**Metro Railway Reservation Management**

Team members:-

A.Venkata DineshReddy(RA1911028010098)

CH.Vamshi Krishna(RA1911028010099)

K.Spurgeon(RA1911028010109)

Introduction:

It is difficult to buy tickets manually in this busy world. This system is a web based application which provides information regarding reservation booking. This system also manages public feedback about services through its complaint management system. It also contains an user module where users can recharge their railway wallet online through this site and can buy the metro tickets in the form of QR-code which will be saved in user’s smart phone in an encrypted form. The information about a particular user is stored in cloud database for continuous and easy availability reached. For security reasons the information about every user is stored in cloud which is to be accessed for each ticket booking for validation purpose.

PROBLEM STATEMENT:-

To design and develop an application for maintaining the details of the user, QR-Code generation and validation of the ticket for getting reservation.

OBJECTIVES:-

• To provide the Smart way to scan the ticket through QR code.

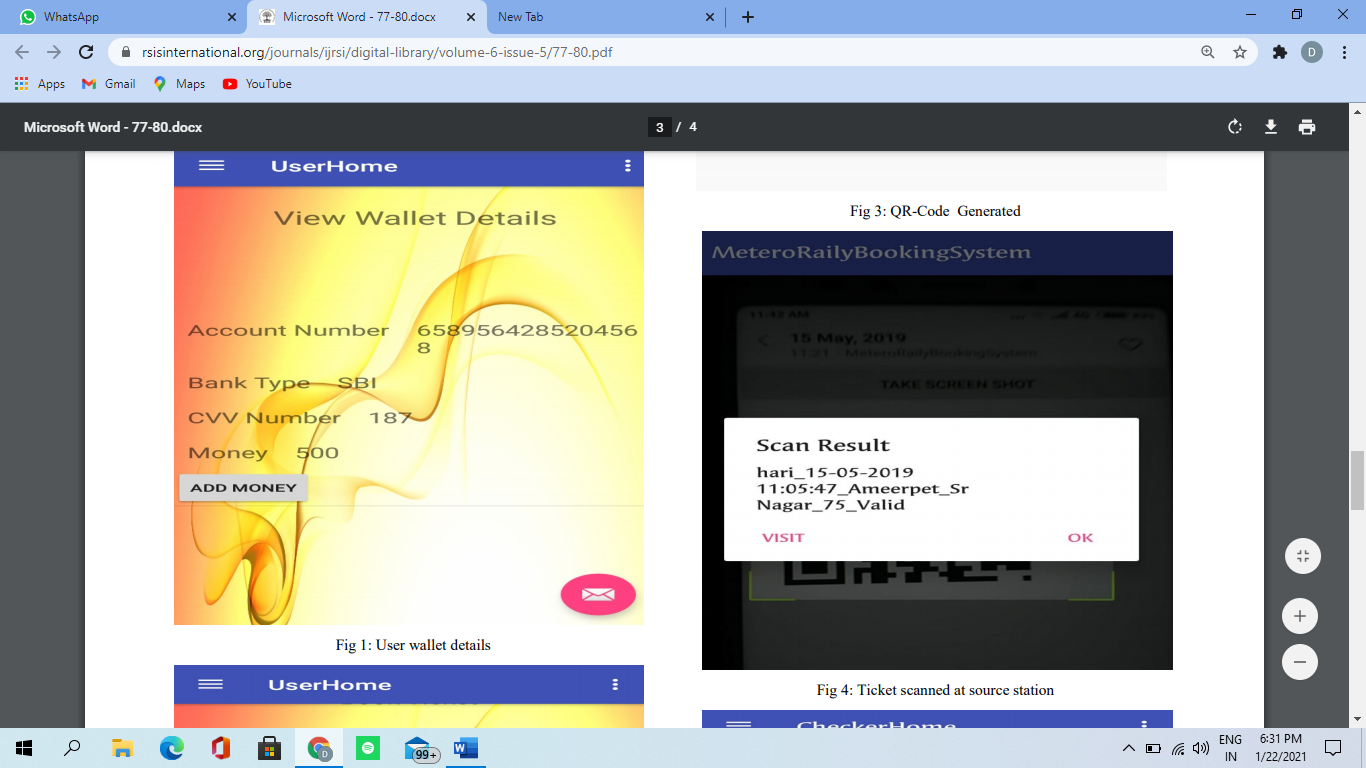
• To provide Check-in, Check-out and on-spot booking procedure

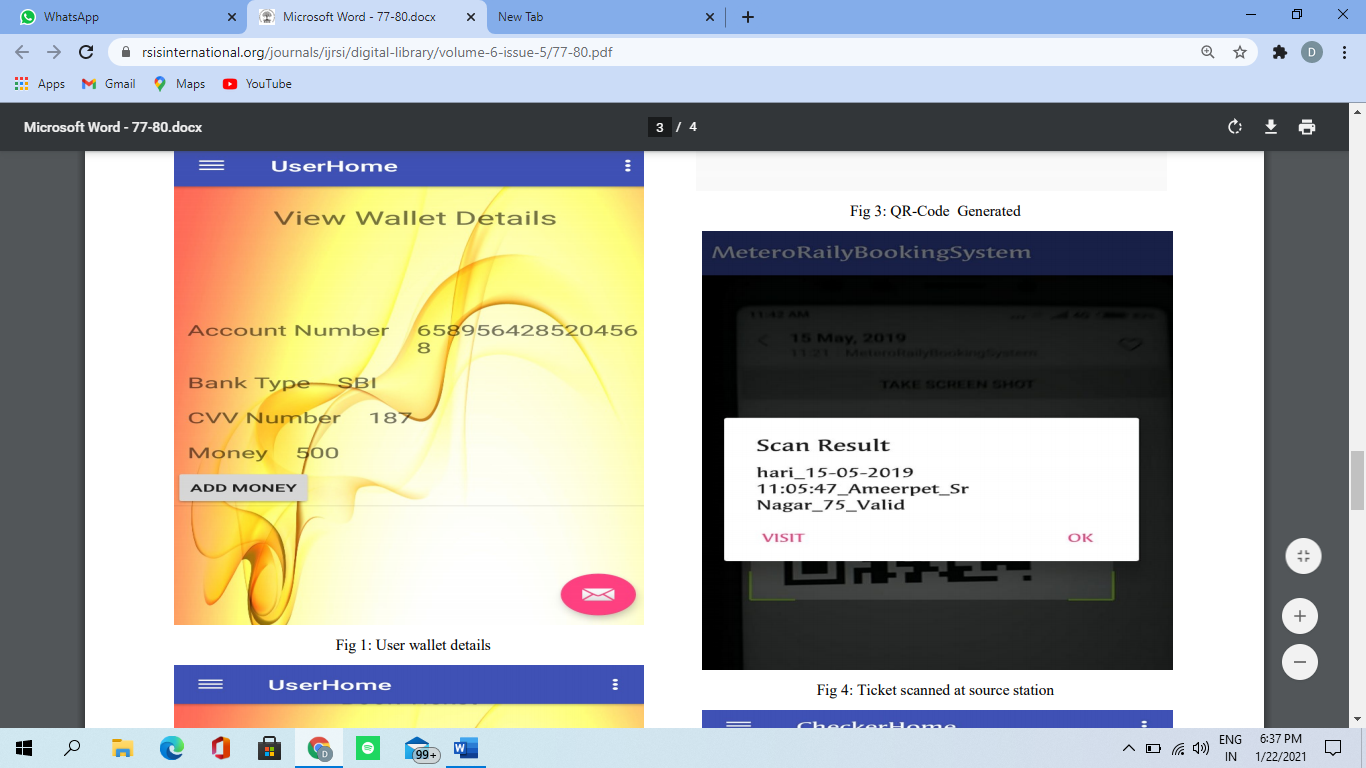
PROPOSED METHODOLOGY :-

This system is actually a cloud based application. The data will be saved in the cloud database. Initially, the user has to install the android application where the he has to enter his details where adhaar number is mandatory. During signup, the user has to enter his details like adhaar number, username, mobile number etc which will be stored in the firebase. When the user want to buy the ticket he can enter username and password which is sent to the database for security purpose if it is valid then ticket is generated accordingly. If the user lost his phone he can open it in others phones and is not restricted only to his phone. When the user enters his details it will be sent to the firebase with the help of database.

IMPLEMENTATION:-

For booking the metro ticket the user has to register by entering his details like name, adhaar number etc. After registering the user can login, if he enters invalid password /username then he will not be able to book his ticket, but if it is valid, then he can book his ticket. The ticket is generated in the form of QR-Code and ticket checker scans the ticket by QR-Code Reader to validate the ticket.



Working:-

1. Personal Information Gathering: The work here starts with installing our application. It gathers the basic information about the user like Adhaar number, Name, mobile number, etc and these details are stored in the firebase. When the user wants to buy the ticket just he need to enter his username and password which is sent to firebase for security purpose.
2. Ticket Buying: Once the user has successfully login, then he can book the ticket by selecting the source station, destination station and number of tickets required. The cost of booked tickets will be displayed below when user click on submit button the amount will be deducted from user railway wallet and ticket will be generated.
3. Generating Quick Response Code: When the user books the ticket then the information is sent to firebase and the QR-Code is generated on the server side and sent back to the application. The QR-Code contains the ticket number, username, source and destination stations.
4. GPS Ticket Validation: Here the GPS plays the role of ticket checker. It contains details like when the user buys ticket, source geo points, destination geo points, number of tickets, date, expiry time everything will be stored in firebase. The service checks the user location with destination geo points. When the ticket is rescanned then the ticket will get expired and ticket will be valid only for one day within the metro timings.
5. Checking QR-Code with QR-Reader: In this module, the checker will be having the QR-Code Reader and scans the QR-Code in the application in order to validate the ticket. If the user has exceeded the destination station then amount will be deducted from his wallet. Otherwise, if user has dropped in between station then amount will be refunded back to his account.

ADVANTAGES:-

Today our country is overpopulated. we can see that every public place is over crowded. Same way railway station and trains are also over crowded. We cannot find even place to stand in the train in general compartment. So it is not possible to undertake journey without proper seat or berth. if duration of journey is long then it becomes very difficult and we cannot think of journey by train without reservation.

So railway reservation is very useful Even becomes very necessary. Therefore we must get our reservation in advance so that our purpose can be solved. So it has very good advantage because in case of emergency or in acute requirements we can book a ticket and can start journey.

FUTURE SCOPE :-

There is always scope for innovation when it comes for technology. Even our project is no exception. Some improvements has to be made in the future. And making use of GPS to track down the location of the passenger and the nearest station.

Problem statement:

To design and develop an application for maintaining the details of the user, QR-Code generation and validation of the ticket for getting reservation.

Requirements:

Though the requirements are simple but definitely user needs a printer , desktop, sufficient storage and security.

Assumptions:

Since application is internet based , the user need a basic knowledge of choosing correct destination and should have the knowledge of making payment.

Updating of data may take some time.

Recovery of previous data depends on user.

Risks:

* Careful while doing payment
* Selecting of correct destination

**IMPLEMENTATION AND GOVERNANCE:**

Required skills:

|  |  |
| --- | --- |
| Skills | Info |
| UX designer | Designing experience of user |
| Backend developer | Design services / API |
| Testing | Developing test cases |
| Project management | Planning, scheduling, executing, monitoring and controlling |
| Deployment team | Responsible for changes and updating of application |

Project team structure and Roles and Responsibilities:

|  |  |  |
| --- | --- | --- |
| Project role | Responsibilities | Assigned to |
| Project steering committee | Understanding whole project and analyzing | Review the progress and performance of project |
| Project sponsor | Evaluating the project actuals progress | Managing the sponsorship |
| Project manager | Planning,organizing and directing the completion of specific projects. | Organising and tracking the work |
| Technical lead | Overseas the complete company’s technical team and all risks of the project. | Leading developer and tester to build the application |
| Business analyst | Creating a detailed businessanalysis,outlining problems | Understands the needs and problems of clients |
| Developer | Writing and implementing efficient code. | Developing the application |
| tester | Executes all levels of testing. | Creating test plan and develop test cases |

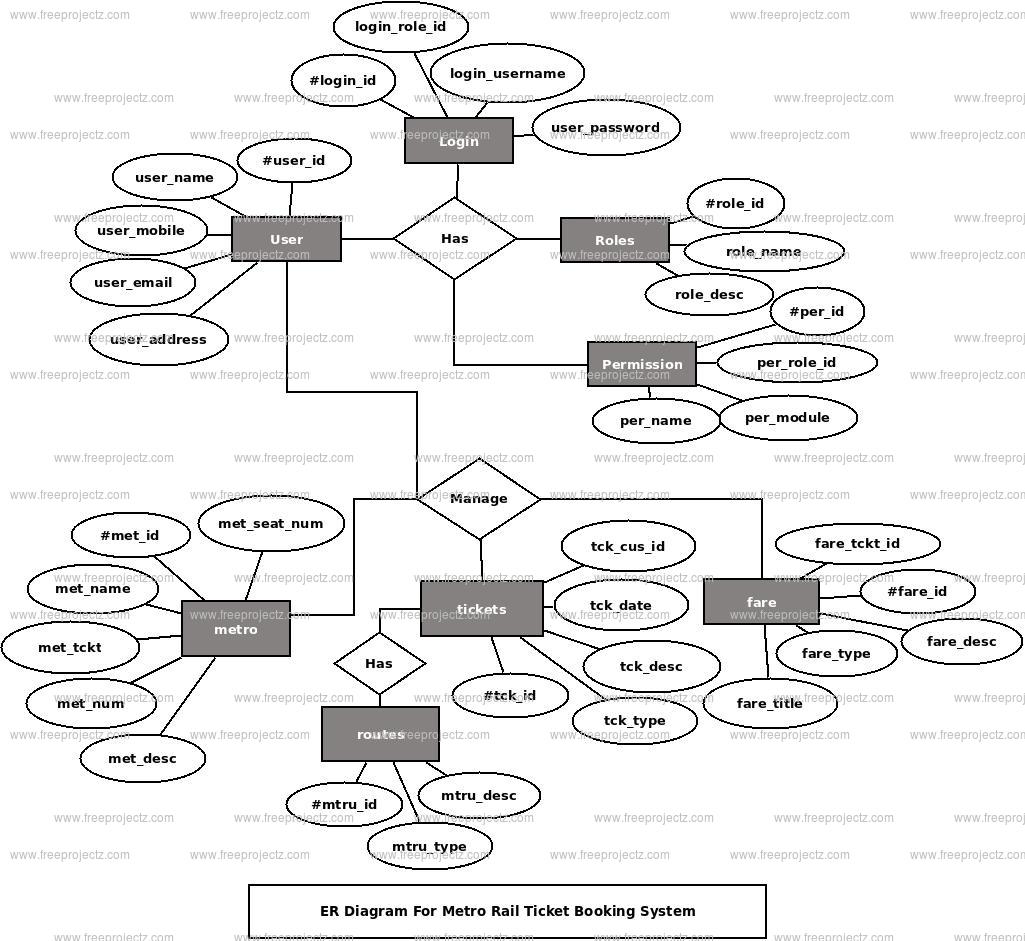
Use Case Diagram:-

A use case diagram is a dynamic or behaviour diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. The "actors" are people or entities operating under defined roles within the system.



