Introduction

Welcome to the doc for CSS453_SEiMCS setup tutorial.
The link to Github: https://github.com/xxth0/CSS453_SEiMCS

Prerequisites

In this setup, I'll provide Windows setup guides. The **required software** are

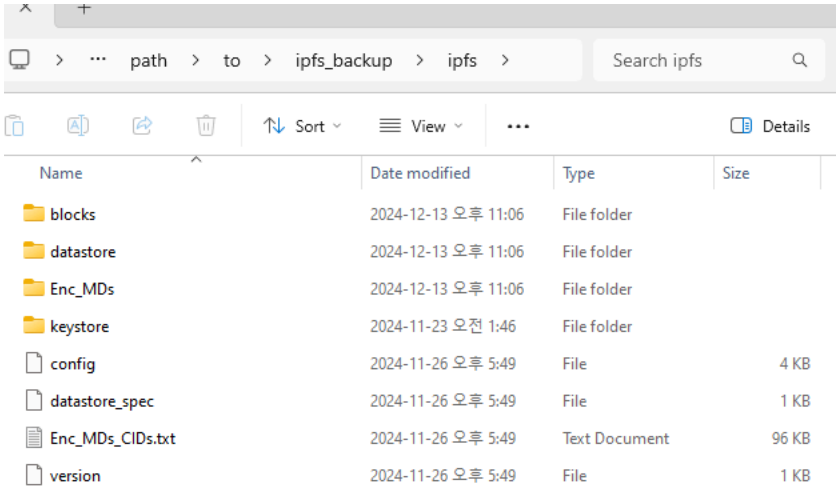
1. **Visual Studio Community** (or any Development Environment)
2. **Ganache** A local Blockchain for development
3. **Docker Desktop** A local IPFS for development
4. **Node Js (v14 - 18)** This one is used for compiling Smart Contract using Truffle library and using a method like migrate to deploy. **Use version 14 - 18 only!**
5. **Python 3** Code language for this project.

The required libraries are

-npm install -g truffle
-pip install flask
-pip install hashlib
-pip install pybloom-live
-pip install web3
-pip install pycryptodome
-pip install fuzzywuzzy
-pip install werkzeug
-pip install requests
-pip install python-dotenv

Docker Setup

Step 1: Find a folder **Local_Docker** in the **SEiMCS** folder that you have downloaded. Go into the deepest of the sub-folder until finding the **ipfs** content folder which looks like this.



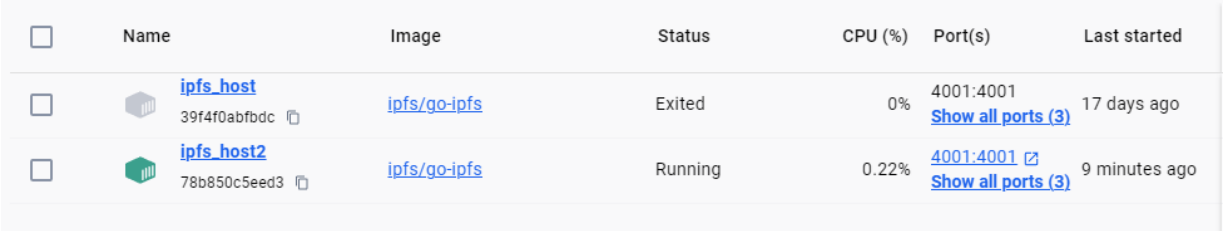
Copied the path that can be used to create a container from backup. In my case **C:\Users\WINDOWS\Documents\CSS453_SEiMCS\Local_Docker\path\to\ipfs_backup\ipfs**

Step 2: Open **Powershell** (Not command prompt) replace the docker command with

```
docker run -d --name ipfs_host `
    -v REPLACE WITH YOUR NEW DIRECTORY FROM STEP 1:/data/ipfs `
    -p 4001:4001 `
    -p 5001:5001 `
    -p 8080:8080 `
    ipfs/go-ipfs
```

