**Software Requirement Specification (SRS) Document**

1. **Introduction**

**1.1 Purpose**

The purpose of this document is to define the software requirements for an e-commerce website resembling "[StarTech](https://www.startech.com.bd/)". This platform aims to facilitate online shopping for customers while providing a comprehensive admin dashboard for efficient business management. The project intends to deliver a scalable, user-friendly, and secure application that meets the growing needs of an online retail business.

**1.2 Scope**

This e-commerce website will include the following features:

* **Customer Features**: Product browsing, search, and purchase options, along with a wish list and shopping cart.
* **Admin Features**: Tools for managing users, products, orders, and generating reports.
* Integration of payment gateways for seamless transactions.
* Responsive design ensuring compatibility across desktop, tablet, and mobile devices.
* Real-time order tracking and notifications for both customers and admins.

**1.3 Audience**

This document is intended for the following stakeholders:

* Development Team (Frontend and Backend Developers)
* Project Managers
* Quality Assurance Team
* HR and Business Analysts
* End-users (Customers and Administrators)

**1.4 Definitions and Acronyms**

* **MERN**: MongoDB, Express.js, React, Node.js
* **CRUD**: Create, Read, Update, Delete
* **UI/UX**: User Interface / User Experience
* **API**: Application Programming Interface
* **JWT**: JSON Web Token
* **CI/CD**: Continuous Integration/Continuous Deployment

**2. Overall Description**

**2.1 Product Perspective**

The e-commerce platform will closely mimic the design and functionality of "StarTech" to ensure a familiar user experience. It will use a MERN stack architecture to deliver a modern, high-performance application.

**2.2 Product Features**

**Customer Features**

* User authentication and profile management.
* Search and filter functionality for products by categories, price range, and brands.
* Secure payment options integrated with gateways like Stripe and PayPal.
* Wishlist, cart management, and order history tracking.
* Real-time notifications for order confirmation and shipment updates.

**Admin Features**

* A dashboard displaying key business metrics such as sales, active users, and pending orders.
* CRUD operations for products, categories, and user management.
* Order processing, including viewing, updating, and tracking statuses.
* Reporting tools to analyze sales performance and inventory levels.

**2.3 Constraints**

* **Performance**: The system must support at least 500 simultaneous users.
* **Browser Compatibility**: Ensure smooth performance across Chrome, Firefox, Safari, and Edge.
* **Mobile Responsiveness**: Must function seamlessly on devices of varying screen sizes.
* **Data Consistency**: Use MongoDB for efficient data management.

**2.4 Assumptions**

* All users will have access to a stable internet connection.
* Admin users will possess a basic understanding of digital interfaces.

**3. Functional Requirements**

**3.1 User Authentication**

* **Registration and Login**: Users must register and log in using secure credentials.
* **JWT Tokens**: Token-based authentication for secure access.

**3.2 Product Management**

* Customers can view detailed product descriptions, specifications, and reviews.
* Admins can create, edit, and delete products and categories via the admin dashboard.

**3.3 Order Management**

* Customers can place orders, track their progress, and view history.
* Admins can manage order statuses such as "pending," "processed," and "shipped."

**3.4 Payment Integration**

* Integration with payment gateways such as Stripe and PayPal.
* Real-time payment verification and status updates for customers.

**3.5 Notifications**

* Email and SMS notifications for order confirmations and shipping updates.
* Admin alerts for low inventory levels and pending orders.

**4. Non-Functional Requirements**

**4.1 Performance**

* The website should load within 3 seconds on a standard broadband connection.
* Ensure smooth operation for 500+ concurrent users.

**4.2 Security**

* Implement HTTPS for secure communication.
* Encrypt sensitive data such as passwords and payment details using industry-standard algorithms.

**4.3 Usability**

* Intuitive navigation and clear CTAs (Call-to-Actions).
* Provide user guides or tooltips for complex operations.

**4.4 Scalability**

* The backend should support scaling up to accommodate increased traffic and data growth.

**5. System Design**

**5.1 Architecture**

* **Frontend**: Built using React.js with Tailwind CSS for design.
* **Backend**: Node.js and Express.js to handle server-side operations.
* **Database**: MongoDB with Mongoose for schema modeling and data manipulation.

**5.2 Folder Structure**

**Backend**

* config/db.js: Database connection setup.
* controllers/userController.js: User-related logic.
* middlewares/: Custom middleware for handling errors and authentication.
* models/userModel.js: Mongoose schema for user data.
* routes/: Defines backend API routes for users, products, and orders.

**Frontend**

* src/App.jsx: Main React component.
* src/components/: Reusable components like navbar, product cards, and forms.
* tailwind.config.js: Tailwind CSS configuration for consistent styling.

**5.3 API Endpoints**

**User APIs**

* POST /api/register: Create a new user.
* POST /api/login: Authenticate user.
* GET /api/users: Retrieve all users (admin-only access).

**Product APIs**

* GET /api/products: Fetch all products.
* POST /api/products: Add a new product (admin-only access).
* PUT /api/products/:id: Update product details (admin-only access).

**Order APIs**

* POST /api/orders: Create a new order.
* GET /api/orders/:id: Retrieve details of a specific order.

**6. UI/UX Design**

**6.1 Wireframes**

Wireframes will replicate the structure and layout of "StarTech" for familiarity and ease of use.

**6.2 Responsive Design**

* Tailwind CSS will be utilized to ensure a responsive design.
* Rigorous testing across devices (mobile, tablet, and desktop).

**7. Testing**

**7.1 Testing Plan**

* **Unit Testing**: Validate individual components and modules.
* **Integration Testing**: Ensure smooth interactions between modules.
* **End-to-End Testing**: Test user journeys, including product browsing and order placement.
* **Load Testing**: Assess system performance under peak traffic.

**8. Deployment**

**8.1 Environments**

* **Development**: Local testing with vite for frontend and nodemon for backend.
* **Production**: Deploy to cloud services such as AWS or Vercel.

**8.2 CI/CD Pipeline**

* Use GitHub Actions for automated builds, testing, and deployment.

**9. Risks and Mitigations**

* **Data Loss**: Implement daily backups for MongoDB.
* **Downtime**: Host the website on platforms with 99.9% uptime guarantees.
* **Security Threats**: Use OWASP-recommended practices to safeguard the application.

**10. Appendix**

**10.1 References**

* "[StarTech](https://www.startech.com.bd/)" as the reference site.
* Official documentation for MERN stack and Tailwind CSS.

**10.2 Tools Used**

* **Frontend**: React.js, Tailwind CSS.
* **Backend**: Node.js, Express.js, MongoDB.
* **Version Control**: GitHub for versioning and collaboration.
* **Development Tools**: VS Code, Postman, and Docker for containerization.

**Prepared by:** Minhaz Reza , Arnab Chakraborty , Yeasin Arafat  
**Date:** 2024-12-15