BUS MANAGEMENT SYSTEM

INTRODUCTION:

The **Bus Management System** is a software application developed to automate and streamline the operations of a bus transportation service. It aims to replace traditional manual processes involved in managing bus schedules, bookings, and user details with a more efficient, accurate, and user-friendly digital platform.

This system is designed with two main modules: **Admin** and **User**. The **Admin module** allows administrators to manage buses, routes, schedules, and view booking records. It provides full control over system operations, ensuring smooth and organized management. The **User module** enables passengers to search for buses, book tickets, select seats, and receive booking confirmations. It simplifies the entire booking process, making it accessible and convenient for users.

The project is developed using **Python** for the backend and **MySQL** for database management, ensuring reliable performance and secure data storage. The interface is built to be simple, responsive, and easy to navigate, enhancing the overall user experience.

OBJECTIVE:

The primary objective of the Bus Management System is to develop an efficient and reliable platform that simplifies the management of bus operations. This includes automating tasks such as ticket booking, schedule management, bus allocation, and user handling. The system aims to enhance accuracy, reduce manual workload, and provide a smooth user experience for both passengers and staff.

PROPOSED SYSTEM:

The **Bus Booking System** is a software application designed to automate the process of booking bus tickets for passengers. It provides two main user roles:

- **Admin** (Manages schedules, views bookings, updates routes)
- **User** (Books tickets, searches, updates, cancels bookings)

The system is built using **Python (Tkinter for GUI)** and **MySQL** for database storage.

1. Existing System Features

Admin Module

1. **Authentication**

- Admin login with username and password (stored in MySQL)

2. **Schedule Management**
    - Add new bus schedules (Boarding, Destination, Fare, Dates, Code, Seats)
    - Update existing schedules (Edit departure, destination, fare, dates)
3. **View Bookings**
    - Display all tickets booked for a specific bus (using bus code)

User Module

1. **Ticket Booking**
    - Enter passenger details (Name, Age, Gender, Email, Contact, Luggage)
    - Select departure and destination
    - View available buses and fares
    - Payment simulation (Total fare calculation based on age and luggage)
    - Generate and download ticket (saved as .txt file)
2. **Search Ticket**
    - Search by ticket number to view booking details
3. **Update Ticket**
    - Modify passenger details (Name, Age, Gender, Email, Contact, Luggage)
4. **Cancel Ticket**
    - Delete booking using ticket number (updates seat availability)

2. System Architecture

Backend

- **Python** (Core logic)
- **MySQL** (Database)
- **Tkinter** (GUI)

Frontend

- **User Interface** (Tkinter windows)
- **Admin Panel** (Tkinter-based dashboard)

Data Flow

1. **User** → Books Ticket → Database Updated
2. **Admin** → Manages Schedules → Database Updated
3. **System** → Generates Tickets → Saves as .txt

