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.



2



|  |  |
| --- | --- |
| **Contents** |  |
| [**Acknowledgment**](#page2) | [**2**](#page2) |
| [**Contents**](#page3) | [**3**](#page3) |
| [**Introduction**](#page4) | [**4**](#page4) |
| [**Technical Framework and Environment Setup**](#page4) | [**4**](#page4) |
| [Versions Used:](#page5) | [5](#page5) |
| [Node](#page5) | [5](#page5) |
| [Npm](#page5) | [5](#page5) |
| [Ganache CLI](#page6) | [6](#page6) |
| [Remix Browser](#page7) | [7](#page7) |
| [Web3.js](#page7) | [7](#page7) |
| [Visual Studio](#page8) | [8](#page8) |
| [**Environment Setup Steps:**](#page8) | [**9**](#page8) |
| [**System Design**](#page15) | [**15**](#page15) |
| [Interaction - Implemented Block Diagram](#page16) | [16](#page16) |
| [Web3 Interface :](#page16) | [16](#page16) |
| [Smart Contract](#page17) | [17](#page17) |
| [Blockchain Network](#page18) | [18](#page18) |
| [Service Structure Flow](#page19) | [19](#page19) |
| [**Implementation**](#page20) | [**20**](#page20) |
| [RealID](#page21) | [21](#page21) |
| [Functions:](#page21) | [21](#page21) |
| [Functions:](#page21) | [21](#page21) |
| [Practice Exams](#page22) | [22](#page22) |
| [**Driver’s Services**](#page23) | [**23**](#page23) |
| [Selling/Donating Vehicle](#page24) | [23](#page24) |
| [Functions:](#page24) | [23](#page24) |
| [Titling a Vehicle in Virginia](#page25) | [25](#page25) |
| [Functions:](#page25) | [25](#page25) |
| [**Vehicle Services**](#page26) | [**25**](#page26) |
| [Vehicle Registration/Renewal](#page26) | [26](#page26) |
| [Functions:](#page26) | [26](#page26) |
| 3 |  |



|  |  |
| --- | --- |
| [Address Change](#page27) | [26](#page27) |
| [Functions:](#page27) | [27](#page27) |
| [Driver’s Licence Renewal](#page28) | [27](#page28) |
| [Functions:](#page28) | [27](#page28) |
| [Report a Vehicle Sold/Traded](#page29) | [28](#page29) |
| [Functions:](#page29) | [28](#page29) |
| [**Limitations**](#page30) | [**29**](#page30) |
| [**Conclusion**](#page30) | [**30**](#page30) |

**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
  1. Driver/ ID services
     + - Practice exams
       - Real ID
       - Obtaining a vital record

1. Vehicle services
   * + - Selling/donating a vehicle
       - Titling a vehicle in Virginia

4

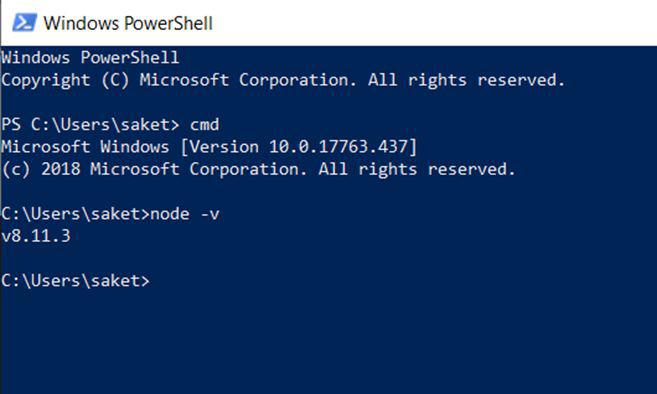


**Technical Framework and Environment Setup**

**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.



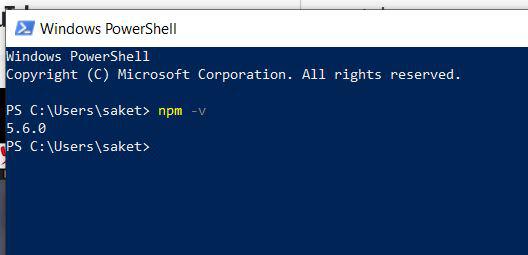
**Npm**

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.

5

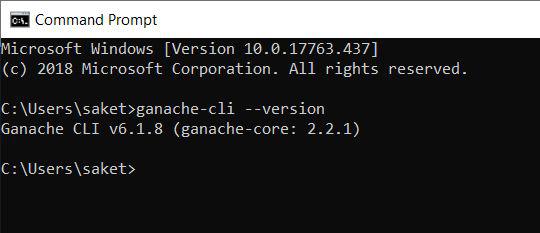




**Ganache CLI**

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***

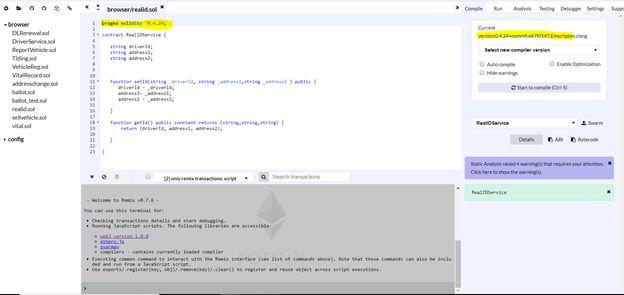


6



**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 aftermany experimentations.

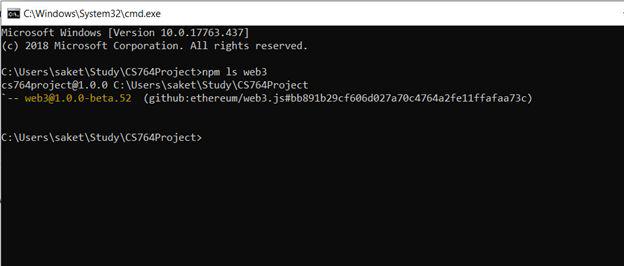


**Web3.js**

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.

