PROJECT REPORT

SMART SOLUTIONS FOR RAILWAYS

1. **INTRODUCTION**
    1. Project Overview
    2. Purpose
2. **LITERATURE SURVEY**
    1. Existing problem
    2. References
    3. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
    1. Empathy Map Canvas
    2. Ideation & Brainstorming
    3. Proposed Solution
    4. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
    1. Functional requirement
    2. Non-Functional requirements
5. **PROJECT DESIGN**
    1. Data Flow Diagrams
    2. Solution & Technical Architecture
    3. User Stories
6. **PROJECT PLANNING & SCHEDULING**
    1. Sprint Planning & Estimation
    2. Sprint Delivery Schedule
    3. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
    1. Feature 1
    2. Feature 2
    3. Database Schema (if Applicable)

8. **TESTING**
    1. Test Cases
    2. User Acceptance Testing
9. **RESULTS**
    1. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

    Source Code

    GitHub & Project Demo Link

1. INTRODUCTION

1. Project Overview

Smart solutions for railways is to manage Indian Railways is the largest railway network in Asia and additionally world's second largest network operated underneath a single management. As Railways are the most preferable means of transport, it is principal mode of transportation for freight and passengers across the country for business, sight seeing, education ,etc and also serves a crucial role in the growth of developing nation. Hence, there is a need to optimize the processes surrounding rail transportation to deliver an efficient way to support a massive users.

2. PURPOSE

i) Smart solutions for railways help railways successfully manage passenger safety, operational efficiency, and the passenger experience. Smart sensors can be used to track important assets, manage passenger flow, and enable predictive maintenance.

ii) The purpose of this project is to develop a digitized ticket generation using QR Code to ease the e-ticket booking , so as to boost paper-less tickets and to promote paper-less work of the TTE and to ensure timeliness among the public by providing them facility to track the train location.

iii) Connect people, sensors, trains and automated train systems with the highest security. Transform your communications and operations from departure to destination and beyond. Secure communications. Enhancing overall service. Lower operational cost IoT applications.

2. LITERATURE SURVEY

1. Existing problem

i) Passengers must reach the station and have to wait in the queues to get their tickets reserved. This makes them hesitate to use the railways to which they prefer private vehicles.

ii) They must preserve their tickets safe till the end of the journey which adds up to their burden.

iii) The TTE must validate each tickets physically by using pen and paper

iv) The Passengers keep on looking for the arrival of trains without knowing whether the train has left or yet to arrive or where its location is.

2. References

S.NO	TITLE	AUTHOR	YEAR	KEY TECHNOLGY
1.	A REVIEW ON IOT BASED AUTOMAED SEAT ALLOCATION	1. Sarvath Saba 2. Sharon Philip 3. Shriharsha 4. Mukund Naik,	2022	Solution for over croed in trains using QR Code using IoT.

	AND VERIFICATION USING QR CODE	**5. Sudeep Sherry**		
2.	**TRAIN TRACKING SYSTEM USING RFID AND IOT**	**1. Yogesh Nimbalkar 2.AkshayMemane 3. Rahul Ahire**	**2017**	**A unique RFID code is given to each train through which they are being identified using IOT**
3.	**A SECURE FREIGHT TRACKING SYSTEM IN RAILS USING GPS TECHNOLOGY**	**1.Priyadharshini.T 2. Raghavi.N 3.Rajashree.B**	**2017**	**Infrared sensors are used to diagnose obstacles and all the informations are communicated via GPS.**
4.	**TRAIN TICKETING SYSTEM**	**1. Ramesh.C**	**2013**	**The rechargeable smart card**

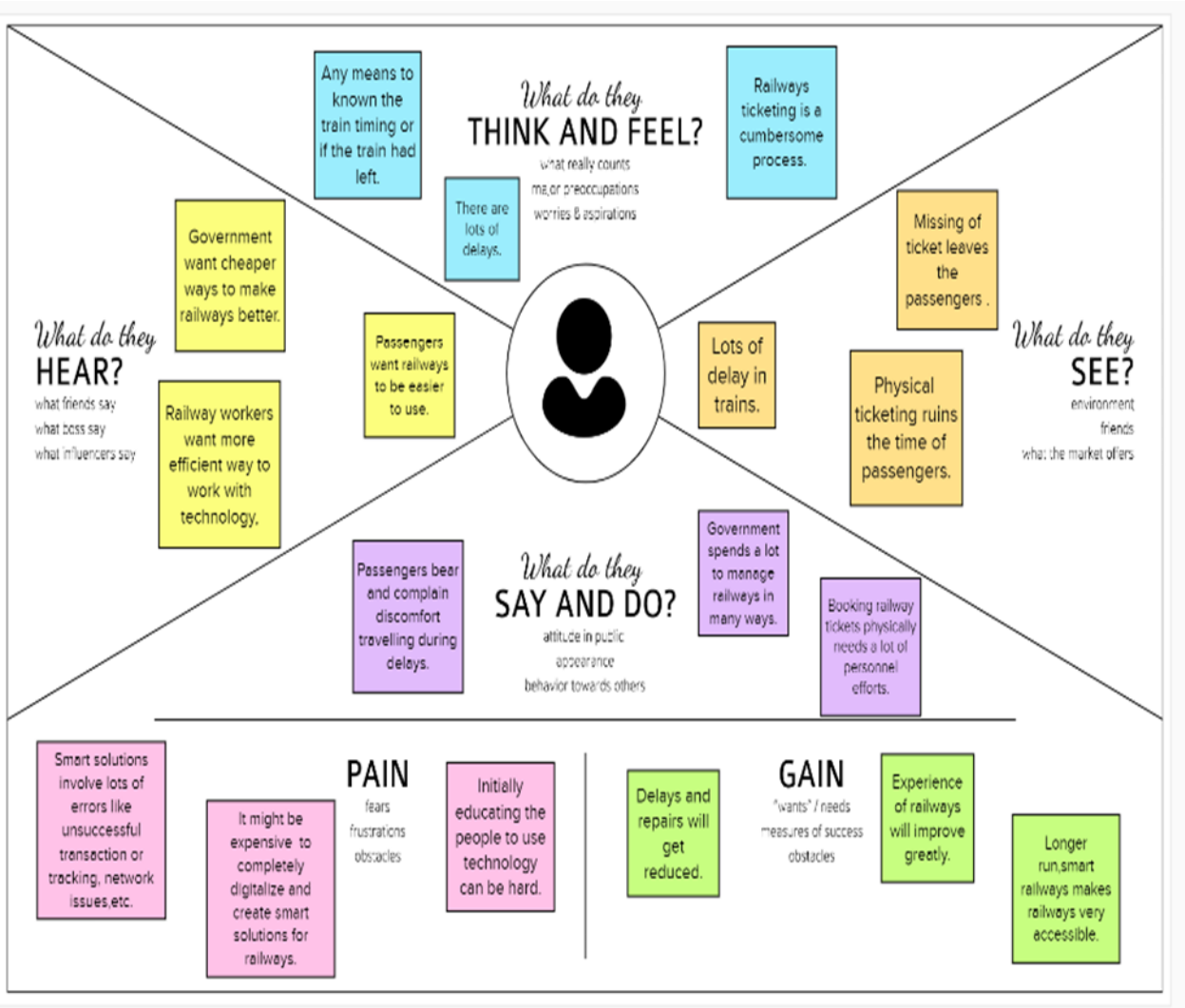
	USING SMART CARD			**can be operated after giving a correct secret password to get the availabitity of seats that will be printed at the end**
5.	**ENHANCEMENT OF RAILWAY RESERVATION SYSTEM USING INTERNET OF THINGS**	**1.B.Mallikarjuna**	**2018**	**Survey of broandBand technologies that benefits enormously.**

