BILKENT UNIVERSITY

CS492 - Senior Design Project II

Detailed Design Report

T2423 - Evalio

Eren Hayrettin Arım	22002306
Ahmet Reşat Demir	22002299
Mehmet Burak Demirel	22003396
Dilara Mandıracı	22101643
Yusuf Toraman	22002885

TABLE OF CONTENTS

1. Introduction

Even if the traditional paper-based exam method is widely used, it often falls short of accurately measuring student success and meeting modern educational needs. Research from Purdue University shows the importance of mapping and analyzing questions during the exam preparation phase to enhance exam quality and performance [1]. In response, our team aims to develop Evalio, an advanced exam database and management application designed to streamline exam creation and improve assessment effectiveness. Evalio enables instructors to create practical, high-quality exams through a shared, searchable question database that supports multiple courses. With features like question usage tracking, editing, versioning, export options, and detailed performance analytics, Evalio helps create balanced assessments while managing assignments and exams. Additionally, it offers secure, multi-university collaboration. It ultimately contributes to improved teaching and learning outcomes.

1.1 Purpose of the system

Traditional exam preparation and evaluation methods often present significant challenges for educators and academic institutions. Managing large question banks, ensuring balanced assessments, analyzing student performance, and organizing exam result review sessions require extensive effort and time. Additionally, accreditation processes necessitate structured documentation of academic materials, further increasing the administrative workload for instructors. To address these challenges, Evalio is designed as a comprehensive, scalable, and secure platform that streamlines exam creation, question management, performance analysis, and academic portfolio organization for universities and educational institutions.

The primary objective of Evalio is to facilitate efficient exam preparation by providing instructors with a structured, searchable question database. The platform enables users to create, store, and manage a vast collection of categorized questions with tagging, classification, and version control. Instructors can manually select questions or use automated exam generation based on predefined criteria

such as topic distribution and difficulty levels. Furthermore, the Exam Builder feature supports multiple export formats, including LaTeX, PDF, and Google Docs, ensuring seamless integration with academic workflows.

Beyond exam creation, Evalio provides advanced data analytics to enhance performance evaluation. The platform allows institutions to analyze success rates per question, exam difficulty trends, and correlations between student performance and attendance. By leveraging real-time data visualization tools such as graphs and statistical reports, instructors can make informed decisions to improve the quality of assessments. Additionally, Evalio supports accreditation efforts by generating compliance reports aligning with ABET and MÜDEK standards, ensuring educational programs meet regulatory requirements.

Another key aspect of Evalio is its ability to streamline post-exam result viewing and objection handling. Teaching assistants can efficiently organize exam result viewing sessions by creating structured time slots for students to review their graded exams. Students can book available slots on a first-come, first-served basis through the system, ensuring fairness and transparency. Additionally, Evalio facilitates structured objection handling, allowing students to submit concerns securely while instructors and TAs can efficiently review, manage, and resolve disputes.

In addition to exam management, Evalio serves as a comprehensive portfolio management system. Educators can create and maintain detailed course portfolios that include exam samples, grading records, attendance sheets, and accreditation documentation. The system automates the generation of structured PDF portfolios, reducing manual effort while ensuring compliance with institutional and accreditation requirements.

Security and scalability are fundamental to Evalio’s design. The system implements Role-Based Access Control (RBAC) to restrict access based on user roles, ensuring that only authorized instructors and teaching assistants can access sensitive data. User authentication is reinforced through institutional verification, preventing unauthorized users from registering. The platform’s microservices-based

architecture ensures modularity, allowing independent updates to various system components without disrupting overall functionality.

By integrating question management, exam building, performance analysis, result viewing, and portfolio creation into a single platform, Evalio aims to significantly enhance efficiency, improve assessment quality, and support inter-university collaboration. The system reduces the administrative burden on educators, ensuring they can focus on delivering high-quality education while maintaining academic integrity and compliance with accreditation standards.

1.2 Design Goals

Evalio is designed to be a comprehensive, efficient, and scalable platform that addresses the needs of instructors, teaching assistants, and academic institutions. The system's architecture and functionality are developed with key design goals to ensure usability, security, maintainability, and overall system effectiveness. Below are the primary design goals that guide the development of Evalio:

1.2.1 User Friendliness / Usability

Evalio aims to be user-friendly with its easy and understandable interface designed to facilitate users while creating exams by taking advantage of the modularity of the questions. We know that currently, instructors can create exams using different tools, so this product focuses on providing advanced ease of use. Therefore, the user interface should be as simple as possible while providing the functionality of the basic modules.

1.2.2 Maintainability

Since Evalio targets various types of end users, such as instructors, students, and teaching assistants, and includes different modules, we aim to make it as sustainable as possible by splitting the architecture into services. A modular codebase will enable isolated updates without breaking other services or modules.

1.2.3 Reliability

Evalio aims to be reliable. Given the variety of end-user types and use cases, a single point of failure should not affect other modules, and any failure should be

quickly recoverable. We aim to increase reliability by splitting the application's architecture into separate services.

1.2.4 Security

Since we aim to support different user types (e.g., instructors and institutions), the application should include a user authentication and authorization system to ensure that each user type has the correct permissions to access specific modules. Additionally, passwords and potentially sensitive information should be stored in a coded form in the database to ensure security.

1.3 Definitions, Acronyms, and Abbreviations

This section provides definitions and explanations of terms, acronyms, and abbreviations used throughout the document to ensure clarity and consistency.

1.3.1 Definitions

- **Evalio** – Our digital platform for managing exam questions, creating assessments, analyzing student performance, and organizing exam-related activities.
- **Question Database** – A repository that stores, categorizes, and manages exam and quiz questions for reuse and analysis.
- **Exam Builder** – A tool within Evalio that allows instructors to manually or automatically generate exams by selecting and arranging questions.
- **Portfolio Builder** – A module that enables teaching assistants and instructors to compile course portfolios, including exam papers, attendance sheets, and grading records, for accreditation and institutional documentation.
- **Objection Session Manager** – A feature that allows teaching assistants to schedule and manage student review sessions for graded exams.
- **Role-Based Access Control (RBAC)** – A security mechanism that restricts system access based on predefined roles (e.g., Instructor, TA, Admin).
- **Accreditation Reports** – Institutional compliance reports (e.g., ABET, MÜDEK) generated by Evalio to assist in meeting regulatory requirements.

1.3.2 Acronyms and Abbreviations

- **ABET** – Accreditation Board for Engineering and Technology
- **API** – Application Programming Interface
- **CI/CD** – Continuous Integration / Continuous Deployment
- **DBMS** – Database Management System
- **JWT** – JSON Web Token
- **LMS** – Learning Management System
- **LXP** – Learning Experience Platform
- **MÜDEK** – Association for Evaluation and Accreditation of Engineering Programs in Turkey
- **RBAC** – Role-Based Access Control
- **UI/UX** – User Interface / User Experience
- **UML** – Unified Modeling Language

1.4 Overview

This document serves as the detailed design specification for Evalio, an advanced exam management and analytics platform aimed at improving assessment processes in academic institutions. It provides a technical roadmap for the development team, defining the system architecture, key components, data management strategies, security policies, and scalability considerations. The report ensures that every aspect of Evalio's design is structured, efficient, and aligned with best practices for performance and maintainability.

The document begins with an overview of the system architecture, describing how Evalio is structured and how its different components interact. It introduces the proposed architecture, breaking down core subsystems such as the client-side interface, API gateway, logic services, data storage, and external integrations. Each subsystem is explained in terms of its role, responsibilities, and interaction with other parts of the system, ensuring a cohesive and maintainable design. Additionally, data persistence ways, and access control mechanisms are outlined to enhance security, performance, and reliability.

Beyond the technical framework, this report includes test cases and engineering constraints. Finally, the document details team collaboration strategies, development workflows, and the tools used throughout the project. By providing a structured and well-defined design, this report ensures that Evalio is developed as a well designed, efficient, and future proof solution for modern exam management.

