

SAVITRIBAI PHULE PUNE UNIVERSITY



A PRELIMINARY PROJECT REPORT ON
**PEER TO PEER CARPOOLING USING
BLOCKCHAIN TECHNOLOGY**

SUBMITTED TOWARDS THE
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE
DEGREE

**BACHELOR OF ENGINEERING
(Computer Engineering)**

BY

Pratik Bagad
Akash Maher
Vedant Pahune
Rachna Adhav

Exam No: 72025576H
Exam No: 72025762L
Exam No: 72025810D
Exam No: 72025853H

Under The Guidance of
Dr. R. A. Khan



DEPARTMENT OF COMPUTER ENGINEERING
Modern Education Society's College of Engineering, Pune-411 001
2022-23

Modern Education Society's College of Engineering, Pune 01



**DEPARTMENT OF COMPUTER ENGINEERING
CERTIFICATE**

This is to certify that the Project Entitled

Peer to Peer Carpooling using Blockchain Technology

Submitted by

Pratik Bagad

Exam No: 72025576H

Akash Maher

Exam No: 72025762L

Vedant Pahunne

Exam No: 72025810D

Rachna Adhav

Exam No: 72025853H

is a bonafide work carried out by Students under the supervision of Dr. R. A. Khan and it is submitted towards the partial fulfillment of the requirement of Savitribai Phule Pune University, Pune for the award of the degree of Bachelor of Engineering (Computer Engineering). Project.

Dr. R. A. Khan
Internal Guide

Prof. Dr. N. F. Sheikh
Head Of Department
Dept of Computer Engineering

External Examiner

Dr.M.P.Dale
Principal
MES College of Engineering, Pune-01

Date:

Place: Pune

A Project Title

Peer to Peer Carpooling using Blockchain Technology

Is successfully completed by

Pratik Bagad

Exam No: 72025576H

Akash Maher

Exam No: 72025762L

Vedant Pahune

Exam No: 72025810D

Rachna Adhav

Exam No: 72025853H

at

DEPARTMENT OF COMPUTER ENGINEERING

Modern Education Society's College of Engineering, Pune

SAVITRIBAI PHULE PUNE UNIVERSITY,PUNE

ACADEMIC YEAR 2022-2023

Dr. R. A. Khan

Prof. Dr. N. F. Shaikh

Internal Guide

H.O.D

Dept. of Computer Engg.

Dept. of Computer Engg.

ABSTRACT

Peer to peer carpooling using blockchain technology is an application that will be booking cabs without any help of third-party applications which are being used currently. It will increase security as well as transparency between drivers and passengers.

The carpooling service enables drivers to share rides with passengers, adding to the alluring advantages of traffic congestion. There are many applications available, such as ola and uber, that offer carpooling services these days. However, as this carpooling system relies on a single database server, it is frequently vulnerable to hacker attacks increasing security issues. Utilizing blockchain technology is the best solution to these problems. Blockchain is an unchangeable, decentralized technology that cannot be changed. The primary focus of the application we are proposing is carpooling application relating to smart contracts and blockchain innovation. We are going to use Ethereum Blockchain to implement this project. Without having to drive a car, the traveler will be able to reach their destination quickly. The client may use popular crypto tokens like ethers to establish payments to drivers.

In this report we are presenting the system architecture of how the system or application is going to work and what will be the benefits of this system to the users.

Keywords:Peer to peer, Carpooling, Blockchain, Ethereum

ACKNOWLEDGEMENTS

It gives us great pleasure in presenting the preliminary project report on '**Peer to Peer Carpooling Using Blockchain Technology**'.

I would like to take this opportunity to thank my internal guide **Prof. R. A. Khan** for giving me all the help and guidance I needed. I am really grateful to them for their kind support. Their valuable suggestions were very helpful.

I am also grateful to **Dr. N. F. Shaikh**, Head of Computer Engineering Department, Modern Education Society's College of Engineering, Pune for his indispensable support, suggestions.

In the end our special thanks to **Dr. M. P. Dale**, Principal, MES College of Engineering, Pune for providing various resources such as a laboratory with all needed software platforms, continuous Internet connection, for Our Project.

Pratik Bagad
Akash Maher
Vedant Pahune
Rachna Adhav
(B.E. Computer Engg.)

Contents

1	Synopsis	2
1.1	Project Title	3
1.2	Internal Guide	3
1.3	Project Option	3
1.4	Technical Keywords	3
1.5	Problem Statement	3
1.6	Abstract	4
1.7	Goals and objectives	4
1.8	Relevant mathematics associated with the Project	5
1.9	Names of Conferences / Journals where papers can be published	5
2	Technical Keywords	6
2.1	Area of Project	7
2.2	Technical Keywords	7
3	Introduction	8
3.1	Project Idea	9
3.2	Motivation	9
3.3	Literature Survey	9
4	Problem Definition and scope	12
4.1	Problem Statement	13
4.1.1	Goals and objectives	13
4.1.2	Statement of scope	13
4.2	Major Constraints	15
4.3	Methodologies of Problem solving and efficiency issues	16
4.4	Outcome	17
4.5	Applications	18
4.6	Hardware Resources Required	18
4.7	Software Resources Required	19

5	Project Plan	20
5.1	Project Estimates	21
5.1.1	Reconciled Estimates	21
5.1.2	Project Resources	21
5.2	Risk Management w.r.t. NP Hard analysis	22
5.2.1	Risk Identification	22
5.2.2	Risk Analysis	24
5.2.3	Overview of Risk Mitigation, Monitoring, Management	24
5.3	Project Schedule	28
5.3.1	Project task set	28
5.3.2	Task network	30
5.3.3	Timeline Chart	31
5.4	TEAM ORGANIZATION:	31
5.4.1	Team structure:	31
5.4.2	Management reporting and communication:	33
6	Software Requirements Specification	34
6.1	Introduction	35
6.1.1	Purpose and Scope of Document	35
6.1.2	Overview of responsibilities of Developer	35
6.2	Usage Scenario	36
6.2.1	User profiles	37
6.2.2	Use-cases	37
6.2.3	Use Case View	37
6.2.4	Use Case Diagram	39
6.3	Data Model and Description	39
6.3.1	Data Description	39
6.3.2	Data objects and Relationships	40
6.4	Functional Model and Description	40
6.4.1	User Registration and Authentication:	40
6.4.2	Location and GPS Tracking:	40
6.4.3	Ride Request and Matching:	40
6.4.4	Driver Acceptance and Routing:	41
6.4.5	Payment Integration	41
6.4.6	Smart Contract Communication:	41
6.5	Data Flow Diagrams	41
6.5.1	Data Flow diagram Level 0(Level 0 DFD)	42
6.5.2	Data Flow Diagram Level 1(Level 1 DFD)	43
6.5.3	Data Flow Diagram Level 2 (Level 2 DFD)	44
6.5.4	Activity diagram	45
6.6	Nonfunctional Requirements	46

6.6.1	Performance Requirements	46
6.6.2	Safety Requirements	46
6.6.3	Security Requirements	46
6.6.4	Software Quality Attributes	46
6.6.5	State Diagram:	47
6.6.6	Design Constraints	48
6.6.7	Software Interface Description	49
7	Detailed Design Document using Appendix A and B	50
7.1	Introduction	51
7.2	Architectural Design	51
7.2.1	Deployment diagram	52
7.3	Data design (using Appendices A and B)	52
7.3.1	Internal software data structure	52
7.3.2	Global data structure	53
7.3.3	Temporary data structure	53
7.3.4	Database description	53
7.4	Component Design	53
7.4.1	Class Diagram	54
7.4.2	Sequence diagram	55
7.4.3	Component diagram	56
8	Project Implementation	57
8.1	Introduction	58
8.1.1	Overview	58
8.2	Tools and Technologies Used	58
8.3	Methodologies/Algorithm Details	59
8.3.1	Algorithm: User Location Retrieval	60
8.3.2	Algorithm: Cryptocurrency Payment Processing	60
8.4	Verification and Validation for Acceptance	61
9	Results	62
9.1	Screen shots	63
9.1.1	Passenger Section	63
9.2	Driver	67
9.3	Outputs	70
9.3.1	Ride Booking by passenger	70
9.3.2	Active Ride Details	71
9.3.3	Payment approval from passenger	72
9.3.4	Cancel booking	73
9.3.5	Transactions	74

10 Deployment and Maintenance	77
10.1 Installation and un-installation	78
10.1.1 Frontend Deployment	78
10.1.2 Smart Contract Deployment	78
10.1.3 Maintenance	79
10.2 User Help	79
11 Conclusion & Future Scope	81
11.1 Conclusion	82
11.2 Future Scope	82
Annexure A Referances	83
Annexure B Laboratory assignments on Project Analysis of Al-	
gorithmic Design	85
Annexure C Laboratory assignments on Project Quality and	
Reliability Testing of Project Design	87
Annexure D Project Planner	88
Annexure E Reviewers Comments of Paper Submitted	89
E.1 Reviewers Comments of Paper Submitted	90
Annexure F Plagiarism Report	91
Annexure G Term-II Project Laboratory Assignments	92
Annexure H Information of Project Group Members	93

List of Figures

5.1	Timeline Chart (Phase 1)	31
5.2	Timeline Chart (Phase 2)	31
6.1	Use Case Diagram	39
6.2	Level 0 Data Flow diagram	42
6.3	Level 1 Data Flow diagram	43
6.4	Level 2 Data Flow diagram	44
6.5	Activity Diagram	45
6.6	State Machine Diagram	47
7.1	System Architecture of our System	51
7.2	Deployment Diagram	52
7.3	Class Diagram	54
7.4	Sequence Diagram	55
7.5	Component Diagram	56
9.1	Home page	63
9.2	Login page	64
9.3	User Activity	65
9.4	Account Info	66
9.5	View Available Rides	67
9.6	Account Info	68
9.7	Dashboard	69
9.8	Ride booking page	70
9.9	Active Ride Details	71
9.10	Payment approval from passenger	72
9.11	Cancel booking	73
9.12	Ride booking	74
9.13	Payment withdrawal	75
9.14	Transaction History on Smart Contract	76
F.1	Plagiarism report	91

List of Abbreviations

IDE	Integrated Development Environment
EVM	Ethereum Virtual Machine
UML	Unified Modeling Language
IJERT	International Journal of Engineering Research and Technology
IEEE	Institute of Electrical and Electronics Engineers
SDLC	Software Development Life Cycle
HDD	Hard Disk Drive
SSD	Solid State Drive
NVM	Node Version Manager

Chapter 1

Synopsis

1.1 Project Title

Peer to Peer Carpooling using Blockchain Technology

1.2 Internal Guide

Dr. R. A. Khan

1.3 Project Option

Internal Project

1.4 Technical Keywords

1. Blockchain Technology
2. Carpooling
3. Peer to peer
4. Ethereum Blockchain
5. Smart Contract
6. Crypto Currency
7. Metamask

1.5 Problem Statement

In this project, we are going to make a website based application which will be more secure and trustable compared to the applications which people are currently using for carpooling. Using blockchain technology, third party applications will be removed from the process and driver and passenger will be connected to each other with the help of smart contract which will increase trust and transparency between them.

1.6 Abstract

We have developed a carpooling website that utilizes GPS, user location, Google Maps, and Metamask integration for crypto payments. The website is built using Next.js, with the smart contract developed and deployed using Solidity. Our frontend communicates with the smart contract through RPC Etherjs, and we have utilized Tailwind CSS for styling. This blockchain-based peer-to-peer carpooling website enhances security and transparency between drivers and passengers. By eliminating the need for third-party applications, it provides a secure and transparent booking process. Users can conveniently make payments using cryptocurrencies like Ethereum, leveraging the benefits of blockchain technology. This report presents the system architecture and outlines the advantages of this platform to users.

1.7 Goals and objectives

1. Seamless and reliable transportation service: We want to connect users with nearby drivers in a quick and efficient manner, ensuring that they can find convenient rides whenever they need them.
2. Transparency and trust: By utilizing GPS and user location tracking, we aim to provide real-time updates on the driver's location. This feature will enable users to track their rides and enhance transparency throughout the entire journey.
3. Secure and cost-effective payments: Integrating Metamask for cryptocurrency payments will allow for secure and efficient transactions. By reducing reliance on traditional payment methods and minimizing transaction fees, we aim to provide users with a convenient and cost-effective payment solution.
4. Improved user experience: Our goal is to create an intuitive user interface that makes it easy for users to request rides, view driver details, and provide feedback. By focusing on enhancing the overall user experience, we aim to differentiate ourselves from existing transportation platforms.
5. Resource optimization: By analyzing demand patterns and utilizing GPS data, we intend to optimize the allocation of drivers. This approach will reduce wait times for users and maximize the overall efficiency of our transportation network.

1.8 Relevant mathematics associated with the Project

System Description:

- Output:
- Input:
- Identify data structures, classes, divide and conquer strategies to exploit distributed/parallel/concurrent processing, constraints.
- Functions : Identify Objects, Morphisms, Overloading in functions, Functional relations
- Mathematical formulation if possible
- Success Conditions:
- Failure Conditions:

1.9 Names of Conferences / Journals where papers can be published

- IEEE/ACM Conference/Journal 1.
- Conferences/workshops in IITs.
- Central Universities or SPPU Conferences.
- IEEE/ACM Conference/Journal 2.

Chapter 2

Technical Keywords

2.1 Area of Project

- Blockchain Technology
- Web Developement

2.2 Technical Keywords

1. Blockchain Technology
2. Carpooling
3. Peer to peer
4. Ethereum Blockchain
5. Smart Contract
6. Crypto Currency
7. Metamask

Chapter 3

Introduction

3.1 Project Idea

Our project idea revolves around developing a transportation app that leverages technologies such as GPS, user location tracking, Google Maps integration, and Metamask integration for cryptocurrency payments. The frontend of our app is built using Next.js, providing a seamless and responsive user interface. To communicate with the smart contract, we'll be utilizing RPC through Ether.js. This decentralized approach aims to address the shortcomings of traditional transportation systems by providing a transparent, efficient, and secure ride-hailing experience.

3.2 Motivation

Due to increase in pollution and huge traffic issues, now people are preferably using carpooling services. In present situation there are many carpooling services available but all of them are working on centralized server which reduces the privacy of the user as well as increasing security threat to the user's data. Many times the fare changes during the ride means there is huge difference between the fare at the time of booking and the fare available after the ride.

To avoid the problems we are facing in centralized system, we are developing a decentralized application which will be more secure. As we are using blockchain ethereum in our system, the security problem will be automatically solved. And there will be no fare change throughout the ride. Also before starting the ride passenger have to pay the amount to the contract hence there will be payment security available for driver.

3.3 Literature Survey

[2]David Schuff and Robert St Louis explains in "Centralization vs. Decentralization of Application Software" Placing an application files in several central locations gives IT significant control over software configuration, but can degrade network performance and lead to user dissatisfaction. Distributing application files to individual PCs maximizes network performance but makes enforcing configuration standards and maintaining control much more difficult. It is clear that blockchain is an emerging, secure, robust and unmanipulated technology that will contribute to the development of more modern, robust and secure systems. A decentralized platform, such as blockchain technology, is considered one of the best solutions due to the immutable nature of smart contracts.

[3]A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak in "Blockchain: A distributed solution to automotive security and privacy" explain that interconnected intelligent vehicles provide a complex array of services that benefit vehicle owners, transportation regulators, automakers, and service providers. other. This has the potential to expose smart vehicles to a range of security and privacy threats such as location tracking or remote vehicle hijacking. The use of blockchain in car sharing has been highlighted as one of the best possible future uses of blockchain.

[4]T. Report explains in "Filecoin: A cryptocurrency operated file storage network" that unlike Bitcoin's computation-only proof-of-work, Filecoin's proof-of-work function includes a proof-of-retrievability component, which requires nodes to prove they store a particular file. The Filecoin network forms a fully distributed file storage system, whose nodes are incentivized to store as much data as possible across the entire network. The currency is awarded for storing files, and is transferred in transactions, as in Bitcoin. In addition to cryptocurrencies, blockchain technology is essential in enabling many applications such as supply chain management, fintech, healthcare, data sharing and objects on the internet (IoT), etc It transformed the sharing economy with applications like Filecoin.

[5]Yong Yuan, Fei-Yue Wang[4] explains in "Towards blockchain-based intelligent transportation systems" Blockchain can be used to establish a secure, trusted and decentralized autonomous ITS ecosystem, creating better use of ITS infrastructure and resources (ITS Transaction System) intelligence), especially effective for community service delivery technology. Blockchain allows drivers to offer carpooling services without third party intervention. Hence increasing transparency and trust.

[6]Surbhi Dhar, Sandra Arun, Vivek Dubey and Nilesh Kulal explains in "App for Ride Sharing" as A smart contract that allows stakeholders to use a blockchain-powered peer-to-peer car rental service for two parties to actually participate based on predefined essential requirements. An app will let the user know if vehicles are available for carpooling in the desired route and will allow them to connect to it. In decentralized carpooling applications, riders can verify the operations of a ride-hailing company through the

blockchain's ability to create accountability. As a result, it always offers accurate pricing and a system that enhances transparency.

[7]M. Baza, N. Lasla, M. M. Mahmoud ,G. Srivastava, M. Abdallah explains in "B-Ride: Ride Sharing with Privacy-preservation, Trust and Fair Payment atop Public Blockchain" that the majority of current rideshare services rely on a central third party to organize the service, which exposes them to a single point of failure and privacy issues from inside attackers and outside. Additionally, they are vulnerable to Sybil and Distributed Denial of Service (DDoS) attacks initiated by malicious users and external attackers. In addition, high service fees are paid to the ride-sharing applications. Allows drivers to provide carpooling services without the involvement of a third party.

[8]Panchalika Pal explain in "BlockV: A Blockchain Enabled Peer-Peer Ride Sharing Service" that the BlockV architecture discussed in this paper satisfies the privacy requirement as well as the two fairness goals. The reputation score assigned to each user makes the system more user-focused because it shows how successfully the user has operated on the whole. The system also encourages users to behave in an anonymous manner, allowing them to transact with various addresses while yet maintaining their reputation score. Allows drivers to provide carpooling services without the involvement of a third party.

