Sri Sivasubramaniya Nadar College of Engineering
Kalavakkam – 603 110
(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Information Technology

UIT1312 – DATABASE MANAGEMENT SYSTEMS AND APPLICATIONS LABORATORY

Mini Project
2021-2022

III SEMESTER, IT
INSTANCE A

RAILWAY ENQUIRY SYSTEM

TEAM MEMBERS:

1. ANUSH RAJAGOPALAN -205002012
2. MAHALAKSHMI M – 205002046
3. SAI SHANMAT -205002077

PROBLEM STATEMENT:

Design a database solution for the railway network. The enquiry system should have the following information:

- Station names
- Train IDs with names
- Weekly Schedules of the trains
- Ticket Availability
- Cost of trip

The train schedules should have information on the stations from where the train starts and by when it reaches the destination. It should also include information about all stations it passes through during its journey.

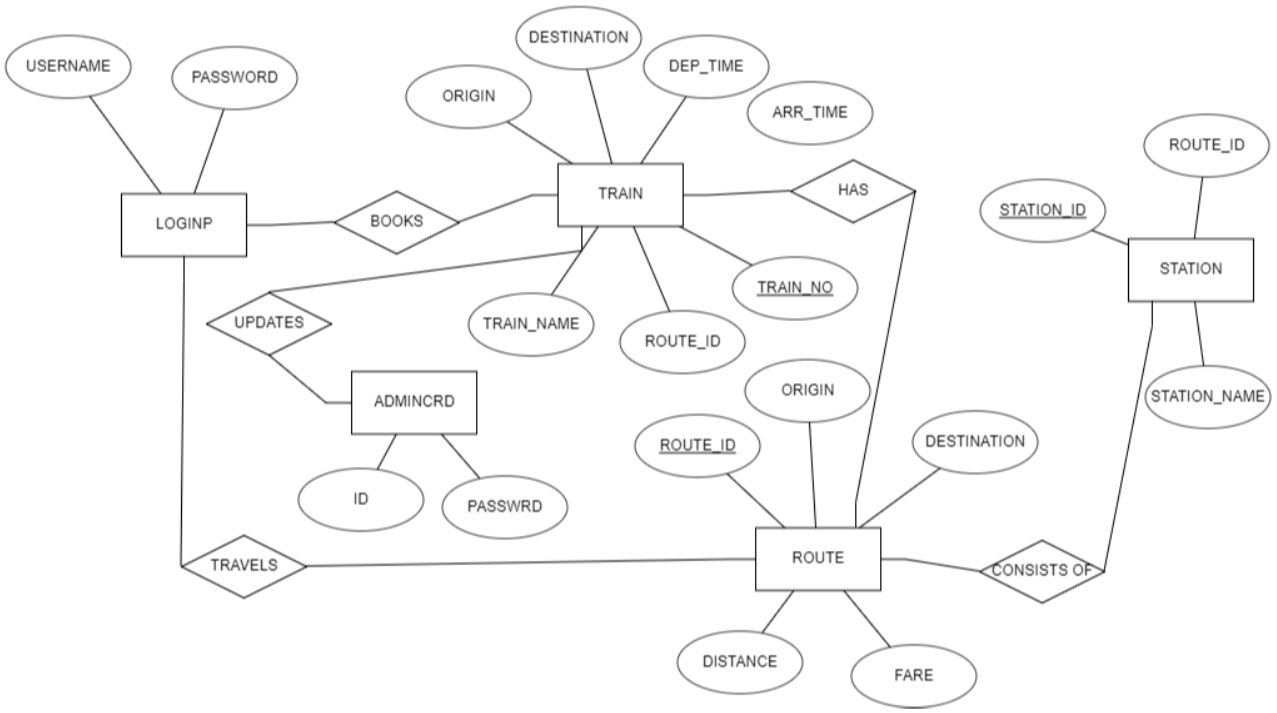
REQUIREMENT ANALYSIS:

Given a route ID, we need to display the available trains, available routes, available stations that a train passes through.

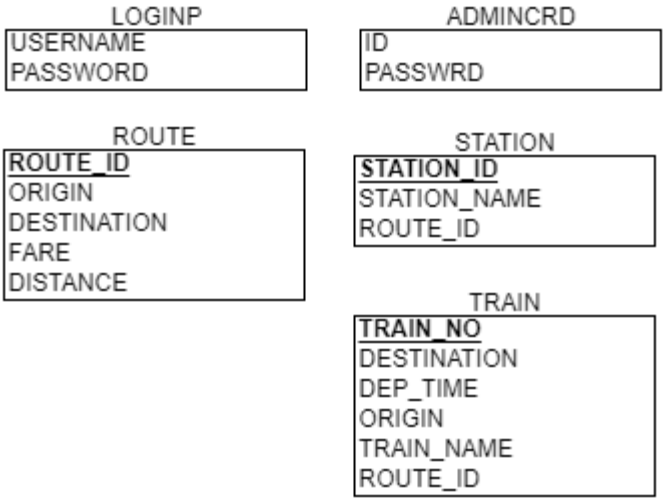
REAL TIME APPLICATION:

In real time, we save the details of users, their booking requirements and also keep track of their past bookings, upcoming bookings, transaction history.

ER DIAGRAM:



SCHEMA DIAGRAM:



NORMALIZATION:

LOGINP={ USERNAME,PASSWORD }

- **1NF NOMALIZATION:**

  No Multivalued Attributes. So, there is no need to do 3NF

- **2NF NOMALIZATION:**

  There is no partial dependency. So, there is no need to do 2NF.

- **3NF NOMALIZATION:**

  There is no transitive dependency. So, there is no need to do 3NF.

ADMINCRD={ ID,PASSWRD }

- **1NF NOMALIZATION:**

  No Multivalued Attributes. So, there is no need to do 3NF

- **2NF NOMALIZATION:**

  There is no partial dependency. So, there is no need to do 2NF.

- **3NF NOMALIZATION:**

  There is no transitive dependency. So, there is no need to do 3NF.

STATION={ STATION_ID, ROUTE_ID,STATION_NAME }

- **1NF NOMALIZATION:**

  No Multivalued Attributes. So, there is no need to do 3NF

- **2NF NOMALIZATION:**

  There is no partial dependency. So, there is no need to do 2NF.

- **3NF NOMALIZATION:**

  There is no transitive dependency. So, there is no need to do 3NF.

ROUTE={ ROUTE_ID, ORIGIN,DESTINATION,DISTANCE,FARE }

- **1NF NOMALIZATION:**

  No Multivalued Attributes. So, there is no need to do 3NF

- **2NF NOMALIZATION:**

  There is no partial dependency. So, there is no need to do 2NF.

- **3NF NOMALIZATION:**

  There is no transitive dependency. So, there is no need to do 3NF.

- TRAIN = { TRAIN _NO,ROUTE_ID, TRAIN_NAME,DEP_TIME,ARR_TIME, ORIGIN,DESTINATION }

- **1NF NOMALIZATION:**

  No Multivalued Attributes. So, there is no need to do 3NF

- **2NF NOMALIZATION:**

  There is no partial dependency. So, there is no need to do 2NF.

- **3NF NOMALIZATION:**

  There is no transitive dependency. So, there is no need to do 3NF.

SQL COMMANDS:

1. ROUTE TABLE

```
CREATE TABLE ROUTE(
    -> ROUTE_ID INT(5) PRIMARY KEY,
    -> ORIGIN CHAR(20),
    -> DESTINATION CHAR(20),
    -> FARE INT(5),
    -> DISTANCE INT(5));
```

```
+-------------+----------+------+-----+---------+-------+
| Field       | Type     | Null | Key | Default | Extra |
+-------------+----------+------+-----+---------+-------+
| ROUTE_ID    | int      | NO   | PRI | NULL    |       |
| ORIGIN      | char(20) | YES  |     | NULL    |       |
| DESTINATION | char(20) | YES  |     | NULL    |       |
| FARE        | int      | YES  |     | NULL    |       |
| DISTANCE    | int      | YES  |     | NULL    |       |
+-------------+----------+------+-----+---------+-------+
```

2. STATION TABLE:

```
CREATE TABLE STATION (
    -> ROUTE_ID INT(5),
    -> STATION_ID INT(5),
    -> STATION_NAME CHAR(20),
    -> FOREIGN KEY(ROUTE_ID) REFERENCES ROUTE(ROUTE_ID));
```

```
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| ROUTE_ID     | int      | YES  | MUL | NULL    |       |
| STATION_ID   | int      | YES  |     | NULL    |       |
| STATION_NAME | char(20) | YES  |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
```

3. TRAIN TABLE:

```
CREATE TABLE TRAIN(
    -> TRAIN_NO INT(10) PRIMARY KEY,
    -> ROUTE_ID INT(5),
    -> TRAIN_NAME CHAR(20),
    -> ORIGIN CHAR(20),
    -> DESTINATION CHAR(20),
    -> DEP_TIME CHAR(15),
    -> ARR_TIME CHAR(15),
    -> TRAVEL_DATE CHAR(15),
    -> CAPACITY INT(5),
    -> INDEX(ROUTE_ID),
    -> FOREIGN KEY(ROUTE_ID) REFERENCES ROUTE(ROUTE_ID));
```

```
+-------------+----------+------+-----+---------+-------+
| Field       | Type     | Null | Key | Default | Extra |
+-------------+----------+------+-----+---------+-------+
| TRAIN_NO    | int      | NO   | PRI | NULL    |       |
| ROUTE_ID    | int      | YES  | MUL | NULL    |       |
| TRAIN_NAME  | char(20) | YES  |     | NULL    |       |
| ORIGIN      | char(20) | YES  |     | NULL    |       |
| DESTINATION | char(20) | YES  |     | NULL    |       |
| DEP_TIME    | char(15) | YES  |     | NULL    |       |
| ARR_TIME    | char(15) | YES  |     | NULL    |       |
| TRAVEL_DATE | char(15) | YES  |     | NULL    |       |
| CAPACITY    | int      | YES  |     | NULL    |       |
+-------------+----------+------+-----+---------+-------+
```

4. PASSENGER LOGIN TABLE (LOGINP):

```
CREATE TABLE LOGINP(
    -> USERNAME_CHAR(20),
    -> PASSWORD CHAR(20));
```

```
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| USERNAME | char(20) | YES  |     | NULL    |       |
| PASSWORD | char(20) | YES  |     | NULL    |       |
+----------+----------+------+-----+---------+-------+
```

5. ADMIN LOGIN TABLE (ADMINCRD):

```
CREATE TABLE ADMINCRD(
    -> ID_CHAR(20),
    -> PASSWRD CHAR(20));
```

```
+---------+----------+------+-----+---------+-------+
| Field   | Type     | Null | Key | Default | Extra |
+---------+----------+------+-----+---------+-------+
| id      | int      | YES  |     | NULL    |       |
| passwrd | char(30) | YES  |     | NULL    |       |
+---------+----------+------+-----+---------+-------+
```

