**Name**: HARI DHEJUS.VS

**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

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 facea 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

## 2.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 datato 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 lotof 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
- Front End: Python 3.7 or above
- Visual Studio Code
- RAM: 2GB or More
- Hard Disk: 2.00 GB
- 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

**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.
2. Flight ticket booking module: this module is a software component that allows users to search, select, and purchase airline tickets.

3. Flight ticket cancelling module: The cancellation feature allows users to cancel their flight bookings under specific conditions.
4. Payment module : The module calculates the total cost of the booking
5. Flight manager module :This module consist of inserting flights and  deleting flights
6. 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.

# 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 thedevelopers 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.

**Phase6**: Installation / Deployment

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

**Phase7**: 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:**

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 `go_back()` to manage backward navigation, clearing the current frame and displaying the previous one.

### 3.2 SessionBased 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 sessionaware controls for userspecific 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.***

# 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:

Hash passwords, encrypt sensitive data, and use prepared statements for SQL queries to protect against SQL injection.

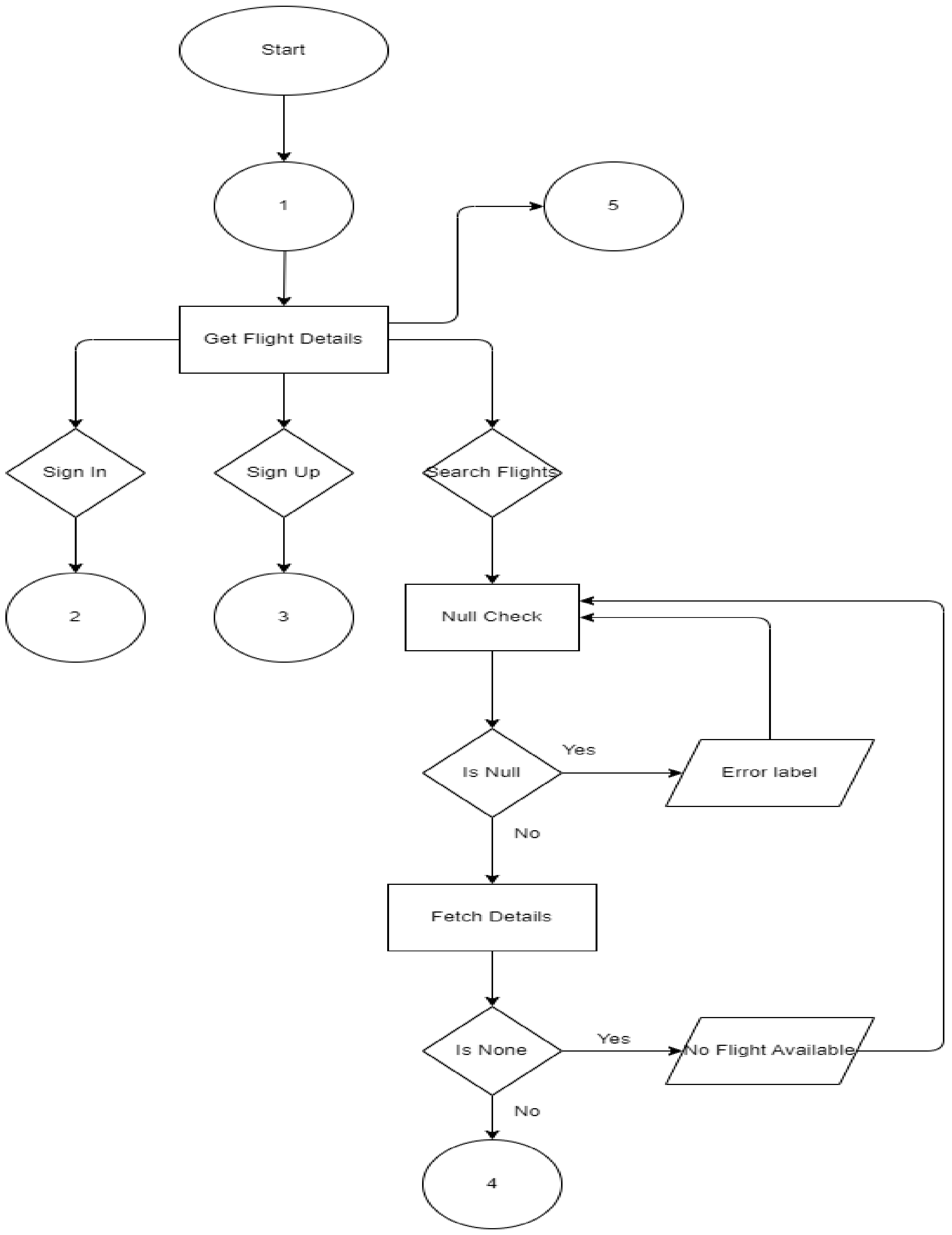
### Feature Extensions:

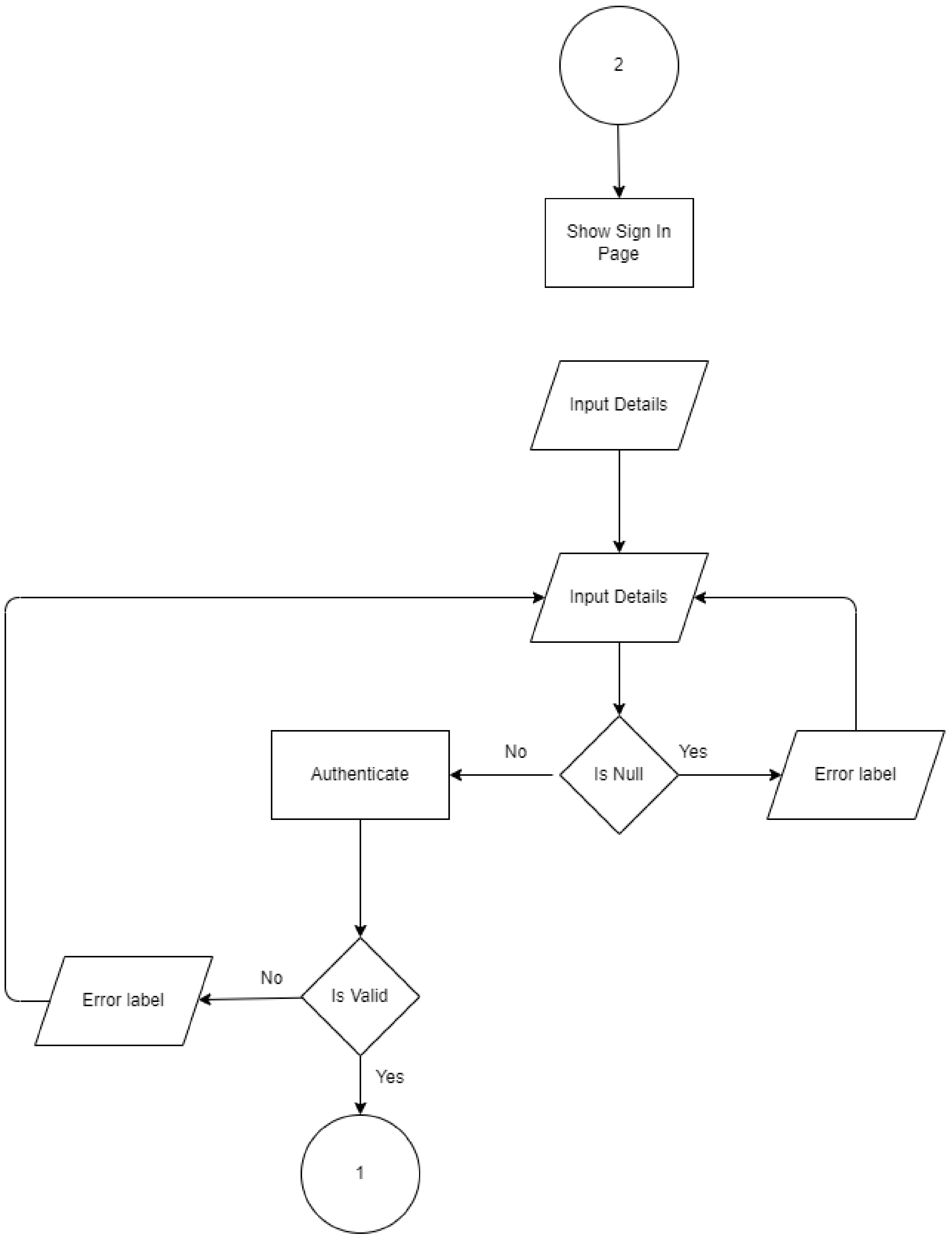
Add more payment options and integrate thirdparty APIs for additional functionality.

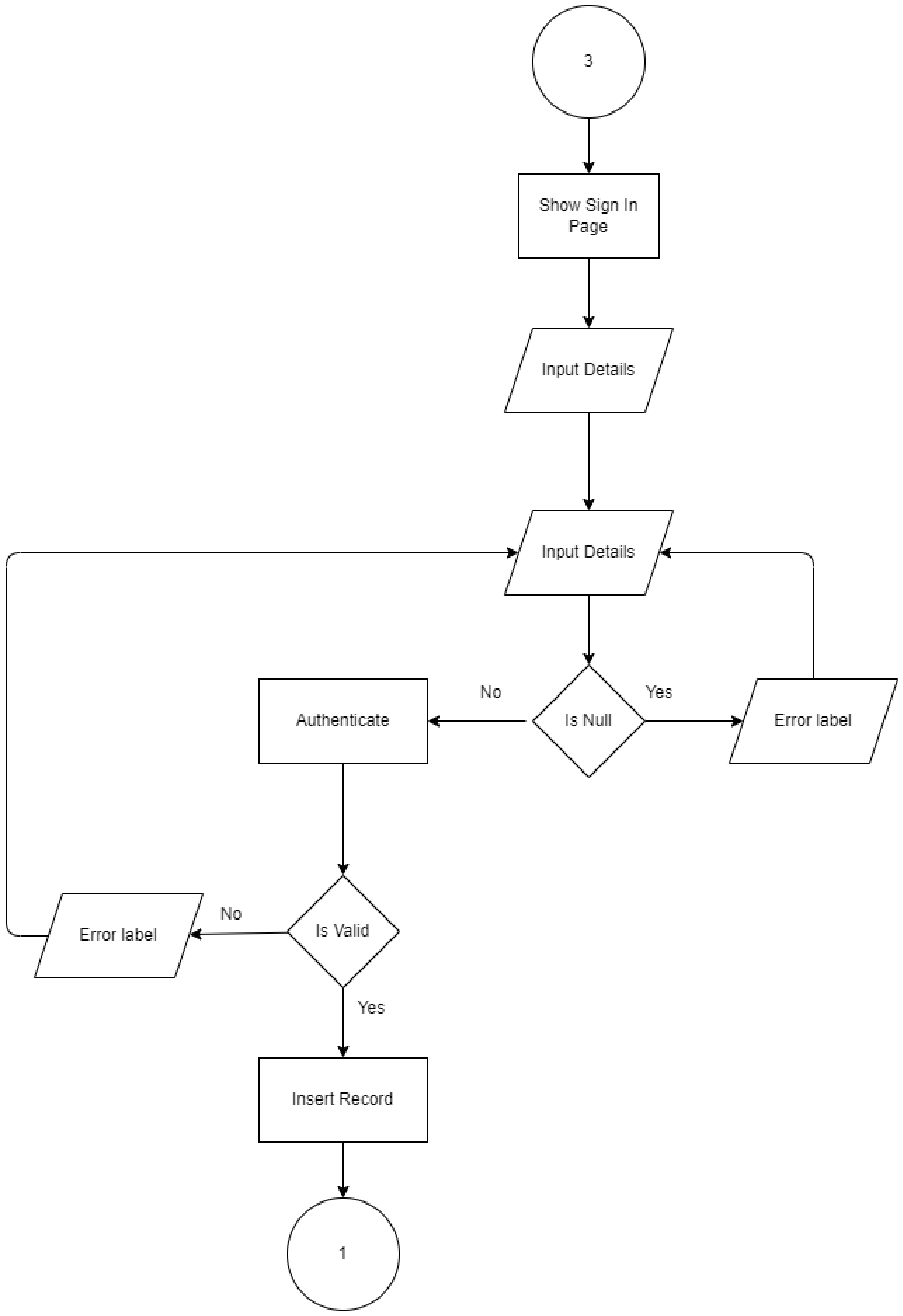
### Notifications:

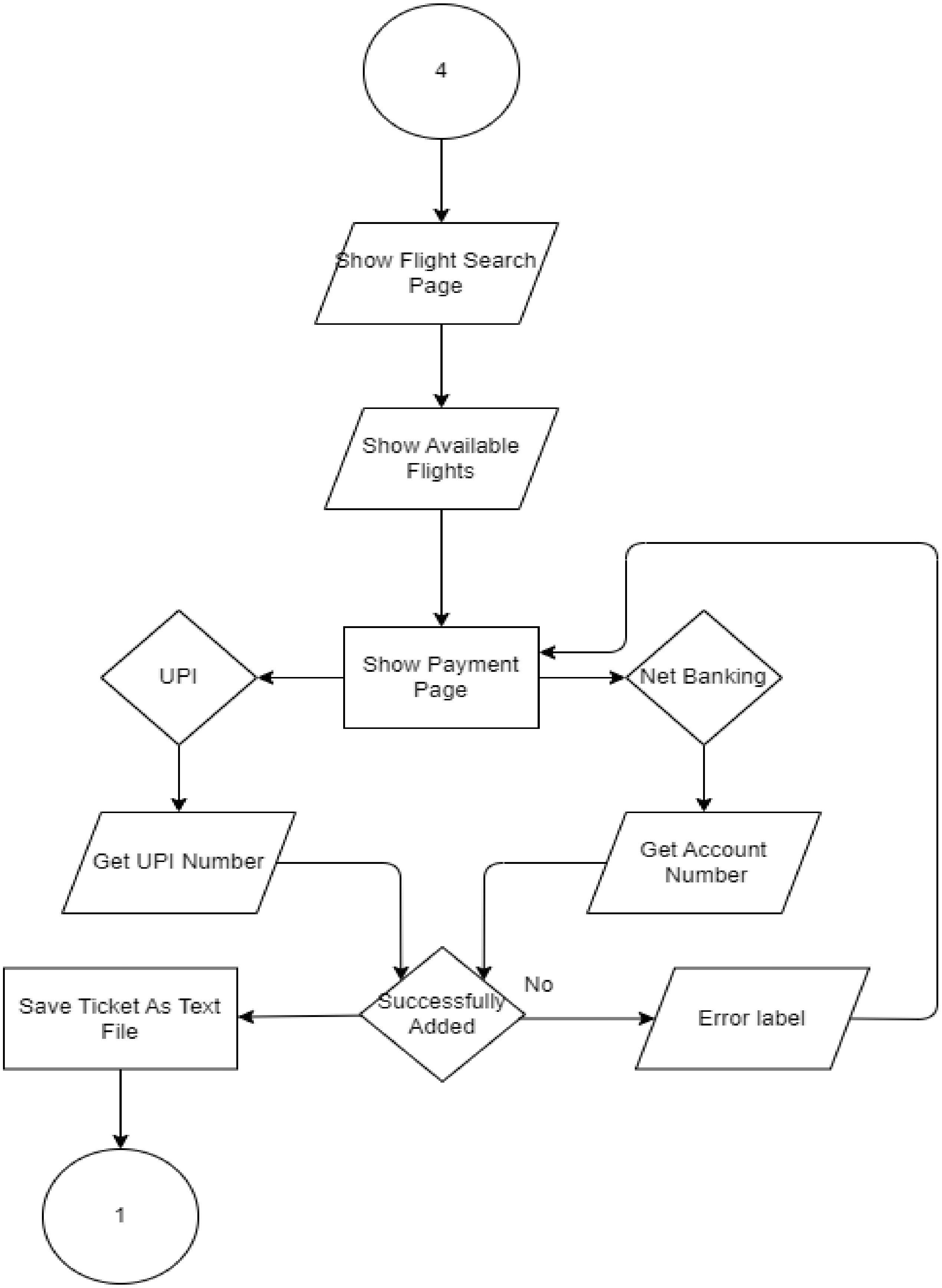
Implement email or inapp notifications for payment confirmation and booking reminders.

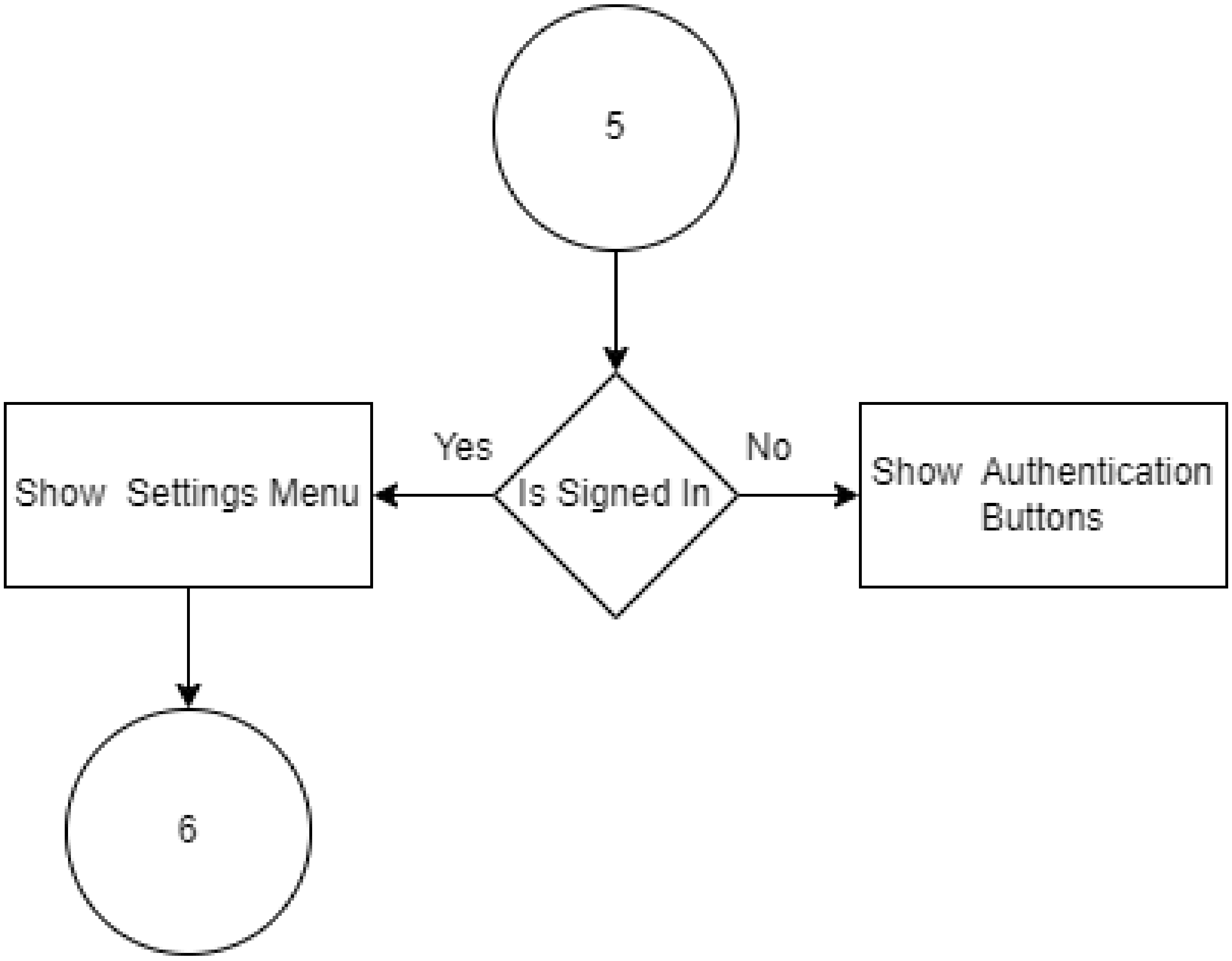
# 11. FLOWCHART

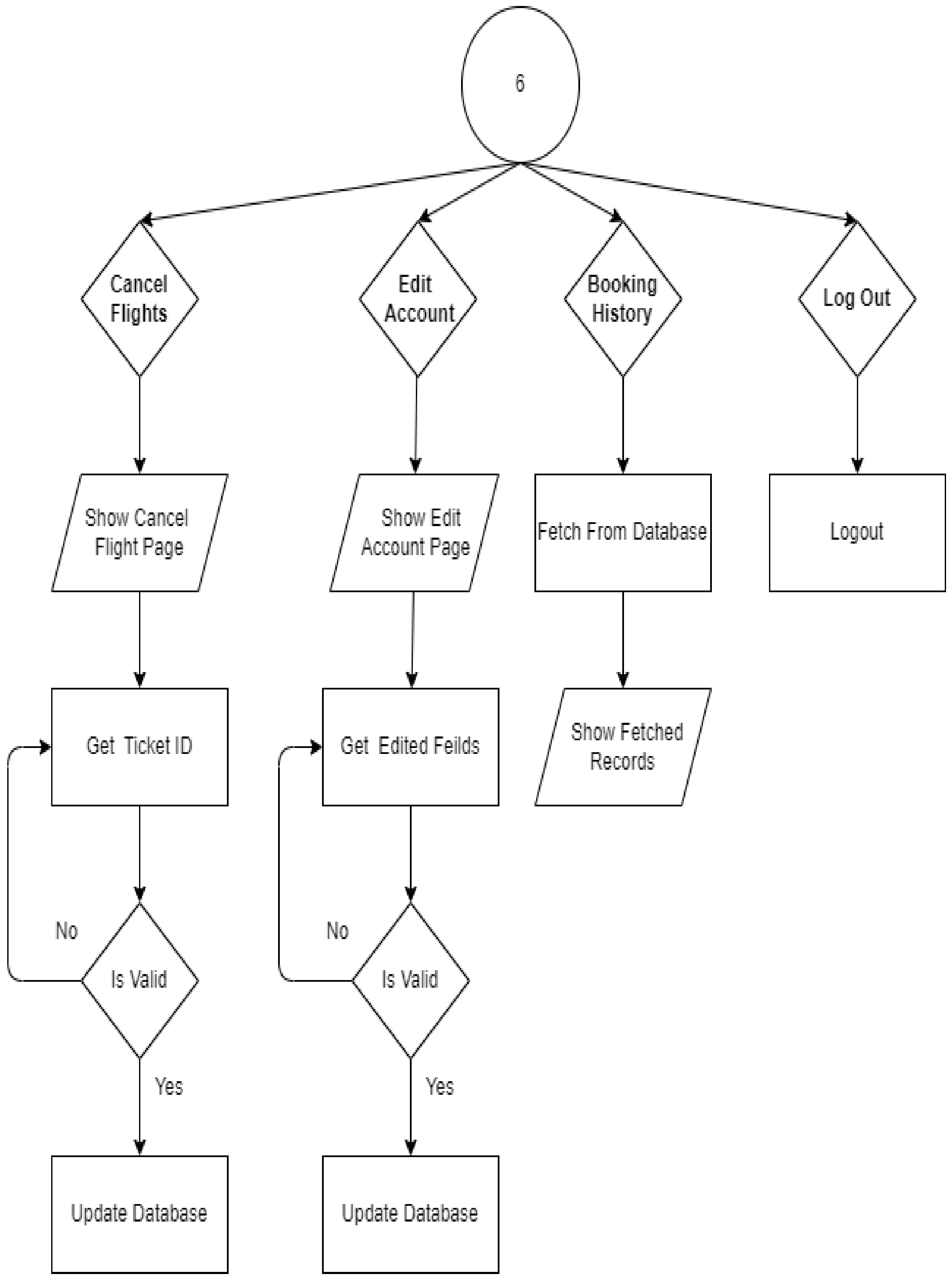

Start
1
5
Get Flight Details
Sign In
Sign Up
Search Flights
2
3
Null Check
Is Null
Yes
Error label
No
Fetch Details
Is None
Yes
No Flight Available
No
4


