**Middleware Deployment for Ethereum:**

**Installation Requirements:**

**Install NodeJs and Angular:**

1. ***sudo apt-get install nodejs***
2. ***sudo apt-get install npm***

Nodejs is installed

***Update the Node version:***

1. ***Sudo npm install -g n***
2. ***sudo n stable***

**Node is updated**

1. ***npm install -g @angular/cli***

***Also we need to add proxy configuration in ~/.npmrc for nodejs app to communicate with any external API:***

Location of npmrc file can be found by **npm config list**

proxy=http://pxxxx: password @cosmos-vip.intra.renault.fr:3128/

https-proxy=http:// pxxxx:password@cosmos-vip.intra.renault.fr:3128/

http-proxy=http:// pxxxx: password @cosmos-vip.intra.renault.fr:3128/

http\_proxy=http:// pxxxx: password @cosmos-vip.intra.renault.fr:3128/

https\_proxy=http:// pxxxx: password @cosmos-vip.intra.renault.fr:3128/

HTTP\_PROXY=http:// pxxxx: password @cosmos-vip.intra.renault.fr:3128/

HTTPS\_PROXY=http:// pxxxx: password @cosmos-vip.intra.renault.fr:3128/

After Deploying the Smart Contract, the final procedure will be the deployment of the Middleware for each of the Nodes for ensuring the interacting with the smart contracts which will be either:

1. Manual through Each Method Invocation
2. Scheduled Jobs which will call the methods in the smart contract

Middleware satisfies the following functionalities apart from core functionalities listed in the Smart contract:

1. **Get the Car Data**: Car Data will be in this context from an API or ECU unit which will be odometer distance as well as GPS Co-ordinates of each trip
2. **Validate the Car Data**: Validate both sets of data, Odometer Data against GPS data calculated distance ( By Haversine Formula) and look for deviation of distance and report the deviation, validate true if within specified range. Range Can be set in the following Javascript file: **PARAM1:** <https://gitlabee.dt.renault.com/swlabs/blockchain/ethereum_poo/blob/master/api_admin1/odometerapi/services/locationservices/tripvalidation.js>

Current deviation percent acceptable set is 30

1. Perform the API Interaction with the Smart Contract:

* As mentioned earlier ABI for each of the smart contract is present in

<https://gitlabee.dt.renault.com/swlabs/blockchain/ethereum_poo/tree/master/api_admin1/odometerapi/blockchainapi/ethereum/abi>

* So, there is no need to modify the ABI which is saved as .json file unless or otherwise there is a change in the Smart contract variable or design.
* The Smart Contract address are specified as property file variables in. env of the main project directory. **PARAM2:** <https://gitlabee.dt.renault.com/swlabs/blockchain/ethereum_poo/blob/master/api_admin1/odometerapi/.env>

ethereum\_maincontract='0xf9eae0f545c0d5b1f1538df4746e46bf2c90d381'

ethereum\_transactioncontract='0x81c3b8de05e26519970a3a150db46fad9014d4e2'

ethereum\_walletcontract='0x92e9180e02fd62c91f509686c6457066b59266ca'

* Ethereum Node which we want to connect with is to be specified in the following property file representation inside .**env** files **PARAM3:**

Here both http and ws( websocket) is mentioned to allow some api services to communicate as some require ws.

ethereum\_host='http://10.214.221.229:2000'

ethereum\_wshost='ws://10.214.221.229:2000'

Other Main functionalities covered in the Middleware are:

|  |  |
| --- | --- |
| **AdminService** | Set/Get Main Contract Address |
| Set/ Get Transaction Contract Address |
| CRUD on Participant |
| CRUD on Follower |
| CRUD on Leader |
| Delete Transaction |
| Change Participant/Follower/Leader/Overall/token Transfer/Token Retract Consensus |
| Increase Token Supply |
| Token Transfer |
| Add Signature (Job Scheduling for the same is also done) |
| Remove Signature |
| Get Transaction Details |

|  |  |
| --- | --- |
| **User Service** | Token Request |

|  |  |
| --- | --- |
| **Common Service** | Check Leader/Participant /Follower |
| Get All leaders/Participants/Followers |
| Get Participant Details |
| Get AssetBalanceofNode |
| Get Peers |
| Get NodeInfo |
| Get HashRate |
| Get GasPrice |
| Get Current BlockNo |
| Get Transaction Details |
| Get Transaction Pool Content |
| Get Memstats |
| Get TxPool No |

**Scheduled Jobs:**

**PARAM 4:**

This parameter is used by the JavaScript file in <https://gitlabee.dt.renault.com/swlabs/blockchain/ethereum_poo/blob/master/api_admin1/odometerapi/services/blockchainservices/ethereum/adminservice.js>

Purpose of this scheduled Jobs will be only in **Admin Node Middleware** to provide the automatic signature to each of the transaction. (This job is useful if you trust the transaction or you are welcome to manually approve each transaction through this REST API middleware invoked by the Front End). Here time is specified in milliseconds

timer\_admin\_ether\_adminservice\_addsignature=500

Other Scheduled Jobs for Monitoring Purpose: (These can be removed when moving to production)

These jobs are used to look at the rate of transactions being processed.

test\_timer\_admin\_ether\_monitorservice\_listfinal=140000

test\_timer\_admin\_ether\_monitorservice\_lookconfirmation=5500

test\_timer\_admin\_ether\_monitorservice\_getallied=1000

test\_timer\_user\_ether\_assettransfer\_validateandprocess=100009999

This completes the explanation along with the important **PARAMS** to look for in this middleware. Design of Wallet in Ethereum is 70% shared between Ethereum Network + Smart contract, 30 % with Middleware Layer.

Midddleware written here is a NodeJs app which makes use of express js for routing purposes.

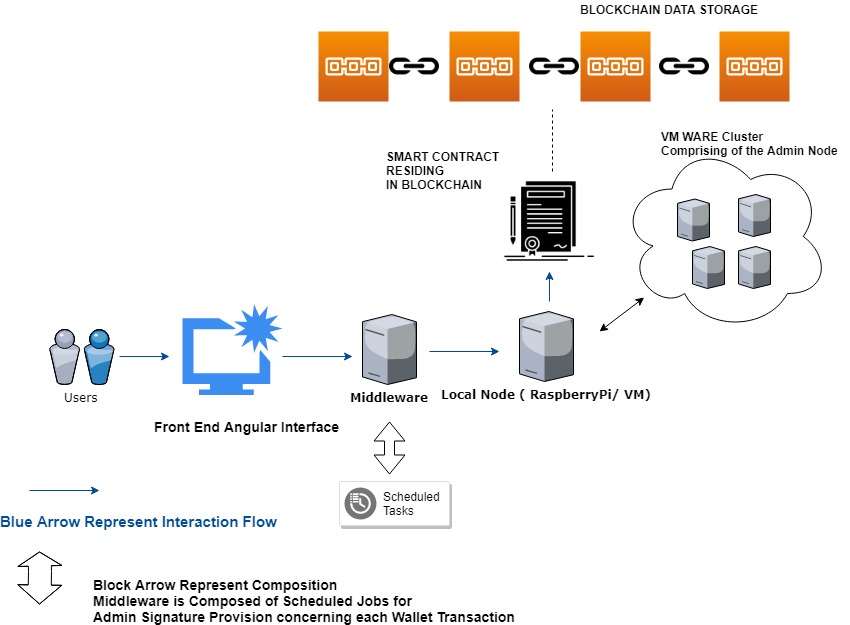
To compile and run the middleware we use the following command: **nohup npm start &**

For Now, this will be available at the following address: localhost:3000

**Front-End Interfacing with Middleware:**

Front-End was mostly built to show the simulation of the Routes and the Trip Validation with the Main purpose allowing the User as well as the Admin to interact with Node and then through the Smart Contract.

This diagram will explain it:

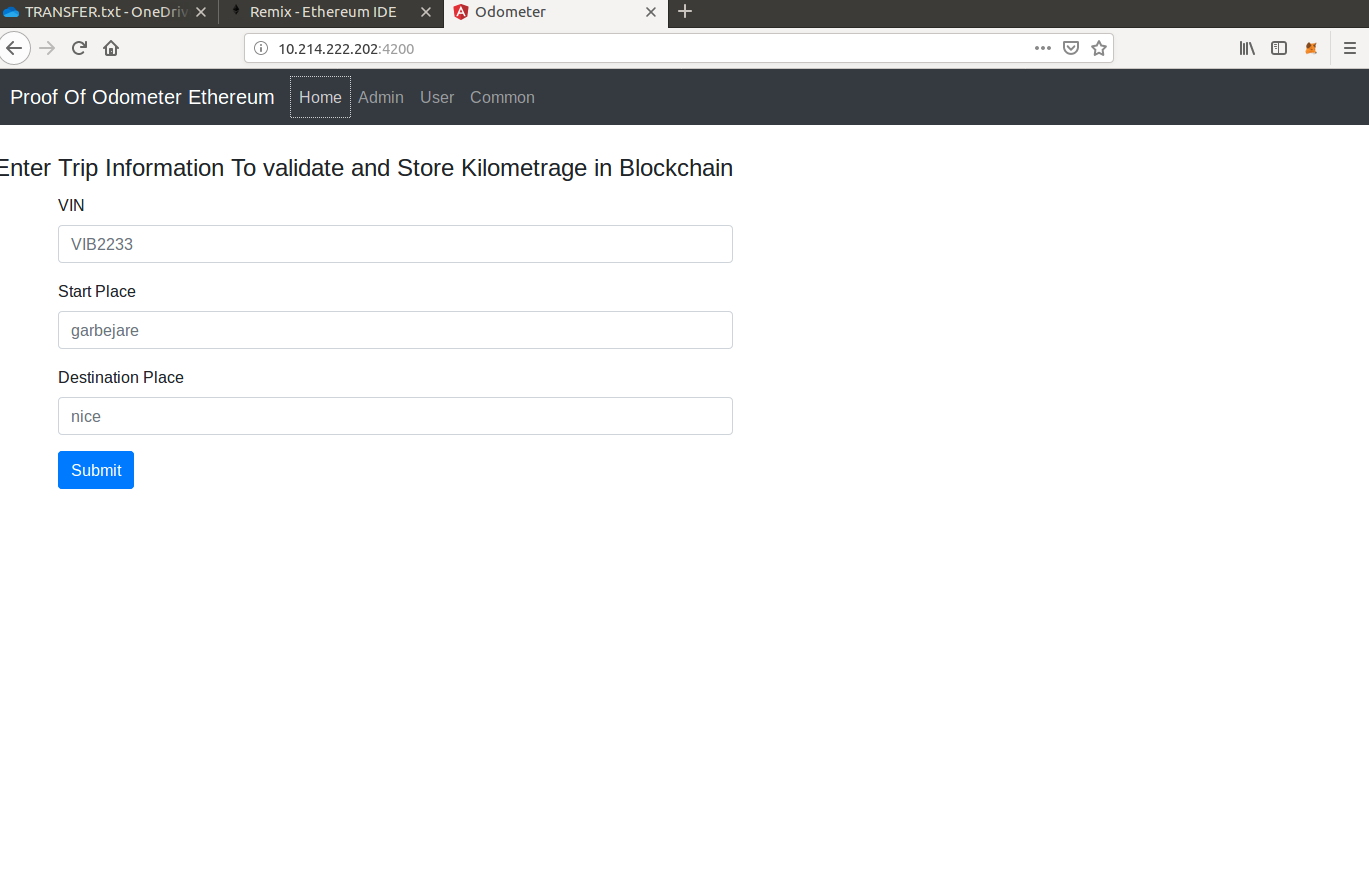


**What Front-End Contains?**

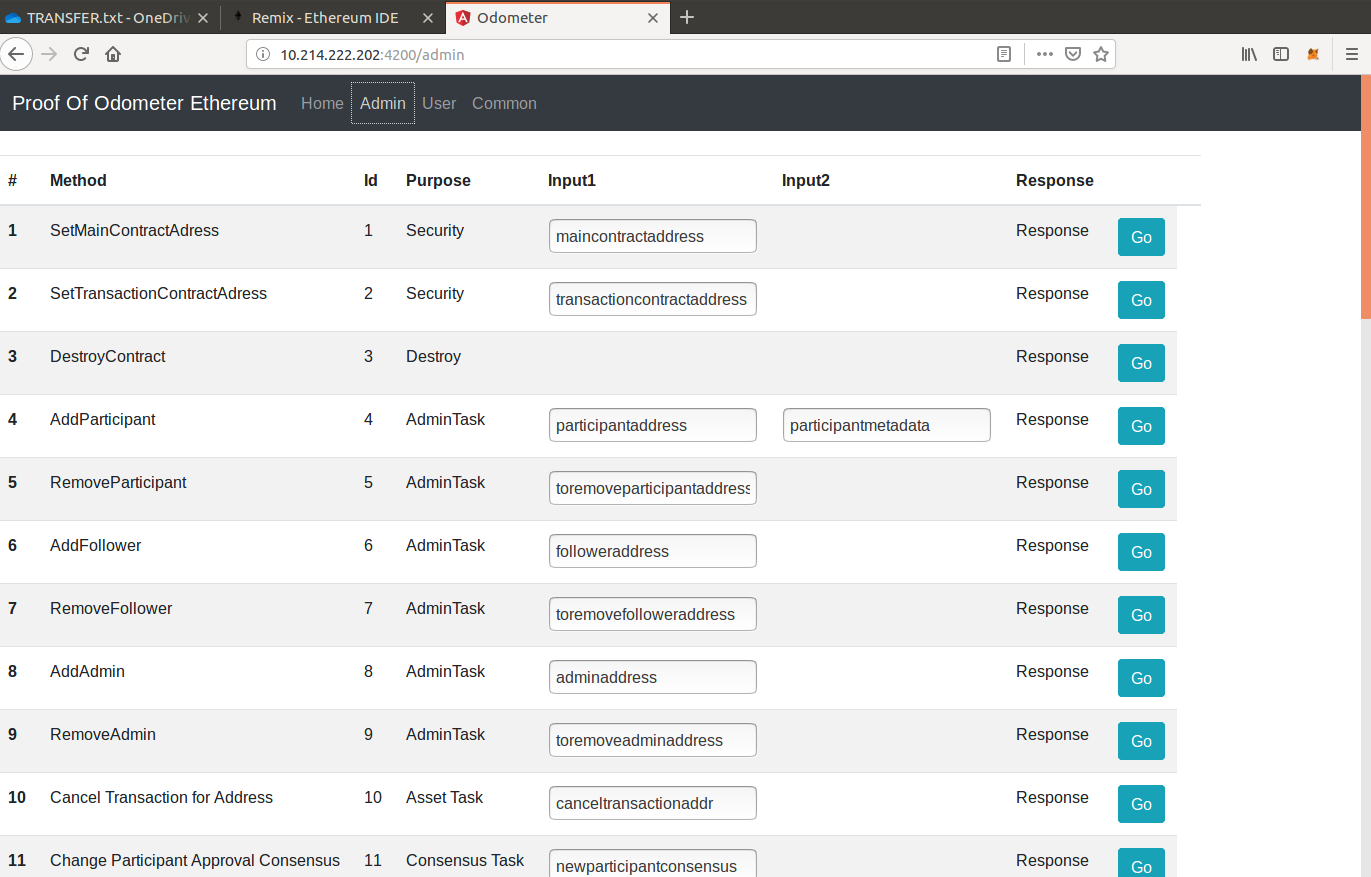
* Contains the Trip Simulation by reading Data from OSRM API to get the Actual Distance as well as calculate the Haversine Distance
* Then it shows the validation process followed in the Middleware
* It enables the transfer of validated data to the blockchain via an interaction with Smart Contract
* Also, It has both Admin as well as User Functionalities mentioned earlier in the article which can be invoke through front end
* In General, Blockchain Health and state of running can be derived from the Common Functionality page.

**Some Screenshots:**

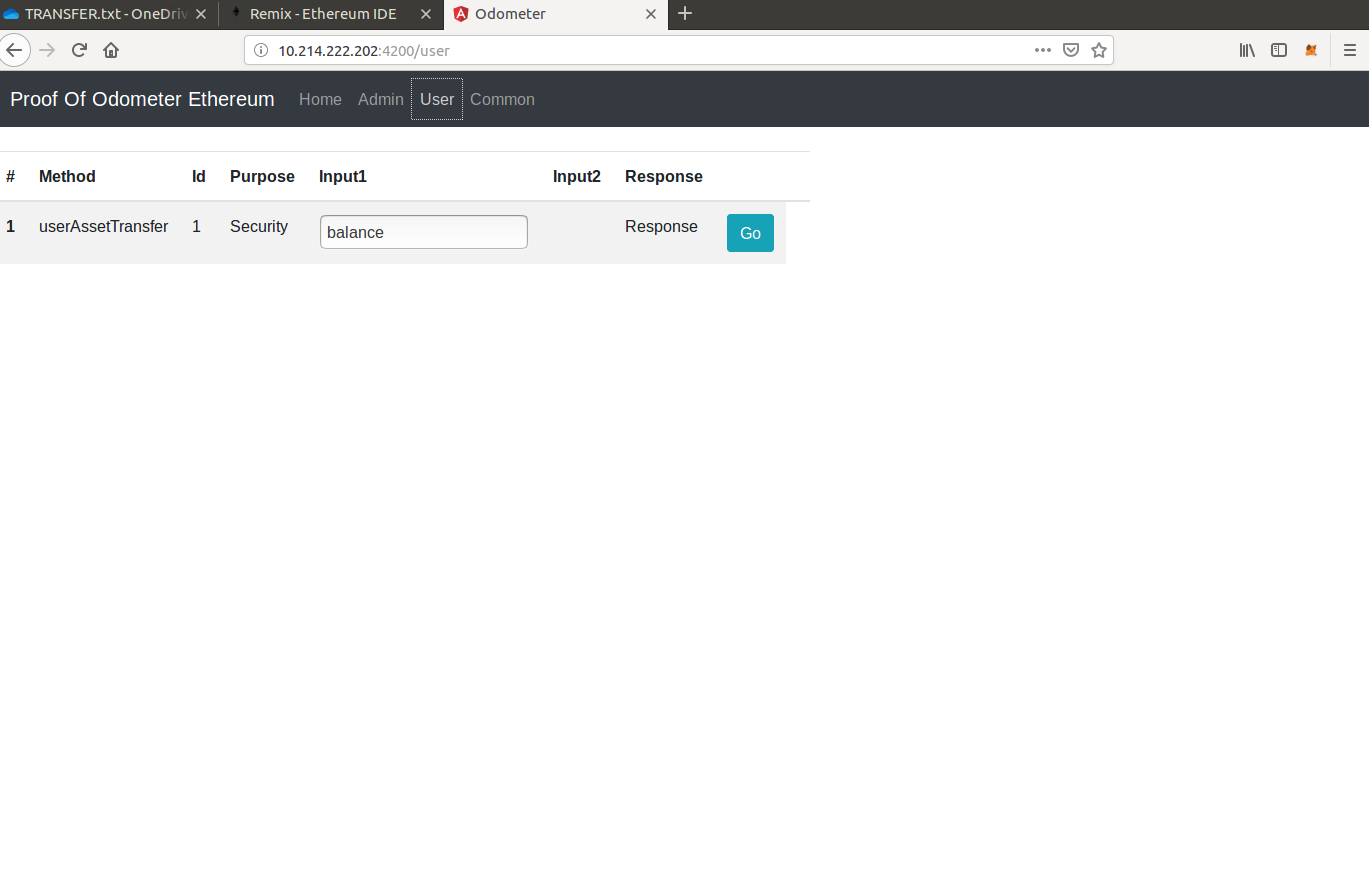
**Home Page:**

****

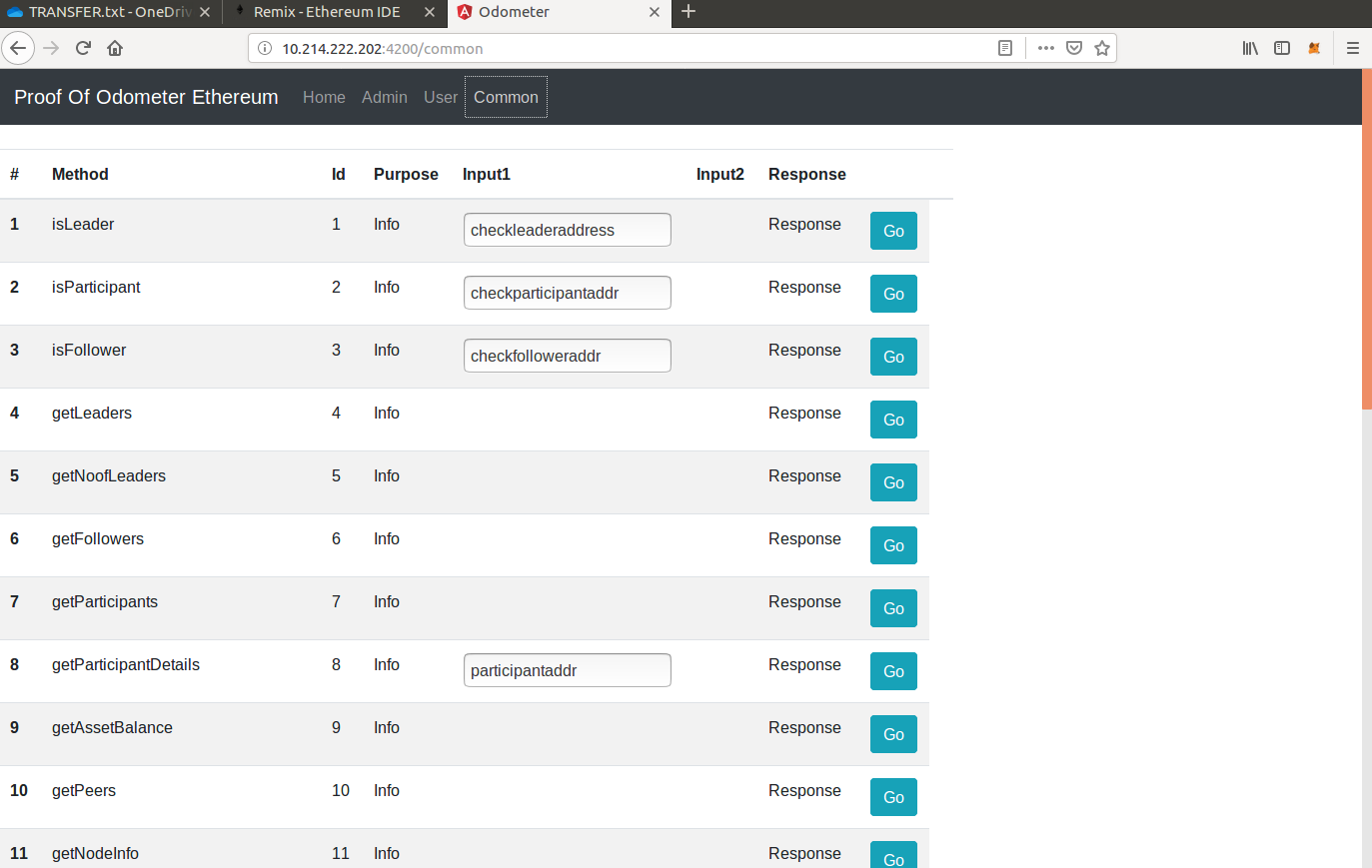
**Admin Page:**



**User Page:**



**Common Page:**



**How to Run this Front End?**

* **Clone the Project on a VM or Raspberry Pi:**

<https://gitlabee.dt.renault.com/swlabs/blockchain/ethereum_poo/tree/master/front-ethereum>

* **Then Inside the Main Project Run the angular server.**

**nohup ng serve --proxy-config proxy.conf.json --host 0.0.0 &**

**Here we use the proxy server to redirect all our rest calls towards the already deployed middleware locally on Port 3000**

**It Means: Front End (running on localhost:4200) ----🡪 Communicates with ---🡪 Middleware (Running on loclhost:3000)**

**Front-End can be accessed at localhost:4200**

**There is no additional configuration to be done and front end runs perfectly.**