BILKENT UNIVERSITY

CS491 - PROJECT SPECIFICATION DOCUMENT

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..... 4

1.1 Description..... 4

1.2 High-Level System Architecture & Components of Proposed Solutions..... 5

1.2.1 Application Layer..... 5

1.2.2 API Gateway Layer..... 6

1.2.3. Microservices Layer..... 6

1.2.3.1 Exam/Solution Builder Service..... 6

1.2.3.2 Question Service..... 7

1.2.3.3 Auth Service..... 7

1.2.3.4 Course Portfolio Builder Service..... 8

1.2.3.5 Objection Service..... 8

1.2.3.6 Analytics Service..... 8

1.2.4. Infrastructure Layer..... 9

1.2.5. Data Layer..... 9

1.2.5.1 Database..... 9

1.2.5.2 Redis..... 9

1.2.5.3 Object Storage Service..... 10

1.3 Constraints..... 10

1.3.1 Implementation Constraints..... 10

1.3.1.1 Authentication..... 10

1.3.1.2 Data Access..... 10

1.3.1.3. Question Scanning..... 11

1.3.2 Economic Constraints..... 11

1.3.2.1 Cloud Service..... 11

1.3.2.2 Advertisement..... 11

1.3.3 Time Constraints..... 11

1.4 Professional and Ethical Issues..... 12
1.5 Standards..... 12
1.5.1 Modeling Standards..... 12
1.5.2 Requirements Documentation:..... 12
1.5.3 Security Standards:..... 13
2. Design Requirements..... 13
2.1 Functional Requirements..... 13
2.1.1 Questions Database and Exam Builder:..... 13
2.1.2 Analysis Tools..... 14
2.1.3 Portfolio Builder:..... 14
2.1.4 Exam Result Viewing Session Organizer:..... 15
2.2 Non-Functional Requirements..... 15
2.2.1 User Friendliness / Usability..... 15
2.2.2 Maintainability..... 16
2.2.3 Reliability..... 16
2.2.4 Security..... 16
3. Feasibility Discussion..... 16
3.1 Market and Competitive Analysis..... 16
3.2 Academic Analysis..... 17
3.2 Target User Analysis..... 18
4. Glossary..... 19
5. References..... 20

1. Introduction

1.1 Description

Academic institutions use various exam methods to evaluate student performance. The most common and traditional method is the "paper-based exam”. However, how accurate are these exams in measuring student success? How can we increase the effectiveness of assessment and evaluation? These questions led us to develop software that improves the examination procedure.

According to an article published by Purdue University, mapping questions and subjecting them to certain classifications and analyses before designing exams increases the efficiency and performance of exams. Using such mapping during the exam preparation phase provides significant convenience for exam creators (instructors) and higher-quality exams for students. Such mapping and analysis processes allow instructors to predict the exams they prepare [1].

Our team aims to design an application that helps instructors easily prepare for their exams and contribute to education and learning by assisting them in designing the most practical exams for their students and courses in light of these studies. Therefore, the Evalio exam database and management application provide modular exam creation and management for universities and institutions. It allows instructors to create exams and tests from a shared, searchable question database that supports multiple courses. The system tracks question usage, supports editing, versioning, and retirement, and offers export options such as LaTeX, Google Docs, and PDF. It provides detailed analytics on question performance to create balanced

assessments. The system can also manage assignments and exams, offering secure multi-university collaboration with authentication.

