FCCU CARPOOL SERVICE

SOFTWARE REQUIREMENTS SPECIFICATION

A. DEPARTMENT OF COMPUTER SCIENCE

Forman Christian College (A Chartered University)

FCCU Carpool Service

(FORMANRIDE)

A Webapp for riders and drivers to share ride to FCCU

Senior Project

BY LOVE SERVE ONE ANOTHER

Estd. 1864

FORMAN CHRISTIAN COLLEGE

(A Chartered University)

Primary Advisor: **Akheem Yousaf**
Secondary Advisor: **Dr. Aasia Khanum**

Presented by:

231485855	AFNAN AHMED
241547206	HATIB ZUBAIR
231485432	MUHAMMAD MUZAMMIL

Table of Contents

Table of Contents ..... **iii**
Revision History ..... Error! Bookmark not defined.
1. Introduction and Background **1**
1.1 Product (Problem Statement) 1
1.2 Background 2
1.3 Scope 2
1.4 Objective(s)/Aim(s)/Target(s) 2
1.5 Challenges 3
1.6 Learning Outcomes 3
1.7 Nature of End Product 4
1.8 Completeness Criteria 4
1.9 Business Goals 4
1.10 Related Work/ Literature Survey/ Literature Review 5
2. Overall Description **8**
2.1 Product Features 8
2.2 User Classes and Characteristics 9
2.3 Operating Environment 9
2.4 Design and Implementation Constraints 9
2.5 Assumptions and Dependencies 10
3. Functional Requirements **11**
3.1 Name of Use-Case 1 11
3.3 Requirements Analysis and Modeling 18
4. Nonfunctional Requirements **23**
4.1 Performance Requirements 23
4.2 Safety Requirements 23
5. Other Requirements **24**
6. Revised Project Plan **25**
7. References **25**
Appendix A: Glossary **26**
Appendix B: IV & V Report **27**
(Independent verification & validation) **27**

1. Introduction and Background

Ridesharing is increasing in developed nations due to everyday commute expenditures increasing daily. Rising costs of fuel have made daily commutes expensive to the masses. Students have a burden of expenses. Cars are one of the primary sources of air pollution; burning gases are deteriorating the atmosphere. Considering the commute costs and the air pollution due to transportation, we have devised an efficient idea to decrease the usage of more cars by creating a carpool app for FCCU Students who would share their ride and the fuel prices among them. It will significantly decrease the fuel expense of students. Over time, the Parking space of FCC is also less compared to the number of students, due to which long lines of cars, wrong parking, and lack of space cause students hurdles and wastage their class time. The Unique point of this app is that the problem here is that no two students go at the same time to university and vice versa. Due to liberal arts education at FCCU, all have different timings. FCCU Car Pooling Service will unite two students through a website who will be going to the same destination. The student willing to share his ride will notify; if a student also wants to join, he can reply and contact to ride.

1.1 Product (Problem Statement)

There are security risks involved in a third-party commute service; keeping that in mind, a dedicated web app, which is university specific, does the job. However, suppose a ridesharing app made only university-specific, granting access to students based on verifying their student card will maximize security, safety and peace of mind. In that case, it is a win-win situation for both students, and both are saving and getting affordable transportation. Parking in FCCU is a significant problem. It causes blockage inside and outside of campus. Too many students bring their vehicles with FCCU, which requires more space.

Another major thing that this carpool app would help with is dealing with traffic and pollution. Too many vehicles on the road contribute to more traffic and pollution because of the smoke. The carpooling app will help reduce the number of vehicles as many students would come together rather than alone.

1.2 Background

This web application would have multiple functionalities such as:

1. A driver and rider portal to be used respectively.
2. The driver pinning his route on the map with respective timings and other details, which will then be uploaded on a listing.
3. Rider, with the feature of seeing the listings of all drivers coming from nearby areas.
4. Option to accept the ride and get connected with each other.

This application would have databases of drivers and users and make them connected. The software finds people nearby and offers them to ride share. We use the following methods for creating it:

1. A website hosted on an SSL Certified domain.
2. A backend connected with databases for information.
3. API integration of Maps by Google.

1.3 Scope

Web Application is designed considering students' current challenges due to rising fuel and expensive transportation costs. It is a purposefully student-centered web app to solve their problem. My motive is to tackle the challenges students face and provide a peer-to-peer ride-sharing platform to facilitate them in the era of rising fuel and maintenance costs. Air pollution is the topic of the day, and an application that encourages sharing of rides and reduces carbon footprints is highly beneficial. Which also provides opportunities for socializing between students on the go. It would be a student-to-student service and would reduce the need for a third person to provide pick-and-drop, eventually saving costs. The web app is intended to automate and modernize how students travel to university.

1.4 Objective(s)/Aim(s)/Target(s)

The aims and objectives of the project would be:

- Reducing overall traffic congestion on the roads
- Reduce peak hour congestion

- Reducing single occupancy car trips by implementing a carpooling system.
- Promoting alternative modes of transport.
- Improve parking in FCC
- Save money by sharing the cost of driving one car.
- Reduce the number of cars on the road.
- Reduce pollution and carbon dioxide emissions.
- Reduces driving-related stress for participants
- Provide social connections for university students

1.5 Challenges

One of the challenges we may encounter during development is integrating geo-locations and mapping functionality using the Google Maps API. The use of these services may incur costs and charges, which need to be carefully considered and managed within the free quota of google plans. Additionally, it's important to note that certain ride tracking services provided by Google may be exclusive to specific customers and not available for the general public. We will need to assess and ensure that the required services align with our project goals and resources.

1.6 Learning Outcomes

This project offers several valuable learning outcomes. First, we will gain experience connecting the front-end and back-end components of a web application, which is an essential skill in modern web development. Instead of relying on outdated technologies, we will have the opportunity to work with state-of-the-art tools like ReactJS, which is widely used in the industry for creating dynamic and interactive websites.

