ZSJF001 USER MANAGEMENT

This module — **User Management** — is the **foundation** of your eCommerce platform. It handles:

- Creating new users (registration or admin creation)
- Updating user details
- Deleting users
- Viewing all users
- Viewing a specific user by ID

Every intern working on this task will understand how the **Spring Boot backend** interacts with the **MySQL database** (via stored procedures) and how the **Angular frontend** communicates with APIs.

---

1. Database Layer

Database `zeptoware_ecommerce`

Table: `tbl_users`

This table stores basic user information.

Column	Type	Description
`user_id`	`INT AUTO_INCREMENT PRIMARY KEY`	Unique user ID
`full_name`	`VARCHAR(100)`	User's full name
`email`	`VARCHAR(100)`	User's email (must be unique)
`password`	`VARCHAR(255)`	Encrypted password
`role_id`	`INT`	References the role from `tbl_roles` master
`phone`	`VARCHAR(20)`	Contact number
`is_active`	`TINYINT DEFAULT 1`	1 = Active, 0 = Inactive
`created_at`	`DATETIME DEFAULT NOW()`	Record creation time

Key Learning:

This design separates **user roles** into another table (`tbl_roles`), ensuring flexibility.
Example: Admin, Customer, Vendor, etc.

---

2. Stored Procedures (CRUD)

Using **stored procedures** standardizes database operations and ensures **all logic remains inside MySQL**.
That means — Spring Boot only calls procedures; no direct SQL inside Java.

a) **`sp_tbl_users_insert`** Adds a new user.

b) **`sp_tbl_users_update`**

Modifies existing user data.

c) **`sp_tbl_users_delete`** Removes a user by ID.

d) **`sp_tbl_users_get_all`** **Lists all users.**

e) **`sp_tbl_users_get_by_id`**

Key Learning:
Using consistent naming (`sp_tbl_<table>_<action>`) helps organize and automate procedure calls later.

---

3. Spring Boot Repository

This Java class directly communicates with the database using `JdbcTemplate`. It calls the stored procedures created above.

```
@Repository
public class UserManagementRepository
```

Key Learning:
Interns see **how stored procedures replace inline SQL queries**, ensuring cleaner Java code and database-driven logic.

---

4. Spring Boot Controller

This is the **REST API layer** that receives HTTP requests (from Angular) and calls the repository methods.

```
@RestController
@RequestMapping("/api/users") public
class UserManagementController
```

Key Learning:
Every endpoint corresponds to a specific stored procedure call, making debugging and mapping very simple.

---

5. Angular Integration

Your Angular frontend will use `HttpClient` to call the REST APIs.

Key Learning:
Each Angular method directly maps to one backend endpoint — simple and consistent.

---

6. Expected JSON Input & Output

Example Request (Insert)

```
{
  "full_name": "John Doe",
  "email": "john@example.com",
  "password": "hashedPassword123",
  "role_id": 1,
  "phone": "9876543210",
  "is_active": 1,
  "created_at": "2025-11-12 10:00:00"
}
```

Example Response

```
{
  "status": "success",
  "message": "User added successfully!"
}
```

Example Request (Fetch User)

```
GET /api/users/1
```

Example Response

```
{
  "user_id": 1,
  "full_name": "John Doe",
  "email": "john@example.com",
  "role_id": 1,
  "phone": "9876543210",
  "is_active": 1,
  "created_at": "2025-11-12T10:00:00"
}
```

---

7. Intern Deliverables

Each intern should submit:

1. SQL file with `CREATE TABLE` + all CRUD stored procedures.
2. Java files (Repository + Controller).
3. Angular service and component integration.
4. Postman collection (to test all 5 APIs).
5. Short README explaining module flow and test results.
