**Name**: ANANDHA KRISHNAN.V

**Class:** XII A

**Roll No.:**

## Reg No.:

**SESSION**

**2024-25**

***CERTIFICATE***

*This is to certify that the project entitled* ***“AIRWAYS MANAGEMENT SYSTEM”*** *submitted by**, Reg No: is bona-fide work carried out for class XII Practical Examination of Central Board of Secondary Education 2024-2025*

*Teacher in charge External Examiner*

*Principal*

**ACKNOWLEDGEMENT**

I would like to express my special thanks of gratitude to our Principal   
**Mr. K. Prathap Rana** and as well as to my teacher **Mr. Sunil Kumar N.**,   
Department of Computer Science, who gave me the guidance for the golden opportunity to do this wonderful project on the topic **“*AIRWAYS MANAGEMENT SYSTEM*”** which also helped me in doing a lot of researches and came to know about so many new things. I am really thankful to them.

Secondly I would like to thank my parents and friends who helped me lot in finishing this project within the limited time frame. The last but not the least I would like to thank the almighty god for helping me to carry out the work to a good and great success.

**ABSTRACT**

Airways management system is an application for maintaining a person's flight booking. In this project, the working of Air Ticket Booking Management System includes flight schedules, fare fees, passenger reservations and ticket accounts. An Airline Management System is a managerial software which targets to control all operations of an airline. Airlines provide transport services for their passengers. They carry or hire aircraft for this purpose. All operations of an airline company are controlled by this airline management system.

**CONTENTS**

1. MANUAL SYSTEM 6

2. PROPOSED SYSTEM 7

3. INTRODUCTION OF THE PROJECT 8

4. TEAM MEMBERS AND TEAM DETAILS 9

5. OBJECTIVE OF THE PROJECT 10

6. SYSTEM SPECIFICATIONS 11

7. WORKING ENVIRONMENT 12

8. MODULE DESCRIPTION 18

9. SYSTEM DEVELOPMENT LIFE CYCLE 20

10. ALGORITHM 23

11. FLOWCHART 29

12. SOURCE CODE 36

13. TABLE DESIGN 84

14. CONCLUSION 85

15. FUTURE ENHANCEMENT 86

16. APPENDIX 87

17. BIBLIOGRAPHY 103

## MANUAL SYSTEM

Manual system means a system which does its work itself not by help of any technology in which paper work has some special place. All conventional methods are in more use instead of new technologies.

Now as everybody knows that computer graphs at it extend means the more you can use computer system the more you can make your work easier. And if in this case, system is not computerized then it must face a lot of problems. Because every task gets complex and time consumable.

Problem in Manual System includes:

⦁ Cost

⦁ Difficulty in searching the records

⦁ Maintenance Problem

⦁ Time consuming

⦁ Tedious

## PROPOSED SYSTEM

Proposed system is a system which is computerized in every manner. Computerized system is not just adding machines but they can do much complex, tedious and cumbersome task. Processing of data by hand is satisfactory only when the amount of data to be processed is small and also the manual processing is slow, monotonous and often subject to error. Above explanation is clearly telling us that existing system contains a lot of deficiencies which can be removed only by following the proposed system.

Now a day, computer graph is at its extent. Computerization contains a lot of benefits so that everyone is chasing and following computerized items. Now, question arises what kind of help this project or computerized system can give to remove all disadvantages of this existing system.

## 3. INTRODUCTION

In day-to-day work process of an organization, there are lots of things happening, which need to be keep tracked, for our ease in future. Of course, there is lots of way to perform this task. We can either maintain these records manually or by using computerized system. In current scenario, where we always thing to increase productivity utilizing our working hours more and more, it is always recommended to go for such automated system, which provide me maximum facility to do the same in a less time and minimum manpower. So we have designed a commercial project for Airways Management System.

A computer-based management system is designed to handle all the primary information required to book flight tickets and cancel.

Separate database is maintained to handle all the details required for booking and canceling flight tickets. This project intends to introduce various user friendly activities, such as record updating, maintenance, and searching. The searching of record has been made quite simple as all the details of the flights can be obtained by simply logging in and flight ticket booking and cancelling can also be accomplished. These details are also being promptly automatically updated in the master file thus keeping the record up to-date.

For processing the data we have chosen MySQL as back end and Python as front end, which can manage database for a very large class of the possible application. This project is client-server-based application system to computerize at bank work. The modules involved currently in this system are Authentication, Ticket booking, Payment and Ticket cancelling.

**TEAM MEMBERS AND TEAM DETAILS**

Team Members:

ANANDHA KRISHNAN.V

DHANALEKSHMI.R

HARI DHEJUS.VS

Team details:

The project “Airways Management System" has been designed and developed solely by the fore mentioned individuals. The various components of the project were shared between the team members.

## 5. OBJECTIVE OF THE PROJECT

The main objective of Airways Management system is to manage details of flight tickets. This system is designed to provide travelers with a simple and efficient online platform to search, select, and purchase air tickets for domestic and international flights an air ticket booking project can significantly enhance the travel booking experience, providing convenience and efficiency to users while supporting the operational needs of travel agencies, airlines . Whether booking for business or leisure, the system aims to reduce the complexities associated with flight reservations, providing customers with a seamless travel planning experience from starting to ending.

## 6. SYSTEM SPECIFICATIONS

Hardware specification**:**

* + LED Monitor
  + Keyboard and mouse
  + Processor Speed: 533MHz
  + RAM: 2GB or More
  + Hard Disk: 2.00 GB

Software specification:

* + Operating System: Windows 10 or above
  + IDE: IDLE Python & Visual Studio Code
  + Visual Studio Code
  + RAM: 2GB or More
  + Hard Disk: 2.00 GB
  + Front End: Python 3.7 or above & CustomTkinter
  + Middleware : Python 3.7 or above
  + Back End: MySQL server 5.0 or above

**7. Working environment**

**Python:**

Python is a popular programming language. It was created by Guido van

Rossum, and released in 1991.

It is used for:

i) Web development (server side)

ii) Software development

iii) Data analysis

iv) System scripting

Python is a dynamic, high-level, free open source, and interpreted Programming language. It supports object-oriented programming as well as procedural-oriented programming. Python Is a high-level, general- purpose programming language. Its design philosophy emphasizes code read ability with the use of significant indentation.

**Features in Python**

There are many features in Python, some of which are :

**1. Free and Open Source**

Since it is open-source, this means that source code is also available to the

public. So, we can download it, use it as well as share it.

**2. Easy to code**

Python is a high-level programming language. Python is very easy to learn the   
 language as compared to other languages like C, C#, Java script, Java, etc. It   
 is very easy to code in the Python language and anybody can learn Python   
 basics in a few hours or days. It is also a developer-friendly language.

**3. Easy to Read**

Learning Python is quite simple. As was already established, Python's   
 syntax is straightforward. The code block is defined by the indentations rather   
 than by semicolons or brackets.

**4. Object-Oriented Language**

One of the key features of Python is Object-Oriented programming.   
 Python supports object-oriented language and concepts of classes, object   
 encapsulation, etc.

**5. GUI Programming Support**

Graphical User interfaces can be made using a module such as PyQt5,   
 PyQt4, wxPython, or Tk in python. PyQt5 is the most popular option for   
 creating graphical apps with Python.

**6. High-Level Language**

Python is a high-level language. When we write programs in Python, we do   
 not need to remember the system architecture, nor do we need to manage the   
 memory.

**7. Extensible feature**

Python is an Extensible language. We can write some Python code into C or   
 C++ language and, we can compile that code in C/C++ language.

**8. Easy to Debug**

Excellent information for mistake tracing. We will be able to quickly   
 identify and correct the majority of your program's issues once you understand   
 how to interpret Python's error traces. Simply by glancing at the code, you can   
 determine what it is designed to perform.

**9. Python is a Portable language**

Python language is also a portable language. For example, if we have   
 Python code for windows and if we want to run this code on other platforms   
 such as Linux, Unix, and Mac then we do not need to change it, we can run   
 this code on any platform.

**10. Python is an integrated language**

Python is also an integrated language because we can easily integrate   
 Python with other languages like C, C++, etc.

**11. Interpreted Language:**

Python is an Interpreted Language because Python code is executed line by   
 line at a time. like other languages C, C++, Java, etc. there is no need to   
 compile Python code this makes it easier to debug our code. The source code   
 of Python is converted into an immediate form called byte code.

**12. Large Standard Library**

Python has a large standard library that provides a rich set of modules   
 and functions so you do not have to write your own code for everything.   
 There are many libraries present in Python such as regular expressions, unit-  
 testing, web browsers, etc.

**13. Dynamically Typed Language**

Python is a dynamically-typed language.

**14. Frontend and backend development**

With a new project py script, you can run and write Python codes in   
 HTML with the help of some simple. Backend is the strong forte of Python   
 it's extensively used for this work because of its frameworks like Django and   
 Flask

**15. Allocating Memory Dynamically**

In Python, the variable data type does not need to be specified. The memory   
 is automatically allocated to a variable at runtime when it is given a value.

**What python can do?**

Python can be used on a server to create web applications.

Python can be used alongside software to create workflows.

Python can connect to database systems. It can also read and modify files.

Python can be used to handle big data and perform complex mathematics.

Python can be used for rapid prototyping, or for production-ready software development.

**What is MySQL?**

MySQL, the most popular Open-Source SQL database management system, is developed, distributed, and supported by Oracle Corporation. MySQL is the world's most popular open-source database. According to DB-Engines, MySQL powers many of the most accessed applications, including Facebook, Twitter, Netflix, etc. Since MySQL is open source, developers love its high performance, reliability, and ease of use. MySQL is fast, reliable, scalable, and easy to use. It was originally developed to handle large databases quickly and has been used in highly demanding production environments for many years. Although MySQL is under constant development, it offers a rich and useful set of functions MySQL's connectivity, speed, and security make it highly suited for accessing databases on the internet.

MySQL's key benefits include

* + - * Ease of use
      * Reliability
      * Scalability
      * Performance
      * High availability
      * Security
      * Flexibility

**Interface Python with MySQL**

MYSQL Python/Connector is an interface for connecting to a MySQL database server from Python. It implements the Python Database API and is built on top of the MySQL.

General work flow of interface Python with MySQL

1. Connect to the MySQL server.

2. Create a new database.

3. Connect to the newly created or an existing database.

4. Execute a SQL query and fetch results.

5. Inform the database if any changes are made to a table.

6. Close the connection to the MySQL server.

## 8. MODULE DESCRIPTION

To develop software which maintains all the record of day to day booking, canceling, searching available flight etc. All the records are stored in MYSQL Database. The project consists of four modules:

1. Authentication module

2. Flight ticket booking module

3. Flight ticket cancelling module

4. Payment module

5. Flight manager

6. Flight Confirmation module

7. Page Navigation

**MODULE DESCRIPION:**

1. Authentication module: This module used for user authentication check for valid users using username and password.

* **Sign up:** Create a new account by providing username and password. The system checks for unique usernames to prevent duplication
* **Sign in:** login using a registered username and password. The system validates to credentials to ensure secure access.

1. Flight ticket booking module: this module is a software component that allows users to search, select, and purchase airline tickets.
2. Flight ticket cancelling module: The cancellation feature allows users to cancel their flight bookings under specific conditions.
3. Payment module : The module calculates the total cost of the booking
4. Flight manager module :This module consist of inserting flights and deleting flights
5. Flight confirmation module: Once the payment is successful, users receive a confirmation screen with a unique booking reference number summary of the booked flight(s), including dates, times, and location.
6. Page Navigation module : More logically navigating through pages using the concept of ‘Stack’.

## 9. SYSTEM DEVELOPMENT LIFE CYCLE

SDLC is a step-by-step procedure or systematic approach to develop software and it is followed within a software organization. It consists of various phases with describe how to design, develop, enhance, and maintain particular software.

**Phase 1:** Requirement collection and analysis

In this phase mainly focus on gathering the business needs from the customer. It determines the system? Who is going to use the system? What should be output data by the system? These questions are getting answered during this phase.

**Phase 2:** Feasibility study

In this step, we examine the feasibility of the proposed system.

This decision is taken based on the cost, time, resources etc.

**Phase 3:** Design

Design is a blue print of the application and it helps in specifying hardware and requirements of the system and helps in defining architecture of the system.

**Phase 4:** Coding

Once the system design document is ready in this phase, developers starts writing the code using any programming language i.e., they start developing the software.

Generally, task is divided in units or modules and assigned to the developers and this coding phase is the longest phase of SDLC.

**Phase 5:** Testing

During this phase, test engineers may encounter some bugs/defects which need to be sent to developers, the developers fix the bug and sent back to test engineers for testing.

**Phase 6**: Installation / Deployment

Once the product developed, tested, and works according to the requirement it is installed/deployed at customer place for their use.

**Phase 7**: Maintenance

When the customer starts using the software, they may face some issues and needs to be solved from time to time means need to fix those issue, tested and handed over back to the customer as soon as possible, which is done in the maintenance phase.

**10. ALGORITHM**

**1. System Setup and Initialization**

**1.1 Environment Preparation:**

**Install Required Packages:**

Import necessary libraries: `customtkinter` for GUI, `Pillow` for images, `pymysql` for database operations, and `tkcalendar` for date selection.

For each package, try importing it; if not installed, use `subprocess` to install it automatically via `pip`.

**1.2 Main Database Connection:**

**Establish MySQL Connection:**

Connect to MySQL using `pymysql.connect()` with credentials (host, user, password, database).

Create a cursor (`cur`) for executing SQL queries.

**1.3 GUI Window Configuration:**

**Initialize Main Window:**

Set up the main window using `CTk()` with parameters for title, geometry, and appearance mode.

Center the window using `Global\_Config` functions for a consistent display across devices.

**1.4 Database Initialization:**

**Create Tables if Needed:**

Use `DB\_INIT\_()` function to ensure required tables are present:

`user\_details`: Stores user information.

`flights`: Stores available flight data.

`booking`: Tracks bookings with references to users and flights.

`payment`: Manages payment records for each booking.

***Outcome: The main GUI window and database are set up, and essential tables are ready for interaction.***

**2. User Authentication Management**

**2.1 Registration Process (`PG\_Sign\_Up`):**

**Display SignUp Form:**

Show input fields for name, username, password, email, phone, gender, and DOB (using `tkcalendar` for date input).

**Validate Inputs:**

Check that no fields are left empty, passwords match, and the username is unique by querying `user\_details`.

**Store User Data:**

If validation succeeds, insert user data into `user\_details`.

Provide feedback on successful registration or display error messages for issues.

**2.2 Login Process (`PG\_Sign\_in`):**

**Display Login Form:**

Show fields for username/email and password.

**Verify Credentials:**

Encrypt password entered

Query `user\_details` to authenticate credentials.

If valid, update `\_isSignedIn` to `True` and assign `User` the username.

**Session Management:**

Redirect authenticated users to the flight search or payment page based on session state.

***Outcome: Users can register and log in, with sessions managed for authenticated access to specific features.***

**3. Navigation and Page Transition Management**

**3.1 Navigation Buttons:**

**Define Navigation Elements:**

Use `CTkButton()` to create buttons for key pages (Sign Up, Sign In, Flight Search).

Implement `GoBack()` to manage backward navigation using stack, clearing the current frame and displaying the previous one.

**3.2 Session Based Content Display:**

**Conditional Rendering:**

Display buttons and page content based on `\_isSignedIn` state, ensuring users only access relevant features based on their session status.

***Outcome: Seamless navigation across pages, with session aware controls for user specific access.***

**4. Flight Search and Selection**

**4.1 Flight Search Page (`PG\_Get\_Flight\_Details`):**

**Display Search Form:**

Show dropdowns for selecting departure and destination, radio buttons for oneway or return, and date pickers for travel dates.

**Validate Inputs:**

Ensure all fields are complete and dates are valid (e.g., departure before return).

**Query Flights:**

Build and execute SQL queries to fetch available flights from `flights` based on input criteria.

**4.2 Display Search Results (`PG\_search\_flight\_`):**

**Scrollable Results Display:**

Use `CTkScrollableFrame` to display flight options with details like departure, arrival, airline, and price.

**Interactive Selection:**

Add buttons for each flight entry. Authenticated users are redirected to payment; nonauthenticated users are prompted to sign in.

***Outcome: Users can search for flights and view matching options, with booking access based on session state.***

**5. Booking and Ticket Management**

**5.1 Booking Confirmation (`booking()`):**

**Generate Ticket Code:**

Use `Ticket\_Code\_Gen.Gen\_Code()` to create a unique code for each booking.

**Store Booking Data:**

Insert booking details into the `booking` table, associating it with the user and flight.

**Allow Ticket Save:**

Provide users with the option to save ticket details to a local file using `filedialog.asksaveasfilename`.

***Outcome: Bookings are stored with unique ticket codes, and users have access to download their ticket.***

**6. Integrated Payment System**

**6.1 Payment Interface (`PG\_Payment()`):**

**Display Payment Options:**

Create options for UPI and net banking payments. Ensure each option clears previous components in the frame to display relevant input fields.

**6.2 UPI Payment Workflow (`on\_UPI\_btn\_click()`):**

**Display UPI Input:**

Show input fields for UPI number and display the total flight price.

**Validation and Database Entry:**

Check that fields are not empty, validate input, and insert payment record in `payment`.

**PostPayment Handling:**

Display confirmation message, initiate booking, and redirect back to the main page.

**6.3 Net Banking Workflow (`on\_NET\_btn\_click()`):**

**Display Banking Fields:**

Show fields for entering account number and display the total price.

**Validation and Database Entry:**

Validate inputs and insert the payment record in `payment`.

**PostPayment Handling:**

Confirm payment, initiate ticket booking, and navigate to the main page.

***Outcome: Payment is processed with a choice of methods, updating the database and confirming booking completion.***

**7. User Interface Feedback and Error Handling**

**7.1 Dynamic Feedback (`errorLabeling()`):**

**RealTime Feedback:**

Use `errorLabeling()` to display temporary messages for successful or failed actions, ensuring users receive immediate feedback.

**7.2 Validation and Exception Handling:**

**Input Validation:**

Check all input fields before database operations, and handle exceptions with tryexcept blocks to manage SQL errors.

***Outcome: The system provides clear feedback and error handling, enhancing user experience and stability.***

**8. Additional Account Features**

**8.1 Account Management:**

**Settings and History:**

Implement options for flight cancellation, booking history, account edits, and logout.

**Flight Cancellation:**

Prompt the user for a ticket ID, verify it, and update the booking to inactive if validated.

**View Booking History:**

Retrieve booking records for the loggedin user from `booking` and display them.

**Edit Account:**

Provide options to update user details, ensuring validation of changes.

***Outcome: Users can manage bookings, view history, and update account information securely.***

**8.2 Flight And Admin Management (For Admins Only)**

**Add Flights:**

Provides option to add new flights.

**Remove Flights:**

Provides option to remove flights

**Add Admin:**

Provides option to add new admins.

**Remove Admin:**

Provides option to remove an user from admin position.

**Modify Flights:**

Provides option to modify existing flights.

