PROJECT REPORT

SMART CHAIN AUTO RENTALS

A Blockchain-Powered Decentralized Car Rental Platform

Prepared and Submitted by:
Ajaymishra Saddi
Srilekha Kashireddy
Mahesh Vamsi Kode

Project Advisor:
Dr. Feng George Yu

ABSTRACT

Smart Chain Auto Rentals is a blockchain-based decentralized rental management system designed to transform every aspect of the traditional automobile rental process. Conventional rental systems heavily depend on centralized databases, human verification, manual approval steps, and trust-based interactions between customers and rental companies. These structures, although widely used, suffer from recurring problems such as hidden or unexplained charges, lengthy processing times, tampered rental histories, inconsistent availability updates, and the potential for unauthorized modifications of customer or rental data. In addition, disputes often arise due to the lack of verifiable evidence, unclear communication, and the absence of an immutable transaction record. Smart Chain Auto Rentals addresses these long-standing issues by replacing the centralized model with a fully automated, trust less, and transparent system powered by Ethereum-compatible smart contracts.

The platform introduces a series of fundamental improvements to the rental workflow by automating transactions, enforcing rental rules through code instead of human decisions, and ensuring that every rental event becomes a permanent and verifiable blockchain entry. Through decentralized identity integration using MetaMask, users can authenticate themselves securely without relying on traditional passwords or centralized authentication servers. Automated escrow mechanisms ensure that payments remain locked and tamper-proof during the rental process, and funds are released only when contract-defined conditions are fulfilled. This eliminates disputes commonly seen in traditional systems where customers must rely on the honesty or judgment of a rental agent. The deterministic nature of the smart contract logic guarantees that all users—renters and car owners—are treated equally and according to the exact rules encoded on-chain.

Furthermore, Smart Chain Auto Rentals uses modern Web3 development tools such as Solidity for smart contract construction, Hardhat for blockchain simulation and testing, and ethers.js for seamless communication between the front-end interface and the blockchain network. These components work in harmony to create a system where the user interface directly reflects blockchain state, ensuring that car availability, rental conditions, and payment statuses are always accurate, real-time, and impossible to manipulate. The clean and responsive user interface abstracts the complexity of the underlying blockchain operations, enabling users with little or no technical knowledge to interact fluidly with decentralized technology.

As a decentralized car rental platform, Smart Chain Auto Rentals goes far beyond a simple digital marketplace. It represents an entirely new operational framework in which smart contracts autonomously manage vehicle reservations, rental payments, return confirmations, and revenue collections. Because every action is recorded on a distributed ledger that cannot be altered or erased, the system dramatically reduces the risk of human errors, fraudulent modifications, and unfair practices. This immutability also strengthens accountability, as all payment flows, rental durations, and state changes are permanently available for auditing.

In summary, Smart Chain Auto Rentals demonstrates how blockchain technology can revolutionize mobility services by delivering an efficient, secure, transparent, and fully automated rental process. This report provides a comprehensive exploration of the project's architecture, design principles, implementation details, smart contract structure, system behavior under various test conditions, security considerations, limitations, and future enhancements. Through this project, we show how decentralized technologies can reshape traditional business workflows and create next-generation rental systems that operate without intermediaries while maintaining the highest standards of security, fairness, and trust.

INTRODUCTION

The automotive rental industry, despite presenting itself as modern and digitally advanced, continues to operate on deeply rooted centralized infrastructures controlled entirely by rental companies. Although many services offer sleek websites and mobile applications that give customers the impression of convenience and technological sophistication, the truth is that the core processes behind these platforms remain manual, slow, and vulnerable to manipulation. Most rental companies still rely on human verification for identity checks, manual data entry for updating vehicle status, subjective assessments for damage evaluation, and internal decision-making for applying additional charges. These outdated practices often result in inconsistencies and create gaps in accountability. Because the entire system depends on centralized databases owned and controlled by the rental company, there is no external mechanism to verify the accuracy or fairness of rental records. Companies can modify, edit, or delete entries such as rental history, payment details, or damage assessments without leaving behind any audit trail, ultimately denying transparency to the customer.

In addition to data manipulation risks, centralized rental systems also create opportunities for unfair pricing practices or unexpected charges that renters cannot dispute effectively. Since customers have no way to independently verify the rental conditions or the integrity of stored records, they must rely solely on the rental provider's words. This imbalance of power frequently leads to disputes, dissatisfaction, and mistrust in the system. Moreover, centralized rental databases are attractive targets for cyberattacks. If a security breach occurs, sensitive customer information—including personal identification and payment data—may be compromised, exposing customers to identity theft or fraud. These long-standing issues expose the fragility of centrally controlled rental systems and highlight the urgent need for a more secure, transparent, and verifiable alternative.

Blockchain technology introduces a transformative paradigm shift by providing decentralized, transparent, and tamper-proof storage of rental records while ensuring code-driven automation of rental agreement execution. Smart Chain Auto Rentals uses smart contracts—self-executing programs deployed on a blockchain—to define, enforce, and automate every aspect of the rental lifecycle. This eliminates the need for human oversight or company-controlled decision-making. Instead of depending on passwords or centrally stored user accounts, Smart Chain enables renters and owners to authenticate themselves through MetaMask, which acts as a decentralized identity and secure transaction signing layer. By using wallet-based authentication, the platform significantly reduces the risk of unauthorized access or identity theft, freeing the system from traditional authentication vulnerabilities.

The adoption of smart contracts allows Smart Chain Auto Rentals to automate essential rental processes such as verifying availability, calculating rental costs based on duration, recording rental entries, updating car status, and releasing funds only after the rental is completed. Every interaction between the renter and owner is logged as an immutable blockchain event, ensuring that all records are fully transparent, verifiable, and resistant to tampering. Because these entries cannot be altered, deleted, or manipulated after creation, the platform ensures fairness for all participants and eliminates the possibility of disputes caused by inconsistent or unverified data.

This report presents a comprehensive exploration of Smart Chain Auto Rentals, detailing the motivation behind its development, the limitations of legacy rental systems, and the architectural decisions that shaped the final implementation. It examines the technical foundations of the system, including smart contract design, blockchain behavior, event-driven synchronization, and front-end integration. The report also evaluates the security posture of the system, presenting how smart contracts protect against common attacks and ensure the safety of user funds. Experimentation and testing are discussed in depth, demonstrating how the system performs under various rental scenarios and multi-user interactions. Finally, the report looks ahead to the long-term potential of Smart Chain, highlighting possible enhancements, scalability opportunities, and future integrations with advanced Web3 technologies.

PROBLEM STATEMENT

The problems inherent in traditional rental systems stem from their centralized nature. Renters often experience opaque billing practices, hidden charges, or unexpected deductions for damages that are not verifiably documented. Because rental companies maintain control over databases, there is no cryptographic guarantee that rental histories or condition reports reflect the truth. Customers cannot independently validate the legitimacy of the records, creating mistrust and recurring disputes.

