**Project Report**

# 1.INTRODUCTION

1. Project Overview
2. Purpose

## 2. LITERATURE SURVEY

1. Existing Systems
2. References
3. Problem Statement Definition

## 3.IDEATION & PROPOSED SOLUTION

1. Empathy Map Canvas
2. Ideation & Brainstorming
3. Problem Solution fit
4. Proposed Solution

## 4. REQUIREMENT ANALYSIS

1. Functional Requirement
2. Non-functional Requirement

## 5. PROJECT DESIGN

1. Data Flow Diagram
2. Solution & Technical Architecture
3. User Stories

## 6. PROJECT PLANNING AND SCHEDULING

1. Sprint planning & estimation
2. Sprint delivery schedule
3. Reports from Jira

## 7. CODING SOLUTIONING

[1. Feature 1](#_Toc47751)

[2. Feature3. Database Schema 8. **TESTING** 2](#_Toc47752)

1. Test cases
2. User acceptance testing
3. **RESULTS**

1. Performance Metrics

1. **ADVANTAGES AND DISADVANTAGES**
2. **CONCLUSION**
3. **FUTURE SCOPE**
4. **APPENDIX**

**INTRODUCTION**

**1. INTRODUCTION**

## 1.1 PROJECT OVERVIEW

**SMART SOLUTIONS FOR RAILWAYS** is to manage Indian railways which is the largest railway network in Asia and additionally it is the 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 move 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-tocomputer 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**

## 2.1 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 color 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**.**

## 2.2 REFERENCES

1. D. Hesse, “Rail Inspection Using Ultrasonic Surface Waves” Thesis, Imperial College of London, 2007.
2. Md. Reya Shad Azim1, Khizir Mahmud2 and C. K. Das. Automatic railway
3. track switching system, International Journal of Advanced Technology, Volume 54, 2014.
4. S. Somal Raju, V. Murali, G. saha and V. Vaidehi, “Title-robust railway crack detection scheme using LED (Light Emitting Diode) - LDR (Light Dependent Resistor) assembly IEEE 2012.
5. S. Srivastava, R. P. Chourasia, P. Sharma, S. I. Abbas, N. K. Singh,
6. “Railway Track Crack detection vehicle”, IARJSET, Vol. 4, pp. 145-148, Issued in 2, Feb 2017.
7. U. Mishra, V. Gupta, S. M. Ahzam and S. M. Tripathi, “Google Map Based
8. Railway Track Fault Detection Over the Internet”, International Journal of Applied Engineering Research, Vol. 14, pp. 20-23, Number 2, 2019.
9. R. A. Raza, K. P. Rauf, A. Shafeeq, “Crack detection in Railway track usingImage processing”, IJARIIT, Vol. 3, pp. 489-496, Issue 4, 2017. 7. N. Bhargav, A.
10. Gupta, M. Khirwar, S. Yadav, and V. Sahu, “Automatic Fault Detection ofRailway Track System Based on PLC (ADOR

TAST)”, International Journal  **xi.** Recent Research Aspects, Vol. 3, pp. 91-94, 2016**.**

