Project Report

1. **INTRODUCTION**
   - 1.1 Project Overview
   - 1.2 Purpose
2. **LITERATURE SURVEY**
   - 2.1 Existing problem
   - 2.2 References
   - 2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   - 3.1 Empathy Map Canvas
   - 3.2 Ideation & Brainstorming
   - 3.3 Proposed Solution
   - 3.4 Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   - 4.1 Functional requirement
   - 4.2 Non-Functional requirements
5. **PROJECT DESIGN**
   - 5.1 Data Flow Diagrams
   - 5.2 Solution & Technical Architecture
   - 5.3 User Stories
6. **PROJECT PLANNING & SCHEDULING**
   - 6.1 Sprint Planning & Estimation
   - 6.2 Sprint Delivery Schedule
   - 6.3 Reports from JIRA 7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   - 7.1 Feature 1
   - 7.2 Feature 2
   - 7.3 Database Schema (if Applicable)
8. **TESTING**
   - 8.1 Test Cases
   - 8.2 User Acceptance Testing
9. **RESULTS**
   - 9.1 Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

    Source Code

GitHub & Project Demo Link

1.INTRODUCTION

1.1 PROJECT OVERVIEW

This paper is mainly streamed towards child safety solutions by developing a gadget which can be tracked via its GPS locations and also a panic button on gadget is provided to alert the parent via GSM module calling for help. Parental android app is developed to manage and track the device anytime. Smart gadget device is always connected to parental phone which can receive and make phone calls and also receive SMS on gadget via GSM module, also a wireless technology is implemented on device which is useful to bound the device within a region of monitoring range, if device is moving out of monitoring range then an alert will be triggered on binding gadget, this helps you keep a virtual eye on child. Health monitoring system on gadget checking for parameters like heart beat/pulse rate and temperature is included which can be monitored on parental app. Gadget also monitors whether it is plugged on hand or not using contact switch and alert the parent as soon as it is unplug

1.2 PURPOSE

The internet of things (IoT) refers to the set of devices and system that stay interconnected with real-world sensor and to the internet. During years' Child safety is under threat and it is very important to provide a technology-based solution which will help them under panic situations and monitor them using a smart gadget. The

proposed system is equipped with GSM and GPS modules for sending and receiving call and SMS between safety gadget and parental phone, the proposed system also consists of Wi-Fi module used to implement IoT and send all the monitoring parameters to the cloud for android app monitoring on parental phone. Android application can be used to track the current location of safety gadget using its location coordinates on parental phone android app and also via SMS request from parent phone to safety gadget. Panic alert system is used during panic situations and automatic SMS alert and phone call is triggered from safety gadget to the parental phone seeking for help and also monitored for plug and unplug from hand, as soon the gadget is unplugged from hand a SMS is triggered to parental phone and the alert parameter is also updated to the cloud.

Heart-beats, temperature is monitored and the values are updated to cloud continuously for parent app monitoring. Boundary monitoring system is implemented on safety gadget with the help of BEACON technology, as soon as the safety gadget moves far away from the binding gadget an alert is provided to parent on binding gadget. the system is used to monitor the health parameters and also used for location tracking during necessary situations in safety concern.

2.LITERATURE SURVEY

2.1 EXISTING SYSTEM

[1] Authors: M Nandini Priyanka, S Murugan, K. N. H. Srinivas, T. D. S. Sarveswararao, E. Kusuma Kumari.
Title: Smart IoT Device for Child Safety and Tracking.
Published in: 2019 IEEE.
The system is developed using Link-It ONE board programmed in embedded C and interfaced with temperature, heartbeat, touch sensors and also GPS, GSM & digital camera modules. The novelty of the work is that the system automatically alerts the parent/caretaker by sending SMS, when immediate attention is required for the child during emergency. Merits: The parameters such as touch, temperature & heartbeat of the child are used for parametric analysis and results are plotted for the same.
Demerits: To implement the IoT device which ensures the complete solution for child safety problems. [2] Authors: Akash Moodbidri, Hamid Shahnasser Title: Child safety wearable device.
Published in: 2017 IEEE.
The purpose of this device is to help the parents to locate their children with ease. At the moment there are many wearable's in the market which helps to track the daily activity of children and also helps to find the child using Wi-Fi and Bluetooth services present on the device.
Merits: This wearable over other wearable is that it can be used in any phone and it is not necessary that an expensive smartphone is required and doesn't want to be very tech savvy individual to operate.
Demerits: As, this device's battery gives short life-time. High power efficient model will have to be used which can be capable of giving the battery life for a longer time.

