**SCHOOL OF COMPUTING (SOC)**

|  |  |
| --- | --- |
| **Date of Submission:** | 17 Feb 2019 |

|  |  |
| --- | --- |
| **Prepared for:** | Dora Chua |

|  |  |
| --- | --- |
| **Class:** | DIT/FT/3A/32 |

|  |  |
| --- | --- |
| **Submitted by:** |  |

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| 1626430 | Samuel Chua |
| 1646322 | Soh Wee Kiat |

**IOT CA2**

**Step-by-step Tutorial**

**DIPLOMA IN BUSINESS INFORMATION TECHNOLOGY**

**DIPLOMA IN INFORMATION TECHNOLOGY**

**DIPLOMA IN INFOCOMM SECURITY MANAGEMENT**

**ST0324 Internet of Things (IOT)**

**2017/2018 Semester 1**

**Table of Contents**

[Section 1 Overview of project 4](#_Toc1315898)

[A. Where we have uploaded our tutorial 4](#_Toc1315899)

[B. What is the application about? 4](#_Toc1315900)

[C. How does the final RPI set-up looks like? 4](#_Toc1315901)

[D. How does the web or mobile application look like? 5](#_Toc1315902)

[Section 2 Hardware requirements 10](#_Toc1315903)

[Hardware checklist 10](#_Toc1315904)

[Section 3 Setup Hardware on Raspberry Pi 14](#_Toc1315905)

[A. Completed Fritzing Diagram 14](#_Toc1315906)

[B. Connect the Components! 15](#_Toc1315909)

[C. Add a 10k ohms resistor 17](#_Toc1315912)

[Section 4 Installing libraries On Raspberry Pi 19](#_Toc1315913)

[A. Enable SPI and prepare the MFRC522 libraries 19](#_Toc1315914)

[B. Install the necessary libraries (LCD) 21](#_Toc1315915)

[C. Install the necessary libraries (Bcrypt) 21](#_Toc1315916)

[D. Install the necessary libraries (MySQL-Python) 21](#_Toc1315917)

[E. Install the necessary libraries (Boto3) 22](#_Toc1315918)

[F. Install the necessary libraries (AWSIoTMQTTClient) 22](#_Toc1315919)

[Section 4 Installing libraries On Aws EC2 Server 23](#_Toc1315920)

[A. Install the necessary libraries (Flask) 23](#_Toc1315921)

[B. Install the necessary libraries (Bcrypt) 23](#_Toc1315922)

[C. Install the necessary libraries (MySQL-Python) 23](#_Toc1315923)

[Section 5 Create mySQL database 24](#_Toc1315924)

[Section 6 Code the program 26](#_Toc1315925)

[(Shared codes between AWS & IoT) 26](#_Toc1315926)

[A. Configs.py 26](#_Toc1315927)

[B. Database.py 27](#_Toc1315928)

[C. helpers.py 29](#_Toc1315929)

[D. Room.py 30](#_Toc1315930)

[E. EnviroInfo.py 33](#_Toc1315931)

[F. AccessLog.py 35](#_Toc1315932)

[G. User.py 37](#_Toc1315933)

[H. AccessRight.py 40](#_Toc1315934)

[I. MotionEvent.py 42](#_Toc1315935)

[J. AccessRequest.py 43](#_Toc1315936)

[Section 7 Code the program (IoT) 45](#_Toc1315937)

[A. AwsIot.py 45](#_Toc1315938)

[B. AwsS3.py 46](#_Toc1315939)

[C. LCDScroller.py 47](#_Toc1315940)

[D. MFRC522.py 49](#_Toc1315941)

[E. MotionDetection.py 57](#_Toc1315942)

[F. SmartSecurity.py 59](#_Toc1315943)

[G. Updater.py 62](#_Toc1315944)

[H. UpdaterMatrixCreator.py 64](#_Toc1315945)

[Section 8 Creating html files 67](#_Toc1315946)

[A. base.html 67](#_Toc1315947)

[B. 404.html 69](#_Toc1315948)

[C. external\_base.html 70](#_Toc1315949)

[D. login.html 71](#_Toc1315950)

[E. Internal\_base.html 72](#_Toc1315951)

[F. dashboard.html 76](#_Toc1315952)

[G. manage\_rooms.html 78](#_Toc1315953)

[H. room\_access\_rights.html 81](#_Toc1315954)

[Section 9 Creating js files 84](#_Toc1315955)

[A. dashboard.js 84](#_Toc1315956)

[B. login.js 91](#_Toc1315957)

[C. manage\_rooms.js 93](#_Toc1315958)

[D. room\_access\_rights.js 96](#_Toc1315959)

[E. Internal\_base.js 99](#_Toc1315960)

[Section 10 Creating server files 103](#_Toc1315961)

[A. SigV4Utils.py 103](#_Toc1315962)

[B. server.py 104](#_Toc1315963)

[C. telegramBot.py 112](#_Toc1315964)

[D. TeleRunner.py 116](#_Toc1315965)

[Section 11 Test the program 117](#_Toc1315966)

[A. Run the program 117](#_Toc1315967)

[B. Expected Results 117](#_Toc1315968)

# Section 1 Overview of project

* 1. Where we have uploaded our tutorial

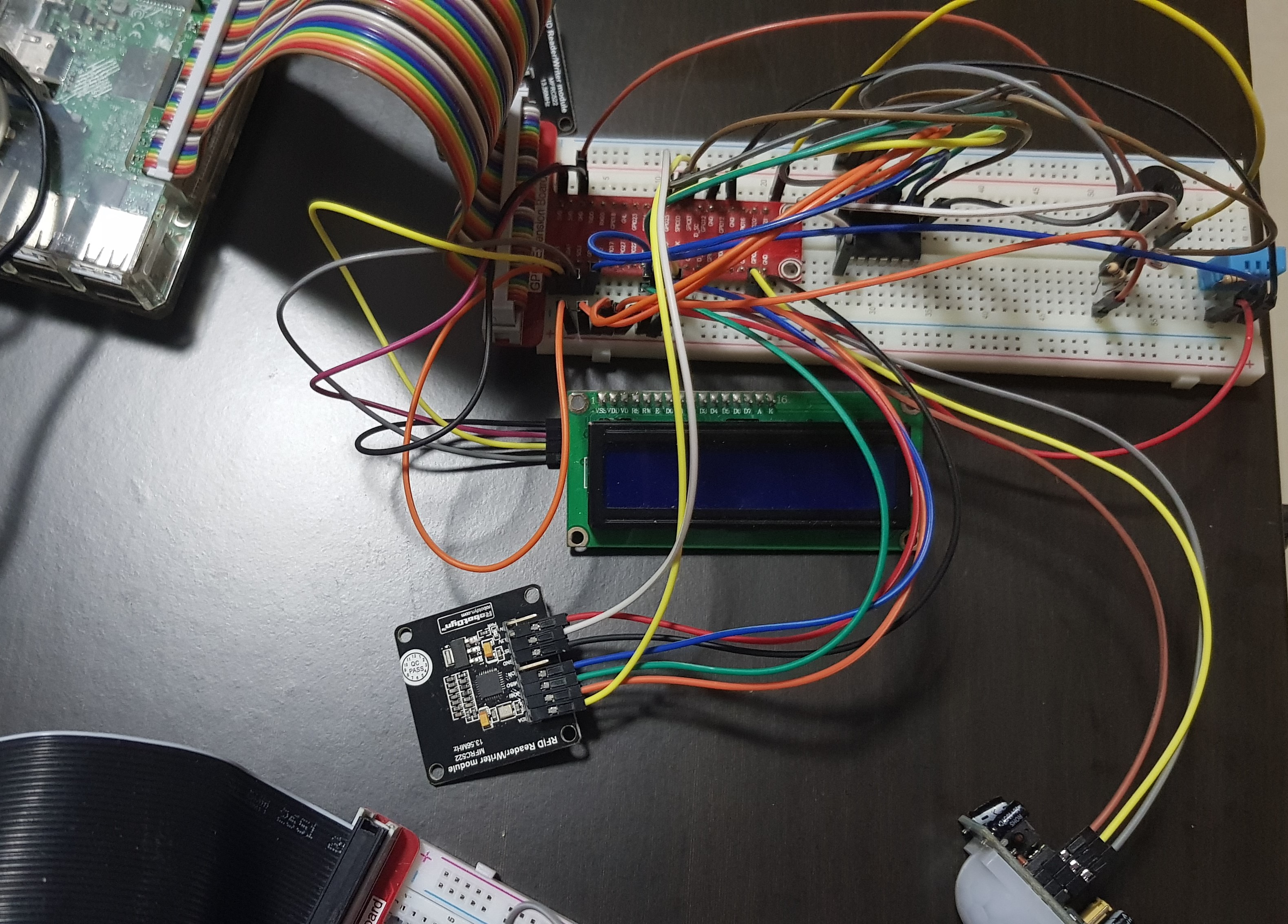
Paste the link here of your Youtube and tutorial document here

* 1. What is the application about?

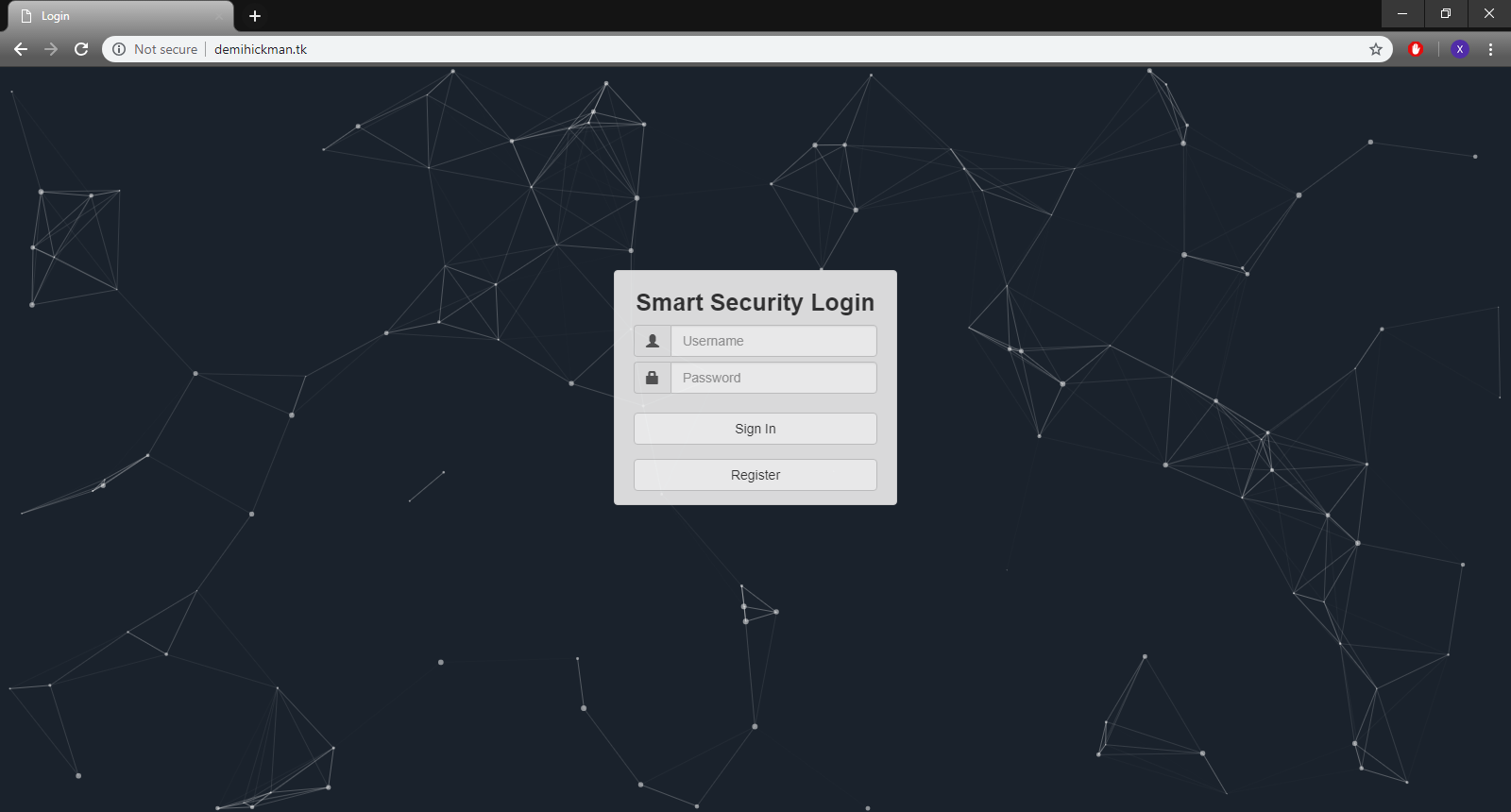
The application, Smart Security, allow the administrators to manage their room security, with motion detection and video recording, the administrator can add unlimited amount of rooms provided there is enough hardware. Each hardware has a RFID card reader, to allow access to room, a temperature sensor to monitor the room’s temperature, a LED screen to display room name and messages to user, a motion sensor & camera. Administrator can keep track & allow specific card to access certain rooms, the motion sensor will be disarmed if room is currently being accessed. Upon any irrgularity such as motion detected , the application will start a video recording and will be recorded down into a file and it will notify the administrator through telegram with the video.

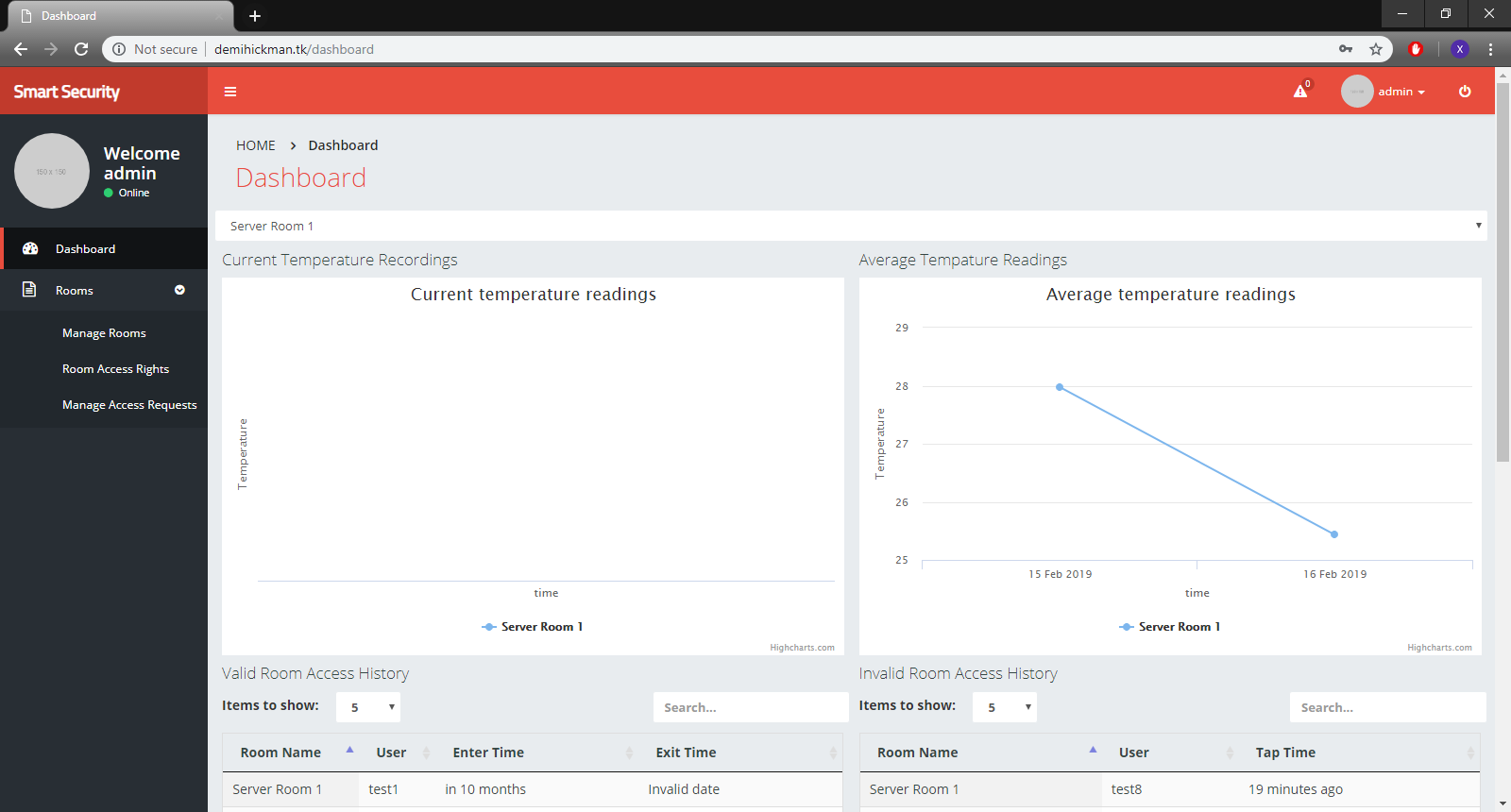
The web interface will allow the administrator view historial temperature,access history & motion detected recordings.

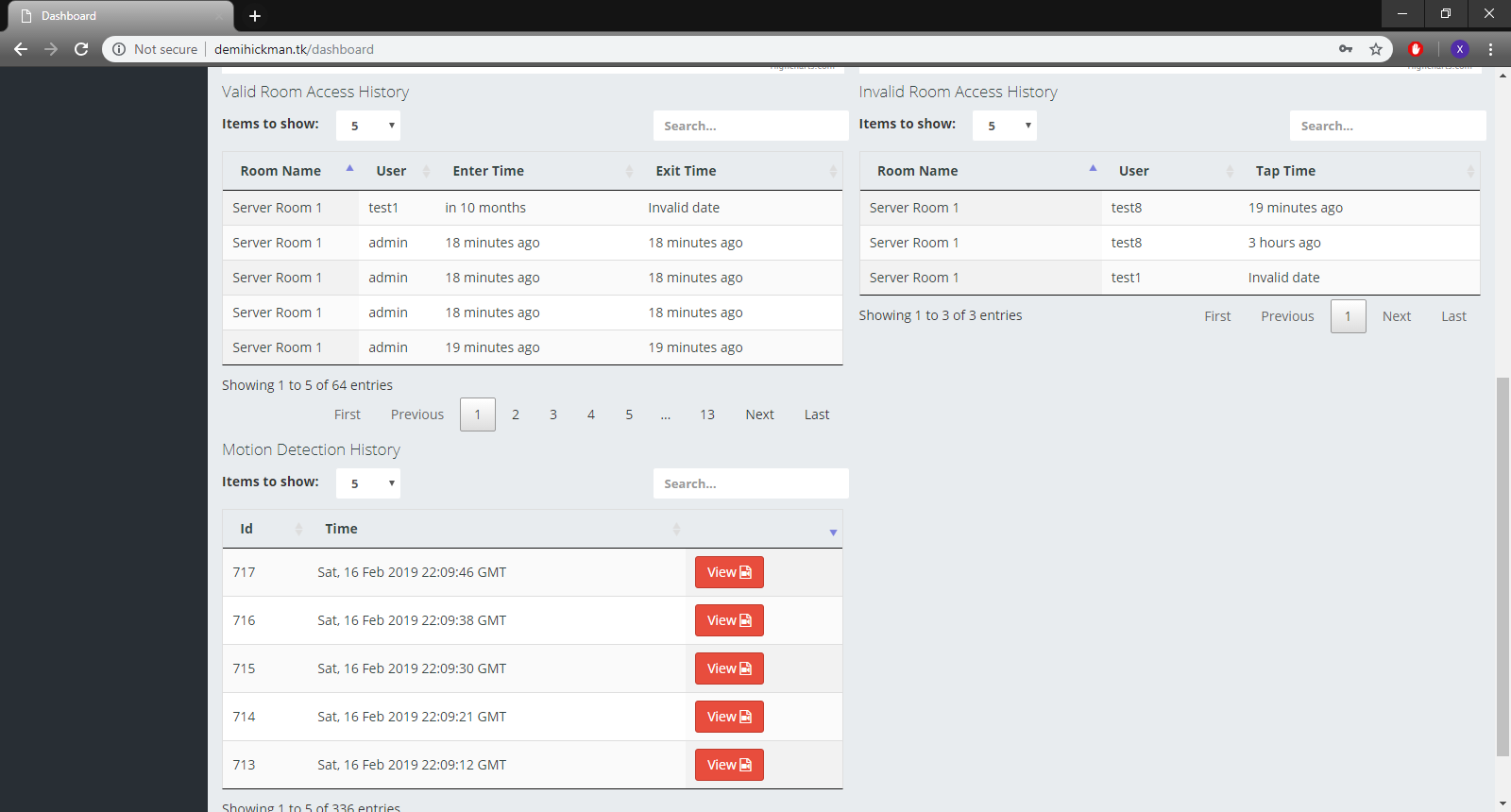
* 1. How does the final RPI set-up looks like?

