For Sweetu Dumbu

1. Admin: POST /api/v1.0/flight/airline/inventory/add

Test ID	Type	Test Case Description	Expected Result
TC_INV_1.1	Positive	Successfully add a new, valid flight schedule for an existing airline.	201 Created / 200 OK. Flight is saved in the database.
TC_INV_1.2	Positive	Successfully add a flight with only the minimum required fields.	201 Created / 200 OK. Flight is saved.
TC_INV_1.3	Positive	Successfully add a flight with all optional fields populated.	201 Created / 200 OK. Flight is saved.
TC_INV_1.4	Negative	Attempt to add a flight with missing required fields (e.g., no flight number, no price).	400 Bad Request with a clear error message.
TC_INV_1.5	Negative	Attempt to add a flight with invalid data types (e.g., price as text, invalid date format).	400 Bad Request.
TC_INV_1.6	Negative	Attempt to add a flight with logically invalid data (e.g., negative seat count).	400 Bad Request.
TC_INV_1.7	Negative	Attempt to add a flight where arrival time is before departure time.	400 Bad Request.

TC_INV_1.8	Negative	Attempt to add a flight with a departure date in the past.	400 Bad Request.
TC_INV_1.9	Negative	Attempt to add a duplicate flight (same airline, flight number, and departure time).	409 Conflict or 400 Bad Request.
TC_INV_1.10	Security	An unauthenticated (not logged-in) user attempts to access this endpoint.	401 Unauthorized.
TC_INV_1.11	Security	A logged-in **User** (not Admin) attempts to access this endpoint.	403 Forbidden.

2. User: POST /api/v1.0/flight/search

Test ID	Type	Test Case Description	Expected Result
TC_SEARCH_2.1	Positive	Search for a **one-way** flight with valid 'from', 'to', and 'date'.	200 OK with a JSON list of matching flights.
TC_SEARCH_2.2	Positive	Search for a **round-trip** flight with valid 'from', 'to', and 'dates'.	200 OK with lists for both outgoing and return flights.
TC_SEARCH_2.3	Positive	Verify search results contain: Flight Date/time, Airline Name/Logo, and Price.	All required fields are present and correctly formatted.
TC_SEARCH_2.4	Negative	Search with a missing required field (e.g., no 'from' location, no date).	400 Bad Request.
TC_SEARCH_2.5	Negative	Search with the same 'from' and 'to' location.	400 Bad Request.
TC_SEARCH_2.6	Negative	Search for a flight with a date in the past.	400 Bad Request.
TC_SEARCH_2.7	Negative	Search using an invalid date format (e.g., "DD/MM/YYYY" if "YYYY-MM-DD" is expected).	400 Bad Request.
TC_SEARCH_2.8	Edge Case	Search for a flight	200 OK with an

		route that has no scheduled flights.	empty list [].
TC_SEARCH_2.9	Edge Case	Search for a flight on the current day (today).	200 OK with a list of flights.

3. User: POST /api/v1.0/flight/booking/{flightid}

Test ID	**Type**	**Test Case Description**	**Expected Result**
TC_BOOK_3.1	Positive	A logged-in user successfully books a single seat.	201 Created / 200 OK. Response includes a valid PNR number.
TC_BOOK_3.2	Positive	A logged-in user successfully books multiple (e.g., 3) seats.	201 Created / 200 OK. Response includes a PNR.
TC_BOOK_3.3	Positive	Successful booking with all passenger details, "Veg" meal, and seat selection.	201 Created / 200 OK.
TC_BOOK_3.4	Negative	An unauthenticated (not logged-in) user attempts to book a ticket.	401 Unauthorized (as per business rule).
TC_BOOK_3.5	Negative	Attempt to book with an invalid or non-existent flightid.	404 Not Found.
TC_BOOK_3.6	Negative	Attempt to book with missing required body data (e.g., no passenger details, no email).	400 Bad Request.
TC_BOOK_3.7	Negative	Attempt to book with an invalid email format.	400 Bad Request.
TC_BOOK_3.8	Negative	Attempt to book 0 seats or a negative number of seats.	400 Bad Request.

TC_BOOK_3.9	Negative	Attempt to book more seats than are currently available on the flight.	400 Bad Request (e.g., "Not enough seats available").
TC_BOOK_3.10	Negative	Attempt to book a seat number that is already taken.	409 Conflict or 400 Bad Request.
TC_BOOK_3.11	Negative	Attempt to book a flight that is in the past.	400 Bad Request (e.g., "Flight has already departed").
TC_BOOK_3.12	Edge Case	Attempt to book the very last available seat on a flight.	201 Created / 200 OK. Booking succeeds.

4. User: GET /api/v1.0/flight/ticket/{pnr}

Test ID	**Type**	**Test Case Description**	**Expected Result**
TC_GET_4.1	Positive	Retrieve ticket details using a valid, existing PNR.	200 OK. Response contains all booking details.
TC_GET_4.2	Positive	Verify retrieved details match what was provided during booking.	All fields (names, seats, meal) are correct.
TC_GET_4.3	Negative	Attempt to retrieve a PNR that does not exist.	404 Not Found.
TC_GET_4.4	Negative	Attempt to retrieve a PNR with an invalid format (e.g., "123").	400 Bad Request.
TC_GET_4.5	Security	A logged-in user attempts to retrieve the PNR for a booking made by a *different* user.	403 Forbidden or 404 Not Found.

