A Privacy-Preserving Decentralized Aggregated Ride-Sharing Platform Utilizing Blockchain and Differential Privacy

by

Himika Tasnim
21201178
Tahsin Tasnim
22101198
Arnob Sarker Partho
22101003

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

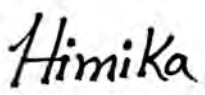
Department of Computer Science and Engineering
Brac University
October 2025

Declaration

It is hereby declared that

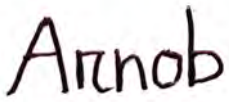
1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Himika Tasnim
21201178

Tahsin Tasnim
22101198



Arnob Sarker Partho
22101003

Approval

The thesis titled "A Privacy-Preserving Decentralized Aggregated Ride-Sharing Platform Utilizing Blockchain and Differential Privacy " submitted by

1. Himika Tasnim (21201178)
2. Tahsin Tasnim (22101198)
3. Arnob Sarker Partho (22101003)

of Summer, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science in October 10, 2025.

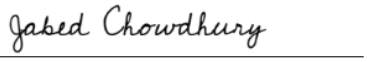
Examining Committee:

Supervisor:
(Member)

Dr. Md Sadek Ferdous

Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)



Dr. Mohammad Jabed Morshed Chowdhury

Senior Lecturer
Cyber Security Program
La Trobe University, Australia

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi

Associate Professor & Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Ride-sharing platforms have emerged as significant components in modern urban mobility providing flexible, cost-effective, and efficient transportation solutions. With a growing user base, these platforms have access to the location data of millions of users and this data can be used in predicting traffic conditions and enhancing the overall ride-sharing experience. However, the traditional ride-sharing platforms rely solely on their own data, overlooking the broader traffic conditions shaped by multiple service providers (SPs), which highlights the need for aggregated systems to get more accurate traffic insight. However, aggregated platforms are vulnerable to critical privacy threats because of their dynamic and interconnected nature, creating a broad attack surface, hence publishing data while ensuring data privacy becomes a significant challenge here. Additionally, traditional ride-sharing platforms rely on centralized systems which are associated with the risk of single points of failure, and potential attacks from malicious participants. To address these limitations, this paper presents a decentralized and privacy-preserving aggregated ride-sharing framework that integrates Blockchain, Differential Privacy (DP), and OAuth-based access delegation. The aggregated platform provides information of all available drivers and active search requests in a specific area, gathered from multiple SPs. The use of blockchain helps to establish an immutable, verifiable, and tamper-resistant control layer for managing users' metadata, while hashed or encrypted operational data is maintained in an off-chain database for efficiency. DP is used to share location data between SPs and aggregators in an privacy preserving, yet utility-retaining manner, while real data is exchanged only when the user selects a verified SP for booking, ensuring information sharing occurs exclusively within consent-based, OAuth-secured transactions. In this proposal, this research aim to bridge together blockchain with DP and explore it as a novel solution to provide two layers of security and privacy, paving the way for a decentralized and privacy preserving aggregated ride-sharing platform. By showing all SPs' available drivers and nearby users, this aggregated approach enhances ride-matching efficiency and promotes a more comprehensive and accurate traffic and demand prediction system.

Keywords: Aggregated System, Differential Privacy, Blockchain, OAuth.

Table of Contents

Declaration i

Approval ii

Abstract iv

Acknowledgment v

Table of Contents v

List of Figures ix

List of Tables x

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Motivation . . . 2
  - 1.2.0.1 Motivation of Aggregated Platform . . . 2
  - 1.2.0.2 Motivation of using Differential Privacy (DP) . . . 2
  - 1.2.0.3 Motivation of using Blockchain . . . 2
- 1.3 Problem Statement . . . 3
- 1.4 Research Objectives . . . 3
- 1.5 Methodology . . . 4
- 1.6 Scopes and Challenges . . . 4
- 1.7 Report structure . . . 4

2 Literature Review 6
- 2.1 Preliminaries . . . 6
  - 2.1.1 Blockchain . . . 6
    - 2.1.1.1 History of Blockchain . . . 6
    - 2.1.1.2 Description of Blockchain . . . 7
    - 2.1.1.3 Application of Blockchain . . . 7
  - 2.1.2 DP . . . 8
    - 2.1.2.1 History of DP . . . 8
    - 2.1.2.2 Description of DP . . . 8
    - 2.1.2.3 Application of DP . . . 9
  - 2.1.3 OAuth 2.0 . . . 9
    - 2.1.3.1 History of OAuth . . . 9
    - 2.1.3.2 Description of OAuth 2.0 . . . 10

2.1.3.3 Application of OAuth . . . 10
2.2 Review of Existing Research . . . 11
2.2.1 Blockchain-Based Approaches . . . 11
2.2.2 DP-Based Approaches . . . 13
2.2.3 Integration of Blockchain and DP . . . 14
2.3 Summary of Key Findings . . . 15

3 Requirements, Impacts and Constraints 17
3.1 System Proposal . . . 17
3.2 System Specification and Requirement . . . 17
3.2.1 Threat Modeling . . . 17
3.2.2 Requirements Analysis . . . 19
3.2.2.1 Functional Requirements . . . 19
3.2.2.2 Privacy Requirements . . . 19
3.3 Societal Impact . . . 20
3.4 Environmental Impact . . . 20
3.5 Ethical Issues . . . 20
3.6 Economic Analysis . . . 21
3.7 Project Management Plan . . . 21
3.8 Risk Management . . . 22

4 Proposed Methodology 23
4.1 Methodology Overview . . . 23
4.2 System Architecture . . . 24
4.2.1 User . . . 24
4.2.2 SP . . . 25
4.2.3 DApp . . . 25
4.2.4 Blockchain . . . 26
4.3 Protocol Flow Design . . . 26
4.3.1 Data Model . . . 27
4.3.1.1 Registration . . . 28
4.3.1.2 Login . . . 28
4.3.1.3 Permission Management . . . 28
4.3.1.4 Aggregated Query . . . 28
4.3.1.5 Booking . . . 29
4.3.1.6 Permission Revocation . . . 29
4.3.2 Protocol Flow . . . 29
4.3.2.1 Registration Protocol . . . 30
4.3.2.2 Login Protocol . . . 31
4.3.2.3 Access Control Protocol . . . 32
4.3.2.4 Dynamic Mobility Discovery Protocol . . . 35
4.3.2.5 Booking Protocol . . . 36

5 Implementation 38
5.1 Implementation of Registration Protocol . . . 38
5.1.1 Secure Personal Data Storage . . . 39
5.1.2 Email Verification . . . 39
5.1.3 Blockchain-based Flag Storage . . . 39
5.2 Implementation of Login Protocol . . . 40

5.3 Implementation of Access Control Protocol . . . 40
5.3.1 Permission Grant Protocol . . . 40
5.3.1.1 OAuth 2.0-based Authorization . . . 40
5.3.1.2 Blockchain-based Flag Storage . . . 41
5.3.1.3 Secure Token Storage in the Database . . . 41
5.3.2 Permission Revoke Protocol . . . 42
5.4 Implementation of Dynamic Mobility Discovery Protocol . . . 42
5.4.1 DAPP Session Initialization of DApp . . . 42
5.4.2 DP Application in DApp . . . 43
5.4.2.1 Noise Generation Using Geo-Indistinguishability . . . 43
5.4.2.2 Selection of the Privacy Budget (ϵ) . . . 44
5.4.3 SP Response Generation . . . 44
5.4.3.1 SP Session Initialization . . . 45
5.4.3.2 Approximate Fare Calculation . . . 45
5.4.3.3 Applying DP in SP . . . 46
5.4.3.4 Sending Response to DApp . . . 46
5.4.4 Aggregation and Visualization . . . 46
5.5 Booking . . . 47

6 Result Analysis and Discussion **51**
6.1 Performance Evaluation . . . 51
6.1.1 API Testing Tool . . . 51
6.1.2 API Testing Approach . . . 51
6.1.2.1 System Performance Under Increasing Load . . . 52
6.1.2.2 System Performance Under Sudden Traffic Surges . . . 52
6.1.3 System Configuration for Testing . . . 52
6.2 Test Set Up . . . 52
6.2.1 GTIT Objective . . . 52
6.2.2 ST Objective . . . 53
6.3 Statistical Analysis . . . 53
6.3.1 Response Time Trends and Bottlenecks Under Gradual Load . . . 53
6.3.2 Throughput Trends and System Scalability Under Gradual Load . . . 53
6.3.3 Response Time Trends Under Sudden Traffic Surges . . . 54
6.4 Final Design Adjustment Based on Performance Evaluation . . . 55
6.4.1 Storage Management . . . 55
6.4.2 Scalability Enhancements . . . 56
6.4.3 API Optimization . . . 56
6.5 Analysis of Design Solutions . . . 56
6.5.1 Mitigation Strategies for Identified Threats . . . 56
6.5.1.1 Linkability (T1) . . . 56
6.5.1.2 Identifying (T2) . . . 57
6.5.1.3 Disclosure of Data (T3) . . . 57
6.5.1.4 Unawareness and Unintervenability (T4) . . . 57
6.5.1.5 Non-compliance (T5) . . . 57
6.5.1.6 Session Integrity (T6) . . . 58

6.5.2 Functional Requirements Implementation 58
6.5.2.1 Decentralized Registration and Identity Binding . . 58
6.5.2.2 Consent-Driven Access Control 59
6.5.2.3 OAuth-Based Token Management 59
6.5.2.4 Session-Oriented Query Processing 59
6.5.3 Privacy and Security Requirements Implementation 59
6.5.3.1 DP for Data Protection 59
6.5.3.2 Secure Data Sharing 60
6.5.3.3 Transparent Consent Mechanisms 60
6.5.3.4 Regulatory Compliance 60
6.5.3.5 Secure Session Management 60
6.5.4 Achievement of Research Objectives 60
6.5.4.1 Development of Aggregated Ride-Sharing Platform . 60
6.5.4.2 Publication of Traffic Insights with Privacy Protection 61
6.5.4.3 Decentralized Consent Management 61
6.5.4.4 Performance and Functionality Evaluation 61
6.6 Comparisons and Relationships 61

7 Conclusion **63**
7.1 Summary of Findings 63
7.2 Contributions to the Field 63
7.3 Recommendations for Future Work 64

Bibliography **68**

List of Figures

4.1 Methodology 23
4.2 System Architecture 25
4.3 Registration Flow 31
4.4 Login Flow 32
4.5 Permission Grant Flow 33
4.6 Permission Revoke Flow 34
4.7 Dynamic Mobility Discovery Flow 36
4.8 Booking Flow 37

5.1 MetaMask Prompt During Registration 39
5.2 Visualization of Map Generated During Dynamic Mobility Discovery 47
5.3 Comparison of Distance While Searching and Booking 48

6.1 Gradual Load Testing 54
6.2 Spike Testing 55

List of Tables

2.1 Comparative Analysis of Existing Papers and Our Proposed System . 16

3.1 Cost Breakdown of Blockchain Transaction 21
3.2 Research Methodology Phases 22

4.1 Cryptographic and other Notations 27
4.2 Data Model 28
4.3 Registration Protocol 30
4.4 Login Protocol 32
4.5 Permission Granting Protocol 33
4.6 Permission Revocation Protocol 34
4.7 Dynamic Mobility Discovery Protocol 35
4.8 Booking Protocol 37

5.1 Average Noise with Respect to Different ϵ Values 45

Chapter 1

Introduction

1.1 Background

Ride-sharing platforms including Uber and Lyft are gaining popularity as a way of transportation with the advancement of Web 2.0 [30]. To be more specific, in 2021-2022, more than 1.8 billion trips (2.5 million trips per day, or around 30 rides every second) were completed by Uber only in the US which reflects the increasing demand of ride-sharing platforms [42]. Nevertheless, with the rising demand substantial challenges associated with data privacy, transparency, accuracy and centralization risks have also emerged as key concerns. Primarily, the quality of the location based services depends on the accuracy of the location data of the users. On the contrary, the exposure of precise location information heightens the risk of data privacy being compromised [7]. Therefore, the necessity of research on user's location data while ensuring data privacy of users has become very crucial. During data collection, user consent must be securely obtained and managed and during data publication it should be ensured that the data privacy will not be compromised. The most obvious way to maintain data privacy is anonymity. However, different anonymization techniques showed vulnerabilities. Attackers may re-identify the user even after anonymization by using another dataset of background information or analyzing patterns of the data through different attacks [8]. Moreover, the traditional ride-sharing platforms use centralized systems, but the centralized system has fundamental risks regarding data privacy. The centralization of control can lead to security issues including single points of failure and malicious actors may also be able to attack the system [21]. Additionally, to get more precise predictions about the demand or traffic condition, research on mass data is essential but the existing ride-sharing platforms have access to the data of their dedicated users only which raises the concern of accuracy. To sum up, despite the increasing demand of ride-sharing platforms, the existing systems have vulnerabilities regarding privacy, security, and accuracy. Therefore, a system which can analyze mass data without violating privacy is a challenge. The research focuses on designing an aggregated ride sharing platform which can automatically obtain user consent for data access, store it securely in a distributed ledger without the intervention of a third-party, and enables analysis and publication of user's location data without compromising privacy.

1.2 Motivation

In urban environments, reliance on ride-sharing applications for day to day life commutation is growing exponentially. On the contrary, the traditional ride-sharing platforms are dealing with usability, ethical issues, and also rising user data privacy concerns. Research to develop a decentralized, aggregated, and privacy-preserving ride-sharing system was spurred by an urgent requirement for a system that strikes the balance between user convenience and privacy protections.

1.2.0.1 Motivation of Aggregated Platform

An aggregated platform allows users to access the features of several ride-sharing services through a single interface, eliminating the need to manually switch between apps in order to compare prices, availability, and travel times. Furthermore, it introduces competition among service providers (SPs), potentially leading to better service quality and pricing as well. For instance, if a user in Dhaka is trying to find a ride using Uber, he may not be able to book a preferred ride on time due to the volume of requests on Uber from that specific area. Then he might need to switch to other apps like Pathao, Obhai to book a ride. Since the prices, driver availability, and wait times may vary in different apps, it becomes difficult for the user to choose an app that will be suitable according to his preference. This not only consumes time but also causes user dissatisfaction. In this scenario, we propose an aggregated app which can enhance user experience by allowing users to avail all the facilities using one single app .

1.2.0.2 Motivation of using Differential Privacy (DP)

When users search for rides in our DApp, it shows fare approximation, available rides, number of booking requests collected from multiple SPs. Though this aggregation is necessary for accurate traffic inside retrieval but it also raises a serious privacy risk due to the involvement of sensitive geolocation data of the users. To mitigate this, our system suggests integration of DP engine with the backend of ride sharing SP and the DApp. DP engine adds statistical noise to the location data which prevents revealing of exact location coordinates. It applies when a user's current location and destination is being sent to the SPs from DApp and also when the SPs send location data of other users and drivers to the DApp. This is how DP provides a mathematical guarantee of privacy, making it an important component in preserving user privacy without compromising utility of data. Furthermore, the growing focus on data protection regulations, DP serves as a commitment to secure data handling practices along with legal compliance.

1.2.0.3 Motivation of using Blockchain

Blockchain introduces trust, transparency, data integrity and auditability in our decentralized aggregated platform. By keeping the records of registrations, permission grants and revocation it eliminates the disadvantages of depending on a central authority. Additionally, it increases user trust and provides users with fine-grained control over their data sharing preferences. As the transactions are immutable and resistant to tempering it enables verifiable audit trails. Both the SPs and users

can trace permission flows and data access events, which ensures that data-sharing agreements between users and SPs are tamper-proof and transparent. Integration of blockchain in the system enhances privacy and accountability across the platform.

1.3 Problem Statement

Due to the rising demand of ride sharing services, numerous independent ride-sharing platforms have emerged to meet diverse user needs. But most of the time, users experience frustration when navigating between multiple ride-sharing applications to compare fares, availability, and service quality in order to make an informed decision. To bring all the facilities under the same platform data aggregation is required, but if location data is transmitted without adequate protection it increases the risk of exposing users to potential data misuse, surveillance, or re-identification attacks. The lack of privacy-preserving mechanisms not only undermines user trust but also violates data protection laws and regulations. Additionally, the centralized control over user registration, data access, and consent management gives users limited visibility and control over how their data is shared, stored, or utilized by third-party services which risks transparency, accountability, and auditability.
This research aims to address the following key challenges in the current ride-sharing ecosystem:

- Absence of privacy preserving unified platform to compare and avail services of multiple providers.
- Exposure of sensitive user location data without privacy guarantee.
- Centralized consent management with limited user control and auditability.

There is no existing ride-sharing platform that effectively integrates data aggregation, effective privacy-preserving techniques, and decentralized data control within a single unified system. Our proposed solution combines data aggregation, DP, and blockchain to overcome these limitations and provide a secure, transparent, and convenient ride-sharing platform.

1.4 Research Objectives

RO1: To develop an aggregated ride-sharing platform so that research and analysis can be conducted using the data of mass users from multiple platforms.

RO2: To enable analysis of users' location data and publication of traffic insight without compromising privacy for the improvement of demand and traffic condition prediction.

RO3: To develop a decentralized and optimized system for user consent management to enhance transparency and auditability.

RO4: To assess the performance and functionality of the proposed system.

1.5 Methodology

Our research is structured into multiple steps, each contributing to the progressive development and refinement of the proposed system. Our research had begun with the identification of the key limitations of existing ride-sharing platforms through real-world observation and background study and based on these insights we proposed a prototype. To build a real-life application we followed multiple steps beginning from threat modeling including requirement analysing, architecture design, protocol flow design, implementation, and ultimately concluding with a comprehensive evaluation of the system.

1.6 Scopes and Challenges

The scope of our research focuses on building a decentralized and privacy-preserving aggregated ride-sharing platform. Our securely system integrates approximate fare, ride availability, booking requests data from multiple SPs. Our research focused on ensuring that a user can make informed decisions using the aggregated data while ensuring data privacy of other users and having full control over their personal data. In our system, we redirect users to the particular SP's interface but real-time ride dispatching after redirection or driver-side logistics which are managed by individual SPs fall outside of our research scope. Additionally, we also do not address the internal data privacy practices and compliance mechanisms of third-party SPs and user-side device vulnerabilities. These aspects can be considered in future extensions beyond the current research scope.

For implementation of the system we include a combination of Web3 and blockchain based user registration, DP for location data and blockchain-based consent management. However, technical difficulties may arrive while incorporating multiple tools. Integration of diverse ride-sharing services' APIs into a standardized aggregation framework is a key challenge. Additionally, while applying DP, maintaining balance between data utility and privacy requires careful calibration. Furthermore, performance-related limitations may arise utilizing blockchain transactions in a scalable and low-latency manner. This challenges will be addressed by our research.

1.7 Report structure

- In Chapter 2, the background of Blockchain and DP will be discussed. Additionally, we delve into the previous research works related to our paper in this chapter. Here, existing studies will be evaluated to identify the research gap.
- In Chapter 3, our system proposal along with the tools and technology used to develop the system is introduced.
- In Chapter 4, detailing of the underlying threats along with the system's functional and privacy requirements is added. This chapter also gives a comprehensive overview of the system architecture and protocol flow of the suggested system.

- In Chapter 5, implementation procedure of the mentioned protocols including the associated algorithms, system components, tools are described.
- In Chapter 6, the performance evaluation along with the analysis of system's overall functionality and privacy preservation.
- Finally, Chapter 7 concludes the report presenting the summary of the research outcome as well as its limitations and future work.

Chapter 2

Literature Review

2.1 Preliminaries

2.1.1 Blockchain

Blockchain is an emerging technology which has introduced a new dimension in securely storing data or transactions. It is a decentralized and distributed ledger technology which solves many problems in the industrial environment ensuring trust, transparency, security and reliability of data processing [19].

2.1.1.1 History of Blockchain

Blockchain technology was introduced by Satoshi Nakamoto, whose true identity is still unknown, through the publication of "Bitcoin: A Peer-to-Peer Electronic Cash System" in 2008 and subsequently in January 2009 Bitcoin's open-source software was also released [25]. Bitcoin aims to eliminate the need for third-party intermediaries in the financial sector by designing a decentralized and cryptographically secure digital currency. Unlike other decentralized payment systems, which have failed previously, blockchain provides secured foundation by ensuring immutability, trust and transparency in transaction through cryptographic proof. Though bitcoin and blockchain were used as synonyms previously, over time, these two became distinct concepts with blockchain being the underlying technology of bitcoin for digital payment processing. The open-source nature of bitcoin encourages developers to create bitcoin-like projects and with time the use of blockchain spreads beyond cryptocurrency.

