

Day 2 - Test case solution document

Code Used:

BaseTest.java

```
package com.ibm.test;

import java.io.File;
import java.io.IOException;
import java.util.Date;
import org.apache.commons.io.FileUtils;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.testng.Assert;
import org.testng.annotations.Test;
import com.ibm.pages.CatalogPage;
import com.ibm.pages.CategoryPage;
import com.ibm.pages.DriverLaunch;
import com.ibm.pages.LoginPage;
import com.ibm.pages.TabsPage;

public class BaseTest extends DriverLaunch
{
    @Test(testName="NegativeTestCaseValidation")
    public void DayTwoTestCase() throws IOException
    {
        //Creating the objects for the pages
        LoginPage login=new LoginPage(driver,wait);
        CatalogPage catalog=new CatalogPage(driver,wait);
        TabsPage tab=new TabsPage(driver,wait);
        try
        {
            //getting the data from the properties file using Hash map
            String url=hashData.get("url");
            String username=hashData.get("username");
            String password=hashData.get("password");
            String tagreqMsg=hashData.get("tagrequiredmsg");
            String selectReqMsg=hashData.get("tagSelectReqMsg");
            String tagname=hashData.get("tagname");
            String tagsort=hashData.get("tagsort");
            String tagstatus=hashData.get("tagstatus");

            //navigating to the Admin URL of atozgroceries
            driver.navigate().to(url);

            //login page is used to provide the input of username and
            password
            login.EnterEmail(username);
            login.EnterPassword(password);
            login.clickLogin();

            //In catalog page we will find the catalog element and click on
            that to open Tabs
        }
    }
}
```

```

        catalog.OpenCatalog();

        //In Tab page, we will click on tabs using openTabs()
        tab.openTabs();

        //In Tab page we will click on "+" to add new Tab
        tab.addNewTab();

        //Providing input values for Sort Order and Status for validating
        the "Tab Name" as mandatory field.
        String validMsg=tab.reqFieldNameValidation(tagsort,tagstatus);
        Assert.assertTrue(validMsg.contains(tagreqMsg), "Assertion on Tag
Name as mandatory field");

        //Providing input values for Tag Name and Status for validating
        the "Sort Order" as mandatory field
        validMsg=tab.reqFieldSortValidation(tagname,tagstatus);
        Assert.assertTrue(validMsg.contains(tagreqMsg), "Assertion on
Sort Order as mandatory field ");

        //Providing input values for Tag Name and Sort Order for
        validating the "Status" as mandatory field
        validMsg=tab.reqFieldStatusValidation(tagname,tagsort);
        Assert.assertTrue(validMsg.contains(selectReqMsg), "Assertion on
Status as mandatory field ");

    }
    catch(Exception e)// catching if any exception occurs and takes a
    screenshot of the page
    {
        TakesScreenshot ts=(TakesScreenshot) driver;
        File file = ts.getScreenshotAs(OutputType.FILE);
        Date date = new Date();
        String currentDate = date.toString().replace(":", "-");
        FileUtils.copyFile(file, new
        File("./Screenshots/Day2_TestCase_Error_" + currentDate + ".jpg"));
        System.out.println(e.getMessage());
        Assert.fail();
    }
    finally // regardless of error this block will execute to logout the
page
    {
        login.ClickLogout();
    }
}

```

LoginPage.java

```

package com.ibm.pages;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

```

```

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

public class LoginPage {
    @FindBy(name="email")
    WebElement emailEle;

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

    @FindBy(xpath="/html/body/div/div/div/div[2]/form/button")
    WebElement loginEle;

    @FindBy(xpath="//a[@title='Logout']")
    WebElement logoutEle;

    WebDriverWait wait;
    WebDriver driver;

    public LoginPage(WebDriver driver, WebDriverWait wait)
    {
        PageFactory.initElements(driver, this);
    }
    public void EnterEmail(String username)
    {
        emailEle.sendKeys(username);
    }
    public void EnterPassword(String password)
    {
        passwordEle.sendKeys(password);
    }
    public void clickLogin()
    {
        loginEle.click();
    }
    public void ClickLogout()
    {
        logoutEle.click();
    }
}

```

CatalogPage.java

```

package com.ibm.pages;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
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.WebDriverWait;

public class CatalogPage {

    @FindBy(partialLinkText="Catalog")

```

```

WebElement catalogEle;

WebDriver driver;
WebDriverWait wait;

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

public void OpenCatalog()
{
    wait.until(ExpectedConditions.visibilityOf(catalogEle));
    catalogEle.click();
}

}

```

TabsPage.java

```

package com.ibm.pages;

import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.ui.Select;
import org.openqa.selenium.support.ui.WebDriverWait;

public class TabsPage
{
    //Finding the WebElements
    @FindBy(xpath="/html/body/div[1]/div[1]/div/ul/li[2]/ul/li[1]/a")
    WebElement tabsEle;

    @FindBy(xpath="//a[@title='Add New']")
    WebElement addNewEle;

    @FindBy(xpath="//button[@title='Save']")
    WebElement saveEle;

    @FindBy(name="name")
    WebElement tagNameEle;

    @FindBy(name="sort")
    WebElement tagSortEle;

    @FindBy(name="status")
    WebElement tagStatusEle;

    WebDriver driver;
    WebDriverWait wait;
}