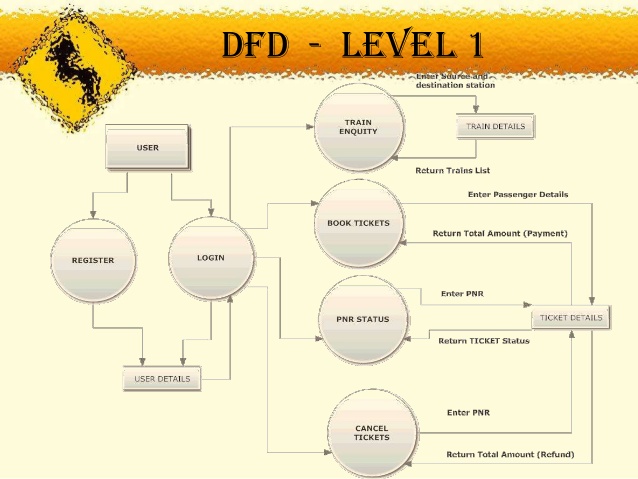
ER Diagram:-

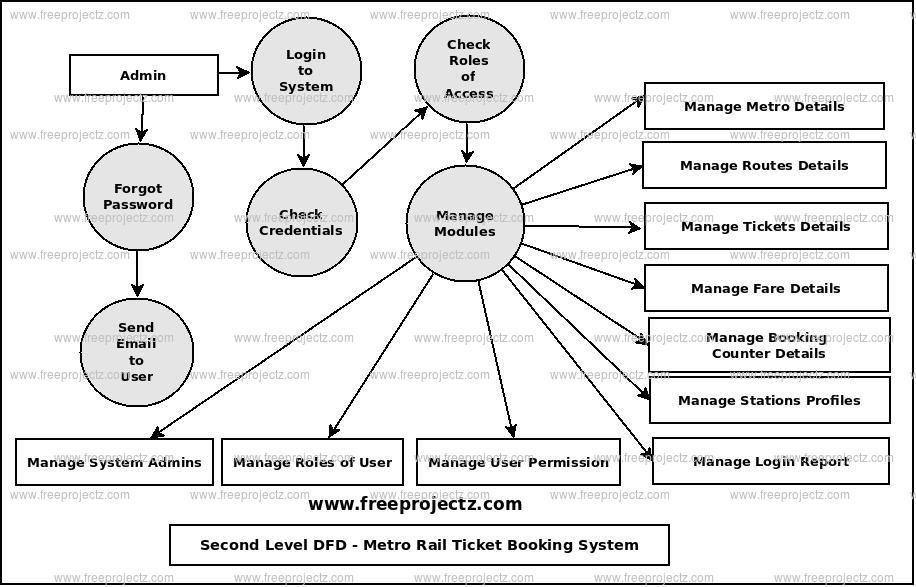
An entity relationship diagram (ERD) is a representation of data within a domain. It consists of entities as well as relationships between entities.Every entity must have at least one attribute that can be used to uniquely identify the entity, which is known as the entity's primary key.



DFD Diagram:-

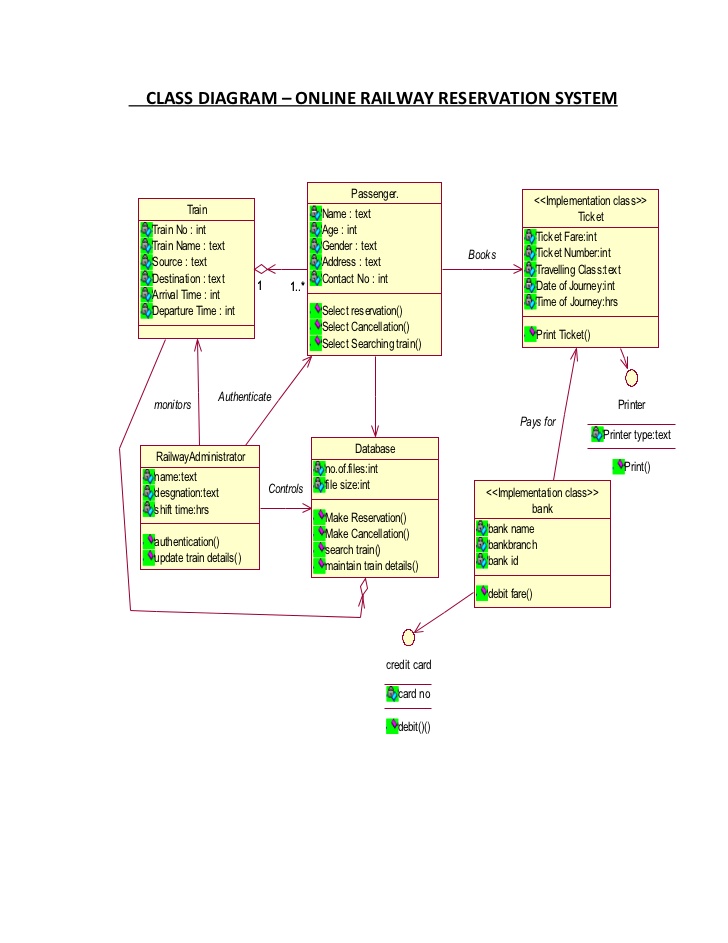
A data-flow diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself.When using UML, the activity diagram typically takes over the role of the data-flow diagram.



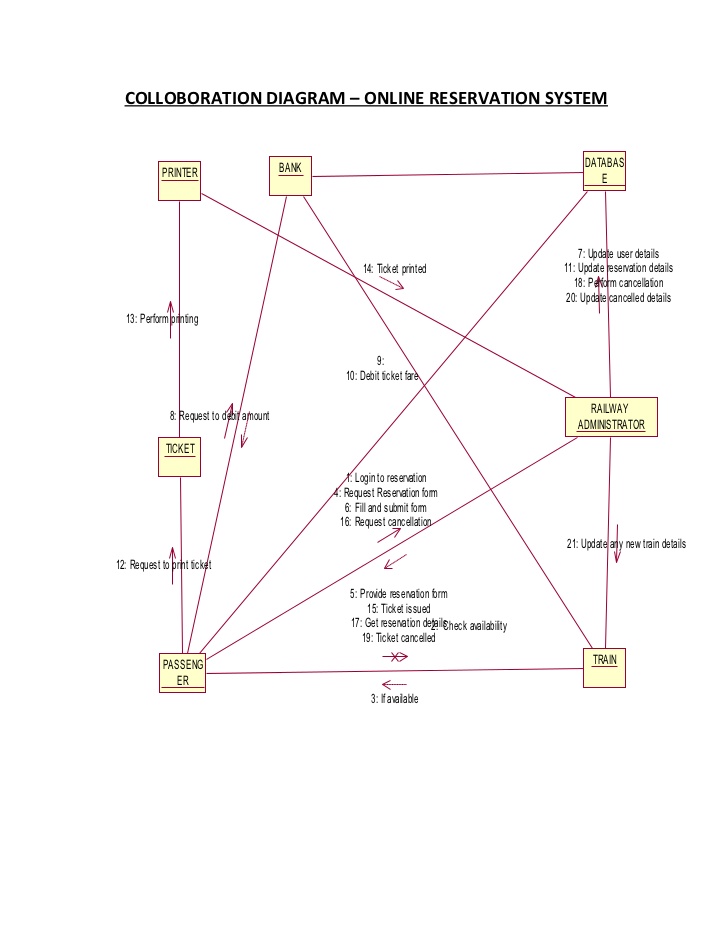


Class Diagram:-

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's: classes, their attributes, operations (or methods), and the relationships among objects.

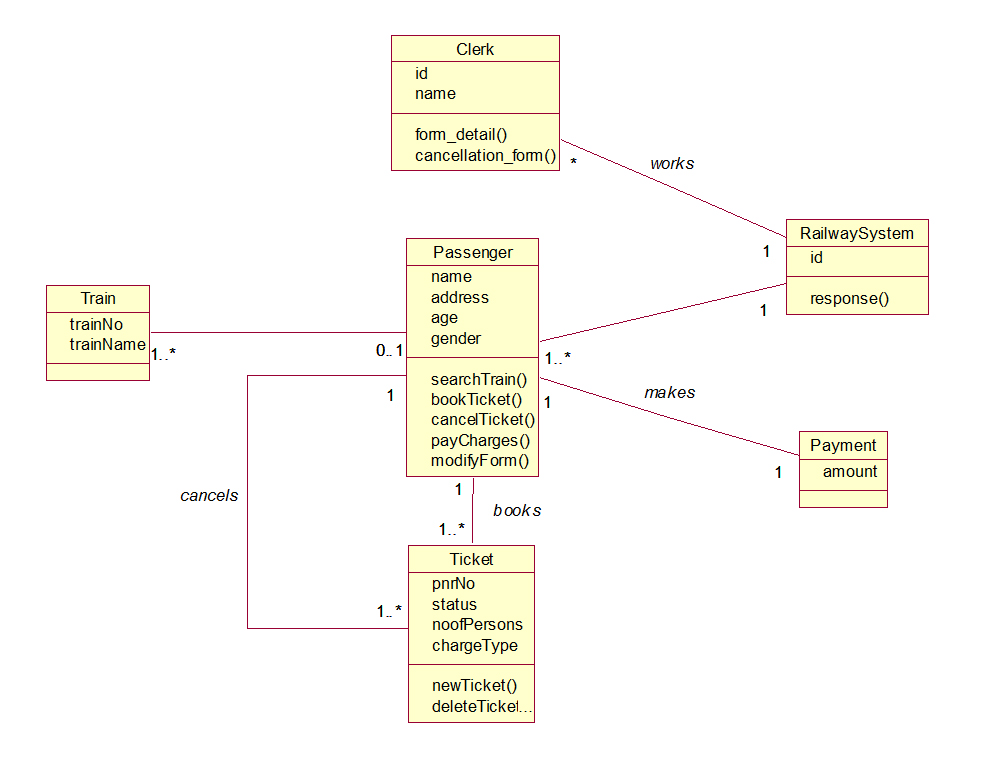


Collaboration Diagram:-Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case.  They are the primary source of information used to determining class responsibilities and interfaces.



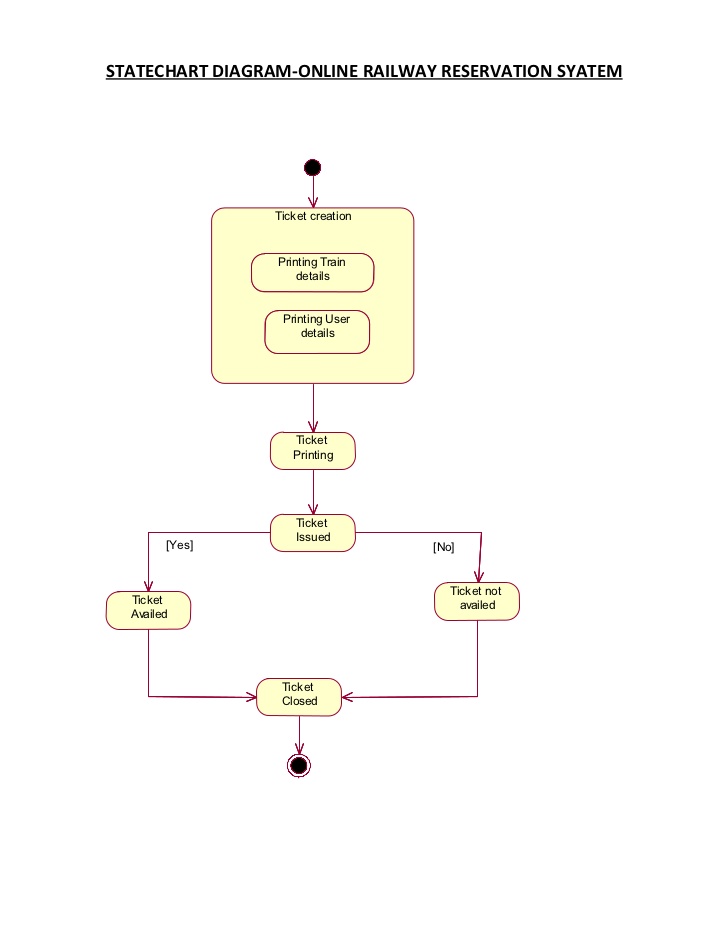
Architecture Diagram :-

An architectural diagram is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. It is an important tool as it provides an overall view of the physical deployment of the software system and its evolution roadmap.

[](http://www.startertutorials.com/uml/wp-content/uploads/2013/10/RRS-Class-Diagram.jpg)

State Diagram:-

A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. State diagrams are used to give an abstract description of the [behavior](https://en.wikipedia.org/wiki/Behavior) of a [system](https://en.wikipedia.org/wiki/System). This behavior is analyzed and represented by a series of events that can occur in one or more possible states. Here each diagram usually represents objects of a single class and track the different states of its objects through the system



Deployment Diagram:-

UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They capture the hardware that will be used to implement the system and the links between different items of hardware. It is a model of physical hardware elements and the communication paths between them

13
3.8 DEPLOYMENT DIAGRAM:
Deployment diagrams are a set of nodes and their relationships. These nodes are
physical entiti...

Front end design:-

The frontend of a software program or website is everything with which the user interacts. Websites must work well on multiple devices and screen sizes, which is why modern web development typically involves responsive design

Softwareused:Python(Tkinter)

Code:-

from tkinter import \*

from tkinter import messagebox

import os

import pyqrcode

window=Tk()

window.title("Login")

window.geometry("1450x750")

window.config(bg="light green")

def con(a):

a.config(bg="light green")

def b(j):

j.config(bg="light grey")

login=Label(window,text="Metro Railway Reservation Management",font=(10),width=40)

login.place(x=525,y=180)

login.config(bg="Violet")

user=Label(window,text="Username")

user.place(x=560,y=220,width=100)

con(user)

pw=Label(window,text="Password")

pw.place(x=560,y=240,width=100)

con(pw)

u=Entry(window,width=50)

u.place(x=650,y=220,width=100)

b(u)

p=Entry(window,show="\*",width=50)

p.place(x=650,y=245,width=100)

b(p)

def check():

user=u.get()

passwd=p.get()

if(user == "DineshReddy" and passwd == "DineshReddy"):

dashboard()

else:

msg=messagebox.showinfo("error","Entered username or password is wrong!")

def IN():

window1=Tk()

def back():

window1.destroy()

dashboard()

window1.geometry("600x730")

window1.config(bg="pink")

window1.title("QRCode")

dash=Button(window1,text="Dashbaord",bg="blue",command=back).place(x=475,y=35)

def generate():

if len(Subject.get())!=0 :

global qr,photo

qr = pyqrcode.create(Subject.get())

photo = BitmapImage(data = qr.xbm(scale=8))

else:

messagebox.showinfo("Please Enter some Subject")

try:

showcode()

except:

pass

def showcode():

imageLabel.config(image = photo)

subLabel.config(text="QR of " + Subject.get())

def save():

dir = os.getcwd() + "\\QR Codes"

if not os.path.exists(dir):

os.makedirs(dir)

try:

if len(name.get())!=0:

qr.png(os.path.join(dir,name.get()+".png"),scale=8)

else:

messagebox.showinfo("Please enter a File Name")

except:

messagebox.showinfo("Generate the QR code first!")