***Outcome: Admin can manage Flights and access privilage.***

**9. System Exit and Resource Management**

**9.1 Resource Cleanup:**

**Database Connection Closure:**

Ensure `cur.close()` and `con.close()` are called on application exit to avoid resource leaks.

**Application Termination:**

End the main loop with `root.mainloop()`, ensuring a smooth and errorfree shutdown.

***Outcome: The application exits gracefully, with all resources properly released.***

**Potential Enhancements and Security Considerations**

**Security Enhancements:**

Use stored procedure (Stored Proc) for SQL queries to protect against SQL injection.

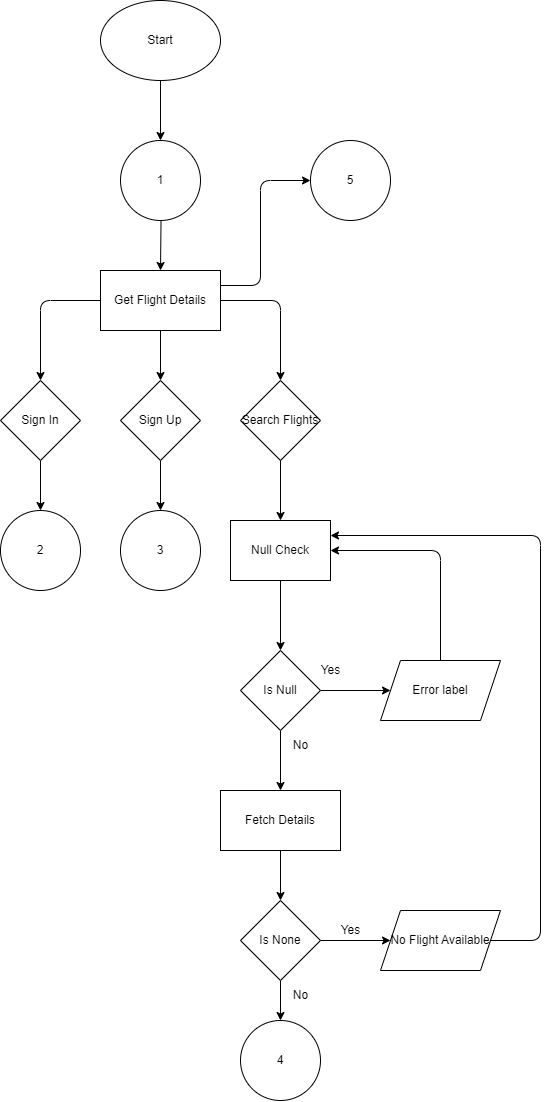
**Feature Extensions:**

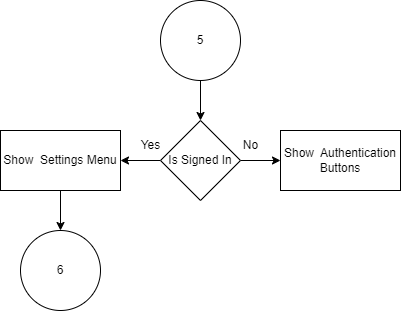
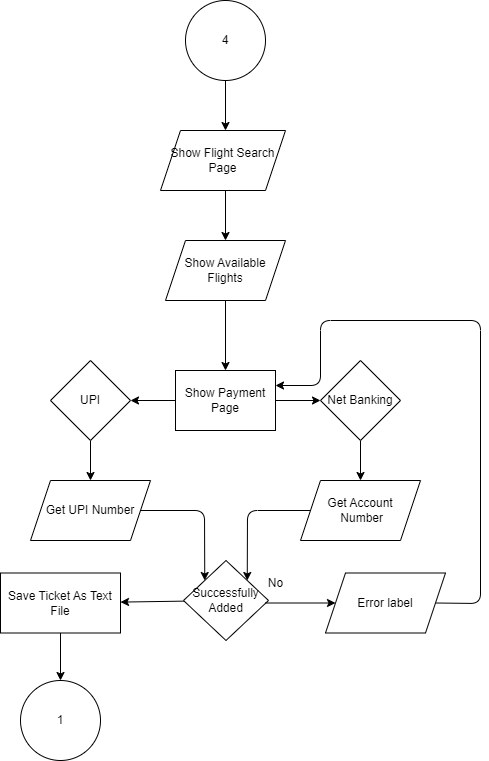
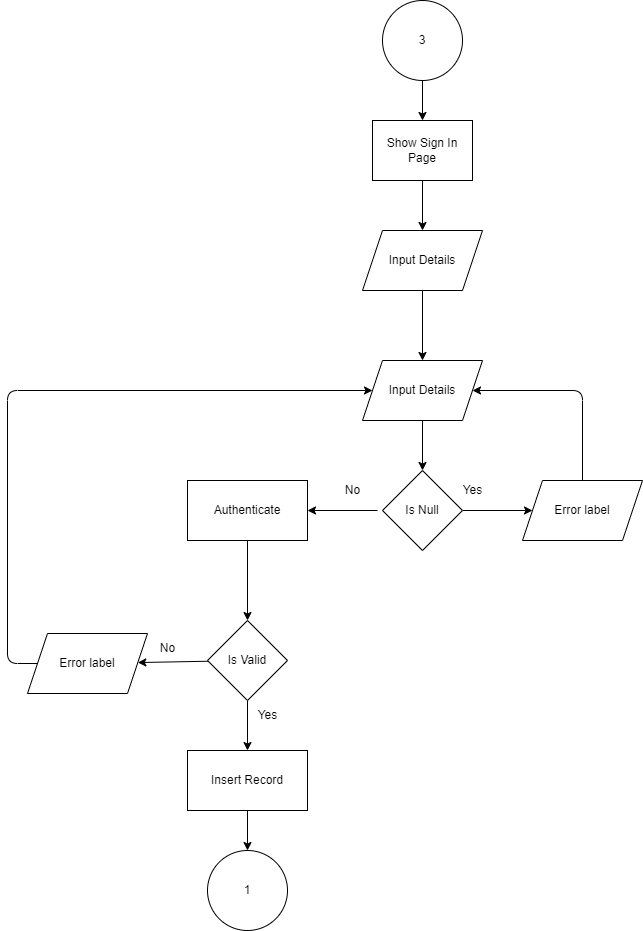
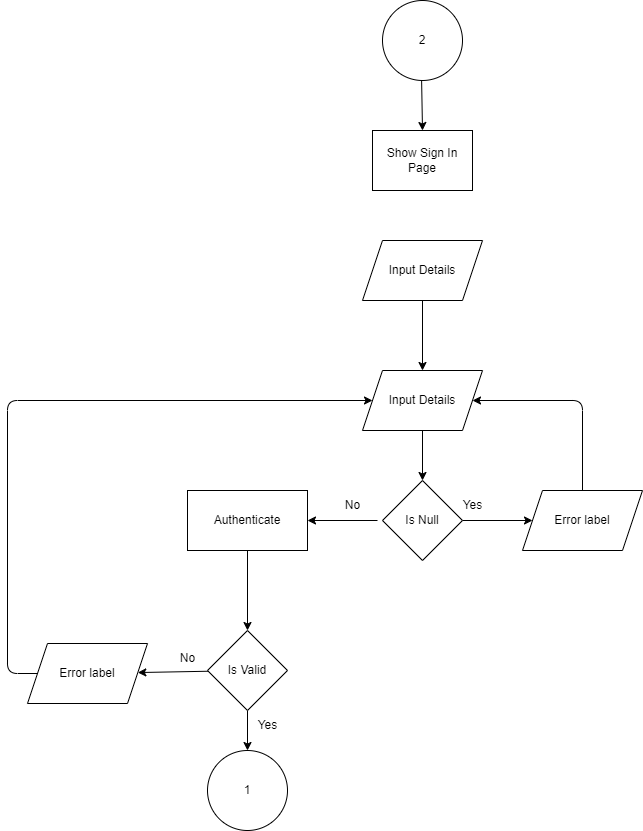
Add more payment options and integrate third party APIs for additional functionality.

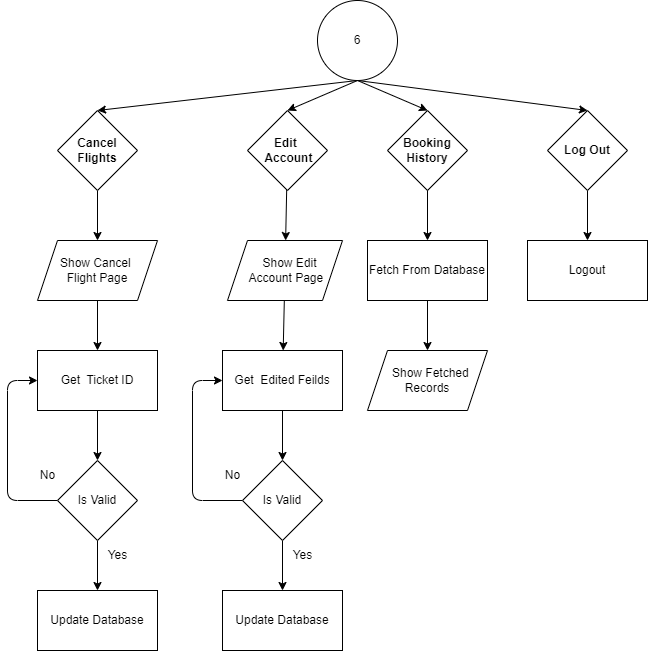
**Notifications:**

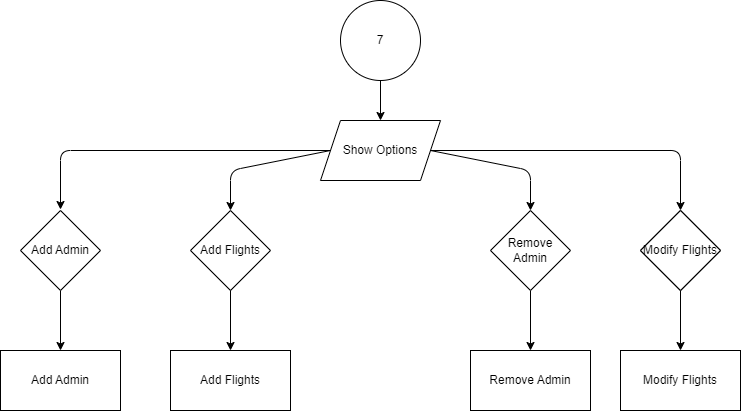
Implement email or in app notifications for payment confirmation and booking

**11.FLOWCHART**

****







**12. Source code**

**index.py**

**import subprocess**

**import smtplib**

**import os**

**import sys**

**def install(package):**

**subprocess.check\_call([sys.executable, "-m", "pip", "install", package])**

**required\_packages = ["customtkinter",**

**"matplotlib",**

**"pillow",**

**"pymysql",**

**"colorama",**

**"tkcalendar"**

**]**

**for package in required\_packages:**

**try:**

**\_\_import\_\_(package)**

**print(f"{package} Succesfully installed.")**

**except ImportError:**

**install(package)**

**# Created Modules**

**import Ticket\_Code\_Gen as TCG**

**from Usable\_screen import ScreenGeometry as SG**

**from flights import major\_airports**

**import Global\_Config as GC**

**from Crypter import crypt**

**from Stack import stack**

**# Global Modules**

**from customtkinter import \***

**import pymysql**

**from tkcalendar import Calendar**

**import tkinter as tk**

**from pathlib import Path**

**from tkcalendar import Calendar**

**from datetime import datetime**

**from tkinter import filedialog**

**m\_r\_width, m\_r\_height = SG().GetUsableScreenSize()[0], SG().GetUsableScreenSize()[1]**

**root = CTk()**

**root.title("http:www.HADAirlineManagementSystem.com/")**

**set\_appearance\_mode("Dark")**

**GC.centreScreen(root,root, m\_r\_width, m\_r\_height)**

**root.state("zoomed")**

**root.minsize(m\_r\_width, m\_r\_height)**

**root.geometry(f"{m\_r\_width}x{m\_r\_height}")**

**con = pymysql.connect(**

**host = "localhost",**

**user = "root",**

**passwd = "\*password\*11",**

**)**

**cur = con.cursor()**

**#---------GLOBAL VARIABLES --------**

**Stack = stack()**

**pageDict = {**

**1: "PG\_Get\_Flight\_Details",**

**2: "PG\_search\_flight\_",**

**3: "PG\_Sign\_in",**

**4: "PG\_Sign\_Up",**

**5: "PG\_Payment",**

**6: "\_on\_UPI\_btn\_click",**

**7: "on\_NET\_btn\_click",**

**}**

**global on\_NET\_btn\_click**

**global \_on\_UPI\_btn\_click**

**\_isSignedIn = False**

**is\_flight\_details\_obtained = False**

**User = ""**

**isAdmin = False**

**prev\_page = 0**

**glb\_clr\_1 = "blue"**

**glb\_clr\_2 = "green"**

**BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent**

**glb\_clr\_3 = "yellow"**

**destroy\_after = None**

**global PG\_Get\_Flight\_Details**

**def DB\_INIT\_():**

**try :**

**cur.execute("CREATE DATABASE IF NOT EXISTS `airwaysms2\_0` /\*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci \*/ /\*!80016 DEFAULT ENCRYPTION='N' \*/;")**

**cur.execute("USE airwaysms2\_0")**

**cur.execute("""CREATE TABLE IF NOT EXISTS `user\_details` (**

**`UID` int NOT NULL AUTO\_INCREMENT,**

**`UF\_name` varchar(100) DEFAULT NULL,**

**`UL\_name` varchar(100) DEFAULT NULL,**

**`U\_name` varchar(100) NOT NULL,**

**`U\_Gmail` varchar(100) NOT NULL,**

**`U\_phno` varchar(12) DEFAULT NULL,**

**`U\_password` varchar(100) NOT NULL,**

**`U\_dob` date DEFAULT NULL,**

**`U\_AGE` int DEFAULT NULL,**

**`U\_gender` varchar(5) DEFAULT NULL,**

**`U\_isAdmin` tinyint DEFAULT '0',**

**`U\_isActive` tinyint DEFAULT '1',**

**PRIMARY KEY (`UID`),**

**UNIQUE KEY `U\_name\_UNIQUE` (`U\_name`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;""")**

**cur.execute("""CREATE TABLE IF NOT EXISTS `flights` (**

**`F\_ID` int NOT NULL AUTO\_INCREMENT,**

**`F\_Departure` varchar(100) DEFAULT NULL,**

**`F\_Arrival` varchar(100) DEFAULT NULL,**

**`F\_Airline` varchar(45) NOT NULL,**

**`F\_price` int DEFAULT NULL,**

**`F\_IsActive` int DEFAULT 1,**

**PRIMARY KEY (`F\_ID`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;**

**""")**

**cur.execute("""CREATE TABLE IF NOT EXISTS `booking` (**

**`BID` varchar(100) NOT NULL,**

**`BU\_NAME` varchar(100) DEFAULT NULL,**

**`B\_FLIGHT` int DEFAULT NULL,**

**`IS\_ACTIVE` int DEFAULT "1",**

**PRIMARY KEY (`BID`),**

**KEY `B\_FORIEGN\_KEY\_idx` (`B\_FLIGHT`),**

**CONSTRAINT `B\_FORIEGN\_KEY` FOREIGN KEY (`B\_FLIGHT`) REFERENCES `flights` (`F\_ID`)**

**) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;""")**

**cur.execute("""CREATE TABLE IF NOT EXISTS `payment` (**

**`PID` int NOT NULL AUTO\_INCREMENT,**

**`P\_UID` int DEFAULT NULL,**

**`AMOUNT` int DEFAULT NULL,**

**`P\_STATUS` int DEFAULT NULL,**

**`P\_ACC\_NUM` varchar(45) DEFAULT NULL,**

**`P\_UPI\_NUM` varchar(45) DEFAULT NULL,**

**`P\_METHOD` varchar(45) DEFAULT NULL,**

**PRIMARY KEY (`PID`),**

**KEY `P\_UID\_idx` (`P\_UID`),**

**CONSTRAINT `P\_UID` FOREIGN KEY (`P\_UID`) REFERENCES `user\_details` (`UID`)**

**) ENGINE=InnoDB AUTO\_INCREMENT=19 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;**

**""")**

**except Exception as e:**

**print(e)**

**finally :**

**print("Sucessfully Initialized Database.")**

**cur.execute("USE airwaysms2\_0 ;")**

**DB\_INIT\_()**

**#--------------------- GLOBAL FUNCTIONS ---------------------------------**

**def GoBack(PREV\_PAGE = 0):**

**for i in Main\_fame.winfo\_children():**

**i.destroy()**

**Stack.Pop()**

**PREV\_PAGE = Stack.Pop()**

**Main\_frm\_Authentication\_Btns()**

**if PREV\_PAGE == 1:**

**PG\_Get\_Flight\_Details()**

**if PREV\_PAGE == 2:**

**PG\_search\_flight\_()**

**if PREV\_PAGE == 3:**

**PG\_Sign\_in()**

**if PREV\_PAGE == 4:**

**PG\_Sign\_Up()**

**if PREV\_PAGE == 5:**

**PG\_Payment()**

**if PREV\_PAGE == 6:**

**\_on\_UPI\_btn\_click()**

**if PREV\_PAGE == 7:**

**on\_NET\_btn\_click()**

**def createRadioButton (\_frame ,\_text : str , \_value, \_variable, \_command, \_xpos : int, \_ypos : int):**

**tmpRdBtn = CTkRadioButton(\_frame, text = \_text , value = \_value, variable = \_variable,width = 75, command = lambda :(\_command()) )**

**tmpRdBtn.place(x =\_xpos, y = \_ypos)**

**return tmpRdBtn**

**def errorLabeling(\_Master, \_text : str, \_font = ("Bradley Hand ITC" , 18, "italic", "bold"), \_textcolor = "red", \_x = 10, \_y = 10, \_cooldowntime = 3000):**

**error\_label = CTkLabel(\_Master, text = \_text, font = \_font, text\_color = \_textcolor)**

**error\_label.place(x = \_x, y = \_y)**

**def refresh():**

**error\_label.destroy()**

**error\_label.after(\_cooldowntime, refresh)**

**#------------------------------------------------------**

**Main\_fame = CTkFrame(root, width = m\_r\_width, height= m\_r\_height-24, border\_width=2, border\_color= "#007acc", fg\_color="transparent")**

