Instructions For JavaAngular Shopping Cart

Introduction

The following markdown instructions has to be read by each assessment applicant and it contains the following

- Case Study Details
- Case Study Modules
- Prerequisites
- Instructions for codecommit repository ,Mysql DB , Java and Angular related instructions pertaining to casestudy
- Rest API, Unit Testing and general DONTS instructions.

Case Study Details

The casestudy is to design an online shopping website that sells products online. The features to be implemented are

- Design a website through which a customer can order products online.
- The web application should have a product gallery with different type of products.
- Customers can select the product from the product gallery which contains description,discounts, original value and price.
- Customers can purchase any no. of product quantity based on stock availability.
- Customer can easily add, edit, and delete his /her cart.
- Once added to cart, the user can finally do a checkout.
- When the checkout happens the user gets an unique order number generated by the system.
- The product master data for the application is given as a set of tables with preloaded data by test admin.
- The applicant just needs to use the master data and not modify the data.
- The CaseStudy has two modules namely Registration and OnlineShopping

Case Study Modules -Registration

The registration has only 2 major functionality namely

- Product catalog
- Login-Functionality

Case Study - Online Shopping

The online shopping functional requirements are

- Navigation of Products with Product Catalog
- Product Catalog displays the product description, discounts if any, quantity available and price.
- Selection of Products from catalog
- Add products to the cart
- Remove a Product/All product from the cart.
- Review the cart and additionally select the delivery type from the available options
- Place an Order
- Mandatory Business Validation of choosing the delivery type is to be done.

Prerequisites

- OS - Windows or Linux
- IDE/Tools - Visual Studio Code 1.23.x or higher https://code.visualstudio.com/download, Eclipse Oxygen or higher https://www.eclipse.org/downloads/packages/release/oxygen
- MySQL Work Bench 6.3.x or higher https://dev.mysql.com/downloads/workbench/
- Database - Local MySQL 5.6.x or higher https://dev.mysql.com/downloads/windows/installer/8.0.html
- Java SE Development Kit 8 or higher Link is https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Node Package Manager version 5.6.x or higher https://nodejs.org/en/download/
- Apache Tomcat 8.5.29 version or higher https://tomcat.apache.org/download-80.cgi
- Browser Chrome
- Selenium jars/wars 3.9.x https://jar-download.com/?search_box=selenium-3.9.0

Getting Started with AWS Code Commit

- The Administrator will create a AWS Code Commit Account exclusively only with your empid for the case study.
- Instructions will be mailed with a welcome mail from administrator to every applicant detailing the AWS Code Commit Repository connection .
- AWS Code Commit is very similiar to GITHUB and once the CodeCommit basic security and configuration is done
- A complete Boiler Plate code will be available , the boiler plate includes Java Framework code , MYSQL sql files , Angular Base Code Base, RestAssured and Selenium Lib Jars/Wars which will be used for unit testing by the applicant.

- The applicant will have the code in the master branch of AWSCOde Commit with an unique repository created for each user appended with EmpID (xxxxx).
- Applicant can create branch from the master work on the case study in the branch , finally push the code to master and submit (More details are given subsequently)

Setting up Database in mysql.

- The name of the database should be retail_xxxxx. (Replace xxxxx with your employeeId).
- Run the mysql dump files and import into mysql as supplied in boiler plate codecommit code codebase
- Create a user with username: user and password: password.
- Grant full rights on database retail_xxxxx to the created user.
- Your mysql should be running on only port 3306.
- Once the dump is successfully imported into a new mysql schema say retail_xxxxx ,5 tables will be there with needed master data . The 5 tables are orders,products,shopping_cart,shopping_cart_details and user.

Instructions for Java-SpringBoot-Hibernate.

