

Capstone Project

Project Name: Human Resource Management

Submitted By: Ritik Prajapat [StarAgile Student Trainee]

Submitted To:  [StarAgile Consulting Pvt Ltd]

GitHub link:

<https://github.com/ritikprajapat8085/AutomationProjects/tree/OHRM>

Table of Contents

1. Project Overview
2. Objectives
3. Scope of the Project
4. Tools & Technologies Used
5. Test Case Scenarios
6. Testcase Design & Execution
7. Test Result Analysis & Reporting
8. Conclusion

1. Project Overview

OrangeHRM is an open-source Human Resource Management tool used globally to manage employee records, leave, attendance, and recruitment. In this capstone project, I automated critical HR functionalities using Selenium, Java, and TestNG to reduce manual effort and enhance test reliability.

Human Resource Management is a product that offers services like below:

- Adding new employees,
- Modifying existing employees
- Deleting existing employees
- Searching employees
- Raising a request for hiring a new employee
- Approving a new request
- Submitting timesheet
- Approving timesheet, etc.

2. Objective

- To validate the core functionalities of the OrangeHRM application, like user login/logout, adding employees, managing leaves, etc., and ensure they are working as expected.
- Implement Excel-based data-driven testing to verify login and other functionality against multiple data sets, including valid and invalid combinations.
- Design and implement a modular, scalable, and maintainable test framework using the Page Object Model design pattern.
- Automate the execution of test cases through TestNG and generate detailed execution reports using ExtentReports to track the status of each test case.

3. Scope of the Project:

The project covers UI-based functional testing of the OrangeHRM application modules like Login, Add Employee, Search Employee, and Logout. It includes positive and negative test cases, Excel data-driven testing, and result reporting.

- 1 . Login/logout
2. Admin
 - 2.1. User Management

4. Tools & Technologies Used:

| Tool / Tech | Purpose |
|----------------------|--------------------------------|
| • Java | Programming Language |
| • Selenium WebDriver | Automation Testing Tool |
| • TestNG | Test Management and Assertions |
| • Apache POI | Reading Excel files |
| • ExtentReports | Report Generation |
| • Eclipse / IntelliJ | IDE |
| • Maven | Project Build Tool |
| • GitHub | Version Control |
| • OrangeHRM | Application Under Test |

5. Test Case Scenarios:

Scenario 1:

Automate Orange HRM login and logout with 5 different datasets, including valid and invalid datasets.

1. Save the dataset in an Excel file
2. Write a script to read data from Excel
3. Prepare a script for Login and logout
4. Perform assertion for a valid data set (use Username: Admin and Password:admin123) test case should pass, and an invalid data set (other than the given data) test case should fail.
5. Capture a Screenshot for every login functionality
6. Generate an Extent Report for the same.

Scenario 2:

Create a Page Object Model for two pages Login Page and the Admin Pageand write an automation script for the Login functionality and Admin search feature.

Create a Page and Class for the Login Page and automate the functionality of Orange HRM login for Valid Test Data: (username: Admin and Password: admin123).

Create a Page and a class for Admin where you can prepare 4 important test cases:

1. Create a test case to get all 12 options from the left side menu and print the count, which should be 12 from that list. Click on Admin, and then the Admin page will open.
2. Create a test case for the search For Existing Employee search by username (): Here, send the username Admin to the username text box and click on the search button and display the total number of records found and refresh the page.
3. Create a test case for search For Existing Employee search ByUserRole(): here, automate the dropdown and select Role Admin and click on the search button and display the total record found and refresh the page
4. Create a test case for the search For Existing Employee search ByUserStatus(): here, automate the dropdown and select status Enabled or Disabled, then click on the search button and display the total record found.

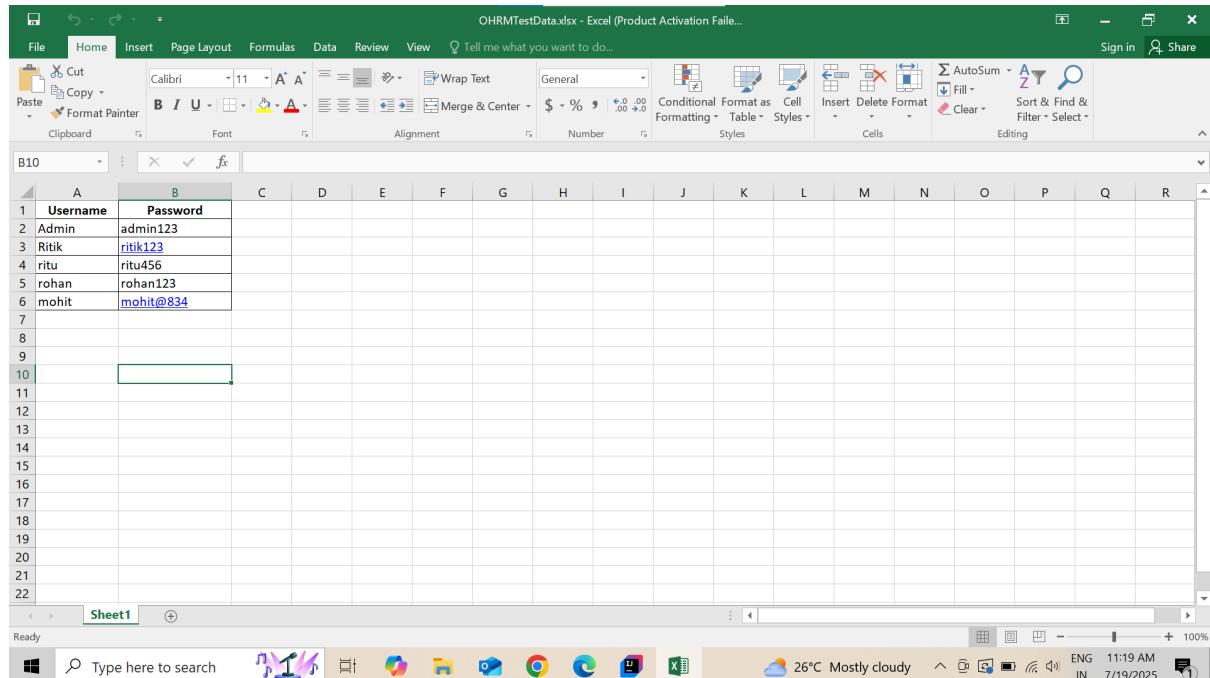
Note: In the Second scenario, use Selenium TestNG and design a Page Object Model where you can implement other Object Oriented Principles.

6. Testcase Design & Execution:

Scenario 1:

1. Save the dataset in an Excel file

Test data is maintained in an Excel file. The Apache POI library is used to read multiple sets of data, including usernames and passwords.

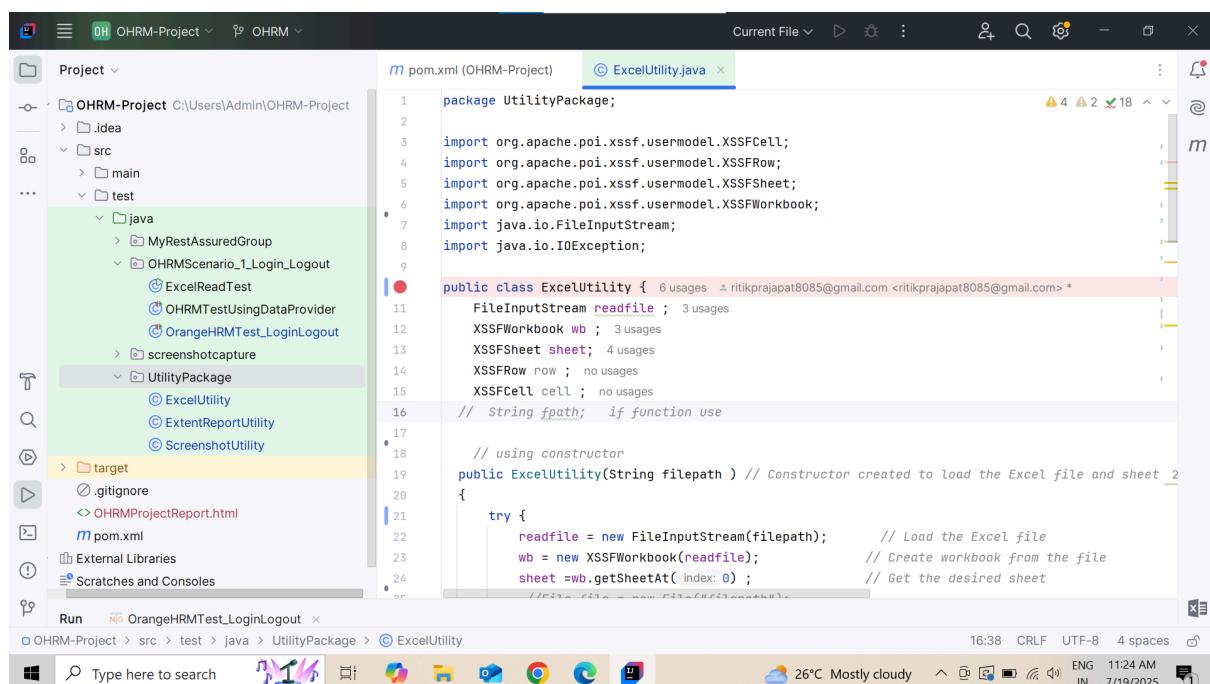


The screenshot shows a Microsoft Excel spreadsheet titled "OHRMTestData.xlsx". The data is organized in a table with two columns: "Username" and "Password". The rows contain the following data:

| | Username | Password |
|---|----------|-----------|
| 2 | Admin | admin123 |
| 3 | Ritik | ritik123 |
| 4 | ritu | ritu456 |
| 5 | rohan | rohan123 |
| 6 | mohit | mohit@834 |

2. Write a script to read data from Excel

I have created a separate Excel utility class.



