Blockchain Based System For DMV

Sai Siva Saketh Kantimahanthi - 01099378

Kavyashree Sirigere Prakash - 01090444

Pragathi Lankapalli - 01072601

Acknowledgment

We would like to express our sincere gratitude to Dr. Ravi Mukkamala, for the continuous support in this course and project. Thanks for his patience, motivation, and immense knowledge. His guidance helped us throughout the course and the project.

Contents

Introduction

The idea of Project is to implement a completely online blockchain-based system for DMV of Virginia. Currently, DMV’s system is based on Traditional database systems and customer agent is required to provide or approve any service so as to authenticate the documents and person.DMV provides several services to its customers such as Drivers, Vehicle Owners, Police Departments, Car Dealers and more. Various features asked to implement were

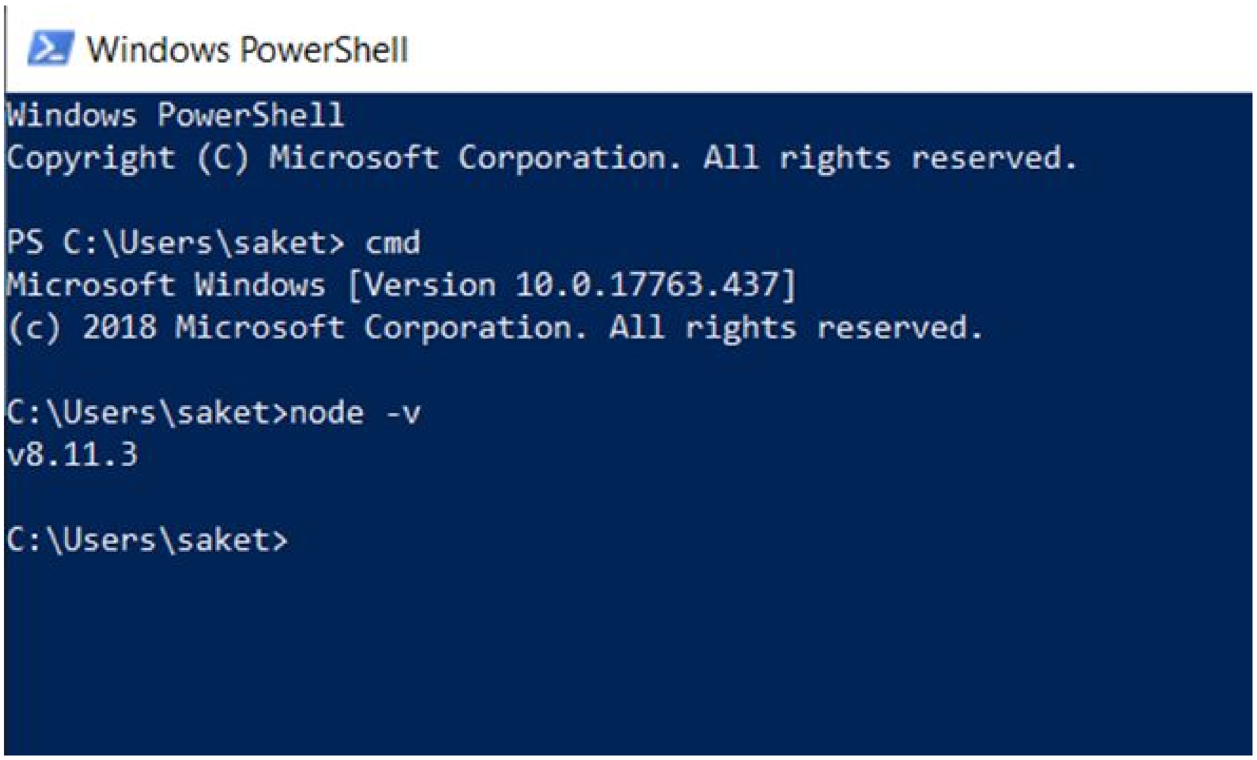
1. Online Services which includes
    - Vehicle registration renewal
    - Address change,
    - DL renewal
    - Report a vehicle sold/traded
2. Driver/ ID services
    - Practice exams
    - Real ID
    - Obtaining a vital record
3. Vehicle services
    - Selling/donating a vehicle
    - Titling a vehicle in Virginia

Technical Framework and Environment Setup

Versions Used:

Node

Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. Node.js is an open source, a cross-platform runtime environment for developing server-side and networking applications. The current version we used is v8.11.3.

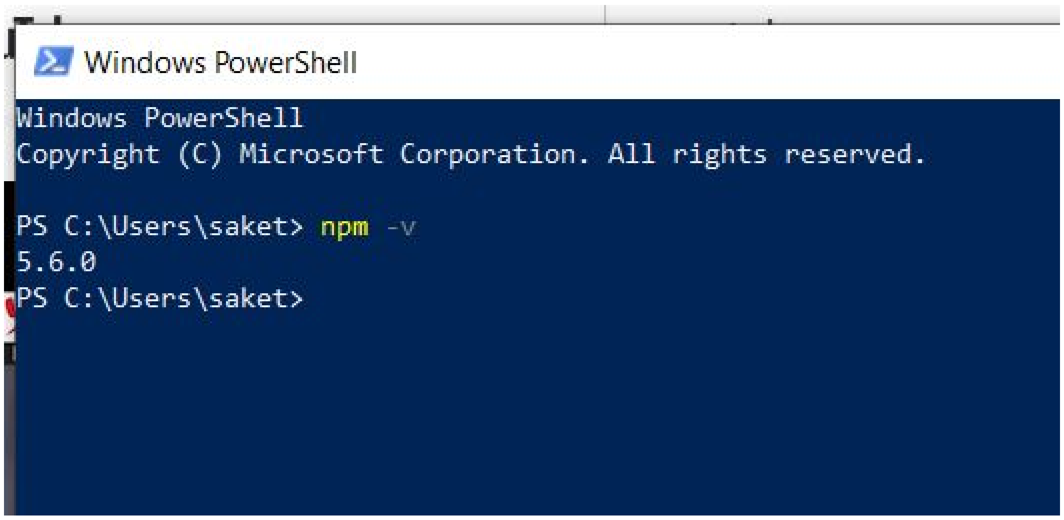


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\saket> cmd
Microsoft Windows [Version 10.0.17763.437]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\saket>node -v
v8.11.3

C:\Users\saket>
```

Npm

npm is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages called the npm registry.

The npm version used for this project is 5.6.0.

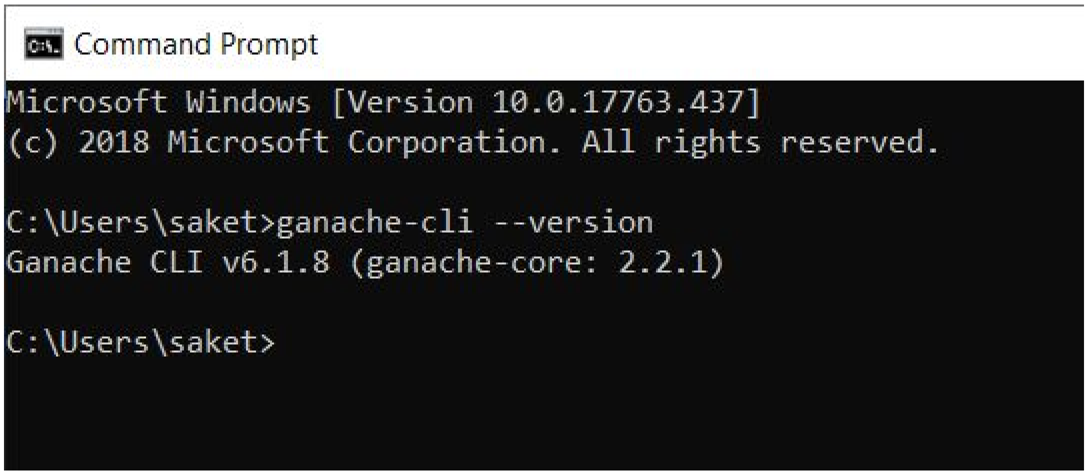


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\saket> npm -v
5.6.0
PS C:\Users\saket>
```

Ganache CLI

Ganache-cli Is the latest version of TestRPC: a fast and customizable blockchain emulator. It allows you to make calls to the blockchain without the overheads of running an actual Ethereum node. The current version of Ganache CLI is v6.1.8. To install type the following command in command prompt.

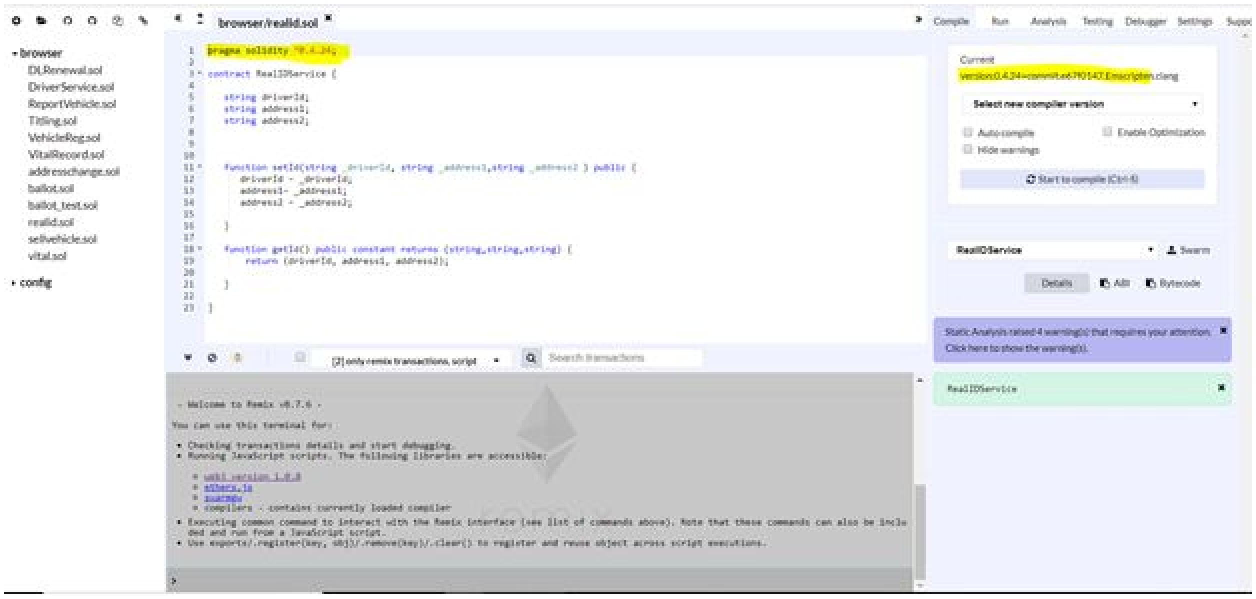
npm install -g ganache-cli



```
Microsoft Windows [Version 10.0.17763.437]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\saket>ganache-cli --version
Ganache CLI v6.1.8 (ganache-core: 2.2.1)

C:\Users\saket>
```

Remix Browser

We chose to write the smart contracts in Solidity programming language and IDE as Remix browser to program and run it. The compiler version of Solidity is ***version:0.4.24+commit.e67f0147.Emscripten.clang.*** We found that this version is stable after many experimentations.



```
pragma solidity ^0.4.24;

contract RealIDService {

    string driverId;
    string address1;
    string address2;



    function setId(string _driverId, string _address1,string _address2 ) public {
        driverId = _driverId;
        address1= _address1;
        address2 = _address2;

    }

    function getId() public constant returns (string,string,string) {
        return (driverId, address1, address2);

    }

}
```

Current
version:0.4.24+commit.e67f0147.Emscripten.clang

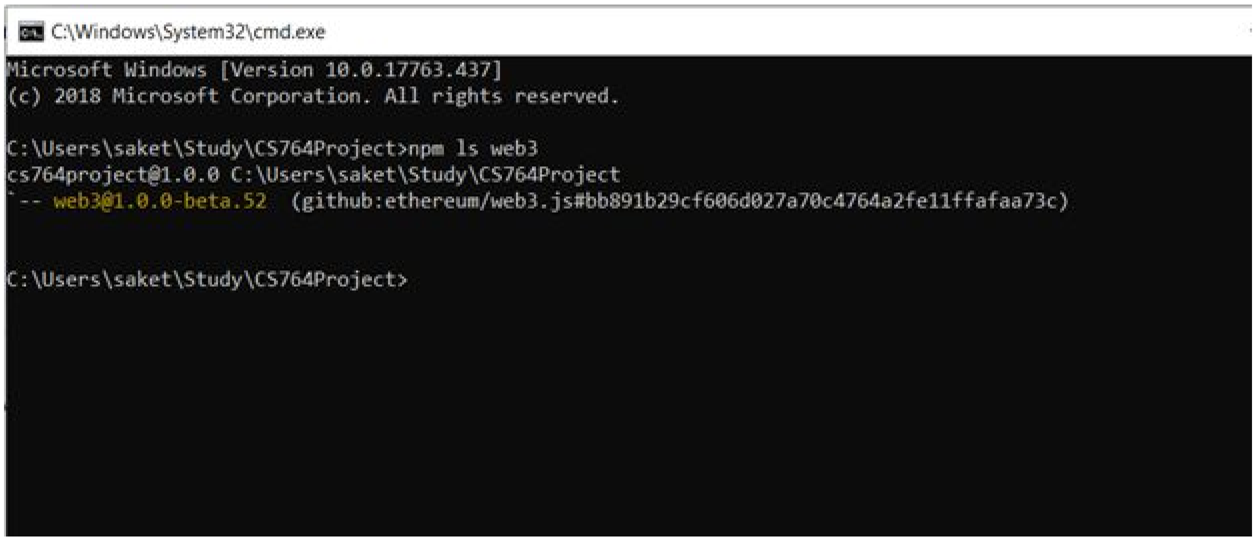
Static Analysis raised 4 warning(s) that requires your attention.
Click here to show the warning(s).

