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

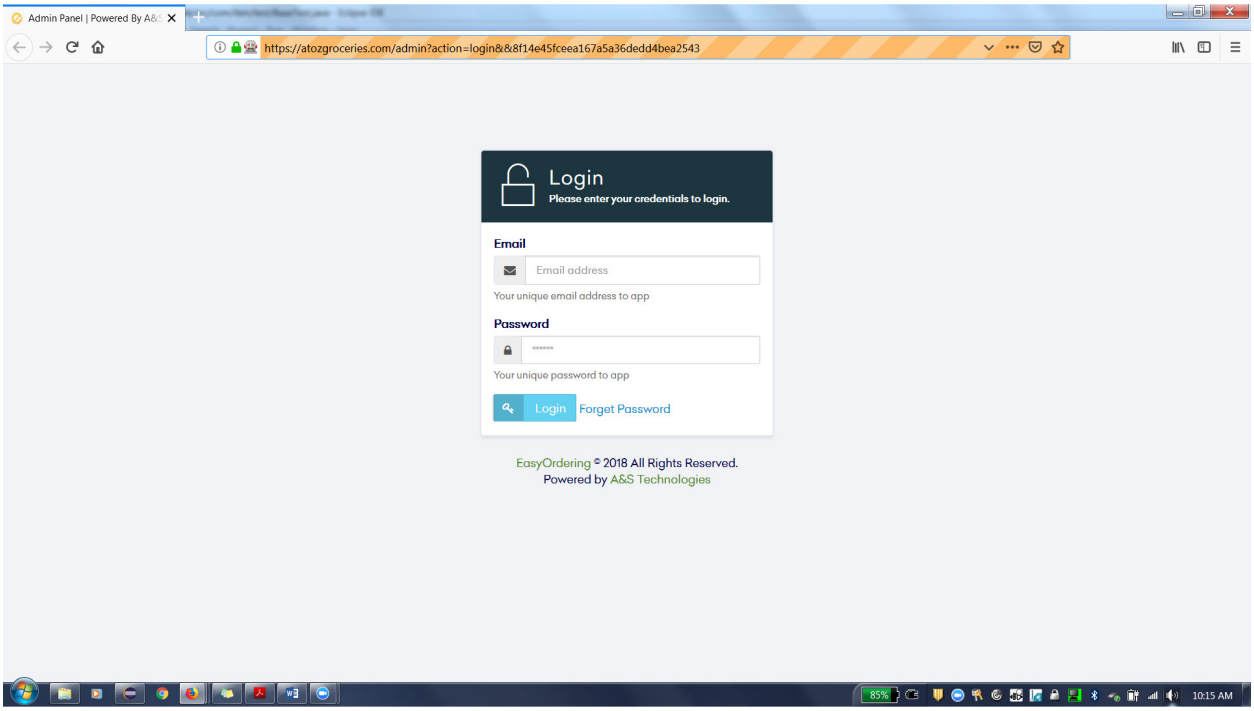
```
url=https://atozgroceries.com/admin
username=demo@atozgroceries.com
password=456789
categoryname=Milk Products
tagtitle=Milk Products
sortorder=12
status=Enabled
show=All
tagname=Milk
tagsort=10
tagstatus=Enabled
tagrequiredmsg=fill out this field
tagSelectReqMsg=select an item in the list
```

testing.xml

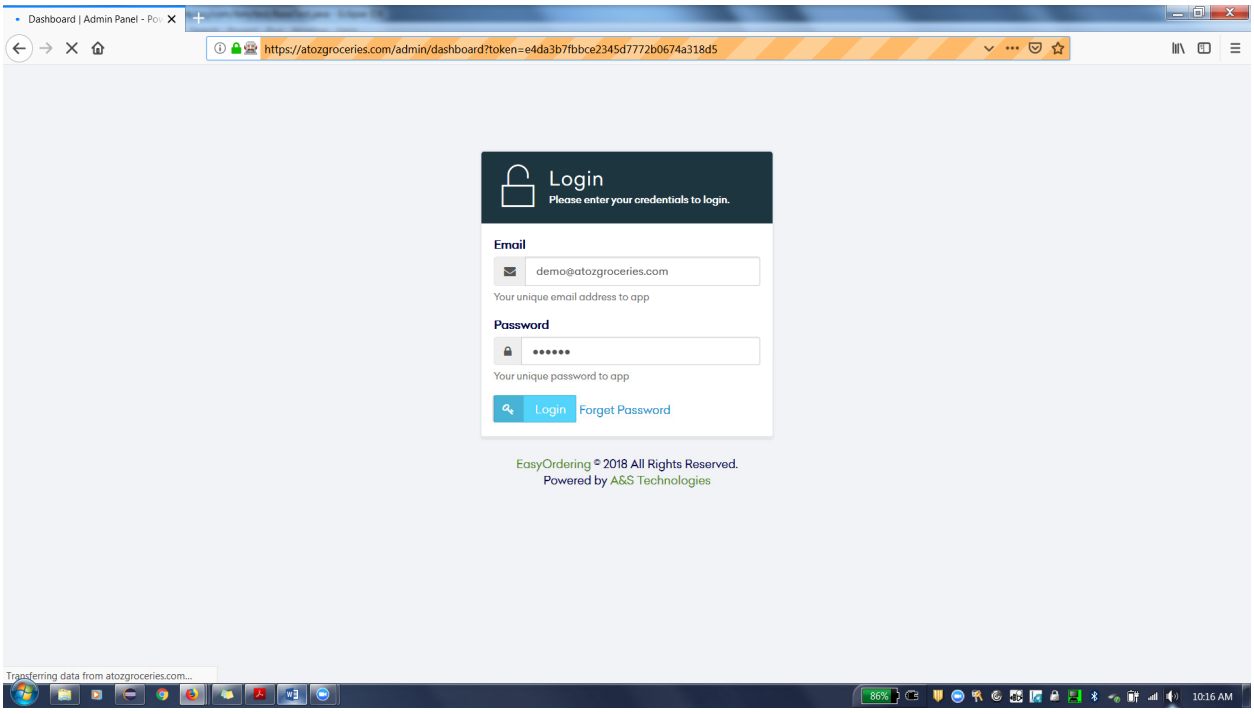
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Suite">
  <test thread-count="5" name="Test">
  <parameter name="browser" value="ff"></parameter>
    <classes>
      <class name="com.ibm.test.BaseTest"/>
    </classes>
  </test> <!--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

