Full Stack Development with MERN

Phase 3 : Requirement Analysis

Project Title – ShopEZ : One-Stop shop for online purchases

Team Members :

1. **Diviz Pandey (Team Lead) – Backend Developer**
2. **Bhavya Jain – Frontend Developer**
3. **Karmanya Batra – Database and Deployment**
4. **Anaswar L – Tester**

Team ID - SWTID1743669870

3.1 Customer Journey Map

The customer journey was carefully mapped out to ensure a smooth and efficient experience for both end users (shoppers) and admins (store operators). The map outlines every stage a customer goes through — from the first interaction to completing an order and post-purchase engagement.

Customer Flow:

1. **Awareness:**
    - User lands on the homepage or arrives via search/social links.
    - Site loads quickly; key categories are visible above the fold.
2. **Consideration:**
    - Users browse product categories, view detailed product pages.
    - Filtering and sorting help refine choices.
    - Product images, specs, reviews, and price transparency help build trust.
3. **Purchase Decision:**
    - Items are added to the cart.
    - User registers or logs in for checkout.
    - Secure checkout collects address, payment method, and order confirmation.
4. **Post-Purchase:**
    - Order is tracked from dashboard.
    - Email notifications (planned) confirm the order.
    - Wishlist and order history provide engagement continuity.
5. **Re-engagement:**
    - Future version will support recommendations based on past orders.
    - Admin may offer discounts or promos (future scope).

This linear yet feedback-friendly journey was central to feature planning and UI wireframing.

3.2 Solution Requirements

We categorized our requirements into **Functional** and **Non-Functional** to ensure clarity and proper development prioritization.

Functional Requirements

Feature	**Description**
User Auth	Registration, Login, JWT-based token handling
Product Browsing	Filterable, searchable catalog with images, descriptions, stock
Shopping Cart	Add/remove items, quantity adjustment, total price calc
Order System	Checkout flow, address entry, and backend order creation
Admin Dashboard	Manage users, orders, products
User Dashboard	View past orders, account details, and wishlist (optional)
Search System	Autocomplete & keyword match for fast access

Non-Functional Requirements

- **Security:**
  - JWT authentication
  - Password hashing with bcrypt
  - Role-based access (user/admin)
  - Environment-configured secrets
- **Performance:**
  - Sub-3s average page load time
  - Efficient image rendering
  - Database indexing on key fields (product name, email)
- **Scalability:**
  - Modular folder structure
  - Separated frontend/backend logic
  - MongoDB's schema-less flexibility for future enhancements
- **Maintainability:**
  - RESTful API design
  - Reusable React components

    - Redux for global state management
- **Responsiveness:**
    - Tailwind CSS used to ensure UI works across all screen sizes
- **Reliability:**
    - Server structured for production with robust error handling and logging

---

3.3 Data Flow Diagram (DFD)

To visualize the interactions within the ShopEZ ecosystem, we can conceptualize the **Data Flow** between client, backend, and database as follows:

Level 0: Context DFD

[User] ---> (ShopEZ System) ---> [Database]

Level 1: Key System Components

1. **User Registers/Login**
    - Data: Email, Password → Auth Service → JWT Token → Stored in local/session storage
2. **Browse Products**
    - Request from UI → Product API → MongoDB fetch → Return product list
3. **Add to Cart / Place Order**
    - UI → Cart API → Update cart state
    - UI → Order API → Save order to MongoDB → Return order confirmation
4. **Admin Product Management**
    - Admin Dashboard → Product API (POST/PUT/DELETE) → Database updated accordingly

This flow demonstrates how the system maintains **separation of concerns** and handles **asynchronous state updates** through Redux Toolkit and AsyncThunk.

3.4 Technology Stack

ShopEZ is built entirely on the **MERN stack**, enhanced with additional tools to support development efficiency, deployment, and user security.

Core Stack

Layer	**Technology**	**Purpose**
Frontend	React.js	SPA, component-based architecture
Styling	Tailwind CSS	Responsive, utility-first CSS
State Mgmt	Redux Toolkit	Global state for cart, auth, products
Routing	react-router-dom	Page-level routing
Backend	Node.js + Express	API server, REST endpoints
Auth	JWT + bcryptjs	Secure token-based login, role-based control
Database	MongoDB + Mongoose	NoSQL DB with schema definitions

Supporting Tools

Tool	**Purpose**
Postman	API testing and mock data requests
Nodemailer	Order confirmations and password reset emails
Vite	Frontend build tool for fast reloads
.env Configuration	Secure management of sensitive environment data
Git/GitHub	Source control and versioning

3.5 Entity Relationships

Using MongoDB (via Mongoose), we defined several collections with relational references:

1. **Users**
   Fields: name, email, password, role, address, resetToken
   - Used for authentication, order reference, and cart ownership.
2. **Products**
   Fields: title, description, price, stock, image, category
   - Core catalog items, accessed by both user and admin flows.
3. **Orders**
   Fields: userId, items[], totalAmount, shippingAddress, status
   - Each order links back to a user and contains product metadata.

4. **Categories**
   Fields: name, description
    - Used to organize and filter products.
5. **Wishlists**
   Fields: userId, productIds[]
    - Planned feature for favorites.

By referencing IDs between collections, MongoDB enables flexible, scalable schema linking while maintaining performance through indexing.

3.6 Summary

The Requirement Analysis phase helped crystallize what ShopEZ needed to deliver in terms of features, architecture, and system performance. From journey mapping and functional expectations to database design and tech stack finalization, this phase became the bridge between ideation and system design.

It allowed us to:

- Define clear development goals.
- Establish secure, scalable backend logic.
- Ensure the frontend remains intuitive and fast.
- Avoid scope creep by planning for optional features separately.

The next phase — **Project Design** — builds directly on these foundations, transforming requirements into visual architecture and proposed solutions.