RealIDService

```
- Welcome to Remix v0.7.6 -

You can use this terminal for:

 • Checking transactions details and start debugging.
 • Running JavaScript scripts. The following libraries are accessible:

   ◦ web3 version 1.0.0
   ◦ ethers.js
   ◦ swarmgw
   ◦ compilers - contains currently loaded compiler
 • Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be inclu
   ded and run from a JavaScript script.
 • Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.
```

Web3.js

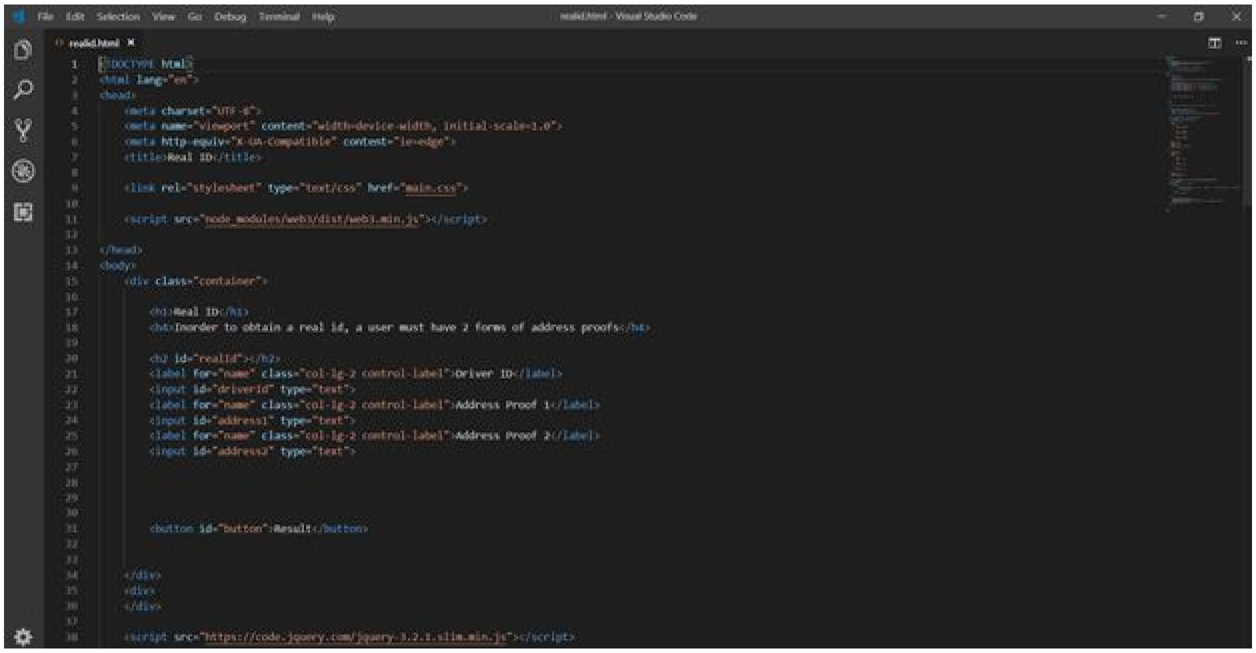
web3.js is a collection of libraries which allow you to interact with a local or remote Ethereum node, using an HTTP, WebSocket or IPC connection.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17763.437]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\saket\Study\CS764Project>npm ls web3
cs764project@1.0.0 C:\Users\saket\Study\CS764Project
`-- web3@1.0.0-beta.52  (github:ethereum/web3.js#bb891b29cf606d027a70c4764a2fe11ffafaa73c)


C:\Users\saket\Study\CS764Project>
```

Visual Studio

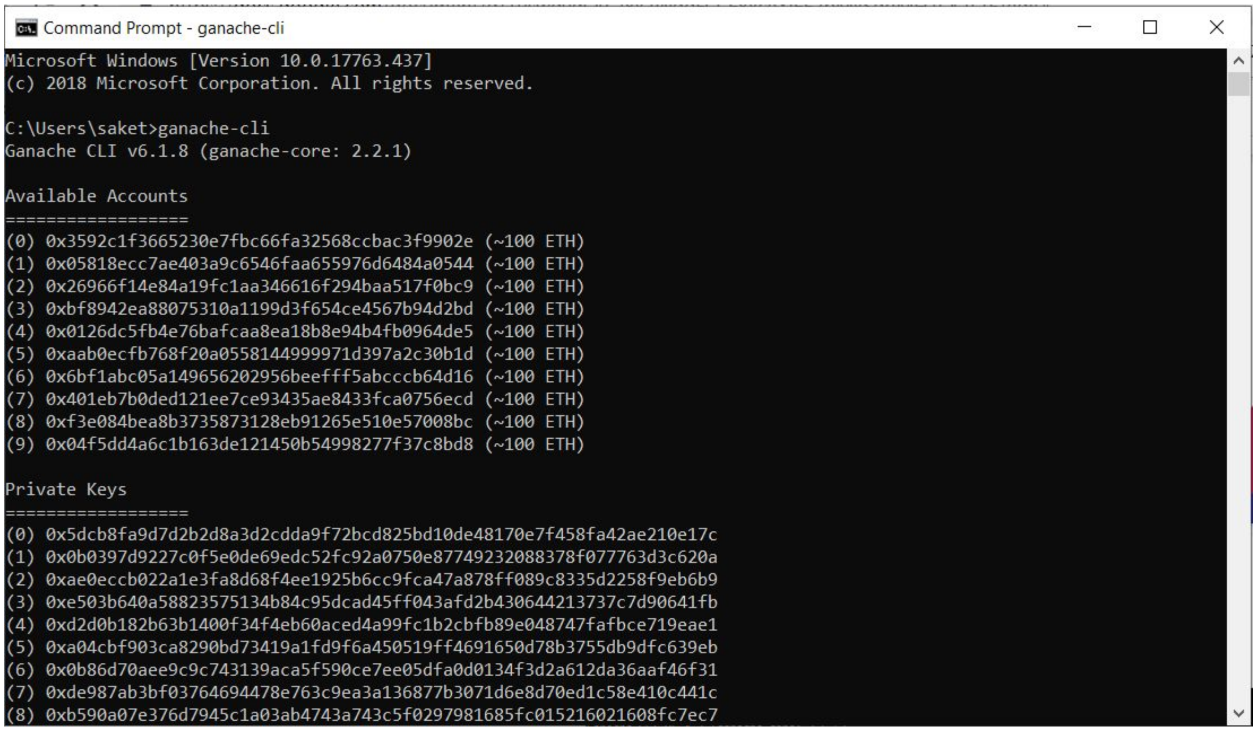
Used it to edit all the HTML and JavaScript files. Any IDE of your comfort can be used instead on the mentioned IDE.


File Edit Selection View Go Debug Terminal Help
realid.html - Visual Studio Code
realid.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Real ID</title>

    <link rel="stylesheet" type="text/css" href="main.css">

    <script src="node_modules/web3/dist/web3.min.js"></script>

</head>
<body>
    <div class="container">

        <h1>Real ID</h1>
        <h4>Inorder to obtain a real id, a user must have 2 forms of address proofs</h4>

        <h2 id="realId"></h2>
        <label for="name" class="col-lg-2 control-label">Driver ID</label>
        <input id="driverId" type="text">
        <label for="name" class="col-lg-2 control-label">Address Proof 1</label>
        <input id="address1" type="text">
        <label for="name" class="col-lg-2 control-label">Address Proof 2</label>
        <input id="address2" type="text">



        <button id="button">Result</button>


    </div>
    <div>
    </div>

    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
```

Environment Setup Steps:

1. Install Node.js on your system. Recommended version 8.11.3.
2. Install Node Package Manager npm. Recommended version 5.6.0.
3. Install Ganache client by following the command **npm install -g ganache-cli** on command prompt.
4. Start you ganache by running the command **ganache-cli** on command prompt. This will initiate your ganache and will generate public and private keys and runs the blockchain.



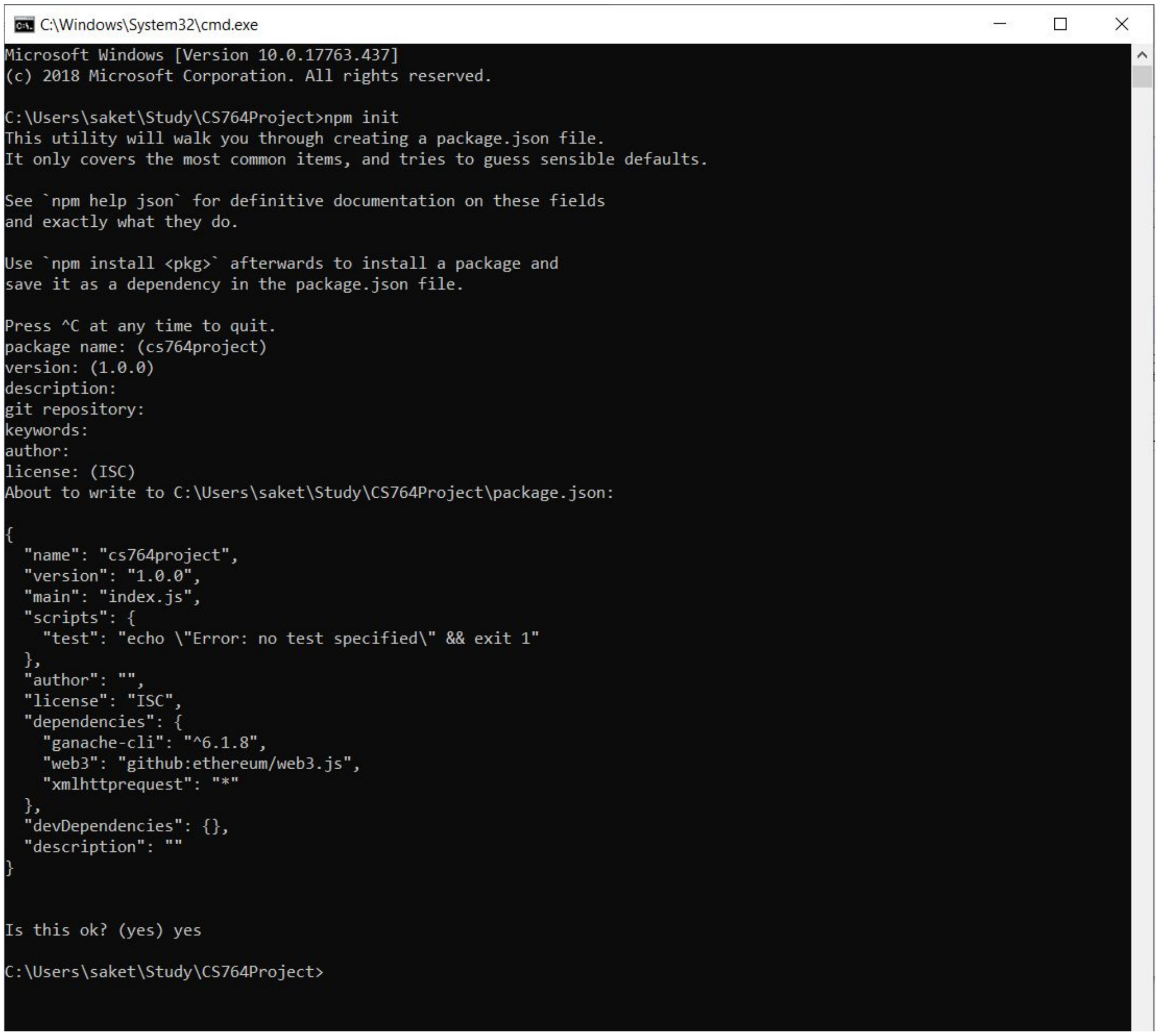
```
Microsoft Windows [Version 10.0.17763.437]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\saket>ganache-cli
Ganache CLI v6.1.8 (ganache-core: 2.2.1)

Available Accounts
==================
(0) 0x3592c1f3665230e7fbc66fa32568ccbac3f9902e (~100 ETH)
(1) 0x05818ecc7ae403a9c6546faa655976d6484a0544 (~100 ETH)
(2) 0x26966f14e84a19fc1aa346616f294baa517f0bc9 (~100 ETH)
(3) 0xbf8942ea88075310a1199d3f654ce4567b94d2bd (~100 ETH)
(4) 0x0126dc5fb4e76bafcaa8ea18b8e94b4fb0964de5 (~100 ETH)
(5) 0xaab0ecfb768f20a0558144999971d397a2c30b1d (~100 ETH)
(6) 0x6bf1abc05a149656202956beefff5abcccb64d16 (~100 ETH)
(7) 0x401eb7b0ded121ee7ce93435ae8433fca0756ecd (~100 ETH)
(8) 0xf3e084bea8b3735873128eb91265e510e57008bc (~100 ETH)
(9) 0x04f5dd4a6c1b163de121450b54998277f37c8bd8 (~100 ETH)

Private Keys
==================
(0) 0x5dcb8fa9d7d2b2d8a3d2cdda9f72bcd825bd10de48170e7f458fa42ae210e17c
(1) 0x0b0397d9227c0f5e0de69edc52fc92a0750e87749232088378f077763d3c620a
(2) 0xae0eccb022a1e3fa8d68f4ee1925b6cc9fca47a878ff089c8335d2258f9eb6b9
(3) 0xe503b640a58823575134b84c95dcad45ff043afd2b430644213737c7d90641fb
(4) 0xd2d0b182b63b1400f34f4eb60aced4a99fc1b2cbfb89e048747fafbce719eae1
(5) 0xa04cbf903ca8290bd73419a1fd9f6a450519ff4691650d78b3755db9dfc639eb
(6) 0x0b86d70aee9c9c743139aca5f590ce7ee05dfa0d0134f3d2a612da36aaf46f31
(7) 0xde987ab3bf03764694478e763c9ea3a136877b3071d6e8d70ed1c58e410c441c
(8) 0xb590a07e376d7945c1a03ab4743a743c5f0297981685fc015216021608fc7ec7
```

5. Create a directory for the project.
   a. Cd Project
   b. Mkdir CS764Project
   c. Cd CS764Project

The following steps take you to the project folder.

6. Initiate the node package manager in the project directory path by running the following command in command prompt. Follow the prompts and give the command "yes" once all the information is correct.

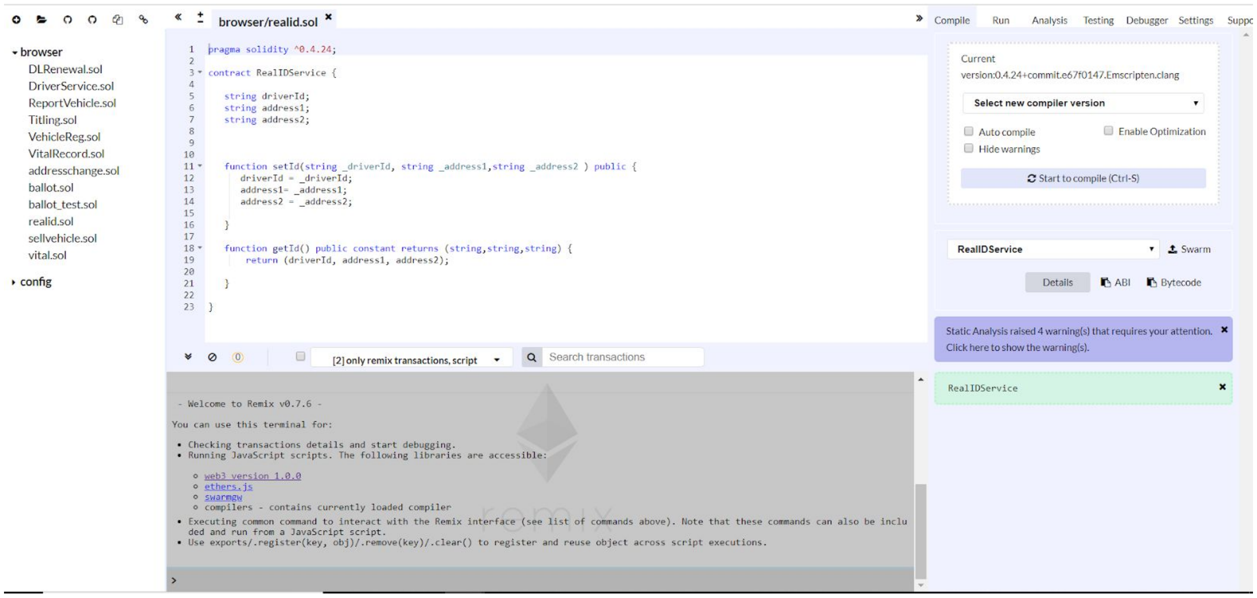
"npm init"



C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.17763.437]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\saket\Study\CS764Project>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (cs764project)
version: (1.0.0)
description:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\saket\Study\CS764Project\package.json:

{
  "name": "cs764project",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "ganache-cli": "^6.1.8",
    "web3": "github:ethereum/web3.js",
    "xmlhttprequest": "*"
  },
  "devDependencies": {},
  "description": ""
}


Is this ok? (yes) yes

C:\Users\saket\Study\CS764Project>
```

7. Open https://remix.ethereum.org to write your smart contract based on the requirement. Set the compiler to 0.4.24 version and then compile. Once there are no errors you can deploy your contract under the run tab on the right top corner of the browser.



```
pragma solidity ^0.4.24;

contract RealIDService {

    string driverId;
    string address1;
    string address2;


    function setId(string _driverId, string _address1,string _address2 ) public {
        driverId = _driverId;
        address1= _address1;
        address2 = _address2;

    }

    function getId() public constant returns (string,string,string) {
        return (driverId, address1, address2);

    }

}
```

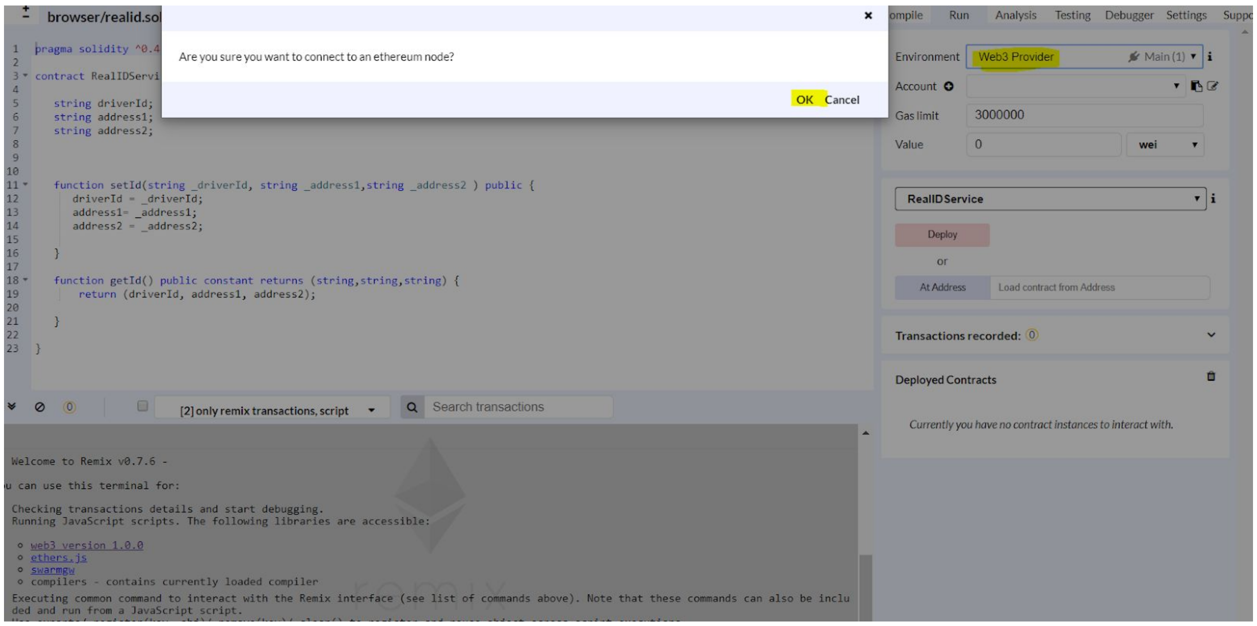
Current
version:0.4.24+commit.e67f0147.Emscripten.clang

Static Analysis raised 4 warning(s) that requires your attention.
Click here to show the warning(s).

RealIDService

```
- Welcome to Remix v0.7.6 -

You can use this terminal for:

• Checking transactions details and start debugging.
• Running JavaScript scripts. The following libraries are accessible:
    o web3 version 1.0.0
    o ethers.js
    o swarmgw
    o compilers - contains currently loaded compiler
• Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be inclu
  ded and run from a JavaScript script.
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.
```

Select Web3 Provider and click OK to set the Web3 connection.



Are you sure you want to connect to an ethereum node?

Environment: Web3 Provider — Main (1)
Account:
Gas limit: 3000000
Value: 0 wei

RealIDService

Transactions recorded: 0

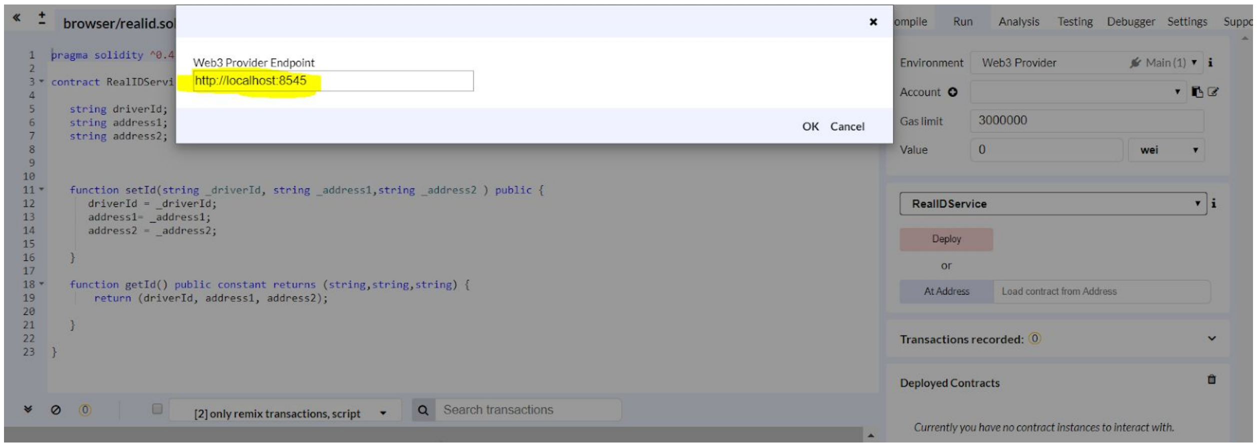
Deployed Contracts

Currently you have no contract instances to interact with.

```
pragma solidity ^0.4

contract RealIDServi

    string driverId;
    string address1;
    string address2;


    function setId(string _driverId, string _address1,string _address2 ) public {
        driverId = _driverId;
        address1= _address1;
        address2 = _address2;

    }

    function getId() public constant returns (string,string,string) {
        return (driverId, address1, address2);

    }

}
```

```
Welcome to Remix v0.7.6 -

u can use this terminal for:

Checking transactions details and start debugging.
Running JavaScript scripts. The following libraries are accessible:

  o web3 version 1.0.0
  o ethers.js
  o swarmgw
  o compilers - contains currently loaded compiler
Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be inclu
ded and run from a JavaScript script.
```

Provide the same localhost name where the ethereum's ganache client is running so it listens on the same endpoint to establish the Web3 connection.



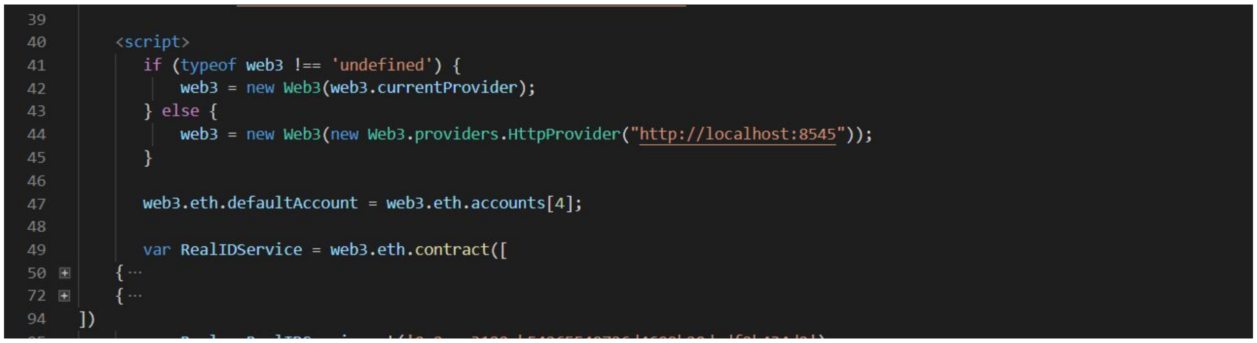
```
pragma solidity ^0.4

contract RealIDServi

    string driverId;
    string address1;
    string address2;



    function setId(string _driverId, string _address1,string _address2 ) public {
        driverId = _driverId;
        address1= _address1;
        address2 = _address2;

    }

    function getId() public constant returns (string,string,string) {
         return (driverId, address1, address2);

    }

}
```

Web3 Provider Endpoint: http://localhost:8545

Environment: Web3 Provider — Main (1)
Account:
Gas limit: 3000000
Value: 0 wei
RealIDService

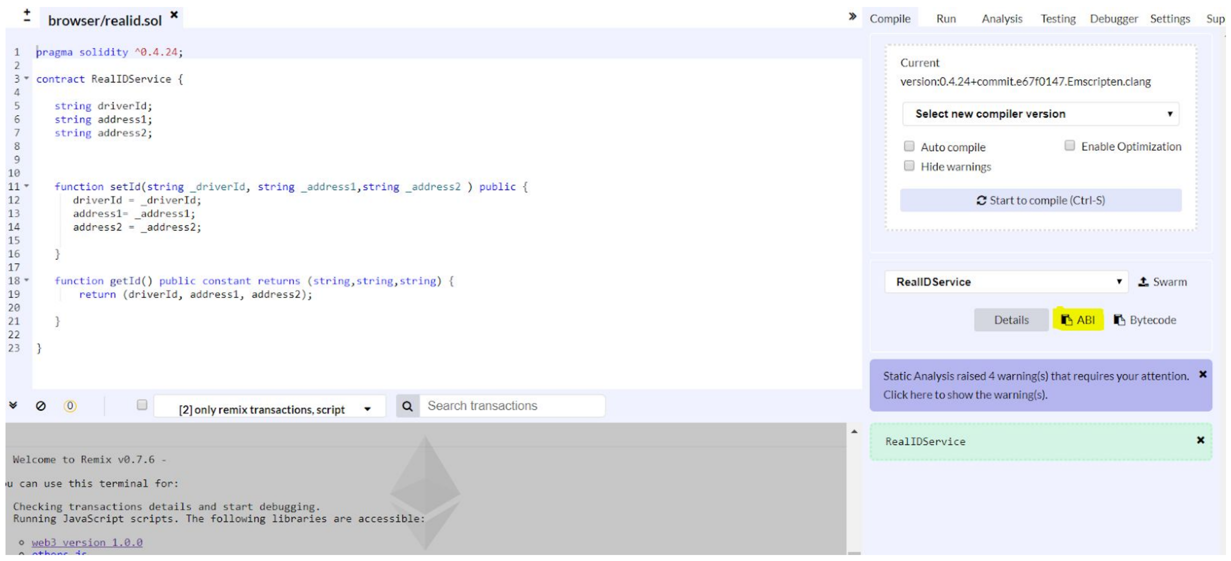
8. Create Home.html and main.css files in Visual studio with the user interface you need. Make sure to save these files in your project directory.

9. Write the Web3 connection inside the index.html file to connect it to the blockchain.



```

    <script>
      if (typeof web3 !== 'undefined') {
          web3 = new Web3(web3.currentProvider);
      } else {
          web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));
      }

      web3.eth.defaultAccount = web3.eth.accounts[4];

      var RealIDService = web3.eth.contract([
    {…
    {…
 ])
```

10. Once the contract is compiled on Remix browser, copy the ABI code from the compiled contract. This must be pasted inside the .html file under the Web3 connection to specify the same application binary interface. Below are the screenshots of the ABI code pasted inside the Web3 provider.



```
pragma solidity ^0.4.24;

contract RealIDService {

    string driverId;
    string address1;
    string address2;



    function setId(string _driverId, string _address1,string _address2 ) public {
        driverId = _driverId;
        address1= _address1;
        address2 = _address2;

    }

    function getId() public constant returns (string,string,string) {
         return (driverId, address1, address2);

    }

}
```

Current
version:0.4.24+commit.e67f0147.Emscripten.clang

Auto compile: [ ]
Enable Optimization: [ ]
Hide warnings: [ ]

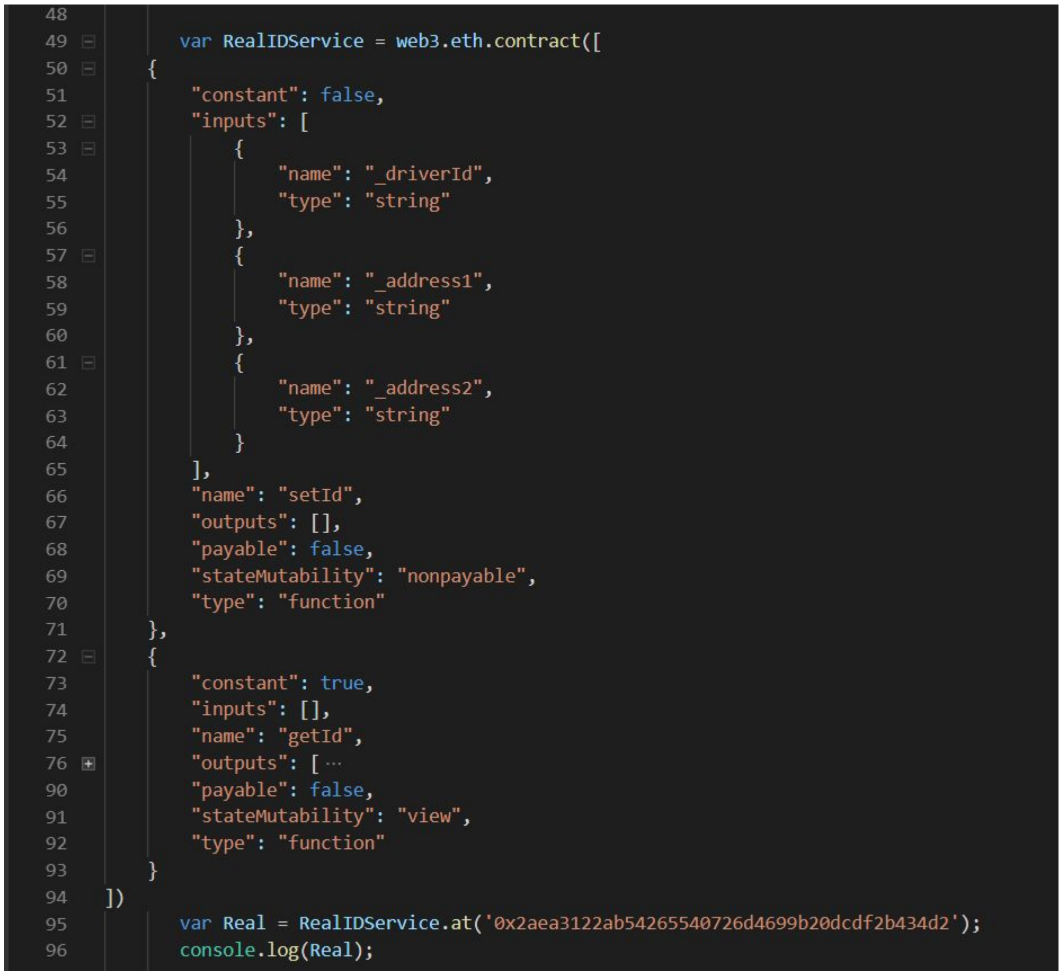
RealIDService

Static Analysis raised 4 warning(s) that requires your attention.
Click here to show the warning(s).

RealIDService

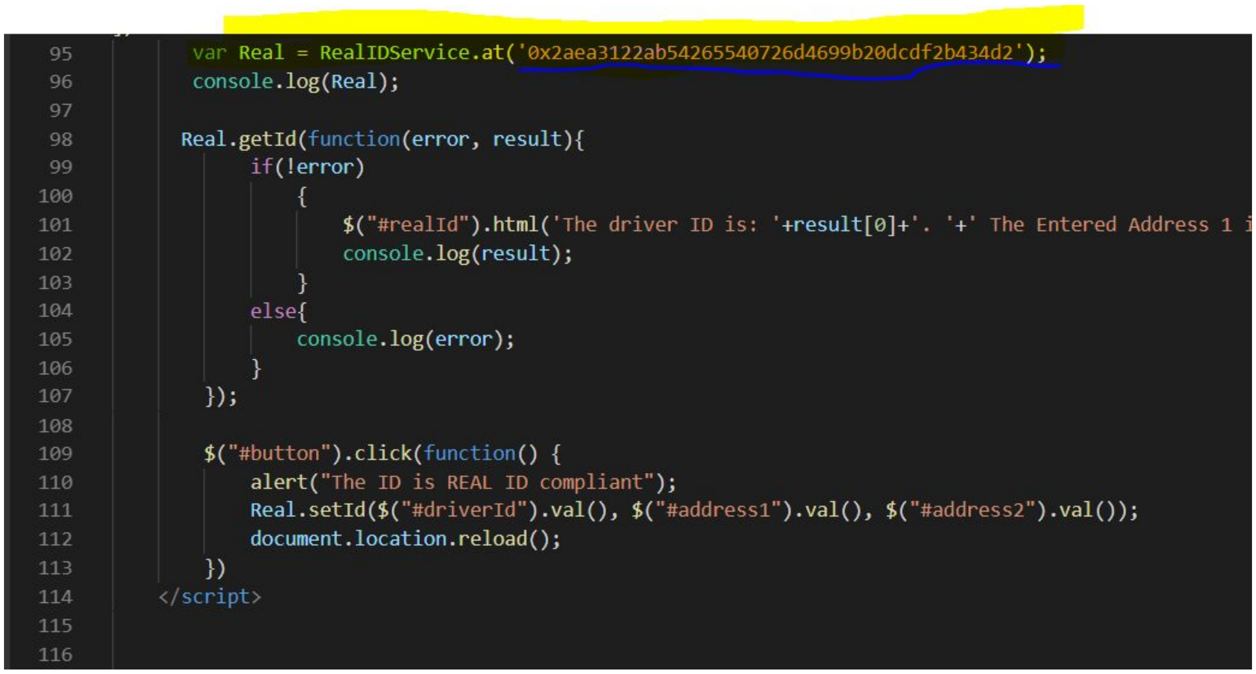
[2] only remix transactions, script

