Smart Solutions for Railways

Category: *Internet of Things*

PROJECT REPORT

SUBMITTED BY

Team ID: PNT2022TMID42986

NAME	REGISTER NUMBER
1. Karthika Rani A	712819106901
2. Arun Manikandan R	712819106002
3. Arun Kumar A	712819106301
4. Jeevika	712819106006

INPARTIAL FULFILLMENT FOR THE

AWARD OF THE DEGREE

Of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

RVS COLLEGE OF ENGINEERING AND TECHNOLOGY

COIMBATORE-641402

ANNA UNIVERSITY: CHENNAI - 600025

Project Report Format

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

As trains are one of the most preferred modes of transportation among middle class and impoverished people as it attracts for its amenities. Simultaneously there is an increase at risk from thefts and accidents like chain snatching, derailment, fire accident. In order to avoid or in better words to stop all such brutality we came up with a solution by providing an application which can be accessed by the user after booking their tickets. With a single click this app addresses issues by sending a text message to TC and RPF as an alert. In our project we use Node-Red service, app-development, IBM cloud platform to store passenger data. **1.2 Purpose**

The purpose of this project is to report and get relived from the issues related to trains.

2. LITERATURE SURVEY

2.1 Existing problem

- A Web page is designed for the public where they can book tickets by seeing the available seats.
- After booking the train, the person will get a QR code which has to be shown to the Ticket Collector while boarding the train.
- The ticket collectors can scan the QR code to identify the personal details.
- A GPS module is present in the train to track it. The live status of the journey is updated in the Web app continuously
- All the booking details of the customers will be stored in the database with a unique ID and they can be retrieved back when the Ticket Collector scans the QR Code.

2.2 References

S.NO	TITLE	AUTHOR	YEAR	KEY TECHNOLOGY
1	Main geotechnical problems of railways and roads in kriolitozone and their solutions.	Kondratiev, Valentin G	2017	Main problems in railways

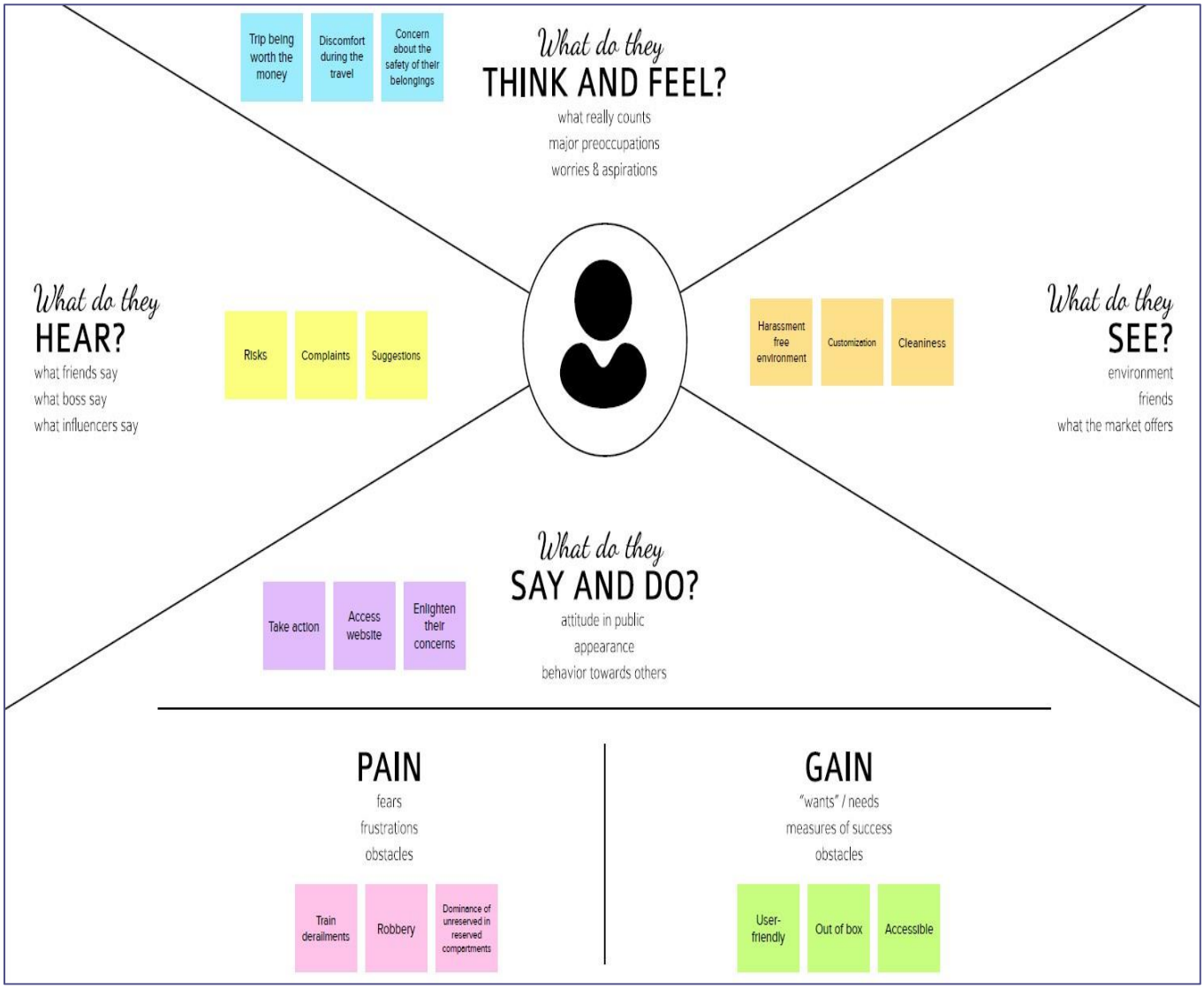
2	Construction and Building Materials	Sañudo, Roberto, Marina Miranda, Carlos García, and David García-Sanchez	2019	Drainage in railways
3	Problems of Indian Railways	Benjamin	2021	Common problems in Indian railways
4	A comparative study of Indian and worldwide railways.	Sharma, Sunil Kumar, and Anil Kumar	2014	Study of Indian railways
5	Ticketing solutions for Indian railways using RFID technology	Prasanth,Venugopal, and K.P. Soman	2009	Solution for ticketing using RFID

