Interview Questions

1. **Question (Beginner):** What is API testing and why is it important?

   Answer: API testing involves testing application programming interfaces (APIs) directly and as part of integration testing to determine if they meet expectations for functionality, reliability, performance, and security. It's important because it can help to detect issues early in the development cycle, making them less costly to resolve compared to finding them later in the process.

2. **Question (Beginner):** What is the difference between SOAP and REST APIs?

   Answer: SOAP (Simple Object Access Protocol) is a protocol which was designed before REST and came into the picture. It is a protocol for exchanging structured information in the implementation of web services using XML. REST (Representational State Transfer), on the other hand, is not a protocol but an architectural style. It is simpler and uses less bandwidth and resources.

3. **Question (Beginner):** How do you test APIs using Postman?

   Answer: Postman is a tool used for API testing. It allows you to send requests to an API and view the responses in a user-friendly interface. You can create a new request, enter the URL of the API endpoint you want to test, select the HTTP method, and enter any required headers or body data. After sending the request, Postman will display the response status, headers, and body.

4. **Question (Intermediate):** What is REST Assured? How do you use it for API testing?

   Answer: REST Assured is a Java library that provides a domain-specific language (DSL) for writing powerful, maintainable tests for RESTful APIs. With REST Assured, you can write tests in a BDD-like syntax, which makes

your tests easy to write, read, and maintain. You can use it to send HTTP requests to a RESTful API and validate the response.

5. **Question (Intermediate):** How do you automate API tests with Postman?

   Answer: Postman supports automated testing through the use of test scripts. You can write tests for your API using JavaScript and the Postman Sandbox API. After writing the test script, when you send the request, Postman will run the test script and display the results. You can save the request with the test script for future use. This allows you to re-run the test whenever you want.

6. **Question (Intermediate):** How do you handle authentication in API testing?

   Answer: Authentication in API testing can be handled in several ways depending on the type of authentication used by the API. For basic authentication, you can include the username and password in the request header. For token-based authentication, you can first send a request to the authentication endpoint to get the token, and then include the token in the header of your subsequent requests.

7. **Question (Advanced):** How do you test a RESTful API for concurrency issues?

   Answer: Concurrency issues can be tested by making multiple API calls at the same time and checking if the API can handle them correctly. This can be done using a tool like JMeter or Gatling, which allows you to simulate multiple users sending requests to the API concurrently.

8. **Question (Advanced):** How do you use Newman for running Postman collections?

   Answer: Newman is a command-line collection runner for Postman. It allows you to run and test a Postman collection directly from the command-line. You can install Newman globally on your system using npm, and then run a collection with Newman by providing the URL or file path of the collection.

9. **Question (Advanced):** How would you set up a CI/CD pipeline for your API tests?

Answer: You can set up a CI/CD pipeline for your API tests using tools like Jenkins, Travis CI, or CircleCI. First, you would write your API tests using a tool like REST Assured or Postman. Then, you would configure your CI/CD tool to run your tests whenever changes are pushed to your repository. If any tests fail, the CI/CD tool can alert you so that you can fix the issues before they reach production.

10. **Question (Beginner):** What is the difference between GET and POST methods in HTTP?

    Answer: GET is used to retrieve data from a server. It's safe and idempotent, meaning calling it any number of times will have the same effect. POST is used to send data to a server to create a new resource. It's neither safe nor idempotent.

11. **Question (Intermediate):** How do you test for idempotency in APIs?

    Answer: You can test for idempotency by making the same request multiple times and checking that the API's response is the same each time and that it doesn't change the state of the resource beyond the first request.

12. **Question (Advanced):** How would you handle rate limiting in API testing?

    Answer: Rate limiting can be handled by designing your tests to stay within the API's rate limits. If you exceed the rate limit, you would need to handle the API's rate limit response, typically a 429 Too Many Requests status code, and retry the request after a suitable delay.

13. **Question (Beginner):** What is a status code in API testing?

    Answer: A status code is a number that indicates the result of the HTTP request. It's returned by the server for every request and tells the client whether the request was successful or not, and why.

14. **Question (Intermediate):** What is the role of headers in API testing?

    Answer: Headers in API testing provide additional parameters for the HTTP request or response. They can be used for many purposes, such as providing information about the body content, controlling caching, defining the behavior of proxies and clients, or managing cookies.

15. **Question (Advanced):** How do you test APIs for security vulnerabilities?