From the perspective of car owners or rental companies, the absence of automated and immutable systems makes enforcement inconsistent. Vehicles may be returned late or in poor condition, yet documentation errors and subjective evaluations complicate accountability. Operational overhead is high because of repetitive administrative tasks such as verifying documents, updating availability, manually processing payments, and reviewing rental logs.

At the systemic level, centralized platforms introduce major risks. Server downtime halts business operations. Data breaches expose customer information. Insider manipulation can modify records or payments. As the number of customers grows, maintaining accuracy and security becomes more complex.

The industry clearly needs a platform that automates rental logic, secures financial transactions, and ensures that no party — renter or owner — can alter historical data. Smart Chain addresses these needs through a decentralized, self-governing rental system

➢ EXISTING SYSTEM ANALYSIS

The traditional automobile rental ecosystem operates on centralized architectures where all data, processes, and user interactions are controlled by a single rental company or an authorized third-party service provider. Despite offering online booking interfaces, these systems retain manual workflows and centralized databases at their core. This structure introduces several operational risks, inefficiencies, and trust-related challenges.

Centralized Data Management

- ➢ All rental information—including user identities, booking histories, car availability, payment logs, and rental agreements—is stored in a centralized server controlled exclusively by the rental company.
- ➢ Because data is not cryptographically protected or distributed, administrators have full power to modify, delete, or fabricate rental entries.
- ➢ Any corruption, breach, or malicious alteration of the centralized database compromises the integrity of the entire rental platform.
- ➢ Customers have no ability to verify whether the information displayed to them reflects actual historical data or modified records.

Manual Verification and Approval Processes

- ➢ Rental workflows require human staff to verify driver credentials, approve booking requests, inspect vehicles, and update rental status in the system.
- ➢ These manual tasks lead to processing delays, especially during peak demand or when staff availability is limited.

- Human involvement increases the likelihood of errors, misjudgments, inconsistent reporting, and operational bottlenecks.
- Paper-based verification or physical documentation still plays a major role in many rental companies, further slowing down the process.

Opaque Pricing and Payment Handling

- Customers are frequently subjected to hidden fees, deposits, insurance surcharges, and penalties that are not clearly disclosed at the time of booking.
- Payment processing relies on centralized gateways (credit card systems, vendor-controlled billing software), increasing exposure to fraud and chargeback disputes.
- Refunds and security deposit returns are often delayed because they require manual approval and processing by rental company staff.

Lack of Immutable Record-Keeping

- Traditional systems do not provide immutable or tamper-proof logs for rental history, vehicle condition, mileage, or charges applied during returns.
- This lack of immutable records results in disputes between renters and companies, with customers having no cryptographically verified proof to challenge incorrect charges.
- Damage reports or extra fee assessments rely heavily on human judgment and cannot be independently verified later.

Single Point of Failure

- The entire system depends on the reliability of the rental company's central server.
- Server outages, cyberattacks, or database corruption can halt all rental activities across regions.
- If the central database is compromised, customer data—including payment details—can be leaked or manipulated.

Security Vulnerabilities

- Centralized systems store sensitive customer information such as identity documents, driver's license copies, and payment card details.
- These databases are prime targets for hackers, and breaches can expose thousands of users to identity theft or financial fraud.
- Data encryption, although present, cannot prevent insider manipulation or unauthorized modifications by system administrators.

Limited Transparency in Vehicle Status & Availability

- Availability updates depend on manual or semi-automated processes, making it difficult to ensure real-time accuracy.
- Double bookings, incorrect availability statuses, or unreported vehicle downtime frequently occur.
- Customers cannot verify whether a car shown as "available" is actually unused or being processed in another branch.

Inefficient Fleet Management

- Rental companies rely on internal tracking systems and staff reports to monitor vehicle usage, condition, and maintenance needs.
- Delayed updates or incomplete data lead to underutilized vehicles and inefficient fleet distribution.
- Managers often rely on estimates rather than real-time analytics.

Customer-Company Disputes

- Disagreements commonly arise over:
  - Mileage usage
  - Fuel levels
  - Minor damages
  - Returning time discrepancies
  - Additional costs applied after the rental period
- Because records are mutable and controlled solely by the rental company, customers often feel powerless to challenge discrepancies.

Lack of Global Accessibility and Interoperability

- Traditional platforms restrict users to local branches or regional databases, limiting mobility.
- Booking a car in another region or country often requires separate user accounts or new verification processes.
- Systems are not designed for universal interoperability with third-party services.

METHODOLOGY

The development of Smart Chain Auto Rentals followed a carefully structured methodology to ensure the system was robust, transparent, secure, and fully decentralized. The methodology spans planning, design, contract development, integration, testing, and deployment. Below is the refined version with expanded bullet points and light technical references.

1. Requirement Analysis and Domain Understanding

- The team began by analyzing shortcomings in traditional rental systems, identifying gaps such as centralization, lack of transparency, delayed confirmations, and manual data handling.
- User roles (car owner, renter) and functional needs (listing, renting, returning, revenue collection) were mapped to blockchain equivalents.
- The objective was clear: **design a system where rental processes rely on code, not human decisions**, ensuring trustless automation.

2. System Architecture Design

- A multi-layer architecture was conceptualized consisting of smart contracts, blockchain ledger, wallet interaction, and a lightweight UI.
- The architecture ensured that each function — such as rental updates or revenue collection — originated from a validated blockchain transaction rather than a server-controlled process.
- Data flow diagrams and rental lifecycle visuals were established to guide development from start to finish.

3. Smart Contract Planning

- Before writing any code, the structure of on-chain data was carefully designed.

- The team defined what information a car should store (name, price, owner, availability) and what a rental record should capture (renter, duration, amount paid, completion status).
- Rental rules were converted into logical conditions — for instance:
    - A car cannot be rented if already marked unavailable.
    - Payment must match the required rental cost.
    - Owners should only receive funds after rentals are completed.
- This step ensured that the smart contract acted as an **autonomous rental administrator**.

4. Smart Contract Development (Minimal Code)

- Solidity contracts were written to encode rental logic directly on the blockchain.
- The development emphasized clarity, security, and strict validation to prevent incorrect or malicious actions.
- Only basic and safe programming patterns were used, such as confirming availability before renting and ensuring revenue cannot be withdrawn twice.
- Code remained minimalistic to reduce gas usage and make debugging easier.

5. Hardhat Local Blockchain Setup

- Hardhat was configured to simulate a full Ethereum-like blockchain locally.
- It provided multiple test accounts and allowed the team to repeatedly deploy and test the contract without real costs.
- This environment ensured consistent testing conditions and enabled fast iteration cycles

6. Frontend Integration with Blockchain

- The UI was developed to interact directly with the blockchain using ethers.js.
- Every button — such as “List Car,” “Rent Car,” or “Return Car” — was linked to a corresponding contract function.
- No centralized server or backend database was used; the UI fetched all data directly from the smart contract, ensuring maximum transparency.
- MetaMask served as the identity and confirmation tool for all transactions.

7. User Interface & Dashboard Construction

- A simple dashboard was created to help users easily understand car availability, active rentals, and revenue.
- All displayed values were read from the blockchain in real time, ensuring accuracy and consistency.
- Clear labels and intuitive buttons allowed non-technical users to interact smoothly with decentralized technology.

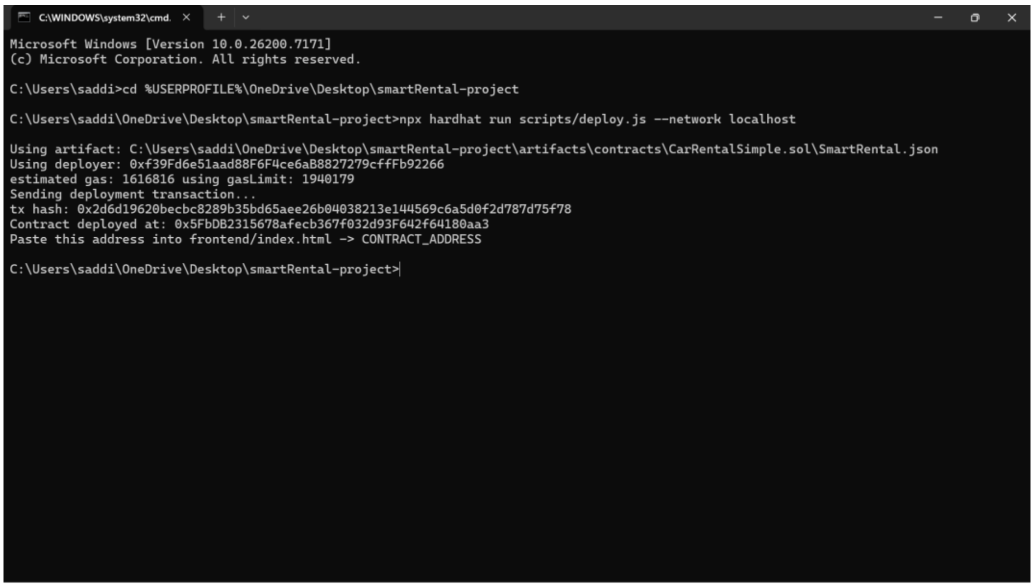
8. Testing and Scenario Simulation

- Rental transactions were tested under various scenarios:
    - Renting cars that are available
    - Attempting to rent unavailable cars

  - Returning a rented vehicle
  - Collecting revenue after rental completion
- Incorrect actions (e.g., insufficient payment, double rental attempts) were expected to fail, and they did — confirming the contract's correctness.
- Testing ensured that each step in the rental lifecycle behaved predictably and safely.

9. Deployment, Debugging, and Optimization

- The contract was deployed multiple times to ensure reliability.
- Debugging involved checking transaction logs, verifying contract state updates, and ensuring the UI correctly refreshed.
- The system was optimized to reduce unnecessary on-chain computations, improving overall performance.
- The final version integrated seamlessly with the UI, providing a stable decentralized rental system.



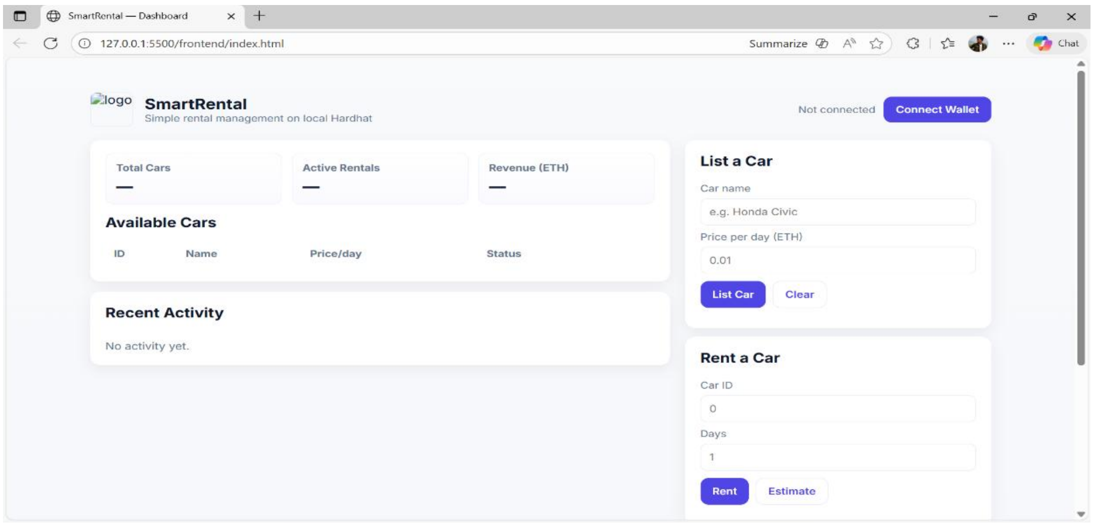
```
Microsoft Windows [Version 10.0.26200.7171]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saddi>cd %USERPROFILE%\OneDrive\Desktop\smartRental-project

C:\Users\saddi\OneDrive\Desktop\smartRental-project>npx hardhat run scripts/deploy.js --network localhost

Using artifact: C:\Users\saddi\OneDrive\Desktop\smartRental-project\artifacts\contracts\CarRentalSimple.sol\SmartRental.json
Using deployer: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
estimated gas: 1616816 using gasLimit: 1940179
Sending deployment transaction...
tx hash: 0x2d6d19620becbc8289b35bd65aee26b04038213e144569c6a5d0f2d787d75f78
Contract deployed at: 0x5FbDB2315678afecb367f032d93F642f64180aa3
Paste this address into frontend/index.html -> CONTRACT_ADDRESS

C:\Users\saddi\OneDrive\Desktop\smartRental-project>
```

10. Documentation and Final Review

- After ensuring system stability, detailed documentation was created explaining the architecture, development workflow, testing results, and user instructions.
- Screenshots, diagrams, and structured explanations were compiled to make the system understandable to both technical and non-technical stakeholders.
- This phase completed the methodology and ensured the system was presentation-ready.

➢ PROPOSED SYSTEM — SMART CHAIN

The Smart Chain Auto Rentals system proposes a complete re-engineering of the traditional car rental workflow by introducing blockchain automation, decentralized identity, and immutable rental histories. Instead of depending on centralized authorities or conventional record-keeping, Smart Chain uses smart contracts to enforce rental logic, store data, and secure payments. The proposed system brings the following innovations:



Blockchain-Backed Rental Records

- Each car listing, rental transaction, return confirmation, and payout operation is stored directly on the blockchain, ensuring that all actions are permanently recorded.
- This eliminates the possibility of data tampering, fraudulent entries, or retroactive modifications which are common in centralized systems.
- Immutable logs allow both renters and owners to rely on objective truth rather than trusting a company's internal database.

Smart Contract–Driven Automation

- All rental rules are embedded into smart contract code, which executes automatically and neutrally without human intervention.
- The contract verifies availability, calculates payment requirements, stores rental details, locks funds securely, updates car status, and releases revenue only when appropriate events occur.
- By automating these tasks, the system eliminates human bias, manual paperwork, and operational delays.

Decentralized Identity via MetaMask

- Users authenticate using blockchain wallet addresses rather than email/password credentials.
- This enhances security by eliminating the need for storing sensitive information on a centralized server.
- MetaMask signatures ensure that only the true account owner can initiate or approve transactions.

Real-Time Availability Synchronization

- Because availability is controlled on-chain, cars instantly transition from "available" to "rented" the moment a transaction is confirmed.
- No backend caching or staff intervention is needed — availability is always accurate and globally consistent.

Transparent and Auditable Rental Lifecycle

- Every action—listing, renting, returning, collecting—is associated with blockchain event logs.
- Renters and owners can monitor their transaction history with complete transparency, enabling effortless auditing and dispute resolution.

No Central Server or Database

- Unlike traditional systems that rely on company databases, Smart Chain stores everything on a decentralized ledger.
- This eliminates single points of failure, prevents hacking of user data, and ensures system integrity.

UI Integration with Blockchain Logic

- The user interface directly reads blockchain state through ethers.js, ensuring that displayed information reflects actual on-chain values.
- The front-end serves only as a window into blockchain data, not a storage engine.

The proposed system thus provides a decentralized, secure, transparent, and universally accessible rental platform unlike any centralized alternative.

➢ SYSTEM ARCHITECTURE

The Smart Chain Auto Rentals architecture is built as a multi-tier decentralized system where each layer performs a distinct, essential function. Together, these layers create a secure, scalable, autonomous rental ecosystem that operates without centralized control.

1. Blockchain Execution Layer

- The core foundation of the architecture is the blockchain network, responsible for executing, verifying, and recording smart contract transactions.
- Every action—car listing, rental initiation, return confirmation, and payout—is processed by the Ethereum Virtual Machine (EVM) in a deterministic manner.
- The blockchain enables distributed consensus ensuring that no single authority controls system data or logic, making the system tamper-proof and censorship resistant.

2. Smart Contract Logic Layer

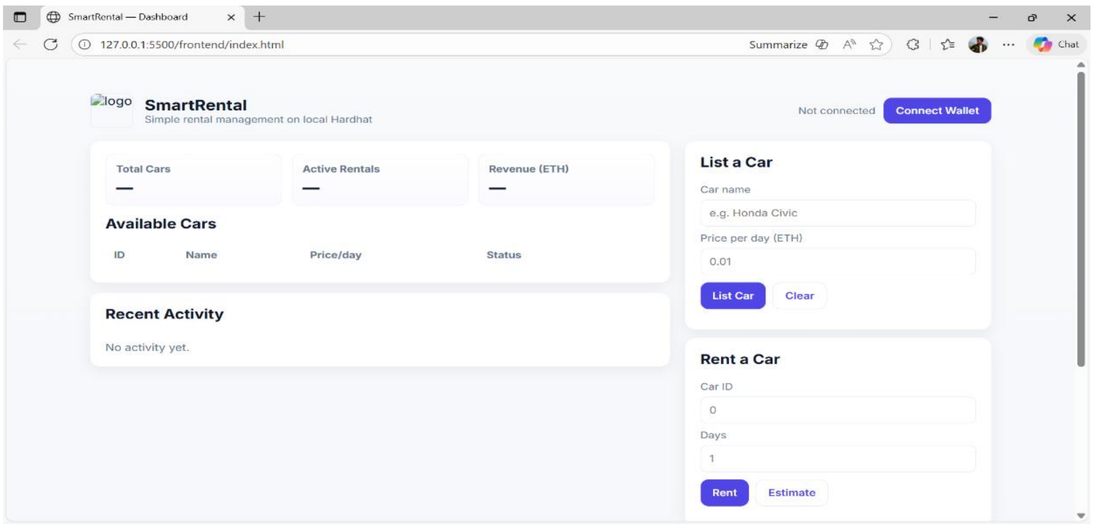
- This layer holds the Solidity-based contract that encodes all business rules of the rental ecosystem.
- It defines essential structures (Car, Rental) and state variables that track car availability, rental durations, renter addresses, and payment statuses.
- Functions in this layer govern how rentals occur, how funds are locked, how returns are processed, and how owners collect revenue.
- Validation rules prevent unauthorized actions, ensuring renters cannot bypass payment or rent unavailable cars.

3. Integration Layer (ethers.js Communication Bridge)

- This layer handles communication between the front-end and blockchain, allowing users to perform actions through the UI that translate directly into blockchain transactions.
- Ethers.js listens for on-chain events and updates the front-end in real time based on changes in contract state.
- It formats function calls, sends them to the blockchain via MetaMask, and retrieves updated data.

4. Presentation Layer (Frontend UI)

- The front-end provides a visual display of rental options, statistics, and transaction outcomes, serving as the user's main entry point into the platform.
- Information displayed—such as available cars, rented cars, total revenue—is derived directly from smart contract getter functions.
- The UI is intentionally simple and stateless, ensuring system transparency and reducing reliance on traditional servers.


SmartRental — Dashboard

127.0.0.1:5500/frontend/index.html

SmartRental
Simple rental management on local Hardhat

Not connected
Connect Wallet

Total Cars	Active Rentals	Revenue (ETH)
—	—	—

Available Cars

ID	Name	Price/day	Status

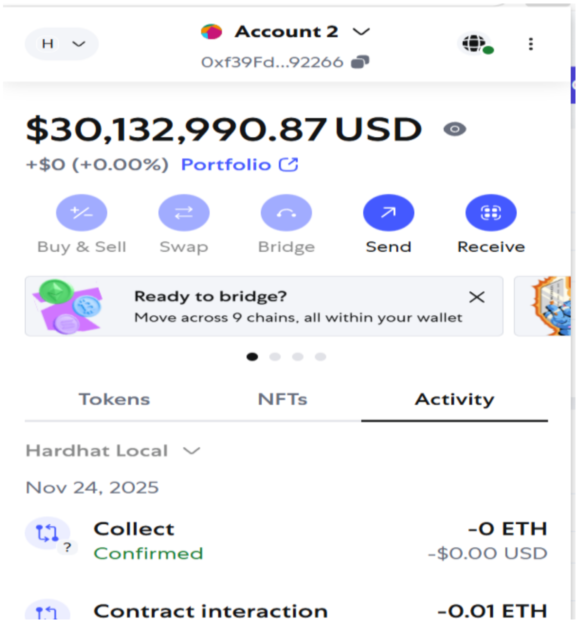
Recent Activity
No activity yet.

