**What is difference in oauth1 and oauth 2.**

**Oauth 1 is more complex way to generate token. Every time you have to give permission to make new request.**

**But in oauth 2 is simple and generate bearer toke once, and this token is used for every request to make.**

1. What is an API?
2. What is a Web Service? API vs Web Service?
3. What types of Web Services are there? What are the differences?
4. Which Protocol is used by RESTful Web Services?
5. Most Commonly Used HTTP Methods in REST?
6. Can a GET request be used instead of PUT to create a resource?
7. What are the differences between PUT and POST requests?
8. Which HTTP Status Codes do you know?
9. What is API Testing?
10. What are the advantages of API Testing?
11. What is an Endpoint?
12. What is a URI?
13. What API Documentation tools do you know?
14. What is required to send POST, GET, PUT, PATCH, and DELETE calls?
15. What would you expect in a response?
16. What is JSON?
17. What are the two types of Parameters sent with a URI?
18. What are Headers?
19. What is a Payload?
20. How do you verify a value in your response body?
21. What are the main challenges faced in API Testing?
22. What is JSON Path?
23. What would you do if no URI or details are provided, but you have to do API testing?
24. Which data do you compare your API responses with?
25. How do you validate status codes in your project?
26. How do you write a feature file in Cucumber for API Testing?
27. What tools do you use for API testing?
28. What is API versioning? Why is it important?
29. What are the differences between synchronous and asynchronous calls?
30. How do you handle authentication in API testing?

------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1. **What are API’s?**

Application (different software components, UI button, text box web elements) programming (interact with code, not manually) interface (how will interact using set of rules, protocols, like http)

---not a user facing software.

**An API is not a tool or software itself, but it is a set of rules and protocols that allows different software applications to communicate with each other.**

1. **Application**: This refers to different parts of a software system, like the buttons, text boxes, and web elements you see in a program or website. These are the visual components users interact with.
2. **Programming**: This involves writing code to make the application work. Instead of manually interacting with the UI (like clicking buttons), you can use code to automate or control how things function behind the scenes.
3. **Interface**: An interface defines **how** different parts of a system communicate with each other using a set of rules or protocols. For example, **HTTP** is a protocol that defines how a web browser communicates with a server to display a webpage. The interface sets the guidelines for this interaction.

**2. What is a Web Service? API vs Web Service?**

**The services which required internet to communicate with them are called web services.**

All the API which needs internet to connect the website or any application are called **web services. (web mean internet)**

**Remember:**

**All Web Services are APIs, but not all APIs are Web Services.**

Desktop API are not web services they are just APIs. API are parent and web services are child.

**3. What type of Web Services do you know? What are the differences?**

**There are two types of web services???**

**1. REST (Representational State Transfer):**

**The API which has small band width and takes less time to transfer data is called Rest API.**

Bandwidth maximum amount of data that can be transmitted over a network connection in a given amount of time, usually measured in bits per second (bps).

The API which uses HTTP protocols to make request.

**The API which has large band width and takes more time to transfer data is called SOAP API.**

**SOAP (simple object access protocol) more secure, banks use this web service**

**The API which has large band width to transfer the data.**

1. Because it take more bandwidth to transfer the data, it need more time to travel data, it has more information that’s why it take more time.
2. It accept only one language which is xml.
3. Soap is more secure, banks use this soap web services.
4. **Which Protocol is used by RESTful Web Services?**

RESTful web services use HTTP/HTTPS protocols as a medium of communication

between client and server.

**5. Most Commonly Used HTTP Methods supported By REST?**

**POST** – It submits information to the service for processing; it should typically return the modified or new resource → **Create**

**GET** -It requests a resource at the request-URI. It should not contain a request body →

**Retrieve**

**PUT** – Replaces all current representations of the target resource with the uploaded content → **Update**

**PATCH** – Updates only a selected key-paired value → **Update**

**DELETE** – Removes all current representations of the target resource given by a URI →

**Delete**

**6. Can a GET request be used instead of PUT to create a resource?**

The POST or PUT method should be used to create a resource. PUT can be used to

update a resource. GET is only used to request data from a specified resource.

