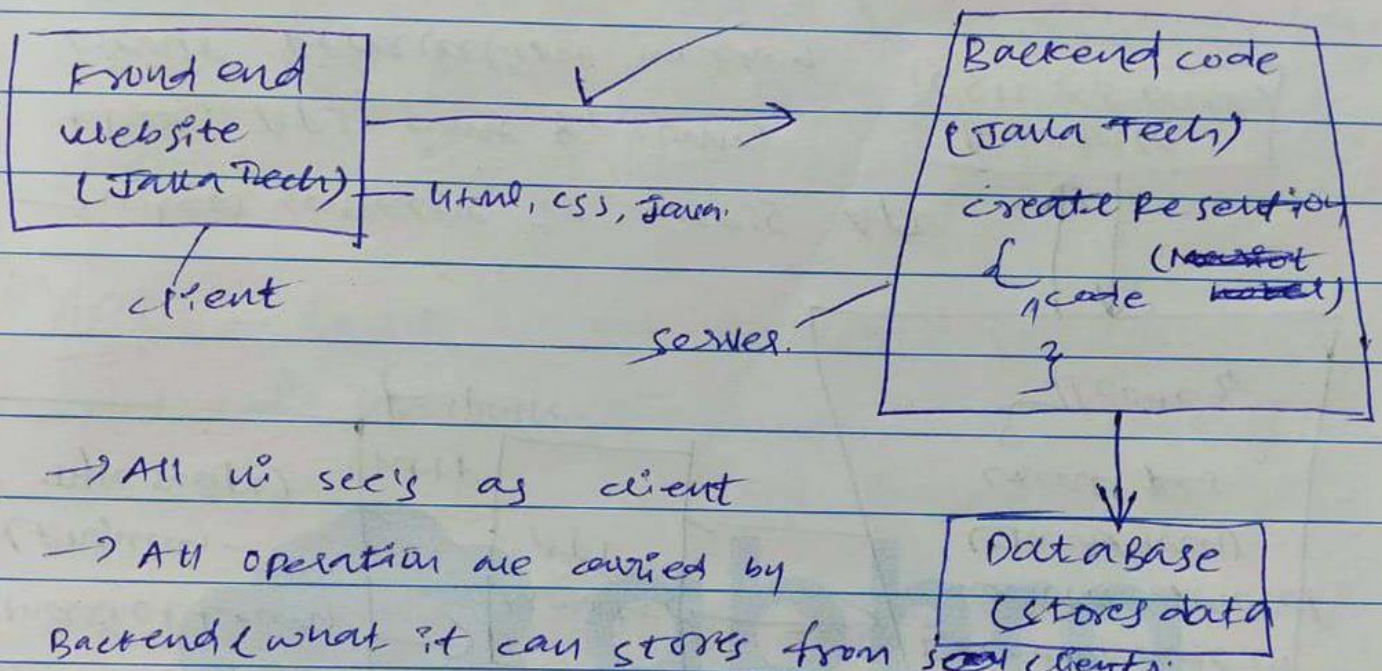
03/03/23

REST - API AUTOMATION

API :- Application programing Interface.

Eg:- Mario hotel in Hotels.com

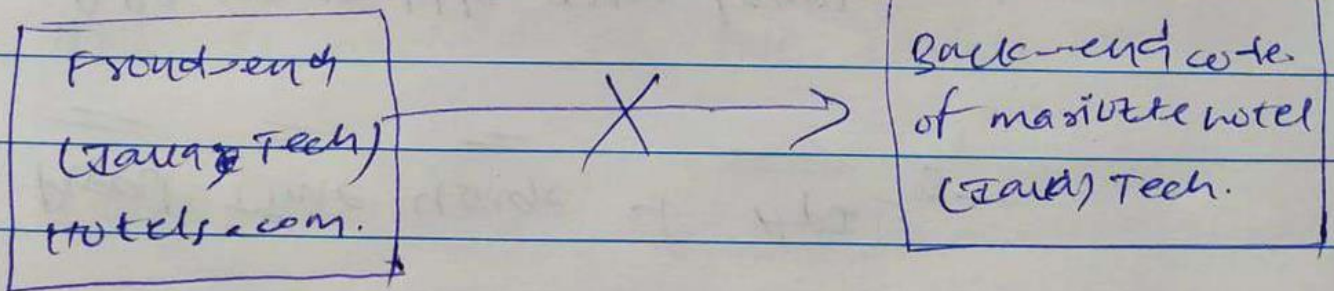
Stand:-1 :- Direct through Mariot website.



→ All ui sees as client

→ All operation are carried by Backend (what it can stores from client).

