**Operating Systems Project Report**

**Title:** Flight Reservation System

**Group Members:**

22k-4002 Ali Bhellar **(Group Leader)**

22k-4027 Toheed Ali

22k-4031 Arbaz Narejo

22K-8701 Muhammad Hateem Siddiqui

Submitted To: Miss Saeeda

Department of Artificial Intelligence

***Semester:*** Spring 2025

**Introduction:-**

This project simulates a **Flight Reservation System** using C++ with multithreading, mutual exclusion (mutex), and custom semaphores. It enables users to book flights, manage seat reservations, and log transactions efficiently.Each task demonstrates real-world synchronization issues and methods used to resolve them effectively, ensuring thread-safe and efficient operations.

**Key Features:-**

* Users can **book**, **cancel**, and **view** flight tickets.
* Implements **thread synchronization** using **mutexes** and **semaphores** to ensure data consistency during concurrent access.

**Modules: -**

**Customer Module:** Allows users to **book**, **cancel**, and **view** their flight reservations. Input validation and confirmation messages are provided for better user interaction.

**Admin Module:** Enables administrators to **view all bookings** and monitor flight seat availability. This helps in managing user reservations and system load.

**Data Management:** Handles **file I/O** and **shared memory** to store and retrieve booking information. Ensures data persistence and consistency across sessions.

**Synchronization Mechanism: -**

* Mutex (pthread\_mutex\_t) ensures that only one thread can modify a flight’s data (e.g., seat count) at a time.
* Semaphores are used to simulate a bounded system (e.g., only a fixed number of seats available).
* Prevents race conditions and maintains data consistency when multiple users (threads) try to book simultaneously.

**Project Structure:-**

**Core Components and Functionalities**

main.cpp : -

* Entry point of the program
* Initializes the system and presents user interface for booking and checking flights

flight.cpp / flight.h:

* Manages flight details like ID, destination, available seats
* Supports booking and cancellation functionalities

reservation\_system.cpp / reservation\_system.h: -

* Core reservation logic including:
* Adding flights
* Searching for available flights
* Booking seats
* Uses mutex and semaphores to manage concurrency

semaphore\_wrapper.cpp / semaphore\_wrapper.h: -

* Custom binary semaphore implementation using pthread
* Controls access to critical sections (e.g., concurrent seat booking)

logger.cpp / logger.h: -

* Logs reservation activities and system messages to a file or console

ui\_utils.cpp / ui\_utils.h: -

* Provides user interface utilities (e.g., printing menus and prompts)

**Output: -**

Users can run the program to:

* View flight list
* Reserve or cancel a seat
* See logs of their transactions

The program handles concurrent operations without corruption or crashes

**Tools and Technologies Used**

Language: C++

Compiler: GCC

OS: Linux (Ubuntu / VirtualBox )

Synchronization: POSIX Threads (pthreads), Mutexes, Semaphores

**Conclusion: -**

This project effectively demonstrates key aspects of operating system-level programming, focusing on multithreading and synchronization. Using POSIX threads, mutexes, and semaphores, it ensures safe concurrent access to shared resources in a flight reservation system. The implementation reflects real-world concurrency handling and resource management techniques used in production-level systems.