

SPRINT-4

Date	10 November 2022
Team ID	PNT2022TMID00940
Project Name	SMART SOLUTIONS FOR RAILWAYS

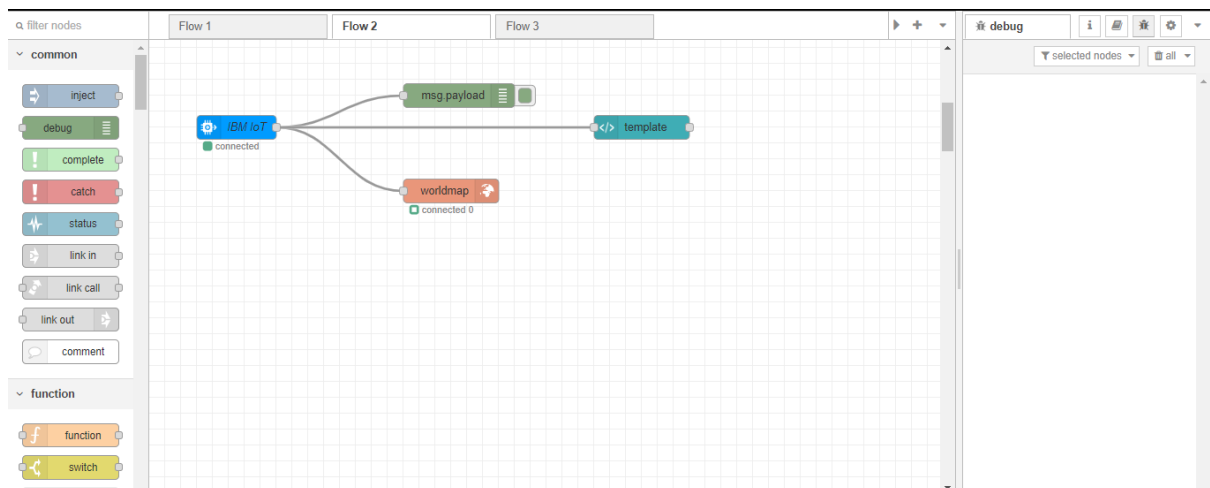
WEB APP CREATION AND TESTING

An IOT device “GPS” is created at IBM Watson:

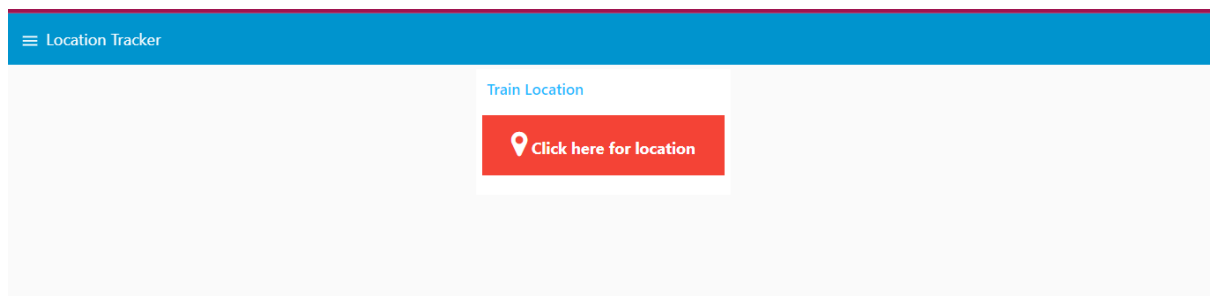
The screenshot displays the IBM Watson IoT Platform interface. At the top, the header shows the user's email (abhinesh1801@gmail.com) and ID (ojhri). The main navigation bar includes tabs for Browse, Action, Device Types, and Interfaces. A sidebar on the left contains various icons for navigation. The central area shows a table of devices with columns for Device ID, Status, Device Type, Class ID, and Date Added. A device with ID 12345, status Disconnected, and type GPS is highlighted. Below the table, a detailed view of the selected device is shown, including its identity, device information, recent events, state, and logs. The device information section lists the following details:

- Device ID: 12345
- Device Type: GPS
- Date Added: Nov 9, 2022 8:04 PM
- Added By: abhinesh1801@gmail.com
- Connection Status: Disconnected
- Last Connected: Nov 12, 2022 10:57 AM
- Client Address: 49.37.219.83 SecureToken
- Duration: 5 minutes
- Data Transferred: 11.2 KB

