POPUPS

With Respect to Selenium popups are catagorized into 6 different types such as

(i) Java Script
(ii) Hidden Division
(iii) File Upload
(iv) File Download
(v) Child Browser
(vi) Window

(i) Java Script Popup:-

Characteristics:-

Java Script Popups are catagorized into ~~two~~ three types they are

(a) Alert Popup
(b) Confirmation Popup
(c) Prompt

⇒ We can't move the popup.
⇒ We can't inspect the popup
⇒ The colour is Black & White.

→ If the popup cantains only 'OK' button then we call it as Alert Popup.
→ If the popup contains 'OK' & 'Cancel' button then we call it as Confirmation popup.
→ We can handle this popup by using the statement

```
driver.switchTo().alert();
```

→ In Alert(I) we have different methods they are

(i) accept();
(ii) dismiss()
(iii) getText()
(iv) sendKeys()

→ accept() is used to click on 'OK' button
→ dismiss() is used to click on Cancel button
→ getText() is used to get the text which is present on the Popup.
→ If the popup is not present and still we try to switching to the popup it will throw NoAlertPresentException.

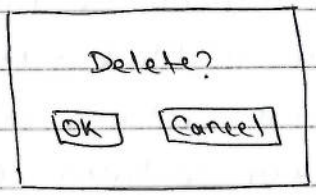
```
e.g: p s v m (s[] args)
 {
    WebDriver driver = new ChromeDriver();
    driver.get("http://irctc.co.in/");
    driver.findElement(By.id("loginbutton")).click();
    Alert a = driver.switchTo().alert();
    // To get the text present on pop up
    String text = a.getText();
    s.o.p(text);
    // To click on OK
    a.accept();
    driver.close();
 }
```

o/p

Enter User ID

Sample Webpage:-

```
<script>
    function c()
    {
        confirm("Delete?");
    }
</script>
<input type="submit" onclick="c()" id="dd" />
```




```
e.g: p s v m (s[] args)
 {
    WebDriver driver = new ChromeDriver();
    driver.get("file:///C:/Users/Admin/Desktop/confirm.html");
    driver.findElement(By.id("dd")).click();
    Alert a = driver.switchTo().alert();
    String text = a.getText();
    s.o.p(text);
    a.dismiss(); → It will click on Cancel Button
    driver.close();
 }
```

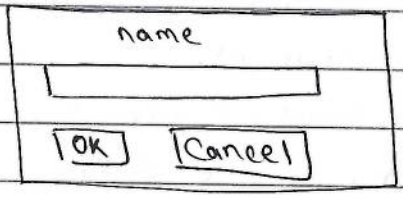
o/p

Delete?

(c) Prompt pop up:-

→ It can be handled by using sendkeys() of Alert().

e.g. Sample Webpage:-




```
<script>
    function p()
    {
      Var name
      name = prompt("name");
      document.getElementById('r').innerHTML = "Welcome" + name;
    }
</script>
<p id="r"></p>
<input type="submit" onclick="p()" id="dd" />
```

eg: p s v m(S[] args)

```
{
    WebDriver driver = new FirefoxDriver();
    driver.get("file:///C:/Users/Admin/Desktop/prompt.html");
    driver.findElement(By.id("dd")).click();
    Alert a = driver.switchTo().alert();
    // To enter the text in prompt
    a.sendKeys("rakesh");
    // To click on OK
    a.accept();
}
```

Assignment:-

WAS for the following scenario

(i) Open the browser and enter the url

(ii) Click on Submit button

(iii) Enter the text in text field & click on OK button

(iv) Verify the Welcome message

(iii) Hidden Division:-

Characteristics:-

⇒ We can't move the pop up.

⇒ We can inspect the popup.

⇒ It is colourfull.

→ We can handle Hidden Division pop up by using findElement().

→ Calender is one of the example for Hidden Division pop up.

eg:

```
p s v m(S[] args)
{
    Date date = new Date();
    SimpleDateFormat s1 = new SimpleDateFormat("d");
    String day = s1.format(date);
    SimpleDateFormat s2 = new SimpleDateFormat("MMMM");
    String month = s2.format(date month);
    WebDriver driver = new FirefoxDriver();
    driver.get("https://www.cleartrip.com");
    driver.findElement(By.id("Depart Date")).click();
    String xp = "//span[.='"+month+"']/../../..//a[.='"+day+"']";
    driver.findElement(By.xpath(xp)).click();
}
```

Assignment:-

1. Login to actiTime appl^n
(ii) Click on Tasks module
(iii) Select the Task
(iv) Click on Delete
(v) Click on cancel

```
p s v m(S[] args) throws InterruptedException
{
    WebDriver driver = new FirefoxDriver();
    driver.get("https://demo.actitime.com/login.do");
```

```
driver.findElement(By.id("username")).sendKeys("admin");
driver.findElement(By.name("pwd")).sendKeys("manager");
driver.findElement(By.xpath("//div[.='Login']").click();
WebElement tasks = driver.findElement(By.xpath("//div[text()='TASKS']"));
Thread.sleep(3000);
tasks.click();
String xp1 = "//div[.='Updating content']/../../..//div[@class='img']";
driver.findElement(By.xpath(xp1)).click();
driver.findElement(By.xpath("//span[.='Delete']")).click();
Thread.sleep(2000);
String xp2 = "//div[@class='cancelBtn greyButton']";
driver.findElement(By.xpath(xp2)).click();
driver.close();
}
```

① p s v m (S[] args) throws InterruptedException

```
{
S.O.p("Enter the Name: ");
Scanner sc = new Scanner(System.in);
String name = sc.nextLine();
WebDriver driver = new FirefoxDriver();
driver.get("file:///D:/HTML/PromptPopUpAss.html");
driver.findElement(By.id("dd").click();
Alert a = driver.switchTo().alert();
a.sendKeys(name);
a.accept();
String aText = driver.findElement(By.id("s")).getText();
S.O.p(aText);
String eText = driver.findElement(By."Welcome "+name;
S.O.p(eText);
Thread.sleep(1000);
```

```
if(aText.equals(eText))
{
}
else
{
    S.O.P("Pass");
    S.O.P("Fail");
driver.close();
}
```

28/02/2018

i) File Upload Pop up:-

⇒ When you click on browse button a pop up will be displayed with the name file upload which is called as File Upload pop-up.

⇒ We can move the pop up.

⇒ We can't inspect the pop up.

⇒ It is colourfull

⇒ It contains two buttons i.e. Open & Cancel.

→ We can handle File Upload popup by using sendKeys() only if the type of the element is file.

→ If the type of the element is other than file like attachment button of gmail then we can handle that element by using Robot class / a third party tool which is called as AutoIT.

NOTE:- In sendKeys() we have to specify the absolute path of the file using double backward slash (\\).

e.g.

```
p s v m(s[] args) throws InterruptedException
{
    WebDriver driver = new FirefoxDriver();
    driver.get("https://www.actitime.com/support.php");
    Thread.sleep(2000);
    String path = "c:\\Users\\Admin\\Desktop\\arg.docx";
    driver.findElement(By.name("screenshot")).sendKeys(path);
}
```

Inspect browse button

type: "file" → pop up
sendKeys() → Absolute path of file
Robot class
AutoIT

Handling File Upload Popup using Robot class:-

```
p s v m (SP) args) throws InterruptedException, AWTException
{
    WebDriver driver = new FirefoxDriver();
    driver.get("https://www.actitime.com/support.php");
    String path = "C:\\Users\\Admin\\Desktop\\avg.docx";
    StringSelection s = new StringSelection(path);
    driver.findElement(By.name("screenshot")).click();
    Toolkit.getDefaultToolkit().getSystemClipboard().setContents(s,null);
    Thread.sleep(2000);
    Robot r = new Robot();
    r.keyPress(KeyEvent.VK_CONTROL);
    r.keyPress(KeyEvent.VK_V);
    Thread.sleep(2000);
    r.keyRelease(KeyEvent.VK_V);
    r.keyPress(KeyEvent.VK_CONTROL);
    Thread.sleep(2000);
    r.keyPress(KeyEvent.VK_ENTER);
    r.keyRelease(KeyEvent.VK_ENTER);
}
```

Assignment

① (i) Navigate to naukri.com
(ii) Click on upload CV
(iii) Browse and select the file
(iv) Click on open
(v) Fill all the mandatory fields
(vi) Click on Register

② (i) Login to facebook
(ii) Click on photo/video
(iii) Browse and select the photo
(iv) Click on open
(v) Specify the caption
(vi) Click on post

5) (i) Login to Actitime

(ii) Click on Settings

(iii) Click on Logo & Color Scheme

(iv) Select use custom logo button

(v) Click on browse

(vi) Browse and select the file & click on open.

(vii) Click on Save settings

```
) p s v m (S[] args)
{
    WebDriver driver = new ChromeDriver();
    driver.get("https://www.naukri.com");
    driver.findElement(By.id("uploadBtnCont").click();
    String Resumepath = "C:\\Users\\sabina\\Desktop\\Resume_28thFeb.docx";
    StringSelection s = new StringSelection(Resumepath);
    Toolkit.getDefaultToolkit().getSystemClipboard().setContents(s,null);
    Robot r = new Robot();
    r.keyPress(KeyEvent.VK_CONTROL);
    r.keyPress(KeyEvent.VK_V);
    Thread.sleep(2000);
    r.keyRelease(KeyEvent.VK_V);
    r.keyRelease(KeyEvent.VK_CONTROL);
    Thread.sleep(2000);
    r.keyPress(KeyEvent.VK_ENTER);
    r.keyRelease(KeyEvent.VK_ENTER);
    driver.findElement(By.name("password").sendkeys("");
    driver.findElement(By.xpath("//button[.='Register']")).click();
    driver.quit();
}
```

Setting → Logo & color scheme →

01/03/2018

(iv) File Download Pop Up:-

Characteristics:-

⇒ We can move the popup.

⇒ We can't inspect the popup.

⇒ It is colourful.

⇒ It contains two radio buttons i.e. Save File & Open With

→ We can handle this pop up by using Robot class.

NOTE:- In ChromeBrowser we will not get the file download popup. It will available only in FirefoxBrowser.

e.g. p s v m(s[] args) throws InterruptedException, AWTException

{

WebDriver driver = new FirefoxDriver();

driver.get("https://www.seleniumhq.org/download/");

String xp = "//td[.='Java']/../../a[.='Download']";

driver.findElement(By.xpath(xp)).click();

Thread.sleep(2000);

Robot r = new Robot();

r.keyPress(KeyEvent.VK_ALT); → To select save file radio button

r.keyPress(KeyEvent.VK_S);

Thread.sleep(2000);

r.keyRelease(KeyEvent.VK_S);

r.keyRelease(KeyEvent.VK_ALT);

Thread.sleep(2000);

r.keyPress(KeyEvent.VK_ENTER);

r.keyRelease(KeyEvent.VK_ENTER);

}

Assignment:-

WAS for the following scenario

(i) Login to actitime appln (iii) Click on Export to CSV (v) Click on Download the file
↳ means click save file button

(ii) Click on Reports (iii) Click on Monthly Team Performance

(v) Child Browser Popup:-

Characteristics:-

⇒ We can move the popup.