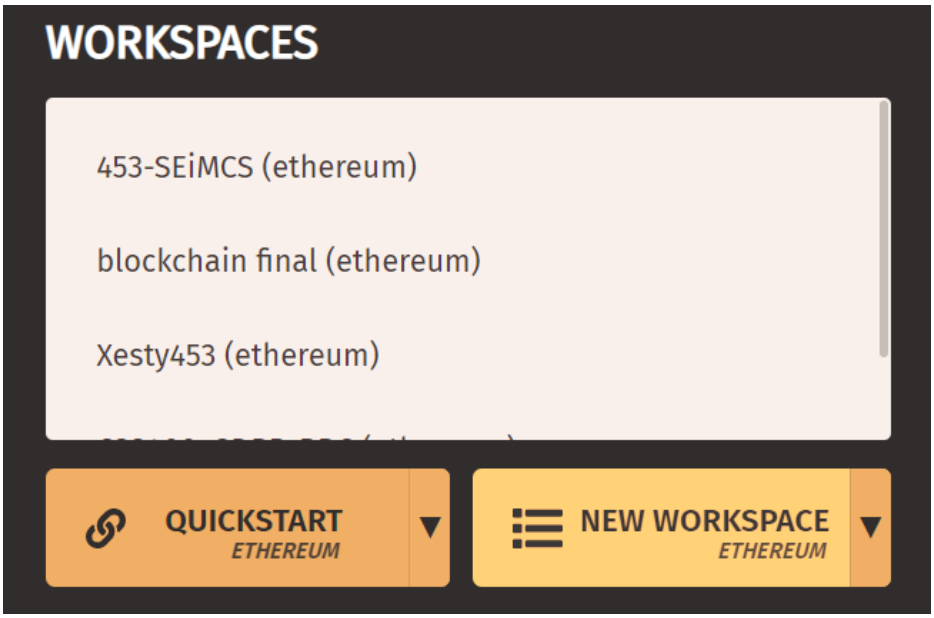
```
PS C:\Users\WINDOWS> docker run -d --name ipfs_host2 `
>>     -v C:\Users\WINDOWS\Documents\CSS453_SEiMCS\Local_Docker\path\to\ipfs_backup\ipfs:/data/ipfs `
>>     -p 4001:4001 `
>>     -p 5001:5001 `
>>     -p 8080:8080 `
>>     ipfs/go-ipfs
fe88743ec75fc45fbadf92cf9936ffa48d78b7e5c4d5d8c78b137630e1f292a6
PS C:\Users\WINDOWS>
```

If the command is executed correctly, it will return a hash like in this image above. And in **Docker**, you should see a new container up and running.

