T.C.

BAHÇEŞEHİR UNIVERSITY

FACULTY OF ENGINEERING AND NATURAL SCIENCES

CAPSTONE PROJECT PROPOSAL

BAU-EVAL: LLM-Based Personalized Assignment Evaluator #1

Süleyman Kaan Ataç

Beyza Bayrak

Atena Jafari Parsa

Aleyna Kurt

Advisors:

Dr. Binnur Kurt - Department of Artificial Intelligence Engineering

Dr. Fatih Kahraman - Department of Artificial Intelligence Engineering

ISTANBUL, December 2024

STUDENT DECLARATION

By submitting this report, as partial fulfillment of the requirements of the Capstone course, the students promise on penalty of failure of the course that

- they have given credit to and declared (by citation), any work that is not their own (e.g. parts of the report that is copied/pasted from the Internet, design or construction performed by another person, etc.);
- they have not received unpermitted aid for the project design, construction, report or presentation;
- they have not falsely assigned credit for work to another student in the group, and not take credit for work done by another student in the group.

Key Words: LLM, Backend, Docker Frontend, Llama, React.js, Next.js, cloud services, Assignment Evaluator, web based personalised examination panel, Real-time assessment

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

LLM	Large Language Models
NLP	Natural Language Processing
IEEE	The Institute of Electrical and Electronics Engineers

1. OVERVIEW

The use of large language models (LLMs) for automatic scoring has become an important evaluation method in NLP research. In this project, a cutting-edge LLM-based exam and homework checking and evaluation/scoring system will be developed. The LLM-based platform will be used for students taking programming, machine learning/artificial intelligence, etc. courses at Bahçeşehir University to achieve the specified skill set and the proficiency. The objectives are to develop a platform that creates specific questions/quizzes for students, conducts the exam specifically for the student within the specified time, and records the evaluation result. The platform to be developed has the following 3 main components. These are, specialised LLM model for course content and related functionalities, Web-based personalised examination panel and Cloud-native backend software/services.

Süleyman Kaan Ataç, Beyza Bayrak, Atena Jafari Parsa and Aleyna Kurt from the artificial intelligence department will be working on this project for both LLM subsystem and full stack subsystem.

1.1. Identification of the need

Timely and fair evaluation of students' homework and exams is very important for educational institutions such as Bahçeşehir University. Assessments to measure students' skills, knowledge and proficiency in various courses are generally carried out in the form of manual scoring or general multiple-choice tests. These types of evaluation methods lack the advantage of student-specific learning, consume too much time, cause problems in giving timely feedback, and can lead to unfair evaluations.

Classical evaluation methods, especially for instructors of courses with a large number of students, create difficulties in conveying these evaluations to students individually, even though the instructor can make evaluations in a timely manner. Because of this problem, students have problems learning about their shortcomings and mistakes in their exams and assignments and have difficulty improving themselves in areas where they are inadequate.

This project aims to solve these problems for machine learning, programming, artificial intelligence and similar courses by developing an LLM-based platform and providing a new solution for evaluating students and giving feedback.

Students taking relevant courses at Bahçeşehir University and instructors of these courses constitute the target user audience of this platform

Students will benefit from this platform with assignments and exams specially created for them, their evaluations and the feedback they will receive. It will guide students in their learning journey by providing real-time feedback to students, learning about their deficiencies in relevant courses and addressing these deficiencies without wasting time. Thus, students' performance and proficiency in their courses will be increased.

Lecturers of relevant courses will also benefit from this platform, which automatically manages, evaluates exams and assignments and observes students. Thanks to automatic evaluations, human-induced bias problems will also be eliminated. This will prevent any problems regarding grades between the student and the instructor. In addition, lecturers will not waste time preparing and evaluating the questions themselves, as the system has manual evaluation processes. Thus, lecturers will be able to save their time and focus more on their students.

In addition to these academic impacts, the system to be developed will also have significant technological effects. The use of LLM is a big step forward on the modern education system. In addition, thanks to the cloud-based platform architecture, it is aimed for many users to use the system without any problems. This is especially useful to meet the needs of the increasing number of students.

The potential for further development of Bahçeşehir University's reputation is also among the effects of this project at the institutional level. Thanks to this state-of-the-art, artificial intelligence-based platform, Bahçeşehir University can take itself to a significant point in educational innovation. The system can also lead to cost efficiencies over time, reducing the need for manual grading and administrative efforts associated with exams and reallocating resources to areas that will directly benefit student learning.

1.2. Definition of the problem

There are functional and performance requirements and constraints that need to be considered in order to develop this project in accordance with its purpose.

The platform should be able to create special exams and assignments for students and evaluate them in real time. In addition, it should provide tools for educators to control these exams and assignments and audit the evaluations made by artificial intelligence.

The platform must be highly reliable, minimal latency and scalable. The process of creating

questions and evaluating students' answers should occur within seconds and be highly accurate.
The platform must adhere to data privacy rules, university guidelines, and the budget and resources provided.

1.2.1. Functional requirements

Question Generation for Homework and Exam Generation:

- Preparing specific questions for students based on the course dynamically selected using LLM.
- Determining parameters such as time and number of questions for the exams and assignments to be created.

Evaluation and Feedback:

- The platform should be able to automatically evaluate and score exams and assignments using LLM.
- The platform should be able to provide detailed feedback to students about exams.

Managing the Evaluations:

- Instructors should monitor student performance through a dashboard.
- The evaluations made should be explained by the LLM and the instructors should be able to supervise these evaluations and manage the scores of the students.

1.2.2. Performance requirements

Accuracy

- Question generation and evaluation should be with at least 90% accuracy.

Time

- Questions should be generated in 5 seconds per question.
- Evaluation results should be provided within 5 seconds for most question types, with coding challenges taking up to 10 seconds.

Scalability

- System should handle up to 1.000 concurrent users with low latency.