## 2.3 PROBLEM STATEMENT DEFINITION

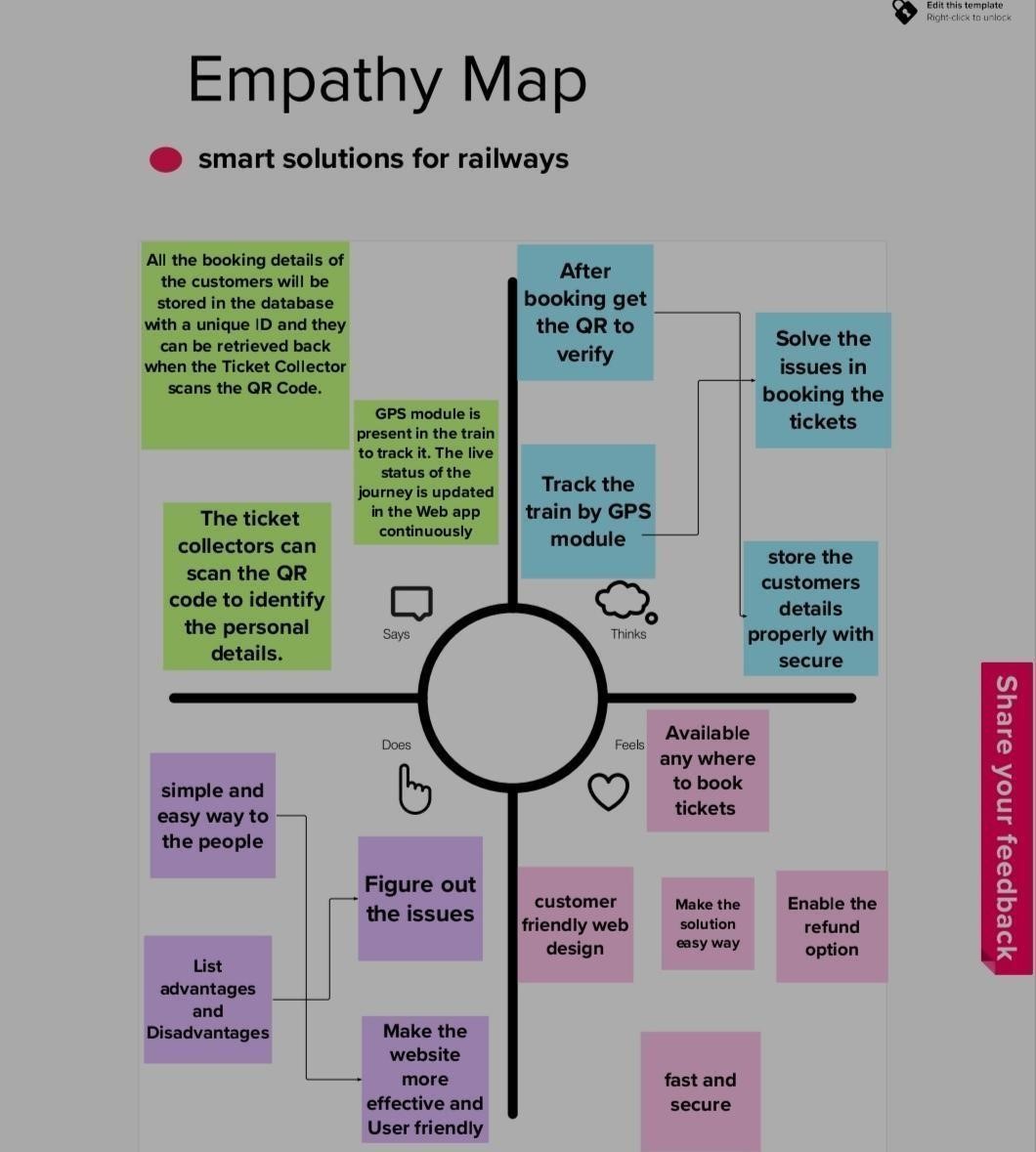
Among the various modes of transport, railways is one of the biggest modes of transport in the world. Though there are competitive threats from airlines, luxury buses, public transports, and personalized

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”

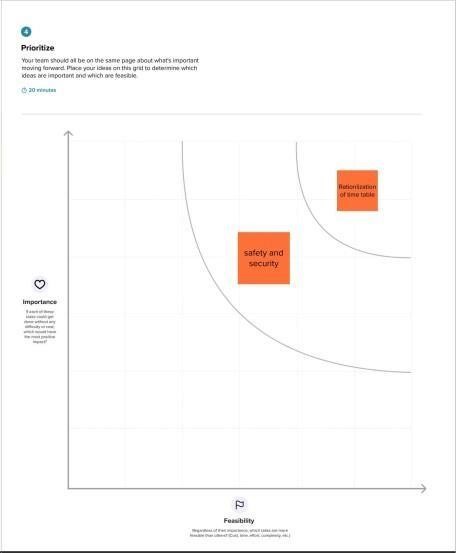
**IDEATION AND PROPOSED SOLUTION**

**3. IDEATION AND PROPOSED SOLUTON**

## 3.1 EMPATHY MAP CANVAS



## 3.2 IDEATION & BRAINSTORMING



## 3.3 PROPOSED SOLUTION

|  |  |  |
| --- | --- | --- |
| **S.NO** | **PARAMETERS** | **DESCRIPTIONS** |
| 1 | Problem  Statement  (Problem to be solved) | In order to satisfy the passengers, the Railways provides various services to its passengers But the passengers can face some problems. |
| 2 | Idea / Solution description | The idea is to minimize the ticket booking problems among the passengers by providing Online mode ofbooking rather than papers. . In queues in front of the ticket counters in railway stations have been drastically increased over the time. |
| 3 | Novelty / Uniqueness | Online mode of booking is most common and so ease of access to everyone that makes more efficient uniqueness  of utilizing the technique. People can book their ticket through online, and they get a QR  code through SMS |
| 4 | Social Impact /  Customer  Satisfaction | Customers for sure they get satisfied as they are in the fast-roaming world this technique makes more easier for travelling passengers. A web page is  designed in which the user can book tickets and will  be provided with the QR code, which will beshown to the ticketcollector and by scanning the QR code the ticket collector will get the passenger details |
| 5 | Business Model (Revenue Model) | A web page is designed in which the user can book tickets and will be provided with the QR code, which will be shown to the ticket collector and byscanningthe QR code the ticket collector will get the passenger  details. The booking details of the user will be stored in the database, which can be retrieved any time |
| 6 | Scalability of the  Solution | The scalability of this solution is most feasible among the passengers who are willing to travel. No need of taking printout Counter ticket has to behandled with  care, but SMS on mobile is enough. No need to taking out wallet and showing your ticket to TTR just tellyour name to TTR that you are a passenger with  valid proof |

