Security In Computing: Implementation Report

Group Members

- Justin Naismith - s3605206 (WED-9:30-10:30)
- Siddhesh Kale - s3854666 (FRI-10:30-11:30)

Implementation Details

Our J&S Real Estate application is a blockchain marketplace that uses Smart Contracts to facilitate transactions between accounts.

React JS, Solidity and Ganache were used to develop this application. React is the front end, controlling the routes and the user input. React interacts with the blockchain through the MetaMask extension, which allows us to use Ethereum in the browser. The blockchain itself is running on Ganache. Contracts are created using Solidity and tested using Chai.

The most important code is the `Marketplace.sol` and `App.tsx`. These two files contain the code vital to the blockchain running, and contain comments explaining what the code is doing. The majority

of the rest of the code is React components for displaying the data we are transacting in the blockchain marketplace.

Setup

The instructions on running the application and its dependencies are outlined in the README.md file.

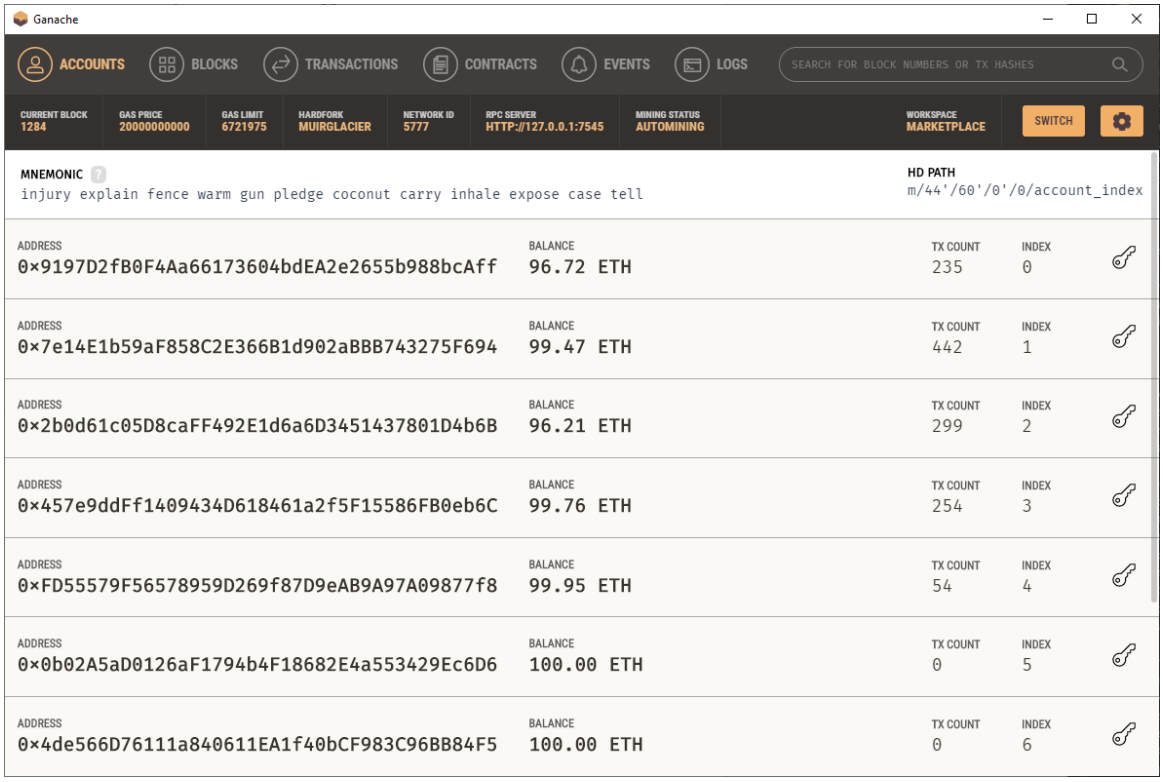
The following steps are required to run the application.

Node

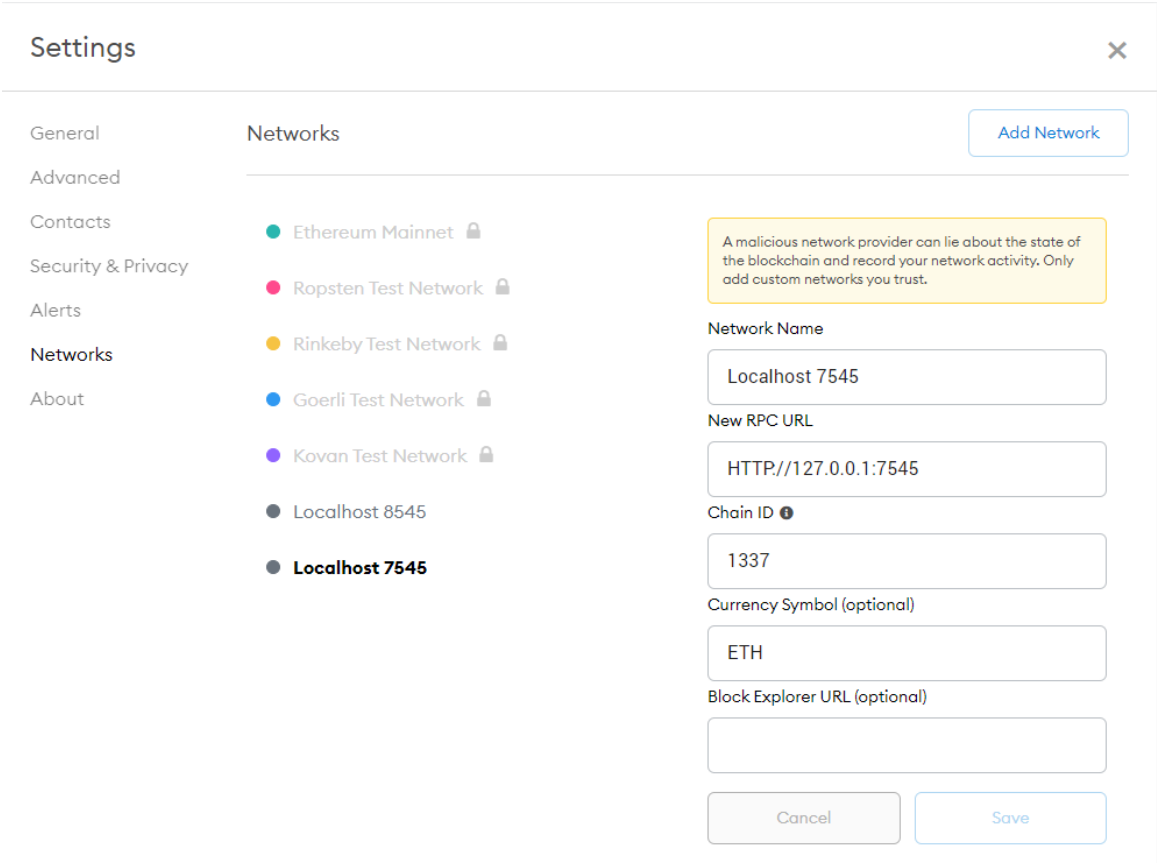
1. Install node.
2. Install the node modules using the `node i` command.

Ganache and MetaMask

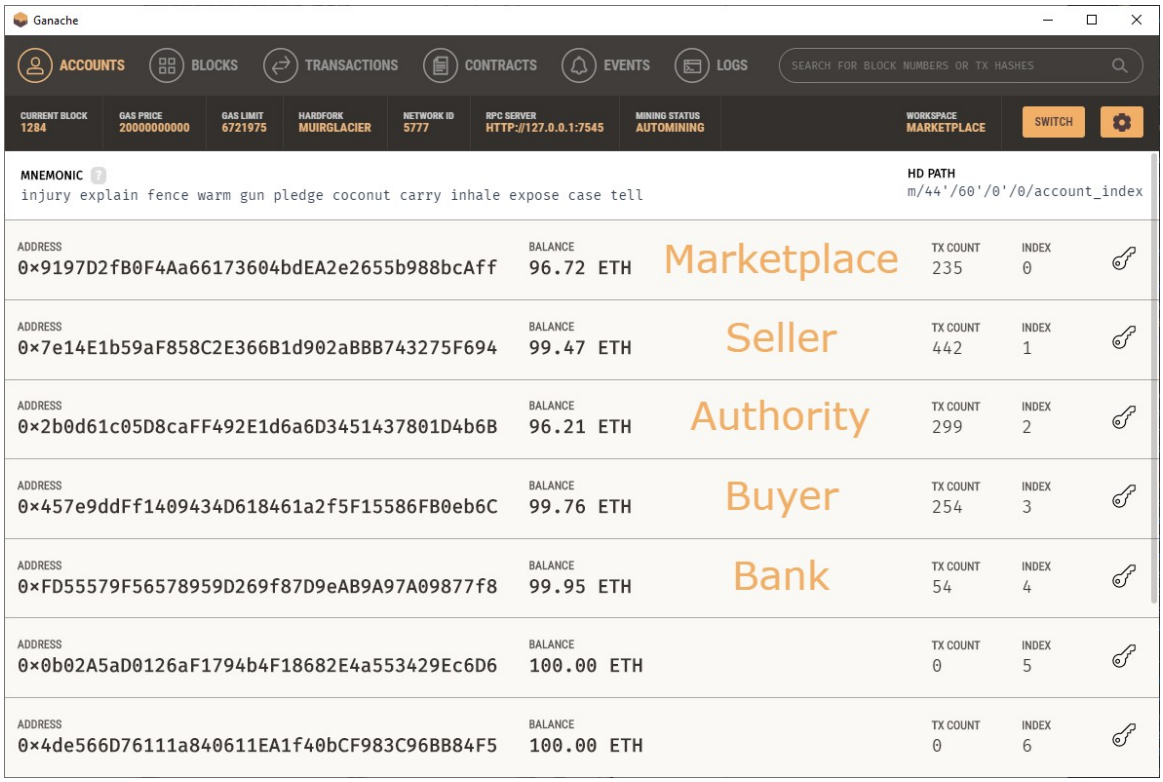
1. Download and configure Ganache. Found at this link.
2. Run the ganache application, select the create a new workspace option and name its **marketplace.**
3. Press the add project button and select the `truffle-config.js` file and save the workspace.
4. Doing so should present with the following screen:



5. Install the MetaMask browser extension which will be used to interact directly with Ganache. The extension can be downloaded here.
6. Create a test network as shown in the following image:



7. Import accounts to MetaMask. Further information is available here. Add them in the order shown below.



8. Replace the addresses located in `src/contracts/Marketplace.sol` and `src/utils/addresses.ts` with the ones from your Ganache.

```
src > utils > addresses.ts > ...
// Justin's addresses
export const Marketplace = '0x9197D2fB0F4Aa66173604bdEA2e2655b988bcAff';
export const Seller = '0x7e14E1b59aF858C2E366B1d902aBBB743275F694';
export const Authority = '0x2b0d61c05D8caFF492E1d6a6D3451437801D4b6B';
export const Buyer = '0x457e9ddFf1409434D618461a2f5F15586FB0eb6C';
export const Bank = '0xFD55579F56578959D269f87D9eAB9A97A09877f8';

// Siddhesh's addresses
// export const marketplace = '0x728Ad52C853e97Bb907F83F842043567266e3483';
// export const Seller = '0x386c0b9C66a334cEedf0059b1B4E44E43C2821a6';
// export const Buyer = '0x728Ad52C853e97Bb907F83F842043567266e3483';
// export const Authority = '0x2E11fF485F0B543222e5cb392395f3ec748E37B8';
// export const Bank = '0x5c5857f8cEB15DA1DF8d0217EcA61c8B19db501E';
```

```
constructor() public {
  name = 'SICI Real Estate Marketplace';
  // Justins addresses
  marketplace = 0x9197D2fB0F4Aa66173604bdEA2e2655b988bcAff;
  seller = 0x7e14E1b59aF858C2E366B1d902aBBB743275F694;
  authority = 0x2b0d61c05D8caFF492E1d6a6D3451437801D4b6B;
  buyer = 0x457e9ddFf1409434D618461a2f5F15586FB0eb6C;
  bank = 0xFD55579F56578959D269f87D9eAB9A97A09877f8;

  // Sids addresses
  // seller = 0x386c0b9C66a334cEedf0059b1B4E44E43C2821a6;
  // authority = 0x2E11fF485F0B543222e5cb392395f3ec748E37B8;
  // buyer = 0x728Ad52C853e97Bb907F83F842043567266e3483;
  // bank = 0x5c5857f8cEB15DA1DF8d0217EcA61c8B19db501E;
}
```

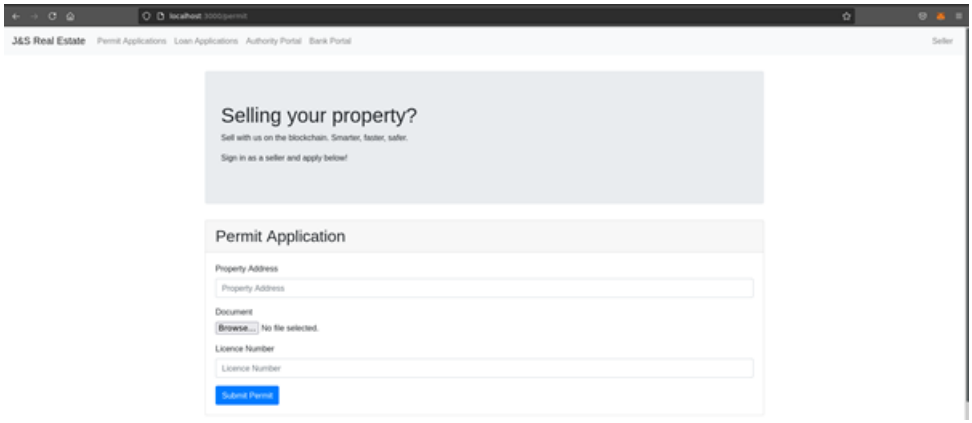
9. Run the following commands in the terminal.

```
truffle compile
truffle migrate
truffle deploy
npm start
```

Simulating user interactions

Seller

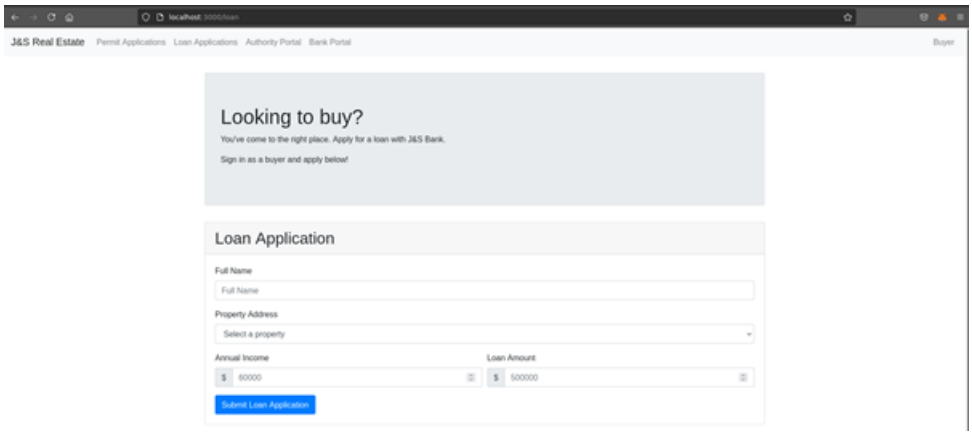
- To simulate Seller actions, swap to the seller account on MetaMask and refresh the page.



- If the Seller is logged in successfully the above should be visible.
- A seller can then apply for a permit by providing the necessary information.

Buyer

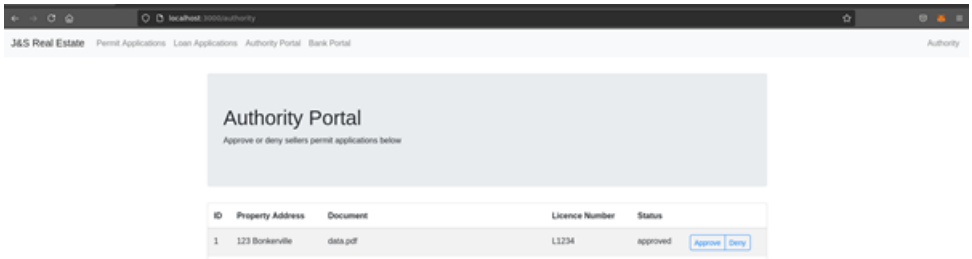
- To simulate Buyer actions, swap to the seller account on Meta mask and refresh the page.



- If the Buyer is logged in successfully the above should be visible.
- A buyer can then apply for a loan by providing the necessary information.

Authority

- To simulate Authority actions, swap to the Authority account on MetaMask and refresh the page.

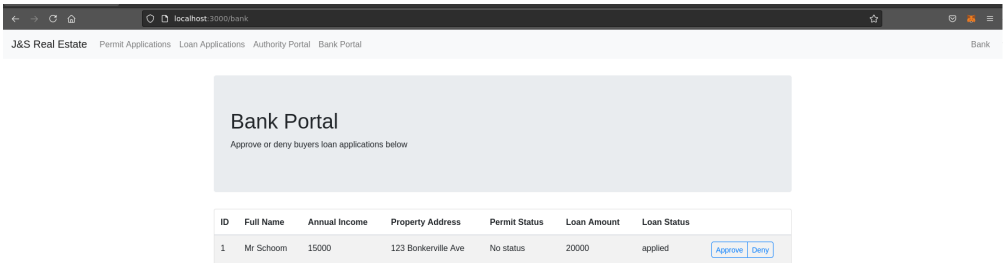


- If the Authority is logged in successfully the above should be visible.

- The Authority can then Approve or Deny the the Permit applications.

Bank

- To simulate Bank actions, swap to the Bank account on MetaMask and refresh the page.



- If the Bank is logged in successfully the above should be visible.
- The Bank can then Approve or Deny the Loan applications.

Bibliography

- Dapp University. (2019, August 12). Intro To Blockchain Programming (Etherum, Web3.js & Solidity Smart Contracts) [FULL COURSE]. YouTube. https://youtu.be/VH9Q2lf2mNo
- (2019, July 20). dappuniversity/marketplace. GitHub. https://github.com/dappuniversity/marketplace/tree/part-3

Security In Computing: Implementation Report