PESU Center for Internet of Things

Connected Living Spaces

4th Review Presentation

Project ID: PW20CBR01M

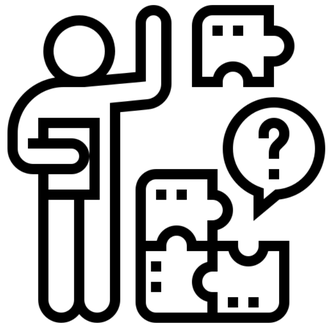
Project Guide:
Prof. Prasad Honnavalli

Co-Guide:
Prof. Charanraj B R

Project Team:
Chirag N Vijay - 01FB16ECS099
D G Sudheer - 01FB16ECS101
Dhanush Ravi - 01FB16ECS112

Project Abstract

1. Establish an ecosystem where various devices in a house ***communicate*** with each other.
2. Enable existing infrastructure to be ***communication ready*** with upcoming technologies.
3. Build safe and reliable homes that are ***self sustained*.**
4. Understand the current underlying infrastructure in Internet of Things.
5. Set up processes that ***increase the efficiency*** of the day to day tasks.



Literature Survey

1. Sensor based home automation and security system
   M. H. Assaf, R. Mootoo, S. R. Das, E. M. Petriu, V. Groza and S. Biswas, 2012 IEEE

2. Microcontroller based Home Security System with Remote Monitoring
   Nikhil Agarwal, 2012 ICEDSP

3. Internet of things based smart environmental monitoring using the Raspberry-Pi computer
   M. Ibrahim, A. Elgamri, S. Babiker and A. Mohamed, 2015 ICDIPC

4. Messaging Queue Telemetry Transport IOT based Messaging Protocol
   Suvam Mohanty, Sagar Sharma , Vaibhav Vishal, 2016 IRJET

5. Integrating Internet of Things with Web Services and Cloud Computing
   Moataz Soliman, Tobi Abiodun, 2013 IEEE

6. Room Temperature Control and Fire Alarm/Suppression IoT Service Using MQTT
   Do-Hun Kang, Min-Sung Park, 2017 PlatCon

7. IoT real time data acquisition using MQTT protocol
   R A Atmoko & R Riantini , Vaibhav Vishal, 2016 ICoPLA

Competitor Survey

Feature/Vendor	Vendor 1 – ATS	Vendor 2 – Solus	Vendor 3 – BNB	Our Solution
Centralised Control	No	Yes	Yes	Yes
Setup Programming	No	Will be done by the vendor at a one time cost (mandatory, ~25,000/-)		Tailor Made
Wireless Access	No	No, all devices connected via cat cable		Yes
Use Existing Network	A new wired network needs to be set up by the vendor			Yes, PESU Wifi
Lights and Fans Control	No, only performs access control			Yes
RFID Access	Yes			
Cost per Room	12,250/-	17,000/-	45,000/-	2249.5/-
Cost per 100 Rooms	12,25,000/-	17,00,000/-	45,00,000/-	2,24,950/-

Salient Features

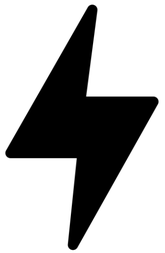
- Integration of **Lights & Fans**
- **In-house developed codebase**
- **Built using current and trending technologies**
- **Face Recognition** validation
- Can be accessed only when connected to PESU Network

User Classification

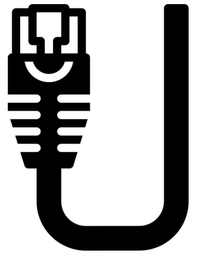
Level 1	Chair-Person, Head of Centre & Administrative Officers
Level 2	Professors, Laboratory Instructors & Teaching Assistants
Level 3	Student CRs, Attenders & Housekeeping

Dependencies & Risks

Primary Dependencies



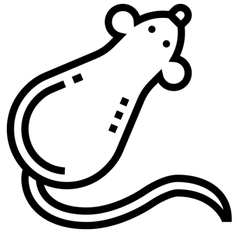
Electricity



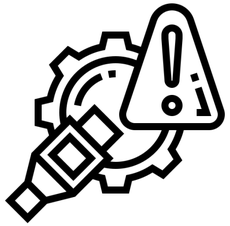
Network

- Server is up 24/7
- Firewall should not block the ports 1880, 1883 & Alexa Ports
- For Database, time is received from the NTP Server