3.Problem Statement Definition

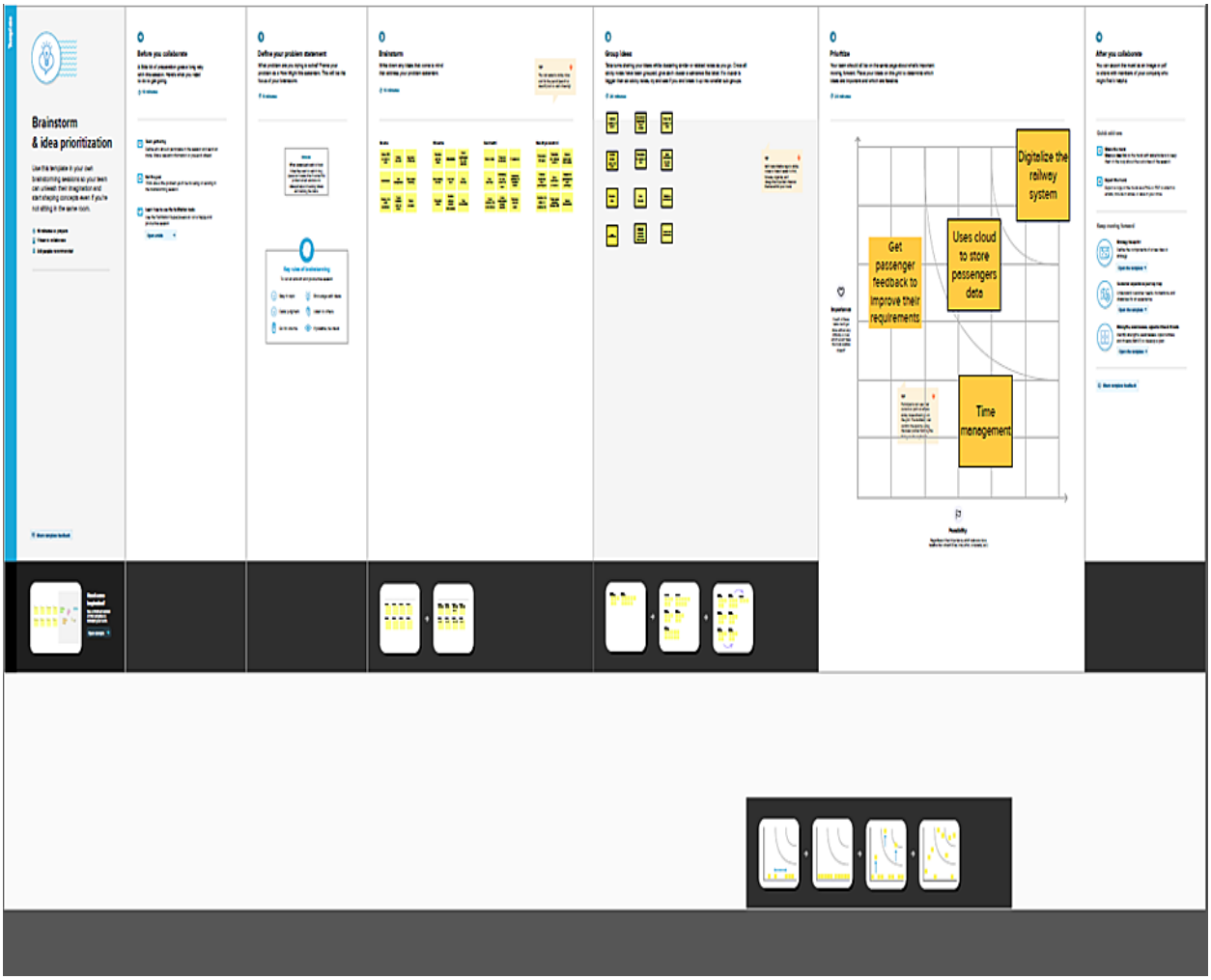
Problem Statement	I am	I am trying to	But	Because	Which makes me feel
PS-1	Passenger	Get a ticket in the station.	Need to wait for a long time in a queue	Massive users of railways and unupdated technology of railways.	Hesitated
PS-2	TTE	Check the tickets physically.	Miserable situation between me and the passengers.	Tickets missed by the passengers	Unanswerable
PS-3	Passenger	Board a train	Waiting for the train to arrive	No facilities to get the train location	Irritated

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)	· To provide an efficient way by introducing paperless tickets using QR code · To design a GPS module to track the location of the train.
2.	Idea / Solution description	· GPS tracker is placed in the train so that the passengers can track the location of the train even it is delayed. · Passengers can book their tickets using the website which is possible at anytime, anywhere. · Smart ticketing to avail seasons so that physical work is eradicated.
3.	Novelty / Uniqueness	· This project stands unique from the existing ones, by implementing facilities for getting train seasons online and the users can get to know the status which reduces the visit of customers and makes the task of railway workers easier.

4.	Social Impact / Customer Satisfaction	No Queuing to get tickets and burdenless because of e-tickets. · Elimination of dilemma whether the train has left or yet to arrive. · Can get the status and avail of e-seasons instead of visiting the station physically every time.
5.	Business Model (Revenue Model)	· This project enables railways to optimise their services by implementing e-ticketing when compared to the cost involved in paper ticketing thereby profiting with an increase in the number of users.
6.	Scalability of the Solution	The solution comprises high scalability to meet the increasing demand of users over the nation for more efficient and comfortable services.