2.3 Problem Statement Definition

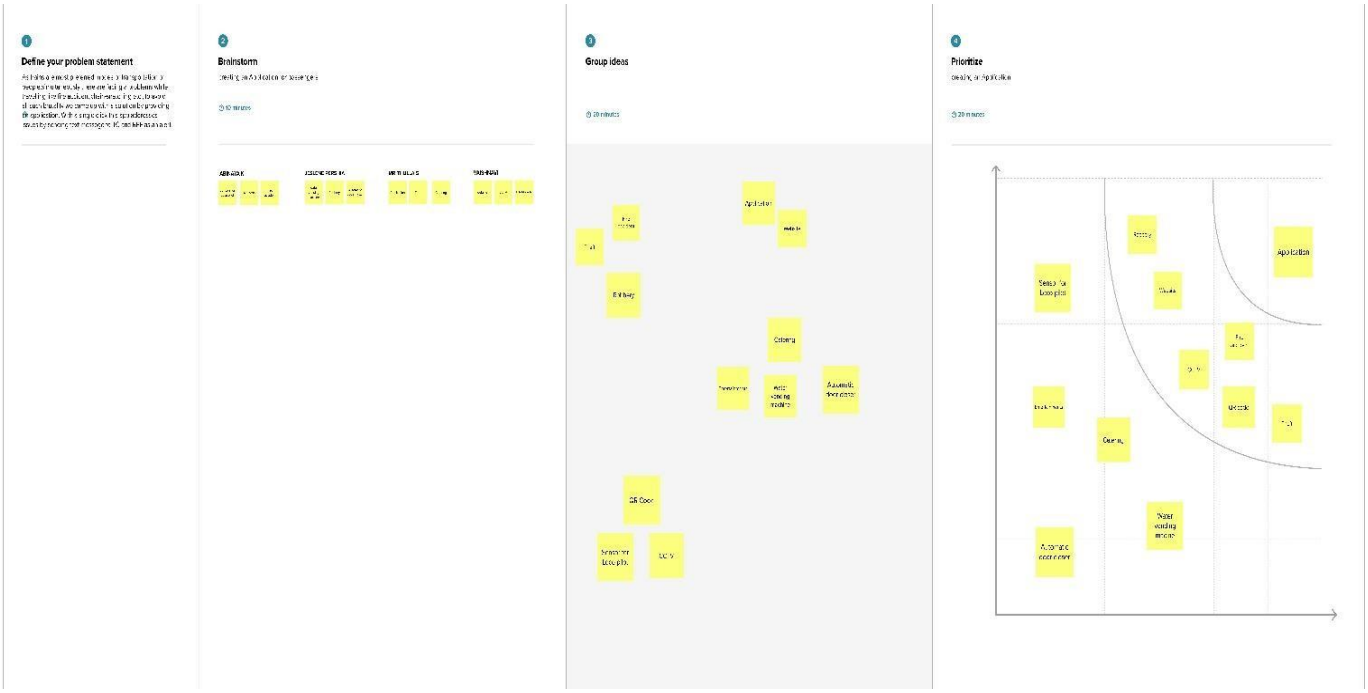
Smart Solutions for railways are designed to reduce the work load of the user and the use of paper.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

