**Flight Price Prediction**

PROJECT REPORT

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

By:

|  |  |  |  |
| --- | --- | --- | --- |
| *S.no.* | *Name* | *Roll No.* | *Registration no.* |
| *1.* | *Arush Nigam* | *25* | *11902201* |
| *2.* | *Mohit Pal Singh* | *23* | *11902185* |
| *3.* | *Swarit Srivastava* | *11* | *11902083* |

**Courses Code: INT213**

**School of Computer Science and Engineering**

Lovely Professional University

Phagwara, Punjab (India)

Page: 1

**Work Distribution:**

***Mohit Pal Singh:***

*Backend part and algorithmic part of the project, GitHub upload.*

***Swarit Srivastava:***

*The frontend Part of project using GUI/ design of interface.*

***Arush Nigam:***

*Database creation and connectivity, Report Making, power point presentation.*

Page 2

**Objective:**

The main objective of this project is to implement the knowledge we have gained through the course of Python programming (INT213) and to create a Graphical User Interface for our project that if flight price perdition interface with all the required functionalities. The major of this project is to provide a user-friendly interface so, the user can easily access the interface and could easily get the required information. Through this small project we are providing a kind of helpful page to those who wants to know about the flight rates from one place to another just by filling some required inputs. If a person wants to check a rough price prediction of his/her journey, he/she can use our project for that.

Our experience that we have gained from this project are listed below: -

* Team work - as we have a team of 3 members so we have tried to divide the tasks equally in the whole team.
* Time management.
* Practical implementation of python programming.
* Database connectivity.
* Some approaches for resolving the error.

.

Page 3

**Introduction:**

Our project, the flight price predictor is an interface where, just by filling some inputs and by selecting the required option one can view the predicted price for his journey form one place to another. As a person opens our project our first welcome page is displayed containing the name of project, the logo and the names of creators and by clicking into the given button user can view the flight price predictor. The entries/input that are required to be field by the user are listed below:

* One - way or Two-way trip.
* Boarding
* Destination
* Date
* Timing
* Preferred Airline

There are only two major screens in this project that is a welcome page and then our predictor. We have not kept any login window because our project is just like a calculator in which we give some logical inputs and the result is ready in a single click so, we have just kept it simple and user friendly.

In this project we have used tkinter programming, tkinter widgets like buttons etc., GUI and database more functions used can be seen below in the report int the source code part.

Page 4

A flight price predictor can help the user in many ways like: -

* One can check if the price of his journey matches his budget or not. So, he can plan further accordingly.
* One can check the price according to the timing he wants and also the dates he wants.
* One can also check the predicted price of the return trip also.

Now let’s go with the project description of our output screen using the screenshots.

* **Welcome screen:**

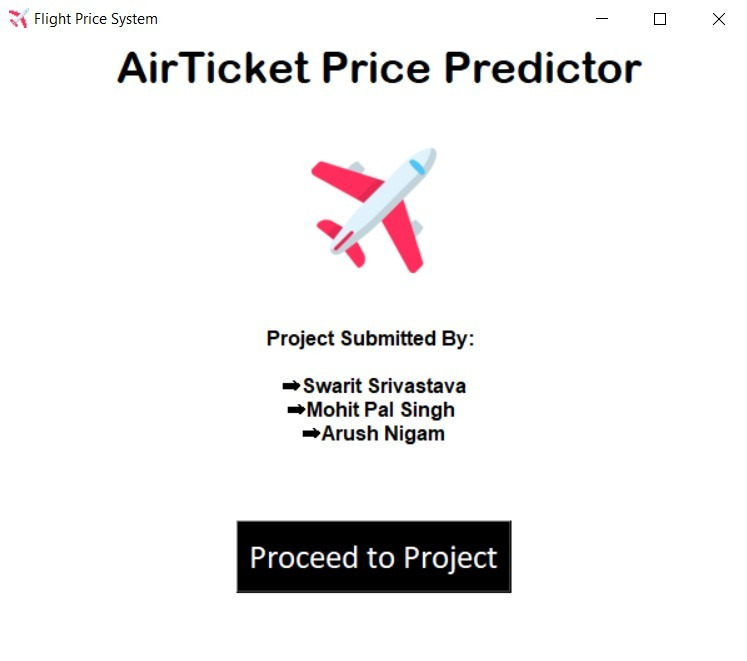
As shown blow in the figure 1, This is our welcome screen that is displayed firstly. In this we have added the logo of our project which is a plane as shown, the name of our project – AIR Ticket Price Predictor, is also shown.

Then after we have added the names of the creators of this project below the logo.

On the most bottom a button is created with a black background color, by clicking on this button user will be forwarded to the price predictor.

Page 5

The welcome screen screenshot –



**Figure 1**

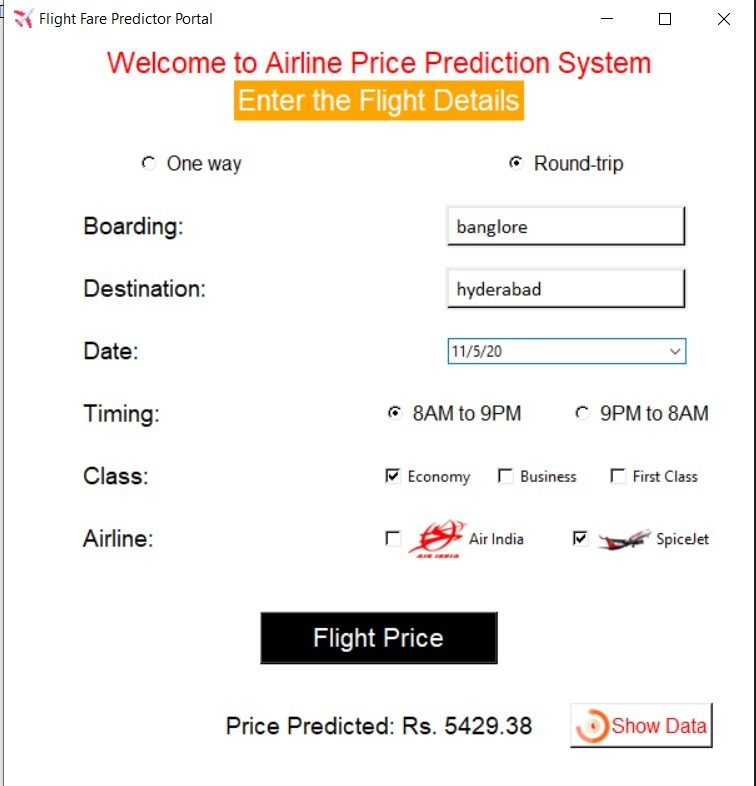
* **The main price predictor:**

As shown below in the figure 2, this is the main window of our project, it welcomes the user and asks to fill up the required details to get the result. User gets various inputs which are necessary to be filled as these inputs acts as a base for the prediction according to our algorithm.

Page 6

For example, like we have kept an option of one - way trip or round – trip so, accordingly the price will differ and also, we have added an option to choose the preferred airlines because every company may have its different prices. One more important entry is the time because in some flights the price differs according to the time also. After giving all these details user can display the predicted price by clicking on the button below. At the last the show data button is created to show the data/history saved.

