

Test Case Day 9

PFA day 9 test case, which includes DB validations

Process	Steps To Execute	Expected Results
Delete an user	1. Navigate to Systems section, shipping Option 2. Choose Users 3. Delete an user	Verify the deleted user is not present on the database.

BaseTest.java program

```
package com.ibm.test;
```

```
import static org.testng.Assert.assertEquals;
```

```
import static org.testng.Assert.assertTrue;
```

```
import java.io.IOException;
```

```
import java.sql.SQLException;
```

```
import java.util.ArrayList;
```

```
import java.util.HashMap;
```

```
import java.util.concurrent.TimeUnit;
```

```
import org.junit.Assert;
```

```
import org.openqa.selenium.By;
```

```
import org.openqa.selenium.JavascriptExecutor;
```

```
import org.openqa.selenium.WebDriver;
```

```
import org.openqa.selenium.WebElement;
```

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

```
import org.openqa.selenium.firefox.FirefoxDriver;
```

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

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

```
import org.testng.annotations.AfterMethod;
```

```
import org.testng.annotations.BeforeMethod;
```

```
import org.testng.annotations.BeforeSuite;
import org.testng.annotations.BeforeTest;
//import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;
```

```
import com.ibm.pages.AdminPage;
import com.ibm.utilities.DbConnectionUser;
//import com.ibm.utilities.ExcelUtil;
import com.ibm.utilities.PropertiesFileHandler;
```

```
public class BaseTest {
    WebDriver driver;
    WebDriverWait wait;
    PropertiesFileHandler propFileHandler;
    HashMap<String, String> data;
    //Use properties to solve the simple test cases.
    @BeforeSuite
    public void testcase1() throws IOException
    {
        //Locating property file
        String file="./TestData/data.properties";
        propFileHandler = new PropertiesFileHandler();
        data= propFileHandler.getPropertiesAsMap(file);
    }

    @BeforeMethod
    public void Initialization()
    {
```

```

//Open chrome and wait
System.setProperty("webdriver.gecko.driver","./drivers/geckodriver.exe");
//System.setProperty("webdriver.chrome.driver","./drivers/chromedriver.exe");
//driver=new ChromeDriver();
driver=new FirefoxDriver();
wait=new WebDriverWait(driver,60);
driver.manage().window().maximize();
driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

}

```

@AfterMethod

```

public void closeBrowser() {
    driver.quit();
}

```

```

@Test(priority=9)
public void DeleteUserFromUserTable() throws IOException,
InterruptedException, SQLException
{
    //using data from property file
    String url = data.get("url");
    String userName=data.get("username");
    String passWord=data.get("password");
    driver.get(url);

    AdminPage adminPage=new AdminPage(driver, wait);

    adminPage.enterEmail(userName);
    adminPage.enterPassword(passWord);

    adminPage.CaptureScreenshot(driver);

    adminPage.clickOnLogin();
    adminPage.CaptureScreenshot(driver);

    adminPage.systemSection();
    adminPage.ClickOnUserTab();
    Thread.sleep(1000);
    adminPage.CaptureScreenshot(driver);
}

```

```

        adminPage.DeleteUser();

        Thread.sleep(1000);
        adminPage.CaptureScreenshot(driver);
        adminPage.ClickOnDeleteConfirm();

        DbConnectionUser dbConnect=new DbConnectionUser(driver,
wait);

        dbConnect.DbUserDelete();

    }
}

```

AdminPage.java program

```
package com.ibm.pages;
```

```
import static org.testng.Assert.assertEquals;
```

```
import static org.testng.Assert.fail;
```

```
import java.io.File;
```

```
import java.io.FileInputStream;
```

```
import java.io.FileNotFoundException;
```

```
import java.io.IOException;
```

```
import java.util.Date;
```

```
import java.util.List;
```

```
import java.util.Properties;
```

```
import java.util.concurrent.TimeUnit;
```

```
import org.apache.commons.io.FileUtils;
```

```
import org.openqa.selenium.By;
```

```
import org.openqa.selenium.JavascriptExecutor;
```

```
import org.openqa.selenium.Keys;
```

```
import org.openqa.selenium.OutputType;
```

```
import org.openqa.selenium.TakesScreenshot;
```

```

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.Select;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;
import org.testng.annotations.Test;

import com.ibm.test.LoginPage;

public class AdminPage {

    //xpath for locating webelements in webpage
    @FindBy(name="email")
    WebElement emailAdd;

    @FindBy(name="pword")
    WebElement enterPass;

    @FindBy(xpath="//button[@class='btn btn-labeled btn-info m-b-5']")
    WebElement loginEle;

    @FindBy(xpath="//*[text()=' Catalog']")
    WebElement catalogEle;

