CSE1006 Blockchain & Cryptocurrency Technology

Car Fuel Mileage Management System

Theory DA/Project

Sudeep Roy Kurian
20BDS0113 - A2 + TA2

Prof. Dr. K.R. Jothi
SCOPE - Associate Professor Sr.

***Abstract*—A car fuel mileage management system using blockchain is a decentralized application that allows users to record and manage their car's fuel mileage data on the blockchain. It provides an immutable and transparent ledger that ensures the accuracy and integrity of the data. Essential for a corruption-free and trustable market based on car conditions managed by the blockchain. In this project, a car fuel mileage management system has been implemented using smart contracts on an Ethereum blockchain.**

Keywords—Ethereum, Smart Contracts, Fuel Mileage, Blockchain.

I. Introduction

Blockchain technology offers the potential to store data in a traceable, secure, and transparent manner. The whole system works on a de-centralized P2P network of nodes working to receive, validate and put transactions into blocks which are finally added to a blockchain. Their incentives lie in the block reward and transaction fees given to the node which successfully mines the newly added block. Such storage systems work well for applications intending to store data in an immutable manner for future reference.

One such application is a car fuel mileage management system which stores car mileage details for later reference, for example, to evaluate current car condition and value. Car mileage can refer to two things - (1) Total distance traveled over a given period, or (2) Fuel efficiency or distance traveled per unit of fuel. Usually calculated in Km/L or mi/gal. This mileage is used to estimate the wear and tear on a car, as well as determine its resale value. Lower a car's mileage, the better its condition meaning better resale value. Presently used technology uses analog or digital meters, fuel tracking apps, etc to show current car mileage. Although reliable and easy to use and install, they can easily be modified, tampered with, or reset to increase or decrease a car's condition on paper.

This paper proposes a novel car fuel mileage management system using blockchain technology to provide a secure, reliable, and easy-to-use application to create, modify, and delete mileage records in an immutable ledger. It ensures users make informed decisions regarding the car's fuel consumption, maintenance, and resale value. The blockchain platform used to implement this application is Ethereum.

Ethereum is a blockchain-based platform that enables the creation of smart contracts and decentralized applications (DApps). It was introduced in 2015 and is currently the second-largest cryptocurrency by market value after Bitcoin.

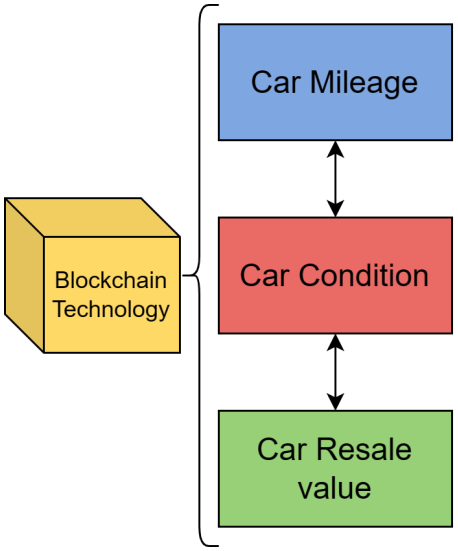


Fig 1. Car Mileage and Resale Value Relation

This project uses smart contracts which are self-executing digital contracts that automatically follow some pre-defined set of rules and conditions agreed upon by two or more parties. They allow for secure and transparent transactions without any intermediary third parties, reducing fraud and errors.

Ethereum smart contracts are coded in a programming language called Solidity and can be tested using online VMs like Remix or private VMs like Ganache. These smart contracts can be integrated with a front-end using React JS and libraries like Web3, used to build decentralized applications and web pages for the internet.

The following sections provide design, implementation, and results for smart contract implementation and front end.