SOURCE CODE:

```
# IMPORTING REQUIRED MODULES
import mysql.connector as ms
import tkinter as tk
from tkinter import messagebox
from tkinter import filedialog
import random
import pickle
import os

# CONNECTING TO MYSQL
mycon = ms.connect(host='localhost', user='root', passwd='7410')
mycursor = mycon.cursor()
mycursor.execute("use ms")

# MAIN HOME PAGE
def main():
    global root
    root = tk.Tk()
    root.geometry("400x411")
    root.title("Bus Booking System")
    root.minsize(400, 411)
```

```
    root.maxsize(400, 411)
    img = tk.PhotoImage(name='image', file="bg.png")
    img_label = tk.Label(root, image=img)
    img_label.place(x=0, y=0)
    label1 = tk.Label(root, text="Select:", font=("Arial Bold", 20), fg='#0A3075')
    label1.place(x=220, y=120)
    button1 = tk.Button(root, text="Admin", font=("Arial ", 18), fg='white', bg='#0A3075',
command=admin)
    button1.place(x=220, y=200)
    button2 = tk.Button(root, text="User", font=("Arial ", 18), fg='white', bg='#0A3075',
command=user_home_page)
    button2.place(x=230, y=260)


    root.mainloop()


# ADMIN AUTHENTICATION
def admin():
    root.destroy()
    global root1
    root1 = tk.Tk()
    root1.geometry("500x250")
    root1.title("Bus Booking System")
    au = tk.Label(root1, text='Authentication', font='Arial 20 bold', fg='dark orange')
    au.pack(padx=40, pady=30)
    uname = tk.Label(root1, text="Enter Admin Name:", font='Arial 18 bold', fg='black')
    uname.place(x=30, y=90)
    global adname_ent
    adname_ent = tk.Entry(root1, width=30)
```

```
    adname_ent.place(x=265, y=100)
    passwd = tk.Label(root1, text="Enter Admin Password:", font='Arial 18 bold', fg='black')
    passwd.place(x=20, y=130)
    global passwd_ent
    passwd_ent = tk.Entry(root1, width=30,show="*")
    passwd_ent.place(x=295, y=140)
    button3 = tk.Button(root1, text="Enter", width=20, height=2, bg='white',
command=check_passwd)
    button3.place(x=170, y=180)


# CHECKING PASSWORD
def check_passwd():
    adminname = adname_ent.get()
    pswd = passwd_ent.get()
    mycursor.execute("select * from admin")
    mydata = mycursor.fetchall()
    flag = False
    for x in mydata:
        if x[0] == adminname and x[1] == pswd:
            flag = True
            messagebox.showinfo(title="HI", message=f"Welcome {adminname}")
            admin_home_page()
    if not flag:
        messagebox.showerror(title="ACCESS DENIED!", message="Incorrect Username or
Password")


# ADMIN HOME PAGE
def admin_home_page():
```

```
    try:
        root1.destroy()
    except:
        pass
    global root2
    root2 = tk.Tk()
    root2.geometry("410x420")
    root2.title("Bus Booking System : Admin")
    root2.protocol("WM_DELETE_WINDOW", on_closing_root2_admin)
    label = tk.Label(root2, text="Select Function:", font=("Arial Bold", 20), fg='dark orange')
    label.pack(padx=30, pady=30)
    button4 = tk.Button(root2, text="Add Schedules", width=20, height=2, bg='white',
                        command=insert_schedule_interface)
    button4.place(x=125, y=100)
    button5 = tk.Button(root2, text="Update Schedules", width=20, height=2, bg='white',
                        command=update_schedule_info_interface)
    button5.place(x=125, y=180)
    button6 = tk.Button(root2, text="Display Per Bus", width=20, height=2, bg='white',
                        command=display_per_bus_interface)
    button6.place(x=125, y=260)
    button7 = tk.Button(root2, text="Log Out", width=20, height=2, bg='white',
                        command=admin_log_out)
    button7.place(x=125, y=340)


# ADMIN FUNCTION 1 : ADD SCHEDULE5
def insert_schedule_interface():
    root2.destroy()
    global root3
```

```
root3 = tk.Tk()
root3.geometry("600x600")
root3.title("Bus Booking System")

label = tk.Label(root3, text="Enter the following details:", font=("Arial Bold", 18))
label.pack(padx=30, pady=30)

boar = tk.Label(root3, text="Enter Departure:", font='Arial 18 bold', fg='black')
boar.place(x=40, y=130)
global boar_ent
boar_ent = tk.Entry(root3, width=30)
boar_ent.place(x=325, y=140)

des = tk.Label(root3, text="Enter Destination", font='Arial 18 bold', fg='black')
des.place(x=40, y=170)
global des_ent
des_ent = tk.Entry(root3, width=30)
des_ent.place(x=325, y=180)

fare = tk.Label(root3, text="Enter Fare:", font='Arial 18 bold', fg='black')
fare.place(x=40, y=210)
global fare_ent
fare_ent = tk.Entry(root3, width=30)
fare_ent.place(x=325, y=220)

dob = tk.Label(root3, text="Enter Date of Departure:", font='Arial 18 bold', fg='black')
dob.place(x=40, y=250)
global dob_ent
```

```
    dob_ent = tk.Entry(root3, width=30)
    dob_ent.place(x=325, y=260)

    doa = tk.Label(root3, text="Enter Date of Arrival:", font='Arial 18 bold', fg='black')
    doa.place(x=40, y=290)
    global doa_ent
    doa_ent = tk.Entry(root3, width=30)
    doa_ent.place(x=325, y=300)

    code = tk.Label(root3, text="Enter Code:", font='Arial 18 bold', fg='black')
    code.place(x=40, y=330)
    global code_ent
    code_ent = tk.Entry(root3, width=30)
    code_ent.place(x=325, y=340)

    ts = tk.Label(root3, text="Enter Total Seats:", font='Arial 18 bold', fg='black')
    ts.place(x=40, y=370)
    global ts_ent
    ts_ent = tk.Entry(root3, width=30)
    ts_ent.place(x=325, y=380)

    button3 = tk.Button(root3, text="Enter", width=20, height=2, bg='white',
command=add_schedule)
    button3.place(x=200, y=500)

# INSERTING INTO DATABASE/BINARY FILE
def add_schedule():
    boar = boar_ent.get()
```

```
    dest = des_ent.get()
    fare = fare_ent.get()
    doB = dob_ent.get()
    doA = doa_ent.get()
    code = code_ent.get()
    ts = ts_ent.get()
    vs = ts
    try:
        query = "insert into Schedule values('{}','{}',{},'{}','{}',{},{},{})".format(boar, dest,
fare, doB, doA, code,
                                                                    ts, ts)
        mycursor.execute(query)


        f = open("schedules.dat", "ab")
        info = [boar, dest, fare, doB, doA, code, ts, vs]
        pickle.dump(info, f)
        f.close()


        messagebox.showinfo(title="Bus Booking System", message="Schedule Added")
        root3.destroy()
        admin_home_page()
        mycon.commit()
    except:
        messagebox.showerror(title="Error",message="Invalid Input")


# ADMIN FUNCTION 2 : UPDATION OF SCHEDULE DATABASE/BINARY FILE
def update_schedule_info_interface():
```

```
    root2.destroy()
    global root3
    root3 = tk.Tk()
    root3.geometry("400x250")
    root3.title("Bus Booking System")

    label = tk.Label(root3, text="Updation:", font=("Arial Bold", 20), fg='dark orange')
    label.pack(padx=30, pady=30)

    code = tk.Label(root3, text="Enter Code:", font='Arial 18 bold', fg='black')
    code.place(x=25, y=100)
    global code_ent1
    code_ent1 = tk.Entry(root3, width=30)
    code_ent1.place(x=175, y=110)

    button = tk.Button(root3, text="Enter", width=20, height=2, bg='white',
command=updating_options_schedule)
    button.place(x=120, y=160)

# UPDATING OPTIONS
def updating_options_schedule():
    mycursor.execute("select * from schedule")
    mydata = mycursor.fetchall()
    code1 = code_ent1.get()
    c = 0
    try:
        for b, d, fa, dob, doa, co, ts, vs in mydata:
            if co == int(code1):
```

```
        global cod
        cod = co
        c += 1
        root3.destroy()
        global root4
        root4 = tk.Tk()
        root4.geometry("400x430")
        root4.title("Bus Booking System")
        label = tk.Label(root4, text="Select your choice:", font=("Arial Bold", 18), fg='dark
orange')
        label.pack(padx=30, pady=30)


        button1 = tk.Button(root4, text="Departure", width=20, height=2, bg='white',
command=new_dep)
        button1.place(x=130, y=100)


        button2 = tk.Button(root4, text="Destination", width=20, height=2, bg='white',
command=new_des)
        button2.place(x=130, y=150)


        button3 = tk.Button(root4, text="Fare", width=20, height=2, bg='white',
command=new_fare)
        button3.place(x=130, y=200)


        button4 = tk.Button(root4, text="Date of Boarding", width=20, height=2,
bg='white', command=new_dob)
        button4.place(x=130, y=250)


        button5 = tk.Button(root4, text="Date of Arrival", width=20, height=2, bg='white',
command=new_doa)
```

```
            button5.place(x=130, y=300)
        if c == 0:
            messagebox.showerror(title="Error", message="No matches found")
    except:
        messagebox.showerror(title="Error", message="Invalid Input")


# OPTION 1 : DEPARTURE
def new_dep():
    root4.destroy()
    global root5
    root5 = tk.Tk()
    root5.geometry("490x175")
    root5.title("Bus Booking System:Update Schedule")
    dep = tk.Label(root5, text="Enter New Departure:", font='Arial 18 bold', fg='black')
    dep.place(x=15, y=40)
    global dep_ent
    dep_ent = tk.Entry(root5, width=30)
    dep_ent.place(x=270, y=50)
    button = tk.Button(root5, text="Enter", width=20, height=2, bg='white',
command=up_dep)
    button.place(x=170, y=100)


# DEPARTURE UPDATION
def up_dep():
    ndep = dep_ent.get()
    mycursor.execute("update schedule\
            set Boarding='%s'\
            where code=%d " % (ndep, cod))
```

```
    f = open("schedules.dat", "rb+")
    while True:
        try:
            pos = f.tell()
            mydata = pickle.load(f)
            if mydata[5] == cod:
                mydata[0] = ndep
                f.seek(pos)
                pickle.dump(mydata, f)
                break
        except EOFError:
            f.close()
    messagebox.showinfo(title="", message="UPDATION SUCCESSFUL")
    root5.destroy()
    admin_home_page()
    mycon.commit()


# OPTION 2 : DESTINATION
def new_des():
    root4.destroy()
    global root5
    root5 = tk.Tk()
    root5.geometry("490x175")
    root5.title("Bus Booking System")
    des = tk.Label(root5, text="Enter New Destination:", font='Arial 18 bold', fg='black')
    des.place(x=10, y=40)
    global des_ent
    des_ent = tk.Entry(root5, width=30)
```

```
    des_ent.place(x=280, y=50)
    button = tk.Button(root5, text="Enter", width=20, height=2, bg='white', command=up_des)
    button.place(x=170, y=100)

# DESTINATION UPDATION
def up_des():
    ndes = des_ent.get()
    mycursor.execute("update schedule\
            set Destination='%s'\
            where code=%d " % (ndes, cod))
    f = open("schedules.dat", "rb+")
    while True:
        try:
            pos = f.tell()
            mydata = pickle.load(f)
            if mydata[5] == cod:
                mydata[1] = ndes
                f.seek(pos)
                pickle.dump(mydata, f)
                break
        except EOFError:
            f.close()
    messagebox.showinfo(title="", message="UPDATION SUCCESSFUL")
    root5.destroy()
    admin_home_page()
    mycon.commit()

# OPTION 3 : FARE
```

```
def new_fare():
    root4.destroy()
    global root5
    root5 = tk.Tk()
    root5.geometry("490x175")
    root5.title("Bus Booking System")
    fa = tk.Label(root5, text="Enter New Fare:", font='Arial 18 bold', fg='black')
    fa.place(x=45, y=40)
    global fa_ent
    fa_ent = tk.Entry(root5, width=30)
    fa_ent.place(x=235, y=50)
    button = tk.Button(root5, text="Enter", width=20, height=2, bg='white',
command=up_fare)
    button.place(x=170, y=100)


# FARE UPDATION
def up_fare():
    nfa = fa_ent.get()
    mycursor.execute("update schedule\
            set Fare=%s\
            where code=%d " % (nfa, cod))
    f = open("schedules.dat", "rb+")
    while True:
        try:
            pos = f.tell()
            mydata = pickle.load(f)
            if mydata[5] == cod:
                mydata[2] = nfa
```

```
                f.seek(pos)
                pickle.dump(mydata, f)
                break
            except EOFError:
                f.close()
    messagebox.showinfo(title="", message="UPDATION SUCCESSFUL")
    root5.destroy()
    admin_home_page()
    mycon.commit()


# OPTION 4 : DATE OF BOARDING
def new_dob():
    root4.destroy()
    global root5
    root5 = tk.Tk()
    root5.geometry("490x195")
    root5.title("Bus Booking System")
    dob = tk.Label(root5, text="Enter New Date of Boarding:", font='Arial 18 bold', fg='black')
    dob.place(x=85, y=40)
    global dob_ent
    dob_ent = tk.Entry(root5, width=30)
    dob_ent.place(x=155, y=90)
    button = tk.Button(root5, text="Enter", width=20, height=2, bg='white',
command=up_dob)
    button.place(x=170, y=135)


# DATE OF BOARDING UPDATION
def up_dob():
```

```
    ndob = dob_ent.get()
    mycursor.execute("update schedule\
            set DOB='%s'\
            where code=%d " % (ndob, cod))
    f = open("schedules.dat", "rb+")
    while True:
        try:
            pos = f.tell()
            mydata = pickle.load(f)
            if mydata[5] == cod:
                mydata[3] = ndob
                f.seek(pos)
                pickle.dump(mydata, f)
                break
        except EOFError:
            f.close()
    messagebox.showinfo(title="", message="UPDATION SUCCESSFUL")
    root5.destroy()
    admin_home_page()
    mycon.commit()


# OPTION 5 : DATE OF ARRIVAL
def new_doa():
    root4.destroy()
    global root5
    root5 = tk.Tk()
    root5.geometry("490x195")
    root5.title("Bus Booking System")
```

```
    doa = tk.Label(root5, text="Enter New Date of Arrival:", font='Arial 18 bold', fg='black')
    doa.place(x=95, y=40)
    global doa_ent
    doa_ent = tk.Entry(root5, width=30)
    doa_ent.place(x=155, y=90)
    button = tk.Button(root5, text="Enter", width=20, height=2, bg='white',
command=up_doa)
    button.place(x=170, y=135)


# DATE OF ARRIVAL UPDATION
def up_doa():
    ndoa = doa_ent.get()
    mycursor.execute("update schedule\
            set DOA='%s'\
            where code=%d " % (ndoa, cod))
    f = open("schedules.dat", "rb+")
    while True:
        try:
            pos = f.tell()
            mydata = pickle.load(f)
            if mydata[5] == cod:
                mydata[4] = ndoa
                f.seek(pos)
                pickle.dump(mydata, f)
                break
        except EOFError:
            f.close()
    messagebox.showinfo(title="", message="UPDATION SUCCESSFUL")
```

```
    root5.destroy()
    admin_home_page()
    mycon.commit()


# ADMIN FUNCTION 3 : DISPLAYING OF TICKETS BOOKED FOR A PARTICULAR BUS
# TAKES THE BUS CODE AND DISPLAYS ALL THE TICKETS BOOKED FOR THAT BUS
def display_per_bus_interface():
    root2.destroy()
    global root3
    root3 = tk.Tk()
    root3.geometry("420x180")
    root3.title("Bus Booking System")
    code = tk.Label(root3, text="Enter Code:", font='Arial 18 bold', fg='black')
    code.place(x=40, y=50)
    global code_ent2
    code_ent2 = tk.Entry(root3, width=30)
    code_ent2.place(x=190, y=60)
    button = tk.Button(root3, text="Enter", width=20, height=2, bg='white',
command=display_per_bus)
    button.place(x=130, y=100)


# DISPLAYS TICKETS
def display_per_bus():
    try:
        mycursor.execute("select * from tickets where code=%s" % int(code_ent2.get()))
        mydata = mycursor.fetchall()
        if mydata == []:
```

```
        messagebox.showinfo(title="Info", message="No tickets booked")
    else:
        root3.destroy()
        global root4
        root4 = tk.Tk()
        root4.geometry("1000x400")
        root4.title("Bus Booking System")
        code = tk.Label(root4, text="Tickets Booked:", font='Arial 18 bold', fg='orange')
        code.pack(padx=30, pady=30)

        # calls main() on closing window
        root4.protocol("WM_DELETE_WINDOW", on_closing_root4_admin)

        labelframe = tk.Frame(root4)
        labelframe.columnconfigure(0, weight=1)
        labelframe.columnconfigure(1, weight=1)
        labelframe.columnconfigure(2, weight=1)
        labelframe.columnconfigure(3, weight=1)
        labelframe.columnconfigure(4, weight=1)
        labelframe.columnconfigure(5, weight=1)
        labelframe.columnconfigure(6, weight=1)
        labelframe.columnconfigure(7, weight=1)
        labelframe.columnconfigure(8, weight=1)
        labelframe.columnconfigure(9, weight=1)

        x = 0
        name = tk.Label(labelframe, text="Name", font='Arial 12 bold', fg='#20D2B5')
        name.grid(row=x, column=0, sticky=tk.W + tk.E)
```

```
        age = tk.Label(labelframe, text="Age", font='Arial 12 bold', fg='#20D2B5')
        age.grid(row=x, column=1, sticky=tk.W + tk.E)
        gen = tk.Label(labelframe, text="Gender", font='Arial 12 bold', fg='#20D2B5')
        gen.grid(row=x, column=2, sticky=tk.W + tk.E)
        email = tk.Label(labelframe, text="Email", font='Arial 12 bold', fg='#20D2B5')
        email.grid(row=x, column=3, sticky=tk.W + tk.E)
        con = tk.Label(labelframe, text="Contact", font='Arial 12 bold', fg='#20D2B5')
        con.grid(row=x, column=4, sticky=tk.W + tk.E)
        lugg = tk.Label(labelframe, text="Luggage", font='Arial 12 bold', fg='#20D2B5')
        lugg.grid(row=x, column=5, sticky=tk.W + tk.E)
        tkn1 = tk.Label(labelframe, text="Ticket Number", font='Arial 12 bold',
fg='#20D2B5')
        tkn1.grid(row=x, column=6, sticky=tk.W + tk.E)
        code1 = tk.Label(labelframe, text="Code", font='Arial 12 bold', fg='#20D2B5')
        code1.grid(row=x, column=7, sticky=tk.W + tk.E)
        sno = tk.Label(labelframe, text="Seat Number", font='Arial 12 bold', fg='#20D2B5')
        sno.grid(row=x, column=8, sticky=tk.W + tk.E)

        r = 1
        for n, a, g, e, c, l, tkn, co, sn in mydata:
            name = tk.Label(labelframe, text=n, font='Arial 12 ', fg='black')
            name.grid(row=r, column=0, sticky=tk.W + tk.E)
            age = tk.Label(labelframe, text=a, font='Arial 12 ', fg='black')
            age.grid(row=r, column=1, sticky=tk.W + tk.E)
            gen = tk.Label(labelframe, text=g, font='Arial 12 ', fg='black')
            gen.grid(row=r, column=2, sticky=tk.W + tk.E)
            email = tk.Label(labelframe, text=e, font='Arial 12 ', fg='black')
            email.grid(row=r, column=3, sticky=tk.W + tk.E)
```

```
            con = tk.Label(labelframe, text=c, font='Arial 12 ', fg='black')
            con.grid(row=r, column=4, sticky=tk.W + tk.E)
            lugg = tk.Label(labelframe, text=l, font='Arial 12 ', fg='black')
            lugg.grid(row=r, column=5, sticky=tk.W + tk.E)
            tkn = tk.Label(labelframe, text=tkn, font='Arial 12 ', fg='black')
            tkn.grid(row=r, column=6, sticky=tk.W + tk.E)
            code1 = tk.Label(labelframe, text=co, font='Arial 12 ', fg='black')
            code1.grid(row=r, column=7, sticky=tk.W + tk.E)
            sno = tk.Label(labelframe, text=sn, font='Arial 12 ', fg='black')
            sno.grid(row=r, column=8, sticky=tk.W + tk.E)
            r += 1
        labelframe.pack(fill='x')
    except:
        messagebox.showerror(title="Error", message="Invalid Input")


# ADMIN FUNCTION 4 : LOGGING OUT
def admin_log_out():
    if messagebox.askyesno(title="Log Out", message="Want to log out?"):
        root2.destroy()
        main()


# USER HOME PAGE
def user_home_page():
    try:
        root.destroy()
    except:
        pass
```

```
    global root2
    root2 = tk.Tk()
    root2.geometry("500x600")
    root2.title("Bus Booking System : User")
    label = tk.Label(root2, text="Select Function:", font=("Arial Bold", 18), fg='dark orange')
    label.pack(padx=30, pady=30)
    button4 = tk.Button(root2, text="Book Tickets", width=20, height=2, bg='white',
                        command=book_tickets_interface)
    button4.place(x=170, y=100)
    button5 = tk.Button(root2, text="Search Ticket", width=20, height=2, bg='white',
                        command=search_ticket_information)
    button5.place(x=170, y=200)
    button6 = tk.Button(root2, text="Update Ticket", width=20, height=2, bg='white',
                        command=update_ticket_information)
    button6.place(x=170, y=300)
    button7 = tk.Button(root2, text="Cancel Booking", width=20, height=2, bg='white',
                        command=ticket_cancellation_interface)
    button7.place(x=170, y=400)
    button8 = tk.Button(root2, text="Exit", width=20, height=2, bg='white',
                        command=user_exit)
    button8.place(x=170, y=500)


# USER FUNCTION 1 : BOOK TICKETS
def book_tickets_interface():
    global root3
    root3 = tk.Tk()
    root3.geometry("600x600")
    root3.title("Bus Booking System:Book Tickets")
```

```
    root2.destroy()
    label = tk.Label(root3, text="Enter the following details:", font=("Arial Bold", 18),
fg='Orange')
    label.pack(padx=30, pady=30)

    name1 = tk.Label(root3, text="Enter Name:", font='Arial 18 bold', fg='black')
    name1.place(x=40, y=130)
    global name_ent1
    name_ent1 = tk.Entry(root3, width=30)
    name_ent1.place(x=325, y=140)

    age1 = tk.Label(root3, text="Enter Age", font='Arial 18 bold', fg='black')
    age1.place(x=40, y=170)
    global age_ent1
    age_ent1 = tk.Entry(root3, width=30)
    age_ent1.place(x=325, y=180)

    gender1 = tk.Label(root3, text="Enter Gender:", font='Arial 18 bold', fg='black')
    gender1.place(x=40, y=210)
    global gen_ent1
    gen_ent1 = tk.Entry(root3, width=30)
    gen_ent1.place(x=325, y=220)

    email1 = tk.Label(root3, text="Enter Email:", font='Arial 18 bold', fg='black')
    email1.place(x=40, y=250)
    global em_ent1
    em_ent1 = tk.Entry(root3, width=30)
    em_ent1.place(x=325, y=260)
```

```
    mob1 = tk.Label(root3, text="Enter Mobile:", font='Arial 18 bold', fg='black')
    mob1.place(x=40, y=290)
    global mob_ent1
    mob_ent1 = tk.Entry(root3, width=30)
    mob_ent1.place(x=325, y=300)

    lugg1 = tk.Label(root3, text="Enter Luggage:", font='Arial 18 bold', fg='black')
    lugg1.place(x=40, y=330)
    global lugg_ent1
    lugg_ent1 = tk.Entry(root3, width=30)
    lugg_ent1.place(x=325, y=340)

    fro = tk.Label(root3, text="Enter From:", font='Arial 18 bold', fg='black')
    fro.place(x=40, y=370)
    global fro_ent
    fro_ent = tk.Entry(root3, width=30)
    fro_ent.place(x=325, y=380)

    to = tk.Label(root3, text="Enter To:", font='Arial 18 bold', fg='black')
    to.place(x=40, y=410)
    global to_ent
    to_ent = tk.Entry(root3, width=30)
    to_ent.place(x=325, y=420)

    button3 = tk.Button(root3, text="Next", width=20, height=2, bg='white',
command=display_schedule_information)
    button3.place(x=200, y=500)
```

```
# DISPLAYING THE AVAILABLE BUSES
def display_schedule_information():
    global name, age, gender, email, mobile, luggage, age1, lug1
    name = name_ent1.get()
    age = str(age_ent1.get())
    age1 = int(age)
    gender = gen_ent1.get()
    email = em_ent1.get()
    mobile = mob_ent1.get()
    luggage = str(lugg_ent1.get())
    lug1 = int(luggage)
    fro_ent1 = (fro_ent.get()).strip()
    to_ent1 = (to_ent.get()).strip()
    if name == "" or age == "" or gender == "" or email == "" or mobile == "" or luggage ==
"":
        messagebox.showerror(title="Error", message="All information to be entered")
    else:
        mycursor.execute("select * from schedule\
                    where boarding='%s' and destination='%s' " % (fro_ent1, to_ent1))
        mydata = mycursor.fetchall()

        if mydata == []:
            messagebox.showinfo(title="Error", message="No buses found")
        else:
            root3.destroy()
            global root6
            root6 = tk.Tk()
```

```
        root6.geometry("1000x350")
        root6.title("Bus Booking System")
        title = tk.Label(root6, text="AVAILABLE BUSES:", font='Arial 18 bold',
fg='Orange')
        title.pack(padx=30, pady=30)

        global root7
        root7 = tk.Tk()
        root7.geometry("400x220")
        root7.title("Bus Booking System")

        co = tk.Label(root7, text="Enter code:", font='Arial 18 bold', fg='black')
        co.place(x=40, y=40)
        global code_ent3
        code_ent3 = tk.Entry(root7, width=30)
        code_ent3.place(x=190, y=45)

        labelframe = tk.Frame(root6)
        labelframe.columnconfigure(0, weight=1)
        labelframe.columnconfigure(1, weight=1)
        labelframe.columnconfigure(2, weight=1)
        labelframe.columnconfigure(3, weight=1)
        labelframe.columnconfigure(4, weight=1)
        labelframe.columnconfigure(5, weight=1)
        labelframe.columnconfigure(6, weight=1)
        labelframe.columnconfigure(7, weight=1)

        x = 0
```

```
        boar1 = tk.Label(labelframe, text="DEPARTURE", font='Arial 12 bold',
fg='#20D2B5')

        boar1.grid(row=x, column=0, sticky=tk.W + tk.E)

        des1 = tk.Label(labelframe, text="DESTINATION", font='Arial 12 bold',
fg='#20D2B5')

        des1.grid(row=x, column=1, sticky=tk.W + tk.E)

        fa1 = tk.Label(labelframe, text="FARE", font='Arial 12 bold', fg='#20D2B5')

        fa1.grid(row=x, column=2, sticky=tk.W + tk.E)

        dob1 = tk.Label(labelframe, text="DATE OF BOARDING", font='Arial 12 bold',
fg='#20D2B5')

        dob1.grid(row=x, column=3, sticky=tk.W + tk.E)

        doa1 = tk.Label(labelframe, text="DATE OF ARRIVAL", font='Arial 12 bold',
fg='#20D2B5')

        doa1.grid(row=x, column=4, sticky=tk.W + tk.E)

        co1 = tk.Label(labelframe, text="CODE", font='Arial 12 bold', fg='#20D2B5')

        co1.grid(row=x, column=5, sticky=tk.W + tk.E)

        ts1 = tk.Label(labelframe, text="TOTAL SEATS", font='Arial 12 bold',
fg='#20D2B5')

        ts1.grid(row=x, column=6, sticky=tk.W + tk.E)

        vs1 = tk.Label(labelframe, text="VACANT SEATS", font='Arial 12 bold',
fg='#20D2B5')

        vs1.grid(row=x, column=7, sticky=tk.W + tk.E)

        r = 1

        for b, d, f, dob, doa, co, ts, vs in mydata:

            boar = tk.Label(labelframe, text=b, font='Arial 12 ', fg='black')

            boar.grid(row=r, column=0, sticky=tk.W + tk.E)

            des = tk.Label(labelframe, text=d, font='Arial 12 ', fg='black')

            des.grid(row=r, column=1, sticky=tk.W + tk.E)

            fa = tk.Label(labelframe, text=f, font='Arial 12 ', fg='black')

            fa.grid(row=r, column=2, sticky=tk.W + tk.E)
```

```
            dob = tk.Label(labelframe, text=dob, font='Arial 12 ', fg='black')
            dob.grid(row=r, column=3, sticky=tk.W + tk.E)
            doa = tk.Label(labelframe, text=doa, font='Arial 12 ', fg='black')
            doa.grid(row=r, column=4, sticky=tk.W + tk.E)
            co = tk.Label(labelframe, text=co, font='Arial 12 ', fg='black')
            co.grid(row=r, column=5, sticky=tk.W + tk.E)
            ts = tk.Label(labelframe, text=ts, font='Arial 12 ', fg='black')
            ts.grid(row=r, column=6, sticky=tk.W + tk.E)
            vs = tk.Label(labelframe, text=vs, font='Arial 12 ', fg='black')
            vs.grid(row=r, column=7, sticky=tk.W + tk.E)
            r += 1
        labelframe.pack(fill='x')

        button3 = tk.Button(root7, text="Next", width=20, height=2, bg='white',
command=payment_procedure)
        button3.place(x=120, y=110)

# PAYMENT
def payment_procedure():
    mycursor.execute("select * from schedule \
        where code=%s" % int(code_ent3.get()))
    mydata = mycursor.fetchall()
    for dep, des, fa, dob, doa, code, ts, vs in mydata:
        if vs == 0:
            messagebox.showinfo(title="Bus Info", message="All seats are booked")
        else:
            global root8, tfa, code3
            root8 = tk.Tk()
```

```
root8.geometry("600x300")
root8.title("Payment")
code3 = int(code_ent3.get())
root6.destroy()  # display_schedule_info
root7.destroy()
mycursor.execute("select * from schedule \
   where code=%s" % code3)
mydata = mycursor.fetchall()
for dep, des, fa, dob, doa, code, ts, vs in mydata:
   if age1 >= 18:
      tfa = fa + (lug1 * 200)
   else:
      tfa = (fa * 0.95) + (lug1 * 150)
fa = tk.Label(root8, text="Total Fare:", font='Arial 18 bold', fg='black')
fa.place(x=40, y=40)
fa_ = tk.Label(root8, text=str(tfa), font='Arial 18 bold', fg='black')
fa_.place(x=300, y=40)

name2 = tk.Label(root8, text="Enter Name:", font='Arial 18 bold', fg='black')
name2.place(x=40, y=80)
name_ent2 = tk.Entry(root8, width=30)
name_ent2.place(x=300, y=90)

crno = tk.Label(root8, text="Enter Credit Card No.:", font='Arial 18 bold', fg='black')
crno.place(x=40, y=120)
crno_ent = tk.Entry(root8, width=30)
crno_ent.place(x=300, y=125)
```

```
        sc = tk.Label(root8, text="Enter Security Code:", font='Arial 18 bold', fg='black')
        sc.place(x=40, y=160)
        sc_ent = tk.Entry(root8, width=30)
        sc_ent.place(x=300, y=165)

        button3 = tk.Button(root8, text="Enter", width=20, height=2, bg='white',
command=book_tickets)
        button3.place(x=200, y=200)

# ADDING THE RECORD TO THE DATABASE/BINARY FILE
def book_tickets():
    root8.destroy()
    s = ["A", "B", "C", "D", "E", "F", "G", "H"]
    c = random.randint(1, 15)
    i = random.randint(0, 7)
    seat = s[i] + str(c)
    global ticketno
    ticketno = random.randint(1000, 9999)
    query = "insert into tickets values('{}',{},'{}','{}','{}',{},{},{},'{}')".format(name, age,
gender, email,
                                                                    mobile, luggage, ticketno,
                                                                    code3, seat)
    mycursor.execute(query)
    mycursor.execute("select * from schedule where code=%s" % code3)
    mydata = mycursor.fetchall()
    for b, d, f, dob, doa, co, ts, vs in mydata:
        while vs > 0:
            vs -= 1
```

```
            mycursor.execute("update schedule set vacantseats=%s where code=%d" % (vs,
code3))
            break
    mycon.commit()


    f = open("ticket1.dat", "ab")
    info = [name, age, gender, email, mobile, luggage, ticketno, code3, seat]
    pickle.dump(info, f)
    f.close()


    # vacant seats decrement by 1 after booking
    f = open("schedules.dat", "rb+")
    while True:
        try:
            pos = f.tell()
            mydata = pickle.load(f)
            if mydata[5] == code3:
                mydata[7] = int(mydata[7] - 1)
                f.seek(pos)
                pickle.dump(mydata, f)
                break
        except EOFError:
            f.close()


    ticket_generator()
    messagebox.showinfo(title="Bus Booking System", message="Booking Successful")


# FINAL DISPLAY OF TICKET
```

```
def ticket_generator():
    mycursor.execute("select Name,Age,Gender,Email,Contact,\
        Luggage,TicketNumber,SeatNo,Boarding,Destination,Fare,DOB,DOA from
tickets,schedule \
        where TicketNumber=%s and tickets.code=schedule.code" % ticketno)
    mydata = mycursor.fetchall()
    if mydata == []:
        messagebox.showerror(title="Info", message="Ticket Not Found")
    else:
        global root4
        root4 = tk.Tk()
        root4.geometry("375x520")
        root4.title("Bus Booking System:Search Ticket")
        heading = tk.Label(root4, text="Ticket Booked:", font='Arial 18 bold', fg='orange')
        heading.place(x=100, y=30)

        # calls user() when window is closed
        root4.protocol("WM_DELETE_WINDOW", on_closing_root4_user)

        labelframe = tk.Frame(root4)

        y = 1
        name = tk.Label(labelframe, text="Name", font='Arial 12 bold', fg='#20D2B5')  # dark
cyan
        name.grid(row=0, column=y, sticky=tk.W)
        age = tk.Label(labelframe, text="Age", font='Arial 12 bold', fg='#20D2B5')
        age.grid(row=1, column=y, sticky=tk.W)
        gen = tk.Label(labelframe, text="Gender", font='Arial 12 bold', fg='#20D2B5')
```

```
        gen.grid(row=2, column=y, sticky=tk.W)
        email = tk.Label(labelframe, text="Email", font='Arial 12 bold', fg='#20D2B5')
        email.grid(row=3, column=y, sticky=tk.W)
        con = tk.Label(labelframe, text="Contact", font='Arial 12 bold', fg='#20D2B5')
        con.grid(row=4, column=y, sticky=tk.W)
        lugg = tk.Label(labelframe, text="Luggage", font='Arial 12 bold', fg='#20D2B5')
        lugg.grid(row=5, column=y, sticky=tk.W)
        tkn1 = tk.Label(labelframe, text="Ticket Number", font='Arial 12 bold', fg='#20D2B5')
        tkn1.grid(row=6, column=y, sticky=tk.W)
        sno = tk.Label(labelframe, text="Seat Number", font='Arial 12 bold', fg='#20D2B5')
        sno.grid(row=7, column=y, sticky=tk.W)
        depp = tk.Label(labelframe, text="From", font='Arial 12 bold', fg='#20D2B5')
        depp.grid(row=8, column=y, sticky=tk.W)
        dess = tk.Label(labelframe, text="To", font='Arial 12 bold', fg='#20D2B5')
        dess.grid(row=9, column=y, sticky=tk.W)
        faa = tk.Label(labelframe, text="Total Fare", font='Arial 12 bold', fg='#20D2B5')
        faa.grid(row=10, column=y, sticky=tk.W)
        dobb = tk.Label(labelframe, text="Date of Boarding", font='Arial 12 bold',
fg='#20D2B5')
        dobb.grid(row=11, column=y, sticky=tk.W)
        doaa = tk.Label(labelframe, text="Date of Arrival", font='Arial 12 bold', fg='#20D2B5')
        doaa.grid(row=12, column=y, sticky=tk.W)


        r = 2
        for n, a, g, e, c, l, tkn, sn, dep, des, fa, dob, doa in mydata:
            global ticketno1
            ticketno1 = tkn
            if a >= 18:
```

```
    tfa = fa + (l * 200)
else:
    tfa = (fa * 0.95) + (l * 150)
name = tk.Label(labelframe, text=n, font='Arial 12 bold', fg='black')
name.grid(row=0, column=r, sticky=tk.W)
age = tk.Label(labelframe, text=a, font='Arial 12 bold', fg='black')
age.grid(row=1, column=r, sticky=tk.W)
gen = tk.Label(labelframe, text=g, font='Arial 12 bold', fg='black')
gen.grid(row=2, column=r, sticky=tk.W)
email = tk.Label(labelframe, text=e, font='Arial 12 bold', fg='black')
email.grid(row=3, column=r, sticky=tk.W)
con = tk.Label(labelframe, text=c, font='Arial 12 bold', fg='black')
con.grid(row=4, column=r, sticky=tk.W)
lugg = tk.Label(labelframe, text=l, font='Arial 12 bold', fg='black')
lugg.grid(row=5, column=r, sticky=tk.W)
tkn1 = tk.Label(labelframe, text=tkn, font='Arial 12 bold', fg='black')
tkn1.grid(row=6, column=r, sticky=tk.W)
sno = tk.Label(labelframe, text=sn, font='Arial 12 bold', fg='black')
sno.grid(row=7, column=r, sticky=tk.W)
depp = tk.Label(labelframe, text=dep, font='Arial 12 bold', fg='black')
depp.grid(row=8, column=r, sticky=tk.W)
dess = tk.Label(labelframe, text=des, font='Arial 12 bold', fg='black')
dess.grid(row=9, column=r, sticky=tk.W)
faa = tk.Label(labelframe, text=tfa, font='Arial 12 bold', fg='black')
faa.grid(row=10, column=r, sticky=tk.W)
dobb = tk.Label(labelframe, text=dob, font='Arial 12 bold', fg='black')
dobb.grid(row=11, column=r, sticky=tk.W)
doaa = tk.Label(labelframe, text=doa, font='Arial 12 bold', fg='black')
```

```
        doaa.grid(row=12, column=r, sticky=tk.W)
    labelframe.place(x=40, y=80)


    button = tk.Button(root4, text="Download", width=20, height=2, bg='white',
command=download_ticket)
    button.place(x=100, y=430)


# DOWNLOAD TICKET IN TEXT FILE FORMAT IN A USER SELECTED PATH
def download_ticket():
  mycursor.execute("select * from tickets,schedule \
    where TicketNumber=%s and tickets.code=schedule.code" % ticketno)
  mydata = mycursor.fetchall()
  for n, a, g, e, c, l, tkn, co, sn, dep, des, fa, dob, doa, code, ts, vs in mydata:
    if a >= 18:
      tfa = fa + (l * 200)
    else:
      tfa = (fa * 0.95) + (l * 150)
    fname = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text
Documents", ".txt")])
    f = open(fname, "w")
    f.write("NAME:")
    f.write(n + "\n")
    f.write("AGE:")
    f.write(str(a) + "\n")
    f.write("GENDER:")
    f.write(g + "\n")
    f.write("EMAIL:")
    f.write(e + "\n")
```

```
        f.write("CONTACT:")
        f.write(c + "\n")
        f.write("LUGGAGE:")
        f.write(str(l) + "\n")
        f.write("TICKET NUMBER:")
        f.write(str(tkn) + "\n")
        f.write("CODE:")
        f.write(str(co) + "\n")
        f.write("SEAT NUMBER:")
        f.write(sn + "\n")
        f.write("FROM:")
        f.write(dep + "\n")
        f.write("TO:")
        f.write(des + "\n")
        f.write("TOTAL FARE:")
        f.write(str(tfa) + "\n")
        f.write("DATE OF BOARDING:")
        f.write(str(dob) + "\n")
        f.write("DATE OF ARRIVAL:")
        f.write(str(doa) + "\n")
        f.close()
        messagebox.showinfo(title="Download Ticket", message="File Downloaded")

# USER FUNCTION 2 : SEARCH TICKET
def search_ticket_information():
    root2.destroy()
    global root3
    root3 = tk.Tk()
```

```
    root3.geometry("500x180")
    root3.title("Bus Booking System:Search Ticket")
    tkn2 = tk.Label(root3, text="Enter Ticket Number:", font='Arial 18 bold', fg='black')
    tkn2.place(x=25, y=50)
    global tkn_ent2
    tkn_ent2 = tk.Entry(root3, width=30)
    tkn_ent2.place(x=285, y=60)
    button = tk.Button(root3, text="Enter", width=20, height=2, bg='white',
command=display_ticket_information)
    button.place(x=170, y=100)


# DISPLAYING THE SEARCHED TICKET
def display_ticket_information():
    try:
        mycursor.execute("select Name,Age,Gender,Email,Contact,\
        Luggage,TicketNumber,SeatNo,Boarding,Destination,Fare,DOB,DOA from
tickets,schedule \
        where TicketNumber=%s and tickets.code=schedule.code" % int(tkn_ent2.get()))
        mydata = mycursor.fetchall()
        if mydata == []:
            messagebox.showerror(title="Info", message="Ticket Not Found")
        else:
            root3.destroy()
            global root4
            root4 = tk.Tk()
            root4.geometry("375x520")
            root4.title("Bus Booking System:Search Ticket")
            heading = tk.Label(root4, text="Ticket Booked:", font='Arial 18 bold', fg='orange')
```

```
        heading.place(x=100, y=30)


        # calls user() when window is closed
        root4.protocol("WM_DELETE_WINDOW", on_closing_root4_user)


        labelframe = tk.Frame(root4)


        y = 1
        name = tk.Label(labelframe, text="Name", font='Arial 12 bold', fg='#20D2B5')  #
dark cyan
        name.grid(row=0, column=y, sticky=tk.W)
        age = tk.Label(labelframe, text="Age", font='Arial 12 bold', fg='#20D2B5')
        age.grid(row=1, column=y, sticky=tk.W)
        gen = tk.Label(labelframe, text="Gender", font='Arial 12 bold', fg='#20D2B5')
        gen.grid(row=2, column=y, sticky=tk.W)
        email = tk.Label(labelframe, text="Email", font='Arial 12 bold', fg='#20D2B5')
        email.grid(row=3, column=y, sticky=tk.W)
        con = tk.Label(labelframe, text="Contact", font='Arial 12 bold', fg='#20D2B5')
        con.grid(row=4, column=y, sticky=tk.W)
        lugg = tk.Label(labelframe, text="Luggage", font='Arial 12 bold', fg='#20D2B5')
        lugg.grid(row=5, column=y, sticky=tk.W)
        tkn1 = tk.Label(labelframe, text="Ticket Number", font='Arial 12 bold',
fg='#20D2B5')
        tkn1.grid(row=6, column=y, sticky=tk.W)
        sno = tk.Label(labelframe, text="Seat Number", font='Arial 12 bold', fg='#20D2B5')
        sno.grid(row=7, column=y, sticky=tk.W)
        depp = tk.Label(labelframe, text="From", font='Arial 12 bold', fg='#20D2B5')
        depp.grid(row=8, column=y, sticky=tk.W)
```

```
        dess = tk.Label(labelframe, text="To", font='Arial 12 bold', fg='#20D2B5')
        dess.grid(row=9, column=y, sticky=tk.W)
        faa = tk.Label(labelframe, text="Total Fare", font='Arial 12 bold', fg='#20D2B5')
        faa.grid(row=10, column=y, sticky=tk.W)
        dobb = tk.Label(labelframe, text="Date of Boarding", font='Arial 12 bold',
fg='#20D2B5')
        dobb.grid(row=11, column=y, sticky=tk.W)
        doaa = tk.Label(labelframe, text="Date of Arrival", font='Arial 12 bold',
fg='#20D2B5')
        doaa.grid(row=12, column=y, sticky=tk.W)


        r = 2
        for n, a, g, e, c, l, tkn, sn, dep, des, fa, dob, doa in mydata:
            global ticketno
            ticketno = tkn
            if a >= 18:
                tfa = fa + (l * 200)
            else:
                tfa = (fa * 0.95) + (l * 150)
            name = tk.Label(labelframe, text=n, font='Arial 12 bold', fg='black')
            name.grid(row=0, column=r, sticky=tk.W)
            age = tk.Label(labelframe, text=a, font='Arial 12 bold', fg='black')
            age.grid(row=1, column=r, sticky=tk.W)
            gen = tk.Label(labelframe, text=g, font='Arial 12 bold', fg='black')
            gen.grid(row=2, column=r, sticky=tk.W)
            email = tk.Label(labelframe, text=e, font='Arial 12 bold', fg='black')
            email.grid(row=3, column=r, sticky=tk.W)
            con = tk.Label(labelframe, text=c, font='Arial 12 bold', fg='black')
```

```
            con.grid(row=4, column=r, sticky=tk.W)
            lugg = tk.Label(labelframe, text=l, font='Arial 12 bold', fg='black')
            lugg.grid(row=5, column=r, sticky=tk.W)
            tkn1 = tk.Label(labelframe, text=tkn, font='Arial 12 bold', fg='black')
            tkn1.grid(row=6, column=r, sticky=tk.W)
            sno = tk.Label(labelframe, text=sn, font='Arial 12 bold', fg='black')
            sno.grid(row=7, column=r, sticky=tk.W)
            depp = tk.Label(labelframe, text=dep, font='Arial 12 bold', fg='black')
            depp.grid(row=8, column=r, sticky=tk.W)
            dess = tk.Label(labelframe, text=des, font='Arial 12 bold', fg='black')
            dess.grid(row=9, column=r, sticky=tk.W)
            faa = tk.Label(labelframe, text=tfa, font='Arial 12 bold', fg='black')
            faa.grid(row=10, column=r, sticky=tk.W)
            dobb = tk.Label(labelframe, text=dob, font='Arial 12 bold', fg='black')
            dobb.grid(row=11, column=r, sticky=tk.W)
            doaa = tk.Label(labelframe, text=doa, font='Arial 12 bold', fg='black')
            doaa.grid(row=12, column=r, sticky=tk.W)
        labelframe.place(x=40, y=80)

        button = tk.Button(root4, text="Download", width=20, height=2, bg='white',
command=download_ticket)
        button.place(x=100, y=430)
    except:
      messagebox.showerror(title="Error",message="Invalid Input")

# USER FUNCTION 3 : UPDATION OF TICKET INFORMATION
# INPUTS THE TICKET NUMBER
def update_ticket_information():
```

```
    root2.destroy()
    global root3
    root3 = tk.Tk()
    root3.geometry("450x300")
    root3.title("Bus Booking System")

    label = tk.Label(root3, text="Updation:", font=("Arial Bold", 20), fg='dark orange')
    label.pack(padx=30, pady=30)

    tkn = tk.Label(root3, text="Enter Ticket Number:", font='Arial 18 bold', fg='black')
    tkn.place(x=100, y=110)
    global tkn_ent2
    tkn_ent2 = tk.Entry(root3, width=30)
    tkn_ent2.place(x=130, y=155)

    button = tk.Button(root3, text="Enter", width=20, height=2, bg='white',
command=update_ticket_information_interface)
    button.place(x=146, y=200)

# DISPLAYS OPTIONS TO UPDATE
def update_ticket_information_interface():
    global tkn
    tkn = int(tkn_ent2.get())
    mycursor.execute("select * from tickets\
                     where TicketNumber=%s" % tkn)
    mydata = mycursor.fetchall()
    if mydata == []:
        messagebox.showerror(title="Error", message="Ticket Not Found")
```

```
    else:
        root3.destroy()
        global root4
        root4 = tk.Tk()
        root4.geometry("470x450")
        root4.title("Bus Booking System:Update Ticket")


        he = tk.Label(root4, text="Select Your Choice:", font='Arial 18 bold', fg='dark orange')
        he.pack(padx=30, pady=30)


        button1 = tk.Button(root4, text="Name", width=20, height=2, bg='white',
command=new_name)
        button1.place(x=150, y=100)


        button2 = tk.Button(root4, text="Age", width=20, height=2, bg='white',
command=new_age)
        button2.place(x=150, y=150)


        button3 = tk.Button(root4, text="Gender", width=20, height=2, bg='white',
command=new_gen)
        button3.place(x=150, y=200)


        button4 = tk.Button(root4, text="Email", width=20, height=2, bg='white',
command=new_email)
        button4.place(x=150, y=250)


        button5 = tk.Button(root4, text="Contact", width=20, height=2, bg='white',
command=new_con)
        button5.place(x=150, y=300)
```

```
        button6 = tk.Button(root4, text="Luggage", width=20, height=2, bg='white',
command=new_lug)
        button6.place(x=150, y=350)


# OPTION 1 : NAME
def new_name():
    root4.destroy()
    global root5
    root5 = tk.Tk()
    root5.geometry("500x250")
    root5.title("Bus Booking System:Update Ticket")
    nnam = tk.Label(root5, text="Enter New Name:", font='Arial 18 bold', fg='black')
    nnam.place(x=40, y=70)
    global nname_ent
    nname_ent = tk.Entry(root5, width=30)
    nname_ent.place(x=250, y=80)
    button = tk.Button(root5, text="Enter", width=20, height=2, bg='white',
command=up_name)
    button.place(x=170, y=150)


# NAME UPDATION
def up_name():
    nname = nname_ent.get()
    mycursor.execute("update tickets\
                set Name='%s'\
                where TicketNumber=%d " % (nname, tkn))
    f = open("ticket1.dat", "rb+")
```

```
    try:
        while True:
            pos = f.tell()
            mydata = pickle.load(f)
            if mydata[6] == tkn:
                mydata[0] = nname
                f.seek(pos)
                pickle.dump(mydata, f)
                break
    except EOFError:
        f.close()
    messagebox.showinfo(title="Update Ticket", message="UPDATION SUCCESSFUL")
    root5.destroy()
    user_home_page()
    mycon.commit()

# OPTION 2 : AGE
def new_age():
    root4.destroy()
    global root5
    root5 = tk.Tk()
    root5.geometry("500x250")
    root5.title("Bus Booking System:Update Ticket")
    nage1 = tk.Label(root5, text="Enter New Age:", font='Arial 18 bold', fg='black')
    nage1.place(x=50, y=70)
    global nage_ent
    nage_ent = tk.Entry(root5, width=30)
    nage_ent.place(x=240, y=80)
```

```
    button = tk.Button(root5, text="Enter", width=20, height=2, bg='white',
command=up_age)
    button.place(x=170, y=150)


# AGE UPDATION
def up_age():
    nage = int(nage_ent.get())
    mycursor.execute("update tickets\
                set Age=%s\
                where TicketNumber=%d " % (nage, tkn))
    f = open("ticket1.dat", "rb+")
    try:
        while True:
            pos = f.tell()
            mydata = pickle.load(f)
            if mydata[6] == tkn:
                mydata[1] = nage
                f.seek(pos)
                pickle.dump(mydata, f)
                break
    except EOFError:
        f.close()


    messagebox.showinfo(title="Update Ticket", message="UPDATION SUCCESSFUL")
    root5.destroy()
    user_home_page()
    mycon.commit()
```

```
# OPTION 3 : GENDER
def new_gen():
    root4.destroy()
    global root5
    root5 = tk.Tk()
    root5.geometry("500x250")
    root5.title("Bus Booking System:Update Ticket")
    ngen1 = tk.Label(root5, text="Enter New Gender:", font='Arial 18 bold', fg='black')
    ngen1.place(x=25, y=70)
    global ngen_ent
    ngen_ent = tk.Entry(root5, width=30)
    ngen_ent.place(x=250, y=80)
    button = tk.Button(root5, text="Enter", width=20, height=2, bg='white',
command=up_gen)
    button.place(x=170, y=150)


# GENDER UPDATION
def up_gen():
    ngen = ngen_ent.get()
    mycursor.execute("update tickets\
                set Gender='%s'\
                where TicketNumber=%d " % (ngen, tkn))
    f = open("ticket1.dat", "rb+")
    try:
        while True:
            pos = f.tell()
            mydata = pickle.load(f)
            if mydata[6] == tkn:
```

```
                mydata[2] = ngen
                f.seek(pos)
                pickle.dump(mydata, f)
                break
    except EOFError:
        f.close()
    messagebox.showinfo(title="Update Ticket", message="UPDATION SUCCESSFUL")
    root5.destroy()
    user_home_page()
    mycon.commit()

# OPTION 4 : EMAIL
def new_email():
    root4.destroy()
    global root5
    root5 = tk.Tk()
    root5.geometry("500x250")
    root5.title("Bus Booking System:Update Ticket")
    nemail1 = tk.Label(root5, text="Enter New Email:", font='Arial 18 bold', fg='black')
    nemail1.place(x=40, y=70)
    global nemail_ent
    nemail_ent = tk.Entry(root5, width=30)
    nemail_ent.place(x=250, y=80)
    button = tk.Button(root5, text="Enter", width=20, height=2, bg='white',
command=up_email)
    button.place(x=170, y=150)

# EMAIL UPDATION
```

```
def up_email():
    nemail = nemail_ent.get()
    mycursor.execute("update tickets\
                set Email='%s'\
                where TicketNumber=%d " % (nemail, tkn))
    f = open("ticket1.dat", "rb+")
    try:
        while True:
            pos = f.tell()
            mydata = pickle.load(f)
            if mydata[6] == tkn:
                mydata[3] = nemail
                f.seek(pos)
                pickle.dump(mydata, f)
                break
    except EOFError:
        f.close()

    messagebox.showinfo(title="Update Ticket", message="UPDATION SUCCESSFUL")
    root5.destroy()
    user_home_page()
    mycon.commit()

# OPTION 5 : CONTACT NUMBER
def new_con():
    root4.destroy()
    global root5
    root5 = tk.Tk()
```

```
    root5.geometry("500x250")
    root5.title("Bus Booking System:Update Ticket")
    ncon1 = tk.Label(root5, text="Enter New Contact:", font='Arial 18 bold', fg='black')
    ncon1.place(x=25, y=70)
    global ncon_ent
    ncon_ent = tk.Entry(root5, width=30)
    ncon_ent.place(x=260, y=80)
    button = tk.Button(root5, text="Enter", width=20, height=2, bg='white',
command=up_con)
    button.place(x=170, y=150)


# CONTACT NUMBER UPDATION
def up_con():
    ncon = ncon_ent.get()
    mycursor.execute("update tickets\
                set Contact='%s'\
                where TicketNumber=%d " % (ncon, tkn))
    f = open("ticket1.dat", "rb+")
    try:
        while True:
            pos = f.tell()
            mydata = pickle.load(f)
            if mydata[6] == tkn:
                mydata[4] = ncon
                f.seek(pos)
                pickle.dump(mydata, f)
                break
    except EOFError:
```

```
        f.close()
    messagebox.showinfo(title="Update Ticket", message="UPDATION SUCCESSFUL")
    root5.destroy()
    user_home_page()
    mycon.commit()

# OPTION 6 : LUGGAGE
def new_lug():
    root4.destroy()
    global root5
    root5 = tk.Tk()
    root5.geometry("500x250")
    root5.title("Bus Booking System:Update Ticket")
    nlug = tk.Label(root5, text="Enter New Luggage:", font='Arial 18 bold', fg='black')
    nlug.place(x=25, y=70)
    global nlugg_ent
    nlugg_ent = tk.Entry(root5, width=30)
    nlugg_ent.place(x=270, y=80)
    button = tk.Button(root5, text="Enter", width=20, height=2, bg='white', command=up_lug)
    button.place(x=170, y=150)

# LUGGAGE UPDATION
def up_lug():
    nlugg = int(nlugg_ent.get())
    mycursor.execute("update tickets\
                set Luggage=%s\
                where TicketNumber=%d " % (nlugg, tkn))
    f = open("ticket1.dat", "rb+")
```

```
    try:
        while True:
            pos = f.tell()
            mydata = pickle.load(f)
            if mydata[6] == tkn:
                mydata[5] = nlugg
                f.seek(pos)
                pickle.dump(mydata, f)
                break
    except EOFError:
        f.close()
    messagebox.showinfo(title="Update Ticket", message="UPDATION SUCCESSFUL")
    root5.destroy()
    user_home_page()
    mycon.commit()

# USER FUNCTION 4 : CANCEL TICKET
def ticket_cancellation_interface():
    root2.destroy()
    global root3
    root3 = tk.Tk()
    root3.geometry("500x180")
    root3.title("Bus Booking System:Cancel Ticket")

    tkn = tk.Label(root3, text="Enter Ticket Number:", font='Arial 18 bold', fg='black')
    tkn.place(x=20, y=50)
    global tkn_ent
    tkn_ent = tk.Entry(root3, width=30)
```

```
    tkn_ent.place(x=270, y=60)


    button = tk.Button(root3, text="Enter", width=20, height=2, bg='white',
command=cancel_ticket)
    button.place(x=170, y=110)


# DELETING THE RECORD FROM THE DATABASE/BINARY FILE
def cancel_ticket():
    tkn = int(tkn_ent.get())
    mycursor.execute("select * from tickets")
    mydata = mycursor.fetchall()
    c = 0
    for n, a, g, e, c, lug1, tn, co, sn in mydata:
        if tn == tkn:
            c = 1
            mycursor.execute("delete from tickets\
                    where TicketNumber=%s" % tkn)


            f = open('ticket1.dat', 'rb')
            n = open('ticket2.dat', 'wb')
            try:
                while True:
                    s = pickle.load(f)
                    if s[6] != tkn:
                        pickle.dump(s, n)
            except EOFError:
                f.close()
                n.close()
```