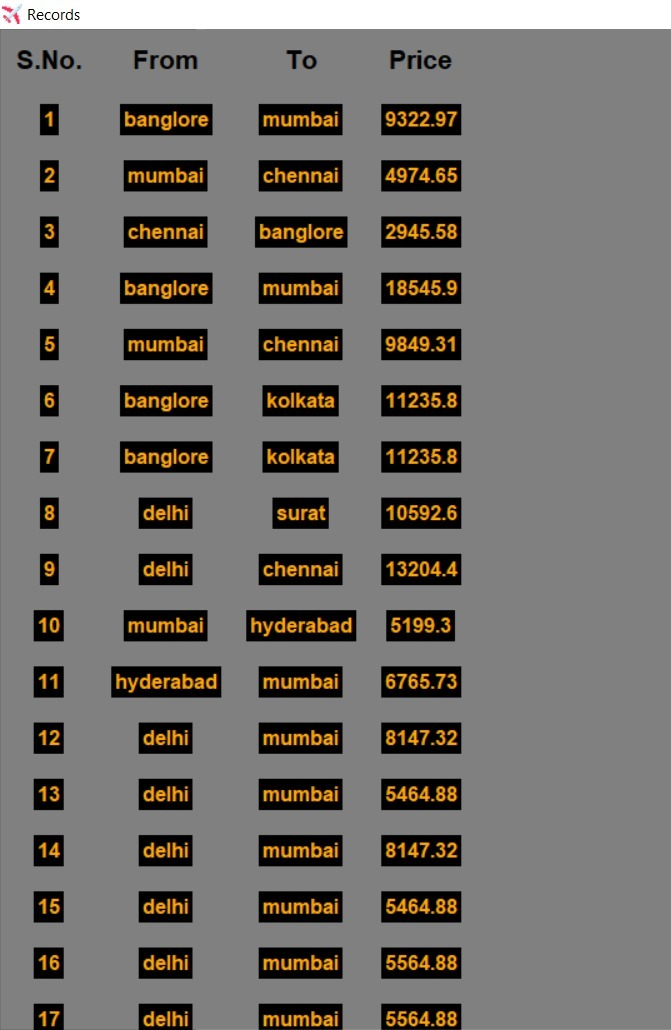
The main window screenshot –



**Figure 2**

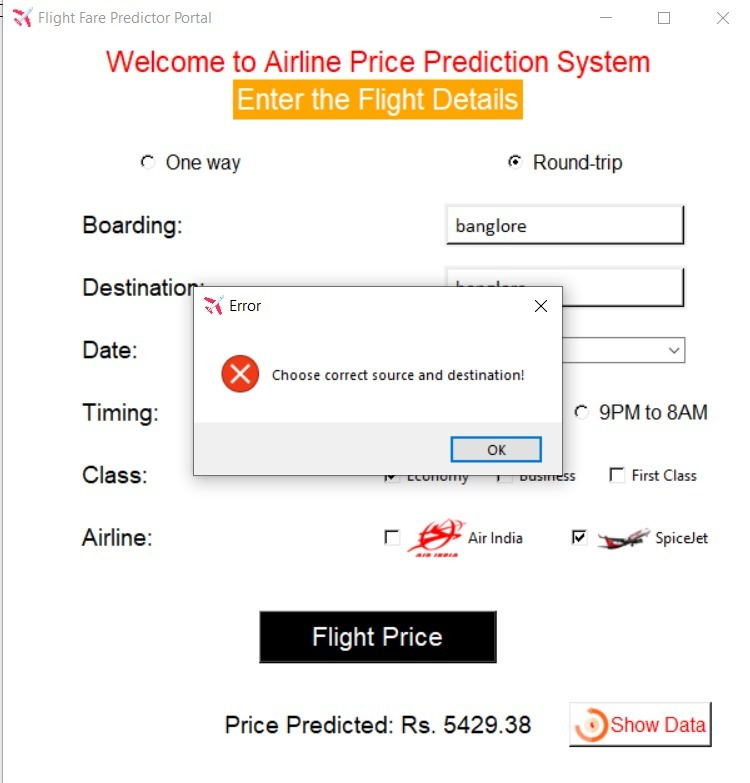
Page 7

When a person clicks on a show data part a screen is displayed showing the saved history.

The screenshot of database window screenshot:

Page 8

**Note:** if a person fills a wrong entry, same entries in destination and boarding then an error message is displayed as shown below:

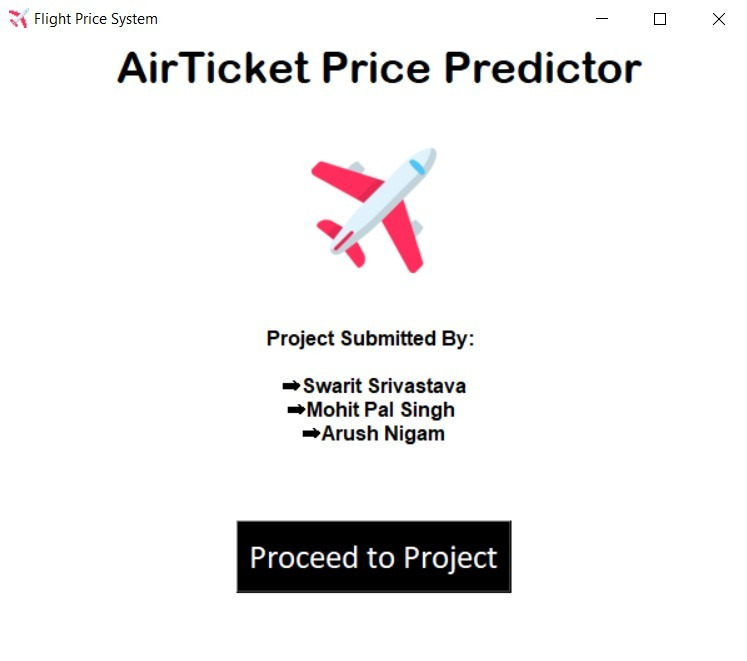


Page 9

**Steps to Execute the portal:**

Step 1.

