PROJECT REPORT ON

Bus Reservation System

Submitted By:

Ranjeet Saliya

Vishal Rathod

T. N. RAO COLLEGE, RAJKOT

Bachelor Of Computer Application

Year: 2024- 25

Project Guide:

Krishna Shukla

❖ Acknowledgement

I would like to express my heartfelt gratitude to everyone who has supported me in successfully completing this project. Every achievement is rooted in the encouragement, goodwill, and assistance of those around us.

As a Sixth-semester BCA student, I was required to undergo practical training in a software project, which provided me with valuable insights and expanded my knowledge in the field. I am thankful to all who contributed to the successful realization of this project.

I would especially like to acknowledge the contributions of everyone involved, both directly and indirectly, in this endeavour at the BCA Department of T.N. Rao College. This experience has greatly enriched my learning journey and will undoubtedly assist me in my future endeavours.

My sincere appreciation goes to Krishna Shukla and all the faculty members of our department for their unwavering inspiration and guidance throughout the project. Their support has been invaluable from the start to the completion of this undertaking.

Thank you all once again!

❖Preface

In today's fast-paced world, efficient and reliable transportation systems are essential for ensuring seamless connectivity and convenience for travelers. The Bus Reservation System project is a step towards modernizing and streamlining the process of booking and managing bus tickets, making it easier for passengers to plan their journeys and for administrators to manage operations effectively.

This project is designed to address the challenges faced by both passengers and bus operators in the traditional ticketing system. By leveraging technology, the Bus Reservation System aims to provide a user-friendly platform that simplifies the booking process, reduces manual errors, and enhances overall customer satisfaction. It offers features such as real-time seat availability, online payment integration, and automated ticket generation, ensuring a hassle-free experience for users.

INDEX

❖Abstract

The Bus Reservation System is a Python-based desktop application using Tkinter for the GUI and SQLite for database management. It automates bus ticket booking, allowing users to search buses, view seat availability, book tickets, and manage reservations. Administrators can update schedules, monitor bookings, and generate reports. SQLite ensures secure and efficient data storage, while Tkinter provides an intuitive interface. This project demonstrates the integration of programming, database management, and GUI development to create a scalable, user-friendly solution. By digitizing the reservation process, the system enhances efficiency, reduces errors, and supports sustainable practices, offering a modern approach to public transportation management.

❖Project Introduction

Project Overview:

- A **Bus Booking System** is a software application that allows passengers to book bus tickets online or through a simple interface. This system helps manage bus schedules, routes, ticket bookings. in a more organized and automated way.
- The goal of this project is to develop an efficient, user-friendly, and scalable platform for bus ticket booking that integrates seamlessly with backend systems, and route management.

Features:

The Bus Booking Project in Python comes with a variety of features that streamline the process of booking and managing bus tickets for both users and administrators. Below are the key features of the project:

1.Client panel:

• Route Search: Users can search for buses by specifying departure and destination locations, travel dates, and preferred times.

• Check Buses Details: shows a list of a buses with relevant details like departure time, pricing, and bus details (e.g., type of bus, amenities).

- Seat Booking: Users can book bus ticket themselves.

2.Admin Panel:

• Bus Details: Admins can add, edit, or remove bus schedules, routes. They can also manage bus timings and set the frequency of trips.

• Manage Bookings: Admins can view and manage all user bookings, including booking modifications and cancellations.

Technical Specifications:

Frontend

- **Python**:
  Used for structuring and styling the application.
- **Tkinter**:
  Provides the **Graphical User Interface (GUI)** for the system, ensuring a simple and intuitive user experience.

Backend

- **SQLite**:
  Used for **database management** storing.

❖Software and Hardware Requirements

- **Hardware Configuration**

Server Configuration:

Pentium IV processor

512 MB RAM

256 Cache memory

80GB HDD

104 Keyboard

1.44 MB

Optical Mouse

- **Client Configuration**

128 MB or higher RAM

64 Cache memory

16GB HDD 7

- **Software Configuration**

• Window – 11

• SQLite database

• Front-end tool- Python

• Vs code

❖Tools and technology used

Tools and Technologies Used in Bus Reservation System Project

- **Programming Language**

Python: A versatile, high-level programming language used for developing the core logic and functionality of the system.

- **Graphical User Interface (GUI):**

Tkinter

- **Database:**

SQLite

- **Editor:**

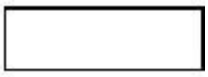
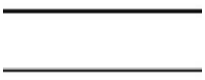
Visual Studio code

❖System analysis and design - System Analysis, System Design (DFDs)

Data flow diagram

A data flow diagram is a graphical view of how data is processed in a system in terms of input and output. The data flow diagram (DFD) contains some symbol for drawing the data flow diagram. Explanation of main symbols which are used in data flow diagram is given below:

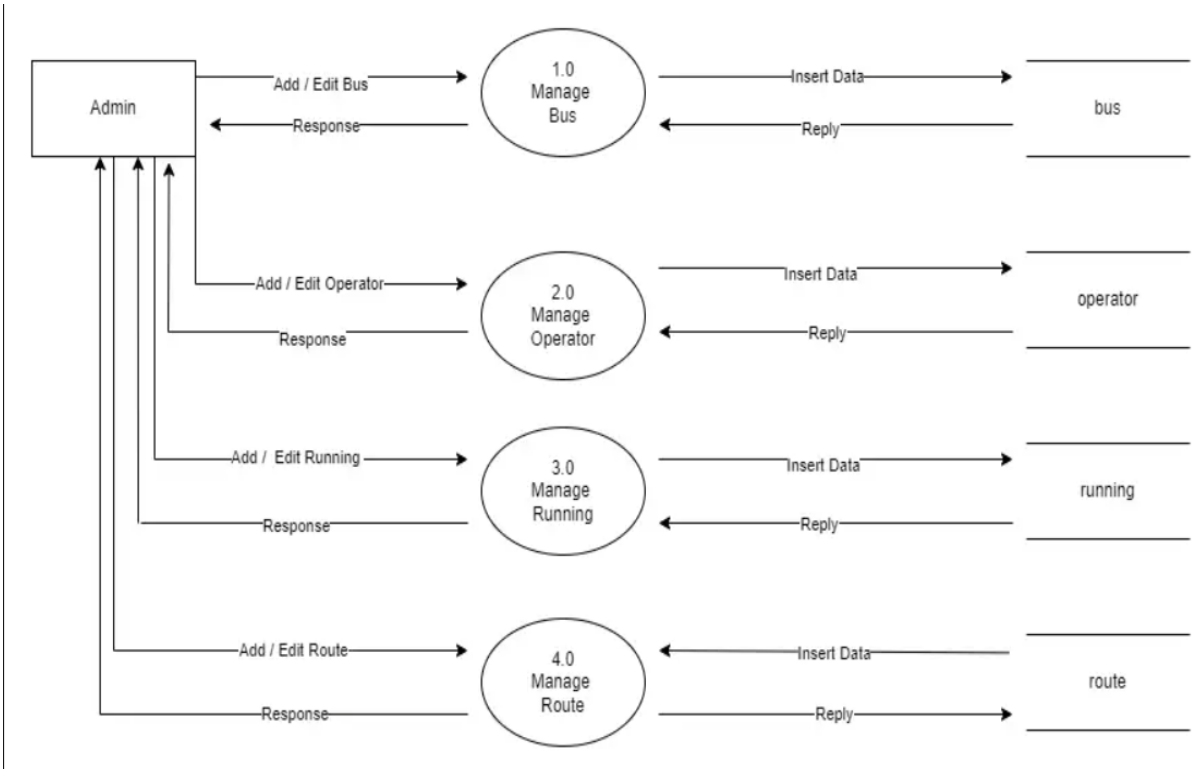
Data flow diagram symbol:

Symbol	Description
	**Data Flow** – Data flow are pipelines through the packets of information flow.
	**Process:** A Process or task performed by the system.
	**Entity:** Entity are object of the system. A source or destination data of a system.
	**Data Store:** A place where data to be stored.