Stand:-2 :- Hotels.com to mariot website.

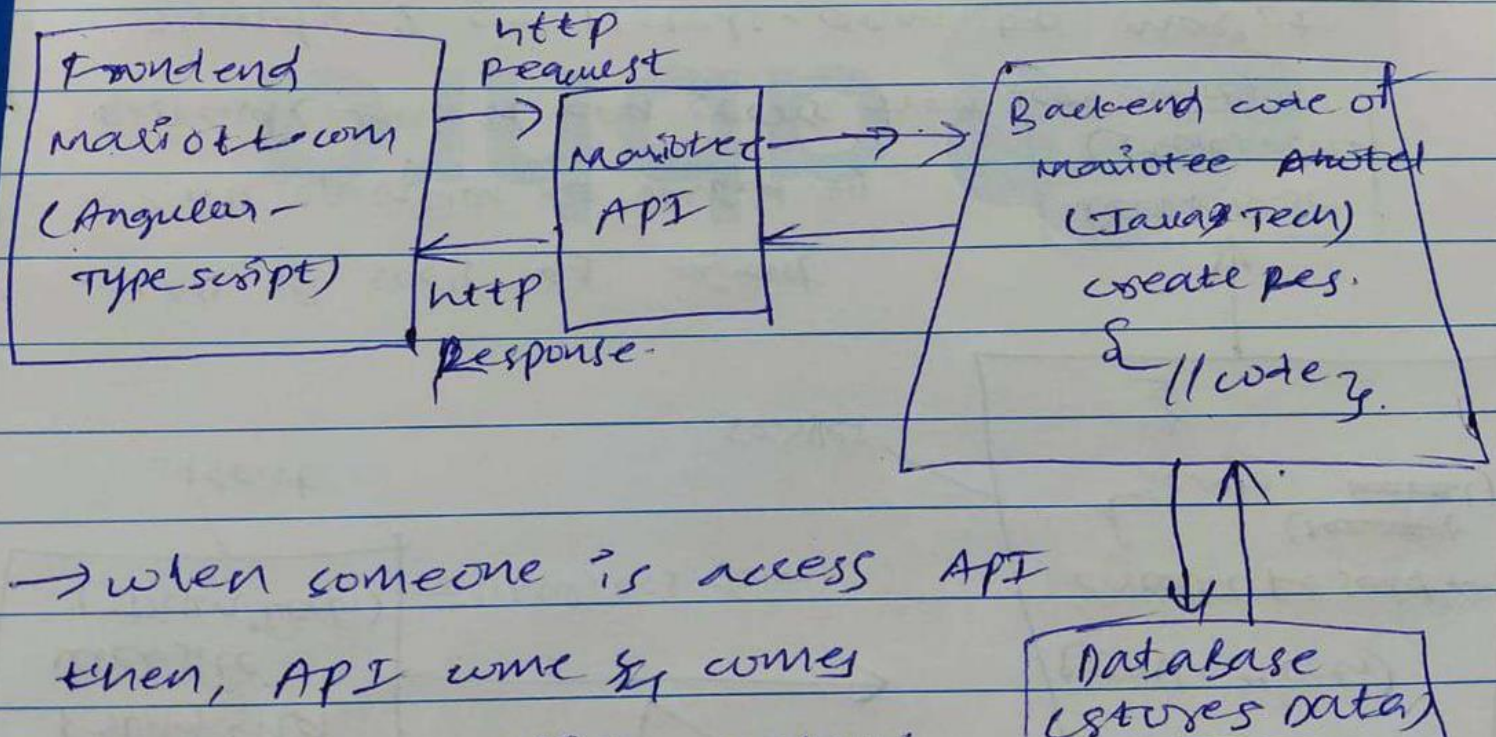


→ It is not accept and agree because mariot hotel is one stack & Hotels.com is another stack.

→ Even though frontend is Angular, Typescript and backend is Java Tech, but there is design, so API is not there.

Real Time usage of API's :-

API :- Middle Type layer.



→ when someone is access API then, API come & comes create reservation method.

→ API is Independent of lang.

Eg :- Front-end capature's all the data and sends to API by using http Protocol

→ Front-end of details & data in the format of Json (or) xml lang to API.

- abhra

→ Then all data API is send's to create Reservation method

→ From Backend Respond will gives to API then API is sends Response through https protocol by using Json (or) xml lang format to Front-end they all the all details is shown over the page.

API :- It is an Interface (or) comunication protocol between client & server intended to simplify the building of client side software

Steps :- 1

1. End point / Base url :- Address where API is hosted on the server.

Http methods which are commonly used to comunicate with rest API's are.

