DESIGN AND DEVELOPMENT OF A PRIVACY-PRESERVING SEMI-PUBLIC BLOCKCHAIN-BASED RIDE-SHARING SYSTEM USING RAFT CONSENSUS WITH IPFS-ENABLED SECURE DISTRIBUTED STORAGE

Rayhan Ferdous Srejon, Mostafizur Rahman Fahim, Sk. Md. Shadman Ifaz, Md. Khairul Hasan, Rafiul Awal Nafi and Nabila Rahman

Department of CSE, Ahsanullah University of Science and Technology (AUST), Dhaka-1208, Bangladesh

ABSTRACT

Ride-sharing platforms have revolutionized urban mobility, offering millions of users convenient and costeffective transportation. However, mainstream centralized platforms such as Uber and Lyft continue to face pressing concerns including data privacy breaches, high service charges, security vulnerabilities, and a lack of transparency due to centralized control. To address these limitations, this research proposes a semipublic blockchain-based ride-sharing platform integrating Hyperledger Fabric for secure and permissioned data management with Ethereum smart contracts for transparent ride booking, fare calculation, and payments. The platform leverages the InterPlanetary File System (IPFS) for immutable, decentralized storage and uses the Cosmos SDK to enable seamless interoperability between public and private blockchains. A user-centric pay-as-you-drive model is introduced to ensure fair and distance-based billing. Preliminary evaluations show that our system outperforms traditional blockchain consensus methods (PoW, PoA) in throughput, latency, and resource usage. At the same time, it remains economically viable with an operational cost of under 33,000 BDT per node. Future improvements include benchmarking with Hyperledger Caliper, transitioning from Vagrant to Docker for better scalability, and implementing backend services using Node.js or Golang with MongoDB for efficient metadata handling. Together, these enhancements support a secure, decentralized, and scalable alternative to existing ride-sharing systems.

KEYWORDS

Blockchain, Hyperledger Fabric, IPFS, Cosmos SDK, RAFT, PoA, PoS, PoW, Ride Sharing.

1. INTRODUCTION

Ride-sharing has fundamentally changed urban transportation, providing users with a more flexible and affordable alternative to traditional taxi services. Companies such as Uber, Lyft, and Pathao have demonstrated the value of real-time ride coordination through centralized platforms.

However, while these systems are effective in terms of usability and scalability, they come with significant drawbacks, particularly around data privacy, high operational costs, and centralized control [1, 2]. As more personal and transactional data is collected and stored on centralized servers, users are increasingly vulnerable to data breaches and surveillance [2, 3]. In addition,

drivers and riders remain at the mercy of platform policies and pricing structures, without true control over their interactions [4]. In this context, decentralization using blockchain technologies presents a promising path forward [5].

2. PROBLEM STATEMENT

Existing ride-sharing platforms operate on a centralized architecture, where sensitive user information such as identity, location, and financial details is managed and stored by a single entity. This creates a critical point of failure, making the system vulnerable to cyberattacks, data breaches, manipulation, and unregulated third-party access [2, 3, 6]. Furthermore, these platforms impose high service fees and offer limited transparency in ride price, decision making, and payment distribution [4, 1]. The lack of user control and the opacity of platform operations reduce trust and exclude emerging regions from building their local ecosystems [5].

3. MOTIVATION

The limitations of centralized ride-sharing platforms have become increasingly evident in recent years, particularly as concerns over data misuse, transparency, and platform control continue to grow. Users entrust these platforms with highly sensitive information, including their location history, payment credentials, and behavioral data—often with little insight into how this information is used or shared [2]. Moreover, platform operators act as sole authorities over pricing, commissions, dispute resolution, and access policies, creating a power imbalance that undermines trust and user autonomy [4].

At the same time, the rapid evolution of decentralized technologies provides an opportunity to reimagine the ride-sharing ecosystem. Blockchain technologies, in particular, offer built-in mechanisms for transparency, immutability, and trustless interaction—qualities that directly address the shortcomings of current centralized platforms. Public blockchains like Ethereum allow smart contracts to automate and verify transactions in a transparent manner, reducing the need for third-party arbitration [7]. However, public chains often suffer from limitations such as high gas fees, limited throughput, and privacy concerns.

To address this, permissioned blockchains like Hyperledger Fabric present a compelling alternative. With support for identity management and access control, Fabric enables selective data disclosure and ensures compliance with privacy-sensitive use cases [3] [8] [9] . In our system, sensitive user data such as identity and ride history is stored securely on a Hyperledger Fabric chain, while ride coordination logic is deployed through Ethereum smart contracts to maintain transaction transparency. This hybrid architecture aims to strike a balance between confidentiality and openness.

Additionally, off-chain data storage remains a major challenge in blockchain-based applications. To mitigate this, our platform integrates the InterPlanetary File System (IPFS), a decentralized storage network that uses content-addressable hashes (CIDs) to retrieve immutable files. This allows large, non-sensitive data such as GPS traces and vehicle logs to be efficiently stored offchain while still maintaining verifiability [10]. To further enhance the system's flexibility and extensibility, the Cosmos SDK is used to bridge the Fabric and Ethereum blockchains, enabling secure and seamless inter-chain communication [5].

Collectively, these technologies make it possible to design a ride-sharing platform that is not only secure and privacy-preserving but also scalable and user-friendly. By eliminating centralized intermediaries, we reduce operational overhead and give users greater control over

their data and interactions. Our motivation is to harness the strengths of decentralized architectures particularly blockchain, IPFS, and Cosmos to create a next-generation ride-sharing platform that empowers both riders and developers, with the potential for broad adoption in regions underserved by current models.

4. OBJECTIVES

This research aims to design and implement a decentralized ride-sharing platform to overcome centralized systems' issues—data insecurity, opacity, monopolistic control, and rigid pricing. The platform will be scalable, privacy-focused, and user-centric, using a hybrid blockchain that blends public and permissioned networks for optimal benefits.

