**Project Design Phase**

**Solution Architecture**

|  |  |
| --- | --- |
| Date | 20 February 2026 |
| Team ID | LTVIP2026TMIDS82947 |
| Project Name | ShopSmart |
| Maximum Marks | 4 Marks |

**🧠 Solution Architecture: ShopSmart**

The solution architecture for ShopSmart is designed to provide an end-to-end digital grocery shopping experience. It ensures smooth interaction between users, the database, and the admin backend through a scalable and modular structure built on the MERN stack (MongoDB, Express.js, React.js, Node.js). The system supports real-time product browsing, cart operations, order management, admin controls, and secure authentication**.**

**🔹 Architecture Objectives**

* To provide users with a seamless grocery shopping experience online.
* To enable sellers/admins to manage products, inventory, and orders digitally.
* To offer responsive design, secure access, and real-time updates.
* To establish a modular system that can scale with increasing user base and features.

**🔧 Components & Descriptions**

| **Component** | **Description** |
| --- | --- |
| **User Interface** | A React.js-based frontend for customers and admins to interact with the system. |
| **Authentication System** | Uses JWT for secure login, role-based access for users and admins. |
| **Product Management** | Allows admins to add, edit, delete products and categories. |
| **Cart & Order Module** | Handles add-to-cart, checkout, and order placement operations. |
| **Feedback System** | Allows customers to submit product reviews and feedback. |
| **Admin Dashboard** | Visual panel for sellers to view orders, users, inventory, and feedback. |
| **Database Layer** | MongoDB collections for users, products, carts, orders, categories, and reviews. |

**🔁 Architecture Flow**

1. User accesses the application via React frontend.
2. Frontend sends HTTP requests to backend APIs (Node.js + Express).
3. Backend handles logic and interacts with MongoDB for data storage/retrieval.
4. Based on user role, either shopping interface (user) or admin dashboard is served.
5. Orders and feedback are processed and saved in the database.
6. UI updates in real-time to reflect inventory, order status, and user activity.

**🛠 Technology Stack**

* **Frontend:** React.js, HTML, CSS, Bootstrap
* **Backend:** Node.js, Express.js
* **Database:** MongoDB
* **Authentication:** JSON Web Tokens (JWT)
* **Deployment:** Vercel (Frontend), Render/Heroku (Backend), MongoDB Atlas

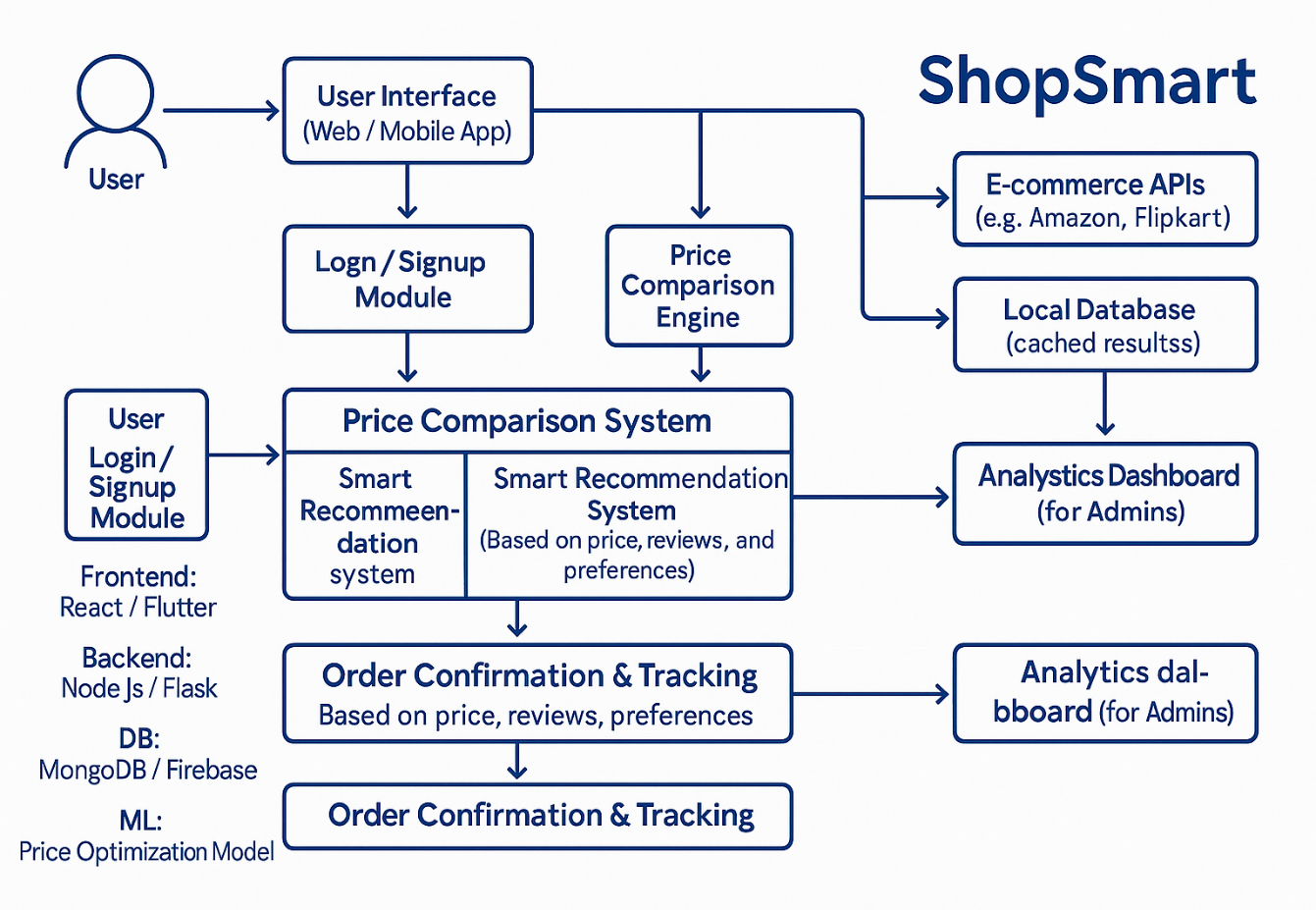
**🚀 Development Phases**

1. UI/UX Design and React Setup
2. Backend API and MongoDB Integration
3. Authentication (JWT) and Role-Based Access
4. Cart, Checkout, and Order Flow Implementation
5. Admin Dashboard Development
6. Testing, Optimization, and Deployment

**📈 Scalability Considerations**

* Follows modular and RESTful API architecture
* MongoDB supports horizontal scaling for growing data
* Admin and user portals are separated for role-specific experiences
* Future integration planned for:
  + Mobile App (React Native)
  + Online Payments (Stripe/Razorpay)
  + Real-time Delivery Tracking (GPS APIs)
  + AI-based Product Recommendations

**Solution Architecture Diagram:**

****