Availability

- Platform required to maintain 99.9% uptime during operational hours.

1.2.3. Constraints

Scheduling: Our team consists of 4 artificial intelligence engineering students. Our team members are people with experience in various fields of LLM. We also have sufficient knowledge in Backend and Frontend. Thus, with the support of our advisor, our members

have sufficient requirements for the project.

We have 14 weeks to complete the project. If the results from the project's performance are not as planned, additional time will be spent to improve performance. This may limit the progress of the project.

Costs: We have a limited budget, so there may be deficiencies in some purchasing parts.

Standards: We aim to build reliable, fair and consistent systems with following ethical guidelines and standards. The system will be user-friendly and accessible with protecting user data.

Other constraints: Number of data and quality of the data may cause undesired performance of the model because of limited computational resources. Also, feedback system integration may delay the progress.

1.3. Conceptual solutions

Using Large Language Model (LLM) based evaluation systems for students are becoming more popular and crucial because of their performance with real-world data. Since these systems can be customized, they meet the crucial and specific needs of education. They have ability to generate questions to each student, evaluate answers and provide feedback with detecting student's needs in real-time.

In this project, we will explore various conceptual solutions for developing BAU-EVAL: LLM-Based Personalized Assignment Evaluator. This project consists of a specialized LLM system, web-based integration for students/instructors with the system and backend service. We will also focus on designing a fairness and transparent system by taking factors such as cost, complexity and performance into consideration.

1.3.1. Literature Review

Before continuing with our conceptual solutions, we want to share our findings about related works. It is important to examine related works to understand how our system will be different or which parts will be similar.

The paper *"Large Language Model as an Assignment Evaluator: Insights, Feedback, and Challenges in a 1000+ Student Course"* by Chiang et al. (2024) explains the use of LLMs for automatic evaluation. This paper highlights important points to be developed because of their instruction-following inability and vulnerability to prompt hacking [1].

The paper *"LLM-EVAL: Unified Multi-Dimensional Automatic Evaluation for Open-Domain Conversations with Large Language Models"* by Lin and Chen (2024) proposes a multidimensional automatic evaluation approach for open-domain discussions with large LLMs. To reduce expenses and time-consuming work, they designed a method called single prompt based evaluation. The paper also shows the importance of choosing a proper LLM model after performance tests on various benchmark datasets [2].

The paper *"Aligning with Human Judgement: The Role of Pairwise Preference in Large Language Model Evaluators"* by Liu et al. (2024) explains the study about how current LLM evaluators do not match with human evaluations. They highlight that today's calibration approaches for minimizing LLM biases are insufficient. To handle this issue, they introduce the Pairwise-preference Search (PAIRS) method inspired by Reinforcement Learning from Human Feedback (RLHF) [3].

The paper by Hirunyasiri, Thomas, Lin, Koedinger, and Aleven (2024), titled *Comparative Analysis of GPT-4 and Human Graders in Evaluating Praise Given to Students in Synthetic Dialogues*, evaluates ability of GPT-4 model on providing feedback by analyzing 30 dialogues in a tutor-student setting. Their aim is to evaluate GPT-4's accuracy in identifying each praise criterion by using mainly two methods which are the zero-shot chain of thought and the few-shot chain of thought [4].

The paper "PRE: A Peer Review Based Large Language Model Evaluator" by Chu et al. (2024), proposes a new system inspired by academic peer reviews to address fairness and scalability problems. Since current methods can be expensive, biased and have low ability of generalization, they focus on providing a more personalized learning experience by doing a small qualification exam to select reviewers [5].

1.3.2. Concepts

For this project, we mainly focus on five conceptual solutions and we analyzed them in terms of cost, complexity, performance and features as shown in the table below. In each concept, we use different approaches to build an LLM-based evaluator and we aim to design a fast, fair and accurate system for users.

Concept 1: Using ready-to-use LLM

Ready-to-use LLMs can be used in various fields and with different purposes. Using ready-to-use models can be not suitable for adding complex features. It can lead to undesired performance levels because of the efficiency of them.

Concept 2: Open-source LLM

Since open-source model's code is publicly available, we can edit these models with no cost. This concept is easy to implement, flexible and customizable but generally these models are not the best option for high performance because of the model's complexity and architecture.

Concept 3: Closed-source LLM

Closed-source LLM model's architecture, training data and algorithms are generally not reachable. They mostly offer high security levels under controlled limitations. These models have licensing fees and extra cost for usage but compared to open-source models, they have advanced features and capabilities but they may have less customized options.

Concept 4: Hybrid System with Feedback System

In the hybrid system, responses are analyzed by LLM and feedback which is given by the instructors. With the feedback system, we can build a more accurate and reliable system. This concept is effective for optimizing the learning process of students and making the system powerful in terms of performance. Designing hybrid systems has balanced complexity architecture and advanced features with less costs.

Concept 5: Designing Specialized LLM

A specialized LLM system refers to specifically trained models with deeper knowledge. In this concept, we are able to get higher accuracy and a more reliable system with different features. We are able to add improved features or abilities with small pricing. It requires more technical knowledge to build the system. Also, implementation is not as easy as other options to reach optimized performance.

Table 1 compares different conceptual solutions with respect to the four most important requirements; Concept 4 is chosen for this project due to its low cost and reasonable score.

Table 1. Comparison of the three conceptual solutions.

	Concept 1	Concept 2	Concept 3	Concept 4	Concept 5
Cost	low	low	high	low	medium
Complexity	low	medium	medium	high	high
Performance	low	low	high	high	high
Features	low	low	high	medium	high

1.4. Physical architecture

In order to show the physical architecture of BAU-EVAL with an interface (system) diagram, the architecture is divided into two subsystems; Full Stack Development subsystem and LLM subsystem. The Full Stack Development subsystem consists of two main components; Frontend and Backend. Firstly, the input would be retrieved by the front end. The input could be the submitted answers by the students, or, question generation topics and student details queries from the user. After the input information is retrieved, it will be redirected to the backend through the frontend controller. The backend will structure the received information in proper format and store the necessary parts in the database, provide the keywords and required information for the Prompt Engineering Controller in the LLM subsystem.

