

Configuration Manual

MSc Research Project
Cybersecurity

Rony Paul
Student ID: 22233717

School of Computing
National College of Ireland

Supervisor: Mark Monaghan

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Rony Paul
Student ID: x22233717
Programme: MSc in Cybersecurity **Year:** 2023-2024
Module: Configuration Manual
Lecturer: Mark Monaghan
Submission Due Date: 12/08/2024
Project Title: Securing Hospital Management Systems: Towards Decentralization and Enhanced Security with Smart Contracts
Word Count: 567 **Page Count:** 7

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Rony Paul
Date: 12/08/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Rony Paul
Student ID: 22233717

1. Introduction

This guide offers detailed instructions for setting up an Electronic Health Record (EHR) system that is blockchain-based and decentralized. System requirements, installation methods, setting up a blockchain network, and security parameters are all covered. It is intended for IT professionals working in the healthcare industry and guarantees the safe, effective, and legal handling of patient data in a private blockchain setting.

2. Configuration

Python	3.11.5
Flask	2.2.2
Web3.py	6.20.0
Truffle	V5.11.5
Ganache	7.9.1
Node.js	20.15.1

3. Implementation

- 1) Create a virtual environment and activate the environment.
- 2) Install the required packages using pip.[1]
 - Flask = 2.2.2
 - Flask-SQLAlchemy = 2.5.1
 - Flask-Migrate = 4.0.4
 - web3 = 6.2.0
 - parsimonious = 0.8.1
- 3) Create Solidity Smart Contracts: Create smart contracts for the following essential features: data retrieval, medical history updates, patient and physician registration, and patient registration.
To assemble and launch these contracts on a local blockchain emulator (Ganache), use Truffle Suite.
- 4) Deploy Contracts on Local Blockchain using Truffle and Ganache.
- 5) Create Flask Application: Set up a Flask application to serve as the backend for the EHR system.
- 6) Configure SQLite Database.

- 7) Design API Endpoints: To enable communication between the database and smart contracts, provide API endpoints.
Add endpoints for patient and physician registries, dashboard access, and medical record editing.
- 8) Develop a simple Frontend Interface: To create HTML templates, use the `render_template` method in Flask.
To create an interface that is easy to use, incorporate input forms and employ certain CSS styles.
- 9) Integrate Web3.py with Smart Contracts: To communicate with the deployed smart contracts, use Web3.py. Allow features like utilizing private keys to sign transactions and upload them to the blockchain.
- 10) Testing and Validation by Conducting many functional tests to verify that every aspect of the program functions as intended and verify the legitimacy and immutability of the transactions on the local Ganache blockchain. [2]

4. Procedure and Screenshots

- I. This image displays the complete environmental setup and command codes required to compile, deploy, and host the project

```
Administrator: C:\WINDOWS\system32\cmd.exe - sqlit3 app.db
(env) C:\Users\ronyp\EHR-System>cd..

(env) C:\Users\ronyp\EHR-System>truffle compile

Compiling your contracts...
=====
> Compiling .\contracts\EHR.sol
> Artifacts written to C:\Users\ronyp\EHR-System\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang

(env) C:\Users\ronyp\EHR-System>truffle migrate --reset
This version of @ms is not compatible with your Node.js build:

Error: Cannot find module './binaries/uws_win32_x64_127.node'
Require stack:
  - C:\Users\ronyp\AppData\Roaming\npm\node_modules\truffle\node_modules\ganache\node_modules@trufflesuite\@ms\src\src.js
  - C:\Users\ronyp\AppData\Roaming\npm\node_modules\truffle\node_modules\ganache\dist\node\core.js
  - C:\Users\ronyp\AppData\Roaming\npm\node_modules\truffle\build\migrate.bundled.js
  - C:\Users\ronyp\AppData\Roaming\npm\node_modules\truffle\node_modules\original-require\index.js
  - C:\Users\ronyp\AppData\Roaming\npm\node_modules\truffle\build\cli.bundled.js
  - C:\Users\ronyp\AppData\Roaming\npm\node_modules\truffle\build\cli.bundled.js
Falling back to a NodeJS implementation; performance may be degraded.

Compiling your contracts...
=====
> Compiling .\contracts\EHR.sol
> Artifacts written to C:\Users\ronyp\EHR-System\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang

(env) C:\Users\ronyp\EHR-System>npm update

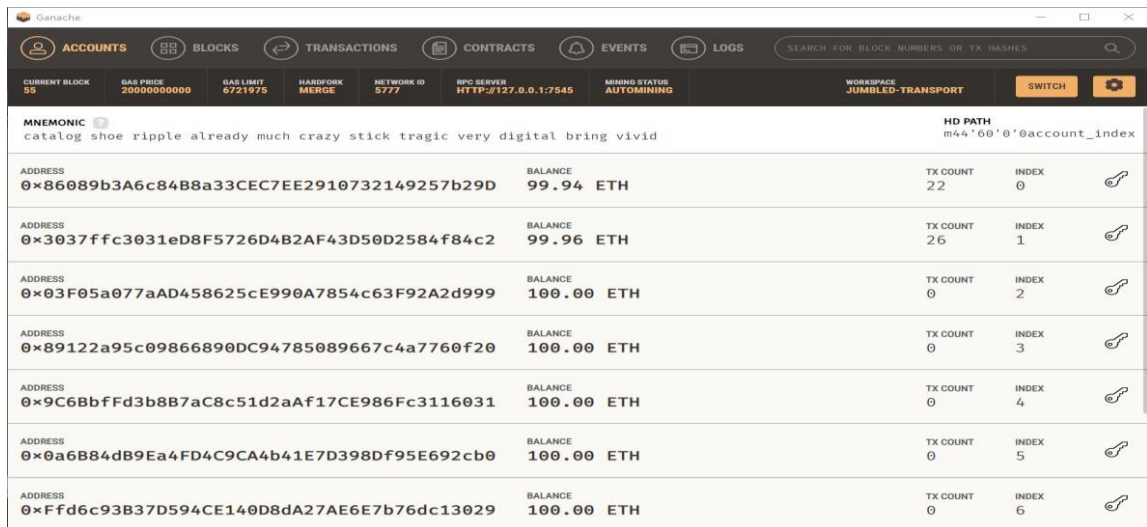
up to date, audited 4 packages in 53s

found 0 vulnerabilities

(env) C:\Users\ronyp\EHR-System>flask run

Connected to Ganache
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [11/Aug/2024 12:56:44] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2024 12:56:44] "GET /static/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [11/Aug/2024 12:56:45] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [11/Aug/2024 12:56:56] "GET /register_doctor HTTP/1.1" 200 -
127.0.0.1 - - [11/Aug/2024 12:56:56] "GET /static/styles.css HTTP/1.1" 304 -
```

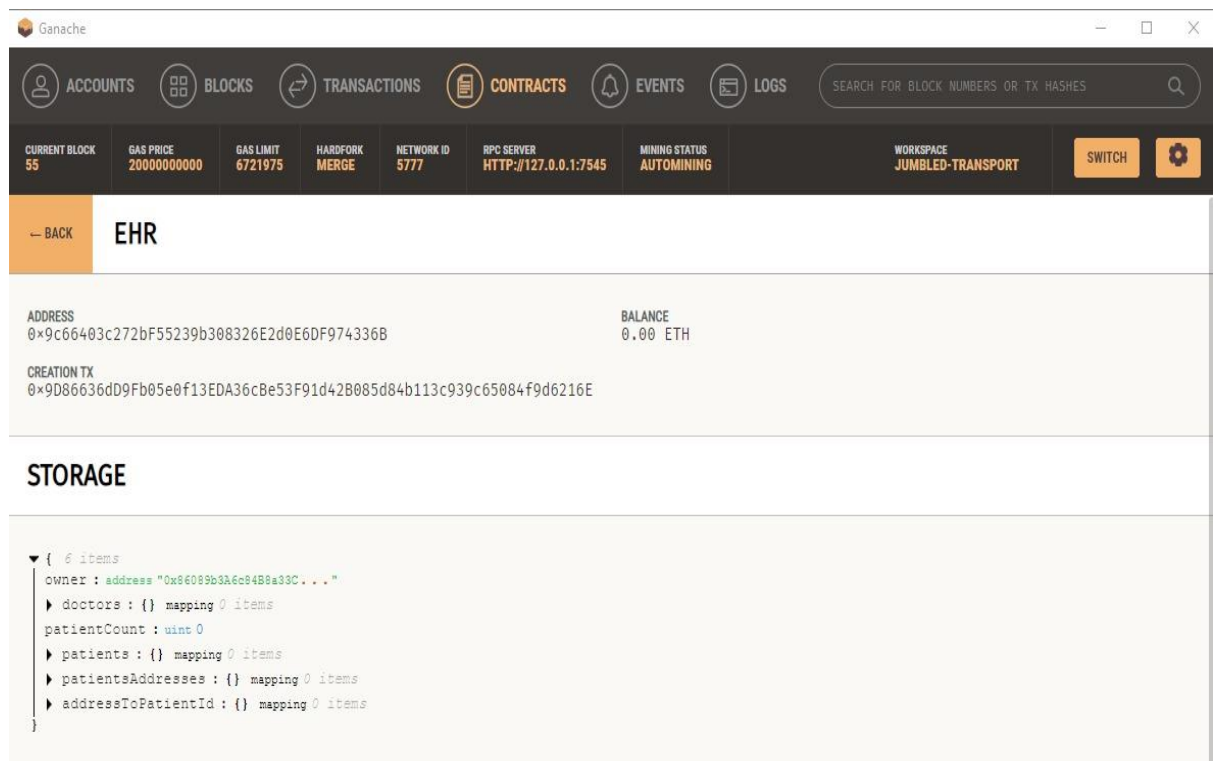
- II. This image shows the Ganache interface which was used for testing blockchain as it provided crypto ETH with its address and private key.



The screenshot shows the Ganache application window. The top navigation bar includes tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below the navigation bar, there's a status bar with various metrics like CURRENT BLOCK, GAS PRICE, GAS LIMIT, HARDFORK, NETWORK ID, RPC SERVER, MINING STATUS, and WORKSPACE. The main content area displays a list of accounts with columns for ADDRESS, BALANCE, TX COUNT, and INDEX. The mnemonic phrase is also visible at the top.

ADDRESS	BALANCE	TX COUNT	INDEX
0x86089b3A6c84B8a33CEC7EE2910732149257b29D	99.94 ETH	22	0
0x3037ffc3031eD8F5726D4B2AF43D50D2584f84c2	99.96 ETH	26	1
0x03F05a077aAD458625cE990A7854c63F92A2d999	100.00 ETH	0	2
0x89122a95c09866890DC94785089667c4a7760f20	100.00 ETH	0	3
0x9C6BbfFd3b8B7aC8c51d2aAf17CE986Fc3116031	100.00 ETH	0	4
0x0a6B84dB9Ea4FD4C9CA4b41E7D398Df95E692cb0	100.00 ETH	0	5
0xFfd6c93B37D594CE140D8dA27AE6E7b76dc13029	100.00 ETH	0	6

- III. This interface shows the details of the owner after the successful deployment of the contract to the blockchain environment. Here address, storage, and mapping of the data can be monitored.



The screenshot shows the Ganache application window with the CONTRACTS tab selected. The contract name is EHR. The interface displays the ADDRESS, BALANCE, and CREATION TX. Below this, the STORAGE section shows the contract's state, including owner, doctors, patientCount, patients, patientsAddresses, and addressToPatientId.

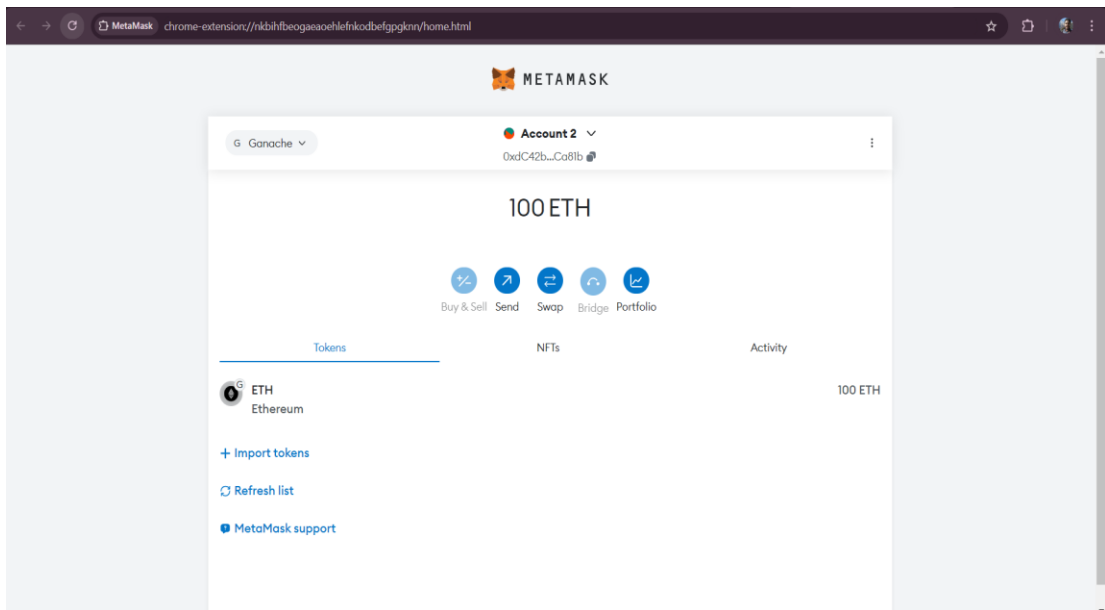
Contract Details:

- Contract Name: EHR
- Address: 0x9c66403c272bf55239b308326E2d0E60F974336B
- Balance: 0.00 ETH
- Creation TX: 0x9D86636dD9Fb05e0f13EDA36c8e53F91d42B085d84b113c939c65084f9d6216E

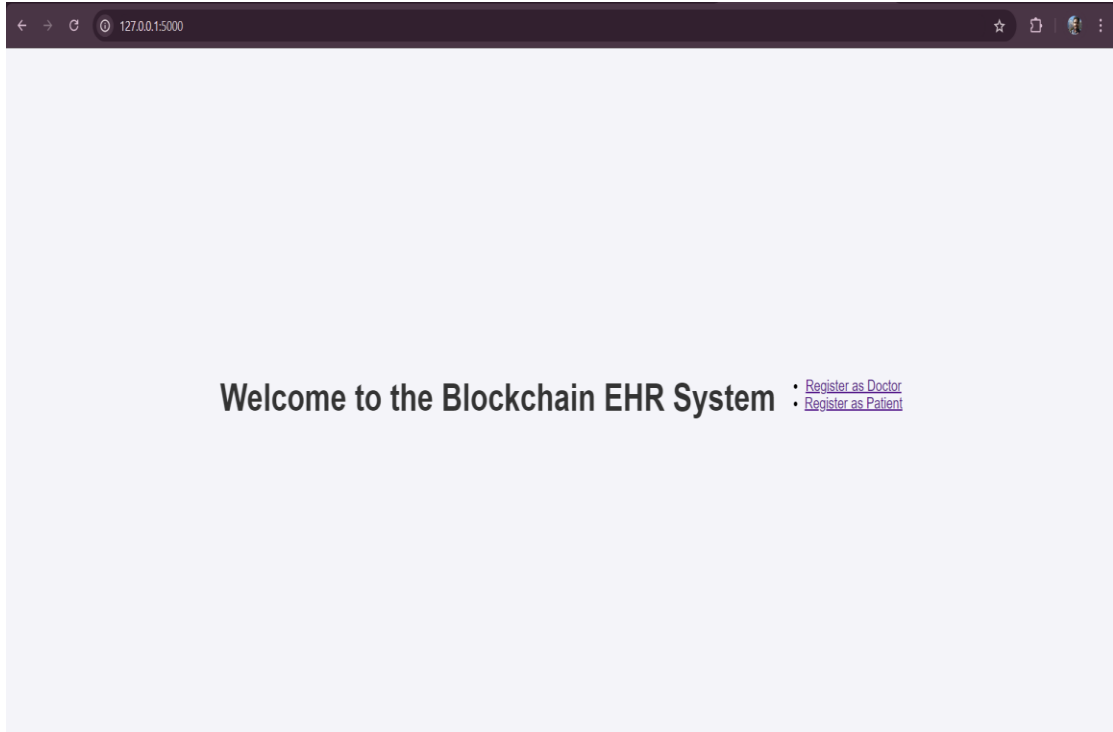
Storage:

```
{
  6 items
  owner : address "0x86089b3A6c84B8a33C..."
  doctors : {} mapping 0 items
  patientCount : uint 0
  patients : {} mapping 0 items
  patientsAddresses : {} mapping 0 items
  addressToPatientId : {} mapping 0 items
}
```

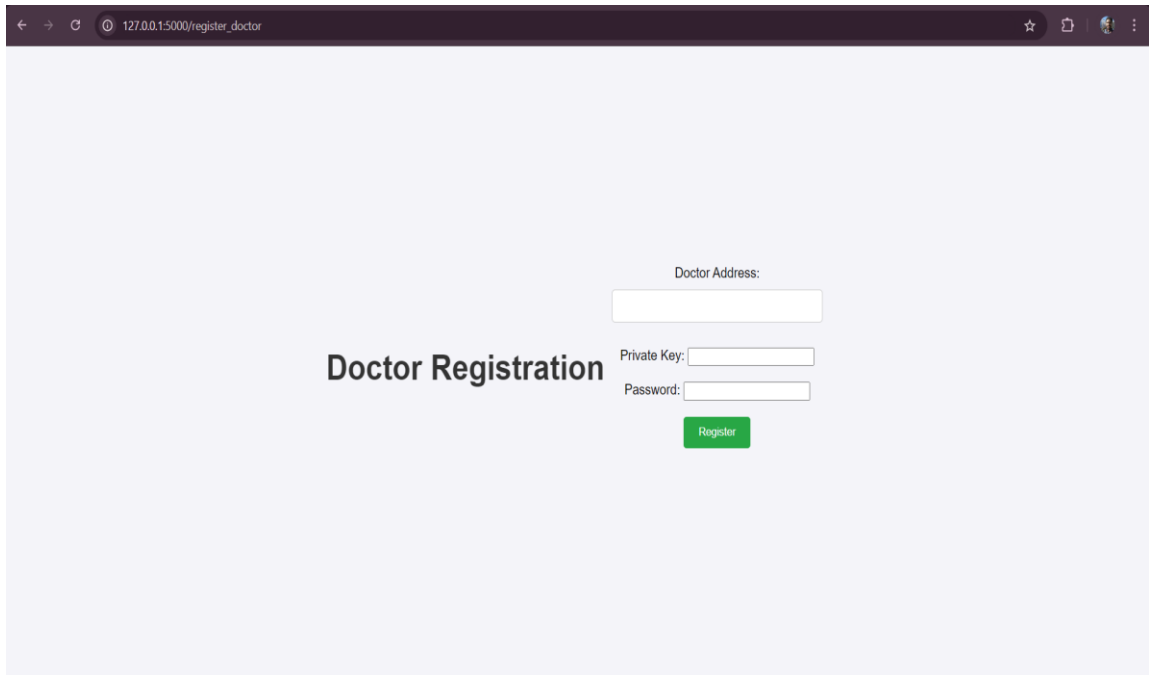
- IV. Make sure your MetaMask is connected to Ganache and is linked to the local host network.



- V. Frontend of the Project where Admin/owner could handle this page and insert, update, and delete the doctor's and patient's details.

