**COMPUTER SCIENCE PROJECT**

**AZURE AVIATION**

**…Catering to your wings**



**NAME: SUNIL THUNGA**

**CLASS: XII A**

**SCHOOL: SINDHI HIGH SCHOOL**

**REGISTRATION NUMBER: 18604701**

**CERTIFICATE**

This is to certify that SUNIL THUNGA of Class 12 A of Sindhi High school Hebbal has satisfactorily completed the computer project on the topic Airline Reservation System as prescribed by CBSE for AISSCE Board practical examination. For the year 2021-22.

SIGNATURE OF EXTERNAL EXAMINER:

SIGNATURE OF INTERNAL EXAMINER:

SIGNATURE OF PRINCIPLE:

DATE OF EXAMINATION:

**ACKNOWLEDGEMENT**

It would be my utmost pleasure to express my sincere thanks to my computer teacher Mrs Girija Nagarajan, as well as our principal Mrs Maithreyi Satyadev in providing a helping hand in this project. I also take this opportunity to gratefully acknowledge my lab teacher Mr. Manjunath. Their valuable guidance, support and supervision all through this project titled ‘Azure Aviation ‘ Airlines reservation system are responsible for attaining its present form.

Secondly, I would also like to thank my parents and friends who helped me a lot in completing this project in the stipulated time frame.

**INDEX**

1. SYNOPSIS
2. FLOW CHART
3. MODULES USED
4. TABLES USED
5. SYSTEM REQUIREMENTS
6. SOURCE CODE
7. SCREENSHOTS
8. BIBLIOGRAPHY.

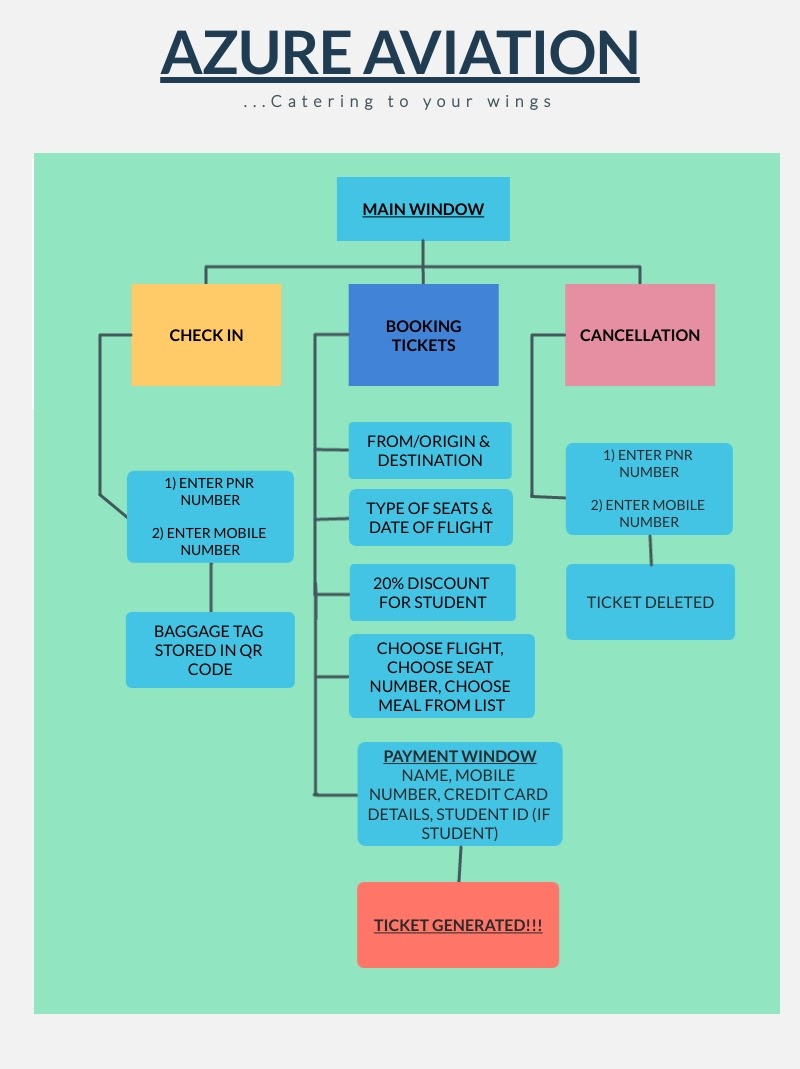
**SYNOPSIS**

Make your flying dreams come true with Azure Aviation!

Here’s presenting to you an easy and effective way of booking airline tickets! Azure Aviation allows you to book tickets, cancel tickets and also check in to your flight efficiently with utmost convenience.

You can book your ticket as per your convenience in terms of timings, dates, seating choice, and food. This System provides a 20% discount to students as well.

Thanks for flying with Azure Aviation. Happy flying!

**FLOWCHART**

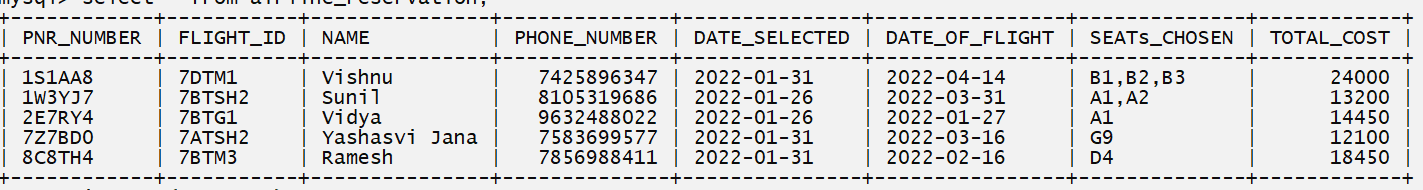
**MODULES USED**

1. Tkinter
2. MySQL connector
3. Prettytable
4. Tkcalendar
5. Pyqrcode
6. Datetime
7. Colorama
8. Random

**TABLES USED**

Database = Airline

Table = Airline\_reservation

****

**SYSTEM REQUIREMENTS**

**PROCESSOR:**

11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz

**OPERATING SYSTEM:**

Windows 11 Home Single Language

**INSTALLED RAM:**

8.00 GB

**SYSTEM TYPE:**

64-bit operating system, x64-based processor

**SOFTWARE REQUIREMENT:**

Python 3.95

MySQL Connector

**CODE**

# Importing required modules

from tkinter import \*

from tkinter import messagebox,ttk

from tkcalendar import \* # To have a graphic calendar object

from datetime import date # To use current date

from prettytable import PrettyTable # To print the overall ticket in nice format

import mysql.connector as myc

import random,colorama

from colorama import Fore # To add colors to terminal

from pyqrcode import create # To generate a qr code for baggage tag

colorama.init(autoreset=True)

# Initialising Main GUI root

root = Tk()

root.title("AZURE AVIATION!") # Adding a title to main root

root.iconbitmap('Logo.ico') # Adding an icon to root

root.geometry("1064x800") # Geometry of main Root window

canvas = Canvas(root,bg = "#ffffff",height = 800,width = 1064,bd = 0,highlightthickness = 0,relief = "ridge")

canvas.place(x = 0, y = 0)

# Initialising the mysql connection

database = myc.connect(user = 'root',password = 'tiger',host = 'localhost',database = 'airline')

mycursor = database.cursor()

SQL1 = 'CREATE TABLE IF NOT EXISTS AIRLINE\_RESERVATION(PNR\_NUMBER CHAR(6) PRIMARY KEY,FLIGHT\_ID VARCHAR(6),NAME VARCHAR(50),PHONE\_NUMBER BIGINT,DATE\_SELECTED DATE,DATE\_OF\_FLIGHT DATE,SEATs\_CHOSEN TEXT,TOTAL\_COST DOUBLE)' # TEXT is used here because we need to separate seat numbers with comma and put in mysql database

mycursor.execute(SQL1)

# Returns the flightID, date of flight and seats chosen as a dictionary in which key is a tuple and value is a string

# If 2 passengers have same flight ID and same flight date then it takes their seats and puts them to values of dictionary

def CheckAllSeatsBooked():

D = {}

SQL = 'SELECT \* FROM AIRLINE\_RESERVATION'

mycursor.execute(SQL)

result = mycursor.fetchall()

for i in result:

if (i[1],i[5]) in D:

a = ','+i[6]

D[(i[1],i[5])]+=a # If 2 people have same flight id and date of flight then it appends their seats chosen to the dicitonary

else:

T = (i[1],i[5]) # If 2 people do not have same flight id or date of flight then simply put to dictionary

D[T] = i[6]

return D

def AllBooked():

flightIds = []

D = CheckAllSeatsBooked() # Returning the dictionary with flightids and their respective seats chosen

a = list(D.items())

for i in a:

if len(i[-1]) == 278: # if all seats are chosen then the length of such a list if 278

flightIds.append(i[0][0]) # Append all the flight ids of such flights with all seats booked to flightIds

return flightIds

# Validating if PNR is not entered and if PNR is entered proceed to PNRGeneration()

def CheckPNR():

