Rowan Rideshare Design Document

Wen Cao, Tyler Carberry, Benny Chen,
Benny Liang, Michael Matthews, Tapan Soni,
John Stranahan, Nicholas La Sala

10/26/18
Senior Project Fall 2018

Table of Contents

High Level Description

The purpose of the application is to give users a way to easily find other commuters and organize carpools to and from Rowan University by matching them with other students who also go to Rowan University and are looking for a carpool.

When the user creates an account they will then enter their general weekly schedule along with an address that will be used to determine their location relative to Rowan. Once the app has this information stored, the user can then request a search of other users that are also looking to organize carpools. The app will match people based on how similar their routes to and from Rowan are, and from similar schedules. The app will also add the new user to the pool of users that can be found as a potential match.

Once the user has found other students that are looking for carpools they can then enter into a chat with them and work out any necessary details. This includes pick up/drop off times, along with what kind of car they should expect to pick them up.

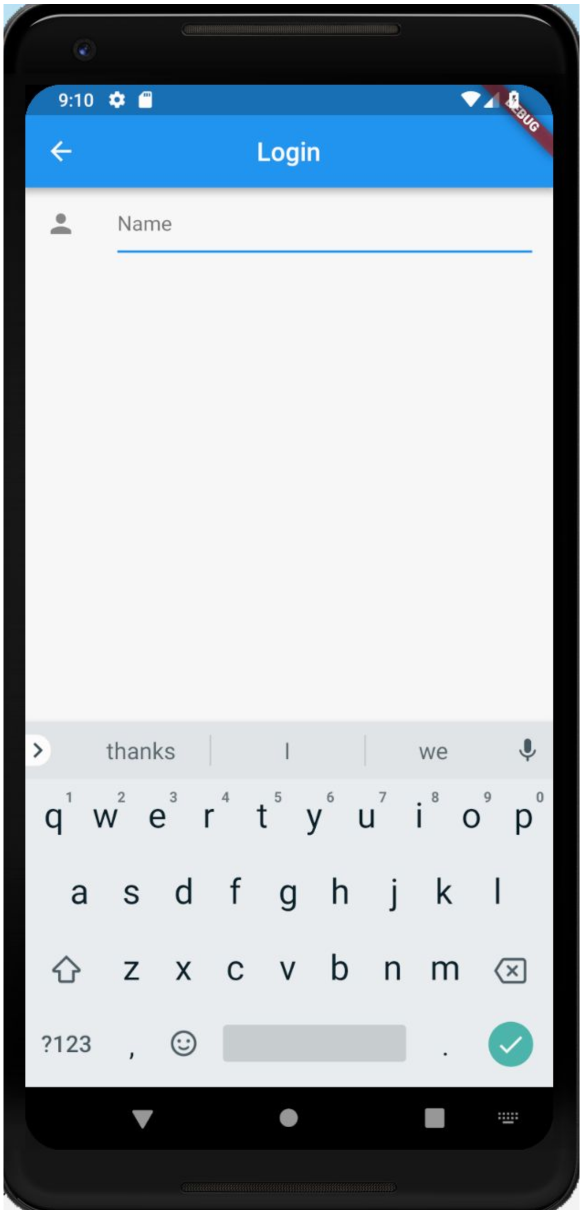
Problem Solving Approaches

The biggest problem facing the development of the app was how the user’s schedule was to be handled. The two approaches that were considered were a simple arriving and leaving time for each day of the week, or allowing the user to create blocks of critical times they would need rides that they could choose to search for. Ultimately the simpler arrival and departure for each day was selected because it would provide users the largest number of potential matches possible, which is invaluable to an app with an inherently low maximum number of users. This would also allow for each users schedule to be returned in a higher number of searches which would improve overall visibility for the profile.

Screens

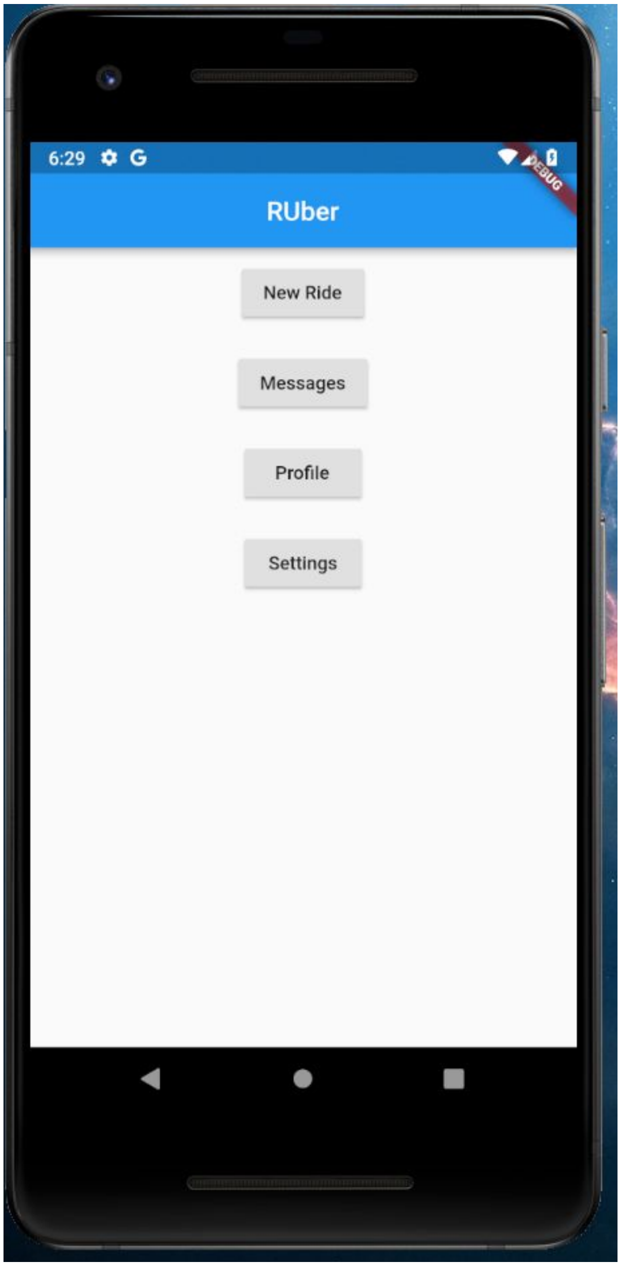
Login Screen

The Login Screen will be the first screen that users who are not already logged in will see when they start the app. Here the users will enter their Rowan login credentials before being redirected to the home screen. If the user launches the app and is already logged in then this screen will be skipped.



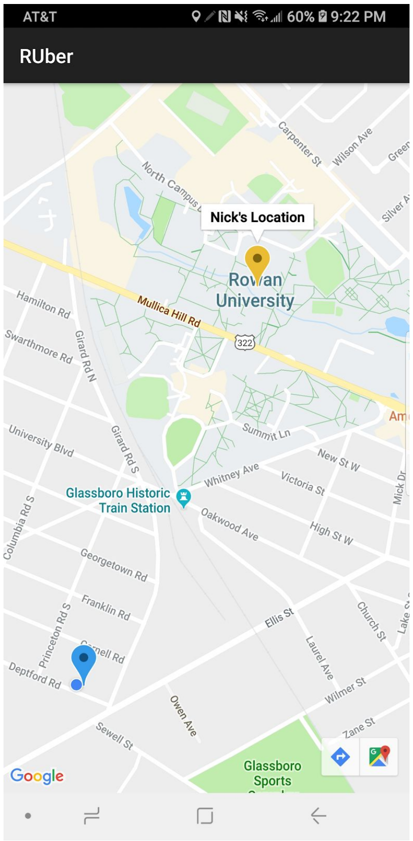
Home Screen

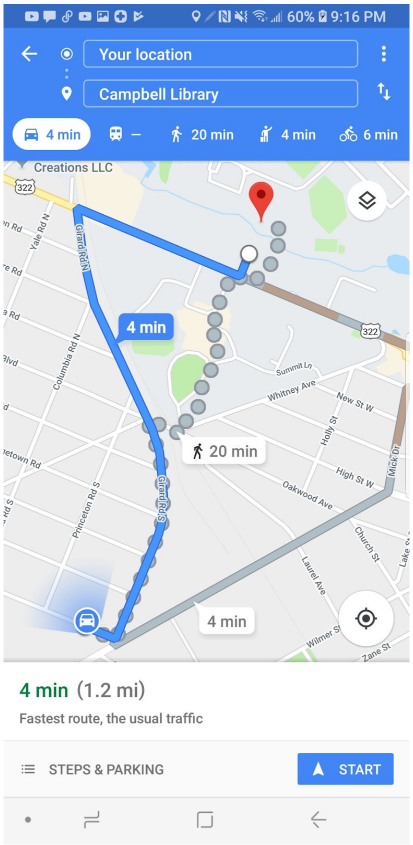
This is the primary screen available to the user, and it features a menu to navigate the app. In the final version of the app it will feature buttons to access the new ride, messages screen, profile edit screen, and setting.



Map Screens

The user can select those whom they decide to start a carpool with and view their location on a map screen. The user can then choose to load the route from google maps if they wish to use that for navigation.




Your location

Campbell Library

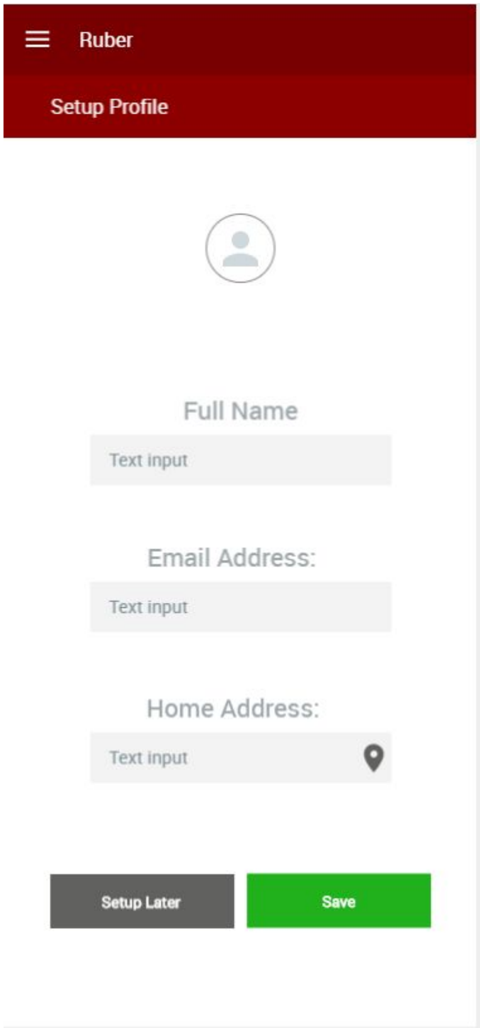
4 min (1.2 mi)

