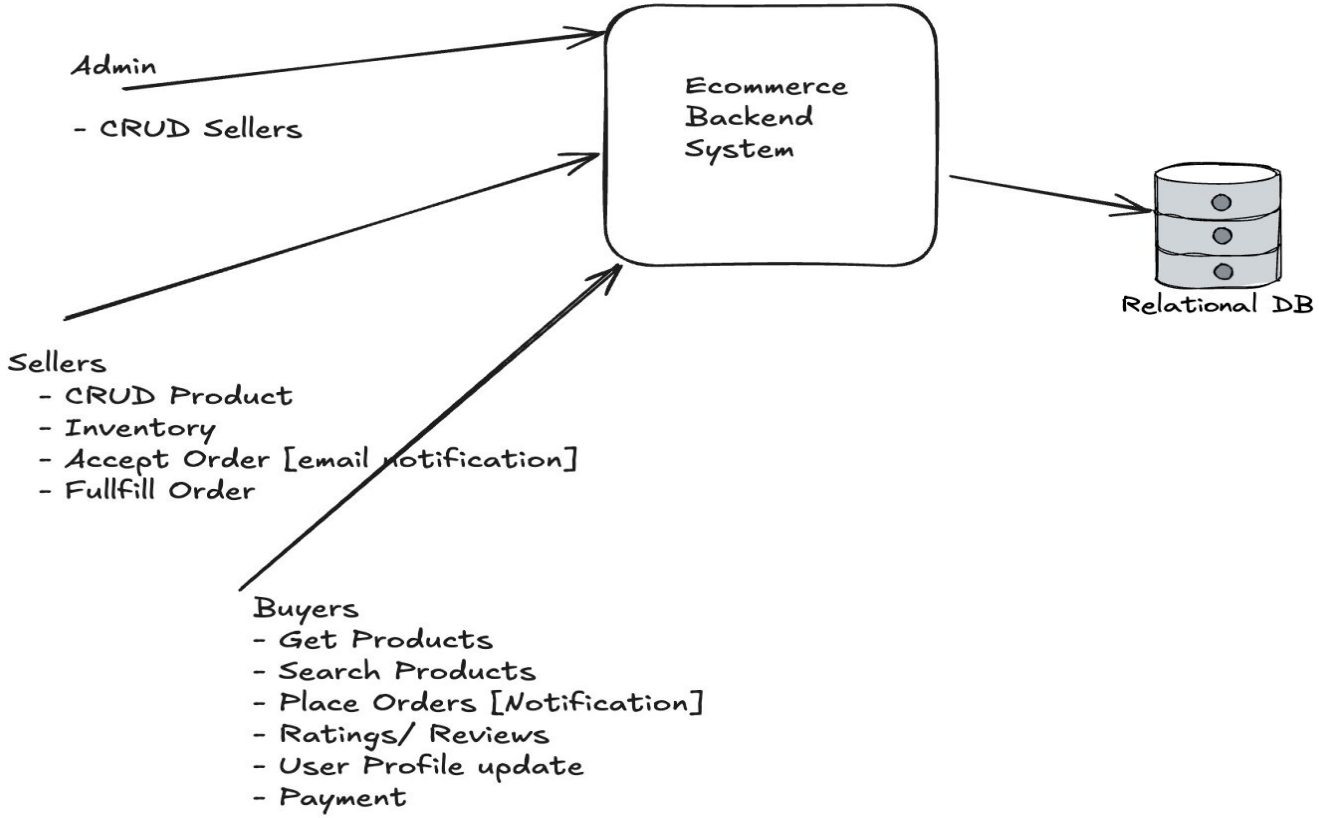
# Java Backend Development Live-85

lecture-15

# Agenda

- Minor Project-1 Completion
  - Pagination
  - CSV upload
  - Image upload
  - Scheduled Job
- Run application jar
- -Xms & -Xmx Flags
- Swagger

# Ecommerce Backend System



# Functional Requirements

Admin

- CRUD Sellers

Sellers

- CRUD Products
- Accept Order
- Fulfil Order

Buyers

- Search Products
- Add product to Order
- Place Order
- Order History
- Profiles
- Payments

# Tables

User (id,name, email, password, role[ADMIN, SELLER, CUSTOMER], created_at, updated_at)

Company (id, name, number, is_active, user_id, created_at, updated_at)