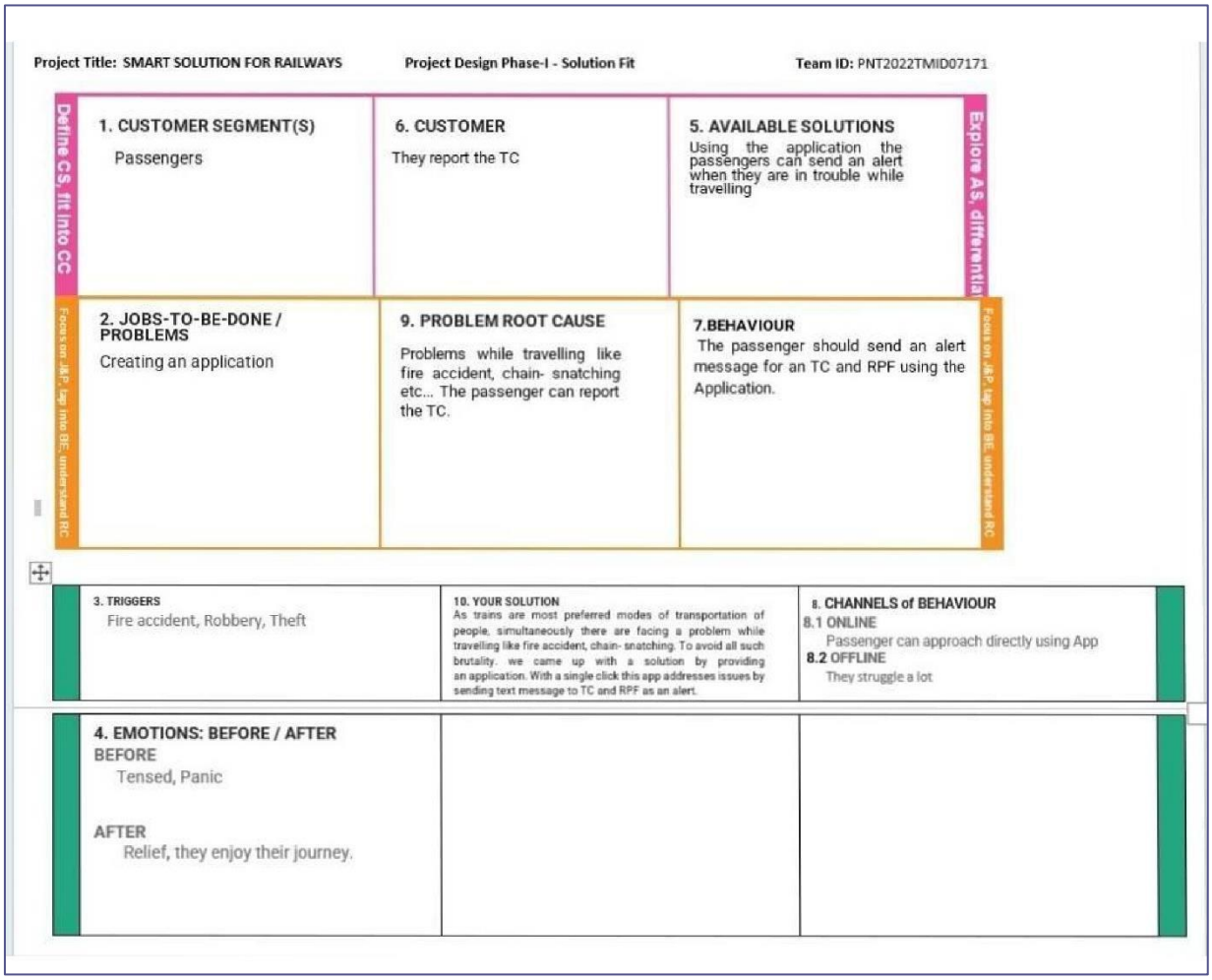


3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Problems in the railways like robbery, fire accidents etc..
2.	Idea / Solution description	Developing an app for the passengers.
3.	Novelty / Uniqueness	The passengers can send an alert to the respective officials during the travel time through the app when they are in trouble so that they can easily solve it.
4.	Social Impact / Customer Satisfaction	Usage of this app can be a great relief to the passengers, so that they can travel without any fear.
5.	Business Model (Revenue Model)	5000

6.	Scalability of the Solution	This solution will be useful for passengers while travelling. They can use the app between the time of their travel. The users will fell more secured, in-case of an emergency by simply clicking on a button the alert signal will be sent to the respective officials and the corresponding measures will be taken.

3.4 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

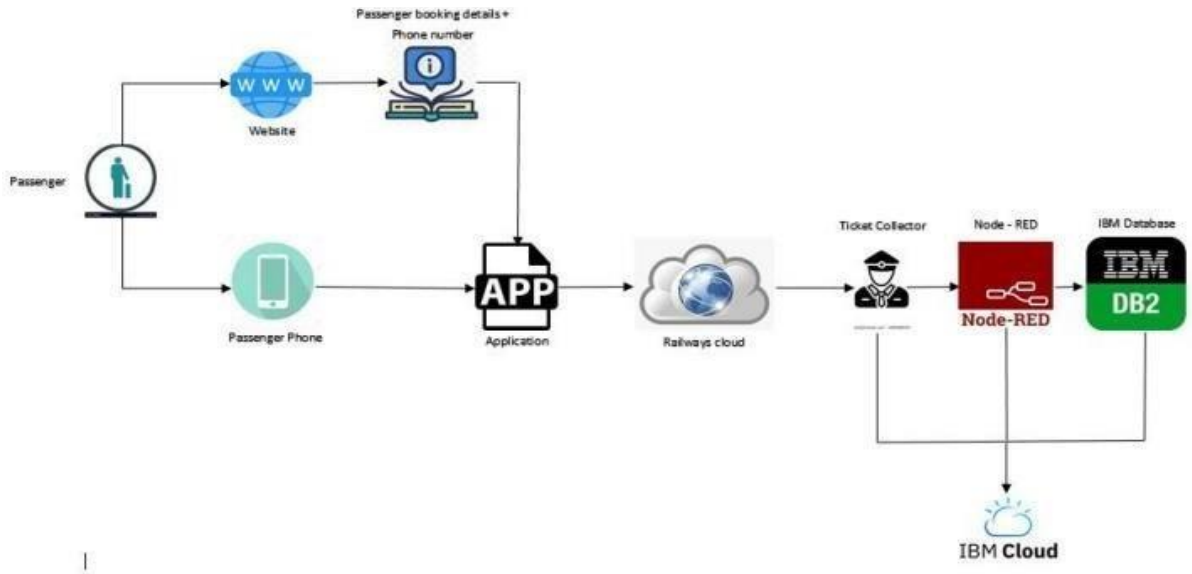
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Online Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Application installation	The application is installed through the given link
FR-4	User access	Access the app requirements

