**Software Requirements Specification for TechNest**

**1. Introduction**

**1.1 Purpose**

This document outlines the requirements for TechNest, a scalable web application designed to provide a platform for shops to sell technology products in a stock market-like environment, enabling customers to browse, purchase, and manage tech products efficiently.

**1.2 Scope**

TechNest is an e-commerce platform that connects multiple tech shops with customers, offering features like user authentication, product browsing, cart management, and secure payments. The application aims to provide a seamless user experience, scalability for high traffic, and flexibility for shop owners to manage inventory.

**1.3 Definitions, Acronyms, and Abbreviations**

* **TechNest**: The web application for tech product sales.
* **Shop**: A vendor or retailer selling technology products.
* **MVP**: Minimum Viable Product, the initial version with core features.
* **Clerk**: Authentication service for user login/signup.
* **Stripe**: Payment processing service.
* **CookieJS**: Library for managing cart data in browser cookies.

**1.4 References**

* NextJS Documentation: https://nextjs.org/docs
* Clerk Documentation: https://clerk.com/docs
* Stripe Documentation: https://stripe.com/docs

**2. Overall Description**

**2.1 Product Perspective**

TechNest is a standalone web application with a REST API, built using modern web technologies to ensure scalability and maintainability. It integrates third-party services like Clerk for authentication and Stripe for payments.

**2.2 Product Functions**

* User authentication (login/signup) via Clerk.
* Product browsing and search for tech products from multiple shops.
* Cart management using CookieJS for adding, removing, and persisting items.
* Secure payment processing via Stripe.
* Shop owner dashboard for managing inventory (MVP feature).
* Progress tracking for orders (MVP feature).

**2.3 User Classes and Characteristics**

* **Customers**: Browse products, add to cart, and make purchases. Require an intuitive interface.
* **Shop Owners**: Manage product listings and view sales. Require a secure dashboard.
* **Admins**: Oversee platform operations, including user management and dispute resolution.

**2.4 Operating Environment**

* **Client**: Web browsers (Chrome, Firefox, Safari) on desktop and mobile devices.
* **Server**: NodeJS (v18+) runtime with NextJS (v15.2.0) and Turbopack compilers.
* **Database**: PostgreSQL or MongoDB (TBD based on scalability needs).

**3. Functional Requirements**

**3.1 User Authentication (Clerk)**

* **Description**: Users can sign up or log in using email/password or OAuth (Google, GitHub).
* **Input**: User credentials or OAuth tokens.
* **Output**: Authenticated session with JWT token.
* **Error Handling**: Invalid credentials prompt error messages.

**3.2 Product Browsing**

* **Description**: Users can view and search tech products listed by shops.
* **Input**: Search queries, filters (e.g., price, category).
* **Output**: List of products with details (name, price, shop).

**3.3 Cart Management (CookieJS)**

* **Description**: Users can add/remove items to/from a cart, persisted via browser cookies.
* **Input**: Product selection, quantity.
* **Output**: Updated cart view with total price.

**3.4 Payment Processing (Stripe)**

* **Description**: Users can pay for cart items using Stripe.
* **Input**: Payment details (card, digital wallet).
* **Output**: Payment confirmation or error message.

**3.5 Shop Owner Dashboard (MVP)**

* **Description**: Shop owners can add, edit, or remove product listings.
* **Input**: Product details (name, price, description, stock).
* **Output**: Updated product catalog.

**3.6 Order Progress Tracking (MVP)**

* **Description**: Users can track the status of their orders (e.g., processing, shipped).
* **Input**: Order ID.
* **Output**: Order status and estimated delivery.

**4. Non-Functional Requirements**

**4.1 Performance**

* The application must handle up to 10,000 concurrent users with response times under 2 seconds.
* Page load times should not exceed 3 seconds under normal conditions.

**4.2 Scalability**

* The system must support horizontal scaling via cloud providers (e.g., Vercel, AWS).
* Database choice (PostgreSQL/MongoDB) will prioritize scalability for product data.

**4.3 Security**

* User data must be encrypted (HTTPS, JWT).
* Compliance with GDPR and PCI-DSS for payment processing.

**4.4 Usability**

* The interface must be responsive and accessible (WCAG 2.1 compliant).
* Intuitive navigation for both customers and shop owners.

**5. System Architecture**

**5.1 Tech Stack**

* **Programming Languages**: TypeScript (TSX for React components).
* **Markups**: HTML, CSS.
* **Component Packages**: Shadcn UI for reusable UI components.
* **Core Stack**:
  + **Runtime Environment**: NodeJS (v18+).
  + **Package Management**: NPM (v11.9.1).
  + **Library**: ReactJS (v19.1.0).
  + **Framework**: NextJS (v15.2.0) with App Router, TypeScript, ‘@’ aliases, and Turbopack compilers.
* **Database**: PostgreSQL or MongoDB (TBD).
* **APIs**: REST API provided by NextJS for client-server communication.

### Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Technology** | **Version** | **Purpose** |
| Programming Language | TypeScript | Latest | Static typing for scalable and maintainable code |
| Programming Language | TSX | Latest | TypeScript with JSX for React component development |
| Markup Language | HTML | 5 | Structure and content of web pages |
| Markup Language | CSS | Latest | Styling and layout of web pages |
| Component Package | Shadcn UI | Latest | Reusable UI components for consistent and modern design |
| Runtime Environment | NodeJS | 24.0.1 | Server-side JavaScript runtime for executing NextJS |
| Package Management | NPM | 11.9.1 | Dependency management for JavaScript packages |
| Library | ReactJS | 19.1.0 | Building dynamic and interactive user interfaces |
| Framework | NextJS | 15.2.0 | Server-side rendering, static site generation, and REST API routes |
| Framework Configuration | App Router | N/A | NextJS routing system for optimized page navigation |
| Framework Configuration | ‘@’ Aliases | N/A | Simplified import paths for cleaner code organization |
| Framework Configuration | Turbopack | N/A | Rust-based compiler for faster builds and development |
| Database | PostgreSQL | Latest | Relational database for structured data (TBD) |
| Database | MongoDB | Latest | NoSQL database for flexible, scalable data storage (TBD) |
| API | REST  (NextJS) | 15.2.0 | Backend API for client-server communication |
| Third-Party Service | Clerk | Latest | User authentication and session management |
| Third-Party Service | Stripe | Latest | Secure payment processing for transactions |
| Third-Party Service | CookieJS | Latest | Client-side cart management using browser cookies |

**5.2 System Components**

* **Frontend**: ReactJS with Shadcn UI for dynamic, responsive interfaces.
* **Backend**: NextJS API routes for handling requests.
* **Database**: Stores user, product, and order data.
* **Third-Party Services**: Clerk (authentication), Stripe (payments), CookieJS (cart).

**6. Constraints**

* Budget and timeline constraints set by Lixril agency.
* No ORM will be used; database interactions will use raw queries or a lightweight library.
* The application must run on modern browsers without requiring specific plugins.

**7. Assumptions**

* Users have access to stable internet connections.
* Shop owners are familiar with basic e-commerce operations.
* Third-party services (Clerk, Stripe) are reliable and available.

**8. Project Metadata**

* **Name**: TechNest
* **Agency**: Lixril
* **Author**: Abdul Quddus
* **Email**: qudduslarek@proton.me