```

```

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

//openTabs method to click on Tabs
public void openTabs()
{
    tabsEle.click();
}

//addNewtab method to click on "+"
public void addNewTab()
{
    addNewEle.click();
}

//Providing input values for Sort Order,Status and using the javascript to get
the element of validation message
//and returning the returning the same
public String reqFieldNameValidation(String tagsort,String tagstatus)
{
    JavascriptExecutor js=(JavascriptExecutor) driver;
    tagSortEle.sendKeys(tagsort);
    Select tagselect=new Select(tagStatusEle);
    tagselect.selectByVisibleText(tagstatus);
    saveEle.click();
    String nameReq = js.executeScript("return
document.getElementsByName('name')[0].validationMessage").toString().trim();
    return nameReq;
}

//Providing input values for Tag Name, Status and using the javascript to get
the element of validation message
//and returning the returning the same
public String reqFieldSortValidation(String tagname,String tagstatus)
{
    JavascriptExecutor js=(JavascriptExecutor) driver;
    tagSortEle.clear();
    tagNameEle.sendKeys(tagname);
    Select tagselect=new Select(tagStatusEle);
    tagselect.selectByVisibleText(tagstatus);
    saveEle.click();
    String sortReq=js.executeScript("return
document.getElementsByName('sort')[0].validationMessage").toString().trim();
    return sortReq;
}

//Providing input values for Tag Name, Sort Order and using the javascript to
get the element of validation message
//and returning the returning the same
public String reqFieldStatusValidation(String tagname,String tagsort)

```

```

    {
        JavascriptExecutor js=(JavascriptExecutor) driver;
        Select tagselect=new Select(tagStatusEle);
        tagselect.selectByIndex(0);
        tagNameEle.sendKeys(tagname);
        tagSortEle.sendKeys(tagsort);
        saveEle.click();
        String statusReq=js.executeScript("return
document.getElementsByName('status')[0].validationMessage").toString().trim();
        return statusReq;
    }
}

```

DriverLaunch.java

```

package com.ibm.pages;

import java.io.IOException;
import java.util.HashMap;
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.ie.InternetExplorerDriver;
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.Optional;
import org.testng.annotations.Parameters;

import com.ibm.utilities.PropertiesFileHandler;

public class DriverLaunch {

    public WebDriver driver;
    public WebDriverWait wait;
    public PropertiesFileHandler propFileHandle;
    public HashMap<String, String> hashData;

    @BeforeSuite
    public void preSetForTest() throws IOException
    {
        String file=".TestData/data.properties";
        propFileHandle =new PropertiesFileHandler();
        hashData=propFileHandle.getPropertiesAsMap(file);
    }
    @BeforeMethod
    @Parameters({"browser"})
    public void initialisation(@Optional("ch")String browser)
    {
        browserInitialization(browser);
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
    }
}

```

```

        wait=new WebDriverWait(driver, 60);
    }
    @AfterMethod
    public void closingBrowser()
    {
        driver.quit();
    }

    public void browserInitialization(String browser)
    {
        switch(browser.toLowerCase())
        {
            case "ff":
                System.setProperty("webdriver.gecko.driver",
"./drivers/geckodriver.exe");
                driver=new FirefoxDriver();
                break;
            case "ch":
                System.setProperty("webdriver.chrome.driver",
"./drivers/chromedriver.exe");
                driver=new ChromeDriver();
                break;
            case "ie":
                System.setProperty("webdriver.ie.driver",
"./drivers/IEDriverServer.exe");
                driver=new InternetExplorerDriver();
                break;
        }
    }
}

```

PropertiesFileHandler.java

```

package com.ibm.utilities;

import java.io.FileInputStream;
import java.io.IOException;
import java.util.HashMap;
import java.util.Properties;
import java.util.Set;

public class PropertiesFileHandler {

    public HashMap<String, String> getPropertiesAsMap(String file) throws
IOException {
        HashMap<String, String> magentoMap = new HashMap<String, String>();

        FileInputStream fileIn = new FileInputStream(file);
        Properties prop = new Properties();
        prop.load(fileIn);

        Set<Object> keysProp = prop.keySet();
        for (Object key : keysProp) {
            magentoMap.put(key.toString(), prop.getProperty(key.toString()));
        }
    }
}

```

```

        }
        prop.clear();
        return magentoMap;
    }

}

```

data.properties

```

url=https://atozgroceries.com/admin
username=demo@atozgroceries.com
password=456789
categoryname=Milk Products
tagtitle=Milk Products
sortorder=12
status=Enabled
show=All
tagname=Milk
tagsort=10
tagstatus=Enabled
tagrequiredmsg=fill out this field
tagSelectReqMsg=select an item in the list