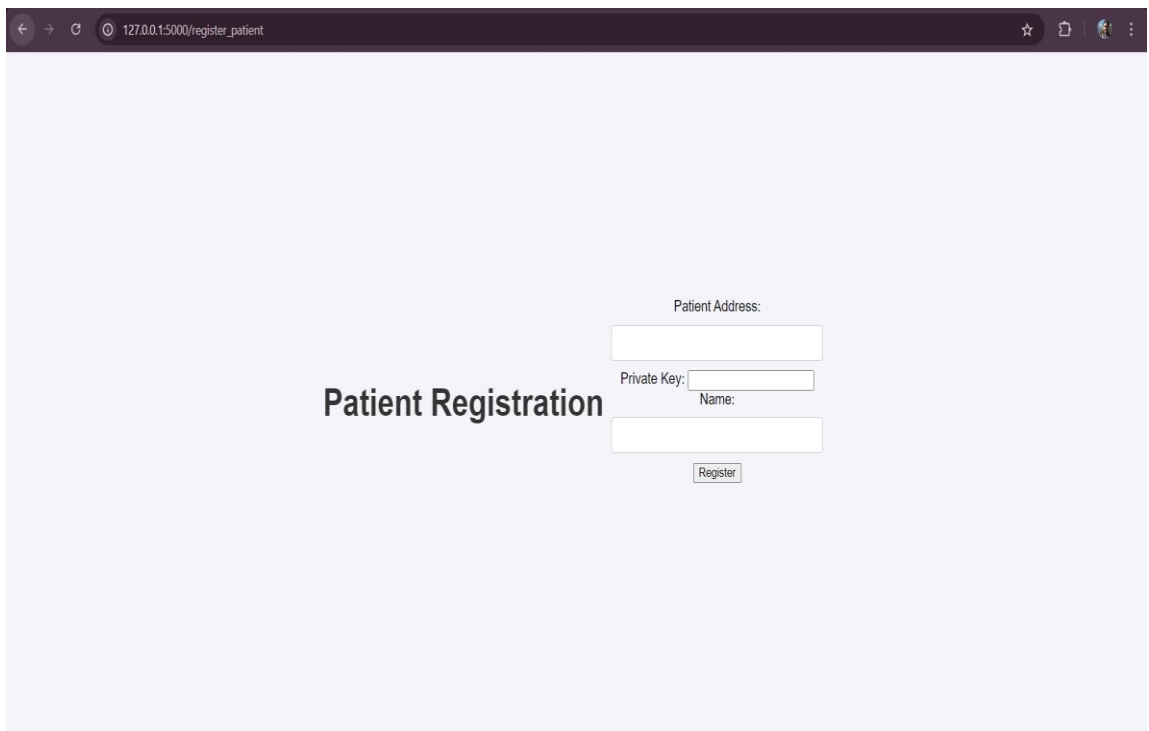


VI. The Doctor's Registration interface and its form to insert a doctor's detail:



A screenshot of a web browser displaying the 'Doctor Registration' interface. The browser's address bar shows '127.0.0.1:5000/register_doctor'. The page has a light purple background. On the left, the text 'Doctor Registration' is displayed in a bold, dark font. To the right, there is a registration form with three input fields: 'Doctor Address:', 'Private Key:', and 'Password:'. Below these fields is a green button labeled 'Register'.

VII. The Patient's Registration interface and its form to insert a patient's details:



A screenshot of a web browser displaying the 'Patient Registration' interface. The browser's address bar shows '127.0.0.1:5000/register_patient'. The page has a light purple background. On the left, the text 'Patient Registration' is displayed in a bold, dark font. To the right, there is a registration form with three input fields: 'Patient Address:', 'Private Key:', and 'Name:'. Below these fields is a button labeled 'Register'.

VIII. Doctor’s Dashboard where a doctor can detail a patient's disease and prescriptions and pass the private key. Submitted values with the key will be displayed, this is only viewed by the owner and doctor.

