

RideMe Final Report



Course: CSCI 485 – Software Design

Instructor: Dr Othman Soufan

Group B

Cole Beasant	202001106	x2020asw@stfx.ca
Devon Eaton	201903479	x2019cjo@stfx.ca
Liam Johnston	201902759	x2019brj@stfx.ca
Goutham Regu	202005622	x2020ftn@stfx.ca

Table of Contents

Section 1: Introduction	3
Section 2: UI Description	3
2.1 New User Signup	3
2.2 Existing User Login	5
2.3 Logged in Prospective Post Owner	7
2.4 Logged in Post Owner	8
2.5 Logged in Prospective Passenger	13
2.6 Logged in User Wanting to Leave a Review	16
2.7 Logged in User Wanting to View and Update their Profile Information or Logout	19
Section 3: Future Work	21
Appendix: SRS, Architecture and Low-Level Design, and Team Meetings Documents	22

List of Figures:

Figure 1: RideMe landing page	4
Figure 2: Signup form	5
Figure 3: Postings page	6
Figure 4: Reset password security question	7
Figure 5: Add posting page	8
Figure 6: Conversations page	9
Figure 7: Messages page	10
Figure 8: User details page	11
Figure 9: My driver trips page	12
Figure 10: Manage posting page	13
Figure 11: Posting details page	14
Figure 12: Notifications for approval as passenger and closed posting	15
Figure 13: My passenger trips page	16
Figure 14: Users to review page	17
Figure 15: Add review page	18
Figure 16: Another user details page	19
Figure 17: My profile page	20

Section 1: Introduction

RideMe is a ride sharing web application that is designed to address the commuting needs of StFX students who do not have access to their own means of transportation. The current options of students in that scenario are either Maritime Bus or to post a message in the StFX Carpool Facebook group and hope that somebody responds. RideMe is a formal version of the StFX Carpool Facebook group, where those who plan on commuting and have available seats in their vehicle can add a posting with a given date and time, and those who are in need of a ride will be able to message the post owners to request to be a passenger for their trip. RideMe users can also review each other to mitigate the effect of malicious users. RideMe can also foster the growth of the StFX community as travelling together may form friendships between people who may otherwise not have ever met.

Section 2: UI Description

RideMe's UI was built to address its 7 use cases¹:

1. New User Signup
2. Existing User Login
3. Logged in Prospective Post Owner
4. Logged in Post Owner
5. Logged in Prospective Passenger
6. Logged in User Wanting to Leave a Review
7. Logged in User Wanting to View and Update their Profile or Logout

The following is a description of how RideMe is used by users of each use case, along with screenshots of the UI during their journey. Note that a RideMe user may be more than one use case at any given time.

2.1 New User Signup

When a user opens RideMe, they are directed to the *landing page*, shown in *Figure 1*.

¹ Note that prospective post owner is a RideMe user who wants to be a post owner and prospective passenger is a RideMe user who wants to be an approved passenger, which is a passenger who has been approved for the trip by the post owner

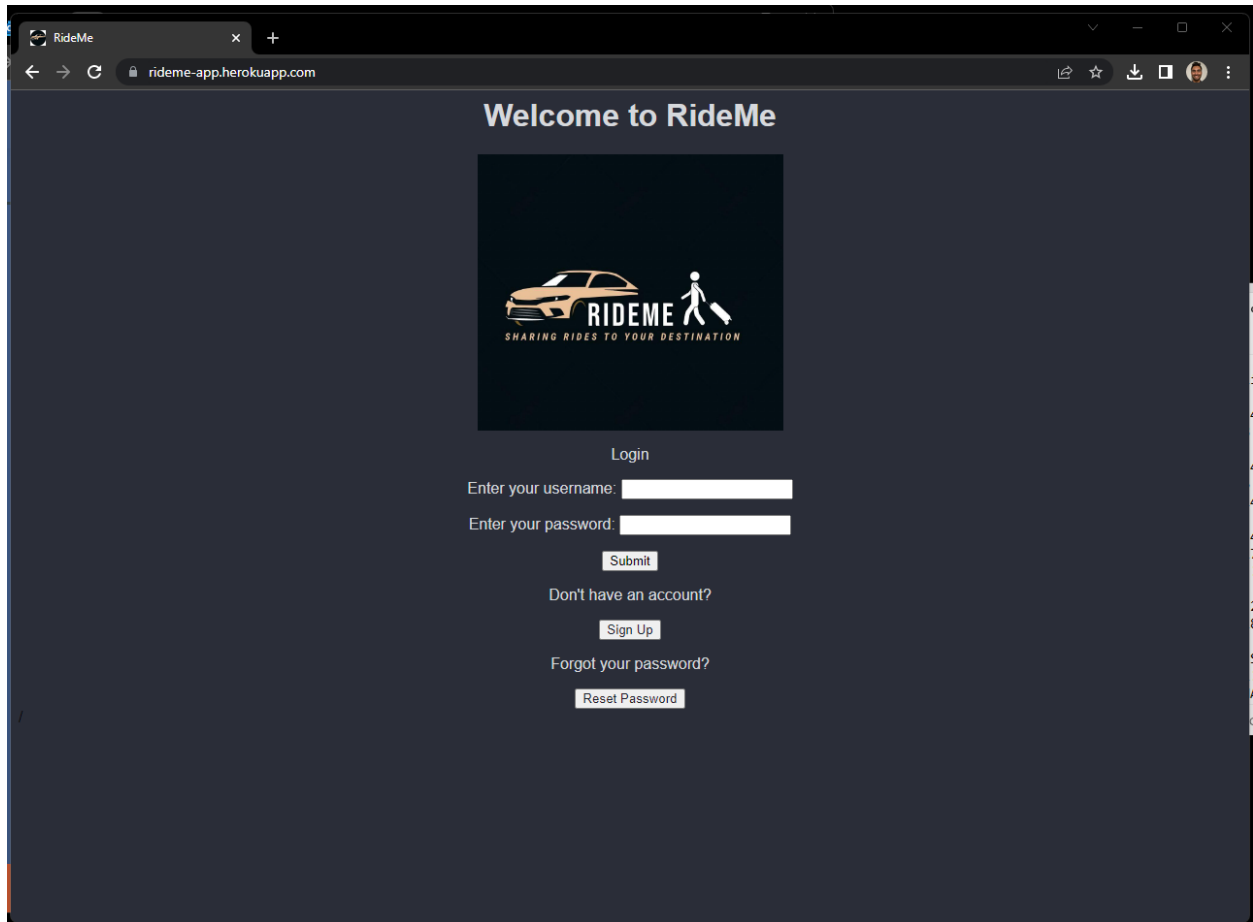


Figure 1: RideMe landing page

If they do not have an account, they will hit the *Sign Up* button, and RideMe will navigate them to the signup form shown in *Figure 2*. Users can then enter their information into the form to create an account.

The image shows a web browser window with the URL `rideme-app.herokuapp.com/signup/`. The page has a dark blue background and is titled "Welcome to RideMe". Below the title, it says "Enter the following information to register:". The form contains the following fields and controls:

- Enter your username:
- Enter your First Name:
- Enter your Last Name:
- Enter your email address:
- Enter your password:
- Confirm your password:
- Enter a security question:
- Enter the answer to the security question:
- Upload your profile picture (optional): No file chosen
-

Below the form, there is a link: "Already have an account? [Login](#)".

Figure 2: Signup form

If there is an issue with their form that caused the account to not be created, RideMe will keep the user on the page shown in *Figure 2* with an error message that notifies the user of the issue with the form. If their account was successfully created, they will be redirected back to the *landing page* with a message informing them that their account was successfully created.

2.2 Existing User Login

Users can login to RideMe on the *landing page* with their username and password that they gave themselves in the signup form. If the login attempt was successful, they will be redirected to the *postings* page shown in *Figure 3* and are informed of their successful login.

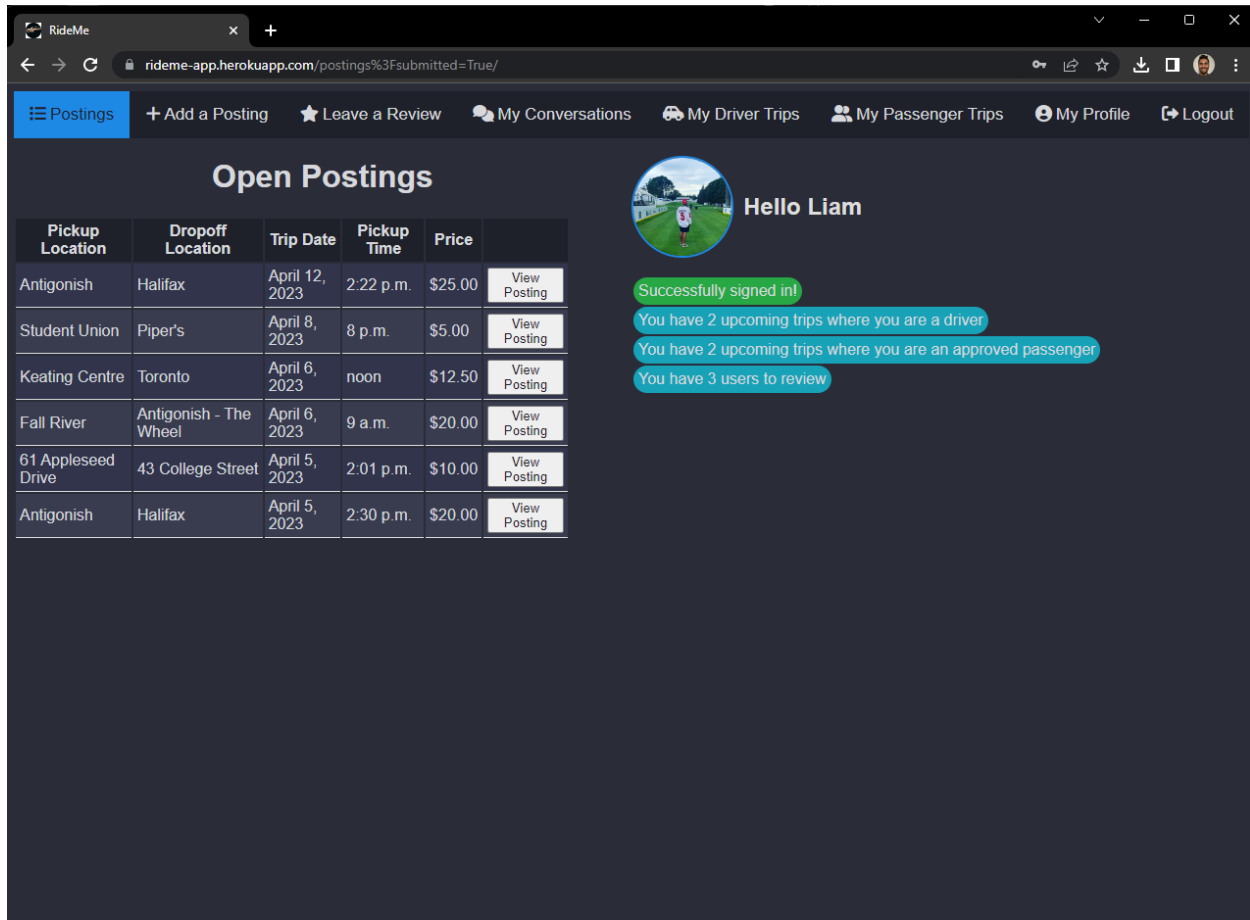


Figure 3: Postings page

The open postings query on the *postings* page lists all postings with trip dates in the future that have available seats.

If the login attempt was unsuccessful, the user will remain on the landing page with a message informing them of their unsuccessful login attempt. If they forget their password, they can hit the *reset password* button on the *landing page*. RideMe will then prompt the user to enter their username. If their username is one of an existing RideMe user, RideMe will show the security question associated with the user profile containing the entered username and prompt the user to enter the answer to the security question, as shown in *Figure 4*.

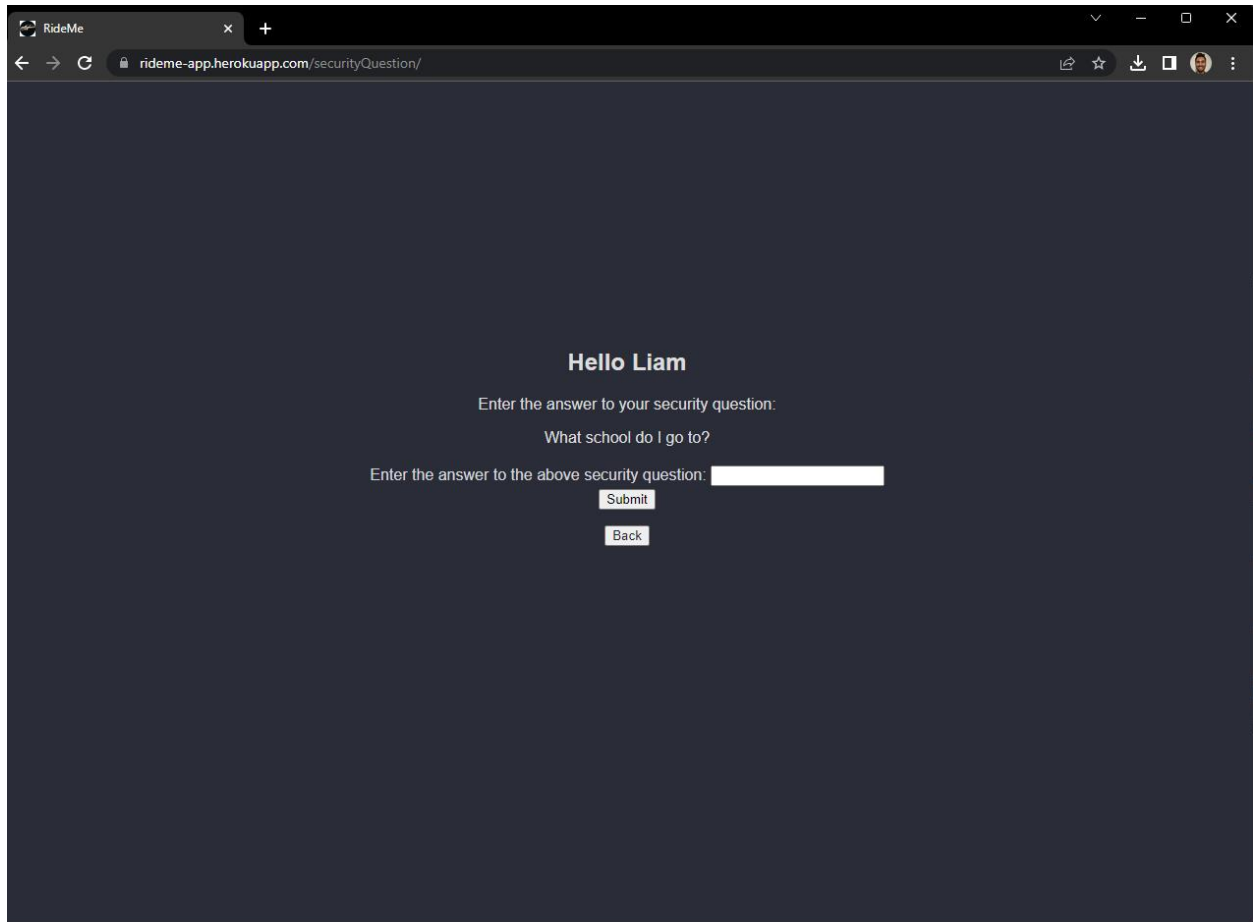


Figure 4: Reset password security question

If the user answers to the security question correctly, they will be directed to a page to enter and confirm their new password. If both passwords are the same, the user will be redirected to the *landing page* with a message informing them that their password was successfully reset. If the passwords are not the same, the user will be notified of this and will be prompted to enter the passwords again.

If the user answers the security question incorrectly, they will be notified and remain on the page shown in *Figure 4*.

2.3 Logged in Prospective Post Owner

A logged in prospective post owner will hit the *Add a Posting* button on the NavBar, which is present on all pages that are accessible after login. Hitting this button will direct the user to the *add posting* page shown in *Figure 5*.

RideMe

rideme-app.herokuapp.com/postings/addPostings%3Fsubmitted=True/

Postings +Add a Posting ★ Leave a Review My Conversations My Driver Trips My Passenger Trips My Profile Logout

Add a Posting

Enter the following information to add a new posting:

Enter the number of available seats for your trip: 2

Enter the date the trip will be occurring: 04 / 05 / 2023

Enter the time the trip will be occurring: 09 : 59 PM

Enter the general location where you can pick up passengers: Antigonish

Enter the general location where you can drop off passengers: Halifax

Enter the price you wish passengers to pay for the trip: 10

Enter the make and model of the vehicle you will be making the trip with: Honda Civic

Submit

Figure 5: Add posting page

If the information was entered incorrectly (i.e. number of available seats not being an integer, trip date being in the past, trip price being negative, etc), the prospective post owner will remain on the *Add a Posting* page with a message notifying them of the issue. If the information was entered correctly, the prospective post owner is now a post owner, and will be redirected to the *postings* page with a message indicating that the posting was successfully added and their number of upcoming driver trips being incremented by 1.

2.4 Logged in Post Owner

Logged in post owners can receive messages from prospective passengers regarding their trips. When they are on the *postings* page, they will receive a notification indicating how many unread messages they have. They can select to view their conversations by hitting the *My Conversations* button on the NavBar, which will navigate them to the *conversations* page shown in Figure 6.

Unread Messages	My Post	User		Posting Open	Pickup Location	Dropoff Location	Trip Date	Pickup Time	Trip Price	
Yes	Yes	@ritanigam20	View Messages	Yes	Antigonish	Halifax	April 5, 2023	9:59 p.m.	\$10.00	View Associated Posting
Yes	Yes	@Sie123	View Messages	Yes	Antigonish	Halifax	April 5, 2023	9:59 p.m.	\$10.00	View Associated Posting
No	Yes	@gt24	View Messages	Yes	Antigonish	Antigonish	April 1, 2023	2:30 p.m.	\$10.00	View Associated Posting
No	Yes	@cbeasant	View Messages	Yes	Antigonish	Halifax	April 6, 2023	4:26 p.m.	\$10.00	View Associated Posting
No	No	@osoufan	View Messages	Yes	Antigonish	Halifax	April 12, 2023	2:22 p.m.	\$25.00	View Associated Posting
No	Yes	@camj	View Messages	No	Antigonish	Fall River	April 7, 2023	12:30 p.m.	\$30.00	View Associated Posting
No	Yes	@cbeasant	View Messages	No	Antigonish	Fall River	April 7, 2023	12:30 p.m.	\$30.00	View Associated Posting
No	No	@camj	View Messages	Yes	Fall River	Antigonish - The Wheel	April 6, 2023	9 a.m.	\$20.00	View Associated Posting
No	No	@bren101	View Messages	Yes	bloomfield	muhlrooney	March 29, 2023	1:15 p.m.	\$0.00	View Associated Posting
No	No	@nevons	View Messages	No	Antigonish	Halifax	March 31, 2023	7:01 p.m.	\$20.00	View Associated Posting
No	No	@nevons	View Messages	Yes	Antigonish	Halifax	March 31, 2023	6:56 p.m.	\$20.00	View Associated Posting

Figure 6: Conversations page

As shown, the post owner can see which conversations have messages they have yet to read and if the conversation is regarding their post or not. A post owner would then select a conversation where they have unread messages in a conversation regarding their post and be directed to the *messages* page of their selected conversation, as shown in *Figure 7*.