**What are the differences between PUT and POST requests?**

Using **POST** request, our intent is to create a new object on the server whereas with **PUT**

request, our intent is to replace an object by another object (Update)

**8. Which HTTP Status codes do you know?**

**1xx** → Informational

**2xx** → Success (request was accepted successfully) (200→ Ok, 201→ Created, 202→

Accepted, 204→ No Content)

**3xx** → Redirection

**4xx** → Client Error (400-Bad Request, 401-Unauthorized, 403-Forbidden, 404-Not

Found, 405-Method not allowed)

**5xx** → Server Error (500-Internal server Error, 501-Not implemented, 502-Bad Gateway,

503-Service Unavailable)

**9. What is API Testing?**

**What is API testing?? Why do we need API testing???**

**Testing the connection between client and server is created or not we need API testing.**

**------------------------------------------------------------------------------------------------------**

To communicate between different systems (front end and back end), developed in different languages (html,css , java ,python ,c) we need API.

-----------------------------------------------------------------------

**10. What are the advantages of API Testing?**

**Supporting business models:** We need to test web services for many business related reasons. Think about a third party vendor like Expedia. Expedia generates data provided from the producers(Airlines, Hotels, etc.) and that information needs to be correct. If it is not correct, there is a high chance of the business losing money.

**Early Detection of Issues**: Since APIs are tested at the code level, bugs and issues are identified earlier in the development process.

**Platform Independence**: APIs are independent of the user interface, allowing tests to be conducted even when the UI is incomplete. This also ensures that APIs function correctly across various platforms (web, mobile, etc.).

**12. What is End Point?**

An endpoint by itself is the location where actual resources or data is present in the data base.

**13. What is a URI?**

Uniform Resource Identifier

URI = Domain/Base URL + endpoint URL/service URL/resource

**14. Do you have an API documentation website for your APIs? Any other API documentation that you know of?**

Some of the API documentation templates:

● Swagger

● FlatDoc

● RestDoc

● API blueprint

**15. Can you tell me what is required to send a POST, GET, PUT, PATCH, and DELETE**

**calls?**

With **POST** you will need:

● URI

● Headers

● BODY/Payload (your data in JSON, XML, String, etc)

With **GET** you will need:

● URI

● Headers

● No BODY/Payload is required since GET you are only retrieving data from a

server and not creating

● If you need to send data with a GET call to narrow down your search then you

can send your data in form of JQuery Parameters or Path Parameters

With a **PUT**(update) call you will need:

● URI

● Headers

● AND a body/payload

● Note: If you are attempting to update information that does not exist in given

server then PUT will behave as a POST call and create the information UNLESS

developers have restricted that from happening

With a **DELETE** call you will need:

● URI

● Headers

● AND you may/may not need a payload

● If a payload is not required then you will send data in form as JQuery parameter

or PATH parameter

HTTP request method is made up of four components:

● Request Method: Get, Post, Put, Delete

● Request URI: complete URL of the resource

● Request Header: Accept, Content-Type

● Request Body: data to be sent to the resource

**16. What would you expect in a response?**

HTTP response method is made up of three components:

● Response Status Code:200, 201, 400, 404, 500

● Response Header: Date, Server, Last-Modified, Content-Type

● Response Body:data that comes back to the client from the server

**17. What is JSON?**

**Is a light weight data interchange format to make request and get response in rest api.**

● It is JavaScript Object Notation (is a minimal, readable format for structuring

data.)

● It is used primarily to transmit data between a server and web application, as an

alternative to XML (a lightweight version of XML)

● Represents Data in a Key : Value format

● JSON is NOT a programming language

**18. What are two types of Parameters sent with a URI?**

Parameters are options you can pass with the endpoint to influence the response.