SQL = 'SELECT \* FROM AIRLINE\_RESERVATION'

mycursor.execute(SQL)

result = mycursor.fetchall()

# If no records in mysql database

if len(result) == 0:

pass

else:

for i in result:

if i[0] == PNR: # If such a PNR already exists then make a new PNR

PNRGeneration()

else:

pass

# Formatting the list containing seats so as to add ',' after each seat number and put it as text in mysql database

def format(List\_Pretty):

string = ','

return string.join(List\_Pretty)# The output will look something like 'A1,A2,A3'

# Inserting the data into mysql table

def MYSQL\_Insertion(List\_SQL\_Data):

SQL2 = 'INSERT INTO AIRLINE\_RESERVATION(PNR\_NUMBER,FLIGHT\_ID,NAME,PHONE\_NUMBER,DATE\_SELECTED,DATE\_OF\_FLIGHT,SEATs\_CHOSEN,TOTAL\_COST) VALUES (%s,%s,%s,%s,%s,%s,%s,%s)'

mycursor.execute(SQL2,List\_SQL\_Data)

database.commit()

#=====GENERATING PNR NUMBER=====#

def PNRGeneration():

global PNR

L = [chr(i) for i in range(65,91)] # Generating Aphabets from A to Z

first,second,third = random.choice(L),random.choice(L),random.choice(L) # The 3 random alphabets to go in PNR number

PNR = str(random.randint(0,9)) + first + str(random.randint(0,9)) + second + third + str(random.randint(0,9)) # Generating random PNR number having 6 places of the form A1A2A3 randint from 0-9 means 0 and 9 both included

return PNR

def FlightSeat(Selected\_FlightID):

L = []

SQL = 'SELECT \* FROM AIRLINE\_RESERVATION'

mycursor.execute(SQL)

result = mycursor.fetchall()

# Formating the time so as to come in MYSQL time format of year-month-day

def format(n):

return n.strftime("%Y-%m-%d")

# If nothing is in MYSQL database

if len(result) == 0:

return L

else:

for i in result:

if i[1] == Selected\_FlightID and format(i[-3]) == g:

if len(i[-2]) == 1:

L.append(i[-2])

elif len(i[-2]) > 1:

Lst = [item.strip() for item in i[-2].split(',')]

L.extend(Lst)

elif len(i) == 0:

pass

return L

# Obtain the seat cost according to which flight selected and if student then give discount of 20%

def seatCost(seat\_type\_cost):

global Price

if is\_Stud == 1:

if sel1.get() == 1:

Price = seat\_type\_cost[0] - seat\_type\_cost[0]\*disc

elif sel2.get() == 1:

Price = seat\_type\_cost[1] - seat\_type\_cost[1]\*disc

elif sel3.get() == 1:

Price = seat\_type\_cost[2] - seat\_type\_cost[2]\*disc

else:

if sel1.get() == 1:

Price = seat\_type\_cost[0]

elif sel2.get() == 1:

Price = seat\_type\_cost[1]

elif sel3.get() == 1:

Price = seat\_type\_cost[2]

return Price

# Generating the Final ticket

def ticket():

global l1

global Selected\_FlightID

# Obtaining the Flight ID according to what user chooses

if sel1.get() == 1:

Selected\_FlightID = FlightID[0]

elif sel2.get() == 1:

Selected\_FlightID = FlightID[1]

elif sel3.get() == 1:

Selected\_FlightID = FlightID[2]

# Checking if user wants food

if wantFood == True:

food\_type = Food\_taken

cost\_index = Food\_List.index(food\_type)

FoodCost = Food\_Cost\_List[cost\_index]

quanCost = FoodCost\*int(quan)

else:

quanCost = 0

# Checking which type of seat is chosen by user

if seat\_Chosen == 'Economy':

sC = seatCost(cost\_Economy)

elif seat\_Chosen == 'First Class':

sC = seatCost(cost\_1st\_Class)

elif seat\_Chosen == 'Business':

sC = seatCost(cost\_Business)

# Calculating The Total Cost and generating Ticket

Total\_Cost = seatno\*sC + quanCost # Getting Total cost having seat and food quantity

Selected\_Date = str(sel.get()) # Finding Date selected by user

Current\_Date = str(today) # Getting current date

prettyprint\_List = [PNRGeneration(),Selected\_FlightID, name\_entry.get(),Mobile\_entry.get(),Current\_Date,Selected\_Date,format(l1),Total\_Cost] # Data of ticket in List format

CheckPNR() # Checks if generated PNR already exists in mysql database if exists then generates new one

MYSQL\_Insertion(prettyprint\_List) # Inserts data into mysql database

table = PrettyTable(['PNR Number','Flight ID','Name','Phone Number','Date Selected ','Date of Flight','Seats Chosen','Total Cost'])

table.add\_row(prettyprint\_List) # Adding the row into pretty printed table

ticket\_Payment = Toplevel() # Creating a new window

ticket\_Payment.iconbitmap('Logo.ico')

ticket\_Payment.title('ticket')

ticket\_Lbl = Label(ticket\_Payment,text = 'Your Ticket').grid(row=0,column=0)

nametable = Label(ticket\_Payment , text = table , bg = 'white' , fg = 'black' , font = ('COURIER NEW' , 10)).grid(row=1,column=0)

Payment\_root.destroy() # Destroying the main Payment window

# Checking if the Credit Card is valid by returning True and False

def checkCCcard():

cc\_card = CreditCard\_entry.get()

if cc\_card.isdigit() == False or len(str(cc\_card)) != 16 or int(cc\_card)<=0:

return False

else:

return True

# Checking Validity of mobile number

def CheckMobile(MobileParameter):

Mobile = MobileParameter.get()

if Mobile.isdigit() == False or len(MobileParameter.get())!=10:

return False

else:

return True

# Checking if name entry is empty or if Credit Card is invalid or if Mobile is invalid

def FinalPay():

if name\_entry.get() == '' or checkCCcard() == False or CheckMobile(Mobile\_entry) == False :

messagebox.showerror('ERROR', 'Select Appropriate Data!',parent = frame1\_Payment)

else:

ticket()

# Main Payment Window

def Payment():

global frame1\_Payment

global name\_entry

global bg\_color\_payment

global Mobile\_entry

global CreditCard\_entry

global Payment\_root

Food\_Display.destroy() # Destroying the Food window

# Configuring the Payment window

bg\_color\_payment = '#00BFFF'

Payment\_root = Toplevel()

Payment\_root.iconbitmap('Logo.ico')

Payment\_root.geometry('800x600')

Payment\_root.title('PAYMENT')

Payment\_root.configure(bg = bg\_color\_payment) # Adding a color to Payment window

# Creating the payment Frame

frame1\_Payment = Frame(Payment\_root,bg = bg\_color\_payment)

frame2\_Payment = Frame(Payment\_root,bg = bg\_color\_payment)

# Creating and placing the objects in payment frame

Payment\_Display = Label(frame2\_Payment,text = 'Payment',font = ('Courier',30),bg=bg\_color\_payment).grid(row=0,column=0,padx=20)

payment\_name = Label(frame1\_Payment,text = 'Enter Name',bg = bg\_color\_payment).grid(row=0,column=0)

name\_entry = Entry(frame1\_Payment)

name\_entry.grid(row=0,column=1,pady=5,padx=20)

payment\_Mobile = Label(frame1\_Payment, text = 'Enter Mobile Number',bg = bg\_color\_payment).grid(row = 1, column=0)

Mobile\_entry = Entry(frame1\_Payment)

Mobile\_entry.grid(row=1, column=1,pady=5)

payment\_CreditCard = Label(frame1\_Payment, text = 'Enter Credit Card Number',bg = bg\_color\_payment).grid(row = 2, column=0,pady=5)

CreditCard\_entry = Entry(frame1\_Payment)

CreditCard\_entry.grid(row=2, column=1,pady=5)

# Asking for student ID if user has ticked the Are you Student check box

if is\_Stud == 1:

payment\_StudID = Label(frame1\_Payment, text = 'Enter Student ID',bg = bg\_color\_payment).grid(row = 3, column=0)

StudID\_entry = Entry(frame1\_Payment).grid(row=3, column=1,pady=5)

# Final Proceed to Payment Button

Payment\_Btn = Button(frame1\_Payment, text = 'Proceed to pay', command = FinalPay, font = ('Courier', 10)).grid(row = 4, column = 0, pady = 50, columnspan=2)

# Placing the Payment Frames

frame1\_Payment.place(relx = 0.5,rely = 0.5,anchor=CENTER)

frame2\_Payment.place(relx = 0.5,rely = 0.1,anchor=CENTER)

# Validating the quanity input given by user

def CheckQuan():

global quan

quan = quantity\_input.get()

try:

# Making sure quantity is integer greater than 0

if int(quan)>0:

return False

else:

return True

except Exception: # If any other value given the return True

return True

# Validating for cases when food or quantity is invalid or empty fields

def FinalFood():

global Food\_taken

if Food\_Combo.get() == '' or CheckQuan():

messagebox.showerror('ERROR', 'Select Appropriate Data!',parent = frame1\_Food)

else:

Food\_taken = Food\_Combo.get()

Payment()

# If Food button is unchecked

def no\_Btn\_click():

global wantFood

wantFood = False

Food\_Display.destroy()

Payment()

# If Food button is checked

def yes\_Btn\_click():

global quantity\_input

global frame1\_Food

global Food\_Combo

global wantFood

global Food\_Cost\_List

global Food\_List

wantFood = True

yes\_Btn = Button(frame1\_Food,text = 'YES',command = yes\_Btn\_click, state = DISABLED, font = ('Courier', 10)).grid(row=0,column=1)

no\_Btn = Button(frame1\_Food, text = 'NO', command = no\_Btn\_click, state = DISABLED, font = ('Courier', 10)).grid(row=0, column=2)

# List of available foods

Food\_List = ['Panner Sandwich (₹100)', 'Noodles (₹200)', 'Chicken Biryani (₹500)', 'Tacos (₹150)', 'Vada Pav (₹75)', 'Popcorn (₹100)', 'Cheese Pizza (₹250)','Pastry (₹80)','Mango Smoothi (₹50)', 'Mixed Fruit Juice (₹40)','Chocolate Mousse (₹90)']

# Respective Food Costs

Food\_Cost\_List = [100, 200, 500, 150 , 75, 100, 250, 80, 50, 40, 90]

# Adding objects to main Payment Window

Display\_food\_Lbl = Label(frame2\_Food,text = 'Choose your Food',font = ('Courier',30),bg = bg\_color\_cancel).grid(row=0,column=0,padx=20)

Choose\_Food = Label(frame1\_Food,text = 'Choose your food',bg = bg\_color\_cancel).grid(row=1,column=0,pady=5,padx=20)

Food\_Combo = ttk.Combobox(frame1\_Food,values = Food\_List)

Food\_Combo.grid(row=1,column=1,padx=20)

quantity = Label(frame1\_Food,text = 'Enter Quantity ',bg = bg\_color\_cancel).grid(row=2,column=0,padx=20)

quantity\_input = Entry(frame1\_Food,width=23)

quantity\_input.grid(row=2,column=1,pady=5,padx=20)

PrcdToPay\_Btn = Button(frame1\_Food, text = 'Proceed to pay', command = FinalFood, font = ('Courier', 10)).grid(row = 3, column = 0, pady = 50, columnspan=2)

#Creating a window for users to select Food

def Food():

global Food\_Display

global frame1\_Food

global frame2\_Food

global bg\_color\_cancel

# Configuring the main Food window

bg\_color\_cancel = '#00BFFF'

Food\_Display = Toplevel()

Food\_Display.iconbitmap('Logo.ico')

Food\_Display.geometry('800x600')

Food\_Display.title('CHOOSE FOOD')

Food\_Display.configure(bg = bg\_color\_cancel)

# Creating the Frames to hold objects in Food window

frame1\_Food = Frame(Food\_Display,bg = bg\_color\_cancel)

frame2\_Food = Frame(Food\_Display,bg = bg\_color\_cancel)

# Adding objects to the main food window

want\_food = Label(frame1\_Food,text = 'Do you want food?',bg = bg\_color\_cancel).grid(row=0,column=0,pady=5)

yes\_Btn = Button(frame1\_Food,text = 'YES',command = yes\_Btn\_click, font = ('Courier', 10)).grid(row=0,column=1,pady=5)

no\_Btn = Button(frame1\_Food, text = 'NO', command = no\_Btn\_click, font = ('Courier', 10)).grid(row=0, column=2,pady=5)

# Placing the food frames

frame1\_Food.place(relx = 0.5,rely = 0.5,anchor=CENTER)

frame2\_Food.place(relx = 0.5,rely = 0.1,anchor=CENTER)

#=====MAIN FUNCTION TO GENERATE SEATING ARRANGEMENT=====#

def Seating():

Display\_Level.destroy()

def Display\_Seating():

global seats\_Booked

if sel1.get() == 1:

Selected\_FlightID = FlightID[0]

elif sel2.get() == 1:

Selected\_FlightID = FlightID[1]

elif sel3.get() == 1:

Selected\_FlightID = FlightID[2]

seats\_Booked = FlightSeat(Selected\_FlightID)

seat\_List = [chr(i)+str(j) for i in range(65,74) for j in range(1,11)] # List containing all seat Numbers

# Displays seating and adds red NA to those seats already booked for a particular flight

def seat\_selection():

for i in range(65,74):

for j in range(1,11):

if (chr(i)+str(j)) in seats\_Booked:

print(Fore.RED+'NA',sep='',end=' ')

else:

print(chr(i),j,sep='',end=' ')

print()

seat\_selection()

global seatno

print()

def again(seatno):

global l1

l1=[]

# After user selects number of seats

for i in range(1,seatno+1):

print('CHOOSING SEAT ',i)

choice=input("ENTER THE SEAT NUMBER: ")

# Validating if the seat entered by user is already booked by checking with mysql database

if choice in seats\_Booked:

print(Fore.RED+'INVALID INPUT')

print(Fore.RED+"PLEASE ENTER ALL VALUES AGAIN")

print()

again(seatno) # Again goes to seat choosing function

else:

# If seat not in current List --> seat\_List then append it to l1

# l1 is temporary list but seat\_List is list of seats in that particular flight from mysql database

if choice in seat\_List and choice not in l1:

l1.append(choice)

else:

print()

print(Fore.RED+'INVALID INPUT')

print(Fore.RED+"PLEASE ENTER ALL VALUES AGAINs")

again(seatno)

# Checking if the number of seats entered is valid and not a string or a decimal by using try and except

def ValidSeat():

global seatno

try:

print(Fore.YELLOW+'\*'\*35)

seatno=int(input('ENTER THE NUMBER OF SEATS YOU WANT:'))

# Validating seatno

if seatno<=0:

print(Fore.RED+'Enter Appropriate Data!')

print(Fore.YELLOW+'\*'\*30)

print()

Display\_Seating()

else:

again(seatno) # Again to display choosing seat

for i in range(65,74):

for j in range(1,11):

if (chr(i)+str(j)) in l1 or (chr(i)+str(j)) in seats\_Booked: # If the seat being chosen already exists in seats\_Booked or in the temporary list l1

print(Fore.RED+'NA',end=' ')

else:

print(chr(i),j,sep='',end=' ')

print()

except Exception:

print(Fore.RED+"Invalid Seat number!")

print()

print(Fore.YELLOW+'\*'\*30)

print()

Display\_Seating()

ValidSeat()

Display\_Seating()

print()

print(Fore.YELLOW+"SELECT FOOD")

Food()

# Data Validation for Check Button

def OptError():

if sel1.get() == 1 and sel2.get() == 1 and sel3.get() == 1:

messagebox.showerror('ERROR', 'Select Appropriate Flight!',parent = frame1\_Display\_Flight)

elif sel1.get() == 1 and sel2.get() == 1 and sel3.get() == 0:

messagebox.showerror('ERROR', 'Select Appropriate Flight!',parent = frame1\_Display\_Flight)

elif sel1.get() == 1 and sel2.get() == 0 and sel3.get() == 1:

messagebox.showerror('ERROR', 'Select Appropriate Flight!',parent = frame1\_Display\_Flight)

elif sel1.get() == 0 and sel2.get() == 1 and sel3.get() == 1:

messagebox.showerror('ERROR', 'Select Appropriate Flight!',parent = frame1\_Display\_Flight)

elif sel1.get() == 0 and sel2.get() == 0 and sel3.get() == 0:

messagebox.showerror('ERROR', 'Select Appropriate Flight!',parent = frame1\_Display\_Flight)

if sel1.get() == 1 and sel2.get() == 0 and sel3.get() == 0:

Seating()

if sel1.get() == 0 and sel2.get() == 1 and sel3.get() == 0:

Seating()

if sel1.get() == 0 and sel2.get() == 0 and sel3.get() == 1:

Seating()