In 2013, Vitalik Buterin proposed Ethereum. Later in 2014, it was crowdfunded and in 2015 it was launched [24]. Ethereum introduced decentralized applications and smart contracts - digital autonomous contracts programmed on the blockchain. This advancement of blockchain is also known as Blockchain 2.0. Furthermore, between 2020 and 2022 Ethereum 2.0 was introduced which aims to enhance the network's speed, security, and scalability [35]. In 2015, with the announcement of the Hyperledger project by the Linux Foundation, the concept of permissioned blockchain was introduced. In Hyperledger Fabric [18], the participants need to be verified before joining the network and it has different frameworks from the permissionless blockchain, Ethereum and Bitcoin [35]. Because of the rapid growth of blockchain, it's usage as extended far beyond the sphere of cryptocurrency. In

recent times, blockchain is widely used in numerous decentralized services including finance and hanking, healthcare, identity management systems, transportation and in many diversified decentralized systems.

2.1.1.2 Description of Blockchain

The key characteristics of blockchain include decentralization, data immutability, persistence, audibility, and pseudo-anonymity. To ensure these features blockchain uses some core technologies, such as digital signatures, cryptographic hash, consensus algorithms, and smart contracts.

Blockchain uses cryptography and collaboration to create that trust which eliminates the need for centralized third-party intermediaries. Public Key Cryptography, Zero-Knowledge Proof, Hash Functions are cryptography building blocks used by blockchain [35]. Because of being a decentralized system, a single entity or node doesn't control the whole system, rather when a transaction is initiated, it is broadcasted to a peer-to-peer network and using different consensus algorithms nodes (network participants) validate the transactions. Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof-of-Stake (DPoS), Proof of Elapsed Time (PoET), Practical Byzantine Fault Tolerance (PBFT), Directed Acyclic Graph (DAG) are some commonly used consensus algorithms [35]. After verification, valid transactions are bundled in a block. These blocks are added chronologically in the ledger and each of them contains the hash of the previous subsequent block making blockchain immutable [23]. Moreover, hackers will need to modify the whole chain to change a single entry which makes it fraud resistant, and diables unauthorized modifications [23]. After verifying a block, it is added to the blockchain. In this system, all the nodes have a copy of the ledger and it ensures all the nodes maintain a common state of distributed ledger making it synchronized and tamper-proof. In blockchain, each transaction is stored with a timestamp and so, users are able to check and trace prior records by accessing any node in the distributed network which ensures auditability [37]. Blockchain also ensures pseudo-anonymity. In this structure, rather than using the directly real-world identity of users, they are represented by cryptographic addresses. Although these addresses do not reveal personal details, using advanced analytics or linking external data sources users can potentially be de-anonymized since all transactions are traceable.

With the advancement of blockchain smart contracts, computer protocols or codes where logics can be executed, are introduced in the system. This is designed to autonomously facilitate, verify, and enforce the negotiation and agreement among multiple untrustworthy parties. Smart contracts are used in several blockchain-based development platforms including Ethereum , Hyperledger Fabric to automatically execute events and actions [32].

2.1.1.3 Application of Blockchain

Blockchain was first used to enable decentralized transactions with cryptocurrencies like Bitcoin. Blockchain based transaction systems offer low cost, fast transactions, and secured peer-to-peer payment [35], [37]. Moreover, due to the transparency and traceability of the blockchain system, its use in the supply chain has increased.

Permissioned blockchain-based data sharing solutions that are released by IBM is an indication of the growing utilization of blockchain. Additionally Dubai is taking initiative to transform the role of government utilizing blockchain [35]. Apart from this, blockchain is also widely used in diversified decentralized applications which includes crowdfunding, Internet-of-Things (IoT), reputation system (web community, academics, etc.), security and privacy (security enhancement, risk management, privacy protection, etc.), healthcare, insurance, digital records, intrusion detection, digital ownership management and in many more sectors [23].

2.1.2 DP

DP is a privacy-preserving technique that guarantees information of an individual that cannot be identified with the background information of all other data points.

2.1.2.1 History of DP

In recent years, digital data has become a vital component for ensuring better services. Especially with the development of machine learning, the data collection has increased drastically for research purposes by profit and non-profit organizations and even by governments which has led to massive datasets. Nevertheless, with the increased amount of dataset, privacy has also become a prime concern.

There are roughly two stages in the data publishing and analysis process - data collection, where personal data of the users are collected by the curators, and data publishing or analysis, where the data is analyzed and datasets or the results are shared to public users. Privacy risk arises at both stages - an untrustworthy curator may leak sensitive data of users, anonymized data may be re-identified after publication if the anonymization technique is simple [16]. To preserve privacy in the data collection stage, several privacy models are utilized including k-anonymity [2], l-diversity [4], ı-presence [5], and t-closeness [3]. But the main weakness of these models is the attackers may re-identify anonymized data utilizing background information. Minimality attacks [42], composition attacks [6], deFinetti attacks[20], and foreground knowledge [10] attacks can be launched in this regard.

With the motivation to resolve the vulnerabilities of existing models, DP emerged in 2003 and it was formally introduced in 2006. DP is a promising privacy-preserving model which guarantees that even if the adversary gets to know n-1 background information, he/she cannot infer n-th record which he/she actually wants to identify. This privacy preserving technique is utilized in diversified research areas including privacy community and data science communities (machine learning, data mining, statistics) [42].

2.1.2.2 Description of DP

DP ensures that no personal information can be linked to the background data of any other data point. In Dp controlling the balance between privacy and usability is very crucial and for that the privacy budget (ϵ) is used. The lower value of the privacy budget indicates the stronger privacy. In DP, a privacy mechanism M gives (ϵ, δ) DP for every set of outputs S, for a dataset D and its neighbouring

dataset D' (a dataset that differs from the original one by a single data point), if M satisfies Equation (1), where Pr denotes probability and δ denotes the probability of violating (ϵ, δ)-DP [16],[43].

$$Pr[M(D) \in S] \leq \exp(\epsilon) \cdot Pr[M(D') \in S] + \delta. \tag{1}$$

To ensure that queries on a dataset do not compromise the privacy of individuals, random noise needs to be added to the dataset and the required noise for each dataset depends on sensitivity which is defined by Equation (2). It refers to the change in output if one data point is added or removed. If the sensitivity of the dataset is higher, more noise needs to be added to ensure privacy [16],[43].

$$\Delta f = \max_{D,D'} \|f(D) - f(D')\| \tag{2}$$

To gurantee DP, three different mechanisms are widely used. Among them the Laplace mechanism and the Gaussian mechanism are used for for numeric queries and the exponential mechanism is used for non-numeric queries [16]. Moreover, for location privacy Geo-Indistinguishability, a specialized form of Laplace-based DP, is the most suitable approach that is designed to geographic coordinates while preserving data utility [11]. In this system, Geo-Indistinguishability, a local DP mechanism, will be applied since the system works with only geographic coordinates.

2.1.2.3 Application of DP

With the advancement of technology, a massive amount of data is stored in databases for further analysis both locally and on the cloud. To ensure data privacy without compromising usability of the data, DP has emerged as a powerful tool. In the healthcare sector, all the personal sensitive information is stored in databases and this data is accessible by doctors, medical staff, and often third-party for research purposes. To ensure the privacy of the patient, DP can be used [29]. Moreover, in location-based services including ride-sharing platforms, DP has been applied to minimize privacy risks without affecting the quality of services [41]. Additionally, in the research where machine learning models need to be trained on sensitive data, DP can be an effective tool. To sum up, DP can be applied to publish mass dataset ensuring that any individual cannot be re-identified including census data, utility usage, traffic patterns, and other public statistics.

2.1.3 OAuth 2.0

OAuth 2.0 is an authorization framework that enables a user to delegate controlled access to their data without providing sensitive credentials.

2.1.3.1 History of OAuth

Open Authorization (OAuth) was first proposed in 2006 by Blaine Cook and Chris Messina while discussing the need for a standardized authorization protocol. The goal was to allow third-party applications to gain limited access to user data without requiring them to share their credentials. The OAuth Core 1.0 was published in 2007, and it was adopted by large platforms such as Twitter and Google. However, OAuth 1.0 relied heavily on cryptographic signatures and had complexities.

Then in 2010, the OAuth 2.0 framework was released, which relied on HTTPS for transport layer security instead of complex signatures. [12] OAuth 2.0 introduced the concept of authorization flows adapted for different types of clients such as web apps, SPAs, mobile apps and emphasized token-based authorization. Thus, OAuth 2.0 has become the industry standard for delegated authorization, enabling secure access delegation in a wide range of applications, from social logins to enterprise systems and API integrations.

2.1.3.2 Description of OAuth 2.0

In Oauth 2.0, instead of directly sharing a username and password, the user grants a third-party application, known as the client, limited access rights by issuing an access token through a trusted authorization server. This token is then presented to the resource server to get the requested resources. The framework ensures that access is time bound and scope restricted, which enable secure permission management. [14]

The OAuth 2.0 specification defines several standardized flows for different application contexts. The authorization code flow provides high security by exchanging a short lived authorization code for tokens. The implicit flow was created for single-page applications, which has been largely replaced by the Proof Key for Code Exchange (PKCE) extension. It strengthens the authorization code flow against interception attacks. [14] The client credentials flow enables machine-to-machine communication where access is granted directly to the client application. The device code flow supports input-constrained devices, such as smart TVs, which cannot do traditional browser based redirections. [53]

OAuth 2.0 introduces two key tokens. The access token is a credential presented by the client to the resource server to get protected resources. This token is typically short-lived and may be formatted as a structured JSON Web Token (JWT). The refresh token allows the client to renew an expired access token without requiring the user to re-authenticate, which improves usability while maintaining security. These tokens enable OAuth 2.0 to achieve secure, delegated access while reducing the exposure of long-term credentials. [14], [53]

2.1.3.3 Application of OAuth

OAuth 2.0 has become the backbone of secure access delegation in modern web and mobile ecosystems. One of the most common applications is social login, where users authenticate with platforms such as Google, Facebook, or GitHub to access third-party websites without creating separate credentials. OAuth is also used in API integrations, enabling enterprises to securely give customer data or services to external providers under fine grained scopes. In the financial sector, OAuth allows customers to grant third-party applications permission to access banking data and initiate payments while maintaining account security.[14], [40]

In our system, OAuth plays a central role in granting permission to SPs. Once a user registers with their blockchain wallet and verifies their email, they can securely grant SPs permission through an OAuth flow. The user is redirected to the SP's

authorization page, logs in with their registered credentials, and upon consent, receives an authorization code. This code is exchanged for access and refresh tokens, which are then stored and used to authorize future API calls. This ensures that sensitive user data remains private.

2.2 Review of Existing Research

2.2.1 Blockchain-Based Approaches

Baza et al. [22] introduced a decentralized ride-sharing service called B-Ride. It is based on public blockchain to address privacy and trust issues which exist in centralized ride-sharing platforms. B-Ride employs smart contracts to facilitate direct transactions between drivers and riders. It removes dependency on a central authority and the risk of a single point of failure. It ensures the confidentiality of trip details, including pickup and drop-off locations by integrating privacy-preserving mechanisms such as spatial cloaking and zero-knowledge set membership proofs. B-Ride also introduces a time-locked deposit protocol to prevent malicious behavior. Both drivers and riders must send a deposit to the blockchain. The driver must prove their arrival at the pick-up location using zero-knowledge proofs. Moreover, the system implements a decentralized reputation management model, which tracks driver reliability. It is based on some metrics like arrival at pickup points and trip completion rates. The authors did a proof-of-concept implementation on the Ethereum blockchain, demonstrating its practicality and cost-effectiveness. B-Ride addresses privacy concerns by ensuring data confidentiality, preserving anonymity, and preventing malicious behavior.

Di Wang and Xiaohong Zhang [28] proposed a secure ride-sharing framework built on a consortium blockchain. To eliminate reliance on a central authority, the system decentralizes the storage and processing of data which mitigates the risk of single-point attacks and data misuse. It employs an improved Delegated Proof of Stake consensus mechanism. It efficiently verifies transactions and ensures the integrity of the blockchain. The framework integrates an attribute-based proxy re-encryption algorithm. This allows passengers to set access structures for their data, ensuring that only authorized entities can access sensitive trip information. The system aims to penalize bad behavior and reward good behavior by giving drivers and passengers performance scores. This framework improves ride-sharing services operational efficiency, security, and privacy by combining cryptographic techniques with the decentralized nature of blockchain technology.

To address privacy and security concerns in cab-sharing systems, Ahmed et al. [39] propose a decentralized and secure cab-sharing system using blockchain technology to eliminate the need for third-party. The suggested solution ensures the confidentiality of sensitive user information, personal details, travel prices, pick-up and drop-off locations, and trip schedules in contrast to traditional systems that depend on centralized authorities and are vulnerable to single points of failure, service fees and privacy risks. In order to promote trust and accountability, the system also integrates a decentralized reputation mechanism that allows drivers and riders to be rated according to their travel history and behaviors. The system is implemented

on the Ethereum blockchain using smart contracts, ensuring secure and transparent interactions between users. The system maintains efficiency and low computational overhead, making it a scalable alternative to traditional cab-sharing platforms.

Mahmoud et al. [38] propose a framework that integrates blockchain technology with the Inter Planetary File System (IPFS) to enhance the scalability, efficiency, and data security of ride-sharing services. The system uses the decentralized storage of IPFS for storing large data such as trip records and user credentials. It addresses issues of limited storage capacity and high costs of on-chain storage by storing metadata on the blockchain and actual data on IPFS. The framework ensures transparency by using smart contracts for ride-matching, payment settlements, and service validation. It implements encryption techniques that protect sensitive data such as pick-up and drop-off locations. Through Ethereum and IPFS, the system improves system efficiency and reduces computational overhead. The integration offers a cost-effective alternative to fully on-chain solutions. Thus, this work combines blockchain and decentralized storage to address challenges in ride-sharing services.

Oksuz [45] introduced a smart-contract-based ride-sharing system using a consortium blockchain which emphasized user privacy and data security. Vickrey auctions are used by the system to guarantee transparent passenger selection and reasonable prices. Driver-passenger interactions are managed by smart contracts which also enforce regulations like collateral deposits to determine fraud and compensate impacted parties in the case a user violates agreements. Whereas traditional systems use cryptographic operations, the framework uses secure randomization techniques to protect user identities and travel data. The decentralized nature of the blockchain eliminates single points of failure, while the transparency of transactions allows for trust without compromising security.

Kato et al. [20] suggested a blockchain-based ride-sharing framework that replaces centralized authorities like Uber to ensure transparency and optimize resource allocation. The system encourages drivers to participate as both drivers and miners by using a special two-coin mechanism that includes a working coin and a mining coin thereby preserving the blockchain system. Drivers get coins based on their activity, which increases the probability of being matched with riders. The transactions are safely stored on the blockchain while drivers and riders are dynamically matched. A case study evaluates the system which shows that drivers benefit more from the proposed system compared to centralized ride-sharing platforms due to transparency, reduced fees, and incentive mechanism.

Kudva et al. [26] proposed a decentralized application built on a consortium blockchain called PEBERS (Practical Ethereum Blockchain-based Efficient Ride-Hailing Service). Ethereum smart contracts are used by the system to control ride-related operations like ride creation, auto deposit transfers, cancellations, and ride completion. It ensures a lightweight architecture as it uses delegated proof-of-stake consensus algorithm. Thus, the system does not need miners and relies on validator nodes to maintain the blockchain's integrity. As semi-trusted agents for localized ride-matching the authors also incorporated fog computing nodes which reduces latency and enhances scalability. According to experimental results, PEBERS offers a

safe, verifiable, and immutable transaction ledger while lowering operating expenses for both drivers and passengers.

Nosouhi et al. [27] proposes a decentralized blockchain-based framework to address location proof mechanisms problems such as location privacy threats and Prover-Prover and Prover-Witness collusions. The system securely verifies and stores spatiotemporal data by utilizing the immutable ledger of blockchain technology. The system uses cryptographic commitments to protect privacy by hiding raw location data and mobile users serve as witnesses to verify the location claims of others. By giving witnesses and verifiers cryptocurrency, an incentive system promotes participation. Because of its decentralized architecture the framework is scalable and cost-effective because it does not require a central authority. Security evaluations show that the scheme is resistant to location spoofing and collusion attacks and experimental findings from an Android prototype show how effective it is in comparison to other LP systems. Future works call for incorporating smart contracts and allowing cryptocurrency-based payments for transaction fee.

Zou et al. [34] proposed CrowdHB, a decentralized location privacy-preserving crowdsensing system based on a hybrid blockchain network. It presents a hybrid blockchain-based framework to improve mobile crowdsensing systems efficiency, security and privacy. By combining public and private blockchains the suggested CrowdHB system solves issues like performance constraints, privacy leakage and inefficiencies in centralized systems. It uses a Consistency Optimization (ACO) method and a Location Privacy-Preserving Optimization Mechanism (LPPOM) to strike a balance between task assignment efficiency and privacy. The system protects user privacy by optimizing query regions and anonymizing worker data using smart contracts and k-anonymity.

2.2.2 DP-Based Approaches

Guo et al. [36] proposed a location-entropy-based DP protocol to improve the privacy of location data in LBS. Laplace noise and location entropy are combined in the system's user-controllable DP mechanism to prevent background knowledge attacks and guarantee user anonymity. Smart contracts for reputation evaluation are incorporated into the protocol to guarantee the integrity of user interactions and deter malicious behavior among participants. The protocol also makes use of Dijkstra's algorithm to optimize query results preserving user privacy and data availability. Experimental findings showed that this strategy balances privacy and usability, offering strong defense against attacks and achieve location privacy by utilizing adaptive anonymity sets.

Hien et al. [13] proposed a framework for differentially private publication of location entropy. The authors created a baseline algorithm that ensures -DP by calculating tight bounds for the global sensitivity of location entropy and injecting noise proportionate to this sensitivity. However, the study introduced optimization techniques such as truncation of user activities and smooth sensitivity methods to balance privacy protection with data utility because the baseline method introduces excessive noise. These enhancements improved privacy-utility trade-offs by

reducing sensitivity. In order to suppress certain locations while publishing entropy for those with adequate user activity the authors also suggested a crowd-blending technique. Tests conducted on both artificial and real-world datasets showed how effective these techniques are at preserving location privacy without sacrificing the usefulness of published data.

Wang et al. [15] propose a novel privacy-protected place of activity mining (P-PAM) approach to find dynamic significant geographic regions (POAs) using extensive spatiotemporal data while maintaining privacy protection. The issues of parameter sensitivity, high computational complexity, and privacy leakage in the current clustering-based POA identification methods are addressed in this work. The authors incorporate DP mechanisms into location entropy evaluations and the clustering process to safeguard user privacy. To both clustering results (e.g., geographic centroids) and location entropy values, noise is added using the Laplace mechanism, guaranteeing -DP. A truncation mechanism is also used in the method to lessen sensitivity which enhances utility while upholding robust privacy guarantees. Experiments conducted on extensive location datasets obtained from geo-tagged social media show that the suggested method effectively finds POAs while guaranteeing high privacy protection levels and minimizing information distortion.

2.2.3 Integration of Blockchain and DP

Sun et al. [33] proposed a two-stage privacy protection mechanism based on blockchain for mobile crowdsourcing using localized DP to improve privacy and security of participants. Firstly, the double disturbance localized DP algorithm adds disturbance factors to the location data of workers to protect their privacy before uploading it to the blockchain. By this mechanism, concerns of location inference by attackers are being addressed. The second stage is integrating blockchain technology. It is used to replace third-party platforms which ensures data integrity and prevents privacy leaks. Edge nodes interact with the blockchain for data storage and a Merkle tree structure enhances data integrity and search efficiency. Beijing taxi trajectory datasets have been used to conduct comparative experiments. It demonstrated the DDLDP algorithm's superior performance in protecting location privacy, reducing service quality loss, and maintaining higher data availability compared to existing approaches like Laplace noise, Gaussian noise, and k-anonymity mechanisms. The paper suggests considering the credibility of workers in the privacy protection model.

Alnemari et al. [17] proposes a novel framework for using the blockchain technology and DP to protect the sensitive infrastructure data. Firstly, this discusses how traditional access control schemes such as Role Based Access Control (RBAC) are ill suited for the protection of privacy or avoiding data breaches. Differentially private query results are layered with blockchain for decentralized immutable access control, for safe data analysis while not losing user privacy, as suggested for the proposed architecture. The framework dynamically adapts access permissions based on user roles and query sensitivity, for example from emergency services and healthcare. It points out possible scalability and performance issues with the study, but provides a solid method for safe data sharing in the critical infrastructure domain.

