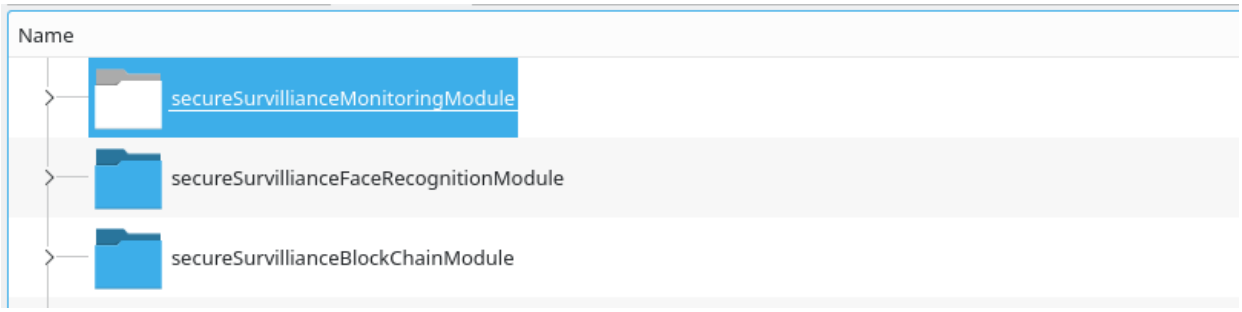
Secure System for Person Recognition using Blockchain and Deep Learning

Steps to execute:

1. Install Prerequisite:
   Install below:
   A) Python 3.10
   B) Anaconda
   C) Pip
   D) Flask~=1.1
   E) requests~=2.22
   F) opencv
   G) jproperties
2. System/machine details:
   A) Number of machine >= 2 (to create physical nodes)
   B) OS = Windows, Ubuntu
3. Get the latest code.



4. Train the model:
   A) Go to secureSurvillianceFaceRecognitionModule and run MainVideoToImgForPreparingDataSet.py

```
sarwar@sarwar-hp:~/Downloads/code/secureSurvillianceFaceRecognitionModule$ python MainVideoToImgForPreparingDataSet.py
./trainingImages/tmp/frame0.jpg saved
./trainingImages/tmp/frame1.jpg saved
./trainingImages/tmp/frame2.jpg saved
./trainingImages/tmp/frame3.jpg saved
./trainingImages/tmp/frame4.jpg saved
./trainingImages/tmp/frame5.jpg saved
./trainingImages/tmp/frame6.jpg saved
./trainingImages/tmp/frame7.jpg saved
./trainingImages/tmp/frame8.jpg saved
./trainingImages/tmp/frame9.jpg saved
```

   B) It will capture the video and save in image form in trainingImages/tmp folder.
   C) Rename this folder by a next id eg 3.
   D) Run MainTrainModel.py to train the model.

```
sarwar@sarwar-hp:~/Downloads/code/secureSurvillianceFaceRecognitionModule$ python MainTrainModel.py
--------------------------------------------------
path :  trainingImages
subdirnames :  ['1', '2', '0']
filenames :  []
--------------------------------------------------
--------------------------------------------------
path :  trainingImages/1
subdirnames :  []
filenames :  ['dc_Cover_fuvdkphnaid742mnsdmuq60i16_20171210110652.Medi_0.jpeg', 'kangana_saree_look_anita_
ld_2019_02_da01d787_7539_45d2_b1a2_e9969b8071ea_Kanagana_ranaut.jpg', '05_07_2018_kng_shdi_1_18160994.jpg
t_6.jpg', 'kangnaranaut001.jpg', 'kangana_ranaut_1553246965.jpeg', 'Kangana_Ranaut_rare_pictures.jpg', 'a
edi.jpeg', 'aa_Cover_567vnisio43sb4ngroftgfljf1_20180619225555.Medi.jpeg', 'kangana_ranaut_1524646227.jpe
a3227.image.jpg', 'have_been_harassed_by_actors_on_sets__says_kangana_ranaut_2019_01_21.jpg', 'Kan.jpg',
```

E) It will create a YML file with file name “trainingData.yml”.

5. Create node using Node Server module:

A) Go to secureSurvillianceBlockChainModule and run MainNodeServer.py to create a node.

```
sarwar@sarwar-hp:~/Downloads/code/secureSurvillianceBlockChainModule$ python MainNodeServer.py
---------Blockchain.init()-----------
---------Blockchain.create_genesis_block()-----------
---------Block.__init__()-----------
---------Block.compute_hash()-----------
block_string:  {"index": 0, "nonce": 0, "previous_hash": "0", "timestamp": 0, "transactions": []}
 * Serving Flask app "MainNodeServer" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:8000/ (Press CTRL+C to quit)
 * Restarting with stat
---------Blockchain.init()-----------
---------Blockchain.create_genesis_block()-----------
---------Block.__init__()-----------
---------Block.compute_hash()-----------
block_string:  {"index": 0, "nonce": 0, "previous_hash": "0", "timestamp": 0, "transactions": []}
 * Debugger is active!
 * Debugger PIN: 299-716-647
```