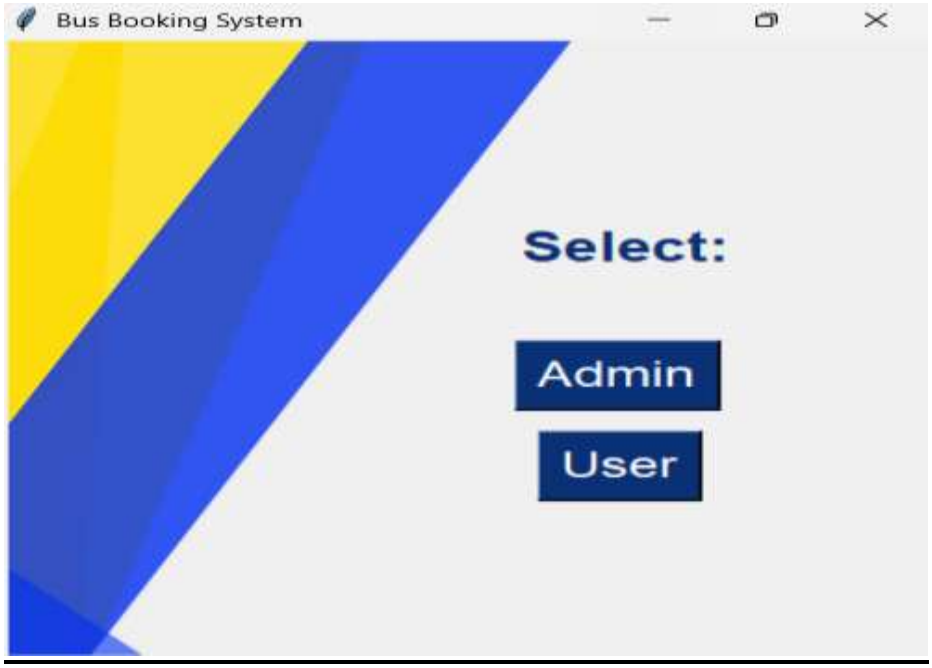
```
            os.remove('ticket1.dat')
            os.rename('ticket2.dat', 'ticket1.dat')

        messagebox.showinfo(title="Cancel Ticket", message="Cancellation Successful")
        mycursor.execute("select * from schedule where code=%s" % co)
        mydata = mycursor.fetchall()
        for b, d, f, dob, doa, co, ts, vs in mydata:
            vs += 1
            mycursor.execute("update schedule set vacantseats=%s where code=%d" % (vs,
co))

        f = open("schedules.dat", "rb+")
        while True:
            try:
                pos = f.tell()
                mydata = pickle.load(f)
                if mydata[5] == co:
                    mydata[7] = int(mydata[7] + 1)
                    f.seek(pos)
                    pickle.dump(mydata, f)
                    break
            except EOFError:
                f.close()
        root3.destroy()
        user_home_page()
        mycon.commit()
        break
    else:
```

```
        messagebox.showerror(title="Error", message="Ticket Not Found")


# USER FUNCTION 5 : EXIT
def user_exit():
    if messagebox.askyesno(title="Quit?", message="Back to Home Page?"):
        root2.destroy()
        main()


# DELETES THE RECORDS FROM SCHEDULE AND TICKETS DATABASE WHO'S
DATE OF BOARDING HAS PASSED THE CURRENT DATE
def delete_daily_database_records():
    mycursor.execute("select *,curdate() from schedule")
    mydata = mycursor.fetchall()
    for dep, des, fa, dob, doa, code, ts, vs, curdate in mydata:
        mycursor.execute("select code from schedule\
        where DOB<'%s' " % curdate)
        mydata2 = mycursor.fetchall()
        mycursor.execute("delete from schedule\
        where DOB<'%s' " % curdate)


        for code in mydata2:
            mycursor.execute("delete from tickets\
            where code=%d " % code)
            f = open('schedules.dat', 'rb')
            n = open('schedules2.dat', 'wb')
            try:
                while True:
                    s = pickle.load(f)
```