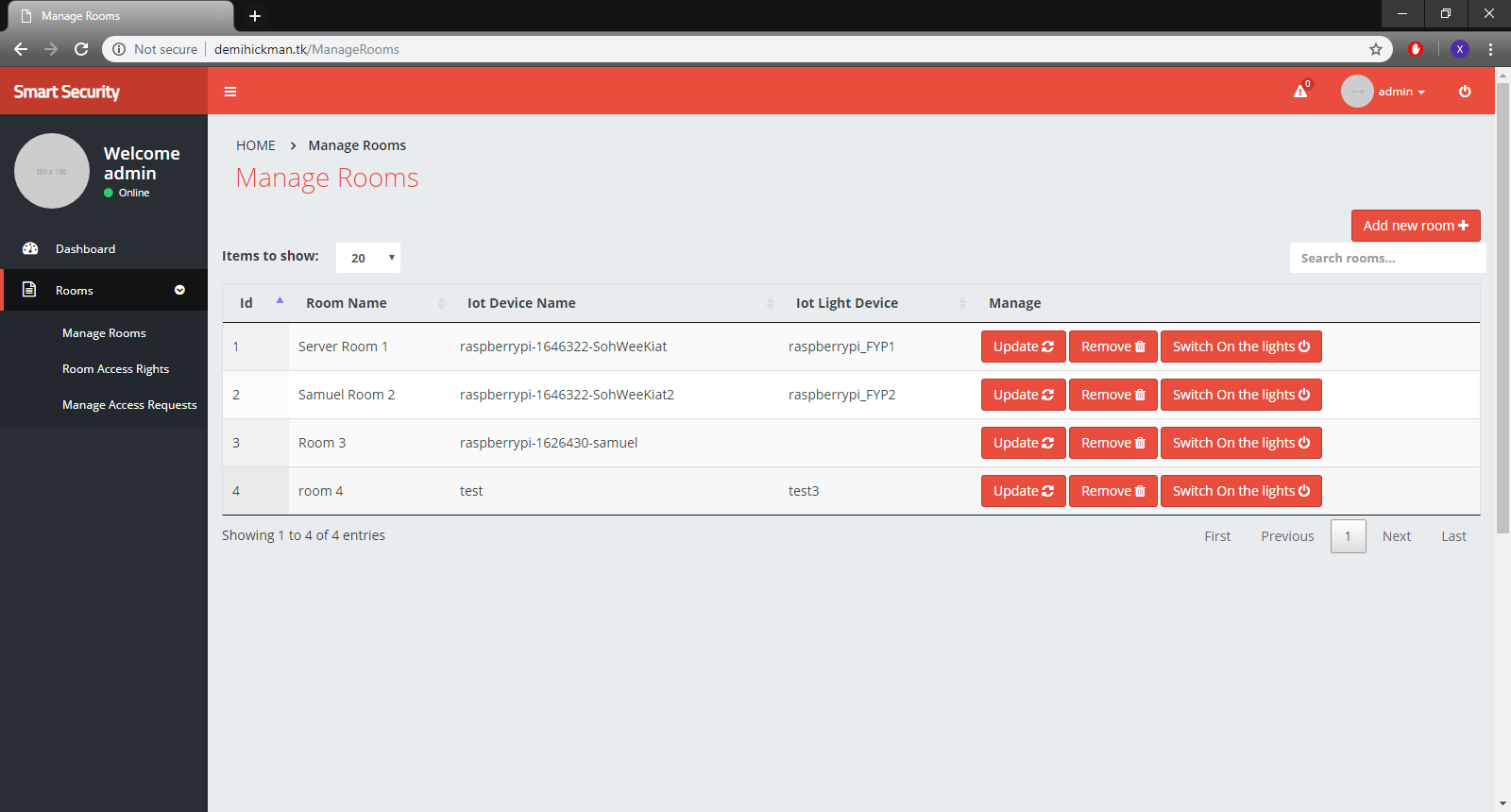


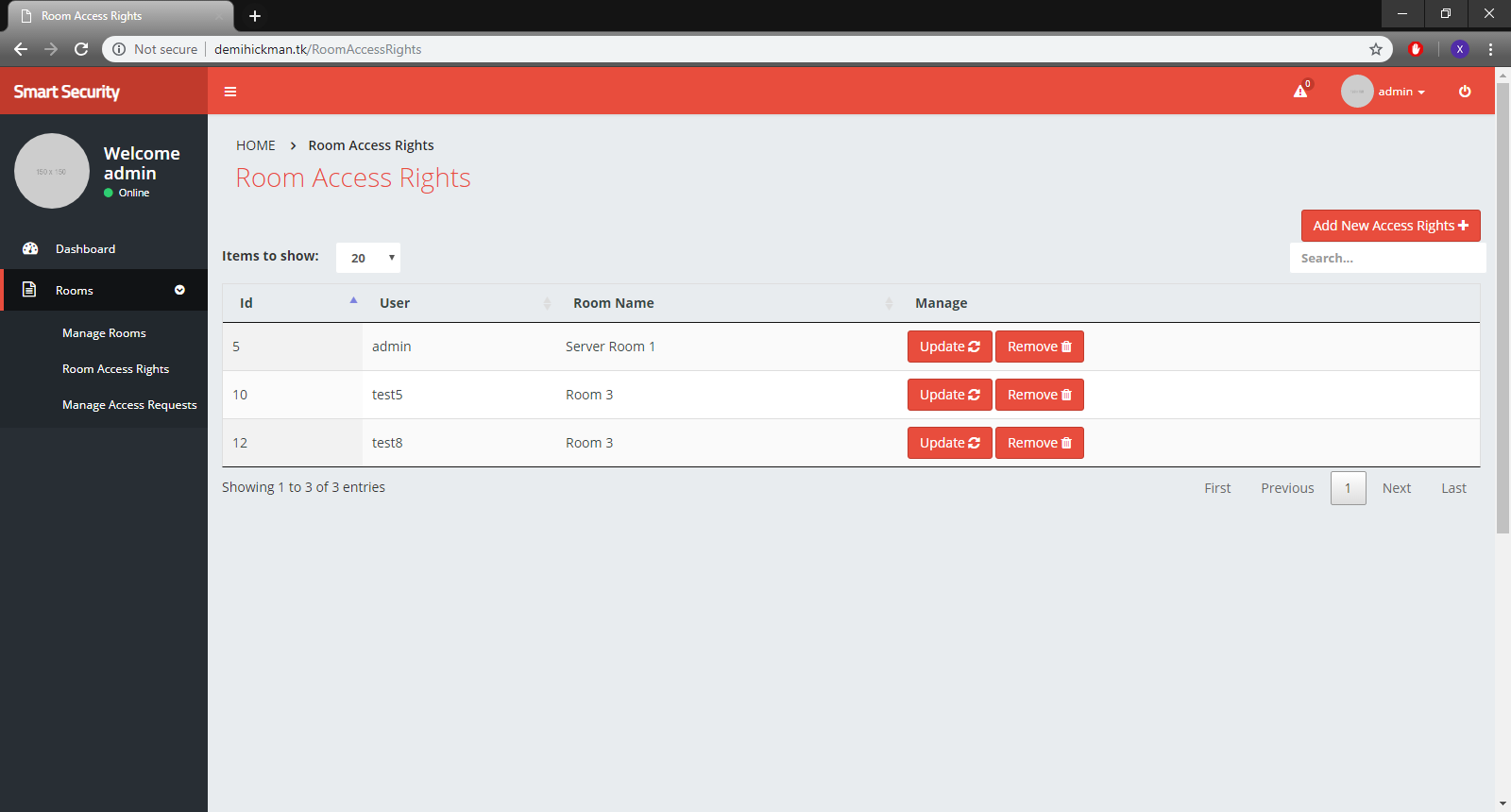
* 1. How does the web or mobile application look like?

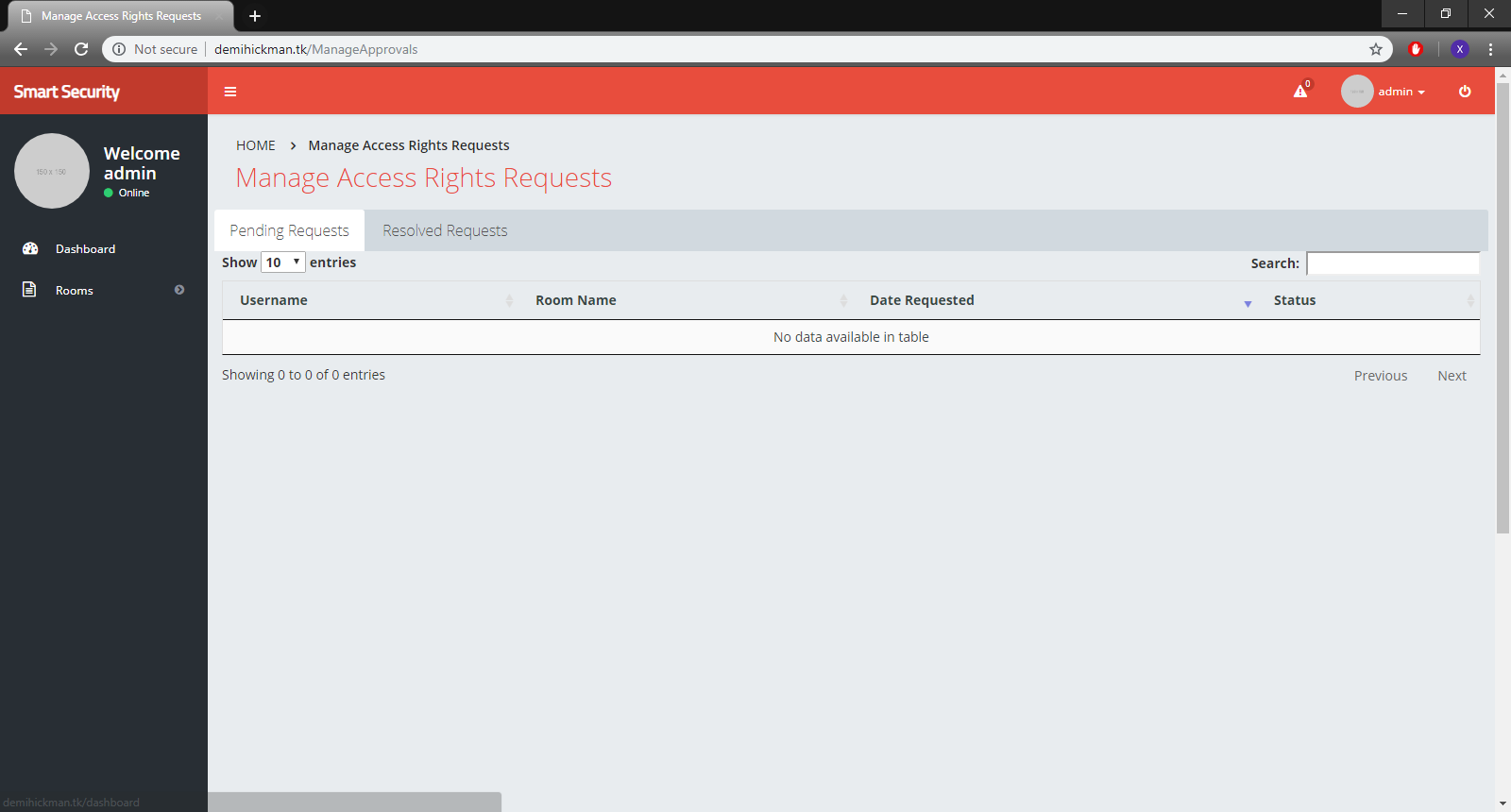




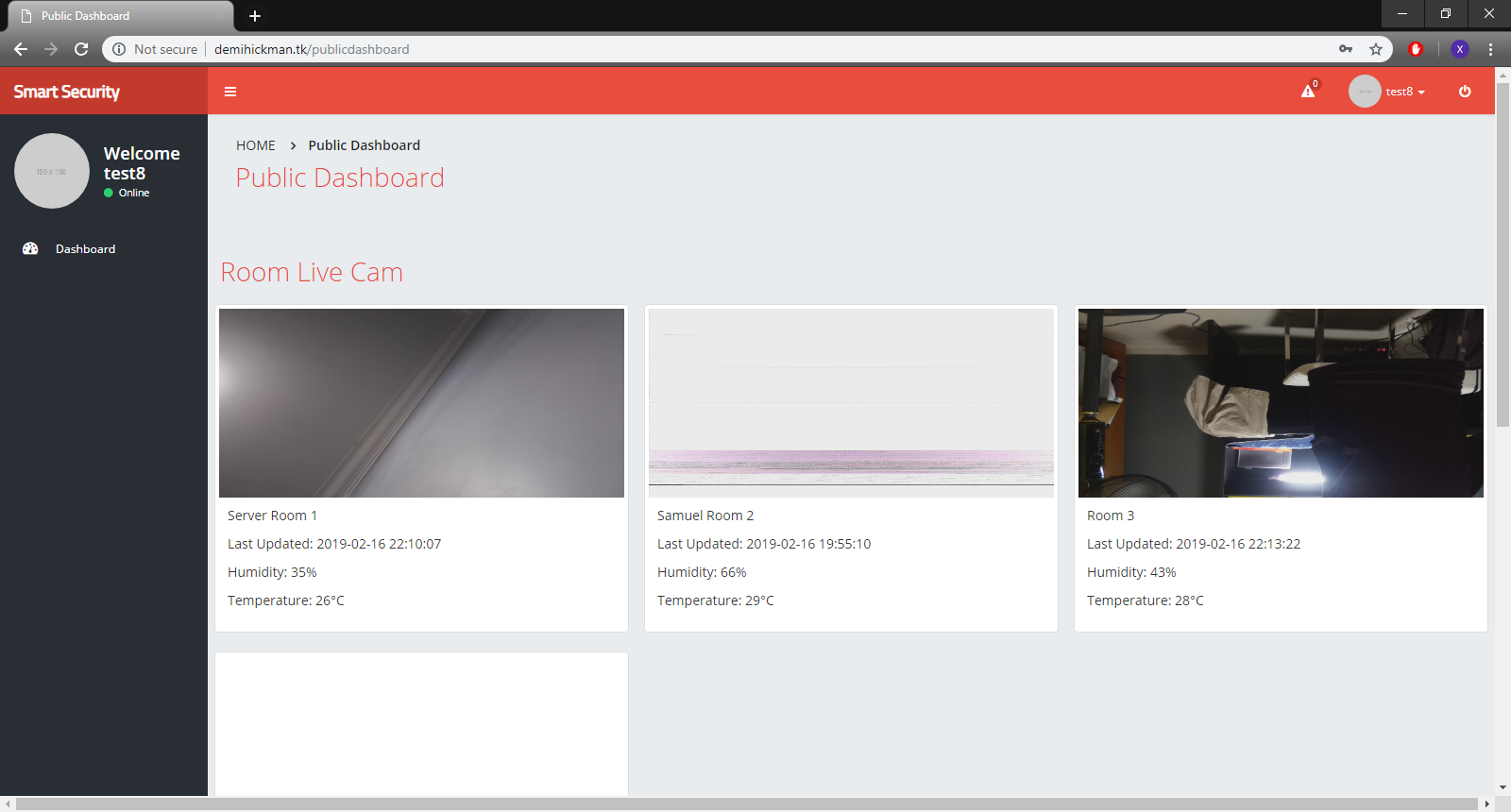


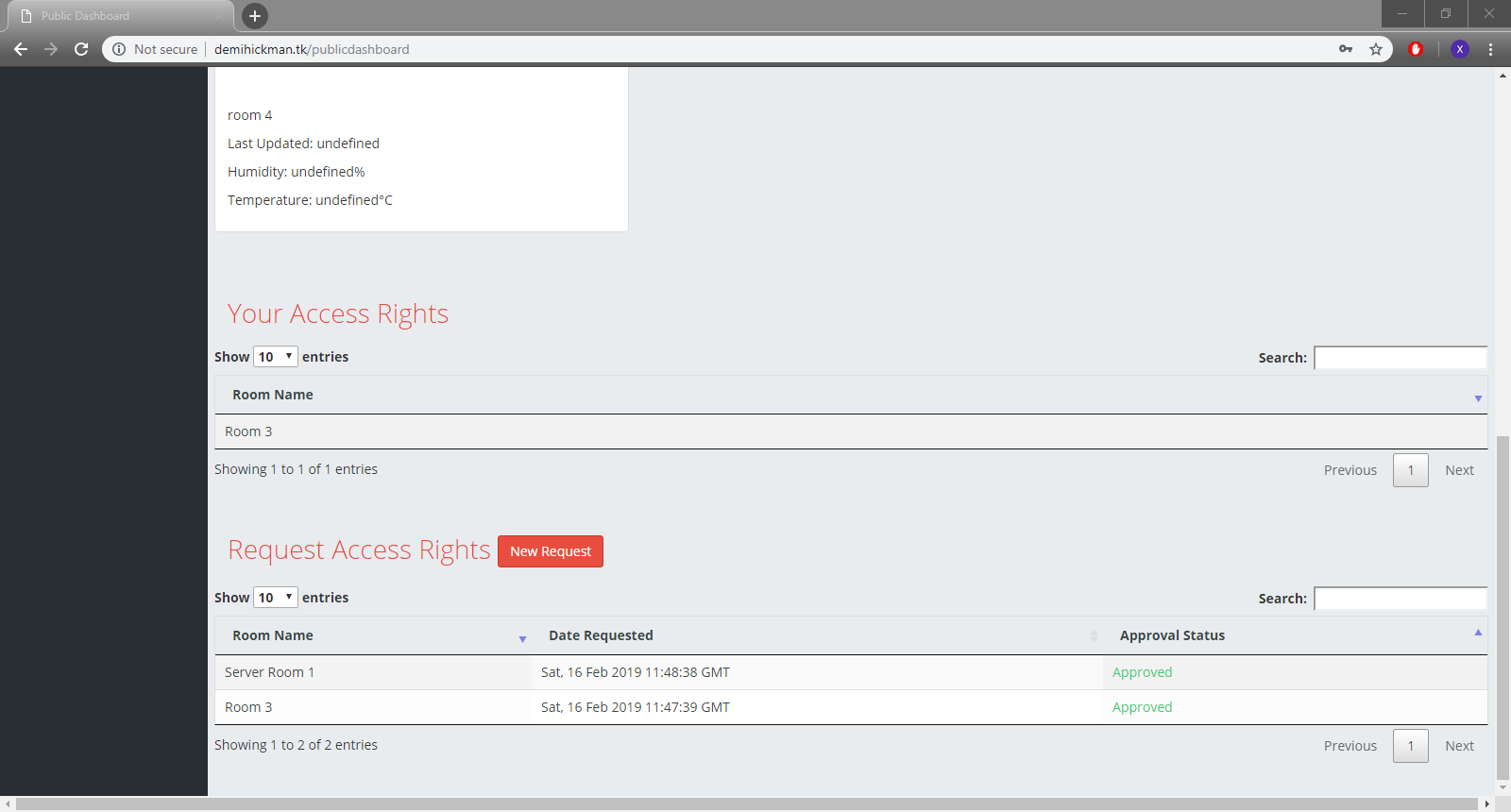












* 1. Member Contribution

|  |  |
| --- | --- |
| Soh Wee Kiat | * Base project from CA1   + Motion detection   + NFC card authenicator   + Dashboard     - Current temp graph     - Average temp graph     - Motion detection history     - Valid/Invalid access   + Telegram bot   + User/Room/CardId manager   + Login * AWS S3 * AWS EC2 hosting * AWS IOT MQTT * Matrix Creator LED light controller * Updated Telegram bot * Updated Database structure * Updated IoT codes |
| Samuel Chua | * Public dashboard   + Snapshot of rooms   + Current room stats * Registration * Manage access right request * Discord Bot * WhatsApp Bot * Github tutorial Writing * HTTPS certificate |

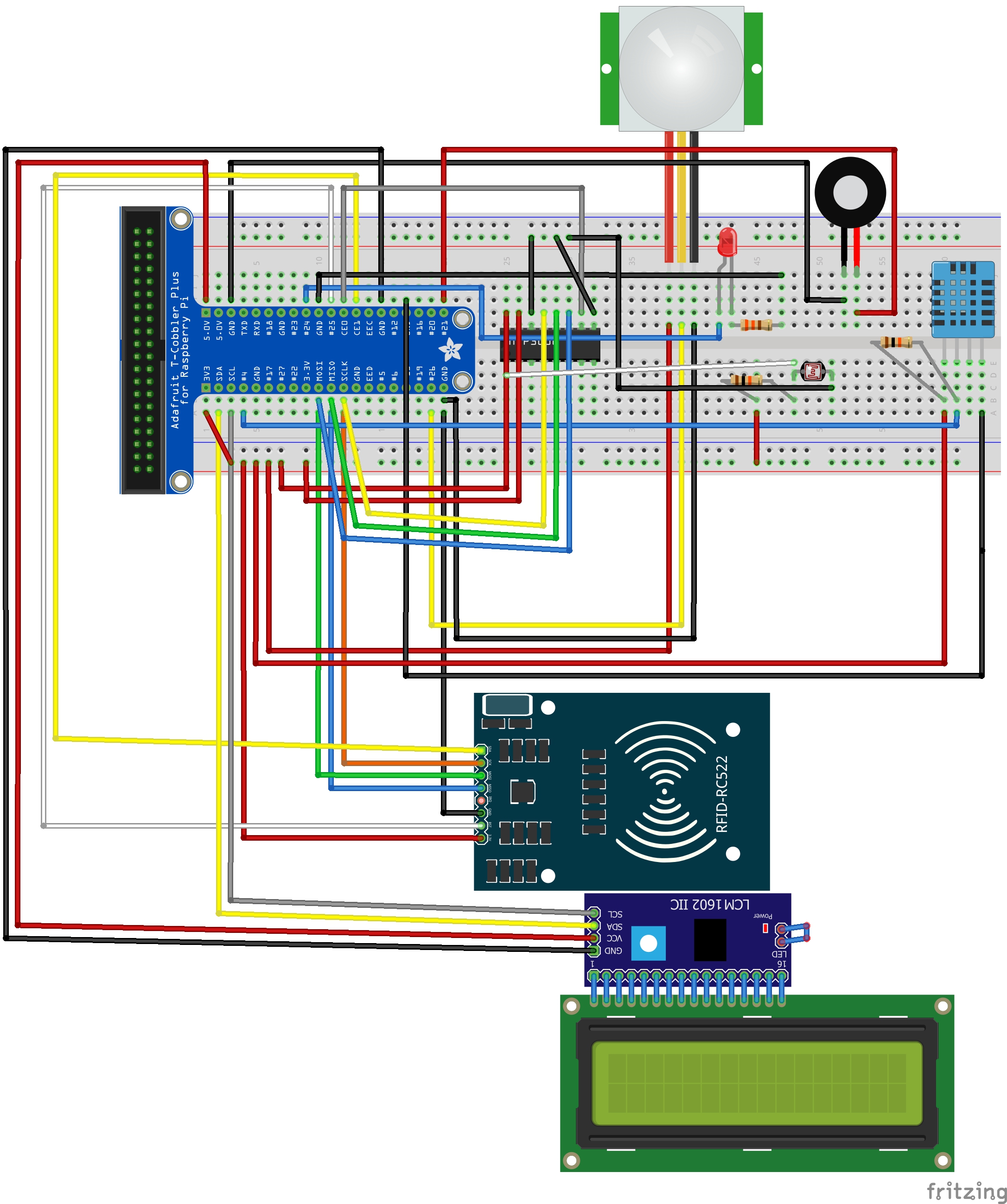
# Section 2 Hardware requirements

Hardware checklist

|  |  |  |
| --- | --- | --- |
| **Hardware Name** | **Quantity** | **Image** |
| 1. DHT11 Temperature and Humidity Sensor | 1 | •The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor.  •It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed).  •Its fairly simple to use, but requires careful timing to grab data.  •The only real downside of this sensor is you can only get new data from it once every 2 seconds  •The sensor that has been given to you has 4 pins.  oVCC – this is the one you connect to power  oDATA – this is an output value you read to determine the temperature  oNC – Stands for no connection  oGND – this is the one you connect to ground |
| 1. HC-SR501   PIR Motion Sensor | 1 | A PIR Sensor can tell when something nearby moves.  The sensor detects the pattern of infrared energy in its surroundings. If the pattern changes, the sensor outputs a high signal.  The sensor has 3 pins.   * + VCC – Connect this to power (5V or 3.3V)   + VOUT – Connect this to a GPIO pin to read its value   GND – Connect this to ground |
| 1. RFID/ NFC reader MFRC522 | 1 | •RFID technology is commonly used in many applications to “tag” items for the purpose of identification or tracking.  •For example, an RFID tag attached to an automobile during production can be used to track its progress through the assembly line; RFID-tagged pharmaceuticals can be tracked through warehouses; and implanting RFID microchips in livestock and pets allows animals to be identified.  •In this lab, you will write a simple Python program that is able to read RFID cards/ tags with a MFRC522 card reader module attached to RPi  •This reader is also capable of reading NFC Mifare A cards.  •NFC is based on RFID protocols but goes farther than RFID as it is capable of two-way communication—unlike RFID’s one-directional limitation |
| 1. piCam | 1 | •The Raspberry Pi camera is an official accessory that hooks up to a special connector on the Raspberry Pi.  •It can be used on any model of the Raspberry Pi except the first version of Pi Zero  •The camera is mounted on a small printed circuit board and connects through a ribbon cable.  •The connector provides direct access between the camera and the processor. This is more efficient than using a webcam, which needs to connect through the USB protocol  •The picture on the right shows the camera connected to a Raspberry Pi. |
| 1. I2C LCD Screen | 1 | •Adding a LCD display in your IoT projects will definitely spice it up by one notch! LCD displays are a fun and easy way to have your microcontroller project talk back to you.  •LCDs are available in tons of colors and sizes. For example, you might have 8x1 LCDs or 20x4 LCDs  •For this lab, we will use the commonly available 16x2 LCD which can display up to 32 characters.  •We will use the i2C version which require you to connect only 2 GPIO pins to your Raspberry Pi.  •If you are buying your own LCDs, do make sure you buy the i2C version though they might cost a bit more. The non-i2C versions require you to connect double the number of GPIO pins! |
| 1. LED | 1 | • The longer leg (anode) should be connected to the positive supply of the circuit (power)  • The shorter leg (cathode) should be connected to the negative side of the power supply (ground)  •LEDs will only work if power is supplied the correct way round  •You will not break the LEDs if you connect them the wrong way – they just don’t light.  •If the LED doesn’t light when you finish your lab later, maybe you have connected it the wrong way |
| 1. Buzzer | 1 | A buzzer is an audio signaling device which is commonly found in circuits to create a buzzing or beeping noise.  Buzzers can be categorized as active buzzers and passive ones. For our lab, we will use active buzzers as they are a lot simpler to use than passive ones though they are slightly more expensive.  An active buzzer can be connected just like an LED but they are even easier to use because a resistor is not needed.  A buzzer typically has 2 pins  oVOUT – Connect this to a GPIO pin to control its value  oGND – Connect this to ground |
| 1. Resistor for LED   **330Ω** | 1 | Figure 1: Orange,Orange,Brown   * Resistors can range from 100Ω to 10000Ω * The value of a resistor is marked with coloured bands along the length of the resistor body. * In this lab, you will be using a **330Ω** resistor. * You can identify the 330Ω resistors by the colour bands along the body. The colour coding will depend on how many bands are on the resistors supplied:   For example, on a four color band resistors, a 330Ω resistor is colored Orange, Orange, Brown, and then Gold. On a five color band 330Ω resistor, the colours are Orange, Orange, Black, Black, Brown. |
| 1. Resistor for DH11 Sensor   10K Ω | 2 | •To make this circuit work, we need to add in a “pull-up resistor” on the DATA line of our DHT11 sensor connected to the VCC line  •The pullup resistor is needed because the data line to the sensor is bidirectional.  •The sensor only sends data after its been told to by our RPi.  •For the circuit in this lab, we will need to add a 10K ohms resistor  •You can recognise a 10K ohms resistor by its color bands (brown:black:orange:gold) |
| 1. Light-Dependant Resistor (LDR) | 1 | * Light-Dependant Resistor (LDR) are light sensitive resistors which change resistance based on how much light they are exposed to. * The more light a LDR receives, the less resistant it becomes, i.e. lets more current flow * When it’s in the dark, the resistance is very high * The resistance of an LDR may typically have the following resistances:   + Daylight = 5000Ω   + Dark = 20000000Ω |
| 1. Analog-to-Digital Converter | 1 | * The Raspberry Pi has no built in analogue inputs which makes it difficult to work with analogue sensors such as temperature or light sensors which produce analog values. * RPi programmers get around this by using ADCs such as the affordable MCP3008 ($3.20) which are able to read analogue inputs and convert them to digital signals for the RPi. |

