

# **PROJECT REPORT**

Project Name: **SMART SOLUTIONS FOR RAILWAYS**

Team ID: **PNT2022TMID44729**

Team: **MANGAIYARKARASI.E (Team leader)**

Team Member: **JAI KRISHNAN.A.P**

Team Member: **RAHUL.M**

Team Member: **SHANMUGAPRIYA.A**

# **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 relieved 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	A REVIEW ON SECURITY AS A SAFETY ISSUE IN RAIL COMMUNICATION AT INFORMATION SECURITY RESEARCH CENTRE QUEENSLAND UNIVERSITY OF TECHNOLOGY.	J.SMITH, S.RUSSELL AND M.LOOI	2003	Systems whose failure can lead to the damage of property or the environment, or loss of human life are regarded as safety-critical systems. It is no longer adequate to build safety-critical systems based on the control of errors and failures alone.
2	“A SECURE RAILWAY CROSSING SYSTEM USING IOT”, INTERNATIONAL CONFERENCE ON ELECTRONICS.	E.AMARNATH REDDY,I.KAVATI, K.SRINIVAS RAO AND G.KIRANKUMAR	2017	COMMUNICATION AND AEROSPACE TECHNOLOGY ICECA
3	“SAFEGUARD OF RAILWAY CROSSING USING IOT”, JOURNAL OF TELECOMMUNICATIONS SYSTEM & MANAGEMENT.	B.KUMAR REDDY,S.KUMAR REDDY,R.REDDY AND NAVYA	2018	The main prototype of this paper is to develop an application based on Internet of Things. In the present day world, railway gates at the crossings were monitored and operated manually. The master of the station gets the information about the arrival

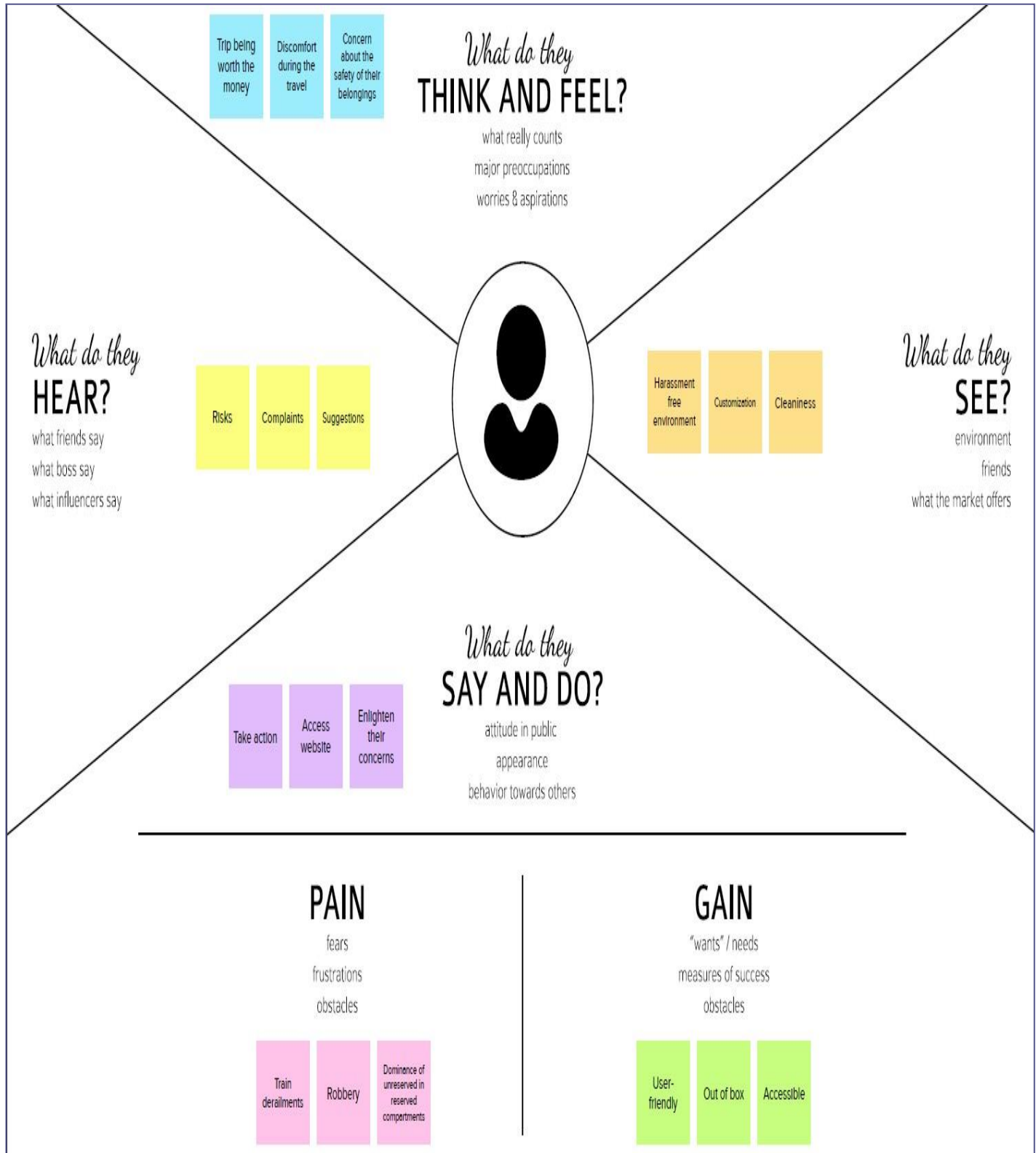
4	RAILROAD GRADE CROSSING MONITORING SYSTEM	E.GOOLSBY,M.J.VICKICH, A.P.VOIGT	2003	In North America, highway- railway grade crossings can lead to significant travel delays for emergency responders trying to reach an incident. Grade separation cannot be justified for most grade crossings, but a grade crossing monitoring system (GCMS) can detect a blockage and communicate the information to local emergency dispatchers in real-time.