In the LLM subsystem, after creating the relevant prompts by usage of the Prompt Engineering module, the prompts will go through Input Preprocessing. This module will clean and format the input prompts and also includes other transformations such as tokenization and embedding. Thus, the LLM module will handle generating questions, evaluating student answers and giving feedback processes by using the output from the input preprocessing module.

In the Post Processing module, the outputted raw text data will be structured and formatted by the backend for questions and feedback. Additionally, in this module, metadata such as timestamps and student metrics will be provided. The Fine Tuning module will then be used to adjust the LLM model and improve its performance by receiving updated course materials and more data as inputs.

As the preferred concept is Hybrid System with Feedback Mechanism, instructors will examine the evaluation and feedbacks of the LLM model to students' answers and give feedback to improve the model performance.

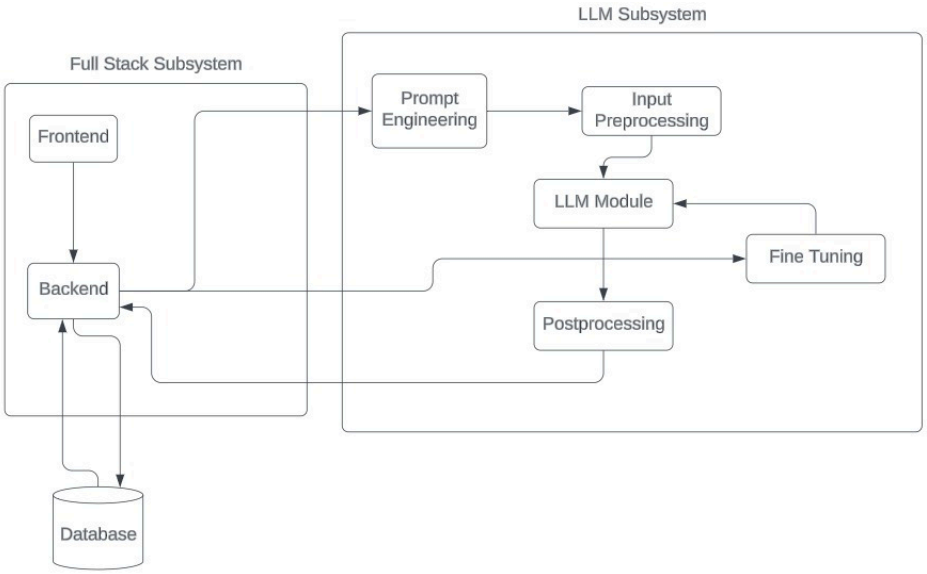


Figure 1. Interface diagram for the system

The flowchart diagram starts with the initial user interaction. Then, similar to the interface diagram, the front end will prompt the user for input, whether it is answer submission by the student or queries by either the instructor or the student. The backend will then check for validity of the input, if it is invalid, a proper error message will be generated by the backend and shown to the user and will automatically go back to input retrieval step. If the input is valid, as checked by the backend, the program will proceed to the next step which is extracting and structuring the information present in the input by the backend for storage in the database. The keywords (extracted by backend) will be provided for prompt engineering as well as the database itself for the purpose of generating precise and relevant prompts. The prompts will then be directly sent to the LLM and the responses of the LLM will be retrieved. After the retrieval of the responses generated by the LLM, the response will be evaluated by the instructor, if the generated response is acceptable, the response will be outputted to the user and also sent to the backend for storage. Else, the prompt engineering process will be repeated. Then, the program will end until the next user interaction.

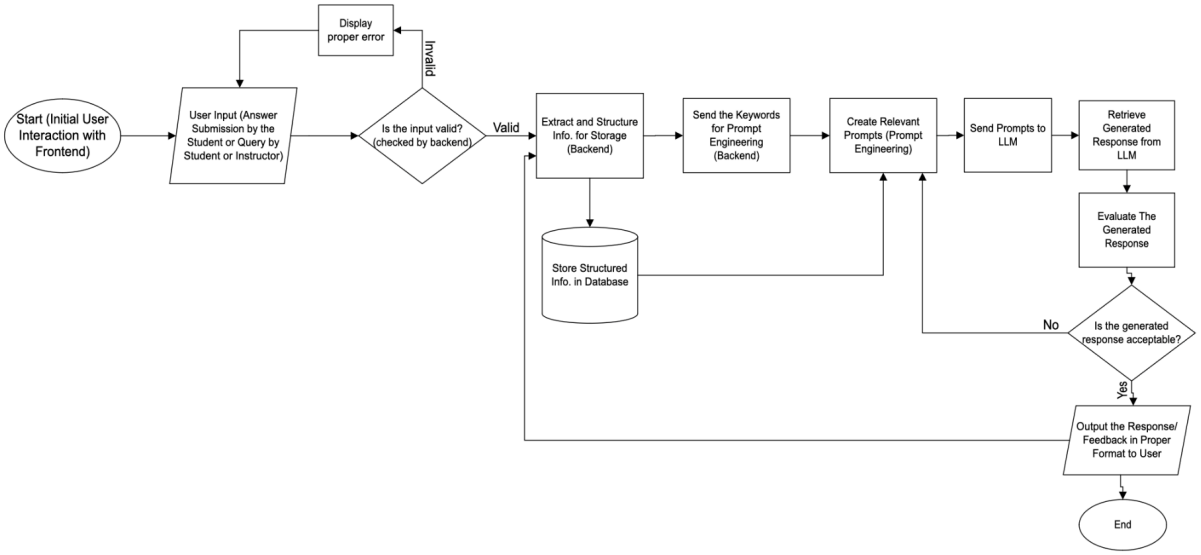


Figure 2. Process chart for the system

2. WORK PLAN

The work plan will start with Work Breakdown Structure (WBS), which basically uses a divide and conquer structure to break down all the work that needs to be done in order to build the project and puts it in a clear format. Then, there is the responsibility matrix which shows every group member's divided share of work required to materialize the project. Finally, the project network explores the divided work and responsibility of the group members in a visual graph for simplicity and understandability of tasks.

