**Project Report**

Project Name: **SMART SOLUTIONS FOR RAILWAYS**

Team ID: **PNT2022TMID18625**

Team: **VISHNU M - TEAM LEAD**

**Team members: Sowmiya R B**

**Vidhyasri K V**

**Vennilaakshaya V**

# 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

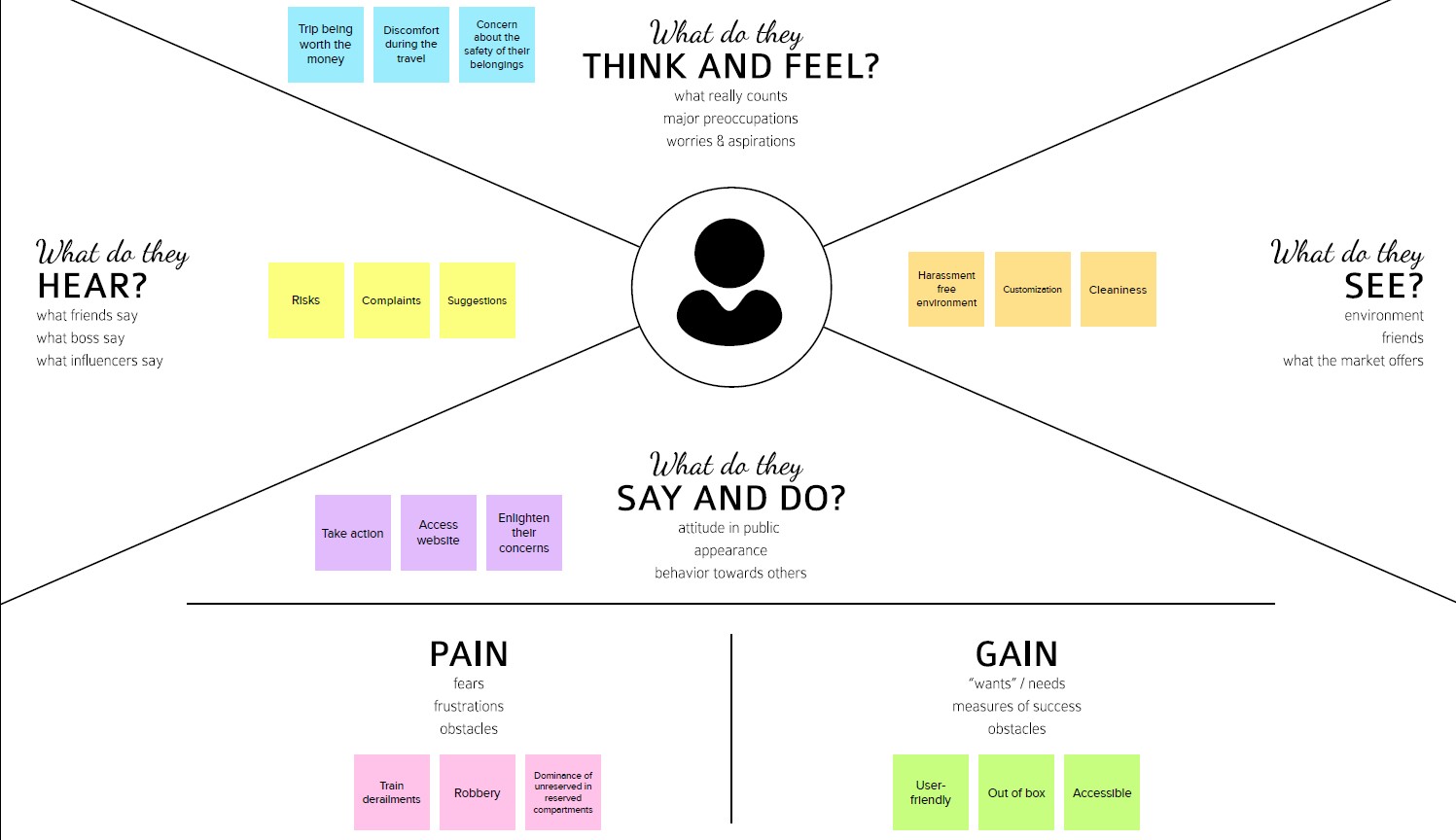
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO | TITLE | AUTHOR | YEAR | KEY TECHNOLOGY |
| 1 | Main geotechnical problems of  railways and roads in kriolitozone and their solutions. | Kondratiev, Valentin G | 2017 | Main problems in railways |
| 2 | Construction and Building Materials | Sañudo, 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