# Section 3 Setup Hardware on Raspberry Pi

## Completed Fritzing Diagram





## Connect the Components!

| Task | |
| --- | --- |
| Connect the PIR motion sensor with the RPi as follows: | |  |  |  | | --- | --- | --- | | **Jumper color** | **PIR pin** | **RPi pin** | | Red | VCC | 3.3V | | Black | GND | GND | | Yellow | VOUT | BCM26 | |

| Task | |
| --- | --- |
| There are about 9 pins on the MFRCF522 reader given in your IoT kit.  We will only be using 7 of them.  Connect the pins on the MFRCF522 card reader to the RPi as indicated below.   |  |  |  | | --- | --- | --- | | **Jumper color** | **MFRCF522pin** | **RPi pin** | | Yellow | SDA | CE0 | | Orange | SCK | SCLK | | Green | MOSI | MOSI | | Blue | MISO | MISO | |  | IDR |  | | Black | GND | GND | | White | RST | GPIO25 | | Red | 3.3V | 3.3V | |  | 5V |  | |  |

| Task | |
| --- | --- |
| * Connect your camera module to the CSI port on your Raspberry Pi; this is the long thin port adjacent to the HDMI socket. * Gently lift the collar on top of the CSI port and slide the ribbon cable of the camera module into the port with the blue side facing the Ethernet port * Once the cable is seated in the port, press the collar back down to lock the cable in place. * The picture shows a well-seated camera cable with the correct orientation. |  |

| Task | |
| --- | --- |
| Connect the pins on the LCD to the RPi as follows:   |  |  |  | | --- | --- | --- | | **Jumper color** | **LCD pin** | **RPi pin** | | White | SCL | SCL | | Yellow | SDA | SDA | | Black | GND | GND | | Red | Vcc | 5V | |  |

|  |
| --- |
| CONNECT LED AND RESISTOR |
| Plug in the LED into the breadboard as shown. Place the longer end of the LED nearer to the T-Cobbler Kit. | |
| Plug in the 330 Ω resistor into the breadboard as shown. One end of the resistor should be in the same column as the short leg of the LED. | |
| LED   1. Using a blue cable, connect the long end of LED to BCM24 | |
| Resistor (Orange-Orange-Brown)   1. Using a black cable, connect the free end of the 330 Ω resistor to GND | |

| Task | |
| --- | --- |
| Connect the Buzzer with the RPi as follows:   |  |  |  | | --- | --- | --- | | **Jumper color** | **PIR pin** | **RPi pin** | | Black | GND | GND | | Blue | VOUT | BCM21 | | **BCM21** |



## Add a 10k ohms resistor

|  | Task |
| --- | --- |
|  | Add a 10K ohms resistor in the breadboard as shown be;pw   * One end of the resistor should connected to **3.3V** * The other end of the resistor should be connected to **Pin 1** of the MCP3008 ADC and the **LDR**. |

# Section 4 Installing libraries On Raspberry Pi

## Enable SPI and prepare the MFRC522 libraries

If you are working on a fresh Raspbian install, there are some things you would need to prepare before the code given in the next section can work.

To save time, I have already pre-configured the ST0324 IoT Raspbian image so that you need not go through these steps anymore. I am including the steps in the lab for your future reference.

|  | Task | |
| --- | --- | --- |
| << Enable SPI via raspi-config >> | | |
|  | Run raspi-config, choose menu item “5 Interfacing Options” and enable SPI.  sudo rasp-config |  |
| << Enable device tree in boot.txt>> | | |
|  | Modify the /boot/config.txt to enable SPI  sudo nano /boot/config.txt | |
|  | Ensure these lines are included in *config.txt*  device\_tree\_param=spi=on  dtoverlay=spi-bcm2835 | |
|  | **<< Install Python-dev>>** | |
|  | Install the Python development libraries  sudo apt-get install python-dev | |
|  | **<< Install SPI-Py Library >>** | |
|  | Set up the SPI Python libraries since the card reader uses the SPI interface  cd ~  git clone https://github.com/lthiery/SPI-Py.git  cd ~/SPI-Py  sudo python setup.py install | |
|  | **<< Install RFID library >>** | |
|  | Clone the MFRC522-python library to your home folder as follows:-  cd ~  git clone https://github.com/mxgxw/MFRC522-python.git | |

## Install the necessary libraries (LCD)

You can skip this section as the necessary libraries have already been installed for you.

|  | Task |
| --- | --- |
|  | The LCD display is a relatively complicated device and the code to interface with it is not trivial.  Fortunately, the IoT community has written many LCD manipulation libraries which simplify the coding needed to talk to the LCD. Hence, we will use one of these libraries to make things easier for us!  There are many LCD libraries which you can use. For this lab, you will be using the **rpi-lcd** library created by Adam Bogdał which can be installed using step (b) below. |
|  | Install the rpi-lcd library using the commands below.  sudo pip install rpi-lcd |

## Install the necessary libraries (Bcrypt)

|  | Task |
| --- | --- |
|  | Install bcrypt (https://pypi.org/project/bcrypt/)  sudo pip install bcrypt |

## Install the necessary libraries (MySQL-Python)

|  | Task |
| --- | --- |
|  | First, ensure that the Python to MySQL libraries are installed on your RPI  sudo pip install mysql-connector-python  sudo pip install mysql.connector |

## Install the necessary libraries (Boto3)

|  | Task |
| --- | --- |
|  | First, ensure that the Python to boto3 libraries are installed on your RPI  sudo pip install boto3 |

## Install the necessary libraries (AWSIoTMQTTClient)

|  | Task |
| --- | --- |
|  | First, ensure that the Python to boto3 libraries are installed on your RPI  sudo pip install AWSIoTPythonSDK |

# Section 4 Installing libraries On Aws EC2 Server

## Install the necessary libraries (Flask)

|  | Task |
| --- | --- |
|  | Install Flask (http://flask.pocoo.org/)  sudo pip install flask |

## Install the necessary libraries (Bcrypt)

|  | Task |
| --- | --- |
|  | Install bcrypt (https://pypi.org/project/bcrypt/)  sudo pip install bcrypt |

## Install the necessary libraries (MySQL-Python)

|  | Task |
| --- | --- |
|  | First, ensure that the Python to MySQL libraries are installed on your RPI  sudo pip install mysql-connector-python  sudo pip install mysql.connector |

# 

# Section 5 Create mySQL database

| Task | |
| --- | --- |
| 1. Open mysql and connect to your respective mysql server |  |
| 1. Once you are logged in successfully to the mysql, you should see the dashboard similar to that below. | |
| 1. Click on the “Data Import” button | |
| 1. Click “…” button | |
| 1. Choose “Database.sql” file | |

# Section 6 Code the program

# (Shared codes between AWS & IoT)

## Configs.py

Next, let’s write the main python script that will be handling the tapping in and tapping out of users.

|  | Task |
| --- | --- |
|  | Create a folder to store your assignment files  mkdir ~/CA2 |
|  | Change directory to CA2  cd ~/CA2 |
|  | Create a folder to store your shared files  mkdir ~/Shared |
|  | Create a python script Configs.py with the code below  sudo nano ~/CA2/Shared/Configs.py |
|  | class Configs:  def \_\_init\_\_(self):  self.db\_host = "Fill in your own host"  self.db\_user = "fill in your own username"  self.db\_password = " fill in your own password "  self.databaseName = "iotdb"  self.aws\_region = "us-east-1"  self.aws\_endpoint = " fill in your own aws iot endpoint"  self.aws\_access\_key = "fill in your own access key"  self.aws\_secret\_key = "fill in your on secret key"  self.aws\_S3\_access\_key = "fill in your own s3 access key"  self.aws\_S3\_secret\_key = "fill in your own s3 secret key"  self.aws\_S3\_bucket = 'iotsmartroom'  self.aws\_S3\_endpoint = 'https://s3-us-west-2.amazonaws.com/iotsmartroom/'  global Config  Config = Configs() |

## Database.py

Next, let’s write the main python script that will be handling the tapping in and tapping out of users.

|  | Task |
| --- | --- |
| a) | Create a python script Database.py with the code below  sudo nano ~/CA2/Shared/Database.py |
| b) | import mysql.connector  import sys  from Shared.Configs import Config  class Database:  def query(self,sql):  try:  cnx = mysql.connector.connect(host=Config.db\_host,user=Config.db\_user,password=Config.db\_password  ,database=Config.databaseName,autocommit=True)  cursor = cnx.cursor()  cursor.execute(sql)  row\_headers=[x[0] for x in cursor.description]  results = cursor.fetchall()  data = []  for result in results:  data.append(dict(zip(row\_headers,result)))  return data  except:  print(sys.exc\_info()[0])  print(sys.exc\_info()[1])  return None  def insert(self,sql):  try:  cnx = mysql.connector.connect(host=Config.db\_host,user=Config.db\_user,password=Config.db\_password  ,database=Config.databaseName,autocommit=True)  cursor = cnx.cursor()  cursor.execute(sql)  cnx.commit()  return cursor.lastrowid  except:  print(sys.exc\_info()[0])  print(sys.exc\_info()[1])  return 0  def update(self,sql):  try:  cnx = mysql.connector.connect(host=Config.db\_host,user=Config.db\_user,password=Config.db\_password  ,database=Config.databaseName,autocommit=True)  cursor = cnx.cursor()  cursor.execute(sql)  cnx.commit()  return True  except:  print(sys.exc\_info()[0])  print(sys.exc\_info()[1])  return False  def remove(self,sql):  try:  cnx = mysql.connector.connect(host=Config.db\_host,user=Config.db\_user,password=Config.db\_password  ,database=Config.databaseName,autocommit=True)  cursor = cnx.cursor()  cursor.execute(sql)  cnx.commit()  return True  except:  print(sys.exc\_info()[0])  print(sys.exc\_info()[1])  return False  global db  db = Database() |

## helpers.py

helpers.py is a python script that contains misc functions such as data\_to\_json

|  | Task |
| --- | --- |
|  | Create a python script helpers.py with the code below   |  | | --- | | sudo nano ~/CA2/Shared/helpers.py | |
|  | import json  import numpy  import datetime  import decimal  class GenericEncoder(json.JSONEncoder):    def default(self, obj):  if isinstance(obj, numpy.generic):  return numpy.asscalar(obj)  elif isinstance(obj, datetime.datetime):  return obj.strftime('%Y-%m-%d %H:%M:%S')  elif isinstance(obj, decimal.Decimal):  return float(obj)  else:  return json.JSONEncoder.default(self, obj)  def data\_to\_json(data):  json\_data = json.dumps(data,cls=GenericEncoder)  return json.loads(json\_data) |

## Room.py

Room.py is a python class script that contains Room class, with all the sql queries

|  | Task |
| --- | --- |
|  | Create a python script Room.py with the code below   |  | | --- | | sudo nano ~/CA2/Shared/Room.py | |
|  | from Shared.Database import db  from Shared.helpers import data\_to\_json  class Room:  def \_\_init\_\_(self,\*\*kwargs):  self.Id = int(kwargs.get('Id', 0))  self.RoomName = kwargs.get('RoomName', "")  self.IotDeviceName = kwargs.get('IotDeviceName', "")  self.IoTLightDevice = kwargs.get('IoTLightDevice', "")  def IsValidForAdd(self):  return self.RoomName != "" and self.IotDeviceName != ""  def IsValidForUpdate(self):  return self.Id > 0 and self.IsValidForAdd()  def TryAdd(self):  self.Id = db.insert("insert into rooms(RoomName,IotDeviceName,IoTLightDevice) VALUES('{}','{}','{}')"  .format(self.RoomName,self.IotDeviceName,self.IoTLightDevice))  return self.Id > 0  def TryUpdateDb(self):  return db.update("update rooms set RoomName = '{}', IotDeviceName = '{}', IoTLightDevice = '{}' where Id = {}"  .format(self.RoomName,self.IotDeviceName,self.IoTLightDevice,self.Id))  def TryRemove(self):  return db.remove("delete from rooms where Id = {}"  .format(self.Id))  @staticmethod  def ParseFromForm(Form):  Id = Form.get('IotId')  RoomName = Form.get('RoomName')  IotDeviceName = Form.get('IotDeviceName')  IoTLightDevice = Form.get('IoTLightDevice')  if Id == None: Id = 0  if RoomName == None: RoomName = ""  if IotDeviceName == None: IotDeviceName = ""  return Room(Id = Id,RoomName = RoomName,IotDeviceName = IotDeviceName,IoTLightDevice = IoTLightDevice)  @staticmethod  def TryGetRoom(DeviceName):  results = db.query("select \* from rooms where IotDeviceName = '{}'"  .format(DeviceName))  if len(results) <= 0:  return None  result = results[0]  return Room(Id = result["Id"],RoomName = result["RoomName"],IotDeviceName = result["IotDeviceName"],IoTLightDevice = result["IoTLightDevice"])  @staticmethod  def TryGetRoomByLight(DeviceName):  results = db.query("select \* from rooms where IoTLightDevice = '{}'"  .format(DeviceName))  if len(results) <= 0:  return None  result = results[0]  return Room(Id = result["Id"],RoomName = result["RoomName"],IotDeviceName = result["IotDeviceName"],IoTLightDevice = result["IoTLightDevice"])  @staticmethod  def TryGetRoomById(Id):  results = db.query("select \* from rooms where Id = {}"  .format(Id))  if results == None or len(results) <= 0:  return None  result = results[0]  return Room(Id = result["Id"],RoomName = result["RoomName"],IotDeviceName = result["IotDeviceName"],IoTLightDevice = result["IoTLightDevice"])  @staticmethod  def GetAllRoomsJSON():  return data\_to\_json(db.query("select \* from rooms"))  @staticmethod  def GetAllRooms():  RoomsList = []  results = db.query("select \* from rooms")  if results == None or len(results) <= 0:  return None  for r in results:  RoomsList.append(Room(Id = r["Id"],RoomName = r["RoomName"],IotDeviceName = r["IotDeviceName"],IoTLightDevice = r["IoTLightDevice"]))  return RoomsList  @staticmethod  def TryGetRoomByRoomName(RoomName):  RoomsList = []  results = db.query("select \* from rooms where lower(RoomName) = '{}'".format(RoomName.lower()))  if results == None or len(results) <= 0:  return None  for r in results:  RoomsList.append(Room(Id = r["Id"],RoomName = r["RoomName"],IotDeviceName = r["IotDeviceName"],IoTLightDevice = r["IoTLightDevice"]))  return RoomsList |

## EnviroInfo.py

EnviroInfo.py is a python class script that contains EnviroInfo class, with all the sql queries, stores tempature,humidity, time taken and room id

|  | Task |
| --- | --- |
|  | Create a python script EnviroInfo.py with the code below   |  | | --- | | sudo nano ~/CA2/Shared/EnviroInfo.py | |
|  | from Shared.Database import db  from Shared.helpers import data\_to\_json  import time  class EnviroInfo:  def \_\_init\_\_(self, \*\*kwargs):  self.Id = kwargs.get('Id', 0)  self.temp = kwargs.get('temp', 0)  self.humidity = kwargs.get('humidity', 0)  self.light\_value = kwargs.get('light\_value', 0)  self.time = kwargs.get('time', time.strftime('%Y-%m-%d %H:%M:%S'))  self.roomId = kwargs.get('roomId', 0)  def WriteToDb(self):  self.Id = db.insert("insert into enviro\_info(temp,humidity,time,roomId,light\_value) VALUES({},{},'{}',{},{})"  .format(self.temp,self.humidity,self.time,self.roomId,self.light\_value))  if self.Id > 0:  return True  else:  return False  @staticmethod  def GetTop10EnviroInfo(RoomId):  db\_results = db.query("SELECT ei.temp, ei.humidity, ei.time FROM enviro\_info ei where ei.roomId = {} order by ei.time desc limit 10"  .format(RoomId))  results\_flipped = db\_results[::-1]  humid = []  temp = []  for x in results\_flipped:  humid.append([x['time'],x['humidity']])  temp.append([x['time'],x['temp']])  return data\_to\_json({"temp": temp,"humidity":humid})  @staticmethod  def GetAvgEnviroInfoByDay(RoomId):  db\_results = db.query("SELECT Avg(temp) 'AvgTemp', time from enviro\_info where roomId = {} group by DAY(time)"  .format(RoomId))  humid = []  temp = []  for x in db\_results:  temp.append([x['time'],x['AvgTemp']])  return data\_to\_json(temp)  @staticmethod  def GetLatestEnviroInfo(RoomId):  db\_results = db.query("SELECT ei.temp, ei.humidity, ei.light\_value, ei.time FROM enviro\_info ei where ei.roomId = {} order by ei.time desc limit 1"  .format(RoomId))  if db\_results == None or len(db\_results) <= 0:  return None  result = db\_results[0]  return data\_to\_json({"time": result['time'], "temp": result['temp'],"humidity": result['humidity'], "light": result['light\_value']}) |

## AccessLog.py

AccessLog.py is a python class script that contains AccessLog class, with all the sql queries, stores room id, card id, access time, exit time

|  | Task |
| --- | --- |
|  | Create a python script AccessLog.py with the code below   |  | | --- | | sudo nano ~/CA2/Shared/AccessLog.py | |
|  | from Shared.Database import db  from Shared.helpers import data\_to\_json  from Shared.Room import Room  import time  class AccessLog:  def \_\_init\_\_(self,\*\*kwargs):  self.Id = int(kwargs.get('Id', 0))  self.RoomId = int(kwargs.get('RoomId', 0))  self.UserId = kwargs.get('UserId', "")  self.Time = kwargs.get('Time', time.strftime('%Y-%m-%d %H:%M:%S'))  self.ExitTime = kwargs.get('ExitTime', "")  self.IsValid = kwargs.get('IsValid', False)  if self.RoomId > 0:  self.room = Room.TryGetRoomById(self.RoomId)  else:  self.room = None  def UpdateExitTime(self):  self.ExitTime = time.strftime('%Y-%m-%d %H:%M:%S')  self.TryUpdateDb()  def TryAdd(self):  self.Id = db.insert("insert into access\_logs(RoomId,UserId,Time,IsValid) VALUES({},'{}','{}',{})"  .format(self.RoomId,self.UserId,self.Time,self.IsValid))  return self.Id > 0  def TryUpdateDb(self):  return db.update("update access\_logs set Exit\_time = '{}' where Id = {}"  .format(self.ExitTime,self.Id))  @staticmethod  def GetAccessLogForRoom(RoomId):  db\_results = db.query("select \* from access\_logs where RoomId = {} order by Time".format(RoomId))  if db\_results == None or len(db\_results) <= 0:  return {}  return db\_results  @staticmethod  def GetValidExitedAccessLogForRoom(RoomId):  db\_results = db.query("SELECT r.RoomName, u.Username, al.Time, al.Exit\_time FROM access\_logs al " +  "JOIN rooms r on al.RoomId = r.Id " +  "JOIN users u on al.UserId = u.Id " +  "where al.RoomId = {} and al.IsValid = 1 order by al.Time desc".format(RoomId))  if db\_results == None or len(db\_results) <= 0:  return {}  return db\_results  @staticmethod  def GetInvalidExitedAccessLogForRoom(RoomId):  db\_results = db.query("SELECT r.RoomName, u.Username, al.Time FROM access\_logs al " +  "JOIN rooms r on al.RoomId = r.Id " +  "JOIN users u on al.UserId = u.Id " +  "where al.RoomId = {} and al.IsValid = 0 order by al.Time desc".format(RoomId))  if db\_results == None or len(db\_results) <= 0:  return {}  return db\_results  @staticmethod  def GetLatestExitAccessLog(RoomId):  db\_results = db.query("SELECT r.RoomName,al.Id, al.Time, al.Exit\_time FROM access\_logs al " +  "JOIN rooms r on al.RoomId = r.Id where al.IsValid = 1 and al.RoomId = {} order by al.Id desc LIMIT 1"  .format(RoomId))  if db\_results == None or len(db\_results) <= 0:  return {}  return db\_results  @staticmethod  def IsRoomAvailable(RoomId):  db\_results = db.query("SELECT \* FROM access\_logs al " +  "where al.RoomId = {} and al.IsValid = 1 and al.Exit\_time is null".format(RoomId))  if db\_results == None or len(db\_results) <= 0:  return True  return False |

## User.py

User.py is a python class script that contains User class, with all the sql queries, stores username, hashpassword, telegram chat id

|  | Task |
| --- | --- |
|  | Create a python script User.py with the code below   |  | | --- | | sudo nano ~/CA2/Shared/User.py | |
|  | from Shared.Database import db  import bcrypt  import time  class User:  def \_\_init\_\_(self,\*\*kwargs):  self.Id = kwargs.get('Id', 0)  self.Username = kwargs.get('Username', "")  self.Password = kwargs.get('Password', "")  self.reTypePassword = kwargs.get('reTypePassword', "")  self.LastLogin = kwargs.get('LastLogin', "")  self.UserType = kwargs.get('UserType', 0)  self.CardId = kwargs.get('CardId', "")  self.ChatId = kwargs.get('ChatId', 0)  def IsValid(self):  return self.Username != "" and self.Password != ""  def IsAdmin(self):  return self.UserType == 1  def registerUser(self):  if self.Username != "" and self.Password != "":  if self.Password == self.reTypePassword:  hashed = bcrypt.hashpw(self.Password.encode("utf-8"), bcrypt.gensalt()).decode("utf-8")  result = db.insert("insert into users(username,PasswordHash) VALUES('{}','{}')".format(self.Username,hashed))  return result  return "2"  return "1"  def UpdateLastLogin(self):  LastLogin = time.strftime('%Y-%m-%d %H:%M:%S')  db.update("update users set LastLogin = '{}' where Id = {}"  .format(LastLogin,self.Id))  def UpdateChatId(self,ChatId):  db.update("update users set ChatId = {} where Id = {}"  .format(int(ChatId),self.Id))  def TryLogin(self):  result = db.query("select \* from users where Username = '{}'"  .format(self.Username))  if result == None or len(result) <= 0:  return False  StoredHash = result[0]['PasswordHash'].encode("utf-8")  if bcrypt.hashpw(self.Password.encode("utf-8"),StoredHash) != StoredHash:  return False  self.Id = result[0]['Id']  self.CardId = result[0]['CardId']  self.UserType = result[0]['UserType']  self.UpdateLastLogin()  return True  @staticmethod  def ParseFromForm(Form):  username = Form.get('username')  password = Form.get('password')  if username == None: username = ""  if password == None: password = ""  return User(Username = username,Password = password)  @staticmethod  def ParseRegistrationForm(Form):  username = Form.get('username')  password = Form.get('password')  retypepassword = Form.get('retypepassword')  if username == None: username = ""  if password == None: password = ""  if retypepassword == None: retypepassword = ""  return User(Username = username,Password = password,reTypePassword = retypepassword)  @staticmethod  def TryGetUserById(Id):  results = db.query("select \* from users where Id = {}"  .format(Id))  if len(results) <= 0:  return None  result = results[0]  return User(Id = result['Id'],Username = result['Username'],ChatId = result['ChatId'],CardId = result['CardId'])  @staticmethod  def GetUsers():  UsersList = []  results = db.query("select \* from users")  if len(results) <= 0:  return None  for r in results:  UsersList.append(User(Id = r['Id'],Username = r['Username'],ChatId = r['ChatId'],CardId = r['CardId']))  return UsersList  @staticmethod  def GetChatIdUsers():  UsersList = []  results = db.query("select \* from users where ChatId is not null")  if results == None or len(results) <= 0:  return None  for r in results:  UsersList.append(User(Id = r['Id'],Username = r['Username'],ChatId = r['ChatId'],CardId = r['CardId']))  return UsersList  @staticmethod  def TryGetUserByCardId(CardId):  results = db.query("select \* from users where CardId = '{}'"  .format(CardId))  if results == None or len(results) <= 0:  return None  result = results[0]  return User(Id = result['Id'],Username = result['Username'],ChatId = result['ChatId'],CardId = result['CardId'])  @staticmethod  def addCardId(cardID, userID):  if (cardID != None):  id = db.update("update users set cardId = '{}' where Id = {}".format(cardID, userID))  return id  return None |

