**Project Title:** Flight Central Reservation System (Flight CRS)

**1. Project Overview:**  
The Flight Central Reservation System (Flight CRS) is a backend application developed using **NestJS**, designed to manage flight data, airport information, cabin class categories, and user authentication. It serves as a simplified model of how airlines manage reservations, ensuring proper validation, access control, and data persistence using JSON file-based storage.

**2. Software Development Life Cycle (SDLC) Model Used: Waterfall Model**

**Why Waterfall?**  
\* The project has clear and fixed requirements.  
\* Each phase is completed before moving to the next.  
\* Documentation is a priority.

**SDLC Phases:**

**1. Requirement Analysis:**  
\* Users must be authenticated to access flight, airport, and cabin class data.  
\* Admin should be able to perform CRUD operations on all entities.  
\* Data must be stored persistently (JSON file used).  
\* System should validate input formats (e.g., IATA codes, time format).

**2. System Design:**  
\* Modular folder structure separating controllers, services, DTOs, and data files.  
\* Data is stored in structured JSON files instead of a database.  
\* Input validation using class-validator.  
\* Auth Guard ensures only logged-in users access secure routes.

**3. Implementation:**  
\* Developed using NestJS framework.  
\* Used DTOs (Data Transfer Objects) for request validation.  
\* Implemented routes for CRUD operations.  
\* Created login-only authentication mechanism without JWT.

**4. Testing:**  
\* Manual testing using Postman.  
\* Test cases for login, add/update/delete flight, airport, cabin class.

**5. Deployment:**  
\* Project pushed to GitHub.  
\* Can be deployed on any server supporting Node.js.

**6. Maintenance:**  
\* Future enhancements can include database integration, JWT auth, passenger booking, etc.

**3. ER Diagram:**

Flight CRS(Auth)

🡪Flight Name

🡪Flight ID

🡪Origin

🡪Destination

🡪Arrival Time

🡪Departure Time

🡪Airport code

🡪Airport name

🡪Economy class

🡪First Class

🡪Business Class

🡪Prem Economy

🡪prem Business

**4. Project Module-wise Structure:**

**1. Auth Module**  
\* auth.controller.ts - Handles login route.  
\* auth.service.ts - Authenticates user.  
\* auth.guard.ts - Guards protected routes.  
\* login.dto.ts - Validates login credentials.

**2. Flight Module**  
\* flight.controller.ts - CRUD endpoints.  
\* flight.service.ts - JSON storage logic.  
\* create-flight.dto.ts & update-flight.dto.ts - Validation.

**3. Airport Module**  
\* airport.controller.ts - CRUD endpoints.  
\* airport.service.ts - JSON storage logic.  
\* create-airport.dto.ts & update-airport.dto.ts

**4. CabinClass Module**  
\* cabin-class.controller.ts  
\* cabin-class.service.ts  
\* create-cabin-class.dto.ts

**5. Tools & Technologies:**  
\* NestJS (Node.js framework)  
\* TypeScript  
\* class-validator (DTO validation)  
\* Postman (API Testing)  
\* Git & GitHub (Version Control)

**6. Login Session Explanation:**  
\* A basic login system is implemented using AuthService.  
\* Users must log in with a predefined username and password.  
\* Upon login, the session is marked as authenticated in memory.  
\* A custom AuthGuard protects all other modules—access to flights, airports, and cabin classes is denied if the user is not logged in.

**7. Endpoint Summary Table:**

| Module | Method | Endpoint | Description |
| --- | --- | --- | --- |
| Auth | POST | /auth/login | Login with credentials |
| Flight | GET | /flight | Get all flights |
| Flight | POST | /flight | Add a new flight |
| Flight | PUT | /flight/:id | Update flight by ID |
| Flight | DELETE | /flight/:id | Delete flight by ID |
| Airport | GET | /airport | Get all airports |
| Airport | POST | /airport | Add a new airport |
| Cabin Class | GET | /cabin-class | Get all cabin classes |
| Cabin Class | POST | /cabin-class | Add a new cabin class |

**9. Future Scope:**  
\* Integrate MongoDB/PostgreSQL for scalable data handling.  
\* Add booking and payment modules.  
\* Enhance authentication with JWT.  
\* Add frontend with Angular or React.