[3] Authors: Aditi Gupta, Vibhor Harit.
Published in: 2016 IEEE.
Title: Child Safety & Tracking Management System by using GPS.
This paper proposed a model for child safety through smart phones that provides the option to track the location of their children as well as in case of emergency children is able to send a quick message and its current location via Short Message services.
Merits: The advantages of smart phones which offers rich features like Google maps, GPS, SMS etc.
Demerits: This system is unable to sense human behavior of child.
[4] Authors: Dheeraj Sunehera, Pottabhatini Laxmi Priya. Title: Children Location Monitoring on Google Maps Using GPS and GSM.
Published in: 2016 IEEE.
This paper provides an Android based solution for the parents to track their children in real time. Different devices are connected with a single device through channels of internet. The concerned device is connected to server via internet. The device can be used by parents to track their children in real time or for women safety. The proposed solution takes the location services provided by GSM module. It allows the parents to get their child's current-location via SMS.
Merits: A child tracking system using android terminal and hoc networks.
Demerits: This device cannot be used in rural areas.

2.3 PROBLEM STATEMENT DEFINITION

Safety Gadget
Figure 1 shows the block diagram of the proposed child safety device. It consists of inbuilt Wi-Fi, GSM, GPS and Bluetooth modules. The link it one board is similar to the

Arduino board and it is termed as all-in-one prototyping board for safety and IoT devices. The link it one is a robust development board for the hardware and also used for industrial applications. Different components such as temperature sensor, heartbeat sensor, panic button, contact switch are connected to the link it ONE board along with built in GSM, GPS modules. Safety gadget consists of BEACON and BLE packet is transmitted through it, this packet is received by binding gadget which has BLE receiver module, the packet usually contains information such as identification number, signal strength etc. Temperature is one of the most commonly measured variables. For measuring body temperature of the child DS18B20 temperature sensor is used. The heartbeat sensor is used in the proposed system for measuring the pulse rate. There is a heartbeat/pulse sensor which is combined to simple optical heart rate sensor with amplification and nullification circuitry making it is fast and easy to get reliable pulse reading. The GSM/GPRS block is activated with a SIM card on the board.