List a Car
Car name: e.g. Honda Civic
Price per day (ETH): 0.01
List Car | Clear

Rent a Car
Car ID: 0
Days: 1
Rent | Estimate

5. Wallet and Identity Layer (MetaMask)

- MetaMask acts as the decentralized identity layer by providing cryptographic authentication.
- Instead of creating accounts or storing user credentials, Smart Chain depends entirely on wallet addresses for identity and authorization.
- Every rental action is approved by the user through MetaMask's transaction confirmation interface.


Account 2
0xf39Fd...92266

$30,132,990.87 USD
+$0 (+0.00%) Portfolio

Ready to bridge?
Move across 9 chains, all within your wallet

Hardhat Local

Nov 24, 2025

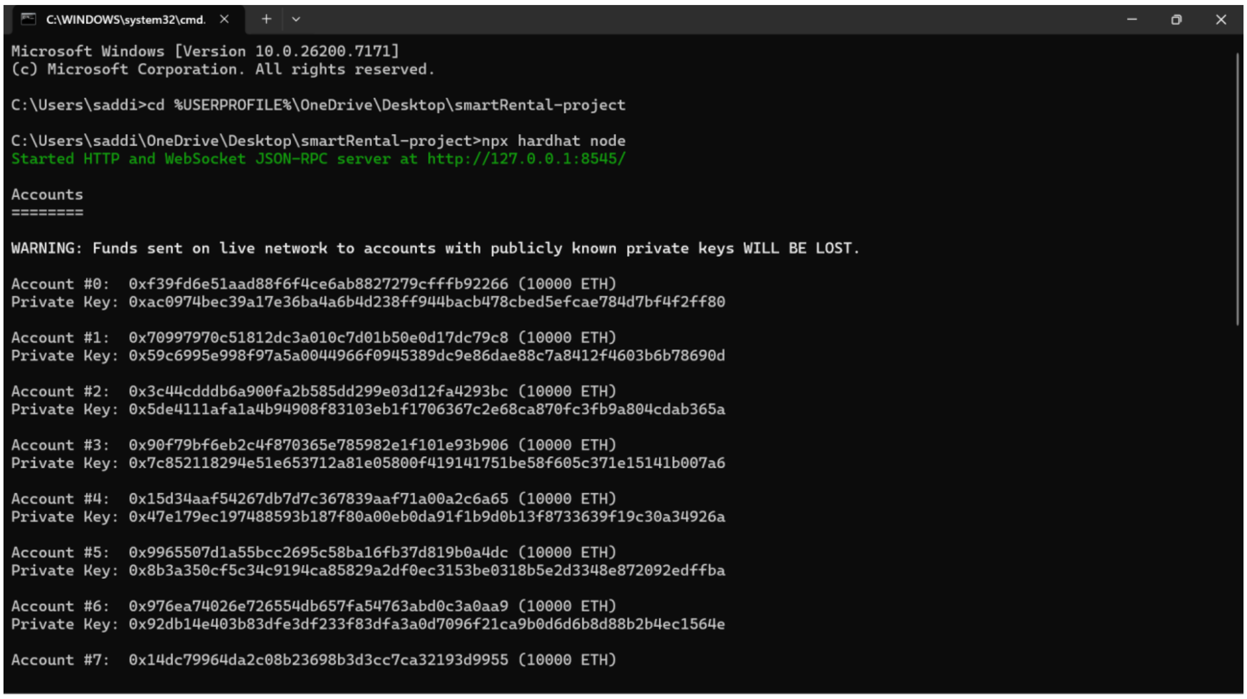
Collect — Confirmed — -0 ETH — -$0.00 USD

Contract interaction — -0.01 ETH

6. Deployment and Development Layer (Hardhat)

- Hardhat simulates the Ethereum blockchain locally for testing and deployment.
- It provides:
  - 20 pre-funded accounts with 10,000 ETH each

   - Complete debugging tools
   - Console-based transaction logs
   - Script-based contract deployment
- This layer ensures that Smart Chain behaves identically during development and eventual deployment to testnets or mainnets.



```
Microsoft Windows [Version 10.0.26200.7171]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saddi>cd %USERPROFILE%\OneDrive\Desktop\smartRental-project

C:\Users\saddi\OneDrive\Desktop\smartRental-project>npx hardhat node
Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/

Accounts
========

WARNING: Funds sent on live network to accounts with publicly known private keys WILL BE LOST.

Account #0:  0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266 (10000 ETH)
Private Key: 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80

Account #1:  0x70997970c51812dc3a010c7d01b50e0d17dc79c8 (10000 ETH)
Private Key: 0x59c6995e998f97a5a0044966f0945389dc9e86dae88c7a8412f4603b6b78690d

Account #2:  0x3c44cdddb6a900fa2b585dd299e03d12fa4293bc (10000 ETH)
Private Key: 0x5de4111afa1a4b94908f83103eb1f1706367c2e68ca870fc3fb9a804cdab365a

Account #3:  0x90f79bf6eb2c4f870365e785982e1f101e93b906 (10000 ETH)
Private Key: 0x7c852118294e51e653712a81e05800f419141751be58f605c371e15141b007a6

Account #4:  0x15d34aaf54267db7d7c367839aaf71a00a2c6a65 (10000 ETH)
Private Key: 0x47e179ec197488593b187f80a00eb0da91f1b9d0b13f8733639f19c30a34926a

Account #5:  0x9965507d1a55bcc2695c58ba16fb37d819b0a4dc (10000 ETH)
Private Key: 0x8b3a350cf5c34c9194ca85829a2df0ec3153be0318b5e2d3348e872092edffba

Account #6:  0x976ea74026e726554db657fa54763abd0c3a0aa9 (10000 ETH)
Private Key: 0x92db14e403b83dfe3df233f83dfa3a0d7096f21ca9b0d6d6b8d88b2b4ec1564e

Account #7:  0x14dc79964da2c08b23698b3d3cc7ca32193d9955 (10000 ETH)
```