The specific objectives are:

- Ensure data privacy, integrity, and user ownership: The platform uses Hyperledger Fabric for a permissioned blockchain with role-based access and certificate-based identities, protecting personal data, ride history, and payments from unauthorized access or tampering. Following self-sovereign identity principles, users retain full control of their data.
- Enable transparent, auditable, and tamper-proof transactions: Ethereum smart contracts will automate ride creation, fare calculation, and payments, ensuring immutability, removing central authority reliance, and reinforcing trust and accountability.
- Utilize decentralized file storage for scalable metadata handling: The platform uses IPFS to store large, non-sensitive files (e.g., driver logs, GPS data) off-chain, with CIDs anchored on-chain, balancing scalability, security, and verifiability.
- Achieve cross-chain interoperability for seamless integration: To bridge blockchain fragmentation, the platform uses Cosmos SDK for secure, modular communication between Ethereum and Hyperledger Fabric, enabling synchronized cross-chain operations and future integrations.
- Introduce a real-time, usage-based pricing model: The platform uses a dynamic 'pay-asyou-drive' model, calculating fares from real-time trip duration, distance, and location data to ensure transparent, fair pricing and discourage manipulation.
- Evaluate technical performance and economic feasibility: The system will be benchmarked with Hyperledger Caliper for throughput, latency, and resource use, alongside a cost analysis of infrastructure, bandwidth, energy, and scalability to ensure technical and economic viability.
- Design for modular extensibility and upgradeability: The modular architecture supports adding new blockchain services, APIs, or consensus mechanisms without major redesign, ensuring long-term adaptability.
- Promote decentralized governance and control: The system reduces central authority reliance by enabling trustless interactions via smart contracts, fostering fair governance and enhancing credibility.

By achieving these objectives, the proposed platform aims to serve as a robust, secure, and fair alternative to current ride-sharing solutions, benefiting both urban commuters and underrepresented regions lacking infrastructure for centralized services.

5. LITERATURE REVIEW

Several blockchain-based approaches have been explored to decentralize and enhance ride-sharing systems, with varying focuses on transparency, privacy, trust, scalability, and interoperability. This chapter reviews key works in the field, synthesizing their methodologies, findings, and limitations to position our research in the existing academic landscape.

Naik et al. [7] proposed a decentralized ride-sharing platform using Ethereum smart contracts to eliminate intermediaries and promote trustless ride-matching and payment. Their solution automated core operations such as ride booking and fare handling via a decentralized application (DApp). While the system demonstrated improved transparency, it suffered from high gas fees, limited scalability, and the absence of user data privacy mechanisms. Enhancements such as incorporating off-chain storage and Layer-2 scaling solutions were suggested to address these concerns.

Baza et al. [11] introduced B-Ride, a privacy-preserving ride-sharing framework built atop the Ethereum blockchain. Their system leveraged zero-knowledge proofs (ZKPs) and time-locked deposit protocols to ensure secure, fair, and anonymous transactions. The design effectively reduced the need for user trust and provided strong privacy guarantees. However, the approach was hampered by Ethereum's performance limitations and the complexity of implementing ZKPs, which restricted usability in real-world environments. The authors recommended using scalable chains and alternative deposit models to improve practicality.

Shivers et al. [12] explored the use of Hyperledger Fabric in building a decentralized ride-hailing platform for autonomous vehicles. Their system employed smart contracts to manage vehicle coordination, user authentication, and secure data logging within a permissioned blockchain environment. The platform showed robust performance under synthetic loads but lacked integration with public chains for transparency and failed to address off-chain data storage. Future work was proposed to incorporate decentralized file systems and real-world deployment testing.

Mahmoud et al. [10] proposed a hybrid ride-sharing platform that integrates Ethereum with the InterPlanetary File System (IPFS) to improve scalability and data efficiency. In their architecture, non-sensitive ride metadata was uploaded to IPFS, and only content hashes were stored on the blockchain to reduce on-chain bloat. While this design effectively minimized storage overhead, it faced issues such as data retrieval latency, reliance on public IPFS nodes, and unencrypted data exposure. The authors highlighted the need for encryption, private IPFS clusters, and improved integration with backend systems.

Namasudra and Sharma [13] presented a decentralized cab-sharing system using CiphertextPolicy Attribute-Based Encryption (CP-ABE) and Delegated Proof of Stake (DPoS) for secure access control and consensus. Their solution provided fine-grained control over data sharing and reduced dependency on centralized platforms. Despite its strong privacy model, the system lacked support for dynamic ride-matching and integration with decentralized storage or public smart contracts. Improvements were suggested in the form of hybrid models with smart contracts and IPFS integration for broader utility.

Wang and Zhang [14] proposed a consortium blockchain-based ride-sharing framework that emphasized secure ride-matching through attribute-based encryption and proxy re-encryption. Their implementation used Delegated Proof of Stake (DPoS) for efficient consensus and aimed to protect sensitive user data within a controlled network. The design offered high data confidentiality but struggled with infrastructure costs, a lack of public transparency, and limited

data interoperability. Proposed future directions included cross-chain protocols like Cosmos SDK and off-chain storage for metadata.

Chang et al. [4] developed a smart contract-driven ride-sharing platform using Ethereum that incorporated automated fare calculation, user authentication, and trust scoring. The system allowed decentralized execution of core platform functions and reduced operational overhead. However, it lacked privacy features and showed poor performance at scale due to Ethereum's congestion and limited throughput. The authors acknowledged the potential for privacy-preserving techniques and recommended shifting to Layer-2 solutions or combining with permissioned blockchains for better performance.

Tariq et al. [15] proposed a fully decentralized peer-to-peer ride-sharing platform built on Ethereum. The system leverages smart contracts for ride creation, user verification, matching, and payments, eliminating the need for intermediaries. A token-based incentive structure promotes user engagement, and the platform was deployed on Ethereum testnets to analyze gas consumption and transaction latency. Despite achieving transparency and decentralization, the platform suffers from high gas costs, lacks privacy mechanisms such as zero-knowledge proofs (ZKPs), and omits decentralized storage integration like IPFS.

Koubaa et al. [16] presented a comprehensive survey of blockchain-based ride-sharing systems, categorizing them based on architecture types, consensus mechanisms (e.g., PoW, PoS, PBFT), and privacy models. The survey identifies key issues such as scalability bottlenecks, identity management, secure data sharing, and throughput limitations. It also discusses potential future integrations with IoT for real-time mobility. However, the work is entirely theoretical, lacking any implementation, benchmarking, or discussion of decentralized storage or blockchain interoperability solutions such as Cosmos SDK.

Zhang and Wen [17] introduced a hybrid IoT blockchain framework that combines vehicle telemetry data with Ethereum-based smart contracts to enable dynamic ride-matching and fraud prevention. The system uses sensor data (e.g., location, speed) for route verification and incorporates a reputation model to deter malicious actors. While it supports real-time transparency and trust, the framework depends on stable IoT infrastructure, incurs high operational costs due to gas fees, and lacks privacy-preserving techniques or decentralized data offloading.