The screenshot shows an IDE interface with a Java project named "OHRM-Project". The code editor displays the "ExcelUtility.java" file, which contains the following code:

```
package UtilityPackage;

import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import java.io.FileInputStream;
import java.io.IOException;

public class ExcelUtility {
    FileInputStream readfile ; 3 usages
    XSSFWorkbook wb ; 3 usages
    XSSFSheet sheet ; 4 usages
    XSSFRow row ; no usages
    XSSFCell cell ; no usages
    // String fpath; if function use

    // using constructor
    public ExcelUtility(String filepath) // Constructor created to load the Excel file and sheet
    {
        try {
            readfile = new FileInputStream(filepath); // Load the Excel file
            wb = new XSSFWorkbook(readfile); // Create workbook from the file
            sheet = wb.getSheetAt(index: 0); // Get the desired sheet
        }
    }
}
```

The screenshot shows a Java code editor with the file `ExcelUtility.java` open. The code implements a utility class for reading data from an Excel file. It uses the Apache POI library to handle the file. The code includes methods to get cell data by row and column index, get the total number of rows and columns, and close the workbook.

```
public class ExcelUtility { // using constructor
    public ExcelUtility(String filepath) // Constructor created to load the Excel file and sheet
    {
        try {
            readfile = new FileInputStream(filepath); // Load the Excel file
            wb = new XSSFWorkbook(readfile); // Create workbook from the file
            sheet = wb.getSheetAt(index: 0); // Get the desired sheet
            //File file = new File("filepath");
            //writefile = new FileOutputStream(file); //Configure fileoutputstream only after Configuration
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }

    public String getCellData(int r, int c) // Method to get username/password cell data
    {
        return sheet.getRow(r).getCell(c).toString();
    }

    public int getRowCount() // Optional: Get total row count
    {
        // return sheet.getLastRowNum(); // gives index of last row (starts at 0).
        return sheet.getPhysicalNumberOfRows(); // Count rows
    }

    public int getColumnCount() // Optional: Get total column count in first row
    {
        return sheet.getRow(0).getLastCellNum();
    }

    // Close workbook
    public void closeWorkbook() throws IOException
    {
        wb.close();
        readfile.close();
    }
}
```

The screenshot shows the same Java code editor with additional code added to the `closeWorkbook` method. This new code releases the resources used by the workbook and file streams.

```
public void closeWorkbook() throws IOException
{
    wb.close();
    readfile.close();
}
```

3. Prepare a script for Login and logout

```
package ORHMScenario_1_Login_Logout;

import UtilityPackage.ExcelUtility;
import UtilityPackage.ExtentReportUtility;
import UtilityPackage.ScreenshotUtility;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.*;
import java.io.IOException;
import java.time.Duration;

public class OrangeHRMTest_LoginLogout { @ritikprajapat8085@gmail.com <ritikprajapat8085@gmail.com>
    WebDriver driver; 13 usages
    ExcelUtility excel; 5 usages
    ScreenshotUtility screenshotUtil; 3 usages
    ExtentReportUtility reportUtil; 6 usages

    // These variables are for each test iteration
    String username ; 7 usages
    String password ; 4 usages
    int currentRow = 1; // for reading excel Row index start from 1 and column :0 3 usages
```

```
String username ; 7 usages
String password ; 4 usages
int currentRow = 1; // for reading excel Row index start from 1 and column :0 3 usages

@BeforeTest new *
public void setup() throws IOException {
    excel = new ExcelUtility(filepath: "C:\\Users\\Admin\\Documents\\Staragile Automation course\\CapstoneProjects\\OHRMProject\\OHRMTestData.xlsx");
    reportUtil = new ExtentReportUtility(); // Setup Extent Report only once
}

@BeforeMethod @ritikprajapat8085@gmail.com <ritikprajapat8085@gmail.com>
public void loginSetup() throws Exception {
    driver = new ChromeDriver();
    driver.manage().window().maximize(); // Maximize window
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10)); // wait max for 10 sec
    driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login"); // Open OrangeHRM site
    Thread.sleep(2000); // Pause to let page load
    screenshotUtil = new ScreenshotUtility(driver); // here bcz we are not setting for one time we are calling every time
}
```

```
@Test(invocationCount = 5) // Run this test 5 times for 5 rows of data @ritikprajapat8085@gmail.com <ritikprajapat8085@gmail.com>
public void logintest() throws InterruptedException, IOException {

    int rowCount = excel.getRowCount(); // calling the function from excelutility class

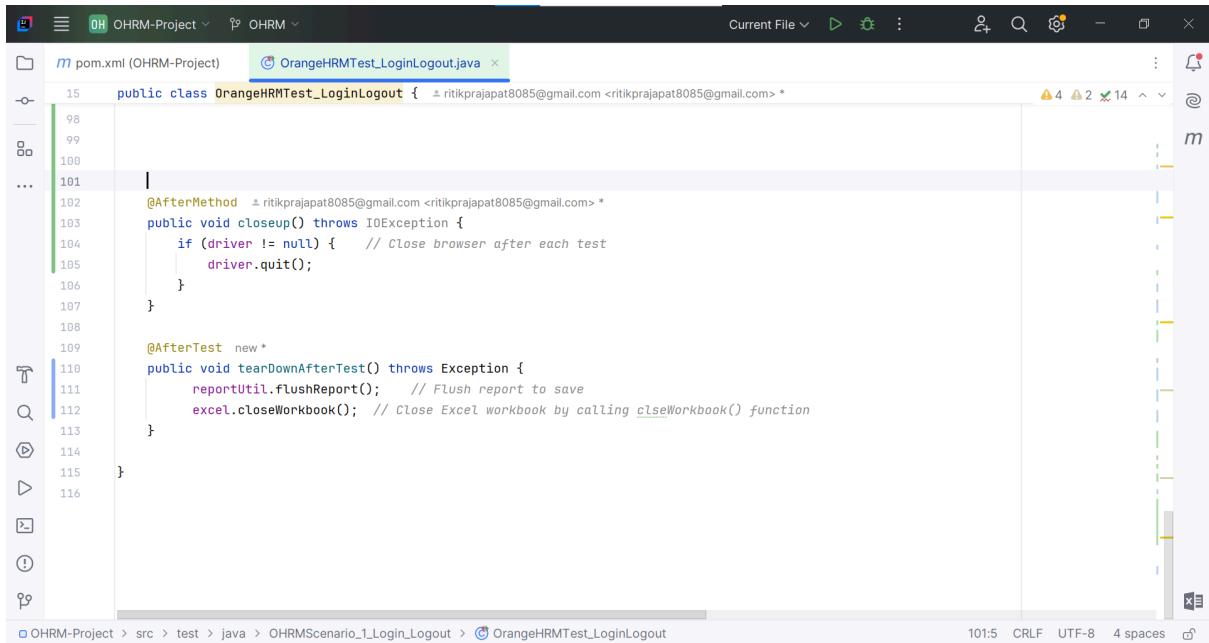
    username = excel.getCellData(currentRow, c: 0); // Get current row's username and password
    password = excel.getCellData(currentRow, c: 1); // Get current row's username and password

    reportUtil.createTest( testName: "Login Test: " + username + " / " + password); // Create report entry by calling extentreportutility

    driver.findElement(By.name("username")).sendKeys(username);
    driver.findElement(By.xpath( xpathExpression: "//input[@type='password']")).sendKeys(password);
    WebElement btn = driver.findElement(By.tagName("button"));
    btn.click();
    Thread.sleep(3000); // Wait for login to process

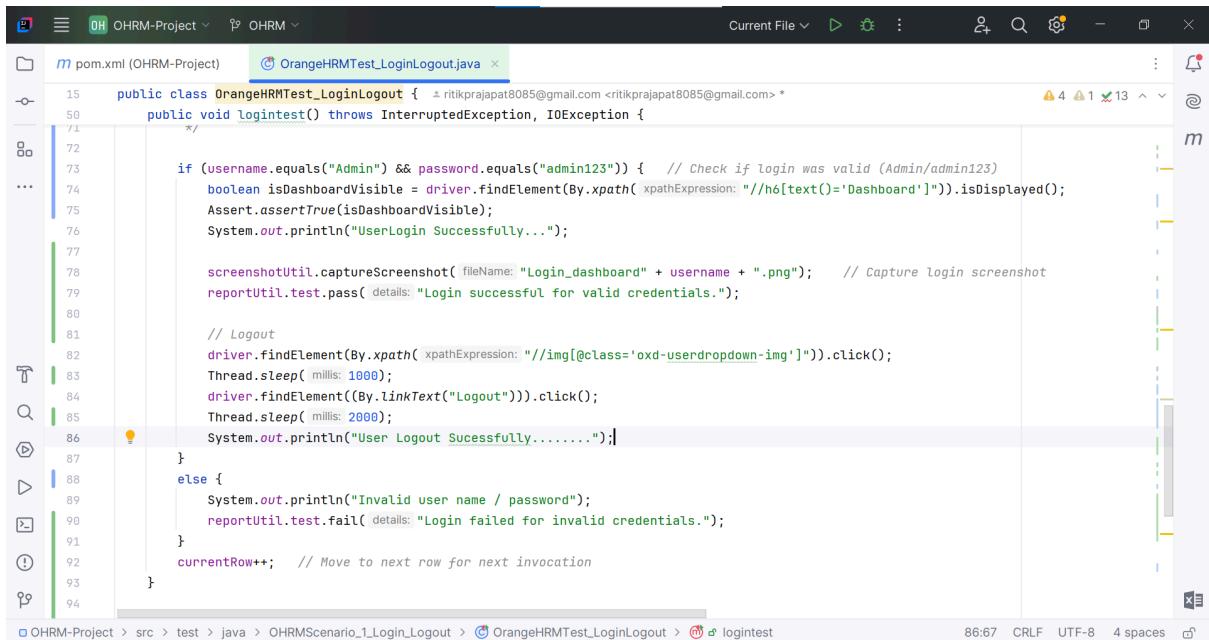
    screenshotUtil.captureScreenshot( fileName: "Login_" + username + ".png"); // Capture login screenshot
    reportUtil.test.addScreenCaptureFromPath("src/test/java/screenshotcapture/Login_" + username + ".png");

    /*
        Assert.assertTrue(driver.getCurrentUrl().contains("dashboard")); // if you assert here then invalid test will failed x marks no
        System.out.println("User Login Sucessfully....");
    */
}
```



```
public class OrangeHRMTest_LoginLogout { ...  
    @AfterMethod  
    public void closeup() throws IOException {  
        if (driver != null) { // Close browser after each test  
            driver.quit();  
        }  
    }  
  
    @AfterTest new *  
    public void tearDownAfterTest() throws Exception {  
        reportUtil.flushReport(); // Flush report to save  
        excel.closeWorkbook(); // Close Excel workbook by calling closeWorkbook() function  
    }  
}
```

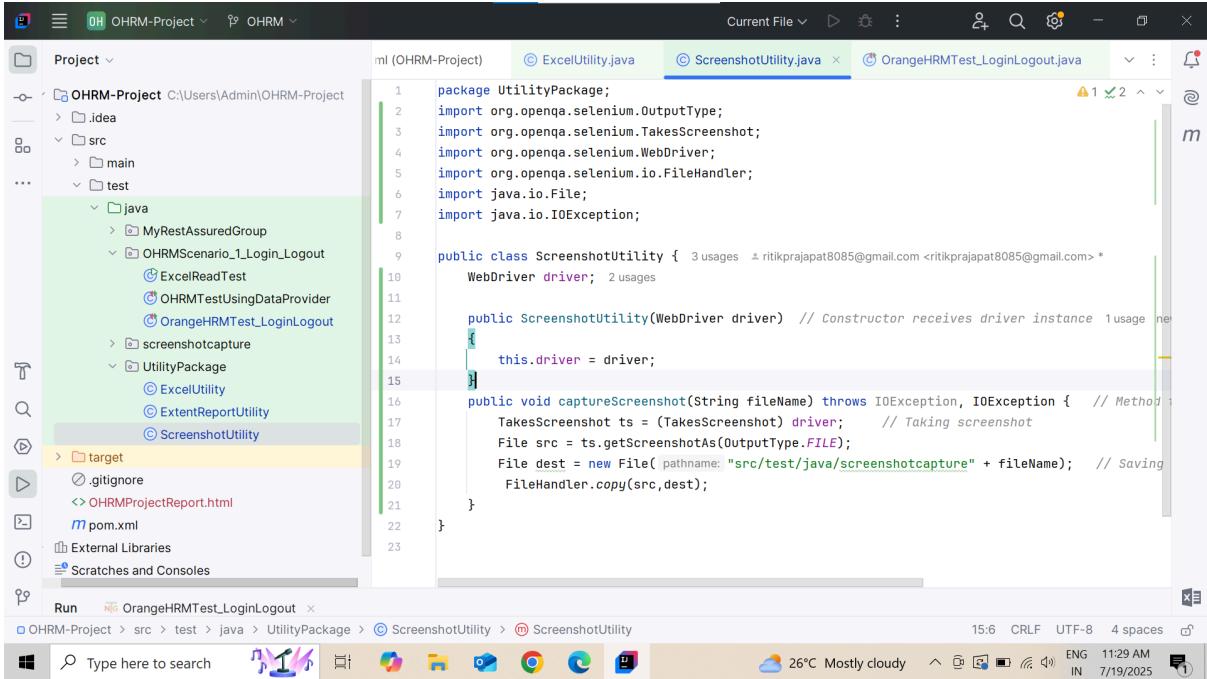
4. Perform assertion for a valid data set (use Username: Admin and Password:admin123) test case should pass, and an invalid data set (other than the given data) test case should fail.



```
public void logintest() throws InterruptedException, IOException {  
    /*  
     *  
     * Check if login was valid (Admin/admin123)  
     */  
    if (username.equals("Admin") && password.equals("admin123")) {  
        boolean isDashboardVisible = driver.findElement(By.xpath(xpathExpression: "//h6[text()='Dashboard']")).isDisplayed();  
        Assert.assertTrue(isDashboardVisible);  
        System.out.println("UserLogin Successfully....");  
  
        screenshotUtil.captureScreenshot(fileName: "Login_dashboard" + username + ".png"); // Capture login screenshot  
        reportUtil.test.pass(details: "Login successful for valid credentials.");  
  
        // Logout  
        driver.findElement(By.xpath(xpathExpression: "//img[@class='oxd-userdropdown-img']")).click();  
        Thread.sleep(millis: 1000);  
        driver.findElement(By.linkText("Logout")).click();  
        Thread.sleep(millis: 2000);  
        System.out.println("User Logout Sucessfully.....");  
    }  
    else {  
        System.out.println("Invalid user name / password");  
        reportUtil.test.fail(details: "Login failed for invalid credentials.");  
    }  
    currentRow++; // Move to next row for next invocation  
}
```