Sub = Label(window1,text="Enter subject")

Sub.grid(row =0,column =0,sticky=N+S+W+E)

FName = Label(window1,text="Enter FileName")

FName.grid(row =1,column =0,sticky=N+S+W+E)

Subject = StringVar()

SubEntry = Entry(window1,textvariable = Subject)

SubEntry.grid(row =0,column =1,sticky=N+S+W+E)

name = StringVar()

nameEntry = Entry(window1,textvariable = name)

nameEntry.grid(row =1,column =1,sticky=N+S+W+E)

button = Button(window1,text = "Generate",width=15,command = generate)

button.grid(row =0,column =3,sticky=N+S+W+E)

imageLabel = Label(window1)

imageLabel.grid(row =2,column =1,sticky=N+S+W+E)

subLabel = Label(window1,text="")

subLabel.grid(row =3,column =1,sticky=N+S+W+E)

saveB = Button(window1,text="Save as PNG",width=15,command = save)

saveB.grid(row =1,column =3,sticky=N+S+W+E)

Rows = 3

Columns = 3

for row in range(Rows+1):

window1.grid\_rowconfigure(row,weight=1)

for col in range(Columns+1):

window1.grid\_columnconfigure(col,weight=1)

window1.mainloop()

def UPIid():

UPIid=Tk()

def back():

UPIid.destroy()

dashboard()

UPIid.geometry("600x730")

UPIid.config(bg="pink")

UPIid.title("UPI")

dash=Button(UPIid,text="Dashbaord",bg="blue",command=back).place(x=475,y=35)

ti=Label(UPIid,text="UPI Payments",font=("bold",20),bg="Red").place(x=200,y=0)

UPI1=Label(UPIid,text="Enter@UPI",bg="coral1").place(x=100,y=150)

t1=Entry(UPIid,width=35).place(x=170,y=150)

Pay=Button(UPIid,text="Pay",height=2,width=10,bg="violet").place(x=240,y=220)

UPIid.mainloop()

def Sourcestation():

Sourcestation=Tk()

def back():

Sourcestation.destroy()

dashboard()

Sourcestation.geometry("600x650")

Sourcestation.config(bg="yellow")

Sourcestation.title("Sourcestation")

dash=Button(Sourcestation,text="Dashbaord",bg="blue",command=back).place(x=475,y=35)

ti=Label(Sourcestation,text="Sourcestation",font=("bold",20),bg="magenta").place(x=180,y=0)

a=["Anna Nagar East","Anna Nagar Tower","Alandur","Arumbakkam","Ashok Nagar","Chennai International Airport","Ekkattuthangal","Government Estate","Guindy","High Court","Kaladipet","Kilpauk","Koyambedu","St. Thomas Mount","Teynampet","Thirumangalam","Tiruvottriyur","Tondiarpet","Vadapalani","Washermanpet","Wimco Nagar"]

c=StringVar(Sourcestation)

droplist=OptionMenu(Sourcestation,c, \*a)

droplist.config(width=12)

c.set("Sourcestation")

droplist.place(x=20,y=100)

Sourcestation.mainloop()

def Destinationstation():

Destinationstation=Tk()

def back():

Destinationstation.destroy()

dashboard()

Destinationstation.geometry("600x650")

Destinationstation.config(bg="yellow")

Destinationstation.title("Destinationstation")

dash=Button(Destinationstation,text="Dashbaord",bg="blue",command=back).place(x=475,y=35)

ti=Label(Destinationstation,text="Destinationstation",font=("bold",20),bg="magenta").place(x=180,y=0)

a=["Anna Nagar East","Anna Nagar Tower","Alandur","Arumbakkam","Ashok Nagar","Chennai International Airport","Ekkattuthangal","Government Estate","Guindy","High Court","Kaladipet","Kilpauk","Koyambedu","St. Thomas Mount","Teynampet","Thirumangalam","Tiruvottriyur","Tondiarpet","Vadapalani","Washermanpet","Wimco Nagar"]

c=StringVar(Destinationstation)

droplist=OptionMenu(Destinationstation,c, \*a)

droplist.config(width=15)

c.set("Destinationstation")

droplist.place(x=20,y=100)

Destinationstation.mainloop()

def transaction():

transaction=Tk()

transaction.title("Net BAnking Payment")

transaction.geometry("600x740")

transaction.config(bg="light green")

def con(a):

a.config(bg="light green")

def b(j):

j.config(bg="light grey")

login=Label(transaction,text="Net Banking Payment",font=(10),width=30)

login.place(x=50,y=180)

login.config(bg="Violet")

user=Label(transaction,text="Username")

user.place(x=50,y=220,width=100)

con(user)

pw=Label(transaction,text="Password")

pw.place(x=50,y=240,width=100)

con(pw)

u=Entry(transaction,width=50)

u.place(x=150,y=220,width=100)

b(u)

p=Entry(transaction,show="\*",width=50)

p.place(x=150,y=245,width=100)

b(p)

Pay=Button(transaction,text="Pay",height=2,width=10,bg="violet").place(x=150,y=300)

transaction.mainloop()

def MOREOPTIONS():

MOREOPTIONS=Tk()

def back():

MOREOPTIONS.destroy()

dashboard()

MOREOPTIONS.geometry("600x730")

MOREOPTIONS.config(bg="pink")

MOREOPTIONS.title("MOREOPTIONS")

dash=Button(MOREOPTIONS,text="Dashbaord",bg="blue",command=back).place(x=475,y=45)

ti=Label(MOREOPTIONS,text="MORE OPTIONS",font=("bold",20),bg="Red").place(x=200,y=0)

MOREOPTIONS1=Label(MOREOPTIONS,text="Any Query",bg="coral1").place(x=50,y=90)

t1=Entry(MOREOPTIONS,width=85).place(x=50,y=120)

MOREOPTIONS2=Label(MOREOPTIONS,text="Solutions to improve our application",bg="coral1").place(x=50,y=150)

t2=Entry(MOREOPTIONS,width=85).place(x=50,y=180)

MOREOPTIONS3=Label(MOREOPTIONS,text="Give Rating(Out of 5)",bg="coral1").place(x=50,y=210)

t3=Entry(MOREOPTIONS,width=20).place(x=50,y=240)

MOREOPTIONS3=Label(MOREOPTIONS,text="Feedback",bg="coral1").place(x=50,y=270)

t3=Entry(MOREOPTIONS,width=20).place(x=50,y=300)

Submit=Button(MOREOPTIONS,text="Submit",height=2,width=10,bg="violet").place(x=225,y=375)

MOREOPTIONS.mainloop()

def Credit():

Credit=Tk()

def back():

Credit.destroy()

dashboard()

Credit.geometry("600x730")

Credit.config(bg="pink")

Credit.title("Credit card")

dash=Button(Credit,text="Dashbaord",bg="blue",command=back).place(x=475,y=45)

ti=Label(Credit,text="Credit card Payments",font=("bold",20),bg="Red").place(x=200,y=0)

Credit1=Label(Credit,text="Card Number",bg="coral1").place(x=50,y=90)

t1=Entry(Credit,width=40).place(x=50,y=120)

Credit2=Label(Credit,text="Expiry",bg="coral1").place(x=50,y=150)

t2=Entry(Credit,width=15).place(x=50,y=180)

Credit3=Label(Credit,text="CVV",bg="coral1").place(x=200,y=150)

t3=Entry(Credit,width=10).place(x=200,y=181)

Credit4=Label(Credit,text="Name on card",bg="coral1").place(x=50,y=210)

t4=Entry(Credit,width=40).place(x=50,y=240)

Pay=Button(Credit,text="Pay",height=2,width=10,bg="violet").place(x=240,y=300)

Credit.mainloop()

def Debit():

Debit=Tk()

def back():

Debit.destroy()

dashboard()

Debit.geometry("600x730")

Debit.config(bg="pink")

Debit.title("Debit card")

dash=Button(Debit,text="Dashbaord",bg="blue",command=back).place(x=475,y=45)

ti=Label(Debit,text="Debit card Payments",font=("bold",20),bg="Red").place(x=200,y=0)

Debit1=Label(Debit,text="Card Number",bg="coral1").place(x=50,y=90)

t1=Entry(Debit,width=40).place(x=50,y=120)

Debit2=Label(Debit,text="Expiry",bg="coral1").place(x=50,y=150)

t2=Entry(Debit,width=15).place(x=50,y=180)

Debit3=Label(Debit,text="CVV",bg="coral1").place(x=200,y=150)

t3=Entry(Debit,width=10).place(x=200,y=181)

Debit4=Label(Debit,text="Name on card",bg="coral1").place(x=50,y=210)

t4=Entry(Debit,width=40).place(x=50,y=240)

Pay=Button(Debit,text="Pay",height=2,width=10,bg="violet").place(x=240,y=300)

Debit.mainloop()

def Banks():

Banks=Tk()

def back():

Banks.destroy()

dashboard()

Banks.geometry("600x730")

Banks.config(bg="blue")

Banks.title("PAYMENTS")

dash=Button(Banks,text="Dashbaord",bg="blue",command=back).place(x=475,y=35)

ti=Label(Banks,text="Banks",font=("bold",20),bg="magenta").place(x=180,y=0)

Bank1=Button(Banks,text="State Bank of India",font=("bold",10),command=transaction).place(x=20,y=120)

Bank2=Button(Banks,text="City Union Bank",font=("bold",10),command=transaction).place(x=20,y=180)

Bank3=Button(Banks,text="Andhra Bank",font=("bold",10),command=transaction).place(x=20,y=240)

Bank4=Button(Banks,text="Canara Bank",font=("bold",10),command=transaction).place(x=20,y=300)

Bank5=Button(Banks,text="Axis Bank",font=("bold",10),command=transaction).place(x=20,y=360)

Bank6=Button(Banks,text="Syndicate Bank",font=("bold",10),command=transaction).place(x=20,y=420)

Bank7=Button(Banks,text="ICICI Bank",font=("bold",10),command=transaction).place(x=20,y=480)

Bank8=Button(Banks,text="HDFC Bank",font=("bold",10),command=transaction).place(x=20,y=540)

Bank9=Button(Banks,text="CITI Bank",font=("bold",10),command=transaction).place(x=20,y=600)

Banks.mainloop()

def UPI():

UPI=Tk()

def back():

UPI.destroy()

dashboard()

UPI.geometry("600x730")

UPI.config(bg="pink")

UPI.title("UPI")

dash=Button(UPI,text="Dashbaord",bg="blue",command=back).place(x=475,y=35)

ti=Label(UPI,text="UPI Payments",font=("bold",20),bg="Red").place(x=200,y=0)

UPI1=Button(UPI,text="Paytm",font=("bold",10),command=UPIid).place(x=100,y=120)

UPI2=Button(UPI,text="Gpay",font=("bold",10),command=UPIid).place(x=100,y=300)

UPI3=Button(UPI,text="Phone pe",font=("bold",10),command=UPIid).place(x=400,y=120)

UPI4=Button(UPI,text="Amazon pay",font=("bold",10),command=UPIid).place(x=400,y=300)

UPI.mainloop()

def payments():

payments=Tk()

def back():

payments.destroy()

dashboard()

payments.geometry("600x730")

payments.config(bg="blue")

payments.title("PAYMENTS")

dash=Button(payments,text="Dashbaord",bg="blue",command=back).place(x=475,y=35)

ti=Label(payments,text=" Payments",font=("bold",20),bg="magenta").place(x=180,y=0)

pay1=Button(payments,text="Debit card",font=("bold",10),command=Debit).place(x=20,y=120)

pay2=Button(payments,text="Credit card",font=("bold",10),command=Credit).place(x=20,y=180)

pay3=Button(payments,text="Net Banking",font=("bold",10),command=Banks).place(x=20,y=240)

pay4=Button(payments,text="UPI payments",font=("bold",10),command=UPI).place(x=20,y=300)

payments.mainloop()

def reports():

rep=Tk()

def back():

rep.destroy()

dashboard()

rep.geometry("600x730")

rep.config(bg="yellow")

rep.title("REPORTS")

dash=Button(rep,text="Dashbaord",bg="blue",command=back).place(x=475,y=35)

ti=Label(rep,text=" REPORTS",font=("bold",20),bg="magenta").place(x=180,y=0)

year1=Button(rep,text="REPORT FOR 2014-2015",font=("bold",10)).place(x=20,y=60)

year2=Button(rep,text="REPORT FOR 2015-2016",font=("bold",10)).place(x=20,y=120)

year3=Button(rep,text="REPORT FOR 2016-2017",font=("bold",10)).place(x=20,y=180)

year4=Button(rep,text="REPORT FOR 2017-2018",font=("bold",10)).place(x=20,y=240)

year5=Button(rep,text="REPORT FOR 2018-2019",font=("bold",10)).place(x=20,y=300)

year6=Button(rep,text="REPORT FOR 2019-2020",font=("bold",10)).place(x=20,y=360)

year7=Button(rep,text="REPORT FOR 2020-2021",font=("bold",10)).place(x=20,y=420)

update=Button(rep,text="Update",height=2,width=10,bg="violet").place(x=50,y=500)

prin=Button(rep,text="Print",height=2,width=10,bg="violet").place(x=250,y=500)

generate=Button(rep,text="Generate",height=2,width=10,bg="violet").place(x=450,y=500)

rep.mainloop()

def dashboard():

window1=Tk()

window1.geometry("1450x750")

window1.title("Dashbaord")

def quit():

window1.destroy()

messagebox.showinfo("status","logged out successfully")

window1.config(bg="thistle2")

name=Label(window1,text="Metro Railway Reservation MANAGEMENT",font=(15),width=40,bg="blue",fg="pink").place(x=550,y=10)

home=Button(window1,text="Settings",font=(6),width=10,height=1,bg="maroon1").place(x=20,y=10)

quit=Button(window1,text="Log out",font=(6),width=10,height=1,bg="light grey",command=quit).place(x=1275,y=10)

sd=Button(window1,text="Source station",font=(10),width=20,height=3,bg="light green",command=Sourcestation).place(x=60,y=150)

bill=Button(window1,text="Destination station",font=(10),width=20,height=3,bg="silver",fg="red",command=Destinationstation).place(x=600,y=150)

cusdb=Button(window1,text="Payments",font=(10),width=20,height=3,bg="dark olive green3",command=payments).place(x=1200,y=150)

invo=Button(window1,text="INVOICES",font=(10),width=20,height=3,bg="gold",command=IN).place(x=60,y=500)

rep=Button(window1,text="REPORTS",font=(10),width=20,height=3,bg="sky blue",command=reports).place(x=600,y=500)

more=Button(window1,text="MORE\nOPTIONS",font=(10),width=20,height=3,bg="sienna2",command=MOREOPTIONS).place(x=1200,y=500)

window1.mainloop()

log=Button(text="Login",width=100,command=check)

log.place(x=650,y=300,width=100)

log.config(bg="orange")

don=Label(window,text="Don't have an account?")

don.place(x=600,y=340,width=130)

con(don)

def sign\_up():

supage=Tk()

supage.title("Register")

supage.geometry("600x600")

supage.config(bg="mistyrose2")

head=Label(supage,text="Register your account",bg="red",font=("bold",18)).grid(column=1,row=0,columnspan=3,padx=10,pady=10)

fn=Label(supage,text="First name ").grid(column=0,row=1,padx=10,pady=10)

t2=Entry(supage,width=30,bg="white").grid(column=1,row=1,padx=10,pady=10)

ln=Label(supage,text="Last name ").grid(column=0,row=2,padx=10,pady=10)

t3=Entry(supage,width=30).grid(column=1,row=2,padx=10,pady=10)

dob=Label(supage,text="Date of Birth")

dob.grid(column=0,row=3,padx=10,pady=10)

dates=[ 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31]

months=["january","febrary","march","april","may","june","july","august","september","october","november","december"]

years=[1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012,2013,2015,2016]

c=StringVar(supage)

droplist=OptionMenu(supage,c, \*dates)

droplist.config(width=5)

c.set("date")

droplist.grid(row=3,column=1,padx=10,pady=10)

c=StringVar(supage)

droplist=OptionMenu(supage,c, \*months)

droplist.config(width=5)

c.set("month")

droplist.grid(row=3,column=2,padx=10,pady=10)

c=StringVar(supage)

droplist=OptionMenu(supage,c, \*years)

droplist.config(width=5)

c.set("year")

droplist.grid(row=3,column=3,padx=10,pady=10)

bn=Label(supage,text="Gender ")

bn.grid(column=0,row=4,padx=10,pady=10)

rad1 = Radiobutton(supage,text='male',value=1).grid(column=1,row=4,padx=10,pady=10)

rad2 = Radiobutton(supage,text='female',value=0).grid(column=2,row=4,pady=10,padx=10)

em=Label(supage,text="Email address ").grid(column=0,row=5,padx=10,pady=10)

t6=Entry(supage,width=30).grid(column=1,row=5,padx=10,pady=10)

mn=Label(supage,text="Mobile number ").grid(column=0,row=6,padx=10,pady=10)

t5=Entry(supage,width=30).grid(column=1,row=6,pady=10,padx=10)

pw=Label(supage,text="Password ").grid(column=0,row=7,padx=10,pady=10)

t7=Entry(supage,show="\*",width=30).grid(column=1,row=7,padx=10,pady=10)

cpw=Label(supage,text="Confirm Password ").grid(column=0,row=8,padx=10,pady=10)

t7=Entry(supage,show="\*",width=30).grid(column=1,row=8,padx=10,pady=10)

st=Label(supage,text="State ").grid(column=0,row=9,padx=10,pady=10)

t8=Entry(supage,width=30).grid(column=1,row=9,padx=10,pady=10)

ctry=Label(supage,text="Country ").grid(column=0,row=10,padx=10,pady=10)

t8=Entry(supage,width=30).grid(column=1,row=10,padx=10,pady=10)

def created():

supage.destroy()

msg=messagebox.showinfo("Registered","Completed registration successfully")

submit=Button(supage,text="Submit",font=("bold",10),fg="Black",bg="green3",command=created).grid(row=11,column=1,columnspan=3)

supage.mainloop()