2
Show Sign In Page
Input Details
Input Details
Is Null
No
Yes
Authenticate
Error label
Is Valid
No
Error label
Yes
1


3
Show Sign In Page
Input Details
Input Details
Is Null
No
Yes
Authenticate
Error label
Is Valid
No
Yes
Error label
Insert Record
1


4
Show Flight Search Page
Show Available Flights
UPI
Show Payment Page
Net Banking
Get UPI Number
Get Account Number
Save Ticket As Text File
Successfully Added
No
Error label
1


5
Is Signed In
Yes
No
Show Settings Menu
Show Authentication Buttons
6


6
Cancel Flights
Edit Account
Booking History
Log Out
Show Cancel Flight Page
Show Edit Account Page
Fetch From Database
Logout
Get Ticket ID
Get Edited Feilds
Show Fetched Records
No
Is Valid
No
Is Valid
Yes
Yes
Update Database
Update Database

# 12. Source code

```
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

# Global Modules
```

```
from customtkinter import *
from PIL import Image
from subprocess import call
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 Toplevel
import time
import random
import ast
from colorama import Fore
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 --------
```

```
_isSignedIn = FALSE
is_flight_details_obtained = False
User = ""
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("""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_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 `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,
                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 NULL,
                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 `payment` (
                `PID` int NOT NULL AUTO_INCREMENT,
                `P_UID` int DEFAULT NULL,
                `AMOUNT` int DEFAULT NULL,
                `P_STATUS` int DEFAULT NULL,
                `P_ACC_NUM` int DEFAULT NULL,
                `P_UPI_NUM` int 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:
        pass

    finally :
        print("Sucessfully Initialized Database.")
        cur.execute("USE airwaysms2_0 ;")
DB_INIT_()
#--------------------- GLOBAL FUNCTIONS ---------------------------------

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()

    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)))

    def go_back():
        PG_search_flight_()

    dummy_back_btn = CTkButton(Main_fame, text="Back", command = go_back)
    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)) )

    def on_UPI_btn_click():
        root.title ("http:www.HADAirlineManagementSystem.com/Payment/UPI")
        for i in form_frm.winfo_children():
            i.destroy()

        def go_back():
            PG_Payment()

        dummy_back_btn = CTkButton(Main_fame, text="Back", command = go_back)
        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,UPI_Number, "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()

        def go_back():
            PG_Payment()

        dummy_back_btn = CTkButton(Main_fame, text="Back", command = go_back)
        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,ACC_Number, "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()

    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 go_back():
        for i in Main_fame.winfo_children():
            i.destroy()
        PG_Sign_in()

    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 = pass_Entry.get()
        _re_pass = re_pass_Entry.get()
        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 = go_back)
        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()

    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))
                        )

    def go_back():
        global _isSignedIn
        if _isSignedIn == True :
            for i in Main_fame.winfo_children():
                i.destroy()
            Main_frm_Authentication_Btns()
            PG_search_flight_()
        else :
            for i in Main_fame.winfo_children():
                i.destroy()
            Main_frm_Authentication_Btns()
            PG_Get_Flight_Details()
    dummy_back_btn = CTkButton(Main_fame,text="Back", command = go_back)
    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 = _pass.get()
        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 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
                User = qry_result[0]
                _isSignedIn = True
                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_():
    prev_page = 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")
    def go_back():
        for i in Main_fame.winfo_children():
            i.destroy()
        Main_frm_Authentication_Btns()
        PG_Get_Flight_Details()

    dummy_back_btn = CTkButton(Main_fame,text="Back", command = go_back)
    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()

    prev_page = 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,d
iv_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")

    Origin_Airport = CTkComboBox(div_frame,width=din_frm_widget_width,
                            values=major_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=major_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

            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")) #errorLabeling(temp_frame, "Successfully
Canceled", _textcolor = "green", _x = 110, _y = (tmp_root_height/(2)-(28/2)+38) )
                    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 go_back():
    for i in Main_fame.winfo_children():
        i.destroy()
    PG_Sign_in()

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 = pass_Entry.get()
    _re_pass = re_pass_Entry.get()
    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)
                # errorLabeling(form_frm, "Succesfully Updated", _textcolor = "green", _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 =  ""
            PG_Get_Flight_Details()

      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()
```

# TABLE DESIGN

| DATABASE | TABLE | COLUMN | DATATYPE | CONSTRAINT |
|---|---|---|---|---|
| | | | | |
| airwaysms2_0 | user details | UF_name | VARCHAR(100) | Primary Key, Auto Increment |
| | | 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 | |
| | | | | |
| | 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 |
| | | 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.

# 17. Future enhancement

## Potential Enhancements and Security Considerations

**Security Enhancements:**

Hash passwords, encrypt sensitive data, and use prepared statements for SQL queries to protect against SQL injection.

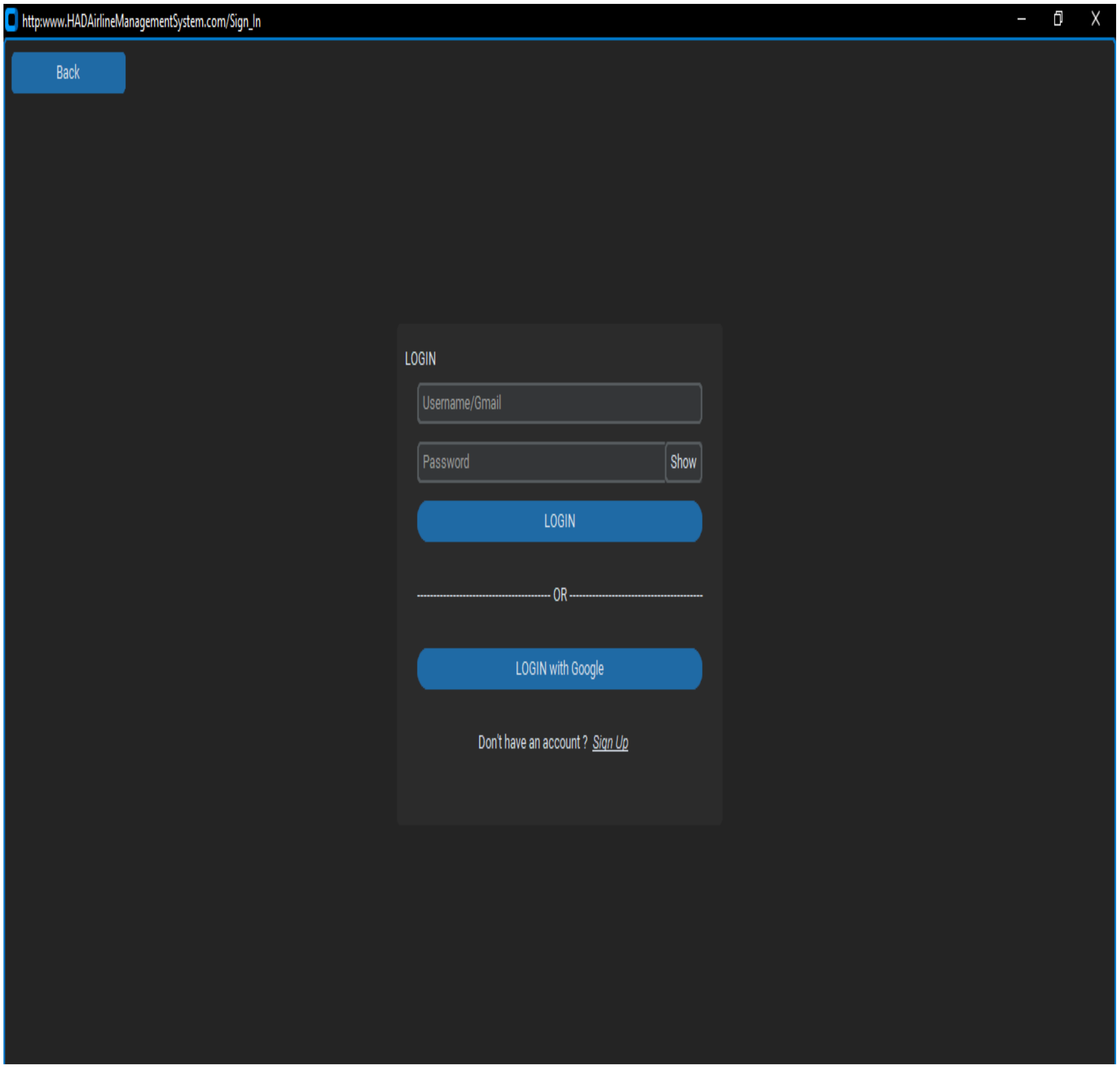
**Feature Extensions:**

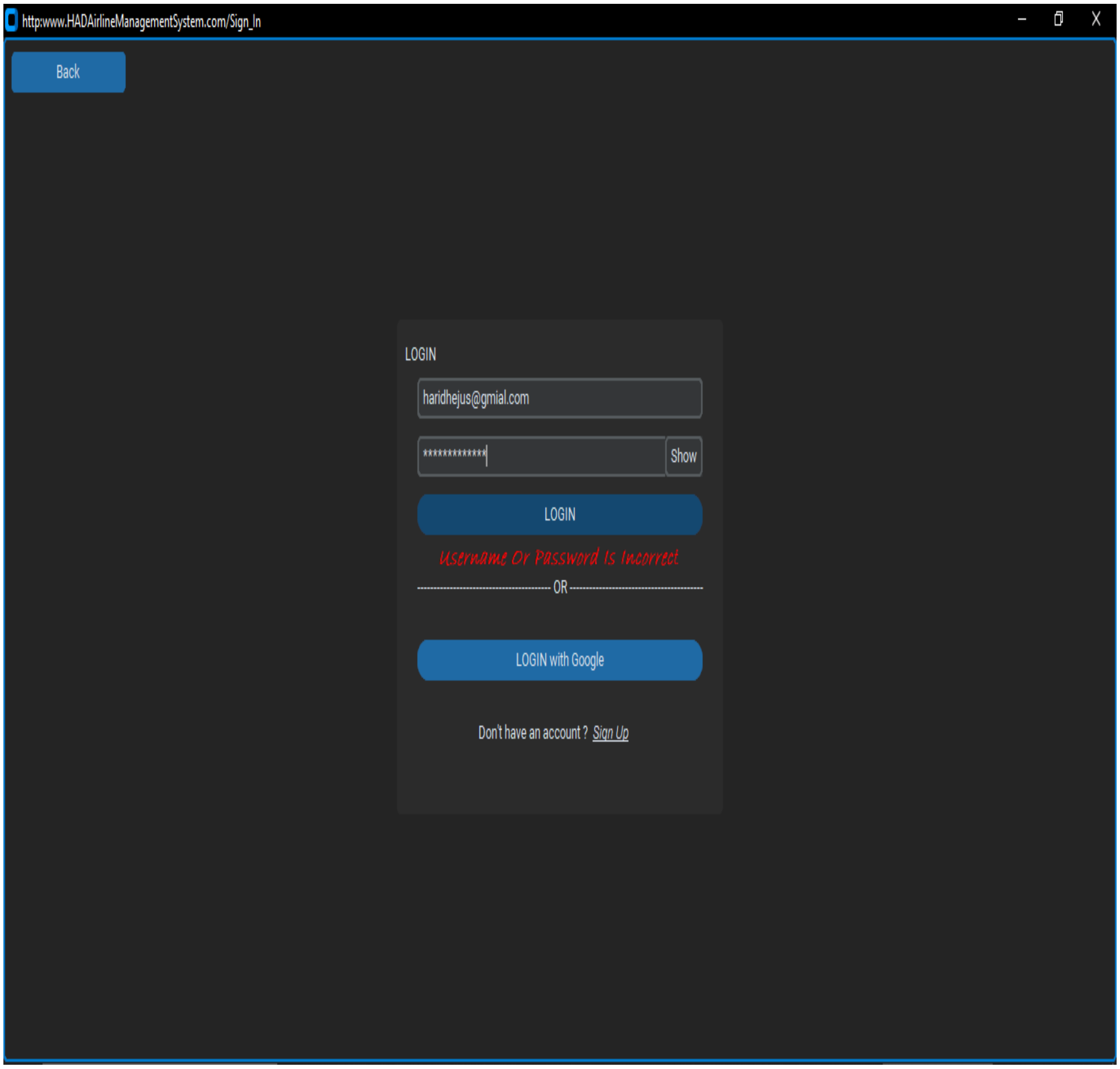
Add more payment options and integrate thirdparty APIs for additional functionality.

