

# **B.Tech Project Report**

## **IT-414**

**on**

## **Vehicle Rental System using Blockchain**

**BY**

**ANKIT SINGH (11510586)**  
**ARCHIT AGARWAL (11510618)**  
**SANIL CHUGH (11510574)**

**Under the Supervision of**  
**Dr. A.K. Singh**



**DEPARTMENT OF COMPUTER ENGINEERING**  
**NATIONAL INSTITUTE OF TECHNOLOGY**  
**KURUKSHETRA – 136119, HARYANA (INDIA)**  
**December, 2018 – April, 2019**



## CERTIFICATE

We hereby certify that the work which is being presented in this B.Tech. Minor Project (IT-414) report entitled “**Vehicle Rental System using Blockchain**”, in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Computer Engineering** is an authentic record of my own work carried out during a period from December 2018 to April 2019 under the supervision of Dr. A.K Singh, Professor, Computer Engineering Department.

The matter presented in this project report has not been submitted for the award of any other degree elsewhere.

*Signature of Candidate*

**ANKIT SINGH (11510586)**

**ARCHIT AGARWAL (11510618)**

**SANIL CHUGH (11510574)**

This is to certify that the above statement made by the candidates is correct to the best of our knowledge.

Date:

*Signature of Supervisor*

**Dr. A.K Singh**  
**Professor**

## **LIST OF FIGURES**

1. Snapshot of Metamask browser extension
2. Setup between network and client
3. Context diagram or Level - 0 data flow diagram
4. Level - 1 data flow diagram
5. Level - 2 data flow diagram
6. Sending transaction through Metamask
7. Home page of the DApp
8. Vehicle registration page
9. Vehicle description and pay rent page
10. Vehicle details after it is booked
11. Vehicle details edit page accessible to manager when vehicle is available
12. Deployed contract details on Etherscan

## TABLE OF CONTENTS

Section No.	TITLE	Page no.
	ABSTRACT	
I	INTRODUCTION	1
II	MOTIVATION	2
III	LITERATURE SURVEY	3
	III.1 Web3.js	3
	III.2 React	3
	III.3 Solidity	3
	III.4 Metamask	4
	III.5 Infura	5
	III.6 Blockchain	5
	III.7 Ethereum	5
IV	SMART CONTRACT PART	6
V	THE WEB APPLICATION PART	8
VI	DATA FLOW DIAGRAM	11
	VI.1 Level 0 DFD	11
	VI.2 Level 1 DFD	11
	VI.3 Level 2 DFD	12
VII	RESULTS	13
VIII	CONCLUSION	17
	REFERENCES	18
APPENDIX:		
A	COMPLETE CONTRIBUTARY SOURCE CODE	19

## **ABSTRACT**

Motivated by the recent hike in interest and hype, though justified, around blockchains, we explore whether they make a good fit for Internet of Things (IoT) sector. Blockchains enables us to have a distributed network, i.e., a peer to peer network. It works by allowing non-trusting members to trust each other without a trusted intermediary, in a verifiable manner. In our project, we try to use blockchain technology in combination with the IoT devices. The main limitations of the IoT platform are its security and integrity problems. The blockchain technology is used to make the communication of IoT devices more secure.

To depict this concept, vehicle rental system model is implemented. In this model, we have made a decentralized platform to the users that either want to rent their vehicle or the users that want to take the vehicle on rent. A smart contract handles all the transactions, thus eliminating the need of a third party, providing the functionalities of an escrow.

This project also comprises of blockchain environment which is a new on-going technology that will help us to move forward with a new wave of technology.

## **I. INTRODUCTION**

Technology industry is ever changing and the changes are for the better. New technologies are emerging fast that have changed the perception and domain of how things worked before their introduction.

Merely about nine years ago, the phrase cryptocurrency was unheard of. The first cryptocurrency Bitcoin, as it became famous, it changed the perception of transaction and how we invest money. Bitcoin also introduced a new technology named ‘Blockchain’ upon which it works. There are numerous applications of this technology and it is now incorporated with other concepts. For example, ethereum uses it for smart contracts. Ethereum is a programmable blockchain. It is different from bitcoin which aims to decentralized the currency whereas Ethereum is intended to decentralize the applications. Ethereum does this by providing the feature of smart contracts that is a help executing programmable script written in Solidity where they can create their own rules for ownership, transaction formats and state transition functions.

For instance, Internet of Things (IoT) is a technology known for more than ten years but its use has recently skyrocketed with its market capital increasing about a 100 billion dollars each year. This is possible due to overcoming of the limitation in this sector earlier, with the new innovations. The main limitations of the IoT technology were its security and integrity problems but the combination of blockchain with IoT makes it more useful and secure.

With our aim to make Vehicle Rental System the best rental system in the current marketplace, we have developed as a web application that means any user, let it be an Android or IOS, or even a Windows user, everyone can have access to this rental system irrespective of his/her operating system.

.

## **II. MOTIVATION**

During initial discussions of possible project ideas, it was decided by the team that a concept that is different from other projects, interesting, and most importantly beneficial and of practical use in its field, should be pursued. After considering numerous ideas and much thinking over, it was decided that work should be done in the area of Blockchain, particularly. There was no awareness of any similar projects in the past years, so this seemed as a good decision.

This project aims to deliver a platform which is web application in our case, enabling users or clients to come together to rent their vehicles with their choice of rent per day and security fees. The system is implemented using blockchain so there is no cost of deploying the web application as blockchain is decentralized and not a single server. Hence, this is cost-efficient. Moreover, there is no third party involved for managing transactions and procedures. Instead, this is handled using smart contracts that provides the functionality of an escrow.

Therefore, the main motivation behind this project is to make the vehicle rental system cost-effective and secure. There are traditional applications that use a third party and charge membership fee or commission fee. No such fee is charged in this implementation rather a small gas fee is charged which is very less as compared to previously mentioned fee.

Advancements can be made using sensors such as vibration sensors, smoke sensors and gate unlockers for enhanced security with availability of funds.

.

### **III. LITERATURE SURVEY**

#### **III.1 Web3.js**

Web3.js is a Javascript API. It enables any application to communicate to the smart Contracts. The web3.js requires an Abstract Binary Interface (ABI) in order to communicate to smart contracts. This is a technology that talks to the network through code. This is used by the developers as compared to metamask which is for the general users. It allows us to send money, store data, deploy contract or do essentially whatever we want to do on the network. Web3.js won't work if you don't have metamask installed.

#### **III.2 React**

ReactJS basically is a JavaScript library. It is open- source and is intended for designing user interfaces particularly for single page applications. React is a declarative, efficient, and flexible JavaScript library. It allows to compose complex User Interfaces from small and standalone pieces of code called “components”.

Its function is to handle view layer for web and mobile apps. Reusable UI components can also be designed using React. It is used to make reusable UI components to present data that changes over time.

React makes it easy to make interactive User Interfaces. It can be used for constructing simple views for each state in your application. Its plus point being that when the data changes, React will efficiently update and render the right components. Your code becomes predictable, clear using Declarative Views. It also makes debugging easier.

