Course-end Project 1

Mohammed Kaif

Make an E-commerce Website for Sporty Shoes

GitHub link: [kaif2106/sportyShoes (github.com)]

Sporty Shoes is a company that manufactures and sells sports shoes. This project demonstrates a basic e-commerce backend with admin capabilities. It's designed to be a prototype that can be expanded into a full-fledged web application. The use of Spring Boot and JDBC provides a solid foundation for building more complex features and scaling the application as needed.

Java Concepts Used

Object-Oriented Programming (OOP):

Encapsulation: We use private fields with public getters and setters in our entity classes (User, Product, Order).
Inheritance: The application implicitly extends from Spring Boot's base classes.
Polymorphism: Used implicitly through Spring's dependency injection system.

Spring Framework:

Dependency Injection: We use @Autowired to inject dependencies like repositories into services.
Inversion of Control: Spring manages the lifecycle of our beans (components marked with @Service, @Repository).

Spring Boot:

Auto-configuration: Spring Boot automatically configures the application based on the dependencies in the classpath.
Embedded server: The application can run standalone without an external web server.

JDBC (Java Database Connectivity):

We use Spring's JdbcTemplate to interact with the MySQL database.

Repositories:

Data access layer that encapsulates database operations.

Services:

Business logic layer that coordinates between the repositories and the user interface.

Annotations:

@SpringBootApplication, @Service, @Repository, @Autowired are used to configure the Spring application.

Exception Handling:

Implicit in the JdbcTemplate operations.

Collections:

List interface is used to return multiple users, products, or orders.

Date Handling:

java.util.Date is used for order dates.

Generic Features of the Product:

User Management:

The application can store and retrieve user information. Users have attributes like username, password, and email.

Product Management:

Products can be added to the system.
Products have attributes like name, category, and price.
The system can list all products.

Order Management:

The system can record orders, associating users with products.
Orders include the date of purchase.

Admin Functionality:

Password Change: Admins can change their password.
User Listing: Admins can view all registered users.
Product Management: Admins can add new products and view all products.

Reporting:

While not fully implemented, there's a placeholder for viewing purchase reports.

Database Integration:

The application integrates with a MySQL database to persist data.

Modular Design:

The application is divided into entities, repositories, and services, allowing for easy maintenance and extension.

Scalability:

The use of Spring Boot and a database allows for potential scaling of the application.

Extensibility

The current structure allows for easy addition of new features, such as user registration, more detailed product management, or a web interface.

Screenshots: