DHANALAKSHMI SRINIVASAN COLLEGE OF ENGINERING ANDTECHNOLOGY

Project Report

Smart Solution for Railways

TEAM ID : PNT2022TMID26872

TEAM LADER : RAJALAKSHMI N
TEAM MEMBER 1 : KEERTHANA M
TEAM MEMBER 2 : KEERTHIGA R
TEAM MEMBER 3 : MEGALA A

1. INTRODUCTION
Project Overview
Purpose
2. LITERATURE SURVEY
Existing problem
References
Problem Statement Definition
3. IDEATION & PROPOSED SOLUTION
Empathy Map Canvas
Ideation & Brainstorming
Proposed Solution
Problem Solution fit
4. REQUIREMENT ANALYSIS
Functional requirement
Non-Functional requirements
5. PROJECT DESIGN
Data Flow Diagrams
Solution & Technical Architecture
User Stories
6. PROJECT PLANNING & SCHEDULING
Sprint Planning & Estimation
Sprint Delivery Schedule
Reports from JIRA
7. CODING & SOLUTIONING (Explain the features added in the project along with code)
Feature 1
Feature 2
Database Schema (if Applicable)
8. TESTING
Test Cases
User Acceptance Testing
9. RESULTS
Performance Metrics
10. ADVANTAGES & DISADVANTAGES
11. CONCLUSION
12. FUTURE SCOPE
13. APPENDIX
Source Code | GitHub & Project Demo Link

INTRODUCTION

1. INTRODUCTION

1.1 PROJECT OVERVIEW

SMART SOLUTIONS FOR RAILWAYS is to manage Indian Railways is the largest railway network in Asia and additionally world's second largest network operated underneath a single management. Due to its large size it is difficult to monitor the cracks in tracks manually. This paper deals with this problem and detects cracks in tracks with the help of ultrasonic sensor attached to moving assembly with help of stepper motor. Ultrasonic sensor allows the device to moves back and forth across the track and if there is any fault, it gives information to the cloud server through which railway department is informed on time about cracks and many lives can be saved. This is the application of IoT, due to this it is cost effective system. This effective methodology of continuous observation and assessment of rail tracks might facilitate to stop accidents. This methodology endlessly monitors the rail stress, evaluate the results and provide the rail break alerts such as potential buckling conditions, bending of rails and wheel impact load detection to the concerned authorities.

1.2. PURPOSE

Internet is basically system of interconnected computers through network. But now its use is changing with changing world and it is not just confined to emails or web browsing. Today's internet also deals with embedded sensors and has led to development of smart homes, smart rural area, e-health care's etc. and this introduced the concept of IoT . Internet of Things refers to interconnection or communication between two or more devices without human-to-human and human-to-computer interaction. Connected devices are equipped with sensors or actuators perceive their surroundings. IOT has four major components which include sensing the device, accessing the device, processing the information of the device, and provides application and services. In addition to this it also provides security and privacy of data . Automation has affected every aspect of our daily lives. More improvements are being introduced in almost all fields to reduce human effort and save time. Thinking of the same is trying to introduce automation in the field of track testing. Railroad track is an integral part of any company's asset base, since it provides them with the necessary business functionality. Problems that occur due to problems in railroads need to be overcome. The latest method used by the Indian railroad is the tracking of the train track which requires a lot of manpower and is time-consuming.

LITERATURE SURVEY

2. LITERATURE SURVEY

EXISTING SYSTEM

In the Existing train tracks are manually researched. LED (Light Emitting Diode) and LDR (Light Dependent Resister) sensors cannot be implemented on the block of the tracks]. The input image processing is a clamorous system with high cost and does not give the exact result. The Automated Visual Test Method is a complicated method as the video colour inspection is implemented to examine the cracks in rail track which does not give accurate result in bad weather. This traditional system delays transfer of information. Srivastava et al., (2017) proposed a moving gadget to detect the cracks with the help of an array of IR sensors to identify the actual position of the cracks as well as notify to nearest railway station. Mishra et al., (2019) developed a system to track the cracks with the help of Arduino mega power using solar energy and laser. A GSM along with a GPS module was implemented to get the actual location of the faulty tracks to inform the authorities using SMS via a link to find actual location on Google Maps. Rizvi Aliza Raza presented a prototype in that is capable of capturing photos of the track and compare it with the old database and sends a message to the authorities regarding the crack detected. The detailed analysis of traditional railway track fault detection techniques is explained in table

REFERENCES

1. “5G Key Technologies for Smart Railways” Bo At: Andreas F. Molisch: Markus Rupp: Zhang-Dui Zhong Proceedings of the IEEE Year: 2020 Volume: 108.

2. Internet of Things for Smart Railway: Feasibility and Applications Ohyun Jo: Yong-kyu kim: Juyeop kim IEEE Internet of Things Journal Year: 2018 Volume: 5.

3. Controlling Railway Gates Using Smart Phones By tracking trains with GPS R.Velayutham; T. Sangeethavani; K. Sundaralakshmi 2017 International Conference on Circuit Power and Computing Technologies (ICCPCT).

4. “Automated Level Crossings - A Futuristic solution Enabling Smart City Infrastructure” Saxena 2017 IEEE National Aerospace And Electronics Conference(NAECON) Year: 2019

5. “The IDex Case Study on the Safety Measures of AloT-based Railway Infrastructures” Chung Kit Wu: Yaging He; kim Fung Tsang: Stefan Mozar 2020 IEEE International Symposium on Product Compliance Engineering-Asia (SPCE-CN) Year: 2020 Conference Paper

6. “Analysis of Experimental Railway Point Electric Heating System” Ruslans Muhitovs; Mareks Mezitis; Andrejs Spunitis; Vladimirs Iriskovs 2020 IEEE 8th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE) Year: 2021

7. “Electrical Power Distribution Design & Voltage Profile Improvement for Metro Railway Station” Mansi R. Patel; Nirali A. Rathod; Brijal Mehta 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4) Year: 2020

8. “Improved modelling for wind turbines on trains” Mario Hyman; Mohd. Hasan Ali 2019 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT) Year: 2019

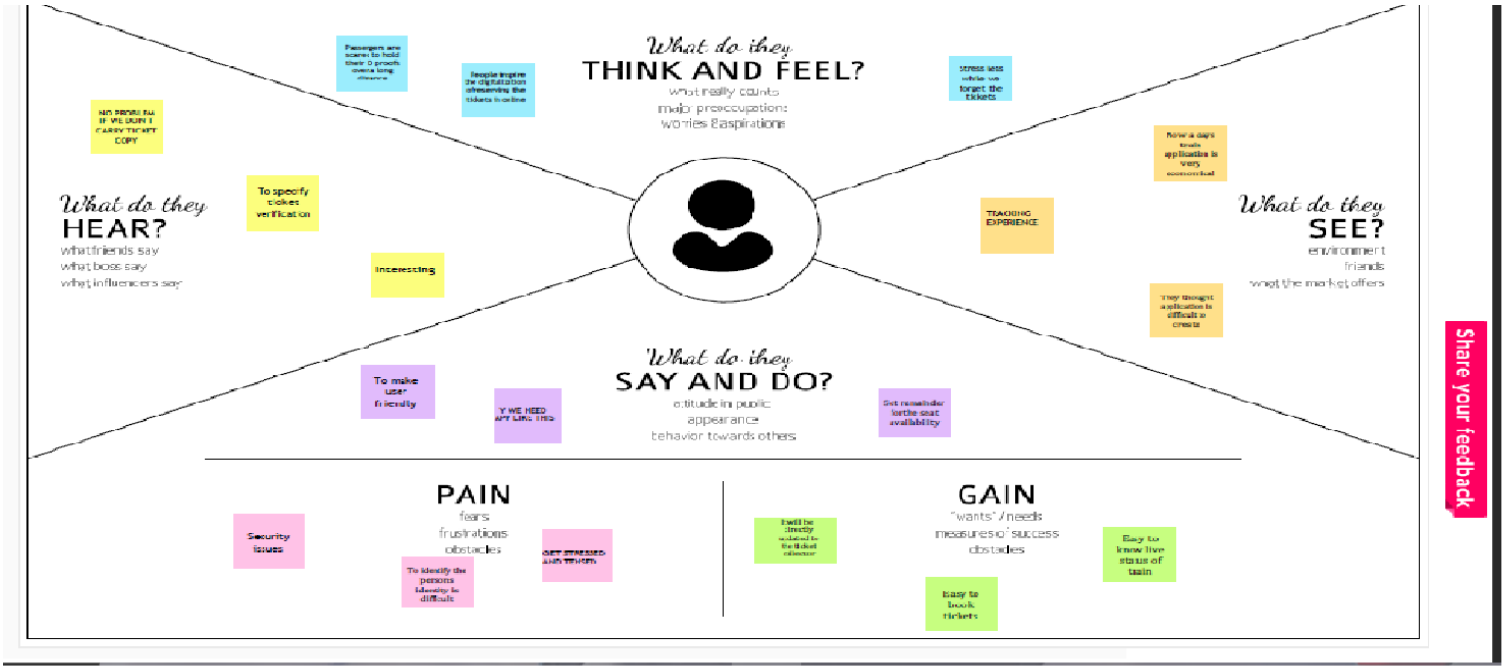
PROBLEM STATEMENT DEFINITION

Among the various modes of transport, railways is one of the biggest modes of transport in the world. Though there are competitive threats from airlines, luxury buses, public transports, and personalized transports the problem statement is to answer the question “What are the problems faced by the passengers while travelling by train at station and on board”

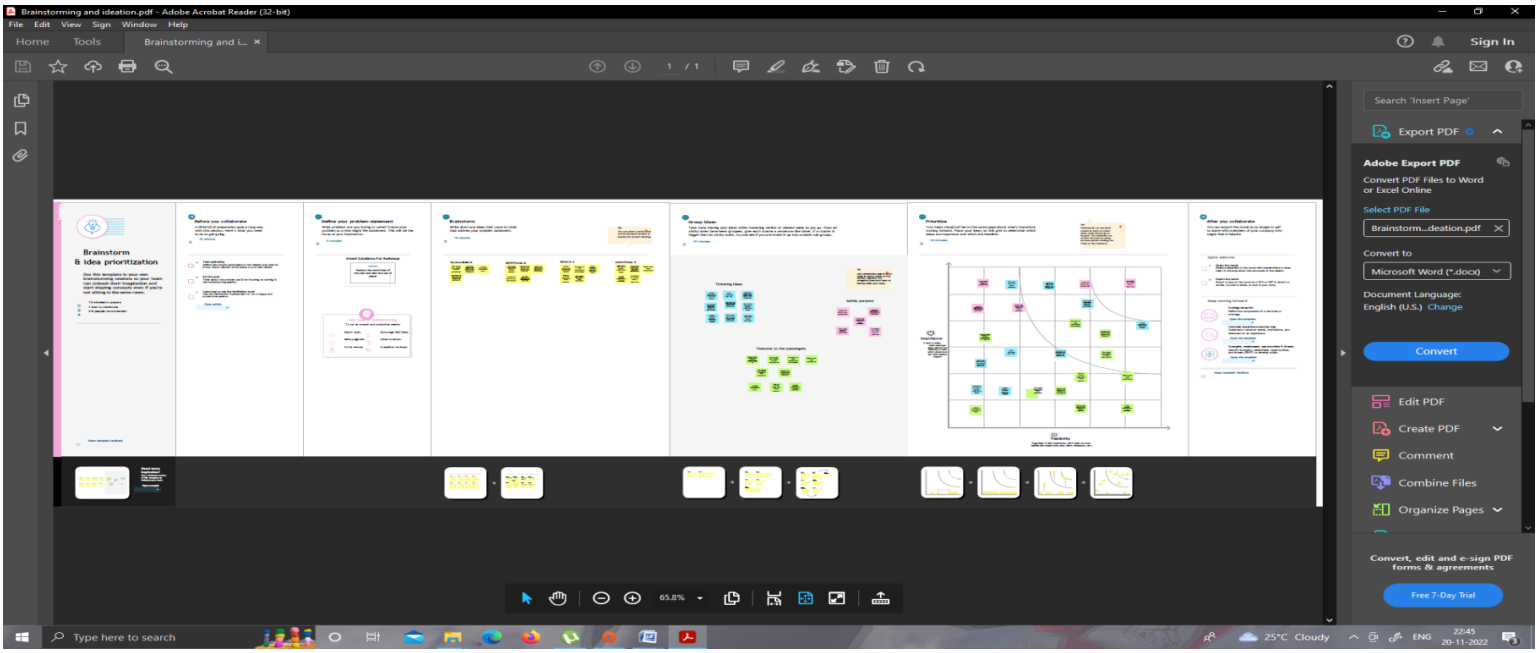
IDEATION AND PROPOSED SOLUTION

3. IDEATION AND PROPOSED SOLUTON

EMPATHY MAP CANVAS



IDEATION & BRAINSTORMING



PROPOSED SOLUTION

S.NO	PARAMETERS	DESCRIPTIONS
1.	Problem Statement (Problem to be solved)	The travel without tickets. Through various social media platforms. Railways are in regular touch with passengers to enhance security and to address their security concentration.
2.	Idea / Solution description	Passenger rail activity increases in the high rail scenario to 15 trillion passengers kilometres in 2050.
3.	Novelty / Uniqueness	QR code tickets. Location tracked.
4.	Social Impact / Customer Satisfaction	Railways Stretches its hands in conducting activities like business, sightseeing, pilgrimage along with transportation of goods.

5.	Business Model (Revenue Model)	Railways Promote economic growth while cutting greenhouse gas emissions.
6.	Scalability of the Solution	Smart sensors can be used to track important assets, manage passengers flow, and enable predictive maintenance.

PROBLEM SOLUTION FIT

REQUIREMENT ANALYSIS

4. REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENTS

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Unique accounts	Every online booking needs to be associated with an account
FR-2	Booking options	Search results should enable users to find the most recent and relevant booking options
FR-3	Mandatory fields	System should only allow users to move to payment only when mandatory fields such as date, time, location has been mentioned
FR-4	Synchronization	System should consider time zone synchronisation when accepting bookings from different time zones
FR-5	Authentication	Booking confirmation should be sent to user to the specified contact details

NON-FUNCTIONAL REQUIREMENTS

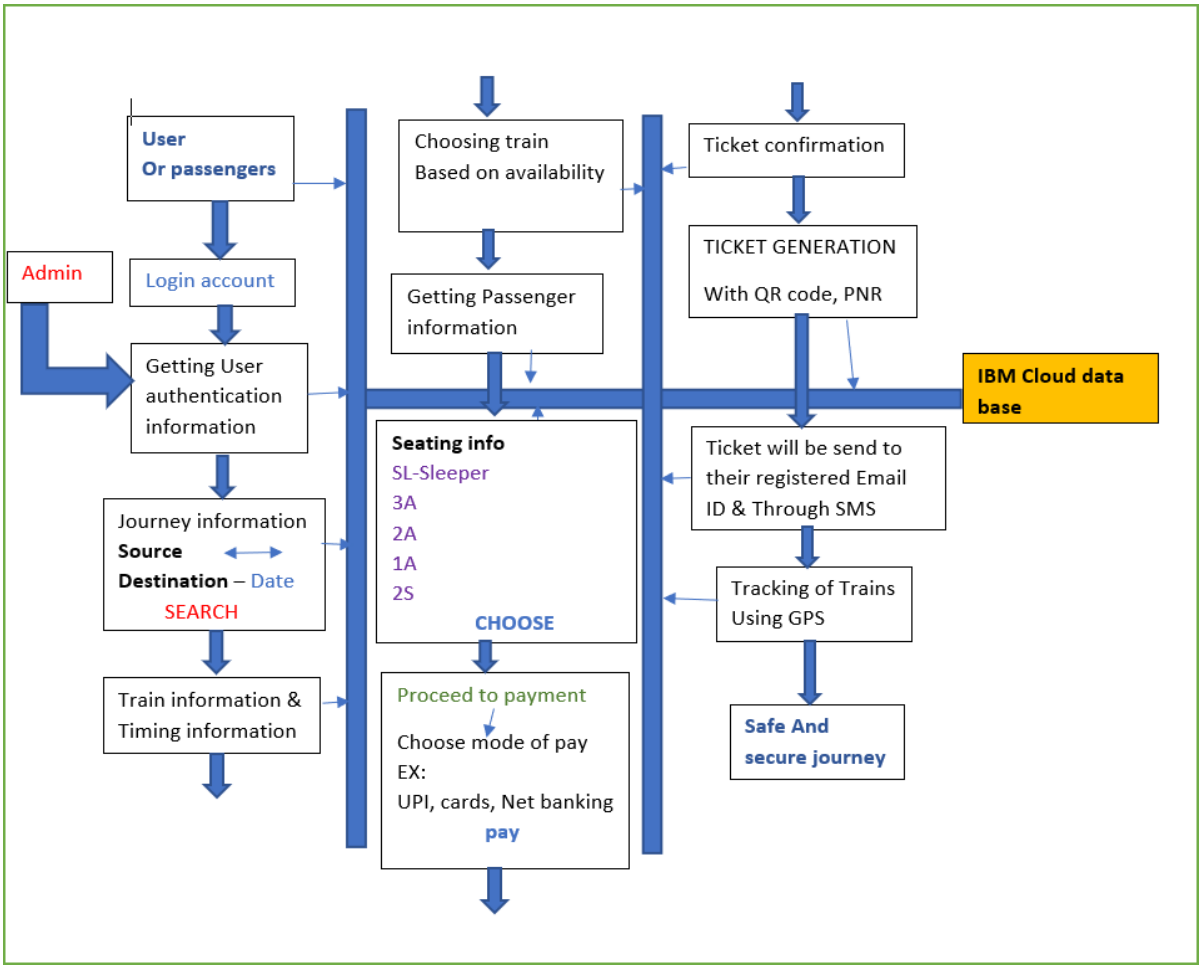
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Search results should populate within acceptable time limits
NFR-2	**Security**	System should visually confirm as well as send booking confirmation to the user's contact
NFR-3	**Reliability**	System should accept payments via different payment methods, like PayPal, wallets, cards, vouchers, etc
NFR-4	**Performance**	Search results should populate within acceptable time limits
NFR-5	**Availability**	User should be helped appropriately to fill in the mandatory fields, in case of invalid input

