Overview

Design E-Commerce website to manage Product catalogue, Inventory, Customer, Order and Fulfilment. System should be secure, highly scalable and available to support high traffic.

Capacity Estimation

Assumption

1. 20M daily active users (visits)
2. 2:1 Ratio (50% of visitors buy from platform), thus 10M Sales per day
3. Avg. 1KB data generated per sales = 1KB * 10M = 10GB per day
4. 1M Products, Avg. 1MB metadata per product = 1TB

Estimation

Type	Estimation
Daily active user	20M
Request per sec	~250 requests per sec
Storage per day	1GB
Storage per year	1TB + 3.65TB = ~5TB
Bandwidth	Avg. Per request 500KB 250 * 500KB = 125MB / sec

Design Approach

Microservices architecture is going to be used which helps with horizontal scaling and decouple services. Each service will have ownership of its own data model and infra can be decided independently. Further system will aggregate individual service data in silos into Data Warehouse or Data Lake for advance use cases like Analytics, AI and Recommendation to improve CX.

Core Services

- User Service
- Product Catalogue Service
- Search Service
- Inventory Service
- Order Service
- Fulfilment Service
- Notification Service
- Analytics Service
- AI Services for GenAI, Recommendations

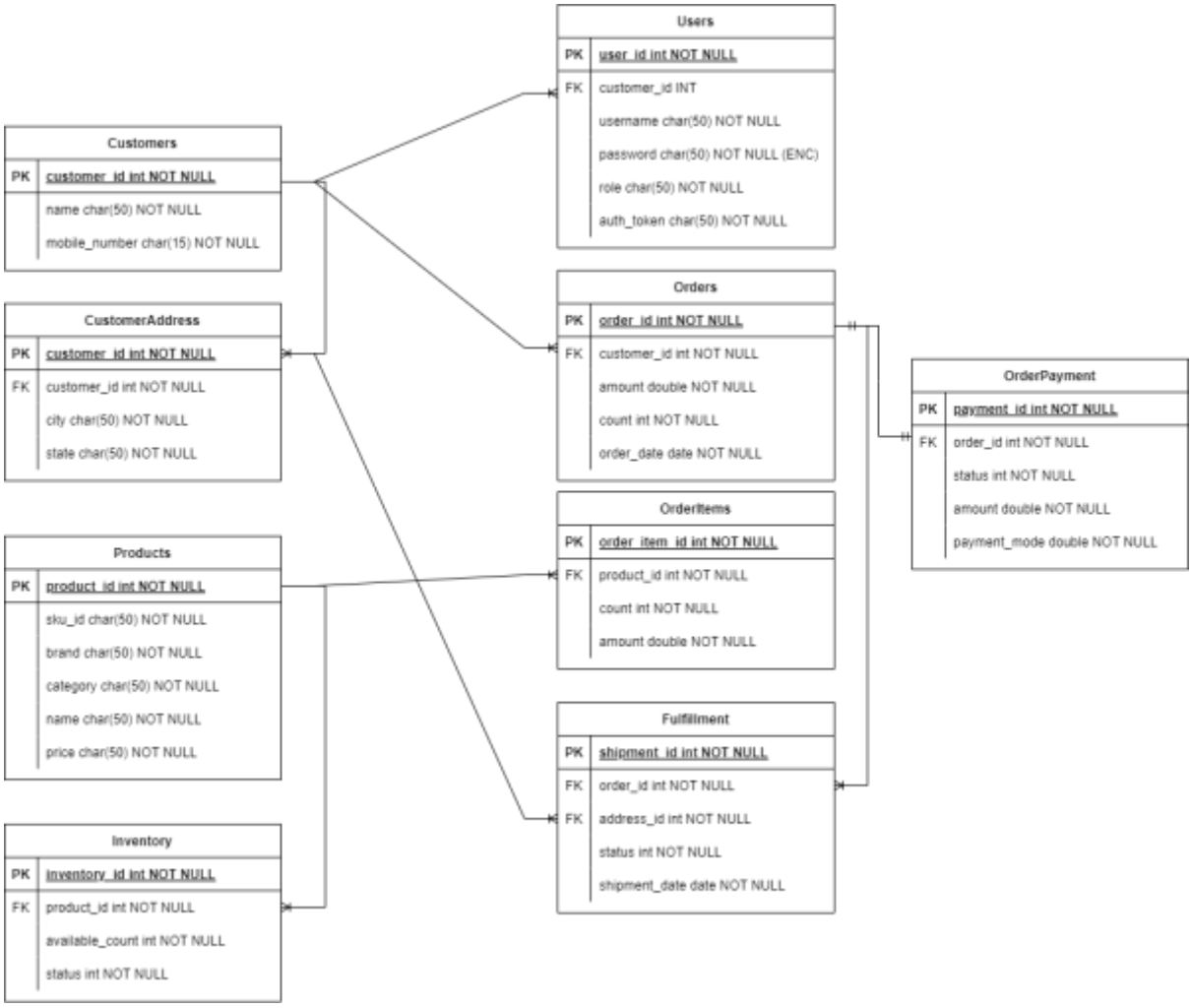
Data Storage Approach

Catalogue (Product):
Flexible schema is required considering variable Product attributes, provision for horizontal scaling to support high traffic and availability. Document Database like **Elastic Search** can be used for storing Product metadata with flexible schema and **Full-text Search** ability.

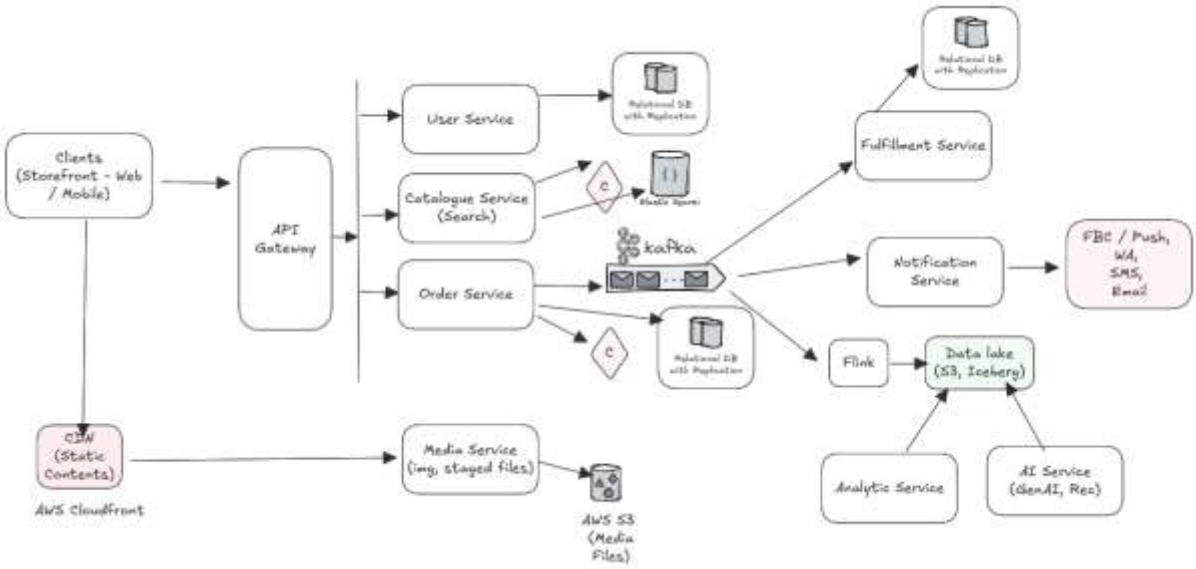
Shopping Cart or Wish list:
This may not require relational model or ACID guarantee; also storage will grow fast as random items may be added and removed. NoSQL storage (like **Cassandra, Scylladb**) with support for high scale and availability can be used.

Order:
This requires ACID guarantees and relational model. We can use any RDBMS database (like **MySQL, Postgresql**).