**Main\_fame.place(x = 0, y = 0)**

**global booking**

**def booking ():**

**Ticket\_Code = TCG.Gen\_Code()**

**cur.execute("SELECT \* FROM flights WHERE F\_ID ='%s'", Flight\_ID)**

**cur.execute("SELECT COUNT(\*) FROM booking")**

**F\_id = cur.fetchone()[0]**

**global User**

**cur.execute("INSERT INTO booking (BID, BU\_NAME, B\_FLIGHT) VALUES (%s, %s, %s)", (Ticket\_Code, User, Flight\_ID))**

**con.commit()**

**file\_path = filedialog.asksaveasfilename(defaultextension = ".txt",**

**filetypes=[("Text Files", "\*.txt"),**

**("All Files", "\*.\*")**

**],**

**initialfile= "SELECT WHERE TO SAVE THE TICKET.txt")**

**temp\_qry = f"SELECT F\_Departure, F\_Arrival, F\_Airline, F\_price FROM flights WHERE flights.F\_ID = %s ;"**

**cur.execute(temp\_qry, (Flight\_ID,))**

**result = cur.fetchone()**

**if file\_path:**

**with open(file\_path, "w")as f:**

**f.write(f"Ticket Id : {Ticket\_Code}, User : {User}, Departure : {result[0]}, Arrival : {result[1]}\**

**Airline : {result[2]}, Price : {result[3]}")**

**errorLabeling(form\_frm, "Sucessfully Booked", \_textcolor = "green", \_x = 120, \_y = 170)**

**global PG\_Payment**

**def PG\_Payment():**

**root.title ("http:www.HADAirlineManagementSystem.com/Payment")**

**for i in Main\_fame.winfo\_children():**

**i.destroy()**

**Stack.Push(5)**

**form\_frm\_width = 400**

**form\_frm\_height = 210**

**global form\_frm**

**form\_frm = CTkFrame(Main\_fame, width=form\_frm\_width, height=form\_frm\_height)**

**form\_frm.place(x = (m\_r\_width/(2))-(form\_frm\_width/2), y = (m\_r\_height/(2)-(form\_frm\_height/2)))**

**dummy\_back\_btn = CTkButton(Main\_fame, text="Back", command = GoBack)**

**dummy\_back\_btn.place(x =10, y = 10)**

**Payment\_Method\_label = CTkLabel(form\_frm, text= "Select Payment Method")**

**Payment\_Method\_label.place(x = 25, y = 10)**

**CBF\_width = 290**

**CBF\_height = 28**

**Centre\_btn\_frame = CTkFrame(form\_frm, height=CBF\_height, width= CBF\_width, fg\_color= "transparent")**

**Centre\_btn\_frame.place(x = (form\_frm\_width/(2))-(CBF\_width/2),**

**y = (form\_frm\_height/(2)-(CBF\_height/2)) )**

**global on\_NET\_btn\_click**

**global on\_UPI\_btn\_click**

**def on\_UPI\_btn\_click():**

**root.title ("http:www.HADAirlineManagementSystem.com/Payment/UPI")**

**for i in form\_frm.winfo\_children():**

**i.destroy()**

**Stack.Push(6)**

**dummy\_back\_btn = CTkButton(Main\_fame, text="Back", command = GoBack)**

**dummy\_back\_btn.place(x =10, y = 10)**

**Temp\_Entry\_Width = 350**

**UPI\_Method\_label = CTkLabel(form\_frm, text= "UPI -")**

**UPI\_Method\_label.place(x = 25, y = 10)**

**UPI\_Number\_Entry = CTkEntry(form\_frm, placeholder\_text= "UPI Number",width= Temp\_Entry\_Width)**

**UPI\_Number\_Entry.place(x =25, y = 50)**

**cur.execute(f"SELECT F\_Price FROM flights WHERE F\_ID = '{Flight\_ID}'")**

**Amount = str(cur.fetchone()[0])**

**Amount\_LBL = CTkLabel(form\_frm,text= f"Amount : {Amount}",width= Temp\_Entry\_Width)**

**Amount\_LBL.place(x =25, y = 90)**

**def \_on\_UPI\_pay\_btn\_click():**

**UPI\_Number = UPI\_Number\_Entry.get()**

**cur.execute(f"SELECT F\_Price FROM flights WHERE F\_ID = '{Flight\_ID}'")**

**Amount = str(cur.fetchone()[0])**

**if UPI\_Number == "" or Amount == "":**

**errorLabeling(form\_frm, "Feilds Can Not Be Empty", \_x = 90, \_y = 170)**

**elif UPI\_Number.isdigit() == False or Amount.isdigit() == False:**

**is\_flight\_details\_obtained**

**else:**

**Amount = int(Amount)**

**UPI\_Number = int(UPI\_Number)**

**tempqry = "SELECT COUNT(\*) FROM payment"**

**cur.execute(tempqry)**

**p\_id = int(cur.fetchone()[0])+1**

**global User**

**tempqry = f"SELECT UID FROM user\_details WHERE U\_name = '{User}'"**

**cur.execute(tempqry)**

**Uid = int(cur.fetchone()[0])**

**tempqry = "INSERT INTO payment (PID, AMOUNT, P\_STATUS, P\_UPI\_NUM, P\_METHOD) VALUES (%s, %s, %s, %s, %s)"**

**cur.execute(tempqry, (p\_id,Amount,400,crypt(str(UPI\_Number)).encrypt(), "UPI"))**

**errorLabeling(form\_frm, "Payment Sucessful", \_textcolor = "green", \_x = 110, \_y = 170)**

**def delayed\_lbl():**

**errorLabeling(form\_frm, "Booking Ticket ...", \_textcolor = "green", \_x = 1100, \_y = 170)**

**def delayed():**

**booking()**

**PG\_Get\_Flight\_Details()**

**root.after(3000, delayed)**

**con.commit()**

**root.after(3000, delayed\_lbl)**

**Pay\_btn = CTkButton(form\_frm, width= Temp\_Entry\_Width, text = "Pay", corner\_radius= 100, command= \_on\_UPI\_pay\_btn\_click)**

**Pay\_btn.place(x = 25, y = 130)**

**UPI\_btn = CTkButton(Centre\_btn\_frame, text = "UPI", command= on\_UPI\_btn\_click)**

**UPI\_btn.place(x = 0, y = 0)**

**def on\_NET\_btn\_click():**

**root.title ("http:www.HADAirlineManagementSystem.com/Payment/UPI")**

**for i in form\_frm.winfo\_children():**

**i.destroy()**

**Stack.Push(7)**

**dummy\_back\_btn = CTkButton(Main\_fame, text="Back", command = GoBack)**

**dummy\_back\_btn.place(x =10, y = 10)**

**Temp\_Entry\_Width = 350**

**NETB\_Method\_label = CTkLabel(form\_frm, text= "NET BANKING -")**

**NETB\_Method\_label.place(x = 25, y = 10)**

**NETB\_Number\_Entry = CTkEntry(form\_frm, placeholder\_text= "Account Number",width= Temp\_Entry\_Width)**

**NETB\_Number\_Entry.place(x =25, y = 50)**

**cur.execute(f"SELECT F\_Price FROM flights WHERE F\_ID = '{Flight\_ID}'")**

**Amount = str(cur.fetchone()[0])**

**Amount\_LBL = CTkLabel(form\_frm,text= f"Amount : {Amount}",width= Temp\_Entry\_Width)**

**Amount\_LBL.place(x =25, y = 90)**

**def \_on\_NETB\_pay\_btn\_click():**

**ACC\_Number = NETB\_Number\_Entry.get()**

**cur.execute(f"SELECT F\_Price FROM flights WHERE F\_ID = '{Flight\_ID}'")**

**Amount = str(cur.fetchone()[0])**

**if ACC\_Number == "" or Amount == "":**

**errorLabeling(form\_frm, "Feilds Can Not Be Empty", \_x = 90, \_y = 170)**

**elif ACC\_Number.isdigit() == False or Amount.isdigit() == False:**

**is\_flight\_details\_obtained**

**else:**

**Amount = int(Amount)**

**ACC\_Number = int(ACC\_Number)**

**tempqry = "SELECT COUNT(\*) FROM payment"**

**cur.execute(tempqry)**

**p\_id = int(cur.fetchone()[0])+1**

**global User**

**tempqry = f"SELECT UID FROM user\_details WHERE U\_name = '{User}'"**

**cur.execute(tempqry)**

**Uid = int(cur.fetchone()[0])**

**tempqry = "INSERT INTO payment (PID, AMOUNT, P\_STATUS, P\_UPI\_NUM, P\_METHOD) VALUES (%s, %s, %s, %s, %s)"**

**cur.execute(tempqry, (p\_id,Amount,400,crypt(str(ACC\_Number)).encrypt(), "NET"))**

**con.commit()**

**errorLabeling(form\_frm, "Payment Sucessful", \_textcolor = "green", \_x = 110, \_y = 170)**

**def delayed\_lbl():**

**errorLabeling(form\_frm, "Booking Ticket ...", \_textcolor = "green", \_x = 1100, \_y = 170)**

**def delayed():**

**booking()**

**PG\_Get\_Flight\_Details()**

**root.after(3000, delayed)**

**con.commit()**

**root.after(3000, delayed\_lbl)**

**Pay\_btn = CTkButton(form\_frm, width= Temp\_Entry\_Width, text = "Pay", corner\_radius= 100, command= \_on\_NETB\_pay\_btn\_click)**

**Pay\_btn.place(x = 25, y = 130)**

**NetBanking\_btn = CTkButton(Centre\_btn\_frame, text = "Net Banking", command= on\_NET\_btn\_click)**

**NetBanking\_btn.place(x = 150, y = 0)**

**Main\_frm\_Authentication\_Btns()**

**#=> --------Sign Up --------------------**

**global PG\_Sign\_Up**

**def PG\_Sign\_Up() :**

**root.title ("http:www.HADAirlineManagementSystem.com/SignUp")**

**for i in Main\_fame.winfo\_children():**

**i.destroy()**

**Stack.Push(4)**

**form\_frm\_width = 400**

**form\_frm\_height = 600**

**form\_frm = CTkFrame(Main\_fame, width=form\_frm\_width, height=form\_frm\_height)**

**form\_frm.place(x = (m\_r\_width/(2))-(form\_frm\_width/2),**

**y = (m\_r\_height/(2)-(form\_frm\_height/2))**

**)**

**def NullCheck():**

**global DOB\_selected\_date, cal**

**F\_name = F\_name\_Entry.get()**

**L\_name = L\_name\_Entry.get()**

**U\_name = U\_name\_Entry.get()**

**\_Gender = Gender.get()**

**if "cal" in globals():**

**dob = cal.get\_date()**

**dob\_dt = datetime.strptime(dob, "%Y-%m-%d")**

**today = datetime.today()**

**Age = today.year - dob\_dt.year**

**if (today.month, today.day) > (dob\_dt.month, dob\_dt.day):**

**Age +=1**

**else :**

**dob = ""**

**Gmail = gmail\_Entry.get()**

**\_pass = crypt(pass\_Entry.get()).encrypt()**

**\_re\_pass = crypt(re\_pass\_Entry.get()).encrypt()**

**phonenumber = phonnumber\_Entry.get()**

**tmp\_qry =f"SELECT U\_name FROM user\_details WHERE U\_name= '{U\_name}'"**

**cur.execute(tmp\_qry)**

**row = cur.fetchone()**

**if F\_name == "" or L\_name == "" or U\_name == "" or dob == "" or Gmail == "" or \_pass == "" or \_re\_pass == "" or phonenumber == "" or \_Gender == "other" :**

**errorLabeling(form\_frm, "Feilds Cannot Be Null", \_x = 110, \_y = 410)**

**if \_pass != \_re\_pass:**

**errorLabeling(form\_frm, "Passwords Don't Match", \_x = 110, \_y = 410)**

**elif row :**

**errorLabeling(form\_frm, "Username Already Exist", \_x = 110, \_y = 410)**

**else:**

**cur.execute("SELECT Count(\*) FROM user\_details")**

**ans = cur.fetchone()**

**new\_uid = ans[0] + 1**

**tmp\_qry = """**

**INSERT INTO user\_details (UID, UF\_name, UL\_name, U\_name, U\_Gmail, U\_phno, U\_password, U\_dob, U\_gender, U\_AGE)**

**VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)**

**"""**

**cur.execute(tmp\_qry, (new\_uid, F\_name, L\_name, U\_name, Gmail, phonenumber, \_pass, dob, \_Gender, Age))**

**con.commit()**

**cur.execute("SELECT Count(\*) FROM user\_details")**

**ans2 = cur.fetchone()**

**con.commit()**

**if ans2 > ans :**

**errorLabeling(form\_frm, "Succesfully Added", \_textcolor = "green", \_x = 110, \_y = 410)**

**def \_fun():**

**global \_isSignedIn, User**

**\_isSignedIn = True**

**User = U\_name**

**PG\_Get\_Flight\_Details()**

**errorLabeling(form\_frm, "Succesfully Added", \_textcolor = "green", \_x = 110, \_y = 410)**

**fun\_lbl = CTkLabel(form\_frm)**

**fun\_lbl.after(4000, \_fun)**

**else:**

**errorLabeling(form\_frm, "Error Occured While Inserting Try Restarting", \_x = 5, \_y = 410)**

**dummy\_back\_btn = CTkButton(Main\_fame,text="Back", command = GoBack)**

**dummy\_back\_btn.place(x =10, y = 10)**

**F\_name\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text="First Name")**

**F\_name\_Entry.place(x = 25, y = 10)**

**L\_name\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text="Last Name")**

**L\_name\_Entry.place(x = 25, y = 50)**

**def Func\_radio\_btn():**

**global \_Gender**

**\_Gender = Gender.get()**

**global Gender**

**Gender = StringVar(value = "other")**

**rd\_btn\_y\_pos = 90**

**male\_radio\_btn = createRadioButton(form\_frm, "Male","M",Gender,Func\_radio\_btn,25, rd\_btn\_y\_pos)**

**female\_radio\_btn = createRadioButton(form\_frm,"Female", "F", Gender,Func\_radio\_btn, 25+130, rd\_btn\_y\_pos)**

**other\_radio\_btn = createRadioButton(form\_frm,"Other", "O", Gender,Func\_radio\_btn, 25+130 + 130, rd\_btn\_y\_pos)**

**global cal**

**def DOB\_open\_date\_picker():**

**top = CTkToplevel(form\_frm)**

**top.title("Select a Date")**

**top.attributes("-topmost", True)**

**global cal**

**cal = Calendar(top, selectmode='day', date\_pattern = "yyyy-mm-dd")**

**cal.pack(pady=10)**

**def select\_date():**

**global DOB\_selected\_date**

**DOB\_selected\_date = cal.get\_date()**

**DOB\_Date\_label.configure(text=f"Selected Date : {DOB\_selected\_date}")**

**top.destroy()**

**select\_button = CTkButton(top, text="Select Date", command=select\_date)**

**select\_button.pack(pady=10)**

**DOB\_Date\_Btn = CTkButton(form\_frm, text="Date Of Birth", command=DOB\_open\_date\_picker, corner\_radius=100)**

**DOB\_Date\_Btn.place(x=25, y =rd\_btn\_y\_pos+40)**

**DOB\_Date\_label = CTkLabel(form\_frm, text= "Select Date")**

**DOB\_Date\_label.place(x = 25+170, y = 90+40)**

**U\_name\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text="Username")**

**U\_name\_Entry.place(x = 25, y = 130+40)**

**gmail\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text="Gmail")**

**gmail\_Entry.place(x = 25, y = 170+40)**

**def Show\_pass():**

**if pass\_Entry.cget('show') == '\*':**

**pass\_Entry.configure(show='')**

**show\_btn.configure(text=" Hide ")**

**else:**

**pass\_Entry.configure(show='\*')**

**show\_btn.configure(text="Show")**

**def Re\_Show\_pass():**

**if re\_pass\_Entry.cget('show') == '\*':**

**re\_pass\_Entry.configure(show='')**

**re\_show\_btn.configure(text=" Hide ")**

**else:**

**re\_pass\_Entry.configure(show='\*')**

**re\_show\_btn.configure(text="Show")**

**pass\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text="Password", show = "\*")**

**show\_btn = CTkButton(pass\_Entry, width = 22, height=28, text="Show",border\_color="#565b5e",border\_width=2, fg\_color="transparent", command=Show\_pass)**

**show\_btn.place(x = 304, y=0)**

**pass\_Entry.place(x = 25, y = 170+80)**

**re\_pass\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text="Re-Password", show = "\*")**

**re\_show\_btn = CTkButton(re\_pass\_Entry, width = 22, height=28, text="Show",border\_color="#565b5e",border\_width=2, fg\_color="transparent", command=Re\_Show\_pass)**

**re\_show\_btn.place(x = 304, y=0)**

**re\_pass\_Entry.place(x = 25, y = 210+80)**

**phonnumber\_Entry = CTkEntry(form\_frm, 350, placeholder\_text="Phone Number")**

**phonnumber\_Entry.place(x = 25, y =250+80)**

**Create\_acc\_btn = CTkButton(form\_frm, width = 350, text="Create Account", corner\_radius=100, command = NullCheck)**

**Create\_acc\_btn.place(x = 25, y = 290+80)**

**tempxpos = 100**

**tempypos = 410**

**Dnt\_hv\_acc\_lbl = CTkLabel(form\_frm, text="Already have an account ? ")**

**Dnt\_hv\_acc\_lbl.place(x = tempxpos, y = tempypos)**

**Sign\_up\_lbl = CTkLabel(form\_frm, text="Sign In", font = ("Arial" , 12, "italic", "underline"))**

**Sign\_up\_lbl.place(x = tempxpos + 150, y = tempypos)**

**#=>3------Sign In Page --------------------------------------**

**global PG\_Sign\_in**

**def PG\_Sign\_in():**

**root.title ("http:www.HADAirlineManagementSystem.com/Sign\_In")**

**for i in Main\_fame.winfo\_children():**

**i.destroy()**

**Stack.Push(3)**

**form\_frm\_width = 400**

**form\_frm\_height = 340**

**form\_frm = CTkFrame(Main\_fame, width=form\_frm\_width, height=form\_frm\_height)**

**form\_frm.place(x = (m\_r\_width/(2))-(form\_frm\_width/2),**

**y = (m\_r\_height/(2)-(form\_frm\_height/2))**

**)**

**dummy\_back\_btn = CTkButton(Main\_fame,text="Back", command = GoBack)**

**dummy\_back\_btn.place(x =10, y = 10)**

**login\_lb = CTkLabel(form\_frm, text = "LOGIN")**

**login\_lb.place(x = 10, y = 10)**

**def Login\_authentication(\_username, \_pass):**

**global \_isSignedIn, User**

**\_isSignedIn = True**

**\_username= \_username.get()**

**\_pass = crypt(\_pass.get()).encrypt()**

**if (\_username == "" or \_pass == "" ) :**

**errorLabeling(form\_frm, "Feilds Cannot Be Empty", \_x = 90, \_y = 150)**

**else:**

**temp\_qry = f"SELECT U\_name, U\_Gmail, U\_password, U\_isAdmin FROM user\_details WHERE U\_name = '{\_username}' or U\_Gmail = '{\_username}' and U\_password = '{\_pass}'"**

**cur.execute(temp\_qry)**

**qry\_result = cur.fetchone()**

**if qry\_result == None :**

**errorLabeling(form\_frm, "Username Or Password Is Incorrect", \_x = 50, \_y = 150)**

**else:**

**global User, isAdmin**

**User = qry\_result[0]**

**\_isSignedIn = True**

**isAdmin = qry\_result[-1]**

**if is\_flight\_details\_obtained == True :**

**PG\_Payment()**

**else:**

**PG\_Get\_Flight\_Details()**

**user\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text= "Username/Gmail")**

**user\_Entry.place(x = 25, y = 40)**

**def Show\_pass():**

**if pass\_Entry.cget('show') == '\*':**

**pass\_Entry.configure(show='')**

**show\_btn.configure(text=" Hide ")**

**else:**

**pass\_Entry.configure(show='\*')**

**show\_btn.configure(text="Show")**

**pass\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text="Password", show = "\*")**

**show\_btn = CTkButton(pass\_Entry, width = 22, height=28, text="Show",border\_color="#565b5e",border\_width=2, fg\_color="transparent", command=Show\_pass)**

**show\_btn.place(x = 304, y=0)**

**pass\_Entry.place(x = 25, y = 80)**

**Login\_btn = CTkButton(form\_frm, text= "LOGIN", width= 350, corner\_radius=100, command=lambda :(Login\_authentication(user\_Entry, pass\_Entry)))**

**Login\_btn.place(x = 25, y = 120)**

**No\_of\_hyphen = 41**

**line\_lbl = CTkLabel(form\_frm, text = f"{'-'\*No\_of\_hyphen} OR {'-'\*No\_of\_hyphen}")**

**line\_lbl.place(x = 25, y = 170)**

**def loginWithGoogle():**

**temp\_TL = CTkToplevel(root)**

**temp\_TL.title("http:www.HADAirlineManagementSystem.com/Error?loginWithGoogle")**

**temp\_TL.attributes("-topmost", True)**

**temp\_TL.geometry("400x300")**

**errorLabeling(temp\_TL, \_text="""**

**We are Extremly Sorry for the Inconvenience:**

**We at HAD are currently working to**

**bring in that feature.""",\_textcolor = "#007acc", \_cooldowntime = None)**

**Login\_btn2 = CTkButton(form\_frm, text= "LOGIN with Google", width= 350, corner\_radius=100, command=loginWithGoogle)**

**Login\_btn2.place(x = 25, y = 220)**

**tempxpos = 100**

**tempypos = 270**

**Dnt\_hv\_acc\_lbl = CTkLabel(form\_frm, text="Don't have an account ? ")**

**Dnt\_hv\_acc\_lbl.place(x = tempxpos, y = tempypos)**

**Sign\_up\_lbl = CTkLabel(form\_frm, text="Sign Up", font = ("Arial" , 12, "italic", "underline"))**

**Sign\_up\_lbl.place(x = tempxpos + 140, y = tempypos)**

**Sign\_up\_lbl.bind("<Button-1>", lambda event, : PG\_Sign\_Up())**

**Sign\_up\_lbl.bind("<Enter>", lambda event, lbl = Sign\_up\_lbl: lbl.configure(text\_color = "#007acc"))**

**Sign\_up\_lbl.bind("<Leave>", lambda event, lbl = Sign\_up\_lbl: lbl.configure(text\_color = "Light Gray"))**

**# =>2------Show Flights details --------------------------------------**

**global PG\_search\_flight\_**

**def PG\_search\_flight\_():**

**Stack.Push(2)**

**root.title ("http:www.HADAirlineManagementSystem.com/Search\_flights")**

**if div\_frame.winfo\_exists():**

**for i in div\_frame.winfo\_children():**

**i.destroy()**

**div\_frame.destroy()**

**temp\_frm\_width = 900**

**temp\_frm\_height = 550**

**global fligt\_search\_result\_frm**

**fligt\_search\_result\_frm = CTkScrollableFrame (Main\_fame, width=temp\_frm\_width, height = temp\_frm\_height, scrollbar\_button\_color= None)**

**fligt\_search\_result\_frm.place(x = (m\_r\_width/(2))-(temp\_frm\_width/2),**

**y = (m\_r\_height/(2)-(temp\_frm\_height/2))**

**)**

**PG\_Heading = CTkLabel(fligt\_search\_result\_frm, text = "AVAILABLE FLIGHTS")**

**PG\_Heading.pack(padx = 25, pady = 10, anchor = "w")**

**dummy\_back\_btn = CTkButton(Main\_fame,text="Back", command = GoBack)**

**dummy\_back\_btn.place(x =10, y = 10)**

**btns = []**

**global radio\_val, dest\_airport, origin\_airport**

**temp\_qry = "SELECT F\_Departure, F\_Arrival, F\_Airline, F\_price, F\_ID FROM flights WHERE F\_Departure = '" + origin\_airport+ "' AND F\_Arrival = '" + dest\_airport+ "';"**

**cur.execute(temp\_qry)**

**result = cur.fetchall()**

**btn\_data= {}**

**FLIGHT\_ID\_dict = {}**

**key = 1**

**for i in result :**

**btn\_data[key] = f"{i[0]} -----> {i[1]} : {i[2]} : {i[3]}"**

**FLIGHT\_ID\_dict[key] = i[4]**

**key += 1**

**def on\_btn\_click(index):**

**global Flight\_ID**

**Flight\_ID = FLIGHT\_ID\_dict[index]**

**if \_isSignedIn == True:**

**PG\_Payment()**

**else :**

**PG\_Sign\_in()**

**pass**

**for id, label in btn\_data.items() :**

**btn = CTkButton(fligt\_search\_result\_frm, text = label,bg\_color="transparent", command = lambda idx=id: on\_btn\_click(idx)).pack(pady = 10, padx = 10, anchor = "w")**

**btns.append(btn)**

**con.commit()**

**Main\_frm\_Authentication\_Btns()**

**# =>1-----Get Flight details---------------------------------------------------------------------**

**def PG\_Get\_Flight\_Details():**

**global div\_frame, \_isSignedIn, User**

**for i in Main\_fame.winfo\_children():**

**i.destroy()**

**Stack.Push(1)**

**temp\_frm\_width = 500**

**temp\_frm\_height = 320**

**div\_frame = CTkFrame(Main\_fame,width=temp\_frm\_width, height = temp\_frm\_height)**

**div\_frm\_xpos = 75**

**din\_frm\_widget\_width = 350**

**div\_frame.place(x = (m\_r\_width/(2))-(temp\_frm\_width/2),**

**y = (m\_r\_height/(2)-(temp\_frm\_height/2)))**

**def Func\_radio\_btn():**

**global radio\_val**

**radio\_val = book\_a\_fligt\_radio\_val.get()**

**if radio\_val == "ReturnRadio":**

**Dest\_Date\_Btn.configure(state = tk.NORMAL)**

**else:**

**Dest\_Date\_Btn.configure(state = tk.DISABLED)**

**global book\_a\_fligt\_radio\_val**

**book\_a\_fligt\_radio\_val = StringVar(value = "other")**

**rd\_btn\_y\_pos = 20**

**return\_radio\_btn = createRadioButton(div\_frame,"Return","ReturnRadio",book\_a\_fligt\_radio\_val,Func\_radio\_btn,div\_frm\_xpos, rd\_btn\_y\_pos)**

**one\_way\_radio\_btn = createRadioButton(div\_frame,"One Way", "OnewayRadio", book\_a\_fligt\_radio\_val,Func\_radio\_btn, div\_frm\_xpos+130, rd\_btn\_y\_pos)**

**def Combo\_get\_origin\_val(origin\_combo\_value):**

**global origin\_airport**

**origin\_airport = origin\_combo\_value**

**departure\_place = StringVar(value="dep\_combo\_other")**

**departure\_place.set("Departure")**

**cur.execute("SELECT F\_Departure AS combined\_column FROM flights UNION SELECT F\_Arrival AS combined\_column FROM flights")**

**airports = []**

**for i in cur.fetchall():**

**for j in i :**

**airports.append(j)**

**Origin\_Airport = CTkComboBox(div\_frame,width=din\_frm\_widget\_width,**

**values=airports,**

**variable= departure\_place, command = Combo\_get\_origin\_val)**

**Origin\_Airport.place (x = div\_frm\_xpos, y = rd\_btn\_y\_pos+50)**

**def Combo\_get\_dest\_val(dest\_combo\_value):**

**global dest\_airport**

**dest\_airport = dest\_combo\_value**

**global arrival\_place**

**arrival\_place = StringVar(value="des\_combo\_other")**

**arrival\_place.set("Destination")**

**Dest\_Airport = CTkComboBox(div\_frame,width=din\_frm\_widget\_width,**

**values=airports,variable= arrival\_place, command= Combo\_get\_dest\_val)**

**Dest\_Airport.place (x = div\_frm\_xpos, y = rd\_btn\_y\_pos+82)**

**arrival\_place = arrival\_place.get()**

**def Dept\_open\_date\_picker():**

**top = CTkToplevel(root)**

**top.title("Select a Date")**

**top.attributes("-topmost", True)**

**cal = Calendar(top, selectmode='day', date\_pattern = "yyyy-mm-dd")**

**cal.pack(pady=10)**

**def select\_date():**

**global Dept\_selected\_date**

**Dept\_selected\_date = cal.get\_date()**

**Dept\_date\_label.configure(text=f"Selected Date : {Dept\_selected\_date}")**

**top.destroy()**

**select\_button = CTkButton(top, text="Select Date", command=select\_date)**

**select\_button.pack(pady=10)**

**Dept\_Date\_Btn = CTkButton(div\_frame, text="Departure Date", command=Dept\_open\_date\_picker, corner\_radius=100)**

**Dept\_Date\_Btn.place(x=div\_frm\_xpos, y =rd\_btn\_y\_pos+114)**

**Dept\_date\_label = CTkLabel(div\_frame, text= "Select Date")**

**Dept\_date\_label.place(x = div\_frm\_xpos+170, y = rd\_btn\_y\_pos+114)**

**def Dest\_open\_date\_picker():**

**top = CTkToplevel(root)**

**top.title("Select a Date")**

**top.attributes("-topmost", True)**

**cal = Calendar(top, selectmode='day', date\_pattern = "yyyy-mm-dd")**

**cal.pack(pady=10)**

**def select\_date():**

**global Dest\_selected\_date**

**Dest\_selected\_date = cal.get\_date()**

**Dest\_date\_label.configure(text=f"Selected Date : {Dest\_selected\_date}")**

**top.destroy()**

**select\_button = CTkButton(top, text="Select Date", command=select\_date)**

**select\_button.pack(pady=10)**

**Dest\_Date\_Btn = CTkButton(div\_frame, text="Arrival Date", command=Dest\_open\_date\_picker, corner\_radius=100, state=tk.DISABLED)**

**Dest\_Date\_Btn.place(x=div\_frm\_xpos, y =rd\_btn\_y\_pos+147)**

**Dest\_date\_label = CTkLabel(div\_frame, text= "Select Date")**

**Dest\_date\_label.place(x = div\_frm\_xpos+170, y = rd\_btn\_y\_pos+147)**

**def Null\_Check():**

**error\_name = ""**

**try :**

**global radio\_val, dest\_airport, origin\_airport**

**radio\_val**

**dest\_airport**

**origin\_airport**

**Dept\_selected\_date**

**if radio\_val == "ReturnRadio" :**

**Dest\_selected\_date**

**if dest\_airport not in globals() or origin\_airport not in globals() or radio\_val not in globals():**

**pass**

**except NameError :**

**error\_name = "NameError"**

**error\_label = CTkLabel(div\_frame, text = "Feilds Cannot Be Empty", font = ("Bradley Hand ITC" , 18, "italic", "bold"), text\_color = "red")**

**error\_label.place(x = 140, y = 238)**

**def refresh():**

**error\_label.destroy()**

**error\_label.after(3000, refresh)**

**finally:**

**if error\_name != "NameError":**

**global is\_flight\_details\_obtained**

**is\_flight\_details\_obtained = True**

**PG\_search\_flight\_()**

**Search\_Fligths\_btn = CTkButton(div\_frame, text = "Search Fligths", width = din\_frm\_widget\_width, corner\_radius=75, command=lambda :(Null\_Check()))**

**Search\_Fligths\_btn.place(x = div\_frm\_xpos, y =rd\_btn\_y\_pos + 180)**

**con.commit()**

**Main\_frm\_Authentication\_Btns()**

**# =>-----MAIN FRAME---------------------------------------------------------------------**

**global Main\_frm\_Authentication\_Btns**

**def Main\_frm\_Authentication\_Btns():**

**if \_isSignedIn == True:**

**global User**

**User\_Label = CTkLabel(Main\_fame, text = f"{User}", text\_color= "#007acc", font= ("Bradley Hand ITC" , 18, "italic", "bold"), width = 20)**

**User\_Label.place(x = m\_r\_width-200, y= 10)**

**User\_Label.update\_idletasks()**

**lbl\_width = User\_Label.winfo\_width()**

**User\_Label.place(x = (m\_r\_width/(1.2))-(lbl\_width/2),y = 10)**

**def on\_select(option):**

**global \_isSignedIn, User , isAdmin**

**if option == "Cancel Flights":**

**Setting\_btn.set("Settings")**

**tmp\_root = CTkToplevel(root)**

**tmp\_root\_width , tmp\_root\_height = 400, 300**

**tmp\_root.geometry(f"{tmp\_root\_width}x{tmp\_root\_height}")**

**tmp\_root.attributes("-topmost", True)**

**temp\_frame = CTkFrame(tmp\_root, width= tmp\_root\_width, height = tmp\_root\_height, border\_color= "#007acc", border\_width= 2)**

**temp\_frame.pack()**

**Flight\_ID\_entry = CTkEntry(temp\_frame, placeholder\_text= "Ticket ID",width= 350)**

**Flight\_ID\_entry.place(x = (tmp\_root\_width/(2))-(350/2),**

**y = (tmp\_root\_height/(2)-(28/2))-38)**

**def Cancel\_tickets():**

**Flight\_id = Flight\_ID\_entry.get()**

**cur.execute("SELECT BID, BU\_NAME FROM booking WHERE BID = %s AND IS\_ACTIVE = 1", Flight\_id)**

**row = cur.fetchone()**

**global User**

**if row and row[1] == User :**

**cur.execute("UPDATE booking SET IS\_ACTIVE = 0 WHERE BID = %s AND IS\_ACTIVE = 1", Flight\_id)**

**lbl = CTkLabel(temp\_frame, text ="Successfully Canceled", text\_color="green", font = ("Bradley Hand ITC", 18, "italic", "bold"))**

**lbl.place(x = 110, y = (tmp\_root\_height/(2)-(28/2)+38))**

**def w8():**

**tmp\_root.destroy()**

**con.commit()**

**lbl.after(3000,w8)**

**else:**

**errorLabeling(temp\_frame, "Flight ID is Incorect", \_x = 110, \_y = (tmp\_root\_height/(2)-(28/2)+38))**

**Cancel\_btn = CTkButton(temp\_frame, text="Cancel Flight", width = 350 , corner\_radius= 100, command= Cancel\_tickets)**

**Cancel\_btn.place(x = (tmp\_root\_width/(2))-(350/2),**

**y = (tmp\_root\_height/(2)-(28/2)))**

**if option == "Booking History" :**

**Setting\_btn.set("Settings")**

**tmp\_root = CTkToplevel(root)**

**tmp\_root\_width , tmp\_root\_height = 900, 300**

**tmp\_root.geometry(f"{tmp\_root\_width}x{tmp\_root\_height}")**

**tmp\_root.attributes("-topmost", True)**

**tmp\_root.resizable(False, False)**

**temp\_frame = CTkScrollableFrame(tmp\_root, width= tmp\_root\_width, height = tmp\_root\_height, border\_color= "#007acc", border\_width= 2)**

**temp\_frame.pack()**

**global User**

**cur.execute("SELECT BU\_NAME, BID, F\_Departure, F\_Arrival, F\_Airline, F\_price FROM booking B, flights F WHERE B.BU\_NAME = %s AND B.B\_FLIGHT = F.F\_ID AND B.IS\_ACTIVE = 1", User)**

**row = cur.fetchall()**

**for i in row :**

**CTkLabel(temp\_frame, text = i).pack(padx = 20, pady = 5, anchor = "w")**

**if option == "Edit Account" :**

**Setting\_btn.set("Settings")**

**tmp\_root = CTkToplevel(root)**

**tmp\_root\_width , tmp\_root\_height = 400, 600**

**tmp\_root.geometry(f"{tmp\_root\_width}x{tmp\_root\_height}")**

**tmp\_root.attributes("-topmost", True)**

**tmp\_root.resizable(False, False)**

**form\_frm\_width = 400**

**form\_frm\_height = 600**

**form\_frm = CTkFrame(tmp\_root, width=form\_frm\_width, height=form\_frm\_height)**

**form\_frm.place(x = 0, y = 0)**

**def NullCheck():**

**global DOB\_selected\_date, cal**

**F\_name = F\_name\_Entry.get()**

**L\_name = L\_name\_Entry.get()**

**U\_name = U\_name\_Entry.get()**

**\_Gender = Gender.get()**

**if "cal" in globals():**

**dob = cal.get\_date()**

**dob\_dt = datetime.strptime(dob, "%Y-%m-%d")**

**today = datetime.today()**

**global Age**

**Age = today.year - dob\_dt.year**

**if (today.month, today.day) > (dob\_dt.month, dob\_dt.day):**

**Age +=1**

**else :**

**dob = ""**

**Gmail = gmail\_Entry.get()**

**\_pass = crypt(pass\_Entry.get()).encrypt()**

**\_re\_pass = crypt(re\_pass\_Entry.get()).encrypt()**

**phonenumber = phonnumber\_Entry.get()**

**if \_pass != \_re\_pass:**

**errorLabeling(form\_frm, "Passwords Don't Match", \_x = 110, \_y = 410)**

**if F\_name != "" :**

**cur.execute("UPDATE user\_details set UF\_name = %s", F\_name)**

**if L\_name != "" :**

**cur.execute("UPDATE user\_details set UL\_name = %s", L\_name)**

**if U\_name != "" :**

**cur.execute("UPDATE user\_details set U\_name = %s", U\_name)**

**if \_Gender != "other" :**

**cur.execute("UPDATE user\_details set U\_gender = %s", \_Gender)**

**if phonenumber != "" :**

**cur.execute("UPDATE user\_details set U\_phno = %s", phonenumber)**

**if \_pass != "" :**

**cur.execute("UPDATE user\_details set U\_password = %s", \_pass)**

**if Gmail != "" :**

**cur.execute("UPDATE user\_details set U\_Gmail = %s", Gmail)**

**if dob != "" :**

**cur.execute("UPDATE user\_details set U\_dob = %s", dob)**

**cur.execute("UPDATE user\_details set U\_AGE = %s", Age)**

**if F\_name == "" and L\_name == "" and U\_name == "" and dob == "" and Gmail == "" and \_pass == "" and \_re\_pass == "" and phonenumber == "" and \_Gender == "other" :**

**errorLabeling(form\_frm, "Feilds Cannot Be Null", \_x = 110, \_y = 410)**

**else :**

**lbl = CTkLabel(form\_frm, text= "Succesfully Updated", text\_color= "green", font=("Bradley Hand ITC" , 18, "italic", "bold"))**

**lbl.place(x = 110, y = 410)**

**def w8():**

**tmp\_root.destroy()**

**lbl.after(3000, w8)**

**con.commit()**

**cur.execute("SELECT UF\_name, UL\_name, U\_name, U\_Gmail, U\_phno, U\_password, U\_dob, U\_gender, U\_AGE FROM user\_details WHERE U\_name = %s", User)**