Fastest route, the usual traffic

Mock Ups

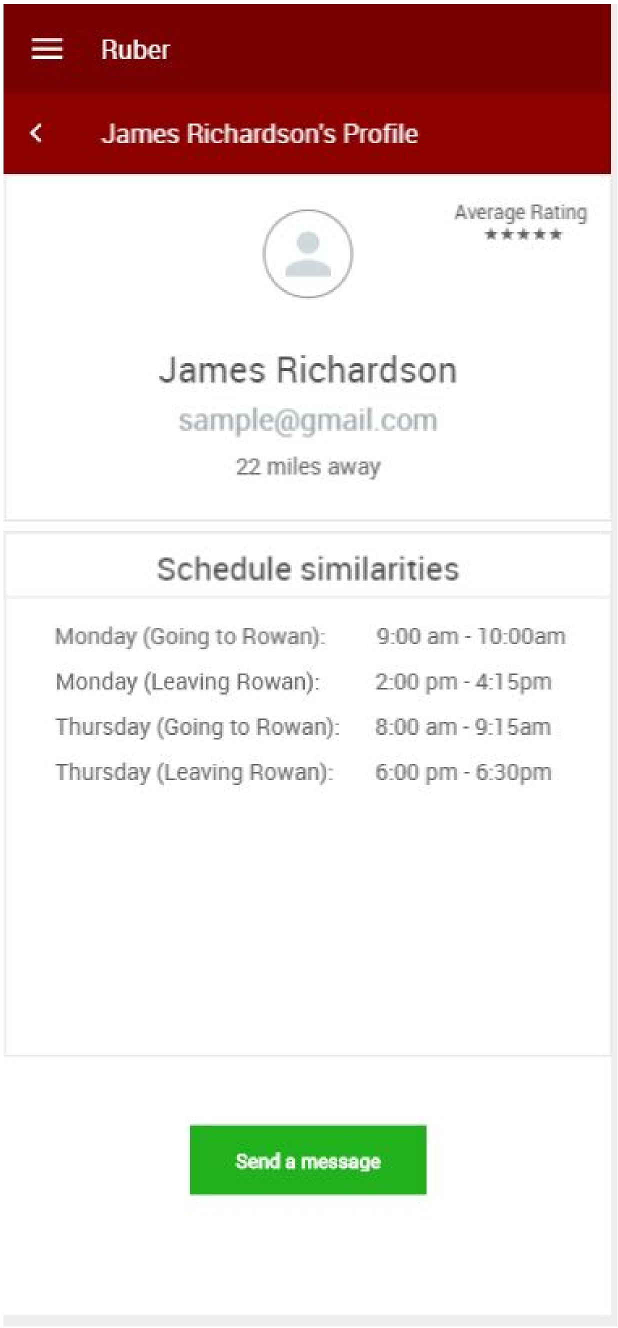
Profile Editor

This screen is where user can choose to change any information about them that will be displayed on the app.



Profile View

This screen allows the user to view other user's profiles, and decide if they want to begin a chat with them.



