🔹 Booking API QA Testing Cheat Sheet

1️⃣ API Testing Basics

• /booking API: Create, read, update, delete hotel bookings; ensures data integrity.
• HTTP Methods: GET (retrieve bookings), POST (create), PUT (update full booking), PATCH (partial update), DELETE (remove).
• Status Codes: 200 OK (success), 201 Created, 400 Validation error, 403 Auth required, 404 Not found.
• Positive vs Negative Tests: Positive = valid booking succeeds; Negative = invalid/missing data fails safely.

2️⃣ Authentication & Security

• Auth Required: PUT, PATCH, DELETE require token; GET and POST usually do not.
• Authentication vs Authorization: Auth = verify user identity; AuthZ = permission to modify bookings.
• Token Handling: Validate format, expiration; unauthorized or invalid tokens → 403/404.
• Security Risks: Missing auth or invalid token tests prevent unauthorized modifications or deletions.

3️⃣ Input Validation

• Required Fields: firstname, lastname, totalprice, depositpaid, bookingdates (checkin/checkout).
• Optional Fields: additionalneeds; test presence and absence.
• Data Types: Validate string, number, boolean; check boundary cases (e.g., negative price, check-in after check-out).
• Invalid Data Handling: API should return 400 with descriptive validation errors.

4️⃣ CRUD Operations & Lifecycle

• Create Booking: POST valid data → expect 200 + bookingid.
• Read Booking: GET /booking → returns list; GET /booking/{id} → returns specific booking or 404 if missing.
• Update Booking: PUT = full update; PATCH = partial update. Auth required; expect 200 on success.
• Delete Booking: DELETE /booking/{id} with auth → 200/201; without auth → 403; non-existent → 404.
• Full Lifecycle Test: Create → Retrieve → Update → Partial Update → Delete → Confirm deletion.

5️⃣ Error Handling & Edge Cases

• Missing Fields: Expect 400 with detailed messages.

• Invalid Types or Formats: Check 400 responses for type mismatches or invalid dates.
• Non-existent booking IDs: Ensure API returns 404, no crashes.
• Duplicate IDs: Booking IDs auto-generated sequentially; uniqueness maintained.
• Boundary Dates: Test check-in after check-out, same-day bookings, far-future dates.

6️⃣ Testing Tools & Automation

• Playwright: Automation for API tests, assertions, and dynamic value handling.
• Key Functions: test() = define test case; expect() = assertions; request.post()/get()/put()/patch()/delete() = send requests.
• Dynamic Data: Capture bookingid from POST to reuse in subsequent tests.
• CI/CD: Run `npx playwright test` in pipelines (GitHub Actions/Jenkins) for automated validation.

7️⃣ Advanced QA Considerations

• Partial Updates: Verify only specified fields are updated; others unchanged.
• Invalid Token Handling: Test multiple invalid tokens → confirm 403/404, data unchanged.
• Test Reproducibility: Reset test DB, use consistent test data, mock external dependencies.
• Load & Performance: Measure response time, ensure server handles multiple simultaneous requests.
• Reporting: Capture status, request/response, validation errors, lifecycle steps; Playwright HTML/JSON reports recommended.

💡 **Tip:** For Booking API testing, always include **positive/negative flows, CRUD lifecycle, authentication checks, input validation, error handling, and performance/load checks** in your test plan.