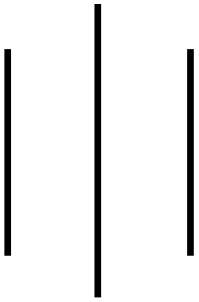
LA GRANDEE INTERNATIONAL COLLEGE

Simalchaur, Pokhara Nepal

A Project Report

On

"Ride sharing system"



Submitted To:

Bachelor of Computer Application (BCA) Program

In partial fulfilment of the requirements for the degree of BCA under

Pokhara University

Submitted by:

Name:	Course	Semester	P.U. Registration Number
Nabaraj Bhandari	BCA	8th	2017-1-53-0120
Sajida khatun	BCA	8th	2017-1-53-0230
Lu Bahadur Karki	BCA	8th	2017-1-53-0218
Sagar Bohora	BCA	8th	2017-1-53-0229

Date:14/09/2022

Acknowledgement

The satisfaction that accompanies after the successful completion of any task will be incomplete without mentioning the people whose ceaseless and relentless cooperation, constant guidance and encouragement made this project possible.

We are grateful to our project supervisor **Mr Ritesh Thapa**, and **BCA** coordinator **Mr Ramesh Chalise** for the guidance, inspiration and constructive suggestions that helped us in the preparation of this project.

We are also appreciative of each other and have understood that teamwork, the designation of the task per the skillset one portrays constant synchronisation and monitoring of progress and installing new knowledge and skill is imperative for the success of any given work.

Sincerely,

Nabaraj Bhandari

Sajida Khatun

Lu Bahadur Karki

Sagar Bohora

Declaration for

"Taxi Ride"

Student's Declaration

We, hereby declare that the presented report of **PROJECT IV** entitled as **"Taxi Ride App"** is submitted to LA Grandee International College, Faculty of Science and Technology, under the affiliation of Pokhara University for the partial fulfilment of Bachelor of Computer Application, under the supervision of **Mr. Ritesh Thapa.**

We also confirm that the report is only prepared for our academic requirement, not for any other purpose. No other sources other than the listed here have been used in this work.

Furthermore, it has not been and will not be used elsewhere for any other purposes.

_____

Name of the Students

Nabaraj Bhandari, 18530119

Sajida Khatun, 18530128

Lu Bahadur karki, 18530117

Sagar Bohora, 18530127

BCA, 8th Semester

Date: 14/09/2022

Supervisor's Declaration

I hereby recommend that **Nabaraj Bhandari, Sajida Khatun, Lu Bahadur Karki and Sagar Bohora** did this project entitled "Ride sharing system" under my supervision during their 8th Semester in partial fulfilment of the requirements for the degree of **BCA** under **Pokhara University** is completed to my satisfaction and be processed for final evaluation.

_____

Mr. Ritesh Thapa

Date: 14/09/2022

Letter of Approval

This is to certify that Nabaraj Bhandari, Sajida khatun, Lu Bahadur Karki, & Sagar Bohora, student of BCA 8th Semester at LA Grandee International College with PU Registration number as(2017-1-53-0220,2017-1-53-0230,2017-1-53-0218,2017-53-1-0227),respectively, has successfully completed their project entitled as "Taxi Riding App ".

We have examined the project report and hereby approve their work as per the requirement for the partial fulfilment of the degree of Bachelor of Computer Application under the affiliation of Pokhara University.

……..	……..	……...
Mr. Ritesh Thapa	Mr. Ramesh Chalise	Er. Kiran K.C.
Supervisor	Academic Coordinator	Principal

…….……

Dr. Uday Raj Dhungana

External Examiner

Date: 14/09/2022

Abstract

This report is presented to the Department of Computer Application, LA Grandee International College as a part of the Project of BCA 8th Semester. Moreover, this report can be considered as the overview of the project **Taxi Riding App.**

Taxi Riding App is an app through which you can easily book a taxi without any hassle. In this era, we're living in, everything is done with a click then why not booking a taxi with a click. It will help users to easily book a taxi, estimate the taxi fare and ultimately save time and provide security to the users. It is a secured way of booking a taxi as the mandatory details of the taxi driver and passenger will be stored safely as a reference.

Taxis have always been a convenient way of transportation. People use taxis for its comfort and convenience. Even so, the daily commuters find it easier to take a bus or own a two-wheeler (bike/scooter) for their daily commute. Undoubtedly, taxis have their own comfort and use for many purposes. Sometimes spending a few more in taxi fare turns out to be the best decision as it helps us to reach places at the right time.

This app is developed for Flutter, Rest API from Rest framework, Docker and Azure cloud computing. To embed the necessary features we have also used open source Openstreetmap.

Internet services have made us cling to apps and online services in this decade. Shopping to studying, chatting to meetings, social media, and everything is possible over the internet. In fact, the internet has made our work and daily lives easier. That is the reason,we choose to develop an online taxi-riding app to make taxi riding easier and hassle free.

Table of Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
DFD	Data Flow Diagram
IT	Information Technology
JWT	JSON Web Token
PU	Pokhara University
RDBMS	Relational Database Management System
SMS	Short Message Service
SS	Screenshot
UI	User Interface
VM	Virtual machine
ER Diagram	Entity Relationship Diagram

1. Introduction

It is so amazing to experience and to be the witness of modern innovation and emerging technologies making our life more convenient and easier to sustain day by day. The evolution of the internet and web connectivity technology has changed everything in the history of human civilization which has the power to convert a huge global into a mini village. There is a tremendous growth of users of mobile devices and computers through which we can connect from any corner of the world. There were 2.71 billion smartphone users worldwide in 2019 and had reached 3.5 billion in 2020, marking a 9.3% increase rate. Due to the large numbers of users of mobile devices, the use of mobile applications is also high (Department).

Application software, designed to run on a mobile device, are mobile applications created with a limited function and are generally known as mobile applications for specific purposes. **'Taxi Riding'** is a mobile application that is being built to help **"taxi ride"** in Nepal through a mobile application platform. This 'Taxi Sharing' application is specially targeted for taxi drivers and passengers or any kind of travellers inside Nepal. The reason for doing work on this idea and especially for Nepal has huge Ride potential.

Taxi Sharing Ride App - Taxi Riding is an app through which you can easily ride a taxi without any hassle. In this era, we're living in, everything is done with a click then why not a taxi with a click ride. It will help users to easily ride a taxi, estimate the taxi fare and ultimately will save time and provide security to the users. It is a secured way of booking a taxi as the mandatory details of the taxi driver and passenger will be stored safely as a reference.

Taxis have always been a convenient way of transportation. People use taxis for its comfort and convenience. Even so, the daily commuters find it easier to take a bus or own a two-wheeler (bike/scooter) for their daily commute. Undoubtedly, taxis have their own comfort and use for many purposes. Sometimes spending a few more in taxi-fare turns out to be the best decision as it helps us to reach places at the right time.

Internet services have made us cling to apps and online services in this decade. Shopping to studying, chatting to meetings, social media, and everything is possible over the internet. In fact, the internet has made our work and daily lives easier. That's

the reason, I chose to develop an online Taxi Riding App to make taxi rides easier and hassle free.

1.1 Background

There are many reasons why mobile applications are so popular. The growing popularity of mobile applications and technologies leads almost all sectors to develop relations with people through mobile applications. The most important factor that determines the future of any People is time, so many people has focused on switching the time saving to E- Ride from the normal way for transport to both time and cost saving. The mobile application can play a vital role in making the tourism sector more digital by providing various services and platforms to the passenger and driver.

The Ride Sharing mobile application will be one of the suitable application techniques that can be used to pursue e-ride in Nepal. This application focuses on providing the best services to all passenger and driver to the country and increasing the possibility of both cost and time saving during the travel in Nepal.

1.2 Aim of the project

The primary purpose of this project is to design and develop a fully functional 'Taxi Ride'

mobile application to aid the driver and passenger who choose Taxi riding trails as their travelling destination.

1.3 Limitations

Some of the limitations of this research project are as follows:

1. Hard to identify all trials and detailed figures of taxis from Pokhara and Nepal.

2. Providing the security of travellers in real-time.

3. Network issues in the upper trails of Nepal.

4. Multiple Taxi Ride App on the marketplace.

2. Problem Statement

Online taxi riding is one area where we are still under the growing phase. Some companies have also developed a few taxi booking apps. However, they are still under testing mode with respect to their usability and reliability. The available taxi booking apps are still on the road to creating an impact on people.

