**COMP301.2 - Software Architecture and Tools**

**Term Project Report**

**Submitted By: Yasin Aydın , Batuhan Kaya , Hüseyin Can Kayım**

**Submitted To: Kemal Serdaroğlu**

**Date: 16.01.2025**

**Section:2**

**Project Title (Topic):**  
**Microservices-Based Educational Management System**  
This project implements an educational management system using microservices architecture. Each service is specialized for different functionalities, including announcements, authentication, course management, messaging, and user management.

**Group Members:**

* **Yasin Aydın** –Frontend Developer (User interface and messaging system)
* **Batuhan Kaya** – Backend Developer (Development of Auth and User services)
* **Hüseyin Can Kayım** – DevOps Engineer (Docker integration and deployment)

**Abstract:**  
This project aims to develop an educational management system using microservices architecture for educational platforms. Each service focuses on a distinct functionality. Backend services written with Spring Boot are containerized using Docker to ensure scalability. Users can manage announcements, course content, and messages while performing authentication and authorization tasks. The project adopts modern software development principles, offering a sustainable and modular structure.

**Introduction:**  
In modern educational platforms, user management, course tracking, and messaging functionalities are crucial. This project aims to provide these features through independent microservices, enhancing scalability and flexibility. Microservices architecture allows each component to be developed and managed independently.

**Research and Literature Review:**

* Martin Fowler, *Microservices Guide* – A foundational resource explaining the advantages and disadvantages of microservices.
* Spring Boot Documentation – The primary library used in the development process.

**Design:**

* **Class Diagram** – Classes for User and Role management.
* **Use Case Diagram** – Representing user authentication and course creation functionalities.
* **Database Design** – Separate databases or schemas for each service.

**Methodology:**

* RESTful APIs were developed for each service using Spring Boot.
* Containerization was achieved using Docker.
* Authentication was implemented with OAuth2.

**Implementation and Analysis:**  
Each microservice was developed independently and deployed in Docker containers. Scalability tests demonstrated improved performance compared to traditional monolithic architectures.

**Further Studies and Recommendations:**

* Integration of a real-time notification system.
* More complex deployment strategies using Kubernetes.

**References and Documentation:**

* Fowler, M. *Microservices*.
* Official documentation for Spring Boot and Docker.