⇒ We can inspect the popup.

⇒ It is colourfull.

⇒ It contains minimize, maximize & close button.

→ We can handle this popup by using getWindowHandles method.

→ To switch from one window to another window we use the method

`driver.switchTo().window()`

→ After switching to child windows we can handle that by using findElement() & findElements().

Q:- What is WindowHandle?

→ It is the unique address of the windows.

→ The address of the windows are different from browser to browser.

Q:- What are the difference betn getWindowHandle method & getWindowHandles().

→ getWindowHandle()

- It returns the address of parent browser/window.
- Returntype is String.

getWindowHandles

- It returns the address of all the windows opened by selenium.
- Return type is Set<String>.

e.g. getWindowHandle():-

```
p s v m(S[] args)
{
    WebDriver driver = new ChromeDriver();
    driver.get("https://www.naukri.com/");
    String winHand = driver.getWindowHandle();
    S.O.P(winHand);
}
```

O/P CDwindow-(AFAG7FGB1F7CFADB32BFBFF249BFA27A9)

Address of Driver ... Address will different in different browser & every time it returns different address

getWindowHandler:-

```
p s v m(S[] args)
{
    WebDriver driver = new ChromeDriver();
    driver.get("https://www.naukri.com/");
    Set<String> winHand = driver.getWindowHandles();
    for(String wh:winHand)
    {
        S.O.p(wh);
} }
```

o/p

```
CDwindow-(        )
CDwindow-(        )
CDwindow-(        )
```

eg:WAC to close all the browsers with out using quit().

```
p s v m(S[] args) throws InterruptedException
{
    WebDriver driver = new ChromeDriver();
    driver.get("https://www.naukri.com/");
    Set<String> winHand = driver.getWindowHandles();
    for(String wh:winHand)
    {
        Thread.sleep(1000);
        driver.switchTo().window(wh);   → pass the address
        driver.close();
    }
}
```

① ~~WebDrive~~ p s v m (S[] args)

```
{
    WebDriver driver = new FirefoxDriver();
    driver.get("https://demo.actitime.com/login.do");
    driver.findElement(By.id("username")).sendKeys("admin");
    driver.findElement(By.name("pwd")).sendKeys("manager");
    driver.findElement(By.xpath("//div[.='Login ']").click();
    WebElement reports = driver.findElement(By.xpath("//div[.='REPORTS']"));
```

```
Thread.sleep(2000);
report1.click();
driver.findElement(By.xpath("//div[.='Monthly Team Performance']")).click();
driver.findElement(By.xpath("//td[@class='headerFooterCell clickableCell
                               exportToCsv cellWithBorder']")).click();
Thread.sleep(2000);
Robot r = new Robot();
r.keyPress(KeyEvent.VK_ALT);
r.keyPress(KeyEvent.VK_S);
Thread.sleep(2000);
r.keyRelease(KeyEvent.VK_S);
r.keyRelease(KeyEvent.VK_ALT);
Thread.sleep(2000);
r.keyPress(KeyEvent.VK_ENTER);
r.keyRelease(KeyEvent.VK_ENTER);
} driver.close();
```

e.g. WAS to print title of all the windows. 02/03/2018

```
p s v m (S[] args)
{
    WebDriver driver = new ChromeDriver();
    driver.get("https://www.naukri.com/");
    // get address of all windows
    Set<String> winHand = driver.getWindowHandles();
    // Iterate through all the windows
    for (String wh : winHand)
    {
        Thread.sleep(1000);
        driver.switchTo().window(wh); // switch from one - another window
        String title = driver.getTitle(); // get the title
        S.o.p(title);
    }
```

e.g. WAS to close the specified browser.

```
p s v m(s[] args)
{
    WebDriver driver = new ChromeDriver();
    driver.get("https://www.naukri.com/");
    // get address of all the windows
    Set<String> winHand = driver.getWindowHandles();
    // Iterate through all windows
    for(String wh: winHand)
    {
        // switch from one - another window
        driver.switchTo().window(wh);
        // get the title
        String title = driver.getTitle();
        if(title.equals("Amazon"))
        {
} } } driver.close();
```

e.g. WAS to close all the child browsers.

```
p s v m(s[] args)
{
    WebDriver driver = new ChromeDriver();
    driver.get("https://www.naukri.com/");
    Set<String> winHand = driver.getWindowHandles();
    // get the address of parent window
    String pw = driver.getWindowHandle();
    for(String wh: winHand)
    { driver.switchTo().window(wh);
        if(!wh.equals(pw)) → if window not equal to parent window
        { driver.close();   then close that window
} } }
```

In your class main & static method present
so no need to call that class name.

e.g. WAS to break the control in a specified window. → perform action only on one window

```
p s v m (S[] args)
{
    WebDriver driver = new ChromeDriver();
    driver.get("https://www.naukri.com");
    Set<String> winHand = driver.getWindowHandles();
    for(String wh: winHand)
    {
        driver.switchTo().window(wh);
        String title = driver.getTitle();
        if(title.equals(Amazon))
        {
            break; → Breaking the control in a specified window
    } }
    Thread.sleep(3000);
} driver.manage().window().maximize(); → To know whether the control is present in that window or not
```

Generic Method for above Script/Code:-

```
public static void winHandle(WebDriver driver, String eTitle) → Write this method before main method.
{
    Set<String> winHand = driver.getWindowHandles();
    Iterator<String> itr = winHand.iterator();
    while(itr.hasNext())
    {
        String wh = itr.next();
        driver.switchTo().window(wh);
        if(driver.getTitle().equals(eTitle))
        {
            break;
} } }
```

Assignment:-

Create the generic methods

(i) close the parent Browser

(ii) Close the child window

Handling Tabs:-

→ Multiple tabs can be handled by using getWindowHandles()

e.g. WAP to count no. of Tabs

```
p s v m(S[] args)
{
    WebDriver driver = new ChromeDriver();
    driver.get("https://demo.actitime.com/login.do");
    WebElement link = driver.findElement(By.xpath("//a[text()='actiTIME Inc.']"));
    Actions act = new Actions(driver);
    act.sendKeys(Keys.CONTROL).click(link).perform();   → To open the link in another tab i.e. to perform complete action
    Set<String> allTabs = driver.getWindowHandles();
    int count = allTabs.size();
    S.O.P(count);
}
```

Assignment:-

WAP to (i) print title of allTabs

(ii) to close allTabs

(iii) to switch the control to specified tab

```
p s v m(S[] args)
{
    WebDriver driver = new FirefoxDriver();
    driver.get("https://demo.actitime.com/login.do");
    WebElement link = driver.findElement(By.xpath("//a[.='actiTIME Inc.']"));
    Actions act = new Actions(driver);
    act.sendKeys(Keys.CONTROL).click(link).perform();
    Set<String> allTabs = driver.getWindowHandles();
```

```
for (String tab: allTabs)
{
    Thread.sleep(2000);
    driver.switchTo().window(tab);
    String title = driver.getTitle();
    S.o.p(title);
    driver.close();
}}
```

① p s v m(String args)
p s v winHand(WebDriver driver)

```
{
    Set<String> winHand = driver.getWindowHandles();
    String pw = driver.getWindowHandle();
    Iterator<String> itr = winHand.iterator();
    while(itr.hasNext())
    {
        String wh = itr.next();
        driver.switchTo().window(wh);
        if(wh.equals(pw))
        {
            driver.close();
}}}}
```

② p s v winHand(WebDriver driver)

```
{
    Set<String> winHand = driver.getWindowHandles();
    String pw = driver.getWindowHandle();
    Iterator<String> itr = winHand.iterator();
    while(itr.hasNext())
    {
        String wh = itr.next();
        driver.switchTo().window(wh);
        if(!wh.equals(pw))
        {
            driver.close();
        }
    }
```

07/03/2018

(vi) Window Pop Up:-

→ The pop up which is displaying is not a java script, hidden Division, File upload, File Download, child Browser that pop up is called as Window Pop up.

Characteristics:-

⇒ We can move the popup.

⇒ We can't inspect the popup.

⇒ It is Colourfull.

→ Using selenium we can't handle window popups.

→ To handle the window popup we use a third party tool which is called as AutoIT

→ AutoIT can be downloaded from following website.

URL --> https://www.autoitscript.com/site/autoit/downloads/

FileName --> autoit-v3-setup.exe

→ To install autoit double click on the exe file and follow the default instruction till finish.

AutoIT:-

→ AutoIT is a freely available automation tool which is used to automate the Window based appl^n's.

→ To handle the window popups we can take any automation tools which has following features.

- It should be free.
- It should be able to handle window popups.
- It should be easy to learn and write the script.
- It should be able to integrate with selenium.

Steps to inspect GUI:-

- Go to start → All Programs
- Click on AutoIT
- Click on AutoIT Window Info
- Drag and drop the finder tool on the appl^n/Gui object. which will give the properties such as class, title, name, Instance id etc

Steps to write AutoIT Script:-

- Go to start → All Programe
- Click on AutoIT
- Select scite Script Editor
- Write the AutoIT script
- Save the file in a preferred loc^{n} (.au)
- Compile the autoIT script (Tools → Compile)(control+f7) which will generate the exe file.
- Double click on exe file which will execute the autoIT script.

```
eg. Run("C:\Windows\system32\calc.exe"),
    WinWaitActive("Calculator"),
     Sleep(1000)
     Send("10")
     Sleep(1000)
     Send("-")
     Sleep(1000)
     send("5")
     sleep(1000)
     Send("{ENTER}")
```

↳ Path of the calculator
Go to Properties → Security

- Write script to integrated with Selenium.

```
p s v m(S[] args) throws IOException
{
    Runtime.getRuntime().exec("D:\\HSSM19\\calc.exe");
}
```

↳ Path where exe file present

↳ Runtime r = Runtime.getRuntime();
r.exec("");

eg 1.) AS to download selenium 3.10.0 version.

```
WinWaitActive("Opening selenium-server-standalone-3.10.0.jar")
Sleep(1000)
Send("{LEFT}")
Sleep(1000)     Send("{ENTER}")
```

↳ Title of popup window present on top of the pop up

Anything present between " -- " -- is called Comment

```
p s v m(s[] args) throws InterruptedException, IOException
{
    WebDriver driver = new FirefoxDriver();
    driver.get("https://www.seleniumhq.org/download/");
    String xp = "//a[contains(.,'3.10.0')]";
    driver.findElement(By.xpath(xp)).click();
    Thread.sleep(2000);
    Runtime.getRuntime().exec("D:\\Hslm19\\win.exe");
}
```

Assignment:-

WAS for the following scenario

(i) Open firefox Browser

(ii) Press Ctrl key P

(iii) Click on Print Ok cancel

08/03/2018

Encapsulation:-

→ Binding the data members or restricting the direct access to the variables is called as Encapsulation.