2.1. Work Breakdown Structure (WBS)

The Work Breakdown Structure (WBS), uses a divide and conquer structure to visually break down every work that needs to be done in order to materialize the project. It uses a general-to-specific (inverted triangle) method and order is important. Hence, the components are numbered carefully. First, as the most general module of the project, there is the project itself (LLM-Based Personalized Assignment Evaluator). Then, the work is broken down to six main components, starting with Planning. To do proper planning for the project, firstly clear and time-sensitive objectives should be set. Moreover, resource management must be considered. The second component is Data Preparation. This component consists of Data Collection and Data Cleaning (to prevent possible problems that might later rise because of unclean data). The third component is The Development which is divided into two main components; LLM Development and Full Stack Development. The LLM Development requires Model Training, and then, Model Testing and Validation ensuring the model functions properly. The Full Stack Development component necessitates first Backend and

then Frontend Development. These two components can also be developed simultaneously (in parallel) as they are deeply integrated in full stack development. The fourth main component is Integration. For this module, first the systems have to be configured correctly, then the integration must be tested to reassure the system's proper functionality. The 5th component is Platform Verification which requires these five modules in the right order; Functional Testing (using the Functional Requirements stated in 1.2.1. in the proposal), Performance Testing (utilizing the Performance Requirements stated in 1.2.2. in the proposal), Bug Reporting and Resolution, and lastly, Pre-Deployment Review. The last remaining main component in the WBS, is Reporting which refers to precisely documenting every progress in conducting the project and every possible setback, challenge or constraints revealed as the work is progressing.

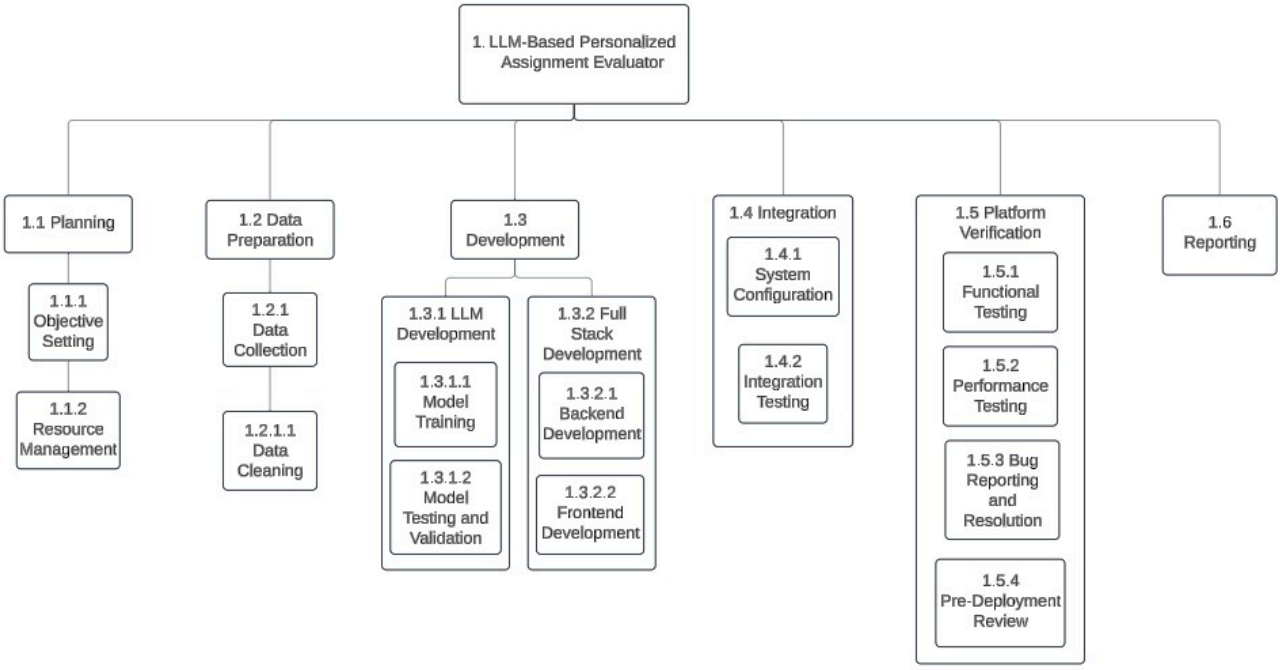


Figure 3.Work Breakdown structure for the project

2.2. Responsibility Matrix (RM)

As all of the students in this project are artificial intelligence students, everyone is expected to highly contribute to the LLM subsystem part of the project. All students will be working with harmony to solve the problems faced when developing the project.

Table 2. Responsibility Matrix for the team

Task	Aleyna Kurt	Atena Jafari Parsa	Beyza Bayrak	Süleyman Kaan Ataç
Planning	Supporter	Supporter		**Responsible**
Data Preparation	**Responsible**	Supporter	Supporter	Supporter
LLM Subsystem	**Responsible**	Supporter	Supporter	Supporter
Backend	Supporter	**Responsible**	Supporter	Supporter
Frontend	Supporter	Supporter	**Responsible**	Supporter
Integration	Supporter		Supporter	**Responsible**
Verification		Supporter	**Responsible**	Supporter
Reporting	Supporter	**Responsible**	Supporter	

2.3. Project Network (PN)

The Project Network (PN), refers to the breakdown of the responsibilities that need to be executed in a visual graph format for the purpose of simplicity. The project begins with planning and data preparation steps. Objectives of the project will be discussed and the necessary resources for the project will be managed. In the same time period, the data preparation phase will be handled. At first, necessary data will be collected from various sources and the collected data will be cleaned for better LLM model performance.