### 2.3 Problem Statement Definition

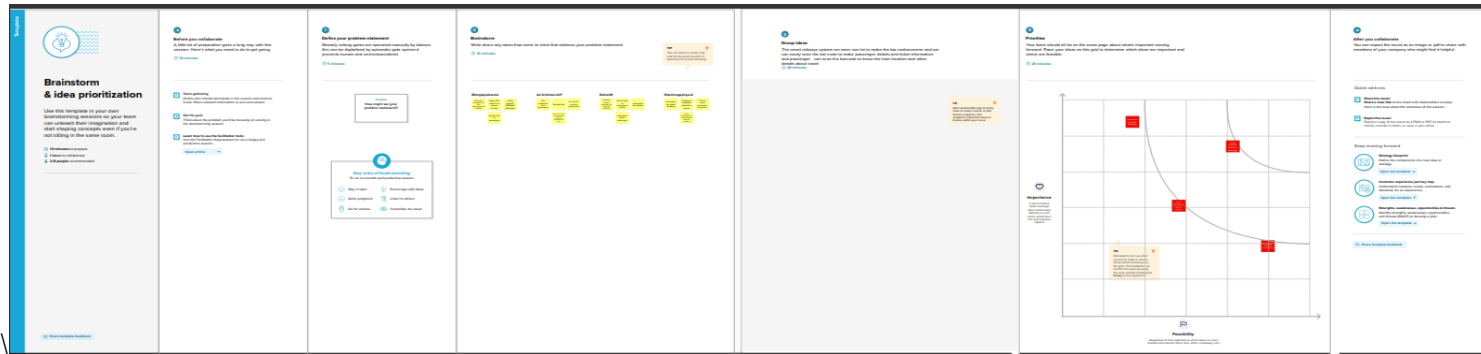
Smart Solutions for railways are designed to reduce the work load of the user and the use of paper. In current system there are many disadvantages which are to be rectified. The main thing which comes under is allocation of lower berths. Even for senior citizens, medically ill and pregnant ladies. during verification there could possibilities for fake identifications also. So there could possibility of unauthenticated travel by stranger also. More over the main disadvantage is about payment for waiting list passengers and un travelled passengers. They are not given any refund and those un travelled seats were sold out for the officers profit. Even in verification there is a lot's of quantity of paper is used.

