1. UI : User Interface Automation Review
   1. **Tools we use for UI Automation Testing** 
      1. Selenium Webdriver
         1. How to initialize a browser
            1. Chrome
            2. IE – Edge
            3. Firefox
            4. Headless browser
         2. Lunch website – driver.get() or driver.navigate().to()
         3. Waits
            1. Implicit wait

Global wait: certain amount of time before driver throws exceptions.

* + - * 1. Explicit wait

Applies to certain elements

WebDrvierWait obj to use its methods

* + - * 1. Fluent wait
      1. Find Element syntax and return type and exception it throws
         1. Finds first element
         2. Return type is WebElement
         3. Exceptions
      2. Find Elements syntax and return type and exception it throws
         1. Find elements with similar tag
         2. Return type is List
         3. Return a list of zero index or empty list
      3. Locators
         1. ID – preferable
         2. Name
         3. Classname
         4. linkText
         5. partiallinkText
         6. xpath

absolute

relative

* + - * 1. cssSelector
        2. difference between xpath and CssSelector
      1. Verification points in selenium
         1. Title
         2. Text of element
         3. Is element displayed
         4. Is element present
      2. Select Class
         1. Select by index
         2. Select by value
         3. Select by visible text
      3. Dynamic dropdows : how to automate dynamic dropdowns
         1. Find element
         2. Use findElements to store values of dropdown in a list
         3. Write a loop or iterator
         4. Inside the loop you will write an if condition
         5. And write click method when value of dropdown matches the if condition
      4. Action class
         1. Drag and drop
         2. Scroll down or up
         3. Scroll to element
         4. Right click
      5. Switch method
         1. Handle browser pop ups – alert
         2. Switch between window tabs

Window handle and window handles

* + - * 1. Iframe
      1. sendKeys to upload a file in selenium
      2. .getText method to get the text of element
      3. JavaScriptExecutor
         1. Click on element which is not interactable
         2. Click on element which throws Stale Element Exceptions
         3. Send Keys value
         4. Select calendar value
         5. Scroll to a element
      4. Take Screen Shots
    1. **Java OOPS concepts for programming language** 
       1. Inheritance using Extends keyword
          1. We create a base class and store all properties of

Webdriver

Logger

Reading from properties file

* + - * 1. Encapsulation

Store WebElements in private WebElment

Access them with public methods

* + - * 1. Interface

ITestListeners for Interfaces

WebDriver is an interface

* + - * 1. Abstractions

In POM we use by id, xpath, cssSelector or locators to store WebElements and throw out of coding we don’t see the locators values

* + - * 1. Polymorphism

Method overloading

Implicit wait method where you can overload with Sec. Min, Hours

Method overriding

* + 1. Maven for Build Management Tool
       1. The heart of maven project is Pom.xml
          1. Required dependencies

Selenium < version > - know the version

* + - * 1. Plugins for triggering the build

Maven surefire plugin

Maven compiler plugin

* + - 1. Maven Life cycle
         1. Clean
         2. Install
         3. Test
         4. Build
      2. .m2 folder: local repository folder where it stores jar files downloaded from maven central repository
      3. In companies for security issues they create a private maven repository and provide that url for downloading the dependencies and plugin.
    1. **Behavior Driven Development Framework** 
       1. This FrameWork is build on top of the maven
       2. We are following POM ( Page Object Model) Design pattern where for each tab of our application we have a separate class to store its WebElment and methods. We store elements with private access modifiers and get them with public methods.
       3. We are using pageFactory for implementation of POM. Where we use PageFactory.initElements method to initialize the WebElements which we store them with @FindBy annotations.
          1. Study difference between @FindBy and @By annotations
          2. Study @CacheLookup annotations for interviews
       4. We are using Cucumber for Behavior Driven Development
          1. Feature file

This file has an extension of .feature

Scenario

Scenario Outline – runs same scenario with different set of data using examples

Examples

Gherkin keywords

Given

When

And

Then

Background keyword: write initial repetitive steps under it.

* + - * 1. Step Definitions:

Actual Test case steps using java methods

* + - * 1. TestRunner Class

Runs with CucumberOptions api

Feature

Glue

Tags

Monochrome

dryRun

* + - * 1. BeforeHooks and AfterHooks

They run before and after each scenario

* + - * 1. Cucumber dataTables

Where we pass set of data for test case in feature file

* + - 1. Utilities package
         1. JDCB connectivity

Connection class to establish connection with DB

DriverManager to pass info of DB

Statement class to send query

ResultSet class to store query output

Close the connections

* + - * 1. Webdriver reusable methods
        2. Excel Utility – POI
        3. ScreenShots
      1. Reporting : Cucumber JVM Extent Report
      2. Log4J for logger
    1. **Data Driven Framework** 
       1. This FrameWork is build on top of the maven
       2. We are following POM ( Page Object Model) Design pattern where for each tab of our application we have a separate class to store its WebElment and methods. We store elements with private access modifiers and get them with public methods.
       3. We are using pageFactory for implementation of POM. Where we use PageFactory.initElements method to initialize the WebElements which we store them with @FindBy annotations.
          1. Study difference between @FindBy and @By annotations
          2. Study @CacheLookup annotations for interviews
       4. We are using TestNG framework for Data Driven Test
          1. TestNG annotations
          2. TestNG groups
          3. TestNG Parameters for parameterization
          4. TestNG DataProvider for reading data from Excel
          5. TestNG priorities
          6. TestNG Assertions

Hard assertions : stops execution if assertion failed

Soft Assertions: execution completes and failed assertions will be showed in log or console

* + - * 1. TestNG cross browser Testing (parallel execution)
        2. TestNG Extent Report
        3. TestNG ITestlisteners
        4. Retry failed test cases
        5. TestNG invocation Count
        6. TestNG skip a test case
        7. TestNG Depended methods
      1. Utilities package
         1. JDCB connectivity

Connection class to establish connection with DB

DriverManager to pass info of DB

Statement class to send query

ResultSet class to store query output

Close the connections

* + - * 1. Webdriver reusable methods
        2. Excel Utility – POI
        3. ScreenShots
      1. Reporting
         1. TestNG Extent Report
         2. Index.html ( default reporting of TestNG)
    1. Git and Git Hub for Version control
       1. GitHub is repository where we store our codes as a version control
       2. Git is tool to perform certain activities in repository
          1. Git clone: when we clone project from repository for first time
          2. Git push: when we push modified code from local repository
          3. Git pull: when we pull updated code from repository
          4. Git branch:
          5. Git checkout:
    2. Jenkins for CICD pipeline
       1. There are two types of Jenkins usage in Test Automation
          1. Pipeline
          2. Non-Pipeline