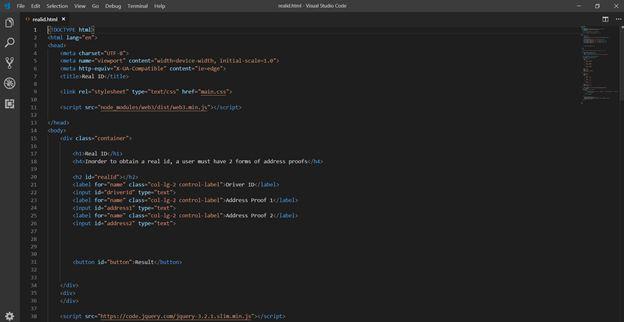
7





**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.

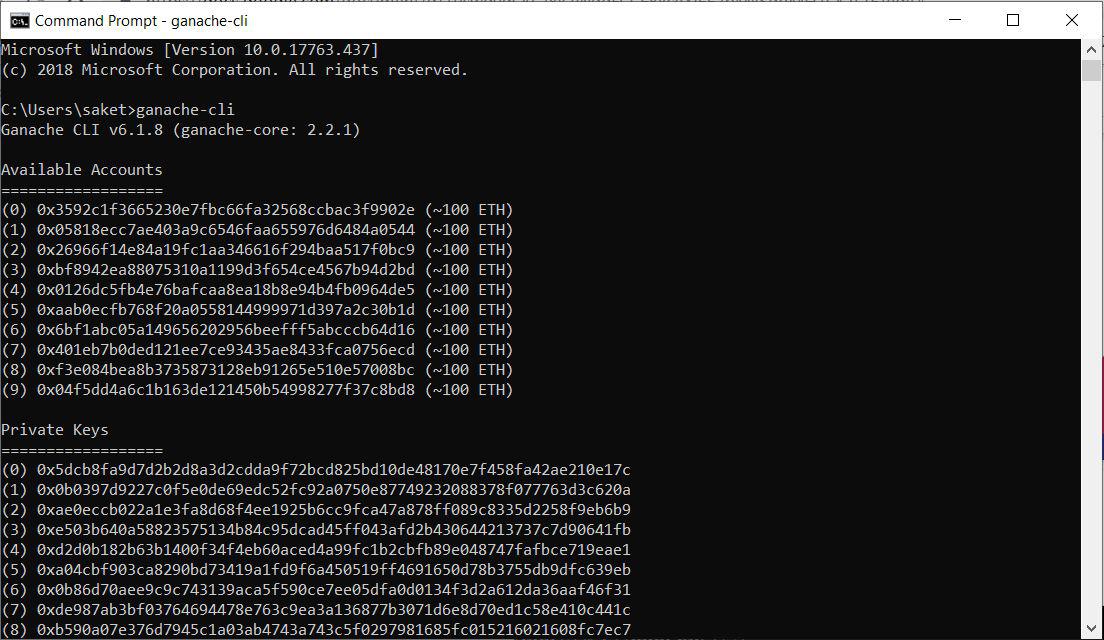


8



**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.



1. Create a directory for the project.
   1. Cd Project
   2. Mkdir CS764Project
   3. Cd CS764Project

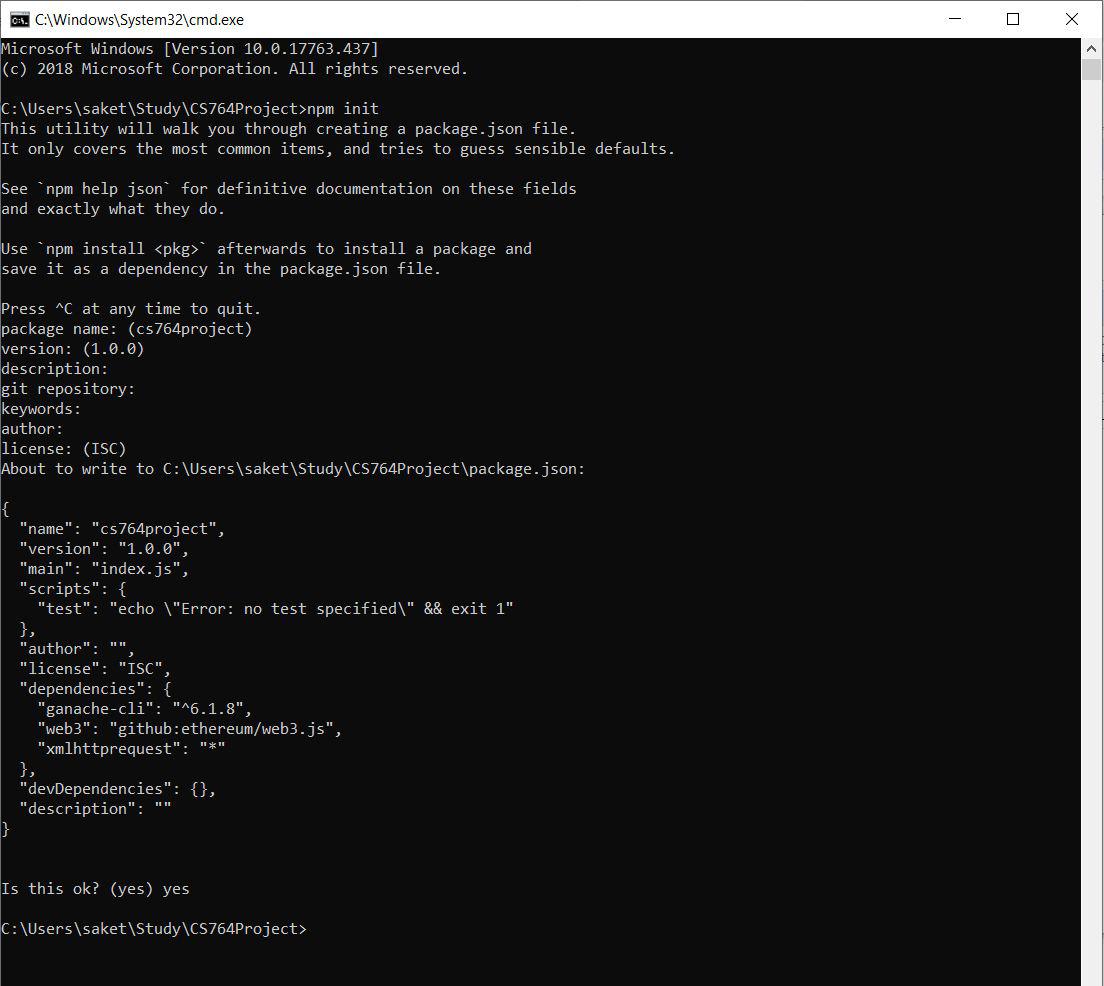
The following steps take you to the project folder.

9



1. 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”***

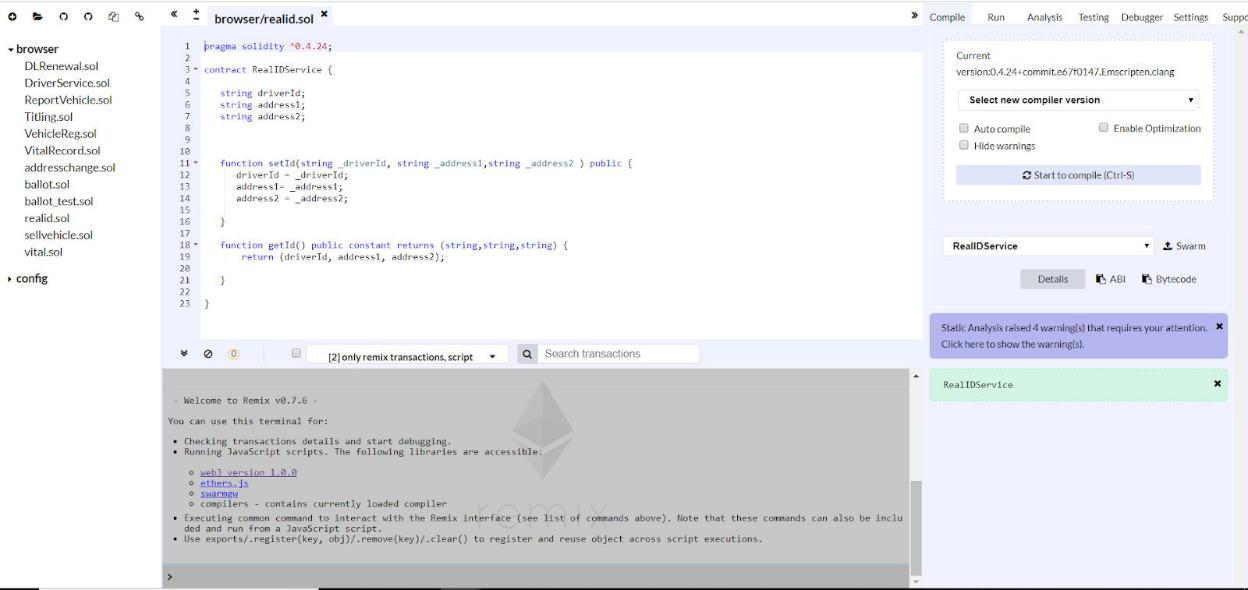


1. Open [https://remix​.ethereum.org](https://remix.ethereum.org/) to write your smart contract based on the requirement. Set the compiler to 0.4.24 version and then compile. Once there are no errors you can deploy your contract under the run tab on the right top corner of the browser.

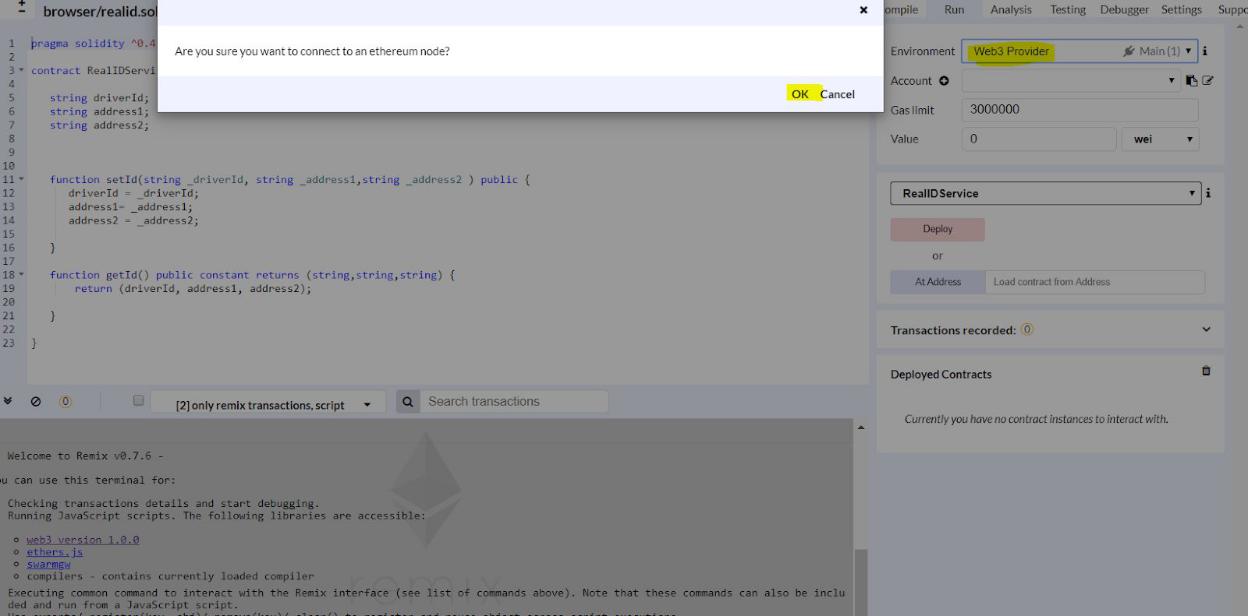


10





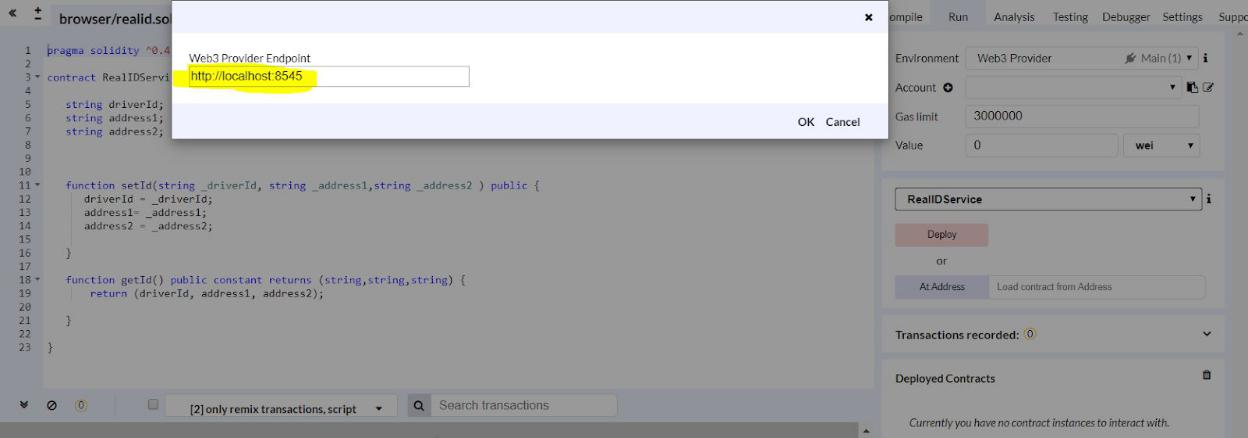
Select Web3 Provider and click OK to set the Web3 connection.



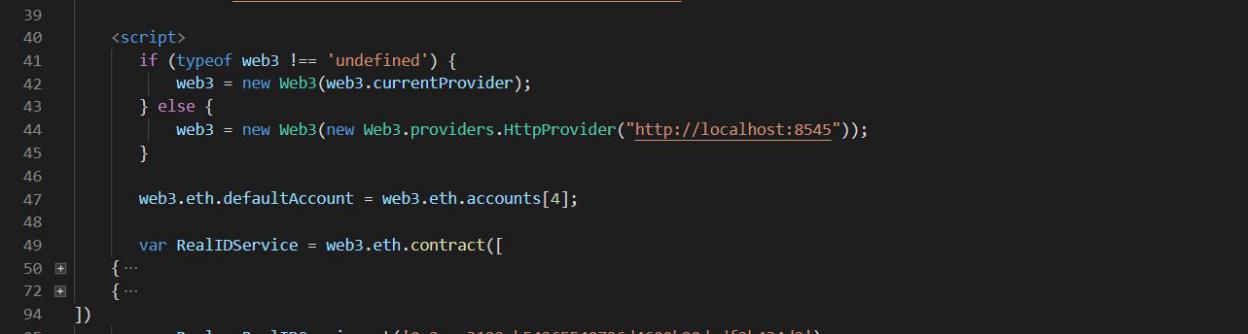
Provide the same localhost name where the ethereum’s ganache client is running so it listens on the same endpoint to establish the Web3 connection.

11





1. 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.
2. Write the Web3 connection inside the index.html file to connect it to the blockchain.

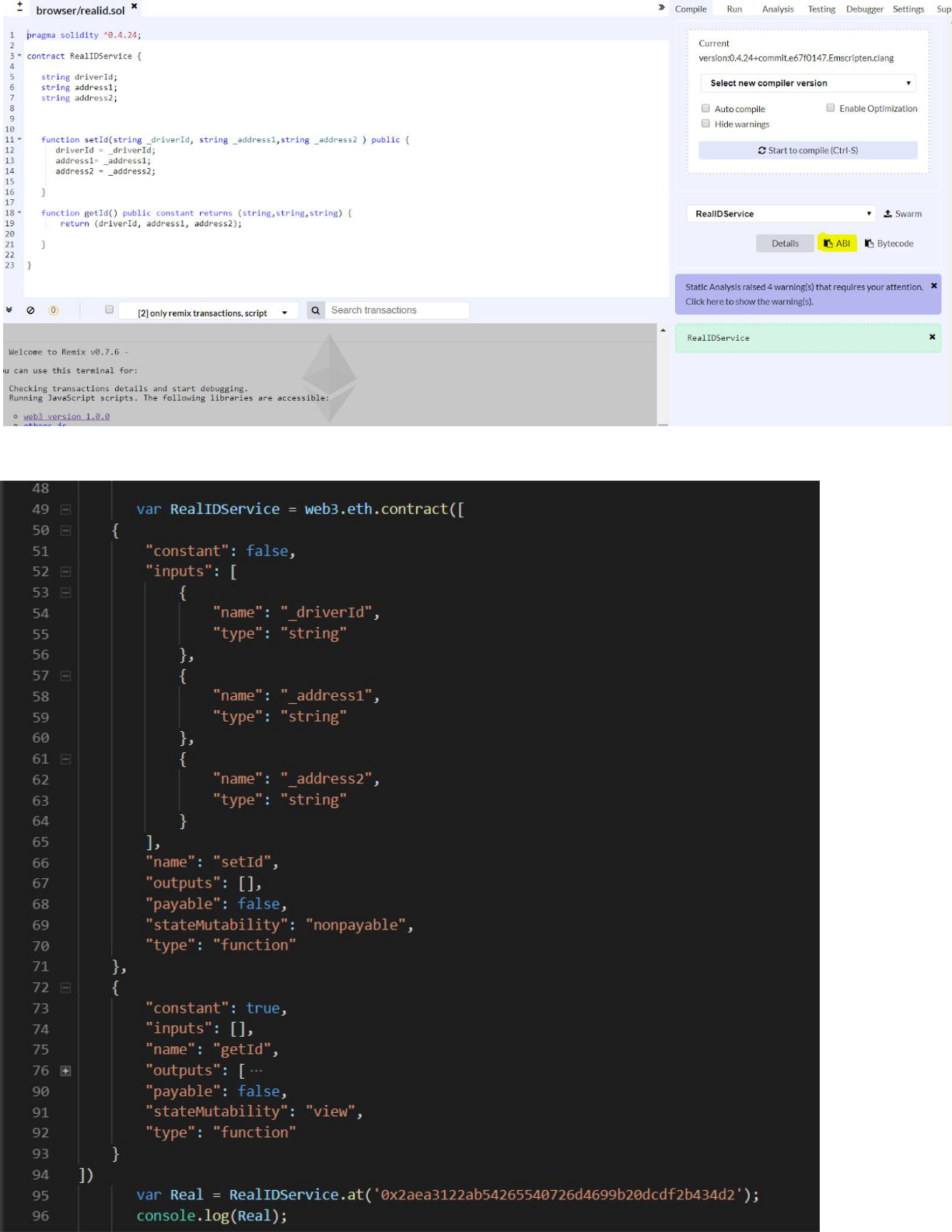


1. Once the contract is compiled on Remix browser, copy the ABI code from the

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.

