

Project report for Smart Agriculture system based on IoT - SB13826

Project summary:

The project is based on the four technology stack mainly - IBM Watson platform, Node-red, a bit of python programming and Open weather map API. The main motive behind this project is to get information from the Openweather platform and IBM IoT simulator and import it to the node red which we will use to make a dashboard in UI that will show us the levels in gauges and also create buttons that will help us to control motors on the field. The idea is to create a simulation of the real world agricultural sector and how IoT can help in it.

Steps taken to complete the internship:

First we create an Ibm Cloud account and install IoT platform in it. The IoT platform can be found by searching in the catalog section.

Next we log in the Internet of Things cloud account and add a device Type:

IBM Watson IoT Platform

Browse Action **Device Types** Interfaces

Add Device Type +

Q Type the name to search...

<input type="checkbox"/>	Name	Description	Number of Devices	Class ID	Date Added
> <input type="checkbox"/>	iotplatform		1	Device	May 26, 2020 1:40 AM
> <input type="checkbox"/>	nodemcu		1	Device	Jun 8, 2020 9:52 AM
> <input type="checkbox"/>	smart		0	Device	Jun 6, 2020 6:31 PM

Items per page 10 | 1-3 of 3 items

1 of 1 page

1 Simulation running

Next we add a device under the device type we just added and note down the device type, device id and authentication token which will be required to start the lot simulator:

IBM Watson IoT Platform

Browse Action Device Types Interfaces

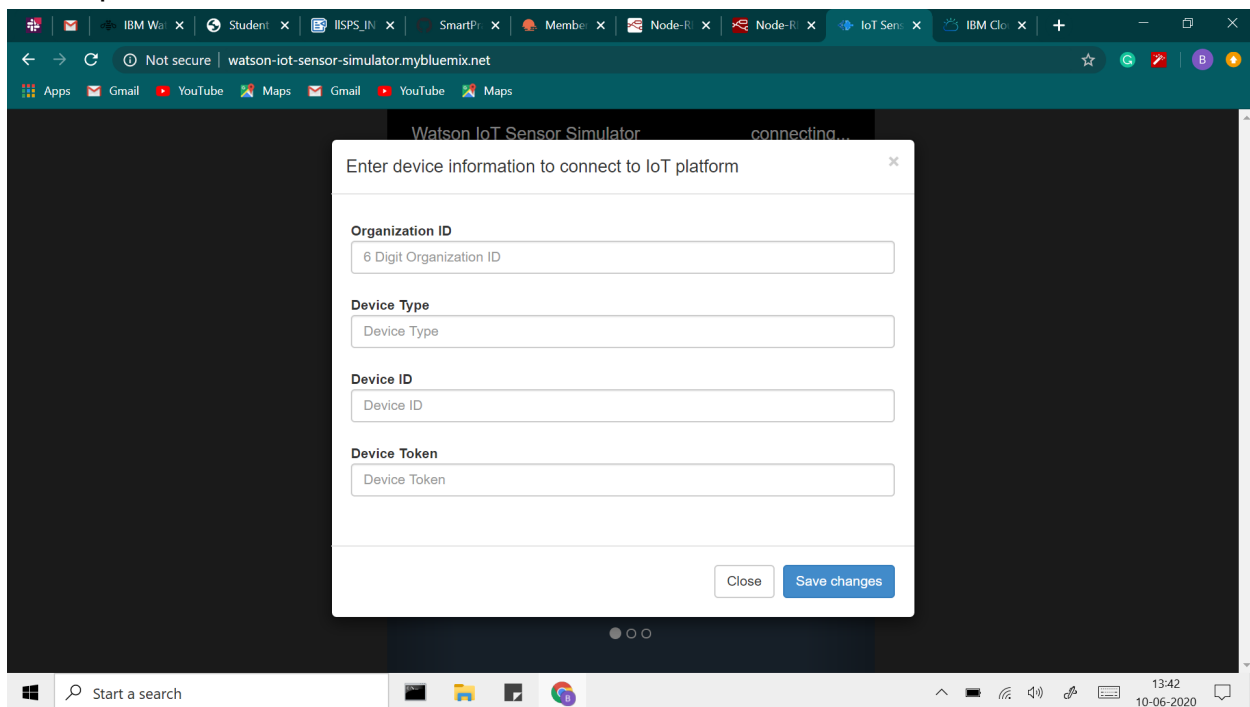
Add Device +

Q Search by Device ID

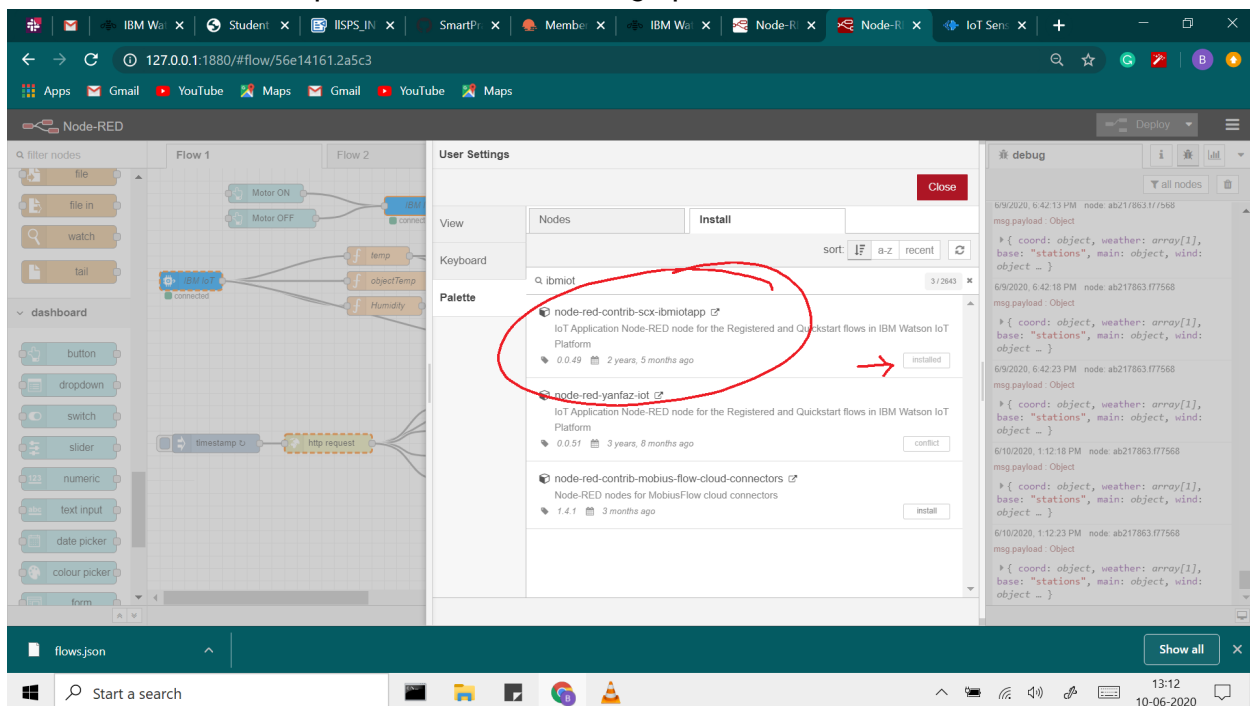
Device Simulator ☒

<input type="checkbox"/>	Device ID	Status	Device Type	Class ID	Date Added
> <input type="checkbox"/>	IBMIOT-device	Disconnected	iotplatform	Device	May 26, 2020 1:43 AM
> <input type="checkbox"/>	123456789	Disconnected	nodemcu	Device	Jun 8, 2020 9:52 AM

Next open IoT simulator and enter the simulator details:



Next we install Node Red on our computer and start Node red, after starting node red we will install IBM lot palette from the manage palette section:



We then go to the Open Weather platform and open an account there and get the API key:

