LinkedHU_CENG
Architecture Notebook

1. Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It serves as a communication medium between the software architect and other project team members regarding architecturally significant decisions which have been made on the project

2. Architectural goals and philosophy

The Software Architecture Notebook provides a comprehensive architectural overview of the LinkedHu_CENG It presents a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system. The architecture should provide ease of use to students, graduates and academics so that there is no problem during the use of the features determined in the project (meeting, chat, presentation, upload files, etc.). When developing the architecture, all the requirements from the old documents will be taken into account.

3. Assumptions and dependencies

Team members need to have good teamwork skills.They need to be able to work in harmony in order to make the app right and on time.

Application is going to be on the web, therefore the internet will be needed.

Students and academicians of Hacettepe University Computer Engineering department will be able to efficiently communicate with each other , and academic information will be attainable by them.

Space/capacity of the database that is needed for storing information does not need to be too large since it will be limited to only Hacettepe University Computer Engineering academicians and students(current or former).

The passwords will be secured by encryption for ensuring users security.

4. Architecturally significant requirements

System is on the web.

System has to respond within 5 seconds.

Encryption for passwords is a must for security reasons.

Chat between users has to be simultaneous.Also it needs to be stored.

5. Decisions, constraints, and justifications

- GitHub is used to develop the project in order to make development easier.The system is flexible, adding and removing new things is easy.
- Branches are used in GitHub. Every developer uses his own branch to add changes to the project.
- The same IDE is used across all team members to be more compatible.

- Open source technologies are used such as MySQL, SpringBoot, ReactJS.
- The code must be clean and object oriented.
- The system is a website that requires an internet connection to work.

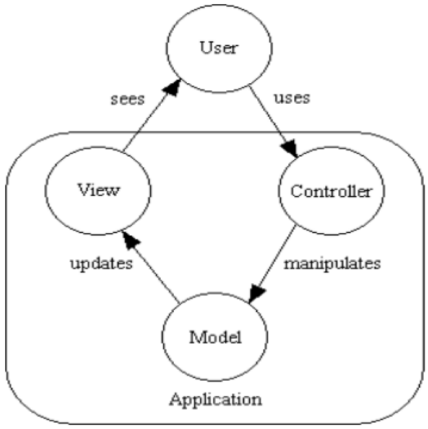
6. Architectural Mechanisms

The Model-View-Controller Architectural Pattern (MVC) is basically applied for interactive software systems. The Model component contains the core functionality and data, the View component gives information to the user and the Controller component handles user input using validation. View and Controller together comprise the user interface. A change-propagation mechanism ensures consistency between the user interface and the model

Model:Model component of MVC encapsulates data and functionality. It is independent of its input and output behavior of the interactive system. It contains no logic describing how to present the data to a user.

View:The view component interacts with the user and always displays the information to the user. It knows how to access the model's data, but does not know how to change it.

Controller:The controller mediates between the model and the view and handles state changes. So the controller always follows the change propagation mechanism.



7. Key abstractions

Administrator: User abstraction that performs the most authoritative operations on the system
Student/Academic/Graduate: End user abstractions who will use the system and perform operations on the system.
Database: The abstract layer where all actions and records on the system are kept.
Business Logic: The abstract unit where logical operations, services and security operations take place.

8. Layers or architectural framework

In this project, there is primarily a web user interface and a UI layer developed with React. After this layer, user requests are directed to the server, that is, to the back-end layers. The spring framework provides these layers.