12

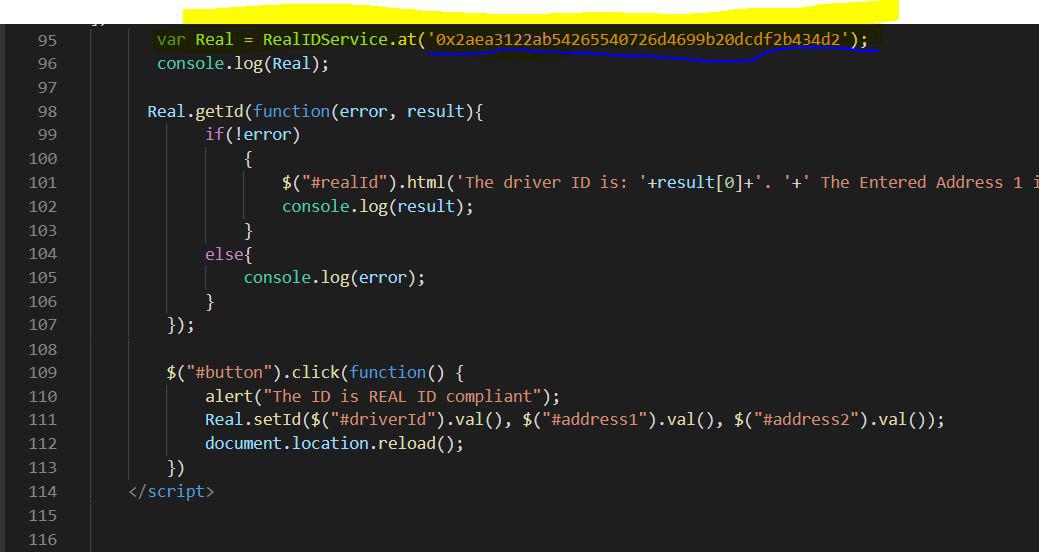




13



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

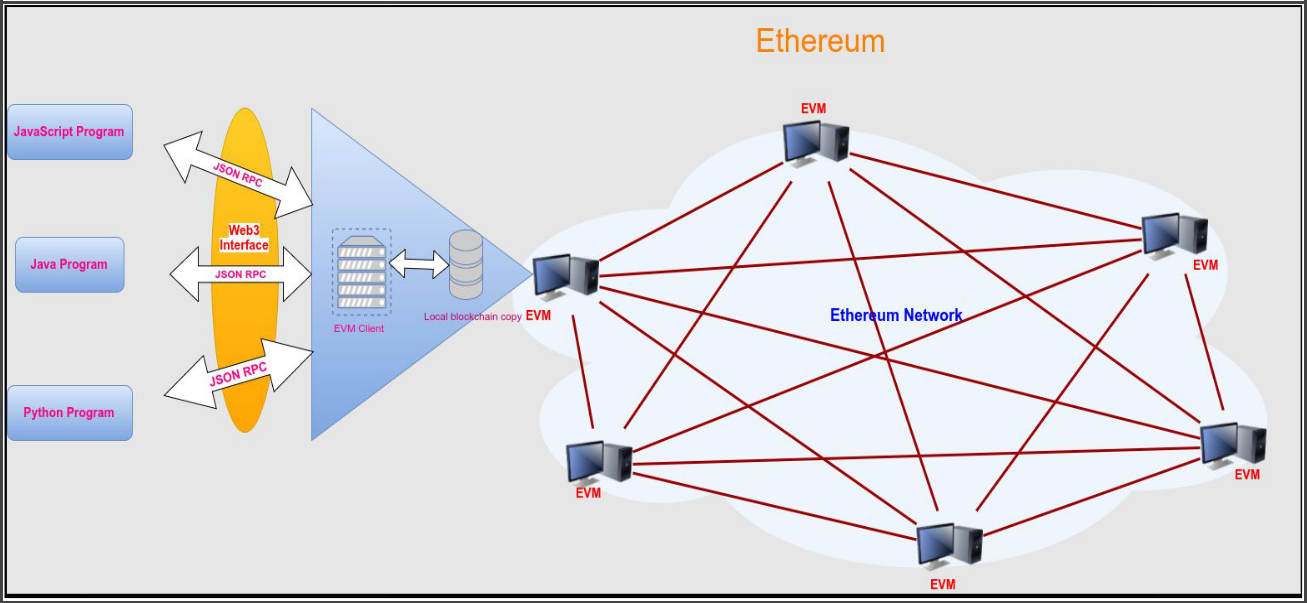


14



**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.

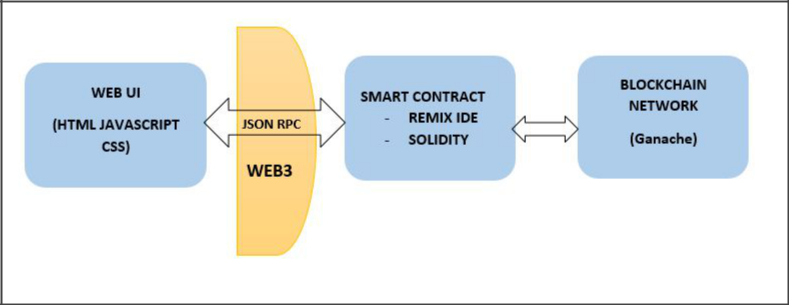
The EVM clients are developed using Go programming language or C++ programming language which implements the full functionalities of the EVM. The client always a local copy of the Ethereum blockchain database for backup purpose. The Ethereum network consists of several Ethereum virtual machines which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

15



**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 contenton 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.

16



**Sample ABI:**



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**

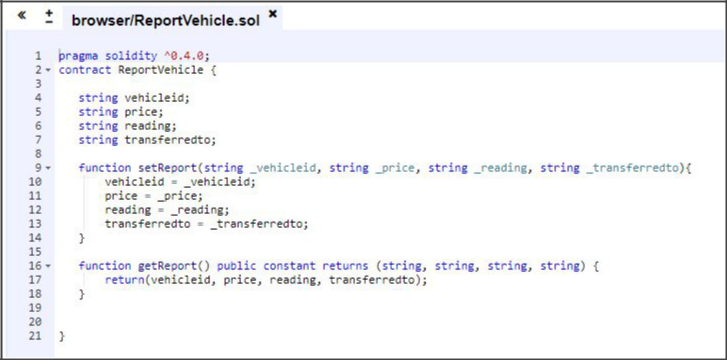
The smart contract code is written using the Solidity programming language. The IDE used for writing the smart contracts is REMIX. Remix is a Solidity IDE that consists of a set of tools to