# Generating Flights-IDs, Flight-Time

def From\_To\_Places(From\_Entry,To\_Entry,flight\_IDs):

global sel1,sel2,sel3

global opt\_box\_1,opt\_box\_2,opt\_box\_3

global is\_Stud

global FlightID

global cost\_Business,cost\_1st\_Class,cost\_Economy

global disc

FlightID = flight\_IDs

def Display(cost):

flight\_1 = Label(frame1\_Display\_Flight,text = "Flight 1 (Connect)",font = ("Arial",12), bg = bg\_color).grid(row=0,column=0,padx=30,pady=5)

flight\_2 = Label(frame1\_Display\_Flight,text = "Flight 2 (Non Stop)",font = ("Arial",12), bg = bg\_color).grid(row=1,column=0,padx=30,pady=5)

flight\_3 = Label(frame1\_Display\_Flight,text = "Flight 3 (Non Stop)",font = ("Arial",12), bg = bg\_color).grid(row=2,column=0,padx=30,pady=5)

flight\_id\_1 = Label(frame1\_Display\_Flight,text = flight\_IDs[0],font = ("Arial",12), bg = bg\_color).grid(row=0,column=1,padx=30,pady=5)

flight\_id\_2 = Label(frame1\_Display\_Flight,text = flight\_IDs[1],font = ("Arial",12), bg = bg\_color).grid(row=1,column=1,padx=30,pady=5)

flight\_id\_3 = Label(frame1\_Display\_Flight,text = flight\_IDs[2],font = ("Arial",12), bg = bg\_color).grid(row=2,column=1,padx=30,pady=5)

flight\_time\_1 = Label(frame1\_Display\_Flight,text = Flight\_Times[0],font = ("Arial",12), bg = bg\_color).grid(row=0,column=2,padx=30)

flight\_time\_2 = Label(frame1\_Display\_Flight,text = Flight\_Times[1],font = ("Arial",12),bg = bg\_color).grid(row=1,column=2,padx=30)

flight\_time\_3 = Label(frame1\_Display\_Flight,text = Flight\_Times[2],font = ("Arial",12),bg = bg\_color).grid(row=2,column=2,padx=30)

# If Not a student then no 20% discount

if is\_Stud == 0:

cost\_1 = Label(frame1\_Display\_Flight,text = '₹'+str(cost[0]),font = ("Arial",12),bg = bg\_color).grid(row=0,column=3,padx=30)

cost\_2 = Label(frame1\_Display\_Flight,text = '₹'+str(cost[2]),font = ("Arial",12),bg = bg\_color).grid(row=1,column=3,padx=30)

cost\_3 = Label(frame1\_Display\_Flight,text = '₹'+str(cost[1]),font = ("Arial",12),bg = bg\_color).grid(row=2,column=3,padx=30)

# If student then giving 20% discount

elif is\_Stud == 1:

cost\_1 = Label(frame1\_Display\_Flight,text = '₹'+str(cost[0]-cost[0]\*disc),font = ("Arial",12),bg = bg\_color).grid(row=0,column=3,padx=30)

cost\_2 = Label(frame1\_Display\_Flight,text = '₹'+str(cost[2]-cost[2]\*disc),font = ("Arial",12),bg = bg\_color).grid(row=1,column=3,padx=30)

cost\_3 = Label(frame1\_Display\_Flight,text = '₹'+str(cost[1]-cost[1]\*disc),font = ("Arial",12),bg = bg\_color).grid(row=2,column=3,padx=30)

opt\_box\_1 = Checkbutton(frame1\_Display\_Flight,variable=sel1,onvalue=1,offvalue=0,bg = bg\_color).grid(row=0,column=4)

opt\_box\_2 = Checkbutton(frame1\_Display\_Flight,variable=sel2,onvalue=1,offvalue=0,bg = bg\_color).grid(row=1,column=4)

opt\_box\_3 = Checkbutton(frame1\_Display\_Flight,variable=sel3,onvalue=1,offvalue=0,bg = bg\_color).grid(row=2,column=4)

D = AllBooked() # Getting the list with only flight ids where all seats are booked

if len(D) == 0: # If such length is 0 then dont do anything

pass

else:

for i in D:

if i not in FlightID: # If the flight id in allBooked is not in the selectedFlight id

pass

else:

index = FlightID.index(i) # Getting index of the flight id from its list

if index == 0:

opt\_box\_1 = Checkbutton(frame1\_Display\_Flight,variable=sel1,onvalue=1,offvalue=0,bg = bg\_color,state = DISABLED).grid(row=0,column=4)

booked\_lbl = Label(frame1\_Display\_Flight,text = "(All seats Booked!)",foreground='red').grid(row=0,column=5)

if index == 1:

opt\_box\_2 = Checkbutton(frame1\_Display\_Flight,variable=sel2,onvalue=1,offvalue=0,bg = bg\_color,state = DISABLED).grid(row=1,column=4)

booked\_lbl = Label(frame1\_Display\_Flight,text = "(All seats Booked!)",foreground='red').grid(row=1,column=5)

if index == 2:

opt\_box\_3 = Checkbutton(frame1\_Display\_Flight,variable=sel3,onvalue=1,offvalue=0,bg = bg\_color,state = DISABLED).grid(row=2,column=4)

booked\_lbl = Label(frame1\_Display\_Flight,text="(All seats Booked!)",foreground='red').grid(row=2,column=5)

else:

pass

display\_Btn = Button(frame1\_Display\_Flight,text = 'CONFIRM',command = OptError, font = ('Courier', 10)).grid(row=4,column=1,padx=20,ipadx=30,pady=(40,0),columnspan=2)

From = From\_Entry

To = To\_Entry

is\_Stud = var1.get()

sel1 = IntVar() # 0 if button not clicked, 1 if button clicked

sel2 = IntVar() # 0 if button not clicked, 1 if button clicked

sel3 = IntVar() # 0 if button not clicked, 1 if button clicked

disc = 0.2 # Discount offered by Azure Aviation for students

Flight\_Times = ['6:00 AM','2:00 PM','8:00 PM'] # Flight times

# Seat costs for different types

cost\_Economy = [6000.0,8000.0,10000.0]

cost\_Business = [10000.0,12000.0,14000.0]

cost\_1st\_Class = [14000.0,16000.0,18000.0]

if seat\_Chosen == 'Economy':

Display(cost\_Economy)

elif seat\_Chosen == 'First Class':

Display(cost\_1st\_Class)

elif seat\_Chosen == 'Business':

Display(cost\_Business)

# Main Search Flight window

def Search\_Flight():

global frame1\_Display\_Flight

global frame2\_Display\_Flight

global Flight\_Ids

global bg\_color

global Display\_Level

global Flight\_ID\_NestedList

global booked

# Configuring the Search Flight window

Top.destroy()

Display\_Level = Toplevel()

Display\_Level.iconbitmap('Logo.ico')

bg\_color = '#00BFFF'

frame1\_Display\_Flight = Frame(Display\_Level,bg = bg\_color)

frame2\_Display\_Flight = Frame(Display\_Level,bg = bg\_color)

# Creating the search flight window

Display\_Level.geometry('850x600')

Display\_Level.title('Display Flights')

Display\_Level.configure(bg = bg\_color)

# Creating main objects on search flight window

Display\_flight\_Lbl = Label(frame2\_Display\_Flight,text = 'Displaying Flights...',font = ('Courier',30),bg = bg\_color).grid(row=0,column=0)

SubHeading\_Lbl = Label(frame2\_Display\_Flight,text = '(Flights, Flight ID, Departure Time, Price)',font = ('Arial',15), bg = bg\_color).grid(row=1,column=0,pady=(0,40))

#=====FLIGHT IDs=====#

Flight\_ID\_NestedList = [['7BTD1','7BTD2','7BTD3'],['7BTM1','7BTM2','7BTM3'],['7BTA1','7BTA2','7BTA3'],['7BTSH1','7BTSH2','7BTSH3'],['7BTG1','7BTG2','7BTG3'],['7DTB1','7DTB2','7DTB3'],['7DTM1','7DTM2','7DTM3'],['7DTA1','7DTA2','7DTA3'],['7DTSH1','7DTSH2','7DTSH3'],['7DTG1','7DTG2','7DTG3'],['7MTB1','7MTB2','7MTB3'],['7MTD1','7MTD2','7MTD3'],['7MTA1','7MTA2','7MTA3'],['7MTSH1','7MTSH2','7MTSH3'],['7MTG1','7MTG2','7MTG3'],['7ATSH1','7ATSH2','7ATSH3'],['7ATD1','7ATD2','7ATD3'],['7ATB1','7ATB2','7ATB3'],['7ATM1','7ATM2','7ATM3'],['7ATG1','7ATG2','7ATG3'],['7SHTD1','7SHTD2','7SHTD3'],['7SHTB1','7SHTB2','7SHTB3'],['7SHTA1','7SHTA2','7SHTA3'],['7SHTM1','7SHTM2','7SHTM3'],['7SHTG1','7SHTG2','7SHTG3'],['7GTB1','7GTB2','7GTB3'],['7GTD1','7GTD2','7GTD3'],['7GTM1','7GTM2','7GTM3'],['7GTA1','7GTA2','7GTA3'],['7GTSH1','7GTSH2','7GTSH3']]