1.2 High-Level System Architecture & Components of Proposed Solutions

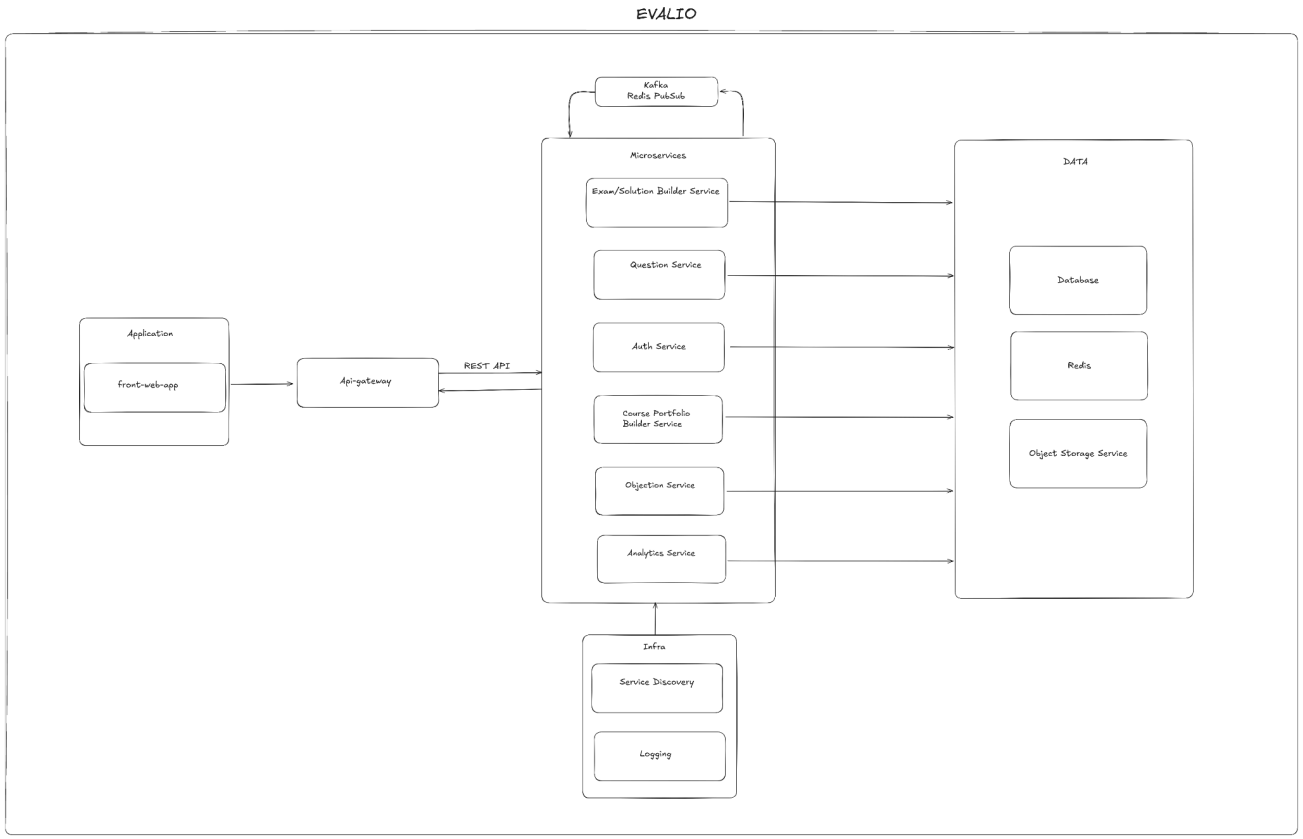

Figure 1: High-Level System Architecture Diagram

1.2.1 Application Layer

The Application Layer is the user-facing layer of the architecture. It includes a front-end web application that delivers a seamless and intuitive user experience. This layer manages interactions between the users and the backend services. It will be built using the React framework and implement features like exam-preparing interfaces, panels, and course portfolio management. The front end communicates

with the API Gateway through RESTful APIs to fetch or send data, and it typically includes mechanisms for handling authentication, session management, and content rendering.

1.2.2 API Gateway Layer

The API Gateway Layer is the single entry point for all client requests. It simplifies communication between the front end and microservices by routing requests to the appropriate backend services. The API Gateway provides features like authentication and authorization by validating tokens (e.g., JWT) to ensure secure access. It also supports load balancing to distribute traffic evenly across multiple instances of services and rate limiting to prevent abuse by throttling excessive requests. Additionally, it handles request and response transformations, enabling the front end to consume backend data in a simplified format.