    Answer: Security testing for APIs involves techniques like input validation to prevent injection attacks, testing for weak authentication and session management, checking for sensitive data exposure, testing access controls to prevent unauthorized access, and testing for misconfigurations that could expose the system to attack.

16. **Question (Beginner):** What is the difference between stateless and stateful APIs?

    Answer: Stateless APIs do not store any information about the client between requests, meaning each request must contain all the information needed to process it. Stateful APIs, on the other hand, store information about the client's state between requests, such as session data.

17. **Question (Intermediate):** How do you handle versioning in APIs?

    Answer: Versioning can be handled in several ways, such as including the version in the URL or as a parameter, or using custom headers. The goal is to allow multiple versions of the API to exist at the same time, to avoid breaking changes for existing clients.

18. **Question (Advanced):** How would you design an API to be scalable?

    Answer: Scalability can be achieved by designing the API to be stateless, allowing it to be easily load balanced across multiple servers. Other techniques include using caching to reduce load on the server, paginating and limiting response data to reduce network load, and designing the API for asynchronous processing where appropriate.

19. **Question (Beginner):** What is JSON and why is it commonly used in APIs?

    Answer: JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate. It's commonly used in APIs because it's text-based, language-independent, and has a simple syntax, making it a good fit for data exchange between client and server.

20. **Question (Advanced):** How do you handle error reporting and diagnostics in API testing?

Answer: Error reporting can be handled by designing the API to return meaningful error messages and status codes. Diagnostics can be aided by logging important events and using monitoring and alerting tools to detect and notify of any issues.

21. **Question (Beginner):** What is the role of a payload in API testing?

    Answer: The payload in API testing refers to the data sent in the body of the HTTP request or response. In a POST or PUT request, the payload typically contains the data to be created or updated on the server.

22. **Question (Intermediate):** How do you validate the response of an API call?

    Answer: The response of an API call can be validated by checking the status code, headers, and body of the response. This can involve checking that the status code matches the expected value, that the headers contain certain expected values, and that the body contains the correct data.

23. **Question (Advanced):** How do you test an API that uses OAuth for authentication?

    Answer: Testing an API that uses OAuth involves first obtaining an access token by sending a request to the OAuth server with your client credentials and the desired scopes. Once you have the access token, you can include it in the Authorization header of your API requests.

24. **Question (Beginner):** What is the difference between PUT and PATCH methods in HTTP?

    Answer: The PUT method is used to update a resource completely through a specific resource. If the specific resource doesn't exist, then PUT method will create one. The PATCH method is used to apply partial modifications to a resource.

25. **Question (Intermediate):** How do you use environment variables in Postman?

    Answer: Environment variables in Postman can be used to store values that can be reused across multiple requests. You can define environment

variables in the environment settings, and then use them in your requests by wrapping the variable name in double curly braces, like `{{variableName}}`.

26. **Question (Advanced):** How do you test the performance of an API?

    Answer: Performance testing of an API can be done using tools like JMeter or Gatling. You can simulate multiple users sending requests to the API concurrently and measure how the API performs under load. You would typically look at metrics like response time, throughput, and error rate.

27. **Question (Beginner):** What is a RESTful API?

    Answer: A RESTful API is an API that follows the principles of Representational State Transfer (REST). It uses standard HTTP methods, is stateless, and allows for easy communication with other software.

28. **Question (Intermediate):** How do you handle dependencies between API tests?

    Answer: Dependencies between API tests can be handled by using setup and teardown methods to prepare and clean up the test environment for each test. If one test needs to run before another, you can use a test runner that supports test ordering or dependencies.

29. **Question (Advanced):** How do you test a GraphQL API?

    Answer: Testing a GraphQL API involves sending a POST request to the GraphQL endpoint with a JSON payload that contains your query. The response should also be JSON, which you can validate just like a REST API response. You can also test for specific GraphQL issues, like checking that the API returns only the requested fields.

30. **Question (Beginner):** What is an endpoint in API testing?

    Answer: An endpoint in API testing refers to the URL of the API service that you are testing. It's the URI (Uniform Resource Identifier) where the API can be accessed by a client application.

31. **Question (Intermediate):** How do you use the Postman Collection Runner?

    Answer: The Postman Collection Runner allows you to run all requests in a Postman collection in a specified sequence. You can select the collection

and environment, adjust the iteration and delay settings, and then run the collection. The Collection Runner will execute each request in order and display the results.

32. **Question (Advanced):** How do you handle file uploads in API testing?

    Answer: File uploads in API testing can be handled by sending a POST or PUT request with the file included in the body of the request, typically using the multipart/form-data content type. You would then check the response to ensure that the file was uploaded successfully.