James Richardson

sample@gmail.com

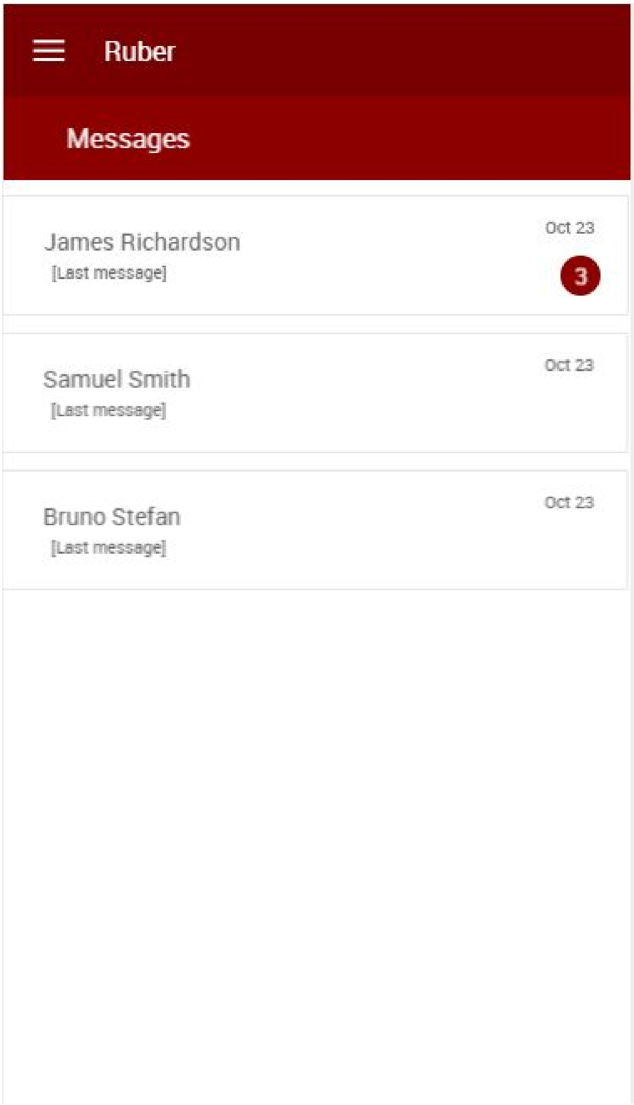
22 miles away

Schedule similarities

Monday (Going to Rowan):	9:00 am - 10:00am
Monday (Leaving Rowan):	2:00 pm - 4:15pm
Thursday (Going to Rowan):	8:00 am - 9:15am
Thursday (Leaving Rowan):	6:00 pm - 6:30pm

Messenger Screen

This screen will allow the user to view all of their chats that they started and open the chat screen with whomever they select.



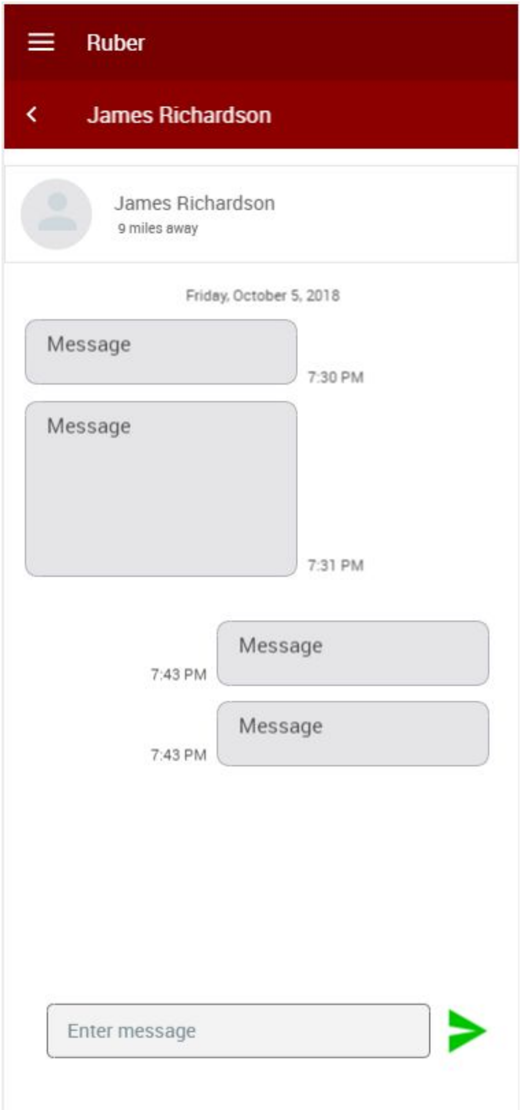
James Richardson
[Last message]
Oct 23
3

Samuel Smith
[Last message]
Oct 23

Bruno Stefan
[Last message]
Oct 23

Chat Screen

This screen allows users will be able to send and receive messages with the other users that they match with.



Friday, October 5, 2018

Message 7:30 PM

Message 7:31 PM

7:43 PM Message

7:43 PM Message

Screen Navigation

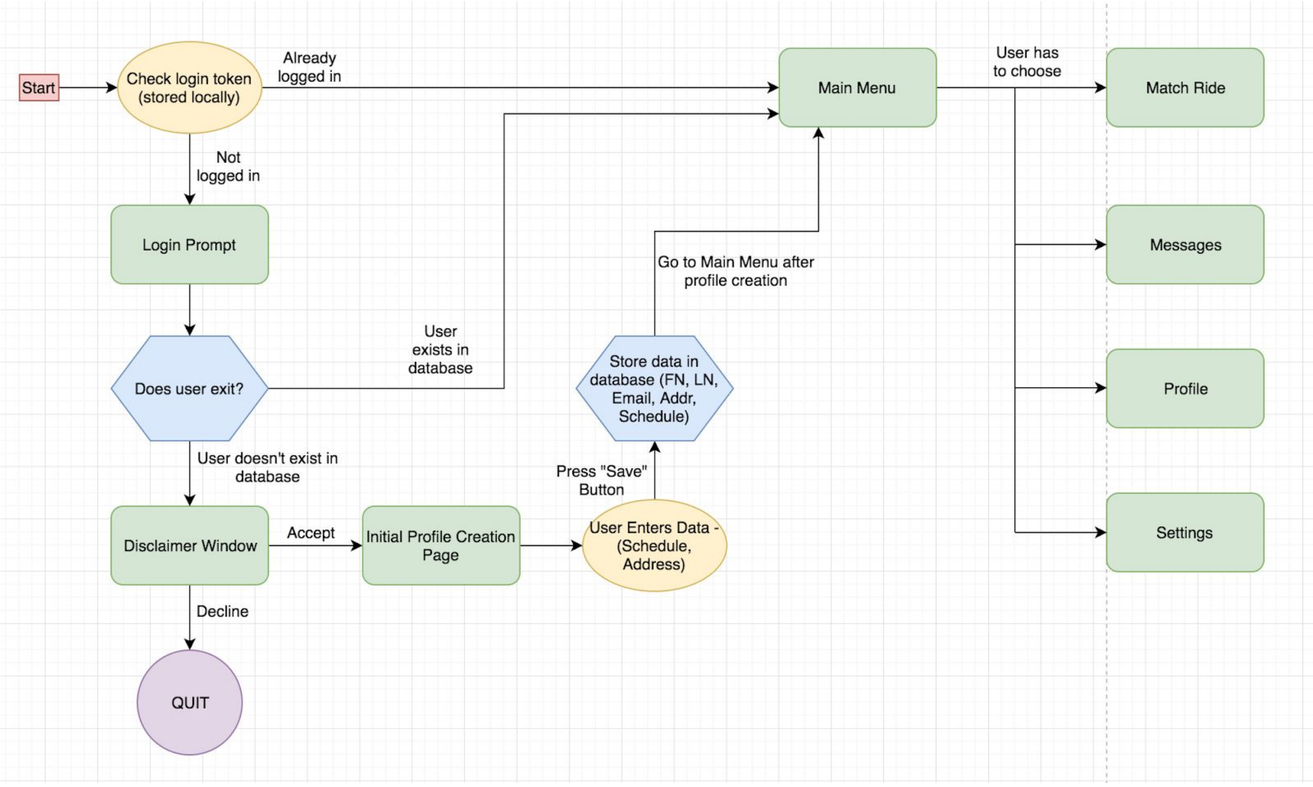
When a new user opens the app for the first time they be presented a disclaimer, followed by the login screen. The users login using their rowan account credentials, and from there they are redirected to a profile edit screen. This is screen allows the user verify their information and begin creating a schedule. After the user has set up their profile for the first time they will be redirected to the home screen which will be the default screen on future app launches. The home screen will act as a hub that allows the user to access any of the functions that the app provides them.