## 3.4 PROBLEM SOLUTION FIT



**REQUIREMENT ANALYSIS**

**4.REQUIREMENT ANALYSIS**

## 4.1. 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  ∙One account cannot be associated with multiple users |
| 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 timezone synchronisation when accepting bookings from different timezones |
| FR-5 | Authentication | ∙Booking confirmation should be sent to user to the specified contact details |

## 4.2. NON-FUNCTIONAL

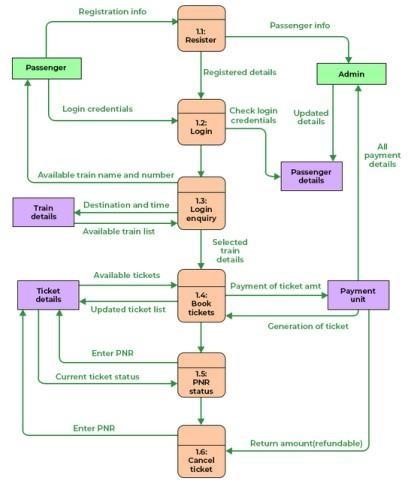
### REQUIREMENTS

|  |  |  |
| --- | --- | --- |
| **FR**  **No.** | **Non-Functional Requirement** | **Description** |
| NFR1 | **Usability** | ∙Search results should populate within acceptable time limits |
| NFR2 | **Security** | ∙System should visually confirm as well as send booking confirmation to the user's contact |
| NFR3 | **Reliability** | ∙System should accept payments via different payment methods, like  PayPal, wallets, cards, vouchers, etc |
| NFR4 | **Performance** | ∙Search results should populate within acceptable time limits |
| NFR5 | **Availability** | ∙User should be helped appropriately to fill in the mandatory fields, incase  of invalid input |
| NFR6 | **Scalability** | ∙Use of captcha and encryption to avoid bots from booking tickets |

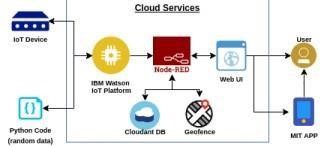
**PROJECT DESIGN**

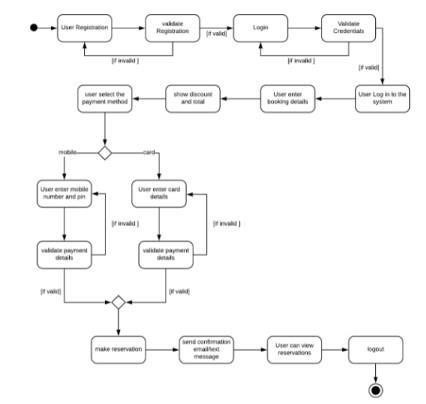
**5.PROJECT DESIGN**

## 5.1 DATA FLOW DIAG



## 5.2 SOLUTION & TECHNICAL ARCHITECTURE





## 5.3 USER STORIES

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional Requirement**  **(Epic)** | **User**  **Story**  **Number** | **User Story / Task** | **Acceptance criteria** | **Priority** | **Release** |
| Customer  (Mobile user, Web user) | Registration | USN-1 | As a user, I can register through the form byFilling in my details | I can register and create my Account/  Dashboard | High | Sprint1 |
|  |  | USN-2 | As a user, I can register throughphonenumbers, Gmail,facebook or other social sites | I can register & create my dashboard with Facebook login or other social sites | High | Sprint2 |
|  | Conformation | USN-3 | As a user, I will  receive confirmation through email orOTP onceregistration is  successful | I can receive confirmation  email & click confirm. | High | Sprint1 |
|  | Authentication/Login | USN-4 | As a user, I can login via login id and password or through OTP received on register phone number | I can login and access my  account/dashboard | High | Sprint1 |
|  | Display Train details | USN-5 | As a user, I can enter the start anddestination to get the list of trains available connecting the above | I can view the  train details  (name & number),correspondingroutes it passes through based on the start and | High | Sprint1 |
|  |  |  |  | destination entered. |  |  |
|  | Booking | USN-6 | As a use, I can provide the basic details such as a  name, age, gender etc… | I will view, modify orconfirm the details enter. | High | Sprint1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Payment | USN-8 | As a user, I can choose to pay through  credit  Card/debit card/UPI. | I can view the paymentoptions available and select mydesirable choice  To proceed with the payment | High | Sprint-1 |
|  |  | USN-9 | As a user, I will be redirected to the  selected  Payment gateway and upon successfulcompletion of payment. Ill be redirected to the booking website | I can pay through thepayment portaland confirm thebooking if any changes need tobe done, I can move back to the initial payment page | High | Sprint-1 |