#### **III.3 Solidity**

Solidity is a high-level, object-oriented language. It is used for implementing smart contracts. The programs that define or guide the behavior of accounts within the Ethereum state are called smart contracts.

It is a language local to Ethereum. Solidity was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM). Solidity is statically typed. It is an object-oriented language; therefore, it supports inheritance, libraries and complex user-defined types among other features.

Contracts can be created with solidity for the intention or use in cases such as voting, blind



auctions and crowdfunding. Latest released version of Solidity should be used at the time of deploying contracts. This is due to breaking changes as well as introduction of new features and bug fixes that are introduced regularly.

### III.4 Metamask

For people who want to work on Ethereum but do not know what programming is, they use Metamask. It is a browser extension that allows user to interact with the Ethereum network. Another alternate is the Mist browser. We have worked on metamask because it is easier to use but more importantly because mist browser is in its early beta. Metamask is the most realistic way to expect common people on the Ethereum network right now.

We can turn any browser into a blockchain enabled one. With the help of Metamask, we can connect to the Ethereum mainnet but also the test networks such as Rinkeby, Rovan and Ropsten testnet. We have used Rinkeby testnet in our project.

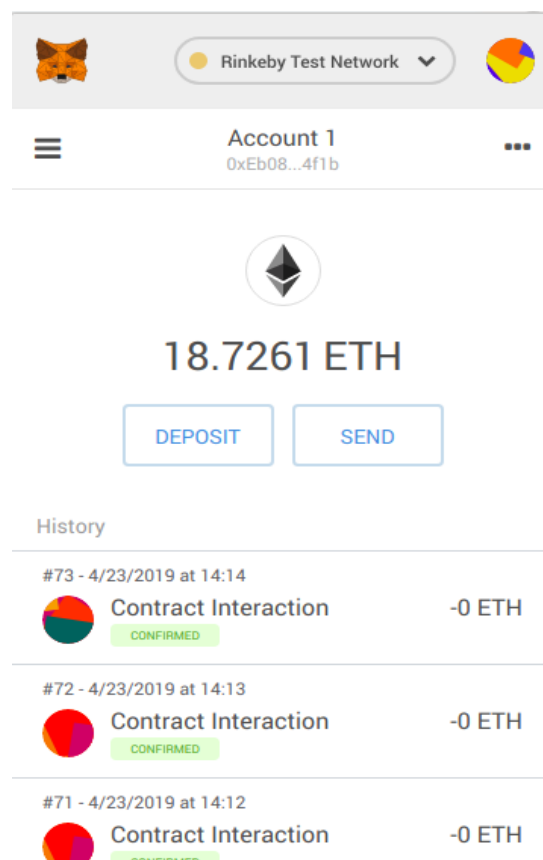


Figure 1- Snapshot of Metamask browser extension

### **III.5 Infura**

For growth of Ethereum and the entire blockchain system, the community built the infrastructure components that keep the network running. Infura is a scalable back-end infrastructure that simplifies access to Ethereum data across the Ethereum network. Infrastructure-as-a-Service (IaaS) is the most of the essential products. Infura leads the way by offering developers a set of tools to connect their apps to the Ethereum network. It is used for building DApps on the Ethereum blockchain. It is a method for connecting to the Ethereum network without having to run a full node. Infura allows the developers to connect these full nodes through its interface.

### **III.6 Blockchain**

A blockchain is a growing list of records. It consists of blocks which are linked using cryptography. Each block contains transaction data, timestamp and hash value of previous block. A blockchain is resistant to any modification by design. It was devised as the backbone technology for first ever introduced cryptocurrency known as Bitcoin by Satoshi Nakamoto. Today it has grown into so much more. It is incorporated the feature of smart contracts and resulted into Ethereum. In simple terms, a blockchain is a time-stamped series of immutable record of data that not owned by any single entity but managed by cluster of computers

There are four main pillars in the blockchain technology- ledger, consensus, cryptography and smart contracts.

### **III.7 Ethereum**

Ethereum is a blockchain-based distributed computing platform providing smart contract functionality. It is open-source and public. It supports a modified version of Bitcoin's Blockchain via transaction-based state transitions. Ether is a token or currency of blockchain of Ethereum platform. It is a decentralized platform that provides smart contracts feature that enables us to make applications that run exactly as programmed eliminating any possibility of fraud, censorship and third-party interference.

## IV. THE SMART CONTRACT PART

### Approach

Solidity is the language used for writing Smart Contracts. It is the most popular language for the same. Solidity version 0.4.17 is used specifically because this is almost fully compatible with the web3.js library which is used for accessing full node on Ethereum network. Initially, the person who hosts the main website deploys the first contract named 'FactoryRent'. This contract is the first to be deployed. When a lessor rents his/her vehicle, a new smart contract named 'Rent' is deployed using the functions defined in 'FactoryRent' contract and its address is saved in 'deployedRents' property of 'FactoryRent'.

The pseudo code cited below explains that the details of the vehicle can only be modified or deleted by its manager only. A manager is its lessor who is responsible for creation of 'Rent' contract.

```
/* impose restrictions on transaction sent to contract*/
modifier restricted(){
    require(msg.sender == manager);
    _;
}

/* only manager can call this function so that other can not change details of this contract
properties*/
function editDetails(/*parameters which have to be changed*/ ) public restricted{
    //update new values to properties
}
```

In the pseudo-code written below, the manager is confirmed by checking the account number to the saved account number in 'manager' property of respective 'Rent' contract. The code below shows that the fund transferred is held into contract when the vehicle rented is booked and is released only when the vehicle is returned. When the vehicle is booked, the vehicle becomes unavailable. Security is returned to the lessee and the rent amount is transferred to manager.

```

pragma solidity ^0.4.17;

/* main contract which handles all vehicle details*/
contract Rent{
    /* address who deployed this contract*/
    address public manager;
    bool available;
    uint popularity;
    /*when lessee request for vehicle, this payable method holds the amount send by
    lessee in the contract*/
    function takeRent(uint days) public payable{
        uint total=days*rentPerday+security;
        require(msg.value >= total);
        require(available);
        available=false;
        popularity++;
    }
    /* return security money to lessee and pay rent amount to lessor*/
    function returnSecurity() public{
        require(msg.sender == pastRent[pastRents.length-1]);
        msg.sender.transfer(security);
        manager.transfer(this.balance);
        available = true;
    }
}

```

## V. THE APPLICATION PART

### Approach

We have made a DApp, i.e., a decentralized application which for the demonstration purpose is only employed only on the local host. DApp is made using React for the UI provided. It is beneficial for presenting the data that changes with time as it automatically renders just the right components when the data changes. In the background, the DApp uses Web3.js library to interact with the Ethereum full node. Web3 library cannot work alone, therefore it needs Metamask or Blockchain enabled Mist browser. For our project, we have preferred the Metamask chrome browser extension because it is very easy to use and it is much more developed. For communication with Smart Contracts, Abstract Binary Interface (ABI) is required. As contracts are deployed on the Ethereum network in the form of Bytecode so you can access the binary data in the contract using this ABI.

