Project Report

Online Marketplace Platform (CUlex)

By

LIKHIL N MAIYA

23BCS11938

KRG 1B

25-08-2025

CHANDIGARH UNIVERSITY

Abstract

This report details the design, development, and implementation of a full-stack e-commerce web application, an Online Marketplace Platform. The primary objective of this project is to create a user-friendly platform that allows individuals to buy and sell products seamlessly. The system enables users to register and log in to post advertisements for their products, including details such as title, price, description, and images. All visitors can browse the listed products, but only registered users can post items for sale or engage in purchasing. The application features a personal dashboard for sellers to manage their listings.

The project is developed using a modern technology stack, featuring a Java Spring Boot backend that provides robust RESTful APIs. The frontend is a dynamic and responsive single-page application built with React.js and styled with Tailwind CSS for a clean and modern user interface. MongoDB serves as the NoSQL database for flexible data storage, with considerations for a future migration to a SQL database for more structured data handling. This report covers the project's objectives, architecture, functionalities, and potential future enhancements.

---

1. Introduction

1.1. Background

In the current digital age, online marketplaces have become an integral part of commerce. They provide a convenient platform for individuals and businesses to reach a wider audience for their products. This project aims to develop a simplified, yet powerful, online

marketplace where users can easily list items for sale and browse for products they wish to purchase.

1.2. Problem Statement

Many existing online marketplace platforms can be complex and overwhelming for casual users. There is a need for a straightforward and intuitive platform where the process of listing a product is quick and managing personal sales is simple. This project addresses that need by creating a clean, efficient, and easy-to-navigate C2C (consumer-to-consumer) e-commerce website.

1.3. Objectives

The core objectives of this project are:

- To develop a secure user authentication system for registration and login.
- To allow authenticated users to create, manage, and delete their product listings.
- To provide a public-facing gallery or dashboard where anyone can view all available products.
- To create a user-specific dashboard to view all products posted by that individual.
- To ensure a responsive and intuitive user interface for a seamless experience across different devices.
- To build a scalable backend API to handle all business logic and data management.

1.4. Scope

The scope of this project includes the following key features:

- User registration and login functionality.
- A form for creating a new product advertisement with a title, price, location, description, and image uploads.
- A main page that displays all product listings for public viewing.
- A user dashboard to view and manage personal ad listings.
- A secure logout feature.

The project does not include a payment gateway, a real-time chat system between buyer and seller, or an admin panel for site-wide management, which are considered for future development.

2. System Requirements Specification

2.1. Functional Requirements

- **User Management:**
    - Users must be able to create an account (Register).
    - Users must be able to log in to their account (Sign In).
    - Users must be able to log out of their account.
- **Product/Ad Management:**
    - Authenticated users can post a new product ad.
    - An ad must contain a title, price, receiving location, a description, and pictures.
    - Authenticated users can view all the ads they have personally posted.
- **Browsing:**
    - Any user (authenticated or not) can scroll through and view all product ads posted on the website.
- **Purchasing:**
    - A user must be signed in to initiate the process of buying a product (e.g., contacting the seller, though the contact mechanism is outside the current scope).

2.2. Non-Functional Requirements

- **Security:** User passwords must be encrypted. The application should be protected against common web vulnerabilities.
- **Performance:** The website pages should load efficiently, providing a smooth user experience.
- **Usability:** The user interface should be intuitive, responsive, and easy to use.
- **Scalability:** The backend architecture should be capable of handling an increasing number of users and listings.

2.3. Technology Stack

- **Backend:** Java Spring Boot
- **Frontend:** React.js
- **Styling:** Tailwind CSS
- **Database:** MongoDB
- **Runtime Environment:** Node.js (for React)
- **IDE/Editor:** Visual Studio Code, IntelliJ IDEA

---

3. System Design and Architecture

3.1. System Architecture

The application is built on a **Client-Server Architecture**.

- **Client-Side (Frontend):** A Single-Page Application (SPA) built with React.js. It is responsible for rendering the user interface and handling user interactions. It communicates with the backend via REST API calls. Tailwind CSS is used for utility-first styling, enabling rapid UI development.
- **Server-Side (Backend):** A RESTful API developed with Java Spring Boot. This server handles all business logic, including user authentication, data validation, and communication with the database.
- **Database:** MongoDB is used as the database. It stores user data and product listings in a JSON-like document format, which aligns well with the data structures used in modern web applications.