II. Literature Survey

Title/Author/Year	Summary	Novelty	Limitations/Future Work
Trusted systems of records based on Blockchain technology - a prototype for mileage storing in the automotive industry Katarina P., Moritz B., Jan-Paul B., Carolin S. January 2020	This paper offers a reliable system for storing mileage records in a blockchain and using IoT to capture and send information to the system. It gives an overview of all parts of the implementation from the smart contracts to the hardware and software side of the IoT sensors. It also uses a decentralized app for user interaction.	No restrictions in reading, writing & editing mileage histories. Tamper-proof data ledger with high user interaction using a decentralized app Automated processes of soring all relevant information from car sensors like timestamps, speed, etc. Maintains user anonymity	*Limitations in the system:* - Limited Ethereum throughput. - Limited scalability of the system - Unpredictable transaction costs - Varied latency *Future Works:* - Improve scalability - Harness better vehicle data from ECUs
PetroBlock: A Blockchain-Based Payment Mechanism for Fueling Smart Vehicles Faisal J., Omar C., Harun J., Anis K., Abdelouahid D., Mohamed A.F. 2021	This paper provides an overview of a decentralized system for users to buy and sell fuel. It also evaluates the usability and efficiency of the system and provides a basic implementation with various diagrams. It also uses IoT sensors for data collection in all the stages of the system.	Tamper-proof ledger to record transactions for assets and participants with create, read, update, and delete operations. The Business model works closely with a smart contract implementation to create an efficient business plan	*Limitations in the system:* - Latency in block processing - Varied transaction costs *Future Works:* - Test the interoperability of the blockchain framework with other IoT-blockchain platforms - Improve consensus algorithm for fast querying.
Towards blockchain-IoT based shared mobility: Car-sharing and leasing as a case study Sophia A., Sophia N., Somnath M., Raghava R.M. January 2022	This paper provides an implementation of a high-level architecture integrating blockchain and IoT systems. It implements this architecture in a car-sharing and leasing service and provides a case study on the same.	The novelty of this system involves combining a system with security, authenticity, traceability, reliability, interoperability, and scalability. Uses immutable blockchains with GDPR compliance to building a decentralized app powered by smart contracts.	*Limitations in the system:* - Integrating IoT with blockchain in a scalable manner because of the huge amount of data transfer. *Future Works:* - Improving authenticity of accessing IoT devices - Scalability improvements - Analysing leasing and insurance processes
An Efficient Distributed Framework Model for Automotive Industry in Smart City Using Ethereum Blockchain Leya M.S., Sreeraj R. October 2020	This paper proposes a framework for a blockchain-based automotive industry in a smart city. It addresses common issues and includes a validation algorithm proof-of-authority. The model is simulated on a private Ethereum blockchain platform for performance evaluation.	Novel architecture containing manufacturers, dealers, regulators, easers, and end users along with maintenance and recycling. Also provides a new validation algorithm called proof-of-authority.	Further improvements on the architecture and identifying issues in the system can be included in future works.

TABLE I. Literature Survey

III. Design

This fuel mileage management system uses smart contracts to define the rules and logic for adding, updating, and deleting mileage records. Each car is represented by a unique address on the blockchain, and multiple cars can be managed by a single user.When a user adds a new mileage record, the system calculates the gas mileage and gasoline cost based on the fuel levels and distances traveled.The system provides various features such as the ability to view all mileage records for a specific car by its index, update existing records, and delete records that are no longer needed. Users can also view the gas mileage and number of cars and records on the blockchain ledger.

Since the system is built on a blockchain, it provides several advantages such as transparency, immutability, and security. The data is stored on a decentralized network of nodes, ensuring that there is no single point of failure or data tampering.

This smart contract runs on an Ethereum VM which is used to test and evaluate efficiency before deploying it on the actual Ethereum blockchain. For improving user interactive ability, a front-end website is designed using React JS and Web3 library.

- The input structure includes car details, current fuel levels, last refuel time, transaction timestamp, total mileage, etc.
- The output structure includes car details, fuel mileage or efficiency, and in future implementations time-varied graphs and estimated car value.

The target users include:

- **Car Owners** - Each having a private key for their car and a public identifier used by other users to query details and mileage.
- **Manufacturers/Operators/ Dealers/ Insurance** - Initialize smart contracts and accounts comprising a public and private key for each car.

A public blockchain is suitable for a fuel mileage management system, where known and trusted users plus unknown users are free to join and access the blockchain. This helps in creating secondary applications, like marketplaces, maintenance centers, etc.