CODE AND IMPLEMENTATION:

```
import mysql.connector
import tkinter as tk
from tkinter import *
from datetime import date
from datetime import time

def connectionfunc():

cnxn=mysql.connector.connect(host="localhost",user="root",pas
swd="Anush@29msgo",database="mysql")
    c = cnxn.cursor()
    return cnxn, c

def passengerlogin():
    def newpassengerlogin():
        def newpass():
            cnxn, c =connectionfunc()
            usr=str(username.get())
            pas=str(password.get())

            add=("INSERT INTO loginp"
                    "(username,password) "
                    "VALUES (%s, %s)")
            data=(usr,pas)
            c.execute(add,data)
            cnxn.commit()
            cnxn.close()
            username.delete(0, END)
            password.delete(0, END)
            scs=tk.Label(root,text="Registration
Successful!",fg='green',bg="white")
            scs.place(x=140, y= 270)
            displaypassenger()

        cnxn, c =connectionfunc()
        root = tk.Tk()
        root.geometry("500x500")
        root.title("NEW PASSENGER LOGIN")
        root.configure(bg="#209aa7")
        firstrow = tk.Label(root, text="Username",
fg='#cc0000',bg='#ffd700' )
        firstrow.place(x=60,y=100,width=90,height=30)
        username = tk.Entry(root,bg="#ffffff")
        username.place(x=190, y=100, width=150,height=30)
        secrow = tk.Label(root,
text="Password",fg='#cc0000',bg='#ffd700')
        secrow.place(x=60,y=140,width=90,height=30)
        password = tk.Entry(root,show="*" ,bg="#ffffff")
        password.place(x=190, y=140, width=150,height=30)
```

```
        btn = tk.Button(root, text="Enter", bg='#e0bed3' ,
fg='#cc0000', command=newpass)
        btn.place(x=140,y=200,width=100,height=30)
        root.mainloop()

    def expassengerlogin():
        def expass():
            flag=0
            usr=str(eusername.get())
            pas=str(epassword.get())
            cnxn, c =connectionfunc()
            chk=("SELECT * FROM LOGINP")
            #dt=(usr,pas)
            c.execute(chk)
            for i in c:
                if usr==i[0] and pas==i[1]:
                    flag=1
                    scs=tk.Label(root,text="Login
Successful!",fg='green',bg="white")
                    scs.place(x=140, y= 270)
                    displaypassenger()
            if flag==0:
                scs=tk.Label(root,text="Login failed! RE
ENTER",fg='red',bg="white")
                scs.place(x=140, y= 270)
            cnxn.commit()
            cnxn.close()
            eusername.delete(0, END)
            epassword.delete(0, END)

        root = tk.Tk()
        root.geometry("500x500")
        root.title("EXISTING PASSENGER LOGIN")
        root.configure(bg="#209aa7")
        firstrow = tk.Label(root, text="Username",
fg='#cc0000',bg='#ffd700' )
        firstrow.place(x=60,y=100,width=90,height=30)
        eusername = tk.Entry(root,bg="#ffffff")
        eusername.place(x=190, y=100, width=150,height=30)
        secrow = tk.Label(root,
text="Password",fg='#cc0000',bg='#ffd700')
        secrow.place(x=60,y=140,width=90,height=30)
        epassword = tk.Entry(root,show="*", bg="#ffffff")
        epassword.place(x=190, y=140, width=150,height=30)
        btn = tk.Button(root, text="Enter", bg='#e0bed3' ,
fg='#cc0000', command=expass)
        btn.place(x=140,y=200,width=100,height=30)
        root.mainloop()

    root=tk.Tk()
    root.geometry("500x500")
```

```
    root.configure(bg="#209aa7")
    root.title(" PASSENGER LOGIN")
    nbtn=tk.Button(root,text="New Passenger Login",
bg='#ffd700' , fg='#cc0000',command=newpassengerlogin)
    nbtn.place(x=90,y=110,width=300,height=40)
    pbtn=tk.Button(root,text="Existing Passenger Login",
bg='#ffd700' , fg='#cc0000',command=expassengerlogin)
    pbtn.place(x=90,y=200,width=300,height=40)

def adminlogin():
    def adminc():
        flag=0
        ausr=int(aid.get())
        apas=str(apassword.get())
        cnxn, c =connectionfunc()
        ahk=("SELECT * FROM ADMINCRD")
        c.execute(ahk)

        for j in c:
            if ausr==j[0] and apas==j[1]:
                flag=1
                acs=tk.Label(root,text="Login
Successful!",fg='green',bg="white")
                acs.place(x=140, y= 270)
                insertadmin()
        if flag==0:
            bcs=tk.Label(root,text="Login failed! RE
ENTER",fg='red',bg="white")
            bcs.place(x=140, y= 270)
        cnxn.commit()
        cnxn.close()
        aid.delete(0, END)
        apassword.delete(0, END)

    root = tk.Tk()
    root.geometry("500x500")
    root.title("ADMIN LOGIN")
    root.configure(bg="#209aa7")
    firstrow = tk.Label(root, text="Admin ID",
fg='#cc0000',bg='#ffd700' )
    firstrow.place(x=60,y=100,width=110,height=30)
    aid = tk.Entry(root,bg="#ffffff")
    aid.place(x=200, y=100, width=150,height=30)
    secrow = tk.Label(root, text="Admin
password",fg='#cc0000',bg='#ffd700')
    secrow.place(x=60,y=140,width=110,height=30)
    apassword = tk.Entry(root,show="*", bg="#ffffff")
    apassword.place(x=200, y=140, width=150,height=30)
    btn = tk.Button(root, text="Enter", bg='#e0bed3' ,
fg='#cc0000', command=adminc)
    btn.place(x=140,y=200,width=100,height=30)
```

```
    root.mainloop()


def insertadmin():
    def insertroutemain():
        def insertroute():
            rid = int(route_id.get())
            ori = str(origin.get())
            des= str(dest.get())
            fa=int(fare.get())
            dis=int(dist.get())
            cnxn, c =connectionfunc()
            add=("INSERT INTO ROUTE "

"(ROUTE_ID,ORIGIN,DESTINATION,FARE,DISTANCE) "
                    "VALUES (%s, %s,%s,%s,%s)")
            data=(rid,ori,des,fa,dis)
            c.execute(add,data)
            cnxn.commit()
            cnxn.close()
            route_id.delete(0, END)
            origin.delete(0, END)
            dest.delete(0, END)
            fare.delete(0, END)
            dist.delete(0,END)

        cnxn, c =connectionfunc()
        root = tk.Tk()
        root.geometry("500x500")
        root.title("ADD ROUTE DETAILS")
        root.configure(bg="#209aa7")

        box1 = tk.Label(root, text="Route
ID",fg="#cc0000",bg="#ffd700")
        box1.place(x=90,y=80,width=80,height=30)
        route_id = tk.Entry(root,bg='#ffffff')
        route_id.place(x=200,y=80,width=150,height=30)

        box2 = tk.Label(root,
text="Origin",fg="#cc0000",bg="#ffd700")
        box2.place(x=90,y=130,width=80,height=30)
        origin = tk.Entry(root,bg='#ffffff')
        origin.place(x=200,y=130,width=150,height=30)

        box3 = tk.Label(root,
text="Destination",fg="#cc0000",bg="#ffd700")
        box3.place(x=90,y=180,width=80,height=30)
        dest = tk.Entry(root,bg='#ffffff')
        dest.place(x=200,y=180,width=150,height=30)

        box4 = tk.Label(root,
```

```
text="Fare",fg="#cc0000",bg="#ffd700")
        box4.place(x=90,y=230,width=80,height=30)
        fare = tk.Entry(root,bg='#ffffff')
        fare.place(x=200,y=230,width=150,height=30)

        box5 = tk.Label(root,
text="Distance",fg="#cc0000",bg="#ffd700")
        box5.place(x=90,y=280,width=80,height=30)
        dist = tk.Entry(root,bg='#ffffff')
        dist.place(x=200,y=280,width=150,height=30)

        btn = tk.Button(root, text="Enter", bg='#e0bed3' ,
fg='#cc0000', command=insertroute)
        btn.place(x=150,y=350,width=150,height=30)
        root.mainloop()

    def insertstationmain():
        def insertstation():
            rid = int(route_id.get())
            sid = str(station_id.get())
            sname= str(station_name.get())
            cnxn, c =connectionfunc()
            add=("INSERT INTO STATION "
                    "(ROUTE_ID,STATION_ID,STATION_NAME) "
                    "VALUES (%s, %s,%s)")
            data=(rid,sid,sname)
            c.execute(add,data)
            cnxn.commit()
            cnxn.close()
            route_id.delete(0, END)
            station_id.delete(0, END)
            station_name.delete(0, END)


        cnxn, c =connectionfunc()
        root = tk.Tk()
        root.geometry("500x500")
        root.title("ADD STATION DETAILS")
        root.configure(bg="#209aa7")

        box1 = tk.Label(root, text="Route
ID",fg="#cc0000",bg="#ffd700")
        box1.place(x=90,y=80,width=80,height=30)
        route_id = tk.Entry(root,bg='#ffffff')
        route_id.place(x=200,y=80,width=150,height=30)

        box2 = tk.Label(root, text="Station
ID",fg="#cc0000",bg="#ffd700")
        box2.place(x=90,y=130,width=80,height=30)
        station_id = tk.Entry(root,bg='#ffffff')
        station_id.place(x=200,y=130,width=150,height=30)
```

```
        box3 = tk.Label(root, text="Station
Name",fg="#cc0000",bg="#ffd700")
        box3.place(x=90,y=180,width=80,height=30)
        station_name = tk.Entry(root,bg='#ffffff')
        station_name.place(x=200,y=180,width=150,height=30)


        btn = tk.Button(root, text="Enter", bg='#e0bed3' ,
fg='#cc0000', command=insertstation)
        btn.place(x=150,y=300,width=150,height=30)
        root.mainloop()


    def inserttrainmain():
        def inserttrain():
            tid=int(t_no.get())
            rid = int(r_id.get())
            tna = str(t_name.get())
            o= str(origin.get())
            d= str(desti.get())
            dtime= str(dep_time.get())
            atime= str(arr_time.get())
            date1= str(date.get())
            cap= int(capacity.get())
            cnxn, c =connectionfunc()
            add=("INSERT INTO TRAIN "

"(TRAIN_NO,ROUTE_ID,TRAIN_NAME,ORIGIN,DESTINATION,DEP_TIME,AR
R_TIME,TRAVEL_DATE,CAPACITY) "
                 "VALUES (%s, %s,%s,%s, %s,%s,%s, %s,%s)")
            data=(tid,rid,tna,o,d,dtime,atime,date1,cap)
            c.execute(add,data)
            cnxn.commit()
            cnxn.close()
            t_no.delete(0, END)
            r_id.delete(0, END)
            t_name.delete(0, END)
            origin.delete(0, END)
            desti.delete(0, END)
            dep_time.delete(0, END)
            arr_time.delete(0, END)
            date.delete(0, END)
            capacity.delete(0, END)

        cnxn, c =connectionfunc()
        root = tk.Tk()
        root.geometry("500x500")
        root.title("ADD TRAIN DETAILS")
        root.configure(bg="#209aa7")
```

```
        box1 = tk.Label(root, text="Train
No.",fg="#cc0000",bg="#ffd700")
        box1.place(x=40,y=20,width=100,height=30)
        t_no = tk.Entry(root,bg='#ffffff')
        t_no.place(x=170,y=20,width=200,height=30)

        box2 = tk.Label(root, text="Route
ID",fg="#cc0000",bg="#ffd700")
        box2.place(x=40,y=60,width=100,height=30)
        r_id = tk.Entry(root,bg='#ffffff')
        r_id.place(x=170,y=60,width=200,height=30)

        box3 = tk.Label(root, text="Train
Name",fg="#cc0000",bg="#ffd700")
        box3.place(x=40,y=100,width=100,height=30)
        t_name = tk.Entry(root,bg='#ffffff')
        t_name.place(x=170,y=100,width=200,height=30)

        box4 = tk.Label(root,
text="Origin",fg="#cc0000",bg="#ffd700")
        box4.place(x=40,y=140,width=100,height=30)
        origin = tk.Entry(root,bg='#ffffff')
        origin.place(x=170,y=140,width=200,height=30)

        box5 = tk.Label(root,
text="Destination",fg="#cc0000",bg="#ffd700")
        box5.place(x=40,y=180,width=100,height=30)
        desti = tk.Entry(root,bg='#ffffff')
        desti.place(x=170,y=180,width=200,height=30)

        box6 = tk.Label(root, text="Departure
Time",fg="#cc0000",bg="#ffd700")
        box6.place(x=40,y=220,width=100,height=30)
        dep_time = tk.Entry(root,bg='#ffffff')
        dep_time.place(x=170,y=220,width=200,height=30)

        box7 = tk.Label(root, text="Arrival
Time",fg="#cc0000",bg="#ffd700")
        box7.place(x=40,y=260,width=100,height=30)
        arr_time = tk.Entry(root,bg='#ffffff')
        arr_time.place(x=170,y=260,width=200,height=30)

        box8 = tk.Label(root,
text="Date",fg="#cc0000",bg="#ffd700")
        box8.place(x=40,y=300,width=100,height=30)
        date = tk.Entry(root,bg='#ffffff')
        date.place(x=170,y=300,width=200,height=30)

        box9 = tk.Label(root,
text="Capacity",fg="#cc0000",bg="#ffd700")
        box9.place(x=40,y=340,width=100,height=30)
```

```
        capacity = tk.Entry(root,bg='#ffffff')
        capacity.place(x=170,y=340,width=200,height=30)

        btn = tk.Button(root, text="Enter", bg='#e0bed3' ,
fg='#cc0000', command=inserttrain)
        btn.place(x=120,y=400,width=100,height=30)
        root.mainloop()

    cnxn, c =connectionfunc()
    root = tk.Tk()
    root.geometry("500x500")
    root.title("ADMIN UPDATE OPTIONS")
    root.configure(bg="#209aa7")
    btn1 = tk.Button(root, text="Update Route", bg='#ffd700'
, fg='#cc0000', command=insertroutemain)
    btn1.place(x=90,y=110,width=300,height=40)
    btn2 = tk.Button(root, text="Update Station",
bg='#ffd700' , fg='#cc0000', command=insertstationmain)
    btn2.place(x=90,y=200,width=300,height=40)
    btn3 = tk.Button(root, text="Update Train", bg='#ffd700'
, fg='#cc0000', command=inserttrainmain)
    btn3.place(x=90,y=290,width=300,height=40)
    root.mainloop()

def displaypassenger():
    def display_routes():
        root = tk.Tk()
        root.title("AVAILABLE ROUTES")
        root.geometry('600x600')
        root.configure(bg="#209aa7")
        root.resizable(width=False, height=False)
        label1=tk.Label(root, text="Route
ID",fg="#cc0000",bg="#ffd700")
        label1.place(x=10,y=30,width=100,height=30)
        label2=tk.Label(root,
text="Origin",fg="#cc0000",bg="#ffd700")
        label2.place(x=130,y=30,width=100,height=30)
        label3=tk.Label(root,
text="Destination",fg="#cc0000",bg="#ffd700")
        label3.place(x=250,y=30,width=100,height=30)
        label4=tk.Label(root,
text="Distance",fg="#cc0000",bg="#ffd700")
        label4.place(x=370,y=30,width=100,height=30)
        label5=tk.Label(root,
text="Fare",fg="#cc0000",bg="#ffd700")
        label5.place(x=490,y=30,width=100,height=30)

        lb1=tk.Listbox(root,bg='#ffffff')
        lb1.place(x=10,y=70,width=100,height=500)
        lb2=tk.Listbox(root,bg='#ffffff')
        lb2.place(x=130,y=70,width=100,height=500)
```

```
        lb3=tk.Listbox(root,bg='#ffffff')
        lb3.place(x=250,y=70,width=100,height=500)
        lb4=tk.Listbox(root,bg='#ffffff')
        lb4.place(x=370,y=70,width=100,height=500)
        lb5=tk.Listbox(root,bg='#ffffff')
        lb5.place(x=490,y=70,width=100,height=500)

        cnxn, c =connectionfunc()
        c.execute("SELECT * FROM ROUTE")
        for i in c:
            lb1.insert('end', i[0])
            lb2.insert('end', i[1])
            lb3.insert('end', i[2])
            lb4.insert('end', i[3])
            lb5.insert('end', i[4])
        root.mainloop()
        cnxn.close()

    def display_stations():
        root = tk.Tk()
        root.title("STATIONS COVERED")
        root.geometry('600x600')
        root.configure(bg="#209aa7")
        root.resizable(width=False, height=False)
        label1=tk.Label(root, text="Route
ID",fg="#cc0000",bg="#ffd700")
        label1.place(x=70,y=30,width=100,height=30)
        label2=tk.Label(root, text="Station
ID",fg="#cc0000",bg="#ffd700")
        label2.place(x=240,y=30,width=100,height=30)
        label3=tk.Label(root, text="Station
Name",fg="#cc0000",bg="#ffd700")
        label3.place(x=410,y=30,width=100,height=30)

        lb1=tk.Listbox(root,bg='#ffffff')
        lb1.place(x=70,y=70,width=100,height=500)
        lb2=tk.Listbox(root,bg='#ffffff')
        lb2.place(x=240,y=70,width=100,height=500)
        lb3=tk.Listbox(root,bg='#ffffff')
        lb3.place(x=410,y=70,width=100,height=500)

        cnxn, c =connectionfunc()
        c.execute("SELECT * FROM STATION")
        for i in c:
            lb1.insert('end', i[0])
            lb2.insert('end', i[1])
            lb3.insert('end', i[2])
        root.mainloop()
        cnxn.close()

    def outer_calc():
```

```
        def calc():

            root = tk.Tk()
            root.title("BOOKING DETAILS")
            root.geometry('600x600')
            root.configure(bg="#209aa7")
            root.resizable(width=False, height=False)
            label1=tk.Label(root, text="Passenger
Name",fg="#cc0000",bg="#ffd700")
            label1.place(x=40,y=70,width=150,height=30)
            label3=tk.Label(root, text="Train
No.",fg="#cc0000",bg="#ffd700")
            label3.place(x=40,y=120,width=150,height=30)
            label4=tk.Label(root, text="Route
ID",fg="#cc0000",bg="#ffd700")
            label4.place(x=40,y=170,width=150,height=30)
            label5=tk.Label(root, text="No. of
Passengers",fg="#cc0000",bg="#ffd700")
            label5.place(x=40,y=220,width=150,height=30)
            label6=tk.Label(root,
text="Origin",fg="#cc0000",bg="#ffd700")
            label6.place(x=40,y=270,width=150,height=30)
            label7=tk.Label(root,
text="Destination",fg="#cc0000",bg="#ffd700")
            label7.place(x=40,y=320,width=150,height=30)
            label8=tk.Label(root,
text="Date",fg="#cc0000",bg="#ffd700")
            label8.place(x=40,y=370,width=150,height=30)
            label9=tk.Label(root,
text="Time",fg="#cc0000",bg="#ffd700")
            label9.place(x=40,y=420,width=150,height=30)
            label10=tk.Label(root, text="Total
Fare",fg="#cc0000",bg="#ffd700")
            label10.place(x=40,y=470,width=150,height=30)

            lb1=tk.Listbox(root,bg='#ffffff')
            lb1.place(x=220,y=70,width=250,height=30)
            lb3=tk.Listbox(root,bg='#ffffff')
            lb3.place(x=220,y=120,width=250,height=30)
            lb4=tk.Listbox(root,bg='#ffffff')
            lb4.place(x=220,y=170,width=250,height=30)
            lb5=tk.Listbox(root,bg='#ffffff')
            lb5.place(x=220,y=220,width=250,height=30)
            lb6=tk.Listbox(root,bg='#ffffff')
            lb6.place(x=220,y=270,width=250,height=30)
            lb7=tk.Listbox(root,bg='#ffffff')
            lb7.place(x=220,y=320,width=250,height=30)
            lb8=tk.Listbox(root,bg='#ffffff')
            lb8.place(x=220,y=370,width=250,height=30)
            lb9=tk.Listbox(root,bg='#ffffff')
            lb9.place(x=220,y=420,width=250,height=30)
```

```
            lb10=tk.Listbox(root,bg='#ffffff')
            lb10.place(x=220,y=470,width=250,height=30)
            cnxn ,c=connectionfunc()
            rt=int(ur_id.get())
            no=int(unop.get())
            tid=int(utid.get())
            name=str(namep.get())
            print("name:",name)
            print("rt:",rt)
            fl=0
            cal=("SELECT * FROM ROUTE")
            c.execute(cal)
            for k in c:
                if k[0]==rt:
                    fl=1
                    amt=k[3]*no
            if fl==0:
                print("error")
            lb1.insert('end',name)
            lb3.insert('end', tid)
            lb4.insert('end', rt)
            lb5.insert('end', no)
            trn=("SELECT * FROM TRAIN ")
            c.execute(trn)
            for v in c:
                if v[1]==rt and v[0]==tid:
                    lb6.insert('end', v[3])
                    lb7.insert('end', v[4])
                    lb8.insert('end', v[5])
                    lb9.insert('end',v[6])
            lb10.insert('end', amt)
            cnxn.close()
            rt.delete(0, END)

        root = tk.Tk()
        root.title("DETAILS")
        root.geometry('500x500')
        root.configure(bg="#209aa7")
        root.resizable(width=False, height=False)

        b1 = tk.Label(root, text="Enter Train No. to
Book",fg="#cc0000",bg="#ffd700")
        b1.place(x=100,y=40,width=280,height=30)
        utid = tk.Entry(root,bg='#ffffff')
        utid.place(x=140,y=90,width=200,height=30)

        b1 = tk.Label(root, text="Enter Passenger
Name",fg="#cc0000",bg="#ffd700")
        b1.place(x=100,y=150,width=280,height=30)
        namep = tk.Entry(root,bg='#ffffff')
        namep.place(x=140,y=200,width=200,height=30)
```

