APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 7 No. 1 (2026) p. 1208-1228
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

A Blockchain-Based Vehicle Rental System Using Ethereum

Ooi Poh Qin[1], Sofia Najwa Ramli[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: sofianajwa@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2026.07.01.065

Article Info

Received: 8 June 2026
Accepted: 23 June 2026
Available online: 30 June 2026

Keywords

Vehicle Rental System, Blockchain Technology, Smart Contract

Abstract

In this digital age, vehicle rental platforms have become one of the important methods for people to rent a car temporarily for short-term travel at anytime and anywhere. However, traditional centralized vehicle rental platforms possess several problems including high intermediary fees and risk of disputation over rental agreements. The project aims to develop a partially decentralized blockchain-based vehicle rental system using Ethereum to provide a transparent, immutable rental service with minimal third-party involvement. Agile methodology is used to develop this system. Blockchain is implemented in the system to store smart contracts and transaction information to ensure immutability and tamper-proof agreements. The system successfully developed modules including registration, login, dashboard, chat, search, vehicle details, rent, review and rate, post module, rental record, damage report, damage report management, and admin management modules. The system successfully passed the designed test plan and security checklist to ensure both functionality and security of the system. The developed system will foster trust among users by ensuring tamper-proof agreements, efficient payments, and reduced costs. Future work can be done to integrate advanced features such as integrating common payment methods and launch on Mininet to further enhance the system's utility.

1. Introduction

Nowadays, vehicles such as cars and motorbikes have become one of the most important modes of travelling in people's daily lives, especially in Malaysia due to lack of accessible public transport in most areas. Hence, vehicle rental companies emerge by offering short-term rental services to people, providing a more convenient and comfortable way of travelling for short periods of time. However, the existing vehicle rental systems possesses weaknesses which lower the trust from stakeholders. New technologies such as blockchain and smart contracts therefore need to be applied in current vehicle rental system to address the weaknesses and brings better experience to stakeholders.

The existing vehicle rental system has shortcomings that cannot be ignored. Current vehicle rental system consists of several weaknesses in both operational of the business and system structure. Firstly, traditional vehicle rental services are provided through manual management which could lead to human errors such as incorrect or missing record, causing challenges in tracking agreements. Moreover, digitalized vehicle rental services typically require a centralized platform to manage and assist in transactions between renters and owners. For example, Easybook.com which is owned by a company acts as an intermediary between renter and owners [1]. This can lead to high intermediary fees, lack of trust between parties, delayed transactions, and

possibilities to fraud. In addition, renters and owners often face disputes over rental agreements due to the absence of transparent, tamper-proof contracts. These problems result in low trust from the stakeholders including both renters and owners. Owners will hesitate whether to rent out their vehicles due to fear of not getting paid or only paid in small amount whereas renters are concerned about hidden fees that could result in loss of money. Furthermore, existing blockchain-based vehicle rental system utilize owns or less popular cryptocurrency for payment, causing inconvenience and extra cost for users.

This project aims to develop a user-friendly vehicle rental system integrated with blockchain technology and smart contracts. The proposed system aims to provide peer-to-peer vehicle rental service in which guest or renters can rent a vehicle, and hosts or owners can rent out vehicles through the system. Hence, the system allows vehicle registration, vehicle rental, payment and other modules required to ensure the functionalities of the proposed system. Integration of smart contracts automates the rental processes, and blockchain ensures the immutability of the transaction data and smart contracts, and transparency of the rental processes.

2. Related Work

This section will discuss and explain the literature review done on related topics including vehicle rental system, blockchain technology, smart contract, existing systems and comparison between the existing systems and proposed system.

2.1 Vehicle Rental System

A rental service is one where customers come to seek the rental of a rental unit whereas vehicle rental service is a service that provided the customers with an option to temporarily rent a vehicle for short term use at an agreed amount of fee [2]. Traditional vehicle rental services lack systematic management, and the rental processes are done manually which in turn lead to high risks for both owners and renters due to lack of proper handling of rental records. Vehicle rental system without implementation of blockchain usually relies on centralized platform which means a third party takes control of the platform and acts as intermediary between owners and renters. This results in high intermediary fees and hidden fees collected from both owners and renters. For example, Hertz which is an American car rental company charges $118 for fuel, which was more than the actual cost to fill the tank [3]. These centralized systems or platforms usually store users' sensitive information such as credit card numbers and personal details. This brings the risk of security issues such as data breach which would cause loss for both users and service providers.

2.2 Blockchain Technology

Blockchain technology gained popularity after Satoshi Nakamoto published a paper which explained the concept of Bitcoin and the underlying technology, blockchain. Blockchain is a distributed, decentralized, public ledger for securely exchanging digital currency and transaction information [4]. Blockchain is supported by main technologies including distributed database, consensus algorithm and secure cryptographic protocol [5]. These technologies allow blockchain to have key characteristics such as decentralization, transparency, tamper proof and security.

The distributed structure of blockchain enables the blockchain to run on itself without the management by third-party. All nodes across the blockchain have equal rights and power to monitor, validate and verify the transactions and added it into blockchain. The transparency of the blockchain is also assured due to the decentralization characteristic. Consensus mechanism ensures the validity of every block and reject the tampered blockchain, resulting in the tamper proof characteristic of blockchain. This enables the development of decentralized vehicle rental system that offers transparent rental service in this project.

Ethereum is a decentralized blockchain platform that allows developers to create, test and deploy blockchain and smart contracts integrated with decentralized Application (dApp). In the development of project, Ethereum usually works together with Truffle Suite including Ganache and Truffle to aid in rapid development and testing of the dApps. Truffle Suite simplifies the complex processes associated with Blockchain development, offering features like automated contract testing, network management for deploying to any number of public and private networks, and an interactive console for direct contract communication [6]. Hence, Ethereum and Truffle Suite is utilized in this project to aid in the development of vehicle rental system integrated with blockchain technology and deployment of smart contracts. Truffle is used to create, test and manage the smart contract of the system proposed and the smart contract is then deployed to the Ethereum for execution. Ganache provides a personal Ethereum blockchain for testing the proposed system without requiring real costs or using live networks.

2.3 Smart contract

Smart contract is digital contract written in programming language which will execute automatically when the pre-defined conditions and agreements are met [7]. A smart contract is usually stored in blockchain and inherits the immutable characteristic of blockchain. This ensures that nobody can change the agreements to gain personal interest after a smart contract is deployed and in turn, protecting the interest of the individuals who are involved in the smart contract. Unlike traditional contracts which need a trustable intermediary is needed to facilitate and verify the agreements and monitor the process, smart contract is decentralized and executed automatically without the involvement of third-party. This increases the transparency of the transactions and increases trust among those who are involved in smart contracts.

By deploying a smart contract instead of using traditional contract in the proposed system, several advantages can be achieved. Firstly, smart contract is store in blockchain and immutable, it reduces the risk of contents of the contract such as agreements being tampered. In addition, all transactions are traceable because the records are stored and copied to all nodes on the blockchain. Secondly, the costs decrease as there is no intermediary fees and services fees for third-party such as agent or bank and increases the efficiency of business process. The automation of smart contracts allows the agreements to execute automatically reduces the delays which often occurs in traditional way.

2.4 Easybook.com

Easybook.com is a one stop-platform founded in 2006 which sells bus tickets in Malaysia and Singapore [1]. Besides providing bus ticket booking services, Easybook.com now has added more features including car rental service, ferry ticket and train ticket booking service and parcel delivery service. The goal of Easybook.com is to constantly provide the best service for all of their customers.

Users can register and login in the Easybook.com website. For car rental service, Easybook.com providing search function for user to search for desired car based on location, date, and time. The details of car including image, car information, rate, company, location, and price will be displayed. User can proceed to fill up the rental reservation form to rent a car. The payment methods include online banking, credit or debit card and e-wallet.

2.5 Moovby

Moovby is the largest peer-to-peer car-sharing service in Malaysia founded in 2018. Moovby allows customers to rent vehicles from a network of local hosts within the community and aims to provide convenient 24/7 access to shared cars nearby in big cities in Malaysia [8]. The webpage of Moovby provides register and login functions, search function based on location, date and time, results page listing available vehicles and car details page. However, users will need to download Moovby app for making payment using credit or debit card. In addition, as a peer-to-peer car-sharing platform, users can sign up as a host to rent their car on the website.

2.6 Car Rental Blockchain System

This is a car rental blockchain system published on GitHub which smart contracts to create decentralized car rental company. [9] User can rent a car after depositing small amount of eth (GeroliETH) on contract address. A timer is used in the system which will start counting how long does the user rent the car before dropping off. User is only allowed to rent one vehicle at a time. The system will calculate the total cost after the car is dropped off. User can pay for the rental whenever they want, and user is not allowed to rent another car before the payment is made. The currency used in this system is GeorliETH.

2.7 Comparison with Existing Systems

Table 1 shows the comparison between the existing systems including Easybooking.com, Moovby, Car Rental Blockchain System and the proposed system. All existing system and proposed system provided a login page for users to login into the system. The currency accepted by Easybook.com and Moovby is Ringgit Malaysia (RM) and for Car Rental Blockchain System on GitHub is GeorliETH which is a type of cryptocurrency whereas in the proposed system, world second largest cryptocurrency Ether is accepted for payment. Cryptocurrency eliminates the troublesome of exchanging between different types of payment methods and is more secure as the transaction data is stored in the blockchain as compared to using Ringgit Malaysia. Besides, GeorliETH used by Car Rental Blockchain System on GitHub is less popular and mainly use as testnet instead of having real-money value. For Business model, Car Rental Blockchain System on GitHub and Easybooking.com provide peer-to-business model whereas Moovby and the proposed system provide peer-to-peer which fulfil the requirements for vehicle owners to rent their car through the system. In addition, both Car Rental Blockchain System on GitHub and the proposed system applied blockchain technology and smart contracts to ensure immutability and transparency of the transaction as compared to Easybooking.com and Moovby which do not apply blockchain and smart contract. Both Car Rental Blockchain System on GitHub and the proposed system are

decentralized platform which reduce hidden costs and risk of fraud by eliminating third party involvement as compared to Easybooking.com and Moovby which are centralized platform controlled by third party. Lastly, for Easybooking.com, Moovby and the proposed system require payments before the rental of the vehicle whereas the Car Rental Blockchain System on GitHub require payment after the rental and user is allowed to choose when the payment is made. This increases the risks of no payment made after rental process. Overall, the proposed system shows improvement by implementing blockchain technology and smart contracts, and accepting popular cryptocurrency, Ether for payment.

