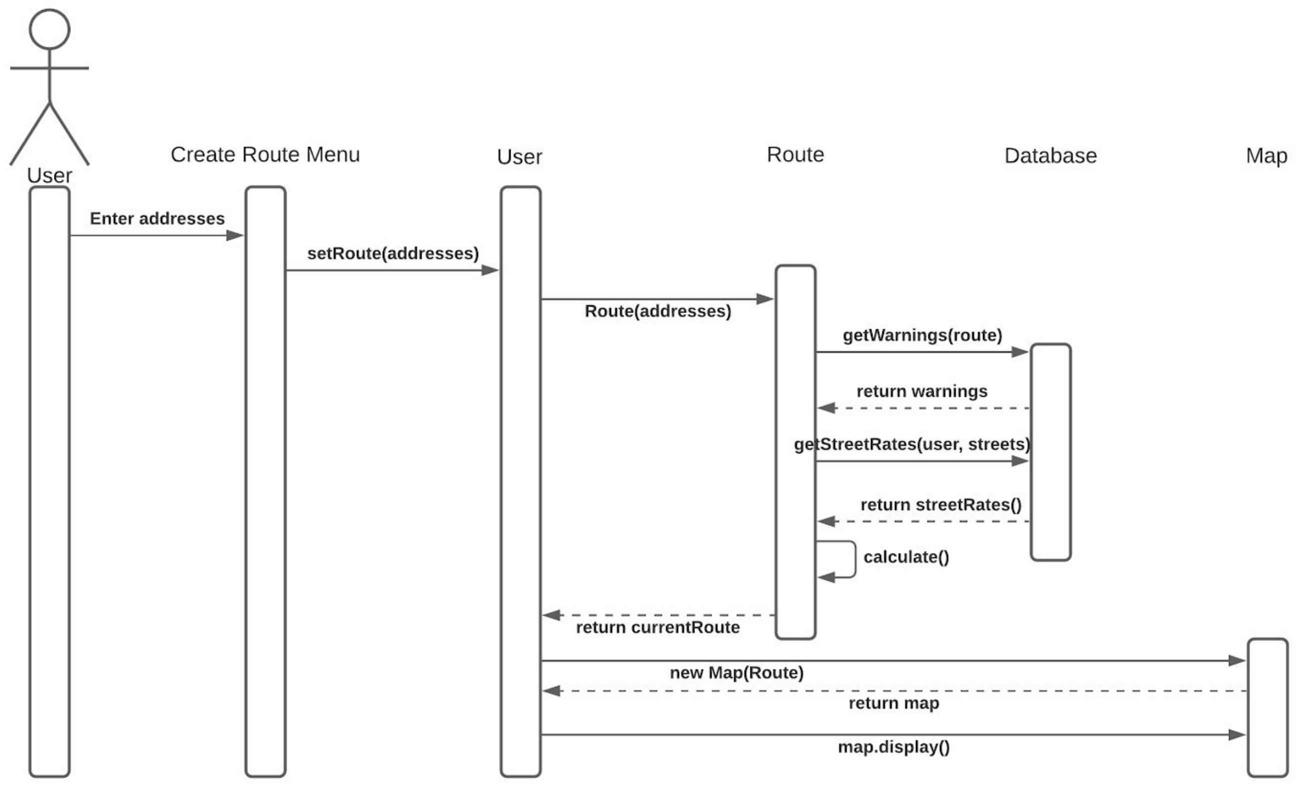


Figure 5: Sequence Diagram for creating a route

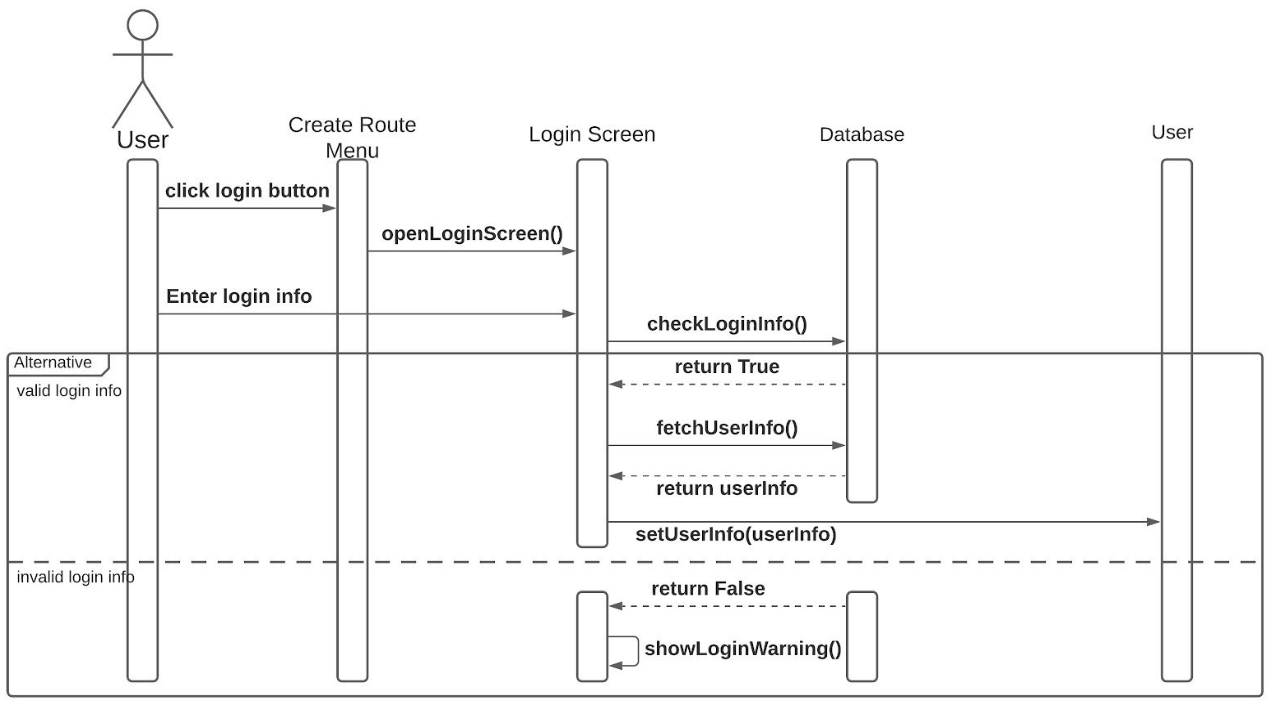


Figure 6: Sequence Diagram for the login process

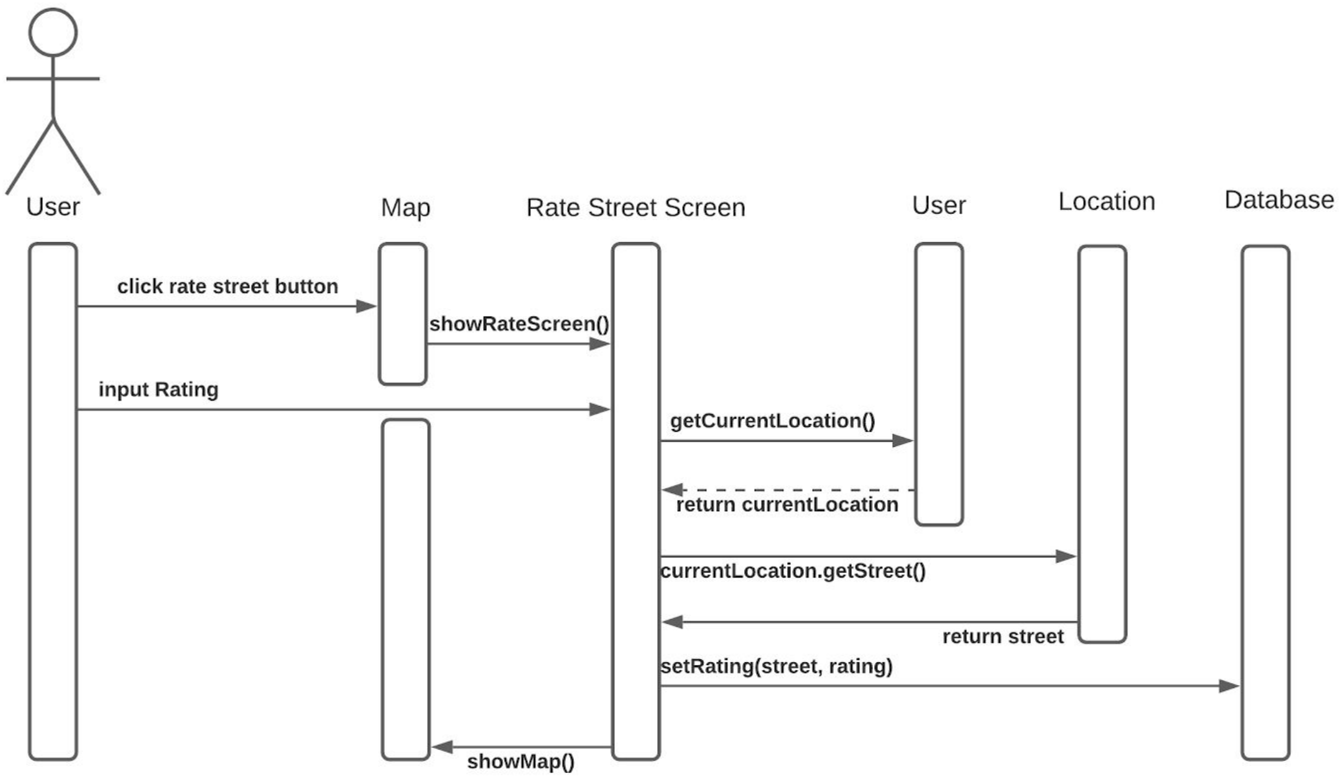