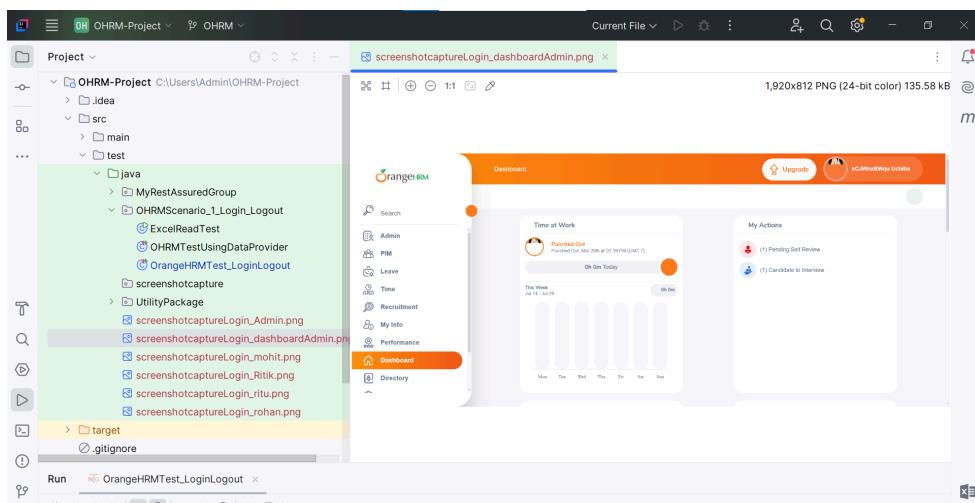
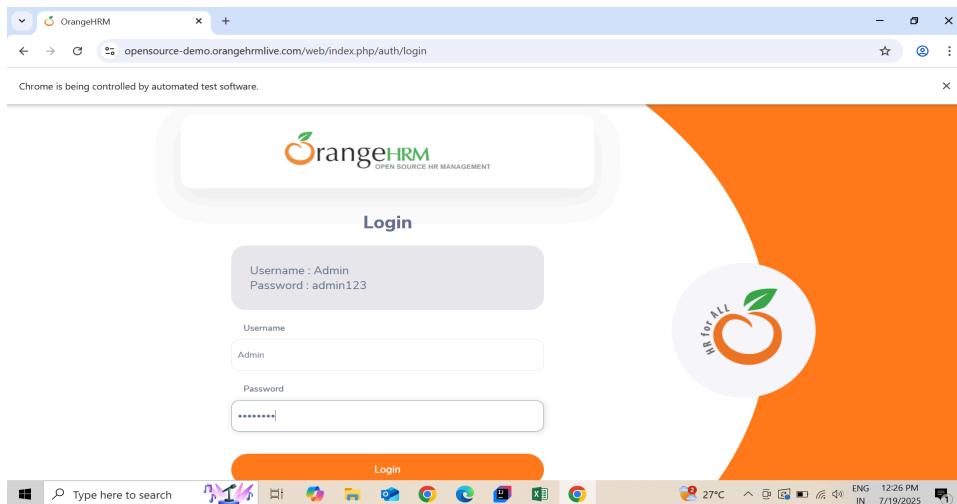
5. Capture a Screenshot for every login functionality

I have created a separate **Screenshot** utility class.

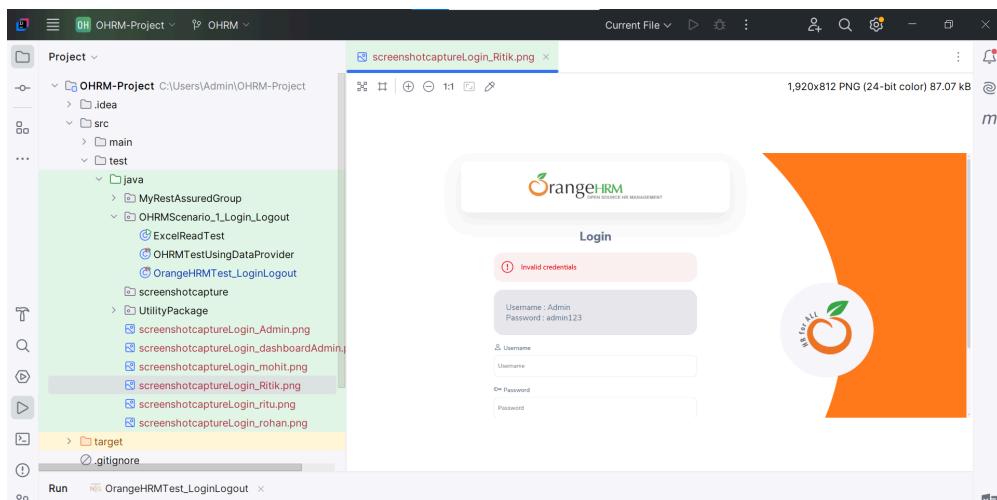
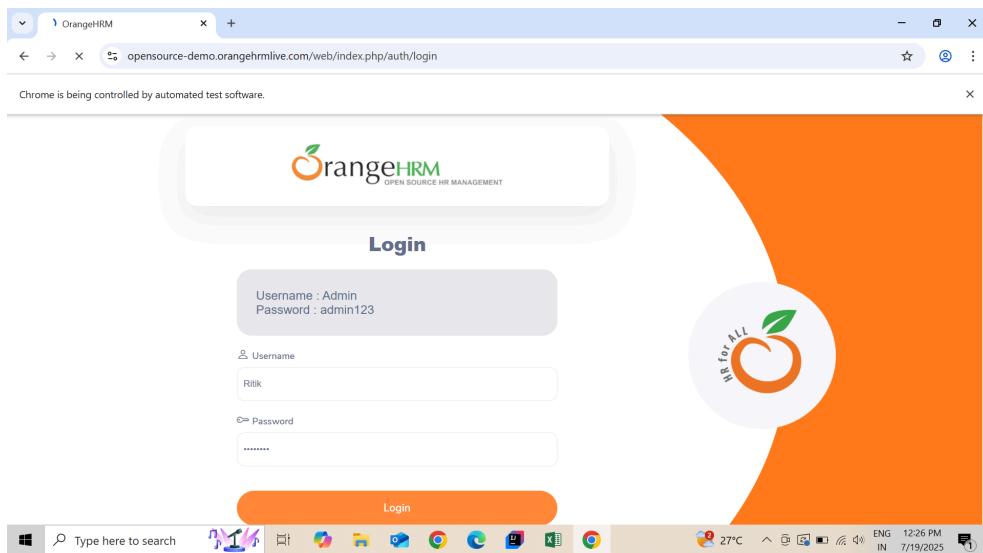


```
1 package UtilityPackage;
2 import org.openqa.selenium.OutputType;
3 import org.openqa.selenium.TakesScreenshot;
4 import org.openqa.selenium.WebDriver;
5 import org.openqa.selenium.io.FileHandler;
6 import java.io.File;
7 import java.io.IOException;
8
9 public class ScreenshotUtility { 3 usages ± ritikprajapat8085@gmail.com <ritikprajapat8085@gmail.com> *
10   WebDriver driver; 2 usages
11
12   public ScreenshotUtility(WebDriver driver) // Constructor receives driver instance 1 usage new
13     {
14       this.driver = driver;
15     }
16
17   public void captureScreenshot(String fileName) throws IOException, IOException { // Method
18     TakesScreenshot ts = (TakesScreenshot) driver; // Taking screenshot
19     File src = ts.getScreenshotAs(OutputType.FILE);
20     File dest = new File( pathname: "src/test/java/screenshotcapture" + fileName); // Saving
21     FileHandler.copy(src,dest);
22   }
23 }
```

