Software Requirement Specification (SRS)

1. Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for the Ecommerce Application. It outlines the application's features, functional and non-functional requirements, and technical specifications to ensure clarity in development and implementation.

1.2 Scope

The Ecommerce Application is a full-featured web application that enables users to browse, search, filter, and purchase products securely. The application also includes an admin dashboard to manage products, categories, orders, and users.

1.3 Definitions and Acronyms

- **Ecommerce Application** – A web-based platform for online shopping.
- **Admin** – A user role with access to manage products, users, and orders.
- **User** – A registered customer who can browse and purchase products.
- **JWT** – JSON Web Token, used for secure authentication.
- **CRUD** – Create, Read, Update, and Delete operations.

1.4 References

- ReactJS official documentation
- Node.js & Express.js official documentation
- MongoDB official documentation
- Tailwind CSS & ShadCN documentation
- Cloudinary & Paypal API documentation

2. Overall Description

2.1 Product Perspective

This application is a standalone product that provides a seamless online shopping experience. The frontend is built using ReactJS, while the backend is powered by Node.js and Express.js, with MongoDB as the database.

2.2 Product Functions

- **User Authentication** – Registration, login, and JWT-based authentication.
- **Product Management** – Browse, search, and filter products.
- **Shopping Cart** – Add, update, and remove items.
- **Order Management** – Secure checkout and order tracking.
- **Admin Panel** – Manage products, users, and orders.
- **Payment Integration** – Secure transactions via PayPal.
- **Responsive Design** – Mobile and desktop-friendly interface.

2.3 User Characteristics

- **Customers** – Users who browse, add products to the cart, and place orders.
- **Admins** – Users with higher privileges who manage the store.

2.4 Constraints

- The system should be secure and prevent unauthorized access.
- The platform should support high traffic and handle multiple concurrent users.
- The database should ensure data consistency and integrity.

3. Functional Requirements

3.1 User Authentication

- Users should be able to register, login, and log out securely.
- JWT-based authentication should be used for session management.

3.2 Product Management

- Users can browse products by categories, price, and brands.
- Users can search for products using keywords.
- Admins can add, update, and delete products.

3.3 Shopping Cart

- Users can add, update, and remove items from the cart.
- Cart items should persist across sessions.

3.4 Order Management

- Users can place orders with a secure checkout process.
- Users can view order history and track order status.
- Admins can view, update, and manage orders.

3.5 Payment Integration

- Users should be able to make payments via PayPal.
- Payment status should be updated upon successful transaction.

3.6 Admin Dashboard

- Admins can manage products, users, and orders.
- Admins can view sales reports and analytics.

4. Non-Functional Requirements

4.1 Performance Requirements

- The system should support at least 1000 concurrent users.
- The response time for API requests should not exceed 2 seconds.

4.2 Security Requirements

- User passwords should be encrypted using bcrypt.
- Authentication tokens should be securely stored.

- Admin access should be restricted using role-based authentication.

4.3 Usability Requirements

- The UI should be intuitive and responsive.
- The application should work on both mobile and desktop.

4.4 Scalability Requirements

- The system should be deployable on cloud platforms like Render.
- The database should be optimized for large datasets.

5. External Interfaces

5.1 User Interface

- Frontend developed using ReactJS, Tailwind CSS, and ShadCN.

5.2 Hardware Interfaces

- The application should be accessible from any modern device with an internet connection.

5.3 Software Interfaces

- Backend: Node.js and Express.js.
- Database: MongoDB.
- Payment: PayPal API.
- Cloud Storage: Cloudinary API.

6. System Architecture

6.1 Frontend

- ReactJS with Redux for state management.
- Tailwind CSS and ShadCN for styling.

6.2 Backend

- Node.js with Express.js.
- MongoDB with Mongoose ORM.
- JWT for authentication.

6.3 Deployment

- Frontend deployed on Vercel.
- Backend deployed on Render.
- Database hosted on MongoDB Atlas.

7. Glossary

- **API** – Application Programming Interface
- **CRUD** – Create, Read, Update, Delete
- **JWT** – JSON Web Token
- **ORM** – Object-Relational Mapping
- **UI/UX** – User Interface/User Experience

8. Conclusion

This SRS document provides a detailed overview of the Ecommerce Application, including its features, technical requirements, and architecture. It serves as a reference for developers, testers, and stakeholders involved in the project development lifecycle.