In REST we 2 types of Parameters:

● Path Parameters

As part of the URL-path (i.e. /api/resource/parametervalue )

● Query Parameters

As a query argument (i.e. /api/resource?parameter=value )

**19. What are Headers?**

In Postman, a **header** in an API request refers to additional information sent along with the request, providing metadata about the request. Headers are key-value pairs that can include data about:

Headers provide meta-information about a request.

In my project when I send POST or PUT requests, as headers, I specify the ContentType

and an authorization JSON Web Token.

When I receive a response, I verify response headers such as the Content-Type.

**RestAssured.baseURI="http://pure-ravine-92491.herokuapp.com/syntax";**

**Response rsp=**

**given().**

**accept(ContentType.JSON).**

**header("Content-Type", "application/json").**

**body(map).**

**when().**

**post("/api/createStudentProfile")**

**rsp.then().assertThat().statusCode(201).**

**and().header("Content-Type", equalTo("application/json;charset=UTF-8"));**

**20. What is a Payload?**

Requests and response bodies of every HTTP message includes request data called

Payload.

We send a payload with POST, PUT, and PATCH calls but not in GET and DELETE

calls.

The response payload is often a response of you GET (returning a record or a list of

records), or a confirmation from a POST.

**21. How do you verify a value in your Response body?**

**Chai assertion library using postman object**

**2. HamCrest Matchers for automation in java.**

**RestAssured.baseURI="https://got-quotes.herokuapp.com";**

**Response rsp=**

**given().**

**queryParam("char", "varys").**

**when().**

**get("/quotes")**.

**then().**

**and().**

**assertThat().statusCode(20**

**0).**

**Assert that ().body ("character", equal to ("Varys"));**

**22. What are the main challenges faced in API testing?**

**1.** **Selecting proper parameters** **and its** **combinations** (what if you do not have correct documentation and you need to work with parameters).

**2.** **Difficult Verifying and validating the of response body** (Request & Response)

**3. Due to absence of GUI it is quite difficult to provide input values**

**----------------------------------------------------------------------------------------------**

**Selecting Proper Parameters and Combinations**:

* **Challenge**: Incomplete or inaccurate documentation can lead to confusion about which parameters to use and their valid combinations.

**Categorizing Parameters Properly**:

* **Challenge**: Differentiating between path parameters and query parameters can be tricky, especially in complex APIs.

**Verifying and Validating Output**:

* **Challenge**: Ensuring that the response matches expected outcomes can be complex, especially when dealing with dynamic data or large response bodies.

**Absence of GUI**:

* **Challenge**: Without a graphical interface, providing input values can be cumbersome, especially for non-technical stakeholders.

**23. What is the JSON path?**

Json Path () is a method provided by RestAssured java library that allows you to extract data from response JSON body.

**Methods Available in JSONPath:**

* **getString()**: Returns the value as a String.
* **getInt()**: Returns the value as an int.

JsonPath () to extract values from JSON responses

**24. What would you do if you do not have URI or anything else provided but you have to do API testing?**

If no **URI** (Uniform Resource Identifier) or other specific details are provided for API testing, you can still approach the task using the following strategies:

**Ask for missing details**: Reach out to the responsible person (developer, project manager, etc.)

use **network tools** (like the browser’s developer tools or a proxy tool like **Postman Interceptor** or **Fiddler**) to inspect network requests and extract API URIs.

**25. Which data do you compare your API responses with?**

"I would compare the API responses with the API documentation to ensure they match the expected structure and values. Additionally, if required, I would validate the data directly against the database to confirm accuracy

**26. How do you validate status codes in your project?**

I validate status codes using the Chai assertion library with JavaScript for manual testing, leveraging its built-in assertion methods. For automation, I use the Hamcrest assertion library, where I can validate responses with syntax like response.then().assertThat().statusCode("expected status code").

**27. How do you write a feature file in cucumber for API testing?**

"In my project, I write a feature file in Cucumber by defining scenarios that include a Background section to generate a token as a precondition for any execution.