## AccessRight.py

AccessRight.py is a python class script that contains AccessRight class, with all the sql queries, stores Card Id, room id, user Id

|  | Task |
| --- | --- |
|  | Create a python script AccessRight.py with the code below   |  | | --- | | sudo nano ~/CA2/Shared/AccessRight.py | |
|  | from Shared.Database import db  from Shared.helpers import data\_to\_json  from Shared.User import User  from Shared.Room import Room  class AccessRight:  def \_\_init\_\_(self,\*\*kwargs):  self.Id = int(kwargs.get('Id', 0))  self.RoomId = int(kwargs.get('RoomId', 0))  self.UserId = int(kwargs.get('UserId', 0))  if self.RoomId > 0:  self.room = Room.TryGetRoomById(self.RoomId)  else:  self.room = None  if self.UserId > 0:  self.user = User.TryGetUserById(self.UserId)  else:  self.user = None  def IsValidForRemove(self):  return self.Id > 0  def IsValidForAdd(self):  return self.RoomId != 0 and self.UserId != 0  def IsValidForUpdate(self):  return self.Id > 0 and self.IsValidForAdd()  def TryAdd(self):  if self.HasAccessRight():  return False  self.Id = db.insert("insert into access\_rights(roomId,userId) VALUES({},{})"  .format(self.RoomId,self.UserId))  return self.Id > 0  def TryUpdateDb(self):  return db.update("update access\_rights set roomId = {}, userId = {} where Id = {}"  .format(self.RoomId,self.UserId,self.Id))  def TryRemove(self):  return db.remove("delete from access\_rights where Id = {}"  .format(self.Id))  def HasAccessRight(self):  return len(db.query("SELECT \* from access\_rights where userId = {} and roomId = {}".format(self.UserId,self.RoomId))) > 0  @staticmethod  def ParseFromForm(Form):  Id = Form.get('Id')  UserId = Form.get('UserId')  RoomId = Form.get('RoomId')  if Id == None: Id = 0  if UserId == None: UserId = 0  if RoomId == None: RoomId = 0  return AccessRight(Id = Id,UserId = UserId,RoomId = RoomId)  @staticmethod  def getUserAccessRights(UserId):  return data\_to\_json(db.query("SELECT r.RoomName from access\_rights ac " +  "JOIN rooms r ON r.Id = ac.RoomId where ac.UserId = {}".format(UserId)))  @staticmethod  def GetAllAccessRights():  return data\_to\_json(db.query("SELECT ac.Id, u.Username,u.Id UserId, r.RoomName,r.Id RoomId from access\_rights ac " +  "JOIN users u ON u.Id = ac.UserId " +  "JOIN rooms r ON r.Id = ac.RoomId")) |

## MotionEvent.py

MotionEvent.py is a python class script that contains MotionEvent class, contains sql queries to store and retrieve motion event, room Id , time and file path

|  | Task |
| --- | --- |
|  | Create a python script MotionEvent.py with the code below   |  | | --- | | sudo nano ~/CA2/Shared/MotionEvent.py | |
|  | from Shared.Database import db  from Shared.helpers import data\_to\_json  import time  class MotionEvent:  def \_\_init\_\_(self,\*\*kwargs):  self.Id = int(kwargs.get('Id', 0))  self.RoomId = int(kwargs.get('RoomId', 0))  self.Time = kwargs.get('Time', time.strftime('%Y-%m-%d %H:%M:%S'))  self.FilePath = kwargs.get('FilePath', "")  def TryAdd(self):  self.Id = db.insert("insert into motion\_events(RoomId,Time,FilePath) VALUES({},'{}','{}')"  .format(self.RoomId,self.Time,self.FilePath))  return self.Id > 0  def GetDirectory(self):  index = self.FilePath.rindex("/")  return self.FilePath[:index]  def GetFileName(self):  index = self.FilePath.rindex("/") + 1  return self.FilePath[index:]  @staticmethod  def GetMotionEventsCount():  results = db.query("select \* from motion\_events")  if results == None:  return 0  return len(results)  @staticmethod  def GetMotionEvents(roomId):  results = db.query("select me.Id, r.RoomName, me.Time, me.FilePath from motion\_events me JOIN rooms r ON me.RoomId = r.Id where me.RoomId = {} order by me.Time desc".format(roomId))  if results == None or len(results) <= 0:  return {}  return results  @staticmethod  def GetMotionEventsById(Id):  results = db.query("select \* from motion\_events where Id = {}".format(Id))  if len(results) <= 0:  return None  result = results[0]  return MotionEvent(Id = result["Id"],RoomId = result["RoomId"],Time = result["Time"],FilePath = result["FilePath"]) |

## AccessRequest.py

AccessRequest.py is a python class script that contains Access Request class, contains sql queries to retrieve the access requests of users, insert new access requests of users and update access rights of users if an access request is approved.

|  | Task |
| --- | --- |
| a) | Create a python script MotionEvent.py with the code below   |  | | --- | | sudo nano ~/CA2/Shared/AccessRequest.py | |
| b) | from Shared.Database import db  from Shared.helpers import data\_to\_json  from Shared.Room import Room  import datetime  class AccessRequest:  @staticmethod  def getUserAccessRequests(UserId):  db\_results = db.query("SELECT r.RoomName, ar.DateRequested, ar.IsApproved FROM access\_request ar "  "JOIN rooms r on ar.RoomId = r.Id where ar.UserId = {}".format(UserId))  if db\_results == None or len(db\_results) <= 0:  return {}  return db\_results  @staticmethod  def checkIfAlreadyRequested(roomID, userID):  currentTime = datetime.datetime.now()  db\_results = db.query("SELECT \* from access\_request where userID = {} and roomID = {}".format(userID, roomID))  #have not created a request to this room  if db\_results == None or len(db\_results) <= 0:  return 0  result = db\_results[0]  status = result['IsApproved']  if status is not None:  if status == 1:  return 1  if status == 0:  approvalDelay = result['DateRequested'] + datetime.timedelta(minutes = 10)  if currentTime < approvalDelay:  return 2  return 3  else:  return 4  @staticmethod  def requestNewAccess(roomID, userID):  currentTime = datetime.datetime.now()  id = db.insert("insert into access\_request(userid, roomid, DateRequested) values ({}, {},'{}')"  .format(userID, roomID, currentTime))  return id  @staticmethod  def getAllPendingRequests():  db\_results = db.query("SELECT ar.Id, ud.Username, r.RoomName, ar.DateRequested, ar.IsApproved FROM access\_request ar " +  "JOIN rooms r on ar.RoomId = r.Id " +  "JOIN users ud on ar.UserId = ud.Id where ar.IsApproved is null")  if db\_results == None or len(db\_results) <= 0:  return {}  return db\_results  @staticmethod  def getCompletedRequests():  db\_results = db.query("SELECT ud.Username, r.RoomName, ar.DateRequested, ar.IsApproved FROM access\_request ar " +  "JOIN rooms r on ar.RoomId = r.Id " +  "JOIN users ud on ar.UserId = ud.Id where ar.IsApproved is not null")  if db\_results == None or len(db\_results) <= 0:  return {}  return db\_results  @staticmethod  def updateRequestApprovalStatus(requestID, approval):  if (requestID != None and approval != None):  result = db.update("update access\_request set IsApproved = {} where Id = {}".format(approval, requestID))  if result is True and approval == '1':  requestObject = db.query("SELECT roomID, UserId from access\_request where id = {}".format(requestID))  result = requestObject[0]  id = db.insert("insert into access\_rights(RoomId, UserId) values({}, {})".format(result['roomID'], result['UserId']))  return result  else:  return False |

# Section 7 Code the program (IoT)

## AwsIot.py

AwsIot.py is a python class script that contains AwsIot class, contains basic constructor to start up aws iot mqtt

|  | Task |
| --- | --- |
|  | Create a python script AwsIot.py with the code below   |  | | --- | | sudo nano ~/CA2/AwsIot.py | |
|  | from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient  host = "a1mnegx95sc2wp-ats.iot.us-east-1.amazonaws.com"  rootCAPath = "certs/root-CA.crt"  certificatePath = "certs/IotRoom2.cert.pem"  privateKeyPath = "certs/IotRoom2.private.key"  class AwsIot:  def \_\_init\_\_(self, Name, Queue):  self.my\_rpi = AWSIoTMQTTClient(Name)  self.my\_rpi.configureEndpoint(host, 8883)  self.my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)  self.my\_rpi.configureOfflinePublishQueueing(Queue) # Infinite offline Publish queueing  self.my\_rpi.configureDrainingFrequency(10) # Draining: 2 Hz  self.my\_rpi.configureConnectDisconnectTimeout(10) # 10 sec  self.my\_rpi.configureMQTTOperationTimeout(1) # 5 sec  self.my\_rpi.connect() |

## AwsS3.py

AwsS3.py is a python class script that contains AwsS3 class, contains basic constructor to start up aws s3 as well as upload method

|  | Task |
| --- | --- |
|  | Create a python script AwsS3.py with the code below   |  | | --- | | sudo nano ~/CA2/AwsS3.py | |
|  | import boto3  import botocore  import sys  import os  sys.path.append(os.path.abspath('..'))  from Shared.Configs import Config  class AwsS3:  def \_\_init\_\_(self):  self.s3 = boto3.resource('s3',  aws\_access\_key\_id=Config.aws\_S3\_access\_key,  aws\_secret\_access\_key=Config.aws\_S3\_secret\_key)  def Upload(self,file\_path, file\_name):  self.s3.Object(Config.aws\_S3\_bucket, file\_name).put(Body=open(file\_path, 'rb'))  print("File uploaded") |

## LCDScroller.py

LCDScroller.py is a python class script that contains LCDScroller class, help to scroll text horizontially if text is too long on LCD

|  | Task |
| --- | --- |
|  | Create a python script LCDScroller.py with the code below   |  | | --- | | sudo nano ~/CA2/ LCDScroller.py | |
|  | from rpi\_lcd import LCD  import time  class LCDScroller:  def \_\_init\_\_(self):  self.lcd = LCD(rows = 2)  self.lcd.clear()  self.internalclear()  self.update\_time = time.time()    def text(self, text, line):  print(text)  self.lcd.text(text,line)  line -= 1  self.index[line] = 0  self.texts[line] = text    def update(self):  if time.time() - self.update\_time < 0.5:  return  self.update\_time = time.time();  line = 0  for txt in self.texts:  text\_len = len(txt)  if text\_len > 16:  self.index[line] += 1  new\_text = txt[self.index[line] : self.index[line] + 16]  if self.index[line] + 17 > text\_len:  if len(new\_text) <= 0:  self.index[line] = 0  new\_text = txt[0 : 16 - len(new\_text)]  else:  new\_text += ' ' + txt[0 : 16 - len(new\_text)]  self.lcd.text(new\_text,line + 1)  line += 1    def clear(self):  self.lcd.clear()  self.texts  self.internalclear()    def internalclear(self):  self.texts = ['','']  self.index = [0,0] |

## MFRC522.py

MFRC522.py is a python class script that contains MFRC522 class, it is modified to use with NFC reader class

|  | Task |
| --- | --- |
|  | Create a python script MFRC522.py with the code below   |  | | --- | | sudo nano ~/CA2/ MFRC522.py | |
|  | #!/usr/bin/env python  # -\*- coding: utf8 -\*-  #  # Copyright 2014,2018 Mario Gomez <mario.gomez@teubi.co>  #  # This file is part of MFRC522-Python  # MFRC522-Python is a simple Python implementation for  # the MFRC522 NFC Card Reader for the Raspberry Pi.  #  # MFRC522-Python is free software: you can redistribute it and/or modify  # it under the terms of the GNU Lesser General Public License as published by  # the Free Software Foundation, either version 3 of the License, or  # (at your option) any later version.  #  # MFRC522-Python is distributed in the hope that it will be useful,  # but WITHOUT ANY WARRANTY; without even the implied warranty of  # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  # GNU Lesser General Public License for more details.  #  # You should have received a copy of the GNU Lesser General Public License  # along with MFRC522-Python. If not, see <http://www.gnu.org/licenses/>.  #  import RPi.GPIO as GPIO  import spi  import signal  import time  class MFRC522:  NRSTPD = 22  MAX\_LEN = 16  PCD\_IDLE = 0x00  PCD\_AUTHENT = 0x0E  PCD\_RECEIVE = 0x08  PCD\_TRANSMIT = 0x04  PCD\_TRANSCEIVE = 0x0C  PCD\_RESETPHASE = 0x0F  PCD\_CALCCRC = 0x03  PICC\_REQIDL = 0x26  PICC\_REQALL = 0x52  PICC\_ANTICOLL = 0x93  PICC\_SElECTTAG = 0x93  PICC\_AUTHENT1A = 0x60  PICC\_AUTHENT1B = 0x61  PICC\_READ = 0x30  PICC\_WRITE = 0xA0  PICC\_DECREMENT = 0xC0  PICC\_INCREMENT = 0xC1  PICC\_RESTORE = 0xC2  PICC\_TRANSFER = 0xB0  PICC\_HALT = 0x50  MI\_OK = 0  MI\_NOTAGERR = 1  MI\_ERR = 2  Reserved00 = 0x00  CommandReg = 0x01  CommIEnReg = 0x02  DivlEnReg = 0x03  CommIrqReg = 0x04  DivIrqReg = 0x05  ErrorReg = 0x06  Status1Reg = 0x07  Status2Reg = 0x08  FIFODataReg = 0x09  FIFOLevelReg = 0x0A  WaterLevelReg = 0x0B  ControlReg = 0x0C  BitFramingReg = 0x0D  CollReg = 0x0E  Reserved01 = 0x0F  Reserved10 = 0x10  ModeReg = 0x11  TxModeReg = 0x12  RxModeReg = 0x13  TxControlReg = 0x14  TxAutoReg = 0x15  TxSelReg = 0x16  RxSelReg = 0x17  RxThresholdReg = 0x18  DemodReg = 0x19  Reserved11 = 0x1A  Reserved12 = 0x1B  MifareReg = 0x1C  Reserved13 = 0x1D  Reserved14 = 0x1E  SerialSpeedReg = 0x1F  Reserved20 = 0x20  CRCResultRegM = 0x21  CRCResultRegL = 0x22  Reserved21 = 0x23  ModWidthReg = 0x24  Reserved22 = 0x25  RFCfgReg = 0x26  GsNReg = 0x27  CWGsPReg = 0x28  ModGsPReg = 0x29  TModeReg = 0x2A  TPrescalerReg = 0x2B  TReloadRegH = 0x2C  TReloadRegL = 0x2D  TCounterValueRegH = 0x2E  TCounterValueRegL = 0x2F  Reserved30 = 0x30  TestSel1Reg = 0x31  TestSel2Reg = 0x32  TestPinEnReg = 0x33  TestPinValueReg = 0x34  TestBusReg = 0x35  AutoTestReg = 0x36  VersionReg = 0x37  AnalogTestReg = 0x38  TestDAC1Reg = 0x39  TestDAC2Reg = 0x3A  TestADCReg = 0x3B  Reserved31 = 0x3C  Reserved32 = 0x3D  Reserved33 = 0x3E  Reserved34 = 0x3F  serNum = []  def \_\_init\_\_(self, dev='/dev/spidev0.1', spd=1000000):  spi.openSPI(device=dev,speed=spd)  GPIO.setmode(GPIO.BCM)  GPIO.setup(25, GPIO.OUT)  GPIO.output(25, 1)  self.MFRC522\_Init()    def MFRC522\_Reset(self):  self.Write\_MFRC522(self.CommandReg, self.PCD\_RESETPHASE)  def Write\_MFRC522(self, addr, val):  spi.transfer(((addr<<1)&0x7E,val))  def Read\_MFRC522(self, addr):  val = spi.transfer((((addr<<1)&0x7E) | 0x80,0))  return val[1]  def SetBitMask(self, reg, mask):  tmp = self.Read\_MFRC522(reg)  self.Write\_MFRC522(reg, tmp | mask)  def ClearBitMask(self, reg, mask):  tmp = self.Read\_MFRC522(reg);  self.Write\_MFRC522(reg, tmp & (~mask))  def AntennaOn(self):  temp = self.Read\_MFRC522(self.TxControlReg)  if(~(temp & 0x03)):  self.SetBitMask(self.TxControlReg, 0x03)  def AntennaOff(self):  self.ClearBitMask(self.TxControlReg, 0x03)  def MFRC522\_ToCard(self,command,sendData):  backData = []  backLen = 0  status = self.MI\_ERR  irqEn = 0x00  waitIRq = 0x00  lastBits = None  n = 0  i = 0  if command == self.PCD\_AUTHENT:  irqEn = 0x12  waitIRq = 0x10  if command == self.PCD\_TRANSCEIVE:  irqEn = 0x77  waitIRq = 0x30  self.Write\_MFRC522(self.CommIEnReg, irqEn|0x80)  self.ClearBitMask(self.CommIrqReg, 0x80)  self.SetBitMask(self.FIFOLevelReg, 0x80)  self.Write\_MFRC522(self.CommandReg, self.PCD\_IDLE);  while(i<len(sendData)):  self.Write\_MFRC522(self.FIFODataReg, sendData[i])  i = i+1  self.Write\_MFRC522(self.CommandReg, command)  if command == self.PCD\_TRANSCEIVE:  self.SetBitMask(self.BitFramingReg, 0x80)  i = 2000  while True:  n = self.Read\_MFRC522(self.CommIrqReg)  i = i - 1  if ~((i!=0) and ~(n&0x01) and ~(n&waitIRq)):  break  self.ClearBitMask(self.BitFramingReg, 0x80)  if i != 0:  if (self.Read\_MFRC522(self.ErrorReg) & 0x1B)==0x00:  status = self.MI\_OK  if n & irqEn & 0x01:  status = self.MI\_NOTAGERR  if command == self.PCD\_TRANSCEIVE:  n = self.Read\_MFRC522(self.FIFOLevelReg)  lastBits = self.Read\_MFRC522(self.ControlReg) & 0x07  if lastBits != 0:  backLen = (n-1)\*8 + lastBits  else:  backLen = n\*8  if n == 0:  n = 1  if n > self.MAX\_LEN:  n = self.MAX\_LEN  i = 0  while i<n:  backData.append(self.Read\_MFRC522(self.FIFODataReg))  i = i + 1;  else:  status = self.MI\_ERR  return (status,backData,backLen)  def MFRC522\_Request(self, reqMode):  status = None  backBits = None  TagType = []  self.Write\_MFRC522(self.BitFramingReg, 0x07)  TagType.append(reqMode);  (status,backData,backBits) = self.MFRC522\_ToCard(self.PCD\_TRANSCEIVE, TagType)  if ((status != self.MI\_OK) | (backBits != 0x10)):  status = self.MI\_ERR  return (status,backBits)  def MFRC522\_Anticoll(self):  backData = []  serNumCheck = 0  serNum = []  self.Write\_MFRC522(self.BitFramingReg, 0x00)  serNum.append(self.PICC\_ANTICOLL)  serNum.append(0x20)  (status,backData,backBits) = self.MFRC522\_ToCard(self.PCD\_TRANSCEIVE,serNum)  if(status == self.MI\_OK):  i = 0  if len(backData)==5:  while i<4:  serNumCheck = serNumCheck ^ backData[i]  i = i + 1  if serNumCheck != backData[i]:  status = self.MI\_ERR  else:  status = self.MI\_ERR  return (status,backData)  def CalulateCRC(self, pIndata):  self.ClearBitMask(self.DivIrqReg, 0x04)  self.SetBitMask(self.FIFOLevelReg, 0x80);  i = 0  while i<len(pIndata):  self.Write\_MFRC522(self.FIFODataReg, pIndata[i])  i = i + 1  self.Write\_MFRC522(self.CommandReg, self.PCD\_CALCCRC)  i = 0xFF  while True:  n = self.Read\_MFRC522(self.DivIrqReg)  i = i - 1  if not ((i != 0) and not (n&0x04)):  break  pOutData = []  pOutData.append(self.Read\_MFRC522(self.CRCResultRegL))  pOutData.append(self.Read\_MFRC522(self.CRCResultRegM))  return pOutData  def MFRC522\_SelectTag(self, serNum):  backData = []  buf = []  buf.append(self.PICC\_SElECTTAG)  buf.append(0x70)  i = 0  while i<5:  buf.append(serNum[i])  i = i + 1  pOut = self.CalulateCRC(buf)  buf.append(pOut[0])  buf.append(pOut[1])  (status, backData, backLen) = self.MFRC522\_ToCard(self.PCD\_TRANSCEIVE, buf)  if (status == self.MI\_OK) and (backLen == 0x18):  print "Size: " + str(backData[0])  return backData[0]  else:  return 0  def MFRC522\_Auth(self, authMode, BlockAddr, Sectorkey, serNum):  buff = []  # First byte should be the authMode (A or B)  buff.append(authMode)  # Second byte is the trailerBlock (usually 7)  buff.append(BlockAddr)  # Now we need to append the authKey which usually is 6 bytes of 0xFF  i = 0  while(i < len(Sectorkey)):  buff.append(Sectorkey[i])  i = i + 1  i = 0  # Next we append the first 4 bytes of the UID  while(i < 4):  buff.append(serNum[i])  i = i +1  # Now we start the authentication itself  (status, backData, backLen) = self.MFRC522\_ToCard(self.PCD\_AUTHENT,buff)  # Check if an error occurred  if not(status == self.MI\_OK):  print "AUTH ERROR!!"  if not (self.Read\_MFRC522(self.Status2Reg) & 0x08) != 0:  print "AUTH ERROR(status2reg & 0x08) != 0"  # Return the status  return status  def MFRC522\_StopCrypto1(self):  self.ClearBitMask(self.Status2Reg, 0x08)  def MFRC522\_Read(self, blockAddr):  recvData = []  recvData.append(self.PICC\_READ)  recvData.append(blockAddr)  pOut = self.CalulateCRC(recvData)  recvData.append(pOut[0])  recvData.append(pOut[1])  (status, backData, backLen) = self.MFRC522\_ToCard(self.PCD\_TRANSCEIVE, recvData)  if not(status == self.MI\_OK):  print "Error while reading!"  i = 0  if len(backData) == 16:  print "Sector "+str(blockAddr)+" "+str(backData)  def MFRC522\_Write(self, blockAddr, writeData):  buff = []  buff.append(self.PICC\_WRITE)  buff.append(blockAddr)  crc = self.CalulateCRC(buff)  buff.append(crc[0])  buff.append(crc[1])  (status, backData, backLen) = self.MFRC522\_ToCard(self.PCD\_TRANSCEIVE, buff)  if not(status == self.MI\_OK) or not(backLen == 4) or not((backData[0] & 0x0F) == 0x0A):  status = self.MI\_ERR  print "%s backdata &0x0F == 0x0A %s" % (backLen, backData[0]&0x0F)  if status == self.MI\_OK:  i = 0  buf = []  while i < 16:  buf.append(writeData[i])  i = i + 1  crc = self.CalulateCRC(buf)  buf.append(crc[0])  buf.append(crc[1])  (status, backData, backLen) = self.MFRC522\_ToCard(self.PCD\_TRANSCEIVE,buf)  if not(status == self.MI\_OK) or not(backLen == 4) or not((backData[0] & 0x0F) == 0x0A):  print "Error while writing"  if status == self.MI\_OK:  print "Data written"  def MFRC522\_DumpClassic1K(self, key, uid):  i = 0  while i < 64:  status = self.MFRC522\_Auth(self.PICC\_AUTHENT1A, i, key, uid)  # Check if authenticated  if status == self.MI\_OK:  self.MFRC522\_Read(i)  else:  print "Authentication error"  i = i+1  def MFRC522\_Init(self):  GPIO.output(25, 1)  self.MFRC522\_Reset();  self.Write\_MFRC522(self.TModeReg, 0x8D)  self.Write\_MFRC522(self.TPrescalerReg, 0x3E)  self.Write\_MFRC522(self.TReloadRegL, 30)  self.Write\_MFRC522(self.TReloadRegH, 0)  self.Write\_MFRC522(self.TxAutoReg, 0x40)  self.Write\_MFRC522(self.ModeReg, 0x3D)  self.AntennaOn() |