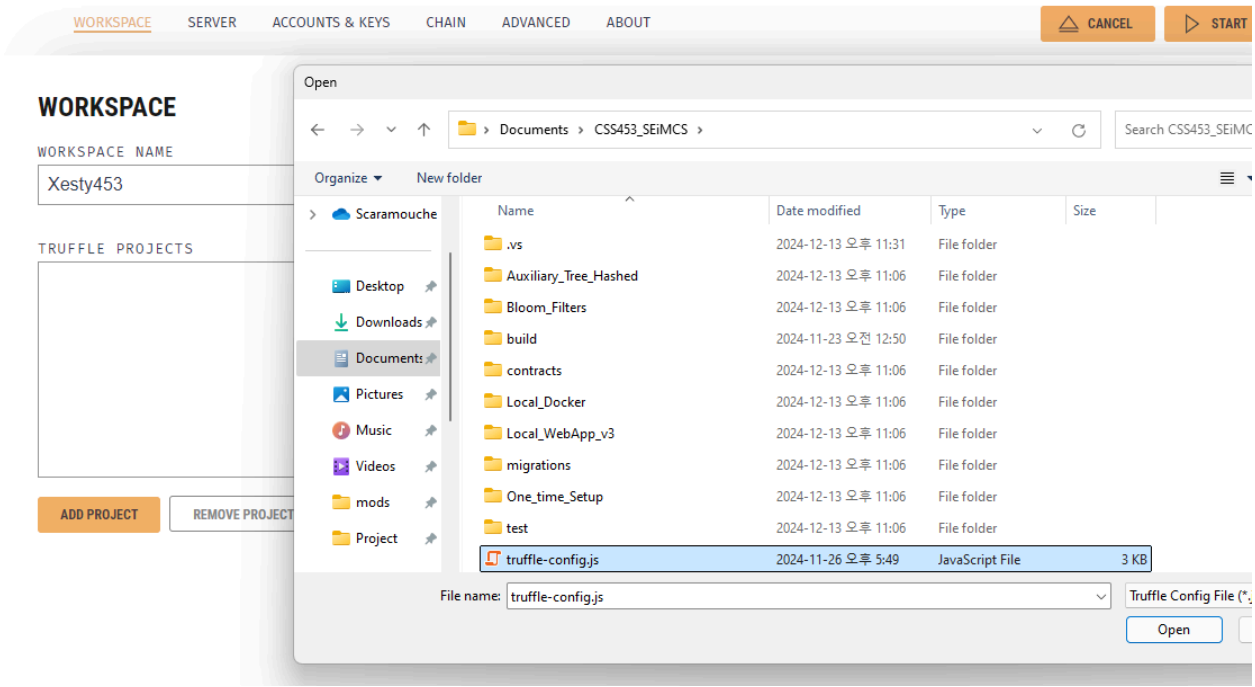


Ganache Setup

Step 1: Create a new workspace using the **New Workspace** option (Quickstart won't save the project).



Step 2: Define Workspace name, in this case I choose **Xesty453**. And in **SEiMCS** folder, choose **Add Project** and then add **truffle-config.js** into the workspace. This will help keep track of smart contract deploy addresses for easy access. Once done, you can start the project. Other options can leave as it is.



Any deployed Smart Contracts are shown here

Step 3: Deploy Smart Contract in Visual Studio code using command

---

truffle compile

```
PS C:\Users\WINDOWS\Documents\CSS453_SEiMCS> truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\AuthContract.sol
> Compiling .\contracts\StoreCIDs.sol
> Artifacts written to C:\Users\WINDOWS\Documents\CSS453_SEiMCS\build\contracts
> Compiled successfully using:
   - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang
```

---

truffle migrate --network development

```
PS C:\Users\WINDOWS\Documents\CSS453_SEiMCS> truffle migrate --network development

Compiling your contracts...
===========================
> Compiling .\contracts\AuthContract.sol
> Compiling .\contracts\StoreCIDs.sol
> Artifacts written to C:\Users\WINDOWS\Documents\CSS453_SEiMCS\build\contra
                                                                        acts
> Compiled successfully using:
   - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      5777                                        605e0688bfe7e4f9e
```

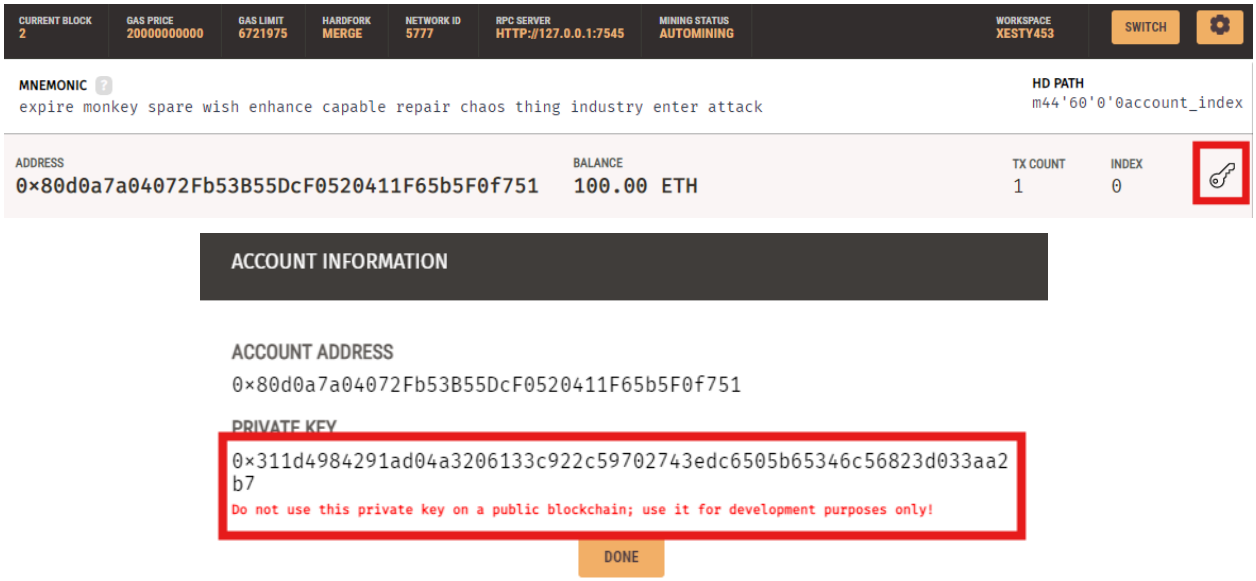
Step 4: Verify if the Smart Contract is deployed correctly and the Contract Address is obtained.