# Cases to check From and To Location from users choice and pass to function From\_To\_Places along with its Flight ID

if From\_Entry == 'Bengaluru' and To\_Entry == 'Delhi':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[0])

elif From\_Entry == 'Bengaluru' and To\_Entry == 'Mumbai':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[1])

elif From\_Entry == 'Bengaluru' and To\_Entry == 'Ahmedabad':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[2])

elif From\_Entry == 'Bengaluru' and To\_Entry == 'Shillong':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[3])

elif From\_Entry == 'Bengaluru' and To\_Entry == 'Goa':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[4])

elif From\_Entry == 'Delhi' and To\_Entry == 'Bengaluru':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[5])

elif From\_Entry == 'Delhi' and To\_Entry == 'Mumbai':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[6])

elif From\_Entry == 'Delhi' and To\_Entry == 'Ahmedabad':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[7])

elif From\_Entry == 'Delhi' and To\_Entry == 'Shillong':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[8])

elif From\_Entry == 'Delhi' and To\_Entry == 'Goa':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[9])

elif From\_Entry == 'Mumbai' and To\_Entry == 'Bengaluru':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[10])

elif From\_Entry == 'Mumbai' and To\_Entry == 'Delhi':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[11])

elif From\_Entry == 'Mumbai' and To\_Entry == 'Ahmedabad':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[12])

elif From\_Entry == 'Mumbai' and To\_Entry == 'Shillong':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[13])

elif From\_Entry == 'Mumbai' and To\_Entry == 'Goa':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[14])

elif From\_Entry == 'Ahmedabad' and To\_Entry == 'Shillong':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[15])

elif From\_Entry == 'Ahmedabad' and To\_Entry == 'Delhi':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[16])

elif From\_Entry == 'Ahmedabad' and To\_Entry == 'Bengaluru':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[17])

elif From\_Entry == 'Ahmedabad' and To\_Entry == 'Mumbai':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[18])

elif From\_Entry == 'Ahmedabad' and To\_Entry == 'Goa':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[19])

elif From\_Entry == 'Shillong' and To\_Entry == 'Delhi':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[20])

elif From\_Entry == 'Shillong' and To\_Entry == 'Bengaluru':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[21])

elif From\_Entry == 'Shillong' and To\_Entry == 'Ahmedabad':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[22])

elif From\_Entry == 'Shillong' and To\_Entry == 'Mumbai':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[23])

elif From\_Entry == 'Shillong' and To\_Entry == 'Goa':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[24])

elif From\_Entry == 'Goa' and To\_Entry == 'Bengaluru':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[25])

elif From\_Entry == 'Goa' and To\_Entry == 'Delhi':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[26])

elif From\_Entry == 'Goa' and To\_Entry == 'Mumbai':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[27])

elif From\_Entry == 'Goa' and To\_Entry == 'Ahmedabad':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[28])

elif From\_Entry == 'Goa' and To\_Entry == 'Shillong':

From\_To\_Places(From\_Entry,To\_Entry,Flight\_ID\_NestedList[29])

booked = Flight\_ID\_NestedList[29]

# Placing the Display Flight Frames

frame1\_Display\_Flight.place(relx = 0.5,rely = 0.5,anchor=CENTER)

frame2\_Display\_Flight.place(relx = 0.5,rely = 0.1,anchor=CENTER)

# Check Button for availing for Student discount

def Check\_Btn():

global var1

var1 = IntVar()

student\_chk = Checkbutton(frame1, text ="Are You a student?", variable = var1,onvalue=1,offvalue=0, bg = bgcolor,font=('Arial', 12))

student\_chk.deselect()

student\_chk.grid(row=4,column=0,ipadx=50,pady=(30,0),columnspan=3)

Submit\_Btn = Button(frame1,text = 'Search Flight', command=Search\_Flight, font = ('Courier', 10)).grid(row=5,column=0,ipadx=50,pady=30,columnspan=3)

# Building A Date Picker calendar

def calendar():

global sel

def submit():

global string

global g

g = sel.get() # Getting Date entered by User

string = str(g)

crnt\_date = str(today) # Getting current Date

year = int(string[2:4])

if g<=str(crnt\_date) or year!=22: # Validating the Date

messagebox.showerror("ERROR",'Choose Correct Date',parent = frame1)

else:

Check\_Btn()

sel = StringVar() # Holder to hold the date selected

Date\_Lbl = Label(frame1,text = 'Departure Date',font = ("Arial",15),bg = bgcolor).grid(row=3,column=0)

Date\_Entry = DateEntry(frame1, selectmode = 'day', year = 2022, month = 2, day=1,date\_pattern='yyyy-mm-dd', textvariable = sel,width=20) # Tkinter calendar to select the date of flight

Date\_Entry.grid(row = 3, column=1)

global today

today = date.today() # Obtaining todays date

cal\_Submit = Button(frame1,text = 'SUBMIT',command=submit, font = ('Courier', 10)).grid(row=3,column=2, pady = 5)

# Checking is no seats chosen else run calendar function

def seatSubmit():

global seat\_Chosen

seat\_Chosen = seat\_type.get()

if seat\_Chosen == '':

messagebox.showerror("ERROR",'Choose Appropriate Data',parent = frame1)

else:

calendar()

# Function to be called after user selects destination Flight

def To():

global seat\_type

global To\_Entry

seat = StringVar()

To\_Entry = To\_Places.get()

if To\_Entry == '': # Checking if To Entry combobox is emppty

messagebox.showerror("ERROR", "Please Fill the Field!",parent = frame1)

else:

seat\_types = ['First Class','Business','Economy']

pass\_lbl = Label(frame1, text = "Seat Type",bg = bgcolor,font = ("Arial",15)).grid(row=2,column=0)

seat\_type = ttk.Combobox(frame1,values = seat\_types)

seat\_type.grid(row=2,column=1)

pass\_entry\_submit = Button(frame1,text = 'SUBMIT',command=seatSubmit, font = ('Courier', 10)).grid(row=2,column=2,padx=20,pady=5)

# Function to the button SUBMIT next to From places

def From():

global To\_Places

global From\_Entry

L = ['Bengaluru', 'Delhi','Mumbai','Ahmedabad', 'Shillong', 'Goa']

From\_Entry = From\_places.get() # Fetching input from From\_places box

# Checking for validity

if From\_Entry == '':

messagebox.showerror("ERROR", "Please Fill the Field!",parent = frame1)

else:

L.remove(From\_Entry) # If the From place exists in To places remove it

To\_Lbl = Label(frame1,text = "To",bg=bgcolor,font = ("Arial",15)).grid(row=1,column=0)

To\_Places = ttk.Combobox(frame1,values = L)

To\_Places.grid(row = 1, column=1)

to\_places\_submit = Button(frame1,text = 'SUBMIT',command = To, font = ('Courier', 10)).grid(row=1,column=2,padx=20,pady=5)

#=====BOOKING WINDOW=====#

def Booking():

global bgcolor

global frame1

global From\_places

global Top

L = ['Bengaluru', 'Delhi','Mumbai','Ahmedabad', 'Shillong', 'Goa']

bgcolor = '#00BFFF'

Top = Toplevel()

Top.iconbitmap('Logo.ico')

Top.geometry('800x600')

Top.title('Booking Ticket')

Top.configure(bg = bgcolor)

# Creating Booking Frames

frame1 = Frame(Top,bg = bgcolor)

frame2 = Frame(Top,bg = bgcolor)

# Creating and placing the Objects on the booking Frame

Booking\_Lbl = Label(frame2,text = 'BOOKING TICKET',bg = bgcolor ,font = ('Courier',30) ).grid(row=0,column=0)

From\_lbl = Label(frame1, text = "From",bg =bgcolor,font = ("Arial",15)).grid(row=0,column=0)

From\_places = ttk.Combobox(frame1,values=L)

From\_places.grid(row=0,column=1)

from\_places\_submit = Button(frame1,text = 'SUBMIT',command=From, font = ('Courier', 10)).grid(row=0,column=2,padx=20,pady=5)

space\_label = Label(frame1,text = ' '\*50,bg = bgcolor).grid(row=3,column=0)

# Placing the Frames

frame2.place(relx = 0.5,rely = 0.1,anchor=CENTER)

frame1.place(relx=0.5,rely=0.5,anchor=CENTER)

#=====CANCELLATION - DATA VALIDATION=====#

def CancelTicket():

string\_PNR = str(PNR\_entry\_no)

# Displaying the details when ticket is deleted

SQL = "DELETE FROM AIRLINE\_RESERVATION WHERE PNR\_NUMBER = %s"

statement = f'''

Ticket Deleted!