**result = cur.fetchone()**

**F\_name\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text=result[0])**

**F\_name\_Entry.place(x = 25, y = 10)**

**L\_name\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text=result[1])**

**L\_name\_Entry.place(x = 25, y = 50)**

**def Func\_radio\_btn():**

**global \_Gender**

**\_Gender = Gender.get()**

**global Gender**

**Gender = StringVar(value = "other")**

**rd\_btn\_y\_pos = 90**

**male\_radio\_btn = createRadioButton(form\_frm, "Male","M",Gender,Func\_radio\_btn,25, rd\_btn\_y\_pos)**

**female\_radio\_btn = createRadioButton(form\_frm,"Female", "F", Gender,Func\_radio\_btn, 25+130, rd\_btn\_y\_pos)**

**other\_radio\_btn = createRadioButton(form\_frm,"Other", "O", Gender,Func\_radio\_btn, 25+130 + 130, rd\_btn\_y\_pos)**

**global cal**

**def DOB\_open\_date\_picker():**

**top = CTkToplevel(form\_frm)**

**top.title("Select a Date")**

**top.attributes("-topmost", True)**

**global cal**

**cal = Calendar(top, selectmode='day', date\_pattern = "yyyy-mm-dd")**

**cal.pack(pady=10)**

**def select\_date():**

**global DOB\_selected\_date**

**DOB\_selected\_date = cal.get\_date()**

**DOB\_Date\_label.configure(text=f"Selected Date : {DOB\_selected\_date}")**

**top.destroy()**

**select\_button = CTkButton(top, text="Select Date", command=select\_date)**

**select\_button.pack(pady=10)**

**DOB\_Date\_Btn = CTkButton(form\_frm, text="Date Of Birth", command=DOB\_open\_date\_picker, corner\_radius=100)**

**DOB\_Date\_Btn.place(x=25, y =rd\_btn\_y\_pos+40)**

**DOB\_Date\_label = CTkLabel(form\_frm, text= "Select Date")**

**DOB\_Date\_label.place(x = 25+170, y = 90+40)**

**U\_name\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text=result[2])**

**U\_name\_Entry.place(x = 25, y = 130+40)**

**gmail\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text=result[3])**

**gmail\_Entry.place(x = 25, y = 170+40)**

**def Show\_pass():**

**if pass\_Entry.cget('show') == '\*':**

**pass\_Entry.configure(show='')**

**show\_btn.configure(text=" Hide ")**

**else:**

**pass\_Entry.configure(show='\*')**

**show\_btn.configure(text="Show")**

**def Re\_Show\_pass():**

**if re\_pass\_Entry.cget('show') == '\*':**

**re\_pass\_Entry.configure(show='')**

**re\_show\_btn.configure(text=" Hide ")**

**else:**

**re\_pass\_Entry.configure(show='\*')**

**re\_show\_btn.configure(text="Show")**

**pass\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text="Password", show = "\*")**

**show\_btn = CTkButton(pass\_Entry, width = 22, height=28, text="Show",border\_color="#565b5e",border\_width=2, fg\_color="transparent", command=Show\_pass)**

**show\_btn.place(x = 304, y=0)**

**pass\_Entry.place(x = 25, y = 170+80)**

**re\_pass\_Entry = CTkEntry(form\_frm, width = 350, placeholder\_text="Re-Password", show = "\*")**

**re\_show\_btn = CTkButton(re\_pass\_Entry, width = 22, height=28, text="Show",border\_color="#565b5e",border\_width=2, fg\_color="transparent", command=Re\_Show\_pass)**

**re\_show\_btn.place(x = 304, y=0)**

**re\_pass\_Entry.place(x = 25, y = 210+80)**

**phonnumber\_Entry = CTkEntry(form\_frm, 350, placeholder\_text=result[4])**

**phonnumber\_Entry.place(x = 25, y =250+80)**

**Create\_acc\_btn = CTkButton(form\_frm, width = 350, text="Alter Account", corner\_radius=100, command = NullCheck)**

**Create\_acc\_btn.place(x = 25, y = 290+80)**

**temp\_frame2 = CTkFrame(form\_frm, width = 100, height= 200, border\_color= "light Grey", border\_width= 2,fg\_color= "transparent")**

**lbl = CTkLabel(temp\_frame2, text="Enter The Feilds That You Want to Change",fg\_color= "transparent")**

**lbl.pack()**

**def onhover(event):**

**temp\_frame2.place(x = 25, y =410+40)**

**info\_Btn = CTkButton(form\_frm, text = "i", width= 10, height = 5, corner\_radius= 100, hover= "on\_hover")**

**info\_Btn.place(x = 25, y = 370+40)**

**info\_Btn.bind("<Enter>", onhover)**

**info\_Btn.bind("<Leave>", lambda event : temp\_frame2.place\_forget())**

**if option == "Logout":**

**\_isSignedIn = False**

**User = ""**

**isAdmin = False**

**PG\_Get\_Flight\_Details()**

**if option == "Adiministrate":**

**temp\_TL = CTkToplevel(root)**

**temp\_TL.title("http:www.HADAirlineManagementSystem.com/Admin")**

**temp\_TL.attributes("-topmost", True)**

**temp\_TL.geometry("400x300")**

**def allhistory():**

**temp\_TL.destroy()**

**Setting\_btn.set("Settings")**

**tmp\_root = CTkToplevel(root)**

**tmp\_root\_width , tmp\_root\_height = 900, 300**

**tmp\_root.geometry(f"{tmp\_root\_width}x{tmp\_root\_height}")**

**tmp\_root.attributes("-topmost", True)**

**tmp\_root.resizable(False, False)**

**temp\_frame = CTkScrollableFrame(tmp\_root, width= tmp\_root\_width, height = tmp\_root\_height, border\_color= "#007acc", border\_width= 2)**

**temp\_frame.pack()**

**global User**

**cur.execute("SELECT BID, BU\_NAME, B\_FLIGHT FROM booking ")**

**row = cur.fetchall()**

**for i in row :**

**CTkLabel(temp\_frame, text = f"{i[0]} --> {i[1]} --> {i[2]}").pack(padx = 20, pady = 5, anchor = "w")**

**def addAdmin():**

**temp\_TL.destroy()**

**temp\_TL2 = CTkToplevel(root)**

**temp\_TL2.title("http:www.HADAirlineManagementSystem.com/Admin")**

**temp\_TL2.attributes("-topmost", True)**

**temp\_TL2.geometry("400x300")**

**def add():**

**user = user\_Entry.get()**

**cur.execute("UPDATE user\_details SET U\_isAdmin = 1 WHERE U\_name = %s", user)**

**con.commit()**

**if cur.execute("SELECT U\_isAdmin from user\_details WHERE U\_name = %s", user)==1:**

**label = CTkLabel(temp\_TL2, text= "Successfully Updated", font= ("Bradley Hand ITC" , 18, "italic", "bold"), text\_color= "green")**

**label.place(x = 110, y = 178)**

**def dest():**

**label.destroy()**

**temp\_TL2.destroy()**

**label.after(3000, dest)**

**else:**

**label = CTkLabel(temp\_TL2, text= "Username Not Found Or Server Down ", font= ("Bradley Hand ITC" , 18, "italic", "bold"), text\_color= "red")**

**label.place(x = 45, y = 178)**

**def dest():**

**label.destroy()**

**label.after(3000, dest)**

**user\_Entry = CTkEntry(temp\_TL2, placeholder\_text= "UserName To Change As Admin", width=240)**

**user\_Entry.place(x = 80, y = 100)**

**user\_Entry\_btn = CTkButton(temp\_TL2, text= " Change", command=add)**

**user\_Entry\_btn.place(x = 130, y = 138)**

**def delAdmin():**

**temp\_TL.destroy()**

**temp\_TL2 = CTkToplevel(root)**

**temp\_TL2.title("http:www.HADAirlineManagementSystem.com/Admin")**

**temp\_TL2.attributes("-topmost", True)**

**temp\_TL2.geometry("400x300")**

**def \_del():**

**user = user\_Entry.get()**

**cur.execute("UPDATE user\_details SET U\_isActive = 0 WHERE U\_name = %s", user)**

**con.commit()**

**if cur.execute("SELECT U\_isActive from user\_details WHERE U\_name = %s", user)==0:**

**label = CTkLabel(temp\_TL2, text= "Username Not Found Or Server Down ", font= ("Bradley Hand ITC" , 18, "italic", "bold"), text\_color= "red")**

**label.place(x = 45, y = 178)**

**def dest():**

**label.destroy()**

**label.after(3000, dest)**

**else:**

**label = CTkLabel(temp\_TL2, text= "Successfully Deleted", font= ("Bradley Hand ITC" , 18, "italic", "bold"), text\_color= "green")**

**label.place(x = 110, y = 178)**

**def dest():**

**label.destroy()**

**temp\_TL2.destroy()**

**label.after(3000, dest)**

**user\_Entry = CTkEntry(temp\_TL2, placeholder\_text= "UserName To Delete From Admin", width=240)**

**user\_Entry.place(x = 80, y = 100)**

**user\_Entry\_btn = CTkButton(temp\_TL2, text= " Change", command=\_del)**

**user\_Entry\_btn.place(x = 130, y = 138)**

**def addFlights():**

**temp\_TL.destroy()**

**temp\_TL2 = CTkToplevel(root)**

**temp\_TL2.title("http:www.HADAirlineManagementSystem.com/Admin")**

**temp\_TL2.attributes("-topmost", True)**

**temp\_TL2.geometry("400x300")**

**def add():**

**dept = Dept\_Entry.get()**

**arr = Arr\_Entry.get()**

**Air = Air\_Entry.get()**

**price = price\_Entry.get()**

**cur.execute("SELECT COUNT(\*) FROM flights")**

**tempCount = cur.fetchone()[0]**

**if dept ==""or arr ==""or Air ==""or price =="":**

**errorLabeling(temp\_TL2, "Fields Cannot Be Empty", \_x = 30, \_y = 190+35)**

**elif not price.isdigit():**

**errorLabeling(temp\_TL2, "Check The Price", \_x = 30, \_y = 190+35)**

**else:**

**cur.execute("INSERT INTO flights(F\_Departure,F\_Arrival,F\_Airline, F\_price ) VALUES (%s, %s,%s,%s)", (dept, arr, Air, int(price)))**

**con.commit()**

**cur.execute("SELECT COUNT(\*) FROM flights")**

**check\_Count = cur.fetchone()[0]**

**if check\_Count ==tempCount+1:**

**label = CTkLabel(temp\_TL2, text= "Successfully Added", font= ("Bradley Hand ITC" , 18, "italic", "bold"), text\_color= "green")**

**label.place(x = 110, y = 190+35)**

**def dest():**

**label.destroy()**

**temp\_TL2.destroy()**

**label.after(3000, dest)**

**else:**

**label = CTkLabel(temp\_TL2, text= " Error Occured While Inserting ", font= ("Bradley Hand ITC" , 18, "italic", "bold"), text\_color= "red")**

**label.place(x = 45, y = 190+35)**

**def dest():**

**label.destroy()**

**label.after(3000, dest)**

**tempy = 50**

**Dept\_Entry = CTkEntry(temp\_TL2, placeholder\_text= "Depature Place", width=240)**

**Dept\_Entry.place(x = 80, y = tempy)**

**Arr\_Entry = CTkEntry(temp\_TL2, placeholder\_text= "Arrival Place", width=240)**

**Arr\_Entry.place(x = 80, y = tempy+35)**

**Air\_Entry = CTkEntry(temp\_TL2, placeholder\_text= "Airline Name", width=240)**

**Air\_Entry.place(x = 80, y = tempy+35+35)**

**price\_Entry = CTkEntry(temp\_TL2, placeholder\_text= "Price", width=240)**

**price\_Entry.place(x = 80, y = tempy+35+35+35)**

**add\_Flight\_btn = CTkButton(temp\_TL2, text="Add", command=add)**

**add\_Flight\_btn.place(x = 125, y = tempy+35+35+35+35)**

**def delflights():**

**temp\_TL.destroy()**

**temp\_TL2 = CTkToplevel(root)**

**temp\_TL2.title("http:www.HADAirlineManagementSystem.com/Admin")**

**temp\_TL2.attributes("-topmost", True)**

**temp\_TL2.geometry("760x300")**

**def \_del():**

**flightId = FlightIDEntry.get()**

**if flightId =="":**

**errorLabeling(temp\_TL2, "Feild Empty", \_x = 345, \_y = 1)**

**elif flightId.isalpha():**

**errorLabeling(temp\_TL2, "Invalid ID", \_x = 346, \_y = 1)**

**else:**

**cur.execute("SELECT F\_ID FROM flights WHERE F\_IsActive = 1")**

**result = cur.fetchall()**

**if (int(flightId),) not in result:**

**errorLabeling(temp\_TL2, "ID Not Found", \_x = 344, \_y = 1)**

**else:**

**cur.execute("UPDATE flights SET F\_IsActive = 0 WHERE F\_ID = %s", flightId)**

**con.commit()**

**label = CTkLabel(temp\_TL2, text= "Successfully Deleted", font= ("Bradley Hand ITC" , 18, "italic", "bold"), text\_color= "green")**

**label.place(x = 336, y = 1)**

**def dest():**

**label.destroy()**

**temp\_TL2.destroy()**

**label.after(3000, dest)**

**tempx, tempy = 10, 30**

**FlightIDEntry = CTkEntry(temp\_TL2, placeholder\_text="Flight ID")**

**FlightIDEntry.place(x = tempx+240, y = tempy)**

**mod\_btn = CTkButton(temp\_TL2, text="Modify", command=\_del)**

**mod\_btn.place(x = tempx+390, y = tempy)**

**sckrlble\_frame = CTkScrollableFrame(temp\_TL2, width = 720, height=220)**

**sckrlble\_frame.place(x = tempx, y = tempy+35)**

**cur.execute("SELECT \* FROM Flights WHERE F\_IsActive = 1")**

**result = cur.fetchall()**

**for i in result:**

**lbl = CTkLabel(sckrlble\_frame, text=f"{i[0]} {i[1]} {i[2]} {i[3]} {i[4]}", text\_color= "Light Gray")**

**lbl.pack(padx = 50,pady = 10, anchor = "w")**

**def ModFlights():**

**temp\_TL.destroy()**

**temp\_TL2 = CTkToplevel(root)**

**temp\_TL2.title("http:www.HADAirlineManagementSystem.com/Admin")**

**temp\_TL2.attributes("-topmost", True)**

**temp\_TL2.geometry("760x300")**

**def mod():**

**flightId = FlightIDEntry.get()**

**if flightId =="":**

**errorLabeling(temp\_TL2, "Feild Empty", \_x = 345, \_y = 1)**

**elif flightId.isalpha():**

**errorLabeling(temp\_TL2, "Invalid ID", \_x = 346, \_y = 1)**

**else:**

**cur.execute("SELECT F\_ID FROM flights WHERE F\_IsActive = 1")**

**result = cur.fetchall()**

**if (int(flightId),) not in result:**

**errorLabeling(temp\_TL2, "ID Not Found", \_x = 344, \_y = 1)**

**else:**

**temp\_TL2.destroy()**

**temp\_TL3 = CTkToplevel(root)**

**temp\_TL3.title("http:www.HADAirlineManagementSystem.com/Admin")**

**temp\_TL3.attributes("-topmost", True)**

**temp\_TL3.geometry("400x300")**

**tempx, tempy = 10 , 30**

**cur.execute("SELECT \* FROM flights WHERE F\_ID = %s", flightId)**

**result = cur.fetchone()**

**def Check():**

**Dept = Dept\_Entry.get()**

**Arr = Arr\_Entry.get()**

**Air = Air\_Entry.get()**

**price = price\_Entry.get()**

**if Dept != "":**

**cur.execute("UPDATE flights SET F\_Departure = %s WHERE F\_ID =%s", (Dept,flightId))**

**if Arr != "":**

**cur.execute("UPDATE flights SET F\_Arrival = %s WHERE F\_ID =%s", (Arr,flightId) )**

**if Air != "":**

**cur.execute("UPDATE flights SET F\_Airline = %s WHERE F\_ID =%s", (Air,flightId) )**

**if price != "":**

**cur.execute("UPDATE flights SET F\_price = %s WHERE F\_ID =%s", (price,flightId))**

**con.commit()**

**label = CTkLabel(temp\_TL3, text= "Successfully Updated", font= ("Bradley Hand ITC" , 18, "italic", "bold"), text\_color= "green")**

**label.pack(pady = 5)**

**def dest():**

**label.destroy()**

**temp\_TL3.destroy()**

**label.after(3000, dest)**

**Dept\_Entry = CTkEntry(temp\_TL3, placeholder\_text=result[1],width = 240)**

**Dept\_Entry.pack(pady = 5)**

**Arr\_Entry = CTkEntry(temp\_TL3, placeholder\_text=result[2],width = 240)**

**Arr\_Entry.pack(pady = 5)**

**Air\_Entry = CTkEntry(temp\_TL3, placeholder\_text=result[3],width = 240)**

**Air\_Entry.pack(pady = 5)**

**price\_Entry = CTkEntry(temp\_TL3, placeholder\_text=result[4],width = 240)**

**price\_Entry.pack(pady = 5)**

**Upd\_BTN = CTkButton(temp\_TL3, text="Update",width = 240, command=Check)**

**Upd\_BTN.pack(pady = 5)**

**tempx, tempy = 10, 30**

**FlightIDEntry = CTkEntry(temp\_TL2, placeholder\_text="Flight ID")**

**FlightIDEntry.place(x = tempx+240, y = tempy)**

**mod\_btn = CTkButton(temp\_TL2, text="Modify", command=mod)**

**mod\_btn.place(x = tempx+390, y = tempy)**

**sckrlble\_frame = CTkScrollableFrame(temp\_TL2, width = 720, height=220)**

**sckrlble\_frame.place(x = tempx, y = tempy+35)**

**cur.execute("SELECT \* FROM Flights WHERE F\_IsActive = 1")**

**result = cur.fetchall()**

**for i in result:**

**lbl = CTkLabel(sckrlble\_frame, text=f"{i[0]} {i[1]} {i[2]} {i[3]} {i[4]}", text\_color= "Light Gray")**

**lbl.pack(padx = 50,pady = 10, anchor = "w")**

**tempx = 55**

**tempy = 150**

**Add\_Admin\_btn = CTkButton(temp\_TL, text="Add Admin", command=addAdmin)**

**Add\_Admin\_btn.place(x = tempx, y = tempy-30)**

**Add\_Flight\_btn = CTkButton(temp\_TL, text="Add Flights", command=addFlights)**

**Add\_Flight\_btn.place(x = tempx+150, y = tempy-30)**

**Rem\_Flight\_btn = CTkButton(temp\_TL, text="Delete Flights", command= delflights)**

**Rem\_Flight\_btn.place(x = tempx+150, y = tempy+10)**

**Rem\_Admin\_btn = CTkButton(temp\_TL, text="Delete Admin", command=delAdmin)**

**Rem\_Admin\_btn.place(x = tempx, y = tempy+10)**

**Mod\_Flight\_btn = CTkButton(temp\_TL, text="Modify Flights", command= ModFlights)**

**Mod\_Flight\_btn.place(x = tempx, y = tempy+50)**

**all\_booking\_btn = CTkButton(temp\_TL, text="All History", command=allhistory)**

**all\_booking\_btn.place(x = tempx+150, y = tempy+50)**

**if isAdmin==1:**

**option = ["Cancel Flights", "Booking History", "Edit Account","Adiministrate", "Logout"]**

**else:**

**option = ["Cancel Flights", "Booking History", "Edit Account", "Logout"]**

**Setting\_btn = CTkOptionMenu(Main\_fame, values= option, command= on\_select)**

**Setting\_btn.place(x = ((m\_r\_width/(1.2))-(lbl\_width/2)) +lbl\_width+ 10, y = 10)**

**Setting\_btn.set("Settings")**

**else:**

**temp\_xpos = m\_r\_width-300**

**Sign\_in\_btn = CTkButton(Main\_fame, text = "Sign In", command= PG\_Sign\_in)**

**Sign\_in\_btn.place(x = temp\_xpos, y = 10)**

**Sign\_up\_btn = CTkButton(Main\_fame, text = "Sign Up", fg\_color="transparent", border\_color= "grey", border\_width=2, command=PG\_Sign\_Up)**