## MotionDetection.py

MotionDetection.py is a python class script that contains MotionDetection class, a class to handle motion detection, take video upon motion

|  | Task |
| --- | --- |
|  | Create a python script MotionDetection.py with the code below   |  | | --- | | sudo nano ~/CA2/MotionDetection.py | |
|  | import os  from Shared.MotionEvent import MotionEvent  from gpiozero import MotionSensor  from gpiozero import Buzzer  from picamera import PiCamera  from AwsS3 import AwsS3  import time  class MotionDetection:  def \_\_init\_\_(self, BasePath, TelgramId, room):  print("Initializing Camera")  self.camera = PiCamera()  self.camera.resolution = (640, 480)  print("Initializing AWS")  self.s3 = AwsS3()  print("Initializing MotionSensor")  self.pir = MotionSensor(26, sample\_rate=1000,queue\_len=1,threshold=0.9)  self.pir.when\_motion = self.OnMotion  self.pir.when\_no\_motion = self.OnStopMotion  self.bz = Buzzer(21)  self.DisarmDetector = False  self.has\_detected\_motion = False  self.BasePath = BasePath  self.room = room  self.MotionCallback = None  if not os.path.isdir(BasePath):  os.makedirs(BasePath)  self.update\_time = time.time()  self.snapshot\_file\_path = str(room.Id) + '.png'  def SaveRecording(self):  self.camera.stop\_recording()  video\_file\_name = "video{}.mp4".format(self.video\_index)  vid\_path = self.BasePath + "/" + video\_file\_name  os.system("MP4Box -add " + self.BasePath + "/video0.h264 " + vid\_path)  s3\_path = 'videos/' + video\_file\_name  self.s3.Upload(vid\_path,s3\_path)  if self.MotionCallback:  self.MotionCallback(s3\_path)  self.has\_detected\_motion = False  self.bz.off()  def OnMotion(self):  if not self.DisarmDetector:#record  print("Motion detected")  if self.has\_detected\_motion:  self.SaveRecording()  self.has\_detected\_motion = True  self.video\_index = MotionEvent.GetMotionEventsCount()  File = 'videos/video{}.mp4'.format(self.video\_index)  ME = MotionEvent(RoomId = self.room.Id,FilePath = File)  ME.TryAdd()  self.camera.start\_recording(self.BasePath + '/video0.h264')  self.bz.on()  def OnStopMotion(self):  if not self.DisarmDetector and self.has\_detected\_motion:#stop recording  print("Motion stopped")  self.SaveRecording()  elif self.DisarmDetector and self.has\_detected\_motion:  print("Motion stopped")  self.SaveRecording()  def UpdateCameraSnapshot(self):  if time.time() - self.update\_time < 5:  return  self.update\_time = time.time()  my\_file = open(self.snapshot\_file\_path, 'wb')  try:  self.camera.capture(my\_file)  except:  print("encounter issue with camera")  self.camera.close()  self.camera = PiCamera()  self.camera.resolution = (640, 480)  my\_file.close()  self.s3.Upload(self.snapshot\_file\_path,'snapshot/' + self.snapshot\_file\_path)  def SetCallbackForMotion(self,Callback):  self.MotionCallback = Callback |

## SmartSecurity.py

SmartSecurity.py is a python script that contains core functionality of refreshing the LCD on certain event, motion detection, telegram bot, NFC card authorisation

|  | Task |
| --- | --- |
|  | Create a python script SmartSecurity.py with the code below   |  | | --- | | sudo nano ~/CA2/SmartSecurity.py | |
|  | import socket  from Shared.Room import Room  from LCDScroller import LCDScroller  from time import sleep  import RPi.GPIO as GPIO  import MFRC522  import signal  from Shared.User import User  from Shared.AccessRight import AccessRight  from Shared.AccessLog import AccessLog  from MotionDetection import MotionDetection  from Shared.Database import db  import thread  from AwsIot import AwsIot  import json  room = Room.TryGetRoom(socket.gethostname())  access\_right = None  access\_user = None  access\_log = None  mfrc522 = MFRC522.MFRC522()  lcd = LCDScroller()  if room == None:  print("Failed to get room")  exit()  awsiot = AwsIot('room' + str(room.Id),2)  awsiot\_topic = 'rooms/' + str(room.Id) + '/LightSwitch'  def OnMotionEvent(VideoPath):  global awsiot  message = {  'roomName': room.RoomName,  'FilePath': VideoPath  }  json\_data = json.dumps(message)  awsiot.my\_rpi.publish('rooms/MotionEvents',json\_data,1)  md = MotionDetection('/home/pi/labs/CA2/recordings','587634217:AAGfWPAPbd6GV91XL5g6gszCZec06WsiXJo',room)  md.SetCallbackForMotion(OnMotionEvent)  #---------------------------------------------------------------------------------  continue\_reading = True  def end\_read(signal,frame):  global continue\_reading  print "Ctrl+C captured, ending read."  db.Continue = False  print("stopping camera")  md.camera.close()  continue\_reading = False  GPIO.setmode(GPIO.BCM)  GPIO.cleanup(25)  signal.signal(signal.SIGINT, end\_read)  def snapshot\_thread():  global continue\_reading  global md  while continue\_reading:  sleep(5)  md.UpdateCameraSnapshot()  thread.start\_new\_thread(snapshot\_thread, ())  #---------------------------------------------------------------------------------  def ShowDefault():  lcd.text(room.RoomName, 1)  lcd.text('Tap card to access.',2)  def ShowWelcomeMsg():  lcd.clear()  lcd.text("Welcome {}".format(access\_right.user.Username),1)  #---------------------------------------------------------------------------------  ShowDefault()  while continue\_reading:  (status,TagType) = mfrc522.MFRC522\_Request(mfrc522.PICC\_REQIDL)  if status == mfrc522.MI\_OK:  (status,uid) = mfrc522.MFRC522\_Anticoll()  print("New card detected! UID of card is {}".format(uid))  if access\_right == None:  access\_user = User.TryGetUserByCardId(uid)  if access\_user != None:  access\_right = AccessRight(RoomId = room.Id,UserId = access\_user.Id)  if not access\_right.HasAccessRight():  access\_right = None  md.DisarmDetector = access\_right != None  lcd.clear()  if access\_right == None:  lcd.text("Card rejected!",1)  lcd.text("Please try again!",2)  if access\_user != None:  Invalid\_al = AccessLog(RoomId = room.Id,UserId = access\_user.Id,IsValid = False)  Invalid\_al.TryAdd()  sleep(3)  ShowDefault()  else:  access\_log = AccessLog(RoomId = room.Id,UserId = access\_user.Id,IsValid = True)  access\_log.TryAdd()  ShowWelcomeMsg()  awsiot.my\_rpi.publish(awsiot\_topic,"1",1)  sleep(3)  elif access\_user.CardId == str(uid):  access\_right = None  md.DisarmDetector = False  access\_log.UpdateExitTime()  access\_log = None  lcd.clear()  lcd.text("Thank you.",1)  lcd.text("Have a nice day.",2)  while not awsiot.my\_rpi.publish(awsiot\_topic,"0",1):  sleep(1)  sleep(3)  ShowDefault()  else:  lcd.clear()  lcd.text("Please use back old",1)  lcd.text("card to lock",2)  sleep(3)  ShowWelcomeMsg()  sleep(0.2)  if not continue\_reading:  exit()  lcd.update() |

## Updater.py

Updater.py is a python script that contains update functionality of refreshing the database on room information such as humidity & temperature readings

|  | Task |
| --- | --- |
|  | Create a python script Updater.py with the code below   |  | | --- | | sudo nano ~/CA2/Updater.py | |
|  | from Shared.Room import Room  from Shared.EnviroInfo import EnviroInfo  import socket  import Adafruit\_DHT  from time import sleep  from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient  import logging  import datetime as datetime  from Shared.helpers import data\_to\_json  import json  from gpiozero import MCP3008  from AwsIot import AwsIot  room = Room.TryGetRoom(socket.gethostname())  if room == None:  print("Failed to get room")  exit()  logger = logging.getLogger("AWSIoTPythonSDK.core")  logger.setLevel(logging.DEBUG)  streamHandler = logging.StreamHandler()  formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')  streamHandler.setFormatter(formatter)  logger.addHandler(streamHandler)  aws = AwsIot('room\_updater' + str(room.Id),-1)  adc = MCP3008(channel=0)  while True:  humidity, temperature = Adafruit\_DHT.read\_retry(11, 4)  print('Temp: {:.1f} C'.format(temperature))  print('Humidity: {:.1f}'.format(humidity))  light = adc.value \* 1024  print('light\_value: {}'.format(light))  en\_info = EnviroInfo(roomId = room.Id, temp = temperature,humidity = humidity,light\_value = light)  en\_info.WriteToDb()  message = {}  message["time"] = datetime.datetime.now()  message["temp"] = temperature  message["humid"] = humidity  aws.my\_rpi.publish("rooms/" + str(room.Id) + "/enviroData", json.dumps(data\_to\_json(message)), 0)  sleep(5) |

## UpdaterMatrixCreator.py

UpdaterMatrixCreator.py is a python script that contains mqtt functionality turning off/on the lights upon someone entering

|  | Task |
| --- | --- |
|  | Create a python script UpdaterMatrixCreator.py with the code below   |  | | --- | | sudo nano ~/CA2/UpdaterMatrixCreator.py | |
|  | ## Set Initial Variables ##  import os # Miscellaneous operating system interface  import zmq # Asynchronous messaging framework  import time # Time access and conversions  from random import randint # Random numbers  import sys # System-specific parameters and functions  from matrix\_io.proto.malos.v1 import driver\_pb2 # MATRIX Protocol Buffer driver library  from matrix\_io.proto.malos.v1 import io\_pb2 # MATRIX Protocol Buffer sensor library  from multiprocessing import Process, Manager, Value # Allow for multiple processes at once  from zmq.eventloop import ioloop, zmqstream# Asynchronous events through ZMQ  from utils import register\_error\_callback  from AwsIot import AwsIot  from Shared.Room import Room  import socket  import json  matrix\_ip = '127.0.0.1' # Local device ip  everloop\_port = 20021 # Driver Base port  led\_count = 0 # Amount of LEDs on MATRIX device  led\_switch = False  room = Room.TryGetRoomByLight(socket.gethostname())  if room == None:  print("Failed to get room")  exit()  awsiot = AwsIot('room\_light' + str(room.Id),0)  rgb\_color = None  def customCallback(client, userdata, message):  print("GOT CALLBACK")  print(message.payload)  global led\_switch  if message.payload == "1":  led\_switch = True  else:  led\_switch = False  def colourChangeCallback(client, userdata, message):  global rgb\_color  print(message.payload)  rgb\_color = json.loads(message.payload)  awsiot.my\_rpi.subscribe('rooms/' + str(room.Id) + '/LightSwitch', 1, customCallback)  awsiot.my\_rpi.subscribe('rooms/' + str(room.Id) + '/ColourChange', 1, colourChangeCallback)  def config\_socket(ledCount):  context = zmq.Context()  socket = context.socket(zmq.PUSH)  socket.connect('tcp://{0}:{1}'.format(matrix\_ip, everloop\_port))  while True:  driver\_config\_proto = driver\_pb2.DriverConfig()  image = []  light\_value = 0  global led\_switch  global rgb\_color  if led\_switch:  light\_value = 255  for led in range(ledCount):  ledValue = io\_pb2.LedValue()  if rgb\_color and led\_switch:  ledValue.blue = rgb\_color.get('b')  ledValue.red = rgb\_color.get('r')  ledValue.green = rgb\_color.get('g')  ledValue.white = 0  else:  ledValue.blue = light\_value  ledValue.red = light\_value  ledValue.green = light\_value  ledValue.white = 0  image.append(ledValue)  driver\_config\_proto.image.led.extend(image)  socket.send(driver\_config\_proto.SerializeToString())  time.sleep(0.05)  def ping\_socket():  context = zmq.Context()  ping\_socket = context.socket(zmq.PUSH)  ping\_socket.connect('tcp://{0}:{1}'.format(matrix\_ip, everloop\_port+1))  ping\_socket.send\_string('')  def everloop\_error\_callback(error):  print('{0}'.format(error))  ## DATA UPDATE PORT ##  def update\_socket():  context = zmq.Context()  socket = context.socket(zmq.SUB)  socket.connect('tcp://{0}:{1}'.format(matrix\_ip, everloop\_port+3))  socket.setsockopt(zmq.SUBSCRIBE, b'')  stream = zmqstream.ZMQStream(socket)  def updateLedCount(data):  global led\_count  led\_count = io\_pb2.LedValue().FromString(data[0]).green  print('{0} LEDs counted'.format(led\_count))  if led\_count > 0:  ioloop.IOLoop.instance().stop()  print('LED count obtained. Disconnecting from data publisher {0}'.format(everloop\_port+3))  stream.on\_recv(updateLedCount)  print('Connected to data publisher with port {0}'.format(everloop\_port+3))  ioloop.IOLoop.instance().start()  ## START PROCESSES ##  if \_\_name\_\_ == '\_\_main\_\_':  ioloop.install()  Process(target=register\_error\_callback, args=(everloop\_error\_callback, matrix\_ip, everloop\_port)).start()  ping\_socket()  update\_socket()  try:  config\_socket(led\_count)  except KeyboardInterrupt:  print(' quit') |

# 

# Section 8 Creating html files

## base.html

The base.html contains the basic for every single page of the web interface, we will be making use of jinja template inheritance to archive clean code

|  | Task |
| --- | --- |
|  | Create a html script base.html with the code below |
|  | ﻿<!DOCTYPE html>  <html lang="en" xmlns="http://www.w3.org/1999/xhtml">  <head>  {% block head %}  <script src="https://code.jquery.com/jquery-2.2.4.min.js"integrity="sha256-BbhdlvQf/xTY9gja0Dq3HiwQF8LaCRTXxZKRutelT44=" crossorigin="anonymous"></script>  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.css" crossorigin="anonymous">  <script src="https://cdnjs.cloudflare.com/ajax/libs/noty/3.1.4/noty.js" crossorigin="anonymous"></script>  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/3.7.0/animate.min.css">  <link rel="stylesheet" href="{{url\_for('static', filename='css/site.css')}}">  <title>{% block title %}{% endblock %}</title>  {% endblock %}  </head>  {% block body %}  <body>  <div id="content">  {% block content %}{% endblock %}  </div>  </body>  {% endblock %}  </html> |

## 404.html

The 404.html contains the webpage for handling any invalid web links when visiting the site.

|  | Task |
| --- | --- |
|  | Create a html script 404.html with the code below |
|  | {% extends "external\_base.html" %}  {% block title %}404{% endblock %}  {% block content %}  <div class="panel panel-default">  <h1>Error 404 not found</h1>  </div>  {{super()}} |

## external\_base.html

The external\_base.html contains the web layout for external pages that doesn’t require user to be signed in

|  | Task |
| --- | --- |
|  | Create a html script external\_base.html with the code below |
|  | ﻿{% extends "base.html" %}  {% block head %}  {{super()}}  <script src="http://cdn.jsdelivr.net/particles.js/2.0.0/particles.min.js"></script>  <link rel="stylesheet" href="{{url\_for('static', filename='css/login.css')}}">  {% endblock %}  {% block content %}  <div id="particles-js"></div>  <script src="{{url\_for('static', filename='js/login.js')}}"></script>  {% endblock %} |

## login.html

The login.html contains the web layout for user to login

|  | Task |
| --- | --- |
|  | Create a html script login.html with the code below |
|  | ﻿{% extends "external\_base.html" %}  {% block title %}Login{% endblock %}  {% block content %}  <div class="panel panel-default">  <h1>Smart Security Login</h1>  {% if error %}  <div class="alert alert-danger" role="alert">  {{error}}  </div>  {% endif %}  <form method="post" action="api/login">  <div class="input-group">  <span class="input-group-addon glyphicon glyphicon-user"></span>  <input type="text" class="form-control" placeholder="Username" name="username">  </div>  <div class="input-group">  <span class="input-group-addon glyphicon glyphicon-lock"></span>  <input type="password" class="form-control" placeholder="Password" name="password">  </div>  <button type="submit" class="btn btn-default">Sign In</button>  <button type="button" class="btn btn-default" onclick="window.location.href='/register'">Register</button>  </form>  </div>  {{super()}}  {% endblock %} |

## Internal\_base.html

The internal\_base.html contains the web layout for internal web page that require login

|  | Task |
| --- | --- |
|  | Create a html script internal\_base.html with the code below |
|  | ﻿{% extends "base.html" %}  {% block head %}  {{super()}}  <link rel="stylesheet" type="text/css" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">  <link rel="stylesheet" type="text/css" href="{{url\_for('static', filename='css/dashboard\_layout.css')}}">  <link rel="stylesheet" type="text/css" href="{{url\_for('static', filename='css/dashboard\_elements.css')}}">  <link rel="stylesheet" type="text/css" href="{{url\_for('static', filename='css/font.css')}}">  <script type="text/javascript" src="{{url\_for('static', filename='js/internal\_base.js')}}"></script>  {% endblock %}  {% block body %}  <body class="theme-red">  <div id="sticky-anchor"></div>  <div id="sticky">  <header class="navbar" id="header-navbar">  <div class="container">  {% if IsAdmin %}  <a href="/dashboard" style="padding-top:16px;" class="navbar-brand">  {% else %}  <a href="/publicdashboard" style="padding-top:16px;" class="navbar-brand">  {% endif %}  Smart Security  </a>  <div class="clearfix">  <button class="navbar-toggle" data-target=".navbar-ex1-collapse" data-toggle="collapse" type="button">  <span class="sr-only">Toggle navigation</span>  <span class="fa fa-bars"></span>  </button>  <div class="nav-no-collapse navbar-left pull-left hidden-xs">  <ul class="nav navbar-nav pull-left">  <li>  <a class="btn" id="make-small-nav">  <i class="fa fa-bars"></i>  </a>  </li>  </ul>  </div>  <div class="nav-no-collapse pull-right" id="header-nav">  <ul class="nav navbar-nav pull-right">  <li class="dropdown hidden-xs">  <a class="btn dropdown-toggle" data-toggle="dropdown">  <i class="fa fa-warning"></i>  <span class="count">0</span>  </a>  <ul class="dropdown-menu notifications-list">  <li class="pointer">  <div class="pointer-inner">  <div class="arrow"></div>  </div>  </li>  <li class="item-header">Notifications</li>  <li class="item">  <a>  <i class="fa fa-info"></i>  You have no pending notifications  </a>  </li>  </ul>  </li>  <li class="dropdown profile-dropdown">  <a href="#" class="dropdown-toggle" data-toggle="dropdown">  <img src="http://placehold.it/150x150" alt="" />  <span class="hidden-xs">{{ Username }}</span>  <b class="caret"></b>  </a>  <ul class="dropdown-menu">  <li>  <a href="/api/logout">  <i class="fa fa-power-off"></i>Logout  </a>  </li>  </ul>  </li>  <li class="hidden-xxs">  <a href="/api/logout" class="btn">  <i class="fa fa-power-off"></i>  </a>  </li>  </ul>  </div>  </div>  </div>  </header>  </div>  <div id="page-wrapper" class="container">  <div class="row">  <div id="nav-col">  <section id="col-left" class="col-left-nano">  <div id="col-left-inner" class="col-left-nano-content">  <div id="user-left-box" class="clearfix hidden-xs">  <img alt="" src="http://placehold.it/150x150" />  <div class="user-box">  <span class="name">Welcome<br />{{ Username }}</span>  <span class="status">  <i class="fa fa-circle"></i> Online  </span>  </div>  </div>  <div class="collapse navbar-collapse navbar-ex1-collapse" id="sidebar-nav">  <ul class="nav nav-pills nav-stacked">  <li class="{% if title == "Dashboard" %}active{% endif %}">  <a href="/dashboard">  <i class="fa fa-dashboard"></i>  <span>Dashboard</span>  </a>  </li>  {% if IsAdmin %}  <li class="{% if "Room" in title %}active{% endif %}">  <a href="#" class="dropdown-toggle">  <i class="fa fa-file-text-o"></i>  <span>Rooms</span>  <i class="fa fa-chevron-circle-right drop-icon"></i>  </a>  <ul class="submenu">  <li> <a href="/ManageRooms"> Manage Rooms </a> </li>  <li> <a href="/RoomAccessRights"> Room Access Rights </a> </li>  <li> <a href="/ManageApprovals"> Manage Access Requests </a> </li>  </ul>  </li>  {% endif %}  </ul>  </div>  </div>  </section>  </div>  <div id="content-wrapper">  <div class="row">  <div class="col-lg-12">  <div class="row">  <div class="col-lg-12">  <ol class="breadcrumb">  <li><a href="dashboard.php">Home</a></li>  <li class="active"><span>{{ title }}</span></li>  </ol>  <h1>{{ title }}</h1>  </div>  </div>  <div class="row">  {% block content %}{% endblock %}  </div>  </div>  </div>  </div>  </div>  </div>  </body>  {% endblock %} |

## dashboard.html

The dashboard.html contains the web layout of the web interface, that contains the graph and table about room access

|  | Task |
| --- | --- |
|  | Create a html script dashboard.html with the code below |
|  | ﻿<!DOCTYPE html>  {% extends "internal\_base.html" %}  {% set title = 'Dashboard' %}  {% block title %}{{ title }}{% endblock %}  {% block head %}  {{super()}}  <script src="https://cdnjs.cloudflare.com/ajax/libs/highcharts/6.2.0/highcharts.js"></script>  <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.22.2/moment.js"></script>  <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css">  <script type="text/javascript" src="https://cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>  <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.min.js"></script>  <script type="text/javascript" src="{{url\_for('static', filename='js/dashboard.js')}}"></script>  {% endblock %}  {% block content %}  <select id="RoomSelect" class="form-control" name="RoomId">  {% for room in rooms %}  <option value="{{room.Id}}">{{room.RoomName}}</option>  {% endfor %}  </select>  <div class="col-md-6 col-sm-12">  <h4>Current Temperature Recordings</h4>  <div id="current\_temp"></div>  </div>  <div class="col-md-6 col-sm-12">  <h4>Average Tempature Readings</h4>  <div id="avg\_temp\_history"></div>  </div>  </div>  <div class="row">  <div class="col-md-6 col-sm-12">  <h4>Valid Room Access History</h4>  <table id="VRATable" class="display">  <thead>  <tr>  <th>Room Name</th>  <th>User</th>  <th>Enter Time</th>  <th>Exit Time</th>  </tr>  </thead>  <tbody></tbody>  </table>  </div>  <div class="col-md-6 col-sm-12">  <h4>Invalid Room Access History</h4>  <table id="IVRATable" class="display">  <thead>  <tr>  <th>Room Name</th>  <th>User</th>  <th>Tap Time</th>  </tr>  </thead>  <tbody></tbody>  </table>  </div>  </div>  <div class="row">  <div class="col-md-6 col-sm-12">  <h4>Motion Detection History</h4>  <table id="MAHTable" class="display">  <thead>  <tr>  <th>Id</th>  <th>Time</th>  <th></th>  </tr>  </thead>  <tbody></tbody>  </table>  </div>  <div class="modal fade" id="VideoModal" tabindex="-1" role="dialog" aria-hidden="true">  <div class="modal-dialog">  <div class="modal-content">  <div class="modal-header">  <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>  <h4 class="modal-title">Motion Preview</h4>  </div>  <div class="modal-body">  <video id="VideoPreview" src="" type="video/mp4" controls="controls" width="560">  </video>  </div>  </div><!-- /.modal-content -->  </div><!-- /.modal-dialog -->  </div><!-- /.modal -->  {% endblock %} |

