

1. System architecture

For start of project I choose monolithic architecture (Modular-monolithic) because for that is can build fast that microservices and less complex. Also using modular monolithic it can be convert as microservices when it needed scale. For example customers growing or need multi region expansion.

Client(Browser or Mobile) ➡ HTTP Requests ➡ Backend

Then inside backend,

Security Check (Authentication, Authorization),

Controllers (thin controllers)

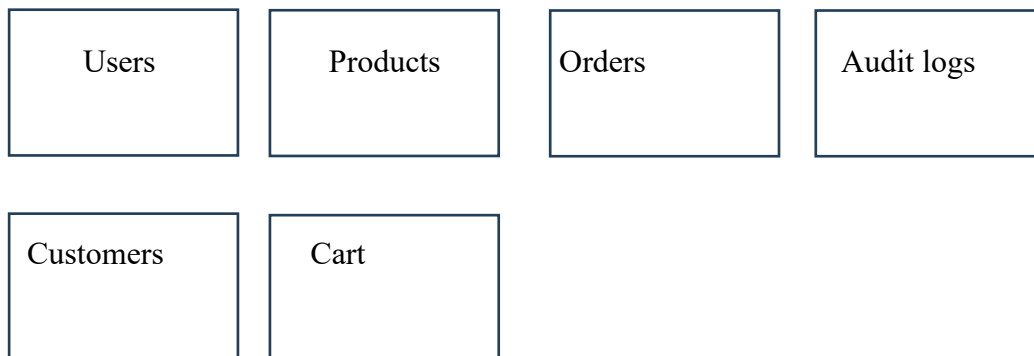
Services (where business logic)

Repository (Database access layer)

Then after processing it return response to client.

2. Database Design

I choose no-sql db for this like MongoDB Atlas. Because using this I can reduce my manual work for scaling, backup and security when I need simple setup first. Also this has flexible schema that can use for when first phase of project. But we move to payment section it need well defined and solid variable because consider security.



Like these we can configure basic database collection first when it required field and optional fields,

User

- First_name
- Last_name
- Email
- Mobile
- isActive
- createdAt
- username

Products

- name
- category
- uom
- price
- quantity
- status
- last_stock_date

orders

- name
- customer – foreignkey
- date
- status

audit logs

- entity
- action
- prevValue
- newValue
- ip
- performedBy foreignkey
- createdAt

Cart must be collection with temporary documents based userId.

3. I prefer separate backend and frontend because it is easily scalable if we move microservices. And other optimization and code clarity also increases.

4. Tech Stack

Frontend – Next.js

Backend – Nest.js

DB – MongoDB

Auth – Auth0 (if need multifactor auth or SSO)

5. Cloud Infrastructure

I Choose AWS and EC2 for this and also S3 for images, if we need we can use CloudFront also performance.

6. Scalability

Horizontal scaling with AWS and MongoDB support. And also need use load balancer.