Table 1 *Comparison with Existing Systems*

Features	Easybooking.com [1]	Moovby [9]	Car Rental Blockchain System [10]	Proposed System
Login Page	Provided	Provided	Provided	Provided
Accepted Currency	Ringgit Malaysia	Ringgit Malaysia	GeorliETH (Cryptocurrency)	Ether (Cryptocurrency)
Business Model	Peer-to-Business	Peer-to-Peer	Peer-to-Business	Peer-to-Peer
Blockchain Technology	Not applied	Not applied	Applied	Applied
Smart Contract	Not applied	Not applied	Applied	Applied
Database	Centralized	Centralized	Decentralized	Decentralized
Payment	Payment made before rental	Payment made before rental	Payment made after rental	Payment made before rental

3. Methodology

This section will discuss the methodology implemented which is Agile methodology in the development of the proposed system. Agile Methodology consists of requirement, design, development, testing, deployment and review.

3.1 Agile Methodology

Agile Methodology, which is founded by Ken Schwaber and Jeff Sutherland in 2001 is used in the development of the proposed system. The purposes of the Agile methodology are to guide developers in software development, streamlining the process of software development and allowing the development team to manage their project in a more flexible way [12]. Fig. 1 shows the phases involved in Agile Methodology. There are 6 phases including requirements, design, development, testing, deployment, and review. Agile methodology can be iterative, which means a developer can revisit and refine the phases when the requirements of the project changed and some changes need to be made in the particular phase.

Fig. 1 *Agile Software Development Diagram [13]*

The first phase is requirement phase where all requirements needed for the project is collected and documented. In this phase, the stakeholders of the system are identified as the vehicle owners who want to rent their vehicle and the renters who want to rent a vehicle for own usage. Next, the requirements are collected through observing and examining existing system, identifying the problems and possible improvements which can be made in this project. Based on the analysis of the existing system and the requirements collected, the objectives and the scopes of the system are defined.

In the Design Phase, an Object-Oriented approach is used for the development of the proposed system. System requirements are documented and system is designed using Unified Modelling Language (UML)

diagrams such as use-case diagram, sequence diagram, activity diagram, class diagram, and system flow charts. The system architecture is designed, and database design is planned by developing data dictionary. Wireframes are illustrated to design the system's interfaces. Test plan and User Acceptance Testing form are developed for future use.

The development phase includes the implementation of the system proposed based on the design created in the previous design phase. During this phase, several important steps are involved. Firstly, a database is created by using phpMyAdmin through SQL statements based on the data dictionary developed. The system interfaces are developed using HTML and CSS, while JavaScript makes the interfaces interactive and responsive. PHP and Node.js ensures the interfaces relate to and can access the database for inserting and retrieving data. Blockchain technology and smart contracts are integrated using tools like Ganache and Solidity, and security features are applied to enhance system security.

The testing phase involves the evaluation of the functionality of the developed system to ensure the system developed meets the requirements defined. During this phase, system testing, security testing, and User Acceptance Testing are carried out to ensure system's functionality and users' satisfaction. This helps in discover the weaknesses of the system in usability, functionality and security. Based on the results of this phase, the requirements of this project may change and revisit previous phase for adjustment due to the iterative characteristic of Agile methodology.

In deploying phase, the system developed will be deployed and published. Users are allowed to use the system developed in this phase. Hence, several steps must be done to ensure smooth user experience for the system developed. Firstly, the system and necessary applications should be installed and configure. In addition, user manual or user guide is developed to guide user through the system and the functions of the system. Moreover, any enhancement and adjustment on identified problems and issues can be made to improve the system's functionality and usability.

In the reviewing phase, the feedback is collected from the stakeholders including the clients and the users. Furthermore, the performance of the system is evaluated to check if it meets the requirements and objectives defined for the project and fulfils users' expectations. This is to ensure the system developed meets the stakeholders' needs and further improvement can be made based on the problems identified.

4. Analysis and Design

This section will present and explain the processes and works developed for analysis and design of the proposed system. Object-Oriented approach is used in the development of the proposed system. System requirements, Unified Modelling Language (UML) diagrams, system architecture, database design, interface design are defined and illustrated in this section.

4.1 System Development Workflow

Table 2 shows the tasks completed in each phase and the outcome produces. This outlines the flow of developing the proposed system.

Table 2 *System Development Workflow*

Phase	Task	Output
Requirement Phase	• Investigate existing system to identify requirements • Define objectives and scopes based on the requirement gathered	• Requirement document
Designing Phase	• Design a system architecture • Draw wireframe for all interfaces of the system proposed • Draw user case diagram • Draw sequence diagram • Draw activity diagram • Draw class diagram • Draw system flow chart • Develop data dictionary • Prepare test plan and User Acceptance Testing (UAT) form	• System architecture • System design • User interface design • Database design • Test plan • User Acceptance Testing form

Table 2 *(cont)*

Phase	Task	Output
Developing Phase	• Write code for user interfaces using HTML and CSS. • Write code for accessing data using php. • Write code for making interfaces interactive and responsive using JavaScript. • Create database • Write code for integrating blockchain and smart contract to the system	• User Interface • Database • Blockchain and Smart Contract • Full system
Testing Phase	• Test system usability • Test system security	• Test plan • User Acceptance Testing form
Deploying Phase	• Install system and necessary applications • Configure. setting • Write user guide	• Installed system • Usable system • User guide
Review Phase	• Collect feedback from stakeholders • Identify problems for improvement	• Possible improvement based on identified problems

4.2 System Requirements

Analyzing system requirements is an important step in the development of a system which can determines the success or failure of the developed system. This is because this step involves identifying and defining the requirements to be met by the system to ensure the developed system can fulfil user's expectation. Table 3 shows the functional requirements and Table 4 shows the non-functional requirements of the proposed system.

Table 3 *Functional Requirements*

No.	User Modules	Description
1.	Login / Register Modules	• The system should allow users to register by filling in the registration form. • The system should allow users to login using email and password.
1.	Login / Register Modules	• The system should alert users to invalid input. • The system should alert users to wrong credentials. • The system should redirect users to login page after successful registration. • The system should redirect users to home page after successful login.
2.	Search Module	• The system should allow users to search vehicles based on location, start date and end date. • The system should list the vehicles that fulfill searching conditions selected by users.
3.	Vehicle Details Module	• The system should show users the images, model, rates, and price of a vehicle to the users.
4.	Review and Rate Module	• The system should allow user to rate and write review about the rental experience after the rental period ended.
5.	Chat Module	• The system should allow users to communicate with each other.
6.	Rent Module	• The system should allow user to rent a vehicle by providing personal information.
7.	Post Module	• The system should allow users to post the vehicles details including model, images, description, and prices.
8.	Rental Record Module	• The system should allow users to view and manage rental record.