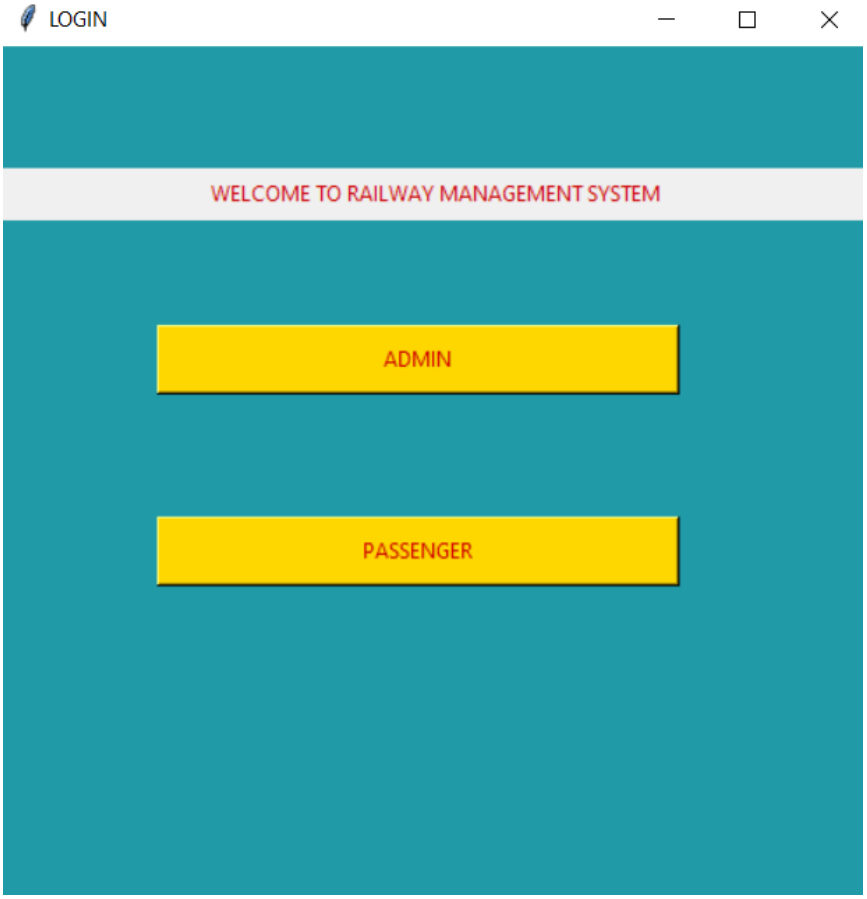
```
        button1=tk.Button(root, text="Display Details",
bg='#ffd700' , fg='#cc0000', command=calc)
        button1.place(x=150,y=300,width=180,height=30)

    def outerdisplay_trains():
        def display_trains():
            root = tk.Tk()
            root.title("AVAILABLE TRAINS")
            root.geometry('1000x700')
            root.configure(bg="#209aa7")
            root.resizable(width=False, height=False)
            label1=tk.Label(root, text="Train
No.",fg="#cc0000",bg="#ffd700")
            label1.place(x=120,y=20,width=100,height=30)
            label2=tk.Label(root, text="Route
ID",fg="#cc0000",bg="#ffd700")
            label2.place(x=10,y=20,width=100,height=30)
            label3=tk.Label(root, text="Train
Name",fg="#cc0000",bg="#ffd700")
            label3.place(x=230,y=20,width=100,height=30)
            label4=tk.Label(root,
text="Origin",fg="#cc0000",bg="#ffd700")
            label4.place(x=340,y=20,width=100,height=30)
            label5=tk.Label(root,
text="Destination",fg="#cc0000",bg="#ffd700")
            label5.place(x=450,y=20,width=100,height=30)
            label6=tk.Label(root,
text="Depart.Time",fg="#cc0000",bg="#ffd700")
            label6.place(x=560,y=20,width=100,height=30)
            label7=tk.Label(root, text="Arrival
Time",fg="#cc0000",bg="#ffd700")
            label7.place(x=670,y=20,width=100,height=30)
            label8=tk.Label(root,
text="Date",fg="#cc0000",bg="#ffd700")
            label8.place(x=780,y=20,width=100,height=30)
            label9=tk.Label(root,
text="Capacity",fg="#cc0000",bg="#ffd700")
            label9.place(x=890,y=20,width=100,height=30)

            lb1=tk.Listbox(root,bg='#ffffff')
            lb1.place(x=10,y=70,width=100,height=510)
            lb2=tk.Listbox(root,bg='#ffffff')
            lb2.place(x=120,y=70,width=100,height=510)
            lb3=tk.Listbox(root,bg='#ffffff')
            lb3.place(x=230,y=70,width=100,height=510)
            lb4=tk.Listbox(root,bg='#ffffff')
            lb4.place(x=340,y=70,width=100,height=510)
            lb5=tk.Listbox(root,bg='#ffffff')
            lb5.place(x=450,y=70,width=100,height=510)
            lb6=tk.Listbox(root,bg='#ffffff')
```