2. Current software architecture

2.1 Evalio Current Architecture

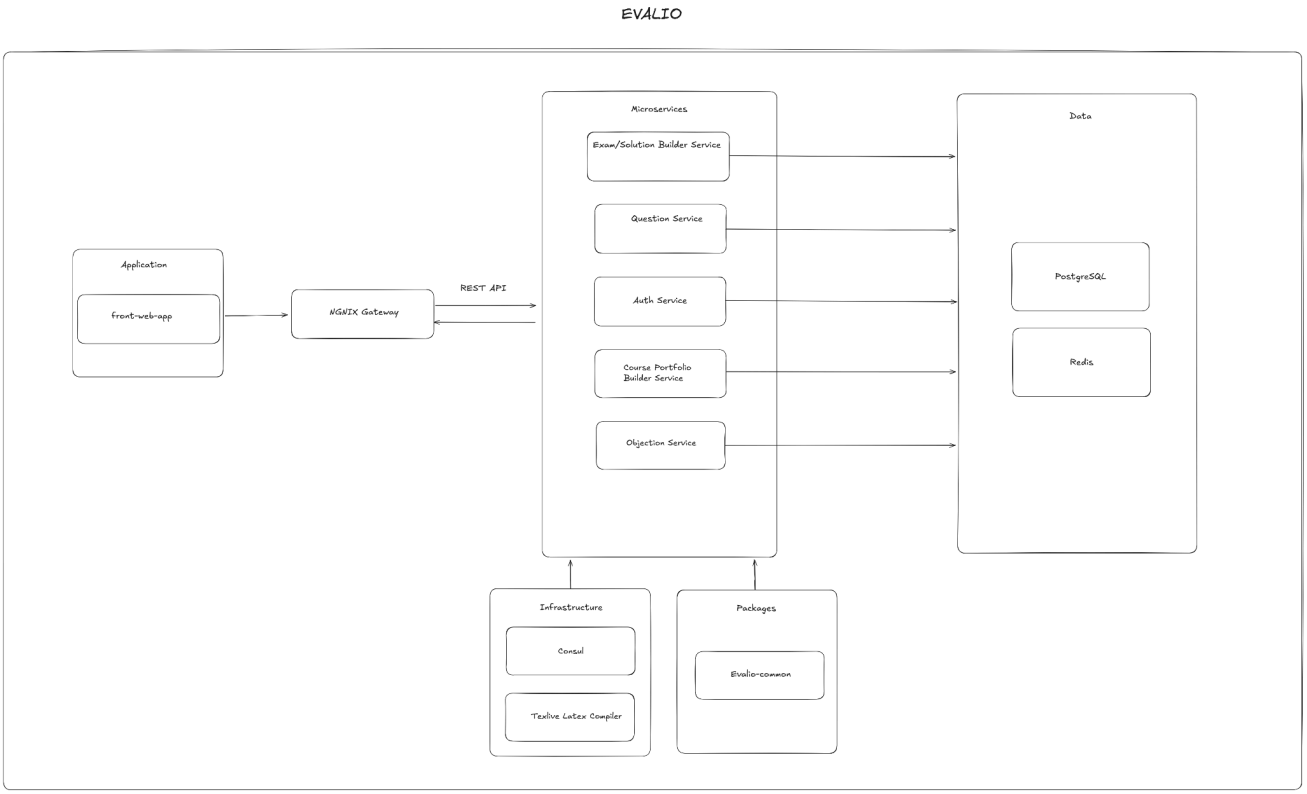

Figure 1: Current High-Level System Architecture Diagram

Evalio's architecture follows a microservices-based approach, ensuring modularity, scalability, and maintainability. It consists of a frontend, API gateway, microservices, a data layer, and shared utilities. Microservices handle exam creation, question management, authentication, course portfolio management, and objections, operating independently for seamless updates and expansion. The data layer includes PostgreSQL for structured storage and Redis for caching, ensuring high performance. Consul supports service discovery, while TeXLive LaTeX Compiler enables LaTeX-based exam generation. Evalio also uses Evalio-Common, a shared

library with reusable database models, authentication logic, and schema definitions, ensuring consistency across services. This modular design enables secure access control, efficient data handling, and streamlined exam management for universities and institutions.

2.2 Competitors, Alternative and Current Solutions

Evalio is a modular and scalable exam management system focusing on efficient exam creation, question management, and institution collaboration. Compared to most current systems, the system provides a structured question database, detailed analysis of question performance, and secure collaboration among several universities.

LearnUpon (Competitor 1 - LMS Platform)

LearnUpon is an LMS that provides course and exam management to organizations. However, it is primarily suited for course delivery rather than exam-focused modular question management.

Key Differences & Advantages of Evalio:

- Evalio is neither an LMS nor an LXP but rather a dedicated examination management platform and thus, examination-building is more dynamic, structured, and data-centric.
- Unlike LearnUpon, Evalio facilitates cross-institutional question sharing so teachers can tap into a centralized repository.
- In-depth question analysis in Evalio helps instructors to fine-tune their tests with student performance insights, which LearnUpon lacks.

Potential Challenges & Limitations:

- LearnUpon already has a user base, which will give Evalio less chance of being accepted into institutions already invested in an LMS.
- Since LearnUpon integrates with numerous third-party platforms, Evalio may demand LMS compatibility features for broader applications.

- Some institutions may prefer an entire LMS with exam management, meaning Evalio must position itself as a complementary, specialized tool rather than a competitor to complete LMS platforms.

How We Overcome These Challenges:

- Evalio will enable integration into current LMS platforms (e.g., LearnUpon, Moodle, and Canvas) using APIs and won't replace them.
- Emphasizing modular test development and question performance tracking can be a basis for encouraging instructors to implement Evalio alongside LMS capabilities.

HackerRank (Competitor 2 - Coding Assessment Platform)

HackerRank is a typical tech testing platform firms and universities employ to test students/applicants via automated coding exams. While Evalio shares the general goal of constructing well-organized exams, its applicability is broader, covering many fields rather than just coding.

Key Differences & Advantages of Evalio:

- Evalio supports multi-format exams (text exams, multiple choice, problem-solving, LaTeX-based math exams, etc.), whereas HackerRank is coding-focused.
- Cross-institution question sharing allows universities to collaborate and build wealthier question banks, which HackerRank does not emphasize.
- Evalio provides test-making in different forms (PDF, LaTeX, Google Docs), more suitable for classroom settings.

Potential Challenges & Limitations:

- HackerRank utilizes an automated grading system, whereas Evalio is more manual and organized in exam construction.
- Specific customers may expect automatic question creation and scoring, which Evalio does not fully support.

How We Overcome These Challenges:

- Evalio introduces AI-assisted features such as converting images to latex.

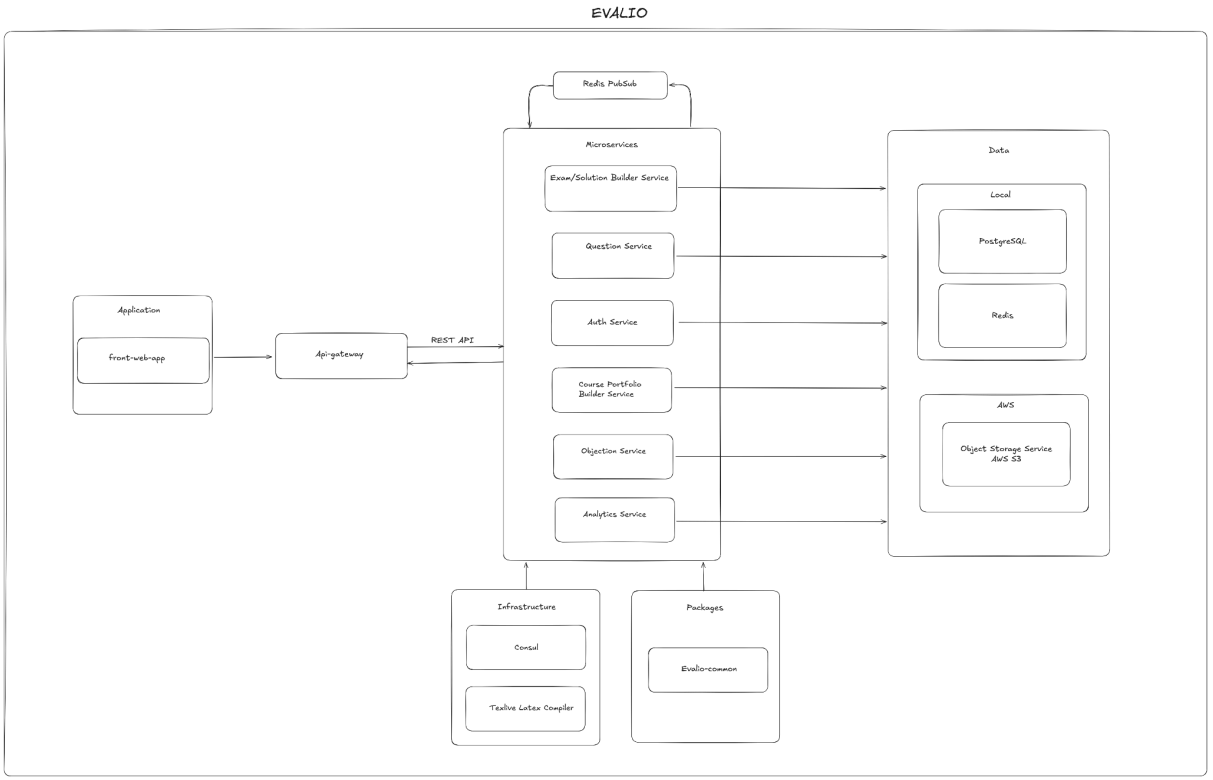

Figure 2: Proposed High-Level System Architecture Diagram

3. Proposed software architecture

3.1 Overview

Evalio follows a distributed microservices architecture, which separates core functionalities into independent services. The system consists of the following key components:

- **Application Layer**: A web-based front-end that provides users with an intuitive interface for exam creation, question management, and analytics visualization.
- **API Gateway**: A central entry point for all client requests, ensuring secure communication between services and managing request routing.

- **Microservices Layer**:
    - Exam/Solution Builder Service: Manages exam creation, LaTeX rendering, and question selection.
    - Question Service: Stores, retrieves, and manages exam questions with metadata (difficulty, success rate, versioning).
    - Authentication Service (Auth Service): Handles user authentication, role-based access control (RBAC), and session management.
    - Course Portfolio Builder Service: Enables the compilation of institutional course portfolios for accreditation.
    - Objection Service: Organizes post-exam result viewing sessions and manages student objections.
    - Analytics Service: Generates performance reports, success rate statistics, and accreditation compliance reports.
- **Data Layer**: Manages storage through:
    - PostgreSQL (relational database for structured exam and user data)
    - Redis (caching and real-time session management)
    - AWS S3 (cloud-based object storage for scanned exam documents and portfolio files)
- **Infrastructure Layer**:
    - Consul (for service discovery and health checks)
    - TeXLive LaTeX Compiler (to generate LaTeX-based exam PDFs)
- **Messaging and Event Handling**:
    - Redis Pub/Sub is used for inter-service communication, ensuring event-driven updates.

