JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

ODD SEMESTER 2025 (SEMESTER 3)

DATA STRUCTURES LAB (15B17CI371)

PROJECT BASED LEARNING REPORT:

“Smart Travel And Tourism Planner”

Submitted By:

1. Arolika Shanker-2401030034
2. Kirpa Gupta-2401030036
3. Vansh Verma-2401030039
4. Shreyanshi Agarwal-2401030045
5. Yogyata Bhatnagar-2401030056

Submitted to:

Dr. K.Rajlakshmi
Dr. Amitesh

Batch: B-10

ACKNOWLEDGEMENT

We express our heartfelt gratitude to our respected faculty members for their constant guidance, encouragement, and support throughout the development of this project. Their expert insights, valuable feedback, and thoughtful suggestions at every stage greatly contributed to enhancing the quality, structure, and overall direction of Smart Travel and Tourism Planner. Their dedication to teaching and their willingness to help us overcome challenges played an essential role in shaping our understanding of the subject.

We would also like to extend our sincere thanks to our friends and families who consistently motivated us during the entire project journey. Their feedback during the testing phase, suggestions for improvement, and continuous moral support helped us refine the features, strengthen the usability, and improve the functionality of the application. Their involvement ensured that the project evolved into a more user-friendly and efficient system.

Finally, we express our appreciation to everyone who directly or indirectly contributed to the successful completion of the Smart Travel and Tourism Planner. This project has not only improved our understanding of data structures but has also given us valuable exposure to practical data management, teamwork, and real-world problem-solving. The experience gained through this project will be extremely beneficial for our future academic and technical pursuits.

INTRODUCTION

The Smart Travel and Tourism Planner is a console-based C++ application designed to ease and organize the entire process of trip planning. In today’s world, travellers often face difficulty in comparing flights, managing bookings, finding the best prices, and selecting good hotels. This project aims to bring all these tasks into one single system that is easy to use, efficient, and reliable.

This application provides the facility of flight searching, cost comparison, finding the cheapest direct option, and instant ticket booking. Other facilities provided by this application are passenger check-in systems and recommendations for hotels, sorted in order of ratings. To make this system realistic and efficient, a variety of Data Structures are used in this project, including graphs for storing the routes of flights, priority queues for selecting the best deals, stacks for implementing an undo operation for bookings, a queue for check-in processing, and binary search trees for hotel management. Data is persistently saved through file handling, which makes this system act almost like a small, real-world travel planner.

PROBLEM STATEMENT

In real life, planning a trip involves checking flights, comparing prices, managing bookings, and finding hotels. Doing all of this manually is time-consuming and confusing. There is a need for a simple system that organizes travel data and helps users make quick and efficient travel decisions which helps in making trip planning less hectic. Our project solves this by providing a single platform that handles flights, bookings, check-ins, and hotels.

OBJECTIVES

1. To create a user-friendly travel planning system using Data Structures.

2. To store and manage flight information efficiently using graphs.

3. To allow users to find the cheapest direct flight quickly.

4. To maintain booking history with the ability to undo actions.

5. To manage passenger check-ins using a queue.

6. To recommend top-rated hotels using a Binary Search Tree.

7. To implement file handling for saving and loading data permanently.

TOPICS USED

1. Graph (Adjacency List): To store cities and flight connections.
2. Priority Queue: To find the cheapest flight (best deal).
3. Hash Maps: For fast flight lookups based on source-destination.
4. Stack: Used to keep track of all bookings so the last booking can be undone easily.
5. Queue: Used to manage passenger check-in, where passengers are served in the order they arrive.
6. Binary Search Tree: To store hotels and display top-rated ones.
7. File Handling: To save and load flights, bookings, and hotel data.
8. Structures & Classes: To organize data cleanly (Flight, Booking, Hotel).

FEATURES OF THE PROJECT

1. Add and store flights with their distance, time, and cost.
2. Show all the direct flights between two cities.
3. It automatically finds the cheapest direct flight.
4. Book flight tickets and undo the latest booking.
5. Display all bookings made by the user.
6. Passenger check-in and processing system.
7. Add hotels and list the top-rated ones.
8. Automatic saving/loading of all data to/from files.

FLOW AND WORKING OF THE PROJECT

1. Data Loading:

   The program, upon startup, loads all previously saved flights, bookings, and hotels from text files.

2. User Menu:

   The Main menu has options like add flights, view flights, book tickets, check in, hotel management, etc.

3. Flight Management:

   Adding a flight updates the graph and is saved to the file.

   Searching uses the FlightLookup class to quickly find direct flights.

   Best Deal uses a priority queue to pick the lowest-cost flight.

4. Booking Management:

   A booking creates a new entry in a vector.

   A stack for storing bookings for undo support.

   Undo removes the last booking done.

5. Check-In Queue:

   Passengers are added to a queue and processed in FIFO order.

6. Hotel Management:

   Hotels are inserted into a BST, enabling efficient sorting by rating. Top-rated hotels are shown by performing an inorder traversal.

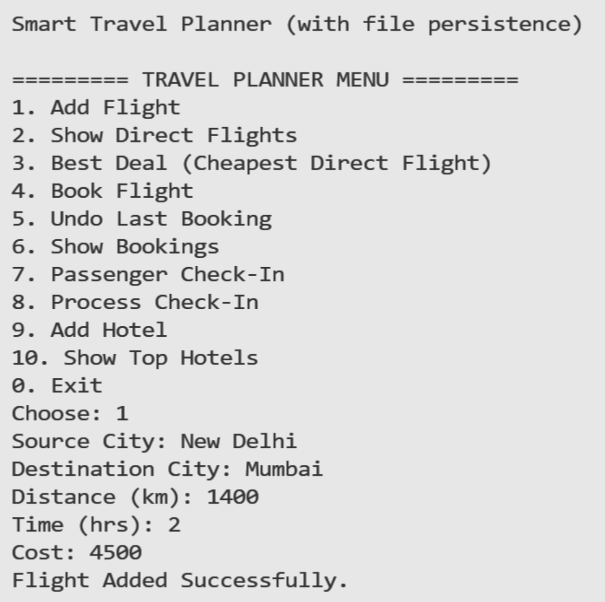
7. Exit:

   Before exit, all the updated data is permanently written back to files.

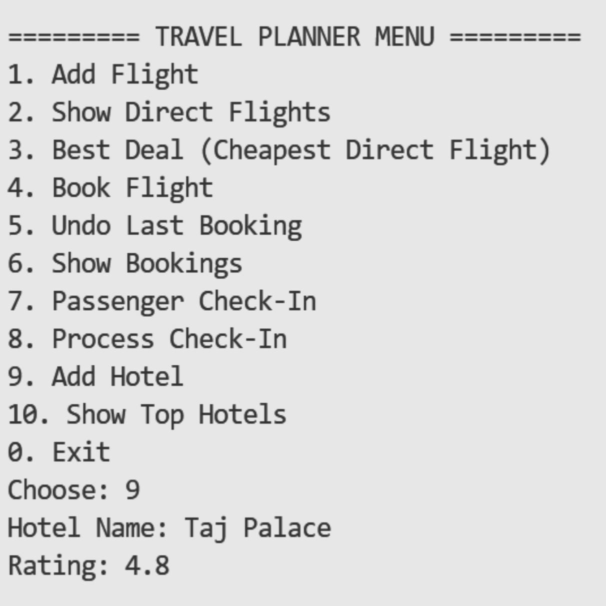
EXPECTED OUTCOMES

1. A simple travel planner that lets users search flights, book tickets, and manage basic trip details.
2. Flight search with clear results and an option to quickly find the cheapest direct flight.
3. A booking system where users can confirm tickets and undo the latest booking using a stack.
4. A queue-based check-in system that processes passengers in the same order as their arrivals.
5. Hotel recommendations arranged by their rating, managed through a BST for quick access.
6. All flights, bookings, and hotel details are saved permanently using file handling.

