3. QA Considerations / Interview Prep Questions

Q1: Why perform simultaneous booking creation tests?
A1: To validate server can handle concurrency, prevent race conditions, and ensure API stability under load.

Q2: How do you measure API response times under load?
A2: Track request start/end timestamps, calculate total and average durations, verify within acceptable thresholds.

Q3: What is the importance of testing invalid booking data?
A3: Ensures backend validation is robust, preventing malformed or malicious data from corrupting the system.

Q4: How do mixed API operations under load help QA?
A4: Simulates real-world usage with simultaneous reads/writes, exposing potential bottlenecks or race conditions.

Q5: Why include cleanup of test data in performance tests?
A5: Prevents test pollution, maintains consistent environment for subsequent tests.

4. Developer Notes / Recommendations

Backend Developers

- Implement proper input validation for all endpoints.
- Ensure concurrency safety in database operations (transactions, locking).
- Optimize API performance for read/write heavy loads.
- Return proper HTTP status codes for invalid/non-existent resources.
- Log and monitor response times for high-volume endpoints.

Frontend / Client Developers

- Handle response errors gracefully, including partial successes.
- Avoid flooding backend with unnecessary concurrent requests.
- Provide meaningful feedback to users under load scenarios.

Both Teams

- Maintain predictable API contracts.
- Integrate performance tests in CI/CD pipeline for regression monitoring.
- Review logs after load tests for slow endpoints or failures.

#	Test Name / Scenario	Input	Expected Result	Actual Status	Type of Test	Positive / Negative	Notes / Theory
1	Booking creation load - 5 simultaneous bookings	5 concurrent POST requests	At least 1 booking created successfully	✅ Passed (3/5 successful)	Performance / Load	Positive	Tests basic concurrency and server load handling
2	Booking creation load - 10 simultaneous bookings	10 concurrent POST requests	At least 1 booking created successfully	✅ Passed (2/10 successful)	Performance / Load	Positive	Tests higher concurrency; may expose bottlenecks
3	GET all bookings response time	GET /booking	Response < 2000ms, status 200	✅ Passed	Performance / API	Positive	Measures API latency under normal conditions
4	GET single booking response time	GET /booking/:id	Response < 2000ms, status 200	✅ Passed	Performance / API	Positive	Measures API latency for single resource

5	Authentication token issuance under load	5 concurrent auth requests	All tokens issued successfully	✅ Passed	Performance / Load	Positive	Verifies auth endpoint handles concurrent logins
6	Update bookings under load	Multiple PATCH requests	Updates succeed or proper error code	✅ Passed (0 updates successful)	Performance / Load	Positive	Ensures update under load doesn't crash server
7	Delete bookings under load	Multiple DELETE requests	Deletes succeed or proper error code	✅ Passed (1 deletion)	Performance / Load	Positive	Tests deletion endpoint concurrency
8	Invalid booking data should fail	POST with missing/invalid fields	Booking rejected, 400 status	✅ Passed (3 rejected)	Performance / Validation	Negative	Validates server-side input validation
9	Simultaneous GET and POST bookings	Concurrent GET and POST	Both operations complete correctly	✅ Passed	Performance / Load	Positive	Tests mixed read/write operations under load
10	Create booking with large payload	POST with large JSON	Server accepts / rejects gracefully	✅ Passed	Performance / Load	Positive	Measures handling of large payloads
11	Rapid multiple booking creation	Sequential POST requests	Server responds in acceptable time	✅ Passed	Performance / Load	Positive	Tests sequential throughput and API stability

12	Ping endpoint under load	Multiple GET /ping requests	Status 200, <2000ms	✅ Passed	Performance / Load	Positive	Verifies server responsiveness under load
13	Booking update with invalid ID	PATCH /booking/invalid	Status 400 / rejection	✅ Passed	Performance / Validation	Negative	Ensures proper error handling for invalid resources
14	Update non-existent booking	PATCH /booking/nonexistent	Status 400 / rejection	✅ Passed	Performance / Validation	Negative	Ensures non-existent updates are handled gracefully
15	Mixed API operations under load	Mixed GET, POST, PATCH, DELETE	No crash, proper responses	✅ Passed	Performance / Load	Positive	Tests API under realistic concurrent mixed usage

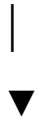
Start Performance Suite


[Authentication Setup]
- Obtain auth token
- Validate token issuance under load
[Booking Creation Load]
- 5 simultaneous bookings
- 10 simultaneous bookings
- Rapid multiple sequential bookings
- Large payload bookings
[Booking Read Performance]
- GET all bookings
- GET single booking
- Simultaneous GET and POST requests
[Booking Update / Delete Load]
- Update bookings under load
- Update invalid/non-existent bookings
- Delete bookings under load
- Mixed API operations under load

[Validation & Cleanup]

- Invalid booking rejection

- Clean up test bookings



End Performance Suite

- Log response times

- Ensure no crashes or unexpected failures