3.2 Subsystem Decomposition

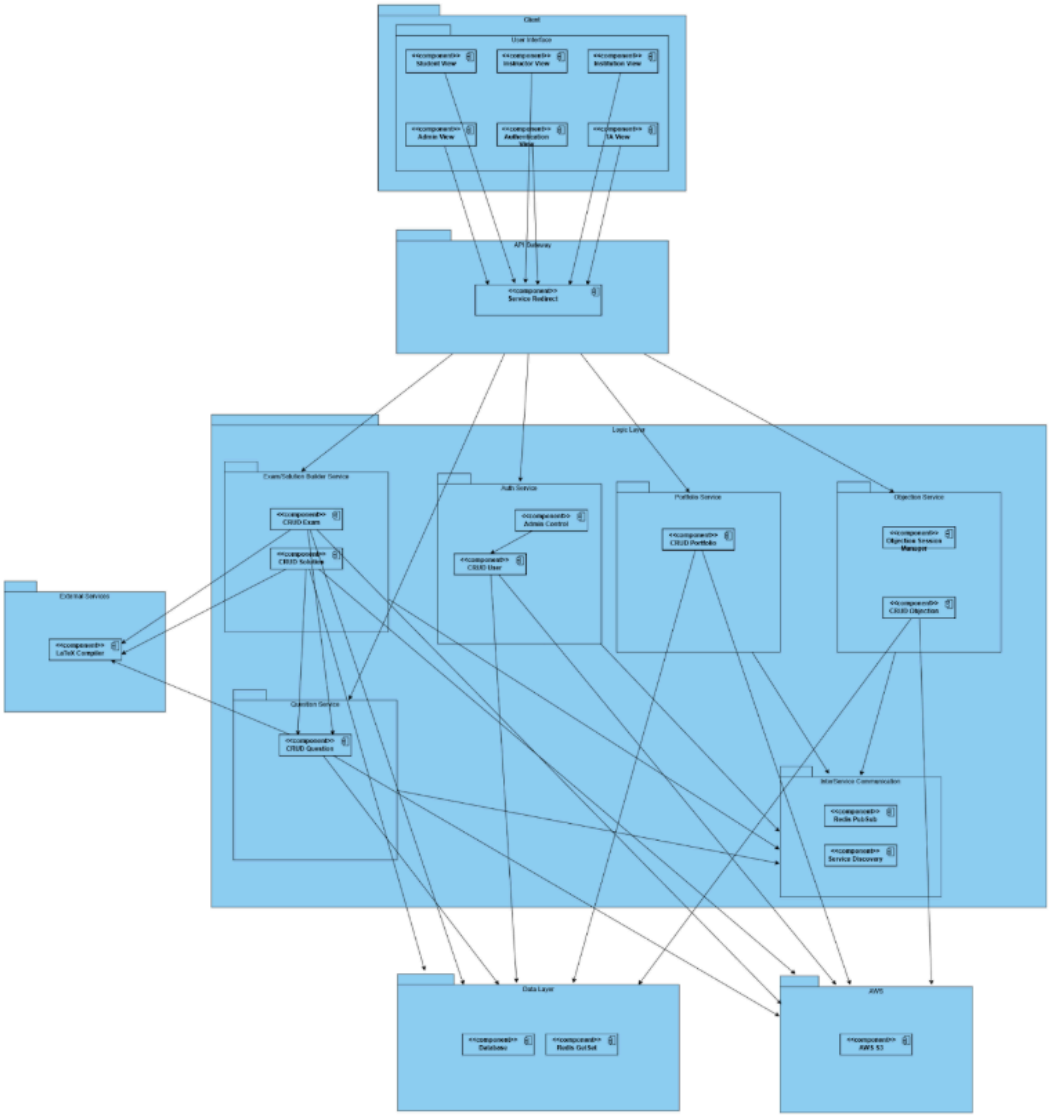

Figure 3: Proposed Subsystem Decomposition Diagram

3.3 Hardware/Software Mapping

Evalio is a software-only system with no dedicated hardware components. It operates on cloud-based infrastructure, utilizing microservices, databases, and caching for scalability and reliability. No hardware-software mapping is required beyond standard server deployments.

3.4 Persistent Data Management

Evalio keeps its structured data in PostgreSQL and utilizes Redis for cache purposes, thus ensuring scalable and efficient data storage. The system also features Amazon S3 as an object storage for storing large documents in the Portfolio Builder Service.

To enhance database scalability and performance, we aim to partition large data sets with table partitioning and query optimization with indexes. S3 lifecycle policies can also manage storage expenses by automatically migrating older documents to lower-cost storage classes. With a responsive and cost-effective system, these solutions ensure effective data persistence, maintainability, and long-term scalability.

3.5 Access Control and Security

This application implements a strict role-based access control system. This is because it contains sensitive information, such as exam questions, making it a critical security concern to control who can access different application areas. We have integrated role-based access control and defined specific access rules.

- Registration can only be performed by the institution. Instructors or TAs cannot register; only the institution can add them to the system.
- When an institution signs in, they gain access only after admin approval.
- Questions are tagged as public or private. Public questions are visible to all registered users in the system, while private questions can only be accessed by individuals affiliated with the respective institution.
- Students cannot register or access the system. They can only access the application through a temporary token sent via a link, which allows them to select a time slot for an objection session.
- The admin is permitted to perform CRUD operations on all entities within the system. This includes modifying, adding, and deleting questions and reports and adding, deleting, or blocking users from the system.

- Instructors are allowed to view all questions in the database.
- TAs can only see the exams assigned to them and are not privileged to know all question databases.
- Only instructors are allowed to modify questions.

In the below figure, you can see the access scope of different user types:

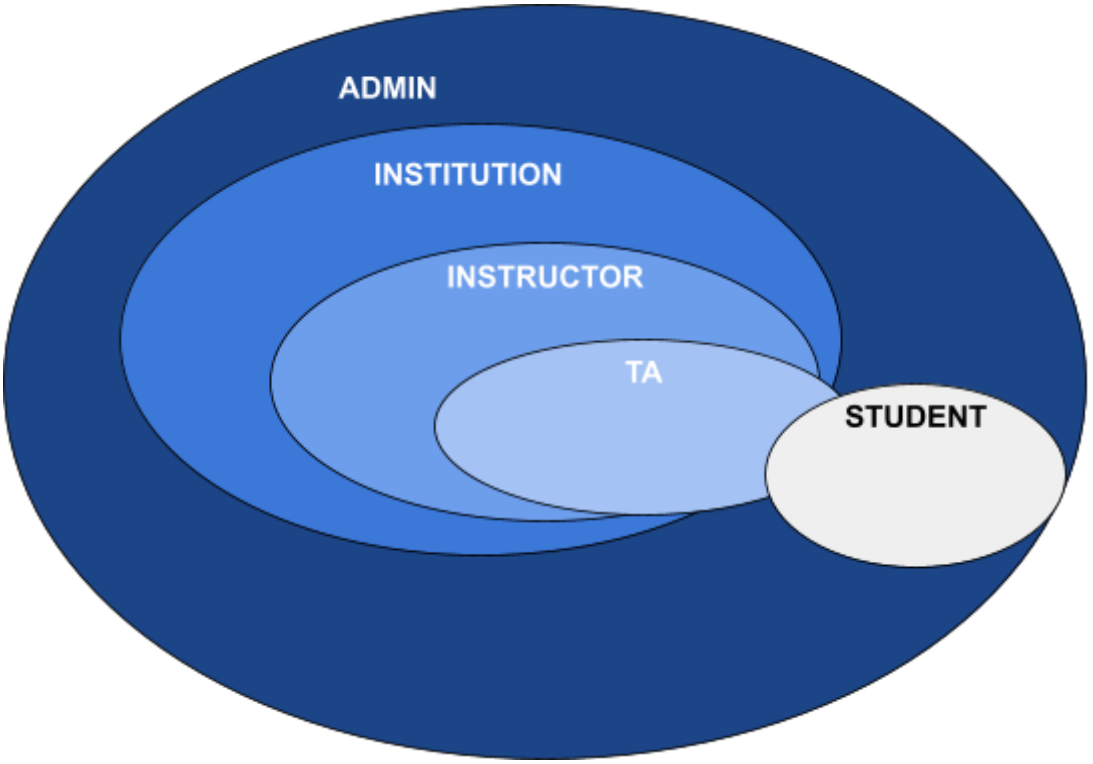

Figure 4: User access scope diagram

4. Subsystem Services

4.1 Client Subsystem

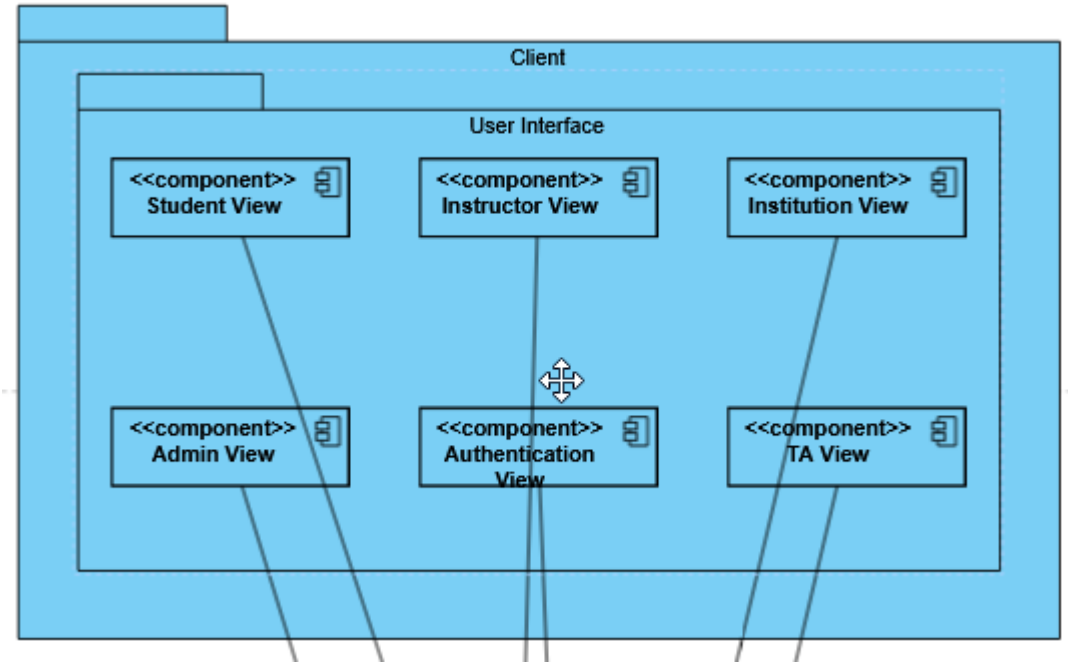

Figure 5: Client Side Subsystem

The Client Subsystem comprises multiple user interface components tailored to a specific user role. These components facilitate interactions with the backend services through REST APIs, ensuring seamless communication between users and the system's microservices

4.1.1 Student View

Students can view exam results, book objection sessions, and track performance. Provides secure login access with authentication verification. Enables students to review assigned exams and objection session schedules.

4.1.2 Instructor View

Provides instructors with tools for creating and managing exams. Allows access to the question database, analytics reports, and accreditation tools. It supports exam performance reviews and student success tracking.

4.1.3 Teaching Assistant (TA) View

Enables TAs to organize exam result viewing sessions and handle student objections. Provides access to the Portfolio Builder for accreditation documentation. Assists in exam evaluation and grading assistance.

4.1.4 Institution View

Designed for institution-level administrators to manage university-wide settings. Supports collaborative exam sharing among different departments. Allows administrators to oversee instructor and student access.

4.1.5 Admin View

Provides system-wide control over user access, institutions, and system configurations. Manages user registration requests, security settings, and role assignments. Ensures compliance with data security and system-wide policies.

4.1.6 Authentication View

Handles user login, password recovery, and authentication token validation. Implements Role-Based Access Control (RBAC) to restrict functionalities based on user roles. Supports OAuth 2.0, JWT authentication, and two-factor authentication (2FA) for enhanced security.

4.2 Gateway Subsystem

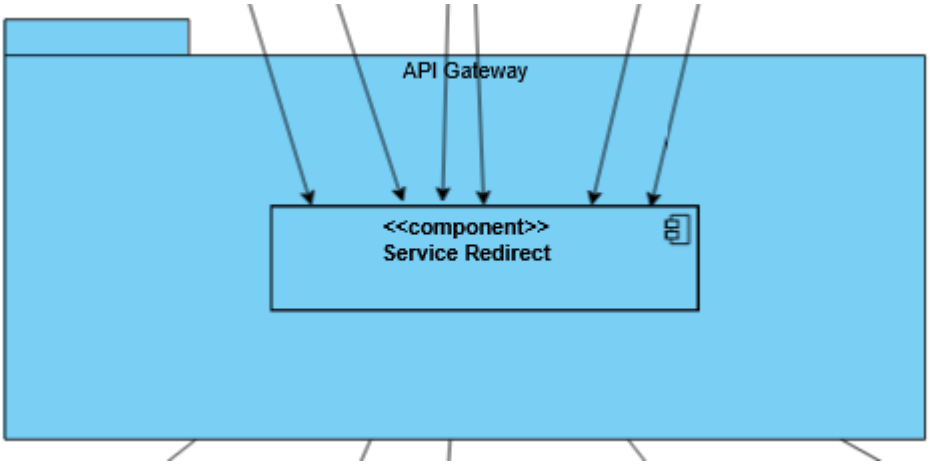

Figure 6: API Gateway Subsystem

The Gateway Subsystem in Evalio serves as the centralized entry point for all client requests, acting as a bridge between the frontend application and the backend microservices. This component ensures secure, efficient, and structured communication, enabling seamless data flow while maintaining authentication, load balancing, and request routing. The API Gateway handles incoming HTTP requests from the Client Subsystem and directs them to the appropriate microservices. It acts as a reverse proxy, ensuring optimized routing, authentication, and security enforcement.

The core component of this subsystem is Service Redirect, which is responsible for routing client requests to the corresponding backend microservice, ensuring security through JWT authentication and role-based access control (RBAC), and handling CORS (Cross-Origin Resource Sharing) policies for secure data exchange.

4.3 Logic Subsystem

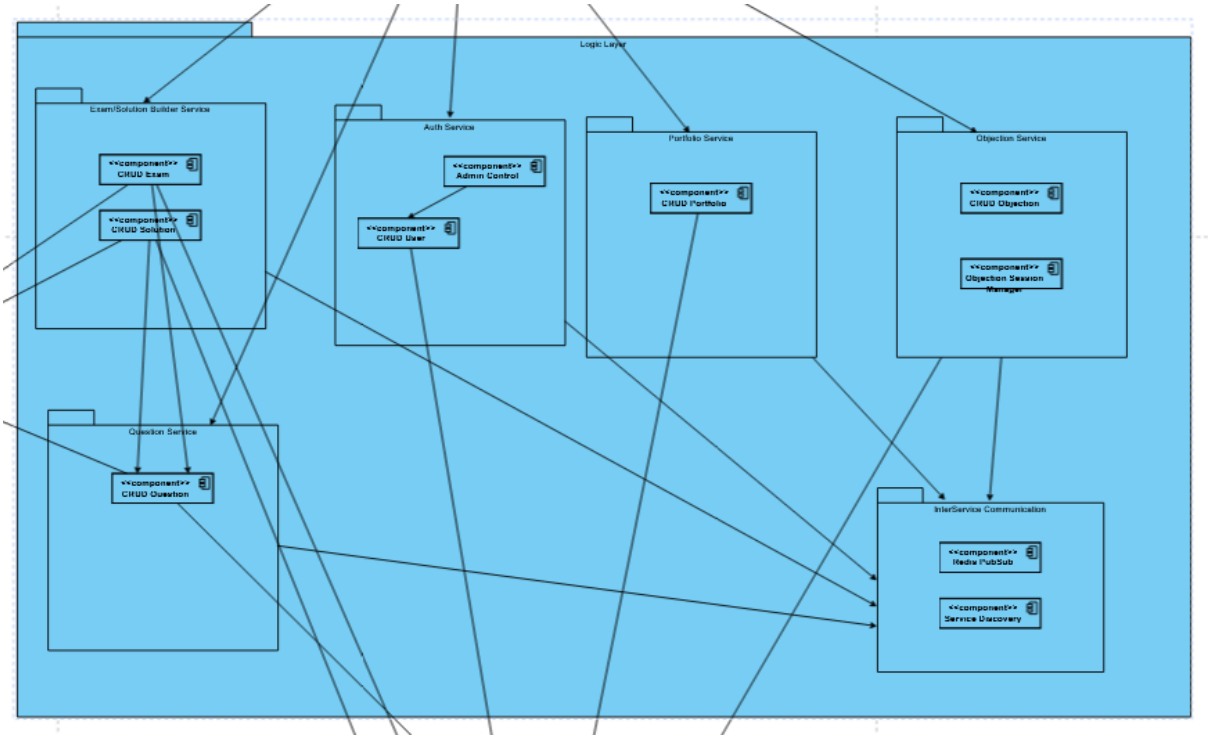

Figure 7: Logic Layer Subsystem

The Logic Subsystem of Evalio executes business logic, data processing, and system interactions across different microservices. This subsystem connects the API

Gateway to backend services, ensuring that all exam management, question processing, authentication, and result-handling operations are performed efficiently. The Logic Subsystem consists of multiple independent microservices, each responsible for a specific functionality. These services interact with each other through REST APIs and message queues, ensuring scalability and modularity. The system follows a microservices architecture, allowing individual services to operate independently and efficiently.

4.3.1 Exam & Solution Builder Service

Handles exam creation, solution generation, and LaTeX processing. Allows manual and automatic exam generation with balanced difficulty levels. Supports exporting exams to LaTeX, PDF, or Google Docs.

4.3.2 Question Service

Manages the question database, including classification, tagging, and version control. Provides APIs for retrieving, updating, and deleting questions. Supports metadata tracking, including difficulty levels and past usage statistics.

4.3.3 Authentication (Auth) Service

Manages user authentication and authorization using OAuth 2.0 and JWT. Implements Role-Based Access Control (RBAC) to ensure secure access management. Supports multi-institution authentication and user registration workflows.

4.3.4 Portfolio Service

Allows teaching assistants and instructors to create course portfolios for accreditation purposes. Stores and manages course materials, exam archives, and institutional reports. Supports exporting portfolios in PDF format for submission to accreditation bodies.

4.3.5 Objection Service

Manages exam result viewing sessions and student objections. Allows students to book, reschedule, or cancel objection slots. Provides APIs for tracking and resolving objections between students and instructors.

4.3.6 Interservice Communication Service

Handles notification services, including system alerts, emails, and announcements. Uses Redis Pub/Sub for event-driven messaging to notify users of exam updates or result availability. Supports real-time messaging for student-instructor communication.

4.4 External Services Subsystem

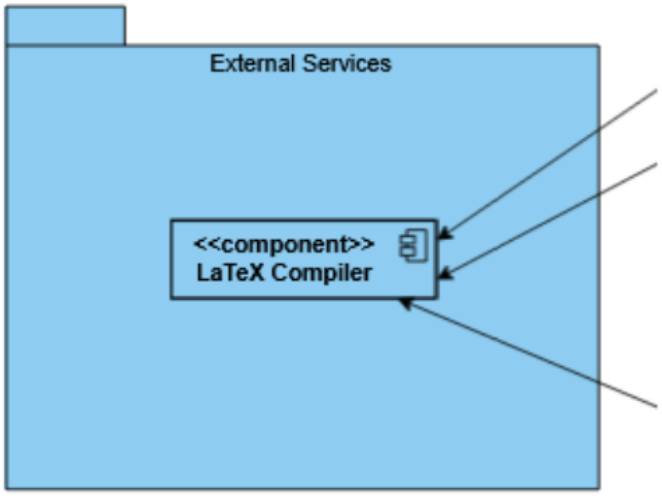

Figure 8: External Services' Subsystem

The External Services Subsystem in Evalio integrates third-party services and external tools necessary for specific functionalities, such as document compilation. One of the key components of this subsystem is the LaTeX Compiler Service which is responsible for rendering and compiling LaTeX documents into PDFs, used in the Exam Builder to generate formatted question papers and answer sheets. It Ensures high-quality, standardized exam layouts for instructors and utilizes TeXLive LaTeX distribution for compilation.