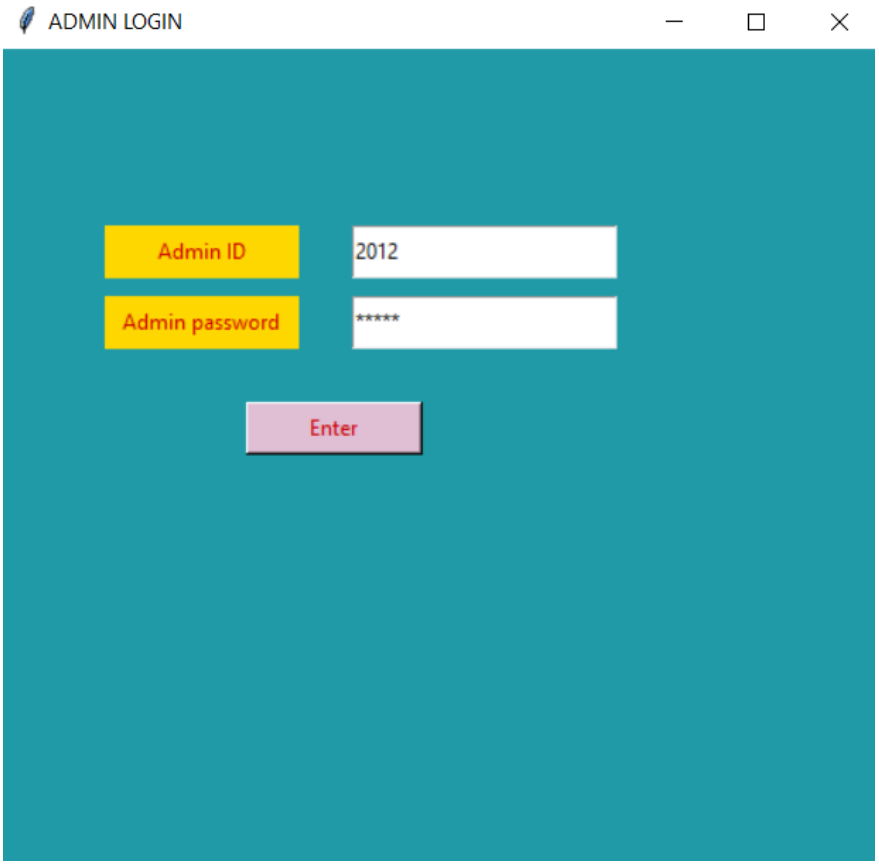
```
            lb6.place(x=560,y=70,width=100,height=510)
            lb7=tk.Listbox(root,bg='#ffffff')
            lb7.place(x=670,y=70,width=100,height=510)
            lb8=tk.Listbox(root,bg='#ffffff')
            lb8.place(x=780,y=70,width=100,height=510)
            lb9=tk.Listbox(root,bg='#ffffff')
            lb9.place(x=890,y=70,width=100,height=510)

            temp=0
            rt=int(ur_id.get())
            print("rt" , rt)
            no=int(unop.get())
            cnxn, c =connectionfunc()
            trn=("SELECT * FROM TRAIN ")
            c.execute(trn)
            for v in c:
                if v[1]==rt:
                    temp=1
                    if( v[8]-no)>0:
                        lb1.insert('end', v[1])
                        lb2.insert('end', v[0])
                        lb3.insert('end', v[2])
                        lb4.insert('end', v[3])
                        lb5.insert('end', v[4])
                        lb6.insert('end', v[5])
                        lb7.insert('end', v[6])
                        lb8.insert('end', v[7])
                        lb9.insert('end', v[8])
                    else:
                        war=tk.Label(root,text="No available
seats!",fg="#cc0000",bg="#ffd700")
                        war.place(x=230,y=600,
width=400,height=30)


            if temp==0:

                label10=tk.Label(root, text="No Trains
Found",fg="#cc0000",bg="#ffd700")

label10.place(x=230,y=600,width=400,height=30)

            root.mainloop()
            cnxn.close()
            ur_id.delete(0, END)
            unop.delete(0, END)

        rt=int(ur_id.get())
        print("rt:",rt)
        display_trains()
        ur_id.delete(0,END)
```

