### **Jwt Auth, admin, file, maps search**

### **Next Steps (When You're Ready)**

1. ****Add User Authentication****: Implement JWT tokens for real login
2. ****Create Admin Panel****: Approve/reject spaces and users
3. ****Add File Upload****: Let users upload images for spaces
4. ****Implement Search****: Filter spaces by location, art form, etc.
5. ****Add Maps****: Integrate Leaflet maps for space locations

### **🔐 Step 1: Add JWT Dependencies to Backend**

#### **Step 1.1: Update pom.xml**

Add these dependencies to your **pom.xml** inside the **<dependencies>** section:

#### **Step 1.2: Create JWT Utility Class**

1. Create new package: **src/main/java/com/underground/c3\_backend/utils**
2. Create file: **JwtUtils.java**

#### **Step 1.3: Add JWT Properties to application.properties**

#### **Step 1.4: Create Authentication Response DTO**

1. In the **dto** package, create: **JwtResponse.java**

#### **Step 1.5: Create Login Request DTO**

1. In the **dto** package, create: **LoginRequest.java**

### **🔐 Step 2: Create Authentication Controller**

#### **Step 2.1: Update AuthController**

#### **Step 2.2: Update SecurityConfig to Enable Authentication**

#### **Step 2.4: Test the Login Endpoint**

<http://localhost:8080/api/auth/signin>

### **🚀 Step 3: Test the Real Authentication with Postman or Frontend**

#### **Option 1: Test with Frontend (Easier)**

#### **Option 2: Test with Postman (More Technical)**

### **🔒 Step 4: Create Auth Context in Frontend**

#### **Step 4.1: Create Auth Context**

1. In **src/**, create a new folder called **context**
2. Create file **src/context/AuthContext.js**:

testuser123

password123

****Step 5: Creating Protected Routes and Implementing Real Authorization****.

### **🔒 Step 5: Create Protected Routes**

#### **Step 5.1: Create Protected Route Component**

1. In **src/**, create a new folder called **components/auth**
2. Create file **src/components/auth/ProtectedRoute.js**: