1. **What are limitations in selenium webdriver?**

* Selenium supports testing of only web based applications.
* Mobile testing cannot be done using selenium.
* Captcha and Bar code readers cannot be tested using selenium.
* Reports can only be generated using third party tools like TestNG or Junit.
* Image based Testing is Difficult using Selenium.
* User should have prior programming language.

1. **Installing/configure selenium?**

* Install java JDK and JRE kit.
* Install Eclipse IDE.
* Download Selenium Java Client Driver
* Configure eclipse IDE with selenium webdriver.

1. **What are different ways of locating elements in selenium?**

*Types of locators in selenium IDE:*

* ID
* Name
* Class Name
* Tag Name
* Link Text
* CSS Selector
* DOM
* Xpath

1. **Which is fastest way to identify elements in web page?**

**By.id()** is the fastest way compared to other 7 locators. Finding elements by XPath is useful for finding elements using very complex selectors, and is the most flexible selection strategy, but it has the potential to be very slow, particularly in IE.  it will take time for traversing.

1. **What is absolute path and relative path in xpath?**

*Absolute Xpath:* Complete path from root element to desired element is called Absolute xpath.

*Relative Xpath:* It does not depend on parent nodes. xpath is written by referencing the desired element.

1. **Write code on how to use xpath functions?**

There are different ways to write xpath Syntax:

* //tagname[@attribute=’value’]
* //tagname[@attribute=’value’ and @attribute2=’value’]
* //tagname[text()='value']
* //tagname[contains(text(),’value’)]/following-sibling::tagname
* //tagname[text()=’value’]/../preceding-sibling::td/tagname
* //tagname[text()='value']/ancestor::td/following::td[1]/tagname
* //tagname[text()='value']/parent::td

1. **Write different types of waits or synchronization in selenium webdriver?**

* ImplicitlyWait Command
* ExpectedConditions Command (ExplicitWait)
* FluentWait Command
* PageLoadTimeout Command
* Sleep Command

*Implicit Wait Syntax:*

**driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);**

*Explicit Wait Syntax:*

**WebDriverWait wait = new WebDriverWait(driver, 10);**

**wait.until(ExpectedConditions.elementToBeClickable(By.id("value")));**

*Fluent Wait Syntax:*

**Waitwait=newFluentWait(driver).withTimeout(30,SECONDS).pollingEvery(5,SECONDS).ignoring(NoSuchElementException.class);**

*PageLoadTimeout Syntax:*

**driver.manage().timeouts().pageLoadTimeout(100, SECONDS);**

*Sleep command Syntax:*

**Thread.sleep(4000);**

1. **Implicit wait, explicit wait, thread.sleep, webdriver wait and fluent wait?**

Implicit Wait: When an implicit wait is implemented it tells WebDriver to poll the DOM for a certain amount of time when trying to find an element if it is not immediately available. The default setting is 0. It is set to a certain value before throwing exception and in that time, it will search for an element but if an element shows up before, then it will continue without waiting further.

Explicit Wait: Explicit wait can only be implemented in cases where synchronization is needed and the rest of the script is working fine. It provides a better control when compared with an implicit wait. Unlike an implicit wait, we can write custom code or conditions for wait before proceeding further in the code.

Fluent Wait: When a FluentWait instance is implemented it defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Furthermore, the user may configure the wait to ignore specific types of exceptions while waiting, such as NoSuchElementExceptions when searching for an element on the page.

Thread.sleep Wait: This command stops the execution for certain specified amount of time. During this time there is no other execution that happens in that thread. If the specified element is visible within the time limit even then thread will wait for the specified time before moving to the next statement in the code.