4.2 Non-Functional requirement

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	• The app can be used during the travelling time • Easy and simple • Efficiency is high
NFR-2	**Security**	By clicking on the icon, the alert will be given to the respective officials
NFR-3	**Reliability**	Highly reliable to use
NFR-4	**Performance**	Low error rate
NFR-5	**Availability**	Free source
NFR-6	**Scalability**	It is scalable enough to support many users at the same time

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution Architecture

As trains are one of the most preferred modes of transportation among middle class and impoverished people as it attracts for its amenities. Simultaneously there is an increase at risk from thefts and accidents like chain-snatching, derailment, fire accident. In order to avoid or in better words to stop all such brutality we came up with a solution by providing an application which can be accessed by the user after booking their tickets. With a single click this app addresses issues by sending a text message to TC and RPF as an alert. In our project we use Node-Red service, app-development, IBM cloud platform to store passenger data.

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
PASSENGER (Mobile user)	Booking registration	USN-1	As a passenger, I book the ticket for the journey by entering my personal information.	I can access the web link to install the application.	High	Sprint-1
	Confirmation	USN-2	As a passenger, I will receive confirmation of the booking once I have registered for the application	I can receive confirmation email & click confirm.	High	Sprint-1

	Applicat ion registrat ion	USN-3	As a passenger, I can register for the application through the weblink.	I can register & access the application through google login.	Low	Sprint-2
	Application access	USN-4	As a passenger, I can access the application during my travel for resolving my issues.		Medium	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

STEP 1	Identify the problem
STEP 2	Prepare an abstract, problem statement
STEP 3	List required objects needed
STEP 4	Create a code and run it
STEP 5	Make a prototype

STEP 6	Test with the created code and check the designed prototype is working
STEP 7	Solution for the problem is found

6.2 Reports from JIRA

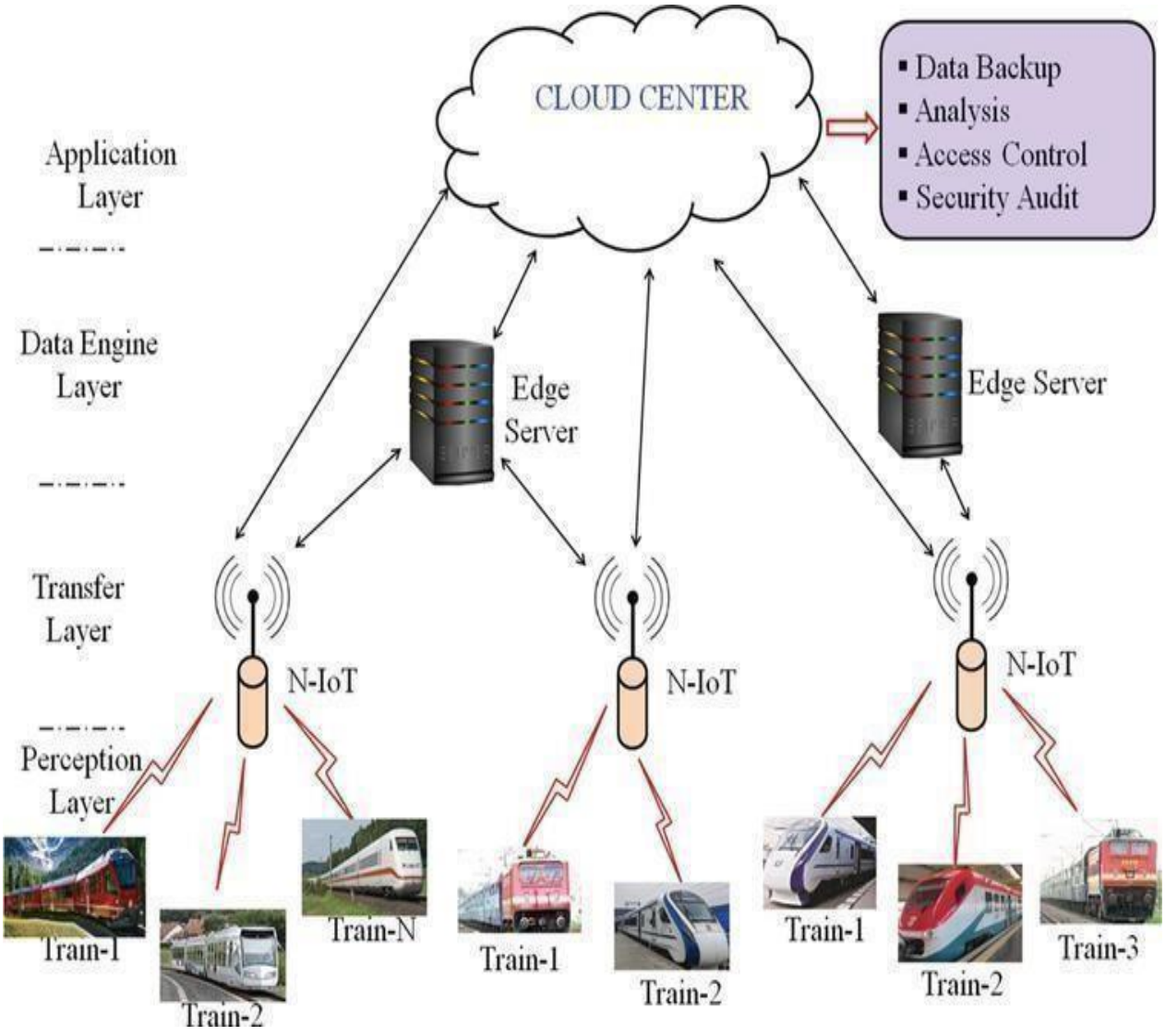
SPRINT 1

```
#include <LiquidCrystal.h>
LiquidCrystal 1cd(5,6,8,9,10,11);   int red1ed = 2; int green1ed = 3; int buzzer
= 4; int sensor = A0;
int sensorThresh = 400;


void setup()
{
pinMode(red1ed,          OUTPUT);          pinMode(green1ed,OUTPUT);
pinMode(buzzer,OUTPUT); pinMode(sensor,INPUT); serial.begin(9600);
1cd.begin(16,2);
}


Void loop()
{
```

