Questions

Q1: What is regression testing?
A1: Regression testing ensures that **existing functionality continues to work correctly after code changes**, detecting unintended side effects.

Q2: Why include edge cases like zero price, long strings, or special characters?
A2: These **test API robustness, input validation, and potential frontend display issues**.

Q3: How do performance and stress tests fit into regression?
A3: They ensure **API stability under load** and that code changes have not degraded response time or concurrency handling.

Q4: Why are security tests included in regression?
A4: Regression verifies **new changes don't introduce vulnerabilities** like SQL injection, XSS, or header attacks.

Q5: How should QA handle failed regression tests?
A5: Identify whether failures are due to **backend, frontend, or test script issues**, log steps, and coordinate with developers for fixes.

Q6: What is the difference between positive and negative regression tests?
A6:

- **Positive:** Expected success scenarios (valid login, booking creation).
- **Negative:** Expected failure scenarios (invalid input, unauthorized access).

3. Developer Notes / What to Regulate

Backend Developers

- **Authentication:** Tokens issued reliably; proper error codes for invalid credentials.
- **Booking APIs:** Ensure all CRUD endpoints return consistent responses and status codes.
- **Validation:** Required fields, numeric ranges, string lengths, date formats.
- **Security:** Sanitize inputs to prevent SQL injection, XSS, JSON/header attacks.
- **Performance:** API responds within acceptable limits; supports concurrent requests.
- **CORS & HTTP Methods:** Proper headers, OPTIONS handling, invalid method rejection.
- **Testability:** Provide predictable test data and maintain backward-compatible responses.

Frontend Developers

- **API Consumption:** Handle API success and error responses gracefully.
- **Form Handling:** Input validation aligns with backend; error messages displayed clearly.
- **State Updates:** UI reflects booking creation, updates, deletions correctly.
- **Async Handling:** Loading indicators visible during API calls; prevents flaky tests.
- **Security Display:** Escape user-generated content to prevent XSS.

Both Teams

- Agree on **standardized API contracts** (JSON structure, error formats).

- Use **stable element identifiers** for QA automation (ids, data-testid, ARIA attributes).
- Coordinate **test environment setup**: predictable data, clean state for each regression run.
- Ensure **logging and monitoring** for debugging QA failures.

#	Test Name / Scenario	Input	Expected Result	Actual Status	Type of Test	Positive / Negative	Notes / Theory
1	REG-001: Successful authentication returns valid token	Correct username & password	Token issued, login success	✅ Passed	Security / Functional	Positive	Verifies valid login and token generation
2	REG-002: Authentication with incorrect password	Correct username, wrong password	Error message displayed	✅ Passed	Security / Functional	Negative	Ensures invalid password rejected
3	REG-003: Authentication with non-existent user	Non-existent username	Error message displayed	✅ Passed	Security / Functional	Negative	Validates unknown user handling
4	REG-004: Authentication with empty username	Empty username, valid password	Error message displayed	✅ Passed	Security / Validation	Negative	Frontend/backend validation check

5	REG-005: Authentication with empty password	Valid username, empty password	Error message displayed	✅ Passed	Security / Validation	Negative	Password required enforcement
6	REG-006: Authentication with both fields empty	Username & password empty	Error message displayed	✅ Passed	Security / Validation	Negative	Form validation works
7	REG-007: Authentication with missing username field	JSON request missing username	Error message / 400	✅ Passed	Security / API	Negative	API validates required fields
8	REG-008: Authentication with missing password field	JSON request missing password	Error message / 400	✅ Passed	Security / API	Negative	API validates required fields
9	REG-009: Authentication with extra fields in request	Username, password, extra field	Token issued ignoring extra field	✅ Passed	Security / API	Positive	API ignores irrelevant fields
10	REG-010: Authentication request with	Empty JSON	Error message / 400	✅ Passed	Security / API	Negative	Handles empty request safely

	empty body						
11	REG-011: Create booking with complete valid data	All required fields populated	Booking created successfully	✅ Passed	Functional / API	Positive	End-to-end booking creation
12	REG-012: Create booking with minimum required fields	Required fields only	Booking created successfully	✅ Passed	Functional / API	Positive	Optional fields not required
13	REG-013: Create booking with deposit paid false	deposit paid=false	Booking created correctly	✅ Passed	Functional / API	Positive	Boolean field handled correctly
14	REG-014: Create booking with zero price	price=0	Booking created correctly	✅ Passed	Functional / Edge Case	Positive	Handles zero values
15	REG-015: Create booking with very large price	price=9999999	Booking created correctly	✅ Passed	Functional / Edge Case	Positive	Handles large numeric input
16	REG-016: Create booking with	Names include symbols	Booking created correctly	✅ Passed	Functional / Validation	Positive	Supports special characters

