University of Mumbai

HealthKard - Decentralized User Health Identity and Health Records Management with Web3, IPFs and the Ethereum Blockchain

Submitted at the end of semester VIII in partial fulfillment of requirements

For the degree of
Bachelors in Technology

by

Umang Thadani
Roll No: 1914061

Anurag Singh
Roll No: 1914058

Guide
Prof. Era Johri

Department of Information Technology
K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Batch 2019-23

Contents

List of Figures 3

List of Tables 4

Introduction 5

Technology Stack 6

Installation Setup 7

1. Preliminary Setup 7
2. Blockchain 9
3. Backend 13
4. Frontend 15

Test Cases 16

1. Features to be tested 17
2. Test Cases 19
3. Test Logs 22

List of Figures

Fig 1: Python Installation Page 7

Fig 2: NodeJS Installation Page 8

Fig 3: Ganache Installation Page 9

Fig 4: Ganache App 10

Fig 5: Healthkard Blockchain Repository 11

Fig 6: Remix IDE Environment 11

Fig 7: Remix IDE Deploy 12

Fig 8: Healthkard Backend Repository 13

Fig 9: Healthkard Backend .env 14

Fig 10: Healthkard Frontend Repository 15

Fig 11: Healthkard Frontend .env 16

List of Tables

Table 1: Features to be tested - a 17

Table 2: Features to be tested - b 18

Table 3: Test Cases - a 19

Table 4: Test Cases - b 20

Table 5: Test Cases - c 21

Table 6: Test Logs - a 22

Table 7: Test Logs - b 23

Table 8: Test Logs - c 23

Table 9: Test Logs - d 24

Table 10: Test Logs - e 25

Introduction

The purpose of this digital and decentralized health identity is to store all your health records in one place to make it convenient for users by avoiding the hassles of carrying physical reports or multiple digital reports, each from a different service.

The system is based on Blockchain and the idea of NFTs so that the unique health identity cannot tamper with and that the health records are stored in a secure manner without centralized control over your sensitive records. We will provide a comfortable User Interface to find patients, request access to health records, and ensure patients are able to grant access with his/her consent and let users store their Electronic Health Records (EHRs)

HealthKard aims to implement the following modules:

- Creation of a unique Health ID using Aadhaar Number
- Storage of Electronic Health Records (EHRs) mapped to Health Identity in the blockchain
- Integration of different sectors in the medical industry
- Encourage better administration of the health sector by utilizing health data analytics

Currently, there is no unified framework in India that facilitates the storage of health records of all citizens. Although private companies/hospitals do solve this problem to some extent by providing e-health record services, they only cater to their own patients/customers. Therefore, there is a need to develop the foundations necessary for supporting digital health infrastructure to maintain health data in a decentralized and secure way. A few major advantages to this project will be ease of access, user consent for every sophisticated transaction, and portability across national borders.