```

testing.xml

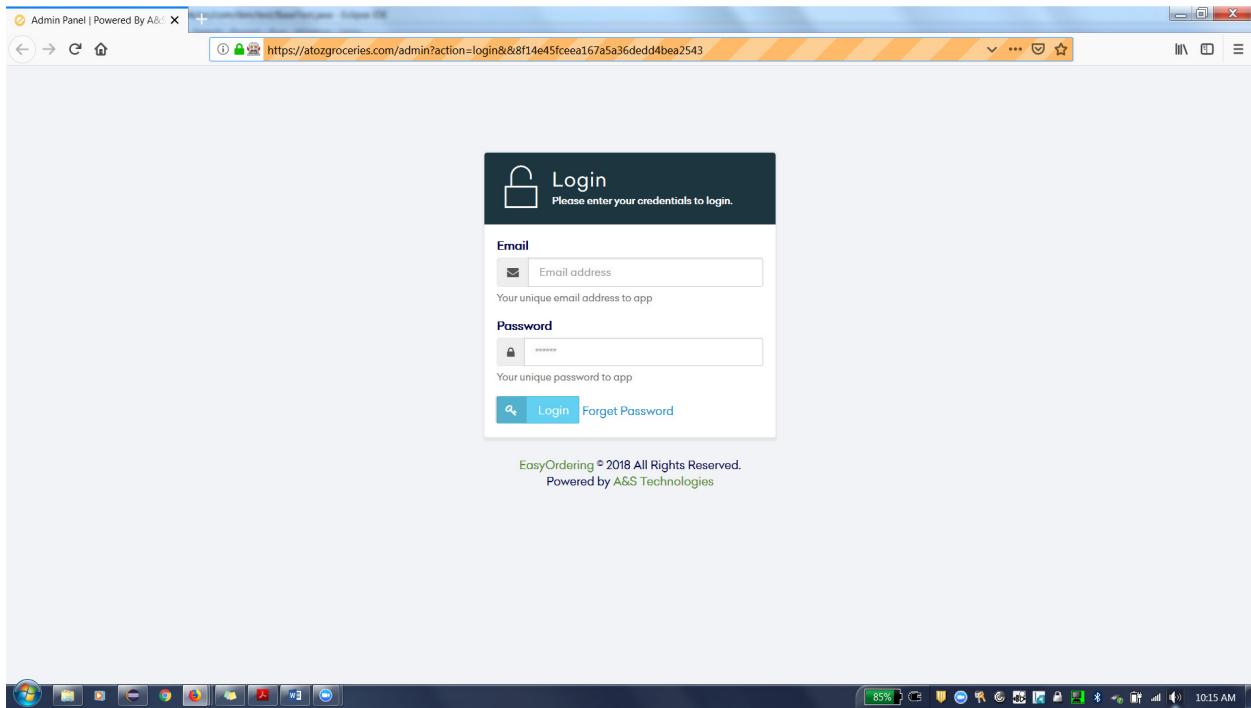
```

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

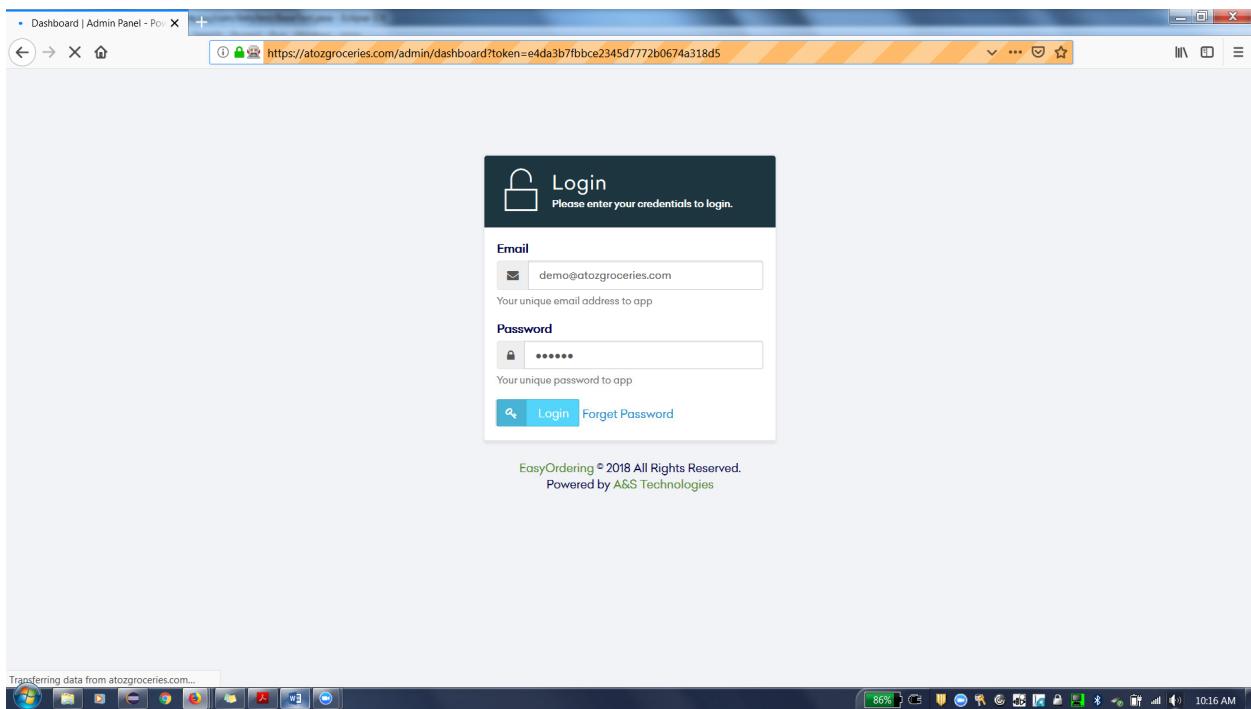
```