Creating a Node-Red Connection using the IBM IOT (connected with GPS(IBM Watson device)) for tracking the location of the train:



Creating a WEB UI with Node-Red to see the location:



Link:

<https://node-red-gitmx-2022-11-08.eu-gb.mybluemix.net/ui/>

Connecting the GPS Device with our Python Code to stimulate the locations:

```
code.py - C:\Users\ABINESH\AppData\Local\Programs\Python\Python37\smart solutions for railways\code.py (3.7.0b4)
File Edit Format Run Options Window Help

import wiotp.sdk.device
import time
import random
myConfig = {
    "identity": {
        "orgId": "ojhlri",
        "typeId": "GPS",
        "deviceId": "12345"
    },
    "auth": {
        "token": "12345678"
    }
}

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)
```

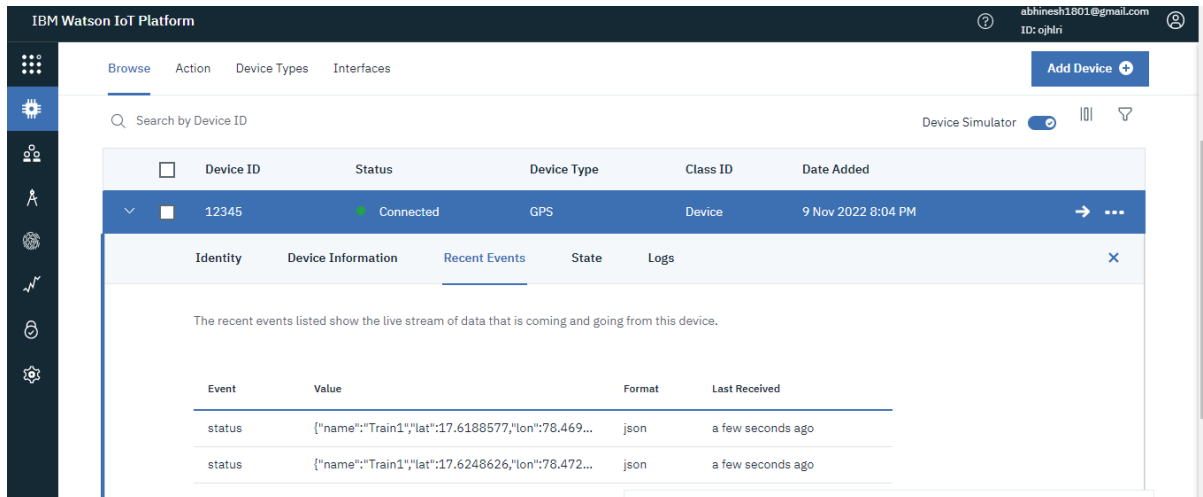
After linking ,the python code is made to run at python IDE:

```
*Python 3.7.0b4 Shell
File Edit Shell Debug Options Window Help

Python 3.7.0b4 (v3.7.0b4:eb96c37699, May 2 2018, 19:02:22) [MSC v.1913 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\ABINESH\AppData\Local\Programs\Python\Python37\smart solutions for railways\code.py
2022-11-12 14:49:30,375 wiotp.sdk.device.client.DeviceClient INFO Connected successfully: d:ojhlri:GPS:12345Published 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}
|
```

The IOT Device is conneted and the locations are Received as output.

Watson Output:



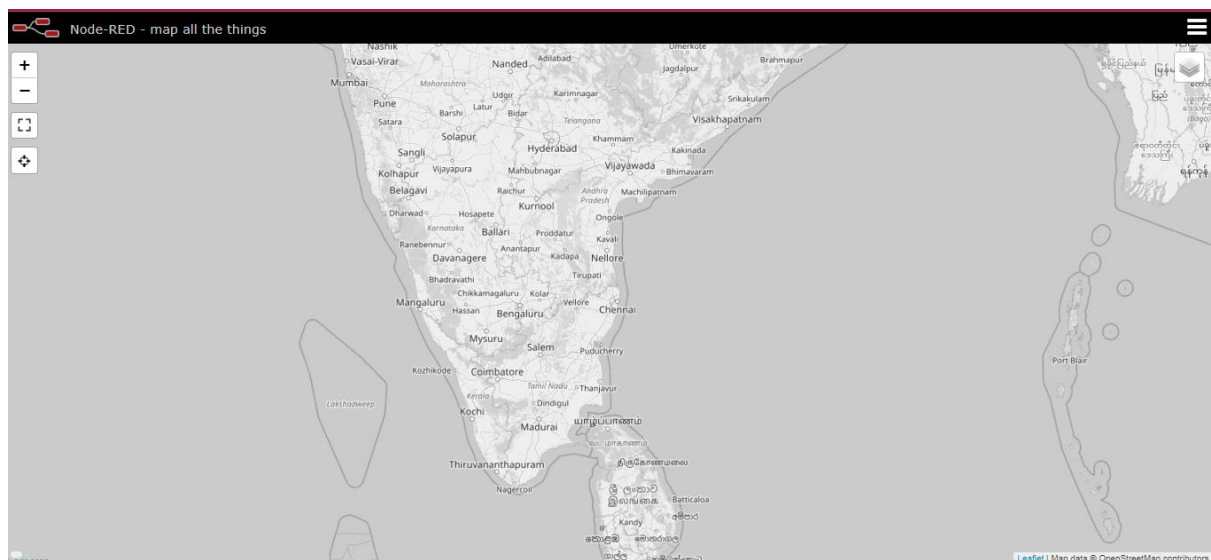
The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is present with the text 'Search by Device ID'. The main content area shows a table with columns: Device ID, Status, Device Type, Class ID, and Date Added. A device with ID 12345 is listed as 'Connected' with a 'GPS' device type. Below this, a 'Recent Events' tab is selected, showing a table of events. The events table has columns: Event, Value, Format, and Last Received. Two events are shown, both with the event name 'status' and a JSON value containing location coordinates. The last received time for both is 'a few seconds ago'.

Device ID	Status	Device Type	Class ID	Date Added
12345	Connected	GPS	Device	9 Nov 2022 8:04 PM

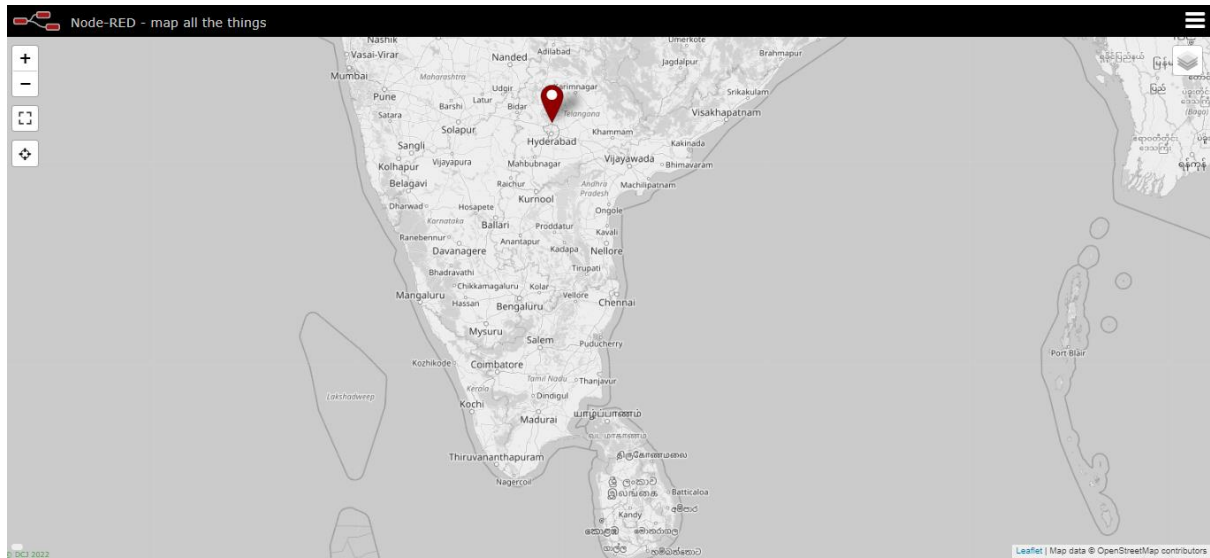
Event	Value	Format	Last Received
status	{"name":"Train1","lat":17.6188577,"lon":78.469...	json	a few seconds ago
status	{"name":"Train1","lat":17.6248626,"lon":78.472...	json	a few seconds ago

Initially before the execution of python program:

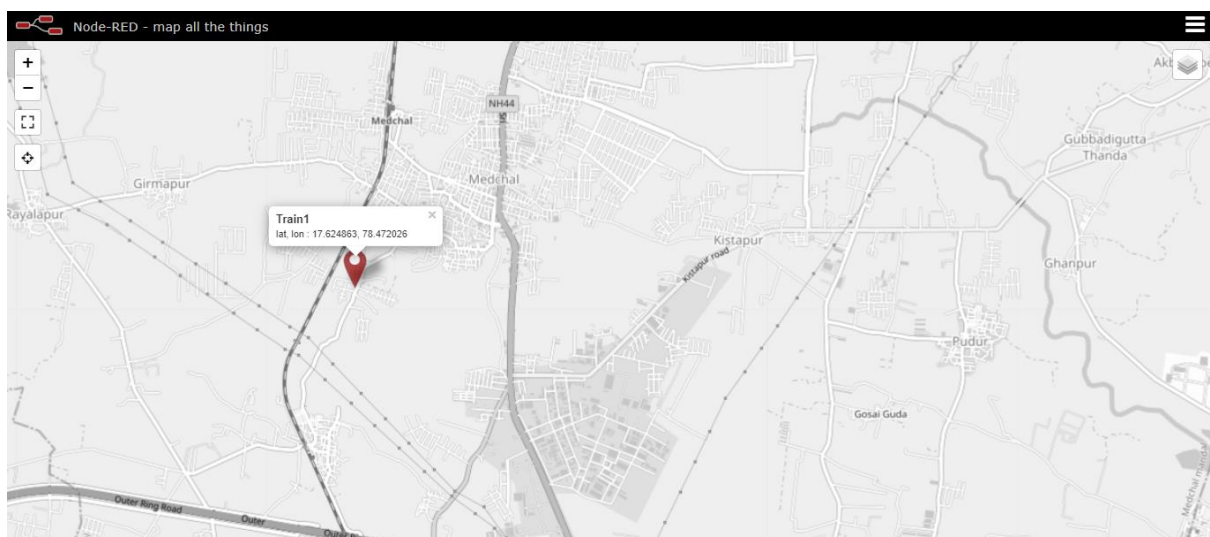
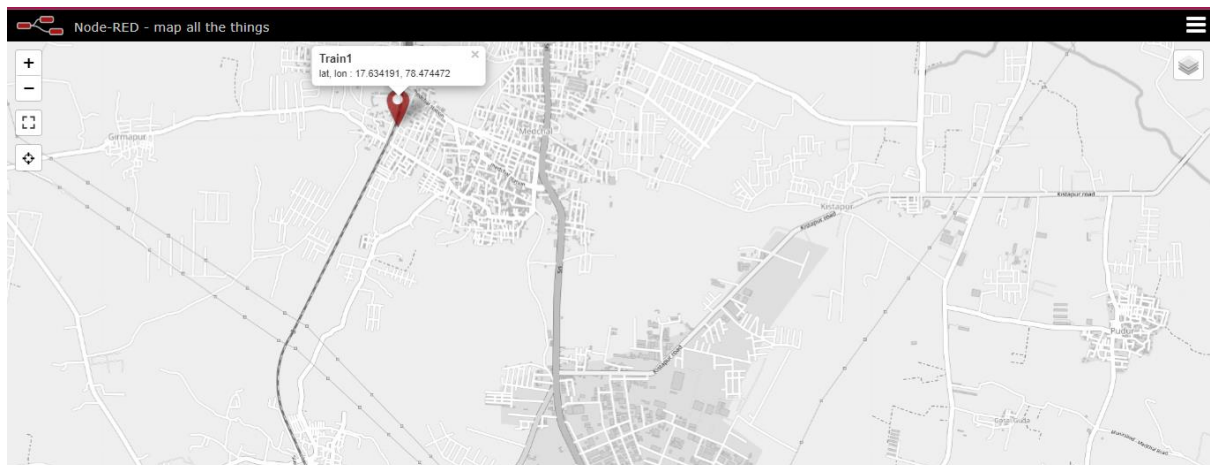
No location found,