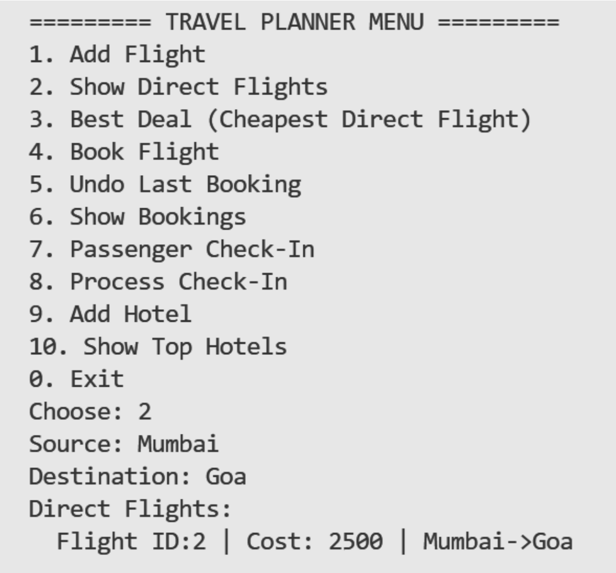
OUTPUTS OF THE COMMAND LINE INTERFACE



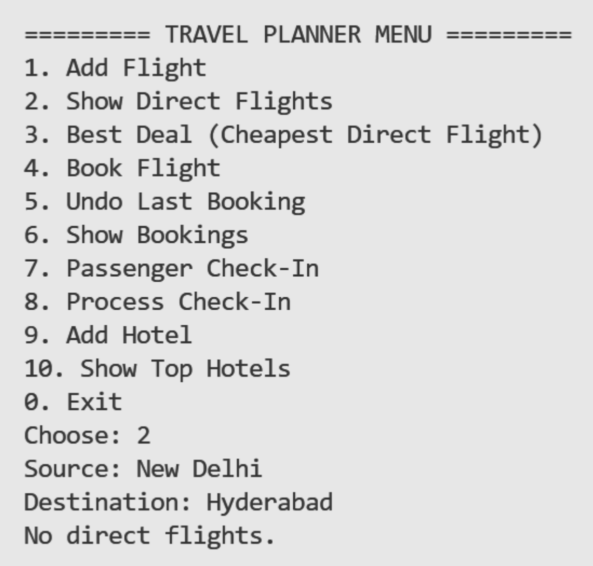
```
Smart Travel Planner (with file persistence)

========= TRAVEL PLANNER MENU =========
1. Add Flight
2. Show Direct Flights
3. Best Deal (Cheapest Direct Flight)
4. Book Flight
5. Undo Last Booking
6. Show Bookings
7. Passenger Check-In
8. Process Check-In
9. Add Hotel
10. Show Top Hotels
0. Exit
Choose: 1
Source City: New Delhi
Destination City: Mumbai
Distance (km): 1400
Time (hrs): 2
Cost: 4500
Flight Added Successfully.
```



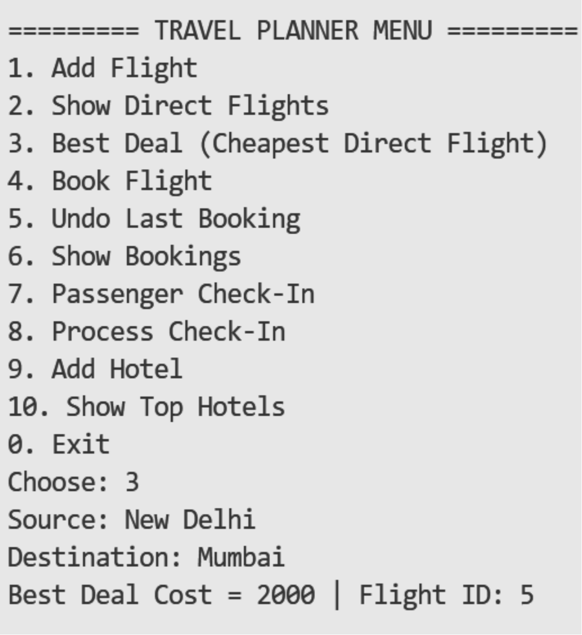
```
========= TRAVEL PLANNER MENU =========
1. Add Flight
2. Show Direct Flights
3. Best Deal (Cheapest Direct Flight)
4. Book Flight
5. Undo Last Booking
6. Show Bookings
7. Passenger Check-In
8. Process Check-In
9. Add Hotel
10. Show Top Hotels
0. Exit
Choose: 9
Hotel Name: Taj Palace
Rating: 4.8
```