Step by step output:

1. Navigating to the Admin Page of “atozgroceries”



2. Providing login details



3. After logging into the webpage

The screenshot shows the Admin Panel dashboard for easyordering. The left sidebar has a 'Catalog' section expanded, showing 'Tabs', 'Banners', 'Categories', 'Products', 'Product Image', and 'Information'. Other sections like 'Sales', 'Customers', 'Marketing', 'System', and 'Reports' are also listed. The main area displays four cards: 'Total Orders' (0), 'Total Sales' (₹783.75), 'Total Customers' (0), and 'Total Products' (122). Below these is a table titled 'Latest Orders' with one row:

#	Order ID	Customer	Status	Total	Date Added	Action
1	#1		Pending	₹1,425.00	2018-12-10	

4. Opening Catalog

The screenshot shows the Admin Panel dashboard after opening the Catalog. The left sidebar now has 'Catalog' expanded, showing 'Tabs', 'Banners', 'Categories', 'Products', 'Product Image', and 'Information'. The main area displays updated statistics: 'Total Orders' (1), 'Total Sales' (₹1,425.00), 'Total Customers' (0), and 'Total Products' (224). The 'Latest Orders' table remains the same.

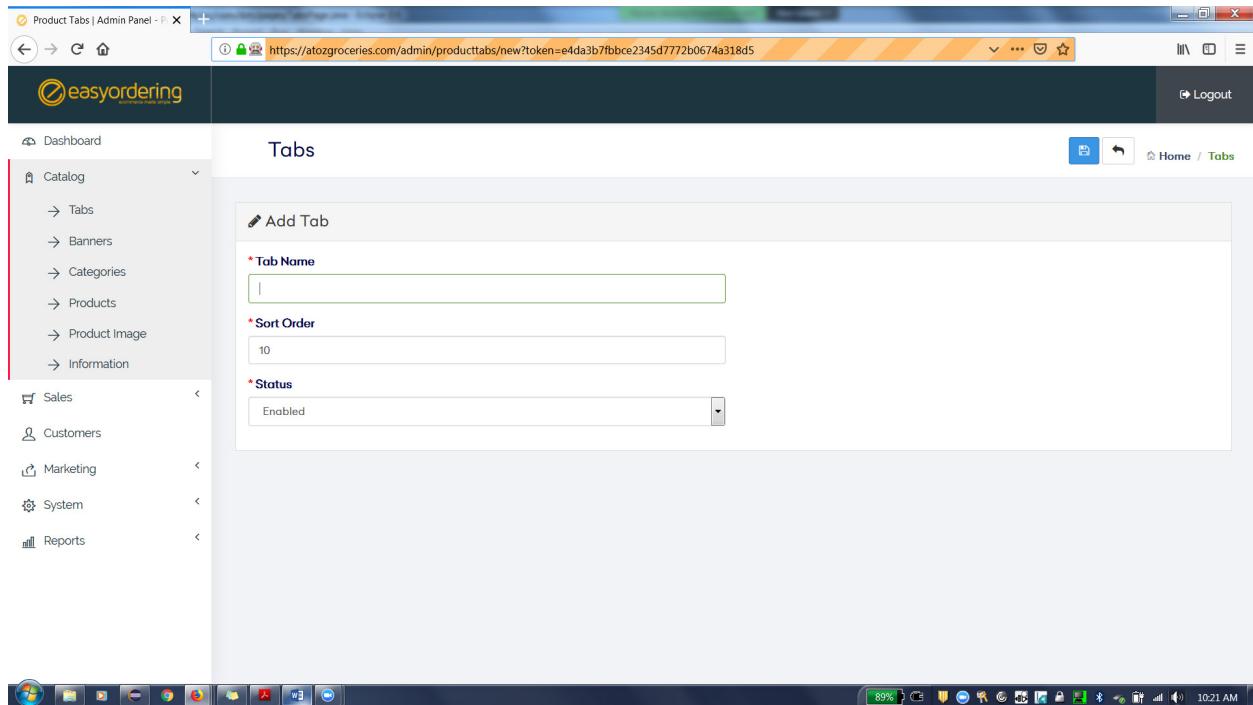
5. Clicking on Tabs

The screenshot shows the 'Product Tabs | Admin Panel' interface. The left sidebar has a 'Catalog' section with 'Tabs' selected, and other sections like Sales, Customers, Marketing, System, and Reports. The main content area is titled 'Tabs' and contains a 'Tab List' table with columns: #, Tab Name, Sort Order, Status, and Action. A message says 'No data available in table'. At the top right, there's a '+ New' button, a 'Home' link, and a 'Logout' link. The browser address bar shows the URL: https://atozgroceries.com/admin/producttabs/tab?token=c4ca4238a0b923820dcc509a6f75849b.

