Boot camp day 2nd on 10/07/2023

**Manual & Automation test cases:**

**Positive scenario:**

**Steps:**

1. Go to google application
2. check search is visible or not
3. Type iphone 13
4. Validated: only iphone 13 visible in drop down

**Test data**: iphone 13

**Expected**: drop down only showed iphone 13

**Negative scenario:**

**Steps:**

1. Go to google application=done
2. check search is visible or not
3. Type iphone 13=done
4. Validated: only iphone 13 should visible in drop down

**Test data**: iphone 13

**Expected**: iphone 13 should visible in drop down

Validation:

1. Search visible or not
2. Iphone 13

Automation test validation: by assertion (java/selenium🡺 part testng/junit)

1. Soft Assert= when test failed🡺 still run all code or test 🡺 Best
2. Hard assert = when test failed 🡺 will stop that line , no go in the code or line

**Boundary scenario:**

**Steps:**

1. Go to google application
2. check search is visible or not
3. Type iphone 13
4. Validated: iphone 12 or 14 should not be visible in drop down

**Test data**: iphone 13

**Expected**: iphone 12 or 14 should not be visible in drop down

==============================================================

How to create manual or Automation test cases

* Positive
  + As many as possible 🡺 to cover acceptance criteria
* Negative
  + As many as possible 🡺 based on user experience
* Optional: Boundary
  + When = if any number or range of numbers
  + How many test cases = at least 3
    - At the level
    - Above the level
    - Below the level

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

Automation:

When do =

1. Manual test at least one time
2. No critical bug open there

Tools & technology:

Application:

* Google 🡺 open with browser = Web based application
* Language:
  + Java
* IDE:
  + Eclipse🡸 use
  + Intellij
* Tool/API =
  + Based on application 🡺Selenium Webdriver
  + Testng = Unit testing framework + Assertion+Report
  + Logger = log4j 🡺 tracking date & time
  + Cucumber = BDD
    - Cucumber testng
    - Cucumber pico container
    - Cucumber java
    - Cucumber maven report
  + Excel = POI
  + DB= JDBC
  + Plugin
    - Java version = maven compiler plugin
    - Report =
      * HTML = Surefire plugin
      * Cucucmber = cucumber maven plugin
* Project🡺 pom.xml🡺 add lib & plugin
  + Add lib
    - Selenium
  + Add plugin
    - Java version 🡺 maven compiler plugin

Test data :

What = any data related to testing

Type of test data:

1. Real data by data team
2. Dummy or fake by me

How to handle test data in automation?

* Configuration data / credentials 🡺 config file (test data in notepad 🡺 read by java properties class)
* Other test data
  + Excel🡺 read by Apache POI
  + DB🡺 JDBC
  + JSON 🡺 JSON parser
* Java handle
  + Single data = variable
  + Multiple data = Array or java collection
* Testng handle
  + @parameter
  + @dataprovider
  + @Factory
* Cucumber:
  + Scenario
  + Scenario outline = multi set

Framework:

* URL 🡺 constants class & java variable
* Test data(iphone 13) 🡺 constants class & java variable
* Selenium locator 🡺 xpath 🡺 Page object model (By method)

List of webElement:

Locator class value

Class =sbct 🡺 6

Class =sbct sbre 🡺 3

Contains method 🡺 //\*[contains(@locator,’value’)] 🡺 //\*[contains(@class,’ sbct’)]

//Wait

Selenium

* Implicit wait
* Explicit wait
* Fluent wait
* Pageload timeOut
* Other
  + TimeUnit.sleep(3000)
  + Selenium.speed

|  |  |  |  |
| --- | --- | --- | --- |
| Implicit | PageLoadTimeOut | Explicit | Fluent |
| Why: wait for HTML DOM page is loaded or not | Why: wait for GUI of the page | Why: wait for one webelement properties – clickable/visible/enable | Why: same as explicit  Extra advantage:   1. Check web element in every 5/10 second 2. Selenium exception = especially stale Element exception |
| Problem: slow down automation |  | Best to use |  |

Java

* Thread.sleep(3000)

How to monitor automation activities:

1. Logger/log
   1. Log4j =1st version of logger
   2. Log4j2 =2nd version of logger
   3. Other logs
      1. Slf4j
   4. Console printing
      1. Print method =syso 🡺 only print text, but not info of date /time/class/method
      2. Logger = provide date & time +text
   5. How to use:
      1. Pom.xml 🡺 add log4j2 lib
      2. Src/main/resource 🡺 add log4j2.xml
      3. Code 🡺 Logger logger = LoggerFactory.getLogger(GoogleSearchFunction.class);
2. Report
   1. Extents report 🡺 need testng/junit
   2. Cucumber maven report 🡺 cucumber