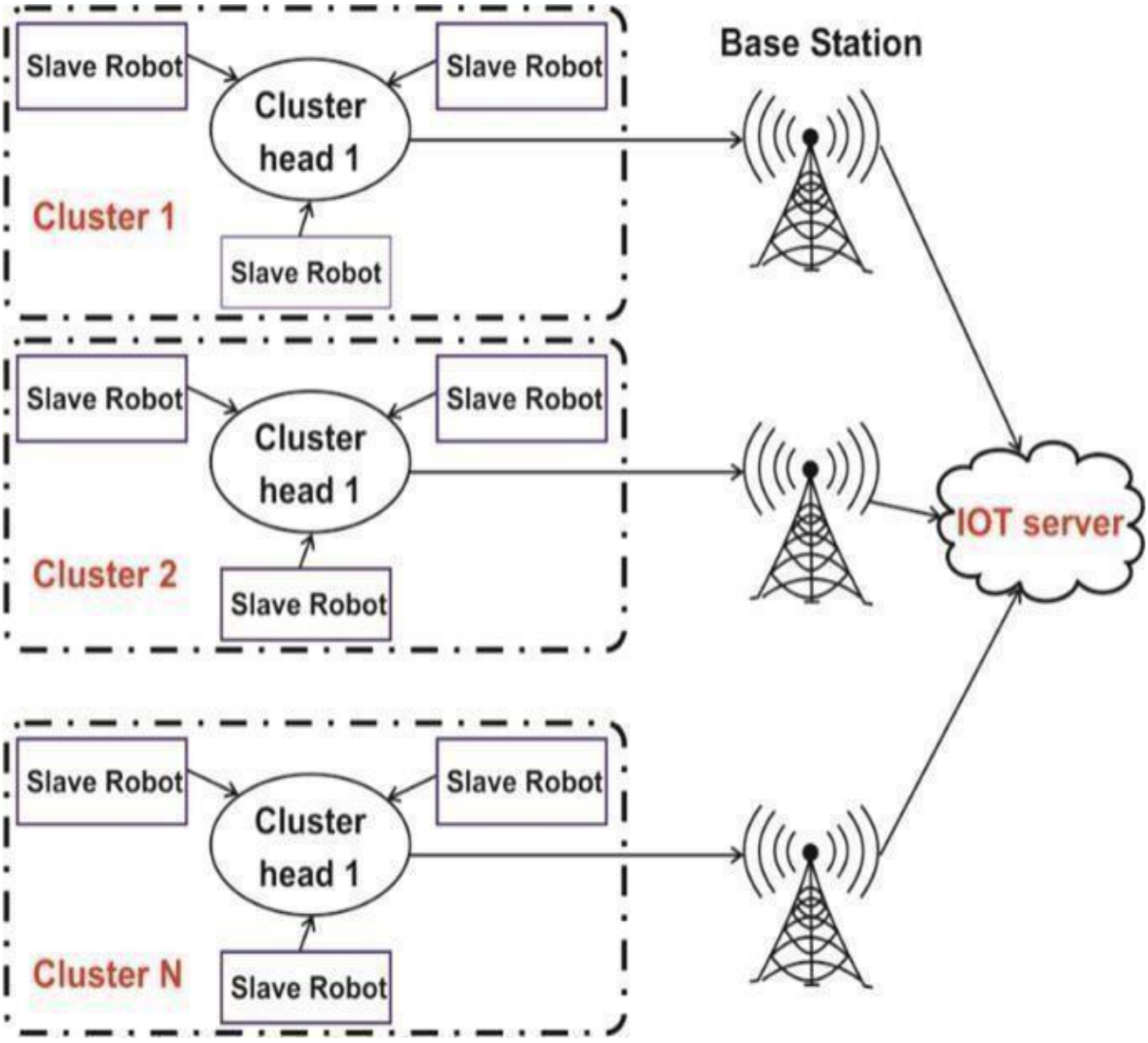
```
int analogValue = analogRead(sensor); Serial.print(analogvalue);
if(analogValue>sensorThresh)
{

  digitalWrite(red1ed,HIGH);          digit1Weite(green1ed,LOW);
  tone(buzzer,1000,10000);
  1cd.clear();
  1cd.setCursor(0,1);
  1cd.print("RAILWAYS"); delay(1000);
  1cd.clear();
  1cd.setCursor(0,1);
  1cd.print("SMART SOLUTION"); delay(1000);
}

else
{

    digitalWrite(greenlad,HIGH);          digitalWrite(red1ed,LOW);
    noTone(buzzer);  1cd.clear();  1cd.setCursor(0,0);  1cd.print("SAFE");
    delay(1000);
    1cd.clear();
    1cd.setCursor(0,1);
                                          1cd.print("ALL CLEAR"); delay(1000);
  }
```