A consensus algorithm is used by Ethereum to validate transactions sent to nodes in the blockchain to maintain trust in the system. Car data can be extracted from IoT devices and other sensors installed in a car.

The front end is a decentralized application accessible by all via a web-application user interface like a browser.

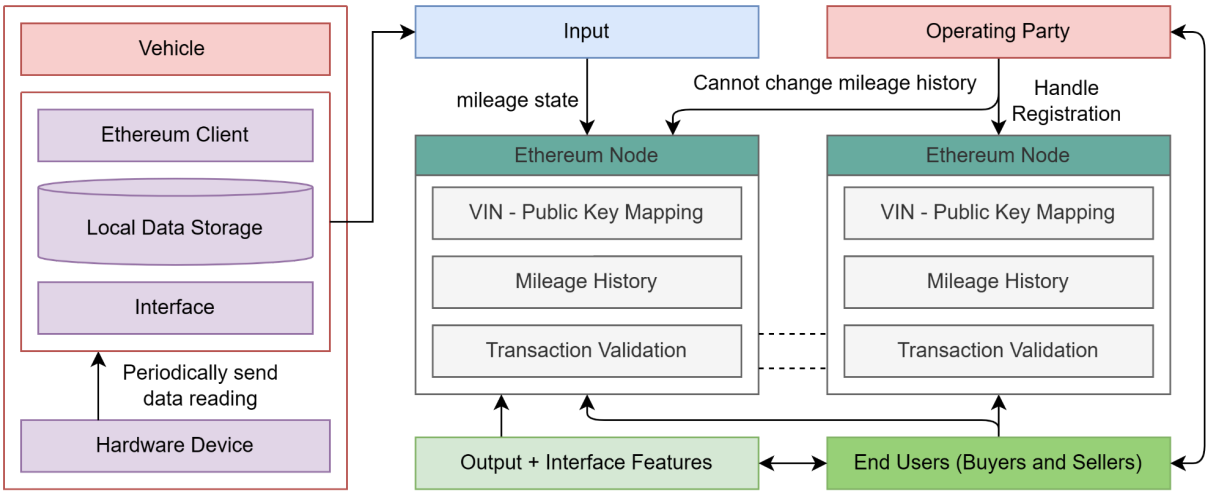


Fig 2. Data Processing from Car to End User

IV. Implementation

The fuel mileage management system proposed and implemented consists of two parts - (1) Smart Contract Implementation, and (2) Front-End Implementation.

(1) Smart Contract Implementation

The implementation involves the following steps:

1. Determine the purpose of the contract
2. Determine all necessary aspects of the contract
3. Choose a blockchain platform
4. Write the code
5. Compile and deploy the contract
6. Monitor and improve the contract

The code includes the ability to calculate gas mileage and gasoline cost. The ***MileageRecord*** struct now has an additional field for the distance traveled, which is used to calculate the gas mileage.

The ***updateMileageRecord*** and ***deleteMileageRecord*** functions allow users to modify or remove existing mileage records by index.

The ***setGasPrice*** function allows users to set the current gas price, which is stored in a mapping. The ***getGasPrice*** allows the users to get the current gas price.

The ***getMileageRecord*** and ***getMileageRecordCount*** allow getting a particular mileage record for the address or the number of records for the address.

The functions ***calculateGasMileage*** and ***calculateGasCost*** are used to return the gas mileage (fuel efficiency) and total cost of fuel in the record.

The smart contract is coded in Remix IDE, in the programming language called Solidity version 0.8.0. The records are defined in a struct data type which can be further expanded in future implementations:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
```

```
	struct MileageRecord {
		uint256 timestamp;
		uint256 fuelLevel;
		uint256 distanceTraveled;
	}
```

A mapping from addresses to the mileage records, and that of addresses to gas prices are defined:

```
mapping(address => MileageRecord[]) public mileageRecords;

mapping(address => uint256) public gasPrices;
```

(2) Front-End Implementation

The implementation for the front-end React application involves:

1. Creating a React project
2. Initializing truffle environment
3. Creating a ganache workspace
4. Writing the smart contract
5. Setting up the migration folder
6. Modifying the truffle config to connect to ganache
7. Add the contract to the React src folder
8. Designing and integrating the React front-end page to the ganache backend with the smart contract.

Ganache is a personal private blockchain for Ethereum development, used for testing and development purposes. It allows developers to create and test new contracts, as well as decentralized applications.

Truffle s a framework used to simplify the development, testing, and deployment of Ethereum smart contracts. it includes its own CLI, Contracts & Migrations folders with config files.

React JS is a popular JavaScript library used to build modern UIs and web applications. Designed to be modular and component-based, improving management and maintenance of large-scale applications.

Web3 JS is a JavaScript library that provides developers with a way to interact with the Ethereum blockchain and build decentralized applications. It is also the official Ethereum JavaScript API and is used to interact with Ethereum nodes, send and receive transactions and access smart contracts.

Smart Contract Code

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract FuelMileageManagementSystem {
    struct MileageRecord {
        uint256 timestamp;
        uint256 fuelLevel;
        uint256 distanceTraveled;
    }
    mapping(address => MileageRecord[]) public mileageRecords;
    mapping(address => uint256) public gasPrices;
    function addMileageRecord(uint256 _fuelLevel, uint256 _distanceTraveled) public {
        MileageRecord memory newRecord = MileageRecord({
            timestamp: block.timestamp,
            fuelLevel: _fuelLevel,
            distanceTraveled: _distanceTraveled
        });
        mileageRecords[msg.sender].push(newRecord);
    }
    function updateMileageRecord(uint256 _index, uint256 _fuelLevel, uint256 _distanceTraveled) public {
        require(_index < mileageRecords[msg.sender].length, "Invalid index");
        MileageRecord storage record = mileageRecords[msg.sender][_index];
        record.fuelLevel = _fuelLevel;
        record.distanceTraveled = _distanceTraveled;
    }
    function deleteMileageRecord(uint256 _index) public {
        require(_index < mileageRecords[msg.sender].length, "Invalid index");
        uint256 lastIndex = mileageRecords[msg.sender].length - 1;
        if (_index != lastIndex) {
            mileageRecords[msg.sender][_index] = mileageRecords[msg.sender][lastIndex];
        }
        mileageRecords[msg.sender].pop();
    }
     function setGasPrice(uint256 _price) public {
        gasPrices[msg.sender] = _price;
    }
    function getGasPrice() public view returns (uint256) {
        return gasPrices[msg.sender];
    }
    function getMileageRecordCount() public view returns (uint256) {
        return mileageRecords[msg.sender].length;
    }
    function getMileageRecord(uint256 _index) public view returns (uint256, uint256, uint256) {
        require(_index < mileageRecords[msg.sender].length, "Invalid index");
        MileageRecord memory record = mileageRecords[msg.sender][_index];
        return (record.timestamp, record.fuelLevel, record.distanceTraveled);
    }
    function calculateGasMileage(uint256 _index) public view returns (uint256) {
        require(_index < mileageRecords[msg.sender].length, "Invalid index");
        MileageRecord memory record = mileageRecords[msg.sender][_index];
        return (record.fuelLevel * 100) / record.distanceTraveled;
    }
    function calculateGasCost(uint256 _index) public view returns (uint256) {
        require(_index < mileageRecords[msg.sender].length, "Invalid index");
        MileageRecord memory record = mileageRecords[msg.sender][_index];
        uint256 gasPrice = gasPrices[msg.sender];
        return (record.fuelLevel * gasPrice) / 100;
    }
}
```

Front-End Code

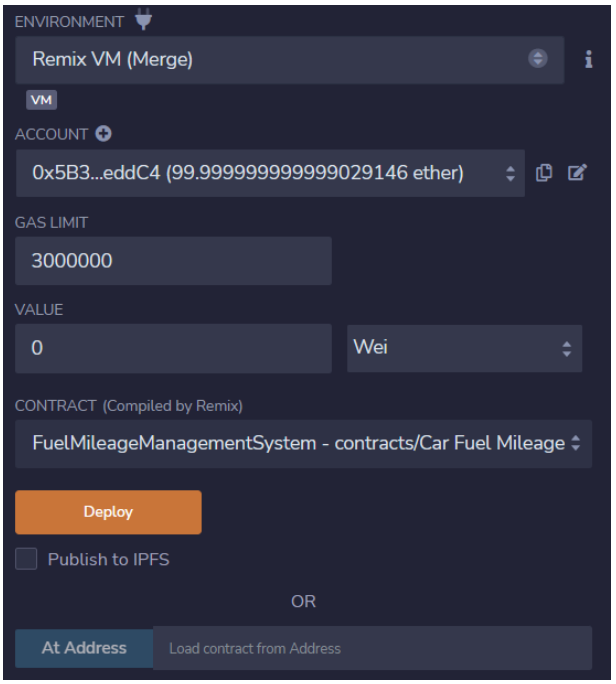
```
const handleAddRecord = async () => {
    try {
      await contract.methods.addMileageRecord(fuel, distance).send({ from: account, gas: '1000000' });
      const newCount = await contract.methods.getMileageRecordCount().call({ from: account, gas: '1000000' });
      const newRecord = await contract.methods.getMileageRecord(0).call({ from: account, gas: '1000000' });
      setCount(newCount);
      const {0: strValue, 1: fuel1, 2: dist1} = newRecord;
      setRecord1(fuel1);
      setRecord2(dist1);
    } catch (error) {console.log(error);}
  }
const handleUpdateRecord = async () => {
    try {
      await contract.methods.updateMileageRecord(index, fuel, distance).send({ from: account, gas: '1000000' });
      const newCount1 = await contract.methods.getMileageRecordCount().call({ from: account, gas: '1000000' });
      const newRecord1 = await contract.methods.getMileageRecord(index).call({ from: account, gas: '1000000' });
      setCount(newCount1);
      const {0: strValue, 1: fuel1, 2: dist1} = newRecord1;
      setRecord1(fuel1);
      setRecord2(dist1);
    } catch (error) {console.log(error);}
}
const handleDeleteRecord = async () => {
    try {
      await contract.methods.deleteMileageRecord(index).send({ from: account, gas: '1000000' });
      const newCount2 = await contract.methods.getMileageRecordCount().call({ from: account, gas: '1000000' });
      setCount(newCount2);
    } catch (error) { console.log(error);}
}
const handleGetRecord = async () => {
    try {
      const newCount1 = await contract.methods.getMileageRecordCount().call({ from: account, gas: '1000000' });
      const newRecord1 = await contract.methods.getMileageRecord(index).call({ from: account, gas: '1000000' });
      setCount(newCount1);
      const {0: strValue, 1: fuel1, 2: dist1} = newRecord1;
      setRecord1(fuel1);
      setRecord2(dist1);
    } catch (error) {console.log(error);}
}
const handleGetGasCost = async () => {
    try {
      const newRecord1 = await contract.methods.calculateGasCost(index).call({ from: account, gas: '1000000' });
      setGasCost(newRecord1);
    } catch (error) {console.log(error);}
}
const handleGetMileage = async () => {
    try {
      const newCount1 = await contract.methods.calculateGasMileage().call({ from: account, gas: '1000000' });
      setMileage(newCount1);
    } catch (error) {console.log(error);}
}
const handleSetGasCost = async () => {
    try {
      await contract.methods.setGasPrice(gasCost1).send({ from: account, gas: '1000000' });
    } catch (error) {console.log(error);}
}
```