=================================================================

Testng:

Why:

1. Assertion /validation
2. Report -> extents report
3. Lots annotation
   1. BeforeSuite =All test
   2. BeforeClass =all class
   3. BeforeTest = all test
   4. BeforeMethod = each method/test
   5. Test
   6. Test
   7. AfterMethod
   8. AfterTest
   9. AfterClass
   10. AfterSuite
4. Others:

Testng/Junit Framework = known unit testing framework

1. Annotation
2. Assertion
3. Report -extents reporting
4. Can handle multiple test , also can run multiple class or test or method by testng.xml
5. Test.xml advantage
   1. Can run test suite = multiple class/test/method
   2. Handle Parameter 🡺 cross browser testing
   3. Parallel test execution
   4. Help to run specific test group

=================================================================

Which framework currently using : BDD with cucumber

Advantage:

1. Easy to understand
2. Can integrate manual test cases
3. Can integrate testng/junit🡺 testng.xml

Do you use unit testing? 🡺 Yes , I worked with dev and using unit testing in dev environment mostly

Which unit testing framework currently using 🡺 Testng based unit testing framework

Cucumber: BDD framework

Component:

1. Feature file = actual manual test cases
   1. Feature = what to test
   2. Optional:
      1. Description = acceptance criteria
      2. Background = common step
   3. Scenario/Scenario outline
   4. Steps
      1. Given =setup
      2. When =test/action
      3. Then =validation
2. Step def = actual code
3. Runner :
   1. Run the test
   2. Cucumber options

BDD: Runner file(single test)🡺xml file(test suite) 🡺 run from maven (add surefire plugin > pass xml file name)🡺 now ready to run from maven🡺 report

Safe our code 🡺 cloud >> GitHub

How to push code:

1st time:

1. Git init
2. Git add .
3. Git commit -m “ dsadasd”
4. Git remoter add origin repoURL
5. Git push -u origin master

Later:

1. Git add .
2. Git commit -m “ dsadasd”
3. Git push -u origin master

How to pull code:

1. Git clone repoURL

============================================

Jenkins:

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

MINUTE HOUR DOM MONTH DOW

|  |  |
| --- | --- |
| MINUTE | Minutes within the hour (0–59) 🡺 0 |
| HOUR | The hour of the day (0–23)🡺 0 |
| DOM | The day of the month (1–31)🡺 \* |
| MONTH | The month (1–12)🡺 \* |
| DOW | The day of the week (0–7) where 0 and 7 are Sunday. |

🡺 1-5

To specify multiple values for one field, the following operators are available. In the order of precedence,

IA:

Chat GPT 🡺 <https://chat.openai.com/>

Boot camp day 3rd on 10/08/2023

API framework practice

===================================

* QA manager / team manager told to do API testing
* Application
  + Web based 🡺 Selenium
  + No GUI 🡺 API
* Discussed with dev always for detail API testing
* Dev will create API documentation 🡺 known as Swagger template or swagger doc
* Dev will send swagger template link >>**[https://httpbin.org/#/](https://httpbin.org/" \l "/)**
* Do manual 1st >> then automation if needed

**API Manual & Automation test cases:**

**Manual** = Postman

**Automation**= Java & Rest Assured

**CURD Function test:**

**R=read/ get method**

**Steps:**

1. Got to <https://httpbin.org/get>
2. Check connection correctly with status code 200 or not
3. Check if response time less than 2000 ms
4. Header🡺Check test data format is json or not
5. Body
   1. Check body response should not be null
   2. Check attribute origin should be there
   3. Check URL value should contains get only

**D=delete function**

**Steps:**

1. Got to <https://httpbin.org/delete>
2. Check connection correctly with status code 200 or not
3. Check if response time less than 2000 ms
4. Header>>Check test data format is json or not
5. Body>>
   1. Check body response should not be null
   2. Check attribute origin should be there
   3. Check URL value should contains delete only

**C=create/ post method**

**Steps:**

1. Got to <https://httpbin.org/post>
2. Send new test data like below

{ “Event”:”Boot Camp”,

“Participant”:”All Batches”

}

1. Check connection correctly with status code 200 or not
2. Check if response time less than 2000 ms
3. Check test data format is json or not
4. Check body response should not be null
5. Check attribute Event & Participant should be there
6. Check value match as
   1. Event = Boot Camp
   2. Participant = All Batches
7. Check URL value should contain post only

**C=Update/ put method**

**Steps:**

1. Got to <https://httpbin.org/post>
2. Send new test data like below

{ “Event”:”Boot Camp 2023”,

“Participant”:”All Old Students”

