

# API INTERVIEW QUESTIONS (100)

---

**Beginner + Advanced + Real-Time Scenarios**

## SECTION 1: API BASICS (Q1–Q30)

### 1. What is an API?

**Definition:** An API (Application Programming Interface) is a set of rules and protocols that allows different software applications to communicate with each other.

**Real-time Scenario:**

When you use a weather app on your phone, it doesn't have weather data built-in. Instead, it calls a weather service API (like OpenWeatherMap), sends your location, and receives current weather data to display.

### 2. What is REST API?

**Definition:** REST (Representational State Transfer) is an architectural style for designing networked applications that use HTTP methods to perform operations on resources.

**Real-time Scenario:**

A food delivery app uses REST APIs:

- `GET /restaurants` - Fetch available restaurants
- `POST /orders` - Place a new order
- `GET /orders/{id}` - Check order status
- `DELETE /orders/{id}` - Cancel an order

### 3. What are HTTP methods?

**Definition:** Standard operations defined by HTTP protocol for web communication.

**Real-time Usage:**

```
GET /api/products      # Get all products (E-commerce site)
```

```
POST /api/cart          # Add item to cart
PUT /api/users/123      # Update user profile
DELETE /api/comments/5  # Remove comment
PATCH /api/orders/10   # Update order status partially
```

## 4. Difference between GET and POST?

### GET:

- Used to retrieve data
- Parameters visible in URL
- Bookmarkable
- Limited data capacity
- Example: Searching products on Amazon: `amazon.com/search?q=laptop`

### POST:

- Used to submit data
- Data in request body (hidden)
- Not bookmarkable
- Larger data capacity
- Example: Submitting login form with username/password

## 5. What is an endpoint?

**Definition:** A specific URL where an API can be accessed to perform operations.

### Real-time Example:

```
Banking Application:
GET    /api/accounts          → List all accounts
GET    /api/accounts/{id}     → Get specific account
POST   /api/transfers         → Initiate money transfer
GET    /api/transactions      → View transaction history
```

## 6. What is request and response?

**Request:** What client sends to server

```
POST /api/login HTTP/1.1
Content-Type: application/json
{"username": "john", "password": "pass123"}
```

**Response:** What server returns to client

```
HTTP/1.1 200 OK
Content-Type: application/json
{"token": "abc123", "user": {"id": 1, "name": "John"}}
```

## 7. What is JSON?

**Definition:** JSON (JavaScript Object Notation) is a lightweight data interchange format that's easy for humans to read and write, and easy for machines to parse.

### Real-time Scenario:

When you fill out a registration form, your data is converted to JSON:

```
{
  "firstName": "Sarah",
  "lastName": "Chen",
  "email": "sarah@email.com",
  "phone": "+1234567890",
  "address": {
    "street": "123 Main St",
    "city": "San Francisco",
    "zipcode": "94107"
  }
}
```

## 8. What is HTTP status code?

**Definition:** A 3-digit code that indicates the result of an HTTP request.

### Real-time Examples:

```
200 OK - Your request succeeded (Google search results)
404 Not Found - Page doesn't exist (broken link)
500 Internal Server Error - Website is down
403 Forbidden - You're not allowed to access (admin panel without login)
```

## 9. Common HTTP status codes?

### Success:

- 200 OK - Request successful
- 201 Created - Resource created successfully (new user registered)
- 204 No Content - Success but no data to return

### Client Errors:

- 400 Bad Request - Invalid input (wrong email format)

- 401 Unauthorized - Not authenticated (missing token)
- 403 Forbidden - Authenticated but not authorized (user trying to access admin area)
- 404 Not Found - Resource doesn't exist
- 429 Too Many Requests - Rate limit exceeded

### Server Errors:

- 500 Internal Server Error - Server problem
- 502 Bad Gateway - Upstream server issue
- 503 Service Unavailable - Server overloaded/maintenance

## 10. What is RESTful API?

**Definition:** An API that follows REST architectural principles including statelessness, cacheability, and proper use of HTTP methods.

### Real-time Scenario - E-commerce API:

GET	/products	→ Returns list of products
POST	/products	→ Create new product (admin)
GET	/products/{id}	→ Get specific product details
PUT	/products/{id}	→ Update entire product
PATCH	/products/{id}	→ Update product price only
DELETE	/products/{id}	→ Remove product

## 11. What is statelessness?

**Definition:** Each request from client to server must contain all the information needed to understand and process the request. Server doesn't store any client context between requests.

### Real-time Example:

When you browse products on Amazon:

- Request 1: GET /products
- Request 2: GET /cart (token required)
- Request 3: GET /products/123

## 12. What is CRUD?

**Definition:** CRUD stands for Create, Read, Update, Delete.

### Real-time Blog System:

Create: POST /articles  
Read: GET /articles/{id}  
Update: PUT /articles/{id}  
Delete: DELETE /articles/{id}

### 13. What is API testing?

**Definition:** Verifying API functionality, performance, security, and reliability.

**Real-time Scenario:**

Testing payment APIs for success, failure, load, and edge cases.

### 14. Tools for API testing?

- Postman
- Swagger/OpenAPI
- SoapUI

### 15. What is payload?

**Definition:** Data sent in request or response body.

**Example:**

```
{  
  "content": "Just launched my new app!",  
  "privacy": "public"  
}
```

Perfect 👍

I'll clearly explain each question (16–30) with:

- Simple definition
- Real-time example
- Interview-friendly explanation
- Practical usage

No shortcuts, no confusion — this is **exactly what interviewers expect**.

### 16. What are headers?

## Definition

Headers are **key–value pairs** sent with an HTTP request or response that contain **metadata** (extra information) about the request or response.

They do **not contain actual data**, but they tell the server **how to process the data**.

## Real-time Example

When a user opens a dashboard after login:

```
GET /api/user/profile
Authorization: Bearer eyJhbGciOiJIUzI1Ni...
Content-Type: application/json
Accept: application/json
```

- **Authorization** → Who the user is
- **Content-Type** → Format of request data
- **Accept** → Expected response format

## Interview Tip

👉 Headers are mainly used for **authentication, content type, caching, and client information**.

# 17. What is Content-Type?

## Definition

Content-Type header specifies **the format of the data being sent** in the request or response body.

## Real-time Examples

### *1. Sending JSON data*

```
Content-Type: application/json
{
  "email": "user@gmail.com",
```

```
"password": "123456"
}
```

## 2. File upload

Content-Type: multipart/form-data

## Why it is important

If the server receives JSON but Content-Type is missing or wrong, the request may **fail or throw errors**.

## Interview Tip

👉 Always set Content-Type correctly, especially for **POST, PUT, PATCH** requests.

# 18. What is Accept header?

## Definition

The Accept header tells the server **which response format the client expects**.

## Real-time Example

### Mobile App Request

```
GET /api/products
Accept: application/json
```

Server responds:

```
{
  "id": 101,
  "name": "Laptop",
  "price": 50000
}
```

### Browser Request

```
Accept: text/html
```

Server responds with an **HTML page**.

## Interview Tip

👉 Accept = response format

👉 Content-Type = request format

## 19. What is query parameter?

### Definition

Query parameters are **key–value pairs appended to the URL** used for **filtering, sorting, searching, or pagination**.

They come **after ?** in the URL.

### Real-time Example (E-commerce Website)

```
GET /api/products?category=mobile&price=low&page=2
```

- category=mobile → Filter
- price=low → Sort
- page=2 → Pagination

## Interview Tip

👉 Query parameters are **optional** and mainly used for **filters and search conditions**.

## 20. What is path parameter?

### Definition

Path parameters are **variables embedded in the URL path** used to **identify a specific resource**.



## Real-time Examples

GET /api/users/25

→ Fetch user with ID **25**

DELETE /api/orders/789

→ Delete order **789**

## Interview Tip

👉 Path parameters are **mandatory** and represent **unique resource identifiers**.

## 21. Difference between query and path parameter?

Feature	Path Parameter	Query Parameter
Purpose	Identify resource	Filter or modify response
Mandatory	Yes	No
Position	Part of URL path	After ?
Example	/users/10	/users?role=admin

## Real-time Example

GET /api/users/10?active=true

- 10 → Path param (specific user)
- active=true → Query param (filter condition)

## 22. What is API documentation?

### Definition

API documentation explains **how to use an API**, including endpoints, request formats, response formats, and error messages.

## Real-time Examples

- **Stripe API docs** → Payment processing
- **Google Maps API docs** → Location services
- **Twitter API docs** → Tweets and timelines

## Interview Tip

👉 Good documentation reduces **developer confusion** and speeds up **integration**.

## 23. What is Swagger / OpenAPI?

### Definition

Swagger (OpenAPI) is a **standard specification** used to **document REST APIs** in a readable and interactive format.

### Real-time Scenario

Frontend developer opens Swagger UI and:

- Sees all endpoints
- Tests APIs without Postman
- Understands request/response formats

### Benefits

- ✓ Interactive testing
- ✓ Auto-generated docs
- ✓ Client SDK generation

## 24. What is versioning in API?

### Definition

API versioning is the practice of **maintaining multiple versions** of an API to support changes without breaking existing clients.

### Real-time Example

```
/api/v1/users    → Old mobile apps  
/api/v2/users    → New mobile apps
```

### Interview Tip

👉 Versioning helps **smooth upgrades** and **backward compatibility**.

## 25. Why versioning is important?

### Definition

Versioning ensures that **existing applications continue to work** even when APIs are updated.

### Real-time Problem

A company changes response format:

Old:

```
{"name": "John"}
```

New:

```
{"fullName": "John Doe"}
```

Without versioning → Old apps break

With versioning → Safe upgrade

## 26. What is REST constraint?

### Definition

REST constraints are **rules** that define how a REST API should behave.

### Key Constraints

1. Client–Server
2. Stateless
3. Cacheable
4. Uniform Interface
5. Layered System

### Real-time Example

Frontend React app → Backend Node API → Database  
Each layer works **independently**.

## 27. What is cache?

### Definition

Cache is **temporary storage** of frequently accessed data to improve **performance and reduce server load**.

### Real-time Example

Weather app:

- First request → API call (slow)
- Next requests → Cached data (fast)

### Interview Tip

👉 Cache improves **speed, scalability, and user experience**.

## 28. What is CORS?

### Definition

CORS (Cross-Origin Resource Sharing) allows a server to specify **which domains are allowed** to access its resources.

### Real-time Scenario

Frontend:

`https://myapp.com`

Backend:

`https://api.myapp.com`

Server sends:

`Access-Control-Allow-Origin: https://myapp.com`

### Interview Tip

👉 CORS is a **browser-side security feature**, not backend logic.

## 29. What is same-origin policy?

### Definition

A browser security rule that prevents a webpage from accessing data from another origin.

**Origin =**

- Protocol
- Domain
- Port

### Real-time Example

❌ Blocked:

`https://siteA.com → https://siteB.com`

✓ Allowed:

`https://siteA.com → https://siteA.com/api`

## 30. What is API authentication?

### Definition

API authentication verifies **who is making the request** before allowing access.

### Common Methods & Real-time Examples

#### 1. API Keys

`GET /api/data?api_key=abc123`

Used in public APIs (maps, weather)

#### 2. JWT Tokens

`Authorization: Bearer eyJhbGciOiJIUzI1Ni...`

Used in login-based systems

#### 3. OAuth

Login using Google / Facebook

#### 4 Basic Authentication

`Authorization: Basic base64(username:password)`

Used in internal or legacy systems

### Interview Tip

👉 Authentication = identity

👉 Authorization = permissions

## SECTION 2: INTERMEDIATE API CONCEPTS (Q31–Q60)

### 31. What is Authentication vs Authorization?

### **Authentication – “Who are you?”**

Verifies the identity of a user or system.