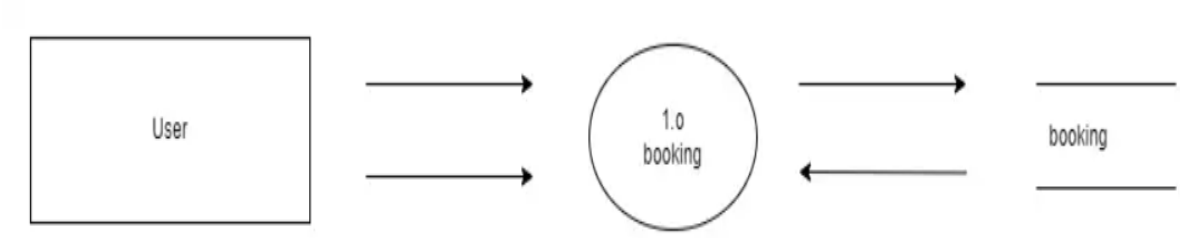
❖Data Flow Diagram:

- **0 level (admin)**

- **1 level (Admin panel)**



- **1 level (Client Panel)**



❖Data Dictionary

- **Table Name : Operator**

operator		CREATE TABLE operator (opr_id TEXT PRIMARY KEY,
opr_id	TEXT	"opr_id" TEXT
name	TEXT	"name" TEXT
address	TEXT	"address" TEXT
phone	TEXT	"phone" TEXT CHECK(length("phone") = 10)
email	TEXT	"email" TEXT

- **Table Name : Bus**

bus		CREATE TABLE bus (bus_id TEXT PRIMARY KEY,
bus_id	TEXT	"bus_id" TEXT
bus_type	TEXT	"bus_type" TEXT
capacity	INTEGER	"capacity" INTEGER
op_id	TEXT	"op_id" TEXT NOT NULL
route_id	TEXT	"route_id" TEXT NOT NULL

- **Table Name : Route**

route		CREATE TABLE route (r_
r_id	TEXT	"r_id" TEXT
s_name	TEXT	"s_name" TEXT
s_id	TEXT	"s_id" TEXT
e_name	TEXT	"e_name" TEXT
e_id	TEXT	"e_id" TEXT

- **Table Name : Running**

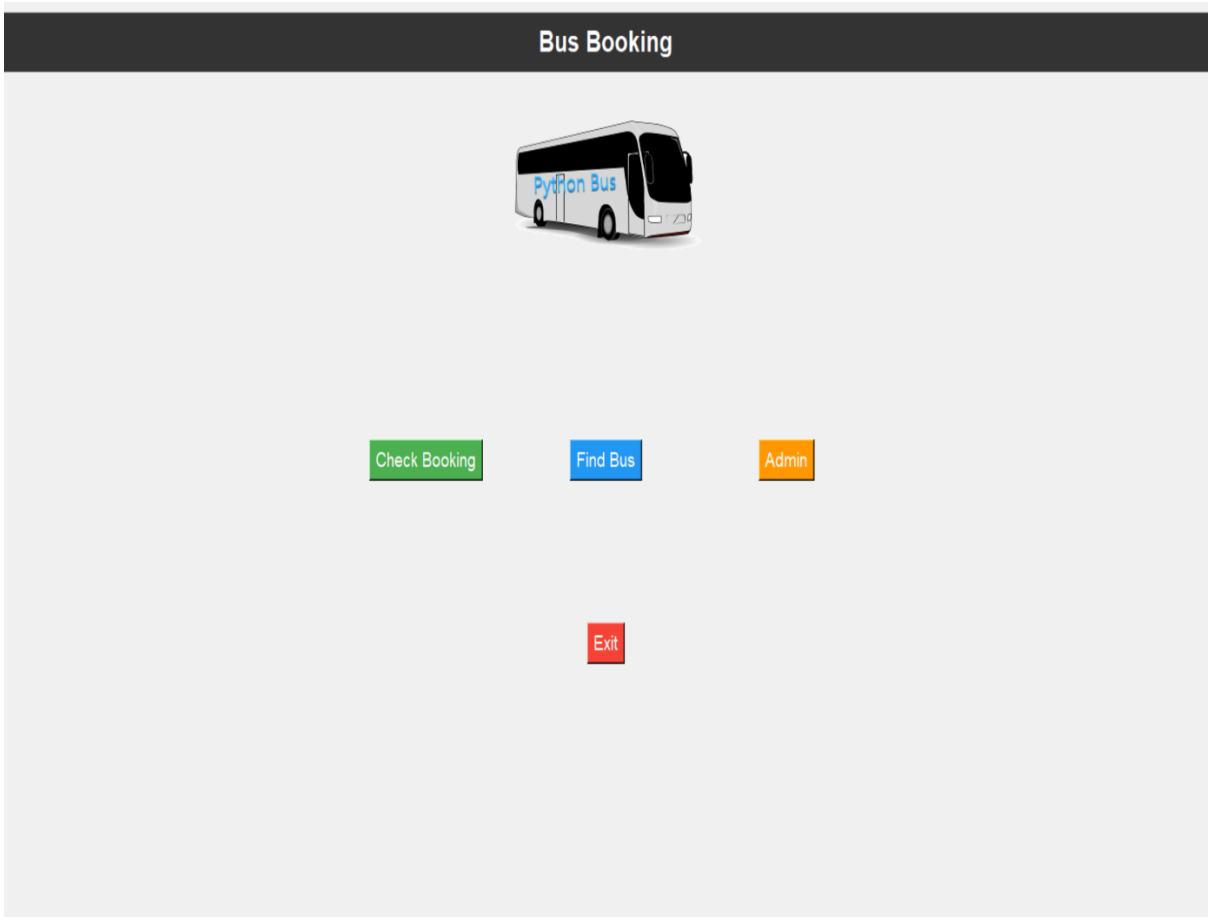
running		CREATE TABLE running (b_id TEXT,
b_id	TEXT	"b_id" TEXT
run_date	DATE	"run_date" DATE
seat_avail	INTEGER	"seat_avail" INTEGER

- **Table Name : Booking**

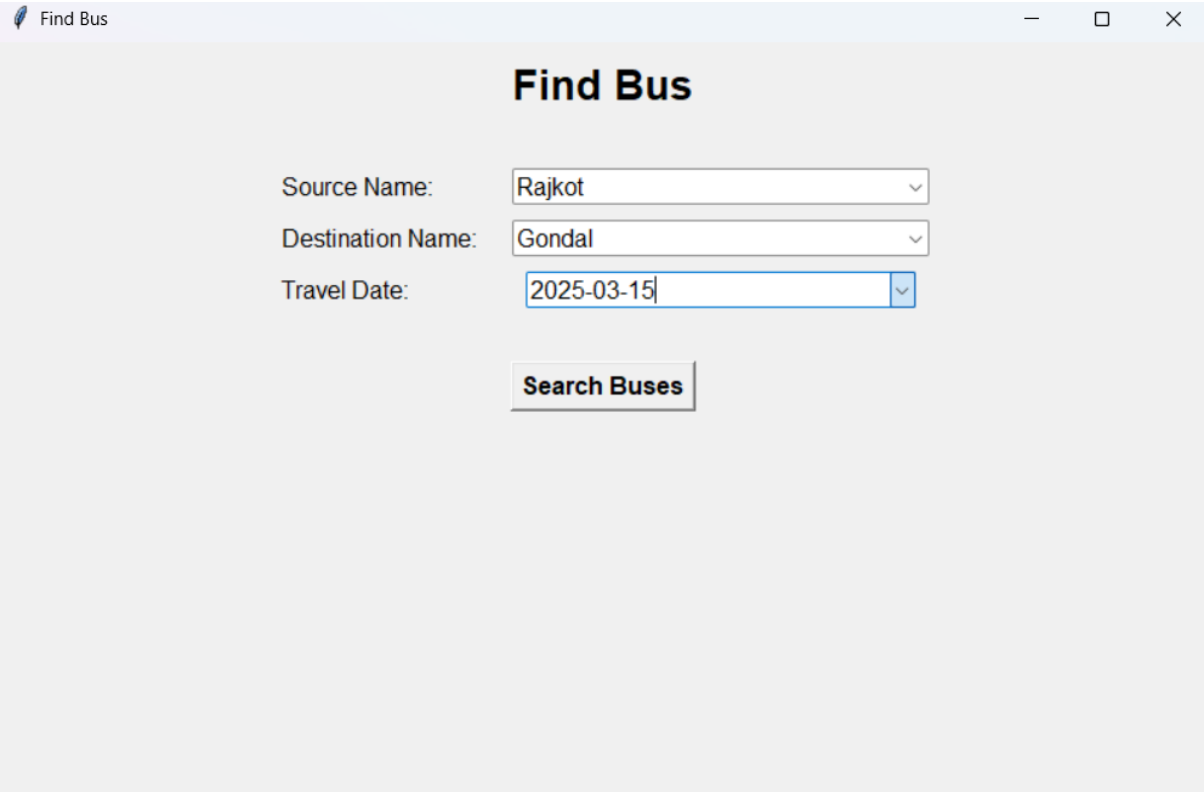
booking		CREATE TABLE booking (booking_id INTEGER
booking_id	INTEGER	"booking_id" INTEGER
b_id	TEXT	"b_id" TEXT NOT NULL
run_date	DATE	"run_date" DATE NOT NULL
user_name	TEXT	"user_name" TEXT NOT NULL
contact	TEXT	"contact" TEXT NOT NULL
seat_number	INTEGER	"seat_number" INTEGER NOT NULL

