1. **Project Objective**

As a Full Stack Developer, complete the features of the application by planning the development and pushing the source code to the GitHub repository.

1. **Problem Statement**

Sporty Shoes is a company that manufactures and sells sports shoes. They have a walk-in store, and now, they wish to launch their e-commerce portal sportyshoes.com.

You’re asked to develop a prototype of the application. It will be then presented to the relevant stakeholders for budget approval. Your manager has set up a meeting where you’re asked to do the following:

● Presenting the specification document which has the product’s capabilities, appearance, and user interactions  
● Setting up Git and GitHub account to store and track your enhancements of the prototype   
● Explaining the Java concepts used in the project  
● Discussing the generic features of the product  
● There will be an admin to manage the website. An administrator login will be required to access the admin page.

**The admin should be able to change his password if he wants, he should be able to:**

● Manage the products in the store including categorizing them  
● Browse the list of users who have signed up and be able to search users  
● See purchase reports filtered by date and category

1. **Application Name**

E-commerce Website for Sporty Shoes.

1. **Developer Details**

Anurag Pal, 216 Java SL (Evening Batch), Phase-3 Final Assessment

1. **Product’s Capabilities**

SportyShoes has a range of capabilities that helps the admin to carry out day-to-day operations smoothly.

The product provides a database called sportyshoes, in which the different tables are present. An Admin table is present, which contains all the details of existing admins.

Similarly, a product table and a category table is also present. The product table contains details of all the products and the category table contains all the details of the categories. The category table and the product table are in a One-To-Many relationship. This means that One category can have Many products and One product can have only One Category. The category ID(cid) is used a foreign key in the product table. Using cid, the category is mapped for a product.

Also, there is a user table in which all the details of a user registered to this product are shown. Now, the user table and the product table are in a Many-To-Many relationship. This means that Many products can be bought by Many users. A join table called orders is used to associate these two tables. The product ID(pid) and the user ID(uid) are taken as foreign keys in the orders table. Also, the date of order of a particular product by a user is also stored in this orders table. Using pid and uid, the products and their details, bought by some users and their details, are mapped.

1. **GitHub Link**

The github link for this product is as follows:

1. **Core Concepts Used**
2. Database and Table creation done in MySQL.
3. Total five tables were created – Admin, User, Category, Product and Orders.
4. Category and Product have a One-To-Many relationship.
5. Product and User have a Many-To-Many relationship. This is accomplished by using a join table called Orders.
6. Spring MVC is followed with different packages for Application, Controller, Entity and Repository, Service.
7. The controller class is a RestController which is used to create RESTful Web Services using Spring MVC.
8. Different types of mapping annotations, mainly RequestMapping, PostMapping, GetMapping, PutMapping and DeleteMapping have been used.
9. Database CRUD operations are carried out using Spring Data JPA.
10. **Generic Features**
11. /api/admin/insert – to insert new admin into the admin table.
12. /api/admin/list – to display details of all the admins from the admin table.
13. /api/admin/updatepwd/{aid} – to change password of the admin whose adminID(aid) is being provided.
14. /api/user/insert – to user new user into the user table.
15. /api/user/list – to display details of all the users from the user table.
16. /api/user/{uid} – to display details of the user whose userID(uid) is being provided.
17. /api/category/insert – to insert new category into the category table.
18. /api/category/list – to display details of all the categories from the category table.
19. /api/product/insert – to insert new product into the product table.
20. /api/product/list – to display details of all the products from the product table.
21. /api/product/{pid} - to display details of the product whose productID(pid) is being provided.
22. /api/product/update/{pid} – to update details of the product whose productID(pid) is being provided.
23. /api /product/{pid} – to delete the product whose productID(pid) is being provided.
24. /api/orders/list – to display details of all the purchase orders from the orders table.
25. /api /orders/filter – to display purchase order filtered by the date that is provided.
26. /api/orders/{cid} – to display purchase order filtered by categoryID(cid) that is provided.
27. /api/orders/update/{oid} – to update details of purchase order whose orderID(oid) is being provided.
28. **Conclusion**

This application is mainly designed using Spring MVC Framework, along with Spring Boot, Spring Data JPA and RESTful Web Services.

**Unique Selling Points (USPs)**

1. Lots of different features helps the admin to carry out the daily operations smoothly.
2. Highly customizable as CRUD operations are mainly done using Spring Data JPA.
3. Admin can check all the details of users, products, categories, orders, etc. easily without any hassle.
4. Admin can also customize the mapping annotations to make data retrieval and data manipulation easier.