Goal:

Implement **JWT authentication and role-based authorization** for an app with ADMIN, SELLER, and CUSTOMER roles.

🔷 Day 1: Fundamentals & Setup

- **Learning:**
  - JWT basics (header, payload, signature).
  - Stateless authentication vs session-based.
  - Basics of Spring Security & RBAC.
- **Tasks:**
  - Create a Spring Boot project (or use an existing one).
  - Add dependencies:
    - spring-boot-starter-security
    - spring-boot-starter-web
    - spring-boot-starter-data-jpa
    - jjwt or auth0 library
  - Create User entity with:
    - id, username, password, roles(Set<Role>)
  - Create Role entity: ADMIN, SELLER, CUSTOMER.
  - Seed DB with sample users for each role.
- **Deliverable:**
  ✅ Running app with DB setup and sample users.

🔷 Day 2: Authentication Endpoint

- **Learning:**
  - Implement UserDetailsService.
  - Password hashing with BCrypt.

- **Tasks:**
    - Build /auth/login endpoint.
    - Verify username/password, generate JWT token with **roles claim**.
- **Deliverable:**
  ✅ POST /auth/login returns a valid JWT containing user roles.

🔷 Day 3: JWT Filter & Security Config

- **Learning:**
    - JWT validation & extracting claims.
    - Custom OncePerRequestFilter in Spring Security.
- **Tasks:**
    - Create JwtFilter to:
        - Parse token from Authorization header.
        - Validate signature, expiration.
        - Set authenticated user in security context.
    - Configure Spring Security to:
        - Allow /auth/** publicly.
        - Secure all other endpoints.
- **Deliverable:**
  All endpoints require a token, except /auth/login.

🔷 Day 4: Role-Based Authorization

- **Learning:**
    - Use Spring annotations: @PreAuthorize, @Secured.
    - Implement endpoint-level restrictions.
- **Tasks:**
    - Create sample endpoints:
        - /admin/** → ADMIN only

        - /seller/** → SELLER & ADMIN
        - /customer/** → CUSTOMER, SELLER, ADMIN
    - Apply method-level security with roles.
- **Deliverable:**
  Tested endpoints with proper access per role.

❖ Day 5: Refresh Tokens & Logout

- **Learning:**
    - Token expiration strategy.
    - Refresh tokens flow.
- **Tasks:**
    - Implement /auth/refresh to issue new access tokens.
    - (Optional) Logout by invalidating tokens (basic blacklist or token rotation).
- **Deliverable:**
  Working login, token refresh, and basic logout mechanism.

❖ Day 6: Testing & Documentation

- **Learning:**
    - Security best practices (secret key storage, HTTPS).
- **Tasks:**
    - Write a README.md with setup instructions.
    - Add sample requests & responses.
    - Prepare a Postman collection to test:
        - Login
        - Access restricted endpoints with each role
        - Refresh token flow
- **Deliverable:**
  Full documentation & Postman tests ready.