PROJECT DESIGN

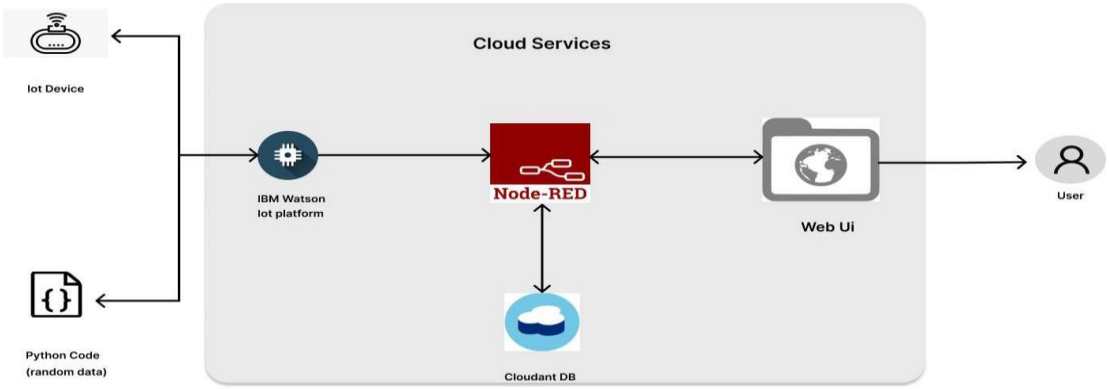
5. PROJECT DESIGN

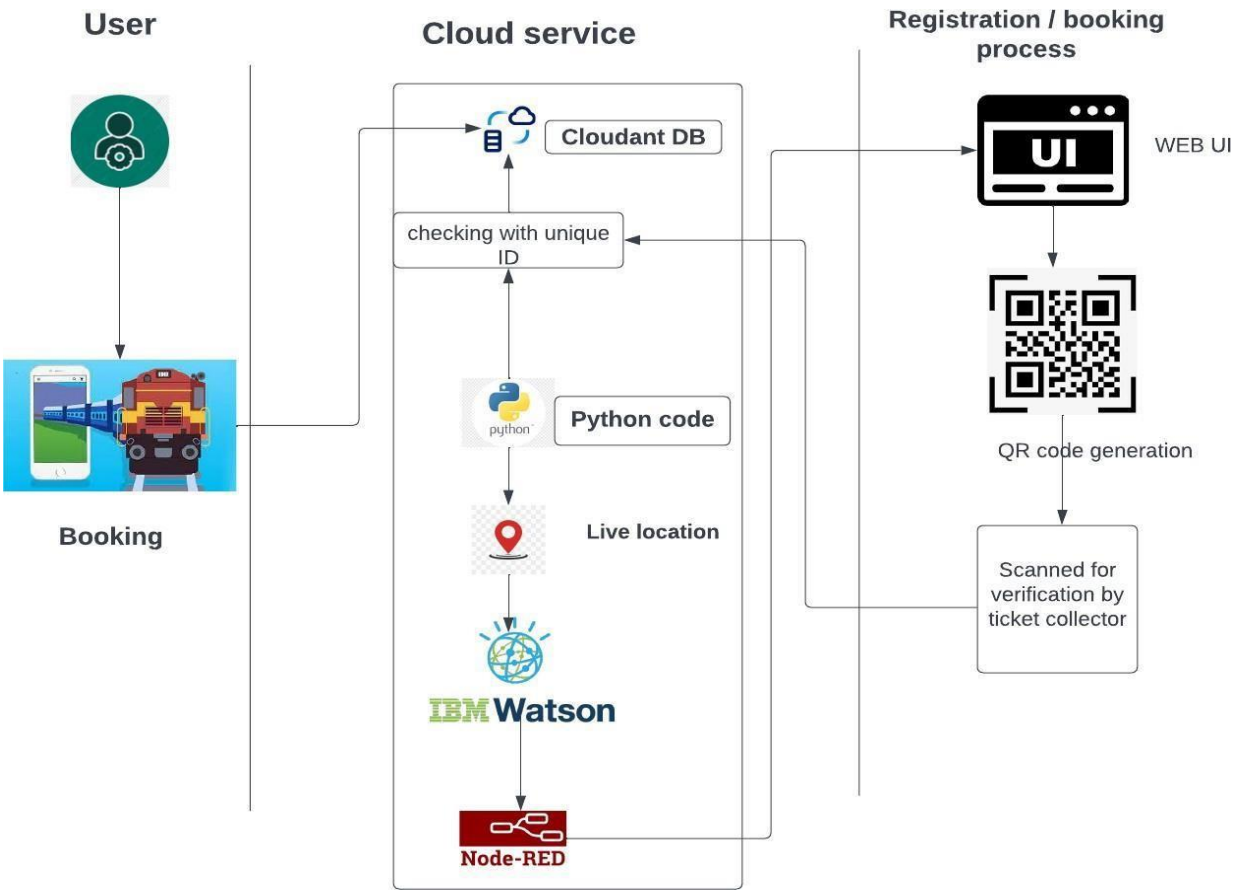
DATA FLOW DIAGRAMS

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.



SOLUTION & TECHNICAL ARCHITECTURE





USER STORIES

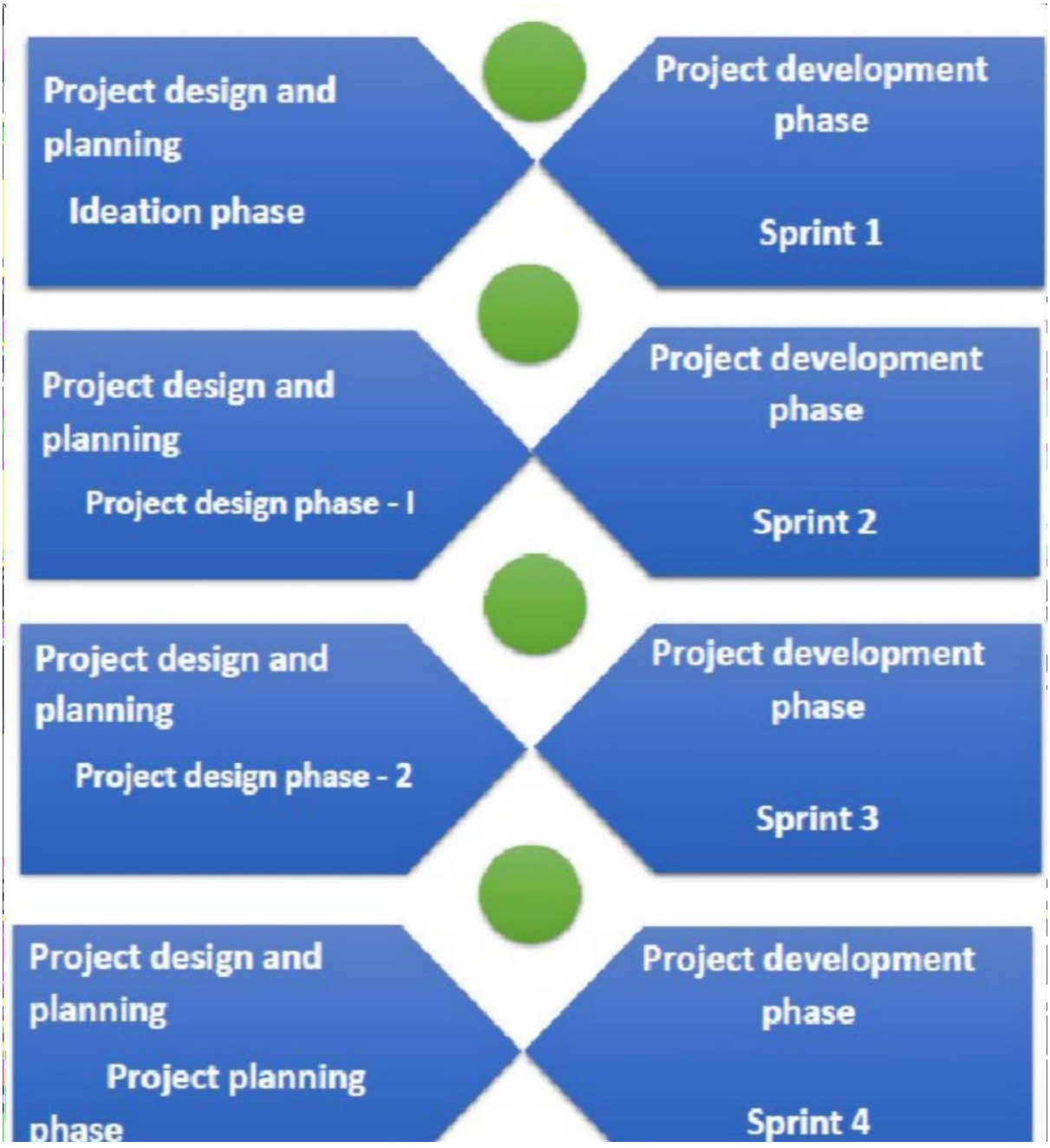
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Passenger	Registration	USN-1	As a passenger, I want to create a login credentials so I can securely access myself service online account.	Input data fields to enter: 1.Username/email 2.Password 3.Re-enter password 4.Security question 5.Security answer	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for creating an account.	I can receive confirmation email & click confirm.	High	Sprint-1
		USN-3	As a user, I can also create an account using Google.	I can register & access my account by using Google Login details.	High	Sprint-2
		USN-4	As a user, I can also create an account using Facebook.	I can register & access my account by using Facebook login details.	Medium	Sprint-3
	Login	USN-5	As a user, I can login to the account by entering my email and password. As a user, I can login to the account through Facebook if I previously registered with it. As a user, I can reset my password if I have forgotten my password.	I can login to the system so that my information can only be accessed by me.	High	Sprint-1
	My Account	USN-6	As a user, I can view my personal account. As a user, I can edit my Profile. .	I can use my personal account for booking process.	High	Sprint-1
Customer Care Executive		CCE-1	As a customer care executive ,I can take complaints ,answer calls from the customers regarding all the queries.	Pays attention to customer satisfaction to understand what services need improvements. Customer care executive should be able to assist	High	