}

1. Check connection correctly with status code 200 or not
2. Check if response time less than 2000 ms
3. Check test data format is json or not
4. Check body response should not be null
5. Check attribute Event & Participant should be there
6. Check value match as
   1. Event = Boot Camp 2023
   2. Participant = All Old Students
7. Check URL value should contain post only

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

Theory API :

1. Tell me your current project API testing process and why you need API testing in your current project
2. Tell me the API post-function detail testing process in manual & automation
3. How to handle swagger document in API
4. How to write manual test cases for post
5. How to do Automation cod for post & how to write test cases script for post function
6. Which framework and  report do you use in the API

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

1. What is API?Why do we need API testing?
2. API vs Selenium testing
3. How do u handle API testing in your current project?
   1. Introduction:
   2. Testing
      1. Initial:
         1. Business flow
         2. Application flow
         3. Requirement
            1. Acceptance criteria
         4. Design doc
         5. API document or template
            1. Swagger template or document

Free / open source

Who create: dev

API protocol:

HTTP/HTTPs🡺 most popular

SMTP

FTP

Others

HTTP methods/ CURD functions

C= create (or send new data 🡺HTTP method =post)

U= update (or edit or update old data🡺 HTTP method = put & patch)

R= read (or get data from App/server🡺 HTTP method= get)

D= delete (or remove existing data🡺 HTTP method = delete)

R=read / get method

Try it out> execute

Status code

Connection

Good = 200/201

Bad =4XX or 5XX

URL >>> postman for manual & also automation

1. Status code:
   1. 1XX = informational
   2. 2XX = Success (200 =OK, 201 =created)
      1. Get/put/patch/delete =200
      2. Post =201
   3. 3XX = re-directional
   4. 4XX = failed (client-side error or Application error)
   5. 5XX= failed (server side error)
   6. Failed : post/put/patch/get/delete 🡺4XX/5XX

A white sheet with blue text and numbers

Description automatically generated

API internal process:

request

NY

Google server & data bases

response

Google application

Request & response: parts🡺 TWO parts

* Header = info
* Body = actual dat

Postman:

Request 🡺 header & body , others = parameter, Authentication, Test

Response🡺 header & body, Test result

Request:

* Get /delete = via URL
* Post /put/patch = request body (but they also need URL)

<https://www.google.com/search?q=London&sca_esv=558569235&sxsrf=AB5stBjFL6eoq_JNQ4VNp-pGV6gsFJ_b1A%3A1692545464301&ei=uDHiZJL-Ef-2ptQPw42h8AI&ved=0ahUKEwiSuqmZx-uAAxV_m4kEHcNGCC4Q4dUDCBA&uact=5&oq=London&gs_lp=Egxnd3Mtd2l6LXNlcnAiBkxvbmRvbjIHECMYigUYJzIKEC4YsQMYigUYQzIHEAAYigUYQzIKEAAYigUYsQMYQzIHEAAYigUYQzIHEAAYigUYQzIKEAAYigUYsQMYQzIHEC4YigUYQzIHEAAYigUYQzIHEAAYigUYQ0jqDFAAWM0IcAB4AZABAJgBlAGgAZ0GqgEDMC42uAEDyAEA-AEBwgIREC4YgAQYsQMYgwEYxwEY0QPCAgsQLhiKBRixAxiDAcICDhAuGIAEGLEDGMcBGNEDwgIEECMYJ8ICDRAuGIoFGLEDGIMBGEPCAg0QLhiDARixAxiKBRhDwgIIEC4YigUYsQPiAwQYACBBiAYB&sclient=gws-wiz-serp>

Get Vs Post

* Meaning
* How to send request
  + Get🡺 URL
  + Post🡺 body
* get can be bookmark & cache it
* success status code
  + get =200
  + post =201

Post Vs put

* meaning
* if u send 🡺 salary =5000
  + post = 3 times 🡺
    - id=1 & salary =5000
    - Id =2 & salary =5000
    - Id =3 & salary =5000
  + put = 3 times🡺
    - No 🡺 only one time
      * Update salary =8000 where id=1

Put Vs patch

* Meaning
  + Put = update
    - Whole server data
    - Specific area
  + Patch =update
    - Specific area

API type:

SOAP vs REST:

* SOAP
  + OLD
  + Web service testing
  + Tool = soupUI
  + Test data format = XML
  + SOAP = simple object access protocol
  + Any protocol
  + Performance = slower
  + Security = more secured
* REST
  + New
  + WEB API or API testing
  + Tool 🡺 manual = postman & automation =rest Assured
  + Test data format = JSON
  + REST = representation state transfer architecture
  + Only HTTP protocol
  + Performance = faster
  + Secured = less secured