→ We can achieve Encapsulation by using private variables and public methods.

```
pubjc class Demo
{
    Public static int n=10;
    p s v m(s[] args)
    {
        Sample s = new Sample();
        s.getN(10);
        s.o.p(s.getN());
        s.gsetN(100);
        s.o.p(s.getN());
    }
}
```

```
public class Sample
{
    private int n;
    public void setN(int i)
    {
        n=i;
    }
    public int getN()
    {
    } return n;
}
```

e.g.

```
public class Account
{
    //Declaration data member as priate
    private int amt;
    // Initialization by using constructor
    public Account(int n)
    {
    } amt = n;
    // Utilizing by using public method
    public int getAmount()
    {
    } return amt;
}
```

```
public class JavaDemo
{
    p s v m(s[] args)
    {
        Account rakesh = new Account(100);
        s.o.p(rakesh.getAmount());
        Account rakshita = new Account(10000);
        s.o.p(rakshita.getAmount());
        Account samantha = new Account(50000);
        s.o.p(samantha.getAmount());
    }
}
```

e.g.

```
public class LoginPage
{
    // Declaration
    private WebElement unTB;
```

→ unTextBox is single element for this we have to write driver.findElement, return type of this method is WebElement

```
private WebElement pwTB;
private WebElement loginBTN;
//Initialization
public LoginPage(WebDriver driver)
{
    unTB = driver.findElement(By.id("username"));
    pwTB = driver.findElement(By.name("pwd"));
    loginBTN = driver.findElement(By.xpath("//div[.='Login ']"));
}
//Utilization
public void enterUserName(String un),
{
    unTB.sendKeys(un);
}
public void enterPassword(String pw)
{
    pwTB.sendKeys(pw);
}
public void clickOnLoginButton()
{
    loginBTN.click();
}  }
```

→ Create another class with main method & create constructor of login page.

```
Public class Demo
{
p s v m(S[] args)
{
    WebDriver driver = new ChromeDriver();
    driver.get("https://demo.actitime.com/login.do");
    LoginPage lp = new LoginPage(driver);
    lp.enterUserName("admin");
    lp.enterPassword("manager");
} }  lp.clickOnLogin();
```

e.g: (Both -ve & +ve testing on LogmPage of Actitime)

```
p s v m (S[] args)
{
    WebDriver driver = new ChromeDriver();
    driver.get("https://demo.actitime.com/logm.do");
    LogmPage lp = new LogmPage(driver);
    lp.enterUserName("admn");
    lp.enterPassword("damager");
    lp.clickOnLogm();
    Thread.sleep(3000);
    lp.enterUserName("admn");
    lp.enterPassword("manager");
    lp.clickOnLogin();
}
```

→ When we execute the above code it will throw StaleElementReferenceException

Q:- What is StaleElementRefrenceException?

Am Reference of the element is Old.

Q:- When do we get StaleElementReferenceException?

Am After identifying the elements before performing the action if the page is reloaded or refresh the address of the element will be changed in such cases if we try to perform the action on the same element it will throw StaleElement Refrence Exception.

Q:- How do you handle StaleElementReferenceException?

Am By using POM

Q:- What is POM?

→ POM stands for Peage Object model.

→ POM is a java design pattern which is used to design, develope and test the appln.

Q:- How do you declare the elements in POM class?

By using @FindBy (FindBy Annotation)

Syntax:-

@FindBy(Locator = "LocatorValue");

e.g.

```
@FindBy(id = "username")
private WebElement unTB;
```

Q: How do you initialize the elements in POM class?

→ By using initElements method of PageFactory class.

→ initElements is a static method which takes two arguments of type

(a) WebDriver

(b) Object

e.g. PageFactory.initElements(driver, this); → Refers to current obj

→ Here this represents the current class object.

e.g.

```
public class LogmPage
{
    //Declaration
    @FindBy(id="username")
    private WebElement unTB;
    @FindBy(name = "pwd")
    private WebElement pwTB;
    @FindBy(xpath = "//div[.='Logm']")
    private WebElement logmBTN;
    //Initialization
    public LogmPage(WebDriver driver)
    {
        PageFactory.initElements(driver, this);
    }
    //Utilization
    public void enterUserName(String un)
    {
        unTB.sendKeys(un);
    }
```

```
public void enterPassword(String pw)
{
    pwTB.sendKeys(pw);
}
public void clickOnLogin()
{
    loginBTN.click();
}
}
```

09/03/2018

Q:- can you create a POM class without initializing the elements?

→ Yes. But we can't execute it.

→ If we try to execute it will throw NullPointerException.

Q:- Can you initialize the elements in test class?

Yes

```
LoginPage lp = new LoginPage(driver);
PageFactory.initElements(driver, lp);
```
→ pass the reference of the obj

Q:- How do you handle multiple elements in POM class?

By using List<WebElement>

public class LoginPage → Selenium download page

```
{
    // Declaration
    @FindBy(xpath = "//a")
    private List<WebElement> allLinks;
    // Initialization
    public LoginPage(WebDriver driver)
    {
        PageFactory.initElements(driver, this);
    }
    // Utilization
    public void getLinks()
    {
```

LoginPage → POM class — creating obj in same class write this

Demo → Test class — creating obj in another class pass reference

LoginPage lp = new LoginPage();
lp.getLinks();

```
int count = allLinks.size();
S.O.P(count);
for(WebElement link : allLinks)
{
    S.O.P(link.getText());
}
}
```

Q:- Where you store all the elements in your framework?

→ In POM class

→ POM class is also called as Element Repository or Object Repository class.

NOTE:- In selenium we will develope two types of class.

(i) POM

(ii) Test

→ The no. of POM class should be same as no. of webpages.

→ If we have 100 webpages we have to create 100 POM class.

→ The name of the POM class should be same as title of the webpage and it should end with the word Page.

e.g. LoginPage
EnterTime-TrackPage
TaskListPage

→ The no. of test class should be same as no. of Manual TestCases i.e. if we have 1000 manual Test cases we have to create 1000 Test classes.

Advantages of POM Class:-

→ We can handle StaleElementReferenceException.

→ Method name and class name will be Realetic.

→ If any changes happen to the element easily we can identify that element.

Test class → executable class

Writing script → Implementation

Person experience with multiple technology → Architect

TestNG

→ TestNG stands for TestNextGeneration.

→ TestNG is a UnitTestingFramework which is basically used by developers to perform WhiteBoxTesting and also used by Automation Team to execute the framework.

Advantages:-

i) BatchExecution: We can execute multiple test classes at a time/in a single shot.

ii) ParallelExecution: We can execute the framework in multiple browsers parallely.

iii) GroupExecution: We can execute only set up test classes.

iv) Report Generation: Using TestNG automatically we can generate the report.

→ TestNG is avaiable eg as Plugin in Eclipse.

Steps to Install TestNG:-

- In eclipse go to Help
- Click on Eclipse MarketPlace
- Search for TestNG
- Click on Install button for TestNG for Eclipse
- Follow the default instruction till finish. (I accept the terms of the license agreement)
- Restart the eclipse

→ To use the TestNG in the framework we have to add TestNG libraries.

→ To add TestNG libraries

- Right click on the Project (Create a new JavaProject with name TestNGDemo)
- Go to BuildPath
- Click on Add libraries
- Select TestNG
- Click on Next & Finish

Do's & Don'ts in TestNG:-

Don't	Do
(i) Default Package is not allowed.	(i) Use user defined Package.
(ii) main method is not allowed.	(ii) Use Test method.
(iii) S.O.P() is not allowed.	(iii) Use Reporter.log() statement.

↳ o

NOTE:- To print the message on both console and report we use the statement Reporter.log( ) with the boolean value true.

Test Class :-

→ Any java class which contains atleast one test methods is called as Test Class.

→ Any method which is developed using @Test (Test Annotation) is called Test method.

Assignment:-

Create POM class for FacebookLognPage.

e.g. package qsp;

```
import org.testng.Reporter;
import org.testng.annotations.Test;
public class Demo
{
    @Test
    public void testAC()
    {
        Reporter.log("hiiii", false);
    }
}
```

① public class LoginPage

```
{
    @FindBy(id="email")
    private WebElement unTB;
    @FindBy(name="pass")
    private WebElement pwTB;
    @FindBy(id="loginbutton")
    private WebElement loginBTN;
    public LoginPage(WebDriver driver)
    {
    } PageFactory.initElements(driver, this);
```

```
public void enterUserName(String un)
{
    unTB.sendKeys(un);
}
public void enterPassword(String pw)
{
    pwTB.sendKeys(pw);
}
public void clickOnLogin()
{
    loginBTN.click();
}}
p s v m (S[] args)
{
    WebDriver driver = new ChromeDriver();
    driver.get("https://www.facebook.com/");
    LoginPage lp = new LoginPage(driver);
    lp.enterUserName("...");
    lp.enterPassword("...");
    lp.clickOnLogin();
}
```

10/02/2018

→ To run the test class, right click on the test class go to Run as and click on TestNG test.

→ In order to get the report refresh the project which will generate a folder called Test Output.

- Expand test-output folder
- Right click on emailable-report.html
- Go to Open with
- Click on WebBrowser

→ To export the report to excel format - Right click on the Report

→ Allow opening Test'N Report

- Click on Export to Microsoft Excell and follow the default instructions.

NOTE:- To Export the report to excel format MS office should have been installed.

TestNG Suite:-

→ It is an XML file which contains all the test classes present in the framework.

→ To create the TestNG Suite
- Right click on the Project
- Go to TestNG
- Click on Convert to TestNG & follow the default instructions which will create a file called TestNG.xml

e.g.
```
<suite name="Suite">
  <test name="Test">
    <classes>
      <class name="qsp.DemoA"/>
      <class name="qsp.DemoB"/>
    <classes>
  </test>
</suite>
```

→ To run the TestNG suite
- Right click on TestNG.xml file
- Go to Run As
- Click on TestNG Suite

Q:- Can we have multiple test methods in a Single Test Class?

