Project Design Phase

Solution Architecture

Date: 20 JULY 2025
Team ID: LTVIP2025TMID48243
Project Name: Shopmart
Maximum Marks: 4 Marks

Solution Architecture Overview

The solution architecture for **Shopmart** aims to provide a secure, scalable, and modular backend for e-commerce operations. It addresses the challenges of repetitive backend setup, lack of reusable codebases, and the need for role-based access and feedback systems. This architecture defines how various components of the system work together to deliver seamless backend functionality for user authentication, product/order management, and customer interaction.

Objectives of the Solution Architecture

- **Solve Business Problems:** Provide small teams and developers with a plug-and-play backend for online shopping operations, minimizing development time and avoiding redundant code.
- **Clear System Representation:** Illustrate the structure, data flow, and responsibilities of each core component including routes, models, middleware, and controllers.
- **Phase-wise Development:** Divide development into modular components: auth module, product/order modules, feedback module, and admin middleware.
- **Well-Defined Specifications:** Define how to integrate authentication, database models, API routing, and protected access layers clearly.

Architecture Components & Flow Description

1. **Frontend Interface (Future Scope)**
    - Will consume REST APIs to allow customer and admin interactions such as browsing products, placing orders, or viewing feedback.
2. **Backend Logic (Node.js + Express.js)**
    - **User Authentication (JWT)**
      Handles registration, login, and token-based route protection.
    - **Product & Order APIs**
      RESTful routes to manage product catalog and customer orders.
    - **Feedback Controller**
      Stores and retrieves user-submitted feedback.

   - **Role-Based Access Middleware**
     Restricts admin-only actions.

3. **Database Integration (MongoDB via Mongoose)**
   - Defines schema for Users, Products, Orders, and Feedback.
   - Ensures data persistence, indexing, and efficient querying.
4. **Deployment**
   - Local development with nodemon.
   - Production-ready for deployment on platforms like **Render**, **Vercel (API routes)**, or **AWS EC2 with PM2**.
   - Environment configuration using .env.
5. **API Architecture**
   - Follows **REST principles** for clarity, scalability, and frontend compatibility.
   - Routes grouped under /api/users, /api/products, /api/orders, /api/feedback.
6. **Admin Control Panel (Optional/Future)**
   - Admin can view users, delete products, and moderate feedback.
   - Role-checking via isAdmin flag in JWT middleware.

Diagram Suggestion

Frontend (React or HTML UI)

↓

REST API (Express.js)

↓

Controllers ↔ Middleware ↔ Routes

↓

MongoDB (Models via Mongoose)