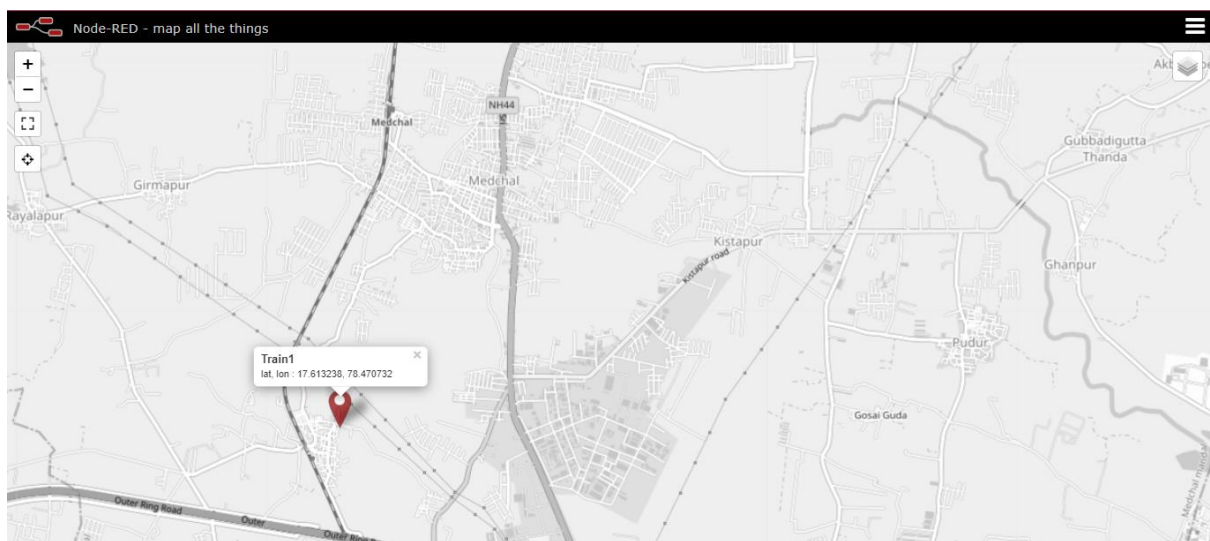
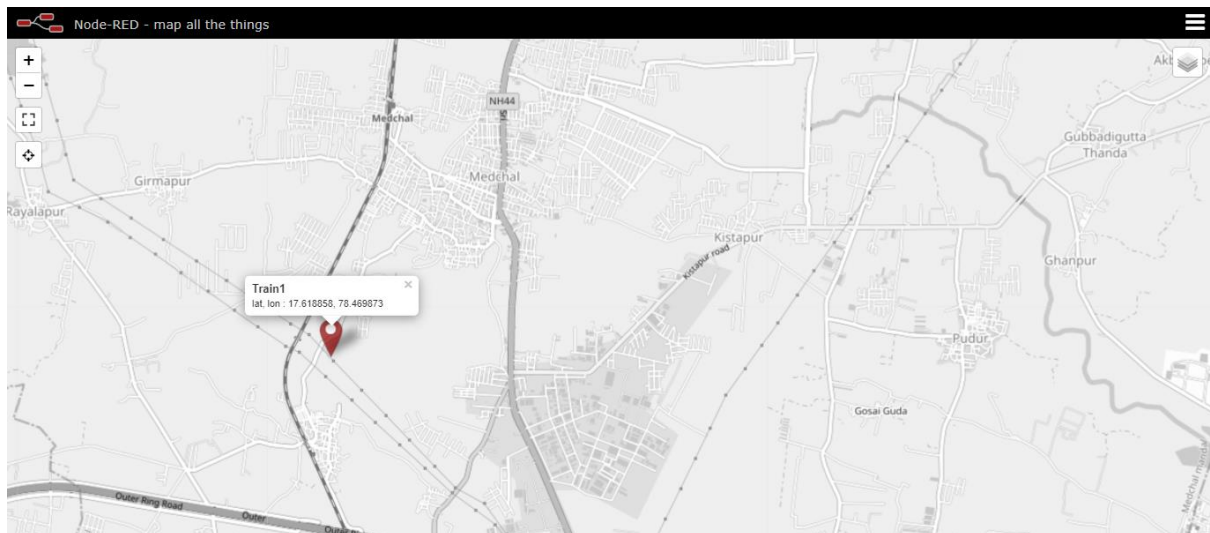


After execution,



Below Four images of map show the movement of the train:

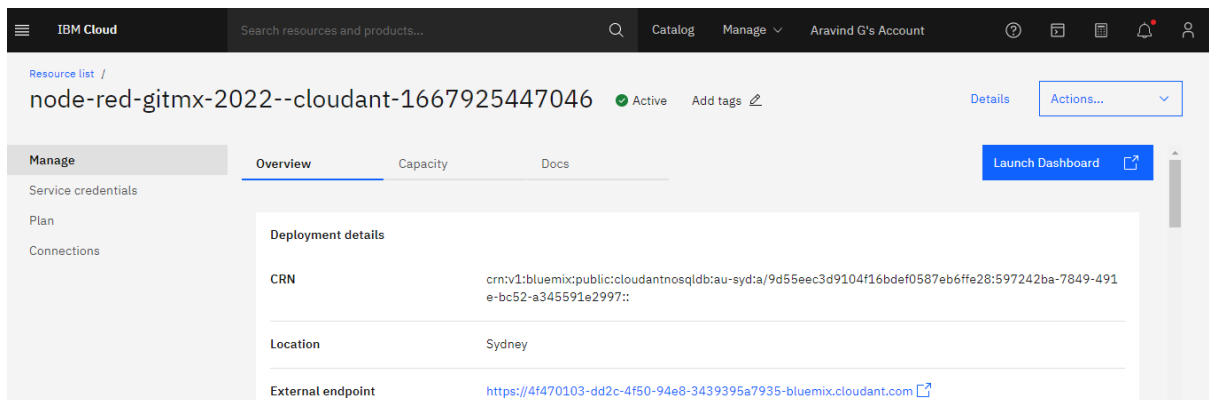


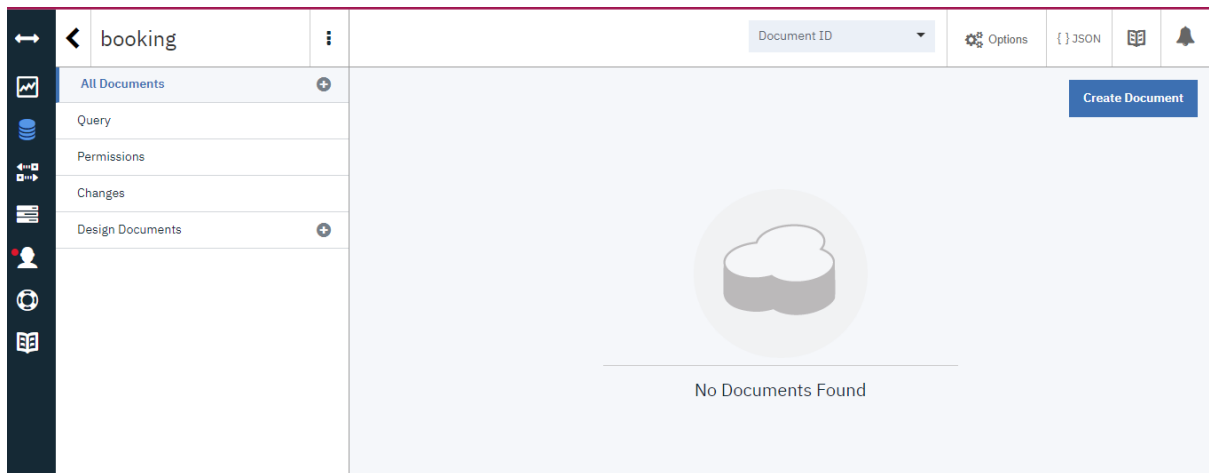


LINK FOR THE MAP VIEW:

<https://node-red-gitmx-2022-11-08.eu-gb.mybluemix.net/worldmap/>

Node-Red Connection for Ticket Booking and QR Generation:





WEB Application created from Node-Red:

A screenshot of a web application titled 'QR Code Gen'. The main content area is titled 'Booking Corner'. It contains a form with the following elements: 'Boarding Station' dropdown menu with 'Chennai' selected; 'Destination' dropdown menu with 'Bangalore' selected; 'Seat' dropdown menu with 'Select o...' selected; three input fields labeled 'Name *', 'Age *', and 'Mobile Number *'; and three buttons: 'SUBMIT', 'CANCEL', and 'CLEAR'.