### **Authorization – “What are you allowed to do?”**

Determines permissions after authentication.

### **Real-time Scenario (Company HR System):**

1. Employee logs in → Authentication succeeds
2. Employee tries to access payroll data:
  - HR Manager → Allowed
  - Intern → Forbidden (403)

## **32. What is JWT (JSON Web Token)?**

### **Definition:**

JWT is a compact, URL-safe token used to securely transmit information between client and server.

### **Why JWT is used:**

- Stateless authentication
- No server-side session storage
- Scales easily in microservices

### **Real-time Scenario:**

User logs into an e-commerce site → receives JWT → uses it to access orders, cart, and profile.

## **33. What is the structure of JWT?**

JWT consists of **three parts** separated by dots:

`Header.Payload.Signature`

### **Details:**

- **Header:** Token type & algorithm
- **Payload:** User data (claims)
- **Signature:** Verifies token integrity

### **Example Payload:**

```
{  
  "userId": 123,  
  "role": "admin",  
  "exp": 1700000000  
}
```

### 34. What is a Bearer token?

**Definition:**

A Bearer token is an access token sent in HTTP headers to authenticate API requests.

**Real-time Usage:**

```
GET /api/orders  
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Why “Bearer”?**

Whoever holds the token can access the resource — so it must be protected.

### 35. How does JWT work in real time?

**Step-by-step Login Flow:**

1. User sends credentials (POST /login)
2. Server validates credentials
3. Server generates JWT
4. JWT sent to frontend
5. Frontend stores token (localStorage/cookies)
6. Token sent with every API request

**Why this matters in interviews:**

Shows understanding of **stateless auth flow**.

### 36. What is OAuth?

**Definition:**

OAuth is an authorization framework that allows third-party apps to access user data without exposing passwords.

**Real-time Scenario:**

“Login with Google”:



- App never sees your Google password
- Google issues an access token
- App accesses profile/email with permission

### **37. What is a Refresh Token?**

**Definition:**

A refresh token is used to obtain a new access token without requiring the user to log in again.

**Real-time Scenario:**

- Access token expires in 15 minutes
- Refresh token valid for 7 days
- Seamless user experience without frequent logins

### **38. What is API Rate Limiting?**

**Definition:**

Restricts the number of API requests a client can make within a specific time window.

**Real-time Example:**

100 requests per minute per user

**Why needed:**

- Prevent abuse
- Protect server resources
- Ensure fair usage

### **39. Why is Rate Limiting important?**

**Real-time Problem:**

A public API without limits gets:

- Bot attacks
- DDoS attempts
- Server crashes

**Solution:**

Return:

429 Too Many Requests

## 40. What is Idempotency?

**Definition:**

An operation is idempotent if performing it multiple times produces the same result.

**Real-time Example (Payments):**

POST /payments (with idempotency-key)

If request retries due to network failure:

- Payment is processed only once

## 41. Which HTTP methods are idempotent?

**Idempotent Methods:**

- GET
- PUT
- DELETE

**Not Idempotent:**

- POST (creates new resource each time)

## 42. What is Pagination?

**Definition:**

Pagination divides large datasets into smaller chunks to improve performance.

**Real-time Scenario:**

Instagram feed:

- Loads 10 posts at a time
- Scroll loads next page

## 43. Pagination Example

```
GET /api/posts?page=2&limit=20
```

### Alternative Approaches:

- Offset-based pagination
- Cursor-based pagination (better for large datasets)

## 44. What is Filtering?

### Definition:

Filtering restricts API responses based on conditions.

### Real-time Example:

```
GET /orders?status=delivered&payment=completed
```

## 45. What is Sorting?

### Definition:

Sorting arranges API response data in a specific order.

### Real-time Example:

```
GET /products?sort=price&order=asc
```

## 46. What is HATEOAS?

### Definition:

HATEOAS (Hypermedia As The Engine Of Application State) means API responses include links to related actions.

### Real-time Example:

```
{
  "orderId": 123,
  "status": "shipped",
  "links": {
    "track": "/orders/123/track",
    "cancel": "/orders/123/cancel"
  }
}
```

}

## 47. What is an API Gateway?

### **Definition:**

An API Gateway is a single entry point that manages requests to multiple backend services.

### **Responsibilities:**

- Authentication
- Rate limiting
- Routing
- Logging

## 48. What is Microservices Architecture?

### **Definition:**

An architecture where applications are split into small, independent services.

### **Real-time Example (E-commerce):**

- User Service
- Order Service
- Payment Service
- Inventory Service

Each has its own API.

## 49. API Gateway – Real-Time Use Case

### **Flow:**

Client → API Gateway → Appropriate Microservice

### **Benefits:**

- Simplifies frontend logic
- Centralized security
- Easier monitoring

## 50. Difference between REST and SOAP?

Feature	REST	SOAP
Protocol	HTTP	XML-based
Format	JSON	XML
Speed	Fast	Slower
Complexity	Simple	Complex
Usage	Modern web apps	Legacy systems

## 51. What is SOAP?

### Definition:

SOAP (Simple Object Access Protocol) is a messaging protocol using XML.

### Real-time Use:

- Banking systems
- Enterprise legacy systems

## 52. What is WSDL?

### Definition:

WSDL describes SOAP service structure, operations, and endpoints.

### Analogy:

WSDL = Contract

API = Implementation

## 53. What is GraphQL?

### Definition:

GraphQL is a query language that allows clients to request exactly the data they need.

**Real-time Example:**

```
{
  user(id: 1) {
    name
    email
  }
}
```

**54. REST vs GraphQL**

**REST Issue:** Over-fetching / Under-fetching

**GraphQL Solution:** Precise data fetching

**55. What is API Throttling?****Definition:**

Temporary control over request rates when server load is high.

**Real-time Example:**

Peak sale on Amazon → throttle traffic to avoid crash.

**56. What is Load Balancing?****Definition:**

Distributes incoming requests across multiple servers.

**Real-time Scenario:**

Netflix traffic spread across multiple servers worldwide.

**57. What is API Mocking?****Definition:**

Simulating API responses without actual backend.

**Real-time Usage:**

Frontend team works before backend is ready.

## 58. What is Contract Testing?

### Definition:

Ensures API provider and consumer follow the same contract.

### Real-time Example:

Frontend expects `email` field → backend must not remove it.

## 59. What is API Latency?

### Definition:

Time taken from request sent to response received.

### Impact:

High latency = poor user experience.

## 60. How to Reduce API Latency?

### Techniques:

- Caching
- Database indexing
- CDN
- Async processing

Perfect 👍

Here is the **final continuation – SECTION 3 (Q61–Q100)** in the **same detailed, definition + elaboration + real-time scenario + interview-ready style**.

## SECTION 3: ADVANCED & REAL-TIME SCENARIOS (Q61–Q100)

## 61. How do you secure an API?

**Definition:**

API security ensures only authorized and valid clients can access resources while protecting data from attacks.

**Key Security Measures:**

- HTTPS (SSL/TLS encryption)
- Authentication (JWT, OAuth, API Keys)
- Authorization (Role-based access)
- Input validation
- Rate limiting
- Firewall & WAF

**Real-time Scenario (Banking App):**

- HTTPS encrypts transactions
- JWT validates users
- Rate limiting prevents brute force
- Validation prevents SQL injection

**62. Explain real-time Login API Flow****Step-by-step Flow:**

1. User enters username/password
2. Client sends request → `POST /login`
3. Server validates credentials
4. Server generates JWT
5. Token returned to client
6. Client sends token in future requests
7. Server verifies token

**Real-time Example:**

Amazon, Netflix, Gmail login flows.

**63. How do you handle API errors?****Definition:**

Returning meaningful responses when something fails.

**Best Practices:**



- Proper HTTP status codes
- Clear error messages
- Logging errors
- Not exposing internal stack traces

#### **Example Response:**