Eg :-

Get, post, PUT and DELETE (CURD operations)

GET :- The GET method is used to extract information from the given server using a given url. While using GET requests, it should only extract data and should have no other effect on the data. NO payload/Body required.

POST :-
A post request is used to send data to the server, (Eg:- customer Info, File upload) using html forms.

PUT :- Replaces all current representations of the target resource with uploaded document

DELETE :- Removes all current representations of the target resource given by a URL.

→ When you send requests to API need some concepts, by using that we can follow.

1. Resources —

Resources represent API/collection which can be accessed from the server.

eg :- google.com/search
google.com/images
google.com/maps.

2. path parameters :—

Path parameters are variable parts of a URL path. They are typically used to point of to a specific resource within a collection, such as a user identified by ID.

eg :- https://www.google.com/images/1234
https://www.google.com/docs/1123/2
https://www.google.com/orders/112

3. Query parameters —

→ Query parameter is used to sort/filter the resource's.

→ Query parameters are Identified with ?

eg :- https://amazon.com/orders?sort_by=21/20/22
Baseurl resource

Note :- Query parameters are always starts '?'

→ '&' symbol is present in query parameters.

04/03/22

End point Request URL can be constructed:-

→ Base URL/resource/(Query (path) parameters.

Headers/Policies :-

(additionaly)
Headers represent the meta-data associated with the API request and response. In layman terms, we were sending Additional details to API to process our Request.
eg :- Authorization details.

Java (Session - 22)

String :-

```
String str = "payment $100 paid";
str.
String str1 = new String("payment $100 paid");
str1.
```

Notes:- String Object can create by using string class.

Reverse a String :- 05/03/22

Reverse a String :- 05/03/22

```
class p.s.v.m c){
String s = "Rana";
String t = "";
for(int i=0;s.length()-1;i>=0;i--)
{
  t=t+s.charAt(i);
}
s.o.p(t);
if(s=t)
{  }
```

Interfaces :-

→ Interfaces does not have the Body. and it only Just signature of method

→ In Interface 100% abstraction should be done.

→ In class can define the Implement the Interface methods.

Eg :- public Interface centralTraffic {

```
public void greenOn()
public void redStop()
public void flashYellow()
}
```

Eg:-

```
Public class AustralianTrafic implements CentralTrafic{
p.s.v.m() {
}
@override
Public void greenGO() {
  // code. }
@override
public void redstop() {
  // code. }
}
```

Session-3

07/03/23

Introduction to postman and google maps :-

→ Search the postman in google

→ Download win 64 bit API.

TO Get some API's from Rahulshetty.com.

→ Search for www.Rahulshetty.com.

→ click practise

→ scroll down page & click API contracts Google maps.

→ Before Going to API Testing, need some API contracts for that specific project.

Create a collection :-

→ Under collection Tab click on (+) create collection.

→ Given project name DR_Maps.

→ click (...) view more actions

→ under that click on Add Request Tab. and give name as Add place

→ Then enter Base url of maps & along with Resource, query parameters & click on send button.

Understand Add place API and execute it through postman.

→ select as post type Introductory.
Enter in url :- for query param (?)
→ enter key & key value. under params.
→ under Body select Raw Type and Text as JSON format.
→ Then paste the format in Body section.
→ check the headers as content-Type in key.
→ click on send & Response in Generated as per API Requests.
→ In Response check 200 ; if generated then it is working fine.

Understand Get & Delete place API using GET, DELETE HTTP methods using postman.

Section - 4.

Rest - Assured setup for API :- 08/03/23

1. Java (JDK)
2. Eclipse editor
3. create Java project & convert into maven project.
4. Add dependencies testng, hamcrest & rest-Assured
5. create a class & write the code for the Add place and to get Response. verify.

Section - 5

Validating Rest API's Responses :-

Class-19 Assertion on Json Response Body and headers through automated code

~~class~~ { package files;
~~public class~~ { public class payload {
public static String Addplace() {
return " {
body
}
}.

12

```
package firstAPI
import io.restassured.RestAssured;
import static io.restassured.RestAssured.*;
import static org.hamcrest.Matchers.*;
import files.payload
public class Basics {
    public static void main(String[] args) {
        RestAssured.baseURI = "https://rahul.";
        given().log().all().queryParam("key", "qaclick123").header("Content-Type", "application/json")
        .body(payload.AddPlace()).when().post("maps/api/place/add/json").then().log().all()
        .assertThat().statusCode(200).body("scope", equalTo("APP")).header("Server", "Apache/2.4.41 (Ubuntu)");
    }
}
```

abhra

Terminologies :-

given() :- Defines for user Inputs.

when() :- submit the API Responses, http methods.

then() :- validate the response.
↳ (from session response)

log all() :- log particular response (or) request ~~Points in console at given to~~ and points in console if have log all. (as body)
then() conditions.

assertThat() :- validate the responses

20). parsing the JSON Response 09/03/23
body using JSonpath class.

