API

1. **What is API?**
   - It means connectivity. I mean API is the messenger that takes requests and tells a system what you want to do and then returns the response back to you.
   - API is the acronym for **Application Programming Interface** (which is software intermediary) that allows how applications to talk to each other.

2. **API vs Webservices?**
   - API = *browser*: Selenium WebDriver, *database*: JDBC, *MsOffice*: Apache POI
   - Webservices = if an API uses internet for communications, it is a webservices. *All webservices are API.
   - No UI (user interface) → web application with UI and we use Selenium Webdriver
   - We use:
     - SOAP ⟶ XML
     - REST ⟶ JSON, XML, TEXT
     - Postman, Rest Assured Library

3. **What is SoapUI? and how did you use it in your current project?**
   - SOAP UI is the leading open source cross-platform API Testing tool
   - SOAPUI allows testers to execute automated functional, regression, compliance, and load tests on different Web API.
   - SOAPUI supports all the standard protocols and technologies to test all kinds of API's.
   - SOAPUI interface is simple that enables both technical and non-technical users to use seamlessly.

4. **Name of some commonly used HTTP methods in REST based architecture?**
   - Create ⟶ POST (send data to the server)
   - Read ⟶ GET (retrieves data from given server using a given URI)
   - Update ⟶ PUT (Replaces all current representations of the target resource with the uploaded content)
   - Delete ⟶ DELETE (Removes all current representations of the target resource given by a URI.)

5. **HTML Status Codes?**
   - 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, 502-Bad Gateway, 501-Not implemented, 503-Service Unavailable)

6. **What first thing you check when you get response?**
   - Status quote (200 always mean Ok)
   - We always check the 404 means not found
   - rest-assured.io==> for automation to find the ECS machine in search type remote Desktop

7. **Http methods and request types**
   - Get does not requires body
   - Put requires body means UPDATE information
   - Post requires body means CREATE information
   - Delete does not requires body
   - GET -> READ , POST -> CREATE, PUT -> UPDATE, DELETE -> DELETE
   - POST VS PUT

8. **Parameters api**

- 2 TYPES:
  - PATH PARAMETER(VALUE WILL BE PART OF URL) QUERY/REQUEST
  - PARAMETERS (KEY+ VALUE FORMAT)

9. **What is Hamcrest Matcher for?**

- Hamcrest is a framework for writing matcher objects allowing 'match' rules to be defined declaratively.

```
import org.junit.jupiter.api.Test;
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.*;

public class BiscuitTest {
  @Test
  public void testEquals() {
    Biscuit theBiscuit = new Biscuit("Ginger");
    Biscuit myBiscuit = new Biscuit("Ginger");
    assertThat(theBiscuit, equalTo(myBiscuit));
   assertThat("chocolate chips", theBiscuit.getChocolateChipCount(), equalTo(10));
   assertThat("hazelnuts", theBiscuit.getHazelnutCount(), equalTo(3));
  }
}
```

```
// verify if first argument is equal to the second
assertThat(str1, is("Kunkka"));
assertThat(str1, is(str2));

// verify if first argument is NOT equal to the second
assertThat(str1, is(not("Tidehunter")));

// compare ignoring case
assertThat(str1, equalToIgnoringCase("kunkka"));

// compare ignoring space before and after
assertThat(str1, equalToIgnoringWhiteSpace(" Kunkka "));

// compare numbers
assertThat(10, greaterThan(9));
assertThat(10, lessThan(11));
assertThat(10, lessThanOrEqualTo(11));

// verify not null
assertThat(str1, notNullValue());

List<String> list = Arrays.asList("one", "too", "tree");
   assertThat(list, hasSize(3));
   assertThat(list, containsInAnyOrder("too", "tree", "one"));
   assertThat(list, hasItems("one", "too"));

List<Integer> numbers = Arrays.asList(11, 12, 13);
assertThat(numbers, everyItem(greaterThan(9)));
```