Yes

Q:- If a test class has multiple test methods then what will be the order of execution?

Ans Alphabetical Order

Q:- How do you execute test methods in required order?

Ans By using Priority

NOTE:-

→ When we use priority then the test methods will be executed in the ascending order of priority.

→ Default priority value is 0.

→ If multiple test methods have the same priority then TestNG will be execute in alphabetical order.

→ We can use +ve no.s, -ve no.s and 0 as priority values.

→ If we want to use variables then it must be declared as final.

e.g.
```
public class DemoB
{
    @Test(priority=1)
    public void registerUser()
    {
        Reporter.log("register....", true);
    }

    @Test(priority=2)
    public void deleteUser()
    {
        Reporter.log("delete...", true);
    }
}
```

O/P
```
register....
delete...
```

Q:- How do you execute a test method on success of another test method ?

By using dependsOnmethods

→ When we use dependsOnmethods if the independent method is failed then the dependent method will be skipped.

e.g.
```
public class DemoB
{
    @Test
    public void registerUser()
    {
    } Reporter.log("register...", true);
```

```
@Test(dependsOnMethods = "registerUser")
public void deleteUser()
{
    Reporter.log("delete...", true);
}
}
```

NOTE:-

We can specify the dependency on multiple methods.

e.g. @Test(dependsOnMethods = {"registerUser", "editUser"})

TestNG Annotations:-

12/03/2018

→ TestNG Annotation gives instructions to TestNG Compiler.

→ These annotation decides the execution flows.

→ Types of TestNG Annotations:-

(i) @BeforeMethod (v) @AfterMethod (ix) @DataProvider

(ii) @BeforeClass (vi) @AfterClass

(iii) @BeforeTest (vii) @AfterTest

(iv) @BeforeSuite (viii) @AfterSuite

(i) @BeforeMethod:-

→ BeforeMethod() will be executed before execution of each and every test method()

(ii) @AfterMethod:-

→ AfterMethod() will be executed after execution of each and every test method().

(iii) @BeforeClass:-

→ Before execution of each and every test class before class will be executed.

(iv) @AfterClass:-

→ After execution of each and every test class after class will be executed.

(v) @BeforeTest:-

→ BeforeTest will be executed before execution of each and every test block.

→ Display only no. of test methods

- Test refers to test block of test suite
- Put all the common methods in BaseTest class & inherit all the class from base test class.

(vi) @AfterTest :-

→ AfterTest will be executed after execution of each and every test block.

(vii) @BeforeSuite :-

→ BeforeSuite will be executed before execution of entire suite file.

(viii) @AfterSuite :-

→ AfterSuite will be executed after execution of entire suite file.

e.g:

```
public class BaseTest
{
    @BeforeMethod
    public void beforemethod()
    {
        Reporter.log("before method", true);
    }
    @AfterMethod
    public void aftermethod()
    {
        Reporter.log("after method", true);
    }
    @BeforeClass
    public void beforeClass()
    {
        Reporter.log("before class", true);
    }
    @AfterClass
    public void afterClass()
    {
        Reporter.log("after class", true);
    }
    @BeforeTest
    public void beforeTest()
    {
```

```
        Reporter.log("before test", true);
    }
    @AfterTest
    public void afterTest()
    {
        Reporter.log("after test", true);
    }
    @BeforeSuite
    public void beforeSuite()
    {
        Reporter.log("before suite", true);
    }
    @AfterSuite
    public void afterSuite()
    {
        Reporter.log("after suite", true);
}   }
public class DemoA extends BaseTest
{
    @Test
    public void testA()
    {
        Reporter.log("testA()", true);
}   }
public class DemoB extends BaseTest
{
    @Test
    public void testB()
    {
}   }   Reporter.log("testB()", true);
```

o/p

before suite
before test
before class
before method
testA()
after method
after class
before class
before method
testB()
after method
after class → before method
testC()
after test after method
after suite

```
    @Test
    public void test1()
    {
        Reporter.log("test1()", true);
    }
}
```

Q:- How do you execute same test method multiple times?

Ans:- By using invocationCount

NOTE:- Default invocationCount value is 1.

→ If we use 0 & -ve values then those test methods will not be executed.

e.g.
```
public class DemoA
{
    @Test(invocationCount = 5)
    public void createUser()
    {
        Reporter.log("userA", true);
    }
}
```

(14) @DataProvider:-

→ To run the test method with multiple datas we use dataProvider Annotation.

→ DataProvider method returns the value which is of type two dimensional object array.

e.g.
```
public class DemoA
{
    @DataProvider
    public Object[][] getData()
    {
        Object[][] data = new Object[2][2];
        data[0][0] = "userA";
        data[0][1] = "pass1";
        data[1][0] = "userB";
```

```
        data[1][1] = &Pr1e;
        return data;
    }
    @Test(dataProvider = "getData")
    public void createUser(String un, Object pw)
    {
        Reporter.log(un+" "+pw, true);
    }
}
```

NOTE:- No. of iteration of test methods depends on the no. of Rows.
- No. of arguments of test methods depends on the No. of Columns.

TestNG Groups:- 13/02/2018

→ In order to perform Regional Regression testing we use TestNG Groups.

e.g.

```
public class DemoA
{
    @BeforeMethod(alwaysRun=true)
    public void loginToApplication()
    {
        Reporter.log("Login...", true);
    }
    @AfterMethod(alwaysRun=true)
    public void logoutFromApplication()
    {
        Reporter.log("Logout...", true);
    }
    @Test(priority=1, groups={"user", "smoke"})
    public void createUser()
    {
        Reporter.log("create user...", true);
    }
```

o/p

Login
CreateTask
Logout
Login
CreateUser
Logout

```
@Test(priority=2, groups="user")
public void editUser()
{
  Reporter.log("Edit User...", true);
}
@Test(priority=3, groups="user")
public void deleteUser()
{
  Reporter.log("Delete User...", true);
}
@Test(priority=4, groups={"task","smoke"})
public void createTask()
{
  Reporter.log("Create Task...", true);
}
@Test(priority=5, groups="task")
public void editTask()
{
  Reporter.log("Edit Task...", true);
}
@Test(priority=6, groups="task")
public void deleteTask()
{
  Reporter.log("Delete Task...", true);
}
}
```

XML File:-

```
<suite name="Suite">
<test thread-count="5" name="Test">
<groups>
```

```
        <run>
            <include name = "smoke"/>
            <!--<exclude name = "user"/>-->
        </run>
    </groups>
    <classes>
      < class name = "qsp.DemoA"/>
    </classes>
    </test>
    </suite>
```

Q:- How do you execute the test methods irrespective of specified group?

By using alwaysRun = true

Q:- How do you disable the test method?

By using enabled = false

e.g.

```
public class DemoA
{
    @Test(priority = 1)
    public void createUser()
    {
        Reporter.log("Create User...", true);
    }
    @Test (priority = 2 , enabled = false)
    public void deleteUser()
    {
        Reporter.log ("Delete User...", true);
    }
}
```

NOTE :- In the above example deleteUser() will not be executed.

Enabled default → true

By default alwaysRun = false

Verification in TestNG:-

→ In TestNG we will use Assert class to perform the verification.

→ Assert class contains lot of methods. All the methods are static and overloaded.

Methods of Assert Class:-

(i) AssertEquals ( )
(ii) assertNotEquals ( )
(iii) assertSame ( )
(iv) assertNotSame ( )
(v) assertNull ( )
(vi) assertNotNull ( )
(vii) assertTrue ( )
(viii) assertFalse ( )
(ix) fail ( )

→ Assert class is also called as HardAssert.

e.g.
```
@Test
public void testA()
{
    System.setProperty("webdriver.chrome.driver", "./drivers/chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    driver.get("https://demo.actitime.com/login.do");
    String aTitle = driver.getTitle();
    String eTitle = "actiTIME - Login";
    Assert.assertEquals(aTitle, eTitle);
}
```
↳ static method can be access by using class name.

```
if (aTitle.equals(eTitle))
{
    Reporter.log("Pass", true);
}
else
{
    Reporter.log("Fail", true);
}
```
If you use if cond^n always it will give out put as Pass or fail but it will not change the status so to avoid this, go for assert, always it will show passed only

Q:- Write to verify login button is displayed or not.

```
@Test
public void testA()
{
    System.setProperty("webdriver.chrome.driver", "./drivers/chromedriver.exe");
    WebDriver driver = new ChromeDriver();
```

```
    driver.get("https://demo.actitime.com/login.do");
    WebElement login = driver.findElement(By.xpath("//div[.='Login']"));
    boolean v = login.isDisplayed();
    Assert.assertTrue(v);
}
```

or To print the message present or not present.

```
driver.get("https://demo.actitime.com/login.do");
try
{
    WebElement login = driver.findElement(By.xpath("//div[.='Login']"));
    boolean v = login.isDisplayed();
    Assert.assertTrue(v);
    Reporter.log("Present", true);
}
catch(Exception e)
{
    Reporter.log("Not Present", true);
    Assert.fail(); → status will change to failed otherwise it will show passed.
}
```

Assert.assertEquals(driver.getTitle(), "actiTime - Login");
driver.close(); → it will not execute

NOTE:- In Assert class if the comparison fails it will not execute the remaining statements of current test method. But it will execute remaining test methods.

→ To overcome the above problem we use another type of class which is called as SoftAssert.

Soft Assert:-

→ All the methods which are present in Assert class are also present in SoftAssert class but here it is non static methods.

→ In Soft-Assert class even if the comparison fails it will execute the remain statements of the test method but it will not update the result to the status.

→ In order to update the result to the status we have to use the method AssertAll() and it should be the last statement of the test method.

e.g. @Test

```
public void testA()
{
    System.setProperty("webdriver.chrome.driver", "./drivers/chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    driver.get("https://demo.actitime.com/login.do");
    SoftAssert sa = new SoftAssert();
    sa.assertEquals(driver.getTitle(), "actiTIME-Logi");
    driver.close();
    sa.assertAll(); → If you will not use this line then status will not change to failed for the wrong comparison also, always status will be passed.
}
```

Q:- What are the difference bet^n Assert & Soft-Assert

Assert	SoftAssert
• All the methods are static methods.	• All the methods are non static methods.
• If the comparison fails it will not execute the remaining statements of the current test methods.	• Even if the comparison fails it will execute the remaining statement of the current test method.
• We do n't call assertAll().	• We have to call assertAll() and it should be the last statement.

Q:- When do we use Assert & SoftAssert?