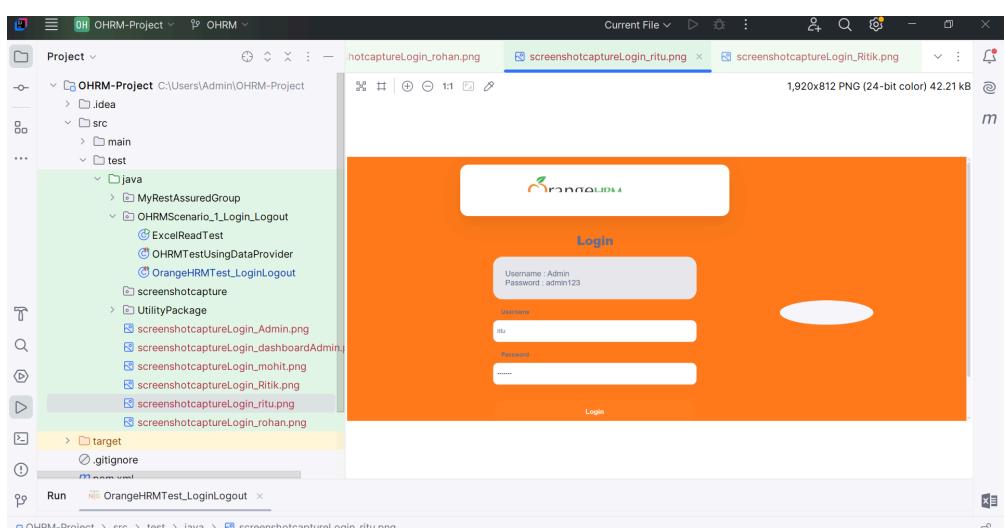
1. Username:Admin & Password: admin123

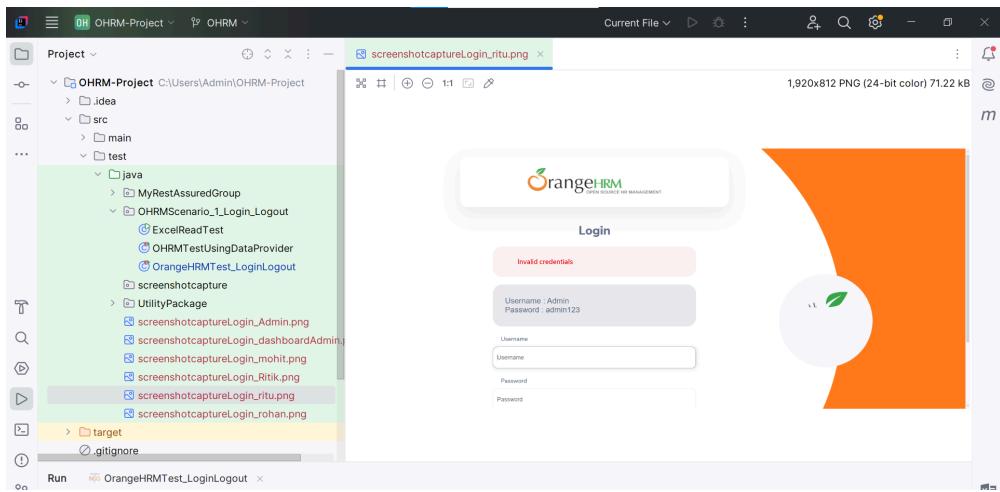


2. Username:ritik & Password: ritik123

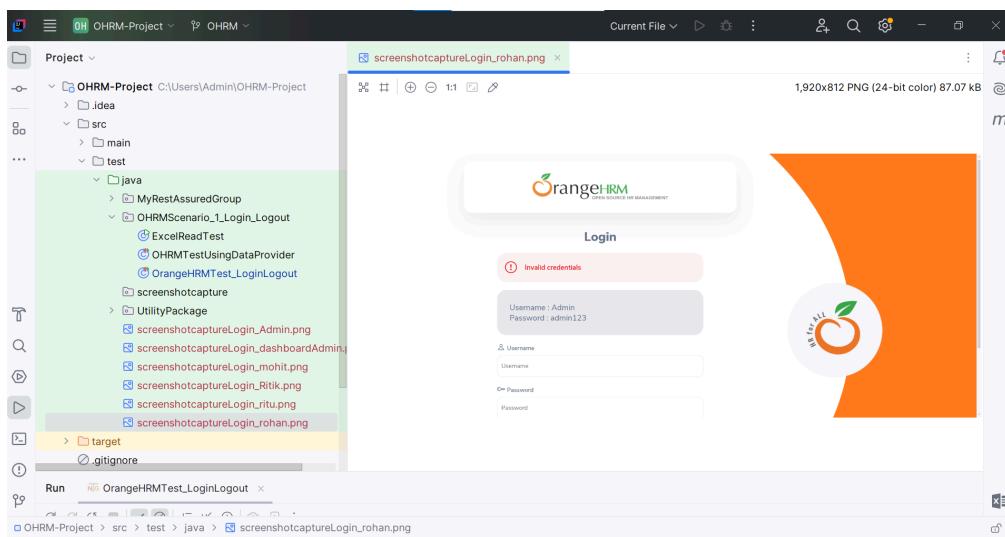
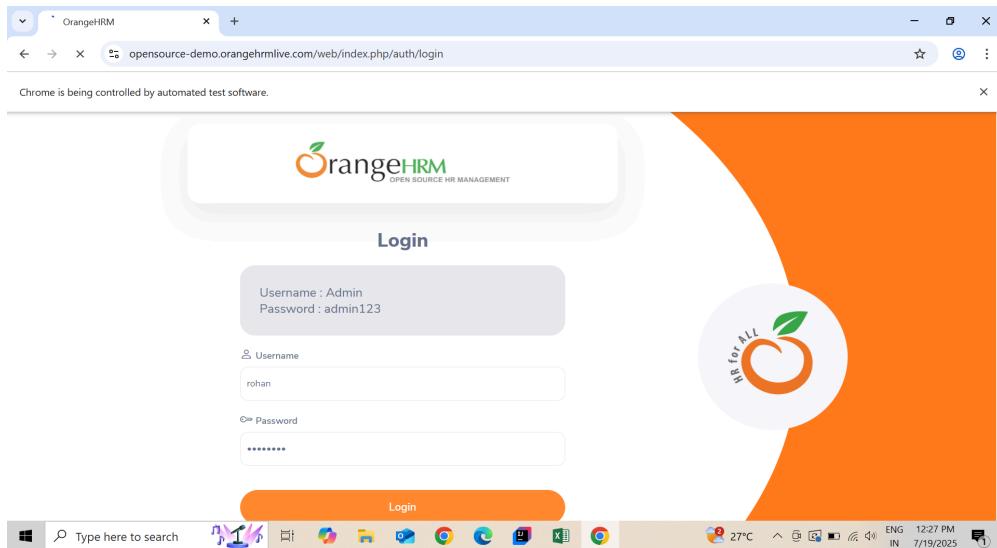


3. Username:ritu & Password:ritu456

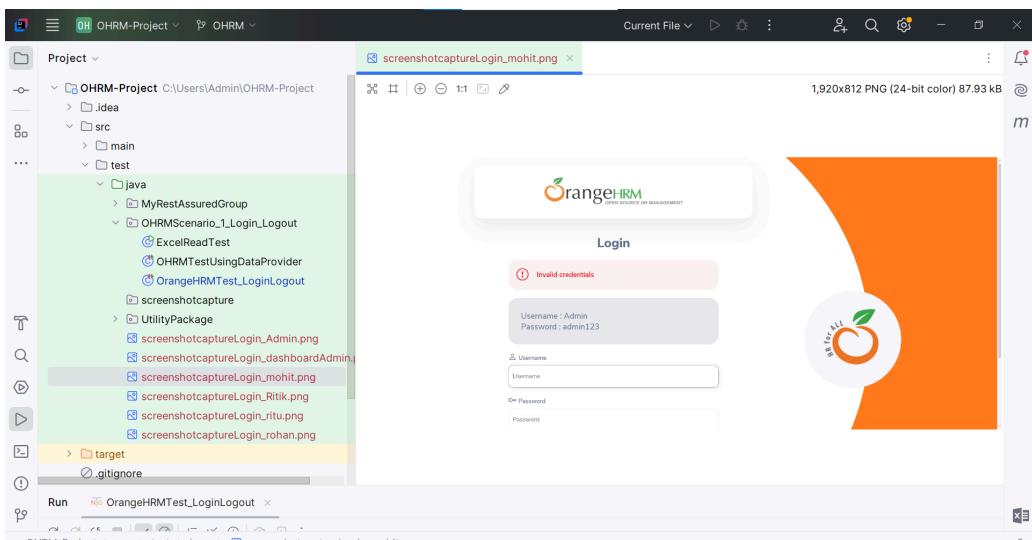
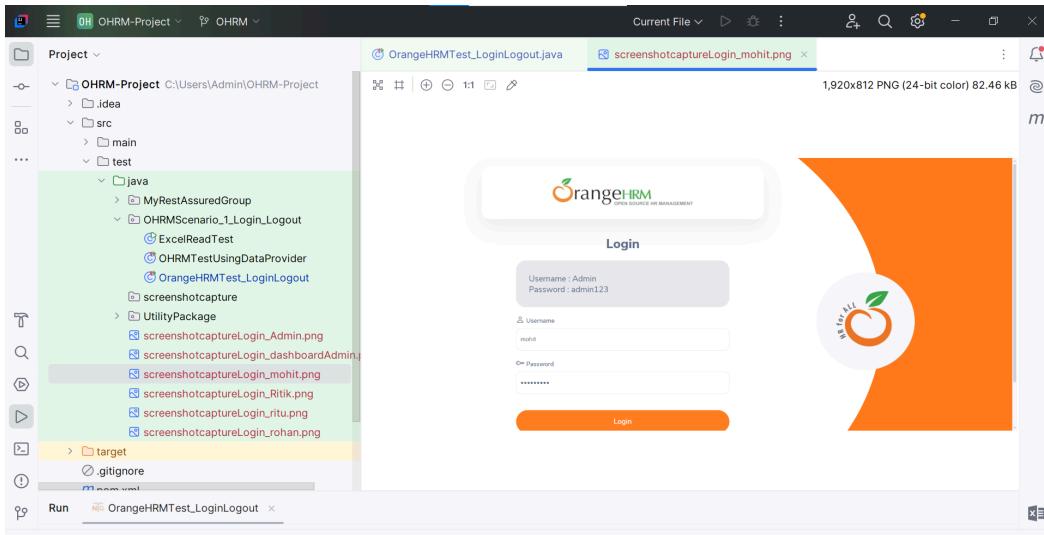




4. Username:rohan & Password: rohan123



5. Username:mohit & Password: mohit@834



6. Generate an Extent Report for the same.

I have created a separate **Extent Report** utility class

A screenshot of the IntelliJ IDEA interface showing the code for the 'ExtentReportUtility' class. The code is part of the 'UtilityPackage' and defines a class 'ExtentReportUtility' with methods for managing reports, running extent tests, and setting up report paths. It also includes configuration for the report, such as document title, report name, theme, and timestamp format. The code is annotated with Javadoc comments explaining its purpose and usage.

```

public class ExtentReportUtility {
    public ExtentReportUtility() {
        // Add environment information
        extent.setSystemInfo("DeviceName", "Lenovo");
        extent.setSystemInfo("Generation", "Core-i3-10Gen");
        extent.setSystemInfo("Operating System", "Windows 10");
        extent.setSystemInfo("Browser", "Chrome");
        extent.setSystemInfo("TesterName", "Ritik Prajati");
        extent.setSystemInfo("Application Under Test", "OrangeHRM ");
    }
    // Method to create new test case in report
    public void createTest(String testName) {
        test = extent.createTest(testName);
    }
    // Final report save whatever we did so far in report
    public void flushReport() {
        extent.flush();
    }
}

```

Scenario 2:

Create a Page Object Model for two pages **Login Page** and the **Admin Page**, and write an automation script for the Login functionality and Admin search feature.

1. Create a Page and Class for the Login Page and automate the functionality of Orange HRM login for Valid Test Data: (username: Admin and Password: admin123).