URL vs URI

L = location

I =identity

Manual testing:

1. Tool = POSTman
   1. Project =workspace
      1. CURD function
         1. Create
         2. Update
         3. Read
            1. Swagger template

URL

Send

Status code

200🡺 write manual test cases

* + - 1. Delete

Postman top section: known as request🡺 Test = here need to write manual test cases

Common API test cases/scenario:

1. Status code
2. Response time
3. Header 🡺Data format or header contain type= Json or not
4. Body🡺
   1. Should not null
   2. Check specific attribute there or not
   3. Check attribute value

How to write manual test cases in postman:

* Top section >test
* Language: JS
* Test data format: JSON
* Pm.test(“test name”, function(){

Code= based my requirement

})

Get test cases examples:

pm.**test**("Check status is  200 or not", **function** () {

    pm.response.to.have.status(200);

});

pm.**test**("Response time is less than 2000ms", **function** () {

    pm.expect(pm.response.responseTime).to.be.below(2000);

});

pm.**test**("Content-Type is json or not", **function** () {

    pm.response.to.have.header("Content-Type");

});

pm.**test**("Body should not null", **function** () {

    pm.expect(pm.response.text()).to.be.not.empty;

});

pm.**test**("Body should contain attribute name = origin", **function** () {

    pm.expect(pm.response.text()).to.include("origin");

});

pm.**test**("Attribute URL value", **function** () {

    var jsonData **=** pm.response.json();

    pm.expect(jsonData.url).to.eql("https://httpbin.org/get");

});

pm.**test**("Attribute Host value", **function** () {

    var jsonData **=** pm.response.json();

    pm.expect(jsonData.headers.Host).to.eql("httpbin.org");

});

delete test cases examples:

pm.**test**("Check status is  200 or not", **function** () {

    pm.response.to.have.status(200);

});

pm.**test**("Response time is less than 2000ms", **function** () {

    pm.expect(pm.response.responseTime).to.be.below(2000);

});

pm.**test**("Content-Type is json or not", **function** () {

    pm.response.to.have.header("Content-Type");

});

pm.**test**("Body should  null", **function** () {

    pm.expect(pm.response.text()).to.be.empty;

});

pm.**test**("Body should contain attribute name = origin", **function** () {

    pm.expect(pm.response.text()).to.be.not.include("origin");

});

pm.**test**("Attribute URL value", **function** () {

    var jsonData **=** pm.response.json();

    pm.expect(jsonData.url).to.be.not.eql("https://httpbin.org/get");

});

pm.**test**("Attribute Host value", **function** () {

    var jsonData **=** pm.response.json();

    pm.expect(jsonData.headers.Host).to.be.not.eql("httpbin.org");

});

Post or put:

Add test data in body>raw>json  
  
Common API testing challenges:

* Status code =400/500
* Response time= got 3 second 🡺 too slow
* Contain type = json=start & end with {} 🡺 xml start & end <>
* Body
  + Should not null 🡺 came null
  + Attribute present or not = name
  + Attribute value name = sarower , but found name= Sobhan
* Authentication:
  + Basic auth
    - Name
    - Pass
  + Bearer auth = token

Market process:

1. **Resume – Me or your teacher**
2. **Interview support = Me or your teacher**
3. **Job support = Me or your teacher**
4. IT job types
   1. Fulltime
      1. Salary less
      2. Benefit package – 401K, health insurance, paid vaccation
   2. Consultant /contract
      1. Salary more
      2. No benefit package
5. IT work type
   1. From home =remote
   2. From office = on site or in person
   3. Home & office = Hybrid
6. Payment/rate
   1. Hourly = 40 to 50
   2. Yearly = 90k+
   3. W2 = paid 1 tax + 4 tax paid by your salary🡺 house or car or load
   4. 1099= all by your
   5. C2C =
7. Why are you looking for job in my company
8. What is your future goal or where do u want to see yourself after 5 years
9. How to handle conflict with colleagues or dev
10. Do u have any question
11. What kind of job are you looking for
12. Why do u interest in QA job
13. I see your education is non-IT, but how did you get job in IT
14. If they ask, do u have experience in AzurDevOPPs or any unknown tool
15. Theorical Qs
    1. Manual
    2. Java
    3. Selenium
    4. SQL
    5. API -get & post
    6. Others
       1. GitHub
       2. Maven
       3. Jenkins
       4. Cucumber
       5. Testng
16. Coding:
    1. Selenium
       1. xpath
    2. Java
    3. SQL