**SCHOOL OF COMPUTING (SOC)**

|  |  |
| --- | --- |
| **Date of Submission:** | 17th February 2019 |

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

|  |  |
| --- | --- |
| **Class:** | DIT/FT/2A/33 |

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

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| 1727526  1727740  1727836 | Jasmin Yap Yi  Taperla Anne Muriele Tatac  Kevan Ong Jing Jie |

**IOT CA2**

**Step-by-step Tutorial**

**DIPLOMA IN INFORMATION TECHNOLOGY**

**ST0324 Internet of Things (IOT)**

**2018/2019 Semester 2**

**Table of Contents**

[Section 1 Overview of the application 4](#_Toc1270162)

[A. Where have we uploaded our tutorial? 4](#_Toc1270163)

[B. Why have we chosen to upload to this site? 4](#_Toc1270164)

[C. What have we uploaded? 4](#_Toc1270165)

[D. What is the application about? 4](#_Toc1270166)

[E. Summary of steps 4](#_Toc1270167)

[F. What does the final setup look like? 5](#_Toc1270168)

[Room One 6](#_Toc1270169)

[Room Two 7](#_Toc1270170)

[G. How does the web application look like? 7](#_Toc1270171)

[login.html 8](#_Toc1270172)

[Index.html 8](#_Toc1270173)

[RoomOne.html 9](#_Toc1270174)

[RoomTwo.html 9](#_Toc1270175)

[H. How does the final Fritzing diagram look like? 10](#_Toc1270176)

[Room One 10](#_Toc1270177)

[Room Two 11](#_Toc1270178)

[Section 2 Hardware requirements 12](#_Toc1270179)

[Hardware checklist 12](#_Toc1270180)

[DHT11 Temperature and Humidity Sensor 12](#_Toc1270181)

[Resistors (10K Ω and 220/330 Ω) 12](#_Toc1270182)

[LED 13](#_Toc1270183)

[Buzzer 13](#_Toc1270184)

[RFID/NFC MFRC522 Card Reader Module 13](#_Toc1270185)

[PIR Motion Sensor 14](#_Toc1270186)

[Section 3 Setting up the database using command-line 14](#_Toc1270187)

[A. Run mysql command and create database 14](#_Toc1270188)

[B. Create rootadmin to access assignmentDB 15](#_Toc1270189)

[C. Login as rootadmin and create table 15](#_Toc1270190)

[D. Insert data into UID and login table 16](#_Toc1270191)

[Section 4 Create IoT “Thing” 17](#_Toc1270192)

[A. Sign into the AWS IoT Console 17](#_Toc1270193)

[B. Create and register your “Thing” 17](#_Toc1270194)

[C. Create a policy for your Thing 20](#_Toc1270195)

[D. Attach Security Policy and Thing to your Certificate 20](#_Toc1270196)

[E. Copy the REST API endpoint of your “Thing” 22](#_Toc1270197)

[Section 5 Install the AWS Python SDK 23](#_Toc1270198)

[A. Install the AWS Python Library 23](#_Toc1270199)

[Section 6 Create tables in DynamoDB 24](#_Toc1270200)

[A. Access DynamoDB 24](#_Toc1270201)

[B. Create RBM\_temp 24](#_Toc1270202)

[C. Create RBM\_rfidRecords 25](#_Toc1270203)

[Section 7 Setting up AWS rules 28](#_Toc1270204)

[A. Create Rule for Temperature and Humidity Topic 28](#_Toc1270205)

[B. Create Rule for RFID Topic 31](#_Toc1270206)

[Section 8 Setting up DHT11 (Room One & Two) 34](#_Toc1270207)

[A. Frizting diagram (DHT11 ONLY) 34](#_Toc1270208)

[B. Connecting the components 34](#_Toc1270209)

[C. dht.py 34](#_Toc1270210)

[Section 9 Setting up RFID/NFC reader (Room One & Two) 37](#_Toc1270211)

[A. Fritzing Diagram (MFRC522 ONLY) 37](#_Toc1270212)

[B. Connecting the components 37](#_Toc1270213)

[C. Enable SPI and Prepare the MFRC522 libraries 38](#_Toc1270214)

[D. read\_display.py 39](#_Toc1270215)

[Section 10 Setting up Buzzer (Room One & Two) 42](#_Toc1270216)

[A. Fritzing Diagram (Buzzer ONLY) 42](#_Toc1270217)

[B. Connecting the components 42](#_Toc1270218)

[Section 11 Setting up LED (Room One ONLY) 43](#_Toc1270219)

[A. Fritzing Diagram (LED ONLY) 43](#_Toc1270220)

[B. Connect the components 43](#_Toc1270221)

[C. lights.py 43](#_Toc1270222)

[Section 12 Setting up Motion Sensor Light (Room Two Only) 45](#_Toc1270223)

[A. Fritzing Diagram (Motion sensor & LED ONLY) 45](#_Toc1270224)

[B. Connecting the components 45](#_Toc1270225)

[C. Pushover 45](#_Toc1270226)

[D. testt.py 47](#_Toc1270227)

[Section 13 Code for Web Application 49](#_Toc1270228)

[A. Room One fritzing diagram 49](#_Toc1270229)

[B. Room Two fritzing diagram 49](#_Toc1270230)

[C. server.py 50](#_Toc1270231)

[D. dynamoDB.py 54](#_Toc1270232)

[E. jsonconverter.py 56](#_Toc1270233)

[F. Templates 57](#_Toc1270234)

[Index.html 58](#_Toc1270235)

[login.html 64](#_Toc1270236)

[RoomOne.html 66](#_Toc1270237)

[RoomTwo.html 73](#_Toc1270238)

[E. Run the program 79](#_Toc1270239)

# Section 1 Overview of the application

* 1. Where have we uploaded our tutorial?

<https://www.hackster.io/NMuriele/random-security-company-101-997b28>

* 1. Why have we chosen to upload to this site?

We have chosen to upload our project on Hackster as it is the world’s fastest growing developer community for learning and building hardware. They are notable for their neat navigation system, and the projects are filtered according to the hardware platform you wish to use or learn. This will help the users save time learning how to navigate the site as it is already organised. Apart from this, Raspberry Pi projects are in abundance in Hackster, making it a sensible choice.

* 1. What have we uploaded?

We have uploaded the step-by-step tutorial, fritzing diagram and the full source code.

* 1. What is the application about?

This application is meant to monitor two temperature and humidity sensitive rooms where authorised staff need to scan a card to enter. A web application will display the temperature and humidity data collected from the room in a table and graph. The card ID as well as the date and time of successful entries will also be displayed in a table.

As mentioned above, both rooms require the staff to scan their cards upon entry. An authorised card will make one short buzz to simulate a successful entry. Meanwhile, unauthorised cards, will make two short buzzes to simulate a failed entry.

Room One has LED light that can be controlled through the web application. On the other hand, Room Two has motion sensor controlled LED light. When motion is detected in the room, the light turns on and a push notification is sent to the admin’s smartphone. This is achieved using Pushover.

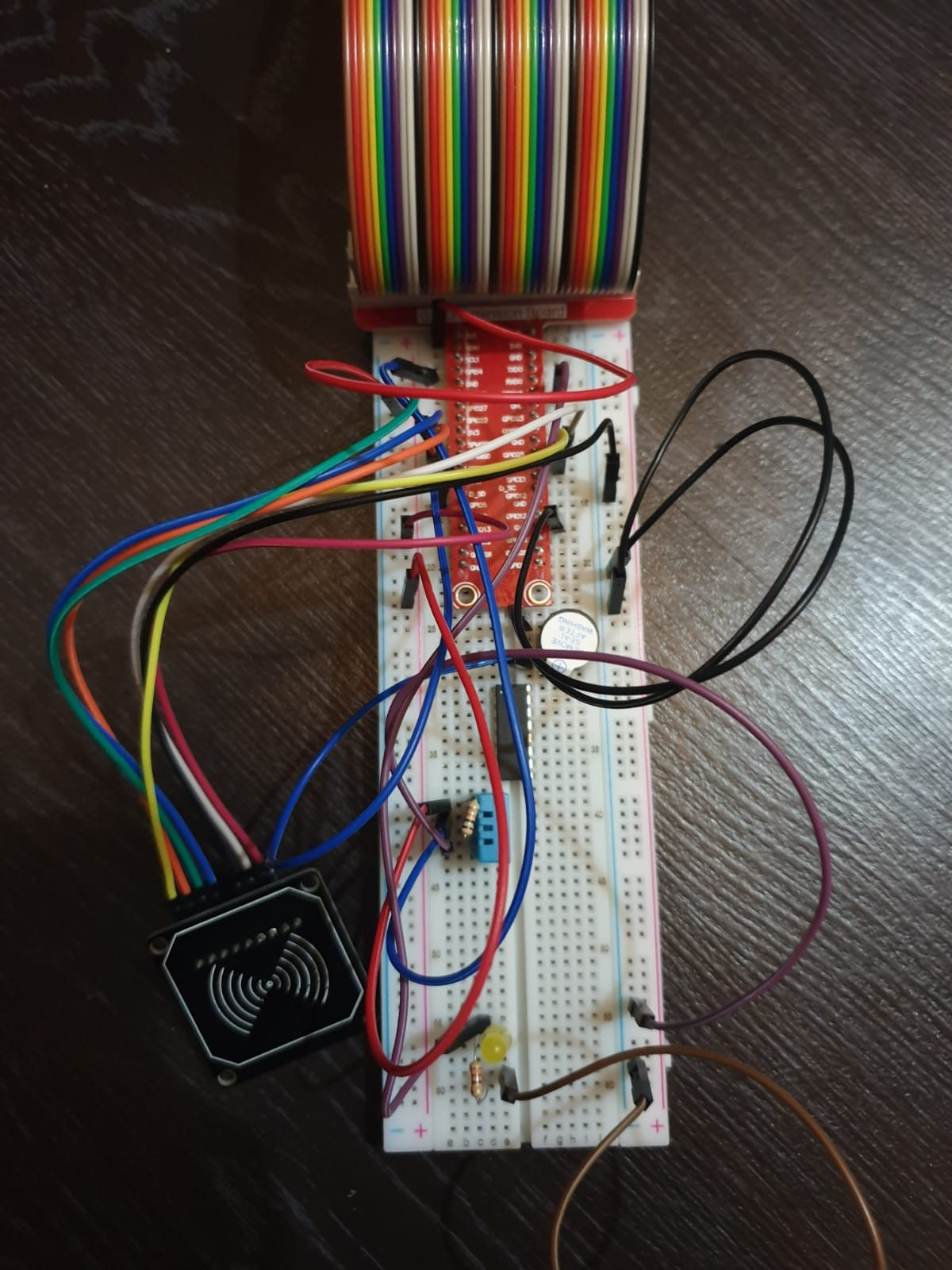
* 1. Summary of steps

|  |  |  |
| --- | --- | --- |
|  | Section | Description |
|  | Overview | What the project is about and what it will look like. |
|  | Hardware requirements | Provides overview of hardware required. |
|  | Setting up the database | Provides code to set up the required database and tables. |
|  | Create IoT “Thing” | We will sign into Amazon Web Services and set up the necessary items for the programme to run. |
|  | Install the AWS Python SDK | Before you can start coding an IoT app on the AWS Cloud, you will need to install the AWS Python software libraries first. |
|  | Create tables in DynamoDB | We will create the necessary tables in DynamoDB for the programme to run and be accessed from cloud. |
|  | Setting up AWS rules | We will create and configure a rule to send the data received from a device to DynamoDB |
| Section 8-13 will show you how to set up the application | | |
|  | Setting up DHT11 | Connect the hardware components and write python code for DHT11. |
|  | Setting up RFID/NFC Reader | Connect the hardware components for RFID/NFC reader. Code to read the RFID is also included. |
|  | Setting up Buzzer | Connect the hardware components for Buzzer and write python code for Buzzer. |
|  | Setting up LED | Connect the hardware components for LED. |
|  | Setting up Motion Sensor Light | Connect the hardware components for Motion Sensor and write python code for Motion Sensor. |
|  | Code for Web Application | Writing the code for server and front end of the web application. |

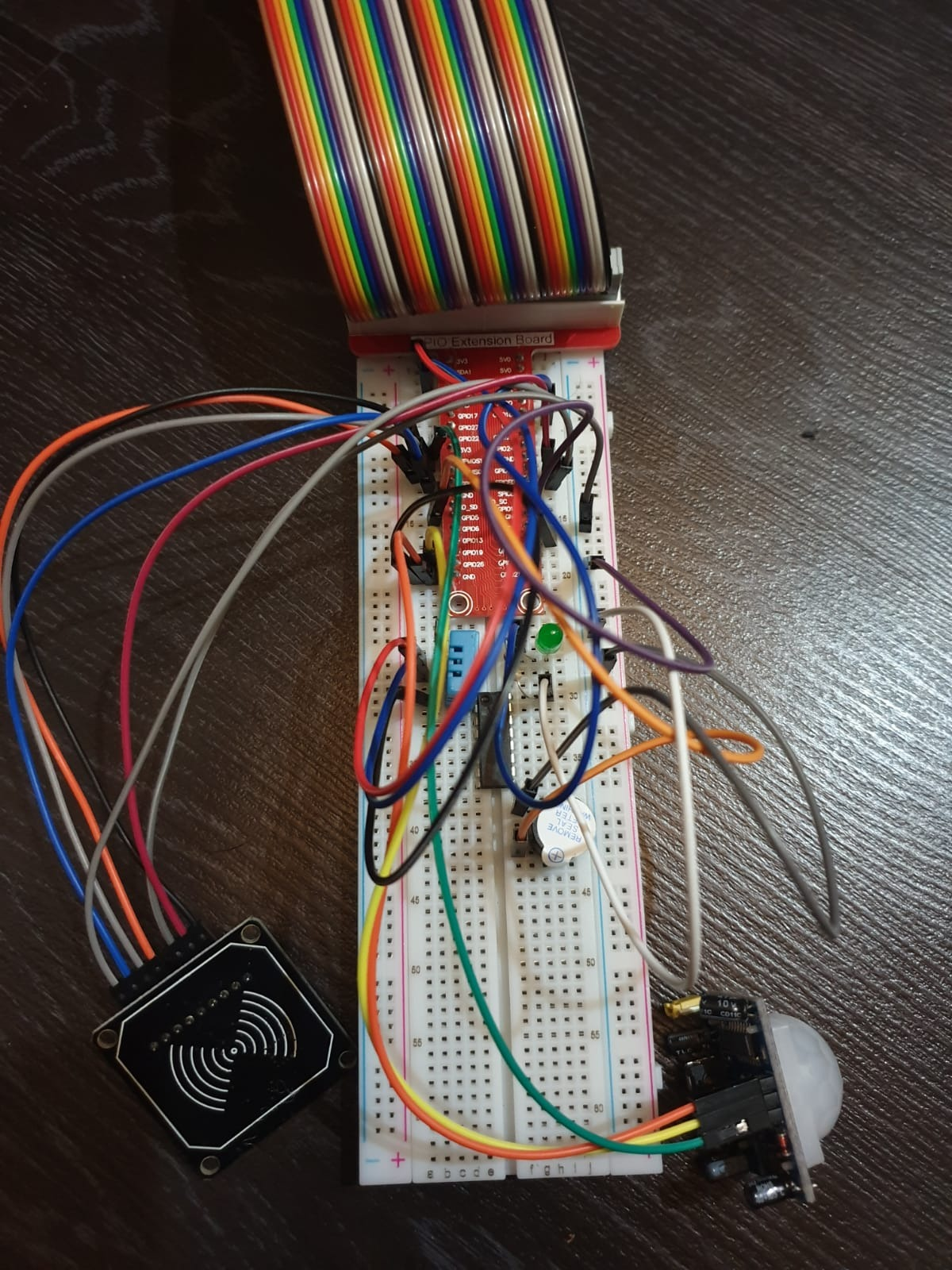
**Note: we will be doing the connection part by part, therefore please do not remove any connections done when moving on to the next section**

* 1. What does the final setup look like?