```
{
  "error": "Invalid credentials",
  "code": 401
}
```

## **64. What is Centralized Error Handling?**

#### **Definition:**

Handling all application errors in one place (middleware/global handler).

#### **Benefits:**

- Cleaner code
- Consistent error responses
- Easier debugging

#### **Real-time Use:**

Express.js error middleware

Spring Boot global exception handler

## **65. How do you design a REST API?**

#### **Best Practices:**

- Use nouns, not verbs
- Use correct HTTP methods
- Versioning
- Proper status codes
- Consistent naming
- Pagination, filtering
- Security

#### **Good Design:**

```
GET /users
POST /users
GET /users/{id}
```

**Bad Design:**

```
/getUsers  
/createUser
```

**66. What is API Schema Validation?****Definition:**

Ensures incoming requests match expected structure.

**Why important:**

- Prevents invalid data
- Improves security
- Avoids crashes

**Tools:**

- Joi
- Zod
- JSON Schema
- OpenAPI validation

**67. What is OpenAPI Specification?****Definition:**

A standard format to define REST APIs.

**Uses:**

- Auto-generate documentation
- Generate client SDKs
- Validate requests

**Real-time Example:**

Swagger UI for Stripe, GitHub, PayPal.

**68. What is Middleware?****Definition:**

Code executed between request and response.

**Real-time Uses:**

- Authentication
- Logging
- Validation
- Error handling
- Rate limiting

**Flow:**

Request → Middleware → Controller → Response

**69. Middleware – Real-Time Example****Scenario:**

Every request to `/api/orders`:

1. Verify JWT
2. Log request
3. Validate input
4. Process request

**70. How do you upload files via API?****Definition:**

File uploads use `multipart/form-data`.

**Real-time Example:**

Uploading profile picture to Facebook/Instagram.

**Headers:**

`Content-Type: multipart/form-data`

**71. How do you handle large API responses?****Techniques:**

- Pagination
- Streaming
- Compression (gzip)
- Caching

- Lazy loading

**Real-time Scenario:**

Downloading transaction history from a bank.

## 72. What is API Monitoring?

**Definition:**

Tracking API performance, uptime, and failures.

**Metrics:**

- Response time
- Error rate
- Traffic volume
- Availability

## 73. Tools for API Monitoring

**Common Tools:**

- New Relic
- Datadog
- Prometheus
- Grafana
- ELK Stack

## 74. What is a Webhook?

**Definition:**

A webhook is a way for a server to send real-time data to another system automatically when an event occurs.

**Difference from API:**

- API → Client requests data
- Webhook → Server pushes data

## 75. Webhook – Real-Time Example

### Payment Gateway:

- Payment success
- Payment failure
- Refund processed

Stripe sends webhook → Your server updates order status.

## 76. What is an Event-Driven API?

### Definition:

API communication triggered by events instead of direct requests.

### Examples:

- Order placed
- User registered
- Payment completed

Used in microservices.

## 77. What is a Message Queue?

### Definition:

A queue that stores messages for asynchronous processing.

### Why used:

- Improves scalability
- Prevents blocking
- Handles traffic spikes

## 78. Message Queue Tools

- RabbitMQ
- Kafka
- AWS SQS
- Redis Queue

**Real-time Example:**

Order processing in Amazon.

**79. Difference between Sync vs Async APIs**

<b>Synchronous</b>	<b>Asynchronous</b>
Client waits	Client continues
Blocking	Non-blocking
Slower for heavy tasks	Better for scalability

**Example:**

- Sync: Fetch user profile
- Async: Sending email, generating report

**80. What is Retry Mechanism?****Definition:**

Retrying failed API requests automatically.

**Real-time Scenario:**

Network failure while placing order → system retries safely.

**81. What is Circuit Breaker?****Definition:**

Stops sending requests to failing service to prevent system collapse.

**Real-time Example:**

If payment gateway fails → temporarily stop calls → fallback service.

**82. How do you prevent duplicate API requests?****Techniques:**

- Idempotency keys
- Request hashing

- Unique transaction IDs

**Example:**

Prevent double payment when user clicks “Pay” twice.

### **83. What is API Schema Evolution?**

**Definition:**

Managing changes to API without breaking clients.

**Techniques:**

- Versioning
- Backward compatibility
- Optional fields

### **84. How do you log API calls?**

**Logging Includes:**

- Timestamp
- Endpoint
- Status code
- Response time
- User ID
- Errors

**Why important:**

- Debugging
- Auditing
- Monitoring

### **85. What is API Abuse?**

**Definition:**

Misuse of API intentionally or unintentionally.

**Examples:**

- Brute force attacks
- Scraping
- DDoS
- Excessive requests

**Prevention:**

- Rate limiting
- Captcha
- Authentication

## **86. How do you prevent SQL Injection via API?**

**Techniques:**

- Input validation
- Prepared statements
- ORM
- Escaping inputs

**Bad Example:**