1.2.3. Microservices Layer

The Microservices Layer is the core of the architecture, where business logic resides. Each service is designed to handle a specific functionality domain, enabling independent deployment and scalability. These microservices communicate asynchronously through event-driven mechanisms, using Kafka or Redis Pub/Sub to ensure loose coupling and scalability.

1.2.3.1 Exam/Solution Builder Service

The Exam/Solution Builder Service is the backbone of the system's examination functionality. It manages the entire lifecycle of exams, from creation to distribution. This service allows for the creation of various exams in custom formats. Additionally,

it integrates closely with the Question Service to fetch relevant questions dynamically. A critical feature of this service is generating solution keys. Its ability to handle user-specific configurations and progress tracking makes it adaptable to different use cases.

1.2.3.2 Question Service

The Question Service is a central repository for all questions in the system. It stores questions with detailed metadata, such as topic, difficulty level, and tags, to enable easy categorization. This service supports dynamic question fetching, ensuring that exams can be tailored to specific criteria or randomly generated based on parameters. With built-in functionality for versioning and updates, the service ensures the integrity of the question bank, even when questions are modified over time. It also integrates seamlessly with the Exam/Solution Builder Service, allowing exams to pull relevant questions.

1.2.3.3 Auth Service

The Auth Service provides authentication and authorization for the entire system. It ensures secure access by verifying user credentials and JWT tokens for session management. This service implements role-based access control (RBAC), allowing different user roles—such as students, teachers, and administrators—to access only the resources and features they can use. It might support integration with external authentication providers, enabling Single Sign-On (SSO) and third-party login options like Google or Facebook. The Auth Service maintains the system's security and protects sensitive data by handling API token verification for every request.

1.2.3.4 Course Portfolio Builder Service

The Course Portfolio Builder Service organizes and manages courses within the system. It allows teachers to structure their courses into portfolios, associating them with exams, study materials, attendance, and other learning resources. This service offers the design and customization of course portfolios, enabling educators to create course summary forms from available data. Users can browse and search for courses easily through the APIs provided by this service.

1.2.3.5 Objection Service

The Objection Service is critical in handling user feedback and disputes related to exams. It allows users to raise objections, such as reporting incorrect answers, unfair grading, or other issues they encounter. Teachers and administrators can review these objections through tools provided by the service, track their status, and resolve them effectively. This service enhances user satisfaction and trust in the system by offering a structured process for handling complaints.

1.2.3.6 Analytics Service

The Analytics Service is the system's data analysis and reporting tool, providing critical insights for teaching assistants and teachers. One of its key features is tracking and reporting the success rate per exam question, enabling educators to identify which questions were particularly challenging or accessible for students. This ability helps refine future assessments and improve the overall learning experience. Additionally, the service provides analysis tools to compare grades versus attendance, offering valuable insights into the correlation between class participation and academic performance. The service can visualize these data according to the

different parameters. For institutional reporting, the Analytics Service generates comprehensive reports for ABET and MÜDEK**,** ensuring compliance with accreditation standards. By providing insights and reporting, the Analytics Service will meet the institutional needs.

1.2.4. Infrastructure Layer

The Infrastructure Layer provides supporting services essential for the system's functioning. It includes Service Discovery, which allows microservices to locate and communicate with each other dynamically without hard-coded dependencies. This ensures high availability and flexibility. The Logging component is a centralized system aggregating logs from all services, making monitoring, debugging, and auditing the system easier.

1.2.5. Data Layer

The Data Layer ensures data persistence and storage for the application.

1.2.5.1 Database

Stores structured data, such as user profiles, exam details, questions, and course information. It uses relational or NoSQL databases, depending on the system's requirements.

1.2.5.2 Redis

A high-performance in-memory data store is used to cache frequently accessed data or manage sessions to improve response times and reduce database load. Also, Redis Pub/Sub might be used to stream messages between microservices.