**Sign\_up\_btn.place(x = temp\_xpos+150, y = 10)**

**Main\_frm\_Authentication\_Btns()**

**PG\_Get\_Flight\_Details()**

**#----------------------------------------------------------------------------------**

**root.mainloop()**

**cur.close()**

**con.commit()**

**Crypt.py**

**import string**

**import random**

**chars = " " + string.digits + string.ascii\_letters**

**chars = list(chars)**

**key = ['d', 'H', 'v', 'b', 'r', '7', 'T', 'S', 'M', 'E', 'W', 'A', 'K', 'e', 'i', 'C', 'G', 'w', ' ', 'P', 'y', 'x', 'R', 'q', 'U', 'B', 'g', 'O', 'k', 'N', '0', 't', 'u', 'X', 'L', '4', '8', 'n', 'Q', 'J', 'h', 'o', 'V', '5', 'I', 'l', 's', '9', 'z', 'f', 'Z', 'F', 'c', '1', 'p', 'j', 'Y', 'D', 'm', '3', '6', '2', 'a']**

**class crypt :**

**#ENCRYPT**

**def \_\_init\_\_(self, \_text):**

**self.text = \_text**

**def encrypt(self):**

**cipher\_text = ""**

**for letter in self.text:**

**if letter in string.punctuation :**

**cipher\_text += letter**

**else :**

**index = chars.index(letter)**

**cipher\_text += key[index]**

**return cipher\_text**

**#DECRYPT**

**def decrypt(self):**

**plain\_text = ""**

**if type(self.text) == list or type(self.text) == tuple :**

**for word in self.text:**

**for letter in word:**

**if letter in string.punctuation :**

**plain\_text += letter**

**else :**

**index = key.index(letter)**

**plain\_text += chars[index]**

**plain\_text+=" "**

**return plain\_text**

**else :**

**for letter in self.text:**

**if letter in string.punctuation :**

**plain\_text += letter**

**else :**

**index = key.index(letter)**

**plain\_text += chars[index]**

**return plain\_text**

**Ticket\_Code\_Gen.py**

**from datetime import datetime**

**from string import \***

**from random import \***

**def Gen\_Code():**

**date\_time = datetime. now()**

**todays\_date = date\_time.date()**

**print(todays\_date)**

**todays\_date = list(str(todays\_date))**

**corrected\_date = ""**

**for i in todays\_date:**

**if i in punctuation :**

**pass**

**else :**

**corrected\_date = corrected\_date + i**

**#print(corrected\_date)**

**corrected\_date = list(corrected\_date)**

**yyyy = corrected\_date[0:4]**

**mm = corrected\_date[4:6]**

**dd = corrected\_date[6:8]**

**corrected\_date = []**

**corrected\_date.extend(dd)**

**corrected\_date.extend(mm)**

**corrected\_date.extend(yyyy)**

**corrected\_date\_ = ""**

**for i in corrected\_date :**

**corrected\_date\_ = corrected\_date\_ + i**

**print(corrected\_date\_)**

**sec = list(ascii\_letters)**

**#print(sec)**

**rand\_elem = sample(sec, 8)**

**rand\_code = "".join(map(str,rand\_elem))**

**#print(rand\_code)**

**ticket\_code = corrected\_date\_ + rand\_code**

**return ticket\_code**

**def return\_date\_time(millisecond = False):**

**if millisecond == True :**

**now\_time = str(datetime. now())**

**return now\_time**

**else:**

**now\_time = str(datetime. now()).split(".")**

**return now\_time[0]**

**#print(Gen\_Code())**

**#print( return\_date\_time())**

**13.TABLE DESIGN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DATABASE | TABLE | COLUMN | DATATYPE | CONSTRAINT |
|  | | | | |
| airwaysms2\_0 | user\_details | UID | INT | Primary Key, Auto Increment |
| UF\_name | VARCHAR(100) |  |
| UL\_name | VARCHAR(100) |  |
| U\_name | VARCHAR(100) | Unique |
| U\_Gmail | VARCHAR(100) |  |
| U\_phno | VARCHAR(12) |  |
| U\_password | VARCHAR(100) |  |
| U\_dob | DATE |  |
| U\_AGE | INT |  |
| U\_gender | VARCHAR(5) |  |
| U\_isActive | TINYINT |  |
| U\_isAdmin | TINYINT |  |
|  | | | |
| flights | F\_ID | INT | Primary Key, Auto Increment |
| F\_Departure | VARCHAR(100) |  |
| F\_Arrival | VARCHAR(100) |  |
| F\_Airline | VARCHAR(45) |  |
| F\_price | INT |  |
|  | | | |
| booking | BID | VARCHAR(100) | Primary Key, Auto Increment |
| BU\_NAME | VARCHAR(100) |  |
| B\_FLIGHT | INT | Foreign Key, `flights` (`F\_ID`) |
| IS\_ACTIVE | INT |  |
|  | | | |
| payment | PID | INT | Primary Key, Auto Increment |
| P\_UID | INT | Foreign Key, `user\_details` (`UID`) |
| AMOUNT | INT |  |
| P\_STATUS | INT |  |
| P\_ACC\_NUM | INT |  |
| P\_UPI\_NUM | INT |  |
| P\_METHOD | VARCHAR(45) |  |

|  |
| --- |
|  |

**14. Conclusion**

As the aviation industry continues to evolve, the need for more dynamic and adaptable airways management systems will only grow, particularly with the expansion of unmanned aircraft systems and the implementation of next generation air traffic control technologies. Moving forward, investment in innovation and infrastructure will be essential to maintaining the safety, security, and sustainability of air travel, while also addressing environmental concerns through more efficient routing and flight management strategies. Overall, the continuous development of airways management systems will remain central to the evolution of global aviation.