### Room One

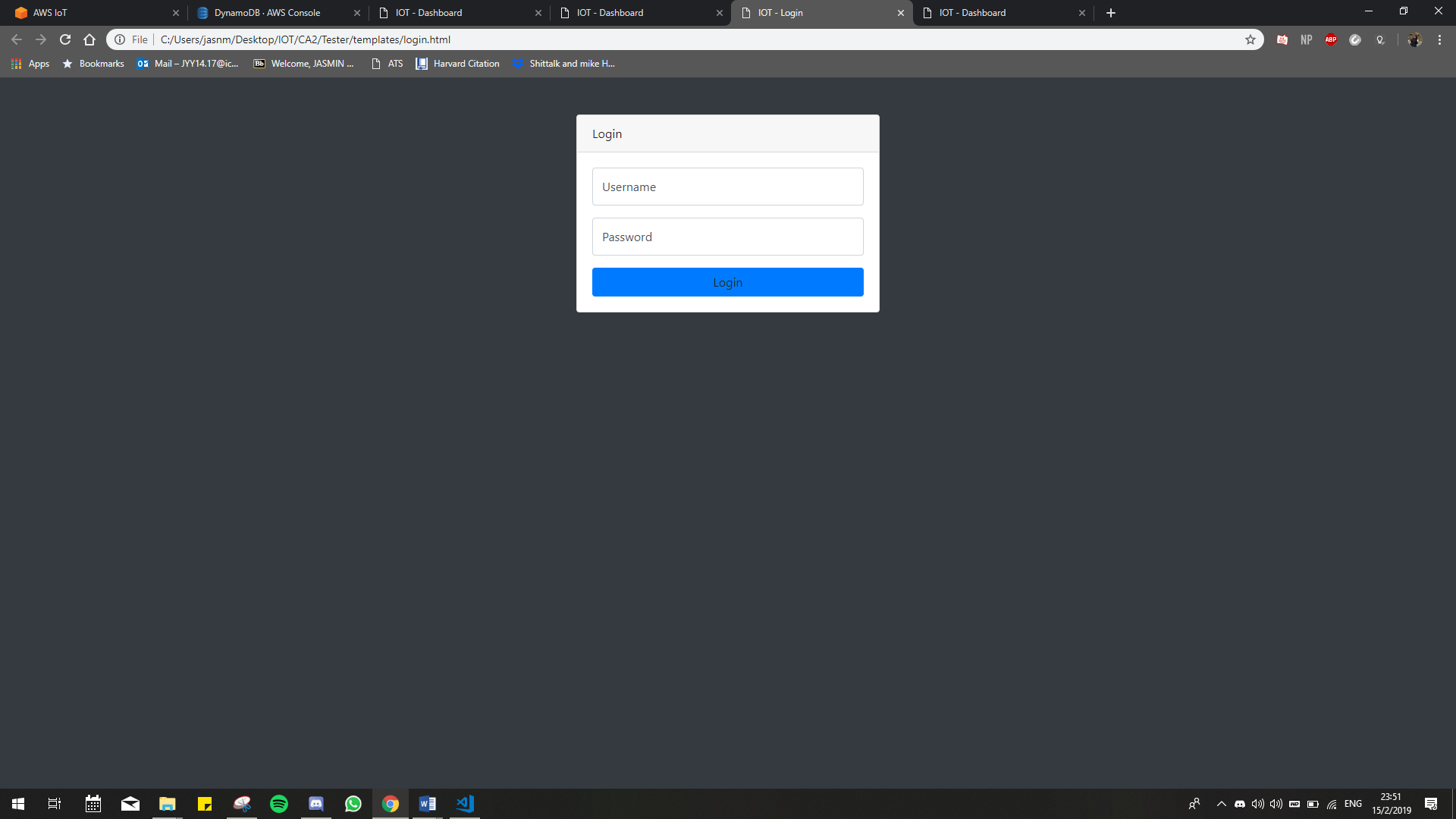


### Room Two

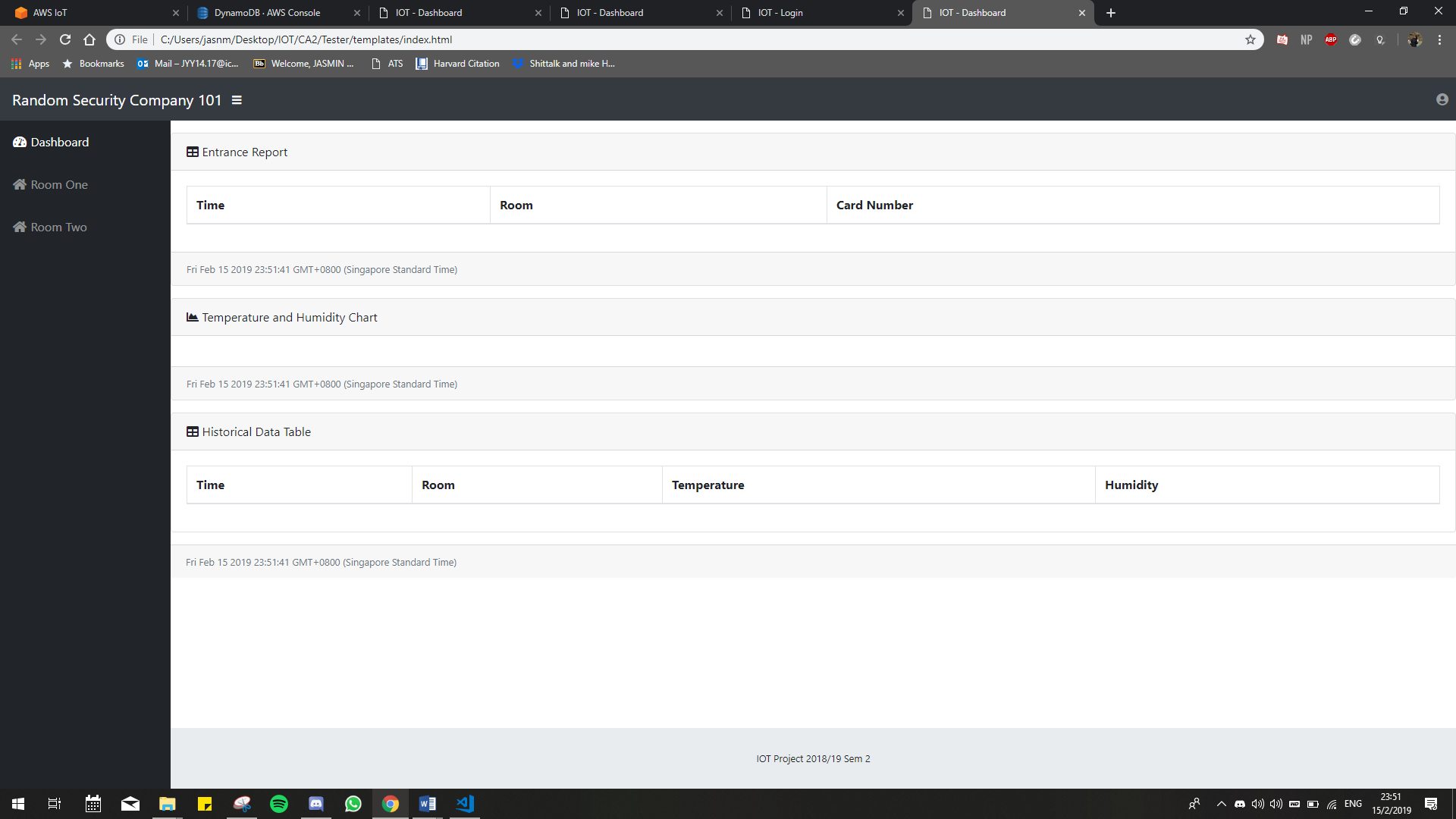


* 1. How does the web application look like?