Chapter 4

Problem Definition and scope

4.1 Problem Statement

In this project, we are going to make a website based application which will be more secure and trustable compared to the applications which people are currently using for carpooling. Using blockchain technology, third party applications will be removed from the process and driver and passenger will be connected to each other with the help of smart contract which will increase trust and transparency between them.

4.1.1 Goals and objectives

1. Seamless and reliable transportation service: We want to connect users with nearby drivers in a quick and efficient manner, ensuring that they can find convenient rides whenever they need them.
2. Transparency and trust: By utilizing GPS and user location tracking, we aim to provide real-time updates on the driver's location. This feature will enable users to track their rides and enhance transparency throughout the entire journey.
3. Secure and cost-effective payments: Integrating Metamask for cryptocurrency payments will allow for secure and efficient transactions. By reducing reliance on traditional payment methods and minimizing transaction fees, we aim to provide users with a convenient and cost-effective payment solution.
4. Improved user experience: Our goal is to create an intuitive user interface that makes it easy for users to request rides, view driver details, and provide feedback. By focusing on enhancing the overall user experience, we aim to differentiate ourselves from existing transportation platforms.
5. Resource optimization: By analyzing demand patterns and utilizing GPS data, we intend to optimize the allocation of drivers. This approach will reduce wait times for users and maximize the overall efficiency of our transportation network.

4.1.2 Statement of scope

- **Description of the Software:**

The software is a basic transportation application designed to facilitate ride bookings and payments using cryptocurrency. It utilizes GPS technology, user location data, Google Maps integration, and incorporates the Metamask wallet for secure crypto transactions. The application is built using Next.js, a JavaScript framework for developing web applications.

In addition to the mentioned features, the software also includes the development and deployment of smart contracts using Solidity, a programming language specifically designed for creating contracts on blockchain platforms. These smart contracts provide additional functionality and security for the application's payment system, ensuring transparent and reliable cryptocurrency transactions.

- **Size of Input:**

The application can handle various sizes of input, including user requests for ride bookings, GPS coordinates for determining location, payment details involving cryptocurrency transactions, and input related to smart contract interactions.

- **Bounds on Input:**

The software operates within the bounds of the available GPS coverage and the supported cryptocurrency networks. It relies on accurate and valid user-provided inputs, such as pickup and drop-off locations, payment details, user authentication, and input required for smart contract interactions.

- **Input Validation:**

Input validation is performed to ensure the integrity and accuracy of the data provided by users. This includes validating GPS coordinates, verifying the authenticity of payment transactions through Metamask integration, validating user input for completeness and correctness, and validating input for smart contract interactions to maintain the integrity of the blockchain-based payment system.

- **Input Dependency:**

The software relies on inputs such as user location data, pickup and drop-off locations, preferred payment methods, user authentication information, and input required for smart contract interactions to provide the desired ride booking services. It also depends on the availability and functionality of the GPS system, Metamask wallet integration, and the blockchain network for smart contract execution.

- **I/O State Flow:**

The application follows an input/output state flow, where users input their desired ride details, such as pickup and drop-off locations, preferred payment method, authentication information, and input required for smart contract interactions. The software processes these inputs, retrieves relevant data from the GPS system, Metamask wallet, and blockchain network, and outputs ride confirmation, real-time ride tracking information, payment transaction status, and relevant information related to smart contract interactions.

- **Major Inputs:**

1. User ride request details (pickup and drop-off locations, preferred payment method)
2. User authentication information
3. GPS location data
4. Metamask wallet integration for cryptocurrency payment transactions
5. Input required for smart contract interactions (e.g., contract addresses, method calls, input parameters)

- **Outputs:**

1. Ride confirmation
2. Real-time ride tracking information
3. Payment transaction status
4. Output related to smart contract interactions (e.g., transaction receipts, contract state changes)

4.2 Major Constraints

- **Privacy and Security:** Protect user data and cryptocurrency transactions with encryption and secure protocols.
- **Compatibility:** Support various devices and browsers for a consistent user experience.
- **GPS and Network Connectivity:** Handle GPS and network disruptions gracefully.

- **Cryptocurrency Network Limitations:** Consider fees, confirmation times, and scalability of the chosen cryptocurrency network.
- **Scalability and Performance:** Scale to handle increased users and transactions efficiently.
- **User Experience and Usability:** Provide an intuitive interface and clear instructions for easy use.
- **Maintenance and Updates:** Regularly update software for security and feature enhancements.

4.3 Methodologies of Problem solving and efficiency issues

Problem-Solving Methodologies:

- **Define the problem:** Clearly identify the problem you are trying to solve. In this case, it could be providing a reliable ride-hailing service using GPS, user location, and Google Maps integration.
- **Break it down:** Divide the problem into smaller, manageable components. For example, handling user authentication and payment processing.
- **Design and plan:** Create a design and architecture that addresses the problem and its components. Determine the technologies, frameworks, and libraries required for implementation.
- **Implement and iterate:** Begin development by coding the app according to the design. Break the implementation into incremental iterations, allowing for testing, feedback, and improvements along the way.
- **Test and validate:** Conduct rigorous testing to ensure the app functions correctly and meets the defined requirements. This includes unit tests, integration tests, and end-to-end testing.
- **Deploy and maintain:** Deploy the app to production and continue monitoring its performance. Maintain and update the app to address bugs and add new features.

Efficiency Issues:

- Performance optimization: Identify potential bottlenecks in the app and optimize them. This could involve optimizing code, reducing database queries, implementing caching mechanisms, and utilizing background processing where applicable.
- Scalability: Plan for scalability from the beginning to accommodate increasing user demand. This may involve horizontal scaling (adding more servers) or vertical scaling (upgrading server resources).
- Network efficiency: Minimize network requests and data transfers to reduce latency. Reduce the number of API calls by batching multiple requests into a single call. Send only necessary data fields and compress payloads to minimize data transferred between the app and smart contract. Utilize lightweight and easily parseable formats like JSON.
- Error handling and recovery: Implement proper error handling mechanisms to handle unexpected situations gracefully. Provide informative error messages and handle exceptions to prevent crashes and data loss.
- Security and privacy: Protect user data, securely handle transactions (Metamask integration), and implement necessary authentication mechanisms.

4.4 Outcome

- Developed and deployed a feature-rich app with GPS functionality, user location tracking, Google Maps integration, and Metamask integration for cryptocurrency payments.
- Created a website using Next.js, a React framework known for its server-side rendering capabilities, to provide a seamless user experience.
- Implemented a smart contract using Solidity, enabling communication between the app's frontend and the blockchain network.
- Leveraged RPC (Remote Procedure Call) to facilitate secure and decentralized transactions through the smart contract.
- Enabled users to make payments using cryptocurrencies, thanks to the Metamask integration.

- Achieved the goal of creating a functional app that offers location-based services while leveraging blockchain technology for secure and efficient transactions.

4.5 Applications

1. Peer-to-Peer Ride Sharing: Your app can also facilitate peer-to-peer ride sharing, allowing users to offer rides to others heading in the same direction. This feature promotes carpooling and helps reduce traffic congestion and carbon emissions.
2. Ride-hailing Service: The primary application of your app is to provide a ride-hailing service, similar to Uber. Users can request rides from their current location and request a ride. This use case caters to individuals who need transportation on-demand.

4.6 Hardware Resources Required

Table 4.1: Minimum Requirements for Different Parameters

Sr. No.	Parameter	Minimum Requirement	Justification
1	CPU Speed	2.5 GHz	Faster CPU speeds ensure smoother multitasking.
2	RAM	8 GB	Sufficient RAM allows for efficient operations.
3	ROM	256 GB	Sizable ROM provides ample storage capacity.

4.7 Software Resources Required

Table 4.2: Software Resources Required

Sr. No.	Parameter	Minimum Requirement	Justification
1	Operating System	Windows 10	Stable and widely supported platform.
2	IDE	Visual Studio Code, Remix	Popular and feature-rich development environments.
3	Programming Language	TypeScript, JavaScript, Solidity	Widely used languages for web and blockchain development.

Chapter 5

Project Plan

5.1 Project Estimates

Use Waterfall model and associated streams derived from assignments 1,2, 3, 4 and 5(Annex A and B) for estimation.

5.1.1 Reconciled Estimates

Cost Estimates

1. Development Cost: \$20,000 to \$60,000 or more
2. Server Costs: \$50 to \$300 per month
3. Payment Integration Costs: Consult with Metamask or the chosen payment gateway for accurate estimates.
4. Miscellaneous Costs: Domain registration, SSL certificate, third-party API usage fees, etc. (costs may vary).

Time Estimates

1. Website Development: 1-3 months
2. Smart Contract Development: Few weeks to several months
3. Frontend Integration: Few days to a couple of weeks
4. QA and Testing: Few weeks to a couple of months

It is estimated that project will be completed in around 6 months.

5.1.2 Project Resources

1. GPS Integration: We have incorporated GPS functionality into our application, allowing users to track their location in real-time.
2. User Location Tracking: Our app includes the capability to track the user's location, enabling efficient routing and navigation.
3. Google Maps Integration: We have seamlessly integrated Google Maps into our application, providing users with accurate and interactive maps for navigation purposes.

4. MetaMask Integration for Crypto Payments: To facilitate secure and convenient payments, we have integrated MetaMask, allowing users to make transactions using cryptocurrencies.
5. Website Development with Next.js: Our website has been developed using Next.js, a popular JavaScript framework for building server-rendered React applications.
6. Smart Contract Development and Deployment: We have created a smart contract using Solidity, a programming language specific to Ethereum. This contract has been successfully deployed on the blockchain.
7. RPC Communication with Ether.js: The frontend of our application communicates with the smart contract using the Ether.js library, utilizing Remote Procedure Calls (RPCs) to interact with and retrieve data from the contract.

By combining these resources, we have created a comprehensive application that leverages GPS, user location tracking, Google Maps, MetaMask integration for crypto payments, and a website built with Next.js. The smart contract, developed using Solidity, enables secure and decentralized transactions, while the frontend communicates with it using Ether.js for seamless interaction with the blockchain.

5.2 Risk Management w.r.t. NP Hard analysis

This section discusses Project risks and the approach to managing them.

5.2.1 Risk Identification

1. Have top software and customer managers formally committed to support the project?
 - Risk: Lack of commitment from top managers may result in insufficient resources and support for the project.
2. Are end-users enthusiastically committed to the project and the system/product to be built?
 - Risk: Lack of enthusiasm from end-users may lead to low user adoption and dissatisfaction with the app.

3. Are requirements fully understood by the software engineering team and its customers?
 - Risk: Inadequate understanding of requirements may result in delivering an app that does not meet user expectations.
4. Have customers been involved fully in the definition of requirements?
 - Risk: Insufficient customer involvement may lead to incomplete or inaccurate requirements, resulting in a mismatch between the app's functionality and user needs.
5. Do end-users have realistic expectations?
 - Risk: Unrealistic expectations may result in dissatisfaction with the app's features, performance, or usability.
6. Does the software engineering team have the right mix of skills?
 - Risk: Insufficient or mismatched skills within the team may lead to difficulties in implementing complex features or addressing technical challenges.
7. Are project requirements stable?
 - Risk: Frequent changes or unstable requirements may cause delays, rework, and uncertainty in delivering a stable and functional app.
8. Is the number of people on the project team adequate to do the job?
 - Risk: Insufficient team size may lead to increased workload, longer development timelines, and compromised quality.
9. Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?
 - Risk: Lack of agreement among stakeholders may result in conflicting priorities, scope creep, and difficulty in meeting customer expectations.

5.2.2 Risk Analysis

The risks for the Project can be analyzed within the constraints of time and quality

ID	Risk Description	Probability	Impact		
			Schedule	Quality	Overall
1	Data Breach	Low	Low	Low	Low
2	Poor User Adoption	Medium	Medium	Medium	Medium
3	Technical Issues	Medium	High	High	High
4	Regulatory Compliance	Low	High	High	High
5	Competitive Pressure	Medium	Medium	Medium	Medium
6	Market Volatility	Low	Medium	Medium	Medium

Table 5.1: Risk Table

5.2.3 Overview of Risk Mitigation, Monitoring, Management

Following are the details for each risk.

Risk ID	1
Risk Description	Data Breach
Category	Security/Privacy
Source	Potential vulnerabilities in data storage or communication
Probability	High
Impact	High
Response	Implement security measures, audits, and encryption
Strategy	Data protection policies, secure coding, training
Risk Status	Open (requires ongoing monitoring and mitigation efforts)

Risk ID	2
Risk Description	Poor User Adoption
Category	Market/User
Source	Lack of interest, competition, or inadequate marketing
Probability	Medium
Impact	Medium
Response	Market research, user feedback, and marketing strategy
Strategy	User satisfaction, targeted campaigns, partnerships
Risk Status	Open (requires continuous monitoring and adaptation)

Risk ID	3
Risk Description	Technical Issues
Category	Technical
Source	Software bugs, downtime, compatibility, integration issues
Probability	Medium
Impact	High
Response	Testing, redundancy, communication with integrations
Strategy	Thorough testing, proactive issue resolution
Risk Status	Open (requires ongoing monitoring and prompt resolution)

Risk ID	4
Risk Description	Regulatory Compliance
Category	Legal/Compliance
Source	Non-compliance with regulations or data protection
Probability	Low
Impact	High
Response	Stay updated, establish policies, conduct audits
Strategy	Collaboration with legal advisors, data protection measures
Risk Status	Open (requires regular compliance assessment and adherence)

Risk ID	5
Risk Description	Competitive Pressure
Category	Market
Source	Increased competition, market forces
Probability	Medium
Impact	Medium
Response	Monitor market, develop unique value propositions
Strategy	Competitor analysis, innovation, customer loyalty
Risk Status	Open (requires ongoing market monitoring and analysis)

Risk ID	6
Risk Description	Market Volatility
Category	Market
Source	Cryptocurrency fluctuations, economic conditions
Probability	Low
Impact	Medium
Response	Diversify payment options, monitor market trends
Strategy	Financial collaboration, flexible payment options
Risk Status	Open (requires ongoing monitoring and adaptation)

5.3 Project Schedule

5.3.1 Project task set

Major Tasks in the Project stages are:

1. Planning
 - (a) Define project scope and objectives.
 - (b) Identify key features and functionalities of the app.
2. Design and Prototyping
 - (a) Create wireframes and UI/UX designs.
 - (b) Develop a prototype to visualize the app's user flow.
 - (c) Iterate on the design based on feedback and usability testing.
3. Backend and Smart Contract Development
 - (a) Set up the development environment.
 - (b) Implement server-side logic and APIs.
 - (c) Develop and deploy the smart contract using Solidity.
 - (d) Integrate the smart contract with the backend.
4. Frontend Development
 - (a) Set up the frontend development environment.
 - (b) Implement user interfaces based on the approved designs.
 - (c) Integrate user location and Google Maps functionality.
 - (d) Connect the frontend with the backend and smart contract using Ether.js.
5. Payment Integration and Security
 - (a) Integrate MetaMask for cryptocurrency payments.
 - (b) Implement secure payment processing and verification.
 - (c) Conduct thorough testing to ensure payment functionality and security measures are in place.
6. Testing and Quality Assurance

- (a) Perform comprehensive testing of the application.
- (b) Conduct functional, performance, and security testing.
- (c) Identify and fix any bugs or issues.
- (d) Ensure the application meets the desired quality standards.

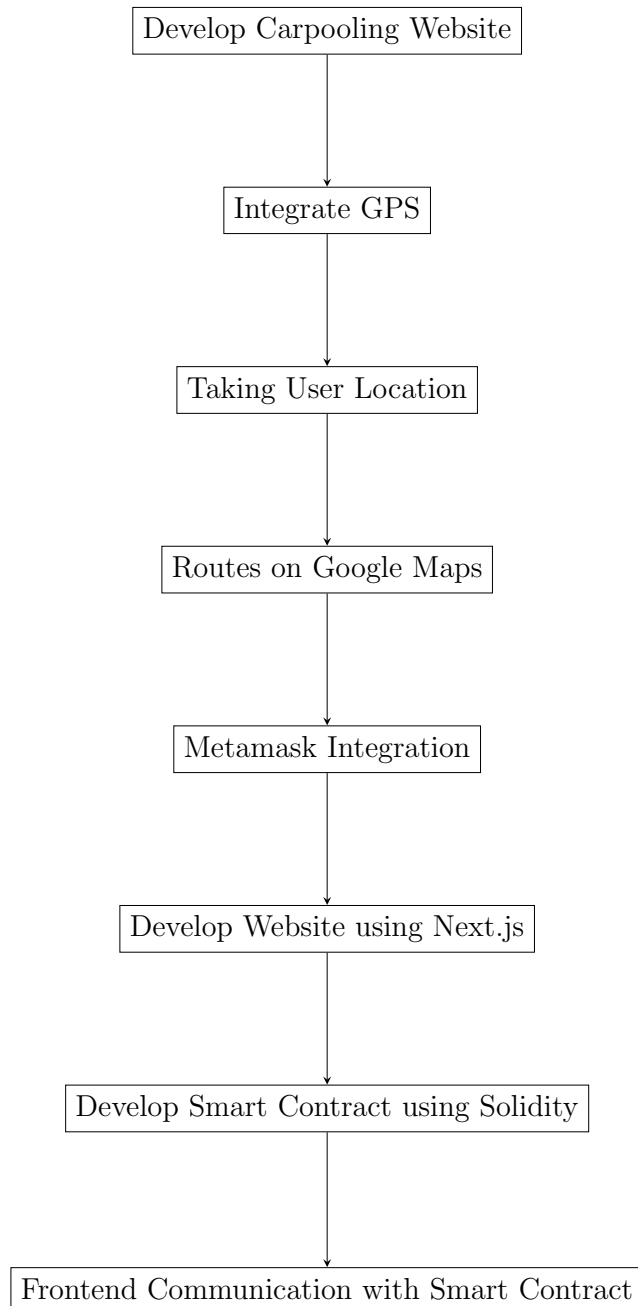
7. Deployment and Launch

- (a) Prepare the application for deployment.
- (b) Set up hosting and server configurations.
- (c) Deploy the application to a production environment.
- (d) Conduct final checks and ensure the app is ready for public use.