**15. Future Enhancement**

**Potential Enhancements and Security Considerations**

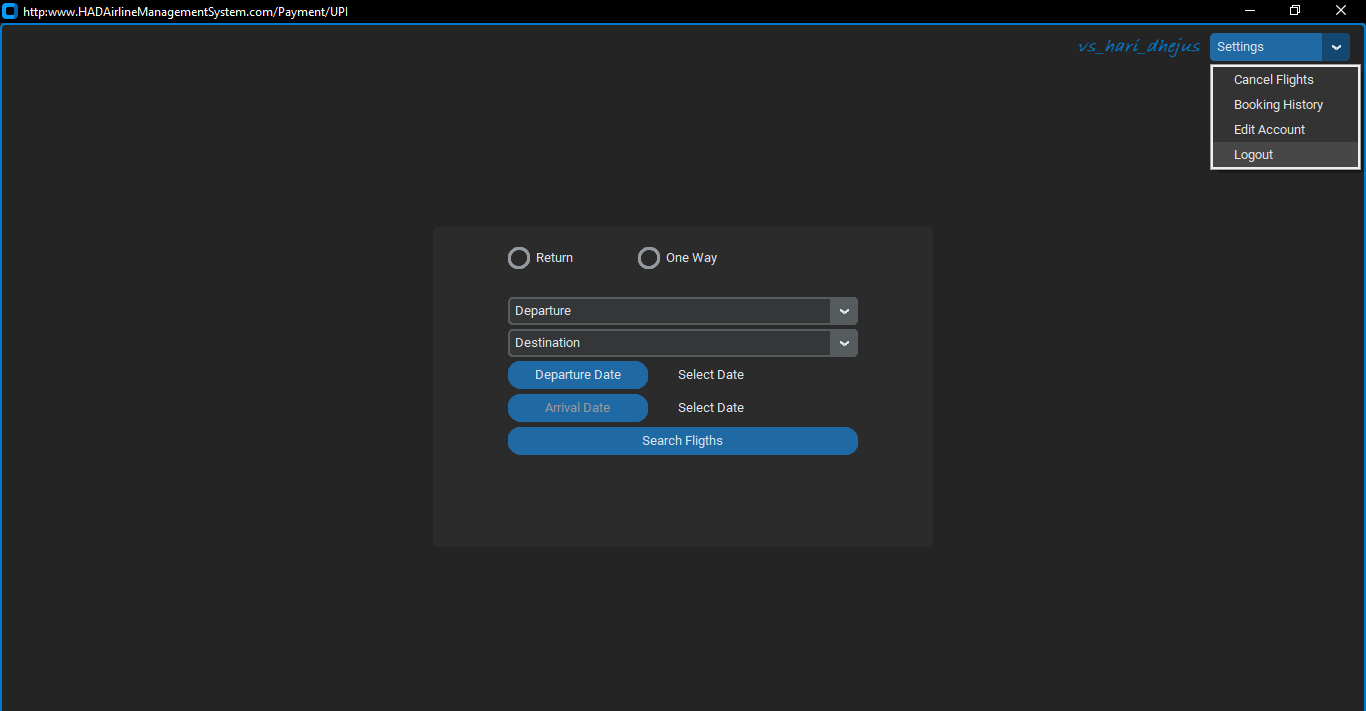
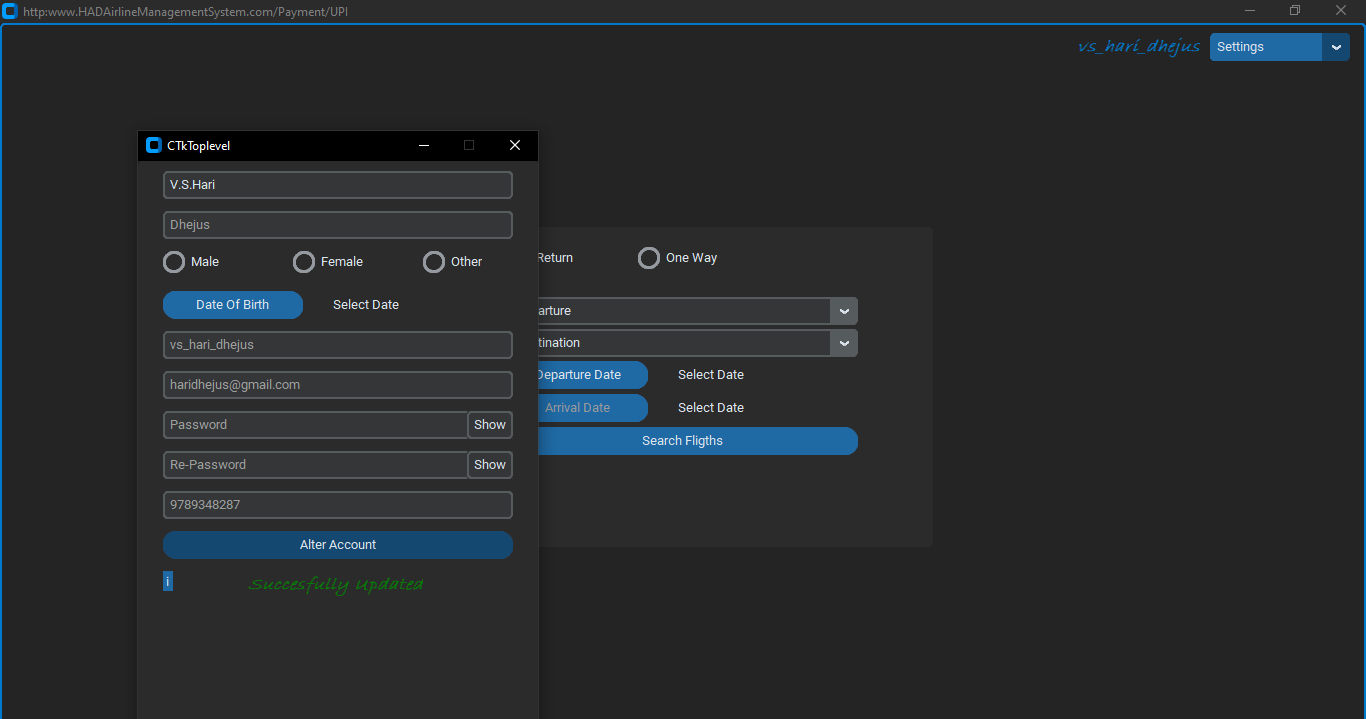
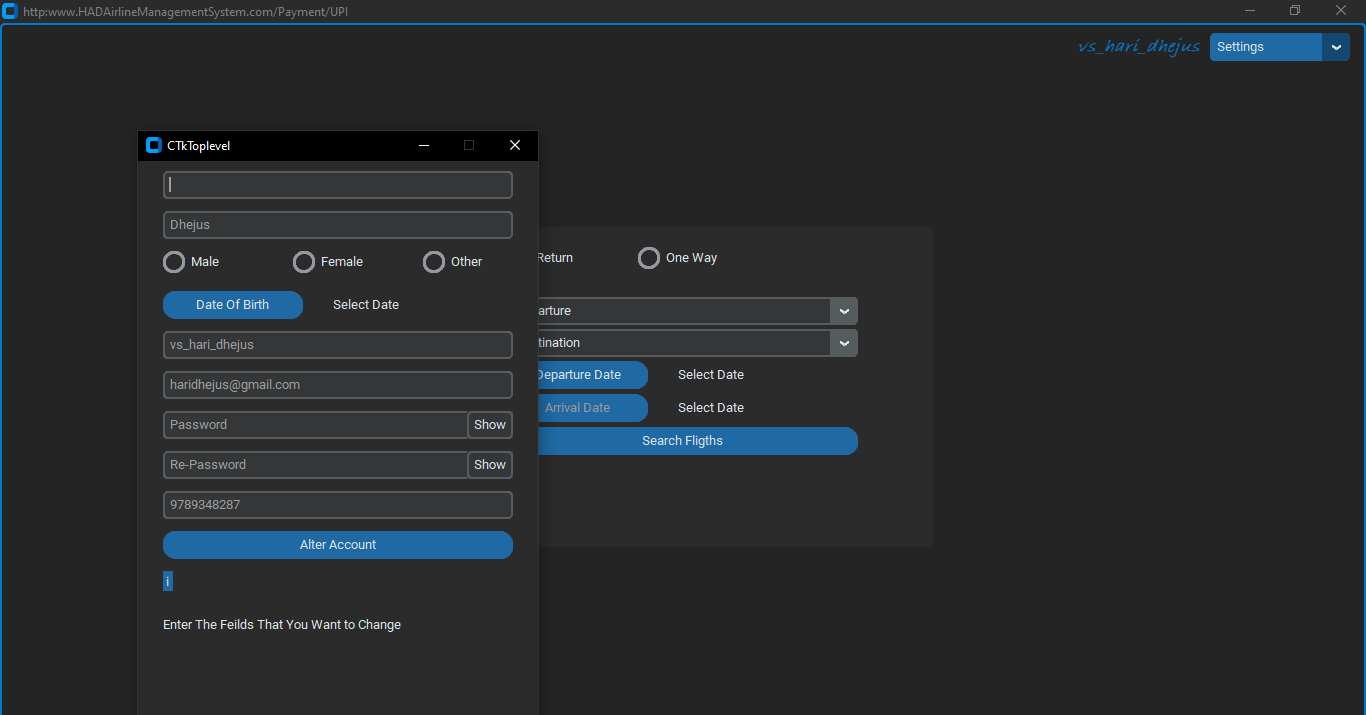
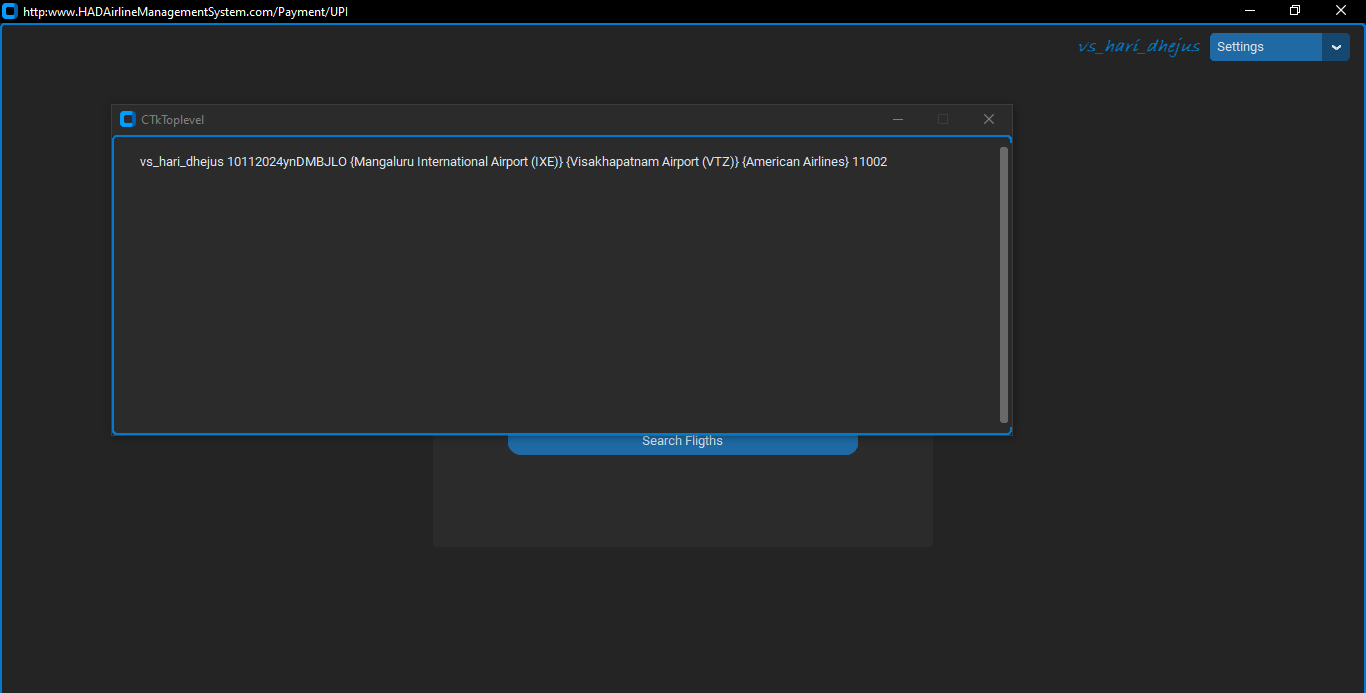
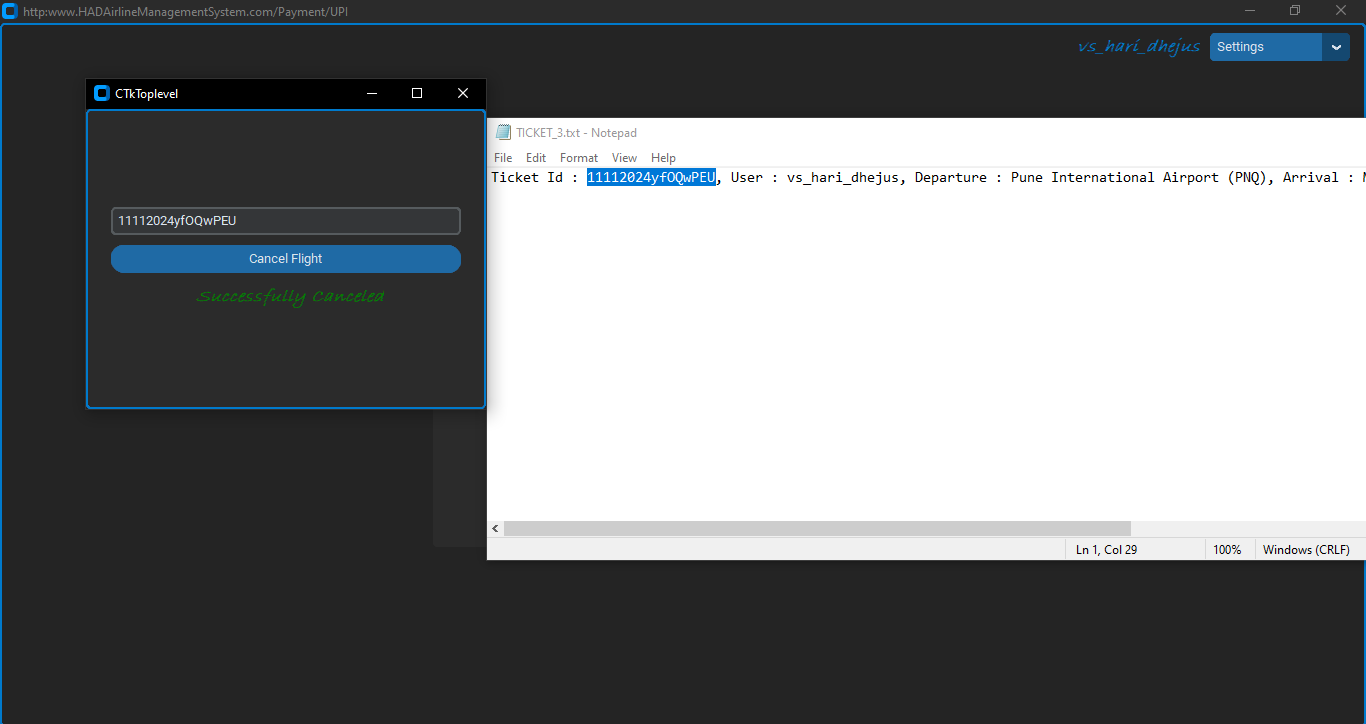
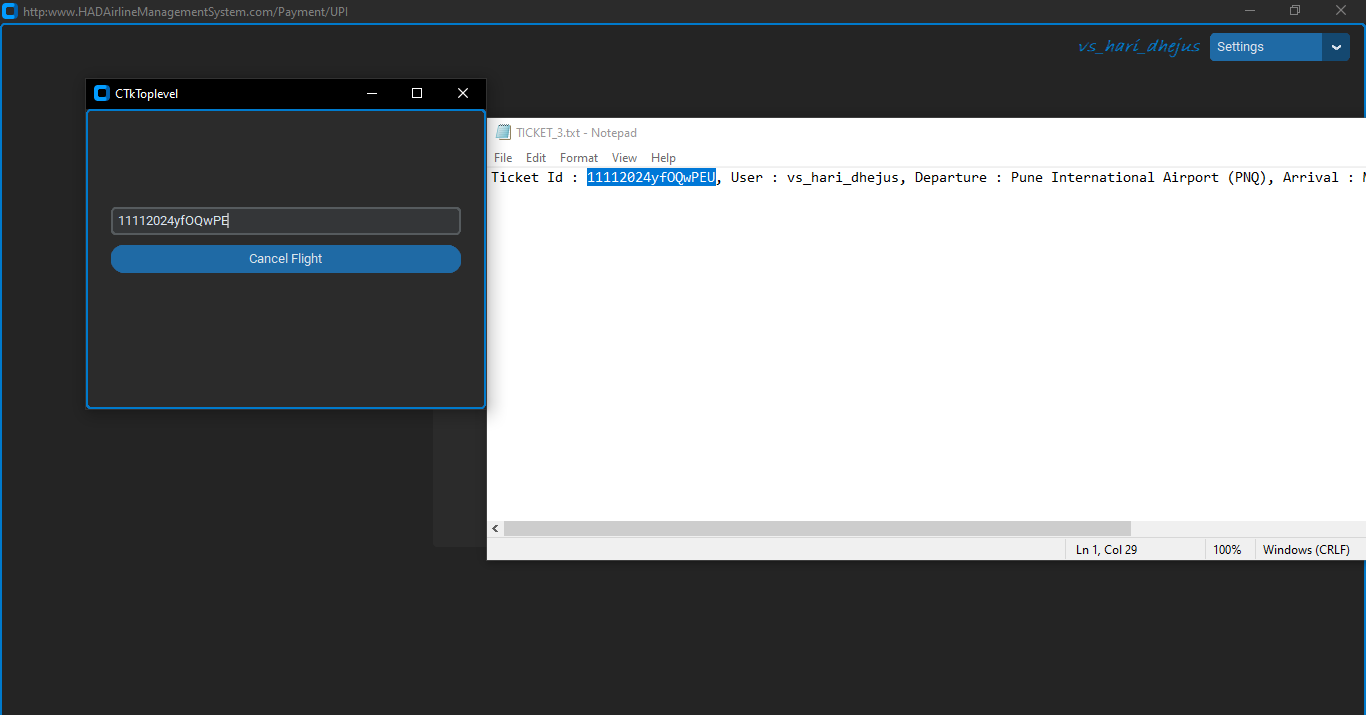
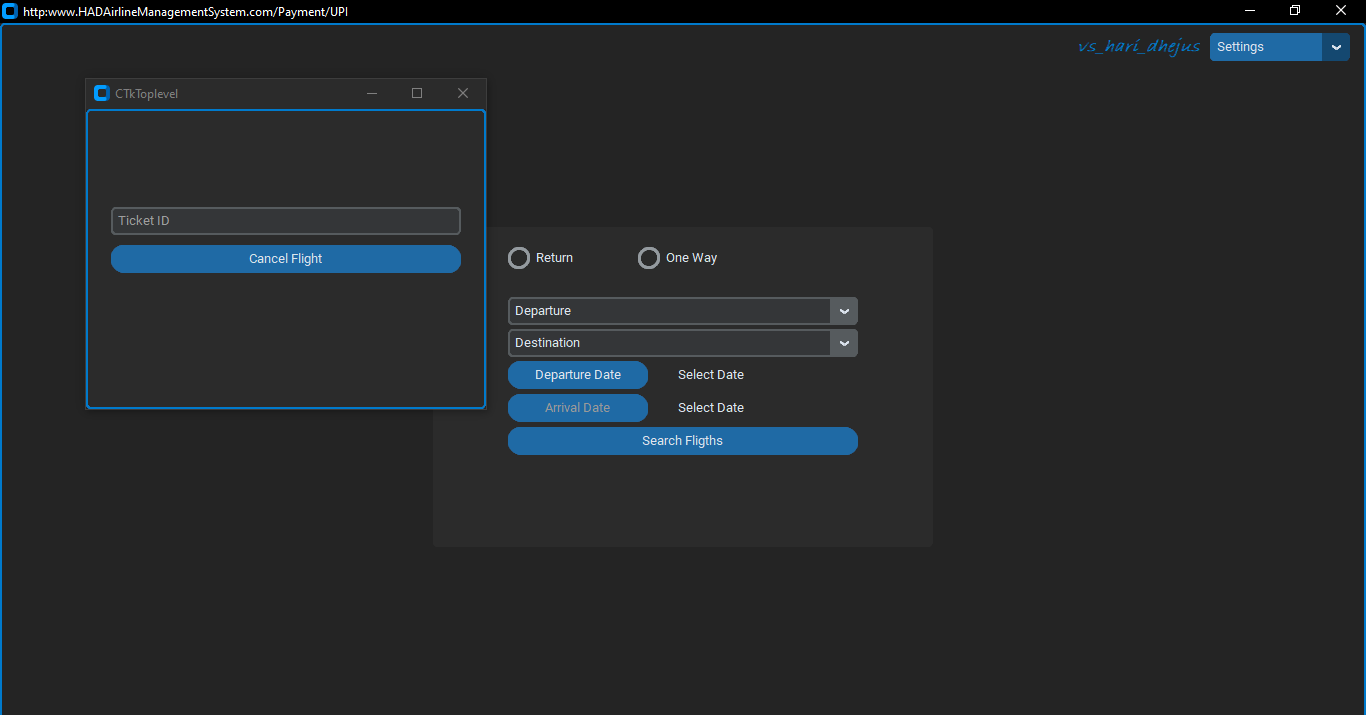
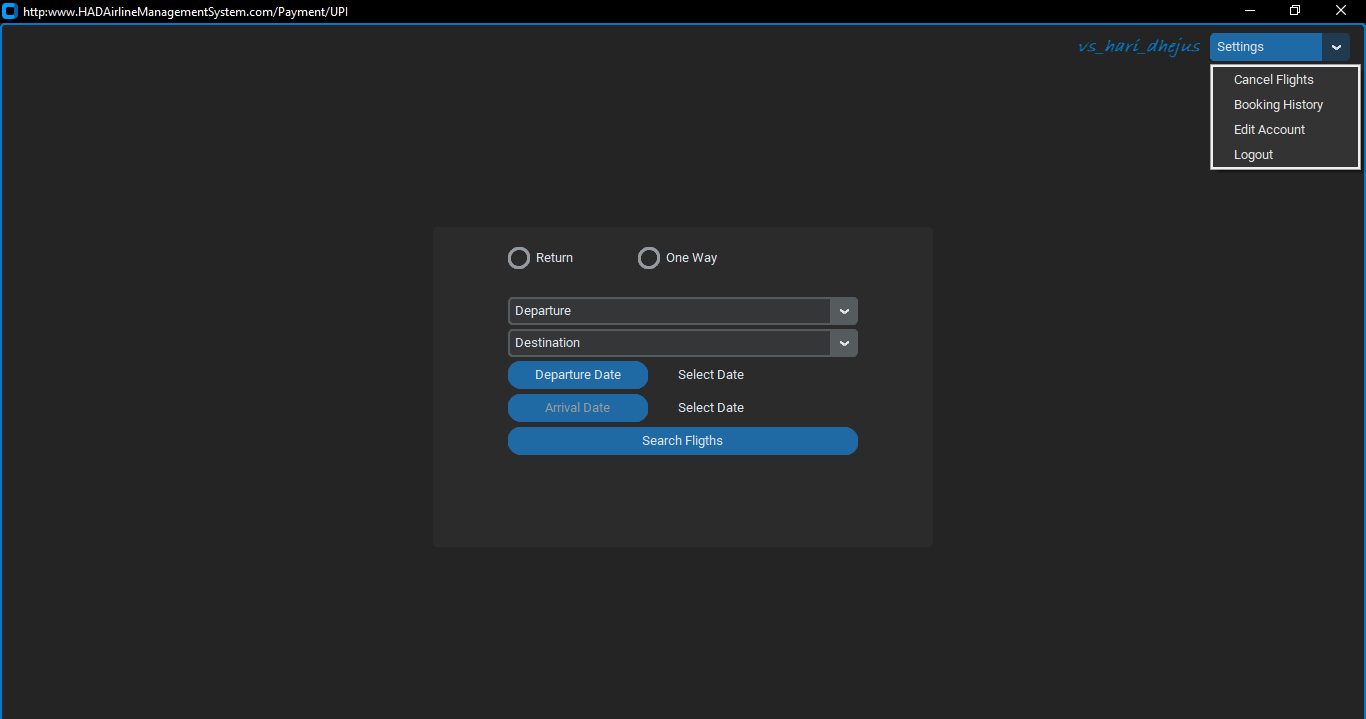
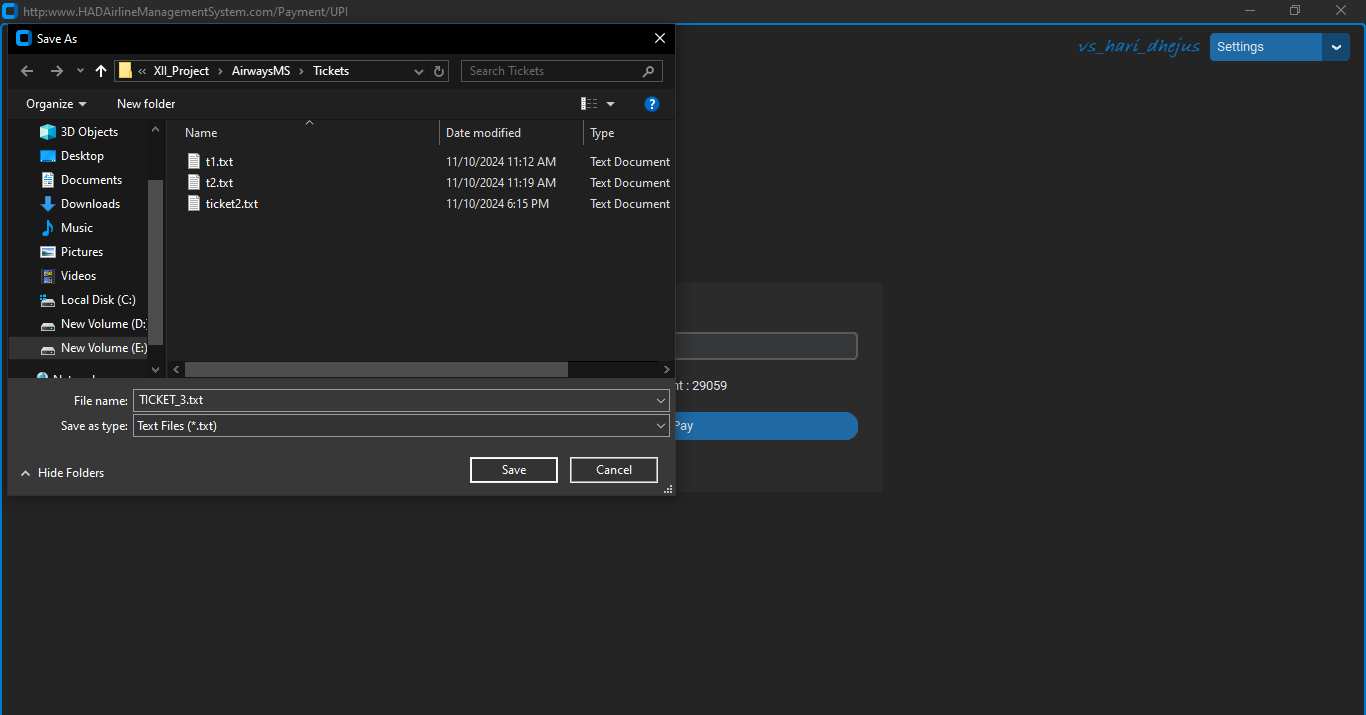
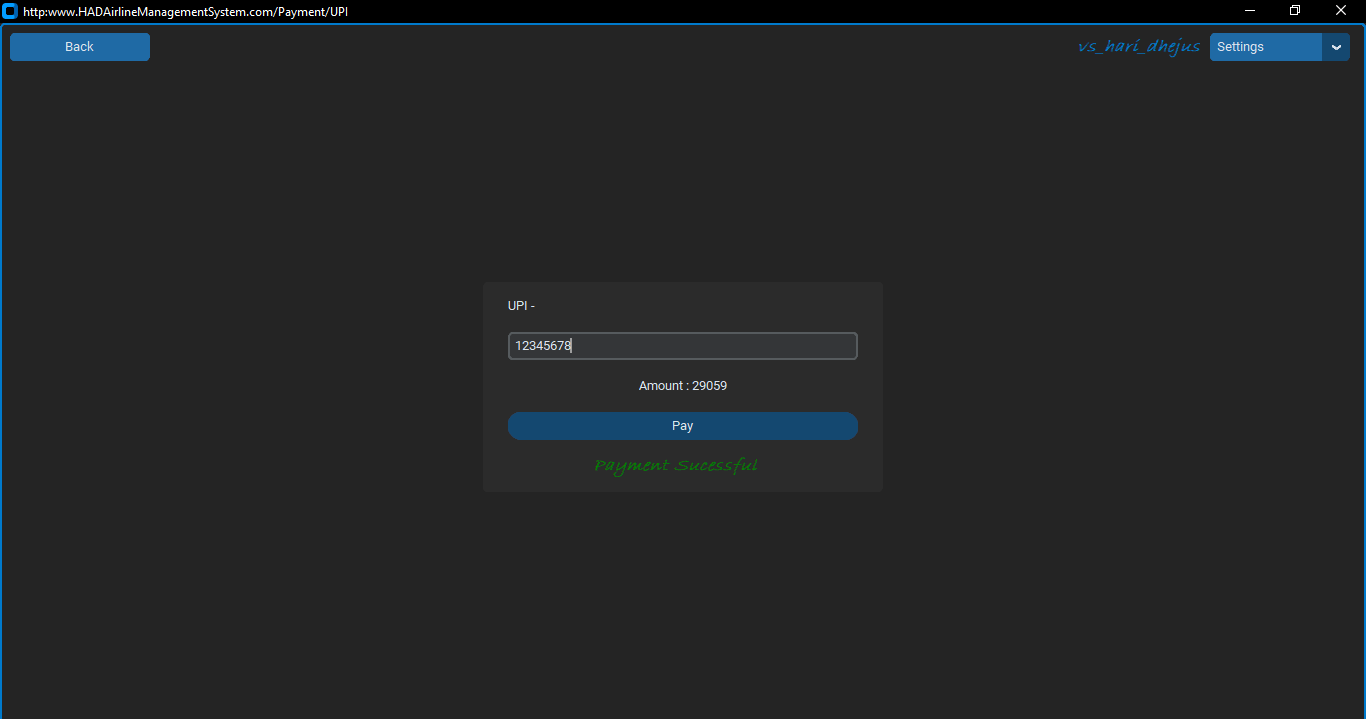
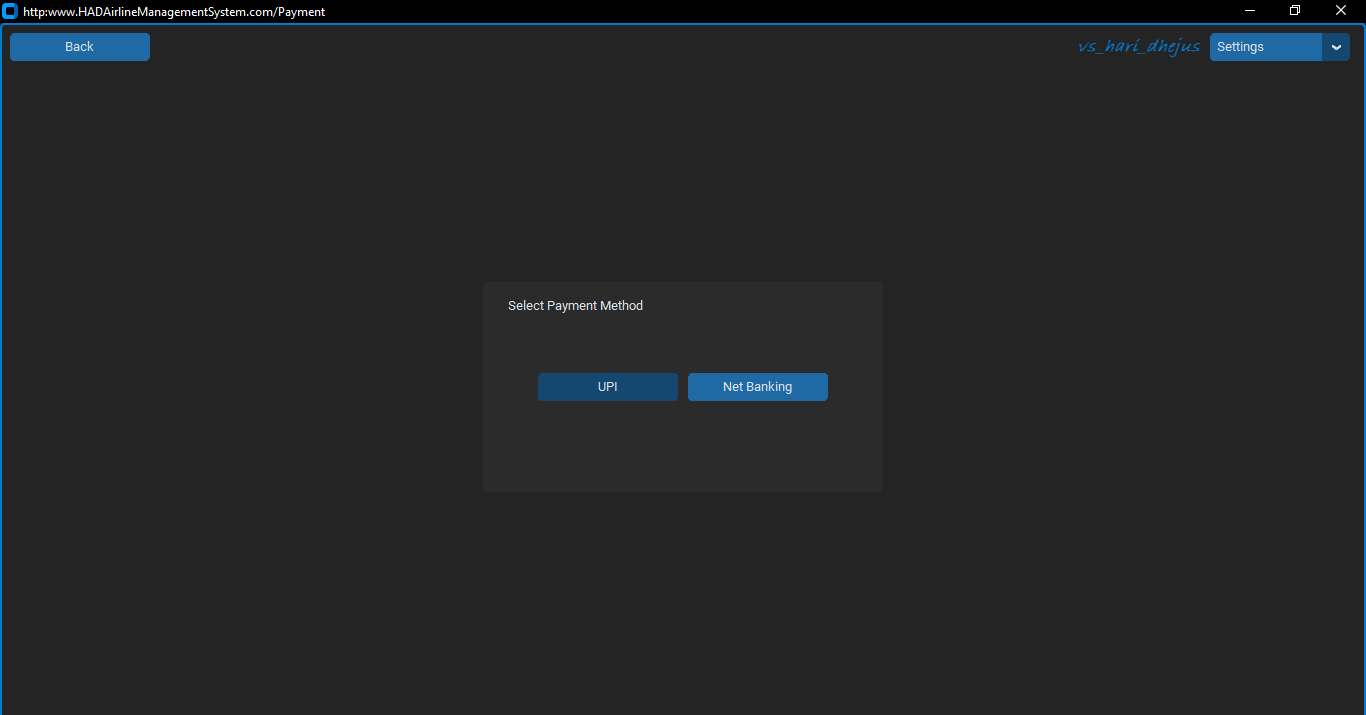
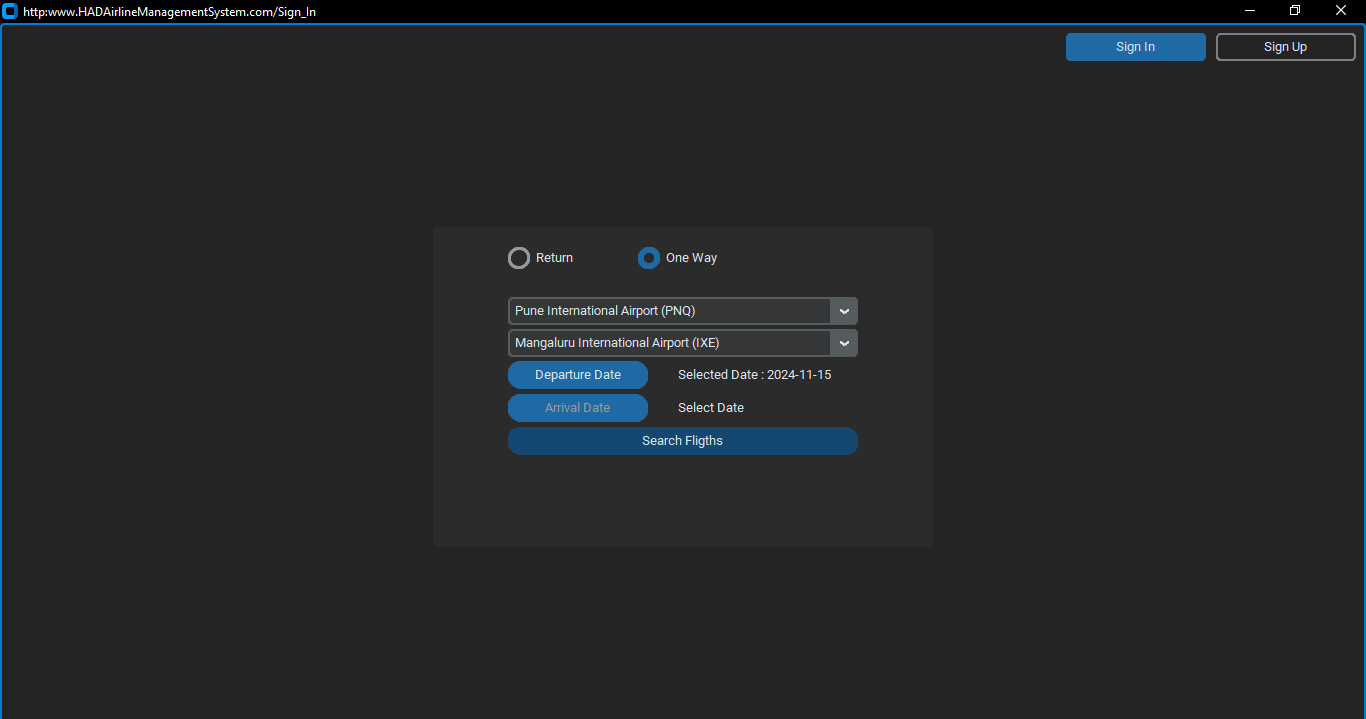
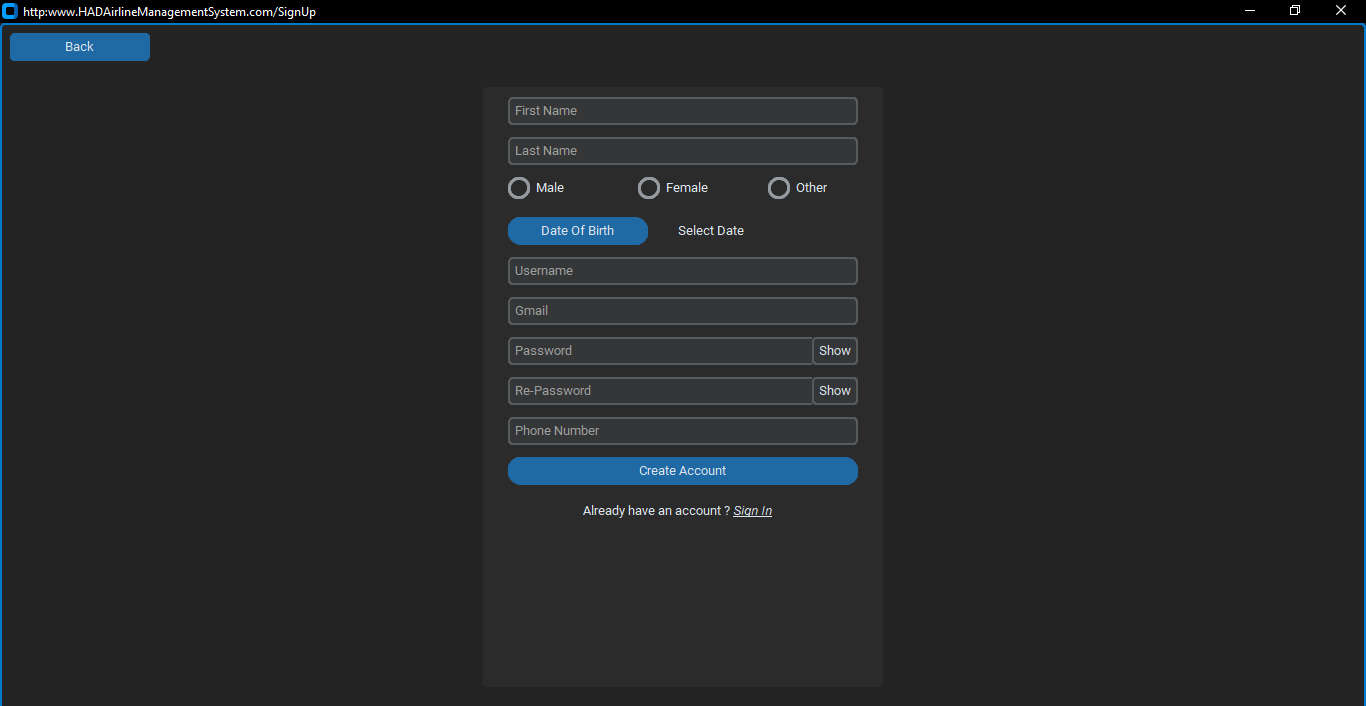
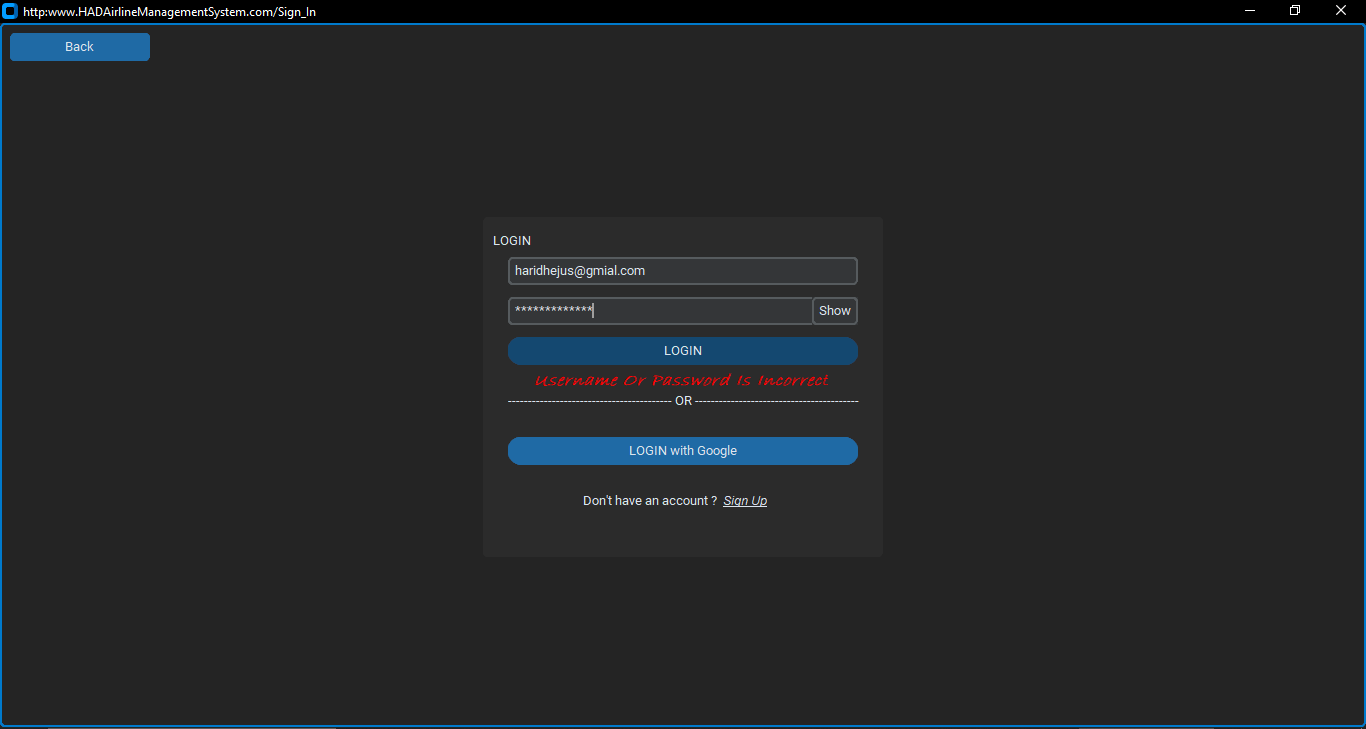
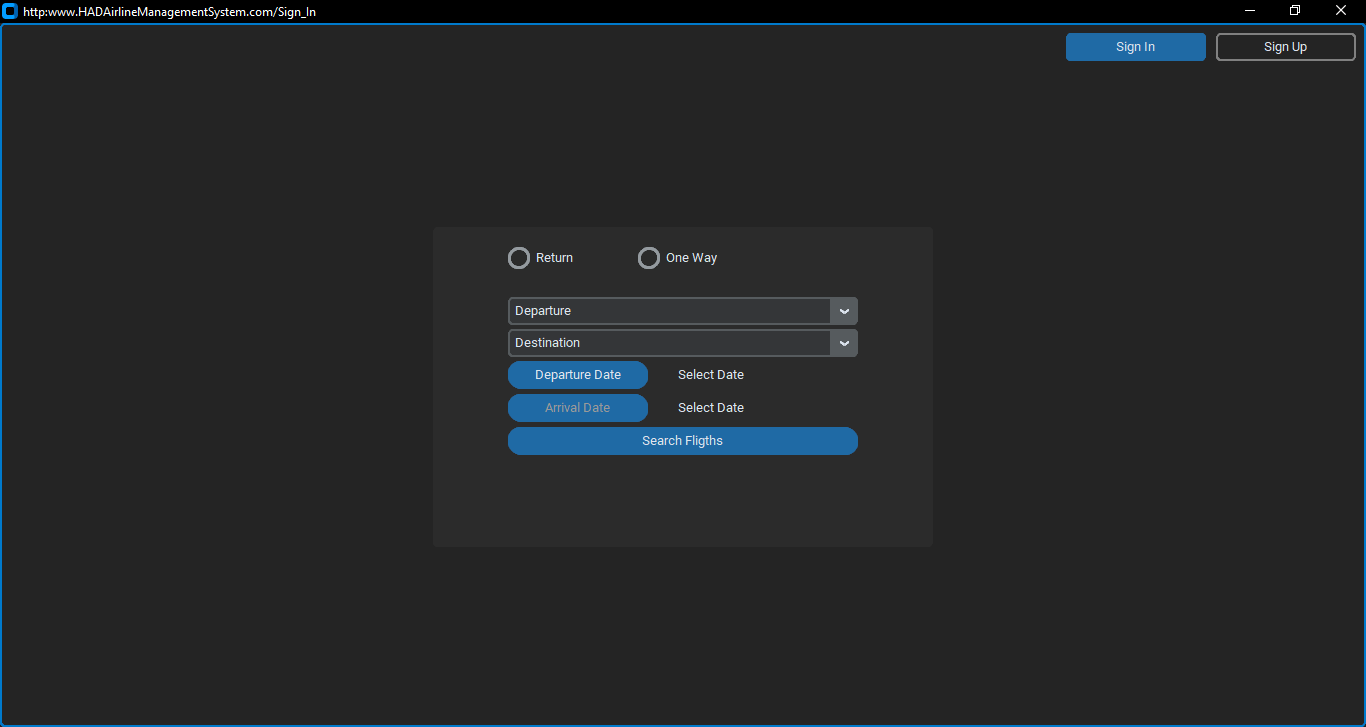
**Feature Extensions:**