```
Welcome to Remix v0.7.6 -

u can use this terminal for:

Checking transactions details and start debugging.
Running JavaScript scripts. The following libraries are accessible:

o web3 version 1.0.0
```



```

    var RealIDService = web3.eth.contract([
  {
    "constant": false,
    "inputs": [
      {
        "name": "_driverId",
        "type": "string"
      },
      {
        "name": "_address1",
        "type": "string"
      },
      {
        "name": "_address2",
        "type": "string"
      }
    ],
    "name": "setId",
    "outputs": [],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "constant": true,
    "inputs": [],
    "name": "getId",
    "outputs": [ ···
    "payable": false,
    "stateMutability": "view",
    "type": "function"
  }
])
    var Real = RealIDService.at('0x2aea3122ab54265540726d4699b20dcdf2b434d2');
    console.log(Real);
```

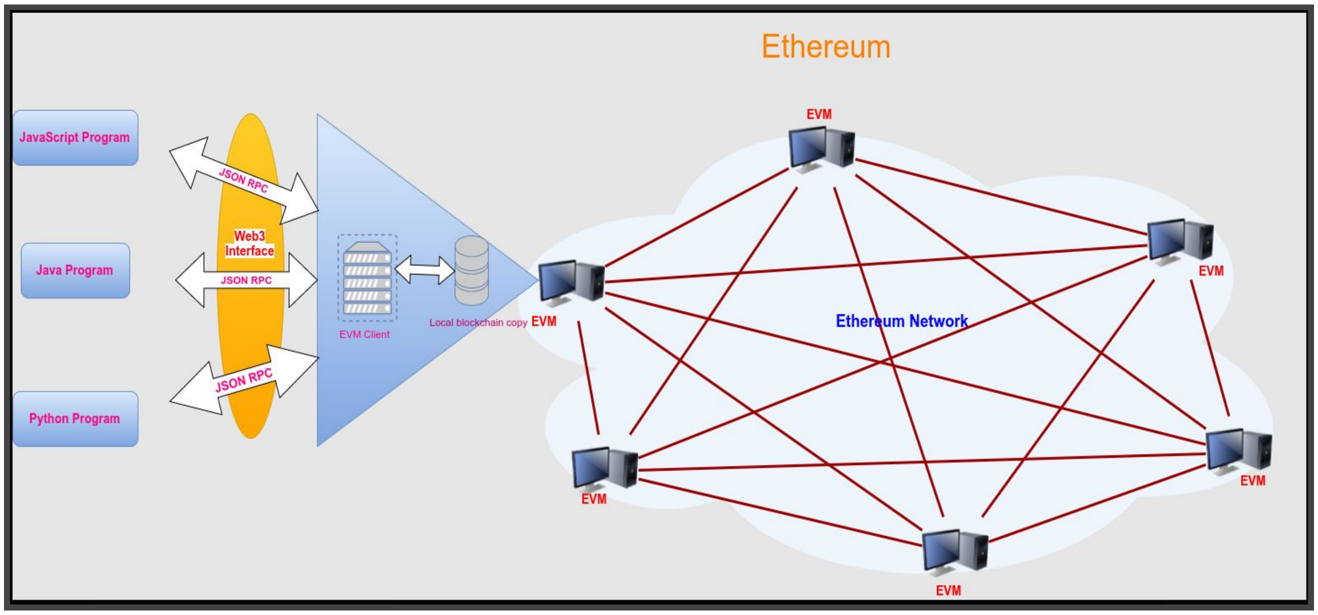
11. Once the contract is deployed, copy the address of the deployed contract and paste in .html file to connect it to the blockchain so the data changes are saved inside the blocks and to retrieve the information from blocks.



```
        var Real = RealIDService.at('0x2aea3122ab54265540726d4699b20dcdf2b434d2');
        console.log(Real);

       Real.getId(function(error, result){
             if(!error)
                 {
                     $("#realId").html('The driver ID is: '+result[0]+'. '+' The Entered Address 1 i
                     console.log(result);
                 }
             else{
                 console.log(error);
             }
         });

         $("#button").click(function() {
             alert("The ID is REAL ID compliant");
             Real.setId($("#driverId").val(), $("#address1").val(), $("#address2").val());
             document.location.reload();
         })
     </script>
```

System Design

Interaction - General Block Diagram



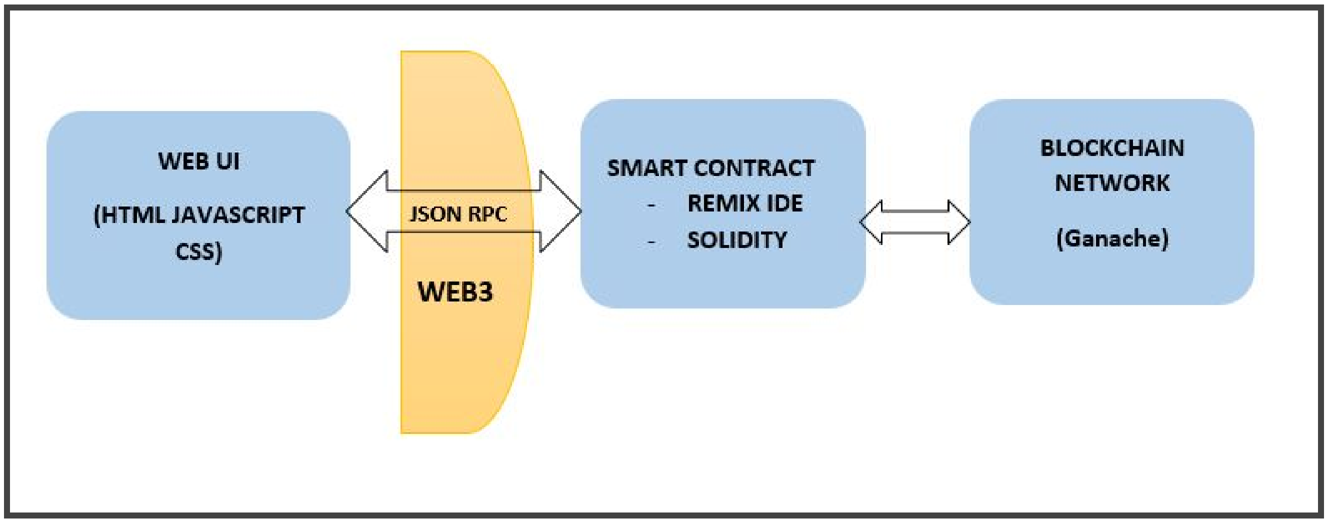
Applications that interact with the Ethereum network are called Distributed Apps (DApps). These applications can be written in 3 programming languages such as Javascript program, Java program, and the Python program.

The programs interact with the Ethernet virtual machine client (EVM client) using the web3 interface. Web3 technologies have libraries which will expose the JSON RPC methods to the Ethereum clients. This will ensure that the interaction between web3 and Ethereum client is established.

The EVM clients are developed using Go programming language or C++ programming language which implements the full functionalities of the EVM. The client always a local copy of the Ethereum blockchain database for backup purpose. The Ethereum network consists of several Ethereum virtual machines which work together and validate the transactions using proof of work consensus algorithm.

Interaction - Implemented Block Diagram

Below is the block diagram of the framework for our implementation



Web UI :

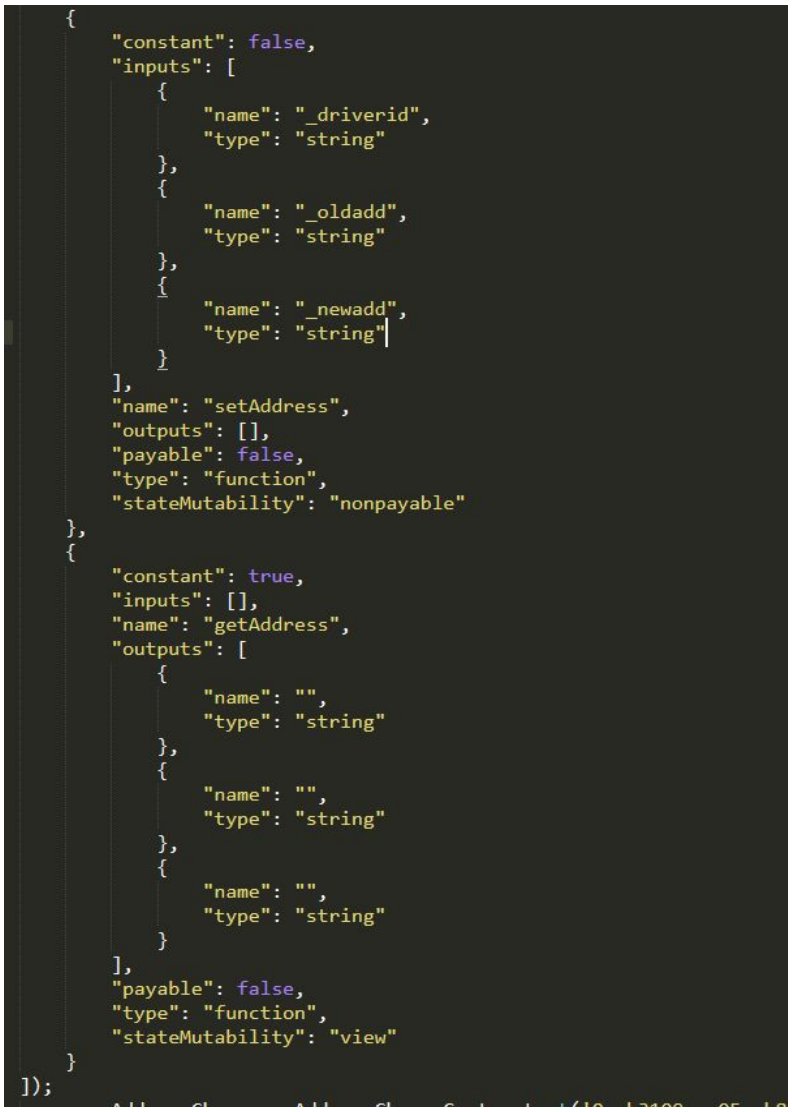
The Web User Interface for end-user interaction is written using 3 web technologies

1. **HTML** (Hypertext Markup Language) - Describes the layout for the web pages
2. **JAVASCRIPT** - Enables the interaction between web pages and the smart contract
3. **CSS** (Cascading Style Sheets) - Describes the styling and the presentation of the content on the web pages

Web3 Interface :

The Web3 interface enables the interaction with the blockchain network. It consists of a collection of libraries which will help to perform actions such as reading data from the smart contracts, writing data to the smart contracts, creating smart contracts, etc.Web3 speaks to the blockchain network using JSON RPC. RPC is abbreviated as Remote Procedure Call. It is a protocol that one program can use to request a service from another which is in a remote location of the network. In our case, JSON RPC will enable the interaction between Javascript (written in the HTML document) and the smart contract written in solidity language. For this interaction to work, we will have the copy the **ABI** (**Application Binary Interface**) from the REMIX IDE, which is obtained after deploying the smart contract and pasting it in the Javascript code written for web3 interaction.

Sample ABI:



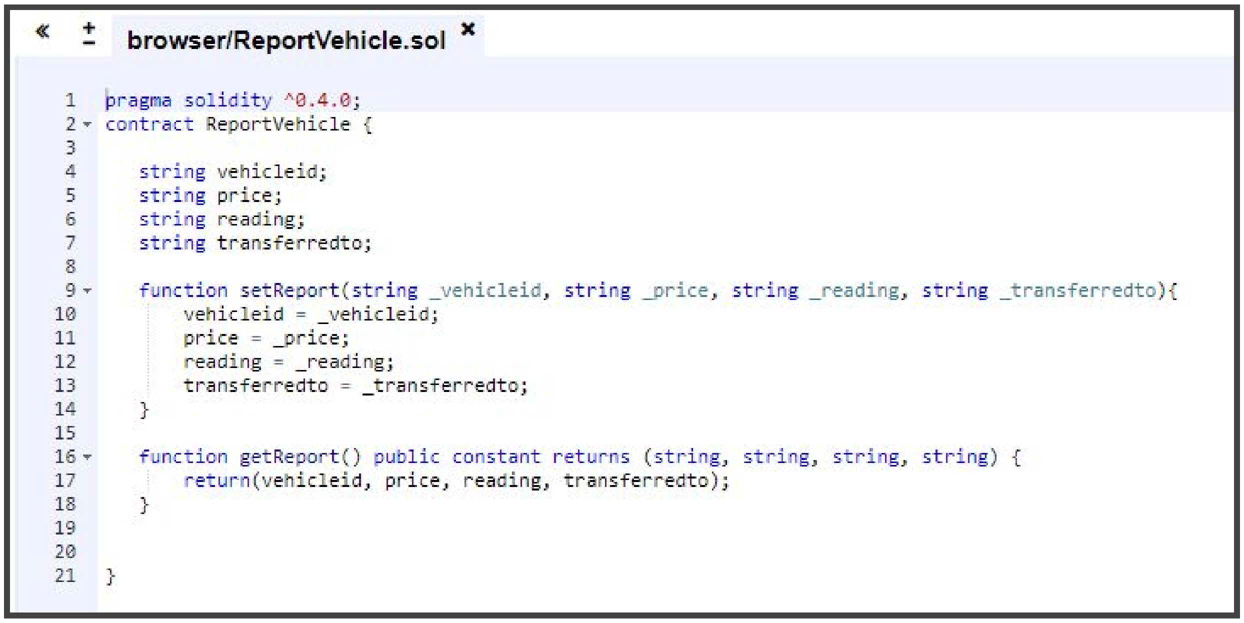
```
    {
        "constant": false,
        "inputs": [
            {
                "name": "_driverid",
                "type": "string"
            },
            {
                "name": "_oldadd",
                "type": "string"
            },
            {
                "name": "_newadd",
                "type": "string"
            }
        ],
        "name": "setAddress",
        "outputs": [],
        "payable": false,
        "type": "function",
        "stateMutability": "nonpayable"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getAddress",
        "outputs": [
            {
                "name": "",
                "type": "string"
            },
            {
                "name": "",
                "type": "string"
            },
            {
                "name": "",
                "type": "string"
            }
        ],
        "payable": false,
        "type": "function",
        "stateMutability": "view"
    }
]);
```

ABI is in JSON format and it basically represents the number of inputs and the type of inputs that the smart contract is expecting from the front-end. This is in JSON format because smart contracts understand the input in the JSON format.

Smart Contract

The smart contract code is written using the Solidity programming language. The IDE used for writing the smart contracts is REMIX. Remix is a Solidity IDE that consists of a set of tools to enable interaction with the blockchain network. It is simple to use because it is a browser-based IDE. It helps to write, deploy and run the solidity smart contracts. Solidity is an object-oriented programming language which was developed for Ethereum projects.

Sample Smart contract:

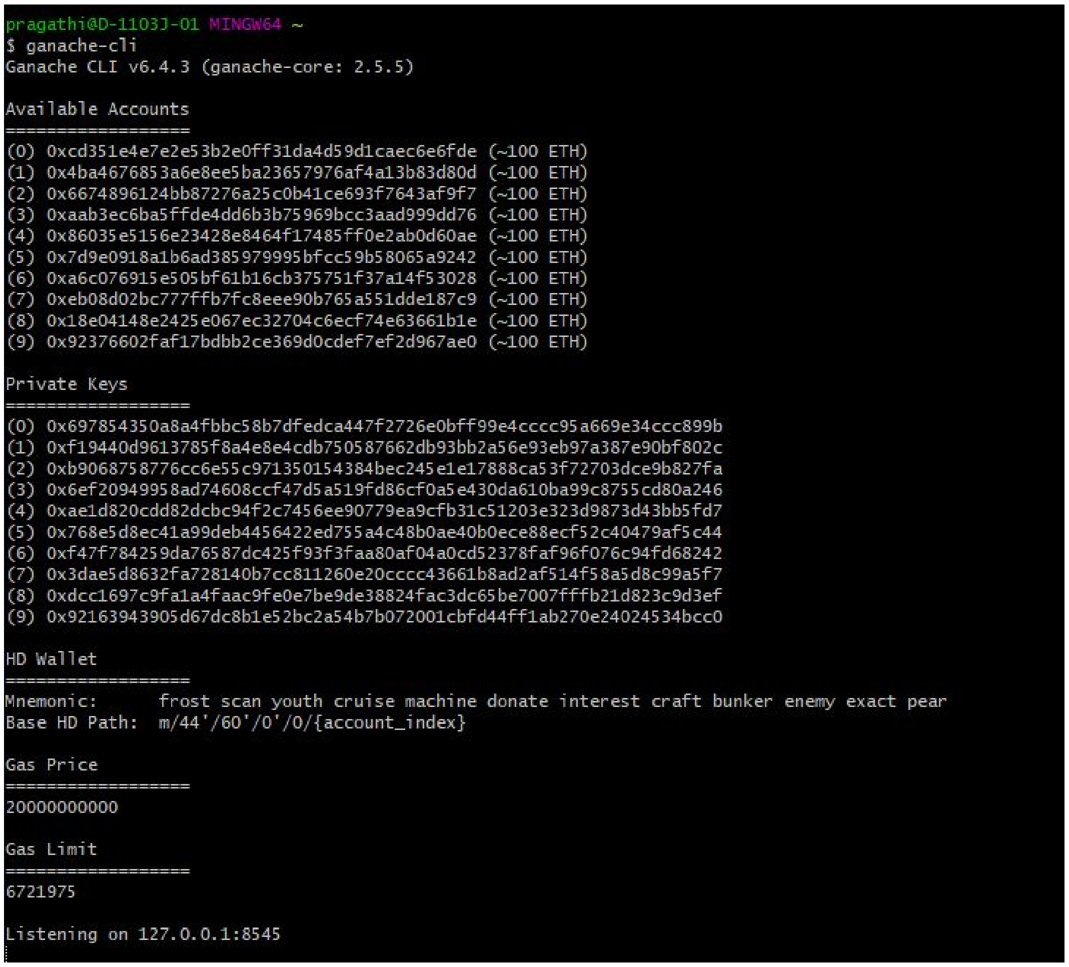