1. People still use traditional ways to book a taxi like standing on the roadside, calling a known taxi, or going to the taxi stand.

2. Difficulty in estimating the duration and taxi fare.

3. No information about taxi drivers and passengers which increase the risk of insecure travel.

4. Not finding a taxi in an emergency.

5. Problem of overcharged fees.

6. Irresponsibility of people.

Keeping in mind all these situations, having an easy-to-use taxi booking app in the smartphone can help a lot to make travel easier and hassle-free for people.

As an IT student, now is the prime time, I can try our best to develop a taxi booking app. Though this project is totally intended as a college project, it equally carries a value of providing services to people

3. Objectives

- To tracks all the information of Customer, Destination, Booking price
- To provide an app that can make travel secured by keeping the details of the taxi and the passenger
- To send notification after ride request to passenger

4. Background Study

In the context of Nepal, some enthusiastic people have started the trend of online shopping, online payment, and many more online services for the Nepalese people. Overall, we are doing quite well in making internet services a part of our daily lives.

Here is a statistical data showing the information about internet users in Nepal according to DataReportal.com (DataReportal):

- There are more than 10.78 million internet users till January, 2021 in Nepal.
- Above 13 million Nepalese people use social media (January, 2021)
- The number of mobile connections in Nepal till January, 2021 is above 32 million.

We, Nepalese people are now adapting our lifestyles according to the Digital Nepal theme. Today, internet usage is not only limited to entertainment, communication and information. People are now using internet services like online shopping, online payment and online studying. The success of the popular online shopping portal Daraz and other similar shopping portals have built trust for both developers and users. Moreover, we can now project that introducing a well-built taxi booking app may not turn into a disaster anymore.

Along with that, with the increase in people using internet and online services we can now project that the internet is now no longer just a medium of communication and internet. Nevertheless, these days' people are establishing their startups over the internet.

Another, very remarkable illustration is that people are using Online Payment Services. This is truly a milestone for Nepalese developers. Financial transaction over the internet not only requires a secured payment gateway. Equally important, trust is one of the major issues as sending their hard money virtually is a sceptical thing to most people who are not very familiar with digital payment systems. Thus, we can conclude that people have started to use internet services for their daily purposes as well.

Now, this project is an Online Taxi Booking App. It is equally important to understand the present context of the taxi system in Nepal.

Here are some points to show how the taxi services operate in Nepal especially in Pokhara.

- There is an umbrella Taxi Vyavasaya Samiti in different locations to look over the taxi services.
- The taxis should be registered in the convenient taxi stand of a locality.
- Unless the passengers call personally for a taxi, the taxis which are still in queue at the taxi stand should follow the queue.

Now, here are some points showing how people book or take a taxi usually without any apps:

- People personally call known taxi whenever required
- Another way is, people stand at the roadside looking for a vacant taxi
- People take a taxi from the taxi stand.

In such a situation, we believe a taxi riding app will be useful for people.

We would again like to mention that online taxi Riding is still not totally reliable due to which it hasn't created an impact on Nepalese people yet. We are still using the traditional ways for booking a taxi.

Now is a prime time for developing a user-friendly app because the competition is not high, and we can experiment with the app for its reliability.

5. Methodology

The right methodology can play a huge role in developing quality software for any company or organisation or individual (geeksforgeek). For the development of this undergraduate project Ride Sharing Mobile Application, I have decided to use an agile methodology. This is the way of managing a project by breaking the project into several phases of planning, modelling, development, testing, evaluation, and meeting, where requirements and solutions evolve through the collaborative effort and again the phases are repeated and finally, the product is handed over to the client.

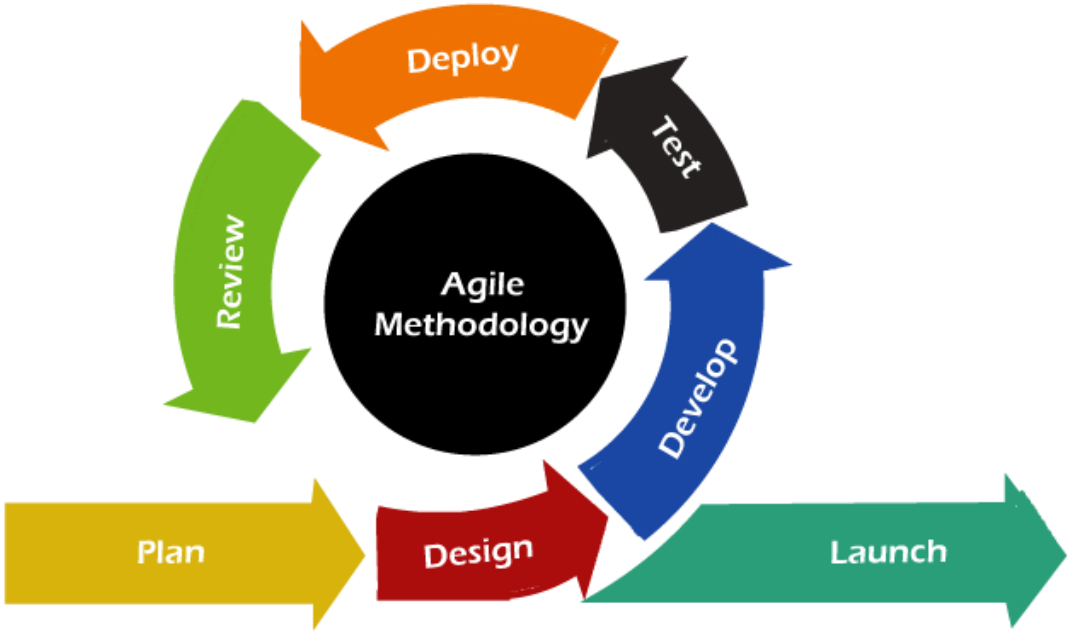


Figure 5.1 Agile Development Methodology

I decided to use this methodology because it allows both the clients and app developers to have proper communication which can lead to high customer satisfaction. As the regular communication between clients and developers the following advantages can be gained:

a. Requirement Change can be taken along with workflow.
b. Fault Detection
c. Increased Performance
d. Reduced risks
e. The flexibility in designing and quality improvement
f. Superior quality product

6. Requirement Document

6.1 Tools and Software Required

1. Android Studio
2. Dell i5 laptop ubuntu
3. GitHub
4. Android Smartphone
5. Rest API
6. VSCode
7. Azure Cloud Computing
8. Microsoft Word
9. Microsoft PowerPoint
10. Thunder client

6.2 Requirement Analysis

When the new software is built or made any update there will be many expectations of clients and users and the process of defining the user expectations for the upcoming or updated version of the software product is known as requirement analysis or requirement engineering of that software (techtarget). Requirement analysis of any software helps to determine the actual requirements of stockholders and review the overall graphical model of the software and its development. For this Ride Sharing mobile application there are the following functional and non-functional requirements:

6.2.1 Functional Requirement

Functional requirements of this application are listed below:

1. User Login and registration

2. User can see the OpenStreeMap and other information

3. Users can insert their own estimated cost and destination to create ride

4. User can share their ride request to all driver via ride button

6.2.2 Non-functional Requirement

1. Performance

The performance of any software product determines the quality of that software a lot. The user of this mobile application can have the expectation of highly reliable app performance so, for a great user experience the backend and frontend will be integrated properly.

2. Usability

The user of this mobile application software can have the expectation of attractive GUI and features which are designed by keeping the concentration of user-centred design.

3. Scalability

This Ride Share mobile application accepts and supports the end-users and satisfies the required things that it tries to deliver to its users.

4. Reliability

7. System Design

This is actually the first step of bringing our imagination into reality. Here, I designed the basics as well as all the functionality of how the app will work. Designing phase is where we draw the blueprint of our project.

System design is from where the app development actually begins. With a full-proof design in hand we can develop a good final product. In this section, there are some figures showing the basic design or skeleton of the app.