B) Apply above steps to other machines also to create a new node in different machine.

```
┌──(sarwar㉿kali)-[~/Downloads/code/secureSurvillianceBlockChain]
└─$ python MainNodeServer.py
---------Blockchain.init()-----------
---------Blockchain.create_genesis_block()-----------
---------Block.__init__()-----------
---------Block.compute_hash()-----------
block_string:  {"index": 0, "nonce": 0, "previous_hash": "0", "timestamp": 0, "transactions": []}
 * Serving Flask app 'MainNodeServer' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://192.168.97.228:8000/ (Press CTRL+C to quit)
 * Restarting with stat
---------Blockchain.init()-----------
---------Blockchain.create_genesis_block()-----------
---------Block.__init__()-----------
---------Block.compute_hash()-----------
block_string:  {"index": 0, "nonce": 0, "previous_hash": "0", "timestamp": 0, "transactions": []}
 * Debugger is active!
 * Debugger PIN: 663-547-753
```

C) Don’t close it.

6. Synchronize Node:

A) Note down both machine IP addresses.

```
sarwar@sarwar-hp:~/Downloads/code/secureSurvillianceBlockChainModule$ ifconfig wlo1
wlo1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.97.74  netmask 255.255.255.0  broadcast 192.168.97.255
        inet6 2409:4042:4e93:fad6:2057:1297:2e01:71a0  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::e310:5ea0:189a:78aa  prefixlen 64  scopeid 0x20<link>
        inet6 2409:4042:4e93:fad6:38fb:9686:4d8c:ec20  prefixlen 64  scopeid 0x0<global>
        ether 54:14:f3:9c:b4:9f  txqueuelen 1000  (Ethernet)
        RX packets 2124689  bytes 724291455 (724.2 MB)
        RX errors 0  dropped 6167  overruns 0  frame 0
        TX packets 477374  bytes 175835585 (175.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
┌──(sarwar㉿kali)-[~/Downloads/code/secureSurvillianceBlockChain]
└─$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.97.228  netmask 255.255.255.0  broadcast 192.168.97.255
        inet6 fe80::a00:27ff:fe18:cecf  prefixlen 64  scopeid 0x20<link>
        inet6 2409:4042:4e93:fad6:a00:27ff:fe18:cecf  prefixlen 64  scopeid 0x0<global>
        inet6 2409:4042:4e93:fad6:cedb:6e91:b13d:cead  prefixlen 64  scopeid 0x0<global>
        ether 08:00:27:18:ce:cf  txqueuelen 1000  (Ethernet)
        RX packets 41  bytes 12056 (11.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 57  bytes 7028 (6.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

B) Ping both machines to ensure they are connected.
C) Run below to sync each other.
curl -X POST
http://<first node ip>:8000/register_with
-H 'Content-Type: application/json'
-d '{"node_address": "http://<second node ip>:8000"}'

curl -X POST
http://<second node ip>:8000/register_with
-H 'Content-Type: application/json'
-d '{"node_address": "http://<first node ip>:8000"}'

```
sarwar@sarwar-hp:~/Downloads/code/secureSurvillianceBlockChainModule$ curl -X POST \
       http://192.168.97.74:8000/register_with \
       -H 'Content-Type: application/json' \
       -d '{"node_address": "http://192.168.97.228:8000"}'
sarwar@sarwar-hp:~/Downloads/code/secureSurvillianceBlockChainModule$ curl -X POST \
       http://192.168.97.228:8000/register_with \
       -H 'Content-Type: application/json' \
       -d '{"node_address": "http://192.168.97.74:8000"}'
Registration successfulsarwar@sarwar-hp:~/Downloads/code/secureSurvillianceBlockChainModule$
```

7. Run Monitoring module:
   A) Goto secureSurvillianceMonitoringModule.
   B) Open application appConfig.properties file and add below line.
      MINING_NODE_ADDRESS= http://<Node IP>:8000
   Eg.
   add below first node:
      MINING_NODE_ADDRESS= http://192.168.97.74:8000

add below in second node:

MINING_NODE_ADDRESS= http://192.168.97.228:8000

C) Run MainMonitoring.py in both machines.

```
sarwar@sarwar-hp:~/Downloads/code/secureSurvillianceMonitoringModule$ python MainMonitoring.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

```
┌──(sarwar㉿kali)-[~/Downloads/code/secureSurvillianceMonitoringModule]
└─$ python MainMonitoring.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
```

D) Don't close it.

E) Open http://192.168.97.74:5000/ & http://192.168.97.228:5000/ in browser for respective machines.

8. Run Face Recognition module:

A) Go secureSurvillianceFaceRecognitionModule and open appConfig.properties and add below lines.