FRAMEWORK

14/03/2018

→ Framework is a standard set of rules, guidelines. It should be followed by all the automation engineers while automating the applications.
→ Framework is also called as Semi Develope Application.
→ We have to use the framework to have the consistency in the project.
→ There are 3 stages in framework.
(i) Automation Framework Design
(ii) Automation Framework Implementation
(iii) Automation Framework Execution

(i) Automation Framework Design:-
→ Automation Framework is mostly design by lead/PM based on the project experience.
→ While designing the automation framework they will set standard rules, guidelines and also they will develope some generic libraries.
→ Initially they will list out all the required softwares and files.
e.g. (a) jdk
(b) eclipse with TestNG
(c) Browsers
(d) AutoIT
(e) Ms Office
(f) Application Under Testing
→ All the above softwares should have been installed in the local work station.
(g) jar files (Selenium, poi)
(h) driver executable files
(i) Excel File
→ All the above file should have been downloaded.

Standard Foldeer Structure:-
→ Go to preferred loc^n and create a workspace.
→ The name of the workspace should be same as domain of the application.
e.g. ProjectManagement
→ Open the eclipse with the above created workspace.

→ Create a new java project.

→ The name of the project should be same as name of the application.

eg. ActiTime

→ Associate TestNG libraries. → Right click on the project → BuildPath → Add libraries → click TestNG → Next → Finish

→ Under source folder create three packages and name it as

eg. com.actitime.generic

com.actitime.pages

com.actitime.tests

→ Under java project create 4 folders and name it as

eg. data

drivers

jars

screenshots

→ Go to the data folder loc^n → data → Right click → Properties → Location → Go to that location → Create the excel file and name it as input.xlsx

→ Store all the driver executable files under drivers folder.

→ Download poi jar file from the following website

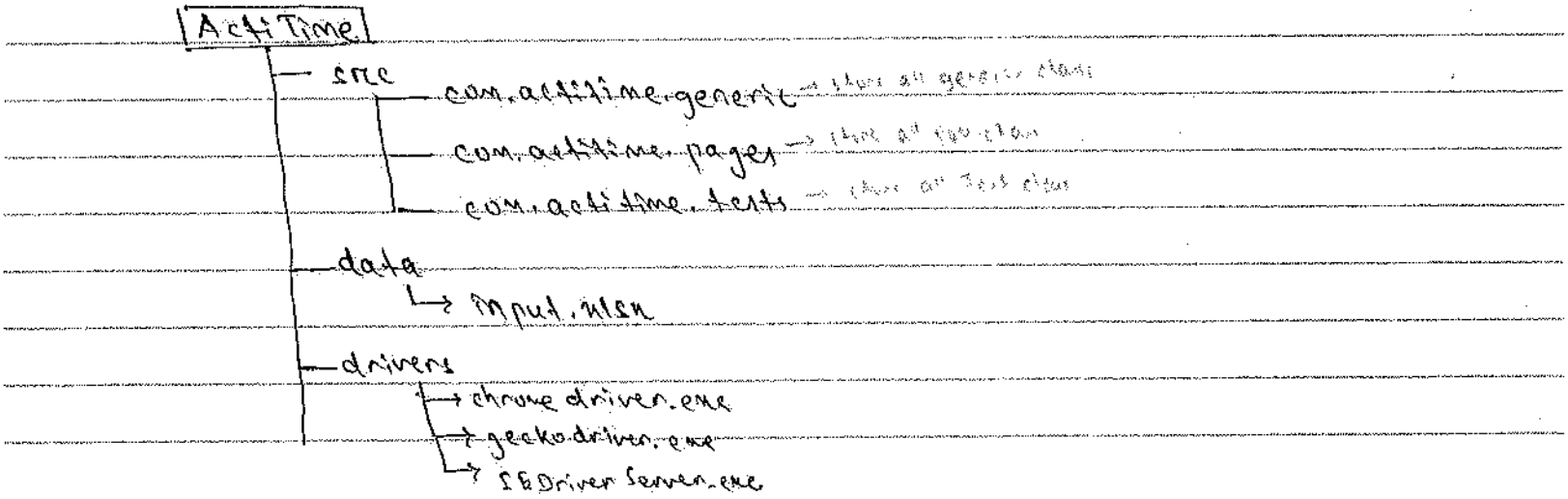
http://poi.apache.org/download

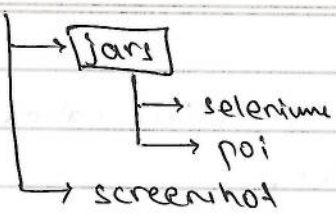
poi-bin-3.17-20170915.zip

→ Extract the poi jar file and copy all the 13 jar files and store it in jars folder.

→ Associate all the jar files with current java project.

Folder Structure :-





Script to get status & name of the Test Method:-

```
public class Demo
{
    @Test
    public void testA()
    {
        Assert.fail();
        Reporter.log("hiii", true);
    }
    @AfterMethod
    public void postCondition (ITestResult res)
    {
        // to get the status if s=1 then pass, if s=2 then fail
        int s = res.getStatus();
        Reporter.log ("status:" + s, true);
        // To get the name of the test method
        String name = res.getMethod().getMethodName();
        Reporter.log(name, true);
    }
}
```

By using this we can get the status

Interface present in TestNG

To get the name of the test method which already executed

15/02/2018

→ Interface and abstract class in Framework:-

→ In our automation framework some of the values such as key and value of the driver executable files, path of the excel files etc. will never change.

→ These values should be declared as constant.

→ According to java standard Interface is the best place to store the constant

Status = 1 → Passed
Status = 2 → Failed

variables.

→ In our automation framework we have created an Interface called AutoConstant which contains all the constant variables.

e.g.
```
public interface AutoConstant
{
    String chrome_key = "webdriver.chrome.driver";
    String chrome_value = "./drivers/chromedriver.exe";
    String gecko_key = "webdriver.gecko.driver";
    String gecko_value = "./drivers/geckodriver.exe";
    String file_path = "./data/input.xlsx";
}
```

→ In our automation framework for every Test case we have created separate test class.

→ In all the test classes there are some common repeatative steps such as
- Open the browser
- Enter the url
- Close the browser

→ Instead of writing same statements in all the test classes we have created a class called BaseTest which contains the common statements of Test classes.

→ In BaseTest class we have created two method i.e. preCondition() and postCondition().

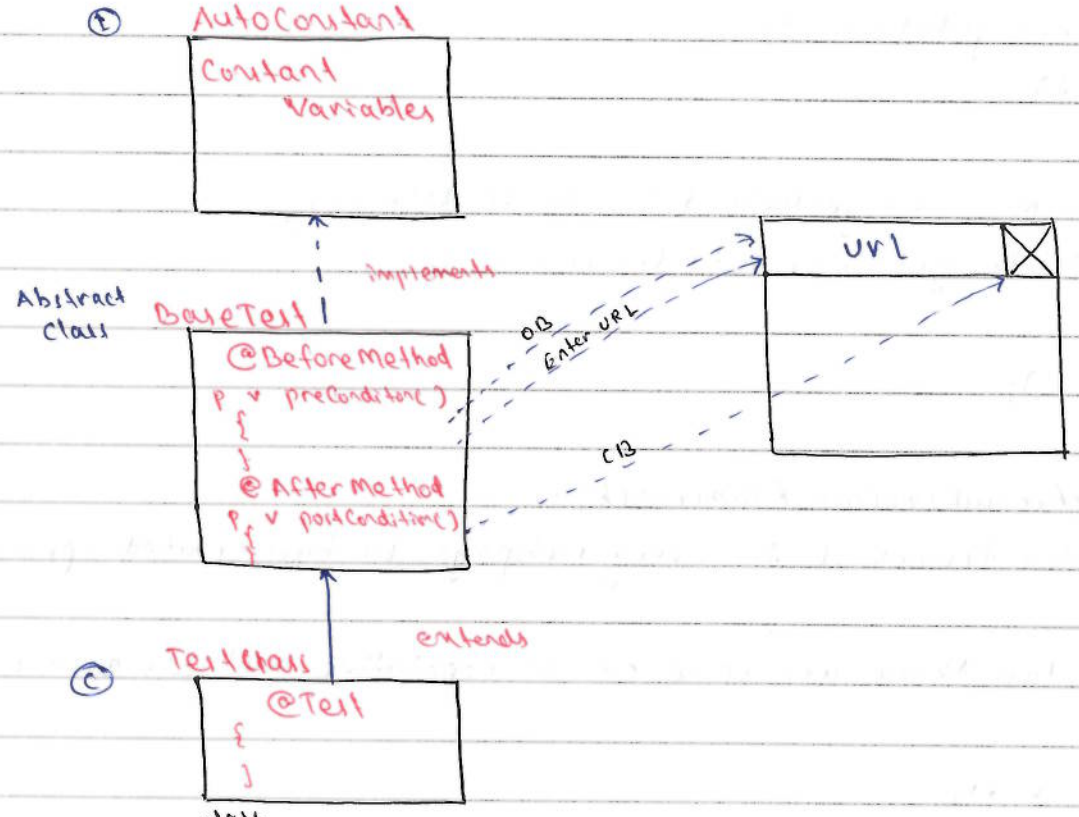
→ In preCondition() we have written the code to open the browser and enter the url, In postCondition() we have written code to close the browser.

→ To avoid the multiple time object creation we have declared preCondition() with @Beforemethod and postCondition() with @Aftermethod.

→ BaseTest class implements AutoConstant Interface.

→ In our automation framework all the test classes extends from BaseTest class.

→ BaseTest class doesn't contain any test method and it is an incomplete class Hence we declare it as Abstract.



e.g. public abstract class BaseTest implements AutoConstant

```
{
    public WebDriver driver;
    @BeforeMethod
    public void preCondition()
    {
        System.setProperty(chrome_key, chrome_value);
        System.setProperty(gecko_key, gecko_value);
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        driver.get("https://demo.actitime.com/login.do");
    }
    @AfterMethod
    public void postCondition(ITestResult res)
    {
```

```
int status = res.getStatus();
if(status==2)
{
    String name = res.getMethod().getMethodName();
    GenericUtils.getScreenshot(driver,name);
}
driver.close();
} }
```

Inheritance in ~~cucu~~ automation Framework :-

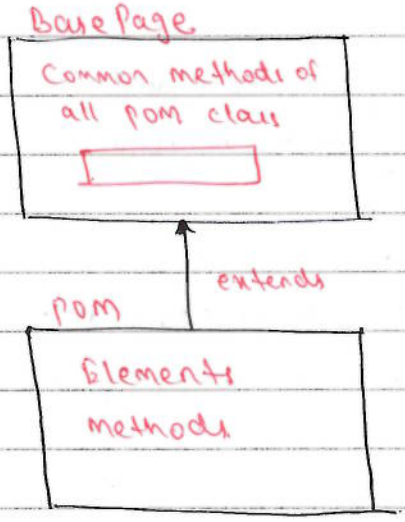
→ In our automation framework for every webpage we have created separate POM class.