8. Post-launch Support and Maintenance

- (a) Monitor the application's performance and user feedback.
- (b) Address any post-launch issues or bug reports.
- (c) Provide ongoing maintenance and support for the application.
- (d) Plan for future enhancements and updates based on user feedback.

5.3.2 Task network



5.3.3 Timeline Chart

A project timeline chart is presented. This may include a time line for the entire project. Above points should be covered in Project Planner as Annex C and you can mention here Please refer Annex C for the planner

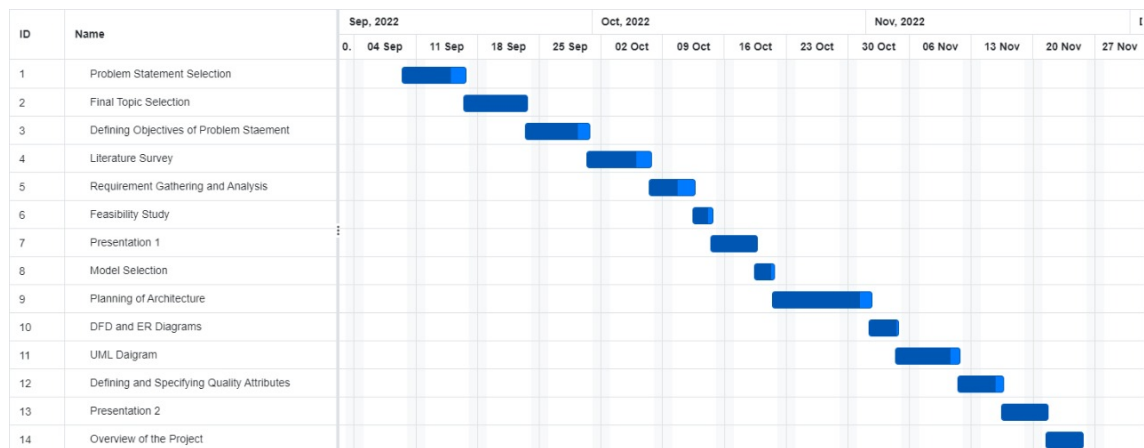


Figure 5.1: Timeline Chart (Phase 1)

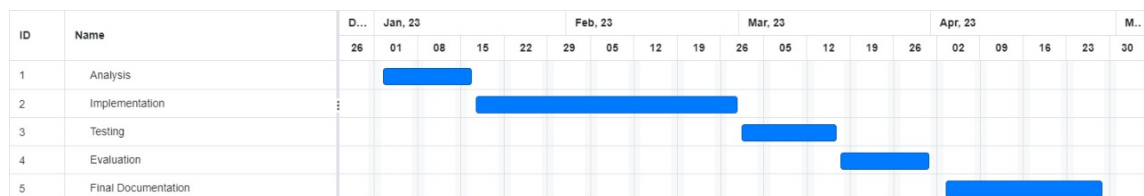


Figure 5.2: Timeline Chart (Phase 2)

5.4 TEAM ORGANIZATION:

5.4.1 Team structure:

1. Akash:
 - Role: Team Lead and Developer
 - Responsibilities:

- Lead the team and ensure smooth coordination
- Develop and deploy smart contracts
- Create the frontend and integrate it with smart contracts
- Manage the overall project timeline and deliverables
- Integrate Metamask wallet functionality

2. Pratik:

- Role: Data Analyst and Frontend Developer
- Responsibilities:
 - Analyze data from testing
 - Assist in frontend development designs
 - Help develop the payment structure using blockchain technology
 - Collaborate with the team for effective data-driven decision-making
 - Implement data visualization and reporting techniques for better insights

3. Vedant:

- Role: Developer
- Responsibilities:
 - Design and integrate Google Maps into the frontend development
 - Work on testing data
 - Implement Google Map functions and features

4. Rachana:

- Role: Data Collector and Tester
- Responsibilities:
 - Collect information about rides
 - Assist in testing the website
 - Provide feedback and suggestions for improvement

5.4.2 Management reporting and communication:

To facilitate effective communication and progress tracking within the team, we utilized various tools and platforms:

1. **Communication Tools:** We primarily used Telegram and WhatsApp as our communication channels. These instant messaging platforms allowed us to have real-time discussions, share updates, and address any queries or concerns promptly.
2. **Version Control and Collaboration:** We utilized Git as our version control system, enabling seamless collaboration among team members. Git allowed us to track changes, merge code contributions, and maintain a centralized repository, ensuring efficient and organized development workflows.

By leveraging these communication tools and version control system, we were able to maintain a smooth flow of information, collaborate effectively, and address any challenges or changes throughout the project lifecycle.

Overall, our team structure, with A focusing on smart contract integration, P contributing to website design and frontend development, V handling Google Maps integration and testing, and R participating in development tasks, allowed us to effectively utilize each team member's expertise and ensure a well-rounded approach to the project.

Chapter 6

Software Requirements Specification

6.1 Introduction

6.1.1 Purpose and Scope of Document

The scope of this project includes project developer assisted by project guide. The scope thus far has been the completion of the basic interfaces that will be used to build the system. The setup should be precise for the capturing of the exact location of driver as well as passenger. The constraints felt thus far by the developer have only been our weekly story cards, the end-to-end side of the interface, and time to time brushing on methodology of implementation which schedule the completion of the project in April 2023.

In this project we are going to make an application which will be decentralized. No third party app will be there so it will increase the security, transparency and trust between driver and passenger because of blockchain which is more secure and trustable.

6.1.2 Overview of responsibilities of Developer

What all activities carried out by developer

1. **Requirements Gathering and Analysis:** Collaborate with stakeholders to understand their needs and gather requirements for the app's features and functionality.
2. **Design and Architecture:** Design the app's overall structure and architecture, ensuring scalability, performance, and maintainability. Determine the appropriate technologies and frameworks to be used.
3. **Frontend Development:** Develop the user interface (UI) components of the app using frontend technologies like HTML, CSS, and JavaScript. Implement features such as user registration, login, ride booking, real-time tracking, and payment integration.
4. **Backend Development:** Build the backend infrastructure to support the app's functionalities. Develop server-side logic, handle data storage and retrieval, and integrate external services like Google Maps and Metamask for payment processing.
5. **API Integration:** Integrate third-party APIs, such as Google Maps API for location services, to provide accurate mapping and navigation features.

6. **Smart Contract Development:** Develop and deploy smart contracts using Solidity to handle cryptocurrency payments and interact with the Ethereum blockchain.
7. **Database Management:** Design and implement the database structure, ensuring efficient data storage and retrieval. Manage data persistence and handle user-related information, ride history, and other relevant data.
8. **Testing and Quality Assurance:** Perform unit testing, End-to-end testing to ensure the app functions as expected. Identify and fix bugs and ensure the app meets quality standards.
9. **Security and Privacy Considerations:** Implement security measures to protect user data and transactions. Follow best practices to secure the app against common vulnerabilities and ensure compliance with privacy regulations.
10. **Deployment and Maintenance:** Deploy the app to production environments, configure servers, and ensure smooth operation. Monitor the app's performance, handle updates and bug fixes, and provide ongoing maintenance and support.
11. **Collaboration and Communication:** Work closely with designers, project guide, and other team members to coordinate efforts, provide progress updates, and address any development-related issues.
12. **Documentation:** Document the app's architecture, codebase, and deployment processes. Provide clear and comprehensive documentation for future reference and troubleshooting.

6.2 Usage Scenario

This section provides various usage scenarios for the system to be developed.

1. **User Requests a Ride:** A user opens the app, enters their desired pickup and drop-off locations, selects a ride option, and requests a ride. The app displays nearby drivers available for the requested ride type.
2. **Driver Accepts Ride Request:** A driver receives a ride request notification, reviews the pickup location and estimated fare, and accepts the request. The app provides navigation directions to the pickup location.

3. **Real-Time Ride Tracking:** As the ride progresses, the user can track the driver's real-time location on the map, ensuring transparency and an estimated time of arrival.
4. **Payment with Cryptocurrency:** At the end of the ride, the app automatically calculates the fare based on distance and time. The user confirms the payment using their Metamask-integrated cryptocurrency wallet.
5. **Ride History and Receipts:** Users can access their ride history, view previous rides, and generate digital receipts with details like pickup/drop-off locations, fare, and driver information.
6. **Account Management:** Users can update their profile information, add multiple payment methods, and manage notifications and preferences within the app.
7. **Smart Contract Integration:** The app interacts with the deployed smart contract using RPC (Ether.js), enabling secure and transparent transaction processing and validating payments.

6.2.1 User profiles

There will be 3 main classes in our product which are User, Smart-Contract and Ride.

1. The User class will be having 2 subclasses Passenger class and Driver class. User class will have registration info which will be common for both Passengers and Drivers. And the information which is different will be added in their separate classes.
2. Smart-Contract class is the class which will have all information about driver, passenger and ride information. All the classes are linked to each other with this class.

6.2.2 Use-cases

All use-cases for the software are presented. Description of all main Use cases using use case template is to be provided.

6.2.3 Use Case View

Use Case Diagram.

Sr No.	Use Case	Description	Actors	Assumptions
1	User Registration	Users can create an account	User	User has a compatible device and an internet connection
2	User Login	Registered users can log into the app	User	User has valid credentials
3	Book a Ride	Users can request a ride from one location to another	User, Passenger	User has a valid payment method
4	Driver Accepts Ride	Drivers can accept ride requests from passengers	Driver	Driver is logged in and available
5	Payment Processing	Payments for rides are processed using cryptocurrency	User, Passenger	User has sufficient funds in their cryptocurrency wallet
6	Ride History	Users can view their past ride details	User	User has completed at least one ride
7	Smart Contract Interaction	Smart Contract communicates with deployed smart contract using RPC	Smart Contract	Smart contract is deployed and accessible on the Ethereum network

6.2.4 Use Case Diagram

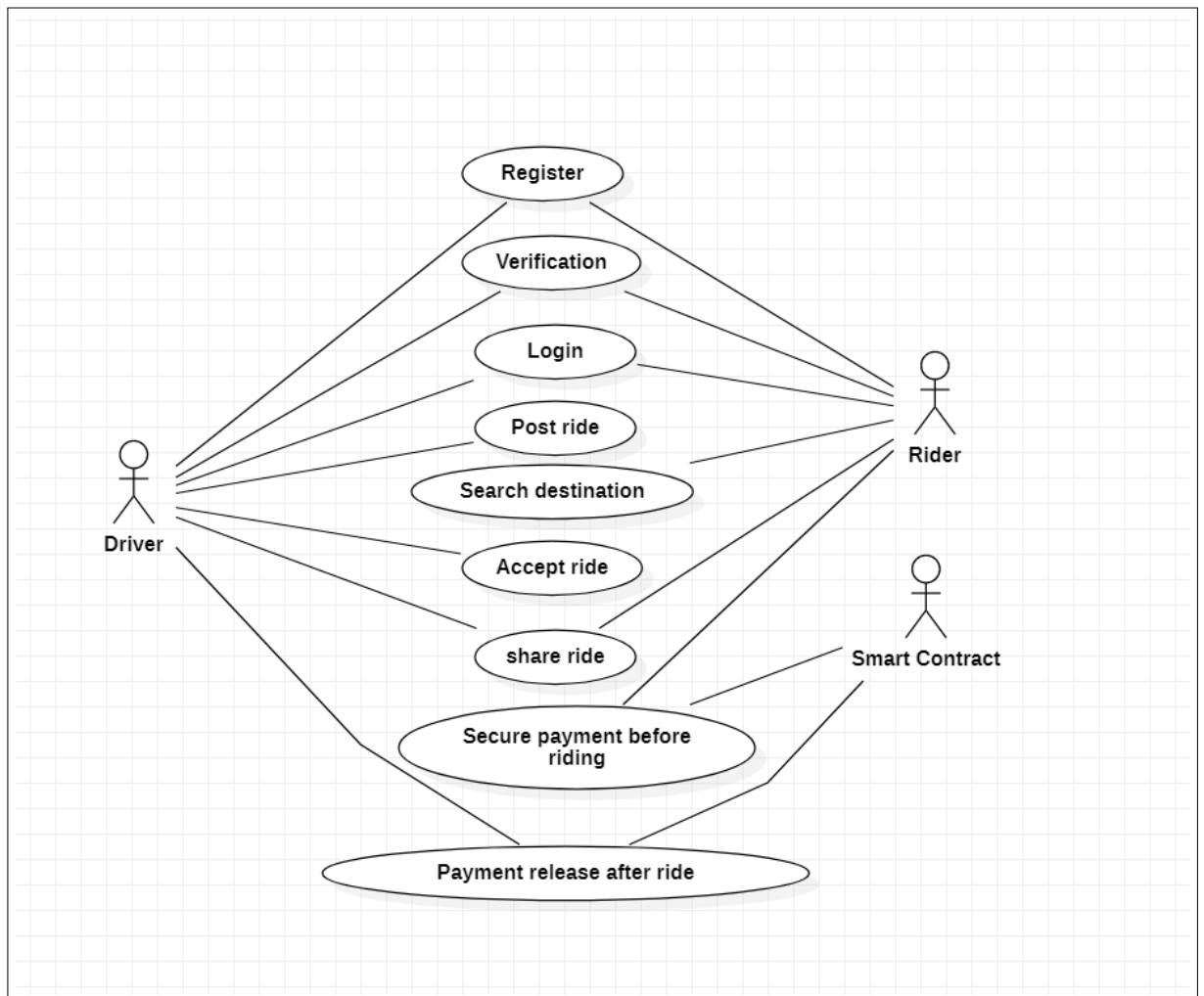


Figure 6.1: Use Case Diagram

6.3 Data Model and Description

6.3.1 Data Description

Data objects that will be managed/manipulated by the software are described in this section. The database entities or files or data structures required to be described. For data objects details can be given as below

6.3.2 Data objects and Relationships

Data objects and their major attributes and relationships among data objects are described using an ERD- like form.

6.4 Functional Model and Description

A description of each major software function, along with data flow (structured analysis) or class hierarchy (Analysis Class diagram with class description for object oriented system) is presented.

6.4.1 User Registration and Authentication:

1. Function: Allows users to register and authenticate themselves within the app.
2. Data Flow: Users provide their personal information, such as name, email, and password, during the registration process. This information is securely stored in a database. When logging in, users enter their credentials, which are validated against the stored data.

6.4.2 Location and GPS Tracking:

1. Function: Tracks and updates the user's current location and provides location-based services.
2. Data Flow: The app utilizes GPS functionality to track the user's location in real-time. The GPS data is retrieved and processed to determine the user's coordinates, which are then used for various purposes like finding nearby drivers or estimating arrival times.

6.4.3 Ride Request and Matching:

1. Function: Allows users to request rides and matches them with available drivers.
2. Data Flow: Users input their desired destination, and the app sends the request along with the user's location to the server. The server matches the request with available drivers based on their proximity and availability. The matched driver is notified of the ride request.

6.4.4 Driver Acceptance and Routing:

1. Function: Enables drivers to accept ride requests and provides navigation instructions.
2. Data Flow: When a driver receives a ride request, they can accept or decline it. If accepted, the server sends the driver's location to the user and provides navigation instructions to reach the user's pickup location. Once the driver arrives, the app guides them to the destination using Google Maps.

6.4.5 Payment Integration

:

1. Function: Integrates the app with the Metamask wallet for secure cryptocurrency payments.
2. Data Flow: Users' Metamask wallets are linked to their app accounts. When a ride is completed, the fare amount is calculated, and the app initiates a payment request to the user's wallet through Metamask. The user approves the transaction, and the payment is securely transferred to the driver's wallet.

6.4.6 Smart Contract Communication:

1. Function: Facilitates communication between the frontend and the deployed smart contract.
2. Data Flow: The frontend application uses the Ether.js library to communicate with the smart contract deployed on the blockchain. It can perform functions such as fetching ride history, updating ride status, and verifying payment transactions.