LINK FOR Booking Corner:

<https://node-red-gitmx-2022-11-08.eu-gb.mybluemix.net/ui/>

Booking Happens :

QR Code Gen

Booking Corner

Boarding Station

Hyderabad

Destination

Vijayawada

Seat

2

Name *

Arun

Age *

21

Mobile Number *

753216497

SUBMIT

CANCEL

CLEAR

QR Code Gen

Booking Corner

Boarding Station

Hyderabad

Destination

Chennai

Seat

Select o...

Name *

Age *

Mobile Number *

SUBMIT

CANCEL

CLEAR

Ticket is Generated

OK

QR Generation:

QR Code Gen

Age *

Mobile Number *

SUBMIT

CANCEL

CLEAR



Details will be stored at Cloudant Database:

The screenshot shows the Cloudant web interface for a database named 'booking'. On the left is a sidebar with navigation icons. The main area has a top bar with 'booking' and a 'Document ID' dropdown. Below this is a 'Query' section with tabs for 'Table', 'Metadata', and 'JSON'. The 'Table' tab is active, displaying a table with columns: Boarding, Destination, Name, Age, and Mobile Number. The first row of data is: Hyderabad, Chennai, Arun, 21, 753216497. A 'Create Document' button is visible in the top right.

The screenshot shows the Cloudant web interface for a document in the 'booking' database. The document ID is '2022-11-12,15:27:37'. The 'JSON' tab is active, displaying the document's data in JSON format. The data is:

```
{
  "_id": "2022-11-12,15:27:37",
  "_rev": "1-c31e1d7186bd8ccf6104252d74a3e8bd",
  "Name": "Arun",
  "Age": 21,
  "Mobile Number": 753216497,
  "Boarding": "Hyderabad",
  "Destination": "Chennai"
}
```

Building a python code for the Ticket Collector to scan the QR and get details from Cloud:

```
QR_Scanner.py - C:\Users\ABINESH\AppData\Local\Programs\Python\Python37\smart solutions for railways\QR_Scanner.py (3.7.0b4)
File Edit Format Run Options Window Help

import cv2
import numpy as np
import time
import pyzbar.pyzbar as pyzbar
from ibmcloudant.cloudant_v1 import CloudantV1
from ibmcloudant import CouchDbSessionAuthenticator
from ibm_cloud_sdk_core.authenticators import BasicAuthenticator

authenticator = BasicAuthenticator('apikey-267fe83f32ec443b804f3c76d688d2e3', 'b1456cbe3c712c2d555623fe')
service = CloudantV1(authenticator=authenticator)

service.set_service_url('https://4f470103-dd2c-4f50-94e8-3439395a7935-bluemix.cloudant.com')

cap= cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_PLAIN

while True:
    _, frame = cap.read()
    decodedObjects = pyzbar.decode(frame)
    for obj in decodedObjects:
        #print ("Data", obj.data)
        a=obj.data.decode('UTF-8')
        cv2.putText(frame, "Ticket", (50, 50), font, 2, (255, 0, 0), 3)

        #print (a)
        try:
            response = service.get_document(
                db='booking',
                doc id = a
```

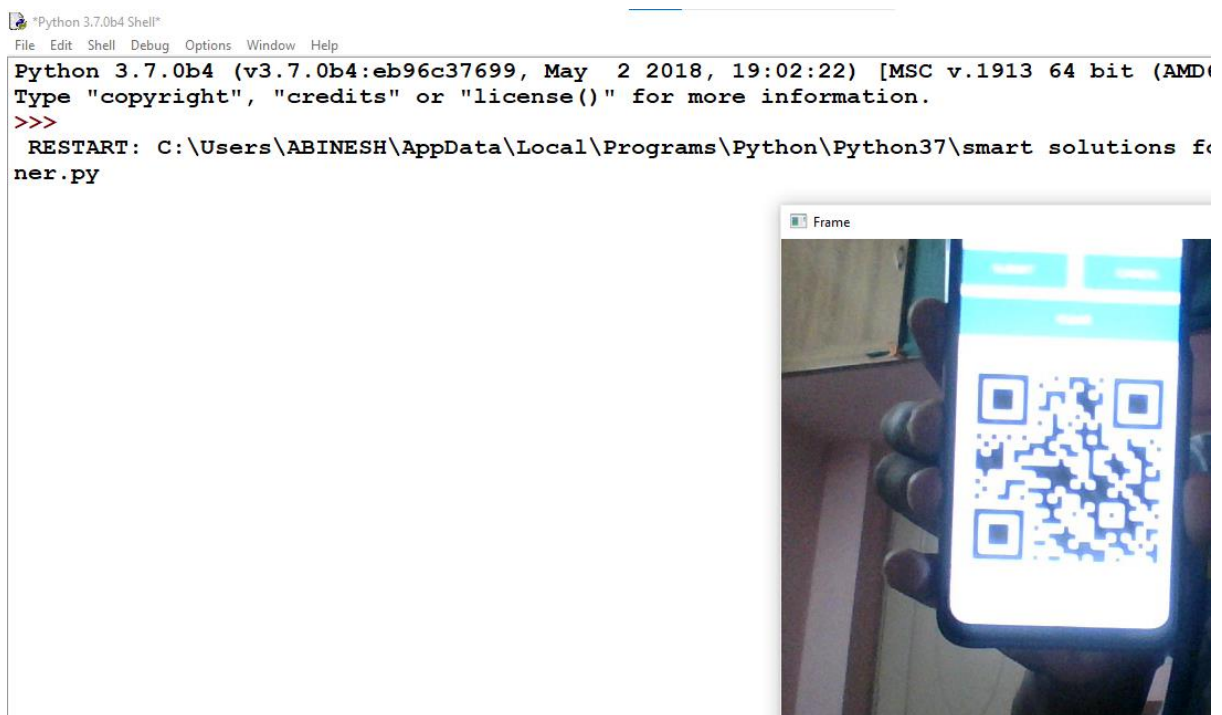
TESTING:

Executing the program:

Scanner Opens :

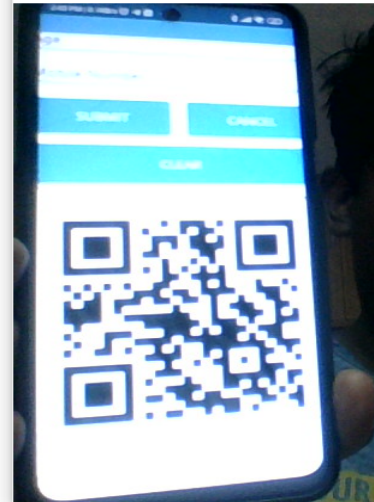


QR is shown:



Data Fetched from Cloud by scanning the QR:

```
Python 3.7.0b4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0b4 (v3.7.0b4:eb96c37699, May 2 2018, 19:02:22) [MSC v.1913 64 bi
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\ABINESH\AppData\Local\Programs\Python\Python37\smart solut
ner.py
{'_id': '2022-11-12,15:27:37', '_rev': '1-c31e1d7186bd8ccf6104252d74a3e8bd',
, 'Mobile Number': 753216497, 'Boarding': 'Hyderabad',
```



```
Python 3.7.0b4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0b4 (v3.7.0b4:eb96c37699, May 2 2018, 19:02:22) [MSC v.1913 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\ABINESH\AppData\Local\Programs\Python\Python37\smart solutions for railways\QR_Scan
ner.py
{'_id': '2022-11-12,15:27:37', '_rev': '1-c31e1d7186bd8ccf6104252d74a3e8bd', 'Name': 'Arun', 'Age': 21
, 'Mobile Number': 753216497, 'Boarding': 'Hyderabad', 'Destination': 'Chennai'}
```

Showing a Random Qr From Google:



Output: (Shows “Not a Valid Ticket”)

```
Python 3.7.0b4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0b4 (v3.7.0b4:eb96c37699, May 2 2018, 19:02:22) [MSC v.1913 64 b
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\ABINESH\AppData\Local\Programs\Python\Python37\smart solu
ner.py
{'_id': '2022-11-12,15:27:37', '_rev': '1-c31e1d7186bd8ccf6104252d74a3e8bd',
, 'Mobile Number': 753216497, 'Boarding': 'Hyderabad',
Not a Valid Ticket
```



```
Python 3.7.0b4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0b4 (v3.7.0b4:eb96c37699, May 2 2018, 19:02:22) [MSC v.1913 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\ABINESH\AppData\Local\Programs\Python\Python37\smart solutions for railways\QR_Scan
ner.py
{'_id': '2022-11-12,15:27:37', '_rev': '1-c31e1d7186bd8ccf6104252d74a3e8bd', 'Name': 'Arun', 'Age': 21
, 'Mobile Number': 753216497, 'Boarding': 'Hyderabad', 'Destination': 'Chennai'}
Not a Valid Ticket
```