PNR Number : {record[0][0]}

Passenger Name : {record[0][2]}

Flight Number : {record[0][1]}

Mobile Number : {record[0][3]}

'''

mycursor.execute(SQL,(string\_PNR,))

database.commit()

messagebox.showinfo("DELETED!",statement,parent = frame1\_cancellation)

# Checking if the Phone number entered really exists for that particular PNR no by comparing with mysql database

def Checker(result,PNR\_entry\_no):

global record

temp = 0

record = []

for i in result:

if PNR\_entry\_no == i[0] and int(Phone.get())==i[3]:

record.append(i) # Appending record with details where PNR\_entry\_no and Phone.get() matches with record in mysql

temp = 1 # Temporary variable to hold 1

if temp == 0:

messagebox.showerror("ERROR", "Ticket Does not Exist!",parent = frame1\_cancellation) # Parent makes the window pops on the same frame

# If temp !=0

else:

CancelTicket()

# Checking if the ticket entered really exists by comparing with mysql database

def TicketExists():

global PNR\_entry\_no

PNR\_entry\_no = PNRno.get()

SQL = 'SELECT \* FROM AIRLINE\_RESERVATION'

mycursor.execute(SQL)

result = mycursor.fetchall()

# If no records in mysql table

if len(result) == 0:

messagebox.showerror("ERROR", "Ticket does not exist!",parent = frame1\_cancellation)

else:

Checker(result,PNR\_entry\_no)

# Cancellation of Record

def CancelDetails():

# If phone number if invalid

if CheckMobile(Phone) == False:

messagebox.showerror("ERROR", "INVALID ENTRIES!",parent = frame1\_cancellation)

else:

TicketExists()

#=================CANCELLATION WINDOW==============#

def Cancellation():

global Phone

global frame1\_cancellation

global PNRno

bg\_color\_cancel = '#00BFFF'

Display = Toplevel()

Display.iconbitmap('Logo.ico')

Display.geometry('800x600')

Display.title('CANCELLATION')

Display.configure(bg = bg\_color\_cancel)

# Creating Frames for cancellation

frame1\_cancellation = Frame(Display,bg = bg\_color\_cancel)

frame2\_cancellation = Frame(Display,bg = bg\_color\_cancel)

# Creating and placing the window objects

Cancel\_Lbl = Label(frame2\_cancellation,text = 'Cancellation',bg = bg\_color\_cancel ,font = ('Courier',30)).grid(row=0,column=0)

PNR\_lbl = Label(frame1\_cancellation, text = 'Enter PNR Number', bg = bg\_color\_cancel, font = ('Arial',12)).grid(row = 0,column=0,pady=5, padx =20)

PNRno = Entry(frame1\_cancellation, width = 20)

PNRno.grid(row=0,column=1)

Phone\_lbl = Label(frame1\_cancellation, text = 'Enter Phone No', bg = bg\_color\_cancel,font = ('Arial',12)).grid(row = 2,column=0,pady=5)

Phone = Entry(frame1\_cancellation, width = 20)

Phone.grid(row=2,column=1)

Final\_Btn = Button(frame1\_cancellation,text = 'Confirm',command = CancelDetails, font = ('Courier', 10)).grid(row=3,column=0,pady = 20, ipadx = 20, columnspan=2)

# Placing the frames

frame1\_cancellation.place(relx = 0.5,rely = 0.5,anchor=CENTER)

frame2\_cancellation.place(relx = 0.5,rely = 0.1,anchor=CENTER)

#===================BAGGAGE TAG====================#

def baggage\_in\_QR():

# Final printing when QR code is scanned

s = f'''

\t\t\t\tBAGGAGE TAG

\t\t\t\tAzure Aviation Happy Flying!

\t\t\t\t25Kg Luggage Permitted

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PNR Number : {Check\_in\_record[0][0]}

Passenger Name : {Check\_in\_record[0][2]}

Flight Number : {Check\_in\_record[0][1]}

Seat(s) Chosen: {Check\_in\_record[0][6]}

Date of Flight: {Check\_in\_record[0][5]}

Mobile Number : {Check\_in\_record[0][3]}

'''

return s # Returning the formatted string

#=====BAGGAGE TAG WINDOW=====#

def Show\_baggage\_tag(PNR\_entry\_no):

global frame1\_Check\_in

bg\_color\_baggage\_tag = '#00BFFF'

Display = Toplevel()

Display.iconbitmap('Logo.ico')

Display.geometry('800x600')

Display.title('Baggage Tag')

Display.configure(bg = bg\_color\_baggage\_tag)

# Creating the window Frames

frame1\_baggage\_tag = Frame(Display,bg = bg\_color\_baggage\_tag)

frame2\_baggage\_tag = Frame(Display,bg = bg\_color\_baggage\_tag)

# Baggage tag window objects

Baggage\_tag\_heading = Label(frame2\_baggage\_tag, text = 'Baggage Tag', bg = bg\_color\_baggage\_tag, font = ('Courier',30)).grid(row = 0, column=0)

Baggage\_tag\_SubHeading = Label(frame2\_baggage\_tag, text = 'Your Baggage Tag Has Been Successfully Generated ✅ \n Scan to Print Baggage Tag!', bg = bg\_color\_baggage\_tag,font = ('Courier',15)).grid(row = 1, column=0)

image\_view = Label(frame1\_baggage\_tag)

image\_view.grid(row=0,column=1)

#=====QR CODE=====#

QR\_Data = baggage\_in\_QR()

img = create(QR\_Data) # Creating the QR code from QR\_Data

test = img.xbm(scale = 2) # Scaling the image to 2

global xbm\_image

xbm\_image = BitmapImage(data=test,foreground='Blue',background='yellow')# Creating a BitmapImage out of the QR code

image\_view.config(image = xbm\_image) # Configuring the Label with the QR code

# Packing the Frames

frame1\_baggage\_tag.place(relx = 0.5,rely = 0.5,anchor=CENTER)

frame2\_baggage\_tag.place(relx = 0.5,rely = 0.1,anchor=CENTER)

# Checking validity of PNR and Phone entered with mysql database

def Checker\_baggage\_tag(result,PNR\_entry\_no):

global Check\_in\_record

Check\_in\_record = []

temp = 0

for i in result:

if PNR\_entry\_no == i[0] and int(Phone.get())==i[3]: # If PNR\_entry\_no == i[0] and phone.get() then append to Check\_in\_record

Check\_in\_record.append(i)

temp = 1

if temp == 0:

messagebox.showerror("ERROR", "Ticket Does not Exist!",parent = frame1\_Check\_in)

else:

Show\_baggage\_tag(PNR\_entry\_no)

# Checking validity of PNR number by connecting to mysql

def Ticket\_baggage():

global result

global PNR\_entry\_no

PNR\_entry\_no = PNRno.get()

SQL = 'SELECT \* FROM AIRLINE\_RESERVATION'

mycursor.execute(SQL)

result = mycursor.fetchall()

# If no records in MYSQL database

if len(result) == 0:

messagebox.showerror("ERROR","INVALID ENTRIES!",parent = frame1\_Check\_in)

else:

Checker\_baggage\_tag(result,PNR\_entry\_no)

# Checking if Entries are valid

def Baggage\_tag():

global PNR\_entry\_no

PNR\_entry\_no = PNRno.get()

# Checking if PNR entry is empty or is Phone entry is invalid or if Phone entered is not valid

if PNRno.get() == '' or Phone.get() == '':

messagebox.showerror("ERROR", "Please Fill the Field",parent = frame1\_Check\_in)

elif Phone.get().isalpha():

messagebox.showerror("ERROR", "Please Fill appropriately",parent = frame1\_Check\_in)

else:

Ticket\_baggage()

#=====CHECK IN WINDOW=====#

def Check\_in():

global frame1\_Check\_in

global PNRno

global Phone

# Configuring the Check in Window

bg\_color\_check\_in = '#00BFFF'

Display = Toplevel()

Display.iconbitmap('Logo.ico')

Display.geometry('800x600')

Display.title('Check in')

Display.configure(bg = bg\_color\_check\_in)

# Creating Check in Frames

frame1\_Check\_in = Frame(Display,bg = bg\_color\_check\_in)

frame2\_Check\_in = Frame(Display,bg = bg\_color\_check\_in)

# Creating and placing objects in Check in window

Checkin\_Lbl = Label(frame2\_Check\_in,text = 'CHECKING IN',bg = bg\_color\_check\_in ,font = ('Courier',30)).grid(row=0,column=0)

PNR\_lbl = Label(frame1\_Check\_in, text = 'Enter PNR number', bg = bg\_color\_check\_in,font = ('Arial',12)).grid(row = 0,column=0,pady=5, padx = 20)

PNRno = Entry(frame1\_Check\_in, width = 20)

PNRno.grid(row=0,column=1)

Phone\_lbl = Label(frame1\_Check\_in, text = 'Enter Phone no', bg = bg\_color\_check\_in,font = ('Arial',12)).grid(row = 2,column=0,pady=10)

Phone = Entry(frame1\_Check\_in, width = 20)

Phone.grid(row=2,column=1)

Final\_Btn = Button(frame1\_Check\_in,text = 'Confirm', font = ('Courier', 10),padx=3, pady = 1, command = Baggage\_tag).grid(row=3,column=0,pady = 15, ipadx = 15, columnspan=2)

# Placing the Check in Frames

frame1\_Check\_in.place(relx = 0.5,rely = 0.5,anchor=CENTER)

frame2\_Check\_in.place(relx = 0.5,rely = 0.1,anchor=CENTER)