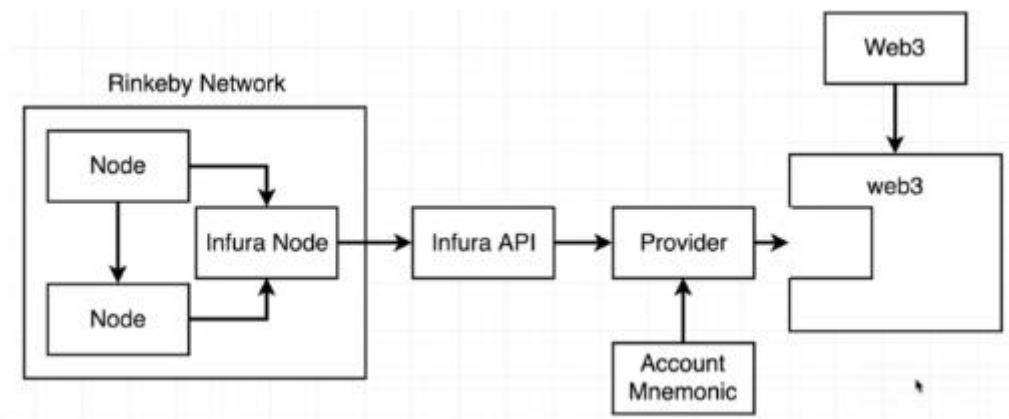


Figure 2 - Setup between network and client

This DApp mainly serves two types of users – the lessee and lessor. Lessor is the user who has to provide a vehicle for rent. Lessee is the user who takes the available and preferred vehicle on rent.

The home page of the DApp opens, showing the list of all the vehicles registered on the application. Each vehicle is attached with information such as its short name, popularity and its availability at the current time. Popularity is defined as the number of times the vehicle has been rented. The lessor of the car decides the rent per day and the security fee to be charged. These details can be viewed by any user interested in taking the car for rent but can only be changed by manager of the contract, who has created the contract, i.e., the lessor. The home page of the DApp allows anyone to become a lessor. As soon as a user registers his vehicle,

becoming a lessor, the list of vehicles gets updated. The lessee selects an available car of preference and enters the days he/she wants to take the vehicle on rent for; the charge is automatically calculated. Upon paying the amount in wei, the vehicle in the list becomes unavailable and Request Security option is opened for the respective lessee who took the vehicle on rent. On clicking this button after returning the vehicle, the security amount will be refunded to the lessor. Late returning of vehicle results in deduction of security money.

## Pseudo code

The working of a few important components of the Distributed Application are shown below using pseudo code:

### **viewHomePage()**

1. if(mist or metamsk present) :
2.     web3 = instance from current window
3.     accounts[] = web3.getAccounts()
4. else:
5.     give warning during transaction metamask not present
6.     factory = new web3.eth.Contract(interface,address)
7.     get list of all deployed Rent contracts from ethereum blockchain
8.     for each rent in deployedContracts:
9.         rent.methods().getDetails().call() // Details of each vehicle
10.     create react object of each contract details
11. display or render details of each contract in web page

### **createRentContract()**

1. import web3 and factory
2. account = web3.getAccounts()[0] // get first account
3. get details from form filled by manager
4. resultAddress = factory.methods.createRentContract(vehicle details).send(account)
5. if(transaction confirm):
6.     deduct gas fee from account
7.     manger = account.getAddress()
8.     web3.eth.addDeployedRent(resultAddress)
9.     deployedContract.push(resultAddress)
10.     route('Home Page')
11. else:
12.     display error Details
13. update Home page list of vehicles
14. render details

### **takeOnRent()**

1. if(availability == true and metamask running):
2.     user enter amount of days for rent
3.     days=form.getDays()
4.     security=rentContract.getSecurityAmount(address).call()
5.     pay\_amount = days\*rentPerDay+security // calculate total money to pay
6.     //below method is payable so amount of money go to contract
7.     rentContract.methods.takeRent(pay\_amount).send(account[0])
8.     if(transaction confirm):
9.         deduct money from account
10.        availability=false
11.        popularity++
12.     else:
13.        display error message
14. else:
15.     show metamask not running
16.     render page

### **takeSecurity()**

1. if( metamask running ) :
2.     web3=instance for window
3.     account=web.eth.getAccounts()[0] // get first account
4.     timeLimit=rentContract.methods.getTimeLimit().call()
5.     if(timeLimit > currentTime):
6.         require(account==tenant)
7.         rentContract.methods.returnSecurity().send()
8.     else:
9.         require(account==tenant)
10.        rentContract.methods.cutSecurity( cut\_percentage ).send()
11.        availability = true
12. else:
13.     show metamask not running
14. render Page
15. Split data set in training and testing sets

## VI. DATA FLOW DIAGRAM

### VI.1 Level - 0 DFD

Level – 0 data flow diagram or Context diagram is used to provide the highest-level overview of the entire system of the application. Users can either become a lessor or a lessee. A lessor can rent his/her vehicle. A lessee can book any of the available vehicles using the same DApp platform.

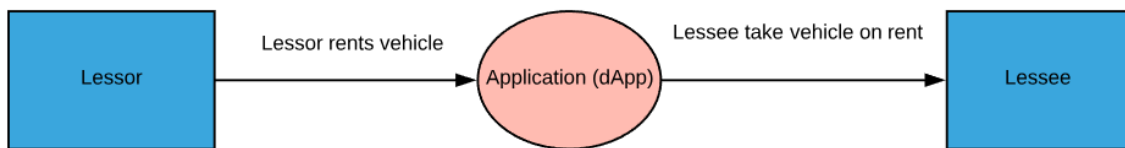


Figure 3 - Level-0 DFD

### VI.2 Level – 1 DFD

Level – 1 data flow diagram or Context diagram is used to provide a more detailed breakout of pieces of the entire system of the application. It highlights the main functions performed by the system and breaks the high-level process into its sub processes.

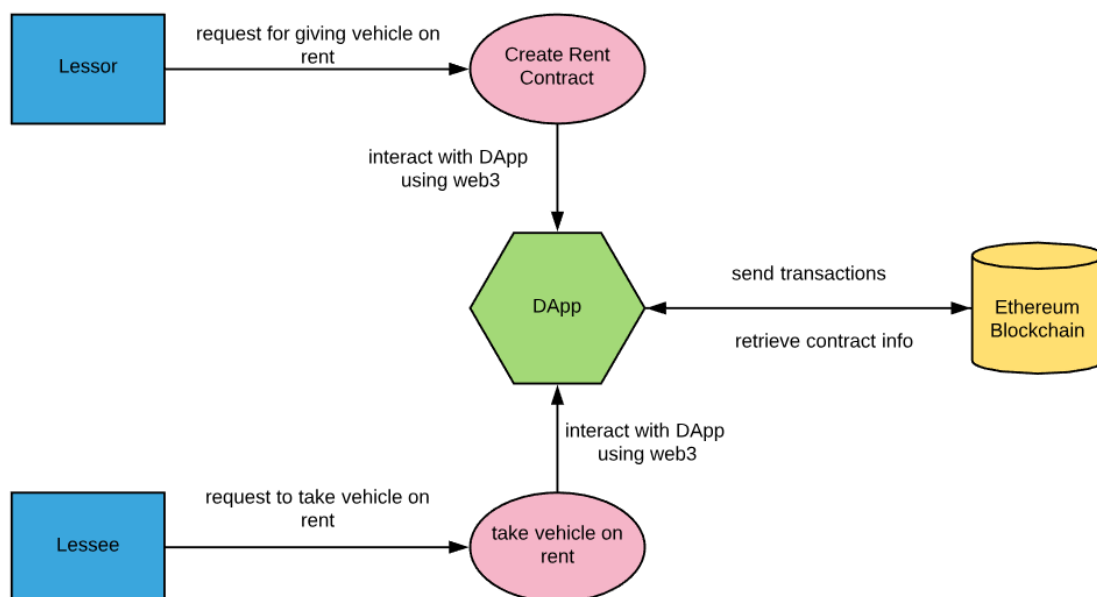


Figure 4 - Level-1 DFD



As our aim was to design a DApp that provides a platform where users can rent their vehicle or book a available vehicle. This process is secured using Ethereum blockchain. Web3 library is used to interact with full node on Ethereum network.

### VI.3 Level – 2 DFD

Level – 2 data flow diagram or Context diagram it goes one step deeper into parts of Level – 1 DFD. It can be used to plan or record the specific makeup of a system. A decentralized Application (DApp) provides the list of all registered vehicles. Gps receiver provides the location of the rented vehicle accessible on the DApp only by the manager. Metamask is the browser extension used to manage accounts.

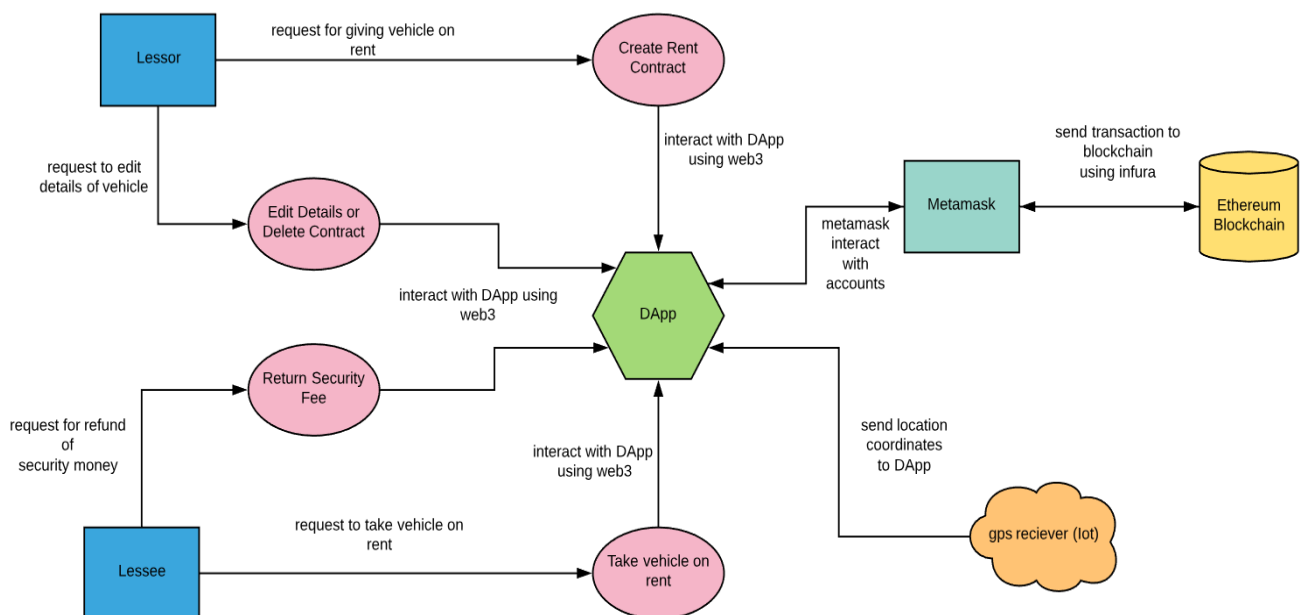


Figure 5 - Level-2 DFD

## VII. RESULT

Our project provides a platform where users can come together and become a lessor or lessee for their vehicles to implement a vehicle rental system. This implementation is a proof of concept that Blockchain technology can be used for decentralized applications, overcoming their security and integrity problems.

The decentralized application of vehicle rental system has been deployed successfully on the testnet. Among many available testnets, we have used Rinkeby network. The address of deployed smart contract is “0xA99A875B32BA87bb8F7b53A6fb80a335AE1916a9”.

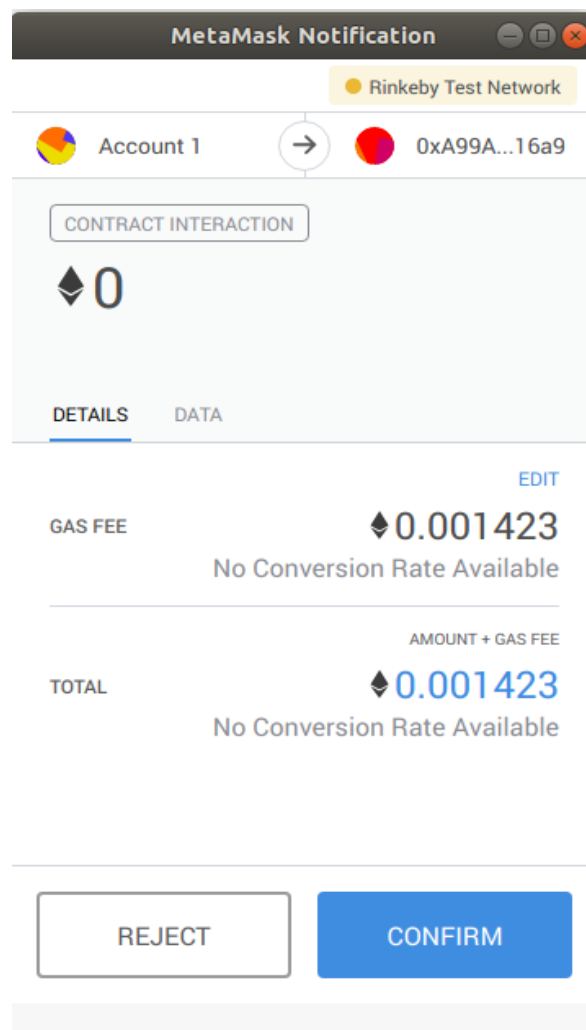


Figure 6- Sending transaction through Metamask

Rental System
Registered vehicles list
+

### Registered Vehicles

**Ciaz vdi**  
popularity of vehicle - 0  
vehicle available  
[Check Details](#)

Rent your vehicle

**Creta SX**  
popularity of vehicle - 1  
vehicle unavailable  
[Check Details](#)

**Ferrari Pista 428**  
popularity of vehicle - 0  
vehicle available  
[Check Details](#)

**Yamaha YZF R1**  
popularity of vehicle - 0  
vehicle available

Figure 7 - Home page of the DApp

localhost:3000/rents/new x
+

localhost:3000/rents/new

Rental System
Registered vehicles list
+

### Register your vehicle

Name of vehicle  
Ciaz vdi
Short Name

Minimum Security Amount  
2000
wei

Address and Description of Vehicle  
nagina chowk, Dhampur, UP-246761: ph-457896321
Complete Details

Rent Per Day  
1100
wei

register!