6.5 Data Flow Diagrams

A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops

6.5.1 Data Flow diagram Level 0(Level 0 DFD)

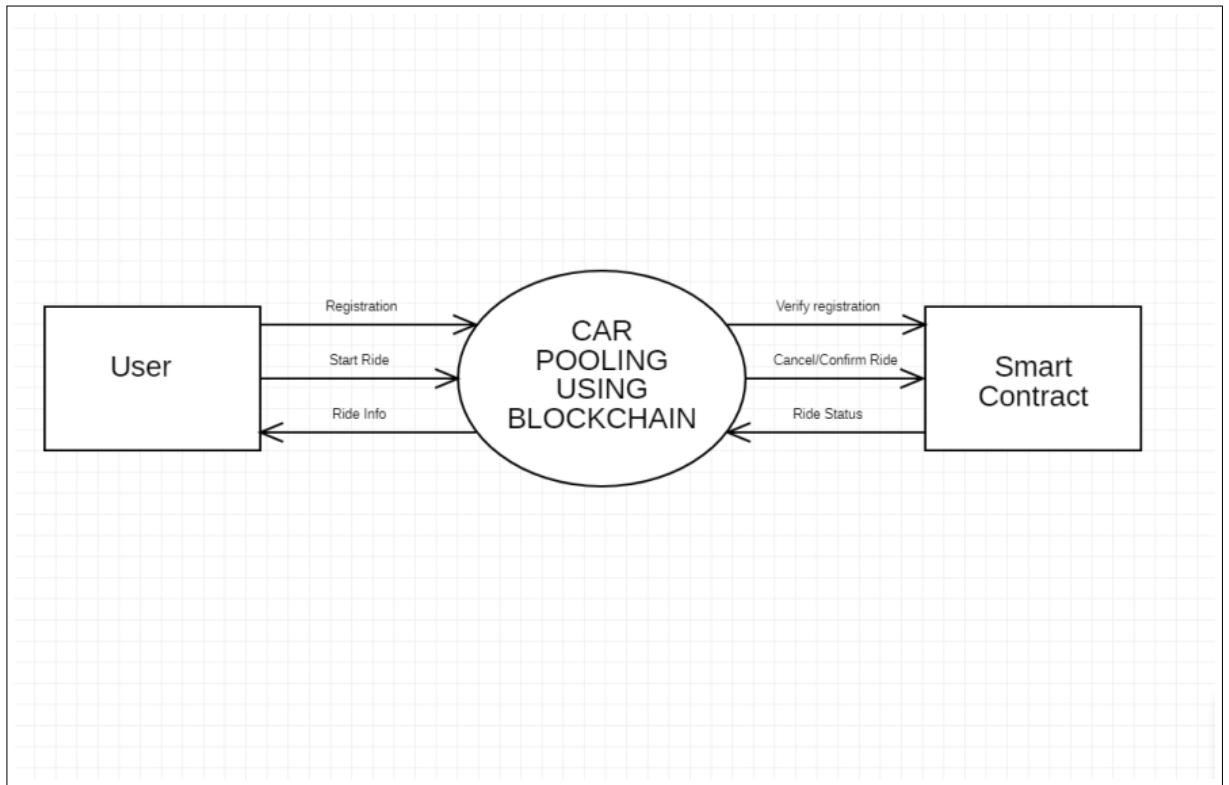


Figure 6.2: Level 0 Data Flow diagram

6.5.2 Data Flow Diagram Level 1(Level 1 DFD)

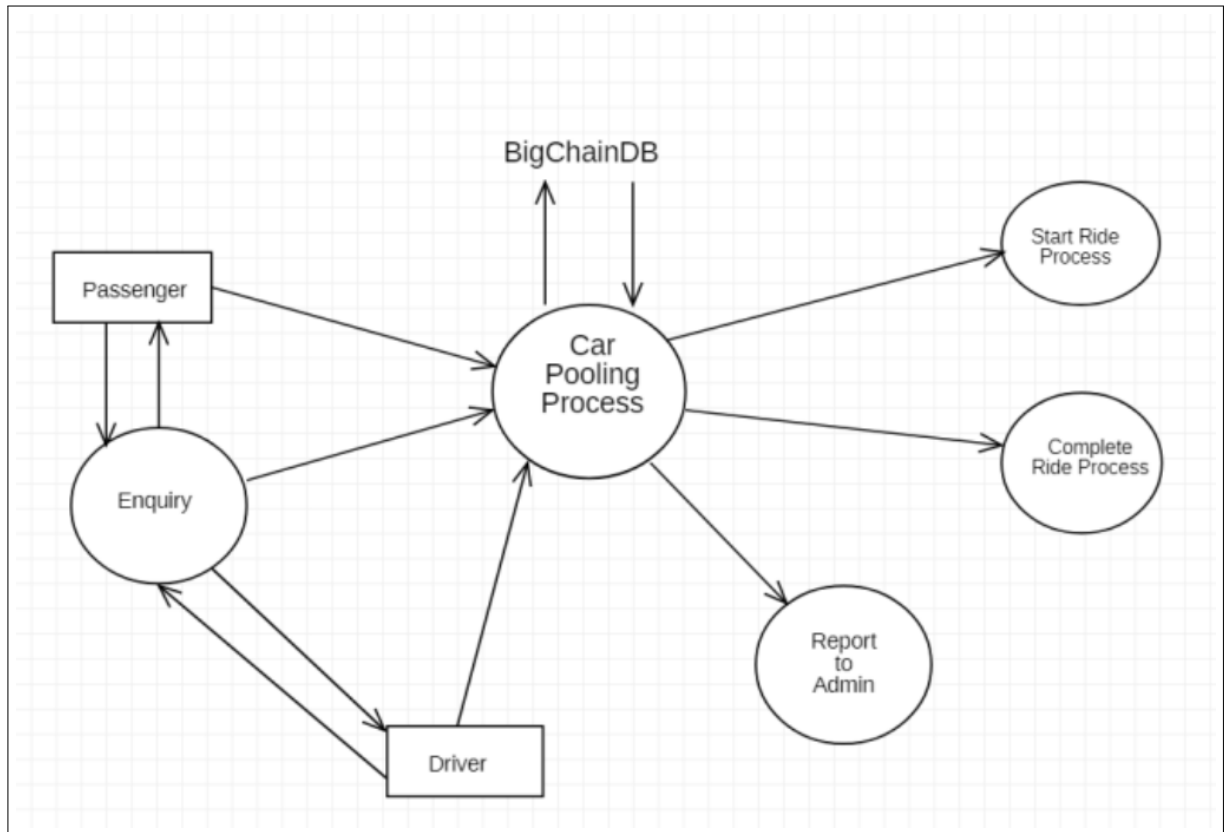


Figure 6.3: Level 1 Data Flow diagram

6.5.3 Data Flow Diagram Level 2 (Level 2 DFD)

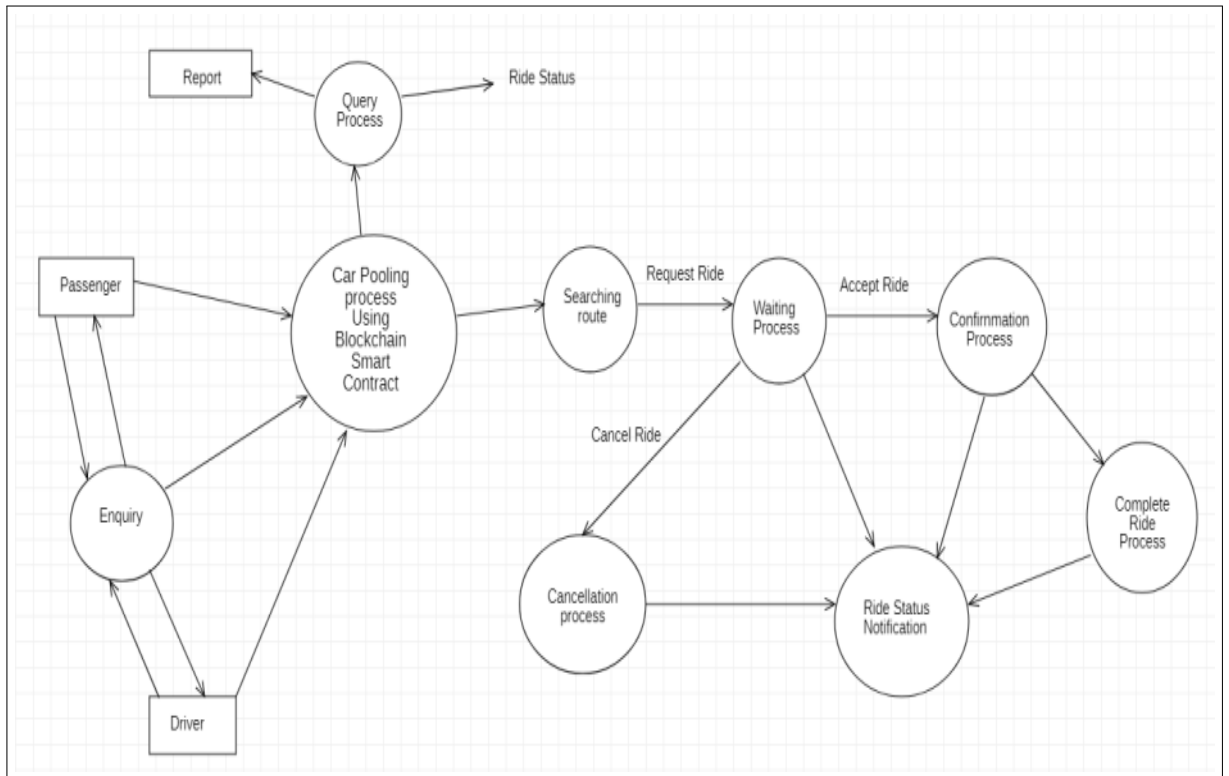


Figure 6.4: Level 2 Data Flow diagram

6.5.4 Activity diagram

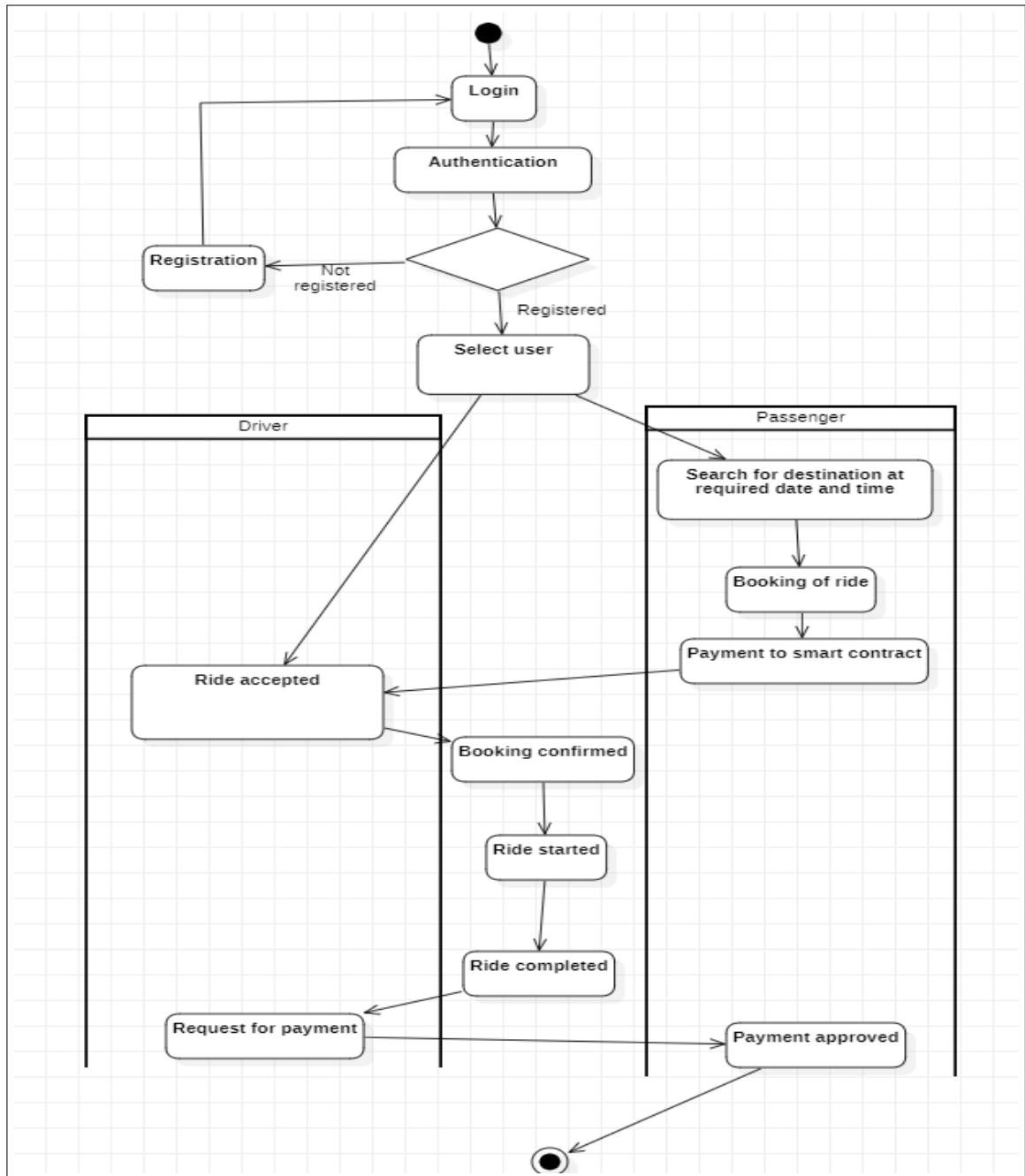


Figure 6.5: Activity Diagram

6.6 Nonfunctional Requirements

6.6.1 Performance Requirements

Performance of the system would be very good and commission less. As other car pooling systems are taking charges, both persons (passenger and driver) have to pay the charges. but our system doesn't have any third party receivers. The money will directly transfer between peers through smart contracts. It's a time saver and instant service.

6.6.2 Safety Requirements

By using kyc there will be fraud users wouldn't get access to the system. All data is stored on blockchain ethereum virtual machine (EVM) and the blockchain uses asymmetric cryptography so no one can access users data without authorized users. payment transfer securely to the receiver and there will be no third party agent in between them.

6.6.3 Security Requirements

Blockchain systems use asymmetric cryptography to secure transactions between users. In these systems, each user has a public and private key. These keys are random strings of numbers and are cryptographically related. It is mathematically impossible for a user to guess another user's private key from their public key.

6.6.4 Software Quality Attributes

There are functions that can only be accessible by roles (passenger, driver). Only passengers can request rides. and drivers can accept rides. All functions are performed easily and instantly. Users can delete their data anytime by themselves easily. The system would be user friendly so both passengers and drivers can use the system app easily from their device. The system is reusable, if a user mistakenly chooses the wrong role or any other information like name, age, etc then the user can edit their info easily and it will update instantly. Every user has full access to their data.

6.6.5 State Diagram:

State Transition Diagram

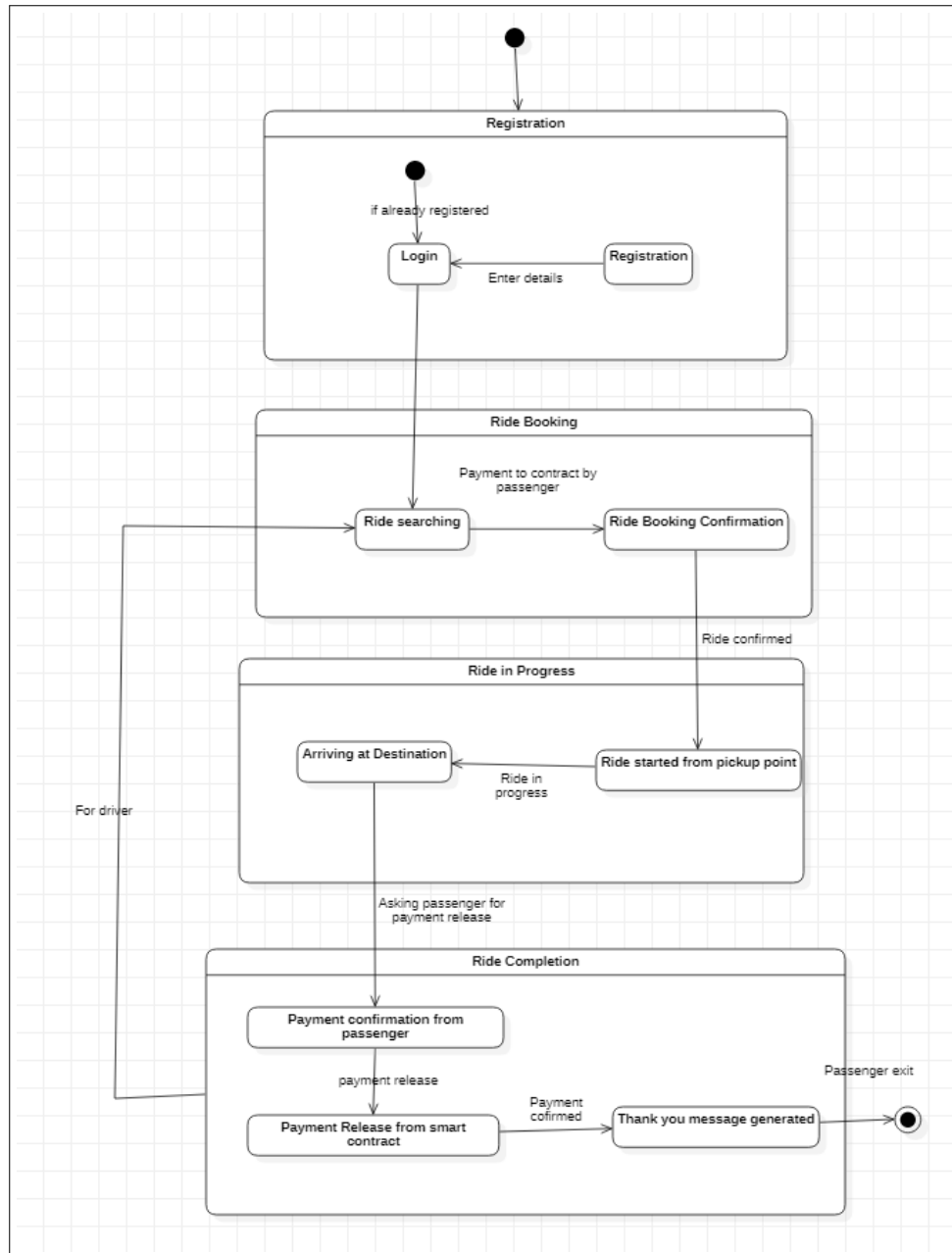


Figure 6.6: State Machine Diagram

6.6.6 Design Constraints

Any design constraints that will impact the subsystem are noted.

1. Hardware Constraints:
 - (a) Are there any specific hardware requirements or limitations for running the application? For example, does the app require certain GPS capabilities or device sensors?
 - (b) Are there any constraints on the target platforms or devices the app should support? For instance, does it need to run on specific operating systems or device types?
2. Compatibility Requirements:
 - (a) Does the application need to integrate with any existing systems, databases, or APIs? Are there any compatibility constraints or dependencies with these systems?
 - (b) Are there any specific software versions or libraries that the application must be compatible with?
3. Regulatory Compliance:
 - (a) Are there any legal or regulatory requirements that the application must adhere to? For example, data privacy regulations, security standards, or payment processing regulations.
4. Performance Constraints:
 - (a) Are there any specific performance requirements or limitations for the application? For instance, response time thresholds or constraints on memory or processing power.
5. Scalability Constraints:
 - (a) - Are there any anticipated scalability requirements for the application? For example, will it need to handle a large number of concurrent users or transactions? - Are there any limitations or constraints on the scalability of the underlying infrastructure, such as cloud service providers or databases?

6.6.7 Software Interface Description

The software interface(s) to the outside world is(are) described. The requirements for interfaces to other devices/systems/networks/human are stated.