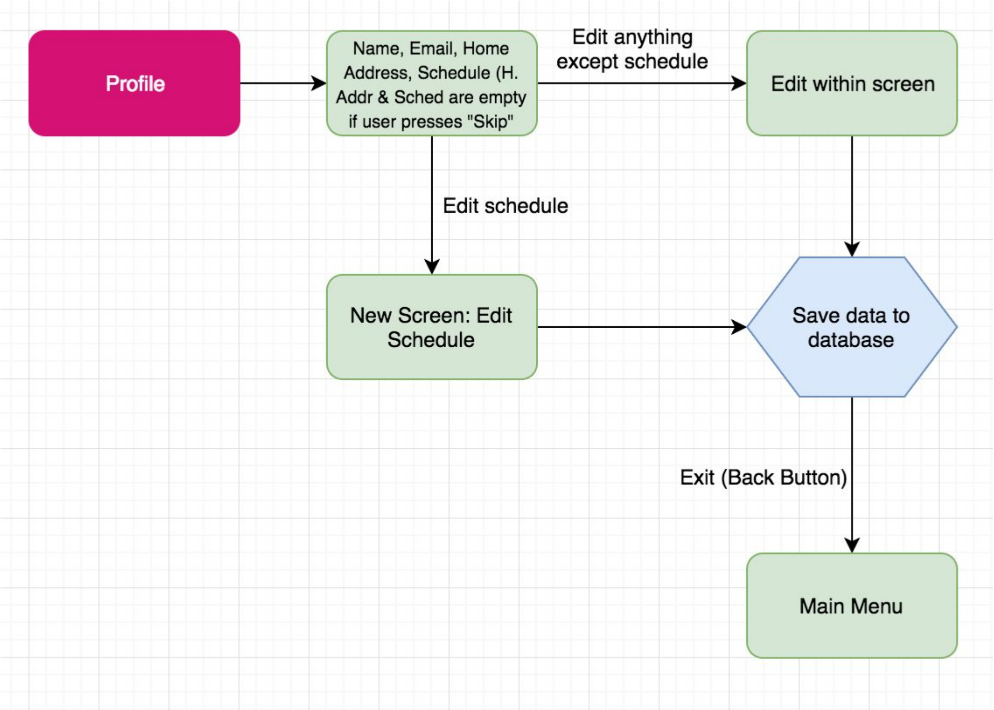
Since this is a new user that just adjusted their settings, the next thing they would likely do is begin searching for people to form a carpool with. The user would select the search option from the main menu that would bring them to a screen that list all of the students that would be a potential match for the user. The user can send ride requests to whomever they think would be a good match from the students that are returned to them. Once the other party has accepted the ride request, a chat between the users is created.