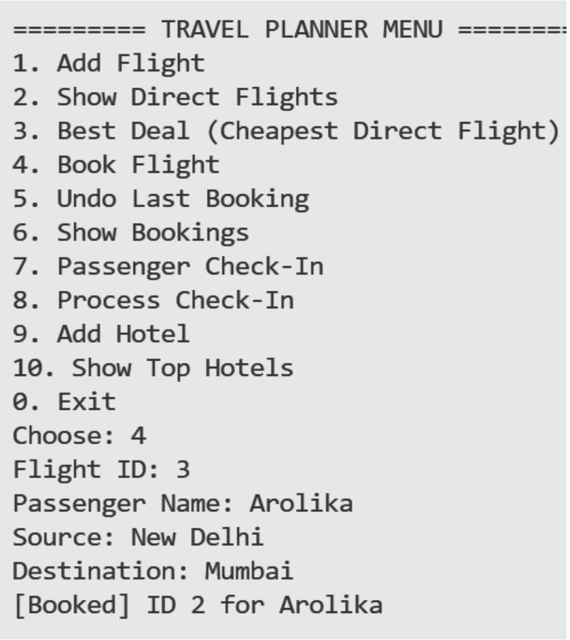
```
========= TRAVEL PLANNER MENU =========
1. Add Flight
2. Show Direct Flights
3. Best Deal (Cheapest Direct Flight)
4. Book Flight
5. Undo Last Booking
6. Show Bookings
7. Passenger Check-In
8. Process Check-In
9. Add Hotel
10. Show Top Hotels
0. Exit
Choose: 2
Source: Mumbai
Destination: Goa
Direct Flights:
  Flight ID:2 | Cost: 2500 | Mumbai->Goa
```



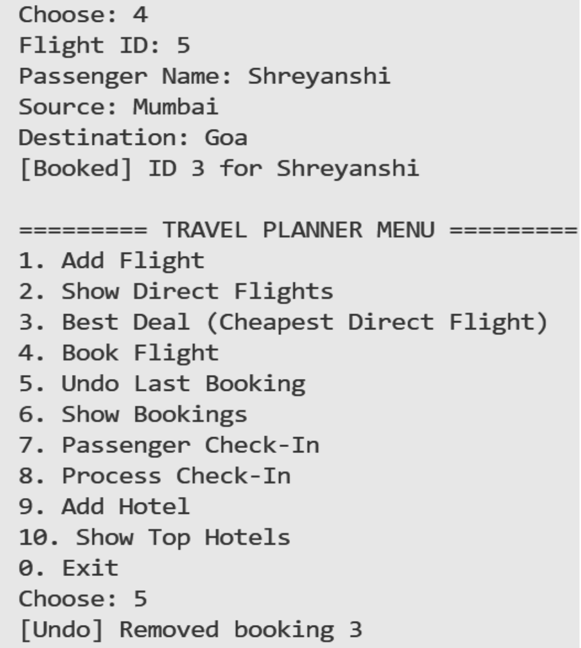
```
========= TRAVEL PLANNER MENU =========
1. Add Flight
2. Show Direct Flights
3. Best Deal (Cheapest Direct Flight)
4. Book Flight
5. Undo Last Booking
6. Show Bookings
7. Passenger Check-In
8. Process Check-In
9. Add Hotel
10. Show Top Hotels
0. Exit
Choose: 2
Source: New Delhi
Destination: Hyderabad
No direct flights.
```