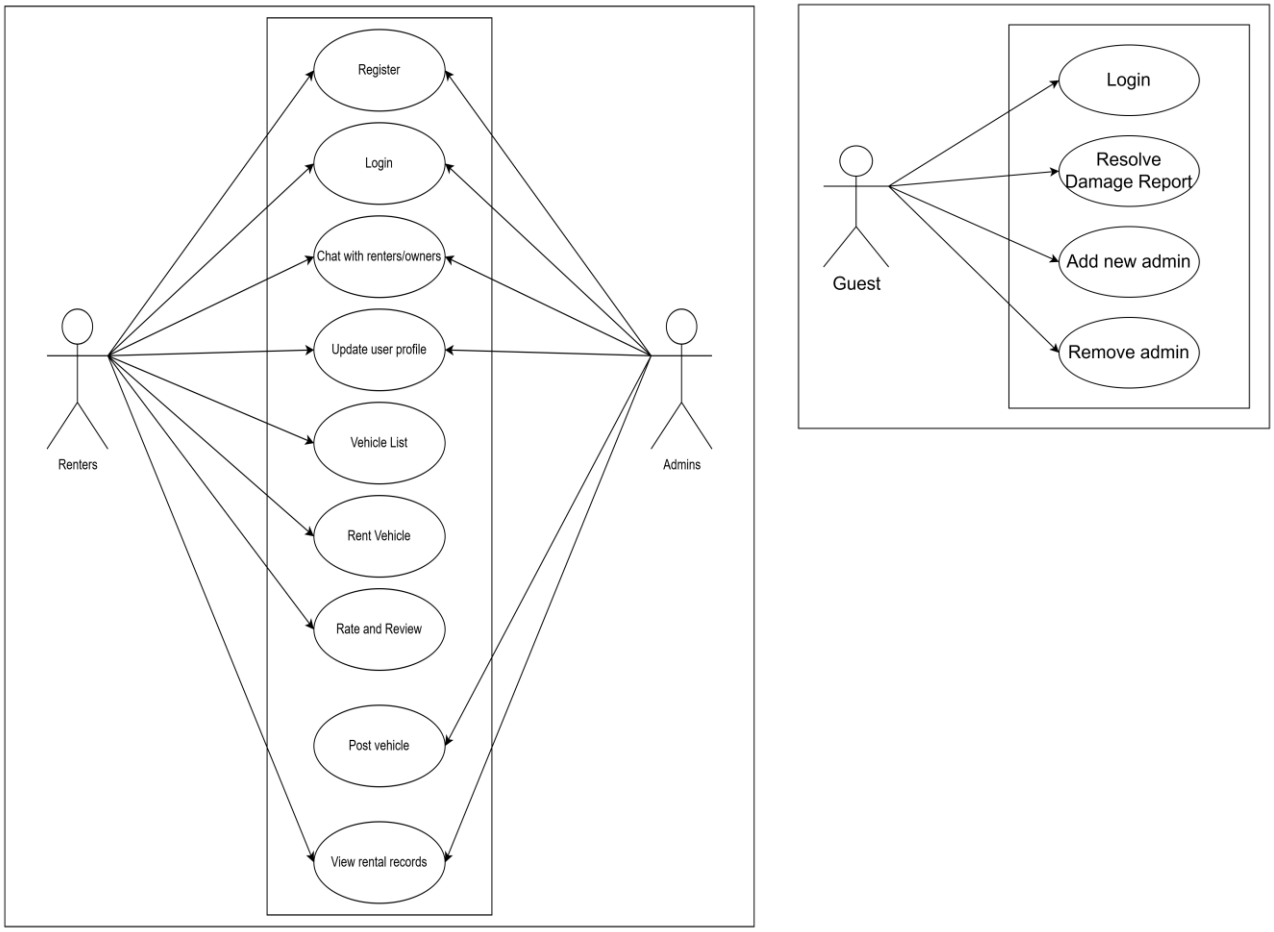
Table 4 *Non-functional Requirements*

No.	Requirements	Description
1.	Operational	• The system should be able for use at any time.
2.	Performance	• The system should be user friendly
3.	Security	• The user can only access their own accounts with correct email and password. • Blockchain technology should ensure the transaction data and smart contract's integrity.

4.3 System Analysis

System analysis is a method of determining system performance under the assumption that the system structure is known [13]. During system analysis, the system is examined, and the requirements and functionalities of the system us identified and documented by using Unified Modelling Language (UML).

Fig. 2 shows the use-case diagram of the proposed system. A guest can perform tasks such as register, login, chat with hosts, update user profile, search vehicle, rent vehicle, rate, and write review. A host can register, login, chat with guests, update user profile, post vehicles, and view rental records. An admin can login, resolve damage report, add new admin and remove admin. Fig. 3 shows the class diagram for the proposed system. Fig. 3 shows the class diagram. There are nine classes in the class diagram including Guest, Host, Vehicle, Rent, Payment, Review, Auth, ChatRoom and Message classes. Each class contains attributes and methods required.



(a) (b)

Fig. 2 *Use Case Diagram(a) Use Case Diagram for Guest and Host; (b) Use Case Diagram for Admin*

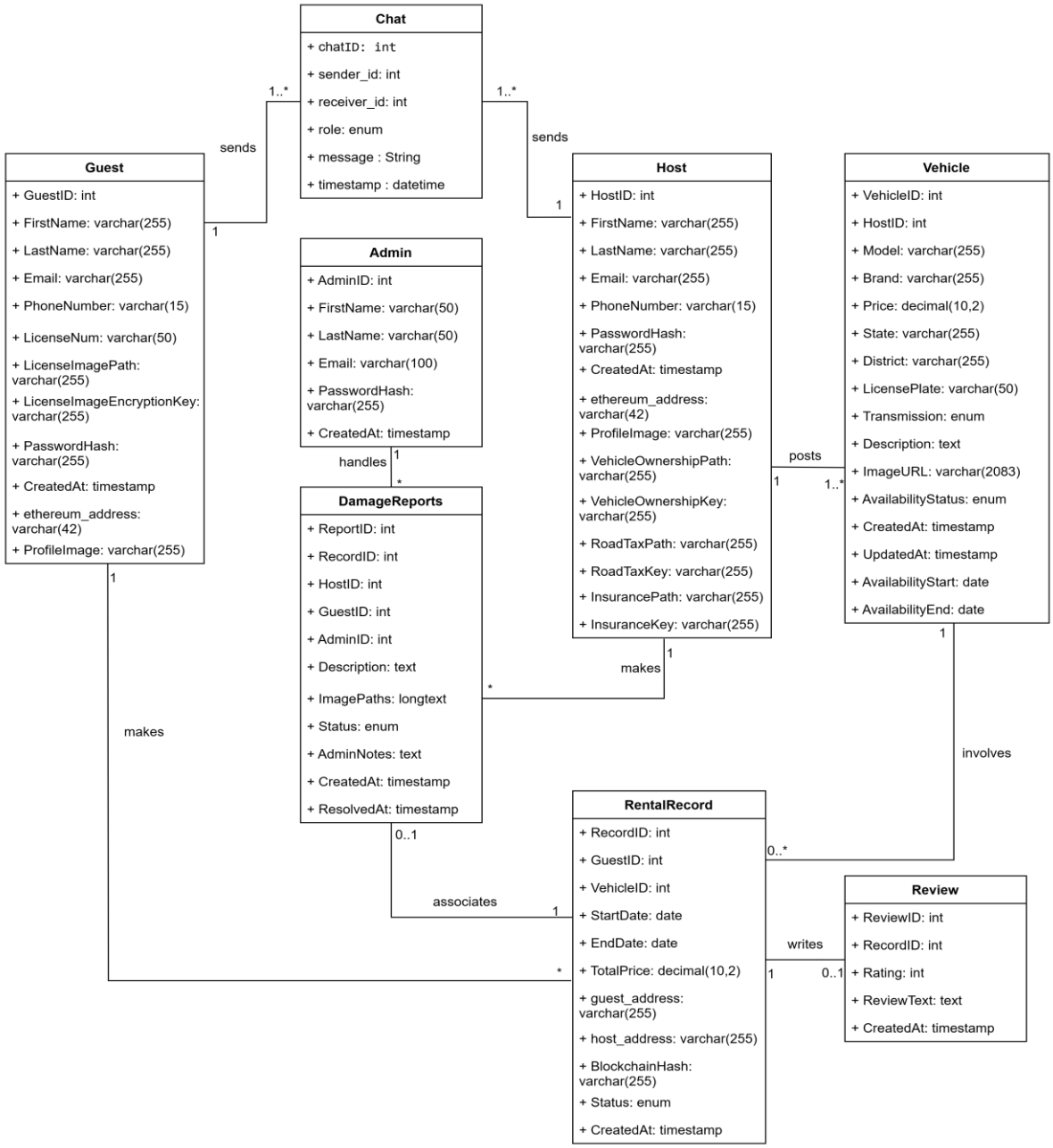
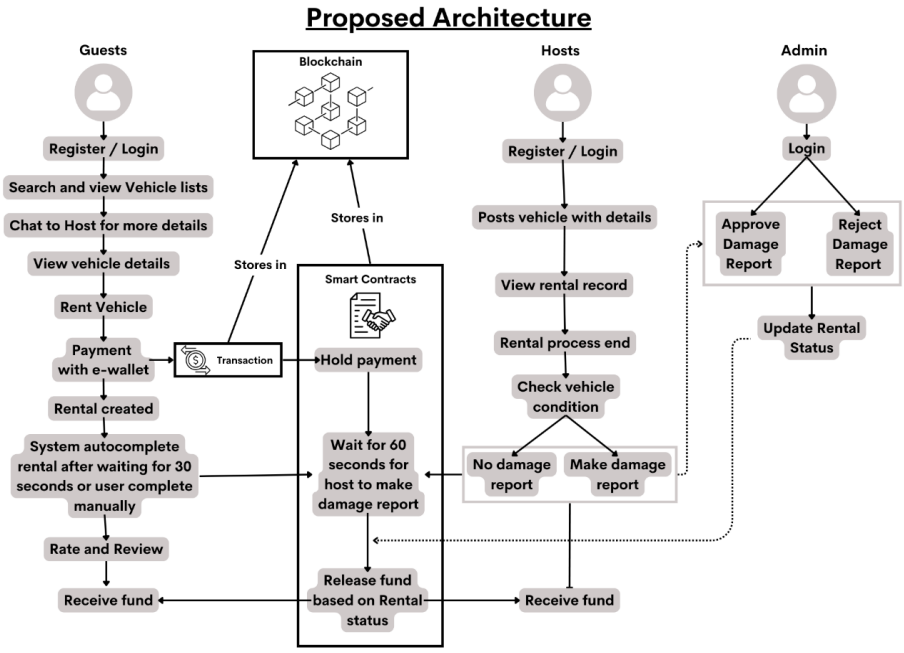


Fig 3. *Class Diagram*

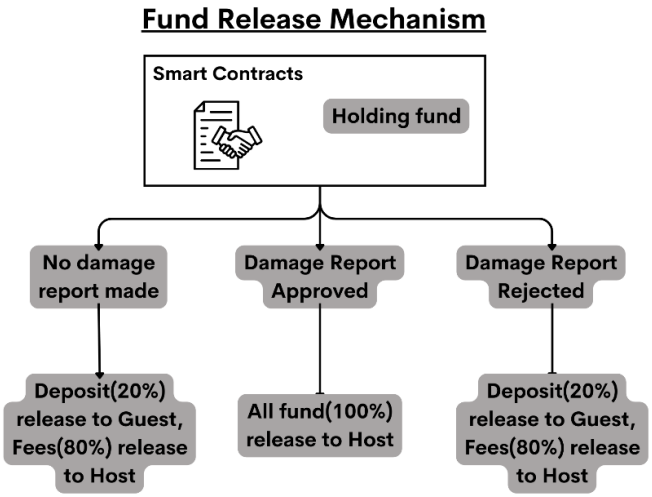
4.4 System Design

Fig. 4(a) shows the system's architecture. First, the users can register and log in as guests, hosts or admins. For guest, the user can search and browse for suitable vehicles, chat with the host for more details of the vehicle or rental process. The guests can then proceed to make payment with cryptocurrency e-wallet to rent a vehicle. Once the transaction is complete, a smart contract is created which will hold the payment until the rental period end. Both transaction data and smart contract will be stored in the blockchain. The guests will be reminded to reach out to the owner through chatting module to confirm details such as location to collect and return the vehicle. When the rental period end, the guest can complete the rental manually or the system will autocomplete the rental after waiting for 30 seconds in case the guest forgot to complete rental. Now, the system will wait for another 60 seconds for host to check on the vehicles' conditions and decide whether to make a damage report. Both durations should be 24 hours in production, however for demonstration purposes, the durations are set to a shorter time. If no damage report is made, the smart contract will proceed to fund releasing stage. If a damage report is made, the fund will continue to be held by the smart contract until admin makes decision on approving or rejecting the damage report. Based on the admin's decision, the fund will be released by smart contract.

