

R.M.D ENGINEERING COLLEGE

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY
AND ENTREPRENEURSHIP

PROJECT REPORT

Project Name: SMART SOLUTIONS FOR RAILWAYS

Team ID: PNT2022TMID14953

Team: MADALA SAI SREEKANTH (11519106078) (Team Lead)

MANNE NAGA BHARATH (111519106084)

KOLLU RUPESH KUMAR (111519106071)

PENUMADULA VENKATA SAI TEJA (111519106302)

Project Report Format:

1. INTRODUCTION	2
2. LITERATURE SURVEY	2
3. IDEATION & PROPOSED SOLUTION	4
4. REQUIREMENT ANALYSIS	9
5. PROJECT DESIGN	10
6. PROJECT PLANNING & SCHEDULING	12
7. TESTING AND RESULTS	13
8. ADVANTAGES	15
9. DISADVANTAGES	15
10. CONCLUSION	16
11. FUTURE SCOPE	17
12. APPENDIX	17

1. INTRODUCTION

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

Purpose

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

2. LITERATURE SURVEY

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.

References

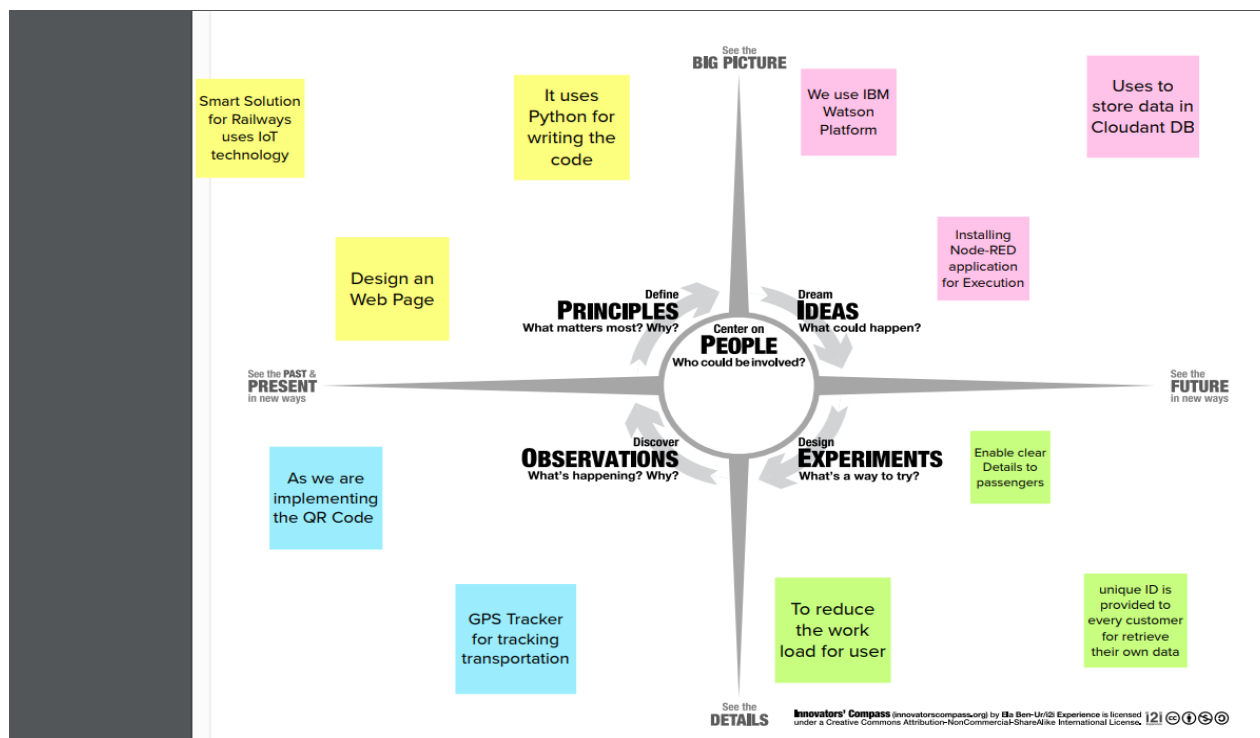
S.NO	Title	Author	Year	Key Technology
1.	Construction and building Materials	Sañudo, Roberto, Marina Miranda, Carlos García, and David GarcíaSanchez	2019	Drainage in railways
2.	Ticketing solutions for Indian railways using RFID technology	Prasanth,Venugopal, and K.P. Soman	2009	Solution for ticketing using RFID
3.	Problems of Indian Railways	Benjamin	2021	Common problems in Indian railways

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

Empathy Map Canvas



Ideation & Brainstorming

Define your problem statement

As trains are most preferred modes of transportation of people, The main problem to face, how to booking train immediately for their purpose. The elder people did not go to book a train ticket

on ticket counter not easy .




Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
1	Due to the various business environments and travel cultures, the e-ticketing strategy has been well utilized in the airline industry, particularly the rail, metro, and bus travel segments.	Like an airline's ticket, online ticket's is also prepaid. Each ticket is assigned a unique number – the Passenger Number Record (PNR) and is personalized to each booking.	Modifications to travel plans are not possible after confirmed booking	After booking the user details will be locked	Ease to user.

Brainstorm

Creating an Application for passengers Digital Railway Solution Digital Twin— Digital platform for railways and airways. Role of sensors in predictive maintenance. Predictive maintenance and CMMS. The IoT-connected trains. Big Data analytics for smart railways. Safety is a key area of concentration

Template



Brainstorm & idea prioritisation


Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room

⌚ 10 minutes to prepare

🕒 1 hour to collaborate

👥 2-8 people recommended

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Before you collaborate

A little bit preparation goes a long way with this session. Here's what you need to do to get going

⌚ 10 minutes

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorm session.

C

Learn how to use the facilitation tools

Use the facilitation superpowers to run a happy and productive session.

[Open article →](#)

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How might We statement. This will be the focus of your brainstorm

⌚ 5 minutes

Problem

How might We book tickets using QR code in railway ticket booking system

Problem

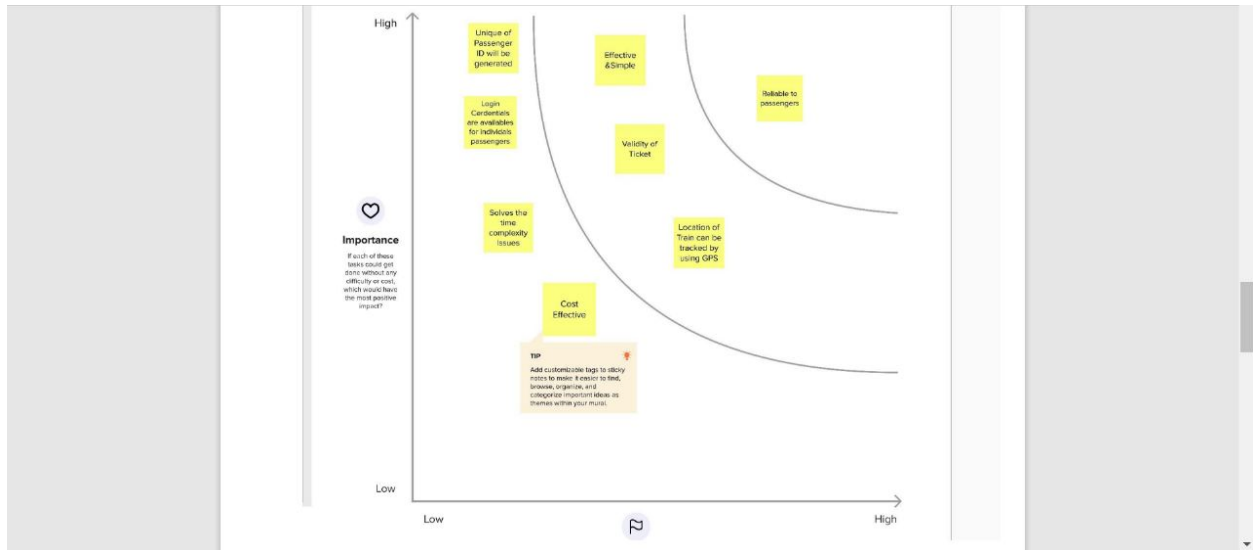
How might We get the details of the passengers?

Problem

How might We track the location?

Problem

How might We get the unique ID?