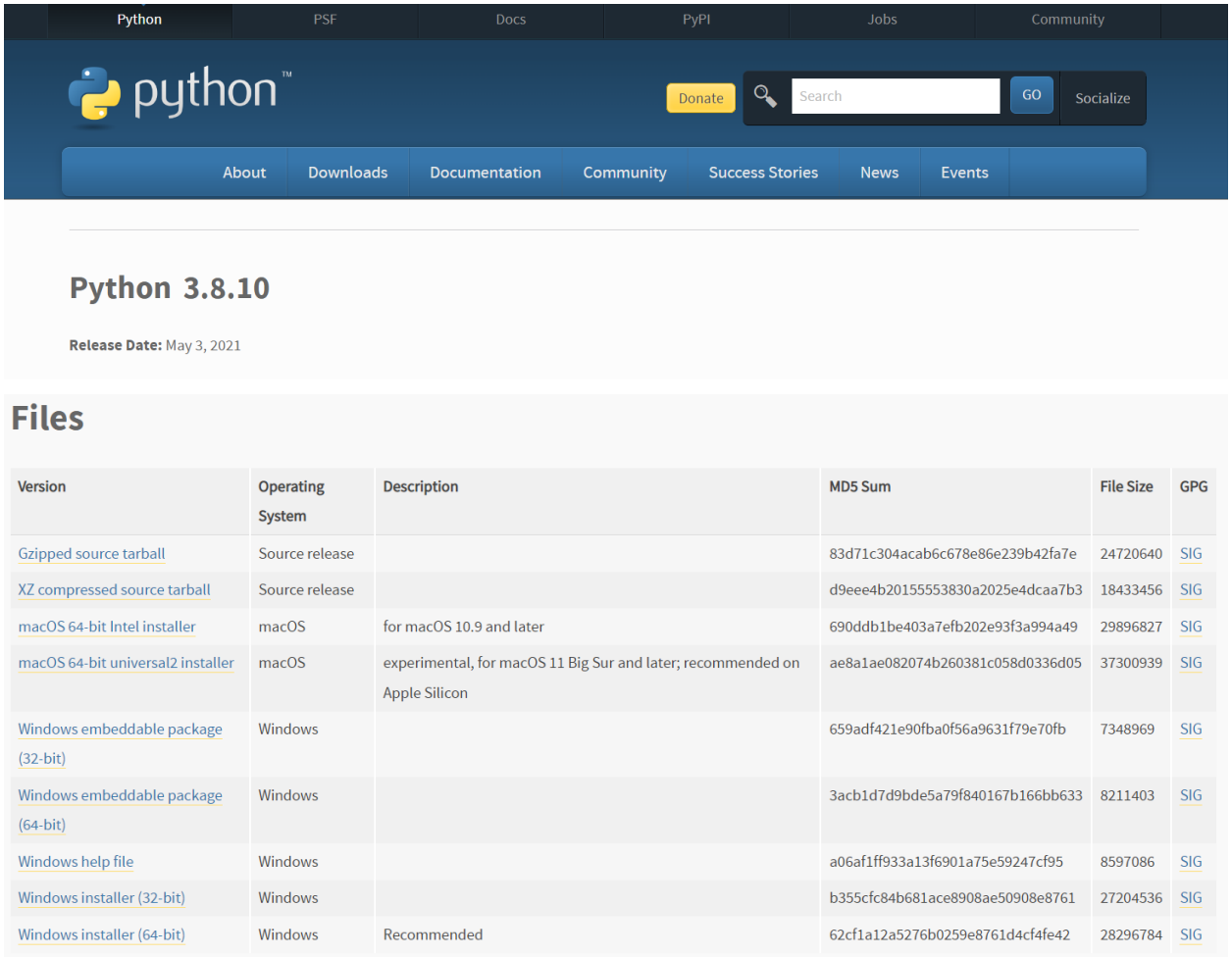
Technology Stack

- SRS, SPMP, SDD, and STD Document
  - Overleaf
- Blockchain
  - Solidity v0.8.7
  - Ganache v2.7.1
  - Goerli Testnet
  - Remix IDE
  - IPFS
  - Metamask
- Back End
  - Django REST Framework v3.14.0
  - Python v3.8.10
  - Web3 v5.31.1
  - Cryptography v40.0.1
- Front End
  - ReactJS v18.2.0
  - Mantine v11.10.5
- Hardware Interfaces
  - Processor: x86 or x64
  - RAM: 512 MB (minimum), 1 GB (recommended)
  - Hard disk: up to 512 MB of available space may be required.
- Software Interfaces
  - Operating System: Any OS that can support a gunicorn server.
  - Browser Requirements: Any modern web browser having MetaMask extension.
- Communication Protocols
  - HTTP (HyperText Transfer Protocol)

Installation Setup

Preliminary Steps

1. Install Python and Setup Environment Variables:
    a. Download the v3.8.10 of Python from the official website (https://www.python.org/downloads/release/python-3810/).



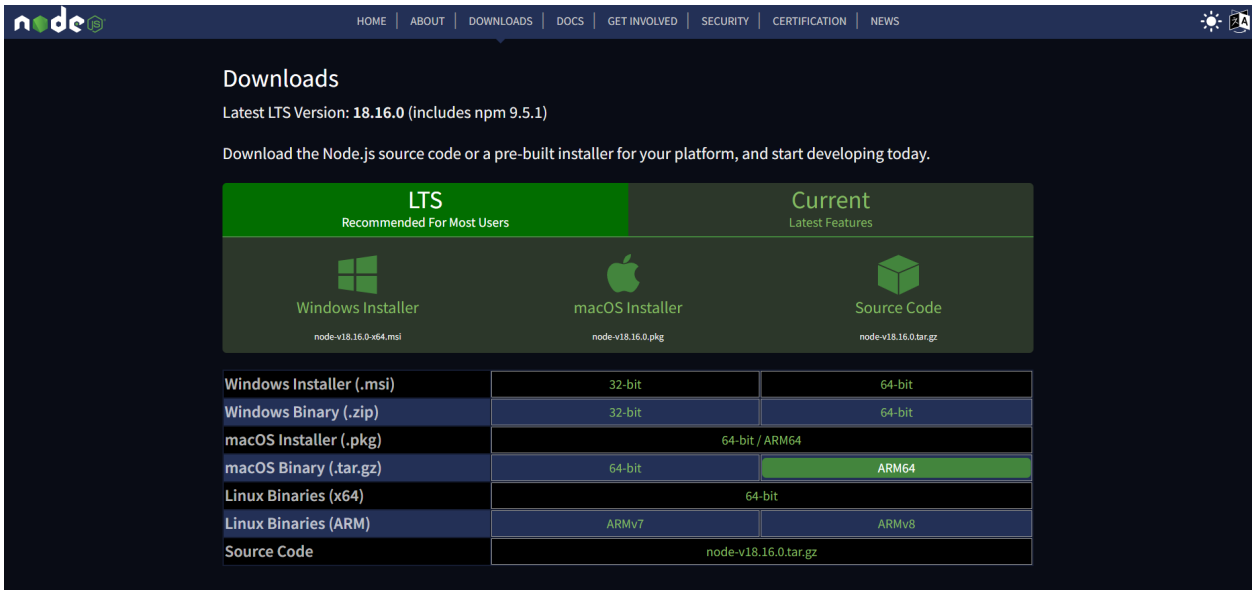
Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		83d71c304acab6c678e86e239b42fa7e	24720640	SIG
XZ compressed source tarball	Source release		d9eee4b20155553830a2025e4dcaa7b3	18433456	SIG
macOS 64-bit Intel installer	macOS	for macOS 10.9 and later	690ddb1be403a7efb202e93f3a994a49	29896827	SIG
macOS 64-bit universal2 installer	macOS	experimental, for macOS 11 Big Sur and later; recommended on Apple Silicon	ae8a1ae082074b260381c058d0336d05	37300939	SIG
Windows embeddable package (32-bit)	Windows		659adf421e90fba0f56a9631f79e70fb	7348969	SIG
Windows embeddable package (64-bit)	Windows		3acb1d7d9bde5a79f840167b166bb633	8211403	SIG
Windows help file	Windows		a06af1ff933a13f6901a75e59247cf95	8597086	SIG
Windows installer (32-bit)	Windows		b355cfc84b681ace8908ae50908e8761	27204536	SIG
Windows installer (64-bit)	Windows	Recommended	62cf1a12a5276b0259e8761d4cf4fe42	28296784	SIG