33. **Question (Intermediate):** How do you send a GET request to an API using REST Assured?

    Answer: You can send a GET request using REST Assured's `get` method. Here's an example:

    ```
    Response response = RestAssured.get("https://api.exampl
    e.com/users/1");
    ```

34. **Question (Intermediate):** How do you check the status code of an API response using REST Assured?

    Answer: You can check the status code using REST Assured's `statusCode` method. Here's an example:

    ```
    given().
    when().
        get("https://api.example.com/users/1").
    then().
        statusCode(200);
    ```

35. **Question (Advanced):** How do you send a POST request with a JSON body using REST Assured?

    Answer: You can send a POST request with a JSON body using REST Assured's `post` method and `body` method. Here's an example:

```
String jsonBody = "{ \\"name\\": \\"John\\", \\"email
\\": \\"john@example.com\\" }";

given().
    contentType("application/json").
    body(jsonBody).
when().
    post("<https://api.example.com/users>").
then().
    statusCode(201);
```

36. **Question (Advanced):** How do you extract data from an API response using REST Assured?

    Answer: You can extract data from the response using REST Assured's `jsonPath` method. Here's an example:

```
String name =
given().
when().
    get("https://api.example.com/users/1").
then().
    extract().
    path("name");
```

37. **Question (Advanced):** How do you handle authentication in REST Assured?

    Answer: REST Assured supports several types of authentication, including Basic, Digest, Form, and OAuth. Here's an example of Basic authentication:

```
given().
    auth().
    basic("username", "password").
when().
    get("https://api.example.com/users/1").
then().
    statusCode(200);
```

38. **Question (Advanced):** How do you test for specific values in an API response using REST Assured?

    Answer: You can test for specific values using REST Assured's `body` method along with a matcher. Here's an example:

```
given().
when().
    get("https://api.example.com/users/1").
then().
    body("name", equalTo("John"));
```

39. **Question (Advanced):** How do you handle cookies in REST Assured?

    Answer: You can handle cookies in REST Assured using the `cookie` method for a single cookie, or `cookies` for multiple cookies. Here's an example:

```
given().
    cookie("session_id", "123456").
when().
    get("https://api.example.com/users/1").
then().
    statusCode(200);
```

40. **Question (Advanced):** How do you handle redirects in REST Assured?

    Answer: By default, REST Assured automatically follows redirects. If you want to change this behavior, you can use the `redirects().follow(false)` configuration. Here's an example:

```
given().
    redirects().follow(false).
when().
    get("https://api.example.com/users/1").
then().
    statusCode(302);
```

41. **Question (Advanced):** How would you set up REST Assured to use a proxy server?

    Answer: You can set up REST Assured to use a proxy server using the `RestAssured.proxy` method. Here's an example:

    ```
    RestAssured.proxy("proxyhost", 8080);
    ```

42. **Question (Advanced):** How would you send a multipart/form-data request with a file upload using REST Assured?

    Answer: You can send a multipart/form-data request with a file upload using REST Assured's `multiPart` method. Here's an example:

    ```
    File file = new File("/path/to/file.txt");

    given().
        multiPart("file", file).
    when().
        post("https://api.example.com/upload").
    then().
        statusCode(200);
    ```

43. **Question (Advanced):** How would you handle an API that uses OAuth 2.0 authentication with REST Assured?

    Answer: REST Assured supports OAuth 2.0 authentication using the `oauth2` method. Here's an example:

    ```
    given().
        auth().
        oauth2("your_token").
    when().
        get("https://api.example.com/users/1").
    then().
        statusCode(200);
    ```

44. **Question (Advanced):** How would you validate that an API response is sorted in a certain order using REST Assured?

    Answer: You can validate that an API response is sorted using REST Assured's `body` method along with a custom matcher. Here's an example:

    ```
    given().
    when().
        get("https://api.example.com/users").
    then().
        body("name", is(sorted()));
    ```

    Note: The `sorted()` matcher is not built into REST Assured or Hamcrest, so you would need to write it yourself or use a library that provides it.

45. **Question (Advanced):** How would you extract a value from an API response and use it in a subsequent request with REST Assured?

    Answer: You can extract a value from an API response using REST Assured's `extract` method, and then use it in a subsequent request. Here's an example:

    ```
    String userId =
    given().
    when().
        get("https://api.example.com/users/1").
    then().
        extract().
        path("id");

    given().
        pathParam("id", userId).
    when().
        get("https://api.example.com/users/{id}").
    then().
        statusCode(200);
    ```

46. **Question (Advanced):** How would you handle cookies or sessions in REST Assured when making multiple requests?