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 Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Train Tracking using GPS system	Tracking the Train location via Web Application
FR-4	Train Season Generation	Registration through Form
FR-5	User Confirmation	Confirmation via Email Confirmation via OTP

4.2 Non-Functional requirements

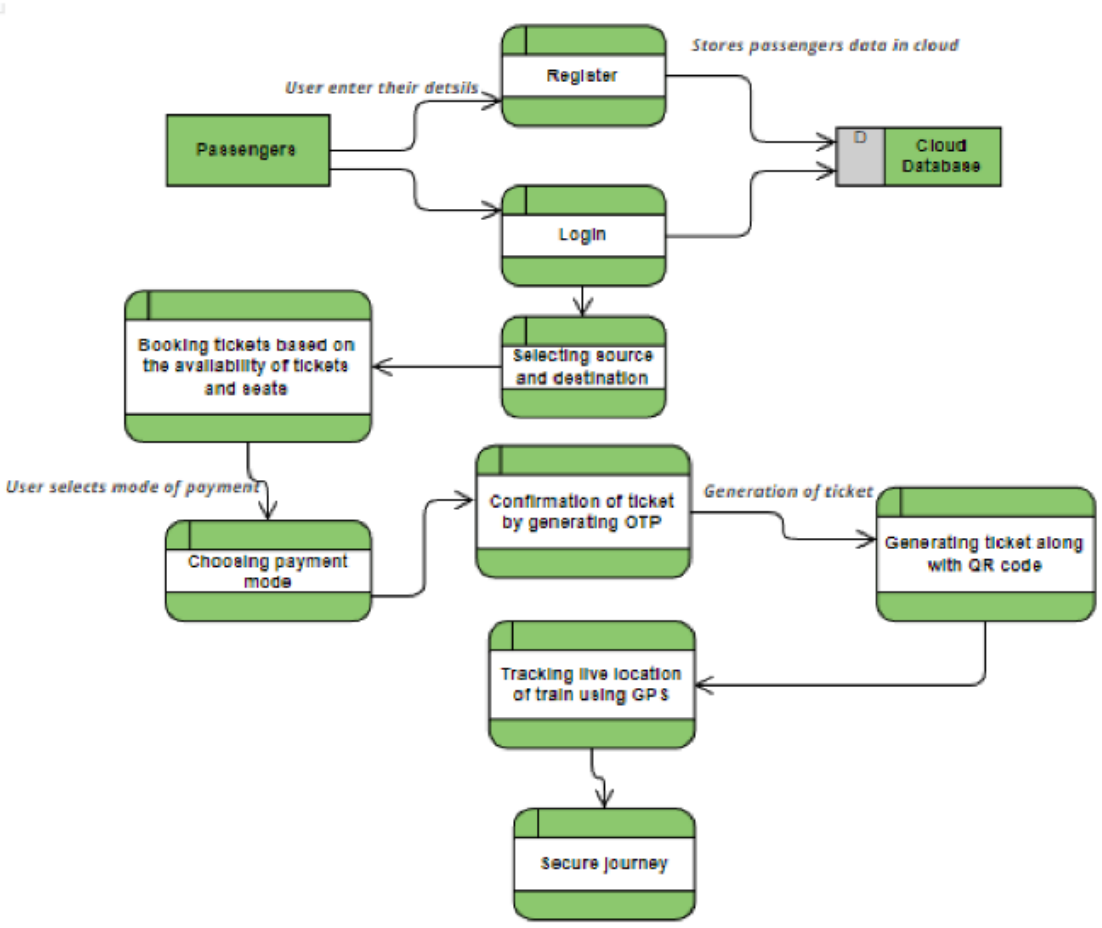
FR No	Non-Functional Requirement	Description
NFR-1	Usability	A friendly functionality that supports novice users and makes burdenless instead of maintaining the paper tickets.
NFR-2	Security	Users personal data is highly secured because no leakage of data to the third parties.
NFR-3	Reliability	Well grounded due to the confirmation and uniqueness of QR Code.

NFR-4	Performance	It provides quick and accurate information .This helps in timeliness of passengers and TTEs.
NFR-5	Availability	This web application requires only browsers so that it can support any device of all size and the user need not worry about its storage.
NFR-6	Scalability	Since the application is built on cloud, it is highly scalable enabling large amount of users to use it efficiently.