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
				the users by easily communicating with them.		
Administrator		ADMIN-1	As an administrator I receive an email notification when a new user is registered.	The admin has the control over the new user by receiving a notification. .	High	
		ADMIN-2	As an administrator I am able to add a new person to the database and backup can also be done.	The admin has the ability to access the database.	Medium	
		ADMIN-3	As an administrator I am able to view content that to be viewed.	The details of the user should be given to the administrator impeccably when they request it.	Medium	

PROJECT PLANNING AND SCHEDULING

6. PROJECT PLANNING AND SCHEDULING

SPRINT PLANNING & ESTIMATION



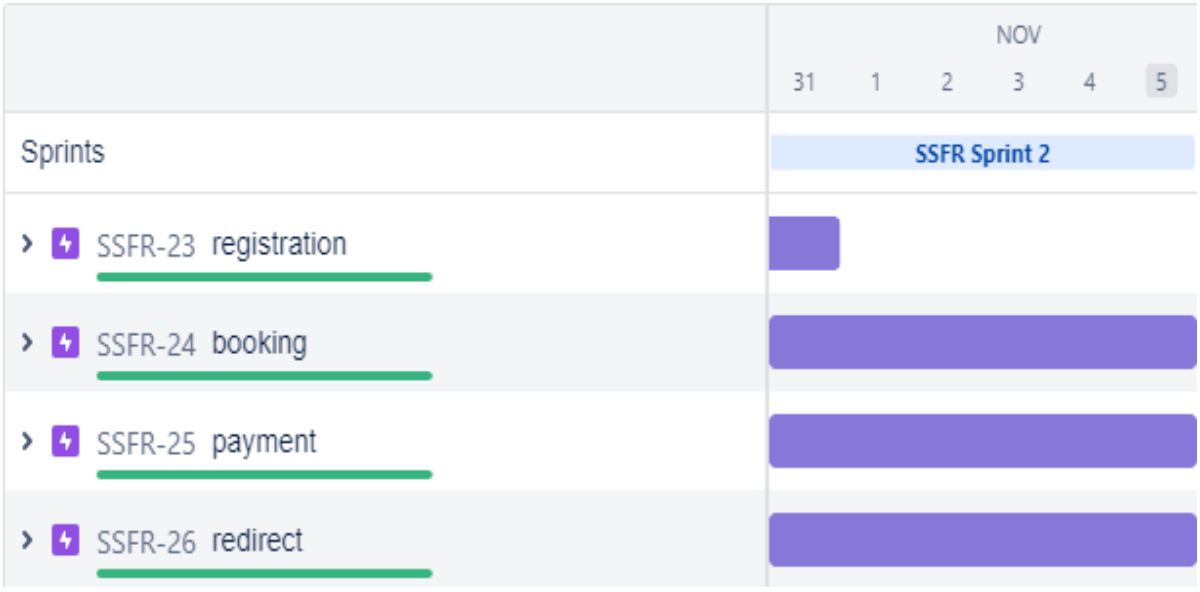
SPRINT PLAN

1. Identify the Problem
2. Prepare a Abstract ,Problem Statement
3. List a Require Needed
4. Create a Code and Run it
5. Make a Prototype
6. Test With The Created Code and check the designed prototype
7. Solution for the Problem is Found !!!

SPRINT DELIVERY SCHEDULE

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
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	5 Nov 2022
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov2022

REPORTS FROM JIRA



CODING AND SOLUTIONING

7. CODING AND SOLUTIONING

FEATURE 1

- IOT device
- IBM Watson platform
- Node red
- Cloudant DB
- Web UI
- Geofence
- MIT App
- Python code

FEATURE 2

- Registration
- Login
- Verification
- Ticket Booking
- Payment
- Ticket Cancellation
- Adding Queries

```
from tkinter import*
base = Tk()
base.geometry("500x500")
base.title("registration form")

labl_0 = Label(base, text="Registration form",width=20,font=("bold", 20))
labl_0.place(x=90,y=53)

lb1= Label(base, text="Enter Name", width=10, font=("arial",12))
lb1.place(x=20, y=120)
en1= Entry(base)
en1.place(x=200, y=120)

lb3= Label(base, text="Enter Email", width=10, font=("arial",12))
lb3.place(x=19, y=160)
en3= Entry(base)
en3.place(x=200, y=160)

lb4= Label(base, text="Contact Number", width=13,font=("arial",12))
lb4.place(x=19, y=200)
en4= Entry(base)
en4.place(x=200, y=200)

lb5= Label(base, text="Select Gender", width=15, font=("arial",12))
lb5.place(x=5, y=240)
var = IntVar()
Radiobutton(base, text="Male", padx=5,variable=var, value=1).place(x=180, y=240)
Radiobutton(base, text="Female", padx =10,variable=var,
value=2).place(x=240,y=240)
Radiobutton(base, text="others", padx=15, variable=var,
value=3).place(x=310,y=240)

list_of_cntry = ("United States", "India", "Nepal", "Germany")
cv = StringVar()
drplist= OptionMenu(base, cv, *list_of_cntry)
drplist.config(width=15)
cv.set("United States")
lb2= Label(base, text="Select Country", width=13,font=("arial",12))
lb2.place(x=14,y=280)
drplist.place(x=200, y=275)

lb6= Label(base, text="Enter Password", width=13,font=("arial",12))
lb6.place(x=19, y=320)
```

```
en6= Entry(base, show='*')
en6.place(x=200, y=320)

lb7= Label(base, text="Re-Enter Password", width=15,font=("arial",12))
lb7.place(x=21, y=360)
en7 =Entry(base, show='*')
en7.place(x=200, y=360)

Button(base, text="Register", width=10).place(x=200,y=400)
base.mainloop()
```

def generateOTP() :

```
# import library
import math, random

# function to generate OTP
def generateOTP() :

    # Declare a digits variable
    # which stores all digits
    digits = "0123456789"
    OTP = ""

   # length of password can be changed
   # by changing value in range
    for i in range(4) :
        OTP += digits[math.floor(random.random() * 10)]

    return OTP

# Driver code
if _name_ == "_main_" :

    print("OTP of 4 digits:", generateOTP())
```