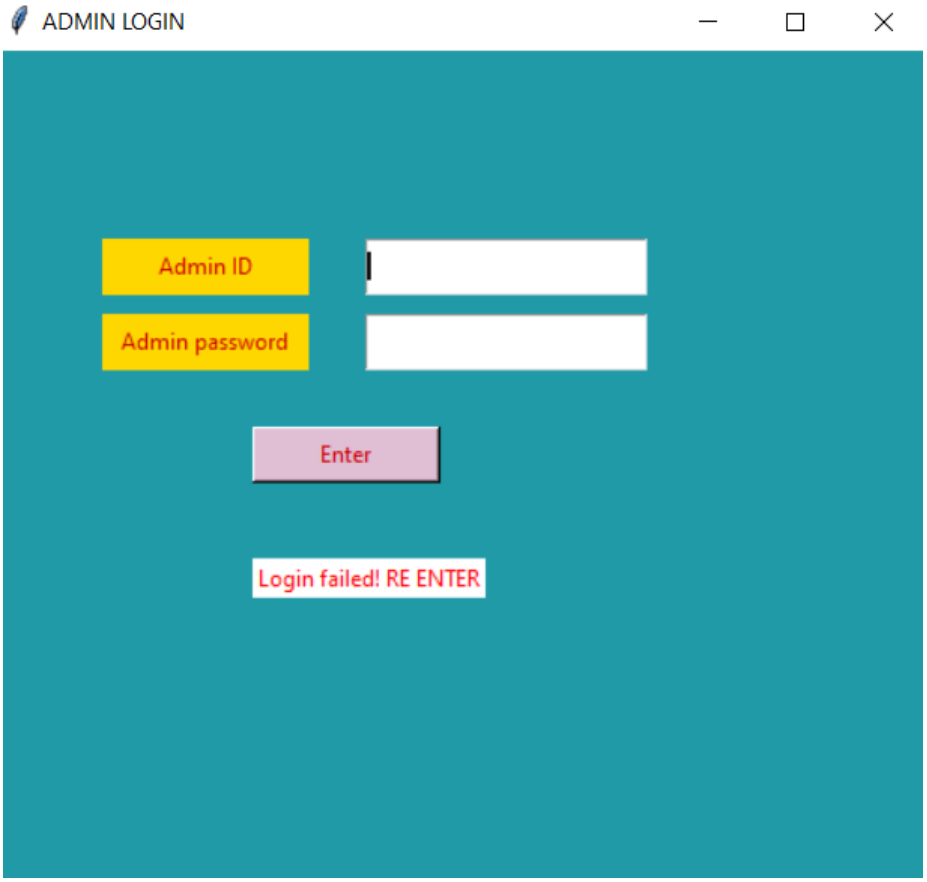
```
    cnxn, c =connectionfunc()
    root = tk.Tk()
    root.geometry("600x600")
    root.title("BOOKING")
    root.configure(bg="#209aa7")
    bttn1 = tk.Button(root, text="Avaialble Routes",
bg='#ffd700' , fg='#cc0000', command=display_routes)
    bttn1.place(x=130,y=20,width=300,height=40)
    bttn2 = tk.Button(root, text="Stations Covered",
bg='#ffd700' , fg='#cc0000', command=display_stations)
    bttn2.place(x=130,y=80,width=300,height=40)

    msg1=tk.Label(root, text="*** Check Available Routes and
Stations Covered Before Booking
***",fg="#cc0000",bg="#ffffff")
    msg1.place(x=0,y=140,width=600,height=30)

    msg2=tk.Label(root, text="Enter Following Details to
Check Trains",fg="#cc0000",bg="#ffd700")
    msg2.place(x=70,y=190,width=450,height=30)

    bx1 = tk.Label(root, text="Route
ID",fg="#cc0000",bg="#ffd700")
    bx1.place(x=90,y=240,width=180,height=30)
    ur_id = tk.Entry(root,bg='#ffffff')
    ur_id.place(x=310,y=240,width=180,height=30)

    bx6 = tk.Label(root, text="No. of
Passengers",fg="#cc0000",bg="#ffd700")
    bx6.place(x=90,y=310,width=180,height=30)
    unop = tk.Entry(root,bg='#ffffff')
    unop.place(x=310,y=310,width=180,height=30)

    bttn3=tk.Button(root, text="Check Trains", bg='#ffd700' ,
fg='#cc0000', command=outerdisplay_trains)
    bttn3.place(x=230,y=450,width=100,height=25)

    bttn4=tk.Button(root, text="Proceed To Next Step",
bg='#ffd700' , fg='#cc0000', command=outer_calc)
    bttn4.place(x=230,y=500,width=150,height=25)



cnxn, c =connectionfunc()
root=tk.Tk()
root.geometry("500x500")
root.title("LOGIN")
root.configure(bg="#209aa7")
msg=tk.Label(root, text="WELCOME TO RAILWAY MANAGEMENT
SYSTEM",fg="#cc0000")
```

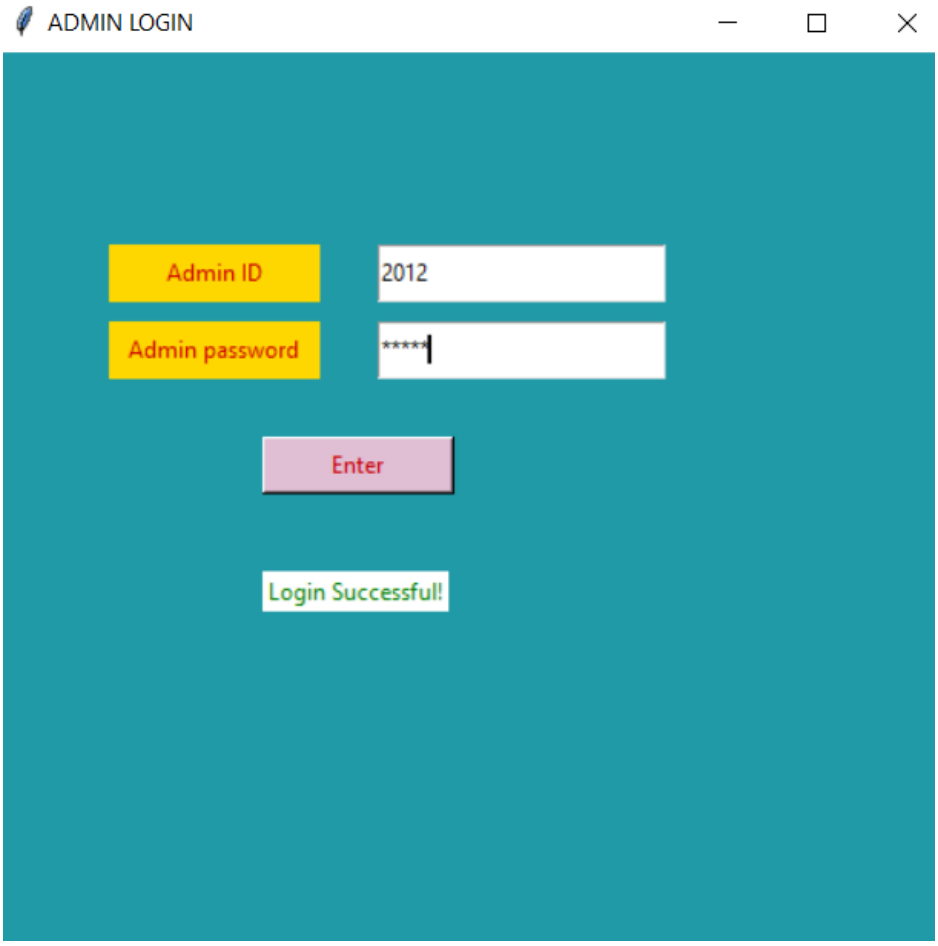
```
msg.place(x=0,y=70,width=500,height=30)
abtn=tk.Button(root,text="ADMIN", bg='#ffd700' ,
fg='#cc0000',command=adminlogin)
abtn.place(x=90,y=160,width=300,height=40)
pbtn=tk.Button(root,text="PASSENGER", bg='#ffd700' ,
fg='#cc0000',command=passengerlogin)
pbtn.place(x=90,y=270,width=300,height=40)
root.mainloop()
```

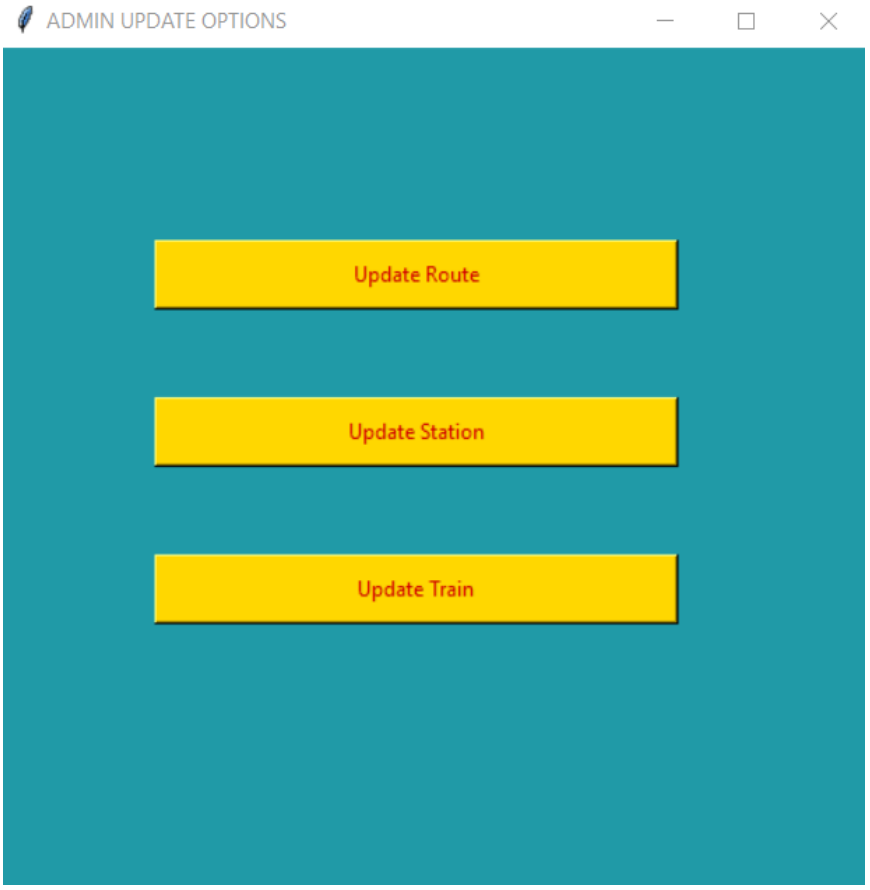
OUTPUT SCREENSHOTS:

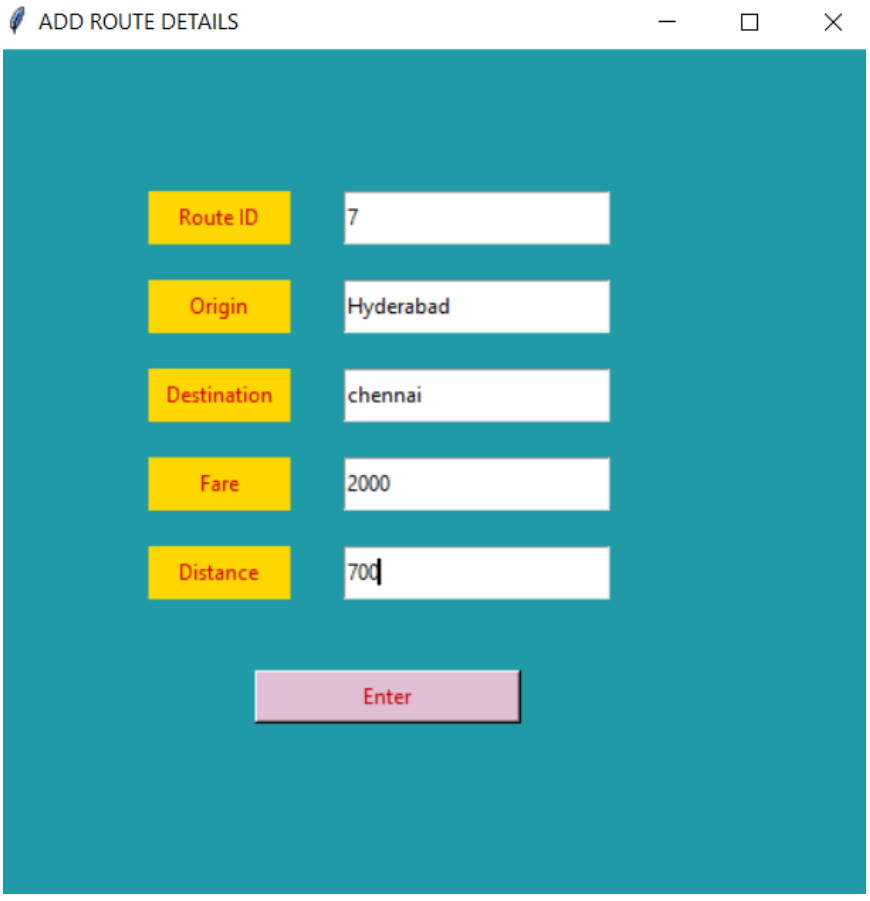


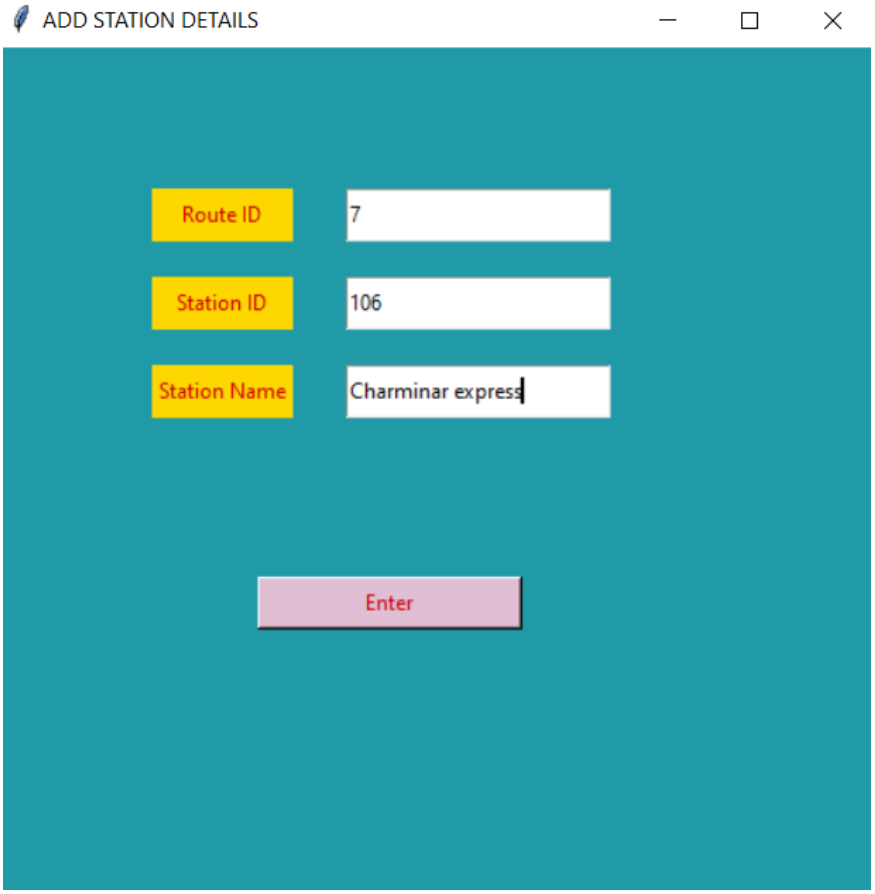



ADMIN LOGIN
Admin ID
Admin password
Enter
Login failed! RE ENTER

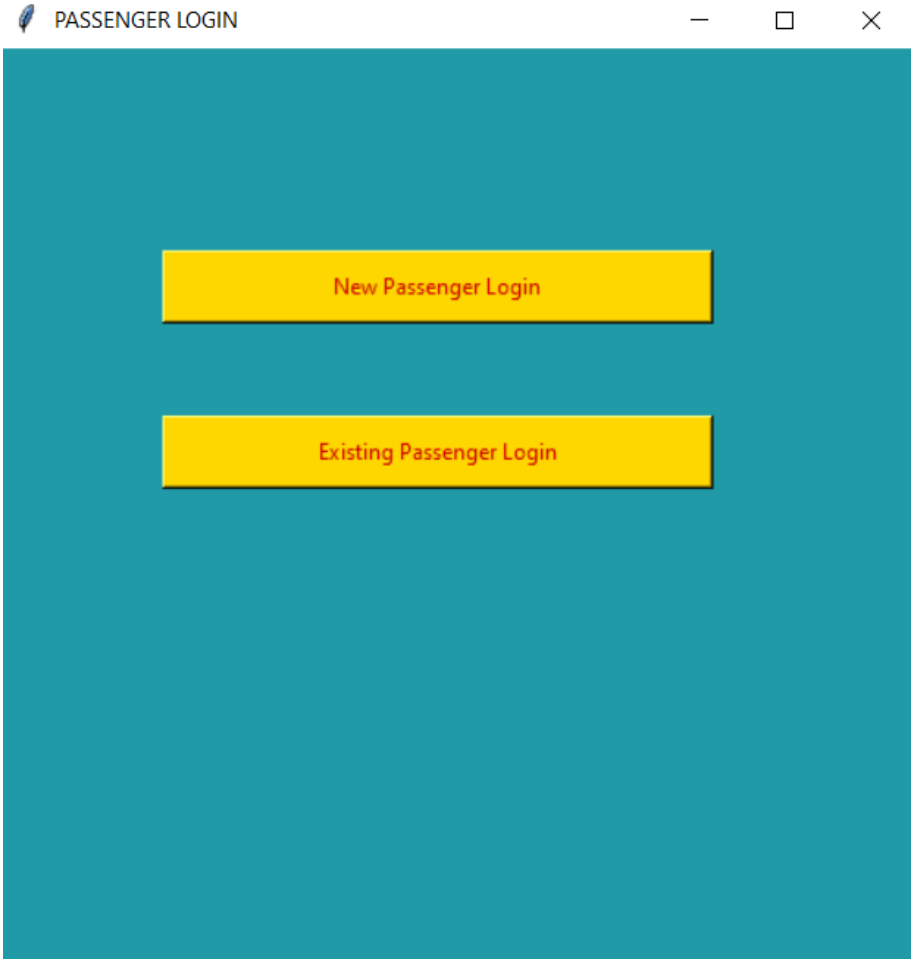

ADMIN LOGIN
Admin ID
2012
Admin password
*****
Enter
Login Successful!

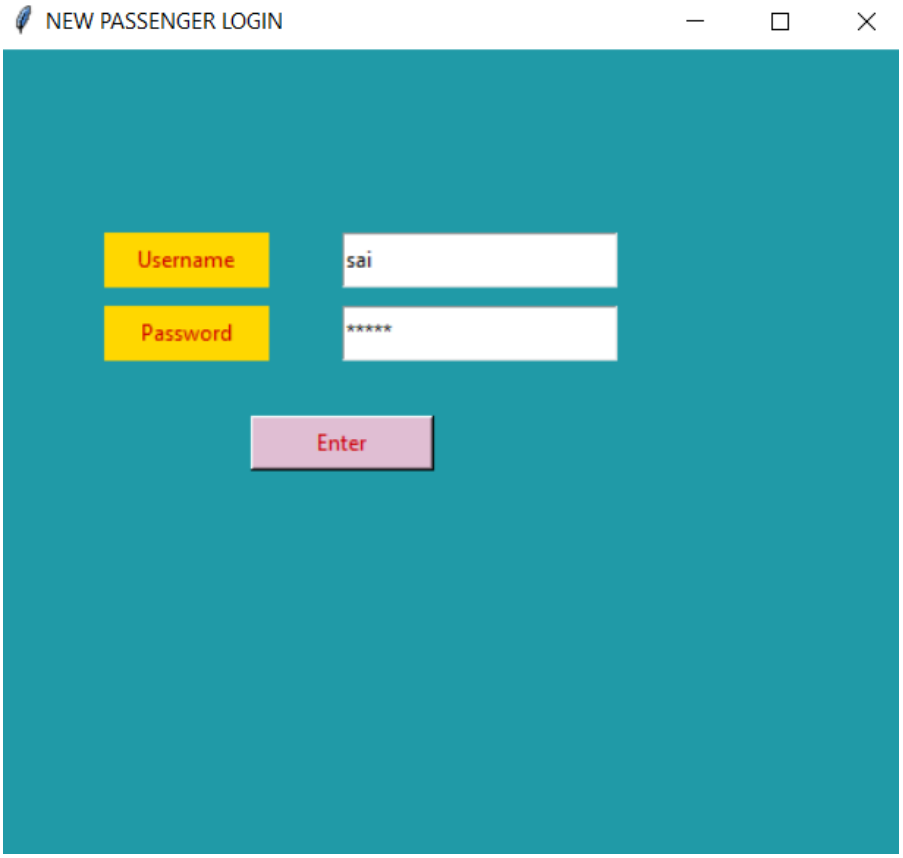

ADMIN UPDATE OPTIONS
Update Route
Update Station
Update Train

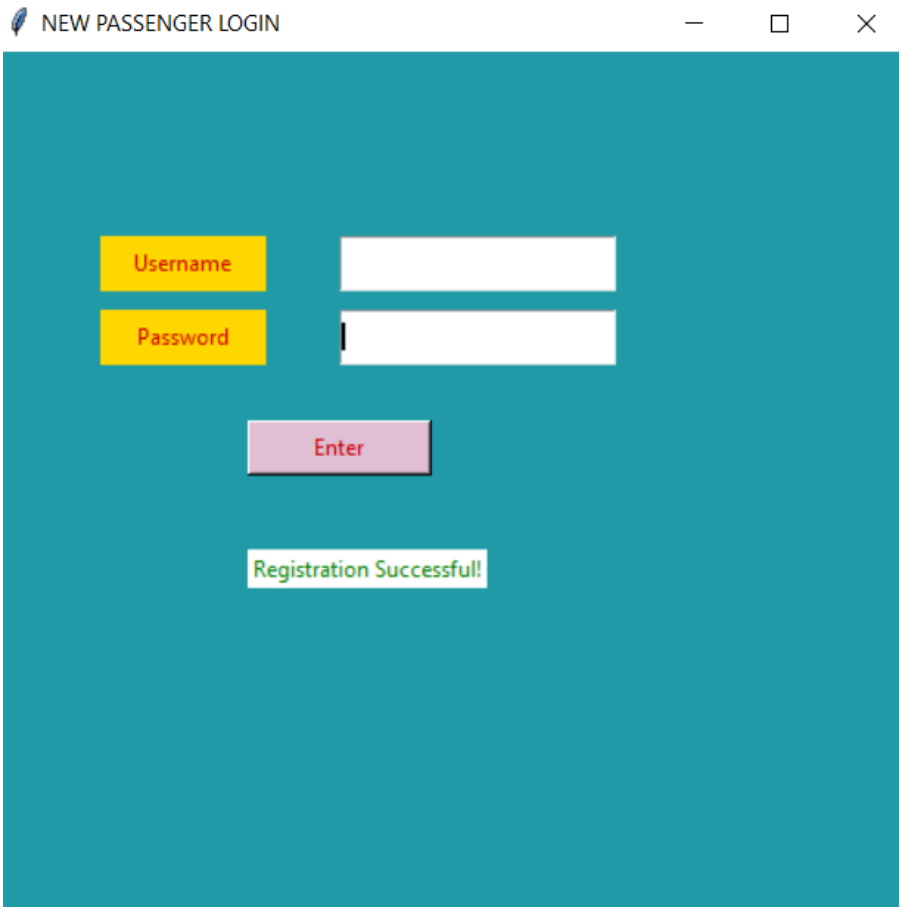

ADD ROUTE DETAILS
Route ID
7
Origin
Hyderabad
Destination
chennai
Fare
2000
Distance
700
Enter