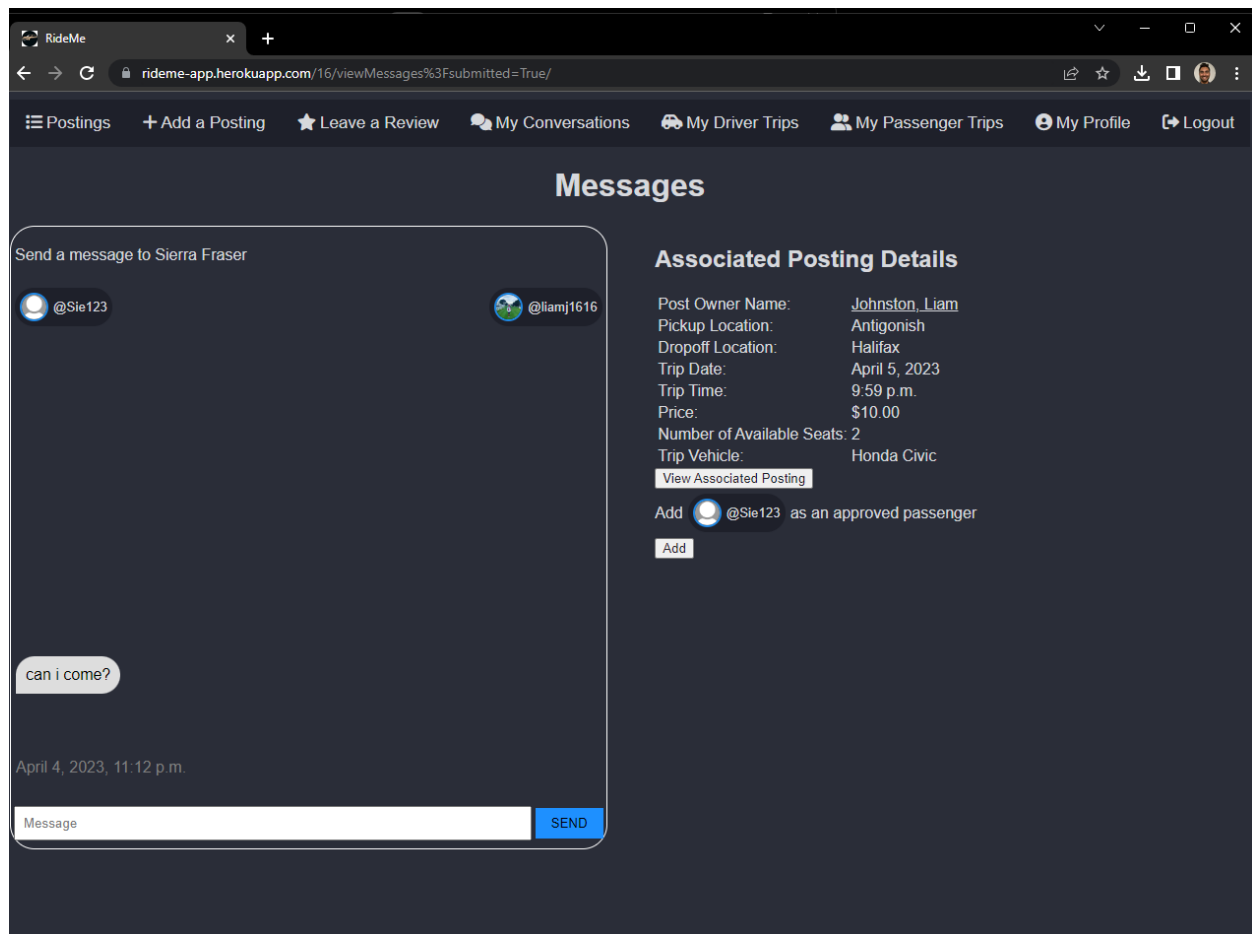


Figure 7: Messages page

The post owner can view the prospective passenger's profile by hitting any button with the prospective passenger's username and profile picture, which contains information the post owner can use to make a decision on if they want to add them as an approved passenger. This page is shown in *Figure 8*.

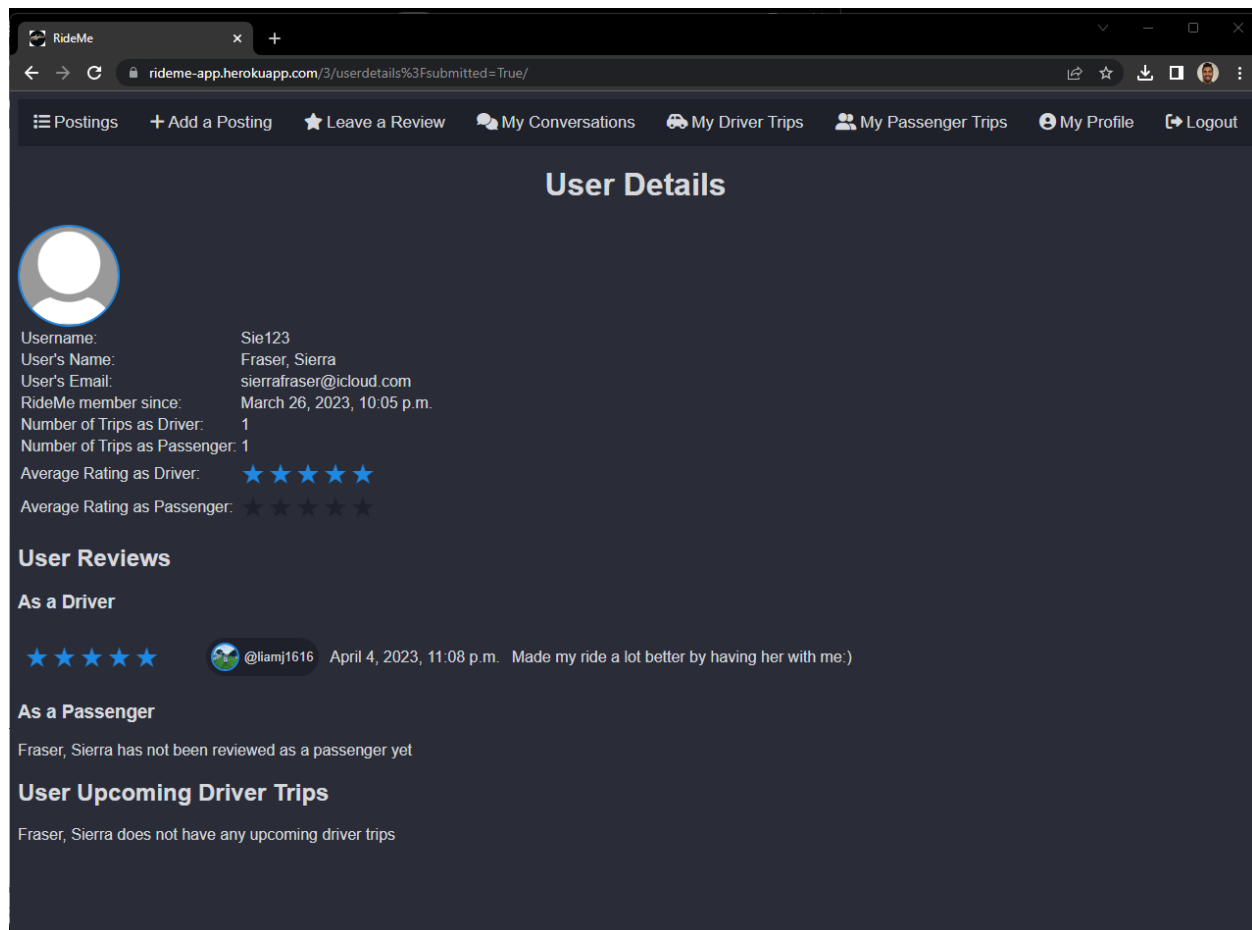


Figure 8: User details page

Based on the information shown in *Figure 8* and the messaging with the prospective passenger, the user can make the passenger into an approved passenger by clicking the *Add* button on the *messages* page shown in *Figure 7*. This action will decrement the number of available seats by 1, add the user to the posting's approved passenger list, and send an automated message in the conversation from the post owner to the newly approved passenger that notifies them of the approval. Note that if the number of available seats is 0, if the post owner hits the *Add* button, they will be notified that the trip is already full and that no more approved passengers can be added.

Additionally, logged in post owners can hit the *My Driver Trips* button on the NavBar to be navigated to the *my driver trips* page shown in *Figure 9*.

[Postings](#)
[+ Add a Posting](#)
[★ Leave a Review](#)
[My Conversations](#)
[My Driver Trips](#)
[My Passenger Trips](#)
[My Profile](#)
[Logout](#)

My Driver Trips

Past Trips that Need Action

Pickup Location	Dropoff Location	Trip Date	Pickup Time	Price	Trip Occurred?
Antigonish	Halifax	April 5, 2023	9:59 p.m.	\$10.00	<input type="button" value="Yes"/> <input type="button" value="No"/>

My Upcoming Driver Trips

Pickup Location	Dropoff Location	Trip Date	Pickup Time	Price	Manage Posting
Antigonish	Fall River	April 7, 2023	12:30 p.m.	\$30.00	<input type="button" value="Manage"/>

My Past Driver Trips

Pickup Location	Dropoff Location	Trip Date	Pickup Time	Price	View Details
Antigonish	Fall River	April 3, 2023	4:10 p.m.	\$10.00	<input type="button" value="Details"/>
Antigonish	Halifax	March 28, 2023	8 p.m.	\$10.00	<input type="button" value="Details"/>

Figure 9: My driver trips page

There are 3 queries on the *my driver trips* page: past driver trips needing action, upcoming driver trips, and past driver trips. Past driver trips needing action are the post owner's postings with trip dates in the past that require the post owner to acknowledge if the trip actually occurred. If the trip occurred, they hit the *Yes* button shown in *Figure 9*, which will mark the posting as "complete", increment the number of trips completed as a driver for the post owner by 1, increment the number of trips as a passenger for all approved passengers by 1, and close all conversations associated with the posting. If the trip did not occur, the post owner will hit the *No* button and the posting will be marked as "cancelled". All users who messaged the post owner regarding this posting will receive a notification in the posting's associated conversation stating that the trip was cancelled.

Upcoming driver trips are the post owner's postings with trip dates in the future. They can hit the *manage* button for postings in this query shown in *Figure 9*, which will navigate them to the *manage posting* page shown in *Figure 10*.

Manage Posting

Pickup Location: Antigonish [Update](#)

Dropoff Location: Fall River [Update](#)

Trip Date: April 7, 2023 [Update](#)

Trip Time: 12:30 p.m. [Update](#)

Number of Available Seats: 0 [Update](#)

Trip Price: 30.00 [Update](#)

Trip Vehicle: Honda Civic [Update](#)

[Cancel Trip](#)

Approved Passengers

- @camj
- @cbeasant

Associated Conversations

Unread Messages	User	Approved?	
No	Chris Johnston @camj	Yes	View Messages
No	Cole Beasant @cbeasant	Yes	View Messages

Figure 10: Manage posting page

In the *manage posting* page, they can update the trip information or cancel the posting. Both of these actions will result in the users who messaged the post owner regarding this posting to be notified through their associated conversation. The *manage posting* page also shows the approved passenger list and all conversations associated with the posting.

Past driver trips are the post owner's postings that are marked as complete.

2.5 Logged in Prospective Passenger

Logged in prospective passengers will view the open postings query on the *postings* page shown in Figure 3. If they find a trip that suits their needs, they will hit the *view posting* button for that trip, which will navigate them to the *posting details* page shown in Figure 11.

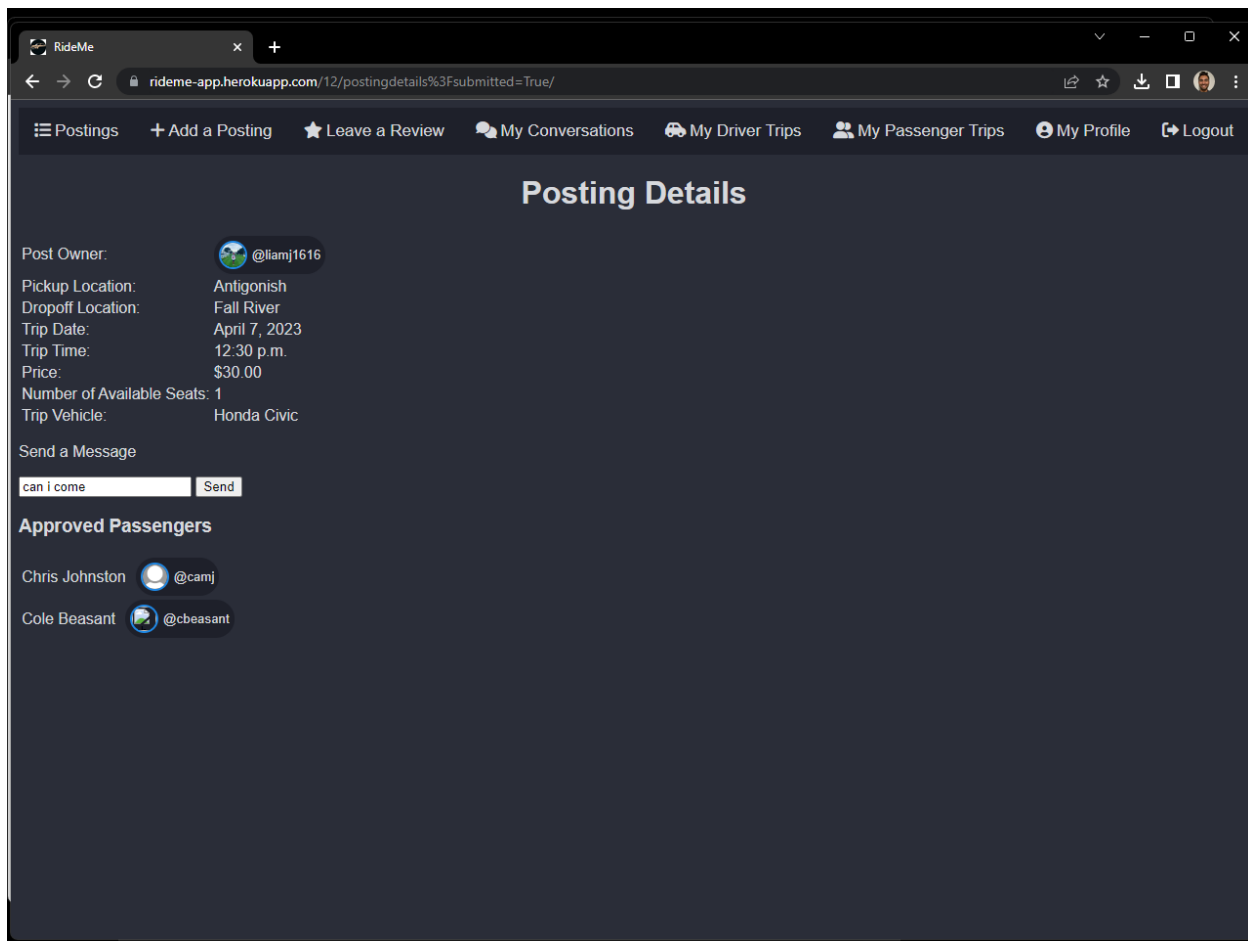


Figure 11: Posting details page

They can view the post owner's and approved passenger's *user details* pages shown in Figure 8 by hitting their user buttons to decide if they would like to request to be a passenger. If they have decided to do so, they can type in a message into the send a message form. This will open a conversation between the prospective passenger and the post owner. Note that if a prospective passenger already has a conversation associated with the posting, RideMe will block them from creating another one.

If the post owner adds to prospective passenger to the approved passenger list, they will receive an automated message to notify them, as shown in Figure 12.

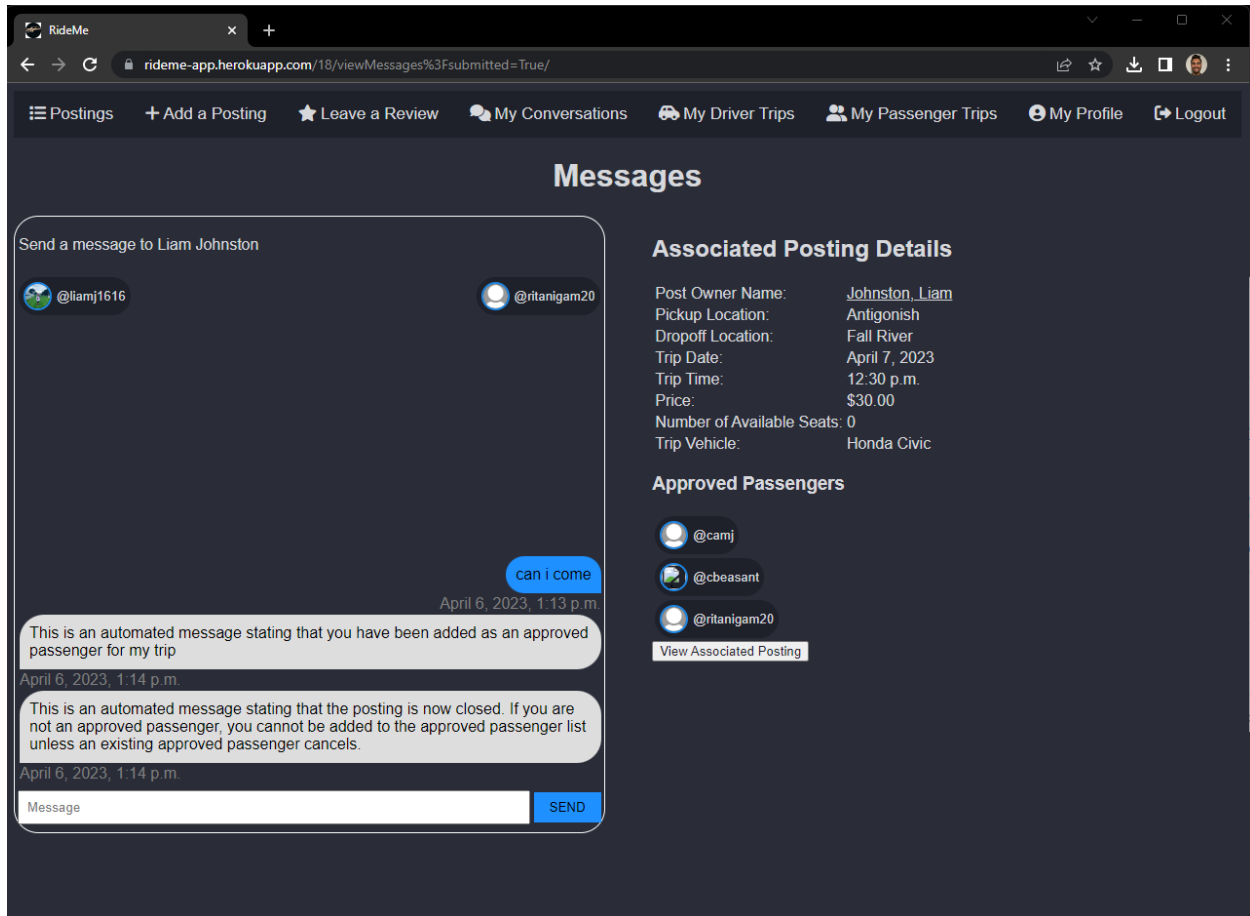


Figure 12: Notifications for approval as passenger and closed posting

Note that because the number of available seats on the posting became 0 by adding this passenger, the posting has been closed so passenger was notified. Passengers will also receive notifications if the post owner updates any trip information or if they cancel the trip. Conversations between post owners and prospective or approved passengers remain open until the post owner marks the posting as complete.