1. **Writing code with expected wait conditions?**

**package** synchronization;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.NoSuchElementException;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.FluentWait;

**import** org.openqa.selenium.support.ui.Wait;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**public** **class** Waits {

**public** **static** **void** main(String[] args) **throws** InterruptedException {

System.*setProperty*("webdriver.chrome.driver", "C:\\Project\\Resource\\chromedriver.exe");

WebDriver driver=**new** ChromeDriver();

//using implicitly wait

driver.manage().timeouts().implicitlyWait(5,TimeUnit.***SECONDS***) ;

driver.get("https://www.google.com/");

System.***out***.println(driver.getTitle());

//using explicitly wait (visibilityOfElementLocated)

WebDriverWait wait=**new** WebDriverWait(driver,5);

wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*("//a[@class='gb\_P']")));

driver.findElement(By.*xpath*("//a[@class='gb\_P']")).click();

//using sleep command

Thread.*sleep*(2000);

System.***out***.println(driver.getTitle());

System.***out***.println(driver.getCurrentUrl());

//using explicitly wait (titleIs)

wait.until(ExpectedConditions.*titleIs*("Gmail - Free Storage and Email from Google"));

//using explicitly wait (elementToBeClickable)

wait.until(ExpectedConditions.*elementToBeClickable*(By.*xpath*("//a[text()='Sign In']")));

driver.findElement(By.*xpath*("//a[text()='Sign In']")).click();

driver.findElement(By.*xpath*("//input[@id='identifierId']")).sendKeys("bru");

//using explicitly wait (textToBePresentInElementValue)

wait.until(ExpectedConditions.*textToBePresentInElementValue*(By.*xpath*("//input[@id='identifierId']"),"bru"));

driver.findElement(By.*xpath*("//span[text()='Next']")).click();

//using fluent wait

Wait<WebDriver> wait1=**new** FluentWait<WebDriver>(driver).withTimeout(15, TimeUnit.***SECONDS***).pollingEvery(5, TimeUnit.***SECONDS***).ignoring(NoSuchElementException.**class**);

wait1.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*("//span[text()='Forgot email?']")));

driver.findElement(By.*xpath*("//span[text()='Forgot email?']")).click();

driver.navigate().back();

driver.navigate().back();

driver.navigate().back();

driver.manage().window().maximize();

driver.findElement(By.*xpath*("//a[text()='Store']")).click();

//using page load timeout

driver.manage().timeouts().pageLoadTimeout(1,TimeUnit.***SECONDS***);

driver.findElement(By.*xpath*("//a[text()='Accessories']")).click();

Thread.*sleep*(4000);

driver.close();

}

}

1. **How to save screen shots using selenium webdriver?**

File src=((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(src,new File("C:\\Users\\sai\\screenshot.png"));

1. **How to handle multiple windows in selenium webdriver?**

String parent=driver.getWindowHandle();

Set<String>s1=driver.getWindowHandles();

Iterator<String> I1= s1.iterator();

while(I1.hasNext())

{

String child\_window=I1.next();

if(!parent.equals(child\_window))

{

driver.switchTo().window(child\_window);

System.out.println(driver.switchTo().window(child\_window).getTitle());

driver.close();

}

}

driver.switchTo().window(parent);

1. **How to launch webpage using chrome driver?**

Download chromedriver file from the website.

Write a program to launch webpage.

Use below two lines in the program.

System.*setProperty*("webdriver.chrome.driver","C:\\Project\\Resource\\chromedriver.exe");

WebDriver driver=**new** ChromeDriver();

1. **What is desired capabilities in selenium webdriver?**

The desired capability is a series of key/value pairs that stores the browser properties like browser name, browser version, the path of the browser driver in the system, etc. to determine the behavior of the browser at run time.

Desired capability can also be used to configure the driver instance of Selenium WebDriver. We can configure driver instance like FirefoxDriver, ChromeDriver, InternetExplorerDriver by using desired capabilities. Desired Capabilities are more useful in cases like:

In mobile application automation, where the browser properties and the device properties can be set. In Selenium grid when we want to run the test cases on a different browser with different operating systems and versions.

* **getVersion() -** public java.lang.String getVersion()
* **setVersion() -** public void setVersion(java.lang.String version)
* **getBrowserName() -** public java.lang.String getBrowserName()
* **setBrowserName()** - public void setBrowserName(java.lang.String browserName)
* **getCapability Method** - public java.lang.Object getCapability(java.lang.String capabilityName)
* **getPlatform()** - public Platform getPlatform()

1. **How to set the browser language while opening website?**

*Using Firefox Browser:*

FirefoxProfile profile = new FirefoxProfile();

//setting the locale french : ‘fr’

profile.setPreference(“intl.accept\_languages”,”fr”);

driver = new FirefoxDriver(profile);

driver.get(“http://www.google.com/”);

*Using Chrome Browser:*

System.setProperty(“webdriver.chrome.driver”,”D:/DollarArchive/chromedriver.exe”);

ChromeOptions options = new ChromeOptions();

options.addArguments(“–lang= sl”);

ChromeDriver driver = new ChromeDriver(options);

driver.get(“http://www.google.com/”);

Unfortunately, it won’t work for IE browser, We need to change it manually.