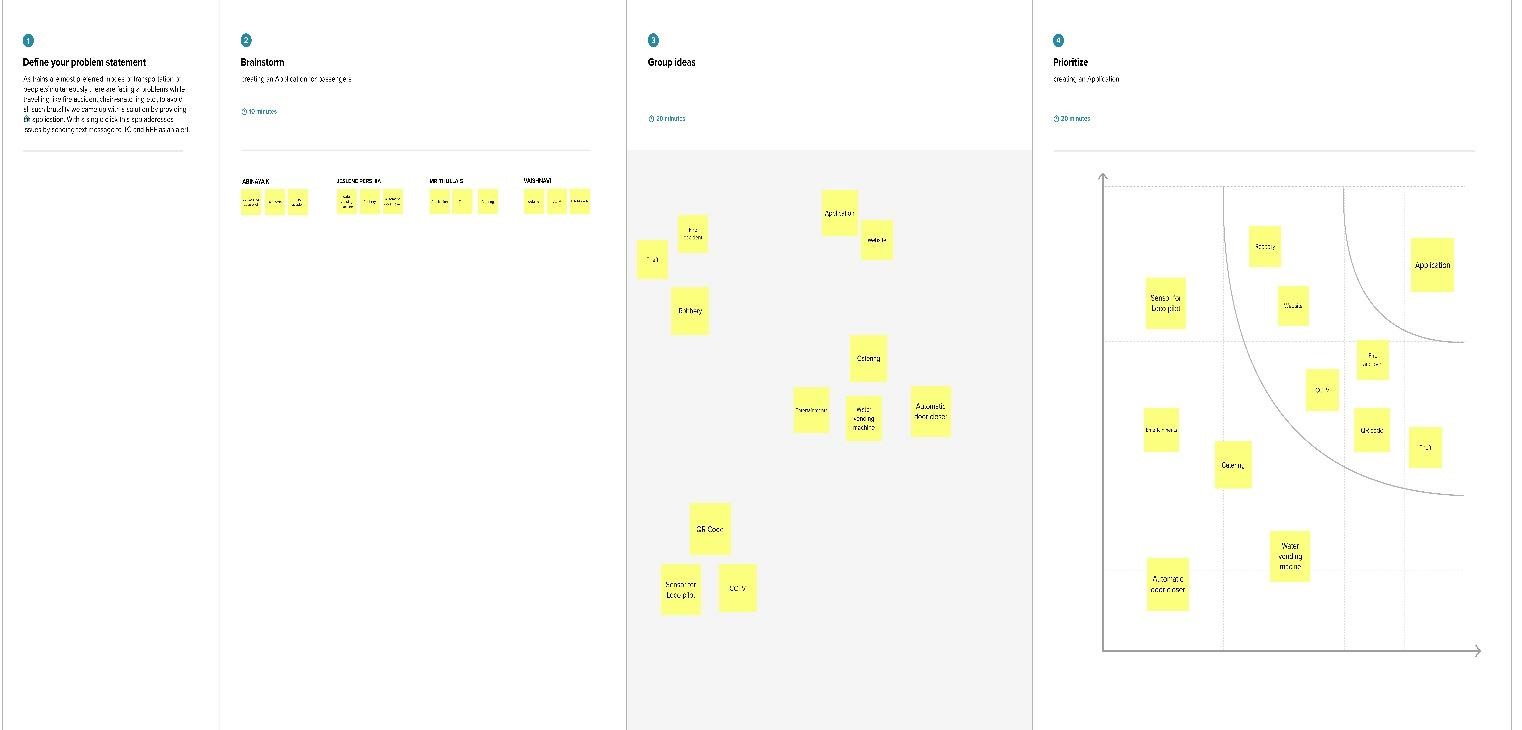
Smart Solutions for railways are designed to reduce the work load of the user and the use of paper.