Fotiou et al. [31] proposed a novel marketplace framework for safely sharing sensitive data. Data providers can independently add noise to their data using local DP which shields them from system operators and data consumers. Using blockchain technology, more especially Ethereum smart contracts, enforces equitable data exchanges and preserves immutable logs. The system enables scalable participation of hundreds or thousands of data providers by enabling data consumers to query statistical insights while maintaining privacy. By addressing issues like filtering, compensation, and dispute resolution a smart-grid measurement sharing use case illustrates the approach's feasibility . In addition to discussing the systems low computational and blockchain-based overheads the paper emphasizes the trade-offs between accuracy and privacy.

2.3 Summary of Key Findings

The existing literature demonstrates significant gaps in integrating blockchain and DP in the sector of ride-sharing platforms. There is also a notable lack of research focused on aggregated ride-sharing platform that integrates multiple SPs within a unified, privacy-preserving, and decentralized framework. Several studies have implemented blockchain technology focusing on decentralization, eliminating single points of failure and privacy leakage to improve the privacy, security, and efficiency of ride-sharing platforms. Other studies incorporated DP which primarily explored protecting location-based data like location entropy, but they lack the implementation of DP in the specific context of ride-sharing platforms. Some studies have integrated blockchain and DP, but these have been confined to sectors like mobile crowdsourcing, and protecting sensitive infrastructure data. To the best of our knowledge, none of the papers combine blockchain and DP to comprehensively address the dual challenges of privacy preservation and system security in an aggregated ride-sharing platform. Our proposed system is an aggregated platform which collects data from multiple ride-sharing platforms and generates aggregated insights into traffic conditions, demand patterns, and location trends, benefiting not only individual users but also urban planners and policy makers. Additionally, our proposed dual-layered approach, a privacy-preserving decentralized aggregated ride-sharing platform bridges this gap by integrating blockchain and DP and is expected to raise the level of security and privacy above the existing systems.

Table 2.1 shows a comparative analysis of existing papers and our proposed system where the symbol ✓ has been used to denote certain criteria are satisfied by the respective work whereas the symbol ✗ indicates that a certain criterion is not satisfied. The table demonstrates that only our proposed system meets all these criteria, highlighting its novelty and contribution to privacy-preserving decentralized ride-sharing platforms.

Paper	**Decentralized Platform**	**Aggregated Platform**	**Strong Re-identification Protection**	**Data Insight**	**Optimized Data Storage**
Baza et al. [22]	✓	✗	✗	✗	✗
Wang & Zhang [28]	✓	✗	✗	✗	✗
Ahmed et al.[39]	✓	✗	✗	✗	✗
Mahmoud et al. [38]	✓	✗	✗	✗	✓
Oksuz [45]	✓	✗	✗	✗	✗
Kato et al.[20]	✓	✗	✗	✓	✗
Kudva et al. [26]	✓	✗	✗	✗	✗
Nosouhi et al. [27]	✓	✗	✗	✗	✗
Zou et al. [34]	✓	✗	✗	✗	✗
Wang et al. [15]	✗	✗	✓	✓	✗
Guo et al. [36]	✓	✗	✓	✗	✗
Hien et al. [13]	✗	✗	✓	✓	✗
Alnemari et al. [17]	✓	✗	✓	✗	✗
Fotiou et al. [31]	✓	✗	✓	✗	✓
Sun et al. [33]	✓	✗	✓	✗	✗
Proposed System	✓	✓	✓	✓	✓

Table 2.1: Comparative Analysis of Existing Papers and Our Proposed System

Chapter 3

Requirements, Impacts and Constraints

3.1 System Proposal

This research aims to develop a decentralized aggregated ridesharing platform that helps users to view and compare available ride options from multiple SPs in a single interface. It offers a more efficient and user-friendly experience for users. Beyond convenience, the platform also prioritizes privacy and security. Sensitive location data from SPs is processed by DP, even before using, so that individual user location remains confidential, even in aggregated analysis. The platform protects user identities while still providing valuable insights on traffic, demand patterns and fare trends for the specific given time. The adoption of DP also demonstrates compliance with global and local data privacy regulations. Moreover, Smart contract and Blockchain technology will be used for managing user registrations, consents management to ensure transparency. User consents are stored immutably on the blockchain, providing verifiable audit trails for both users and SPs.

Our proposed system will manage data in an optimized and efficient manner by storing low-volume data, such as registrations or user permissions on the blockchain, and wide-scale datasets, such as comprehensive location information, outside the blockchain. The decentralized architecture of the system and the utilization of privacy-preserving techniques on location data improves data privacy, auditability of the data, and increases user trust. This project focuses on designing a novel framework to analyze data of ride-sharing networks while addressing privacy concerns, which will further boost innovation in data-driven urban mobility.

3.2 System Specification and Requirement

3.2.1 Threat Modeling

Focusing on potential vulnerabilities in the decentralized ride-sharing platform, we will model privacy threats using threat modeling framework, LINDDUN [9]. It is a privacy-focused threat modeling framework, which systematically identifies and classes the risks in seven classes, namely Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, and Non-compliance. It pro-

vides standardized instructions for exploring software architectures' privacy risks. Though the LINDDUN threat model offers a comprehensive classification of privacy threats, we have focused on a subset of categories that are most relevant to our decentralized aggregated ride-sharing platform. Besides LINDDUN, we have also incorporated system-specific privacy threats that arise uniquely from the architectural design of our proposed system. This hybrid approach guarantees a thorough evaluation of privacy risks specific to our platform.

T1. Linkability: Linkability is the capacity to combine, to associate or to link data, often actions to learn more about an individual or group aiming to get more insight from combined data. This threat arises when an adversary can correlate multiple activities or data points with a particular user, compromising their privacy. In our model, an adversary may correlate the aggregated data or requests from multiple SPs, with a particular user, compromising anonymity. Additionally, repeated patterns of approximate location requests can leak a person's behavior in the long term. Therefore, prevention of linkability must be ensured while aggregating the data for the platform.

T2. Identifying: Identifying is to infer the true identity of a user behind any behaviour, action or data. In the context of our system, an attacker is capable of re-identifying an individual from information by matching information such as riding pattern or location retained with other such as ZIP code, financial spending. If the information is not properly anonymized and is easily cross-matched against other sources, users can be identified. As it increases the risk of exposing individual user identities, the data should be secured and protected.

T3. Disclosure of Data: Disclosure of data means unauthorized collection or exposure of personal data. In a decentralized environment, when a node is compromised, leakage of private information such as user location, session details, or preferences is possible. This is a severe privacy threat, and it is experienced when redirection or data transfer is occurring. So, the access control of the system and anonymization methodology should be dealt carefully.

T4. Unawareness and Unintervenability: It refers to lack of transparency over usage of users personal data. They may be unaware of where information is collected, stored, or transferred. The disclosure of privacy policies, procedures for obtaining a user's permission, and withdrawing permission can enhance awareness and trust in the system for users.

T5. Non-compliance: Non compliance means failure to comply with the security and data management standards. The system may inadvertently violate privacy regulations like GDPR by failing to implement data anonymization or user rights mechanisms. So, all should be designed in such a way that complies with the regulations and standards.

In addition to these, we have considered a system-specific threat beyond LINDDUN to cover session-related vulnerabilities in our decentralized, redirection-based flow.

T6. Session Integrity: Session integrity refers to the protection of ongoing user interactions from unauthorized interference or manipulation. While rerouting SPs, the abusers can attempt session hijacking or session spoofing in an effort to mimic

the rider, add malicious data, or alter decision path. The session hijacking can compromise rider privacy as well as ride choice control. Therefore, ensuring robust session validation and secure rerouting mechanisms is essential to preserve session authenticity and trust.

3.2.2 Requirements Analysis

3.2.2.1 Functional Requirements

To ensure seamless user experience the proposed system adheres to the following functional requirements.

1. Decentralized Registration and Identity Binding: Users need to register to the DApp via a Web3-compatible wallet. The backend must verify user email to bind a verifiable real-world identity with the blockchain-based digital identity.

2. Consent-Driven Access Control: Users must be able to selectively grant or revoke data access permissions. All such permission flags need to be recorded immutably on the blockchain to ensure transparency.

3. OAuth-Based Token Management: In order to securely acquire and store access tokens and refresh tokens for authorized data access, the system must support OAuth flows for each SP.

4. Session-Oriented Query Processing: Upon a user query, the system needs to initialize a unique session that persists across both the data retrieval and subsequent ride booking phase to ensure consistency throughout the entire process.

3.2.2.2 Privacy Requirements

In an effort to address identified threats in our threat model, our proposed system should fulfill the following requirements for privacy.

1. DP for Data Protection: To mitigate Linkability (T1) and Identifiability (T2) DP approaches would be used for the protection of all types of user data including ride history and location data. DP as a replacement for typical anonymization ensures that it is not possible for lone user data to be identified by adding precisely calibrated noise in data or in query output. To prevent attribution of individual trips, local differential privacy (LDP) would add noise in ride queries and obscure users devices GPS coordinates before putting location data in. Firm privacy is obtained with data utility preserved using this method which ensures it is not possible for a lone individual even when aggregate data is studied, for it to be identified.

2. Secure Data Sharing: To address Disclosure of Information (T3), all private data must be encrypted or hashed before storage or data transmission. Smart contracts should be used for regulating access control in a manner where only qualified services have access to data owned by a user.

3. Transparent Consent Mechanisms: Users must have clear privacy policies specifying no personal data is stored and no linkage with external SPs is maintained

without their consent. Data handling should remain transparent, and users should be able to review or revoke any authorizations at any time (T4).

4. Regulatory Compliance: To manage noncompliance (T5), it should be possible for all platforms to be in conformity with privacy legislation like GDPR. This is possible in the form of a right of revocation and a user's data access capability and for following information use.

5. Secure Session Management: As a countermeasure against the Session Integrity Threat (T6), time-limited session tokens should be used for redirections to SPs while booking. The tokens are transmitted and stored securely over HTTPS/TLS to ensure confidentiality and integrity. OAuth 2.0 is employed for authorization, ensuring that only authenticated users can complete the redirection and act on external services.

3.3 Societal Impact

The proposed decentralized, privacy preserving ride sharing platform can benefit the society by enhancing user privacy. Integrating DP, the system ensures that users' sensitive information, such as location data, remains protected. Moreover, by incorporating Blockchain and OAuth, the platform empowers users by giving them more control over their personal data while ensuring transparency and auditability within the system. This builds user trust and confidence in ride-sharing services. Additionally, DApp consisting of multiple SPs in one platform also simplifies the user experience. It offers more efficient transportation options. The increased availability of such services can improve urban mobility. This is how it can contribute to overall societal well-being.

3.4 Environmental Impact

The environmental impact of this platform is primarily indirect, because it facilitates efficient ride-sharing, which may passively contribute in reducing vehicle emissions and alleviating traffic congestion. But the main environmental concern arises from the energy consumption associated with blockchain transactions. However, the system uses Ethereum which now operates on a Proof-of-Stake (PoS) consensus mechanism that drastically reduces energy consumption compared to Proof-of-Work (PoW). Moreover, in our system, blockchain is used only to store registration flags and consent flags. Thus, the total energy consumption remains very low. To summarize, the platform offers a solution where technological innovation and environmental responsibility coexist by combining minimal energy usage with the potential to foster sustainable urban mobility.

3.5 Ethical Issues

This research utilizes only synthetic data ensuring no real user data has been used. Moreover, the platform prioritizes user privacy, security, ethical concerns regarding data exchange and consent management, making these considerations central to its

design and implementation. The system ensures users' consent is taken while linking their SP account to DAPP account and users have the option to withdraw access at any moment. Additionally, since the design of the platform relies on DP for location data exchange, it provides a formal guarantee that the users' sensitive data remains protected. These measures reflect a user-centric and responsible approach in handling ethical issues.

3.6 Economic Analysis

In the economic aspect, the proposed system may increase the revenue by integrating several SPs, attracting in more users and raising transaction volume. Additionally, by maximizing ride matching and reducing vehicle idle times, it can lower operating costs while saving the time of both the SPs and the users. Since, the system utilizes blockchain, the cost associated with it becomes very crucial and therefore the cost of each blockchain transaction is calculated in USD. To calculate the gas fee Equation 3.1 is used and the transaction fee is of each transaction is calculated using Equation 3.2. This calculation was performed on 9th October, 2025 when the gas price was 0.31 Gwei and 1 ETH = $4,384.67$ [50].

$$\text{Gas Fee (ETH)} = \text{Gas Used} \times \text{Gas Price (Gwei)} \times 10^{-9} \text{ ETH} \tag{3.1}$$

$$\text{Transaction Fee (USD)} = \text{Gas Fee (ETH)} \times \text{ETH Price (USD)} \tag{3.2}$$

The detailed cost breakdown of the blockchain implementation that shows the cost of each blockchain transaction in USD is presented in Table 3.1. The table represents that the blockchain transaction cost is not significant for our system since it is a one-time cost. Therefore, this initial investment can bring the long-term benefits include improved user trust, enhanced market competitiveness, and potential scalability, making the system economically viable in the long run.

Table 3.1: Cost Breakdown of Blockchain Transaction

Operation	**Gas Used**	**Gas Fee (ETH)**	**Transaction Fee (USD)**
Registration	21,064	6.68×10^{-6}	\$0.0293
Permission Grant	21,344	6.77×10^{-6}	\$0.0297
Permission Revoke	21,332	6.77×10^{-6}	\$0.0297

3.7 Project Management Plan

Our research plan is divided into three phases. Each phase addresses specific aspects, starting from threat modeling, going through multiple steps, and finally concluding at evaluation. An overview of these phases along with their respective timelines is presented in Table 3.2.

Table 3.2: Research Methodology Phases

Phase	Work Done
Phase 1	• Have performed threat modeling using LINDDUN framework. • Have identified functional and non-functional (privacy) requirements. • Have proposed a privacy-preserving and storage-efficient design to address the significant issues of existing ride-sharing platforms.
Phase 2	• Designed system architecture and described protocol flow to specify secure interaction among components. • Performed partial implementation of the proposed system to validate core functionalities and demonstrate feasibility.
Phase 3	• Completed the implementation of the system. • Evaluated of performance based on latency, response time, functionality, privacy preservation. • Analyzed limitations, and defined future work.

3.8 Risk Management

Risk management for the proposed system primarily focuses on addressing potential vulnerabilities in user data privacy. A key concern is ensuring that users have full control over their personal data. To achieve this, the platform integrates Blockchain for transparency and auditability, along with OAuth for secure and user-controlled access management. The platform also guarantees that the users can easily revoke consent at any time when necessary. Additionally, to ensure the privacy and security of users' sensitive location data Geo-Indistinguishability DP is utilized. A key concern in this approach is ensuring that the noise added to location data does not degrade utility of the data. To address this challenge, the system undergoes rigorous testing and careful tuning of the privacy budget, balancing privacy with usability.

Chapter 4

Proposed Methodology

4.1 Methodology Overview

The research focuses on creating a user's consent-driven, smart contract and blockchain-integrated, privacy-preserving aggregated ride-sharing platform. Our research methodology is structured through a 6-step process. Figure 4.1 illustrates the steps of our research.

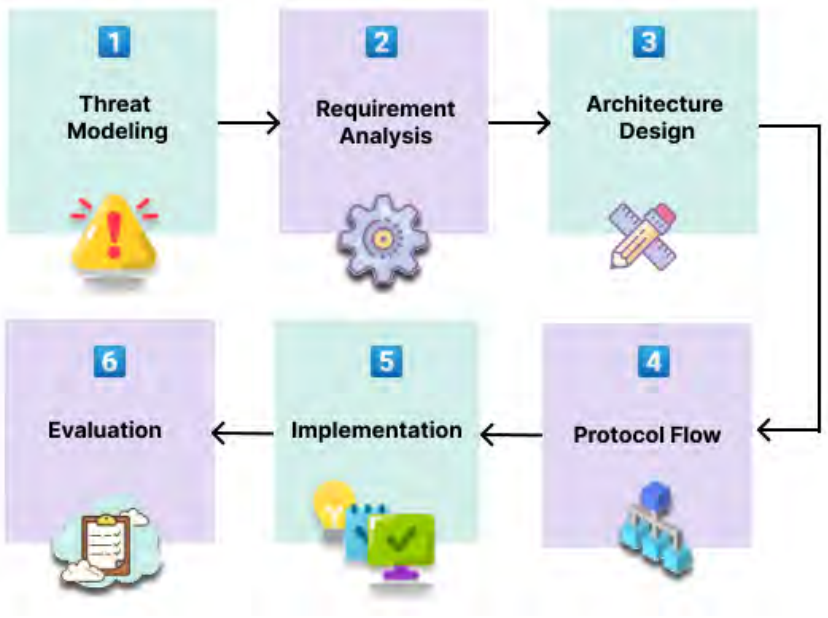


Figure 4.1: Methodology

Threat Modeling: Privacy threats were identified and assessed through systematic threat modeling, as explained in 3.2.1.

Requirements Analysis: A thorough analysis was conducted to define functional, security, and privacy requirements, as detailed in 3.2.2.

System Architecture Design: The architectural design phase utilized multiple tools and frameworks to outline the platform's components and data flows, discussed in 4.2.

Protocol Flow Design: Interaction protocols among the entities in different subsystem is carefully designed ad discussed in 4.3.2.

Implementation: The system is completely implemented following the designed protocol. The implementation outlines the procedures, tools, and algorithms employed to develop the system which is elaborated in Chapter 5.

Evaluation: The evaluation phase elaborated in Chapter 6 represents quantitative and qualitative analysis to evaluate response latency, privacy guarantees, and the usefulness of the system.

4.2 System Architecture

Before diving into the system architecture, we need to discuss some of the assumptions that we have taken into account to successfully deploy the proposed aggregated ride-sharing platform.

Assumption 1: To enable secure access to user accounts, all the ride-sharing SPs need to support standardized OAuth-based APIs.

Assumption 2: Since registration is accomplished via smart contract deployed on the Ethereum blockchain, users must have an Ethereum-compatible Web3 wallet to register on the platform.

Assumption 3: Each SP is equipped with a DP Engine which converts sensitive location or ride-related data into differentially private outputs before sending to the aggregated platform.

Assumption 4: The application can only be deployed if user data portability, digital consent, and ride-sharing interoperability are aligned with the law of the government of the territory.

Our proposed system, illustrated in Figure 4.2, is a web-based aggregated platform which uses blockchain and DP to ensure secure data management. The architecture consists of four main components which are User (4.2.1), SP (4.2.2), Aggregated Platform (DApp) (4.2.3), and Blockchain (4.2.4). In this high-level architecture, the registration flow is marked using red, the access-control process (permission to connect the SP account) is marked using green, and the dynamic mobility discovery or search for traffic insights and booking is marked using orange.

4.2.1 User

In the proposed system, a user is an individual looking to compare ride options among different SPs and book transportation from suitable SP via a single platform. At first an user needs to connect his Ethereum wallet and submit a registration form after accessing the Aggregated Platform. Registration includes an email verification process and permission-granting process. In the email verification step, DApp backend sends a verification to user email. Once registered and authorized, the user can

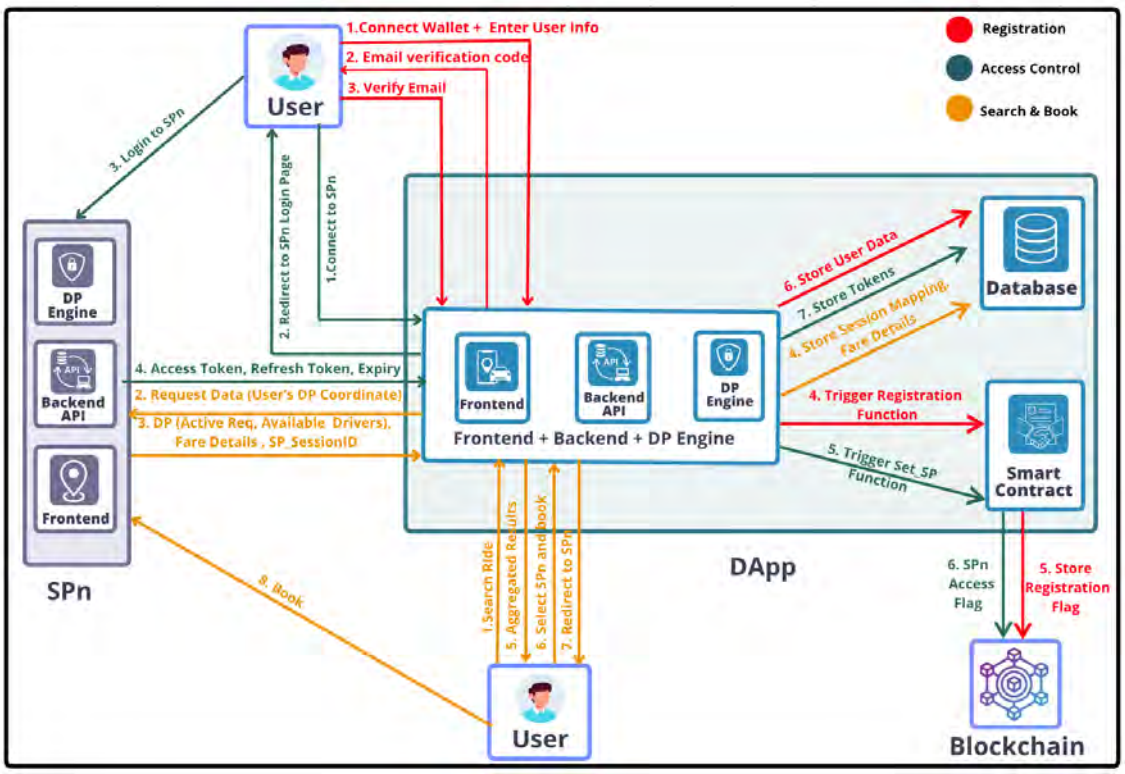


Figure 4.2: System Architecture

query ride availability for a selected location. The user then can proceed to booking from preferred SP after giving permission to link his SP account.

4.2.2 SP

SPs are third-party ride-sharing platforms. Each SP must support OAuth 2.0 protocols for secure token exchange. When an user wants to link his SPn account to DApp, SPn send access token, refresh token, and expiry to the DApp upon verifying user's identity which will be user to securely transfer data in bookingphase.

SPs also have integrated Differential Privacy (DP) Engine for privacy preserving data transfer while discovering ride. The DP engine utilizes Geo-Indistinguishability, a Local DP mechanism to add controlled laplace noise. After receiving requests from the DApp, the SP converts raw location data of the drivers and active users to a DP-protected format via its DP engine and then pass the Dp-data and fare details along with sessionD to the DApp. During the booking process, the user selects an specific SP and redirected to SPn's booking interface with access token, sessionID and other details.

4.2.3 DApp

DApp functions as a middleware that coordinates communication between the blockchain, SP, and users. DApp plays a crucial role in establishing a trust anchor between the user's digital identity (wallet address) and their real-world identity (email address) by email verification during registration via verification code sent to the user's email.

Upon successful verification, a hashed version of user data is stored on-chain in the user's registration record transaction. This linkage guarantees a trustworthy association between verifiable user identities and pseudonymous blockchain identities which is essential for enforcing accountability and enabling future audits. Additionally, OAuth flow is also supported by the DApp. Its backend manages token exchange and safely stores encrypted access token, refresh token, and expiry in the database which is later used in booking. User can link or unlink his SP account from DApp at any point which ensured user control over consent. Registration and consent management is controlled via smart contract and flags are stored in blockchain to ensure transparency and auditability.

When a user queries for traffic data retrieval from a specific location, the backend of the DApp initiates a session by generating a unique session ID tied to the user's wallet address and geolocation. Then it calls API of all the SPS and receives DP-protected active requets and available driver data, fare details, SP session ID from each of the SPs. Upon receiving the response of all SPs, the DApp aggregates the data to show the combined result to the user. Moreover, to validate and manage the session during the booking phase, it stores the mapping of the session id of the DApp and the SPs in the session database. This mapping is essential for providing user persistent data while searching and booking for a seamless user experience.

4.2.4 Blockchain

The system utilizes Ethereum based public blockchain system which supports smart contracts and wallet-based authentication via standard Web3 interfaces. It serves as a decentralized trust layer by ensuring transparent, secure, and auditable coordination among entities without relying on centralized intermediaries.

Smart contracts play a key role in user registration, authorization, and permission management To guarantee that only the user has control over their account, all registration and authorization procedures are carried out as transactions that are signed by the user's wallet. Every permission that is granted or revoked, every registration, and every crucial interaction are recorded and stored immutably via Smart contracts. These timestamped and cryptographically protected logs guarantee auditable transparency throughout the system.

Storage of personal or sensitive data on-chain can potentially violate privacy and storing large and dynamic data can be inefficient and costly. Therefore, such data is maintained off-chain and the blockchain only stores essential metadata. This hybrid architecture ensures data integrity, auditability, and scalability while complying with privacy regulations.

4.3 Protocol Flow Design

This section outlines the 5 major protocols that are utilized to develop the proposed system. Before describing the flows the relevant notations are introduced in Table 4.1) and the data model in Table 4.2.

4.3.1 Data Model

Table 4.1: Cryptographic and other Notations

Notation	Meaning
u	User
D	Aggregator DApp (Frontend + Backend)
SC	Smart Contract
BCH	Blockchain ledger
SP	SP
DB	Database
$H(x)$	SHA-256 hash of x
$[\dots]_{\text{https}}$	HTTPS-encrypted payload
N_k	Fresh nonce k
req$(type, data)$	Request tuple $\langle type, data \rangle$
resp	Response tuple $\langle responsereq, returnedData \rangle$
$registerData$	$\langle walletAddr, name, email, phone \rangle$
$permissionData$	$\langle walletAddr, SP, true/false \rangle$
$loginData$	$\langle walletAddr \rangle$
$placeQuery$	$\langle lat, lng, destination, carType, radius \rangle$
$aggregatedResult$	List$\langle SP, approxFare, DP_driver, DP_nearby_users \rangle$
$providerSessionMapping$	$\langle agg\_session\_id, sp\_session\_id, ratio \rangle$
$ratio$	Adjustment factor to map approximate DP-based fare to true fare
$bookingRequest$	$\langle provider, walletAddr, sp\_session\_id, token, ratio \rangle$
$bookingConfirmation$	$\langle booking\_url, mainFare \rangle$
$mainFare$	Final computed fare at booking
$revokeRequest$	$\langle walletAddr, SP \rangle$
$tokenRecord$	$\langle walletAddr, SP, access, refresh, expiry \rangle$
K_{DB}	Encryption key for the DB (symmetric or public key)
$\{\texttt{tokenRecord}\}_{K_{\text{DB}}}$	Encrypted token record under key K_{DB}
$\emptyset$	Empty/null

All user interactions within the system are represented as structured requests, denoted by `req`. It consists of two components: a *type* and *data*. The *type* indicates the category of operation and the associated *data* holds the relevant parameters for the request. Formally, this is defined as:

$$req \coloneqq \langle type,\ data \rangle, \quad type \in TYPE, \quad data \in DATA$$

The system generates a structured response tuple for each request:

$$resp \coloneqq \langle responsereq,\ returnedData \rangle$$

Here, *responsereq* refers to the outcome of the operation such as success or failure, and *returnedData* contains additional return values if needed. When no additional data is required, *returnedData* is empty ($\emptyset$).
The defined $TYPE$ set includes six categories of user interactions: `registration`, `login`, `permissionGrant`, `aggQuery`, `bookRide`, and `revokePermission`. Each type corresponds to a structured input defined within the $DATA$ set.

Table 4.2: Data Model

$req \coloneqq \langle type, data \rangle$
$resp \coloneqq \langle responsereq, returnedData \rangle$
$TYPE \coloneqq \langle$registration, login, permissionGrant, aggQuery, bookRide, revokePermission$\rangle$
$DATA \coloneqq \langle registerData, loginData, permissionData, placeQuery,$ $providerSessionMapping, bookingRequest, revokeRequest \rangle$
$registerData \coloneqq \langle walletAddr, name, email, phone \rangle$
$loginData \coloneqq \langle walletAddr \rangle$
$permissionData \coloneqq \langle walletAddr, provider, permission \rangle$
$placeQuery \coloneqq \langle lat, lng, destination, carType, radius \rangle$
$aggregatedResult \coloneqq \text{List}\langle SP, approxFare, DP\_driver, DP\_nearby\_users \rangle$
$providerSessionMapping \coloneqq \langle agg\_session\_id, sp\_session\_id, ratio \rangle$
$bookingRequest \coloneqq \langle provider, walletAddr, sp\_session\_id, token, ratio \rangle$
$bookingConfirmation \coloneqq \langle booking\_url, mainFare \rangle$
$revokeRequest \coloneqq \langle walletAddr, SP \rangle$

4.3.1.1 Registration

The *registration* phase is a hybrid identity management process that links a user's verified email identity with their pseudonymous blockchain wallet. The user begins by connecting their MetaMask wallet, which serves as a unique identifier. If unregistered, they provide personal details which are name, phone number, and email, which are securely hashed using the HMAC-SHA256 algorithm before being stored in the database to ensure anonymization. A verification code is then emailed via Gmail SMTP using Nodemailer, and upon successful verification, the system record the wallet address, registration flag, and timestamp immutably on the blockchain.

4.3.1.2 Login

In the *login* protocol, explicit user authentication is not required. Instead, the DApp automatically checks whether the connected *walletAddr* is registered. If registered, the user is automatically logged in.

4.3.1.3 Permission Management

The *permission management* process, represented by *permissionData*, enables users to grant or revoke access to specific SPs. Permissions are enforced on-chain via the smart contract while OAuth tokens are issued off-chain. These tokens are encrypted using the DB encryption key (K_{DB}) and stored securely:

$$\{\texttt{tokenRecord}\}_{K_{\text{DB}}}$$

This hybrid approach ensures both immutable on-chain consent management and secure off-chain token handling.

4.3.1.4 Aggregated Query

The $type$ = `aggQuery` represents user-initiated ride availability and fare queries across multiple SPs. The submitted query includes differentially private user location, destination, car type, and radius:

$$\texttt{placeQuery} = \langle lat, lng, destination, carType, radius \rangle$$

Each SP responds with approximate results:

- *sp_session_id*: Identifier from SP's system.
- *DP_driver*: Driver positions with DP applied.
- *DP_nearby_users*: Other nearby ride requests, with DP applied.
- *approxFare*: Estimated fare from DP data.
- *ratio*: Adjustment factor to map approximate fares to true fares.

The DApp compiles results into:

`aggregatedResultList`$\langle SP, approxFare, DP\_driver, DP\_nearby\_users \rangle$

Mappings between sessions and ratios are also stored:

`providerSessionMapping`$\langle agg\_session\_id, sp\_session\_id, ratio \rangle$

4.3.1.5 Booking

The *booking transaction*, *type* = `bookRide`, allows a user to confirm rides with a selected SP. The booking request contains both DApp and provider session identifiers, wallet address, tokens, and the stored ratio:

`bookingRequest`$\langle provider, walletAddr, sp\_session\_id, token, ratio \rangle$

The SP then computes the actual fare using the ratio and real location data, returning:

`bookingConfirmation`$\langle booking\_url, mainFare \rangle$

This ensures that the user sees the true fare only at the SP's interface, preserving both integrity and privacy.

4.3.1.6 Permission Revocation

Finally, users can revoke SP access through *type* = `revokePermission`. Revocation requests:

`revokeRequest`$\langle walletAddr, SP \rangle$

invalidate OAuth tokens off-chain and update the smart contract's on-chain permission state, ensuring consent management and preventing further unauthorized data use.
This structured data model, leveraging blockchain immutability, DP, OAuth, and cryptographic primitives, guarantees secure, transparent, and privacy-preserving operations throughout all system interactions.

4.3.2 Protocol Flow

This section describes the protocol flow, illustrating user interactions in the system which are decentralized user registration, login, permission management, data aggregation by applying DP, and booking functionalities. Each protocol is accompanied by tables and figures which clearly specifies message exchanges and cryptographic notations.

4.3.2.1 Registration Protocol

Table 4.3: Registration Protocol

Msg	Flow	Description
M1	$u \rightarrow D$	$[N_1,$ `req(registration, registerData)`$]_{\text{https}}$
M2	$D \rightarrow DB$	$[N_2,$ `storePD(walletAddr, nameHash, emailHash, phoneHash)`$]$
M3	$D \rightarrow u$	$[N_3,$ `emailVerificationLink(code)`$]_{\text{smtp}}$
M4	$u \rightarrow D$	$[N_3,$ `verifyEmail(code)`$]_{\text{https}}$
M5	$D \rightarrow SC$	$N_4,$ `req(registration, registerData)`
M6	$SC \rightarrow BCH$	$N_5,$ `registerData`
M7	$BCH \rightarrow SC$	$N_5,$ TRUE
M8	$SC \rightarrow D$	$N_4,$ resp $= \langle$UserRegistered, $\emptyset\rangle$
M9	$D \rightarrow u$	$[N_1,$ `resp` $=$ (Registration Successful, $\emptyset)]_{\text{https}}$

To be a part of the system, users need to complete a blockchain-based registration process integrated with conventional web-based verification systems. The registration flow is detailed in Table 4.3 and visualized in Figure 4.3. Each registration request is represented as:

$$req = \langle type,\ data \rangle, \quad type = \texttt{registration}, \quad data = \texttt{registerData}$$

which contains the user's blockchain wallet address, personal information (name, email, phone number), and metadata.

- **Step 1:** The user initiates registration by connecting their blockchain wallet via the frontend interface using MetaMask, which serves as the user's unique digital identity. The backend checks if the user is already registered. If not, the DApp prompts the user to input their personal details, including name, phone number, and email address. The application collects these details, converts them to lowercase, trims whitespace, and processes them using the HMAC-SHA256 hashing algorithm. The hashed data is then securely stored in the database (message M_2).
- **Step 2:** After successful wallet verification, the frontend sends the user a verification link or code to the entered email address (message M_3) via Gmail SMTP service using the Nodemailer API. The user receives the email and clicks the verification link or enters the code (message M_4).
- **Step 3:** Upon receiving the verification code, the frontend verifies it against the database's stored hashed code. After successful verification, the backend nullifies the code in the database to prevent reuse.
- **Step 4:** After email verification, the DApp prompts MetaMask to initiate a blockchain transaction, which triggers the function of the User Smart Contract. The smart contract stores the user's wallet address, registration flag, and timestamp immutably on the blockchain, creating a blockchain-based proof of registration. The backend logs the blockchain reference to the database (messages M_5 and M_6).

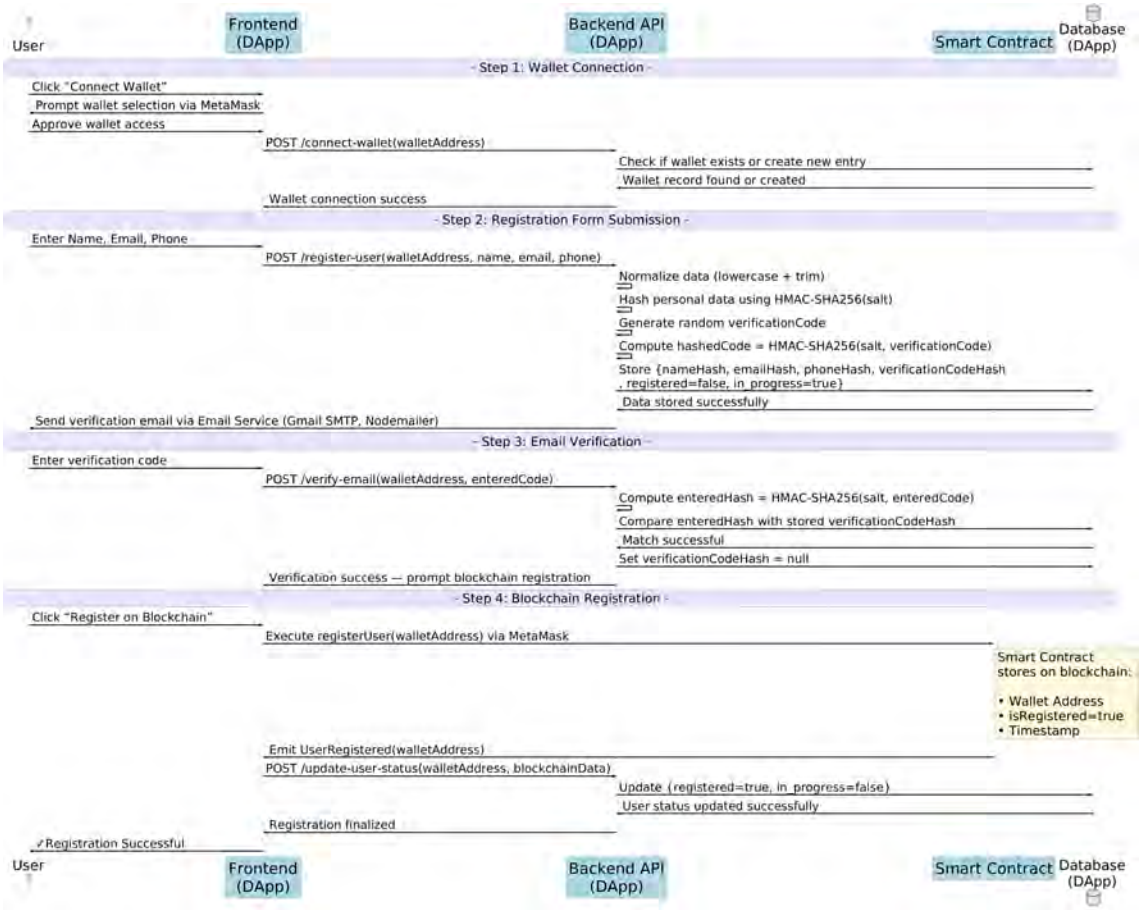


Figure 4.3: Registration Flow

- **Step 5:** The smart contract emits a `UserRegistered` event on successful registration (message M_7).
- **Step 6:** The DApp frontend receives the registration status, and the UI is updated accordingly, confirming the registration process (message M_8).
- **Step 7:** Finally, the frontend delivers a registration success confirmation to the user (message M_9), and the user is logged in automatically:

$$\text{resp} = \langle \text{Registration Successful}, \emptyset \rangle$$

4.3.2.2 Login Protocol

The system consists of an automatic wallet-based login protocol. Instead of requiring explicit user login, the DApp checks if the wallet address is already registered and automatically logs the user in if valid. The detailed flow is shown in Table 4.4 and Figure 4.4.

- **Step 1:** Upon visiting the DApp, the frontend sends the connected `walletAddr` to the backend to check registration status (message M_1).
- **Step 2:** The backend queries the database to confirm whether the wallet is already registered (messages M_2, M_3).

Table 4.4: Login Protocol

Msg	Flow	Description
M1	$u \rightarrow D$	$[N_1,$ `req(checkRegistration, walletAddr)`$]_{\text{https}}$
M2	$D \rightarrow DB$	$N_2,$ `checkRegistrationInDB(walletAddr)`
M3	$DB \rightarrow D$	$N_2,$ {`Registered/NotRegistered`}
M4	$D \rightarrow u$	$[N_1,$ resp $= \langle \text{AutoLogin}, status \rangle]_{\text{https}}$

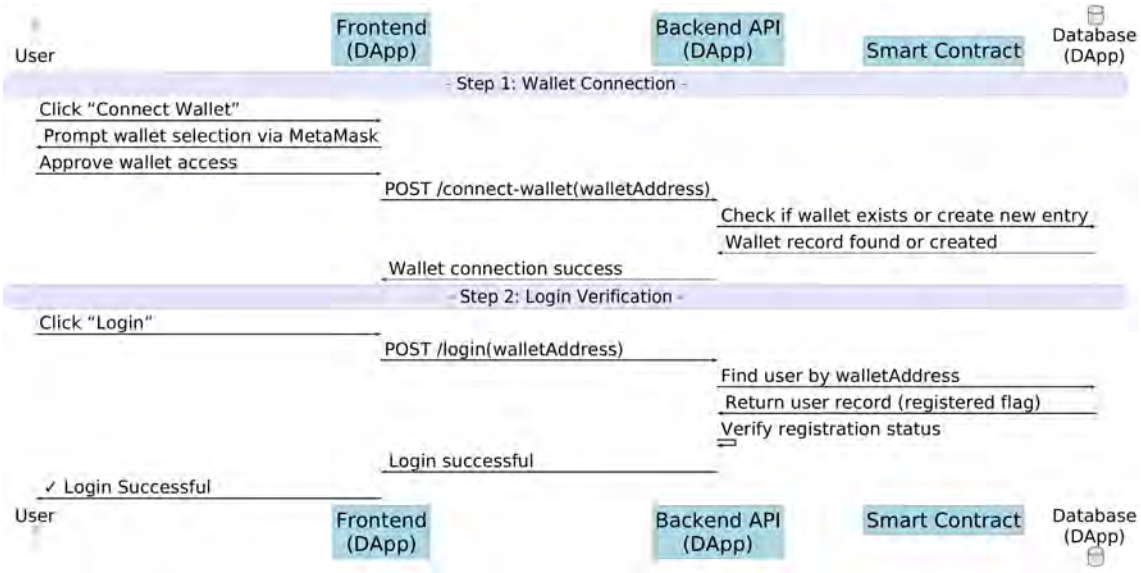


Figure 4.4: Login Flow

- **Step 3:** If the wallet is registered, the DApp automatically logs the user in without requiring explicit credentials and updates the frontend accordingly (message M_4).

4.3.2.3 Access Control Protocol

We have divided the Access Control Protocol into two distinct sections including Permission Granting Protocol and Permission Revocation Protocol.

1. Permission Granting Protocol: After the wallet-based registration, users can authorize specific SPs to access their data. The permission granting protocol is detailed in Table 4.5 and Figure 4.5.

- **Step 1:** For each selected SP, the user initiates a permission-granting request from the frontend. This request is sent to the SP via OAuth redirect (messages M_1 and M_2).
- **Step 2:** The OAuth server returns an authorization code upon successful user login and consent (message M_3).
- **Step 3:** The DApp backend exchanges the received authorization code for OAuth tokens, including `access_token`, `refresh_token`, and expiry details (messages M_4 and M_5).

Table 4.5: Permission Granting Protocol

Msg	Flow	Description
M1	$u \rightarrow D$	$[N_1,$ `req(permissionGrant,permissionData)`$]_{\text{https}}$
M2	$D \rightarrow SP_n$	N_2, redirect to `/auth?wallet={walletAddr}&provider={SP`$_n$`}`
M3	$SP_n \rightarrow D$	N_2, `auth_code`
M4	$D \rightarrow SP_n$	N_3, `exchange(auth_code`$\rightarrow$`access,refresh)`
M5	$SP_n \rightarrow D$	N_3, {`access, refresh, expiry`}
M6	$D \rightarrow SC$	N_4, `req(permissionGrant,permissionData)`
M7	$SC \rightarrow BCH$	N_5, `permissionData`
M8	$BCH \rightarrow SC$	N_5, TRUE
M9	$SC \rightarrow D$	N_4, ack $= \langle$Permission Granted, $\emptyset\rangle$
M10	$D \rightarrow DB$	N_6, {`tokenRecord`}$_{K_{\text{DB}}}$
M11	$D \rightarrow u$	$[N_1,$ `resp` $=$ (Permission Granted, $\emptyset)]_{\text{https}}$

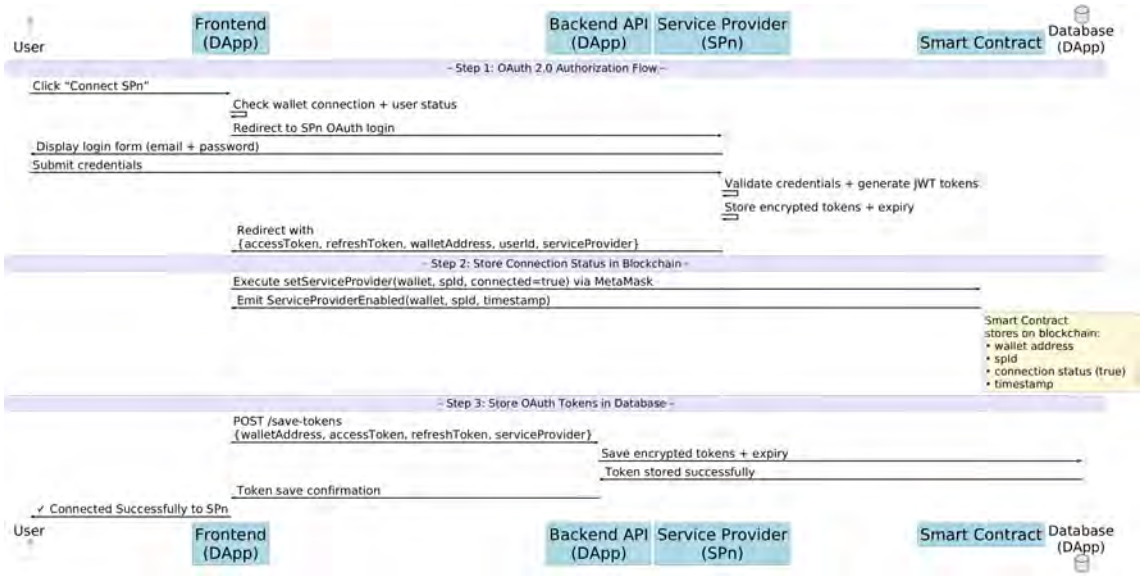


Figure 4.5: Permission Grant Flow

- **Step 4:** After receiving OAuth tokens, the frontend submits the permission grant request to the smart contract (message M_6):

$$\texttt{req(permissionGrant, permissionData)} = \langle \texttt{walletAddr}, \texttt{SP}, \texttt{true} \rangle$$

- **Step 5:** The smart contract records the permission grant and updates the blockchain state accordingly. It emits a `PermissionSet` event (messages M_7 and M_8).

- **Step 6:** After successful on-chain permission setting, the smart contract confirms to the frontend via an acknowledgment (message M_9):

$$\text{ack} = \langle \text{Permission Granted}, \emptyset \rangle$$

- **Step 7:** The DApp backend securely encrypts and stores the OAuth tokens as `tokenRecord` off-chain within the DB (message M_{10}):

$$\{\texttt{tokenRecord}\}_{K_{\text{DB}}}$$

- **Step 8:** Finally, the DApp sends a confirmation response to the user, indicating that the permission grant was successful (message M_{11}):

$$\text{resp} = \langle \text{Permission Granted}, \emptyset \rangle$$

2. Permission Revocation Protocol: Users can revoke previously granted permissions to SPs, which ensures consent management and user privacy, outlined in Table 4.6 and Figure 4.6.

Table 4.6: Permission Revocation Protocol

Msg	Flow	Description
M1	$u \rightarrow D$	$[N_1,$ `req(revokePermission,revokeRequest)`$]_{\text{https}}$
M2	$D \rightarrow SP_n$	$N_2,$ `revokeTokenAPI(access_token)`
M3	$D \rightarrow SC$	$N_3,$ `req(permissionGrant,permissionData)`
M4	$SC \rightarrow BCH$	$N_4,$ `permissionData`
M5	$BCH \rightarrow SC$	$N_4,$ TRUE
M6	$SC \rightarrow D$	$N_3,$ resp $= \langle$Permission Revoked, $\emptyset \rangle$
M7	$D \rightarrow DB$	$N_5,$ `removeTokens(walletAddr,SP)`
M8	$D \rightarrow u$	$[N_1,$ `resp` $=$ (Permission Revoked, $\emptyset)]_{\text{https}}$

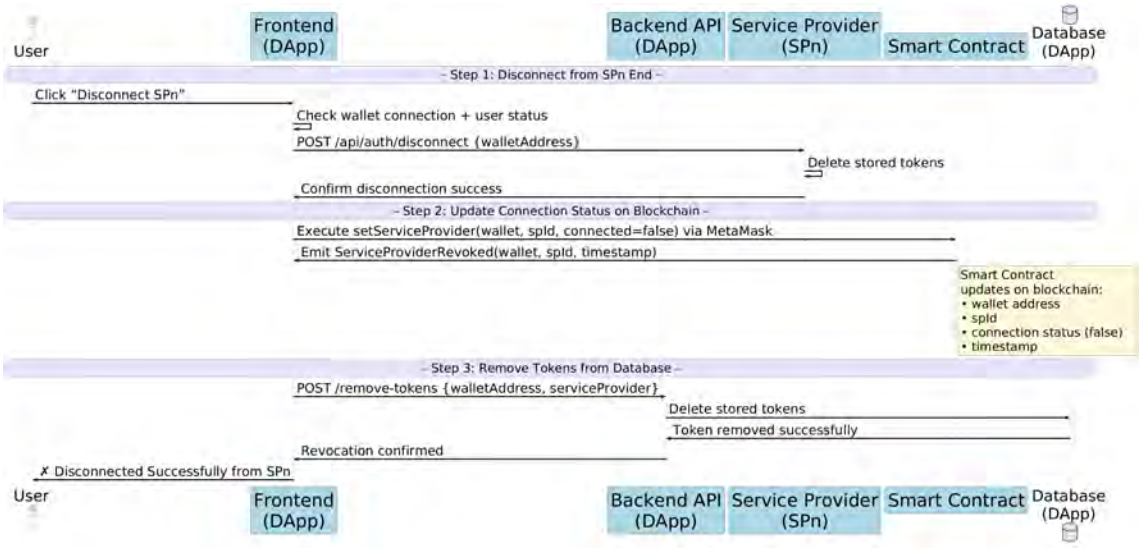


Figure 4.6: Permission Revoke Flow

- **Step 1:** Users initiate revocation by selecting an SP via the frontend and triggering a revocation request, `req(revokePermission,revokeRequest)` (message M_1).
- **Step 2:** The backend invalidates the SP's OAuth access token via `revokeTokenAPI(access_token)`, ensuring that the SP can no longer issue API calls using the previously granted token (message M_2).
- **Step 3:** The backend updates the on-chain permission state by invoking the smart contract function, where `permissionData = (walletAddr,SP,false)`, emitting a `PermissionRevoked` event if successful (messages M_3, M_4, M_5, M_6).
- **Step 4:** After blockchain confirmation, the backend removes the corresponding OAuth token records from the DB to ensure no residual tokens remain off-chain (message M_7).
- **Step 5:** Finally, the backend returns the revocation confirmation, $\langle$"Permission Revoked", $\emptyset\rangle$, updating the UI accordingly (message M_8).

4.3.2.4 Dynamic Mobility Discovery Protocol

This protocol describes how a user initiates a fare and ride availability query across multiple SPs, incorporating DP to preserve both location and destination privacy. The process is presented in Table 4.7 and Figure 4.7.

Table 4.7: Dynamic Mobility Discovery Protocol

Msg	Flow	Description
M1	$u \rightarrow D$	$[N_1,$ `req`(`aggQuery`, `placeQuery`)$]_{\text{https}}$
M2	$D \rightarrow DB$	$N_2,$ `storeSessionInDB`(`agg_session_id`)
M3	$D \rightarrow SP_n$	$N_{3n},$ `query`(`DP_lat`, `DP_lng`, `DP_destination`, `carType`, `radius`)
M4	$SP_n \rightarrow D$	$\langle N_{3n},\ sp\_session\_id,\ DP\_driver,\ DP\_nearby\_users,\ approxFare,\ ratio_n\rangle$
M5	$D \rightarrow DB$	$N_4,$ `storeMappingInDB`(`agg_session_id`, `sp_session_id`, `ratio_n`)
M6	$D \rightarrow u$	$[N_1,$ `resp` $= \langle$AggregationComplete, $aggregatedResult\rangle]_{\text{https}}$

- **Step 1:** The user submits an aggregated query to the DApp by providing:

  $$\texttt{placeQuery} = \langle lat, lng,\ destination,\ carType,\ radius\rangle$$

  The DApp generates a unique aggregator session ID and stores it in the session DB:

  $$aggregatorSession \coloneqq \langle agg\_session\_id,\ created\_at\rangle$$

  (Messages M1, M2).
- **Step 2:** The DApp applies DP to the lat and lng of user location and destination by using DP engine and forwards the DP-protected query to all authorized SPs. Each query contains the differentially private user location, destination, car type, and radius (Message M3).
- **Step 3:** Each SP responds with:

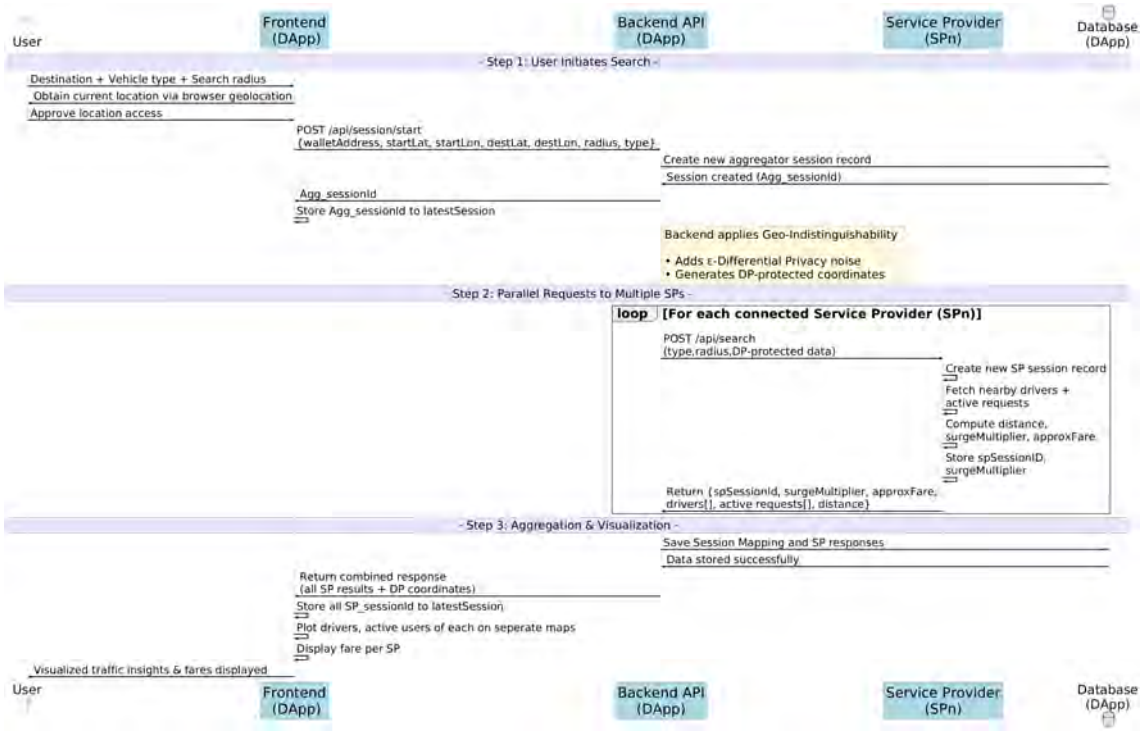


Figure 4.7: Dynamic Mobility Discovery Flow

- $sp\_session\_id$: Session identifier from SP.
- DP_driver: Locations of nearby drivers with DP applied.
- DP_nearby_users: Locations of other users searching nearby, with DP applied.
- $approxFare$: Approximate fare computed from DP data.
- $ratio_n$: Adjustment factor to reconcile approximate fare with true fare at booking.

(Message M4).

- **Step 4:** The DApp stores session mappings in DB:

$$providerSessionMapping := \langle agg\_session\_id,\ sp\_session\_id,\ ratio \rangle$$

(Message M5).

- **Step 5:** The DApp aggregates responses from all SPs and returns

$$aggregatedResult := \text{List}\langle SP,\ approxFare,\ DP\_driver,\ DP\_nearby\_users \rangle$$

which is delivered to the user (Message M6).

4.3.2.5 Booking Protocol

The booking protocol secures interactions between the DApp and SPs, maintaining fare integrity and authenticated access via OAuth. It additionally ensures that SP permissions are verified before booking initiation. The process is shown in Table 4.8 and Figure 4.8.

Table 4.8: Booking Protocol

Msg	Flow	Description
M1	$u \rightarrow D$	$[N_1,$ `req(bookRide,bookingRequest)`$]_{\text{https}}$
M2	$D \rightarrow SC$	$N_2,$ `querySmartContrac(walletAddr,SP)`
M3	$SC \rightarrow D$	$N_2,$ {`TRUE	FALSE`}
M4	$D \rightarrow u$	If FALSE, request permission grant (redirect to `/auth?wallet,SP`)
M5	$D \rightarrow DB$	$N_3,$ `checkSessionInDB(agg_session_id,sp_session_id)`
M6	$D \rightarrow SP$	$N_4, \langle walletAddr, token, sp\_session\_id, ratio, realLocation \rangle$
M7	$SP \rightarrow u$	$N_4, \langle booking\_url, mainFare \rangle$

Figure 4.8: Booking Flow

- **Step 1:** After selecting an SP from aggregated results, the user triggers a booking request,

  `req(bookRide,bookingRequest)` $= \langle$`provider`, `walletAddr`, `sp_session_id`, `token`, `ratio`$\rangle$

  which is securely sent to the DApp (message M_1).

- **Step 2:** The DApp verifies if the selected SP already has permission granted by the user. It queries the smart contract (message M_2). - If permission = FALSE, the DApp prompts the user to grant permission via OAuth (message M_4). - If TRUE, the booking continues (message M_3).

- **Step 3:** The DApp validates session integrity by checking the stored session data for the given `agg_session_id` and `sp_session_id` in DB (message M_5).

- **Step 4:** After successful validation, the DApp sends a secure booking payload to the SP containing wallet address, token, SP session ID, ratio and real location (message M_6).

- **Step 5:** The SP computes the real fare using the ratio and actual location data. Then, instead of routing back through the DApp, the SP directly presents the `booking_url` and the `mainFare` to the user (message M_7).

Chapter 5

Implementation

5.1 Implementation of Registration Protocol

The registration protocol is implemented as a hybrid identity management system integrating blockchain technology with conventional web-based verification systems which links the user's pseudo-anonymous blockchain wallet and a validated email identity. To use the system, the user first needs to connect his wallet address using Metamask which serves as the unique digital identity of an user. Using a backend API it is checked if the user is registered, if not then the DApp prompts the user to input their personal details, including name, phone number, and email address. Upon submission, personal data is stored in the database, email verification is performed, and registration flag is recorded in blockchain, as illustrated in Algorithm 1.

Algorithm 1 Registration Protocol

function REGISTRATION(userData)
  userData ← req.data
  walletAddr, name, phone, email ← userData.walletAddr, userData.name, userData.phone, userData.email
  nameHash ← HMAC_SHA256(name, globalSalt)
  emailHash ← HMAC_SHA256(email, globalSalt)
  phoneHash ← HMAC_SHA256(phone, globalSalt)
  storePD$\big(\langle walletAddr, nameHash, emailHash, phoneHash\rangle\big)$ in database
  verificationCode ← generateVerificationCode()
  emailVerificationLink(email, verificationCode) via SMTP
  if verifyEmail(userInput) == TRUE **then**
    trigger blockchain registration process
    blockchainTransactionResult ← blockchainRegister(walletAddr, isRegistered=true, timestamp)
    if blockchainTransactionResult == TRUE **then**
      emit USERREGISTERED(walletAddr)
      store registration status in database as registered
    else
      return error
    end if
  end if
end function

5.1.1 Secure Personal Data Storage

Firstly, each personal data field is converted to lowercase and whitespace is trimmed. After that the data is processed using a cryptographic HMAC-SHA256 hashing algorithm, where a securely stored global salt value is used to enhance security. Finally the hashed data is stored on the database. This mechanism guarantees irreversible anonymization of user data in compliance with privacy protection principles.

5.1.2 Email Verification

After registration form submission, the user receives an email consisting of a unique verification code with a 24-hour predetermined validity period generated by the backend along with detailed instructions for completing the verification process via the Gmail SMTP service through the Nodemailer API. Once the user enters the verification code the system matches it with the hashed code stored in the database and upon successful verification the code is turned to null in the database to prevent reuse.

5.1.3 Blockchain-based Flag Storage

After successful email verification, the DApp prompts MetaMask to initiate a blockchain transaction that triggers the registerUser() function of the User Smart Contract and the contract stores the user's wallet address, registration flag, and timestamp immutably, creating a blockchain-based proof of registration. Upon successful transaction, the backend updates the user's status and logs the blockchain reference to the database. Figure 5.1 illustrates the Metamask prompt that appears after email verification at the registration phase.

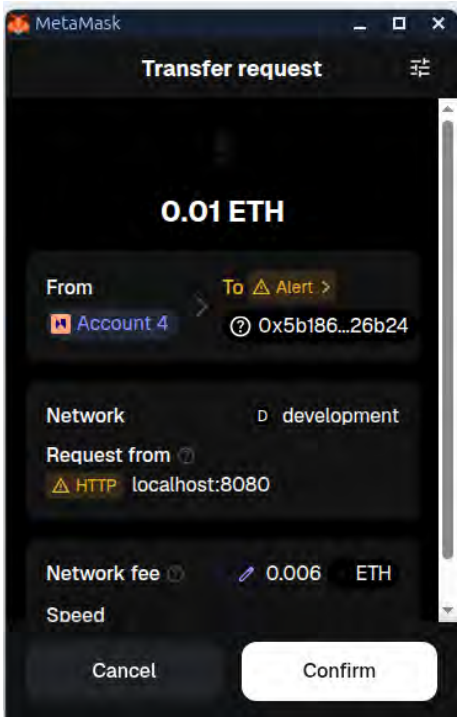


Figure 5.1: MetaMask Prompt During Registration

Through this design the system ensures that every user is both authenticated by email and registered with blockchain, thus creating a trust-based identity that is tamper proof.