Then LLM development is going to be initiated. Prepared data is going to be used to train the LLM model. After the training process, the trained model will be tested to analyze how well it performs. Both backend and frontend systems are going to be developed in parallel to ensure compatibility between both systems. The project then moves into the integration phase. In this stage, all subsystems will be integrated with each other. System configurations will be handled and then integration tests should ensure seamless interaction between all parts of the system. After that, functional and performance requirements will be verified during the platform verification stage to confirm the platform operates as intended. If all the requirements are met, the project will be concluded with the reporting and documentation stage.

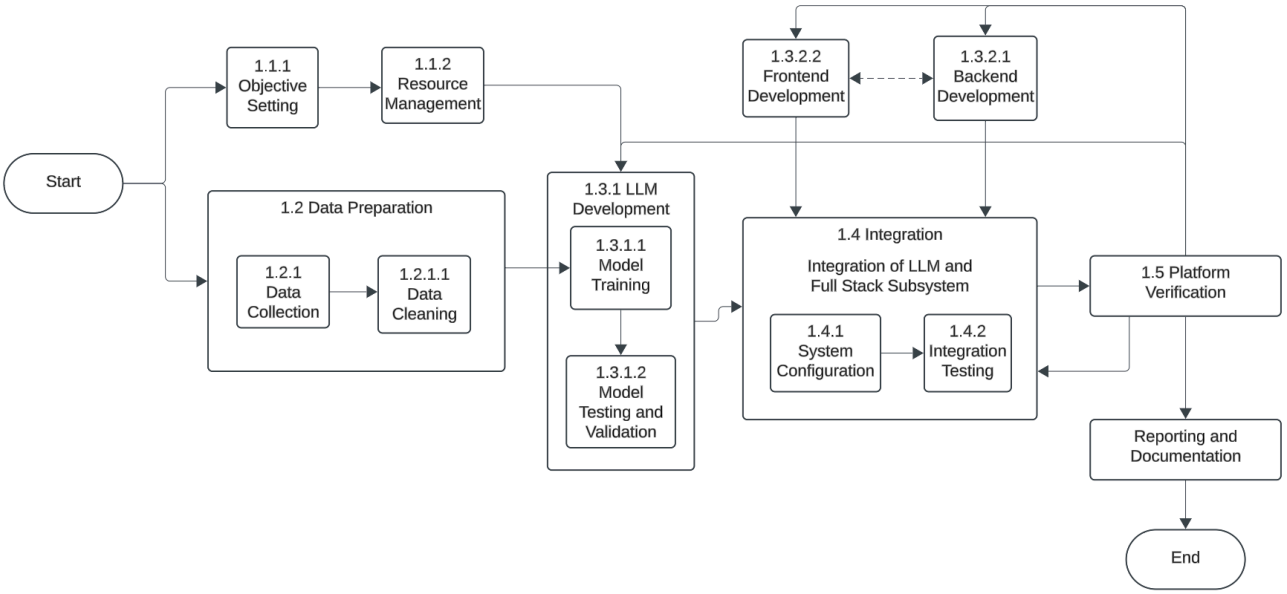


Figure 4. The project network

2.4. Gantt chart

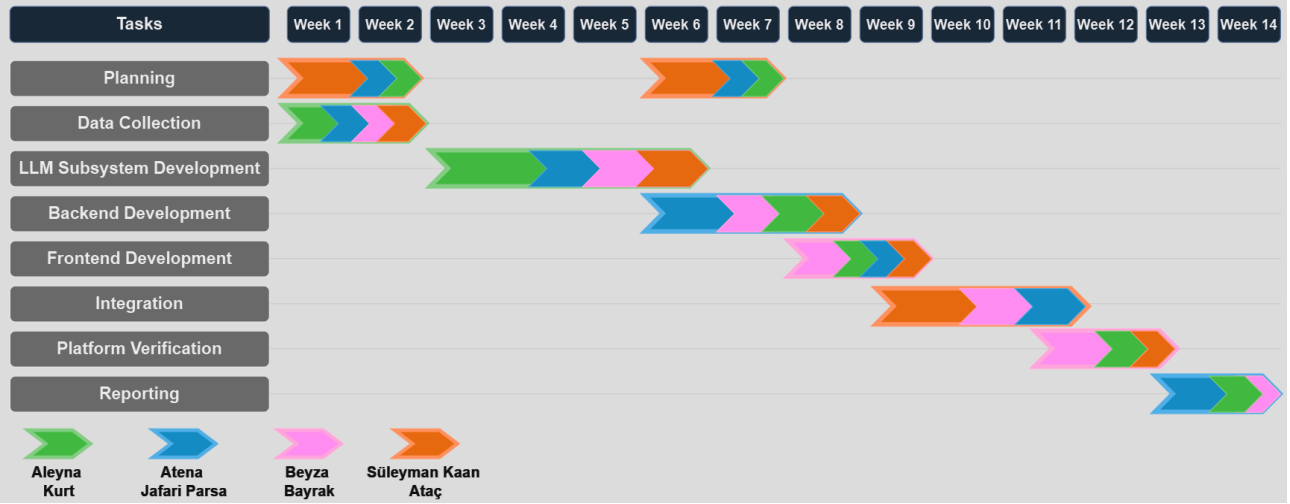


Table 3. Gantt chart for the materialisation phase of the project

2.5. Costs

In this project, we consider both software and hardware infrastructure in terms of tools and services necessary for the development and deployment process. Cloud services and microservices are the main parts of the project. To host and manage the process Cloud services are essential. Microservices are the best option for handling backend functionalities. We have to consider minor expenses that can occur for advanced or specialized wanted features. To protect our system for any cyber attacks or protecting user data, implementing authentication protocols should be added to the system with small costs or can be added via free options. Tools for Frontend and Backend are mostly free but advanced and customized features can be added with small expenses.