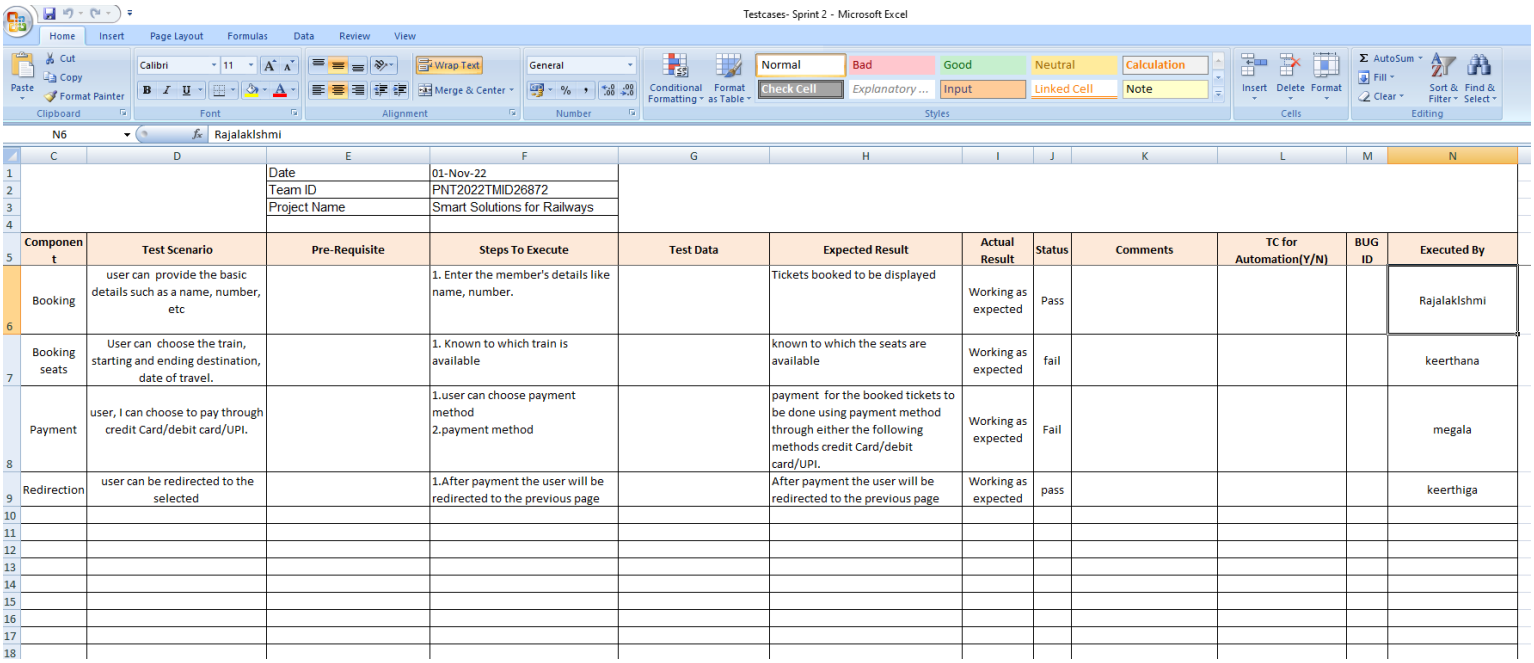
def ticket booking()

```
print("\n\nTicket Booking System\n")
restart = ('Y')

while restart != ('N','NO','n','no'):
        print("1.Check PNR status")
        print("2.Ticket Reservation")
        option = int(input("\nEnter your option : "))

        if option == 1:
                print("Your PNR status is t3")
                exit(0)

        elif option == 2:
                people = int(input("\nEnter no. of Ticket you want : "))
                name_l = []
                age_l = []
                sex_l = []
                for p in range(people):
                        name = str(input("\nName : "))
                        name_l.append(name)
                        age  = int(input("\nAge  : "))
                        age_l.append(age)
                        sex  = str(input("\nMale or Female : "))
                        sex_l.append(sex)

                restart = str(input("\nDid you forgot someone? y/n: "))
                if restart in ('y','YES','yes','Yes'):
                        restart = ('Y')
                else :
                        x = 0
                        print("\nTotal Ticket : ",people)
                        for p in range(1,people+1):
                                print("Ticket : ",p)
                                print("Name : ", name_l[x])
                                print("Age  : ", age_l[x])
                                print("Sex : ",sex_l[x])
                                x += 1
```

TESTING

8. TESTING

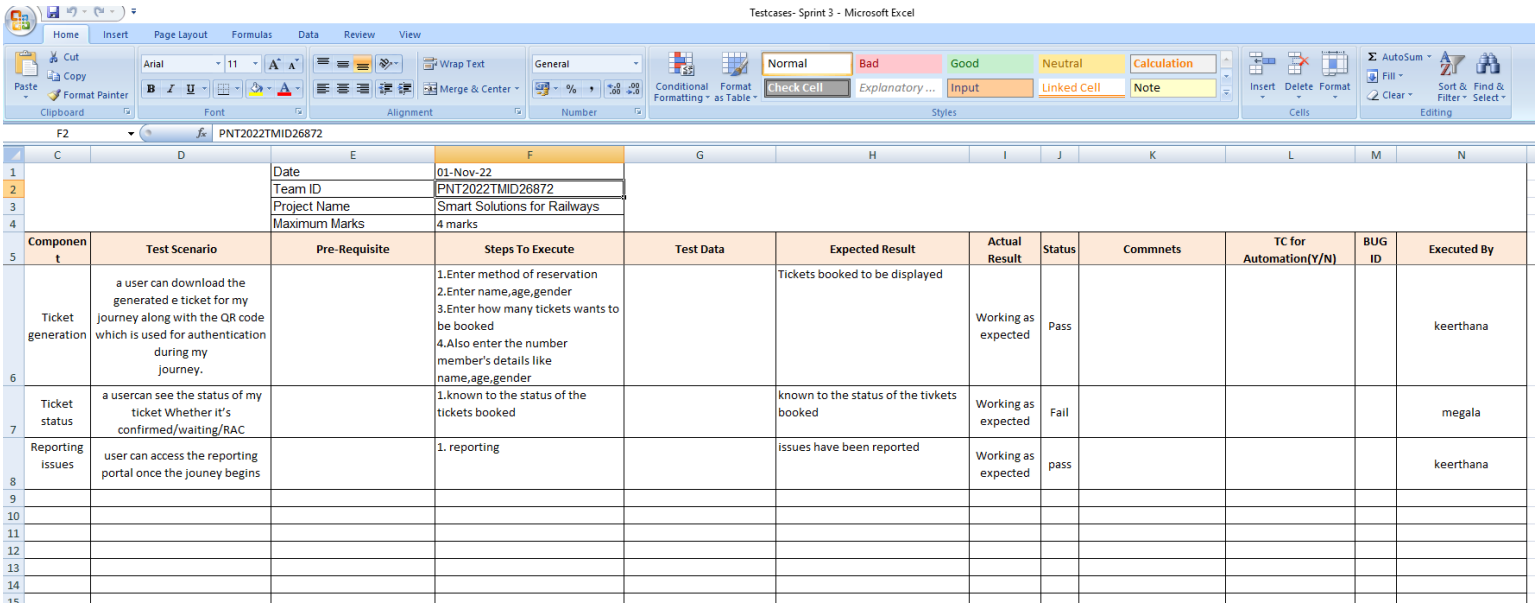
8.1.TEST CASES



Testcases- Sprint 2 - Microsoft Excel

Date	01-Nov-22
Team ID	PNT2022TMID26872
Project Name	Smart Solutions for Railways

Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
Booking	user can provide the basic details such as a name, number, etc		1. Enter the member's details like name, number.		Tickets booked to be displayed	Working as expected	Pass				Rajalakshmi
Booking seats	User can choose the train, starting and ending destination, date of travel.		1. Known to which train is available		known to which the seats are available	Working as expected	fail				keerthana
Payment	user, I can choose to pay through credit Card/debit card/UPI.		1.user can choose payment method 2.payment method		payment for the booked tickets to be done using payment method through either the following methods credit Card/debit card/UPI.	Working as expected	Fail				megala
Redirection	user can be redirected to the selected		1.After payment the user will be redirected to the previous page		After payment the user will be redirected to the previous page	Working as expected	pass				keerthiga



Testcases- Sprint 3 - Microsoft Excel

Date	01-Nov-22
Team ID	PNT2022TMID26872
Project Name	Smart Solutions for Railways
Maximum Marks	4 marks

Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
Ticket generation	a user can download the generated e ticket for my journey along with the QR code which is used for authentication during my journey.		1.Enter method of reservation 2.Enter name,age,gender 3.Enter how many tickets wants to be booked 4.Also enter the number member's details like name,age,gender		Tickets booked to be displayed	Working as expected	Pass				keerthana
Ticket status	a usercan see the status of my ticket Whether it's confirmed/waiting/RAC		1.known to the status of the tickets booked		known to the status of the tivkets booked	Working as expected	Fail				megala
Reporting issues	user can access the reporting portal once the jouney begins		1. reporting		issues have been reported	Working as expected	pass				keerthana

F	G	H	I	J	K	L	M	N
01-Nov-22								
PNT2022TMID26872								
Smart Solutions for Railways								
4 marks								
Steps To Execute	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Commnets**	**TC for Automation(Y/N)**	**BUG ID**	**Executed By**
1.tickets to be cancelled		Tickets booked to be cancelled	Working as expected	Fail				megala
1.information feeding on trains		information feeding on trains	Working as expected	pass				keerthiga

RESULTS

9. RESULTS

9.1.PERFORMANCE METRICS

ADVANTAGES &DISADVANTAGES

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

- Openness – compatibility between different system modules, potentially from different vendors;
- Orchestration – ability to manage large numbers of devices, with full visibility over them;
- Dynamic scaling – ability to scale the system according to the application needs, through resource virtualization and cloud operation;
- Automation – ability to automate parts of the system monitoring application, leading to better performance and lower operation costs.