The software interfaces of our application can be categorized as follows:

1. User Interface:
 - (a) The app provides a user interface through which users can interact with various features, such as registration, ride request, tracking, and payment. The interface allows users to input their ride preferences, view available drivers, track the driver's location, and monitor the progress of their ride.
2. GPS and Location Services:
 - (a)
 - The app integrates with GPS and location services to track the user's current location, provide accurate pickup and drop-off coordinates, and calculate route distances and estimated arrival times.
3. Google Maps API:
 - (a) The app utilizes the Google Maps API for functionalities like displaying maps, rendering navigation routes, and geocoding addresses.
 - (b) The interface with the API allows the app to obtain map data, retrieve directions, and display markers for drivers and user locations.
4. Metamask Integration:
 - (a) The app integrates with the Metamask wallet using the Metamask API for cryptocurrency payment processing.
 - (b) The interface enables the app to request payments from users, obtain payment authorization, and verify transaction status.
5. Smart Contract Communication:
 - (a) The app communicates with the deployed smart contract using the Ether.js library and RPC (Remote Procedure Call).
 - (b) The interface enables the app to interact with the smart contract by calling functions, reading data, and subscribing to events.

Chapter 7

Detailed Design Document using Appendix A and B

7.1 Introduction

This document specifies the design that is used to solve the problem of Product.

7.2 Architectural Design

A description of the program architecture is presented. Subsystem design or Block diagram,Package Diagram,Deployment diagram with description is to be presented.

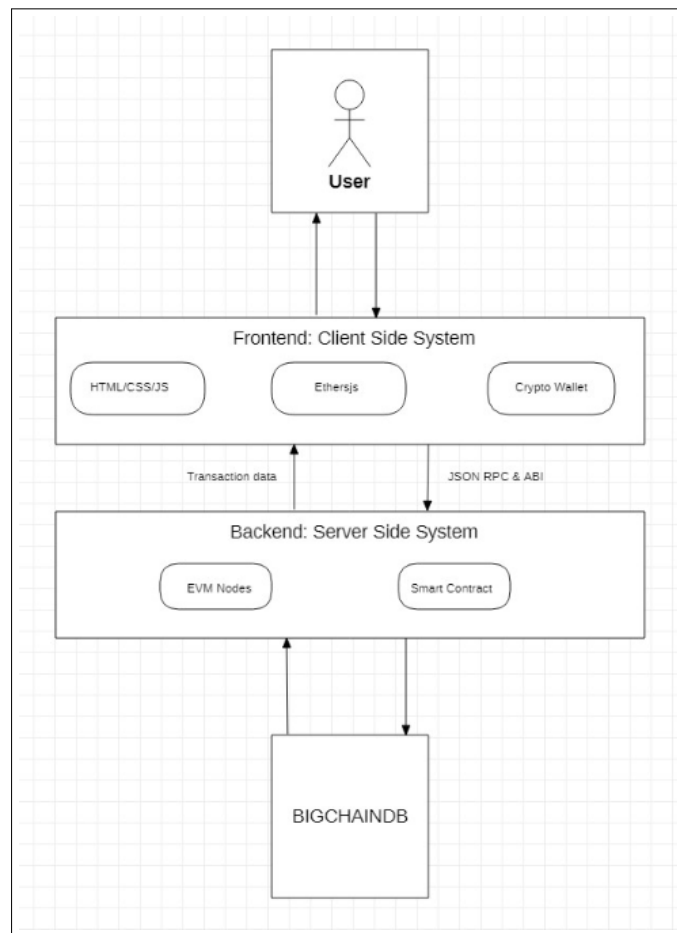


Figure 7.1: System Architecture of our System

7.2.1 Deployment diagram

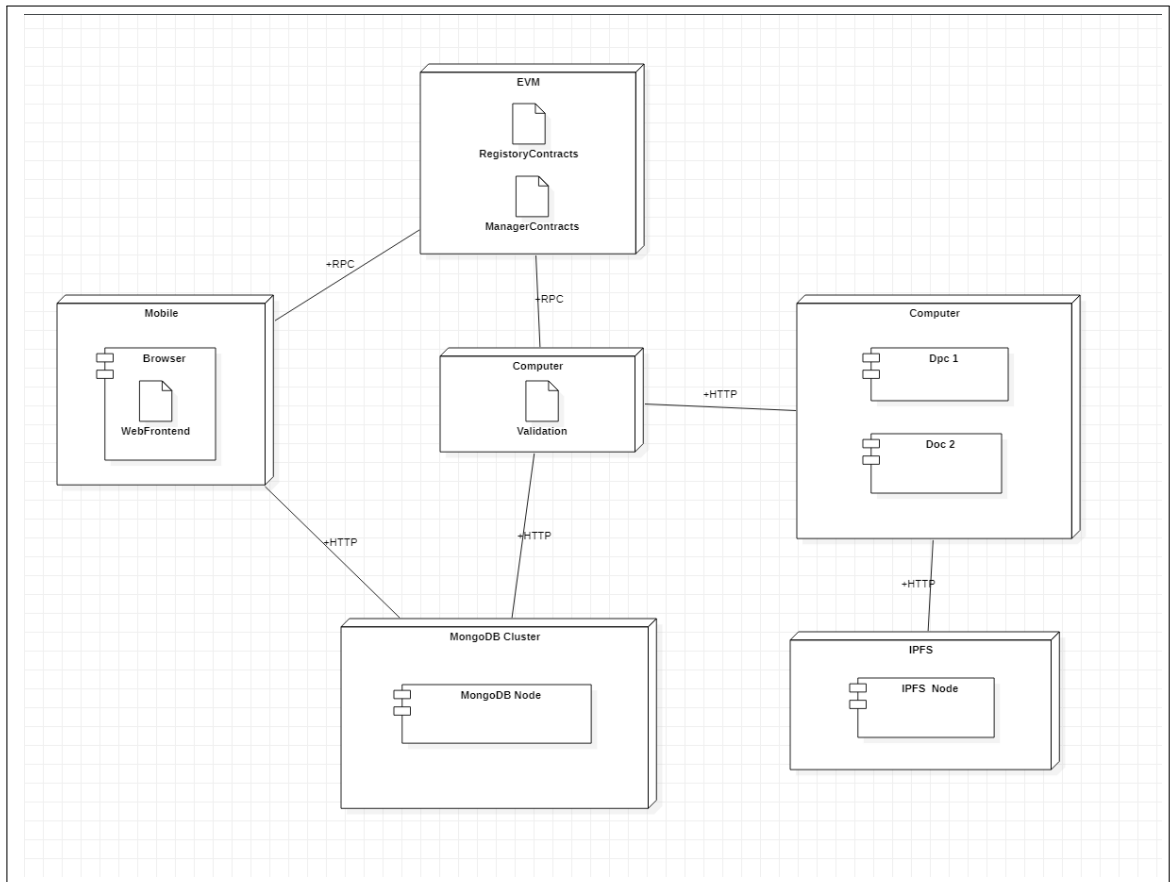


Figure 7.2: Deployment Diagram

7.3 Data design (using Appendices A and B)

A description of all data structures including internal, global, and temporary data structures, database design (tables), file formats.

7.3.1 Internal software data structure

Data structures that are passed among components the software are described.

7.3.2 Global data structure

Data structured that are available to major portions of the architecture are described.

7.3.3 Temporary data structure

Files created for interim use are described.

7.3.4 Database description

Database(s) / Files created/used as part of the application is(are) described.

7.4 Component Design

Class diagrams, Interaction Diagrams, Algorithms. Description of each component description required.

7.4.1 Class Diagram

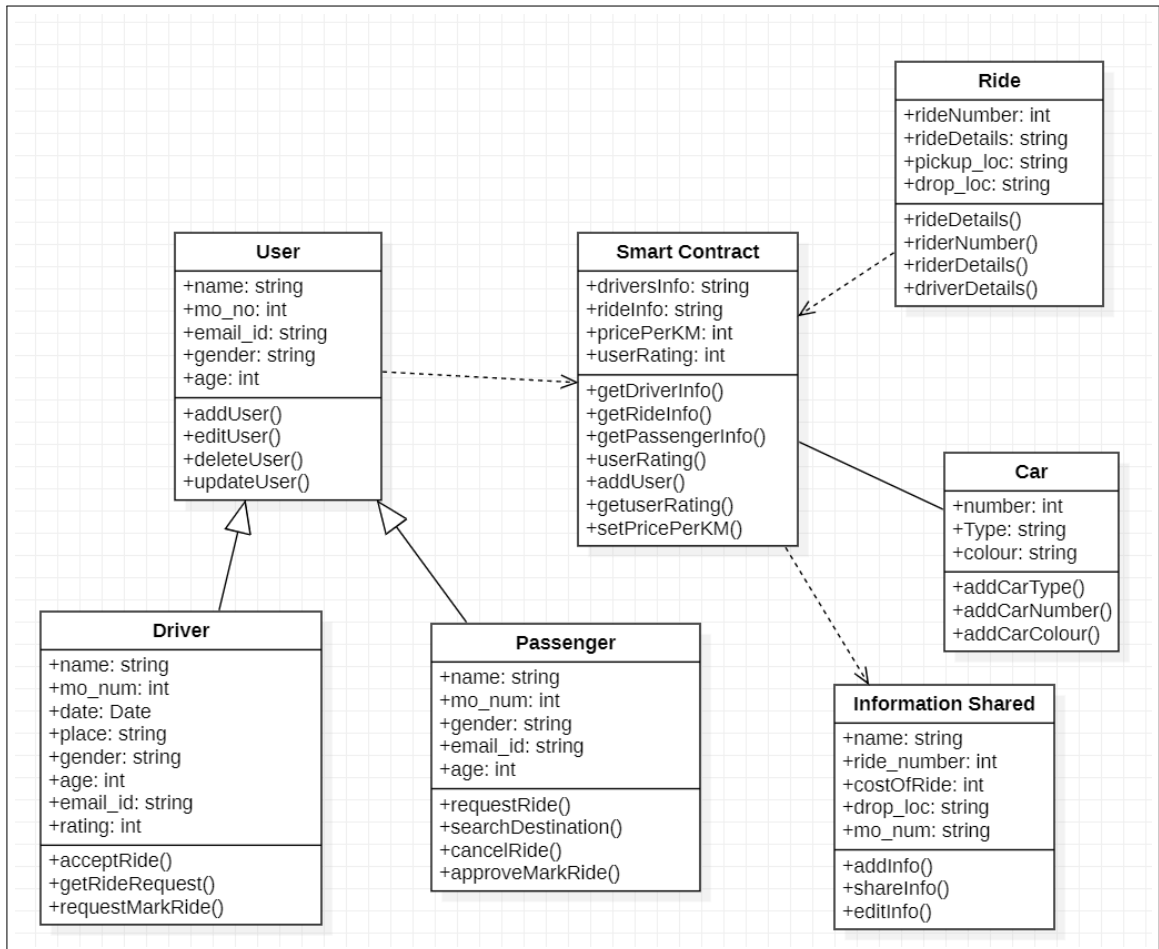


Figure 7.3: Class Diagram

7.4.2 Sequence diagram

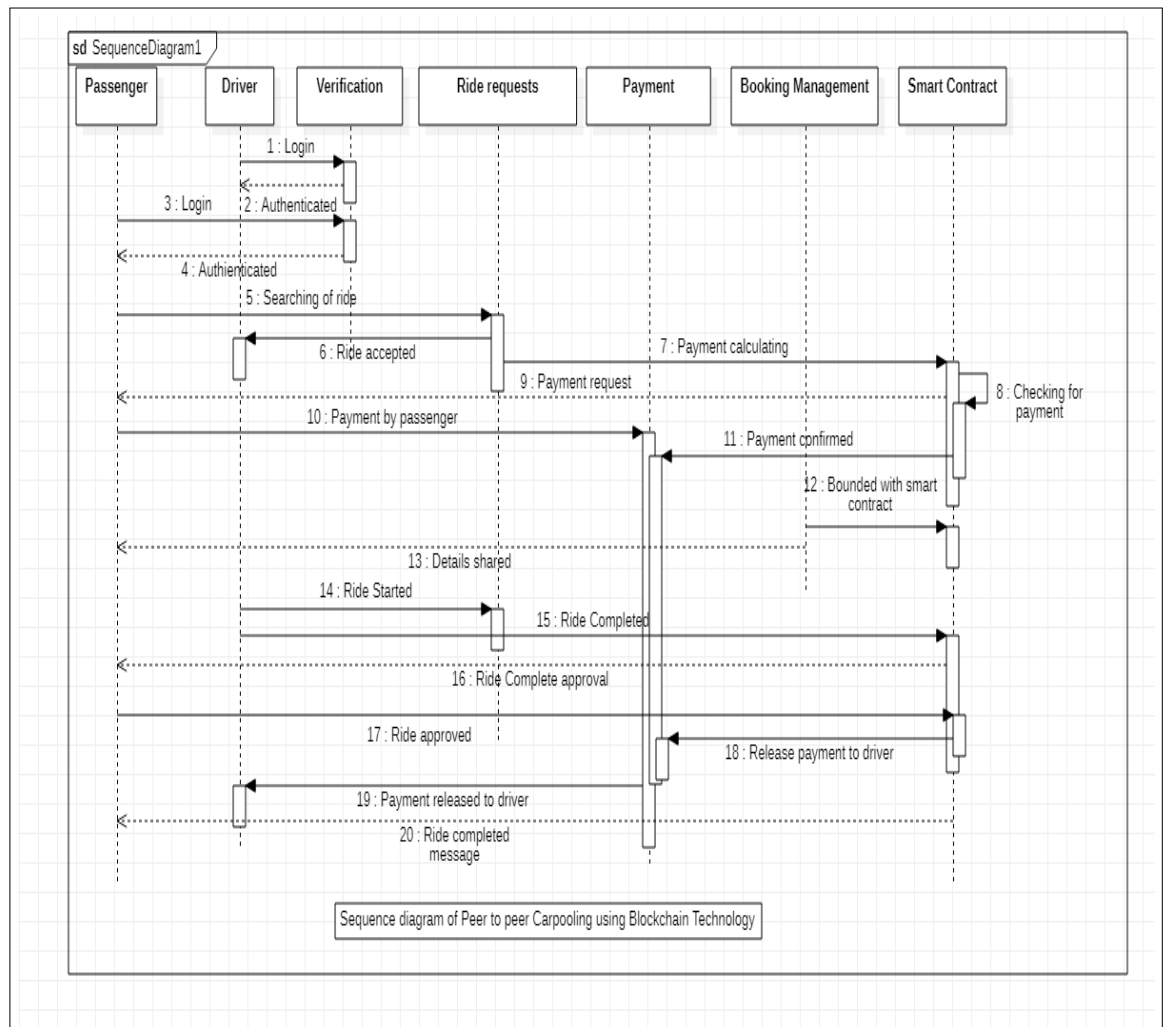


Figure 7.4: Sequence Diagram

7.4.3 Component diagram

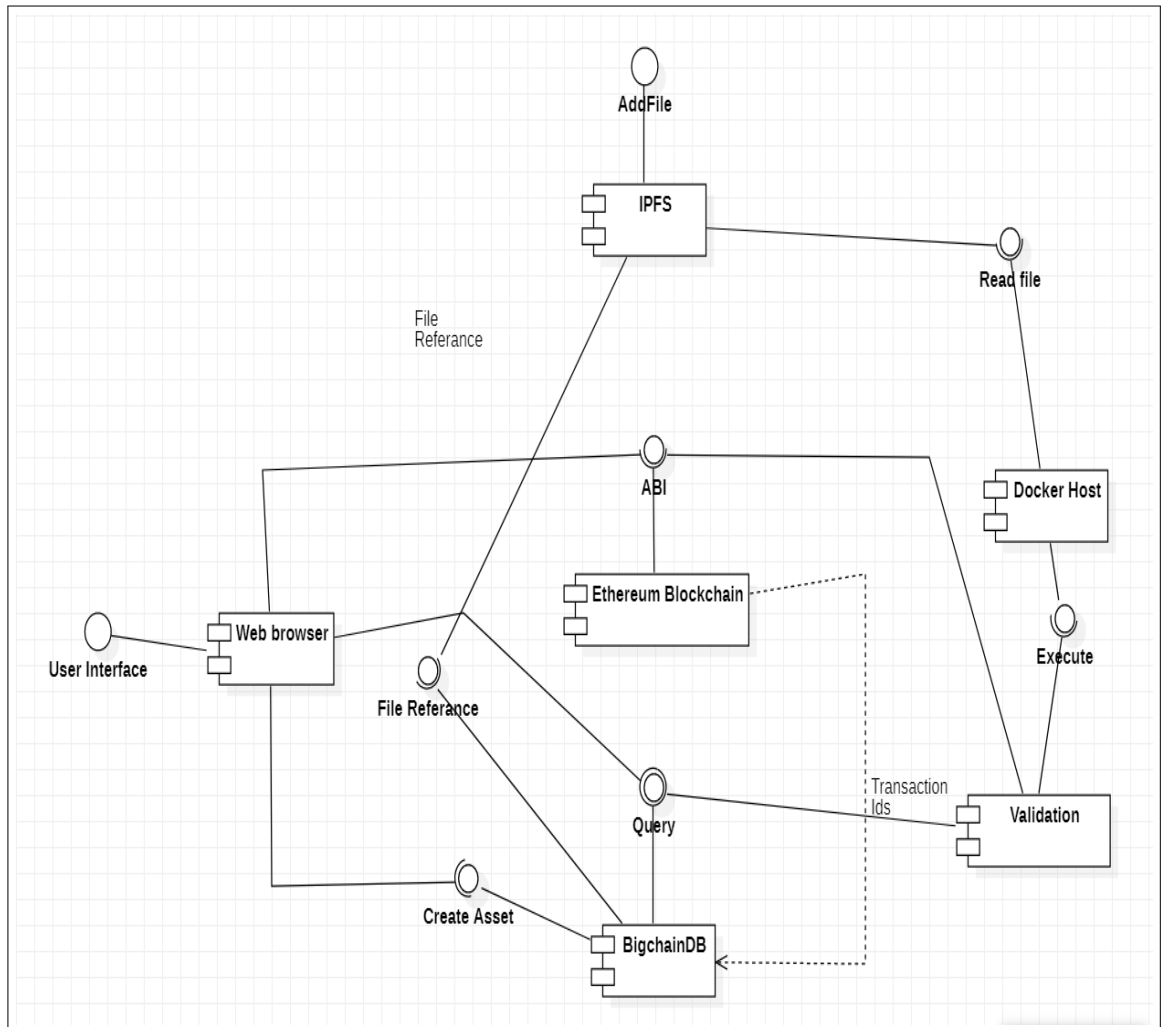


Figure 7.5: Component Diagram

Chapter 8

Project Implementation

8.1 Introduction

8.1.1 Overview

The objective of the project was to develop a platform that enables users to request transportation services, incorporating features such as GPS, user location, Google Maps integration, and cryptocurrency payments through Metamask. Below we outline the tools, technologies, methodologies, algorithms, and verification and validation processes used during the project's implementation.