❖Project Implementation- Screenshot

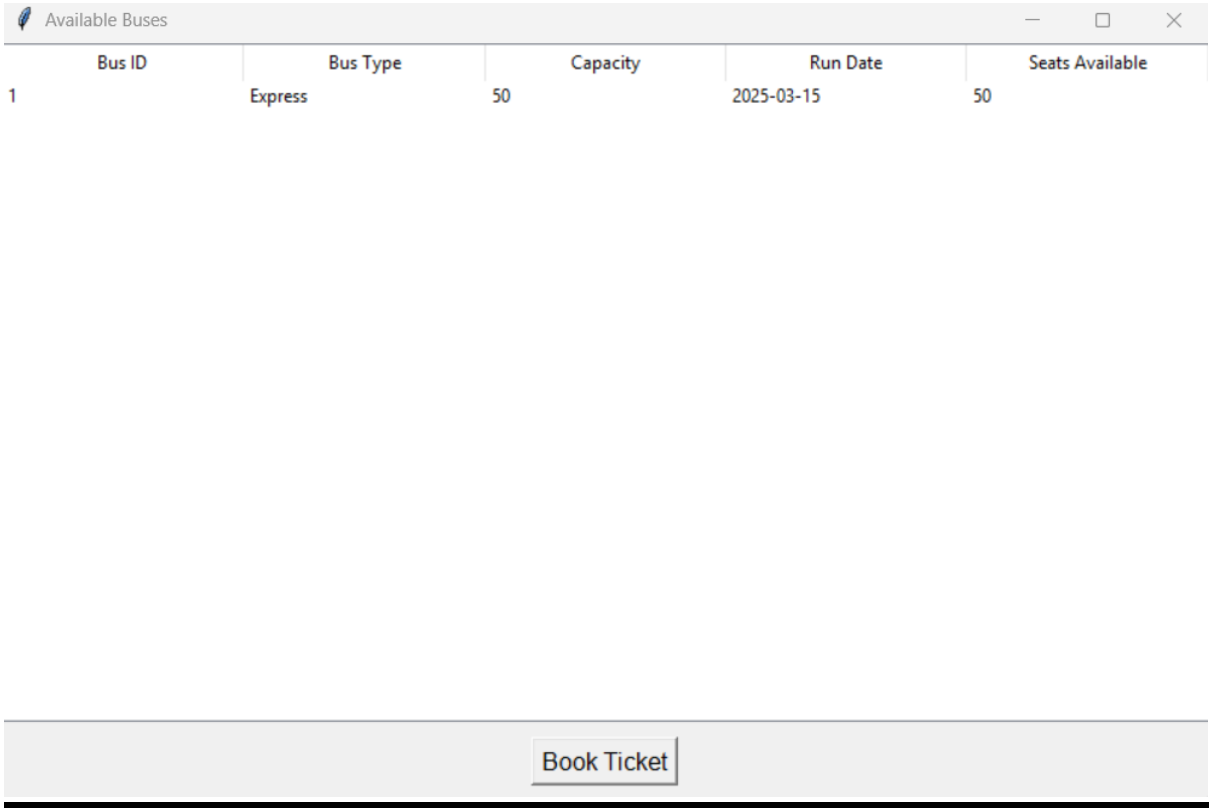
- **Client Panel**



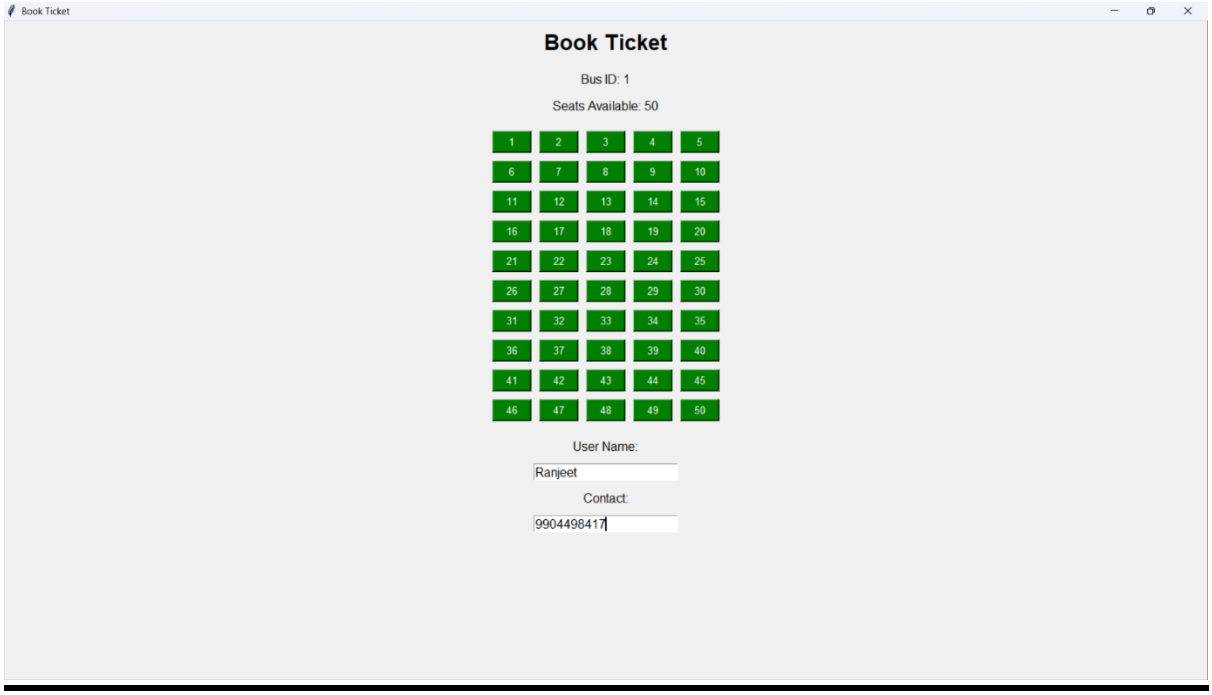
- **Find bus:**


Find Bus
Find Bus
Source Name:
Rajkot
Destination Name:
Gondal
Travel Date:
2025-03-15
Search Buses

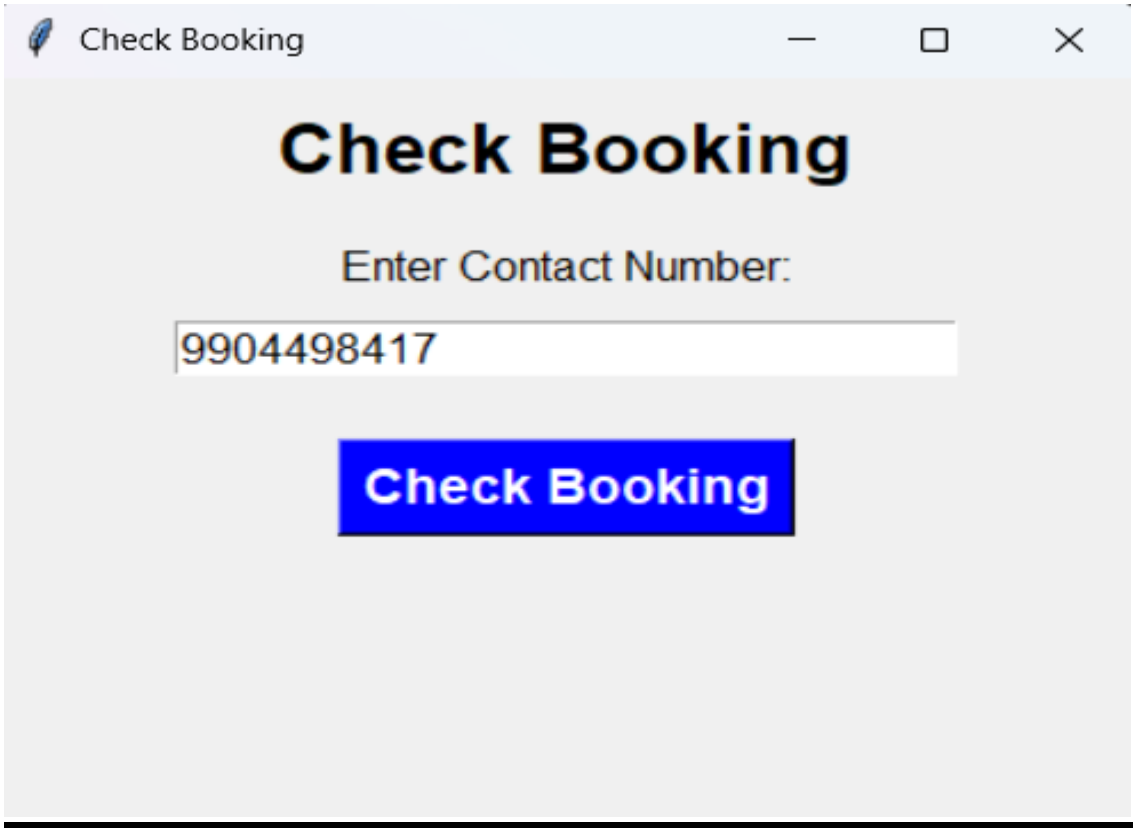
- **Select bus:**


Available Buses
Bus ID
Bus Type
Capacity
Run Date
Seats Available
1
Express
50
2025-03-15
50
Book Ticket