#=============MAIN WINDOW OBJECTS=============#

check\_in\_pic = PhotoImage(file = f"Check\_in.png")

Checkin\_button = Button(image = check\_in\_pic,borderwidth = 0,highlightthickness = 0,relief = "flat",command=Check\_in)

Checkin\_button.place(x = 671, y = 556,width = 294,height = 65)

img1 = PhotoImage(file = f"cancel.png")

Cancellation\_button = Button(image = img1,borderwidth = 0,highlightthickness = 0,relief = "flat",command=Cancellation)

Cancellation\_button.place(x = 671, y = 442,width = 294,height = 69)

img2 = PhotoImage(file = f"booking.png")

Booking\_button = Button(image = img2,borderwidth = 0,highlightthickness = 0,relief = "flat",command=Booking)

Booking\_button.place(x = 671, y = 334,width = 293,height = 65)

background\_img = PhotoImage(file = f"background.png")

background = canvas.create\_image(529.0, 400.0,image=background\_img)

# Making the Main window non-resizable

root.resizable(False, False)

# Running the mainloop application

root.mainloop()

'''

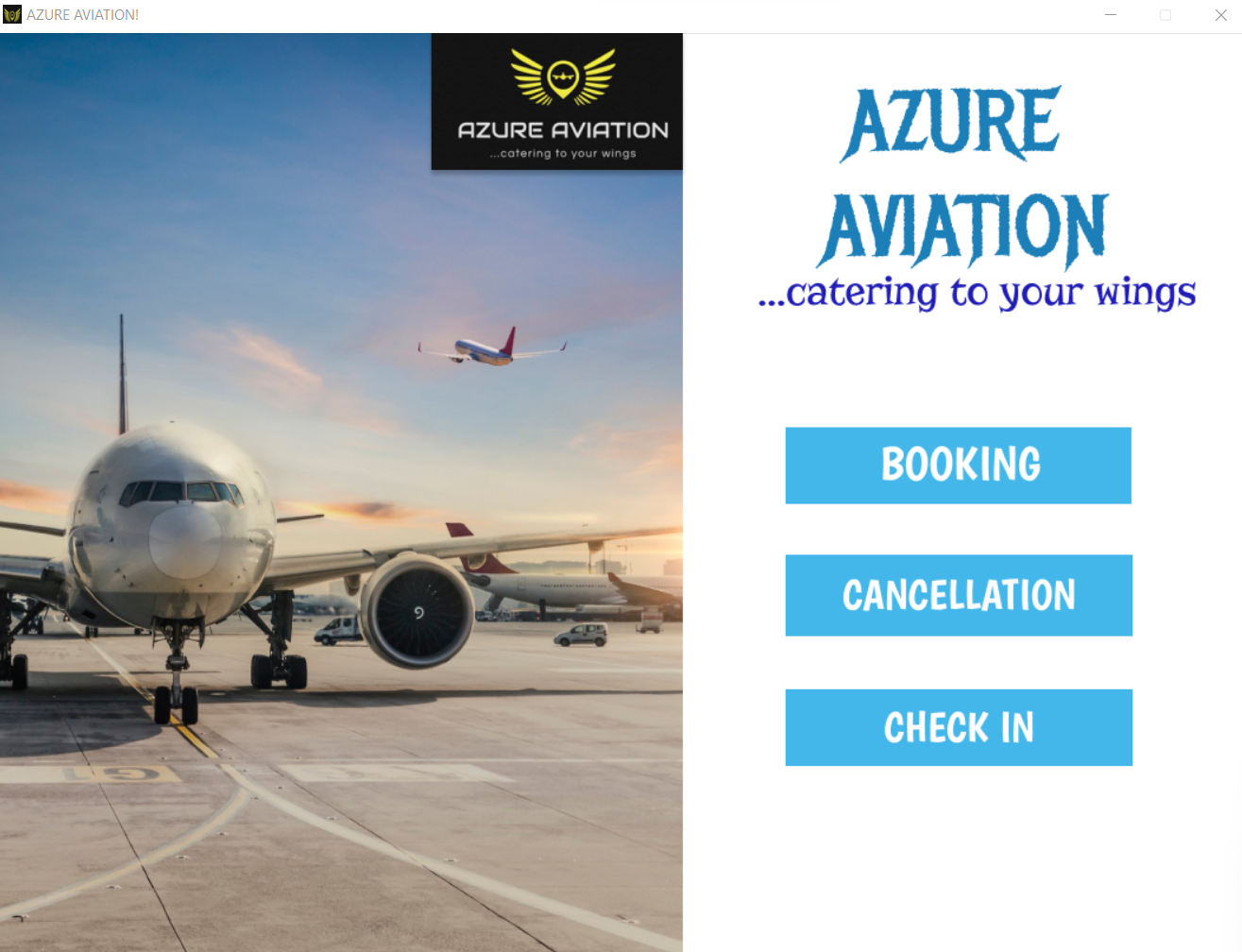
To View QR code on PC

QR SCANNER: https://www.imgonline.com.ua/eng/scan-qr-bar-code.php

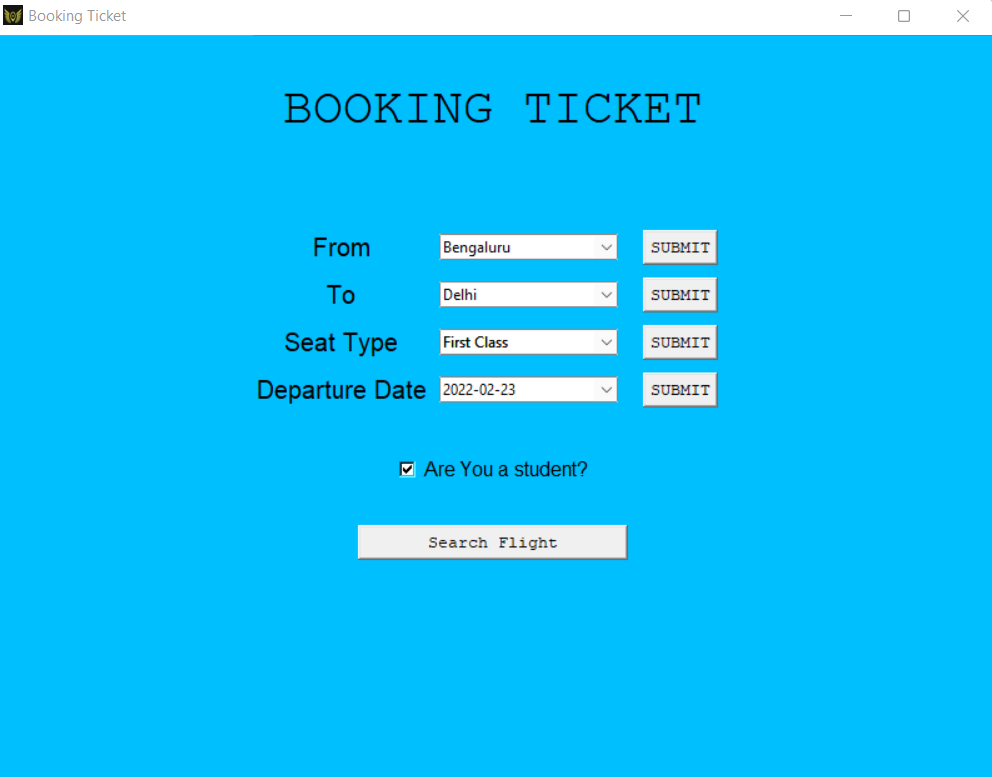
'''