8.2 Tools and Technologies Used

The development of our app involved the use of various tools and technologies, including:

- Version Control System:
 - Git was used as the Version Control System (VCS) to manage the source code and facilitate collaboration among developers.
 - GitHub was utilized as the hosting platform for the Git repository, enabling seamless collaboration and version tracking.
- Frontend Development:
 - Visual Studio Code (VSCode) was the preferred IDE for frontend development tasks. It provided a versatile and efficient coding environment.
 - TailwindCSS, a utility-first CSS framework, was employed for styling the frontend components, allowing for rapid and consistent UI development.
- Smart Contract Development:
 - The Remix IDE was employed for smart contract development. It offered a user-friendly interface and powerful features for writing and testing Solidity contracts.
- Programming Languages:
 - Solidity was used for smart contract development, enabling the creation of secure and decentralized applications.

- JavaScript was utilized for implementing the frontend functionality and interactivity.
- Frameworks/Libraries:
 - Next.js was the chosen framework for building the website and UI components. It provided server-side rendering, efficient routing, and a rich set of tools for building modern web applications.
 - Ether.js was used for seamless communication with the smart contract, facilitating interactions with the blockchain network.
- APIs/Services:
 - Google Maps API was integrated into our app to leverage location services and enable mapping functionality. This allowed users to interact with the app using geolocation features.
 - Additionally, Metamask was utilized for authentication and seamless integration of cryptocurrency payments, providing a secure and user-friendly experience.
- Development Environment:
 - We utilized IDEs (Integrated Development Environments) for coding, allowing developers to write and manage their code efficiently.
 - Version control systems, such as Git, were used to track changes, collaborate, and manage code revisions.
 - Deployment platforms were utilized to host the app, making it accessible to users.

8.3 Methodologies/Algorithm Details

During the implementation of our project, we followed an Agile development methodology. This iterative approach allowed us to quickly adapt to changing requirements and continuously deliver working increments of the app. The development process involved the following steps:

1. Requirement gathering and analysis: We identified the key features and functionalities required for our app, considering user needs and market trends.

2. Design and architecture: We designed the system architecture, including the frontend components, backend infrastructure, and integration with external services.
3. Implementation: The development team began implementing the app's features, utilizing the chosen technologies and following best coding practices.
4. Testing and debugging: We performed unit testing to verify the functionality of individual components and integration testing to ensure proper interaction between different modules.
5. Deployment and delivery: The app was deployed to a hosting platform, making it accessible to users.

8.3.1 Algorithm: User Location Retrieval

1. Initialize the app and request location permissions from the user.
2. If permission is granted:
 - Utilize the GPS module to retrieve the user's current location.
 - Store the latitude and longitude coordinates.
 - Update the user's location on the map interface.
3. If permission is denied:
 - Display an error message to the user indicating that location services are required for app functionality.

8.3.2 Algorithm: Cryptocurrency Payment Processing

1. Connect the app with Metamask for cryptocurrency payment integration.
2. Retrieve the payment details from the user, including the amount, recipient address, and selected cryptocurrency.
3. Verify that the user's Metamask account is authenticated and has sufficient funds.
4. Initiate the transaction by invoking the smart contract's payment function with the necessary parameters.

5. Wait for the transaction confirmation from the blockchain network.
6. Update the payment status on the app's interface, indicating the success or failure of the transaction.

8.4 Verification and Validation for Acceptance

To ensure the app's acceptance and quality, we conducted various verification and validation activities:

- Unit testing: We developed and executed unit tests for individual components and functions to verify their correctness and expected behavior.
- Integration testing: We performed integration testing to verify the interaction between different modules and external services, such as the smart contract and Metamask integration.
- User acceptance testing: We engaged a group of beta users to test the app's functionality, usability, and overall user experience. Feedback from the users was collected and incorporated into the final version of the app.
- Security testing: We conducted security assessments to identify and address any potential vulnerabilities in the app, especially concerning user data privacy and cryptocurrency transactions.

Throughout the verification and validation process, we encountered a few challenges, such as compatibility issues with different devices and intermittent connectivity problems. These issues were addressed through rigorous debugging, testing, and optimization efforts.

This concludes an overview of the implementation of our project. The app integrates GPS, user location, Google Maps, Metamask integration for cryptocurrency payments, and has been developed using Next.js for the website and Ether.js for frontend communication with the smart contract.

Chapter 9

Results

9.1 Screen shots

9.1.1 Passenger Section

9.1.1.1 Home page

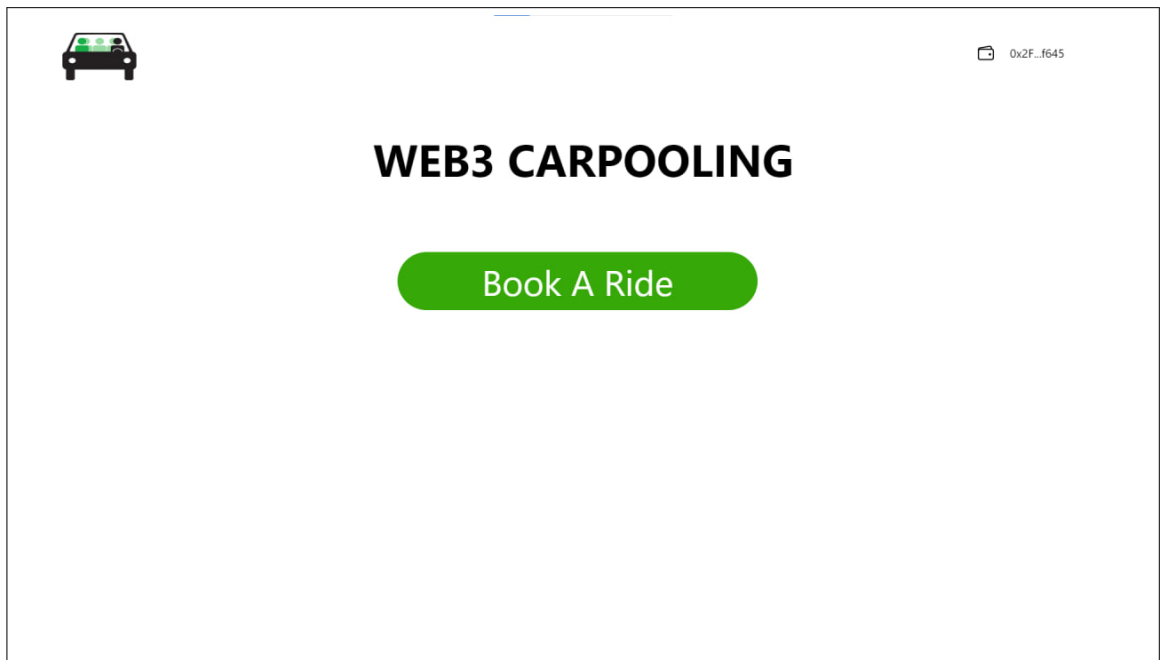


Figure 9.1: Home page

9.1.1.2 Login page

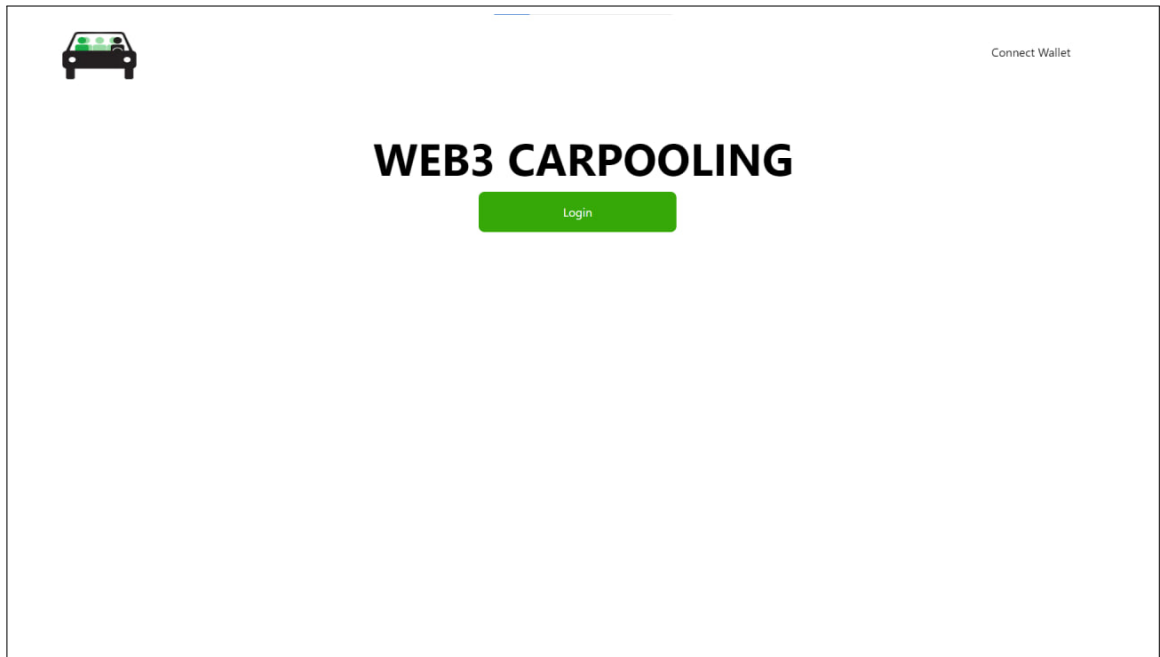


Figure 9.2: Login page

9.1.1.3 User Activity

User Activities

Ride ID	Event	Driver Cost	From	To
7	Completed <u>0x8_272</u>	20	Shaniwar Wada, Shaniwar Peth, Pune, Maharashtra, India	Kem Hospital, Somwar Peth, Pune, Maharashtra, India
7	Accepted <u>0x8_272</u>	20	Shaniwar Wada, Shaniwar Peth, Pune, Maharashtra, India	Kem Hospital, Somwar Peth, Pune, Maharashtra, India
7	Requested -	20	Shaniwar Wada, Shaniwar Peth, Pune, Maharashtra, India	Kem Hospital, Somwar Peth, Pune, Maharashtra, India
5	Cancelled -	60	562/45a, Manthan Society, Pune, Somnath Nagar, Wadgaon Sheri, Pune, Maharashtra 411014, India	Modern Education Society's College of Engineering (MESCOE), Bund Garden Road, Sangamvadi, Pune, Maharashtra, India
5	Requested -	60	562/45a, Manthan Society, Pune, Somnath Nagar, Wadgaon Sheri, Pune, Maharashtra 411014, India	Modern Education Society's College of Engineering (MESCOE), Bund Garden Road, Sangamvadi, Pune, Maharashtra, India
4	Cancelled -	60	Sr. no 45, near Somnath Temple, Somnath Nagar, Wadgaon Sheri, Pune, Maharashtra 411014, India	Modern Education Society's College of Engineering (MESCOE), Bund Garden Road, Sangamvadi, Pune, Maharashtra, India
4	Requested -	60	Sr. no 45, near Somnath Temple, Somnath Nagar, Wadgaon Sheri, Pune, Maharashtra 411014, India	Modern Education Society's College of Engineering (MESCOE), Bund Garden Road, Sangamvadi, Pune, Maharashtra, India
3	Completed <u>0x8_272</u>	20	Modern Education Society's College of Engineering (MESCOE), Bund Garden Road, Sangamvadi, Pune, Maharashtra, India	Pune Station Bus Stand, Unnammed Road, Agarkar Nagar, Pune, Maharashtra, India
3	Accepted <u>0x8_272</u>	20	Modern Education Society's College of Engineering (MESCOE), Bund Garden Road, Sangamvadi, Pune, Maharashtra, India	Pune Station Bus Stand, Unnammed Road, Agarkar Nagar, Pune, Maharashtra, India
3	Requested -	20	Modern Education Society's College of Engineering (MESCOE), Bund Garden Road, Sangamvadi, Pune, Maharashtra, India	Pune Station Bus Stand, Unnammed Road, Agarkar Nagar, Pune, Maharashtra, India