# 3. IDEATION & PROPOSED SOLUTION

## 3.1 Empathy Map Canvas



## 3.2 Ideation & Brainstorming

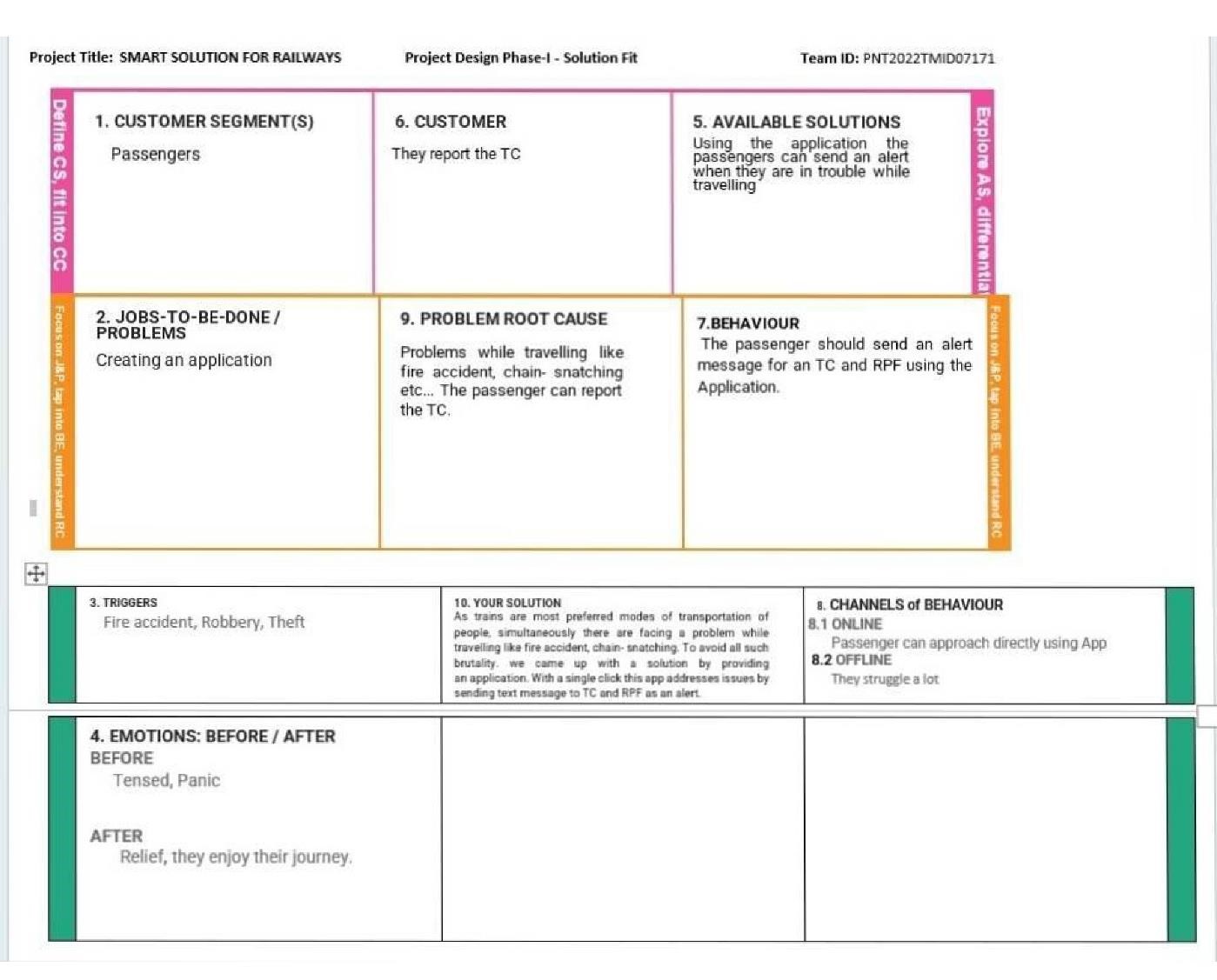


## 3.3 Proposed Solution

|  |  |  |
| --- | --- | --- |
| **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

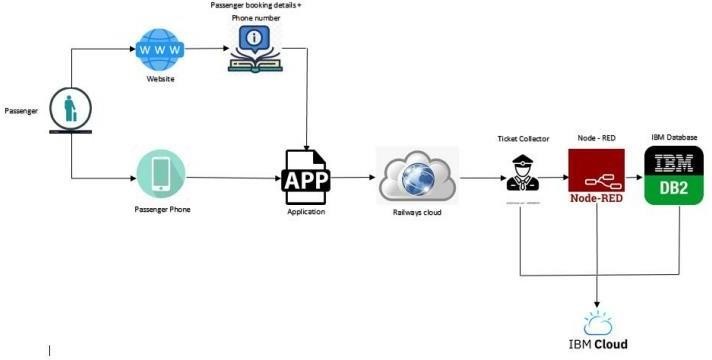
|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Registration | Registration through Online Registration through Gmail |
| FR-2 | User Confirmation | Confirmation via Email Confirmation via OTP |
| FR-3 | Application 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**  **Num**  **ber** | **User Story / Task** | **Acceptance criteria** | **Priority** | **Release** |
| PASSENGER  (Mobile user) | Booking registrat ion | 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 |