1. **How to handle windows based popups (upload and dropdown)?**

* Alerts are a small box that appears on the display screen to give you some kind of information or to warn you about a potentially damaging operation or it may even ask you for the permissions for the operation.
* *There are popularly two types of alerts–*
  + Windows-based alert pop ups
  + Web-based alert pop ups
* Prior to the actual scripting, we need to import a package to be able to create a WebDriver script for handling a drop-down and making the Select class accessible.
* WebDriver offers the users with a very efficient way to handle these pop ups using Alert interface.
* *dismiss()* – The *dismiss()* method clicks on the “Cancel” button as soon as the pop up window appears.
* *accept()* – The *accept()* method clicks on the “Ok” button as soon as the pop up window appears.
* *getText()*– The *getText()* method returns the text displayed on the alert box.
* *sendKeys(“”)* – The *sendKeys()* method enters the specified string pattern into the alert box.
* *Handling window based pop-ups* have always been a little tricky as we know Selenium is an automation testing tool which supports only web application testing, that means, it doesn’t support windows based applications and window alert is one of them.

1. **Write code to verify any application login page is working or not?**

**import** **static** org.junit.Assert.\*;

**import** java.net.MalformedURLException;

**import** java.net.UnknownHostException;

**import** org.junit.Before;

**import** org.junit.Test;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.htmlunit.HtmlUnitDriver;

**public** **class** testapp {

**public** **static** **void** main(String[] args) {

//public class LoginPageTest extends IntegrationTest {

System.*setProperty*("webdriver.chrome.driver", "C:\\Project\\Resource\\chromedriver.exe");

WebDriver driver = **new** ChromeDriver(){

**public** HtmlUnitDriver driver;

@Before

**public** **void** setup() **throws** MalformedURLException, UnknownHostException{

driver = **new** HtmlUnitDriver(**true**);

driver.get(System.*getProperty*("login.url"));

}

@Test

**public** **void** testAuthenticationFailureWhenProvidingBadCredentials(){

driver.findElement(By.*id*("username")).sendKeys("fakeuser");

driver.findElement(By.*id*("password")).sendKeys("fakepassword");

driver.findElement(By.*id*("login")).click();

*assertTrue*(driver.getCurrentUrl().endsWith("failed"));

}

@Test

**public** **void** testAuthenticationSuccessWhenProvidingCorrectCredentials(){

driver.findElement(By.*id*("username")).sendKeys("validuser");

driver.findElement(By.*id*("password")).sendKeys("validpassword");

driver.findElement(By.*id*("login")).click();

*assertTrue*(driver.getCurrentUrl().endsWith("/<name\_of\_webapp>/"));

}

};

}}

1. **How to select items from dropdown/select box?**

* Import the select package.

import org.openqa.selenium.support.ui.Select;

* Create an object like

Select sel=new Select(driver.findElement(By.id(“ “)));

* Use the following methods to select the elements from the drop down menu.

sel.selectByIndex(3);

sel.selectByValue("Wednesday");

sel.selectByVisibleText("Saturday");

1. **How to know if checkbox is checked or not in webpage?**

isSelected() is the method used to verify if the web element is selected or not. isSelected() method is pre-dominantly used with radio buttons, dropdowns and checkboxes. It is designed to result a Boolean value with each success and failure.

**System.*out*.println(driver.findElement(By.*xpath*("//input[@class='cbelement']")).isSelected());**

1. **Tell me code to pass values from parent window to child window?**

String winHandleBefore = \_driver.getWindowHandle();

for (String winHandle : \_driver.getWindowHandles())

{

// Switch to child window

driver.switchTo().window(winHandle);

}

// Do some operation on child window and get child window handle.