Risks



Pests biting wires

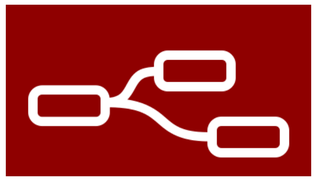


Hardware Failure

- MQTT Protocol might fail due to network congestion

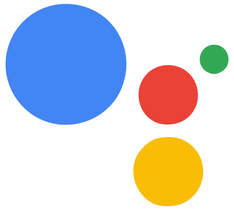
Technologies Used

Python



Node-RED

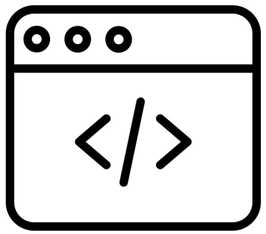
Arduino IDE



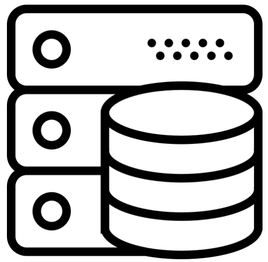
Google Assistant

Spotify

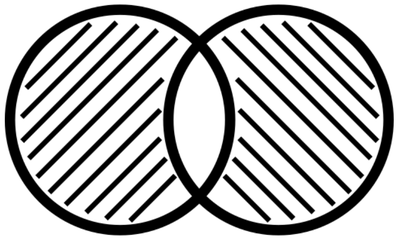
Alexa



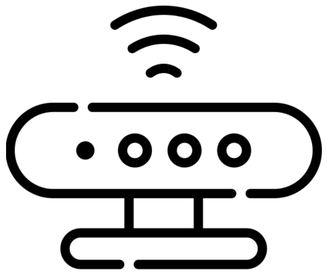
User Interface Development



Database



Control Logic

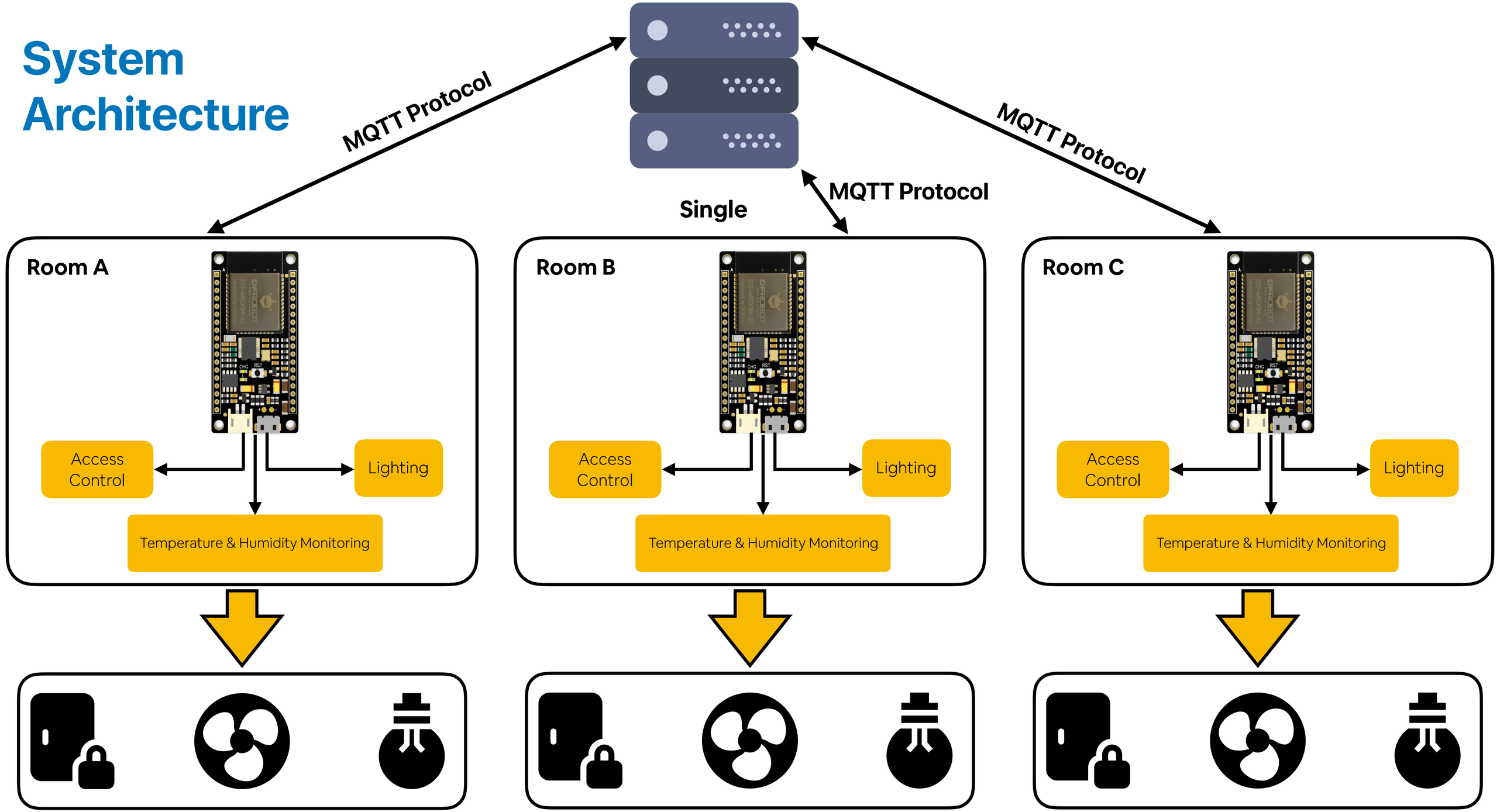


Understanding Sensor Data

Image Processing

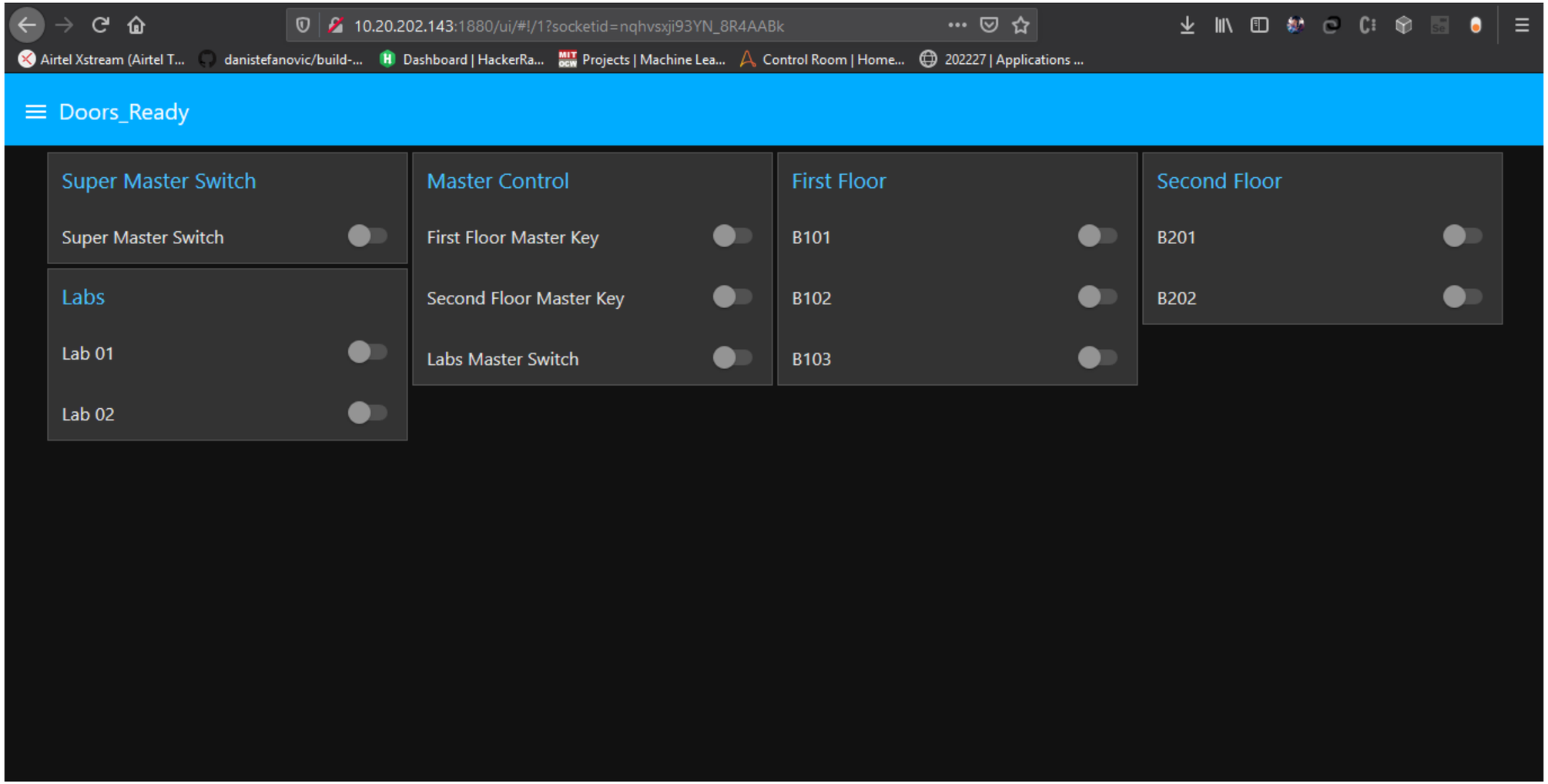
Data Security

System Architecture



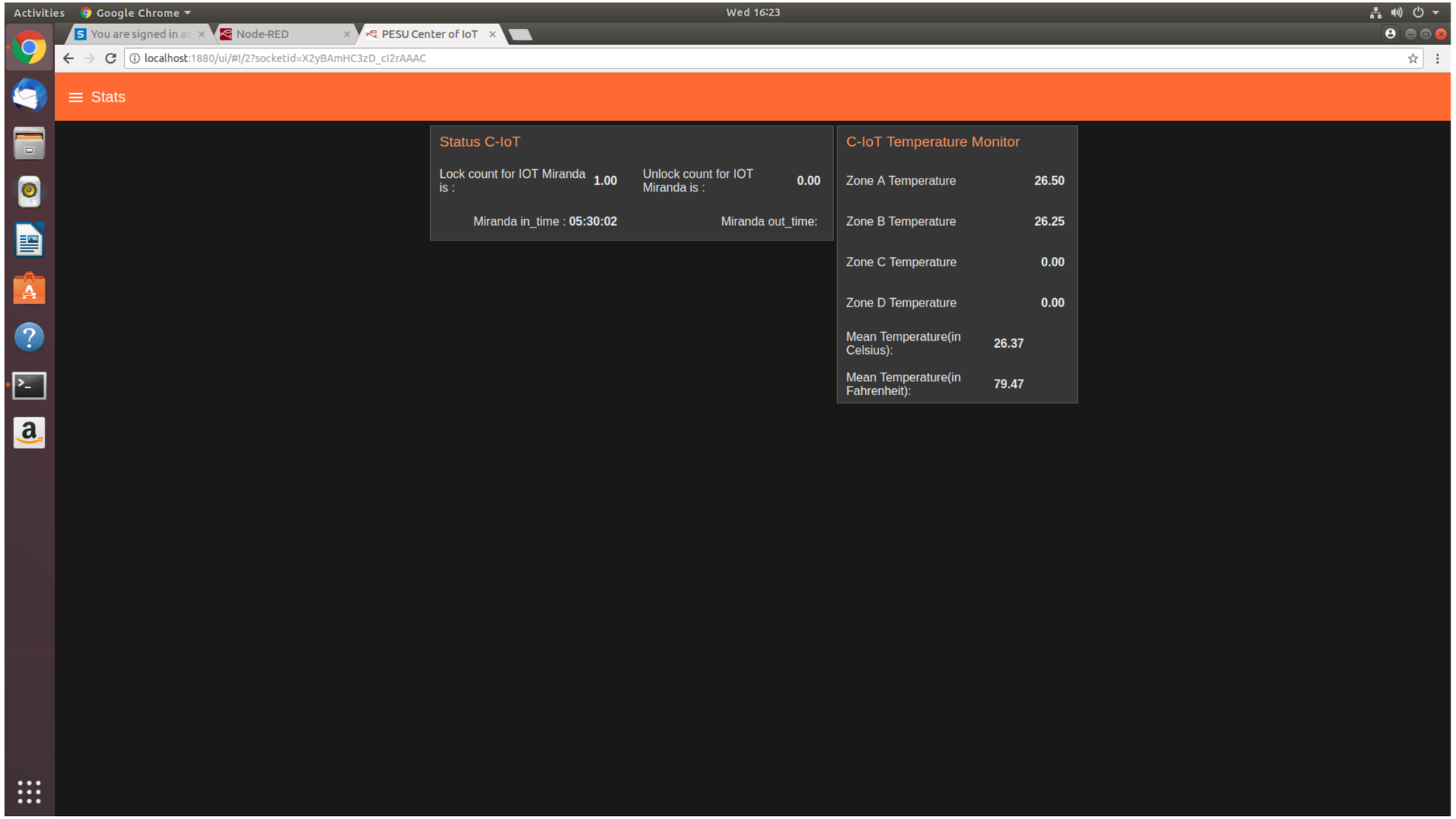
User Interface

Access Control

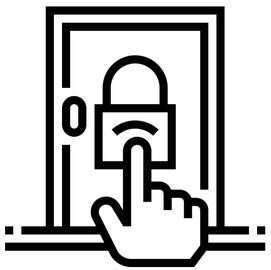


User Interface

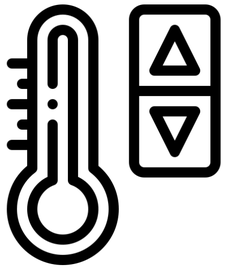
Temperature Control



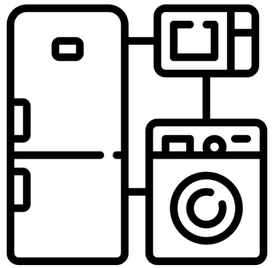
Project Modules



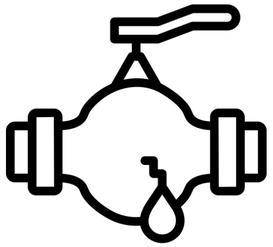
Access Control



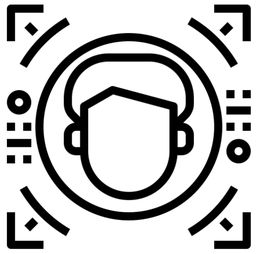
Temperature & Humidity Management



Lighting & Appliance Management

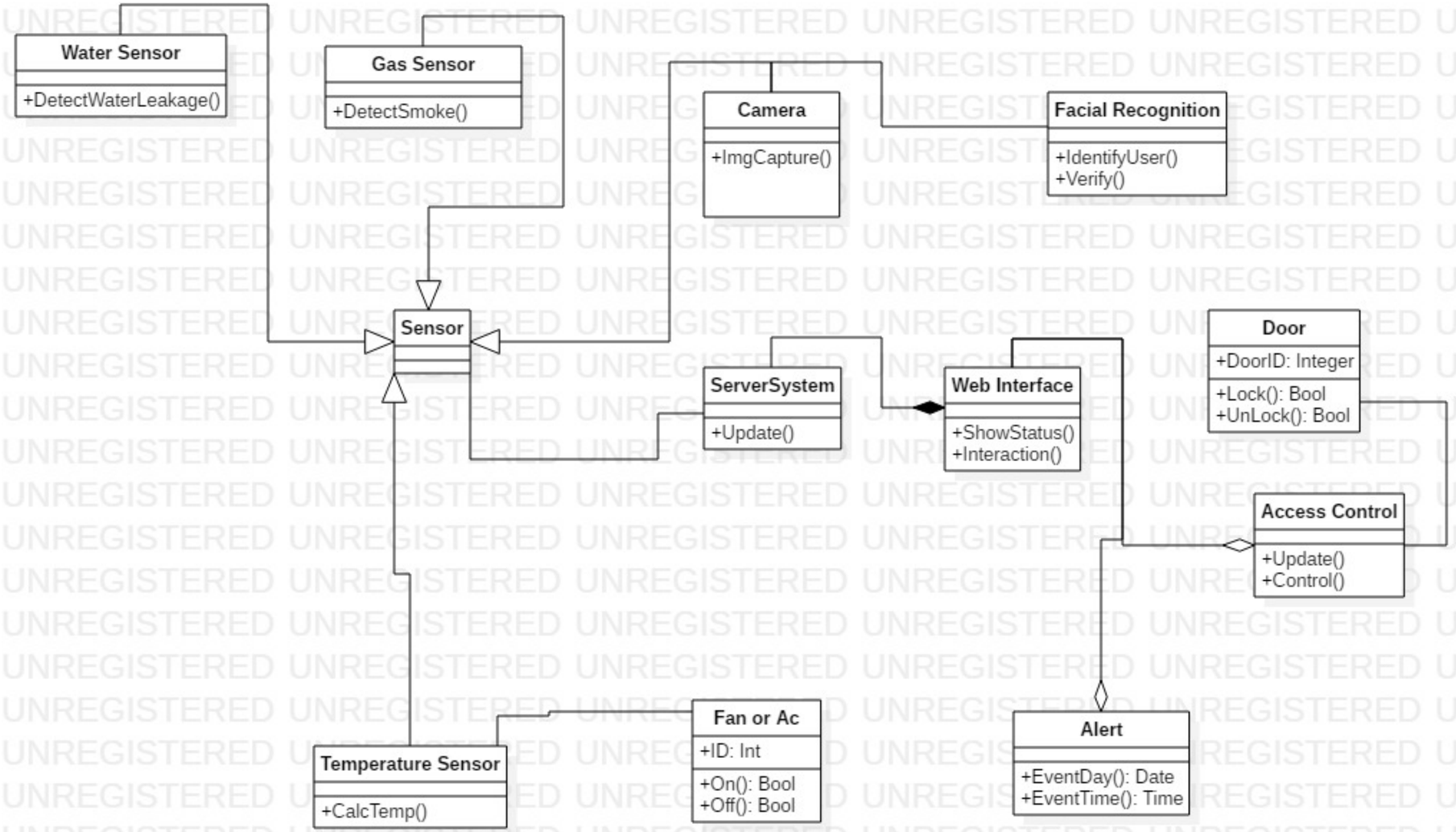


Leakage Detection & Alerts



Face & Mood Detection

Class Diagram

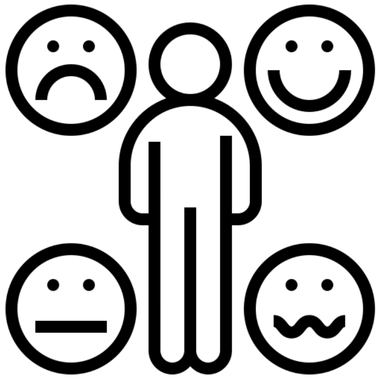


Use Case Diagram

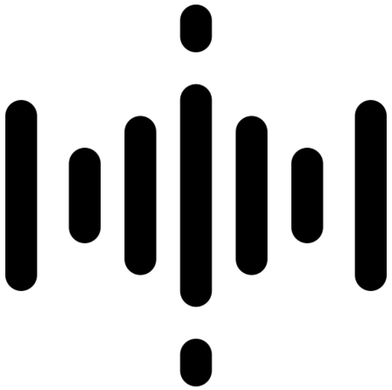
Progress Timeline

Module Number	Module Name	Submodules	Prototype Status	Testing Status	Final Implementation	Month of Completion
1	Access Control	Time Based	Halfway	Halfway	Halfway	January - Mid Feb
		UI Based	Yes	Yes	Yes	