Moreover, by choosing Django as our back-end framework instead of traditional options like PHP, we will enhance our understanding of modern web development practices. Django provides pre-built libraries and modules that simplify common tasks, allowing us to focus more on the core functionality of our web app. This hands-on experience with Django will undoubtedly contribute to our skill set and provide us with valuable insights into building robust and scalable applications.

Overall, this project promises to be an exciting learning experience, enabling us to explore new technologies, improve our development capabilities, and gain practical knowledge in creating web applications with enhanced functionality and efficiency.

1.7 Nature of End Product

The nature of this product is a utility web app, a platform independent, which makes the driver and rider get connected and split the ride costs, neglecting any interference or unusual charges by the third-party app. It would be students centric and FCCU centric as well. Safe and Secure as well.

1.8 Completeness Criteria

The ultimate goal of this project is to provide a useful and efficient platform for connecting drivers and riders. The success of the project would be measured by the extent to which both drivers and riders find value in utilizing its services.

For riders, the web app would serve as a convenient tool for their daily commute. They would be able to request rides and easily connect with available drivers in their area. This would save them time and provide a reliable means of transportation to their desired destinations.

On the other hand, drivers would benefit from the web app by gaining access to ride requests from nearby riders. They would be able to view the location of the rider, enabling them to plan their routes and efficiently pick up and drop off passengers. This would not only optimize their driving routes but also provide them with a steady source of income.

By successfully facilitating this connection between drivers and riders, the web app would meet its criteria for fulfillment. It would enhance the convenience and reliability of the commuting experience for both parties involved, making their journeys more efficient and enjoyable.

1.9 Business Goals

At present, this service is created by students, specifically for students, and it is offered completely free of charge. The website is available for everyone to use without any fees. However, in the future, if there is a need for improvements and expansion that require financial resources, we might consider integrating Google ads to sustain its operation. This would allow us to generate revenue from advertisements while keeping the website accessible to users without any direct cost. By incorporating this approach, we can continue to enhance and grow the service while ensuring its availability to a wider audience.

1.10 Related Work/ Literature Survey/ Literature Review

Ridesharing has snowballed in popularity since the introduction of the first ridesharing app in 2009. Research has found that the main factors driving the growth of the ridesharing industry are convenience, cost-effectiveness, and increased access to transportation for underserved communities.

There are many carpooling applications all over the world. Uber, Careem, and Ola are some of their examples. These applications allow the user to share the commute with different users who are going the same way so that the fare can be shared and everyone reaches the destination on time. These applications provide the solution to high rates of single rides and help adjust with others to be affordable. For about the same price as public transit, passengers save time and enjoy a more comfortable ride, while drivers save some money by sharing the commute cost. This option is better than public transport, like the bus, because it picks you up from your location and you don't have to wait at stations for buses or trains. It allows you to have a good experience at a low cost. The disadvantages of these solutions are that it is for everyone. A person headed to work will share with a student headed to school or university. This can have a time constraint on both people, and they could be late because of the other. When headed to different locations, you will meet different people every day, which could not be a good experience as some might come late, making the user late for their work or university. The individuals who carpool together may finish work at different times and have to reschedule their rides, which could affect the other user. Some users might cancel at the last minute, which would mean the fare is high for the other passengers. These are the issues facing the carpool applications, with users not very fond of using these applications.

To counter these issues, our application should be university-specific. This will help the students head to one place to get together and ride. The ending destination for all the users will be the same. Users who come from the same area and use their cars can use this application to benefit the city's traffic. FCCU has minimal parking space, and many times, there have been blockages because of no parking space, and many students must park their cars outside the university. The carpool will solve this problem, and less parking space will be used if students do not bring their cars and use the carpool. This will also allow the students to interact with their peers and get to know each other. They could decide in university if somebody has a problem going back so that the rest can manage accordingly. This will help in reducing petrol consumption and pollution.

Analysis of similar apps and websites:

UBER/Careem:

They all are pioneers of digitizing and making the commutes accessible and fares reasonable.

The central point in these services, which will be a plus point in our web app, would be that a third-person driver would give the services with his revenue in providing service regularly. If the rider and driver both are students, the need for another person and taking his services will be bypassed, saving the extra costs a student would need to get to a destination.

The car, driver, and service-providing software costs will be eliminated, and the commute will become more reasonable. It does not have peak factor charges which makes the drives expensive.

Airlift, SWYL:

Before the Covid times, these startups provided ride-hailing services using public transport for commutes, making travelling cheap. It was more affordable than going even on a bike.

However, the economic crisis in Pakistan after covid made their business unsustainable, and they closed their operations. So, they are not working anymore, and not all areas were covered, and they would not come individually to a place or residence on the map pinned location.

OLA Cabs:

It is a successful carsharing app and a website used in India, but it is not university specific and caters to a general audience. It is in India and has a web app as well in use.

Carpolyn:

It is a Pakistani App with a database of users, their car registration number, and their transportation route. It does not have a map facility to track or find drivers live. It lacks detailed features like tracking, contacting users, sending or receiving ride requests, etc. It is a static app with little instructiveness.

2. Overall Description

2.1 Product Features

The product features include:

User Authentication: This would help to limit the use of the web app, within the accessibility of FCCU Students only. Students would need to login through their university email account and create their account. And then would sign up as Rider or Driver

DRIVER: This will open all the features and functionalities for the driver, e.g., information of car, license number, location, per/km charges. They can add their own schedule, return time and arrival time. The status of Requests from the riders.

RIDER: This will open the features and functionalities for the rider, e.g. students going from nearby with their schedule would be shown to them, timings in a listing manner with option to request a ride and contact the driver, and request for a specific date or time as well as round trip or one sided.

REQUEST: A feature for the rider to select preferable driver according to their own preferences from the listing.

Maps Location: This will be used by the driver, to see the place of rider from the drivers origin.

All Rides: This will be the portion for riders, where all the lists of nearby drivers with filters based on proximity of driver and rider.

ACCEPT / DECLINE: A feature for the driver to accept a request of rider for ridesharing.

Status: The rider will be shown the status of his ride, if his request was accepted or rejected or still pending.

2.2 User Classes and Characteristics

By the scope of this project, the users will be divided into two categories. One will be the Rider, and one will be the Driver.