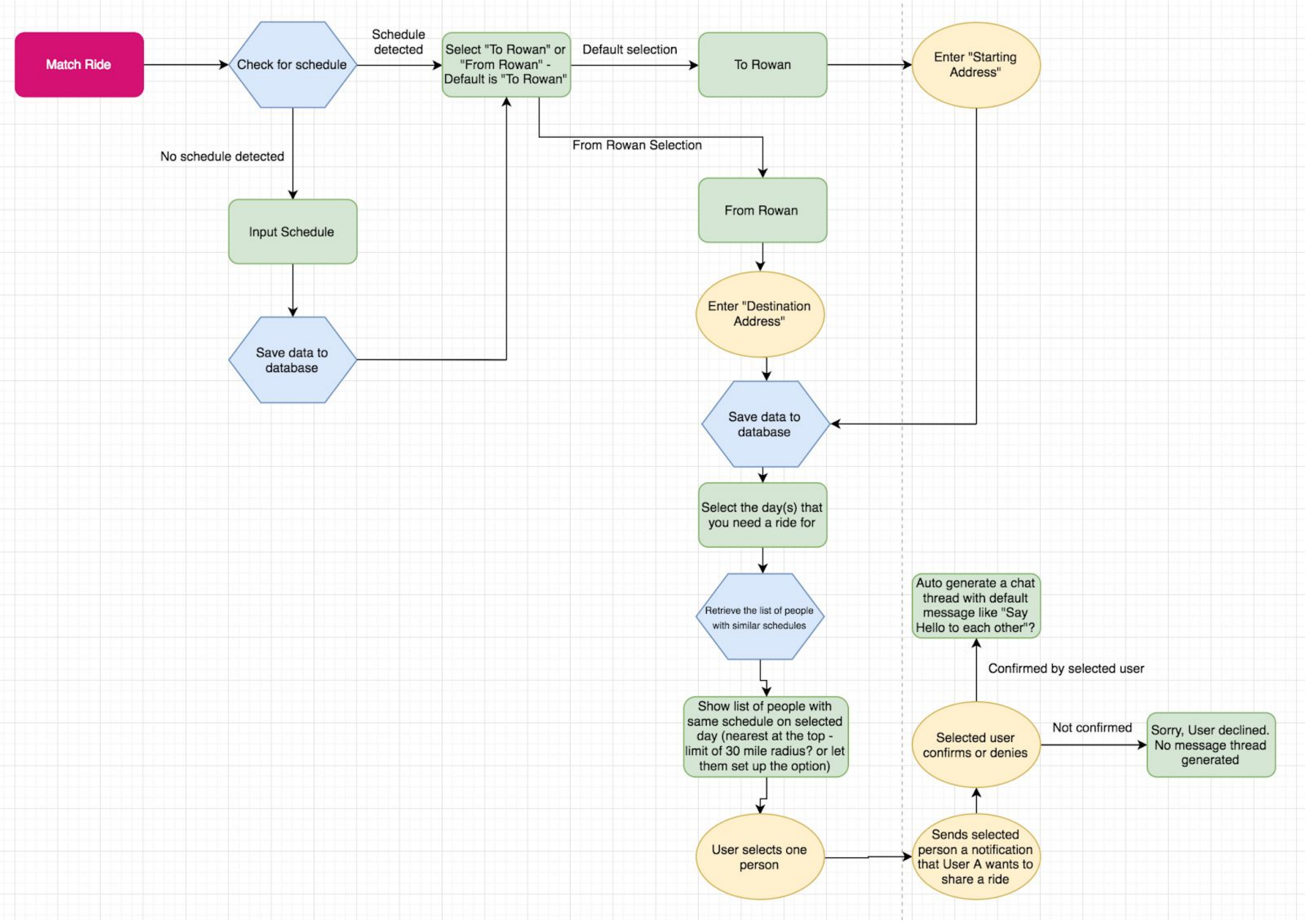
Users can select the chat button from the home screen that bring them to a screen that shows them all of their active chats sorted by the most recently posted in, with any newly formed chats at the top of the list. Once a user selects a chat that they want to post in the app will redirect to the chat page where the user can type a message along with read old ones.

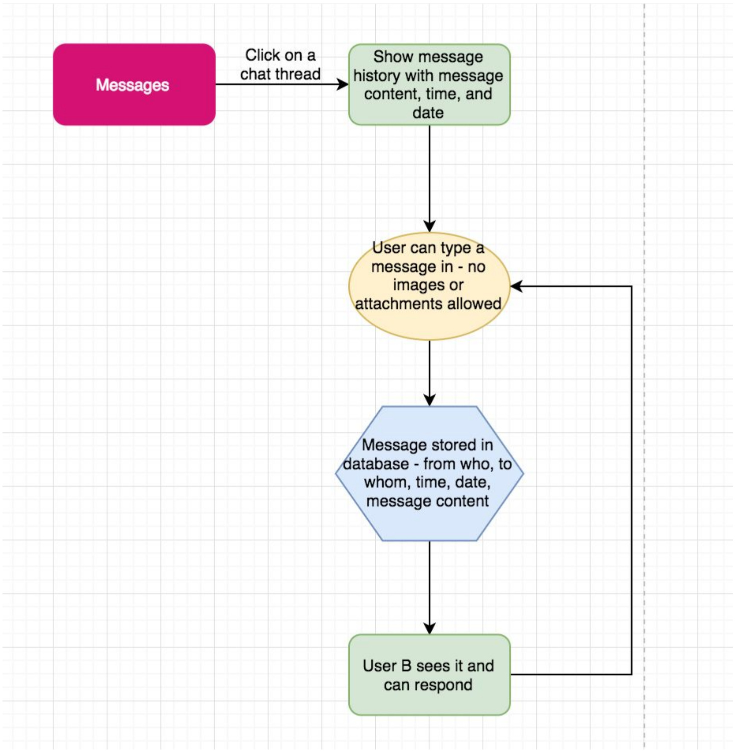
Other pages that can be selected from the home screen are the profile edit screen, a settings screen. The profile edit screen is largely the same as when the user first entered information when their profile was first created. The user can choose to revisit this screen to adjust their profile whenever needed. The settings screen is used to control any app related options such as notifications, or accessibility settings.