		Data Logging	Yes	Yes	Yes	
		Multiple Locks	Yes	Yes	Yes	
		RFID	Yes	To be done	To be done	
2	Facial Recognition	Parallel - Independent of Access Control	Yes	Yes	Yes	December - Current
		Series - Control Access Control	To be done	To be done	To be done	
3	HVAC Module	Temperature Monitoring and Display	Yes	Yes	Yes	Mid Feb - Current
		Controlling AC Devices from the Data Collected	Yes	Yes	Yes	
		Detecting Faulty Temperature Sensors	Yes	Halfway	Halfway	
		Controlling the speed of the AC devices based on the data collected	Yes	Halfway	Halfway	
4	Lighting and other appliances	Contolling the Lights using the UI	Yes	Yes	Yes	March - Current
		Collecting the data from the light sensors	Halfway	Halfway	Halfway	
		Using the data from the light sensors to control the lights	Halfway	Halfway	Halfway	
5	Spotify mood detection	Detecting the mood	Yes	Yes	Yes	December - Current
		Creating the playlist from the observed mood	Yes	Yes	Yes	
6	Water Leakage	Measuring water level	Yes	Halfway	Halfway	Mid March - Current
		Data Logging	Yes	Yes	Yes	
		Controlling the pump	Yes	Yes	Yes	
		Controlling the electronic valves	Yes	To be done	To be done	
7	Gas Leakage	Checking for Gas Leakage	Yes	To be done	To be done	April
		Finding the Place of Leakage	To be done	To be done	To be done	
		Linking with HVAC and Access Control	To be done	To be done	To be done	
		Controlling the electronic valves to prevent further leak	To be done	To be done	To be done	
		Data Logging	To be done	To be done	To be done	