→ In all the POM class there are some common repetative methods such as verify the title
- verify the element etc.

→ Instead of writting same methods in all the POM classes we have created a class called BasePage.

→ BasePage class contains the common ~~repet~~ methods of all the POM class.

→ In our automation framework all the POM classes extends from BasePage class.



eg.

```
public class BasePage
{
    public WebDriver driver;
    public BasePage(WebDriver driver)
    {
        this.driver = driver;
    }

    // To verify the title
    public void verifyTitle(String eTitle)
    {
        WebDriverWait wait = new WebDriverWait(driver, 10);
        try
        {
            wait.until(ExpectedConditions.titleIs(eTitle));
            Reporter.log("Title is matching: " + eTitle, true);
        }
        catch(Exception e)
        {
            Reporter.log("Title is not matching: Actual title is: " + driver.getTitle(), true);
            Assert.fail();
        }
    }

    // To verify the Element
    public void verifyElement(WebElement element)
    {
        WebDriverWait wait = new WebDriverWait(driver, 10);
        try
        {
            wait.until(ExpectedConditions.visibilityOf(element));
            Reporter.log("Element is present: " + element.getText(), true);
        }
```

methods present in ExpectedConditions class i.e. Predicates

```
    catch (Exception e)
    {
        Reporter.log("Element is not present: ",true);
        Assert.fail();
} } }
```

Q:- Have you implemented abstraction in your Automation Framework?

Ans Yes

Q:- Have you implemented inheritence in your Automation Framework?

Ans Yes

Generic Utils:-

→ In our automation framework we have created a class called GenericUtils which contains the generic methods to handle listbox, popup, action class etc.

e.g. public class GenericUtils

```
{
    // To take Screenshot
    public static void getScreenshot(WebDriver driver, String name)
    {
        try
        {
            TakesScreenshot t = (TakesScreenshot) driver;
            File src = t.getScreenshotAs(OutputType.FILE);
            File dest = new File("./screenshots/"+name+".png");
            FileUtils.copyFile(src,dest);
        }
        catch (IOException e)
        {
    } }
    // To select by index
    public static void selectByIndex(WebElement element, int index)
    {
```

Instead of hard coding always pass it as an argument.

```
    Select select = new Select(element);
    select.selectByIndex(index);
}
// To select by value
p s v selectByValue(WebElement element, String value)
{
    Select select = new Select(element);
    select.selectByValue(value);
}
// To select by text
p s v selectByText(WebElement element, String text)
{
    Select select = new Select(element);
    select.selectByVisibleText(text);
} }
```

Data Driven Testing :-

16/02/2018

→ Testing the application with multiple datas is called as Data Driven Testing / Parameterization.

→ Manual Testing team will derive the test datas while using Test Case Design Technique such as Error Guessing, Equivalence Partition & Boundary Value Analysis.

→ These test data usually stored in an excel file.

→ Automation team will take the test datas from manual Testing team and they will organise in a proper manner.

→ To take the data from the excel file we use an API which is provided by Apache which is called as POI (Poor Obfuscation Implementation).

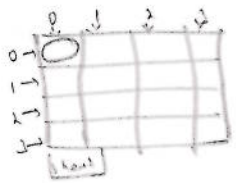
→ POI stands for Poor Obfuscation (Abstraction) Implementation.

Steps to Get data from excel file:-

(1) Create and read the file

(2) Create Workbook

50



(3) Get Sheet

(4) Get Row

(5) Get Cell

(6) Get Data

Eg: WAS to get the data from excel file.

```
p s v m (S[] args) throws Exception
{
    // Create and Read the file
    File file = new File("./data/input.xlsx");
    FileInputStream fis = new FileInputStream(file);
    // Create Workbook
    Workbook wb = WorkbookFactory.create(fis);
    // Get sheet
    Sheet sh = wb.getSheet("sheet1");
    // Get Row
    Row r = sh.getRow(0);
    // Get Cell
    Cell c = r.getCell(0);
    // Get Data
    String data = c.getStringCellValue();
    S.o.p(data);
}
```

e.g. WAS to count no. of Rows present in a sheet.

```
p s v m (S[] args) throws Exception
{
    // Create and Read the file
    File file = new File("./data/input.xlsx");
    FileInputStream fis = new FileInputStream(file);
    // Create Workbook
    Workbook wb = WorkbookFactory.create(fis);
```

Data -> Location

Index value of rows cell -> 0, 1, 2, ...

```
// Get sheet
Sheet sh = wb.getSheet("sheet1");
// Get number of Rows
int rn = sh.getLastRowNum();
s.o.p(rn);
}
```

→ Give the index value of last used Row. It starts from 0 so o/p is 2 bcoz 3 rows used

eg: WAS to count no. of cells present in a Row.

```
p s v m(S[] args) throws Exception{
{
    // Create and Read the file
    File file = new File("./data/input.xlsx");
    FileInputStream fis = new FileInputStream(file);
    // Create Workbook
    Workbook wb = WorkbookFactory.create(fis);
    // Get sheet
    Sheet sh = wb.getSheet("sheet1");
    // Get Rows
    Row r = sh.getRow(0);
    // To count number of cells in a Row
    int cc = r.getLastCellNum();
    s.o.p(cc);
}
```

→ It will give no. of cells in a Row not the index value.

NOTE:-

→ getLastRowNum() returns index value of Last Row.

→ getLastCellNum() returns no. of cells present in a Row.

eg.

A1	B1	C1
A2	B2	C2
A1	B1	C1

WAS to print all the datas in the table format.

```
Ans: p s v m (S[] args) throws Exception
{
    // Create and Read the file
    File file = new File("./data/input.xlsx");
    FileInputStream fis = new FileInputStream(file);
    // Create Workbook
    Workbook wb = WorkbookFactory.create(fis);
    // Get sheet
    Sheet sh = wb.getSheet("Sheet1");

    int rn = sh.getLastRowNum();  -> count no. of Rows, op will be last index value
                                     index value starts from 0
    for(int i=0; i<=rn; i++)
    {                 0   0<=2
        int cc = sh.getRow(i).getLastCellNum();  -> get each row count no. of columns
        for(int j=0; j<cc; j++)
        {                  0; 0<4
            s.o.print(sh.getRow(i).getCell(j).toString()+" ");
                      i                 j   -> to get the data
        }
        s.o.println();
    }
}
```

e.g. Whenever you want to count row no., column no. and get data call these methods by using the class the name below these methods are static methods.

```
public class ExcelData -> write in generic pkg
{
    // To get the data
    public static String getData(String filePath, String sheetName, int rn, int cn)
    {
        try
        {
            FileInputStream fis = new FileInputStream(new File(filepath));
            Workbook wb = WorkbookFactory.create(fis);
```

Add exception to existing catch

cell c = ro.getCell(c);

```
    String data = wb.getSheet(sheetName).getRow(rn).getCell(cn).toString();
    return data;
  }
  catch(Exception e)
  {
      return " ";
  }
}
```

↳
```
Sheet s = wb.getSheet(sheetName);
Row r = s.getRow(rn);
Cell c = r.getCell(cn);
String data = c.toString();
```

→ we have to return in both try & catch block.

```
// To get Row Count
public static int getRowCount(String filePath, String sheetName)
{
    try
    {
        FileInputStream fis = new FileInputStream(new File(filePath));
        Workbook wb = WorkbookFactory.create(fis);
        int rc = wb.getSheet(sheetName).getLastRowNum();
        return rc;
    }
    catch(Exception e)
    {
        return 0;
    }
}
// To get Cell Count
public static int getCellCount(String filePath, String sheetName, int rn)
{
  try
  {
      FileInputStream fis = new FileInputStream(new File(filePath));
      Workbook wb = WorkbookFactory.create(fis);
      int cc = wb.getSheet(sheetName).getRow(rn).getLastCellNum();
  }   return cc;
```

```
    catch(Exception e)
    {
        return "0";   -> return type is int
    }
}
```

TestNG Suite:-

→ In our automation framework we have created TestNG Suite to execute the framework.

→ In the TestNG.XML file instead of specifying individual class names we have specified the pkg which contains all the test classes.

```
<suite name="Suite">
  <test thread-count="5" name="Test">
    <packages>
      <package name="com.actitime.tests"/>
    </packages>
  </test>
</suite>
```

(ii) Automation framework Implementation:- 17/01/2018

→ It is mostly done by the senior Automation Engineers.

→ In this stage we will convert manual Tc into Automation Script.

Q:- What kind of Tc do we automate?

Ans Regression Tc

Q:- Do we automate smoke Tc ?

Ans Yes, If it is a part of Regression Suite

→ In selenium smoke Testing is called as Dry Run.

Q:- On what basis do we select Tc for automation ?

Ans → It should be a part of Regression Suite.

→ It should not contain any manual interventions.

Q:- Can we achieve 100% automation?

Ans NO bcoz

→ Technology will not support.
→ Cost will be high.

Q:- Give me some example for Manual Interventions (Involvement)?

Ans - OTP
- Captch
- Audios / vedios
- Game Testing
- Embedded System (related to H/W)
- Bar code Reader
- Swipe Machines
- Setter Box etc.

Q:- Explain your Automation Framework?
Q:- What is the total time required to execute your framework?
Q:- What is the total duration of your project?
Q:- How many Test Cases are there in your project?
Q:- How many Regression Test Cases are there in your project?
Q:- How many Automatable Regression Test Cases are there in your Project?
Q:- How many scripts are there in your framework?
Q:- How many scripts you have written in your project?
Q:- What is you Automation Team Size?

Test Case 1: valid Login Logout
Step1: Open the browser.
Step2: Enter the URL
verify Login page is displayed or not
Step3: Enter valid username
Step4: Enter valid password.
Step5: Click on Loginbutton
verify Enter Time-Track page is displayed or not.
Step6: Click on Logout button
verify Login page is displayed or not

step 7: Close browser

Test Case 2: Invalid Login

step 1: Open the browser

step 2: Enter the URL

verify Login page is displayed or not

step 3: Enter invalid user name

step 4: Enter invalid password

step 5: click on Login button

verify error message is displayed or not

step 6: close browser

Test Case 3: Verify Version

step 1: Open the browser

step 2: Enter the URL

verify login page is displayed or not

verify the version of actitime

step 3: close browser

Test Case 4: Verify build number

step 1: Open the browser

step 2: Enter the URL

verify Login page is displayed or not

step 3: Enter valid username

step 4: Enter valid password

step 5: Click on login button

verify Enter Time-Track page is displayed or not.

step 6: click on Help

step 7: Click on About your actiTIME

verify the build number.

step 8: click on Logout button

verify login page is displayed or not

step 9: close browser

NOTE :-

Q:- What will be your approch whegn you get the manual TC ?