```

```
@FindBy(xpath="//*[text()=' Categories']")
```

```
WebElement categorySel;
```

```
@FindBy(xpath="//table[@id='dataTableExample2']/tbody/tr[10]/td[1]")
```

```
WebElement tablelastColValue;
```

```
@FindBy(xpath="//*[@aria-controls='dataTableExample2']")[16]")
```

```
WebElement pageTwoButton;
```

```
@FindBy(xpath="//*[@aria-controls='dataTableExample2']")[18]")
```

```
WebElement nextButtonInPage;
```

```
@FindBy(xpath="//*[@aria-controls='dataTableExample2']")[14]")
```

```
WebElement previousButtonInPage;
```

```
@FindBy(xpath="//*[text()=' Banners']")
```

```
WebElement bannersSel;
```

```
@FindBy(xpath="//a[@title='Add New']")
```

```
WebElement addNewButton;
```

```
@FindBy(xpath="//button[@type='Submit']")
```

```
WebElement clickOnSave;
```

```
@FindBy(name="name")
```

```
WebElement bannerNameField;
```

```
@FindBy(xpath="//*[text()=' Reports']")
```

```
WebElement clickOnReports;
```

```
@FindBy(xpath="//*[text()=' Reports']")
```

```
WebElement clickOnReturnReports;
```

```
@FindBy(name="report")
```

```
WebElement selectDropDown;
```

```
@FindBy(xpath="//span[@class='input-group-addon']")
```

```
WebElement clickOnFilter;
```

```
@FindBy(xpath="//table[@id='dataTableExample2']/tbody/tr")
```

```
List<WebElement> selectReportTable;
```

```
@FindBy(xpath="//table[@id='dataTableExample2']/tbody/tr[1]/td")
```

```
WebElement tableData;
```

```
@FindBy(xpath="//*[text()=' System']")
```

```
WebElement systemSection;
```

```
@FindBy(xpath="//*[text()=' Weight Class']")
```

```
WebElement weightClass;
```

```
@FindBy(xpath="//*[@class='caret']")[5]")
```

```
WebElement clickOnAction;
```

```
@FindBy(xpath="//*[@title='Edit']")[5]")
```

```
WebElement clickOnEdit;
```

```
@FindBy(xpath="//input[@name='name']")
```

```
WebElement editWeightName;
```

```
@FindBy(xpath="//input[@name='unit']")
```

```
WebElement editUnitName;
```

```
@FindBy(xpath="//*[@name='status']")
```

```
WebElement editStatus;
```

```
@FindBy(xpath="//button[@title='Save']")
```

```
WebElement clickOnSavebuttonInSettings;
```

```
@FindBy(xpath="//*[@class='material-ripple']")[17]")
```

```
WebElement clickOnSystem;
```

```
@FindBy(xpath="//*[text()=' Shipping']")
```

```
WebElement clickOnShipping;
```

```
@FindBy(xpath="//*[text()=' Settings']")
```

```
WebElement clickOnSettings;
```

```
@FindBy(xpath="//input[@id='store_name']")
```

```
WebElement storeName;
```

```
@FindBy(xpath="//div[@class='alert alert-success alert-dismissible']")
```

```
WebElement successMsgText;
```

```
@FindBy(xpath="//*[text()=' Statistics']")
```

```
WebElement clickonStatistics;
```



```
@FindBy(xpath="//table[@class='table table-bordered table-striped table-hover']/tbody/tr[5]/td[2]")
```

```
WebElement totalProducts;
```

```
@FindBy(xpath="//*[text()=' Products']")
```

```
WebElement clickOnProducts;
```

```
@FindBy(xpath="//*[@class='caret']")[1]")
```

```
WebElement clickOnActions;
```

```
@FindBy(xpath="//*[text()='Delete']")[7]")
```

```
WebElement clickOnDeleteUnderActions;
```

```
@FindBy(xpath="//button[@class='confirm']")
```

```
WebElement clickOnConfirmDelete;
```

```
@FindBy(id="pro_name")
```

```
WebElement productName;
```

```
@FindBy(xpath="//a[text()='Data']")
```

```
WebElement clickOnDatalink;
```

```
@FindBy(xpath="//a[text()='Link']")
```

```
WebElement clickOnlink;
```

```
@FindBy(id="tabs")
```

```
WebElement selectTabs;
```

```
@FindBy(xpath="//*[text()=' Tabs']")
```

```
WebElement clickOnTabs;
```

```
@FindBy(id="categories")
```

```
WebElement selectCategories;
```

```
@FindBy(xpath="//a[text()='Image']")
```

```
WebElement clickOnImage;
```

```
@FindBy(xpath="//input[@id='pro_image']")
```

```
WebElement selectProductImage;
```

```
@FindBy(xpath="//button[@type='Submit']")
```

```
WebElement clickOnSaveButton;
```

```
@FindBy(xpath="//*[text()=' Users']")
```

```
WebElement clickOnUserTab;
```

```
public void ClickOnUserTab()
```

```
{
```

```
    clickOnUserTab.click();
```

```
}
```

```
WebDriverWait wait;
```

```
WebDriver driver;
```

```
JavascriptExecutor js=(JavascriptExecutor)driver;
```

```
public AdminPage(WebDriver driver, WebDriverWait wait)
```

```

{
    PageFactory.initElements(driver, this);

    this.driver=driver;

    this.wait=wait;
}

//Enter email address
public void enterEmail(String username)
{
    emailAdd.sendKeys(username);
}

//To enter password
public void enterPassword(String password)
{
    enterPass.sendKeys(password);
}

//click on Login button
public void clickOnLogin()
{
    loginEle.click();
}

//Capture screen shot method.. including date in between filename
public void CaptureScreenshot(WebDriver driver) throws IOException
{
    File file=((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

    //Created date method to include date in screen shot file name
    Date date=new Date();