The structure looks like this:

gherkin

Copy code

Feature: API Testing

Background:

Given I have generated a token

Scenario: Verify API response

Given I prepare a request file

When I call the API/endpoint

Then I perform assertions on the response

This approach ensures that the token is available for all scenarios and maintains clarity in the test execution process."

**What is the JSON path?**

**JSON Path** is a query language for JSON data that allows you to extract specific elements using a syntax similar to XPath for XML. It enables users to navigate JSON structures and retrieve values easily, such as accessing properties and filtering data.

**What baseline metrics/requirements are necessary for starting to prepare a performance/load test profile for a service that has had no previous performance/load testing performed?**

I have not performed load or performance testing so in all honesty I am not sure how to

do that.

**30. What performance and testing approaches would you perform on the service?**

**Performance and Testing Approaches**

1. **Review API Documentation**:
   * Start by closely examining the API documentation to understand the endpoints, parameters, and expected behaviors.
2. **Consult Stakeholders**:
   * If anything is unclear, ask Business Analysts (BAs) and developers for clarification to avoid assumptions.
3. **Environment Setup**:
   * Determine if you need to test in different environments (e.g., development, staging, production).
   * Use tools like Postman to create environment variables for easy adjustments across tests.
4. **Use Global Variables**:
   * Set global variables for any common data or parameters. This allows you to reuse them easily in multiple requests, e.g., by using {{variable}}.
5. **Tool Utilization**:
   * Use Postman or SoapUI to run your tests efficiently, leveraging their features for variable management and testing automation.

**31. You just mentioned SoapUI in your last answer, have you worked with SoapUI?**

No I have not, but I was very curious to learn how I would be able to use other

technologies to test API’s so I did some research on the side. - This will make you look

good.

**32. What does 401 status code mean? 301?**

**401** - Unauthorized - meaning we need a token or key to authenticate ourselves and hit

the API successfully.

**33. What is a WSDL and a WADL file?**

### WSDL and WADL Overview

1. **WSDL (Web Service Description Language)**:
   * **Definition**: WSDL is an XML-based document that describes SOAP-based web services.
   * **Purpose**: It defines the service's available methods, the expected input parameters, and the data types returned by these methods.
   * **Usage**: WSDL files can be loaded into API tools, enabling developers to understand how to interact with the web service effectively.
2. **WADL (Web Application Description Language)**:
   * **Definition**: WADL is an XML document used to describe RESTful web services.
   * **Purpose**: It outlines how to access the service, including the available resources, the methods supported, and the expected input and output formats.
   * **Usage**: Similar to WSDL, WADL files can be loaded into API tools, providing users with immediate access to the web service's functionality.

**34. What are some principles of an API test design?**

**Some very important principles of an API test design are:**

Setting up - Setting up your test environment, think of how we set everything up with a

postman. Global, environment variables, JWT generation, etc

Execution - How did we execute our test cases in postman? Did we have a flow?

Verification - what and how did we verify?

Reporting - How did we generate our collection runner or cucumber report?

Clean up - How did we unset variables in postman

**How did you generate your token or key without having to manually do so all the time?**

With Postman - grabbed the token from the JSON object response and stored it as a global variable to be used with all calls that required a token

Rest assured with cucumber - created a separate class, made a call to generate a token and stored the token as a Static variable - we used this as a “Given” background step to be applied to all calls

**36. What is stateful and stateless application?**

**State-full Application:**

The application which Remembers past history of user interactions with application. Server has memory.

* Example: face book, Gmail remember your password and you don’t need provide again and again.

--------------------------------------------------------------------------------------------------------------------------------------------

**Stateless Application:**

The application which Forgets past history of user interactions with application and treats each request

Independently. Server don’t have memory. Generate token.

* Example: banking application.

Stateless application do not have server memory. They generate token .This is the difference only.