**PROJECT PLANNING AND SCHEDULING**

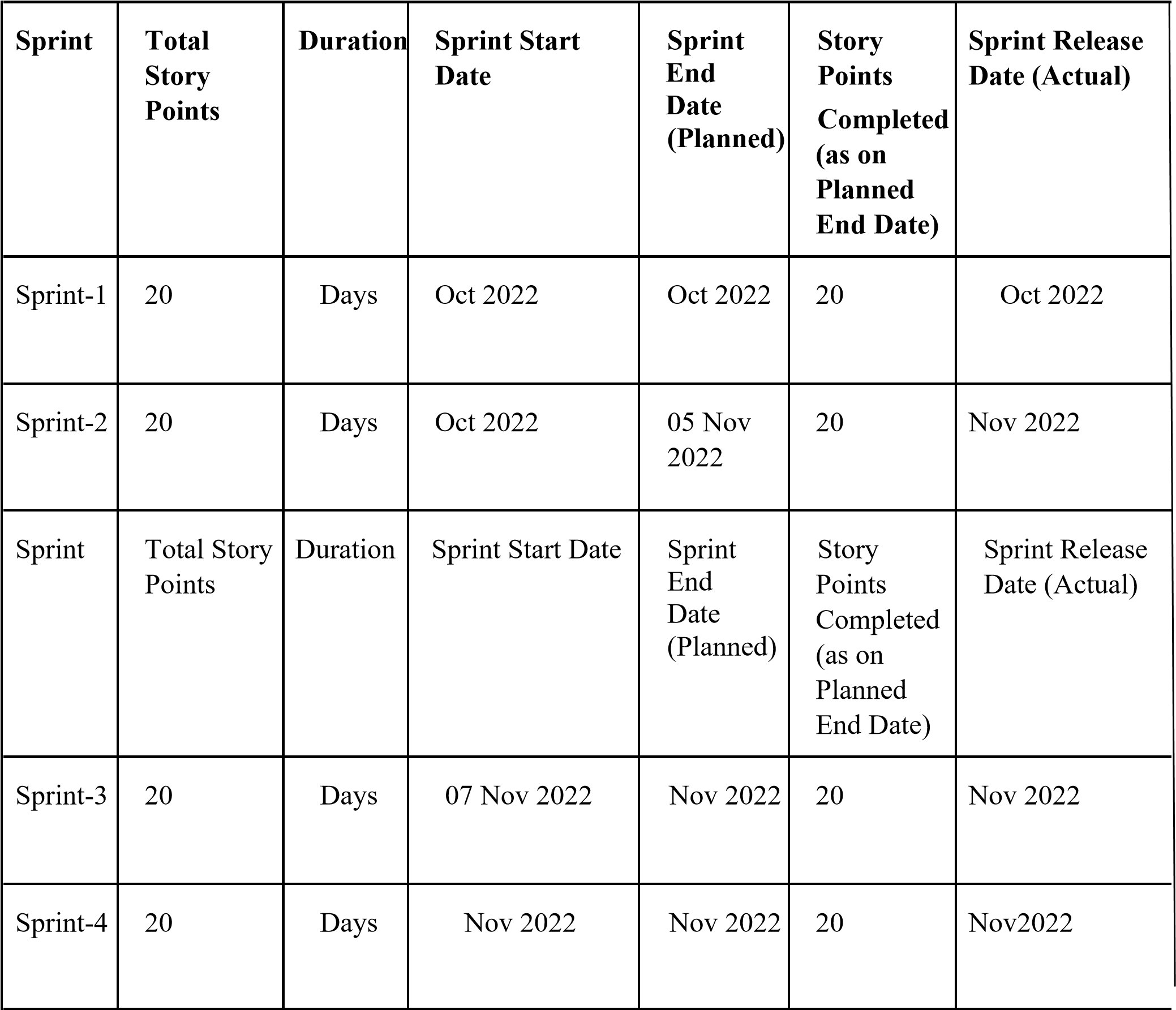
**6.PROJECT PLANNING AND SCHEDULING**

## 6.1. SPRINT PLANNING& ESTIMATION

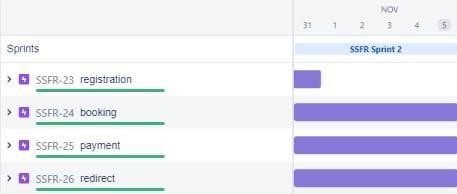
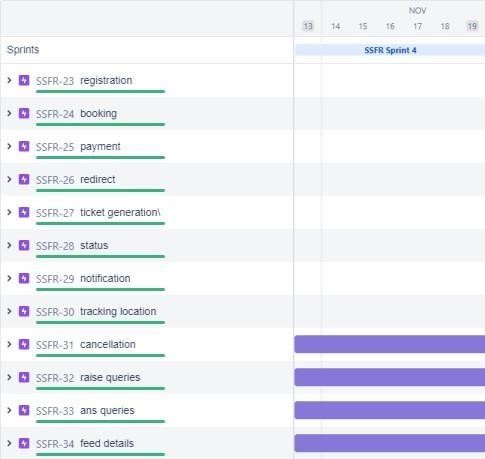
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional**  **Requirement**  **(Epic)** | **User**  **Story**  **Number** | **User Story / Task** | **points** | **Team**  **Members** |
| Sprint-1 | Registration | USN-1 | As a user, I can register throughthe form by  Filling in my details | 2 | Mohamed Safir |
| Sprint-1 |  | USN-2 | As a user, I can register throughphone numbers, Gmail,  Facebook  or other social sites | 1 | Ramakrishnan |
| Sprint-1 | Conformation | USN-3 | As a user, I will receive confirmation through email or OTP once registration is successful | 2 | Prasanna kumar |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sprint-2 | Ticket cancellation | USN-13 | As a user, I can track the train using GPS and can getinformation such as ETA, Current stop and delay | 2 | Palani bala |
| Sprint-3 | Ticket Booked  To Be  Cancelled | USN-14 | As a user, I can cancel my ticketsif there is any change of plan | 1 | Palani Bala |
| Sprint-4 | Raise queries | USN-15 | As a user, I can raise queriesthrough the query box or via mail. | 2 | Ramakrishnan |
| Sprint-4 | Answer the queries | USN-16 | As a user, I will answer the questions/doubts  Raised by the customers. | 2 | Mohamed Safir |
| Sprint-4 | Feed details | USN-17 | As a user, I will feed information about the trains delays and addextra seats if a new compartment is added. | 1 | Mohamed Safir |

## 6.2. SPRINT DELIVERY SCHEDULE



## 6.3. REPORTS FROM JIRA



**CODING AND SOLUTIONING**

## 7. CODING AND SOLUTIONING 1. FEATURES

∙ IOT device

∙ IBM Watson platform

∙ Node red

∙ Cloudant DB

∙ Web UI

**∙** Geofence∙ MIT App

∙ Python code

## 7.2. FEATURE 2

∙ Registration

∙ Login

∙ Verification

∙ Ticket Booking

∙ Payment

∙ Ticket Cancellation