```
pragma solidity ^0.4.0;
contract ReportVehicle {

   string vehicleid;
   string price;
   string reading;
   string transferredto;

   function setReport(string _vehicleid, string _price, string _reading, string _transferredto){
       vehicleid = _vehicleid;
       price = _price;
       reading = _reading;
       transferredto = _transferredto;
   }

   function getReport() public constant returns (string, string, string, string) {
       return(vehicleid, price, reading, transferredto);
   }


}
```

Blockchain Network

In the production level implementation of the blockchain environment, Ethereum virtual machines are used. But in our case, we went for a private environment that replicates the actual blockchain network. For this we used Ganache. Ganache previously known as TestRPC is basically a virtual blockchain that provides us with 10 default Ethereum addresses, the private keys, Gas price, and the Gas limit. Gas or Ether are used for network usage payment on the Ethereum network. So Ganache is basically an Ethereum client which is used for Ethereum development.

Sample Ganache-cli Output :



```
pragathi@D-1103J-01 MINGW64 ~
$ ganache-cli
Ganache CLI v6.4.3 (ganache-core: 2.5.5)

Available Accounts
==================
(0) 0xcd351e4e7e2e53b2e0ff31da4d59d1caec6e6fde (~100 ETH)
(1) 0x4ba4676853a6e8ee5ba23657976af4a13b83d80d (~100 ETH)
(2) 0x6674896124bb87276a25c0b41ce693f7643af9f7 (~100 ETH)
(3) 0xaab3ec6ba5ffde4dd6b3b75969bcc3aad999dd76 (~100 ETH)
(4) 0x86035e5156e23428e8464f17485ff0e2ab0d60ae (~100 ETH)
(5) 0x7d9e0918a1b6ad385979995bfcc59b58065a9242 (~100 ETH)
(6) 0xa6c076915e505bf61b16cb375751f37a14f53028 (~100 ETH)
(7) 0xeb08d02bc777ffb7fc8eee90b765a551dde187c9 (~100 ETH)
(8) 0x18e04148e2425e067ec32704c6ecf74e63661b1e (~100 ETH)
(9) 0x92376602faf17bdbb2ce369d0cdef7ef2d967ae0 (~100 ETH)

Private Keys
==================
(0) 0x697854350a8a4fbbc58b7dfedca447f2726e0bff99e4cccc95a669e34ccc899b
(1) 0xf19440d9613785f8a4e8e4cdb75058766 2db93bb2a56e93eb97a387e90bf802c
(2) 0xb9068758776cc6e55c971350154384bec245e1e17888ca53f72703dce9b827fa
(3) 0x6ef20949958ad74608ccf47d5a519fd86cf0a5e430da610ba99c8755cd80a246
(4) 0xae1d820cdd82dcbc94f2c7456ee90779ea9cfb31c51203e323d9873d43bb5fd7
(5) 0x768e5d8ec41a99deb4456422ed755a4c48b0ae40b0ece88ecf52c40479af5c44
(6) 0xf47f784259da76587dc425f93f3faa80af04a0cd52378faf96f076c94fd68242
(7) 0x3dae5d8632fa728140b7cc811260e20cccc43661b8ad2af514f58a5d8c99a5f7
(8) 0xdcc1697c9fa1a4faac9fe0e7be9de38824fac3dc65be7007fffb21d823c9d3ef
(9) 0x92163943905d67dc8b1e52bc2a54b7b072001cbfd44ff1ab270e24024534bcc0

HD Wallet
==================
Mnemonic:      frost scan youth cruise machine donate interest craft bunker enemy exact pear
Base HD Path:  m/44'/60'/0'/0/{account_index}