**What is a pm object?**

**Pm is the object which gives various methods to manipulate with api.**

The **pm object** in Postman is a built-in object that provides various methods and properties to facilitate testing and scripting within the Postman environment. It allows users to write test cases, validate responses, and perform assertions on API requests.

**38. What are the details bearer token contains?**

In API testing when we create a bearer token (JWT), it contains 3 components namely

header, payload and signatures.

**39. What is REST Assured?**

To perform API testing via automation, Rest Assured (JAVA based Library) is being used for Restful API’s. It follows the BDD approach where given is used for preparing the request, when it is used to hit the end point and then is used to verify the expected response.

**40. What are Presentation, Application and Data layers?**

Presentation - The layer which exists at client end where GUI is available for.

Application - The layer where actual business logic is written.

Data - The layer where data is stored (Database).

**41. What is Authentication and Authorization?**

Authentication is used to check the identity and existence of a user in the system and

Authorization is the process of checking the privileges logged in user has for the system.

**42. What are path parameters and query parameters?**

The parameters are the ones which are considered as the path of BaseURI and come after

/ whereas query parameters are used to access specific data and they come after ? in key

value format

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

 **What is the difference between REST and SOAP APIs?**

* **REST** is an architectural style that uses HTTP methods (GET, POST) and supports formats like JSON and XML. It's lightweight and easy to integrate. **SOAP** is a protocol that uses only XML, is more rigid, and includes built-in features like security (WS-Security) and ACID compliance, making it more complex.

 **What is the difference between PUT and POST?**

* **PUT** is used to update or replace an existing resource, ensuring idempotency (repeated requests produce the same result). **POST** is used to create a new resource or submit data, and multiple POSTs can result in different outcomes.

 **How do you use a collection in API (Postman)?**

* I create **collections** in Postman to group multiple API requests, which helps me run tests in bulk, save time, and share requests with the team. Collections also allow for environment setup and better test organization.

 **How do you do API testing?**

* I test APIs by validating the **response status codes**, headers, and data against the documentation. I also check for edge cases, error handling, and proper authentication using tools like **Postman** for manual testing and **REST Assured** for automation.

 **Which data do you compare API with?**

* I compare the API response with the **API documentation** to verify that the expected structure and data are correct, and sometimes I validate data against the **database** to ensure consistency between the backend and the API.

 **How do you compare two status codes from different APIs?**

* I use tools like **Postman** or write scripts in **JavaScript or REST Assured** to fetch and store the status codes from different APIs, then compare them to ensure they return the expected results.

 **How do you write a feature file in Cucumber for API testing?**

* I write **Gherkin scenarios** in the feature file, outlining steps such as making API calls, validating the response structure, and checking for status codes. I include preconditions like generating tokens in the **Background** section to set up authentication before running the tests.

 **How do you understand End-to-End testing and Back-end testing?**

* **End-to-End testing** ensures the entire user journey, from UI to database, works as expected across different systems. **Back-end testing** focuses on testing APIs, databases, and other server-side logic without involving the user interface to verify data processing and storage.

**---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Modified short answer of your questions**

### 1. ****What is an API?****

An **API (Application Programming Interface)** is a set of protocols and tools that allows different software applications to communicate with each other. It defines the methods and data formats that applications can use to request and exchange information. APIs enable developers to integrate different services and functionalities without needing to understand the underlying code of those services.

### 2. ****What is a Web Service? API vs Web Service?****

A **Web Service** is a specific type of API that operates over a network (typically the internet). It allows different applications to communicate using standardized protocols such as HTTP.

* **Key Difference**:
  + **API**: A general term that refers to any interface that allows software to communicate, regardless of the transport mechanism.
  + **Web Service**: A specific implementation of APIs that uses web protocols, such as HTTP or HTTPS.

### 3. ****What types of Web Services are there? What are the differences?****

There are primarily two types of web services:

1. **REST (Representational State Transfer)**:
   * Utilizes HTTP requests for communication.
   * Can return data in various formats, most commonly JSON or XML.
   * Stateless, meaning each request from client to server must contain all the information the server needs to fulfill the request.
2. **SOAP (Simple Object Access Protocol)**:
   * Utilizes XML exclusively for message format.
   * More protocol-heavy, requiring a strict set of standards for communication.
   * Can be more secure due to built-in error handling and support for ACID compliance.

### 4. ****Which Protocol is used by RESTful Web Services?****

**RESTful web services** primarily use **HTTP/HTTPS** protocols as their communication medium.

### 5. ****Most Commonly Used HTTP Methods in REST?****

* **POST**: Used to create new resources on the server.
* **GET**: Used to retrieve resources from the server.
* **PUT**: Used to update or replace existing resources.
* **PATCH**: Used to make partial updates to a resource.
* **DELETE**: Used to remove resources from the server.

### 6. ****Can a GET request be used instead of PUT to create a resource?****

No, a **GET** request is designed only for retrieving data. To create a resource, a **POST** request should be used, while **PUT** can also be used if the intention is to update an existing resource or create it if it doesn’t already exist.

### 7. ****What are the differences between PUT and POST requests?****

* **POST**:
  + Used to create new resources.
  + Each POST request can create a new resource instance.
* **PUT**:
  + Used to update an existing resource.
  + If the resource does not exist, it may create one, but it replaces the entire resource.

### 8. ****Which HTTP Status Codes do you know?****

* **1xx**: Informational responses (e.g., 100 Continue).
* **2xx**: Success codes (e.g., 200 OK, 201 Created).
* **3xx**: Redirection codes (e.g., 301 Moved Permanently).
* **4xx**: Client error codes (e.g., 400 Bad Request, 404 Not Found).
* **5xx**: Server error codes (e.g., 500 Internal Server Error).

### 9. ****What is API Testing?****

**API Testing** is a software testing practice that evaluates the functionality, reliability, performance, and security of application programming interfaces (APIs). It focuses on testing the endpoints of the API to ensure they are working as intended and can handle various types of requests.

### 10. ****What are the advantages of API Testing?****

* **Early Bug Detection**: Identifying issues during the early stages of development.
* **Platform Independence**: APIs can be tested independently of the user interface, allowing for concurrent development.
* **Cost-Effective**: Detecting issues early reduces costs associated with fixing bugs in later stages.
* **Supports Integration Testing**: Ensures that APIs work correctly when integrated with other components.

### 11. ****What is an Endpoint?****

An **Endpoint** is a specific URL where an API can be accessed. It is a point of interaction between the client and server, defining the location where API requests are sent.

### 12. ****What is a URI?****

A **URI (Uniform Resource Identifier)** is a string used to identify a resource. It consists of two components: the base URL (domain) and the endpoint (path). For example, in https://api.example.com/users, the URI consists of https://api.example.com as the base URL and /users as the endpoint.

### 13. ****What API Documentation tools do you know?****

Some commonly used API documentation tools include:

* **Swagger/OpenAPI**: Provides interactive API documentation and testing.
* **Postman**: Offers API documentation alongside testing capabilities.
* **Apiary**: A collaborative platform for API design and documentation.
* **ReadMe**: A customizable API documentation tool.

### 14. ****What is required to send POST, GET, PUT, PATCH, and DELETE calls?****

* **POST**: Requires the URI, headers (like Content-Type), and body (data to be sent).
* **GET**: Requires the URI and optional headers, but no body is needed.
* **PUT**: Requires the URI, headers, and body (updated data).
* **PATCH**: Requires the URI, headers, and body (partial data for updates).
* **DELETE**: Requires the URI and optional headers, but typically no body.

### 15. ****What would you expect in a response?****

In an API response, you would typically expect:

* **Status Code**: To indicate success or failure (e.g., 200 OK).
* **Response Headers**: Metadata about the response (e.g., Content-Type, Date).
* **Response Body**: The actual data returned, usually in JSON or XML format.