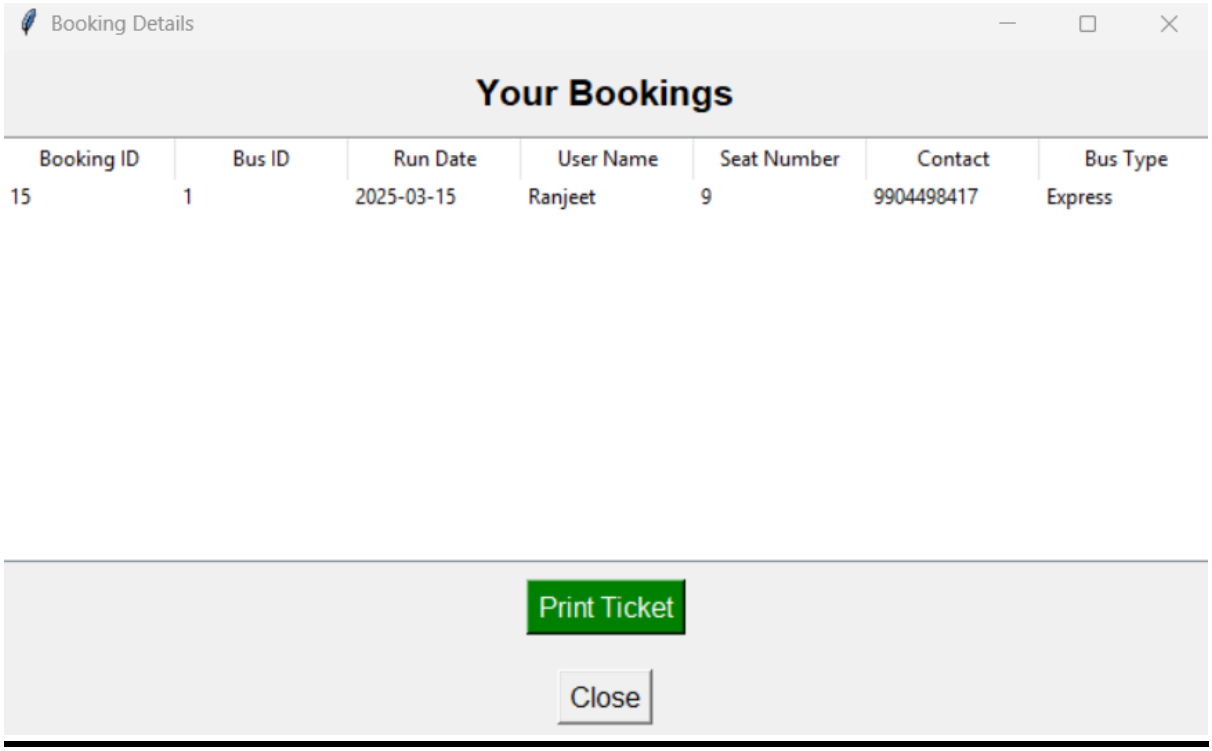
- **Book Ticket:**


Book Ticket
Book Ticket
Bus ID: 1
Seats Available: 50
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
26 27 28 29 30
31 32 33 34 35
36 37 38 39 40
41 42 43 44 45
46 47 48 49 50
User Name:
Ranjeet
Contact:
9904498417

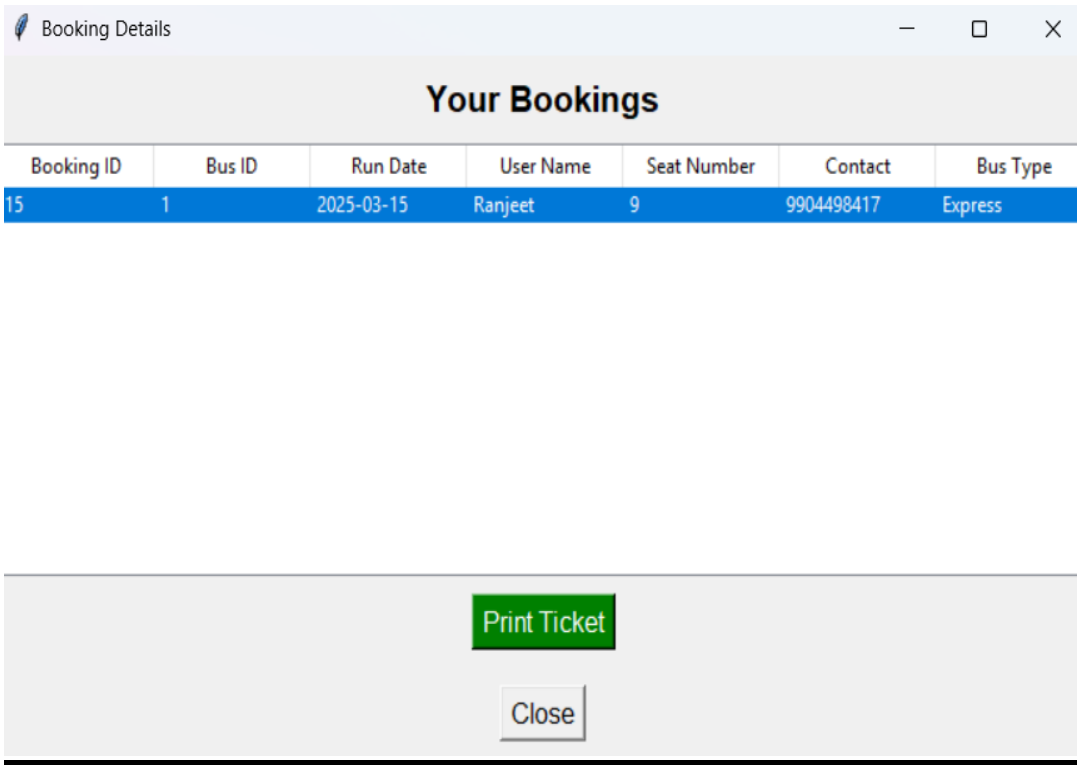
- **Find Ticket:**


Check Booking
Check Booking
Enter Contact Number:
9904498417
Check Booking

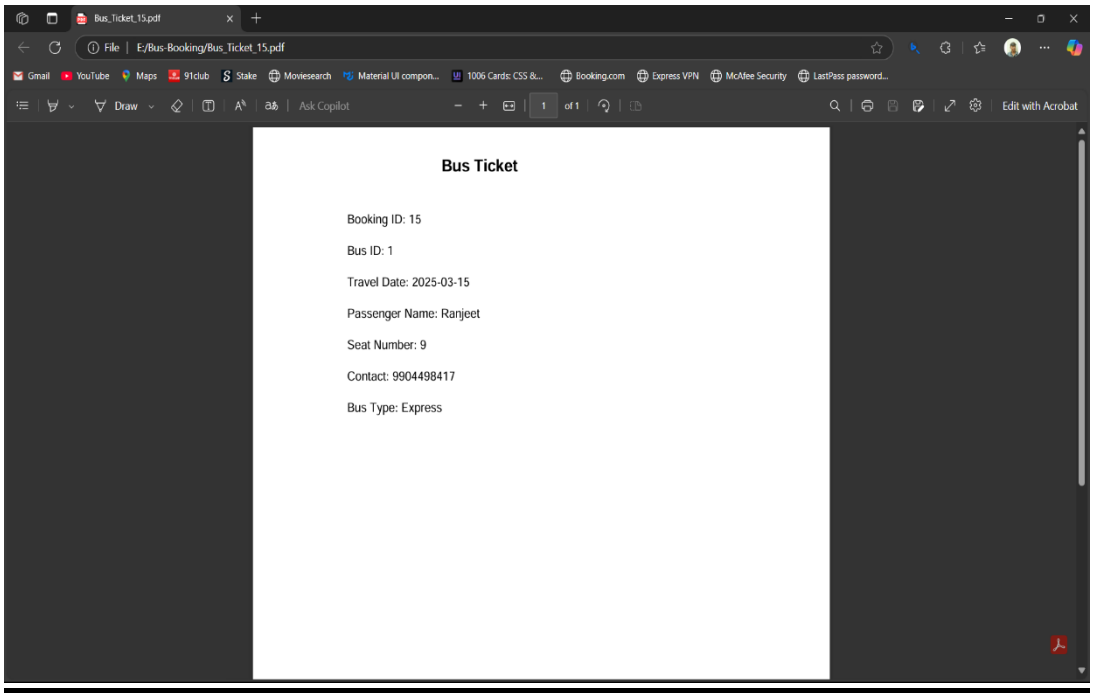
- **Show Ticket:**


Booking Details
Your Bookings
Booking ID
Bus ID
Run Date
User Name
Seat Number
Contact
Bus Type
15
1
2025-03-15
Ranjeet
9
9904498417
Express
Print Ticket
Close