4.5 Data Management Subsystem

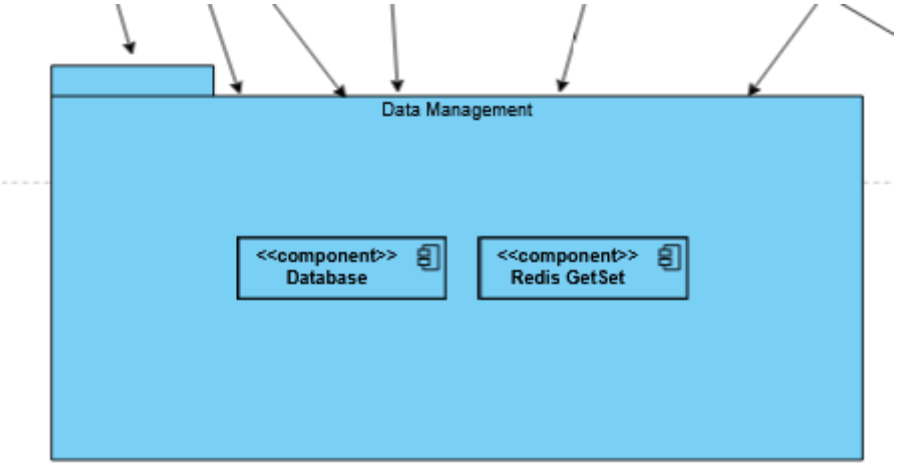

Figure 9: Data Management Subsystem

In the data layer, Evalio deals with two data types: persistent and non-persistent. Persistent data like user information, questions, and exam details are stored under the PostgreSQL database. Non-persistent data like 2FA tokens are stored under the Redis GetSet system.

4.6 AWS Subsystem

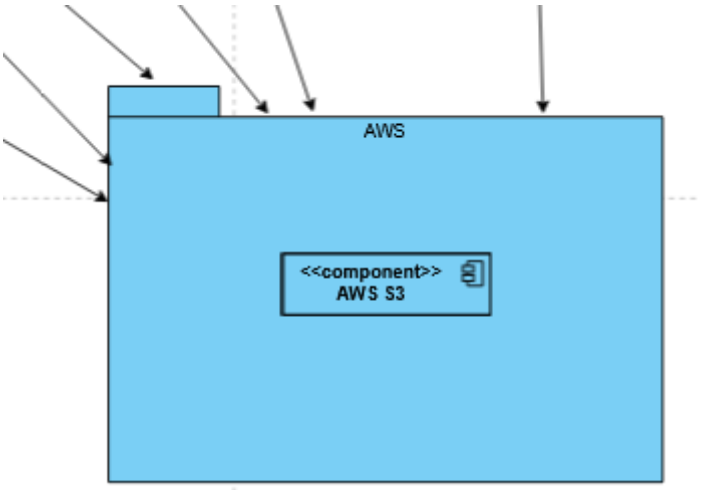

Figure 10: AWS Subsystem

Objects like question images, portfolio documents, and user profile photos are stored in AWS S3 data servers.

5. Test Cases

5.1 Functional Test Cases

5.1.1 Authentication Test Cases

Test ID:	FT001	**Category:**	Authentication	**Severity:**	Major
Objective	Verify that an institution can successfully register to the system.				
Steps	1. Open the application (Role: Institution). 2. Click on the register button. 3. Fill out the necessary information on the registration form. 4. Click the register button to complete the registration request. 5. Once the admin approves the registration request (at most in 2 days), go to the email in the registration form.				
Expected result	The system successfully registers a verified institution.				

Test ID:	FT002	**Category:**	Authentication	**Severity:**	Critical
Objective	Verify that other user types rather than institutions cannot register to the system.				
Steps	1. Open the application (Role: TA, Instructor, Student, or any other user). 2. Click on the register button. 3. Fill out the necessary information on the registration form. 4. Click the register button to complete the registration request. 5. Once the admin receives the request, it will reject registration. 6. The user will get an email about an unauthorized registration request.				
Expected result	The system successfully rejects any other user’s registration request.				

Test ID:	FT003	**Category:**	Authentication	**Severity:**	Critical

Objective	Verify that unregistered/unverified users cannot log in.
Steps	1. User clicks login. 2. Fill out the requested information on the login form. 3. User gets a message ‘there is no registered user with this email.’
Expected result	The system successfully prevents unregistered/unverified people from logging in the application.

Test ID:	FT004	**Category:**	Authentication	**Severity:**	Critical
Objective	Verify that registered/verified users can log in.				
Steps	1. Click login. 2. Fill out the requested information on the login form. 3. The user gets a message: ‘Your 2fa code is sent your email.’ 4. Check the email and copy the OTP code. 5. Return to the login page and enter OTP to verify 2fa. 6. Click the button to continue 7. You logged in.				
Expected result	The system successfully allows registered users to log in with 2-factor authentication.				

Test ID:	FT005	**Category:**	Authentication	**Severity:**	Critical

Objective	Verify that an institution can register multiple TAs/Instructors to the system.
Steps	1. Log in as an institution. 2. Go to the institution profile page. 3. Click on the register TA or Instructor button 4. Click “choose .csv file” 5. Choose the .csv file saved to the computer and already filled with the “name, email” of the respected people. 6. Click Choose. 7. Click the register button.
Expected result	The system successfully registers respective users to the system and sends emails to the users with their temporary passwords.

Test ID:	FT006	**Category:**	Authentication	**Severity:**	Critical
Objective	Verify that an institution can register multiple TAs/Instructors to the system.				
Steps	1. Log in as an institution. 2. Go to the institution profile page. 3. Click on the register TA or Instructor button 4. Fill out the form with the necessary information 5. Click the register button.				
Expected result	The system successfully registers the respective user with the system and sends an email with their temporary password to the user.				

5.1.2 Question Database Test Cases

Test ID:	FT007	**Category:**	Question DB	**Severity:**	Major

Objective	Verify that an instructor can create and add a question to the database.
Steps	1. Log in as an instructor 2. Navigate to the Question List Page 3. Click on “Add Question” 4. Enter valid question details. 5. Click "Submit."
Expected result	The system successfully saves the question in the database and displays a confirmation message.

Test ID:	FT008	**Category:**	Question DB	**Severity:**	Major
Objective	Ensure the system prevents question creation if required fields are missing.				
Steps	1. Log in as an instructor 2. Navigate to the Question List Page 3. Click on “Add Question” 4. Leave one or more blank question fields (e.g., question text or required space). 5. Click "Submit."				
Expected result	The system displays an error message indicating the required fields are missing. The question is not saved.				

Test ID:	FT009	**Category:**	Question DB	**Severity:**	Major

Objective	Ensure that an instructor can successfully edit and update an existing question in the database.
Steps	1. Log in as an instructor 2. Navigate to the Question List Page 3. Select an existing question from the list. 4. Click on the "Edit Question" button. 5. 5. Modify the **q**uestion text, metadata (tags, type, difficulty level), or LaTeX content with valid values. 5. Click "Save Changes."
Expected result	The system successfully updates the question and reflects the changes. A confirmation message is displayed after saving

Test ID:	FT010	**Category:**	Question DB	**Severity:**	Minor
Objective	Ensure that the question description field has a reasonable character limit.				
Steps	1. Log in as an instructor 2. Navigate to the Question List Page 3. Click on “Add Question” 4. Enter a very long question description exceeding the maximum allowed length. 5. Click "Submit."				
Expected result	The system prevents submission and displays a warning about exceeding character limits.				

Test ID:	FT011	**Category:**	Question DB	**Severity:**	Major

Objective	Ensure the system handles question search queries with partial keywords appropriately.
Steps	1. Log in as an instructor 2. Navigate to the Question List Page 3. Enter a partial keyword (e.g., "inte" instead of "integral"). 4. Click the “Search” button or enter the key. 5. Review the displayed results
Expected result	The system should return related results.

Test ID:	FT012	**Category:**	Question DB	**Severity:**	Major
Objective	Ensure that instructors can apply multiple filters (Difficulty Level + Topic + Institution) simultaneously for questions.				
Steps	1. Log in as an instructor 2. Navigate to the Question List Page 3. Select multiple filters (Difficulty Level: Hard, Topic: Probability, Institution: Bilkent University). 4. Click the Apply Filter button. 5. Review the displayed results				
Expected result	The system displays only questions that meet all selected filter criteria. If no matching questions exist, the system displays a "No results found" message.				

5.1.3 LaTeX Service Test Cases

Test ID:	FT013	**Category:**	LaTex	**Severity:**	Moderate

Objective	Ensure that LaTeX syntax is handled correctly in the question text.
Steps	1. Log in as an instructor 2. Navigate to the Question List Page 3. Click on “Add Question” 4. Open the latex editor on that page. 5. Enter a question with the valid LaTeX syntax. 6. Wait for a second for automatic rendering.
Expected result	The system correctly processes and shows the converted LaTex as a PDF without errors on the right side of the page.

Test ID:	FT014	**Category:**	LaTex	**Severity:**	Moderate
Objective	Ensure that incorrect LaTeX syntax is handled in the question text.				
Steps	1. Log in as an instructor 2. Navigate to the Question List Page 3. Click on “Add Question” 4. Open the latex editor on that page. 5. Enter a question with the invalid LaTeX syntax. 6. Wait for a second for automatic rendering.				
Expected result	The system does not convert LaTeX to a PDF, showing “LaTeX syntax is incorrect or missing.”				