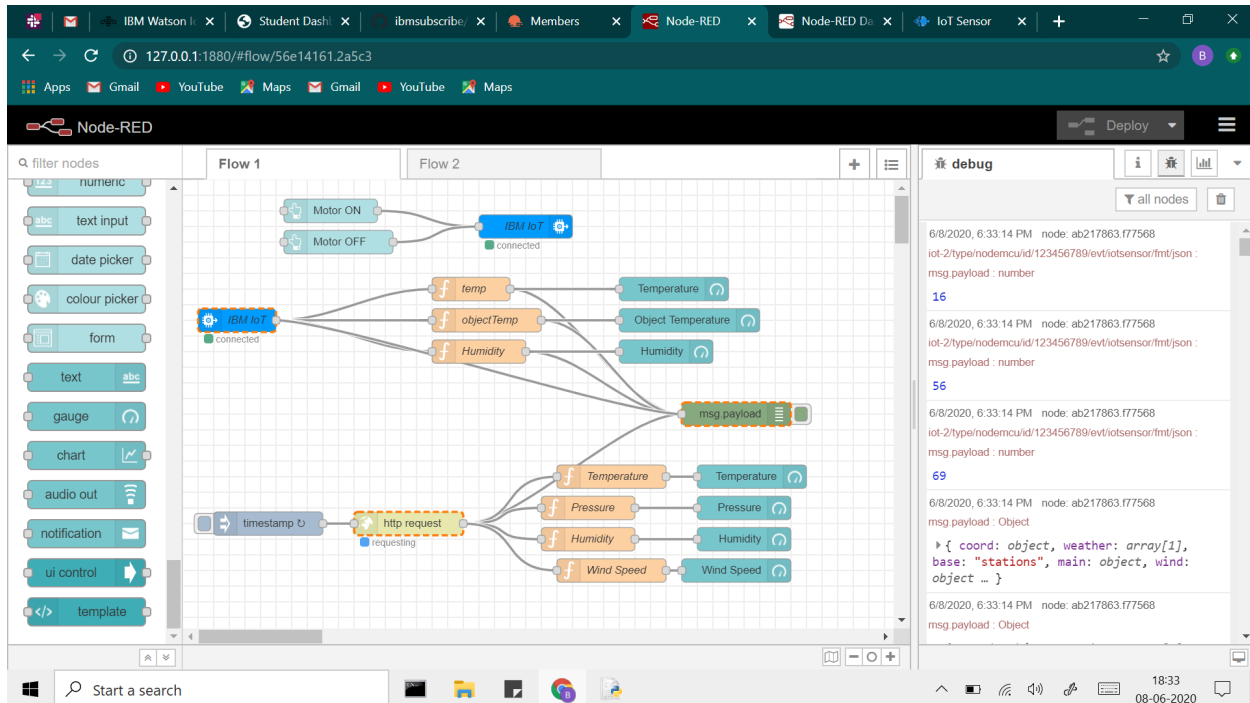
The screenshot shows a web browser window with the URL `home.openweathermap.org/api_keys`. The browser's address bar and tabs are visible at the top. The OpenWeather logo and navigation menu are at the top of the page. The 'API keys' tab is selected in the sub-menu. A message box states: 'You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.'

Key	Name	Create key
b46917f569aaf7646964adf2d2c476e7	Default	<input type="text" value="API key name"/> <button>Generate</button>

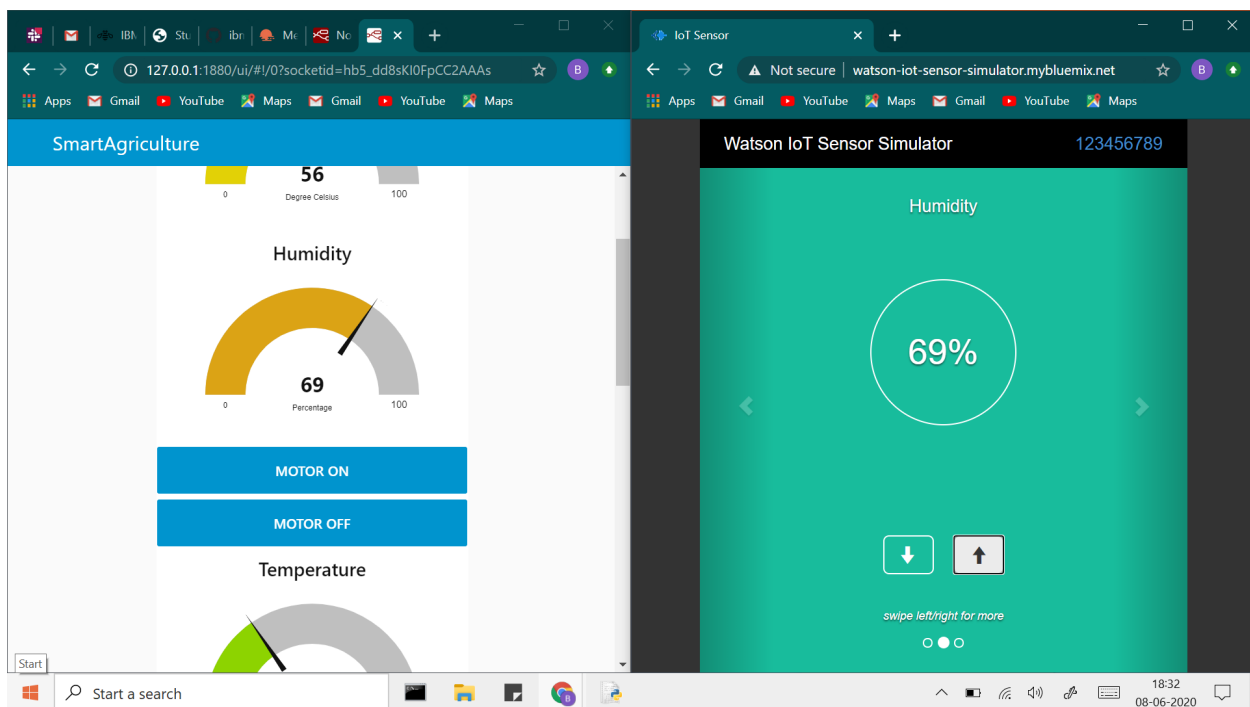
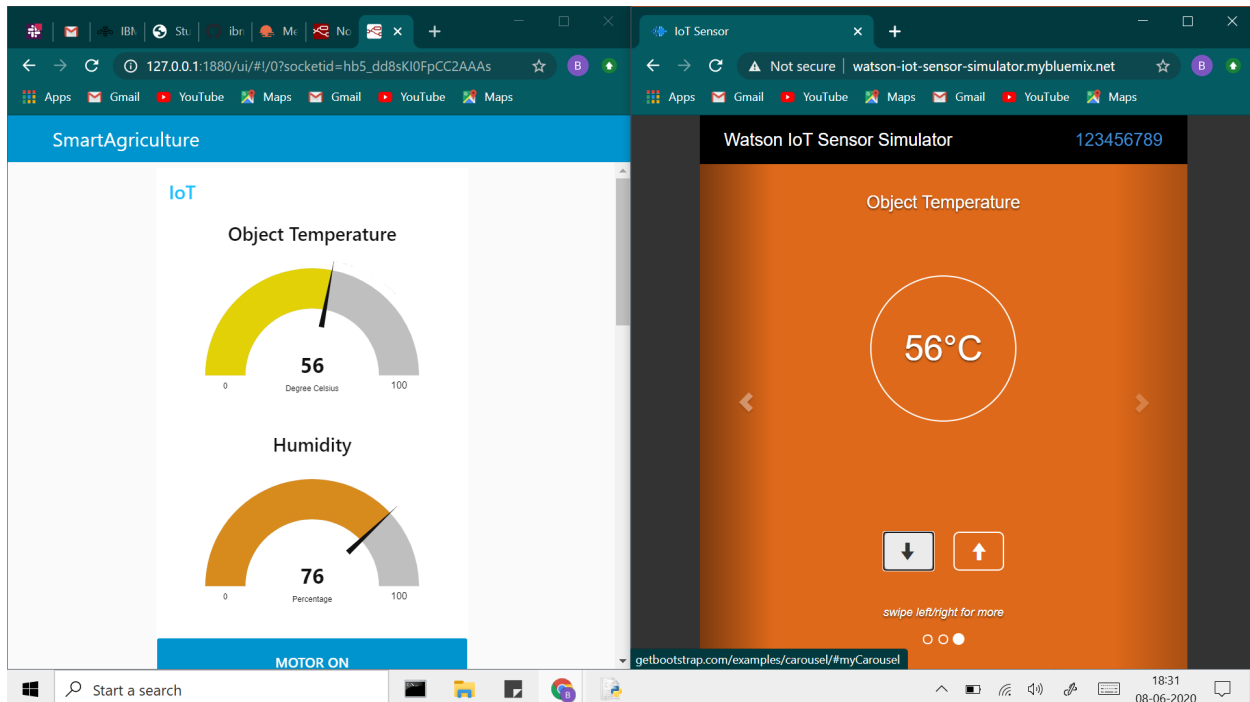
Below the table, there are three columns of links: 'Product Collections' (Current and Forecast APIs, Historical Weather Data), 'Subscription' (How to start, Pricing), and 'About us' (OpenWeather Ltd is a British-based tech company that provides weather and satellite data worldwide. OpenWeather).