These studies reflect growing interest in decentralized mobility solutions but reveal recurring limitations, most notably, the lack of scalable privacy-preserving systems, real-time data handling, and interoperability between hybrid blockchains. Our proposed system addresses these gaps by combining Hyperledger Fabric for permissioned data storage, Ethereum for transparent contract execution, IPFS for decentralized off-chain storage, and the Cosmos SDK for secure inter-chain communication. A novel pay-as-you-drive mechanism further improves fairness and usability over time-locked deposit models seen in earlier works.

6. METHODOLOGY

This chapter outlines the methodology used to develop a decentralized ride-sharing system built on a hybrid blockchain architecture. The system leverages Hyperledger Fabric, the Ethereum public blockchain, the RAFT consensus algorithm, MetaMask wallet integration, the Cosmos SDK for blockchain interoperability, and IPFS for decentralized storage. The approach is design-based, focusing on building a practical and scalable framework that addresses key limitations in existing blockchain-based ride-sharing solutions. This methodology aligns with the research objectives by integrating a secure hybrid blockchain architecture and decentralized

storage mechanisms. It provides a real-world solution to challenges such as lack of transparency, limited traceability, and compromised data integrity. The following subsections detail the system architecture, the technologies employed, and the operational workflow of the proposed solution.

6.1. System Architecture and Design

Figure 1 illustrates the blockchain-enabled architecture of the proposed ride-sharing application, which consists of two main modules: user management and ride-sharing. The system securely registers vehicles, drivers, and passengers through smart contracts, ensuring verified identities and trusted participation. The ride workflow is organized into five clear steps: the driver posts ride information, passengers book the ride, drivers confirm the booking, passengers pay at the end of the trip, and drivers close the ride session, preventing further requests. This structured process promotes transparency, efficiency, and accountability. At the core, a Hyperledger-based private blockchain stores sensitive user data, including personal details and ride histories, under strict permissioned access. To reduce blockchain storage load and improve redundancy, IPFS is integrated to store vehicle-related data such as descriptions and maintenance records, benefiting from its decentralized and censorship-resistant nature. An Ethereum public blockchain powers smart contracts for ride matching, fare calculation, and payment settlements, ensuring automation, transparency, and immutability. To bridge the two blockchain layers, the system employs Cosmos, enabling secure and seamless interoperability. This hybrid approach leverages the privacy and control of private blockchains with the openness and trust guarantees of public blockchains, creating a scalable, secure, and user-centric ride-sharing platform.

i. Decentralized and Immutable Information Storage using IPFS

The system uses a hybrid architecture for secure, scalable storage. Sensitive data—personal details, vehicle info, ride history, and payments—is stored on Hyperledger Fabric's permissioned blockchain, while large non-sensitive data (e.g., GPS logs, ride metadata) is stored on IPFS. During registration, sensitive data is encrypted and stored as tamper-proof blocks on Hyperledger, and bulk data is uploaded to IPFS, generating a unique Content Identifier (CID) anchored on Hyperledger. For retrieval, authorized users decrypt on-chain data or use CIDs to fetch files from IPFS, verifying integrity by matching hashes. This dual-layer design reduces blockchain overhead, ensures immutability, and balances privacy with decentralized scalability for efficient, secure ridesharing.

ii. User Information Verification and Upload

The first step of the system focuses on verifying the authenticity of user information and securely uploading any related files. This verification is crucial, especially in case of drivers, as it ensures

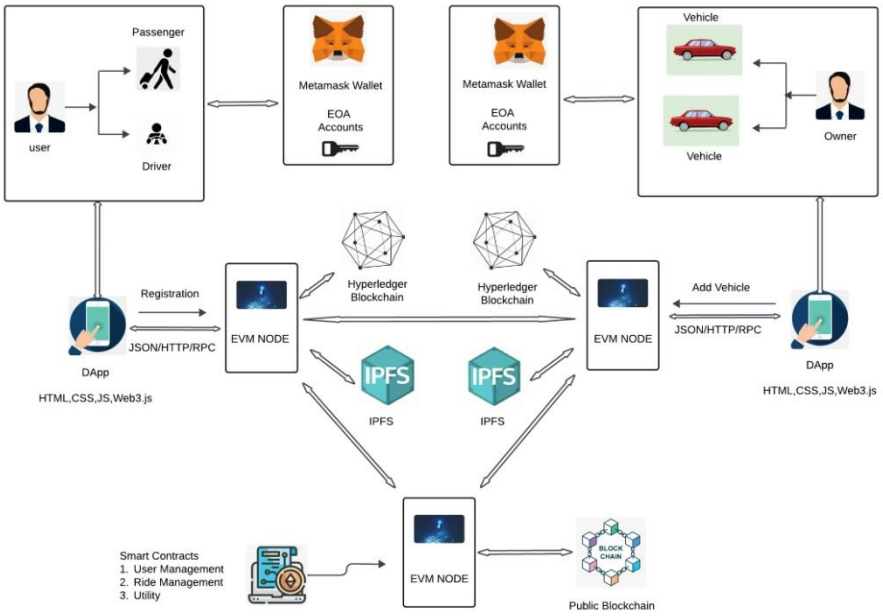

Figure 1: System Architecture

that the information being registered is genuine and trustworthy. Once users log in and register, they begin by submitting all required documents linked to their vehicles, such as driving licenses, national ID card, or vehicle papers.

These submissions are carefully validated through an advanced authentication process that checks multiple aspects of authenticity. First, the system performs a basic format check to make sure everything is complete and correctly structured. Then, it uses cryptographic hashing to create a unique digital fingerprint of the submitted user data. This fingerprint is cross-checked against existing entries on the blockchain to detect duplicates or any signs of unauthorized use.