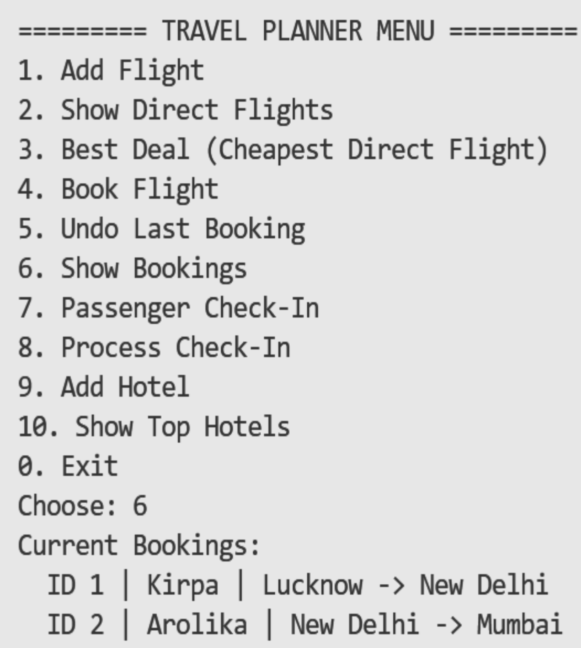
```
========= TRAVEL PLANNER MENU =========
1. Add Flight
2. Show Direct Flights
3. Best Deal (Cheapest Direct Flight)
4. Book Flight
5. Undo Last Booking
6. Show Bookings
7. Passenger Check-In
8. Process Check-In
9. Add Hotel
10. Show Top Hotels
0. Exit
Choose: 3
Source: New Delhi
Destination: Mumbai
Best Deal Cost = 2000 | Flight ID: 5
```



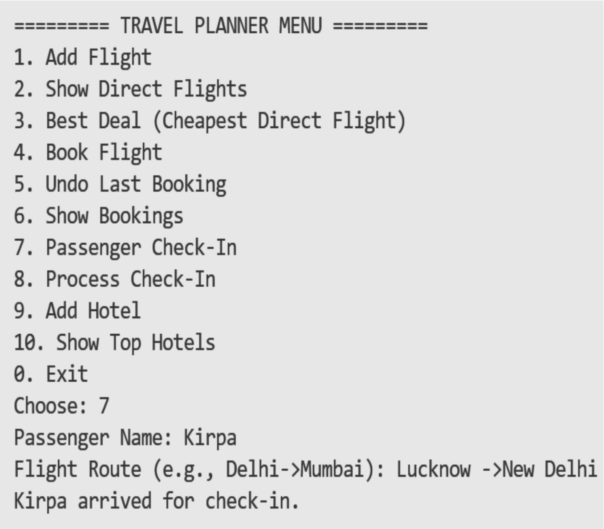
```
========= TRAVEL PLANNER MENU =======
1. Add Flight
2. Show Direct Flights
3. Best Deal (Cheapest Direct Flight)
4. Book Flight
5. Undo Last Booking
6. Show Bookings
7. Passenger Check-In
8. Process Check-In
9. Add Hotel
10. Show Top Hotels
0. Exit
Choose: 4
Flight ID: 3
Passenger Name: Arolika
Source: New Delhi
Destination: Mumbai
[Booked] ID 2 for Arolika
```



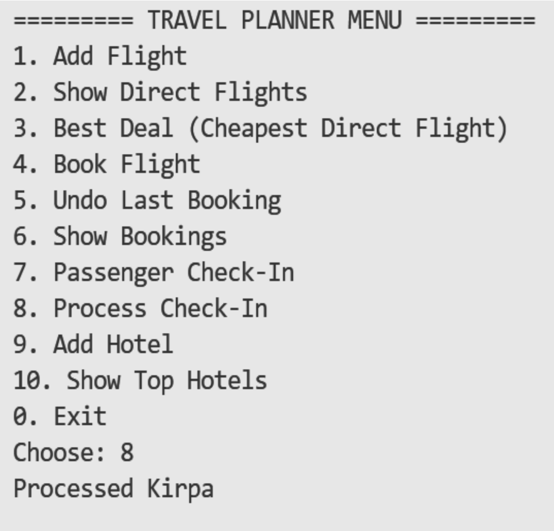
```
Choose: 4
Flight ID: 5
Passenger Name: Shreyanshi
Source: Mumbai
Destination: Goa
[Booked] ID 3 for Shreyanshi

========= TRAVEL PLANNER MENU =========
1. Add Flight
2. Show Direct Flights
3. Best Deal (Cheapest Direct Flight)
4. Book Flight
5. Undo Last Booking
6. Show Bookings
7. Passenger Check-In
8. Process Check-In
9. Add Hotel
10. Show Top Hotels
0. Exit
Choose: 5
[Undo] Removed booking 3
```