Step by step output:

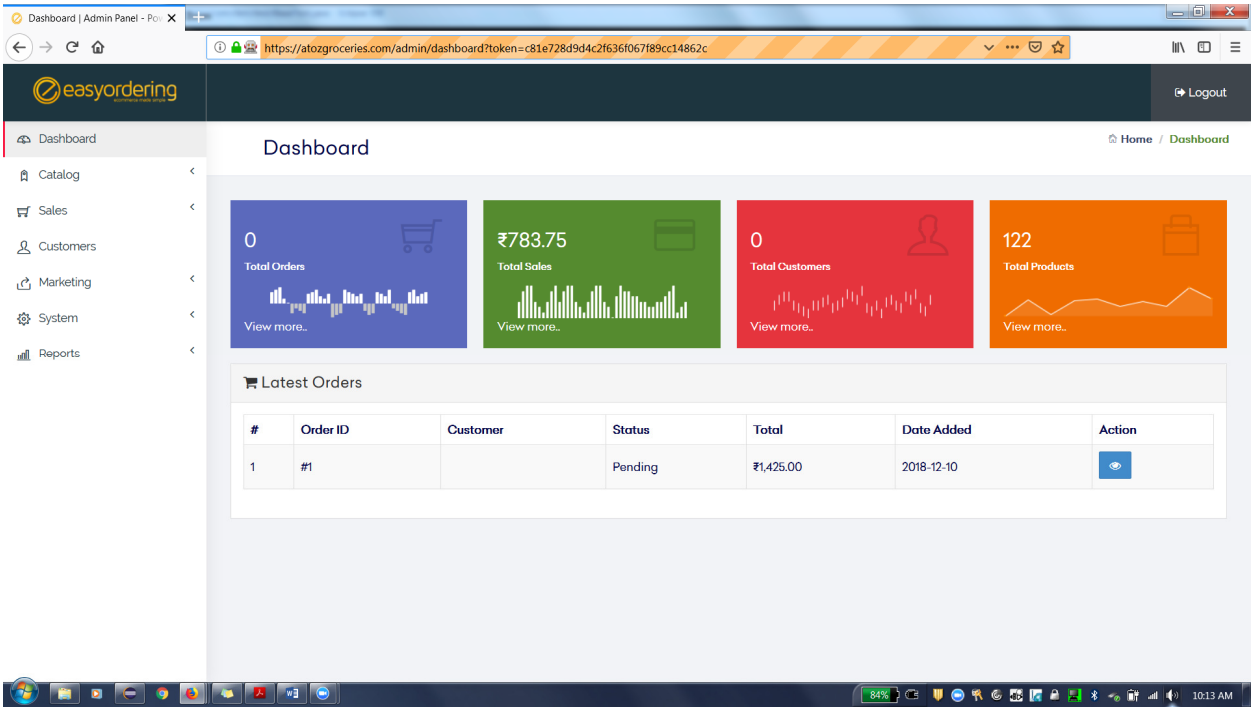
1. Navigating to the Admin Page of "**atozgroceries**"