Step 5: In the **.env** file, change all the info to match where you create a project, including the Contract Address, Private Key, ABI file path, etc.

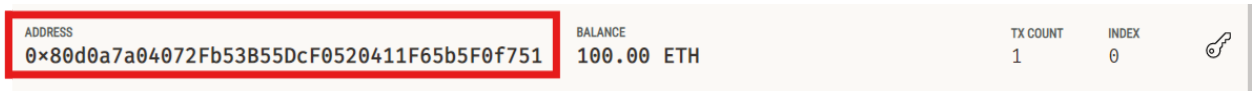
Example of .env file in One_Time_Setup

```
# URL for connecting to Ganache blockchain
GANACHE_URL=http://127.0.0.1:7545

# Address of the deployed smart contract
CONTRACT_ADDRESS=0x18C15bCCf46d47568c6ef2cA53ac686A7f388E7E
AUTH_CONTRACT_ADDRESS=0x9251592602FEa9B53E4dD7d1413a9dD33658CDfd

# Path to the ABI JSON file
ABI_FILE_PATH=C:\Users\WINDOWS\Documents\CSS453_SEiMCS\build\contracts\StoreCIDs.json
AUTH_ABI_FILE_PATH=
C:\Users\WINDOWS\Documents\CSS453_SEiMCS\build\contracts\AuthContract.json

# Private key of the account used for signing transactions
PRIVATE_KEY=0x311d4984291ad04a3206133c922c59702743edc6505b65346c56823d033aa2b7

# File containing HPID to CID mappings (This one doesn't need change)
FILE_PATH=HPIDs to CIDs.txt

# Default chain ID (1337 for Ganache)
CHAIN_ID=1337

# REGISTRATION #
# User's public address for registration
USER_ADDRESS=0x80d0a7a04072Fb53B55DcF0520411F65b5F0f751

# Default password for registration (example)
DEFAULT_PASSWORD=password123

# IGNORE #
# PID 1000 Bug, have to manual import
PID=1000
CID=QmTP86FfrtE1atwdEw7bH5k68piNVox4NtpsxdxzKkCh7p
```

1). Replace **CONTRACT_ADDRESS** with the **StoreCIDs** contract address you've obtained, in this case mine is **0x18C15bCCf46d47568c6ef2cA53ac686A7f388E7E**
2). Similarly **AUTH_CONTRACT_ADDRESS** with the **AuthContract** contract address.
3). **ABI_FILE_PATH** and **AUTH_ABI_FILE_PATH** are the .json files of the contract we compiled and deployed.
4). **PRIVATE_KEY** is obtained from the 1st Blockchain account via key icon right here.



5). Lastly, **USER_ADDRESS** is their public address.



6). Run all three .py files with typical Python command, these will done the following,

- Uploading 1000 transactions of PIDs mapped with CIDs to the Blockchain
- Create a user account for the 1st user that can use it for our web applications.


```
python bcScript.py
python pid1000.py
python registeruser.py
```

```
PS C:\Users\WINDOWS\Documents\CSS453_SEiMCS\One_time_Setup> python registeruser.py
Connected to Blockchain
Registering user: 0x80d0a7a04072Fb53B55DcF0520411F65b5F0f751
Generated Password Hash: b'\rE\xe1\x97f\xc0\xca\xdf\xe3\xafH\xb8\x01\x10*\x9d\xe43~\x
Transaction Hash: ded55b9720fdff00005a15382cad5df7097958b32ed11294761ca4ed2992bfc0
Transaction Receipt: AttributeDict({'transactionHash': HexBytes('0xded55b9720fdff0000
h': HexBytes('0xc34192a5a71ae13bbe72ee0f4a7d147ed44b37fdadd366f5cd3f5148d1f7f78d'),
```