SPRINT 2

Proposed architecture for smart track monitoring system.



Structural health monitoring of railway tracks using IOT-based multi robot system:



Main Program:

```
importwiotp.sdk.device
importtime
importrandom
myConfig={
"identity":{
"orgId":"gagtey",
"typeId":"GPS",
"deviceId":"12345"
```

```
},
"auth":{
"token":"12345678"
}
}
defmyCommandcallback(cmd):
print("messagereceivedfromIBMIOTPlatform:%s"%cmd.data['command'])
m=cmd.data['command']
client=wiotp.sdk.device.deviceclient(config=myConfig,logHandlers=None)
client.connect()
defpub(data):
client.publishEvent(eventId="status",msgFormat="json",data=mydata,qos=0,
print("publishedatasuccessfully:%s",mydata)
whileTrue:
mydata={'name':'Train1','lat':17.6387448,'lon':78.4754336)
pub(myData)
time.sleep(3)
#mydata={'name':'Train2','lat':17.6387448,'lon':78.4754336)
#pub(myData)
#time.sleep(3)
mydata={'name':'Train1','lat':17.6341908,'lon':78.4744722)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6340889,'lon':78.4745052)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6248626,'lon':78.4720259)
pub(myData)
        time.sleep(3)
mydata={'name':'Train1','lat':17.6188577,'lon':78.4698726)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6132382,'lon':78.4707318)
pub(myData)
time.sleep(3)
client.commandCallback=mycommanCallbak
client.disconnect()
```

Code:

```
importcv2
importnumpyasnp
importtime
importpyzbar.pyzbaraspuzbar
```

```
fromibmcloudant.cloudant_v1importcloudantv1
fromibmcloudantimportcouchDbsessionAuthenticator
fromibm_cloud_sdk_core.AuthenticatorsimportBasicAuhtenticator
authenticator=BasicAuthenticator('apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz','b0ab119f45d3e6255eabb978)
service=cloudantv1(authenticator=authenticator)
service.set_service_url('https://apikey-v2-16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz:b0ab119f45d3e6255eabb978
cap=cv2.videoCapture(0)
font=cv 2.FONT_HERSHEY_PLAIN
whileTrue:
,frame=cap.read(0)
decode  bjects=pyzbar.decode(frame)
forobjindecodeObjects:
#print("Data",obj.data)
a=obj.data.decode('UTF-8')
cv2.putText(frame,"Ticket",(50,50),font,2,(255,0,0),3)
#print(a)
try:
responce=service.get_document(db='booking',doc_id=a).get_result()
print(response)
time.sleep(5)
exceptExceptionase:
print("NotvalidTicket")
time.sleep(5)
cap.imshow("Frame",frame)
ifcv2.waitKey{1}&0XFF==ord('q'):
break
cap.release()
cv2.destroyAllWindows()
client.disconnect()
```

SPRINT 3

- This project presents its first ever digital event dedicated to rail transport, the “Smart Mobility Experience” which will take place on March 24th. This event will be the occasion for clients and partners of the rail ecosystem, to discover new products and major innovations, as well as to exchange about the digitalization and future of rail.
- for improved service performance and energy efficiency, and to boost the attractiveness for users.
- It helps transporting passengers safely, and with best possible experience, supervises operations with accurate situation awareness, and optimizes transport service efficiency.
- Using digital technologies such as IoT, cloud and web IT, data analytics , it designs innovative solutions such as digital signalling, train autonomy, mobile ticketing, passenger flow analytics, data driven operation control, smart maintenance, which will drastically impact the way we all travel.
- Provide real-time passenger density insights to public transport operators
- The solution helps alleviate crowding by reducing busy times, and consequently enhances overall passenger safety, comfort, and travel experience.
- The targeted performances of density accuracy are above 90%.