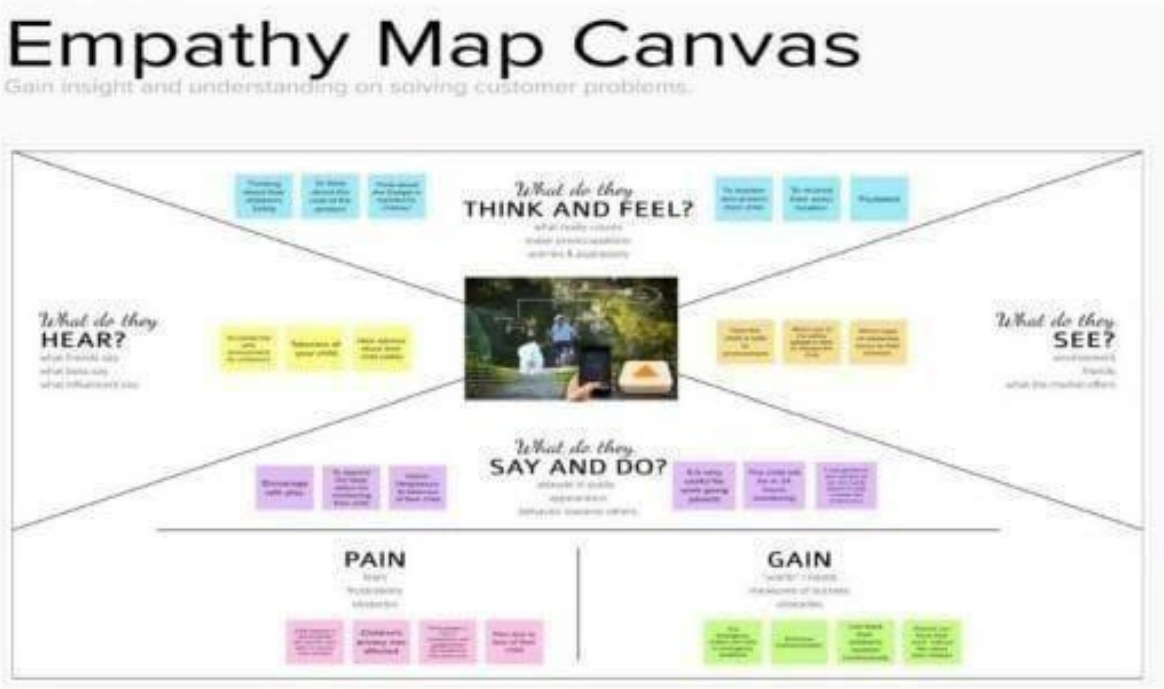
3. IDEATION AND PROPOSED SOLUTON

They mainly differ based on bandwidth and RF carrier frequency. GSM network consists of mobile station, base station subsystem network and operation subsystem. The GPS module is provided for identifying the location of the child. GPS module receives the signals from satellites. The latitude and longitude of the location can be identified by the GPS module. The device sends the monitored parameters data such as temperature and pulse rate to cloud. If any abnormalities occurs in temperature or pulse rate readings, a SMS and call triggers to the parent/caretaker mobile phone immediately and also updated to the mobile app only for the registries mobile no. We can use mobile application, cloud and database as the back end of

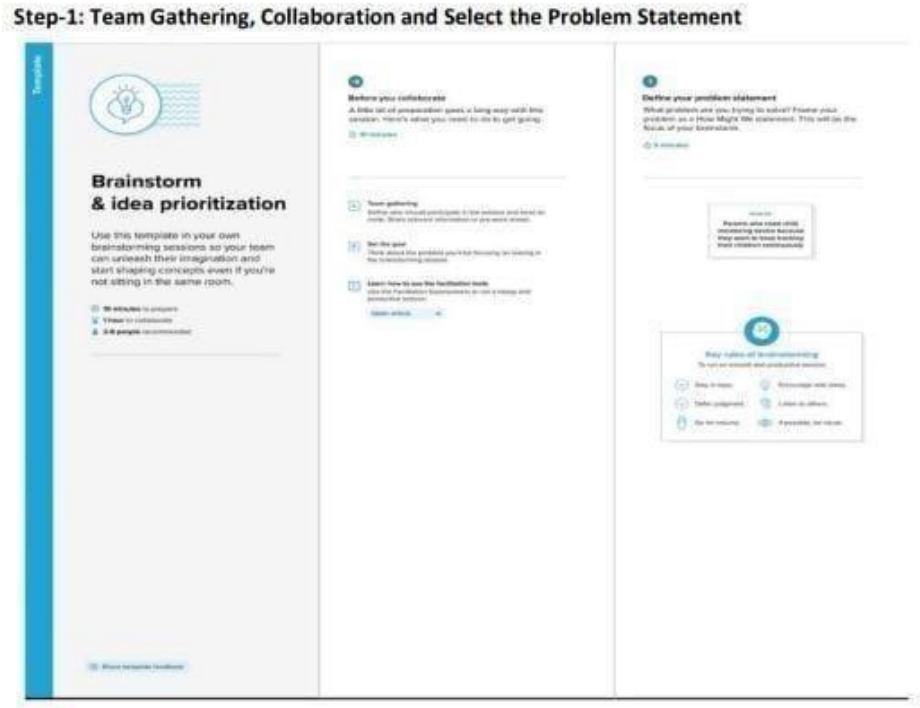
storing and retrieving information and also a device for monitoring.Fig. 1. Block diagram of smart gadget

B. BLE Listener deviceFigure 2 shows the BLE Listener device is the device which is used to satisfy this feature along with safety gadget and parental phone. This gadget is also used to monitor safety gadget within a bounded area using wireless technology as follows, this feature of binding gadget is designed to work independently without phone network signal/internet so that safety gadget can even be under monitoring when it reaches remote areas where communication signals is not reachable like forest. Safety gadget consists of BEACON and BLE packet is transmitted through it, this packet is received by binding gadget which has BLE (Bluetooth Low Energy) receiver module, the packet usually contains information such as identification number, signal strength etc. Whenever the packet is received it checks for all the above information in the receiver device. As the distance between safety gadget and binding gadget increases, the signal strength decreases. Once the safety gadget is moving out of threshold distance from the binding gadget then an alert is provided on binding gadget which will be used by parent/guardian. P

