ICES4HU
Architecture Notebook

1. Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.

2. Architectural goals and philosophy

The architectural approach aims to develop a course and instructor evaluation system that is both user-friendly and efficient. Key considerations include integration with existing legacy systems, ensuring long-term maintenance and stability, and optimizing overall system performance.

- Make sure the system is easy to use and straightforward to use for all users.
- The system must allow creation and management of course and instructor surveys by the instructors with ease.
- Develop a secure and dependable platform for managing and storing evaluation data.

Critical issues that must to be addressed by the architecture:

- Preventing hardware dependencies to get rid of system failures.
- Ensuring the system can be functioning under unusual conditions such as high traffic volume , etc.
- Ensuring the system can handle large amounts of data without a performance decrease.

3. Assumptions and dependencies

List of Assumptions and Dependencies for the ICES4HU project architecture :

- The system assumes that instructors know how to use it well.
- The system assumes that every user can connect to the internet smoothly and use up-to-date web browsers.
- The architecture relies on having enough computing resources and storage capacity to handle and store data efficiently.
- The architecture relies on having a skillful development team to design, build, and maintain the system for the long-term.
- The design of the system relies on the successful integration with the university's current systems, such as the student information system, to function effectively.
- The system assumes that there are no laws or regulations that would prevent the proposed architecture from being used.

4. Architecturally significant requirements

For our ICES4HU project, there are several important requirements that need to be met in order to create a high-quality system. These requirements include making sure the system can handle a large number of users and data (scalability), ensuring that data is safe and protected from unauthorized access (security), ensuring that the system is stable and dependable (reliability), providing the ability to analyze data (data analysis capabilities), allowing the system to integrate with other external systems (external integration), and making sure the system is easy to use for both technical and non-technical users.

- ➢ The system should ensure the safety of user information and to prevent unauthorized entry, the system needs to be secure.
- ➢ The system should be able to handle larger amounts of data and traffic as usage increases.
- ➢ The system should be dependable and accessible, ensuring that users can use it at any time without interruptions.
- ➢ The system should be user-friendly and intuitive, allowing all users to easily interact with its features and functions regardless of their technical expertise.

5. Decisions, constraints, and justifications

Decisions, constraints, and justifications for ICES4HU:

- ➔ Maintainability is a crucial aspect of ICES4HU, and it will be achieved by designing the system architecture with fine-grained and self-contained components that can be easily modified. Data producers and consumers will be separated, and shared data structures will be avoided to ensure maintainability.
- ➔ Developers will prioritize the maintainability of the system over performance to ensure that it does not become outdated quickly. They will create small and easily modifiable components, even if it slightly impacts performance. They will also ensure that the system's usage flow is as seamless as possible by following the time limits specified in the SRS document's system qualities section.
- ➔ Ensuring the security of student data is crucial for ICES4HU. To achieve this, the system is strengthened with the enhancements provided by the Spring Boot Framework. These will be implemented as separate modules, and the framework's protocol will be followed with adequate documentation to prevent any security vulnerabilities.
- ➔ To make sure that the system is dependable and reliable , developers will perform regular regression tests to check the architecture's compatibility with new design changes. This will allow them to add new requirements on top of an established and well-maintained system.

6. Architectural Mechanisms

Architectural Mechanism 1

This mechanism for UI-GUI. Implementing a user-friendly interface. Our aim is to prioritize the user experience.

Architectural Mechanism 2

This mechanism for authentication. Only the permitted users can access the system.

Architectural Mechanism 3

This mechanism for data management. The implementation of the database (PostgreSQL). Interaction between UI and DB.

Architectural Mechanism 4

This mechanism for optimization, where we handle exceptions and optimizations. It also covers improvements such as efficiency and speed.

7. Key abstractions

There are a number of critical reflections that characterize the basic concepts in ICES4HU. We may list them under basic titles.

A course is a representation of a particular class offered by Hacettepe University and includes details like the course id, instructor, semester and department. Students evaluate courses at the end of each semester, and the evaluation data is used to pinpoint areas where the course could be improved.

An instructor at Hacettepe University, is also a faculty member. Students rate instructors at the end of each semester, and those ratings are used to enhance the caliber of the courses.

An evaluation is a student's evaluation of a course or instructor. Evaluations include forms and points on the performance of the instructor or the course in various areas, such as organization, responsiveness to student needs, and clarity of instruction.

A user is an individual who has access to the system, such as a student, instructor and department manager. Users can log in to view existing evaluations and submit new ones. The user authentication and authorization management, as well as the retrieval and updating of user data, will likely be handled by the user abstraction.

The final report or result is an analysis of the evaluation data that contains metrics like average ratings, comments, and historical trends. The system produces reports to give feedback to departments and instructors as well as to pinpoint areas where course quality could be enhanced.

These fundamental abstractions serve as the basis for the ICES4HU and will serve as a guide for the system's development. Classes for handling user authentication and authorization will be needed for the User abstraction. Concentrating on these fundamental abstractions makes it possible to design the system in a modular and extensible way, making it simple to add new features and improvements in the future.

8. Layers or architectural framework

To accomplish its objectives, ICES4HU will combine Model-View-Controller (MVC) and Microservices architectures. The presentation layer and system's business logic will be organized using the MVC architecture. The View component will present the data to the users. The Controller component will serve as a bridge between the Model and View components, processing user input and updating the Model and View as required.

The back-end services that support the application will be managed using the Microservices architecture. User authentication and data management are just a few of the various system functionalities that the Microservices will be in charge of managing. Each Microservice will be created to handle a particular task, and they will all communicate with one another through APIs.

By combining these two architectures, the ICES4HU will be able to improve the caliber of courses at Hacettepe University. They will be able to evaluate courses and instructors using a user-friendly interface provided by the MVC architecture, while the system will be able to handle large amounts of data and complex workflows effectively thanks to the Microservices architecture. Future scaling and updating of the system will be simple thanks to this architecture.

9. Architectural views

[Describe the architectural views that you will use to describe the software architecture. This illustrates the different perspectives that you will make available to review and to document architectural decisions.]

Recommended views

- **Logical:** Describes the structure and behavior of architecturally significant portions of the system. This might include the package structure, critical interfaces, important classes and subsystems, and the relationships between these elements. It also includes physical and logical views of persistent data, if persistence will be built into the system. This is a documented subset of the design.
- **Operational:** Describes the physical nodes of the system and the processes, threads, and components that run on those physical nodes. This view isn't necessary if the system runs in a single process and thread.
- **Use case:** A list or diagram of the use cases that contain architecturally significant requirements.