```
"SELECT * FROM users WHERE id=" + userInput
```

## **87. What is API Timeout?**

**Definition:**

Maximum time a client waits for response.

**Real-time Example:**

Payment API timeout after 30 seconds.

## **88. How do you handle concurrent API requests?**

**Techniques:**

- Async processing
- Thread pools
- Queues
- Locks
- Optimistic concurrency control



**Real-time Example:**

Multiple users booking same seat in a train.

**89. What is Blue-Green Deployment?****Definition:**

Deploying new version without downtime.

**Flow:**

- Blue = Old version
- Green = New version
- Switch traffic after testing

**90. What is Canary Release?****Definition:**

Gradually releasing new API version to small % of users.

**Benefit:**

Detect bugs before full rollout.

**91. How do you test APIs automatically?****Tools:**

- Postman automation
- Jest
- Mocha
- Newman
- RestAssured

**Types:**

- Unit tests
- Integration tests
- Load tests

## **92. What is Schema Registry?**

### **Definition:**

Central repository for API schemas.

### **Used in:**

- Microservices
- Event-driven systems

Ensures consistent data structure.

## **93. How do you document APIs for frontend team?**

### **Best Practices:**

- Swagger/OpenAPI
- Examples
- Sample requests/responses
- Error scenarios
- Authentication guide

## **94. How do you handle Backward Compatibility?**

### **Strategies:**

- Do not remove existing fields
- Add optional fields
- Versioning
- Deprecation policy

## **95. What is an API Sandbox?**

### **Definition:**

A test environment where developers can experiment safely.

### **Real-time Example:**

- PayPal Sandbox
- Stripe Test Mode

## 96. What is Rate Limit Exceeded Error?

### Definition:

Occurs when user exceeds allowed request limit.

### HTTP Code:

429 Too Many Requests

## 97. How do you debug API issues in production?

### Techniques:

- Logs
- Monitoring
- Distributed tracing
- Alerts
- Error tracking (Sentry)

## 98. What is Distributed Tracing?

### Definition:

Tracking a request across multiple services.

### Real-time Example:

Client → API Gateway → Auth Service → Order Service → Payment Service

Used in microservices debugging.

## 99. What is API-First Approach?

### Definition:

Designing API before building frontend/backend.

### Benefits:

- Better collaboration
- Clear contracts
- Parallel development

- Better scalability

## 100. Real-Time Interview Question: “Design a Login API”

### Steps:

1. Endpoint: `POST /login`
2. Input validation
3. Authenticate user
4. Generate JWT
5. Return token
6. Secure storage
7. Error handling
8. Logging
9. Rate limiting

### Sample Response:

```
{
  "token": "eyJhbGciOiJIUzI1NiIs... ",
  "expiresIn": 900,
  "user": {
    "id": 101,
    "name": "Mounika"
  }
}
```