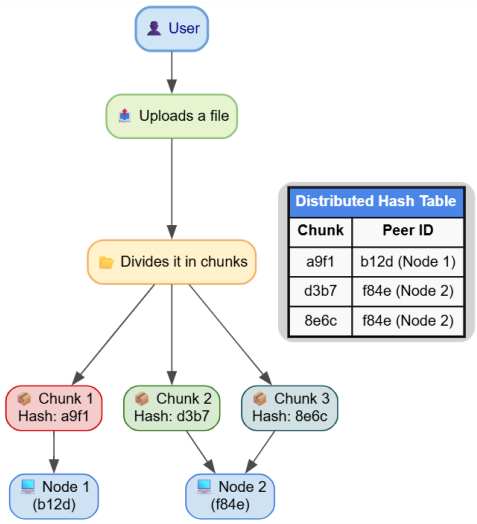

Figure 2: Working Mechanism of IPFS for storing and retrieving user files

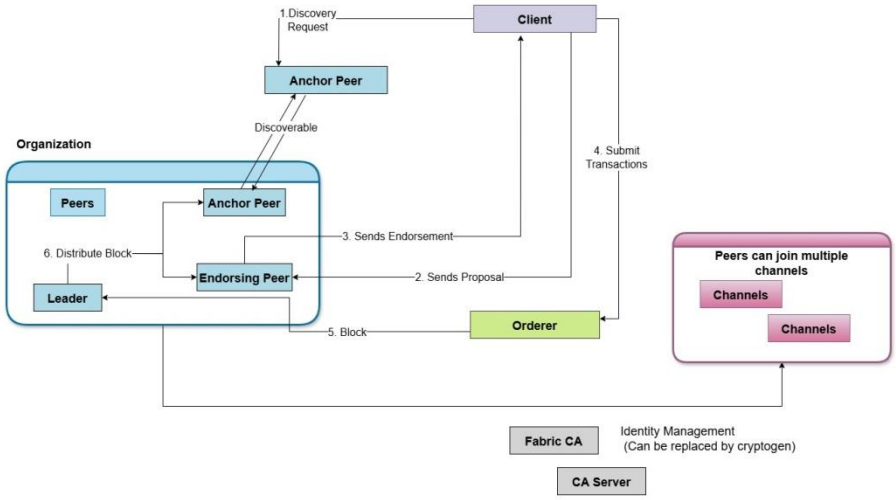

Figure 3: Internal Working Mechanism of Hyperledger Fabric

iii. Uploading User Files to IPFS

Once the user details have been verified and approved, any associated media files—such as images, videos, or documents—go through a preparation phase to ensure they're stored securely and efficiently. These files are first broken down into smaller, manageable chunks using cryptographic hashing. This process generates a unique, fixed-size hash for each segment, helping maintain data integrity and making it easier to track and retrieve the files.

The segmented media is then uploaded to IPFS (InterPlanetary File System), a decentralized storage network that distributes data across a peer-to-peer system. Unlike traditional centralized storage methods, IPFS improves reliability by storing copies of the data on multiple nodes, minimizing the risk of data loss or corruption. This decentralized structure also makes it more secure and scalable—especially useful when handling large media files tied to intellectual property.

After uploading, IPFS assigns each file a unique Content Identifier (CID). This CID acts as a digital fingerprint of the file, permanently linked to its exact content and version. Even a small change in the file will generate a completely different CID, ensuring that the data remains tamperproof. The blockchain then uses this CID to reference the precise version of the media stored in IPFS. Figure ??illustrates how IPFS operates behind the scenes.

iv. Storing the CID

After IPFS generates the CID, it is recorded on Hyperledger Fabric for traceability and integrity, with a client transaction proposal triggering the relevant smart contract.

The transaction is first reviewed by a specific group of endorsing peers, as defined by the network's endorsement policy. Each of these peers simulates the transaction using its current ledger state without actually committing any changes and returns a digitally signed read-write set as its endorsement. Once the client collects enough valid endorsements, it packages the transaction and forwards it to the ordering service. The ordering service, powered by the RAFT consensus algorithm, arranges all submitted transactions in a consistent global order. These transactions are bundled into blocks, which are sent to the leader peer in each organization. The

leader then distributes the blocks to other peers in the network. Every peer independently validates the block

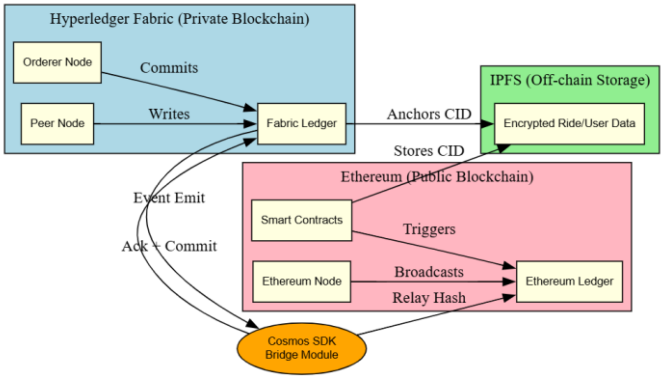

Figure 4: Interoperability architecture for synchronizing transactions across Hyperledger and Ethereum using Cosmos SDK

by verifying the endorsement signatures and ensuring it complies with the set policies. If everything checks out, the block is committed to the ledger, making the CID a permanent part of the blockchain's immutable record. Since the CID is a cryptographic fingerprint of the media stored on IPFS, it serves as a reliable, tamper-proof reference to the original content. Figure 3 illustrates the inner workings of this Hyperledger Fabric–based private blockchain system.

v. Interoperability between Public and Private Blockchains

As shown in Figure 4, the proposed architecture enables secure interoperability between Hyperledger Fabric and Ethereum using the Cosmos SDK Bridge. Fabric handles encrypted ride/user data, stored off-chain in IPFS with CIDs anchored on-chain. Key Fabric events are relayed to Ethereum, where event hashes are recorded for verifiable proof without exposing sensitive data. Users access the system via DApps using MetaMask and Web3.js. This hybrid design combines Fabric's privacy, Ethereum's transparency, and IPFS's efficiency. However, real-world use may face issues like clock skew and differing finality, requiring custom retry and validation logic.

vi. Smart Contracts Workflow

The proposed system consists of two main contracts: the *UserManagementContract* for user management and the *RideSharingContract* to handle ride-sharing functions. Additionally, a utility contract contains utility functions, such as converting poisha to wei, to assist with various calculations within the ride-sharing DApp. Figure 5 illustrates the detailed workflow of the smart contract within the ride-sharing DApp.