- Do not delete anything from pom.xml as this affects the evaluation.
- Do not delete any xml's from the boilerplate code.
- **Replace xxxxx with your employeeId in hibernate.cfg.xml in webapp/WEB-INF**
- **Use 'mvn clean package' instead of 'mvn clean install'.**
- The packaging should be a war not a jar.The package will be shopec_xxxxx.war where xxxxx is empid of the applicant
- Follow checkstyle and pmd rules for codequality.
- war should be deployed in tomcat as shopec_xxxxx. (Replace xxxxx with your employeeId).
- tomcat should run on port 8080
- Java Architecture is based on Sprint Boot, Spring REST , JPA and Hibernate
- Unit testing has to be done to check the maintainability using JUnit . Minimum Junit is 10 cases

Instructions for creating REST API's.

The 3 REST API's should be named as per the api list.

- **GET/products** - Get Method
- **POST/order** - Post Method which has to have the following body for eg,

```
Body:  { "items" : [
            { "quantity" : 1,
             "price" : 5,
             "productId" : 1
            }
           ],
   "grossTotal" : 49.989999999999995,
   "deliveryTotal" : 19.99,
   "itemsTotal" : 30,
   "userId" : 1,
   "deliveryOptionId" : "f488db46-9380-45e2-b15e-4ace7bdb73",
   "deliveryOptionCode" : "D"
  }

Response: {
         "orderNo": 43
       }
```

- Status code: 200.

- **/account/login** - Post Method which has to have the following

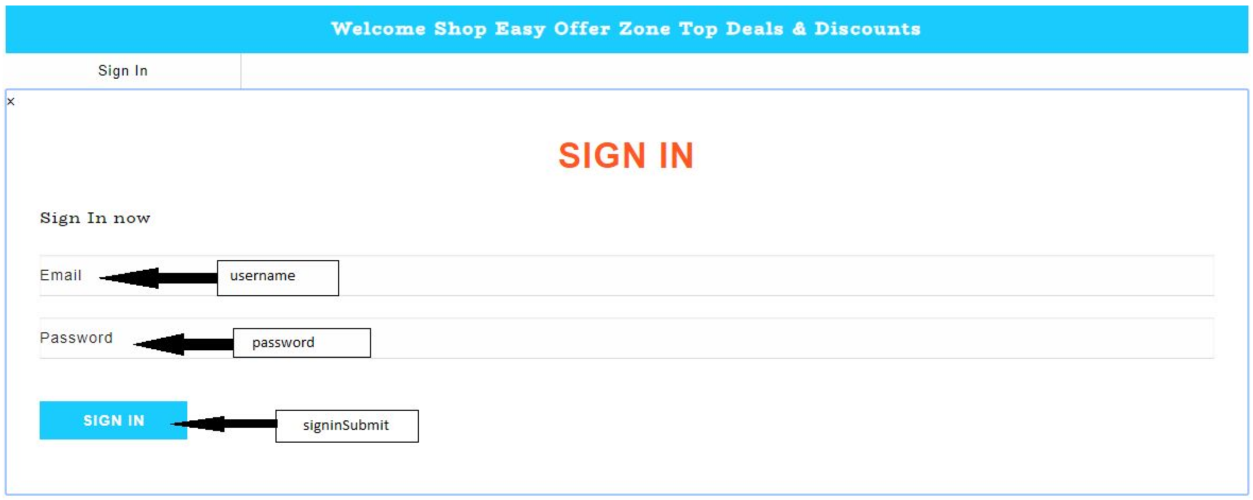
```
Body: {
      "userId" : "User1@g.com",
   "password" : "password"
  }
```

- Status code: 200.