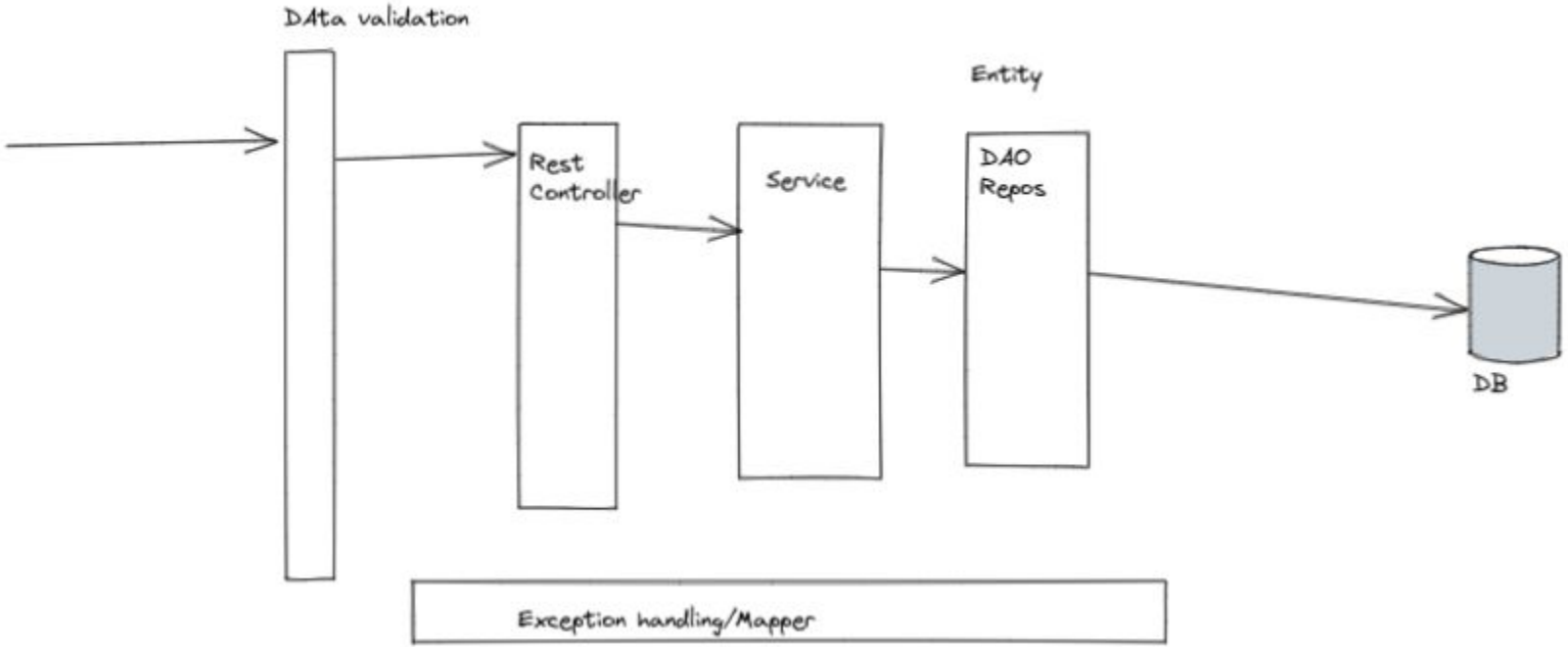
Product(id, name, description, price, stock, company_id, is_active, category_id, created_at, updated_at)

Order (id, user_id, total_amount, status(DRAFT,PLACED,ACCEPTED,SHIPPED,DELIVERED), created_at, updated_at)

Order_Item (id, order_id, product_id, quantity, price)

Category(id, name, description)

# API Request flow



# Admin APIs

**Add a Seller**

`POST /api/admin/sellers`

Request body: Seller details (e.g., name, email, phone)

Response: Seller ID or success message

**View All Sellers**

`GET /api/admin/sellers`

Response: List of all sellers

**Delete a Seller**

`DELETE /api/admin/sellers/{sellerId}`

Response: Success or error message

# Seller APIs

**Add a Product**

`POST /api/seller/products`

Request body: Product details (e.g., name, description, price, stock, category)

Response: Product ID or success message

**View All Products by Seller**

`GET /api/seller/products`

Response: List of all products added by the seller

**Update Product Details**

`PUT /api/seller/products/{productId}`

Request body: Fields to update (e.g., price, stock)

Response: Success or error message

**Delete a Product**

`DELETE /api/seller/products/{productId}`

Response: Success or error message

# Customer APIs

**Browse Products**

`GET /api/products?keyword=laptop`

Response: List of available products

**View Product Details**

`GET /api/products/{productId}`

Response: Product details

**Add Order Item**

`POST /api/customers/order-item`

Response: OrderId

**View Order History**

`GET /api/customers/orders`

Response: List of previous orders

# Pagination

Pagination is often helpful when we have a large dataset and we want to present it to the user in smaller chunks.

**PagingAndSortingRepository:** By extending PagingAndSortingRepository repository interface can get findAll(Pageable pageable) and findAll(Sort sort) methods for paging and sorting.

```
Pageable pageable = Pageable.ofSize(pageSize).withPage(pageNo);
```

# CSV upload and Image upload

Comma Separated Values (CSV) is a popular data exchange format frequently used for importing and exporting data between servers and applications.

**MultipartFile**: The MultipartFile interface is a special data structure Spring Boot provides to represent an uploaded file in a multipart request.

```
public ResponseEntity<String> uploadFile(@RequestParam("file") MultipartFile file)
```

# Scheduled Job

@Scheduled annotation can be used to configure and schedule tasks.

**@Scheduled(fixedDelay = 1000)**: the duration between the end of the last execution and the start of the next execution is fixed. The task always waits until the previous one is finished.

**@Scheduled(cron = "0 15 10 15 * ?")**: This is scheduling a task to be executed at 10:15 AM on the 15th day of every month.

# Run Application JAR

**create jar**:  mvn clean install

**run jar**: java -jar application_name.jar

**Override property**: java -Dserver.port=8081  -jar application_name.jar

**Change profile/environment**: -Dspring.profiles.active=test

java -Dserver.port=8081 -Dspring.profiles.active=test  -jar application_name.jar

**Pass external application.properties:**

-Dspring.config.location="file:///home/shashi/Desktop/external_application.properties"

-Dspring.config.location="/home/shashi/Desktop/external_application.properties"

# -Xms & -Xmx Flags

The flag **Xmx** specifies the maximum memory allocation pool for a Java Virtual Machine (JVM), while **Xms** specifies the initial memory allocation pool.

For example, starting a JVM like below will start it with 256 MB of memory and will allow the process to use up to 2048 MB of memory

java -Xms256m -Xmx2048m

# Swagger

- Swagger is an open-source framework for API documentation and testing.
- It provides an interactive UI to explore and test API endpoints.
- Uses **OpenAPI Specification (OAS)** to define APIs in a standardized format.
- **Swagger Dependency (Spring Boot 3+ with OpenAPI 3.0)**

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.3.0</version>
</dependency>
```

By default, Swagger UI will be available at:
`http://localhost:8080/swagger-ui.html`