Test ID:	FT015	**Category:**	LaTex	**Severity:**	Moderate
Objective	Ensure that the system allows users to import a LaTeX file, validate its content, and display it correctly in the LaTeX editor				

Steps	1. Log in as an instructor 2. Navigate to the Question List Page 3. Click on “Add Question” 4. Click on the "Import LaTeX File" option. 5. Select a .tex file with a valid LaTeX syntax from the local system and upload it. 6. Wait for a second for automatic rendering.
Expected result	The system successfully uploads and displays the content of the .tex file on the left side. Also, the preview renders correctly as a PDF on the right side.

Test ID:	FT016	**Category:**	LaTex	**Severity:**	Moderate
Objective	Ensure the system prevents users from importing LaTeX files with incorrect formatting.				
Steps	1. Log in as an instructor 2. Navigate to the Question List Page 3. Click on “Add Question” 4. Click on the "Import LaTeX File" option. 5. Attempt to upload a corrupted or incorrectly formatted .tex file. 6. Wait for a second for automatic rendering.				
Expected result	The system shows the LaTeX content in the editor but does not display rendered PDF. The system shows a validation error: “LaTeX syntax is incorrect or missing.”				

Test ID:	FT017	**Category:**	LaTex	**Severity:**	Critical
Objective	Ensure that the system prevents users from importing non-LaTeX files.				

Steps	1. Log in as an instructor 2. Navigate to the Question List Page 3. Click on “Add Question” 4. Click on the "Import LaTeX File" option. 5. Attempt to upload an unsupported file format (e.g., .docx, .pdf, .txt). 6. Wait for a second for automatic rendering.
Expected result	The system rejects unsupported file formats and displays an error message.

Test ID:	FT018	**Category:**	LaTex/PDF	**Severity:**	Critical
Objective	Ensure that the system allows instructors to export a question as a LaTeX or PDF file and that the exported content is correctly formatted.				
Steps	1. Log in as an instructor 2. Navigate to the Question List Page 3. Select an existing question from the list. 4. Click on the "Export" button. 5. Choose the desired format: LaTeX (.tex) or PDF (.pdf). 6. Click "Download" and save the exported file.				
Expected result	The system generates an adequately formatted LaTeX (.tex) or PDF (.pdf) file.				

5.1.4 Exam Creation Test Cases

Test ID:	FT019	**Category:**	Exam	**Severity:**	Critical
Objective	Ensure an instructor can successfully build an exam by selecting questions and configuring exam settings.				

Steps	1. Log in as an instructor 2. Navigate to the Profile page. 3. Click the “Create Exam” button. 4. Enter the valid values for the exam fields. 5. Click on the "Create" button. 6. Wait for the “Exam is successfully created” message. 5. Navigate to the Question List page 6. Select a question and click the “Add Question to Exam” button. 7. Click on the selected exam on the opened exams list. 8. Select another question and click the “Add Question to Exam” button. 9. Click on the selected exam on the opened exams list. 10. Navigate to the Profile page. 11. Select and navigate the created exam template in the list on the right side of the page.
Expected result	The system shows exam parameters on the right side with added questions. Also shows the created PDF with the combination of questions on the right side.

Test ID:	FT020	**Category:**	LaTex, Exam	**Severity:**	Major
Objective	Ensure an instructor can edit an existing exam, modify its LaTeX content, and save the changes.				

Steps	1. Log in as an instructor 2. Navigate to the Profile page. 3. Select an existing exam template from the exam list. 4. Click on the “Edit Exam” button. 5. Modify the LaTeX content of a question with valid syntax or exam instructions. 6. Wait for a second for automatic rendering. 7. Click on the “Save Changes” buttons.
Expected result	The exam updates successfully, and a confirmation message is displayed. The updated LaTeX content is correctly rendered and displayed.

Test ID:	FT021	**Category:**	Exam	**Severity:**	Critical
Objective	Ensure that an instructor can successfully export an exam as a PDF or LaTeX file.				
Steps	1. Log in as an instructor 2. Navigate to the Profile page. 3. Select an existing exam template from the exam list. 4. Click on the “Export Exam” button. 5. Choose the desired format: LaTeX (.tex) or PDF (.pdf). 6. Click Download and save the file.				
Expected result	The system successfully generates a LaTeX or PDF file. The exported file contains all exam details formatted correctly.				

5.1.5 Slot Test Cases

Test ID:	FT022	**Category:**	Slot	**Severity:**	Major
Objective	Verify that a TA/Instructor can create a new objection session using valid parameters.				

Steps	1. Log in as a TA/Instructor. 2. Navigate to the “Create Objection Session” page. 3. Fill in all required fields with valid data (startDatetime, endDatetime, slotDurationMin, etc.). 4. Click Create Session to submit.
Expected result	The system responds with a success message and returns the newly created objection session details (including generated slots). The session status is set to ACTIVE.

Test ID:	FT023	**Category:**	Slot	**Severity:**	Major
Objective	Verify that a TA/Instructor who owns the session can disable one or more slot(s) in an ongoing objection session.				
Steps	1. Log in as a TA/Instructor. 2. Open an ACTIVE objection session with multiple slots. 3. Select specific slots to disable. 4. Submit a DisableSlotsRequest with those slots.				
Expected result	The specified slots change status to “disabled,” and students can no longer reserve them. The system confirms with a success message.				

Test ID:	FT024	**Category:**	Slot	**Severity:**	Major
Objective	Ensure that the system appropriately handles invalid date/time inputs (e.g., end time before start time).				

Steps	1. Log in as a TA/Instructor. 2. Navigate to “Create Objection Session.” 3. Enter a startDatetime that is later than the endDatetime (e.g., start = 2025-03-07 14:00, end = 2025-03-05 13:00). 4. Click Create Session.
Expected result	The system rejects the request, returning an error message such as “Invalid time range.” No session is created.

Test ID:	FT025	**Category:**	Slot	**Severity:**	Major
Objective	Verify that a student can successfully reserve an available slot.				
Steps	1. A TA/Instructor has already created an objection session with multiple slots. 2. Access the link to the objection session (sessionPassword required if set). 3. Select an available slot from the session. 4. Click Reserve Slot.				
Expected result	The slot is marked as reserved under the student’s name, and other students can no longer select that same slot. A confirmation message (e.g., “Slot reserved successfully”) appears.				

Test ID:	FT026	**Category:**	Slot	**Severity:**	Moderate
Objective	Confirm that a student cannot overwrite or reserve a slot already taken by another student.				

Steps	1. A TA/Instructor has already created an objection session. 2. Access the link to the objection session (sessionPassword required if set). 3. Another student has already reserved a particular slot. 4. Navigate to the same objection session page as a different user than the user who already reserved a particular slot. 5. Attempt to reserve the already reserved slot.
Expected result	The system disables or grays out the taken slot and prevents selection. An alert or warning message appears (e.g., “This slot is already reserved”).

Test ID:	FT027	**Category:**	Slot	**Severity:**	Moderate
Objective	Verify that a TA/Instructor can close an objection session, preventing further slot reservations or modifications.				
Steps	1. Log in as a TA/Instructor. 2. Navigate to the existing objection session list. 3. Select an ACTIVE session. 4. Click the Close Session button. 5. Confirm the status change.				
Expected result	The objection session status changes to CLOSED, and students can no longer reserve, modify, or leave slots. Previously reserved slots remain visible for reference.				

Test ID:	FT028	**Category:**	Slot	**Severity:**	Moderate
Objective	Verify that a TA/Instructor can delete an objection session,				

	permanently removing all associated slots and reservations.
Steps	1. Log in as a TA/Instructor. 2. Navigate to the existing objection session list. 3. Select a session. 4. Click the Delete Session button. 5. Confirm the deletion.
Expected result	The system permanently deletes the session and all associated slot data. Students no longer see the session in the link.

Test ID:	FT029	**Category:**	Slot	**Severity:**	Critical
Objective	Verify that a student cannot reserve a slot that the TA/Instructor has disabled.				
Steps	1. A TA/Instructor creates an objection session. 2. Access the link to the objection session as Student (sessionPassword required if set). 3. Select a disabled slot from the session. 4. Click to reserve the slot.				
Expected result	The system blocks the selection of the disabled slot and displays an error message like: "This slot is disabled."				

Test ID:	FT030	**Category:**	Slot	**Severity:**	Major
Objective	Verify that a slot cannot accept reservations beyond its capacity.				

Steps	1. A TA/Instructor creates an objection session with slotCapacity = 3. 2. Three students reserve the slot. 3. A fourth student tries to reserve the same slot. 4. Click Reserve Slot.
Expected result	The system blocks the reservation and displays an error message: "This slot is full. Please select another available slot."

Test ID:	FT031	**Category:**	Slot	**Severity:**	Major
Objective	Verify that a student can change their reserved slot to another available one.				
Steps	1. A student enters the session and reserves a slot at 10:00 AM. 2. The student wants to choose a different time and tries to change it. 3. The student selects another available slot (e.g., 11:00 AM) and confirms the change. 4. Click Change Slot.				
Expected result	The system successfully updates the reservation to the new slot and releases the old one for other students.				

Test ID:	FT032	**Category:**	Slot	**Severity:**	Moderate

Objective	Verify that a TA/Instructor can export the details of an objection session as a PDF file.
Steps	1. Log in as a TA/Instructor. 2. Navigate to the objection session list. 3. Select a closed objection session. 4. Click on Export as PDF. 5. Verify that the downloaded PDF contains session details such as date, time, slots, and student reservations.
Expected result	The system successfully generates and downloads a PDF file containing session details.