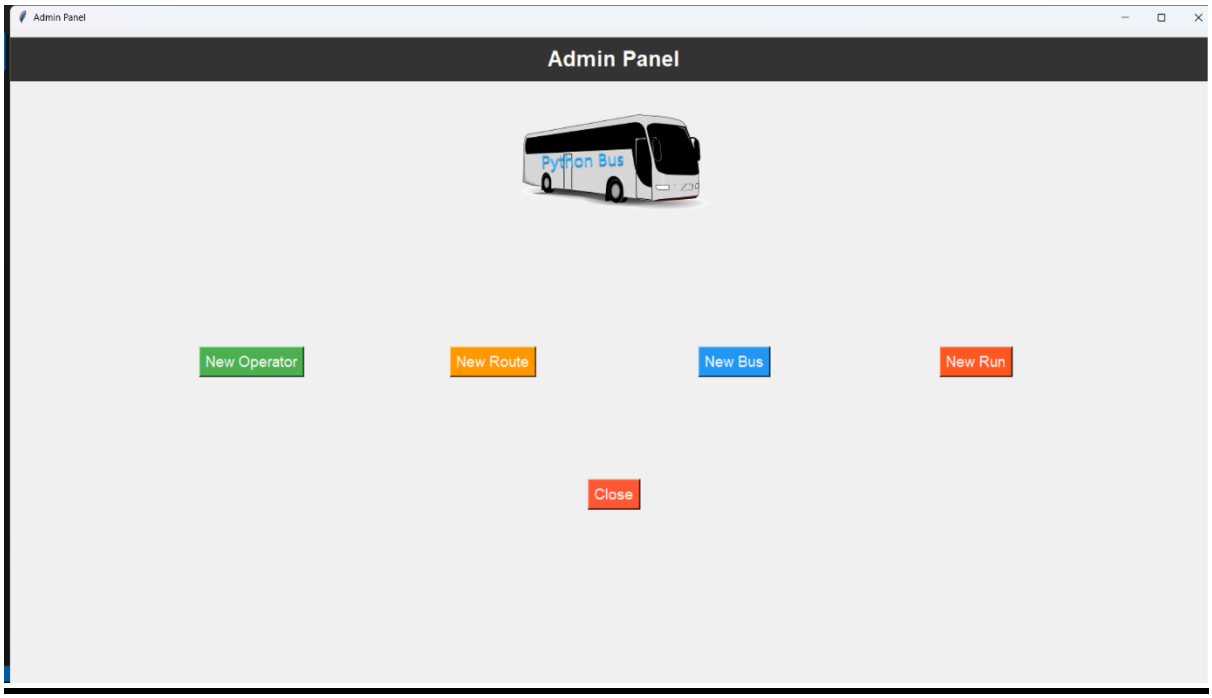
- **Print Ticket**



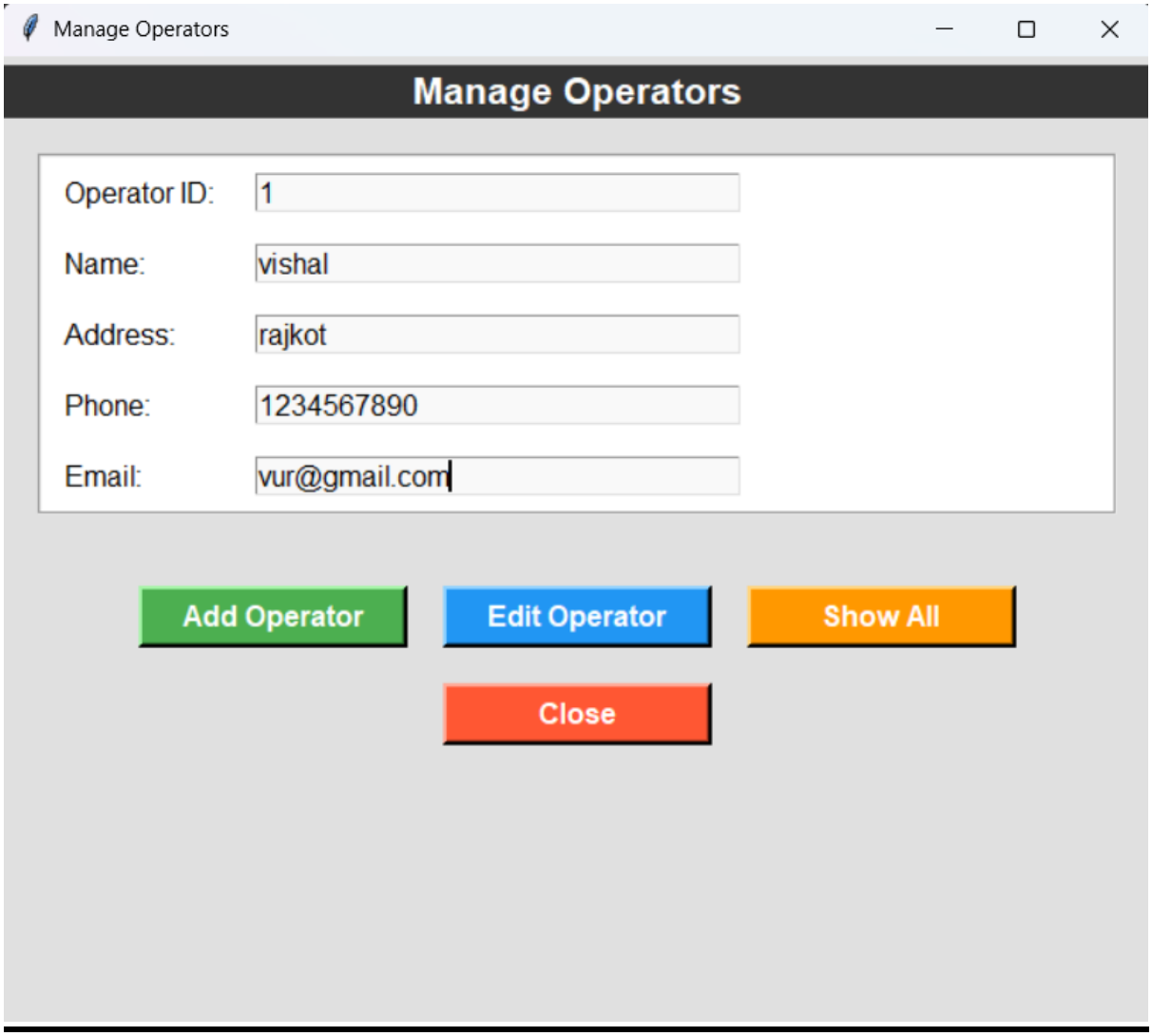
- **Ticket**


Bus_Ticket_15.pdf
File | E:/Bus-Booking/Bus_Ticket_15.pdf
Gmail
YouTube
Maps
91club
Stake
Moviesearch
Material UI compon...
1006 Cards: CSS &...
Booking.com
Express VPN
McAfee Security
LastPass password...
Draw
Ask Copilot
1
of 1
Edit with Acrobat
Bus Ticket
Booking ID: 15
Bus ID: 1
Travel Date: 2025-03-15
Passenger Name: Ranjeet
Seat Number: 9
Contact: 9904498417
Bus Type: Express