### 3. IDEATION & PROPOSED SOLUTION

#### 3.1 Empathy Map Canvas



## 3.2 Ideation & Brainstorming



## 3.3 Proposed Solution

- For booking tickets for multiple passengers, It is necessary to enter have their Aadhar number for booking ticket.
- While in case of verification time, QR code should be verified for all the passengers who boarded in the train.
- If the passenger has a confirmed ticket then it will be verify, Otherwise it will check in the waiting list.
- For waiting list, not authentication means it will go refund if not no refund will be provided. For verification we are going to use a specifically designed mobile application .
- This application is accessed by using an external fingerprint sensor or with an inbuild sensor. each official will have individual login credentials so, by this itself we can identify every thing.
- Untravelled seats are automatically allocated for boarded waiting list passengers. More over with is application. Lower seats preference will be adjusted by the QR code and sensor citizenship claim also.

### 3.4 Problem Solution fit

Project Title: SMART SOLUTION FOR RAILWAYS			Project Design Phase-I - Solution Fit			Team ID: PNT2022TMID07171		
Define CS, fit into CC Focus on JBP, map into BC, understand RC	1. CUSTOMER SEGMENT(S) Passengers	6. CUSTOMER They report the TC	5. AVAILABLE SOLUTIONS Using the application the passengers can send an alert when they are in trouble while travelling	Explore AS, differentia Focus on JBP, map into BC, understand RC				
	2. JOBS-TO-BE-DONE / PROBLEMS Creating an application	9. PROBLEM ROOT CAUSE Problems while travelling like fire accident, chain- snatching etc... The passenger can report the TC.	7.BEHAVIOUR The passenger should send an alert message for an TC and RPF using the Application.					
3. TRIGGERS Fire accident, Robbery, Theft		10. YOUR SOLUTION As trains are most preferred modes of transportation of people, simultaneously there are facing a problem while travelling like fire accident, chain- snatching. To avoid all such brutality, we came up with a solution by providing an application. With a single click this app addresses issues by sending text message to TC and RPF as an alert.		8. CHANNELS of BEHAVIOUR 8.1 ONLINE Passenger can approach directly using App 8.2 OFFLINE They struggle a lot				
4. EMOTIONS: BEFORE / AFTER BEFORE Tensed, Panic  AFTER Relief, they enjoy their journey.								

## 4. REQUIREMENT ANALYSIS

### 4.1 Functional requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through online in any website.
FR-2	User Confirmation	Conformation notification OTP will send through email or other message.
FR-3	User QR code generation	QR code is generated after the confirmation.
FR-4	GPS tracking	Passenger can easily find the location through GPS tracker.

