1. **what is Rest Assured?**

- Rest Assured is java library
- it is used to test the web-based application API based on the JSON or XML format
- Rest Assured support different types of HTTP Request as like GET,POST, PUT,PATCH and DELETE.
- Rest Assured provide the domain specific language for writing the test cases in API.

2. **Diff between RequestSpecification and Response interface.**

- RequestSpecification is an interface present inside the Rest Assured.
- It is used to add the different request parameters as like request header, path parameters, query parameter, request body, different types of authorization and we can select different types of HTTP Requests.
- Response is an interface present inside the Rest Assured.
- It is used to capture data from response as like status code, status line, response time, response body and response headers.

3. **How to add assertion in Rest Assured.**

- We add assertion in rest assured with the help of Hamcrest dependency.
- Hamcrest framework is used for Assertion
- Hamcrest framework is used for Matchers Object.
- TestNG is an testing framework and Hamcrest is an Matchers framework.
- TestNG and Hamcrest Framework are used for assertion but Hamcrest does better job than TestNG Framework
- Hamcrest is very useful for validation and data filtering
- Hamcrest defines some methods which can be used for Number Assertion, String Assertion, Collection Object / Array assertion and Map Assertion

1) Numbers Assertion Methods in Hamcrest Library

i. equalTo() method
ii. greaterThan() method
iii. greaterThanAndEqualTo() method
iv. lessThan(0 method
v. lessThanAndEqualTo() method

2) String Assertion Methods in Hamcrest Library

vi. equalTo() method
vii. equalToIgnoringCase() method
viii. equalToIgnoringWhiteSpace() method
ix. containsString() method
x. startsWith() method
xi. endsWith() method
xii. is() method

3) Map Assertion Methods in Hamcrest Library

xiii. **hasKey() method**

- it check extracted response body contains specific key is present or not

xiv. hasValue()method

- it check extracted response body contains specific Value is present or not

xv. **hasEntry() method**

- it check extracted response body contains specific key-value pair is present or not

4) Collection Objects/Arrays Assertion Methods in Hamcrest Library

xvi. hasItem() method

- check specific single value is present or not inside the Collection Object/array

xvii. hasItems() method

- check multiple value is present or not inside the collection object /array

5) Object Methods in Hamcrest

xviii. anyOf() method

- it will check anyone value is present or not
- it works as like OR operator

xix. allOf() method

- it will check all value is present or not
- it works like AND operator

4. what is JSON format?

- JSON stands for JavaScript Object Notation
- JSON format mainly used to store and transfer data from one technology to another technology
- inside the JSON we use data in key-value pair
- JSON only allows key name is String and inside the value we can mention any data type value
- inside the JSON- we separate key-value pair by using colon (:)
- one key have the one value.
- one key have the multiple value in array format.
- one key value the multiple key-value pair

5. Different methods used for pre conditions , for actions and for post conditions

i. **given() method** : it is used to mention pre condition for each and every requests
return type of given method is RequestSpecification interface
we can mention different types of pre-condition as like

1) add request type any header or content Type header
2) add request query parameter
3) add request path parameter
4) add request authorization
5) add request base parameter
6) add request cookies
7) add request body or payload
8) add request logs

ii. **when() method** : it is used to perform actions or event
return type is RequestSpecification interface
here we mention HTTP Request type
1) get() request
2) post() request
3) put() request
4) patch() request
5) delete() request
6) head() request

iii. **then() method** :it is used to mention expected result or outcome
return type of this method is ValidatableResponse interface
here we add different types of assertion point by using Hamcrest dependency
here we mention assertion point for
1) Response status code
2) response status line

3) response time

4) response size

5) response body or payload

6) response headers

7) response cookies

6. HTTP GET Request Script

```
public static void main(String[] args) {
        // step 1; Set the Base URI
        RestAssured.baseURI = "http://localhost:3000/employees/2";

        // step 2: get the Request specification object so we can select HTTP Request
        RequestSpecification httpRequest = RestAssured.given();

        // step 3: select the HTTP GET Request from RequestSpecification object
        Response resp = httpRequest.get();

        // step 4: capture status code from Response object
        System.out.println(resp.getStatusCode());
        System.out.println(resp.statusCode());

        // step 5: capture status line from Response Object
        System.out.println(resp.getStatusLine());
        System.out.println(resp.statusLine());

        // step 6: capture response time from Response Object
        System.out.println(resp.getTime());
        System.out.println(resp.time());

        // step 7: capture respoonse body from Response object
        System.out.println(resp.getBody().asString());
        System.out.println(resp.body().asString());

        // step 8: capture response headers from Response Object
        Headers allheader = resp.getHeaders();

        for (Header abc : allheader) {
                System.out.println(abc.getName() + "  ::  " + abc.getValue());
        }

}
```