Fig. 4(b) shows the fund release mechanism, if no damage report is made, a deposit equal to 20% of the guest's payment amount will be released to the guest and 80% of the payment amount will be released to the host. If the damage report is approved, all funds which is 100% of payment amount will be released to the host. If damage report is rejected, 20% of the payment amount will be released to the guest whereas the remaining 80% will be refunded to the host.



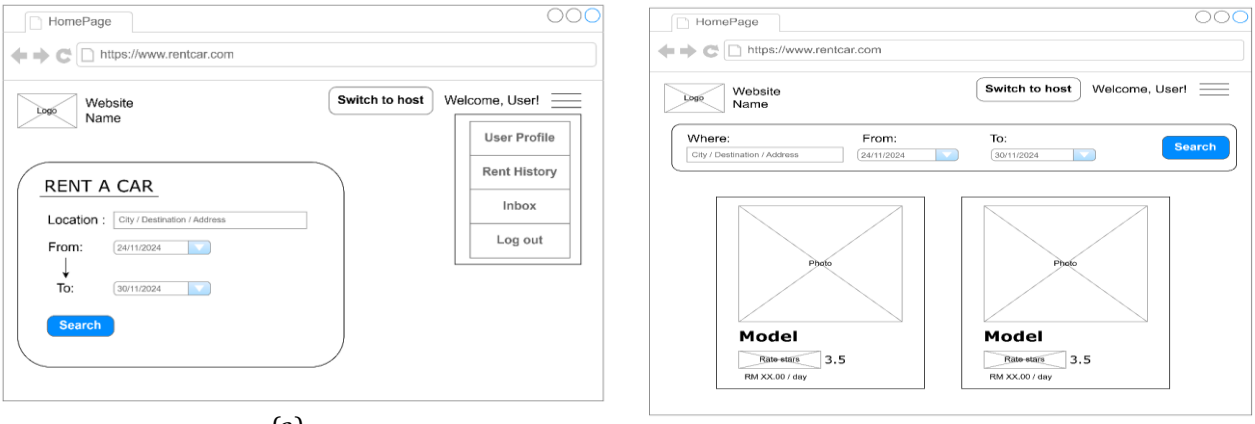
(a)



(b)

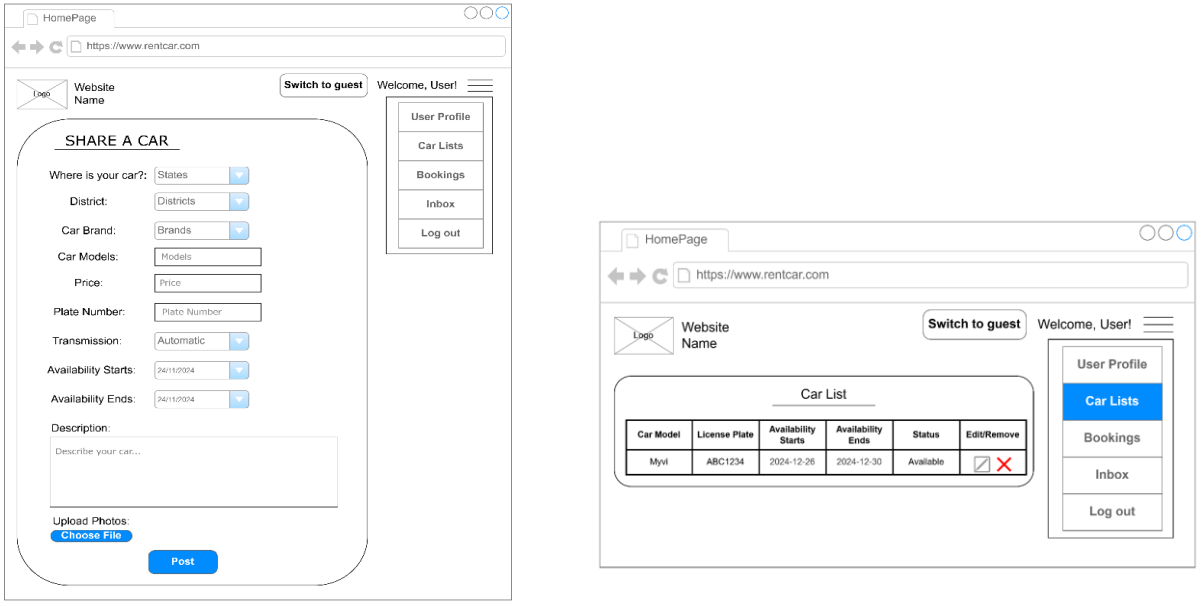
Fig. 4 *System Architecture(a) System Architecture; (b) Fund Release Mechanism*

Fig. 5(a) shows the interface design of guest dashboard. Guest will be directed to this page after login successfully and can proceed to perform search vehicle function or other tasks. Fig. 5(b) shows the search results page which displays lists of vehicles matched with the searching criteria. Host will be directed to this page after login successfully and can proceed to perform post new vehicle function or other tasks. Fig. 6(a) shows the interface design of host dashboard. Fig. 6(b) shows the vehicle lists page which shows the lists of posted vehicles by a host.



(a) (b)

Fig. 5 *Interface Design(a) Guest Dashboard; (b) Search Result Page*



(a) (b)

Fig. 6 *Interface Design(a) Host Dashboard; (b) Vehicle List Page*

4.5 Database Design

Database design defines how the data should be stored and the interrelation between the data. It helps in eliminating redundancy of data and maintaining consistency of data throughout the database.

4.5.1 Data Dictionary

Table 5 shows one of the data dictionaries of the proposed system which is the data dictionary for vehicle table. The data dictionary contains

Table 5 *Data Dictionary for Vehicle Table*

Key	Attribute Name	Data Type	Description
PK	ReviewID	int	Auto-generated unique ID for every review.
FK	RecordID	int	ID of the associated rental record.
	Rating	int	Rating given by the guest.
	ReviewText	text	Review written by the guest.
	CreatedAt	timestamp	Date and time of the review creation.

5. Implementation

This section will discuss the implementation of security modules, blockchain, user modules and admin modules. These modules ensure the security, functionality and usability of the system proposed.

5.1 Implementation of Security Modules

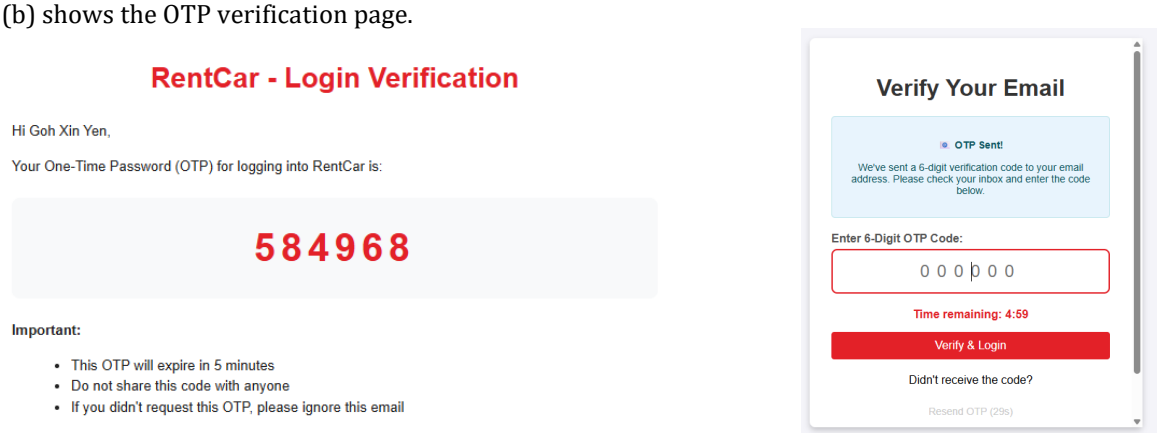
This section will explain the key security modules implemented in the proposed system, including strong password policy, two-factor authentication and file encryption and decryption. Strong password policy helps in enhancing system security by preventing brute-force attacks and reducing the risk of unauthorized access. Two-factor authentication requires user to provide both credentials and One-Time Password (OTP), enhances account security by adding an extra layer of protection against unauthorized access. Moreover, files uploaded by users are encrypted before storing and decrypted when needed helps to ensure the confidentiality of the files containing sensitive data.

During registration, users are required to set password which contains at least 12 characters, one uppercase and lowercase, one digit and one special character. Fig. 7(a) shows the code for implementing strong password policy during registration of users and Fig. 7(b) shows the error message when users try to set password which do not fulfil the requirements.

(a) (b)

Fig. 7 *Strong Password Policy (a) Code for implementing Strong Password Policy (b) Error Message for weak password*

Two-factor authentication requires user to enter correct email, password and select respective role. After verifying the account credentials, an OTP will be sent to user's email and user is required to input the OTP in the OTP verification page. If the OTP code provided by the user match with the OTP sent in the email, the user login successfully and is then redirected to the user's dashboard. Fig. 8(a) shows the OTP sent through email and Fig. 8(b) shows the OTP verification page.