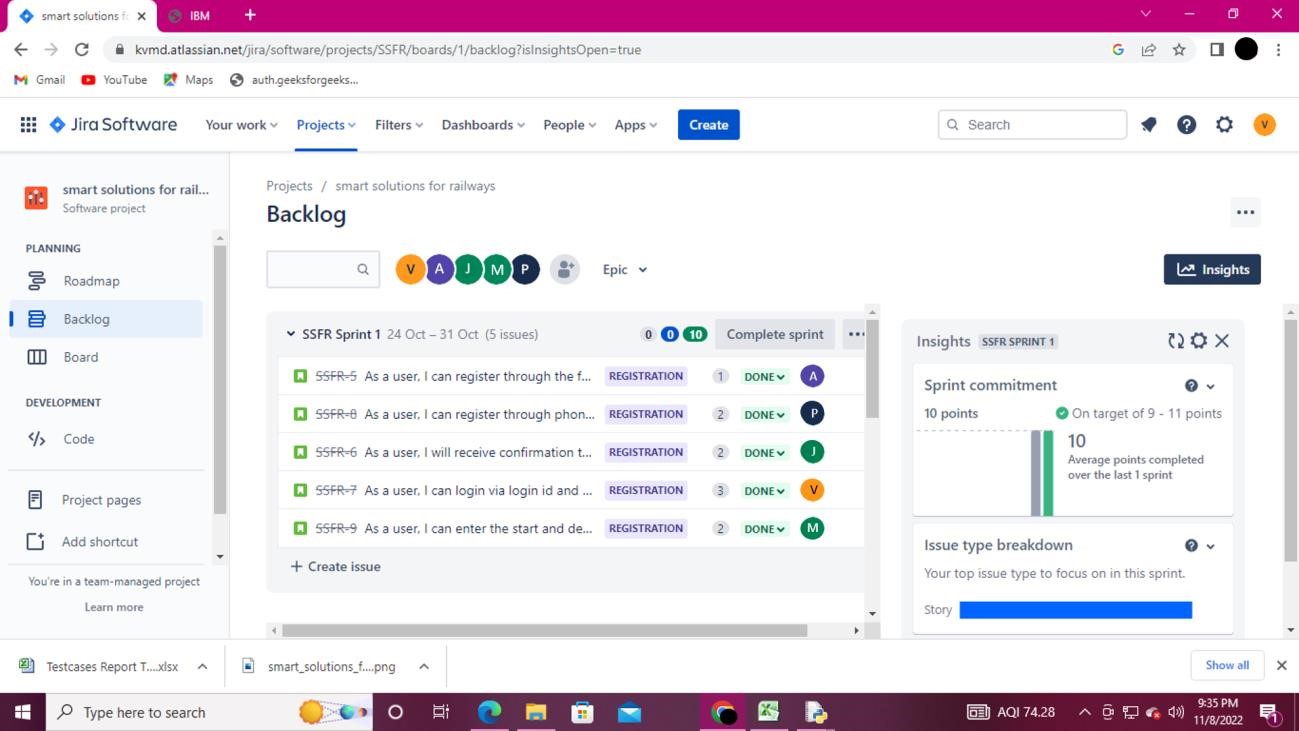
# 6. PROJECT PLANNING & SCHEDULING

## 6.1 Sprint Planning & Estimation

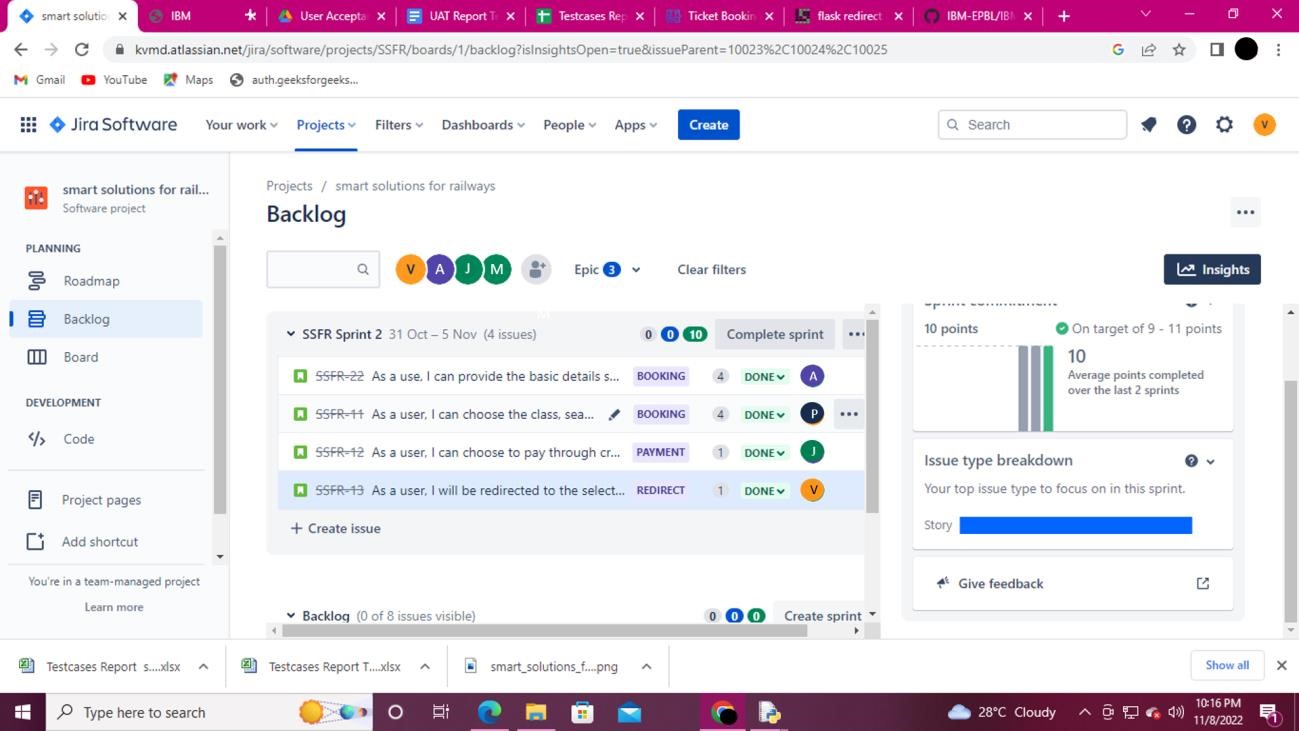
|  |  |
| --- | --- |
| **STEP 1** | Identify the problem |
| **STEP 2** | Prepare an abstract, problem statement |
| **STEP 3** | List required objects needed |
| **STEP 4** | Create a code and run it |
| **STEP 5** | Make a prototype |
| **STEP 6** | Test with the created code and check the designed prototype is working |
| **STEP 7** | Solution for the problem is found |