Passengers can also hit the *My Passenger Trips* button on the NavBar to be navigated to the *my passenger trips* page shown in Figure 13.

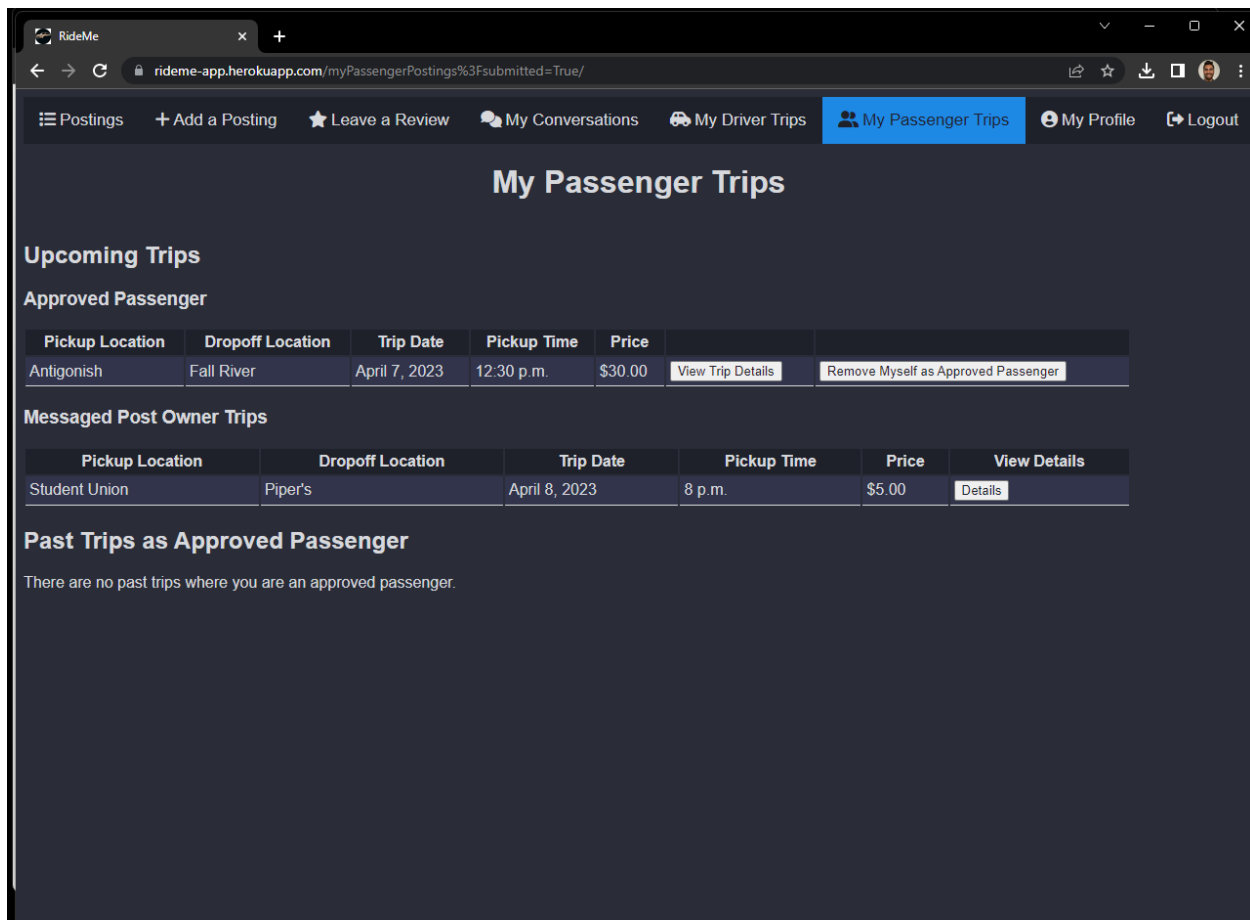


Figure 13: My passenger trips page

As with the *my driver trips* page, there are 3 queries. Upcoming approved passenger trips are postings with trip dates in the future where the user is on the approved passenger list. Upcoming messaged post owner trips are trips with trip dates in the future where the user has messaged the post owner but has yet to be selected as an approved passenger. Past trips as an approved passenger are postings marked as complete where the user was on the approved passenger list.

2.6 Logged in User Wanting to Leave a Review

Logged in users wanting to leave a review can hit the *Leave a Review* button in the NavBar, which will direct them to the *users to review* page shown in Figure 14.

The screenshot shows a web browser window with the URL `rideme-app.herokuapp.com/usersToReview%3Fsubmitted=True/`. The navigation bar includes links for Postings, Add a Posting, Leave a Review (highlighted), My Conversations, My Driver Trips, My Passenger Trips, My Profile, and Logout. The main heading is 'Users to Review'.

User	Interaction Type			Trip Date	Trip Pickup Location	Trip Dropoff Location
horne, Brenden @bren101	My Driver	Leave Review	Dismiss	March 29, 2023	bloomfield	muhlrooney
Beasant, Cole @cbeasant	My Passenger	Leave Review	Dismiss	April 3, 2023	Antigonish	Fall River
Fraser, Sierra @Sie123	My Passenger	Leave Review	Dismiss	April 5, 2023	Antigonish	Halifax
Nigam, Rita @ritanigam20	My Passenger	Leave Review	Dismiss	April 5, 2023	Antigonish	Halifax

Figure 14: Users to review page

The users to review query lists all users the logged in user has had a conversation with regarding postings that have trip dates in the past. Users can elect to review the user by hitting the *Leave Review* button or remove the query entry by hitting *dismiss*, which will permanently delete the object in the database, meaning that the user will now never be able to review the other user regarding that posting. Hitting the *Leave Review* button will direct the user to the *add review* page shown in Figure 15.

The screenshot shows a web browser window with the URL `rideme-app.herokuapp.com/32/addReview%3Fsubmitted=True/`. The navigation bar includes links for Postings, Add a Posting, Leave a Review, My Conversations, My Driver Trips, My Passenger Trips, My Profile, and Logout. The main heading is "Leave a review for Rita Nigam:". Below this, there is a rating input field set to 2.5 and a text area containing the review: "Doesn't stop talking which might be good for some people but is annoying for me". A "submit" button is located at the bottom left of the text area.

Figure 15: Add review page

Once the user submits the review, it will be added to the reviewed user's profile and their average rating will be updated, as shown in *Figure 16*.

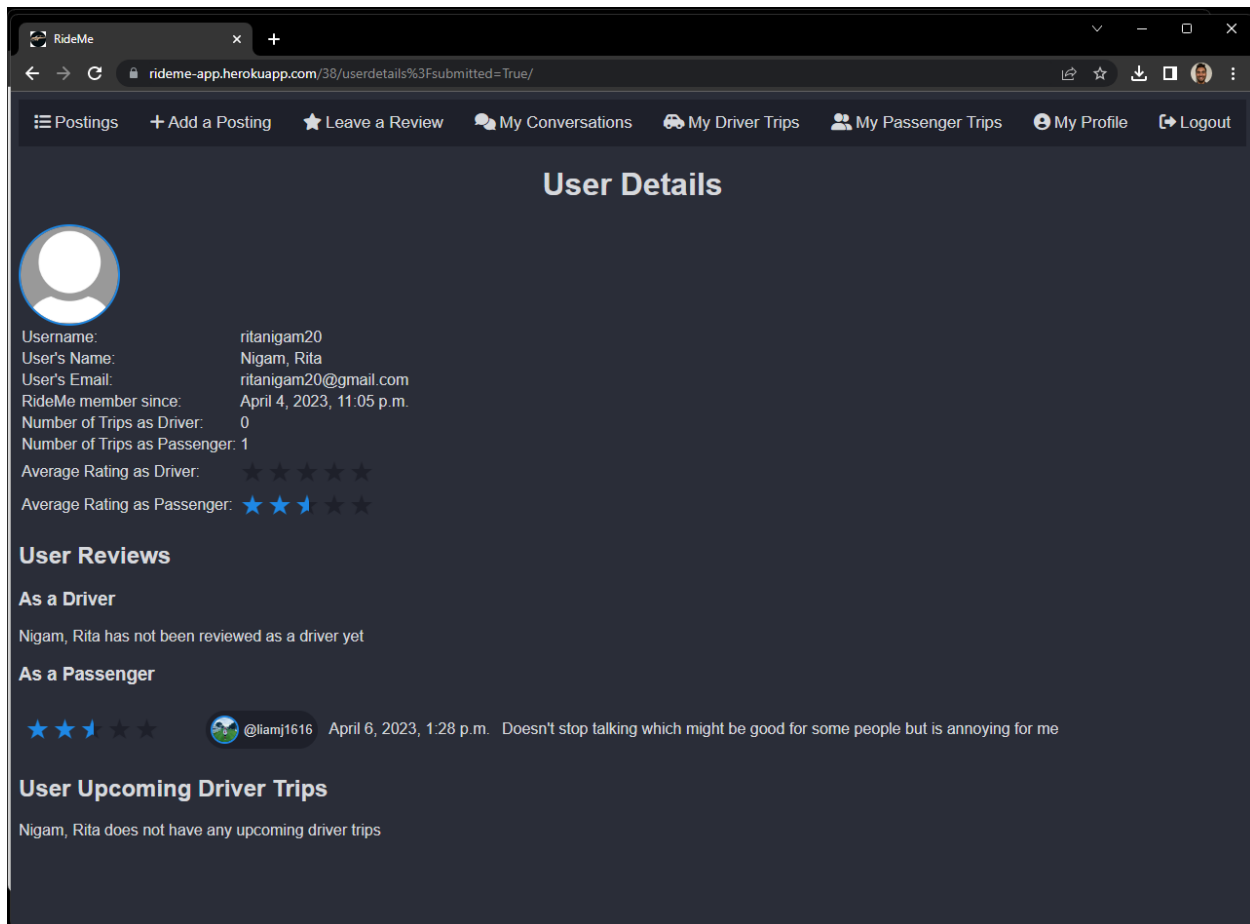


Figure 16: Another user details page

2.7 Logged in User Wanting to View or Update their Profile or Logout

A logged in RideMe user can logout at any time by hitting the *Logout* button on the top right corner of the NavBar. Hitting the *Logout* button will redirect the user to the *landing page* with a message notifying them of their successful logout.

Users can view their profile by hitting the *My Profile* button on the NavBar, which will navigate them to their *my profile* page shown in *Figure 17*.

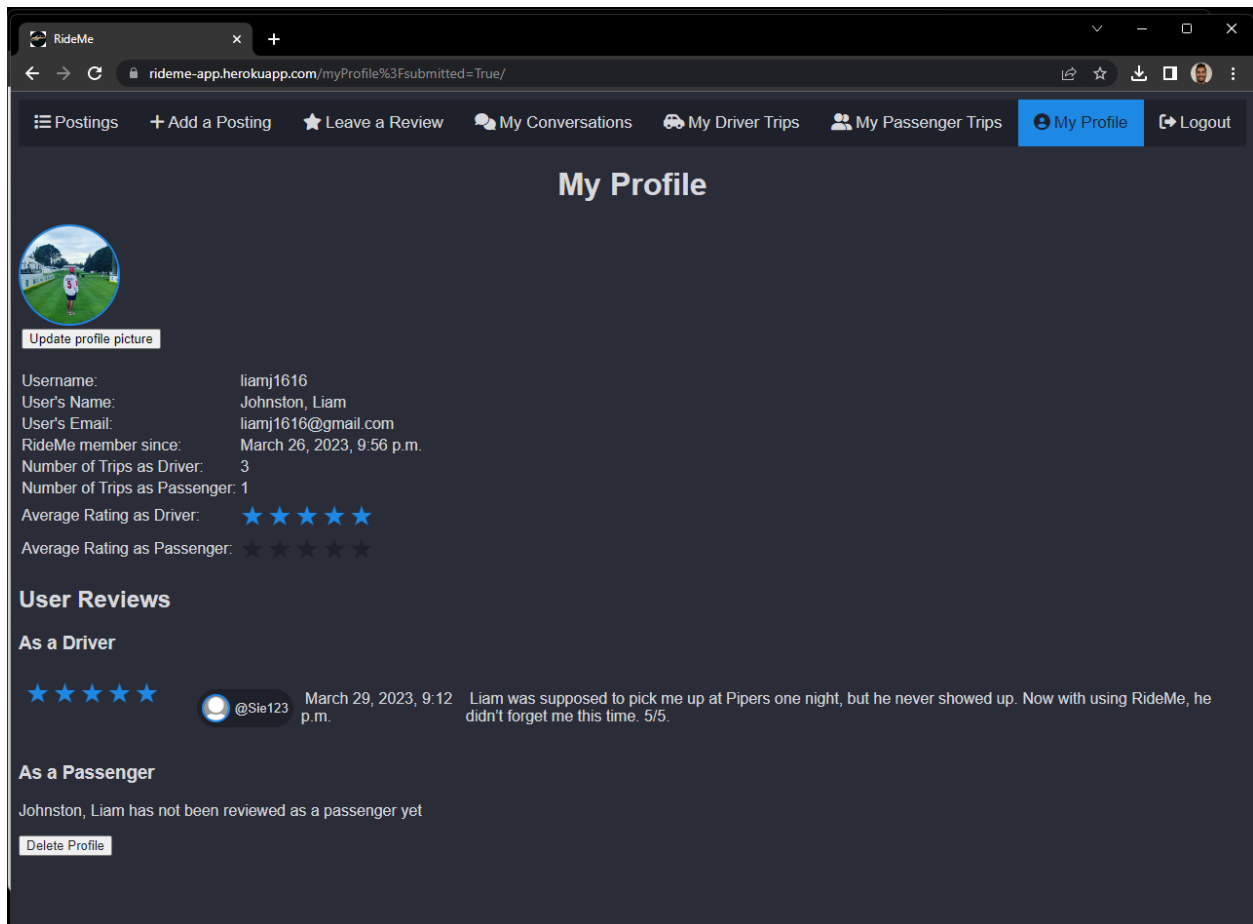


Figure 17: My profile page

As shown, users can see their profile information and their reviews. They also have the option to edit their profile picture or delete their account.

Section 3: Future Work

While RideMe is effective in addressing commuting needs, there are many ways that more time and resources could lead to an improved system.

The first would be to launch RideMe as a mobile application. Given conversations with friends and family, it is highly unlikely that many would make the effort to periodically check a website, for drivers in particular. However, if RideMe was launched as a mobile application, it is much more likely that it would be used, especially since notifications could be sent directly to a user's mobile device, which is something that almost everyone checks very often.

An issue that arose with RideMe is that as more data starts to flood into the site, the queries begin to slow down. This could be addressed by implementing both active and archived data tables for the data models in RideMe. This would allow queries to only need to parse the active tables and the archived tables would only need to be queried if the user specifies a request for their historic data.

Additionally, it was discovered that it is best practice to store time data in Greenwich Mean Time in the database as this avoids the issue of Daylight Savings Time. Currently, if a RideMe user sends a message at 5:08 pm AST, RideMe will display the message timestamp as 8:08 pm, given that 5:08 pm AST is 8:08 pm in Greenwich Mean Time. It was researched that I/O for time data should be done in this fashion: when the user inputs a time, get the time zone of their location then convert the time to Greenwich Mean Time and store in the database and for querying a time, get the location of the user, then query the time in the database and apply the time zone that the user is in to the time data prior to displaying the time data to the user. This was not accomplished in RideMe, but it is something that could be addressed in the future.

Finally, feedback from professor Dr Othman Soufan and other students in the class noted that given RideMe's extensive set of features, the steps for accomplishing use case goals can be confusing. This can be addressed in future versions.

Appendix

SRS, Architecture and Low-Level Design, and Team Meetings Documents

Software Requirements Specification

for

RideMe

Version 1.0

Prepared by

Group Name: Group B

**Cole Beasant
Devon Eaton
Liam Johnston
Goutham Regu**

**202001106
201903479
201902759
202005622**

**x2020asw@stfx.ca
x2019cjo@stfx.ca
x2019brj@stfx.ca
x2020ftn@stfx.ca**

Instructor: Othman Soufan

Course: CSCI 485 – Software Design

Date: February 6, 2023

Contents

CONTENTS	II
REVISIONS	II
RIDEME LOGO	III
1 INTRODUCTION	1
1.1 DOCUMENT PURPOSE	1
1.2 PRODUCT SCOPE	1
1.3 INTENDED AUDIENCE AND DOCUMENT OVERVIEW.....	1
1.4 DEFINITIONS, ACRONYMS AND ABBREVIATIONS.....	2
1.5 DOCUMENT CONVENTIONS	2
1.6 REFERENCES AND ACKNOWLEDGEMENTS	2
2 OVERALL DESCRIPTION	3
2.1 PRODUCT OVERVIEW	3
2.2 PRODUCT FUNCTIONALITY	3
2.3 DESIGN AND IMPLEMENTATION CONSTRAINTS	3
2.4 ASSUMPTIONS AND DEPENDENCIES	4
3 SPECIFIC REQUIREMENTS	5
3.1 EXTERNAL INTERFACE REQUIREMENTS	5
3.2 FUNCTIONAL REQUIREMENTS	5
3.3 USE CASE MODEL.....	8
4 OTHER NON-FUNCTIONAL REQUIREMENTS	14
4.1 PERFORMANCE REQUIREMENTS.....	14
4.2 SAFETY AND SECURITY REQUIREMENTS.....	14
4.3 SOFTWARE QUALITY ATTRIBUTES	14
APPENDIX A – DATA DICTIONARY	15
APPENDIX B - GROUP LOG	19

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Version 1.0	Liam Johnston and Devon Eaton	First Version	02/06/23



1 Introduction

This is the version 1.0 of the Software Requirements Specification (SRS) document for the web application **RideMe** version 1.0.