In Hand’s Connectivity Solution for Rail Transit:

MAIN:

```
importwiotp.sdk.device
importtime
importrandom
myConfig={
"identity":{
"orgId":"gagtey",
        "typeId":"GPS",
"deviceId":"12345"
},
"auth":{
"token":"12345678"
}
}
defmyCommandcallback(cmd):
print("messagereceivedfromIBMIOTPlatform:%s"%cmd.data['command'])
m=cmd.data['command']
client=wiotp.sdk.device.deviceclient(config=myConfig,logHandlers=None)
client.connect()
defpub(data):
client.publishEvent(eventId="status",msgFormat="json",data=mydata,qos=0,
print("publishedatasuccessfully:%s",mydata)
whileTrue:
mydata={'name':'Train1','lat':17.6387448,'lon':78.4754336)
pub(myData)
time.sleep(3)
#mydata={'name':'Train2','lat':17.6387448,'lon':78.4754336)
#pub(myData)
#time.sleep(3)
mydata={'name':'Train1','lat':17.6341908,'lon':78.4744722)
pub(myData)
```

```
time.sleep(3)
mydata={'name':'Train1','lat':17.6340889,'lon':78.4745052)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6248626,'lon':78.4720259)
pub(myData)
        time.sleep(3)
mydata={'name':'Train1','lat':17.6188577,'lon':78.4698726)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6132382,'lon':78.4707318)
pub(myData)
time.sleep(3)
client.commandCallback=mycommanCallbak
client.disconnect()
```

PROGRAM:

```
importcv2
importnumpyasnp
importtime
importpyzbar.pyzbaraspuzbar
fromibmcloudant.cloudant_v1importcloudantv1
fromibmcloudantimportcouchDbsessionAuthenticator
fromibm_cloud_sdk_core.AuthenticatorsimportBasicAuhtenticator
authenticator=BasicAuthenticator('apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz','b0ab119f45d3e6255eabb978)
service=cloudantv1(authenticator=authenticator)
service.set_service_url('https://apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz:b0ab119f45d3e6255eabb978
cap=cv2.videoCapture(0)
font=cv2.FONT_HERSHEY_PLAIN
```

```
whileTrue:
_,frame=cap.read(0)
decodeObjects=pyzbar.decode(frame)
forobjindecodeObjects:
#print("Data",obj.data)
a=obj.data.decode('UTF-8')
cv2.putText(frame,"Ticket",(50,50),font,2,(255,0,0),3)
#print(a)
try:
responce=service.get_document(db='booking',doc_id=a).get_result()
print(response)
time.sleep(5)
exceptExceptionase:
print("NotvalidTicket")
time.sleep(5)
cap.imshow("Frame",frame)
ifcv2.waitKey{1}&0XFF==ord('q'):
break
cap.release()
cv2.destroyAllWindows()
client.disconnect()
```

SPRINT 4

Main:

```
importwiotp.sdk.device
importtime
importrandom
myConfig={
"identity":{
"orgId":"gagtey",
"typeId":"GPS",
```

```
"deviceId":"12345"
},
"auth":{
"token":"12345678"
}
}
defmyCommandcallback(cmd):
print("messagereceivedfromIBMIOTPlatform:%s"%cmd.data['command'])
m=cmd.data['command']
client=wiotp.sdk.device.deviceclient(config=myConfig,logHandlers=None)
client.connect()
defpub(data):
client.publishEvent(eventId="status",msgFormat="json",data=mydata,qos=0,
print("publishedatasuccessfully:%s",mydata)
whileTrue:
mydata={'name':'Train1','lat':17.6387448,'lon':78.4754336)
pub(myData)
time.sleep(3)
#mydata={'name':'Train2','lat':17.6387448,'lon':78.4754336)
#pub(myData)
#time.sleep(3)
mydata={'name':'Train1','lat':17.6341908,'lon':78.4744722)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6340889,'lon':78.4745052)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6248626,'lon':78.4720259)
pub(myData)
time.sleep(3)
mydata={'name':'Train1','lat':17.6188577,'lon':78.4698726)
pub(myData)
```

```
time.sleep(3)
mydata={'name':'Train1','lat':17.6132382,'lon':78.4707318)
pub(myData)
time.sleep(3)
client.commandCallback=mycommanCallbak
    client.disconnect()
```

Program:

```
importcv2
importnumpyasnp
importtime
importpyzbar.pyzbaraspuzbar
fromibmcloudant.cloudant_v1importcloudantv1
fromibmcloudantimportcouchDbsessionAuthenticator
fromibm_cloud_sdk_core.AuthenticatorsimportBasicAuhtenticator
authenticator=BasicAuthenticator('apikey-v2-16u3cr
mdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz','b0ab119f45d3e6255eabb978)
service=cloudantv1(authenticator=authenticator)
service.set_service_url('https://apikey-v2-
16u3crmdpkghhxefdikvpssoh5fwezrmuup5fv5g3ubz:b0ab119f45d3e6255eabb978
cap=cv2.videoCapture(0)
font=cv2.FONT_HERSHEY_PLAIN
whileTrue:
_,frame=cap.read(0)
decodeObjects=pyzbar.decode(frame)
forobjindecodeObjects:
```

7. CODING & SOLUTIONING

7.1 Feature 1

- IoT device
- IBM Watson Platform
- Node red
- Cloudant DB

- Web UI
- MIT App Inventor
- Python code

7.2 Feature 2

- Login
- Verification
- Ticket Booking
- Adding rating

8. TESTING AND RESULTS

8.1 Test Cases

STEP 1	Identify the problem
STEP 2	Prepare an abstract, problem statement
STEP 3	List required objects needed
STEP 4	Create a code and run it
STEP 5	Make a prototype

STEP 6	Test with the created code and check the designed prototype is working
STEP 7	Solution for the problem is found

9. ADVANTAGES

- The passengers can use this application, while they are travelling alone to ensure their safety.
- It is easy to use.
- It has minimized error rate.