DISADVANTAGES

- Approaches to flexible, effective, efficient, and low-cost data collection for both railway vehicles and infrastructure monitoring, using regular trains;
- Data processing, reduction, and analysis in local controllers, and subsequent sending of that data to the cloud, for further processing;
- Online data processing systems, for real-time monitoring, using emerging communication technologies;
- Integrated, interoperable, and scalable solutions for railway systems preventive maintenance.

CONCLUSION

11. CONCLUSION

Accidents occurring in Railway transportation system cost a large number of lives. So this system helps us to prevent accidents and giving information about faults or cracks in advance to railway authorities. So that they can fix them and accidents cases becomes less. This project is cost effective. By using more techniques they can be modified and developed according to their applications. By this system many lives can be saved by avoiding accidents. The idea can be implemented in large scale in the long run to facilitate better safety standards for rail tracks and provide effective testing infrastructure for achieving better results in the future.

FUTURE SCOPE

12. FUTURE SCOPE

In future CCTV systems with IP based camera can be used for monitoring the visual videos captured from the track. It will also increase security for both passengers and railways. GPS can also be used to detect exact location of track fault area, IP cameras can also be used to show fault with the help of video. Locations on Google maps with the help of sensors can be used to detect in which area track is broken.

APPENDIX

13. APPENDIX

SOURCE PROGRAM

```
import wiotp.sdk.device
import time
import random
myConfig = {
   "identity": {
      "orgId": "xfxj98",
      "typeId": "railway23",
      "deviceId":"Device1"
   },
   "auth": {
      "token": "987456321"
   }
}

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,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 ()
```

```
from tkinter import*
base = Tk()
base.geometry("500x500")
base.title("registration form")

labl_0 = Label(base, text="Registration form",width=20,font=("bold", 20))
labl_0.place(x=90,y=53)

lb1= Label(base, text="Enter Name", width=10, font=("arial",12))
lb1.place(x=20, y=120)
en1= Entry(base)
en1.place(x=200, y=120)

lb3= Label(base, text="Enter Email", width=10, font=("arial",12))
lb3.place(x=19, y=160)
en3= Entry(base)
en3.place(x=200, y=160)

lb4= Label(base, text="Contact Number", width=13,font=("arial",12))
lb4.place(x=19, y=200)
en4= Entry(base)
en4.place(x=200, y=200)

lb5= Label(base, text="Select Gender", width=15, font=("arial",12))
lb5.place(x=5, y=240)
var = IntVar()
Radiobutton(base, text="Male", padx=5,variable=var, value=1).place(x=180, y=240)
Radiobutton(base, text="Female", padx =10,variable=var,
value=2).place(x=240,y=240)
```

```
Radiobutton(base, text="others", padx=15, variable=var,
value=3).place(x=310,y=240)

list_of_cntry = ("United States", "India", "Nepal", "Germany")
cv = StringVar()
drplist= OptionMenu(base, cv, *list_of_cntry)
drplist.config(width=15)
cv.set("United States")
lb2= Label(base, text="Select Country", width=13,font=("arial",12))
lb2.place(x=14,y=280)
drplist.place(x=200, y=275)

lb6= Label(base, text="Enter Password", width=13,font=("arial",12))
lb6.place(x=19, y=320)
en6= Entry(base, show='*')
en6.place(x=200, y=320)

lb7= Label(base, text="Re-Enter Password", width=15,font=("arial",12))
lb7.place(x=21, y=360)
en7 =Entry(base, show='*')
en7.place(x=200, y=360)

Button(base, text="Register", width=10).place(x=200,y=400)
base.mainloop()
```

```
def generateOTP() :
```

```
# import library
import math, random

# function to generate OTP
def generateOTP() :

    # Declare a digits variable
    # which stores all digits
    digits = "0123456789"
    OTP = ""

   # length of password can be changed
   # by changing value in range
    for i in range(4) :
        OTP += digits[math.floor(random.random() * 10)]

    return OTP

# Driver code
if _name_ == "_main_" :

    print("OTP of 4 digits:", generateOTP())
```