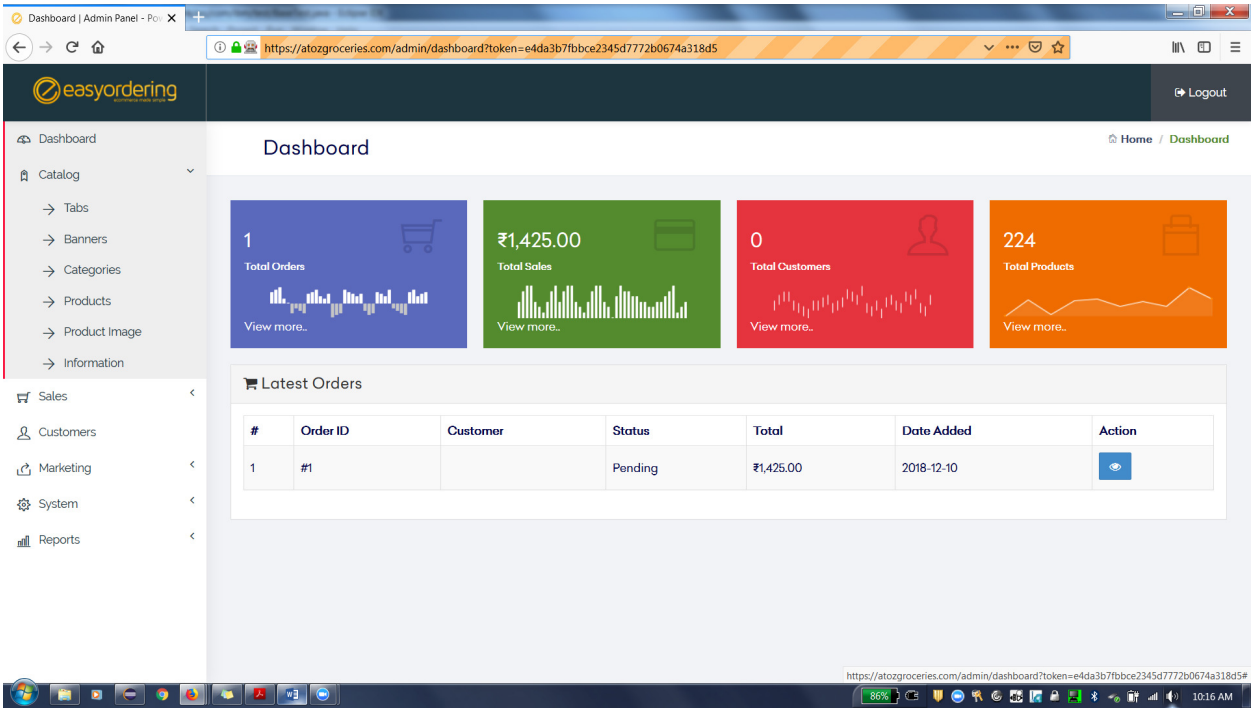
2. Providing login details



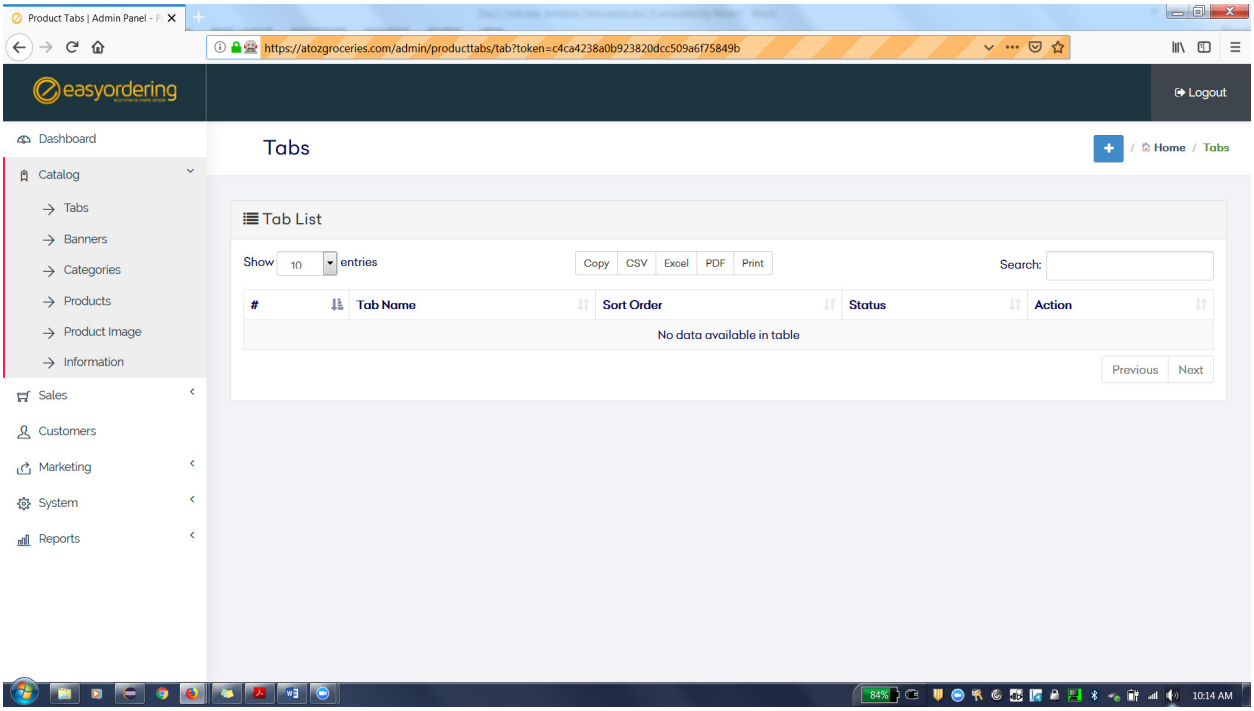
3. After logging into the webpage



