Task: Airline Management System

Project Description

In this task, you are expected to develop the backend of a web application using **Django DRF**, which will enable an airline company to manage its aircraft, flights, and passenger reservations. You can find the project details below.

Requirements

1. Models

Airplane

- Fields:
    - tail_number: string (e.g. TC-NRT)
    - model: string (e.g. Airbus A320)
    - capacity: integer
    - production_year: integer
    - status: boolean
- Relations:
    - An aircraft can be used for multiple flights.

Flight

- Fields:
    - flight_number: string
    - departure: string (e.g. Heathrow Airport)
    - destination: string (e.g. Istanbul Airport)
    - departure_time: datetime
    - arrival_time: datetime
    - airplane: foreign key
- Relations:
    - Each flight is assigned to an aircraft.
    - A flight can have multiple reservations.

Reservation

- Fields:
    - passenger_name: string
    - passenger_email: string
    - reservation_code: string (unique, automatically created)
    - flight: foreign key
    - status: boolean.
    - created_at: datetime.
- Relations:

- Each reservation is assigned to a flight.

2. API Endpoints

Airplane API

- **GET /airplanes/**: List all airplanes.
- **GET /airplanes/{id}/**: Get details of a specific airplane.
- **GET /airplanes/{id}/flights**: Get the flights of a specific airplane.
- **POST /airplanes/**: Add a new airplane.
- **PATCH /airplanes/{id}/**: Update a specific airplane.
- **DELETE /airplanes/{id}/**: Delete a specific airplane.

Flight API

- **GET /flights/**: List all flights.
- **GET /flights/{id}/**: Get details of a specific flight.
- **GET /flights/{id}/reservations**: Get reservations made for a specific flight.
- **POST /flights/**: Add a new flight.
- **PATCH /flights/{id}/**: Update a specific flight.
- **DELETE /flights/{id}/**: Delete a specific flight.

Reservation API

- **GET /reservations/**: List all reservations.
- **GET /reservations/{id}/**: Get details of a specific reservation.
- **POST /reservations/**: Add a new reservation.
- **PATCH /reservations/{id}/**: Update a specific reservation.

3. Features

- **Occupancy Check**: Check if the flight capacity is full when making a reservation.
- **Reservation Code Generation**: Automatically generate a unique reservation code during reservation creation (at least 5 characters, alphanumeric).
- **Filtering (Optional)**: Filter flights by departure location, arrival location, and departure/arrival date. This feature will require changes to the Postman Collection to be delivered at the end of the project.
- **Email Sending (Optional)**: Send an email to the passenger when a reservation is made.

Delivery Criteria

1. **Code Structure**: The development should follow the standard structure of Django projects.
2. **API Testability**: The API endpoints should be easily testable with Postman. The delivered project must include a Postman Collection.

3. **Documentation**: The project should include a README file that provides setup instructions, usage, and descriptions of the API endpoints.
4. **Basic Validation**: Basically validate the fields received through the API.