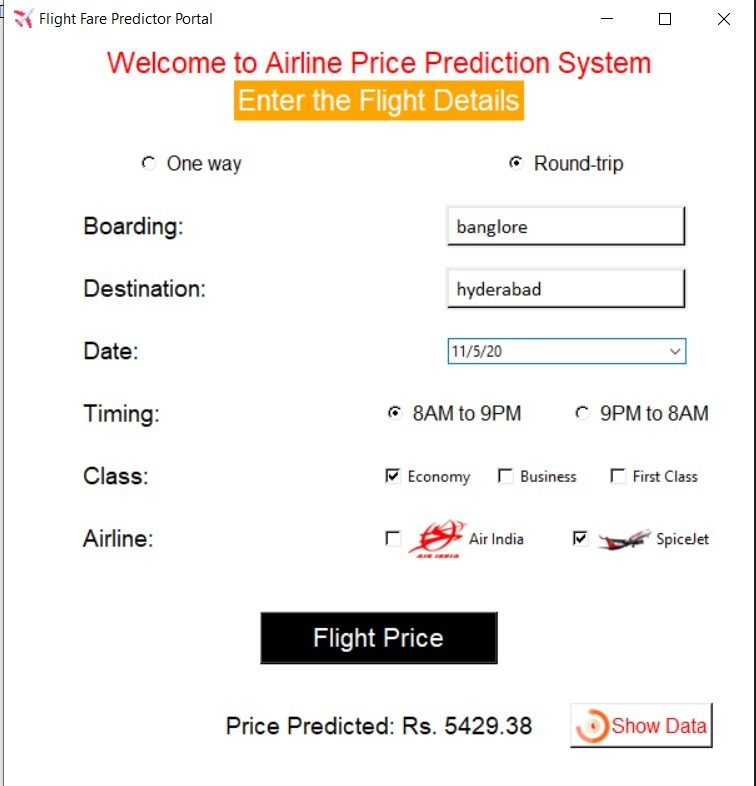
Opening the main window and clicking on the “Proceed to project” button:



Page 10

Step 2.

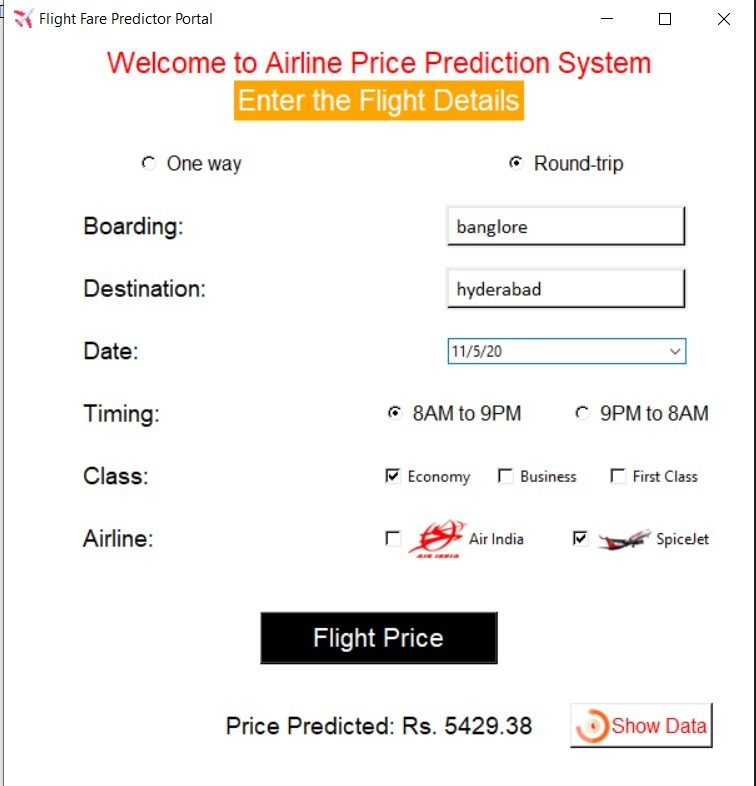
Fill up the required entries and click on the button “Flight Price” button and get the result.



Page 11

Final Step:

Check the saved history by clicking on “Show Data” button.