8	Surveillance	Logging the footage from CCTV	To be done	To be done	To be done	April
9	Assistant Control	Access Control	Yes	Yes	Yes	January
		HVAC	Yes	Yes	Yes	
		Appliances	Yes	Yes	Yes	

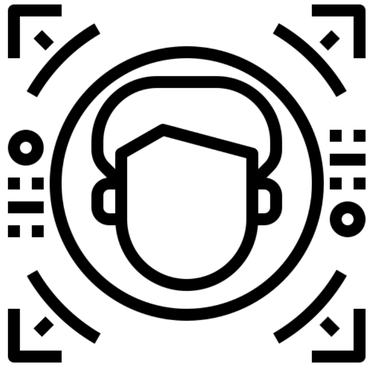
New Modules



Mood Detection



Voice Control



Facial Recognition

Progress Check

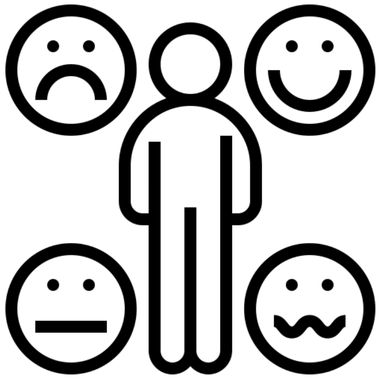
- Access Control Module deployed in G04 Lecture Hall
- Handed over the control to CSE department office

Control Box

Magnet Module

Magnet Module

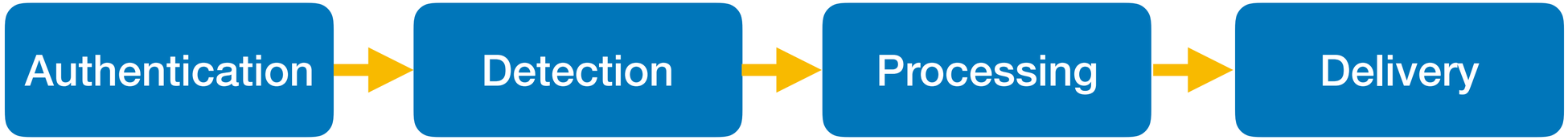
Mood Detection



Abstract

- The user listens to their choice of music on “Spotify” Music App
- The system built monitors the songs being listened to and analyses the mood of the user
- Based on the mood, further recommendations are made

Process



Step 1: Authentication

- In this the user has to provide their client identification number and client secret from his Spotify developer dashboard.
- They also has to give his username for the same.
- We are creating a token from the username, scope, client_id and client_secret provided. The “authenticate_spotify” function does the authentication and it creates a Spotify instance for future tasks and development

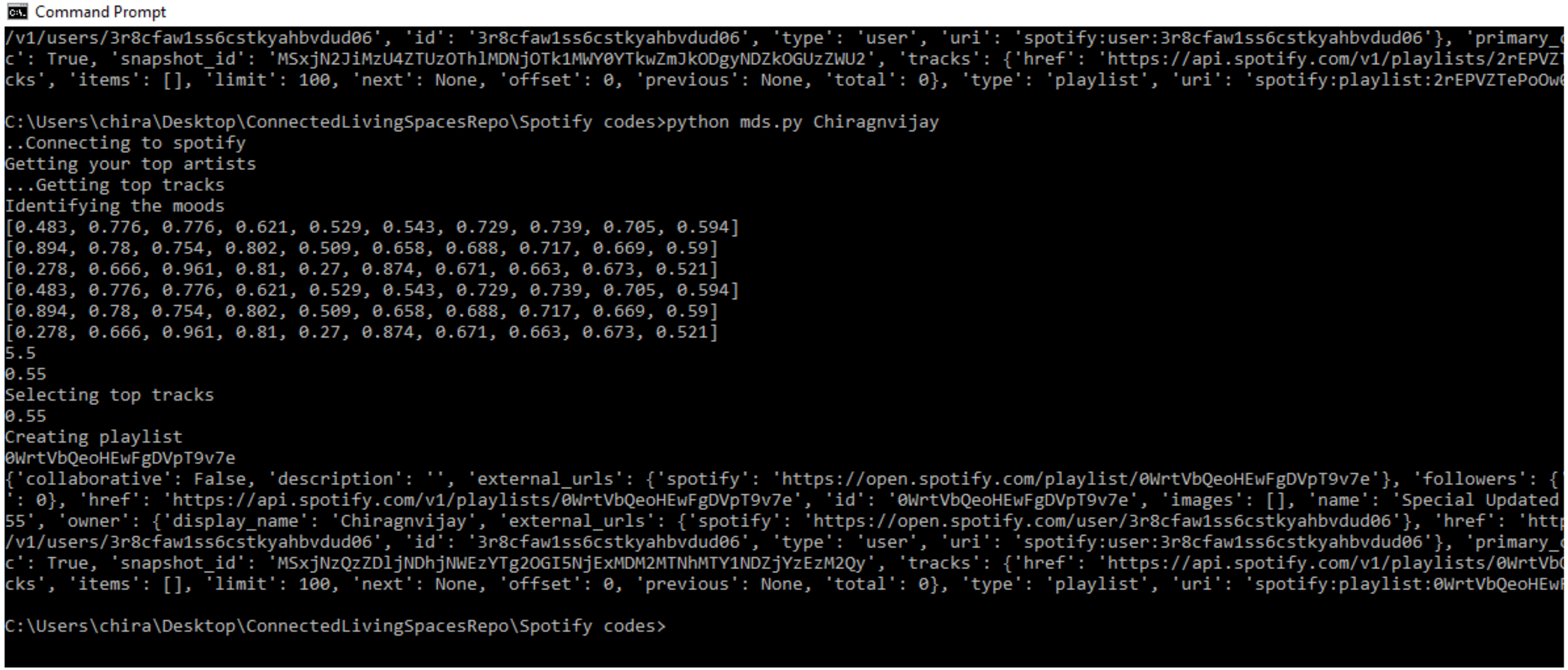
Code :

```
token=util.prompt_for_user_token(username,scope
,client_id=client_id,client_secret=client_secre
t,redirect_uri=redirect_uri)
if token:
  def authenticate_spotify():
        print("..Connecting to spotify")
        sp=spotipy.Spotify(auth=token)
        return sp
```