String winHandleAfter = driver.getWindowHandle();

//switch to child window of 1st child window.

for(String winChildHandle : \_driver.getWindowHandles())

{

// Switch to child window of the 1st child window.

if(!winChildHandle.equals(winHandleBefore)&&!winChildHandle.equals(winHandleAfter))

{

driver.switchTo().window(winChildHandle);

}

}

// to switch to parent window.

driver.switchto.window(winHandleBefore);

1. **Write code to find out if all links are working or not?**

String allLinks[]=selenium.getAllLinks();  
for(int i=0;i<allLinks.length;i++) {  
 selenium.click(allLinks[i]);  
 Thread.sleep(3000);  
 }

1. **Write code on how to use javascript executor?**

JavascriptExecutor js = (JavascriptExecutor)driver;

Js.executeScript(“alert(‘hello’);”);

1. **Difference between assert and verify?**

Assert and Verify commands are used to find whether a given input is present or not on the webpage. Difference between assert and verify:

***Assert:*** When assert command fails test execution **will be stopped**. All the steps after that line of code are skipped and will not be executed. So, to avoid this condition we will use try catch blocks. And in TestNG we will use soft Assert.

In simple, if the assert condition is true then the program will execute the next step. If the condition is false, the program terminates.

This command is mostly used when end result of the check value should pass before moving to the next steps.

***Verify:*** When verify command fails test execution **will continue** by logging the failure. In this case we will move forward even though the end result of check value is failed.

In simple, if the verify condition is true or false, there won’t be any halt in the test execution.

This command is mostly used to check non-critical things.

1. **Difference between driver.close and driver.quit methods?**

close():- Suppose you have opened multiple browser windows with same driver instance, now calling close() on the driver instance will close the current window the driver instance is pointed to. But the driver instance still remain in memory and can be used to handle other open browser windows.

quit():- If you call quit() on the driver instance and there are one or more browser windows open, it will close all the open browser windows and the driver instance is garbage collected i.e. removed from the memory. So now you cannot use this driver instance to do other operation after calling quit() on it. If you do it will throw an Exception.

1. **Common exceptions in selenium?**

* *NoSuchElement:* An element could not be located on the page using the given search parameters.
* *NoSuchFrame:* A request to switch to a frame could not be satisfied because the frame could not be found.
* *StaleElementReference:* An element command failed because the referenced element is no longer attached to the DOM.
* *ElementNotVisible:* An element command could not be completed because the element is not visible on the page.
* *NoSuchWindow:* A request to switch to a different window could not be satisfied because the window could not be found.
* *SessionNotFound:* This Exception occurs when the driver is performing the action after immediately quitting the browser.
* *WebDriverException:* This Exception occurs when the driver is performing the action after immediately closing the browser.

1. **How to handle Ajax calls in selenium?**

AJAX stands for **Asynchronous JavaScript & XML,** and it allows the Web page to retrieve small amounts of data from the server without reloading the entire page. For example, when you click on submit button, JavaScript will make a request to the server, interpret the result and update the current screen without reloading the webpage.  It is a technique used for creating fast and dynamic web pages. This technique is asynchronous and uses a combination of JavaScript and XML. From a tester's point of view, if you are checking the content or the element to be displayed, you need to wait till you get the response. During AJAX call the data is stored in XML format and retrieved from the server.

Handling Ajax call is, knowing the ***loading time for the webpage.*** Loading of the webpage will take within a fraction of seconds. So it is difficult to test using automation tool. For this reason, selenium has to use ***wait method on Ajax call.*** By executing the wait command selenium will suspend execution of current test case and wait for new/expected value. When the new value appears, the suspended test cases will get executed by selenium webdriver.

