Introduction

This project is a comprehensive e-commerce web application that demonstrates the basic requirements of e-commerce sites and the components that need to be implemented to make a successful and functional website that consumers can use to purchase products online. The application was constructed using React and Spring boot technologies for the frontend and backend respectively. The goals of the project were to demonstrate the functionality of a backend API being used to provide data about all the products the store offers as well as user and administrator accounts that are all stored within a SQL database that the backend API is connected to. This includes user functionality such as logging in and out and being able to make API calls to send only products of certain types.

Project Structure

To go into more detail about the structure of the project, the model used was a simple frontend/backend infrastructure that contained the entire UI and visuals within the frontend and the business logic and most logical processes within the backend. The frontend provided dynamic webpages and components that render on demand, while making API calls to the backend to provide product/user information and do CRUD operations, if necessary, per the user request. The project contains a main directory with two sub directories titled NvD Frontend and NvD Backend.

Technologies Used

The app is a React and Spring Boot application. This is a modern web development stack that combines front-end and back-end technologies to create dynamic and robust web applications. React is a JavaScript library for building user interfaces, while Spring Boot is a Java-based framework for building server-side applications. React handles the user interface (UI) of the application, allowing interactive and responsive web pages. It utilizes a virtual DOM to efficiently update only the necessary parts of a web page, resulting in a smoother user experience. Spring Boot, on the other hand, powers the backend of the application. It simplifies the development of Java-based web services and provides features like dependency injection, security, and database access through Spring modules. Spring Boot's embedded web server makes it easy to deploy and run the application. These two technologies work together through RESTful API endpoints. React makes HTTP requests to the Spring Boot back-end, which processes those requests, interacts with databases, and returns data to React for rendering. This combination enables modern, single-page web applications with a responsive user interface and a scalable, secure back-end.[[1]](#footnote-1),[[2]](#footnote-2)

Frontend Analysis

The use of React in the front end allowed for an almost single page design of the web application. The UI contains a simple Navigation bar at the top of the page which includes the links to all the different pages of the web app, as well as shopping cart and authentication button. When going on the website the user is greeted with a landing page that displays banners that serve as an advertisement for the latest and greatest products that are on sale in the shop. Below the banner is a single button to guide users to the products page as quickly as possible, as well as some additional products being displayed under that are on sale or have some kind of special offer. The products page is also quite simple, offering good functionality with a clean finish. The top includes a mini navigation bar between categories in the format of “All products/CPU’s/Intel” as an example. To the left is the filter menu which includes a filter by category, brand, and price range. For users who need to find a very specific product, there is a search bar under the mini navbar, as well as a button to enlarge the product images for users who might have a hard time seeing them at their default size.

Backend Analysis

The backend of the application uses the Spring Boot framework to create an API that can be called by the front-end app. Spring Boot was used with the Model, Service, Controller template rather than the spring MVC template. The models provide Java a detailed “replication” of a model of the real-life objects that are part of the online store, such as the products and the users of the website. This includes all relevant information about both, including product names, prices, categories, and user details such as first and last name, email address, etc. With the use of these models and the ORM (Object relational mapper) provided by Spring, as new products were added through the application, Spring would automatically store them in an SQL database that is all automatically configured by Spring. This is done using Springs powerful annotations to tell the application what relations these different models have with each other, which in turn is translated in the relations within the SQL database. The services in Spring serve to have all methods and functions related to a certain model stored in a single place which makes the code more readable. The service together with a repository that stores the information about each model in the application is used to calculate any business logic the frontend might require, to do CRUD operations and to register/authenticate users properly. Finally, the Spring controller is the part that connects the backend to the frontend. This part of the application contains the endpoints that need to be called by an external application such as the React frontend app, with each endpoint containing a function that it executes when called. Each endpoint also includes the HTTP method type that is expects and returns such as “GET, POST, etc.” and the return code e.g., 200 OK, 303 See other, 404 etc. These endpoints are defined by a URL such as “localhost/API/products”, which would be used to make a call to the products controller, which could then be extended to a specific method within the controller with “/add” as an example which would call a method to add a new product. These URLs are different from what the user sees in their browser URL bar and are only used to communicate with the API.