17



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:**



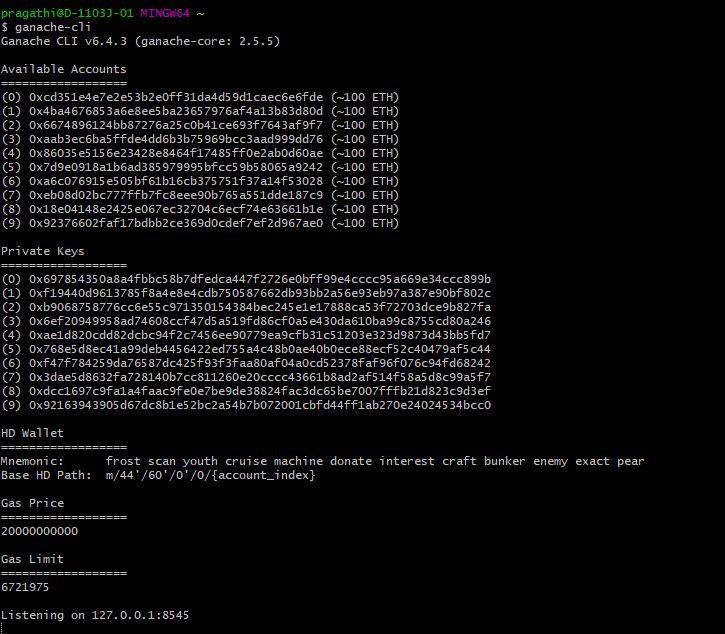
**Blockchain Network**

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.