7.1 Basic Functioning of the App

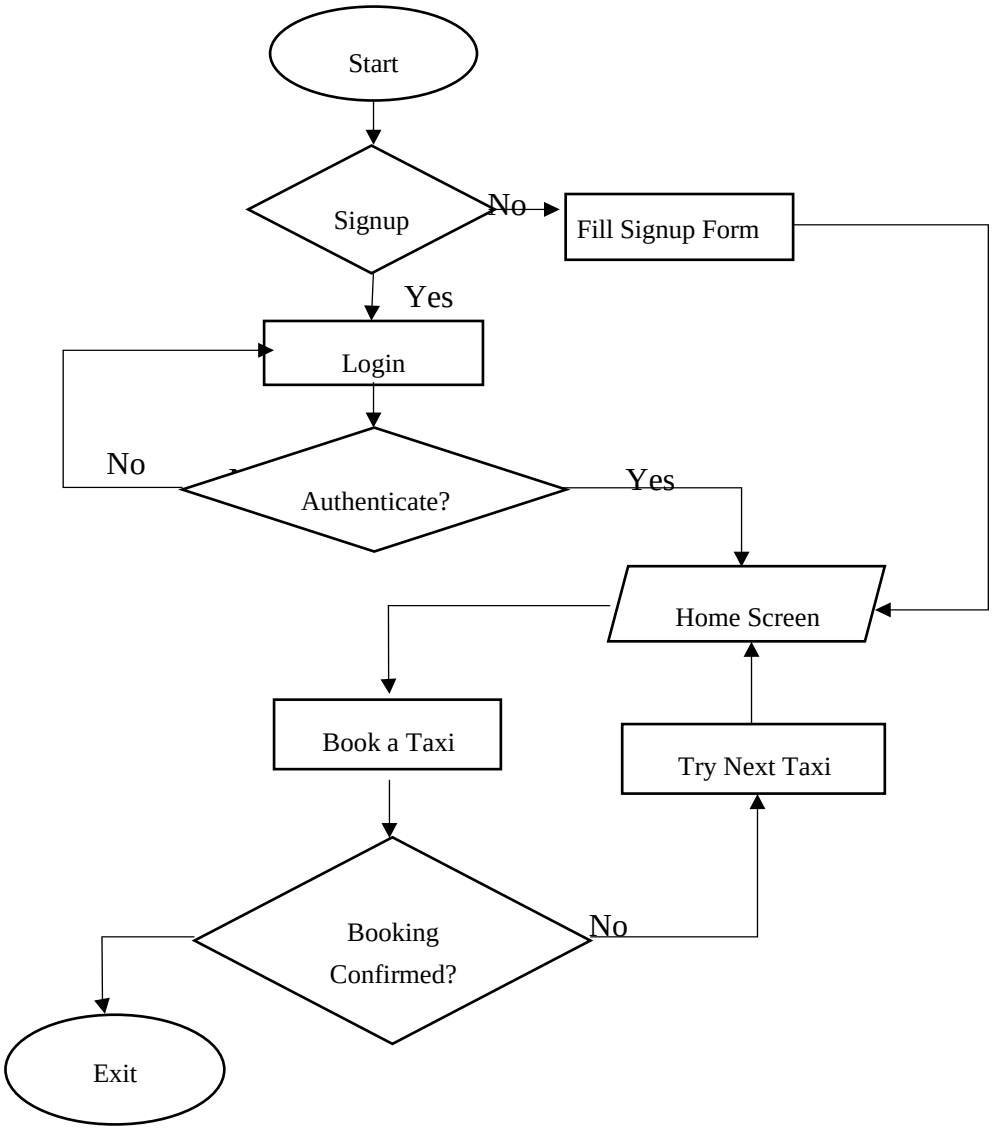


Figure 7.1.1: Basic Functioning of the App

7.2 Login and Signup Mechanism

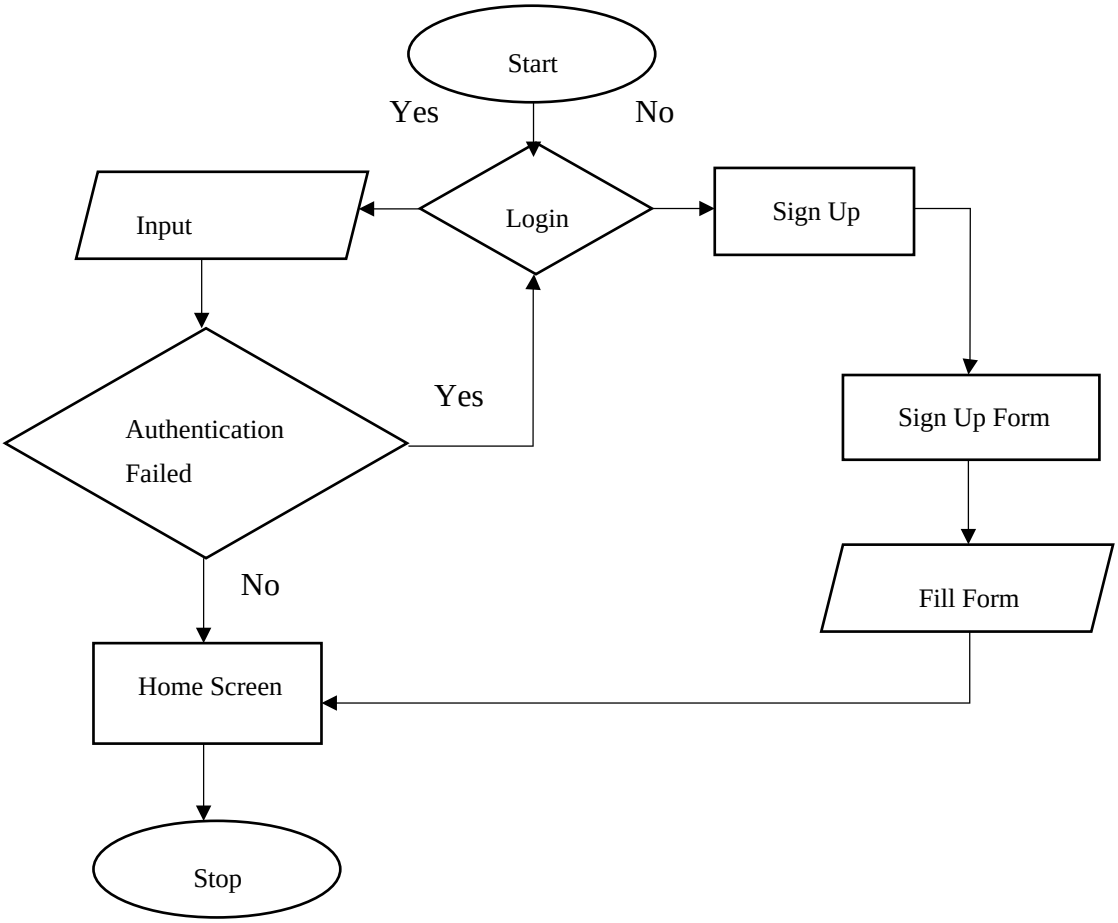


Figure 7.2.1: Flowchart for Login Mechanism

7.3 Taxi Ride Mechanism ER Diagram

ER Model is used to model the logical view of the system from data perspective. It consists of these components Entity, Entity Type, Entity Set (geeksforgeeks). The ER diagram for this Ride Sharing mobile app has general user activity, which are described on below along with the diagram:

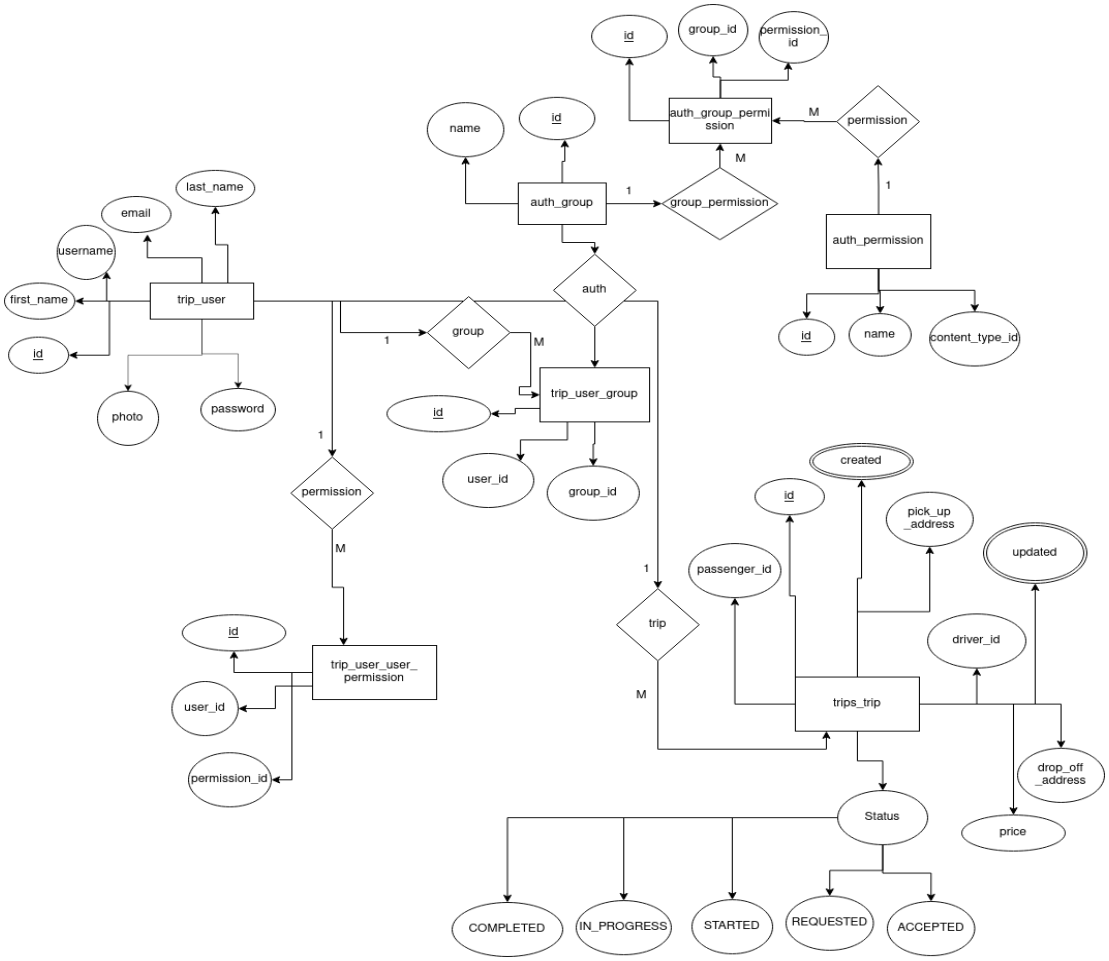


Figure 7.3.1: Taxi Ride Mechanism E-R Diagram

7.4 Taxi Ride Mechanism DFD

DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. The DFD for this Ride Sharing mobile app has general user activity, which are described on **DFD Level 1 and 2** below along with the diagram:

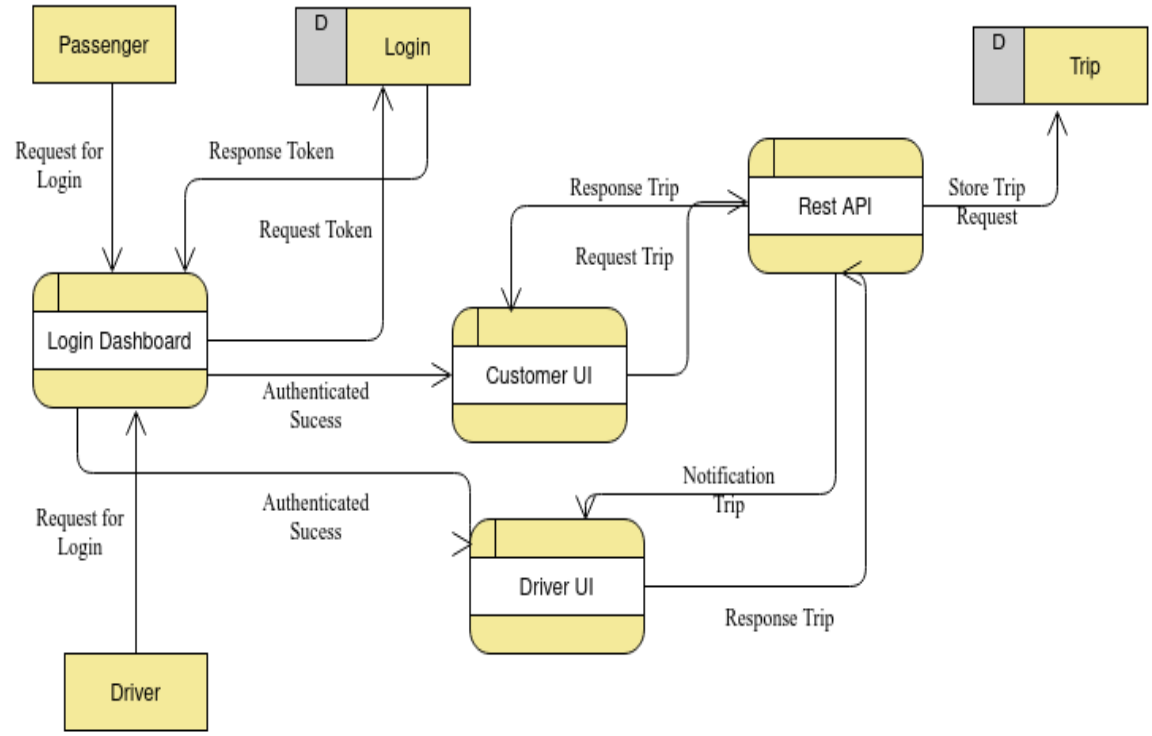


Figure 7.4.1: Mechanism for Ridesharing DFD Level 1

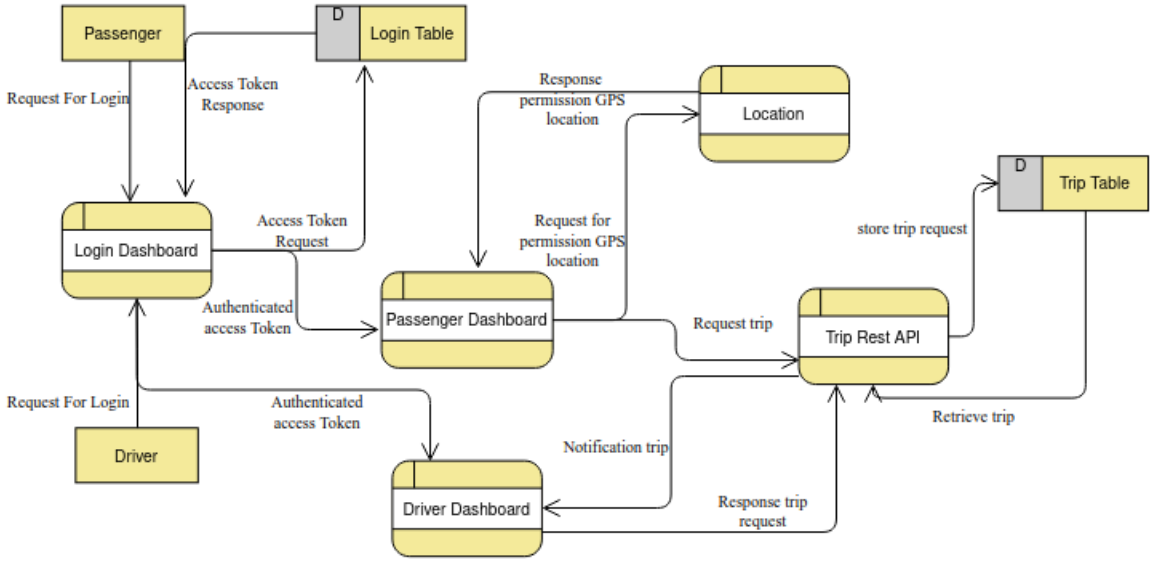


Figure 7.4.2: Mechanism for Ridesharing DFD Level 2

7.5 Taxi Ride Mechanism Activity Diagram

Activity diagrams of a software system are the graphical representations of workflows of activities and actions of users of that system. The activity diagram for this Ride

