#	API / Test File	Test Name / Scenario	Input	Expected Result	Actual Status	Type of Test	Positive / Negative	Notes / Theory
1	auth.spec.js	POST /auth/login - valid credentials	Correct username/password	200 OK + token	200	Functional / Auth	Positive	Valid login generates token
2	auth.spec.js	POST /auth/login - wrong username	Wrong username	401 / 200 + reason	200	Functional / Auth	Negative	Invalid username rejected
3	auth.spec.js	POST /auth/login - wrong password	Wrong password	401 / 200 + reason	200	Functional / Auth	Negative	Invalid password rejected
4	auth.spec.js	POST /auth/login - empty username	Empty username	400 / 200 + reason	200	Functional / Auth	Negative	API handles empty username
5	auth.spec.js	POST /auth/login - empty password	Empty password	400 / 200 + reason	200	Functional / Auth	Negative	API handles empty password
6	auth.spec.js	POST /auth/login - missing username field	Missing username	400 / 200 + reason	200	Functional / Auth	Negative	Missing field handled

7	auth.spec.js	POST /auth/login - missing password field	Missing password	400 / 200 + reason	200	Functional / Auth	Negative	Missing field handled
8	bookings.spec.js	GET /booking - retrieve all bookings	None	200 OK + array	200	Functional / Booking	Positive	Returns all bookings
9	bookings.spec.js	GET /booking/{id} - valid	bookingid	200 OK + booking object	200	Functional / Booking	Positive	Specific booking retrieved
10	bookings.spec.js	GET /booking/{id} - non-existent	bookingid 9999	404 Not Found	404	Functional / Booking	Negative	Invalid booking ID handled
11	bookings.spec.js	POST /booking - without auth	Booking data	401 / 403 Unauthorized	200	Functional / Booking	Negative	No auth should fail
12	bookings.spec.js	POST /booking - empty body	{}	400 / 200 + reason	200	Functional / Booking	Negative	Empty request handled
13	bookings.spec.js	POST /booking - invalid data	Invalid booking fields	400 Validation Error	400	Functional / Booking	Negative	Server validates data
14	bookings.spec.js	POST /booking - extra fields	Booking + extra fields	200 OK + booking	200	Functional / Booking	Positive	Extra fields ignored or

								handled
15	bookings.spec.js	POST /booking - missing required fields	Partial booking	400 Validation Error	66	Functional / Booking	Negative	Validation ensures required fields
16	bookings.spec.js	POST /booking - incomplete booking	Partial booking	400 Validation Error	400	Functional / Booking	Negative	Missing fields trigger validation
17	bookings.spec.js	PUT /booking/{id} - update booking	Valid auth + data	200 OK	200	Functional / Booking	Positive	Booking updated successfully
18	bookings.spec.js	PUT /booking/{id} - update without auth	No token	403 / 401	403	Functional / Booking	Negative	Authentication required
19	bookings.spec.js	PUT /booking/{id} - invalid token	Wrong token	200 / 403 / 404	200 **FAILED**	Functional / Booking	Negative	API returned 200 instead of 403/404
20	bookings.spec.js	PATCH /booking/{id} - partial update	Valid auth + partial data	200 OK	200	Functional / Booking	Positive	Partial update works
21	bookings.spec.js	DELETE /booking/{id}	Valid auth	201 Created	201	Functional / Booking	Positive	Delete might not be imple

		- with auth						mented fully
22	bookings.spec.js	DELETE /booking/{id} - without auth	No token	403 Unauthorized	403	Functional / Booking	Negative	Auth required for delete
23	bookings.spec.js	DELETE /booking/{id} - non-existent	Invalid id	404 Not Found	404	Functional / Booking	Negative	API returns proper error
24	bookings.spec.js	Invalid dates test	checkin > checkout	200 / validation error	200	Functional / Booking	Negative	API allows invalid date (might need fix)
25	bookings.spec.js	Complete booking lifecycle	Create + retrieve + update + delete	Full CRUD success	200	Functional / Booking	Positive	End-to-end workflow
26	health.spec.js	GET /ping - health status	None	201 Created / JSON	201	Functional / Health	Positive	Endpoint accessible
27	health.spec.js	GET /ping - fast response	None	< 2 seconds	9ms	Performance / Health	Positive	Fast response
28	health.spec.js	GET /ping - correct headers	None	text/html or application/json	text/html	Functional / Health	Positive	Content-Type verified
29	health.spec.js	GET /ping - access	None	201	201	Functional / Health	Positive	Public endpoint

		ible without auth						
30	health.spec.js	GET /ping - consistent response	None	201 consistent	201	Functional / Health	Positive	Multiple calls consistent
31	bookings.spec.js	PATCH /booking/{id} - partial update	Partial booking	200 OK	200	Functional / Booking	Positive	Partial update works
32	bookings.spec.js	PUT /booking/{id} - update	Valid auth + data	200 OK	200	Functional / Booking	Positive	Booking updated successfully
33	bookings.spec.js	DELETE /booking/{id} - delete booking	Valid auth	201 / Created	201	Functional / Booking	Positive	Delete endpoint tested