7. HTTP DELETE Request Script

```
public static void main(String[] args) {
        // step 1; Set the Base URI
        RestAssured.baseURI = "http://localhost:3000/employees/2";

        // step 2: get the Request specification object so we can select HTTP Request
        RequestSpecification httpRequest = RestAssured.given();

        // step 3: select the HTTP GET Request from RequestSpecification object
        Response resp = httpRequest.delete();

        // step 4: capture status code from Response object
        System.out.println(resp.getStatusCode());
        System.out.println(resp.statusCode());

        // step 5: capture status line from Response Object
        System.out.println(resp.getStatusLine());
        System.out.println(resp.statusLine());

        // step 6: capture response time from Response Object
        System.out.println(resp.getTime());
        System.out.println(resp.time());

        // step 7: capture response headers from Response Object
        Headers allheader = resp.getHeaders();

        for (Header abc : allheader) {
                System.out.println(abc.getName() + "  ::  " + abc.getValue());
        }

}
```

8. HTTP Post Request Scripts

```
public static void main(String[] args) {
        // Request Body Steps
        // step 1: Create Object of JSONObject class
        JSONObject json = new JSONObject();

        // step 2: add test data or Value inside the JSONObject using put method
        json.put("fname", "Rohit");
        json.put("lname", "Sathe");
        json.put("mobileNo", "90909090");
        json.put("address", "Pune");
        json.put("emailId", "rohit@gmail.com");

        // step 3: convert JSONObject into the String object
        String requestBody = json.toString();

        // ------------------------- Post Request Steps ------------------------------
        // step 1 : set the Base URI
        RestAssured.baseURI = "http://localhost:3000/employees";

        // step 2: get the Request Specification object so we can add request
        // header,request body and select the HTTP Request

        RequestSpecification httpRequest = RestAssured.given();

        // step 3: add Content-Type Request Header
        httpRequest.header("Content-Type", "application/json");

        // step 4: add or attach request body to the HTTP POST Request
        httpRequest.body(requestBody);

        // step 5: select HTTP POST Request from request Specification object
        Response resp = httpRequest.post();

        // step 6: capture status code
```

```
                System.out.println(resp.getStatusCode());

                // step 7: capture status line
                System.out.println(resp.getStatusLine());

                // step 8: capture response time
                System.out.println(resp.getTime());

                // step 9: capture response headers
                Headers allheader = resp.getHeaders();
                for (Header header : allheader) {
                        System.out.println(header.getName() + " " + header.getValue());
                }
                // step 10: capture response body
                System.out.println(resp.getBody().asPrettyString());
        }
}
```

9. HTTP PATCH Request

```
public static void main(String[] args) {
        // Request Body Steps
        // step 1: Create Object of JSONObject class
        JSONObject json = new JSONObject();

        // step 2: add test data or Value inside the JSONObject using put method
        json.put("fname", "Rohit");

        // step 3: convert JSONObject into the String object
        String requestBody = json.toString();

        // step 1 : set the Base URI
        RestAssured.baseURI = "http://localhost:3000/employees";

        // step 2: get the Request Specification object so we can add request
        // header,request body and select the HTTP Request

        RequestSpecification httpRequest = RestAssured.given();

        // step 3: add Content-Type Request Header
        httpRequest.header("Content-Type", "application/json");

        // step 4: add or attach request body to the HTTP PATCH Request
        httpRequest.body(requestBody);

        // step 5: select HTTP POST Request from request Specification object
        Response resp = httpRequest.patch();

        // step 6: capture status code
        System.out.println(resp.getStatusCode());

        // step 7: capture status line
        System.out.println(resp.getStatusLine());

        // step 8: capture response time
        System.out.println(resp.getTime());

        // step 9: capture response headers
        Headers allheader = resp.getHeaders();
        for (Header header : allheader) {
                System.out.println(header.getName() + " " + header.getValue());
        }
        // step 10: capture response body
        System.out.println(resp.getBody().asPrettyString());
}}
```

10. HTTP PUT Request

```
public static void main(String[] args) {
        // Request Body Steps
        // step 1: Create Object of JSONObject class
        JSONObject json = new JSONObject();
        // step 2: add test data or Value inside the JSONObject using put method
        json.put("fname", "Anjali");
        json.put("lname", "Sathe");
        json.put("mobileNo", "123456789");
        json.put("address", "Pune");
        json.put("emailId", "anjali@gmail.com");

        // step 3: convert JSONObject into the String object
        String requestBody = json.toString();

        // ------------------------- Put Request Steps ------------------------------
        // step 1 : set the Base URI
        RestAssured.baseURI = "http://localhost:3000/employees/1";

        // step 2: get the Request Specification object so we can add request
        // header,request body and select the HTTP Request

        RequestSpecification httpRequest = RestAssured.given();

        // step 3: add Content-Type Request Header
        httpRequest.header("Content-Type", "application/json");

        // step 4: add or attach request body to the HTTP POST Request
        httpRequest.body(requestBody);

        // step 5: select HTTP POST Request from request Specification object
        Response resp = httpRequest.put();

        // step 6: capture status code
```

```
                System.out.println(resp.getStatusCode());

                // step 7: capture status line
                System.out.println(resp.getStatusLine());

                // step 8: capture response time
                System.out.println(resp.getTime());

                // step 9: capture response headers
                Headers allheader = resp.getHeaders();
                for (Header header : allheader) {
                        System.out.println(header.getName() + " " + header.getValue());
                }
                // step 10: capture response body
                System.out.println(resp.getBody().asPrettyString());
        }
}
```