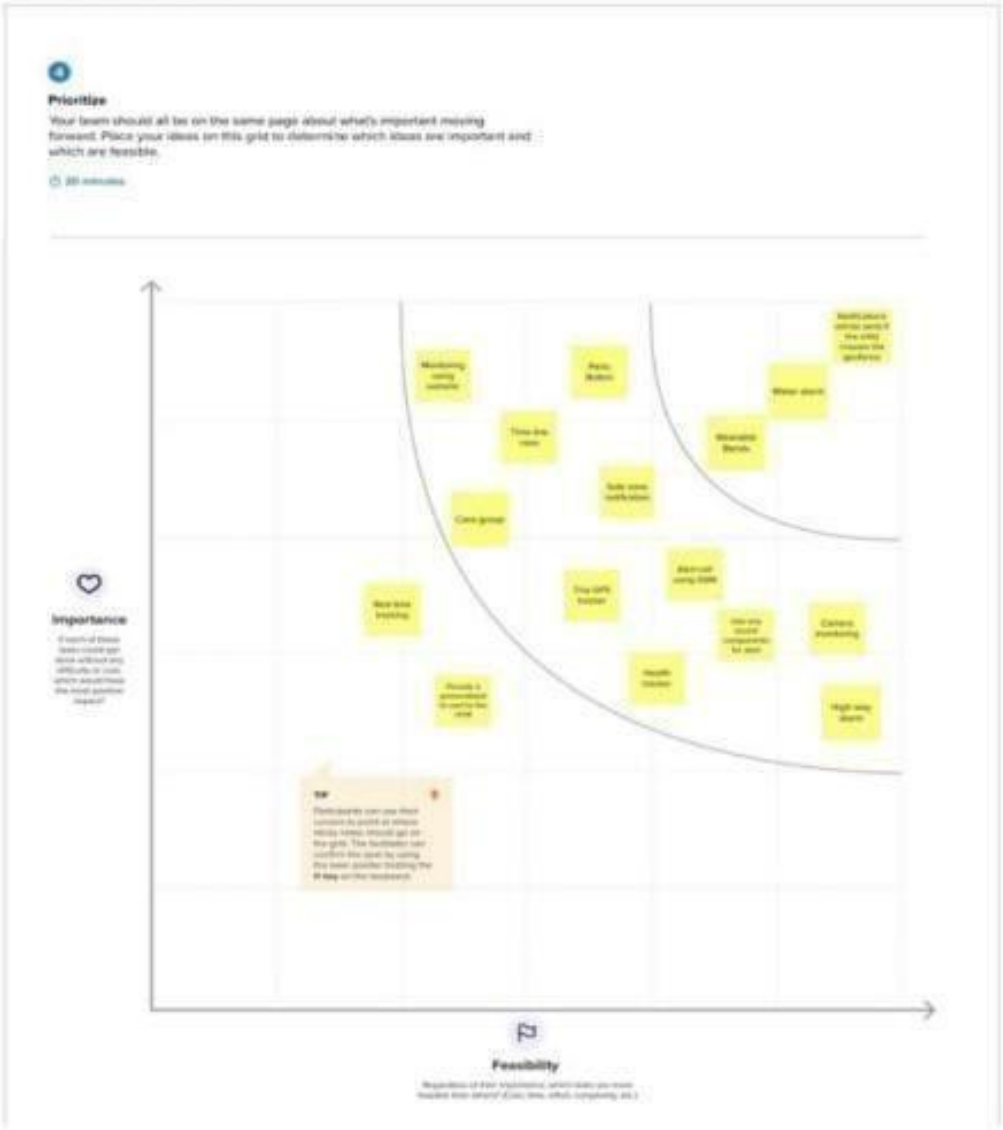
3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING



Step-3: Idea Prioritization



3.3 PROPOSED SOLUTION

S.NO	PARAMETERS	DESCRIPTIONS

1	Problem Statement (Problem to be solved)	Parents who need child monitoring device because they want to keep tracking their children continuously.
2	Idea / Solution description	Create a Child tracker which helps parents with continuously monitoring the child'slocation. The notification will be sent according to the child's location to their parents or caretakers. The entire location data will be stored in the database
3	Novelty / Uniqueness	The novelty of the work is that the system automatically alerts the parent/caretaker by sending notification**, when** immediate attention **is** required **for the** child during emergency
4	Social Impact / Customer Satisfaction	Make children parents more assure about their kid's security, we have a feature in our device called Geo-Fence. Geo-Fencing feature allows you to mark a particular area as safe-zone. Whenever your child crosses that specific area, you will get an instant notification on your phone.

5	Business Model (Revenue Model)	● Easy to use ● Low cost ● Weightless ● Compatible
6	Scalability of the Solution	● Gadget ensures the safety and tracking of the children. ● Parents need not worry about their children.

3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

4.1. FUNCTIONAL REQUIREMENTS

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**Temperature**	If the temperature level exceeds the room temperature then the alert message will be sent using GSM **to** the specified users**.**
FR-2	Pulse sensors	The Pulse sensor is used to detect any abnormal feelings experienced by the child like fear, anxiety, nervousness, drowsiness and several other illnesses which manipulates the normal heart rate.
FR-3	GPS	GPS is used to track the live location of the child who is wearing that device.With the help of GPS, we can easily perform Geo- fencing concept, in which we can feed a particular boundary to that device

FR-4	GSM	If the child goes beyond that particular boundary specified, the respective guardians will receive analert call using GSM
FR-5	Web camera	we can monitor the child lively through live video streaming whenever we get notified in abnormal cases.
FR-6	Raspberry pi microprocessor	Raspberry Pi microprocessor in which all other sensors, GPS and GSM are integrated. The users are required to register using their credentials to use the application.