→ Execute the TC manully to understand the flow of application.

→ While executing we will list out the pages, elements, action on those elements in excel file.

eg page 1 Login

Elements UserName Password Login errMsg version

Action verify the title
Enter UserName
Enter Password
Click on Login
verify error message
verify the version

page2 Enter Time-Track

Elements Logout

Actions verify the Title
click on logout

Rules to develope POM class :-

→ No. of POM class should be same as no. of webpage.

→ The name of the POM class should be same as title of the webpage and it should end with the word Page.

→ For every webpage we have to create separate POM class under com.actitime.page package.

→ All the POM class should extend from BasePage class.

→ In every POM class we should declare the elements using @FindBy. Initialize the elements using constructor, utilize the elements using public methods.

NOTE :- If the super class contains parameterized constructor then we have to call the constructor in sub class also.

```
public class LoginPage extends BasePage -> create this class in pages package
{
    // Declaration
    @FindBy(id = "username")
    private WebElement userNameTB;
    @FindBy(name = "pwd")
    private WebElement passwordTB;
    @FindBy(xpath = "//div[. = 'Login']")
    private WebElement loginBTN;
    @FindBy(xpath = "//span[contains(., 'invalid')]")
    private WebElement errMsg;
    @FindBy(xpath = "//nobr[contains(text(), 'actiTIME')]")
    private WebElement version;
    // Initialization
    public LoginPage(WebDriver driver)
    {
        super(driver);
        PageFactory.initElements(driver, this);
    }
    // Utilization
    public void verifyTheTitle(String eTitle) -> verify the title of all pages
    {
        verifyTitle(eTitle);
    }
    public void enterUserName(String un)
    {
        userNameTB.sendKeys(un);
    }
    public void enterPassword(String pw)
    {
```

```
        passwordTB.sendKeys(pw);
    }
    public void clickOnLogin()
    {
        loginBTN.click();
    }
    public void verifyErrorMessage()
    {
        verifyElement(errMsg);
    }
    public void verifyVersion(String eversion)
    {
        String aversion = version.getText();
        Assert.assertEquals(aversion, eversion);
    }
}
public class EnterTimeTrackPage extends BasePage. → create another class in pages package
{
    // Declaration
    @FindBy(id="logoutLink")
    private WebElement logoutBTN;
    // Initialization
    public EnterTimeTrackPage(WebDriver driver)
    {
        super(driver);
        PageFactory.initElements(driver, this);
    }
    // Utilization
    public void clickOnLogout()
    {
        logoutBTN.click();
    }
}
```

[illegible]

[illegible]

Rules to develope Test Class:-

→ For every manual Test case we have to create the separate test class.

→ For every test case we have to create separate test class under com.actitime.tests package.

→ The name of the test class should be same as testcase name or test case id and it should end with the word test.

→ The name of the test method should start with the word test.

→ In the test method call POM class methods to perform the action.

→ Before calling the POM class method specify the test case step inline (//) comment.

→ If any methods of POM class needs any test datas then those data's should be taken from excel file.

Rules to store the Datas in excel File:-

→ For every webpage we have to create separate sheets and name of the sheet should be same as title of the webpage.

→ In every sheet the first Row should be the header and the data should be inserted from second row.

e.g.:

	0	1	2	3
0	UserName	Password	Title	Version
1	admin	manager	actiTIME - Login	actiTIME - 2017.1
2	qsp	jsp		
3	'1	'2		
4	@	&		

→ Login (Sheet)

→ for integer start with single quote

Title
actiTIME - Enter - Time - Track

→ enter Time Track (Sheet)

```
public class ValidLoginLogoutTest extends BaseTest
{
    @Test
    public void testValidLoginLogout()
    {
        String loginTitle = ExcelData.getData(file_path, "login", 1, 2);
```

[illegible]

```
        String user = ExcelData.getData(file_path,"login",1,0);
        String pass = ExcelData.getData(file_path,"login",1,1);
        String enterTimeTrackTitle = ExcelData.getData(file_path,"enterTimeTrack",1,0
        LoginPage lp = new LoginPage(driver);
        EnterTimeTrackPage ep = new EnterTimeTrackPage(driver);
        // verify login page
        lp.verifyTheTitle(loginTitle);
        // enter valid user name
        lp.enterUserName(user);
        // enter valid password
        lp.enterPassword(pass);
        // click on login
        lp.clickOnLogin();
        // verify enter time track page
        lp.verifyTheTitle(enterTimeTrackTitle);
        // click on logout
        ep.clickOnLogout();
        // verify loginPage
        lp.verifyTheTitle(loginTitle);
    }
}

public class InvalidLoginTest extends BaseTest
{
    @Test
    public void testInvalidLogin() throws InterruptedException
    {
        String loginTitle = ExcelData.getData(file_path,"login",1,2);
        int loginRC = ExcelData.getRowCount(file_path,"login");
        LoginPage lp = new LoginPage(driver);
```

```
// verify login page
lp.verifyTheTitle(logmTitle);
for(int i=2; i<=logmR(); i++)
{                    ↳ Excel data starts from 2

    String user = ExcelData.getData(file_path,"logm", i, 0);
    String pass = ExcelData.getData(file_path,"logm", i, 1);
    Reporter.log("UserName: "+user,true);
    Reporter.log("Password: "+pass,true);
    // enter invalid user name
    lp.enterUserName(user);
    // enter invalid password
    lp.enterPassword(pass),
    // click on Login
    lp.clickOnLogin();
     Thread.sleep(2000);
    // verify error message
    lp.verifyErrorMessage();
} }

public class verifyVersionTest extends BaseTest
{
    @Test
    public void testVersion()
    {
        String logmTitle = ExcelData.getData(file_path,"logm",1,2);
        String version = ExcelData.getData(file_path,"login",1,3);
        LogmPage lp = new LogmPage(driver);
        // verify logmpage
        lp.verifyTheTitle(logmTitle),
        // verify the version
} }     lp.verifyVersion(version);
```

19/03/2018

(iii) Automation Framework Execution:-

→ It is mostly done by the freshers.

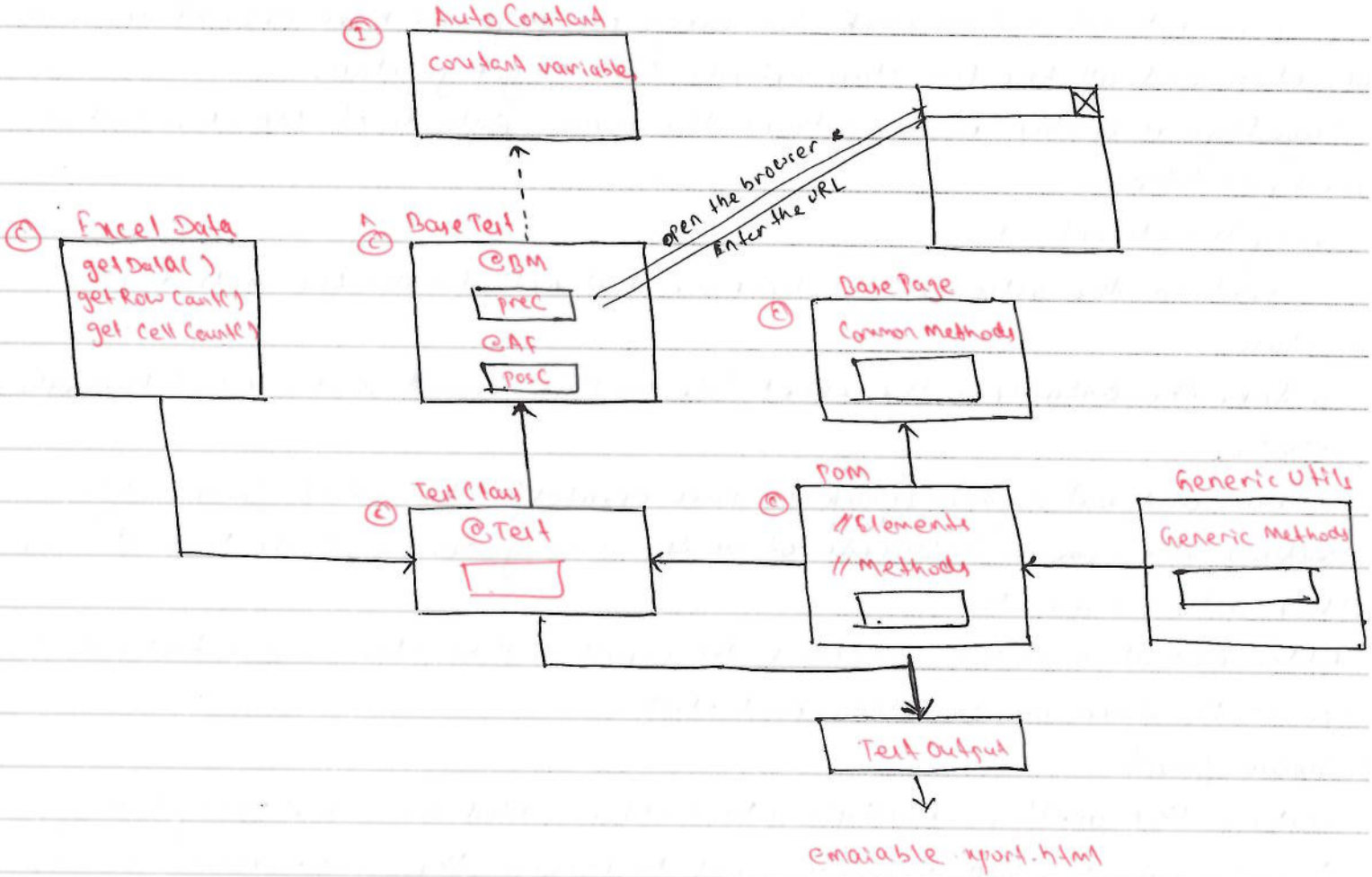
Q:- How do you execute your automation framework?

An By using testng.xml file.

Q:- How do you execute only failed steps? (Using testng-failed.xml)

Q:- Can I execute the framework with out using eclipse? (Yes, Using Command line)

Q:- Explain your automation framework.



→ It is a hybrid framework which is a combination of Data Driven, Method Driven, & TestNG.

→ It is developed using TestNG.

→ In our automation framework we have created an interface called AutoConstant, to store all the constant variable.