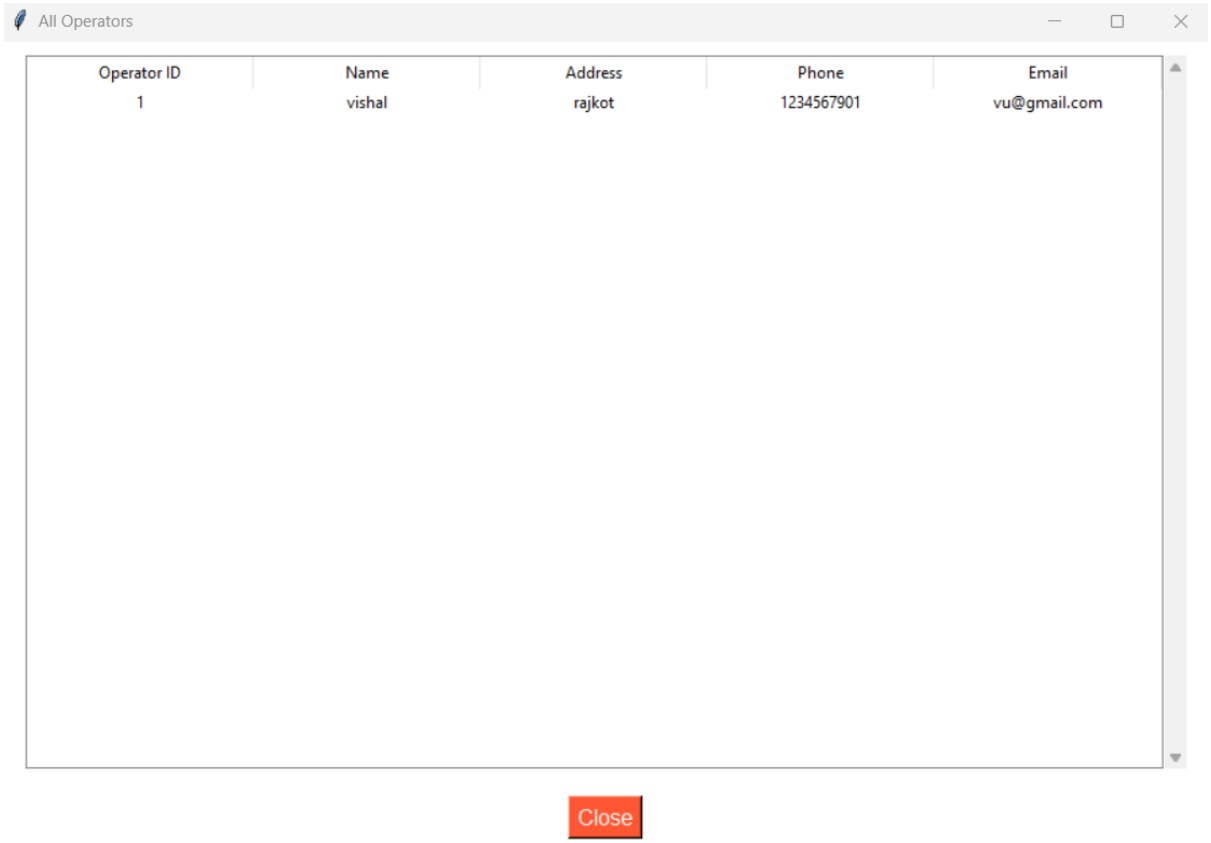
- **Admin Panel:**


Admin Panel
Admin Panel
Python Bus
New Operator
New Route
New Bus
New Run
Close

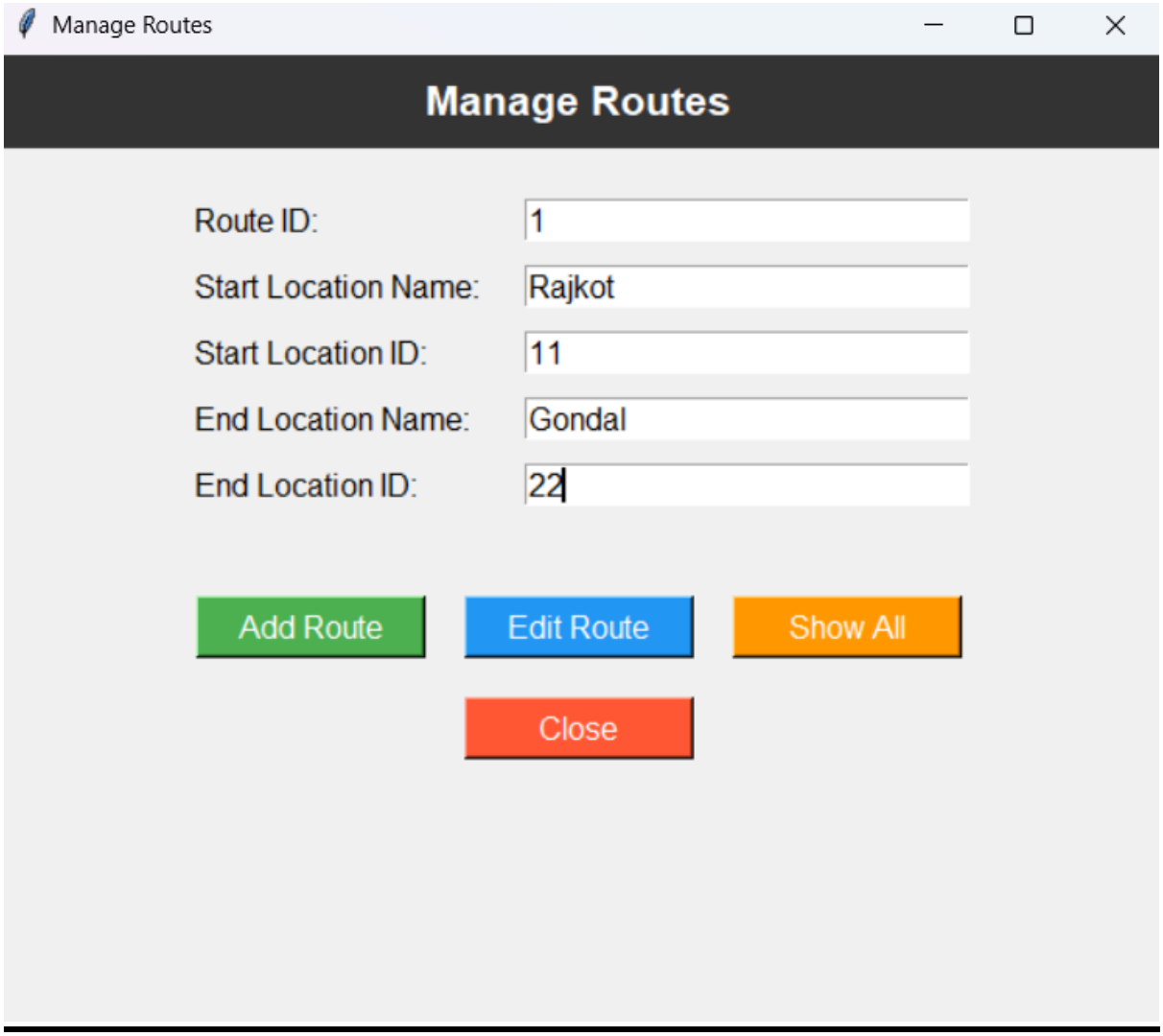
- **Add Operator:**



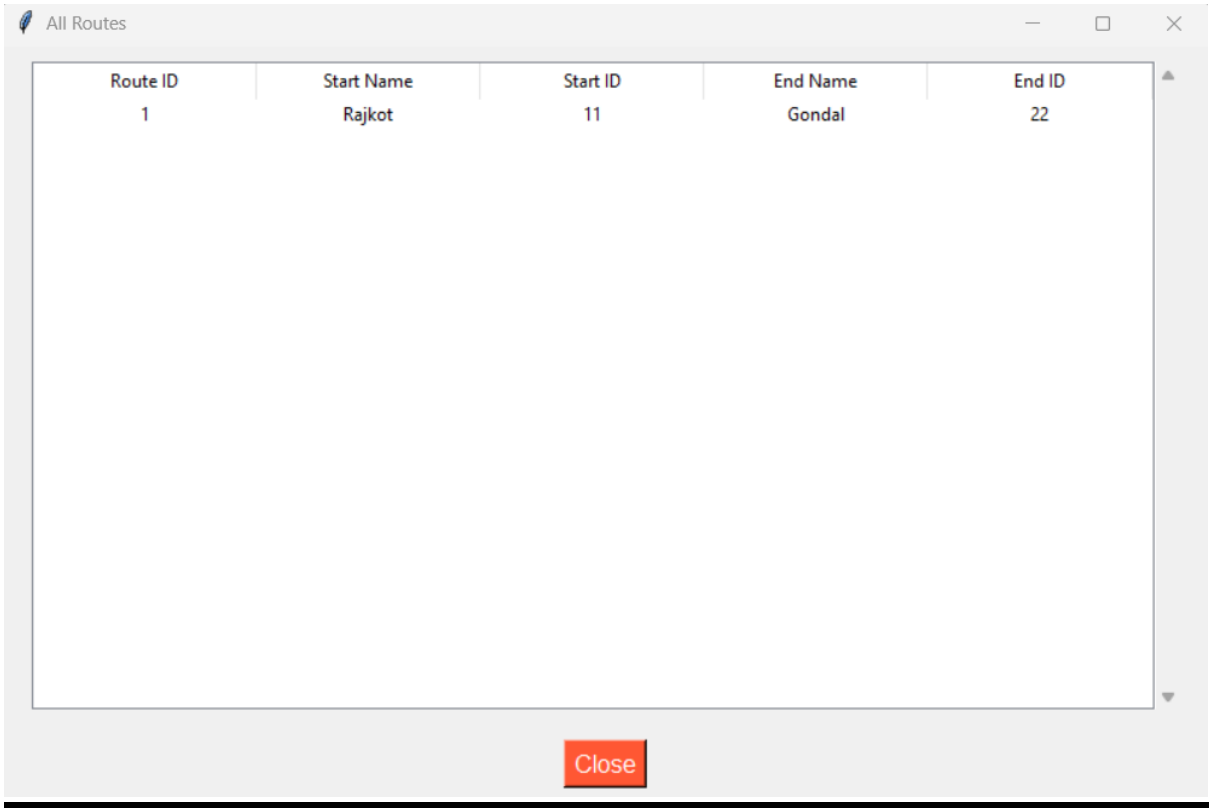
- **Show Operators:**



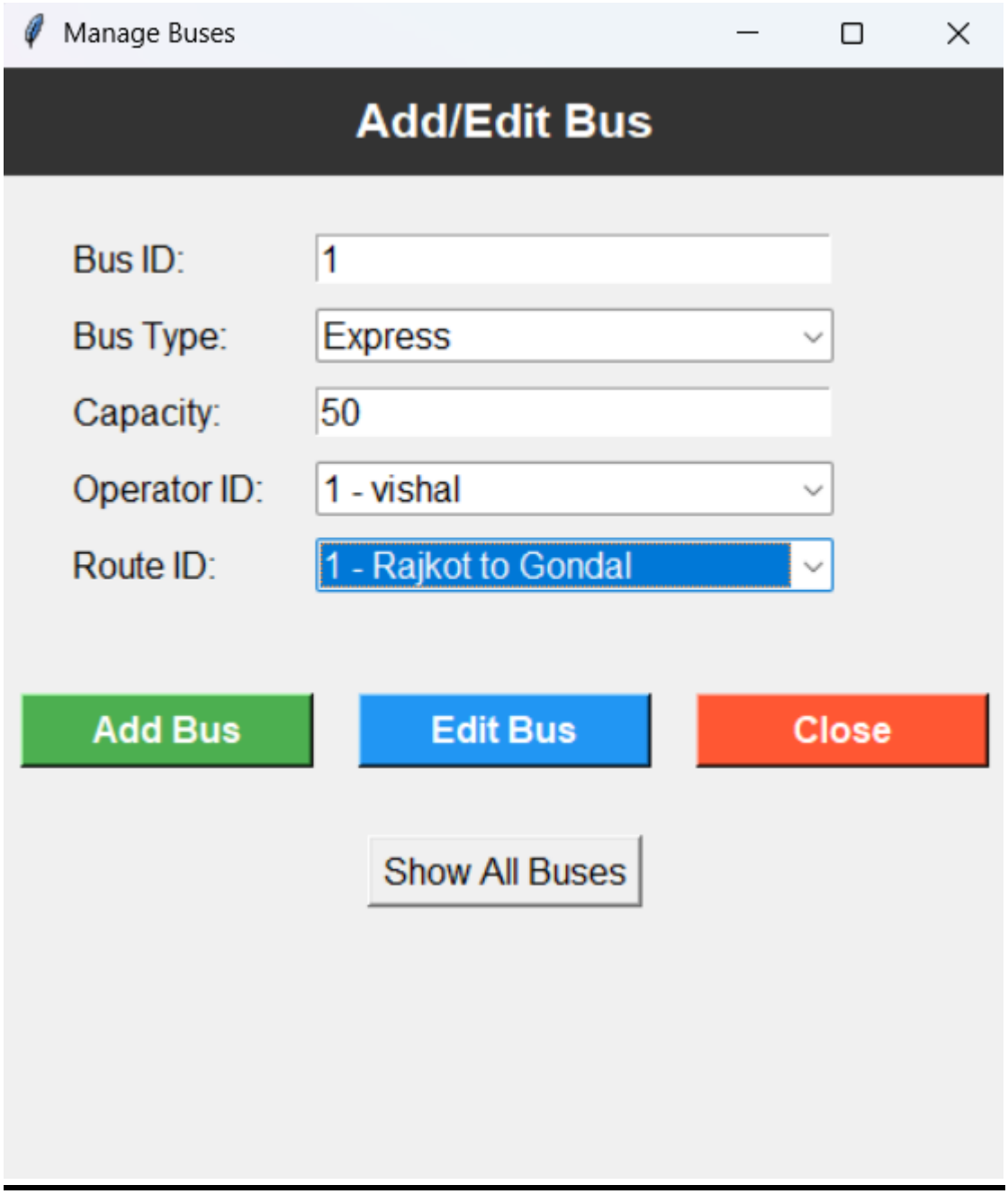
- **Add Route:**


Manage Routes
Manage Routes
Route ID:
1
Start Location Name:
Rajkot
Start Location ID:
11
End Location Name:
Gondal
End Location ID:
22
Add Route
Edit Route
Show All
Close

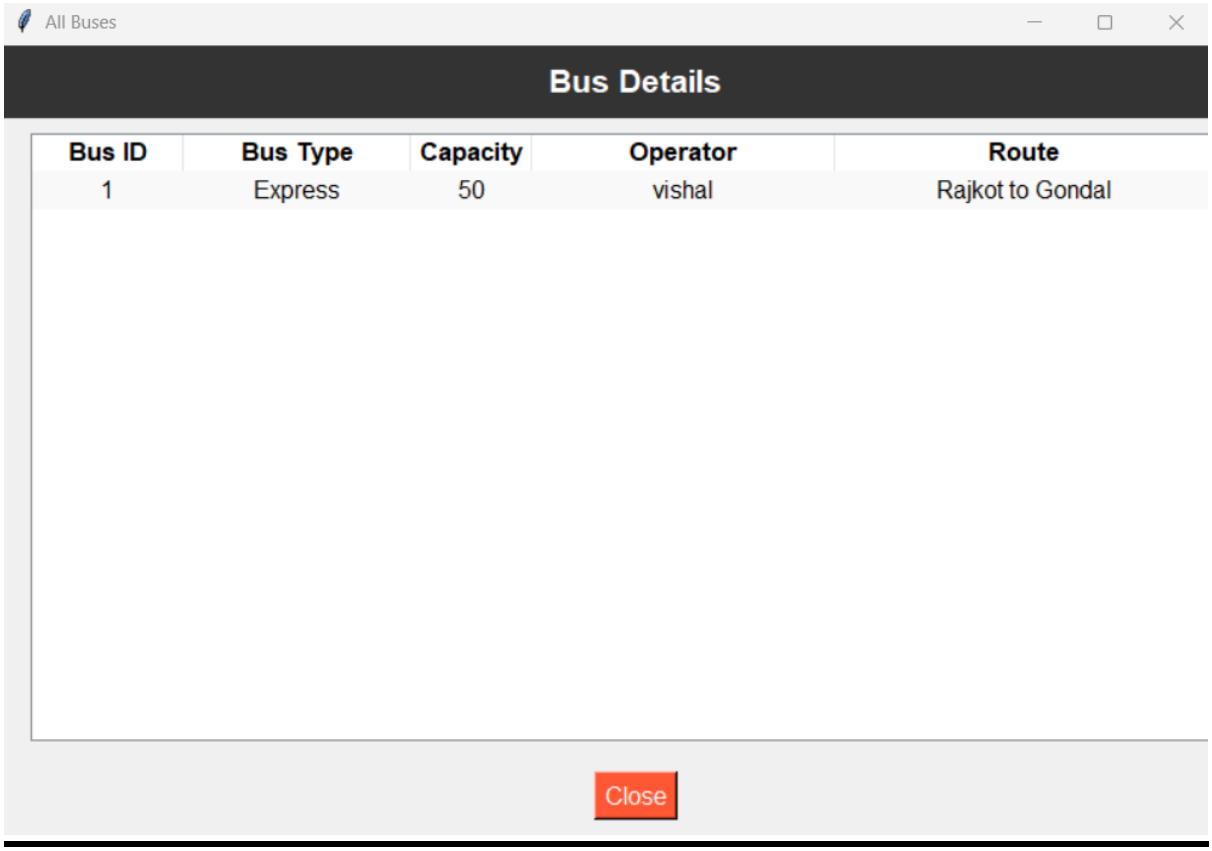
- **Show Routes:**


All Routes
Route ID
Start Name
Start ID
End Name
End ID
1
Rajkot
11
Gondal
22
Close

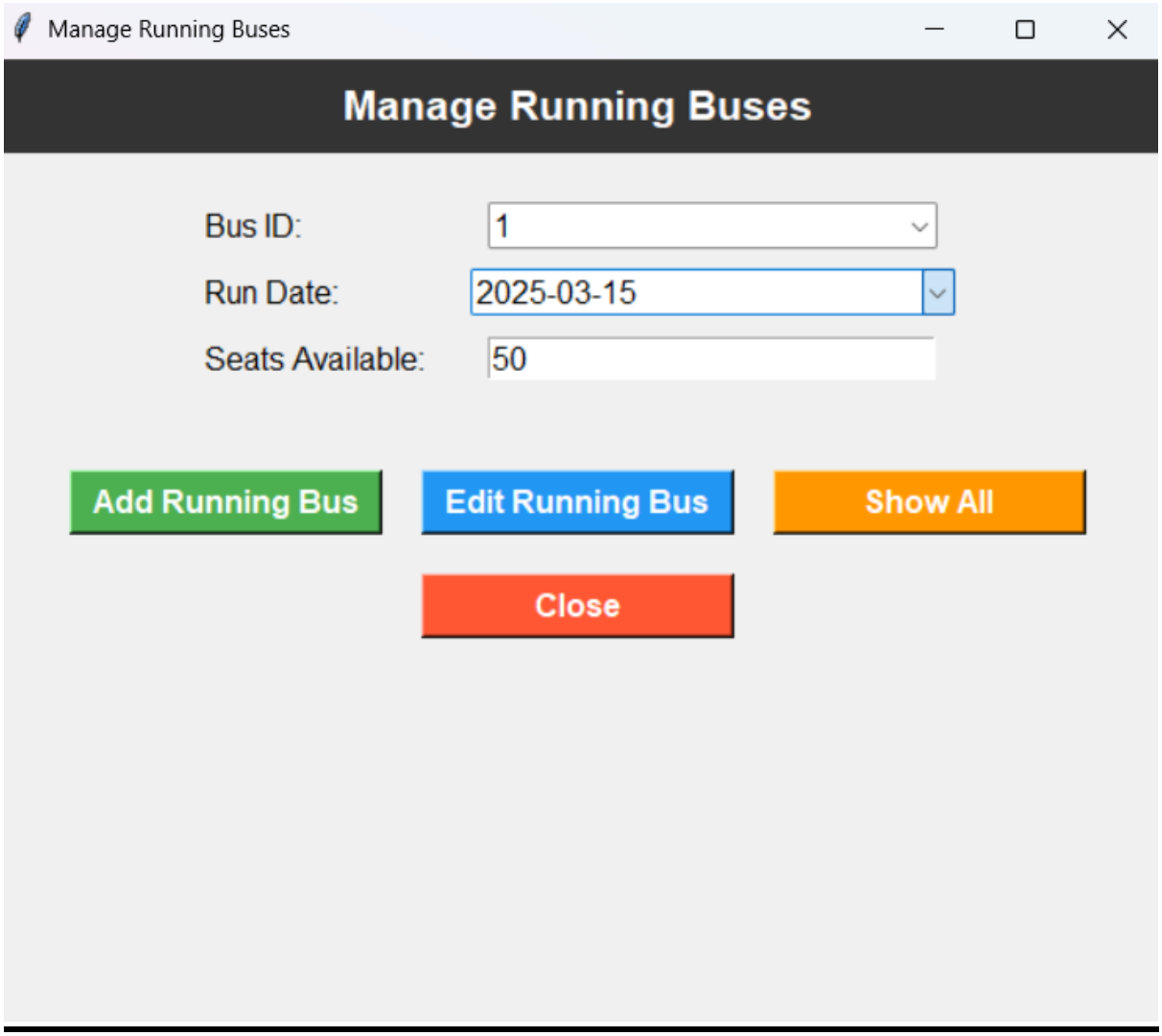
- **Add Bus:**


Manage Buses
Add/Edit Bus
Bus ID:
1
Bus Type:
Express
Capacity:
50
Operator ID:
1 - vishal
Route ID:
1 - Rajkot to Gondal
Add Bus
Edit Bus
Close
Show All Buses