Figure 7: Sequence Diagram for rating a street

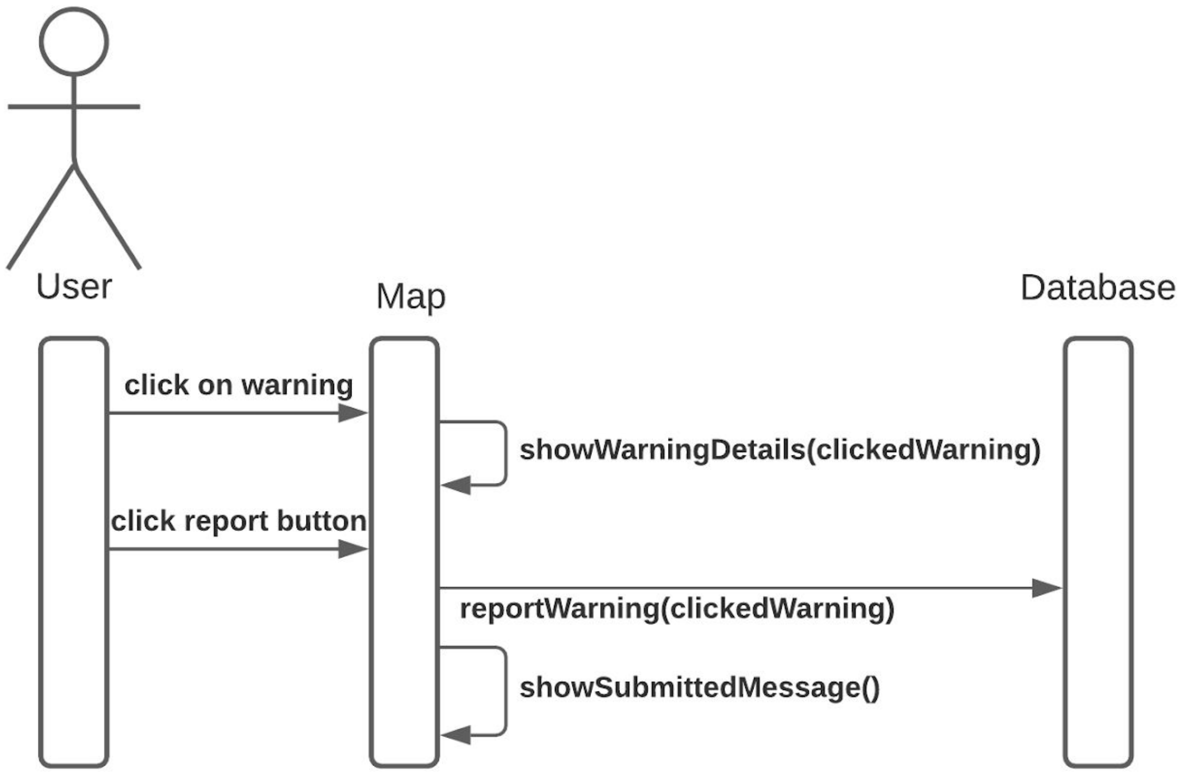


Figure 8: Sequence Diagram for reporting a warning

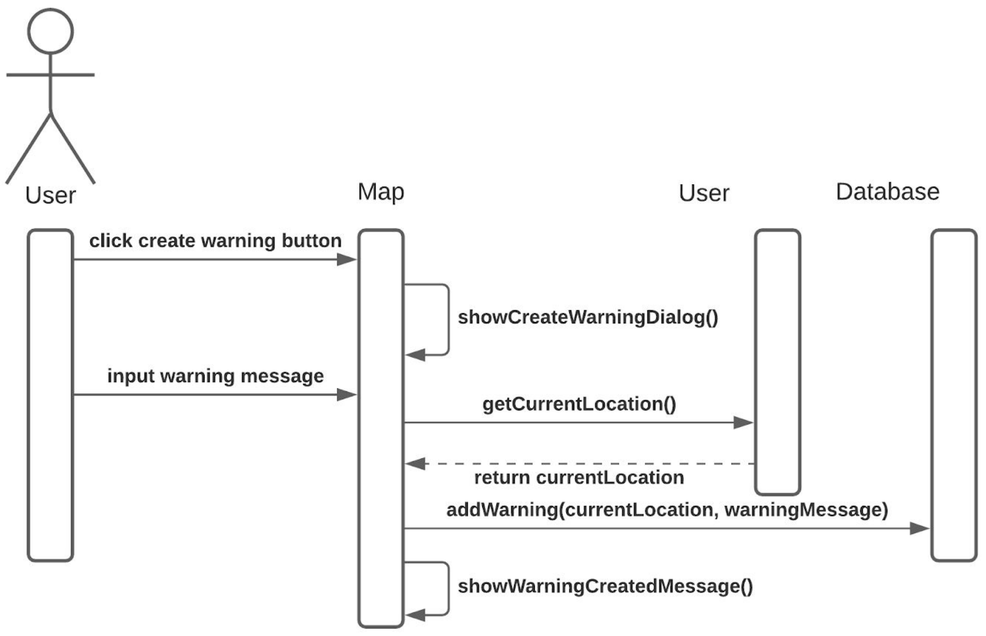

User
Map
User
Database
click create warning button
showCreateWarningDialog()
input warning message
getCurrentLocation()
return currentLocation
addWarning(currentLocation, warningMessage)
showWarningCreatedMessage()