### 4.2 Non-Functional requirement

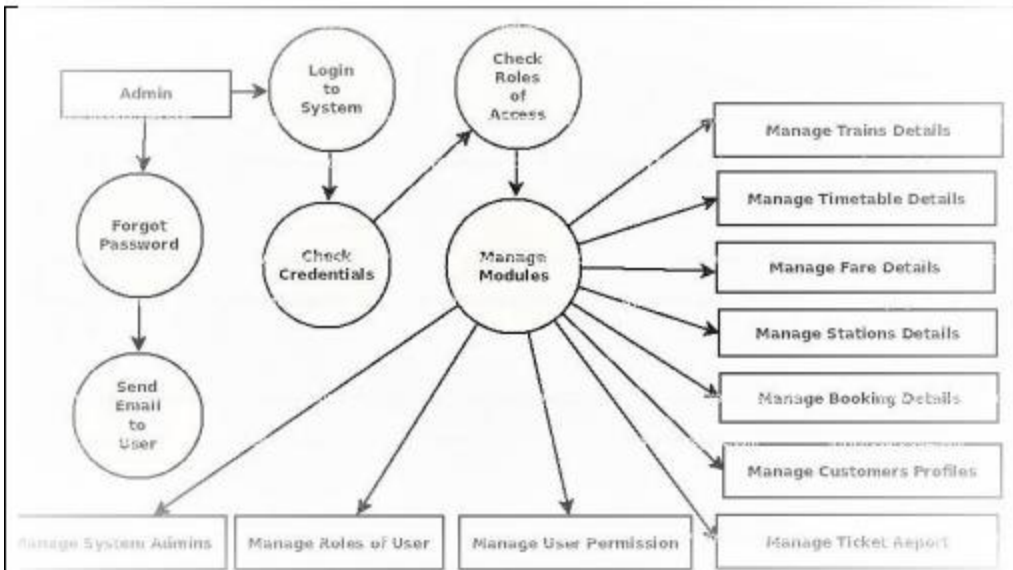
FR No.	Non-Functional Requirement	Description
NFR-1	<b>Usability</b>	Easy to know the availability of seats and location.
NFR-2	<b>Security</b>	User information will be secured and safely stored in the database.
NFR-3	<b>Reliability</b>	Reliable to the user without any failure as it is not fixed to limited number of users.
NFR-4	<b>Performance</b>	All devices are user friendly.
NFR-5	<b>Availability</b>	User can access the availability anytime, anywhere.
NFR-6	<b>Scalability</b>	Support the user with their needs in reserving tickets and location tracking.