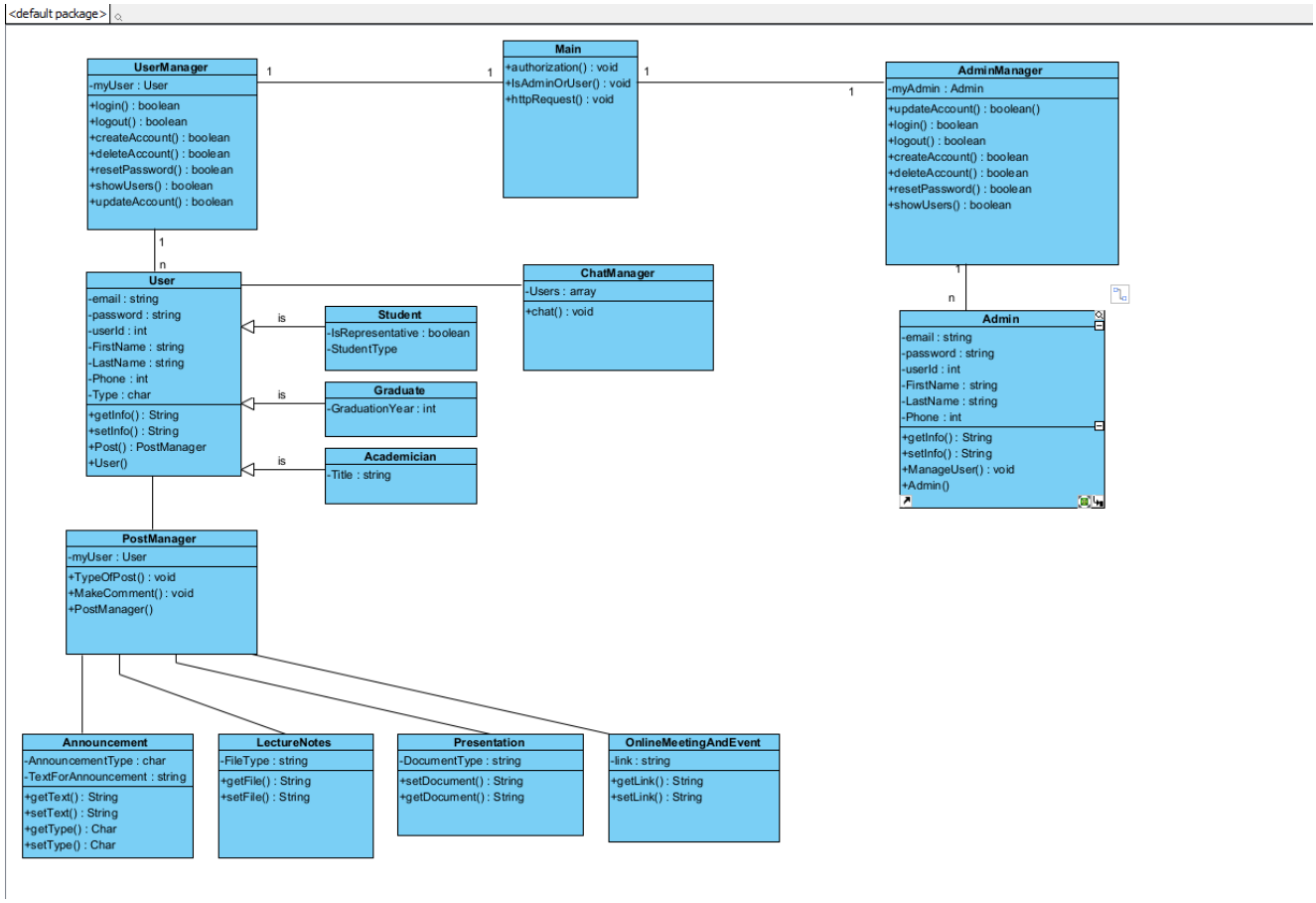
Incoming requests are first met by the Controller layer. In this layer, the HTTP request is filtered and sent to the Business Logic layers, that is, to the layer where the Services are. Then, database operations are performed by accessing the Data Access layer. These layers are designed following the MVC architecture.

9. Architectural views

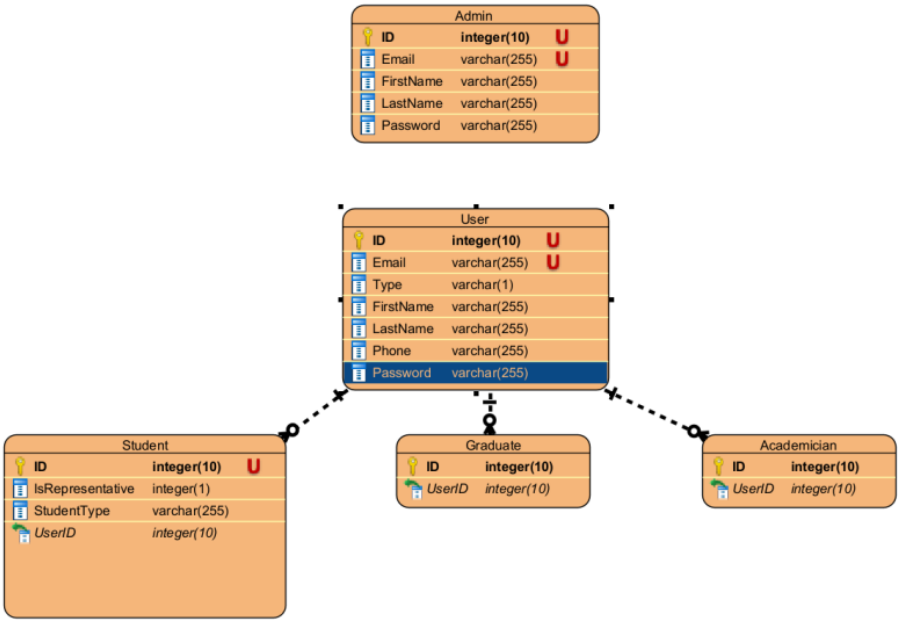
All of the diagrams below are in pdf format and may be accessed in the same folder as this article.