V. Results

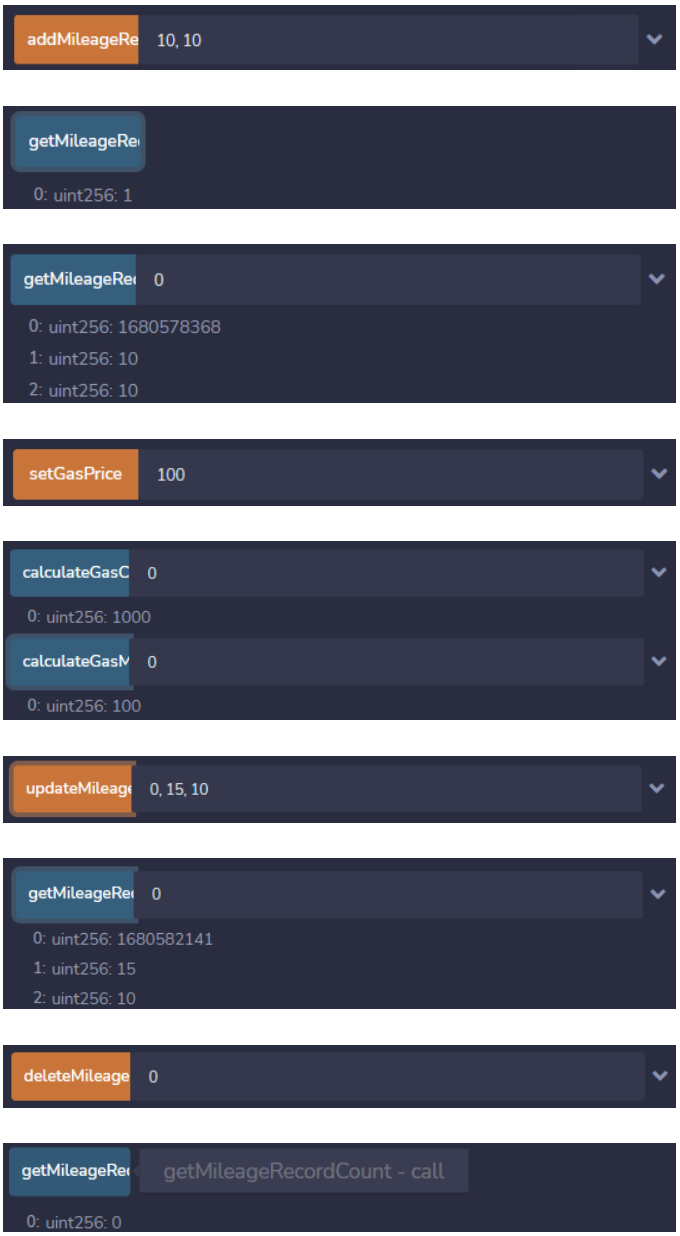
(1) Smart Contract

The contract is compiled and deployed using the Remix VM and one of the accounts available in the VM.

Each transaction requires transaction costs taken from the deployer.