Mood Detection

Step 2: Detection

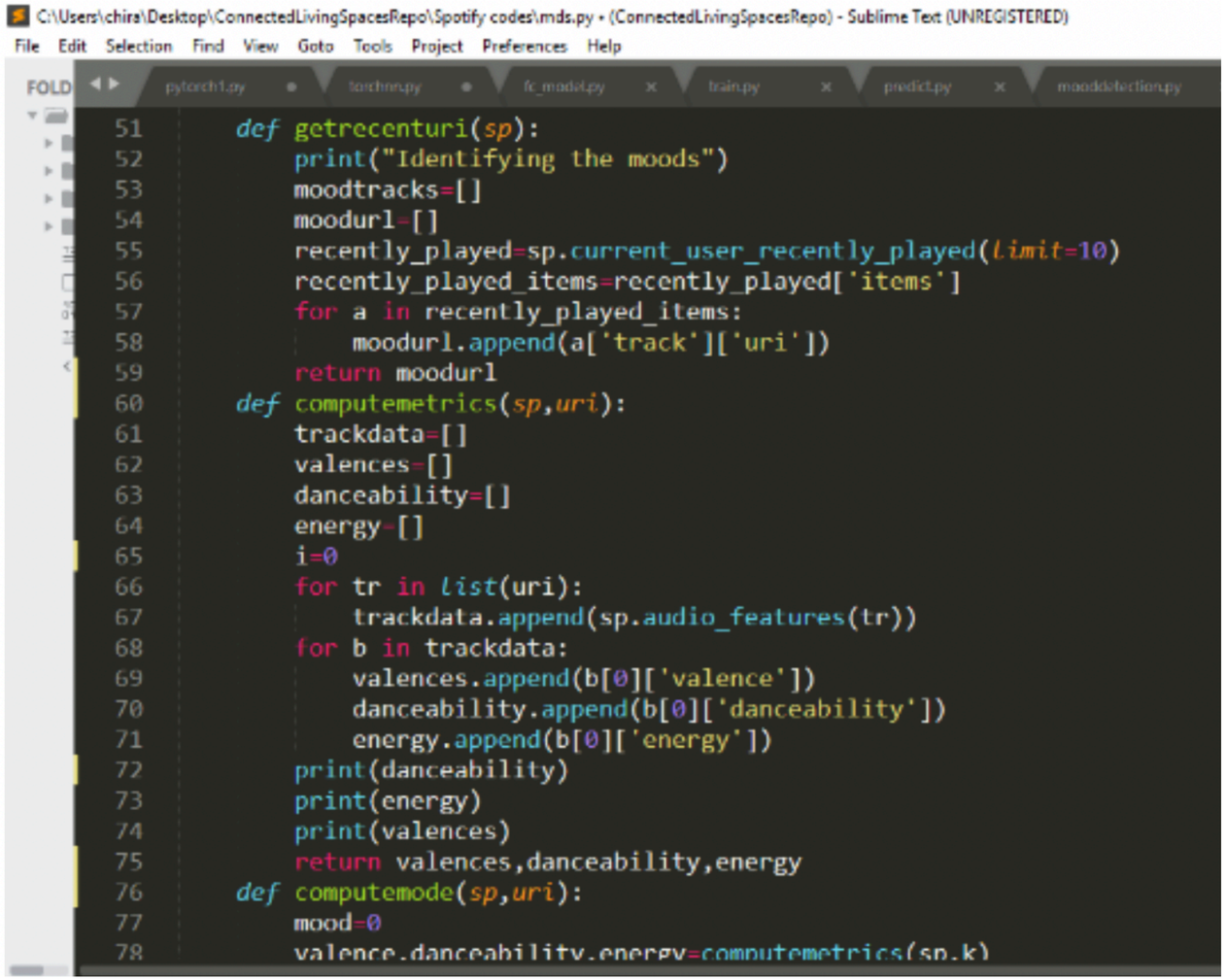
- In this we gather data regarding the last 10 songs the user has listened to.We then gather the audio features of every song and we predict the mood value based on the values of the audio features.



Mood Detection

Step 2: Detection

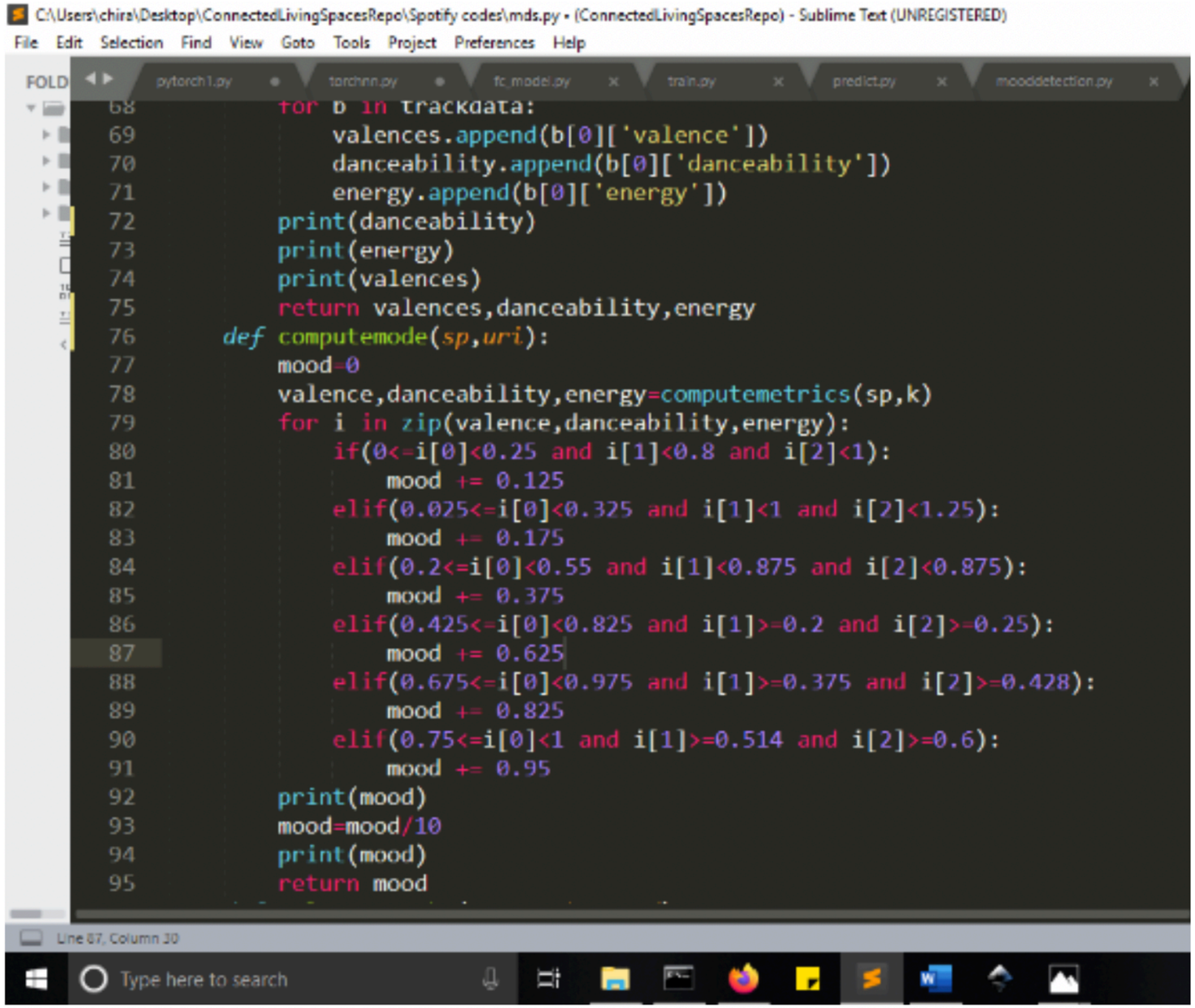
- In getrecenturi function we are appending the URI's of all last 10 listened songs to a list.
- In computemetrics function we are calculating valence, danceability and energy values of the songs in the URI list and appending it to 3 different lists