Gas Price
==================
20000000000

Gas Limit
==================
6721975

Listening on 127.0.0.1:8545
```

Service Structure Flow

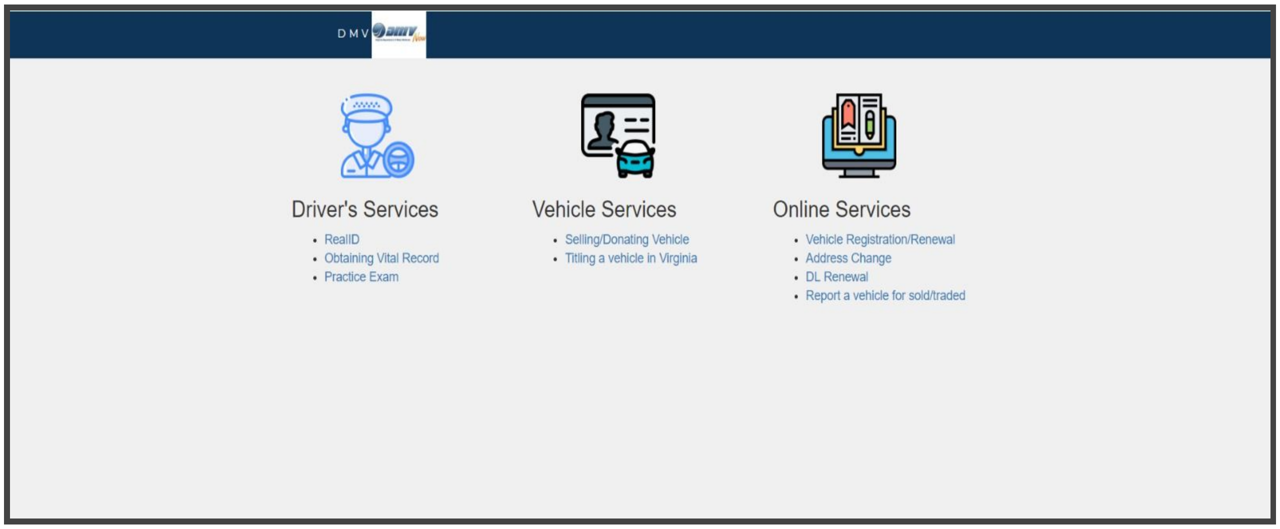
DMV provides several services to its customers: drivers, vehicle owners, police departments, car dealers, etc. The usual implementation of the DMV service requires a third party organization to validate the records of the Driver or to validate the identity of the vehicle such as contacting the registration authority for validating the vehicle's Model number, Brand name, Vehicle owner, etc. But the blockchain implementation eradicates the involvement of any third party organization because blockchain technology provides transparency to both the DMV authorities and the end customers and therefore they can make transactions which are trusted. Anything built on the blockchain in transparent and everyone involved is accountable for whatever they do.

Some of the services that are implemented in this project are listed below

- **Online Services**
  - RealID
  - Obtaining Vital Record
  - Practice Exams
- **Driver's Services**
  - Selling/Donating Vehicle
  - Titling a Vehicle in Virginia
- **Vehicle Services**
  - Vehicle Registration/Renewal
  - Address Change
  - Driver's Licence Renewal
  - Report a Vehicle Sold/Traded

Implementation

The home page of the web application is shown below. It lists all the services provided by DMV.



Online Services

The online services provided are listed below

- RealID
- Obtaining Vital Record
- Practice Exams

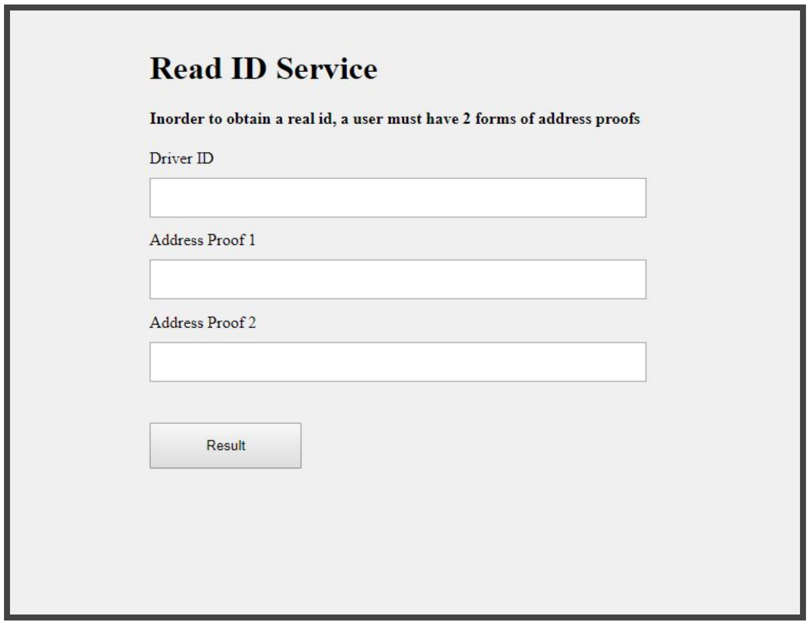
RealID

Functions:

- getId
- setId

Variables:

- realId
- driverId
- address1
- address2



Read ID Service

Inorder to obtain a real id, a user must have 2 forms of address proofs

Driver ID

Address Proof 1

Address Proof 2

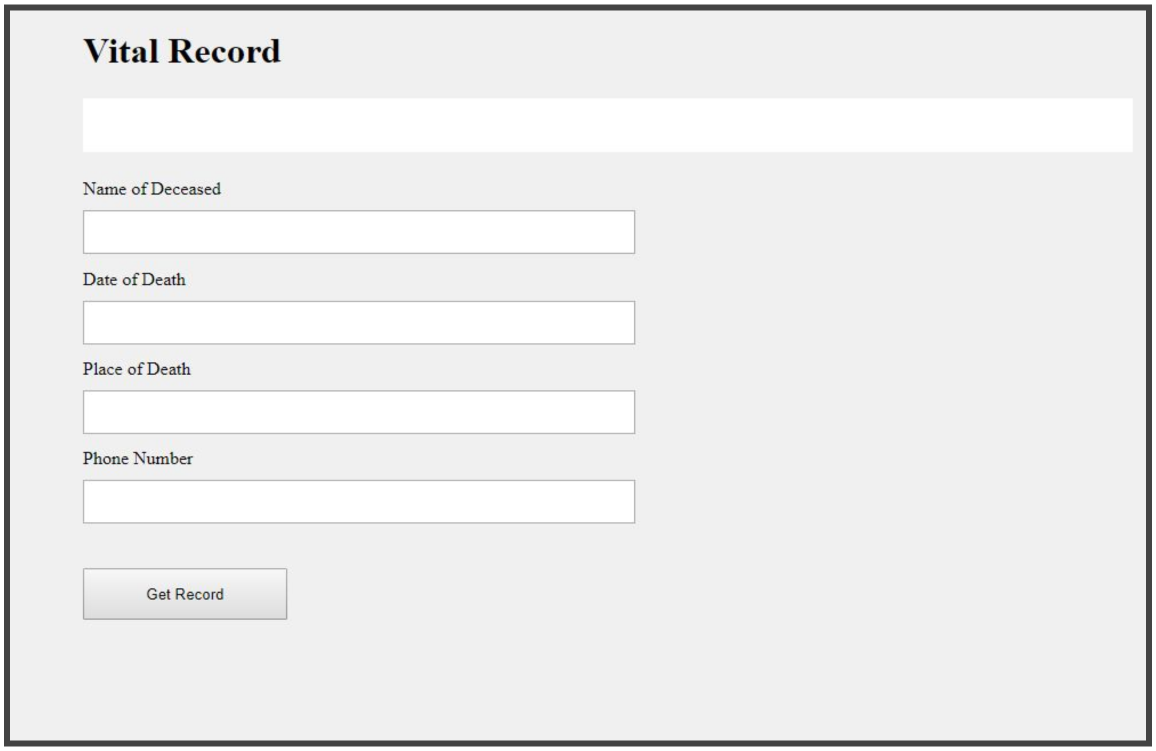
Obtaining Vital Record

Functions:

- getRecord
- setRecord

Variables:

- name
- date
- place
- phone


Vital Record

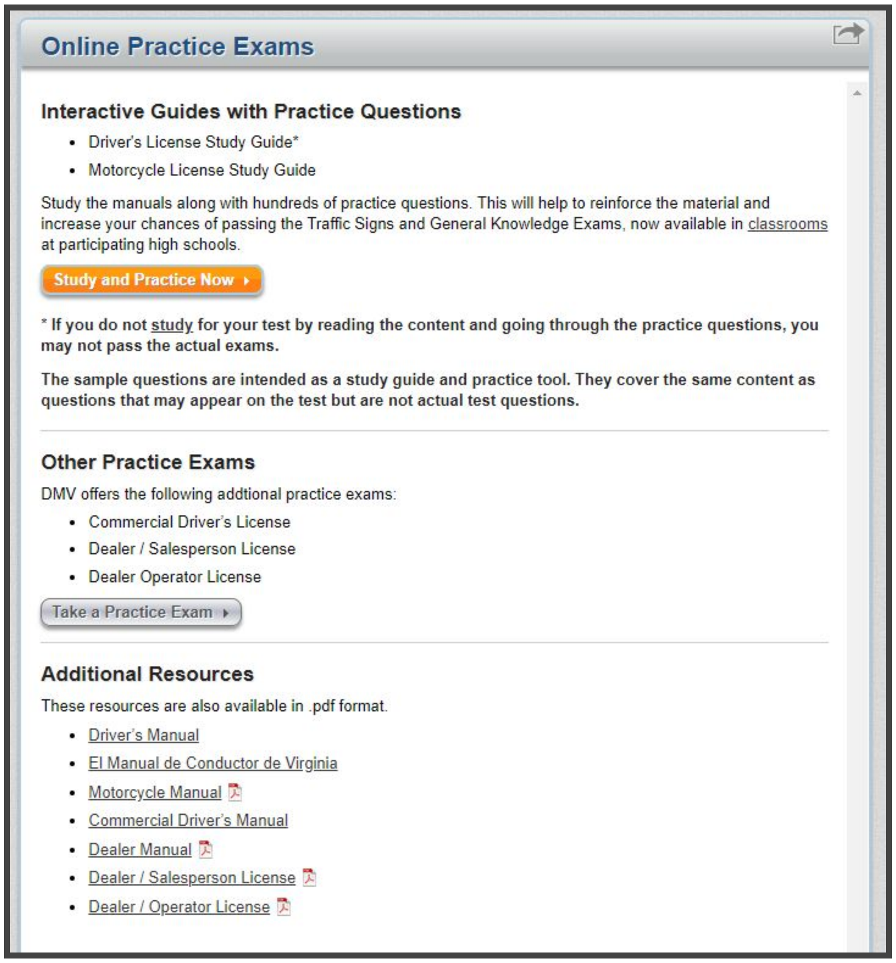
Name of Deceased

Date of Death

Place of Death

Phone Number

Practice Exams


Online Practice Exams

Interactive Guides with Practice Questions

- Driver's License Study Guide*
- Motorcycle License Study Guide

Study the manuals along with hundreds of practice questions. This will help to reinforce the material and increase your chances of passing the Traffic Signs and General Knowledge Exams, now available in classrooms at participating high schools.

* If you do not study for your test by reading the content and going through the practice questions, you may not pass the actual exams.

The sample questions are intended as a study guide and practice tool. They cover the same content as questions that may appear on the test but are not actual test questions.

Other Practice Exams

DMV offers the following addtional practice exams:

- Commercial Driver's License
- Dealer / Salesperson License
- Dealer Operator License

Additional Resources

These resources are also available in .pdf format.

- Driver's Manual
- El Manual de Conductor de Virginia
- Motorcycle Manual
- Commercial Driver's Manual
- Dealer Manual
- Dealer / Salesperson License