Figure 9: Sequence Diagram for creating a warning

3.5.5 User Interface and Screen Mockups

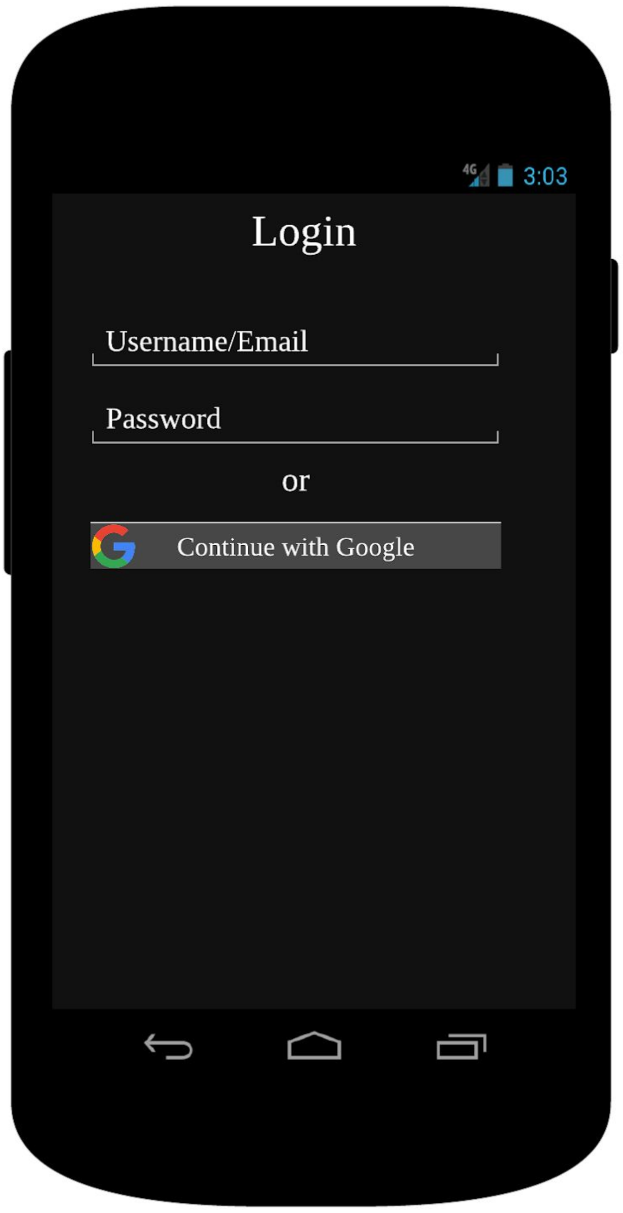


Figure 10: Mockup for login screen

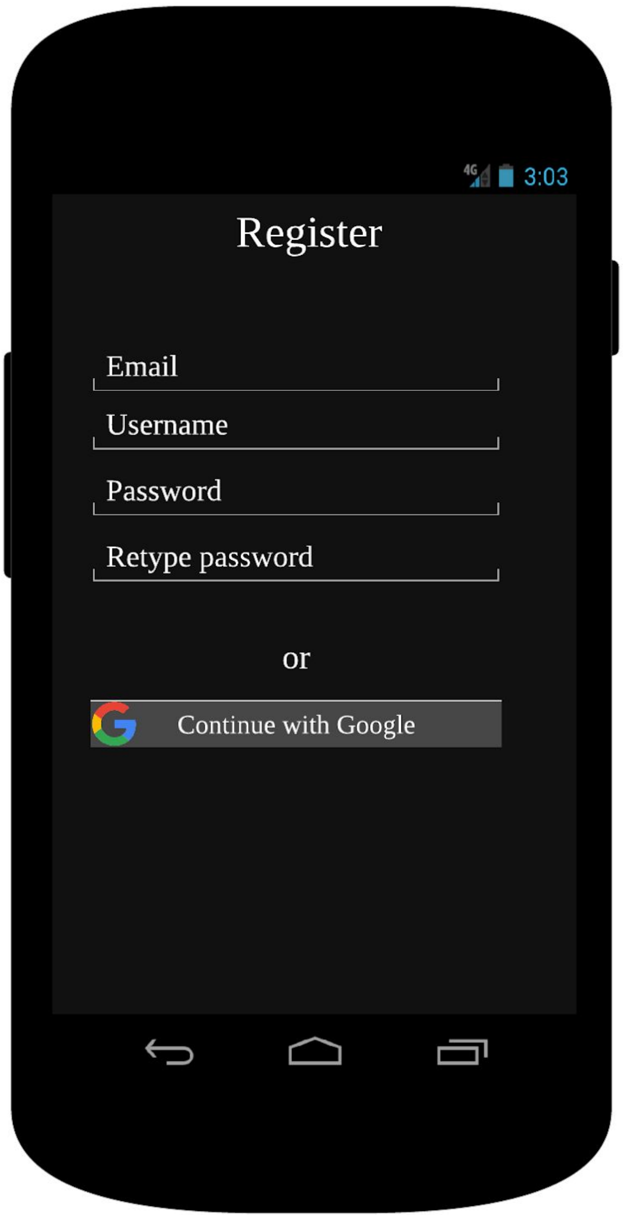
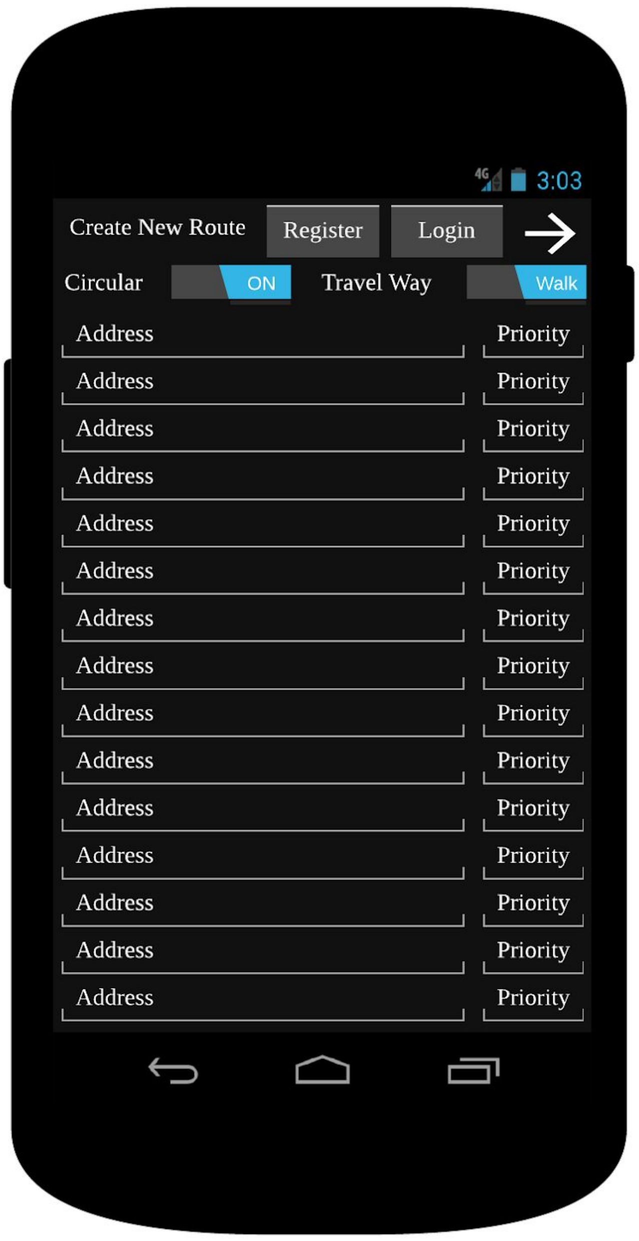