JSonpath js = new JSonpath();

→ JSonpath class is the one which takes an string input and convert that into JSon, It will help parse the JSon. There are lot of methods exposed on JSonpath class.

```
RestAssured.baseURI = " ";
String response = given().log().all().queryparam
("key", "qaclick123").header("content-Type", "application/
.body(payload.Addplace()).when().post(" ").
then().assert that().statuscode(200).body("scope",
equalTo("APP")).header("server", "Apache/2.4.18(Ubntu)")
.extract().response().asString();
```

Extract response from the Text & ~~ass~~.

```
S.O.P(response); → print the response in console.
JsonPath js = new JsPath(response); // for passing
                                    Json.
String placeId = js.getString("place_id");
S.O.P(placeId);
```

21). Integrating the multiple APIs with common Json Response values.

```
//Update place.
String newaddress = "Summer walk, Africa";
given().log().all().queryparam("key" "qaclick123")
.header("content-Type", "application/Json").
```

```
body("{\r\n" +
        "\"place_id\":\" "+placeId+"\",\r\n" +
        "\"address\":\"70 summer walka USA\",\r\n"+
        "\"  ("\"+newAddress+"\")
        "}").
when().put("maps/api/place/update/json")
.then().assertThat().log().all().statusCode(200).
body("msg", equalTO("Address successfully updated"));

// Get place :-                    format()
                                   (Inbuilt-Rest Assured)
String getplaceResponse =
given().log().all().queryparam("key","qaclick123")
.queryparam("place_id",placeId).when().get().
get("maps/api/place/get/json").
then().assertThat().log().all().statuscode(200).
extract().response().asString();

JsonPath js1 = new JsonPath(getplaceResponse);
String actualaddress= js1.getString("address");
System.out.println(actualaddress);
```

Printing response format() (From Java)

↳ path of postman.

Note:- Input JSON convert into string as Input of JSON.

12 16

Note:- → For get place in postman tool, we does not have body, under parameter, we are giving query parameter (key and placeId)

→ So same in Rest-API also.

→ Instead Json class object with one Reusable method in files package.

```
public class ReusableMethod {

    public static JsonPath rawToJson(String getplaceresponse){

        JsonPath js = new JsonPath(getplaceresponse);
        return js;
    }
}
```

Section - 6

24). Understanding structure of Nested Json and it's array notations:-

→ create one class in firstAPI package. as class name complexJsonpaise.

```
public class complexJsonpaised
    P.S.V.M()
    Jsonpath js = new Jsonpath(payload.courseprice);
// 25). retriving Json array size and its element using Jsonpath.
// print no. of courses returned by API.
int count = js.getInt("courses.size()");
S.O.P(count);
// print purchase Amount
int totalAmount = js.getInt("dashboard.purchaseAmount");
S.O.P(total amount);
// print title of first course.
String title firstCourse = js.get("course[0].title");
```

→ create one ~~same~~ method in payload class.

```
public static String courseprices(){
return "{
        ----
        Json body.
        };
```

27). Iterating over ~~every element~~ of Json Array & accessing elements in it.

```
//print all course titles and their respective prices
for(int i=0; i<count; i++)
{
String coursetitle = js.get("courses["+i+"].title");
s.o.p(js.get("courses["+i+"].price").toString());
s.o.p(coursetitles);
}
```

28). Retrieving Json nodes on condition logic using Jsonpath.

```
for(int i=0; i<count; i++)
{
String coursetitle=js.get("courses["+i+"].title");
if(coursetitles.equalIgnoreCase("RPA"))
{
int copies = js.get("courses["+i+"].copies");
S.O.P ("Print no. of copies sold by RPA"+copies);
break;
} }
```

29). Real Time examples to solve Business logic.
calculate total sum of courses

```
public class SumValidation {
@Test
public void sumofCourses() {
int sum = 0;
Jsonpath js = new Jsonpath(payload.courseprice());
int count = js.getint("courses.size()");
for(int i=0; i<count; i++) {
int coursePrice = js.getint("courses["+i+"].price");
int copies = js.getint("courses["+i+"].copies");
```

20.

```
int amount = price * copies;
S.O.p(amount);
sum = sum + amount;
}
S.O.p(sum);
int purchaseAmount = js.getInt("dashboard.purchaseAmount");
Assert.assertEquals(sum, purchaseAmount);
} }
```