∙ Adding Queries

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="ReEnter 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() :

# Declare a digits variable # which stores all digits digits =

"01234

56789" 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())

digits="01 23456789" OTP=""

for i in range(6):

OTP+=digits[math.floor(random.ran dom()\*10)] otp = OTP + " is your

OTP"

msg= otp s = smtplib.SMTP('smtp.gmail.co m', 587)

s.starttls()

s.login("Your Gmail Account", "You app password") emailid = input("Enter your email: ")

s.sendmail('&&&&&&&&&&&'

,emailid,msg) a = input("Enter Your OTP >>: ") if a == OTP: print("Verified")

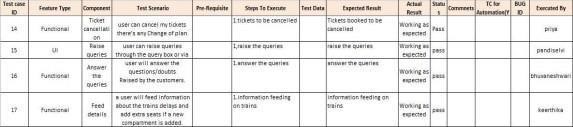
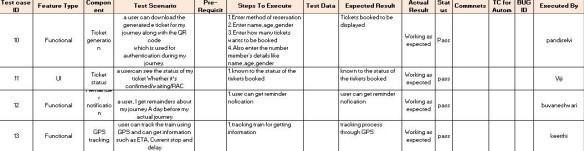
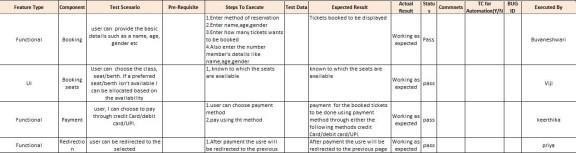
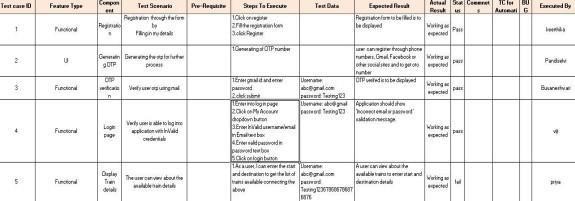
else:

print("Please Check your OTP again")

**TESTING**

**8.TESTING**

### 8.1.TEST CASES



**RESULTS**

**9.RESULTS**

## 9.1. PERFORMANCE METRICS



**ADVANTAGES &DISADVANTAGES**

**10.ADVANTAGES &DISADVANTAGES**

## 10.1. 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.

## 10.2. 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 many 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**

1. **APPENDIX**

**13.1. SOURCE PROGRAM**

import math, random

import os import smtplib import sqlite3 import requests from bs4 import BeautifulSoup from django.contrib.auth.base\_user import AbstractBaseUser from django.db import models

import logging import pandas as pd import pyttsx3 from plyer import notification

import time import numpy as np import matplotlib.pyplot as plt from PIL import Image, ImageDraw from pickle import load,dump

import smtplib, ssl from email.mime.text import MIMEText from email.mime.multipart import MIMEMultipart import

email from email import encoders from email.mime.base import MIMEBase 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 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() :