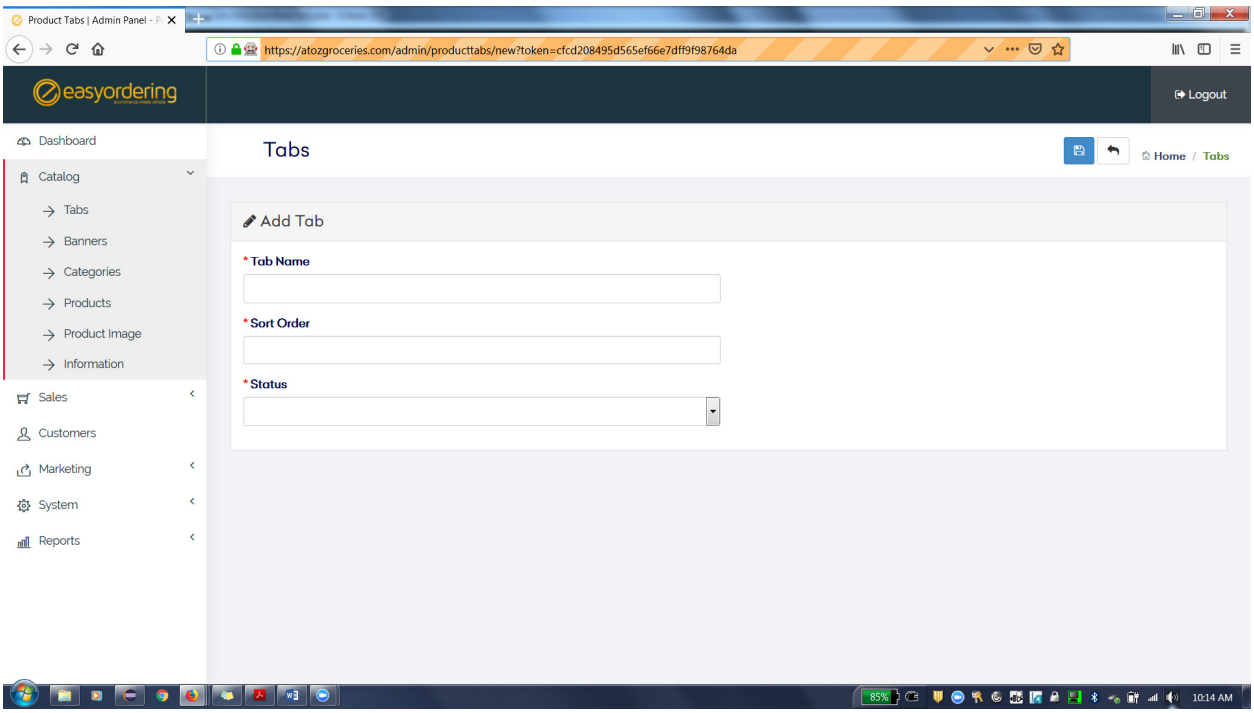
4. Opening Catalog



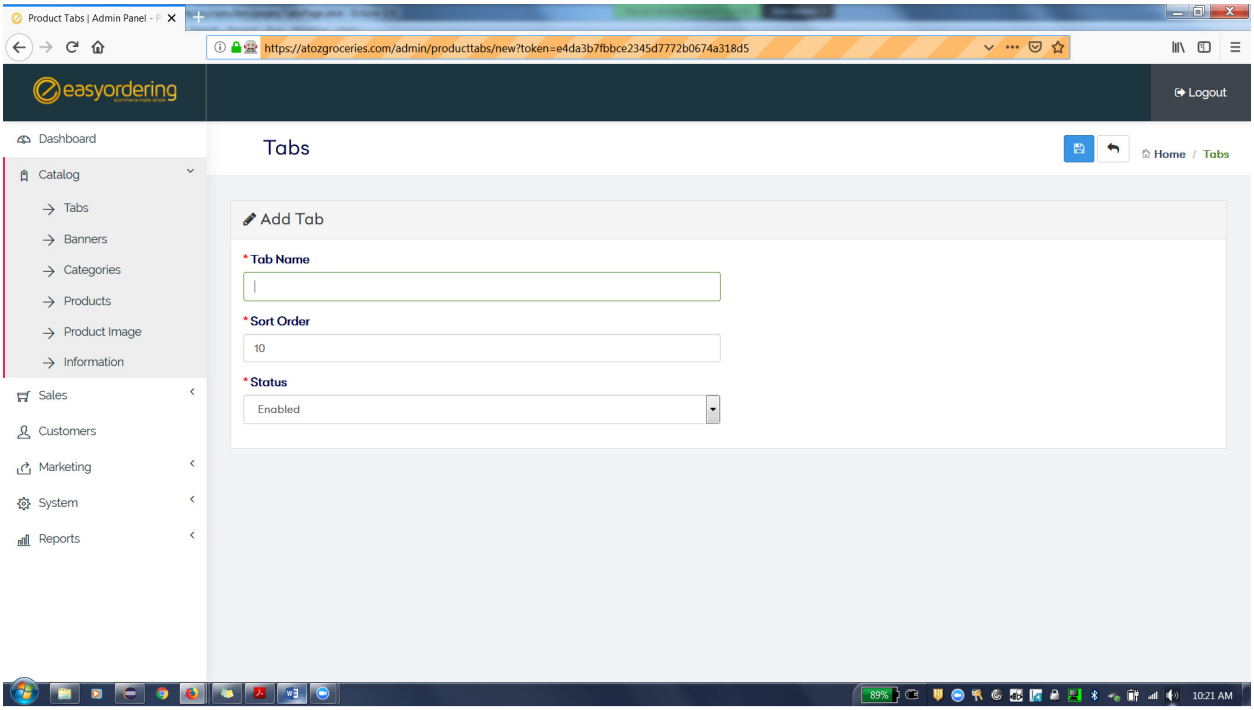
5. Clicking on Tabs



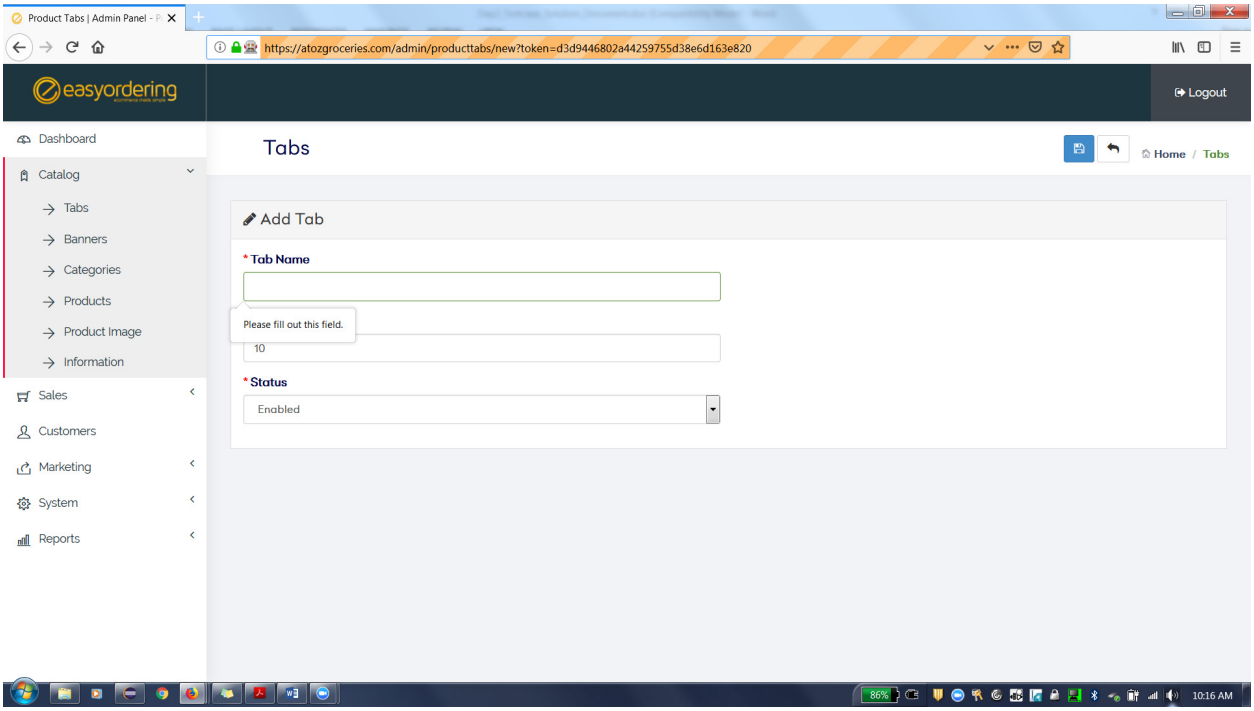