(a) (b)

Fig. 8 *Two-factor Authentication (a) OTP sent to user's email (b) OTP verification page*

File containing sensitive information such as user's driving license image, vehicle ownership certificate, insurance cover note and road tax documentation are encrypted using AES-256-CBC algorithm. AES-256-CBC algorithm is a symmetric encryption algorithm which uses a 256-bit key and Cipher Block Chaining mode to encrypt data in 128-bit blocks. The first block is combined with an initialization vector (IV) using XOR, and the rest of the blocks are combined with the previous ciphertext block using XOR according to the CBC mode. Fig. 9(a) shows the code for encryption and Fig. 9(b) shows the code for decryption.

```php
/**
 * Encrypt an image using AES-256-CBC
 *
 * @param string $imageData Raw image data
 * @param string $encryptionKey Encryption key in hexadecimal format
 * @return string Encrypted image data
 */
2 references | 0 overrides
public function encrypt($imageData, $encryptionKey): string
{
    // Convert hex key to binary
    $key = hex2bin(string: $encryptionKey);

    // Generate an IV (Initialization Vector)
    $iv = openssl_random_pseudo_bytes(length: openssl_cipher_iv_length(cipher_algo: 'aes-256-cbc'));

    // Encrypt the image data
    $encryptedData = openssl_encrypt(data: $imageData, cipher_algo: 'aes-256-cbc', passphrase: $key,
    options: OPENSSL_RAW_DATA, iv: $iv);

    // Prepend the IV to the encrypted data (so we can decrypt it later)
    $encryptedImage = $iv . $encryptedData;

    return $encryptedImage;
}
```

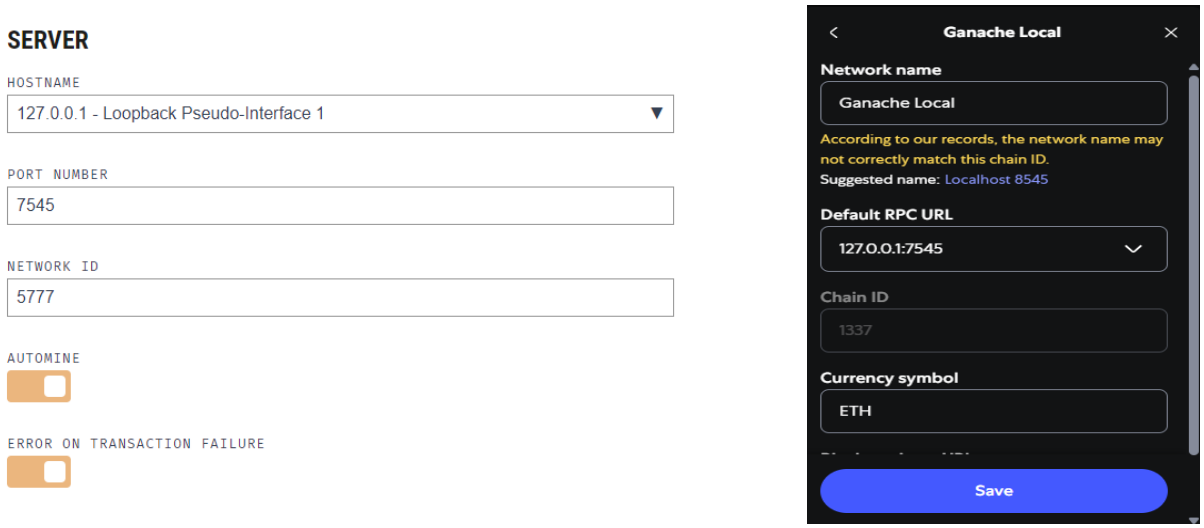
```php
/**
 * Decrypt an encrypted image
 *
 * @param string $encryptedImageData Encrypted image data
 * @param string $encryptionKey Encryption key in hexadecimal format
 * @return string|false Decrypted image data or false on failure
 */
2 references | 0 overrides
public function decrypt($encryptedImageData, $encryptionKey): bool|string
{
    // Convert hex key to binary
    $key = hex2bin(string: $encryptionKey);

    // Extract the IV (first 16 bytes for AES-256-CBC)
    $ivSize = openssl_cipher_iv_length(cipher_algo: 'aes-256-cbc');
    $iv = substr(string: $encryptedImageData, offset: 0, length: $ivSize);

    // Extract the encrypted data (everything after the IV)
    $encryptedData = substr(string: $encryptedImageData, offset: $ivSize);

    // Decrypt the data
    $decryptedData = openssl_decrypt(data: $encryptedData, cipher_algo: 'aes-256-cbc', passphrase: $key,
    options: OPENSSL_RAW_DATA, iv: $iv);

    return $decryptedData;
}
```

(a) (b)

Fig. 9 *File Encryption and Decryption (a) Code for encryption (b) code for decryption*

5.2 Implementation of Blockchain

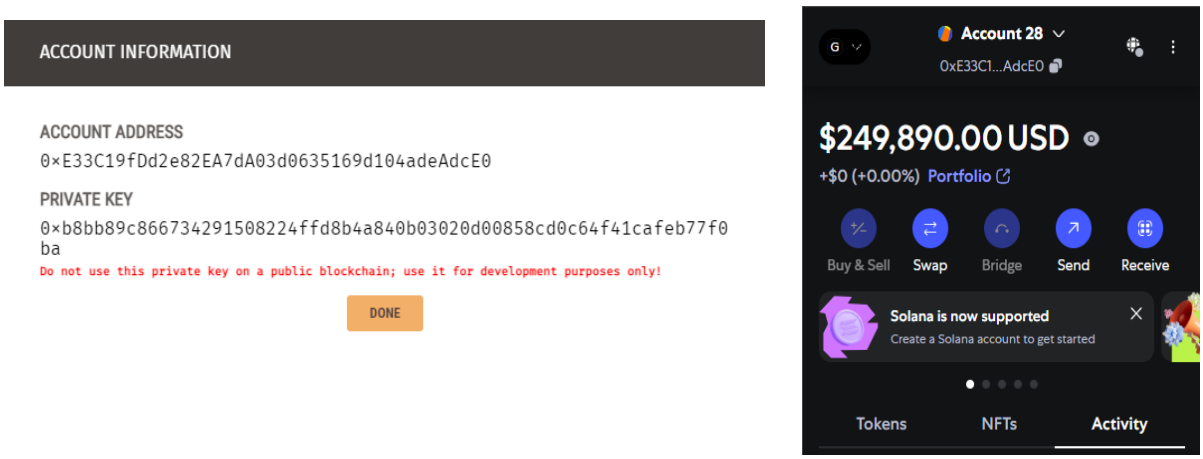
This section will explain about the implementation of blockchain, including MetaMask and Ganache setup, smart contract deployment and Node.js implementation. For setting up blockchain, the user is required to install Ganache for creating local blockchain and obtaining testnet ETH for development purpose, MetaMask e-wallet chrome extension for making transaction and connecting the blockchain to the system, Truffle for compiling and deploying smart contract to blockchain and Node.js for backend processing. A local blockchain needs to be created through Ganache GUI, and with the information provided with the local blockchain, a custom network should be conFig.d in MetaMask for connection between MetaMask and local blockchain. Fig. 10(a) shows the local blockchain created and Fig. 10(b) shows the custom network conFig.d in MetaMask.



(a) (b)

Fig. 10 *Ganache and MetaMask Setup (a) Local Block Chain Created (b) Custom Network ConFig.d*

On the network, account credentials provided in Ganache are used to import the account to MetaMask. Fig. 11(a) shows an example of account credentials provided by Ganache. By using the private key, an e-wallet account is successfully imported into MetaMask in Fig. 11(b). The account will be used for transaction purpose, connecting the system with local blockchain.



(a) (b)

Fig. 11 *MetaMask Account (a) Credentials Provided by Ganache (b) MetaMask Account Imported*

Furthermore, Solidity extension is required to be installed in Visual Studio Code (VS Code) for writing smart contract whereas Truffle which is a development framework that helps in compiling, testing, and deploying smart contracts to the blockchain needed to be installed using npm command. Smart contract is a self-executing contract stored on a blockchain that automatically enforces the terms of an agreement when predefined conditions are met.

In this system, smart contract automates the rental process through several key functions written. createRental() function allows a guest to initiate a rental by specifying the host's address and sending Ether through MetaMask and the payment is automatically split into a 20% deposit and 80% trip fee. completeRental() function allows guest to complete rental manually while autoCompleteRental() function will automatically complete the rental if the guest does not complete the rental within 1 minute. After completion, reportDamage() function allows the host to report any damage within 3 minutes. If no damage is reported after this period, autoFinalizeRental() function will be triggered to release the trip fee to the host and return the deposit to the guest. If a damage is reported by the host, resolveDispute() function allows admin to determine whether the host or guest should receive the deposit. The standardized agreement is stored in smart contract through IPFS link to prevent tampering with the rental agreement. Fig. 12 shows the smart contract written using VS Code.