5.2 Implementation of Login Protocol

During logins, the user needs to connects their MetaMask wallet and the DApp verifies the wallet address from the MongoDB records. If the address corresponds to a verified registration, the user is logged in directly, providing a secure, password-free login mechanism, as illustrated in Algorithm 2.

Algorithm 2 Login Protocol

```
1: function LOGIN(walletAddr)
2:     userData ← req.data
3:     walletAddr ← userData.walletAddr
4:     registrationStatus ← checkRegistrationInDB(walletAddr)
5:     if registrationStatus == NotRegistered then
6:         return error
7:     else
8:         emit USERLOGGEDIN(walletAddr)
9:         return success
10:    end if
11: end function
```

5.3 Implementation of Access Control Protocol

The Access Control Protocol ensures secure and user-centric interaction between the DApp and SPs. It enables protection of user data, preventing unauthorized access, and seamless account linking. Using this approach, users retain full control over data sharing and can revoke access at any time without exposing their private credentials. The protocol is implemented using the OAuth 2.0 framework, which standardizes authorization through token-based access delegation. The access control functionality is divided into two types of protocols including the permission grant, which allows the linkage of user's DApp account with his SP account, and the permission revoke protocol, which enables users to withdraw previously granted permissions.

5.3.1 Permission Grant Protocol

The permission grant protocol is implemented using three consecutive steps as including OAuth 2.0-based authorization, blockchain-based flag storage, and secure token storage in the database. as shown in Algorithm 3

5.3.1.1 OAuth 2.0-based Authorization

Registered users may connect their DApp account to SPs. If the user wants to connect to SPn, DApp initiates an OAuth authorization request by redirecting the user to the SPn's login page, passing the redirect URI and the wallet address as

Algorithm 3 Permission Grant Protocol

```
1: function grantPermission(walletAddr, SP)
2:     userData ← req.data
3:     walletAddr ← userData.walletAddr
4:     SP ← userData.SP
5:     redirect to SP for OAuth authorization
6:     authCode ← receiveAuthorizationCodeFromSP(SP)
7:     accessToken, refreshToken ← exchangeAuthCodeForTokens(authCode)
8:     permissionData ← ⟨walletAddr, SP, true⟩
9:     send permission data to smart contract
10:    blockchainResult ← updatePermissionOnBlockchain(permissionData)
11:    if blockchainResult == TRUE then
12:        encryptedAccessToken ← Encrypt(accessToken, K_DB)
13:        encryptedRefreshToken ← Encrypt(refreshToken, K_DB)
14:        storeTokensInDB(walletAddr, SP, encryptedAccessToken, encryptedRefreshToken)
15:        emit PermissionGranted(walletAddr, SP)
16:        ack ← ⟨Permission Granted, ∅⟩
17:        return ⟨Permission Granted, ∅⟩
18:    else
19:        return error
20:    end if
21: end function
```

parameters. Upon successful authentication, the SPn's backend issues an access token and a refresh token, encoded as JSON Web Tokens (JWTs) which are saved in SPn's database in encrypted format. The access token is short-lived (15 minutes) and used for direct API communication, while the refresh token allows the DApp to automatically obtain new access tokens as long as the user has not revoked permission.

5.3.1.2 Blockchain-based Flag Storage

After authentication, the DApp triggers a transaction through MetaMask that invokes the setServiceProvider() function within the deployed Smart Contract. This function accepts the user's wallet address, SP ID (a numerical identifier for a specific SP), and a boolean authorization flag indicating the connection status. Upon successful transaction, the data is stored in the blockchain which ensures that the user's connection to his SP account is tamper-proof. Since no personal data is stored on the chain it guarantees transparency and auditability without exposing sensitive credentials.

5.3.1.3 Secure Token Storage in the Database

After successful storage of the token in the SP database, the user is redirected back to the DApp's frontend and the tokens are forwarded to the aggregator backend. The backend encrypts the tokens, associates these encrypted tokens with the user's blockchain wallet address and stores them in the database

5.3.2 Permission Revoke Protocol

The permission revoke protocol enables users to withdraw previously granted permissions anytime if they want. When a user initiates revocation from SPn, the DApp sends a revocation request to the SPn and the SPn immediately invalidates both the Access Token and Refresh Token, ensuring that no further API calls can be made under the user's authorization. Then the blockchain flag storage step takes place in the same way as permission grant. The DApp backend then removes the corresponding token entry from its database. The protocol is illustrated in Algorithm 4.

Algorithm 4 Permission Revoke Protocol

```
function REVOKEPERMISSION(walletAddr, SP)
    userData ← req.data
    walletAddr ← userData.walletAddr
    SP ← userData.SP
    // Invalidate the SP's OAuth access token to prevent further API calls
    revokeTokenAPI(accessToken)
    permissionData ← ⟨walletAddr, SP, false⟩
    send permissionData to smart contract for revocation
    blockchainResult ← updatePermissionOnBlockchain(permissionData)
    if blockchainResult == TRUE then
        emit PERMISSIONREVOKED(walletAddr, SP)
        // Remove access tokens from the database
        removeTokens(walletAddr, SP)
        ack ← ⟨Permission Revoked, ∅⟩
        return ack
    else
        return error
    end if
end function
```

5.4 Implementation of Dynamic Mobility Discovery Protocol

In the dynamic mobility discovery user is able to see the nearby driver, active ride requests, and approximate fare from multiple SPs. In this phase, from both SPs and DApp location data is transferred while maintaining data confidentiality through DP. Its implementation consists of four distinguished steps including DApp session initialization, DP application in DApp, SP Response Generation, and Aggregation and Visualization, as shown in Algorithm 5

5.4.1 DAPP Session Initialization of DApp

The frontend of search page collects the user's current geolocation and desired destination using the browser's Geolocation API and the OpenStreetMap geocoding service. The search process begins when the user initiates a search request selecting the destination, preferred vehicle type, and the desired search radius around which he wants to see the data from the DApp interface. At first, the backend creates an aggregator session, identified by a unique session ID and stores it in the database.

This data is transmitted to the Backend and stored in the database associated with the respective session ID.

5.4.2 DP Application in DApp

Before sending user's location data to SPs, DApp Backend introduces random controlled noise using Geo-Indistinguishability to both source and destination latitude and longitude coordinates. Geo-Indistinguishability is an extension of LDP mechanism which specially focus on location data privacy ensuring that two adjacent locations are statistically identical within a distance-dependent privacy factor. For location-based system it offers stronger and more practical protection than traditional DP methods [11]. Formally, a randomized mechanism M satisfies ϵ-Geo-Indistinguishability if, for any two true locations $x_1, x_2 \in R^2$ and for any measurable set of outputs $S \subseteq R^2$,

$$P(M(x_1) \in S) \leq e^{\epsilon \cdot d(x_1, x_2)} \, P(M(x_2) \in S), \tag{5.1}$$

where $d(x_1, x_2) = \|x_1 - x_2\|_2$ denotes the Euclidean distance between the two locations, and ϵ is the privacy budget that controls the trade-off between privacy and utility.

5.4.2.1 Noise Generation Using Geo-Indistinguishability

The mechanism operates by sampling a noise radius r and direction θ according to the *planar Laplace distribution*. The radius r is drawn from a *Gamma* distribution parameterized by the privacy budget ϵ, as shown in Equation 5.2:

$$r \sim \text{Gamma}(k, \theta) \tag{5.2}$$

where the *shape parameter* $k = 2$ and the *scale parameter* $\theta = 1/\epsilon$. The Gamma distribution is a generalization of the exponential distribution. For integer k, a Gamma random variable can be generated as the sum of k independent exponential random variables with the same rate parameter ϵ. Accordingly, in the implementation, r is obtained by summing two independent Exponential(ϵ) random variables, as shown in Equation 5.3:

$$r = \frac{-\ln(u_1) - \ln(u_2)}{\epsilon} \tag{5.3}$$

where $u_1, u_2 \sim \text{Uniform}(0, 1)$. Each term $-\ln(u_i)/\epsilon$ represents one sample from an Exponential(ϵ) distribution, derived from the inverse transform sampling method. This method uses uniformly distributed random numbers to produce samples from the desired exponential distribution. The sum of these two exponential samples follows a Gamma($k = 2, \theta = 1/\epsilon$) distribution, which provides the desired random noise radius for the planar Laplace mechanism.

The direction of the noise is drawn uniformly across the circle, as given in Equation 5.4:

$$\theta \sim \text{Uniform}(0, 2\pi) \tag{5.4}$$

After deriving the polar displacement (r, θ), it is then converted to Cartesian coordinates and translated into geographic units using the constants M_{lat} and M_{lon},

which represent the meters per degree of latitude and longitude respectively:

$$M_{\text{lat}} = 111{,}320, \quad M_{\text{lon}} = 111{,}320 \cdot \cos\left(\frac{\pi \cdot \text{Lat}}{180}\right) \tag{5.5}$$

The final noisy coordinates (lat', lon') are obtained by applying the planar Laplace mechanism as:

$$lat' = lat + \frac{r \cdot \cos(\theta)}{M_{\text{lat}}} \tag{5.6}$$

$$lon' = lon + \frac{r \cdot \sin(\theta)}{M_{\text{lon}}} \tag{5.7}$$

5.4.2.2 Selection of the Privacy Budget (ϵ)

To select the value of ϵ, first a theoretical analysis is conducted using the standard probability theory for the Gamma distribution. Although each realization of r is random and unpredictable, but when a large number of samples are drawn, their average converges to a constant value, known as the expected value $E[r]$. For Equation 5.2, standard probability theory for the Gamma distribution calculates the expected $E[r]$ in meters, as shown in Equation 5.8.

$$E[r] = \frac{k}{\epsilon} \tag{5.8}$$

Practically, for dense urban areas such as Dhaka, perturbing a location point by 50–100 meters typically keeps the noisy point within the same block or street segment. From Equation 5.8 it can be said that $\epsilon = 0.020$–0.04 keeps the expected displacement $E[r] \approx 50$–100 meters. A higher ϵ value results in lower privacy, while a lower ϵ value provides stronger privacy. Therefore, to balance privacy and utility, in this system $\epsilon = 0.025$ is chosen, which corresponds to $E[r] \approx 80$ meters.

After theoretical analysis, to verify whether the practical results align with the theoretical expectations, 10,000 random points were generated using a uniform distribution, and the average noise was measured for different values of ϵ between 0.075 to 0.001. And the result shows $\epsilon = 0.025$ alters the location data by 79.9082909 meter on an average which reflects the proper balance between privacy and utility.

Applying DP ensures that the true location of the user is replaced by a noisy coordinate before being shared with the external SPs. Adding randomness in both distance and direction, and controlling usability by the choice of ϵ, the mechanism offers a formal guarantee of ϵ-Geo-Indistinguishability while maintaining practical utility.

5.4.3 SP Response Generation

The DApp backend forwards the DP coordinates, preferred vehicle type, and the desired search radius to the SPs. Upon receiving the API request from the DApp each SP conduct several actions

Table 5.1: Average Noise with Respect to Different ϵ Values

ϵ	**Avg Noise (km)**	**Avg Noise (m)**
0.75	0.002644374	2.644374489
0.50	0.003986142	3.986141655
0.25	0.008037443	8.037443008
0.10	0.020158426	20.15842629
0.075	0.026753470	26.75346989
0.05	0.039582932	39.58293211
0.025	0.079908291	79.90829093
0.01	0.197852590	197.8525902
0.0075	0.266785301	266.7853013
0.005	0.404573960	404.5739602
0.001	2.014364251	2014.364251

5.4.3.1 SP Session Initialization

At first, each SP creates an SP session, identified by a unique session ID and stores it to its own database.

5.4.3.2 Approximate Fare Calculation

In the next step, the SPs calculate trip distance using the haversine formula, and surge multiplier based on the driver–rider ratio within the radius. SPs have fixed base fare, per-kilometer rate based on the type of the vehicle.

The trip distance between the origin (lat_1, lon_1) and destination (lat_2, lon_2) is computed using the haversine formula as shown in Equation 5.9:

$$d = 2R \cdot \arctan\left(\frac{\sqrt{a}}{\sqrt{1-a}}\right) \tag{5.9}$$

where:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{5.10}$$

and

$$\Delta\phi = (\phi_2 - \phi_1), \quad \Delta\lambda = (\lambda_2 - \lambda_1), \quad R = 6371 \text{ km}.$$

The fare for a given vehicle type is determined by Equations 5.11–5.12.

$$F_{\text{total}} = \max\left((F_{\text{base}} + F_{\text{km}} \times D) \times S,\, F_{\text{min}}\right) \tag{5.11}$$

where:

- F_{base} is the fixed base fare of that vehicle type
- F_{km} is the per-kilometer rate
- F_{min} is the minimum fare threshold

- D is the trip distance in kilometers
- S is the surge multiplier

The surge multiplier depends on the ratio of active user requests to available drivers. The surge multiplier S is computed dynamically, as expressed in Equation 5.12:

$$S = \begin{cases} 0.75, & \frac{N_{\text{req}}}{N_{\text{drv}}} < 0.7 \\ 1, & 1.2 < \frac{N_{\text{req}}}{N_{\text{drv}}} \leq 1.5 \\ 1.25, & 1.5 < \frac{N_{\text{req}}}{N_{\text{drv}}} \leq 2.5 \\ 1.75, & \frac{N_{\text{req}}}{N_{\text{drv}}} > 2.5 \\ 1.0, & \text{otherwise} \end{cases} \tag{5.12}$$

Here, N_{req} denotes the number of user ride requests within the selected radius and N_{drv} represents the number of available drivers. This dynamic pricing model is designed to demonstrate real-world scenario of increasing fares during high demand and low driver availability. The approximate fare, distance and the surge multiplier is stored in the database associated to the specific session ID.

5.4.3.3 Applying DP in SP

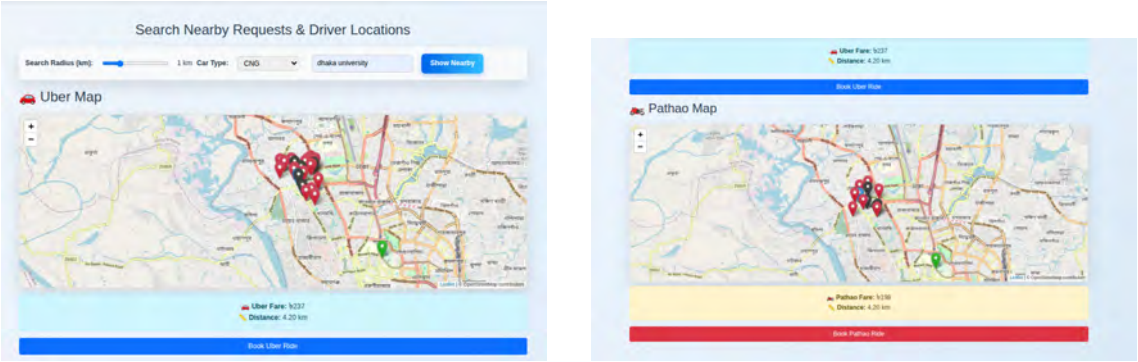
Before sending the available drivers and active requests within the selected radius the SPs utilize DP in the same way as DApp. Since we do not have access to real data, we have demonstrated this using synthetic datasets. Each dataset contains 2000 geospatial points within Dhaka, Bangladesh which replicates realistic ride request patterns since each point represents a potential pickup location which is randomly sampled from a uniform distribution bounded by latitude $[23.68, 23.89]$ and longitude $[90.33, 90.53]$ with no user identification.

5.4.3.4 Sending Response to DApp

Each SP sends a JSON response to DApp consisting the unique session ID, approximate fare, surge multiplier, differentially private available drivers and active requests data within the selected radius.

5.4.4 Aggregation and Visualization

Upon receiving response from each SP the DApp backend stores SP data to the database linking it with the specific aggregated session ID. The frontend renders the combined results using Leaflet.js, displaying both SP results in separate map views. The approximate fare from each SP is also displayed separately under each map.
For implementation purposes, we build two SPs - Uber and Pathao and Figure 5.2 represents the interface of the maps plottted with the data collected from the SPs where distinct color-coded markers represent:
Blue: The user's starting position.
Red: DP-Protected Active requests.
Black: DP-Protected Available drivers.
Green: The user's destination.



(a) Uber Map (b) Pathao Map

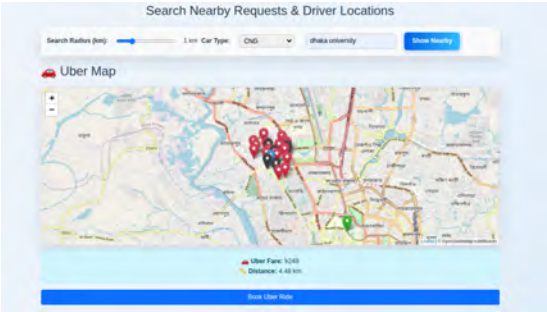
Figure 5.2: Visualization of Map Generated During Dynamic Mobility Discovery

5.5 Booking

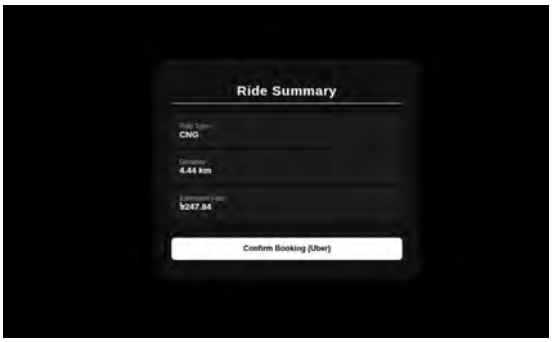
Once a user selects a preferred SP, the DApp verifies SP connectivity. If authenticated, the user is redirected to the SP's booking page along with the access token, wallet address, session details, real latitude and longitude of user's starting point and destination, and surge multiplier sent by that specific SP. Since the surge multiplier is fixed for each session, after sending the real location the fare does not change drastically.

Figure 5.3 depict searches performed from the same source to the same destination. But distance differs in Figure 5.3a and Figure 5.3c, due to the application of DP while searching, which perturbs the location data. In contrast, Figures 5.3b and 5.3d shows identical distances because the actual user location is shared with SPs during the booking process.

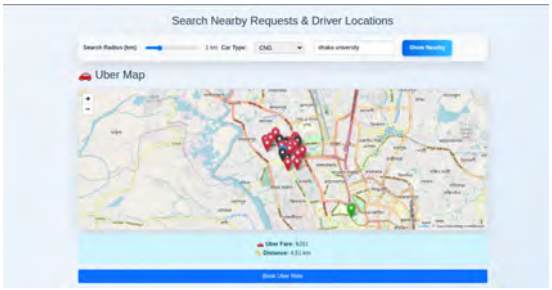
As shown in Algorithm 6, the booking protocol manages secure coordination between the user, decentralized application, and SPs while validating permissions and session integrity before ride confirmation.