1.1 Document Purpose

The purpose of this document is to provide a blueprint to the development and testing of the **RideMe** application.

1.2 Product Scope

If **StFX** students do not have access to their own means of transportation and are looking to commute to Halifax, their current options are either **Maritime Bus** or post a message in the **StFX Carpool Facebook** group requesting a ride and hope that somebody responds. **RideMe** will be a web application that formalizes the **StFX Carpool Facebook** group. Those who plan on commuting and have available seats in their vehicle can add a posting with a given date and tentative time, and those who are in need of a ride will be able to message the post owners to request to be a passenger for their trip.

In addition to providing a platform that will ease the burden of commuting for **StFX** students, **RideMe** will foster growth of the **StFX** community as travelling together will form friendships between people who may otherwise not have ever met.

1.3 Intended Audience and Document Overview

The document is intended for the development team of **RideMe** to refer to during the development and testing sprints in the process of building the application. Particularly, the team will be often referring to the requirements, use cases, and data dictionary, which are included in Section 3.2, Section 3.3, and Appendix A, respectively.

Furthermore, this document is intended for the project's supervisor, Dr. Othman Soufan, so that he can provide feedback to the blueprint of this project. Again, the most relevant sections for him to read are the ones listed in the paragraph above.

Finally, this document may also be relevant for users if they are interested in **RideMe** prior to its launch, as they can also provide feedback on the plans of the application workflows. The most relevant section for them would be the Introduction in Section 1, the Overall Description in Section 2, and the use cases in Section 3.3, to determine if it makes sense to them and if **RideMe** would be an application they use.

1.4 Definitions, Acronyms and Abbreviations

Term	Definition
Application/RideMe	Interchangeably refers to RideMe, the software application that is being developed
Approved Passenger	A user who has been approved by a post owner to ride with him on his trip
Closed Posting	A posting with no available seats
Completed Posting	A posting where the physical trip has been completed
Open Posting	A posting with available seats
Post/Posting	A post on the RideMe application which lists a ride, along with relevant information such as travel date, tentative pickup time, tentative pickup and dropoff locations, and number of available seats
Post Owner	The user who created the post who is also responsible for its management and is also assumed to be the driver for the trip
Prospective Passenger	A user who wants to become a passenger
Prospective Post Owner	A user who wants to become a post owner
Rating	A number from 0 – 5 rating the experience with another RideMe user with 5 being the best
Review	Written text describing a users experience with another user
Ride/Trip	The physical commute
User	An individual that has chosen to use the RideMe application

1.5 Document Conventions

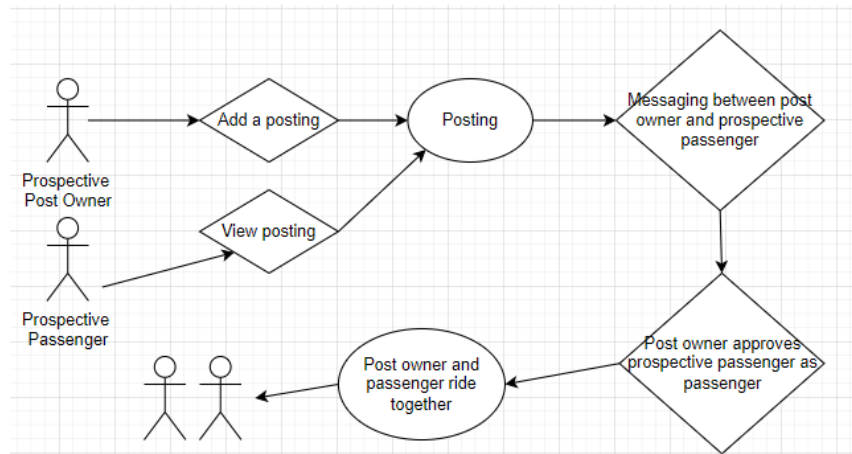
- All names of organizations/entities, such as **RideMe** will be written in **bold**
- Uppercase and italicize when referencing one of the three data models: *User*, *Posting*, and *Conversation*

1.6 References and Acknowledgments

We would like to acknowledge Dr Othman Soufan for providing the IEEE SRS document template and Groups E and F of CSCI485 from the 2021-22 academic year for providing their finished SRS documents to use as an example.

2 Overall Description

2.1 Product Overview



The RideMe application is designed to be a service for people to organize ride sharing. The application is intended to be used for planned trips rather than impromptu trips which makes it less of a competitor to taxi's and **Uber** and more of a competitor to bussing such as **Maritime Bus**. Users create an account and are able to share the seats of their vehicle and communicate with prospective passengers or view the post owners and communicate with them to determine if they are a good fit. Users can see the number of available seats, the status of a posting, the date and time of the trip, and the pickup and drop-off locations.

2.2 Product Functionality

The following will be the major functionalities of **RideMe**:

- Registration, login, and deletion of user accounts
- The ability for users to view existing postings
- The ability for users to add new postings
- The ability for users to message post owners, requesting to be a passenger
- The ability for post owners to manage their postings
- The ability for users to rate and review other users they have interacted with
- The ability for users to view the ratings and reviews of other users

2.3 Design and Implementation Constraints

Application requires an internet connection to use. Limited to capabilities of **Django**. Application is limited to English but could be expanded to include other languages. Additionally, because we are full time students, time is a constraint which will limit our ability to provide out of the box solutions. Furthermore, if post owners would like to charge for their services, they may communicate this through **RideMe**'s messaging but the payment will have to be processed outside of the application.

2.4 Assumptions and Dependencies

It is assumed that **RideMe** will be used by the **StFX** community mainly for students commuting between Halifax and Antigonish. If there becomes a need to scale to more locations, this will be satisfied. Furthermore, the rate and review system implemented in the application will ensure that the impact of malicious users on the **RideMe** community is minimal. If a user has enough negative reviews, they may be banned from the application. It is also expected that post owners and prospective and approved passengers work out specific details through the messaging service. This can include specific meetup times and locations and if the passenger requires extra space to bring large items.

A dependency for **RideMe** is that there must be post owners who are willing to take passengers for their rides.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

RideMe will be posted to the World Wide Web. As a result, users will interface with the application through a web browser, such as **Google Chrome** or **Firefox**. Detailed descriptions of workflows and user options can be found in Section 3.3.

3.1.2 Hardware Interfaces

Any hardware system, such as a laptop or a mobile device, that supports the use of web browsers.

3.1.3 Software Interfaces

As previously mentioned, **RideMe** will be posted to the World Wide Web, so the application will interface with web browsers. Additionally, MongoDB will be used to implement the database for **RideMe** due to the structure of the data models, which is described in detail in Appendix A.

3.2 Functional Requirements

The following are the functional requirements that will be used to develop the **RideMe** application. Note that the listed requirements are considered to be high priority unless otherwise specified.

- 3.2.1 LANDING PAGE:** The landing page for all users upon entering the application. Provides the options to login to an existing account, sign-up for a new account, or reset the password to an existing account if it has been forgotten.
- 3.2.2 SIGN-UP:** If the users on the landing page select to sign-up for a new account, they will be prompted to enter a username, password, first name, last name, email address, and phone number. If they enter all of this information correctly, **RideMe** will notify the user that the account has been successfully created and will redirect the user back to the landing page. In the background, it will create a new instance of a *User* in the database and assign it a userID and a registration time stamp.
- 3.2.3 LOGIN:** The user can enter their username and password into the fields on the landing page to login to their account. Multi-factor authentication using the phone number may be added but this a low priority.
- 3.2.4 RESET PASSWORD:** If an existing user forgets their password, they can select the option on the landing page to reset their password. They will then be directed to a page which requests them to enter their email. Upon doing so, an email will be sent to their email

account with a link for them to reset their password. Upon clicking this link, they will be prompted to enter and confirm their password. When this has been submitted, their password in the database will be updated and the user will be redirected to the landing page. Implementing this requirement is a medium priority.

3.2.5 VIEW RIDE POSTINGS: Upon successful login, users will be directed to a page listing open postings. They will be able to filter the postings by travel date.

3.2.6 ADD RIDE POSTINGS: Upon successful login, prospective post owners will be able to add ride postings. Information that must be included in order to list a posting on the website are the trip date, tentative pickup time, tentative pickup and dropoff locations, number of available seats, and the vehicle to be used for the trip. If they enter all of this information correctly, **RideMe** will notify the user that the posting has been successfully created and will redirect the user back to the page described in 3.2.5. In the background, it will create a new instance of a *Posting* in the database and assign it a postID.

3.2.7 VIEW POST OWNER REVIEWS: The posting will have a button that will send the user to a page listing the post owner's number of trips as a driver, number of trips as a passenger, average rating, all reviews posted to the post owner's page, and how long the post owner has been a member of **RideMe**. It may be implemented to split the reviews and average ratings for when the user was a driver and when the user was a passenger but this is a low priority.

3.2.8 MESSAGE POST OWNER: Prospective passengers who find a posting that satisfies their needs will be able to message the post owner to request to become a passenger. This will be implemented by having a button that the user can click to message post owner. Clicking this button will send the user to a page where they can type in their message, and in the background will create an instance of a *Conversation* in the database and assign it a conversationID. Both users in the conversation will then be able to post messages to this conversation.

3.2.9 VIEW REQUESTED PASSENGER REVIEWS: When a prospective passenger messages a post owner, the post owner will be able to view the same page on the prospective passenger that was described in 3.2.7.

3.2.10 SAVE USERS INTERACTED WITH: When a user first messages a post owner, they will be added to each others list of users interacted with, which will be a viewable list for every user on the website.

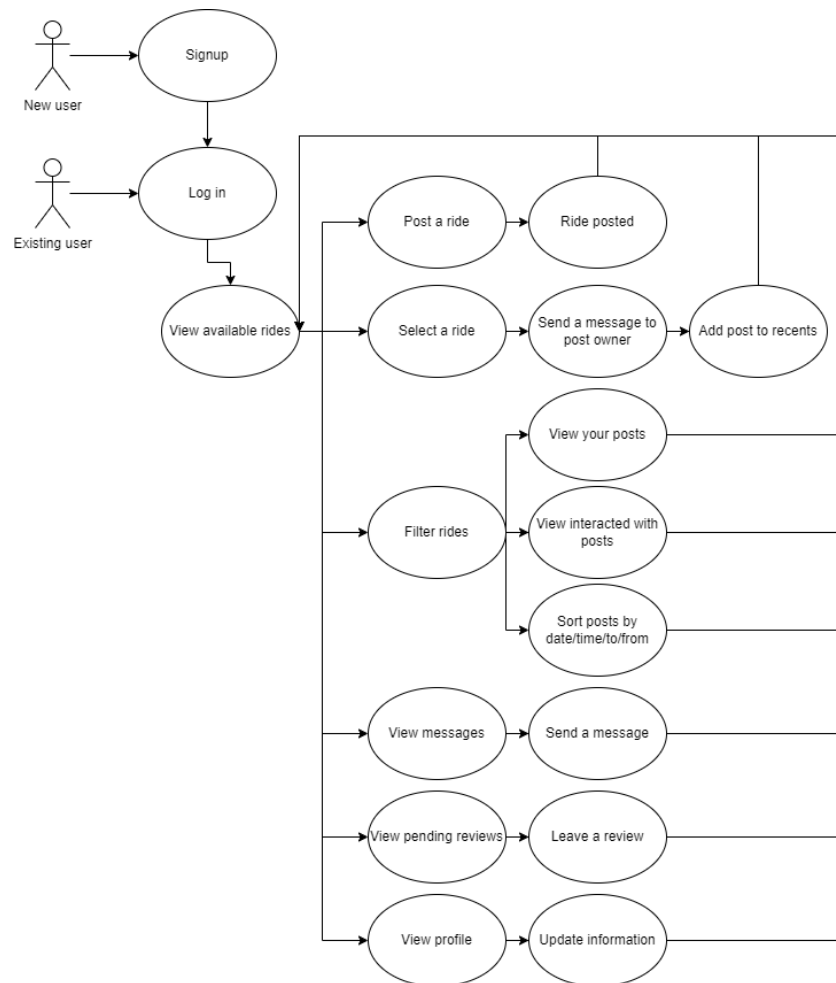
3.2.11 MANAGE RIDE POSTING: A post owner will be able to add a user who requested to be a passenger to the approved passenger list for the trip. Doing so will decrement the number of available seats. If the number of available seats becomes 0, the posting will close, which will send an automated message to the users interacted with who are not listed as passengers that the posting has been closed. Those on the approved passenger list will also be able to remove themselves from the list. Doing so will send an automated message notifying the post owner and increment the number of available seats. If a passenger on a closed posting removes themselves as a passenger, the posting will reopen. Furthermore, if

the post owner updates the details of the trip, a message will be sent to both approved passenger list and the users interacted who are not listed as passengers. Additionally, if the post owner is no longer available for the trip, they will be able to cancel, which will send an automated message to those in the approved passenger list that the trip has been cancelled. Finally, when the physical trip has been completed, post owners will be able to mark that the trip has been completed. This action will increment the number of trips as a driver for the post owner and the number of trips as a passenger for the users on the approved passenger list. The action will also close all conversations associated with the posting.

- 3.2.12 RATE AND REVIEW:** Users will be able to go to their list of users they have interacted with and leave a rating and review on the other users. The rating they leave will be applied in the user's average rating calculation. Furthermore, after leaving the rating, the rated user will be removed from the list of the user who rated them. Users will also have the option to remove other users from this list without leaving a review.
- 3.2.13 VIEW RATING RESULTS:** Users will be able to view their own page that is visible to others described in 3.2.7.
- 3.2.14 NOTIFICATIONS:** Users will be notified to leave a review upon completion of a trip. This is low priority.
- 3.2.15 UPDATE ACCOUNT INFORMATION:** Users will be able to update their account information.
- 3.2.16 DELETE ACCOUNT:** Users will be able to delete their own account, which will remove all of their information from the database.

3.3 Use Case Model

The figure below depicts the workflows of **RideMe**.



3.3.1 Use Case #1: New User Signup

Author – Liam Johnston

Purpose – To provide a new user with a **RideMe** account

Requirements Traceability – 3.2.1, 3.2.2

Priority - High

Preconditions – New user

Post conditions – The user will have a **RideMe** account and will be able to login to the application

Actors – Human

Extends – None

Flow of Events

1. User opens the application and is directed to the landing page
2. User selects the option on the landing page to sign up
3. User enters the necessary information to create an account as specified in 3.2.2 and hits submit
4. If successful, a new instance of the *User* model will be added to the database and if not, an error message will be displayed that specifies the reason for the error

Includes – None

Notes/Issues – If a user tries to sign up with an existing username or an existing or invalid email address, an error message will be shown that specifies what caused the error

3.3.2 Use Case #2: Existing User Login

Author – Liam Johnston

Purpose – To allow an existing user to login to their **RideMe** account

Requirements Traceability – 3.2.3, 3.2.4

Priority - High

Preconditions – The user already has a **RideMe** account

Post conditions – The user will be logged in to **RideMe** and be able to surf the application

Actors – Human

Extends – 3.3.1

Flow of Events

1. User opens the application and is directed to the landing page
2. User selects the option on the landing page to login
3. User enters the necessary credentials to login to their account as specified in 3.2.3 and hits submit

4. If successful, the user will be logged in to **RideMe**, be directed to the view ride postings page, and be able to surf the application and if not, an error message will be displayed that specifies the reason for the error

Includes – None

Notes/Issues – If a user enters invalid credentials, an error message will be shown that states that the username does not exist, or the username and password pairing do not match

Additionally, if a user forgets his password, they will be able to go through the process described in 3.2.4 so they are not permanently locked out of their account

3.3.3 Use Case #3: Logged in Prospective Post Owner

Author – Liam Johnston

Purpose – To allow a logged in user to become a post owner

Requirements Traceability – 3.2.6

Priority - High

Preconditions – The user already is logged in to their **RideMe** account

Post conditions – The user will be a post owner

Actors – Human

Extends – 3.3.2

Flow of Events

1. Logged in user is on the view ride postings page
2. User selects the option to add a posting
3. User enters the necessary information to add a post as specified in 3.2.6 and hits submit
4. If successful, a new instance of the *Posting* model will be added to the database and if not, an error message will be displayed that specifies the reason for the error

Includes – None

Notes/Issues – If a user forgets to include required information, or does not meet certain information criteria, such as if they try to enter a number into number of available seats that is not a positive integer or if they try to enter a date that occurred

prior to when they try to submit the posting, an error message will be shown that specifies what caused the error

3.3.4 Use Case #4: Logged in Post Owner

Author – Liam Johnston

Purpose – To allow a logged in post owner to manage his posts

Requirements Traceability – 3.2.9, 3.2.11

Priority - High

Preconditions – The user is logged in to their **RideMe** account and is a post owner

Post conditions – Modified states of *Posting* and *Conversation* instances

Actors – Human

Extends – 3.3.3

Flow of Events

1. Logged in post owner is on the view ride postings page
2. To manage posts, post owner can either select the option to view their postings or to view their messages
3. If they select to view their postings, they will have the options to update trip details, or cancel or complete the trip
4. If they select to view their messages, they will be able to view the ratings and reviews and message prospective passengers who have messaged them and approve them as passengers provided the posting still has available seats

Includes – None

Notes/Issues – Thorough testing will be conducted to ensure that all post owner actions function as they are intended

3.3.5 Use Case #5: Logged in Prospective Passenger

Author – Liam Johnston

Purpose – To allow a logged in user to become an approved passenger

Requirements Traceability – 3.2.5, 3.2.7, 3.2.8, 3.2.11

Priority - High

Preconditions – The user is logged in to their **RideMe** account

Post conditions – Modified *Posting* and *Conversation* instances

Actors – Human

Extends – 3.3.2

Flow of Events

1. Logged in user is on the view ride postings page
2. User can filter the listed postings to ensure that the ones they are viewing satisfy their needs
3. Users can select a posting, view the rating and reviews of the post owner, and send a message to the post owner which will create an instance of a *Conversation* and add it to their view messages page
4. Users can respond to messages sent by the post owner in hopes that they get listed as an approved passenger for the posting

Includes – None

Notes/Issues – Thorough testing will be conducted to ensure that all prospective passenger actions function as they are intended

3.3.6 Use Case #6: Logged in User Wanting to Leave a Review

Author – Liam Johnston

Purpose – To allow a logged in user to leave a review

Requirements Traceability – 3.2.5, 3.2.7, 3.2.8, 3.2.11

Priority - High

Preconditions – The user is logged in to their **RideMe** account

Post conditions – Modified *User* instance

Actors – Human

Extends – 3.3.2

Flow of Events

1. Logged in user is on the view ride postings page

2. User can go to their view pending reviews page, which will list all users they have interacted with that they have yet to review
3. They can remove users from this list by leaving a review, or they can manually remove them without leaving a review

Includes – None

Notes/Issues – Thorough testing will be conducted to ensure that all prospective passenger actions function as they are intended

Monitoring reviews for profanity or malicious comments may be implemented but this is low priority

3.3.7 Use Case #7: Logged in User Wanting to View and Update their Account Information

Author – Liam Johnston

Purpose – To allow a logged in user to view and update their account

Requirements Traceability – 3.2.13, 3.2.15, 3.2.16

Priority - High

Preconditions – The user is logged in to their **RideMe** account

Post conditions – Modified *User* instance

Actors – Human

Extends – 3.3.2

Flow of Events

1. Logged in user is on the view ride postings page
2. User can go to their account page and can view ratings and reviews that have been posted to their page
3. On their account page, they will have the option to edit certain account information, like phone number or email address, or to delete their account

Includes – None

Notes/Issues – Thorough testing will be conducted to ensure that all prospective passenger actions function as they are intended

4 Other Non-functional Requirements

4.1 Performance Requirements

P1: The system shall allow users to delete their accounts if desired.