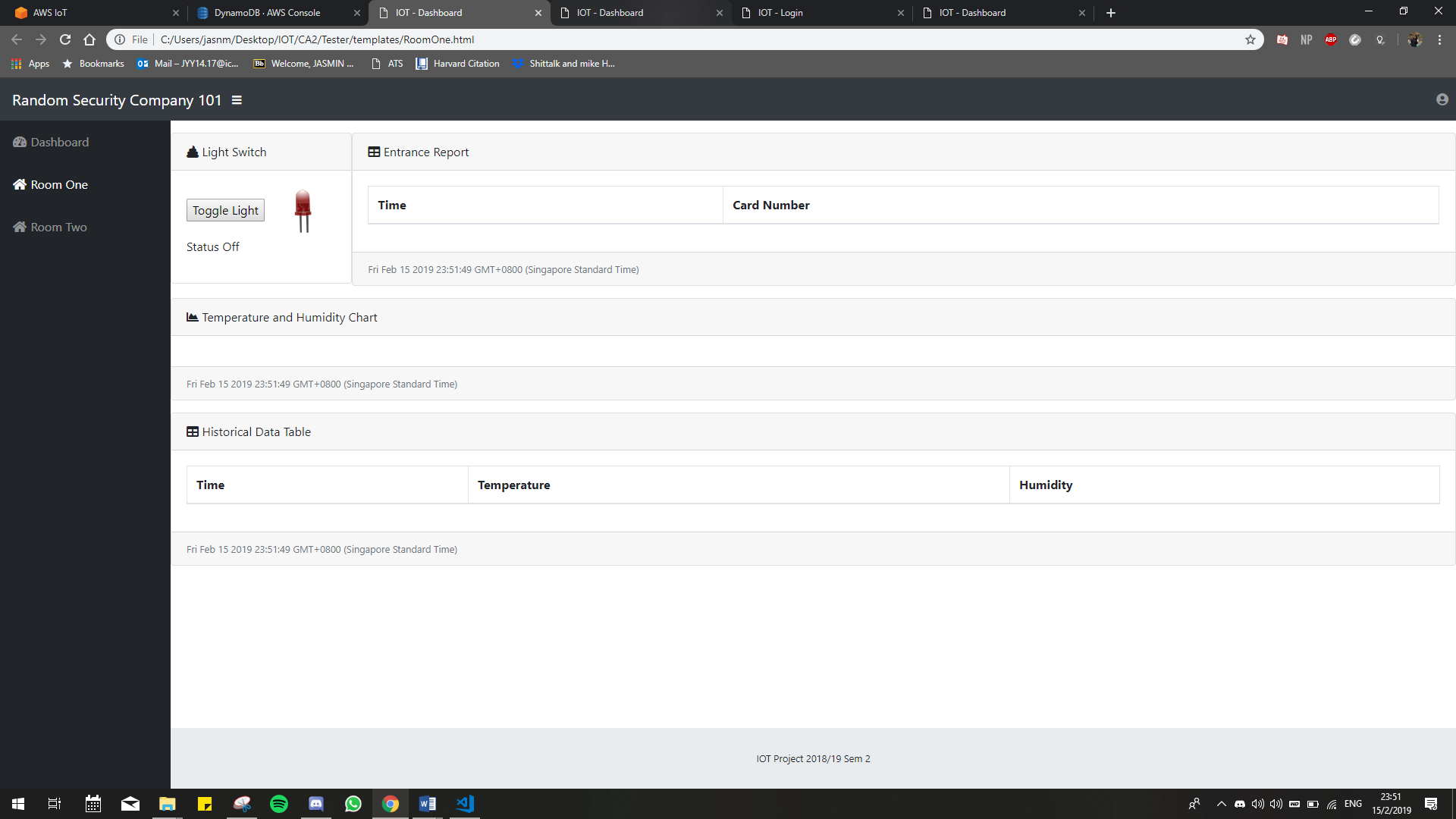
### login.html



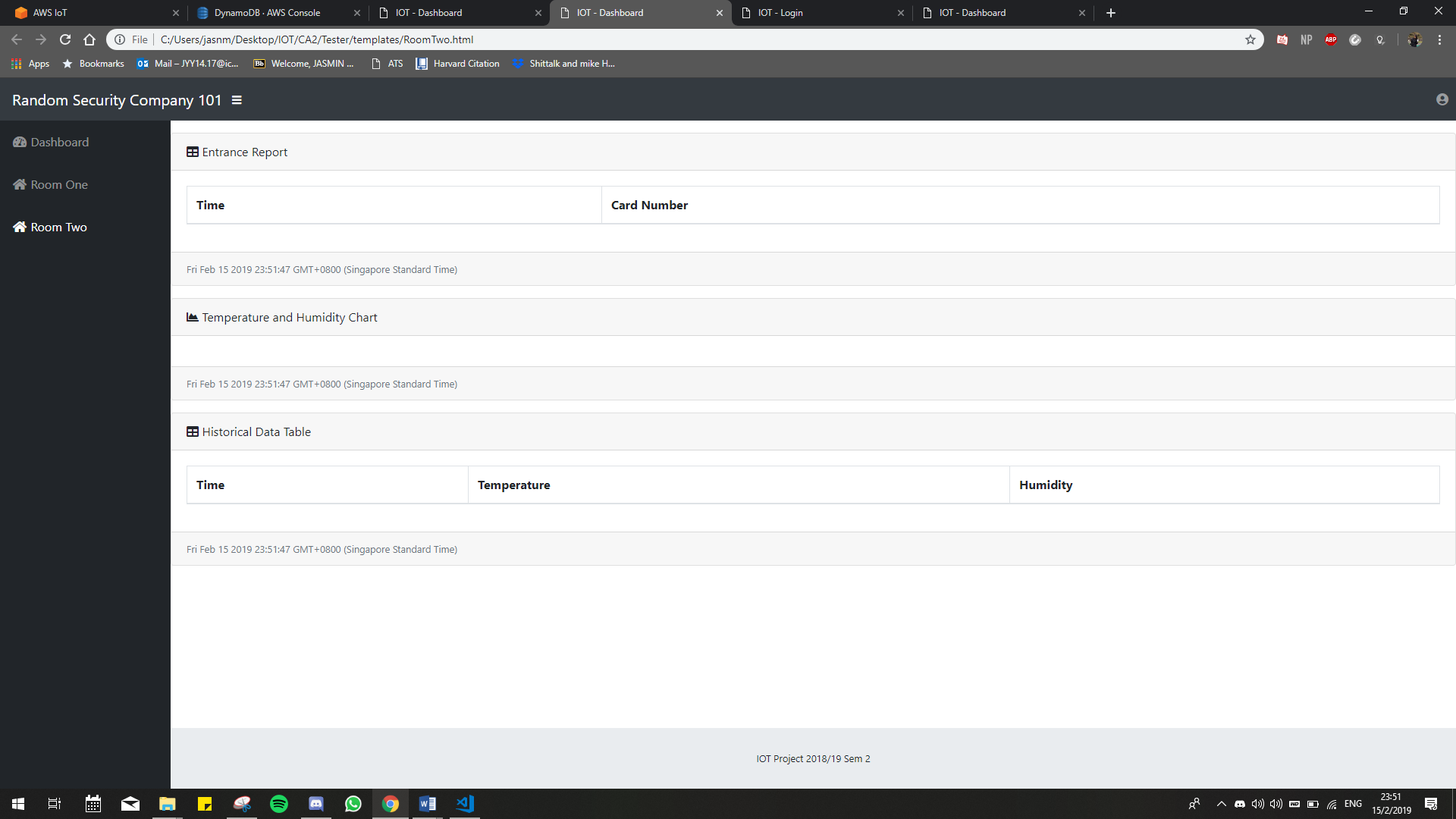
### Index.html



### RoomOne.html

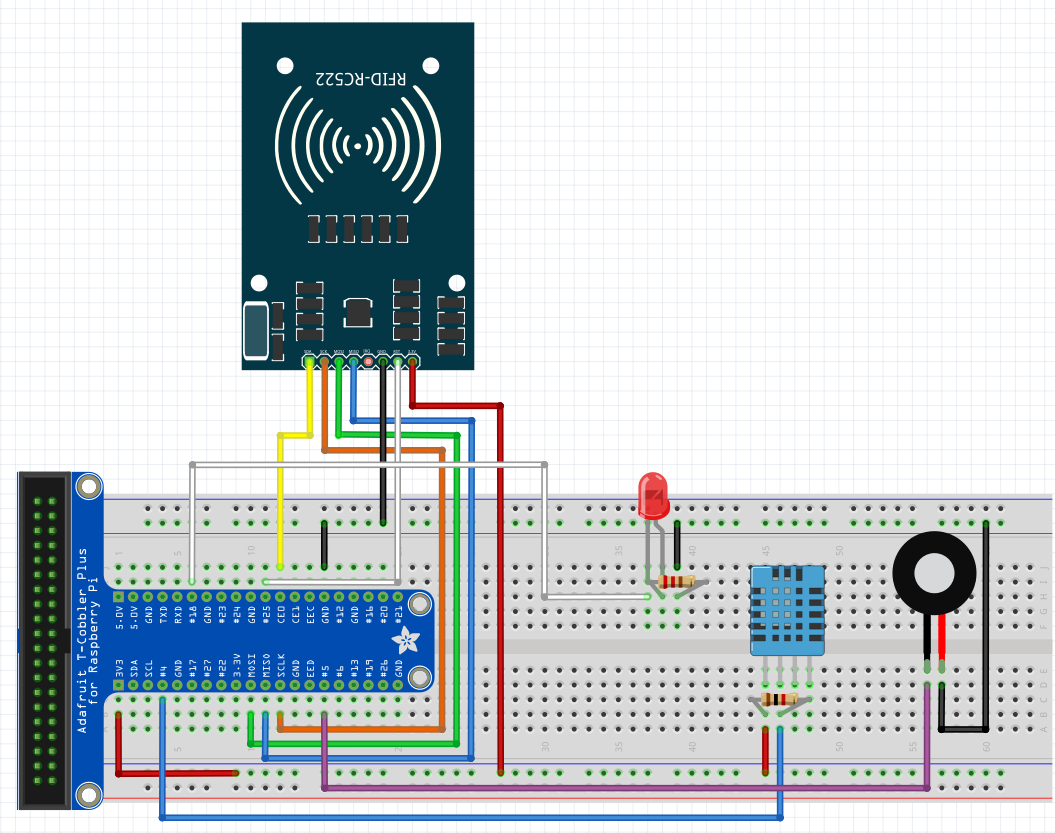


### RoomTwo.html

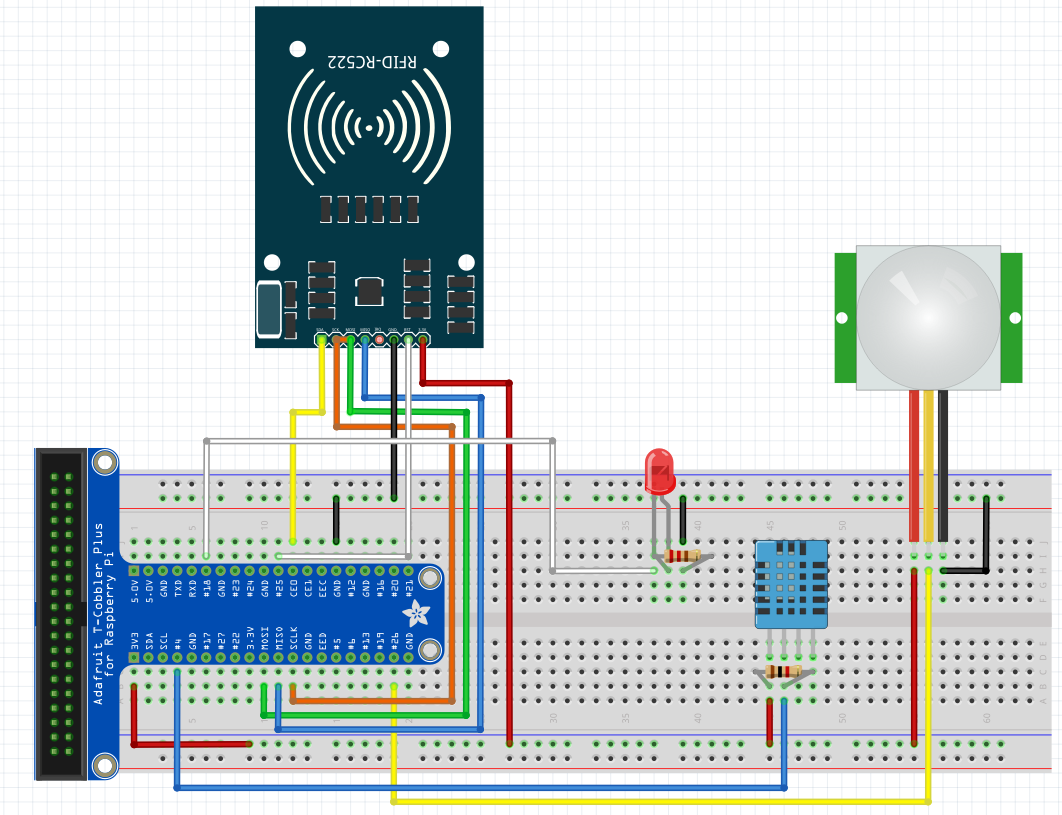


* 1. How does the final Fritzing diagram look like?

### Room One



### Room Two



# Section 2 Hardware requirements

Hardware checklist

### DHT11 Temperature and Humidity Sensor

|  |  |
| --- | --- |
| * Low-cost digital temperature and humidity sensor * Uses capacitive humidity sensor and a thermistor to measure the surrounding air * Produces digital signal on the data pin, no analog input pins needed * Less accurate and has a smaller range compared to the DHT22 but is small and less expensive * Good for 20-80% humidity readings with 5% accuracy * Good for 0-50°C temperature readings ±2°C accuracy   All information above provided by adafruit.com | A picture containing indoor, floor, table, sitting  Description automatically generated  All images credited to adafruit.com |

### Resistors (10K Ω and 220/330 Ω)

|  |  |
| --- | --- |
| * Carbon film * Through-hole axial type * 1/4 Watt max dissipation * 300V max voltage * 5% precision resistors   All information above provided by adafruit.com  **For DHT11 Sensor**: 10K Ω  Due to the data line being bidirectional, a pullup resistor is needed to connect the data line and VCC line of the sensor.  **For LED**: 220 Ω or 330 Ω  Resistor is always required for LEDs as they will draw more power and cause the Raspberry Pi to burn out.  **Note**: I use 220 Ω, 330 Ω looks similar to 220 Ω just that the first two bands are orange instead of red | A picture containing indoor, sitting  Description automatically generated 10k Ω A picture containing indoor, sitting, table  Description automatically generated 220 Ω  All images credited to adafruit.com |

### LED

|  |  |
| --- | --- |
| * 5mm diameter * 660 nm wavelength * 1.85-2.5V Forward Voltage, at 20mA current * 250 mcd typical brightness   All information above provided by adafruit.com  **Things to note**:  Longer leg 🡪 anode (power)  Shoerter leg 🡪 cathode (ground) | A picture containing floor, dog, indoor, sky  Description automatically generated  All images credited to adafruit.com |

### Buzzer

|  |  |
| --- | --- |
| * An audio signaling device found in circuits to create a beeping noise. * Active buzzers and passive ones. For our lab, we will use active buzzers as they are a lot simpler to use. * Do not need a resistor * Typically has 2 pins:   + VOUT – Connect this to a GPIO pin to control its value   + GND – Connect this to ground |  |

### RFID/NFC MFRC522 Card Reader Module

|  |  |
| --- | --- |
| * Contactless reader module * Low power consumption * Small and low cost   All information above provided by sgbotic.com  Radio-frequency identification (RFID) uses electromagnetic fields to identify and track objects that have been tagged.  Examples of uses include EZlink cards, contactless credit cards etc. | A circuit board  Description automatically generated  All images credited to hobbytronics.co.uk |

### PIR Motion Sensor

|  |  |
| --- | --- |
| * Sensors nearby movement * The sensor detects the pattern of infrared energy in its surroundings. If the pattern changes, the sensor outputs a high signal. * has a trimpot on its back for adjusting its time delay and sensitivity * 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 |  |

# Section 3 Setting up the database using command-line

Before setting up the breadboard, we will have to set up the database that stores the login values and the RFID card ID values that will be used in the project.

1. Run mysql command and create database

|  |  |
| --- | --- |
|  | Open the Terminal window and run the following command:  mysql -uroot -pdmitiot |
|  | Once you have connected to MySQL, you can create the database using the following command:  mysql> create database assignmentDB; |
|  | You should see the following message if you are successful:  Query Ok, 1 row affected (0.00 sec) |

1. Create rootadmin to access assignmentDB

|  |  |
| --- | --- |
|  | Use the following command to use the database that you created earlier:  mysql> USE assignmentDB |
|  | Create a user with the password “root” that has access to the database  mysql> CREATE USER ‘rootadmin’@’localhost’ IDENTIFIED by ‘root’;  mysql> GRANT ALL PRIVILEGES ON assignmentDB.\* TO ‘rootadmin’@’localhost’; mysql> FLUSH PRIVILEGES;  mysql> quit |
|  | You should see the following message if you are successful:  Query Ok, 1 row affected (0.00 sec) |
|  | Exit the MySQL prompt using Ctrl+D and close the Terminal window |

1. Login as rootadmin and create table

|  |  |
| --- | --- |
|  | Open a Terminal window and type the following command to log in mySQL. When prompted with the password, type in the password “root”  mysql -urootadmin -p |
|  | Use the following command to use the database that you created earlier:  mysql> USE assignmentDB |
|  | Type in the following code to create a table named login that has 4 fields, id, Name, Username and Password of type int, varchar(100) and varchar(30) for the last two.  mysql> CREATE TABLE dht (id int NOT NULL AUTO\_INCREMENT, Username varchar(30), Password varchar(30), PRIMARY KEY (id)); |
|  | You should see the following message if you are successful:  Query Ok, 1 row affected (0.00 sec) |
|  | Create another table to store the RFID values names UID that has 2 fields, ID and Admin of type varchar(25) and varchar(3).  mysql> CREATE TABLE UID (ID varchar(25) NOT NULL, Admin varchar(3), PRIMARY KEY (ID)); |
|  | You should see the following message if you are successful:  Query Ok, 1 row affected (0.00 sec) |

1. Insert data into UID and login table

In order for the authentication process to work, at least two rows of data (one row admin and one not) needs to be in the UID table and one row of data must be added into the login table before running the project.

|  |  |
| --- | --- |
|  | Get the card ID of two RFID cards and store them in the database.  mysql> INSERT into UID (ID, Admin) VALUES (RFIDno, ‘yes’), (RFIDno, ‘no’);  EXAMPLE:  Replace ‘RFIDno’ with the card ID that you have, eg. [136, 4, 0, 75, 199] and  [136, 4, 16, 170, 54] are the card IDs that I have, thus the code looks like this:  mysql> INSERT into UID (ID, Admin) VALUES (‘[136, 4, 0, 75, 199]’, ‘yes’), (‘[136, 4, 16, 170, 54]’, ‘no’); |
|  | You should see the following message if you are successful:  Query Ok, 1 row affected (0.00 sec)  Records: 2 Duplicates: 0 Warnings: 0 |
|  | Come up with a username and a password for logging into the web application and add it to the database using this command  mysql> INSERT into login (Name, Username, Password) VALUES (‘INSERT USERNAME’, ‘INSERT PASSWORD’); |
|  | You should see the following message if you are successful:  Query Ok, 1 row affected (0.00 sec)  Records: 1 Duplicates: 0 Warnings: 0 |
|  | Exit the MySQL prompt using Ctrl+D and close the Terminal window |

# Section 4 Create IoT “Thing”

Now, we will sign into Amazon Web Services and set up the necessary items for the programme to run.

1. Sign into the AWS IoT Console

Assuming that you already have a functioning AWS account that can be used.

|  |  |
| --- | --- |
|  | Sign in with your AWS console at [https://aws.amazon.com](https://aws.amazon.com/) |
|  | In the AWS dashboard, search for IoT Core. |
|  | On the Welcome page, choose Get Started. |

1. Create and register your “Thing”

|  |  |  |
| --- | --- | --- |
|  | In the navigation pane on the left, choose Manage |  |
|  | On the page, (if you have not made a thing before) it should say “You don’t have any things yet”, choose “Register a thing”.  If you have created a thing before, choose Create.  A thing here represents a device whose status or data is stored in the AWS cloud. The Thing Shadows is the stats of the device, e.g is it “on” or “off” etc  Our Thing will be the RPi that is used to host our server. |  |
|  | Select “Create a single thing” | |
|  | Give your thing a name. | |
|  | Select “Create certificate” | |
|  | Create a folder on your **Raspberry Pi** named “RSC” and download the certificates to that folder.  **Note: The file you download this to must be in the same file as all of your code.**    For the root CA download, right click the first one and select “save link as”:    Afterwards, be sure to click the activate button before clicking done. | |

1. Create a policy for your Thing

|  |  |  |
| --- | --- | --- |
|  | In the navigation pane on the left, choose Secure and Policies  Once in the tab, click “Create”. |  |
|  | On the Create a policy page, key in the following configuration:   |  |  | | --- | --- | | Field | Type this in | | Name | MyRaspberryPiSecurityPolicy | | Action | iot:\* | | Resource ARN | \* | | Allow | Checked | | |
|  | Click “Create”  You now have a Ssecurity Policy that allows all access to IoT Core services. | |

1. Attach Security Policy and Thing to your Certificate

In this section, you will attach both your security policy and your Thing to your certficate that you created previously.

|  |  |  |
| --- | --- | --- |
|  | In the navigation pane on the left, choose Secure and Policies. |  |
|  | Find the Certificate starting with the same string as the one you created and downloaded earlier.  When I created the certificate earlier, it started with “b4cf9a654c”.  So I will select the certificate starting with the same string and click “attach policy” under the “Actions” button. |  |
|  | Select “MyRaspberryPiSecurityPolicy” that was created earlier and click “Attach” |  |
|  | When I created the certificate earlier, it started with “b4cf9a654c”.  So I will select the certificate starting with the same string and click “attach Thing” under the “Actions” button. |  |
|  | Find the Thing you created earlier and select it. |  |

1. Copy the REST API endpoint of your “Thing”

|  |  |  |
| --- | --- | --- |
|  | Click “Manage->Things” and choose “MyRaspberryPi”.  On the next screen, choose “Interact” |  |
|  | Copy and paste the REST API endpoint into a Notepad.  You will need this value later. | |

# Section 5 Install the AWS Python SDK

By now, you have completed the first part of the process to connect your device to AWS cloud.