```

```

        //Changing : into - to understand easily and seperate day-month-year
        String currentDate=date.toString().replace(":", "-");

        FileUtils.copyFile(file, new File("./ProjectOneSS/LoginPage-"+currentDate+".jpg"));
    }

    //Selecting catalog
    public void Catalog()
    {
        catalogEle.click();
    }

    //Selecting category
    public void selCategory()
    {
        categorySel.click();
    }

    //Table ColValue from Last row
    public String tableValue()
    {
        return tablelastColValue.getText();
    }

    //Method to verify Page 2 navigation button is displayed and working or not
    public void pageTwoNavi()
    {
        if(pageTwoButton.isDisplayed() && pageTwoButton.isEnabled())
        {
            System.out.println("Page navigation is displayed and enabled");
            pageTwoButton.click();
        }
    }

```

```

    }
    else
    {
        //Condition to fail test case if next page number 2 not displayed or not enabled
        Assert.fail("Page Navigation is failed");
    }
}

//Method to verify Next Button is displayed and working or not
public void nextButton()
{
    if(nextButtoninPage.isDisplayed() && nextButtoninPage.isEnabled())
    {
        System.out.println("Next Button is displayed and Enabled");
        nextButtoninPage.click();
    }
    else
    {
        //Condition to fail test case if Next button not displayed or not enabled
        Assert.fail("Next Button is disabled");
    }
}

//Method to verify Previous button is displayed and working or not
public void previousButton()
{
    if(previousButtonInPage.isDisplayed() && previousButtonInPage.isEnabled())
    {
        System.out.println("Previous Button is displayed and Enabled");
    }
}

```

```

        previousButtonInPage.click();
    }
    else
    {
        //Condition to fail test case if Previous button not displayed or not enabled
        Assert.fail("Previous button is disabled");
    }
}

//To click on Banners on LHS
public void clickOnBanners()
{
    bannersSel.click();
}

//To click on Add new button in Banners
public void clickOnAddNewButton()
{
    addNewButton.click();
}

//Click on Save button
public void clickOnSaveButton()
{
    clickOnSave.click();
}

//Verifying validation message
public void validationMessage()

```

```

{

    String sText=js.executeScript("return
document.getElementsByName('name')[\"0\"].validationMessage").toString();

    System.out.println(sText);

}

//Enter in Banner Name field
public void bannerName()
{
    bannerNameField.sendKeys("BannerTestCase");
}

public void ClickOnTabs()
{
    clickOnTabs.click();
}

//Click on Reports link
public void clickOnReports()
{
    clickOnReports.click();
}

//Click on reports under Reports
public void returnReports()
{
    clickOnReturnReports.click();
}

//Select Value from Drop down

```

```

public void selectDropDown()
{
    Select select=new Select(selectDropDown);
    select.selectByVisibleText("Returns Report");
}

/*//Click on Filters
public void clickOnFilter()
{
    clickOnFilter.click();
}*/
//Select Table
public void selectReportTable()
{
    int rowsInTable=selectReportTable.size();
    System.out.println("Rows in Table - "+rowsInTable);
}

//print Table data
public void tableData()
{
    String firstRowValue=tableData.getText();
    System.out.println("Table value "+firstRowValue);
}

//To click on System
public void systemSection()
{
    systemSection.click();
}

```



```
//To click on Weight class
```

```
public void weightClass()  
{  
    weightClass.click();  
}
```

```
//Click on Action tab
```

```
public void ClickOnAction()  
{  
    clickOnAction.click();  
}
```

```
//Click on Edit option
```

```
public void ClickOnEdit()  
{  
    clickOnEdit.click();  
}
```

```
//Edit Weight Name
```

```
public void EditWeightName()  
{  
    editWeightName.clear();  
    editWeightName.sendKeys("Kilograms");  
}
```

```
//Edit Unit Name
```

```
public void EditUnitName()  
{
```

```

        editUnitName.clear();

        editUnitName.sendKeys("1");
    }

    //Select Status
    public void SelectStatus()
    {
        Select selDropValue=new Select(editStatus);

        selDropValue.selectByValue("1");
    }

    //Click on Save button
    /*public void ClickOnSaveButton()
    {
        clickOnSaveButton.click();
    }*/

    //loop to validate presence of Edit weight Name in Weight class
    public void verifyWeightName()
    {
        int rowsInWeightClassTable=selectReportTable.size();

        String weightNameInWeightClass=null;

        String statusFromWeightClass=null;

        for(int i=1;i<=rowsInWeightClassTable;i++)
        {
            WebElement
weightName=driver.findElement(By.xpath("//table[@id='dataTableExample2']/tbody/tr['+i+']/td[2]"));

            WebElement
statusInWeightClass=driver.findElement(By.xpath("//table[@id='dataTableExample2']/tbody/tr['+i+']/td[4]"));