## manage\_rooms.html

The manage\_rooms.html contains the web layout of the web interface, that contains the room management, where you can add/remove/edit room information

|  | Task |
| --- | --- |
|  | Create a html script manage\_rooms.html with the code below |
|  | ﻿<!DOCTYPE html>  {% extends "internal\_base.html" %}  {% set title = 'Manage Rooms' %}  {% block title %}{{ title }}{% endblock %}  {% block head %}  {{super()}}  <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css">  <script type="text/javascript" src="https://cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>  <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.min.js"></script>  <script type="text/javascript" src="{{url\_for('static', filename='js/html5kellycolorpicker.min.js')}}"></script>  <script type="text/javascript" src="{{url\_for('protected', filename='lights.js')}}"></script>  <script type="text/javascript" src="{{url\_for('static', filename='js/manage\_rooms.js')}}"></script>  {% endblock %}  {% block content %}  <div class="col-md-12">  <a id="btnAdd" class="btn btn-primary pull-right">Add new room <i class="fa fa-plus" aria-hidden="true"></i></a>  <br />  <table id="roomsTable" class="display">  <thead>  <tr>  <th>Id</th>  <th>Room Name</th>  <th>Iot Device Name</th>  <th>Iot Light Device</th>  <th>Manage</th>  </tr>  </thead>  <tbody></tbody>  </table>  </div>  <div class="modal fade" id="fillModal" tabindex="-1" role="dialog" aria-hidden="true">  <div class="modal-dialog">  <div class="modal-content">  <div class="modal-header">  <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>  <h4 class="modal-title">Update Room Info</h4>  </div>  <div class="modal-body">  <form id="iotForm">  <div class="input-group">  <span class="input-group-addon" id="sizing-addon2">IoT Device Name</span>  <input id="IotId" type="text" class="form-control" placeholder="Iot Id" name="IotId" readonly>  </div>  <br />  <div class="input-group">  <span class="input-group-addon" id="sizing-addon2">Room name</span>  <input id="roomName" type="text" class="form-control" placeholder="Room Name" name="RoomName" pattern=".{3,}" required title="3 characters minimum">  </div>  <br />  <div class="input-group">  <span class="input-group-addon" id="sizing-addon2">IoT Device Name</span>  <input id="IotDeviceName" type="text" class="form-control" placeholder="IoT Device Name" name="IotDeviceName" pattern=".{3,}" required title="3 characters minimum">  </div>  <br />  <div class="input-group">  <span class="input-group-addon" id="sizing-addon2">IoT Light Device</span>  <input id="IoTLightDevice" type="text" class="form-control" placeholder="IoT Light Device" name="IoTLightDevice" pattern=".{3,}" required title="3 characters minimum">  </div>  </form>  </div>  <div class="modal-footer">  <button id="saveIot" type="button" class="btn btn-primary">Save changes</button>  </div>  </div><!-- /.modal-content -->  </div><!-- /.modal-dialog -->  </div><!-- /.modal -->  <div class="modal fade" id="confirmModal" tabindex="-1" role="dialog" aria-hidden="true">  <div class="modal-dialog">  <div class="modal-content">  <div class="modal-header">  <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>  <h4 class="modal-title">Confirmation</h4>  </div>  <div class="modal-body">  Are you sure you want to remove the following room?  <form id="iotRemoveForm">  <input id="IotRemoveId" type="hidden" class="form-control" name="IotId">  <div class="input-group">  <span class="input-group-addon" id="sizing-addon2">Room name</span>  <input id="roomNameR" type="text" class="form-control" placeholder="Room Name" readonly>  </div>  <br />  <div class="input-group">  <span class="input-group-addon" id="sizing-addon2">IoT Device Name</span>  <input id="IotDeviceNameR" type="text" class="form-control" placeholder="IoT Device Name" readonly>  </div>  <br />  <div class="input-group">  <span class="input-group-addon" id="sizing-addon2">IoT Light Device</span>  <input id="IoTLightDeviceR" type="text" class="form-control" placeholder="IoT Light Device" readonly>  </div>  </form>  </div>  <div class="modal-footer">  <button id="removeIoT" type="button" class="btn btn-primary">Remove</button>  </div>  </div><!-- /.modal-content -->  </div><!-- /.modal-dialog -->  </div><!-- /.modal -->  <div class="modal fade" id="colourModal" tabindex="-1" role="dialog" aria-hidden="true">  <div class="modal-dialog">  <div class="modal-content">  <div class="modal-header">  <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>  <h4 class="modal-title">Colour Selection</h4>  </div>  <div class="modal-body">  <input id="IotColourId" type="hidden" class="form-control" name="IotId">  <canvas id="picker"></canvas>  <br>  <div class="input-group">  <input id="color" value="#54aedb" class="form-control">  </div>  </div>  </div><!-- /.modal-content -->  </div><!-- /.modal-dialog -->  </div><!-- /.modal -->  {% endblock %} |

## room\_access\_rights.html

The room\_access\_rights.html contains the web layout of the web interface, that contains the access right management, where you can add/remove/edit card id that can access certain rooms

|  | Task |
| --- | --- |
|  | Create a html script room\_access\_rights.html with the code below |
|  | ﻿<!DOCTYPE html>  {% extends "internal\_base.html" %}  {% set title = 'Room Access Rights' %}  {% block title %}{{ title }}{% endblock %}  {% block head %}  {{super()}}  <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css">  <script type="text/javascript" src="https://cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>  <script type="text/javascript" src="{{url\_for('static', filename='js/room\_access\_rights.js')}}"></script>  {% endblock %}  {% block content %}  <div class="col-md-12">  <a id="btnAdd" class="btn btn-primary pull-right">Add New Access Rights <i class="fa fa-plus" aria-hidden="true"></i></a>  <br />  <table id="AccessRightsTable" class="display">  <thead>  <tr>  <th>Id</th>  <th>User</th>  <th>Room Name</th>  <th>Manage</th>  </tr>  </thead>  <tbody></tbody>  </table>  </div>  <div class="modal fade" id="fillModal" tabindex="-1" role="dialog" aria-hidden="true">  <div class="modal-dialog">  <div class="modal-content">  <div class="modal-header">  <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>  <h4 class="modal-title">Update Access Rights</h4>  </div>  <div class="modal-body">  <form id="accessRightForm">  <div class="input-group">  <span class="input-group-addon" id="sizing-addon2">Id</span>  <input id="Id" type="text" class="form-control" placeholder="Id" name="Id" readonly>  </div>  <br />  <div class="input-group">  <span class="input-group-addon" id="sizing-addon2">User</span>  <select id="UserSelect" class="form-control" name="UserId">  {% for user in users %}  <option value="{{user.Id}}">{{user.Username}}</option>  {% endfor %}  </select>  </div>  <br />  <div class="input-group">  <span class="input-group-addon" id="sizing-addon2">Room</span>  <select id="RoomSelect" class="form-control" name="RoomId">  {% for room in rooms %}  <option value="{{room.Id}}">{{room.RoomName}}</option>  {% endfor %}  </select>  </div>  </form>  </div>  <div class="modal-footer">  <button id="saveAccessRight" type="button" class="btn btn-primary">Save changes</button>  </div>  </div><!-- /.modal-content -->  </div><!-- /.modal-dialog -->  </div><!-- /.modal -->  <div class="modal fade" id="confirmModal" tabindex="-1" role="dialog" aria-hidden="true">  <div class="modal-dialog">  <div class="modal-content">  <div class="modal-header">  <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>  <h4 class="modal-title">Confirmation</h4>  </div>  <div class="modal-body">  Are you sure you want to remove the following access right?  <form id="ACRemoveForm">  <input id="ACRemoveId" type="hidden" class="form-control" name="Id">  <div class="input-group">  <span class="input-group-addon" id="sizing-addon2">User</span>  <input id="UsernameR" type="text" class="form-control" placeholder="Username" readonly>  </div>  <br />  <div class="input-group">  <span class="input-group-addon" id="sizing-addon2">IoT Device Name</span>  <input id="RoomNameR" type="text" class="form-control" placeholder="Room Name" readonly>  </div>  </form>  </div>  <div class="modal-footer">  <button id="removeAccessRight" type="button" class="btn btn-primary">Remove</button>  </div>  </div><!-- /.modal-content -->  </div><!-- /.modal-dialog -->  </div><!-- /.modal -->  {% endblock %} |

## register.html

Register.html contains the web layout of register page and allows users to register for an account through a form

|  | Task |
| --- | --- |
|  | Create a html script register.html with the code below |
|  | {% extends "external\_base.html" %}  {% block title %}Register{% endblock %}  {% block content %}  <div class="panel panel-default">  <h1>Smart Security Register</h1>  {% if error %}  <div class="alert alert-danger" role="alert">  {{error}}  </div>  {% endif %}  <form method="post" action="api/register">  <div class="input-group">  <span class="input-group-addon glyphicon glyphicon-user"></span>  <input type="text" class="form-control" placeholder="Username" name="username">  </div>  <div class="input-group">  <span class="input-group-addon glyphicon glyphicon-lock"></span>  <input type="password" class="form-control" placeholder="Password" name="password">  </div>  <div class="input-group">  <span class="input-group-addon glyphicon glyphicon-lock"></span>  <input type="password" class="form-control" placeholder="Retype Password" name="retypepassword">  </div>  <button type="submit" class="btn btn-default">Register!</button>  </form>  </div>  {{super()}}  {% endblock %} |

## publicDashboard.html

publicDashboard.html contains the web layout of the dashboard for public users

|  | Task |
| --- | --- |
|  | Create a html script publicDashboard.html with the code below |
|  | <!DOCTYPE html>  {% extends "internal\_base.html" %}  {% set title = 'Public Dashboard' %}  {% block title %}{{ title }}{% endblock %}  {% block head %}  {{super()}}  <script src="https://cdnjs.cloudflare.com/ajax/libs/highcharts/6.2.0/highcharts.js"></script>  <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.22.2/moment.js"></script>  <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css">  <script type="text/javascript" src="https://cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>  <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.min.js"></script>  <script type="text/javascript" src="{{url\_for('static', filename='js/publicdashboard.js')}}"></script>  <link rel="stylesheet" href="{{url\_for('static', filename='css/publicdashboard.css')}}">  {% endblock %}  {% block content %}  {% if user.CardId is none %}  <div id="cardIdWrapper">  <div id="warning" class="alert alert-danger" role="alert">  Warning: Set your Card ID, or you cannot access any rooms  </div>  <form id="cardIdForm" class="form-inline">  <div class="form-group">  <h4>Set Your Card ID Now</h4>  <label class="sr-only" for="cardID">Card ID</label>  <input type="text" class="form-control" id="cardID" name="cardID" aria-describedby="emailHelp" placeholder="Enter your Card ID">  <button id="cardIDbtn" type="submit" class="btn btn-primary">Submit</button>  </div>  </form>  </div>  {% endif %}  <div class="row wrapper">  <h1>Room Live Cam </h1>  {% for room in rooms %}  <div class="col-sm-12 col-md-4">  <div class="thumbnail">  <a href="/w3images/lights.jpg" class="room">  <img class="liveImage" alt="{{room.RoomName}} Image" style="width:100%">  <div class="caption">  <p>{{room.RoomName}}</p>  <p class="time"></p>  <p class="light"></p>  <p class="humidity"></p>  <p class="temp"></p>  </div>  </a>  </div>  </div>  {% endfor %}  </div>  <div class="row wrapper">  <div class="col-sm-12">  <h1>Your Access Rights</h1>  <table id="accessRightsTable" class="display">  <thead>  <tr>  <th>Room Name</th>  </tr>  </thead>  </table>  </div>  </div>  <div class="row wrapper">  <div class="col-sm-12">  <h1>Request Access Rights <button id="requestBtn" type="button" class="btn btn-primary" data-toggle="modal" data-target="#myModal">New Request</button></h1>  <table id="requestTable" class="display">  <thead>  <tr>  <th>Room Name</th>  <th>Date Requested</th>  <th>Approval Status</th>  </tr>  </thead>  </table>  </div>  </div>  <!-- Modal -->  <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">  <div class="modal-dialog" role="document">  <div class="modal-content">  <div class="modal-header">  <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>  <h4 class="modal-title" id="myModalLabel">Create New Request</h4>  <!-- <h5 class="orange">Note: You cannot request a room you already requested</h5> -->  </div>  <div class="modal-body">  <form id="requestForm">  <div class="form-group">  <label for="room">Room </label>  <select class="form-control" id="room" name="room">  {% for room in rooms %}  <option value={{room.Id}}>{{room.RoomName}}</option>  {% endfor %}  </select>  <!-- <input type="text" class="form-control" id="room" name="room" placeholder="Select Room"> -->  <div class="text-right spacingtop">  <button id="requestFormBtn" type="submit" class="btn btn-primary right">Request</button>  </div>  </div>  </form>  </div>  <!-- <div class="modal-footer">  <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>  <button type="button" class="btn btn-primary">Save changes</button>  </div> -->  </div>  </div>  </div>  {% endblock %} |

## manage\_approvals.html

manage\_approvals.html contains the web layout for admin side manage access right requests of public users

|  | Task |
| --- | --- |
|  | Create a html script manage\_approvalsx`.html with the code below |
|  | <!DOCTYPE html>  {% extends "internal\_base.html" %}  {% set title = 'Manage Access Rights Requests' %}  {% block title %}{{ title }}{% endblock %}  {% block head %}  {{super()}}  <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css">  <script type="text/javascript" src="https://cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>  <script type="text/javascript" src="{{url\_for('static', filename='js/manage\_approvals.js')}}"></script>  {% endblock %}  {% block content %}  <ul class="nav nav-tabs">  <li class="active"><a data-toggle="tab" href="#pending">Pending Requests</a></li>  <li><a data-toggle="tab" href="#history">Resolved Requests</a></li>  </ul>  <div class="tab-content">  <div id="pending" class="tab-pane fade in active">  <div class="col-md-12 spacingtop">  <table id="pendingTable" class="display">  <thead>  <tr>  <th>Username</th>  <th>Room Name</th>  <th>Date Requested</th>  <th>Status</th>  <th>User ID</th>  </tr>  </thead>  <tbody></tbody>  </table>  </div>  </div>  <div id="history" class="tab-pane fade">  <div class="col-md-12 spacingtop">  <table id="resolvedTable" class="display">  <thead>  <tr>  <th>Username</th>  <th>Room Name</th>  <th>Date Requested</th>  <th>Status</th>  </tr>  </thead>  <tbody></tbody>  </table>  </div>  </div>  </div>  {% endblock %} |

# Section 9 Creating js files

## dashboard.js

The dashboard.js contains the code for drawing varies graphs & table for dashboard page

|  | Task |
| --- | --- |
|  | Create a js script dashboard.js with the code below |
|  | var DefaultoLanguage = {  'sEmptyTable': 'No Data',  'sLoadingRecords': 'Loading. Please wait...',  'sLengthMenu': '<div class="row item\_to\_show"><div class="col-md-7 col-sm-12">Items to show:</div>' +  '<div class="col-md-5 col-sm-12"><select class="form-control">' +  '<option value="5">5</option>' +  '<option value="10">10</option>' +  '<option value="20" selected="selected">20</option>' +  '<option value="25">25</option>' +  '<option value="50">50</option>' +  '<option value="100">100</option>' +  '<option value="250">250</option>' +  '</select></div></div>',  "sSearch": ''  };  var current\_temp = []  var wss\_client = null;  var RoomTempGraph = null;  var VRATable = null;  var IVRATable = null;  var MAHTable = null;  function DrawRoomTempsGraph() {  if (RoomTempGraph != null) {  RoomTempGraph.series[0].update({  name: $('#RoomSelect option:selected').text(),  data: current\_temp  }, true);  return;  }  RoomTempGraph = Highcharts.chart('current\_temp', {  chart: {  type: 'spline'  },  title: {  text: 'Current temperature readings'  },  xAxis: {  title: {  text: 'time'  },  type: "category",  labels: {  formatter: function() {  return new moment(this.value).format("h:mm:ss a");  },  }  },  yAxis: {  title: {  text: 'Temperature'  }  },  legend: {  align: 'center',  verticalAlign: 'bottom',  x: 0,  y: 0  },  series: [{  name: $('#RoomSelect option:selected').text(),  data:current\_temp  }],  responsive: {  rules: [{  condition: {  maxWidth: 500  },  chartOptions: {  legend: {  layout: 'horizontal',  align: 'center',  verticalAlign: 'bottom'  }  }  }]  }  });  }  function GrabAvgRoomTemps() {  $.ajax({  type: "GET",  url: "/api/getAvgRoomTemps/" + $('#RoomSelect').val(),  dataType: "json",  success: function(data) {  Highcharts.chart('avg\_temp\_history', {  chart: {  type: 'spline'  },  title: {  text: 'Average temperature readings'  },  xAxis: {  title: {  text: 'time'  },  type: "category",  labels: {  formatter: function() {  var time = new moment(this.value);  return time.format("D MMM YYYY");  },  }  },  yAxis: {  title: {  text: 'Temperature'  }  },  legend: {  align: 'center',  verticalAlign: 'bottom',  x: 0,  y: 0  },  series: [{  name: data.RoomName,  data: data.data  }],  responsive: {  rules: [{  condition: {  maxWidth: 500  },  chartOptions: {  legend: {  layout: 'horizontal',  align: 'center',  verticalAlign: 'bottom'  }  }  }]  }  });  }  });  }  function onDateTimeColumn(CurDateTime, data) {  var time = new moment(data, "ddd, D MMM YYYY HH:mm:ss zz");  if (CurDateTime.diff(time, 'hours') < 24)  return time.fromNow();  return time.format("ddd, D MMM YYYY HH:mm:ss");  }  function GetValidAccessLogs() {  if (VRATable != null){  VRATable.ajax.url('/api/getValidAccessLog/' + $('#RoomSelect').val()).load();  return;  }  var CurDateTime = new moment();  VRATable = $('#VRATable').DataTable({  'oLanguage': DefaultoLanguage,  'iDisplayLength': 5,  'sPaginationType': 'full\_numbers',  'aoColumns': [{  'bSearchable': true,  'mData': 'RoomName'  },  {  'bSearchable': true,  'mData': 'Username'  },  {  'bSearchable': true,  'mData': 'Time',  'mRender': function(data, type, full) {  return onDateTimeColumn(CurDateTime, data);  }  },  {  'bSearchable': true,  'mData': 'Exit\_time',  'mRender': function(data, type, full) {  return onDateTimeColumn(CurDateTime, data);  }  }  ],  ajax: {  url: '/api/getValidAccessLog/' + $('#RoomSelect').val(),  dataSrc: ''  }  })  $("#VRATable\_filter label input").addClass("form-control").attr("placeholder", "Search...");  }  function GetInvalidAccessLogs() {  if (IVRATable != null){  IVRATable.ajax.url('/api/getInvalidAccessLog/' + $('#RoomSelect').val()).load();  return;  }  var CurDateTime = new moment();  IVRATable = $('#IVRATable').DataTable({  'oLanguage': DefaultoLanguage,  'iDisplayLength': 5,  'sPaginationType': 'full\_numbers',  'aoColumns': [{  'bSearchable': true,  'mData': 'RoomName'  },  {  'bSearchable': true,  'mData': 'Username'  },  {  'bSearchable': true,  'mData': 'Time',  'mRender': function(data, type, full) {  return onDateTimeColumn(CurDateTime, data);  }  }  ],  ajax: {  url: '/api/getInvalidAccessLog/' + $('#RoomSelect').val(),  dataSrc: ''  }  })  $("#IVRATable\_filter label input").addClass("form-control").attr("placeholder", "Search...");  }  function GetMotionDetectedHistory() {  if (MAHTable != null){  MAHTable.ajax.url('/api/getMotionEvents/' + $('#RoomSelect').val()).load();  return;  }  MAHTable = $('#MAHTable').DataTable({  'oLanguage': DefaultoLanguage,  'iDisplayLength': 5,  'sPaginationType': 'full\_numbers',  'aoColumns': [{  'bSearchable': true,  'mData': 'Id'  },  {  'bSearchable': true,  'mData': 'Time'  },  {  'bSearchable': false,  'bSortable': false,  'mData': 'FilePath',  'mRender': function(data, type, full) {  return '<a class="btn btn-primary btnView" href="https://s3-us-west-2.amazonaws.com/iotsmartroom/' + data + '">View <i class="fa fa-file-video-o" aria-hidden="true"></i></a>';  }  }  ],  "order": [  [2, "desc"]  ],  ajax: {  url: '/api/getMotionEvents/' + $('#RoomSelect').val(),  dataSrc: ''  }  }).on('draw', function() {  $(".btnView").click(function(e) {  var link = $(this).attr("href");  $('#VideoPreview').attr('src', link)  $('#VideoModal').modal('show');  e.preventDefault()  });  });  $("#MAHTable\_filter label input").addClass("form-control").attr("placeholder", "Search...");  }  function InitializeWSS() {  DrawRoomTempsGraph()  $.ajax({  type: "GET",  url: "/api/getAwsIotWSS",  success: function(data) {  var clientId = Math.random().toString(36).substring(7);  wss\_client = new Paho.MQTT.Client(data, clientId);  var connectOptions = {  useSSL: true,  timeout: 3,  mqttVersion: 4,  onSuccess: function subscribe() {  wss\_client.subscribe("rooms/" + $('#RoomSelect').val() + "/enviroData");  console.log("subscribed");  }  };  wss\_client.connect(connectOptions);  wss\_client.onMessageArrived = function onMessage(message) {  var status = JSON.parse(message.payloadString);  console.log(status);  current\_temp.push([status.time,status.temp])  DrawRoomTempsGraph()  }  wss\_client.onConnectionLost = function(e) {  console.log(e)  };  }  });  }  $(document).ready(function() {  GetValidAccessLogs();  GetInvalidAccessLogs();  GetMotionDetectedHistory();  GrabAvgRoomTemps();  InitializeWSS();  $roomSelect = $('#RoomSelect');  $roomSelect.data("prev",$roomSelect.val());  $roomSelect.change(function(){  GetValidAccessLogs();  GetInvalidAccessLogs();  GetMotionDetectedHistory();  GrabAvgRoomTemps();  current\_temp = []  if (wss\_client != null){  wss\_client.subscribe("rooms/" + $(this).val() + "/enviroData");  wss\_client.unsubscribe("rooms/" + $roomSelect.data("prev") + "/enviroData");  }  $(this).data("prev",$(this).val());  });  }); |

## login.js

The login.js contains the code for particle js

|  | Task |
| --- | --- |
|  | Create a js script login.js with the code below |
|  | particlesJS("particles-js", {  particles: {  number: { value: 80, density: { enable: true, value\_area: 800 } },  color: { value: "#ffffff" },  shape: {  type: "circle",  stroke: { width: 0, color: "#000000" },  polygon: { nb\_sides: 5 },  image: { src: "img/github.svg", width: 100, height: 100 }  },  opacity: {  value: 0.5,  random: false,  anim: { enable: false, speed: 1, opacity\_min: 0.1, sync: false }  },  size: {  value: 3,  random: true,  anim: { enable: false, speed: 40, size\_min: 0.1, sync: false }  },  line\_linked: {  enable: true,  distance: 150,  color: "#ffffff",  opacity: 0.4,  width: 1  },  move: {  enable: true,  speed: 2,  direction: "none",  random: false,  straight: false,  out\_mode: "out",  bounce: false,  attract: { enable: false, rotateX: 600, rotateY: 1200 }  }  },  interactivity: {  detect\_on: "canvas",  events: {  onhover: { enable: false, mode: "repulse" },  onclick: { enable: false, mode: "push" },  resize: true  },  modes: {  grab: { distance: 400, line\_linked: { opacity: 1 } },  bubble: { distance: 400, size: 40, duration: 2, opacity: 8, speed: 3 },  repulse: { distance: 200, duration: 0.4 },  push: { particles\_nb: 4 },  remove: { particles\_nb: 2 }  }  },  retina\_detect: true  }); |

