PROJECT REPORT

Project name : Smart solutions for railways

Team ID : PNT2022TMID40456

Team members : Pavithra.A

Indhumathi.C

Gopika.K

Shanmugapriya.E

1.INTRODUCTION:

PROJECT REVIEW:

This is the system enables the users to book ticket and track the current location of train with a web based application build with node-RED.It uses the IBM IOT Watson cloud platform as its backend.

PURPOSE:

The purpose of this system is to track the current location of train. It saves the passengers time . The QR code generated contains all the information of the passengers.The ticket booking is made online and users can get all details regarding the railways.It has high safety measures and assures the safety of passengers.

2.LITERATURE SURVEY:

The iot solution applied for smart railways makes it easy to grasp the information distributed over a wide railway area.Most of the people choose this transportation mainly for low cost and it gives comfort ability.To increase this comfort zone and to reduce the number of accidents,iot gives complete solution. Most of these accidents occurs at railway gate level crossings.

It also involves monitoring process to detect the fault in sensors.

2.1PROBLEM STATEMENT:

Mostly railwaygates are operated manually by labours this can be digitalized by automatic gate system.

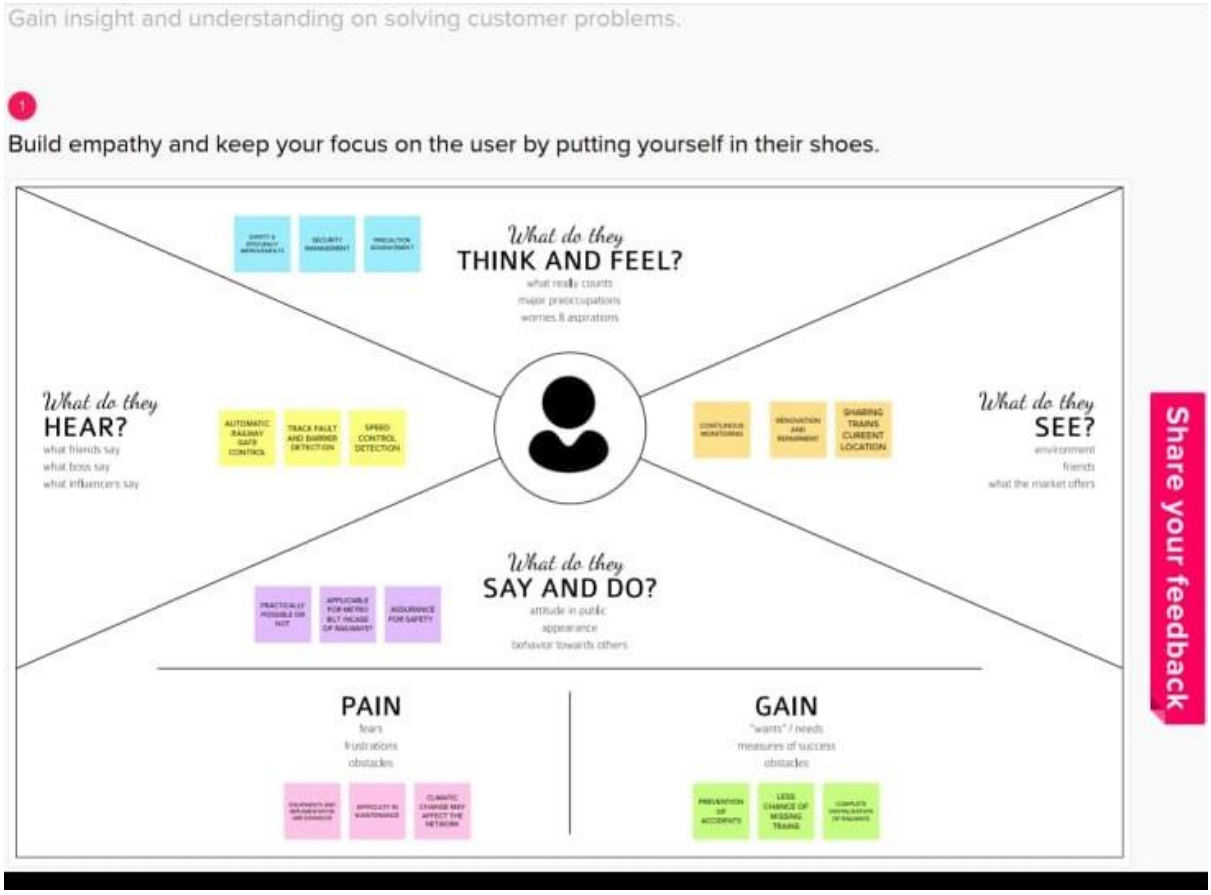
. It prevents human and vehicle accidents .

. It assures improved safety and security . Through continous monitoring of speed we can overcome the problems cause by the speed. . Through track fault and barrier object detection can save many lives.

3.IDEATION AND PROPOSED SOLUTION:

3.1 EMPATHY MAP CANVAS:

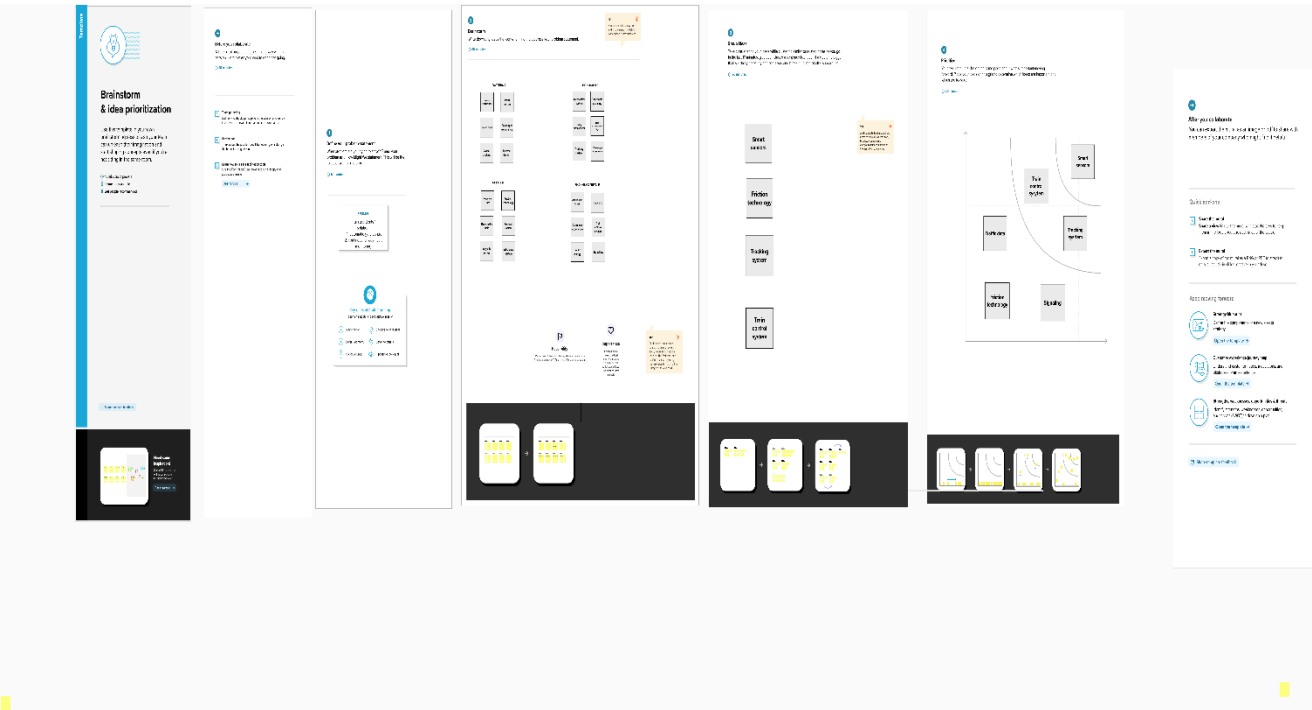
Ideation is the create process of generating, developing, and communicating new ideas, where an is idea understood as a basic element of thought that can be either visual, concrete, or abstract.



3.2 BRAIN STORMING:

brainstorming is one of the most creative ways of problem-solving in which we work on ideas. We can either come up with a new idea or build

on an existing idea as well. Since there is no rule of thumb in brainstorming, it can be applied individually or in a group.

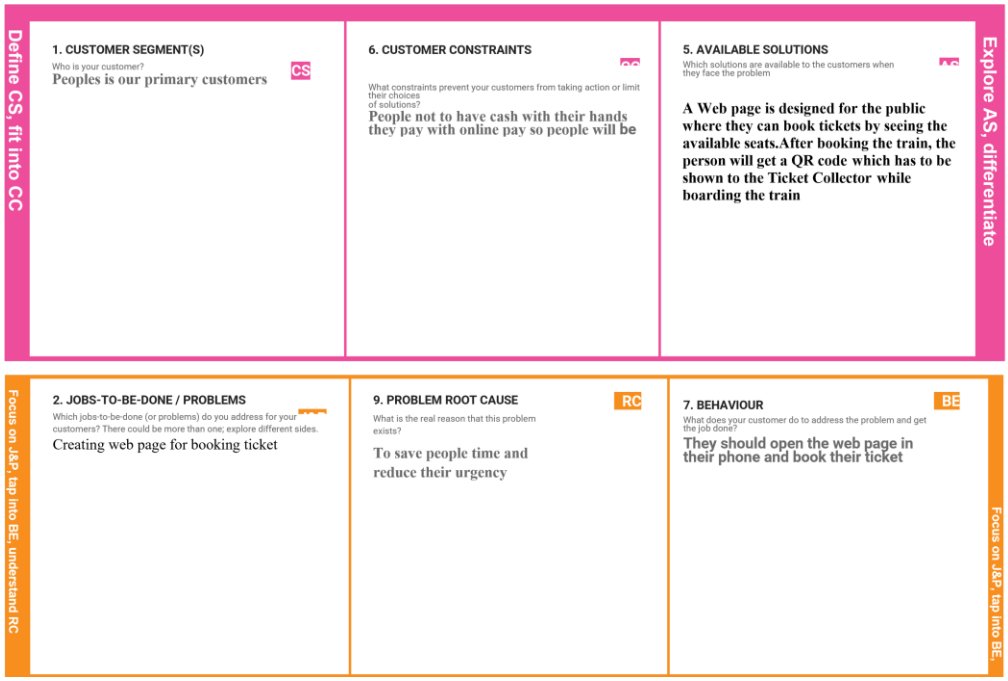


3.3 PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	• Keep track of passengers and schedule their journey accordingly • Information about the route cancellation of tickets ,departure time , arrival time ,number of trains available and other such information. • Store and retrieve information about the various transactions related to rail travel. • Mostly railway gates are operated manually by labours this can be digitalized by automatic gate system.

2.	Idea / Solution description	• . Smart sensors can be used to track important assets, manage passenger flow, and enable predictive maintenance. • IoT devices can also monitor the driver's behaviour and can inform about the driving style and idling time. • The railway gates are operated by automatic gate system.
3.	Novelty / uniqueness	The uniqueness of our proposed paper is that it helps railways successfully manage passengers safety , operational efficiency and passenger experience.
4.	Social Impact / customer satisfaction	Information regarding train arrival and departure time, no of trains available, train current location makes the customer more satisfied.
5.	Business Model (Revenue Model)	It is the cheapest mode of transportation and attracts many customers.
6.	Scalability of the Solution	Iot sensors , vibration and temperature sensor, rail crossing sensors , rail friction sensor , obstacle detecting sensor. These sensors are used for safety and greater reliability .Thus by this proposed solution we can avoid rail line crossing deaths, monitor rail friction , detect obstacles and track maintenance.

3.4 PROBLEM SOLUTION FIT:



4.REQUIREMENT ANALYSIS:

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Mobile no Registration through Gmail
FR-2	User Confirmation	Confirmation via OTP Confirmation via Email
FR-3	Journey details	Provides information such from and to details, date and time of travel.
FR-4	Booking process	Choose a correct train.select seats according to your comfort and confirm your reservation by entering the required details.
FR-5	Confirmation	Ticket confirmation is send to your registered email or mobile number and user can download the e-ticket
FR-6	Tracking	User can view the current location of train.
FR-7	Generation	User can use the QR code which is been generated
FR-8	Reporting issue	User can report the issue
FR-9	Feedback	User can feedback their thoughts.