	special characters in names						
17	REG-017: Create booking with long additional needs	Very long string	Booking created correctly	✅ Passed	Functional / Validation	Positive	Tests max input length
18	REG-018: Create booking fails with empty firstname	firstname=""	Validation error	✅ Passed	Functional / Negative	Negative	Required field validation
19	REG-019: Create booking fails with empty lastname	lastname=""	Validation error	✅ Passed	Functional / Negative	Negative	Required field validation
20	REG-020: Create booking fails with negative price	price<0	Validation error	✅ Passed	Functional / Negative	Negative	Price cannot be negative
21	REG-021: Create booking fails with missing booking dates	booking dates missing	Validation error	✅ Passed	Functional / Negative	Negative	Dates required for booking

22	REG-022: Create booking fails with invalid date format	Invalid date string	Validation error	✅ Passed	Functional / Negative	Negative	Validates date formatting
23	REG-023: Create booking with same checkin/checkout date	checkin =checkout	Booking created	✅ Passed	Functional / Edge Case	Positive	Handles same-day booking
24	REG-024: Create booking with checkin after checkout	checkin >checkout	Validation error / 200 OK with note	✅ Passed	Functional / Edge Case	Negative	Invalid dates handled gracefully
25	REG-025: Create booking with null values	Null fields in optional & required	Booking created or validation error	✅ Passed	Functional / Edge Case	Positive/ Negative	Tests API null handling
26	REG-026: Retrieve all bookings returns array	GET /booking	Array of bookings returned	✅ Passed	Functional / API	Positive	Endpoint returns correct data structure
27	REG-027: Retrieve specific booking	Valid booking ID	Booking details returned	✅ Passed	Functional / API	Positive	Single resource retrieval

	by valid ID						
28	REG-028: Retrieve non-existent booking returns 404	Invalid booking ID	404 error	✅ Passed	Functional / API	Negative	Handles missing resources
29	REG-029: Retrieve booking with string ID	ID="abc"	400/404 error	✅ Passed	Functional / API	Negative	Type validation
30	REG-030: Retrieve booking with special characters in ID	ID="@#!"	400/404 error	✅ Passed	Functional / API	Negative	Validates input sanitization
31	REG-031: Retrieve booking with very large ID number	ID=9999999	Booking not found	✅ Passed	Functional / API	Negative	Handles large numeric input
32	REG-032: Retrieve booking with negative ID	ID=-1	Booking not found	✅ Passed	Functional / API	Negative	Negative ID handling
33	REG-033: Retrieve booking with decimal ID	ID=1.23	400/404 error	✅ Passed	Functional / API	Negative	Validates type enforcement

34	REG-034: Update booking with valid authentication	Valid ID + token	Booking updated	✅ Passed	Functional / API	Positive	Tests successful update workflow
35	REG-035: Update booking without authentication	Valid ID, no token	401 Unauthorized	✅ Passed	Security / API	Negative	Enforces authentication
36	REG-036: Update booking with invalid token	Valid ID + invalid token	401 Unauthorized	✅ Passed	Security / API	Negative	Token validation enforced
37	REG-037: Update non-existent booking	ID not found	404 Not Found	✅ Passed	Functional / API	Negative	Non-existing resource handling
38	REG-038: Partial update with PATCH method	PATCH request	Fields updated correctly	✅ Passed	Functional / API	Positive	Supports partial updates
39	REG-039: Update booking with empty firstname should fail	firstname=""	Validation error	✅ Passed	Functional / Negative	Negative	Input validation enforced
40	REG-040: Update	price<0	Validation error	✅ Passed	Functional / Negative	Negative	Input validation