7. Event Monitoring and Logging Layer

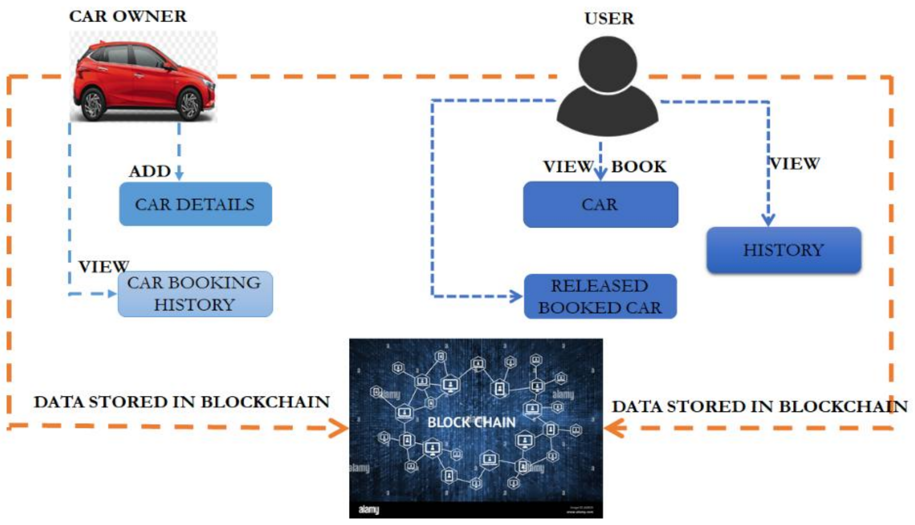
- Blockchain events emitted by the smart contract notify the UI of changes such as car listing updates, rental confirmations, and revenue withdrawals.
- This enables the system to reflect real-time updates without polling or centralized servers.

➢ SYSTEM REQUIREMENTS

Smart Chain is designed to be lightweight and efficient. Users only require a browser and MetaMask. Developers need Node.js, Hardhat, and a Solidity editor. The blockchain environment runs at 127.0.0.1:8545, providing deterministic test accounts. The system is intentionally modular to support upgrades like IoT tracking, AI recommendations, or IPFS storage.

➢ WORKFLOW DIAGRAM

The workflow of Smart Chain Auto Rentals reflects the full transition of the rental lifecycle from traditional, centralized operations into a decentralized, algorithm-driven blockchain model. Each step in the workflow is governed by immutable smart contract logic, ensuring fairness, transparency, and automation. The diagram (to be placed in this section) visually represents how the system directs interactions between the owner, renter, smart contract, and the blockchain.



1. Car Owner Initiates the Process

- The workflow begins when the car owner chooses to list their vehicle on the system.
- The owner provides essential details such as the car's name and daily rental price.
- Once submitted, these details become permanent entries on the blockchain, ensuring that vehicle information is not modified or hidden later.
- This step establishes the initial state for all subsequent rental actions.

2. Blockchain Stores Car Information Permanently

- After listing, the blockchain becomes the authoritative source of truth for the car's availability, ownership, price, and identifier.
- No centralized server or administrator manages these entries; instead, they are preserved immutably in the blockchain's distributed ledger.
- This prevents any manipulation or tampering of car data.

3. Renter Browses Available Vehicles

- Renters interact with the front-end interface to explore the catalog of cars retrieved directly from the smart contract.
- Availability, car names, rental prices, and overall fleet status are all fetched in real-time from blockchain getter functions.
- This guarantees accuracy: a car shown as "available" on the interface is verifiably available on-chain.
- The renter selects a car to initiate a booking.

4. Renter Initiates Rental Request via UI

- When a renter clicks the **Rent** button, the system prepares a blockchain transaction that captures the rental details.

- The UI does not process the request directly but instead signals MetaMask to initiate a transaction.
- This starts the formal rental lifecycle, shifting responsibility from the front-end to blockchain-level validation.

5. MetaMask Confirmation & Authentication

- MetaMask presents the transaction details to the renter for review and approval.
- This step ensures user authentication through cryptographic signatures rather than traditional passwords or centralized logins.
- If the user confirms, MetaMask signs the transaction and broadcasts it to the blockchain.
- This protects against unauthorized rentals since only the legitimate wallet owner can initiate a transaction.

6. Smart Contract Validates Rental Conditions

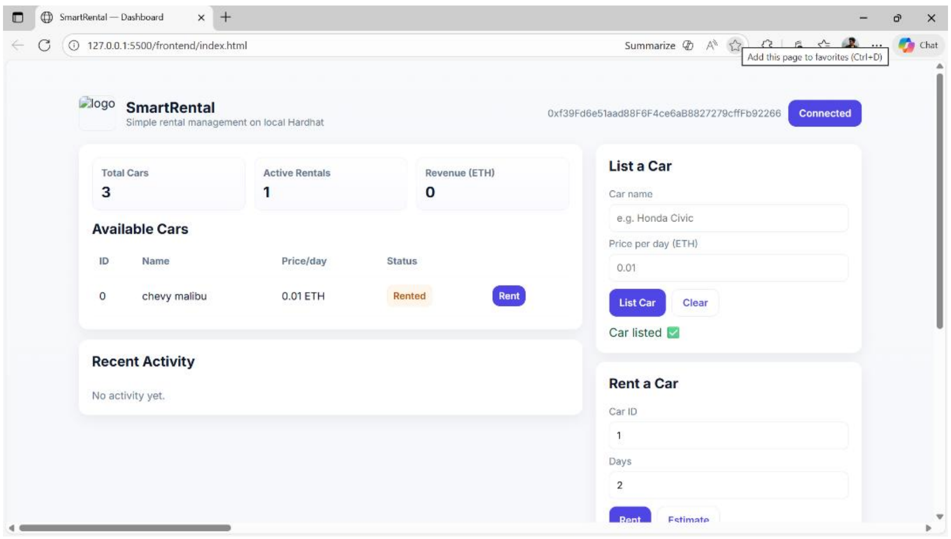
Once the transaction reaches the blockchain, the smart contract performs several checks before finalizing the rental:

- Confirms that the car is available.
- Verifies the correct payment amount based on the number of rental days.
- Ensures that the renter is not the owner of the car.
- Ensures no active rental exists for the same car.

If any condition fails, the transaction is rejected, preventing undesired or invalid operations.

7. Successful Rental → Car Status Updated

- If the rental is valid, the smart contract marks the car as “rented.”
- This update is immediately reflected in the blockchain state, making the car unavailable to others until it is returned.
- The UI listens for contract events and updates the display in real-time, providing instant visual feedback to the user.


SmartRental — Dashboard
127.0.0.1:5500/frontend/index.html
Summarize
Add this page to favorites (Ctrl+D)
Chat

logo SmartRental
Simple rental management on local Hardhat

0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266 Connected

Total Cars	Active Rentals	Revenue (ETH)
3	1	0

Available Cars

ID	Name	Price/day	Status	
0	chevy malibu	0.01 ETH	Rented	Rent

Recent Activity

No activity yet.