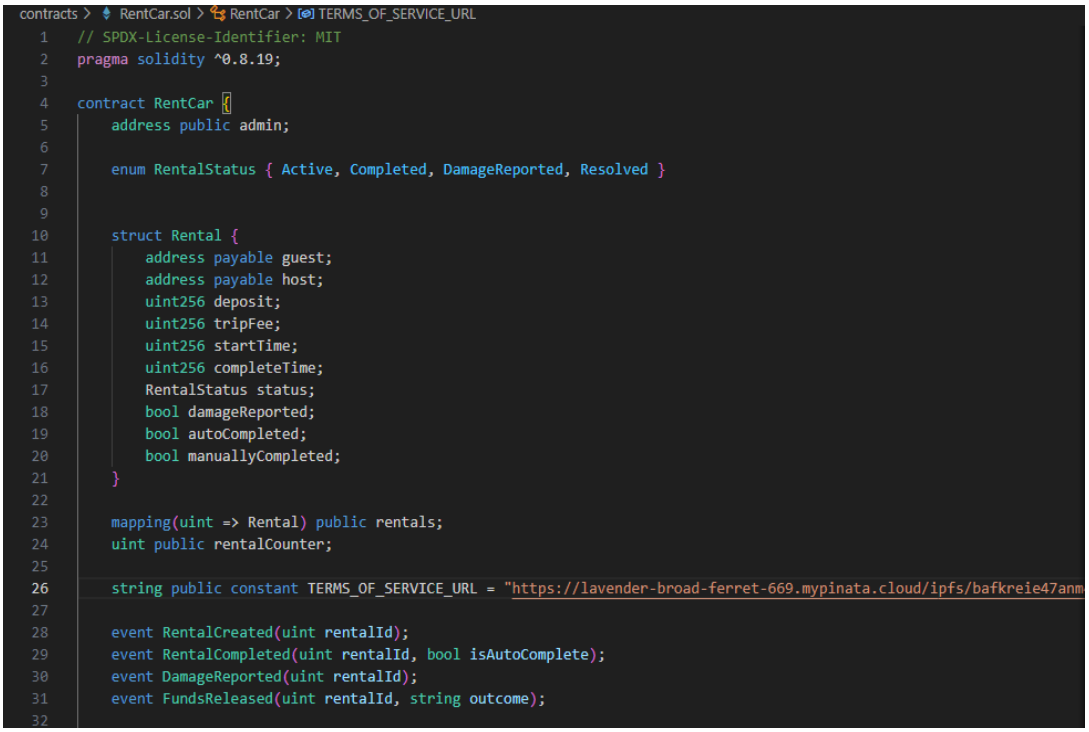
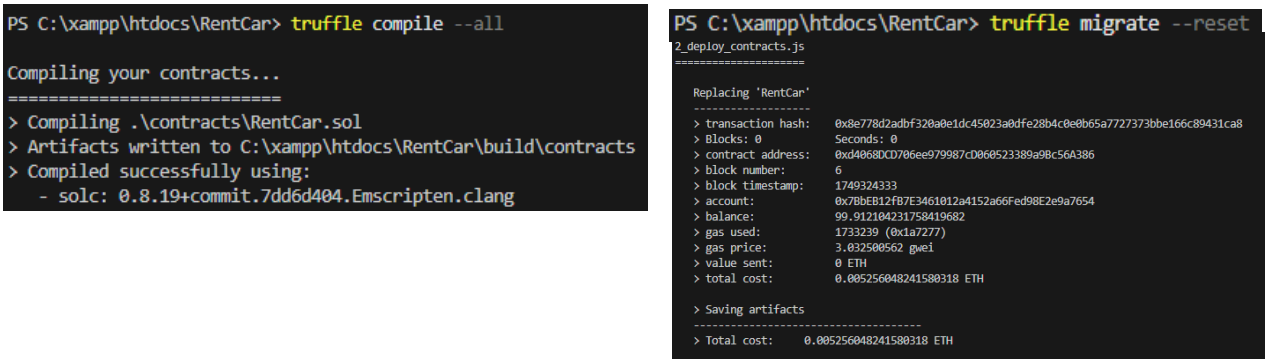


Fig. 12 *Smart Contract*

A Node.js file called truffle-config.js is created and imported into Ganache to connect smart contract to local blockchain. The smart contract is then compiled and deployed using Truffle commands. Fig.13 shows truffle-config.js file. Fig. 14(a) shows the smart contract is compiled and Fig. 14(b) shows the smart contract is deployed using Truffle commands. Fig. 15 shows the deployed smart contract on Ganache.

Fig. 13 *truffle-config.js*



(a) (b)

Fig. 14 *Truffle Commands (a) Commands for Compile Smart Contract (b) Commands for Deploy Smart Contract*

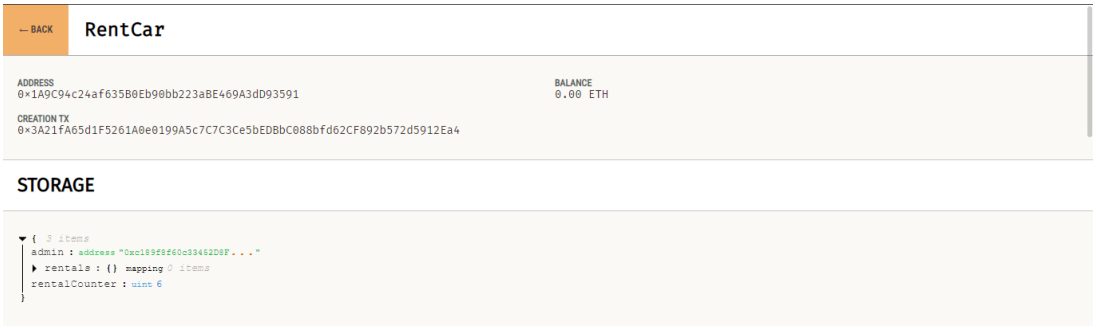


Fig 15. *Deployed Smart Contract in Ganache*

Ganache is a local blockchain for development purpose and thus, unlike the production blockchain, Ganache only advances time when transactions occur. Hence, Node.js file is created with advanceBlockTime() function to ensure the time dependent functions in smart contract can work correctly. This function sends small dummy transactions every five seconds to simulate continuous block creation, causing time in Ganache to move forward, allowing the time-dependent functions to work effectively. Fig. 16 shows the advanceBlockTime() function.

```
// Advance time in Ganache by sending dummy transaction
async advanceBlockTime() {
    try {
        const tx = await this.wallet.sendTransaction({
            to: this.wallet.address,
            value: ethers.utils.parseEther("0.0001")
        });
        await tx.wait();
    } catch (error) {
        //no error, this is only for advancing time
    }
}
```

Fig. 16 *advanceBlockTime() function*

5.3 Implementation of Module

In this section, the modules implemented in the proposed system will be discussed. These modules allow user to perform actions and interact with the system, ensuring the usability and functionality of the system. Fig. 17 shows the Login page of the system.

5.3.1 Login Module

The Login Module allows user to login using different roles such as admin, guest, and host using correct credentials and selected role. Each role is directed to their respective dashboard upon successful authentication, ensuring role-based access control throughout the system.

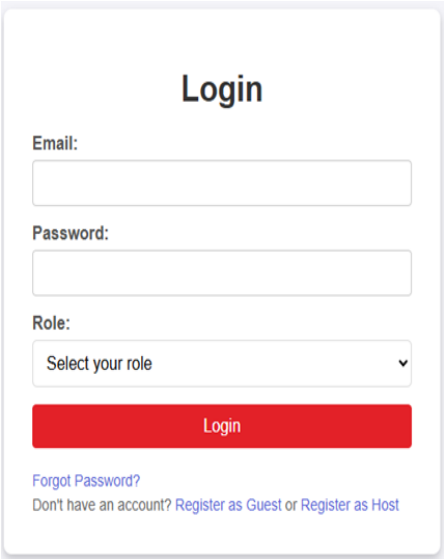
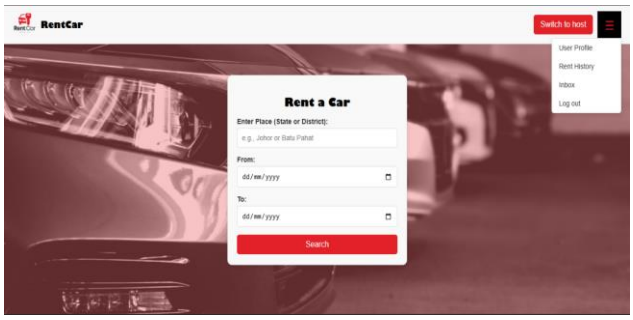


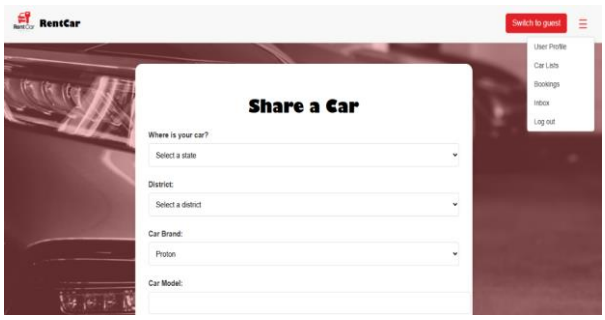
Fig. 17 *Login Page*

5.3.2 Dashboard Module

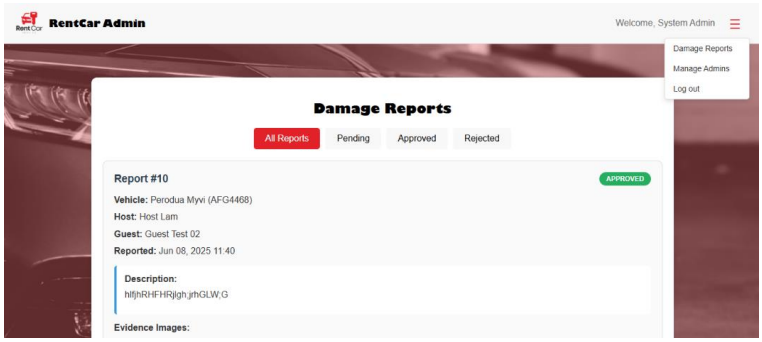
Dashboard module provides role-based centralized interface, in which user can access other modules related to their respective role. The Guest Dashboard allows guest to perform vehicle searching function and navigates to other modules including user profile, rental history, and chat modules. The Host Dashboard allows host to perform vehicle registration function and navigates to other modules including user profile, vehicle management, booking management and chat modules. The Admin Dashboard allows admins to manage reports and manage admin users. Fig. 18(a) shows the Guest Dashboard and Fig. 18(b) shows the Host Dashboard whereas Fig. 18(c) shows the Admin Dashboard.