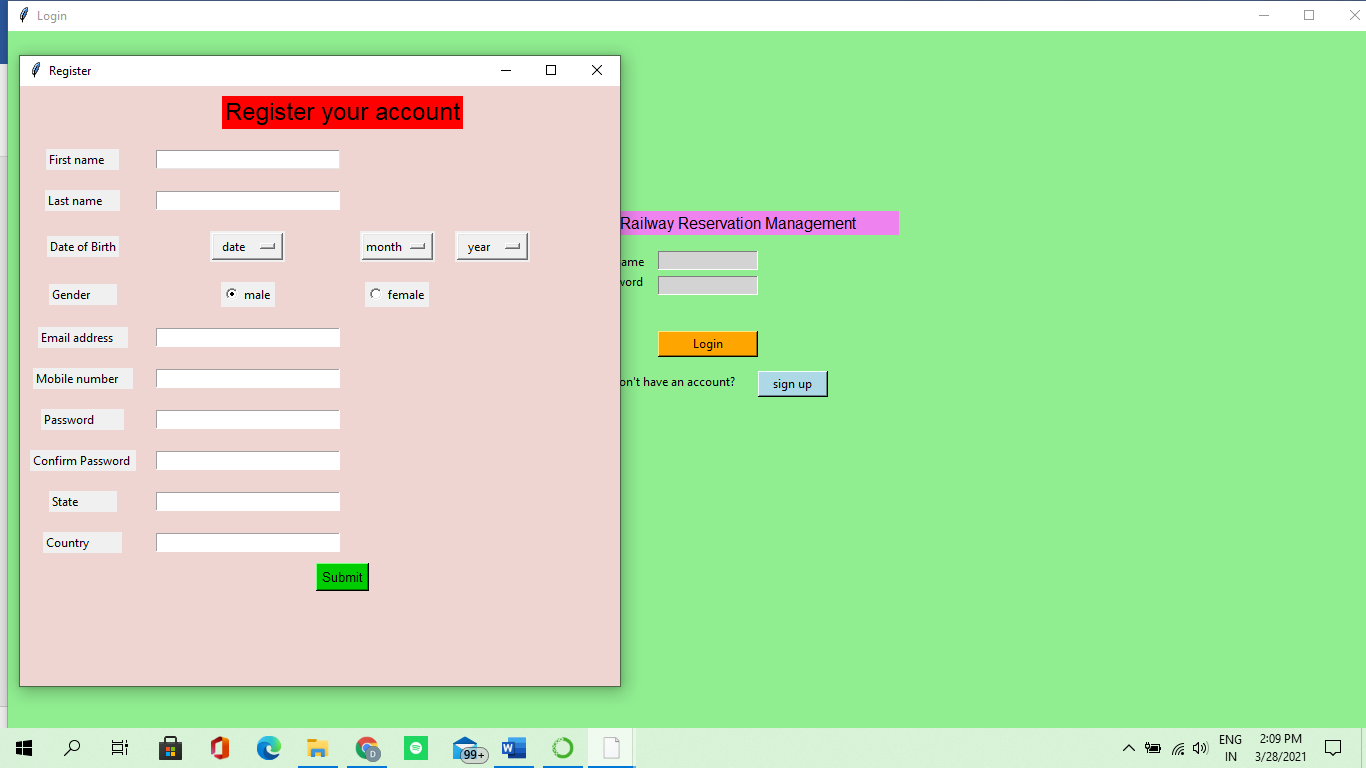
sp=Button(window,text="sign up",width=60,command=sign\_up)

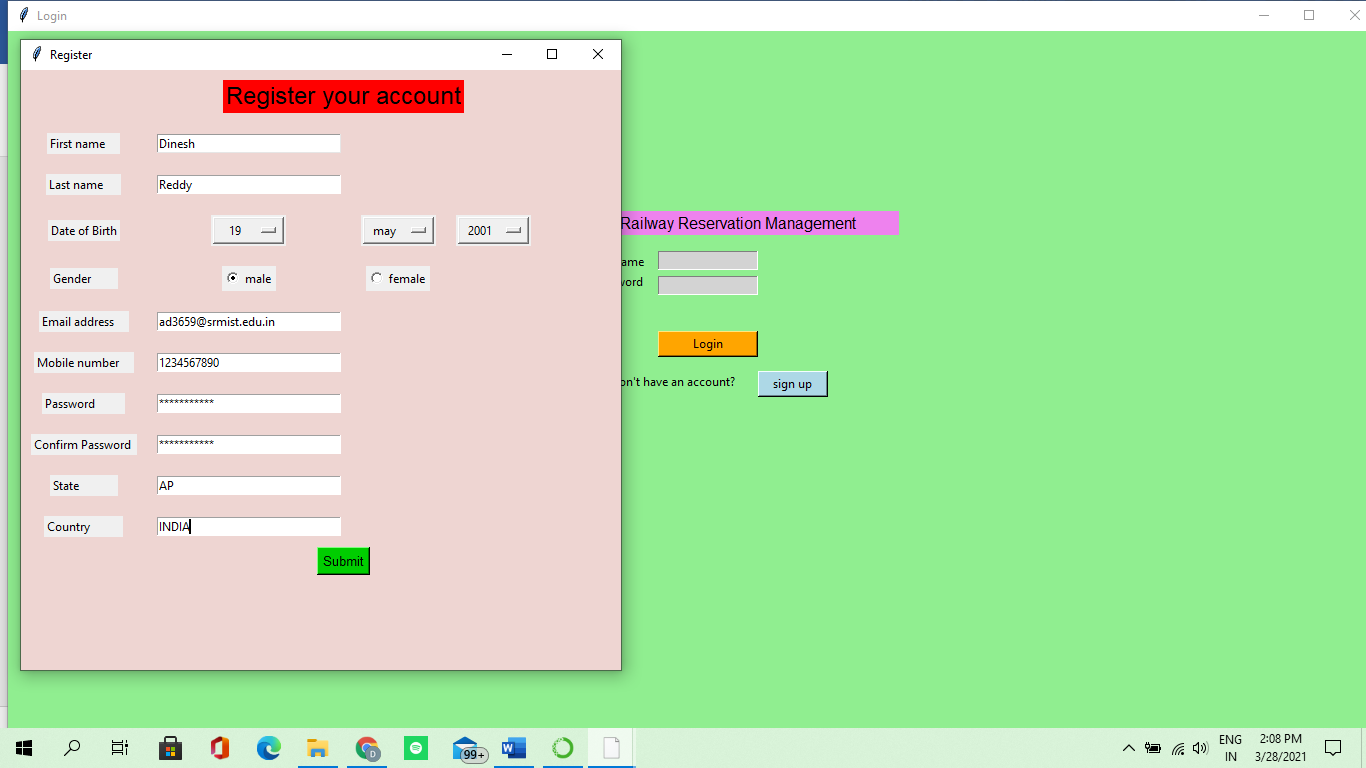
sp.place(x=750,y=340,width=70)

sp.config(bg="light blue")

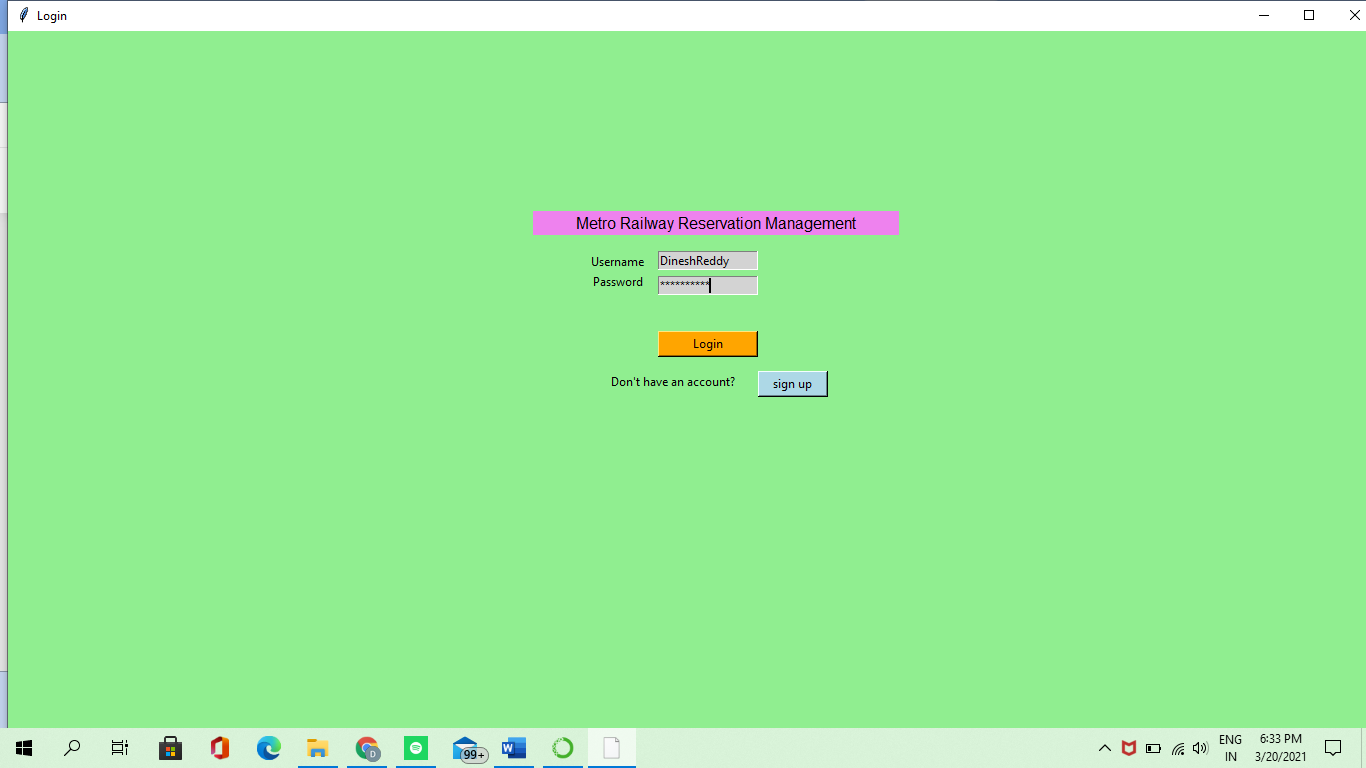
window.mainloop()

If u don’t have an account ,u have to click the signup option and u have to register by filling the basic personal details such as name,password,mobile number,state,password….etc,then u can successfully login.