Step 2: Detection

Mood Detection

- In computemode function based on the output from the computemetrics function we calculate the mood value and then divide it by 10 to get the accurate scaled mood value



Step 3: Processing

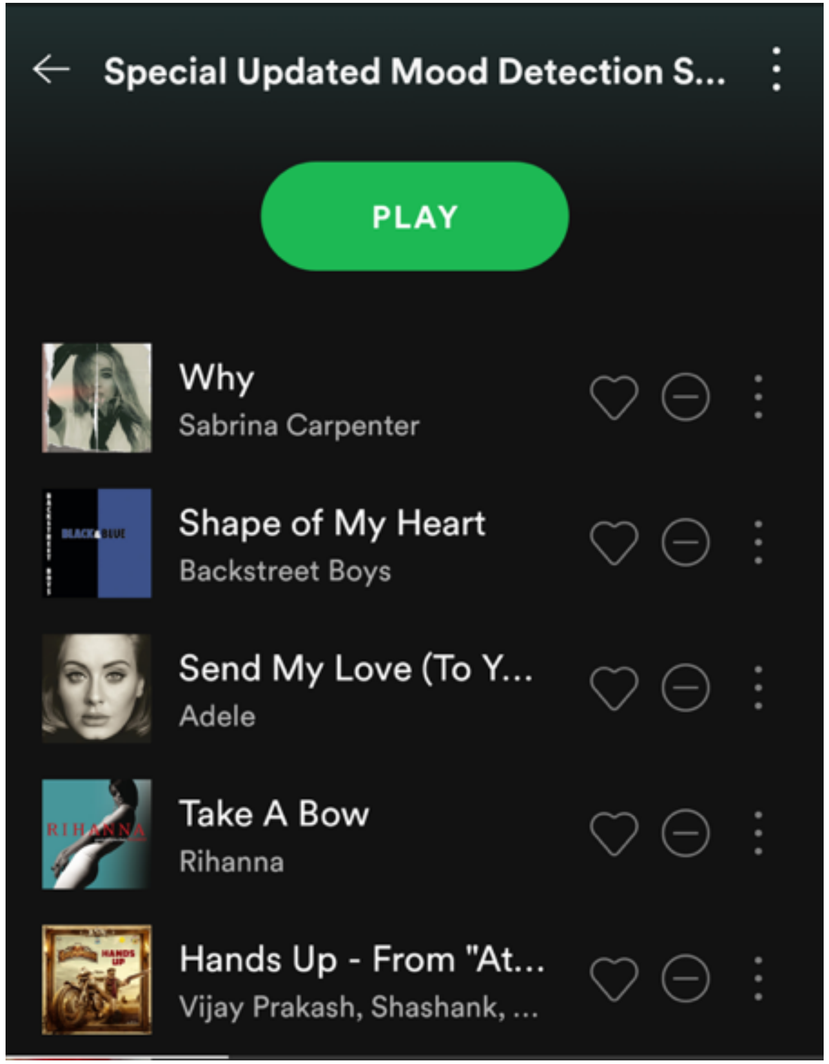
- Aggregate_top_artists function prepares a list of the URI's of the artists the user frequently listens too or follows.aggregate_top_tracks consists of a list of the songs belonging to the artists from aggregate_top_artists.
- In this the output from the compute mode function is passed as a parameter to the select_tracks function.
- In this a list of tracks by the artists the users follows is prepared and then the audio features of those songs are compared with the computed mood values and only the ones which match are appended to a list called selected_tracks

```
def aggregate_top_artists(sp):
    print("Getting your top artists")
    name=[]
    uri=[]
    ranges=['short_term','medium_term','long_term']
    for r in ranges:
        artists_all_data=sp.current_user_top_artists(limit=50,time_range=r)
        artists_data=artists_all_data['items']
        for artist_data in artists_data:
            if artist_data["name"] not in name :
                name.append(artist_data['name'])
                uri.append(artist_data['uri'])
    followed_artists_all_data=sp.current_user_followed_artists(limit=50)
    followed_artists_data=followed_artists_all_data['artists']
    for artist_data in followed_artists_data["items"]:
        if artist_data["name"] not in name :
                name.append(artist_data['name'])
                uri.append(artist_data['uri'])
    return uri
def aggregate_top_tracks(sp,uri):
    print("...Getting top tracks")
    tracks=[]
    for artist in uri:
        top_tracks_all_data=sp.artist_top_tracks(artist)
        top_tracks_data=top_tracks_all_data['tracks']
        for track_data in top_tracks_data:
            tracks.append(track_data['uri'])
    return tracks
```

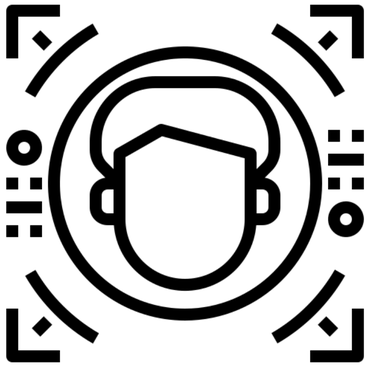
Step 4: Delivery

- We then create a playlist and then add 30 songs from the selected_tracks into the playlist .The user can then access it from the library option from their spotify accounts.

```
def select_tracks(sp,tracks,mood):
    print("Selecting top tracks")
    print(mood)
    selected_tracks=[]
    random.shuffle(tracks)
    for t in list(tracks):
        tracks_all_data=sp.audio_features(t)
        for track_data in tracks_all_data:
            try:
                if mood<0.10:
                    if(0<=track_data["valence"]<=(mood +0.15) and track_data["danceability"]<= (mood*8) and track_data["energy"] <= (mood*10)):
                        selected_tracks.append(track_data["uri"])
                elif 0.10 <= mood <= 0.25:
                    if((mood-0.075)<=track_data["valence"]<=(mood +0.075) and track_data["danceability"]<= (mood*4) and track_data["energy"] <= (mood*5)):
                        selected_tracks.append(track_data["uri"])
                elif 0.25 <= mood <= 0.50:
                    if((mood - 0.05)<=track_data["valence"]<=(mood +0.05) and track_data["danceability"]<= (mood*1.75) and track_data["energy"] <= (mood*1.75)):
                        selected_tracks.append(track_data["uri"])
                elif 0.50 <= mood <= 0.75:
                    if((mood - 0.075)<=track_data["valence"]<=(mood +0.075) and track_data["danceability"]>= (mood/2.5)and track_data["energy"] >= (mood/2)):
                        selected_tracks.append(track_data["uri"])
                elif 0.75 <= mood <= 0.90:
                    if((mood - 0.075)<=track_data["valence"]<=(mood +0.075) and track_data["danceability"]>= (mood/2)and track_data["energy"] >= (mood/1.75)):
                        selected_tracks.append(track_data["uri"])
                elif mood > 0.90:
                    if((mood - 0.15)<=track_data["valence"]<= 1 and track_data["danceability"]>= (mood/1.75)and track_data["energy"] >= (mood/1.5)):
                        selected_tracks.append(track_data["uri"])
            except TypeError as te:
                continue
    return selected_tracks
```