User Functionality

The user functionality included in this application is an authentication service that users can use to register and create their own account for a more personalized experience such as a shopping cart that is saved when leaving. The Functionality of the authentication is quite simple and is implemented on the frontend with a register and login window in the browser, where new users create an account by entering their details in the provided fields such as First name, Last name, email, password, and some additional details, and logging in by entering their email and password. This information is then sent and stored in the backend + database, including the account details and what is currently in the user’s shopping cart, which is updated every time the user adds or deletes a product. The finer details of this implementation are included in the section titled “Security” in this document. Another user functionality is the payment system, which was integrated by using the popular payment service Stripe and their developer API. The API allows the application to take product model classes and convert them to a Stripe product, with all relevant naming and pricing information. The user’s shopping cart is then also converted into a Stripe shopping cart which is then finally sent to the API which generates a unique URL based on that user’s details and their specific shopping cart to generate a payment page where the user can securely enter their card details and pay for their products.

Security

Spring Boot comes with exceptional modern security features that are available with any Spring application, and this application takes advantage of those exact features. Some of these features include limiting API access to certain endpoints only to a group of authorized users or to individual users if the data in question is personalized, encoding passwords using SHA-256 as well as a salt before storing them in the database for additional security and a strict cross-origin policy (CORS). While these features limit unwanted access to the application, another feature was needed to verify if a user is who they claimed they were. This was done using an authentication tool called JSON Web token or JWT. JSON Web Token (JWT) is an open standard ([RFC 7519](https://tools.ietf.org/html/rfc7519)) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the **HMAC** algorithm) or a public/private key pair using **RSA** or **ECDSA**.[[3]](#footnote-3) This token is generated when a user registers a new account and is refreshed on a certain time interval which is set in the backend. Once a user is authenticated on the frontend, their token Is stored in the React app and sent in the request header whenever that user makes a request to the backend API, which allows Spring to compare the users token in the header to the one it has stored for that user in the database.

Product Pages & Filtering

The product pages in the application are intended to be part of the one-page application design and all maintain the same format and functionality and change dynamically depending on the product information requested by the user. The user can access the products page from anywhere on the website though the navbar, with it default URL leading to the “all products” page, which will load all products available in the store. The user can then filter these either by using the filter menu on the left side or searching from the search bar at the top of the page. The categories are dynamically loaded by filtering each category from every product available in the database and sorted into an array that is then sent to the front end, along with all corresponding product information. This allows for scalability and retains the functionality of the website even if 100 or 1000 new products are added to the database, requiring no code changes to facilitate the extra information.

Conclusion

Based on the provided information, the e-commerce website project has successfully demonstrated the fundamental requirements of an e-commerce platform, offering a robust and functional online shopping experience. With a focus on both frontend and backend technologies, this project showcases the synergy of React for the frontend and Spring Boot for the backend, resulting in a modern and responsive web application. The project's frontend, powered by React, provides an intuitive and user-friendly interface. It boasts a visually appealing landing page, efficient navigation, and user-centric features such as product filtering, search functionality, and enlarged product image viewing. The frontend's simplicity and elegance contribute to an enhanced user experience. On the backend, Spring Boot's Model-Service-Controller architecture facilitates seamless data handling. Models accurately represent real-world objects, and Spring's Object-Relational Mapping (ORM) capabilities streamline database operations. Services centralize business logic, ensuring readability and maintainability. Meanwhile, controllers define API endpoints, connecting the backend to the frontend. These endpoints enable smooth communication between the React frontend and the Spring Boot backend, supporting data retrieval, updates, and user authentication. User functionality is a key aspect of this project, offering user registration, login, and shopping cart features. Security measures include password encryption, stringent CORS policies, and the use of JSON Web Tokens (JWT) for authentication. JWTs enhance security by digitally signing and verifying user information during API interactions. Payment integration via Stripe enhances the shopping experience, allowing users to securely purchase products. The project's product pages provide a consistent, dynamic layout and offer efficient filtering options, ensuring scalability as the product catalog grows. In conclusion, this e-commerce project successfully leverages modern web development technologies to create a functional, user-friendly, and secure online shopping platform. Its frontend and backend components work harmoniously, offering a seamless shopping experience to users. With strong security measures, dynamic product pages, and responsive design, the project demonstrates a solid foundation for further development and enhancement.

References

All references from outside sources have been cited within the text and in the footnote and I have provided to links below once again.

1 - [Spring Boot](https://spring.io/projects/spring-boot#overview)

2 - [React](https://react.dev/)

3 - [JSON Web Token Introduction - jwt.io](https://jwt.io/introduction)

1. [Spring Boot](https://spring.io/projects/spring-boot#overview) [↑](#footnote-ref-1)
2. [React](https://react.dev/) [↑](#footnote-ref-2)
3. [JSON Web Token Introduction - jwt.io](https://jwt.io/introduction) [↑](#footnote-ref-3)