6. Clicking on “+” to add new tab



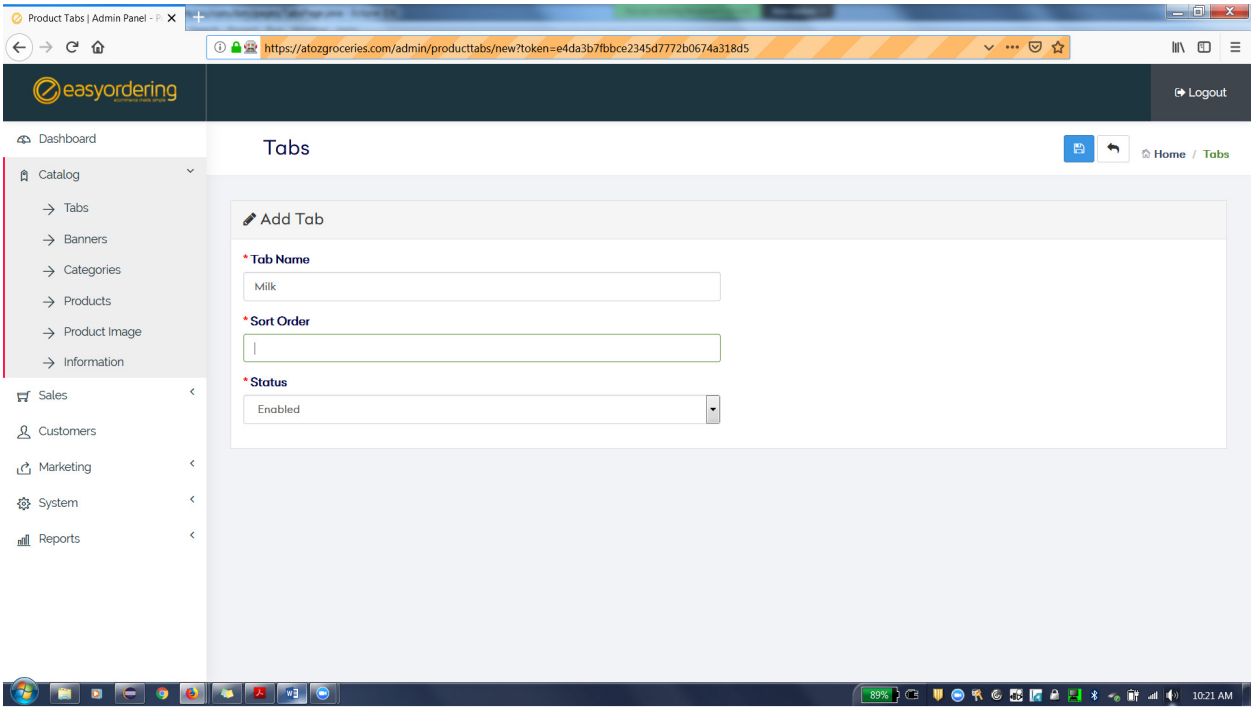
7. Providing input values for Sort Order and Status for validating the “Tab Name” as mandatory field.