- Dealer / Operator License

Driver's Services

The Driver's services provided are listed below

- Selling/Donating Vehicle
- Titling a Vehicle in Virginia

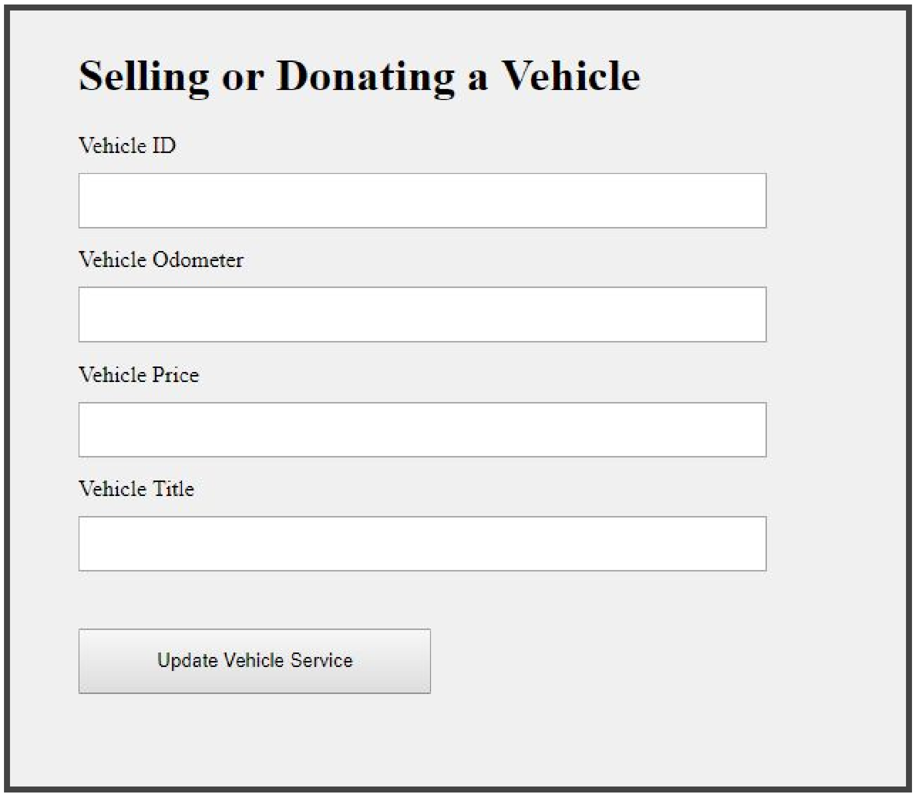
Selling/Donating Vehicle

Functions:

- getVehicle
- setVehicle

Variables:

- vehicleId
- vehicleOdometer
- vehiclePrice
- vehicleTitle



Selling or Donating a Vehicle

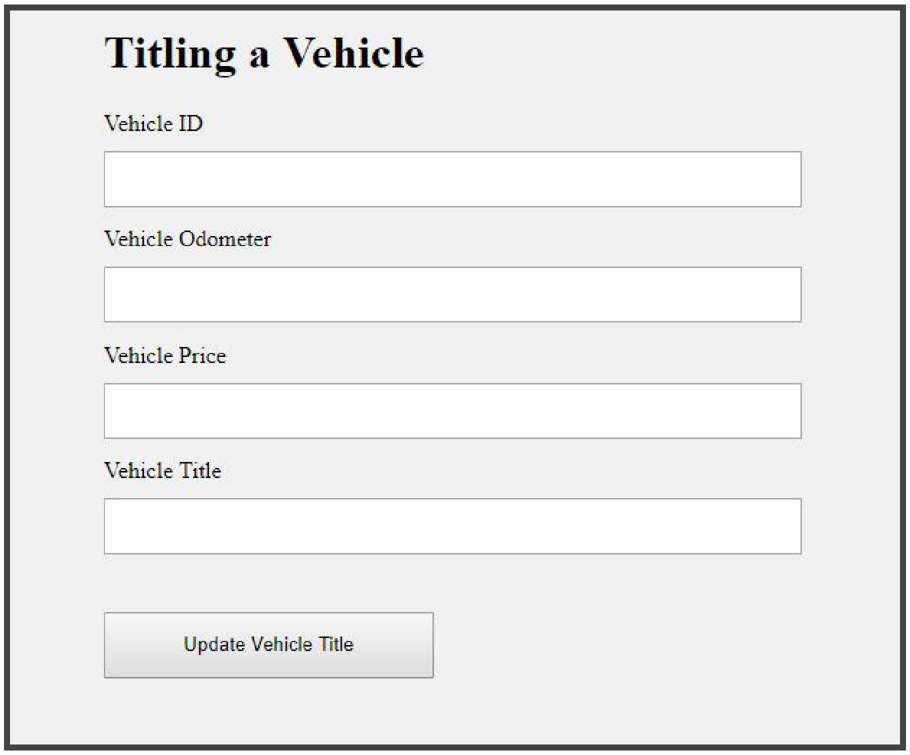
Titling a Vehicle in Virginia

Functions:

- getVehicleTitle
- setVehicleTitle

Variables:

- vehicleId
- vehicleOdometer
- vehiclePrice
- vehicleTitle



Vehicle Services

The Vehicle services provided are listed below

- Vehicle Registration/Renewal
- Address Change
- Driver's Licence Renewal
- Report a Vehicle Sold/Traded

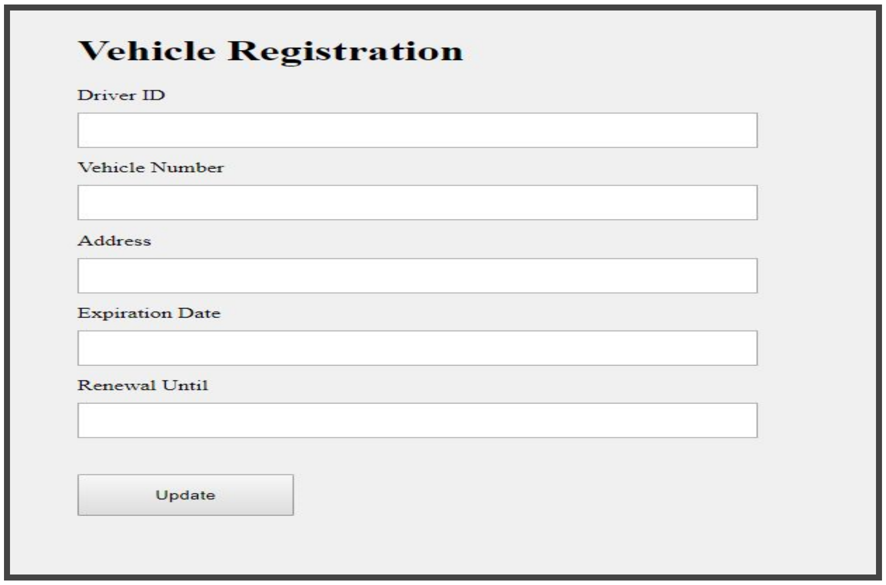
Vehicle Registration/Renewal

Functions:

- getVehicle
- setVehicle

Variables:

- driverId
- vehicleNo
- Address
- expirationDate
- renewalDate



Vehicle Registration

Driver ID

Vehicle Number

Address

Expiration Date

Renewal Until

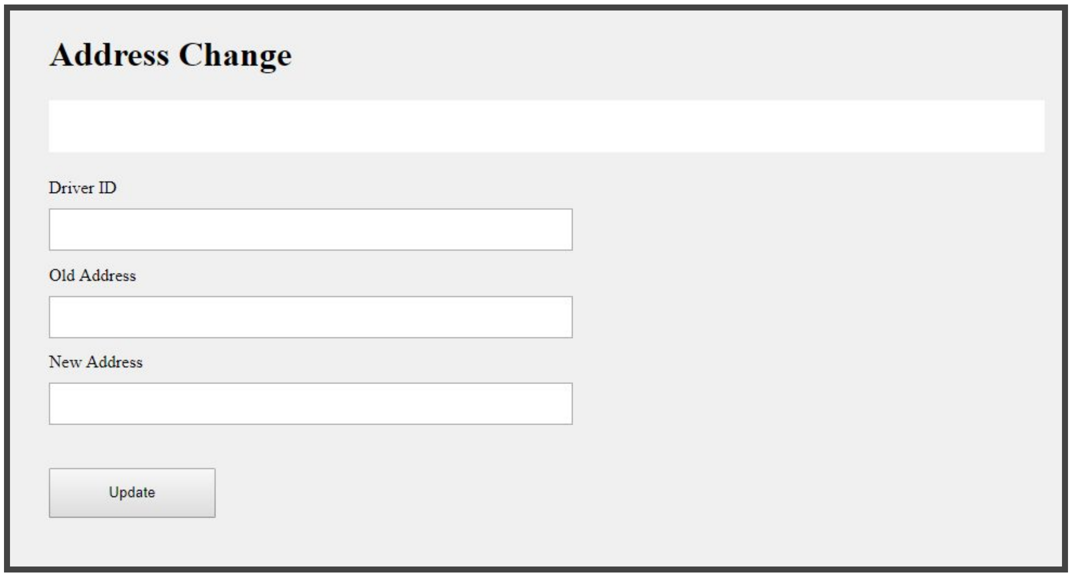
Address Change

Functions:

- getAddress
- setAddress

Variables:

- driverId
- oldadd
- newadd



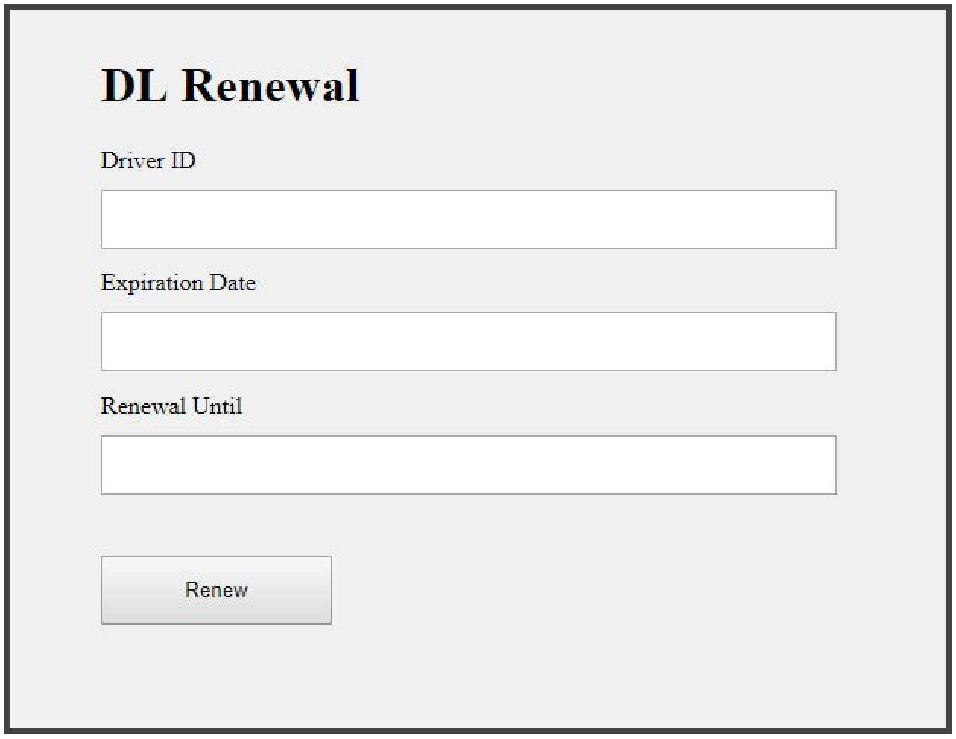
Driver’s Licence Renewal

Functions:

- getRenewal
- setRenewal

Variables:

- driverId
- expirationDate
- newDate



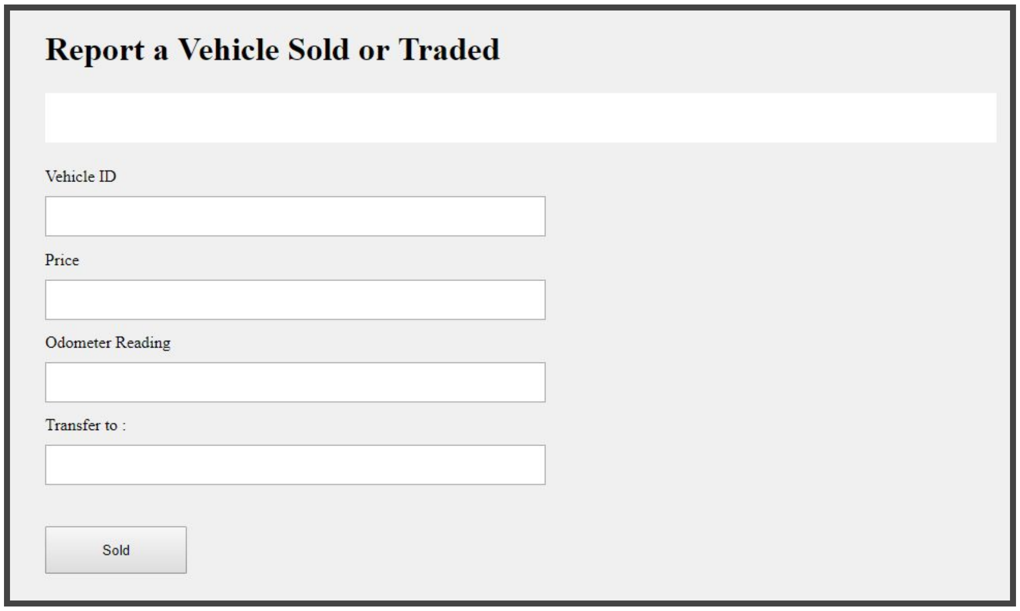
Report a Vehicle Sold/Traded

Functions:

- getReport
- setReport

Variables:

- vehicleid
- price
- reading
- transferredto



Report a Vehicle Sold or Traded

Vehicle ID

Price

Odometer Reading

Transfer to :

Limitations

In terms of Design, we had a hard time finding a technical framework and environment that can work for our requirements. There is no such one place to find all about the setup. Thus, we ended up spending a lot of time to come up with a design and finalize. Finding stable versions of all the tools and testing their interaction was hard. Some versions of tools did not work with some versions of other tools, making the setup unstable. So, we have spent more time on finding stable versions that work together.

In terms of Implementation, for the technical framework, we used limited us to use one blockchain and also default values of Ether balance or gas limit were immutable which led us to run into “run out of gas” error. So as to adapt to a new framework, we did not have much time left to learn something new after finding these limitations. Due to these limitations, we were not able to execute all the features.We have tried our best to put together everything and pulled off what we have learned in the course to build the system.

Conclusion

Within the time frame given, it was very challenging to learn new technical frameworks, design and implement a prototype of Blockchain-based system for DMV. It was really a great learning experience for us. We were able to use the basic knowledge that we gained during the course duration to design the system and understand the requirements. With our current system, more features can be added and multiple blockchains can be used for validation services which we assumed to be existing in our implementation.