```

```

        weightNameInWeightClass= weightName.getText();
        statusFromWeightClass=statusInWeightClass.getText();

        if(weightNameInWeightClass.equals("Kilograms"))
        {
            break;
        }
    }

    //Confirming edited weightName
    if(weightNameInWeightClass.equals("Kilograms") &&
statusFromWeightClass.equals("Enabled"))
    {
        System.out.println("WeightName successfully edited");
    }
    else
    {
        //If Weight Name is not edited fail the test case
        Assert.fail("WeightName is not Edited in Weight Class screen");
    }
}

```

```

//Click on System tab on LHS
public void ClickOnSystem()
{
    clickOnSystem.click();
}

```

```
//Click on Shipping
```

```
public void ClickOnShipping()
```

```
{
```

```
    clickOnShipping.click();
```

```
}
```

```
//Click on Settings on LHS
```

```
public void ClickOnSettings()
```

```
{
```

```
    clickOnSettings.click();
```

```
}
```

```
//Edit store name in Edit Settings
```

```
public void StoreNameEdit(String newStoreName)
```

```
{
```

```
    storeName.clear();
```

```
    storeName.sendKeys(newStoreName);
```

```
}
```

```
//Click on Save button
```

```
public void ClickOnSaveButtonInSettings()
```

```
{
```

```
    clickOnSavebuttonInSettings.click();
```

```
}
```

```
//Validation on edit Store name success message
```

```
public void SuccessMsgTest(String editStoreNameMsg)
```

```
{
```

```
    String storeNameEdited=successMsgText.getText();
```

```
Assert.assertTrue(storeNameEdited.contains(editStoreNameMsg),"Assertion on edit  
Store Name");
```

```
}
```

```
public void clickOnStatistics()
```

```
{
```

```
    clickonStatistics.click();
```

```
}
```

```
public int totalProductsInStatistics()
```

```
{
```

```
    String totalProductsInTable=totalProducts.getText();
```

```
    int prods=Integer.parseInt(totalProductsInTable);
```

```
    //System.out.println("Total Products "+prods);
```

```
    return prods;
```

```
}
```

```
public void ClickOnProducts()
```

```
{
```

```
    clickOnProducts.click();
```

```
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("dataTableExample2")));
```

```
}
```

```
public void ClickOnActions()
```

```
{
```

```
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("dataTableExample2")));
```

```
    clickOnActions.click();
```

```
}
```

```
public void ClickonDeleteUnderActions()
```

```

{
    clickOnDeleteUnderActions.click();
}

public void ClickOnDeleteConfirm()
{
    clickOnConfirmDelete.click();
}

public void ProductName()
{
    productName.sendKeys("Fresh Badham");
}

public void MetaTitle()
{
    metaTitle.sendKeys("Fresh");
}

public void ClickOnDataLink()
{
    Actions action=new Actions(driver);
    action.moveToElement(clickOnDatalink).click().build().perform();
    //clickOnDatalink.click();
    //js.executeScript("arguments[0].click();",clickOnDatalink);
}

@FindBy(id="meta_title")
WebElement metaTitle;

```

```
@FindBy(id="model")
```

```
WebElement model;
```

```
@FindBy(id="gst")
```

```
WebElement gst;
```

```
@FindBy(id="price")
```

```
WebElement price;
```

```
@FindBy(id="special_dis")
```

```
WebElement specialDiscount;
```

```
public void Model()
```

```
{
```

```
    model.sendKeys("Fresh");
```

```
}
```

```
public void GstValue()
```

```
{
```

```
    gst.sendKeys("18");
```

```
}
```

```
public void PriceValue()
```

```
{
```

```
    price.sendKeys("1000");
```

```
}
```

```
public void SpecialDiscount()
```

```
{
```

```
    specialDiscount.sendKeys("0");
```

```
}
```

```
public void ClickOnlinktab()
```

```

{
    clickOnLink.click();
}

public void ClickOnImage()
{
    clickOnImage.click();
}

public int tableDataInProductsTab()
{
    int tableRows=selectReportTable.size();
    int productPriceVal=0;
    for(int i=1;i<=tableRows;i++)
    {
        WebElement
productName=driver.findElement(By.xpath("//table[@id='dataTableExample2']/tbody/tr["+i+"]/td[2]"));
        String productNameText=productName.getText();
        if(productNameText.equals("Fresh Badham"))
        {
            WebElement
productPrice=driver.findElement(By.xpath("//table[@id='dataTableExample2']/tbody/tr["+i+"]/td[5]"));
            String priceOfProduct=productPrice.getText().trim().replace(",","")
            .replace(".00", "").replace("181", "");

            productPriceVal=productPriceVal+Integer.parseInt(priceOfProduct.trim());

            break;
        }
        else