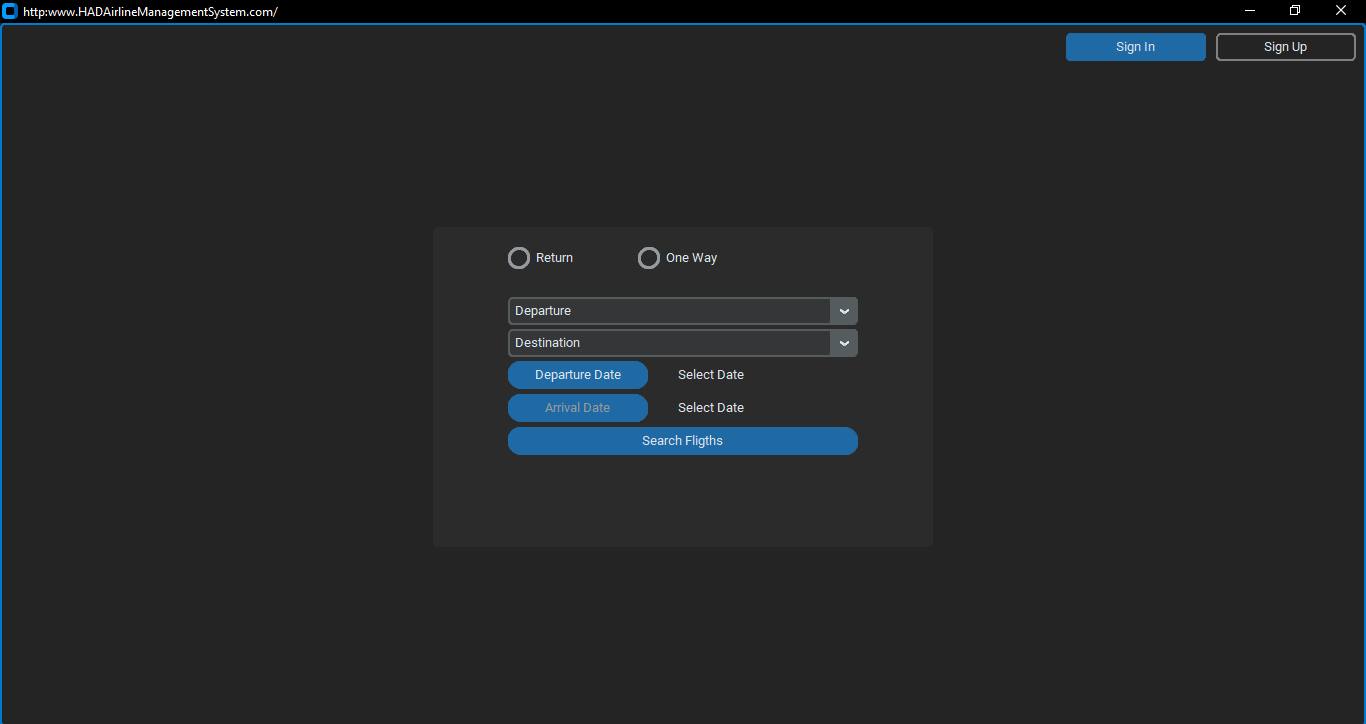
Add more payment options and integrate third party APIs for additional functionality.

**Notifications:**

Implement email or in app notifications for payment confirmation and booking reminders.

**16. Appendix**

****

****

**17. BIBLIOGRAPHY**

To develop this project many references were used:

⦁ Computer Science with Python by Sumita Arora, Dhanpat Rai publications

⦁ https://[www.python.org.in](http://www.python.org.in/)

⦁ https://[www.mysql.org](http://www.mysql.org/)