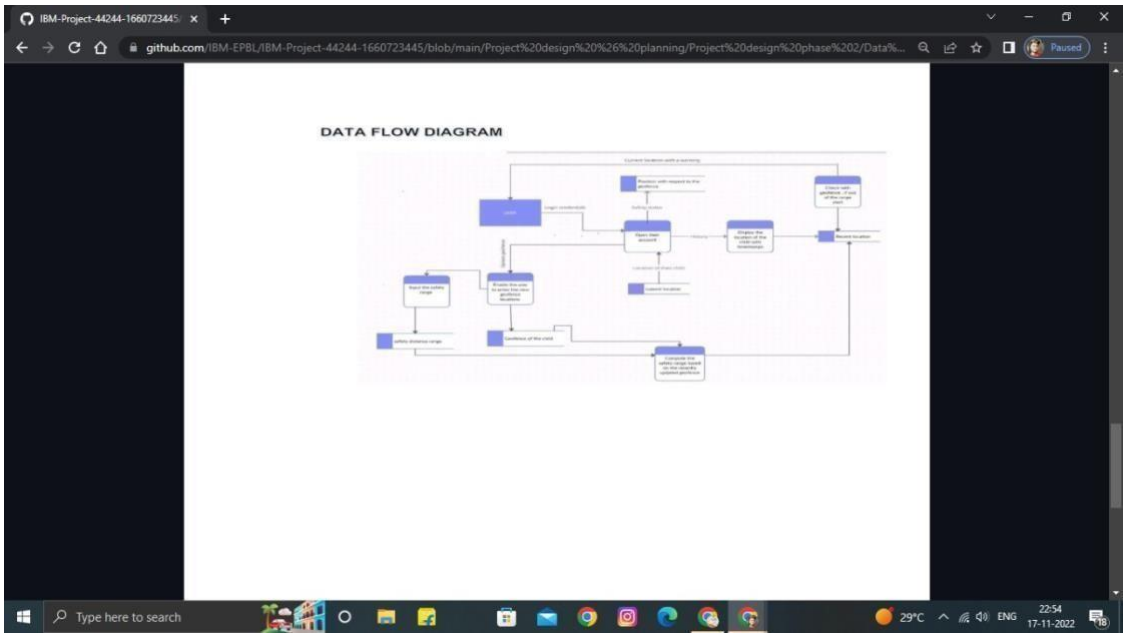
4.2. NON-FUNCTIONAL REQUIREMENTS

FR No.	Non-Functional Requirement	Description

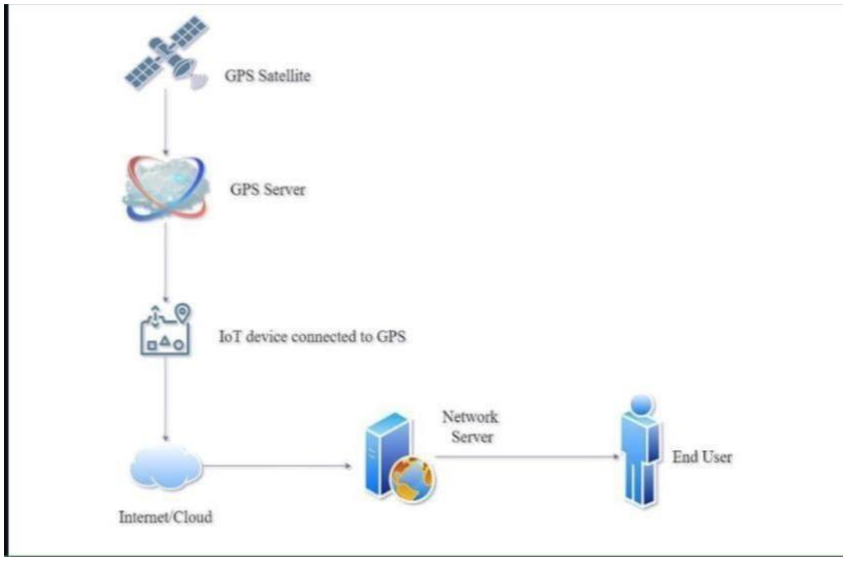
NFR-2	**Security**	To trigger the alarm and enable automatic video recording whenever the emergency button is pressed.
NFR-3	**Reliability**	Enable sending of notification, if the child is out of location or when the device realizes abnormal condition or situations
NFR-4	**Performance**	When a child is facing an emergency situation, device button should be pressed so that the device captures the image along with the user information to the enrolled mobile numbers
NFR-5	**Availability**	Child monitor, audio monitor, location monitor, video monitor
NFR-6	**Scalability**	If problem arises parents can see all the features like location, temperature, heart beat of the child along with live view around the children without hindrance

5.PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS



5.2 SOLUTION & TECHNICAL ARCHITECTURE



Outline Architecture:

6. PROJECT PLANNING AND SCHEDULING

6.1. SPRINT PLANNING& ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register through the form by Filling in my details	2	High	Suvalasini
Sprint-1		USN-2	As a user, I can register through phone numbers, Gmail, Facebook or other social sites	1	High	Shanmuga Priya
Sprint-1	Conformation	USN-3	As a user, I will receive confirmation through email or OTP once registration is successful	2	Low	Hari priya
Sprint-1	login	USN-4	As a user, I can login via login id and password or through OTP received on register phone number	2	Medium	Elavarasi
Sprint-1	Display Train details	USN-5	As a user, I can enter the start and destination to get the list of trains available connecting the above	1	High	Suvalasini
Sprint-2	Booking	USN-6	As a use, I can provide the basic details such as a name, age, gender etc…	2	High	Shanmuga Priya
Sprint-2		USN-7	As a user, I can choose the class, seat/berth. If a preferred seat/berth isn't available I can be allocated based on the availability	1	Low	Hari Priya
Sprint-2	Payment	USN-8	As a user, I can choose to pay through credit Card/debit card/UPI.	1	High	Elavarasi
Sprint-2		USN-9	As a user, I will be redirected to the selected	2	High	Suvalasini

Sprint-3	Ticket generation	USN-10	As a user, I can download the generated e- ticket for my journey along with the QR code which is used for authentication during my journey.	1	High	Shanmuga Priya
Sprint-3	Ticket status	USN-11	As a user, I can see the status of my ticket	2	High	Hari Priya
			Whether it's confirmed/waiting/RAC.			
Sprint-3	Remainders notification	USN-12	As a user, I get remainders about my journey A day before my actual journey.	1	High	Elavarasi

