Airport Management System (Intern Task)

Objective:

Develop a GraphQL-based Airport Management System using NestJS with either TypeORM or Sequelize. The system should allow managing flights, passengers, and airport staff.

Requirements:

1. Flight Management

Create, update, and delete flights.

A flight should have:

- Flight number
- Departure airport
- Destination airport
- Departure time
- Arrival time
- Airline
- Available seats

Implement pagination for retrieving flights.

Allow filtering flights by departure time, destination, and airline.

2. Passenger Management

Register passengers with name, passport number, and nationality.

Book a flight for a passenger.

Implement seat allocation (ensure no duplicate seat assignments).

Allow passengers to check flight details.

3. Staff Management

Manage airport staff (pilot, crew, ground staff, security, etc.).

Assign staff to specific flights.

A staff member should have:

- Name
- Role (pilot, crew, security, etc.)
- Employee ID
- Assigned flight (if applicable)

4. Authentication & Authorization

Use JWT authentication for passengers and staff.

Define roles:

- Admin: Can manage flights, passengers, and staff.
- Staff: Can check and manage assigned flights.
- Passenger: Can book and check flights.

5. Airport Operations & Status Tracking

Implement real-time flight status updates (on time, delayed, canceled). Use GraphQL subscriptions for live flight status updates.

6. Data Optimization

Use DataLoader to optimize queries and prevent the N+1 problem.

Bonus Features (Optional for Extra Challenge):

Multi-Airport Support: Allow managing multiple airports.
Notifications: Send email/SMS alerts for flight delays or booking confirmations.
Baggage Tracking: Track baggage for each passenger.
Queue System: Use BullMQ to handle background tasks like sending notifications.

Deliverables:

A GitHub repo with a README explaining setup and usage.
A GraphQL schema with clear types, queries, and mutations.
Database models using TypeORM or Sequelize.
A Postman collection or GraphQL playground queries for testing.