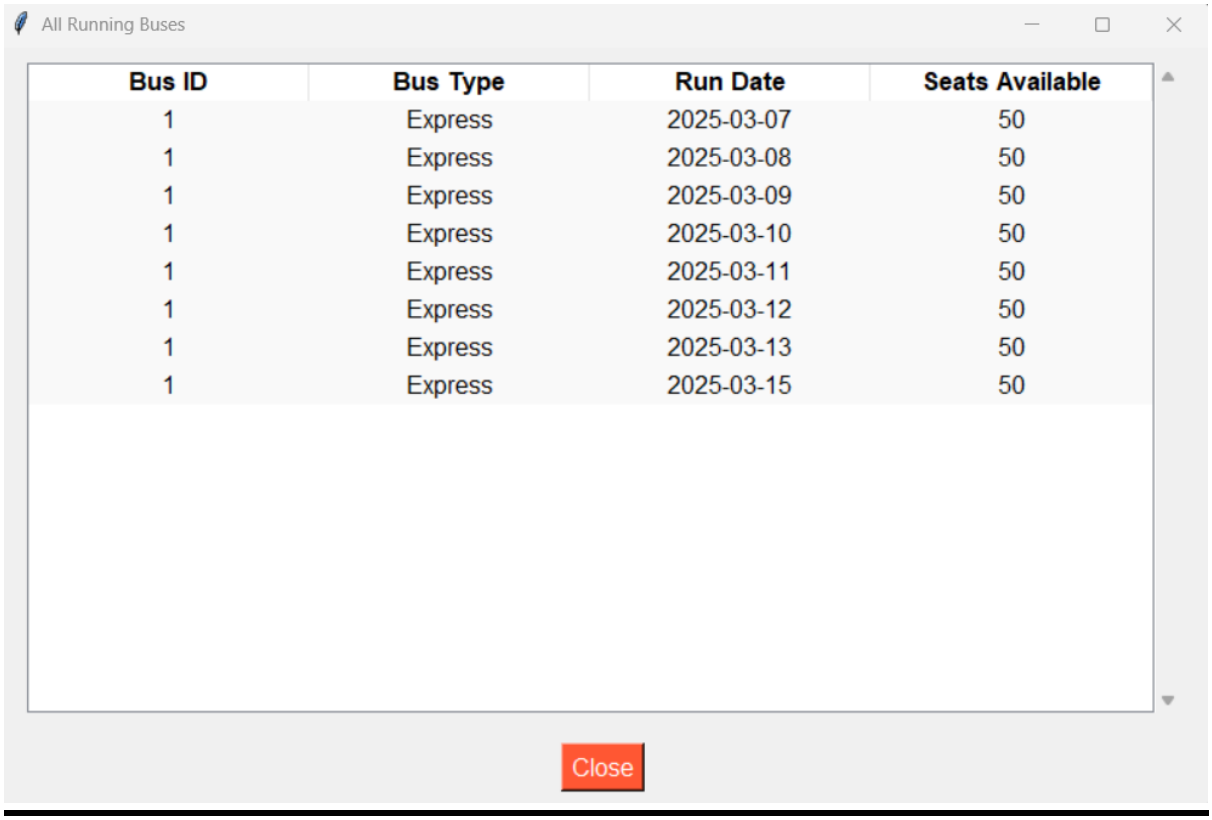
- **Show bus:**


All Buses
Bus Details
Bus ID
Bus Type
Capacity
Operator
Route
1
Express
50
vishal
Rajkot to Gondal
Close

- **Add Running Bus:**



- **Show running bus:**



❖Testing

Testing Standards

- Software testing is the process of evaluating and verifying that a system or component functions as intended.
- The primary goal of testing is to identify and fix defects before software deployment.
- It ensures the quality, performance, security, and usability of the system.
- Testing should include edge cases and boundary values to verify the robustness of the application.
- Inputs outside the specified range should be tested to check the system's error-handling capabilities.

Testing Strategies

- A test strategy defines the overall approach, scope, objectives, resources, and schedule for testing.
- It ensures alignment between testing efforts and project requirements.
- An efficient testing strategy involves testing all levels of the system to prevent failures.

1 Black Box Testing

- Black box testing focuses on functionality without knowledge of the internal code structure.
- The tester interacts with the user interface and validates expected outputs based on given inputs.
- It ensures that the system meets user requirements and behaves correctly.
- Common techniques include equivalence partitioning, boundary value analysis, and error handling tests.

- Example:
- Verifying if a bus search function returns available buses for a given route and date.

- **2 White Box Testing**
- White box testing examines the internal logic, code structure, and flow of the application.
- It requires knowledge of the source code, including loops, conditions, and database queries.
- The goal is to ensure all parts of the code execute correctly under different scenarios.
- Techniques include unit testing, control flow testing, data flow testing, and branch testing.
- Example:
- Testing a fare calculation function to verify correct pricing for multiple seats.

- **3 Gray Box Testing**
- Gray box testing is a hybrid approach that combines black box and white box testing principles.
- Testers have partial knowledge of the internal workings but focus on functional validation.
- It helps in testing database interactions, API calls, and security vulnerabilities.
- Example:
- Checking whether a bus seat booking is correctly updated in the database after confirmation.

- **Test case:**

Test step and expected result:

Step No	Test Scenario	Input Data	Expected Result	Status
1	Positive Scenario: Find Ticket Valid Contact Number	Contact Number Entered	User Successfully Find Booked Ticket With detail	Pass
2	Negative Scenario: Find Ticket Invalid Contact Number	Wrong Contact Number Entered	Error Message “Invalid Contact Number “	Pass

❖Limitation and future Enhancement

Limitations of the Bus Reservation System

1. **Limited Scalability:**

- SQLite, while lightweight, may not handle large-scale data efficiently for high-traffic systems.

2. **No Online Payment Integration:**

- The system currently lacks integration with online payment gateways, requiring manual payment handling.

3. **Single-Platform Support:**

- The application is designed as a desktop-based system and does not support web or mobile platforms.

4. **Basic UI/UX:**

- The Tkinter-based GUI, while functional, may not offer the most modern or visually appealing user experience.

5. **No Real-Time Updates:**

- The system does not support real-time updates or notifications for users, such as seat availability changes.

6. **Limited Security Features:**

- Basic authentication and data encryption are not implemented, which may pose security risks.

7. **No Multi-Language Support:**

- The system currently supports only one language, limiting accessibility for non-English users.

Future Enhancements

1. **Web and Mobile Integration:**

- Develop web and mobile versions of the system using frameworks like Flask/Django (web) and Kivy (mobile).

2. **Online Payment Gateway:**

- Integrate payment gateways like PayPal, Stripe, or Razorpay for seamless online transactions.

3. **Advanced Database Management:**

- Migrate to a more robust database system like MySQL or PostgreSQL for better scalability and performance.

4. **Real-Time Updates:**

- Implement real-time notifications and updates using technologies like Web-Sockets or Firebase.

5. **Enhanced Security:**

- Add features like password encryption, two-factor authentication, and secure data storage.

6. **Improved UI/UX:**

- Use modern GUI frameworks like PyQt for a more visually appealing and user-friendly interface.

7. **Multi-Language Support:**

- Add support for multiple languages to make the system accessible to a global audience.

8. **Admin Dashboard**:

- Develop a comprehensive admin dashboard with advanced analytics and reporting tools.

9. **Integration with GPS and Maps**:

- Include real-time bus tracking and route mapping using APIs like Google Maps.

10. **Cloud Integration**:

- Store data on cloud platforms like AWS or Google Cloud for better accessibility and backup.

By addressing these limitations and implementing future enhancements, the Bus Reservation System can evolve into a more robust, scalable, and user-friendly solution for modern transportation needs.

❖Bibliography:

- https://www.tutorialspoint.com
- https://www.w3schools.com
- https://www.geeksforgeeks.org
- https://www.sqlitetutorial.net
- https://pythonexamples.org
- https://realpython.com

THANK YOU...!