**6.2 Reports from JIRA**

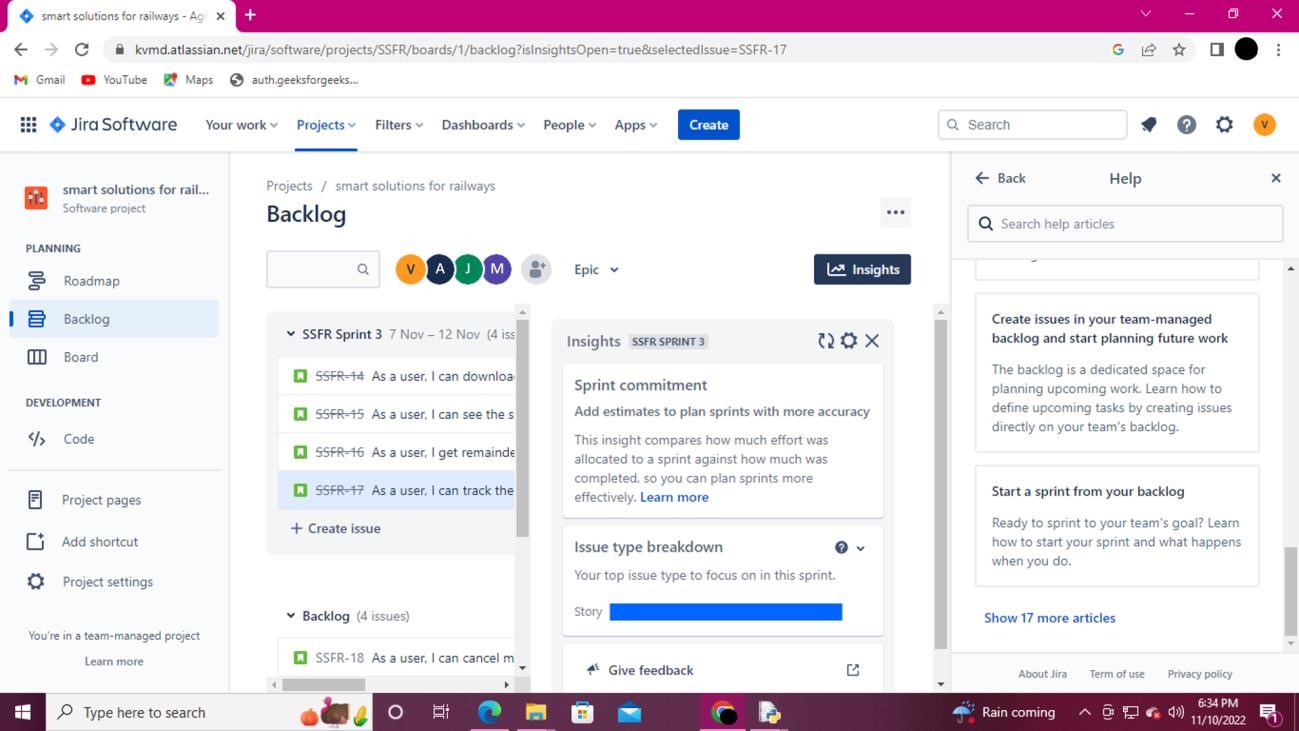
## SPRINT 1



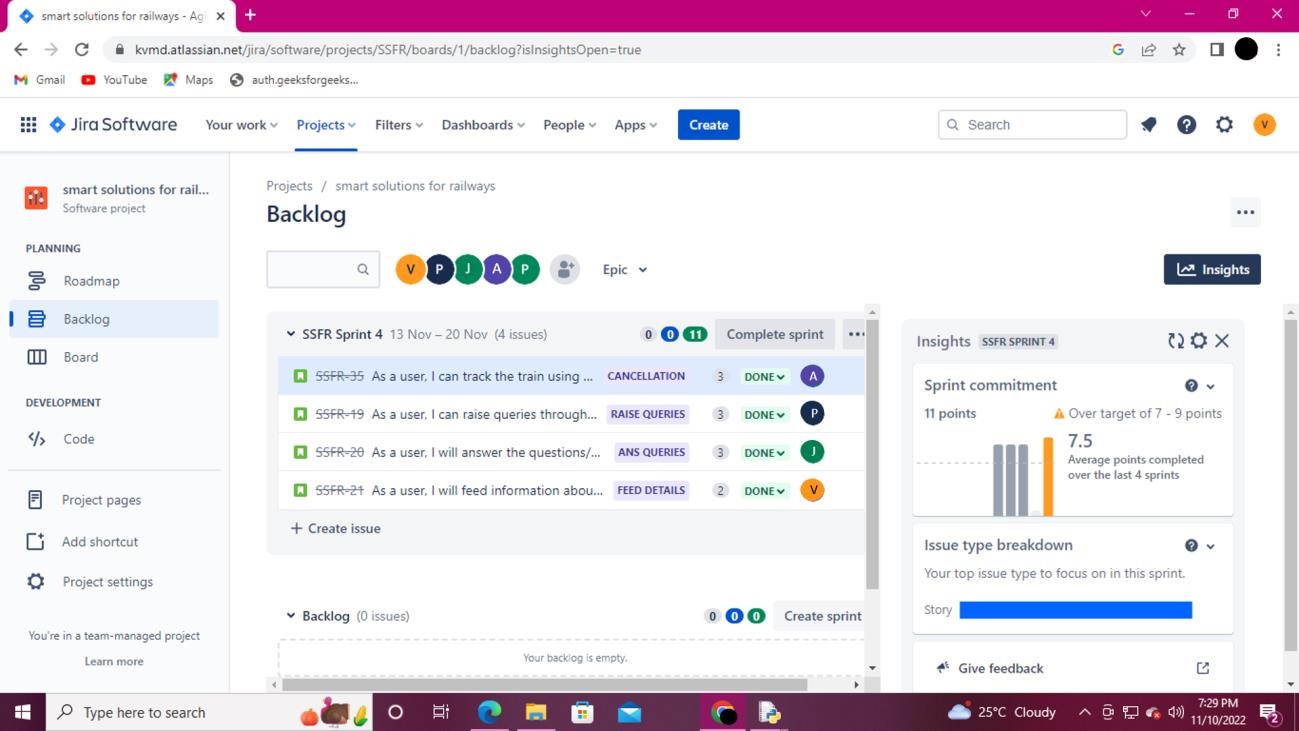
## SPRINT 2



## SPRINT 3



## SPRINT 4



# 7. CODING & SOLUTIONING

## 7.1 Feature 1

* IoT device
* IBM Watson Platform
* Node red
* Cloudant DB
* Web UI
* MIT App Inventor
* Python code