In the next few sections, you will begin to code your app. However, before you can start coding an IoT app on the AWS Cloud, you will need to install the AWS Python software libraries first.

1. Install the AWS Python Library

|  |  |
| --- | --- |
|  | Install the AWS Python library with this command  sudo pip install AWSIoTPythonSDK |

# Section 6 Create tables in DynamoDB

Here, we will create the necessary tables in DynamoDB for the programme to run.

1. Access DynamoDB

|  |  |
| --- | --- |
|  | Navigate to the AWS console and search for DynamoDB. Select DynamoDB |
| 2) | Select Tables from this page |

1. Create RBM\_temp

|  |  |
| --- | --- |
| 1) | Next, click on create table to create a table |
| 2) | Create the table with the following details   |  |  | | --- | --- | | Field | Type this in | | Table Name | RBM\_temp | | Primary Key | | | Partition Key | deviceid | | Sort Key | datetimeid | |
| 3) | You should end with this table. |

1. Create RBM\_rfidRecords

|  |  |
| --- | --- |
| 1) | Next, click on create table to create a table |
| 2) | Create the table with the following details   |  |  | | --- | --- | | Field | Type this in | | Table Name | RBM\_rfidRecords | | Primary Key | | | Partition Key | deviceid | | Sort Key | datetime |     RBM\_rfidRecords |
| 3) | You should end with this table.    RBM\_rfidRecords |

# Section 7 Setting up AWS rules

The AWS IoT rules engine listens for incoming MQTT messages that match a rule. When a matching message is received, the rule takes some action with the data in the MQTT message. In this step, you will create and configure a rule to send the data received from a device to DynamoDB.

1. Create Rule for Temperature and Humidity Topic

|  |  |
| --- | --- |
|  | In the AWS IoT console, in the left navigation pane, choose Act, then “Create a rule” and enter the following data. |
| 1. Ss | Select Add action |
|  | Select the second option ‘Split message into multiple columns of a DynamoDB table (DynamoDBv2) |
|  | Click on configure action and enter the following data.    Once you’ve entered the data, click “add action” |
|  | This should look like what it’ll look like |

1. Create Rule for RFID Topic

|  |  |  |
| --- | --- | --- |
|  | In the AWS IoT console, in the left navigation pane, choose Act, then “Create a rule” and enter the following data. |  |
|  | Select an action | |
|  | Select the second option ‘Split message into multiple columns of a DynamoDB table (DynamoDBv2) | |
|  | Click on configure action and enter the following data.    RBM\_rfidRecords  Click “add action” once you’re done. | |
|  | This is what its suppose to look like | |

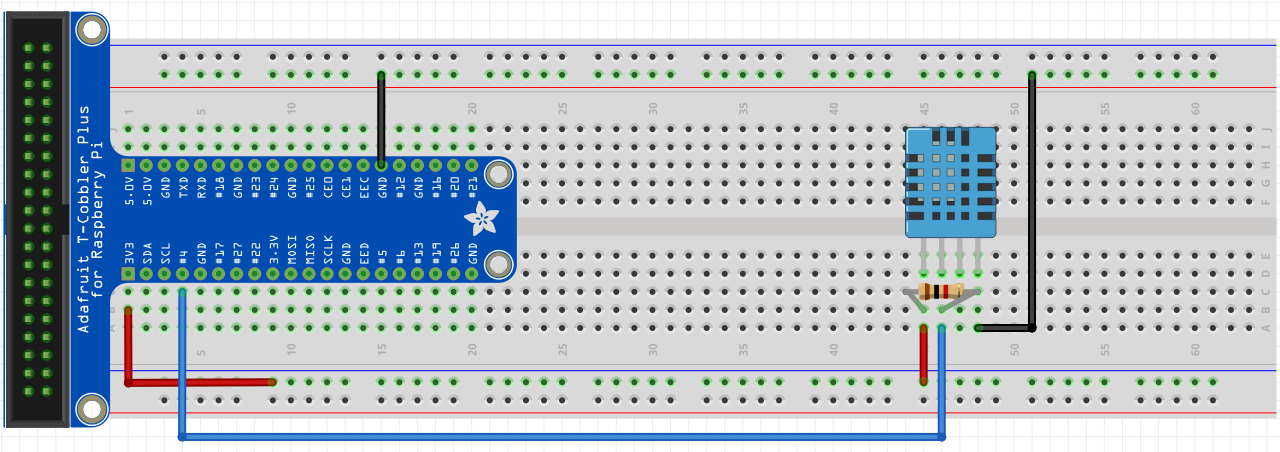
# Section 8 Setting up DHT11 (Room One & Two)

In this section we will be setting up the DHT11 Sensor as well as the code needed to run it.

The DHT11 sensor is needed in order to gather the data that is displayed on the web application.

1. Frizting diagram (DHT11 ONLY)

GROUND RAIL



POWER

GPIO 4

POWER

10k Ω RESISTOR

GROUND

1. Connecting the components

|  |  |
| --- | --- |
|  | Plug the DHT11 sensor into the breadboard at the very end.  This is to ensure that it does not interfere with the other components that will be plugged in later. |
|  | Using a red wire, connect 3.3v to the power rail as shown.  Using a black wire, connect the ground pin to the power rail on the oppsite side as shown. This will be refered to as the ground rail. |
|  | The DHT11 sensor has 4 pins. Add a 10k Ω resistor to the VCC and DATA pin.  Simply connect the pins as shown starting from the left-most pin (closest to the T-Cobbler Kit):   |  |  |  | | --- | --- | --- | | DHT11 Sensor | Raspberry Pi | Jumper Colour | | VCC | Power Rail | Red | | DATA | GPIO 4 | Blue | | NC |  |  | | GND | Ground Rail | Black | |

1. dht.py

Now, we shall write a Python script that will read the current temperature and humidity values from the sensor and store them in the database table “dht” that was created earlier.

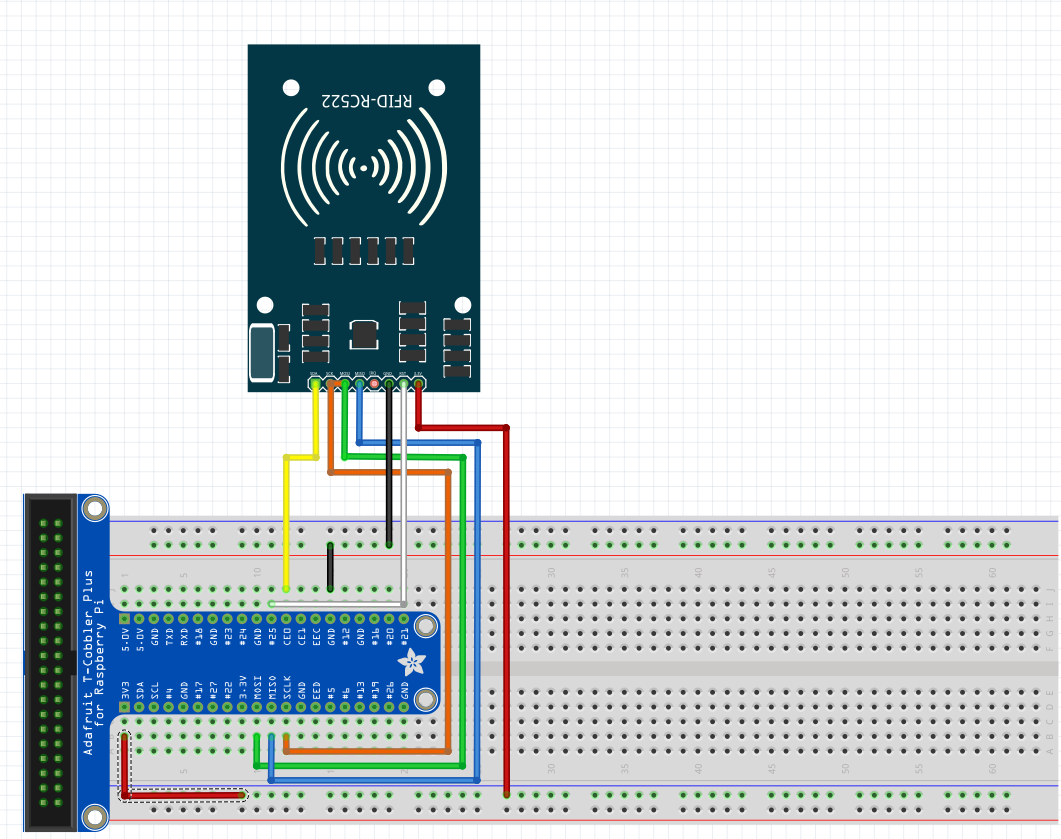
|  |  |
| --- | --- |
|  | Change the directory to RSC, the same file your certificates are in  cd ~/RSC |
|  | Create a python script **dht.py** with the code below  sudo nano ~/RSC/dht.py |
|  | Please take note of the indentations!  Remember to replace the tokens with the ones that you have.  # Import SDK packages  from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient  from time import sleep  import Adafruit\_DHT  # Custom MQTT message callback  def customCallback(client, userdata, message):      print("Received a new message: ")      print(message.payload)      print("from topic: ")      print(message.topic)      print("--------------\n\n")  # to write to AWS dynamoDB  host = ""#Insert your host  rootCAPath = "" #Insert your Amazon Root file  certificatePath = "" #Insert your certificate file  privateKeyPath = "" #Insert your private key file  my\_rpi = AWSIoTMQTTClient("RBMOnebasicPubSub")  my\_rpi.configureEndpoint(host, 8883)  my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)  my\_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing  my\_rpi.configureDrainingFrequency(2) # Draining: 2 Hz  my\_rpi.configureConnectDisconnectTimeout(10) # 10 sec  my\_rpi.configureMQTTOperationTimeout(5) # 5 sec  # Connect and subscribe to AWS IoT  my\_rpi.connect()  my\_rpi.subscribe("RBM/sensors/temp", 1, customCallback)  sleep(2)  # Publish to the same topic in a loop forever  loopCount = 0  while True:  pin = 4  humidity, temperature = Adafruit\_DHT.read\_retry(11, pin)  loopCount = loopCount+1  message = {}  message["deviceid"] = "RBM\_RoomOne" #add room number here  import datetime as datetime  now = datetime.datetime.now()  message["datetimeid"] = now.isoformat()  message["temp"] = temperature  message["hum"] = humidity  import json  my\_rpi.publish("RBM/sensors/temp", json.dumps(message), 1)  sleep(5) |
|  | Run the program  sudo python lights.py |
|  | Repeat the steps for the Raspberry Pi that is hosting Room 2 |

**Note: we will be doing the connection part by part, therefore please do not remove any connections done when moving on to the next section**

# Section 9 Setting up RFID/NFC reader (Room One & Two)

In this section, we will be setting up the MFRC522 card reader. This reader will be playing a part in the authentication that an authoriased staff is accessing the room.

1. Fritzing Diagram (MFRC522 ONLY)



POWER

MFRC522

SDA🡪CEO

SCK 🡪SCLK

MOSI

MISO

GROUND RAIL

GPIO 25

1. Connecting the components

The 9 pins on the MFRC522 have their respective pin names on the back.

Simply connect the pins accordingly:

|  |  |  |
| --- | --- | --- |
| MFRC522 | Raspberry Pi | Jumper Colour |
| SDA | CEO | Yellow |
| SCK | SCLK | Orange |
| MOSI | MOSI | Green |
| MISO | MISO | Blue |
| IDR |  |  |
| GND | Ground Rail | Black |
| RST | GPIO 25 | White |
| 3.3V | Power Rail | Red |
| 5V |  |  |

If the connection has been done correctly, a green light will be lit.