Figure 8 – Vehicle Registration page

Rental System	Registered vehicles list		+
---------------	--------------------------	--	---

**Ciaz vdi**

**0xb726fA4e9b5bBe114BF46eFd40Ef622dAE37aDef**  
Address of Manager  
The manager is owner and wishes to rent his vehicle

**security amount**  
You must give at least this much wei to rent this vehicle  
Minimum security to pay (wei) 2000

**Availability**  
vehicle is available

**Details**  
give below are complete details  
nagina chowk, Dhampur, UP-246761: ph-457896321

**popularity**  
\*no of time vehicle rented - 0

**Rent per Day**  
1100 per day

Enter Number of Days

time

Amount to be paid: - 2000 wei  

Pay!

Edit Details

Unlist vehicle

Figure 9 – Vehicle description and pay rent page

Rental System	Registered vehicles list		+
---------------	--------------------------	--	---

**Creta SX**

**0xC1A8D9326D6e7551E4b7A52D8700511eabF2d721**  
Address of Manager  
The manager is owner and wishes to rent his vehicle

**security amount**  
You must give at least this much wei to rent this vehicle  
Minimum security to pay (wei) 2100

**Availability**  
vehicle unavailable

**Details**  
give below are complete details  
harnathpur kota ,Hapur,UP-245101 : ph-1234567890

**popularity**  
\*no of time vehicle rented - 1

**Rent per Day**  
1200 per day

Request Security Fee

Can not edit vehicle on rent

Figure 10 – Vehicle details after it is booked

Rental System	Registered vehicles list	+
---------------	--------------------------	---

### Edit your vehicle details

**Name of vehicle**

Ciaz vdi Short Name

**Minimum Security Amount**

2000 wei

**Address and description of Vehicle**


nagina chowk, Dhampur, UP-246761: ph-457896321 Complete Details

**Rent Per Day**

1100 wei

[Edit Details!](#)

Figure 11 – Vehicle details edit page accessible to manager when vehicle is available



All Filters
Search by Address / Txhash / Block / Token / Ens

Testnet Network
Home
Blockchain
Tokens
Misc
Rinkeby

### Transaction Details

Overview

[ This is a Rinkeby Testnet Transaction Only ]

Transaction Hash:	0x36975e9094092d0637fe4f460aaca7d861c78f51c845de41c7a82e92dba99b77
Status:	Success
Block:	4258549 4 Block Confirmations
TimeStamp:	56 secs ago (Apr-23-2019 08:21:01 AM +UTC)
From:	0xcf01971db0cab2cbee4a8c21bb7638ac1fa1c38c
To:	[Contract 0xa99a875b32ba87bb8f7b53a6fb80a335ae1916a9 Created]
Value:	0 Ether (\$0.00)
Transaction Fee:	0.001510141 Ether (\$0.000000)

[Click to see more](#)

Figure 12 – Deployed contract details on Etherscan

## **VIII. CONCLUSION**

As a result of this project, the Vehicle Rental System was developed. Blockchain is main underlying technology used. Blockchain is a decentralized, i.e. peer to peer technology. It is a further advancement towards decentralized web. This will give the users on the vehicle rental platform more opportunities and offers. In addition, it provides a much more secure transactions which are cost-efficient as there is no third-party involvement and gas fees is very low.

The experience gained throughout the development process is very helpful in terms of applied knowledge, and will definitely be reflected upon on future projects. Winding up, we can say that initial objectives of this project were achieved to an extent. Due to limited availability of funds, not much has been done in the field of IoT regarding this project. Further advancements such as door unlockers, vibration sensors for accidents and smoke detectors for fire can be used and process can be completely automated.

## REFERENCES

- [1] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", October 31, 2008
- [2] M. Singh, A. Singh, S. Kim, "Blockchain: Game changer for securing IoT data", 2108 IEEE 4<sup>th</sup> World Forum on Internet of Things (WF-IoT), pp. 51-55, 2018
- [3] S. Huh, S. Cho, S. Kim, "Managing IoT devices using blockchain platform", 2017 19<sup>th</sup> International Conference on Advanced Communication Technology (ICACT), pp. 464-467, 2017
- [4] Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for Internet of Things. IEEE Access, 4, 2292-2303
- [5] Vitalik Buterin, "A Next-Generation Smart Contract and Decentralized Application Platform," <https://github.com/ethereum/wiki/wiki/White-Paper>, 2013

## APPENDIX:

### index.js -

```
import React, { Component } from 'react';
import { Card, Button } from 'semantic-ui-react';
import factory from '../ethereum/factory';
import Layout from '../components/Layout';
import { Link } from '../routes';
import RentContract from '../ethereum/rentContract';
// import libraries
class RentContractIndex extends Component {
  // initialize properties
  static async getInitialProps() {
    const listOfRents = await factory.methods.returnDeployedList().call();
    const promiseArray = listOfRents.map(async (address) => {
      const rent = RentContract(address);
      const valueofpopularity = await rent.methods.popularity().call();
      const valueofname = await rent.methods.getName().call();
      const valueofavailability = await rent.methods.getAvailability().call();
      // return values
      return {
        keys: "values",
        addr: address,
        name: valueofname,
        availability: valueofavailability,
        popularity: valueofpopularity
      };
    });
    const finalresults = await Promise.all(promiseArray);
    return { finalresults };
  }
  // check availability
```



```

giveAvailability(check){
  if(check){
    return "available";
  }
  else{
    return "unavailable";
  }
}

// give color according to availability
getColor(check){
  if(check){
    return "green";
  }
  else{
    return "red";
  }
}

// render contract list
renderRentContracts() {
  const items = this.props.finalresults.map((result) => {
    return {
      header: <div style={{ fontSize: '24px', color:'black' }}><h1>{result.name}</h1></div>,
      description: (
        <div style={{ color: this.getColor(result.availability), fontSize: '20px' }}>
          <br></br>
          vehicle {this.giveAvailability(result.availability)}
          <br></br><br></br>
          <Link route={`/rents/${result.addr}`} >
            <a>Check Details</a>
          </Link>
        </div>
      ),
      meta: <div style={{ fontSize: '16px', color:'maroon' }}>popularity of vehicle - {result.popularity}</div>,
      fluid: true
    }
  )
}