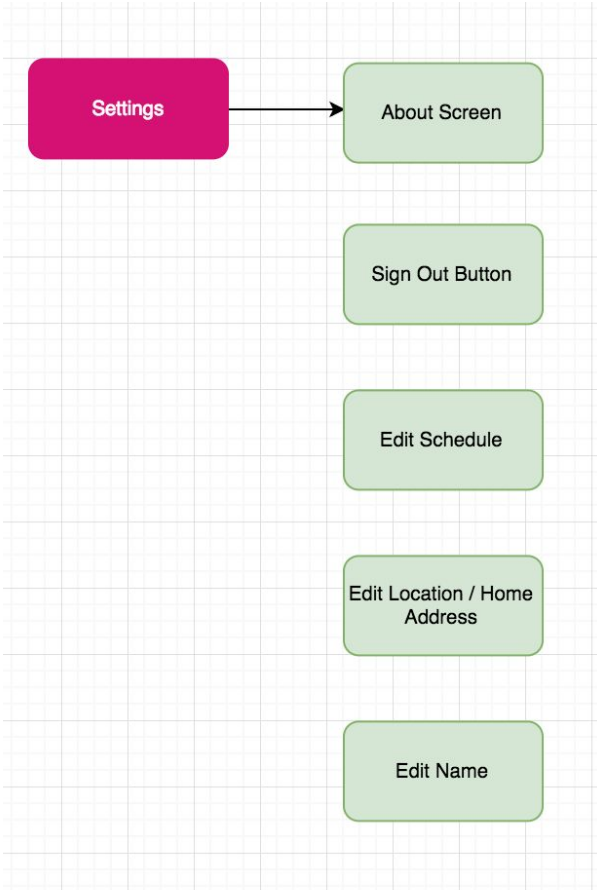
Flow Diagram


Start
Check login token (stored locally)
Already logged in
Main Menu
User has to choose
Match Ride
Not logged in
Login Prompt
Messages
Go to Main Menu after profile creation
User exists in database
Store data in database (FN, LN, Email, Addr, Schedule)
Does user exit?
Profile
User doesn't exist in database
Press "Save" Button
Settings
Disclaimer Window
Accept
Initial Profile Creation Page
User Enters Data - (Schedule, Address)
Decline
QUIT


Profile
Name, Email, Home Address, Schedule (H. Addr & Sched are empty if user presses "Skip"
Edit anything except schedule
Edit within screen
Edit schedule
New Screen: Edit Schedule
Save data to database
Exit (Back Button)
Main Menu


Match Ride
Check for schedule
Schedule detected
Select "To Rowan" or "From Rowan" - Default is "To Rowan"
Default selection
To Rowan
Enter "Starting Address"
No schedule detected
From Rowan Selection
Input Schedule
From Rowan
Save data to database
Enter "Destination Address"
Save data to database
Select the day(s) that you need a ride for
Retrieve the list of people with similar schedules
Auto generate a chat thread with default message like "Say Hello to each other"?
Confirmed by selected user
Show list of people with same schedule on selected day (nearest at the top - limit of 30 mile radius? or let them set up the option)
Selected user confirms or denies
Not confirmed
Sorry, User declined. No message thread generated
User selects one person
Sends selected person a notification that User A wants to share a ride


Messages
Click on a chat thread
Show message history with message content, time, and date
User can type a message in - no images or attachments allowed
Message stored in database - from who, to whom, time, date, message content
User B sees it and can respond


Settings
About Screen
Sign Out Button
Edit Schedule
Edit Location / Home Address
Edit Name

Technology Stack

The frontend for the app is being developed in Android Studio. The user interface is developed using Dart and the Flutter SDK from Google. Since the app is intended to run on android phones, the app is test using emulated hardware using the latest version of Android.

The app uses RESTful APIs such as Google Maps to determine whether two users are an appropriate match based on their distance. Google Maps is also used to provide the user with a visual of what to expect from an upcoming route. We utilize HTTP requests to pull info from the databases to compares users schedules to see if they are suitable.

The application has a Java-based backend that utilizes the Spring Boot framework. We chose Java because of its robustness and the development team's familiarity with the language. Spring Boot was chosen due to its dependency injection capabilities and integration with Hibernate.

The database for the application is hosted on an AWS server. We decided to use MySQL as our database over other alternatives (Apache Cassandra, MongoDB, etc.) due to its simplicity and our expected user count. If the number of users increase to an amount where a relational database is not desirable, we plan to migrate to Cassandra (which is known for scalability). The database schema includes tables for profiles, addresses, schedules, groups and messages. The app uses Hibernate as an ORM and to communicate with MySQL.

Backend Information

The backend is created using Spring Boot and will handle the flow of data that is required to use the app. The backend will communicate with the apps database on AWS, and be responsible for query the database for information that is presented to the user on the front end. It will also need to take information that the user enters into the GUI and store it in the appropriate table on the database.

The backend will also handle the user matching that is integral to the functionality of the app. Any calculations that are performed by the app to determine which users would be strong carpool candidates will performed from the back end.

