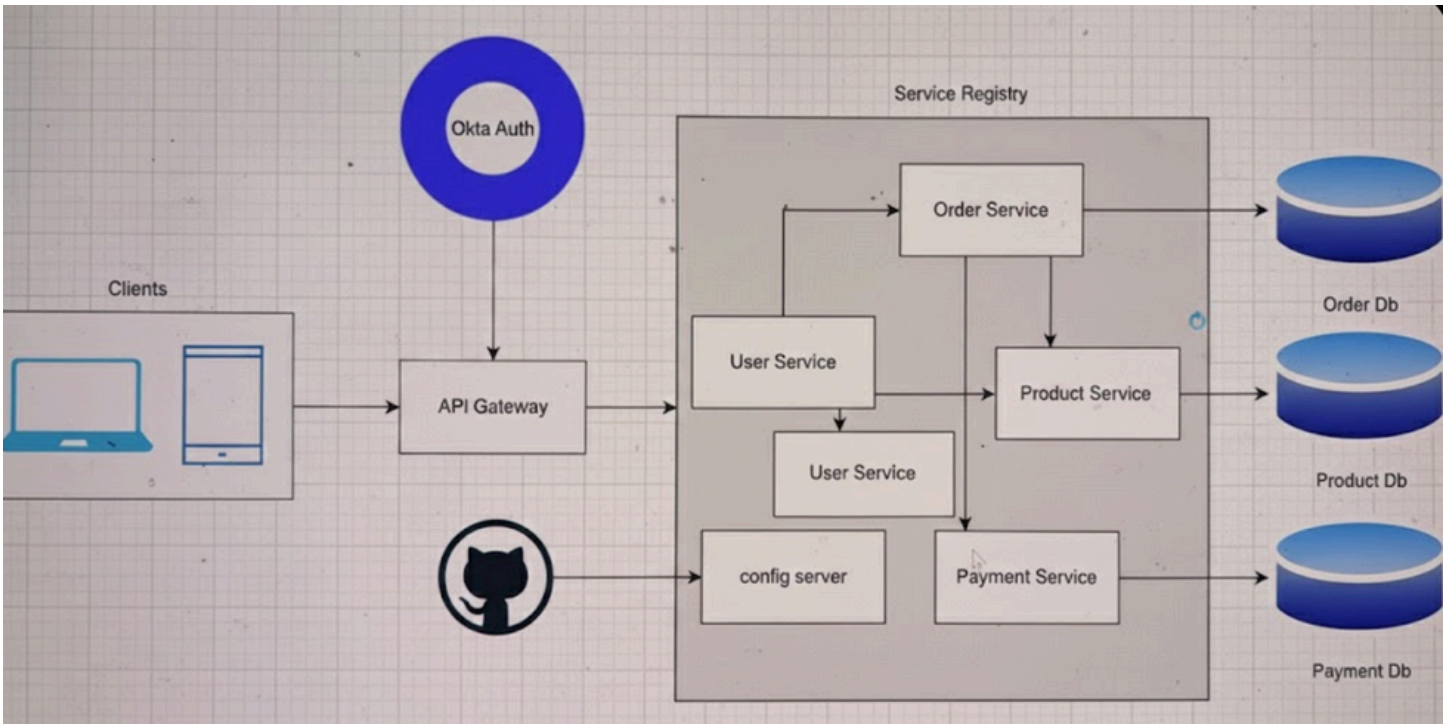

Shopping Cart System – Microservices Architecture (Quick Notes)

🛒 Goal

Build an end-to-end e-commerce application using microservices where users can:

- **Browse products**
- **Place orders**
- **Make secure payments**

Core Functional Services

👉 **Database-per-Service pattern (loose coupling + independent scaling)**

🔹 Product Service

- **Responsibility: Manages product catalog (iPhones, MacBooks, etc.)**
- **Features:**
  - **Add / update products**
  - **Manage stock quantity**
  - **Fetch price details**
- **Database: Product DB**

🔹 Order Service

- **Responsibility: Orchestrates the entire purchase flow**
- **Logic:**
  - **Receives order request**
  - **Calls Product Service → reserve/decrement stock**

- Calls Payment Service → process payment
- Updates order status

- Database: Order DB

🔹 Payment Service

- Responsibility: Simulates a 3rd-party payment gateway
- Logic:
  - Processes payment
  - Stores payment result
- Data Stored:
  - Payment status (Success / Fail)
  - Amount
  - Transaction ID
- Database: Payment DB

Infrastructure & Communication Components

🔹 Service Registry

- Implements Service Discovery
- Services register themselves
- Enables services to find & communicate dynamically

🔹 Config Server

- Centralized configuration management
- Fetches configs from GitHub
- Avoids duplication of configs across services

🔹 API Gateway

- Single entry point for clients
- Responsibilities:
  - Routes requests to correct service
  - Performs authentication & authorization
  - Acts as a security shield

Security & Authentication (Security First)

🔐 Okta

- External Identity Provider (IdP)
- Handles user authentication

🔐 JWT (JSON Web Token)

- Passed from client → gateway → internal services

- Contains user identity & roles

🔐 Role-Based Access Control (RBAC)

- Implemented using Spring Security
- Example:
  - ADMIN → Add products
  - CUSTOMER → Place orders

Observability & Tracing

🔍 Zipkin + Sleuth

- Implements Distributed Tracing
- Tracks a request across services
- Helps identify:
  - Performance bottlenecks
  - Failures
- Example trace:

Gateway → Order Service → Product Service → Payment Service

“Place Order” Workflow (End-to-End Flow)

1. Client sends request to API Gateway with JWT
2. Gateway validates JWT using Okta
3. Request routed to Order Service
4. Order Service → calls Product Service to decrement stock
5. Order Service → calls Payment Service to process payment
6. Payment Service returns status
7. Order Service:
   - Updates order status
   - Sends final response to client

⭐ Key Architectural Highlights

- Microservices with independent databases
- Secure APIs with JWT + Okta
- Centralized config management
- Scalable & observable system
- Real-world production-ready design