1.2.5.3 Object Storage Service

This service handles unstructured data like files, images, and other media. It is optimized for large-scale storage and retrieval of objects, ensuring reliability and scalability for handling non-relational data (e.g. AWS S3)

1.3 Constraints

1.3.1 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.

1.3.1.1 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.

1.3.1.2 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.

1.3.1.3. 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.

1.3.2 Economic Constraints

1.3.2.1 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.

1.3.2.2 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.

1.3.3 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.

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.

1.5 Standards

1.5.1 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].

1.5.2 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].

1.5.3 Security Standards:

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

2. Design Requirements

2.1 Functional Requirements

The functional requirements of this project focus on creating a secure, efficient, and user-friendly system to manage question databases, build exams, analyze performance records, and organize exam result viewing sessions. Each module is designed to streamline workflows for instructors, teaching assistants, and students. The system also supports accreditation needs and inter-university collaboration.

2.1.1 Questions Database and Exam Builder:

- Maintain a question database for each course, categorized by type and topic classification.
- Instructors can add new questions to the database.
- Log the history of question usage, including statistical data.
- Allow the addition, editing, and deprecation of questions.
- Implement permission controls to restrict or allow access to specific course-related question pools.
- Provide an exam builder that supports the following:
    - Question selection and arrangement.

    - Editable space and points allocation for questions.
    - Header section customization (course name, date, name/ID fields, score table).
    - Export to LaTeX, Google Docs, or other visual formatting editors, with optional PDF export.
    - In-editor question editing and creation.
- Enable instructors to build modular exams by selecting questions from the database using filters such as difficulty level, topic, and type.
- Include functionality to auto-generate balanced exams based on predefined criteria, such as equal topic distribution or desired difficulty levels.

2.1.2 Analysis Tools

- University and course instructors can create custom reports based on selected metrics:
    - Correlation between attendance and academic performance.
    - Success rate per question.
    - Success rate comparison with other universities per question.
    - etc…
- University and course instructors can create ABET/MÜDEK reports according to global standards.

2.1.3 Portfolio Builder:

- Enable TAs to upload scans of selected exams (best, average, worst), quizzes, homework, projects, and exam seating plans.
- Upload attendance sheets and grading files.
- Generate course summary forms from uploaded and system data.

- Export a merged PDF containing all portfolio elements, including exam solution keys.

2.1.4 Exam Result Viewing Session Organizer:

- Enable teaching assistants to create exam result viewing sessions.
- Enable teaching assistants to create time slots for created sessions, which allows configurable numbers.
- Allow teaching assistants to cancel or rearrange sessions.
- Allow users to check schedules via the institution's local system to avoid conflicts.
- Allow students to access the session with the provided session key and password.
- Announce and manage session sign-ups on a first-come, first-served basis.

2.2 Non-Functional Requirements

2.2.1 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.

2.2.2 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.

2.2.3 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.

2.2.4 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.

3. Feasibility Discussion

3.1 Market and Competitive Analysis

We found some comparable tools after conducting research online to identify similar apps. There are several platforms where users can create mini-quizzes like Kahoot [6]; online assessments like HackerRank [7], or online exam tools like LearnUpon [8].

However, these tools are typically designed to create exams solved online through the app. We could not find a tool like ours that allows us to create exams specifically designed for PDF export and printed use. Our app, Evalio, aims to provide importable and printable versions of exam papers created through the platform.

Despite this distinction, Evalio shares a few features with these online exam platforms. For example, like many other platforms, it includes a question database from which users can select existing questions or upload new questions for future use. Additionally, these tools usually offer analytics, such as pass rates or average scores. However, Evalio aims to go beyond this by providing more detailed insights, such as the last asked date, semester, and course. Our focus on advanced analytics will differentiate us from our competitors.

Most of these apps require a paid subscription for users to create their exams. Our initial plan is to offer our users a free demo. We may align our pricing policy with our competitors depending on demand and positive user feedback.