# 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()) digits="0123456789" OTP="" for i in range(6):

OTP+=digits[math.floor(random.random()\*10)] otp = OTP + " is your OTP" msg= otp s = smtplib.SMTP('smtp.gmail.com', 587)

s.starttls()

s.login("Your Gmail Account", "You app password") emailid = input("Enter your email:

")

s.sendmail('&&&&&&&&&&&',emailid,msg) a = input("Enter Your OTP >>: ") if a == OTP: print("Verified") else:

print("Please Check your OTP again")

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)

USERNAME = StringVar()

PASSWORD = StringVar()

Top = Frame(root, bd=2, relief=RIDGE)

Top.pack(side=TOP, fill=X)

Form = Frame(root, height=200) Form.pack(side=TOP, pady=20) 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) 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) 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()

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 getdata(url):

r = requests.get(url) return r.text

# input by geek from\_Station\_code = "GAYA" from\_Station\_name = "GAYA"

To\_station\_code = "PNBE"

To\_station\_name = "PATNA"

# url url = "https://www.railyatri.in/booking/trainsbetween stations?from\_code="+from\_Station\_code+"&from\_name="+from\_Stat ion\_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) 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 7.2. FEATURE 2

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")

@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 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

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

# user define function

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

# input by geek train\_name = "03391-rajgir-new-delhi-clonespecial-rgd-to-ndls"

# url url = "https://www.railyatri.in/live-trainstatus/"+train\_name # pass the url # into getdata function htmldata = getdata(url) soup = BeautifulSoup(htmldata, 'html.parser')

# traverse the live status from

# this Html code data = [] for item in soup.find\_all('script', type="application/ld+json"):

data.append(item.get\_text())

# convert into dataframe df = pd.read\_json(data[2])

# display this column of # dataframe print(df["mainEntity"][0]['name']) print(df["mainEntity"][0]['acceptedAnswer']['text'] ) 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:

70

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()

data\_path = 'data.csv' data = pd.read\_csv(data\_path, names=['LATITUDE', 'LONGITUDE'], sep=',') gps\_data = tuple(zip(data['LATITUDE'].values, data['LONGITUDE'].values)) image = Image.open('map.png', 'r') # Load map image. img\_points = [] for d in gps\_data:

x1, y1 = scale\_to\_img(d, (image.size[0], image.size[1])) # Convert GPS coordinates to image coordinates.

img\_points.append((x1, y1))

draw = ImageDraw.Draw(image) draw.line(img\_points, fill=(255, 0, 0), width=2) # Draw converted records to the map image.

image.save('resultMap.png') x\_ticks = map(lambda x: round(x, 4), np.linspace(lon1, lon2, num=7)) y\_ticks = map(lambda x: round(x, 4), np.linspace(lat1, lat2, num=8)) y\_ticks = sorted(y\_ticks, reverse=True) # y ticks must be reversed due to conversion to image coordinates.

fig, axis1 = plt.subplots(figsize=(10, 10)) axis1.imshow(plt.imread('resultMap.png')) # Load the image to matplotlib plot. axis1.set\_xlabel('Longitude') axis1.set\_ylabel('Latitude') axis1.set\_xticklabels(x\_ticks) axis1.set\_yticklabels(y\_ticks) axis1.grid() plt.show() class tickets:

def

\_\_init\_\_(self): self.no\_ofac1s tclass=0 self.totaf=0 self.no\_ofac2n dclass=0 self.no\_ofac3r dclass=0 self.no\_ofslee per=0 self.no\_ofticke ts=0 self.name='' self.age='' self.resno=0 self.status='' def ret(self):

return(self.resno) def retname(self): return(self.name) def display(self): f=0 fin1=open("tickets.dat","rb") if not fin1:

print "ERROR" else: print n=int(raw\_input("ENTER PNR NUMBER : ")) print "\n\n" print ("FETCHING DATA . . .".center(80))

time.sleep(1) print print('PLEASE

WAIT...!!'.center(80)) time.sleep(1) os.system('cls') try: while True: tick=load(fin1) if(n==tick.ret()): f=1 print "="\*80 print("PNR STATUS".center(80)) print"="\*80 print print

"PASSENGER'S NAME :",tick.name print print

"PASSENGER'S AGE :",tick.age print print "PNR NO :",tick.resno print print "STATUS

:",tick.status print

print "NO OF SEATS BOOKED : ",tick.no\_oftickets print except: pass fin1.close() if(f==0): print print "WRONG PNR NUMBER..!!" print def pending(self):

self.status="WAITING LIST" print "PNR NUMBER

:",self.resno print time.sleep(1.2) print "STATUS = ",self.status print print "NO OF

SEATS BOOKED :