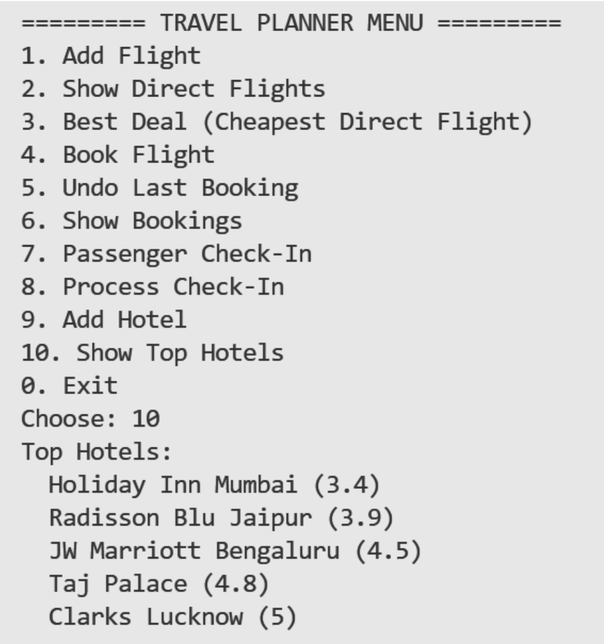
```
========= TRAVEL PLANNER MENU =========
1. Add Flight
2. Show Direct Flights
3. Best Deal (Cheapest Direct Flight)
4. Book Flight
5. Undo Last Booking
6. Show Bookings
7. Passenger Check-In
8. Process Check-In
9. Add Hotel
10. Show Top Hotels
0. Exit
Choose: 6
Current Bookings:
  ID 1 | Kirpa | Lucknow -> New Delhi
  ID 2 | Arolika | New Delhi -> Mumbai
```



```
========= TRAVEL PLANNER MENU =========
1. Add Flight
2. Show Direct Flights
3. Best Deal (Cheapest Direct Flight)
4. Book Flight
5. Undo Last Booking
6. Show Bookings
7. Passenger Check-In
8. Process Check-In
9. Add Hotel
10. Show Top Hotels
0. Exit
Choose: 7
Passenger Name: Kirpa
Flight Route (e.g., Delhi->Mumbai): Lucknow ->New Delhi
Kirpa arrived for check-in.
```



```
========= TRAVEL PLANNER MENU =========
1. Add Flight
2. Show Direct Flights
3. Best Deal (Cheapest Direct Flight)
4. Book Flight
5. Undo Last Booking
6. Show Bookings
7. Passenger Check-In
8. Process Check-In
9. Add Hotel
10. Show Top Hotels
0. Exit
Choose: 8
Processed Kirpa
```



```
========= TRAVEL PLANNER MENU =========
1. Add Flight
2. Show Direct Flights
3. Best Deal (Cheapest Direct Flight)
4. Book Flight
5. Undo Last Booking
6. Show Bookings
7. Passenger Check-In
8. Process Check-In
9. Add Hotel
10. Show Top Hotels
0. Exit
Choose: 10
Top Hotels:
  Holiday Inn Mumbai (3.4)
  Radisson Blu Jaipur (3.9)
  JW Marriott Bengaluru (4.5)
  Taj Palace (4.8)
  Clarks Lucknow (5)
```

CONCLUSION

The Smart Travel and Tourism Planner successfully shows how real-world travel problems can be solved using different Data Structures. By combining graphs for flight routes, stacks for booking undo, queues for check-in, BSTs for hotel sorting, and file handling for saving data, the project creates a complete and practical system. It helps users search flights, compare prices, book tickets, and view hotel options in an organized way. Overall, the project not only makes trip planning easier but also clearly demonstrates how multiple DSA concepts can be applied together to build a useful, interactive, and efficient application.

REFERENCES

1. Data Structures and Algorithms in C++ – Adam Drozdek

2. C++ Primer – Stanley B. Lippman, Josée LaJoie, Barbara E. Moo

3. GeeksforGeeks – Data Structures and Algorithms Tutorials

4. C++ Reference Documentation – cppreference.com