List a Car

Car name: e.g. Honda Civic

Price per day (ETH): 0.01

List Car Clear

Car listed ✅

Rent a Car

Car ID: 1

Days: 2

Rent Estimate

8. Payment Locked in Smart Contract Escrow

- The renter’s payment does not go to the owner directly but is held securely in escrow within the smart contract.
- This ensures that the owner receives payment only after the rental is completed.
- Escrow prevents disputes by ensuring neutral handling of funds.

9. Renter Returns the Car

- When the user finishes their rental, they initiate a **Return Car** action.
- This triggers another blockchain transaction that updates the car's status from “rented” back to “available.”
- The rental history is marked as completed, ensuring accurate record-keeping.

10. Owner Collects Revenue from Escrow

- After the rental is successfully completed, the car owner initiates a **Collect Payment** action.
- The smart contract releases the escrowed funds to the owner’s wallet.
- The transfer is automated and secure, ensuring no delays or human involvement.
- The car owner’s revenue dashboard updates automatically using blockchain reads.

11. Blockchain Records Become Immutable Rental History

- Every transaction—listing, renting, returning, payment—emits an event stored on the blockchain.
- These event logs form the rental’s permanent audit trail.
- Anyone can verify the timeline of actions, eliminating the possibility of disputes.

12. Continuous Cycle for Multiple Rentals

- After the previous rental is completed, the car becomes available again.
- The system supports unlimited future rental cycles with complete transparency.
- The process remains entirely decentralized and automated each time.

13. Complete Workflow Summary

To summarize, the decentralized workflow automates:

- Listing
- Discovering
- Renting
- Returning
- Payment settlement
- Auditing and verification

All steps are governed by smart contract logic rather than human authority.

EXPERIMENT RESULTS

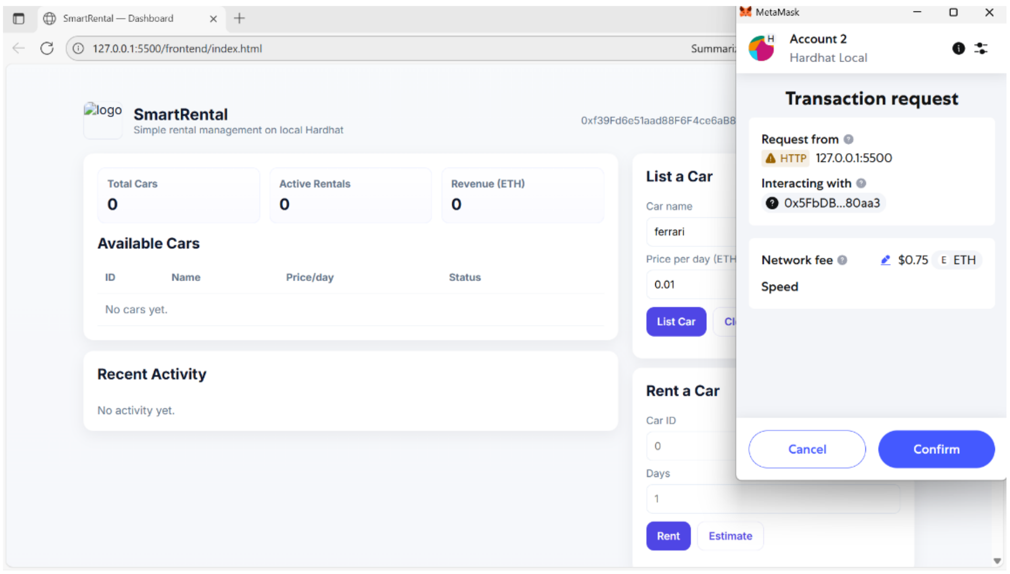
The experimental phase of Smart Chain Auto Rentals focused on validating the system's performance in real-life rental scenarios and measuring the behavior of the decentralized workflow under different operational conditions. Experiments were conducted to assess how the smart contract handles rental requests, manages payments, updates car availability, and synchronizes data with the user interface. Results strongly indicate that the system is robust, predictable, and suitable for decentralized rental management.

1. Car Listing Experiment

- Multiple cars were listed by different test accounts to confirm that the contract handled parallel listings reliably.
- Each newly listed car appeared immediately within the "Available Cars" section on the UI, demonstrating correct interaction between the contract's state and the frontend.
- Listing experiments confirmed that:
  - Car information was stored accurately
  - Availability status was initialized correctly
  - UI event listeners reacted instantly to contract events

2. Rental Initiation Experiment

- The renter selected a car and initiated the rental process using the UI, triggering a MetaMask transaction request.
- The blockchain confirmed the transaction and updated the car's availability to "rented," preventing other users from renting the same vehicle simultaneously.
- This experiment demonstrated strong consistency:
  - Blockchain validation prevented double rentals
  - Payment logic correctly matched the total rental cost
  - UI updated in real time via contract event emissions


SmartRental — Dashboard

127.0.0.1:5500/frontend/index.html

SmartRental
Simple rental management on local Hardhat

0xf39Fd6e51aad88F6F4ce6aB8

Total Cars 0
Active Rentals 0
Revenue (ETH) 0

Available Cars

ID	Name	Price/day	Status
No cars yet.			

Recent Activity
No activity yet.

List a Car
Car name: ferrari
Price per day (ETH: 0.01

Rent a Car
Car ID: 0
Days: 1

MetaMask
Account 2
Hardhat Local
Transaction request
Request from: HTTP 127.0.0.1:5500
Interacting with: 0x5FbDB...80aa3
Network fee: $0.75 ETH
Speed

3. Payment Escrow Experiment

- Once a rental was processed, payment was securely held in smart contract escrow until the rental was marked complete.
- Tests confirmed that owners could not withdraw funds early and renters could not reclaim payments mid-rental.
- Escrow behaved exactly as designed, ensuring fair financial flow for both parties.
- Transactions related to escrow deposits were visible in both MetaMask and the Hardhat console.

4. Rental Completion / Return Experiment

- During experimentation, cars were returned using the UI "Return Car" button, triggering another blockchain transaction.
- The contract updated car availability back to "available," enabling new rental cycles.
- The rental record was permanently stored in the blockchain's rental history structure, confirming immutable record keeping.
- This experiment validated the correctness of state transitions and event notifications.

