# Maintenance Manual

April 2024

## 1 Installing the System

Upon unzipping the program files, you should open the root folder in a Code Editor of choice and open two new terminals. In one terminal you will need to run this command: **node server.js**, and in the second terminal you will need to run: **npm run dev**.

This will ensure that both the Express Server, and React Application are running. All of the Smart Contracts are already deployed, and instructions on how to recompile and redeploy are in Appendix A if required.

## 2 Dependencies and Requirements

The implementation requires the following hardware and software dependencies. The program does utilise NPM package manager, so all of the required libraries can be easily installed using **npm install**. However all required libraries are listed below:

- At least 4GB of RAM and 10GB of storage space are required to run the applications front end React component.
- JavaScript
- "cors" version 2.8.5
- "ethers" version 6.11.1
- "ganache" version 7.9.1
- "merkletreejs" version 0.3.11
- "mocha" version 10.2.0
- "next" version 12.3.4
- "react" version 18.2.0
- "react-dom" version 18.2.0

- "semantic-ui-css" version 2.5.0
- "semantic-ui-react" version 2.1.5
- "solc" version 0.8.9
- "sphincs" version 3.0.4
- "web3" version 4.6.0

# 3 Key File Paths

- Compile script: ./compile.js
- Private Key Storage File: ./keyStorage.json
- Express Server: ./server.js
- Web3 Configuration: ./web3.js
- Smart Contracts : /contracts
- Off-Chain Functionality: ./contracts/SecondaryFunctions
- Bytecode and ABIs of deployed contracts: /deployedArtifacts
- React Front-End Pages: /pages
- Schema Template: /schemas
- Unit Test Files: /test
- Smart Contract Web3 Instances: /web3Deployments

# 4 Directions for Future Improvements

Within the Smart Contracts, gas optimisation techniques including Batch Transactions and Gas Refunds should be implemented. When the solidity code has changed, the compile script will need to be ran again, and the contracts redeployed to a test network such as Sepolia. As well as this the Smart Contract Web3 Instances will need to be updated with the address of the newly deployed contracts.

# 5 List of Source Code Files

# 6 Known Bugs

At time of writing, there is currently no major known bugs within the project.

If in the future, such as when new Solidity Compilers are released, the bug involving nested structs should be investigated again and the existing Smart Contract structs potentially updated.

| Source Code File | Summary of Role |
|---|---|
| compile.js | A script for compiling all of the Solidity Smart Contracts |
| server.js | A script for running the Express Server |
| web3.js | Script which sets up the Digital Wallet Injection as a Web3 instance for React |
| keyStorage.json | Off-chain storage of public keys for each DID |
| web3Deployments | Folder containing Web3 Instances for every deployed Smart Contract |
| test | Folder containing all of the Unit Test Files |
| pages | Folder containing all of the front-end pages |
| deployedArtifacts | Folder containing Bytecode and ABI for each deplpyed contract |
| contracts | Folder containing all of the Smart Contract Solidity files |

Table 1: Source Code