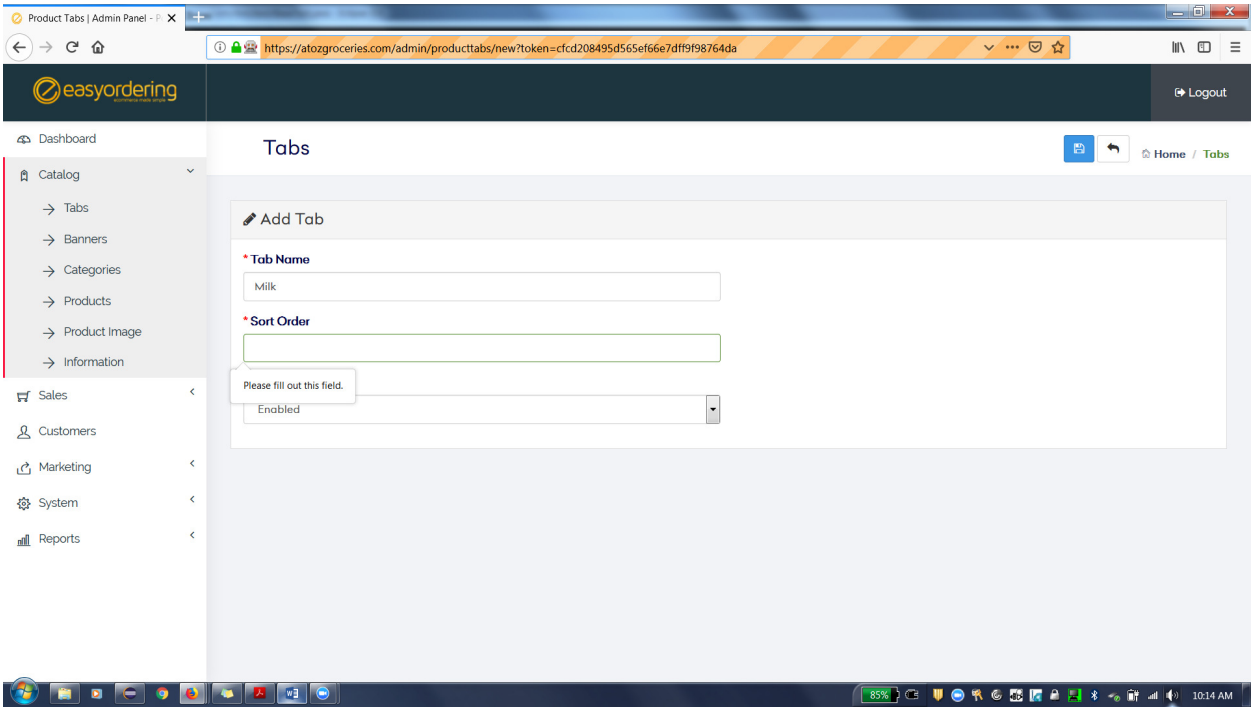
8. After clicking on “Save”, a popup message is showing up on name – “Please fill out this field”



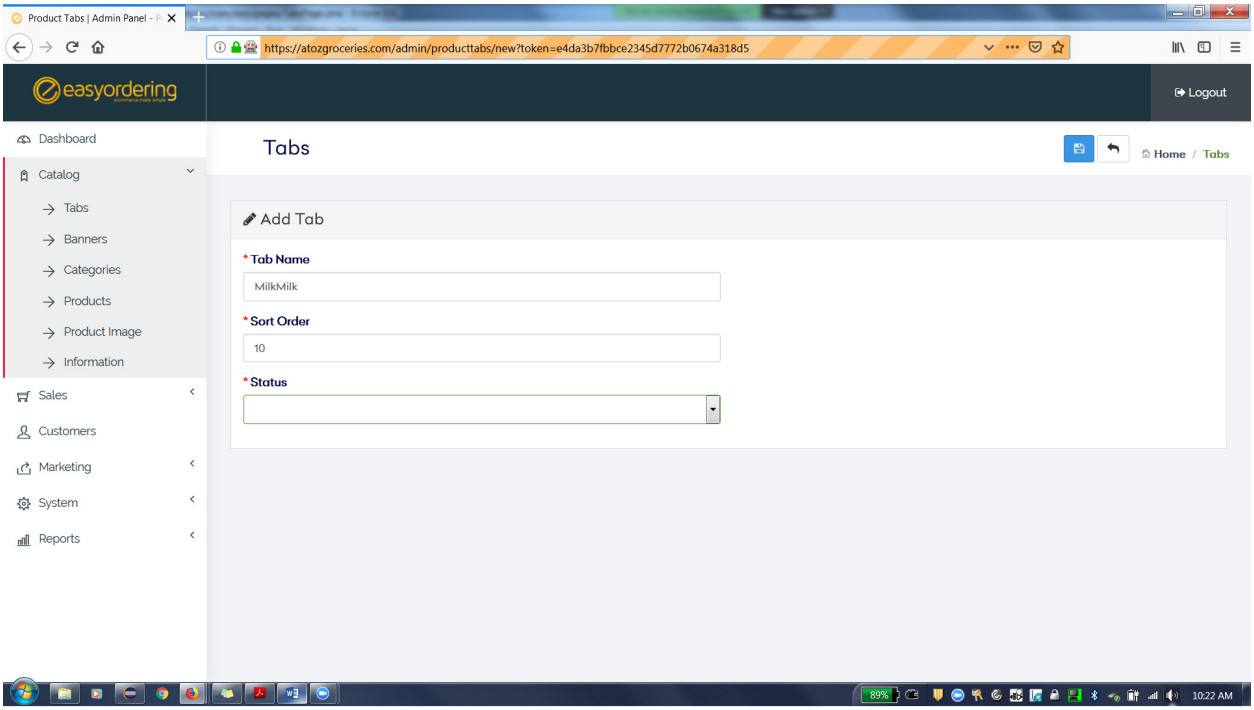
9. Providing input values for Tag Name and Status for validating the “Sort Order” as mandatory field.



