Low Level Design Document

This Low Level Design (LLD) document outlines the core components, data structures, and logic for **PropEase - Real Estate Listings & Booking API**. The project is a Spring Boot backend for managing property listings, bookings, and user profiles, with RESTful APIs, JWT authentication, MySQL storage, and Docker deployment.

---

1. System Components

Component	Description	Key Responsibilities
API Layer	REST Controllers	Expose endpoints, request validation
Service Layer	Business Logic	Process requests, enforce rules
Repository Layer	Data Access (Spring Data JPA)	CRUD operations on MySQL
Security Module	JWT Auth, User Roles	Authenticate, authorize, token handling
Exception Handler	Global Error Handling	Map exceptions to HTTP responses

---

2. Class/Interface Overview

Class/Interface	Description	Key Methods/Attributes
`UserController`	User endpoints	`register()` , `login()` , `getProfile()`
`PropertyController`	Property endpoints	`create()` , `list()` , `getById()`
`BookingController`	Booking endpoints	`book()` , `listByUser()` , `cancel()`
`UserService`	User logic	`registerUser()` , `authenticate()`
`PropertyService`	Property logic	`addProperty()` , `getProperties()`
`BookingService`	Booking logic	`createBooking()` , `cancelBooking()`
`UserRepository`	User DB access	`findByEmail()` , `save()`
`PropertyRepository`	Property DB access	`findAll()` , `findById()` , `save()`
`BookingRepository`	Booking DB access	`findByUser()` , `save()` , `delete()`
`JwtUtil`	JWT utilities	`generateToken()` , `validateToken()`
`GlobalExceptionHandler`	Error handling	`handleException()`

---

3. Data Structure Overview

Entity	Fields
User	`id` , `name` , `email` , `passwordHash` , `role`

Property	`id`, `title`, `description`, `address`, `price`, `ownerId`, `status`
Booking	`id`, `userId`, `propertyId`, `startDate`, `endDate`, `status`

Relationships:

- User (1) — (M) Property
- User (1) — (M) Booking
- Property (1) — (M) Booking

---

4. Algorithms / Logic

Booking Creation Flow (Pseudocode):

```
if (user is authenticated && property is available) {
    if (dates are valid && not overlapping) {
        create booking;
        update property status if needed;
        return success;
    } else {
        return error (invalid dates or overlap);
    }
} else {
    return error (unauthorized or unavailable);
}
```

JWT Authentication Flow:

```
on login/register:
    generate JWT token with userId, role, expiry;
on protected endpoint:
    extract and validate JWT;
    set user context if valid;
    reject if invalid/expired;
```

---

5. Error Handling

Scenario	Handling/Response
Invalid Input/Validation Error	400 Bad Request + error message
Authentication Failure	401 Unauthorized
Authorization Failure	403 Forbidden
Resource Not Found	404 Not Found
Booking Conflict/Overlap	409 Conflict + descriptive message
Internal Server Error	500 Internal Server Error

End of Document