Sharing mobile app has general user activity which are described below along with the diagram:

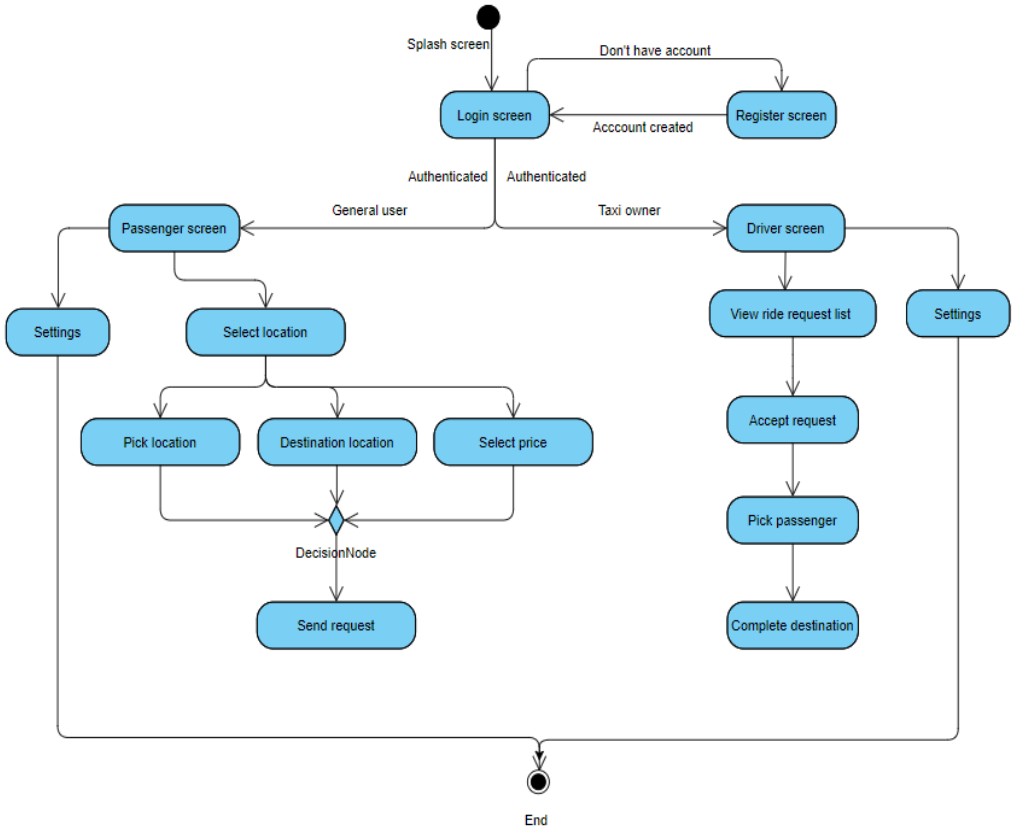


Figure 7.51 Activity Diagram

Passenger and Driver User Activity for this Ride Sharing mobile application

When the general user (passenger and driver) of this mobile app opens the app the splash screen is seen before a login page. If the users do not have an account for this app, they should have to register first. After the successful login, the users can see the user's main dashboard and be able to use all the useful features of this application.

8. Development

The development work was started with the use of a "Agile Methodology" for our project. It is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and

evaluating. Continuous collaboration is vital, both with team members and project stakeholders

As we know the process of planning, organising, coordinating, and controlling resources to achieve specific goals is also referred to as development. Software engineers have created more sophisticated tools known as IDE (Integrated Development Environment) to accommodate the evolution of software development. We have used Visual studio IDE and Android Studio which have developed Flutter App with dart language and Rest API with django framework . Likewise, for effective and worthwhile documentation, many inhouse office products like Ms-excel, MS-word, and PowerPoint were used. For the charts and flowcharts, the tools that are used are visualDiagram .Python and python Rest framework and postgresql are the backend tools used in the system.

We have used docker, Github and Microsoft Teams for our collaborative platform to discuss the problems and task division among the members with our project supervision of the team members in a teamwork manner.

Our system must store information about the booking taxi, destination, customers, Driver and so on. As a result, we've identified the major tables that will be implemented on the chosen RDBMS.So here is the database schema for our project “Taxi Ride”

We group of four individual students were confined to the development of the final year project and this is the most important, the chart of work assignment divided equally among the four members:

S.N.	Name of the member	Work assigned	Remarks
1.	Nabaraj Bhandari	● Deploy database with postgresql ● Deploy Rest API on docker and Azure VM	Completed within the given period of time

		● Mobile App Development On Flutter Dart ● Manage Role on team as Leader of Project ● Testing Both Whitebox and Blackbox testing	
2.	Lu Bahadur Karki	Documentation and system Design on app.diagram	Completed within the given period of time.
3.	Sajida khatun	Documentation and Blackbox Testing	Completed within the given period of time.
4.	Sagar Bohora	● Research and presentation	Completed within the given period of time.

Table 8.1: Task Division

9. APP Testing

Here, app testing is the software testing where the evaluation and verification of the **Taxi Riding app** are done. It is one of the major important parts of software development to verify the software product functional requirements, scalability, and overall performance. There are mainly two types of software testing is Blackbox testing and white box testing under which almost all software testing is categorised. So, for the **Taxi Riding** mobile application this testing is done which is described below.

9.1 Blackbox Testing

Blackbox testing of an application is the testing of external features by giving different combinations of inputs and responses. This testing plays a crucial role in assessing the defence and security controls of the application. Blackbox testing of the Ride Sharing app is given below:

9.1.1 Test case for passenger

9.1.1.1 Registration for User

S.N	Test case description	Expected result	Output result	Pass/Fail
1	Enter details (Username, Password, Confirm password, Email, First name and Last name) and choose Passenger and Driver checkbox. Click on the sign-up button.	Check validation and the user must register successfully to the system and navigate to the login screen.	Validation checked, user, registered, and navigated to the login screen.	**Pass**
2	Username and password validation and give a message if they enter Invalid	Checked the username and password validation and give an invalid message otherwise no message.	Validation for Username and password are checked and give invalid message in case of Invalid otherwise no message.	**Pass**
3	Try to register with the same email that are already registered.	Something Wrong message with email already registered.	Something Wrong message with email already registered.	**pass**

Table 9.1.1.1: Registration for user

Note: SS of UI for these functions are presented in the appendix section.

9.1.1.2 Login Passenger

S.N	Test case description	Expected results	Actual results	Pass/Fail

1	Enter a valid (registered) Username and password and click on the login button.	Successfully logged in and navigate to the user main dashboard.	Successfully logged and navigated to user main dashboard.	**pass**
2	Enter invalid (not registered, wrong Username or password) and press the login button.	Give a message with something wrong	Give a message with something wrong	**pass**

Table 9.1.1.2:Login Passenger

Note: SS of UI for these functions are presented in the appendix section.

9.1.1.3 Track current location

S.N	Test case description	Expected results	Actual results	Pass/Fail
1	If this app is installed on the user's mobile, then the user can see their current position on Openstreetmap if the user gives this permission to track their location on the current screen.	When the user allows this app for location then the user device will be tracked.	When a user gives their location permission to this app device's current location is tracked.	**pass**

Table 9.1.1.9.2: Track current location

Note: SS of UI for these functions are presented in the appendix section.

9.1.1.4 Rideshare

S.N	Test case description	Expected results	Actual results	Pass/Fail

1	If this app is login from passenger then they can ride to insert destination field and price to driver	When passenger ride request from app display msg something wrong if ride request fail	When passenger ride request from app display msg something wrong if ride request fail	**pass**

Table 9.1.1.9.3: Rideshare

Note: SS of UI for these functions are presented in the appendix section.

9.1.1.5 Passenger logout

S.N	Test case description	Expected results	Actual results	Pass/Fail
1	If this app is login from passenger then they can logout from the logout button	Passenger logout from logout button and navigate to login screen or msg invalid if logout failure	Passenger logout from logout button and navigate to login screen or msg invalid if logout failure	**pass**

Table 9.1.1.9.4: Passenger Logout

Note: SS of UI for these functions are presented in the appendix section.