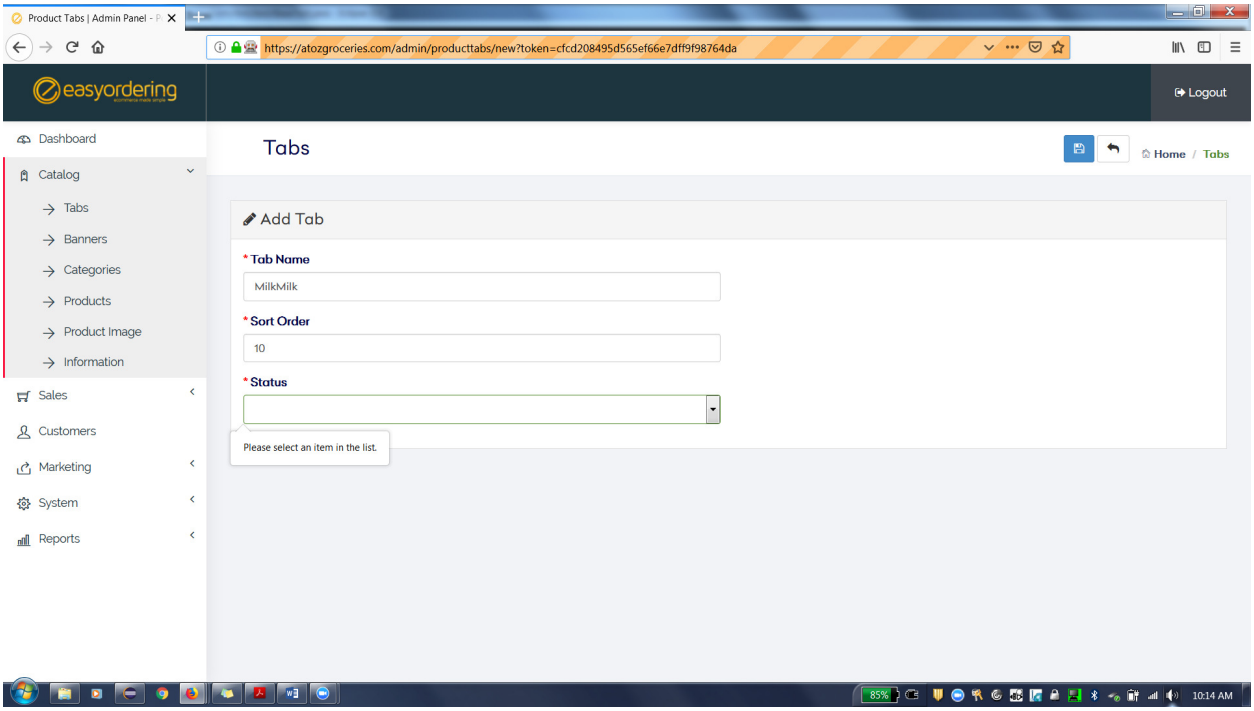
10. After clicking on “Save”, a popup message is showing up on sort order – “Please fill out this field”