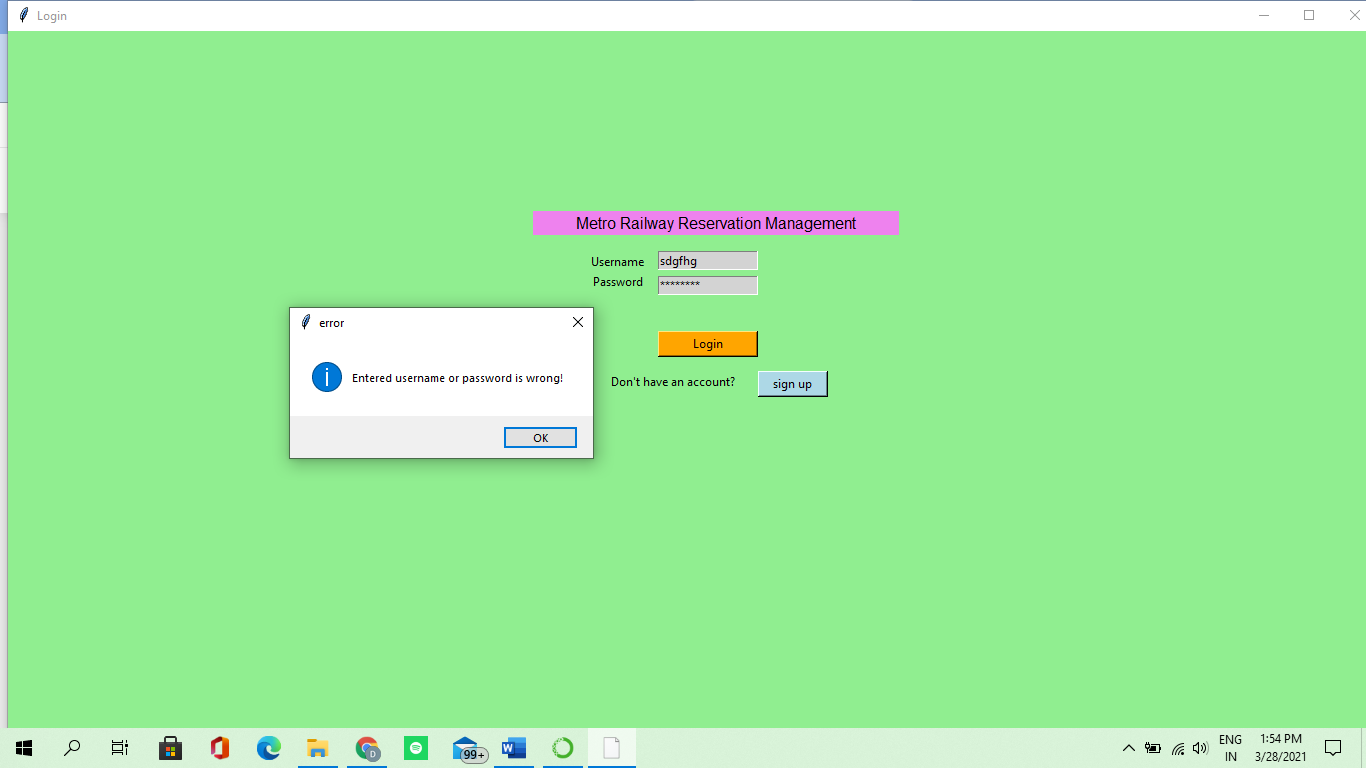




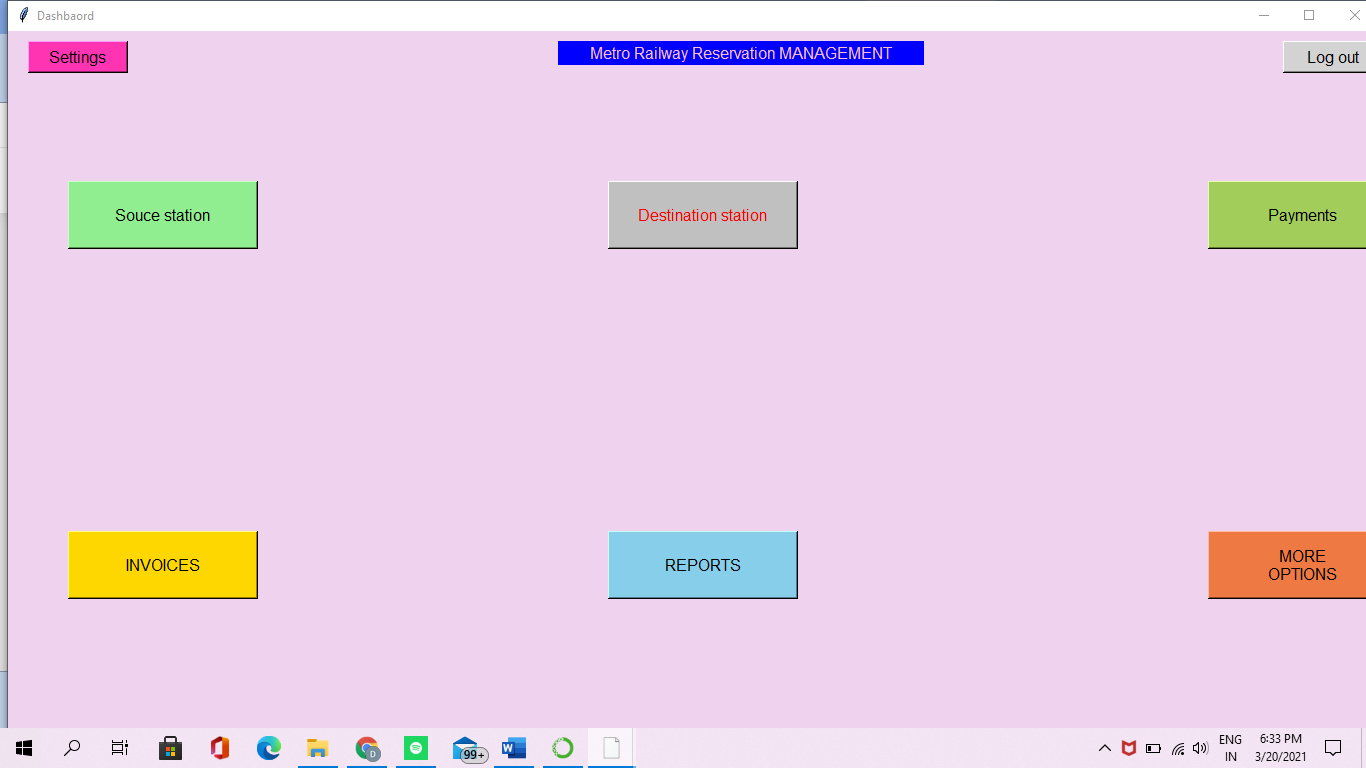
This is the login page ,where we enter the basic details such as the correct username and password.



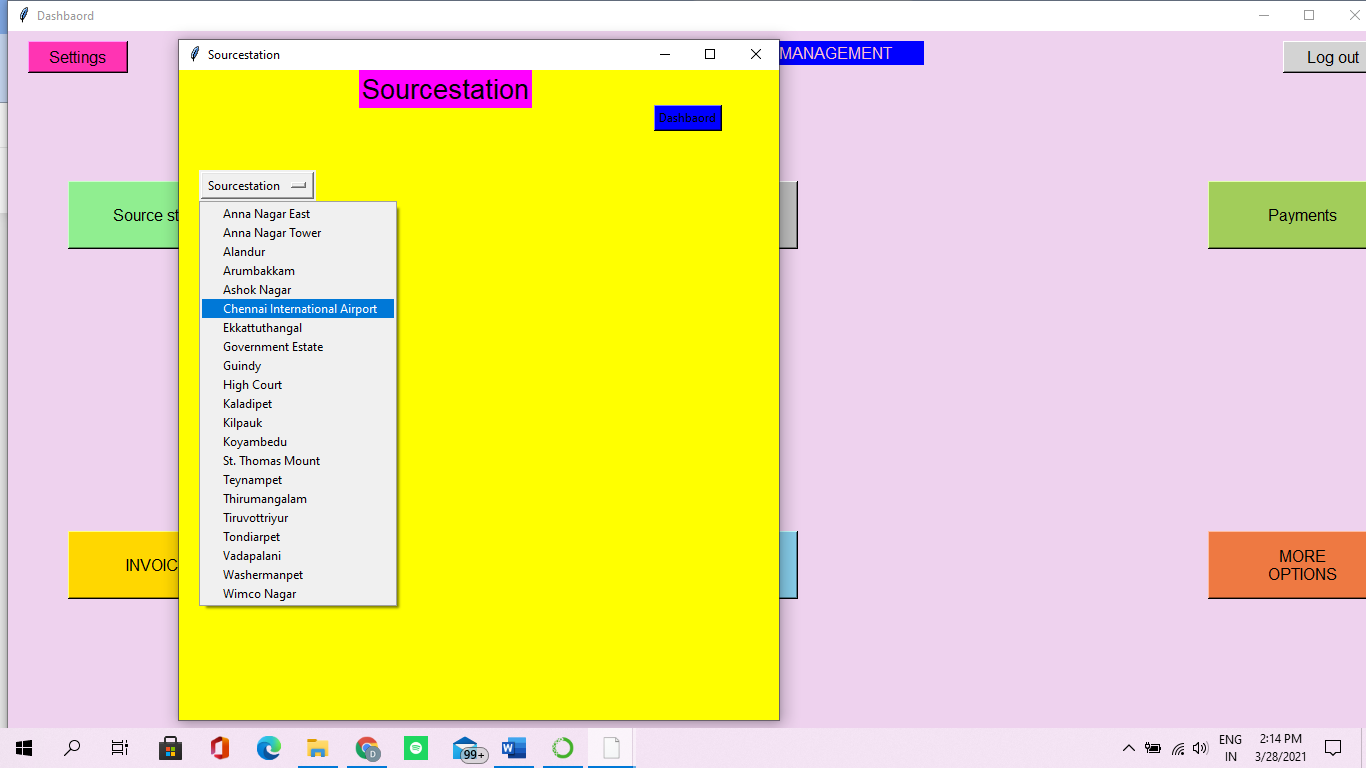
If u enter the wrong username and wrong password,it shows the error that the username and password u have entered is wrong.



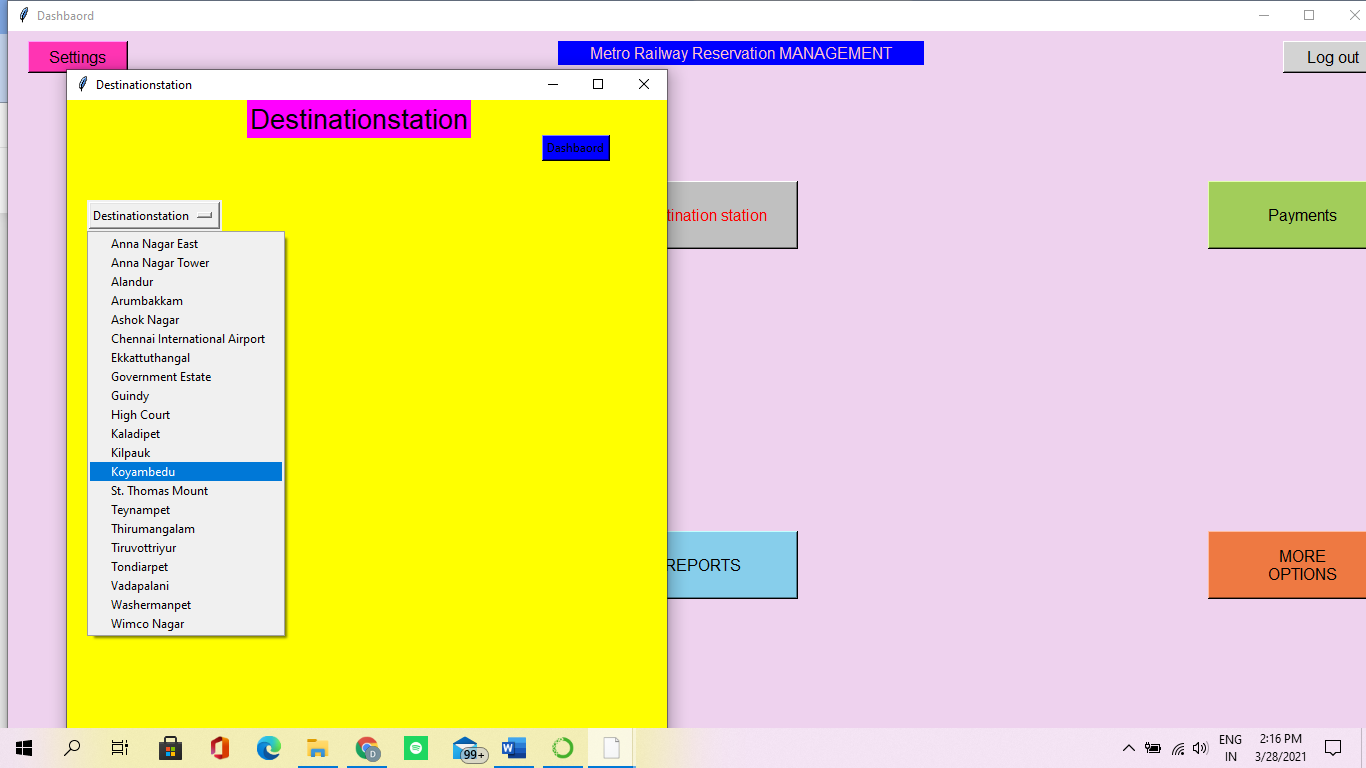
After successfully login,u will enter into one more page where u have to select the source and destination stations and have to do the payment to get the ticket.



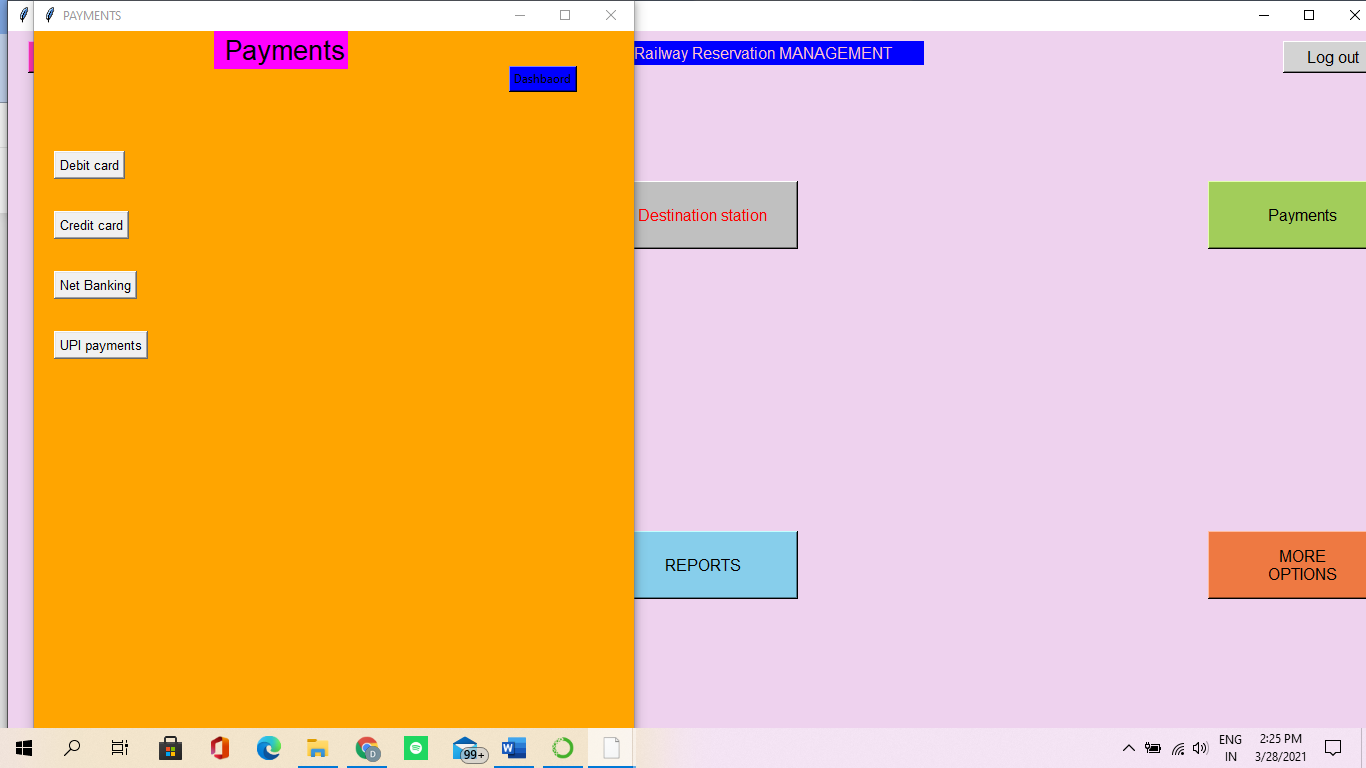
By clicking on the source station,u will be directed to select the starting(source) point of metro station names.



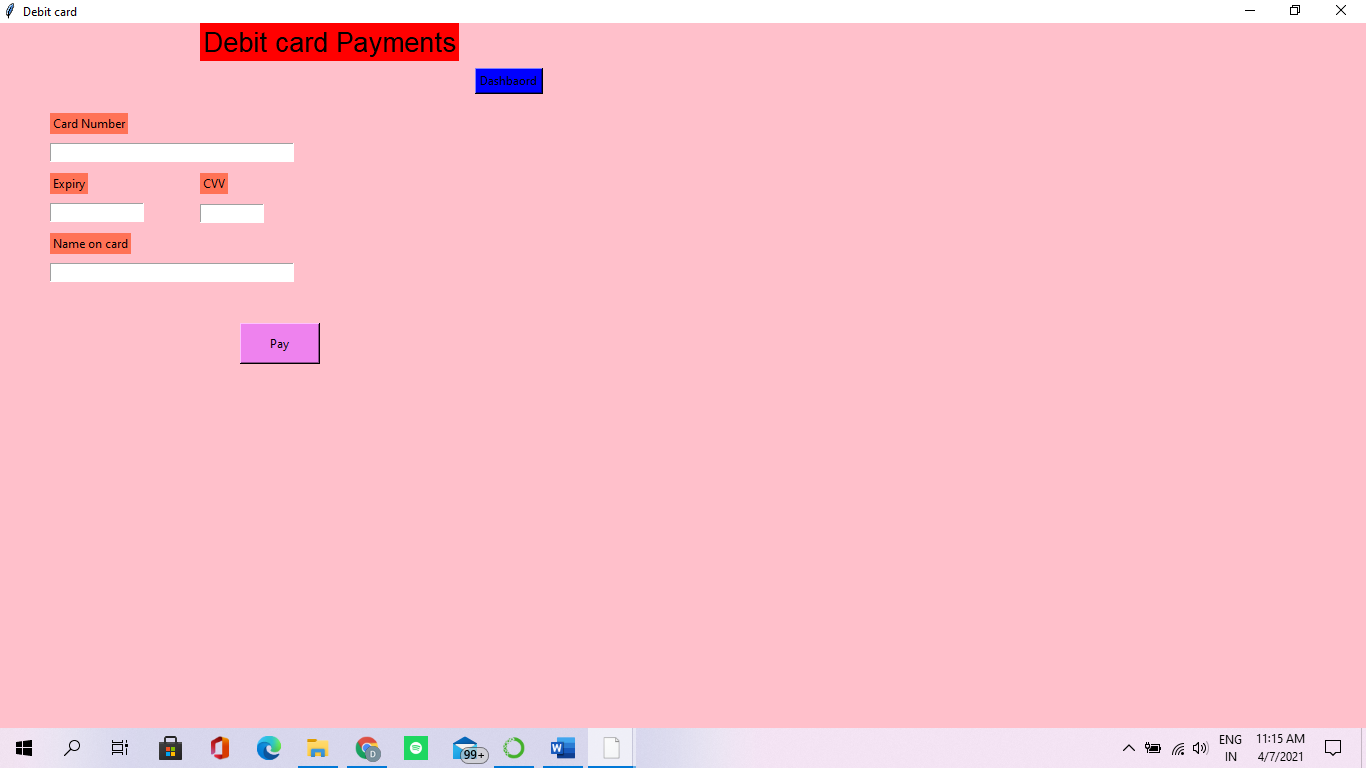
By clicking on the destination station,u will be directed to select the last(destination) point of metro station names.