def ticket booking()

```
print("\n\nTicket Booking System\n")
restart = ('Y')

while restart != ('N','NO','n','no'):
          print("1.Check PNR status")
```

```
print("2.Ticket Reservation")
option = int(input("\nEnter your option : "))

if option == 1:
        print("Your PNR status is t3")
        exit(0)

elif option == 2:
        people = int(input("\nEnter no. of Ticket you want : "))
        name_l = []
        age_l = []
        sex_l = []
        for p in range(people):
                name = str(input("\nName : "))
                name_l.append(name)
                age  = int(input("\nAge  : "))
                age_l.append(age)
                sex  = str(input("\nMale or Female : "))
                sex_l.append(sex)

        restart = str(input("\nDid you forgot someone? y/n: "))
        if restart in ('y','YES','yes','Yes'):
                restart = ('Y')
        else :
                x = 0
                print("\nTotal Ticket : ",people)
                for p in range(1,people+1):
                        print("Ticket : ",p)
                        print("Name : ", name_l[x])
```

```
                print("Age  : ", age_l[x])
                print("Sex : ",sex_l[x])
                x += 1
def scanner():
from http import client
import cv2
import pyzbar
from pyzbar.pyzbar import decode
import time

from ibmcloudant.cloudant_v1 import CloudantV1
from ibmcloudant import CouchDbSessionAuthenticator
from ibm_cloud_sdk_core.authenticators import BasicAuthenticator

authenticator = BasicAuthenticator('apikey-v2-
1oj043bu90m78ng4h2j27w5nob2nvcma6xanc6bk0a7m',
'daf3c00c2cc182af425a5691a07f7b93')
service = CloudantV1(authenticator=authenticator)

service.set_service_url('https://apikey-v2-
1oj043bu90m78ng4h2j27w5nob2nvcma6xanc6bk0a7m:daf3c00c2cc182af425a5691a
07f7b93@932393aa-9f82-4144-9251-2c519fb30962-
bluemix.cloudantnosqldb.appdomain.cloud')

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

while True:
    _, frame = cap.read()
```

```
        decodedObjects = 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
                ).get_result()
                print (response)
                time.sleep(5)
            except Exception as e:
                print(a)
                print ("Not a Valid Ticket")
                time.sleep(5)

        cv2.imshow("Frame",frame)
        if cv2.waitKey(1) & 0xFF ==ord('q'):
            break
cap.release()
cv2.destroyAllWindows()
client.disconnect()

def gps location():
import time
import sys
```

```
import ibmiotf.application
import ibmiotf.device
import random
import requests
import json

#Provide your IBM Watson Device Credentials
organization = "aynel8"
deviceType = "iot_device"        #Credentials of Watson IoT sensor simulator
deviceId = "4016"
authMethod = "token"
authToken = "12345678"



# Initialize the device client.
L=0

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()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting" 10 times
```

```
deviceCli.connect()

while True:
    overpass_url = "http://overpass-api.de/api/interpreter"
    overpass_query = """
    [out:json];area[name="India"];(node[place="village"](area););out;
    """

    response = requests.get(
    overpass_url,
    params={'data': overpass_query}
    )

    coords = []
    if response.status_code == 200:
        data = response.json()
        places = data.get('elements', [])
        for place in places:
            coords.append((place['lat'], place['lon']))
        print ("Got %s village coordinates!" % len(coords))
        print (coords[0])
    else:
        print("Error")

    i = random.randint(1,100)
    L = coords[i]
    #Send random gprs data to node-red to IBM Watson
    data = {"d":{ 'Latitude' : L[0], 'Longitude' : L[1]}}
    #print data
```

```
    def myOnPublishCallback():
        print("Published gprs location = ", L, "to IBM Watson")

    success = deviceCli.publishEvent("Data", "json", data, qos=0,
on_publish=myOnPublishCallback)
    time.sleep(12)
    if not success:
        print("Not connected to IoTF")
    time.sleep(1)

deviceCli.disconnect()

def origin and destination():

# import module
import requests
from bs4 import BeautifulSoup

# user define function
# Scrape the data
def getdata(url):
    r = requests.get(url)
    return r.text


# input by geek
from_Station_code = "NLR"
from_Station_name = "NELLORE"
```

```
To_station_code = "OGL"
To_station_name = "ONGOLE"
# url
url = "https://www.railyatri.in/booking/trains-between-stations?from_code="+from_Station_code+"&from_name="+from_Station_name+"+JN+&journey_date=+Wed&src=tbs&to_code=" + \
   To_station_code+"&to_name="+To_station_name + \
   "+JN+&user_id=-1603228437&user_token=355740&utm_source=dwebsearch_tbs_search_trains"

# pass the url
# into getdata function
htmldata = getdata(url)
soup = BeautifulSoup(htmldata, 'html.parser')

# find the Html tag
# with find()
# and convert into string
data_str = ""
for item in soup.find_all("div", class_="col-xs-12 TrainSearchSection"):
   data_str = data_str + item.get_text()
result = data_str.split("\n")

print("Train between "+from_Station_name+" and "+To_station_name)
print("")

# Display the result
for item in result:
   if item != "":
```

```
        print(item)


def OTP verification():

# import library
import math, random

# function to generate OTP
def generateOTP() :

    # Declare a digits variable
    # which stores all digits
    digits = "0123456789"
    OTP = ""

   # length of password can be changed
   # by changing value in range
    for i in range(4) :
        OTP += digits[math.floor(random.random() * 10)]

    return OTP

# Driver code
if__name__== "__main__" :

    print("OTP of 4 digits:", generateOTP())
```

```
Def login():

from tkinter import *
import sqlite3

root = Tk()
root.title("Python: Simple Login Application")
width = 400
height = 280
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
x = (screen_width/2) - (width/2)
y = (screen_height/2) - (height/2)
root.geometry("%dx%d+%d+%d" % (width, height, x, y))
root.resizable(0, 0)

#==============================VARIABLES=====================
===============
USERNAME = StringVar()
PASSWORD = StringVar()

#==============================FRAMES=======================
================
Top = Frame(root, bd=2,  relief=RIDGE)
Top.pack(side=TOP, fill=X)
Form = Frame(root, height=200)
Form.pack(side=TOP, pady=20)
```

```
#==============================LABELS=========================
===============
lbl_title = Label(Top, text = "Python: Simple Login Application", font=('arial', 15))
lbl_title.pack(fill=X)
lbl_username = Label(Form, text = "Username:", font=('arial', 14), bd=15)
lbl_username.grid(row=0, sticky="e")
lbl_password = Label(Form, text = "Password:", font=('arial', 14), bd=15)
lbl_password.grid(row=1, sticky="e")
lbl_text = Label(Form)
lbl_text.grid(row=2, columnspan=2)

#==============================ENTRY
WIDGETS=================================
username = Entry(Form, textvariable=USERNAME, font=(14))
username.grid(row=0, column=1)
password = Entry(Form, textvariable=PASSWORD, show="*", font=(14))
password.grid(row=1, column=1)




#==============================METHODS=======================
================
def Database():
    global conn, cursor
    conn = sqlite3.connect("pythontut.db")
    cursor = conn.cursor()
```

```
    cursor.execute("CREATE TABLE IF NOT EXISTS `member` (mem_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username TEXT, password TEXT)")
    cursor.execute("SELECT * FROM `member` WHERE `username` = 'admin' AND `password` = 'admin'")
    if cursor.fetchone() is None:
        cursor.execute("INSERT INTO `member` (username, password) VALUES('admin', 'admin')")
        conn.commit()
def Login(event=None):
    Database()
    if USERNAME.get() == "" or PASSWORD.get() == "":
        lbl_text.config(text="Please complete the required field!", fg="red")
    else:
        cursor.execute("SELECT * FROM `member` WHERE `username` = ? AND `password` = ?", (USERNAME.get(), PASSWORD.get()))
        if cursor.fetchone() is not None:
            HomeWindow()
            USERNAME.set("")
            PASSWORD.set("")
            lbl_text.config(text="")
        else:
            lbl_text.config(text="Invalid username or password", fg="red")
            USERNAME.set("")
            PASSWORD.set("")
    cursor.close()
    conn.close()
```

```
#==============================BUTTON
WIDGETS================================
btn_login = Button(Form, text="Login", width=45, command=Login)
btn_login.grid(pady=25, row=3, columnspan=2)
btn_login.bind('<Return>', Login)



def HomeWindow():
   global Home
   root.withdraw()
   Home = Toplevel()
   Home.title("Python: Simple Login Application")
   width = 600
   height = 500
   screen_width = root.winfo_screenwidth()
   screen_height = root.winfo_screenheight()
   x = (screen_width/2) - (width/2)
   y = (screen_height/2) - (height/2)
   root.resizable(0, 0)
   Home.geometry("%dx%d+%d+%d" % (width, height, x, y))
   lbl_home = Label(Home, text="Successfully Login!", font=('times new roman',
20)).pack()
   btn_back = Button(Home, text='Back', command=Back).pack(pady=20, fill=X)

def Back():
   Home.destroy()
   root.deiconify()
```

```
Def seat booking():

def berth_type(s):

   if s>0 and s<73:
      if s % 8 == 1 or s % 8 == 4:
         print (s), "is lower berth"
      elif s % 8 == 2 or s % 8 == 5:
         print (s), "is middle berth"
      elif s % 8 == 3 or s % 8 == 6:
         print (s), "is upper berth"
      elif s % 8 == 7:
         print (s), "is side lower berth"
      else:
         print (s), "is side upper berth"
   else:
      print (s), "invalid seat number"

# Driver code
s = 10
berth_type(s)      # fxn call for berth type

s = 7
berth_type(s)     # fxn call for berth type

s = 0
berth_type(s)      # fxn call for berth type
```

```
Def ticket booking():

print("\n\nTicket Booking System\n")
restart = ('Y')

while restart != ('N','NO','n','no'):
        print("1.Check PNR status")
        print("2.Ticket Reservation")
        option = int(input("\nEnter your option : "))

        if option == 1:
                print("Your PNR status is t3")
                exit(0)

        elif option == 2:
                people = int(input("\nEnter no. of Ticket you want : "))
                name_l = []
                age_l = []
                sex_l = []
                for p in range(people):
                        name = str(input("\nName : "))
                        name_l.append(name)
                        age  = int(input("\nAge  : "))
                        age_l.append(age)
                        sex  = str(input("\nMale or Female : "))
                        sex_l.append(sex)

                restart = str(input("\nDid you forgot someone? y/n: "))
                if restart in ('y','YES','yes','Yes'):
```

```
                restart = ('Y')
        else :
                x = 0
                print("\nTotal Ticket : ",people)
                for p in range(1,people+1):
                        print("Ticket : ",p)
                        print("Name : ", name_l[x])
                        print("Age  : ", age_l[x])
                        print("Sex : ",sex_l[x])
                        x += 1


def payment():

from django.contrib.auth.base_user import AbstractBaseUser
from django.db import models


class User(AbstractBaseUser):
    """
    User model.
    """

    USERNAME_FIELD = "email"

    REQUIRED_FIELDS = ["first_name", "last_name"]

    email = models.EmailField(
        verbose_name="E-mail",
```

```
    unique=True
)

first_name = models.CharField(
    verbose_name="First name",
    max_length=30
)

last_name = models.CharField(
    verbose_name="Last name",
    max_length=40
)

city = models.CharField(