Figure 11: Mockup for register screen

These two pages are for login and registration. First-time users may choose to register, and users with an existing account may choose to log in. The app works without login required, but users who do not log in are unable to use some abilities. Login and register are possible via email or Google account.



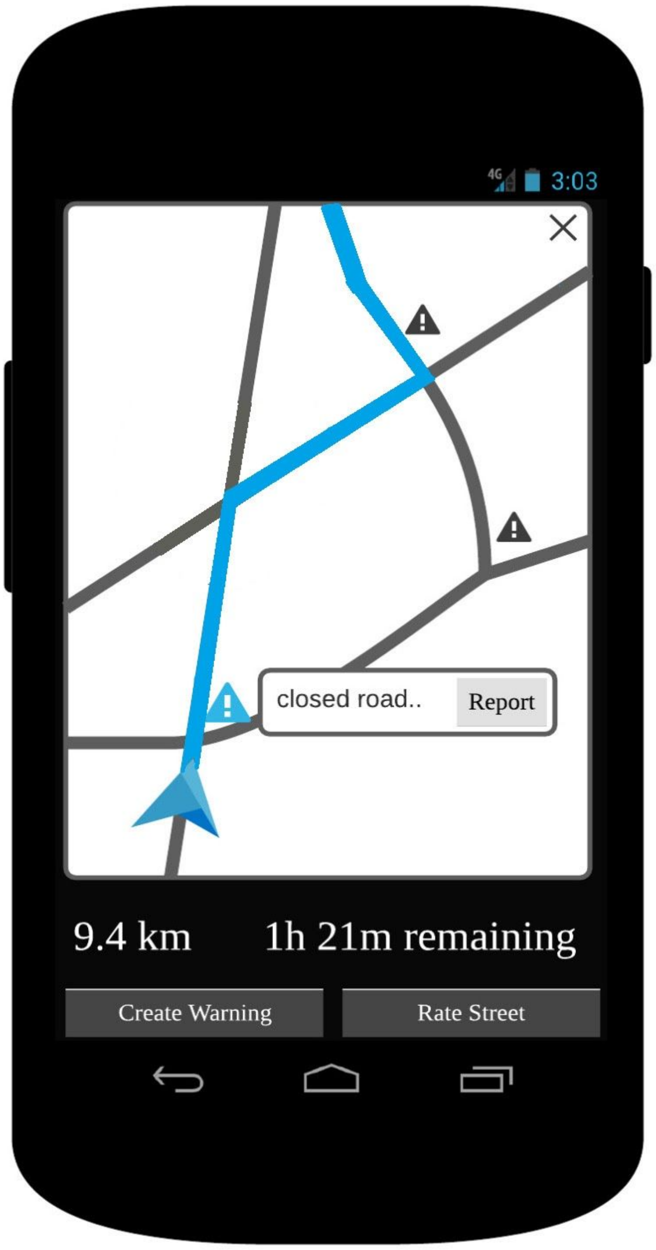


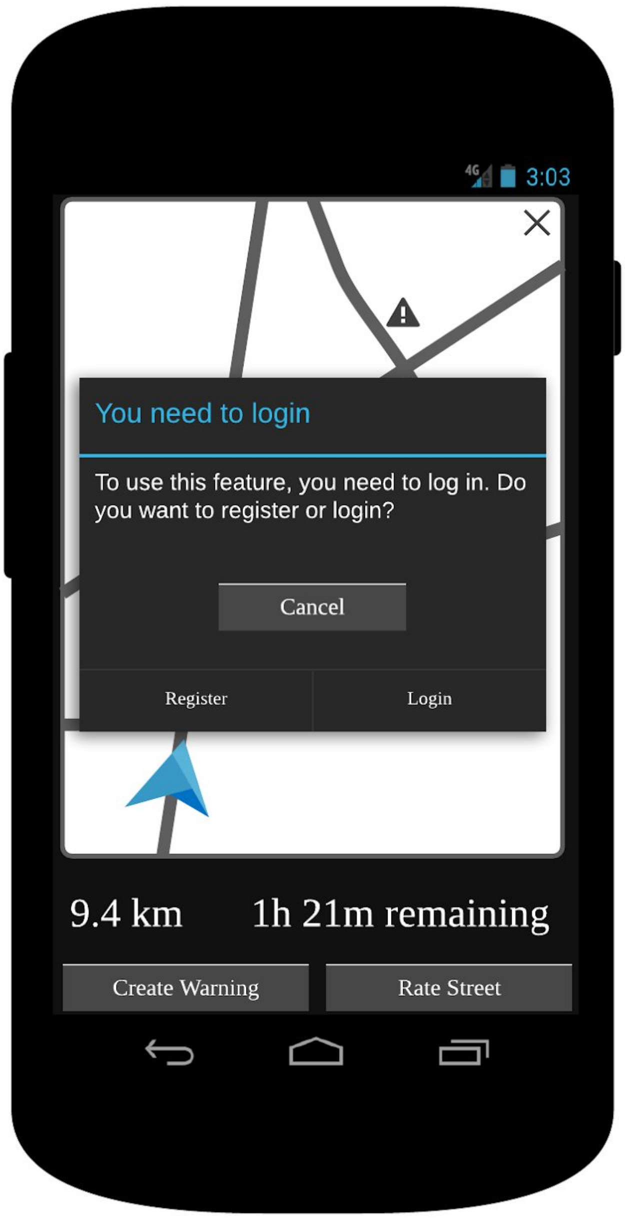
Figure 12: Mockup to create route screen