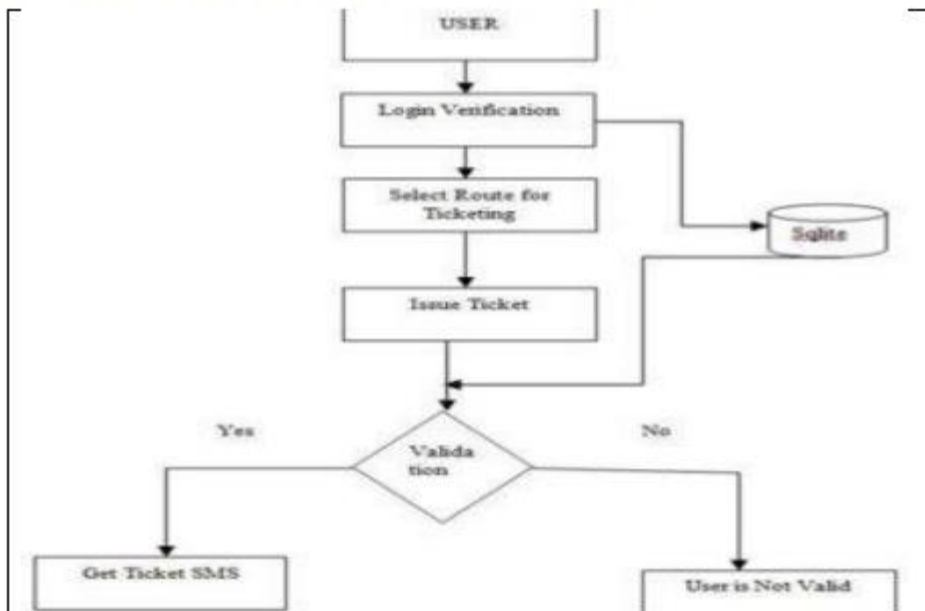
## 5. PROJECT DESIGN

### 5.1 Data Flow Diagrams

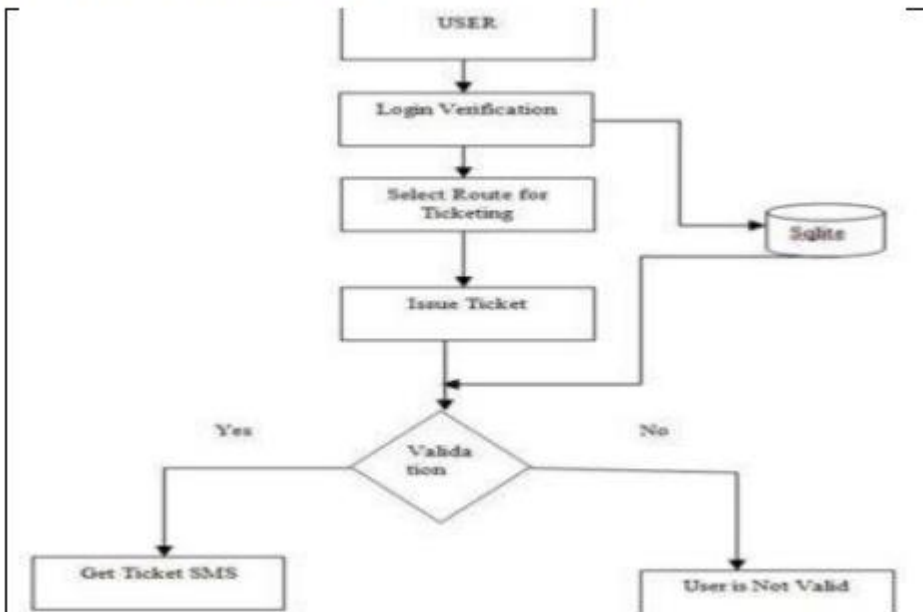
Data flow diagram for the managing system is shown below.



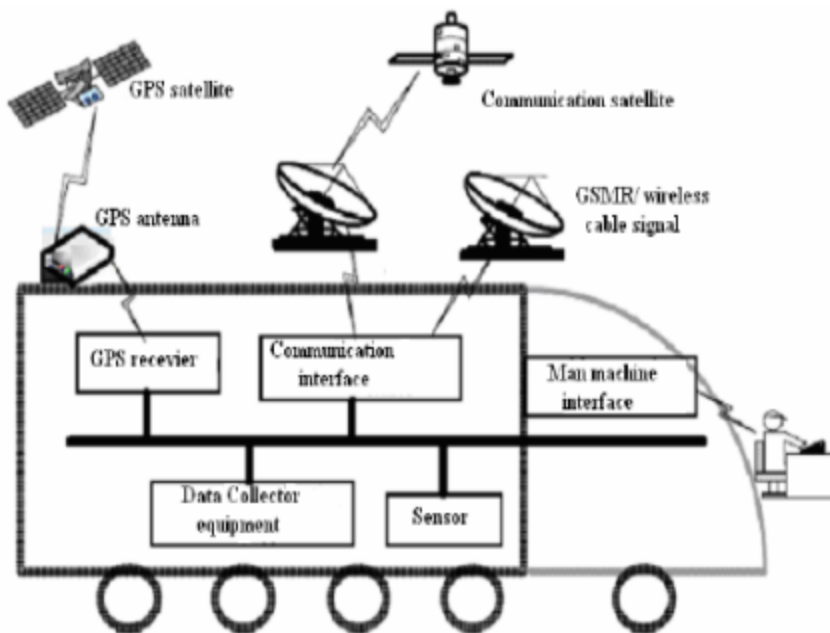
#### DATA FLOW FOR VALIDATION PROCESS:



### DATA FLOW FOR VALIDATION PROCESS:



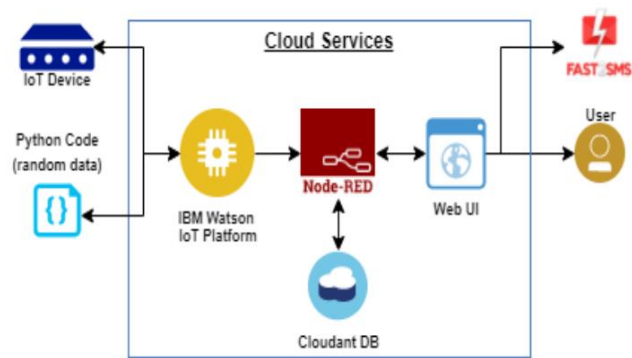
### DATA FLOW DIAGRAM FOR GPS TRACKING:



## 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.

## SOLUTION ARCHITECTURE:



### 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
	Application registration	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

```
import wiotp.sdk.device
import time
import random
myConfig={
    "identity":{
        "orgId":"gagtey",
        "typeId":"GPS",
        "deviceId":"12345"
    },
    "auth":{
        "token":"12345678"
    }
}
def myCommandcallback(cmd):
    print('message received from IBM IOT Platform: %s' % cmd.data['command'])
    m=cmd.data['command']
    client=wiotp.sdk.device.deviceclient(config=myConfig, logHandlers=None)
    client.connect()
    def pub(data):
        client.publishEvent(eventId="status", msgFormat="json", data=mydata, qos=0,
        print('published data successfully: %s' % mydata))
    while True:
        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}
```

## **SPRINT 2**

```
import time
import random
myConfig={
    "identity":{
        "orgId":"gagtey",
        "typeId":"GPS",
        "deviceId":"12345"
    },
    "auth":{
        "token":"12345678"
    }
}

def myCommandcallback(cmd):
    print("messagereceivedfromIBMIOTPlatform:%s"%cmd.data['command'])
    m=cmd.data['command']
    client=wiotp.sdk.device.deviceclient(config=myConfig,logHandlers=None)
    client.connect()
    def pub(data):
        client.publishEvent(eventId="status",msgFormat="json",data=mydata,qos=0,
        print("publishedatasuccessfully:%s",mydata)
    while True:
        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)
```

## SPRINT 3

```
import wiotp.sdk.device
import time
import random
myConfig = {
    "identity": {
        "orgId": "gagtey",
        "typeId": "GPS",
        "deviceId": "12345"
    },
    "auth": {
        "token": "12345678"
    }
}

def myCommandCallback(cmd):
    print("message received from IBM IOT Platform: %s" % cmd.data['command'])
    m = cmd.data['command']
    client = wiotp.sdk.device.deviceclient(config=myConfig, logHandlers=None)
    client.connect()

def pub(data):
    client.publishEvent(eventId="status", msgFormat="json", data=mydata, qos=0,
    print("published data successfully: %s" % mydata))
    while True:
        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)
```



## **SPRINT 4**

```
import wiotp.sdk.device
import time
import random
myConfig={
    "identity":{
        "orgId":"gagtey",
        "typeId":"GPS",
        "deviceId":"12345"
    },
    "auth":{
        "token":"12345678"
    }
}
def myCommandcallback(cmd):
    print('messagereceivedfromIBMIOTPlatform:%s'%cmd.data['command'])
    m=cmd.data['command']
    client=wiotp.sdk.device.deviceclient(config=myConfig,logHandlers=None)
    client.connect()
    def pub(data):
        client.publishEvent(eventId="status",msgFormat="json",data=mydata,qos=0,
        print('publishedatasuccessfully:%s',mydata)
    while True:
        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)
```