Data Model Design



High Level Design

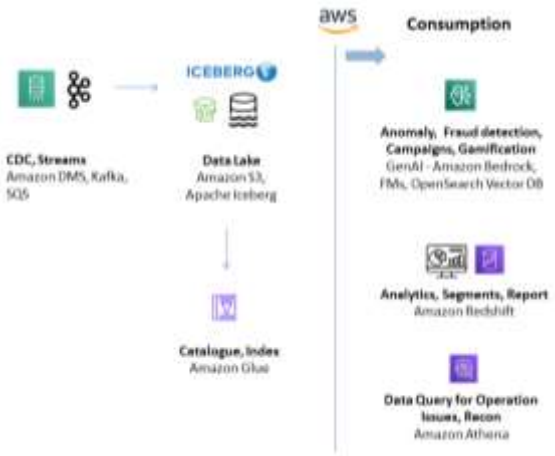


Overall High-level Design

Load Balancer with multi-node setup (Multi-AZ) for Service, Database, Cache, MQs etc. is configured to avoid single-point of failure, backup / disaster recovery and increase availability.

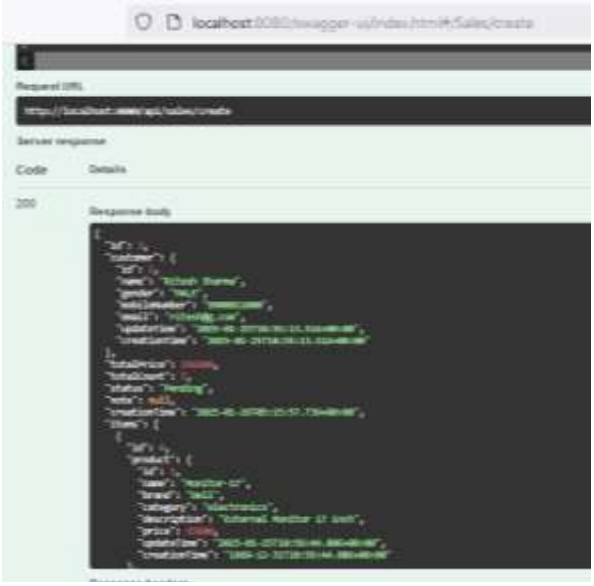
Logging and Observability
Distributed tracing and logging using tools like **AWS Cloudwatch, Prometheous** to monitor and troubleshoot distributed components.

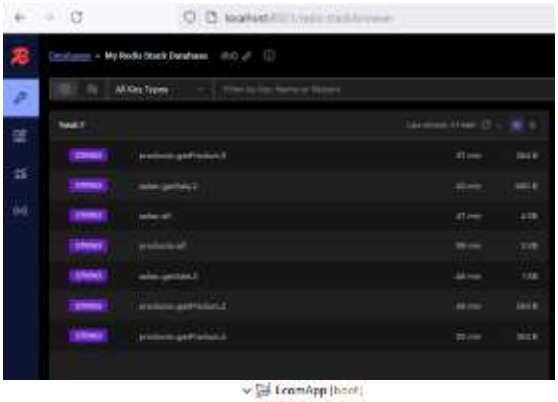
AI & Analytics Service

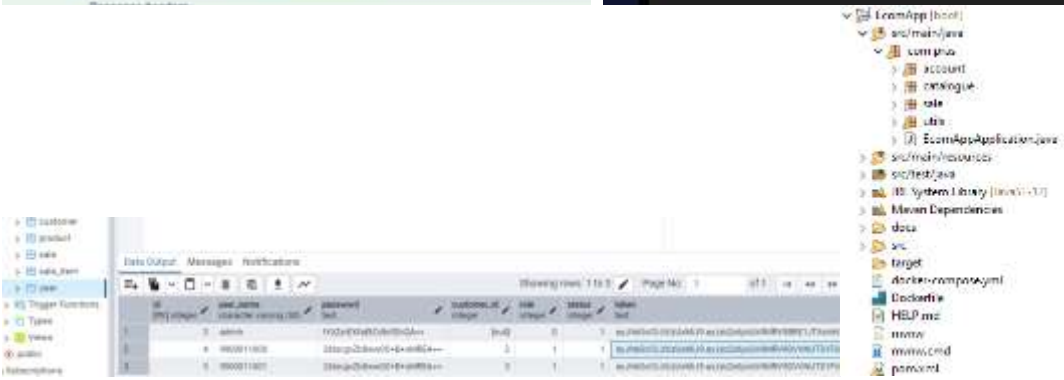


- Data lake (**S3 or Iceberge**) to store data from silos (individual Services) and create a single reference data store for advance use cases.
- **AWS Glue** would help to create Data Catalogue to explore Data lake
- Visualization / Analytics use cases using tools like **AWS Qwicksight, Grafana**
- **AWS Bedrock** and Knowledge base (**OpenSearch** vector store) would help to include GenAI models, RAG for various recommendation use cases