Figure 13: Mockup for the map screen

Users use the screen in Figure 12 to create a new route. If the user is already logged in, Register and Login buttons at the top will not appear. The user needs to select addresses and the priority for each of them on this screen. Addresses with higher priority will be visited earlier. Also, the user has to set settings for the root at the top of the screen. If the user needs to return to the starting point after the travel, he/she needs to activate the

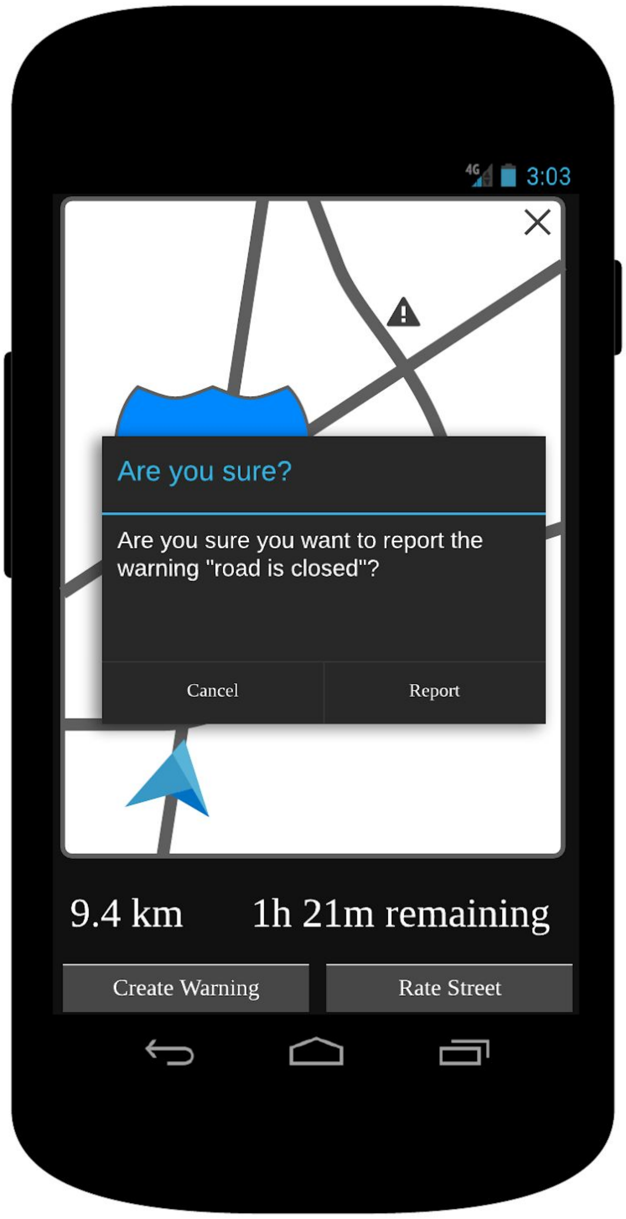
"Circular" option. Also, the user needs to select "Walk" or "Vehicle" according to the way they will visit each place.

After the calculation, the user will see a map with the suggested route, like in Figure 13. Users will see this screen during travel. Users also can see any warnings created by other users around the way. Users can choose to report these warnings if they are inappropriate. Also, they can choose to create a warning themselves or to rate a street.


You need to login

To use this feature, you need to log in. Do you want to register or login?

Figure 14: Mockup for login required warning

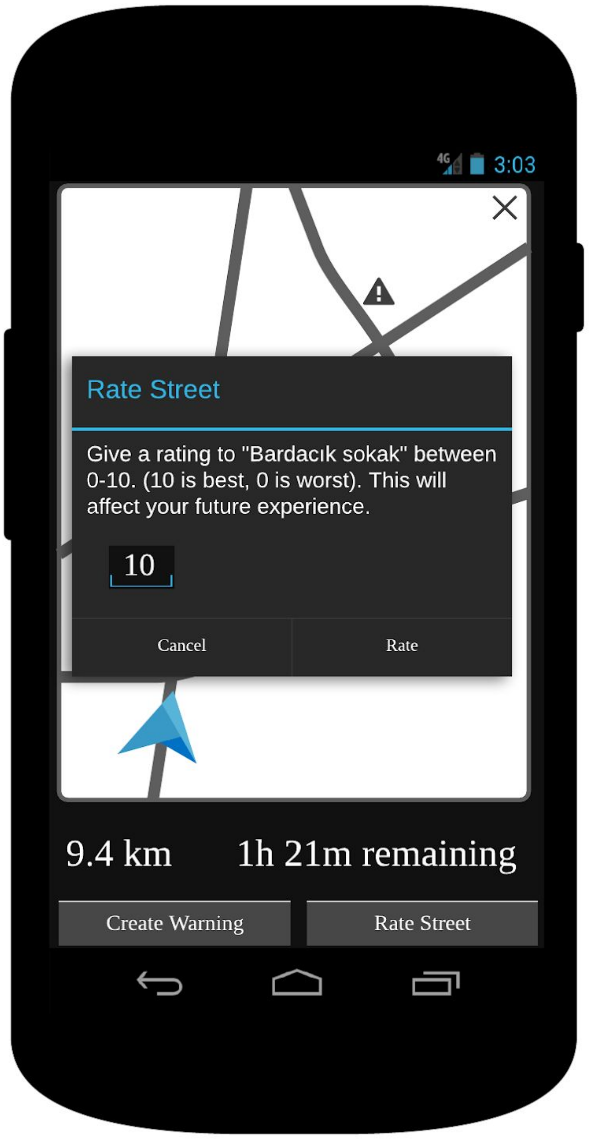

Are you sure?

