**Sporty Shoes E-Commerce Prototype Specification Document**

GitHub URLs:

BE: <https://github.com/timbabs1/sporty-shoes-be>

FE: <https://github.com/timbabs1/sporty-shoes-fe>

**1. Overview**

**Product Name:** Sporty Shoes E-Commerce Portal  
**Project Description:**  
Sporty Shoes is a company that manufactures and sells sports shoes. In addition to their walk-in store, they wish to launch an e-commerce portal—**sportyshoes.com**—that allows both administrators and users to interact with the system. Administrators can manage products, users, and view purchase reports, while users can browse products, add items to a cart, place orders, and register/login to their accounts.

**2. Product Capabilities**

**2.1 Administrator Capabilities**

* **Admin Login & Password Management:**
  + Secure login with an administrator account.
  + Ability to change the admin password via a dedicated endpoint.
* **Product Management:**
  + Create, update, and delete products.
  + Associate each product with a category (e.g., Running, Basketball).
  + Upload an image for each product. The image file is stored on disk (in a folder such as sporty-shoes-images), and its URL is stored in the database.
* **User Management:**
  + View a list of registered users.
  + Search users by username.
  + Create, update, and delete user accounts.
* **Purchase Report Generation:**
  + View purchase reports filtered by date and product category.
  + Reports include purchase details like user, product, date, and quantity.

**2.2 User Capabilities**

* **User Login & Registration:**
  + Ability to login as a registered user.
  + New users can register by providing a username, email, and password.
* **Product Browsing:**
  + View a catalog of products with details including name, price, category, and product image.
* **Shopping Cart & Order Placement:**
  + Users can add products to a shopping cart.
  + Specify the desired quantity for each product.
  + See a running total of the order.
  + Place orders, which will be recorded in the purchase records.
* **Image Preview:**
  + Products display an image preview that is fetched from the Spring Boot backend via a dedicated image-serving endpoint.

**3. User Interface & Appearance**

**3.1 Admin Interface**

* **Dashboard Layout:**
  + **Header:** Displays a welcome message and a logout button.
  + **Navigation:**
    - Links to Products, Users, Reports, and Change Password pages.
  + **Product Management Page:**
    - A form to add new products (with fields for name, price, category, and image upload).
    - A table listing all products with edit and delete options.
  + **User Management Page:**
    - A search bar to filter users.
    - A table displaying user details.
    - Functionality to create, update, or delete user accounts.
  + **Reports Page:**
    - Date range and category filters.
    - Display of purchase records matching the filter.
  + **Change Password Page:**
    - Form to enter and confirm a new password.

**3.2 User Interface**

* **User Dashboard:**
  + **Header:**
    - Displays a welcome message (with the user’s name) and a logout button.
  + **Product Listing:**
    - Each product is presented with its image, name, price, and category.
    - A quantity input allows the user to select how many items to add.
    - An “Add to Cart” button is provided for each product.
  + **Cart View:**
    - Displays a list of items in the cart with quantity and price.
    - Shows the total order amount.
    - A “Place Order” button to submit the order.
  + **Responsive Behaviour:**
    - The interface is designed for ease of use, ensuring that users can quickly find products and complete orders.

**3.3 User Interactions**

* **Navigation:**
  + Admins and users navigate via clearly labelled menu options.
* **Form Validation:**
  + Input forms for login, registration, and product management enforce required fields.
* **File Upload:**
  + Admins can select an image file when creating or updating products.
* **Dynamic Cart:**
  + The user dashboard dynamically updates the cart as products are added, showing updated totals.
* **Feedback:**
  + Alerts and error messages are provided upon successful operations (e.g., order placed) or errors (e.g., invalid credentials).

**4. Java Concepts & Technologies Used**

**4.1 Spring Boot Framework**

* **Spring Boot:**
  + Simplifies application configuration and deployment.
  + Provides embedded Tomcat for easy development and testing.

**4.2 Spring MVC**

* **Controllers:**
  + Use of @RestController and @RequestMapping to create RESTful endpoints.
  + Handling of HTTP requests (GET, POST, PUT, DELETE).

**4.3 Spring Data JPA & Jakarta Persistence**

* **JPA Entities:**
  + Domain models (Admin, Product, Category, User, Purchase) annotated with @Entity and mapped to database tables.
* **Repositories:**
  + Use of JpaRepository to provide CRUD operations and custom queries.
* **Jakarta Persistence:**
  + Annotations like @Entity, @Table, @Id, and @GeneratedValue now use jakarta.persistence package.