[Fig 1: Python Installation Page]

    b. Follow the installation wizard to install Python on your local machine.
    c. After installation, add the Python installation directory to the system path environment variable:
        i. For Windows: `set PATH=%PATH%;C:\PythonXX`, where XX represents the version number.
        ii. For Linux/Mac: `export PATH=$PATH:/usr/local/bin/python`.
    d. Test if Python is installed correctly by running the command `python` in a terminal or command prompt.

2. Install pip and Setup Environment Variables:
    a. pip is a package manager for Python, which is used to install and manage Python packages.
    b. To install pip, run the following command in a terminal or command prompt: `python -m ensurepip`.
    c. After installation, add the pip installation directory to the system path environment variable:
        i. For Windows: `set PATH=%PATH%;C:\PythonXX\Scripts`, where XX represents the version number.
        ii. For Linux/Mac: `export PATH=$PATH:/usr/local/bin/python/Scripts`.
    d. Test if pip is installed correctly by running the command `pip` in a terminal or command prompt.

3. Install Node.js and npm and Setup Environment Variables:
    a. Download the latest version of Node.js from the official website (https://nodejs.org/en/download/).

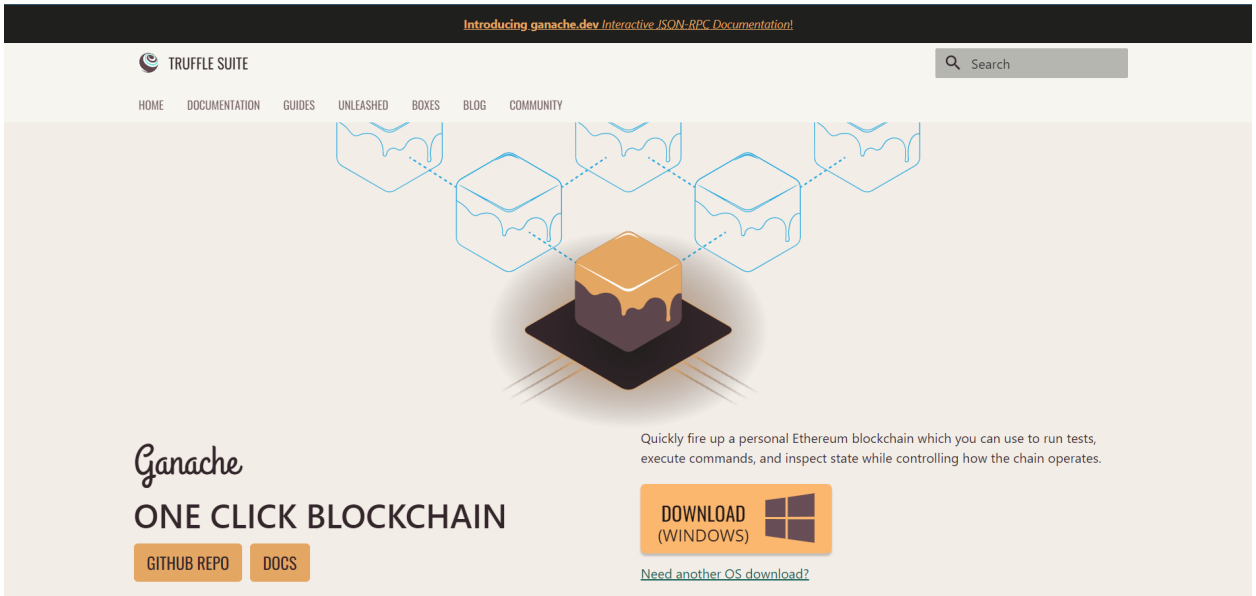


[Fig 2: NodeJS installation page]

    b. Follow the installation wizard to install Node.js on your local machine.
    c. After installation, add the Node.js installation directory to the system path environment variable:
        i. For Windows: `set PATH=%PATH%;C:\Program Files\nodejs`.
        ii. For Linux/Mac: `export PATH=$PATH:/usr/local/bin/node`.
    d. Test if Node.js and npm are installed correctly by running the command `node -v` and `npm -v` in a terminal or command prompt.

Blockchain

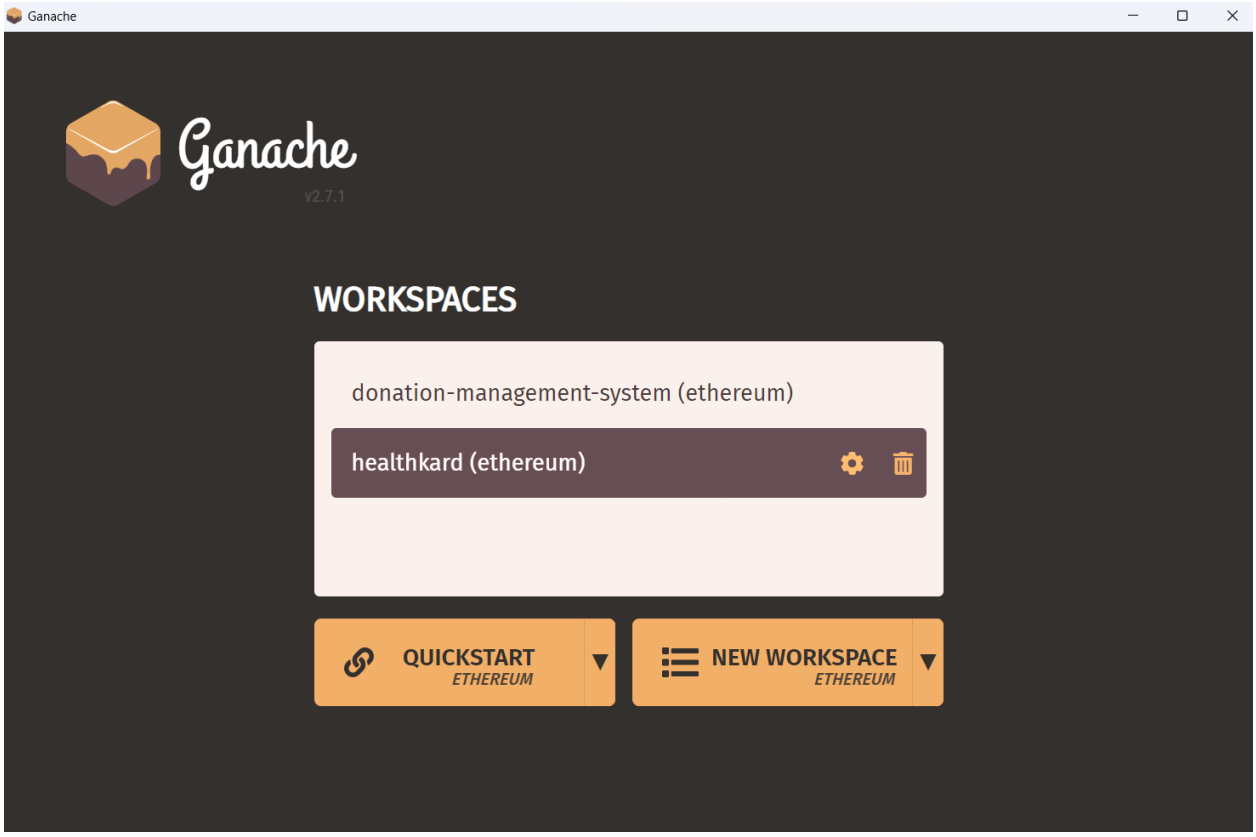
1. Install Ganache:
    a. Ganache is a personal blockchain for Ethereum development, and it can be downloaded from the Truffle Suite website. Go to https://www.trufflesuite.com/ganache and download the appropriate version of Ganache for your operating system.



[Fig 3: Ganache Installation Page]

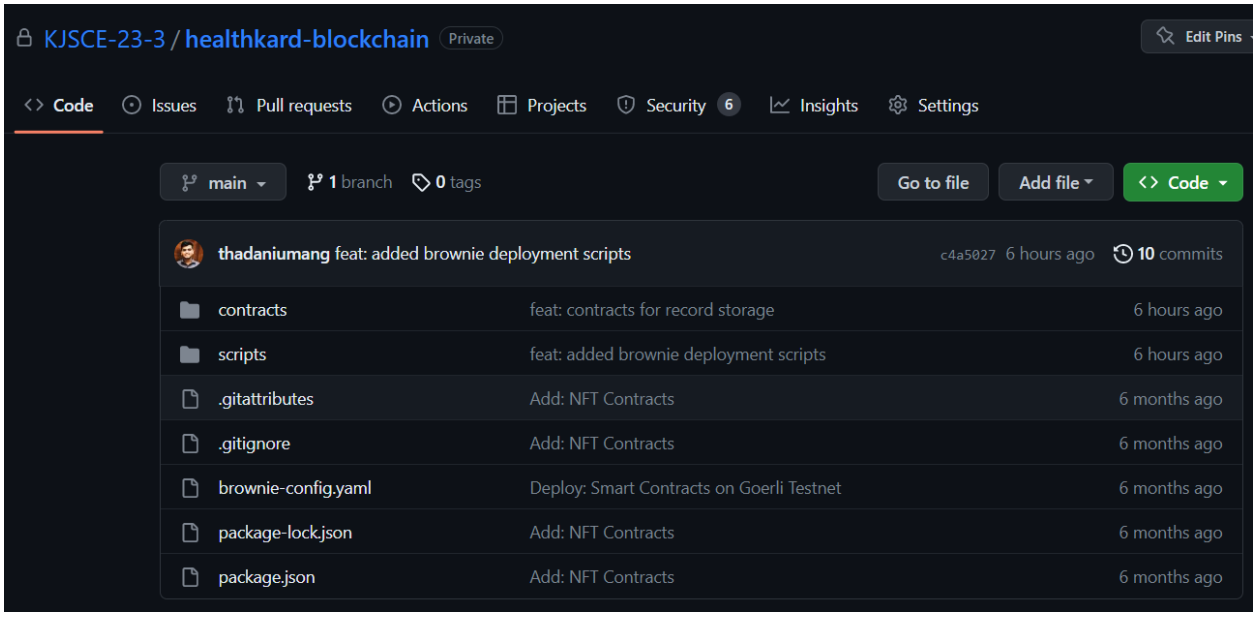
    b. Install Ganache on your local machine by running the installation file and following the on-screen instructions.

2. Start a Workspace:
    a. Launch Ganache and create a new workspace by clicking on the "New Workspace" button.
    b. Configure the workspace settings as per your project requirements.
    c. Click on the "Save Workspace" button to create a new workspace.



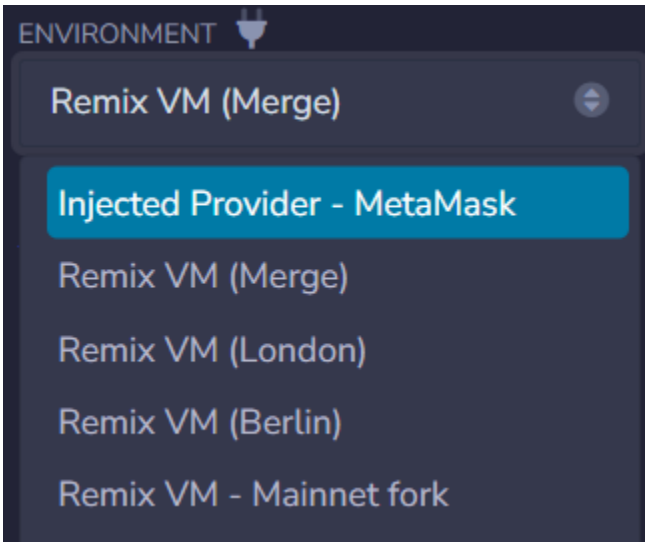
[Fig 4: Ganache App]

3. Pull healthkard-blockchain Github Repo:
    a. Navigate to the Github repository at https://github.com/KJSCE-23-3/healthkard-blockchain for the healthkard-blockchain.
    b. Clone the repository to your local machine using the git clone command or download zip.



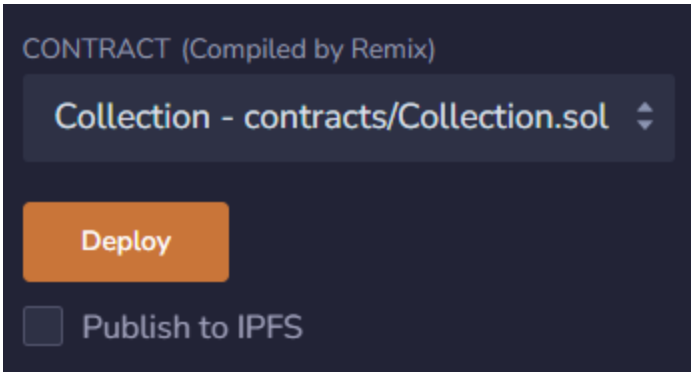
[Fig 5: Healthkard Blockchain Repository]

4. Deploy Contract on Ganache Using Remix IDE:
    a. Open the Remix IDE (https://remix.ethereum.org/) on your local machine.
    b. Open the project folder containing the healthkard-blockchain smart contracts in Remix by selecting the project folder in the file explorer.
    c. Set up the proper provider and configure the environment variables in Remix to connect it to the Ganache blockchain.



[Fig 6: Remix IDE Environment]

    d. Compile the smart contracts and deploy them on the Ganache blockchain using the "Deploy & Run Transactions" tab in Remix.

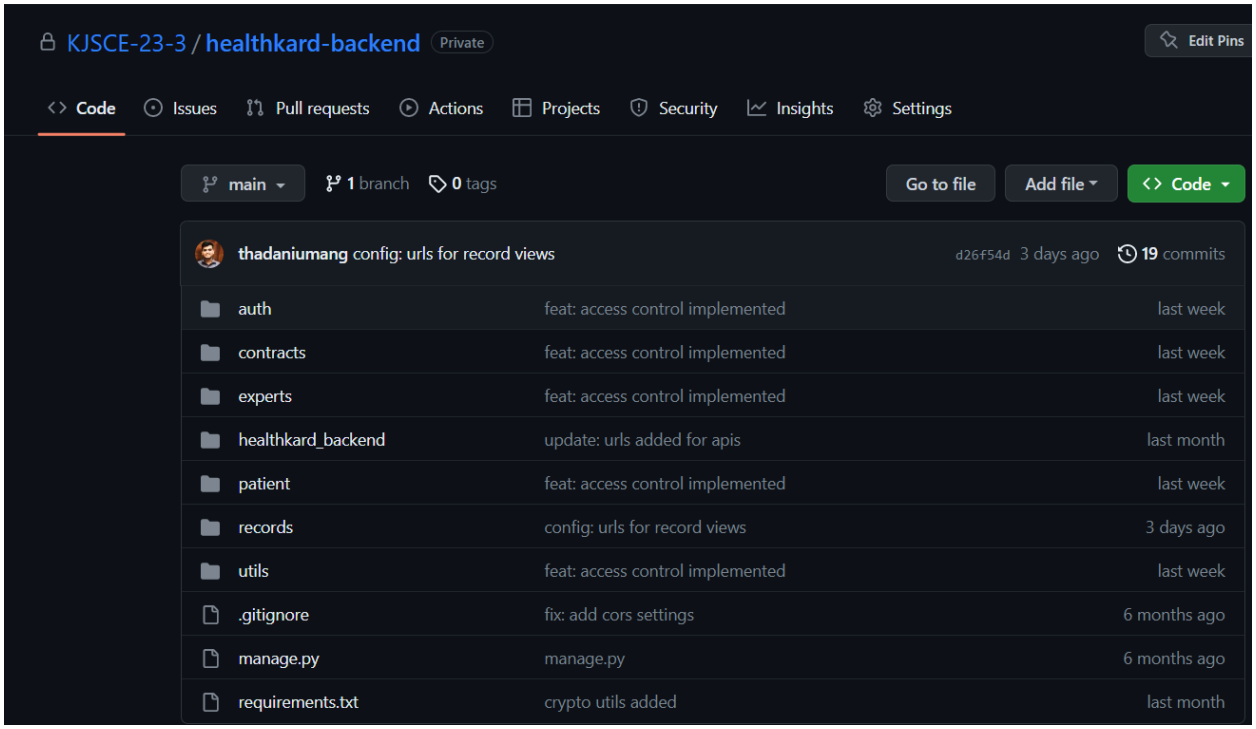


[Fig 7: Remix IDE Deploy]

e. Save the contract addresses generated during deployment for future use.

Backend

1. Download Zip or Clone Repo:
    a. Navigate to the Github repository for the healthkard-backend (https://github.com/KJSCE-23-3/healthkard-backend).



[Fig 8: Healthkard Backend Repository]

    b. Download the zip file of the repository or clone the repository to your local machine using the git clone command.

2. Set up Virtual Environment:
    a. Open a terminal or command prompt on your local machine.
    b. Navigate to the healthkard-backend folder on your local machine.
    c. Create a new virtual environment using the command: `python -m venv env`.
    d. Activate the virtual environment using the command:
        i. For Windows: `env\Scripts\activate`
        ii. For Linux/Mac: `source env/bin/activate`

3. Install Requirements:
    a. Make sure that your virtual environment is activated.
    b. Run the command `pip install -r requirements.txt` to install all the required packages and dependencies.

4. Add .env File:
    a. In the root directory of the healthkard-backend folder, create a new file named ".env".
    b. Add the contract addresses saved in step 4 during blockchain setup to the .env file as environment variables in the following format