### 16. ****What is JSON?****

**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 is commonly used in APIs to send data between clients and servers.

### 17. ****What are the two types of Parameters sent with a URI?****

1. **Path Parameters**: Included in the endpoint URL itself (e.g., /users/{userId}).
2. **Query Parameters**: Sent in the URL after a question mark, in key-value pairs (e.g., /users?name=John).

### 18. ****What are Headers?****

Headers are key-value pairs sent in an API request or response. They provide additional context or information, such as:

* **Content-Type**: Indicates the media type of the resource (e.g., application/json).
* **Authorization**: Contains credentials for authenticating the request.

### 19. ****What is a Payload?****

A **Payload** refers to the data that is sent in the body of an API request, particularly for methods like POST, PUT, and PATCH. It contains the information that the server processes.

### 20. ****How do you verify a value in your response body?****

You can verify values in the response body using testing tools like Postman or automated testing frameworks. For example, in Postman, you can write tests in the Tests tab to assert specific values using the pm object.

Example:

javascript

Copy code

pm.test("Check user ID", function () {

var jsonData = pm.response.json();

pm.expect(jsonData.id).to.eql(1);

});

### 21. ****What are the main challenges faced in API Testing?****

* **Parameter Management**: Handling multiple path and query parameters effectively.
* **Data Validation**: Ensuring the returned data matches expected formats and values.
* **Error Handling**: Testing how the API responds to invalid requests.
* **Limited Documentation**: Sometimes API documentation is incomplete or unclear.

### 22. ****What is JSON Path?****

**JSON Path** is a query language used to extract specific parts of a JSON document. It provides a syntax similar to XPath for XML. For example, the expression $.store.book[0].title would return the title of the first book in a JSON structure.

### 23. ****What would you do if no URI or details are provided, but you have to do API testing?****

If no URI or details are available, I would:

* Reach out to developers or business analysts to request the necessary information.
* Use tools like Postman Interceptor or browser developer tools to capture API calls from the frontend to infer the endpoints.

### 24. ****Which data do you compare your API responses with?****

I compare API responses against the expected output specified in the API documentation. Additionally, I validate the data against the database if necessary to ensure accuracy and consistency.

### 25. ****How do you validate status codes in your project?****

In **Postman**, I use the Chai assertion library to validate status codes. For automation, I employ **Hamcrest** assertions in **REST Assured** to ensure the correct status code is returned for each request.

Example:

java

Copy code

response.then().statusCode(200);

### 26. ****How do you write a feature file in Cucumber for API Testing?****

In Cucumber, I write feature files using **Gherkin syntax**, defining scenarios that describe the behavior of the API.

Example:

gherkin

Copy code

Feature: User API

Scenario: Create a new user

Given I have a valid request body

When I send a POST request to "/users"

Then I should receive a 201 response

### 27. ****What tools do you use for API testing?****

I use tools like:

* **Postman**: For manual testing and API exploration.
* **SoapUI**: For functional and performance testing of SOAP and REST services.
* **REST Assured**: For automation testing of REST APIs.
* **JMeter**: For performance testing of APIs.

### 28. ****What is API versioning? Why is it important?****

**API versioning** is the practice of managing changes to an API over time, ensuring backward compatibility. It allows developers to introduce new features or changes without disrupting existing users. Versioning can be implemented in the URL, headers, or query parameters.

### 29. ****What are the differences between synchronous and asynchronous calls?****

* **Synchronous Calls**: The client sends a request and waits for the server to respond before proceeding. This can lead to slower performance and potential timeouts.
* **Asynchronous Calls**: The client sends a request and continues with other tasks without waiting for the server's response. This improves efficiency and allows for better resource utilization.

### 30. ****How do you handle authentication in API testing?****

Authentication can be managed using various methods:

* **Basic Authentication**: Sending a username and password in the request headers.
* **Bearer Tokens**: Using JSON Web Tokens (JWT) for secure access.
* **OAuth**: Allowing third-party applications to access user data with delegated access.