(Placeholder for a system architecture diagram)

3.2. Database Design

The MongoDB database consists of two main collections: users and products.

a) users Collection Schema:

```
  {
 "_id": "ObjectId",
 "username": "String",
 "email": "String",
 "password": "String (Hashed)",
 "createdAt": "Date"
}
```

b) products Collection Schema:

```
  {
 "_id": "ObjectId",
 "title": "String",
 "description": "String",
 "price": "Number",
 "location": "String",
 "imageUrls": ["String"],
 "sellerId": "ObjectId (Reference to users collection)",
 "createdAt": "Date"
}
```

Note on SQL Migration: If the database were to be migrated to SQL, the users and products collections would become tables. A foreign key relationship would be established where products.sellerId references users.id.

---

4. Implementation and Modules

The project is broken down into several key modules:

4.1. User Authentication Module (Spring Boot & React)

- **Functionality:** Handles user registration, login, and session management (e.g., using JSON Web Tokens - JWT).
- **Backend:** Spring Security is used to manage authentication and authorization. It includes endpoints like /api/auth/register and /api/auth/login. Passwords are hashed using BCrypt before being stored in the database.
- **Frontend:** React components for Sign In and Register forms capture user input and send it to the backend. On successful login, a token is stored in the browser's local storage to authenticate subsequent requests.

4.2. Product Management Module (Spring Boot & React)

- **Functionality:** Allows for the creation and retrieval of product listings.
- **Backend:** A ProductController in Spring Boot defines REST endpoints like POST /api/products to create a new ad and GET /api/products/user/{userId} to fetch ads for a specific user.

- **Frontend:** A dedicated page with a form built in React allows users to input product details and upload images.

4.3. Main Dashboard/Feed Module (React)

- **Functionality:** Displays all product listings to all users.
- **Backend:** A public endpoint GET /api/products is available to fetch all listings from the database, possibly with pagination for performance.
- **Frontend:** The main page of the React application calls this endpoint and maps over the returned data to display each product as a card, showing its image, title, price, and location.

4.4. User Dashboard Module (React)

- **Functionality:** A private page that shows all products listed by the currently logged-in user.
- **Backend:** A protected endpoint, GET /api/products/my-ads, retrieves listings where the sellerId matches the ID of the authenticated user.
- **Frontend:** A route protected for authenticated users fetches and displays the list of their personal ads, with options to edit or delete them.

---

5. Testing

To ensure the application's quality and reliability, several testing strategies were employed:

- **Unit Testing:** Performed on the backend to test individual components and services in the Spring Boot application (e.g., testing that the user service correctly hashes a password).
- **API Testing:** Tools like Postman were used to test the REST API endpoints to ensure they behave as expected, return the correct status codes, and handle errors gracefully.
- **Manual Frontend Testing:** The user interface was manually tested across different web browsers (Chrome, Firefox) to check for functionality, responsiveness, and usability issues. This involved testing the user flow from registration to posting an ad.

---

6. Conclusion and Future Scope

6.1. Conclusion

This project successfully delivered a functional Online Marketplace Platform that meets all the core objectives. It provides a clean, user-friendly interface for browsing products and a secure, streamlined process for registered users to sell their items. The choice of the Java Spring Boot, React, and MongoDB stack proved to be effective for building a modern, scalable web application.

6.2. Future Scope

The platform has a strong foundation that can be extended with many new features in the future:

- **Payment Gateway Integration:** Incorporating Stripe or PayPal to handle transactions directly on the platform.
- **Search and Filter Functionality:** Adding a search bar and filters to allow users to find products by name, category, price range, or location.
- **User-to-User Messaging:** A real-time chat system for buyers and sellers to communicate.
- **Admin Panel:** A dashboard for administrators to manage users, monitor listings, and oversee the platform.
- **Product Categories:** Introducing categories to better organize product listings.
- **Deployment:** Deploying the application to a cloud service like AWS or Heroku for public access.