**Project High-Level Design Demo**

**Project Name:** E-Commerce Application  
**Purpose:** Present the architectural overview, key modules, and data flows of the e-commerce system to stakeholders and development teams.

**📌 1. Objective of the Application**

* Online shopping platform for customers to browse, search, and purchase products.
* Secure payment integration for completing purchases.
* Admin portal for managing products, orders, customers, and reporting.

**📌 2. High-Level Architecture**

**(Present a diagram or visualize as layers):**

[Customer Browser / Mobile App]

|

API Gateway

|

---------------------------------------

| Front-End (React.js / Angular) |

| Back-End APIs (Node.js / Spring) |

| Database (MySQL / PostgreSQL) |

---------------------------------------

|

Payment Gateway / SMS / Email

|

Admin Portal (Web UI)

* **Client Layer:** Web browsers or mobile apps.
* **API Gateway:** Manages API traffic.
* **Front-End:** Product catalog, shopping cart, checkout UI.
* **Backend APIs:** Business logic & data handling.
* **Database:** Transactional data storage.
* **Integrations:** Payment, notifications, reports.
* **Admin Portal:** Backend management console.

**📌 3. Key Modules**

| **Module** | **Purpose** |
| --- | --- |
| User Management | Registration, login, profile updates |
| Product Catalog | Browsing, searching, filtering products |
| Shopping Cart | Add/remove products, update quantities |
| Checkout & Payments | Secure order placement and payments |
| Order Management | View, track, and cancel orders |
| Notifications | SMS/email updates on orders |
| Admin Panel | Product, order, and customer management |
| Reporting | Sales reports, inventory reports |

**📌 4. Data Flow**

* User browses → Adds products to cart → Checkout
* System processes payment via gateway
* Order details stored in DB
* Notifications sent via SMS/Email
* Admin monitors orders via dashboard

**📌 5. Technology Stack**

| **Component** | **Technology** |
| --- | --- |
| Front-End | React.js / Angular |
| Back-End | Node.js / Java Spring Boot |
| Database | MySQL / PostgreSQL |
| Hosting | AWS / Azure / GCP |
| Payment Gateway | Razorpay / PayU / Stripe |
| Notifications | Twilio / AWS SNS |
| Admin Panel | React.js / Angular |
| Deployment | Docker + Jenkins / GitLab |

**📌 6. Security Considerations**

* HTTPS enforced
* JWT authentication
* Password encryption (bcrypt)
* PCI-DSS compliance for payment data
* Role-based access control (RBAC) for admin portal

**📌 7. Benefits of the Architecture**

* Scalable via cloud auto-scaling
* Modular backend services for easy updates
* Secure and reliable payment processing
* Real-time order tracking and notifications
* Admin dashboard for operational control

**✅ Demo Summary**

* Modular, scalable design ready for production.
* Clear separation of user and admin workflows.
* Secure transactions and customer data protection.
* Cloud-native deployment with automation support.