",self.no\_oftickets print def confirmation (self):

self.status="CONFIRMED" print "PNR NUMBER : ",self.resno print time.sleep(1.5) print "STATUS = ",self.status print def cancellation(self): z=0 f=0 fin=open("tickets.dat","rb") fout=open("temp.dat","ab") print r= int(raw\_input("ENTER PNR NUMBER : ")) try: while(True): tick=load(fin) z=tick.ret() if(z!=r):

dump(tick,fout) elif(z==r): f=1 except: pass fin.close() fout.close() os.remove("tickets.

dat") os.rename("temp.d

at","tickets.dat") if

(f==0): print

print "NO SUCH RESERVATION NUMBER FOUND" print time.sleep(2) os.system('cls') else: print print "TICKET CANCELLED" print"RS.600 REFUNDED...." def reservation(self):

trainno=int(raw\_input("ENTER THE TRAIN NO:")) z=0 f=0 fin2=open("tr1details.dat" ) fin2.seek(0) if not fin2:

print "ERROR" else: try: while True:

tr=load(fin2) z=tr.gettrainno() n=tr.gettrainname() if (trainno==z):

print print "TRAIN NAME

IS : ",n f=1 print print "-"\*80 no\_ofac1st=tr.getno\_ofac1stclass() no\_ofac2nd=tr.getno\_ofac2ndclass() no\_ofac3rd=tr.getno\_ofac3rdclass() no\_ofsleeper=tr.getno\_ofsleeper() if(f==1):

fout1=open("tickets.dat","ab") print

self.name=raw\_input("ENTER THE PASSENGER'S NAME ") print self.age=int(raw\_input("PASSENGER'S AGE : ")) print print"\t\t SELECT A CLASS YOU WOULD LIKE TO TRAVEL IN

:- "

print "1.AC FIRST CLASS" print print "2.AC SECOND CLASS" print print "3.AC THIRD CLASS" print print "4.SLEEPER CLASS"

print c=int(raw\_input("\t\t\tENTER YOUR CHOICE = ")) os.system('cls') amt1=0 if(c==1):

self.no\_oftickets=int(raw\_input("ENTER NO\_OF FIRST CLASS AC SEATS TO BE BOOKED : ")) i=1

while(i<=self.no\_oftickets): self.totaf=self.totaf+1 amt1=1000\*self.no\_ofticket s i=i+1 print print "PROCESSING. .",

time.sleep(0.5) print ".", time.sleep(0.3) print'.' time.sleep(2) os.system('cls') print "TOTAL AMOUNT TO BE PAID = ",amt1 self.resno=int(random.randint(1000,2546)) x=no\_ofac1st- self.totaf print if(x>0):

self.confirmation( ) dump(self,fout1) break else: self.pending() dump(tick,fout1) break elif(c==2):

self.no\_oftickets=int(raw\_input("ENTER NO\_OF SECOND CLASS AC SEATS TO BE BOOKED : ")) i=1 def menu(): tr=train() tick=tickets() print print "WELCOME TO PRAHIT

AGENCY".center(80) while True: print print "="\*80 print " \t\t\t\t RAILWAY" print print "="\*80 s print print "\t\t\t1. \*\*UPDATE TRAIN DETAILS." print print "\t\t\t2. TRAIN DETAILS. " print print "\t\t\t3. RESERVATION OF TICKETS." print print "\t\t\t4. CANCELLATION OF TICKETS. " print print "\t\t\t5. DISPLAY PNR STATUS." print "\t\t\t6. QUIT." print"\*\* - office use......" ch=int(raw\_input("\t\t\tENTER YOUR CHOICE :

")) os.system('cls') print

"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\tLOADI

NG. .", time.sleep(1) print ("."), time.sleep(0.5) print (".") time.sleep(2) os.system('cls') if ch==1: 90 j="\*\*\*\*\*"

r=raw\_input("\n\n\n\n\n\n\n\n\n\n\n\t\t\t\tENTER THE

PASSWORD: ") os.system('cls') if (j==r): x='y' while (x.lower()=='y'): fout=open("tr1details.dat","ab")

tr.getinput() dump(tr,fout) fout.close() print"\n\n\n\n\n\n\n\n\n\n\n\t\t\tUPDATING TRAIN LIST

PLEASE WAIT . .",

time.sleep(1) print ("."), time.sleep(0.5) print ("."), time.sleep(2) os.system('cls') print

"\n\n\n\n\n\n\n\n\n\n\n"

x=raw\_input("\t\tDO YOU WANT TO ADD ANY MORE TRAINS DETAILS ? ")

os.system('cls') continue elif(j<>r): print"\n\n\n\n\n" print "WRONG PASSWORD".center(80) elif ch==2: fin=open("tr1details.dat",'rb') if not fin:

91 print

"ERROR"

tick.display() elif ch==6:

quit()

raw\_input("PRESS ENTER TO GO TO BACK

MENU".center(80)) os.system('cls') menu() sender\_email =

"my@gmail.com" receiver\_email =

"your@gmail.com"

password = input("Type your password and press enter:")

message = MIMEMultipart("alternative") message["Subject"] = "multipart test" message["From"] = sender\_email message["To"] = receiver\_email

# Create the plain-text and HTML version of your message text = """\ Hi,

How are you?