Proposed Solution

S. N. O.	Parameter	Description
1	Problem Statement (Problem to be solved)	The problem that to be solved in getting details of applicant that are should be present in the ticket
2	Idea / Solution description	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.
3	Novelty / Uniqueness	Smart Solutions for railways are designed to reduce the work load of the user and also the use of paper.
4	Social Impact / Customer Satisfaction	Transportation systems are complex with respect to technology and operations due to the involvement of a wide range of human actors, organizations and technical solutions. There is a need to apply intelligent

		computerized systems for the operation and such as computerized traffic control systems for coordinating advanced transportation.
5	Business Model (Revenue Model)	An increase in rail passengers has meant rail companies must do more to optimize networks to keep up with demand increasing fuel prices and spiraling road congestion has meant that rail travel is experiencing something of a renaissance.
6	Scalability of the solution	A train ticket is a ticket issued by a railway operator that enables the commuters to travel on the railway network. Tickets can authorize the commuters to travel a set itinerary at a specific time. An automatic ticket vending machine was introduced to scrap cvm coupons and avoid long queues.

Problem Solution Fit:

Project Title: Smart Solution for Railway TeamID: PNT2022TMID14953		Project Design Phase-I - Solution Fit Template	
1. CUSTOMER SEGMENT(S) CS Who is your customers? Public who are travelling in the train.	6. CUSTOMER CONSTRAINTS CC What constraints prevent your customers from taking action or limit their choices of solutions? Spending power, budge friendly, no cash, network connection, seats availability	5. AVAILABLE SOLUTIONS AS Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do solutions have? Customer care center will available 24/7, and the chat bots are also available for queries for public .	Explore AS, differential Focus on J&P, tap into BE, understand RC
2. JOBS-TO-BE-DONE / PROBLEMS J&P Which jobs-to-be-done (or problems) do you address for your customers? In their busy schedule as fast roaming world public in need of online booking process. The queues in front of the ticket counters in railway stations have been drastically increased over the period of time	9. PROBLEM ROOT CAUSE RC What is the real reason that this problem exists? What is the back story behind the need to do this job? The main reason for the problem that has occurred for due to lack of technology earlier since passengers find it difficult to book the ticket and track the location of train. To overcome this problem we have introduced QR code and GPS tracker for booking the ticket and finding the location of the train	7. BEHAVIOUR BE What does your customer do to address the problem and get the job done? By listening to the customer we can provide genuine empathy for the problem regarded. By looking over the ration session we can easily find out how the customer gets issues while using the application.	

3. TRIGGERS What triggers customers to attract? Saves paper and work load	10. YOUR SOLUTION *A webpage is designed in which the user can book tickets and will be provided with a QR code which will be shown to the ticket collector and the ticket collector will be scanning the QR code to get the passenger details. * The webpage also shows the live locations of the train by placing a GPS module in the	8. CHANNELS of BEHAVIOUR ONLINE: People can book their tickets through online and they get a QR code through SMS. OFFLINE: In web application passenger details is stored and the ticket collector can view their details at any time.
4. EMOTIONS: BEFORE / AFTER How do customers feel when they face a problem or a job and afterwards? <ul style="list-style-type: none"> • NO NEED OF TAKING PRINT OUT • COUNTER TICKET HAS TO BE HANDLED WITH CARE, BUT SMS ON MOBILE IS ENOUGH. • YOU ARE BECOMING ENVIRONMENT FRIENDLY AND CONTRIBUTING FOR GREENER PLANET BY IGNORING PRINTOUT. • NO NEED OF TAKING OUT WALLET AND SHOWING YOUR TICKET TO TTR, JUST TELL YOUR NAME TO TTR THAT YOU ARE PASSENGER WITH A VALID PROOF. • WHILE BOOKING COUNTER TICKET YOU HAD TO CARRY CASH AND WHILE BOOKING E-TICKET YOU ARE PAYING THROUGH ONLINE DIRECTLY FROM BANK WHICH MAKES WORK MORE EASY FOR YOU. 	train. The location of the journey will be updated continuously in the webpage. * The booking details of the user will be stored in the database which can be retrieved anytime.	

4. REQUIREMENT ANALYSIS

Functional Requirements:

Following are the functional requirements of the proposed solution.