```



```

        {
            Assert.fail("Unable to locate Fresh Badham product in list");
        }

    }

    return productPriceVal;

}

public void selectTabValue()
{
    Select selectTabValue=new Select(selectTabs);
    selectTabValue.selectByVisibleText("Test01");
}

public void selProductImage()
{
    selectProductImage.sendKeys("C:\\Users\\IBM_ADMIN\\eclipse-
workspace\\FrameworkProjectone\\TestData\\BadhamFrs.JPG");
}

public void SelectCategory()
{
    Select selectCatgory=new Select(selectCategories);
    selectCatgory.selectByVisibleText("Grains");
}

@FindBy(id="search-box")
WebElement searchBoxEle;

```

```

        @FindBy(xpath="//img[@src='https://atozgroceries.com/assets/uploads/products/BadhamFrs.JPG']")
        WebElement searchSelectEle;

        public void SearchBox() throws InterruptedException
        {
            searchBoxEle.click();

            Thread.sleep(1000);
        }

        public void SearchSelectEle()
        {
            searchSelectEle.click();
        }

        /*public void SearchSelectEle()
        {
            int products=searchSelectEle.size();
            for(int j=1;j<=products;j++)
            {
                WebElement freshBadhamPro=driver.findElement(By.xpath("//*[@class='pro-
name']["+j+"]"));

                String badhamProName=freshBadhamPro.getText();

                if(badhamProName.equals("Fresh Badham"))
                {

                    driver.findElement(By.xpath("//img[@src='https://atozgroceries.com/assets/uploads/products/
BadhamFrs.JPG']")).click();

                    break;
                }
                else
                {

```

```

        Assert.fail();
    }
}*/

/*@FindBy(xpath="//*[text()='Fresh Badham']")[1]")
WebElement selectFreshBadhamProduct;

public void SelectFreshBadhamProduct() throws InterruptedException
{
    wait.until(ExpectedConditions.elementToBeSelected(selectFreshBadhamProduct));
    selectFreshBadhamProduct.click();
    Thread.sleep(1000);
}*/

@FindBy(xpath="//input[@type='text']")[2]")
WebElement searchEleInUser;

public void SearchEleInUserscreen()
{
    searchEleInUser.sendKeys("CategoryTest");
    Actions action=new Actions(driver);
    action.sendKeys(Keys.DOWN).build().perform();

}

@FindBy(xpath="//*[@class='product-cost']")[1]")
WebElement selectDiscountPrice;

public int DiscountPriceOfProduct()
{
    String priceofProuct=selectDiscountPrice.getText().trim().replace("₹","").replaceAll(",","");
}

```

```

        int productPrice=Integer.parseInt(priceofProuct.trim().replace("1000.00",
"1000").replace("Discounted price: ", ""));

        System.out.println(productPrice);

        return productPrice;

    }

    public void checkForCategory() throws InterruptedException
    {

        //int rowsInCategory=selectReportTable.size();

        String CategoryName=null;
        String statusOfCategory=null;

        for(int i=1;i<=10;i++)
        {

            WebElement
CateName=driver.findElement(By.xpath("//table[@id='dataTableExample2']/tbody/tr["+i+"]/td[2]"));

            CategoryName= CateName.getText();

            if(CategoryName.equals("CategoryTest"))
            {

                WebElement
statusIncategory=driver.findElement(By.xpath("//table[@id='dataTableExample2']/tbody/tr["+i+"]/td[5]
"));

                statusOfCategory=statusIncategory.getText();

                if(statusOfCategory.equals("Enabled"))
                {

```

```

        driver.findElement(By.xpath("//span[@class='caret']["+i+"]")).click();
        driver.findElement(By.xpath("//a[@title='Edit']["+i+"]")).click();
        Thread.sleep(2000);
        WebElement statusSel=driver.findElement(By.name("status"));
        Select selectStatus=new Select(statusSel);
        selectStatus.selectByValue("0");
        break;
    }

    else
    {
        Assert.fail("unable to find category");
    }
}

}

}

}

public void checkForCategoryUpdate() throws InterruptedException
{
    int rowsInCategory=selectReportTable.size();

    for(int i=1;i<=rowsInCategory;i++)
    {
        WebElement
CateName=driver.findElement(By.xpath("//table[@id='dataTableExample2']/tbody/tr["+i+"]/td[2]"));

        String categoryName= CateName.getText();

```

```

        if(CategoryName.equals("CategoryTest"))
        {
            WebElement
statusIncategory=driver.findElement(By.xpath("//table[@id='dataTableExample2']/tbody/tr["+i+"]/td[5]
"));

            String statusOfCategory=statusIncategory.getText();
            if(statusOfCategory.equals("Disabled"))
            {
                System.out.println("Category updated successfully");
                break;

            }
            else
            {
                Assert.fail("Category not updated");
            }
        }

    }

}

public void DeleteUser() throws InterruptedException
{
    int rowsInUsers=selectReportTable.size();

    System.out.println(rowsInUsers);

    String userNameInTable=null;