LoginPage

```

package OHRM_Scenario2_LoginPage_AdminPage_POM;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
public class LoginPage {
    WebDriver driver; 1 usage
    @FindBy(name = "username") WebElement usernameField; 1 usage
    @FindBy(name = "password") WebElement passwordField; 1 usage
    @FindBy(xpath = "//button[@type='submit']") WebElement loginButton; 1 usage
    public LoginPage(WebDriver driver) { 2 usages
        this.driver = driver;
        PageFactory.initElements(driver, page: this);
    }
    public void login(String username, String password) { 2 usages
        usernameField.sendKeys(username);
        passwordField.sendKeys(password);
        loginButton.click();
    }
}

```

The screenshot shows the IntelliJ IDEA interface with the project 'OHRM-Project' open. The 'test' directory under 'src' contains a package named 'java'. Within this package, there is a class named 'LoginTestCase'. The code for this class is displayed in the main editor window:

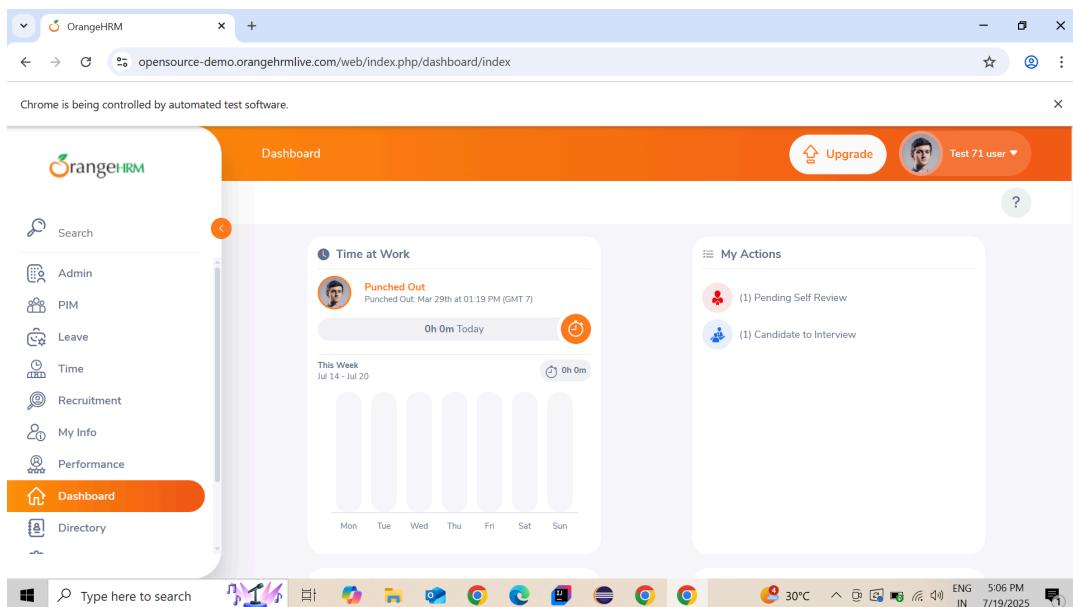
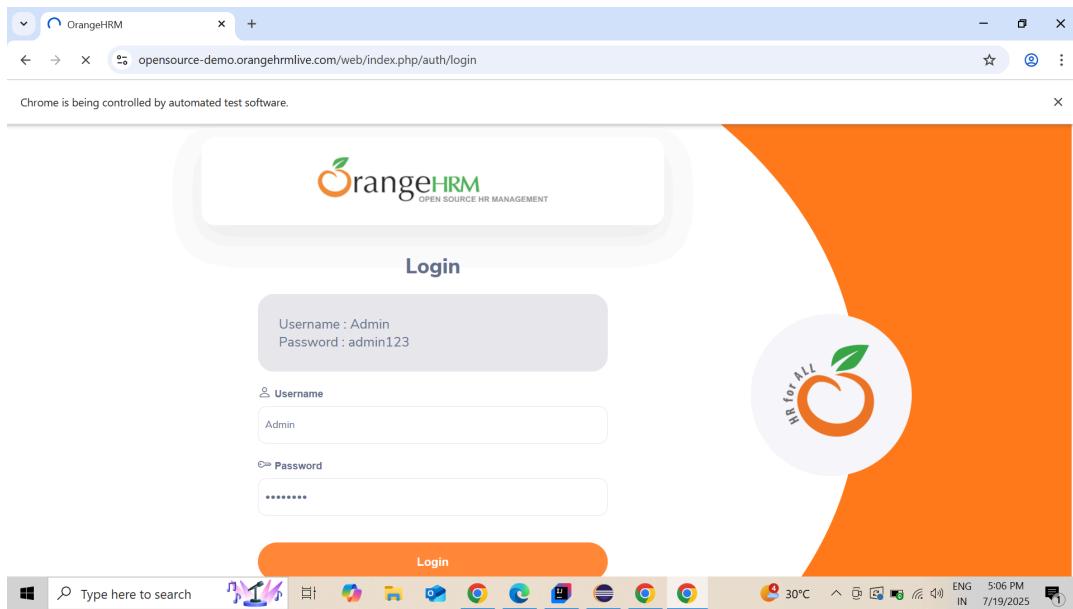
```
1 package OHRM_Scenario2_LoginPage_AdminPage_POM;
2 import org.openqa.selenium.WebDriver;
3 import org.openqa.selenium.chrome.ChromeDriver;
4 import org.testng.Assert;
5 import org.testng.annotations.AfterClass;
6 import org.testng.annotations.BeforeClass;
7 import org.testng.annotations.Test;
8 import java.time.Duration;
9
10 public class LoginTestCase {
11     WebDriver driver; 8 usages
12     LoginPage login ; 2 usages
13
14     @BeforeClass
15     public void setup() {
16         driver = new ChromeDriver();
17         driver.manage().window().maximize();
18         driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
19         driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");
20         login = new LoginPage(driver); // Create LoginPage object and pass driver object to
21     }
22
23     @Test
24 }
```

LoginTestCase

The screenshot shows the IntelliJ IDEA interface with the project 'OHRM-Project' open. The 'test' directory under 'src' contains a package named 'java'. Within this package, there is a class named 'LoginTestCase'. The code for this class is displayed in the main editor window, showing the addition of a test method and assertions:

```
1 package OHRM_Scenario2_LoginPage_AdminPage_POM;
2 import org.openqa.selenium.WebDriver;
3 import org.openqa.selenium.chrome.ChromeDriver;
4 import org.testng.Assert;
5 import org.testng.annotations.AfterClass;
6 import org.testng.annotations.BeforeClass;
7 import org.testng.annotations.Test;
8 import java.time.Duration;
9
10 public class LoginTestCase {
11     WebDriver driver; 8 usages
12     LoginPage login ; 2 usages
13
14     @BeforeClass
15     public void setup() {
16         driver = new ChromeDriver();
17         driver.manage().window().maximize();
18         driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
19         driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");
20         login = new LoginPage(driver); // Create LoginPage object and pass driver object to
21     }
22
23     @Test
24     public void validLoginTest() throws InterruptedException {
25         login.login( username: "Admin", password: "admin123");
26         Thread.sleep( millis: 3000);
27
28         // Verify Title
29         String expectedTitle = "OrangeHRM";
30         String actualTitle = driver.getTitle();
31         Assert.assertEquals(actualTitle, expectedTitle, message: "Title does not match!");
32
33         // Verify URL
34         String currentURL = driver.getCurrentUrl();
35         Assert.assertTrue(currentURL.contains("/dashboard"), message: "URL is not correct!");
36         System.out.println("User login successfully to dashboard...");
37     }
38
39     @AfterClass
40     public void close() {
41         driver.quit();
42     }
43 }
44
45 }
```

Screenshot



2. Create a Page and a class for Admin where you can prepare 4 important test cases:

AdminPage:

The screenshot shows the IntelliJ IDEA interface with three code editors open. The top editor contains the initial skeleton of the AdminPage class. The middle editor shows the implementation of various UI elements using WebDriver and PageFactory. The bottom editor shows additional methods like clickAdminTab() and printLeftMenuOptionsText().

```
1 package OHRM_Scenario2_LoginPage_AdminPage_POM;
2
3 import org.openqa.selenium.WebDriver;
4 import org.openqa.selenium.WebElement;
5 import org.openqa.selenium.support.FindBy;
6 import org.openqa.selenium.support.PageFactory;
7 import org.openqa.selenium.support.ui.Select;
8
9 import java.util.List;
10
11 public class AdminPage {
12     WebDriver driver;
13
14     // Constructor
15     public AdminPage(WebDriver driver) {
16         this.driver = driver;
17         PageFactory.initElements(driver, this);
18     }
19
20 }
```

```
11 public class AdminPage {
23     //1. Left side menu options
24     @FindBy(xpath = "//ul[@class='oxd-main-menu']/li") List<WebElement> leftMenuOptions;
25
26     // Admin tab element
27     @FindBy(xpath = "//span[text()='Admin']") WebElement adminTab;
28
29     // 2. searchByUsername() : Username search field
30     @FindBy(xpath = "(//input[@class='oxd-input oxd-input--active'])[2]") WebElement usernameSearchField;
31
32     //3. searchByUserRole() : UserRole dropdown control
33     @FindBy(xpath = "(//div[@class='oxd-select-text-input'])[1]") WebElement userRoleDropdown;
34     @FindBy(xpath = "//div[@role='listbox']/div/span") List<WebElement> userRoleOptions;
35
36     //4. searchByUserStatus(): status dropdown control
37     @FindBy(xpath = "(//i[@class='oxd-icon bi-caret-down-fill oxd-select-text--arrow'])[2]") WebElement statusDropdown;
38     @FindBy(xpath = "//div[@role='listbox']/div/span") List<WebElement> statusOptions;
39
40     // Search button
41     @FindBy(xpath = "//button[@type='submit']") WebElement searchButton;
42
43     // Reset button
44     @FindBy(xpath = "(//button[@type='button'])[1]") WebElement resetButton;
45
46     // recordbyusername
47 }
```

```
11 public class AdminPage {
43
44     // Reset button
45     @FindBy(xpath = "(//button[@type='button'])[1]") WebElement resetButton;
46
47     // recordbyusername
48     @FindBy(xpath = "//div[@class='orangehrm-horizontal-padding orangehrm-vertical-padding']/span") WebElement recordByUsername;
49
50     public void clickAdminTab() {
51         adminTab.click();
52     }
53
54     public int getLeftMenuOptionsCount() {
55         return leftMenuOptions.size();
56     }
57
58     public void printLeftMenuOptionsText() {
59         System.out.println("Left Menu Options:");
60         for (WebElement option : leftMenuOptions) {
61             System.out.println(option.getText());
62         }
63     }
64
65     public void searchByUsername(String username) {
66         usernameSearchField.sendKeys(username);
67         searchButton.click();
68     }
69 }
```

```

public class AdminPage {
    public void selectUserRole(String role) {
        UserRoleDropdown.click();
        for (WebElement option : UserRoleOptions) {
            if (option.getText().equals(role)) {
                option.click();
                break;
            }
        }
    }

    public void selectUserStatus(String status) {
        StatusDropdown.click();
        for (WebElement option : StatusOptions) {
            if (option.getText().equals(status)) {
                option.click();
                break;
            }
        }
    }

    public String getResultText() {
        return recordByUsername.getText();
    }
}

```

```

public class AdminPage {
    public void selectUserStatus(String status) {
        StatusDropdown.click();
        for (WebElement option : StatusOptions) {
            if (option.getText().equals(status)) {
                option.click();
                break;
            }
        }
    }

    public String getResultText() {
        return recordByUsername.getText();
    }

    public void resetSearch() {
        resetButton.click();
    }
}

```

BeforeTest Method:

Project Structure:

- Project > OHRM-Project > src > main > test > java > OHRM_Scenario2_LoginPage_AdminPage > AdminTestCase