UserManagementContract is responsible for managing operations related to the user within the ride-sharing DApp. It handles user registration, authentication, and profile management. Users can register by providing their necessary information, including their Ethereum wallet address and personal details. The contract verifies the user's identity and stores the relevant information securely on the blockchain. It also allows users to update their profile information, view their ride history, and manage their preferences. The *RideSharingContract* serves as the core contract for facilitating ride-sharing functionalities. It handles operations related to creating rides, joining rides, and managing ride details. Users can create rides by specifying parameters such as the starting location, destination, time, and available seats. The contract validates the

input data, creates a new ride instance, and adds it to the list of active rides. It also maintains a participant

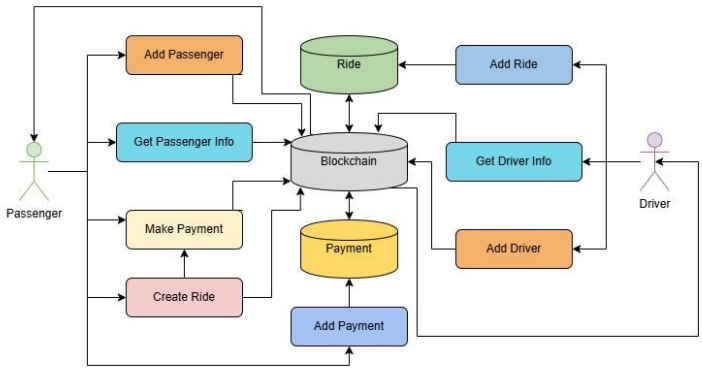

Figure 5: Smart Contract Workflow

list for each ride, keeping track of the users who have joined. Users can join existing rides by interacting with the *RideSharingContract*. The contract verifies seat availability, adds the user to the participant list, and updates the ride details accordingly.

It also allows users to view ride details, including the creator, participants, starting location, destination, and time, providing transparency and facilitating communication between ride participants. The Utility Contract includes utility functions that support various calculations and conversions required within the ride-sharing DApp. For example, it may provide functions to convert poysha (a subdivision of the Bangladeshi Taka) to wei (the smallest denomination of ether in the Ethereum network) to handle payment calculations. These utility functions streamline the process of converting between different currency units, ensuring accurate and consistent calculations throughout the application. By utilizing these main contracts, the proposed system establishes a decentralized ride-sharing platform that allows users to manage their profiles, create and join rides, and perform necessary calculations for payments. The contracts leverage the security and transparency of the Ethereum blockchain, providing a reliable and trustless environment for ride-sharing operations. In failure scenarios like payment drop or sudden ride cancellation, the system queues those states and retries contract validation through a hashed log checkpoint system, ensuring consensus integrity is maintained.

6.2. Network Setup and Configuration

Before setting up the full Hyperledger Fabric network, several key tools and environments were prepared to support the early phases of development and deployment. These prerequisites help ensure that all parts of the system—from chaincode development to container management—can be properly configured and tested.

At this stage, the chaincode includes only an example chaincode and currently uses a static IPFS hash, as the integration with IPFS and Ethereum Blockchain is still pending. Although the complete system is still in progress, the following section outlines the work completed so far.

6.2.1. Prerequisites

The following tools and dependencies were installed:

- Visual Studio Code (VS Code): Used for writing chaincode, network configurations, and scripts. Its integrated terminal and Fabric extensions facilitate development.
- Node.js and npm: Essential for running the Fabric SDK client application and managing project dependencies. Also required to run the React development server, manage dependencies, and build the frontend application.
- Golang (Go): Required for writing and compiling chaincode. Go is the primary language for Hyperledger Fabric smart contracts.
- Git: Used for version control and cloning Fabric samples and network configuration scripts from official repositories.
- Hyperledger Fabric Binaries: Tools like cryptogen, configtxgen, and peer CLI were downloaded and used for generating cryptographic material, creating the genesis block, and managing channels.
- Vagrant: Vagrant is a tool for managing virtual environments. The acloudfan/hlfdev2.2.0 box provides a ready-to-use setup for Hyperledger Fabric v2.2 which includes Fabric tools. It helps developers to launch and test Fabric networks.

With the prerequisites in place, the network setup process involved generating crypto material, configuring channels, deploying smart contracts (chaincode), and launching the Fabric network.

6.2.2. System Configuration Tasks

The following setup tasks were performed to initialize and configure the Hyperledger Fabric network and IPFS integration:

- Crypto Material Generation: Generated certificates and keys using the cryptogen tool based on the crypto-config.yaml file.

```
cryptogengenerate--config=./crypto-config.yaml
```

- Channel Configuration Generation: Created the genesis block and channel transaction configuration using configtxgen.

```
configtxgen-profileAcmeOrdererGenesis\
   –channelIDordererchannel\
   –outputBlock./acme-genesis.block
configtxgen-profileAcmeChannel\
   –channelIDacmechannel\
   –outputCreateChannelTx./acme-channel.tx
```

- Creating the Channel: Created the application channel using the create command.

```
peer channel create -o $ORDERER_ADDRESS \
   -c acmechannel -f $CONFIG_DIRECTORY /acme-channel.tx
```

- Joining the Channel: Peer nodes joined the created channel by referencing the channel block.

```
peerchannelioin-o$ORDERER ADDRESS\
-b./acmechannel.block
```

where the ORDERER ADDRESS=”localhost:7050”

- Chaincode Packaging: Packaged the chaincode (smart contract) into a compressed archive for deployment.

```
peerlifecvclechaincodepackage\
$CC PACKAGE FILE-pchaincode example02\
--label$CC LABEL
```

- Chaincode Installation: Installed the packaged chaincode on each peer node.

```
peerlifecvclechaincodeinstall$CC PACKAGE FILE
```

- Chaincode Approval: Approved the chaincode definition on behalf of each organization.

```
peerlifecvclechaincodeapproveformvorg\
-nipcc-v1.0-Cacmechannel\
--sequence1--package-id$CC PACKAGE ID
```

- Chaincode Commit: Committed the chaincode definition to the channel after collecting required approvals.

```
peerlifecvclechaincodecommit-ngocc\
-v1.0-Cacmechannel--sequence 1.
```

- Chaincode Invocation: Invoked the chaincode by using Invoke function.

```
peerchaincodeinvoke-Cacmechannel\
-ngocc-c’{"Args":["invoke","a","b","10"]}’
```

- IPFS Installation and Setup: Installed IPFS to enable decentralized file storage for IP content and metadata.

```
wget https://dist.ipfs.tech/go-ipfs/
\v0.18.1/go-ipfs_v0.18.1_linux-amd64.tar.gz
tar -xvzf go-ipfs_v0.18.1_linux-amd64.tar.gz
cd go-ipfs sudo bash install.sh ipfs init ipfs daemon
```

- Frontend Design: Some of the pages, such as the homepage, user registration, login, and admin dashboard, are designed using React.js.