Figure 9.3: User Activity

9.1.1.4 Account Info

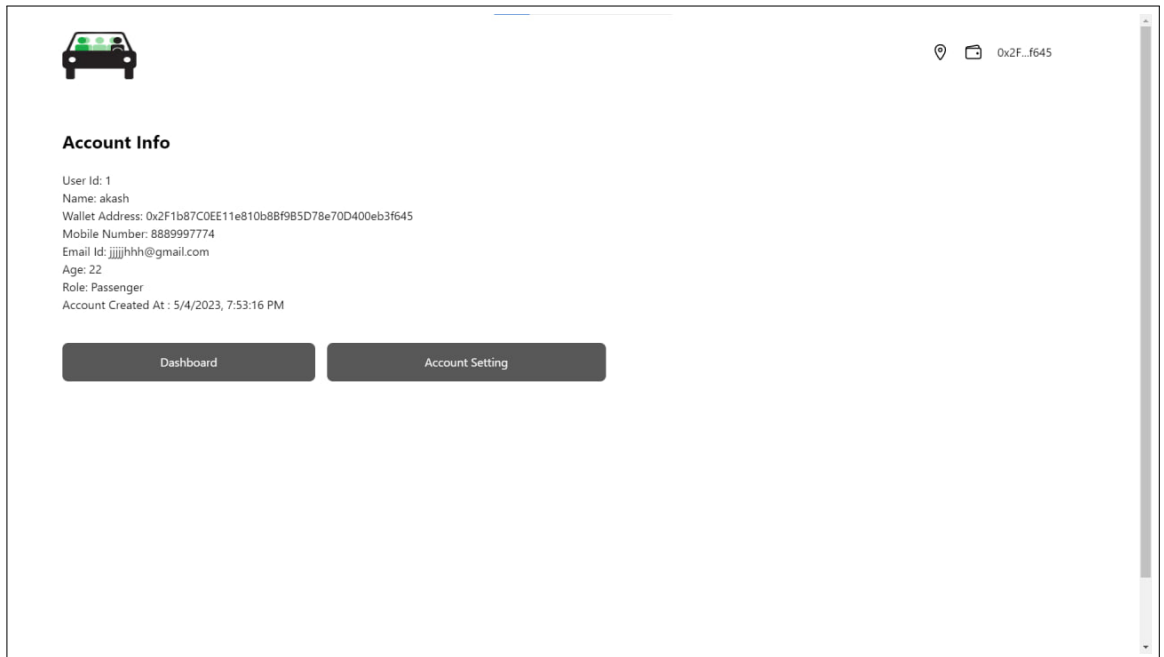


Figure 9.4: Account Info

9.2 Driver

9.2.0.1 View Available rides

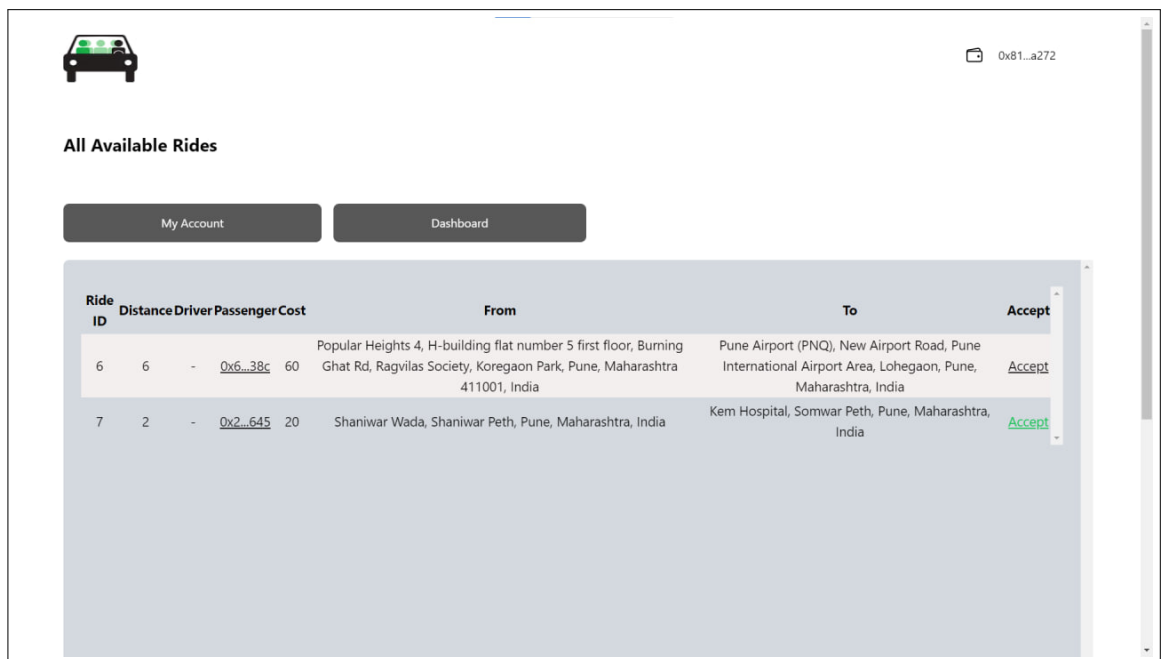


Figure 9.5: View Available Rides

9.2.0.2 Account Info

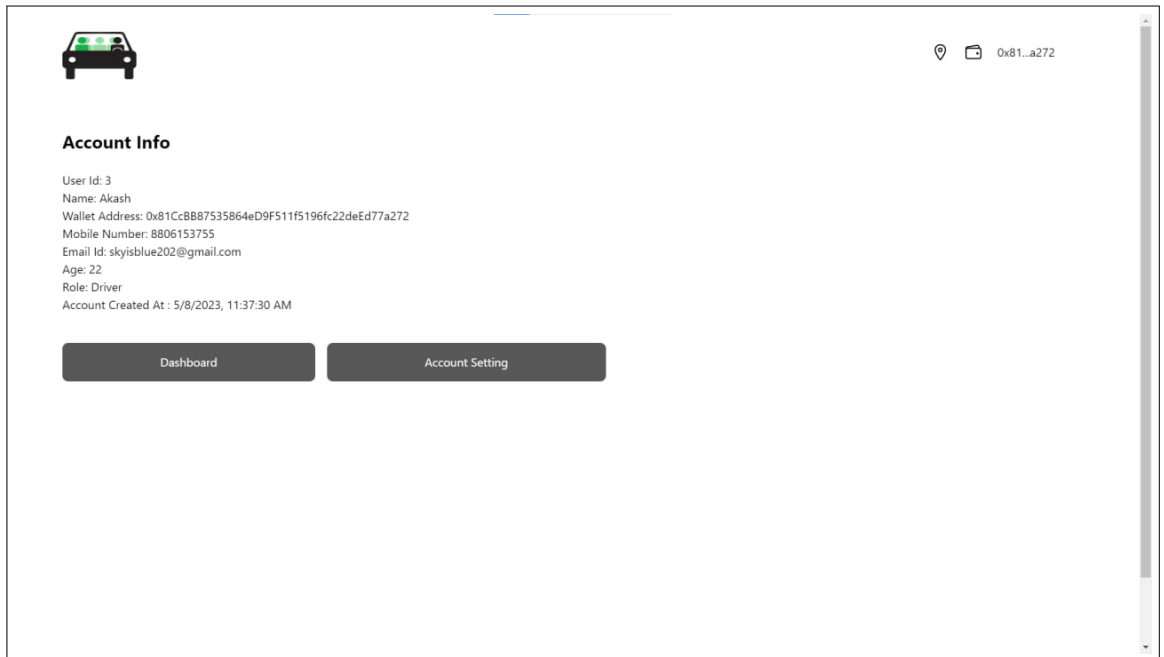


Figure 9.6: Account Info

9.2.0.3 Dashboard

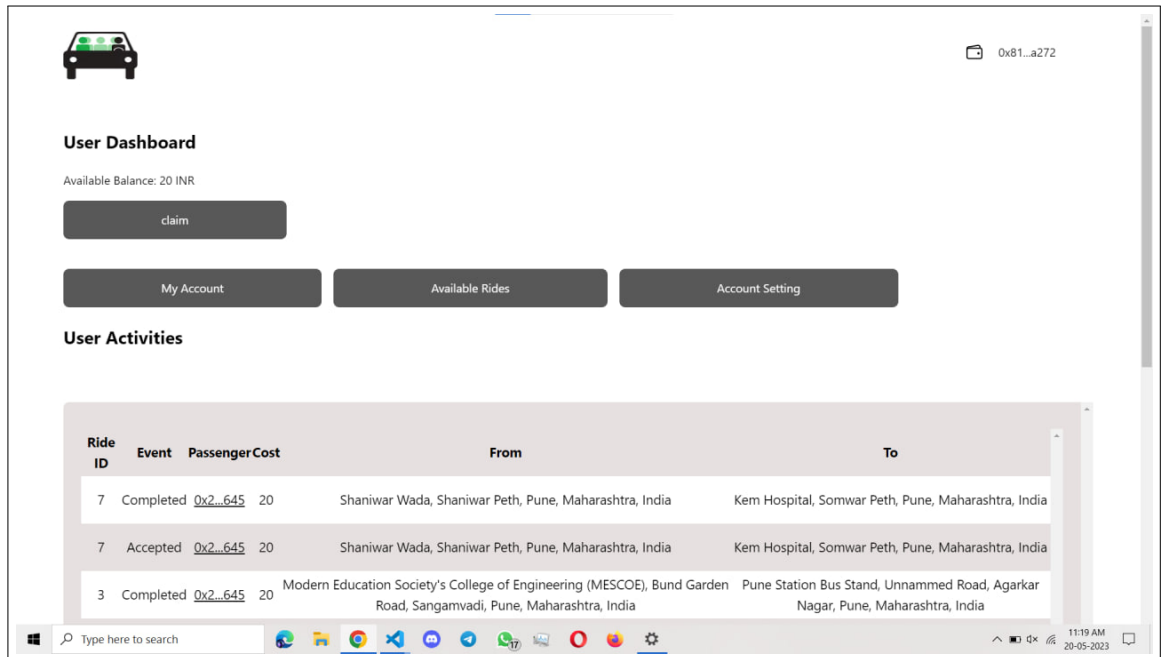


Figure 9.7: Dashboard

9.3 Outputs

9.3.1 Ride Booking by passenger

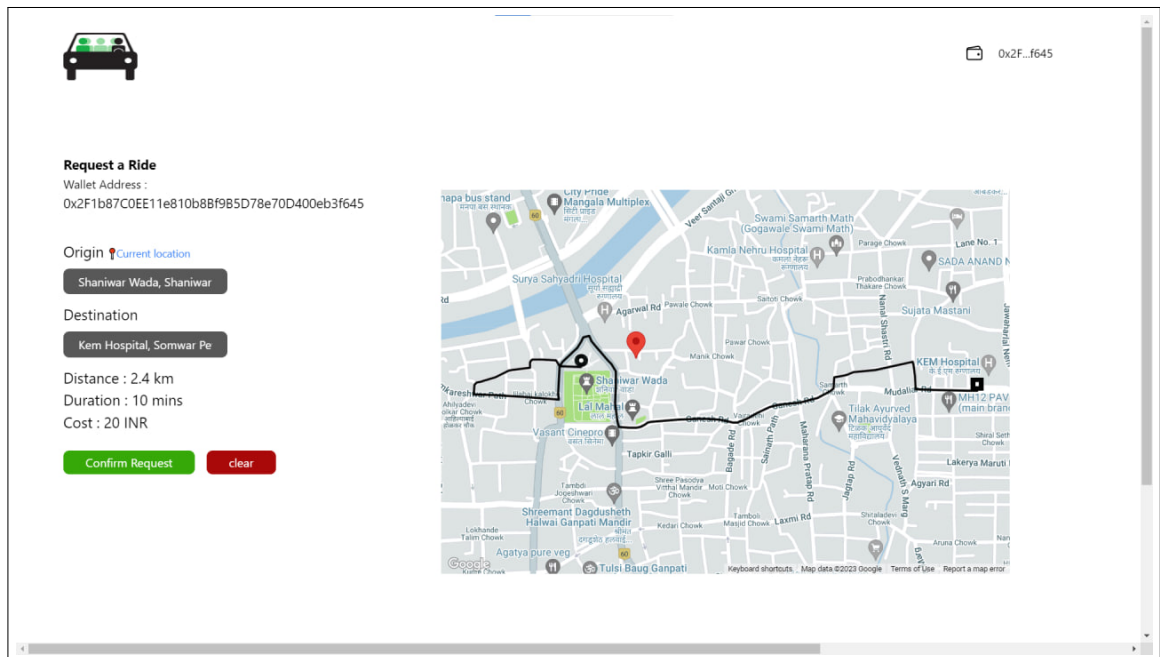


Figure 9.8: Ride booking page

9.3.2 Active Ride Details

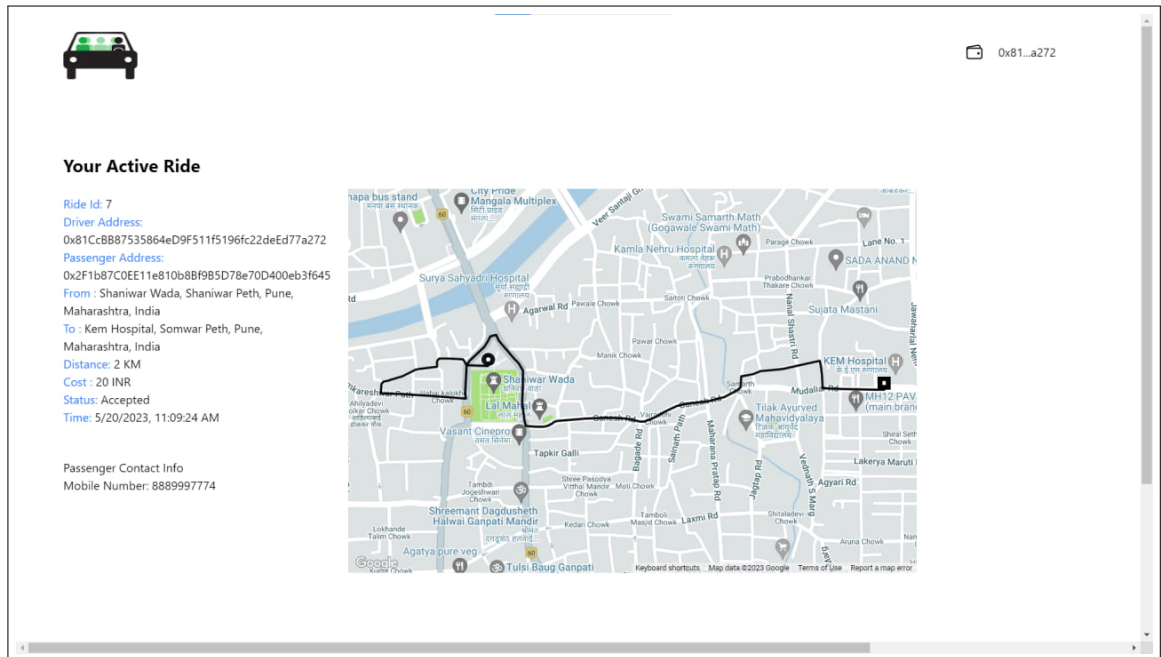


Figure 9.9: Active Ride Details

9.3.3 Payment approval from passenger

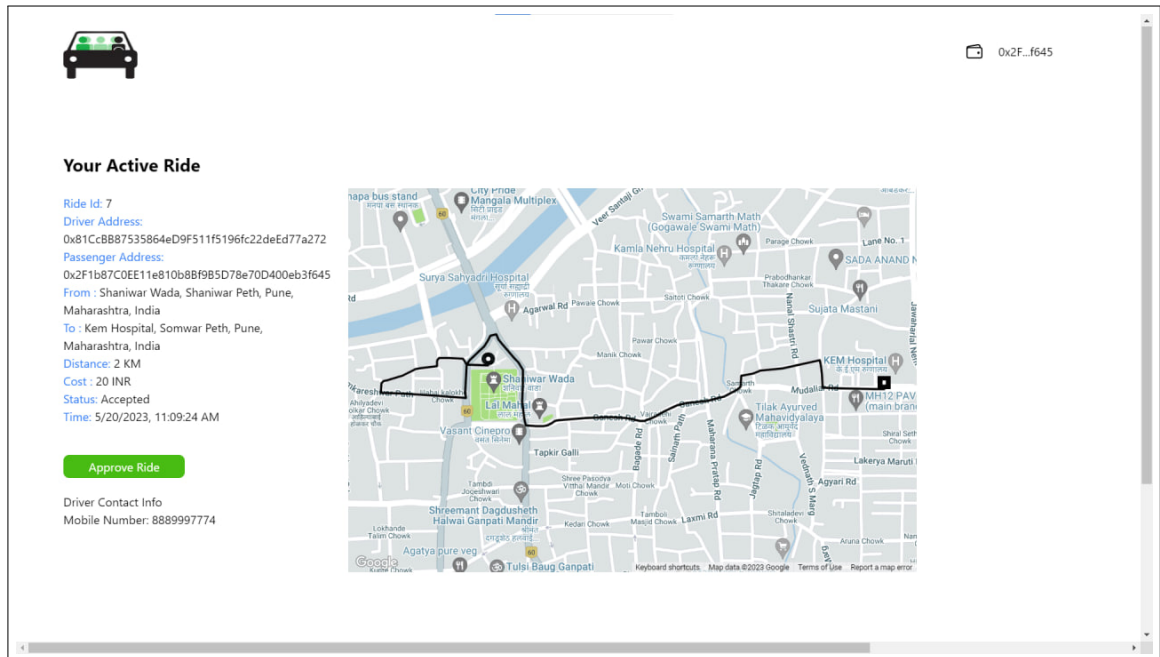


Figure 9.10: Payment approval from passenger

9.3.4 Cancel booking

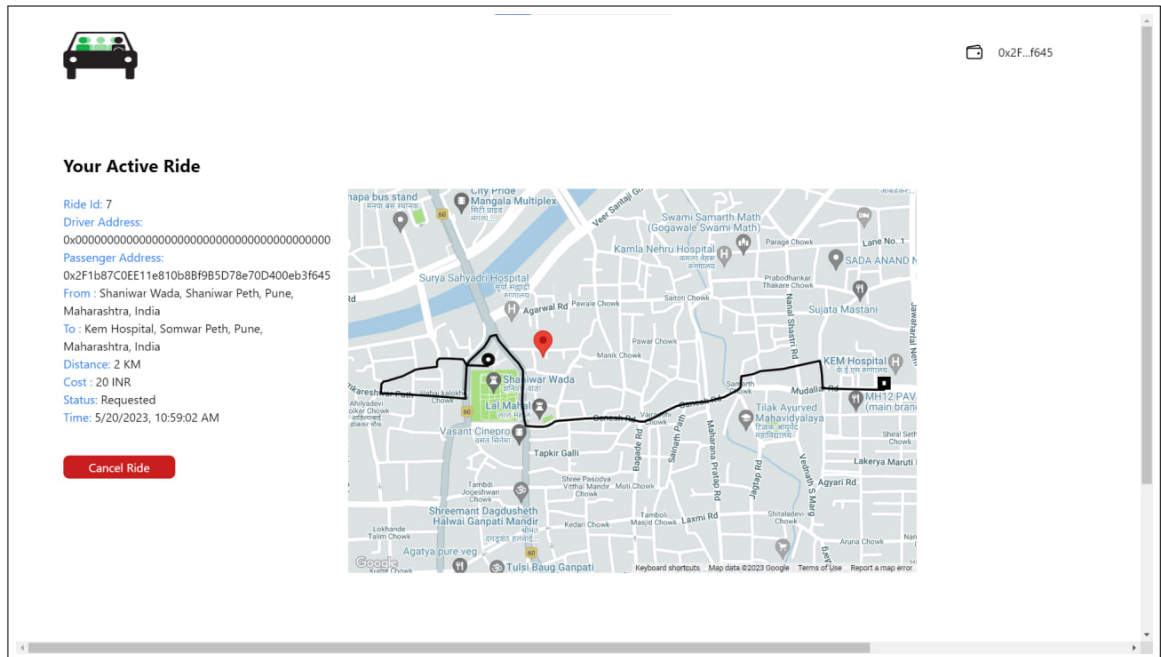


Figure 9.11: Cancel booking

9.3.5 Transactions

9.3.5.1 Ride booking

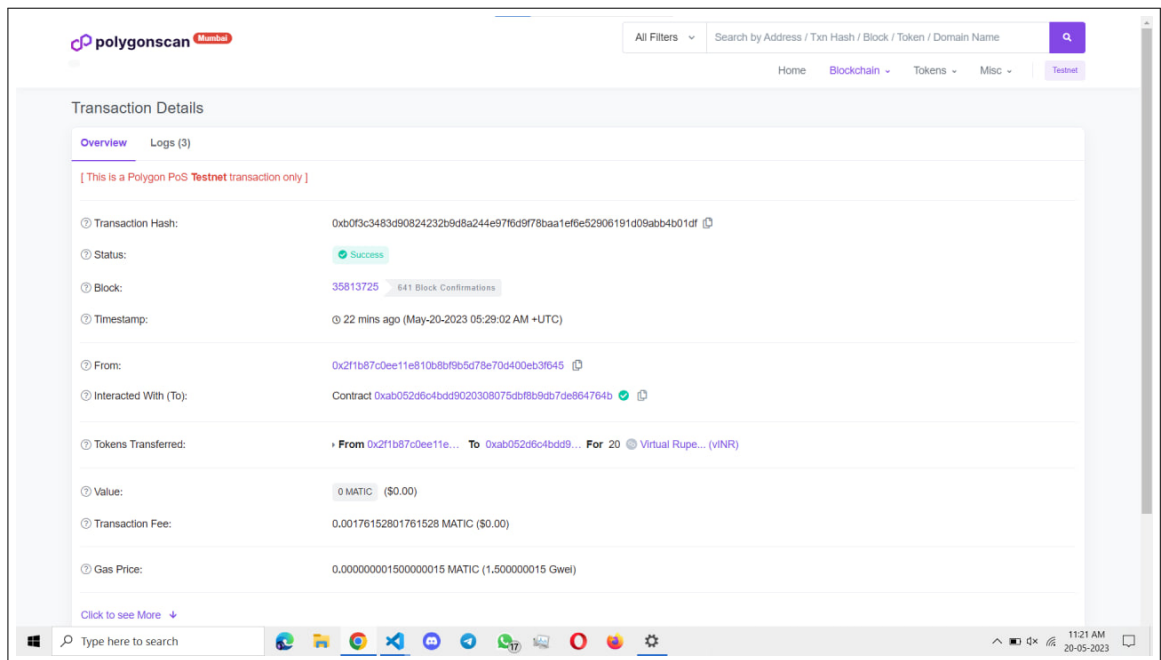


Figure 9.12: Ride booking

9.3.5.2 Payment Withdrawal

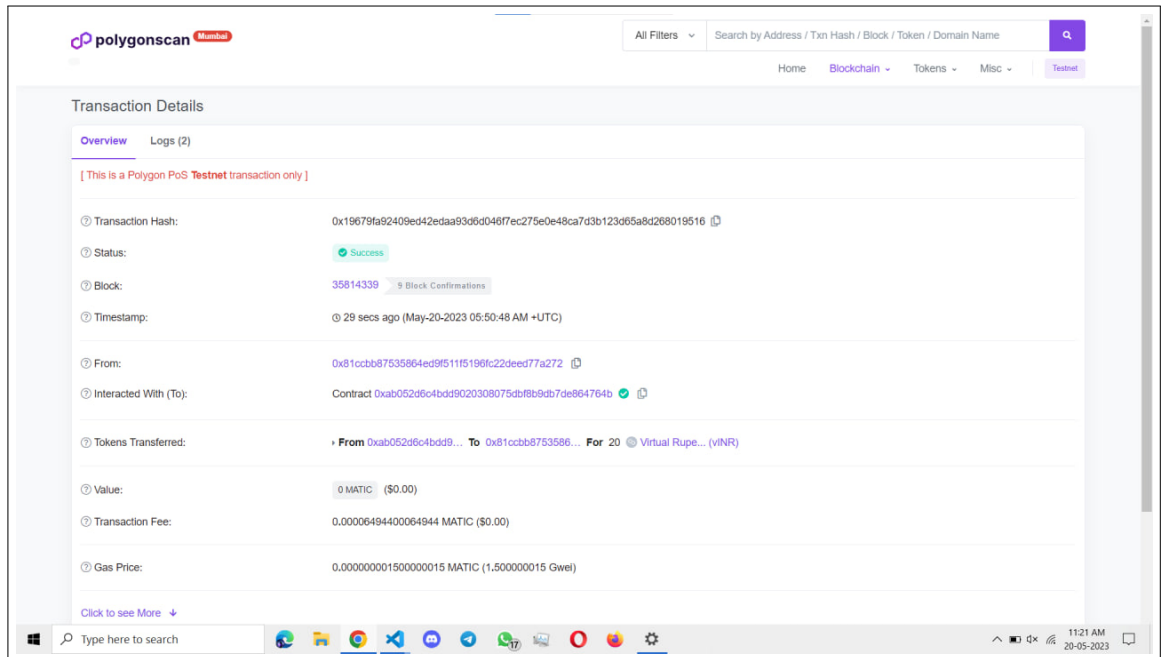


Figure 9.13: Payment withdrawal

9.3.5.3 Transaction History on Smart Contract

The screenshot shows the PolygonScan interface for a smart contract. The contract address is 0xab052d6c4bd9020308075dbf8b9db7DE864764b. The contract overview shows a balance of 0 MATIC and a token value of \$0.00. The transaction history table lists 25 transactions, including methods like Claim, Approve Ride, Accept Ride, Request Ride, Cancel Ride, Update User, and Create User. Each transaction includes details such as Txn Hash, Method, Block, Age, From, To, Value, and Txn Fee.