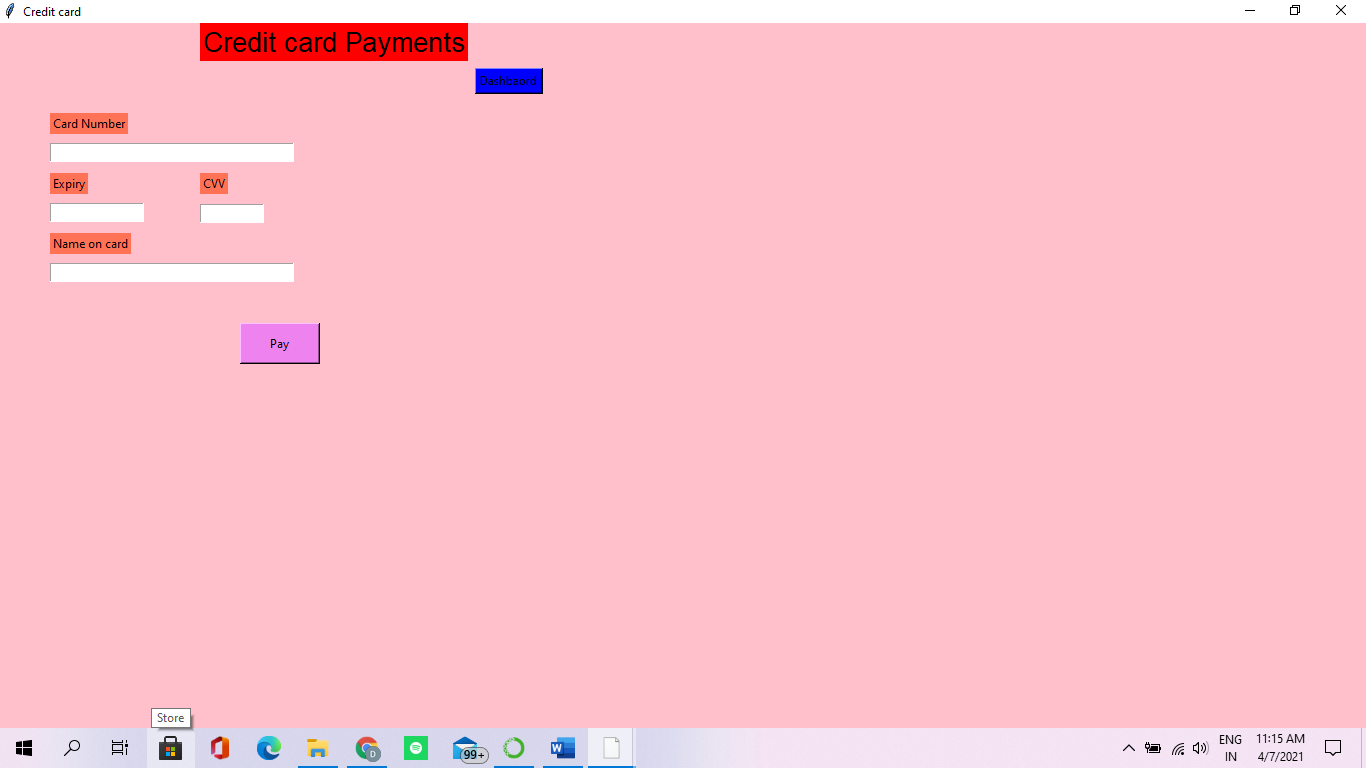
Next ,we have to do the payment by clicking on payment options,it will be directed to different payment methods



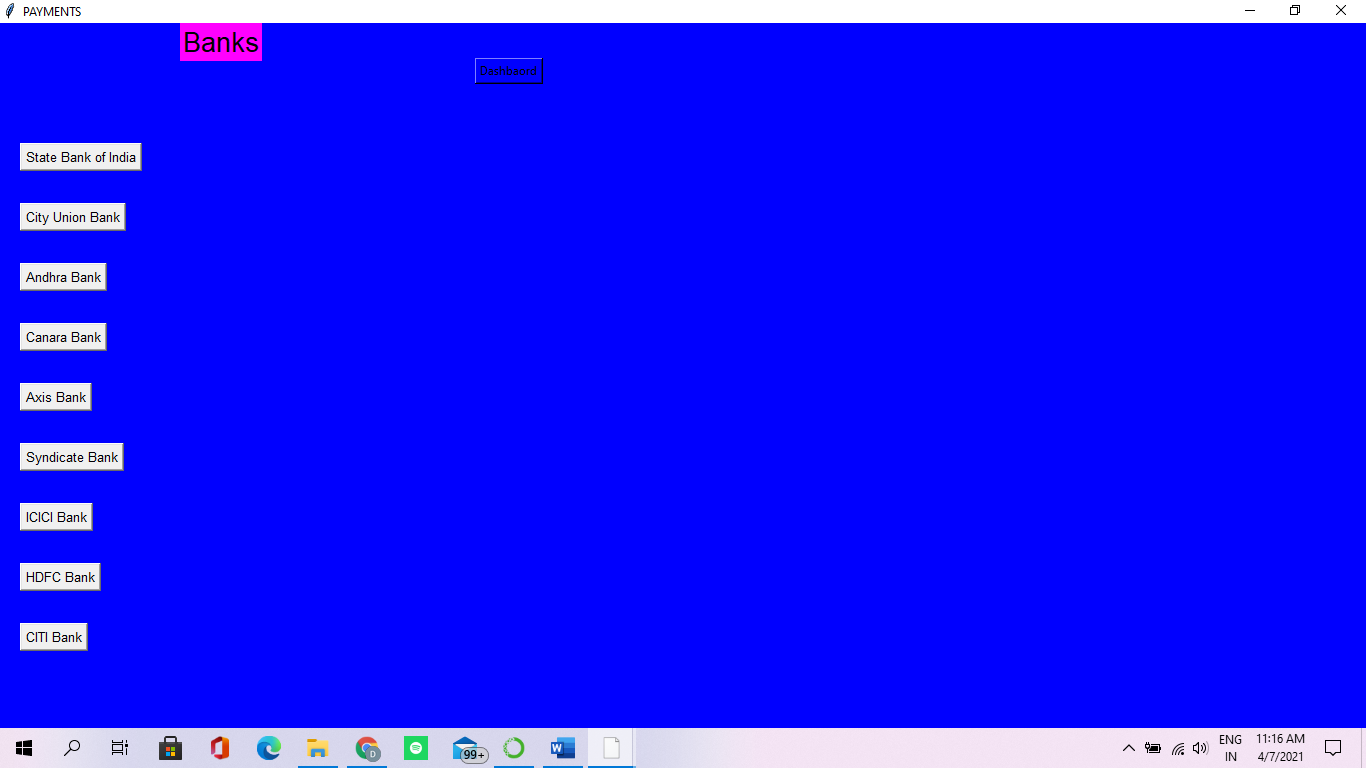
For Debit card payment click debit card and type details related to card,and click on pay button



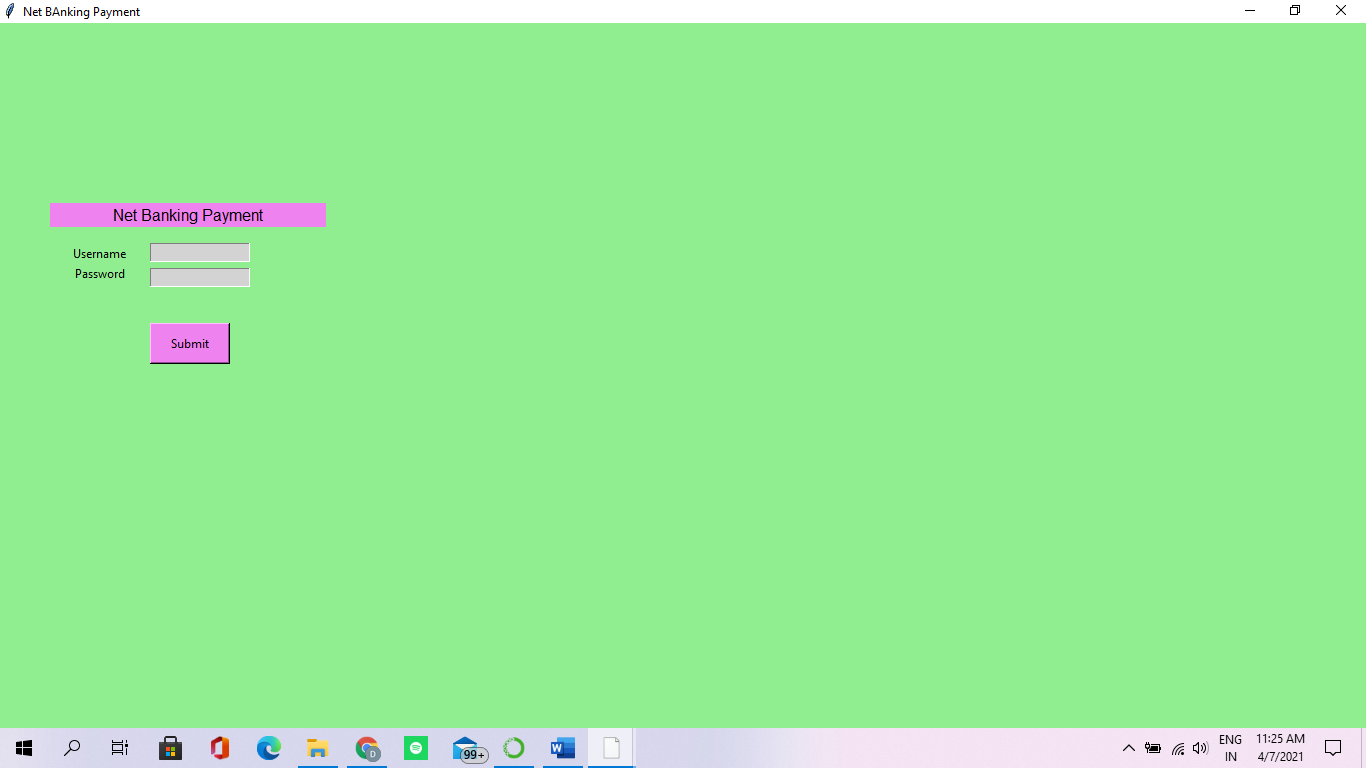
For Credit card payment click credit card and type details related to card,and click on pay button



For Net banking payment click net banking and select ur required bank ,it will be redirected to login page of that bank.



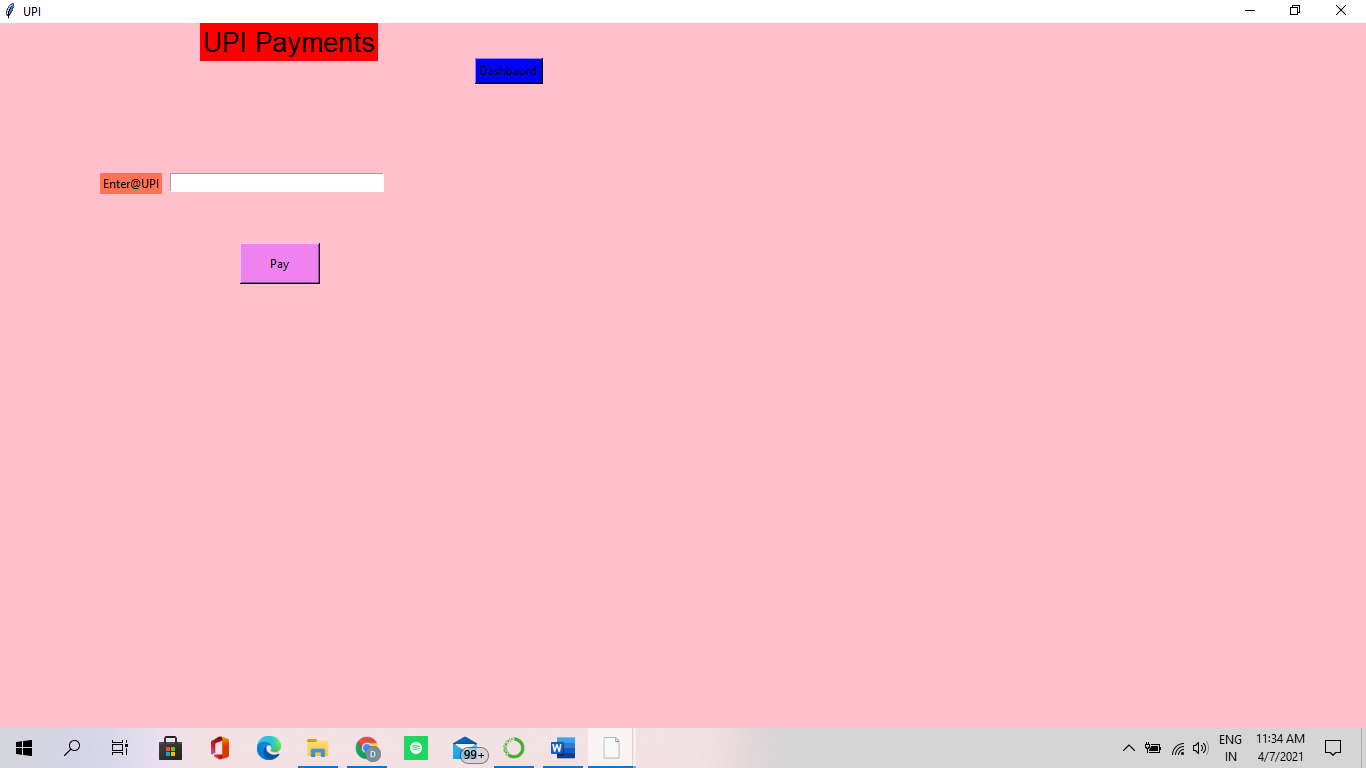
Next,u have to enter the login details of the required bank and click submit button.



If u want to do UPI payment,click UPI payments and it will be redirected to required UPI payment method.