Table 1: Cost analysis for a Hyperledger node setup

Cost Category	Description	Estimated Cost (per unit)
CPU	AMD Ryzen 7 5700X3D for efficiently running the node and smart contracts.	20,000 BDT per CPU
RAM	Least amount of memory required for running the node for smoothly processing transactions (16 GB).	5,500 BDT per 16GB module
Storage	SSD (Solid State Drive) for storing blockchain data, logs, and other metadata.	4,500 BDT per 512 GB SSD
IPFS Storage	Cost for decentralized storage using IPFS (if self-hosted).	5 BDT per GB/month
Stable Internet Connection	A minimum 40 Mbps broadband connection is required for low-latency node communication and faster response.	1,500 BDT per month per node
Power Consumption	Electricity costs for running Hyperledger nodes and supporting the required hardware to run the nodes.	2,500 BDT per month per node
	Total	34,000 BDT (approx.)

7. RESULT ANALYSIS

A cost analysis (Table 1) estimates the monthly expense of deploying a single node at 33,000 BDT. This includes an AMD Ryzen 7 5700X3D CPU (20,000 BDT) for smart contract execution, 16 GB RAM (5,500 BDT) for smooth transaction processing, and a 512 GB SSD (4,500 BDT). IPFS storage adds 5 BDT/GB/month, 40 Mbps broadband costs 1,500 BDT/month, and power consumption is 2,500 BDT/month. The configuration is both cost-effective and scalable, supporting secure and transparent ride-sharing.

7.1. Theoretical Result Analysis

This section provides a theoretical analysis of the performance metrics of a blockchain- based system designed to build a ride-sharing system using Hyperledger Fabric, IPFS, and Ethereum public blockchain. The consensus algorithms compared include RAFT, Proof of Stake (PoS), Proof of Work (PoW), and PoA (Proof of Authority).

7.1.1. CPU Usage Estimation

It shows the proportion of processor power used for processing transactions and reaching a consensus. Greater computational load is implied by higher values. The following generic equation is used to estimate the CPU utilization for each consensus algorithm[18].

$$\text{CPU\%} \approx \text{Base}_{\text{CPU}} + \delta \times \left(\frac{N}{1000} \right) \quad (1)$$

where:

- N is the number of transactions
- $Base_Cpu$ is the base CPU usage percentage
- δ is the CPU growth rate per 1,000 transactions

Table 2: CPU usage coefficients

Consensus	$Base_Cpu$	δ
PoW	60	0.5
RAFT	15	0.3
PoS	20	0.25
PoA	12	0.2

7.1.2. Latency Estimation

Latency indicates the time delay (in milliseconds) from submitting a transaction to its confirmation. Lower latency means faster transaction finality. The latency per transaction batch is estimated by [19]:

$$\text{Latency}(N) = L_0 + L_{\text{ipfs}} + L_{\text{net}} + \gamma \times \left(\frac{N}{1000}\right) \quad (2)$$

where:

- L_0 is the base latency due to consensus protocol
- $L_{\text{ipfs}} \approx 30$ ms is the average IPFS overhead
- $L_{\text{net}} \approx 40$ ms is the assumed network latency
- γ is the queuing/congestion growth factor

7.1.3. Transaction Throughput (TPS) Estimation

TPS measures how many transactions a blockchain system can process per second. Higher TPS indicates better throughput. TPS is derived from latency using [20]:

$$\text{TPS}(N) = \frac{1000 \times N}{L_0 + L_{\text{ipfs}} + L_{\text{net}} + \gamma \times \left(\frac{N}{1000}\right)} \quad (3)$$

This formula estimates the number of transactions per second for a given number of transactions N by inverting the total latency per batch.

The TPS, latency, and CPU utilization percentage for each consensus mechanism were determined using the algorithms mentioned above, and their interpretations are given in the part that follows.

Table 3: Latency coefficients

Consensus	L_0 (ms)	γ (ms/1k tx)
PoW	2500	100
RAFT	50	20
PoS	150	25
PoA	80	35

7.1.4. Graphical Analysis and Interpretation

To assess the performance of the suggested blockchain-based ride-sharing system, we examined how four distinct consensus mechanisms—RAFT, PoS, PoW, and PoA—performed over 10,000 transactions in 1,000-transaction increments. Three major performance measures were used to display the results: CPU utilization, latency, and Transactions Per Second (TPS).

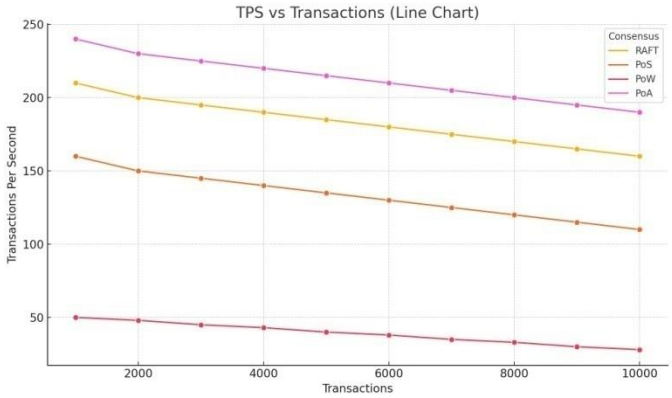

Figure 6: Transactions Per Second (TPS) vs Number of Transactions

1. TPS vs Transactions (Figure 6): From the TPS graph, PoA exhibits the highest throughput across all transaction loads, consistently maintaining over 190 TPS even at 10,000 transactions. This highlights the efficiency of PoA's lightweight, authority-based consensus, which minimizes communication overhead and block validation time. RAFT closely follows, starting above 210 TPS and gradually decreasing to 160 TPS as transaction volume increases. This steady performance reflects RAFT's optimized ordering service in Hyperledger Fabric, though its need for endorsement and block packaging slightly limits scalability under load.

   PoS performs moderately well, starting around 160 TPS but steadily declining to 110 TPS by 10,000 transactions. This drop may be attributed to validator communication overhead and the cost of maintaining network consensus as transaction volume increases. PoW, on the other hand, delivers the lowest throughput, starting near 50 TPS and falling to just 28 TPS. This aligns with its high computational cost and block propagation delay, making PoW unsuitable for real-time, high-throughput systems like decentralized ride-sharing platforms.