DRIVER: This class can create a ride listing, which will be seen by others nearby to request for rideshare, setting up the time table. Only this class would create a listing and will have the authority to accept or decline the requests of rideshare.

RIDER: This class will be able to request a ride from the list of all nearby drivers (Driver Class). Set-up a point of pick-up on the maps. Requesting the driver as well.

2.3 Operating Environment

Our web application is designed to be accessible through web browsers such as Chrome, Safari, and others. It operates on various devices, including mobile phones, laptops, and computers. To provide a seamless experience, the application requires internet connectivity and GPS location services for accurate pickup location tracking. Users can easily set their pickup location by manually pinning it on the map or by using their device's GPS functionality. Whether you're on the go with your mobile phone or working from your computer, our web app ensures that you can access and use it conveniently.

2.4 Design and Implementation Constraints

As our web app is built using ReactJS and Django, there were certain limitations that arose from using these modern technologies. One of the constraints we encountered is the inability to provide users with live routing and directions within the webapp itself. This limitation is primarily due to the restrictions imposed by the Google Maps API, which offers comprehensive services but at a cost, primarily targeting corporate and enterprise customers. To overcome this constraint, we have implemented a workaround by incorporating a functionality that allows users to be redirected to the native Google Maps app on their devices, where they can access and utilize the live directions feature.

It's important to note that another constraint we faced is the usage limitations and costs associated with the Google Maps API. After a certain threshold of usage, the API becomes a paid service, requiring financial resources to sustain its functionality within our web app. We are aware of this constraint and have devised strategies to address it, such as exploring potential monetization options or seeking alternative mapping solutions if necessary.

Despite these constraints, we provide the best user experience possible within the limitations of the technologies and resources available to us.

2.5 Assumptions and Dependencies

Assumptions:

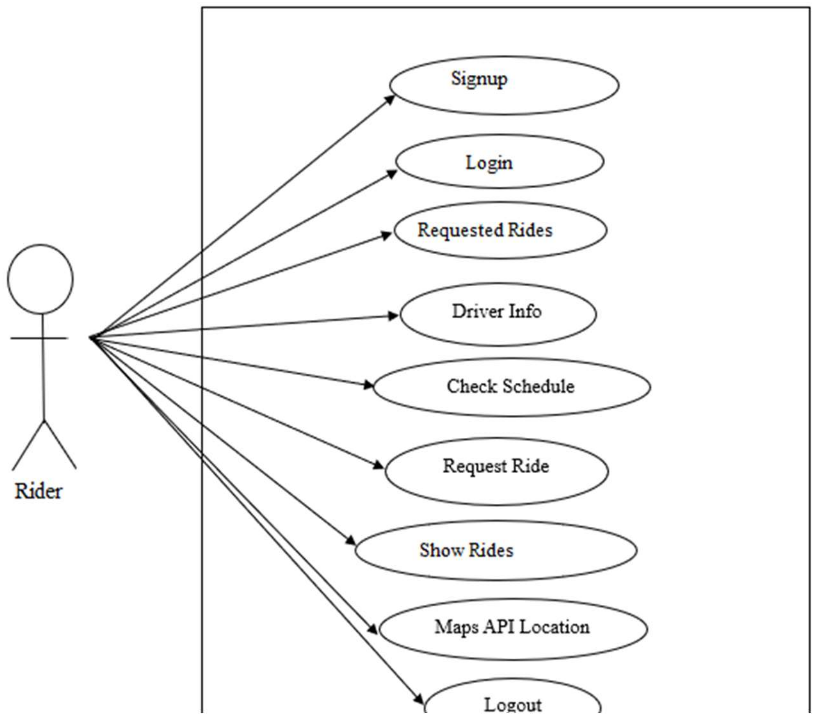
1. User knows basic working of browser and website.
2. Users would have Google Maps downloaded and have cellular or internet connection to connect to server and client.
3. User is a student of FCCU, and if he has valid driving license if he is signed up as Driver.

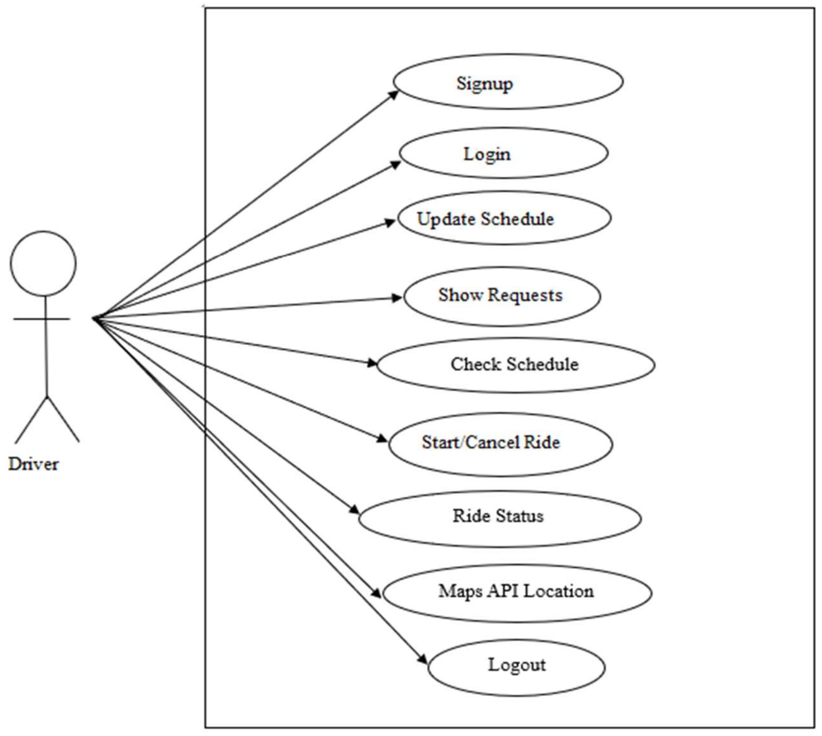
Dependencies:

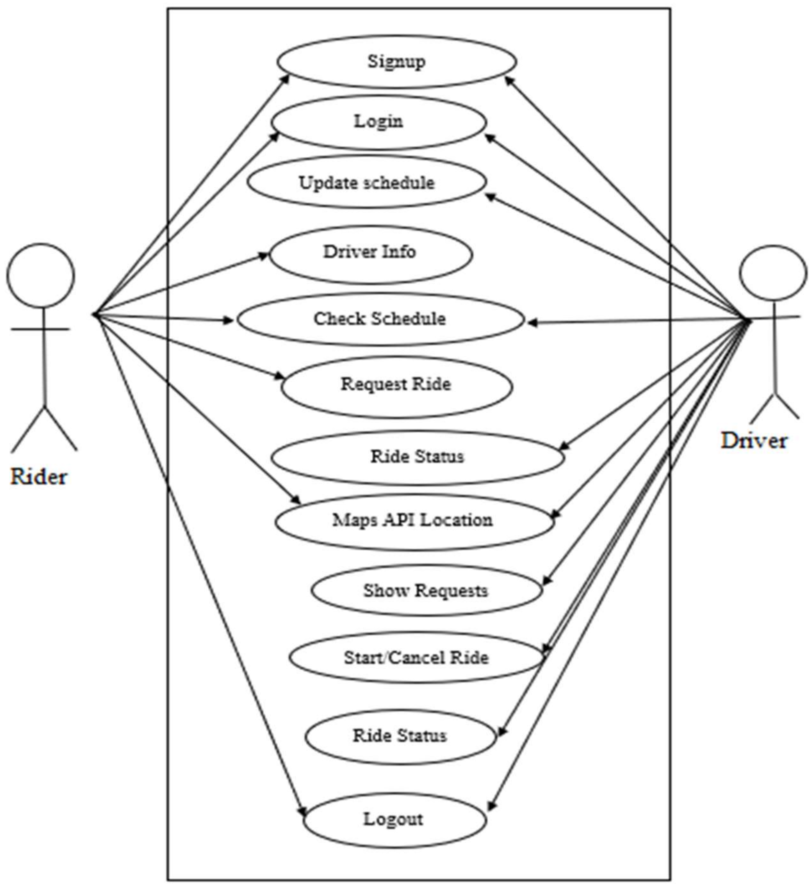
We are relying on Google Maps API, Django as a backend with the SQL Lite as database for storage.
Google is an API provider for maps, for the working of the project, it would be dependent on their allocation of resources.
Some of assumptions are:

1. Django with SQL lite would be enough for the working as a backend.
2. Google Maps API would be free for our limited usage, the threshold to be met. If the threshold of routes, services are more utilized we may integrate google ads for revenue of operational charges.

3. Functional Requirements






Signup
Login
Update schedule
Driver Info
Check Schedule
Request Ride
Ride Status
Maps API Location
Show Requests
Start/Cancel Ride
Ride Status
Logout
Rider
Driver

3.1.1 Use-Case 1: Registration as RIDER or Driver

Identifier	UC-1 Registration	
Purpose	Allows user to register as Driver or Rider	
Priority	High	
Pre-conditions	Sign up Button Clicked on Web Page	
Post-conditions	Relevant and Accurate Details Entered	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	The user selects the “signup” option	Displays Sign up form on screen
2	User opts as a Rider or Passenger	System shows different forms if it is Rider or Driver
3	User enters their details and information	System verifies the details
4	User submits the details	System checks the details and checks if details are different to existing record. Then, saves data and registers the user
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	The user’s account has already been created	System displays that id they are trying to sign up with is already in use

Table 1: UC-1

3.1.2 Use-Case 2: Login

Identifier	Login	
Purpose	Allows user to Login	
Priority	High	
Pre-conditions	Login option selected after launching webpage	
Post-conditions	Relevant details entered in Login option	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	The user selects the Login option.	System verifies their choice and displays the form on screen.
2	User enters his id and password	Verifies the details entered
3	Login clicked	User is logged into his account after verification of details
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	User forgot his password and clicks that option	The system will provide them with an option to reset the password

3.1.3 Use-Case 3: Search Rides (Rider)

Identifier	UC-3 Show Rides
Purpose	Passenger Student can check available rides for upcoming days
Priority	High
Pre-conditions	User has already logged in the system as a Rider
Post-conditions	All available rides shown

Typical Course of Action

S#	Actor Action	System Response
1	The user clicks on Show Rides	System shows all available rides
2	User clicks Check Schedule	More detailed page opens with schedule and address, vehicle info etc
3		

Alternate Course of Action

S#	Actor Action	System Response
1	No rides are shown	System generates a message that no rides are available.

3.1.4 Use-Case 3: Send Request(Rider)

Identifier	UC-3 Send Request
Purpose	Passenger Student can send request to driver with selecting their preferred timings
Priority	High
Pre-conditions	User has already clicked Check Schedule
Post-conditions	Request sent successfully

Typical Course of Action

S#	Actor Action	System Response
1	The user clicks on desired day, time and Return trip or not	System books desired timings and schedule for rider
2	User clicks checkmark of Return	The systems marks for complete Round Trip and not only one sided.
3		

Alternate Course of Action

S#	Actor Action	System Response
1	No time slots are available	System generates response no slots available

3.1.5 Use-Case 4: Check Requested Rides(Rider)

Identifier	UC-4 Requested Rides	
Purpose	User can check the status of his rides	
Priority	Medium	
Pre-conditions	User is logged into the system. User has sent at least one ride request	
Post-conditions	All information with Status displayed	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	User clicks on Requested Rides	System shows all Rides with their Status, Pending or Rejected or Accepted
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	User has not requested any ride previously	System generates a message that no rides history record is available

3.1.6 Use-Case 5: Upload Schedule (Driver)

Identifier	UC-5 Upload Carpool Ride Schedule	
Purpose	Driver User Uploads ride schedule weekly	
Priority	High	
Pre-conditions	Driver has clicked Schedule from navbar	
Post-conditions	Schedule is uploaded and can be seen from Schedules page of from Rider Section in Show Rides	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	User clicks on Time/Day in Schedule	System adds the timings respectedly
2	User enters his Arrival and Return for respected Days	System adds it to their schedule
3	User clicks on submit Schedule	System verifies the information and adds the schedule
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	User enters invalid timings	System gives error to enter correct/accurate timings