5. PROJECT DESIGN

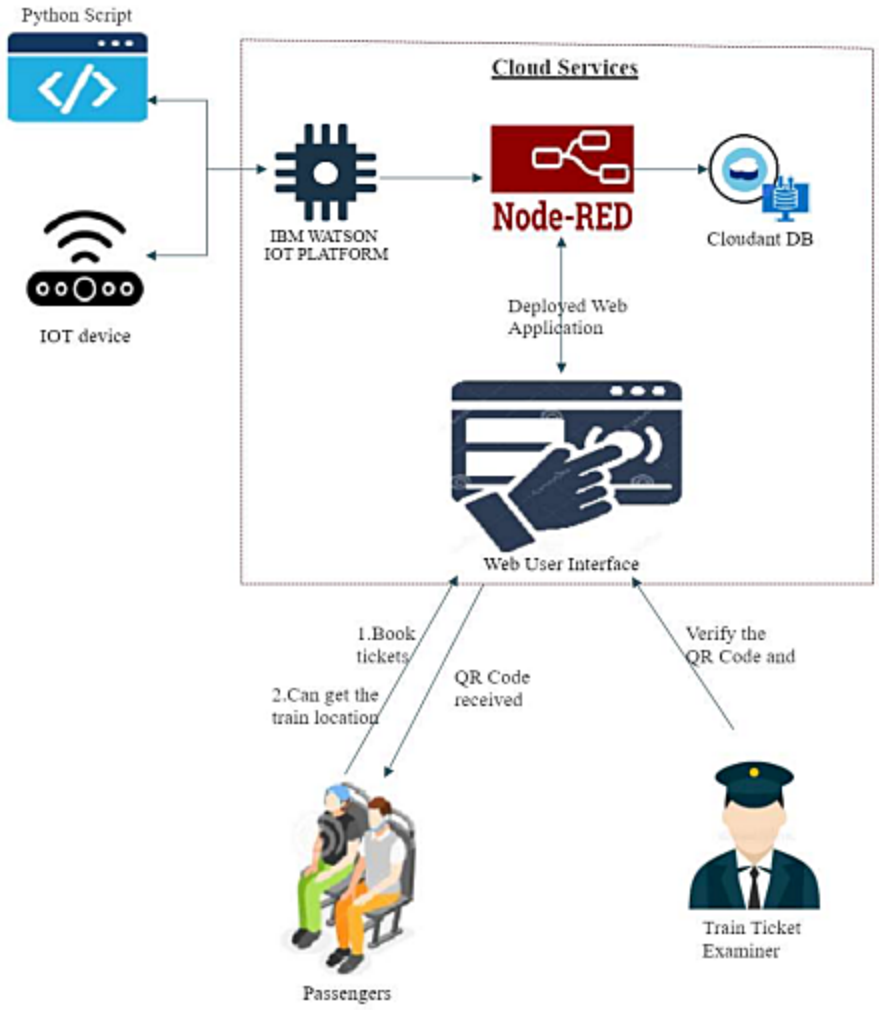
5.1 Data Flow Diagrams

DFD LEVEL 0

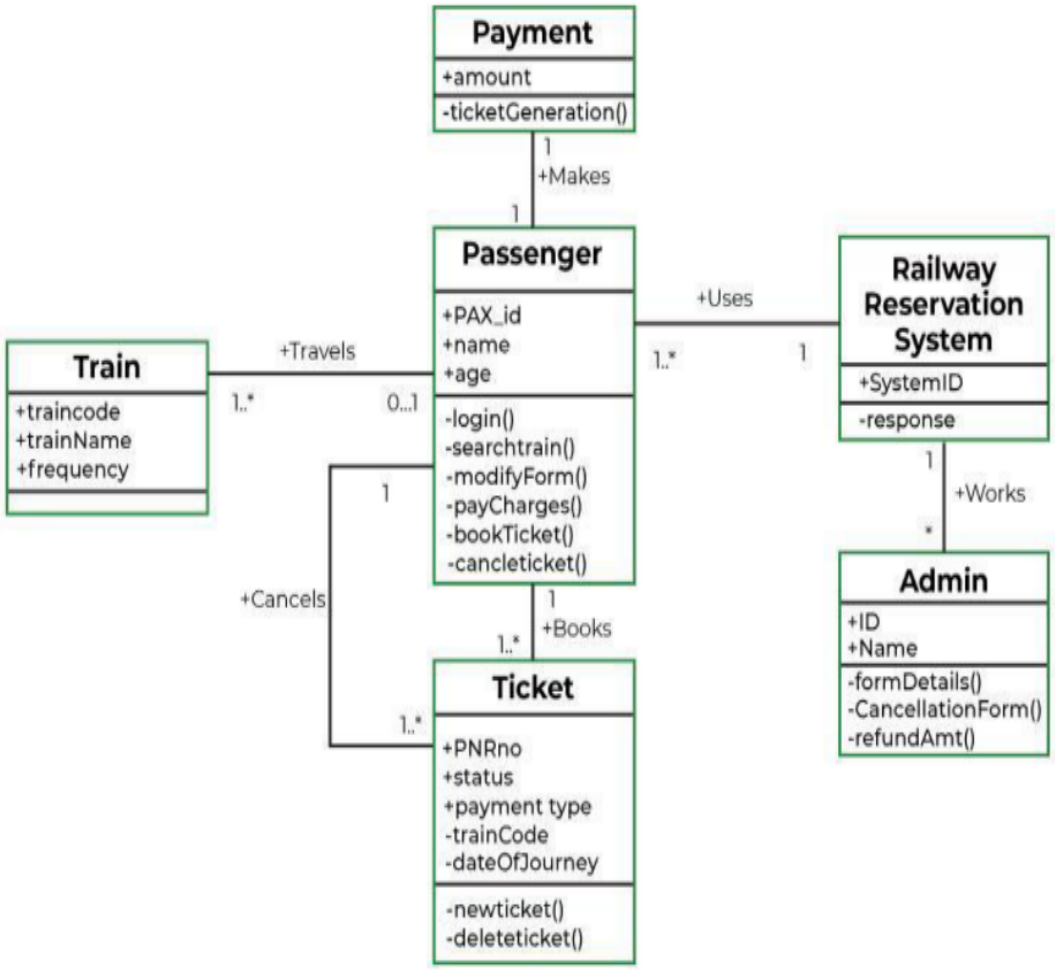


5.2 Solution & Technical Architecture

SOLUTION ARCHITECTURE



TECHNICAL ARCHITECTURE



5.3 User Stories

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
Customer (Mobile user)	Registration	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
Customer (Mobile user)	Registration	USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
Customer (Mobile user)	Registration	USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard with gmail login	Medium	Sprint-1

Customer (Mobile user)	Login	USN-5	As a user, I can log into the application by entering email & password	I can access the website by entering the valid email & password	High	Sprint-1
Customer (Mobile user)	Dashboard	USN-6	As a user, I can see the elements in the dashboard after logging into the application.	I can check for ticket status and generate QR code in dashboard	High	Sprint-2
Customer (Web user)	Ticket Booking	USN-7	As a user, I need to enter details and click the book ticket button.	I can access the generated QR code after reserving the ticket	Medium	Sprint-3
Customer Care Executive	Provide services	USN-8	As a customer care executive, I need to provide necessary services as per customer requests	I can access the customer's site to check for the issues	Medium	Sprint-4
Administrator	Access storage	USN-9	As a admin, I can manage the cloud and database	I can add, delete or update user's data in the database.	High	Sprint-2

6. PROJECT PLANNING & SCHEDULING

1. **Sprint Planning & Estimation**

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Dashboard	USN-1	As a user, I will furnish the details for the journey and disclosing the necessary details and book my trip.	15	High	Kavimathi. A, Nivedha.J, Sneha.E, Sandhiya Lakshmi M.
Sprint-1		USN-2	As a user, I will receive a QR code and take a screensho t.	5	High	Kavimathi A, Nivedha.J, Sneha.E, Sandhiya Lakshmi M.
Sprint-2	Live train location	USN-3	As a passenger, I want to know the current location of the train.	20	Medium	Kavimathi A, Nivedha.J, Sneha.E, Sandhiya Lakshmi M.
	Ticket	USN-4	As a TTE, I will cross-	20		Kavimathi A,

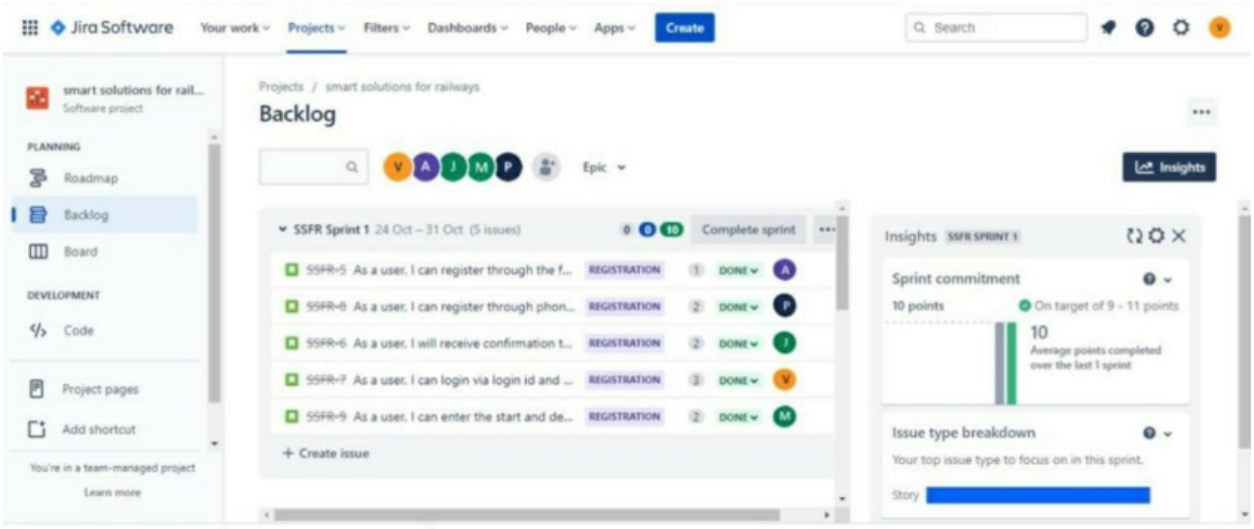
Sprint-3	Verification		check the passenger for their tickets by scanning the QR code		Medium	Nivedha.J, Sneha.E, Sandhiya Lakshmi M.
Sprint-4	Registration	USN-5	As a new user, I can register for the application by entering my email and password .	10	Low	Kavimathi A, Nivedha.J, Sneha.E, Sandhiya Lakshmi M.
Sprint-4	Login	USN-6	As a user, I can log in to the application by entering email & password .	10	Low	Kavimathi A, Nivedha.J, Sneha.E, Sandhiya Lakshmi M.