**Notifications:**

Implement email or inapp notifications for payment confirmation and booking reminders.

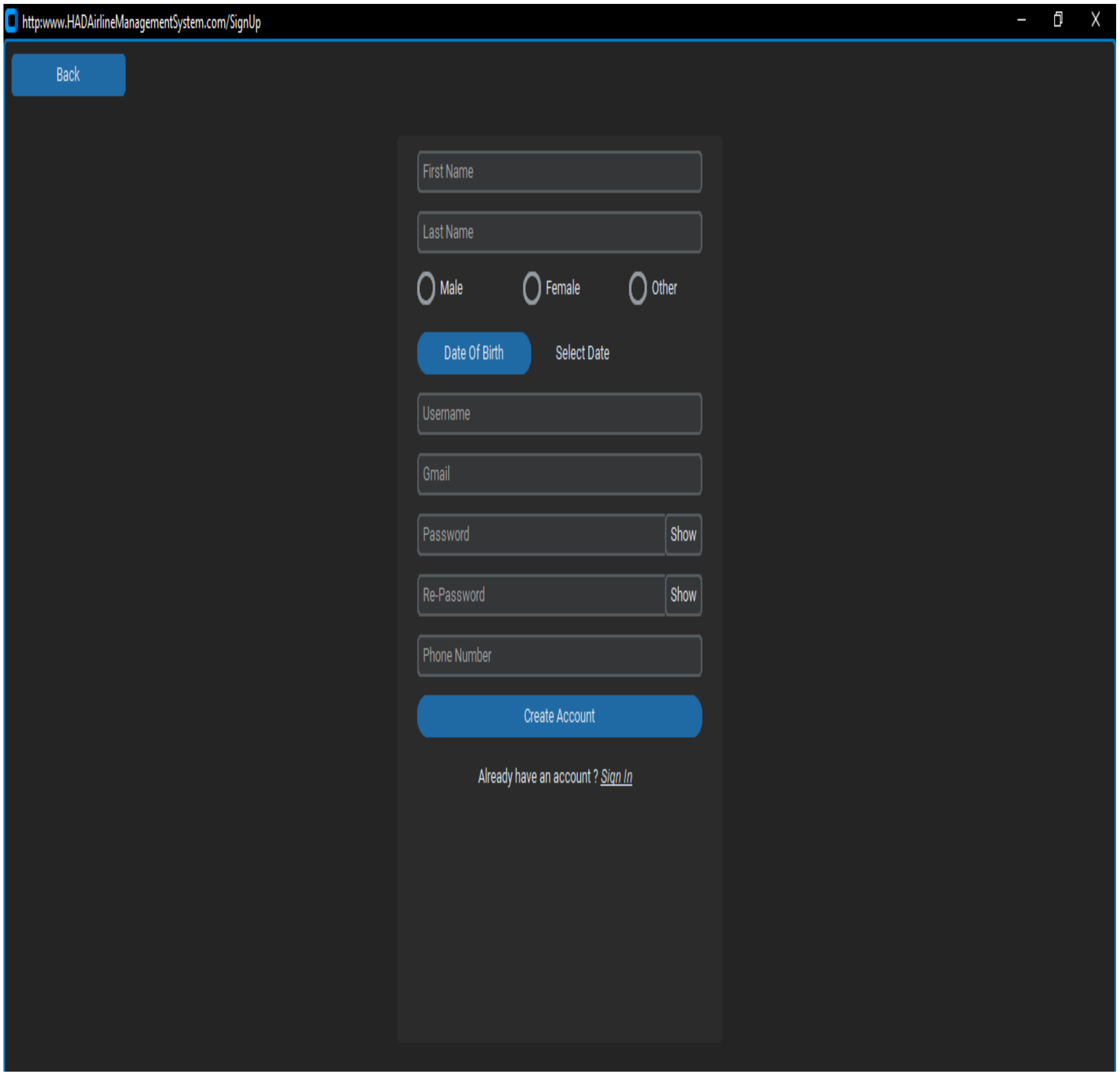
# 16. APPENDIX

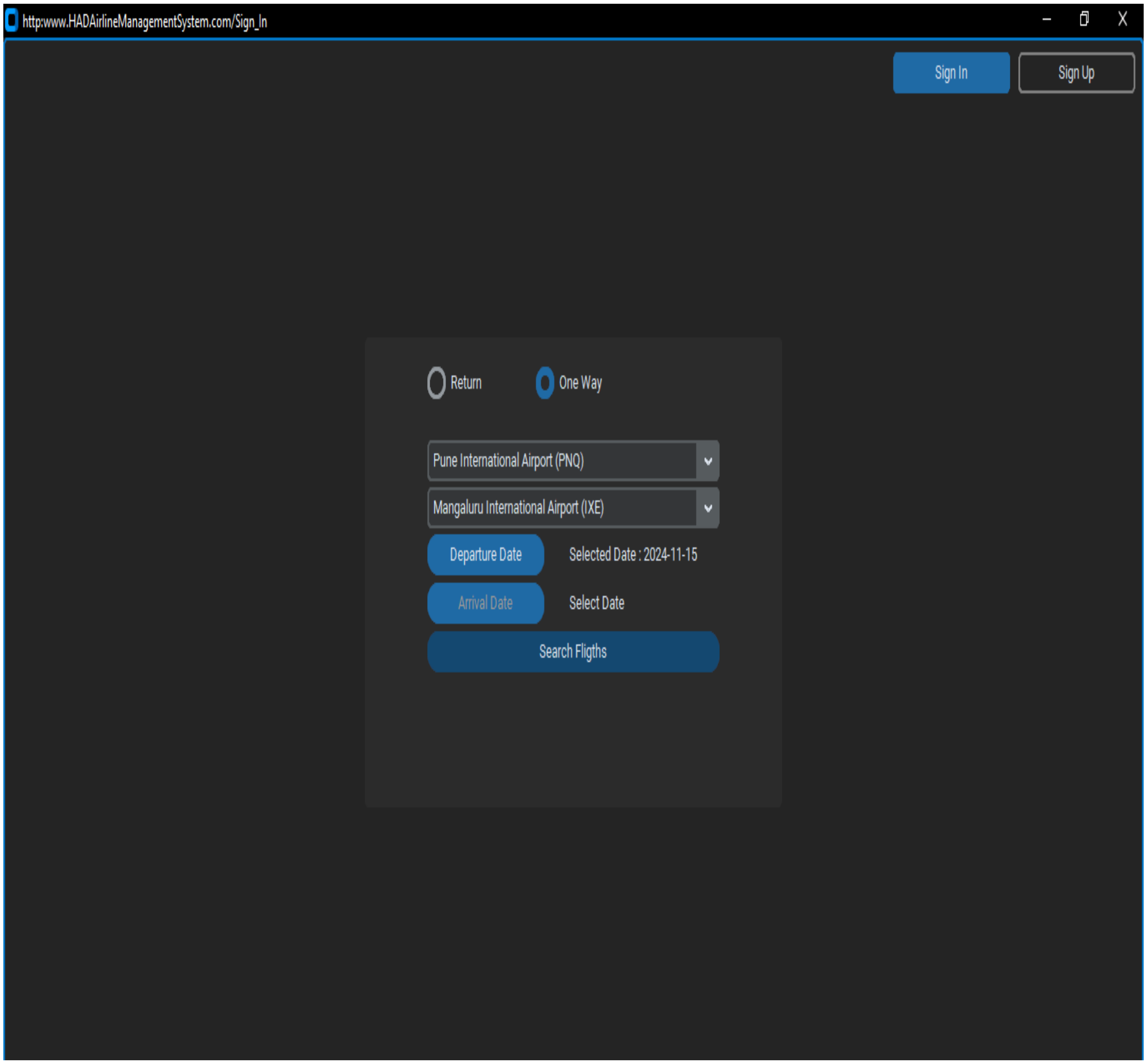

http:www.HADAirlineManagementSystem.com/Sign_In
Back
LOGIN
Username/Gmail
Password
Show
LOGIN
OR
LOGIN with Google
Don't have an account ? Sign Up



http:www.HADAirlineManagementSystem.com/Sign_In

Username Or Password Is Incorrect


http:www.HADAirlineManagementSystem.com/SignUp
Back
First Name
Last Name
Male
Female
Other
Date Of Birth
Select Date
Username
Gmail
Password
Show
Re-Password
Show
Phone Number
Create Account
Already have an account ? Sign In


http:www.HADAirlineManagementSystem.com/Sign_In

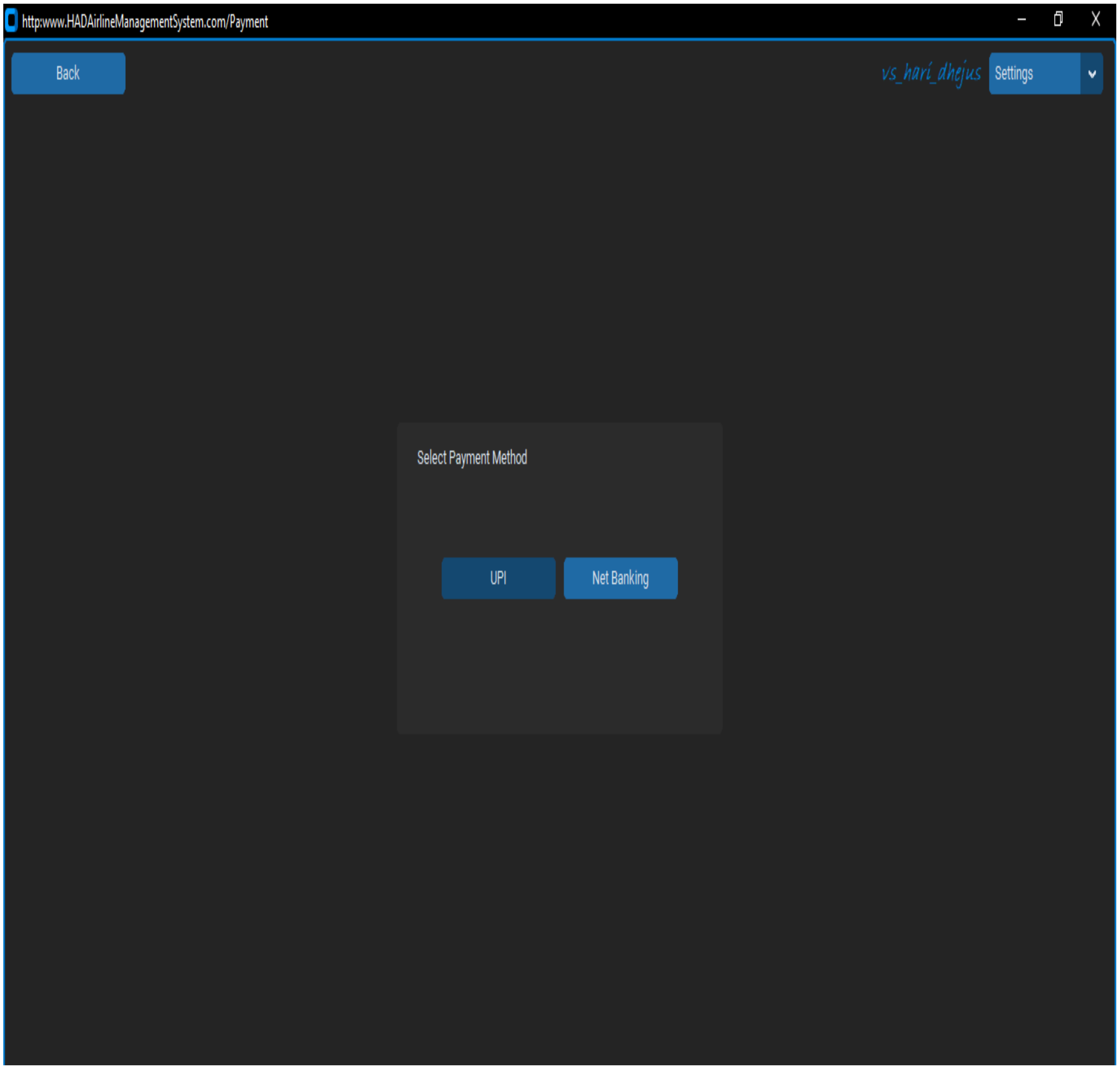
Return [✓ One Way]