Sprint-3	Ticket cancellation	USN-13	As a user, I can track the train using GPS and can get information such as ETA, Current stop and delay	2	High	Suvalasini
Sprint-4		USN-14	As a user, I can cancel my tickets if there's any Change of plan	1	High	Shanmuga Priya
Sprint-4	Raise queries	USN-15	As a user, I can raise queries through the query box or via mail.	2	Medium	Hari Priya
Sprint-4	Answer the queries	USN-16	As a user, I will answer the questions/doubts Raised by the customers.	2	High	Elavarasi
Sprint-4	Feed details	USN-17	As a user, I will feed information about the trains delays and add extra seats if a new compartment is added.	1	High	Suvalasini

7.CODING AND SOLUTIONING

7.1. FEATURE 1

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

7.2. FEATURE 2

- Registration
- Login
- Verification
- Adding Queries

7.3. DATABASE SCHEMA

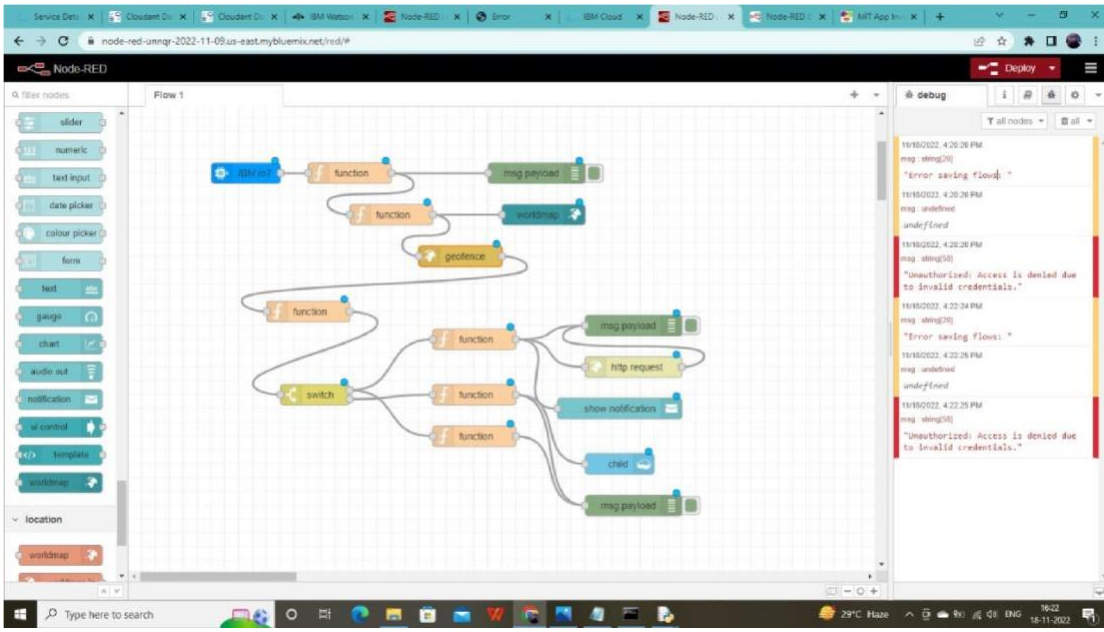
```
labl_0 = Label(base, text="Registration form",width=20,font=("bold",
20))    labl_0.place(x=90,y=53)


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,
```