## manage\_rooms.js

The manage\_rooms.js contains the code for manage rooms

|  | Task |
| --- | --- |
|  | Create a js script manage\_rooms.js with the code below |
|  | $(document).ready( function () {  var Picker = new KellyColorPicker({  place : 'picker',  size : 150,  input : 'color',  }).addUserEvent("change", function(self){  var id = $('#IotColourId').val();  console.log("CHANGE" + id);  wss\_client.send('rooms/' + id + '/ColourChange',JSON.stringify(self.getCurColorRgb()),1,false);  });  function UpdateFillModal(Id,roomName,DeviceName,LightDevice){  if (Id <= 0){  $('#fillModal .modal-title').html("Add room");  $('#IotId').parent().hide();  $('#saveIot').html("Add");  }else{  $('#fillModal .modal-title').html("Update Room Info");  $('#IotId').parent().show();  $('#saveIot').html("Save chanages");  }  $('#IotId').val(Id);  $('#roomName').val(roomName);  $('#IotDeviceName').val(DeviceName);  $('#IoTLightDevice').val(LightDevice);  }  var table = $('#roomsTable').DataTable({  'oLanguage': {  'sEmptyTable': 'Loading. Please wait...',  'sLengthMenu': '<div class="row item\_to\_show"><div class="col-md-7 col-sm-12">Items to show:</div>' +  '<div class="col-md-5 col-sm-12"><select class="form-control">' +  '<option value="5">5</option>' +  '<option value="10">10</option>' +  '<option value="20" selected="selected">20</option>' +  '<option value="25">25</option>' +  '<option value="50">50</option>' +  '<option value="100">100</option>' +  '<option value="250">250</option>' +  '</select></div></div>',  "sSearch": ''  },  'iDisplayLength': 20,  'sPaginationType': 'full\_numbers',  'aoColumns': [  {  'bSearchable': false,  'mData':'Id'  },  {  'bSearchable': true,  'mData':'RoomName'  },  {  'bSearchable': true,  'mData':'IotDeviceName'  }, {  'bSearchable': true,  'mData':'IoTLightDevice'  },  {  'bSearchable': false,  'bSortable': false,  'mData': null,  'defaultContent':  '<a class="btn btn-primary btnUpdate">Update <i class="fa fa-refresh" aria-hidden="true"></i></a>' +  ' <a class="btn btn-primary btnRemove">Remove <i class="fa fa-trash" aria-hidden="true"></i></a>' +  ' <a class="btn btn-primary btnSwitch">Switch On the lights <i class="fa fa-power-off" aria-hidden="true"></i></a>' +  ' <a class="btn btn-primary btnColour">Change Colour</a>'  }  ],  ajax: {  url: '/api/getRooms',  dataSrc: ''  }  }).on('draw',function(){  var datas = table.ajax.json();  $(".btnUpdate").click(function(){  var data = datas[$(this).parent().parent().index()];  UpdateFillModal(data.Id,data.RoomName,data.IotDeviceName,data.IoTLightDevice);  $('#fillModal').modal('show');  });  $(".btnRemove").click(function(){  var data = datas[$(this).parent().parent().index()];  $('#IotRemoveId').val(data.Id);  $('#roomNameR').val(data.RoomName);  $('#IotDeviceNameR').val(data.IotDeviceName);  $('#IoTLightDeviceR').val(data.IoTLightDevice);  $('#confirmModal').modal('show');  });  $(".btnSwitch").click(function(){  var data = datas[$(this).parent().parent().index()];  var IsOn = $(this).text().indexOf("On") >= 0 ? "1" : "0";  $icon = $(this).children("i");  wss\_client.send('rooms/' + data.Id + '/LightSwitch',IsOn,1,false);  $(this).text(IsOn == "1" ? "Switch Off the lights " : "Switch On the lights ");  $(this).append($icon);  });  $(".btnColour").click(function(){  var data = datas[$(this).parent().parent().index()];  $('#IotColourId').val(data.Id);  $('#colourModal').modal('show');  });  });  $("#roomsTable\_filter label input").addClass("form-control").attr("placeholder", "Search rooms...");  $("#btnAdd").click(function(e){  UpdateFillModal(0,"","","");  $('#fillModal').modal('show');  });  $("#saveIot").click(function(e){  var IsAdd = $(this).html().indexOf("Add") >= 0;  var fd = new FormData(document.getElementById("iotForm"));  $.ajax({  type:"POST",  url: IsAdd ? "/api/addRoom" : "/api/updateRoom",  data: fd,  processData: false,  contentType: false,  success: function(data){  table.ajax.reload();  $('#fillModal').modal('hide');  },  error: function(XMLHttpRequest, textStatus, errorThrown) {  alert("Error: " + XMLHttpRequest.responseText);  }  });  });  $("#removeIoT").click(function(e){  var fd = new FormData(document.getElementById("iotRemoveForm"));  $.ajax({  type:"POST",  url:"/api/removeRoom",  data: fd,  processData: false,  contentType: false,  success: function(data){  table.ajax.reload();  $('#confirmModal').modal('hide');  }  });  });  }); |

## room\_access\_rights.js

The room\_access\_rights.js contains the code for room access rights

|  | Task |
| --- | --- |
|  | Create a js script **room\_access\_rights**.js with the code below |
|  | $(document).ready(function() {  function UpdateFillModal(Id, UserId, RoomId) {  if (Id <= 0) {  $('#fillModal .modal-title').html("Add Access Rights");  $('#Id').parent().hide();  $('#saveAccessRight').html("Add");  } else {  $('#fillModal .modal-title').html("Update Access Rights");  $('#Id').parent().show();  $('#saveAccessRight').html("Save chanages");  }  $('#Id').val(Id);  $('#UserSelect').val(UserId);  $('#RoomSelect').val(RoomId);  }  var table = $('#AccessRightsTable').DataTable({  'oLanguage': {  'sEmptyTable': 'No Data',  'sLengthMenu': '<div class="row item\_to\_show"><div class="col-md-7 col-sm-12">Items to show:</div>' +  '<div class="col-md-5 col-sm-12"><select class="form-control">' +  '<option value="5">5</option>' +  '<option value="10">10</option>' +  '<option value="20" selected="selected">20</option>' +  '<option value="25">25</option>' +  '<option value="50">50</option>' +  '<option value="100">100</option>' +  '<option value="250">250</option>' +  '</select></div></div>',  "sSearch": ''  },  'iDisplayLength': 20,  'sPaginationType': 'full\_numbers',  'aoColumns': [{  'bSearchable': false,  'mData': 'Id'  }, {  'bSearchable': true,  'mData': 'Username'  }, {  'bSearchable': true,  'mData': 'RoomName'  }, {  'bSearchable': false,  'bSortable': false,  'mData': null,  'defaultContent': '<a class="btn btn-primary btnUpdate">Update <i class="fa fa-refresh" aria-hidden="true"></i></a>' +  ' <a class="btn btn-primary btnRemove">Remove <i class="fa fa-trash" aria-hidden="true"></i></i></a>'  }],  ajax: {  url: '/api/getAccessRights',  dataSrc: ''  }  }).on('draw', function() {  var datas = table.ajax.json();  $(".btnUpdate").click(function() {  var data = datas[$(this).parent().parent().index()];  UpdateFillModal(data.Id, data.UserId, data.RoomId);  $('#fillModal').modal('show');  });  $(".btnRemove").click(function() {  var data = datas[$(this).parent().parent().index()];  $('#ACRemoveId').val(data.Id);  $('#UsernameR').val(data.Username);  $('#RoomNameR').val(data.RoomName);  $('#confirmModal').modal('show');  });  });  $("#AccessRightsTable\_filter label input").addClass("form-control").attr("placeholder", "Search...");  $("#btnAdd").click(function(e) {  UpdateFillModal(0, "", "", "");  $('#fillModal').modal('show');  });  $("#saveAccessRight").click(function(e) {  var IsAdd = $(this).html().indexOf("Add") >= 0;  var fd = new FormData(document.getElementById("accessRightForm"));  $.ajax({  type: "POST",  url: IsAdd ? "/api/addAccessRight" : "/api/updateAccessRight",  data: fd,  processData: false,  contentType: false,  success: function(data) {  table.ajax.reload();  $('#fillModal').modal('hide');  },  error: function(xhr, error) {  alert(xhr.responseText)  }  });  });  $("#removeAccessRight").click(function(e) {  var fd = new FormData(document.getElementById("ACRemoveForm"));  $.ajax({  type: "POST",  url: "/api/removeAccessRight",  data: fd,  processData: false,  contentType: false,  success: function(data) {  table.ajax.reload();  $('#confirmModal').modal('hide');  }  });  });  }); |

## Internal\_base.js

The Internal\_base.js contains the code for internal layout events

|  | Task |
| --- | --- |
|  | Create a js script **Internal\_base**.js with the code below |
|  | $(function($) {  setTimeout(function() {  $('#content-wrapper > .row').css({  opacity: 1  });  }, 200);  $('#sidebar-nav .dropdown-toggle').on('click', function(e) {  e.preventDefault();  var $item = $(this).parent();  if (!$item.hasClass('open')) {  $item.parent().find('.open .submenu').slideUp('fast');  $item.parent().find('.open').toggleClass('open');  }  $item.toggleClass('open');  if ($item.hasClass('open')) {  $item.children('.submenu').slideDown('fast');  } else {  $item.children('.submenu').slideUp('fast');  }  });  $('body').on('mouseenter', '#page-wrapper.nav-small #sidebar-nav .dropdown-toggle', function(e) {  var $sidebar = $(this).parents('#sidebar-nav');  if ($(document).width() >= 992) {  var $item = $(this).parent();  $item.addClass('open');  $item.children('.submenu').slideDown('fast');  }  });  $('body').on('mouseleave', '#page-wrapper.nav-small #sidebar-nav > .nav-pills > li', function(e) {  var $sidebar = $(this).parents('#sidebar-nav');  if ($(document).width() >= 992) {  var $item = $(this);  if ($item.hasClass('open')) {  $item.find('.open .submenu').slideUp('fast');  $item.find('.open').removeClass('open');  $item.children('.submenu').slideUp('fast');  }  $item.removeClass('open');  }  });  $('#make-small-nav').click(function(e) {  $('#page-wrapper').toggleClass('nav-small');  });  $(window).smartresize(function() {  if ($(document).width() <= 991) {  $('#page-wrapper').removeClass('nav-small');  }  });  $('.mobile-search').click(function(e) {  e.preventDefault();  $('.mobile-search').addClass('active');  $('.mobile-search form input.form-control').focus();  });  $(document).mouseup(function(e) {  var container = $('.mobile-search');  if (!container.is(e.target) && container.has(e.target).length === 0) {  container.removeClass('active');  }  });  $("[data-toggle='tooltip']").each(function(index, el) {  $(el).tooltip({  placement: $(this).data("placement") || 'top'  });  });  });  $.fn.removeClassPrefix = function(prefix) {  this.each(function(i, el) {  var classes = el.className.split(" ").filter(function(c) {  return c.lastIndexOf(prefix, 0) !== 0;  });  el.className = classes.join(" ");  });  return this;  };  (function($, sr) {  var debounce = function(func, threshold, execAsap) {  var timeout;  return function debounced() {  var obj = this,  args = arguments;  function delayed() {  if (!execAsap)  func.apply(obj, args);  timeout = null;  };  if (timeout)  clearTimeout(timeout);  else if (execAsap)  func.apply(obj, args);  timeout = setTimeout(delayed, threshold || 100);  };  }  jQuery.fn[sr] = function(fn) {  return fn ? this.bind('resize', debounce(fn)) : this.trigger(sr);  };  })(jQuery, 'smartresize'); |

## publicdashboard.js

The publicdashboard.js contains the code for public dashboard

|  | Task |
| --- | --- |
|  | Create a js script **publicdashboard**.js with the code below |
|  | $(document).ready(function() {  $("#cardIDbtn").click(function(e) {  var formdata = new FormData();  formdata.append("cardID", $("#cardID").val());  $.ajax({  type: "POST",  url: "/api/setCardID",  data: formdata,  processData: false,  contentType: false,  success: function(data) {  $("#cardIdWrapper").hide();  new Noty({  type: 'success',  layout: 'topCenter',  closeWith: ['click'],  timeout: 1500,  text: "Successfully added your Card ID!",  animation: {  open: 'animated bounceInRight',  close: 'animated bounceOutRight'  },  }).show();  },  error: function(xhr, error) {  new Noty({  type: 'error',  layout: 'topCenter',  modal: true,  closeWith: ['click'],  timeout: 1500,  text: xhr.responseText,  animation: {  open: 'animated bounceInRight',  close: 'animated bounceOutRight'  },  }).show();  }  });  e.preventDefault();  });  loadAccessRequestTable();  loadAccessRightsTable();  initRooms();  });  function initRooms() {  setInterval(function() {  d = new Date();  $(".room").each(function(index) {  var imageSrc = `https://s3-us-west-2.amazonaws.com/iotsmartroom/snapshot/${index+1}.png?` + d.getTime();  $(this).find(".liveImage").attr("src", imageSrc);  retrieveRoomValues(index+1, $(this));  });  }, 2000)  }  function retrieveRoomValues(roomID, room) {  $.ajax({  type: "GET",  url: "/api/LatestEnviroInfo/" + roomID,  dataType: "json",  success: function(data) {  console.log(data);  if (data != null) {  room.find('.caption').find(".time").text("Last Updated: " + data.time);  room.find('.caption').find(".light").text("Light: " + data.light);  room.find('.caption').find(".humidity").text("Humidity: " + data.humidity + "%");  room.find('.caption').find(".temp").text("Temperature: " + data.temp + "°C");  }  },  error: function(xhr, error) {  console.log(xhr.responseText);  }  });  }  function loadAccessRightsTable() {  var table = $('#accessRightsTable').DataTable({  "ajax": {  "url": "/api/getUserAccessRights",  "dataType": "json",  "dataSrc": "",  "contentType": "application/json",  },  "columns": [{  "data": "RoomName"  }  ],  "order": [  [ 0, "desc"]  ]  });  setInterval(function() {  table.ajax.reload();  }, 3000);  }  function loadAccessRequestTable() {  var table = $('#requestTable').DataTable({  "ajax": {  "url": "/api/getUserAccessRequests",  "dataType": "json",  "dataSrc": "",  "contentType": "application/json",  },  "columns": [{  "data": "RoomName"  },  {  "data": "DateRequested"  },  {  "data": "IsApproved",  "render": function(data, type, row) {  return (data == 0) ? '<span class="red">Rejected</span>' :  (data == 1) ? '<span class="green">Approved</span>' : '<span class="orange">Pending</span>';  }  }  ],  "order": [  [ 2, 'asc' ],  [ 0, "desc"]  ]  });  $("#requestFormBtn").click(function(e) {  var formdata = new FormData();  var roomName = $("#room option:selected").text();  formdata.append("roomID", $("#room").val());  $.ajax({  type: "POST",  url: "/api/requestNewAccess",  data: formdata,  processData: false,  contentType: false,  success: function(data) {  new Noty({  type: 'success',  layout: 'topCenter',  closeWith: ['click'],  timeout: 1500,  text: `Successfully requested access for ${roomName}!`,  animation: {  open: 'animated bounceInRight',  close: 'animated bounceOutRight'  },  }).show();  $('#myModal').modal('hide');  },  error: function(xhr, error) {  new Noty({  type: 'error',  layout: 'topCenter',  modal: true,  closeWith: ['click'],  timeout: 1500,  text: xhr.responseText,  animation: {  open: 'animated bounceInRight',  close: 'animated bounceOutRight'  },  }).show();  }  });  e.preventDefault();  table.ajax.reload();  });  setInterval(function() {  table.ajax.reload();  }, 3000);  } |

## manage\_approvals.js

The manage\_approvals.js contains the code for admin side manage approvals for public users access rights

|  | Task |
| --- | --- |
|  | Create a js script **manage\_approvals**.js with the code below |
|  | $(document).ready(function() {  loadPendingAccessRequestTable();  loadResolvedRequestTable();  });  function loadPendingAccessRequestTable() {  var table = $('#pendingTable').DataTable({  "ajax": {  "url": "/api/getPendingRequests",  "dataType": "json",  "dataSrc": "",  "contentType": "application/json",  },  "columns": [{  "data": "Username"  },  {  "data": "RoomName"  },  {  "data": "DateRequested"  },  {  "data": "IsApproved",  "render": function(data, type, row) {  return '<a class="btn btn-success btnApprove">Approve <i class="fa fa-refresh" aria-hidden="true"></i></a>' +  ' <a class="btn btn-primary btnReject">Reject <i class="fa fa-trash" aria-hidden="true"></i></i></a>'  }  },  {  "data": "Id"  }  ],  "order": [  [2, "desc"]  ],  "columnDefs": [{  "targets": [4],  "visible": false,  "searchable": false  }]  });  $('#pendingTable').on('click', '.btnApprove', function() {  var RowIndex = $(this).closest('tr');  var requestID = table.row(RowIndex).data().Id;  var formdata = new FormData();  formdata.append("approval", 1);  $.ajax({  type: "POST",  url: "/api/approveAccessRequest/" + requestID,  data: formdata,  processData: false,  contentType: false,  success: function(data) {  new Noty({  type: 'success',  layout: 'topCenter',  modal: true,  closeWith: ['click'],  timeout: 1500,  text: "Successfully Approved Access Rights Request",  animation: {  open: 'animated bounceInRight',  close: 'animated bounceOutRight'  },  }).show();  },  error: function(xhr, error) {  new Noty({  type: 'error',  layout: 'topCenter',  modal: true,  closeWith: ['click'],  timeout: 1500,  text: xhr.responseText,  animation: {  open: 'animated bounceInRight',  close: 'animated bounceOutRight'  },  }).show();  }  });  table.ajax.reload();  });  $('#pendingTable').on('click', '.btnReject', function() {  var RowIndex = $(this).closest('tr');  var requestID = table.row(RowIndex).data().Id;  var formdata = new FormData();  formdata.append("approval", 0);  $.ajax({  type: "POST",  url: "/api/approveAccessRequest/" + requestID,  data: formdata,  processData: false,  contentType: false,  success: function(data) {  new Noty({  type: 'success',  layout: 'topCenter',  modal: true,  closeWith: ['click'],  timeout: 1500,  text: "Successfully Rejected Access Rights Request",  animation: {  open: 'animated bounceInRight',  close: 'animated bounceOutRight'  },  }).show();  },  error: function(xhr, error) {  new Noty({  type: 'error',  layout: 'topCenter',  modal: true,  closeWith: ['click'],  timeout: 1500,  text: xhr.responseText,  animation: {  open: 'animated bounceInRight',  close: 'animated bounceOutRight'  },  }).show();  }  });  table.ajax.reload();  });  setInterval(function() {  table.ajax.reload();  }, 3000);  }  function loadResolvedRequestTable() {  var table = $('#resolvedTable').DataTable({  "ajax": {  "url": "/api/getCompletedRequests",  "dataType": "json",  "dataSrc": "",  "contentType": "application/json",  },  "columns": [{  "data": "Username"  },  {  "data": "RoomName"  },  {  "data": "DateRequested"  },  {  "data": "IsApproved",  "render": function(data, type, row) {  return (data == 0) ? '<span class="red">Rejected</span>' :  (data == 1) ? '<span class="green">Approved</span>' : '<span class="orange">Pending</span>';  }  }  ],  "order": [  [2, "desc"]  ]  });  setInterval(function() {  table.ajax.reload();  }, 3000);  } |

# Section 10 Creating server files

## SigV4Utils.py

The SigV4Utils.py contains the code to help generate aws websocket endpoint for aws iot mqtt

|  | Task |
| --- | --- |
|  | Create a python script **SigV4Utils.py** with the code below |
|  | import hmac  import hashlib  class SigV4Utils:  @staticmethod  def getSignatureKey(key, dateStamp, regionName, serviceName):  kDate = hmac.new(bytes('AWS4'+ key , 'latin-1'),msg = bytes(dateStamp , 'latin-1'),  digestmod = hashlib.sha256).hexdigest()  kRegion = hmac.new(bytearray.fromhex(kDate),msg = bytes(regionName , 'latin-1'),  digestmod = hashlib.sha256).hexdigest()  kService = hmac.new(bytearray.fromhex(kRegion),msg = bytes(serviceName , 'latin-1'),  digestmod = hashlib.sha256).hexdigest()  kSigning = hmac.new(bytearray.fromhex(kService),msg = bytes('aws4\_request' , 'latin-1'),  digestmod = hashlib.sha256).hexdigest()  return kSigning  @staticmethod  def sign(key, msg):  return hmac.new(bytearray.fromhex(key),msg = bytes(msg , 'latin-1'),  digestmod = hashlib.sha256).hexdigest()  @staticmethod  def sha256(msg):  return hashlib.sha256(bytes(msg, 'latin-1')).hexdigest() |