Test ID:	FT033	**Category:**	Slot	**Severity:**	Major
Objective	Verify that multiple students attempting to reserve the same slot at the same time do not cause inconsistencies or double-booking issues.				
Steps	1. A TA/Instructor creates an objection session with multiple slots and capacity = 1. 2. Multiple students (e.g., 3 students) simultaneously attempt to reserve the same available slot using parallel requests (JMeter).				
Expected result	The system allows only one student to successfully reserve the slot. No duplicate bookings should be created in the database. The database maintains **data integrity** and does not allow conflicting reservations.				

5.1.6 Portfolio Builder Test Cases

Test ID:	FT034	**Category:**	Portfolio	**Severity:**	Moderate
Objective	Verify that the system can create a new course using valid parameters.				
Steps	1. Log in as an Instructor. 2. Navigate the portfolio main page 3. Click the “Add New Course” button 4. Fill all the required fields with valid data (course code, semester) in the opened modal. 5. Click the “Create” button to submit.				
Expected result	The system successfully responds with a success message and navigates the newly created course page.				

Test ID:	FT035	**Category:**	Portfolio	**Severity:**	Major
Objective	Verify that the user can import a document in the valid type to the system.				
Steps	1. Log in as an instructor. 2. Navigate the portfolio main page. 3. Click the course where the document will be imported. 4. Click the “Add File” button 5. Fill all the required fields (type, subtype) and optional fields (assessment date). 6. Click the “Upload a File” button. 7. Select a .pdf, or a .tex file from the local system and upload it. 8. Wait for a second for automatic rendering.				
Expected result	The system successfully responds with a success message and lists the newly created file in the portfolio list.				

Test ID:	FT036	**Category:**	Portfolio	**Severity:**	Major
Objective	Ensure that the system prevents users from importing non-Latex or non-PDF files.				
Steps	1. Log in as an instructor. 2. Navigate the portfolio main page. 3. Click the course where the document will be imported. 4. Click the “Add File” button 5. Fill all the required fields (type, subtype) and optional fields (assessment date). 6. Click the “Upload a File” button. 7. Attempt to upload an unsupported file format other than .pdf or .tex (e.g., docx, .txt). 8. Wait for a second for automatic rendering.				
Expected result	The system rejects unsupported file formats and displays an error message.				

Test ID:	FT037	**Category:**	Portfolio	**Severity:**	Major
Objective	Verify that the system can create a new assignment, portfolio type (e.g. quiz, exam).				
Steps	1. Log in as an instructor. 2. Navigate the portfolio main page. 3. Click the course where the new assignment type will be created. 4. Click the “Add File” button. 5. Click the “Select Type” bar and select “Add a new type”. 6. Type the new assignment type name in the textbox that opens. 7. Click the “Add” button.				
Expected result	The system successfully creates the new assignment type with a success message and lists the newly created type in the “Select Type” list in the “Upload New File” modal.				

Test ID:	FT038	**Category:**	Portfolio	**Severity:**	Moderate
Objective	Verify that the system can successfully edit and update an existing syllabus in the database.				
Steps	1. An instructor has already created a new course. 2. Navigate to the main page of the selected course. 3. Click the “Edit” button in the course summary. 4. Click the “+” button to add a new “assignment component”. 5. Fill all the required fields with valid data (number of assignments, grade contribution) in the opened modal. 6. Click the “Save” button.				
Expected result	The system successfully updates the syllabus and shows the changed syllabus in the summary section. A success message is displayed after saving.				

Test ID:	FT039	**Category:**	Portfolio	**Severity:**	Moderate
Objective	Ensure that an instructor can successfully create a merged portfolio of a course.				
Steps	1. Log in as an instructor. 2. Navigate the portfolio main page. 3. Click the course from where the portfolio will be created. 4. Click the “See Portfolio” button. 5. Select desired documents from the list. 6. Click the “Create Portfolio” button. 7. Click on the "Create" button from the information modal that opens.				
Expected result	The system successfully saves the merged document to the system. A success message is displayed after saving.				

Test ID:	FT040	**Category:**	Portfolio	**Severity:**	Minor
Objective	Ensure that an instructor can successfully export a portfolio of a course.				
Steps	1. Log in as an instructor. 2. Navigate the portfolio main page. 3. Click the course from where the portfolio will be created. 4. Click the “See Portfolio” button. 5. Select desired document(s) from the list. 6. Click the “Export” button.				
Expected result	The system successfully generates a portfolio file and exports it to the local system. A success message is displayed after exporting.				

Test ID:	FT041	**Category:**	Portfolio	**Severity:**	Critical
Objective	Verify that an instructor can delete a course, permanently removing all documents in the portfolio.				
Steps	1. An instructor has already created the course to remove. 2. Login as an instructor. 3. Navigate the portfolio main page 4. Click the “Delete” button from the action tab in the row of the course to remove. 5. Write “DELETE” into the textbox in the pop-up modal. 6. Click on the "delete" button, which will be clickable after entering the text.				
Expected result	The system permanently deletes the course information and all associated portfolio data. A success message is displayed after deleting.				

Test ID:	FT042	**Category:**	Portfolio	**Severity:**	Critical
Objective	Verify that an instructor can delete a document permanently.				
Steps	1. An instructor has already created the document to remove. 2. Log in as an instructor. 3. Navigate the portfolio main page. 4. Click the course from where the document will be removed. 5. Click the “See Portfolio” button. 6. Click the “Delete” button from the action tab in the row of the document to remove.				
Expected result	The system permanently deletes the document information. A success message is displayed after deleting.				

Test ID:	FT043	**Category:**	Portfolio	**Severity:**	Moderate
Objective	Verify that the system can successfully edit and update an existing assignment in a syllabus in the database.				
Steps	1. An instructor has already created a course and the course has at least one type of assignment in it. 2. Log in as an instructor. 3. Navigate to the main page of the selected course. 4. Click the “Edit” button in the course summary. 5. Fill all the desired fields with valid data (date, no. of assignment) in the desired type of assignment. 6.. Click the “Save” button.				
Expected result	The system successfully updates the assignment information and shows the changed information in the summary section. A success message is displayed after saving.				

Test ID:	FT044	**Category:**	Portfolio	**Severity:**	Moderate
Objective	Verify that the system can display an existing document in LaTeX format in the “Edit Exam” page.				
Steps	1. There is at least one .tex type of assignment in the course portfolio. 2. Log in as an instructor. 2. Navigate the portfolio main page. 3. Click the course where the assignment will be opened. 4. Click the “See Portfolio” button. 5. Click the “Edit” button from the action tab in the row of the document to open.				
Expected result	The system successfully navigates the user to the “Edit Exam” section. The LaTeX content is correctly rendered and displayed.				

Test ID:	FT045	**Category:**	Portfolio	**Severity:**	Minor
Objective	Verify that the system can successfully filter the portfolio according to the section.				
Steps	1. An instructor has already created a course and the course has at least one section. 2. Log in as an instructor. 3. Navigate the portfolio main page. 4. Click the “All Sections” button. 5. Choose the desired section among the selections.				
Expected result	The system successfully filters the portfolio page according to the section. A filtered page is displayed.				

5.2 Non-Functional Test Cases

Test ID:	NFT001	**Category**	Usability	**Severity**	Moderate
Objective	Verify that the UI of the software is self-explanatory so that users are not overwhelmed with excessive amounts of content at once.				
Steps	1. Create sample user groups. 2. Prepare appropriate demo sessions for each group and observe their usage behaviors. 3. Ensure they refer to the user manual and ask questions about the software. 4. Evaluate the outputs.				
Expected result	The user manual and UI should be adequate for users to perform core actions of the software. A moderate amount of questions to be asked is expected.				

Test ID:	NFT002	**Category**	Maintainability	**Severity**	Critical
Objective	Verify that the code base is self-documented, non-redundant, and embracing new features.				
Steps	1. Check for redundant code segments. 2. Check for code segments that do not conform to best practices. 3. Ask each developer to cross-check the code blocks that they have not developed. Ask them if those code segments are self-documenting. 4. Determine risky parts that may block the development in the future.				
Expected result	The amount of risky code segments is none or a few at least.				

Test ID:	NFT003	**Category**	Reliability	**Severity**	Critical
Objective	Verify that the software is invulnerable to single-point failure.				
Steps	1. Shut down services one at a time to identify possible single-point failures. 2. Observe the outcomes on the client side.				
Expected result	No or little impact on the client side should be observed. Failures must not cause any sensitive data to leak.				

Test ID:	NFT004	**Category**	Security	**Severity**	Critical
Objective	Verify that non-authorized entities are not allowed to access protected resources.				
Steps	1. Test the application with common attack vectors like: ● XSS ● XSRF ● UI Redressing ● MITM Attack ● SQL Injection ● Brute-Force Attacks				
Expected result	Application is invulnerable to each attack vector.				

Test ID:	NFT005	**Category:**	Security	**Severity:**	Critical
Objective	Verify that only authorized services can communicate with each other using Service-to-Service Authentication (S2S Auth).				
Steps	1. Identify a service that requires S2S authentication (e.g., Objection Service). 2. From an unauthorized service (e.g., Question Service), attempt to send an authenticated request to Objection Service. 3. Observe the system's response. 4. Repeat the request with a valid S2S authentication token and observe the response.				
Expected result	The unauthorized service receives a 401 Unauthorized or 403 Forbidden error with a message like "Service authentication failed"				

Test ID:	NFT006	**Category:**	Security	**Severity:**	Critical
Objective	Verify that users can only access endpoints permitted by their role, ensuring Role-Based Access Control (RBAC) enforcement through JWT authentication.				
Steps	1. Log in as an Institution role user and obtain a valid JWT token. 2. Make a POST request to the Instructor Register endpoint (/api/institution/register-instructors) using the Institution JWT token. 3. Observe the response and confirm that the request is successful. 4. Log in as an Instructor role user and obtain a valid JWT token. 5. Attempt to make the same POST request to /api/institution/register-instructors using the Instructor JWT token.				
Expected result	The Instructor role user receives a 403 Forbidden or 401 Unauthorized response with a message like "Permission denied"				

6. Consideration of Various Factors in Engineering Design

6.1 Constraints

6.1.1 Implementation Constraints

The essential implementation constraints in the project are ensuring secure authentication with Two-factor authentication (2FA), enforcing role-based access to protect exam materials, and implementing tools for accurate scanning and classification of diverse question formats.