2. Sprint Delivery Schedule

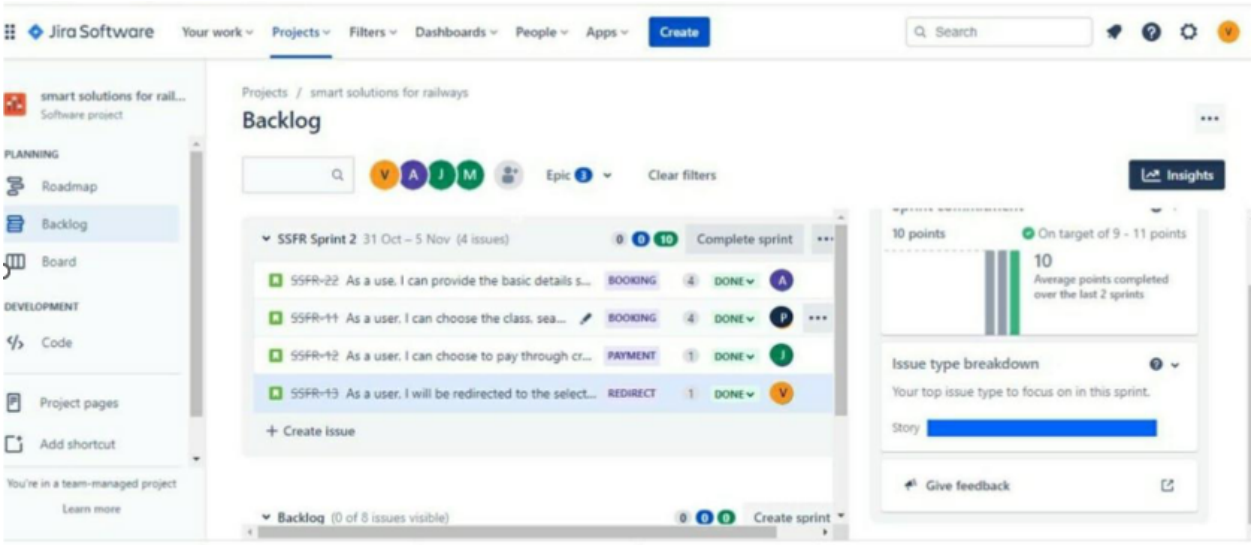
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

3. **Reports from JIRA**

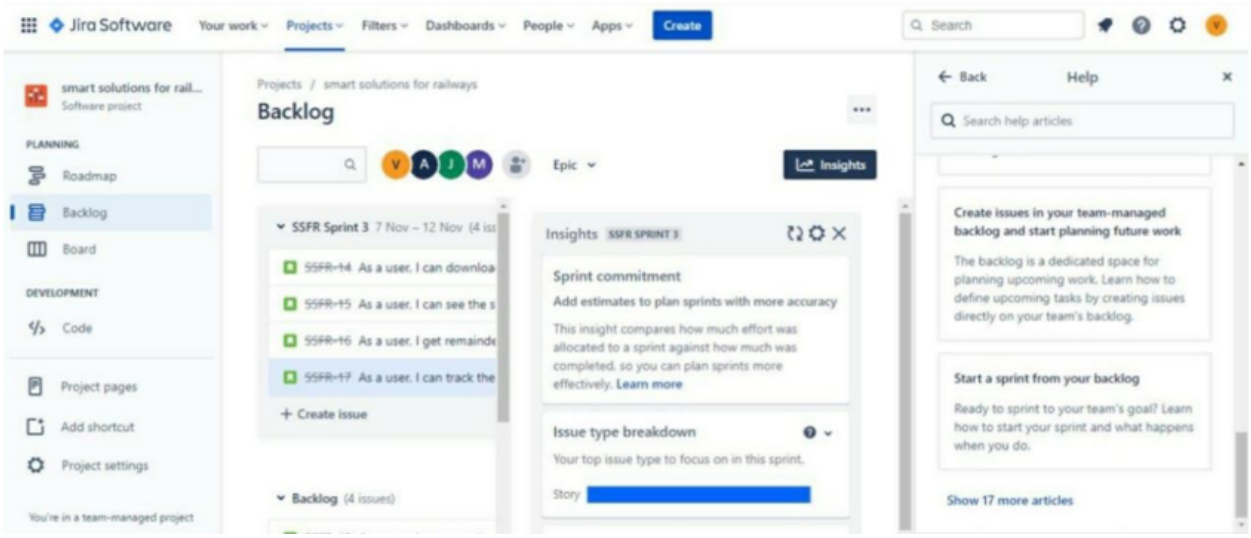
SPRINT 1:



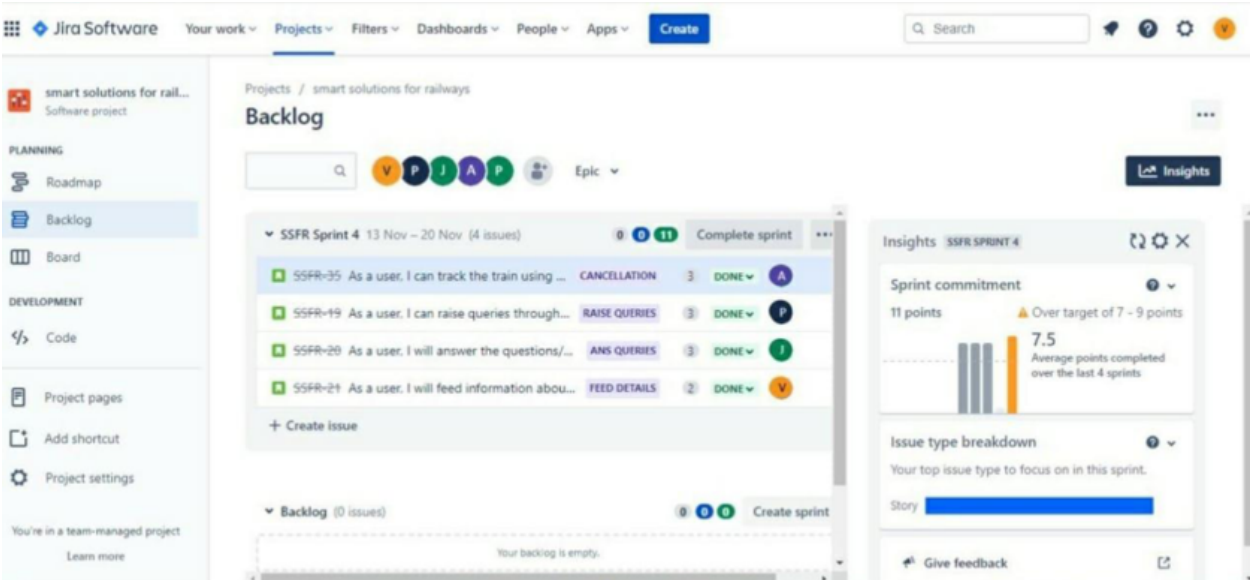
SPRINT 2:



SPRINT 3:



SPRINT 4:



7. CODING & SOLUTIONING

1. FEATURE 1:

Train Ticket booking and QR code generation

Description:

In this part we have designed a webpage using node red to book the train ticket.The user can login in into the webpage using username and password . After successful login, the user wil be redirected to the Ticket booking form. In this form ,users are asked to fill the personal details and the jouney details . After entering the appropriate details the confirmation message is shown and QR code is generated.