10. **RestAssured Log Logging Logs**

- **Request Logging**

```
given().log().all() // Log all request specification details including parameters, headers and body
given().log().params() // Log only the parameters of the request
given().log().body() // Log only the request body
given().log().headers() // Log only the request headers
given().log().cookies() // Log only the request cookies
given().log().method() // Log only the request method
given().log().path() // Log only the request path
```

- **Response Logging**

```
get("/x").then().log().body()
get("/x").then().log().ifError()
get("/x").then().log().all()
get("/x").then().log().statusLine() // Only log the status line
get("/x").then().log().headers() // Only log the response headers
get("/x").then().log().cookies()  // Only log the response cookies
get("/x").then().log().ifStatusCodeIsEqualTo(302)
   // Only log if the status code is equal to 302
get("/x").then().log().ifStatusCodeMatches(matcher)
   // Only log if the status code matches the supplied Hamcrest matcher
```

11. **Serialization and Deserialization**

- Serialization; when we MAP a Java object to API JSON format (CONVERT JAVA OBJECT TO JSON);
  - Java object (POJO(Plain Old Java Object), BEANS) → MAP it to API JSON/XML
  - When we have an object from a class and MAP it to a JSON format in our RESTful API

```
{make: “Toyota”,
Model: “Camry” }
Car car = new Car();
car.setMake(“Toyota”);
car.setModel(“Camry”);
given().body(car).when().post(uri)
```

- Deserialization; API JSON/XML → MAP it to Java Object (JSON TO JAVA OBJECT)

```
Car car2 = new Car();
car2=when().get(uri).body.as(car.class);
car.setMake(“Toyota”);
car.setModel(“Camry”);
```

12. **API/Webservices with RestAssured Library?**

```
import static io.restassured.RestAssured.* ;
URI uri = new URI(“ ... / methods(get,post)”)
```

- GET;

```
Response response = given().accept(ContentType.JSON).when().get(URI);
response.then().assertThat().statusCode(200).
  and().assertThat().ContentType(ContentType.JSON);
```

- POST;

```
Response response = given().ContentType(ContentType.JSON).with().accept(ContentType.JSON)
.and().body(JSONbody).when().post(URI);
response.then().assertThat().statusCode(200);
```

-

```
import static org.hamcrest.Matchers.* ;
then().assertThat().body(“Id”,Matchers.equalTo(123));
```

-

```
JsonPath json = JsonPath(JSONbody);
json.getString(“key”);
json.getInt(“key”);
json.getList(“key1.key2”);
```

13. **What is EndPoint?**

- <protocol>://<service-name>/<ResourceType>/ResourceID → URI (Uniform Resource Identifier)
  Base URI / resource ? Parameters
  (http://www.google.com/search?source=book...) → ? → query parameters

14. **Authorization vs Authentication**

- authentication --> who are you
- authorization --> what rights do you have
- Authentication is user and password
- Authorization has types:
  - no Authorization
  - Basic Authorization
  - Bearer Token
  - Inherit Auth from parent

15. **RESTful Web Service / API**

- REST stands for Representational State Transfer
- RESTful is referred for web services written by applying REST architectural concept.
  - In RESTful, web service http methods like GET, POST, PUT, DELETE can be used to perform CRUD operations.
  - CRUD = Create → Read → Update → Delete

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

- For exp: verify ID contains correct number
  - *Hamster Matcher* is assertion library.

```
then().assertThat().body("Id",Matchers.equalTo(123));
```

  - Parse into JsonPath and use getInt(), getList(), getString() methods to read Id value.
    And, I can use JUnit Assertion:

```
String body = ...thenReturn().body().asString();
JsonPath json = new JsonPath(body);
assertEquals(123,json.getInt("Id"));
```

  - De-serialize into a (POJO) object (or Object Mapping)

```
POJO myPojo = … when().post(url).thenReturn().body().as(Pojo.class);
assertEquals(123,myPojo.getId( ) );
```

And, I can use JUnit Assertion.