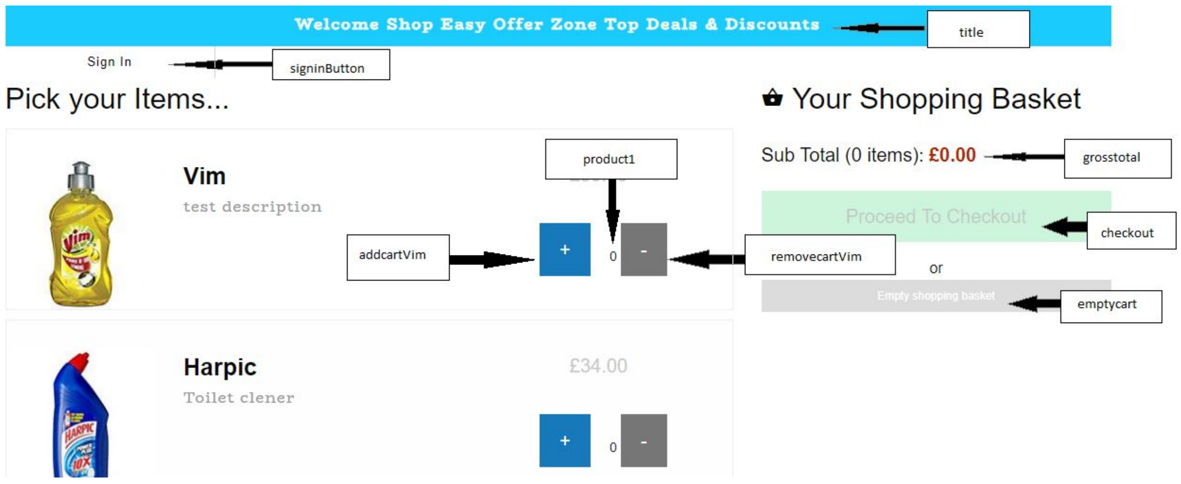
- Each of the RESTAPI should necessarily only in 8080 tomcat port and has to append mandatorily with the applicant empid(xxxxx). For eg http://localhost:8080/shopec_xxxxx/products/list;

Instructions for Angular.

- All Widgets should have an id
- Id's should be added to all the web elements as per the given Id list.This is mandatory
- Check the images folder for the way each ids have to be defined as datadriven. The 4 funcitonality with image screenshot are Login, AddProduct,Checkout and Orderno

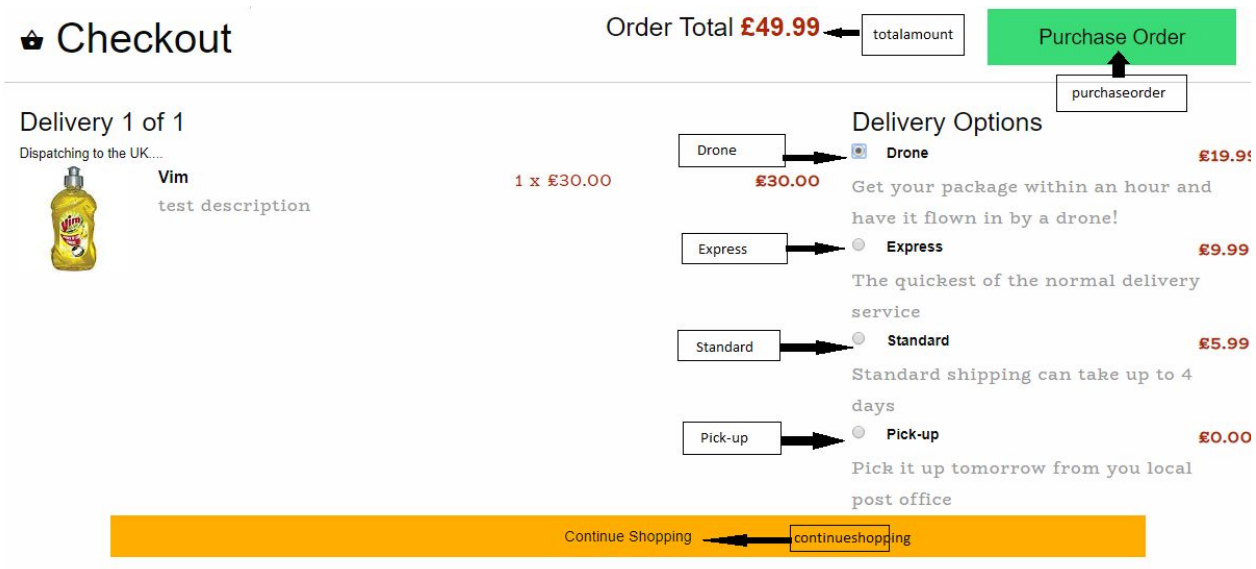


- The Login page has to be designed as above
- *on pressing signinSubmit button it has to call api **/account/login***