Code:

```
import cv2
import numpy as np
import time
import pyzbar.pyzbar as pyzbar
from ibmcloudant.cloudant_v1 import CloudantV1
from ibmcloudant import CouchDbSessionAuthenticator
from ibm_cloud_sdk_core.authenticators import BasicAuthenticator
authenticator = BasicAuthenticator ('apikey-v2-
2stdiumhdpvimevtz15mlesexkro758a3adfu6gzbd2i',
'cf64a065320a016ee67039aa90958562')
service=CloudantV1 (authenticator=authenticator)
service.set_service_url('https://apikey-v2-
2stdiumhdpvimevtz15mlesexkro758a3adfu6gzbd2i:cf64a065320a01
6ee67039aa90958562@03b99f47-5466-4bdd-b1d7-0c89004e3180-
bluemix.cloudantnosqldb.appdomain.cloud')
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("Valid Ticket")
#print (a)
try:
response = service.get_document (db='bookings',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 ()
```

2. FEATURE 2:

TICKET VERIFICATION AND TRAIN TRACKING

DESCRIPTION:

In this we have included a feature to verify the ticket by using the QR code scanner at TTE side and retrieve the passenger details from the Cloudant database to check the details of the passenger . If it is a valid QR code then the ticket is verified. For an invalid QR code the pop up message is displayed " INVALID TICKET" . For Train tracking the current location status of the train is displayed in the webpage . The users can easily know about the arrival and departure of the train .

Code:

```
import wiotp.sdk.device
import time import
random myConfig = {
"identity":{
"orgId":"50flc2",
"typeId":"Train3",
"deviceId":"123456789"
},
"auth":{
"token":"*WwvodeD_Zhc&-!&aB"
}
}
def myCommandCallback(cmd):
print("Message received fromIBM 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':'Mumbai SF Express','lat':11.024938,'lon':76.982315}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':11.220325,'lon':77.570083}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':11.564960,'lon':77.993057}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':11.780142,'lon':78.037002}
pub(myData)
time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.134824,'lon':78.130386}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.489034,'lon':78.009536}
pub(myData)
```

```
time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.655239,'lon':77.866714}
pub(myData)
time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.735622,'lon':77.756851}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.907020,'lon':77.696426}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.665958,'lon':77.136123
}
pub(myData)
time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.548022,'lon':76.921890}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.336809,'lon':76.644485}
pub(myData)
time.sleep(3)
client.commandCallback = myCommandCallback client.disconnect()
```

3.Database Schema

8. TESTING

1.Test Cases

D3: check whether the QR code is generated when the user provides all the necessary details.

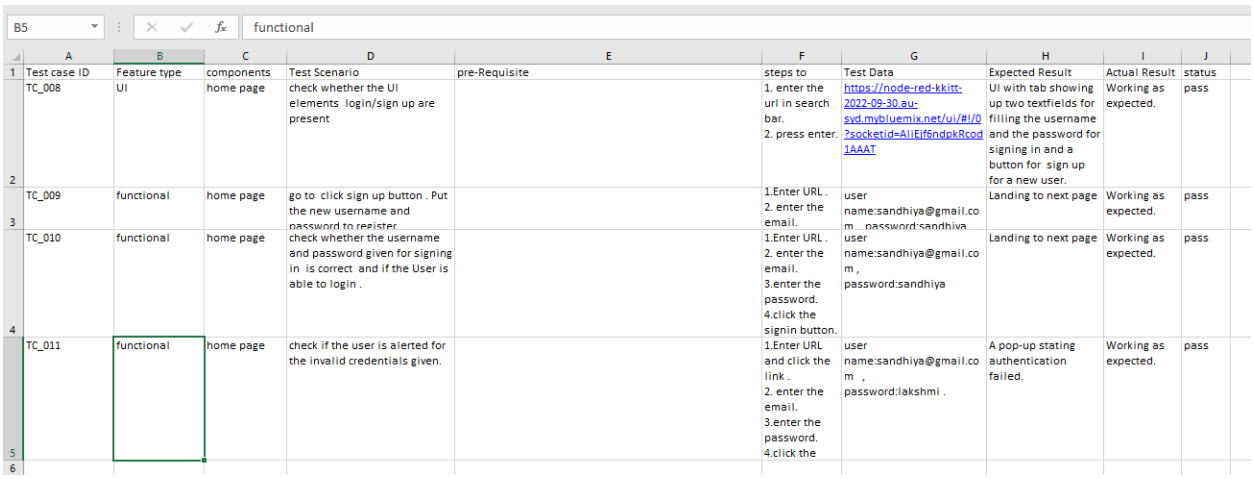
Test case ID	Feature type	components	Test Scenario	pre-Requisite	steps to	Test Data	Expected Result	Actual Result	status
TC_001	UI	booking page	check if the UI elements on landing for booking page is present properly.				UI with tab showing up 3 dropdown list for choosing destination , boarding station and seat . And 3 text fields for filling up name,age and mobile no.	Working as expected.	pass
TC_002	functional	booking page	check whether the QR code is generated when the user provides all the necessary details.		1.Choose the Boarding Station from the droplist. 2. Choose the Destination Station from the droplist. 3. Choose the seat. 4.Enter the name. 5.Enter the age. 6.Enter the mobile no. 7.Click the submit button.	Boarding Station:Hyderabad Destination Station:Chennai seat: 3 name: ria age: 21 mobile no: 9087654321	A QR Code will be presented up in the screen.	Working as expected.	pass
TC_003	functional	booking page	check whether the user receives any confirmation for their reservation if the tickets are booked sucessfully.	Once the user on clicks the submit button in the booking page.			A pop-up text saying the ticket is reserved.	Working as expected.	pass
TC_004	functional	booking page	check whether the user receives any confirmation for their reservation if the tickets are not booked sucessfully.	Once the user on clicks the submit button in the booking page.			A pop-up text saying the ticket is not reserved.	Working as expected.	pass

B2: functional

Test case ID	Feature type	components	Test Scenario	pre-Requisite	steps to	Test Data	Expected Result	Actual Result	status
TC_005	functional	webpage	check whether the location of the selected train is displayed		1. enter the url in search bar. 2. press enter.	https://node-red-kkitt-2022-09-30.au-syd.mybluemix.net/worldmap	showing the loction of the train .	Working as expected.	pass