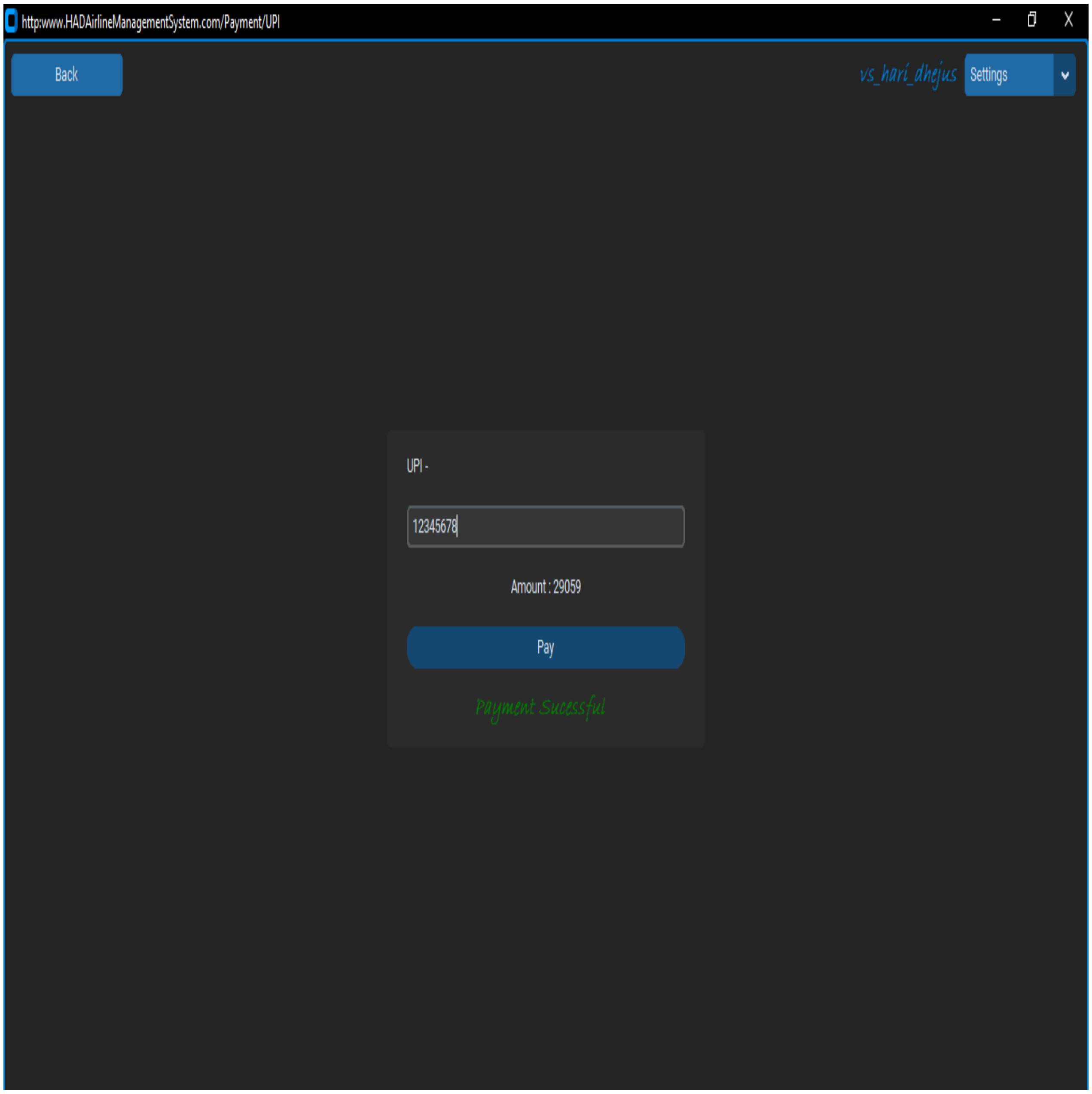
Pune International Airport (PNQ)

Mangaluru International Airport (IXE)

Departure Date — Selected Date : 2024-11-15

Arrival Date — Select Date


http:www.HADAirlineManagementSystem.com/Payment
Back
vs_hari_dhejus
Settings
Select Payment Method
UPI
Net Banking

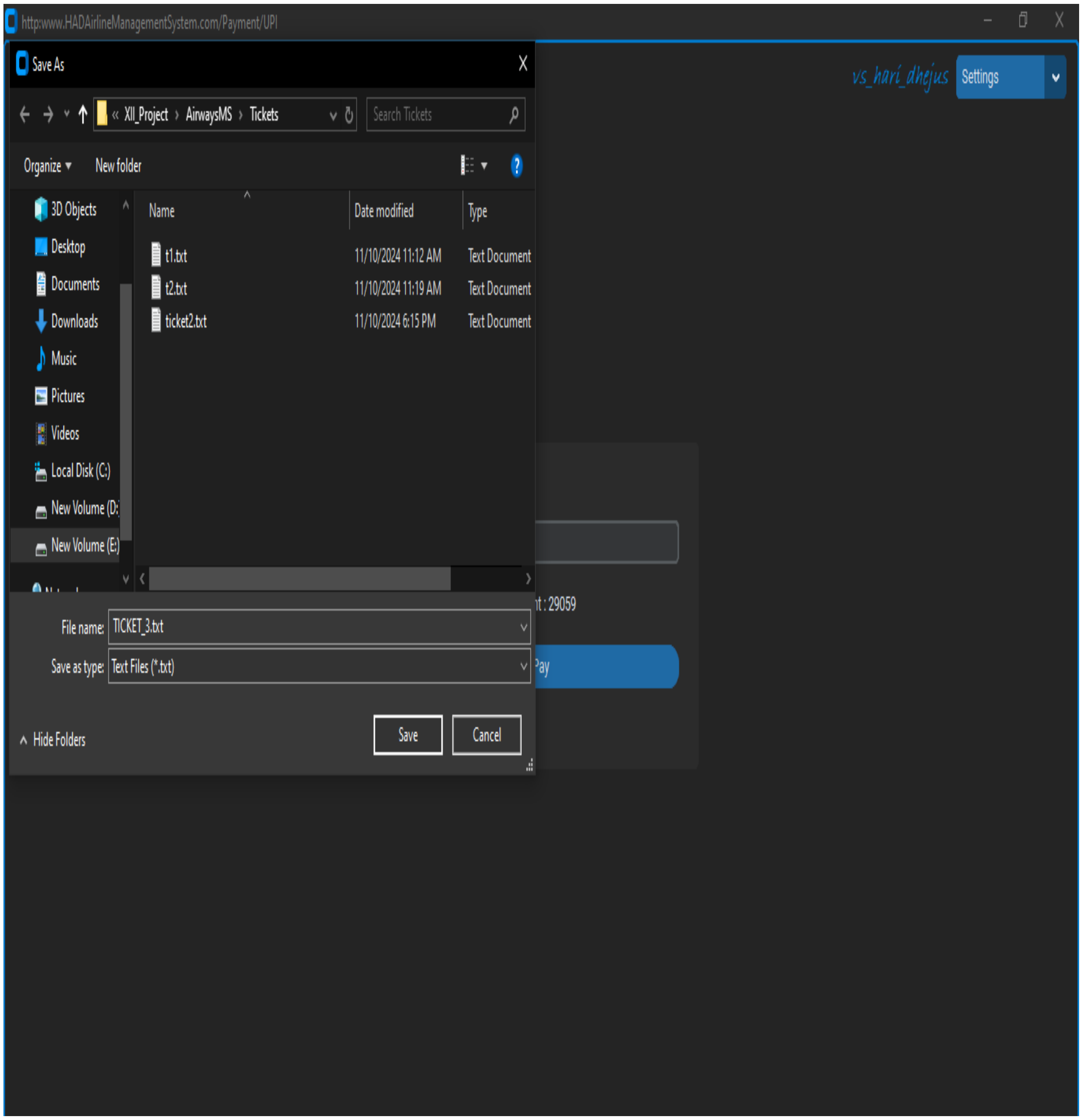


UPI -

12345678

Amount : 29059

Payment Sucessful



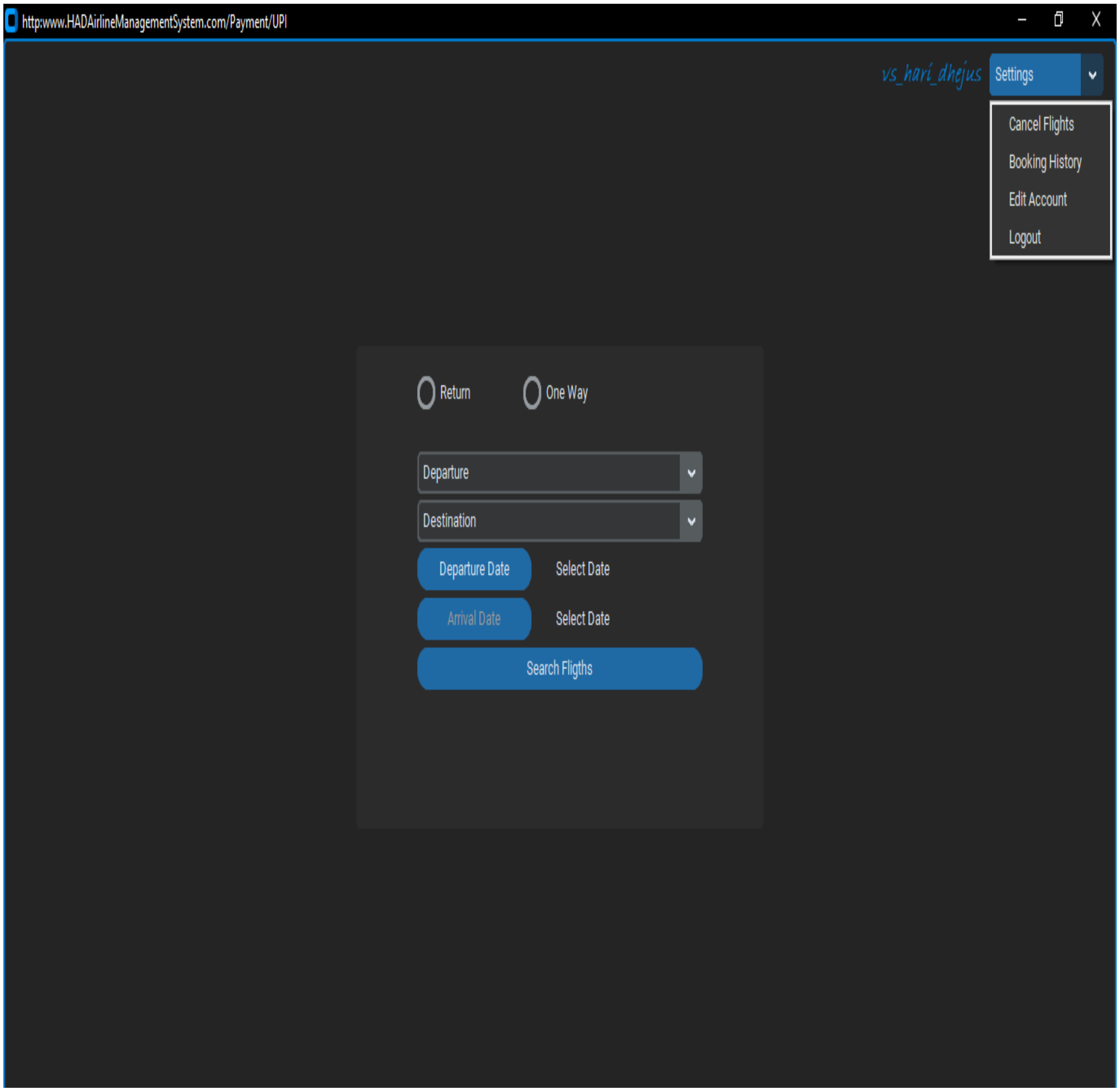
http:www.HADAirlineManagementSystem.com/Payment/UPI