```

```

String userNameAction=null;

for(int i=1;i<=rowsInUsers;i++)
{
    WebElement
userName=driver.findElement(By.xpath("//table[@id='dataTableExample2']/tbody/tr["+i+"]/td[2]"));

    userNameinTable= userName.getText();

    if(userNameinTable.equals("TestDB"))
    {
        WebElement
userAction=driver.findElement(By.xpath("//table[@id='dataTableExample2']/tbody/tr["+i+"]/td[3]"));

        userNameAction=userAction.getText();

        if(userNameinTable.equals("TestDB") &&
userNameAction.equals("Enabled"))
        {

            driver.findElement(By.xpath("//span[@class='caret']["+i+"]")).click();

            driver.findElement(By.xpath("//*[text()='Delete']["+i-
1)+"]")).click();

            Thread.sleep(2000);

            break;
        }
        else
        {
            Assert.fail("Unable to Delete user");
        }
    }
}

```

```

    }

    }

}

```

DbConnectionUser.java program

```

package com.ibm.utilities;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;

public class DbConnectionUser {

    WebDriverWait wait;
    WebDriver driver;

    public DbConnectionUser(WebDriver driver, WebDriverWait wait)
    {
        PageFactory.initElements(driver, this);
        this.driver=driver;
        this.wait=wait;
    }

    public void DbUserDelete() throws SQLException {

        //connection string - jdbc(odbc):server://host:port/db_name, username,
password
        Connection c =
DriverManager.getConnection("jdbc:mysql://foodsonfinger.com:3306/foodsonfinger_atozgr
oceries",
                            "foodsonfinger_atoz", "welcome@123");
        Statement s = c.createStatement();
        /*ResultSet rs = s.executeQuery("SELECT * from as_banner");
        while(rs.next())
        {
            System.out.println(rs.getInt("banner_id"));
            System.out.print("\t"+rs.getString("name"));

```



```

    */
    ResultSet rs=s.executeQuery("SELECT * from as_user where
username=\"TestDB\"");
    while(rs.next())
    {
        System.out.print(rs.getInt("user_id"));
        System.out.println("\t"+rs.getString("username"));
    }
    Assert.assertFalse(rs.next());
}
}

```

Property file

```

url=https://atozgroceries.com/admin
username=demo@atozgroceries.com
password=456789
bannertitle=Banner | Admin Panel - Powered By A&S
editStoreNameMsg=Success: You have successfully updated Store!
newStoreName=ZtoA groceries Store

```

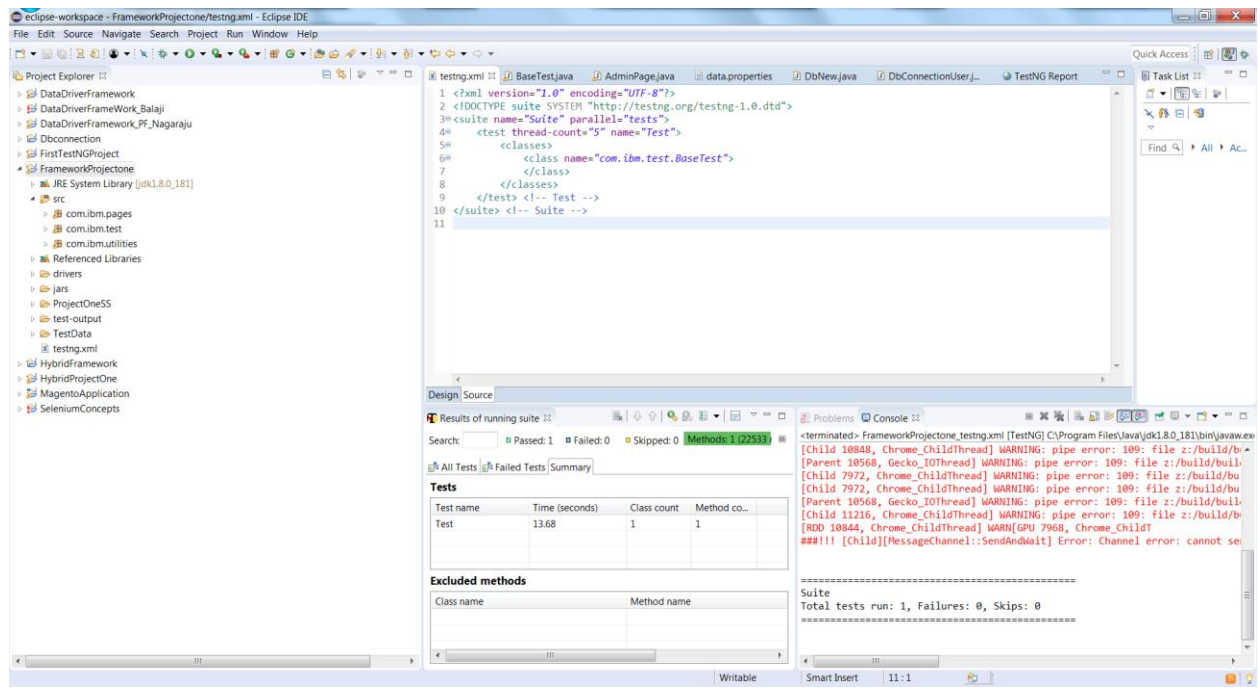
TestNG.xml file

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Suite" parallel="tests">
    <test thread-count="5" name="Test">
        <classes>
            <class name="com.ibm.test.BaseTest">
            </class>
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->