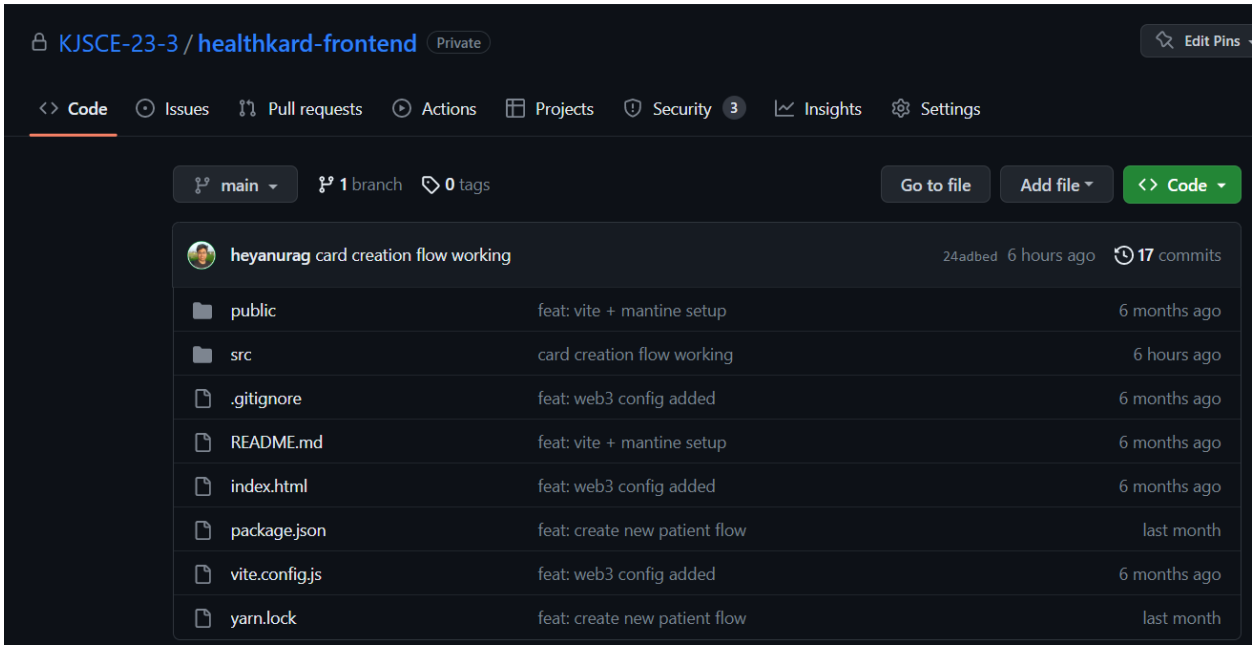
```
# Environment
ENVIRONMENT=development

# Alchemy API Key
ALCHEMY_API_KEY=


# Ganache Contracts
GANACHE_NFT_CONTRACT_ADDRESS=
GANACHE_COLLECTION_CONTRACT_ADDRESS=
GANACHE_SECRET_KEYS_CONTRACT_ADDRESS=
GANACHE_RECORDS_CONTRACT_ADDRESS=
GANACHE_REQUESTS_CONTRACT_ADDRESS=
GANACHE_SIGNER_PUBLIC_KEY=
GANACHE_SIGNER_PRIVATE_KEY=
CHAIN_ID=
```

[Fig 9: Healthkard Backend .env]

5. Run Backend:
    a. Make sure that your virtual environment is activated and the .env file is properly configured.
    b. Run the backend server using the command `python manage.py runserver`.
    c. The backend server should now be up and running. You can access the server at the specified address and port number.
    d. When True is printed on your console, it means your backend is successfully connected to your Ganache network
    e. When False is printed on your console, there is some issue with environment variables setup

Frontend

1. Clone or Download Zip:
    a. Navigate to the Github repository for the healthkard-frontend (https://github.com/KJSCE-23-3/healthkard-frontend).



[Fig 10: Healthkard Frontend Repository]

    b. Download the zip file of the repository or clone the repository to your local machine using the git clone command.

2. Install Dependencies:
    a. Open a terminal or command prompt on your local machine.
    b. Navigate to the healthkard-frontend folder on your local machine.
    c. Run the command `npm install` to install all the required packages and dependencies.

3. Configure .env File:
    a. In the root directory of the healthkard-frontend folder, create a new file named ".env".
    b. Add the following environment variables to the .env file:
        i. VITE_ALCHEMY_ID: This variable should contain your Alchemy API key. You can obtain this key from the Alchemy website (https://alchemyapi.io/).
        ii. VITE_WEB3_STORAGE: This variable should contain the name of the local storage key used by the frontend to store the user's wallet information.

iii. VITE_BACKEND_URL: This variable should contain the URL for the backend server that you set up in the previous step.

```
VITE_ALCHEMY_ID=
VITE_WEB3_STORAGE=
VITE_BACKEND_URL=http://127.0.0.1:8000
```

[Fig 11: Healthkard Frontend .env]

4. Run Frontend:
   - Make sure that the .env file is properly configured.
   - Run the frontend server using the command `npm run dev`.
   - The frontend server should now be up and running. You can access the server at the specified address and port number.

Test Cases

The major functions that would be tested are:

- Metamask Authentication
- Unique Health ID Generation
- Health and Consultation History

Features to be tested

Features	Scenarios
Metamask Authentication	• Login when the user mapped with the same address exists. • Create a new user in case a user with the same address does not exist. Verification through Aadhar has to be done in such a case.
Health ID Generation	• NFT must be generated when the user creates his/her profile • The details are not editable and NFT must be tamper-proof
Health Card Dashboard	• Show details relevant to only that particular user • Authorization is taken care of in this step

Features	Scenarios
Patient Request	• Medical Experts should be able to request patients for access to patient's health records and the same should be reflected on the Patient's account • Patient's health records should not be visible to healthcare professionals without consent.
Manage Approval	• Patients should be able to grant/revoke access to medical practitioners to access health records. • The same should be reflected on the Practitioner's Dashboard.
Adding Medical Records	• Users should be able to add medical health records linked to their Unique Health ID. • Medical experts with access should also be able to add to patients' health records.
Editing Medical Records	• Users should be able to edit medical health records linked to their Unique Health ID.