Txn Hash	Method	Block	Age	From	To	Value	[Txn Fee]
0x19679fa92409ed42ed...	Claim	35814339	1 min ago	0x81cabb87535864ed9f...	IN 0xab052d6c4bd902030...	0 MATIC	0.000064944
0x6a91bb298ace3da0ee...	Approve Ride	35814177	7 mins ago	0x2f1b87c0ee11e810b8...	IN 0xab052d6c4bd902030...	0 MATIC	0.00102749101
0x48ba88095b28e7a86...	Accept Ride	35814017	12 mins ago	0x81cabb87535864ed9f...	IN 0xab052d6c4bd902030...	0 MATIC	0.001594498515
0xb0f3c3483d90824232...	Request Ride	35813725	23 mins ago	0x2f1b87c0ee11e810b8...	IN 0xab052d6c4bd902030...	0 MATIC	0.001761528017
0xfeb346e2af7572f56f02...	Cancel Ride	35795470	11 hrs 9 mins ago	0x2f1b87c0ee11e810b8...	IN 0xab052d6c4bd902030...	0 MATIC	0.000872262009
0x3f09750f1f55a679484...	Request Ride	35701195	2 days 19 hrs ago	0x6542a0a15922e420b1...	IN 0xab052d6c4bd902030...	0 MATIC	0.002458687526
0xe91fea44dbec0be836...	Update User	35701088	2 days 19 hrs ago	0x6542a0a15922e420b1...	IN 0xab052d6c4bd902030...	0 MATIC	0.0000883253
0x9e255be6c86d85cb74...	Create User	35701032	2 days 19 hrs ago	0x6542a0a15922e420b1...	IN 0xab052d6c4bd902030...	0 MATIC	0.000438932004

Figure 9.14: Transaction History on Smart Contract

Chapter 10

Deployment and Maintenance

10.1 Installation and un-installation

10.1.1 Frontend Deployment

- The frontend of the application will be hosted on the Vercel platform.
- Next.js and TypeScript are used for frontend development.
- Use the Vercel CLI or connect the project repository to Vercel for deployment.
- Set up a production environment configuration with necessary environment variables.
- Securely store and limit access to sensitive information.
- Configure automatic deployments triggered by changes pushed to the main branch or specific release tags.
- Regularly monitor deployment logs and error reports.
- Set up monitoring and alerting systems for issue and downtime notifications.

10.1.2 Smart Contract Deployment

- Deploy the smart contract on the Mumbai Polygon network.
- Use MetaMask as the Ethereum wallet client and connect it to the Mumbai test network.
- Compile the smart contract using a Solidity compiler to generate bytecode and ABI.
- Deploy the smart contract using a suitable development framework (e.g., Hardhat or Truffle).
- Test the deployment script locally and address any issues.
- Note down the contract address and relevant information.
- Securely store the deployed smart contract's metadata (address and ABI) for future integration with the frontend.

10.1.3 Maintenance

- Regularly monitor the deployed frontend for issues, bugs, and performance bottlenecks.
- Conduct thorough testing, including functional, integration, and regression testing.
- Keep frontend dependencies (Next.js, TypeScript, Tailwind CSS, etc.) up to date.
- Stay informed about updates and changes in the Mumbai Polygon network and blockchain infrastructure.
- Conduct periodic security audits of smart contract code, frontend application, and integrated libraries/services.
- Follow best practices and guidelines for smart contract development and web application security.
- Implement a robust backup and disaster recovery strategy for both frontend and smart contract.
- Set up monitoring and alerting systems for security incidents, performance issues, and service disruptions.
- Collect user feedback and analyze application usage patterns for improvement and new feature development.
- Plan regular software updates and feature enhancements based on feedback, market trends, and business requirements.

10.2 User Help

1. Getting Started

- Visit the website of the service on your preferred device.
- Sign up or log in to create an account.

2. Location and GPS Services

- Ensure that your device's GPS is enabled to allow accurate location tracking.
- Grant the website permission to access your device's location for seamless navigation.

3. Booking a Service

- On the website, enter your current location.
- Choose your desired destination by either typing it in or selecting it on the map.
- Review the estimated cost and confirm your booking.

4. Crypto Payments

- To make a payment, ensure you have a compatible crypto wallet with funds.
- Link your crypto wallet to the website using Metamask integration.
- During the payment process, choose the crypto payment option and follow the prompts to complete the transaction securely.

5. Frontend and Website

- The website is built using Next.js technology, providing a smooth user experience.
- Explore the website to learn more about the service and its features.

6. Smart Contract and Blockchain Integration

- The website utilizes a smart contract developed and deployed using Solidity.
- The smart contract manages payment transactions and ensures secure and transparent operations.
- The frontend communicates with the smart contract using RPC Ether.js.

Chapter 11

Conclusion & Future Scope

11.1 Conclusion

In summary, the development team has successfully created a platform with advanced functionalities. The website incorporates GPS functionality, user location tracking, and seamless integration with Google Maps. Notably, the team has implemented Metamask integration for secure and convenient cryptocurrency payments. Developed using Next.js, the website ensures a robust and efficient user experience. In addition, the team has designed and deployed a smart contract using Solidity, enabling secure and transparent transactions. The frontend of the website seamlessly communicates with the smart contract using Ether.js, facilitating smooth interaction with the Ethereum blockchain. Overall, this impressive combination of technologies and integrations establishes a solid foundation for a decentralized and modern platform with cryptocurrency payment capabilities.

11.2 Future Scope

1. Ride Sharing: Implement a ride-sharing feature that enables users to find and join shared rides with others heading in the same direction. This feature promotes cost-sharing and reduces traffic congestion by encouraging carpooling, making transportation more economical and eco-friendly.
2. Multi-Language Support: Enhance the app's accessibility and user reach by incorporating multi-language support. By allowing users to choose their preferred language, you can provide a localized experience, catering to users who are more comfortable interacting with the app in languages other than English.
3. Advanced Payment Options: Expand the range of payment options beyond cryptocurrency integration. Integrate additional payment methods such as credit/debit cards, digital wallets, and even a rewards system. By offering diverse payment options, you can accommodate various user preferences and increase convenience for a wider user base.

Annexure A

Referances

1. Tushar S Menon, Aviral Srivastava, Aditya, Dr. Radhika K R, "Ridesharing DApps - A Study on Peer-to-Peer Ridesharing on Ethereum", International Journal of Science and Research (IJSR), [2022]
2. David Schuff, Robert St Louis, "Centralization vs. Decentralization of Application Software" in Research Gate [2021]
3. Ali Dorri, Marco Steger, Salil S. Kanhere, and Raja Jurdak, "Blockchain: A distributed solution to automotive security and privacy", Institute of Electrical and Electronics Engineers [2019]
4. "Filecoin: A cryptocurrency operated file storage network", [2016]
5. Towards blockchain-based intelligent transportation systems, Institute of Electrical and Electronics Engineers (IEEE), [2019]
6. Surbhi Dhar, Sandra Arun, Vivek Dubey, Nilesh Kulal, "App for Ride Sharing", International Research Journal of Engineering and Technology (IRJET) [2020]
7. Mohamed Baza, Nouredine Lasla, Mohamed Mahmoud (Member IEEE), Gautam Srivastava (Senior Member, IEEE), and Mohamed Abdallah (Senior Member, IEEE), B-Ride: Ride Sharing with Privacy-preservation, Trust and Fair Payment atop Public Blockchain, International Research Journal of Engineering and Technology (IRJET) [2019]
8. Panchalika Pal, Indian Statistical Institute Kolkata-700108, India, "BlockV: A Blockchain Enabled Peer-Peer Ride Sharing Service" [2019]
9. Sathya A. Renu*, Barnali Gupta Banik, "Implementation of a Secure Ride-Sharing DApp Using Smart Contracts on Ethereum Blockchain", International Information and Engineering Technology Association (IIETA) [2021]
10. Harsh Agrawal, Piyush Agrawal, Sumit Mali, Apurva Joshi, Ajinkya Ghorpade, "POOL: A PEER-TO-PEER RIDE SHARING APP", International Journal of Engineering Applied Sciences and Technology, 2021
11. MYEONGHYUN KIM 1, JOONYOUNG LEE, KISUNG PARK, YOHAN PARK, KIL HOUM PARK 1, AND YOUNGHO PARK (Member, IEEE) in "Design of Secure Decentralized Car-Sharing System Using Blockchain", IEEE [2019]
12. Sarvesh Wadi, Mrunal Shidore, Malhar Surangalikar, Devarshi Talewar, Vedant Savalajkar, "Sarvesh Wadi, Mrunal Shidore, Malhar Surangalikar, Devarshi Talewar, Vedant Savalajkar", IJERT [2022]

Annexure B

Laboratory assignments on Project Analysis of Algorithmic Design

- To develop the problem under consideration and justify feasibility using concepts of knowledge canvas and IDEA Matrix.
Refer [?] for IDEA Matrix and Knowledge canvas model. Case studies are given in this book. IDEA Matrix is represented in the following form. Knowledge canvas represents about identification of opportunity for product. Feasibility is represented w.r.t. business perspective.

I	D	E	A
Increase	Drive	Educate	Accelerate
Improve	Deliver	Evaluate	Associate
Ignore	Decrease	Eliminate	Avoid

Table B.1: IDEA Matrix

- Project problem statement feasibility assessment using NP-Hard, NP-Complete or satisfy ability issues using modern algebra and/or relevant mathematical models.
- input x , output y , $y=f(x)$

Annexure C

Laboratory assignments on Project Quality and Reliability Testing of Project Design

It should include assignments such as

- Use of divide and conquer strategies to exploit distributed/parallel/concurrent processing of the above to identify object, morphisms, overloading in functions (if any), and functional relations and any other dependencies (as per requirements). It can include Venn diagram, state diagram, function relations, i/o relations; use this to derive objects, morphism, overloading
- Use of above to draw functional dependency graphs and relevant Software modeling methods, techniques including UML diagrams or other necessities using appropriate tools.
- Testing of project problem statement using generated test data (using mathematical models, GUI, Function testing principles, if any) selection and appropriate use of testing tools, testing of UML diagram's reliability. Write also test cases [Black box testing] for each identified functions. You can use Mathematica or equivalent open source tool for generating test data.
- Additional assignments by the guide. If project type as Entrepreneur, Refer [?],[?],[?], [?]

Annexure D

Project Planner

Using planner or alike project management tool.

Annexure E

Reviewers Comments of Paper Submitted

E.1 Reviewers Comments of Paper Submitted

(At-least one technical paper must be submitted in Term-I on the project design in the conferences/workshops in IITs, Central Universities or UoP Conferences or equivalent International Conferences Sponsored by IEEE/ACM)

1. Paper Title:
2. Name of the Conference/Journal where paper submitted :
3. Paper accepted/rejected :
4. Review comments by reviewer :
5. Corrective actions if any :

Annexure F

Plagiarism Report

Proximity Evaluation Based Content Copyrighted Branding or trademarks in Images

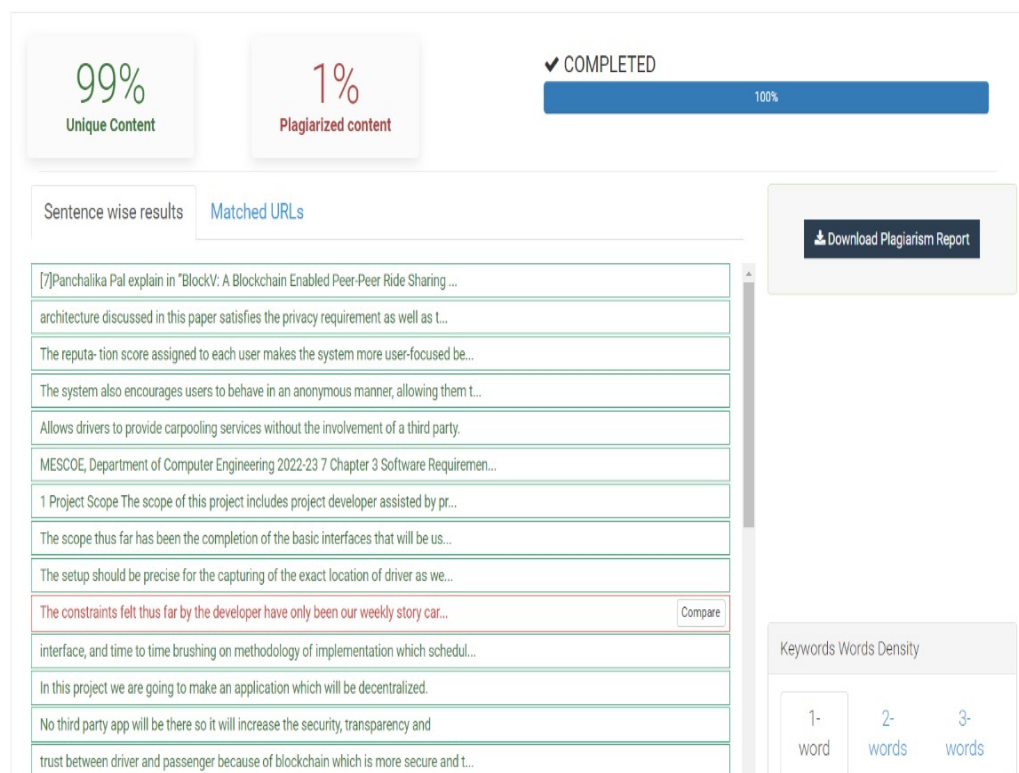


Figure F.1: Plagiarism report

Annexure G

Term-II Project Laboratory Assignments

1. Review of design and necessary corrective actions taking into consideration the feedback report of Term I assessment, and other competitions/conferences participated like IIT, Central Universities, University Conferences or equivalent centers of excellence etc.
2. Project workstation selection, installations along with setup and installation report preparations.
3. Programming of the project functions, interfaces and GUI (if any) as per 1 st Term term-work submission using corrective actions recommended in Term-I assessment of Term-work.
4. Test tool selection and testing of various test cases for the project performed and generate various testing result charts, graphs etc. including reliability testing.
Additional assignments for the Entrepreneurship Project:

5. Installations and Reliability Testing Reports at the client end.

Annexure H

Information of Project Group Members



photo.jpg

1. Name : Akash Uttam Maher
2. Date of Birth : 30/10/2000
3. Gender : Male
4. Permanent Address : Aurangabad, Maharashtra
5. E-Mail : aakashum121@gmail.com
6. Mobile/Contact No. : 8806153745
7. Placement Details :
8. Paper Published :




photo.jpg

1. Name : Pratik Shailendra Bagad
2. Date of Birth : 21/01/2001
3. Gender : Male
4. Permanent Address : Dhule, Maharashtra
5. E-Mail : pratikbagad2001@gmail.com
6. Mobile/Contact No. : 9420968342
7. Placement Details : Placed in Cogitate Technology Solutions
8. Paper Published :



photo.jpg

1. Name : Vedant Yuvraj Pahune
2. Date of Birth : 13/03/2001
3. Gender : Male
4. Permanent Address : Nagpur, Maharashtra
5. E-Mail : vedantpahune@gmail.com
6. Mobile/Contact No. : 9049566848
7. Placement Details : Placed in NTT Data
8. Paper Published :




photo.jpg

1. Name : Rachana Sunil Adhav
2. Date of Birth :
3. Gender : Female
4. Permanent Address : Pune, Maharashtra
5. E-Mail :
6. Mobile/Contact No. :
7. Placement Details :
8. Paper Published :