MIU

Academic Record Management System (ARMS)

Project Defense Presentation

Olamide Akinoso, Hnin Nandar Zaw &
Abdullah Al Zayed

Purpose

The purpose of the Academic Records Management System (ARMS) is to provide an integrated platform for managing student academic information, including course enrollment, grade tracking, classroom allocation, and transcript generation

The system will serve MIU’s Registrar’s Office, faculty, and administrative staff to streamline operations and ensure accurate academic record-keeping.

SCOPE

- Register new students and store personal, academic, and enrollment data.
- Allow administrators and faculty to create and manage course offerings.
- Record student enrolment in multiple courses per semester.
- Capture grades upon course completion
- Dynamically generate official transcripts reflecting completed courses, credits, and GPA.
- Manage classroom assignments with a many-to-many relationship between students and classrooms.
- Provide secure access and role-based permissions for different system users

FEATURES OF THE SOLUTION

FEATURES OF THE MIU **ARMS**

Student Management

Add, update, view, and deactivate student records

Course Management

Create and manage courses with details like course code, title, credit hours, prerequisites.

Enrollment Management

Enroll students in courses, manage drop/add periods.

Grade Management

Record grades for students in each course

Transcript Generation

Automatically compile grades from completed courses into a transcript view/report.

Classroom Management

Assign students to classrooms; maintain historical records of class memberships.

REPORTING

Generate academic performance reports, GPA reports, and enrollment summaries.

User Classes and Characteristics

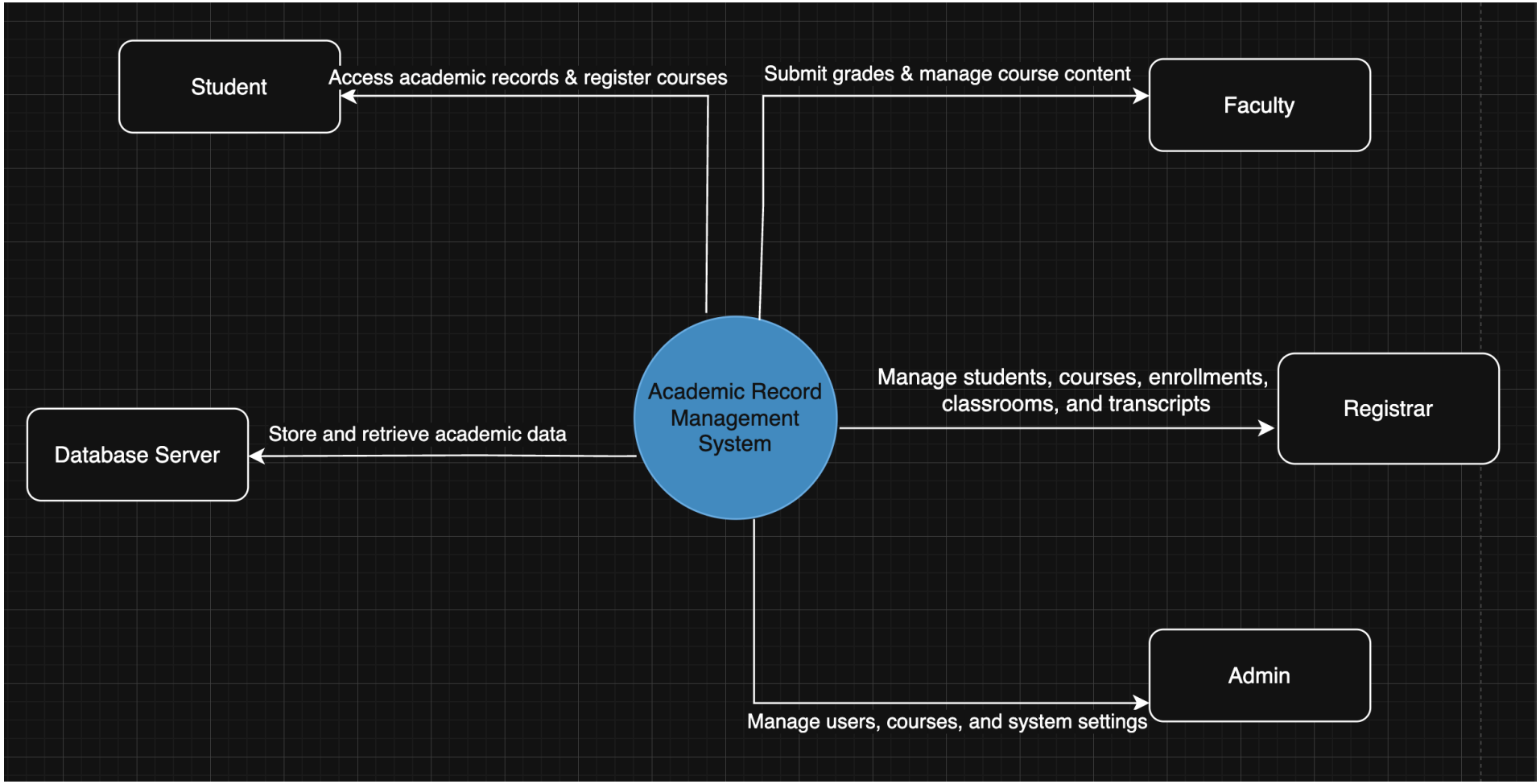
1. Registrar Staff
Full CRUD access to all student, course, enrollment, and transcript data.