18



**Sample Ganache-cli Output :**



**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.

19

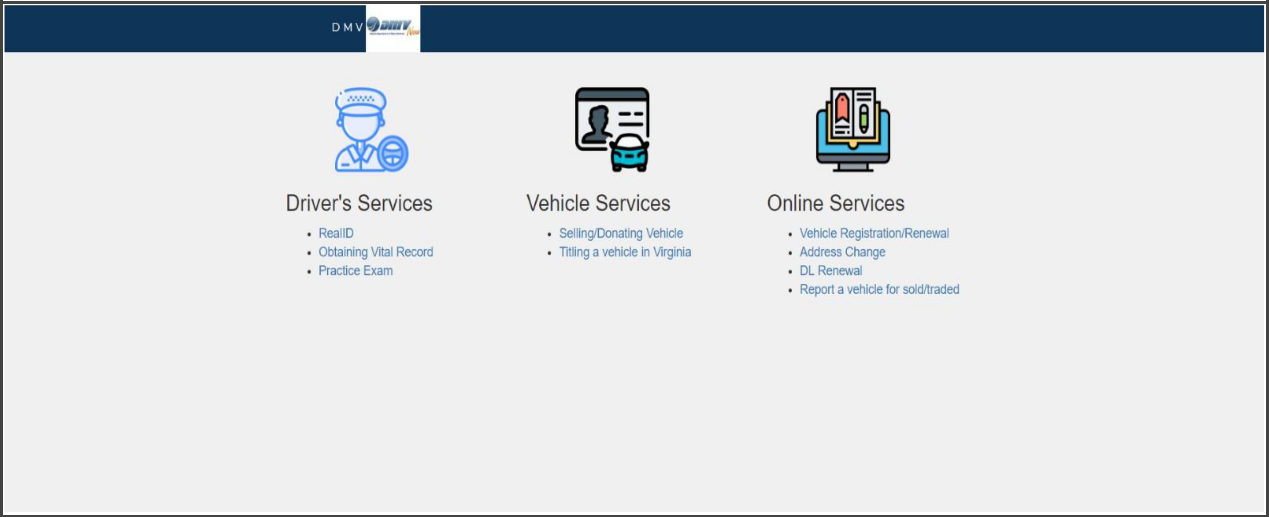


Some of the services that are implemented in this project are listed below

* **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

20



* RealID
* Obtaining Vital Record
* Practice Exams

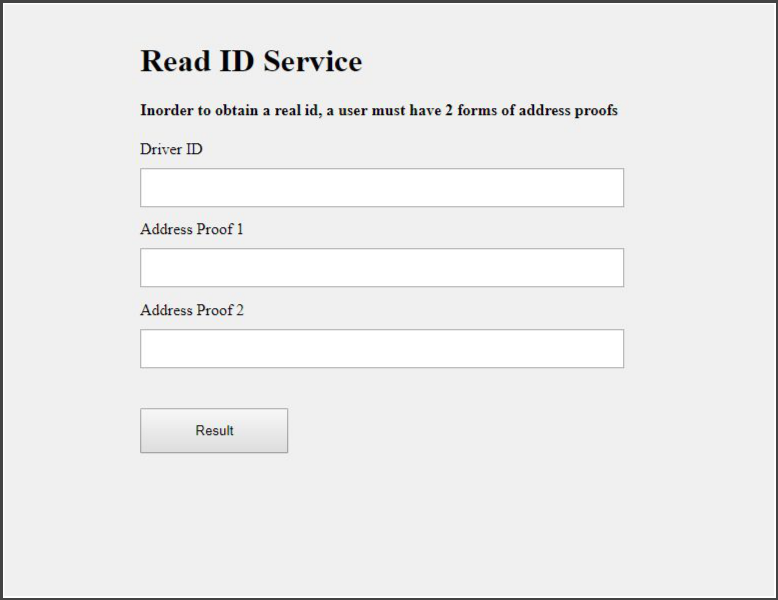
**RealID**

**Functions:**

* getId
* setId

**Variables​**:

* realId
* driverId
* address1
* address2



**Obtaining Vital Record**

**Functions:**

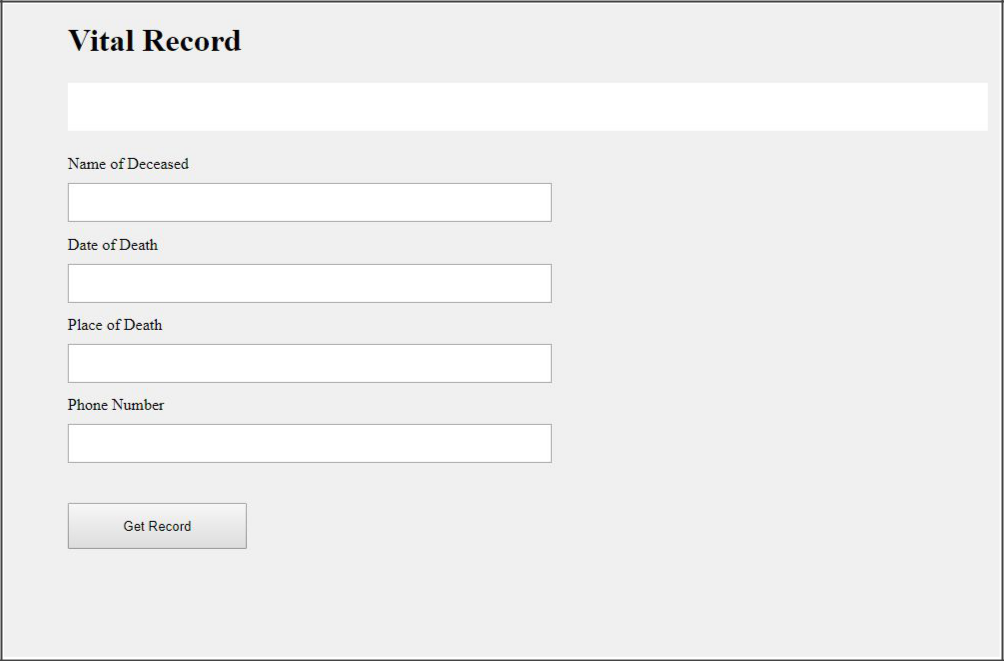
21



* getRecord
* setRecord

**Variables​**:

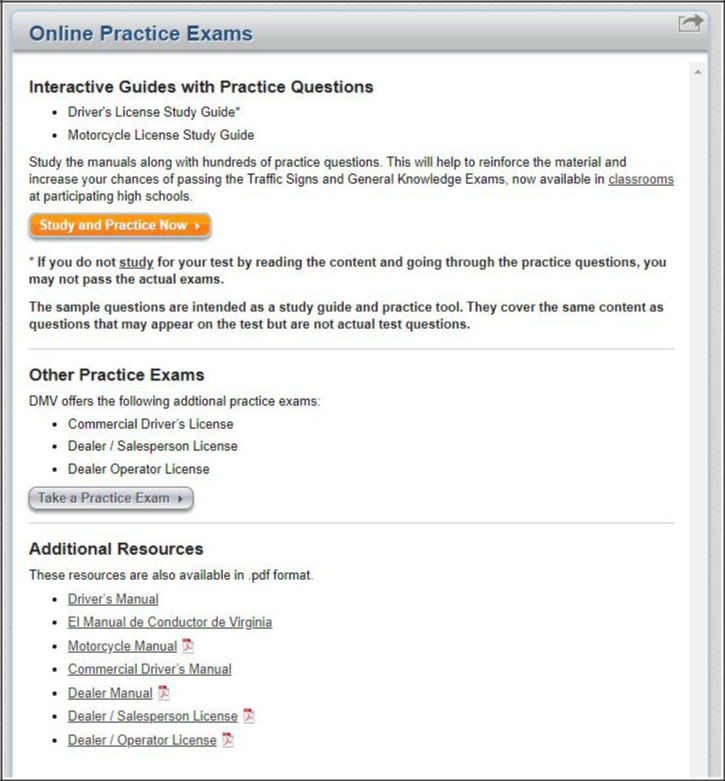
* name
* date
* place
* phone



22



**Practice​​Exams**



**Driver’s Services**

The Driver’s services provided are listed below

* Selling/Donating Vehicle
* Titling a Vehicle in Virginia

23



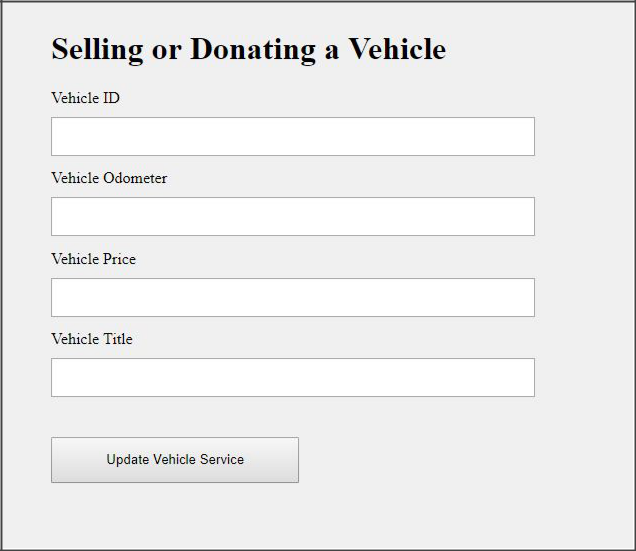
**Selling/Donating Vehicle**

**Functions:**

* getVehicle
* setVehicle

**Variables​**:

* vehicleId
* vehicleOdometer
* vehiclePrice
* vehicleTitle



24



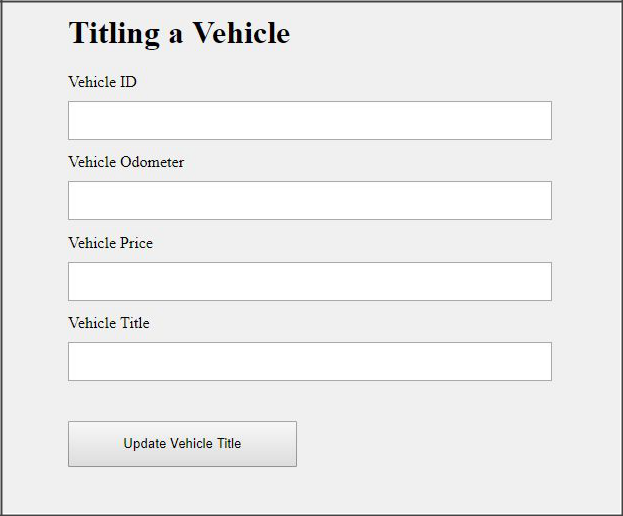
**Titling a Vehicle in Virginia**

**Functions:**

* getVehicleTitle
* setVehicleTitle

**Variables​**:

* vehicleId
* vehicleOdometer
* vehiclePrice
* vehicleTitle



25



**Vehicle Services**

The Vehicle services provided are listed below

* Vehicle Registration/Renewal
* Address Change
* Driver’s Licence Renewal
* Report a Vehicle Sold/Traded