AdminTestCase.java Code:

```

package OHRM_Scenario2_LoginPage_AdminPage_POM;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
import java.time.Duration;

public class AdminTestCase {
    WebDriver driver; 6 usages
    LoginPage login; 2 usages
    AdminPage admin; 4 usages

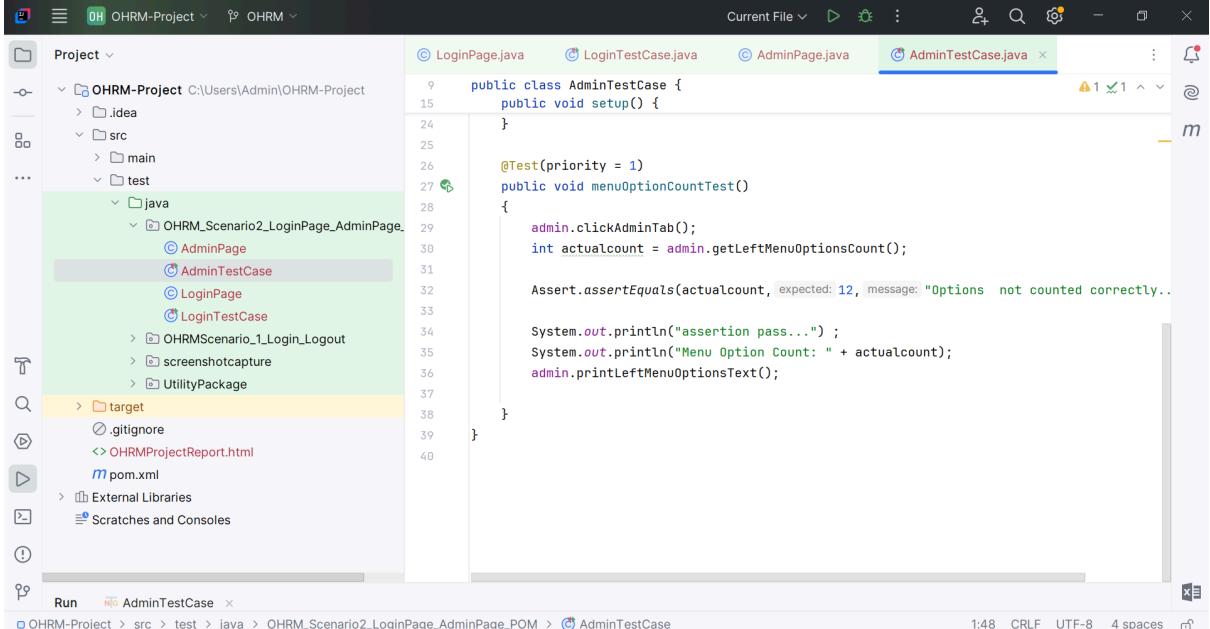
    @BeforeTest
    public void setup() {
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
        driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

        login = new LoginPage(driver);
        login.Login(username: "Admin", password: "admin123"); // Perform login first
        admin = new AdminPage(driver);
    }
}

```

2.1. Create a test case to get all 12 options from the left side menu and print the count, which should be 12 from that list. Click on Admin, and then the Admin page will open.

LeftMenuOptions Method:



The screenshot shows the IntelliJ IDEA interface with the AdminTestCase.java file open. The code is as follows:

```
public class AdminTestCase {
    public void setup() {
    }

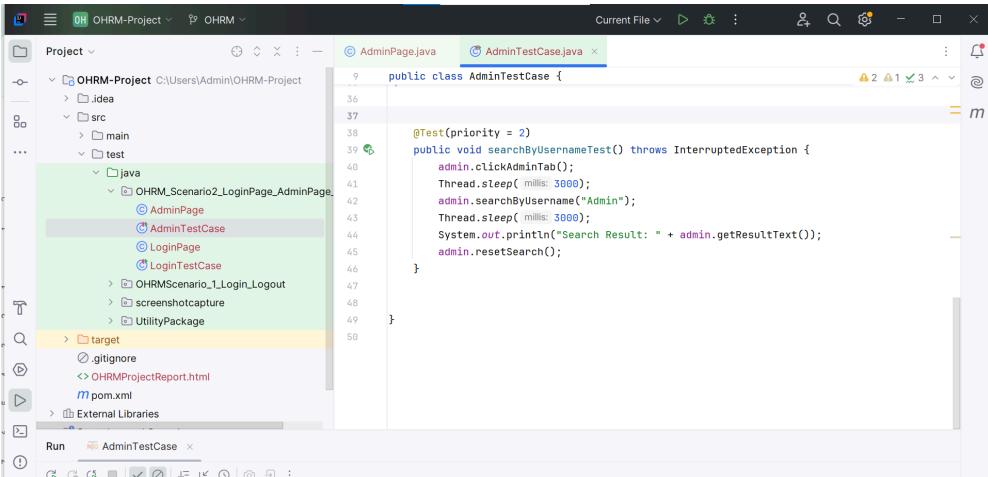
    @Test(priority = 1)
    public void menuOptionCountTest() {
        admin.clickAdminTab();
        int actualcount = admin.getLeftMenuOptionsCount();

        Assert.assertEquals(actualcount, expected: 12, message: "Options not counted correctly..");

        System.out.println("assertion pass...");
        System.out.println("Menu Option Count: " + actualcount);
        admin.printLeftMenuOptionsText();
    }
}
```

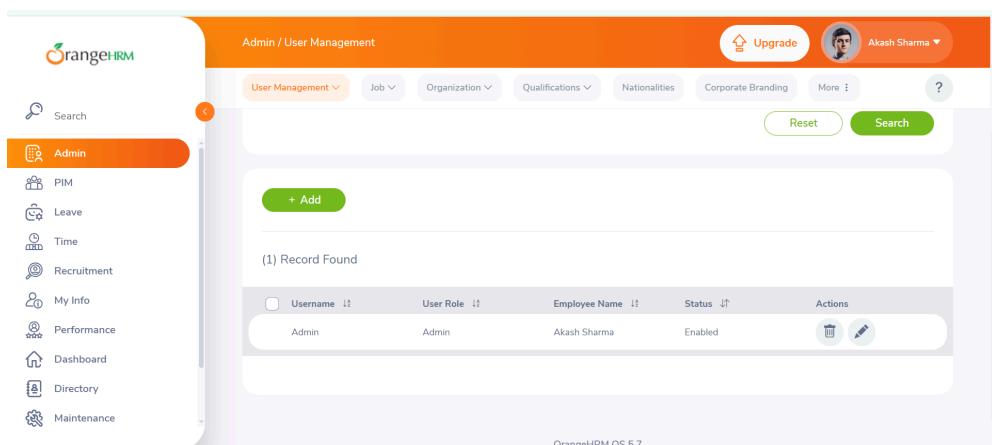
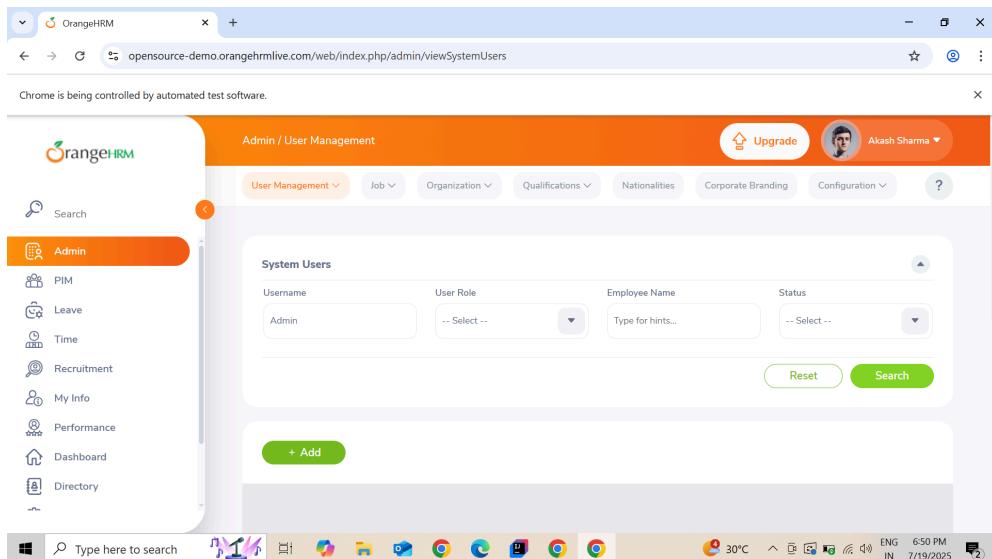
2.2. Create a test case for the search For Existing Employee search by username (): Here, send the username Admin to the username text box and click on the search button and display the total number of records found and refresh the page.

SearchbyUserName()



The screenshot shows the IntelliJ IDEA interface with the AdminTestCase.java file open. The code has been modified to include a search function:

```
public class AdminTestCase {
    public void searchByUsernameTest() throws InterruptedException {
        admin.clickAdminTab();
        Thread.sleep( millis: 3000 );
        admin.searchByUsername("Admin");
        Thread.sleep( millis: 3000 );
        System.out.println("Search Result: " + admin.getResultText());
        admin.resetSearch();
    }
}
```

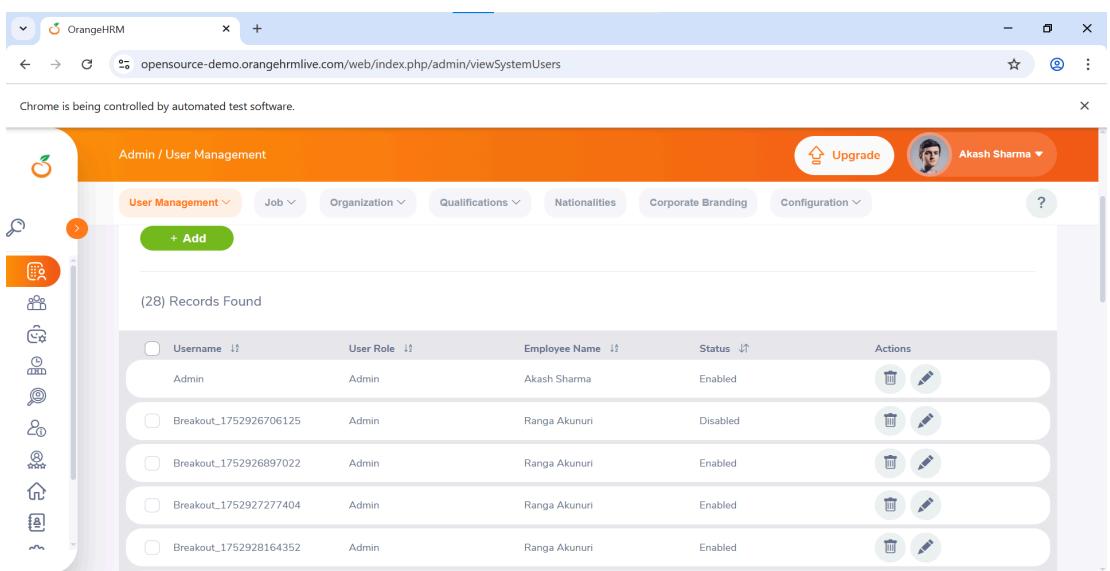
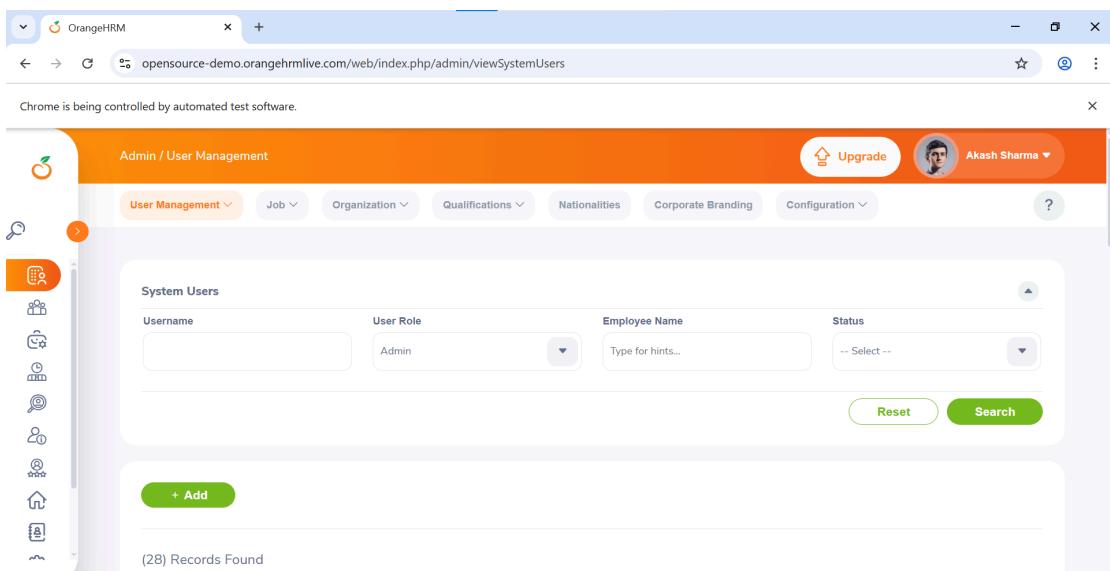