**4.4 Dependency Injection & Inversion of Control (IoC)**

* **Spring Container:**
  + Manages the creation and injection of beans (e.g., services, repositories, controllers) using @Autowired.
* **Loose Coupling:**
  + Encourages a layered architecture where controllers call services, and services call repositories.

**4.5 Exception Handling & Testing**

* **Exception Handling:**
  + Basic error handling in services and controllers to return appropriate HTTP statuses.
* **Testing:**
  + Use of JUnit, Spring Boot Test, and MockMvc for unit and integration tests.
  + Tests for controllers (including file upload endpoints) and service methods.

**4.6 File Handling & Multipart Processing**

* **MultipartFile:**
  + Used in controllers to handle file uploads.
* **File I/O:**
  + Java NIO (e.g., Files.copy, Paths.get) is used to save image files to disk.
* **Resource Serving:**
  + Custom endpoint (or resource handler configuration) to serve images from the file system (ImageController).

**5. Generic Features of the Product**

**5.1 Extensibility & Maintainability**

* **Modular Design:**
  + The application follows a layered architecture (controllers, services, repositories, models) which makes it easier to extend and maintain.
* **Configuration Driven:**
  + Use of Spring Boot’s configuration properties (e.g., for file upload directory) enables easy changes without code modification.

**5.2 Scalability & Performance**

* **Database Access:**
  + Spring Data JPA optimizes database interactions and can be further tuned.
* **File Storage:**
  + Image files are stored on disk; in a production environment, this could be replaced with cloud storage if necessary.
* **Caching & Optimization:**
  + The application can be extended to include caching strategies for frequently accessed data (like product listings).

**5.3 Security**

* **Authentication & Authorization:**
  + Basic login functionality for both administrators and users.
  + In a production system, additional security measures (e.g., Spring Security, HTTPS, JWT) would be implemented.
* **Input Validation:**
  + Basic input validation is implemented; further enhancements can be added to prevent common vulnerabilities.

**5.4 Usability & User Experience**

* **Responsive Design:**
  + The React front end is designed for ease of use with clear navigation and feedback.
* **Role-Based Interfaces:**
  + Separate interfaces for admins and users ensure that each user type sees only the relevant options.
* **Error Handling & Feedback:**
  + The system provides error messages and alerts to guide users during operations such as login, registration, and order placement.

**6. Conclusion**

This prototype for Sporty Shoes’ e-commerce portal demonstrates a comprehensive solution covering both administrative and user interactions. The back end leverages modern Java concepts with Spring Boot, Spring Data JPA, and Jakarta Persistence to provide robust API endpoints, while the React front end delivers a responsive and user-friendly interface. The product is designed with scalability, security, and maintainability in mind, making it a solid foundation for future development and production deployment.

**7. Extras (Screenshots)**

This document should serve as a high-level guide to the capabilities, design, and underlying Java technologies used in the Sporty Shoes E-Commerce project.

GitHub URLs:

BE: <https://github.com/timbabs1/sporty-shoes-be>

FE: <https://github.com/timbabs1/sporty-shoes-fe>   
  
The project makes good use of concepts taught during the live sessions  
  
To run the project clone both the FE and BE

To test the project ensure you have mysql server installed and mysql workbench (for viewing data)

Open project in preferred IDE (IntelliJ recommended)   
  
For the BE ensure you have no other applications running on port 8081 afterwards proceed to create the ‘sporty’ schema/db in your mysql database. Also ensure you input the right datasource username and password

Start by running the application in IntelliJ using the play button at the top right of the editor

A screenshot of a computer

AI-generated content may be incorrect.

The data loader class should auto populate the database with values given in the class

After this you should be able to post to the different end points to run CRUD operations (an example is given below to create a user using the admin/users endpoint)

A screenshot of a computer

AI-generated content may be incorrect.

Unit tests can be found in the tests folder

We have tests covering the controller and services as described above. You can run each test to validate functionality

The Spring app has CORS set to allow origins localhost:3000, and localhost:4200 as seen in the application.properties

For the FE clone down the repository

Run npm I to install dependencies

Run npm start to start the application

Ensure the proxy url is pointing to the right host the BE application is running on (for fetching the images)

On load you will be greeted by the login screen (defaults to admin login)

A screenshot of a computer

AI-generated content may be incorrect.

FE app loaded

Test data to login as admin is username admin, password=test123

Test data to login as user is username john\_doe, passoword=test123

There is a role drop down to select what role the user wants to attain (admin/user)

The user role has 2 views, LoginView and RegisterView

Use loginview to login and registerview to register as a new user (entering email, username and password)