2. Latency vs Transactions (Figure 7): Latency increases progressively across all consensus mechanisms as the transaction count grows from 1,000 to 10,000. Among them, PoW exhibits the

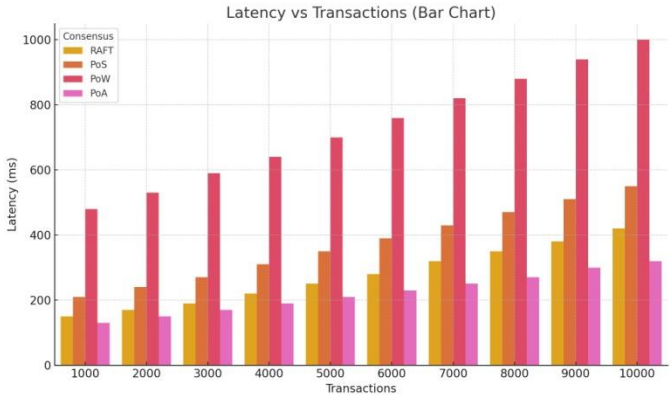

Figure 7: Latency (ms) vs Number of Transactions

highest latency, reaching approximately 1000 ms at peak load, primarily due to the computational burden of mining and slow block propagation. PoS shows moderate latency growth, rising to around 550 ms by the 10,000th transaction—this is attributable to validator rotation, slot scheduling, and network consensus delay.

In contrast, RAFT maintains comparatively low latency, starting at 150 ms and gradually increasing to 420 ms. This reflects the benefits of deterministic finality and efficient leader-based block ordering in Hyperledger Fabric. PoA outperforms all others in latency, staying under 320 ms throughout the test, owing to its simplified block validation model and minimal consensus overhead. These results suggest that for latency-sensitive applications such as real-time ride-matching, PoA and RAFT are more suitable, while PoW may not be ideal due to its high delay.

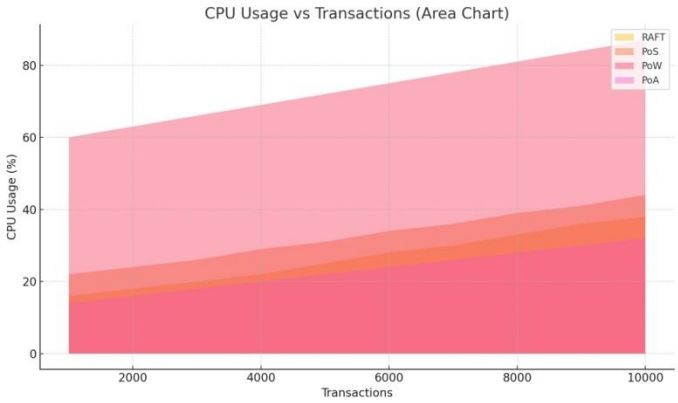

Figure 8: CPU Usage (%) vs Number of Transactions

3. CPU Usage vs Transactions (Figure 8): CPU usage illustrates the relative computational demands of each consensus mechanism under increasing transaction loads. PoW exhibits the steepest curve, beginning at 60% and escalating to nearly 90% at 10,000 transactions, reflecting its intensive mining operations and constant hash computation. This makes PoW highly inefficient and resource-consuming for scalable blockchain applications on standard hardware.

   In contrast, RAFT and PoA maintain low to moderate CPU usage, staying well below 40% even under peak load. RAFT benefits from a streamlined leader-based approach, while PoA

gains efficiency by eliminating competitive consensus entirely. PoS falls between the two extremes, reaching about 44% at maximum load, primarily due to validator messaging and signature verification overhead. These results confirm that RAFT and PoA are the most efficient choices for CPU-constrained environments, making them ideal for real-time, decentralized ride-sharing systems or IP marketplaces.

4. Comparison between RAFT and PoA: Although Proof of Authority (PoA) may outperform RAFT in terms of raw throughput, latency, and CPU usage due to its lightweight and centralized nature, RAFT offers greater reliability, security, and suitability for enterprise applications. As a consensus mechanism used in permissioned networks like Hyperledger Fabric, RAFT ensures deterministic finality, multi-party endorsement, and support for private data handling—features that PoA lacks. While PoA is ideal for scenarios prioritizing speed over decentralization, RAFT is better suited for systems requiring robust governance, privacy, and auditability, making it a more practical choice for secure, multi-organizational ride-sharing platform.

From the graphs and performance analysis, we can conclude that:

- RAFT offers a strong balance of high TPS, low latency, and moderate CPU usage, while also ensuring enterprise-grade reliability, privacy, and deterministic finality. It is best suited for secure, multi-organizational ride-sharing systems requiring auditability and governance.
- PoA demonstrates superior raw performance in terms of throughput, latency, and CPU efficiency due to its lightweight, centralized nature. However, it lacks the decentralization and data control features required for trust-sensitive environments.
- PoS maintains a reasonable compromise between performance and decentralization, with moderate CPU consumption and stable TPS. It may suit open ecosystems where validatorbased trust is acceptable.
- PoWperforms the worst in all metrics, showing high latency and CPU usage with the lowest TPS. Its resource-intensive nature makes it impractical for real-time, large-scale deployments.

These results validate the choice of RAFT as the preferred consensus mechanism for implementing a privacy-preserving, scalable, and efficient hybrid blockchain architecture in the proposed decentralized ride-sharing platform.

7.2. Practical Result Analysis

Using mathematical formulas, a thorough theoretical analysis was carried out in Section 1 to estimate important performance parameters including TPS, latency, and CPU utilization percentage across various consensus techniques (RAFT, PoS, PoW, and PoA). System-level variables including IPFS overhead, network latency, and transaction load were taken into account in these estimations. However, an open source blockchain performance benchmarking tool called Hyperledger Caliper will be used to benchmark the final implementation of the suggested system in order to verify these results under actual execution settings. Real TPS, end-to-end latency, resource use, and transaction success rate are just a few examples of the empirical performance metrics that Caliper will offer. This will guarantee that the system performs as expected under workloads encountered in the real world.

8. CONCLUSION AND FUTURE WORKS

This study proposes a hybrid blockchain ride-sharing architecture using Hyperledger Fabric, IPFS, and Ethereum to tackle data centralization, opacity, and limited user control. Leveraging smart contracts, off-chain storage, and cross-chain sync, it enables secure ride management, private data storage, and auditable fares.

The research demonstrates the platform's feasibility: Hyperledger Fabric with RAFT enables trusted, low-latency transactions; IPFS stores ride logs and documents; Ethereum provides public event verification and ride tokens; and a React.js frontend delivers real-time booking, role-based access, dynamic maps, and live tracking.

Consensus benchmarking found PoA fastest and most efficient, but RAFT delivered the best mix of performance, reliability, and security. PoW was resource-heavy and ill-suited for high-frequency use, supporting RAFT's selection for a secure, multi-organizational ride-sharing platform.

