**Backend Design**

**Features:**

Login, Signup, Logout for Student and Instructor

Instructor can do CRUD on Courses and fetch his own courses

Student can see listing of courses and fetch by category

**Coding Practices:**

**Separation of Concerns (SoC)**

* Controller handles HTTP, Service contains business logic, Repository handles DB access, Config contains only configurations like Password Config to encrypt password and Swagger config for swager

**DTO Pattern**

* Using CreateCourseRequest, UpdateCourseRequest, CourseResponse to avoid exposing entities directly and similarly on user entity.

**Validation**

* Using @Valid and Jakarta annotations like @NotBlank, and validating the input size

**Role-Based Access Control (RBAC)**

* With @PreAuthorize("hasRole('...')"), as in my project, I’ve student and instructor

**Dependency Injection**

* Using @Autowired and @RequiredArgsConstructor.

**JWT Authentication**

* Stateless security using token-based auth

**Design Patterns Used:**

| **Pattern** | **Where Used** |
| --- | --- |
| **Singleton** | All @Component, @Service, @Repository beans in Spring are Singleton by default. Ex: JwtUtil, UserDetailsServiceImpl. |
| **Strategy** | UserDetailsServiceImpl is an implementation of the UserDetailsService strategy interface. |
| **Builder** | @Builder in User entity for object creation. |
| **Adapter** | UserDetailsServiceImpl adapts custom User entity to UserDetails. |
| **Facade** (semi) | CourseServiceImpl acts like a facade simplifying multiple DB operations. |
| **Template Method** | OncePerRequestFilter is a classic Spring pattern, and override doFilterInternal() method. |
|  |  |

**Third Party libraries:**

* Lombok for built-in getters, setters, constructors
* Swagger for documentation

**Database Performance Increase:**

Adding index on category can help filter categories by fast as it would sort it by using B- or may be B+ trees and having pointer along with sorted data

**Example:**

Index: 8, category: A

Index: 5, category: B

Index: 7, category: C

A, B, C is dummy data for categories, in reality it would be coding, maths, science and sorted by lexicographical order and index would be actual position or in programming terms I would say reference of that document.

**Improvement:**

One more improvement can be as string has more characters so comparison would take time to compare character by character, mapping the string to integer would be better for comparison as integers best for comparison in this case.

**Scalability:**

Regarding scalability, as each course has a list of lessons, so it can be turned into a separate collection as if we get many lessons that can’t be handle inside course collection? Why in MONGO DB we have 16 mb limit of a collection according to me. So restricting would be better but it will upscale when lessons increase until then as there is no NEED-TO-KNOW basis case here so fine with current schema.

Course -> List of Lessons inside it