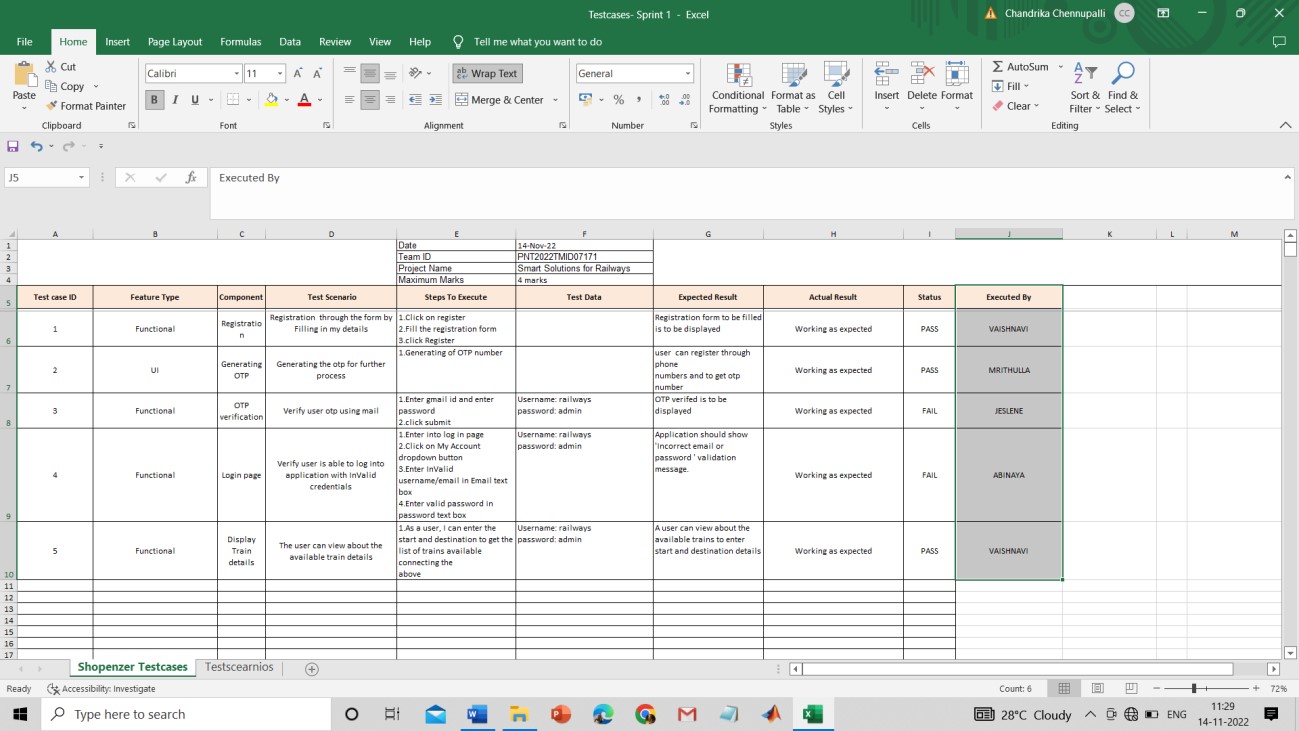
## 7.2 Feature 2

* Login
* Verification
* Ticket Booking
* Adding rating

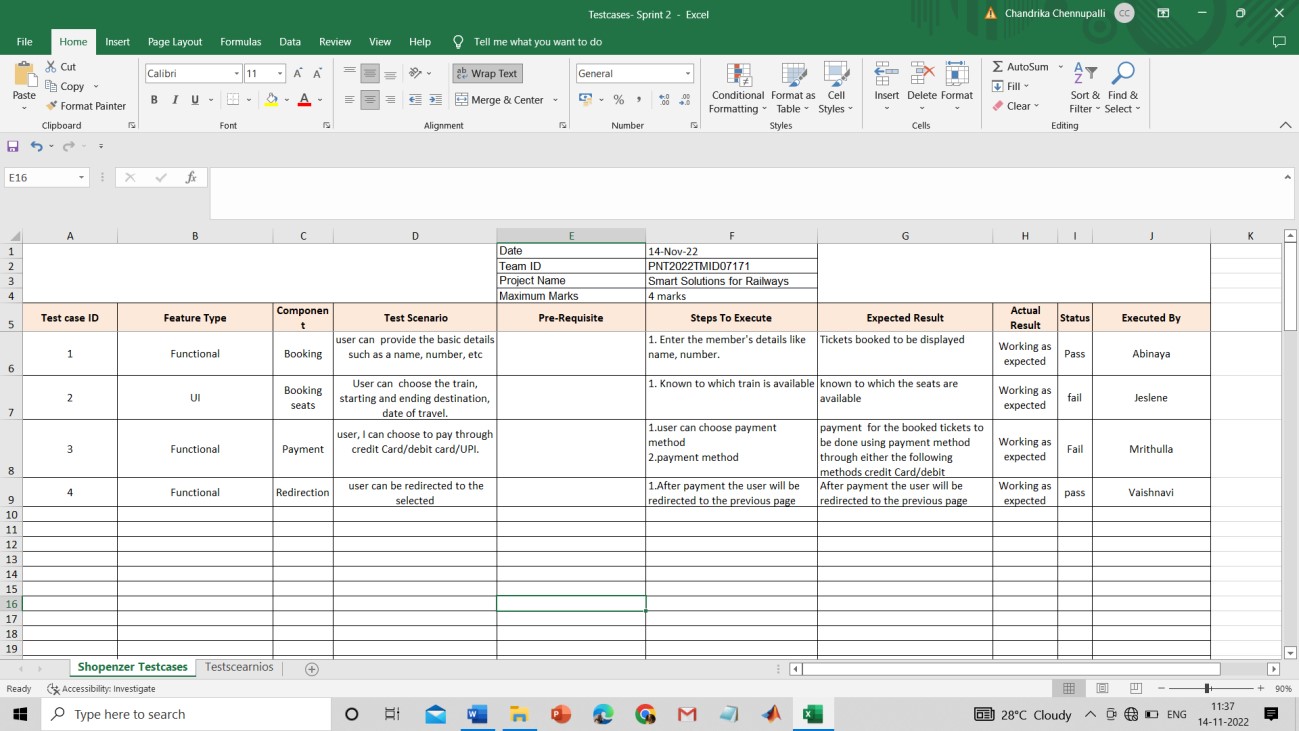