5. Revenue Collection Experiment

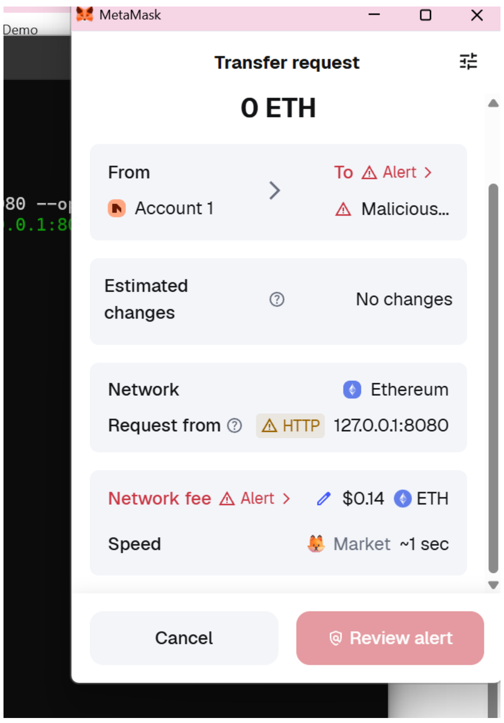
- Once the rental cycle was marked complete, owners used the "Collect Revenue" feature to withdraw the escrowed amount.
- The smart contract accurately calculated and transferred the exact amount paid by the renter.
- Funds appeared instantly in the owner's wallet, confirming secure financial handling.
- This demonstrated the proper functioning of the withdraw pattern and smart contract security safeguards.

6. Invalid Operation Experiments

A series of controlled invalid operation attempts were performed to measure how the system responds to incorrect inputs:

- Renting an already rented car correctly triggered a revert.
- Attempting to rent without enough Ether resulted in transaction rejection.
- Trying to return a car that was never rented failed as expected.
- Attempting to collect funds before rental completion also reverted.

These tests confirmed that Smart Chain is resistant to misuse and enforces strict validation rules


Transfer request

0 ETH

From: Account 1 → To ⚠ Alert: ⚠ Malicious...

Estimated changes: No changes

Network: Ethereum

Request from: ⚠ HTTP 127.0.0.1:8080

Network fee ⚠ Alert: $0.14 ETH

Speed: Market ~1 sec

7. Multi-Rental and Multi-User Simulation

- Multiple accounts were used to simulate realistic, simultaneous rental activity.
- Different users rented different cars without interfering with each other's rental states.
- This proved that the contract handles parallel workflows effectively and maintains independent state for each vehicle.

8. UI Stability and Real-Time Updates

- The system was tested under repeated interactions to ensure that UI elements always reflected on-chain data.
- After each rental, return, and payout action, the user interface refreshed automatically and consistently based on the smart contract event triggers.
- No caching errors, delays, or mismatches were observed.

9. Scalability Observations

- Although the system currently runs on a local Hardhat blockchain, experiments suggest that the approach can scale to public chains.
- Storage reads and writes remained consistent under repeated operations.
- The architecture supports expansion to larger rental fleets.

10. Summary of Experiment Findings

The experiments conclusively demonstrated that:

- Blockchain operations are stable and reliable
- Rental flows function precisely as designed
- Payment logic is secure and predictable
- Availability updates never desynchronize

- Event-driven UI updates occur immediately
- Validation logic prevents improper actions
- The system is ready for broader deployment

CONCLUSION

The Smart Chain Auto Rentals project demonstrates the powerful potential of blockchain technology in redefining traditional service-based industries. By removing the dependency on centralized authorities and manual processes, the system establishes a fully decentralized, transparent, and automated rental ecosystem. Throughout its development and testing, Smart Chain consistently proved that smart contracts are capable of managing the entire rental lifecycle—car listing, booking, return, and revenue distribution—without requiring human intervention or oversight. This not only reduces operational inefficiencies but also eliminates opportunities for fraud, disputes, or data manipulation, which are common in conventional rental environments.

The project shows that trust no longer needs to be placed in rental companies, staff members, or administrators; instead, trust is embedded directly into the blockchain through deterministic code execution and cryptographic verification. Every rental event becomes an immutable record, providing absolute clarity to both renters and owners and ensuring that neither party can alter historical data for their advantage. The integration of MetaMask for decentralized identity and transaction signing reinforces system security while providing a seamless user experience that does not rely on centralized login systems or user databases.

In addition to demonstrating superior transparency and security, Smart Chain illustrates how decentralized technology can dramatically improve user confidence. Renters interact with real-time, blockchain-verified information about car availability, rental status, and payment history, while car owners benefit from guaranteed, automated payment releases that are directly tied to the successful completion of a rental. The system's architecture offers a scalable foundation capable of handling large volumes of rental transactions, multiple users, and diverse vehicle fleets.

Furthermore, the project contributes meaningfully to the broader Web3 ecosystem by showcasing how decentralized applications can solve practical real-world problems. It proves that blockchain is not merely theoretical or confined to financial applications; rather, it is a transformative technology capable of governing complex service workflows with exceptional accuracy and reliability. The implementation of Smart Chain serves as a blueprint for future decentralized mobility solutions and demonstrates how smart contract–based automation can replace entire layers of traditional administrative infrastructure.

Overall, Smart Chain Auto Rentals achieves its goal of creating a transparent, automated, and tamper-proof rental platform that surpasses the limitations of existing centralized systems. Through its decentralized architecture, immutable ledger, and secure, user-controlled identity mechanisms, the project represents a major step forward in the evolution of rental services and highlights the significant advantages of blockchain-driven automation.

CONTRIBUTION OF TEAM MEMBERS

Ajaymishra Saddi

- Developed the main Solidity smart contract and implemented all rental logic.
- Managed Hardhat deployment, blockchain setup, and backend integration.
- Ensured contract security, validations, and correct state transitions.

Srilekha Kashireddy

- Built the frontend interface (dashboard, car views, user actions).
- Integrated MetaMask and ethers.js to connect UI with the blockchain.
- Ensured smooth user experience and accurate real-time updates.

Mahesh Vamsi Kode

- Performed testing of all rental scenarios and validated system behavior.
- Identified bugs, checked blockchain event logs, and ensured UI correctness.
- Prepared documentation, screenshots, and final project report formatting.

Team rotation Note

- Throughout the project, all three members rotated roles when needed, assisting each other in coding, testing, debugging, UI adjustments, and documentation to ensure smooth completion of the Smart Chain project.

REFERENCES

- Rented Car on Blockchain — *Decentralized Car Rental Concept Study*. https://www.researchgate.net/publication/351462937_Decentralized_Car_Rental_System_using_Blockchain
- *Smart Rental Systems using Blockchain Technology* (Academic Study). https://ieeexplore.ieee.org/document/9342908
- Wang, S., Ouyang, L., Yuan, Y., Ni, X., Han, X., & Wang, F. (2019). *Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends*. IEEE Transactions on Systems, Man, and Cybernetics.
- IBM Blockchain — *Blockchain Use Cases in Mobility & Rentals*. https://www.ibm.com/topics/blockchain-use-cases
- *Building a Decentralized Application (DApp)* — Dapp University https://www.youtube.com/watch?v=U9DqE3RnhfA

APPENDIX

https://github.com/ajaysmishra1819/smartRental-project