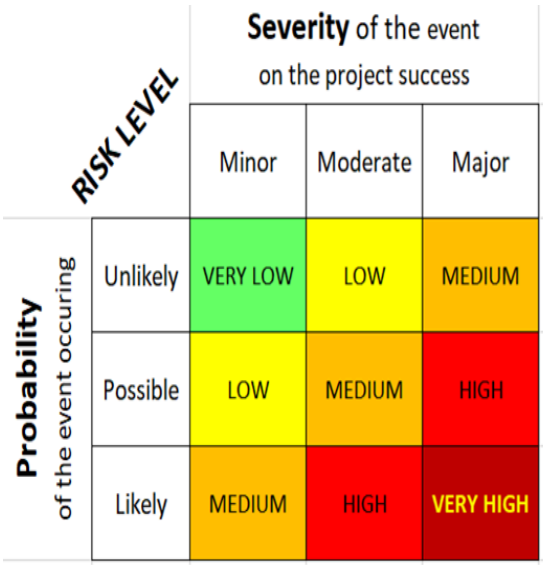
Table 4. Costs

Software and Hardware Infrastructure	
Cloud services	750₺ - 2000₺
Micro-services	0₺ - 500₺
Security Tools	0₺ - 300₺
Frontend Tools and Backend Framework	0₺ - 400₺
Total	**750₺ - 3200₺**

2.6. Risk assessment

We consider potential risks and categorize them into different levels to highlight their impact. Since each risk has different outcomes, we make a plan of action for every risk to minimize their effects.

Table 5. Risk Matrix



VERY LOW	**GPU Shortage**
LOW	**Frontend Problems**
MEDIUM	**Server Errors, Slow System Responses**
HIGH	**Inaccurate Question/Answer Generation**
VERY HIGH	**Scalability Problems**

Table 6. Risk Assessment

Failure Event	Probability	Severity	Risk Level	Plan of action
GPU Shortage	**Unlikely**	**Major**	**VERY LOW**	**Consider Cloud Storage options likeAWS or Azure**
Frontend Problems	**Unlikely**	**Moderate**	**LOW**	**Use modern frameworks**
Server Errors, Slow System Responses	**Likely**	**Moderate**	**MEDIUM**	**Optimize the model and monitor possible bottlenecks**
Inaccurate Question/Answer Generation	**Likely**	**Major**	**HIGH**	**Fine-tune the model by using feedback from users**
Scalability Problems	**Unlikely**	**Major**	**VERY HIGH**	**Use microservices or use containerization tools like Docker**

3. SUB-SYSTEMS

This project's goals are to create customized test questions using the LLM-based platform, schedule individual student exams within a predetermined window of time and facilitate learning by rapidly and equitably evaluating the outcomes.

Students enrolled in programming, machine learning/artificial intelligence, and other courses at Bahçeşehir University will use the LLM-based platform to acquire the required skill set. The three primary components of the platform that will be developed are as follows. These are - Specialized LLM model for course content and related functionalities - Web-based personalized examination panel - Cloud-native backend software/services**.**

3.1. LLM and Conversational AI GUIs AI Integration

In this module, a customized language model that is suited to the subject matter of pertinent courses is developed. In accordance with the course material on programming and artificial intelligence, the LLM will be trained to create questions and quizzes and assess student answers. It will be able to

evaluate open-ended submissions as well as open-ended responses. As new course material is added, the model will be updated and improved continuously to guarantee fair and accurate evaluations.

3.1.1. Requirements

Development of a specialized LLM model customized for course content

Extensions intended to be used:

- LLM: Llama 3, LangChain (or similar LLM frameworks)
- Conversational AI GUIs AI Integration: Hugging Face API, TensorFlow/PyTorch

3.1.2. Technologies and methods

Methods to develop the specialized LLM model will be used to generate both open-ended and multiple-choice questions specific to the course, evaluate students' answers, and provide feedback. It is planned to use Llama for the LLM part and PyTorch for the AI Integration part in this project. Additionally, prompt engineering will be applied in this subsystem. It is used to create clear, well-structured prompts to direct the model to produce precise, targeted and context-sensitive responses.

3.1.3. Conceptualization

For this project, we mainly focused on two potential solutions which are “Hybrid System with Feedback Mechanism” and “Designing a Specialized LLM”. Both concepts have different priorities and methods to build substems. They have different pros and cons in terms of complexity, performance and pricing. As we mentioned before, “Hybrid System with Feedback Mechanism” is the best suitable and optimal solution for this project. Hybrid systems with Feedbacks have more powerful generation talent since the system improves itself. It is more reliable because it is not fully automated and is controlled by instructors. Because of the architectural complexity, implementation can be challenging rather than using a Specialized LLM. We can build a more customized system with a Specialized LLM with high cost but due to its continuous updates, the system's backend can be affected negatively. In short, we choose a hybrid system as the most suitable solution but in case the hybrid does not work , we will plan to continue with a Specialized LLM concept.

3.1.4. Physical architecture

The LLM subsystem will focus on question generation and evaluation using Hybrid LLM system with Feedback Mechanism. There will be functional modules to construct the LLM subsystem to make it work for its purposes. Firstly, an input preprocessing module is needed. This module will clean and format the input prompts. Then tokenization, embedding and other needed transformations will be done. The Core LLM module will handle question generation, evaluation and giving feedback processes by using the output from the input preprocessing module. Then the post processing module will use raw outputs from the LLM module to structure text data for questions and feedbacks. It should also provide metadata such as timestamps and performance metrics for student evaluations. Then a fine-tuning module will be used to adjust the LLM model and increase its performance by inputting updated course materials and more data. To achieve the hybrid feedback mechanism, instructors will examine the evaluation and feedbacks of the LLM model to students' answers and give feedback to improve the model performance.