Are you sure you want to report the warning "road is closed"?

Figure 15: Mockup for reporting a warning

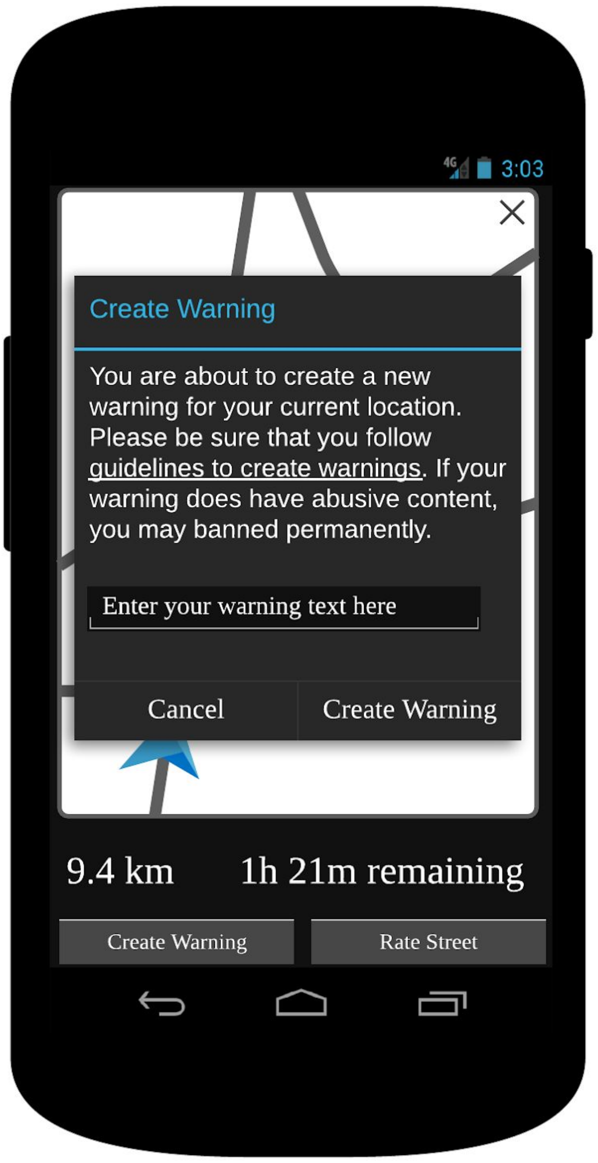
If the user tries to use a function that requires login, a warning like in Figure 14 will appear. From this warning, the user can choose to register, login, or abandon doing this operation. If the user decides to register or login, he/she will resume the route.

If the user is logged in and wants to report a warning, a page like a Figure 15 will appear to avoid taps by accident. If the user chooses to report, the warning information will move to the reported warnings database and will be removed by administrators if it is really against the rules.



Give a rating to "Bardacık sokak" between 0-10. (10 is best, 0 is worst). This will affect your future experience.

Figure 16: Mockup for rating a street



You are about to create a new warning for your current location. Please be sure that you follow guidelines to create warnings. If your warning does have abusive content, you may banned permanently.

Figure 17: Mockup for creating a warning

A user can choose to rate a street. These ratings will be used for the later routes for that particular user. The algorithm will try to use the roads with higher ratings instead of streets with lower ratings. This rating system will provide users with a better experience. If a logged-in user tries to rate a street, a dialogue like a Figure 16 will appear. The user should tap the “Rate” button after providing a rating number.

Finally, users can create their own warnings for other users. Users who want to create a warning will see a dialogue like Figure 17. In this dialogue, users should provide a text to share with other users.

4. Other Analysis Elements

4.1. Consideration of Various Factors in Engineering Design

With the Covid-19 outbreak, public health has become one of the biggest concerns in the world. In our case, to address the needs of public health, our application can only decrease the probability of people, mostly postal workers, getting the virus. This can be achieved. As well as minimising the time, we can optimise the route to select emptier streets. The idea is that the worker will have less human contact by going through empty streets resulting in less likelihood of getting the virus. The effect of public health factors is 5.

To address the needs of public safety, our application can decrease the probability of a traffic accident. This can be done by choosing safe routes. The users of the application can determine the safeness of a route. After using the route, users can rate the safety, and maybe other aspects of the route as well. By continuously rating, users can change the status of the route continually. The effect of public safety factors is 4.

Besides, global factors affect our application. Since we can not launch our own satellite and use images from it, we are constrained to use globally available sources such as Google Maps. Therefore, any error in these sources will directly affect our calculations. The effect of global factors is 8.

	Effect level	Effect
public health	5	selecting empty routes
public safety	4	rating of routes
global	8	correctness of results

Table 1: Factors that can affect analysis and design.

4.2. Risks and Alternatives

One of the risks is, not much likely, that the users do not have a Google account or do not want to use their Google account to register and log into our application. In this case, users can not use our application. If this happens, our plan B is to register and log them into our system via their email.