2. Faculty
Can view and manage students in their courses, enter grades

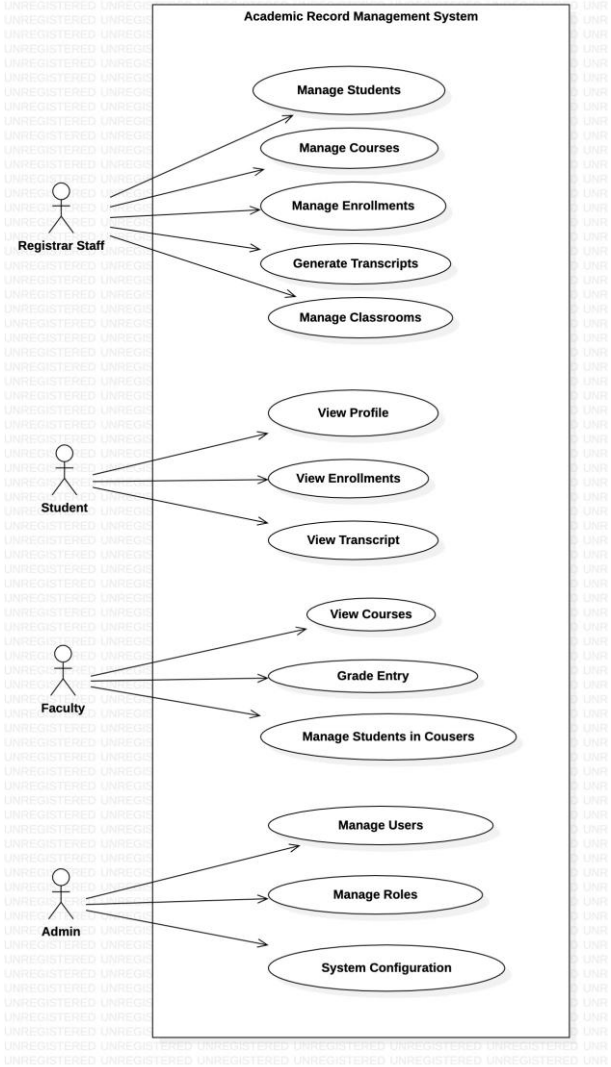
3. Student
View their personal profile, enrolled courses, and transcripts

4. Administrators
Manage system configurations, roles, and permissions

ARMS Context Diagram



Use Case Diagram



User Stories

1. Student Onboarding & Profile Management

- As an **Admin**, I want to **register a new student** so that **they can access the system**.
- As a **Student**, I want to **update my personal details** so that **my records remain accurate**.

2. Course & Classroom Management

- As an **Admin**, I want to **create courses** so that **students can enroll**.
- As an **Admin**, I want to **assign instructors to classrooms** so that **lectures are organized**.
- As an **Admin**, I want to **mark a classroom as active/inactive** so that **only current classrooms are shown**.