17. **Types of API's Authentication**

- Basic
  - Pre-emptive
    - If a service is configured to be pre-emptive, it will not request credentials from a client even though it requires it.
    - If a request doesn't contain credentials, it will return **401 Unauthorized** status code.
  - Challenged
    - When request reaches the API then API will tell that it requires credentials then client will provide credentials.
  - oauth --> types of authentication where keys and tokens from 3rd party is used to authenticate. There are 2 types of oauth:
    - oauth1 → hard to implement
    - oauth2 → more secure
- Digest
  - It is more encrypted than basic. https...

18. **What is the advantage of using SOAP?**

- REST allows a greater variety of data formats, whereas SOAP only allows XML.
- Coupled with JSON (which typically works better with data and offers faster parsing), REST is generally considered easier to work with.
- Thanks to JSON, REST offers better support for browser clients.
- REST provides superior performance, particularly through caching for information that's not altered and not dynamic
- It is the protocol used most often for major services such as Yahoo, Ebay, Amazon, and even Google.
- REST is generally faster and uses less bandwidth. It's also easier to integrate with existing websites with no need to refactor site infrastructure. This enables developers to work faster rather than spend time rewriting a site from scratch. Instead, they can simply add additional functionality.

19. **Difference between SOAP and RESTful web services?**
    - RESTful supports JSON, XML, TEXT, however SOAP supports only XML
    - REST is faster than SOAP based web services

20. **What is URI, purpose and format?**
    - URI stands for Uniform Resource Identifier
    - The purpose of URI is to locate a resource on the server hosting the web service.
    - A URI is of the following format:
        - <protocol>://<service-name>/<ResourceType>/<ResourceID>

21. **What WebServices do you use in your project?**
    - I use Restful which is Representational State of Transfer and it communicates with XML and JSON, but my current project uses JSON

22. **What is XML?**
    - In computing, Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

23. **What is JSON?**
    - 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.
    - Basically, a lightweight version of XML
    - In Key: Value format
    - Key is always in double quotes and value if string its double quotes and if numbers no quotes
    - It is purely based on http protocol, - so it hits the link on the browser and see the results

24. **Do you know swagger? What is swagger**
    - Swagger is an open-source software framework backed by a large ecosystem of tools that helps developers design, build, document, and consume RESTful Web services.
    - Swagger allows you to describe the structure of your APIs so that machines can read them.
    - The ability of APIs to describe their own structure is the root of all awesomeness in Swagger
    - similar to xml schema but for Json

25. **json vs gson**
    - JSON is a format which has key and values
    - GSON is a process of converting
        - from java to json(serialization),
        - from json to java(deserialization)

26. **How and where are you sending request?**
    - Since i am using Rest, it has endpoints. My developers create public URLs and requests are sent to that URL

27. **Do you use any non-web services API?**
    - - I use Selenium API for browser, JDBC for database, and RestAssured for API

28. **Do you have API documentation website for your API?**
    - Yes, we use swagger for our api documentation, and this is where the description and guidelines of API endpoints are

29. **How do you test API in your project?**
    - In my current project we are testing not only our companies api but other external api.
        - For example, we use LinkedIn api to easily transfer the authorized end user's info to our database.
    - As a tester we send a API request and verify the status code, response body and checking the endpoints of the api URL is working as expected
        - For example, in my project, I also do Positive/Negative testing of API 57
    - Positive - I am sending valid requests, headers, parameters, and Json body and verify that response is 200/201
    - Negative- I am sending invalid requests, headers, parameters, and body, expecting to the status to not be 200

30. **How do you test rest api?**
    - I verify if each REST API endpoint is working as expected.
    - I use POSTMAN for manual API testing and use RESTASSURED library in Java for automation.
    - I send POST,PUT,GET, DELETE type of requests and verify response status code and response body, header.
    - I also do positive and negative testing of API.
    - When I do positive testing, I send valid request parameters , valid headers, valid request json body and verify that response status code is 200 successful and Json response body data is also matching the expected.
    - When I do negative testing, I send invalid request parameters , or invalid headers, or invalid request json body and verify that response
    - status code is not 200 and Json response body contains error message.
31. **Can All API endpoints use all of the Http protocols?**
    - It depends, My API developer decides if that URL works with GET,POST,PUT, or DELETE requests