The following are the test cases that were utilized from the above table

Test Case ID	Module Name	Input	Expected Output	Actual Output	Status (Pass/-Fail)
1	Metamask Authentication	User Wallet Address	• Ensure that Metamask is connected without any issues	Metamask is successfully connected	Pass
2	Health ID Generation	• Name • Date of Birth • Blood Group • Gender • Aadhar Number • Phone Number • E-Mail • Photo	• NFT must be generated when the user creates his/her profile • The details are not editable and NFT must be tamper-proof	NFT is generated and cannot be tampered	Pass

Test Case ID	Module Name	Input	Expected Output	Actual Output	Status (Pass/-Fail)
3	Health Card Dashboard	User Wallet Address	• Show details relevant to only that particular user • Authorization is taken care of in this step	Relevant details are shown to the user	Pass
4	Patient Request	Document ID, Patient Address and Health Expert Address	• Medical Experts should be able to request patients for access to patient's health records and the same should be reflected on the Patient's account • Patient's health records should not be visible to healthcare professionals without consent.	The request flow for patient's documents works fine	Pass

Test Case ID	Module Name	Input	Expected Output	Actual Output	Status (Pass/-Fail)
5	Manage Approval	Document ID, Patient Address and Health Expert Address	• Patients should be able to grant/revoke access to medical practitioners to access health records. • The same should be reflected on the Practitioner's Dashboard.	The patients are able to grant and revoke access of EHRs	Pass
6	Adding Medical Records	Document, Wallet Address	• Users should be able to add medical health records linked to their Unique Health ID. • Medical experts with access should also be able to add to patients' health records.	Users are able to add documents to the blockchain	Pass
7	Editing Medical Records	Document ID, Patient Wallet Address	• Users should be able to edit medical health records linked to their Unique Health ID.	Users are able to edit/delete documents	Pass

Below is the list of test logs and measures taken to avoid errors, if any

Sr. No	Test Performed	Input	Expected Output	Actual Output	Pass/ Fail	Date/ Time	Measures
1	GUI Screen is Displayed	Run Command & URL Triggered	GUI is displayed properly	GUI is displayed properly	Pass	02/11/2022 18:06	—
2.	Image is Uploaded	Image of User	Image gets uploaded to IPFS	No Image in IPFS	Fail	10/11/2022 15:34	Reconfigure environment variables
3	Image is Uploaded	Image of User	Image gets uploaded to IPFS	Image gets Uploaded to IPFS	Pass	10/11/2022 16:22	—
4.	Correct User Data is Added	User Data	Token is generated successfully	Token is generated successfully	Pass	10/11/2022 16:44	—
5.	Incorrect Aadhar is added	User Data	Validation Error	Validation Test Passes	Fail	10/11/2022 17:15	Check Regex for Aadhar
6.	Incorrect Aadhar is added	User Data	Validation Error	Validation Error	Pass	10/11/2022 17:23	—

Sr. No.	Test Performed	Input	Expected Output	Actual Output	Pass / Fail	Date / Time	Measures
7	Duplicate data is added for same Aadhar	User Data	User Already Exists.	Creates a new Token	Fail	13/10/2022 21:32	Add a check in the backend
8	Duplicate data is added for same Aadhar	User Data	User Already Exists	User Already Exists	Pass	13/10/2022 22:14	—

Sr. No.	Test Performed	Input	Expected Output	Actual Output	Pass / Fail	Date / Time	Measures
9	Document is uploaded before authorization	EHR	An error saying authorization needed	An error saying authorization needed	Pass	15/03/23 18:15	—
10	EHR is uploaded after authorization	EHR	Successful upload of encrypted document	Document is uploaded but not encrypted	Fail	16/03/23 19:21	Check Encryption Algorithm
11	EHR is uploaded after authorization	EHR	Successful upload of encrypted document	Encrypted document is uploaded	Pass	16/03/23 20:35	—
12	Health Expert requests document access to Patient	Document ID	Patient receives a notification	Patient receives a notification	Pass	16/03/23 22:21	—

S.No.	Test Performed	Input	Expected Output	Actual Output	Pass / Fail	Date / Time	Measures
13	Health Experts requests same document access again	Document ID	Alert saying already have access	Patient receives notification	Fail	17/03/23 17:42	Check list of documents with access
14	Health Experts requests same document access again	Document ID	Alert saying already have access	No Alert is Observed	Fail	17/03/23 17:54	Check code for Alerts
15	Health experts requests same document access again	Document ID	Alert saying already have access	Alert saying already have access	Pass	17/03/23 18:18	—
16	Patient grants access	Document ID	Proper State is stored on Blockchain	Proper State is stored on Blockchain	Pass	17/03/23 19:31	—
17	Patient denies access	Document ID	Proper State is stored on Blockchain	Proper State is stored on Blockchain	Pass	17/03/23 19:44	—

Sr. No.	Test Performed	Input	Expected Output	Actual Output	Pass/ Fail	Date/ Time	Measures
18	Only those documents are listed to Health Experts for which they have access	—	List of Documents for which Health Experts have access	list of all documents for which Health Experts have access	Pass	20/3/23 20:33	—
19	Health Expert wants to see EHR they have access to	Document ID	Health Expert is able to see decrypted EHR	Health Expert sees Encrypted EHR	Fail	22/3/23 19:36	Check Decryption Algorithm
20	Health Expert wants to see EHR they have access to	Document ID	Health Expert is able to see decrypted EHR	Health Expert sees Decrypted EHR	Pass	23/3/23 13:47	—