Another risk is, since our primary focus is on postal workers, that the user might want a circular route in which the start location and end location are the same. In this case, there may be more efficient routes than the non-circular route calculation. If this happens, our plan B is to have a circular route option that also considers the route back to the start location as well.

	Likelihood	Effect on the project	B plan summary
no Google account	not much	user can not log in to the system	register with an email
circular route	probably all the time	the non-circular calculation might not be efficient	circular route option

Table 2: Risks

4.3. Project Plan

WP#	Work package title	Leader	Members involved
WP1	Analysis	Berdan Akyürek	Ömer Olkun, Ekin Üstündağ,Tanay Toksoy, Abdullah Ayberk Görgün
WP2	High level design	Tanay Toksoy	Ömer Olkun, Ekin Üstündağ, Abdullah Ayberk Görgün, Berdan Akyürek

Table 3: List of work packages

WP1: Analysis
Start date: 7.11.2020 **End date**:21.11.2020
Leader: Berdan Akyürek **Members involved**:Ömer Olkun, Ekin Üstündağ,Tanay Toksoy, Abdullah Ayberk Görgün
Objectives: Analyze the details of the problem. All relevant issues must be addressed.
Tasks **Task 1.1:** All the diagrams of the models should be created. **Task 1.2:** All the sections of the analysis report should be written.

Deliverables **D1.1:** Analysis report

Table 4: work package 1

WP2: High level design
Start date: 21.11.2020 **End date**: 27.12.2020
Leader: Tanay Toksoy **Members involved**:Ömer Olkun, Ekin Üstündağ,Berdan Akyürek, Abdullah Ayberk Görgün
Objectives: A high-level design of the solution.
Tasks **Task 2.1:** All the diagrams of the models should be created. **Task 2.2:** All the sections of the high-level design report should be written. **Task 2.3:** A demo of the application should be done. **Task 2.4:** A presentation of the project should be done.
Deliverables **D2.1:** An Analysis report **D2.2:** A Demo of the application **D2.3:** A presentation of the project

Table 5: work package 2

4.4. Ensuring Proper Teamwork

To ensure proper teamwork, we use Github[6] for codes. This way, we can track the contribution of every group member. If somebody is not doing his work, we can see this from Github and warn each other. To see other members' commits makes other members more motivated. The ease of use provided by Github lets us spend less time on synchronisation problems.

Similarly, we use Google Docs[7] for reports. This way, we can monitor the logs and figure out which member had how much workload and we can synchronise our work simultaneously.

Face to face meetings and similar activities are not being used because of the pandemic. So we prefer to use online communication tools such as Whatsapp[8] and Discord[9] to connect easily. This way, we can share work equally and remind our responsibilities to each other. We ask and communicate with each other to help others or receive help from others. We keep every group member accountable this way.

4.5. Ethics and Professional Responsibilities

One of the responsibilities that we have is not to plagiarise. That is, do everything on our own and give references when we use outside sources. We have not plagiarised since the start of the project. We did everything on our own and gave references for the external sources we use in our reports. We will not plagiarise in the future either.

Another responsibility that we have is the protection of personal data. Since we ask for users' emails, we should protect this information. Besides, we must not use this information outside the range of the application as well.

4.6. Planning for New Knowledge and Learning Strategies

Some of us do not have much experience in developing an Android application. So they will need to learn to develop an Android application. To learn, they will use online courses, and also the group members knowing developing an Android application can help them.

Also, we need more information and experience in some fields such as teamwork, database, and server usage, object-oriented programming, API usage, algorithm, and software efficiency. Knowledge from previous courses that we took before may not be enough for a project like this. So we will use appropriate learning strategies to conclude our project correctly. We will receive help from other team members when needed. Every group member will help others with the fields that they are experienced in.

Also, every group member should use self-learning strategies for faster development. Online documentation, tutorials, and books may be helpful in this context.

With the development of our project, we may face new problems, and while searching for solutions, we may need new knowledge that is not planned before. In this case, we will keep learning with appropriate learning strategies and stick to plan. In short, learning will be an integral part of our project.

5. References

[1]"Google Maps", *Google Maps*. [Online]. Available: https://www.google.com/maps. [Accessed: 20- Nov- 2020].

[2] Speedy Route, "Speedy Route - The Delivery Route Planner", *Speedyroute.com*. [Online]. Available: https://www.speedyroute.com/. [Accessed: 20- Nov- 2020].

[3]"Official MapQuest - Maps, Driving Directions, Live Traffic", *Mapquest.com*. [Online]. Available: https://www.mapquest.com/. [Accessed: 20- Nov- 2020].

[4]"Android | The platform pushing what’s possible", *Android*. [Online]. Available: https://www.android.com/. [Accessed: 20- Nov- 2020].

[5]"Download Android Studio and SDK tools | Android Developers", *Android Developers*. [Online]. Available: https://developer.android.com/studio. [Accessed: 20- Nov- 2020].

[6]"GitHub: Where the world builds software", *GitHub*. [Online]. Available: https://github.com/. [Accessed: 20- Nov- 2020].

[7]"Google Docs: Free Online Documents for Personal Use", *Google.com*. [Online]. Available: https://www.google.com/docs/about/. [Accessed: 20- Nov- 2020].

[8]"WhatsApp", *WhatsApp.com*. [Online]. Available: https://www.whatsapp.com/. [Accessed: 20- Nov- 2020].

[9]"Discord | Your Place to Talk and Hang Out", *Discord*. [Online]. Available: https://discord.com/. [Accessed: 20- Nov- 2020].