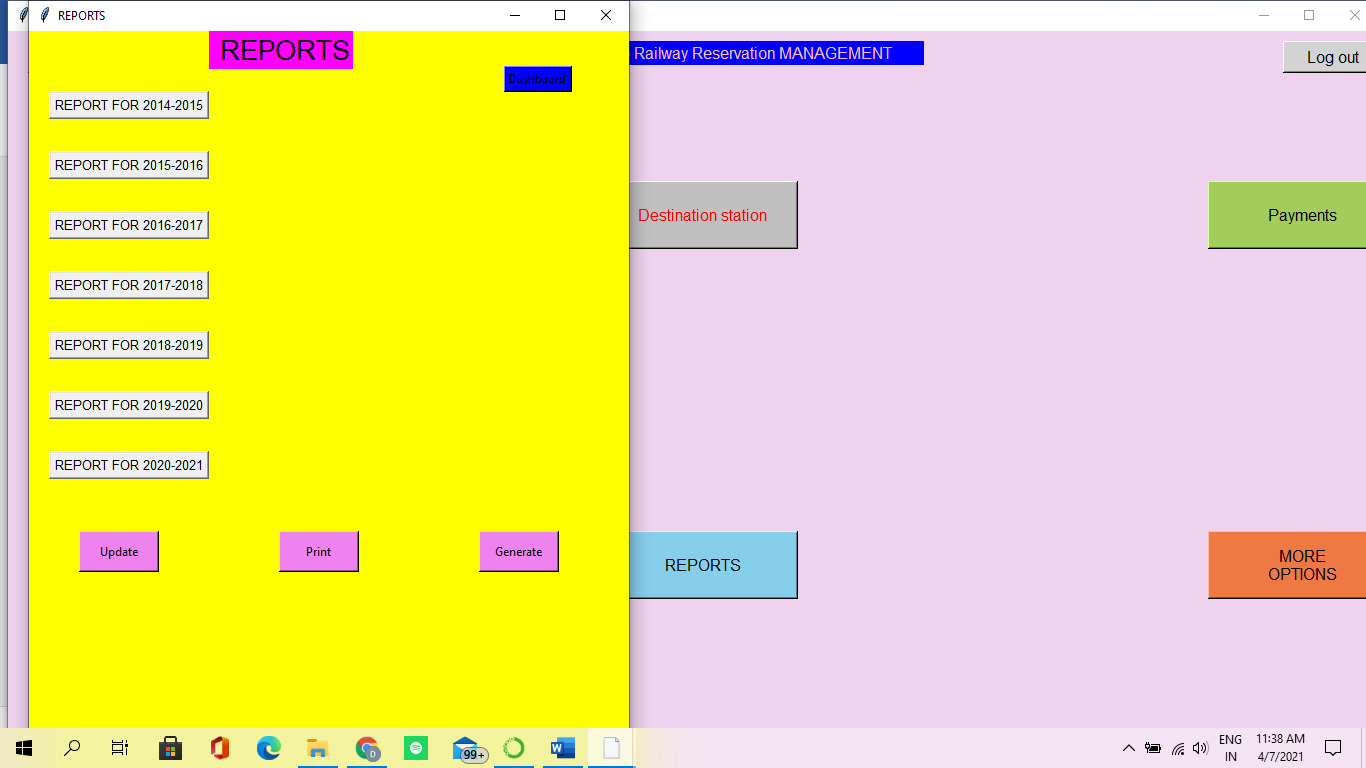
Whatever the UPI payment method u will click,it will ask to enter the upi address and then click pay option for successful payment.



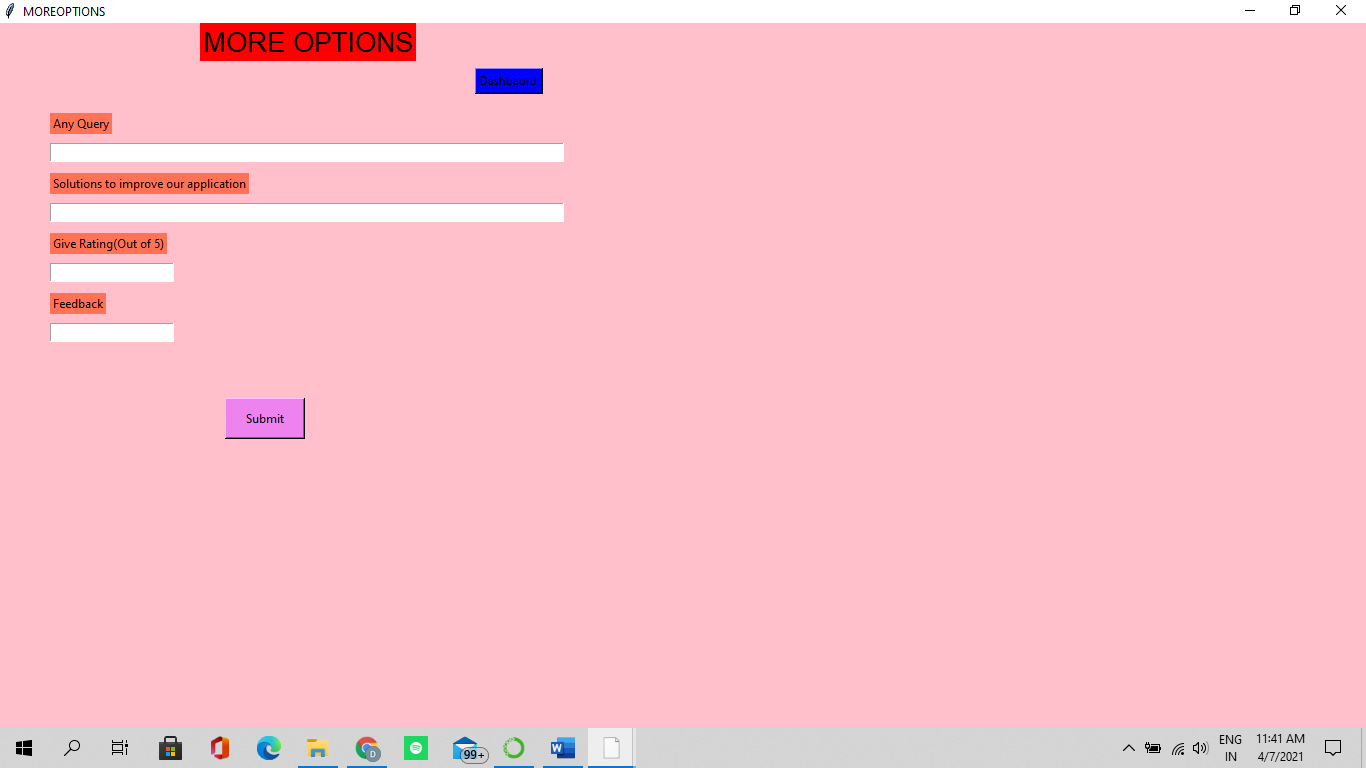
Then comes the QR code generation for the ticket which is inside of Invoices button.



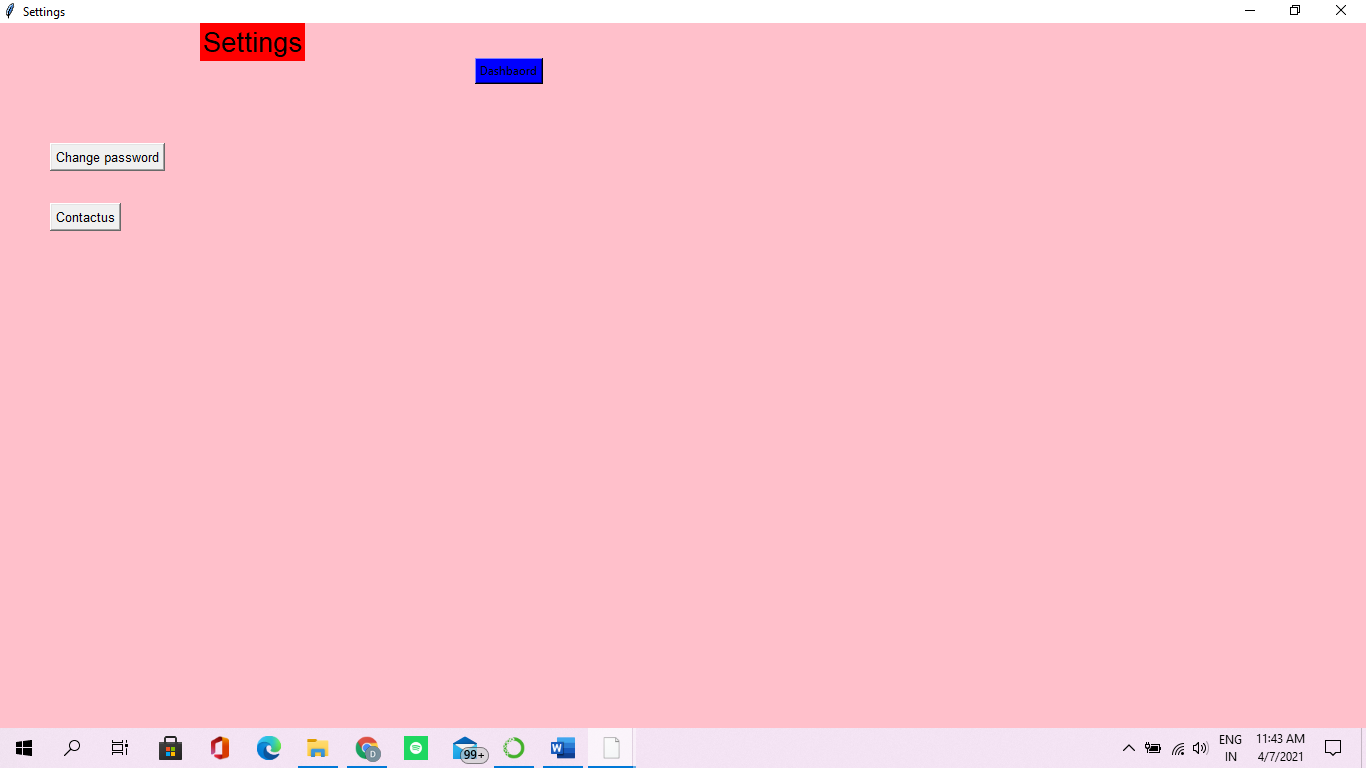
If u click on Reports button ,u will get the previous year bookings list.



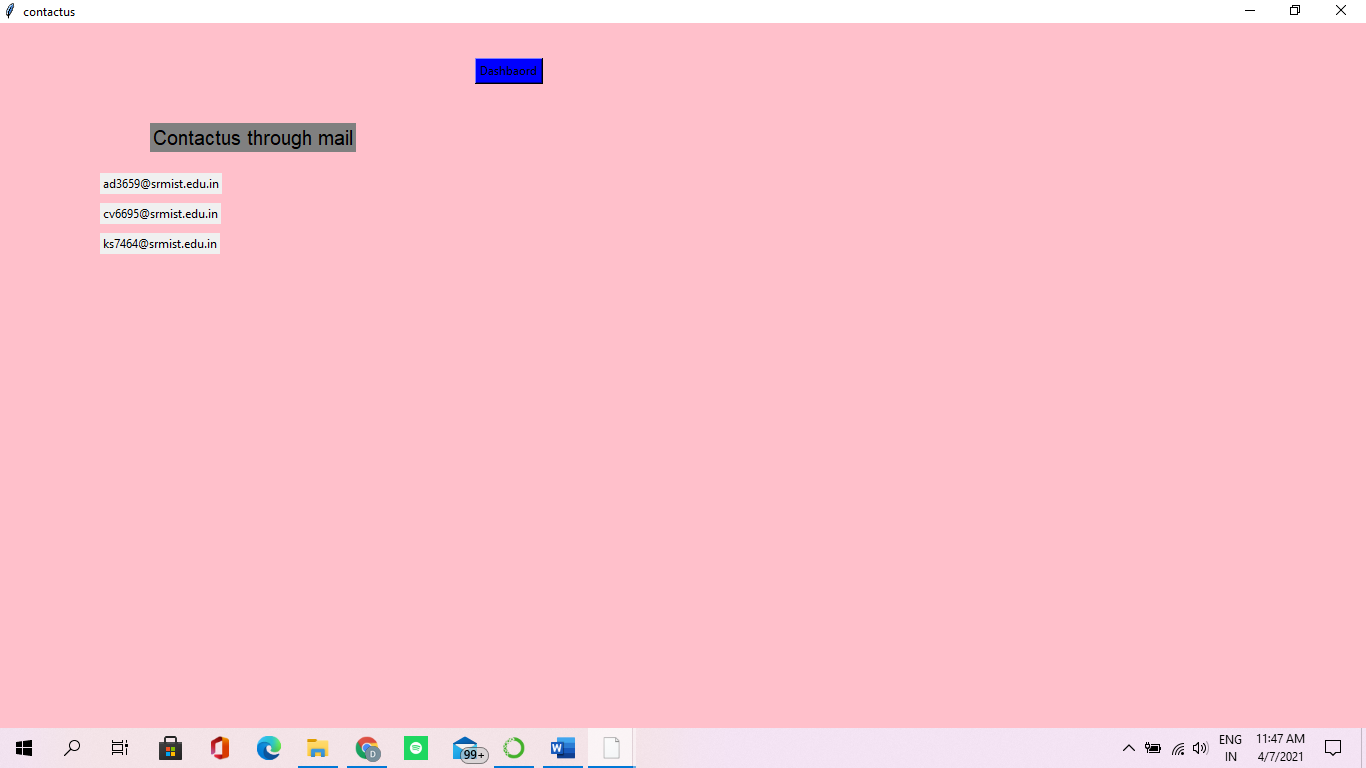
If u have any issue or u want to give any feedback u can click the MoreOptions button



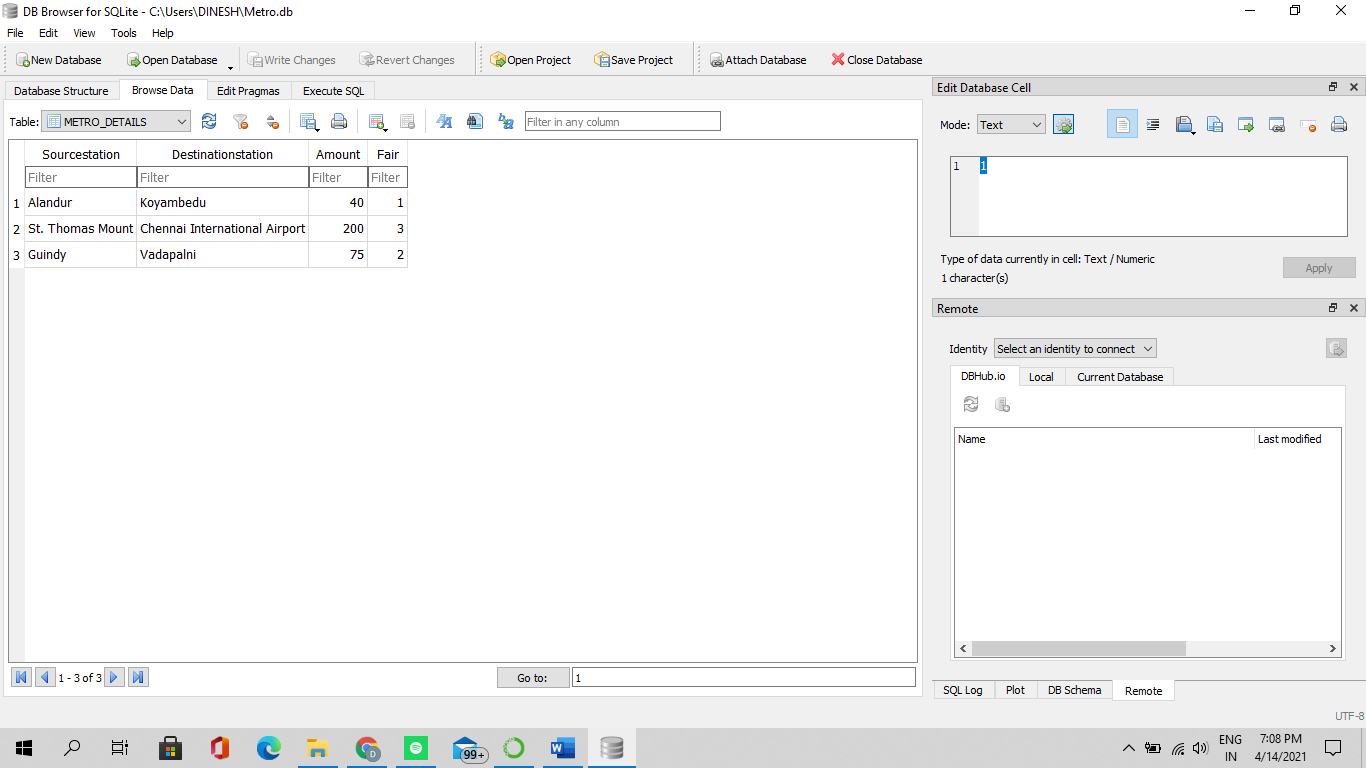
If u want to change the password ,u can go to settings and click the change password button