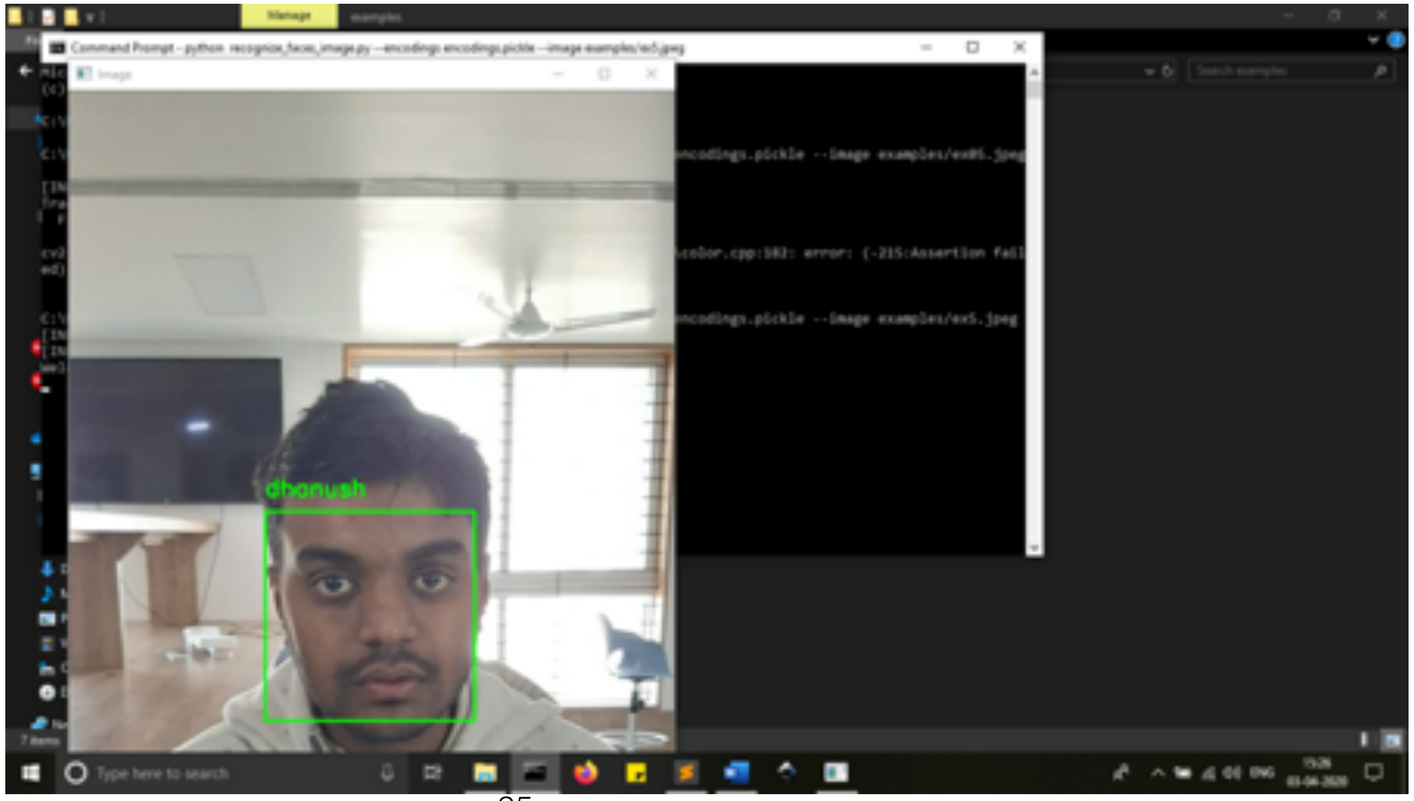


Face Recognition



Abstract

- Set-up a system where the access control can be controlled using a user's face
- This can be used as an alternative to the use of password to unlock doors



Contents

- The repository consists of four folders and three python scripts

•Facial_recog

……> datasets

……> example

……> outputs

Encode_faces.py

Recognize_face_video.py

Recognize_face_image.py

Encodings.pickle

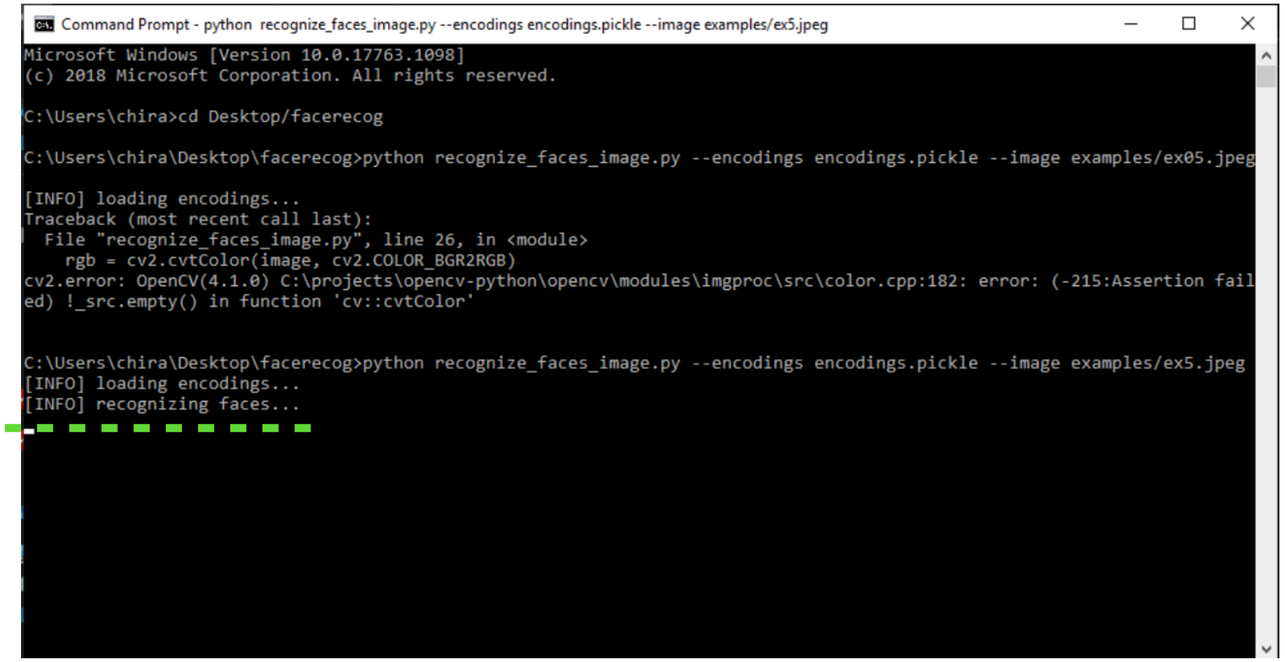
Datasets

•Suppose there are four authorised users then datasets contain four subfolders with their names as the folder names and each folder has the respective user's pictures