9.1.2 API Testing for user

To test for the api we used thundered client tool

9.1.2.1

API Testing for user

S.N	Test case description	Expected result	Output result	Pass/Fail
1	Enter details on json format { "username": "",	Validation check for username, email, password and group and	Validation check for username, email, password and group and	**Pass**

	"1password": "", "password2": "", "email": "", "first_name": "", "last_name": "", "group": "", "photo": null }	display the api data on json format	display the api data on json format	

Table 9.1.2.1:API Testing for user

Note: SS of UI for these functions are presented in the appendix section.

9.1.2.2

/api/log_in testing

S.N	Test case description	Expected result	Output result	Pass/Fail

1	Enter details on json format
	{
	"username": "",
	"password": ""
	}
		Display the refresh and access token on thunder client body	Display the access and refresh bearer token	**Pass**

Table 9.1.2.2:/api/log_in testing

Note: SS of UI for these functions are presented in the appendix section.

9.1.2.3 /api/trip/create testing

S.N	Test case description	Expected result	Output result	Pass/Fail
1	Request a ride through following json format { "pick_up_address": "", "drop_off_address": "", "price": "", "status": “”	Validate the access jwt and response the ride	Validate the access jwt and response the ride	**Pass**

	}			

Table 9.1.2.3:/api/trip/create testing

Note: SS of UI for these functions are presented in the appendix section.

9.1.2.4 / api/trip/ testing

S.N	Test case description	Expected result	Output result	Pass/Fail
1	Get the data after validate jwt access	Response the /api/trip/create/ and passenger and driver detail data on the get method	Response the /api/trip/create/ and passenger and driver detail data on the get method	**Pass**

Table 9.1.2.4:/ api/trip/ testing

Note: SS of UI for these functions are presented in the appendix section.

9.1.3 Test case for driver

9.1.3.1 Registration for driver

S.N	Test case description	Expected result	Output result	Pass/Fail
1	Enter details (Username, Password, Confirm password, Email, First name and Last name) and choose Passenger and Driver checkbox. Click on the sign-up button.	Check validation and the user must register successfully to the system and navigate to the login screen.	Validation checked, user, registered, and navigated to the login screen.	**Pass**
2	Username and password validation and give a message if they enter Invalid	Checked the username and password validation and give an invalid message otherwise no message.	Validation for Username and password are checked and give invalid message in case of Invalid otherwise no message.	**Pass**

3	Try to register with the same email that are already registered.	Something Wrong message with email already registered.	Something Wrong message with email already registered.	**pass**

Table 9.1.3.1: Registration for driver

9.1.3.2 Login driver

S.N	Test case description	Expected results	Actual results	Pass/Fail
1	Enter a valid (registered) Username and password and click on the login button.	Successfully logged in and navigate to the user main dashboard.	Successfully logged and navigated to user main dashboard.	**pass**
2	Enter invalid (not registered, wrong Username or password) and press the login button.	Give a message with something wrong	Give a message with something wrong	**pass**

Table 9.1.3.2: Login driver

Note: SS of UI for these functions are presented in the appendix section.

9.2 White box testing

It is software testing that tests the internal structures or workings of an application as opened to its functionality. The internal folder and code structure of this application is based on clean code architecture where all data, domain, and UI layers are separated. White-box testing of any software involves the testing of the software code to verify for the following:

a. There are no internal security holes

b. No broken or poorly structured paths in the coding process

c. The flow of specific inputs through the code

d. Expected output

e. The functionality of conditional loops

f. Testing of each statement, object, and function on an individual basis.

9.2.1 User registration functionality test case (passenger and driver)

S.N.	Scenario and expected results	Actual results	Test result (Pass/Fail)
1	When the user gives input from the signup screen check the user input is the incorrect data type and it all format is as expected request model then, the code for inserting the user input data into the database and create the user with user username and password.	The data format of user input is verified if all data types are as expected then the user data are inserted in the database and created a user with user email and password.	**pass**
2	When the user input data are not in format due to any reason it will catch the error and return an error message.	When the input data are not as expected then the output will be an error message.	**Pass**
3	When the already used email is	If the input email value is	**pass**

	given as input in email field then it will give an error message.	already registered it gave an error message.	

Table 9.2.1:User registration functionality test case (passenger and driver)

9.2.2 User login functionality test (Passenger and Driver)

S.N.	Scenario and expected results	Actual Results	Test result (Pass/Fail)
1.	The login function for the user is to authenticate the users. For authentication username and password are used. So, when the user enters a validated username and password of the Ride sharing app then only users will get access to the system.	The validated username and password input of the users is sent then the user gets access to the system.	Pass
2.	When the user enters the invalid email or password then it will give a message i.eSomething Wrong.	Invalid email or password input of the user gets an invalid message in response.	Pass

Table 9.2.2: User login functionality test (Passenger and Driver)

9.2.3 Other features of the app

S.N.	Function	Expected results	Actual Results	Test result (Pass/Fail)

1.	Ride Request	When a passenger enters a Ride Request with destination or drop_off_address and price. notification to the response of the ride accept	If the user input is the correct destination and price and pressed on ride request, then it is notified to the driver and response from the driver view site.	Pass
2.	notification	when passenger ride request it notified to all driver and response as the choice of status **ACCEPT and CANCEL**	After a ride request it notification to all driver as a request of ride status choice **Accept and Cancel**	Pass
3.	My current location	When the user tab is on the third tab of the bottom navigation bar then Openstreetmap is displayed with a current location tracked if users allow location permission to this app.	After the user gives location permission to this app then a red mark with the text “My current location” is seen with the current location of the device on openstreetmap	Pass
4.	Logout	For logout Firebase sign Out function is used while sign-out it clears the current user id from the app so, if other users try to login from the same device the current user data will be available.	Log out when clicked on the logout button and clear the user id of the current user.	Pass

Table 9.2.3: Other features of the app

Note: SS of code for these functions are presented in the annexures section.

10. Challenges faced in the Development

As this project is solely a Taxi Riding App, definitely a fully functional Map API and REST API is one of its major requirements. Here are some of the major challenges I faced during the development of this project.

- Creating REST API /api/trip
- Bearer Authentication on flutter
- Display Current Camera on map
- Create VM on Azure

Azure cannot provide a suitable and appropriate free vm and regional vm. I cannot find regional free service on azure so i choose a Europe regional vm that slow the cloud

11. Project Results

Finally, after following the software development lifecycle, I was able to implement the project. Some of the features are still under development. Therefore, I have kept them as the future enhancement of the project.

Major project deliverables:

- App to book a taxi
- Single apps for drivers and passengers
- Enter location manually for destination
- Book a taxi
- Send notification after ride request to passenger

12. Future Enhancements

As for now, this is a college project. Therefore, there are different limitations. However, the concept of the project, an online taxi-booking app is a demanding subject. Therefore, this project can be implemented in real life from a professional level as well.

As a developer, we need more skills to develop an app ready to publish. There are many criteria such as; error free, security, privacy, etc. Therefore, here I have mentioned some of the future enhancement that can be implemented in this project:

- Integrated google maps api
- Online Payment Integration
- SMS or Phone Number Registration
- Creating three separate apps, for taxi owners, passenger, and admin
- SMS Notification
- Taxi Movement Location
- Drivers and Passengers Rating

13. Conclusion

On the endnote, the overall objective of this project is to develop a user-friendly and reliable online taxi-booking app **Taxi Riding App**. This report consists of the overall information of this project. The main motive of this project is to provide a platform to easily book a taxi whenever required.

Taxi Riding App project is developed using Restful API for Django rest framework and PostgreSQL for backend and flutter framework for Android Devices in Android Studio.

Taxi Riding App is an app through which you can easily book a taxi without any hassle. In this era, we are living in everything is done with a click then why not booking a taxi with a click. It will help users to easily book a taxi, estimate the taxi fare and ultimately save time and provide security to the users. It is a secured way of booking a taxi as the mandatory details of the taxi driver and passenger will be stored safely as a reference.

We are sincerely thankful to our Project Supervisor, **Mr. Ritesh Thapa** for his continuous support. Likewise, we are thankful to our class.

As a BCA student, this project helped us to learn project development and project management. Moreover, this project is one of the aspects for our personal as well as professional growth in the field of IT.