```

```

    };
  });

  return <Card.Group items={items} />;
}

// main fuction to render each components
render() {
  return (
    // main layout of page
    <Layout>
      <div>
        <h3>Registered Vehicles</h3>
        <Link route="/rents/new">
          <a>
            <Button
              content="Rent your vehicle"
              floated="right"
              icon="add circle"
              primary
            />
          </a>
        </Link>
        {this.renderRentContracts()}
      </div>
    </Layout>
  );
}
}

```

```
export default RentContractIndex;
```

**new.js -**

```
import React, { Component } from 'react';
```

```

import { Form, Button, Input, Message } from 'semantic-ui-react';
import Layout from '../components/Layout';
import factory from '../ethereum/factory';
import web3 from '../ethereum/web3';
import { Router } from '../routes';
// import libraries
class NewRentContract extends Component {
  state = {
    showName: "",
    description: "",
    minimumSecurity: "",
    rentPerDay: "",
    MessageOfError: "",
    waitingState: false
  };

  FormSubmitFunction = async event => {
    event.preventDefault();
    // change state of button
    this.setState({ waitingState: true, MessageOfError: "" });

    try {
      const wallets = await web3.eth.getAccounts();
      // craete new rent contract
      await factory.methods
        .createRent(this.state.minimumSecurity, this.state.description, this.state.rentPerDay, this.state.showName)
        .send({
          from: wallets[0]
        });
    } catch (err) {
      this.setState({ MessageOfError: err.message });
    }
  }
}

```

```

// change state of button
this.setState({ waitingState: false });
};

render() {
  return (
    // render main layout
    <Layout>
      <h3>Register your vehicle</h3>
      // craete form to sumit information
      <Form FormSubmitFunction={this.FormSubmitFunction} error={!!this.state.MessageOfError}>
        // form input Field
        <Form.Field>
          <label>Name of vehicle</label>
          <Input
            labelPosition="right"
            label="Short Name"
            onChange={event =>
              this.setState({ showName: event.target.value })}
            value={this.state.showName}
          />
          // take security money amount
        </Form.Field>
        <Form.Field>
          <label>Minimum Security Amount</label>
          <Input
            labelPosition="right"
            label="wei"
            onChange={event =>
              this.setState({ minimumSecurity: event.target.value })}
            value={this.state.minimumSecurity}
          />
        </Form.Field>
        // fill address of lessor

```

```

<Form.Field>
  <label>Address and Description of Vehicle</label>
  <Input

    labelPosition="right"
    label="Complete Details"
    onChange={event =>
      this.setState({ description: event.target.value })}
    value={this.state.description}
  />
</Form.Field>
<Form.Field>
  <label>Rent Per Day</label>
  <Input
    label="wei"
    labelPosition="right"
    value={this.state.rentPerDay}
    onChange={event =>
      this.setState({ rentPerDay: event.target.value })}
  />
</Form.Field>
// show error message in page
<Message error header="Oops!" content={this.state.MessageOfError} />
// button to submit form
<Button waitingState={this.state.waitingState} primary>
  register!
</Button>
</Form>
// finish layout
</Layout>
);
}
}

```

```
export default NewRentContract;
```

### **show.js -**

```
import React,{ Component } from 'react';
import { Form,Button,Card,Message,Grid } from 'semantic-ui-react';
import Layout from '../components/Layout';
import RentContract from '../ethereum/rentContract';
import web3 from '../ethereum/web3';
import TakeOnRentForm from '../components/TakeOnRentForm';
import factory from '../ethereum/factory'
import { Link } from '../routes';
import { Router } from '../routes';
// import libraries
class RentShow extends Component {
  // initialize prperties
  static async getInitialProps(props) {
    const rents = RentContract(props.query.managerAddress);
    const results = await rents.methods.getSummary().call();

    return {
      managerAddress: props.query.managerAddress,
      security: results[1],
      availablity: results[2],
      popularity: results[4],
      description: results[3],
      rentPerDay: results[5],
      name: results[6]
    };
  }
  state = {
    buttonLoading:false,
    messageError:"",
    messageError2: "
```

```

};
checkavail(availablity){
  if(availablity){
    return 'vehicle available';
  }
  else{
    return 'vehicle unavailable';
  }
}
// render each cards
displayCards() {
  const {
    managerAddress,
    security,
    availablity,
    description,
    popularity,
    rentPerDay,
    name
  } = this.props;

  const items = [
    {
      header: managerAddress,
      meta: 'Address of Manager',
      description:
        'The manager is owner and wishes to rent his vehicle',
      style: { overflowWrap: 'break-word' }
    },
    {
      header: 'security amount',
      meta:
        'You must give at least this much wei to rent this vehicle',
      description:

```

```

    'Minimum security fee to pay (wei) ' + security
  },
  {
    header: 'Availability',
    meta: "",
    description: this.checkavail(availability)
  },
  {
    header: 'Details',
    meta: 'give below are complete details',
    description: description
  },
  {
    header: 'popularity',
    description: '*no of time vehicle rented - ' + popularity
  },
  {
    header: 'Rent per Day',
    meta: "",
    description: rentPerDay + ' per day'
  }
];

```

```

return <Card.Group items={items} />;

```

```

}

```

```

onSubmitForm = async event => {

```

```

  // to stop default function

```

```

  event.preventDefault();

```

```

// Change state of variables

```

```

this.setState({ messageError: "",buttonLoading: true, });

```

```

try {

```

```

  const wallets = await web3.eth.getAccounts();

```

```

  await factory.methods

```



```

.deleteRent(this.props.managerAddress)
.send({
  from: wallets[0]
});
// change route to home page
Router.replaceRoute('/');
} catch (err) {
  this.setState({ messageError: err.message });
}
// change state of button
this.setState({ buttonLoading: false });
};

onSubmitTime = async event => {
  event.preventDefault();
  // set states of variables
  this.setState({ buttonLoading: true, messageError2: " " });
  // try catch to handle errors
  try {
    const wallets = await web3.eth.getAccounts();
    const rent = RentContract(this.props.managerAddress);
    const tmp = await rent.methods.rentingTime().call();
    const timelimit = (new Date(Number(tmp))).getTime();
    if(timelimit>Date.now()){
      await rent.methods
        .returnSecurity()
        .send({
          from: wallets[0]
        });
    }
  } else{
    const deductamount = parseInt((Date.now()-timelimit)/(60000));
    await rent.methods
      .cutSecurity(deductamount)