1. Enable SPI and Prepare the MFRC522 libraries

If you have never worked with an MFRC522 card reader before, there are some steps that you need to take before the card reader can work. However, if you have worked with it before please skip this part.

|  |  |
| --- | --- |
|  | Run  sudo raspi-config  Choose the 5th menu item and enable SPI  A screenshot of a cell phone  Description automatically generated  A screenshot of a cell phone  Description automatically generated |
|  | Modify the /boot/config.txt to enable SPI  sudo nano /boot/config.txt  Ensure that the following lines are included in config.txt  device\_tree\_param=spi=on  dtoverlay=spi-bcm2835 |
|  | Install the Python development libraries  sudo apt-get install python-dev |
|  | 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 |
|  | Clone the MFRC522-python library to your home folder  cd ~  git clone https://github.com/mxgxw/MFRC522-python.git |
|  | Copy the MFRC522.py into the folder that you are working in:  sudo cp ~/MFRC522-python/\*.py ~/RSC  If you don’t do this, the reader will not work. |

1. read\_display.py

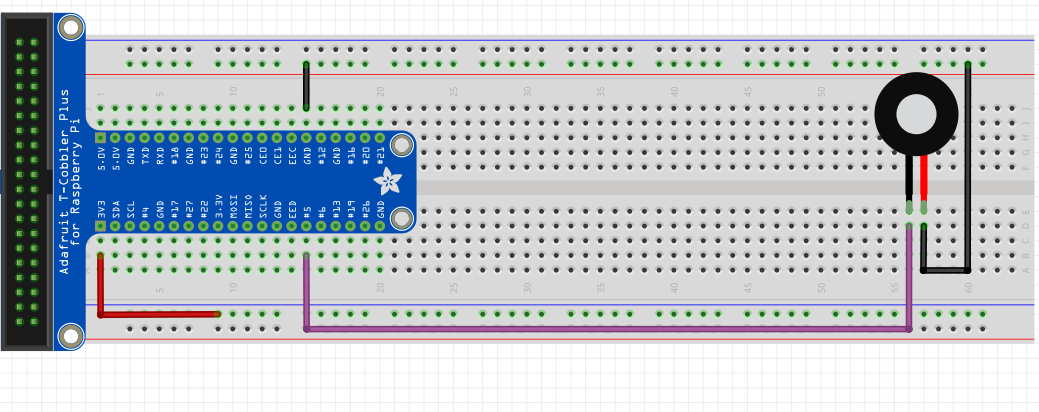
|  |  |
| --- | --- |
|  | Change the directory to RSC  cd ~/RSC |
|  | Create a python script **read\_display.py** with the code below  sudo nano ~/RSC/read\_display.py |
|  | Please take note of the indentations!  Remember to replace the tokens with the ones that you have.  import RPi.GPIO as GPIO  import MFRC522  import signal  import datetime  from gpiozero import Buzzer  from time import sleep  import mysql.connector  from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient  #authorisation  bz = Buzzer(5)  uid = None  continue\_reading = True  GPIO.setmode(GPIO.BCM)  #connect to database  mydb = mysql.connector.connect(  host="localhost",  user="rootadmin",  passwd="root",  database="assignmentDB"  )  mycursor = mydb.cursor()  # to write to AWS dynamoDB  host = ""#Insert your host  rootCAPath = "" #Insert your rootCAPath  certificatePath = "" #Insert your certificatePath  privateKeyPath = "" #Insert your privateKeyPath  my\_rpi = AWSIoTMQTTClient("RBMOnebasicPubSub")  my\_rpi.configureEndpoint(host, 8883)  my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)  my\_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing  my\_rpi.configureDrainingFrequency(2) # Draining: 2 Hz  my\_rpi.configureConnectDisconnectTimeout(10) # 10 sec  my\_rpi.configureMQTTOperationTimeout(5) # 5 sec  # Connect and subscribe to AWS IoT  my\_rpi.connect()  # for authorisation  def checkUID(rfid):  mycursor.execute("SELECT \* FROM UID")  myresult = list(mycursor.fetchall())  for x in myresult:  if x[0] == rfid and x[1] == "yes":  return True  else:  return False  # Capture SIGINT for cleanup when the script is aborted  def end\_read(signal,frame):  global continue\_reading  print("Ctrl+C captured, ending read.")  continue\_reading = False  GPIO.cleanup()  GPIO.setmode(GPIO.BCM)  # end of authorisation functions  # authorisation  # Hook the SIGINT  signal.signal(signal.SIGINT, end\_read)  # Create an object of the class MFRC522  mfrc522 = MFRC522.MFRC522()  while continue\_reading:  # Scan for cards  (status,TagType) = mfrc522.MFRC522\_Request(mfrc522.PICC\_REQIDL)  # If a card is found  if status == mfrc522.MI\_OK:  # Get the UID of the card  (status,uid) = mfrc522.MFRC522\_Anticoll()  print(uid)  #Checks if he is an authorised person  if checkUID(str(uid)):  bz.on()  sleep(1)  bz.off()  message = {}  message["deviceid"] = "RBM\_RoomOne" #add room number  import datetime as datetime  now = datetime.datetime.now()  message["datetime"] = now.isoformat()  message["RFID"] = ("{}").format(uid)  import json  my\_rpi.publish("RBM/sensors/rfid", json.dumps(message), 1)  sleep(2)  else:  bz.on()  sleep(0.5)  bz.off()  sleep(0.5)  bz.on()  sleep(0.5)  bz.off()  # currentDT = (datetime.datetime.now()).strftime("%Y-%m-%d %H:%M:%S")  # filepath = ('/home/pi/CA2/pictures/{0}.jpg').format(currentDT)  # camera.capture(filepath)  # sleep(2)    #end of authorisation |
|  | Run the program  sudo python read\_display.py |
|  | Repeat the steps for the Raspberry Pi that is hosting Room 2 |

**Note: we will be doing the connection part by part, therefore please do not remove any connections done when moving on to the next section**

# Section 10 Setting up Buzzer (Room One & Two)

In this section, we will be setting up the Buzzer that will signal whether the RFID card is accepted with one beep being the card accepted and two beeps being the card is rejected. The code is integrated with the code for the RFID.

1. Fritzing Diagram (Buzzer ONLY)



1. Connecting the components

Connect the Buzzer with the RPi as follows:

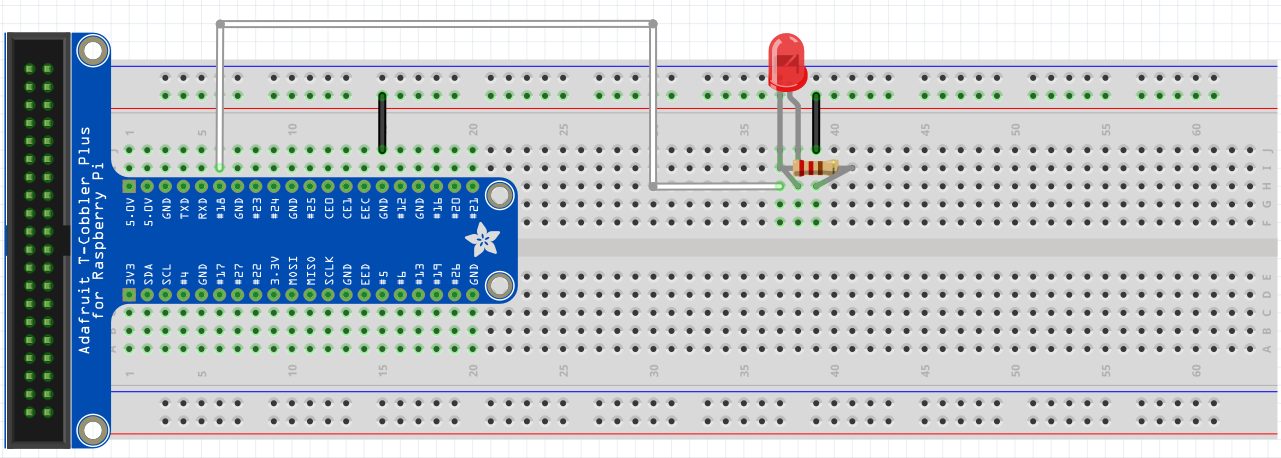
|  |  |  |
| --- | --- | --- |
| Jumper color | PIR pin | RPi pin |
| Black | GND | GND |
| Blue | VOUT | BCM5 |

**Note: we will be doing the connection part by part, therefore please do not remove any connections done when moving on to the next section**

# Section 11 Setting up LED (Room One ONLY)

In this section, we are setting up the MQTT controlled LED in Room One.

1. Fritzing Diagram (LED ONLY)



1. Connect the components

|  |  |
| --- | --- |
|  | Plug the LED into the breadboard.  Place the long end nearer to the T-Cobbler Kit. |
|  | Plug in the 220 Ω resistor into the breadboard.  One end of the resistor should be in the same column as the short leg of the LED. |
|  | Using a white cable, connect the long leg of the LED to GPIO 17. |
|  | Using a black cable, connect the free end of the resistor to the power rail that is connected to the ground (Ground rail). |

1. lights.py

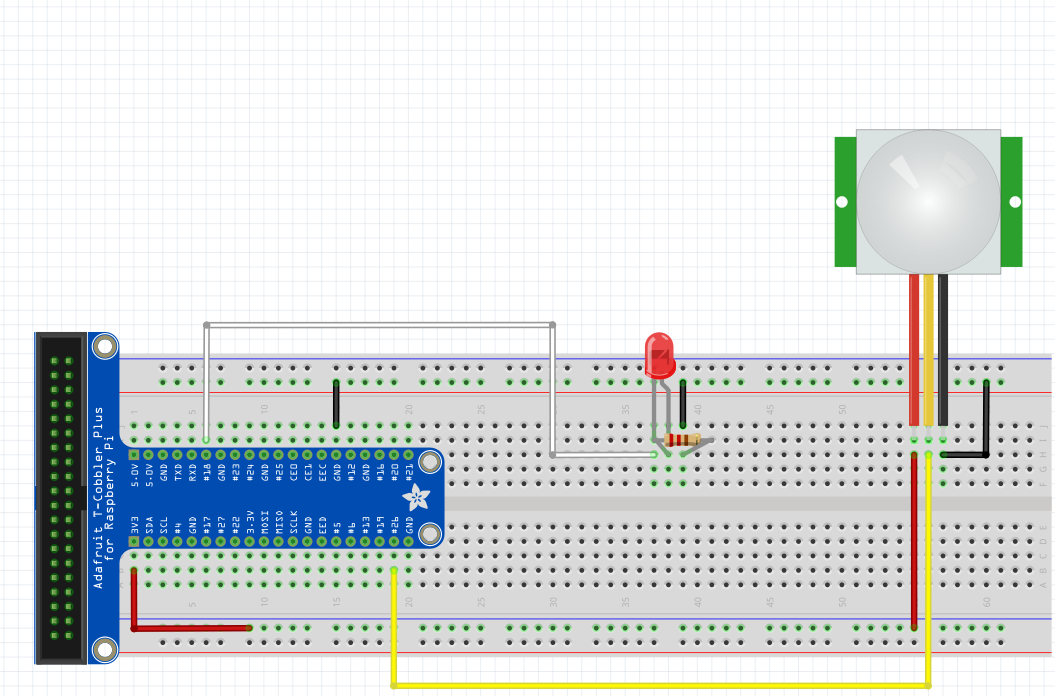
|  |  |
| --- | --- |
|  | Change the directory to RSC  cd ~/RSC |
|  | Create a python script **lights.py** with the code below  sudo nano ~/RSC/lights.py |
|  | Please take note of the indentations!  Remember to replace the tokens with the ones that you have.  from gpiozero import LED  from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient  from time import sleep  import json  light = LED(18)  while True:  try:  # Custom MQTT message callback  def customCallback(client, userdata, message):  print("Received a new message: ")  print(message.payload)  print("from topic: ")  print(message.topic)  print("--------------\n\n")    command = json.loads(message.payload)  print(command["command"])  if command["command"] == "on":  light.on()  elif command["command"] == "off":  light.off()  # to write to AWS dynamoDB  host = "" #Insert your host  rootCAPath = "" #Insert your rootCAPath  certificatePath = "" #Insert your certificationPath  privateKeyPath = "" #Insert your privateKeyPath  my\_rpi = AWSIoTMQTTClient("RBMOnebasicPubSub")  my\_rpi.configureEndpoint(host, 8883)  my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)  my\_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing  my\_rpi.configureDrainingFrequency(2) # Draining: 2 Hz  my\_rpi.configureConnectDisconnectTimeout(10) # 10 sec  my\_rpi.configureMQTTOperationTimeout(5) # 5 sec  # Connect and subscribe to AWS IoT  my\_rpi.connect()  my\_rpi.subscribe("RBM/LightSwitch", 1, customCallback)  except:  import sys  print(sys.exc\_info()) |
|  | Run the program  sudo python lights.py |

**Note: we will be doing the connection part by part, therefore please do not remove any connections done when moving on to the next section**

# Section 12 Setting up Motion Sensor Light (Room Two Only)

In this section, we will be setting up how to connect the motion sensor to the raspberry pi.

1. Fritzing Diagram (Motion sensor & LED ONLY)



1. Connecting the components

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 |

1. Pushover

Here, we will configure Pushover which will send push notifications to your smartphone when motion is detected in the room.

|  |  |
| --- | --- |
|  | Go to <http://pushover.net/> and sign up for an account. |
|  | Remember the user key provided as it will be used later. |
|  | Create an application. |
|  | Remember the API token that was generated when the application is successfully created as it will be used for later. |
|  | Install the Pushover application in your smartphone. <https://pushover.net/clients>  Then, log in with the push over account you have created in Step a).  Add your smartphone as one of the devices. |
|  | You can send a test notification to your smartphone device. |
|  | If the test notification is successful, your phone would show something like this: |

1. testt.py

|  |  |
| --- | --- |
|  | Change the directory to RSC  cd ~/RSC |
|  | Create a python script **testt.py** with the code below  sudo nano ~/RSC/testt.py |
|  | Please take note of the indentations!  Remember to replace the necessary tokens and strings with the ones you have.  from gpiozero import LED, MotionSensor  import time  from time import sleep  import httplib, urllib  pir = MotionSensor(26, sample\_rate=5,queue\_len=1)  light = LED(18)  #Pushover API things  PUSH\_TOKEN = "" #Enter your app token  PUSH\_USER = "" #Enter your user token  PUSH\_MSG = "Motion is detected. Someone has entered the room."  def sendPush( msg ):  conn = httplib.HTTPSConnection("api.pushover.net:443")  conn.request("POST", "/1/messages.json",  urllib.urlencode({  "token": PUSH\_TOKEN,  "user": PUSH\_USER,  "message": msg,  }), { "Content-type": "application/x-www-form-urlencoded" })  conn.getresponse()  return  while True:  light.off()  old\_time = time()  pir.wait\_for\_motion()  new\_time = time()  if new\_time - old\_time > 1:  print(PUSH\_MSG)  sendPush(PUSH\_MSG)  light.on()  sleep(10)  old\_time = new\_time  pir.wait\_for\_no\_motion()  light.off()  new\_time = time()  print("Room is empty.") |
|  | Run the program  sudo python testt.py |

# Section 13 Code for Web Application

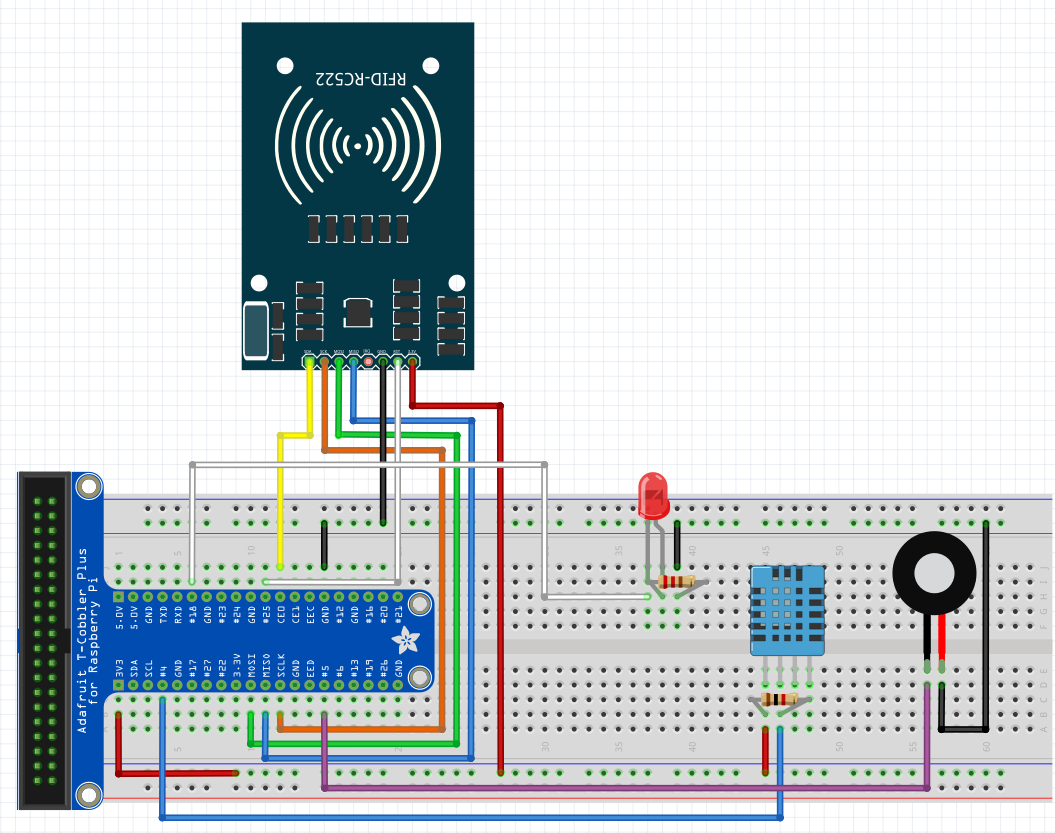
In this section, we will be setting up the code for the web applcation.