6.1.1.1 Authentication

2FA should be considered to enhance security and prevent exam leaks. This additional layer of authentication is vital for protecting sensitive exam-related materials and maintaining the system's confidentiality.

6.1.1.2 Data Access

The system must enforce secure, role-based access to maintain academic integrity and restrict unauthorized access to exam-related materials. For instance, only instructors and authorized TAs should be able to view, edit, or archive exam questions.

6.1.1.3 Question Scanning

When uploading and scanning questions, the system must handle the complexity of extracting and organizing content from various input formats, including text, images, and scanned documents, to ensure accurate classification and usability of the questions in the database.

6.1.2 Economic Constraints

6.1.2.1 Cloud Service

Initially, we plan to run Evalio on a central server. Establishing and maintaining a server from scratch is a burden for our team. So, we decided to build

Evalio's server using a cloud service. Cloud services are the trend nowadays for many services with varying necessities and scales, and they are cost-efficient options for organizations like us [2]. The first economic issue with our application is the operation and possible scaling cost of the cloud service we will provide.

6.1.2.2 Advertisement

The economic issue is the advertisement of the Evalio. We need to introduce how we can improve the operations of universities relating to the examination procedure. Additionally, what we can do should be determined by the demands of the target group. Therefore, target group analysis should be done periodically, and a reasonable budget should be allocated.

6.1.3 Time Constraint

The development and deployment of Evalio are limited by the academic calendar and project deadlines. The system must be fully functional and ready for demonstration within the allocated time frame. This constraint requires efficient time management, prioritization of core features, and potential deferral of less critical functionalities to future phases. Additionally, periodic testing and feedback collection must be integrated into the timeline without disrupting the development schedule.

6.1.4 Professional and Ethical Issues

We need to consider the academic success of the universities to ensure a fair access policy. Evalio should expand its supported languages according to the institutions collaborating and provide reasonable suggested questions. Academic success in Evalio's context should not be a primary factor in the given service. Even if the Evalio is a for-profit application, it must not be allowed to undermine academic integrity.

6.2 Standards

6.2.1 Modeling Standards

UML 2.5 will be used for system modeling, including use cases, sequences, and class diagrams to represent system architecture and workflows [3].

6.2.2 Requirements Documentation

The IEEE 830 standard will guide the structure and format of requirements documentation, ensuring clarity and traceability of functional and non-functional requirements [4].

6.2.3 Security Standards

The project will align with ISO 27001 standards for information security management, ensuring that data confidentiality, integrity, and availability are maintained [5].

7. Teamwork Details

7.1 Ensuring Coordination and Efficient Workflow in the Project

Since our project is a large-scale project which consists of some interconnected modules, it is crucial for team members to work in a harmonious manner in most cases to ensure delivery in time and completely. In this context, we used Jira to coordinate effectively and track the progress of each team member. Jira is a project management tool that helps teams efficiently plan, monitor, release, and support high-quality software with confidence [6]. In Jira, it is possible to observe not only who is assigned to a task but also the deadlines and priority levels, which makes it an effective choice for the project.

Additionally, since the project consists of a lot of interrelated components and different team members need to work compatibility on these different parts, we placed significant importance on documentation. We prepared a comprehensive Google Docs file that serves as a single access point for both code-related manuals and other documents. This file included reports, developer manuals, diagrams, meeting dates, and minutes along with their respective links. Additionally, for quick communication, we used WhatsApp, and for planned meetings, we used Google Meet. As a shared workspace, we used GitHub for version control, Excalidraw and Visual Paradigm for diagram creation, Figma for designs, and Google Drive for file management. All these tools provided a common platform where all team members could track the progress of the project.

In addition to all these tools; to follow teamwork, we organized meetings to convey the progress of the projects in a more effective and more comprehensive way. Depending on the academic workload, these meetings were held between once every two weeks and up to three times within these two weeks. At times, they took place in person, while at other times, they were conducted online via Google Meet. In the early stages, these meetings focused on project planning and task distribution. In later phases, discussions centered around what had been completed, what needed to be done, upcoming challenges, support requests if necessary, and deliverables. To ensure that meeting progress was not forgotten and could be easily tracked later, meeting logs were kept. Through all these efforts, team progress and seamless teamwork were aimed throughout the process.

7.2 Contributing and functioning effectively on the team

As mentioned, Evalio is a project which consists of several different modules. Accordingly, we distributed the tasks among the team members. In the coding phase, all team members contributed to both backend and frontend development. Specifically, Eren H. Arım worked on exam creation and LaTeX processing, while Burak Demirel focused on microservice architecture setup and the question database services. Dilara Mandıracı and Yusuf Toraman handled slot creation, admin operations, and authentication. Ahmet Reşat Demir took part in the portfolio builder service.

For testing, each member was responsible for writing unit tests for the module they worked on. A more broader test was planned collectively with all group members. Everyone contributed to the project documentation and reports equally. The same principle of equality was maintained in the presentation and management processes. However, Yusuf Toraman took on more responsibility in task creation and Jira maintenance since he is more experienced.

7.3 Helping creating a collaborative and inclusive environment

In both coding and other processes such as documentation, specific task distribution was always maintained. However, team members supported each other when there was a need for assistance. Depending on the urgency or importance of the issue, one or more team members provided help when necessary. When there is

a problem, a task was created on Jira, or other team members were informed via WhatsApp based on the urgency. These collaborative efforts were crucial not only for those in need of assistance but also for team members to ensure the progress of the project. All team members were aware of that since a modular part of the project could impact the overall progress of the project, collaborative work and support were essential.

7.4 Taking lead role and sharing leadership on the team

Throughout the project, some tasks were assigned directly, while others were decided through discussions. For example, each team member was responsible for both the coding areas they led and the related parts in the reports.

As mentioned before, Evalio consists of many different modules. These modules were shared among the team members, and each member took leadership of their assigned part. Besides his own tasks, Burak set up the project's microarchitecture and led this process. He informed others about the architectural rules they needed to follow and helped them with their setups. Any changes in the architecture were made with his support. Meanwhile, since Eren and Dilara had taken a graphic design course, they led the UI design process along with their own tasks. They created a base template for the frontend, which other team members followed. Everyone writes their frontend code according to the rules set by these UI leaders. Similarly, Yusuf led the slot creation service. Since he also had experience in project management and Jira, he took the lead in maintaining Jira and managing the project workflow. Ahmet was responsible for leading the portfolio section. By using each team member's strong side and expertise, we made the project development process much easy-to-maintain and more efficient.

8. Glossary

Accreditation Reports: Reports generated to demonstrate compliance with educational standards like ABET and MÜDEK.

API Gateway: A central entry point for managing and routing client requests to backend microservices.

Authentication Service: A microservice responsible for verifying user identities and enforcing Role-Based Access Control (RBAC).

Cloud Storage (AWS S3): A cloud-based storage used for storing documents, images, and other large files in Evalio.

Exam Builder: A feature/module that allows instructors to create exams by selecting and organizing questions from the question database.

LaTeX Compiler: A tool integrated into Evalio for rendering LaTeX-based questions and generating formatted PDF exams.

Microservices Architecture: A software design approach where different services (e.g., question service, authentication service, exam service) operate independently.

Objection Session: A structured process in which students can schedule slots for their graded exams.

Portfolio Builder: A module that enables instructors to compile and manage course materials, and necessary documents.

Redis Cache: An in-memory caching system used to enhance performance by reducing database query times.

Role-Based Access Control (RBAC): A security mechanism that assigns different permissions to users based on their roles (e.g., Instructor, TA, Admin).

Two-Factor Authentication (2FA): An additional security layer requiring users to verify their identity through a secondary authentication method.

UML (Unified Modeling Language): A standardized visual representation method used to design system architecture and workflows.

User Roles: Different access levels assigned to users in Evalio, including Instructor, Teaching Assistant (TA), Institution, and System Admin.

9. References

[1] Purdue University, “Creating Exams,” Purdue Innovative Learning, 2024. [Online]. Available: https://www.purdue.edu/innovativelearning/teaching/module/creating-exams/. [Accessed: Nov. 21, 2024].

[2] Amazon Web Services, Inc., “What is a Cloud Server? - Cloud Servers Explained - AWS,” AWS Documentation, 2024. [Online]. Available: https://aws.amazon.com/what-is/cloud-server/. [Accessed: Nov. 21, 2024].

[3] OMG, “About the Unified Modeling Language Specification Version 2.5.1,” Object Management Group, 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1/About-UML. [Accessed: Nov. 21, 2024].

[4] Docsheets, “IEEE 830 Requirements Specifications,” *Top Requirements Management and Project Management Tools*, Nov. 2, 2023. [Online]. Available: https://www.docsheets.com/ieee-830-requirements-specifications-guide/. [Accessed: Nov. 20, 2024].

[5] ISO, “ISO/IEC 27000 family — Information security management,” International Organization for Standardization, 2024. [Online]. Available: https://www.iso.org/standard/iso-iec-27000-family. [Accessed: Nov. 21, 2024].

[6] Atlassian, “Get started with Jira - Comprehensive beginner's guide,” Atlassian, 2024. [Online]. Available: https://www.atlassian.com/software/jira/guides/getting-started/introduction. [Accessed: Mar. 08, 2025].