S.No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
1	User Registration	Registration through Phone Number Registration through Gmail
2	User Confirmation	Confirmation via Email Confirmation via OTP
3	Login to system	Check Credentials Role of Access
4	Manage Modules	Manage System Admins Manage Roles of User Manage user Permission
5	Check whether details	Candidate Details Booking Details QR Code Generation
6	Log Out	Exit

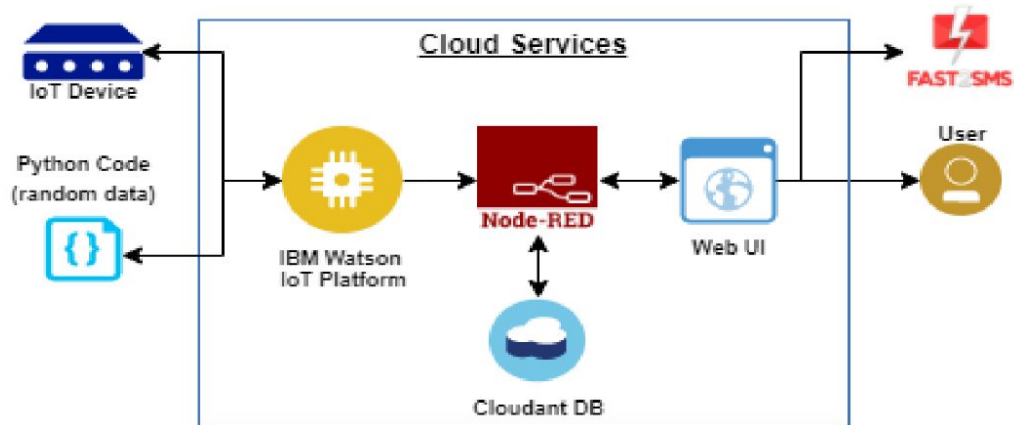
Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

S.No.	Non-Functional Requirement	Description
1.	Usability	Focus on physical accessibility (driven by demo trends) likely stratification by affordability.
2	Security	Smart Design with a holistic view of the design, including safety and hazard issues. While this is an established process, using it for capturing safety risks and hazards is only slowly gaining momentum.
3	Reliability	Modelling of customer relations, embracing the understanding of customer experience, cross - channel coherence including self-service.
4	Performance	This will result in standardization, transparency, and scalability in the data, which operators can then use to gain better insights and increased efficiencies.
5	Availability	Predictive maintenance incorporates condition - based monitoring and is based on a forecast of future condition of a railway component using advance analytics with real -time conditions data .
6	Scalability	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 .

5. PROJECT DESIGN

Data Flow Diagrams



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.

User Stories

User Stories

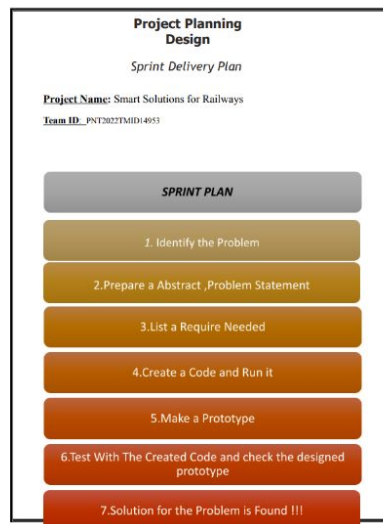
Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Mobile Number	I can register & access the dashboard with Mobile Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard					

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)		USN-6	As a user can view the dashboard, and this dashboard include the check roles and can access the content which was present	I can view the dashboard in Smart Solutions for Railways	Medium	
		USN-7				
Customer Care Executive			As a user once view the manage modules this describes the Roles and can be able to get the required request			
Administrator						

6. PROJECT PLANNING & SCHEDULING

Sprint Planning & Estimation



CODING & SOLUTIONING

Feature 1

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

Feature 2

- Login Verification
- Ticket Booking
- Adding rating

7. Testing and Results

Test Case 1:

	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)
4	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		
5							
6	1.Generating of OTP number		user can register through phone numbers and to get otp number	Working as expected	PASS		
7	1.Enter gmail id and enter password 2.click submit	Username: railways password: admin	OTP verified is to be displayed	Working as expected	FAIL		
8	1.Enter into log in page 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: railways password: admin	Application should show 'incorrect email or password ' validation message.	Working as expected	FAIL		
	1.As a user, I can enter the start and destination to get the list of trains available connecting the	Username: railways password: admin	A user can view about the available trains to enter start and destination details	Working as expected	PASS		

Test Case 2:

4	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)
5	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		
6	1.Generating of OTP number		user can register through phone numbers and to get otp number	Working as expected	PASS		
7	1.Enter gmail id and enter password 2.click submit	Username: railways password: admin	OTP verified is to be displayed	Working as expected	FAIL		
8	1.Enter into log in page 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: railways password: admin	Application should show 'Incorrect email or password ' validation message.	Working as expected	FAIL		
	1.As a user, I can enter the start and destination to get the list of trains available connecting the	Username: railways password: admin	A user can view about the available trains to enter start and destination details	Working as expected	PASS		