**PROJECT SCREENSHOTS**

**MAIN WINDOW:**



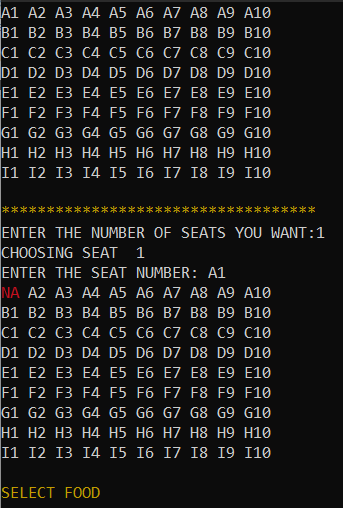
**BOOKING TICKETS:**



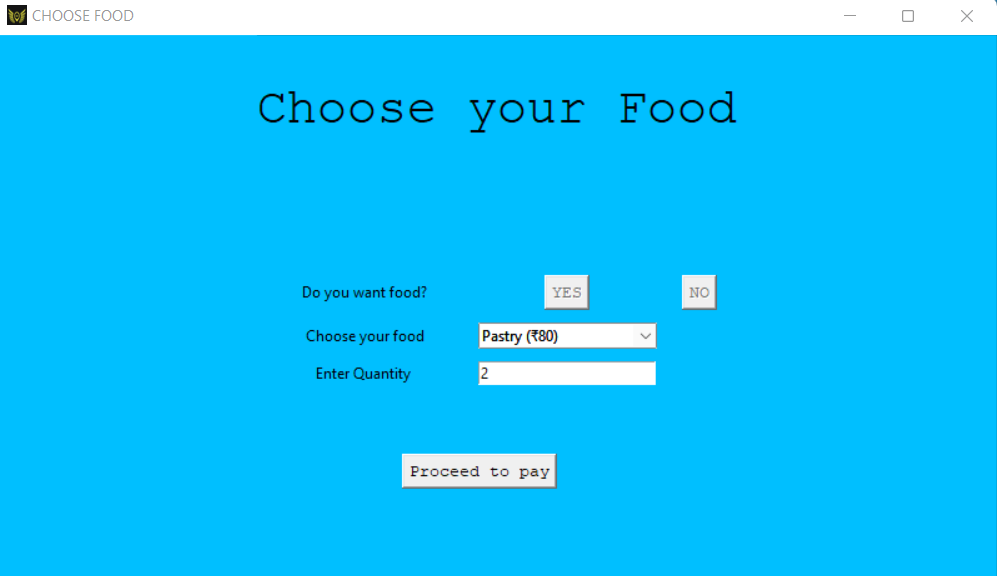
**DISPLAYING TICKETS:**



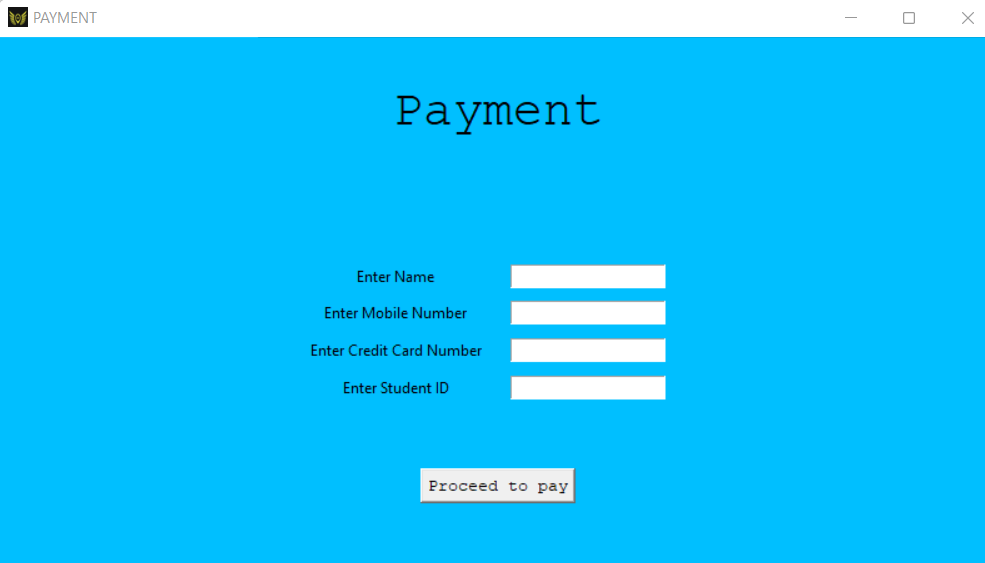
**SEATING:**



**CHOOSING FOOD:**

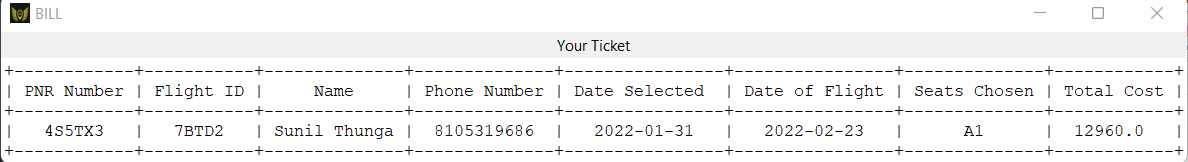


**PAYMENT WINDOW:**

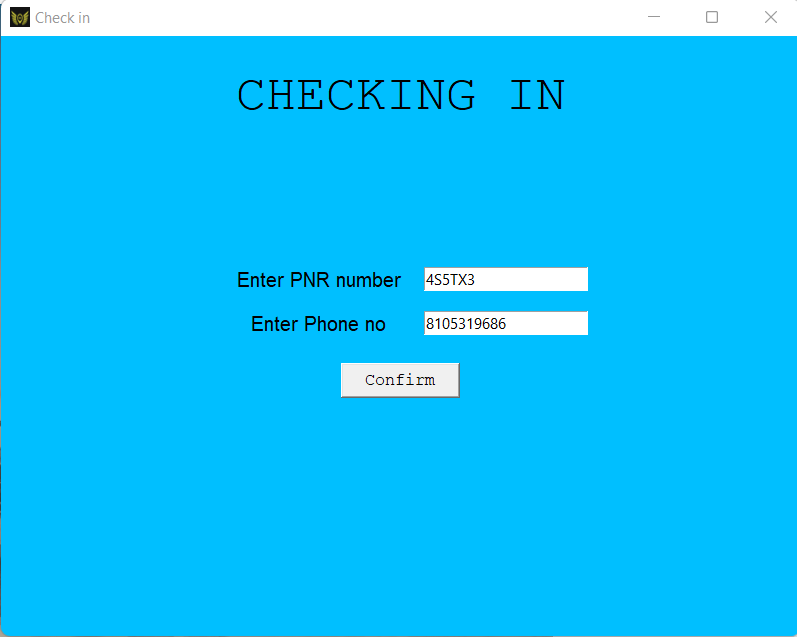


(DETAILS TO BE FILLED IN THE RESPECTIVE REGION )

**TICKET DISPLAY:**



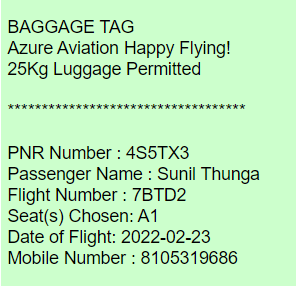
**CHECKING IN TO FLIGHT:**



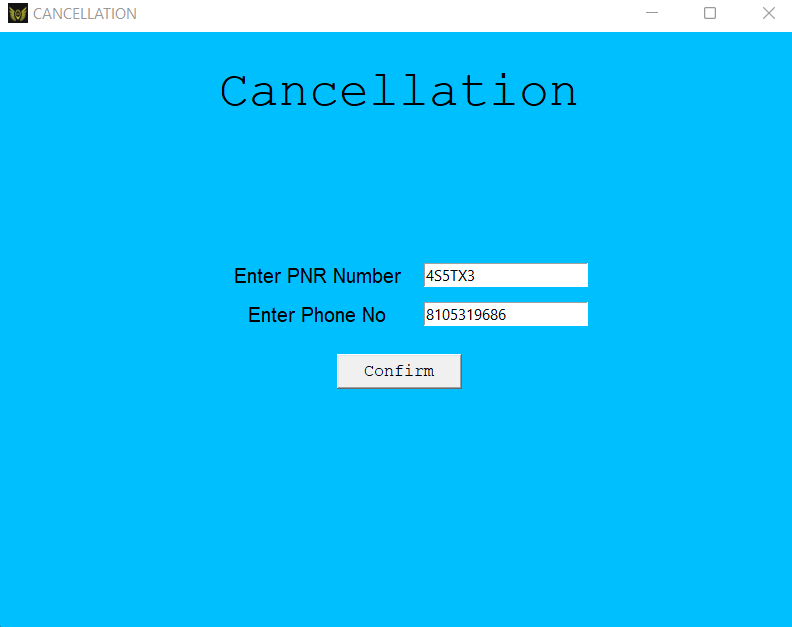
**DISPLAYING BAGGAGE TAG:**



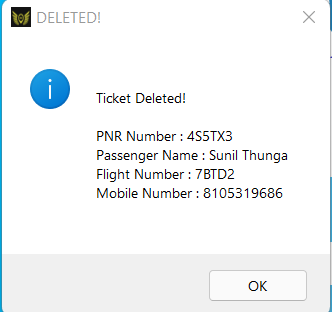
**DISPLAYING QR CODE SCAN:**



**CANCELLING TICKETS:**



**DELETION OF TICKET:**



**FULL FLIGHT BOOKED:**



**BIBLIOGRAPHY**

1. <https://www.tutorialspoint.com/python/python_gui_programming.htm>
2. <https://www.dummies.com/article/technology/programming-web-design/python/using-tkinter-widgets-in-python-141443>
3. <https://youtube.com/playlist?list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV>
4. <https://www.imgonline.com.ua/eng/scan-qr-bar-code.php>

(To Scan QR code on Laptop)

1. <https://stackoverflow.com/questions/45067855/generating-a-list-with-string-and-number>
2. <https://stackoverflow.com/questions/10624937/convert-datetime-object-to-a-string-of-date-only-in-python>