# 8. TESTING AND RESULTS

**8.1 Test Cases**

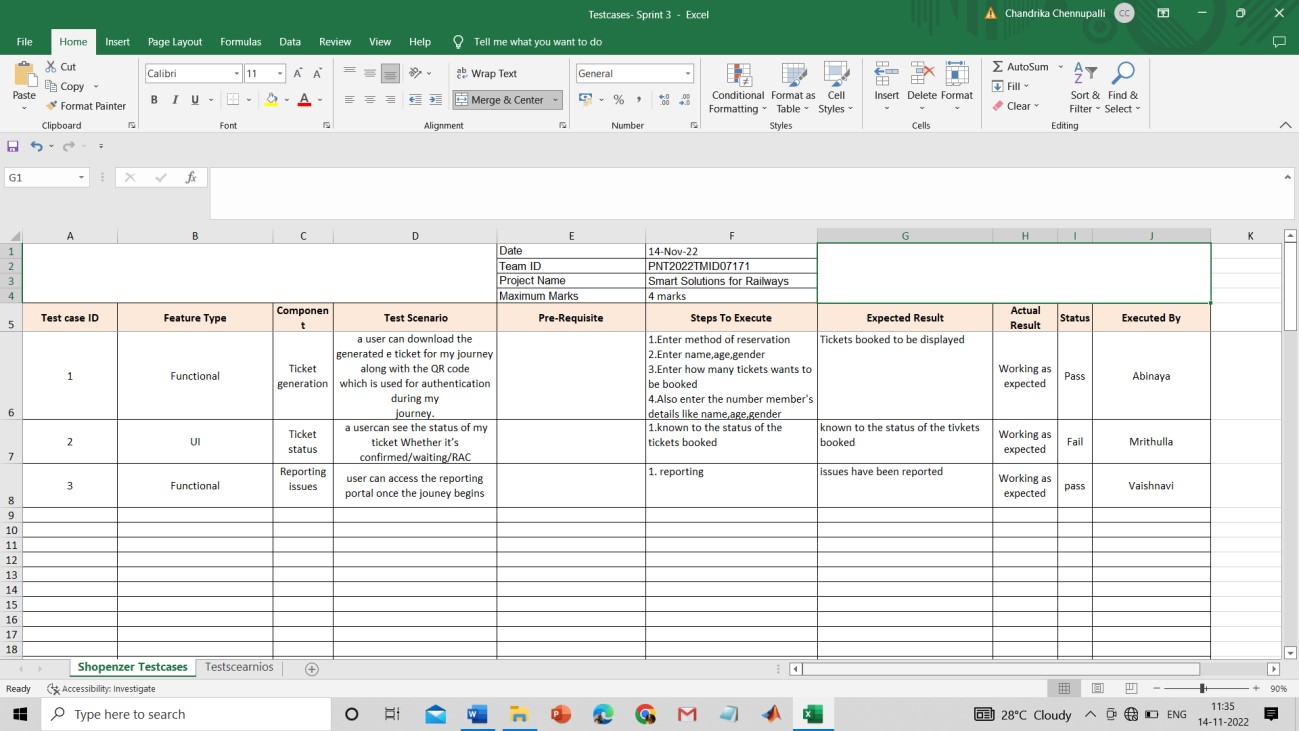
## Test case 1