Application Prototype







Features

1. API to manage Registration, Customer, Product Catalogue and Order / Sales
2. Cache for Products and Sales record
3. Security:
    - Role based access (Admin and Customer)
    - JWT Token based Login
    - Password encryption
4. Exception Handling
5. Swagger Doc and API sandbox
6. Docker compose

Dependency

- Spring Boot 3.4.0, JDK 17, Maven
- Spring Security 6.4.2
- Database: MySQL 8, PostgreSQL 17
- Cache: ConcurrentMap, Redis (7.4.1)
- Modelmapper 3.2, JJWT 0.12.6
- Spring Doc (OpenAPI, Swagger) 2.7.0

Setup using Docker

Install Docker Desktop (v 4.34.2). Open CMD, Go to project root and execute command (`> docker-compose up`)

Execution Flow

Registration Process:

1. Create Admin Account (Username: admin / Password: admin) (predefined username and password)
    - URI: /api/open/registration/admin (Open End Point)
2. Register Customer and create login account (Mobile Number becomes Username, Password: <Min. 4 chars>)
    - URI: /api/open/registration/customer (Open End Point)
    - Request Body: { "password": "user123", "customer": { "name": "Lalit Kumar", "gender": "MALE", "mobileNumber": "9900011001", "email": "lalit.k@g.com" } }
3. Note JWT Token from Registration response and use this token for further API calls

Admin Account:


```
{
  "id": 3,
  "userName": "admin",
  "password": "admin",
  "token":
"eyJhbGciOiJIUzUxMiJ9.eyJyb2xlIjoiUk9MRV9BRE1JTiIsInN1YiI6ImFkbWluIiwiaWF0IjoxNzM3ODMxMjA3LCJleH
AiOjE3MzkzMDI0MzZ9.UySmRsUxxzmLAXvhbukazo_7GKEUpN7iYSXJ7c5PF4Unjsi9JvPO1Plo_xkrNABRCZ6zYA4a
E1mZXxA6z5PYcA",
  "role": "ROLE_ADMIN",
  "customer": null,
  "status": "ACTIVE",
  "updateTime": "2025-01-25T18:53:27.251+00:00",
  "creationTime": "2025-01-25T18:53:27.251+00:00"
}
```


Customer Account:


```
{
  "id": 5,
  "userName": "9900011001",
  "password": "user123",
  "token":
"eyJhbGciOiJIUzUxMiJ9.eyJyb2xlIjoiUk9MRV9DVVNUT01FUiIsInN1YiI6Ijk5MDAwMTEwMDEiLCJpYXQiOjE3Mzc4
Njc0MTgsImV4cCI6MTczOTMzODY0N30.6K9ev1CVVRzK1iOfj8aGUj_xLc1Lkfm2BSSVF_3_gZ0ANB9ZQcdeLEu7W
wdS2kE1MsKmuuLSy5UUNsSOLcSfqQ",
  "role": "ROLE_CUSTOMER",
  "customer": {
    "id": 3,
    "name": "Lalit Kumar",
    "gender": "MALE",
    "mobileNumber": "9900011001",
    "email": "lalit.k@g.com",
    "updateTime": "2025-01-26T04:56:57.967+00:00",
    "creationTime": "2025-01-26T04:56:57.967+00:00"
  },
  "status": "ACTIVE",
  "updateTime": "2025-01-26T04:56:58.941+00:00",
  "creationTime": "2025-01-26T04:56:58.941+00:00"
}
```

Product Catalogue Management:

1. Login (JWT Token) as Admin to Add/Update Products
2. Customers can view product list
3. Cache is enabled for Product list. New (or updated) Product entries are added to Cache

Sales / Order Management:

1. Login as Customer to create Sales / Order entry
    - Request Body: `{ "items": [ { "product": { "id": 1 }, "count": 1 }, { "product": { "id": 3 }, "count": 4 } ] }`
2. Login as Admin to update / change Sales / Order status
3. Cache is enabled for Sales list. New (or updated) Sales entries are added to Cache