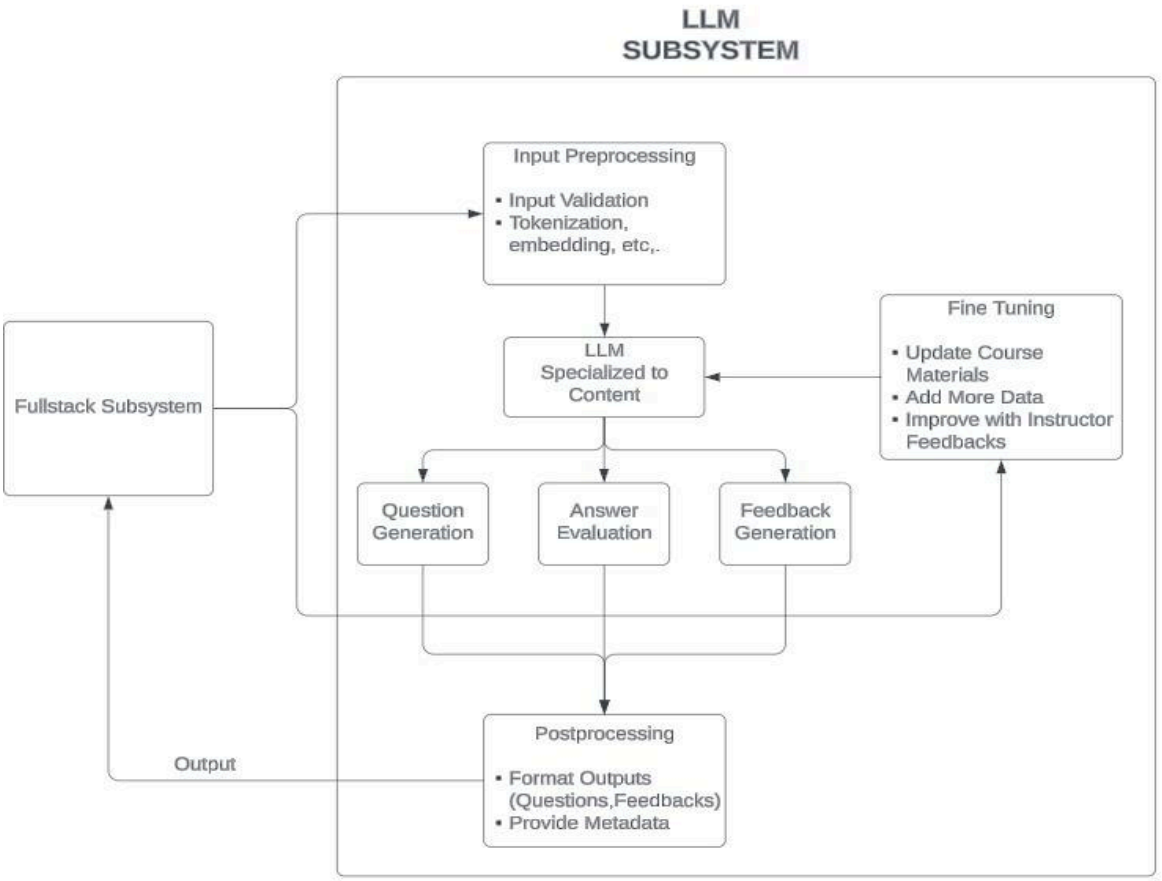


Table.7 LLM Subsystem Architecture

3.1.5. Materialization

Computing resources for model training and deployment, datasets specially selected for Bahçeşehir University's programming and AI syllabus, and appropriate software frameworks like PyTorch or Llma are among the materials required to realize the LLM module. Open-source repositories, university partnerships, and discussions with academic advisors will all be used to acquire these components.

3.1.6. Evaluation

Evaluations based on the model's performance and outputs must be carried out prior to integrating the LLM module with other subsystems. Making sure the LLM satisfies the requirements of creating course-specific questions, accurately evaluating answers, and offering impartial and consistent evaluations is the aim of this assessment.

3.2. Backend and Docker Frontend

In this module, a web-based panel for online exam administration will be created. Within a predetermined time frame, the panel will create customized tests for every student.

It will keep track of student inputs, exam lengths, and submissions, ensuring that students have a smooth exam experience. It will also have an administrative interface that will allow teachers to oversee tests and monitor results.

3.2.1. Requirements

Creation of a web-based, personalized exam management panel - Development of cloud-based, scalable backend software and services - Real-time assessment and recording of student performance - Tools for instructors to manage and monitor exam results.

Extensions intended to be used:

- Backend: FastAPI, PostgreSQL, AWS Lambda (or Azure Functions)
- Docker Frontend: React.js/Next.js, Tailwind CSS

3.2.2. Technologies and methods

Together, the Frontend and Backend subsystems offer a customized, web-based testing environment. Data integrity and security are guaranteed by these tools and techniques.

It is planned to use FastAPI, a modern web framework for building APIs with Python, for the Backend part and React, a JavaScript library, and Next.js, which offers advanced features for SEO performance, to create the user interface during the web development phase for the Docker Frontend part in this project.

Also, the platform to be used for the cloud based part will be determined according to the needs during the creation of the project.

3.2.3. Conceptualization

Our aim is to make the system consistent, customized and reliable. Since we plan to implement a hybrid system, we will explain it. Hybrid System combines Backend APIs like FastAPI with Frontend interfaces like React.js or Next.js. Hybrid systems allow faster implementation and integration with scalable solutions.

Our second option which is building Specialized LLM, needs more Backend and Frontend developments which leads to higher technical complexity and higher costs.