To test Ajax application, different waits should be applied:

* Thread.sleep
* Implicit Wait
* Explicit Wait
* Webdriver Wait
* Fluent Wait

1. **We have webtable, need to click on second row from table?**

List<WebElement> iRows = driver.findElementByXPath("\\path of the tr in the webtable\\");  
iRowCount = iRows.Size();  
List<WebElement> iColumns = driver.findElementByXPath("\\....tr[1]\td");  
iColCount = iColumns .Size();

For(int i=1;i<=iColCount;i++){

driver.findElementByXPath("\\....tr[2]\td[i]")

}

1. **How to assign the value to textbox other than sendkeys method?**

*We use native java script to enter the value in the text box. Like below:*

WebDriver driver = new FirefoxDriver();

JavascriptExecutor executor = (JavascriptExecutor)driver;

executor.executeScript("document.getElementById("textbox\_id").value='new value';);

1. **Selenium grid, how to execute scripts on multiple browser?**

public static void chosebrowser(String sBrowser)

{

String sAutomationPath="C:\\project\\";

System.out.println("\nStarting Selenium Locally\n");

if (sBrowser.equalsIgnoreCase("Firefox")){

//FirefoxProfile firefoxProfile = new FirefoxProfile();

DesiredCapabilities capabilities = DesiredCapabilities.firefox();

File file=null;

capabilities.setCapability("marionette", true);

file = new File(sAutomationPath+"resources"+File.separator+"geckodriver.exe");

System.setProperty("webdriver.gecko.driver",file.getAbsolutePath());

System.out.println("about to create firefoxdriver");

driver = new FirefoxDriver(capabilities);

}

else if (sBrowser.equalsIgnoreCase("Internet Explorer")) {

DesiredCapabilities caps = DesiredCapabilities.internetExplorer();

caps.setCapability(InternetExplorerDriver.INTRODUCE\_FLAKINESS\_BY\_IGNORING\_SECURITY\_DOMAINS,true);

caps.setCapability("ignoreZoomSetting", true);

File file=null;

caps.setCapability(CapabilityType.UNEXPECTED\_ALERT\_BEHAVIOUR,UnexpectedAlertBehaviour.IGNORE);

file = new File(sAutomationPath+"resources"+File.separator+"IEDriverServer.exe");

System.setProperty("webdriver.ie.driver", file.getAbsolutePath());

driver = new InternetExplorerDriver(caps);

} else if (sBrowser.equalsIgnoreCase("Chrome")){

ChromeOptions options = new ChromeOptions();

options.addArguments("chrome.switches","--disable-extensions")

File file =new File(sAutomationPath+"resources"+File.separator+"chromedriver.exe");

System.setProperty("webdriver.chrome.driver", file.getAbsolutePath());

driver=new ChromeDriver(options);

1. **Write code for drag/drop in selenium?**

*Use Actions class*

Actions action = new Actions(driver);

WebElement source = driver.findElement(By.xpath(“ -- -- -”));

//source element which you want to drag

WebElement target = driver.findElement(By.xpath(“ -- -- -”));

//target where you want to drop

action.dragAndDrop(source,target).perform();

1. **Write code for right click in selenium?**

*Use Actions class*

Actions action = new Actions(driver); // where driver is WebDriver

action.moveToElement(webElement).perform();

action.contextClick().perform();

1. **Write code for scroll to specific element?**

WebElement scrollArea = driver.findElement(By.cssSelector("div.slimScrollBar"));

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("arguments[0].scrollTop = arguments[1];",scrollArea, 250);

1. **How to launch different browsers (IE, safari, firefox)?**

*For chrome:*

System.setProperty("webdriver.chrome.driver", "C:\\Project\\Resource\\chromedriver.exe");

WebDriver driver=new ChromeDriver();

*For ie:*

System.setProperty("webdriver.ie.driver","C:\\Project\\Resource\\internetexplorerdriver.exe");

WebDriver driver=new InternetExplorerDriver();

*For firefox:*

System.setProperty("webdriver.gecko.driver", "C:\\Project\\Resource\\geckodriver.exe");

WebDriver driver=new FirefoxDriver();

1. **Ajax element locator factory selenium?**

AjaxElementLocatorFactory is the factory for the element locators. An element locator that will wait for the specified number of seconds for an element to appear, rather than failing instantly if it's not present. This works by polling the UI on a regular basis. The element returned will be present on the DOM, but may not actually be visible. Because this class polls the interface on a regular basis, it is strongly recommended that users avoid locating elements by XPath. It uses SlowLoadableComponent internally.