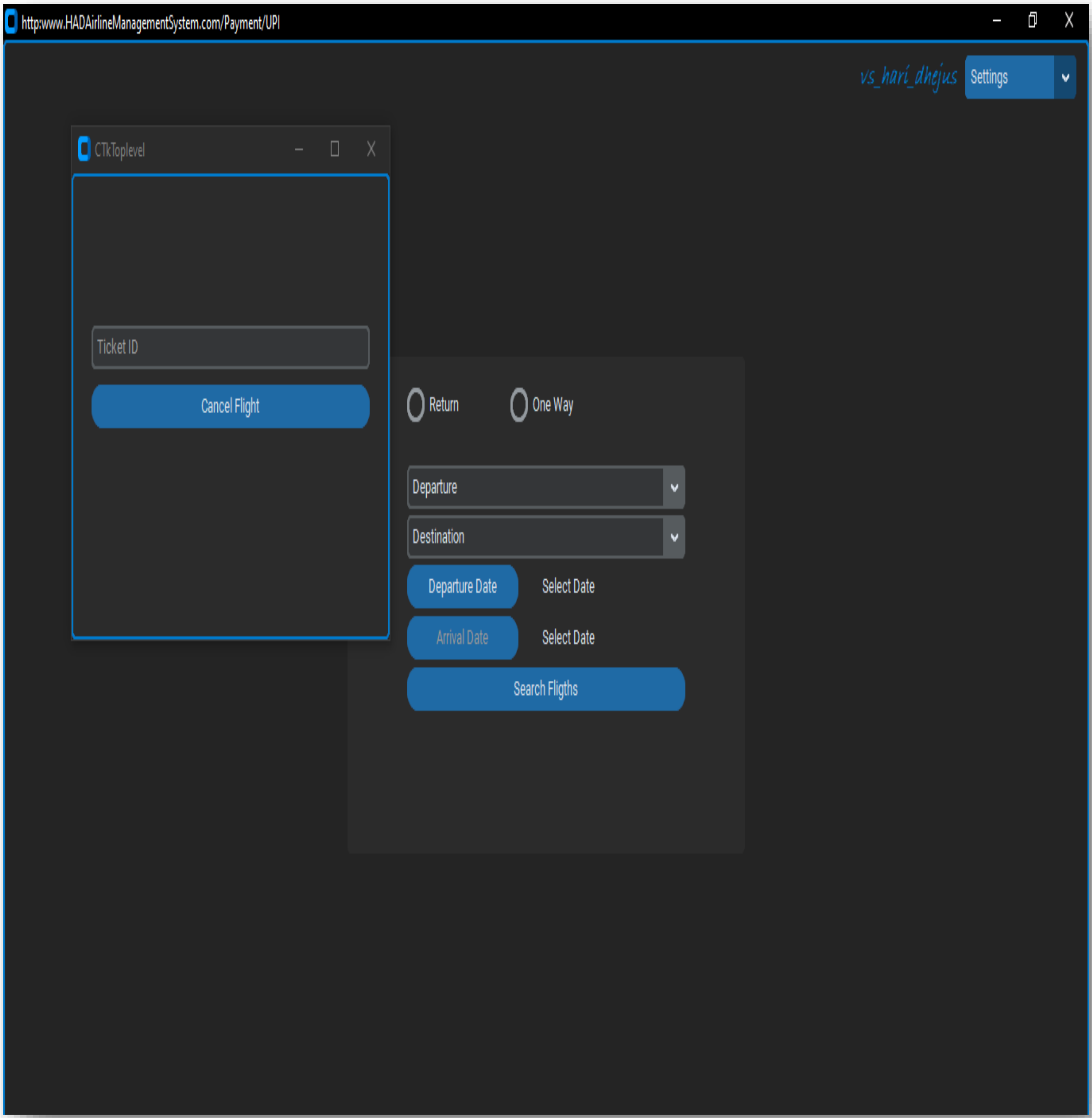
| Name | Date modified | Type |
|---|---|---|
| t1.txt | 11/10/2024 11:12 AM | Text Document |
| t2.txt | 11/10/2024 11:19 AM | Text Document |
| ticket2.txt | 11/10/2024 6:15 PM | Text Document |

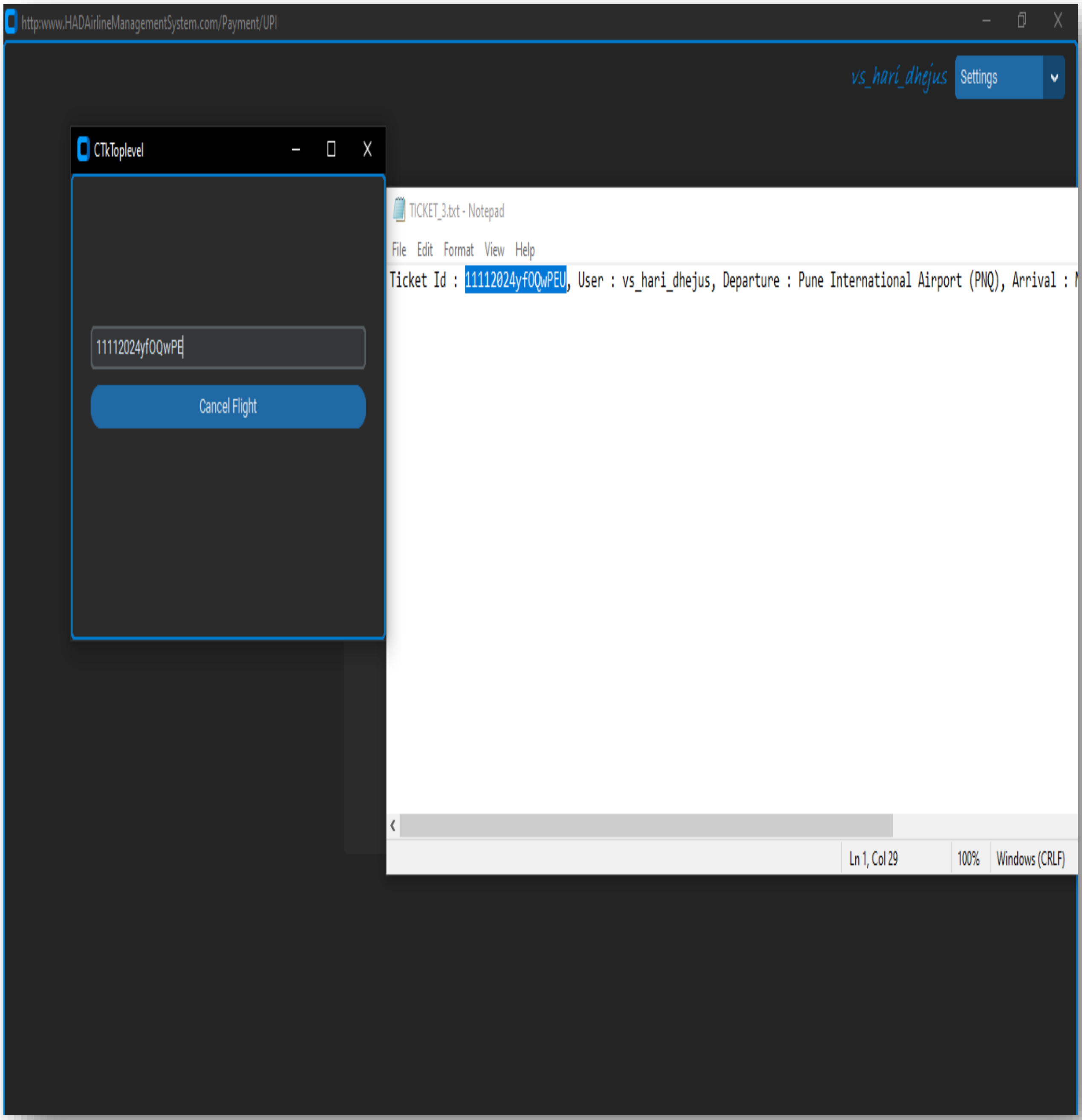
File name: TICKET_3.txt

Save as type: Text Files (*.txt)

nt : 29059


http:www.HADAirlineManagementSystem.com/Payment/UPI
vs_hari_dhejus
Settings
Cancel Flights
Booking History
Edit Account
Logout
Return
One Way
Departure
Destination
Departure Date
Select Date
Arrival Date
Select Date
Search Fligths


http:www.HADAirlineManagementSystem.com/Payment/UPI
vs_hari_dhejus
Settings
CTkToplevel
Ticket ID
Cancel Flight
Return
One Way
Departure
Destination
Departure Date
Select Date
Arrival Date
Select Date
Search Fligths


