Airline Booking & Operations System

Flutter Mobile App + FastAPI Backend

Exam Overview

Duration: 96 hours (4 days)
Allowed tools: Any AI tools (ChatGPT, Copilot, Cursor, etc.)
Backend: FastAPI (Python)
Database: SQLite **or** MS SQL Server
Frontend: Flutter (Android required)
Deployment: Local **or** Cloud
Evaluation: Functionality, correctness, design quality, completeness

---

Scenario

You are hired to build a **mini airline platform** similar to a real-world airline system (e.g., Turkish Airlines).

The system must allow:

- Passengers to search flights, book seats, pay (mock), check in, and receive announcements
- Airline staff to manage flights, aircraft, seat maps, bookings, and operational updates

The solution must consist of:

1. A **FastAPI backend**
2. A **Flutter mobile application**

Technology Constraints

Backend (Mandatory)

- Python 3.10+
- FastAPI
- REST API
- JWT authentication
- SQLite **or** MS SQL Server
- Swagger/OpenAPI available at `/docs`

Mobile App (Mandatory)

- Flutter
- Consumes backend APIs only
- Handles loading, error, and empty states
- Android support required

Deployment

- Backend may run **locally** or be **deployed to the cloud**
- If deployed, provide public API URL
- SQLite is allowed even in cloud mode

User Roles

Passenger

- Searches flights

- Books seats
- Pays (mock)
- Checks in
- Views boarding pass
- Receives announcements

Staff

- Manages flights and airplanes
- Publishes announcements
- Manages bookings and seats

Functional Requirements

---

1. Authentication & Accounts

Requirements

- Passenger registration and login
- JWT-based authentication
- Role-based access control
- Staff users may be seeded or created via admin API

Passenger Profile

Each passenger must have:

- Full name
- Email
- Phone number
- Passport number
- Nationality
- Date of birth

Rule: Profile must be completed before booking a flight.

---

2. Airports & Flights (Passenger)

Airports

- List all available airports

Flight Search

Passengers must be able to search flights by:

- Origin airport
- Destination airport
- Departure date

Search results must display:

- Flight number
- Departure time
- Arrival time

- Duration
- Price
- Available seats
- Flight status

---

3. Flight Details & Seat Map

Flight Details

Flight details must show:

- Route
- Aircraft
- Schedule
- Status
- Gate and terminal (if available)

Seat Map

- Visual seat map per flight
- Show available and unavailable seats
- Seat categories allowed (standard / extra legroom)

---

4. Booking & Tickets

Booking Flow

1. Select flight
2. Add one or more passengers
3. Select seats (or auto-assign)
4. Create booking

Booking Rules

- Seats cannot be double-booked
- Booking is not confirmed until payment succeeds
- Cannot book cancelled or departed flights

Booking Statuses

- CREATED
- CONFIRMED
- CANCELLED

Each passenger must have a **ticket** with:

- Passenger name
- Seat number
- Ticket number

Each booking must have a unique **PNR code**.

---

5. Seat Hold & Concurrency

Seat Hold Rule

- Selected seats must be **held for 10 minutes**
- If payment is not completed within the hold window:
  - seats are released automatically
- System must prevent race conditions and double booking

---

6. Payment (Mock)

Payment Requirements

- Payment is simulated
- Supported mock methods:
  - CARD
  - APPLE_PAY
  - GOOGLE_PAY
- Payment records must be stored

Payment Statuses

- PENDING
- PAID
- FAILED

Rules

- Successful payment confirms the booking

- Payment endpoint must be idempotent

---

7. Manage My Trips (Passenger)

Passengers must be able to:

- View upcoming and past trips
- See booking details
- See passengers and seats
- See flight status updates
- Access announcements for their flight

---

8. Check-in & Boarding Pass

Check-in Rules

- Allowed from **24 hours to 1 hour** before departure
- Only confirmed bookings can be checked in
- Check-in is done per ticket (per passenger)

Boarding Pass

Boarding pass must include:

- Passenger name
- Flight number

- Seat
- Gate
- Boarding time
- QR code (string payload is sufficient)

---

9. Announcements (Passenger)

Passengers must receive flight-related announcements:

- Delay
- Cancellation
- Gate change
- Boarding started
- General information

Announcements must be visible in the mobile app.

---

10. Staff / Admin Features

10.1 Airplanes & Seat Templates

Staff can:

- Create airplane models
- Define seat map templates:

    - rows
    - seat labels
    - seat categories
- Preview generated seat maps

10.2 Flight Management

Staff can:

- Create flights
- Assign airplanes
- Update:
    - schedule
    - gate
    - terminal
    - flight status

Flight Statuses

- SCHEDULED
- BOARDING
- DELAYED
- CANCELLED
- DEPARTED
- LANDED

10.3 Announcements Management

Staff can:

- Create announcements linked to flights
- Choose announcement type
- Provide title and message

10.4 Booking Management

Staff can:

- View bookings by flight
- Search bookings by PNR
- Cancel bookings (before departure)
- Reassign seats (override)

All admin actions must respect seat availability rules.

---

11. Data Model (Minimum)

Your system must include at least:

- User
- PassengerProfile
- Airport

- Airplane
- Flight
- Booking
- Ticket
- Payment
- CheckIn
- Announcement

Relationships must be consistent and logically correct.

---

12. Non-Functional Requirements

Backend

- Clear project structure
- Business logic separated from API routes
- Input validation for all requests
- Consistent error response format
- Seed data for airports and flights
- README with setup instructions

Mobile App

- Clean navigation
- Graceful handling of API errors

- Usable UI (visual design not graded)

---

13. Submission Requirements

Submit a **single Git repository or ZIP** containing:

Backend

- Source code
- README.md with:
  - setup steps
  - database choice
  - how to run
- API base URL
- Admin credentials

Flutter App

- Source code
- README.md with:
  - how to run
  - backend configuration
- APK (optional but recommended)