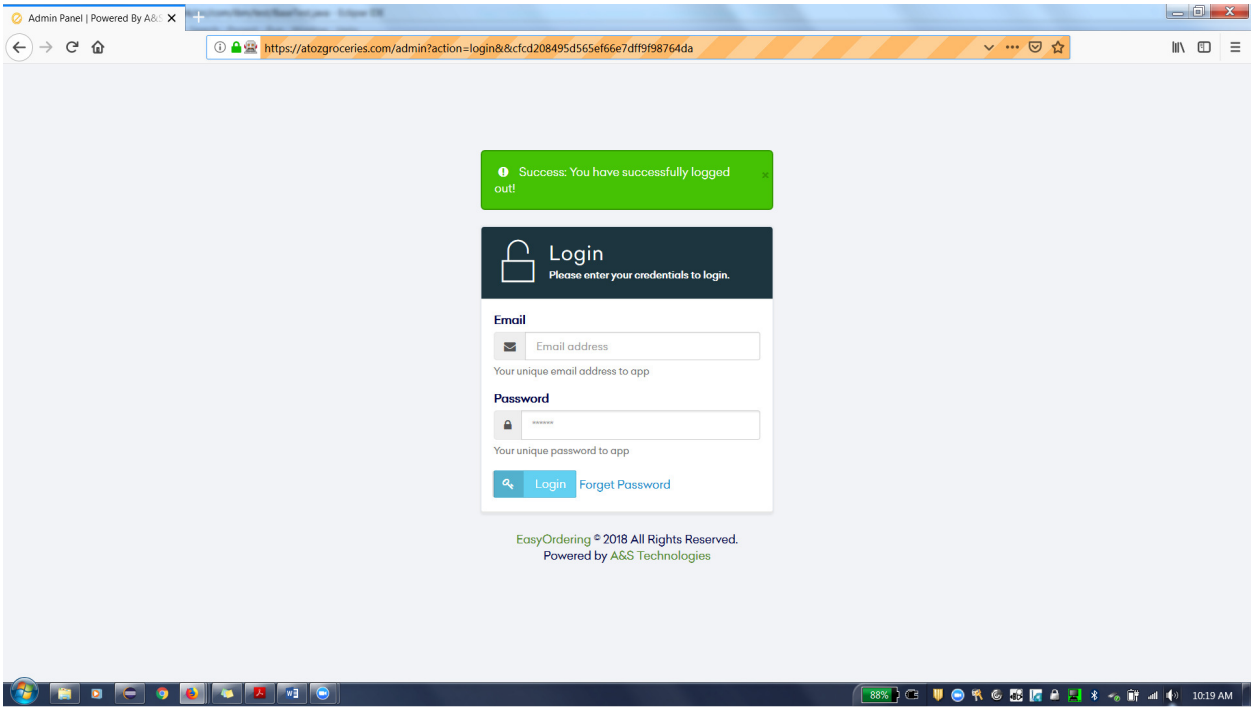
11. Providing input values for Tag Name and Sort Order for validating the “Status” as mandatory field



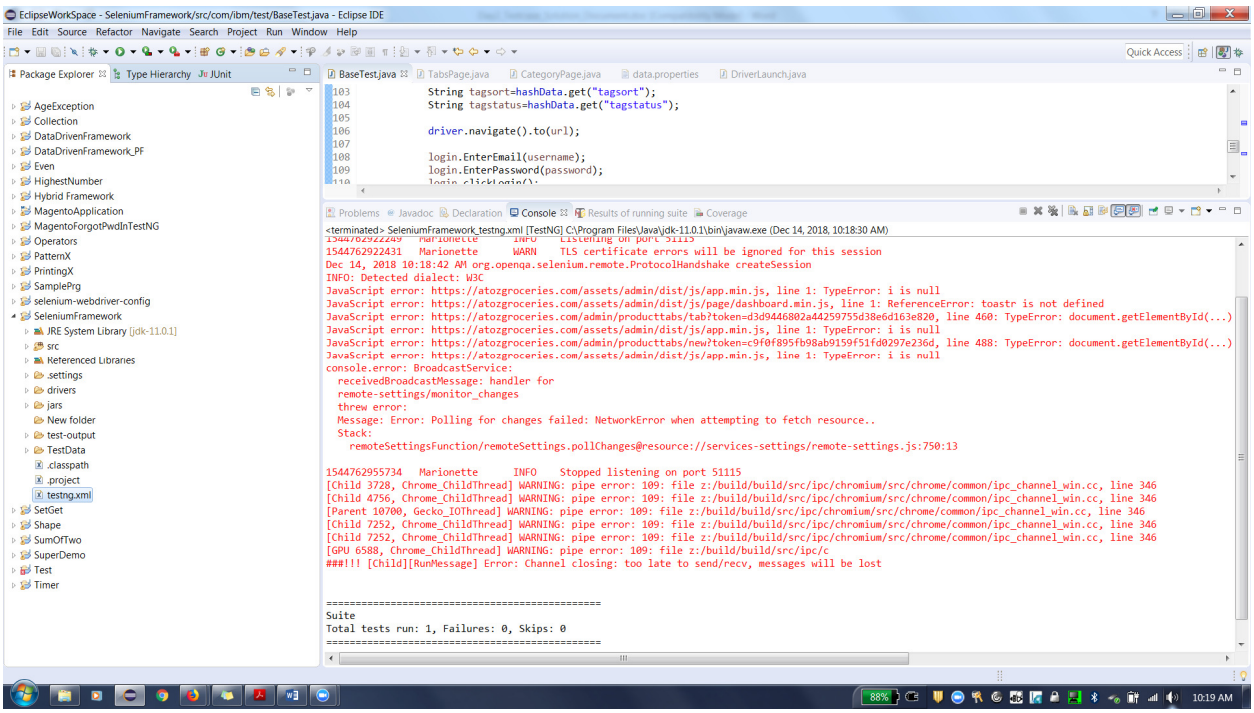
12. After clicking on “Save”, a popup message is showing up on status – “Please select an item in the list”



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



14. Output from the Console



***** End of the document *****