3.1.7 Use-Case 6: Update/Remove Schedule (Driver)

Identifier	UC-6 Update/Remove Carpool Ride Schedule	
Purpose	Driver User Updates his Schedule	
Priority	High	
Pre-conditions	Driver user is logged in the system There is an existing schedule	
Post-conditions	Schedule is uploaded	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	User clicks on Update Schedule	System verifies if there is an existing schedule and then display the same schedule to be updated
2	User enters his new schedule timings or just removes existing schedule	System verifies the information and checks timings are correct or removes existing schedule
3	User clicks on submit Schedule	System verifies the information and updates the ride details
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	User enters invalid timings	System gives error to enter correct/accurate timings
2	User enters same timings as before	System gives warning message that timings are not updated

3.1.8 Use-Case 7: Update/Add Information(Rider/Driver)

Identifier	UC-7 Add/Update Information	
Purpose	User adds/updates his information (name, place, phone veh#)	
Priority	Medium	
Pre-conditions	User is logged into the system	
Post-conditions	Information added/removed/updated	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	User clicks on update information	System displays a page with all information of user
2	User selects specific option they want to change	System displays a new box for information to be updated
3	User clicks on update information	System verifies the information and updates the information
Alternate Course of Action		

S#	Actor Action	System Response
1	User enters invalid information	System gives error to enter correct information

3.1.9 Use-Case 8: Accept/Decline Ride Request (Driver)

Identifier	UC-8 Accept/Decline Ride Request	
Purpose	The driver accepts or declines request of rider	
Priority	High	
Pre-conditions	Driver receives Ride Request and is in Show Requests Page	
Post-conditions	The status is updated according to the driver's response	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	User clicks on Pending Rides when gets notified	System displays a page with all pending rides
2	User selects accept or decline ride button	System adds or rejects the ride in status
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	User clicks on pending rides	System displays "no rides in waiting list"

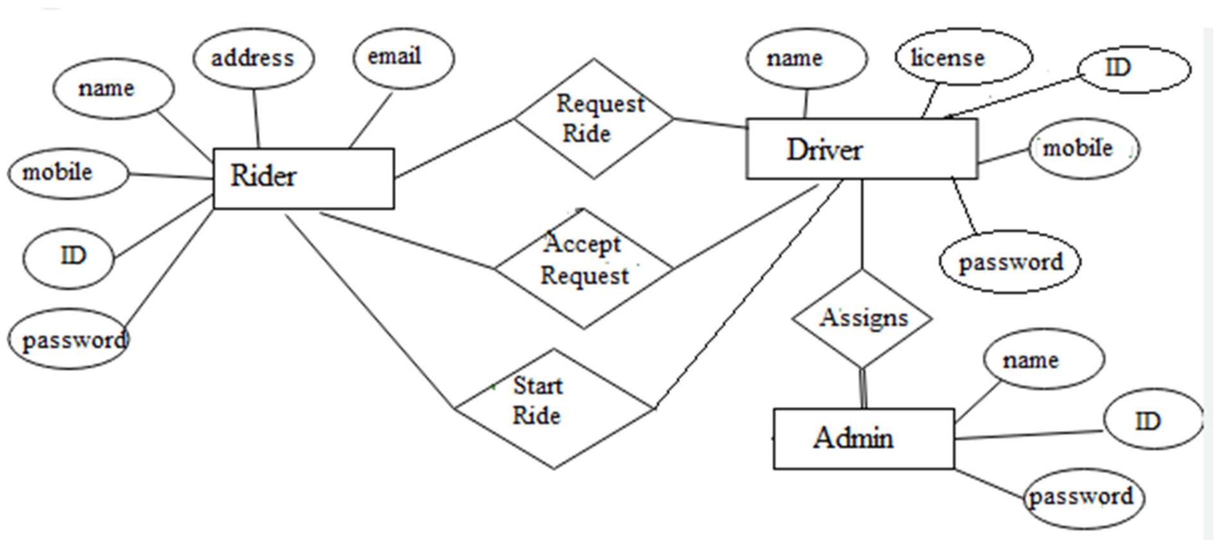
3.1.10 Use-Case 9: All Accepted Rides (Driver)

Identifier	UC-9 All Accepted Rides	
Purpose	Driver can Start or Cancel the Ride which is already accepted	
Priority	Medium	
Pre-conditions	Driver has accepted a Ride	
Post-conditions	Maps Page is opened with option of Redirection to Riders Origin	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Driver clicks on Accepted Rides	System displays a page with buttons to Start Ride or End Ride
Alternate Course of Action		
S#	**Actor Action**	**System Response**
2	User clicks on Accepted Rides	Driver has not accepted any ride previously, so the area is blank

3.1.11 Use-Case 9: Start Ride (Driver)

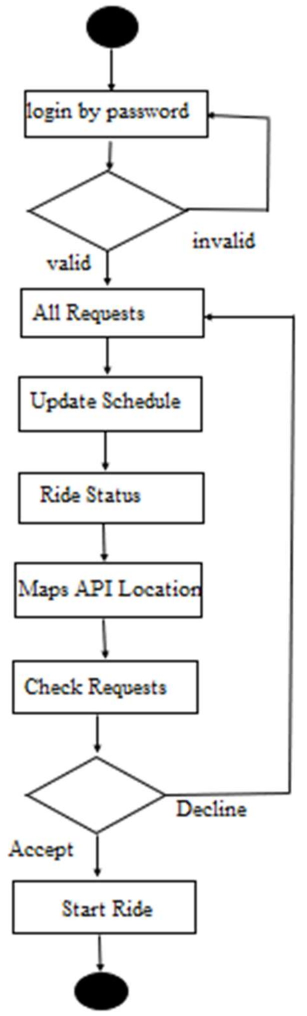
Identifier	UC-9 Start Ride	
Purpose	Driver can Start or Cancel the Ride which is already accepted	
Priority	Medium	
Pre-conditions	Driver has accepted a Ride	
Post-conditions	Maps Page is opened with option of Redirection to Riders Origin	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Driver clicks on Start Ride	System opens a Google Maps, with origin of Rider
Alternate Course of Action		
S#	**Actor Action**	**System Response**
2	User clicks on Cancel Ride	The Ride is cancelled, and no maps page will open