3. Enrollment & Grading

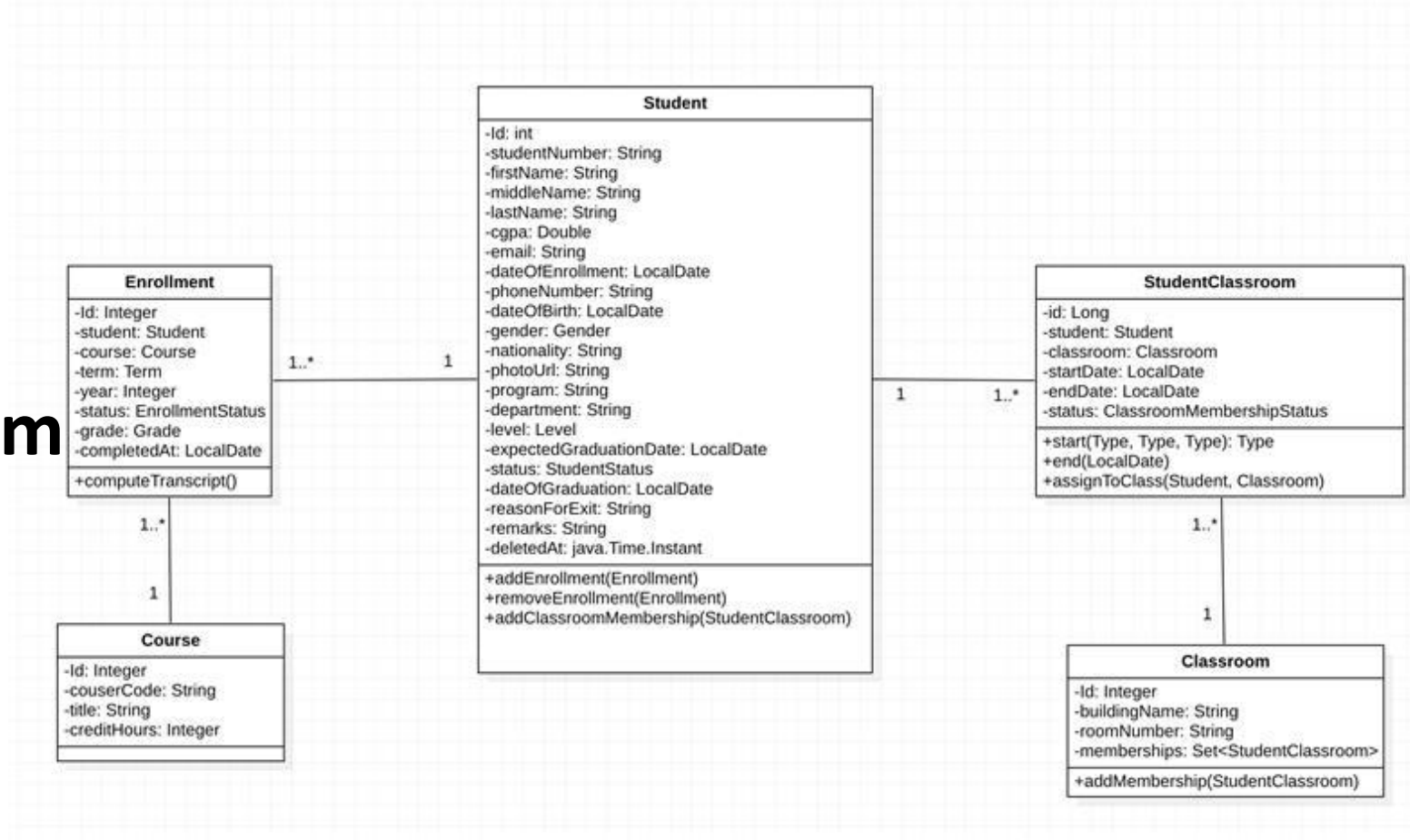
- As an **Admin**, I want to **enroll a student in a course** so that **they can attend classes**.
- As an **Instructor**, I want to **record grades for completed courses** so that **students' performance is tracked**.