## server.py

The server.py contains the code serving webpages

|  | Task |
| --- | --- |
|  | Create a python script **server.py** with the code below |
|  | from flask import Flask, render\_template, jsonify, request, redirect, url\_for,session,send\_from\_directory  from Shared.User import User  from Shared.Room import Room  from Shared.AccessRight import AccessRight  from Shared.AccessLog import AccessLog  from Shared.AccessRequest import AccessRequest  from Shared.EnviroInfo import EnviroInfo  from Shared.MotionEvent import MotionEvent  from Shared.Configs import Config  import signal  import socket  from datetime import datetime  from SigV4Utils import SigV4Utils  from urllib.parse import quote\_plus  import os  from twilio.twiml.messaging\_response import Message, MessagingResponse  from Bots.whatsappbot import WhatsAppBot  app = Flask(\_\_name\_\_)  app.jinja\_env.trim\_blocks = True  app.jinja\_env.lstrip\_blocks = True  app.secret\_key = b'\xe1\xa5\x0f\xe7\x12KQ\xb0\xd3e\xbf\xb5\xfb\xdf]\xbf'  def IsAuthenticated():  if 'UserId' not in session:  return False  return True  def IsAuthorised():  if not IsAuthenticated() or not session['IsAdmin']:  return False  return True  def redirectIfNotLoginAndRender(templateName, \*\*kwargs):  if IsAuthenticated() == False:  return redirect(url\_for('.loginPage'))  return render\_template(templateName,Username = session['Username'],IsAdmin = session['IsAdmin'],\*\*kwargs)  @app.route("/")  def loginPage():  if IsAuthenticated() == True:  return redirect(url\_for('.dashboardPage')) if session['IsAdmin'] else redirect(url\_for('.publicDashBoardPage'))  errorMsg = ""  errorCode = request.args.get('error')  if errorCode != None:  if int(errorCode) == 1:  errorMsg = "Wrong username or password"  return render\_template('login.html',error=errorMsg)  @app.route("/register")  def registerPage():  errorMsg = ""  errorCode = request.args.get('error')  if errorCode != None:  if int(errorCode) == 1:  errorMsg = "Username or password cannot be blank"  if int(errorCode) == 2:  errorMsg = "Passwords do not match"  return render\_template('register.html',error=errorMsg)  @app.route("/dashboard")  def dashboardPage():  if IsAuthenticated() == True:  return redirectIfNotLoginAndRender('dashboard.html',rooms = Room.GetAllRooms()) if session['IsAdmin'] else redirect(url\_for('.publicDashBoardPage'))  return redirect(url\_for('.loginPage'))  @app.route("/publicdashboard")  def publicDashBoardPage():  return redirectIfNotLoginAndRender('publicDashboard.html', user = User.TryGetUserById(session['UserId']), rooms = Room.GetAllRooms()) if not session['IsAdmin'] else redirect(url\_for('.dashboardPage'))  @app.route("/ManageRooms")  def manageRoomsPage():  return redirectIfNotLoginAndRender('manage\_rooms.html')  @app.route("/RoomAccessRights")  def roomAccessRightsPage():  return redirectIfNotLoginAndRender('room\_access\_rights.html',users = User.GetUsers(), rooms = Room.GetAllRooms())  @app.route("/ManageApprovals")  def manageApprovalsPage():  return redirectIfNotLoginAndRender('manage\_approvals.html',users = User.GetUsers(), rooms = Room.GetAllRooms())  #----------------------------------------------------------------------------------------------  @app.route("/api/login",methods = ['POST'])  def loginAPI():  user = User.ParseFromForm(request.form)  if not user.IsValid():  return redirect(url\_for('.loginPage', error=1))  elif not user.TryLogin():  return redirect(url\_for('.loginPage', error=1))  session['UserId'] = user.Id  session['Username'] = user.Username  session['IsAdmin'] = user.IsAdmin()  return redirect(url\_for('.dashboardPage')) if user.IsAdmin() else redirect(url\_for('.publicDashBoardPage'))  @app.route("/api/register",methods = ['POST'])  def registerAPI():  user = User.ParseRegistrationForm(request.form)  if user.registerUser() == "1":  return redirect(url\_for('.registerPage', error=1))  if user.registerUser() == "2":  return redirect(url\_for('.registerPage', error=2))  return redirect(url\_for('.loginPage'))  @app.route("/api/logout")  def logoutAPI():  session.pop('UserId', None)  session.pop('Username', None)  return redirect(url\_for('.loginPage'))  #----------------------------------------------------------------------------------------------  @app.route("/api/getRooms")  def getRoomsAPI():  if not IsAuthorised():  return "Not authorised", 401  return jsonify(Room.GetAllRoomsJSON())  @app.route("/api/updateRoom",methods = ['POST'])  def updateRoomAPI():  if not IsAuthorised():  return "Not authorised", 401  room = Room.ParseFromForm(request.form)  if not room.IsValidForUpdate():  return "Missing Data", 400  if room.TryUpdateDb() == False:  return "Failed to update Database", 401  return "Updated"  @app.route("/api/addRoom",methods = ['POST'])  def addRoomAPI():  if not IsAuthorised():  return "Not authorised", 401  room = Room.ParseFromForm(request.form)  if not room.IsValidForAdd():  return "Missing Data", 400  if room.TryAdd() == False:  return "Failed to add into Database", 401  return "added"  @app.route("/api/removeRoom",methods = ['POST'])  def removeRoomAPI():  if not IsAuthorised():  return "Not authorised", 401  Id = request.form['IotId']  if Id == None:  return "Missing Data", 400  room = Room(Id = Id)  if not room.TryRemove():  return "Failed to remove from Database", 401  return "removed"  @app.route("/protected/<path:filename>")  def protected(filename):  print(filename)  print(app.instance\_path)  if not IsAuthorised():  return "Not authorised", 401  return send\_from\_directory(  os.path.join(app.instance\_path, 'protected'),  filename  )  #----------------------------------------------------------------------------------------------  @app.route("/api/getAccessRights")  def getAccessRightsAPI():  if not IsAuthorised():  return "Not authorised", 401  return jsonify(AccessRight.GetAllAccessRights())  @app.route("/api/updateAccessRight",methods = ['POST'])  def updateAccessRightAPI():  if not IsAuthorised():  return "Not authorised", 401  access\_right = AccessRight.ParseFromForm(request.form)  if not access\_right.IsValidForUpdate():  return "Missing Data", 400  elif not access\_right.TryUpdateDb():  return "Failed to update Database", 401  return "Updated"  @app.route("/api/addAccessRight",methods = ['POST'])  def addAccessRightAPI():  if not IsAuthorised():  return "Not authorised", 401  access\_right = AccessRight.ParseFromForm(request.form)  if not access\_right.IsValidForAdd():  return "Missing Data", 400  elif not access\_right.TryAdd():  return "Data already exist", 400  return "added"  @app.route("/api/removeAccessRight",methods = ['POST'])  def removeAccessRightAPI():  if not IsAuthorised():  return "Not authorised", 401  access\_right = AccessRight.ParseFromForm(request.form)  if not access\_right.IsValidForRemove():  return "Missing Data", 400  elif not access\_right.TryRemove():  return "Failed to remove from Database", 401  return "removed"  #----------------------------------------------------------------------------------------------  @app.route("/api/getAvgRoomTemps/<int:roomId>")  def getAvgRoomTempsAPI(roomId):  if not IsAuthorised():  return "Not authorised", 401  room = Room.TryGetRoomById(roomId)  if room == None:  return "Error",400  result = {  "RoomName":room.RoomName,  "data":EnviroInfo.GetAvgEnviroInfoByDay(room.Id)  }  return jsonify(result)  #----------------------------------------------------------------------------------------------  @app.route("/api/getValidAccessLog/<int:roomId>")  def getValidAccessLogAPI(roomId):  if not IsAuthorised():  return "Not authorised", 401  result = AccessLog.GetValidExitedAccessLogForRoom(roomId)  return jsonify(result)  @app.route("/api/getInvalidAccessLog/<int:roomId>")  def getInvalidAccessLogAPI(roomId):  if not IsAuthorised():  return "Not authorised", 401  result = AccessLog.GetInvalidExitedAccessLogForRoom(roomId)  return jsonify(result)  #----------------------------------------------------------------------------------------------  @app.route("/api/setLightStatus",methods = ['POST'])  def setLightStatusAPI():  if not IsAuthorised():  return "Not authorised", 401  RoomId = request.form.get("RoomId")  Status = request.form.get("Status")  room = Room.TryGetRoom(socket.gethostname())  if room == None or room.Id != int(RoomId):  return "Error, not current room",401  return "success"  #----------------------------------------------------------------------------------------------  @app.route("/api/getMotionEvents/<path:roomId>")  def getMotionEventsAPI(roomId):  if not IsAuthorised():  return "Not authorised", 401  return jsonify(MotionEvent.GetMotionEvents(roomId))  #----------------------------------------------------------------------------------------------  @app.route("/api/getAwsIotWSS")  def getAwsIotWSS():  if not IsAuthorised():  return "Not authorised", 401  time = datetime.utcnow()  date\_stamp = time.strftime('%Y%m%d')  amzdate = date\_stamp + 'T' + time.strftime('%H%M%S') + 'Z'  service = 'iotdevicegateway'  canonicalUri = '/mqtt'  algorithm = 'AWS4-HMAC-SHA256'  credentialScope = date\_stamp + '/' + Config.aws\_region + '/' + service + '/' + 'aws4\_request'  canonicalQuerystring = 'X-Amz-Algorithm=' + algorithm  canonicalQuerystring += '&X-Amz-Credential=' + quote\_plus(Config.aws\_access\_key + '/' + credentialScope)  canonicalQuerystring += '&X-Amz-Date=' + amzdate  canonicalQuerystring += '&X-Amz-SignedHeaders=host'  canonicalHeaders = 'host:' + Config.aws\_endpoint + '\n'  payloadHash = SigV4Utils.sha256('')  canonicalRequest = 'GET' + '\n' + canonicalUri + '\n' + canonicalQuerystring + '\n' + canonicalHeaders + '\nhost\n' + payloadHash  stringToSign = algorithm + '\n' + amzdate + '\n' + credentialScope + '\n' + SigV4Utils.sha256(canonicalRequest)  signingKey = SigV4Utils.getSignatureKey(Config.aws\_secret\_key,date\_stamp,Config.aws\_region,service)  signature = SigV4Utils.sign(signingKey, stringToSign)  canonicalQuerystring += '&X-Amz-Signature=' + signature  return ('wss://' + Config.aws\_endpoint + canonicalUri + '?' + canonicalQuerystring),200  #----------------------------------------------------------------------------------------------  #PUBLIC DASHBOARD API CALLS  @app.route("/api/setCardID", methods = ['POST'])  def setUserCardID():  if not IsAuthenticated():  return "Not authorised", 401  cardID = request.form.get("cardID")  userID = session['UserId']  if cardID == "" or cardID == None:  return "Error, A card ID cannot be blank", 400  result = User.addCardId(cardID, userID)  if result == False:  return "Error, A card with this ID already exists",400  return "success"  @app.route("/api/getUserAccessRequests")  def getUserAccessRequestsWithoutUserID():  if not IsAuthenticated():  return "Not authorised", 401  UserId = session['UserId']  result = AccessRequest.getUserAccessRequests(UserId)  return jsonify(result)  @app.route("/api/requestNewAccess", methods = ['POST'])  def requestNewAccess():  if not IsAuthenticated():  return "Not authorised", 401  roomID = request.form.get("roomID")  userID = session['UserId']  status = AccessRequest.checkIfAlreadyRequested(roomID, userID)  if status == 0 or status == 3:  result = AccessRequest.requestNewAccess(roomID, userID)  if result is False:  return "Error, something went wrong when inserting value", 400  elif status == 1:  return "Error, Your request for this room is already approved", 400  elif status == 2:  return "Error, Your last request has not passed 10 minutes", 400  elif status == 4:  return "Error, Your request is pending", 400  return "Successfully requested for access for Room"  @app.route("/api/getUserAccessRights")  def getUserAccessRightsWithoutUserID():  if not IsAuthenticated():  return "Not authorised", 401  UserId = session['UserId']  result = AccessRight.getUserAccessRights(UserId)  return jsonify(result)  @app.route("/api/LatestEnviroInfo/<int:roomID>")  def GetRoomLatestEnviroInfo(roomID):  if not IsAuthenticated():  return "Not authorised", 401  result = EnviroInfo.GetLatestEnviroInfo(roomID)  return jsonify(result)  #----------------------------------------------------------------------------------------------  # ADMIN ACCESS RIGHTS APPROVAL REQUESTS APIs  @app.route("/api/getPendingRequests")  def getPendingAccessRequests():  if not IsAuthorised():  return "Not authorised", 401  result = AccessRequest.getAllPendingRequests()  return jsonify(result)  @app.route("/api/getCompletedRequests")  def getCompletedRequests():  if not IsAuthorised():  return "Not authorised", 401  result = AccessRequest.getCompletedRequests()  return jsonify(result)  @app.route("/api/approveAccessRequest/<int:requestID>", methods = ['POST'])  def updateRequestApprovalStatus(requestID):  if not IsAuthenticated():  return "Not authorised", 401  approval = request.form.get("approval")  print(approval)  result = AccessRequest.updateRequestApprovalStatus(requestID, approval)  return jsonify(result)  #----------------------------------------------------------------------------------------------  # WhatsAPP Receive and handle incoming messages  @app.route("/api/whatsapp",methods = ['POST'])  def handleWhatsApp():  number = request.form['From']  message\_body = request.form['Body']  WhatsAppBot.createMessage(message\_body,number)  return "successfully sent message"  #----------------------------------------------------------------------------------------------  @app.errorhandler(404)  def page\_not\_found(e):  return render\_template('404.html'), 404  if \_\_name\_\_ == '\_\_main\_\_':  app.run(debug=True,host='0.0.0.0') |

## telegramBot.py

The telegramBot.py contains the code serving telegram users

|  | Task |
| --- | --- |
|  | Create a python script **telegramBot.py** with the code below |
|  | from Shared.Database import db  import telepot  from Shared.User import User  from Shared.Configs import Config  from Shared.Room import Room  from Shared.AccessLog import AccessLog  from Shared.EnviroInfo import EnviroInfo  class TelegramBot:  def \_\_init\_\_(self,token):  self.bot = telepot.Bot(token)  self.bot.message\_loop(self.respondToMsg)  self.ChatUsers = []  self.Commands = {  'NotifyMe': self.DoNotifyMeCommand,  'RoomAvail': self.DoRoomAvailCommand,  'RoomSnap': self.DoRoomSnapCommand,  'CurrentTemp': self.DoCurrentTempCommand  }  def respondToMsg(self,msg):  chat\_id = msg['chat']['id']  command = msg['text']  print('Got command: {}'.format(command))  user = self.FindChatUser(chat\_id)  if user == None:  user = {"Id": chat\_id, "CurrentCommand": '', "CommandStep": 0 }  self.ChatUsers.append(user)  if len(user['CurrentCommand']) > 0:  command = user['CurrentCommand']  if command in self.Commands:  user['CurrentCommand'] = command  self.Commands[command](user,msg['text'])  else:  result = "Unknown command.\nHi there, here are list of commands to get started.\n"  index = 1  for k,v in self.Commands.items():  result += str(index) + ")"  index += 1  result += k + '\n'  self.bot.sendMessage(user['Id'], result)  def DoNotifyMeCommand(self,user,text):  if user['CommandStep'] == 0:  self.bot.sendMessage(user['Id'], "Please enter your username.")  user['CommandStep'] = 1  elif user['CommandStep'] == 1:  user['Username'] = text  self.bot.sendMessage(user['Id'], "Please enter your Password.")  user['CommandStep'] = 2  elif user['CommandStep'] == 2:  user['Password'] = text  U = User(Username = user['Username'], Password = user['Password'])  if not U.TryLogin():  self.bot.sendMessage(user['Id'], "Wrong Username or password, please enter username again.")  user['CommandStep'] = 1  else:  user['CommandStep'] = 0  user['CurrentCommand'] = ""  U.UpdateChatId(user['Id'])  self.bot.sendMessage(user['Id'], "Welcome {}, you have been registered to be notified upon motion detection".format(U.Username))  def DoRoomAvailCommand(self,user,text):  user['CurrentCommand'] = ""  rooms = Room.GetAllRooms()  result = 'Rooms Available:\n'  for room in rooms:  if AccessLog.IsRoomAvailable(room.Id):  result += room.RoomName + '\n'  self.bot.sendMessage(user['Id'], result)  def DoRoomSnapCommand(self,user,text):  if user['CommandStep'] == 0:  self.bot.sendMessage(user['Id'], "Please enter room name.")  user['CommandStep'] = 1  elif user['CommandStep'] == 1:  user['CommandStep'] = 0  user['CurrentCommand'] = ""  rooms = Room.TryGetRoomByRoomName(text)  if rooms == None or len(rooms) <= 0:  return self.bot.sendMessage(user['Id'],"Failed to find room: {}".format(text))  for room in rooms:  self.bot.sendMessage(user['Id'],"Room Name: {}".format(room.RoomName))  self.bot.sendDocument(user['Id'], Config.aws\_S3\_endpoint + 'snapshot/' + str(room.Id) + '.png')  def DoCurrentTempCommand(self,user,text):  if user['CommandStep'] == 0:  self.bot.sendMessage(user['Id'], "Please enter room name.")  user['CommandStep'] = 1  elif user['CommandStep'] == 1:  user['CommandStep'] = 0  user['CurrentCommand'] = ""  rooms = Room.TryGetRoomByRoomName(text)  if rooms == None or len(rooms) <= 0:  return self.bot.sendMessage(user['Id'],"Failed to find room: {}".format(text))  for room in rooms:  self.bot.sendMessage(user['Id'],"Room Name: {}".format(room.RoomName))  info = EnviroInfo.GetLatestEnviroInfo(room.Id)  self.bot.sendMessage(user['Id'],"Temp: {} C\nHumidity: {}\nLight: {}\nLast Updated: {}".format(info['temp'],info['humidity'],info['light'],info['time']))  def NotifyEveryone(self, roomName, VideoPath):  UserList = User.GetChatIdUsers()  file\_id = 0  is\_document = False  for user in UserList:  self.bot.sendMessage(user.ChatId,"Detected motion at {}".format(roomName))  if file\_id <= 0:  try:  r = self.bot.sendVideo(user.ChatId, open(VideoPath, 'rb'))  file\_id = r['video']['file\_id']  except KeyError:  file\_id = r['document']['file\_id']  is\_document = True  else:  if is\_document:  self.bot.sendDocument(chat\_id, file\_id)  else:  self.bot.sendVideo(chat\_id, file\_id)  def NotifyEveryoneRemoteFile(self, roomName, VideoPath):  UserList = User.GetChatIdUsers()  if UserList == None:  return  file\_id = 0  is\_document = False  for user in UserList:  self.bot.sendMessage(user.ChatId,"Detected motion at {}".format(roomName))  if file\_id <= 0:  try:  r = self.bot.sendVideo(user.ChatId, Config.aws\_S3\_endpoint + VideoPath)  file\_id = r['video']['file\_id']  except KeyError:  file\_id = r['document']['file\_id']  is\_document = True  else:  if is\_document:  self.bot.sendDocument(chat\_id, file\_id)  else:  self.bot.sendVideo(chat\_id, file\_id)  def FindChatUser(self,chat\_id):  for user in self.ChatUsers:  if user["Id"] == chat\_id:  return user  return None |

## TeleRunner.py

The TeleRunner.py contains the code serving telegram by using TelegramBot and awsIot class

|  | Task |
| --- | --- |
|  | Create a python script **TeleRunner.py** with the code below |
|  | from TelegramBot import TelegramBot  from RaspberryPi.AwsIot import AwsIot  from Shared.Room import Room  from time import sleep  import json  awsiot = AwsIot('TeleBot',0)  tele = TelegramBot('insert your key here’)  def MotionEventCallback(client, userdata, message):  print(message.payload)  motion\_event = json.loads(message.payload)  tele.NotifyEveryoneRemoteFile(motion\_event.get('roomName'),motion\_event.get("FilePath"))  awsiot.my\_rpi.subscribe('rooms/MotionEvents',1,MotionEventCallback)  while True:  sleep(1) |

## discordbot.py

The discordbot.py contains the code serving discord

|  | Task |
| --- | --- |
|  | Create a python script **discordbot.py** with the code below |
|  | #using discord.py rewrite version  from discord.ext import commands  import discord, datetime  import sys  import os  sys.path.append(os.path.abspath('..'))  from Shared.Database import db  from Shared.Room import Room  from Shared.AccessLog import AccessLog  from Shared.Configs import Config  from Shared.EnviroInfo import EnviroInfo  bot = commands.Bot(command\_prefix="!", status=discord.Status.idle, activity=discord.Game(name="Booting.."))  bot.remove\_command('help')  @bot.event  async def on\_ready():  print("Ready to go!")  print(f"Serving: {len(bot.guilds)} guilds.")  await bot.change\_presence(status=discord.Status.online, activity=(discord.Game(name="Active!")))  @bot.event  async def on\_command\_error(ctx, error):  user = ctx.author  if not isinstance(error, commands.CheckFailure):  await ctx.channel.send(f"{user.mention}, {error} \nType !help for all commands")  @bot.command()  async def help(ctx):  await ctx.channel.send(f"========All Commands==========\n• !rooms - To see all available rooms\n• !snapshot - To get all IOT Sensor data of all rooms\n• !ping - To see the latency of the bot")  #test command 1  #!ping  @bot.command()  async def ping(ctx):  ping\_ = bot.latency  ping = round(ping\_ \* 1000)  await ctx.channel.send(f"My ping is {ping}ms")  #test command 2  #!message test  @bot.command()  async def message(ctx, message):  user = ctx.author  if message != None:  await ctx.channel.send(f"{user} sent {message}")  await ctx.message.delete()  @bot.command()  async def snapshot(ctx):  roomsList = Room.GetAllRooms()  await ctx.channel.send(f"Requested by {ctx.author.mention}")  for oneRoom in roomsList:  roomInfo = EnviroInfo.GetLatestEnviroInfo(oneRoom.Id)  imageUrl = f"{Config.aws\_S3\_endpoint}snapshot/{oneRoom.Id}.png"  if (roomInfo is not None):  await ctx.channel.send(f"====================================\nRoom Name: {oneRoom.RoomName}\nLast Updated: {roomInfo['time']}\nTemperature: {roomInfo['temp']}°C" +  f"\nHumidity: {roomInfo['humidity']}%\nLight: {roomInfo['light']}\n" + imageUrl)  await ctx.message.delete()  #get available rooms  #!rooms  @bot.command()  async def rooms(ctx):  currentDT = datetime.datetime.now().strftime("%Y-%m-%d %I:%M:%S %p")  roomsavailable = []  allRooms = Room.GetAllRooms();  for oneRoom in allRooms:  roomsInfo = AccessLog.GetLatestExitAccessLog(oneRoom.Id)  if (roomsInfo != None):  for x in roomsInfo:  if ([x['Exit\_time']][0] != None):  roomsavailable.append([x['RoomName']])  if roomsavailable == []:  roomsavailable.append("No Rooms Available")  await ctx.channel.send(f"=====================\nTime Requested: {currentDT} \nRequested By: {ctx.author.mention} \nRooms available: {roomsavailable}\n=====================")  #delete command message  await ctx.message.delete()  # @bot.command()  # async def subscribe(ctx, member):  # await ctx.message.delete()  #  # @bot.command()  # async def unsubscribe(ctx):  # await ctx.message.delete()  #shut down the bot  @bot.command()  async def s(ctx):  await bot.close()  bot.run("") |

## whatsappbot.py

The whatsappbot.py contains the code serving whatsapp

|  | Task |
| --- | --- |
|  | Create a python script **whatsappbot.py** with the code below |
|  | import sys  import os  sys.path.append(os.path.abspath('..'))  from flask import Flask, render\_template, jsonify, request, redirect, url\_for,session,send\_from\_directory  from twilio.twiml.messaging\_response import Message, MessagingResponse  from twilio.rest import Client  from Shared.Database import db  from Shared.Room import Room  from Shared.AccessLog import AccessLog  from Shared.Configs import Config  from Shared.EnviroInfo import EnviroInfo  import datetime  account\_sid = 'insert account sid here'  auth\_token = 'insert token here'  client = Client(account\_sid, auth\_token)  botNum = 'whatsapp:+14155238886'  def sendMessage(message, senderNum):  message = client.messages.create(from\_=botNum,body=message,to=senderNum)  class WhatsAppBot:  def createMessage(message, senderNum):  if (message == "/rooms"):  roomsavailable = []  allRooms = Room.GetAllRooms();  for oneRoom in allRooms:  roomsInfo = AccessLog.GetLatestExitAccessLog(oneRoom.Id)  if (roomsInfo != None):  for x in roomsInfo:  if ([x['Exit\_time']][0] != None):  roomsavailable.append([x['RoomName']][0])  if roomsavailable == []:  roomsavailable.append("No Rooms Available")  messageToSend = "Rooms Available \n"  availableInString = '\n'.join(roomsavailable)  messageToSend += availableInString  sendMessage(messageToSend, senderNum)  elif (message == "/help"):  sendMessage("=====All Commands=====\n• /rooms - To see all available rooms\n• /snapshot - To get all IOT Sensor data of all rooms\n", senderNum)  elif (message == "/snapshot"):  messageToSend = ""  roomsList = Room.GetAllRooms()  for oneRoom in roomsList:  roomInfo = EnviroInfo.GetLatestEnviroInfo(oneRoom.Id)  imageUrl = f"{Config.aws\_S3\_endpoint}snapshot/{oneRoom.Id}.png"  if (roomInfo is not None):  roomMessage = f"=============\nRoom Name: {oneRoom.RoomName}\nLast Updated: {roomInfo['time']}\nTemperature: {roomInfo['temp']}°C" + f"\nHumidity: {roomInfo['humidity']}%\nLight: {roomInfo['light']}\n" + imageUrl + "\n"  messageToSend += roomMessage  # sendMessage(f"=====\nRoom Name: {oneRoom.RoomName}\nLast Updated: {roomInfo['time']}\nTemperature: {roomInfo['temp']}°C" +  # f"\nHumidity: {roomInfo['humidity']}%\nLight: {roomInfo['light']}\n" + imageUrl, senderNum)  sendMessage(messageToSend, senderNum)  else:  sendMessage("Unknown Command, Type /help for more information.", senderNum) |

# Section 11 Test the program

Let’s run the python program and properly configure the database data.

## A. Run the program

|  | Task |
| --- | --- |
|  | Start a command prompt and type in the following command to change the directory to CA2  cd CA2 |
|  | Type in the following command to run the Python program  Sudo python SmartSecurity.py |
|  | Launch another terminal, and type in the following command to run the Updater python program  cd CA2  python Updater.py |

## B. Expected Results

|  | Task |
| --- | --- |
|  | If everything goes well, the expected output on your Raspberry Pi screen would look similar to that below |
|  | The admin dashboard will be updated with new data as well |
|  | You should be register for an new account |
|  | You should be able to access the public dashboard with your new account or login with an existing account |
|  | You should be able to set your card ID so that you are able to request rights to access rooms using your card ID Note: You will be given an Card ID by the admin in production scenario |
|  | The public dashboard will be updated with the live data of each room |
|  | You should be able to request for access rights to any room for your card ID from the admin |
|  | You should be able to see your request once you created one |
|  | You should not be able to access the room if your card is not approved yet |
|  | Admin should see a new access right request has been created by you |
|  | You should see your request has been approved if the admin approved it |
|  | You should see your access rights has been updated as well |
|  | You should be able to access the room as your card has been approved for access to the room you chosen |
|  | You should be able to add our discord bot to your discord server using the invite link <https://discordapp.com/oauth2/authorize?client_id=545638414267973632&scope=bot&permissions=8> |
|  | You should see a helper when you type an unknown command |
|  | Find all out all commands available by typing !help |
|  | You should be able to find out which rooms are currently available using !rooms command |
|  | You should be able to get all iot sensor data of all rooms using !snapshot |
|  | You should be able to receive and send messages from our Whatsapp Bot if your phone number is whitelisted by us. To be whitelisted, send ‘join tight-truck’ to +14155238886 |
|  | You should see a helper when you type an unknown command |
|  | You are able to send the message /rooms to see what rooms are available |
|  | You should be able to get latest iot data of all rooms by typing /snapshot |
|  | Invite our Telegram Bot to your telegram by adding @wk\_rpi\_bot |
|  | Example of our telegam bot working |
|  | Allow the telegram bot to notify you when motion has been detected in a room |
|  | If you enabled notifications, you will notified when a motion is detected in a room when no one is inside the room. |

**-- End of CA2 Step-by-step tutorial --**