After completing section 3 – 12, you should a fully working database locally called assignmentDB that contains two tables – login and UID.

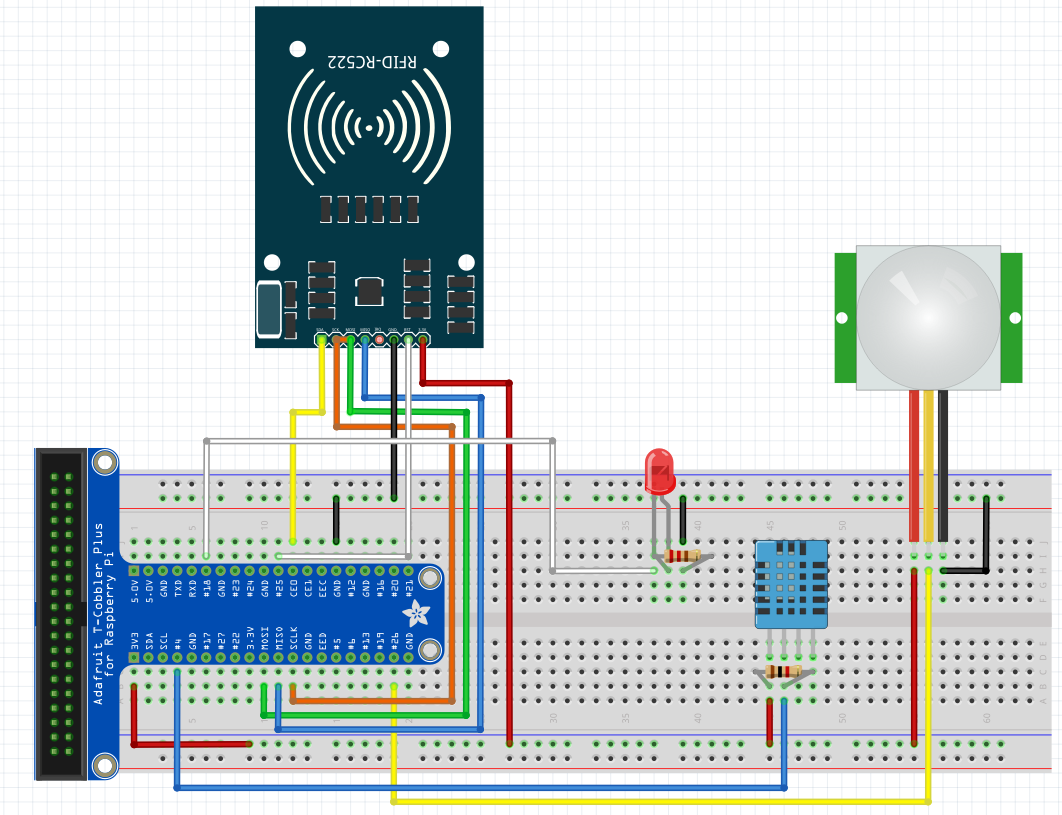
The UID table should have at least two entries of tag IDs, one admin and one not. The login table should also have at least one entry.

You should also have set up the DynamoDB tables RBM\_rfidRecords and RBM\_temp and the necessary components needed for the project and it should look something like this:

1. Room One fritzing diagram



1. Room Two fritzing diagram



1. server.py

|  |  |
| --- | --- |
|  | Change the directory to RSC  cd ~/RSC |
|  | Create a python script **server.py** with the code below  sudo nano ~/RSC/server.py |
|  | Take note of the indentations!  Replace my\_hp with your own number.  # Complusory line, to get flask files  import RPi.GPIO as GPIO  import signal  from time import sleep  import mysql.connector  from flask import Flask, render\_template,request,jsonify, session  import json  import mysql.connector  import dynamoDB  import jsonconverter as jsonc  from gpiozero import LED  from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient  #connect to database  mydb = mysql.connector.connect(  host="localhost",  user="rootadmin",  passwd="root",  database="assignmentDB"  )  mycursor = mydb.cursor()  # to write to AWS dynamoDB  host = ""#Insert your host  rootCAPath = "" #Insert your rootCAPath  certificatePath = "" #Insert your certificatePath  privateKeyPath = "" #Insert your privateKeyPath  my\_rpi = AWSIoTMQTTClient("RBMOnebasicPubSub")  my\_rpi.configureEndpoint(host, 8883)  my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)  my\_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing  my\_rpi.configureDrainingFrequency(2) # Draining: 2 Hz  my\_rpi.configureConnectDisconnectTimeout(10) # 10 sec  my\_rpi.configureMQTTOperationTimeout(5) # 5 sec  # Connect and subscribe to AWS IoT  my\_rpi.connect()  # function and var declaration  light = LED(18)  # mandatory line, create flask ojbect  app = Flask(\_\_name\_\_)  #api functions  @app.route("/api/login", methods = ['POST', 'GET'])  def get\_login():  message = ""  errmsg = ""  if request.method == 'POST':  user = request.form["user"]  password = request.form["password"]  print(user + " " + password)  try:  query = "SELECT \* FROM login"  mycursor.execute(query)  res = mycursor.fetchall()  for x in res:  if x[2] == user and x[3] == password:  session['authorise'] = True  message = "true"  print(authorise)  data = {"message" : message, "error" : errmsg}  return jsonify(data)  else:  message = "false"  errmsg = "Username or password is incorrect"  except:  print("Error fetching values from database")  import sys  print(sys.exc\_info())  data = {"message" : message, "error" : errmsg}  return jsonify(data)  @app.route("/api/tempHum",methods=['POST','GET'])  def dash\_getdata():  if request.method == 'POST':  try:  data = {'data': jsonc.data\_to\_json(dynamoDB.getDashTemp())}  return jsonify(data)  except:  import sys  print(sys.exc\_info())  @app.route("/api/rfid",methods=['POST','GET'])  def dash\_getcarddata():  if request.method == 'POST':  try:  data = {'data': jsonc.data\_to\_json(dynamoDB.getDashRfid())}  return jsonify(data)  except:  import sys  print(sys.exc\_info())  @app.route("/api/tempHum1",methods=['POST','GET'])  def dash\_getdata1():  if request.method == 'POST':  try:  data = {'data': jsonc.data\_to\_json(dynamoDB.getR1temp())}  return jsonify(data)  except:  import sys  print(sys.exc\_info())  @app.route("/api/rfid1",methods=['POST','GET'])  def dash\_getcarddata1():  if request.method == 'POST':  try:  data = {'data': jsonc.data\_to\_json(dynamoDB.getR1rfid())}  return jsonify(data)  except:  import sys  print(sys.exc\_info())  @app.route("/api/tempHum2",methods=['POST','GET'])  def dash\_getdata2():  if request.method == 'POST':  try:  data = {'data': jsonc.data\_to\_json(dynamoDB.getR2temp())}  return jsonify(data)  except:  import sys  print(sys.exc\_info())  @app.route("/api/rfid2",methods=['POST','GET'])  def dash\_getcarddata2():  if request.method == 'POST':  try:  data = {'data': jsonc.data\_to\_json(dynamoDB.getR2rfid())}  return jsonify(data)  except:  import sys  print(sys.exc\_info())  @app.route("/api/Lightsoff", methods = ['POST', 'GET'])  def ACLoff():  message =""  msg = {}  try:  command = request.form["command"]  room = request.form["room"]  msg["room"] = room  msg["command"] = command  my\_rpi.publish("RBM/LightSwitch", json.dumps(msg), 1)  if command == "off":  message = "off"  else:  message = "on"    except:  message = "Error changing state"  import sys  print(sys.exc\_info()[0])  print(sys.exc\_info()[1])  data = {"message" : message}    return jsonify(data)  #template functions  @app.route("/")  def dashboard():  session.clear()  return render\_template('login.html')  @app.route("/index")  def index():  authorise = session.get('authorise')  if authorise:  return render\_template('index.html')  else:  return render\_template('login.html')  @app.route("/RoomOne")  def RoomOne():  authorise = session.get('authorise')  if authorise:  return render\_template('RoomOne.html')  else:  return render\_template('login.html')  @app.route("/RoomTwo")  def RoomTwo():  authorise = session.get('authorise')  if authorise:  return render\_template('RoomTwo.html')  else:  return render\_template('login.html')  app.secret\_key="12345ThisIsASecretKey"  app.run(debug=True, host='0.0.0.0', port=5000) |

1. dynamoDB.py

This python file will help take data from AWS dynamoDB for the web server.

|  |  |
| --- | --- |
|  | Change the directory to RSC  cd ~/RSC |
|  | Create a python script **dynamoDB.py** with the code below  sudo nano ~/RSC/dynamoDB.py |
|  | Take note of the indentations!  Replace my\_hp with your own number.  import boto3  from boto3.dynamodb.conditions import Key, Attr  dynamodb = boto3.resource('dynamodb', region\_name='us-west-2', aws\_access\_key\_id='', # enter your own AWS access key  aws\_secret\_access\_key= '') # enter your own AWS secret key  def getDashRfid():  try:  table = dynamodb.Table('RBM\_rfidRecords')  response = table.scan()  items = response['Items']  n=10 # limit to last 10 items  data = items[:n]  data\_reversed = data[::-1]  return data\_reversed  except:  import sys  print(sys.exc\_info())  def getDashTemp():  try:  table = dynamodb.Table('RBM\_temp')    response = table.scan()  items = response['Items']  n=10 # limit to last 10 items  data = items[:n]  data\_reversed = data[::-1]  return data\_reversed  except:  import sys  print(sys.exc\_info())  def getR1rfid():  try:  table = dynamodb.Table('RBM\_rfidRecords')  response = table.query(  KeyConditionExpression=Key('deviceid').eq('RBM\_RoomOne'), ScanIndexForward=False  )  items = response['Items']  n=10 # limit to last 10 items  data = items[:n]  data\_reversed = data[::-1]  return data\_reversed  except:  import sys  print(sys.exc\_info())  def getR1temp():  try:  table = dynamodb.Table('RBM\_temp')  response = table.query(  KeyConditionExpression=Key('deviceid').eq('RBM\_RoomOne'), ScanIndexForward=False  )  items = response['Items']  n=10 # limit to last 10 items  data = items[:n]  data\_reversed = data[::-1]  return data\_reversed  except:  import sys  print(sys.exc\_info())  def getR2rfid():  try:  table = dynamodb.Table('RBM\_rfidRecords')  response = table.query(  KeyConditionExpression=Key('deviceid').eq('RBM\_RoomTwo'), ScanIndexForward=False  )  items = response['Items']  n=10 # limit to last 10 items  data = items[:n]  data\_reversed = data[::-1]  return data\_reversed  except:  import sys  print(sys.exc\_info())  def getR2temp():  try:  table = dynamodb.Table('RBM\_temp')  response = table.query(  KeyConditionExpression=Key('deviceid').eq('RBM\_RoomTwo'), ScanIndexForward=False  )  items = response['Items']  n=10 # limit to last 10 items  data = items[:n]  data\_reversed = data[::-1]  return data\_reversed  except:  import sys  print(sys.exc\_info())  if \_\_name\_\_ == "\_\_main\_\_":  query\_data\_from\_dynamodb() |

1. jsonconverter.py

This python file will convert the data from dynamoDB to JSON format so that it is readable by the web server.

|  |  |
| --- | --- |
|  | Change the directory to RSC  cd ~/RSC |
|  | Create a python script **jsonconverter.py** with the code below  sudo nano ~/RSC/jsonconverter.py |
|  | Take note of the indentations!  Replace my\_hp with your own number.  from decimal import Decimal  import json  import datetime  import numpy  class GenericEncoder(json.JSONEncoder):    def default(self, obj):  if isinstance(obj, numpy.generic):  return numpy.asscalar(obj)  elif isinstance(obj, Decimal):  return str(obj)  elif isinstance(obj, datetime.datetime):  return obj.strftime('%Y-%m-%d %H:%M:%S')  elif isinstance(obj, Decimal):  return float(obj)  else:  return json.JSONEncoder.default(self, obj)  def data\_to\_json(data):  json\_data = json.dumps(data,cls=GenericEncoder)  print(json\_data)  return json\_data |

1. Templates

If you do not already have the templates folder given:

In order for the server to run and display a webpage, we need a html page.

|  |  |
| --- | --- |
|  | Create a folder called ‘templates’.  This file will contain all the html, css file and resources used for the webpage. |
|  | In the templates folder, create a html file called ‘index’.  This html file will be the main page of the web application. |
|  | Go to [https://startbootstrap.com/template-overviews/sb-admin/](https://startbootstrap.com/template-overviews/sb-admin/%20) and download the bootstrap.  This will serve as the foundation for the layout of the web application. |
|  | In the templates folder, create another folder called ‘static’.  In the static folder, create an ‘assets’ folder.  This folder will contain all the resources from the bootstrap and for images. |
|  | After downloading the bootstrap, copy the ‘css’, ‘js’, ‘scss’ and vendor folders from the bootstrap file to assets.    It should look like this. |
|  | In the assets folder, create an ‘img’ folder.  This folder will store all the images used in the web page. |
|  | Save these two images as png to the img folder with their respective names:   |  |  | | --- | --- | | A close up of a logo  Description automatically generated led\_red\_off | A close up of a logo  Description automatically generated led\_red\_on | |
|  | Copy the following codes into the respective files. |

### **Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>IOT - Dashboard</title>

<!-- Bootstrap core CSS-->

<link href="static/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom fonts for this template-->

<link href="static/assets/vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css">

<!-- Page level plugin CSS-->

<link href="static/assets/vendor/datatables/dataTables.bootstrap4.css" rel="stylesheet">

<!-- Custom styles for this template-->

<link href="static/assets/css/sb-admin.css" rel="stylesheet">

</head>

<body id="page-top">

<nav class="navbar navbar-expand navbar-dark bg-dark static-top">

<a class="navbar-brand mr-1" href="#">Random Security Company 101</a>

<button class="btn btn-link btn-sm text-white order-1 order-sm-0" id="sidebarToggle" href="#">