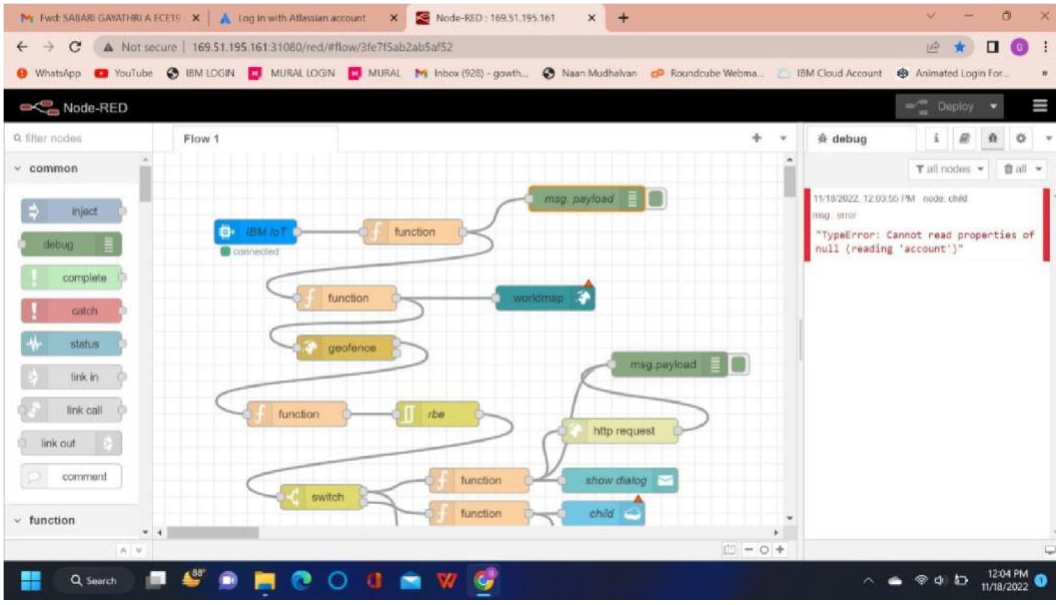
```
text="Select    Country",         width=13,font=("arial",12))
lb2.place(x=14,y=280)
drplist.place(x=200, y=275)

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") roo
```

8.TESTING

8.1.TEST CASES


Node-RED
Deploy
Flow 1
debug
IBM IoT
function
msg.payload
worldmap
geofence
switch
http request
show notification
child
location


Node-RED
Deploy
Flow 1
debug
common
inject
debug
complete
catch
status
link in
link call
link out
comment
function
IBM IoT
connected
msg.payload
worldmap
geofence
rbe
http request
show dialog
switch
child
"TypeError: Cannot read properties of null (reading 'account')"

9.RESULTS

9.1.PERFORMANCE METRICS

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.

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.

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

13.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/trains-between-
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
```

```
              last_name = models.CharField(
```

```
verbose_name="Last name",        max_length=40
   )

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

   stripe_id = models.CharField(


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-
        clonespecialrgdto-ndls"  # url  url =
        "https://www.railyatri.in/livetrainstatus/"+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:

                     Speak("Starting Sir")

                    if "no" in question:
    Sir.")           Speak("We will automatically start after 5 Mins

                     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",
    mins",          message="will tell you to take a break after 45

                     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_ofac1stclass=0
self.totaf=0
self.no_ofac2ndclass=0
self.no_ofac3rdclass=0        self.no_ofsleeper=0
self.no_oftickets=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.dat","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_oftickets                  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
         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          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\n\n\n\n\n\n\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:
          j="*****"
r=raw_input("\n\n\n\n\n\n\n\n\n\n\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\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:
          print "ERROR"
else:          try:          while  True:
          print"*"*80          print"\t\t\t\tTRAIN
DETAILS"
```

```
                print"*"*80                print
        tr=load(fin)                tr.output()



                raw_input("PRESS ENTER TO VIEW NEXT TRAIN
        DETAILS")
                os.system('cls')            except
        EOFError:
                pass
        elif ch==3:             print'='*80
                print "\t\t\t\tRESERVATION OF TICKETS"
        print'='*80             print           tick.reservation()              elif
        ch==4:
                print"="*80
        print"\t\t\t\tCANCELLATION OF TICKETS"
        print           print"="*80          print           tick.cancellation()
        elif ch==5:             print
        "="*80
        print("PNR STATUS".center(80))
                print"="*80            printclass
        tickets:    def __init__(self):
        self.no_ofac1stclass=0      self.totaf=0
        self.no_ofac2ndclass=0
        self.no_ofac3rdclass=0      self.no_ofsleeper=0
        self.no_oftickets=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.dat","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_oftickets                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
         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            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\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:
j="*****"
r=raw_input("\n\n\n\n\ n\n\n\n\n\n\n\t\t\t\tENT
ER 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:
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>
      <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:") # 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(
"Content-Disposition",
```

```
f"attachment; filename= {filename}",
)
# Add attachment to message and convert message to string
message.attach(part)     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: #
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"] # 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  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")
```

13.2.GIT HUB LINK

https://github.com/IBM-EPBL/IBM-project-23538-1659885448