# Domain-Driven Design (DDD) Document

Project: User Management System

Tech Stack: Angular (Frontend), Spring Boot Microservices (Backend), PostgreSQL/MySQL, Spring Cloud, Eureka, API Gateway

## 1. Bounded Contexts

|  |  |
| --- | --- |
| Bounded Context | Description |
| User Management | Handles user CRUD operations, profile, roles, and permissions |
| Authentication Service | Login, token management, password handling |
| Role & Permission | Manages RBAC (Role Based Access Control) |
| Notification (optional) | Sends emails/SMS for signup, password resets, etc. |

## 2. Aggregate: User

Root Entity: User

- ID (UUID)  
- First Name  
- Last Name  
- Email (Unique)  
- Phone Number  
- Username (Unique)  
- Password (Encrypted)  
- IsActive  
- CreatedAt  
- UpdatedAt

Value Objects:  
- Email  
- PhoneNumber  
- Address (Optional, for profile enrichment)

Related Entities:  
- Role  
- Permission

## 3. Domain Layer (Spring Boot)

Entities (User):  
@Entity  
public class User {  
 @Id  
 private UUID id;  
 ...  
}

Value Objects (Email, PhoneNumber):  
@Embeddable  
public class Email {  
 private String value;  
}

Repository:  
public interface UserRepository extends JpaRepository<User, UUID> {  
 Optional<User> findByEmail(Email email);  
}

Domain Services:  
@Service  
public class UserDomainService {  
 public boolean isValidUser(User user) {}  
}

## 4. Application Layer

Services:  
public interface UserService {  
 UserDTO createUser(...);  
 ...  
}

DTOs:  
public class UserDTO {  
 private UUID id;  
 ...  
}

## 5. API Gateway + Security

- Spring Cloud Gateway routes requests to microservices.  
- OAuth2/JWT Authentication via Authentication Service.  
- Angular Frontend talks to Gateway.

## 6. Angular Frontend

Modules:  
- UserModule  
- AuthModule  
- RoleModule

Components:  
- UserListComponent  
- UserFormComponent  
- LoginComponent  
- ProfileComponent

Services:  
- UserService  
- AuthService

Routes Example:  
const routes: Routes = [  
 { path: 'users', component: UserListComponent },  
];

## 7. Inter-Service Communication

- REST via Feign Clients or OpenFeign  
- Authentication via JWT Tokens  
- Events via Spring Cloud Stream (optional)

## 8. Testing

- Unit Tests for Domain Logic  
- Integration Tests using TestContainers  
- End-to-End (E2E) tests using Cypress (Angular)

## 9. Documentation & OpenAPI

- Swagger for REST documentation  
- Postman collections  
- README per microservice