14. References

DataReportal. (n.d.). *DIGITAL 2022: NEPAL*. https://datareportal.com/reports/digital-2022-nepal#:~:text=Internet%20use%20in%20Nepal%20in,percent)%20between%202021%20and%202022.

Department, S. R. (n.d.). Number of mobile cellular subscriptions in Nepal. *https://www.statista.com/statistics/501013/number-of-mobile-cellular-subscriptions-in-nepal/#:~:text=Number%20of%20cellular%20subscriptions%20Nepal%202000%2D2019&text=In%202018%2C%20the%20number%20of,in%20Nepal%20reached%2039.18%20million.*

geeksforgeek. (n.d.). *Agile Development Models.* https://www.geeksforgeeks.org/software-engineering-agile-development-models/.

geeksforgeeks. (n.d.). *Introduction of ER Model.* https://www.geeksforgeeks.org/introduction-of-er-model/.

techtarget. (n.d.). *requirements analysis.* https://www.techtarget.com/searchsoftwarequality/definition/requirements-analysis.

15. Annexures

(a) Screenshot of UI (passenger and driver)

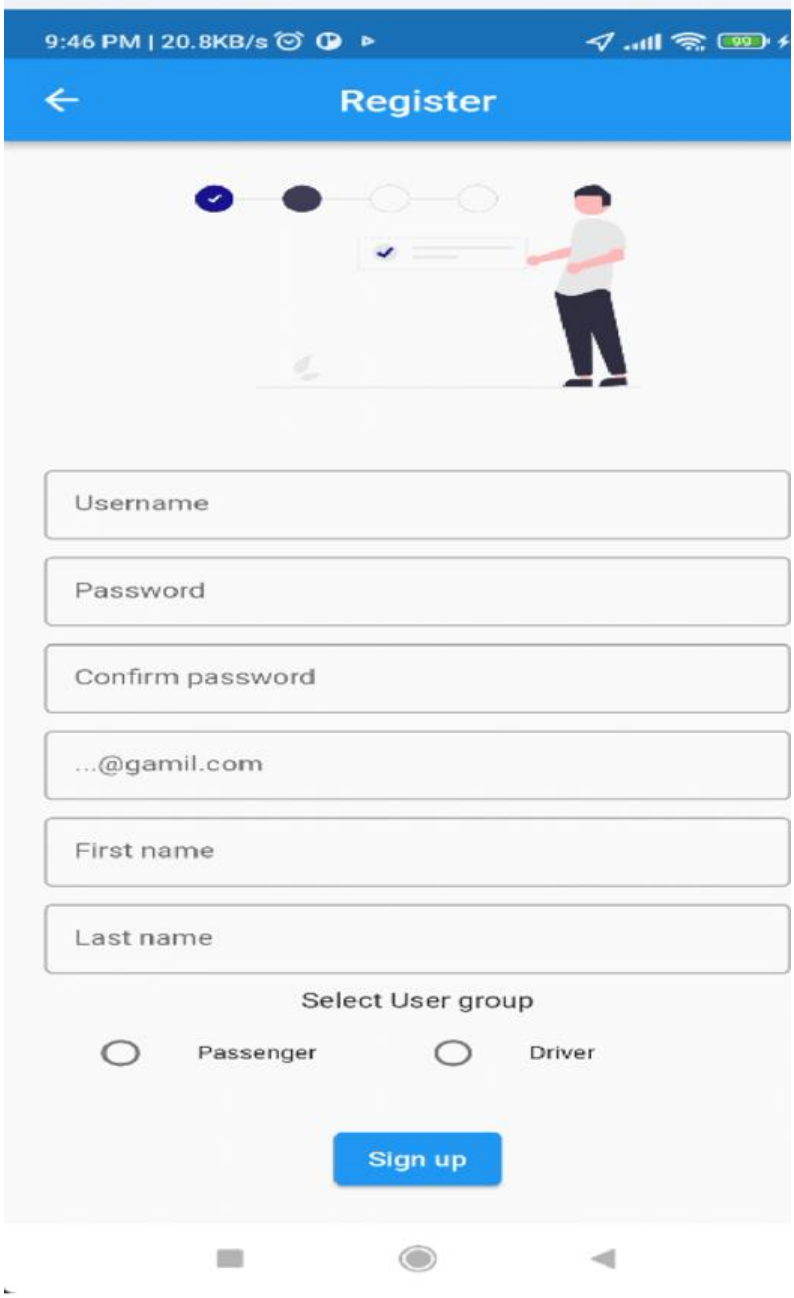


Figure 15.1: Registration Screen

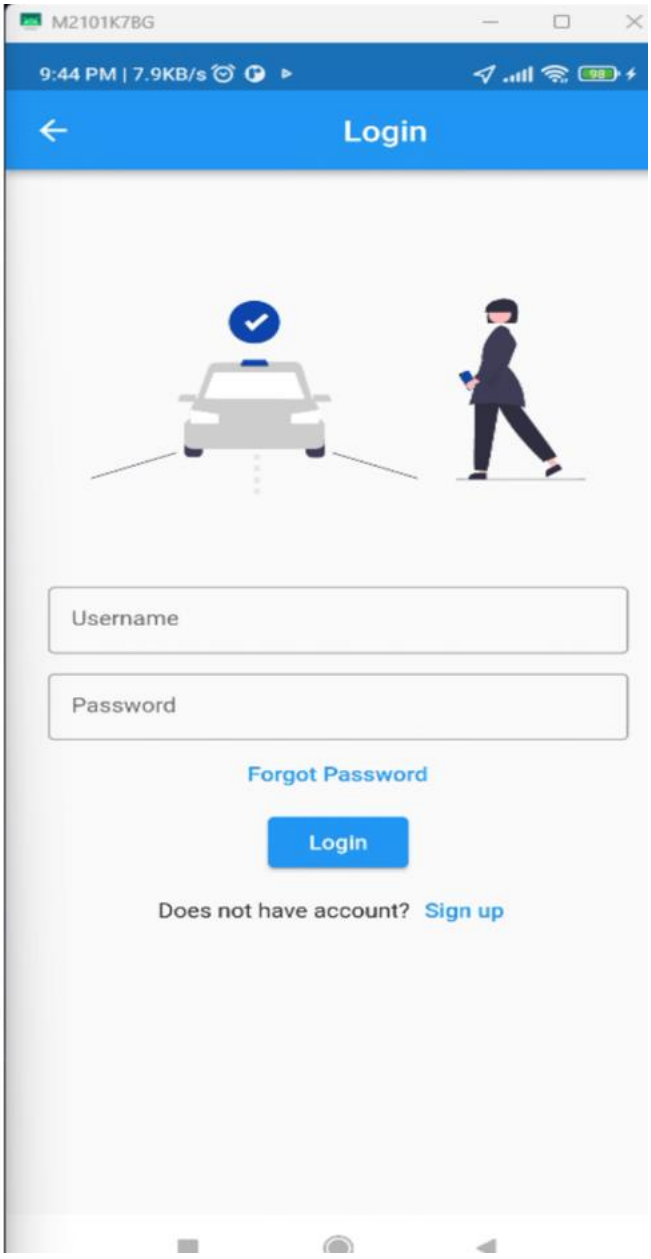


Figure 15.2:Login screen

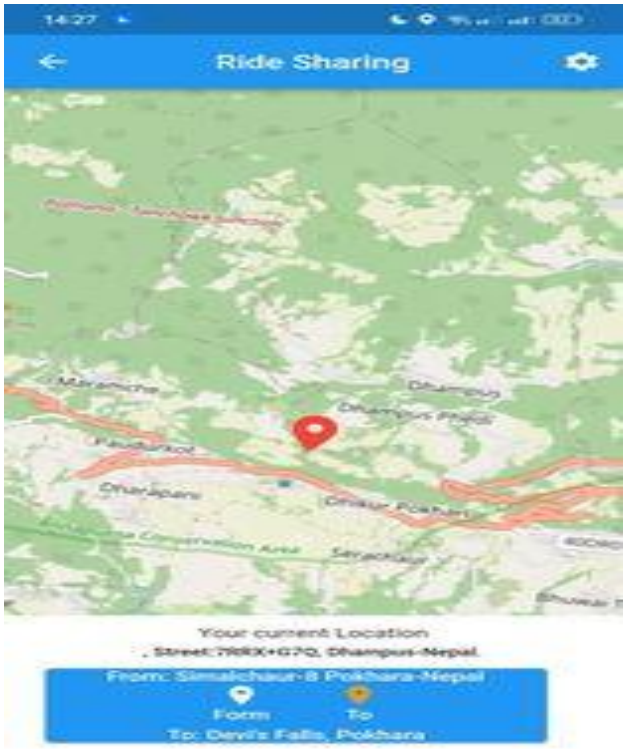


Figure15.3: Homescreen

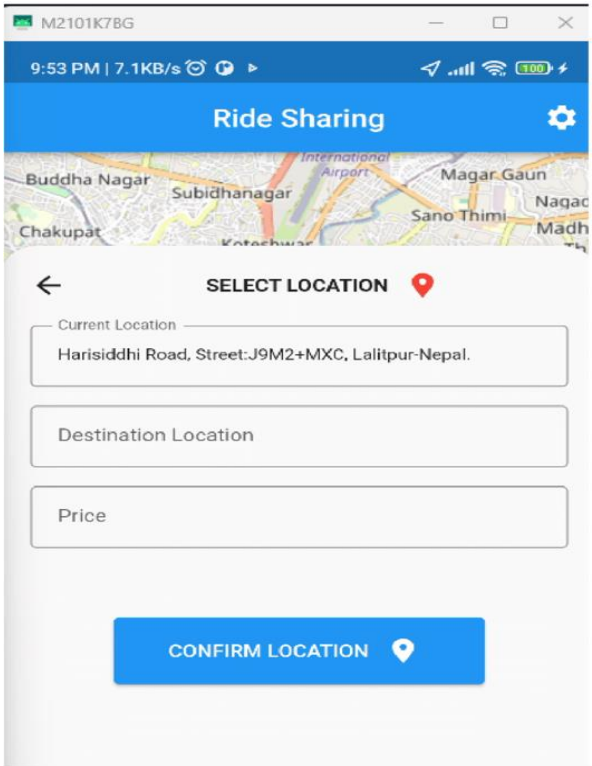


Figure 15.4: Ride Request

Screenshot of app functionality code

```dart
/// splash screen to navigatte user according to their group and login state
void getUserStatus() async {
  final userResponse =
      await ref.read(localDataSourceNotifier).getAuthResponse();
  if (userResponse != null &&
      userResponse.id > 0 &&
      userResponse.username.isNotEmpty) {
    Navigator.of(context).pushReplacement(
      MaterialPageRoute(
        builder: (_) =>
            OpenStreetMapScreen(isDriver: userResponse.group == 'driver'),
      ), // MaterialPageRoute
    );
  } else {
    await Navigator.of(context).pushReplacement(
      MaterialPageRoute(
        builder: (_) => const LoginScreen(),
      ), // MaterialPageRoute
    );
  }
}
```

Figure 15.5: Whitebox Testing Login

```dart
/// geeting all passenger request list business logic
@override
Future<Either<ResuestlistResponse, Failure>> requestList() async {
  try {
    final url = Uri.parse(
      'http://20.24.200.114:8003/api/trip/',
    );

    final userResponse = await _localDb.getAuthResponse();
    final accessToken = userResponse?.access;
    final response = await http.get(
      url,
      headers: {
        'Authorization': 'Bearer $accessToken ',
      },
    );
    final parsed = json.decode(response.body);
    log(parsed.toString());
    final result =
        ResuestlistResponse.fromJson(parsed as Map<String, dynamic>);
    return Left(result);
  } catch (e) {
    return const Right(
      Failure(
        errorMessage: 'Something went wrong !',
        errorCode: '',
      ), // Failure
    ); // Right
  }
 }
}
```

Figure 15.6: Whitebox Passenger Request

```dart
/// hive db (local db) class notifier
final localDataSourceNotifier = Provider((ref) => HiveDataSource());

You, 3 seconds ago | 1 author (You)
class HiveDataSource extends ChangeNotifier {
  /// user login response data cache
  cacheAuthResponse(LoginResponse item) async {
    var cacheBox = await Hive.openBox<LoginResponse>('AUTH_RESPONSE_BOX');
    cacheBox.put('authResponse', item);
    cacheBox.close();
    notifyListeners();
  }

  /// user login response data from cache
  Future<LoginResponse?> getAuthResponse() async {
    final getBox = await Hive.openBox<LoginResponse>('AUTH_RESPONSE_BOX');
    final authRes = getBox.get('authResponse');
    getBox.close();
    return authRes;
  }

  ///clearing the user login response data