    Answer: You can handle cookies or sessions in REST Assured using the `cookie` or `cookies` methods. Here's an example:

```
Response response =
given().
    formParam("username", "john").
    formParam("password", "password").
when().
    post("https://api.example.com/login");

String sessionId = response.cookie("session_id");

given().
    cookie("session_id", sessionId).
when().
    get("<https://api.example.com/users/1>").
then().
    statusCode(200);
```

47. **Question (Advanced):** How would you set up a data-driven test with REST Assured, where the test data comes from an external source like a CSV file or a database?

    Answer: You can set up a data-driven test with REST Assured using a testing framework that supports parameterized tests, like JUnit or TestNG. You would write a method to load the test data from the external source, and then use it as the source for your parameterized test.

48. **Question (Advanced):** How would you handle an API that uses a custom authentication scheme with REST Assured?

    Answer: You can handle a custom authentication scheme in REST Assured by manually adding the necessary headers or parameters to your requests. The exact method would depend on the details of the custom authentication scheme.

49. **Question (Advanced):** How would you test an API that uses WebSockets or Server-Sent Events with REST Assured?

    Answer: REST Assured does not directly support testing APIs that use WebSockets or Server-Sent Events. You would need to use a different library that supports these protocols, like AsyncHttpClient or OkHttp.

50. **Question (Advanced):** How would you set up REST Assured to use a custom SSL/TLS certificate for an API that uses HTTPS?

    Answer: You can set up REST Assured to use a custom SSL/TLS certificate using the `RestAssured.config` method and a `SSLConfig` instance. Here's an example:

```
RestAssured.config = RestAssured.config().sslConfig(SSLC
onfig.sslConfig().relaxedHTTPSValidation());
```

This example sets REST Assured to accept all certificates, which is suitable for testing but not for production use. For production use, you would need to load your custom certificate into a `KeyStore` and configure REST Assured to use it.

51. **Question (Advanced):** How would you use REST Assured to test an API that uses GraphQL?

    Answer: You can use REST Assured to test a GraphQL API by sending a POST request with a JSON payload that contains your GraphQL query. Here's an example:

```
String jsonBody = "{ \\"query\\": \\"{ user(id: 1) { nam
e } }\\" }";

given().
    contentType("application/json").
    body(jsonBody).
when().
    post("https://api.example.com/graphql").
then().
```

```
    statusCode(200).
    body("data.user.name", equalTo("John"));
```

52. **Question (Advanced):** How would you use REST Assured to test an API that uses XML instead of JSON?

    Answer: You can use REST Assured to test an API that uses XML by using the `xmlPath` method to extract data from the response, and the `body` method with XPath expressions to validate the response. Here's an example:

```
given().
when().
    get("https://api.example.com/users/1").
then().
    statusCode(200).
    body(hasXPath("/user/name", equalTo("John")));
```

53. **Question (Advanced):** How would you use REST Assured to test an API that uses pagination?

    Answer: You can use REST Assured to test an API that uses pagination by sending multiple requests with different page parameters and checking the results of each page. Here's an example:

```
for (int page = 1; page <= 10; page++) {
    given().
        queryParam("page", page).
    when().
        get("https://api.example.com/users").
    then().
        statusCode(200);
}
```

54. **Question (Advanced):** How would you use REST Assured to test an API that uses rate limiting?

    Answer: You can use REST Assured to test an API that uses rate limiting

by sending multiple requests and checking that the API returns a 429 Too Many Requests status code when the rate limit is exceeded. Here's an example:

```
for (int i = 0; i < 100; i++) {
    given().
    when().
        get("https://api.example.com/users/1").
    then().
        statusCode(anyOf(equalTo(200), equalTo(429)));
}
```

55. **Question (Advanced):** How would you use REST Assured to test an API that uses HATEOAS?

    Answer: You can use REST Assured to test an API that uses HATEOAS (Hypermedia as the Engine of Application State) by checking that the API response includes the correct links, and optionally following those links to test the linked resources. Here's an example:

```
String userIdLink =
given().
when().
    get("https://api.example.com/users").
then().
    statusCode(200).
    extract().
    path("_links.self.href");

given().
when().
    get(userIdLink).
then().
    statusCode(200);
```

56. **Question (Advanced):** How would you use REST Assured to test an API that uses long polling or streaming?