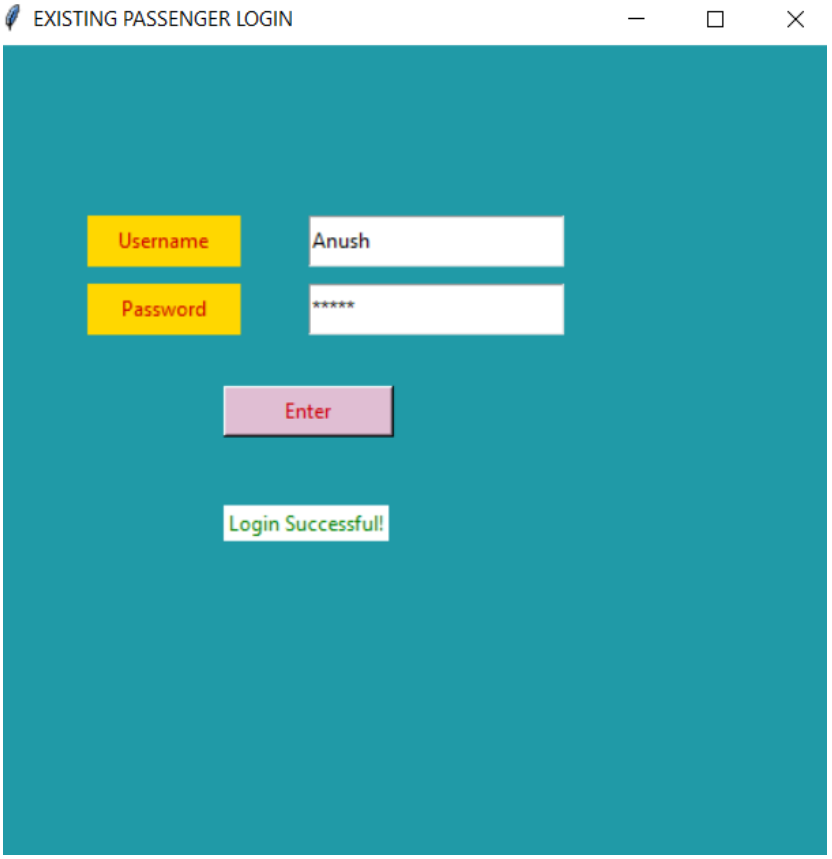

ADD STATION DETAILS
Route ID
7
Station ID
106
Station Name
Charminar express
Enter


ADD TRAIN DETAILS
Train No.
103
Route ID
5
Train Name
super fast express
Origin
chennai
Destination
kolkata
Departure Time
5 AM
Arrival Time
9 PM
Date
20/01/2022
Capacity
100
Enter


PASSENGER LOGIN
New Passenger Login
Existing Passenger Login

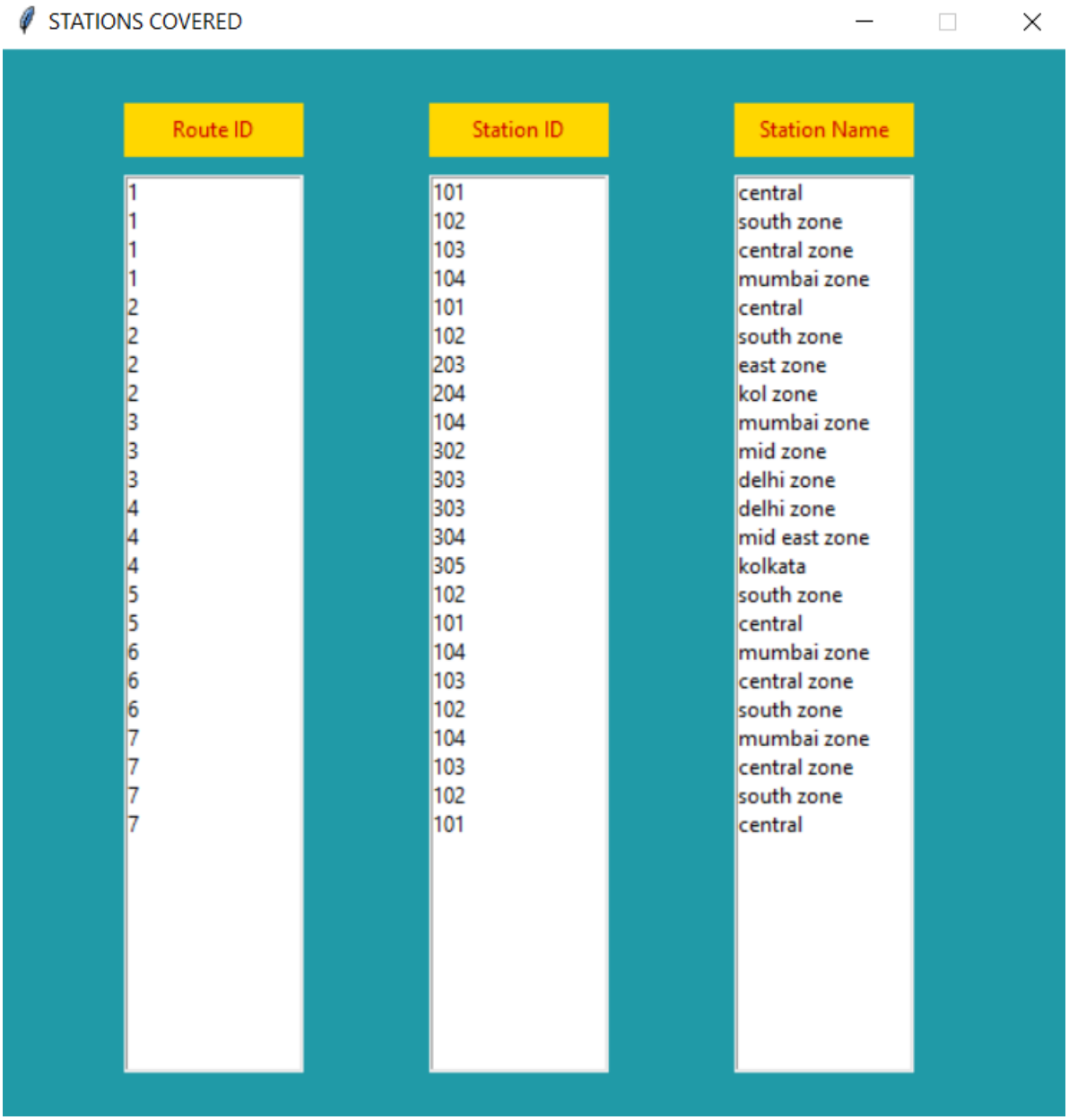

NEW PASSENGER LOGIN
Username
sai
Password
*****
Enter


NEW PASSENGER LOGIN
Username
Password
Enter
Registration Successful!

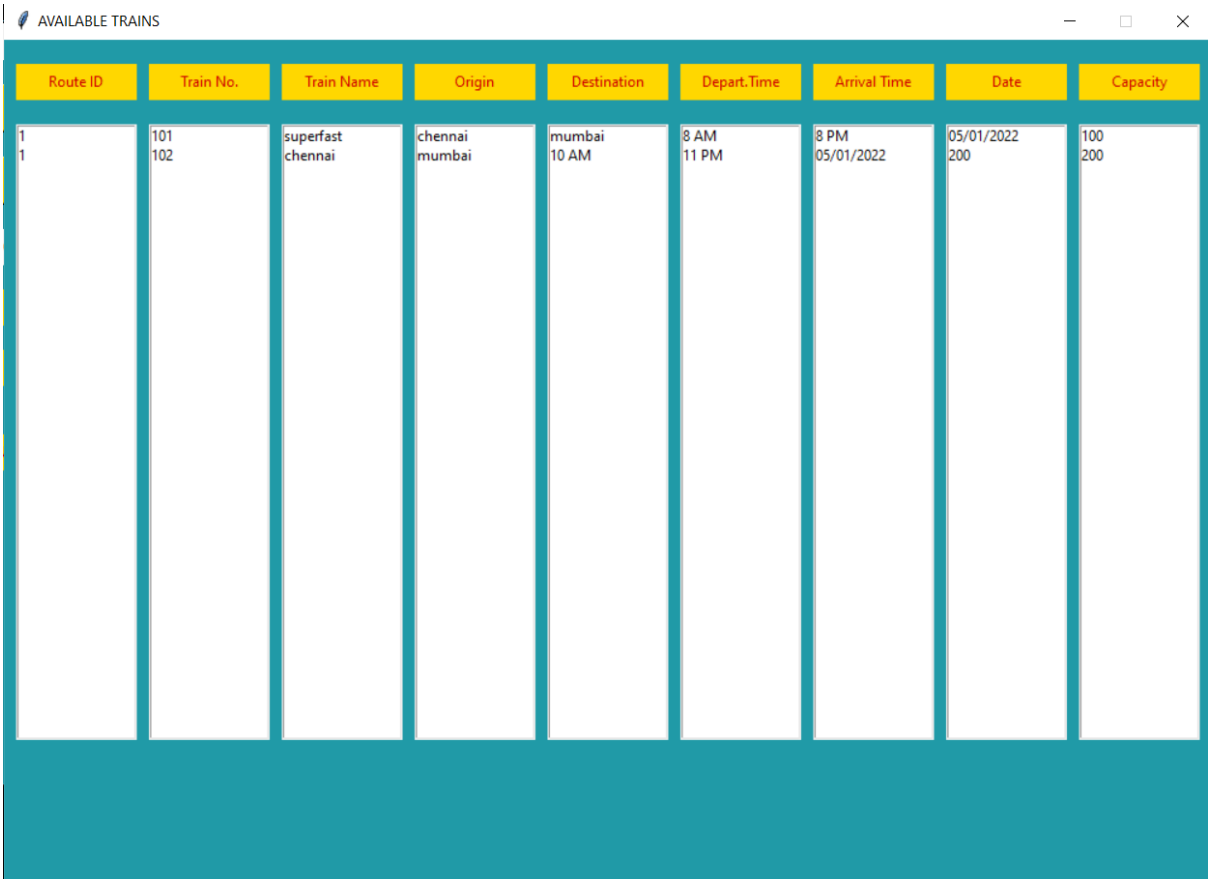

EXISTING PASSENGER LOGIN
Username
Anush
Password
*****
Enter
Login Successful!