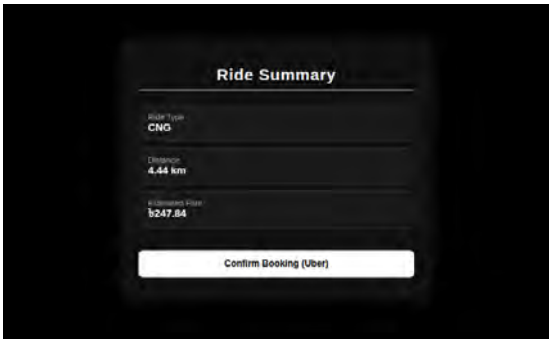
(a) Search 1



(b) Book 1



(c) Search 2



(d) Book 2

Figure 5.3: Comparison of Distance While Searching and Booking

Algorithm 5 Dynamic Mobility Discovery Protocol

function INITIALIZESESSION(userLocation, destination, carType, radius)
  userData ← req.data
  lat, lon ← userData.lat, userData.lon
  destination ← userData.destination
  carType ← userData.carType
  radius ← userData.radius
  aggregatorSessionID ← generateSessionID()
  storeSessionInDB(aggregatorSessionID, created_at)
  return aggregatorSessionID
end function

function APPLYDIFFERENTIALPRIVACY($Start\_lat, Start\_lon, Dest\_lat, Dest\_lon$)
  $u_1, u_2 \sim \text{Uniform}(0, 1)$
  $r = \frac{-\ln(u_1)-\ln(u_2)}{\epsilon}$
  $\theta \sim \text{Uniform}(0, 2\pi)$
  //Compute metric coefficients for latitude and longitude scaling
  $M_{\text{lat}} = 111{,}320$
  $M_{\text{lon}} = 111{,}320 \cdot \cos\left(\frac{\pi \cdot Start\_lat}{180}\right)$
  //Convert polar noise (r, θ) to Cartesian coordinates
  $Start\_lat' \leftarrow Start\_lat + \frac{r \cdot \cos(\theta)}{M_{\text{lat}}}$
  $Start\_lon' \leftarrow Start\_lon + \frac{r \cdot \sin(\theta)}{M_{\text{lqn}}}$
  Apply the same mechanism to destination coordinates
  return $(Start\_lat', Start\_lon', Dest\_lat', Dest\_lon')$
end function

function SENDQUERYTOSPS(lat′, lon′, destination, carType, radius)
  for each SP **do**
    send DP coordinates, carType, and radius to SP
    await SP response with estimated fare, driver locations, and active requests
  end for
end function

function RECEIVESPRESPONSE(SPResponse)
  for each SPResponse **do**
    parseResponse(SP_session_id, DP_driver_locations, DP_nearby_users, approxFare, ratio)
    storeMappingInDB(aggregatorSessionID, SP_session_id, ratio)
  end for
  aggregatedResults ← aggregateAllResponsesFromSPs()
  return aggregatedResults
end function

Algorithm 6 Booking Protocol

```
1: function BOOKRIDE(walletAddr, SP, agg_session_id, sp_session_id, token, ratio)
2:     userData ← req.data
3:     walletAddr ← userData.walletAddr
4:     SP ← userData.SP
5:     agg_session_id ← userData.agg_session_id
6:     sp_session_id ← userData.sp_session_id
7:     token ← userData.token
8:     ratio ← userData.ratio
9:     send ⟨agg_session_id, SP, walletAddr, sp_session_id, token, ratio⟩ to DApp
10:    verify permission for SP
11:    permission ← querySmartContract(walletAddr, SP)
12:    if permission == FALSE then
13:        redirect user to OAuth for permission granting
14:    else
15:        validate session integrity
16:        sessionValid ← checkSessionInDB(agg_session_id, sp_session_id)
17:        if sessionValid == TRUE then
18:            send ⟨walletAddr, token, agg_session_id, sp_session_id, ratio, realLocaion⟩ to SP
19:        else
20:            return error  (invalid session)
21:        end if
22:    end if
23:    bookingURL, mainFare ← receiveBookingDetailsFromSP()
24:    return ⟨bookingURL, mainFare⟩ to user
25: end function
```

Chapter 6

Result Analysis and Discussion

6.1 Performance Evaluation

The performance evaluation of our system is focused on the API testing. It was conducted using Apache JMeter [51]. API testing is conducted using two different approaches respectively Gradual Thread Increase Test (GTIT) and Spike Test (ST).

6.1.1 API Testing Tool

Apache JMeter is an open source performance testing tool. It is widely used for load testing and analyzing the performance of web applications, databases, and APIs. It is popular because of its flexibility, scalability and ability to simulate a variety of user interactions. So,it becomes an ideal choice for both functional and performance testing. JMeter allows the simulation of a heavy load on servers and networks to test their strength and analyze overall performance under varying load conditions. In the context of API performance testing, JMeter is highly effective due to its ability to simulate concurrent API requests, capture response times, error rates and throughput. The tool supports multiple protocols, such as HTTP [1], WebSocket [44]. So, it is versatile for testing RESTful APIs, SOAP services and other backend integrations. For our testing, JMeter was employed to assess the behavior of APIs under different traffic loads and stress conditions to ensure that the system could handle real world usage and scale efficiently.

6.1.2 API Testing Approach

The performance evaluation by JMeter involved two distinct tests,the Gradual Thread Increase Test and the Spike Test. To evaluate how well our system performs under different conditions of load, firstly, GTIT simulated a gradual increase in the number of concurrent virtual users in the same ramp up period interacting with the API to analyze how the system scales with a steadily growing load. In contrast, ST subjected the system to sudden, high traffic volumes without any ramp-up period. It focused on the system's ability to recover from unexpected traffic surges.

6.1.2.1 System Performance Under Increasing Load

The system handled the load efficiently with minimal increase in response time. As the number of virtual users increased, the response times was in acceptable limits. It suggests that the performance of the system was relatively stable. However, as soon as the load continued to increase, a subtle performance degradation had started to appear. After that, response time gradually increased with significant jumps as the system encountered higher thread counts. It indicates that the system was starting to experience pressure. The data from the test suggests that while the system was scalable up to a point. Its ability to handle sustained traffic was limited beyond a certain threshold which was noticed by the drop in throughput at higher virtual users.

6.1.2.2 System Performance Under Sudden Traffic Surges

Initially, as the thread count increased, the system handled the load without any issues, with both the average response time and p95 response time staying relatively low. However, as the traffic increased further both the average response time and p95 response time began to rise significantly. This sharp increase in response times reflected growing pressure on the system as the load increased, and it became clear that the system was struggling to maintain efficiency under the higher number of concurrent users. The increase in both the average and p95 response times highlighted the system's difficulty in scaling effectively under the sudden surge in traffic. The larger increase in p95 response time suggests that a small percentage of requests took significantly longer to process due to resource contention or backend processing delays.

6.1.3 System Configuration for Testing

The performance tests were conducted on a Dell Inspiron 3501 running Ubuntu 22.04. The system is powered by an 11th Gen Intel® Core™ i5-1135G7 processor with 4 cores and 8 threads. It was capable of a maximum clock speed of 4.2 GHz. It has 7.5 GB of RAM and a 96 GB SSD for storage, providing a certain capacity for handling the workload during the test. The laptop was connected to a WiFi network with an IP address of 10.158.124.45.

6.2 Test Set Up

This section provides an overview of the GTIT and ST test setup utilized to evaluate the system's performance under different load conditions.

6.2.1 GTIT Objective

The GTIT was designed to evaluate the system's performance as the number of concurrent users gradually increased. The concurrent users were virtually simulated by JMeter threads. With a 20-second ramp-up period, the number of threads was

incrementally increased from 50 to 700. The goal of this test was to simulate a real-world scenario where the number of users steadily increases and to identify at which point the system would begin to experience performance degradation. The test specifically focused on how the system would handle progressively increasing load over time.

6.2.2 ST Objective

The ST was designed to evaluate the performance of our system under sudden traffic volumes without any ramp-up period (e.g, 0 second) . The test began with 0 to 200 threads to assess how the system reacts and recovers to an immediate load spike. And also to evaluate whether it can maintain efficient response times under such conditions.

6.3 Statistical Analysis

This section provides an in-depth analysis of the statistical trends observed during the performance tests, focusing on key metrics such as response times, throughput, and system scalability under varying load conditions and sudden traffic surges.

6.3.1 Response Time Trends and Bottlenecks Under Gradual Load

One of the most noticeable trends in the GTIT was the increase in response time as the number of concurrent virtual users increased. In Figure 6.1, initially, the system's response time was around 100–150 ms. However, when it crossed 500 threads, response time started to grow gradually. And finally reaching up to 1000 ms at the highest load levels of 720 virtual users. The gradual rise in response times suggests that the system was being pushed closer to its resource limits. As a result, latency spikes and slower processing times are becoming visible. This performance degradation indicates potential bottlenecks, more specifically in the API and backend systems. As more requests were processed, resource contention likely became an issue which led to longer wait times for users to get response. These fluctuations highlight the difficulties the system faced when trying to maintain consistent performance under heavy load.

6.3.2 Throughput Trends and System Scalability Under Gradual Load

Throughput increased steadily with the gradual load. The system showed good scalability, with throughput climbing consistently as thread counts increased. However, the rate of increase started to increase rapidly after 720 threads. This observation trend suggests that the system may soon struggle further at higher loads. In Figure 6.1, at lower loads, the system was able to maintain a higher throughput, handling between 7 to 10 RPS at 50 threads. As the load increased to 400 and 600 threads,

throughput climbed to 55–65 RPS and 90–100 RPS, respectively. It maximizes 150-175 RPS at thread 700. Beyond 750 threads, throughput began to level off, even though response times continued to rise. It meant that the system could no longer handle additional load effectively.

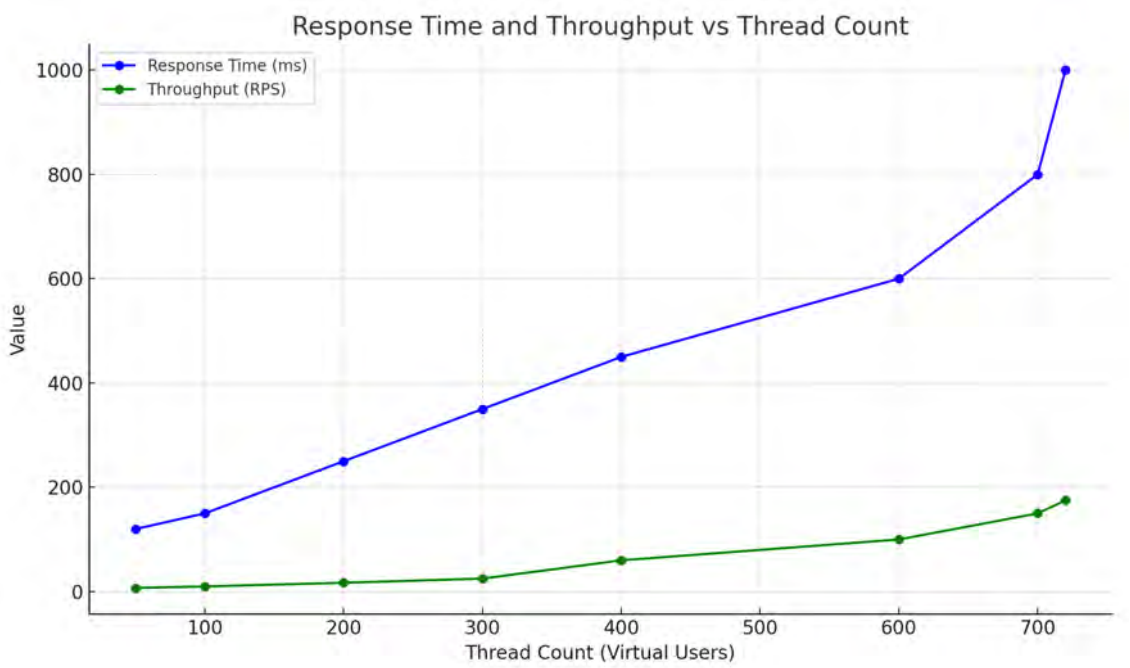


Figure 6.1: Gradual Load Testing

The data suggests that the system was scalable up to a point. But, it became clear that its ability to handle traffic was limited beyond a certain threshold. This is further evidenced by the drop in throughput at higher thread counts.

6.3.3 Response Time Trends Under Sudden Traffic Surges

The system handled under sudden traffic surges without any struggle as the thread count increased from 0 to 50 threads. Both the average response time and p95 response time was showing relatively low. From Figure 6.2, during this part of testing, the average response time was measured at 50 ms. And the p95 response time was also 50 ms. It indicated that the system was processing requests effectively and efficiently without noticeable delays. The average response time and p95 response time significantly rose from 50ms to 1435 ms and from 50 ms to 2130 ms when the virtual users moved from 50 to 100 threads. From this sharp increase in both the response times reflected growing pressure on the system as the load increased in sudden traffic. This was a clear indication of the system's struggle under sudden traffic surges to maintain efficiency under the higher number of concurrent users as threads.

The most dramatic rise in response times occurred between 100 and 200 threads, where the average response time rose to 2350 ms, and the p95 response time also reached 3200 ms. Though most requests were still handled within a reasonable period, a small percentage (e.g, 5 out of 100) of requests faced significant delays because of the system limitations.

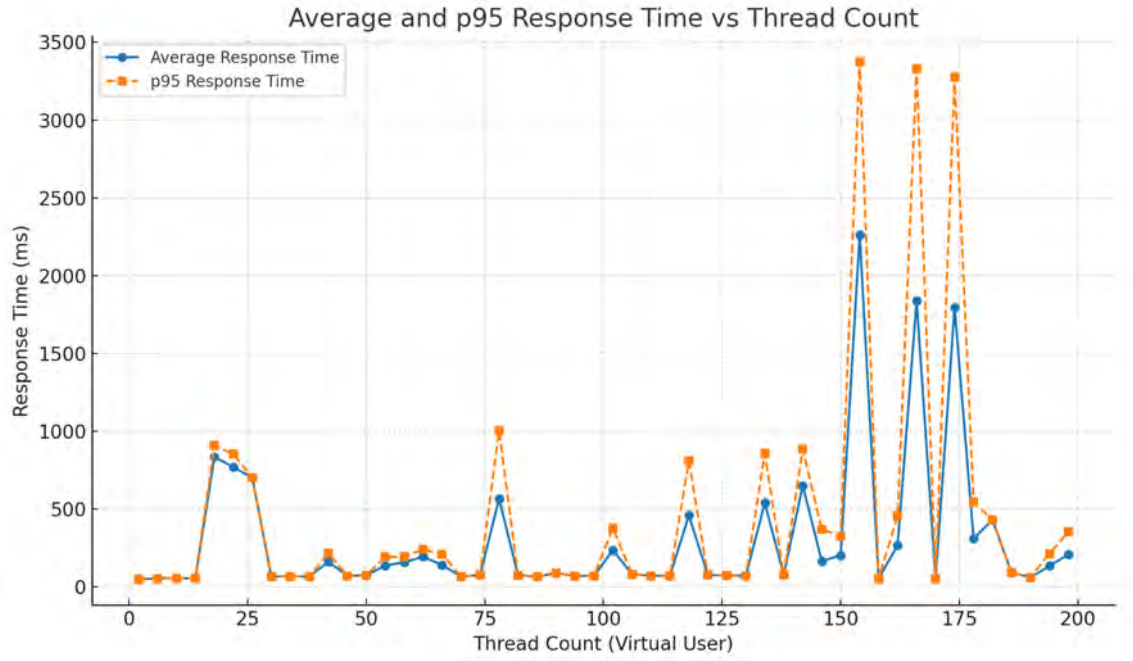


Figure 6.2: Spike Testing

Overall, the performance evaluation through the GTIT and the ST revealed valuable insights about the system's ability to handle increasing loads and sudden traffic surges. Both tests pointed to the system's limitations in handling sustained high traffic and sudden spikes, indicating areas for improvement in infrastructure, scalability, backend processing and database optimization to maintain efficient performance under load.

6.4 Final Design Adjustment Based on Performance Evaluation

The performance evaluation through the GTIT and ST tests has revealed several areas where the system can be improved.

6.4.1 Storage Management

The need for storage management arises when the performance degradation after 700 virtual users in gradual load and increased latency at high thread counts are visible.The free version of MongoDB is responsible for this. This free version has limitations in throughput, database size and scalability.[52] As a result, slower query response had been found. To mitigate, streamlining the database queries and improving resource management are very important steps. Upgrading to a more robust MongoDB plan can help to overcome these limitations. And it will also enhance system performance.efficiency.

6.4.2 Scalability Enhancements

Until the load exceeded 700 threads, the system demonstrated good scalability. To address this, the test of system's infrastructure should be upgraded. So that it will allow for better load balancing, specially more efficient handling of concurrent requests in the system and enhanced scaling capabilities.

6.4.3 API Optimization

Enhancing the API to better manage resources and handle more concurrent requests will be key for the performance degradation under higher load that was observed, . Caching, request prioritization in a branch, and optimizing the response handling will help in maintaining performance under gradual loads and sudden surges.

6.5 Analysis of Design Solutions

This section provides a detailed analysis of the design solutions implemented to address the identified privacy threats and ensure compliance with privacy regulations in our system. Each threat that we identified during the threat modeling process, is analyzed and the respective mitigation strategies are described to demonstrate how they protect user data and maintain system integrity.

6.5.1 Mitigation Strategies for Identified Threats

In our system, we identified six primary threats in the threat modeling process and have implemented various countermeasures to mitigate them. These threats, as discussed in Section 3.2.1, are categorized using the LINDDUN framework, which includes Linkability, Identifiability, Disclosure of Information, Unawareness, Non-compliance ad an adittion threat of Session Integrity. Our system mitigates these threats effectively.

6.5.1.1 Linkability (T1)

Linkability refers to integration of several data points or actions with the same entity to learn more about an individual or group. Since an aggregated platform exchange information with multiple SPs, linkability is a prime concern here. The proposed system mitigates this threat by applying cryptographic hashing, anonymization, and DP. In the DApp, all personally identifiable information (PII) including name, email, and phone number is processed using HMAC-SHA256 with a platform specific salt, minimizing the risk of re-identification via reverse engineering. Moreover, in Section 5.4 no user-identifiable information is transmitted, and the user's shared location coordinates remain protected through DP using the mechanism of Subsection 5.4.2. SPs also reply with DP-protected active request and driver information using the method of Subsection 5.4.3. Since, Geo-indistinguishability is applied to the location co-ordinates data from both ends, an individual users cannot be re-identified based on their activity or location patterns through linkability.

6.5.1.2 Identifying (T2)

Identifying refers to inferring the true identity of a user behind any action or data. To mitigate the threat of identifiability, data collection should be minimized, and aggregation should occur at a higher level or the data should be anonymized before aggregation [46]–[48]. In the proposed system, only necessary data is exchanged between DApp and SP minimizing availability of personal information for potential reidentification. Moreover, in on the blockchain only the essential flags is stored. In the dynamic mobility discovery protocol, it is carefully designed that the DApp or SP doesn't share any user identification which is critically implemented in Section 5.4. Moreover, by applying DP it ensures 2 layer of protection.

6.5.1.3 Disclosure of Data (T3)

Disclosure of Data refers to unauthorized access or leakage of private information. Our system integrates multiple techniques to ensure data confidentiality and integrity across all communication and storage layers. All frontend to backend communications occur over HTTPS secured with Transport Layer Security (TLS), which provides end-to-end encryption, effectively preventing man-in-the-middle (MITM) attack. For email verification, the system employs Simple Mail Transfer Protocol (SMTP) with enforced TLS encryption via Nodemailer in Subsection 5.1.2. This ensures that verification codes are transmitted securely, which gives protections against spoofing. At the data layer, authentication tokens are stored in encrypted form within MongoDB in Subsection 5.3.1.1. These mechanisms help to ensure that user information remains protected throughout its lifecycle.

6.5.1.4 Unawareness and Unintervenability (T4)

Unawareness and Unintervenability refer to users' lack of transparency and control regarding the usage of their personal data. Our system ensures that users remain aware and in control of their data at every stage by blockchain based authorization, and consent mechanisms. Use of Blockchain enhances transparency and user control by recording authorization flags immutably on the blockchain. Each permission granted or revoked is logged as a verifiable on-chain event, ensuring that users can independently verify their consent status, as shown in Subsection 5.3.1.2. Users can revoke previously granted permissions at any time. The blockchain flag is also updated to reflect the access permission states, ensuring transparency. Moreover, users are required to authenticate and grant permissions to SPs before booking initiates, ensuring full awareness of the data exchange process.

6.5.1.5 Non-compliance (T5)

Non-compliance refers to failure in adhering to privacy regulations such as GDPR, resulting in improper data management, unauthorized data sharing, and violation of user rights.[48] The OAuth 2.0 authorization flow, implemented in Subsection 5.3.1.1, ensures that user consent is explicitly obtained before any data sharing takes place. Users are required to authenticate and grant permission for their data

to be shared with the SPs, ensuring compliance with privacy regulations. Additionally, users retain full control over their data, as they can revoke access at any time, reinforcing the system's adherence to user rights. To ensure data privacy, DP is employed in location-sharing protocols in Subsection 5.4.2. The user's location data is perturbed using Geo-Indistinguishability, ensuring that even if shared data is intercepted or analyzed, individual users cannot be re-identified. This provides a formal privacy guarantee, preventing unauthorized access to user data, and ensuring the system remains compliant with data protection regulations. By not storing any personally identifiable information (PII) on the blockchain and only using anonymized location data for processing, the system complies with the GDPR's principles of data minimization and purpose limitation.