Page 12

**Conclusion:**

After completing all the parts of this project successfully, we have finally achieved our objectives. By following the strategy of the team work we could complete the project as the work was equally distributed to every member in the team. The main components of our project are Frontend, backend and database and all these are completed by the team members. Our project shows a user the predicted price for his journey through flight. Where user have to fill some entries and just in a one click, he can get the price, so we can say that our project is user friendly as well.

We have tried to keep our interface simple so that it does not confuse the user. And the best part is a person can check the history of his/her search through our database connectivity by clicking on the “show data” button.

We finally conclude that our displaying the required result with user friendly interface.

**References**

* [**https://www.w3schools.com/python/**](https://www.w3schools.com/python/)
* [**https://www.geeksforgeeks.org/sql-using-python/**](https://www.geeksforgeeks.org/sql-using-python/)
* [**https://stackoverflow.com/**](https://stackoverflow.com/)
* [**www.google.co.in**](http://www.google.co.in)
* [**www.quora.com**](http://www.quora.com)

Page 13

**The source code of the project:**

from tkinter import \*

import mysql.connector

from mysql.connector import Error

from tkinter import ttk, messagebox

from tkcalendar import Calendar, DateEntry

from datetime import datetime

try:

flag = 0

con = mysql.connector.connect(host='localhost', user='root', password='flightroot123',database='flight')

cur = con.cursor(buffered=True)

cur.execute("SHOW DATABASES")

for x in cur.fetchall():

for y in x:

if y == 'flight':

flag = 1

break

if flag != 1:

cur.execute("CREATE DATABASE flight")

cur = con.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS price\_record(From1 varchar(50),To1 varchar(50),price1 FLOAT);")

con.commit()

root = Tk()

root.geometry('600x500')

root.minsize(600, 500) # min width,height

root.maxsize(600, 500)

root.geometry("{}x{}+{}+{}".format(600, 500, int((root.winfo\_screenwidth() / 2) - (600 / 2)),

int((root.winfo\_screenheight() / 2) - (500 / 2))))

root.title("Flight Price System")

main\_icon = PhotoImage(file="travelling.png")

root.iconphoto(True, main\_icon)

root.configure(background="white")

heading1 = Label(root, text=" AirTicket Price Predictor\n", font=("Arial Rounded MT Bold", 26), bg='white',

fg='black')

heading1.pack()

main\_icon = PhotoImage(file='travelling.png')

main\_icon\_label = Label(root, image=main\_icon, bg='white')

main\_icon\_label.pack()

heading2 = Label(root,

text="\n\nProject Submitted By: \n\n➡Swarit Srivastava\n➡Mohit Pal Singh \n➡Arush Nigam\n\n\n",

font=("Helvetica 12 bold"), bg='white', fg='black')

heading2.pack()

def airline\_price():

root.destroy()

root1 = Tk()

root1.focus\_force() # To bring the window in focus

root1.title("Flight Fare Predictor Portal")

root1.geometry('600x600')

root1.minsize(600, 600) # min width,height

root1.maxsize(600, 600)

root1.geometry("{}x{}+{}+{}".format(600, 600, int((root1.winfo\_screenwidth() / 2) - (600 / 2)),

int((root1.winfo\_screenheight() / 2) - (600 / 2))))

root1.configure(background='white')

Label(root1, text="Welcome to Airline Price Prediction System", font=("Arial 93", 17), fg="Red",

bg="White").place(relx=0.5, y=20, anchor='center')

Label(root1, text="Enter the Flight Details", font=("Arial 93", 17), fg="White", bg="Orange").place(relx=0.5,

y=50,

anchor='center')

one\_or\_round = IntVar()

r1 = Radiobutton(root1, text="One way", value=1, variable=one\_or\_round, bg="White", font=("Arial 93", 12))

r1.place(relx=0.25, y=100, anchor='center')

r2 = Radiobutton(root1, text="Round-trip", value=2, variable=one\_or\_round, bg="White", font=("Arial 93", 12))

r2.place(relx=0.75, y=100, anchor='center')

Label(root1, text="Boarding: ", font=("Arial 93", 14), fg="black", bg="White").place(relx=0.1, y=150,

anchor='w')

source = StringVar(root1)

source.set("From ") # default value

w = OptionMenu(root1, source, "delhi", "banglore", "mumbai", "hyderabad", "chennai", "kolkata", "surat")

w.config(font=('calibri', (12)), bg='white', width=22, indicatoron=0, anchor=W)

w.place(relx=0.75, y=150, anchor='center')