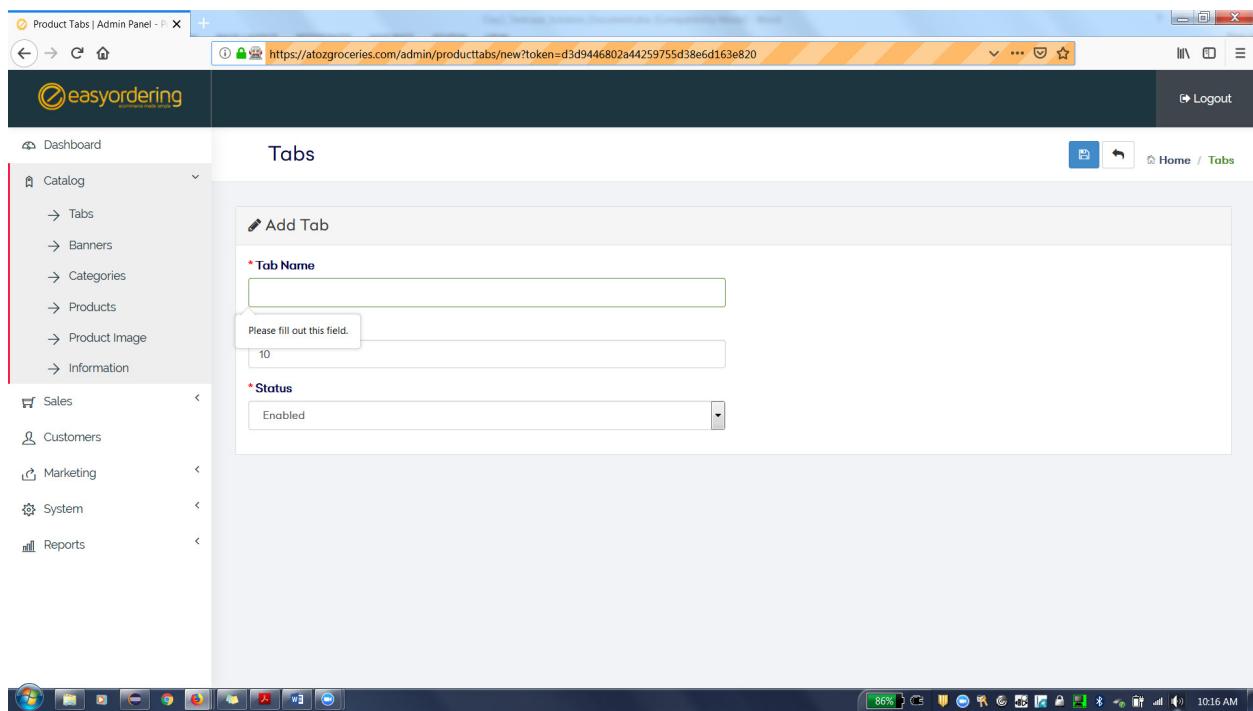
6. Clicking on “+” to add new tab

The screenshot shows the 'Product Tabs | Admin Panel' interface with the 'Catalog' section expanded, showing 'Tabs' selected. The main content area is titled 'Tabs' and contains an 'Add Tab' form with fields: 'Tab Name' (with a red asterisk), 'Sort Order' (with a red asterisk), and 'Status' (with a red asterisk). The browser address bar shows the URL: https://atozgroceries.com/admin/producttabs/new?token=cfcd208495d565ef66e7dff9f98764da.