API Testing Flow (Folder-Level)

```
   │
   ▼
[Playwright test triggers API requests for Auth, Booking,
Health]
    │
    ▼
[Express Router matches endpoint]
    - /auth → authController
    - /booking → bookingController
    - /ping → healthController
```

```
      |
      ▼
[Middleware / Validation]
    - Authentication (token check)
    - Input validation (required fields, types)
    - Date validation, business rules
      |
      ▼
[Controller Logic]
    - Auth: Generate JWT/UUID, validate credentials
    - Booking: Create, read, update, delete bookings
    - Health: Return service status
      |
      ▼
[Database / Storage]
    - MongoDB queries for bookings
    - Token validation / storage
      |
      ▼
[Test Receives Response]
    - Check status code
    - Validate response body fields
    - Verify headers (Content-Type)
    - Measure performance (response time)
      |
      ▼
[Assertions & QA Table]
    - Positive/Negative tests
    - Functional / Performance / Security
    - Document results
```

Developer Action Notes Based on API Test Results

1️⃣ Auth API

- ✅ **Valid Credentials Test Passed**

- Developer should ensure token generation is consistent (UUID/JWT format).
  - Tokens must have proper expiration and server-side validation.
- ⚠️ **Wrong Username/Password Tests Passed, but HTTP 200 Returned**
  - Issue: API returns **200 OK** for invalid login instead of **401 Unauthorized**.
  - Action: Change status code for failed login to **401** (or **403** if needed) to follow REST standards.
  - Reason: Frontend or clients may misinterpret 200 as success.
- ⚠️ **Empty / Missing Fields Tests Passed, but Status 200 Returned**
  - Action: Return **400 Bad Request** for missing required fields.
  - Reason: Makes error handling more predictable and prevents misuse.

2️⃣ Booking API

- ✅ **Retrieve All / Specific Bookings Passed**
  - Ensure pagination if dataset grows large to prevent performance issues.
  - Confirm API consistently returns JSON format for all bookings.
- ⚠️ **Invalid Token / Unauthorized Update/Delete Tests**
  - API returned **200** instead of **403/401**.
  - Action: Correct status codes for failed auth operations.
  - Optional: Include error message `"Authentication token required"` or `"Invalid token"`.
- ⚠️ **Invalid Booking Data / Missing Required Fields**
  - API correctly returned **400** for some invalid requests, but some returned **200 with reason**.
  - Action: Standardize **400 Bad Request** for all validation errors.
  - Add clear validation messages, e.g., `"First name is required"`, `"Total price must be positive"`.
- ⚠️ **DELETE Booking Endpoint**

    - API returned **201 Created** instead of proper delete response (**200/204**).
    - Action: Fix delete logic. Ensure response status and body clearly indicate deletion.
- ⚠️ **Invalid Dates Booking Test**
    - Checkin date > checkout date: API allowed booking (status 200).
    - Action: Implement server-side validation to reject logically invalid dates with **400**.

3️⃣ Health API

- ✅ **Health Endpoint Passed**
    - Response consistent across calls.
    - Fast response time (< 50ms).
- ⚠️ **Response Status 201 Instead of 200**
    - Standard for health check is **200 OK**.
    - Action: Update status code to 200 for `/ping` endpoint.
- ⚠️ **Content-Type Returned as text/html**
    - If intended for JSON clients, consider returning `application/json`.
    - Otherwise, ensure documentation reflects text/html return.

4️⃣ General Notes for Developer

1. **HTTP Status Codes:**
    a. Failed login → 401/403
    b. Validation errors → 400
    c. Successful creation → 201
    d. Successful deletion → 200/204
    e. Health endpoint → 200
2. **Consistency Across Endpoints:**

   a. All responses should have predictable structure (`{ data }`, `{ error }`, or `{ reason }`).
   b. Avoid mixing `200` with errors.

3. **Input Validation:**
   a. Required fields must be enforced server-side.
   b. Type validation (string, number, boolean, date) must reject invalid data.
4. **Token & Auth:**
   a. Ensure token verification works for all protected endpoints.
   b. Invalid or missing token → reject request with **403/401**.
5. **Performance / Response Time:**
   a. Ensure GET, POST, PUT, DELETE under normal load respond < 2s.
6. **Error Messages:**
   a. Include meaningful, consistent messages for failed requests.
   b. Avoid leaking sensitive internal details.