P2: The system shall allow users to create and respond to postings.

P3: The system shall provide users with near real-time updates on available postings and messages.

P4: The system shall provide users with adequate feedback in user interface (descriptive and informative error messages, loading bars, animated buttons)

4.2 Safety and Security Requirements.

- The system shall adhere to all digital privacy laws applicable within Canada and Nova Scotia
- The databases of the system shall be properly secured by appropriate measures (password, encryption, etc.)
- Users can delete their information from system any system databases
- Users are required to sign-in to their account to access services (post, message, etc.)
- System is secure against well known intrusion methods.

4.3 Software Quality Attributes

4.3.1 Useability

The system shall be displayed as simple as possible to enhance clarity and user experience.

4.3.2 Performance

The system shall perform adequately under heavy loads and provide responses in a timely manner.

4.3.3 Correctness

The system shall update the available postings and message responses in a reasonable amount of time.

4.3.4 Availability

System should be active and available during any time of day and allow posting and messaging.

4.3.5 Portability

The system should be designed to work on a variety of web browsers and systems.

Appendix A – Data Dictionary

There will be three data models in this application: *User*, *Posting*, and *Conversation*. The following three tables provide details of the each of the model's attributes.

User Model

Attribute Name	Data Type	Description	Associated Reqs
userID	ID	Primary key to identify the instance of the model	3.2.2, 3.2.10, 3.2.11, 3.2.12
username	string	Used by its associated user, alongside the password to login to the application	3.2.2, 3.2.3
password	password	Used by its associated user, alongside the username, to login to the application	3.2.2, 3.2.3, 3.2.4
firstName	string	The associated user's first name	3.2.2
lastName	string	The associated user's last name	3.2.2
email	email address	The associated user's email address which will be used to reset password	3.2.2, 3.2.4
phone number	phone number	The associated user's phone number which may be used for multi-factor authentication	3.2.2, 3.2.3
numTripsAsDriver	integer	The associated user's number of trips as a driver which is incremented by 1 when the user completes a posting in which they own	3.2.7, 3.2.9, 3.2.11, 3.2.13
numTripsAsPassenger	integer	The associated user's number of trips as a passenger which is incremented by 1 when a posting in which they are on the approved passenger list is closed	3.2.7, 3.2.9, 3.2.11, 3.2.13
reviews	array of individual reviews	An array of the individual reviews posted to the user's	3.2.7, 3.2.9, 3.2.12, 3.2.13

		page which each have three attributes: the userID of the user leaving the review, the rating from 0 – 5 and a description of the experience with the user	
averageRating	float	The average of all the ratings of individual reviews posted to the user's page	3.2.7, 3.2.9, 3.2.12, 3.2.13
registrationTimestamp	date	The date in which the user created their account	3.2.2, 3.2.7, 3.2.9, 3.2.13
usersInteractedWith	array of individual users	Each element in the array has two attributes The userID and a hasReviewed state which is defaulted to False and once a user has left a review, it will switch to true and the element will be removed from the array	3.2.10, 3.2.11, 3.2.12

Posting Model

Attribute Name	Data Type	Description	Associated Reqs
postingID	ID	Primary key to identify the instance of the model	3.2.6
ownerID	ID	Foreign key which is a userID to identify the post owner	3.2.6
approvedPassengers	array of IDs	An array of foreign keys, which are userIDs, to identify approved passengers for the trip	3.2.11
usersInteracted	array of IDs	An array of foreign keys, which are userIDs, listing those who have messaged the post owner but have not been added to approvedPassengers	3.2.11
numAvailableSeats	integer	The number of available seats for the	3.2.6, 3.2.11

		trip which is set by the post owner upon post creation and is decremented when a user is added to approvedPassengers and decremented when a user is removed from approvedPassengers Note that when a new posting is created, the value of numAvailableSeats must be greater than 0 in order for the posting to be added	
isOpen	state	True if numAvailableSeats > 0 and False if numAvailableSeats == 0 Note that when a posting is created, isOpen must be set to True and its value can be swapped both from True to False and False to True	3.2.11
isCancelled	state	Can be True or False When posting is created, isCancelled is set to False If a post owner switches isCancelled to True, this operation is permanent and cannot be undone	3.2.11
isComplete	state	Can be True or False When posting is created, isComplete is set to False If a post owner switches isComplete to True, this operation is permanent and cannot be undone	3.2.11
tripDate	date	Date of the posting's associated trip	3.2.6, 3.2.11
tripTime	time	Time of the posting's associated trip	3.2.6, 3.2.11
pickupLocation	string	Description of the pickup location (may	3.2.6, 3.2.11

		be implemented as a true location but this is low priority)	
dropoffLocation	string	Description of the dropoff location (may be implemented as a true location but this is low priority)	3.2.6, 3.2.11
vehicle	string	Description of the vehicle in which the trip will be taken in	3.2.6, 3.2.11

Conversation Model

Attribute Name	Data Type	Description	Associated Reqs
conversationID	ID	Primary key used to identify the instance of the model	3.2.8
postingID	ID	Foreign key to identify the posting associated with the conversation	3.2.8
postOwnerID	ID	Foreign key to identify the post owner	3.2.8
passengerID	ID	Foreign key to identify the prospective or approved passenger	3.2.8
messages	Array of strings	An array of messages posted to the conversation which may either be posted by the users or the system	3.2.9, 3.2.11
isClosed	state	Closes when isComplete on the associated post is true	3.2.11

Note that because the data type of some attributes are arrays, this renders it necessary to use a NoSQL database. As a result, the database will be implemented using MongoDB.

Appendix B - Group Log

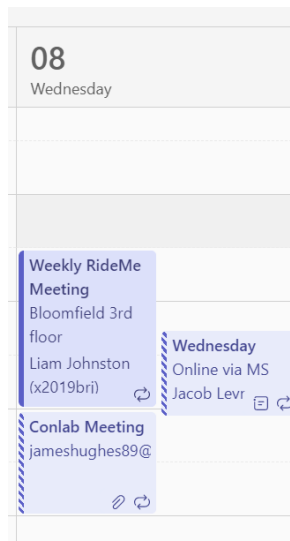
The following are the roles of each group member:

- **Liam Johnston:** Project Lead
 - Will be focused on back-end development during the development sprint given his experience with MongoDB
 - Will be in charge of engineering test cases during the testing sprint
- **Goutham Regu:** Back-end Lead
- **Devon Eaton:** Front-end Lead
- **Cole Beasant:** As-needed developer
 - Will be focused on front-end development during the development sprint to support pair-programming

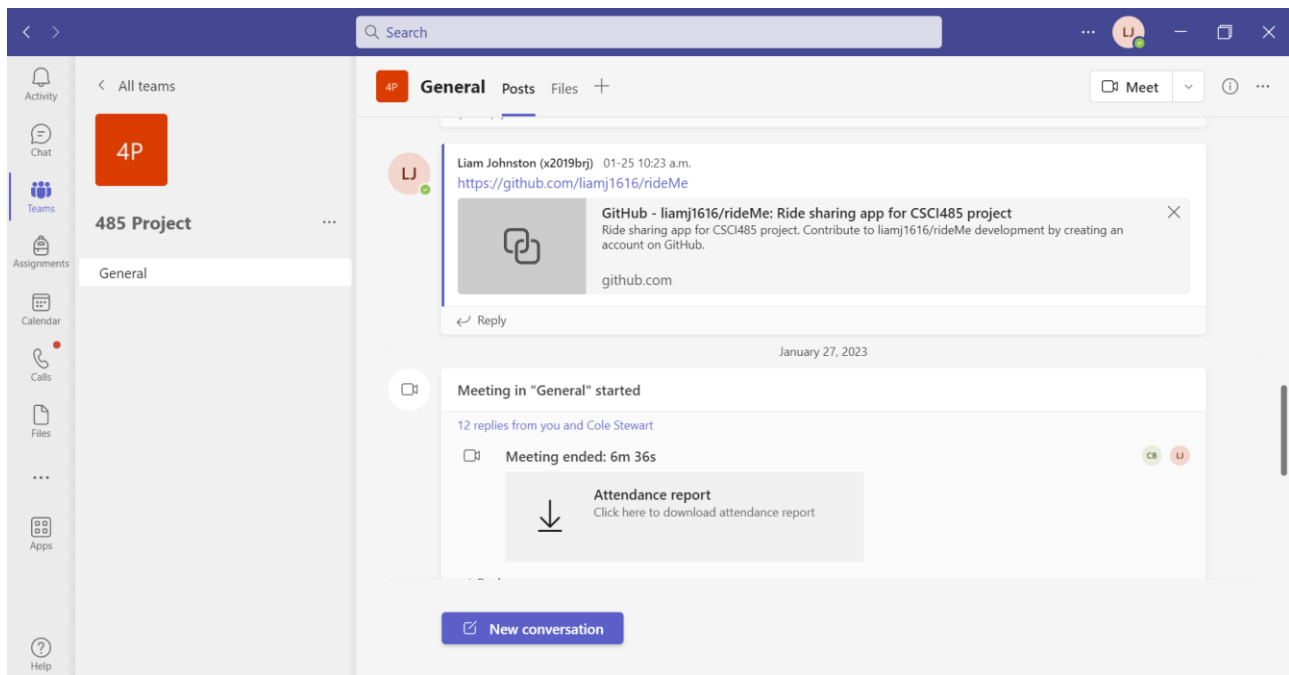
Note that all users will be involved in engineering and completing test cases during the testing sprint.

Group communication occurs during our CSCI485 class lecture hours, as well as on our **Instagram** group chat and on our **Microsoft Teams** page.

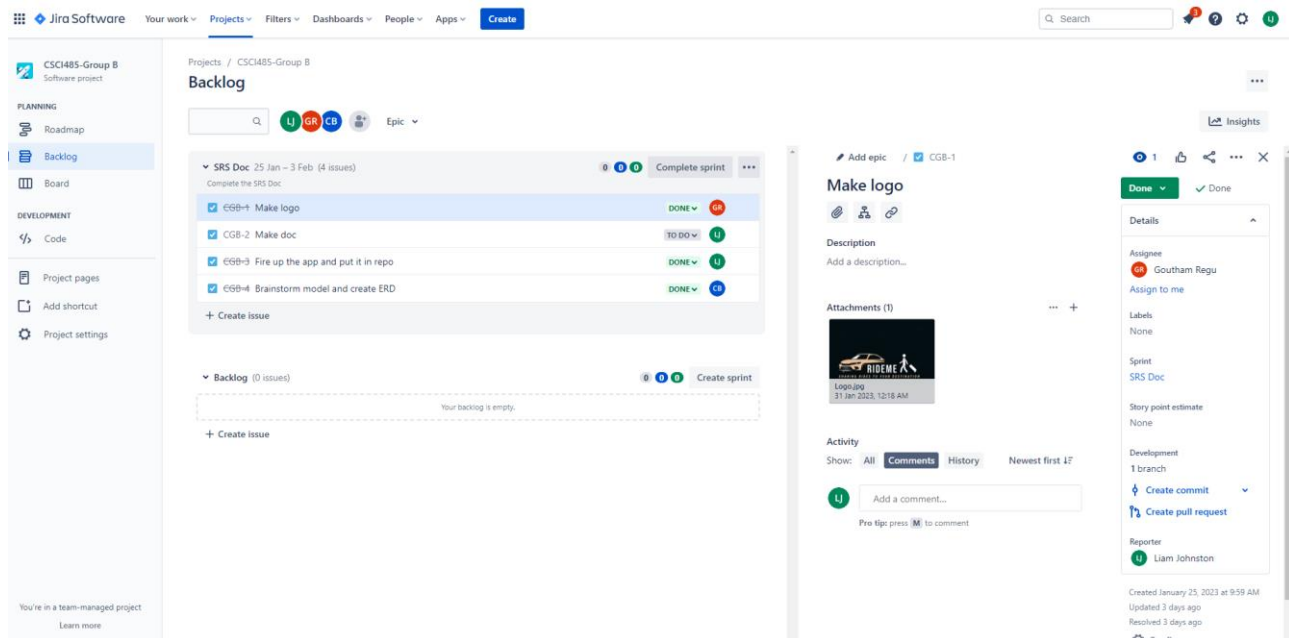
Weekly in-person meetings occur on Wednesdays from 9:30 – 11:00. They have been injected into every member's calendar as shown in the screenshot below:



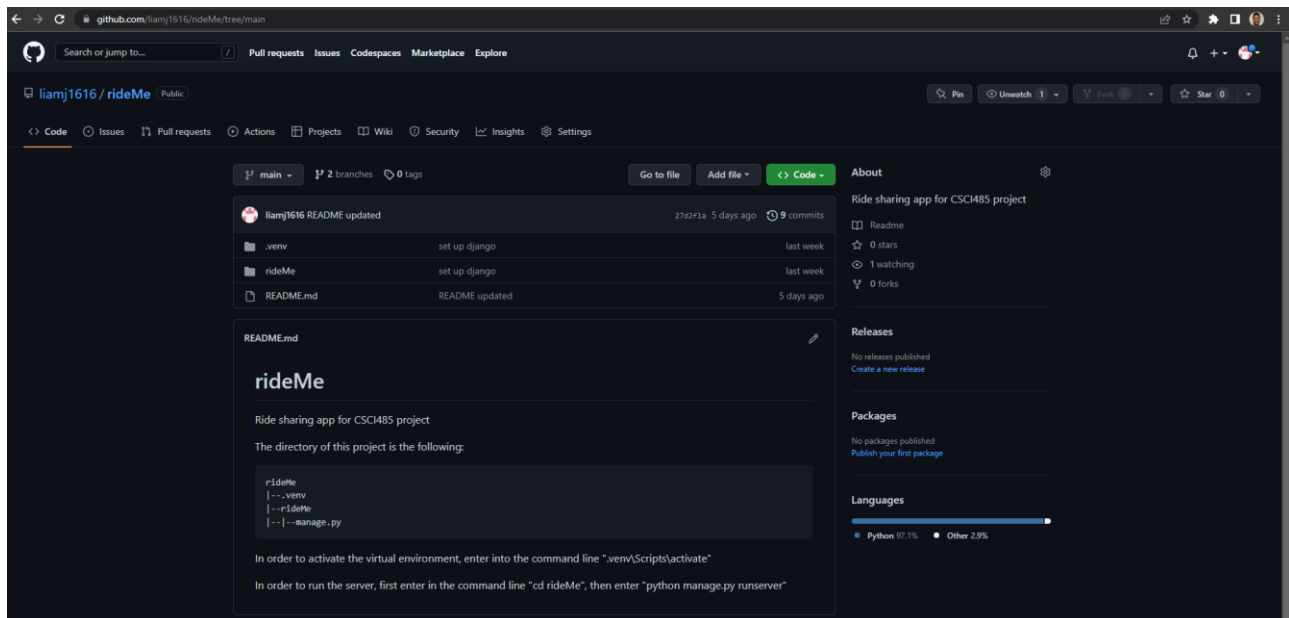
Unscheduled group meetings, either in person or on **Microsoft Teams**, may occur on an as-needed basis as shown in the screenshot below:



Jira is used to assign tasks to group members. Tasks are organized into sprints. The sprint in the screenshot below is the SRS document sprint, which will be completed upon submission of this document:



The codebase for **RideMe** will be on **Github** at the URL <https://github.com/liamj1616/rideMe/tree/main>. Currently, the **django** app has been started and committed to the **Github** page as shown in the screenshot below:



It is planned to discuss in the next group meeting how the **Github** code will be properly maintained and to ensure that no code is overwritten.

Architecture Design and Low-level Design



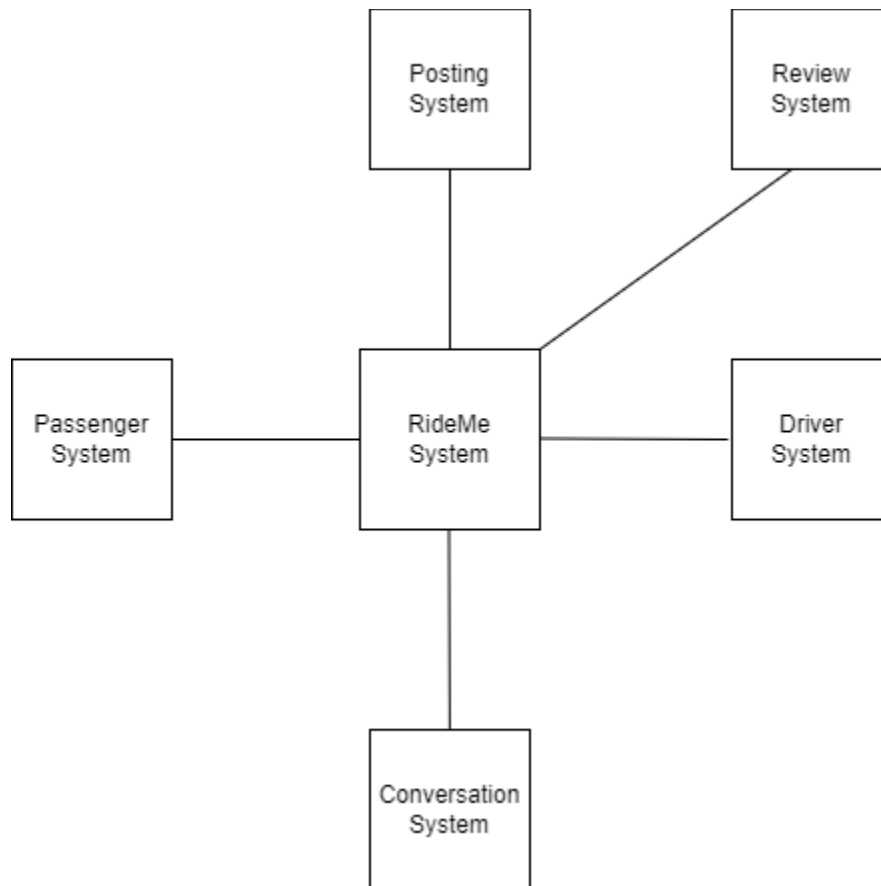
Course: CSCI 485 – Software Design

Instructor: Dr. Othman Soufan

Group B

Liam Johnston	201902759	x2019brj@stfx.ca
Cole Beasant	202001106	x2020asw@stfx.ca
Devon Eaton	201903479	x2019cjo@stfx.ca
Goutham Regu	202005622	x2020ftn@stfx.ca

Context Model



Passenger System: Once a RideMe user successfully signs up or logs in into the system. Users have access to the postings available, choose appropriately and go further with the conversation system. The prospective passenger waiting for approval to be an approved passenger by the driver.

