**OS Module 1**

Made all the classes required with a basic simulation.

**Airline**: Manages airline data like name, type (Commercial, Cargo, etc.), and flight counts. Tracks aircraft assignments and violations. Supports six airlines (PIA, AirBlue, etc.). Provides data for scheduling.

**Runway**: Models runways (RWY-A, RWY-B, RWY-C) for arrivals, departures, or cargo/emergency. Enforces single-aircraft use. Assigns based on flight direction.

**Flight**: Simulates a flight’s lifecycle, including direction, type, and priority. Manages phase transitions and emergency probabilities. Triggers AVNs for violations.

**OS Module 2**

1. **Scheduling Queues and Algorithms**: Flights data is input from the user. It includes arrival or departure times and specifically minutes and seconds. Then the arrival and departure queues are made. These queues are scheduled based on the First Come First Serve method. Then the flights with the same arrival and departure times are further scheduled based on priority.
2. **Threads:** Primarily a Radar thread and Flights threads are made. Flights threads are made based on their arrival and departure queues and timings. Radar thread continuously monitors the violation of speed.
3. **Thread Synchronization**: Threads are synchronized such that semaphores are used but to maintain the conflicting flights order, waiting queues are maintained for each runway as well as waiting time for each flight is also calculated. 3 Runways threads are made which continuously monitors if there are any waiting flights to be assigned runways. They then sleep till the runway is emptied.
4. **Runway Assignment**: Runway A is for Arrivals of Commercial Flights, Runway B for departures of Commercial Flights and Runway C is for Cargo, VIP and Medical flights.
5. **5 Minute Simulation**: Implementing all the above functionalities, simple couts were used to check the simulation. A time was printed every second along with the flights phase transitioning information and violations.

**OS Module 3**

We implemented 4 processes with one main process named ATCS process. Other processes were the child’s of this main process. However, chains of child processes were made due to the nature of communication flow.

ATCS -> AVN Generator -> Stripe Pay -> Airline Portal

Once made, these processes end when the program ends.

The medium of communication is Unnamed pipes. These processes are synchronized with Read/Write pipe operations. One process must read from the previous process and the communication is bidirectional. Total of 6 pipes are used. There functions are as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pipes / Processes | ATCS | AVN Generator | Stripe Pay | Airline Portal |
| fd1 | W | R |  |  |
| fd2 |  | W | R |  |
| fd3 |  |  | W | R |
| fd4 |  |  | R | W |
| fd5 |  | R | W |  |
| fd6 | R | W |  |  |

Empty cells represent closed read/write ends, R represent Read and W represent Write.

**SFML:**

For planes I have used plane sprites for RWY-A and RWY-B and used UFO sprite for RWY-C.

An ATC tower is placed at the right side of the screen. A radar is made out of sf::Circle as I was unable to find radar sprite.

We have stored the data of positions for the different phases which we use to animate plane sprites.