Label(root1, text="Destination: ", font=("Arial 93", 14), fg="black", bg="White").place(relx=0.1, y=200,

anchor='w')

destination = StringVar(root1)

destination.set("To ") # default value

w = OptionMenu(root1, destination, "delhi", "banglore", "mumbai", "hyderabad", "chennai", "kolkata", "surat")

w.config(font=('calibri', (12)), bg='white', width=22, indicatoron=0, anchor=W)

w.place(relx=0.75, y=200, anchor='center')

Label(root1, text="Date: ", font=("Arial 93", 14), fg="black", bg="White").place(relx=0.1, y=250, anchor='w')

global datee

datee = StringVar(root1)

DateEntry(root1, width=28, height=2, bg="black", fg="white", year=2020, textvariable=datee).place(relx=0.75,

y=250,

anchor='center')

Label(root1, text="Timing: ", font=("Arial 93", 14), fg="black", bg="White").place(relx=0.1, y=300, anchor='w')

timing = IntVar()

timing.set(2)

r1 = Radiobutton(root1, text="8AM to 9PM", value=1, variable=timing, bg="White", font=("Arial 93", 12))

r1.place(relx=0.60, y=300, anchor='center')

r2 = Radiobutton(root1, text="9PM to 8AM", value=2, variable=timing, bg="White", font=("Arial 93", 12))

r2.place(relx=0.85, y=300, anchor='center')

Label(root1, text="Class: ", font=("Arial 93", 14), fg="black", bg="White").place(relx=0.1, y=350, anchor='w')

passengerClass = IntVar()

Checkbutton(root1, text='Economy', variable=passengerClass, offvalue=0, onvalue=0, bg="white").place(relx=0.50,

y=350,

anchor='w')

Checkbutton(root1, text='Business', variable=passengerClass, offvalue=0, onvalue=1, bg="white").place(relx=0.65,

y=350,

anchor='w')

Checkbutton(root1, text='First Class', variable=passengerClass, offvalue=0, onvalue=2, bg="white").place(

relx=0.80, y=350, anchor='w')

Label(root1, text="Airline: ", font=("Arial 93", 14), fg="black", bg="White").place(relx=0.1, y=400, anchor='w')

img1 = PhotoImage(file='airindia.png')

img2 = PhotoImage(file='spicejet.png')

airline\_brand = IntVar()

Checkbutton(root1, text='Air India', variable=airline\_brand, offvalue=0, onvalue=1, image=img1, compound=LEFT,

bg="white").place(relx=0.5, y=400, anchor='w')

Checkbutton(root1, text='SpiceJet', variable=airline\_brand, offvalue=0, onvalue=2, image=img2, compound=LEFT,

bg="white").place(relx=0.75, y=400, anchor='w')

def onClickShowAllF2():

showAll = Tk()

showAll.title("Records")

showAll.configure(bg='grey')

showAll.geometry('600x800')

p1 = Label(showAll, text='S.No.', font='time 16 bold', fg='black', bg='grey')

p1.grid(row=1, column=0, padx=10, pady=10)

p2 = Label(showAll, text='From', font='time 16 bold', fg='black', bg='grey')

p2.grid(row=1, column=1, padx=10, pady=10)

p3 = Label(showAll, text='To', font='time 16 bold', fg='black', bg='grey')

p3.grid(row=1, column=2, padx=10, pady=10)

p4 = Label(showAll, text='Price', font='time 16 bold', fg='black', bg='grey')

p4.grid(row=1, column=3, padx=10, pady=10)

# Selecting query form database

query = 'SELECT \* FROM flight.price\_record'

cur.execute(query)

result = cur.fetchall()

num = 2

for x in result: # print all the records................................

sno = Label(showAll, text=num-1, font="time 12 bold", fg="orange", bg='black')

sno.grid(row=num, column=0, pady=10, padx=10)

city = Label(showAll, text=x[0], font="time 12 bold", fg="orange", bg='black')

city.grid(row=num, column=1, padx=10, pady=10)

nigam = Label(showAll, text=x[1], font="time 12 bold", fg="orange", bg='black')

nigam.grid(row=num, column=2, padx=10, pady=10)

arush = Label(showAll, text=x[2], font="time 12 bold", fg="orange", bg='black')