2.3. Create a test case for search For Existing Employee search ByUserRole():
 Here, automate the dropdown and select Role Admin and click on the search button and display the total record found and refresh the page.

```

public class AdminTestCase {
    @Test(priority = 3)
    public void searchByUserRoleTest() throws InterruptedException {
        admin.clickAdminTab();
        Thread.sleep( 3000 );
        admin.selectUserRole("Admin");
        Thread.sleep( 3000 );

        admin.searchButton.click();
        System.out.println("Search Result: " + admin.getResultText());
        admin.resetesearch();
    }
}
  
```



2.4. Create a test case for the search For Existing Employee search ByUserStatus(): here, automate the dropdown and select status Enabled or Disabled, then click on the search button and display the total record found.

File structure:

```

OHRM-Project
  - OHRM-Project
    - .idea
    - src
      - main
      - test
        - java
          - OHRM_Scenario2_LoginPage_AdminPage.java
          - AdminPage.java
          - AdminTestCase.java
          - LoginPage.java
          - LoginTestCase.java
          - OHRMScenario_1_LoginLogout.java
          - screenshotcapture.java
          - UtilityPackage.java
    - target
    - .gitignore
    - OHRMProjectReport.html
    - pom.xml
  - External Libraries
  - Scratches and Consoles

```

Content of AdminTestCase.java:

```

public class AdminTestCase {
    public void searchByStatusTest() {
        admin.clickAdminTab();
        admin.selectUserStatus("Enabled");
        admin.searchButton.click();
        System.out.println("Search Result: " + admin.getResultText());
        admin.resetSearch();
    }
}

```

OrangeHRM

opensource-d демо. orangehrmlive.com/web/index.php/admin/viewSystemUsers

Chrome is being controlled by automated test software.

Admin / User Management

User Management Job Organization Qualifications Nationalities Corporate Branding Configuration

Upgrade Seif user ?

System Users

Username User Role Employee Name Status

+ Add Reset Search

(11) Records Found

OrangeHRM

opensource-d демо. orangehrmlive.com/web/index.php/admin/viewSystemUsers

Chrome is being controlled by automated test software.

Admin / User Management

User Management Job Organization Qualifications Nationalities Corporate Branding Configuration

Upgrade Seif user ?

+ Add

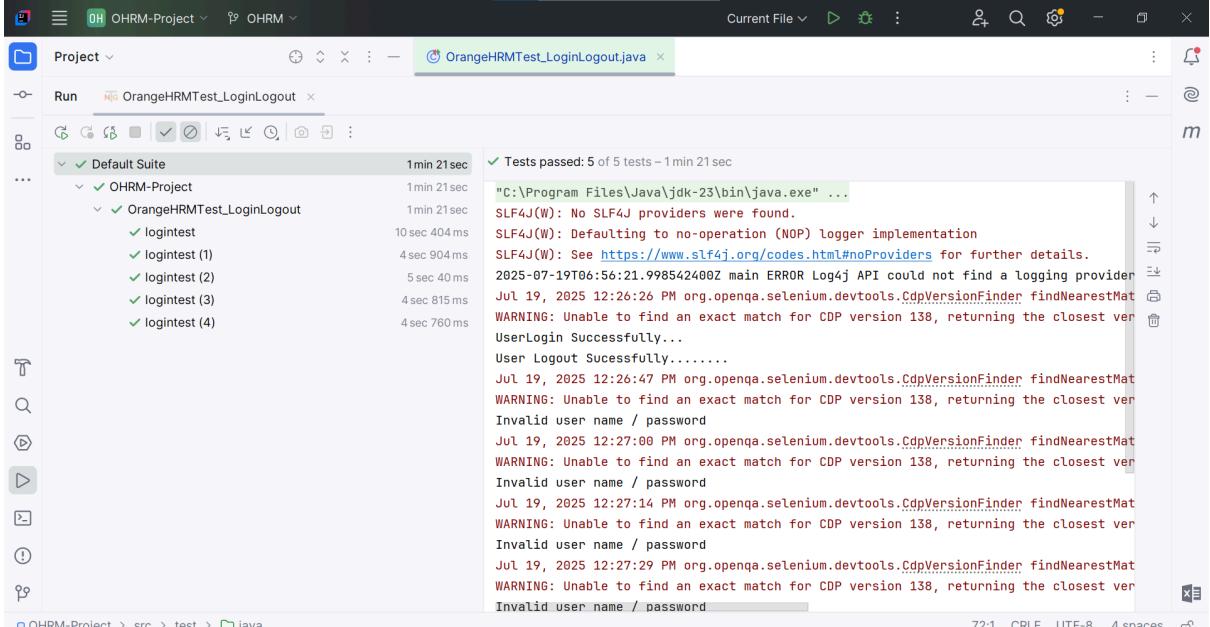
(11) Records Found

| Username | User Role | Employee Name | Status | Actions |
|------------|-----------|---------------|---------|---------|
| 68CsM | Admin | Seif user | Enabled | |
| Admin | Admin | Seif user | Enabled | |
| Dean20601 | Admin | Dean Sammie | Enabled | |
| Eddie49669 | ESS | Eddie Peyton | Enabled | |
| FMLName | ESS | Qwerty LName | Enabled | |

7. Test Result Analysis & Reporting

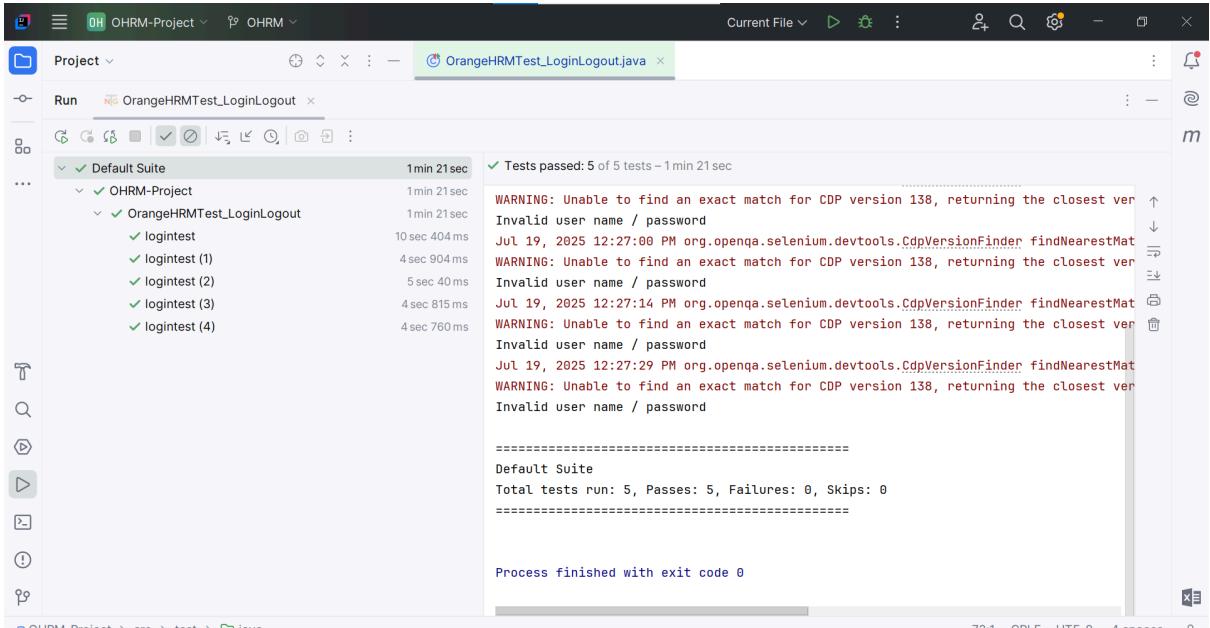
Scenario 1:

TestNG Report:



The screenshot shows the Eclipse IDE interface with the project 'OHRM-Project' selected. In the center, the 'Run' view displays the execution of 'OrangeHRMTest_LoginLogout'. The results show 5 tests passed in 1min 21sec. The log output includes several SLF4J and CdpVersionFinder warnings, and a UserLogin Successfully message.

```
Tests passed: 5 of 5 tests - 1min 21sec
C:\Program Files\Java\jdk-23\bin\java.exe" ...
SLF4J(W): No SLF4J providers were found.
SLF4J(W): Defaulting to no-operation (NOP) logger implementation
SLF4J(W): See https://www.slf4j.org/codes.html#noProviders for further details.
2025-07-19T06:56:21.998542400Z main ERROR Log4j API could not find a logging provider
Jul 19, 2025 12:26:26 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMat
WARNING: Unable to find an exact match for CDP version 138, returning the closest ver
UserLogin Successfully...
User Logout Sucessfully.....
Jul 19, 2025 12:26:47 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMat
WARNING: Unable to find an exact match for CDP version 138, returning the closest ver
Invalid user name / password
Jul 19, 2025 12:27:00 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMat
WARNING: Unable to find an exact match for CDP version 138, returning the closest ver
Invalid user name / password
Jul 19, 2025 12:27:14 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMat
WARNING: Unable to find an exact match for CDP version 138, returning the closest ver
Invalid user name / password
Jul 19, 2025 12:27:29 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMat
WARNING: Unable to find an exact match for CDP version 138, returning the closest ver
Invalid user name / password
```



This screenshot shows the same Eclipse IDE setup, but the log output for the test run is significantly shorter and cleaner, indicating a successful execution.

```
Tests passed: 5 of 5 tests - 1min 21sec
=====
Default Suite
Total tests run: 5, Passes: 5, Failures: 0, Skips: 0
=====

Process finished with exit code 0
```

ExtentReports:

http://localhost:63342/OHRM-Project/OHRMProjectReport.html?_jtt=vihatgdgeq50inlj30f0g_cjkl&_ij_reload=RELOAD_ON_SAVE#

Tests

- Login Test: Admin / admin123
12:26:35 PM / 00:00:06:013 Pass
- Login Test: Ritik / ritik123
12:26:54 PM / 00:00:04:903 Fail
- Login Test: ritu / ritu456
12:27:08 PM / 00:00:05:037 Fail
- Login Test: rohan / rohan123
12:27:22 PM / 00:00:04:812 Fail
- Login Test: mohit / mohit@834
12:27:36 PM / 00:00:04:755 Fail

Login Test: Admin / admin123

07.19.2025 12:26:35 PM | 07.19.2025 12:26:41 PM | 00:00:06:013 | #test-id=1

Status | Timestamp | Details

Pass | 12:26:41 PM | Login successful for valid credentials.