- **Class Diagram :** These diagrams show a system's structure by displaying the system's classes, attributes, operations (or methods), and object connections.
- **Updated ER Diagram :** This diagram illustrates the connections between entity sets in a database. An object, or a data component, is an entity in this context. A collection of comparable entities is referred to as an entity set. These entities can have qualities defined via attributes.
- **Updated Use Case Diagram :** Simplest is a visual depiction of a user's engagement with the system that depicts the user's relationship with the many use cases in which the user is involved.
- **Package Diagram :** The dependencies between the packages that make up a model are shown here.
- **Component Diagram :** The structure of an arbitrarily complicated system is illustrated in this diagram.
- **Deployment Diagram :** The physical deployment of artifacts on nodes is modeled in this diagram.

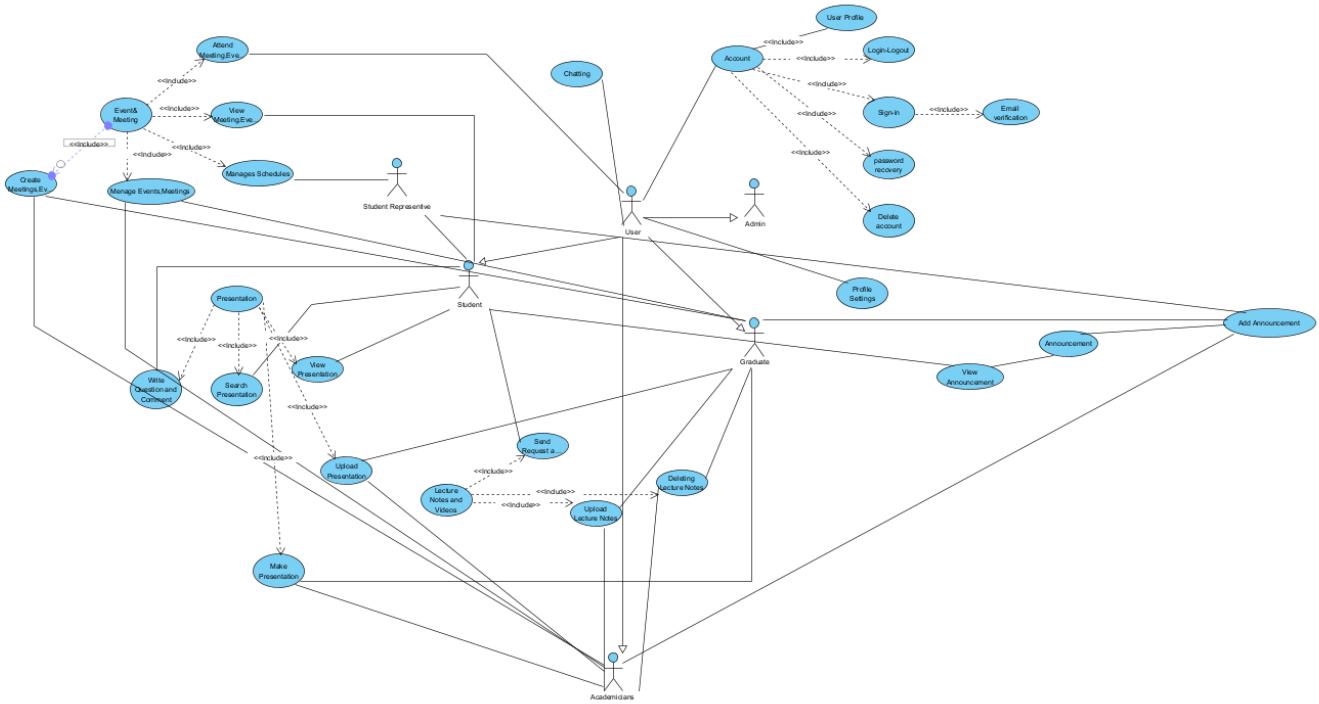
- **Class Diagram:**



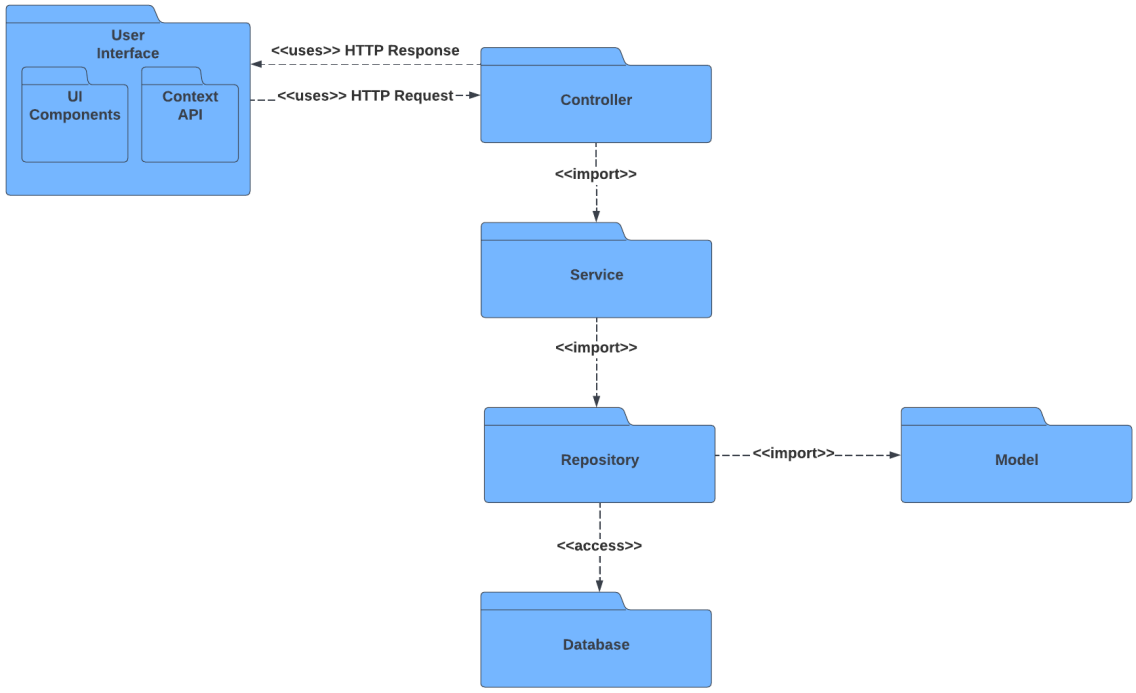
- **Updated ER Diagram:**



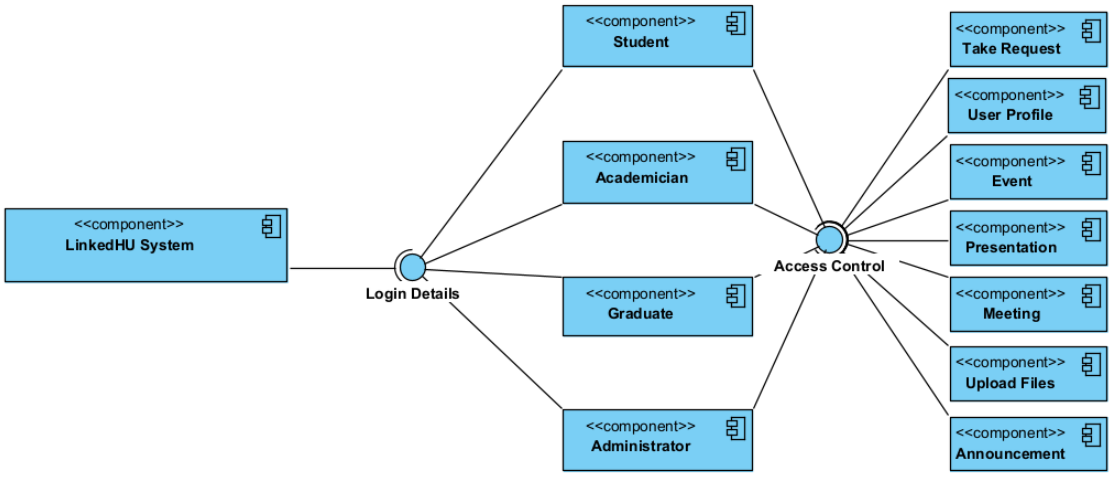
- **Updated Use Case Diagram:**



- **Package Diagram:**



- **Component Diagram:**



- **Deployment Diagram:**