    verbose_name="City",
    max_length=40
)

stripe_id = models.CharField(
    verbose_name="Stripe ID",
    unique=True,
    max_length=50,
    blank=True,
    null=True
)

objects = UserManager()

@property
```

```
    def get_full_name(self):
        return f"{self.first_name} {self.last_name}"

    class Meta:
        verbose_name = "User"
        verbose_name_plural = "Users"


class Profile(models.Model):
    """
    User's profile.
    """

    phone_number = models.CharField(
        verbose_name="Phone  number",
        max_length=15
    )

    date_of_birth = models.DateField(
        verbose_name="Date of birth"
    )

    postal_code = models.CharField(
        verbose_name="Postal  code",
        max_length=10,
        blank=True
    )

    address = models.CharField(
```

```
        verbose_name="Address",
        max_length=255,
        blank=True
    )

    class Meta:
        abstract = True


class UserProfile(Profile):
    """
    User's profile model.
    """

    user = models.OneToOneField(
        to=User, on_delete=models.CASCADE, related_name="profile",
    )

    group = models.CharField(
        verbose_name="Group type",
        choices=GroupTypeChoices.choices(),
        max_length=20,
        default=GroupTypeChoices.EMPLOYEE.name,
    )

    def __str__(self):
        return self.user.email

    class Meta:
```

```
# user 1 - employer
user1, _ = User.objects.get_or_create(
    email="foo@bar.com",
    first_name="Employer",
    last_name="Testowy",
    city="Białystok",
)

user1.set_unusable_password()

group_name = "employer"

_profile1, _ = UserProfile.objects.get_or_create(
    user=user1,
    date_of_birth=datetime.now() - timedelta(days=6600),
    group=GroupTypeChoices(group_name).name,
    address="Myśliwska 14",
    postal_code="15-569",
    phone_number="+48100200300",
)

# user2 - employee
user2, _ = User.objects.get_or_create()
    email="bar@foo.com",
    first_name="Employee",
    last_name="Testowy",
    city="Białystok",
)
```

```
user2.set_unusable_password()

group_name = "employee"

_profile2, _ = UserProfile.objects.get_or_create()
    user=user2,
    date_of_birth=datetime.now() - timedelta(days=7600),
    group=GroupTypeChoices(group_name).name,
    address="Myśliwska 14",
    postal_code="15-569",
    phone_number="+48200300400",
)

response_customer = stripe.Customer.create()
    email=user.email,
    description=f"EMPLOYER - {user.get_full_name}",
    name=user.get_full_name,
    phone=user.profile.phone_number,
)

user1.stripe_id = response_customer.stripe_id
user1.save()

mcc_code, url = "1520", "https://www.softserveinc.com/"

response_ca = stripe.Account.create()
    type="custom",
    country="PL",
```

```
    email=user2.email,
    default_currency="pln",
    business_type="individual",
    settings={"payouts": {"schedule": {"interval": "manual", }}},
    requested_capabilities=["card_payments", "transfers", ],
    business_profile={"mcc": mcc_code, "url": url},
    individual={
        "first_name": user2.first_name,
        "last_name": user2.last_name,
        "email": user2.email,
        "dob": {
            "day": user2.profile.date_of_birth.day,
            "month": user2.profile.date_of_birth.month,
            "year": user2.profile.date_of_birth.year,
        },
        "phone": user2.profile.phone_number,
        "address": {
            "city": user2.city,
            "postal_code": user2.profile.postal_code,
            "country": "PL",
            "line1": user2.profile.address,
        },
    },
)

user2.stripe_id = response_ca.stripe_id
user2.save()

tos_acceptance = {"date": int(time.time()), "ip": user_ip},
```

```
stripe.Account.modify(user2.stripe_id, tos_acceptance=tos_acceptance)

passport_front = stripe.File.create(
   purpose="identity_document",
   file=_file, # ContentFile object
   stripe_account=user2.stripe_id,
)

individual = {
   "verification": {
      "document": {"front": passport_front.get("id"),},
      "additional_document": {"front": passport_front.get("id"),},
   }
}


stripe.Account.modify(user2.stripe_id, individual=individual)

new_card_source = stripe.Customer.create_source(user1.stripe_id, source=token)

stripe.SetupIntent.create(
   payment_method_types=["card"],
   customer=user1.stripe_id,
   description="some description",
   payment_method=new_card_source.id,
)

payment_method = stripe.Customer.retrieve(user1.stripe_id).default_source
```

```
payment_intent = stripe.PaymentIntent.create(
    amount=amount,
    currency="pln",
    payment_method_types=["card"],
    capture_method="manual",
    customer=user1.stripe_id, # customer
    payment_method=payment_method,
    application_fee_amount=application_fee_amount,
    transfer_data={"destination": user2.stripe_id}, # connect account
    description=description,
    metadata=metadata,
)

payment_intent_confirm = stripe.PaymentIntent.confirm(
    payment_intent.stripe_id, payment_method=payment_method
)

stripe.PaymentIntent.capture(
    payment_intent.id, amount_to_capture=amount
)
stripe.Balance.retrieve(stripe_account=user2.stripe_id)

stripe.Charge.create(
    amount=amount,
    currency="pln",
    source=user2.stripe_id,
    description=description
)
```

```
stripe.PaymentIntent.cancel(payment_intent.id)



        unique_together = ("user", "group")

def redirect():
import logging
import attr
from flask import Blueprint, flash, redirect, request, url_for
from flask.views import MethodView
from flask_babelplus import gettext as _
from flask_login import current_user, login_required
from pluggy import HookimplMarker

@attr.s(frozen=True, cmp=False, hash=False, repr=True)
class UserSettings(MethodView):
    form = attr.ib(factory=settings_form_factory)
    settings_update_handler = attr.ib(factory=settings_update_handler)

    decorators = [login_required]

    def get(self):
        return self.render()

    def post(self):
        if self.form.validate_on_submit():
            try:
                self.settings_update_handler.apply_changeset(
```

```
                current_user, self.form.as_change()
            )
        except StopValidation as e:
            self.form.populate_errors(e.reasons)
            return self.render()
        except PersistenceError:
            logger.exception("Error while updating user settings")
            flash(_("Error while updating user settings"), "danger")
            return self.redirect()

        flash(_("Settings updated."), "success")
        return self.redirect()
    return self.render()

def render(self):
    return render_template("user/general_settings.html", form=self.form)

def redirect(self):
    return redirect(url_for("user.settings"))


@attr.s(frozen=True, hash=False, cmp=False, repr=True)
class ChangePassword(MethodView):
    form = attr.ib(factory=change_password_form_factory)
    password_update_handler = attr.ib(factory=password_update_handler)
    decorators = [login_required]

    def get(self):
        return self.render()
```

```
    def post(self):
        if self.form.validate_on_submit():
            try:
                self.password_update_handler.apply_changeset(
                    current_user, self.form.as_change()
                )
            except StopValidation as e:
                self.form.populate_errors(e.reasons)
                return self.render()
            except PersistenceError:
                logger.exception("Error while changing password")
                flash(_("Error while changing password"), "danger")
                return self.redirect()

            flash(_("Password updated."), "success")
            return self.redirect()
        return self.render()

    def render(self):
        return render_template("user/change_password.html", form=self.form)

    def redirect(self):
        return redirect(url_for("user.change_password"))


@attr.s(frozen=True, cmp=False, hash=False, repr=True)
class ChangeEmail(MethodView):
    form = attr.ib(factory=change_email_form_factory)
```

```
update_email_handler = attr.ib(factory=email_update_handler)
decorators = [login_required]

def get(self):
    return self.render()

def post(self):
    if self.form.validate_on_submit():
        try:
            self.update_email_handler.apply_changeset(
                current_user, self.form.as_change()
            )
        except StopValidation as e:
            self.form.populate_errors(e.reasons)
            return self.render()
        except PersistenceError:
            logger.exception("Error while updating email")
            flash(_("Error while updating email"), "danger")
            return self.redirect()

        flash(_("Email address updated."), "success")
        return self.redirect()
    return self.render()

def render(self):
    return render_template("user/change_email.html", form=self.form)

def redirect(self):
    return redirect(url_for("user.change_email"))
```

```
Def notification():

import pyttsx3
from plyer import notification
import time


# Speak method
def Speak(self, audio):

        # Calling the initial constructor
        # of pyttsx3
        engine = pyttsx3.init('sapi5')

        # Calling the getter method
        voices = engine.getProperty('voices')

        # Calling the setter method
        engine.setProperty('voice', voices[1].id)

        engine.say(audio)
        engine.runAndWait()


def Take_break():

        Speak("Do you want to start sir?")
        question = input()
```

```
if "yes" in question:
        Speak("Starting Sir")

if "no" in question:
        Speak("We will automatically start after 5 Mins Sir.")
        time.sleep(5*60)
        Speak("Starting Sir")

# A notification we will held that
# Let's Start sir and with a message of
# will tell you to take a break after 45
# mins for 10 seconds
while(True):
        notification.notify(title="Let's Start sir",
        message="will tell you to take a break after 45 mins",
        timeout=10)

        # For 45 min the will be no notification but
        # after 45 min a notification will pop up.
        time.sleep(0.5*60)

        Speak("Please Take a break Sir")

        notification.notify(title="Break Notification",
        message="Please do use your device after sometime as you have"
        "been continuously using it for 45 mins and it will affect your eyes",
        timeout=10)
```

```
# Driver's Code
if__name__== '_main_':
        Take_break()


def ticket gen():

class Ticket:
   counter=0
   def__init_(self,passenger_name,source,destination):
      self.__passenger_name=passenger_name
      self.__source=source
      self.__destination=destination
      self.Counter=Ticket.counter
      Ticket.counter+=1
   def validate_source_destination(self):
      if (self.__source=="Delhi" and (self.__destination=="Pune" or
self.__destination=="Mumbai" or self.__destination=="Chennai" or
self._destination=="Kolkata")):
         return True
      else:
         return False

   def generate_ticket(self ):
      if True:
         __ticket_id=self._source[0]+self._destination[0]+"0"+str(self.Counter)
         print( "Ticket id will be:",_ticket_id)
      else:
         return False
```

```
def get_ticket_id(self):
    return self.ticket_id
def get_passenger_name(self):
    return self._passenger_name
def get_source(self):
    if self._source=="Delhi":
        return self._source
    else:
        print("you have written invalid soure option")
        return None
def get_destination(self):
    if self._destination=="Pune":
        return self._destination
    elif self._destination=="Mumbai":
        return self._destination
    elif self._destination=="Chennai":
        return self._destination
    elif self._destination=="Kolkata":
        return self._destination

    else:
        return None
```