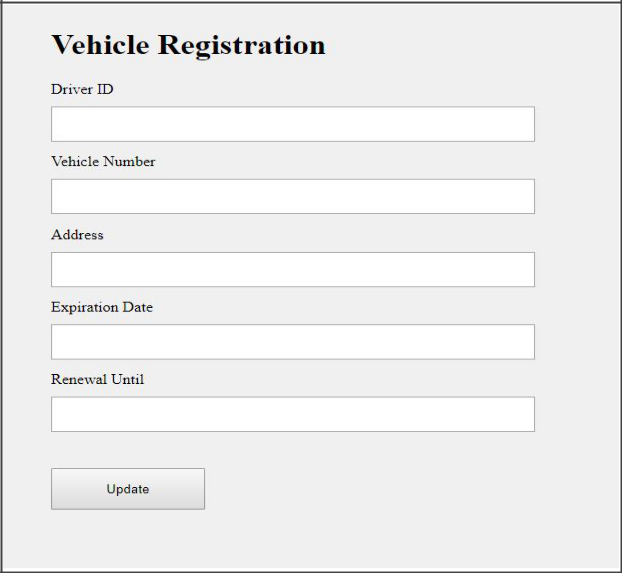
**Vehicle Registration/Renewal**

**Functions:**

* getVehicle
* setVehicle

**Variables​**:

* driverId
* vehicleNo
* Address
* expirationDate
* renewalDate



26



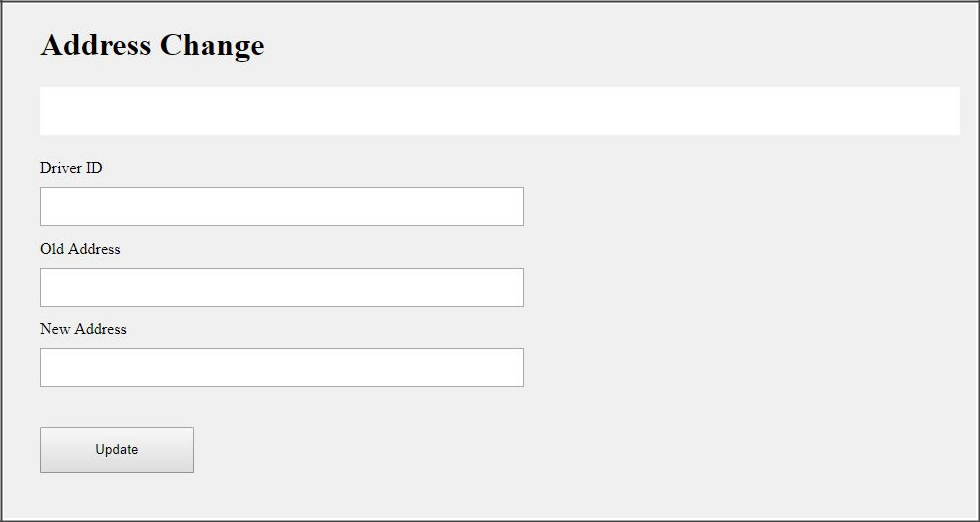
**Address Change**

**Functions:**

* getAddress
* setAddress

**Variables​**:

* driverId
* oldadd
* newadd



27



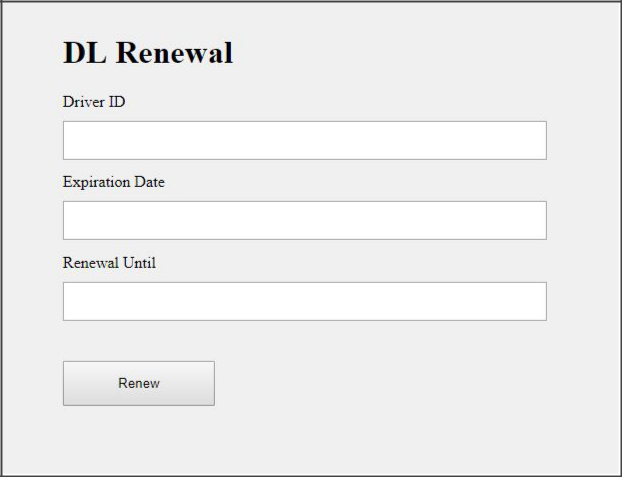
**Driver’s Licence Renewal**

**Functions:**

* getRenewal
* setRenewal

**Variables​**:

* driverId
* expirationDate
* newDate



28



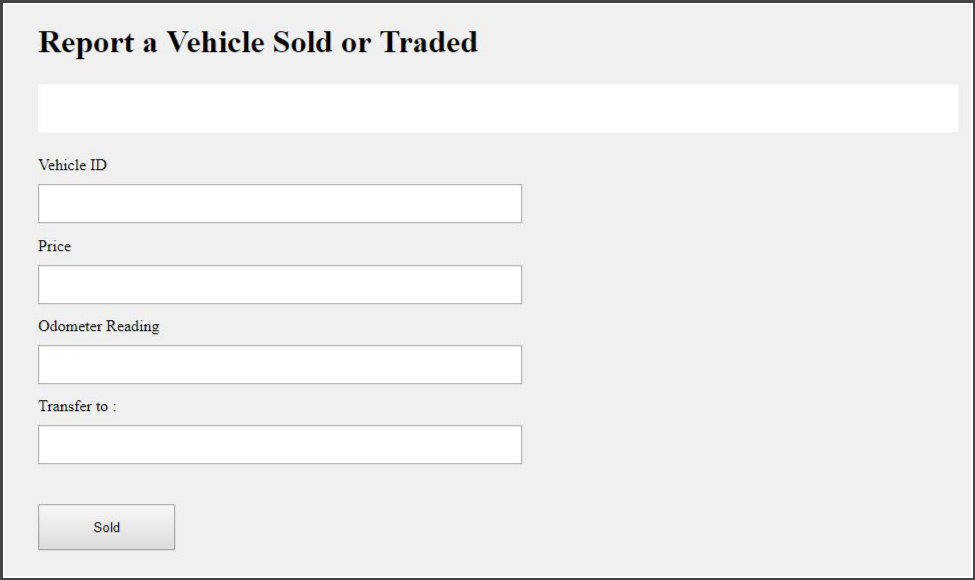
**Report a Vehicle Sold/Traded**

**Functions:**

* getReport
* setReport

**Variables​**:

* vehicleid
* price
* reading
* transferredto



29



**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.

30