Example of .env file in Local_WebApp_v3

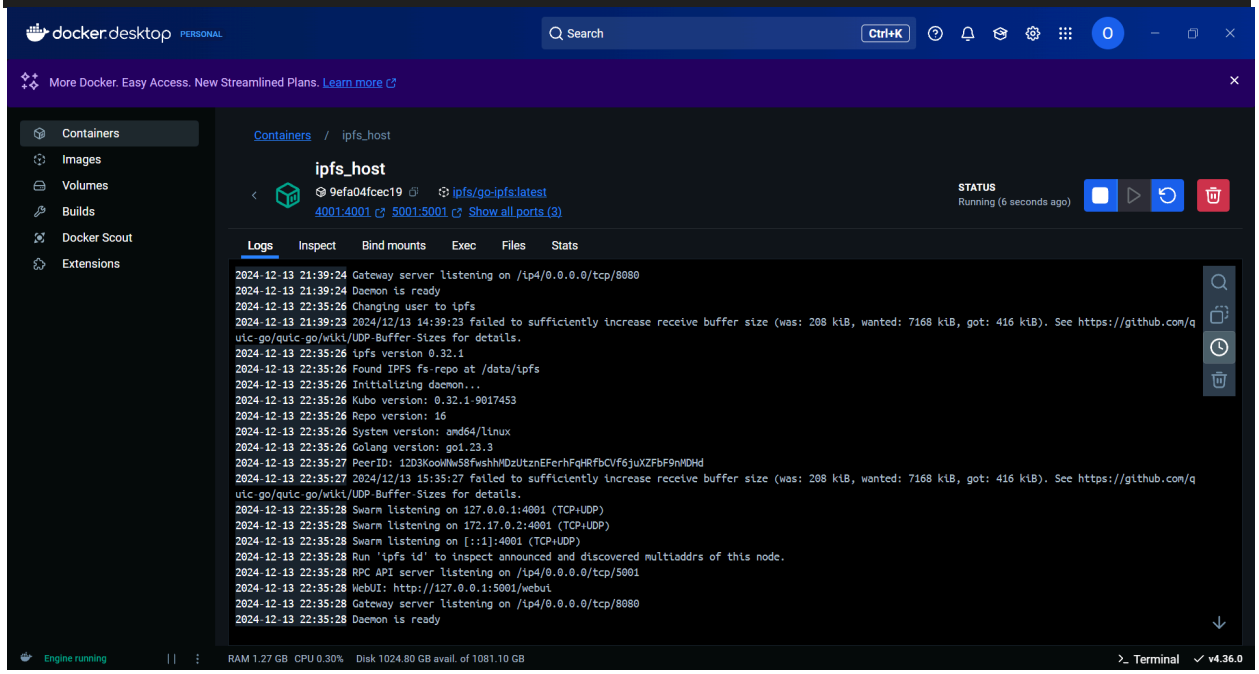
```
# Flask Secret Key
FLASK_SECRET_KEY=your-flask-secret-key

# Blockchain connection URL
GANACHE_URL=http://127.0.0.1:7545

# Smart contract details
SEARCH_CONTRACT_ADDRESS=0x18C15bCCf46d47568c6ef2cA53ac686A7f388E7E
AUTH_CONTRACT_ADDRESS=0x9251592602FEa9B53E4dD7d1413a9dD33658CDfd

# ABI file paths
ABI_FILE_PATH=C:\Users\WINDOWS\Documents\CSS453_SEiMCS\build\contracts\StoreCIDs.json
AUTH_ABI_FILE_PATH=
C:\Users\WINDOWS\Documents\CSS453_SEiMCS\build\contracts\AuthContract.json

# Auxiliary directories
AUXILIARY_PATH=C:\Users\WINDOWS\Documents\CSS453_SEiMCS\Auxiliary_Tree_Hashed
BLOOM_OUTPUT_PATH=C:\Users\WINDOWS\Documents\CSS453_SEiMCS\Bloom_Filters

# File storage directories
UPLOAD_FOLDER=uploads
DECRYPTED_FOLDER=decrypted

# IPFS Gateway
IPFS_GATEWAY=http://127.0.0.1:8080/ipfs/
```

1). Replace **CONTRACT_ADDRESS** with the **StoreCIDs** contract address.
2). Similarly **AUTH_CONTRACT_ADDRESS** with the **AuthContract** contract address.
3). **ABI_FILE_PATH** and **AUTH_ABI_FILE_PATH** are the .json files of the contract we compiled and deployed.
4). **AUXILIARY_PATH** and **BLOOM_OUTPUT_PATH** are the folders path that store the the setting of Auxiliary tree and Bloom filter respectively.