A3: TC_007

Test case ID	Feature type	components	Test Scenario	pre-Requisite	steps to	Test Data	Expected Result	Actual Result	status
TC_006	functional	frame	check whether the TTE is intimated properly when the correct Qrcode is scanned up.		run the QRCodeScanner.py	Any valid Qrcode.	Alerting the text as ticket on the frame and and displaying the booking details of the passenger as output to the	Working as expected.	pass
TC_007	functional	frame	check whether the TTE is intimated accordingly when the incorrect Qrcode is scanned up.		run the QRCodeScanner.py	Any invalid Qrcode.	Alerting the text as not a ticket on the frame and and displaying not a valid ticket as output	Working as expected.	pass



	A	B	C	D	E	F	G	H	I	J
1	Test case ID	Feature type	components	Test Scenario	pre-Requisite	steps to	Test Data	Expected Result	Actual Result	status
2	TC_008	UI	home page	check whether the UI elements login/sign up are present		1. enter the url in search bar. 2. press enter.	https://node-red-kkitt-2022-09-30.au-syd.mybluemix.net/ui/#!/0?socketid=AliEjf6ndpkRcod1AAAT	UI with tab showing up two textfields for filling the username and the password for signing in and a button for sign up for a new user.	Working as expected.	pass
3	TC_009	functional	home page	go to click sign up button . Put the new username and password to register		1.Enter URL . 2. enter the email.	user name:sandhiya@gmail.com password:sandhiya	Landing to next page	Working as expected.	pass
4	TC_010	functional	home page	check whether the username and password given for signing in is correct and if the User is able to login .		1.Enter URL . 2. enter the email. 3.enter the password. 4.click the signin button.	user name:sandhiya@gmail.com , password:sandhiya	Landing to next page	Working as expected.	pass
5	TC_011	functional	home page	check if the user is alerted for the invalid credentials given.		1.Enter URL and click the link . 2. enter the email. 3.enter the password. 4.click the	user name:sandhiya@gmail.com , password:lakshmi .	A pop-up stating authentication failed.	Working as expected.	pass
6										

2. User Acceptance Testing

1. PURPOSE OF DOCUMENT

The purpose of this document is to briefly explain the test coverage and open issues of the project at the time of the release to User Acceptance Testing (UAT)

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level ,and how they was resolved.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	2	3	1	11
External	2	1	0	0	3
Fixed	9	4	5	2	20
Not Reproduced	0	0	1	0	1
Skipped	0	1	0	2	3
Won't Fix	1	0	1	0	2
Duplicate	1	1	0	0	2
Totals	18	0	10	5	42

The defect analysis was resolved by:

1. **Debugging window**
2. **Defect resolution process**
3. **Prioritize and resolving defect**
4. **Reviewing the code and establishing checkpoints.**

3. **Test Case analysis**

Section	Test Cases	Not Tested	Fail	Pass
Home page	3	0	0	3
Login page	4	0	0	4
Booking	10	0	0	10
Passenger Details	6	0	0	6
TTE	3	0	0	3
Train Tracking	2	0	0	2

9. RESULTS

1.Performance Metrics

The performance of smart solutions for railways are Scalability ,Speed, Stability , Reliability ,Security ,Maintainability and code quality.

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES :

☆ Work load of the user is greatly reduced.
☆ Passenger's can book tickets from their convenient place and time
☆ TTE can easily verify tickets without the need to carry several documents pertaining to passenger details.
☆ Since cloudant database retrieval is possible only for the TTE (Admin) secure data handling is made possible.
☆ Passenger's can track the location of the train which gives them more flexibility in planning their schedules. ☆ Cost of implementation is less.

DISADVANTAGES:

☆ Since the solution is built on a Web UI which requires internet facility , this may serve to be an issue.
☆ Network errors can cause serious issues while ticket booking and verification.
☆ System compatibility to adapt the user interface may limit booking facility .
☆ Tracking of trains may be an issue at unlocalized terrains.

11. Conclusion

This project provides smart solutions to bring innovations in the railways. Thus, the reservation system and timeliness of the train tracking facilities has given highly flexible opportunities for the passengers at their own comfort and offers automated services for the railways system to validate the tickets thereby turning it efficient.

12. Future Scope

The further development, study and research will enable us to add newer, fresher and advanced utilities in our project .Due to scalability made possible in the web development, there is a huge scope for development in the future.India is projected to account for 40% of the total global share of rail activity by 2050. This allows great potential for the online ticket booking platform industry.

13. APPENDIX

Source Code

GPS TRACKING CODE:

ChennaiExpress.py

```
import wiotp.sdk.device
 import random
 import time
 myConfig =
 { "identity" :{
"orgId":"50flc2",
 "typeId":"Train1",
 "deviceId":"13579" },
"auth":{
 "token":"uamQOww30eBju!LE)F"
}
 }
def myCommandCallback(cmd):
 print("Message received fromIBM 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':'Chennai Express','lat':13.344279,'lon':80.214367}
pub(myData)
time.sleep(3)
myData = {'name':'Chennai Express','lat':13.515254,'lon':80.093518}
 pub(myData)
time.sleep(3)
myData = {'name':'Chennai Express','lat':13.728799,'lon':80.005627}
 pub(myData)
 time.sleep(3)
```

```
 myData = {'name':'Chennai Express','lat':13.910160,'lon':79.906750}
 pub(myData)
 time.sleep(3)
myData = {'name':'Chennai Express','lat':14.102035,'lon':79.851819}
pub(myData)
time.sleep(3)
myData = {'name':'Chennai Express','lat':14.261807,'lon':79.862805}
pub(myData)
time.sleep(3)
myData = {'name':'Chennai Express','lat':14.623537,'lon':79.950695}
 pub(myData)
time.sleep(3)
 myData = {'name':'Chennai Express','lat':15.111987,'lon':79.994641}
pub(myData)
time.sleep(3)
 myData = {'name':'Chennai Express','lat':15.313413,'lon':80.005627}
pub(myData)
time.sleep(3)
 myData = {'name':'Chennai Express','lat':15.567568,'lon':80.104504}
 pub(myData)
 time.sleep(3)
 myData = {'name':'Chennai Express','lat':15.747405,'lon':80.269299 }
pub(myData)
time.sleep(3)
myData = {'name':'Chennai Express','lat':15.821409,'lon':80.302258}
pub(myData)
 time.sleep(3)
myData = {'name':'Chennai Express','lat':15.927082,'lon':80.445080}
pub(myData)
time.sleep(3)
myData = {'name':'Chennai Express','lat':16.022141,'lon':80.554943}
pub(myData)
 time.sleep(3)
myData = {'name':'Chennai Express','lat':17.033801,'lon':80.295512}
pub(myData)
 time.sleep(3)
myData = {'name':'Chennai Express','lat':18.383088,'lon':18.383088}
pub(myData)
 time.sleep(3)
myData = {'name':'Chennai Express','lat':19.074762,'lon':79.487698}
pub(myData)
```

```
time.sleep(3)
myData = {'name':'Chennai Express','lat':20.179065,'lon':79.001439}
pub(myData)
 time.sleep(3)
myData = {'name':'Chennai Express','lat':21.306421,'lon':78.789356}
pub(myData)
 time.sleep(3)
myData = {'name':'Chennai Express','lat':22.518024,'lon':77.829404}
pub(myData)
time.sleep(3)
myData = {'name':'Chennai Express','lat':23.264139,'lon':77.429333}
pub(myData)
 time.sleep(3)
myData = {'name':'Chennai Express','lat':24.509723,'lon':78.330212}
pub(myData)
time.sleep(3)
myData = {'name':'Chennai Express','lat':25.668840,'lon':78.451062}
pub(myData)
time.sleep(3)
myData = {'name':'Chennai Express','lat':26.177704,'lon':78.170910}
pub(myData)
 time.sleep(3)
myData = {'name':'Chennai Express','lat':27.505914,'lon':77.676526}
pub(myData)
 time.sleep(3)
myData = {'name':'Chennai Express','lat':28.302041,'lon':77.308484}
pub(myData)
 time.sleep(3)
client.commandCallback = myCommandCallback client.disconnect()
```