http:www.HADAirlineManagementSystem.com/Payment/UPI

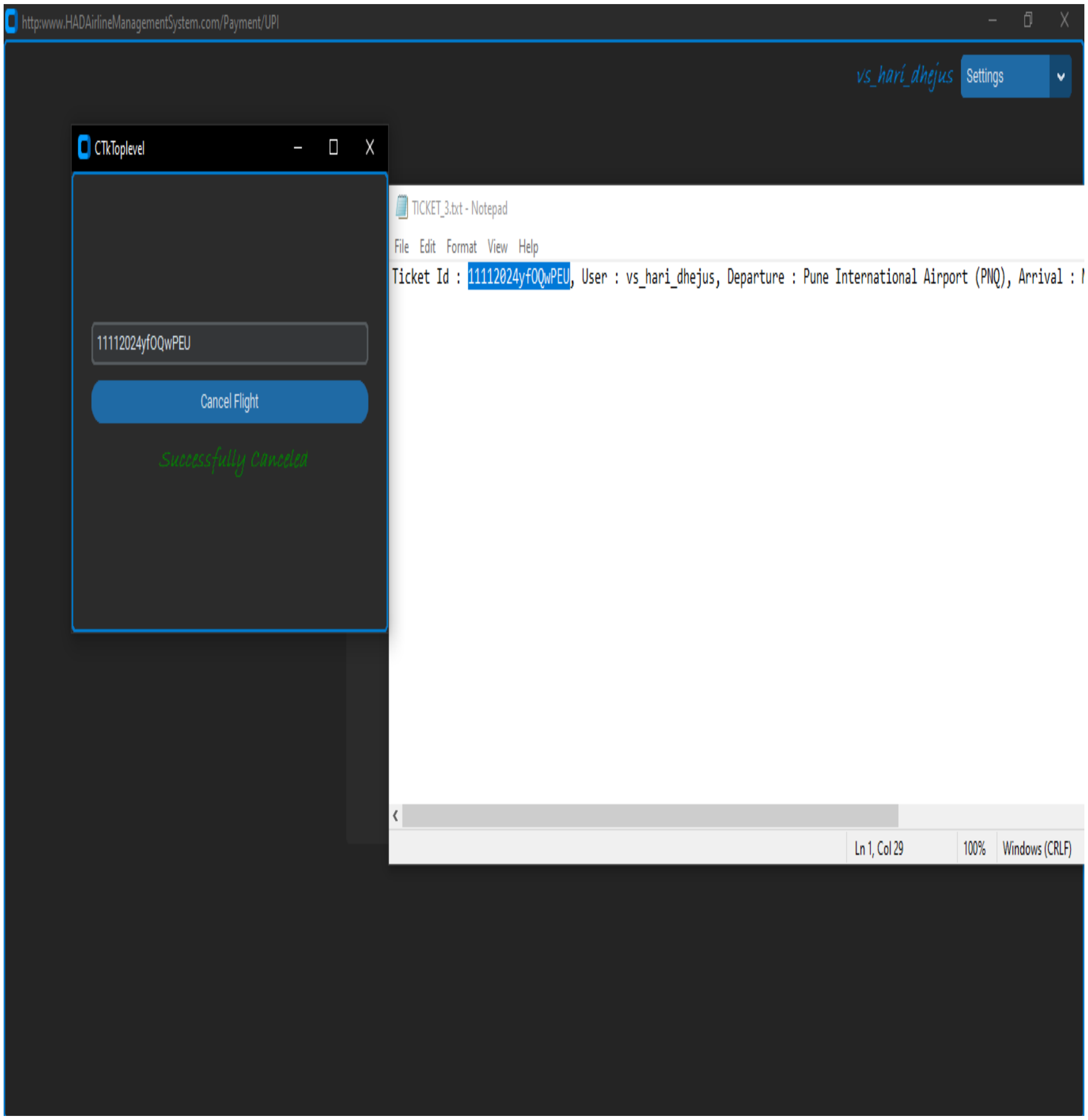
vs_hari_dhejus

CTkToplevel

11112024yfOQwPE

TICKET_3.txt - Notepad

```
Ticket Id : 11112024yfOQwPEU, User : vs_hari_dhejus, Departure : Pune International Airport (PNQ), Arrival :
```


http:www.HADAirlineManagementSystem.com/Payment/UPI
vs_hari_dhejus
Settings
CTkToplevel
TICKET_3.txt - Notepad
File Edit Format View Help

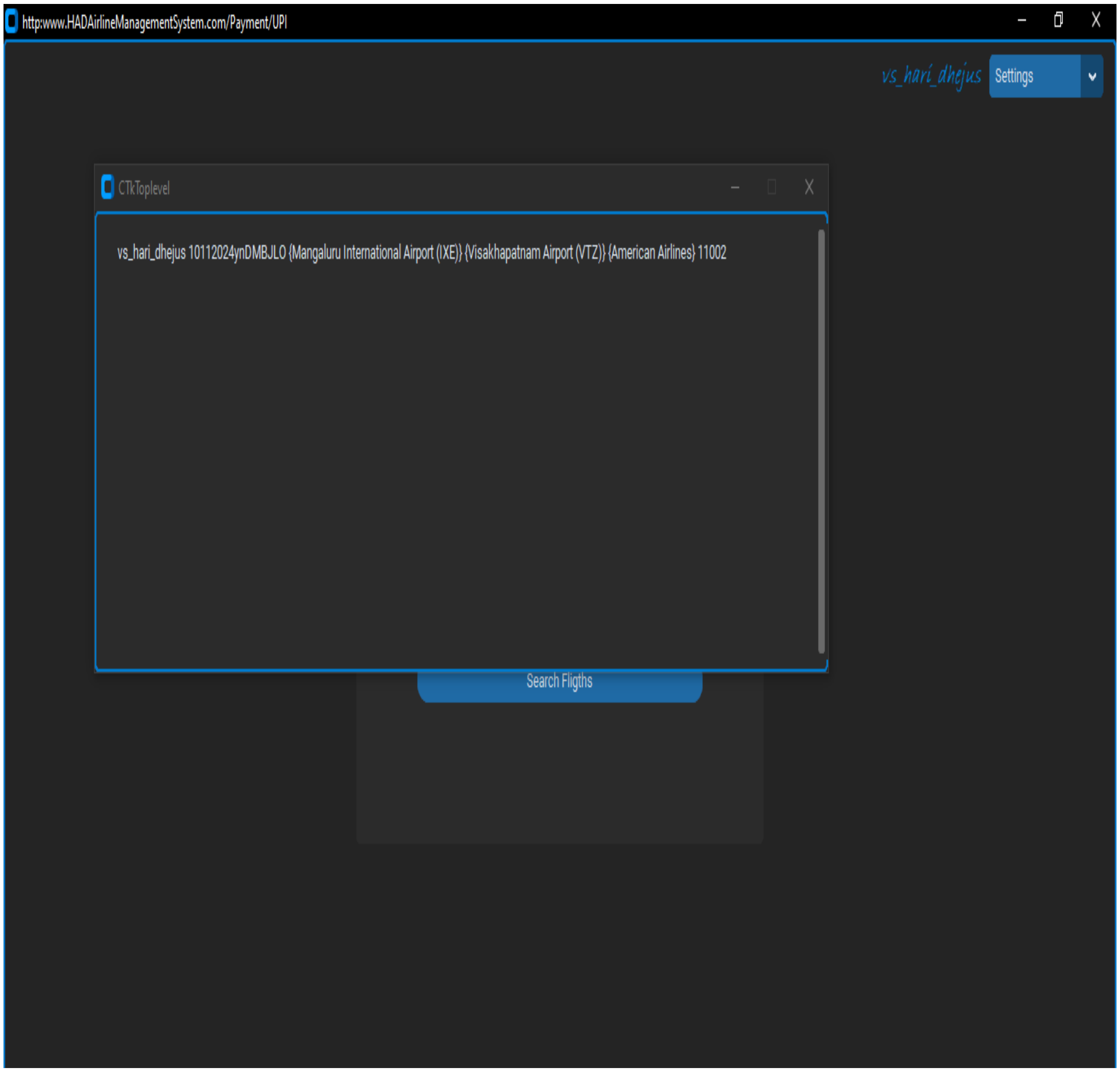
11112024yfOQwPEU

Cancel Flight

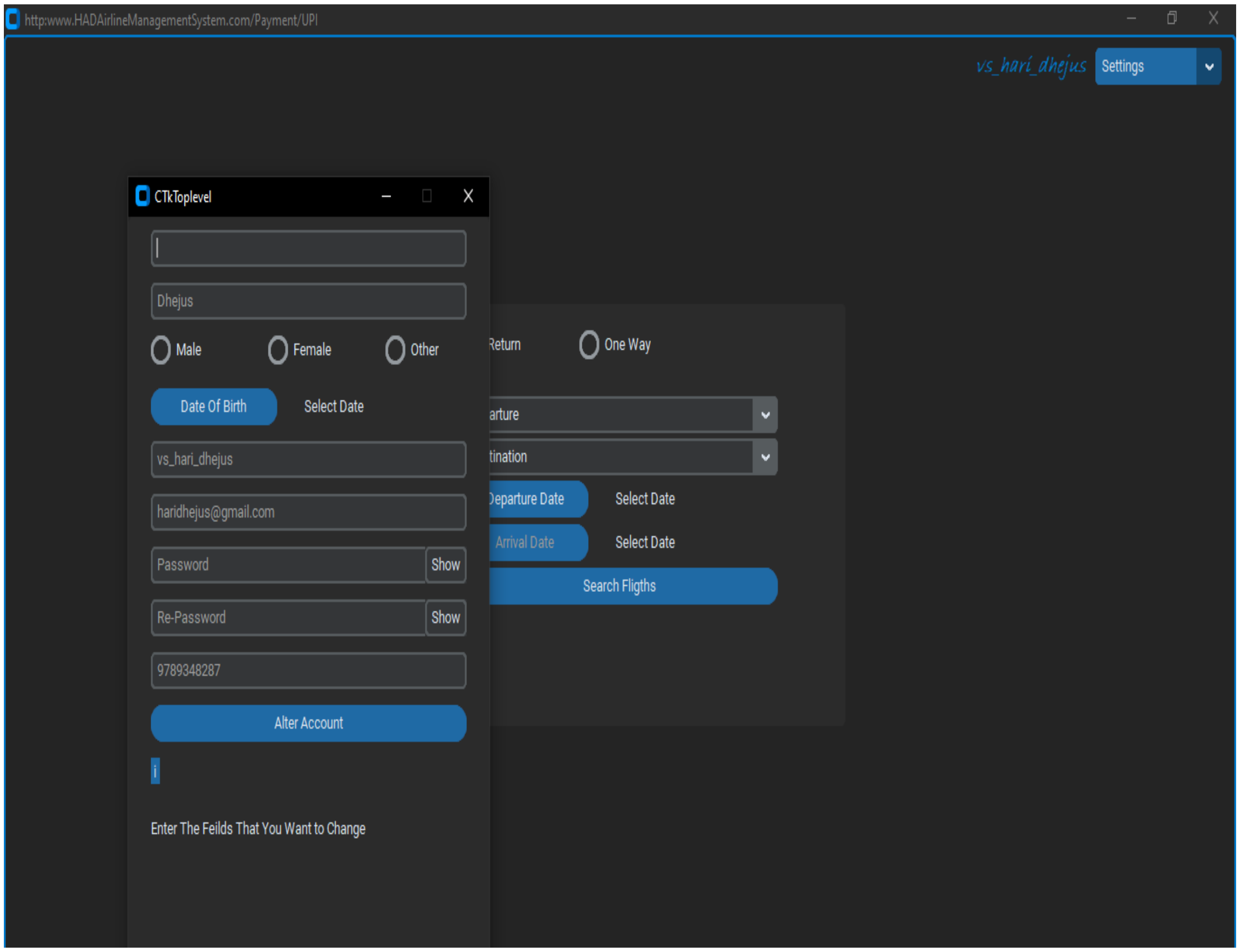
Successfully Canceled

```
Ticket Id : 11112024yfOQwPEU, User : vs_hari_dhejus, Departure : Pune International Airport (PNQ), Arrival :
```