Though still in development, the project's hybrid use of Fabric, Ethereum, and IPFS provides a solid foundation for scalable, decentralized ride-sharing.

Looking ahead, the following future enhancements are planned:

- Smart Contract Completion: Extend chaincode logic to cover all stages of ride lifecycle, including booking, cancellation, fare disputes, and rating. Enable Ethereum-based token integration for payment verification and user incentives.
- Frontend and UX Improvements: Enhance the UI with driver verification, surge pricing display, payment integration, and admin controls, progressing toward a mobile-ready app.
- Backend Middleware Development: A Node.js or Golang backend will handle API calls, blockchain interaction, authentication, and metadata retrieval.
- Metadata Handling and Database Support: MongoDB will be used store off-chain user and transaction metadata.
- Migration to Docker-based Infrastructure: The current Vagrant setup will be replaced with Docker to simplify containerized deployment, CI/CD integration, and system scalability.
- System Benchmarking and Performance Testing: After complete integration, Hyperledger Caliper will benchmark latency, TPS, CPU usage, and fault tolerance to validate results.

In conclusion, this work contributes to the growing research in decentralized mobility systems by proposing a hybrid blockchain framework that is both technically viable and practically scalable. With continued development, the system holds the potential to revolutionize the ride-sharing ecosystem by empowering users with transparency, security, and control over their data and interactions.

REFERENCES

[1] Gasca, M.-V. and Rigo-Mariani, R. and Debusschere, V. and Sidqi, Y., (2025). Fairness in energy communities: Centralized and decentralized frameworks. Renewable and Sustainable Energy Reviews 208, 115054.

[2] Jamal, H. and Algeelani, N. A. and Al-Sammarraie, N., (2024). , 46–54.

[3] Chowdhury, S. J. and Aich, E. and Reno, S. and Ahmed, M., (2022). Utilizing hyperledger based private blockchain technology to secure credit card payment system.
[4] Chang, S. E. and Chang, E. C. and Chen, Y., (2022). , 13732.
[5] Theodorakopoulos, L. and Theodoropoulou, A. and Halkiopoulos, C., (2024). , 7007.
[6] Reno, S. and Sadi, I. and Karmakar, J. and Abir, M., (2021). Counterfeit medicine identification using hyperledger based private blockchain.
[7] Naik, M. and Singh, A. P. and Pradhan, N. R., (2024). Decentralizing ride-sharing: a blockchain-based application with smart contract automation and performance analysis. Multimedia Tools and Applications, 1–28.
[8] Yuan, Y. and Wang, F. Y., (2018). , 3133–3142.
[9] Kang, J. and Yu, R. and Huang, X. and Maharjan, S. and Zhang, Y., (2019). , 111–117.
[10] Mahmoud, N. and Aly, A. and Abdelkader, H., (2022). Enhancing blockchain-based ridesharing services using ipfs. Intelligent Systems with Applications 16, 200135.
[11] Baza, M. and Lasla, N. and Mahmoud, M. M. and Srivastava, G. and Abdallah, M., (2019). , 1214–1229.
[12] Shivers, R. and Rahman, M. A. and Faruk, M. J. H. and Shahriar, H. and Cuzzocrea, A. and Clincy, V., (2021). Ride-hailing for autonomous vehicles: Hyperledger fabric-based secure and decentralize blockchain platform.
[13] Namasudra, S. and Sharma, P., (2022). , 15568–15577.
[14] Wang, D. and Zhang, X., (2020). , 2976–2991.
[15] Tariq, M. and Ali, H. and Rehman, M., (2023). A decentralized ethereum-based ride-sharing platform: Architecture and performance analysis. Journal of Blockchain Applications, 1–17.
[16] Koubaa, A. and Qureshi, H. and Afzal, A., (2023). Blockchain-based ride-sharing systems: A comprehensive survey. IEEE Access 11, 56012–56035.
[17] Zhang, Y. and Wen, J., (2022). , 1456.
[18] Melo, C. and Oliveira, F. and Dantas, J. and Araujo, J. and Pereira, P. and Maciel, R. and Maciel, P., (2022). , 12505–12527.
[19] Piao, X. and Li, M. and Meng, F. and Song, H., (2022). Latency analysis for raft consensus on hyperledger fabric.
[20] Thakkar, P. and Nathan, S. and Viswanathan, B., (2018). Performance benchmarking and optimizing hyperledger fabric blockchain platform.

AUTHORS

Rayhan Ferdous Srejon is currently pursuing his Bachelor of Science (B.Sc.) degree in Computer Science and Engineering at Ahsanullah University of Science and Technology (AUST), Dhaka, Bangladesh. He is deeply passionate about innovative technologies and has developed a strong interest in the fields of Blockchain, Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL).

Mostafizur Rahman Fahim is completing his Bachelor of Science (B.Sc.) degree in Computer Science and Engineering at Ahsanullah University of Science and Technology (AUST), Dhaka, Bangladesh. He has a strong research interest in the fields of Blockchain, Machine Learning (ML), and Deep Learning (DL).

Sk.Md. Shadman Ifaz is a final year student in Computer Science and Engineering at Ahsanullah University of Science and Technology (AUST), Dhaka, Bangladesh. He has developed a strong interest in the fields of Blockchain, Machine Learning (ML), and Deep Learning (DL).

Md. Khairul Hasan is currently working as an Associate Professor in the department of Computer Science and Engineering at Ahsanullah University of Science and Technology. With over 24 years of teaching experience, he has published 13 research paper. He received the B. Sc. degree in Computer Science and Engineering from Ahsanullah University of Science and Technology, Dhaka, Bangladesh and the M.Sc. degree in Computer Science and Engineering from United International University, Dhaka, Bangladesh. His research interests include computational intelligence, neural networks, blockchain, optimization, and their applications.

Raful Awal Nafi has successfully completed his Bachelor of Science (B.Sc.) degree in Computer Science and Engineering at Ahsanullah University of Science and Technology (AUST), Dhaka, Bangladesh. He is passionate about innovative technologies with a strong focus on Blockchain, IoT, Machine Learning (ML), Computer Vision, and web-based intelligent systems.

Nabila Rahman holds a Bachelor of Science (B.Sc.) degree in Computer Science and Engineering from Ahsanullah University of Science and Technology (AUST), Dhaka, Bangladesh. With a strong academic background and a keen interest in emerging technologies, her core areas of focus include Blockchain, Artificial Intelligence (AI), and secure computing.