4. Transcript & Academic History

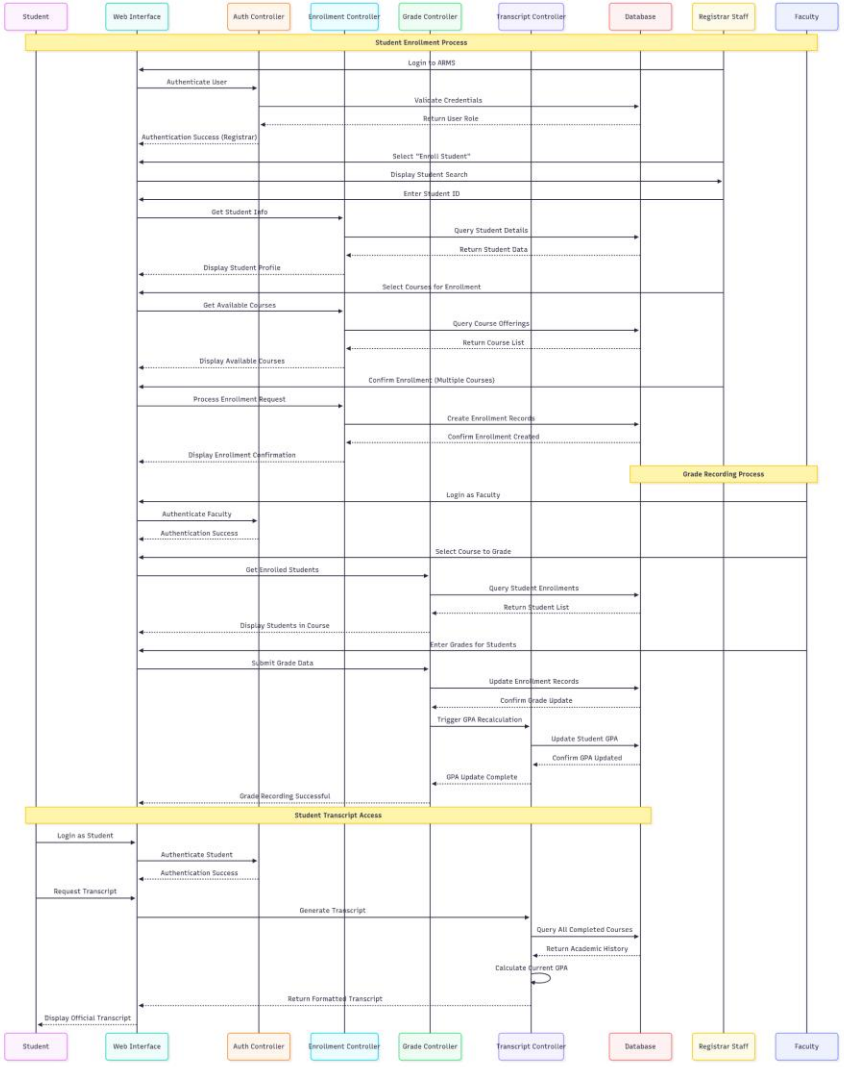
- As a **Student**, I want to **view my transcript** so that **I can see my performance history**.
- As an **Admin**, I want to **generate official transcripts** so that **students can use them for applications**.
- **5. Administration & Reporting**
- As an **Admin**, I want to **view enrollment statistics** so that **I can monitor trends**.
- As an **Admin**, I want to **export reports** so that **they can be shared with university management**.