MumbaiExpress.py

```
import wiotp.sdk.device
import time import
random myConfig = {
"identity":{
"orgId":"50flc2",
"typeId":"Train3",
"deviceId":"123456789"
```

```
},
"auth":{
"token":"*WwvodeD_Zhc&-!&aB"
}
}
def myCommandCallback(cmd):
print("Message received fromIBM 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':'Mumbai SF Express','lat':11.024938,'lon':76.982315}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':11.220325,'lon':77.570083}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':11.564960,'lon':77.993057}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':11.780142,'lon':78.037002}
pub(myData)
time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.134824,'lon':78.130386}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.226105,'lon':78.091934}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.344187,'lon':78.037002}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.489034,'lon':78.009536}
pub(myData)
time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.655239,'lon':77.866714}
pub(myData)
```

```
time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.735622,'lon':77.756851}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.907020,'lon':77.696426}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.987323,'lon':77.646988}
pub(myData)
time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.955205,'lon':77.509659}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.665958,'lon':77.136123
}
pub(myData)
time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.548022,'lon':76.921890}
pub(myData)
 time.sleep(3)
myData = {'name':'Mumbai SF Express','lat':12.336809,'lon':76.644485}
pub(myData)
time.sleep(3)
client.commandCallback = myCommandCallback client.disconnect()
```

BangaloreExpress.py

```
import wiotp.sdk.device
import time import
random myConfig
= {
"identity" :{
"orgId":"50flc2",
"typeId":"Train2",
"deviceId":"02468"
},
"auth":{
"token":"NfVlk+XD?APWdBWucS"
}
}
```

```
def myCommandCallback(cmd):
print("Message received fromIBM 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':'Bangalore Express','lat':11.688572, 'lon':78.098877}
pub(myData)
time.sleep(3)
myData = {'name':'Bangalore Express','lat':11.711433, 'lon':78.076905}
pub(myData)
time.sleep(3)
myData = {'name':'Bangalore Express','lat':11.978226, 'lon':78.116730}
pub(myData)
time.sleep(3)
myData = {'name':'Bangalore Express','lat':12.085676, 'lon': 78.119477}
pub(myData)
time.sleep(3)
myData = {'name':'Bangalore Express','lat':12.402400, 'lon':78.023347}
pub(myData)
 time.sleep(3)
myData = {'name':'Bangalore Express','lat':12.884795, 'lon':77.707490}
pub(myData)
time.sleep(3)
myData = {'name':'Bangalore Express','lat':13.018630,'lon':77.614106}
pub(myData)
 time.sleep(3)
myData = {'name':'Bangalore Express','lat':13.334194, 'lon':77.086762}
pub(myData)
time.sleep(3)
time.sleep(3)
myData = {'name':'Bangalore Express','lat':13.299448, 'lon':76.858796}
pub(myData)
myData = {'name':'Bangalore Express','lat':13.344884,'lon': 76.205109}
pub(myData)
 time.sleep(3)
myData = {'name':'Bangalore Express','lat':13.619985, 'lon':75.966157}
```

```
pub(myData)
 time.sleep(3)
myData = {'name':'Bangalore Express','lat':13.974739, 'lon':76.119965}
pub(myData)
time.sleep(3)
myData = {'name':'Bangalore Express','lat':14.423398, 'lon':75.949677}
pub(myData)
time.sleep(3)
myData = {'name':'Bangalore Express','lat':14.922914, 'lon':75.389374}
pub(myData)
time.sleep(3)
myData = {'name':'Bangalore Express','lat':15.119216, 'lon':75.389374}
pub(myData)
time.sleep(3)
time.sleep(3)
myData = {'name':'Bangalore Express','lat':15.449980, 'lon':74.406230}
pub(myData)
 time.sleep(3)
myData = {'name':'Bangalore Express','lat':15.352006,'lon':74.307353}
pub(myData)
myData = {'name':'Bangalore Express','lat':15.314922, 'lon':74.218089}
pub(myData)
time.sleep(3)
myData = {'name':'Bangalore Express','lat':15.283131, 'lon':74.146678}
pub(myData)
time.sleep(3)
myData = {'name':'Bangalore Express','lat':15.276839, 'lon':74.129855}
pub(myData)
 time.sleep(3)
 time.sleep(3)
myData = {'name':'Bangalore Express','lat':15.282800, 'lon':74.125392}
pub(myData)
time.sleep(3)
 time.sleep(3)
myData = {'name':'Bangalore Express','lat':15.296378,'lon':74.135692}
pub(myData)
time.sleep(3)
client.commandCallback = myCommandCallback client.disconnect()
time.sleep(3)
```

QR CODE GENERATION CODE

```
import cv2
import numpy as np
import time
import pyzbar.pyzbar as pyzbar
from ibmcloudant.cloudant_v1 import CloudantV1
from ibmcloudant import CouchDbSessionAuthenticator
from ibm_cloud_sdk_core.authenticators import BasicAuthenticator
authenticator = BasicAuthenticator ('apikey-v2-
2stdiumhdpvimevtz15mlesexkro758a3adfu6gzbd2i',
'cf64a065320a016ee67039aa90958562')
service=CloudantV1 (authenticator=authenticator)
service.set_service_url('https://apikey-v2-
2stdiumhdpvimevtz15mlesexkro758a3adfu6gzbd2i:cf64a065320a01
6ee67039aa90958562@03b99f47-5466-4bdd-b1d7-0c89004e3180-
bluemix.cloudantnosqldb.appdomain.cloud')
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("Valid Ticket")
#print (a)
try:
response = service.get_document (db='bookings',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 ()
```

GitHub & Project Demo Link

GITHUB LINK: https://github.com/IBM-EPBL/IBM-Project-3059-1658498851

DEMO LINK: .https://youtu.be/SJkF0DIjt00