•All the images are in standard quality for faster training purposes

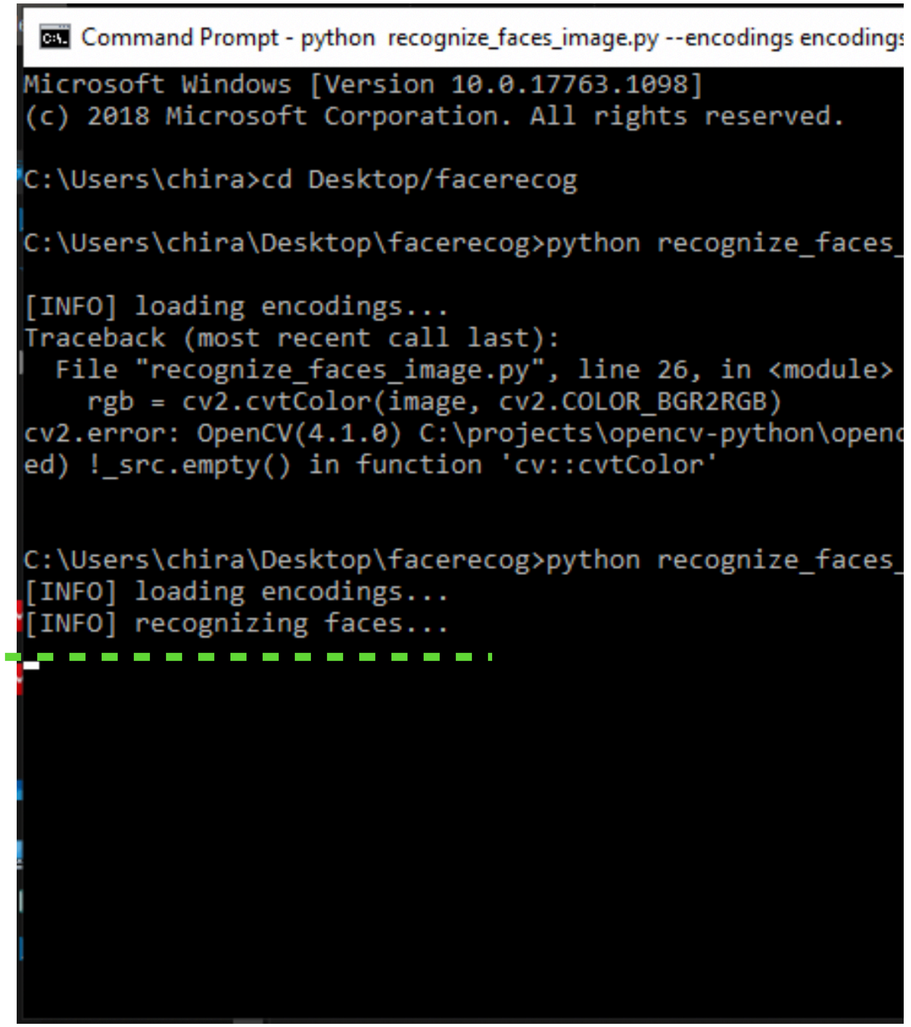
Encodings

- We iterate through every image ,we extract the name from the path, convert the opencv image to dlib ordering and store the ordering in a variable ‘rgb’.
- Then we detect the x and y coordinates of the defining points of the face in the images and store it in ‘boxes’.
- Then we compute the facial encodings using ‘rgb’ and ‘boxes’ and store in encodings
- Then we loop through the encodings and add the encoding and name to known names and encoding
- Then we create a dictionary containing name and encodings and add the known names and encoding and write it to the encodings.pickle file .



Encodings

- We iterate through every image ,we extract the name from the path,convert the opencv image to dlib ordering and store the ordering in a variable ‘rgb’.
- Then we detect the x and y coordinates of the defining points of the face in the images and store it in ‘boxes’.
- Then we compute the facial encodings using ‘rgb’ and ‘boxes’ and store in encodings
- Then we loop through the encodings and add the encoding and name to known names and encoding
- Then we create a dictionary containing name and encodings and add the known names and encoding and write it to the encodings.pickle file .

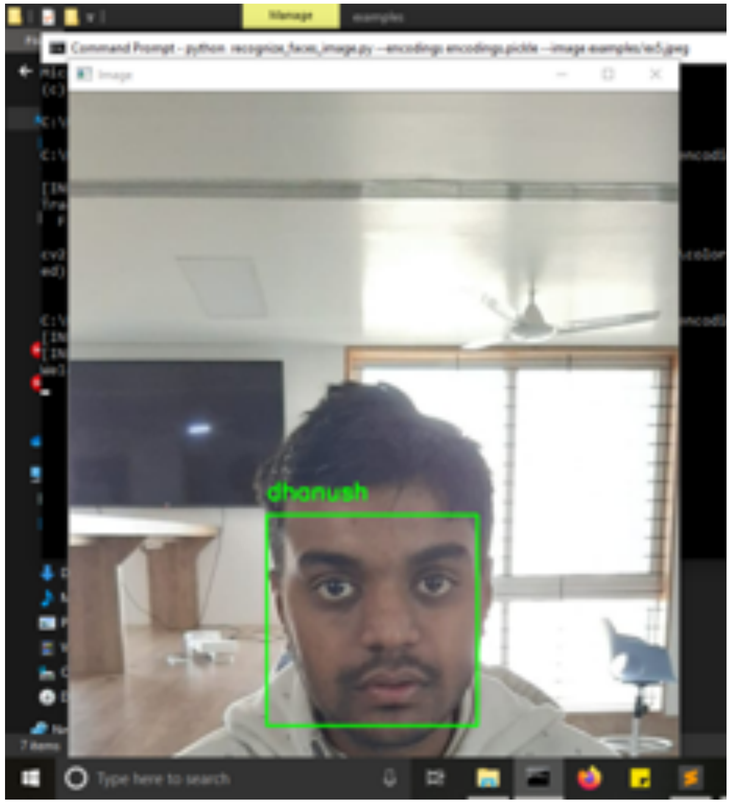


Detection Procedure

1. We can either detect it in real time using a video stream or by an image file.
2. Since the implementation is to be in real-time we stick to the video stream method.
3. We start the video stream first, then convert the opencv image to dlib ordering and store the ordering in a variable ‘rgb’, detect the x and y coordinates of the defining points of the face in the images and store it in ‘boxes’, we compute the facial encodings using ‘rgb’ and ‘boxes’ and store in encodings.
4. We then loop over the encoding ,compare it with the encodings from the pickle file.
5. If a match is found ,we find the indexes of all the matched faces, calculate the total number of times each face was matched and we select the name with the maximum number of votes and append it to names.
6. Then we loop over the recognised faces ,we rescale the face co-ordinates ,construct a rectangle around it and label it with the name of the user from the list ’names’.
7. Then after the user presses q we exit the process
8. Suppose there is no match then the name is displayed as “Unknown”

Face Recognition

End Result




```
Command Prompt - python  recognize_faces_image.py --encodings encodings.pickle --image
Microsoft Windows [Version 10.0.17763.1098]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\chira>cd Desktop/facerecog

C:\Users\chira\Desktop\facerecog>python recognize_faces_image.py -

[INFO] loading encodings...
Traceback (most recent call last):
  File "recognize_faces_image.py", line 26, in <module>
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
cv2.error: OpenCV(4.1.0) C:\projects\opencv-python\opencv\modules\
ed) !_src.empty() in function 'cv::cvtColor'


C:\Users\chira\Desktop\facerecog>python recognize_faces_image.py -
[INFO] loading encodings...
[INFO] recognizing faces...
Welcome to your living spacedhanush
```

How it works

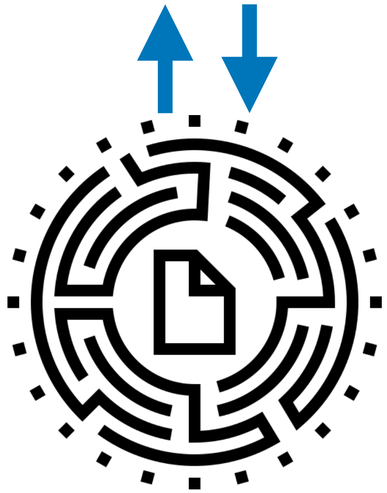
Voice Control

"Okay Google"

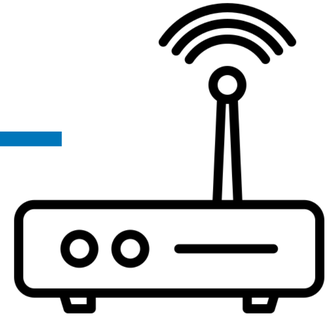
Google assistant Triggered

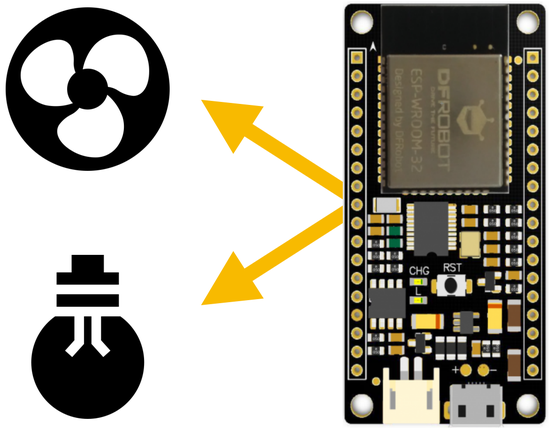
Google assistant sends the request to the IFTTT Server

Understands to request

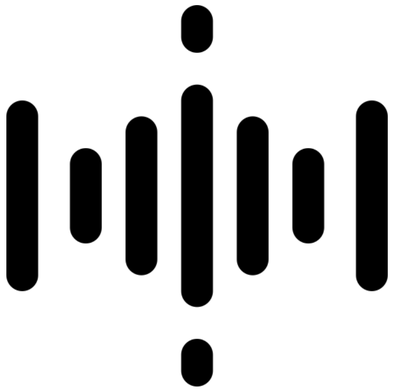


Verifies the Authentication Key sent on the Blynk cloud





Voice Control



Demo

PESU Center for Internet of Things

Thank You

Chirag N Vijay | D G Sudheer | Dhanush Ravi