<i class="fas fa-bars"></i>

</button>

<ul class="navbar-nav ml-auto ml-md-0" style="position:absolute; right:0">

<li class="nav-item dropdown no-arrow">

<a class="nav-link dropdown-toggle" href="#" id="userDropdown" role="button" data-toggle="dropdown"

aria-haspopup="true" aria-expanded="false">

<i class="fas fa-user-circle fa-fw"></i>

</a>

<div class="dropdown-menu dropdown-menu-right" aria-labelledby="userDropdown">

<a class="dropdown-item" href="#" data-toggle="modal" data-target="#logoutModal">Logout</a>

</div>

</li>

</ul>

</nav>

<div id="wrapper">

<!-- Sidebar -->

<ul class="sidebar navbar-nav">

<li class="nav-item active">

<a class="nav-link" href="index">

<i class="fas fa-fw fa-tachometer-alt"></i>

<span>Dashboard</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" href="RoomOne">

<i class="fas fa-fw fa-home"></i>

<span>Room One</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" href="RoomTwo">

<i class="fas fa-fw fa-home"></i>

<span>Room Two</span>

</a>

</li>

</ul>

<div id="content-wrapper">

<!-- Data Table -->

<div class="card mb-3">

<div class="card-header">

<i class="fas fa-table"></i>

Entrance Report</div>

<div class="card-body">

<div class="table-responsive">

<table class="table table-bordered" id="dataTable2" width="100%" cellspacing="0">

<thead>

<tr>

<th>Time</th>

<th>Room</th>

<th>Card Number</th>

</tr>

</thead>

<tbody id="data2">

</tbody>

</table>

</div>

</div>

<div class="card-footer small text-muted" id="updates"></div>

</div>

<!-- Chart -->

<div class="card mb-3">

<div class="card-header">

<i class="fas fa-chart-area"></i>

Temperature and Humidity Chart</div>

<div class="card-body">

<div id="myChart"></div>

</div>

<div class="card-footer small text-muted" id="updates1"></div>

</div>

<!-- Data Table -->

<div class="card mb-3">

<div class="card-header">

<i class="fas fa-table"></i>

Historical Data Table</div>

<div class="card-body">

<div class="table-responsive">

<table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">

<thead>

<tr>

<th>Time</th>

<th>Room</th>

<th>Temperature</th>

<th>Humidity</th>

</tr>

</thead>

<tbody id="data">

</tbody>

</table>

</div>

</div>

</div>

<div class="card-footer small text-muted" id="updates2"></div>

</div>

<!-- /.container-fluid -->

<!-- Sticky Footer -->

<footer class="sticky-footer">

<div class="container my-auto">

<div class="copyright text-center my-auto">

<span>IOT Project 2018/19 Sem 2</span>

</div>

</div>

</footer>

</div>

<!-- /.content-wrapper -->

</div>

<!-- /#wrapper -->

<!-- Logout Modal-->

<div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>

<button class="close" type="button" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">×</span>

</button>

</div>

<div class="modal-body">Select "Logout" below if you are ready to end your current session.</div>

<div class="modal-footer">

<button class="btn btn-secondary" type="button" data-dismiss="modal">Cancel</button>

<a class="btn btn-primary" href="/">Logout</a>

</div>

</div>

</div>

</div>

<!-- Bootstrap core JavaScript-->

<script src="static/assets/vendor/jquery/jquery.min.js"></script>

<script src="static/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<!-- Core plugin JavaScript-->

<script src="static/assets/vendor/jquery-easing/jquery.easing.min.js"></script>

<!-- Custom scripts for all pages-->

<script src="static/assets/js/sb-admin.min.js"></script>

</body>

<!-- Script for google charts -->

<script type="text/javascript" src="https://code.jquery.com/jquery-3.2.1.js"></script>

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

<script type="text/javascript">

// Load the Visualization API and the corechart package.

google.charts.load('current', { 'packages': ['corechart'] });

// Set a callback to run when the Google Visualization API is loaded.

google.charts.setOnLoadCallback(init);

function init() {

setIntervalAndExecute(function () {

getData(function (myRtnA, data) {

if (myRtnA == 'success') {

create\_dataTable(data.data);

drawChart(data.data);

};

}),

getCardData(function (myRtnA, data) {

create\_Table(data.data);

}),

setupdate()

}, 3000);

}

function setupdate() {

now = new Date();

document.getElementById("updates").innerHTML = now;

document.getElementById("updates1").innerHTML = now;

document.getElementById("updates2").innerHTML = now;

}

function getData(callBack) {

jQuery.ajax({

url: "/api/tempHum",

type: 'POST',

beforeSend: function () {

},

success: function (data, textStatus, xhr) {

if (data) {

var myRtnA = 'success';

} else {

var myRtnA = 'fail';

}

return callBack(myRtnA, data);

},

error: function (xhr, textStatus, errorThrown) {

console.log("opps: " + textStatus + " : " + errorThrown);

var myRtnA = "Error";

return callBack(myRtnA); // return callBack() with myRtna

}

});

}

function getCardData(callBack) {

jQuery.ajax({

url: "/api/rfid",

type: 'POST',

beforeSend: function () {

},

success: function (data, textStatus, xhr) {

if (data) {

var myRtnA = 'success';

} else {

var myRtnA = 'fail';

}

return callBack(myRtnA, data);

},

error: function (xhr, textStatus, errorThrown) {

console.log("opps: " + textStatus + " : " + errorThrown);

var myRtnA = "Error";

return callBack(myRtnA); // return callBack() with myRtna

}

});

}

function create\_Table(data) {

var dataTable\_html = '';

var newdata = JSON.parse(data);

for (index = 0; index < newdata.length; index++) {

datetime = (newdata[index].datetime).substring(0, 19);

jsdatetime = new Date(Date.parse(datetime));

device = newdata[index].deviceid;

room = "Room 2";

if (device == "RBM\_RoomOne") {

room = "Room 1";

}

rfid = newdata[index].RFID;

var thisRow\_HTML = '<tr>';

thisRow\_HTML += '<td>';

thisRow\_HTML += jsdatetime;

thisRow\_HTML += '</td>';

thisRow\_HTML += '<td>';

thisRow\_HTML += room;

thisRow\_HTML += '</td>';

thisRow\_HTML += '<td>';

thisRow\_HTML += rfid;

thisRow\_HTML += '</td>';

thisRow\_HTML += '</tr>';

dataTable\_html += thisRow\_HTML;

};

document.getElementById("data2").innerHTML = dataTable\_html;

}

function drawChart(data) {

var dataTb = new google.visualization.DataTable();

dataTb.addColumn('string', 'Time');

dataTb.addColumn('number', 'Temperature');

dataTb.addColumn('number', 'Humidity');

var newdata = JSON.parse(data);

for (index = 0; index < newdata.length; index++) {

datetime = (newdata[index].datetimeid).substring(0, 19);

jsdatetime = new Date(Date.parse(datetime));

jstime = jsdatetime.toLocaleTimeString();

temp = parseInt(newdata[index].temp);

hum = parseInt(newdata[index].hum);

dataTb.addRows([[jstime, temp, hum]])

}

var chart = new google.visualization.LineChart(

document.getElementById('myChart'));

chart.draw(dataTb, {

legend: { position: 'top' }, vAxis: { baseline: 0 },

colors: ['#ffb0b0', '#98f6b3']

});

}

function create\_dataTable(data) {

var dataTable\_html = '';

var newdata = JSON.parse(data);

for (index = 0; index < newdata.length; index++) {

datetime = (newdata[index].datetimeid).substring(0, 19);

jsdatetime = new Date(Date.parse(datetime));

jstime = jsdatetime.toLocaleTimeString();

device = newdata[index].deviceid;

room = "Room 2";

if (device == "RBM\_RoomOne") {

room = "Room 1";

}

temp = parseInt(newdata[index].temp);

hum = parseInt(newdata[index].hum);

var thisRow\_HTML = '<tr>';

thisRow\_HTML += '<td>';

thisRow\_HTML += jstime;

thisRow\_HTML += '</td>';

thisRow\_HTML += '<td>';

thisRow\_HTML += room;

thisRow\_HTML += '</td>';

thisRow\_HTML += '<td>';

thisRow\_HTML += temp;

thisRow\_HTML += '</td>';

thisRow\_HTML += '<td>';

thisRow\_HTML += hum;

thisRow\_HTML += '</td>';

thisRow\_HTML += '</tr>';

dataTable\_html += thisRow\_HTML;

};

document.getElementById("data").innerHTML = dataTable\_html;

}

function setIntervalAndExecute(fn, t) {

fn();

return (setInterval(fn, t));

}

</script>

</html>

### **login.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>IOT - Login</title>

<!-- Bootstrap core CSS-->

<link href="static/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom fonts for this template-->

<link href="static/assets/vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css">

<!-- Custom styles for this template-->

<link href="static/assets/css/sb-admin.css" rel="stylesheet">

</head>

<body class="bg-dark">

<div class="container">

<div class="card card-login mx-auto mt-5">

<div class="card-header">Login</div>

<div class="card-body">

<div id="message"></div>

<form>

<div class="form-group">

<div class="form-label-group">

<input type="text" id="inputUser" class="form-control" placeholder="Username" required="required" autofocus="autofocus">

<label for="inputUser">Username</label>

</div>

</div>

<div class="form-group">

<div class="form-label-group">

<input type="password" id="inputPass" class="form-control" placeholder="Password" required="required">

<label for="inputPass">Password</label>

</div>

</div>

<a class="btn btn-primary btn-block" onclick="login()">Login</a>

</form>

</div>

</div>

</div>

<!-- Bootstrap core JavaScript-->

<script src="static/assets/vendor/jquery/jquery.min.js"></script>

<script src="static/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<!-- Core plugin JavaScript-->

<script src="static/assets/vendor/jquery-easing/jquery.easing.min.js"></script>

</body>

<script>

function login(){

var username = document.getElementById('inputUser').value;

var password = document.getElementById('inputPass').value;

checklogin(username, password);

}

function checklogin(user, pass){

var togglemessage = document.getElementById('message');

jQuery.ajax({

url: "/api/login",

type: 'POST',

data: {user:user, password:pass},

success: function(data,textStatus, jqXHR){

console.log(data.message);

if (data.message === 'true'){

window.location.replace('/index');

} else if (data.message === 'false'){

console.log(data.error);

togglemessage.innerHTML = "<h5>" + data.error + "</h5>";

}

}

})

}

</script>

</html>

### **RoomOne.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>IOT - Dashboard</title>

<!-- Bootstrap core CSS-->

<link href="static/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom fonts for this template-->

<link href="static/assets/vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css">

<!-- Page level plugin CSS-->

<link href="static/assets/vendor/datatables/dataTables.bootstrap4.css" rel="stylesheet">

<!-- Custom styles for this template-->

<link href="static/assets/css/sb-admin.css" rel="stylesheet">

</head>

<body id="page-top">

<nav class="navbar navbar-expand navbar-dark bg-dark static-top">

<a class="navbar-brand mr-1" href="index">Random Security Company 101</a>

<button class="btn btn-link btn-sm text-white order-1 order-sm-0" id="sidebarToggle" href="#">

<i class="fas fa-bars"></i>

</button>

<ul class="navbar-nav ml-auto ml-md-0" style="position:absolute; right:0">

<li class="nav-item dropdown no-arrow">

<a class="nav-link dropdown-toggle" href="#" id="userDropdown" role="button" data-toggle="dropdown"

aria-haspopup="true" aria-expanded="false">

<i class="fas fa-user-circle fa-fw"></i>

</a>

<div class="dropdown-menu dropdown-menu-right" aria-labelledby="userDropdown">

<a class="dropdown-item" href="#" data-toggle="modal" data-target="#logoutModal">Logout</a>

</div>

</li>

</ul>

</nav>

<div id="wrapper">

<!-- Sidebar -->

<ul class="sidebar navbar-nav">

<li class="nav-item">

<a class="nav-link" href="index">

<i class="fas fa-fw fa-tachometer-alt"></i>

<span>Dashboard</span>

</a>

</li>

<li class="nav-item active">

<a class="nav-link" href="RoomOne">

<i class="fas fa-fw fa-home"></i>

<span>Room One</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" href="RoomTwo">

<i class="fas fa-fw fa-home"></i>

<span>Room Two</span>

</a>

</li>

</ul>

<div id="content-wrapper">

<!-- Light Switch -->

<div class="card mb-3" style="float:left">

<div class="card-header">

<i class="fas fa-poop"></i>

Light Switch</div>

<div class="card-body">

<div id="Lights" width="100%">

<div>

<button onclick="LightToggle()">Toggle Light</button>

<img id="LED" src="static/assets/img/led\_red\_off.png" alt="Red LED light off">

<p>Status <a id="onoff">Off</a></br>

</p>

</div>

</div>

</div>

</div>

<!-- Data Table -->

<div class="card mb-3">

<div class="card-header">

<i class="fas fa-table"></i>

Entrance Report</div>

<div class="card-body">

<div class="table-responsive">

<table class="table table-bordered" id="dataTable2" width="100%" cellspacing="0">

<thead>

<tr>

<th>Time</th>

<th>Card Number</th>

</tr>

</thead>

<tbody id="data2">

</tbody>

</table>

</div>

</div>

<div class="card-footer small text-muted" id="updates"></div>

</div>

<!-- Chart -->

<div class="card mb-3">

<div class="card-header">

<i class="fas fa-chart-area"></i>

Temperature and Humidity Chart</div>

<div class="card-body">

<div id="myChart"></div>

</div>

<div class="card-footer small text-muted" id="updates1"></div>

</div>

<!-- Data Table -->

<div class="card mb-3">

<div class="card-header">

<i class="fas fa-table"></i>

Historical Data Table</div>

<div class="card-body">

<div class="table-responsive">

<table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">

<thead>

<tr>

<th>Time</th>

<th>Temperature</th>

<th>Humidity</th>

</tr>

</thead>

<tbody id="data">

</tbody>

</table>

</div>

</div>

<div class="card-footer small text-muted" id="updates2"></div>

</div>

</div>

<!-- /.container-fluid -->

<!-- Sticky Footer -->

<footer class="sticky-footer">

<div class="container my-auto">

<div class="copyright text-center my-auto">

<span>IOT Project 2018/19 Sem 2</span>

</div>

</div>

</footer>

</div>

<!-- /.content-wrapper -->

</div>

<!-- /#wrapper -->

<!-- Logout Modal-->

<div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>

<button class="close" type="button" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">×</span>

</button>

</div>

<div class="modal-body">Select "Logout" below if you are ready to end your current session.</div>

<div class="modal-footer">

<button class="btn btn-secondary" type="button" data-dismiss="modal">Cancel</button>

<a class="btn btn-primary" href="/">Logout</a>

</div>

</div>

</div>

</div>

<!-- Bootstrap core JavaScript-->

<script src="static/assets/vendor/jquery/jquery.min.js"></script>

<script src="static/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<!-- Core plugin JavaScript-->

<script src="static/assets/vendor/jquery-easing/jquery.easing.min.js"></script>

<!-- Custom scripts for all pages-->

<script src="static/assets/js/sb-admin.min.js"></script>

</body>

<!-- Script for google charts -->

<script type="text/javascript" src="https://code.jquery.com/jquery-3.2.1.js"></script>