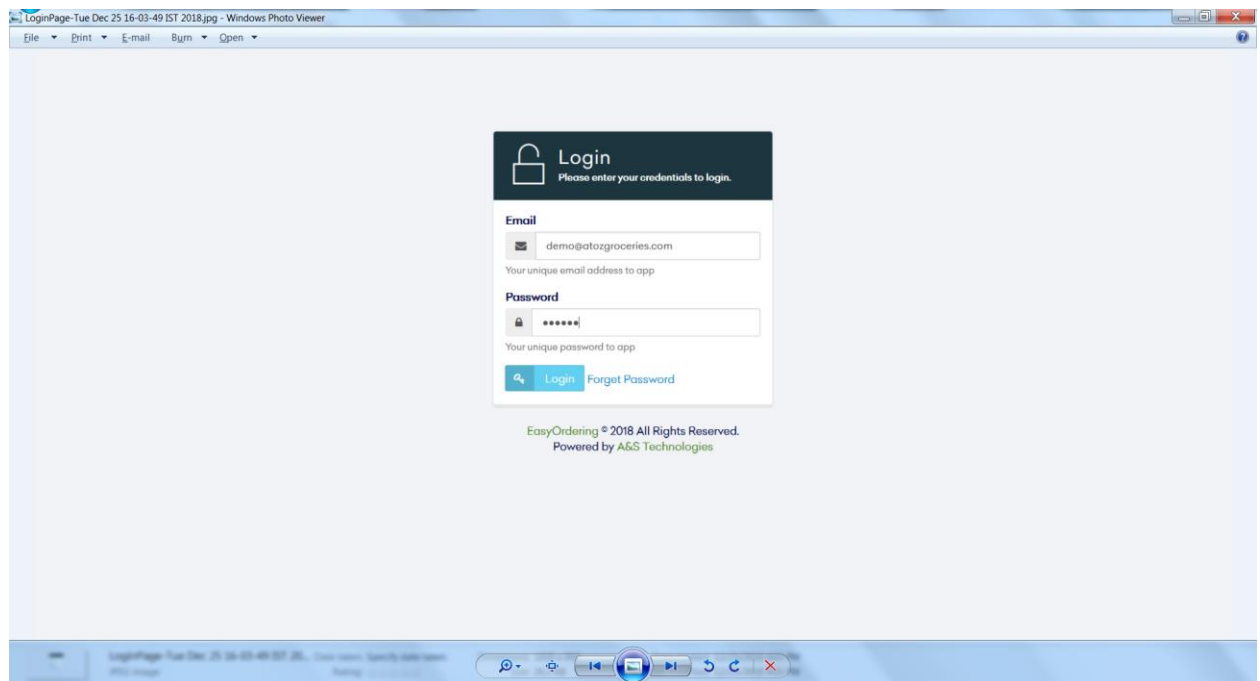
```

Eclipse Output screen shot

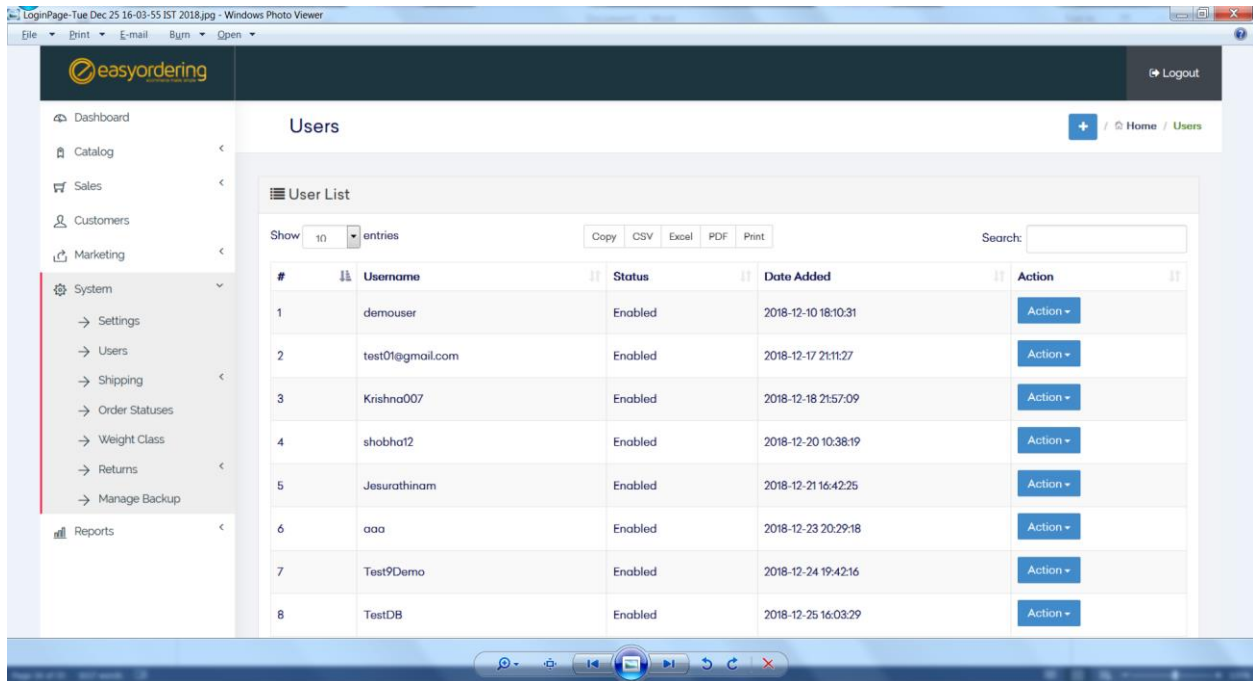


Browser screen shots

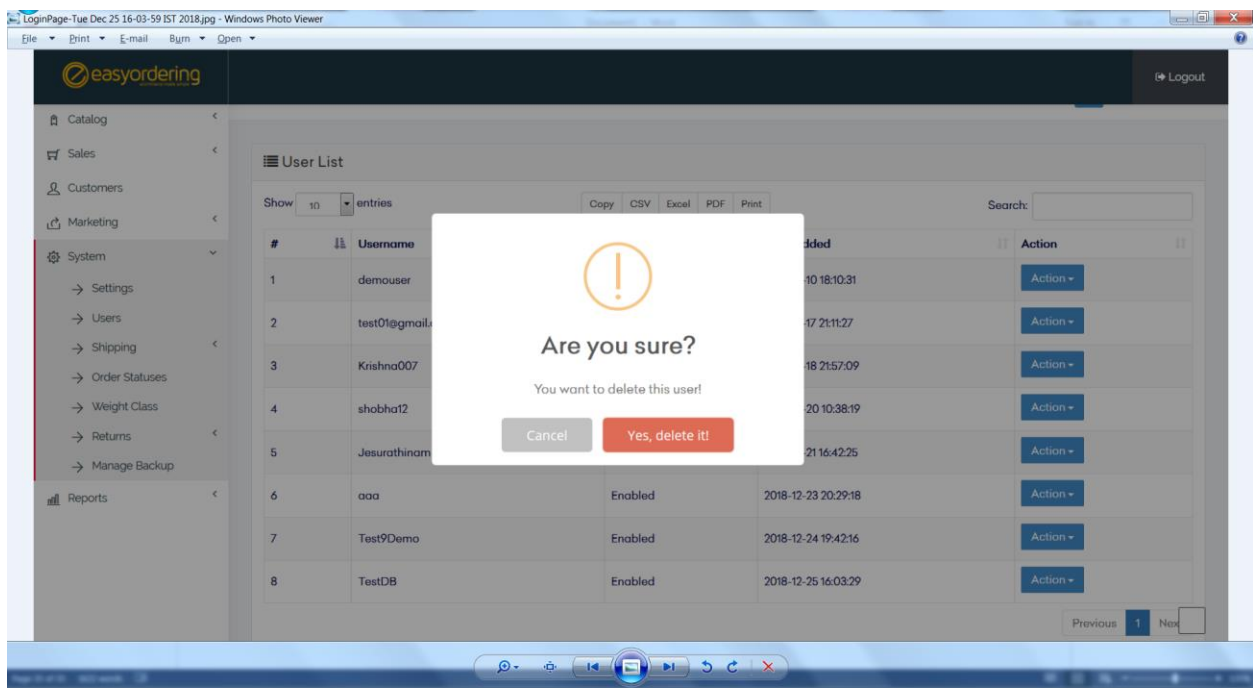
Login page



User page



Deleting TestDB username



Emilable report

Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded Groups
Suite						
Test	1	0	0	22,273		

Class	Method	Start	Time (ms)
Suite			
Test — passed			
com.ibm.test.BaseTest	DeleteUserFromUserTable	1545734028198	13680

Test

com.ibm.test.BaseTest#DeleteUserFromUserTable

[back to summary](#)

Index.html report

Test results

1 suite

All suites

Suite

Info

- C:\Users\IBM_ADMIN\eclipse-workspace\FrameworkProjectone\testng.xml
- 1 test
- 0 groups
- Times
- Reporter output
- Ignored methods
- Chronological view

Results

- 1 method, 1 passed
- Passed methods (hide)
 - ☒ DeleteUserFromUserTable

☒ com.ibm.test.BaseTest

DeleteUserFromUserTable

Test Case Pass