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

num += 1

def show\_result():

def one\_or\_round\_trip():

global basePrice, result

if (one\_or\_round.get() == 2):

basePrice = basePrice \* 1.8

def calculateBasePrice():

global journey, distance, basePrice

journey = source.get() + destination.get()

distance = faresDictionary[journey]

distance = int(distance)

if distance < 1200:

basePrice = distance \* 2.5

elif distance < 1700:

basePrice = distance \* 1.9

elif distance < 2500:

basePrice = distance \* 1.6

def checkDate():

global basePrice

date = datee.get()

date = datetime.strptime(date, '%m/%d/%y')

today = datetime.now()

daysDifference = (date - today).days

if daysDifference < 7:

basePrice = basePrice \* 1.6

elif daysDifference < 18:

basePrice = basePrice \* 1.4

elif daysDifference < 30:

basePrice = basePrice \* 1.2

def checktime():

global basePrice

if (timing.get() == 1):

basePrice = basePrice \* 1.13

else:

basePrice = basePrice \* 1.09

def checkPassengerClass():

global basePrice

if passengerClass.get() == 2:

basePrice = basePrice \* 2

elif passengerClass.get() == 1:

basePrice = basePrice \* 1.5

else:

basePrice = basePrice \* 1

def airline():

global basePrice

if (airline\_brand.get() == 2):

basePrice = basePrice + 800

faresDictionary = {"mumbaidelhi": "1413", "mumbaibanglore": "0981", "mumbaihyderabad": "0709",

"mumbaichennai": "1331", "mumbaikolkata": "2048", "mumbaisurat": "0283",

"delhimumbai": "1413", "delhibanglore": "2177", "delhihyderabad": "1586",

"delhichennai": "2210", "delhikolkata": "1491", "delhisurat": "1157",

"bangloremumbai": "0981", "bangloredelhi": "2177", "banglorehyderabad": "0569",

"banglorechennai": "0346", "banglorekolkata": "1878", "bangloresurat": "1252",

"hyderabadmumbai": "0709", "hyderabaddelhi": "1586", "hyderabadbanglore": "0569",

"hyderabadchennai": "0626", "hyderabadkolkata": "1492", "hyderabadsurat": "0987",

"chennaimumbai": "1331", "chennaidelhi": "2210", "chennaibanglore": "0346",

"chennaihyderabad": "0626", "chennaikolkata": "1672", "chennaisurat": "1605",

"kolkatamumbai": "2048", "kolkatadelhi": "1491", "kolkatabanglore": "1878",

"kolkatahyderabad": "1492", "kolkatachennai": "1672", "kolkatasurat": "2010",

"suratmumbai": "0283", "suratdelhi": "1157", "suratbanglore": "1252",

"surathyderabad": "0987", "suratchennai": "1605", "suratkolkata": "2010"}

try:

global basePrice

calculateBasePrice()

one\_or\_round\_trip()

checkDate()

checktime()

checkPassengerClass()

airline()

basePrice = round(basePrice, 2)

result = "Price Predicted: Rs. " + str(basePrice)

l = Label(root1, text=result, font=("Arial 93", 14), fg="black", bg="White").place(relx=0.5, y=550,

anchor='center')

except:

messagebox.showerror("Error", "Choose correct source and destination!")

finally:

query = 'INSERT INTO flight.price\_record(From1,To1,price1) values(%s, %s, %s);'

values = (source.get(), destination.get(), basePrice)

cur.execute(query, values)

con.commit()

print('sucessfully inserted')

# print(source.get(), destination.get(), basePrice)

Button(root1, text="Flight Price", font=("Arial", 16), fg="white", bg="black", compound="center", width=15,

height=1, cursor="hand2", command=show\_result).place(relx=0.50, y=480, anchor='center')

show\_data = PhotoImage(file='showdata.png')

Button(root1, text="Show Data", font=("Arial", 12), image=show\_data, fg="red", bg="white", compound="left",

cursor="hand2", command=onClickShowAllF2).place(relx=0.85, y=550, anchor='center')

root1.mainloop()

button1 = Button(root, text="Proceed to Project", command=airline\_price, compound="center", bg="black",

font=("Calibri", 20), fg="white", cursor="hand2")

button1.pack()

root.mainloop()

except Error as err:

print(err)

# Finally block for connection giving message for connection close.

finally:

if con.is\_connected():

cur.close()

con.close()

print('connection is closed')