Authentication and Security

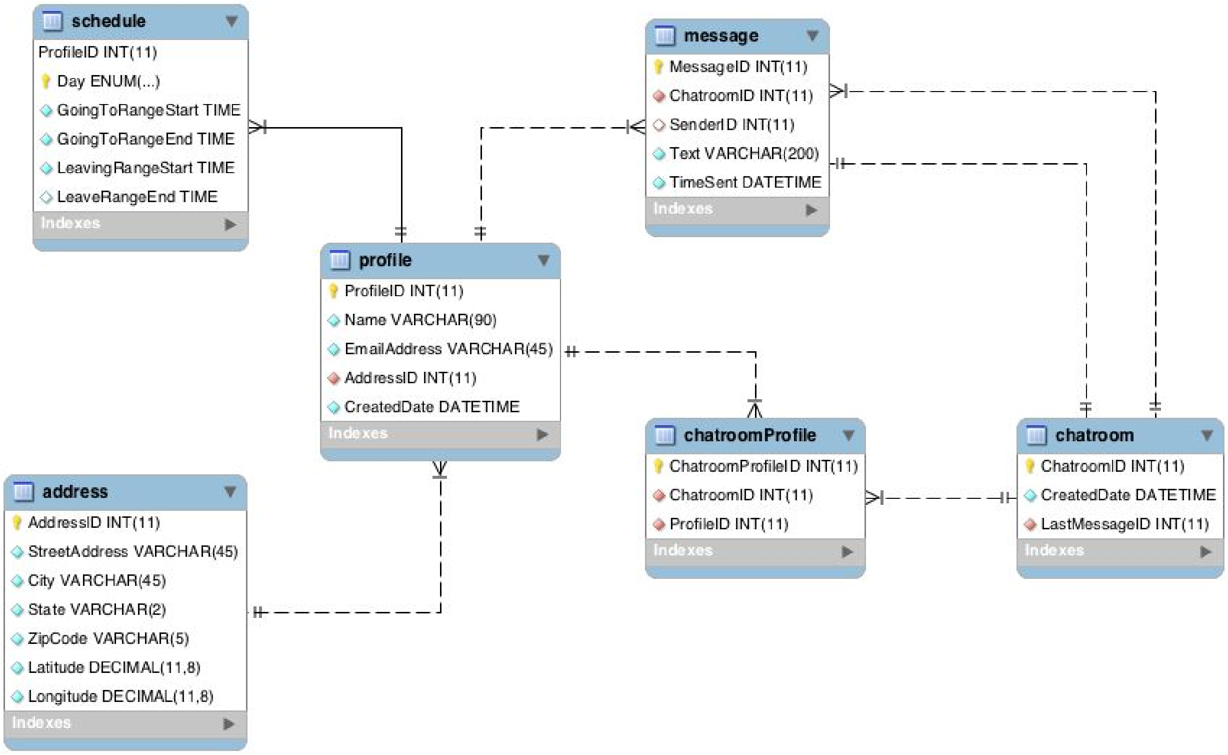
The app's user authentication will be performed using OAuth, and the user will be using their normal Rowan account logins to use the app. Doing so, we don't need to store a user's login or password. Additionally, this app is not intended for those who are not students at Rowan, so there is no need for users to create their accounts from within the app.

Input and Output

Input: The primary method of user input is using buttons to navigate the app’s menus. The app also receives text inputs when the user is typing a message in the chat screen or adding information their profile .

Output: The output provided to the user is a list of other users that are a good match to start a carpool with. The user will also be able to view relevant locations from the map screen, along with a suggested route that could be loaded into google maps.

Database Schema



For schedule, ProfileID and Day form a composite primary key. The Enums for Day are Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday.

There are two relationships between chatroom and message: a 1-to-1 relationship to track the last message sent for a chatroom and a 1-to-Many since a chatroom can have many messages.

The chatroomProfile table exists as a bridging table for the Many-to-Many relationship between profile and chatroom.

Restful Endpoints

The endpoints are labeled in the form of REQUEST TYPE /endpoint/url/. Inputs to an endpoint are denoted by brackets. For example, {user_id} expects an integer. Additional parameters will be handled in through query strings and/or form data (for post requests).

GET /profile/{profile_id}

Returns information about the profile corresponding to the profile_id in the form of JSON.

GET /address/{profile_id}

Returns the address for the specified profile in the form of JSON.

GET /profile/{profile_id}/schedule

Returns the current schedule for the profile in the form of JSON.

GET /profile/{profile_id}/matches

Returns a list of potential matches for the specified profile in the form of JSON. Profiles are matched based on distance and proximity.

GET /chat/{chatroom_id}/messages

Returns a list of messages in the chat conversation that corresponds to the chatroom_id. Output is in form of JSON.

GET /chat/{chatroom_id}

Returns the profiles that are in the chatroom in the form of JSON.

POST /profile/{profile_id}

Updates the current user information. No output.

POST /profile/{profile_id}/schedule

Updates the current schedule for the user. No output.

POST /chat/{chatroom_id}/messages

Adds a new message for the specific chatroom. No output.

POST /chat/{group_id}/{profile_id}

Adds a new profile to the group. No output.

POST /address/{profile_id}

Updates the address for the specified profile_id. No output.