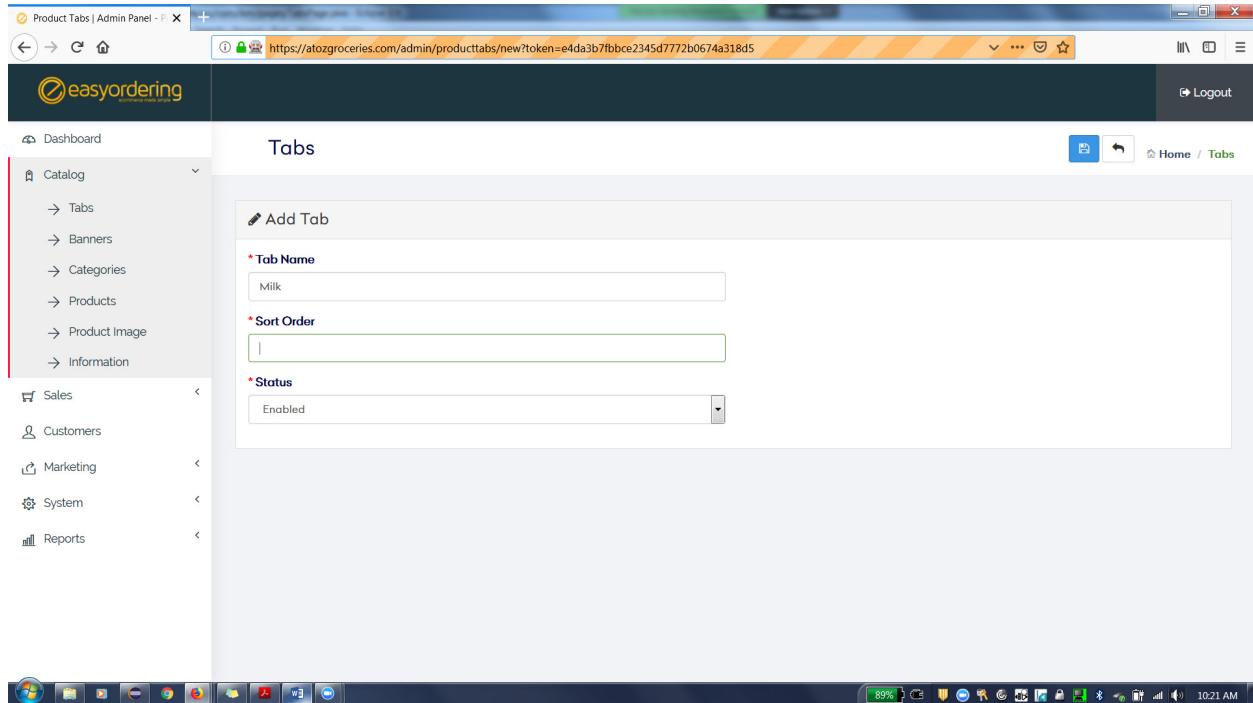
7. Providing input values for Sort Order and Status for validating the “Tab Name” as mandatory field.



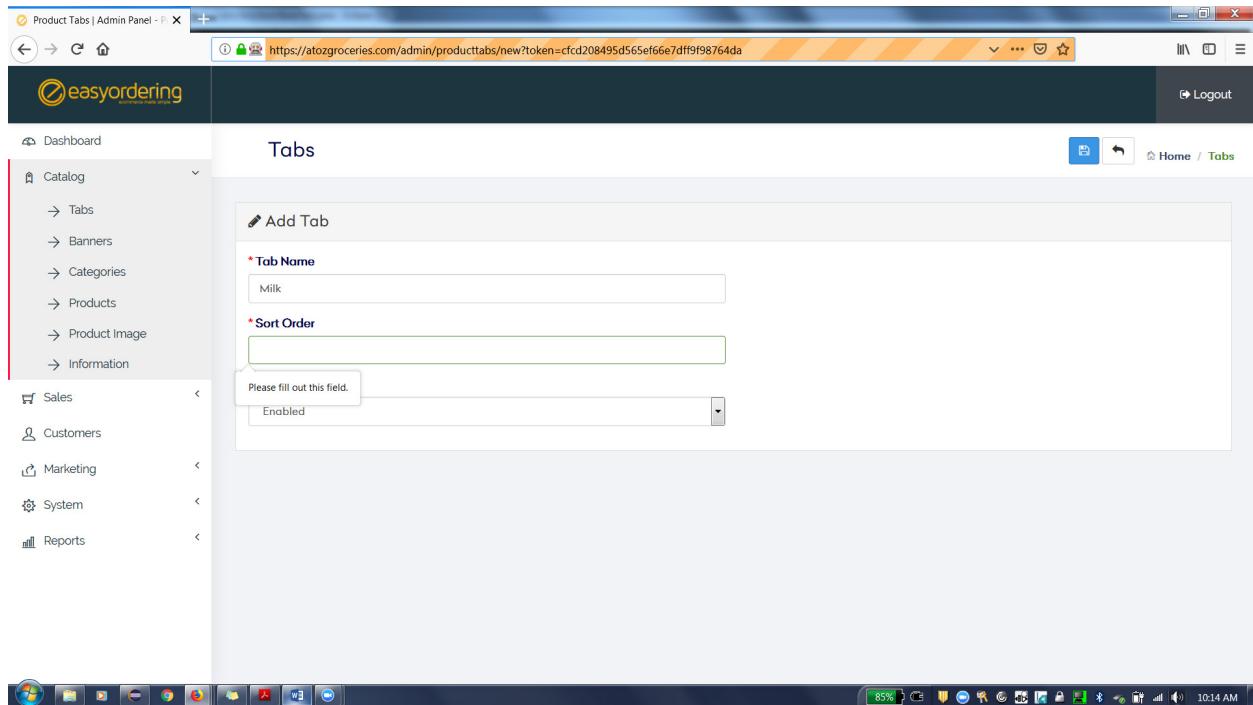
8. After clicking on “Save”, a popup message is showing up on name – “Please fill out this field”