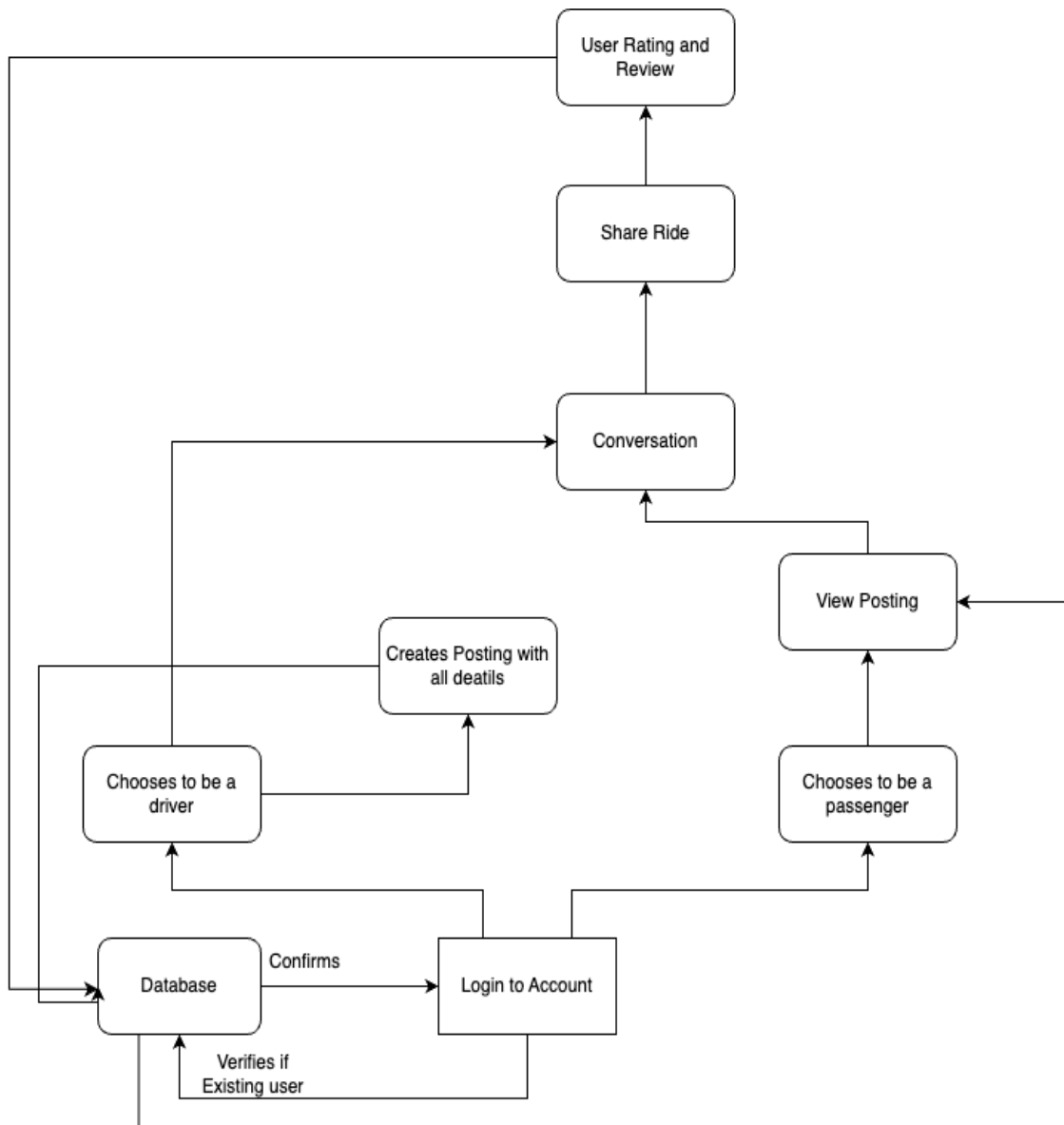
Driver System: User chooses to be a driver he/she has access to the posting system, will be able to post his/her ride on the postings page and will be able to manage postings, approve passengers on the ride and close the posting.

Posting System: This enables the users to select themselves as a driver and list a posting stating their trip date, tentative pickup time, tentative pickup and drop off locations, number of available seats, and the vehicle to be used for the trip. We achieved this by creating a new Posting instance with a unique postID. Upon successful creation the user will be redirected back to the postings page and the driver has access to manage posting as well.

Conversation System: Conversation system is achieved by creating an instance of a Conversation in the database and assign it a conversationID. Both users in the conversation will then be able to post messages to this conversation.

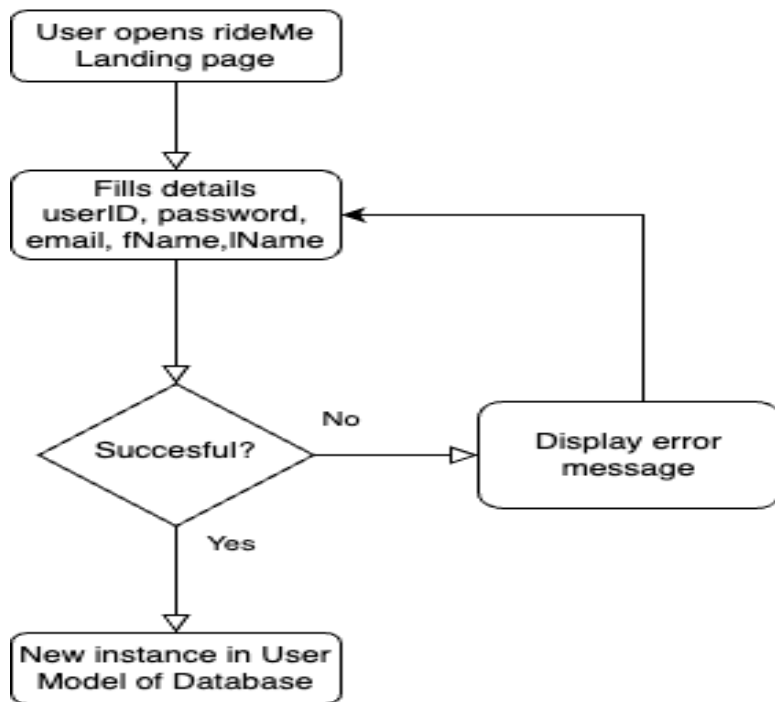
Review System: After the trip date has occurred, both the passenger and the driver are provided with an option of giving a review of each other which would help the future users on RideMe.

Process diagram

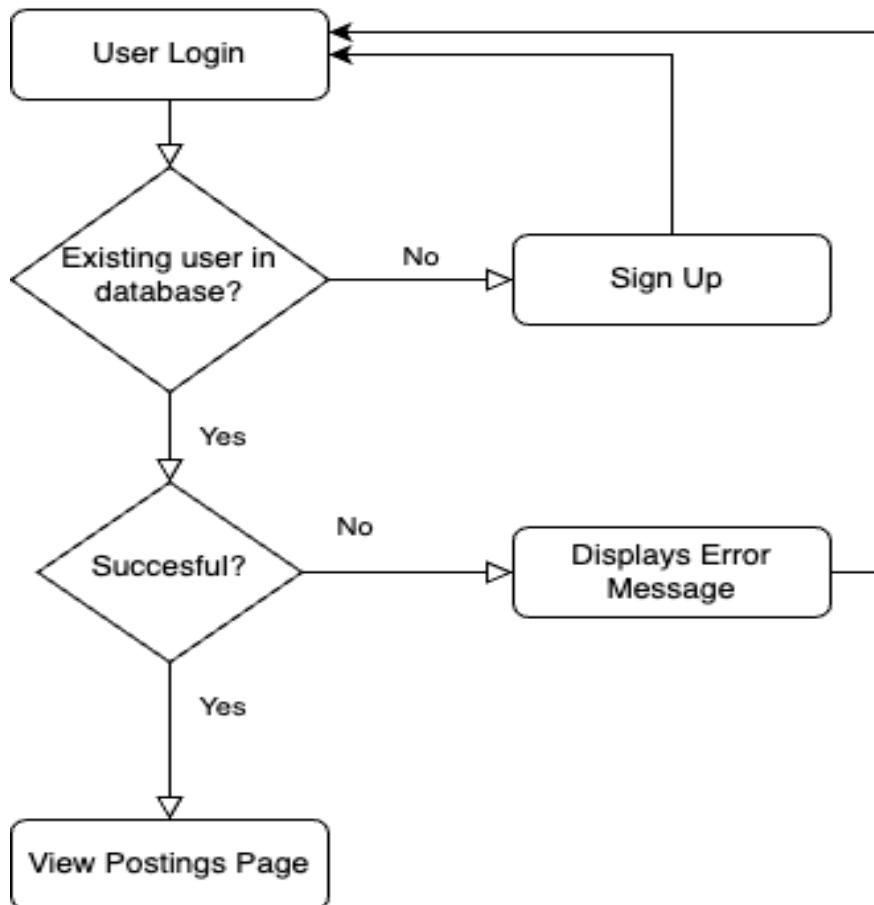


Use Case diagrams

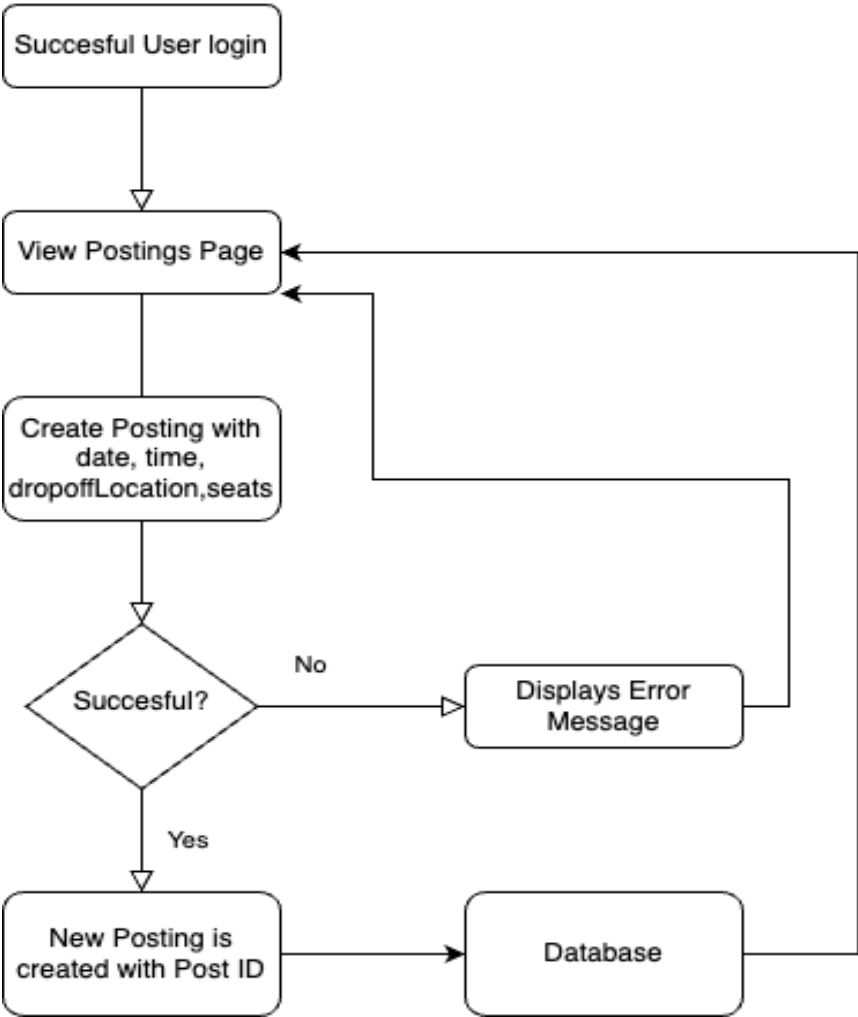
Use Case #1: New User Signup



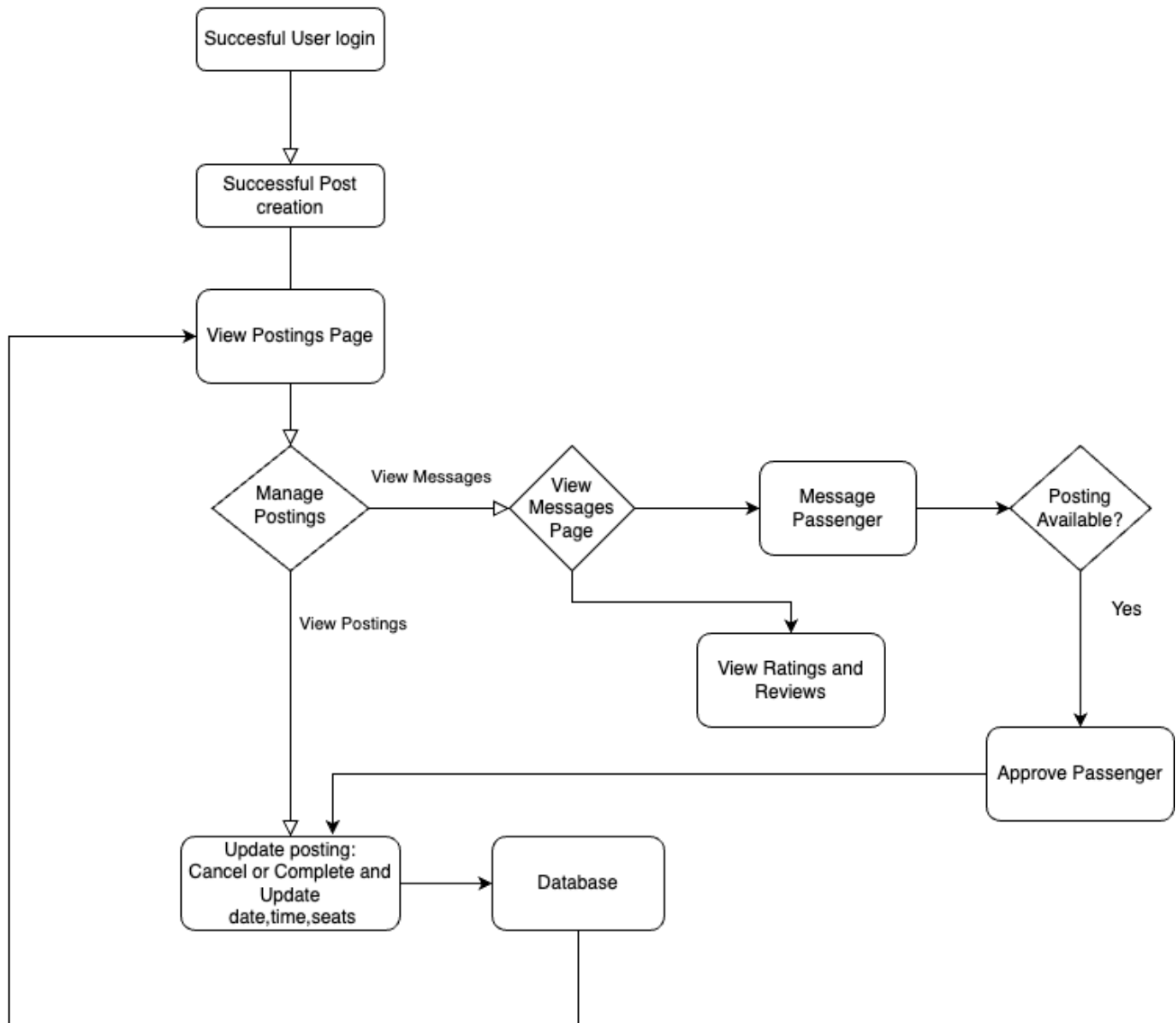
Use Case #2: User Login



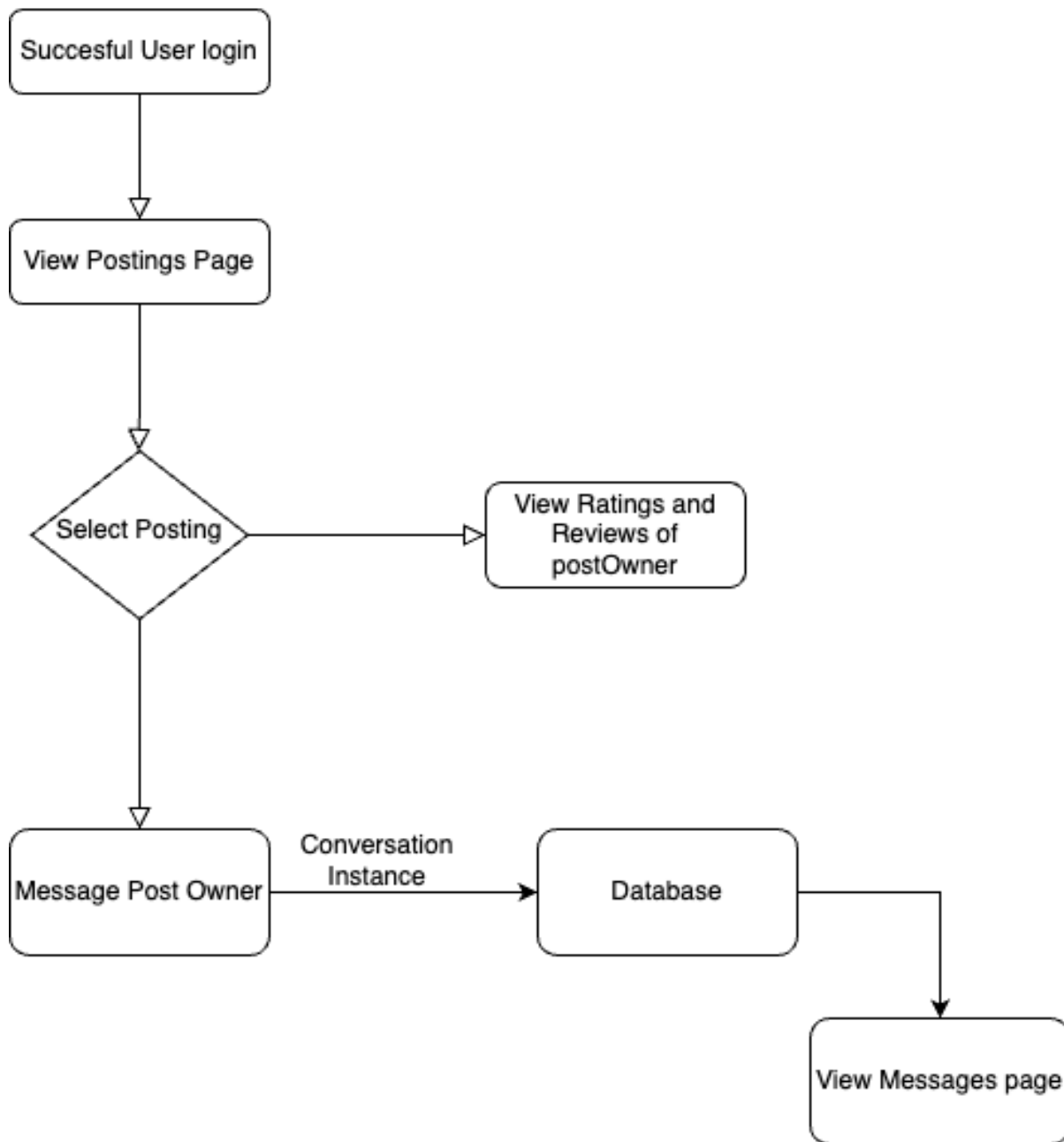
Use Case #3: Logged in Prospective Post Owner