Step 6: Run
1). Check if the Docker is up and running, if not, start Docker either via the UI or with a

command,

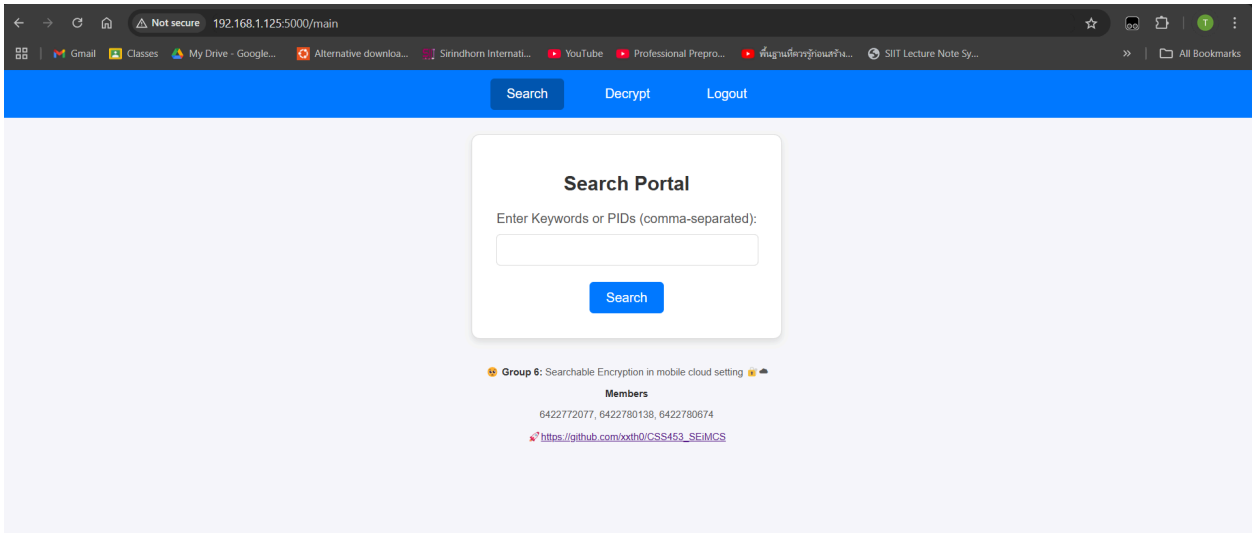
```
Docker start <CONTAINER_ID>
```



2). Go to **Local_WebApp_v3** directory in the terminal, then run the command
python appsearchEnc.py

```
sktop\O\Cyber_crime_project\CSS453_SEiMCS-main\Local_WebApp_v3> python appsearchEnc.py
C:\Users\roblu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-packages\fuzzywuzzy
\fuzz.py:11: UserWarning: Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning')
Connected to Blockchain
 * Serving Flask app 'appsearchEnc'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.125:5000
Press CTRL+C to quit
 * Restarting with stat
C:\Users\roblu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-packages\fuzzywuzzy
\fuzz.py:11: UserWarning: Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning')
Connected to Blockchain
 * Debugger is active!
 * Debugger PIN: 362-805-259
```

3). Then use the displayed ip address to access to searching webapp
There are 2 IP addresses that were shown, the **127.0.0.1:5000** is available only for this local computer (The one that starts the server), and **192.168.1.125:5000** is an IP address website for any devices in the same WiFi SSD to access, including mobile devices.


Not secure
192.168.1.125:5000/main
Gmail
Classes
My Drive - Google...
Alternative downloa...
Sirindhorn Internati...
YouTube
Professional Prepro...
พื้นฐานที่ควรรู้ก่อนสร้าง...
SIIT Lecture Note Sy...
All Bookmarks
Search
Decrypt
Logout
Search Portal
Enter Keywords or PIDs (comma-separated):
Search
Group 6: Searchable Encryption in mobile cloud setting
Members
6422772077, 6422780138, 6422780674
https://github.com/xxth0/CSS453_SEiMCS