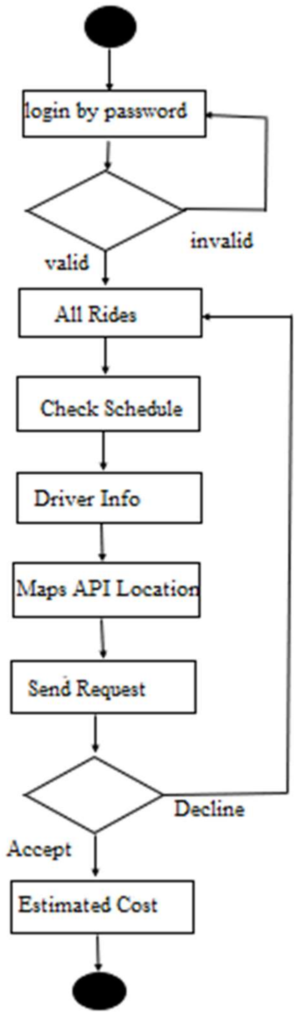
3.2 Requirements Analysis and Modeling



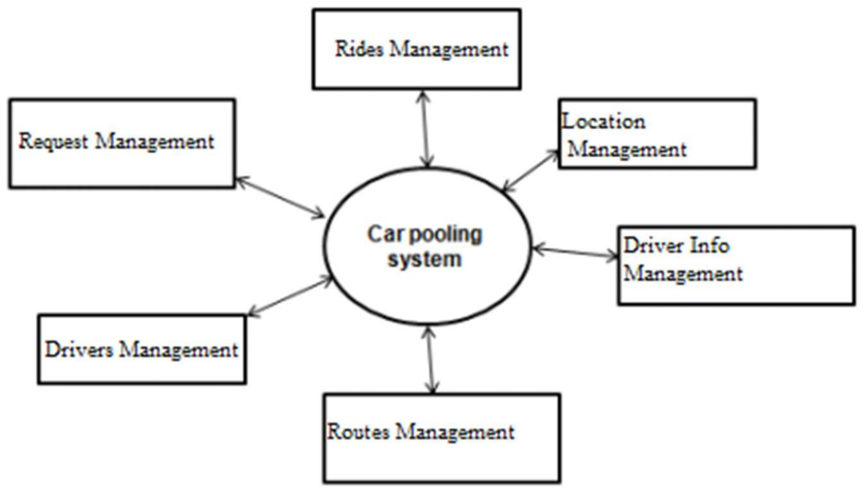
[

The figure 1 shown above shows the ER diagram of our system. This diagram shows the how each use case of our system is linked to the user. Let’s look at one of these use cases. For example, there will be only one server but the users viewing it could be many, and they would have access to all the public information of the server such as the front end and the ability to interact with it.

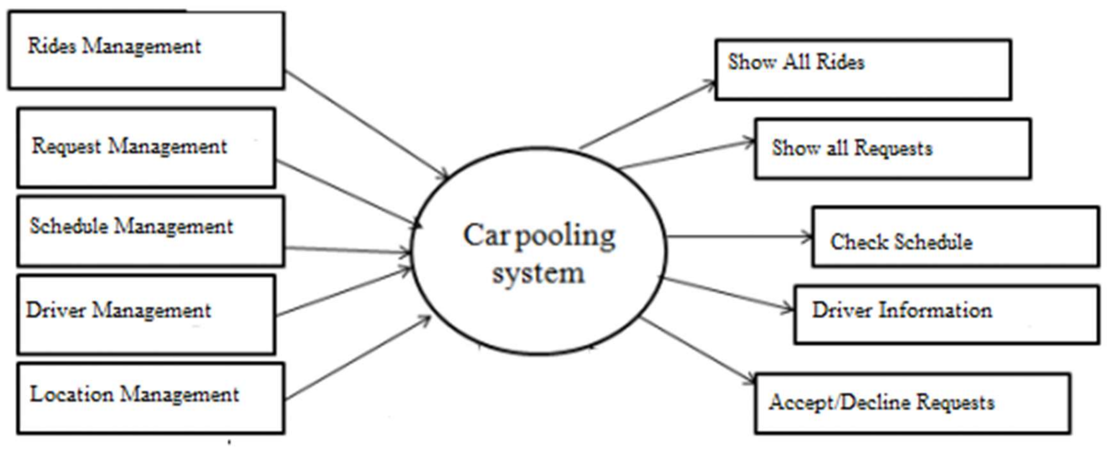

login by password
invalid
valid
All Requests
Update Schedule
Ride Status
Maps API Location
Check Requests
Decline
Accept
Start Ride


login by password
invalid
valid
All Rides
Check Schedule
Driver Info
Maps API Location
Send Request
Decline
Accept
Estimated Cost