Answer: REST Assured does not directly support testing APIs that use long polling or streaming. You would need to use a different library that supports these features, like AsyncHttpClient or OkHttp.

57. **Question (Advanced):** How would you use REST Assured to test an API that uses HTTP/2?

    Answer: REST Assured does not directly support HTTP/2. You would need to use a different library that supports HTTP/2, like OkHttp.

58. **Question (Advanced):** How would you use REST Assured to test an API that uses JWT (JSON Web Tokens) for authentication?

    Answer: You can use REST Assured to test an API that uses JWT by including the JWT in the Authorization header of your requests. Here's an example:

```
given().
    header("Authorization", "Bearer your_token").
when().
    get("https://api.example.com/users/1").
then().
    statusCode(200);
```

59. **Question (Advanced):** How would you use REST Assured to test an API that uses a non-standard status code?

    Answer: You can use REST Assured to test an API that uses a non-standard status code by using the `statusCode` method with the exact status code. Here's an example:

```
given().
when().
    get("https://api.example.com/users/1").
then().
    statusCode(418);
```

60. **Question (Advanced):** How would you use REST Assured to test an API that uses a non-standard HTTP method, like PATCH or PURGE?

Answer: You can use REST Assured to test an API that uses a non-standard HTTP method by using the `request` method with the method name as a string. Here's an example:

```
given().
when().
    request("PATCH", "https://api.example.com/users/1").
then().
    statusCode(200);
```

61. **Question (Advanced):** How would you use variables in Postman to pass data between requests in a collection?

    Answer: You can use variables in Postman to pass data between requests in a collection. You can set a variable in a test script using `pm.variables.set("variableName", "value")` and then use it in a subsequent request using `{{variableName}}`.

62. **Question (Advanced):** How would you use pre-request scripts in Postman to generate dynamic data for a request?

    Answer: You can use pre-request scripts in Postman to generate dynamic data for a request. For example, you could generate a random number or a current timestamp and store it in a variable, which you can then use in your request.

63. **Question (Advanced):** How would you use test scripts in Postman to validate the response of a request?

    Answer: You can use test scripts in Postman to validate the response of a request. You can use the `pm.response` object to access the response data and the `pm.test` function to write assertions. For example, `pm.test("Status code is 200", function () { pm.response.to.have.status(200); });`.

64. **Question (Advanced):** How would you use Newman to run a Postman collection from the command line?

    Answer: You can use Newman to run a Postman collection from the command line. First, you need to export your collection from Postman to

a JSON file. Then, you can run the collection with Newman using the command `newman run collection.json`.

65. **Question (Advanced):** How would you use Newman to run a Postman collection with a specific environment?

    Answer: You can use Newman to run a Postman collection with a specific environment by exporting the environment to a JSON file in Postman, and then providing the environment file to Newman with the `e` or `-environment` option, like `newman run collection.json -e environment.json`.

66. **Question (Advanced):** How would you use Newman to generate a report of a Postman collection run?

    Answer: You can use Newman to generate a report of a Postman collection run by using a reporter. Newman includes several built-in reporters, like the CLI, JSON, and HTML reporters. You can specify the reporter with the `r` or `-reporters` option, like `newman run collection.json -r html`.

67. **Question (Advanced):** How would you use Postman to test an API that uses OAuth 2.0 authentication?

    Answer: You can use Postman to test an API that uses OAuth 2.0 authentication by setting up the authorization in the request or collection to use OAuth 2.0, and then providing your OAuth 2.0 credentials. Postman can also help you obtain an access token from the OAuth 2.0 server.

68. **Question (Advanced):** How would you use Postman to test an API that uses a custom authentication scheme?

    Answer: You can use Postman to test an API that uses a custom authentication scheme by manually adding the necessary headers or parameters to your requests. The exact method would depend on the details of the custom authentication scheme.

69. **Question (Advanced):** How would you use Postman to test an API that uses WebSockets?

    Answer: Postman supports testing WebSocket APIs. You can create a

new WebSocket request, enter the WebSocket server URL, and then connect to the server. You can then send messages to the server and receive messages from the server.

70. **Question (Advanced):** How would you use Postman to test an API that uses GraphQL?

    Answer: You can use Postman to test a GraphQL API by sending a POST request with a JSON payload that contains your GraphQL query. Postman also provides a graphical interface for writing and testing GraphQL queries.

https://innovateqa.com/rest-assured-interview-questions/

https://www.naukri.com/code360/library/rest-assured-interview-questions

https://www.interviewbit.com/postman-interview-questions/