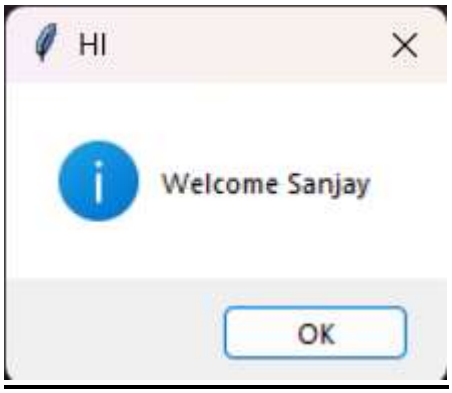
```
                if s[5] != code:
                    pickle.dump(s, n)
            except EOFError:
                f.close()
                n.close()
                os.remove('schedules.dat')
                os.rename('schedules2.dat', 'schedules.dat')

            f = open('ticket1.dat', 'rb')
            n = open('ticket2.dat', 'wb')
            try:
                while True:
                    s = pickle.load(f)
                    if s[5] != code:
                        pickle.dump(s, n)
            except EOFError:
                f.close()
                n.close()
                os.remove('ticket1.dat')
                os.rename('ticket2.dat', 'ticket1.dat')
        mycon.commit()

    # GOES BACK TO THE ADMIN HOME PAGE
    def on_closing_root4_admin():
        if messagebox.askyesno(title="Quit?", message="Back to Home Page?"):
            root4.destroy()
            admin_home_page()
```

```
# GOES BACK TO THE USER HOME PAGE
def on_closing_root4_user():
    if messagebox.askyesno(title="Quit?", message="Back to Home Page?"):
        root4.destroy()
        user_home_page()

# GOES BACK TO THE MAIN HOME PAGE
def on_closing_root2_admin():
    if messagebox.askyesno(title="Quit?", message="Back to Home Page?"):
        root2.destroy()
        main()

main()
# delete_daily_database_records()
```