User Story Map

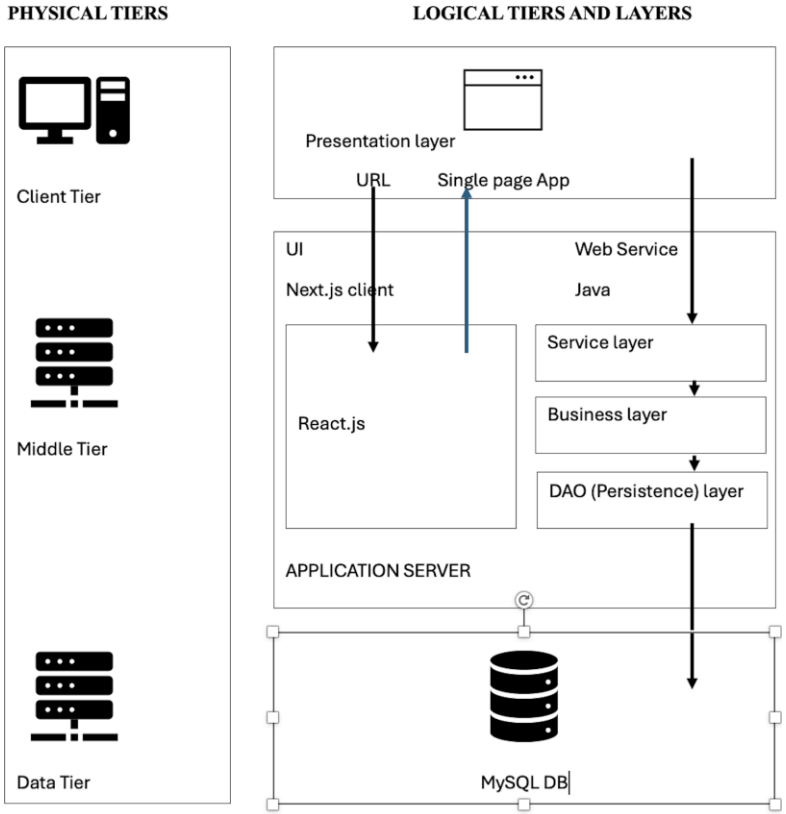
UML Class Diagram



UML Sequence Diagram



System Architecture



TECHNOLOGY

Backend Framework & Core Technologies

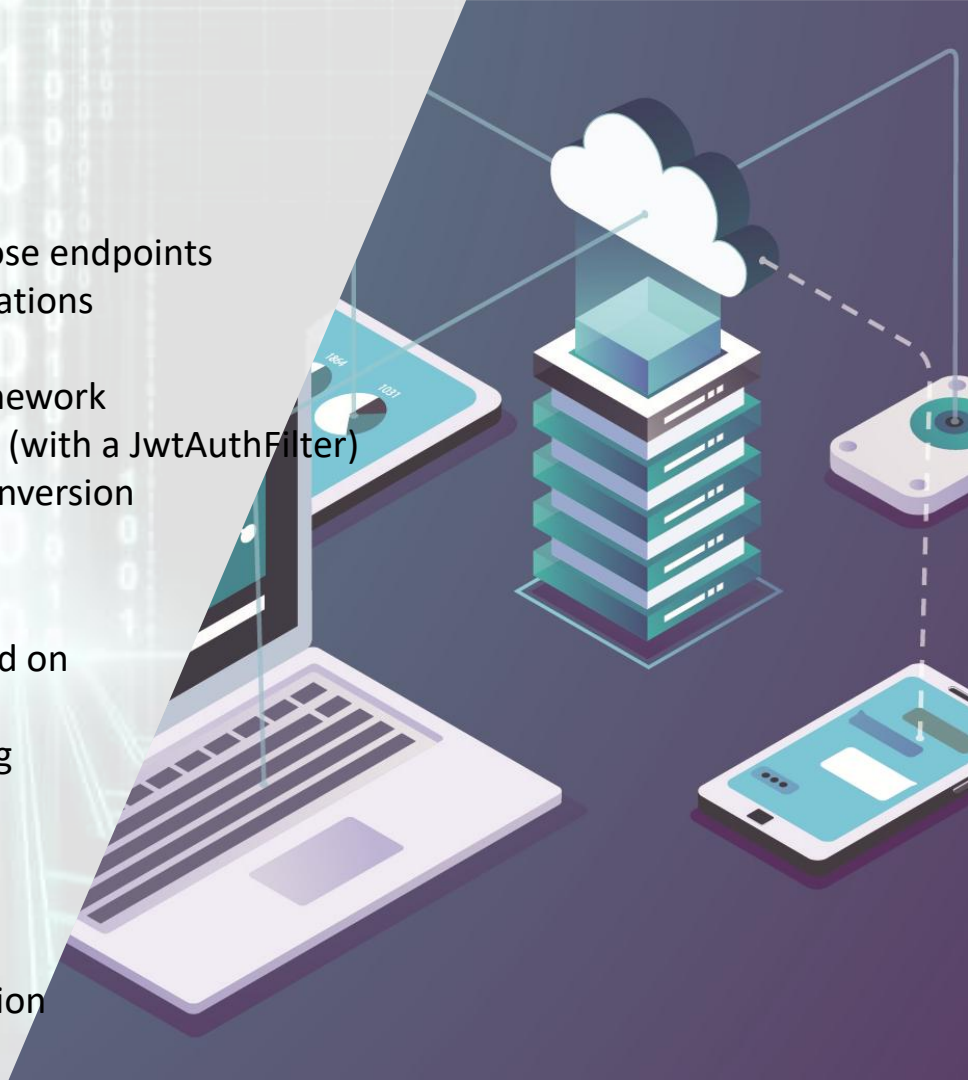
- **Java** 21
- **Spring Boot** – primary application framework
- **Spring Web (Spring MVC)** – For RESTful APIs and expose endpoints
- **Spring Data JPA** – ORM abstraction for database operations
- **Hibernate** – Underlying JPA provider
- **Spring Security** – Authentication & authorization framework
- **JWT (JSON Web Token)** – Token-based authentication (with a JwtAuthFilter)
- **MapStruct** – Java bean mapping for DTO ↔ entity conversion
- **Lombok** – to reduce boilerplate code

Database & Persistence

- **Relational Database** (likely MySQL or PostgreSQL based on typical Spring Boot setups)
- **JPA/Hibernate Entity Mappings** – for domain modeling
- **Schema management** via JPA auto-DDL or tools like Flyway/Liquibase (if configured)

Testing

- **JUnit 5** – unit/integration testing framework.
- **Spring Boot Test** – for application context and integration testing



TECHNOLOGY

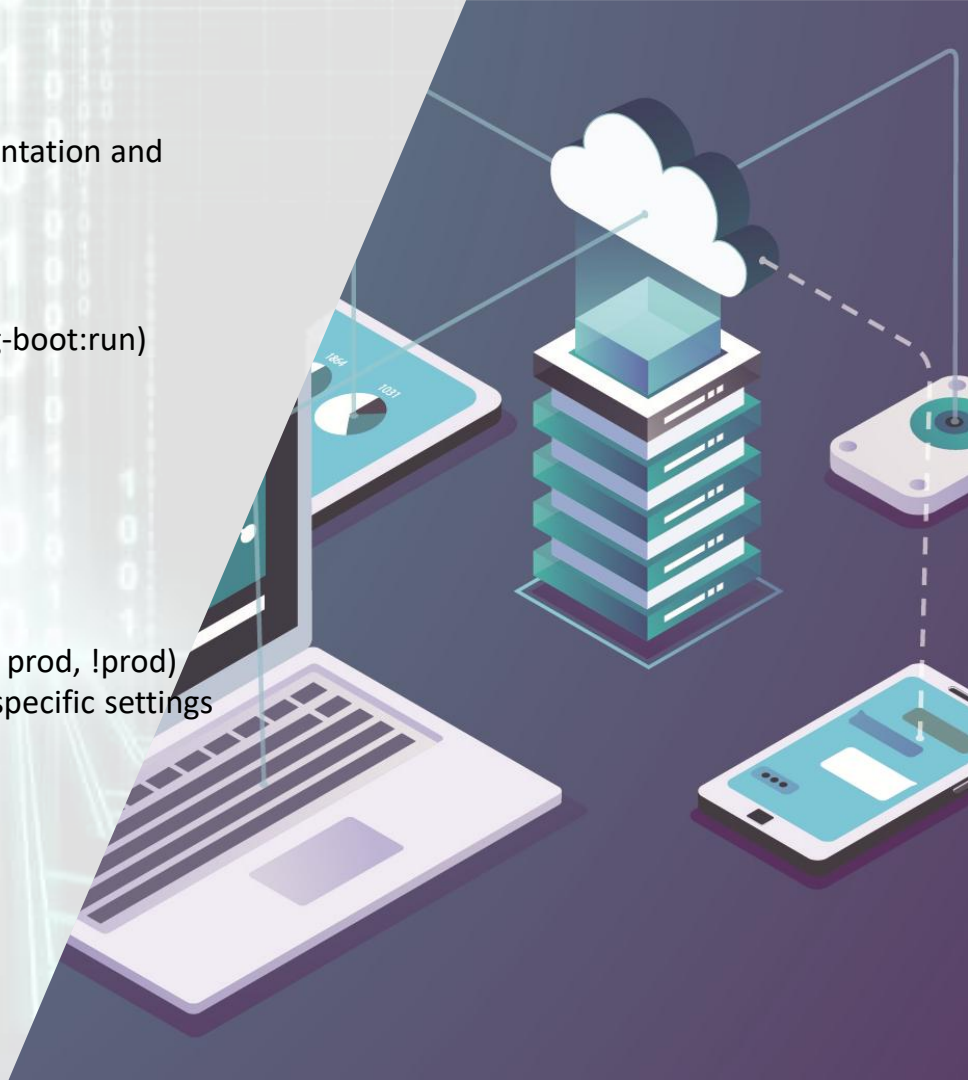
API Documentation
- **Springdoc OpenAPI** or **Swagger UI** – interactive API documentation and testing interface

Build & Dependency Management
- **Maven** – build tool and dependency manager
- spring-boot-maven-plugin – for running the app (mvn spring-boot:run)

Development Tools
- **IntelliJ IDEA** (or Eclipse/VS Code) – IDE for coding
- **Spring Boot DevTools** – auto-restart during development (if enabled)

Profiles & Configuration
- **Spring Profiles** – for environment-based configuration (dev, prod, !prod)
- **application.yml/application.properties** – for environment-specific settings

Security & Authentication
- **Spring Security** configuration with JWT
- **Custom UserDetailsService** for loading users from DB
- **Password hashing** (likely BCrypt)




THANK YOU