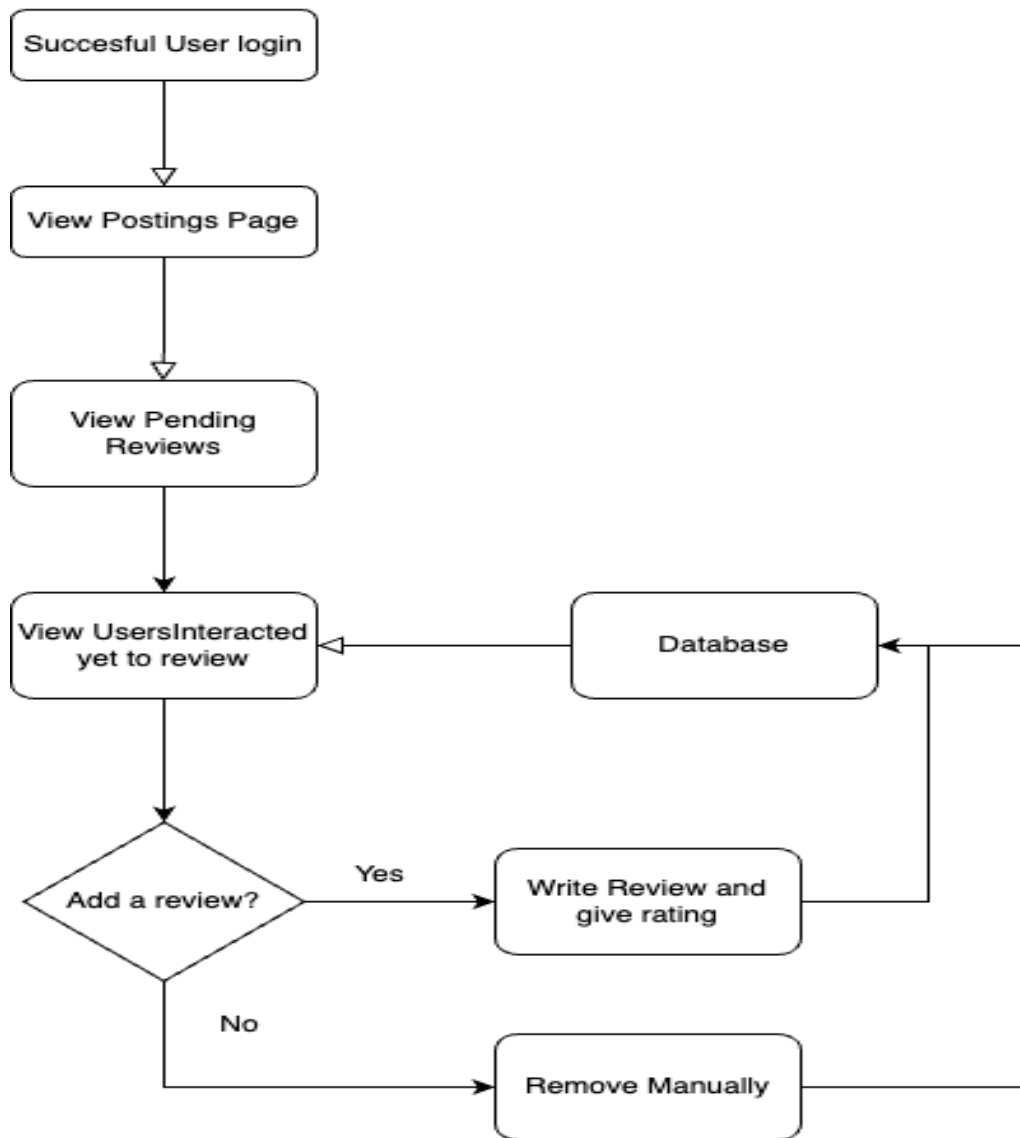
Use Case #4: Logged in Post Owner



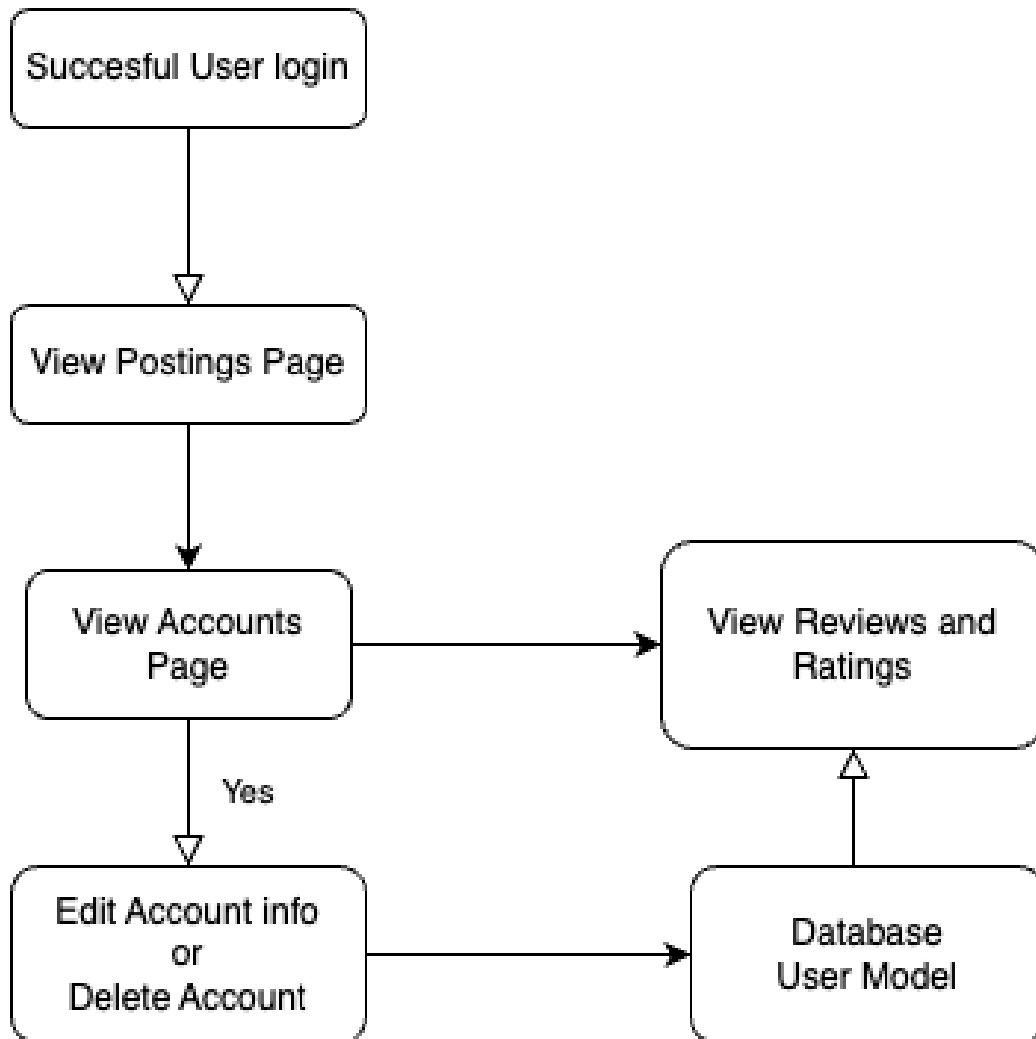
Use Case #5: Logged in Prospective Passenger



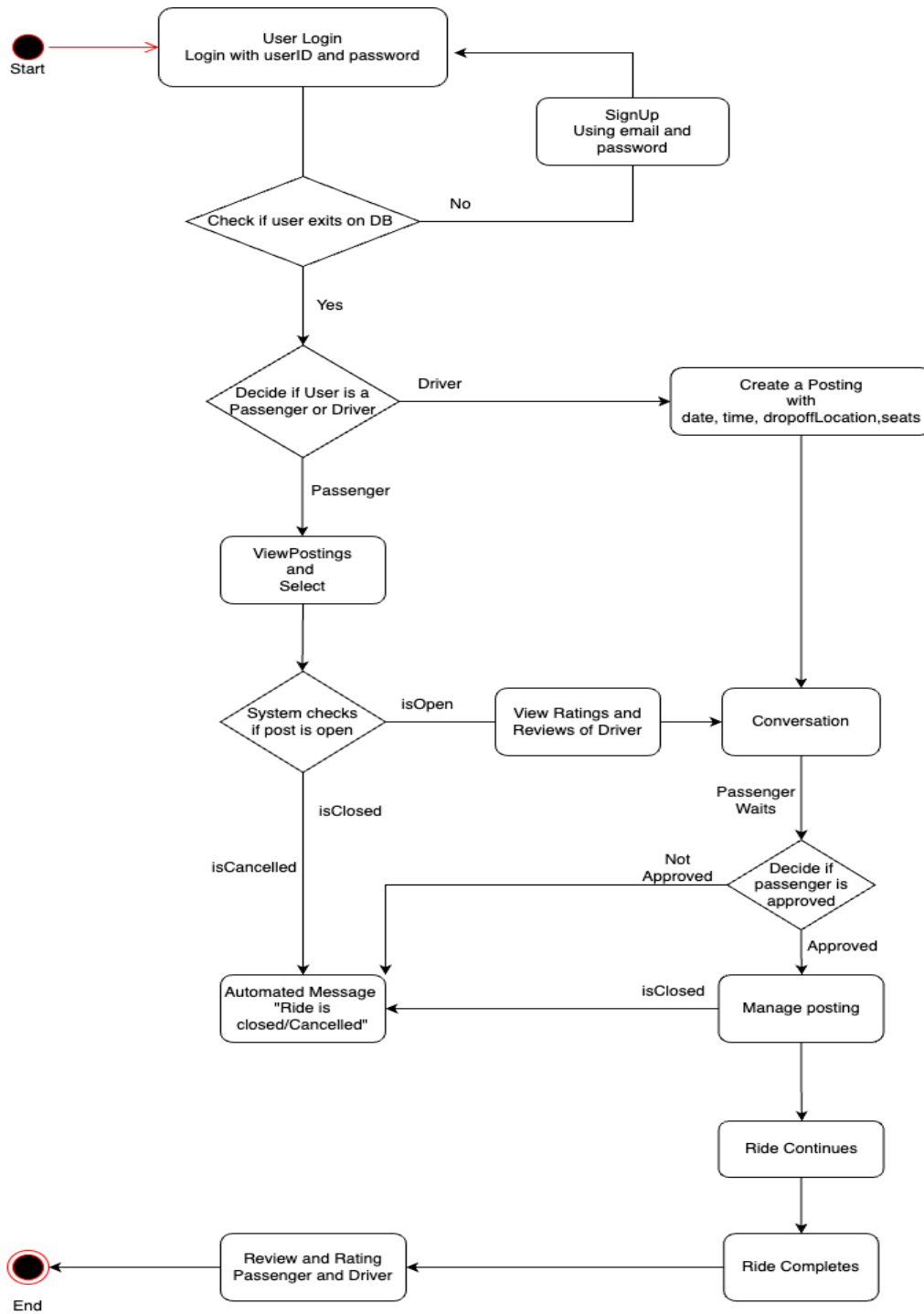
Use Case #6: Logged in User Wanting to Leave a Review



Use Case #7: Logged in User Wanting to View and Update their Account Information



State diagram

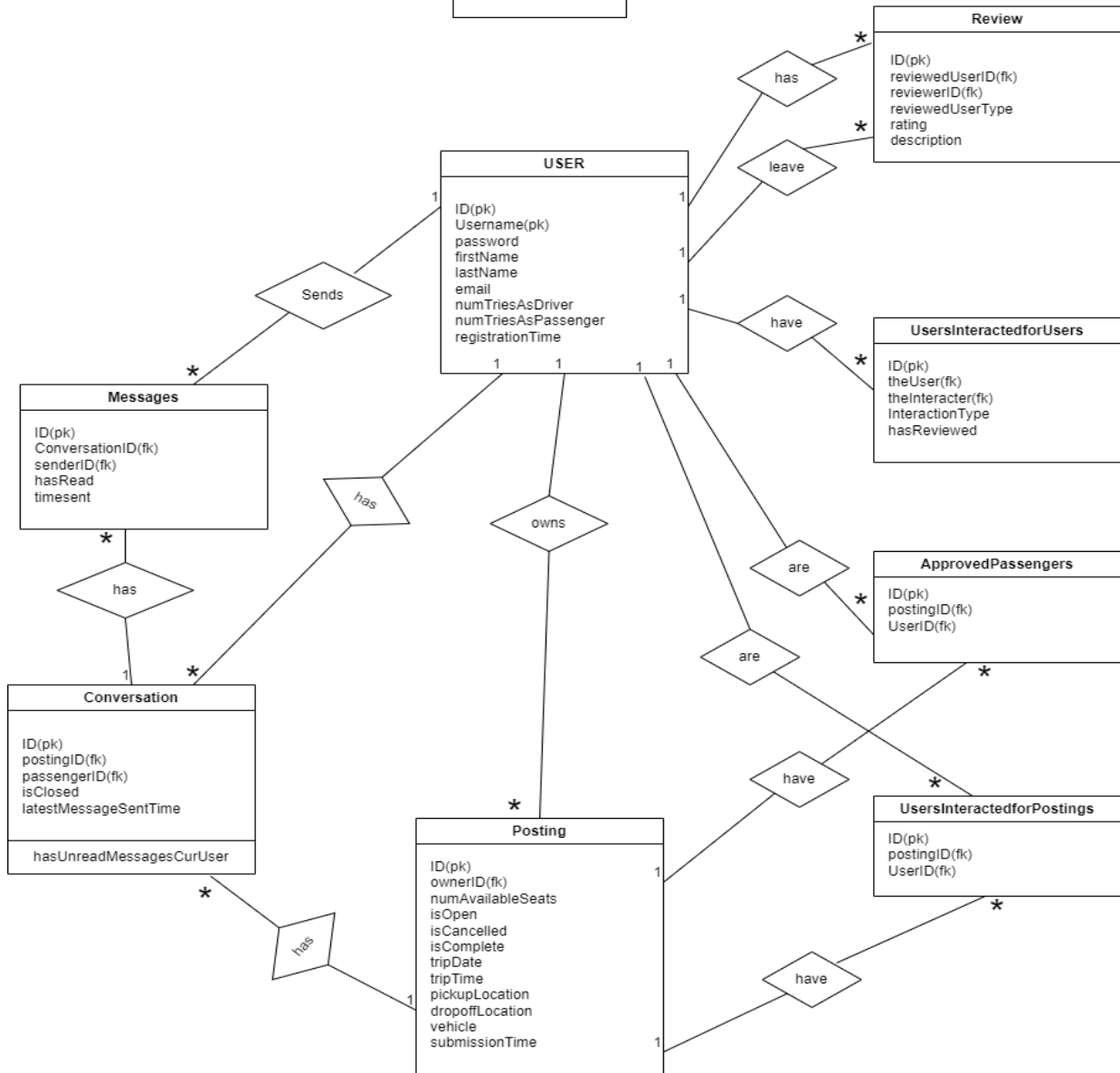


Class diagram



Entity relationship diagram

ERD



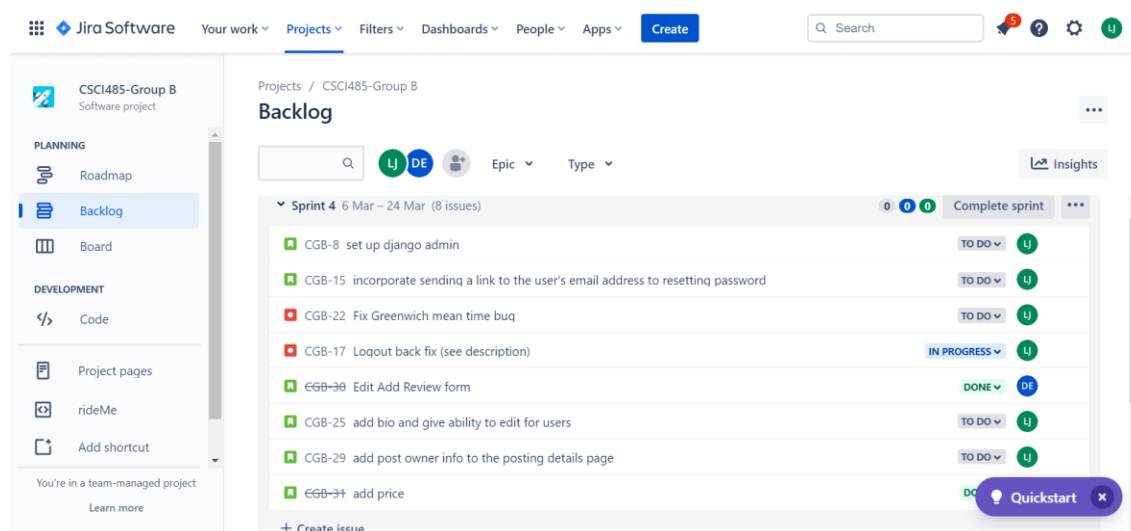
Appendix

Github

Code commits can be viewed at the link <https://github.com/liamj1616/rideMe/commits/main>. We have elected not to include a screenshot as the main repository is hosted on Liam's Github account and he tends to push his code whenever he makes small changes and has them tested, therefore there are a substantial amount of commits from his end. Other group members fork the code from Liam's repository and work on the code in their own forks, then submit a pull request to Liam for him to review and merge to his repository. Devon and Liam have been the only contributors to the coding aspect of this project at this point.

Jira

The following screenshot shows the project backlog for the current sprint:



If you would like to see the backlogs for other sprints, more screenshots can be provided.