```

```

        .send({
          from: wallets[0]
        });
      }
      Router.replaceRoute(`/rents/${this.props.managerAddress}`);
    } catch (err) {
      this.setState({ messageError2: err.message });
    }
    // disable loading of button
    this.setState({ buttonLoading: false });
  };

  renderCheck(){
    if(this.props.availablity){
      return(
        <Grid.Column>
          // form for edit details
          <Form error={!this.state.messageError} onSubmit={this.onSubmitForm}>
            <Link route={`/rents/${this.props.managerAddress}/requests`} >
              <a>
                <Button primary>Edit Details</Button>
              </a>
            </Link>
            <Message error header="Oops!" content={this.state.messageError} />
            // button to unlist vehicle
            <Button loading={this.state.buttonLoading} secondary >Unlist vehicle</Button>
          </Form>
        </Grid.Column>
      );
    }
    else{
      return(
        <Grid.Column style={{color:'red'}} >
          <h3>Can not edit vehicle on rent</h3>

```

```

        </Grid.Column>
    );
}
}
renderColumn(){
    if(this.props.availablity){
        return (
            <Grid.Column width={6}>
                <TakeOnRentForm address={this.props.managerAddress} rentPerDay={this.props.rentPerDay}
security={this.props.security} />
            </Grid.Column>
        );
    }
    else{
        return(
            <Grid.Column width={6}>
                // form for request security money
                <Form onSubmit={this.onSubmitTime} error={!this.state.messageError2}>
                    <Message error header="Oops!" content={this.state.messageError2} />
                    <Button loading={this.state.buttonLoading} secondary >Request Security Fee</Button>
                </Form>
            </Grid.Column>
        );
    }
}

// render componenets
render() {
    return (
        <Layout>
            // show name of vehcile
            <h3>{this.props.name}</h3>
            <Grid>
                <Grid.Row>
                    <Grid.Column width={12}>{this.displayCards()}</Grid.Column>

```

```

        {this.renderColumn()}
      </Grid.Row>
    <Grid.Row>
      {this.renderCheck()}
    </Grid.Row>
  </Grid>
</Layout>
);
}
}

```

export default RentShow;

### **index.js -**

```

import React, { Component } from 'react';
import { Link } from '../routes';
import Layout from '../components/Layout';
import RentContract from '../ethereum/rentContract';
import { Form, Button, Input, Message, Checkbox } from 'semantic-ui-react';
import web3 from '../ethereum/web3';
import { Router } from '../routes';
// here import libraries
class EditRequest extends Component {
  static async getInitialProps(props) {
    const rents = RentContract(props.query.managerAddress);
    const summary = await rents.methods.getSummary().call();

    return {
      managerAddress: props.query.managerAddress,
      security: summary[1],
      availability: summary[2],
      rentPerDay: summary[5],
      description: summary[3],
    };
  }
}

```

```

    popularity: summary[4],
    name: summary[6]
  };
}

```

```

state = {
  managerAddress: this.props.managerAddress,
  showName: this.props.name,
  minimumSecurity: this.props.security,
  description: this.props.description,
  rentPerDay: this.props.rentPerDay,
  MessageErrors: "",
  waiting: false
};

```

```

formSubmit = async event => {
  event.preventDefault();
  //Change state fo button
  this.setState({ waiting: true, MessageErrors: " " });
  // try catch for errors
  try {
    const accountNumber = await web3.eth.getAccounts();
    const rents = RentContract(this.state.managerAddress);
    await rents.methods

```

```

.editDetails(this.state.showName,this.state.minimumSecurity,this.state.description,this.state.rentPerDay,true)
    .send({
      from: accountNumber[0]
    });
    // chnage path to diplay another page
    Router.replaceRoute(`/rents/${this.props.managerAddress}`);
  } catch (err) {
    // chnage state of message
    this.setState({ MessageErrors: err.message });

```

```

    }
// chnage state of messag
    this.setState({ waiting: false });
};
// main render function
render() {
    return (
        // layout tag for page
        <Layout>
            <h3>Edit your vehicle details</h3>
            // form to get information
            <Form formSubmit={this.formSubmit} error={!!this.state.MessageErrors}>
                // name of vehicle
                <Form.Field>
                    <label>Name of vehicle</label>
                    <Input
                        label="Short Name"
                        labelPosition="right"
                        value={this.state.showName}
                        onChange={event =>
                            this.setState({ showName: event.target.value })}
                    />
                </Form.Field>
                //security money
                <Form.Field>
                    <label>Minimum Security Amount</label>
                    <Input
                        labelPosition="right"
                        label="wei"
                        onChange={event =>
                            this.setState({ minimumSecurity: event.target.value })}
                        value={this.state.minimumSecurity}
                    />

```

```

</Form.Field>
// address of vehicle
<Form.Field>
  <label>Address and description of Vehicle</label>
  <Input
    labelPosition="right"
    label="Complete Details"
    onChange={event =>
      this.setState({ description: event.target.value })}
    value={this.state.description}
  />
</Form.Field>
//rent per day of Vehicle
<Form.Field>
  <label>Rent Per Day</label>
  <Input
    labelPosition="right"
    label="wei"
    onChange={event =>
      this.setState({ rentPerDay: event.target.value })}
    value={this.state.rentPerDay}
  />
</Form.Field>
// message for diplay errors
<Message error header="Oops!" content={this.state.MessageErrors} />
// button for submit form
<Button loading={this.state.waiting} primary>
  Edit Details!
</Button>
</Form>
</Layout>
// render these components
);
}

```

```
}
```

```
export default EditRequest;
```

### **Rent.sol -**

```
// specific version of solidity
```

```
pragma solidity ^0.4.17;
```

```
// this contract used to deploy and manage other contracts
```

```
//this contract deployed only once by website host
```

```
contract FactoryRent{
```

```
    // records of Rent Contract deployed
```

```
    address[] public RentDeployed;
```

```
    // Records of managers who deployed Rent Contract
```

```
    mapping(address => address[]) managerContractList;
```

```
    // true if manager exists
```

```
    mapping(address => bool) managerList;
```

```
    // no managers which are lessor
```

```
    uint public managerCount;
```

```
    // constructor
```

```
    function FactoryRent() public {
```

```
        managerCount=0;// initial value
```

```
    }
```

```
    // function for creating Rent Contract by manager
```

```
    function createRent(uint sec,string desc,uint rnt,string nme) public{
```

```
        address newRent = new Rent(sec,desc,rnt,nme,msg.sender);
```

```
        RentDeployed.push(newRent);
```

```
        managerList[msg.sender]=true;
```

```
        managerContractList[msg.sender].push(newRent);
```

```
        if(!managerList[msg.sender]){
```

```
            managerCount++;
```



```

    }
}

// only manager can delete this contract
// its updates managerContractList after deleting Rent Contract
function deleteRent(address toDelete) public {
    require(managerList[msg.sender]);
    bool flag=false;
    for(uint i=0;i<managerContractList[msg.sender].length;i++){
        if(managerContractList[msg.sender][i]==toDelete){
            flag=true;
        }
    }
    require(flag);
    uint index;
    // now delete from list of deployed rent contracts
    for(uint j=0;j<RentDeployed.length;j++){
        if(RentDeployed[j]==toDelete){
            index=j;
        }
    }
    // more than one rent contract peresent
    if(RentDeployed.length>1){
        RentDeployed[index]=RentDeployed[RentDeployed.length-1];
    }
    RentDeployed.length--;
}

// get all deployed Rent Contract
function returnDeployedList() public view returns (address[]){
    return RentDeployed;
}

// check manager exists or not
function searchManager(address mngr) public view returns(bool){