4.2 Non-functional Requirements:

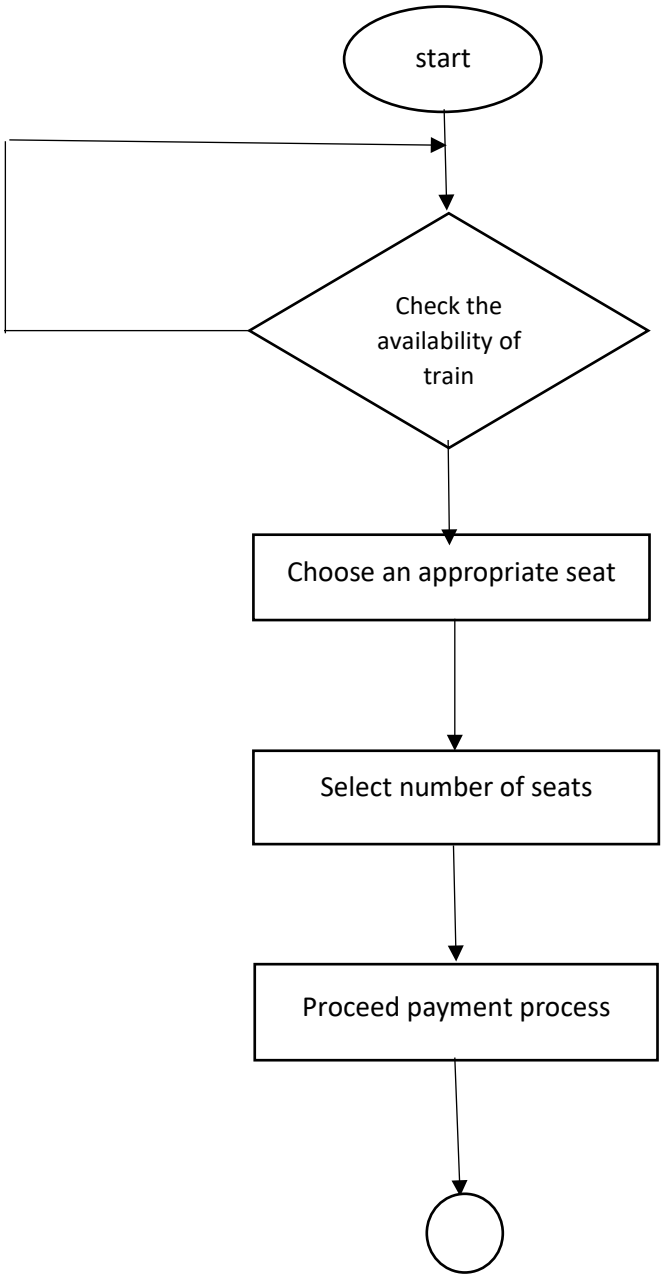
Following are the non-functional requirements of the proposed solution.

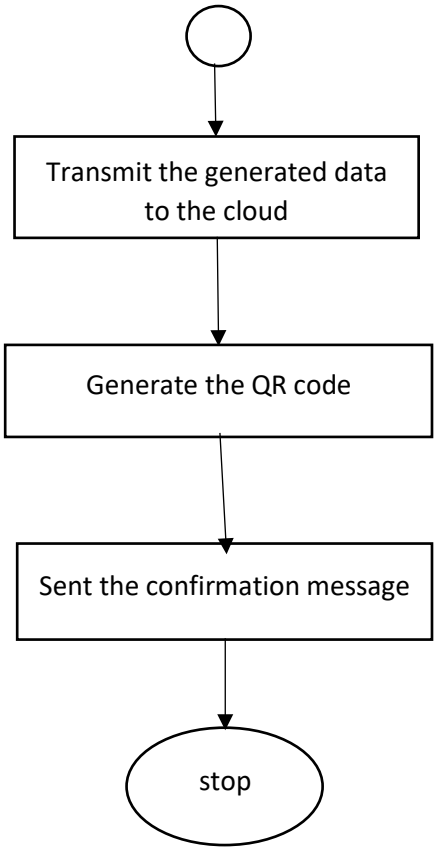
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	the website should be user friendly and easy to use.
NFR-2	**Security**	Strong security used to protect the user password and details of the passengers .
NFR-3	**Reliability**	Easily accessible .Updated and modified as per the convenient of user.Less chance of failure while processing
NFR-4	**Performance**	The request should be accept in a few second Provides real time notification.
NFR-5	**Availability**	Consistent performance monitoring.It is available for the user whenever needed.
NFR-6	**Scalability**	It maintain website traffic with no fast loading speed and top security.

5.PROJECT DESIGN:

5.1DATA FLOW DIAGRAMS AND USER STORIES:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored


start
Check the availability of train
Choose an appropriate seat
Select number of seats
Proceed payment process



User stories:

List all the user stories for the product:

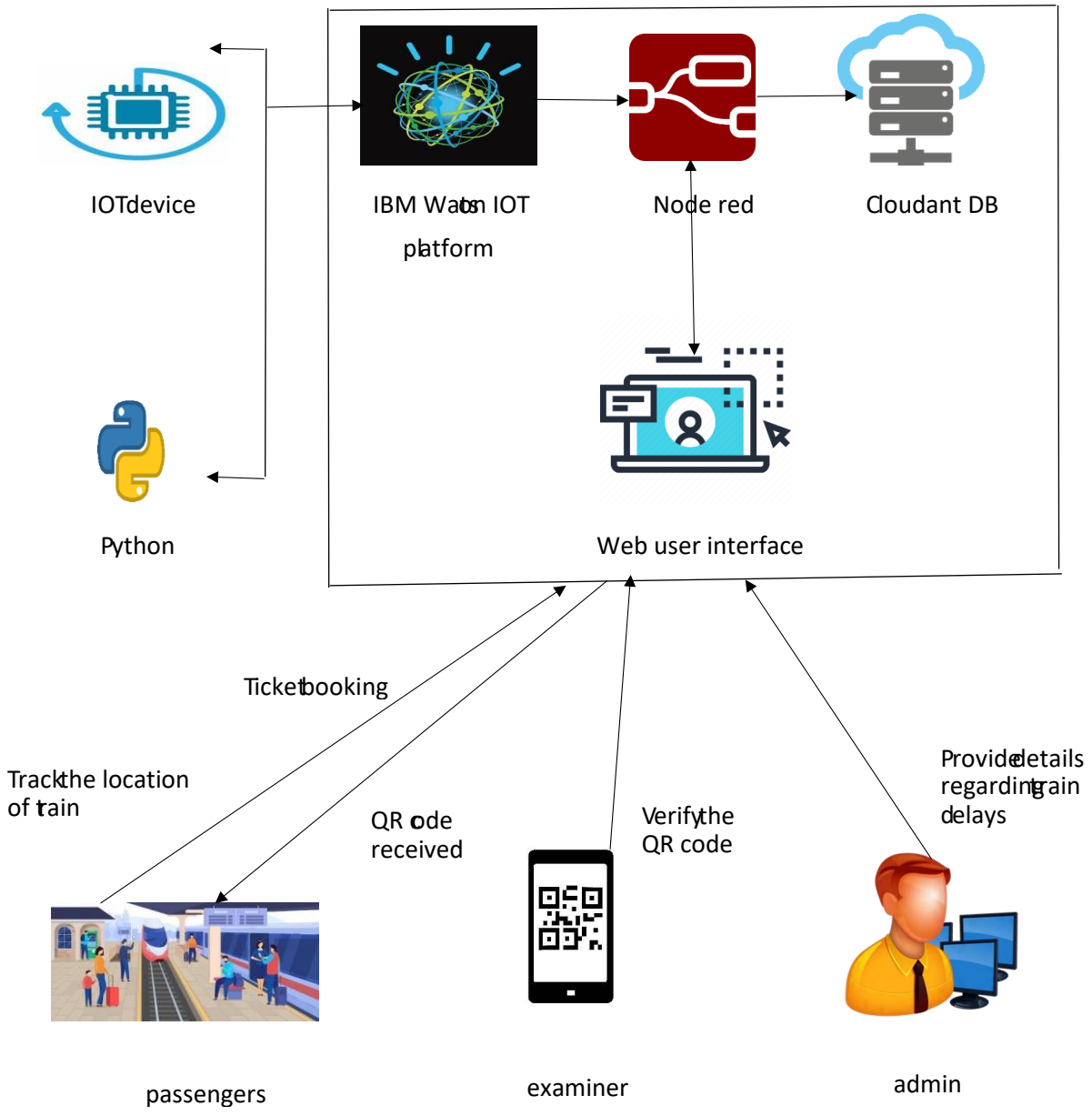
User type	Functional requirements(epic)	User story number	User story/task	Acceptance criteria	priority	release
Customer (mobile user)	Registration	USN-1	Registration through mobile no Registration through Gmail	I can access my account/dashboard	high	Sprint-1
	Confirmation	USN-2	Confirmation via Email Confirmation via OTP	I can receive confirmation email	high	Sprint-1
	login	USN-3	User can login to the application by entering email and password	I can access easily to the dashboard	high	Sprint-1
	Journey details	USN-4	User can browse the train details	I can view train details	High	Sprint-1
	Booking process	USN-5	User can book a ticket by entering the details	I can easily check and confirm the seat selected and reserve ticket	high	Sprint-1

	Confirmation of tickets	USN-6	User can receive booking confirmation message and download the eticket	I can receive confirmation message	high	Sprint-1
	tracking	USN-7	User can track the current location of trains	I can easily track the location of train	High	Sprint-2
	Feedback and reporting issue	USN-8	User can share feedback and report any issues	I can give feedback and report any issue	high	Sprint-2
Customer(web user)	Generation	user	User can use the QR code	I can use the QR code which is been generated	high	Sprint-1
Customer care executive	Connecting the service provider	executor	User can answer the questions raised by the customers	I can solve the queries of customers	high	Sprint-2
administrator	Provide details	admin	User can provide details regarding train delays	I can modify or update the data provided by the customer	high	Sprint-1

5.2 SOLUTION ARCHITECTURE:

Solution Requirements:

- IBM Watson IOT platform
- Node-Red
- Python
- IOT device
- Cloudant DB
- Web user interface



5.3 TECHNICAL ARCHITECTURE:

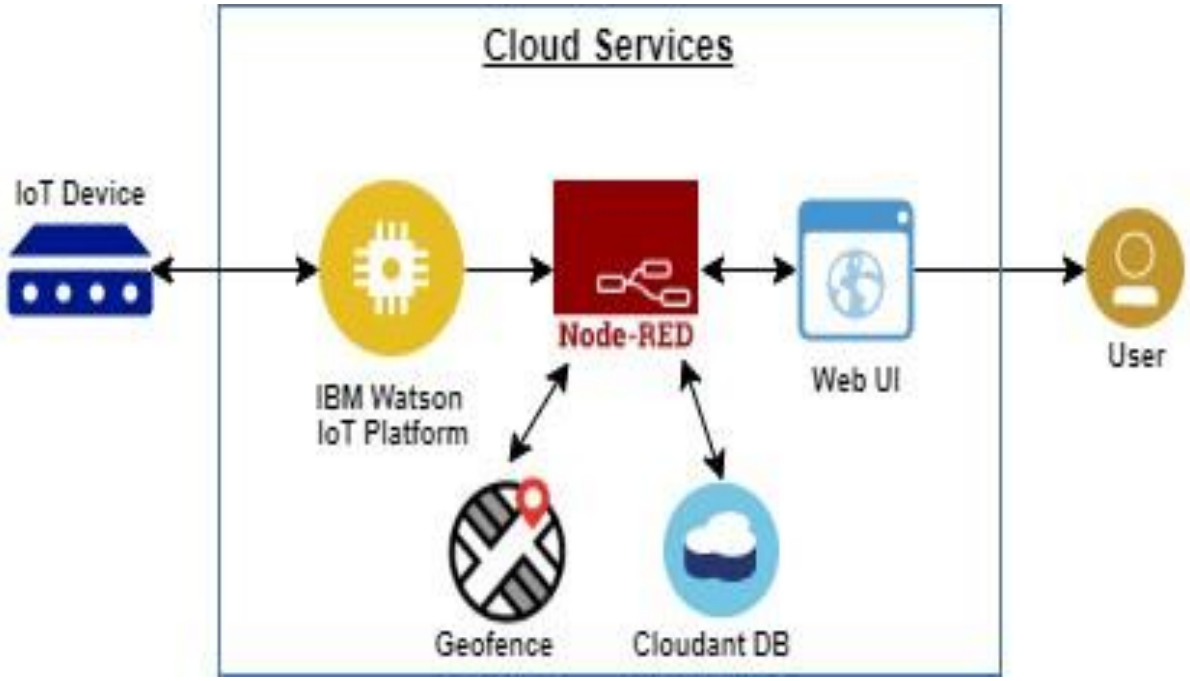


Table-1 : Components & Technologies:

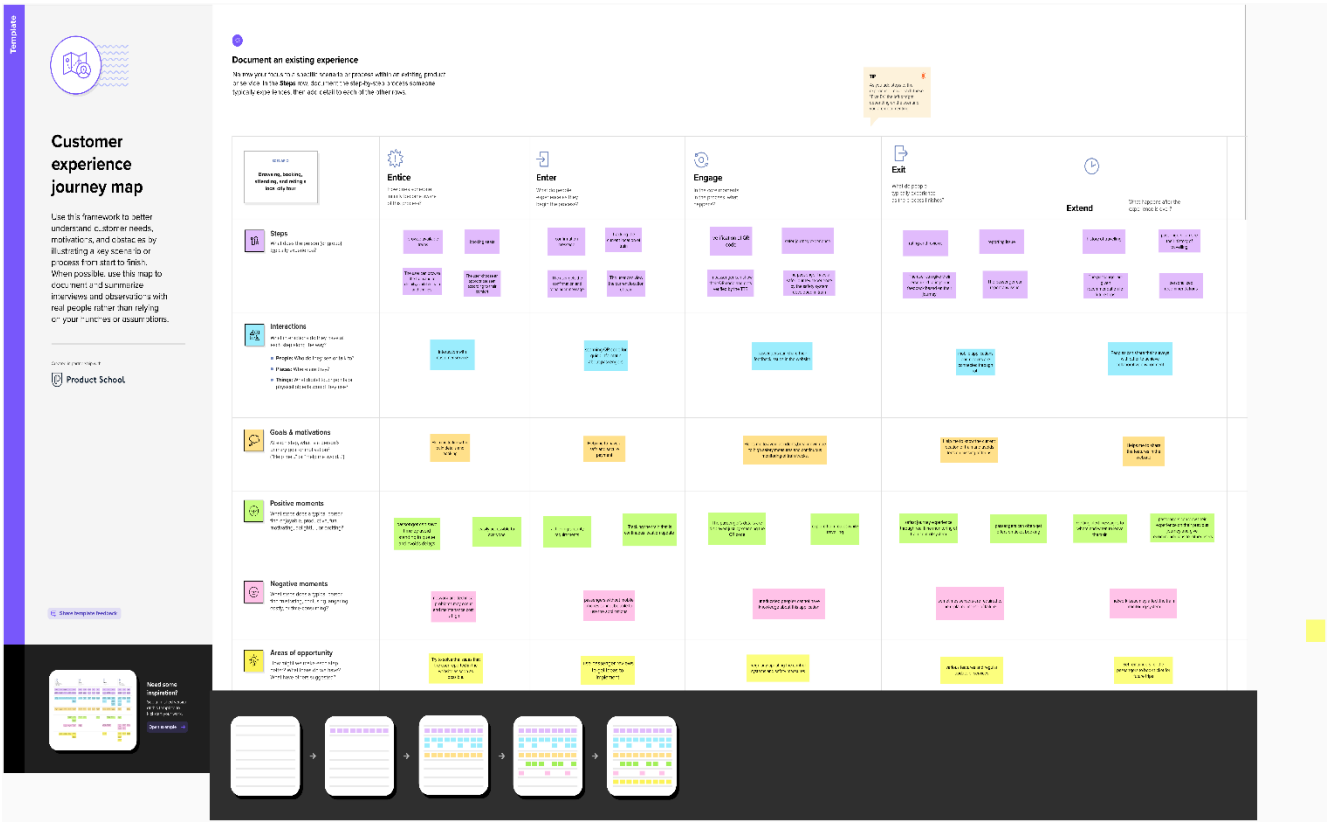
S. No	Component	Description	Technology
1.	User Interface	Web UI	IBM Watson, Node-RED,
2.	Application Logic-1	For a process in the application generate random data	Python , IBM Watson
3.	Database	Data Type, Configurations etc.	MySQLetc.
4.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
5.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
6.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.no	Characteristics	Description	Technology
1.	Open-Source Frameworks	open-source frameworks used for the project	Node-RED
2.	Security Implementations	Strong firewall used to protect user password and data	WEB UI

3.	Scalable Architecture	It should work without negative issue and maintain website traffic	Node-RED(WEB UI)
4.	Availability	It should be available for the user whenever they need	Node-RED(WEB UI)
5.	Performance	The request should be accept in a few second and allow user to use	Node-RED(WEB UI)

5.3 CUSTOMER EXPERIENCE JOURNEY MAP:



6.PROJECT PLANNING AND SCHEDULING:

6.1 MILESTONE AND ACTIVITY LIST:

TITLE	DESCRIPTION	DATE
Literature survey and information gathering	Literature survey on the selected project &gathering information by referring the technical papers ,research publications etc..	17 September 2022
Prepare empathy map	Prepare Empathy map canvas to capture the user pains & gains ,prepare list of problem statements	17 September 2022
Ideation	List by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility& importance	8 October 2022
Proposed solution	Prepare the proposed solution document ,which includes the novelty, feasibility of idea ,business model social impact, scalability of solution, etc..	29 September 2022
Problem solution fit	Prepare problem-solution fit document	20 0ctober 2022
Solution architecture	Prepare solution architecture document	20 October 2022
Customer journey	Prepare the customer journey maps to understand the user interactions& experiences with the applications	20 October 2022

Data flow diagrams	Draw the data flow diagrams and submit for review	20 October 2022
Technology Architecture	Architecture diagram	20 October 2022
Prepare Milestone &Activity list	Prepare the milestone &activity list of the project	23 October 2022
Project Development-Delivery of sprint-1,2,3&4	Develop &submit the developed code by testing it	In progress

6.2 SPRINT DELIVERY SCHEDULE:

Product Backlog, Sprint Schedule, and Estimation

Use the below template to create product backlog and sprint schedule

Sprint	Functional requirements(epic)		User story /task	Story points	Priority	Team Members
Sprint -1	Registration	USN-1	A user can register through the website	2	High	
Sprint -1	Confirmation	USN-2	Confirmation message is received through email or otp through phone	1	High	Indhumathi
Sprint -2	booking	USN-3	A user can book their seat through the web	2	Low	Gopika
Sprint -2	Confirmation	USN-4	A QR code is generated and send through the user	2	Medium	
Sprint -3	verification	USN-5	A ticket collector is verified Through the QR code	1	High	Shanmugapriya
Sprint -4	Location tracking	USN-6	A Gps location of the train is show in the web	2	high	Pavithra

Project Tracker, Velocity & Burndown Chart:

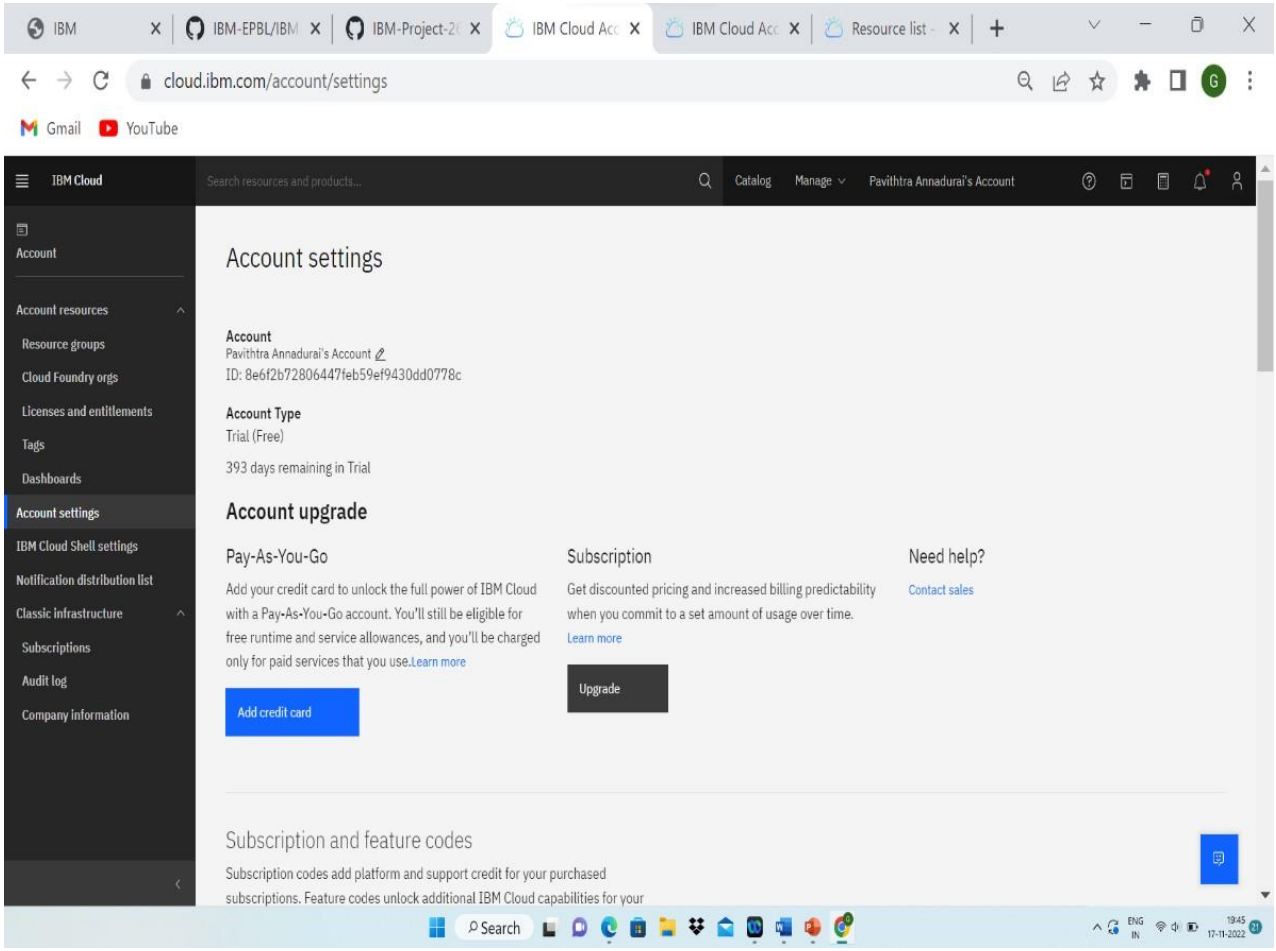
Sprint	Total Story points	Duration	Sprint Start Date	Sprint End Date(planned)	Story Points completed (as on planned End date)	Sprint Release Date
Sprint-1	20	6 Days	24 oct 2022	29 oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 oct 2022	05 Nov 2022	20	05 NOV 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	14 Nov 2022	20	19 Nov 2022

Velocity: Imagine we have a 6 -Day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

AV ***= Sprint duration/Velocity = 20/06 = 3.333***

7.PREREQUISITES:

7.1 IBM CLOUD SERVICE:



7.2 SOFTWARE:

careereducation.smartinternz.com/Student/guided_project_info/2645#

Create IBM Watson IOT Platform And Device

Create Node-RED Service

Develop The Python Script

Develop A Web Application Using Node-RED Service.

Ideation Phase

Project Design Phase – I

Project Design Phase -II

Project Planning Phase

Software

- Install the Python IDLE.

  Click **here** for the reference video of python installation.

- Install the required python libraries:

  Install Watson IoT Python SDK, Cloudant DB, pyzbar, OpenCV to integrate Cloud services, capture the data from the camera, and process the data stored in QR code using the python code.
  Install the above libraries using the commands listed below one after the other in the command prompt.
  - **pip install wiotp-sdk**
  - **pip install ibmcloudant**
  - **pip install opencv-python**
  - **pip install pyzbar**

  Download the required files from the **link**

careereducation.smartinternz.com/Student/guided_project_info/2645#

Create IBM Watson IOT Platform And Device

Create Node-RED Service

Develop The Python Script

Develop A Web Application Using Node-RED Service.