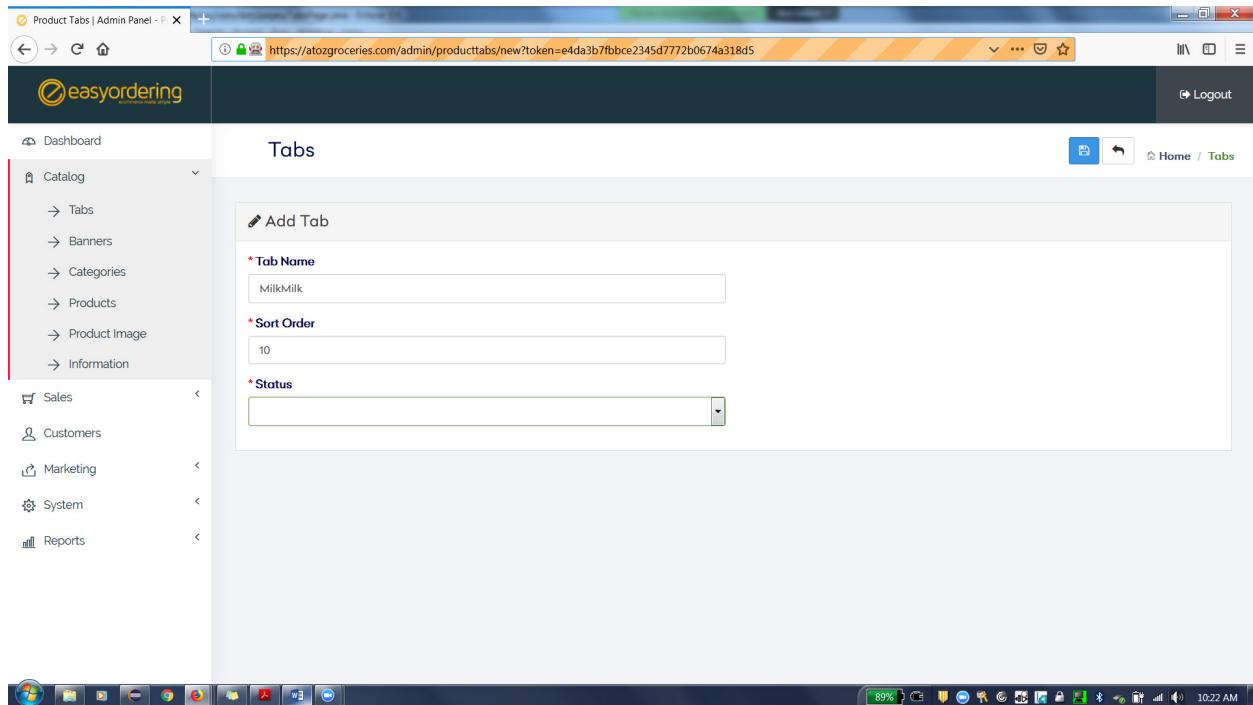
9. Providing input values for Tag Name and Status for validating the “Sort Order” as mandatory field.