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

<script type="text/javascript">

// Load the Visualization API and the corechart package.

google.charts.load('current', { 'packages': ['corechart'] });

// Set a callback to run when the Google Visualization API is loaded.

google.charts.setOnLoadCallback(init);

function init() {

setIntervalAndExecute(function () {

getData(function (myRtnA, data) {

if (myRtnA == 'success') {

create\_dataTable(data.data);

drawChart(data.data);

};

}),

getCardData(function (myRtnA, data) {

create\_Table(data.data);

}),

setupdate()

}, 3000);

}

function setupdate() {

now = new Date();

document.getElementById("updates").innerHTML = now;

document.getElementById("updates1").innerHTML = now;

document.getElementById("updates2").innerHTML = now;

}

function getData(callBack) {

jQuery.ajax({

url: "/api/tempHum1",

type: 'POST',

beforeSend: function () {

},

success: function (data, textStatus, xhr) {

if (data) {

var myRtnA = 'success';

} else {

var myRtnA = 'fail';

}

return callBack(myRtnA, data);

},

error: function (xhr, textStatus, errorThrown) {

console.log("opps: " + textStatus + " : " + errorThrown);

var myRtnA = "Error";

return callBack(myRtnA); // return callBack() with myRtna

}

});

}

function getCardData(callBack) {

jQuery.ajax({

url: "/api/rfid1",

type: 'POST',

beforeSend: function () {

},

success: function (data, textStatus, xhr) {

if (data) {

var myRtnA = 'success';

} else {

var myRtnA = 'fail';

}

return callBack(myRtnA, data);

},

error: function (xhr, textStatus, errorThrown) {

console.log("opps: " + textStatus + " : " + errorThrown);

var myRtnA = "Error";

return callBack(myRtnA); // return callBack() with myRtna

}

});

}

function create\_Table(data) {

var dataTable\_html = '';

var newdata = JSON.parse(data);

for (index = 0; index < newdata.length; index++) {

datetime = (newdata[index].datetime).substring(0, 19);

jsdatetime = new Date(Date.parse(datetime));

rfid = newdata[index].RFID;

var thisRow\_HTML = '<tr>';

thisRow\_HTML += '<td>';

thisRow\_HTML += jsdatetime;

thisRow\_HTML += '</td>';

thisRow\_HTML += '<td>';

thisRow\_HTML += rfid;

thisRow\_HTML += '</td>';

thisRow\_HTML += '</tr>';

dataTable\_html += thisRow\_HTML;

};

document.getElementById("data2").innerHTML = dataTable\_html;

}

function drawChart(data) {

var dataTb = new google.visualization.DataTable();

dataTb.addColumn('string', 'Time');

dataTb.addColumn('number', 'Temperature');

dataTb.addColumn('number', 'Humidity');

var newdata = JSON.parse(data);

for (index = 0; index < newdata.length; index++) {

datetime = (newdata[index].datetimeid).substring(0, 19);

jsdatetime = new Date(Date.parse(datetime));

jstime = jsdatetime.toLocaleTimeString();

temp = parseInt(newdata[index].temp);

hum = parseInt(newdata[index].hum);

dataTb.addRows([[jstime, temp, hum]])

}

var chart = new google.visualization.LineChart(

document.getElementById('myChart'));

chart.draw(dataTb, {

legend: { position: 'top' }, vAxis: { baseline: 0 },

colors: ['#ffb0b0', '#98f6b3']

});

}

function create\_dataTable(data) {

var dataTable\_html = '';

var newdata = JSON.parse(data);

for (index = 0; index < newdata.length; index++) {

datetime = (newdata[index].datetimeid).substring(0, 19);

jsdatetime = new Date(Date.parse(datetime));

jstime = jsdatetime.toLocaleTimeString();

temp = parseInt(newdata[index].temp);

hum = parseInt(newdata[index].hum);

var thisRow\_HTML = '<tr>';

thisRow\_HTML += '<td>';

thisRow\_HTML += jstime;

thisRow\_HTML += '</td>';

thisRow\_HTML += '<td>';

thisRow\_HTML += temp;

thisRow\_HTML += '</td>';

thisRow\_HTML += '<td>';

thisRow\_HTML += hum;

thisRow\_HTML += '</td>';

thisRow\_HTML += '</tr>';

dataTable\_html += thisRow\_HTML;

};

document.getElementById("data").innerHTML = dataTable\_html;

}

function setIntervalAndExecute(fn, t) {

fn();

return (setInterval(fn, t));

}

function LightToggle() {

var toggleswitch = document.getElementById('onoff');

if (toggleswitch.innerHTML === "On") {

ACL('off');

} else {

ACL('on');

}

}

function ACL(commandtext) {

jQuery.ajax({

url: "/api/Lightsoff",

type: 'POST',

data: { command: commandtext, room: 1 },

success: function (data, textStatus, jqXHR) {

console.log(data)

var toggleswitch = document.getElementById('onoff');

var LEDimg = document.getElementById('LED')

if (data.message === "off") {

toggleswitch.innerHTML = "Off"

$('#LED').attr("src", "static/assets/img/led\_red\_off.png");

$('#LED').attr("alt", "Red LED light off");

} else if (data.message === "on") {

toggleswitch.innerHTML = "On";

$('#LED').attr("src", "static/assets/img/led\_red\_on.png");

$('#LED').attr("alt", "Red LED light on");

}

}

})

} //end ACL

</script>

</html>

### RoomTwo.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>IOT - Dashboard</title>

<!-- Bootstrap core CSS-->

<link href="static/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom fonts for this template-->

<link href="static/assets/vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css">

<!-- Page level plugin CSS-->

<link href="static/assets/vendor/datatables/dataTables.bootstrap4.css" rel="stylesheet">

<!-- Custom styles for this template-->

<link href="static/assets/css/sb-admin.css" rel="stylesheet">

</head>

<body id="page-top">

<nav class="navbar navbar-expand navbar-dark bg-dark static-top">

<a class="navbar-brand mr-1" href="index">Random Security Company 101</a>

<button class="btn btn-link btn-sm text-white order-1 order-sm-0" id="sidebarToggle" href="#">

<i class="fas fa-bars"></i>

</button>

<ul class="navbar-nav ml-auto ml-md-0" style="position:absolute; right:0">

<li class="nav-item dropdown no-arrow">

<a class="nav-link dropdown-toggle" href="#" id="userDropdown" role="button" data-toggle="dropdown"

aria-haspopup="true" aria-expanded="false">

<i class="fas fa-user-circle fa-fw"></i>

</a>

<div class="dropdown-menu dropdown-menu-right" aria-labelledby="userDropdown">

<a class="dropdown-item" href="#" data-toggle="modal" data-target="#logoutModal">Logout</a>

</div>

</li>

</ul>

</nav>

<div id="wrapper">

<!-- Sidebar -->

<ul class="sidebar navbar-nav">

<li class="nav-item">

<a class="nav-link" href="index">

<i class="fas fa-fw fa-tachometer-alt"></i>

<span>Dashboard</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" href="RoomOne">

<i class="fas fa-fw fa-home"></i>

<span>Room One</span>

</a>

</li>

<li class="nav-item active">

<a class="nav-link" href="RoomTwo">

<i class="fas fa-fw fa-home"></i>

<span>Room Two</span>

</a>

</li>

</ul>

<div id="content-wrapper">

<!-- Data Table -->

<div class="card mb-3">

<div class="card-header">

<i class="fas fa-table"></i>

Entrance Report</div>

<div class="card-body">

<div class="table-responsive">

<table class="table table-bordered" id="dataTable2" width="100%" cellspacing="0">

<thead>

<tr>

<th>Time</th>

<th>Card Number</th>

</tr>

</thead>

<tbody id="data2">

</tbody>

</table>

</div>

</div>

<div class="card-footer small text-muted" id="updates"></div>

</div>

<!-- Chart -->

<div class="card mb-3">

<div class="card-header">

<i class="fas fa-chart-area"></i>

Temperature and Humidity Chart</div>

<div class="card-body">

<div id="myChart"></div>

</div>

<div class="card-footer small text-muted" id="updates1"></div>

</div>

<!-- Data Table -->

<div class="card mb-3">

<div class="card-header">

<i class="fas fa-table"></i>

Historical Data Table</div>

<div class="card-body">

<div class="table-responsive">

<table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">

<thead>

<tr>

<th>Time</th>

<th>Temperature</th>

<th>Humidity</th>

</tr>

</thead>

<tbody id="data">

</tbody>

</table>

</div>

</div>

<div class="card-footer small text-muted" id="updates2"></div>

</div>

</div>

<!-- /.container-fluid -->

<!-- Sticky Footer -->

<footer class="sticky-footer">

<div class="container my-auto">

<div class="copyright text-center my-auto">

<span>IOT Project 2018/19 Sem 2</span>

</div>

</div>

</footer>

</div>

<!-- /.content-wrapper -->

</div>

<!-- /#wrapper -->

<!-- Logout Modal-->

<div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>

<button class="close" type="button" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">×</span>

</button>

</div>

<div class="modal-body">Select "Logout" below if you are ready to end your current session.</div>

<div class="modal-footer">

<button class="btn btn-secondary" type="button" data-dismiss="modal">Cancel</button>

<a class="btn btn-primary" href="/">Logout</a>

</div>

</div>

</div>

</div>

<!-- Bootstrap core JavaScript-->

<script src="static/assets/vendor/jquery/jquery.min.js"></script>

<script src="static/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<!-- Core plugin JavaScript-->

<script src="static/assets/vendor/jquery-easing/jquery.easing.min.js"></script>

<!-- Custom scripts for all pages-->

<script src="static/assets/js/sb-admin.min.js"></script>

</body>

<!-- Script for google charts -->

<script type="text/javascript" src="https://code.jquery.com/jquery-3.2.1.js"></script>

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

<script type="text/javascript">

// Load the Visualization API and the corechart package.

google.charts.load('current', { 'packages': ['corechart'] });

// Set a callback to run when the Google Visualization API is loaded.

google.charts.setOnLoadCallback(init);

function init() {

setIntervalAndExecute(function () {

getData(function (myRtnA, data) {

if (myRtnA == 'success') {

create\_dataTable(data.data);

drawChart(data.data);

};

}),

getCardData(function (myRtnA, data) {

create\_Table(data.data);

}),

setupdate()

}, 3000);

}

function setupdate() {

now = new Date();

document.getElementById("updates").innerHTML = now;

document.getElementById("updates1").innerHTML = now;

document.getElementById("updates2").innerHTML = now;

}

function getData(callBack) {

jQuery.ajax({

url: "/api/tempHum2",

type: 'POST',

beforeSend: function () {

},

success: function (data, textStatus, xhr) {

if (data) {

var myRtnA = 'success';

} else {

var myRtnA = 'fail';

}

return callBack(myRtnA, data);

},

error: function (xhr, textStatus, errorThrown) {

console.log("opps: " + textStatus + " : " + errorThrown);

var myRtnA = "Error";

return callBack(myRtnA); // return callBack() with myRtna

}

});

}

function getCardData(callBack) {

jQuery.ajax({

url: "/api/rfid2",

type: 'POST',

beforeSend: function () {

},

success: function (data, textStatus, xhr) {

if (data) {

var myRtnA = 'success';

} else {

var myRtnA = 'fail';

}

return callBack(myRtnA, data);

},

error: function (xhr, textStatus, errorThrown) {

console.log("opps: " + textStatus + " : " + errorThrown);

var myRtnA = "Error";

return callBack(myRtnA); // return callBack() with myRtna

}

});

}

function create\_Table(data) {

var dataTable\_html = '';

var newdata = JSON.parse(data);

for (index = 0; index < newdata.length; index++) {

datetime = (newdata[index].datetime).substring(0, 19);

jsdatetime = new Date(Date.parse(datetime));

rfid = newdata[index].RFID;

var thisRow\_HTML = '<tr>';

thisRow\_HTML += '<td>';

thisRow\_HTML += jsdatetime;

thisRow\_HTML += '</td>';

thisRow\_HTML += '<td>';

thisRow\_HTML += rfid;

thisRow\_HTML += '</td>';

thisRow\_HTML += '</tr>';

dataTable\_html += thisRow\_HTML;

};

document.getElementById("data2").innerHTML = dataTable\_html;

}

function drawChart(data) {

var dataTb = new google.visualization.DataTable();

dataTb.addColumn('string', 'Time');

dataTb.addColumn('number', 'Temperature');

dataTb.addColumn('number', 'Humidity');

var newdata = JSON.parse(data);

for (index = 0; index < newdata.length; index++) {

datetime = (newdata[index].datetimeid).substring(0, 19);

jsdatetime = new Date(Date.parse(datetime));

jstime = jsdatetime.toLocaleTimeString();

temp = parseInt(newdata[index].temp);

hum = parseInt(newdata[index].hum);

dataTb.addRows([[jstime, temp, hum]])

}

var chart = new google.visualization.LineChart(

document.getElementById('myChart'));

chart.draw(dataTb, {

legend: { position: 'top' }, vAxis: { baseline: 0 },

colors: ['#ffb0b0', '#98f6b3']

});

}

function create\_dataTable(data) {

var dataTable\_html = '';

var newdata = JSON.parse(data);

for (index = 0; index < newdata.length; index++) {

datetime = (newdata[index].datetimeid).substring(0, 19);

jsdatetime = new Date(Date.parse(datetime));

jstime = jsdatetime.toLocaleTimeString();

temp = parseInt(newdata[index].temp);

hum = parseInt(newdata[index].hum);

var thisRow\_HTML = '<tr>';

thisRow\_HTML += '<td>';

thisRow\_HTML += jstime;

thisRow\_HTML += '</td>';

thisRow\_HTML += '<td>';

thisRow\_HTML += temp;

thisRow\_HTML += '</td>';

thisRow\_HTML += '<td>';

thisRow\_HTML += hum;

thisRow\_HTML += '</td>';

thisRow\_HTML += '</tr>';

dataTable\_html += thisRow\_HTML;

};

document.getElementById("data").innerHTML = dataTable\_html;

}

function setIntervalAndExecute(fn, t) {

fn();

return (setInterval(fn, t));

}

</script>

</html>

1. Run the program 😊

|  |  |
| --- | --- |
|  | In your Raspberry Pi, open the Terminal. |
|  | Change the directory to RSC  cd ~/RSC |
|  | Run **dht.py**  sudo python dht.py  Leave the script running. |
|  | Open another Terminal |
|  | Run **read\_display.py**  sudo python read\_display.py  Leave the script running. |
|  | Open another Terminal |
|  | If you are using the Raspberry Pi that is hosting Room One, run **lights.py**  sudo python lights.py  If you are using the Raspberry Pi that is hosting Room Two, run **testt.py**  sudo python testt.py  Leave the script running. |
|  | Open another Terminal |
|  | Run **server.py**  sudo python server.py |
|  | Go to 0.0.0.0:5000 and see the web application.  Replace 0.0.0.0 with your Raspberry Pi’s IP address. |

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