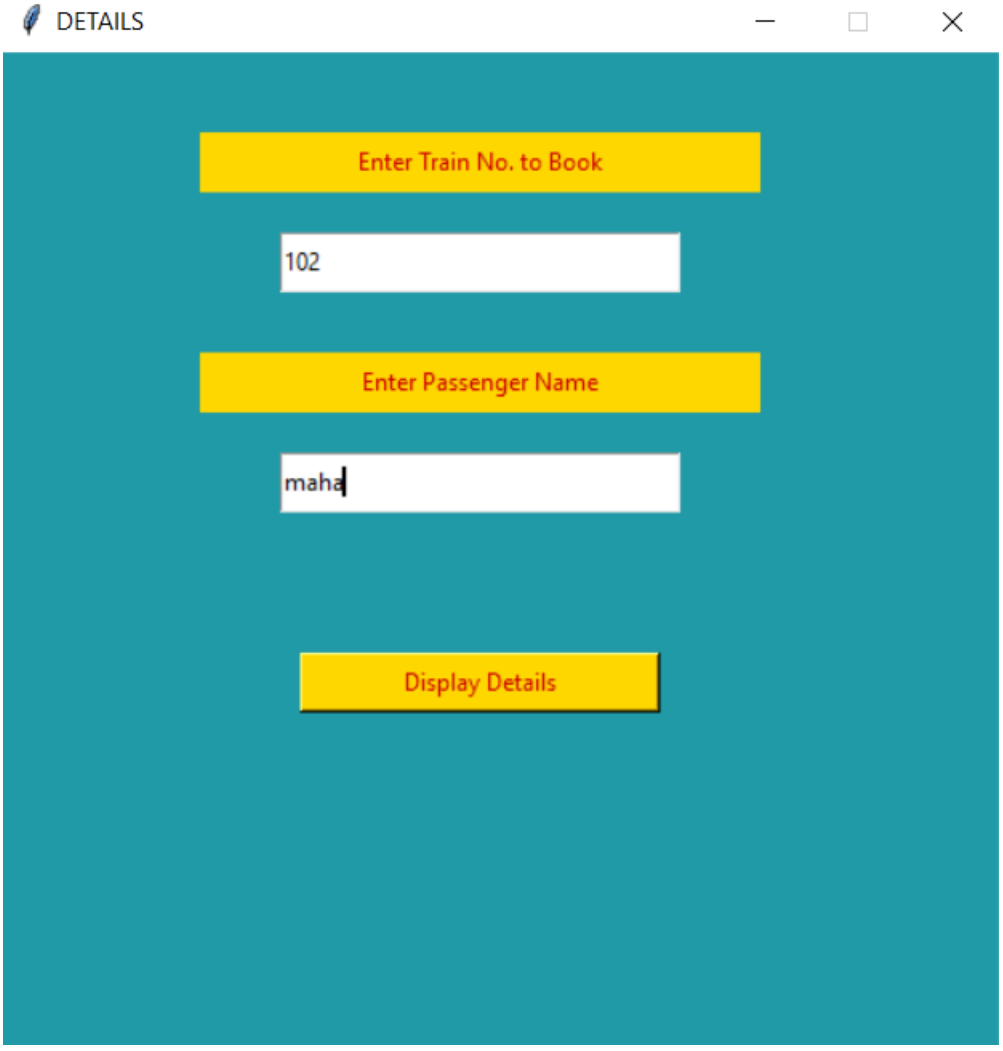

BOOKING
Avaialble Routes
Stations Covered
*** Check Available Routes and Stations Covered Before Booking ***
Enter Following Details to Check Trains
Route ID
No. of Passengers
Check Trains
Proceed To Next Step


AVAILABLE ROUTES
Route ID
Origin
Destination
Distance
Fare
1
2
3
4
5
6
7
chennai
chennai
mumbai
delhi
hyderabad
mumbai
mumbai
mumbai
kolkata
delhi
kolkata
chennai
hyderabad
chennai
2500
3000
1500
950
600
700
2600
1200
1600
700
400
650
800
1200


STATIONS COVERED
Route ID
Station ID
Station Name
1 1 1 1 2 2 2 2 3 3 3 4 4 4 5 5 6 6 6 7 7 7 7
101 102 103 104 101 102 203 204 104 302 303 303 304 305 102 101 104 103 102 104 103 102 101
central south zone central zone mumbai zone central south zone east zone kol zone mumbai zone mid zone delhi zone delhi zone mid east zone kolkata south zone central mumbai zone central zone south zone mumbai zone central zone south zone central


BOOKING
Avaialble Routes
Stations Covered
*** Check Available Routes and Stations Covered Before Booking ***
Enter Following Details to Check Trains
Route ID
1
No. of Passengers
60
Check Trains
Proceed To Next Step


AVAILABLE TRAINS
Route ID
Train No.
Train Name
Origin
Destination
Depart.Time
Arrival Time
Date
Capacity
1
1
101
102
superfast
chennai
chennai
mumbai
mumbai
10 AM
8 AM
11 PM
8 PM
05/01/2022
05/01/2022
200
100
200


DETAILS
Enter Train No. to Book
102
Enter Passenger Name
maha
Display Details


BOOKING DETAILS
Passenger Name
maha
Train No.
102
Route ID
1
No. of Passengers
60
Origin
mumbai
Destination
10 AM
Date
11 PM
Time
05/01/2022
Total Fare
150000

RESULT:

Thus, the required database has been designed and the appropriate data has been inserted, and GUI has been created

APPENDIX - REPORT ON PYTHON GUI TKINTER:

Tkinter is an inbuilt Python module used to create simple GUI apps. It is the most commonly used module for GUI apps in the Python.

The tkinter package ("Tk interface") is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, including macOS, as well as on Windows systems.

Running python -m tkinter from the command line should open a window demonstrating a simple Tk interface, letting you know that tkinter is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

Tkinter supports a range of Tcl/Tk versions, built either with or without thread support. The official Python binary release bundles Tcl/Tk 8.6 threaded. See the source code for
the _tkinter module for more information about supported versions.

Tkinter is not a thin wrapper, but adds a fair amount of its own logic to make the experience more pythonic. This documentation will concentrate on these additions and changes, and refer to the official Tcl/Tk documentation for details that are unchanged.

Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's cross-platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run.

Although Tkinter is considered the de-facto Python GUI framework, it's not without criticism. One notable criticism is that GUIs built with Tkinter look outdated. If you want a shiny, modern interface, then Tkinter may not be what you're looking for.

However, Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top priority is to build something that's functional and cross-platform quickly.

Anything that happens in a user interface is an event. We say that an event is fired whenever the user does something – for example, clicks on a button or types a keyboard shortcut. Some events could also be triggered by occurrences which are not controlled by the user – for example, a background task might

complete, or a network connection might be established or lost.
Our application needs to monitor, or listen for, all the events that we find interesting, and respond to them in some way if they occur.

To do this, we usually associate certain functions with particular events. We call a function which performs an action in response to an event an event handler – we bind handlers to events.

Architecture :

Tcl/Tk is not a single library but rather consists of a few distinct modules, each with separate functionality and its own official documentation. Python's binary releases also ship an add-on module together with it.

Tcl:

Tcl is a dynamic interpreted programming language, just like Python. Though it can be used on its own as a general-purpose programming language, it is most commonly embedded into C applications as a scripting engine or an interface to the Tk toolkit. The Tcl library has a C interface to create and manage one or more instances of a Tcl interpreter, run Tcl commands and scripts in those instances, and add custom commands implemented in either Tcl or C. Each interpreter has an event queue, and there are facilities to send events to it and process them. Unlike Python, Tcl's execution model is designed around cooperative multitasking, and Tkinter bridges this difference (see Threading model for details).

Tk:

Tk is a Tcl package implemented in C that adds custom commands to create and manipulate GUI widgets. Each Tk object embeds its own Tcl interpreter instance with Tk loaded into it. Tk's widgets are very customizable, though at the cost of a dated appearance. Tk uses Tcl's event queue to generate and process GUI events.

Ttk:

Themed Tk (Ttk) is a newer family of Tk widgets that provide a much better appearance on different platforms than many of the classic Tk widgets. Ttk is distributed as part of Tk, starting with Tk version 8.5. Python bindings are provided in a separate module, tkinter.ttk.

Internally, Tk and Ttk use facilities of the underlying operating system, i.e., Xlib on Unix/X11, Cocoa on macOS, GDI on Windows.

When your Python application uses a class in Tkinter, e.g., to create a widget, the tkinter module first assembles a Tcl/Tk command string. It passes that Tcl command string to an internal _tkinter binary module, which then calls the Tcl

interpreter to evaluate it. The Tcl interpreter will then call into the Tk and/or Ttk packages, which will in turn make calls to Xlib, Cocoa, or GDI.

Tkinter Modules Support for Tkinter is spread across several modules. Most applications will need the main tkinter module, as well as the tkinter.ttk module, which provides the modern themed widget set and API:

from tkinter import *from tkinter import ttk

class tkinter.Tk(screenName=None, baseName=None, className='Tk', useTk=1)

The Tk class is instantiated without arguments. This creates a toplevel widget of Tk which usually is the main window of an application. Each instance has its own associated Tcl interpreter.

tkinter.Tcl(screenName=None, baseName=None, className='Tk', useTk=0)

The Tcl() function is a factory function which creates an object much like that created by the Tk class, except that it does not initialize the Tk subsystem. This is most often useful when driving the Tcl interpreter in an environment where one doesn't want to create extraneous toplevel windows, or where one cannot (such as Unix/Linux systems without an X server). An object created by the Tcl() object can have a Toplevel window created (and the Tk subsystem initialized) by calling its loadtk() method.

The modules that provide Tk support include:

tkinter
Main Tkinter module.

tkinter.colorchooser
Dialog to let the user choose a color.

tkinter.commondialog
Base class for the dialogs defined in the other modules listed here.

tkinter.filedialog
Common dialogs to allow the user to specify a file to open or save.

tkinter.font
Utilities to help work with fonts.

tkinter.messagebox
Access to standard Tk dialog boxes.

tkinter.scrolledtext
Text widget with a vertical scroll bar built in.

tkinter.simpledialog
Basic dialogs and convenience functions.

tkinter.ttk
Themed widget set introduced in Tk 8.5, providing modern alternatives for many of the classic widgets in the main tkinter module.

Additional modules:

tkinter
A binary module that contains the low-level interface to Tcl/Tk. It is automatically imported by the main tkinter module, and should never be used directly by application programmers. It is usually a shared library (or DLL), but might in some cases be statically linked with the Python interpreter.

idlelib
Python's Integrated Development and Learning Environment (IDLE). Based on tkinter.

tkinter.constants
Symbolic constants that can be used in place of strings when passing various parameters to Tkinter calls. Automatically imported by the main tkinter module.

tkinter.dnd (experimental)
Drag-and-drop support for tkinter. This will become deprecated when it is replaced with the Tk DND.

tkinter.tix (deprecated)
An older third-party Tcl/Tk package that adds several new widgets. Better alternatives for most can be found in tkinter.ttk.

turtle
Turtle graphics in a Tk window.

SOURCE:

https://docs.python.org/3/library/tkinter.html

https://python-textbok.readthedocs.io/en/1.0/Introduction_to_GUI_Programming.html

https://www.tutorialspoint.com/python/python_gui_programming.html