A screenshot of a web browser displaying the 'Doctor Dashboard' form. The browser's address bar shows '127.0.0.1:5000/doctor_dashboard'. The form is centered on a light purple background and contains the following fields and labels:

- Doctor Address:** A text box containing the hexadecimal string '0xdC42bEd3BBa2BF0EA7225F9C4Fc1326b'.
- Patient Name:** A text box containing the name 'Rony'.
- Disease:** A text box containing the word 'Fever'.
- Medication:** A text box containing the word 'Paracetamol'.
- Private Key:** A text box filled with dots, representing a masked private key.
- Submit:** A button located below the Private Key field.

The title 'Doctor Dashboard' is displayed in a large, bold, black font to the left of the form fields.

A screenshot of the same 'Doctor Dashboard' web page, but with a 'Submitted Values' section added. The form fields and labels are the same as in the previous screenshot. The 'Submitted Values' section is located to the right of the form fields and displays the following data:

Doctor Address:	Patient Name:	Disease:	Medication:	Private Key:
0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b	Rony	Fever	Paracetamol	0x6fb97044e9cc00bdafdc1bbe9e529543629ce3ff197b457e39f32270f6

The 'Submitted Values' section is titled 'Submitted Values' in a bold, black font. The data is presented in a table format with five columns corresponding to the form fields. The 'Private Key' value is a long hexadecimal string.

- IX. The Patient can view the disease details and prescription, they just need to know the doctor's name and the patient's address at which mapping is done.

Patient Address:

Doctor Name:

Patient Name:

Submit

Patient Dashboard Medical History Patient Name: Jacob Disease: Vericous Medication: laser

- X. The patient data is getting stored in the database using SQLite.

```
(env) C:\Users\ronyp\EHR-System>sqlite3 app.db
SQLite version 3.46.0 2024-05-23 13:25:27 (UTF-16 console I/O)
Enter ".help" for usage hints.
sqlite> select * from patient;
1|Sam|Asthma|Inhaler|0x3037ffc3031eD8F5726D4B2AF43D50D2584f84c2
2|Daniel|Cough|Syrup|0x3037ffc3031eD8F5726D4B2AF43D50D2584f84c2
3|Samuel|Heart Disease|Deplat|0x3037ffc3031eD8F5726D4B2AF43D50D2584f84c2
4|Denson|Fever|Paracetamol|0x3037ffc3031eD8F5726D4B2AF43D50D2584f84c2
5|Mitchel marsh|Asthma|Inhaler|0x3037ffc3031eD8F5726D4B2AF43D50D2584f84c2
6|Pamela|Cancer|CART CELL|0x3037ffc3031eD8F5726D4B2AF43D50D2584f84c2
7|Rony|Fever|Dolo|22233717
8|Philip|Lung Conjestion|Cough syrup|22233717
9|Rony|Fever|Paracetamol|22233717
10|Ann|Stomach pain|Glucose|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
11|Ashik|Diabetes|Roseday 50mg|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
12|||
13|Jithin|BP|Meditation|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
14|Rahul|Allergy|citric|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
15|Francis|KK|Adi|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
16|Dani|Fatigue|Exercise|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
17|Geoshin|Vein|Adi|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
18|Anand|Sleepillness|Gastric|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
19|Anand|Sleepillness|Gastric|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
20|Frank|Fever|Syrup|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
21|David|Covid|Rest|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
22|Jacob|Vericous|laser|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
23|Shannan|Alcoholic|deaddiction|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
24|Shannan|Alcoholic|deaddiction|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
25|George|Flew|mask|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
26|Ram|Allergy|Soap|0xdC42bEd3BBa2BF0EA7225F9C4Fc1326eb17Ca81b
sqlite> Program interrupted.
```

5. Reference

- [1] “The Python Standard Library,” Python documentation. Accessed: Aug. 12, 2024. [Online]. Available: <https://docs.python.org/3/library/index.html>
- [2] N. Satrio, S. Sukaridhoto, U. Al Rasyid, R. Putri Nourma Budiarti, I. Al-Hafidz, and E. Fajrianti, “Blockchain integration for hospital information system management,” *Bali Med. J.*, vol. 11, pp. 1195–1201, Sep. 2022, doi: 10.15562/bmj.v11i3.3540.