SPRINT 02

Date	05 November 2022
Team ID	PNT2022TMID11066
Project Name	Smart solutions for railways
Maximum Marks	20 marks

TICKET BOOKING:

```
#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
```

SEATS BOOKING:

```
#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
```

REDIRECT:

```
#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"))
```

GPS TRACKING:

```
#GPS_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')  # 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()
```