6.5.1.6 Session Integrity (T6)

Session integrity ensures that user interactions and data exchange within the system are protected from unauthorized tampering or interference.[49] The proposed system mitigates this threat by using secure session management, robust validation processes, and blockchain technology to safeguard session integrity. The OAuth 2.0 protocol, described in Subsection 5.3.1.1, ensures that each session is authenticated through a token-based authorization process. Any unauthorized access or modification attempts would result in the invalidation of the access and refresh tokens, effectively ending the malicious session. This prevents session hijacking or impersonation. In Subsection 5.3.1.2 ensures that all session states are securely recorded and tamper-proof. When a user connects to an SP, this permission flag is stored in the blockchain, guaranteeing that no unauthorized alterations can take place. The use of a decentralized ledger adds an extra layer of transparency, ensuring that any changes to the session data are publicly verifiable.

6.5.2 Functional Requirements Implementation

In this section, we will discuss how the functional requirements outlined in the previous section were implemented and met within the proposed decentralized, privacy-preserving ride-sharing platform. We used MetaMask as the Web3-compatible wallet and Ethereum as the blockchain for user registration, consent management, and interaction with SPs . The following details outline how each functional requirement was addressed.

6.5.2.1 Decentralized Registration and Identity Binding

The system enables users to register via a Web3-compatible wallet, ensuring the binding of a verifiable real-world identity with a blockchain-based digital identity. This is achieved through the integration of the MetaMask wallet in Subsection 5.3.1.2. After successful authentication via OAuth 2.0, the user's wallet address is associated with their identity, ensuring seamless, decentralized identity binding.

6.5.2.2 Consent-Driven Access Control

Users are given the ability to grant or revoke data access permissions selectively. This is implemented through 5.3.1.1, where users authenticate via OAuth and consent to granting access to their data. Additionally, Subsection 5.3.1.2 records these consent flags immutably on the blockchain, ensuring that the user's consent status is transparent and auditable, meeting the system's consent-driven access control requirements.

6.5.2.3 OAuth-Based Token Management

OAuth flows are implemented for securely acquiring and managing access tokens and refresh tokens for each SP. Subsection 5.3.1.1 facilitates this process, with the DApp requesting authorization from the user and receiving access and refresh tokens encoded as JWTs. The tokens are then securely stored in the SP's database and associated with the user's wallet address, ensuring the secure management of access rights to protected data.

6.5.2.4 Session-Oriented Query Processing

To ensure consistency during the entire user journey, the system initializes a unique session upon the user query and persists it through the ride booking phase. This is achieved in Subsection 5.4.1, where the backend creates a unique session ID. This session ID is then linked to user queries and their subsequent interactions with the SP, ensuring that the session remains consistent throughou

6.5.3 Privacy and Security Requirements Implementation

In this section, we will discuss how our system ensures the confidentiality, integrity, and regulatory compliance of user data. Our system integrates multiple privacy-preserving and security-enhancing mechanisms throughout its implementation. Each privacy and security requirement has been systematically addressed within specific modules of the system, as detailed in Chapter 5, ensuring that user consent, secure data exchange, and protection against unauthorized access are maintained at every operational stage.

6.5.3.1 DP for Data Protection

To mitigate risks of T1 and T2, DP techniques are applied to protect user's senstive data such as location data. This is implemented in Subsection 5.4.2, where Geo-Indistinguishability adds noise to the user's location coordinates before they are transmitted to the SPs. This ensures that individual user data cannot be re-identified, even when aggregated data is studied.

6.5.3.2 Secure Data Sharing

To prevent T3, private data is hashed before storage (e.g, namehash, emailhash) . OAuth 2.0 ensures that only authorized SPs can access the user's data, which is further protected by blockchain technology. In Subsection 5.3.1.3, tokens are stored securely in the backend database and their access is regulated by the OAuth mechanism. Additionally, Subsection 5.3.1.2 ensures that sensitive data access remains secure and transparent.

6.5.3.3 Transparent Consent Mechanisms

The system provides users with clear privacy policies and allows them to authorize, grant and revoke data consent at any time. This is implemented in Subsection 5.3.1.1, where users are explicitly asked to grant consent before booking. Subsection 5.3.1.2 provides transparent proof of consent as a flag and allowing users to withdraw consent through the system.

6.5.3.4 Regulatory Compliance

To ensure compliance with privacy regulations such as GDPR, the system provides mechanisms for user data revocation and access management. Users can revoke consent at any time, and the DApp immediately invalidates both the access and refresh tokens, as described in Subsection 5.3.1.3. These features meet the GDPR requirements for user consent withdrawal and data access rights by flags in Subsection 5.3.1.2.

6.5.3.5 Secure Session Management

To address T6, the system uses time-limited, session tokens for secure redirection to SPs. All communications are conducted using secure HTTPS protocol. Additionally, OAuth is applied to ensure that only the authorized user can complete the redirection and perform actions on the SP's platform. This secures the session from hijacking or spoofing.

6.5.4 Achievement of Research Objectives

This section highlights how each of the stated research objectives has been successfully achieved through the design and implementation of our privacy preserving aggregated ride sharing system.

6.5.4.1 Development of Aggregated Ride-Sharing Platform

To address RO1, we successfully developed an aggregated ride-sharing platform that integrates data from multiple SPs into a single unified system. This aggregation allows users to view and compare ride options from different providers within one interface, enabling them to choose the most efficient and cost effective ride. Beyond

user convenience, the system also generates valuable insights such as demand concentration, and mobility trends. By combining multiple data, the platform enhances accessibility, while ensuring that all user data flows through secure, consent-driven mechanisms.

6.5.4.2 Publication of Traffic Insights with Privacy Protection

For RO2, our system provides insights through showing nearby users and aggregated data visualization features, which reflect real time mobility trends, local demand intensity, and ride availability across different regions. These insights reveal traffic and congestion patterns, which help users in making more efficient ride choices. To ensure privacy during this process, DP mechanisms are applied to all the location data. This guarantees that while collective insights are published for public benefit, no individual user's location can be traced, ensuring privacy protection.

6.5.4.3 Decentralized Consent Management

In fulfillment of RO3, we implemented a blockchain based consent management framework. This allows users to transparently register, grant, and revoke permissions with SPs. The use of smart contracts ensures immutability, and verifiable accountability in every consent transaction which significantly enhance user trust and system transparency.

6.5.4.4 Performance and Functionality Evaluation

To achieve RO4, the system's performance and functionality were evaluated through load testing using Apache JMeter and structured security analysis. Gradual and spike load tests demonstrated stable performance under moderate concurrency, confirming efficient API response and session handling. Functionally, the platform achieved seamless integration between aggregator, SPs, and blockchain components, ensuring data consistency and transaction transparency. Security was validated through LINDDUN threat modeling, addressing linkability, data disclosure threats. Overall, the evaluation confirmed that the system performs reliably, scales efficiently, and maintains robust privacy-preserving functionality suitable for real-world deployment.

6.6 Comparisons and Relationships

Compared to existing blockchain based and privacy preserving ride sharing solutions, our proposed system presents a significant advancement by introducing a decentralized aggregated ride sharing platform that integrates multiple SPs under a privacy preserving framework. While prior works have implemented blockchain primarily for decentralization and transparency, they lack data aggregation and cross platform interoperability. Similarly, studies incorporating DP focus only on obfuscating individual trip or location data without aggregating multiple providers. Our proposed platform uniquely combines blockchain-based auditability with DP driven

location perturbation, ensuring both transparency and strong re-identification resistance. Furthermore, unlike existing systems, our design supports aggregated analytical result publication such as traffic insights and demand insights while preserving individual privacy. This dual-layered integration of blockchain and DP not only bridges the research gap identified in prior literature but also establishes a novel, scalable, and privacy preserving foundation for decentralized mobility aggregation.

Chapter 7

Conclusion

The existing ride sharing platforms face significant challenges related to data privacy, security and user control over personal information. The centralized nature of these systems exposes sensitive user data to potential breaches and misuse, compromising privacy. Additionally, the lack of secure unified platform, where users can get proper traffic insights from multiple SPs leads to poor service and user experiences. In response to these challenges, we designed a platform that integrates blockchain and DP to ensure secure, transparent, and user-centric data handling. The use of blockchain technology facilitates decentralized identity management and consent control, while DP ensures that users' location data is protected by introducing controlled noise, preventing re-identification. Key technical features include the application of Geo-Indistinguishability for location data privacy, OAuth 2.0 for secure authorization flows.

7.1 Summary of Findings

Analysis of existing research is conducted using 5 parameters including decentralization, aggregation, strong re-identification protection, data insight presentation, and optimized data storage. The study reveals that the no existing research meets all these criteria. On the contrary, the proposed achieves decentralization by integrating blockchain-based flag storage, ensures aggregation of data from multiple SPs, provides strong re-identification protection through Geo-Indistinguishability DP, enables data insight presentation, and supports optimized data storage by maintaining large and sensitive data securely in the off-chain database. Moreover, performance tests highlighted the platform's scalability of the system. Overall, the research demonstrated that integrating blockchain and DP and OAuth offers a robust solution for enhancing privacy, scalability, and user experience in ride-sharing systems.

7.2 Contributions to the Field

Our research introduces a novel approach to ride-sharing platforms by integrating decentralized architecture, blockchain, and DP. The key contributions of this work are as follows:

- It provides a privacy-preserving solution that ensures user data protection while enabling data aggregation from multiple SPs.

- The combination of OAuth and blockchain enhances transparency and auditability in user consent management.
- Utilizing DP with controlled noise safeguard sensitive location data guaranteeing that its usability is unaffected.
- Storing sensitive and large data securely in the database achieves a balance between user privacy and operational cost.

Overall, it represents a valuable contribution to the development of secure, efficient, and user-centric ride-sharing technologies.

7.3 Recommendations for Future Work

Future work for our privacy preserving aggregated ride sharing system will focus on some key directions. First, we aim to work on IP address monitoring, rate limiting, and automated blocking mechanisms in the DApp backend and API gateway to detect and mitigate suspicious behaviour, failed auth attempts, token replay, and large-scale scraping. This will improve session integrity, prevent system overload from excessive concurrent search or booking requests through adaptive rate limiting and request throttling strategies. Second, we plan to enhance the security of the user data, for example, integrating user-specific salts for hashing PII while storing in database. This enhancement using a per user salt derived from a high entropy secret such as the HMAC of the wallet address and server secret will significantly increase resistance to precomputation attack and reduces cross platform linkability. We also plan to conduct comprehensive security testing, ensuring maximum system resilience. Additionally, we aim to integrate AI based recommendation system which will automatically analyze aggregated ride data to suggest the most efficient and cost-effective rides as it will reduce user decision effort. While the current implementation primarily focuses on the users of the ride-sharing platforms, future work will extend its functionality to address the perspective of the drivers as well. Furthermore, the system is intended to be deployed in real-world settings in order to verify its functionality, interoperability, and overall usability.

Bibliography

[1] Mozilla, *Mdn web docs*, Accessed: Oct. 14, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web, 1998.

[2] L. Sweeney, "K-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[3] N. Li, T. Li, and S. Venkatasubramanian, "T-closeness: Privacy beyond k-anonymity and l-diversity," in *Proceedings of the International Conference on Data Engineering*, 2007, pp. 106–115.

[4] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, 2007.

[5] M. E. Nergiz, M. Atzori, and C. Clifton, "Hiding the presence of individuals from shared databases," in *SIGMOD '07*, New York, NY, USA: ACM, 2007, pp. 665–676.

[6] S. Ganta, S. Kasiviswanathan, and A. Smith, "Composition attacks and auxiliary information in data privacy," in *Proceedings of the 2008 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 265–273.

[7] T. Wang and L. Liu, "From data privacy to location privacy," in *Machine Learning in Cyber Trust: Security, Privacy, and Reliability*. 2009, pp. 217–246.

[8] G. Cormode, D. Srivastava, N. Li, and T. Li, "Minimizing minimality and maximizing utility: Analyzing method-based attacks on anonymized data," *Proc. VLDB Endow.*, vol. 3, no. 1–2, pp. 1045–1056, 2010.

[9] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, and W. Joosen, "A privacy threat analysis framework: Supporting the elicitation and fulfillment of privacy requirements," *Requir. Eng.*, vol. 16, pp. 3–32, Mar. 2011.

[10] R. C.-W. Wong, A. W.-C. Fu, K. Wang, P. S. Yu, and J. Pei, "Can the utility of anonymized data be used for privacy breaches?" *ACM Transactions on Knowledge Discovery from Data*, vol. 5, no. 3, 16:1–16:24, 2011.

[11] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proc. ACM SIGSAC Conf. on Computer and Communications Security (CCS)*, 2013, pp. 901–914.

[12] D. Fett, R. Küsters, and G. Schmitz, "A comprehensive formal security analysis of oauth 2.0," in *Proc. ACM SIGSAC Conf. on Computer and Communications Security (CCS)*, 2016, pp. 1204–1215.

[13] H. To, K. Nguyen, and C. Shahabi, "Differentially private publication of location entropy," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2016, pp. 1–10.

[14] R. Yang, G. Li, W. C. Lau, K. Zhang, and P. Hu, "Model-based security testing: An empirical study on oauth 2.0 implementations," in *Proc. 11th ACM on Asia Conf. on Computer and Communications Security (ASIA CCS)*, 2016, pp. 651–662.

[15] S. Wang, R. Sinnott, and S. Nepal, "Privacy-protected place of activity mining on big location data," in *2017 IEEE International Conference on Big Data (Big Data)*, IEEE, 2017, pp. 1101–1108.

[16] T. Zhu, G. Li, W. Zhou, and S. Y. Philip, *Differential Privacy and Applications*. Springer, 2017.

[17] A. Alnemari, S. Arodi, V. R. Sosa, *et al.*, "Protecting infrastructure data via enhanced access control, blockchain and differential privacy," in *Critical Infrastructure Protection XII: 12th IFIP WG 11.10 International Conference, ICCIP 2018, Arlington, VA, USA, March 12-14, 2018, Revised Selected Papers 12*, Springer, 2018, pp. 113–125.

[18] E. Androulaki, A. Barger, V. Bortnikov, *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ACM, 2018, p. 30.

[19] J. Golosova and A. Romanovs, "The advantages and disadvantages of the blockchain technology," in *2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, 2018, pp. 1–6.

[20] K. Kato, Y. Yan, and H. Toyoizumi, "Blockchain application for rideshare service," in *2018 8th International Conference on Logistics, Informatics and Service Sciences (LISS)*, IEEE, 2018, pp. 1–5.

[21] K. Kotobi and S. G. Bilen, "Secure blockchains for dynamic spectrum access: A decentralized database in moving cognitive radio networks enhances security and user access," *IEEE Vehicular Technology Magazine*, vol. 13, no. 1, pp. 32–39, 2018.

[22] M. Baza, N. Lasla, M. M. Mahmoud, G. Srivastava, and M. Abdallah, "B-ride: Ride sharing with privacy-preservation, trust and fair payment atop public blockchain," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1214–1229, 2019.

[23] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *IEEE Access*, vol. 7, pp. 117 134–117 151, 2019.

[24] N. Community, *Nxt whitepaper*, Accessed: Jan. 27, 2025. [Online]. Available: http://nxtforum.org/, 2020.

[25] T. Gayvoronskaya and C. Meinel, *Blockchain: Hype or Innovation*, 1st. Cham, Switzerland: Springer, 2020, p. 126.

[26] S. Kudva, R. Norderhaug, S. Badsha, S. Sengupta, and A. Kayes, "Pebers: Practical ethereum blockchain based efficient ride hailing service," in *2020 ieee international conference on informatics, iot, and enabling technologies (iciot)*, IEEE, 2020, pp. 422–428.

[27] M. R. Nosouhi, S. Yu, W. Zhou, M. Grobler, and H. Keshtiar, "Blockchain for secure location verification," *Journal of Parallel and Distributed Computing*, vol. 136, pp. 40–51, 2020.

[28] D. Wang and X. Zhang, "Secure ride-sharing services based on a consortium blockchain," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2976–2991, 2020.

[29] M. T. Zia, M. A. Khan, and H. El-Sayed, "Application of differential privacy approach in healthcare data – a case study," in *2020 14th International Conference on Innovations in Information Technology (IIT)*, 2020, pp. 35–39.

[30] L. Chen, P. Thakuriah, and K. Ampountolas, "Short-term prediction of demand for ride-hailing services: A deep learning approach," *Journal of Big Data Analytics in Transportation*, vol. 3, pp. 175–195, 2021.

[31] N. Fotiou, I. Pittaras, V. A. Siris, G. C. Polyzos, and P. Anton, "A privacy-preserving statistics marketplace using local differential privacy and blockchain: An application to smart-grid measurements sharing," *Blockchain: Research and Applications*, vol. 2, no. 1, p. 100 022, 2021.

[32] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 2901–2925, 2021.

[33] Z. Sun, Y. Wang, Z. Cai, T. Liu, X. Tong, and N. Jiang, "A two-stage privacy protection mechanism based on blockchain in mobile crowdsourcing," *International Journal of Intelligent Systems*, vol. 36, no. 5, pp. 2058–2080, 2021.

[34] S. Zou, J. Xi, G. Xu, M. Zhang, and Y. Lu, "Crowdhb: A decentralized location privacy-preserving crowdsensing system based on a hybrid blockchain network," *IEEE internet of things journal*, vol. 9, no. 16, pp. 14 803–14 817, 2021.

[35] H. Guo and X. Yu, "A survey on blockchain technology and its security," *Blockchain: Research and Applications*, vol. 3, no. 2, p. 100 067, 2022, ISSN: 2096-7209.

[36] P. Guo, B. Ye, Y. Chen, *et al.*, "A differential privacy protection protocol based on location entropy," *Tsinghua Science and Technology*, vol. 28, no. 3, pp. 452–463, 2022.

[37] M. Krichen, M. Ammi, A. Mihoub, and M. Almutiq, "Blockchain for modern applications: A survey," *Sensors*, vol. 22, no. 14, p. 5274, 2022.

[38] N. Mahmoud, A. Aly, and H. Abdelkader, "Enhancing blockchain-based ride-sharing services using ipfs," *Intelligent Systems with Applications*, vol. 16, p. 200 135, 2022.

[39] S. Namasudra and P. Sharma, "Achieving a decentralized and secure cab sharing system using blockchain technology," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 15 568–15 577, 2022.

[40] P. Philippaerts, D. Preuveneers, and W. Joosen, "Oauch: Exploring security compliance in the oauth 2.0 ecosystem," in *RAID '22: Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, Published: 26 October 2022, ACM, 2022, pp. 460–481. DOI: 10.1145/3545948.3545955.

[41] P. Guo, B. Ye, Y. Chen, *et al.*, "A differential privacy protection protocol based on location entropy," *Tsinghua Science and Technology*, vol. 28, no. 3, pp. 452–463, 2023.

[42] Uber, *Ride-hailing services data*, Accessed: Oct. 14, 2025. [Online]. Available: https://rb.gy/ogn7hy, 2023.

[43] G. Heo and I. Doh, "Blockchain and differential privacy-based data processing system for data security and privacy in urban computing," *Computer Communications*, vol. 222, pp. 161–176, 2024.

[44] M. O'Riordan, *The canonical websocket reference*, Accessed: Oct. 14, 2025. [Online]. Available: https://websocket.org/, 2024.

[45] Ö. Öksüz, "A smart contract based secure ride sharing system," *International Journal of Information Security Science*, vol. 13, no. 1, pp. 1–22, 2024.

[46] A. S. Foundation, *Apache kafka*, Accessed: Oct. 7, 2025. [Online]. Available: https://kafka.apache.org/, 2025.

[47] E. Hosseini, S. Chen, and A. Khisti, "Secure aggregation in federated learning using multiparty homomorphic encryption," *arXiv preprint arXiv:2503.00581*, 2025.

[48] A. Robles-González, J. Parra-Arnau, and J. Forné, "A linddun-based framework for privacy threat analysis on identification and authentication processes," *Computers & Security*, vol. 92, p. 101 759, 2025.

[49] G. Cloud, *Best practices for securely using api keys*, Accessed: Oct. 14, 2025. [Online]. Available: https://cloud.google.com/docs/authentication/api-keys.

[50] Etherscan, *Ethereum gas tracker*, Accessed: Oct. 14, 2025. [Online]. Available: https://etherscan.io/gastracker.

[51] T. A. S. Foundation, *Apache jmeter™*, Accessed: Oct. 6, 2025. [Online]. Available: https://jmeter.apache.org/.

[52] I. MongoDB, *Mongodb free version limits*, Accessed: Oct. 14, 2025. [Online]. Available: https://www.mongodb.com/docs/manual/reference/limits/.

[53] A. Parecki, *Oauth: Open authorization*, Accessed: Oct. 14, 2025. [Online]. Available: https://oauth.net/.