10. DISADVANTAGES

 Network issues may arise.

11. CONCLUSION

Almost all the countries across the globe strive to meet the demand for safe, fast, and reliable rail services. Lack of operational efficiency and reliability, safety, and security issues, besides aging railway systems and practices are haunting various countries to bring about a change in their existing rail infrastructure. The global rail industry struggles to meet the increasing demand for freight and passenger transportation due to lack of optimized use of rail network and inefficient use of rail assets. Often, they suffer from the lack in smart technologies and latest technological updates to provide the most efficient passenger services. This is expected to induce rail executives to build rail systems that are smarter and more efficient. The passenger reservation system of Indian Railways is one of the world's largest reservation models. Daily about one million passengers travel in reserved accommodation with Indian Railways. Another sixteen million travel with unreserved tickets in Indian Railways. In this vast system, it is a herculean task to efficiently handle the passenger data, which is a key point of consideration now-a-days. But the implementation of the latest technological updates in this system gradually turns inevitable due to increasing demand for providing the most efficient passenger services. Handling the passenger data efficiently backed by intelligent processing and timely retrieval would help backing up the security breaches. Here we've explored different issues of implementing smart computing in railway systems pertaining to reservation models besides pointing out some future scopes of

advancement. Most significant improvements have been evidenced by more informative and user-friendly websites, mobile applications for real-time information about vehicles in motion, and e-ticket purchases and timetable information implemented at stations and stops. With the rise of Industry, railway companies can now ensure that they are prepared to avoid the surprise of equipment downtime. Like above mentioned, the developed application of our project can lead the passenger who travel can travel safely without any fear.

12. FUTURE SCOPE

This application is ensured for safety for the passengers while they are travelling alone as well as they travel with their family or friends.

In future, this application may also be used by passengers who travel through bus. By further enhancement of the application the passengers can explore more features regarding their safety.

13. APPENDIX

13.1 Source Code

LOGIN

```
from tkinter import *
import sqlite3

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

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

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

```
Form.pack(side=TOP, pady=20)

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

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




#==============================METHODS============================
============= def
Database():
   global conn, cursor
   conn = sqlite3.connect("pythontut.db")
cursor = conn.cursor()
   cursor.execute("CREATE TABLE IF NOT EXISTS `member` (mem_id INTEGER NOT
NULL PRIMARY KEY AUTOINCREMENT, username TEXT, password TEXT)")
cursor.execute("SELECT * FROM `member` WHERE `username` = 'admin' AND
`password` = 'admin'") if
cursor.fetchone() is None:
      cursor.execute("INSERT INTO `member` (username, password) VALUES('admin',
'admin')") conn.commit() def Login(event=None):
Database() if USERNAME.get() == "" or
PASSWORD.get() == "":
      lbl_text.config(text="Please complete the required field!", fg="red")
else:
      cursor.execute("SELECT * FROM `member` WHERE `username` = ? AND `password`
```

```
= ?", (USERNAME.get(), PASSWORD.get()))
if cursor.fetchone() is not None:
        HomeWindow()
        USERNAME.set("")
PASSWORD.set("")
lbl_text.config(text="")        else:
        lbl_text.config(text="Invalid username or password", fg="red")
        USERNAME.set("")
PASSWORD.set("")
   cursor.close()
   conn.close()


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


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

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

REGISTRATION

```
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()
```

START AND DESTINATION

```
# import module import
requests
from bs4 import BeautifulSoup

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


# input by geek from_Station_code
= "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_Station_name+"+JN+&j
ourney_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)
```

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 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 CONFIRMATION
# import module import
requests from bs4 import
BeautifulSoup import pandas
as pd

# 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-clone-special-rgd-to-ndls"

# url
url = "https://www.railyatri.in/live-train-status/"+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'])
```

TICKET GENERATION

```
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
```

OTP GENERATION

```
import         os
import     math
import random
import smtplib

digits = "0123456789"
OTP = ""

for i in range (6):
   OTP += digits[math.floor(random.random()*10)]

otp = OTP + " is your OTP" message
= otp
s = smtplib.SMTP('smtp.gmail.com', 587)
s.starttls()

emailid = input("Enter your email: ")
s.login("YOUR Gmail ID", "YOUR APP PASSWORD")
s.sendmail('&&&&&&',emailid,message)

a = input("Enter your OTP >>: ") if
a == OTP:
   print("Verified") else:
   print("Please Check your OTP again")
```

OTP VERIFICATION

```
import         os
import     math
import random
```

```
import smtplib

digits = "0123456789"
OTP = ""

for i in range (6):
    OTP += digits[math.floor(random.random()*10)]

otp = OTP + " is your OTP" message
= otp
s = smtplib.SMTP('smtp.gmail.com', 587)
s.starttls()

emailid = input("Enter your email: ")
s.login("YOUR Gmail ID", "YOUR APP PASSWORD")
s.sendmail('&&&&&&',emailid,message)

a = input("Enter your OTP >>: ") if
a == OTP:
    print("Verified") else:
    print("Please Check your OTP again")
```

13.2 GitHub

GitHub link:

https://drive.google.com/file/d/1teCIAFXZDsEqv_sa3DQGXl35UCo2CZar/view?usp=drivesdk