PROJECT OVERVIEW AND TECHNICAL ARCHITECTURE

Boutique Project

Name/Surname : Sara Kaya

Student ID: 22040102017

GitHub Repository: https://github.com/ssarakaya/boutique-projem

1. Project Brief Description "Boutique" is a sophisticated e-commerce web application designed to demonstrate the integration of modern frontend frameworks with robust backend services. The project focuses on a "Butik" fashion concept, providing a high-end user experience. It allows users to browse a dynamic catalog, manage a shopping cart, and interact with a persistent database-driven review system. The application is built using a decoupled architecture, ensuring that the React frontend and Spring Boot backend can scale and operate independently.

2. Core Technical Features

- **Dynamic Data Fetching:** Unlike static websites, all product information, images, and prices are fetched in real-time from a RESTful API. This ensures that any update in the database is immediately reflected to the user without code changes.
- **Full CRUD Review Engine:** The project implements a complete lifecycle for user feedback. Users can create reviews, view them instantly, and delete them. Each interaction is validated and processed by the backend, ensuring data consistency.
- **Intelligent Rating Logic:** A key highlight of the project is the dynamic star-rating system. The application automatically calculates the average score for each product by querying all ratings in the PostgreSQL database, providing an authentic social proof mechanism for shoppers.
- **Advanced UI Components:** Utilizing the Material UI (MUI) library, the project implements professional-grade components such as **Responsive Grids**, **Transition Fades**, **Interactive Ratings**, and **Custom Styled Paper** surfaces to mimic a real-world premium e-commerce site.

3. System Architecture

- **Frontend (React.js):** Leverages functional components and advanced Hooks (`useState`, `useEffect`, `useParams`). State management is handled efficiently to ensure smooth transitions between product listings and detail views.
- **Backend (Spring Boot):** Acts as the brain of the application, managing API endpoints, handling CORS policies, and interfacing with the database using Spring Data JPA.

- **Database (PostgreSQL):** Stores relational data for products and comments, ensuring high performance and data persistence.

EVALUATION TABLE AND PROJECT FOLDERS

4. Project Evaluation Table

The following table is provided as a requirement for the grading process, summarizing the implementation of core features.

Feature Category	Implementation Detail	Status
Research and Learning Effort	Integration of a Spring Boot REST API with a React frontend and MUI customization.	
Dynamic Product Listing	Products are dynamically fetched from the backend service and rendered on the home page.	
React Framework Usage	Efficient use of React Hooks, Context API, and modular component structure.	
Material UI Efficiency	Extensive use of MUI Grid, Stack, Rating, and Paper components for a professional UI.	
Design Principles	Implementation of responsive design, visual consistency, and user-centric navigation.	

5. Folder Organization In accordance with the project requirements, the source code is organized under a root folder named **SourceCode**, containing three sub-folders:

- **frontend:** Contains the React application. It is executable with the `npm start` command and includes all necessary dependencies.
- **backend:** A Maven-based Spring Boot project that can be run using IntelliJ IDEA.
- **database:** Contains a single, self-contained SQL export file including the database structure and initial data tables.