MINING_NODE_ADDRESS= http://<Node IP>:8000
MIN_TIME_BETWEEN_TWO_TXNS=3
NODE_LOCATION=<Node/Camera location>

Eg. Add below in first node:

MINING_NODE_ADDRESS= http://192.168.97.74:8000

MIN_TIME_BETWEEN_TWO_TXNS=3
NODE_LOCATION=DIAT CSE Department (Ubuntu Machine), Pune

Add below in second node:
MINING_NODE_ADDRESS= http://192.168.97.228:8000
MIN_TIME_BETWEEN_TWO_TXNS=3
NODE_LOCATION=DIAT CSE Department (Kali Machine), Pune

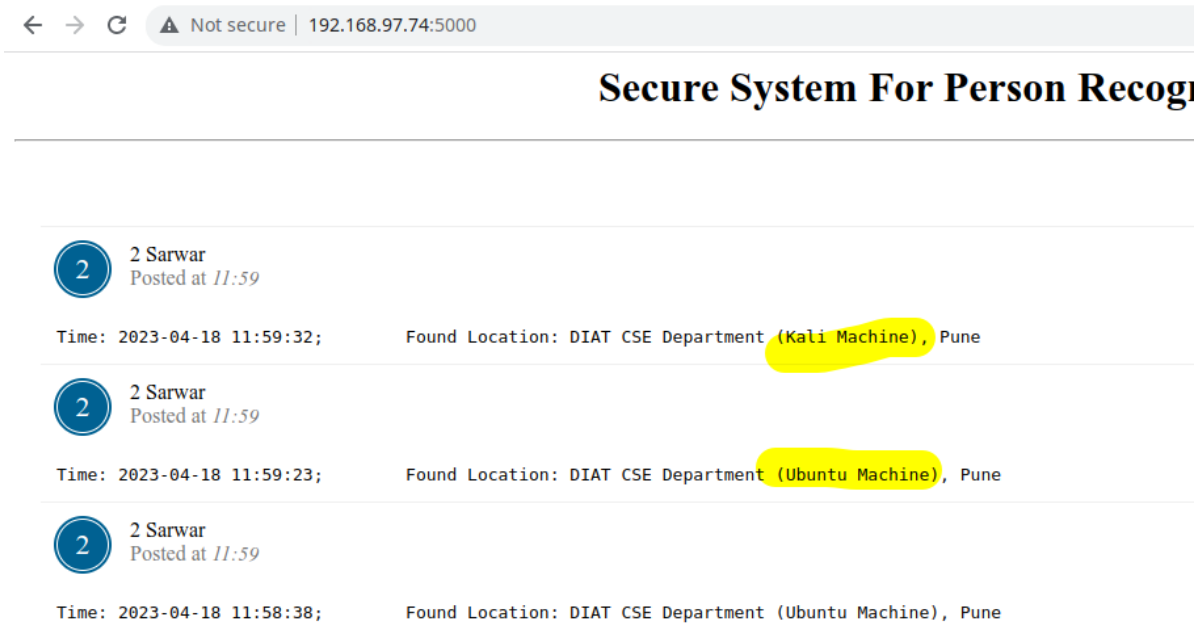
B) Run MainFaceRecognition.py in both machine.

```
sarwar@sarwar-hp:~/Downloads/code/secureSurvillianceFaceRecognitionModule$ python MainFaceRecognition.py
----------------------------------------
confidence: 47.234233518400025
label: 2
detected person is empty********************************************** {}
Time between two txns:  1e-06
{'personid': 2, 'name': 'Sarwar', 'entry_time': datetime.datetime(2023, 4, 18, 11, 21, 51, 564848), 'exit_
564862), 'camera_id': 0}
Person Name:  Sarwar
Entry Time:  2023-04-18 11:21:51.564872
Exit Time:  2023-04-18 11:21:51.564879
Camera location/id:  0
----------------------------------------
confidence: 48.025406473835716
label: 2
detected person is empty********************************************** {'personid': 2, 'name': 'Sarwar', '
, 51, 564848), 'exit_time': datetime.datetime(2023, 4, 18, 11, 21, 51, 564862), 'camera_id': 0}
Time between two txns:  0.130686
{'personid': 2, 'name': 'Sarwar', 'entry_time': datetime.datetime(2023, 4, 18, 11, 21, 51, 564848), 'exit_
```

C) It will open window to capture the video. Move your face for few seconds to make multiple entries in blockchain.

D) See the same in another machine.
E) The first machine should make an entry with location “Ubuntu Machine”, and the second machine should make an entry with location “Kali Machine” according to the appConfig.properties file configuration.

9. See the recognized person in Monitoring window:
A) Go to earlier opened browser and see the updated data.



B) Go to earlier opened browser of second node and see the data. Both browser should be giving the same output which shows both are synced.