The existence of numerous applications similar to our idea, even if not identical, proves a sufficient supply and demand for this type of application and its feasibility.

3.2 Academic Analysis

Numerous academic resources propose and explain systematic approaches to question databases and exam creation platforms. Khaled N. Elsayed has proposed a detailed system description using the Object-Oriented Knowledge-Based Question

Bank [9]. This paper can help us design our system and provide valuable insights if we add AI or intelligent agent support.

Additionally, we reviewed another article that is quite detailed and descriptive about an implementation design similar to our proposal. Sulieman Bani-Ahmad introduced a curriculum-focused, computer-based question bank software, X-Pro Milestone [10], to support partially computerized exams. This will serve as one of our primary academic resources, offering suggested tools and ideas to guide our implementation process. A particularly notable aspect of this paper is its focus on paper-based and online exam versions, which aligns with our goal of creating computer-generated, paper-based exam import files. X-Pro Milestone stands as an intense case study for our project.

3.3 Target User Analysis

This project idea emerged through discussions and brainstorming with our advisor, Can Alkan. Based on our meetings with our advisor, many faculty members at the school expressed their demand for such a platform. The scope of the project and the added modules were determined with the input of these faculty members, and the Evalio development team will refine and implement these boundaries more clearly. While the initial target user group for our product is CS faculty members at our university, we plan to expand support to other departments and institutions in the future.

4. Glossary

2FA: Two-factor authentication

ABET: Accreditation Board for Engineering and Technology, Inc.

API: Application Programming Interface

IEEE: Institute of Electrical and Electronics Engineers

ISO: International Organization for Standardization

JWT: JSON Web Token

MÜDEK: Mühendislik Eğitim Programları Değerlendirme ve Akreditasyon Derneği

RBAC: Role-based access control

REST: Representational State Transfer

SSO: Single Sign-On

UML: Unified Modeling Language

5. References

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

[2] “What is a Cloud Server? - Cloud Servers Explained - AWS,” *Amazon Web Services, Inc.* https://aws.amazon.com/what-is/cloud-server/

[3] “About the Unified Modeling Language Specification Version 2.5.1,” *www.omg.org*. https://www.omg.org/spec/UML/2.5.1/About-UML

[4] “IEEE 830 Requirements Specifications,” *Top Requirements Management and Project Management Tools*, Nov. 02, 2023. https://www.docsheets.com/ieee-830-requirements-specifications-guide/ [accessed Nov. 21, 2024].

[5] ISO, “ISO - ISO/IEC 27000 family — Information security management,” *ISO*. https://www.iso.org/standard/iso-iec-27000-family

[6] Kahoot!, "Make learning awesome!," [Online]. Available: https://kahoot.com/. [Accessed: Nov. 21, 2024].

[7] HackerRank, "Technical Interview and Assessment Platform," [Online]. Available: https://www.hackerrank.com/. [Accessed: Nov. 21, 2024].

[8] LearnUpon, "Exams: Create an exam," [Online]. Available: https://support.learnupon.com/hc/en-us/articles/360004738778-Exams-create-an-exam. [Accessed: Nov. 21, 2024].

[9] K. N. Elsayed, "A Tool for Creating Exams Automatically from an Object-Oriented Knowledge Base Question Bank," *International Journal of Information and Education Technology*, vol. 3, no. 1, pp. 41-45, Feb. 2013. [Online]. Available: https://www.ijiet.org/papers/228-T1002.pdf

[10] S. Bani-Ahmad, "Toward Developing a Syllabus-Oriented Computer-Based Question-Banks Software to Support Partially Computerized Exams," *Journal of Software Engineering and Applications*, vol. 8, no. 5, pp. 279–285, May 2015. DOI: 10.4236/jsea.2015.85026. [Online]. Available: https://www.scirp.org/journal/paperinformation?paperid=56550. [Accessed: Nov. 21, 2024].