Ln 1, Col 29 100% Windows (CRLF)


http:www.HADAirlineManagementSystem.com/Payment/UPI

vs_hari_dhejus 10112024ynDMBJLO {Mangaluru International Airport (IXE)} {Visakhapatnam Airport (VTZ)} {American Airlines} 11002


http:www.HADAirlineManagementSystem.com/Payment/UPI

vs_hari_dhejus Settings

CTkToplevel

Dhejus

Male Female Other

Date Of Birth Select Date

vs_hari_dhejus

haridhejus@gmail.com

Password Show

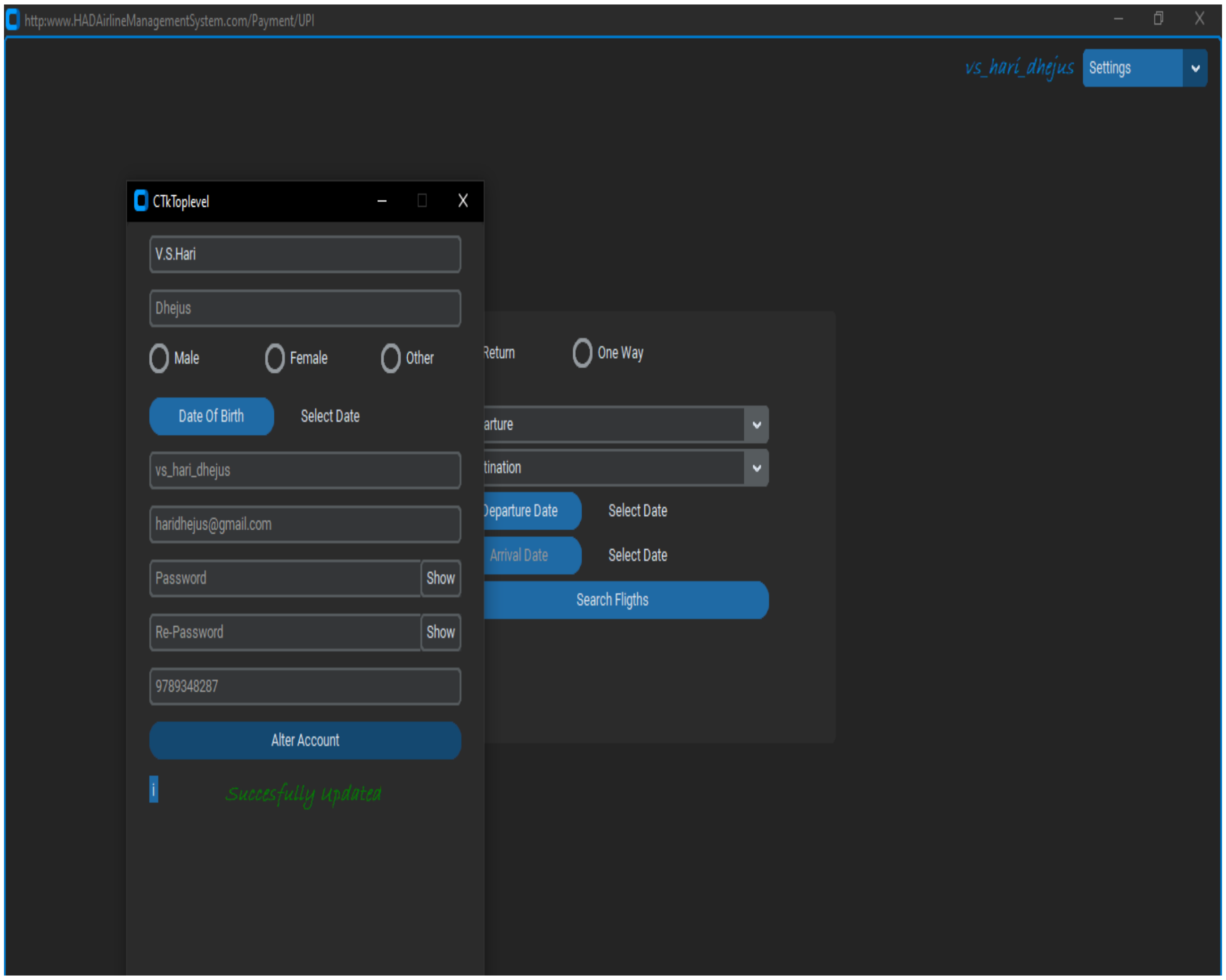
Re-Password Show

9789348287

Alter Account

i

Enter The Feilds That You Want to Change



http:www.HADAirlineManagementSystem.com/Payment/UPI

V.S.Hari

Dhejus

Male Female Other

Date Of Birth Select Date

vs_hari_dhejus

haridhejus@gmail.com

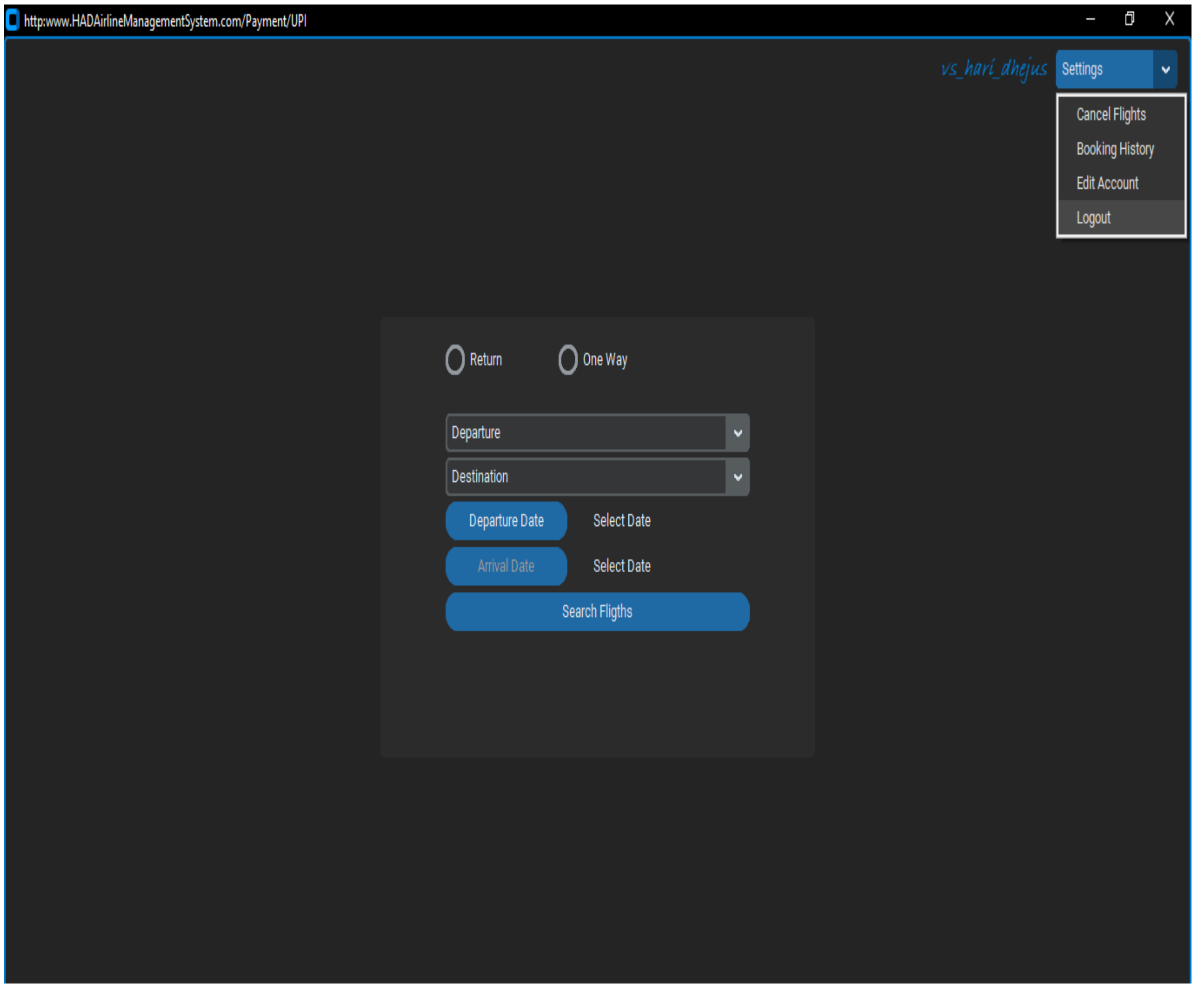
Password Show

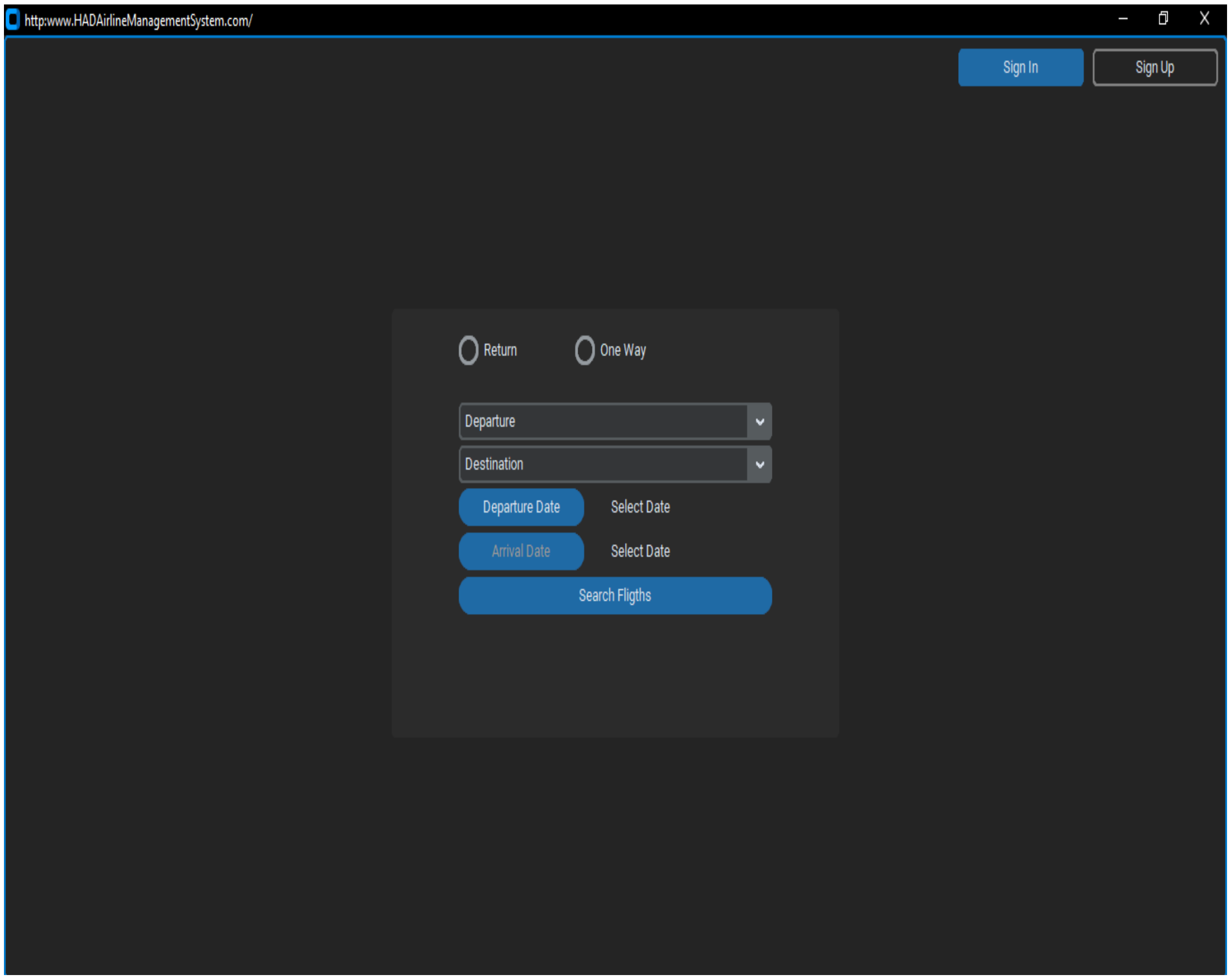
Re-Password Show

9789348287

Alter Account

Succesfully updated


http:www.HADAirlineManagementSystem.com/Payment/UPI
vs_hari_dhejus
Settings
Cancel Flights
Booking History
Edit Account
Logout
Return
One Way
Departure
Destination
Departure Date
Select Date
Arrival Date
Select Date
Search Fligths


http:www.HADAirlineManagementSystem.com/
Sign In
Sign Up
Return
One Way
Departure
Destination
Departure Date
Select Date
Arrival Date
Select Date
Search Fligths

# 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

- https://www.mysql.org