→ This interface is implemented using a class called BaseTest.
→ BaseTest class contains the common statements of all the test classes such as
- open the browser
- Enter the url
- close the browser
→ In our automation framework for every test case we have created separate test class and all the test class extends from BaseTest class.
→ In our automation framework for every webpage we have created separate POM class and all the POM class extends from BagePage class.
→ BagePage is a class which contains the common methods of POM class such as
- verify the title
- verify the element etc.
→ To perform the action on the browser test method calls the methods of POM class.
→ To take the data from the excel file we have created a class called BaseTest.ExcelData.
→ In our automation framework we have created a class called Generic utils. It contains the ~~common statements~~ which contains of all the some of generic methods to handle the listbox, action class etc.
→ After execution of the framework the result will be store under test output folder in the form of emalable report.html

Selenium Grid:-
→ Testing the appln in multiple platforms is called as Compatiablity Testing.
→ In selenium we use selenium grid to perform the compatiablity testing.
→ Here we will take two types of system.
(i) Hub
(ii) Node
→ Hub is a system where the automation framework is present. Technically we call it as server.

→ Node is a system where actual execution will happen. Technically we call it as Client.

→ For one hub we can have one or more nodes.

Configuring Node:-

Tools Required:-

- jdk
- driver executable files
- selenium jar file
- Browsers
- Application Under Testing

→ In the node system create a folder with the name node and store all the driver executable files and selenium jar file.

→ Rename the driver executable file and jar file as c.exe, g.exe and s.jar

→ In the node folder create a file with the following command and save it with .bat extention (batch file).

java -Dwebdriver.chrome.driver=c.exe -Dwebdriver.gecko.driver=g.exe -jar s.jar

→ Here '-D' is stands for Drivers.

→ Double click on the batch file it will execute the command and displays the following message

Selenium Server is up and running

Configuring Hub:-

→ To execute the framework in the remote machine we use a sett class called remote webdriver.

→ Remote webdriver class takes two argument of type URL and Desired Capabilities

URL:-

→ Here we have to specify the site URL of the Remote machine by using the class called URL.

e.g. URL url = new URL ("http://localhost:4444/wd/hub");

Desired Capabilities:-

→ It is used to set the name of the browser.

eg. Update the preCondition() as shown below.

```
@Beforemethod
public void preCondition() throws MalformedURLException
{
    URL url = new URL("http://localhost:4444/wd/hub/");   (import java.net)
    DesiredCapabilities dc = new DesiredCapabilities();
    dc.setBrowserName("chrome");
    driver = new RemoteWebDriver(url,dc);
    driver.manage().window().maximize();
    driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
    driver.get("https://demo.actitime.com/login.do");
}
```

→ After execution of framework the result will be stored in testoutput folder of hub system.

TestNG Parameter :-

→ Passing the value from testng.xml file to test class which is called as TestNG parameters.

eg.
```
<suite name="Suite">
    <test name="Test">
    <parameter name="city" value="Banagalore"/>
    <parameter name="area" value="Banaswadi"/>
    <classes>
        <class name="qsp.Demo"/>
    </classes>
    </test>
</suite>
```

```
public class Demo
{
    @Parameters({"city","area"})
    @Test
```

// you want to execute chrome in 182 &
firefox in 2010 just change the ip address

nodeURl
browser
appURl

```
public void testA(String city, String area)
{
    Reporter.log(city,true);
    Reporter.log(area,true);
} }
```

o/p
Bangalore
Basavanagudi

→ Update TestNG.xml file and preCondition( ) as follows

e.g.
```
<suite name="Suite" parallel="tests">  parallelly execute all the methods present inside test block
<test name="FirefoxTest">
<parameter name="nodeUrl" value="http://localhost:4444/wd/hub/"/>
<parameter name="browser" value="firefox"/>
<parameter name="appUrl" value="http://demo.actitime.com/login.do"/>
<packages>
    <package name="com.actitime.tests"/>
</packages>
</test>
<test name="ChromeTest">
<parameter name="node Url" value="http://localhost:4444/wd/hub/"/>
<parameter name="browser" value="chrome"/>
<parameter name="appUrl" value="https://demo.actitime.com/login.do"/>
<packages>
    <package name="com.actitime.tests"/>
</packages>
</test>
<suite>
```

e.g.
```
@Parameters({"nodeURL","browser","appUrl"})
@BeforeMethod
public void preCondition(String nodeUrl, String browser, String appUrl) throws MalformedURLException
{
    URL url = new URL(nodeUrl);
    DesiredCapabilities dc = new DesiredCapabilities();
```

Allocate jar file store in Referenced Libraries
Not in maven " " " " Maven Dependencies
Normal jar files retrive in maven Repository

```
dc.setBrowserName(browser);
driver = new RemoteWebDriver(url, dc);
driver.manage().timeOuts().implicitlyWait(10, TimeUnit.SECONDS);
driver.get(appUrl);
}
```

Maven:- → Always correct with internet

20/02/2018

→ Maven is a dependency tool.
→ In selenium we use the maven project to update the jar files automatically.
→ Implementation of maven project is a part of automation framework design.
→ There are 3 stages in Maven. They are
(i) Create a maven project
(ii) Specify the dependencies
(iii) Execute the framework

Maven Demo (Java Project)
Right click on java project → Configure → Convert to Maven project

(i) Create a Maven Project:- First create a java project Maven Demo
- Right click on the java project
- Go to configure
- click on Convert to Maven Project
- Follow the default instructions ~~and take~~ until finish, which will create a file call pom.xml.

→ pom stands for Project Object Model

(ii) Specify the dependencies:-
→ Here dependencies refers to jar files.
→ To add the dependencies
- Double click on pom.xml file which will open the Pom editor
- Go to dependencies tab and click on add
- Specify the GroupId, ArtifactId, Version.
- Click on OK and save
- We can get the dependencies from the following website

https://mvnrepository.com/

POM → Project object model
* → some changes happened

Organization Name + GroupId
Name of jar file + ArtifactId
Specify the version + version

MvnRepository
click on POM.xml → Go to dependencies

selenium

```
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.11.0</version>
</dependency>
```

poi

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>3.17</version>
</dependency>
```

poi-ooxml

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>3.17</version>
</dependency>
```

testng

```
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>LATEST</version>
</dependency>
```

→ Here groupId refers to Organisation Name, artifactId refers to product name, version refer to download the latest version of the jar file, specify the version as 'LATEST'.

(iii) Execute the MavenProject:-

→ During the execution time it will download all the specified dependencies into maven local repository (.m2\repository).

→ To compile the maven Project we use Maven-compiler-plugin and to execute the maven project we use maven-surefire-plugin.
→ maven-surefire-plugin will be available from the following website.
https://maven.apache.org/surefire/maven-surfire-plugin/examples/testng.xml
→ Copy the maven-surefire-plugin and paste it after maven-compiler-plugin.
→ To execute the maven project - Right click on pom.xml file
- Go to RunAs
- Click on MavenTest
→ pom.xml file will trigger the testng.xml file and testng.xml file will execute all the test classes.

2:- What are the difference betn Java Project & Maven Project?

Java Project	Maven Project
• All the jar files should be updated manually.	• All the jar files ~~should~~will be updated automatically.
• All the jar files are stored under jars folder.	• All the jar files are stored under .m2\repository (maven local repository)
• All the associated jar files are stored under Reference Libraries.	• All the associated jar files are stored under Maven Dependencies.
• Here we use testng.xml file to execute the framework.	• Here we use pom.xml file to execute the framework.
• The result will be stored under test-output folder.	• The result will be stored under target folder.

Git Hub:—

→ GitHub is a version control tool or an online repository tool which is used to store the automation framework and to manage the automation scripts.
→ Once the framework is designed lead will create a repository in the gitHub server and he will upload the automation framework into github repository.
→ All the automation engineers should download the framework from github to their local systems.

target → surefire plugin → enable reporting → To open the Report

☑ user or create

→ Once after writing the scripts the automation engrs should upload the scripts into the github server.
→ This process is called as Push.
→ To download the files from github we use an option called Pull.

Uploading framework to github:-

→ It is done by lead/project manager.
- Login to following website
  https://github.com/login
- click on start a project
- Specify the repository name. Repository name should be same as Project Name
- select public/private buttons
- click on create Repository which will generate an URL
  https://github.com/SBRakesh/MavenDemo.git
- Copy the URL
- In eclipse Right click on the project, go to Team and click on share project
- Select the checkbox and select the loc? and click on create Repository and finish. which will create a Repository in the local system.
- Right click on the project
- Go to Team
- Click on Commit
- Select all the files & Specify the commit message
- Click on commit & push
- Specify the URL, username, password and click on next.
- Follow the default instructions until OK

Steps to Download framework from GitHub:- 21/03/2018

→ It should be done by all the automation engineers.
→ Once the framework is designed lead will store the framework in github server and he will share the URL to all the team members.
- In eclipse go to file
- click on import

localhost:8080

- Expand Git folder and select projects from git and click on Next
- Select clone url and click on Next
- Specify the url, user Name and password
- Follow the default instructions until finish which will download the framework from github to local workstation.

Steps to upload modified file or new file to GitHub:-

→ It should be done by all the automation Engineers once if they modifying the existing file (>) or create a new file (?)

- Right click on the modified file or new file and click on commit
- Select the files and drag and drop into staged changes and specify the commit message.
- Click on commit & push and follow the default instructions till finish.

Steps to download framework from GitHub:-

- Right click on the project.
- Go to Team
- Click on Pull & follow the default instructions which will download the modified files from github server to local workstation

Jenkins:-

→ Jenkins is a continuous Integration tool which is basically used by developers to create and manage the build.

→ We can integrate the framework with Jenkins so that the framework build will be executed automatically when the new build is created.

→ To integrate the framework with Jenkins we need the following information

- Url of the Jenkins.
- Project Name
- User Name & Password of Jenkins.

NOTE:- Installation & Configuration of Jenkins is done by developers.

→ Build is created by developers.

→ We have to integrate our framework with Jenkins

Steps to integrate framework with Jenkins:-

- Login to Jenkins
- Click on Project
- Click on Configure
- Select Git under source code management
- Specify the url of the github Repository
- Click on add and select Jenkins
- Enter the Jenkins user name & password
- Click on OK
- Click on Apply & save

→ To create the build developer will click on an option Build Now.

When developer performs the Build Now option Jenkins will perform the following steps.

- Download source code from developers Repository
- Compile the source code
- Compress the source code
- Sends email notification
- Downloads the framework
- Execute pom.xml
- Pom.xml will execute the TestNG.xml
- After execution of the framework the result will be stored in Jenkins server
- To check the result click on the project, & click on the link present under Build History and click on Console output