The Windows taskbar at the bottom shows the search bar with 'Start a search', taskbar icons for various applications, and the system clock displaying 13:05 on 10-06-2020.

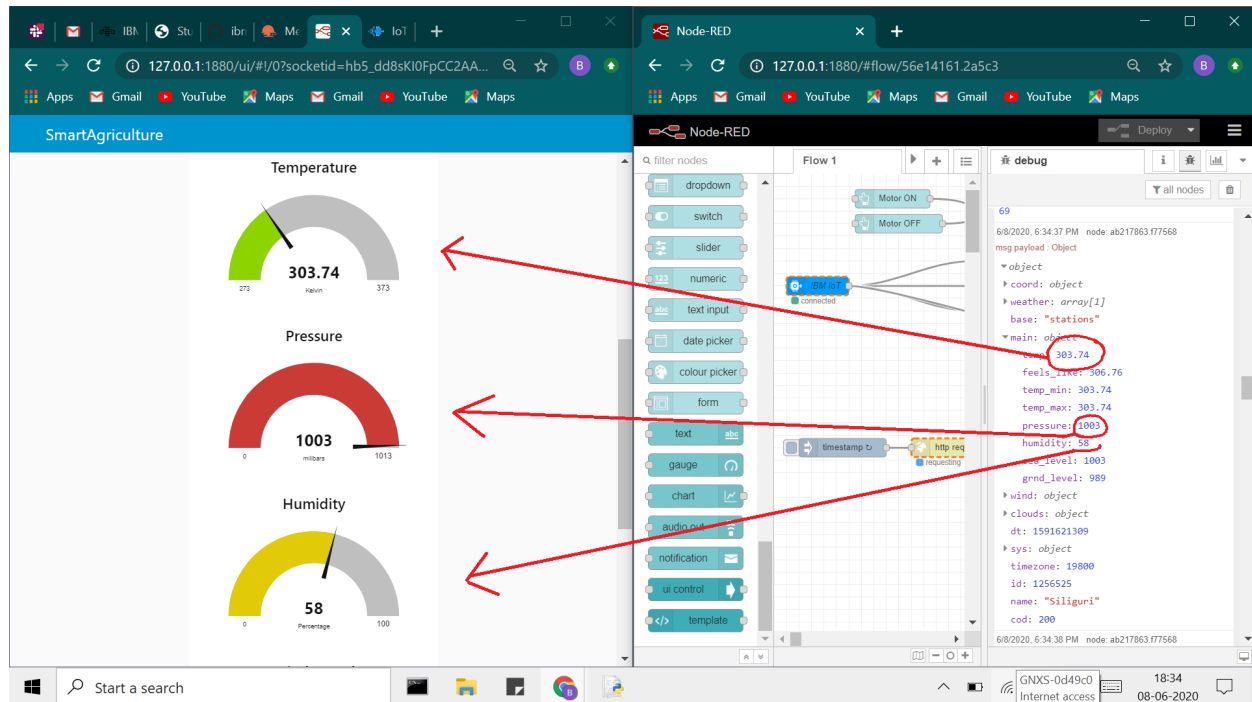
Next we arrange the node red flow as show to get from the IoT simulator and Open Weather API, the IBM lot nodes have been configured with the same API key , device id and authentication token that we used in the IoT simulator and then we add the http request node and configure it with thee API key we got from Open Weather Platform to get the weather details:



Next we deploy the entire flow in UI and get the following dashboard, here we can see the IoT simulator and Nodered side by side and the changes we make to the simulator gets reflected automatically in the Dashboard:



Next we can see the Open weather giving the weather details to the https request node and the results are getting reflected to the UI in real time:



Next step we will have to control the motors using python commands so we install an IBM IoT library in python module by running "pip install ibmiotf" in Python shell:

The screenshot shows a Command Prompt window with the following output:

```

C:\Users\barshan>pip install ibmiotf
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 won't be maintained after that date. A future version of pip will drop support for Python 2.7. More details about Python 2 support in pip, can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support
Collecting ibmiotf
  Downloading https://files.pythonhosted.org/packages/78/05/029ca6f78b788a3c55157fd11bb63922d002d75df982fbb8243f450a750e/ibmiotf-0.4.0.tar.gz (71kB)
    [#####] 81kB 317kB/s
Collecting iso8601>=0.1.12 (from ibmiotf)
  Downloading https://files.pythonhosted.org/packages/ef/57/7162609dab394d38bbc7077b7ba0a6f10fb09d8b7701ea56fa1edc0c4345/iso8601-0.1.12-py2.py3-none-any.whl
Collecting pytz>=2017.3 (from ibmiotf)
  Downloading https://files.pythonhosted.org/packages/4f/a4/879454d49688e2fad93e59d7d4efda588b783c745fd2ec2a3adf87b0808d/pytz-2020.1-py2.py3-none-any.whl (510kB)
    [#####] 512kB 664kB/s
Collecting paho-mqtt>=1.3.1 (from ibmiotf)
  Downloading https://files.pythonhosted.org/packages/59/11/1dd5c70f0f27a88a3a05772cd95f6887ac479fac66d9c7752ee5e16ddbcb/paho-mqtt-1.5.0.tar.gz (99kB)
    [#####] 102kB 758kB/s
Collecting requests>=2.18.4 (from ibmiotf)
  Downloading https://files.pythonhosted.org/packages/1a/70/1935c770cb3be6e3a8b78ced23d7e0fb3187f5cbfab4749523ed65d7c9b1/requests-2.23.0-py2.py3-none-any.whl (58kB)
    [#####] 61kB 393kB/s
Collecting requests_toolbelt>=0.8.0 (from ibmiotf)

```

Next we write a Python code(uploaded to the github repository) to get the motor commands from the Node red dashboard, note that the credentials in this python code is the same as your IoT simulator:

```
finalibmsub.py - C:\Users\barshan\Desktop\smartinterz\finalibmsub.py (2.7.17)
File Edit Format Run Options Window Help

import time
import sys
import ibmiotf.application
import ibmiotf.device

#Provide your IBM Watson Device Credentials
organization = "32s2lj" # repalce it with organization ID
deviceType = "nodemcu" #replace it with device type
deviceId = "123456789" #repalce with device id
authMethod = "token"
authToken = "123456789" #repalce with token

def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data)
    if cmd.data['command']=='motoron':
        print("Motor ON")
    elif cmd.data['command'] == 'motoroff':
        print("Motor OFF")

try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
    #.....

except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
    sys.exit()

deviceCli.connect()

while True:
    T=50;
    H=32;
    #Send Temperature & Humidity to IBM Watson
    data = { 'Temperature': T, 'Humidity': H }
    #print data
    def myOnPublishCallback():
        print ("Published Temperature = %s C" % T, "Humidity = %s %" % H, "to IBM Watson")
```

After running the code we see that the code responds to the buttons we added in the IBM IoT node and can turn the motor ON and OFF as directed:

The screenshot displays a Python 2.7.17 Shell window on the left and a Node-RED dashboard on the right. The shell window shows the execution of the Python code, with output indicating successful connection to the IBM Watson IoT Platform and the receipt of 'motoron' and 'motoroff' commands. The Node-RED dashboard, titled 'SmartAgriculture', features three gauges: 'Object Temperature' (56 Degrees Celsius), 'Humidity' (73 Percentage), and 'Temperature' (308.82). A 'Motor Commands' section contains two buttons: 'MOTOR ON' and 'MOTOR OFF'. Red and blue arrows point from the 'MOTOR ON' and 'MOTOR OFF' buttons respectively to the corresponding command strings in the Python shell's output.

Hence the tasks are updated and the internship has been completed. A big thanks to all the mentors for making this project so simple and easy to follow.