(a)



(b)

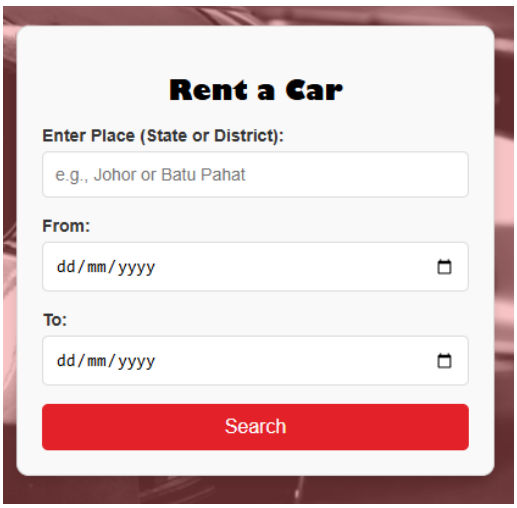


(c)

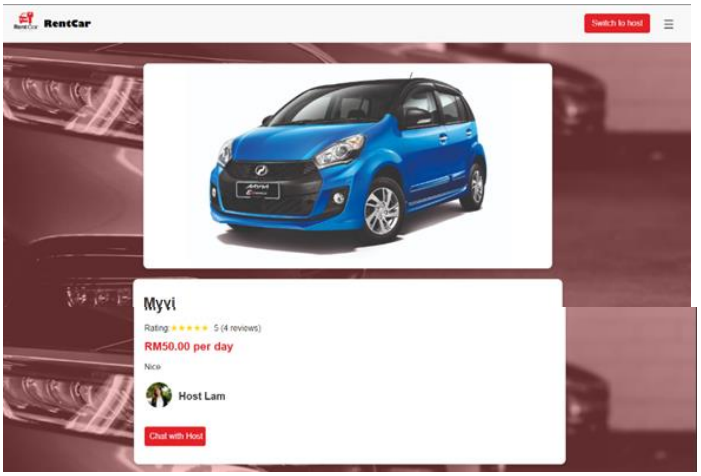
Fig. 18 *Dashboard (a)Guest Dashboard (b)Host Dashboard (c)Admin Dashboard*

5.3.3 Guest-Related Modules

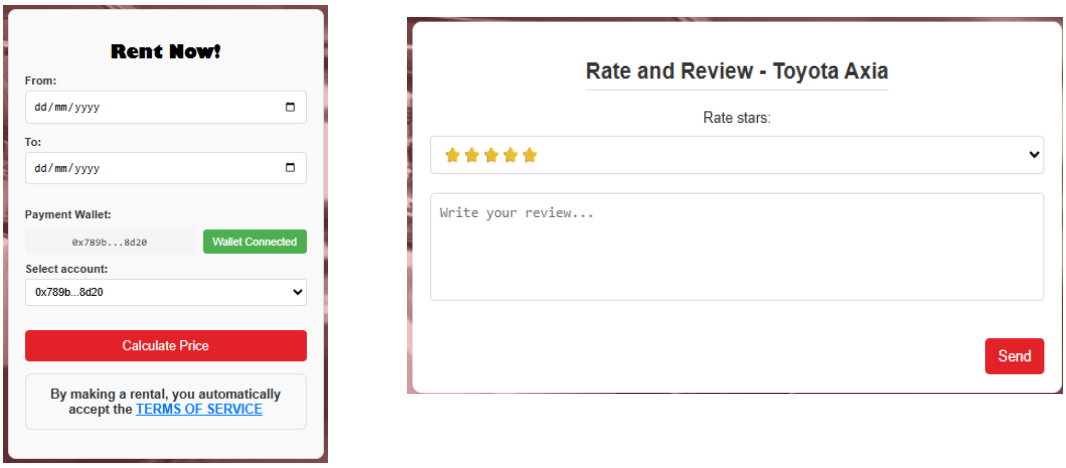
Guest-related modules include search module, vehicle details module, rent module, review and rate module. The Search Module enables users to search vehicles based on location and availability and the matching results will be displayed. The Vehicle Details Module displays information such as images, pricing, host details, and user reviews. The Rent Module manages the entire booking process, including date selection, price calculation, and confirming booking. After completing a rental, users can share their experiences through the Review and Rate Module, which allows rating from 1 to 5 and submitting written reviews that are later displayed on the vehicle details page for future users. Fig. 19 shows the guest-related modules



(a)



(b)

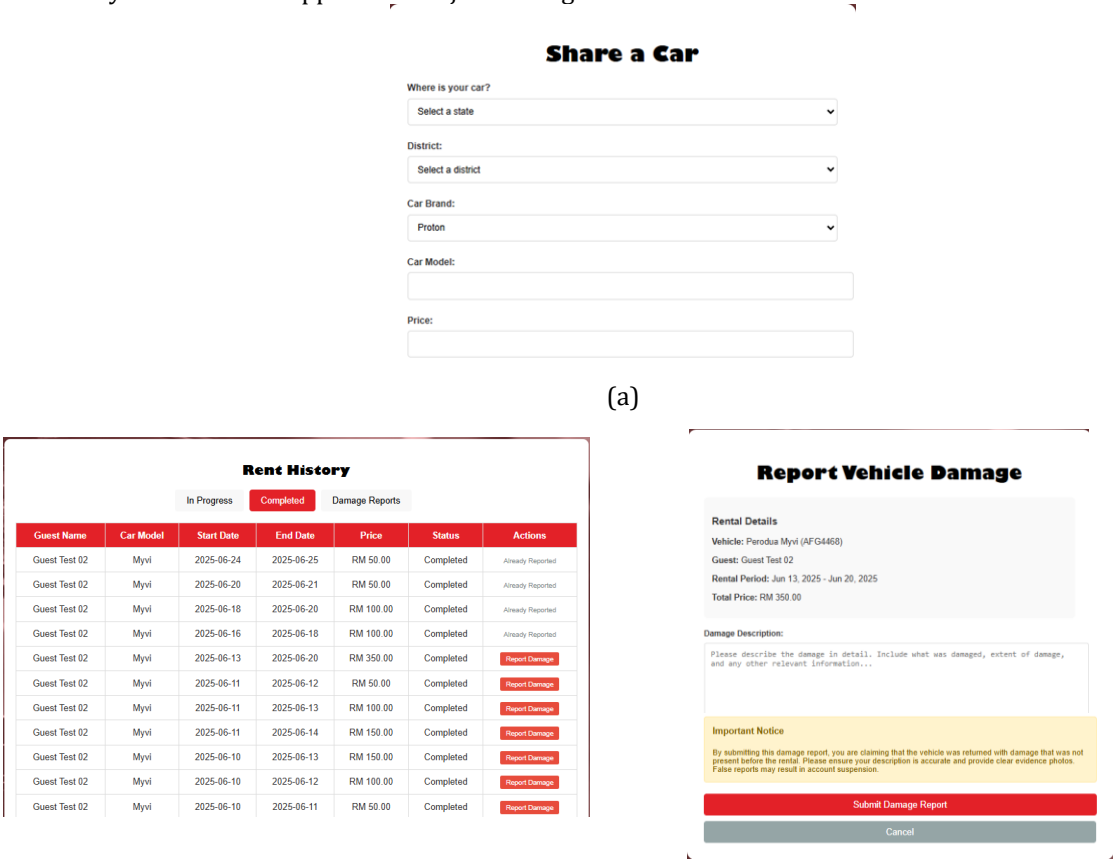


(c) (d)

Fig. 19 *Guest-Related Module (a)Search Module (b)Vehicle Details Module (c)Rent Module (d)Rate and Review Module*

5.3.4 Host-Related Modules

Host-related modules include post module, rental record module and damage report module. The Post Module allows users to register and post new vehicles by providing required information. O After submitting the form, the vehicle can be rented by other users. The Rental Record Module enables hosts to view and manage their rental history, including in-progress rentals, completed rentals, and submitted damage reports. The Damage Report Module allows hosts to submit reports with descriptions and images, which will then be reviewed by the admin for approval or rejection. Fig. 20 shows the host-related modules.



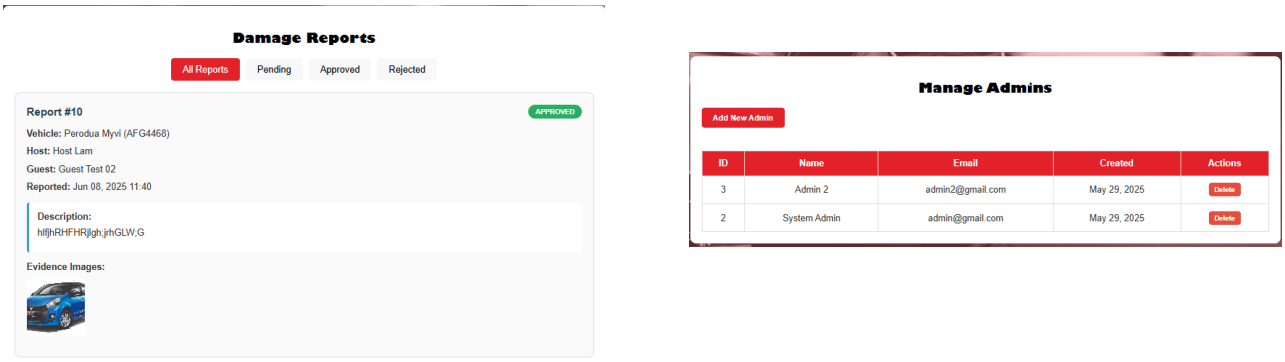
(a)

(b) (c)

Fig. 20 *Host-Related Modules (a) Post Module (b) Rental Record Module (c) Damage Report Module*

5.3.5 Admin-Related Modules

Admin-related modules include damage report management module and admin management module. Damage report management module allow admin to view and approve or reject the report submitted by the users. After viewing the details of the report, the admin can decide to approve the report and release the deposit to the host or reject the report which means the deposit will be returned to the guest. Admin management module provides secure control over administrative accounts within the system. Access to this module is strictly limited to existing admins, ensuring that only authorized personnel can create or remove admin accounts. Fig. 21 shows the admin-related modules.