1. **How to use autoit to implement downlad/uplaod files?**

public class fileoperation {

public static void main(String[] args) throws IOException {

//Instantiation of driver object. To launch Firefox browser

WebDriver driver = new FirefoxDriver();

//To open URL "http://softwaretestingmaterial.com"

driver.get("http://softwaretestingplace.blogspot.com/2015/10/sample-web-page-to-test.html");

//Locating 'browse' button

WebElement browse =driver.findElement(By.id("uploadfile"));

//pass the path of the file to be uploaded using Sendkeys method browse.sendKeys("D:\\SoftwareTestingMaterial\\UploadFile.txt");

driver.navigateto("http://softwaretestingplace.blogspot.com/2015/10/sample-web-page-to-test.html");

//Download Text File

driver.findElement(By.xpath("//\*[@id='post-body-5864649494765988891']/div[1]/form/div[1]/a[1]")).click();

//To call the AutoIt script

Runtime.getRuntime().exec("D:\\SoftwareTestingMaterial\\AutoIt\\DownloadFile.exe"); //'close' method is used to close the browser window

//driver.close();

}

}

1. **RemoteWebDriver Class?**

RemoteWebDriver is an implementation class of the WebDriver interface that a test script developer can use to execute their test scripts via the RemoteWebDriver server on a remote machine. There are two parts to RemoteWebDriver - a server and a client.

* Server- It is a component that listens on a port for various requests from a RemoteWebDriver client. Once it receives the requests it forwards them to any of the specified drivers. (Firefox, chrome, IE).
* Client- It is used when executing tests locally. It translates the test script requests to JSON payload and sends them across to the RemoteWebDriver server using the JSON wire protocol.

When you execute your tests locally, the WebDriver client libraries talk to your Driver directly. Now, when you try to execute your tests remotely, the WebDriver client libraries talk to the RemoteWebDriver server and the server talks to the drivers, whichever the WebDriver client asks for.

1. **How to use actions class?**

public class MouseHoverExample {

public static void main(String[] args) throws Exception {

// Initialize WebDriver

WebDriver driver = new FirefoxDriver();

// Go to URL

driver.get("http://www.myntra.com/");

// Mouse Over On " Men link "

Actions act = new Actions(driver);

WebElement test = driver.findElement(testlink);

act.moveToElement(test).build().perform();

// Click on "link”

driver.findElement(By.linkText("link")).click();

// Click on the categories

driver.findElement(By.xpath("//\*[text()='Categories']//following::li[1]/label")).click();

// Mouse Hover on the first

Actions sel = new Actions(driver);

sel.moveToElement(driver.findElement(By.xpath("//ul[@class='results small']/li[1]"))).build().perform();

// Click on the "Add to Bag" icon of the 1st bag

driver.findElement(By.xpath("//ul[@class='results small']/li[1]/div[1]//div")).click();

// Hover over the shopping bag icon present on the top navigation bar

Actions mov = new Actions(driver);

mov.moveToElement(driver.findElement(By.xpath("//a[contains(@class,'cart')]//div"))).click().build().perform();

// Click on the remove icon

driver.findElement(By.xpath("(//span[@data-hint='REMOVE '])[1]")).click();

// Closing current driver window

driver.close();

}

}

1. **How to compare what is selected in multiple select box and what is database?**

**Boolean** checkSelenium = seleniumCheckbox.**isSelected**();

**Boolean** checkRestApi = restCheckbox.**isSelected**();

1. **How to click on menu->sub menu items in selenium?**

* If that menu has been created by using select tag then we can use the methods-

selectByValue()

selectByIndex()

selectByVisibleText()

These are the methods of the Select class.

* If the menu has not been created by using the select tag then we can simply find the xpath of that element and click() on it.

1. **What is current automation project flow?**

We need to specify in and out of our Test Automation Framework such as programming language used, Type of framework used, Test Base Class (Initializing WebDriver, Implicit Waits), How we separate Element locators and tests (Page Objects, Page Factory), Utility functions file, Property files, TestNG annotations, How we parameterize tests using Excel files, How we capture error screenshots, Generating reports(Extent Reports), Emailing reports, Version Control System used and Continues Integration Tool used.