  Future<void> clearCacheData() async {
    final getBox = await Hive.openBox<LoginResponse>('AUTH_RESPONSE_BOX');
    const data =
        LoginResponse(refresh: '', access: '', username: '', email: '', id: 0);
    getBox.put('authResponse', data);
    cacheAuthResponse(data);      You, 3 days ago • fix: current user address …
  }
}
```

Figure 15.7: Whitebox Testing Notification

```dart
/// ride requesting by passenger business logic
@override
Future<Either<RideResponse, Failure>> rideRequest({
  required RideRequest rideRequest,
}) async {
  try {
    final url = Uri.parse(
      'http://20.24.200.114:8003/api/trip/create/',
    );
    var requestBody = {
      "pick_up_address": rideRequest.pick_up_address,
      "drop_off_address": rideRequest.drop_off_address,
      "price": rideRequest.price,
      "status": rideRequest.status,       You, 2 days ago • update ride req
    };
    final userResponse = await _localDb.getAuthResponse();
    final accessToken = userResponse?.access;
    final response = await http.post(
      url,
      headers: {
        'Accept': 'application/json',
        'Authorization': 'Bearer $accessToken ',
      },
      body: requestBody,
    );
    final parsed = json.decode(response.body);
    log(parsed.toString());
    final result = RideResponse.fromJson(parsed as Map<String, dynamic>);
    return Left(result);
  } catch (e) {
    return const Right(
      Failure(
        errorMessage: 'Something went wrong !',
        errorCode: '',
      ), // Failure
    ); // Right
  }
```

Figure 15.8:Whitebox Testing Ride Request

```dart
/// new user sign up business logic
@override
Future<Either<SignUpResponse, Failure>> userSignup({
  required SignUpRequest signUpRequest,
}) async {
  try {
    final url = Uri.parse(
      'http://20.24.200.114:8003/api/sign_up/',
    );
    var requestBody = {
      "username": signUpRequest.username,
      "password1": signUpRequest.password1,
      "password2": signUpRequest.password2,
      "email": signUpRequest.email,
      "first_name": signUpRequest.first_name,
      "last_name": signUpRequest.last_name,
      "group": signUpRequest.group,
    };
    final response = await http.post(
      url,
      headers: {
        'Accept': 'application/json',
      },
      body: requestBody,
    );
    final parsed = json.decode(response.body);
    log(parsed.toString());
    final result = SignUpResponse.fromJson(parsed as Map<String, dynamic>);
    return Left(result);
  } catch (e) {
    return const Right(
      Failure(
        errorMessage: 'Something wen wrong !',
        errorCode: '',
      ), // Failure
    ); // Right
  }
```

Figure 15.9: Whitebox Testing Sign-up

```dart
/// userlogin business logic
@override
Future<Either<LoginResponse, Failure>> userLogin({
  required String username,
  required String password,
}) async {
  try {
    final url = Uri.parse(
      'http://20.24.200.114:8003/api/log_in/',
    );
    var requestBody = {
      'username': username,
      'password': password,
    };
    final response = await http.post(
      url,
      headers: {
        'Accept': 'application/json',
      },
      body: requestBody,
    );
    final parsed = json.decode(response.body);
    final result = LoginResponse.fromJson(parsed as Map<String, dynamic>);
    _localDb.cacheAuthResponse(result);
    log(parsed.toString());
    return Left(result);
  } catch (e) {
    log(e.toString());
    return const Right(
      Failure(
        errorMessage: 'Something wen wrong !',
        errorCode: '',
      ), // Failure
    ); // Right
  }
}
```

Figure 15.10: Whitebox Testing Login