Semester 3

Design document

LMS

sumaya gabr

Contents

[1. Introduction 1](#_Toc147498086)

[2. Architecture Constraints and Design Decisions 1](#_Toc147498087)

[2.1 Spring Boot 1](#_Toc147498088)

[2.2 React 1](#_Toc147498089)

[2.3 MySQL 2](#_Toc147498090)

[3. C4 Model Diagrams 2](#_Toc147498091)

[3.1 Level 1: System Context Diagram 2](#_Toc147498092)

[3.2 Level 2: Container Diagram 2](#_Toc147498093)

[3.3 Level 3: Component Diagram 3](#_Toc147498094)

[4. Explanations and Design Principles 3](#_Toc147498095)

# 1. Introduction

This design document outlines the architectural constraints, design decisions, and provides C4 model diagrams for the Learning Management System (LMS). It also connects the architectural choices with software design principles such as SOLID, KISS, DRY, and YAGNI.

# 2. Architecture Constraints and Design Decisions

## 2.1 Spring Boot

Spring Boot is chosen for building the backend of the LMS due to its:

* Simplicity: It simplifies the setup and development of Java applications.
* Community and Ecosystem: It has a large and active community, extensive documentation, and a rich ecosystem of libraries.
* Additionally, it aligns with the school's curriculum and best practices in enterprise Java development.

## 2.2 React

React is selected for the frontend of the LMS due to its:

* Component-Based: React's component-based architecture promotes reusability and maintainability of UI elements.
* Virtual DOM: It uses a virtual DOM to optimize rendering performance, ensuring a responsive user interface.
* Large Community: React has a strong community and numerous libraries for UI development.
* The school's curriculum also includes React as a recommended technology for frontend development.

## 2.3 MySQL

MySQL is chosen as the database for the LMS because of:

* Reliability: MySQL is known for its reliability and data integrity, making it suitable for handling student and course data.
* Scalability: It can handle large datasets and scale with the growth of the LMS.
* Compatibility: MySQL is compatible with various programming languages and frameworks.
* This choice is consistent with the school's database curriculum

# 3. C4 Model Diagrams

## 3.1 Level 1: System Context Diagram

The Level 1 diagram provides an overview of the LMS and its interaction with users.

A diagram of a diagram

Description automatically generated

## 3.2 Level 2: Container Diagram

The Level 2 diagram details the major containers within the LMS.

A diagram of a diagram

Description automatically generated

## 3.3 Level 3: Component Diagram

The Level 3 diagram breaks down the components within the Core Application.

A diagram of a diagram

Description automatically generated

# 4. Explanations and Design Principles

In the Learning Management System (LMS) architecture, the application design adheres to the SOLID principles, which emphasize good software design practices. Here's how the SOLID principles are applied in the LMS:

**Single Responsibility Principle (SRP):**

Example: The CreateCourseUseCase, UpdateCourseUseCase, and DeleteCourseUseCase classes each have a single responsibility related to user management. They encapsulate the logic for creating, updating, and deleting user accounts, respectively.

**Open/Closed Principle (OCP):**

Example: The use cases (e.g., CreateCourseUseCaseImpl, UpdateCourseUseCaseImpl) are defined by interfaces (CreateCourseUseCase, UpdateCourseUseCase) that other developers can implement to extend the system's functionality. Existing code that uses these interfaces doesn't need to change when new use cases are added.

**Dependency Inversion Principle (DIP):**

Example: The core components of the LMS (e.g., controllers, services) depend on the use case interfaces (e.g., CreateCourseUseCase, UpdateCourseUseCase) rather than concrete implementations. This decouples the high-level components from low-level details, promoting flexibility and testability.