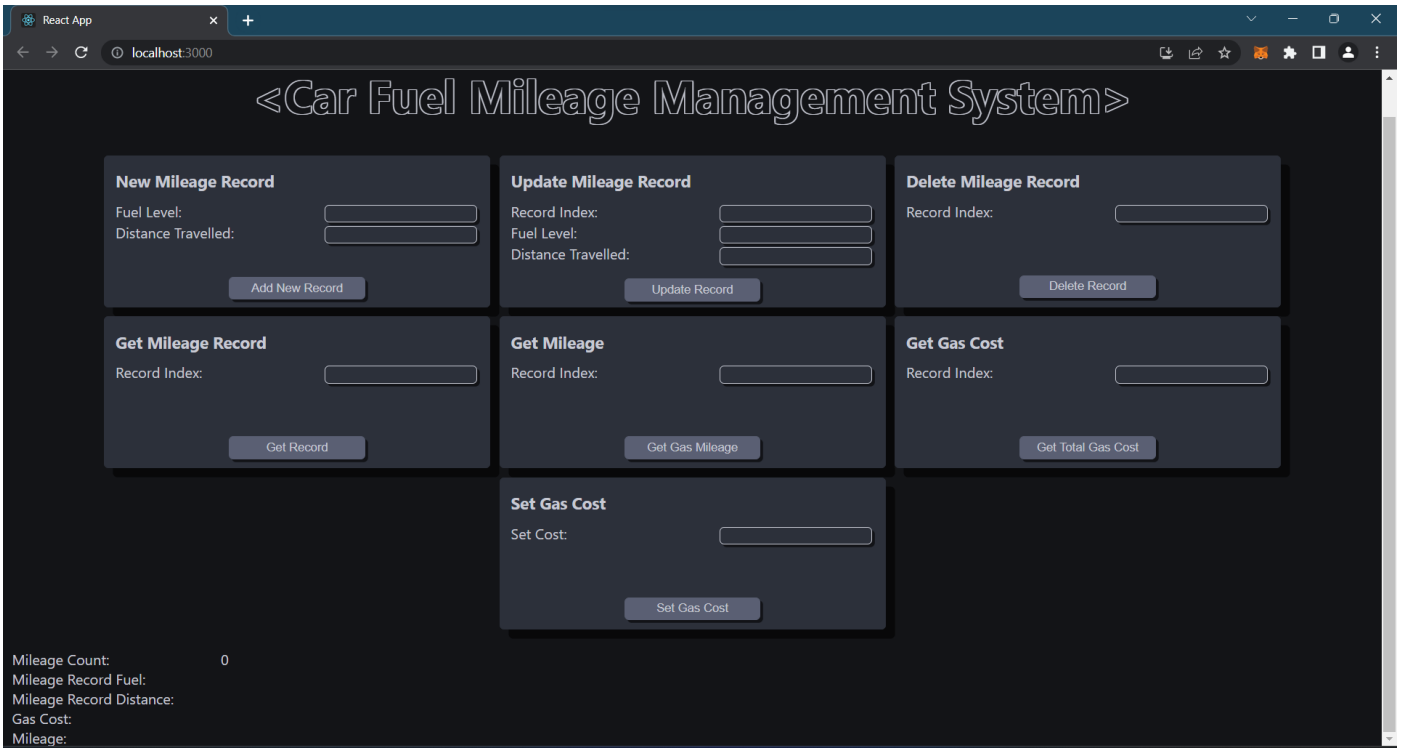


All the functions implemented in the smart contract:



Records as queried by users using indexes, multiple records can correlate to multiple cars or the same car at different times.

(2) Front-End Application



VI. Conclusion

The implementation of a car fuel mileage management system using blockchain technology can greatly improve the accuracy and transparency of tracking fuel consumption and mileage records for vehicles. The decentralized and immutable nature of blockchain provides an ideal solution for storing and sharing data in a secure and trustworthy manner.

By using a blockchain-based system, users can have complete control over their data and be confident that their records cannot be tampered with. Additionally, the use of smart contracts can automate certain processes, such as recording mileage data and triggering alerts when maintenance is needed or when fuel efficiency decreases.

The implementation of a car fuel mileage management system using blockchain has the potential to revolutionize the way we track and manage vehicle fuel consumption and mileage records. By improving accuracy and transparency, this system can lead to better fuel efficiency, reduced maintenance costs, and a more sustainable transportation system overall.

Acknowledgment

I would like to express my gratitude and appreciation to my supervisor **Prof. Dr. K.R. Jothi**, whose help, encouraging suggestions, and teachings helped me along the fabrication process of this project.

References

[1] Preikschat, K., Böhmecke‐Schwafert, M., Buchwald, J., & Stickel, C. (2020). Trusted systems of records based on Blockchain technology ‐ a prototype for mileage storing in the automotive industry. In Concurrency and Computation: Practice and Experience (Vol. 33, Issue 1). Wiley. https://doi.org/10.1002/cpe.5630

[2] Jamil, F., Cheikhrouhou, O., Jamil, H., Koubaa, A., Derhab, A., & Ferrag, M. A. (2021). PetroBlock: A Blockchain-Based Payment Mechanism for Fueling Smart Vehicles. In Applied Sciences (Vol. 11, Issue 7, p. 3055). MDPI AG. https://doi.org/10.3390/app11073055

[3] Auer, S., Nagler, S., Mazumdar, S., & Mukkamala, R. R. (2022). Towards blockchain-IoT based shared mobility: Car-sharing and leasing as a case study. In Journal of Network and Computer Applications (Vol. 200, p. 103316). Elsevier BV. https://doi.org/10.1016/j.jnca.2021.103316

[4] Leya M.S., Sreeraj R. (2020). An Efficient Distributed Framework Model for Automotive Industry in SmartCity Using Ethereum Blockchain. http://iccset2020.royalcet.ac.in/iccsetpapers/ICCSET_CSE098.pdf