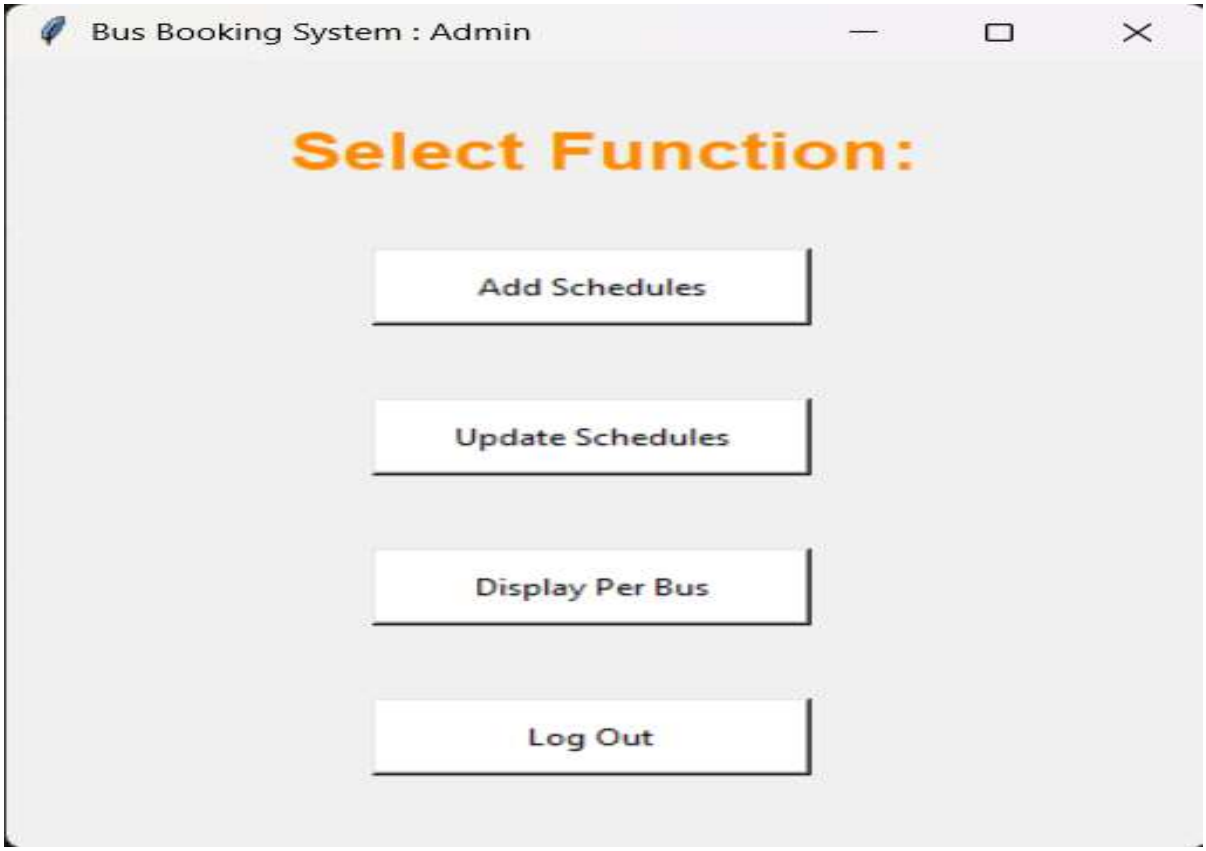
Output:

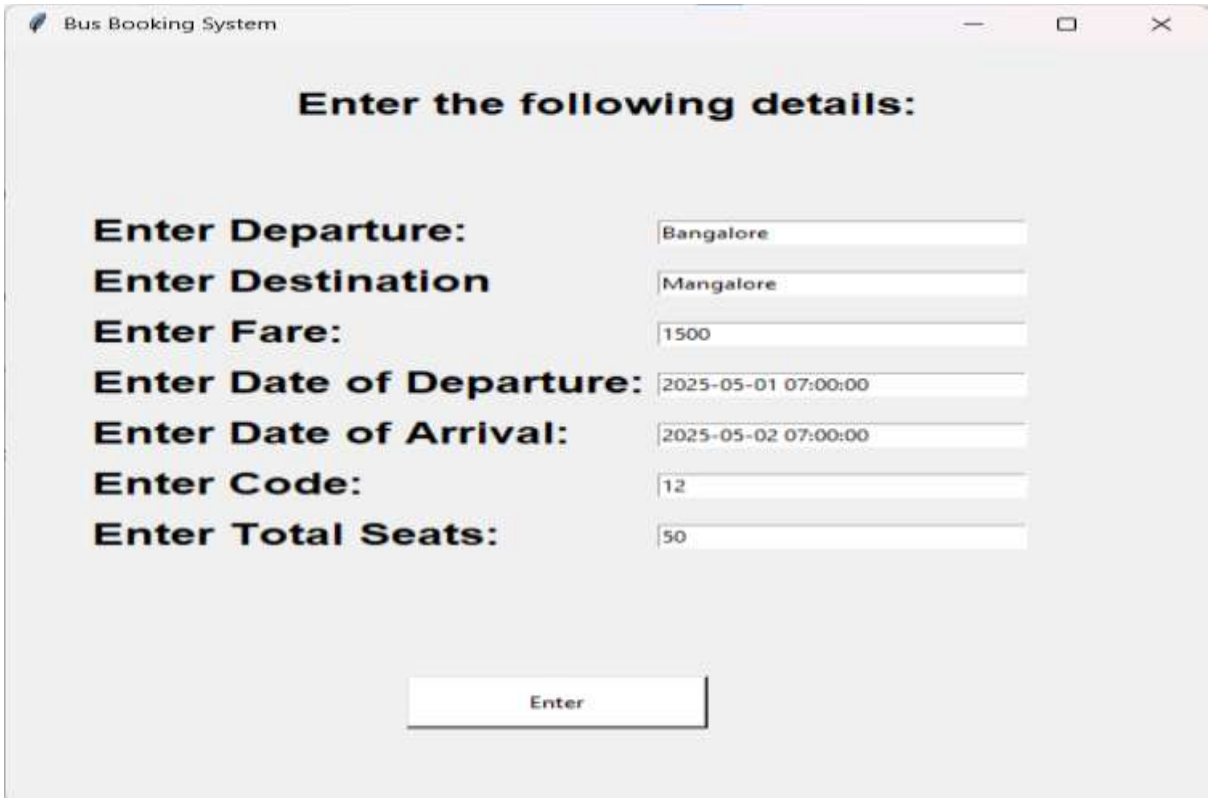

Bus Booking System
Select:
Admin
User

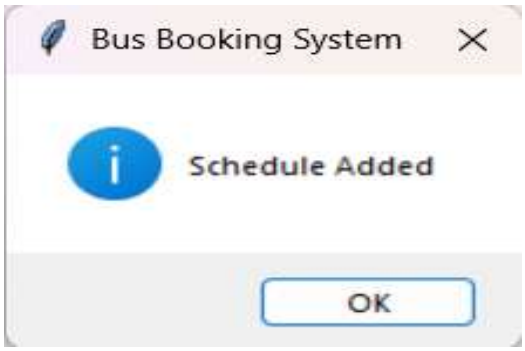

Bus Booking System
Authentication
Enter Admin Name:
Enter Admin Password:
Enter

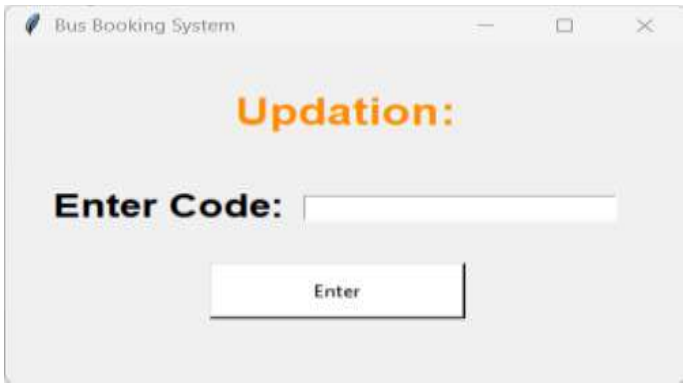

HI
Welcome Sanjay
OK

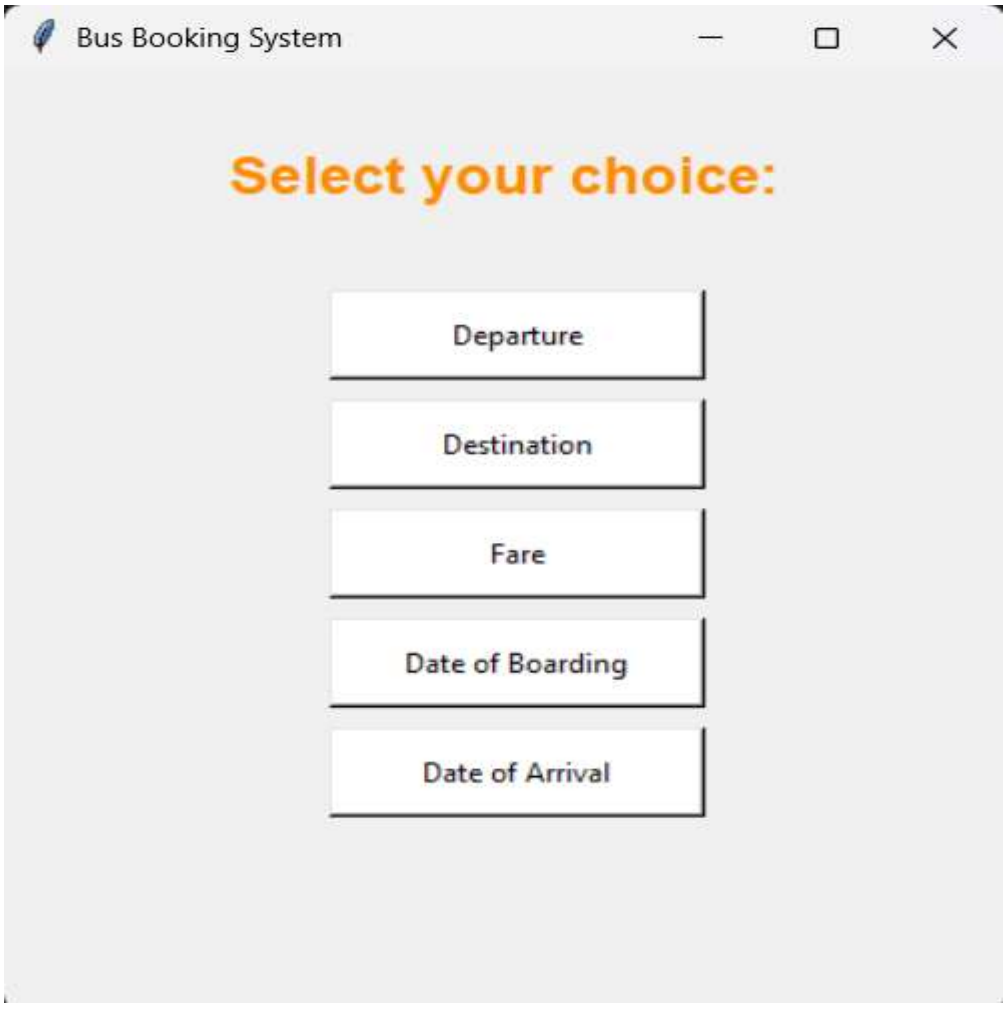
Admin Modules:

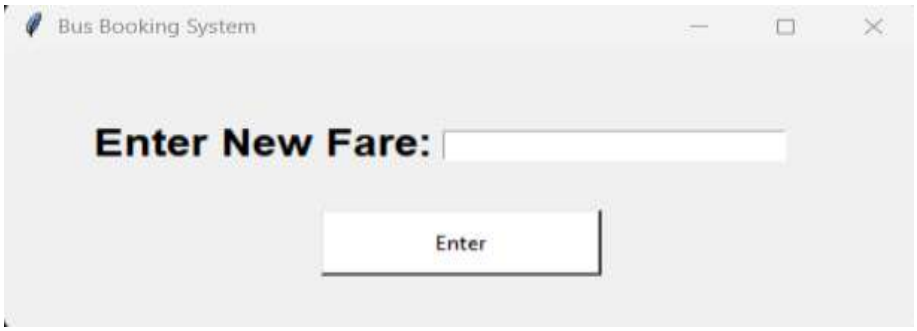




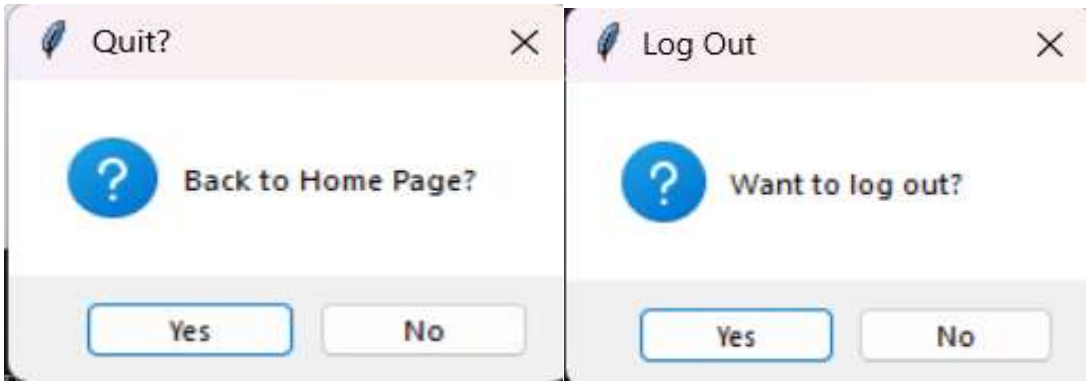

Bus Booking System
Schedule Added
OK


Bus Booking System
Updation:
Enter Code:
Enter

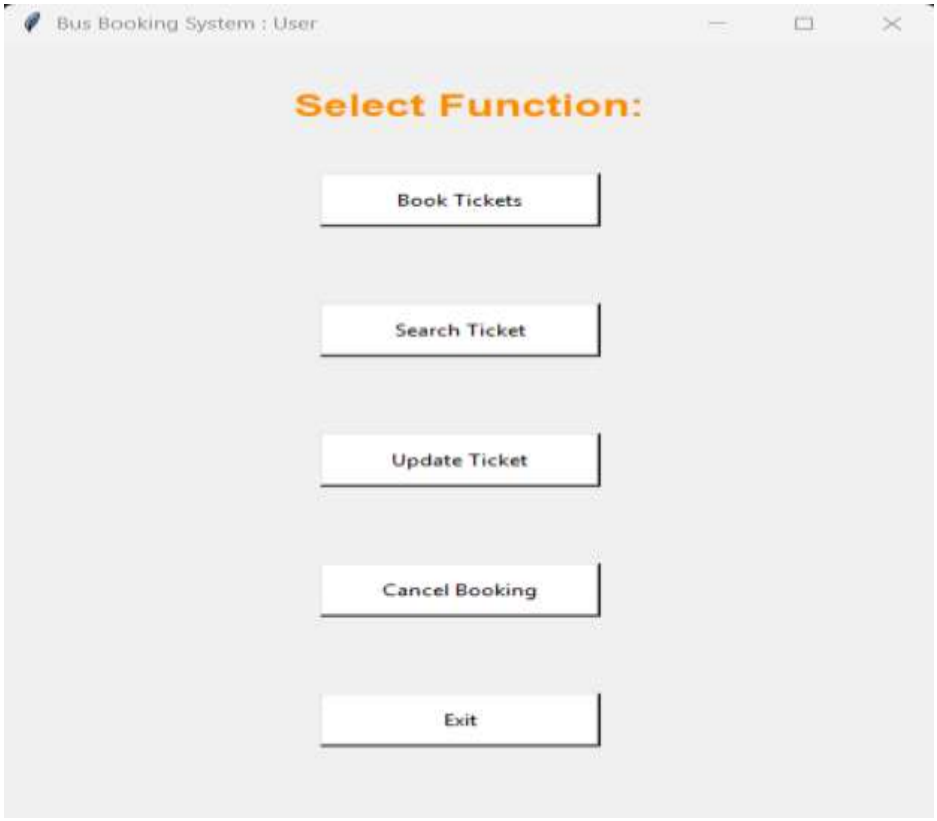

Bus Booking System
Select your choice:
Departure
Destination
Fare
Date of Boarding
Date of Arrival

