**SMART SHOPPING**

A REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE COURSE

**Course name: e-DAC**

**Team member name:**

1. Smita Srivastava : 200950181102
2. Priyanka: 200950181077
3. Vrushali Bhosale :200950181114
4. Rachit Bharadwaj:200950181079
5. Ashish Pandey: 200950181019

**Project Details:**

1. Smart shopping is a online website portal where a user can visit the website and buy any item based on the category of his choice.
2. When the user wants to buy any product he has to create an account on the portal and then he can buy as many product of his choice.
3. The Admin has the authority to manage product i.e. admin can add new product, remove a product and activate and deactivate the product on the basis of availability.
4. Admin can also add different category depending upon the requirement.
5. User after creating the account can view the entire product available and can add as many products to the cart.
6. After the user has added the product to cart he can further proceed to buy the product by selecting the payment method and generate the invoice of order confirmation.
7. We have used spring, Hibernate, Maven, Angular JS and Bootstrap for the entire project.

**Placing an order**

Let us consider a scenario in which user visit the website and want to place an order for a product.

**Presentation:**

1. Cart component shows all the products user has wished to purchase from home.

2. User also has the option to increase or decrease the quantity of every product in the cart.

3. Remove single or all the products completely and make the cart empty

**Service Layer :**

1. When the user click on ‘shopping cart’ icon, request is mapped to the /cart pattern and the control goes to the CartController.java.
2. Service function will make the required changes to database and then return updated data to the controller, which will send a response message object as JSON string which will contain all the required updated data .

**DAO Layer:**

1. Service class uses the interface for queries provided by Hibernate.

2. We have Entity classes for these queries named accordingly where we have mapped the class according to database.

3. A query is made to update the cart table in database which takes product id and quantity as input.

4. On the success of first query second query is made to return the selected data from product and cart table from the database corresponding to the product ids present in the cart table.

**Problem Faced in The Project**

1. We faced the issue in the single product page where we should not be adding the product to the cart whose quantity is zero and instead of that display a Out Of Stock.
2. **We faced no bean named available problem while configuring the pom.xml file.**

We added the following dependency to resolve problem.

<dependency>

<groupId>javax</groupId>

<artifactId>javaee-api</artifactId>

<version>${javaee.version}</version>

<scope>provided</scope>

</dependency>

1. **Active Menu Problem**

We added the id to the list of elements in the navbar.jsp

Then we added a small script element in the head section and create a new property with the name of menu in the window global object.

Then we created a myapp.js file in the js directory created earlier and add a script element at the end.

1. **Connection establishment in H2 database**

Connection.properties file was created with following attribute

jdbc.url=jdbc:h2:tcp://localhost/~/smartshopping

jdbc.driver=org.h2.Driver

jdbc.dialect=org.hibernate.dialect.H2Dialect

jdbc.username=sa

jdbc.password=

## Lessons Learned

1. Dependency Injection and Inversion of Control (IoC) in the Spring Framework.
2. Spring Security
3. Using Spring Initializr.
4. Using Maven to build Spring Projects.
5. How to use JUnit and to test Spring.
6. Apply Spring MVC form validation on user input
7. Java and XML Spring Configuration.
8. Spring MVC and Bootstrap CSS.
9. Apply advanced Hibernate mappings: one-to-one, one-to-many and many-to-many.
10. Read HTML form data using @RequestParam and @PathVariable
11. We learn how to use SLF4J logging facility.
12. Build a complete Spring MVC and Hibernate CRUD Project … all from scratch
13. We learnt how to use AngualrJS and Spring.