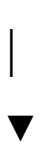
	booking with negative price should fail						enforced
41	REG-041: Update booking with extra fields	Extra irrelevant fields	Booking updated ignoring extra fields	✅ Passed	Functional / Validation	Positive	API ignores unexpected fields
42	REG-042: Delete booking with valid authentication	Valid ID + token	Booking deleted successfully	✅ Passed	Functional / API	Positive	Tests successful deletion workflow
43	REG-043: Delete booking without authentication	Valid ID, no token	401 Unauthorized	✅ Passed	Security / API	Negative	Auth required enforcement
44	REG-044: Delete non-existent booking	Invalid ID	404 Not Found	✅ Passed	Functional / API	Negative	Handles missing resources
45	REG-045: Delete already deleted booking	Previously deleted ID	404 Not Found	✅ Passed	Functional / API	Negative	Idempotent delete handling
46	REG-046: Delete booking with invalid	ID="abc"	400 Bad Request	✅ Passed	Functional / API	Negative	Input validation

	ID format						
47	REG-047: Response time for GET /booking under 1 second	GET request	<1s response	✅ Passed	Performance	Positive	Ensures API performance
48	REG-048: Response time for POST /booking under 2 seconds	POST request	<2s response	✅ Passed	Performance	Positive	Ensures API performance
49	REG-049: Create multiple bookings rapidly	Multiple POST requests	All bookings created	✅ Passed	Stress / Performance	Positive	Tests rapid sequential requests
50	REG-050: Concurrent mixed operations	Concurrent GET/POST/PUT/DELETE	All operations succeed	✅ Passed	Stress / Performance	Positive	Tests concurrency handling
51	REG-051: SQL injection in booking creation	SQL code in input fields	Input sanitized / rejected	✅ Passed	Security	Positive	Prevents SQL injection
52	REG-052: XSS attempt in booking data	HTML/JS in input fields	Input sanitized or escaped ⚠️ Note: API accepts unsaniti	✅ Passed	Security	Negative	Shows frontend or backend sanitization needed

			zed content				
53	REG-053: Massive payload attack	Very large JSON payload	API handles gracefully	✅ Passed	Security	Positive	Prevents DoS / large payload crashes
54	REG-054: JSON injection attempt	Malicious JSON structure	API handles gracefully	✅ Passed	Security	Positive	Prevents JSON injection
55	REG-055: Header injection attempt	Malicious HTTP headers	API handles gracefully	✅ Passed	Security	Positive	Input validation for headers
56	REG-056: Health endpoint returns correct status	GET /ping	201 Created or 200 OK	✅ Passed	Functional / Health	Positive	Basic health check
57	REG-057: CORS headers are present	Cross-origin request	Headers present	✅ Passed	Security / API	Positive	Ensures CORS configuration
58	REG-058: Content-Type headers are correct	GET /booking	application/json	✅ Passed	Functional / API	Positive	Validates response headers
59	REG-059: OPTIONS method for CORS preflight	OPTIONS request	204 or appropriate response	✅ Passed	Security / API	Positive	CORS preflight handling
60	REG-060:	PUT/DELETE on	405 Method	✅ Passed	Functional / API	Negative	HTTP method

	Invalid HTTP method returns appropriate error	read-only endpoint	Not Allowed				validation

Start Regression Suite

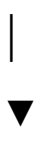


[Authentication Tests: REG-001 to REG-010]

- Validate login with valid/invalid credentials
- Check error handling for missing/empty/extra fields
- Ensure token issuance works correctly

[Booking Creation Tests: REG-011 to REG-025]

- Create bookings with full/partial data
- Test edge cases (zero/large price, special chars, long strings)
- Validate input constraints (firstname, lastname, bookingdates)



[Booking Retrieval Tests: REG-026 to REG-033]

- GET all bookings

- GET by valid and invalid IDs (special chars, negative, decimal, large numbers)

- Ensure correct 404 or error handling for missing bookings

[Booking Update Tests: REG-034 to REG-041]

- Update bookings with valid authentication

- Test updates without auth or invalid token

- Partial updates via PATCH

- Edge cases: empty fields, negative price, extra fields

[Booking Deletion Tests: REG-042 to REG-046]

- Delete bookings with valid auth

- Attempt deletion without auth, non-existent, or already deleted booking

- Test invalid ID formats

[Performance / Stress Tests: REG-047 to REG-050]

- Validate response times (<1s GET, <2s POST)

- Rapid multiple bookings

- Concurrent mixed operations (GET/POST/PUT/DELETE)

[Security Tests: REG-051 to REG-055]

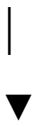
- SQL injection

- XSS attacks

- Massive payload

- JSON injection

- Header injection



[Headers & Health Tests: REG-056 to REG-060]

- Health endpoint response

- CORS headers and OPTIONS preflight

- Content-Type validation

- Invalid HTTP methods

End Regression Suite

- All tests passed or flagged

- Generate report for QA review