5. User: GET /api/v1.0/flight/booking/history/{emailId}

Test ID	**Type**	**Test Case Description**	**Expected Result**
TC_HIST_5.1	Positive	Retrieve history for a user who has multiple bookings.	200 OK with a JSON list of all tickets.
TC_HIST_5.2	Positive	Retrieve history for a user who has only one booking.	200 OK with a list containing one ticket.
TC_HIST_5.3	Positive	Retrieve history for a user who has zero bookings.	200 OK with an empty list [].
TC_HIST_5.4	Negative	Attempt to retrieve history using an invalid email format.	400 Bad Request.
TC_HIST_5.5	Security	An unauthenticated user attempts to get booking history.	401 Unauthorized.
TC_HIST_5.6	Security	A logged-in user (a@test.com) attempts to get the history for a *different* user (b@test.com).	403 Forbidden.

6. User: DELETE /api/v1.0/flight/booking/cancel/{pnr}

Test ID	**Type**	**Test Case Description**	**Expected Result**
TC_DEL_6.1	Positive	Successfully cancel a booking that is **more than 24 hours** from the journey date.	200 OK or 204 No Content.
TC_DEL_6.2	Negative	Attempt to cancel a PNR that does not exist.	404 Not Found.
TC_DEL_6.3	Negative	Attempt to cancel a PNR that has *already* been canceled.	400 Bad Request or 404 Not Found.
TC_DEL_6.4	Negative	Attempt to cancel a PNR for a flight that is already in the past.	400 Bad Request.
TC_DEL_6.5	Security	An unauthenticated user attempts to cancel a ticket.	401 Unauthorized.
TC_DEL_6.6	Security	A logged-in user attempts to cancel a PNR belonging to a *different* user.	403 Forbidden or 404 Not Found.
TC_DEL_6.7	Edge Case	**(Business Rule)** Attempt to cancel a booking **less than**	400 Bad Request (e.g., "Cancellation window closed").

		24 hours from the journey date.	
TC_DEL_6.8	Edge Case	Attempt to cancel a ticket *exactly* 24 hours and 1 minute before the flight.	200 OK. Cancellation succeeds.
TC_DEL_6.9	Edge Case	Attempt to cancel a ticket *exactly* 23 hours and 59 minutes before the flight.	400 Bad Request. Cancellation fails.

7. Advanced Integration & Data Integrity Tests

Test ID	**Type**	**Test Case Description**	**Expected Result**
TC_INT_7.1	Integration	**(Inventory Check)** 1. Search flight, note seats. 2. Book 2 seats. 3. Search flight again.	The available seat count in step 3 must be (count from step 1) - 2.
TC_INT_7.2	Integration	**(Cancellation & Inventory)** 1. Book 1 seat. 2. Note seat count. 3. Cancel booking. 4. Note seat count again.	The seat count in step 4 must be (count from step 2) + 1.
TC_INT_7.3	Integration	**(Concurrency Race Condition)** 1. Find a flight with 1 seat. 2. Simulate 2 users booking it simultaneously.	Only one user's booking is successful (201 Created). The other gets an error (400 or 409).
TC_INT_7.4	Integration	**(Business Rule - Timezone)** 1. Identify flight departure time (e.g., 8:00 AM in UTC+5). 2. Attempt to cancel at 24.5 hours before (in UTC).	This is a design/logic test. The cancellation rule (24h) must be checked against a consistent timezone (e.g., UTC).

8. Advanced Security & Validation Tests

Test ID	**Type**	**Test Case Description**	**Expected Result**
TC_SEC_8.1	Security	**(IDOR)** Logged in as User A, attempt to view/cancel a ticket PNR belonging to User B.	403 Forbidden or 404 Not Found. Details must not be exposed.
TC_SEC_8.2	Security	**(XSS)** Attempt to book with a passenger NAME of <script>alert(1)</script>.	400 Bad Request, OR data is sanitized in the database. No script executes when retrieving this data.
TC_SEC_8.3	Security	**(SQL Injection)** Attempt to get history with emailId of ' OR 1=1; --.	400 Bad Request or 404 Not Found. System must not crash or return all users' history.
TC_SEC_8.4	Security	**(Parameter Tampering)** Attempt to book a flight by manually adding a price=1 parameter to the JSON body.	The custom price parameter is ignored. The ticket is booked at the correct, official price.

9. Project Rubrics Verification Checklist

Check ID	**Requirement**	**Verification**
VC_9.1	Package Structure	Verify the main package is com.flightapp.*.
VC_9.2	Configuration	Verify all Spring Boot config is in application.properties or application.yaml.
VC_9.3	Code Structure	Verify code is logically separated into Controller, Service, Repository, Validation packages.
VC_9.4	Code Quality	Verified code has been analyzed with PMD/SonarQube (or similar tool) for quality.
VC_9.5	Readability	Verify code is clean, with clear, self-documenting names for variables and methods.