10. After clicking on “Save”, a popup message is showing up on sort order – “Please fill out this field”

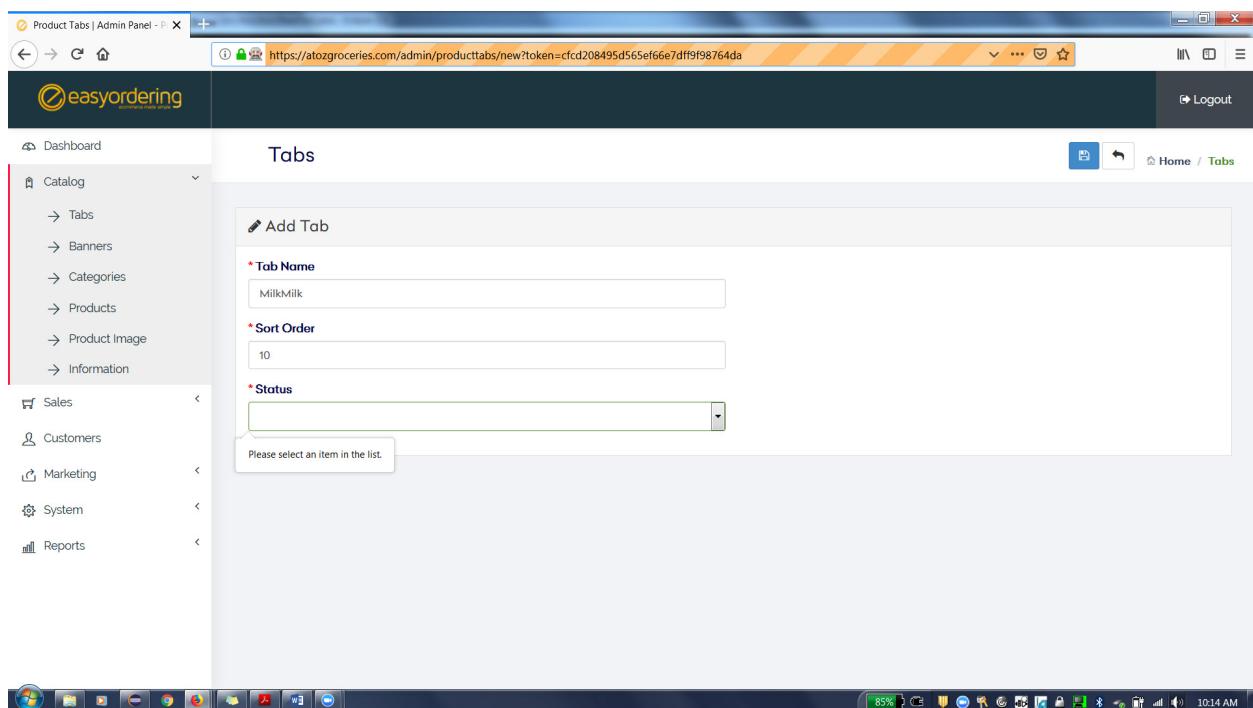


11. Providing input values for Tag Name and Sort Order for validating the “Status” as mandatory field



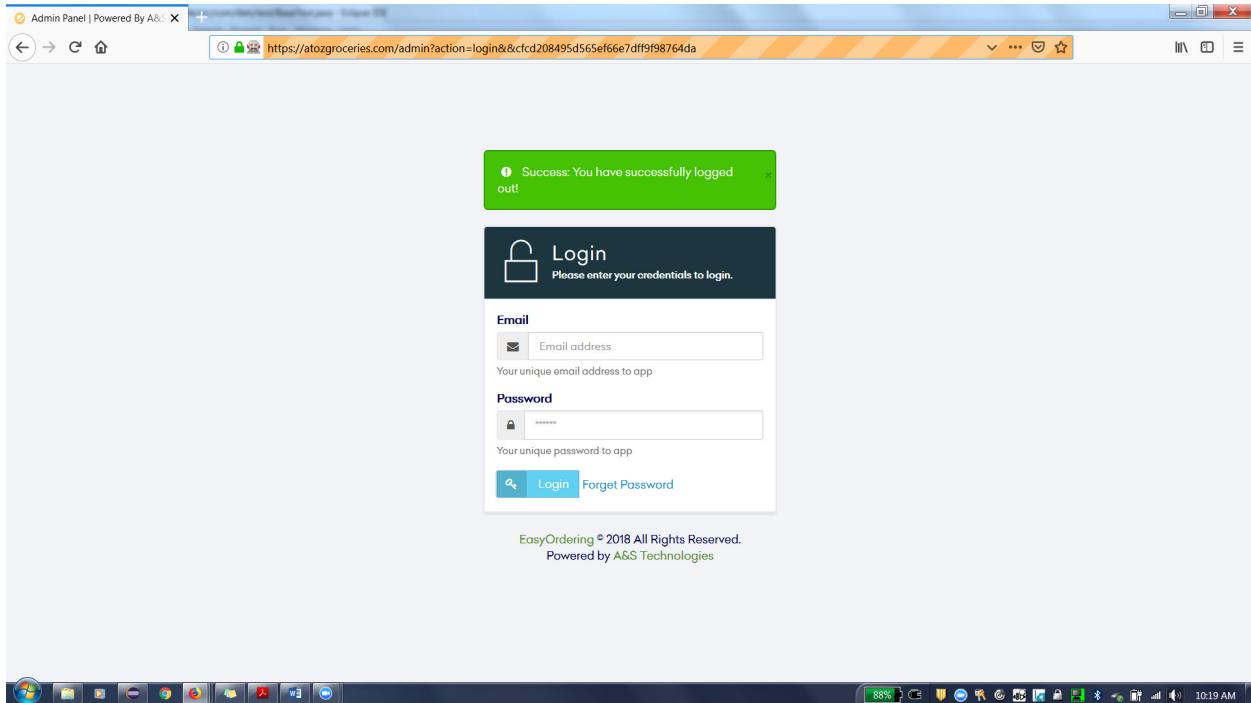
The screenshot shows the 'Product Tabs' section of the easyordering Admin Panel. On the left, there's a sidebar with links like Dashboard, Catalog (Tabs selected), Banners, Categories, Products, Product Image, Information, Sales, Customers, Marketing, System, and Reports. The main area is titled 'Tabs' and has a sub-section 'Add Tab'. It includes fields for 'Tab Name' (set to 'MilkMilk'), 'Sort Order' (set to '10'), and 'Status' (which is empty). The status field has a red asterisk indicating it's a required field.

12. After clicking on “Save”, a popup message is showing up on status – “Please select an item in the list”



This screenshot shows the same 'Add Tab' screen as the previous one, but with a modal dialog box overlaid. The dialog box contains the text 'Please select an item in the list' and is positioned over the empty 'Status' dropdown field. This indicates that the validation for the mandatory status field triggered an error message.

13. After successful validation of mandatory fields, logging out from the page



14. Output from the Console

A screenshot of the Eclipse IDE interface. The left side shows the "Package Explorer" with a tree view of Java files and packages. The central area shows the code for "BaseTest.java". The right side features a "Console" tab displaying command-line output. The output includes log messages from Selenium WebDriver, such as "INFO Listening on port 51113", "WARN TLS certificate errors will be ignored for this session", and "INFO: Detected dialect: W3C". It also shows JavaScript errors from the browser, including "TypeError: i is null" and "Message: Error: Polling for changes failed: NetworkError when attempting to fetch resource.". The bottom of the console output shows the test summary: "Suite Total tests run: 1, Failures: 0, Skips: 0". The Eclipse toolbar and status bar are visible at the bottom.

*** End of the document ***