Sprint 3

Date	11-Nov-22
Team ID	PNT2022TMID18159
Project Name	Smart solutions for railways

Generating Ticket:

```
class Ticket:     counter=0      def
__init__(book,passenger_name,source,destination):
book.__passenger_name=passenger_name
       book.__source=source         book.__destination=destination
book.Counter=Ticket.counter         Ticket.counter+=1     def
validate_source_destination(book):         if (book.__source=="Chennai"
and (book.__destination==”Delhi" or book.__destination=="Pune" or
book.__destination=="Mumbai" or book.__destination=="Kolkata")):
          return True
else:
          return  False          def
generate_ticket(book ):        if
True:
          __ticket_id=book.__source[0]+book.__destination[0]+"0"+str(book.Counter)
print( "Ticket id :",__ticket_id)         else:
          return False     def
get_ticket_id(book):         return
book.ticket_id     def
get_passenger_name(book):
return book.__passenger_name
def get_source(book):         if
book.__source==”Chennai":
          return book.__source
else:
          print("Invalid source option")
return None     def
get_destination(book):         if
book.__destination=="Delhi":
return book.__destination         elif
book.__destination=="Pune":
          return book.__destination         elif
book.__destination=="Mumbai":
          return book.__destination
elif book.__destination=="Kolkata":
          return book.__destination
else:
          return None
```

Ticket Status:

```
import requests from bs4
import BeautifulSoup
import pandas as pd
```

```
def getdata(url):
        r = requests.get(url)
        return r.text

train_name = "18159-chennaiexpress-new-delhi-special-cape-to-ndls"

url = "https://www.railyatri.in/live-train-status/"+train_name

htmldata = getdata(url)
soup = BeautifulSoup(htmldata, 'html.parser')

data = [] for item in soup.find_all('script',
type="application/ld+json"):  data.append(item.get_text())

df = pd.read_json(data[2])

print(df["mainEntity"][0]['name'])
print(df["mainEntity"][0]['acceptedAnswer']['text'])
```

Reminder:

```
import pyttsx3 from plyer
import notification import
time def Speak(self, audio):
        engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)
        engine.say(audio)
engine.runAndWait()  def
Take_break():
Speak("Shall we begin?")
question = input()                if
"yes" in question:
                Speak("Lets Start")
if "no" in question:
                Speak("Process begin in 2mins")
        time.sleep(2*60)
Speak("Started")

        while(True):
  notification.notify(title="Let's Start ",   message="will tell you to
take a break after 50 mins",
                timeout=10)
                time.sleep(0.5*60)
                Speak("Have a break ")
```

```
                notification.notify(title="Notification",
message="Please do continue the process after break",
                timeout=10)

if __name__ == '__main__':
        Take_break()
```

Location Tracking:

```
import pandas as pd import
numpy as np import
matplotlib.pyplot as plt
from PIL import Image, ImageDraw

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')   img_points = [] for d
in gps_data:     x1, y1 = scale_to_img(d, (image.size[0],
image.size[1]))          img_points.append((x1, y1))
draw = ImageDraw.Draw(image)
draw.line(img_points, fill=(255, 0, 0), width=2)

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)

fig, axis1 = plt.subplots(figsize=(10, 10))
axis1.imshow(plt.imread('resultMap.png'))  .
axis1.set_xlabel('Longitude')
axis1.set_ylabel('Latitude')
axis1.set_xticklabels(x_ticks)
axis1.set_yticklabels(y_ticks)
axis1.grid() plt.show()
```