Ideation Phase

Project Design Phase – I

Project Design Phase -II

Project Planning Phase

Software

Type here to search

Pinned

All apps

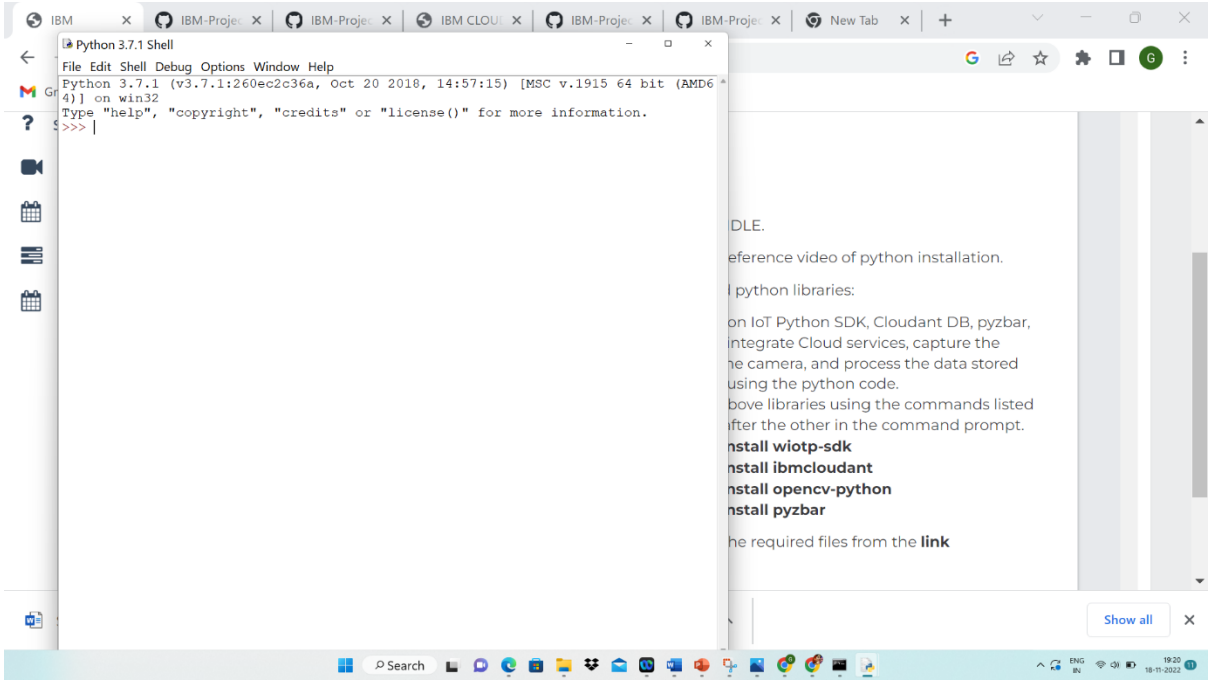
Edge, Word, Excel, PowerP..., Mail, Calendar, Microso..., Photos, Settings, OneNote, Calculator, Clock, Notepad, Paint, File..., Movies..., Tips

Recommended

More

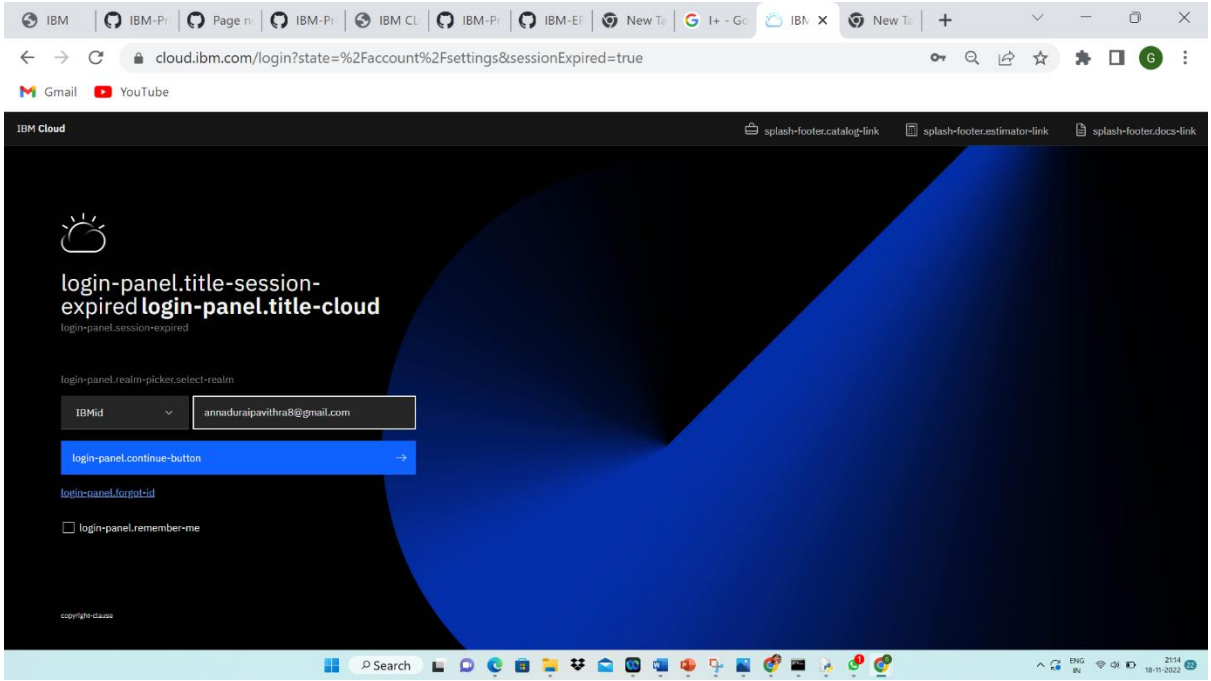
IDLE (Python 3.7 64-..., Python 3.7 Module..., Python 3.7 Manuals..., Python 3.7 (64-bit), Document, Software (3)

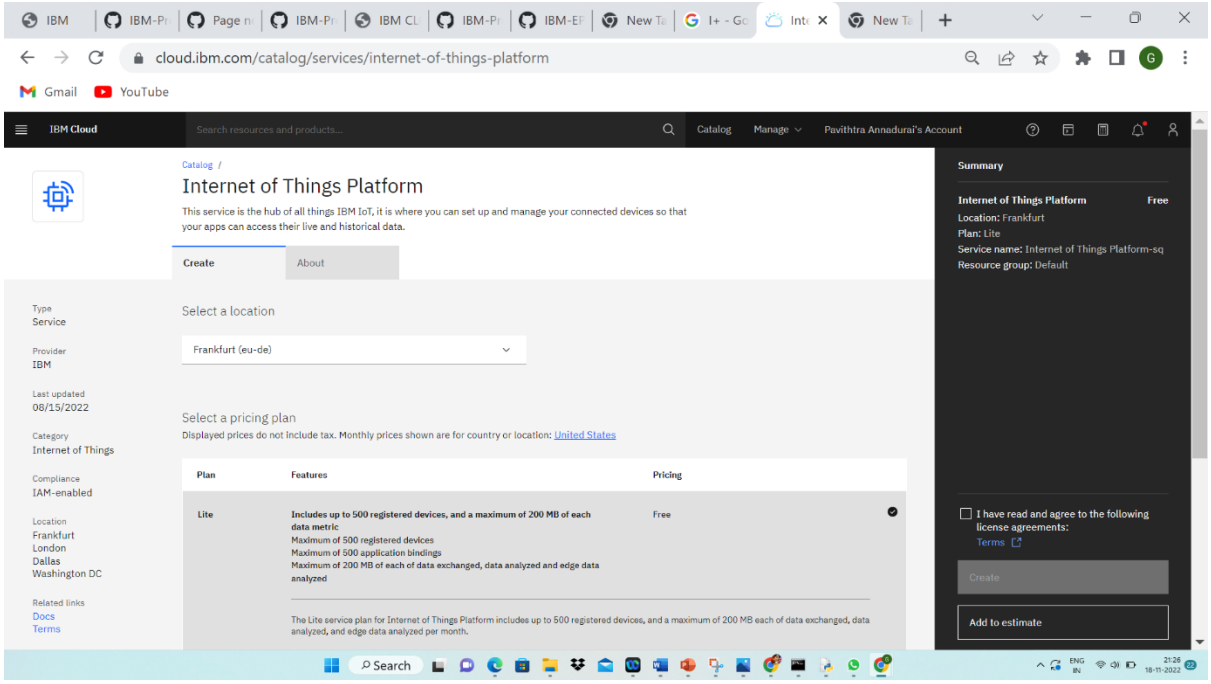
Gopika Karthikeyan

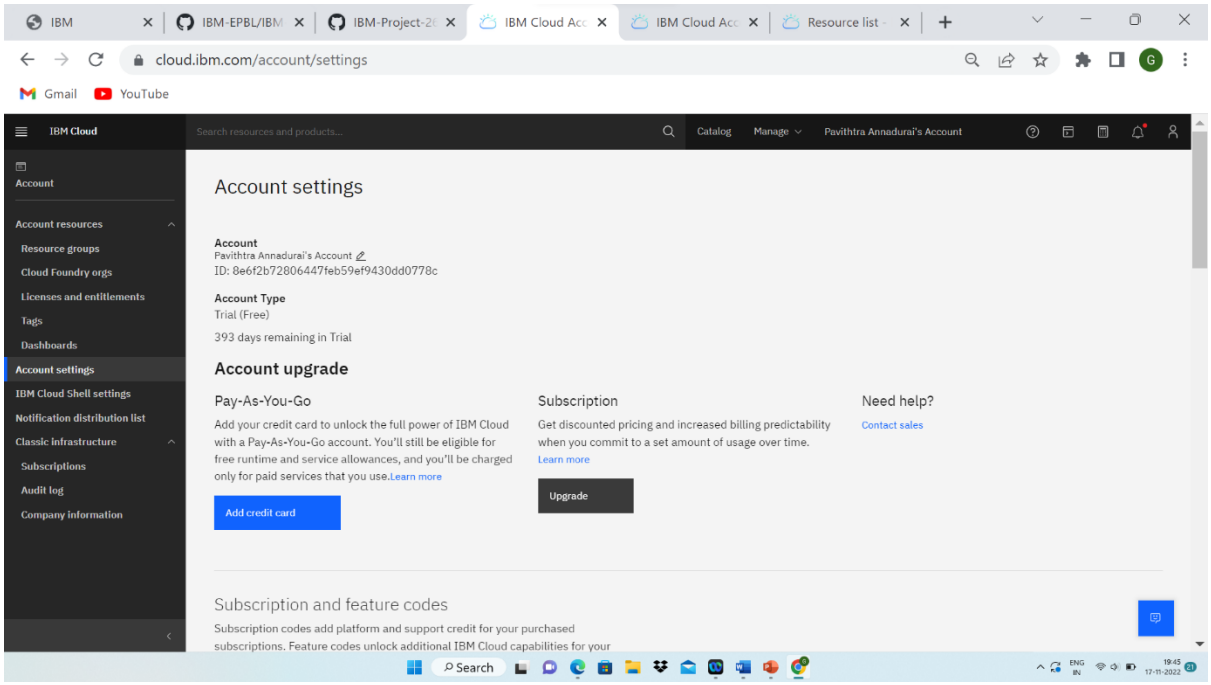


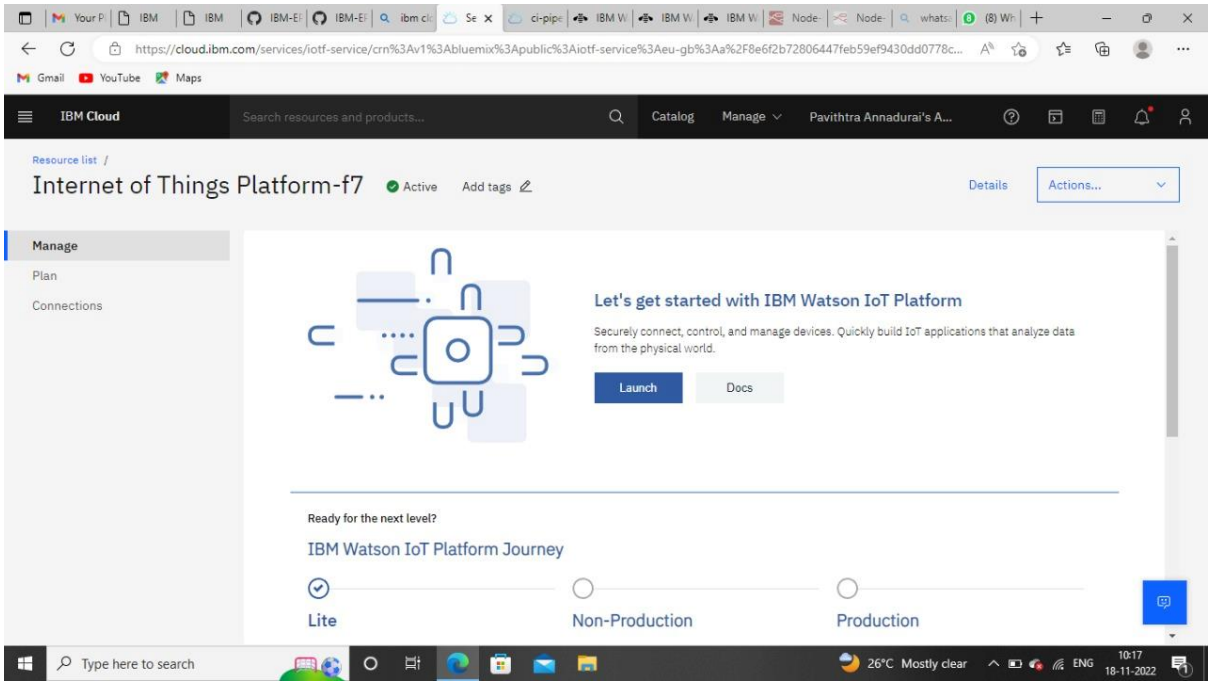
8.CREATE AND CONFIGURE IBM CLOUD SERVICE:

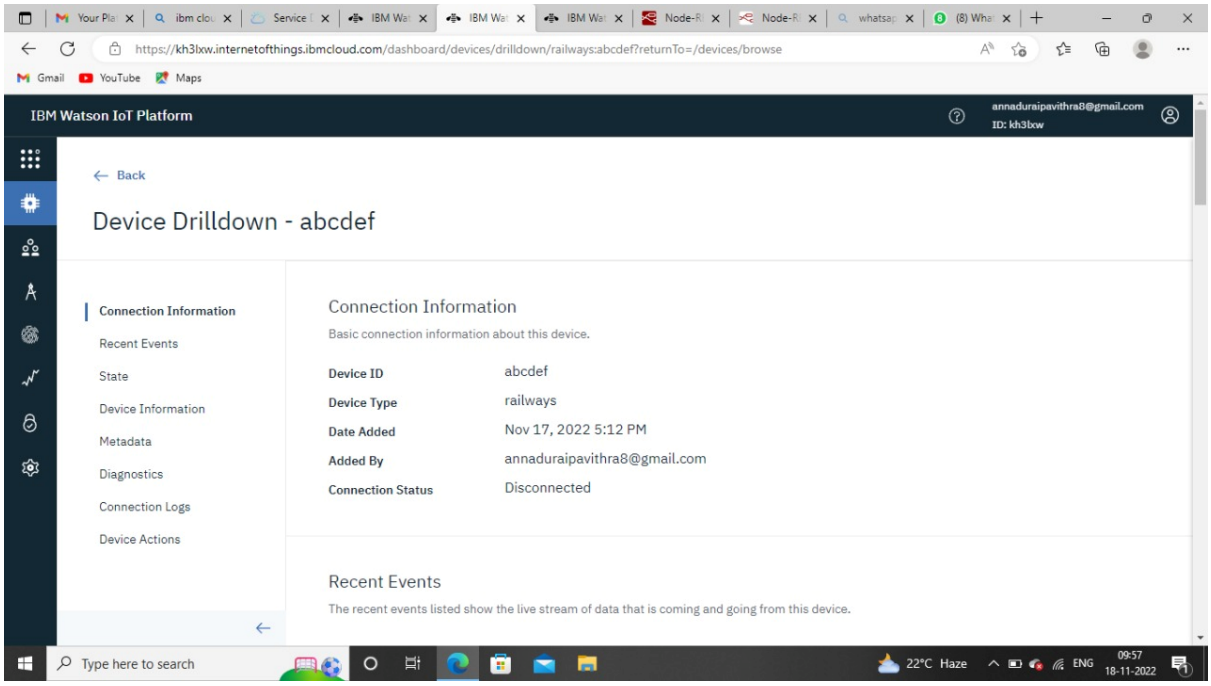
8.1 IBM WATSON PLATFORM:



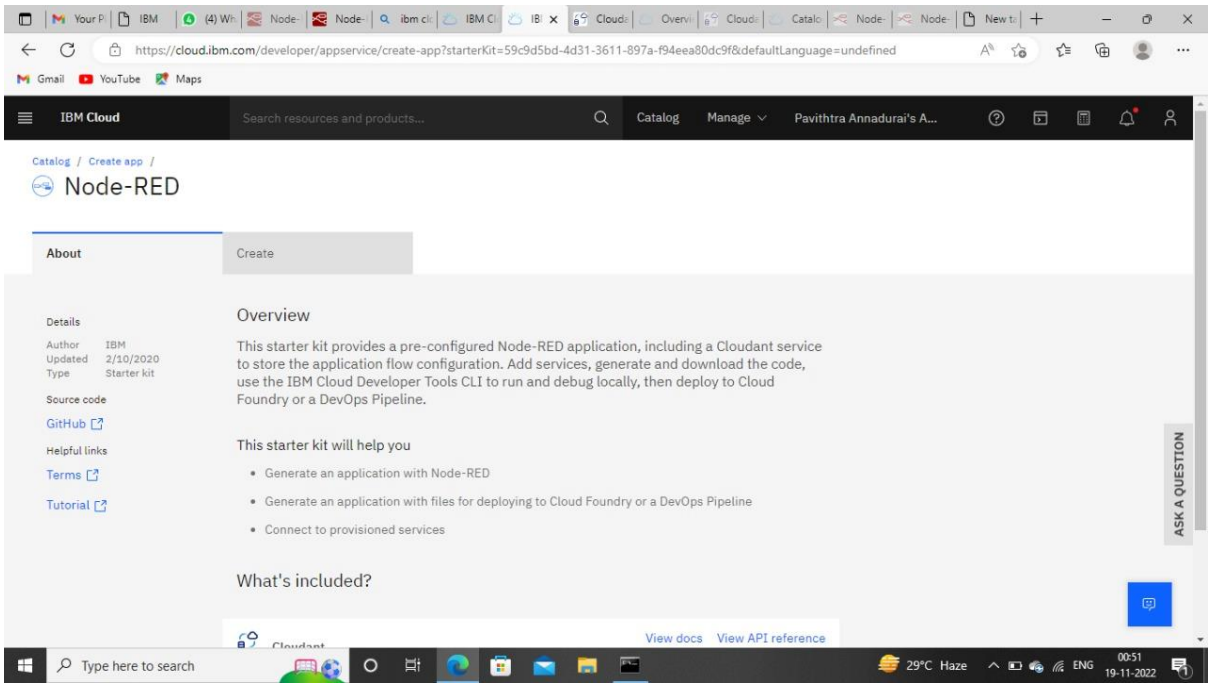

cloud.ibm.com/catalog/services/internet-of-things-platform
IBM Cloud
Catalog
Manage
Pavithtra Annadurai's Account
Internet of Things Platform
This service is the hub of all things IBM IoT, it is where you can set up and manage your connected devices so that your apps can access their live and historical data.
Create
About
Type
Service
Provider
IBM
Last updated
08/15/2022
Category
Internet of Things
Compliance
IAM-enabled
Location
Frankfurt
London
Dallas
Washington DC
Related links
Docs
Terms
Select a location
Frankfurt (eu-de)
Select a pricing plan
Displayed prices do not include tax. Monthly prices shown are for country or location: United States
Plan
Features
Pricing
Lite
Includes up to 500 registered devices, and a maximum of 200 MB of each data metric
Maximum of 500 registered devices
Maximum of 500 application bindings
Maximum of 200 MB of each of data exchanged, data analyzed and edge data analyzed
Free
The Lite service plan for Internet of Things Platform includes up to 500 registered devices, and a maximum of 200 MB each of data exchanged, data analyzed, and edge data analyzed per month.
Summary
Internet of Things Platform
Free
Location: Frankfurt
Plan: Lite
Service name: Internet of Things Platform-sq
Resource group: Default
I have read and agree to the following license agreements:
Terms
Create
Add to estimate

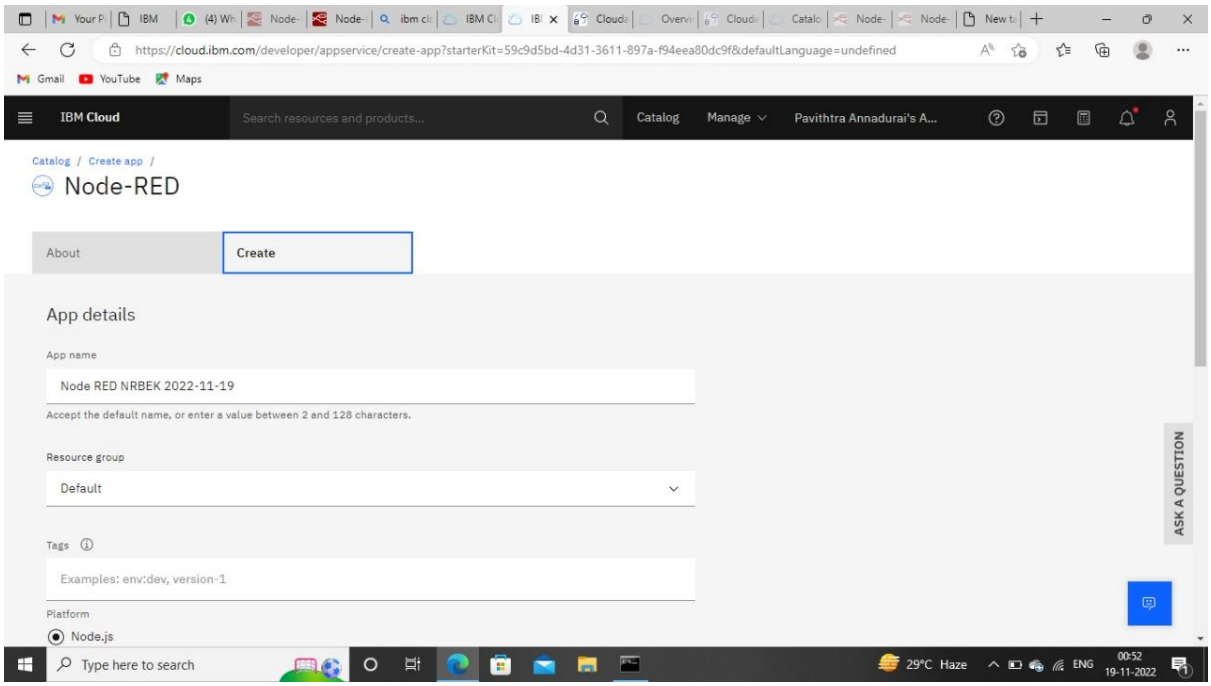

cloud.ibm.com/account/settings
IBM Cloud
Catalog
Manage
Pavithtra Annadurai's Account
Account
Account resources
Resource groups
Cloud Foundry orgs
Licenses and entitlements
Tags
Dashboards
Account settings
IBM Cloud Shell settings
Notification distribution list
Classic infrastructure
Subscriptions
Audit log
Company information
Account settings
Account
Pavithtra Annadurai's Account
ID: 8e6f2b72806447feb59ef9430dd0778c
Account Type
Trial (Free)
393 days remaining in Trial
Account upgrade
Pay-As-You-Go
Add your credit card to unlock the full power of IBM Cloud with a Pay-As-You-Go account. You'll still be eligible for free runtime and service allowances, and you'll be charged only for paid services that you use. Learn more
Add credit card
Subscription
Get discounted pricing and increased billing predictability when you commit to a set amount of usage over time.
Learn more
Upgrade
Need help?
Contact sales
Subscription and feature codes
Subscription codes add platform and support credit for your purchased subscriptions. Feature codes unlock additional IBM Cloud capabilities for your


IBM Cloud
Catalog
Manage
Pavithtra Annadurai's A...
Resource list /
Internet of Things Platform-f7
Active
Add tags
Details
Actions...
Manage
Plan
Connections
Let's get started with IBM Watson IoT Platform
Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world.
Launch
Docs
Ready for the next level?
IBM Watson IoT Platform Journey
Lite
Non-Production
Production


IBM Watson IoT Platform
annaduraipavithra8@gmail.com
ID: kh3lxw
Back
Device Drilldown - abcdef
Connection Information
Recent Events
State
Device Information
Metadata
Diagnostics
Connection Logs
Device Actions
Connection Information
Basic connection information about this device.
Device ID abcdef
Device Type railways
Date Added Nov 17, 2022 5:12 PM
Added By annaduraipavithra8@gmail.com
Connection Status Disconnected
Recent Events
The recent events listed show the live stream of data that is coming and going from this device.