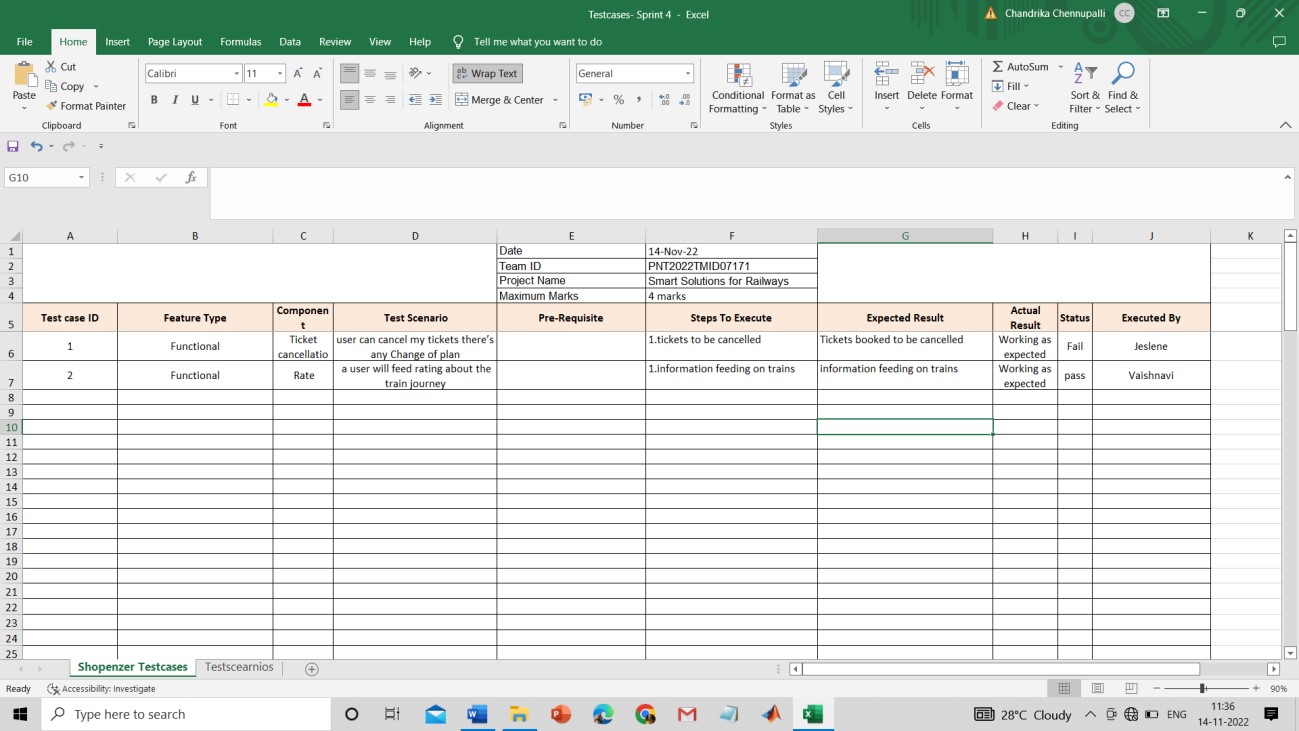
## Test case 2



## Test case 3



## Test case 4



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