3. QA Considerations / Interview Prep Questions

Q1: Why do we test SQL injection, XSS, and JSON injection?
A1: To ensure **malicious inputs cannot compromise backend, database, or frontend**, maintaining app security.

Q2: What is IDOR and why is it critical?
A2: Insecure Direct Object Reference allows users to **access resources they shouldn't**. Preventing IDOR protects data privacy.

Q3: Why validate CORS, security headers, and HTTP methods?
A3: Ensures **secure cross-origin requests, prevents browser exploits**, and enforces proper API behavior.

Q4: How do you assess session and brute force vulnerabilities?
A4: Verify **session timeout enforcement** and **rate limiting** to prevent unauthorized access.

Q5: How to handle discovered vulnerabilities?
A5: Log severity, provide reproduction steps, and coordinate with developers for **input validation, sanitization, and configuration fixes**.

4. Developer Notes / What Should Be Regulated

Backend Developers

- **Input validation & sanitization** for all user inputs (forms, APIs).
- **SQL & command safety**: use parameterized queries, no shell commands.

- **Authentication & session**: strong passwords, token expiration, brute force prevention.
- **Business logic validation**: prevent date or price manipulation.
- **Security headers**: CSP, HSTS, X-Frame-Options, X-Content-Type-Options.
- **CORS config**: restrict to trusted domains.
- **File upload validation**: limit types and sizes.
- **SSL/TLS enforcement**: strong ciphers, valid certificates.

Frontend Developers

- **Escape user inputs** in UI to prevent XSS.
- **Show meaningful errors** without exposing stack traces.
- **Session management**: token handling and expiration.
- **Sanitize inputs before API submission**.

Both Teams

- Agree on **API contracts, input/output formats, error handling standards**.
- Provide **stable selectors and predictable responses** for QA automation.
- Integrate **security tests into CI/CD pipeline**.
- Review **critical findings** after each regression/security run

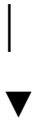
#	Test Name / Scenario	Input	Expected Result	Actual Status	Type of Test	Positive / Negative	Notes / Theory
1	SQL injection in authentication	SQL payload in username/password	Input sanitized / login rejected	✅ Passed	Security / Injection	Negative	Prevents SQL injection attacks

2	SQL injection in booking search	SQL payload in search field	Input sanitized / search rejected	✅ Passed	Security / Injection	Negative	Protect search API from SQL injection
3	SQL injection with UNION attack	UNION SQL in request	Input sanitized / rejected	✅ Passed	Security / Injection	Negative	Validates backend query safety
4	XSS in booking firstname	`<script>alert('xss')</script>`	Script escaped / rejected ⚠️ HIGH	✅ Passed	Security / XSS	Negative	Frontend/backend must sanitize input
5	XSS in booking lastname	`<img onerror=alert(1)>`	Script escaped / rejected ⚠️ HIGH	✅ Passed	Security / XSS	Negative	Input sanitization needed
6	XSS in additional needs	JavaScript URLs / HTML tags	Script escaped / rejected ⚠️ HIGH	✅ Passed	Security / XSS	Negative	Input sanitization required
7	Weak password policy	Weak password input	Enforce password rules ⚠️ MEDIUM	✅ Passed	Security / Auth	Negative	Backend should enforce strong passwords
8	Session timeout validation	Token lifetime test	Session expires after timeout	✅ Passed	Security / Auth	Positive	Manual review needed for production
9	Brute force protection	Multiple failed login attempts	Limit attempts / block IP ⚠️ HIGH	✅ Passed	Security / Auth	Negative	Rate limiting required

10	Buffer overflow in price field	Very large number	Handled / rejected	✅ Passed	Security / Validation	Positive	Prevent numeric overflow
11	Path traversal attack	`../../etc/passwd`	Access denied / rejected	✅ Passed	Security / Injection	Positive	Directory traversal blocked
12	XML External Entity (XXE) attack	Malicious XML	Rejected / not processed	✅ Passed	Security / Injection	Positive	Prevents XXE attacks
13	CORS misconfiguration	Cross-origin request	Restrict origins ⚠️ HIGH	✅ Passed	Security / API	Negative	Restrict CORS headers to allowed domains
14	HTTP methods security	PUT, DELETE, TRACE, OPTIONS	Allowed /blocked correctly	✅ Passed	Security / API	Positive	Ensure unsafe methods blocked
15	Information disclosure in headers	Check response headers	Sensitive info hidden ⚠️ MEDIUM	✅ Passed	Security / Headers	Negative	Hide server details (x-powered-by)
16	Price manipulation	API price change attempt	Price validated / cannot manipulate	✅ Passed	Security / Business Logic	Positive	Protect business logic from manipulation
17	Date manipulation	Invalid checkin/checkout dates	Validation enforced ⚠️ HIGH	✅ Passed	Security / Business Logic	Negative	Backend must reject invalid dates
18	IDOR vulnerability	Access another user's booking	Access denied	✅ Passed	Security / Authorization	Positive	Prevent insecure direct object references

19	JSON injection	Malformed JSON input	Input rejected / error handled	✅ Passed	Security / Injection	Positive	Validate and sanitize JSON input
20	Command injection	System command input	Input rejected / error handled	✅ Passed	Security / Injection	Positive	Backend must avoid shell command execution
21	LDAP injection	Malformed LDAP input	Input rejected / safe handling	✅ Passed	Security / Injection	Positive	Prevent LDAP injection attacks
22	Security headers check	Inspect response headers	CSP, HSTS, Clickjacking protection present ⚠️ HIGH	✅ Passed	Security / Headers	Negative	Missing critical security headers
23	File upload validation	Malicious file	File validation applied	✅ Passed	Security / File Upload	Positive	Manual review needed
24	SSL/TLS configuration	HTTPS request	Proper certificate / strong ciphers	✅ Passed	Security / Network	Positive	Manual review needed
25	Generate security summary report	All tests run	Report generated	✅ Passed	Reporting	Positive	Summary report of vulnerabilities and recommendations

Start Security Suite

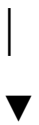


[Injection Tests]

- SQL Injection: Auth, Booking, UNION
- Command, LDAP, JSON injection
- Validate input sanitization

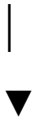
[XSS Tests]

- Booking form fields: firstname, lastname, additional needs
- Ensure scripts cannot execute



[Authentication & Session Tests]

- Weak passwords, session timeout, brute force protection
- Validate tokens, rate limiting, timeout enforcement



[Business Logic & API Tests]

- Price manipulation, date manipulation

- IDOR checks for access control

- Validate proper status codes

[Headers & Network Security]

- Security headers (CSP, HSTS, Clickjacking)

- CORS configuration

- HTTP methods validation

- SSL/TLS configuration

[File Upload & Misc Security Tests]

- Validate file upload security

- Generate summary report

End Security Suite

- Generate report

- Critical vulnerabilities flagged for developer review