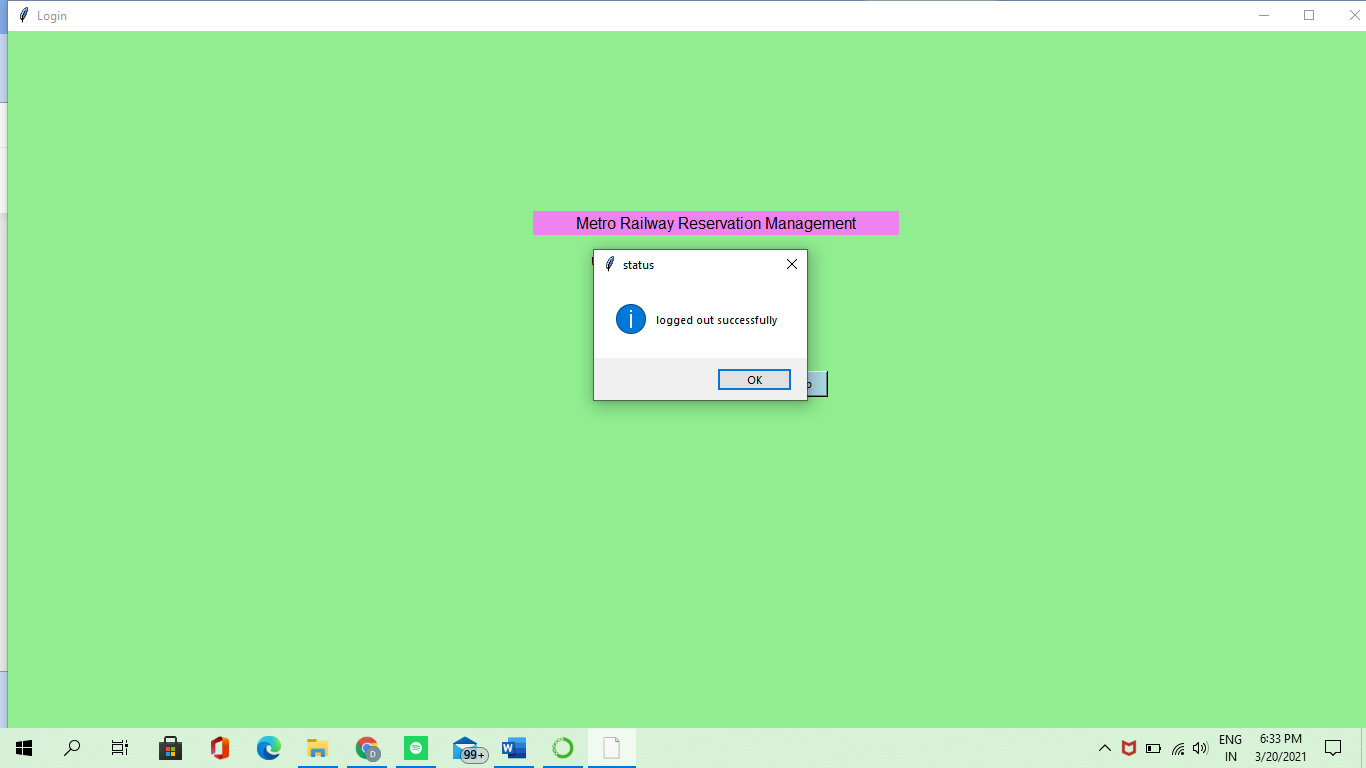
If u want to contactus ,u can contact through our given mailId’s



The selected values will be stored in database of sqlite



After all,u have to click the logout button,then a message will be displayed as u have logout successfully



Scope of testing:

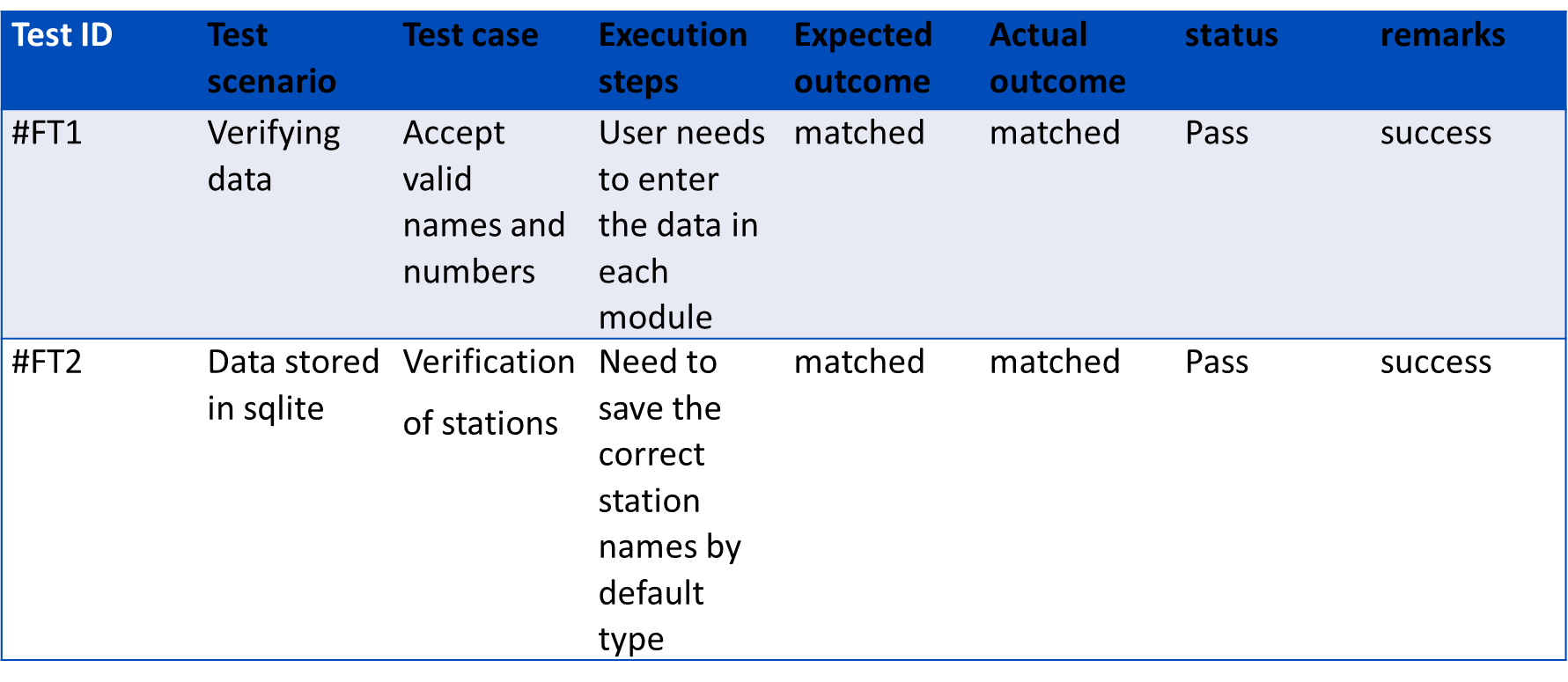
* Satisfying the user requirement
* Easy to understand by the user and operator
* Easy to operate
* Good user interface
* Expandable

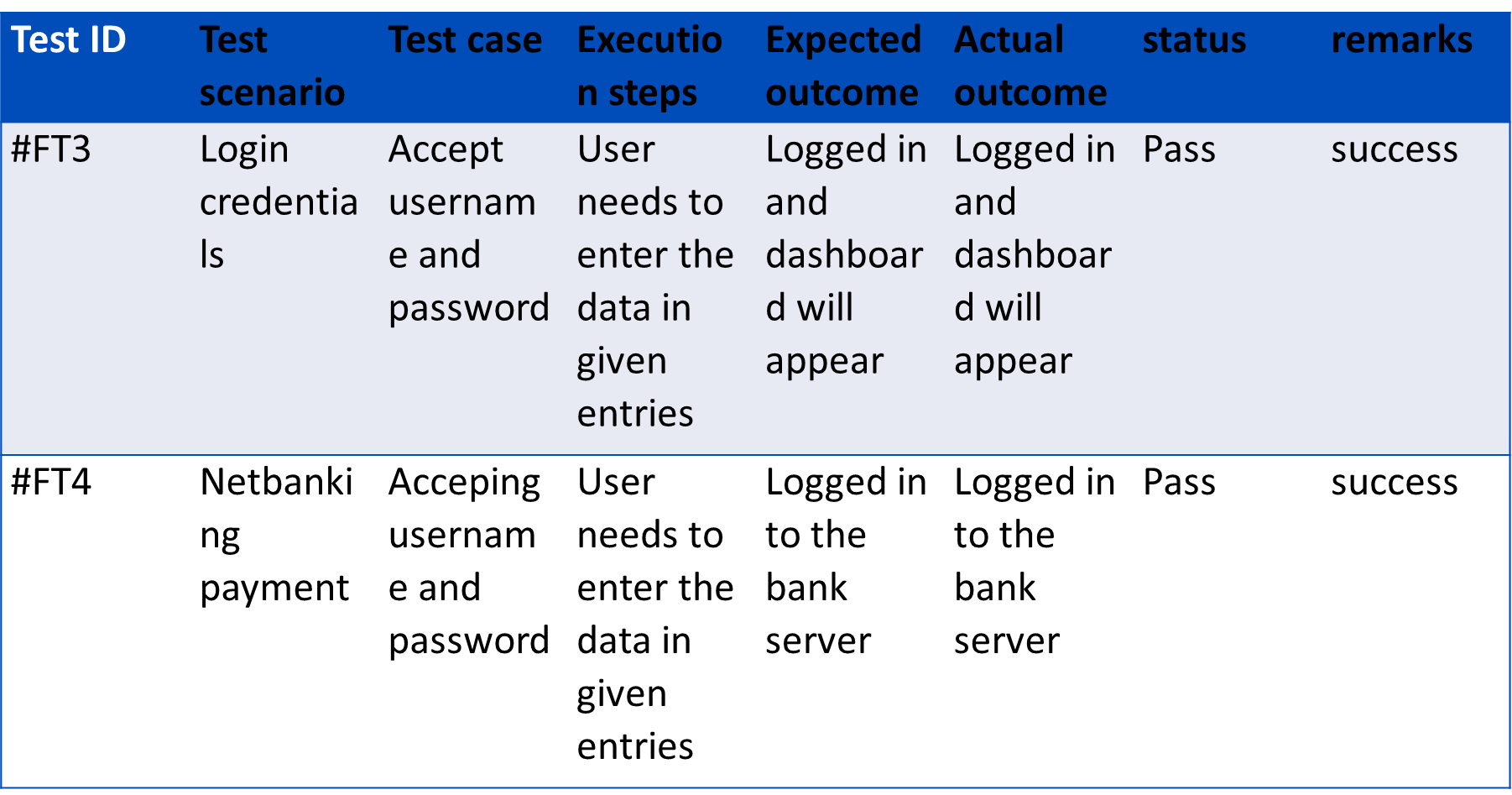
Test deliverables:

* Test plan
* Test strategy
* Bug report
* Test execution report
* Test summary
* User guide
* Installation and configuration guide
* Release note

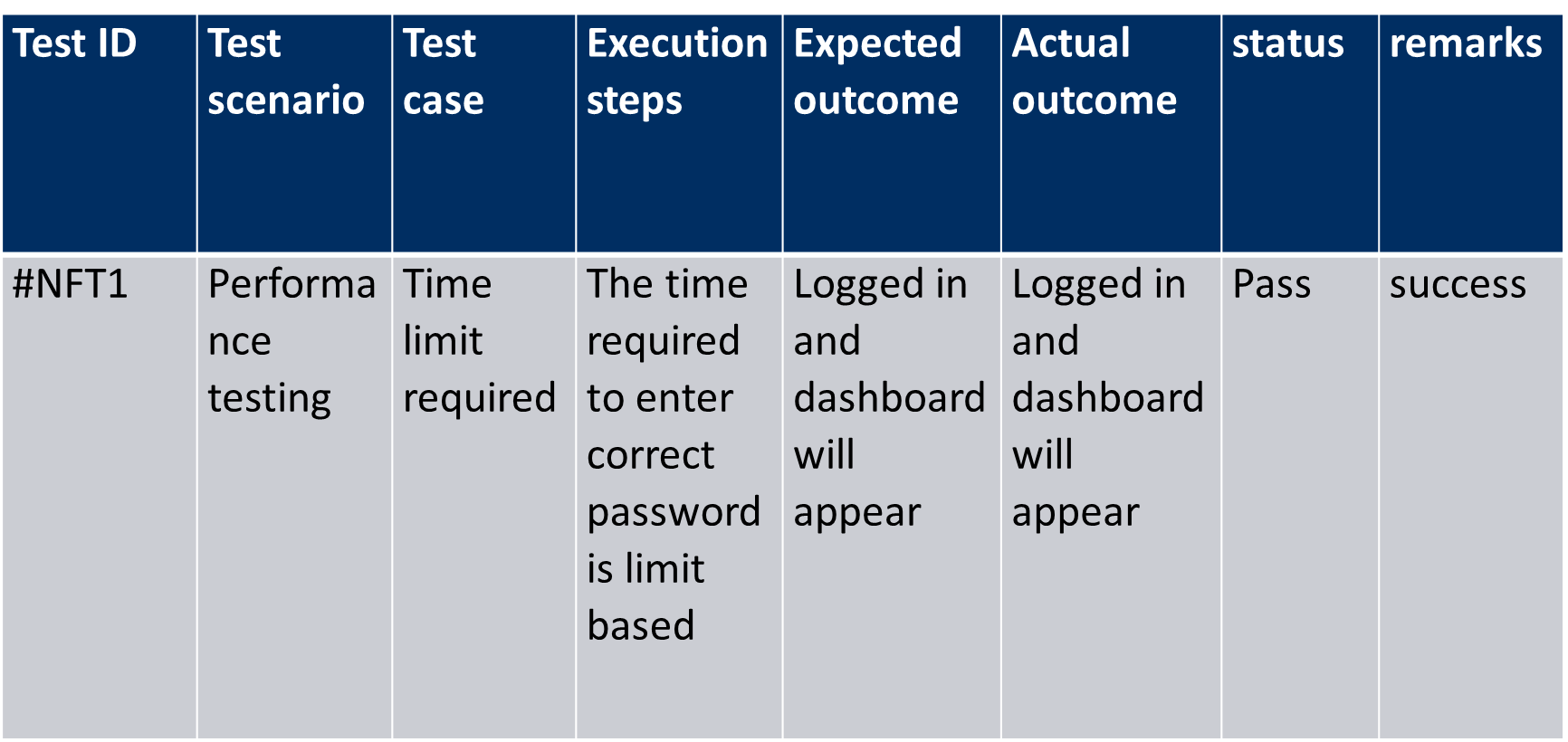
**TEST CASE:**

Functional test cases:

****

****

Non-functional test cases:

****

Defect Log:-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Requirement | Defect ID | Defect Description | Assignee | Status |
| M1R1 | QR code | QR code if we run that function,In the main code the QR code function is not working | Project Developer | Under working |
| M1R2 | Database | Some selected entries of stations are not storing in database | Project Manager | Under working |

Result:-Successfully completed the execution of Metro Railway Reservation Management Project.