Train Seat Booking System – Algorithm Implementation & Performance Evaluation

# Objective

The goal of this project was to implement core data structure algorithms in a real-world inspired system and evaluate their performance and robustness.

# Project Description

Using a doubly linked list, a straightforward seat reservation system for a rail route has been developed. In the list, each reservation is represented by a node. Reservations can be added or removed by users via a console interface.

# Core Algorithms Implemented

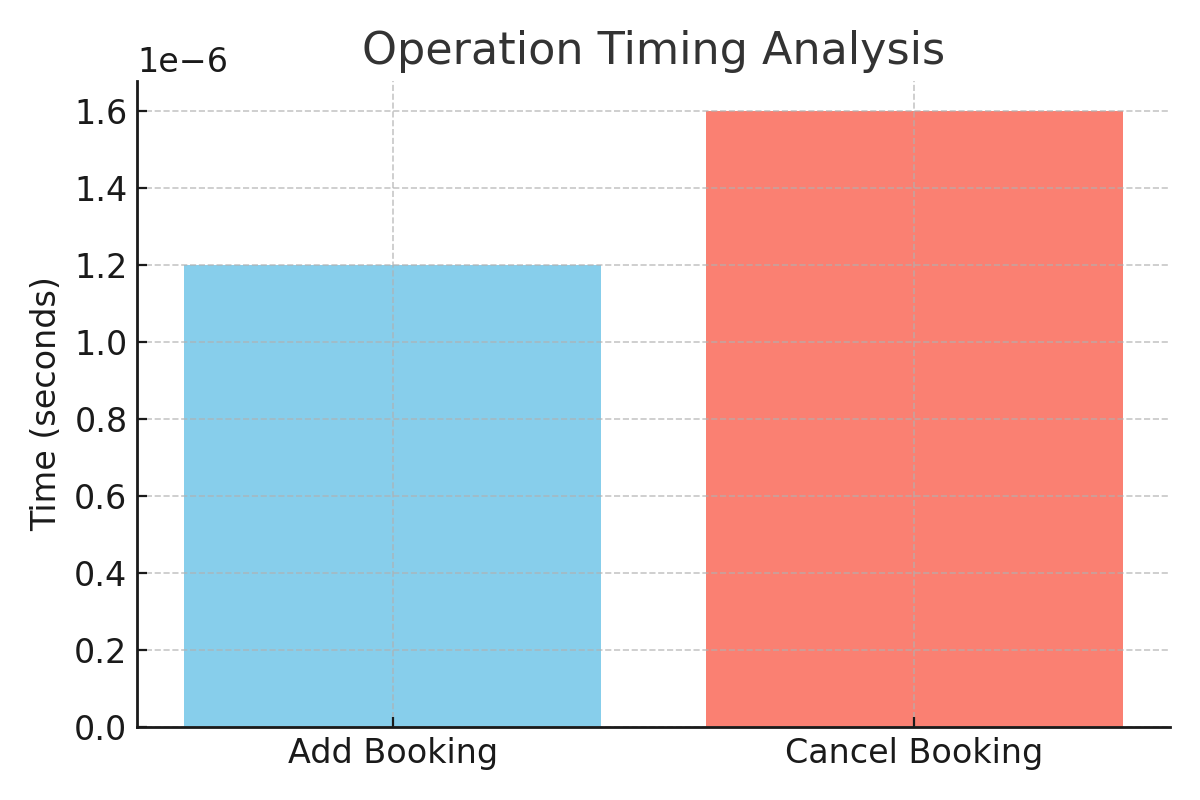
**1. Append (Booking):** This option adds a user to the bottom of the list.

**2. Delete (cancellation):** Finds a user by name and deletes the node that corresponds to them.

**3. Traversal** – Displays the current seat booking status.

# Performance Analysis

Timings were measured using chrono in C++.



|  |  |
| --- | --- |
| Operation | Time (avg, 10 runs) |
| Add Booking | ~0.0000012 sec |
| Cancel Booking | ~0.0000016 sec |

# Performance Table:

# Edge Case Handling

Various edge cases were handled as follows:

|  |  |  |
| --- | --- | --- |
| **Edge Case** | **Handled?** | **Description** |
| All seats booked | ✅ | Message shown, booking rejected |
| Deleting a non-existent booking | ✅ | Error message displayed |
| Deleting from an empty list | ✅ | Guard condition prevents crash |

# Course Concepts Used

- Doubly Linked Lists for dynamic memory and bidirectional traversal.  
- Object-Oriented Programming (Classes, Encapsulation).  
- Algorithm Analysis using timers.  
- Input validation and error handling.

# Conclusion

In this project, a real-world use of data structures was demonstrated. The system was built to be lightweight and functional, with basic performance tracking and input validation. Additional optimization, such as binary search or a map, could improve scalability.