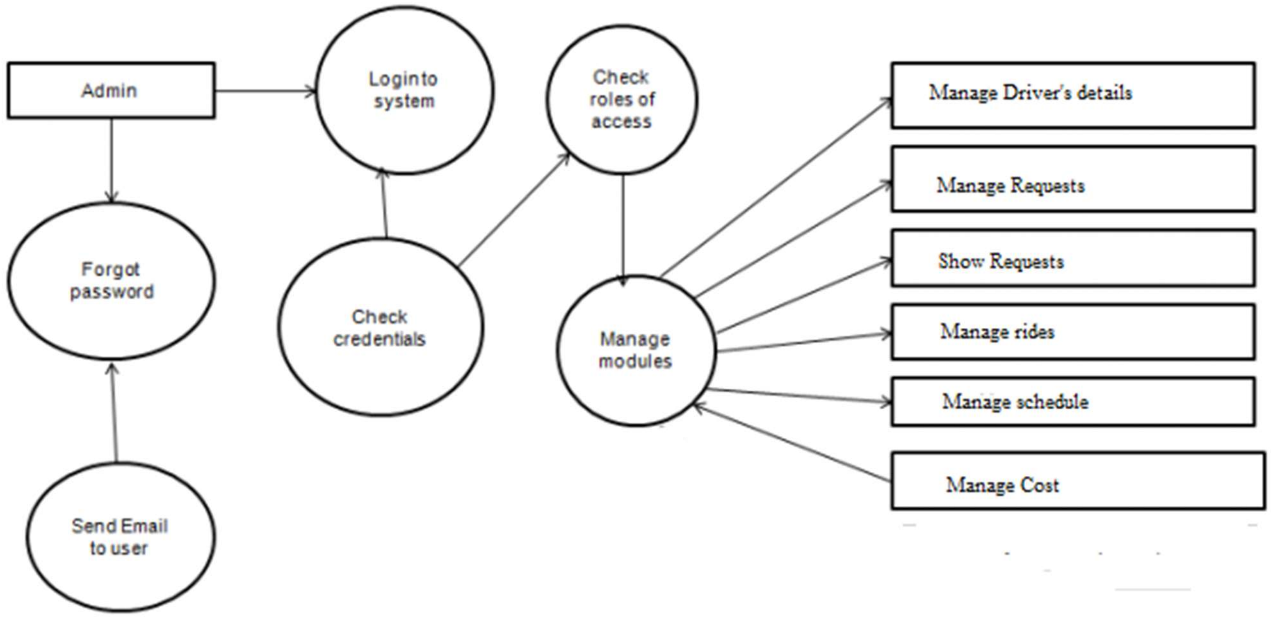
DFD Level 0:



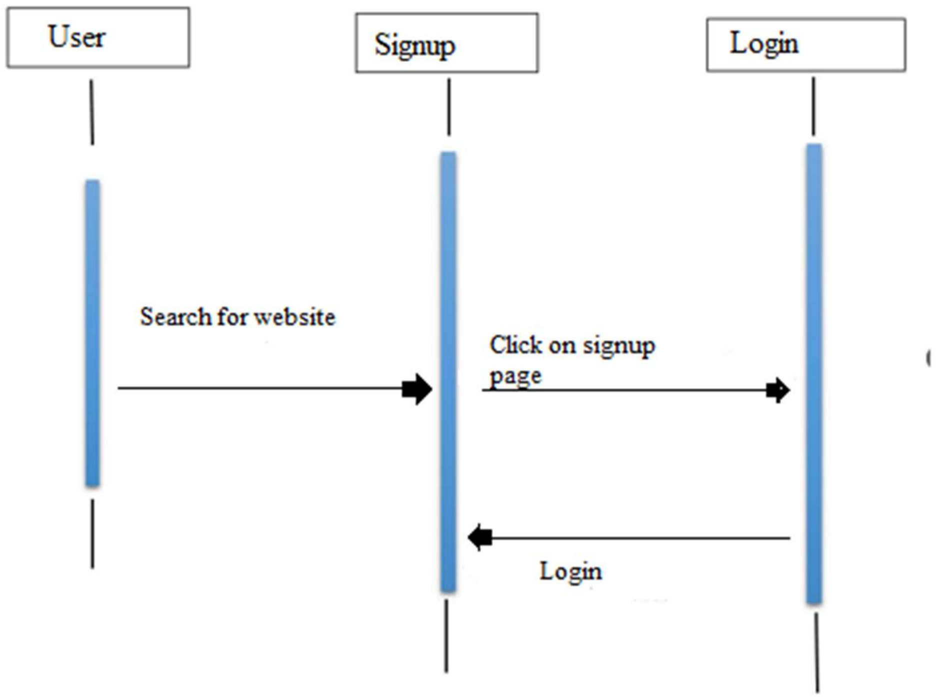
DFD Level 1:

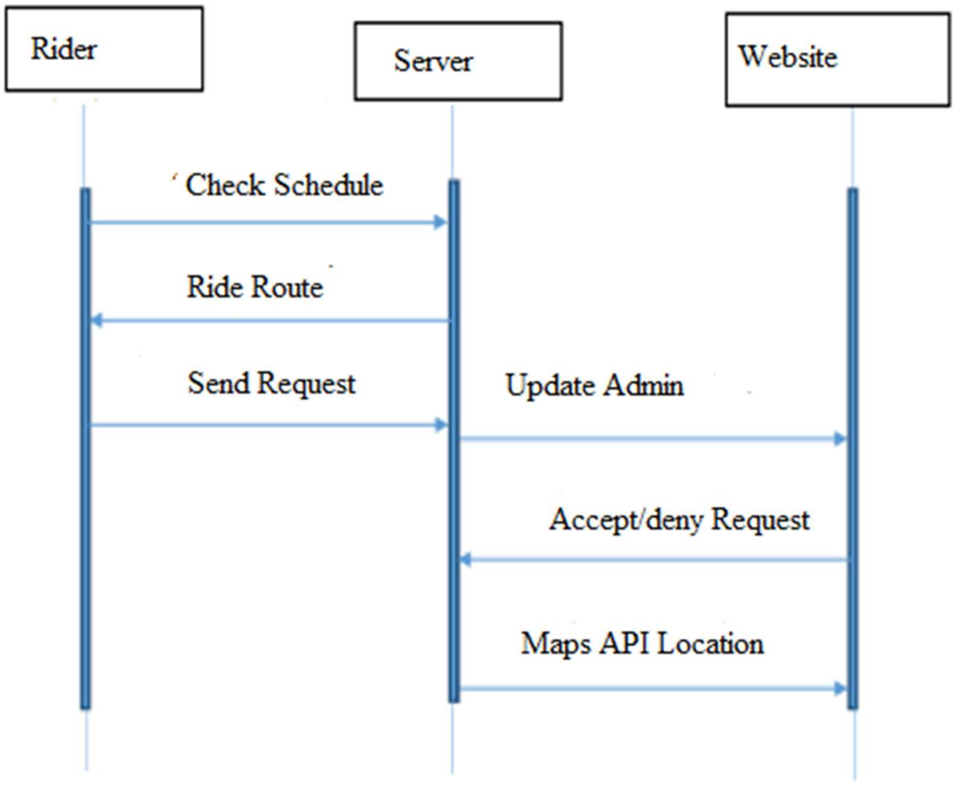


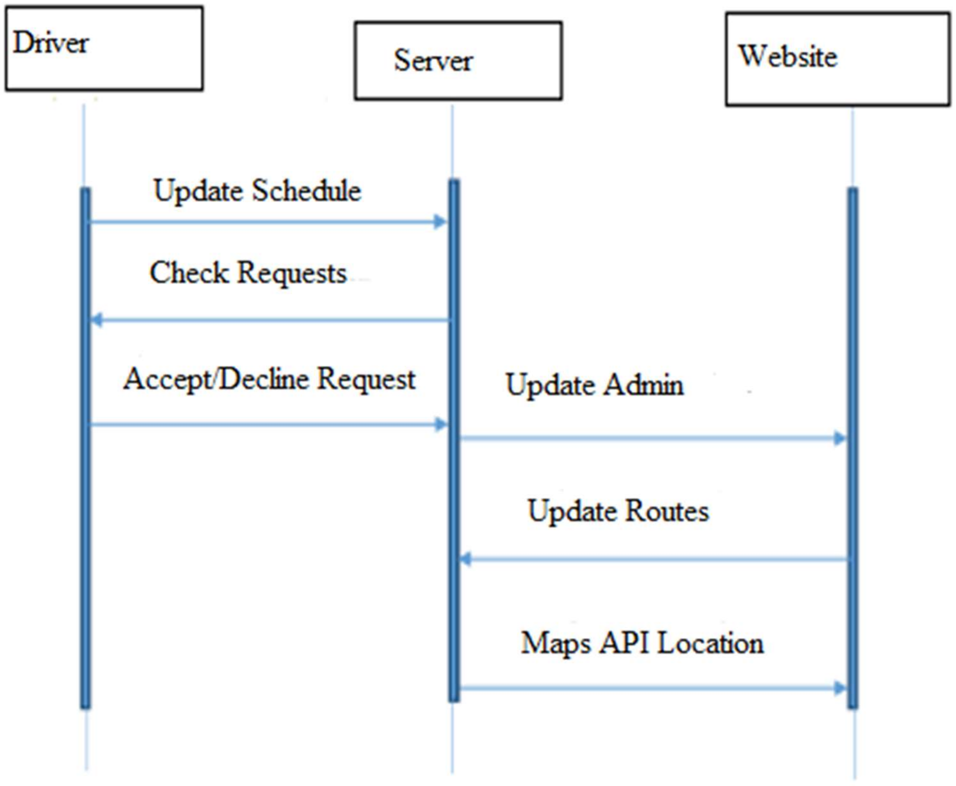
DFD Level 2:



User-Seuquence Diagram




Rider
Server
Website
Check Schedule
Ride Route
Send Request
Update Admin
Accept/deny Request
Maps API Location


Driver
Server
Website
Update Schedule
Check Requests
Accept/Decline Request
Update Admin
Update Routes
Maps API Location

4. Nonfunctional Requirements

4.1 Performance Requirements

The application shall have specifications other than technical requirements. There are performance, protection, safety and efficiency criteria. Complying with the non-functional specifications is necessary to get the project up to the mark. These requirements are to be fulfilled so that the project is well explained and the design feature is suitable according to it. This will help the project elevate and be more descriptive.

4.2 Safety Requirements

The application shall have the privacy of the data and anonymity of the driver using it. It is, therefore, not vulnerable to any leakage of information. It allows the user to have personal information. The application will also keep the user data private, and only authorized people can see it saving the integrity of the application. The information will be kept private so that the leakage of it is prevented, as the information is essential.

Maintainability:

This system can be maintained under low maintenance circumstances. Not much haste is needed to maintain its

Portability:

This system does not require major portability as this will be accessed through webpages on any browser.

Reusability:

The system can be configured to store data from the user, admin, and the driver so that they can login later for better user experience. It will allow them to stay connected with the app and share experience of using it.

5. Other Requirements:

Database Requirements: We need a reliable and scalable database system to store and retrieve data efficiently. It should support the required data structures and keep our data secure.

External Interface Requirements: Our web application should seamlessly work with external hardware, software, or communication systems if needed. This might involve connecting with devices, APIs, or other services.

Internationalization Requirements: As it is a Pakistan and FCCU based webapp so we don't need as such Internationalization Requirements. Because it would be only used as a utility for students of FCCU.

Legal Requirements: Our web application must follow all relevant laws and regulations. This includes privacy laws, data protection rules, and any specific requirements for our industry. We take user data seriously and will obtain necessary consents if needed. Keeping in mind that it is for FCCU, so we would be following the terms and conditions and legalities of using as a student welfare webapp.

Reuse Objectives: We aim to make our code reusable to improve efficiency and maintainability. This means organizing our code in modules, following best practices, and documenting reusable code segments for future use.

These requirements help ensure the success of our project by addressing important factors such as database management, external system integration, user localization, legal compliance, and code reusability.

6. Revised Project Plan

7. References

Google carpool application: https://mapsplatform.google.com/pricing/

Advantages of carpooling: https://www.sciencedirect.com/science/article/pii/S2352146517303721

Polunipool: An application for university students:
https://www.sciencedirect.com/science/article/pii/S187704281101442X

Solution for university Students: https://spaces4learning.com/articles/2020/01/17/ridesharing-for-campus.aspx?m=1

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: IV & V Report

(Independent verification & validation)

IV & V Resource

Name Signature

S#	Defect Description	Origin Stage	Status	Fix Time	
				Hours	Minutes
1					
2					
3					
…					

Table 2: List of non-trivial defects