8.2 NODE-RED SERVICE:





https://cloud.ibm.com/developer/appservice/apps/fdc6cc40-54d3-4dc2-8b2c-5ee9c80057e9

IBM Cloud | Search resources and products... | Catalog | Manage | Pavithtra Annadurai's A...

Resource list / App details /

Node RED WSHPT 2022-11-18

Add tags | Actions...

Details

App URL	http://169.51.204.13:31352
Source	https://us-south.git.cloud.ibm.com/annaduraipavithra8/NodeREDW...
Resource group	Default
Deployment target	Kube/Helm
Created	11/17/2022

Services

Db2

Open dashboard | Documentation | API reference

Credentials

Deployment Automation

Name	NodeREDWSHPT2022-11-18
Location	Dallas
Tool integrations	

Delivery Pipelines

Name	ci-pipeline
Status	Success
Name	pr-pipeline
Status	No stages detected

ASK A QUESTION

Type here to search | 29°C Haze | ENG | 00:54 19-11-2022

Not secure | 169.51.204.13:31352

Node-RED on IBM Cloud

Node-RED

Flow-based programming for the Internet of Things

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

This instance is running as an IBM Cloud application, giving it access to the wide range of services available on the platform.

More information about Node-RED, including documentation, can be found at nodered.org.

Go to your Node-RED flow editor

Learn how to customise Node-RED

Customising your instance of Node-RED

This instance of Node-RED is enough to get you started creating flows.

You may want to customise it for your needs, for example replacing this introduction page with your own, adding http authentication to the flow

Type here to search | 29°C Haze | ENG | 00:54 19-11-2022

9.CODING AND SOLUTIONING:

pythoncode(1).py - C:/Users/gopik/Downloads/pythoncode(1).py (3.7.1)

File Edit Format Run Options Window Help

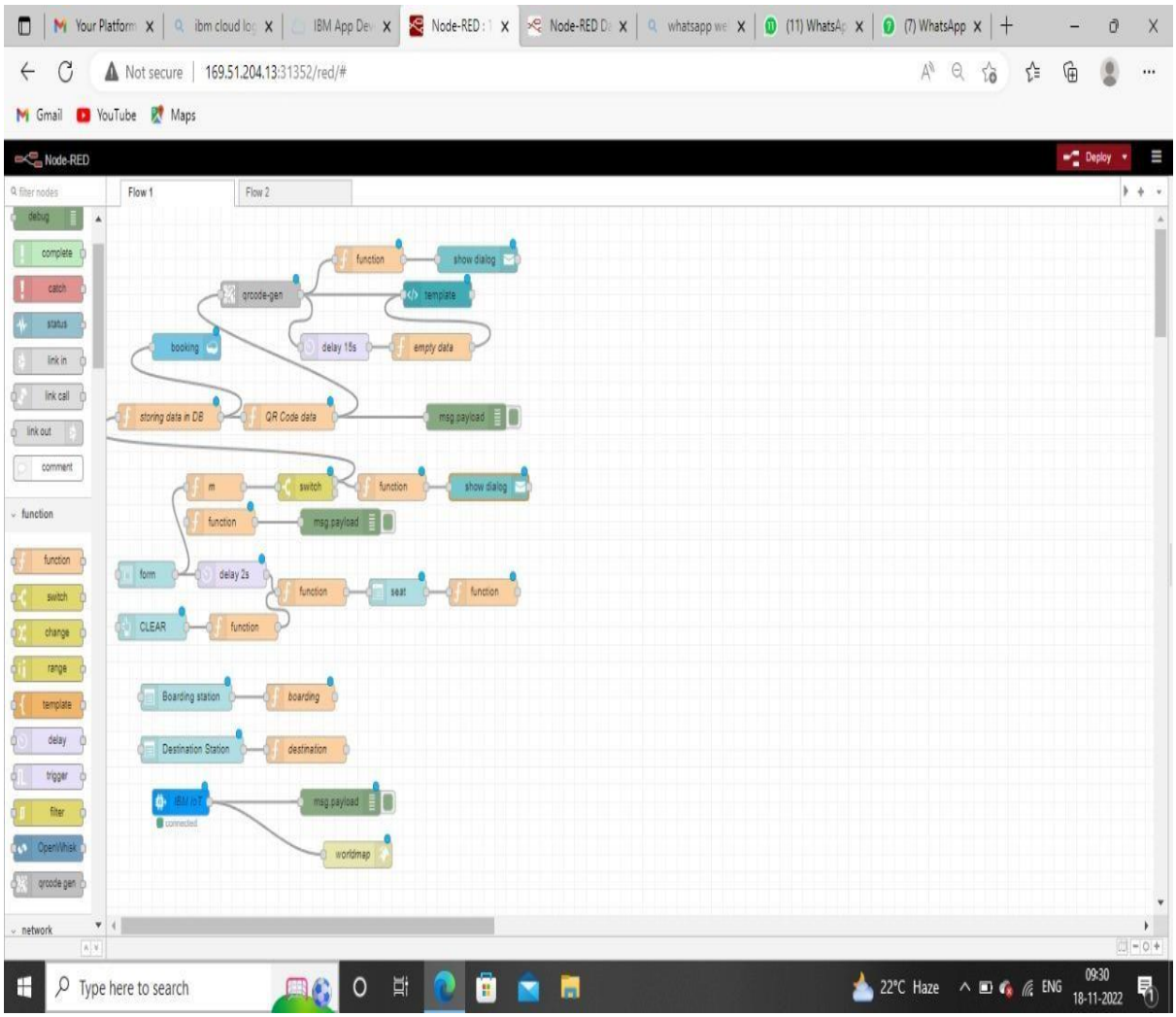
Python Shell
Check Module Alt+X
Run Module F5

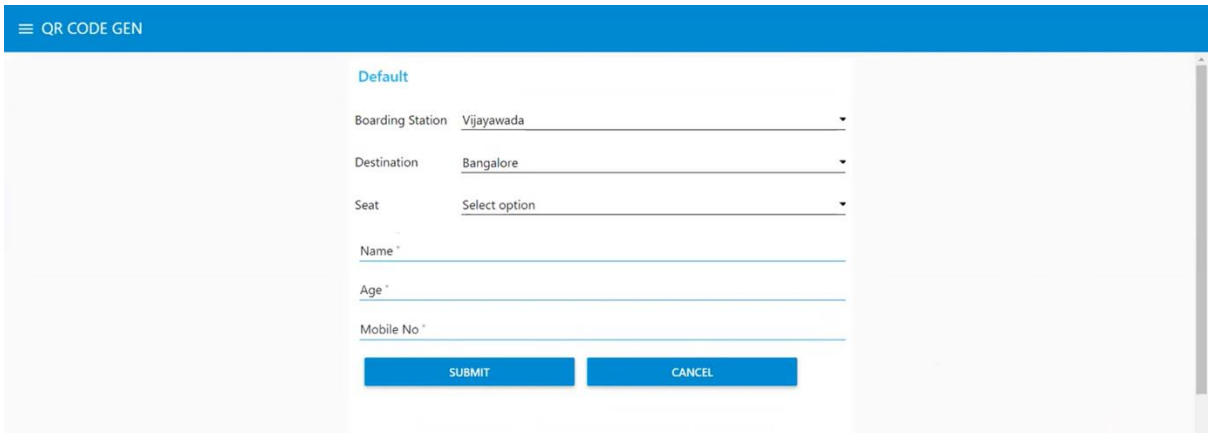
```
import wiotp.
import time
import random
myConfig = {
        "identity": {
                "orgId": "kh31xw",
                "typeId": "pyscript",
                "deviceId":"script"

        },
        "auth": {
                "token": "XFbQaps87Lb0xzGr_h"
        }
}

def myCommandCallback (cmd):
        print ("Message received from IBM IoT Platform: %s" % cmd.data['command'])
        m=cmd.data['command']

client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

def pub (data):
        client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
        print ("Published data Successfully: %s", myData)

while True:
        myData={'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
        pub (myData)
        time.sleep (3)
        #myData={'name': 'Train2', 'lat': 17.6387448, 'lon': 78.4754336}
        #pub (myData)
        #time.sleep (3)
        myData={'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
        pub(myData)
        time.sleep(3)
        myData={'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
        pub (myData)
        time.sleep (3)
        myData={'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
        pub (myData)
```

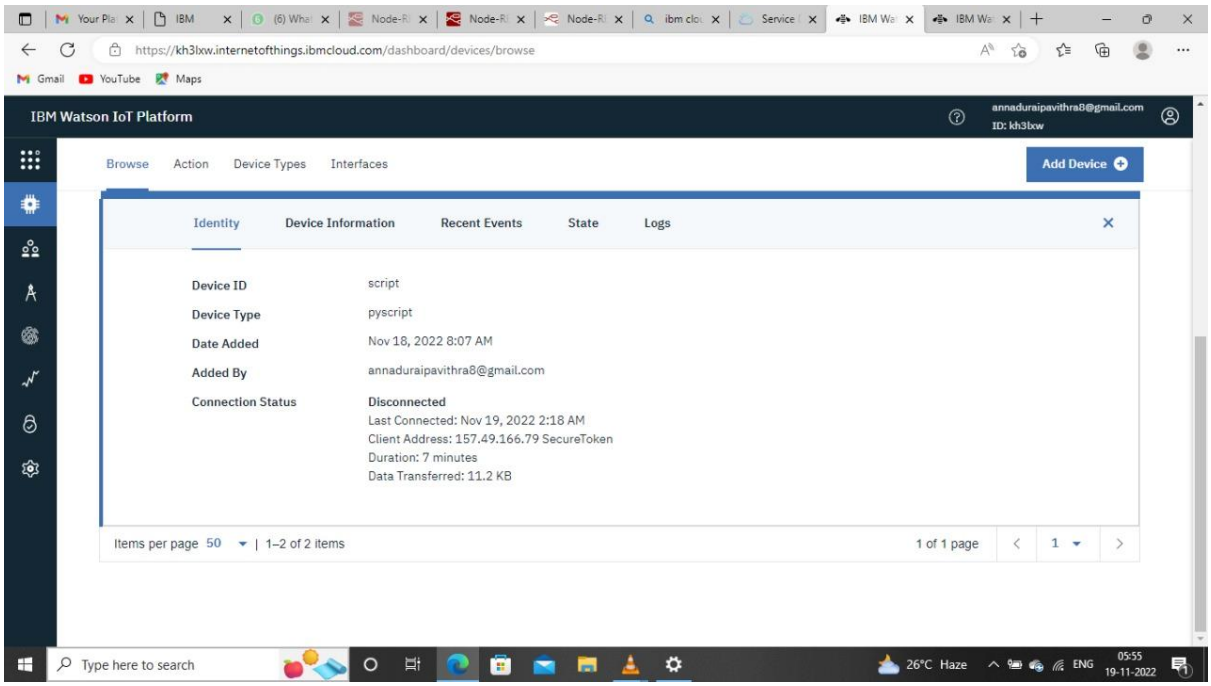
Ln: 23 Col: 16

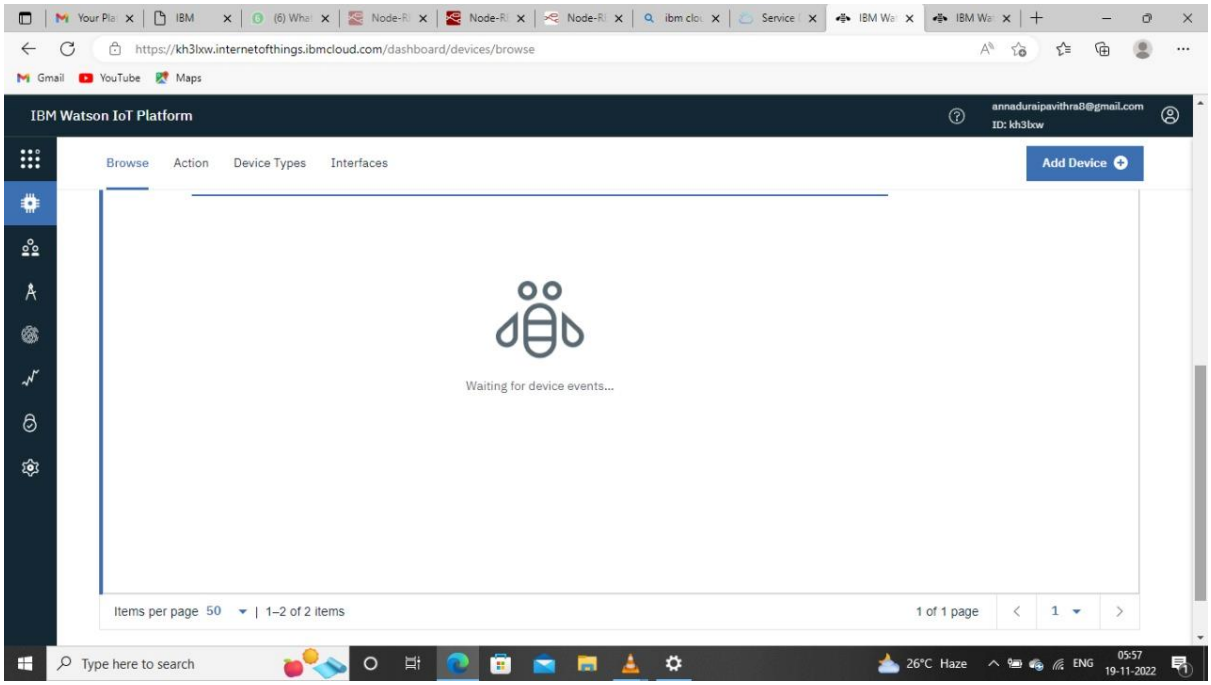
10.TESTING:

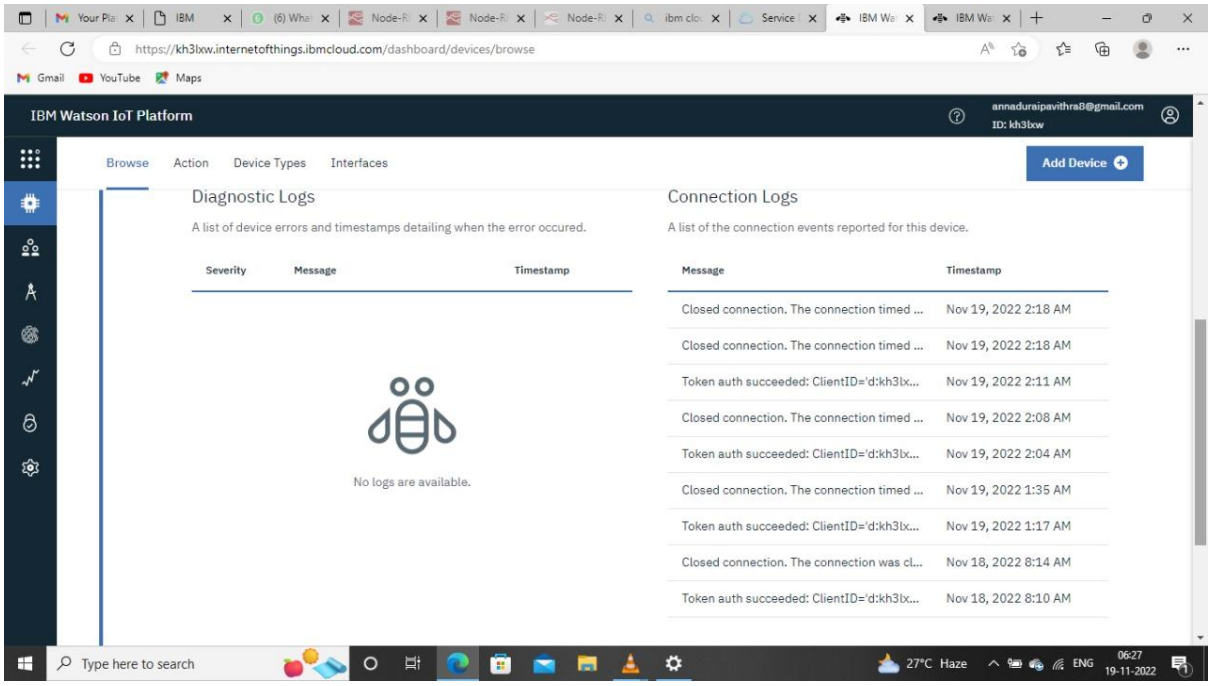
WEB APPLICATION USING NODE RED:

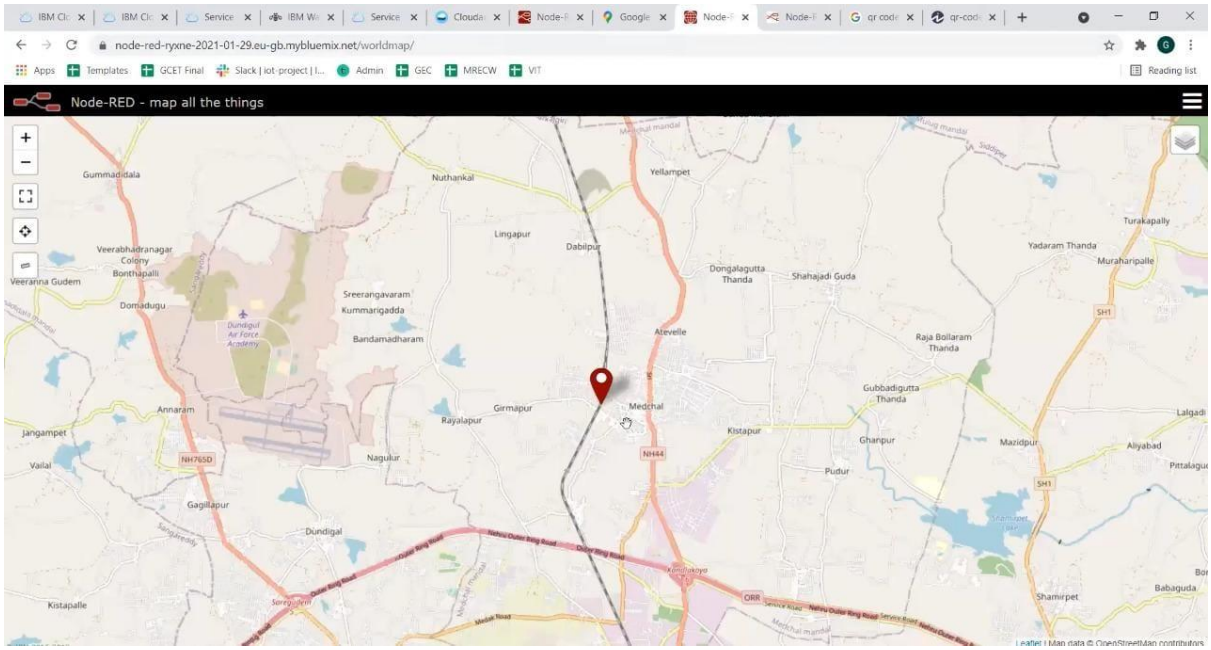

Node-RED
Flow 1
Flow 2
Deploy
function
show dialog
qrcode-gen
template
booking
delay 15s
empty data
storing data in DB
QR Code data
msg.payload
m
switch
function
show dialog
function
msg.payload
form
delay 2s
function
seat
function
CLEAR
function
Boarding station
boarding
Destination Station
destination
IBM IoT
msg.payload
worldmap

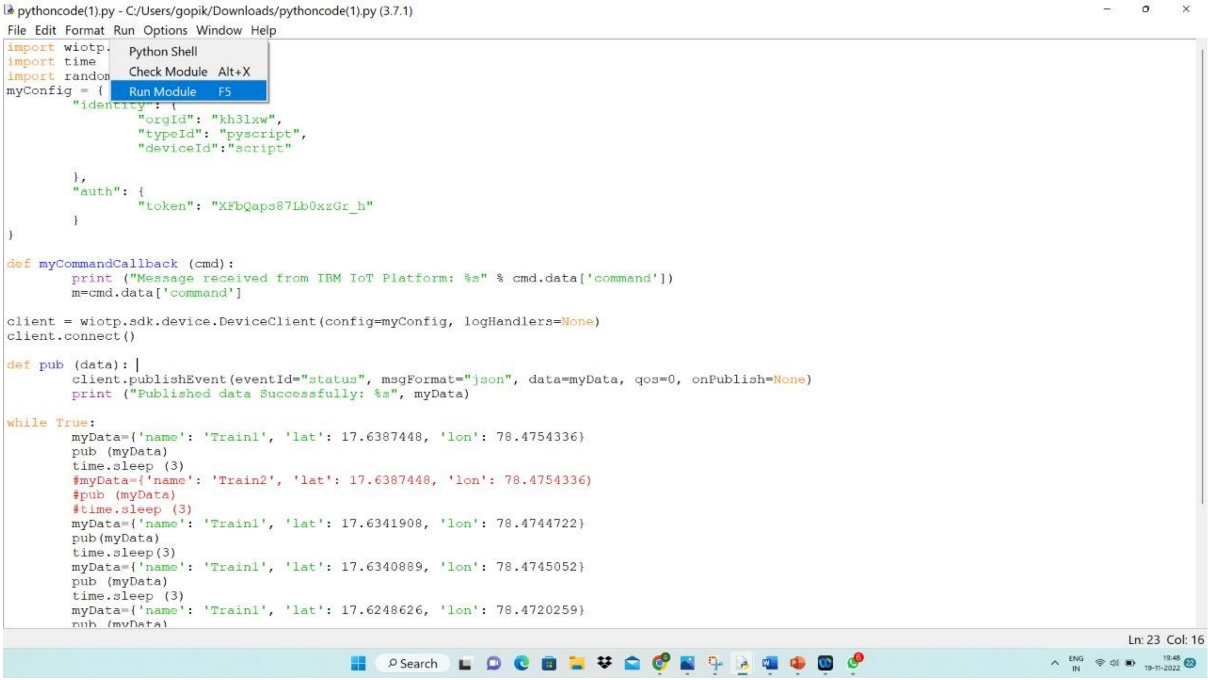

QR CODE GEN
Default
Boarding Station
Vijayawada
Destination
Bangalore
Seat
Select option
Name *
Age *
Mobile No *
SUBMIT
CANCEL


https://kh3lxw.internetofthings.ibmcloud.com/dashboard/devices/browse
IBM Watson IoT Platform
annaduraipavithra8@gmail.com
ID: kh3lxw
Browse
Action
Device Types
Interfaces
Add Device
Identity
Device Information
Recent Events
State
Logs
Device ID
script
Device Type
pyscript
Date Added
Nov 18, 2022 8:07 AM
Added By
annaduraipavithra8@gmail.com
Connection Status
Disconnected
Last Connected: Nov 19, 2022 2:18 AM
Client Address: 157.49.166.79 SecureToken
Duration: 7 minutes
Data Transferred: 11.2 KB
Items per page 50
1–2 of 2 items
1 of 1 page
Type here to search
26°C Haze
ENG
05:55
19-11-2022