## **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 Case**

# Test case 1

Testcases Report_sprint1.xlsx - Microsoft Excel (Product Activation Failed)																
G8 Username: abc@gmail																
Team ID: PNT022MD04739																
Project Name: smart solutions for railways																
Maximum Marks: 4 marks																
Test case ID	Feature Type	Component	Test Scenario	Pre-Requsite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By			
1	Functional	Registration	Registration through the form by filling in my details		1.Click on register 2.Fill the registration form 3.Click Register		Registration form to be filled is to be displayed	Working as expected	Pass				MANGAIYARAKASILE			
2	UI	Generating OTP	Generating the otp for further process		1.Generating of OTP number		user can register through phone numbers, Gmail, Facebook or other social sites and to get into number	Working as expected	Pass				RAHUL M			
3	Functional	OTP verification	Verify user otp using mail		1.Enter gmail id and enter password 2.Click submit	username: abc@gmail.com password: Testing123	OTP verified is to be displayed	Working as expected	Pass				SHANMUGAPRIYA A			
4	Functional	Login page	Verify user is able to log into application with correct credentials		1.Enter user log in page 2.Click on My Account dropdown button 3.Enter invalid username/Email or Email not set 4.Enter valid password in password field 5.Click on login button	username: abc@gmail.com password: Testing123	Registration should show response email or password validation message	Working as expected	Pass				JAI KRISHNA A P			
5	Functional	Display Train details	The user can view about the available train details		1.As a user, I can enter the start and destination to get the list of train available connecting the above	username: abc@gmail.com password: Testing123@7890978907890	A user can view about the available train to enter start and destination details	Working as expected	Pass				MANGAIYARAKASILE			

# Test case 2

Testcases Report_sprint2.xlsx - Microsoft Excel (Product Activation Failed)																
J22																
Team ID: PNT022MD04739																
Project Name: smart solutions for railways																
Maximum Marks: 4 marks																
Test case ID	Feature Type	Component	Test Scenario	Pre-Requsite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By			
1	Functional	Booking	user can provide the basic details such as a name, age, gender etc		1.Enter method of reservation 2.Enter name,age,gender 3.Enter how many tickets wants to be booked 4.Also enter the number member's details like name,age,gender		Tickets booked to be displayed	Working as expected	Pass				MANGAIYARAKASILE			
2	UI	Booking seats	User can choose the class, seat/birth, if a preferred seat/birth isn't available I can be allocated based on the availability		1.known to which the seats are available		known to which the seats are available	Working as expected	Pass				RAHUL M			
3	Functional	Payment	user, I can choose to pay through credit Card/debit card/URL		1.user can choose payment method 2.pay using the method		payment for the booked tickets to be done using payment method through either the following methods credit Card/debit card/URL	Working as expected	Pass				SHANMUGAPRIYA A			
4	Functional	Redirection	user can be redirected to the selected		1.After payment the user will be redirected to the previous page		After payment the user will be redirected to the previous page	Working as expected	Pass				JAI KRISHNA A P			

### Test case 3

[illegible]

## Test case 4

Testcases Report - print.xlsx - Microsoft Excel (Product Activation Failed)

File Home Insert Page Layout Formulas Data Review View

Clipboard Font Alignment Number Styles Cells Editing

Calibri 11

General

Conditional Formatting as Table Cell Styles Insert Delete Format

Σ AutoSum Fill Clear Sort & Find & Filter Select

N8 JAI KRISHNAN.A.P

Test case ID	Feature Type	Component	Test Scenario	Pre-Req	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
1	Functional	Ticket cancellation	user can cancel my tickets there's any change of plan		1.tickets to be cancelled		Tickets booked to be cancelled	Working as expected	pass				MANGAYARAGADILE
2	UI	Raise queries	user can raise queries through the query box or via mail.		1.raise the queries		raise the queries	Working as expected	pass				RAHUL M
3	Functional	Answer the queries	user will answer the queries/doubts Raised by the customers.		1.answer the queries		answer the queries	Working as expected	pass				SHANMUGAPREYA.A
4	Functional	Feed details	a user will feed information about the trains delays and add extra seats if a new compartment is added.		1.information feeding on trains		information feeding on trains	Working as expected	pass				JAI KRISHNAN.A.P

Shopenzer Testcases Testscenarios

## **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.raillyatri.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://github.com/IBM-EPBL/IBM-Project-19581-1659700929>