32. **How do you manually test your API?**
    - I use Postman → it is a REST API client tool that test the REST API URL

33. **What tools for api testing you use?**
    - Postman for manual testing
    - Rest Assured library

34. **What are the types of Request in Rest API?**
    - There are Get, Post, Put, and Delete requests
        - Get read data
        - Post creates data
        - Put updates data
        - Delete deletes data

35. **What are headers in REST API?**
    - I am using Accept.(Content Type.JSON) type - checks what I am receiving should be in JSON or XML format
    - And ContentType.(Contenttype.Json) - checks what i am sending should be in JSON format

36. **What is RestAssured Library?**
    - A non-web service api that's BDD format and helps integrate java code using deserialization and serialization to extract data from the Json and transform it into a java object in order to store, verify, and validate the data to the expected one.

37. **How are you using Enum in your project?**
    - I am using content Type to make sure that my response type is JSON format

38. What is JsonPath?

- Another way to validate response body
- JsonPath j=response.jsonpath;

39. What methods are you using to verify the size of the response data?

- I use Matchers from Hamcrest
  - hasitems()
  - equalTo()

40. How would I use Response interface?

✪ Reporting

- Mvn Verify will run tests even if it fails (it ignores the failure)
  - Waits for all the test to finish
  - It ignores failure b.c we have this in our build configuration
    <testFailureIgnore>true</testFailureIgnore>
  - Verify is a Maven lifecycle that comes after test
- Mvn test will stop running the test if something fails
- The original html report we get is not that great, we need statistical data
- Ex; "html:target/cucumber-report" → Like how many test are pass/failing percentages
- We going to use *Cucumber Sandwich* (this is a dependency file in pom xml) for more statistical data reporting
- In cukesrunner add; "json:target/cucumber.json"
  - It's a html report from a JSON file → How this report works is the Json file is used to generate the report
  - Version 3.15 (from the vid)
  - Add a new build xml in pom (its already in your pom file, the one called TestProject)
    - You will only get this json reporting(cucumber report with graphs and statistics. This report will be only local for YOU to see, not for Jenkins) ONLY if you execute MVN Verify
    - BUT YOU WILL ALWAYS GET A JSON FILE(different from cucumber report) AFTER RUNNING THE TEST, EVEN W.O VERIFY
      - This JSON file is very important for Jenkins - for the cucumber report plugin
- TestProject build:
  - <id>execution</id>
  - <phase>verify</phase> - this is why html (json) report will only generate when using verify
  - <goals>
  - <goal>generate</goal> </goals>
  - The report will also give you a json file
  - To run tests using verify, right click pom file and click on maven build...
    - You can also add parameters (like runner variable and value which is the xml file) - Type in goals: verify
- To run this in the command line
  - Go to location of pom file and type mvn verify
  - Syntax is mvn<lifecycle/goal>
- Order of Execution using mvn verify
  1. Run against the pom file
  2. Pom file runs the xml file
  3. Xml runs the cukesrunner file
  4. Cukesrunner runs the cucumber feature file/test
- Does json cucumber report show screenshot?

41. **How would I write a method where I need parameter with limit of 100 and path parameter of employee id = 100?**
- I would write;
    - .and().params("limit",100)
    - .and().pathParams("employee_id", 110)

42. **What is Backend-API?**
- It is where application logic code is. Your conditions etc.
- How to test?

    1) Manually ⟶ Using tools like Postman etc. By sending requests and verifying responses.

    2) Automation ⟶ Java + RestAssured Library

43. **What are HTTP Request and HTTP Response?**

HTTP request method is made up of four components:
- **Request Method** → Get, Post, Put, Delete (these are the common ones)
- **Request URI** → the URL of the resource
- **Request Header** → Accept-Language, Accept-Encoding, User-Agent, Host
- **Request Body** → this is the data to be sent to the resource

HTTP response method is made up of three components:
- **Response Status Code** → 200, 301, 404, 500 (these are the most common ones)
- **Response Header Fields** → Date, Server, Last-Modified, Content-Type
- **Response Body** → this is the data that comes back to the client from the server