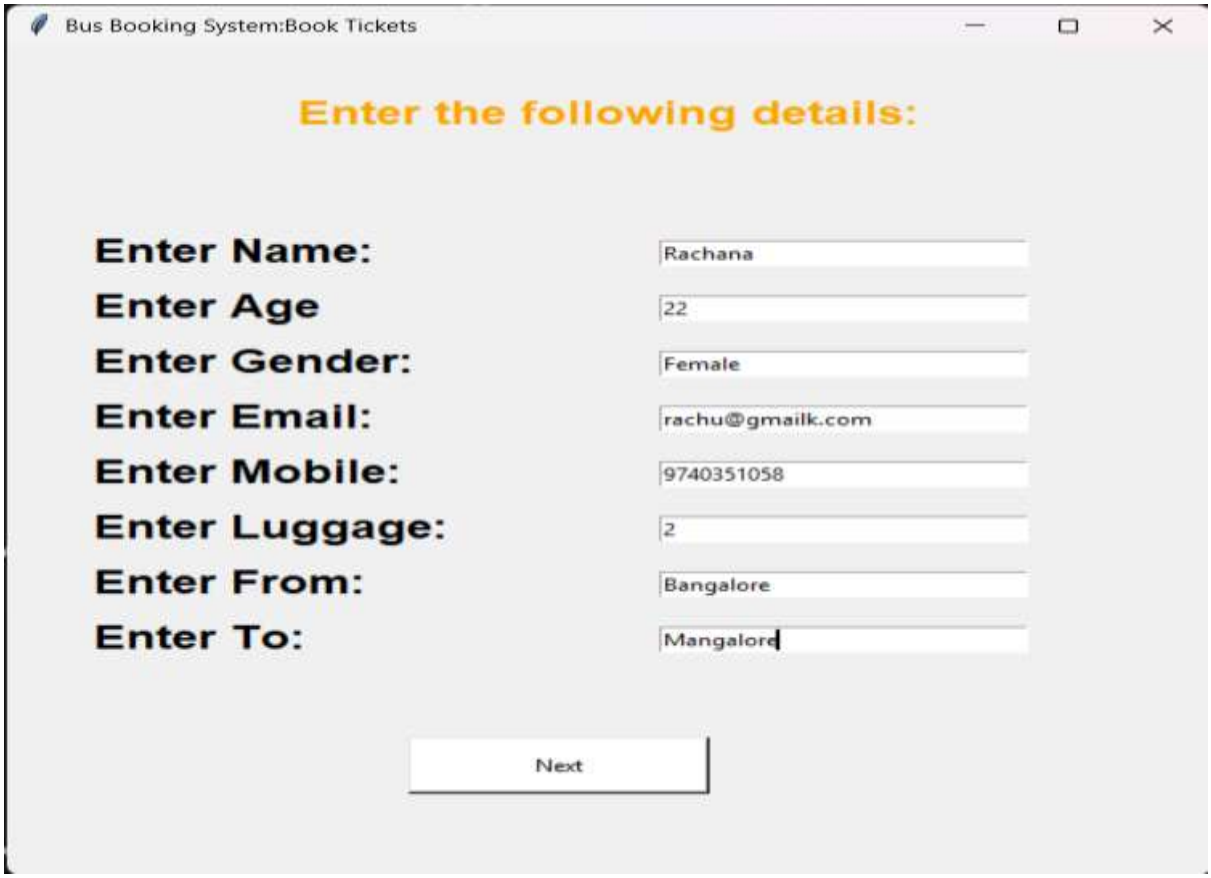


Name	Age	Gender	Email	Contact	Luggage	Ticket Number	Code	Seat Number
Asmit Samuel	22	Male	samasmit@gmail.com	9139129102	1	1002	7	C6
Kevin Jose	17	Male	kevin@gmail.com	9910217991	2	6037	7	D8
Aditya Menon	17	Male	menon@gmail.com	9856423596	2	2534	7	G3
Sanya	23	Female	sanya@gmail.com	9754862145	1	2213	7	H13
Jose Augustine	51	Male	jose64@gmail.com	9978221456	1	8976	7	C10
Alina Mattew	17	Female	alinawth@gmail.com	9982640036	2	9569	7	B1
Alisha Boban	18	Female	alisha21@gmail.com	9301624003	1	8379	7	E1
Samairra	17	Female	sam34@gmail.com	9635874221	1	6419	7	G5



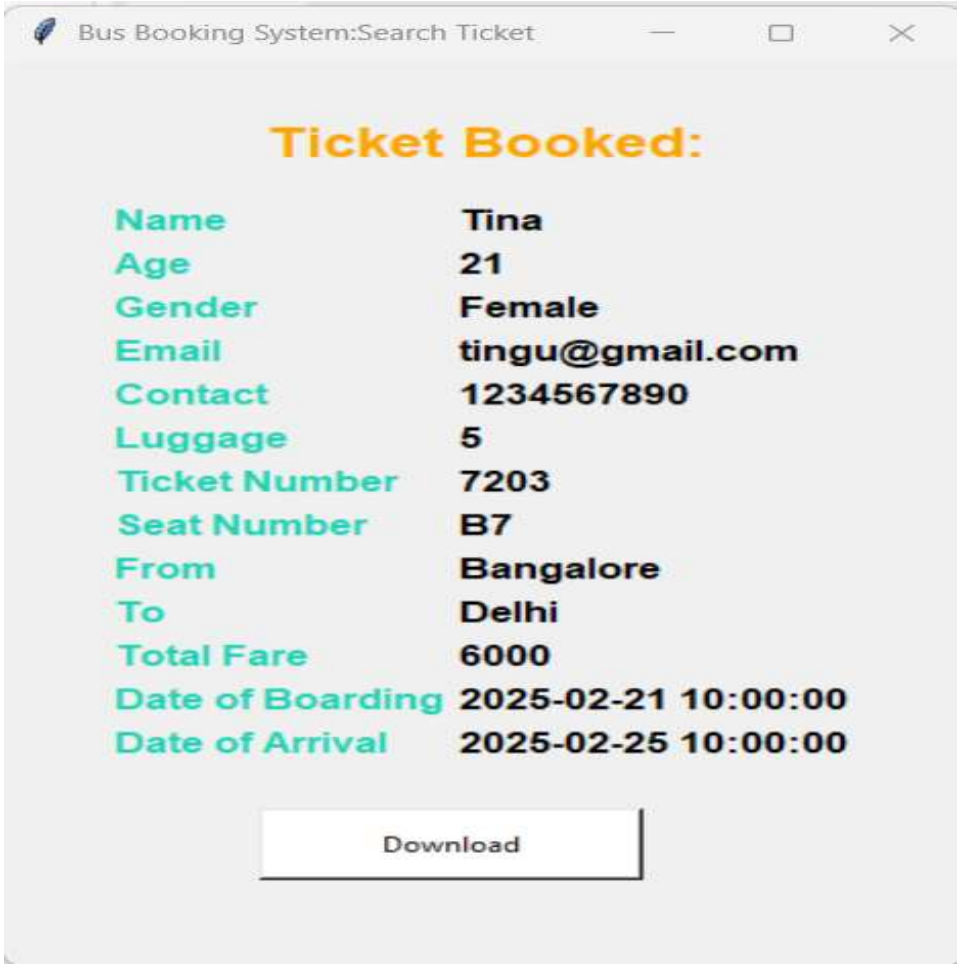
User Modules:





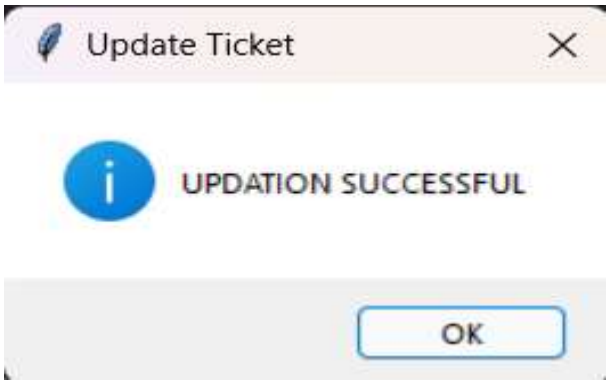

Bus Booking System
AVAILABLE BUSES:
DEPARTURE DESTINATION FARE DATE OF BOARDING DATE OF ARRIVAL CODE TOTAL SEATS VACANT SEATS
Bangalore Mangalore 1500 2025-05-01 07:00:00 2025-05-02 07:00:00 12 50 50

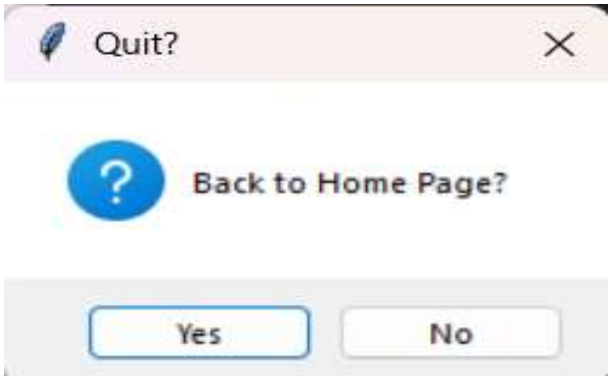

Bus Booking System:Search Ticket
Enter Ticket Number:
7203
Enter


Bus Booking System:Search Ticket
Ticket Booked:
Name Tina
Age 21
Gender Female
Email tingu@gmail.com
Contact 1234567890
Luggage 5
Ticket Number 7203
Seat Number B7
From Bangalore
To Delhi
Total Fare 6000
Date of Boarding 2025-02-21 10:00:00
Date of Arrival 2025-02-25 10:00:00
Download


Bus Booking System
Updation:
Enter Ticket Number:
2636
Enter


Bus Booking System:Update Ticket
Enter New Age:
23
Enter


Update Ticket
UPDATION SUCCESSFUL
OK


Quit?
Back to Home Page?
Yes
No

Conclusion

The **Bus Management System** effectively addresses the challenges faced in managing bus operations manually. By automating key functions such as route and schedule management, ticket booking, and user handling, the system enhances operational efficiency, reduces errors, and provides a more convenient experience for both administrators and users.

The implementation of Python for backend development and MySQL for database management ensures robust performance and secure data handling. The system's intuitive interface allows users to easily interact with its features, making the booking and management process simple and efficient.

Overall, the project successfully demonstrates how digital solutions can transform traditional transportation services into a more organized, scalable, and user-friendly platform. It lays the foundation for future enhancements making it a valuable tool for modern bus service management.