RideMe Team Meetings Document



Course: CSCI 485 – Software Design

Instructor: Dr Othman Soufan

Group B

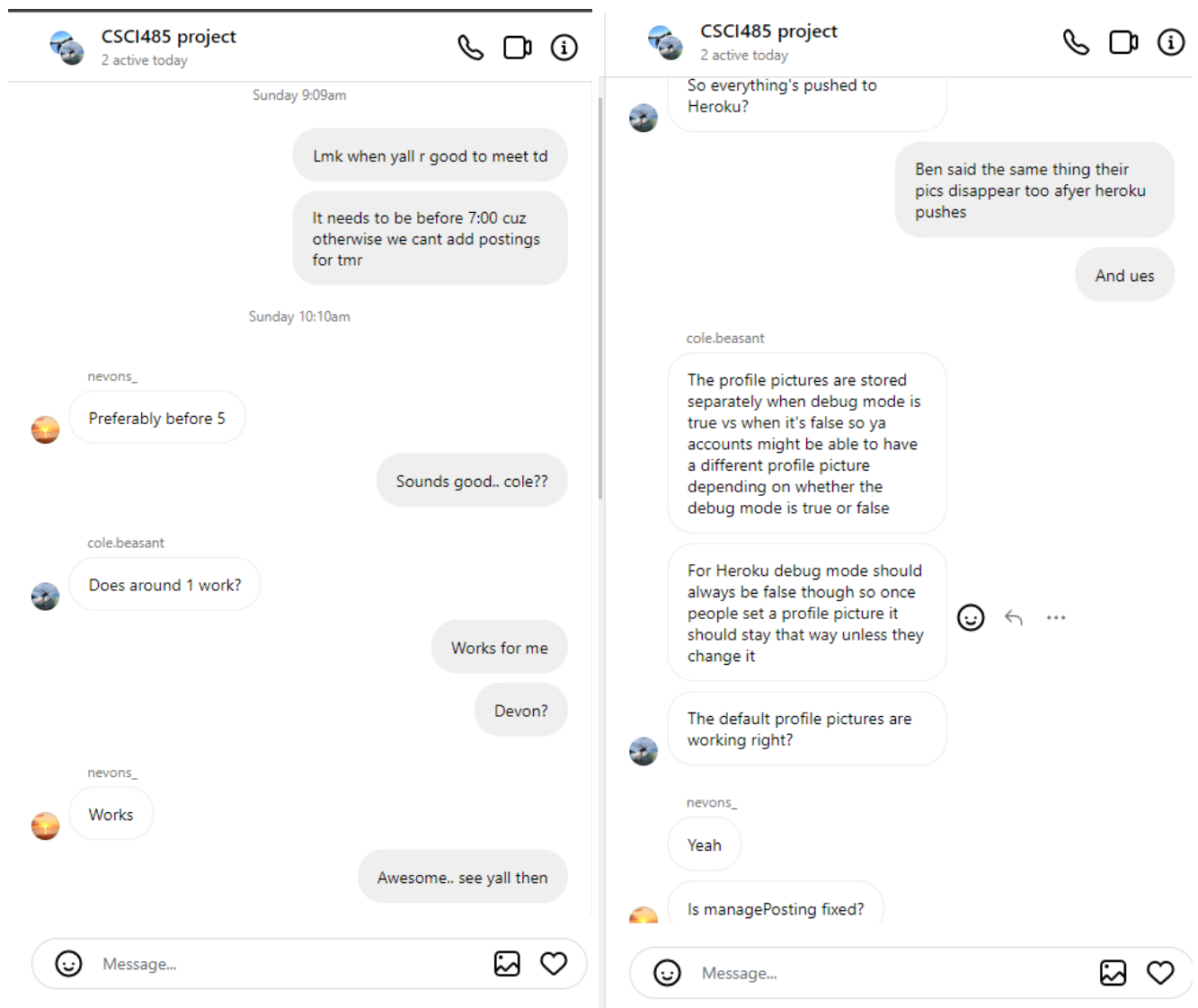
Cole Beasant	202001106	x2020asw@stfx.ca
Devon Eaton	201903479	x2019cjo@stfx.ca
Liam Johnston	201902759	x2019brj@stfx.ca
Goutham Regu	202005622	x2020ftn@stfx.ca

Table of Contents

Team Communication	3
Team Meetings	4
Code Commits	11
Jira Use	12

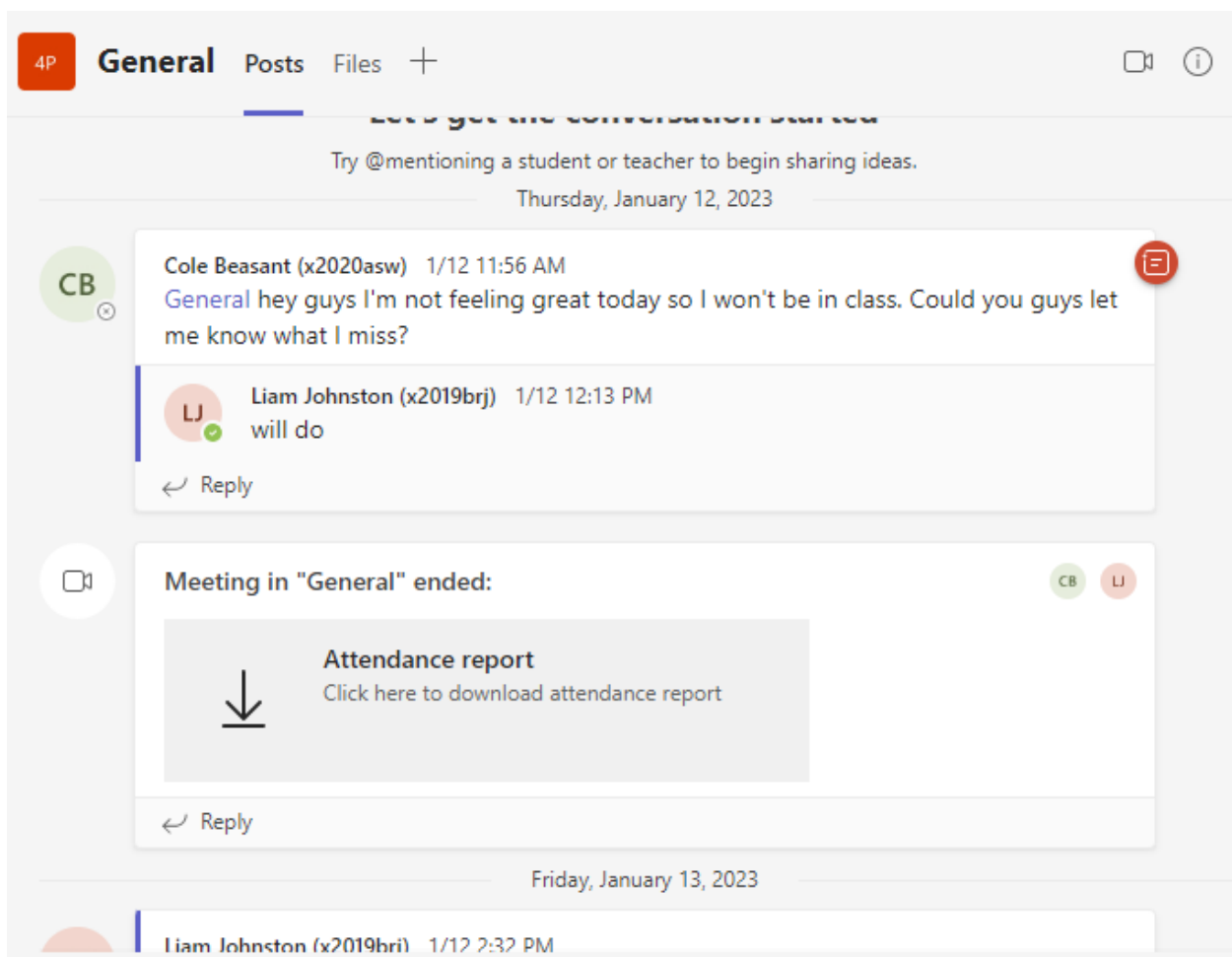
Team Communication:

Team communication occurred during the scheduled class meeting periods and in a group chat on Instagram. In both of these settings, we discussed any difficulties we were having and if we needed support from other team members. These were also used to call team meetings. Two screenshots of the group chat are shown below, which are from the perspective of Liam Johnston. The entire set of group chat messages are not provided given that communication occurred almost daily, but can be made available upon request.



Team Meetings:

Team meetings occurred on an as needed basis. If anyone needed to call a meeting, they would either notify the rest of the group members during the scheduled class meeting periods or send a message to the group in the aforementioned Instagram group chat. This message would include what was needed to be discussed during the meeting. The meeting would then occur on Microsoft Teams. The following are screenshots of the Microsoft Teams channel, which shows all of the team meetings that occurred during this semester. The icons in the top right corner of the “Meeting in “General” ended:” boxes indicate who attended the meeting with CB, DE, and LJ indicating Cole Beasant, Devon Eaton, Liam Johnston, respectively. The icon in the top right with the profile picture indicates Goutham Regu (was only present in the meeting on Sunday, March 5th).






4P

General


Posts







Files


+



3 replies from you and Goutham

 Reply






Liam Johnston (x2019brj) 1/23 12:08 PM


Discussion Topics:


1. Make sure everyone is on Jira
2. Come up with a project name
3. First page, everyone's student IDs
4. I will do the doc
5. Someone make a logo
6. Somebody fire up the code repo- goutham has to do this since he is admin
7. Somebody create the ERD
8. Better way to get ahold of everybody

[See less](#)



First Meeting
Wednesday, January 25, 2023 @ 9:30 AM



 Reply

Wednesday, January 25, 2023



4P

General


Posts

Files

+





Friday, January 27, 2023



Meeting in "General" started


13 replies from you and Cole Stewart

 Reply




New channel meeting started


3 replies from Devon Robert and Cole Stewart

 Reply

Wednesday, February 1, 2023






Meeting in "General" ended:



Attendance report

Click here to download attendance report



 Reply

Monday, February 6, 2023

4P

General

Posts

Files

+

Monday, February 6, 2023

Meeting in General ended: 43m 53s

↓

Attendance report

Click here to download attendance report

DE

LJ

CB

↩

 Reply

Monday, February 13, 2023

Meeting in "General" started

2 replies from you

↩

 Reply

Sunday, February 26, 2023

Meeting in "General" started

2 replies from you and Devon Robert



4P

General


Posts

Files


+




Monday, February 27, 2023





Meeting in "General" started


 Reply



Meeting ended: 7m 1s




 Reply




Meeting in "General" started


4 replies from Goutham


 Reply

Sunday, March 5, 2023




Meeting in "General" ended: 56m 31s





Attendance report

Click here to download attendance report

 New conversation

4P

General

Posts

Files

+

i

...

← Reply

Sunday, March 26, 2023

Meeting in "General" started

3 replies from you

← Reply

Sunday, April 2, 2023

Meeting in "General" started

9 replies from you, Cole Stewart, and Devon Robert

Meeting ended: 4h 43m

DE

CB

LJ

↓

Attendance report

Click here to download attendance report

← Reply

Yesterday

New conversation

4P

General

Posts

Files

+

i

...

Yesterday

Meeting ended: 1h 13m

LJ DE CB

↩

Reply

New channel meeting ended: 10m 33s

DE LJ

↓

Attendance report

Click here to download attendance report

↩

Reply

New channel meeting ended: 18m 23s

LJ DE

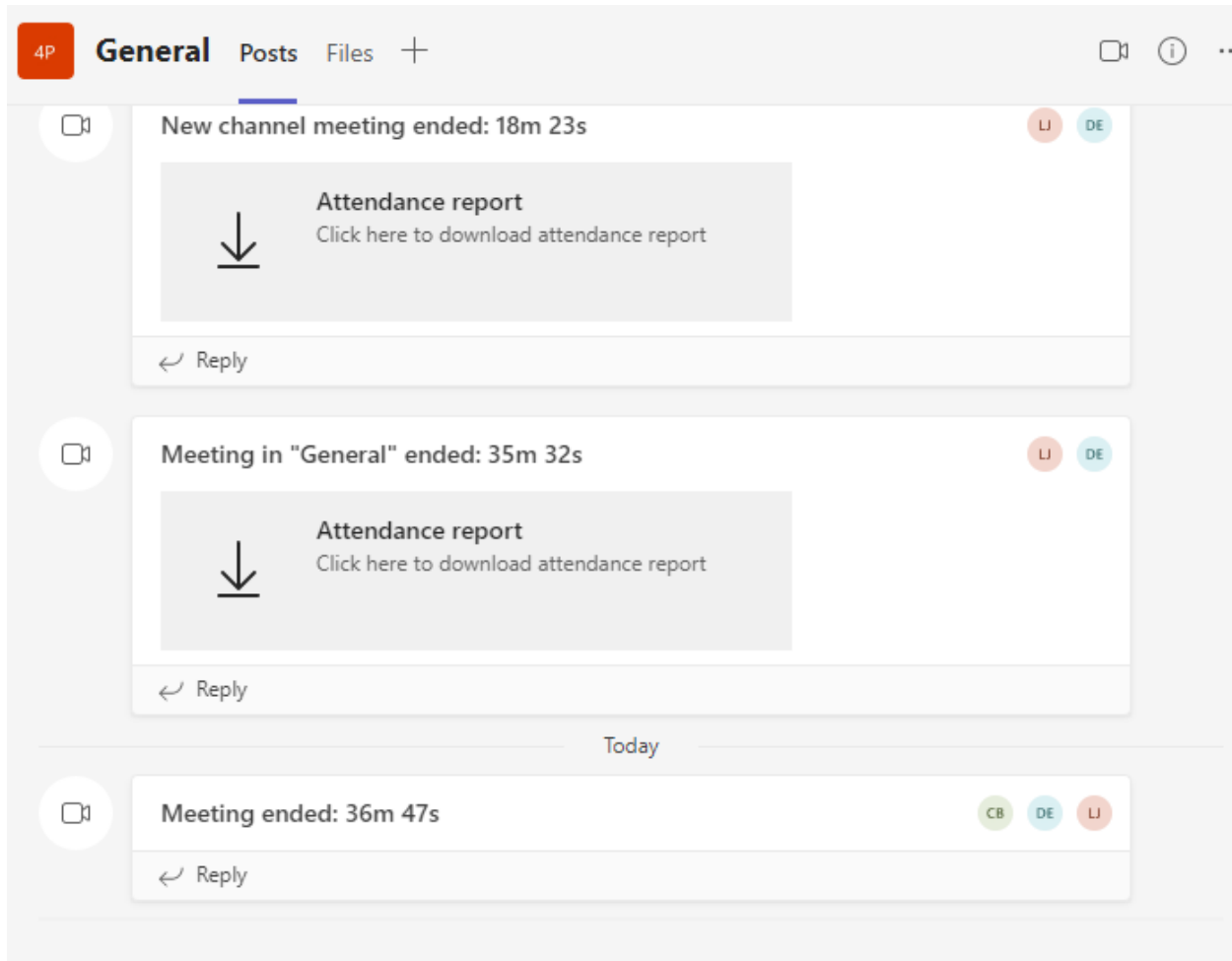
↓

Attendance report

Click here to download attendance report

↩

Reply




Code Commits:

Code commits can be viewed at <https://github.com/Cole-Beasant/rideMe/commits/main>.

Jira Use:

The following are screenshots of the reports for each of the 3 sprints noted in Jira:

Sprint One:

 All issues in sprint have been completed						
Completed issues View in issue navigator						
Key	Summary	Issue type	Epic	Status	Assignee	Story points
CGB-1	Make logo	Task		DONE	GR	-
CGB-2	Make doc	Task		DONE	LJ	-
CGB-3	Fire up the app and put it in repo	Task		DONE	LJ	-
CGB-4	Brainstorm model and create ERD	Task		DONE	CB	-

Sprint Two:

Incomplete issues View in issue navigator						
Key	Summary	Issue type	Epic	Status	Assignee	Story points
CGB-8	set up django admin	Story		TO DO	LJ	-
CGB-15	incorporate sending a link to the user's email address to resetting password	Story		TO DO	LJ	-
CGB-17	Logout back fix (see description)	Bug		IN PROGRESS	LJ	-
CGB-20	Dont allow past trips to be added at the form level	Story		DONE	DE	-
CGB-21	try and make a time widget in the add posting form like the date widget	Story		DONE	DE	-

Completed issues View in issue navigator						
Key	Summary	Issue type	Epic	Status	Assignee	Story points
CGB-5	add logout functionality	Story		DONE	LJ	-
CGB-6	figure out how to access the signed in users information across the site	Story		DONE		-
CGB-7	making exceptions to ensure things arent duplicated and values are proper	Bug		DONE	LJ	-
CGB-10	Fix trip date field on addPosting form	Bug		DONE	LJ	-
CGB-11	add message post owner functionality to postingDetails page	Story		DONE	LJ	-
CGB-12	add 'add review' pages	Story		DONE	LJ	-
CGB-13	add 'view my conversations' page	Story		DONE	LJ	-
CGB-14	add posting management for post owners	Story		DONE	LJ	-
CGB-18	primary key aint generating for ApprovedPassengers	Bug		DONE		-
CGB-19	Discuss my postings query (ie do we want only future postings or keep historical ones ...)	Story		DONE	LJ	-

Sprint Three:

Incomplete issues							View in issue navigator
Key :	Summary :	Issue type :	Epic :	Status :	Assignee :	Story points	
CGB-15	incorporate sending a link to the user's email address to resetting password	Story		TO DO	LJ	-	
CGB-17	Logout back fix (see description)	Bug		IN PROGRESS	LJ	-	
CGB-8	set up django admin	Story		TO DO	LJ	-	
CGB-22	Fix Greenwich mean time bug	Bug		TO DO	LJ	-	
CGB-25	add bio and give ability to edit for users	Story		TO DO	LJ	-	
CGB-29	add post owner info to the posting details page	Story		TO DO	LJ	-	
CGB-30	Edit Add Review form	Story		DONE	DE	-	
CGB-31	add price	Story		DONE	LJ	-	

Completed issues							View in issue navigator
Key :	Summary :	Issue type :	Epic :	Status :	Assignee :	Story points	
CGB-21	try and make a time widget in the add posting form like the date widget	Story		DONE	DE	-	
CGB-20	Dont allow past trips to be added at the form level	Story		DONE	DE	-	
CGB-23	Figure out which non navbar top buttons to delete	Story		DONE	LJ	-	
CGB-24	try and get the send message forms on the same pages as posting details and convers...	Story		DONE	DE	-	
CGB-26	add approved passenger list to the manage posting page	Story		DONE	LJ	-	
CGB-27	remove posts you own from the postings list	Story		DONE	LJ	-	
CGB-28	add associated conversations query to the manage posting page	Story		DONE	LJ	-	
CGB-32	remove user postings from my profile	Story		DONE	LJ	-	

It must be noted that all of the tasks in Jira were assigned to Liam Johnston and Devon Eaton, given that they were the two team members who built the RideMe's functionality. Cole Beasant's tasks of adding profile pictures and re-styling the website and Goutham Regu's task of completing the Low-Level Design Document are not present in Jira.