11. GET Request Final Script

```
public class GetRequest {

	public static void main (String [] args)
	{
		RestAssured.baseURI="http://localhost:3000";
		RestAssured.basePath="/employees";
		RestAssured
		.given()
		.when()
				.get("/1")
		.then()
				.assertThat()
				.statusCode(200)
				.and()
				.statusLine("HTTP/1.1 200 OK")
				.and()
				.contentType(ContentType.JSON)
				.and()
				.body("firstName", Matchers.equalTo("shree"))
				.and()
				.body("address.currentAddress", Matchers.hasKey("fname"))
				.and()
				.body("address.currentAddress",Matchers.hasValue("Pune"))
				.and()
				.body("address.permanentDetails", Matchers.hasEntry("address", "Pune"));
	}

}
```

12. DELETE Request final scripts

```
public class DeleteRequest {

	public static void main (String [] args)
	{
		RestAssured.baseURI="http://localhost:3000";
		RestAssured.basePath="/employees";
		RestAssured
		.given()
		.when()
				.delete("/1")
		.then()
				.assertThat()
				.statusCode(200)
				.and()
				.statusLine("HTTP/1.1 204 No Content")
				.and()
				.contentType(ContentType.JSON);

	}
}
```

13. Post Request final Script

```
public class PostRequest {

	public static void main (String [] args)	{
		EmployeePojo emp= EmployeePojo.builder()
					.id(1)
					.fname("shree")
					.lname("Patil")
					.address("Pune")
					.emailId("shree@gmail.com")
					.mobileNo(9922120304l)
					.build();

		ObjectMapper mapper =new ObjectMapper();
		String reqBody = mapper.writeValueAsString(emp);

		RestAssured.baseURI="http://localhost:3000";
		RestAssured.basePath="/employees";

		RestAssured
		.given()
				.contentType(ContentType.JSON)
				.body(reqBody)
		.when()
				.post()
		.then()
				.assertThat()
				.statusCode(201)
				.and()
				.statusLine("HTTP/1.1 201 Created")
				.and()
				.contentType(ContentType.JSON)
				.and()
				.body("firstName", Matchers.equalTo("shree"))
				.and()
				.body("address.currentAddress", Matchers.hasKey("fname"))
				.and()
				.body("address.currentAddress",Matchers.hasValue("Pune"))
				.and()
				.body("address.permanentDetails", Matchers.hasEntry("address", "Pune"));
	}
}
```

14. PUT Request final Script

```
public class PutRequest {

		public static void main (String [] args)
		{
			EmployeePojo emp= EmployeePojo.builder()
								.fname("Sonali")
								.lname("Bhosale")
								.address("Pune")
								.emailId("sonali@gmail.com")
								.mobileNo(9090909090l)
								.build();
		ObjectMapper mapper =new ObjectMapper();
		String reqBody = mapper.writeValueAsString(emp);


		RestAssured.baseURI="http://localhost:3000";
		RestAssured.basePath="/employees";

		RestAssured
		.given()
			.contentType(ContentType.JSON)
			.body(reqBody)
		.when()
			.post()
		.then()
			.assertThat()
			.statusCode(200)
			.and()
			.statusLine("HTTP/1.1 200 OK")
			.and()
			.contentType(ContentType.JSON)
			.and()
			.body("firstName", Matchers.equalTo("Sonali "))
			.and()
			.body("address.currentAddress", Matchers.hasKey("fname"))
			.and()
			.body("address.currentAddress",Matchers.hasValue("Pune"))
			.and()
			.body("address.permanentDetails", Matchers.hasEntry("address", "Pune"));
	}
}
```

15. PATCH Request Final Scripts

```
public class PatchRequest {

	public static void main (String [] args)
		EmployeePojo emp = EmployeePojo
					.builder()
					.fname("Shital")
					.build();

		ObjectMapper mapper = new ObjectMapper();
		String reqBody = mapper.writerWithDefaultPrettyPrinter().writeValueAsString(emp);

		RestAssured.baseURI = "http://localhost:3000";
		RestAssured.basePath = "/employees";

		RestAssured
		.given()
			.contentType(ContentType.JSON)
			.body(reqBody)
		.when()
			.patch("/1")
		.then()
			.assertThat()
			.statusCode(200)
			.and()
			.statusLine("HTTP/1.1 200 OK")
			.and()
			.contentType(ContentType.JSON)
			.and()
			.body("firstName", Matchers.equalTo("Shital"))
			.and()
			.body("address.currentAddress", Matchers.hasKey("fname"))
			.and()
			.body("address.currentAddress", Matchers.hasValue("Pune"))
			.and()
			.body("address.permanentDetails", Matchers.hasEntry("address", "Pune"));


	}
}
```