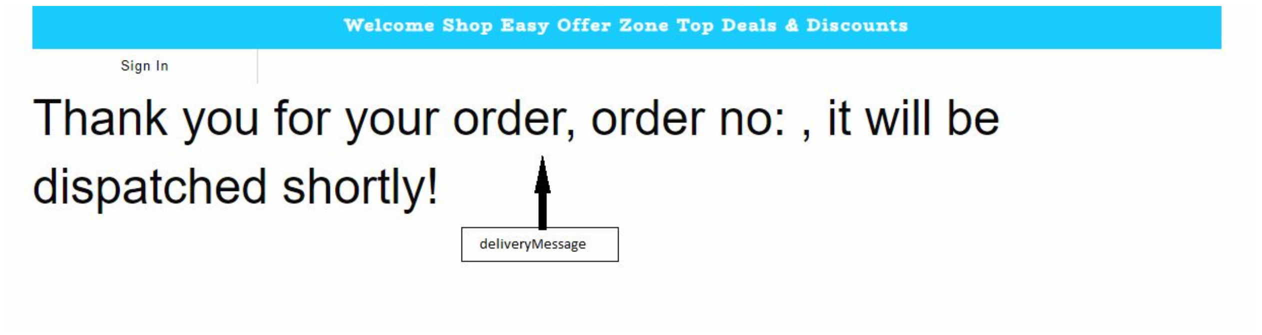


- The Add product has got a list of products enlisted in a list view as shown above.
- Each product is dynamically assigned an id like product1, product2 ,productn etc... where 1,2...n is data driven
- The + and - button for adding and removing cart is also having a dynamically assigned . For eg addcartVim,removecartVim where Vim is product name that is a datadriven field

- Please note the addcart,removecart are all case sensitive.
- to assign id dynamically follow given example
    - eg: for addcartVim id is assigned as id='addcart{{product_name}}'
        - here product_name is a dynamic value in a table assigned using ngFor loop.
- *For getting the products list as shown in the image above you have to use api **/GET/products***
- **This is the landing page.*(/images/shoppingcart-addproduct.png)***



- The Cart checkout page has to be enlisted as shown in the above image.
- Delivery Options is not mysql table driven but is driven from Angular assets
- Picking up DeliveryOption is mandatory and the value of price for deliveryoption is added to the total price
- Continue Shopping functionality can allow user to reedit by adding and removing cart.
- *For getting the delivery option details you have to call delivery-options.json which is in assets folder of Angular boilerplate code*
- *For calling delivery-options.json you have to use url "**./assets/delivery-options.json**"*
- *On clicking purchase order it has to call api **/POST/order***



Thank you for your order, order no: , it will be dispatched shortly!

- The orderpage is the final page which will be shwon to the enduser as above.
- Once cart with delivery option is selected and when purchase order is clicked an unique order no is generated and shown to user.
- The Purchase Order button has to be enabled only when the cart has added products.
- It is mandatory to have the exact messsage when the order is generated as shown in the image.
- TypeScript lint rules shoud be followed for code quality.
- Failure to assign id's leads to failure of functional testing.Recheck for the Angular UI coding to have the exact id as above
- Jasmine/Karma testing has to be written.
- The Angular application has to build using **npm run build** command.
- The build should be deployed in the tomcat as dist_xxxxx. (Replace xxxxx with your employeeId).

DON'TS

- Donot remove EmployeeDetails.Json file or do not modify it.
- Donot make any changes in the Submit.sh file.
- Make sure you pushed the code into master from branch before submitting the test.
- Test can be submitted only once. So, do not click Submit.sh until you wish to submit finally the test.
- In the event of you realizing the submission has to happen again , please contact your test administrator. This is not encouraged
- Prior to submitting by clicking submit.sh make sure both the java code and Angular code is done with a clean build and complete checkin all the code.
- Prior to submitting with the click of submit.sh clean the compiled code and only submit the end .
- If you have require any clarifications only approach the test administrator for more clarifications
- Tests are autoevaluated it is mandatory to follow instructions so naming convention of db schema, rest api conventions , Angular application parameters are all mandatory.