Started 19-July-2025 | Ended 19-July-2025

Tests Passed: 1 | Tests Failed: 4

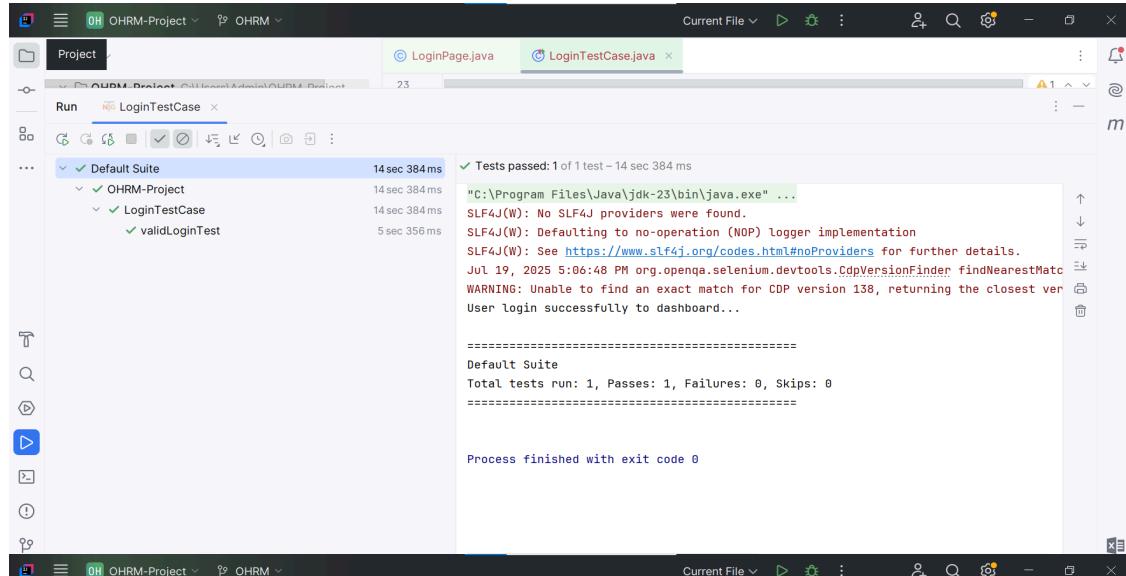
Log events

Timeline

| System/Environment | Value |
|------------------------|---------------|
| Host | Lenovo |
| DeviceName | Lenovo |
| Generation | Core-i3-10Gen |
| Operating System | Windows 10 |
| Browser | Chrome |
| TesterName | Ritik Rajati |
| Application Under Test | OrangeHRM |

Scenario 2:

TestNG report: 1.LoginPageTest

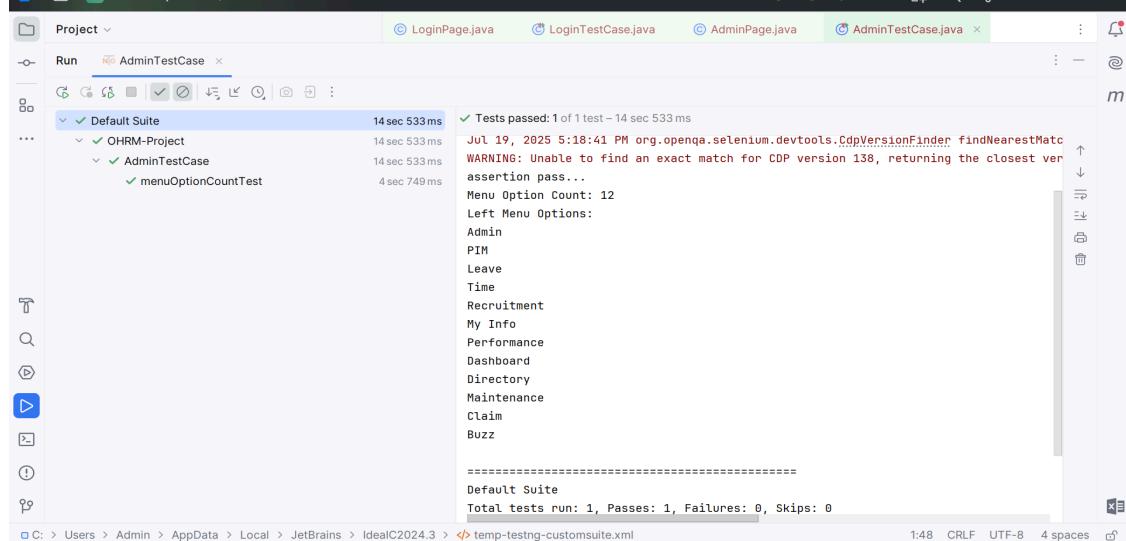


The screenshot shows the TestNG report for the LoginPageTest. It displays a single test named 'validLoginTest' under the 'Default Suite' which passed in 14 sec 384 ms. The output log shows the Java command, SLF4J provider information, and the successful login message.

```
C:\Program Files\Java\jdk-23\bin\java.exe" ...
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
Jul 19, 2025 5:06:48 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 138, returning the closest version
User login successfully to dashboard...

=====
Default Suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
=====

Process finished with exit code 0
```

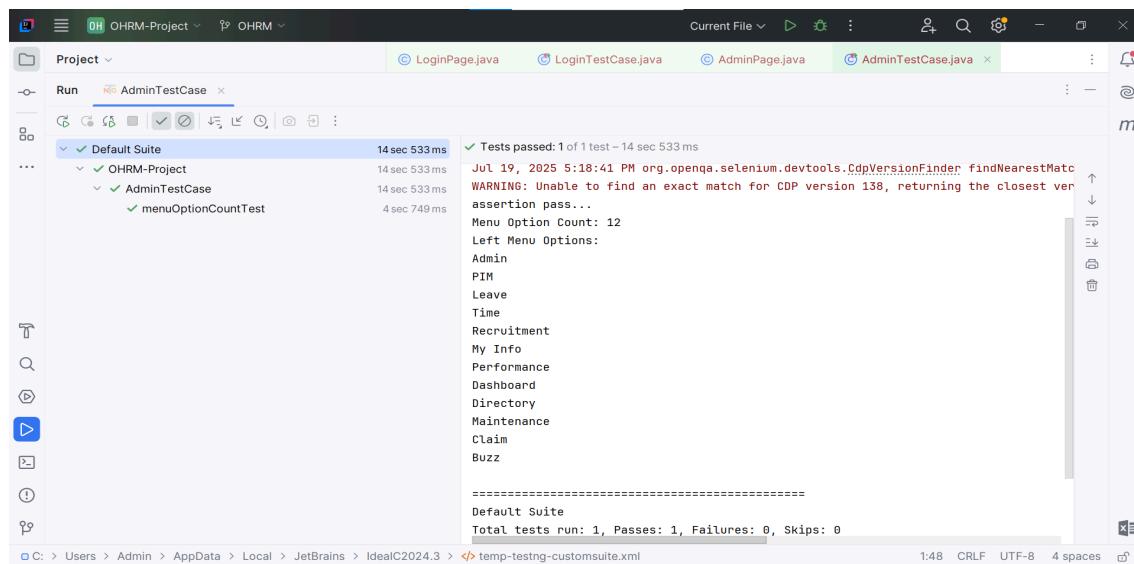



The screenshot shows the TestNG report for the AdminTestCase. It displays a single test named 'menuOptionCountTest' under the 'Default Suite' which passed in 14 sec 533 ms. The output log shows the Java command, SLF4J provider information, and a detailed list of menu options: Admin, PIM, Leave, Time, Recruitment, My Info, Performance, Dashboard, Directory, Maintenance, Claim, and Buzz.

```
Jul 19, 2025 5:18:41 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 138, returning the closest version
assertion pass...
Menu Option Count: 12
Left Menu Options:
Admin
PIM
Leave
Time
Recruitment
My Info
Performance
Dashboard
Directory
Maintenance
Claim
Buzz

=====
Default Suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
```

TestNG report: 2.1 LeftMenuOptions



The screenshot shows the TestNG report for the AdminTestCase. It displays a single test named 'menuOptionCountTest' under the 'Default Suite' which passed in 14 sec 533 ms. The output log shows the Java command, SLF4J provider information, and a detailed list of menu options: Admin, PIM, Leave, Time, Recruitment, My Info, Performance, Dashboard, Directory, Maintenance, Claim, and Buzz.

```
Jul 19, 2025 5:18:41 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 138, returning the closest version
assertion pass...
Menu Option Count: 12
Left Menu Options:
Admin
PIM
Leave
Time
Recruitment
My Info
Performance
Dashboard
Directory
Maintenance
Claim
Buzz

=====
Default Suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
```

TestNG report: 2.2 searchByUserName()

The screenshot shows the IntelliJ IDEA interface with the 'Run' tool window open. The 'Default Suite' section lists a single test case, 'searchByUserNameTest', which passed. The output pane displays the test results:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" ...
SLF4J(W): No SLF4J providers were found.
SLF4J(W): Defaulting to no-operation (NOP) logger implementation
SLF4J(W): See https://www.slf4j.org/codes.html#noProviders for further details.
Jul 19, 2025 6:49:56 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 138, returning the closest version
Search Result: (1) Record Found

=====
Default Suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
=====

Process finished with exit code 0
```

TestNG report: 2.3 searchByUserRole()

The screenshot shows the IntelliJ IDEA interface with the 'Run' tool window open. The 'Default Suite' section lists a single test case, 'searchByUserRoleTest', which passed. The output pane displays the test results:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" ...
SLF4J(W): No SLF4J providers were found.
SLF4J(W): Defaulting to no-operation (NOP) logger implementation
SLF4J(W): See https://www.slf4j.org/codes.html#noProviders for further details.
Jul 19, 2025 7:58:29 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 138, returning the closest version
Search Result: (28) Records Found

=====
Default Suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
=====

Process finished with exit code 0
```

TestNG report: 2.4 searchByUserRole()

The screenshot shows the IntelliJ IDEA interface with the 'Run' tool window open. The 'Default Suite' section lists a single test case, 'searchByStatusTest', which passed. The output pane displays the test results:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" ...
SLF4J(W): No SLF4J providers were found.
SLF4J(W): Defaulting to no-operation (NOP) logger implementation
SLF4J(W): See https://www.slf4j.org/codes.html#noProviders for further details.
Jul 19, 2025 8:14:16 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 138, returning the closest version
Search Result: (11) Records Found

=====
Default Suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
=====

Process finished with exit code 0
```

8. Conclusion

The automation of OrangeHRM using the **Page Object Model (POM)** framework and **TestNG** has successfully validated critical functional components of the application. The testing effort focused on two primary modules—**Login** and **Admin**, which are fundamental for any HRMS user flow.

In **first scenario** the login functionality was automated with valid and invalid credentials (Admin/admin123). The test script successfully located and interacted with the username, password, and login button fields.

In the **second scenario**, I implemented POM for LoginPage and AdminPage. I tested below four key functionalities:

1. All 12 options from the left-side navigation menu were identified and counted.
2. The username field accepted input dynamically. Search was executed for an existing user (**Admin**), and results were displayed correctly.
3. The User Role dropdown was handled programmatically. The script selected the **Admin** role and successfully retrieved all associated users.
4. The Status dropdown was accessed, and both values (Enabled, Disabled) were tested . Results displayed the correct status-filtered data.

----- END -----