(a) (b)

Fig. 21 *Admin-Related Modules (a)Damage Report Module (b)Admin Management Module*

6. Testing

This section will discuss the testing result of the proposed system. The system will be tested in two approaches, which are through test plan and User Acceptance Test (UAT).

6.1 Test Plan Result

In this section, the result of test plan will be displayed. The test plan consists of two parts, including system testing plan and security checklist.

Table 6 *Test Plan Result*

No.	Description	Expected Result	Actual Result
1.	Register (all required field fulfill requirements)	User can fill up registration form and is redirected to Login Page after success registration.	Pass/~~Fail~~
2.	Login (Input correct email, password, role and matched OTP)	User can fill up login form and is redirected to respective dashboard after successful login	Pass/~~Fail~~
3.	Chat (Send and receive messages)	User can send and receive message.	Pass/~~Fail~~
4.	View User Profile (View and update user profile)	User profile is displayed and User can view and user profile.	Pass/~~Fail~~
5.	Search Vehicle (Input location and desired date)	User can fill in search form and search result is displayed.	Pass/~~Fail~~
6.	View Vehicle Details (Select desired vehicle)	Vehicle details including image, host, rate and review are displayed.	Pass/~~Fail~~
7.	Rent Vehicle (Input start date, end date and confirm rental)	User can fill up rental form and rent successful message is displayed.	Pass/~~Fail~~
8.	Rental History (View rentals history)	Current and complete rentals are displayed.	Pass/~~Fail~~
9.	Post New Vehicle (Input all required information)	User can fill up post vehicle form	Pass/~~Fail~~

Table 6 *(cont)*

No.	Description	Expected Result	Actual Result
10.	Posted Vehicle (View and manage posted vehicles)	List of posted vehicles is displayed.	Pass/~~Fail~~
11.	Rental Record (View and manage rental records)	In-progress and completed rentals are displayed.	Pass/~~Fail~~
12.	Report Damage (Input all required information)	User can fill up report damage form.	Pass/~~Fail~~
13.	Resolve Damage Report (View and approve or reject report)	User can view damage report and approve or reject the report.	Pass/~~Fail~~
14.	Manage Admin (Add new admin and remove existing admin)	User can fill up add admin form and remove existing admin.	Pass/~~Fail~~

Table 7 *Security Checklist*

No.	Expected Result	Actual Result
1.	Ensure the error message does not directly indicate which part of the authentication data is incorrect	Pass/~~Fail~~
2.	Enforce the complexity of the password such as a combination of alphabetic and numeric when creating a password.	Pass/~~Fail~~
3.	Enforce the password length which is a minimum of twelve characters.	Pass/~~Fail~~
4.	Enforce password salting and hashing	
5.	Two-factor authentication is implemented using a combination of email and password along with a one-time password (OTP).	Pass/~~Fail~~
6.	Ensure files uploaded are encrypted using AES-256-CBC algorithm and decrypt when needed.	Pass/~~Fail~~
7.	Prevent SQL injection	Pass/~~Fail~~
8.	Session checks are implemented to ensure that only authenticated users can access protected pages, and sessions are destroyed upon logout to prevent unauthorized access.	Pass/~~Fail~~
9.	Smart contract passes security scan using Slither with no critical vulnerabilities	Pass/~~Fail~~

6.2 User Acceptance Form Results for User Module

This section presents the results of the User Acceptance Testing (UAT) for both the guest and host modules. Respondent for guest role are selected among individuals who are experienced in ride-sharing or car rental services whereas respondents for host role are selected among individuals who own a vehicle and experienced in ridesharing or car rental services. Furthermore, admins are selected from staffs who work in car rental services. Table 8 shows the UAT results for Demographic. Table 9 shows the UAT results for guest and host whereas Table 10 shows the UAT results for admin.

Table 8 UAT Results for Demographic

No	Questions	Role	Respondents
1.	You have experience using ridesharing or car rental services as a customer.	Guest	10
2.	You currently own a vehicle and have experience or interested in offering it for ride-sharing or rental purposes.	Host	10
3.	Are you currently working or have previously worked as a staff member in a car rental service company?	Admin	5

Table 9 *UAT Results for Guest and Host*

No.	Question	Result
		Disagree 1 – 5 Agree
Register Module		
1.	User able to register an account as a guest or host	5
Login Module		
1.	User able to login as a guest or host	5
2.	User can receive a One-Time-Password (OTP) through email during login process	5
General Modules		
1.	User able to view and edit user profile	5
2.	User able to chat with guest/host	5
Guest-related Module		
1.	User able to search vehicles	5
2.	User able to view vehicle details	5
3.	User able to make payment via cryptocurrency e-wallet (MetaMask)	5
4.	User able to view rental history	5
5.	User able to rate and write review after completing rental	5
Host--related Modules		
1.	User able to post a new vehicle	5
2.	User able to view posted vehicles	5
3.	User able to view rental records	5
4.	User able to submit a damage report-	5
System Usability		
1.	The system is user-friendly and not confusing	4.9
2.	The system can function properly	4.9

Table 10 *UAT Results for Admin*

No.	Question	Result
		Disagree 1 – 5 Agree
Login Module		
1.	User able to login as an admin	5
2.	User can receive a One-Time-Password (OTP) through email during login process	5
Admin-related Module		
1.	Admin able to view Damage Reports	5
2.	Admin able to approve or reject Damage Reports	5
3.	Admin able to add new admins	5
4.	Admin able to remove existing admins	5
System Usability		
1.	The system is user-friendly and not confusing	4.8
2.	The system can function properly	5

7. Conclusion

In conclusion, a blockchain-based vehicle rental system is developed based on the requirements defined and design illustrated. The system achieves the objectives and scopes defined.

This system provides several advantages including, implementation of strong security features and blockchain technology with smart contracts which ensures transparent, immutable, and tamper-proof rental agreements and provides automated contract execution The system provides a partially decentralized peer-to-peer vehicle rental platform that connects vehicle owners directly with renters, eliminating intermediary fees to reduce costs.

However, the system possesses several disadvantages which are the system can only pay by using cryptocurrency e-wallet, which is less popular in Malaysia, and it is built and launched on the Ethereum Testnet instead of the Mainnet. Moreover, the system operates without intermediary fees which leads to challenges ongoing maintenance.

Based on the advantages and disadvantages outlined, there are several improvements that can be made, including integrating common payment methods such as online banking, credit and debit cards, launching the system on Ethereum Mainnet and introducing ads and sponsored vehicle listings to cover maintenance cost.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: **study conception and design:** *P. Q. Ooi, S. N. Ramli*; **data collection:** *P. Q. Ooi, S. N. Ramli*; **analysis and interpretation of results:** *P. Q. Ooi, S. N. Ramli*; **draft manuscript preparation:** *P. Q. Ooi, S. N. Ramli*. All authors reviewed the results and approved the final version of the manuscript.

References

[1] Easybook.com, "Easybook.com." Accessed: Nov. 14, 2024. [Online]. Available: https://www.easybook.com/en-my

[2] A. Thakur, "Car Rental System," Int J Res Appl Sci Eng Technol, vol. 9, no. VII, pp. 402–412, Jul. 2021, doi: 10.22214/ijraset.2021.36339.

[3] H. Jack, "'Insane,' fumes Hertz customer after finding a hidden $118 charge – firm 'deliberately' make it hard to dispute fees," The U.S. Sun, Jul. 2024, Accessed: Jan. 02, 2025. [Online]. Available: https://www.the-sun.com/motors/12050128/hertz-rental-hidden-charge-dispute-fees/?utm_source=chatgpt.com

[4] A. S. Gaikwad, "Overview of blockchain," Int J Res Appl Sci Eng Technol, vol. 8, no. 6, pp. 2268–2270, 2020.

[5] W. Li, M. He, and S. Haiquan, "An Overview of Blockchain Technology: Applications, Challenges and Future Trends," in 2021 IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC)2021 IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC), 2021, pp. 31–39. doi: 10.1109/ICEIEC51955.2021.9463842.

[6] O. K. Alia, D. Mohammad Suleiman, and H. Ameen Noman, "IHSAN: A Secure and Transparent Crowdfunding Platform Leveraging Comprehensive Decentralized Technologies," IEEE Access, vol. 12, pp. 179050–179064, 2024, doi: 10.1109/ACCESS.2024.3508459.

[7] Z. Zheng et al., "An overview on smart contracts: Challenges, advances and platforms," Future Generation Computer Systems, vol. 105, pp. 475–491, 2020, doi: https://doi.org/10.1016/j.future.2019.12.019.

[8] Moovby, "Moovby." Accessed: Nov. 14, 2024. [Online]. Available: https://www.moovby.com/my/en

[9] N. Gautam, M. Kumar, and N. Kumar, "Car Rental Blockchain System ." Accessed: Dec. 30, 2024. [Online]. Available: https://github.com/Nikhil1601/Car-Rental-Blockchain-System?tab=readme-ov-file#general-information

[10] K. Schwaber and J. Sutherland, "The scrum guide," Scrum Alliance, vol. 21, no. 1, pp. 1–38, 2011.

[11] N. Okeke, "Agile Methodology: Meaning, advantages, disadvantages & more," TargetTrend. Accessed: Nov. 28, 2024. [Online]. Available: https://targettrend.com/agile-methodology-meaning-advantages-disadvantages-more/

[12] S. Sieniutycz, "Systems design: Modeling, analysis, synthesis, and optimization," in Complexity and Complex Thermo-Economic Systems, Elsevier, 2020, pp. 85–115. doi: 10.1016/B978-0-12-818594-0.00005-2.