```

```

    return managerList[mngr];
}
// list of all contracts deployed by manager
function getManagerContracts(address mngr) public view returns(address[]){
    return managerContractList[mngr];
}

}

// main contract which handles all information about renting vehicle
//this contract distribute money to managers and renters
contract Rent{
    address public manager;// who deployed this contract
    string name;// short name of Vehicle
    string public description; // full description about vehicle
    address[] public pastRents;// list of peoples who rent this vehicles
    uint public popularity;// how many times vehicle rented
    uint public security;// security money
    bool public availablity;
    uint public rentPerDay;
    string public rentingTime;//time at which he take vehicle at rent
    uint public timeOfRent;// amount of time for rent

    // restrict permissions only manager can access
    modifier restricted() {
        require(msg.sender == manager);
        _;
    }

    // constructor to set initial properties
    function Rent(uint sec,string desc,uint rnt,string nme,address creator) public{
        security=sec;
        name=nme;
        manager=creator;
    }
}

```

```
description=desc;
availability=true;
popularity=0;
rentPerDay=rnt;
rentingTime="";
timeOfRent=0;
}
```

```
// take vehicle on rent
```

```
// this method send money to contract
```

```
function takeRent(uint dys,string curentTime) public payable{
    uint total=dys*rentPerDay+security;
    require(msg.value>=total);
    require(availability);
    rentingTime=curentTime;
    popularity++;
    availability=false;
    timeOfRent=dys;
    pastRents.push(msg.sender);
}
```

```
// manager can edit details about vehicle
```

```
function editDetails(string newname,uint newsecurity,string newDescription,uint newrentPerDay,bool
newstatus) public restricted{
    name = newname;
    description=newDescription;
    security=newsecurity;
    rentPerDay=newrentPerDay;
    availability=newstatus;
}
```

```
/*
```

```
function editSecurity(uint newSecurity) public restricted{
    security=newSecurity;
}
```

```

function editName(string newName) public restricted{
    name=newName;
}

function editAvaialblity(bool newAvaialble) public restricted{
    availablity=newAvaialble;
}

function editRentPerDay(uint newRentPerDay) public restricted{
    rentPerDay=newRentPerDay;
}

function editDescription(string newDescription) public restricted{
    description=newDescription;
}
*/

// return security amount to renter
// rentpay send to manager of contract
function returnSecurity() public {
    uint len=pastRents.length;
    require(msg.sender == pastRents[len-1]);
    msg.sender.transfer(security);
    manager.transfer(this.balance);
    availablity=true;
    rentingTime="";
    timeOfRent=0;
}

// get short name of the vehicle
function getName() public view returns(string){
    return name;
}

// to deduct certain amount of money from security
function cutSecurity(uint amountTobeDeduct) public{
    uint len=pastRents.length;
    require(msg.sender == pastRents[len-1]);

```

```

    if(amountTobeDeduct<security){
        msg.sender.transfer(security- amountTobeDeduct);
    }
    manager.transfer(this.balance);
    availablity=true;
    rentingTime="";
    timeOfRent=0;
}

// to get all details about contract
function getSummary() public view returns (
    address, uint, bool, string, uint, uint,string
) {
    return (
        manager,
        security,
        availablity,
        description,
        popularity,
        rentPerDay,
        name
    );
}

// to return availablity of vehicle
function getAvailablity() public view returns(bool){
    return availablity;
}
}

```

### **TakeOnRent.js-**

```

import { Input,Form, Button,Message } from 'semantic-ui-react';
import React,{ Component } from 'react';

```

```

import RentContract from '../ethereum/rentContract';
import { Router } from '../routes';
import connectWeb from '../ethereum/web3';
// import libraries
class TakeOnRentForm extends Component {
  state = {
    value: "",
    days:"",
    buttonLoading: false,
    messageError: ""
  };
  // when form is submitted
  onSubmitForm = async event => {
    // prevent default events
    event.preventDefault();
    const rent = RentContract(this.props.address);
    this.setState({ buttonLoading: true, messageError: "" });
    let passTime=new Date(Date.now()+(this.state.days*60*1000));
    try {
      const wallets = await connectWeb.eth.getAccounts();
      await rent.methods.takeRent(this.state.days,passTime.getTime().toString()).send({
        value: this.state.value,
        from: wallets[0],
        gas: 1000000
      });
    }
  // change route to diff pages

    Router.replaceRoute(`/rents/${this.props.address}`);
  } catch (err) {
    this.setState({ messageError: err.message });
  }
  // change state of button
  this.setState({ buttonLoading: false, value: "" });
};

```

```

// main render function
render() {
  return (
    //form to submit details like number of days
    <Form error={!this.state.messageError} onSubmit={this.onSubmitForm} >
    // input Field
    <Form.Field>
      <label>Enter Number of Days</label>
      <Input
        value={this.state.days}
        type='number'
        min={1}
        label="time"
        onChange={event => this.setState({ days: event.target.value })}
        labelPosition="right"
      />
    </Form.Field>
    // new field of form
    <Form.Field>
      <label>Amount to be paid: -
{this.state.value=(Number(this.state.days)*Number(this.props.rentPerDay)+Number(this.props.security))}
wei</label>
    </Form.Field>
    // to show error message
    <Message error header="Oops!" content={this.state.messageError} />
    // button to submit form
    <Button loading={this.state.buttonLoading} primary>
      Pay!
    </Button>
  </Form>
);
}
}

```

```
export default TakeOnRentForm;
```

#### **routes.js-**

```
// define routes to visit diff pages in web application
const routes = require('next-routes')();
// assign different routes to render pages
// this will create new links
routes
  .add('/home','/')
  .add('/rents/new', '/rents/new')
  .add('/rents/:managerAddress', '/rents/show')
  .add('/rents/:managerAddress/requests', '/rents/requests/index');
// export routes
module.exports = routes;
```

#### **compile.js-**

```
// import solc compiler and other libraries
const fileextra = require('fs-extra');
const compilerSolidity = require('solc');
const directory = require('path');

const buildPath = directory.resolve(__dirname, 'build');
//sync with buildPath
fileextra.removeSync(buildPath);

const RentPath = directory.resolve(__dirname, 'contracts', 'Rent.sol');
//Read Rent.sol file
const Pathsource = fileextra.readFileSync(RentPath, 'utf8');
// now compile Rent.sol file
const result = compilerSolidity.compile(Pathsource, 1).contracts;
// if buildPath not exists make one
fs.ensureDirSync(buildPath);
```



```
// now write each contract into diff json files
for (let newcontract in result) {
  fileextra.outputJsonSync(
    directory.resolve(buildPath, newcontract.replace(':', ' ') + '.json'),
    result[newcontract]
  );
}
```