Test Case 3:

4	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
5	a user can download the generated e ticket for my journey along with the QR code which is used for authentication during my journey.		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
6	a user can see the status of my ticket Whether it's confirmed/waiting/RAC		1.known to the status of the tickets booked		known to the status of the tickets booked	Working as expected	Fail
7	user can access the reporting portal once the journey begins		1. reporting		issues have been reported	Working as expected	pass
8							
9							
10							
11							
12							
13							
14							

Test Case 4:

4	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)
5	1.tickets to be cancelled		Tickets booked to be cancelled	Working as expected	Fail		
6	1.information feeding on trains		information feeding on trains	Working as expected	pass		
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							

8. 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

9. DISADVANTAGES

- Network issues may arise.

10. 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

11. 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

12. APPENDIX

Source Code

GPS Tracker:

Scanner:

```
from ibmcloudant import CouchDbSessionAuthenticator
from ibm_cloud_sdk_core.authenticators import BasicAuthenticator

authenticator = BasicAuthenticator('apikey-v2-16u3crmdpk
ghhxefdi kvpssoh5fwezrmuup5fv5g3ubz', 'b0ab119f45d3e6255eabb978')
service = CloudantV1(authenticator=authenticator)

service.set_service_url('https://apikey-v2-16u3ermdpkghhxefdik
vpssoh5fwezrmuup5fv5g3ubz:b0ab119145d3e6255eabb978e7e2f0')

cap= cv2.VideoCapture(0)
```

```
font = cv2.FONT_HERSHEY_PLAIN
```

```
while True:
```

```
    _, frame = cap.read()
    decodedObjects = pyzbar.decode (frame)
    for obj in decodedObjects:
        #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:
            response = service.get_document(
                db='booking',
                doc_id = a
            ).get_result()
            print (response)
            time.sleep(5)
        except Exception as e:
            print ("Not a Valid Ticket")
            time.sleep(5)

        cv2.imshow("Frame",frame)
        if cv2.waitKey(1) & 0xFF ==ord('q'):
            break
```

```
cap.release()
```

```
cv2.destroyAllWindows()
```

```
client.disconnect()
```

Code:

```
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, onPublish=None)
    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}
    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 = myCommandCallback
    client.disconnect ()

```

Web Application:

```
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>FRED-powered chat app</title>
  <script src="http://code.jquery.com/jquery-1.11.3.min.js"></script>
  <script src="http://code.jquery.com/jquery-migrate-1.2.1.min.js"></script>
</head>
<body>
  <div id="messages"></div>
  <!-- Important to configure onsubmit to false to prevent page refresh when pressing the
return key -->
  <form id="form" onsubmit="return false;">
    <input id="text" type="text" onkeypress="return sendText(event)" />
  </form>
  <script type="text/javascript">
```

```
// Open a websocket using FRED.
var publishSocket = new
WebSocket("wss://{yourusername}.fred.sensetecnic.com/api/public/messagereceive");
//server receives
var listenSocket = new
WebSocket("wss://{yourusername}.fred.sensetecnic.com/api/public/messagepublish");
//server publishes
```

```
listenSocket.onmessage = function (event) {
  // When receiving a message append a div child to #messages
  data = JSON.parse(event.data);
  $("#messages").append("<div class='msg sentiment'+data.sentiment+'\"
>"+data.timestamp+" - "+data.msg+"</div>")
  if ($("#messages").children().length > 10 ) { $("#messages :first-child").remove();}
}
```

```
function sendText(event) {
  if (event.keyCode == 13) { // Only if return key pressed
    // Construct object containing the data the server needs.
```

```

d = new Date();
var data = {
msg: $("#text").val(),
timestamp: d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()
};
publishSocket.send(JSON.stringify(data)); // Send the msg object as a JSON-formatted
string.
$("#text").val(""); // Blank the text input element
}
}
</script>
</body>
</html>

```

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.railatri.in/booking/trains-
betweenstations?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.railatri.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):
    ifself. 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")
```

GitHub Link:

<https://github.com/IBM-EPBL/IBM-Project-33412-1660219902>

Demo Link : <https://youtu.be/gDhsl5sT0Qc>