Real Python has many great tutorials:

www.realpython.com""" html = """\ <html> <body>

<p>Hi,<br>

How are you?<br>

92

<a href="http://www.realpython.com">Real Python</a> has many great tutorials.

</p>

</body>

</html>

"""

# Turn these into plain/html MIMEText objects part1 = MIMEText(text, "plain") part2 = MIMEText(html, "html")

# Add HTML/plain-text parts to MIMEMultipart message

# The email client will try to render the last part first message.attach(part1) message.attach(part2)

# Create secure connection with server and send email context = ssl.create\_default\_context() with smtplib.SMTP\_SSL("smtp.gmail.com", 465, context=context) as server: server.login(sender\_email, password) server.sendmail( sender\_email, receiver\_email, message.as\_string()

)

subject = "An email with attachment from Python" body = "This is an email with attachment sent from Python"

sender\_email = "my@gmail.com" receiver\_email = "your@gmail.com" password = input("Type your password and press enter:")

93

# Create a multipart message and set headers

message = MIMEMultipart() message["From"] = sender\_email message["To"] = receiver\_email message["Subject"] = subject

message["Bcc"] = receiver\_email # Recommended for mass emails

# Add body to email

message.attach(MIMEText(body, "plain")) filename =

"document.pdf" # In same directory as script

# Open PDF file in binary mode with open(filename, "rb") as attachment: # Add file as application/octet-stream

# Email client can usually download this automatically as attachment part = MIMEBase("application", "octet-stream") part.set\_payload(attachment.read())

# Encode file in ASCII characters to send by email encoders.encode\_base64(part)

# Add header as key/value pair to attachment part

part.add\_header( "ContentDisposition", f"attachment; filename= {filename}",

)

# Add attachment to message and convert message to string message.attach(part)

94 text = message.as\_string() # Log in to server using secure context and send email context = ssl.create\_default\_context() with smtplib.SMTP\_SSL("smtp.gmail.com", 465, context=context) as server: server.login(sender\_email, password) server.sendmail(sender\_email, receiver\_email, text) api\_key = "Your\_API\_key"

# base\_url variable to store url

base\_url = "https://api.railwayapi.com/v2/pnr-status/pnr/"

# Enter valid pnr\_number pnr\_number = "6515483790"

# Stores complete url address

complete\_url = base\_url + pnr\_number + "/apikey/" + api\_key + "/"

# get method of requests module # return response object response\_ob = requests.get(complete\_url)

# json method of response object convert # json format data into python format data result = response\_ob.json() # now result contains list

# of nested dictionaries if result["response\_code"] == 200:

95

# train name is extracting # from the result variable data train\_name = result["train"]["name"]

# train number is extracting from # the result variable data train\_number = result["train"]["number"]

# from station name is extracting # from the result variable data from\_station = result["from\_station"]["name"]

# to\_station name is extracting from # the result variable data to\_station = result["to\_station"]["name"]

# boarding point station name is # extracting from the result variable data boarding\_point = result["boarding\_point"]["name"]

# reservation upto station name is # extracting from the result variable data reservation\_upto = result["reservation\_upto"]["name"]

# store the value or data of "pnr" # key in pnr\_num variable pnr\_num = result["pnr"] 96

# store the value or data of "doj" key # in variable date\_of\_journey variable date\_of\_journey = result["doj"]

# store the value or data of # "total\_passengers" key in variable total\_passengers = result["total\_passengers"]

# store the value or data of "passengers" # key in variable passengers\_list passengers\_list = result["passengers"]

# store the value or data of # "chart\_prepared" key in variable chart\_prepared = result["chart\_prepared"]

# print following values print(" train name : " + str(train\_name) + "\n train number :

" + str(train\_number)

+ "\n from station : " + str(from\_station)

+ "\n to station : " + str(to\_station)

+ "\n boarding point : " + str(boarding\_point)

+ "\n reservation upto : " + str(reservation\_upto)

+ "\n pnr number : " + str(pnr\_num)

+ "\n date of journey : " + str(date\_of\_journey)

+ "\n total no. of passengers: " + str(total\_passengers)

+ "\n chart prepared : " + str(chart\_prepared))

# looping through passenger list

97 for passenger in passengers\_list:

# store the value or data # of "no" key in variable passenger\_num = passenger["no"]

# store the value or data of # "current\_status" key in variable current\_status = passenger["current\_status"]

# store the value or data of # "booking\_status" key in variable booking\_status = passenger["booking\_status"]

# print following values print(" passenger number : " + str(passenger\_num) + "\n current status : " + str(current\_status) + "\n booking\_status : " + str(booking\_status)) else:

print("Record Not Found") Github Link

https://github.com/IBM-EPBL/IBM-Project-13888-1659534615