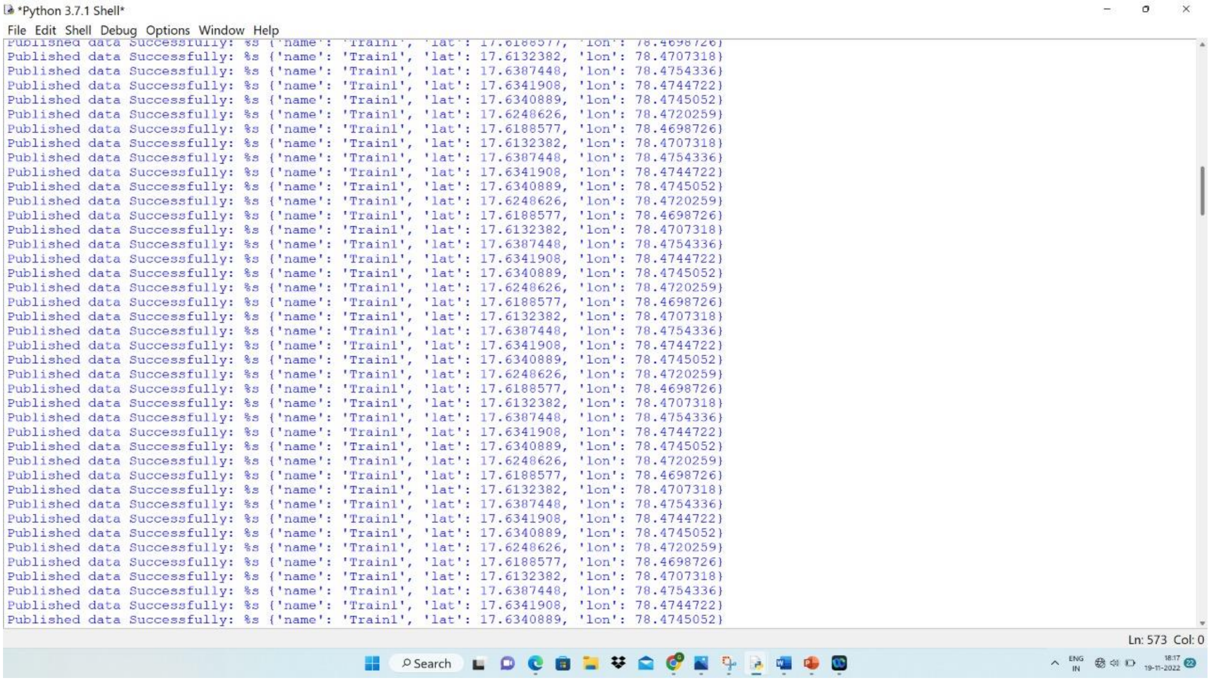

https://kh3lxw.internetofthings.ibmcloud.com/dashboard/devices/browse
IBM Watson IoT Platform
annaduraipavithra8@gmail.com
ID: kh3lxw
Browse
Action
Device Types
Interfaces
Add Device
Waiting for device events...
Items per page 50
1–2 of 2 items
1 of 1 page
Type here to search
26°C Haze
ENG
05:57
19-11-2022


https://kh3lxw.internetofthings.ibmcloud.com/dashboard/devices/browse
IBM Watson IoT Platform
annaduraipavithra8@gmail.com
ID: kh3lxw
Browse
Action
Device Types
Interfaces
Add Device
Diagnostic Logs
A list of device errors and timestamps detailing when the error occured.
Severity
Message
Timestamp
No logs are available.
Connection Logs
A list of the connection events reported for this device.
Message
Timestamp
Closed connection. The connection timed ... Nov 19, 2022 2:18 AM
Closed connection. The connection timed ... Nov 19, 2022 2:18 AM
Token auth succeeded: ClientID='d:kh3lx... Nov 19, 2022 2:11 AM
Closed connection. The connection timed ... Nov 19, 2022 2:08 AM
Token auth succeeded: ClientID='d:kh3lx... Nov 19, 2022 2:04 AM
Closed connection. The connection timed ... Nov 19, 2022 1:35 AM
Token auth succeeded: ClientID='d:kh3lx... Nov 19, 2022 1:17 AM
Closed connection. The connection was cl... Nov 18, 2022 8:14 AM
Token auth succeeded: ClientID='d:kh3lx... Nov 18, 2022 8:10 AM
Type here to search
27°C Haze
ENG
06:27
19-11-2022

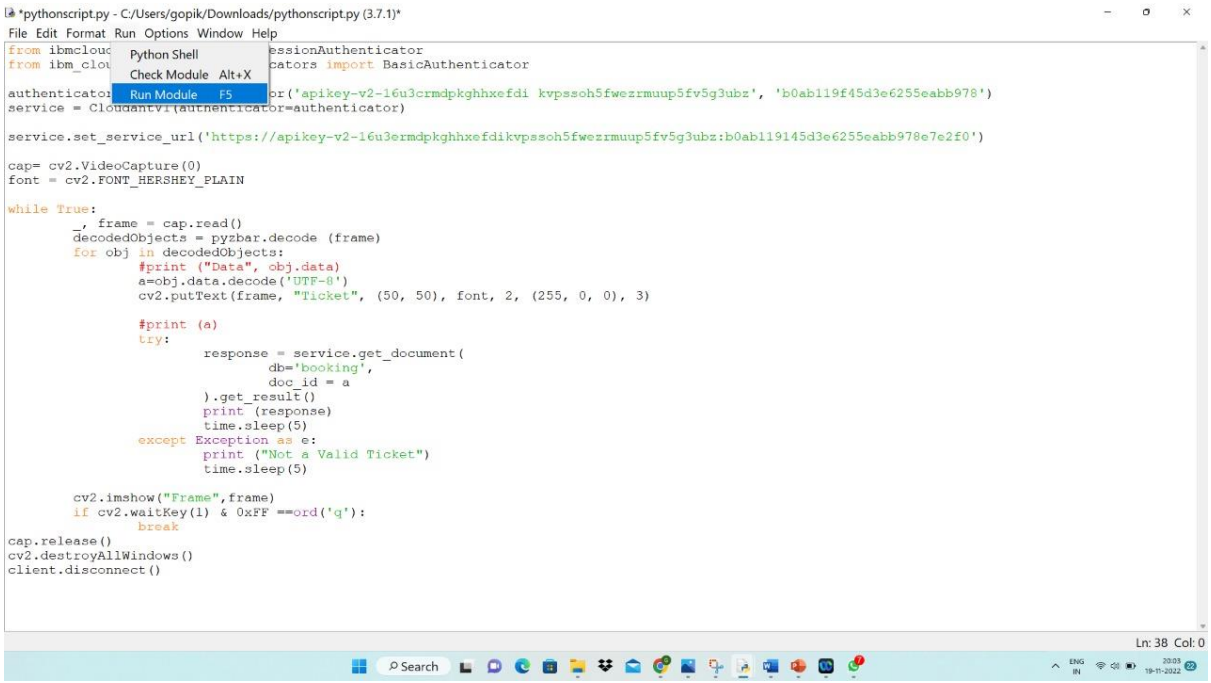

node-red-ryxne-2021-01-29.eu-gb.mybluemix.net/worldmap/
Node-RED - map all the things
Gummadidala
Nuthankal
Yellampet
Lingapur
Dabilpur
Veerabhadranagar Colony
Bonthapalli
Veeranna Gudem
Domadugu
Sreerangavaram
Kummarigadda
Bandamadharam
Dongalagutta Thanda
Shahajadi Guda
Turakapally
Yadaram Thanda
Muraharipalle
Atevelle
Raja Bollaram Thanda
Gubbadigutta Thanda
Annaram
Girmapur
Medchal
Rayalapur
Kistapur
Ghanpur
Mazidpur
Aliyabad
Jangampet
Vailal
Nagulur
Pudur
Gagillapur
Dundigal
Kistapalle
Shamirpet
Babaguda
Leaflet | Map data © OpenStreetMap contributors



```python
import wiotp.
import time
import random
myConfig = {
        "identity": {
                "orgId": "kh3lxw",
                "typeId": "pyscript",
                "deviceId":"script"

        },
        "auth": {
                "token": "XFbQaps87Lb0xzGr_h"
        }
}

def myCommandCallback (cmd):
        print ("Message received from IBM IoT Platform: %s" % cmd.data['command'])
        m=cmd.data['command']

client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

def pub (data):
        client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
        print ("Published data Successfully: %s", myData)

while True:
        myData={'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
        pub (myData)
        time.sleep (3)
        #myData={'name': 'Train2', 'lat': 17.6387448, 'lon': 78.4754336)
        #pub (myData)
        #time.sleep (3)
        myData={'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
        pub(myData)
        time.sleep(3)
        myData={'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
        pub (myData)
        time.sleep (3)
        myData={'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
        pub (myData)
```



```
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
```



11.RESULTS:

11.1 PERFORMANCE METRICES:

12.ADVANTAGES AND DISADVANTAGES:

12.1 ADVANTAGES:

Easily accessible to anyone.passengers can save time by avoid standing in queue and avoid missing of trains .User can track the current location of train.Explore train routes while traveling.passenger details are displayed just by scanning the QR code.Achieving high safety measures. Safest journey experience by real time monitoring.

12.2 DISADVANTAGES:

Network issue may affect the continuous monitoring system.Network and technical may occur .uneducated people may not have knowledge about this application.

13.CONCLUSION:

An IOT based smart solutions for railways using Watson IOT platform, Watson simulator, IBM cloud and Node-RED.

14.APPENDIX:

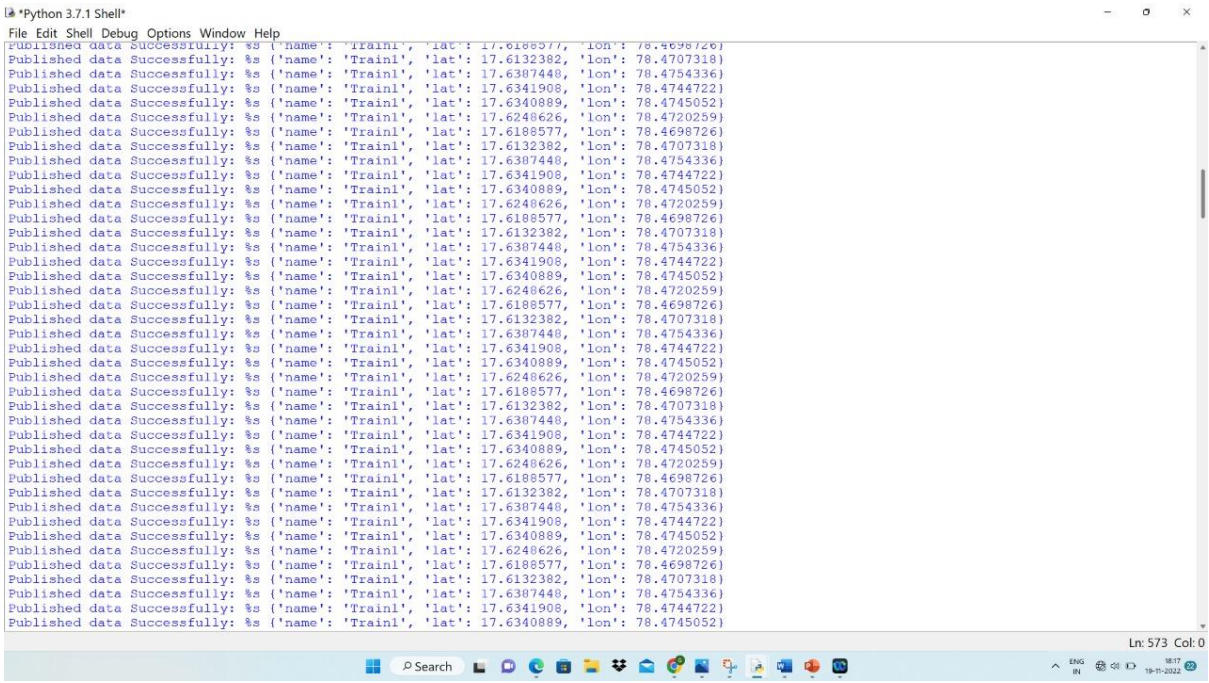
SOURCE CODE:

```
import wiotp.sdk.device

import time

import random

myConfig = {

        "identity": {

                "orgId": "kh3lxw",

                "typeId": "pyscript",

                "deviceId":"script"


        },

        "auth": {

                "token": "XFbQaps87Lb0xzGr_h"

        }

}
```

```
def myCommandCallback (cmd):
        print ("Message received from IBM IoT Platform: %s" % cmd.data['command'])
        m=cmd.data['command']

client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

def pub (data):
        client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
        print ("Published data Successfully: %s", myData)

while True:
        myData={'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
        pub (myData)
        time.sleep (3)
        #myData={'name': 'Train2', 'lat': 17.6387448, 'lon': 78.4754336)
        #pub (myData)
        #time.sleep (3)
        myData={'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
        pub(myData)
        time.sleep(3)
        myData={'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
        pub (myData)
        time.sleep (3)
        myData={'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
        pub (myData)
        time.sleep (3)
        myData={'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
```

```
        pub (myData)
        time.sleep (3)
        myData={'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
        pub (myData)
        time.sleep (3)
        client.commandCallback = myCommandCallback
client.disconnect ()
```

OUTPUT:



```
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6248626, 'lon': 78.4720259}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6188577, 'lon': 78.4698726}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6132382, 'lon': 78.4707318}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6387448, 'lon': 78.4754336}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6341908, 'lon': 78.4744722}
Published data Successfully: %s {'name': 'Train1', 'lat': 17.6340889, 'lon': 78.4745052}
```