In short, building a hybrid system with feedback is more practical for Backend and Docker Frontend. But as we mentioned in section 3.1.3., in case a hybrid system can not fulfill the requirements of the system or any error occurs, we will continue to work with Specialized LLM.

3.2.4. Physical architecture

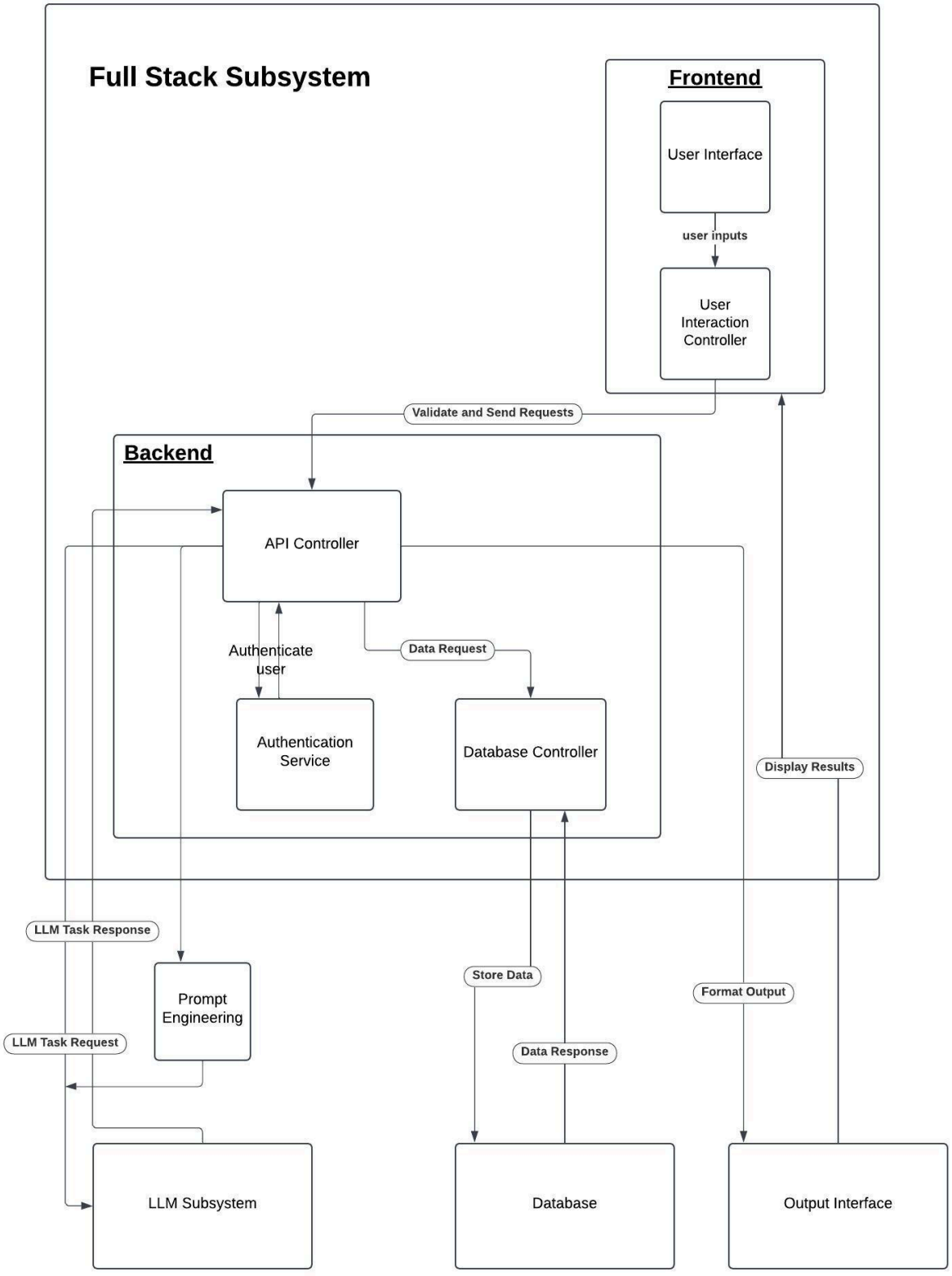


Table.8 Backend and Frontend Subsystem Architecture

3.2.5. Materialization

Cloud infrastructure such as AWS or Azure to host the data to be used for Back-End and Front-End development, and FastAPI for web framework for building APIs with Python are among the materials to be used. The cloud system to be used will be selected after calculating the budget required for these components. The university opportunities and the information we received from our academic advisors will be useful for the following stages.

3.2.6. Evaluation

Backend and frontend development will require some testing to evaluate system performance and user experience. Conditions such as the system's response time, simplification of user navigation on the site, interface consistency, and ability to continue working efficiently in high traffic conditions will be checked. Additionally, data regarding errors encountered during this control will be collected and necessary adjustments will be made.

4. INTEGRATION AND EVALUATION

4.1. Integration

The project architecture consists of two main subsystems, as shown in the system interface diagram (Section 1.4): the Full Stack Development subsystem and the Large Language Model (LLM) subsystem. The Full Stack subsystem includes Frontend and Backend components that manage data entry, processing and storage, connected by a Database. The LLM subsystem consists of various modules such as Prompt Engineering, Input Preprocessing, LLM Module, Postprocessing and Fine Tuning, each of which specializes in improving the interactivity and output quality of the system.
Frontend component, where user inputs, such as student submissions or query topics, are initially collected. This data is then transferred to the Backend where it is formatted and partially stored in the Database. Most importantly, the Backend also provides the necessary keywords and information to the LLM subsystem's Prompts Engineering module. Within the LLM subsystem, integration flows from Prompt Engineering, where prompts are created, to the Input Preprocessing module, which cleans and structures these prompts for optimal LLM performance. The LLM Module then processes these inputs to create educational content or feedback, which are then refined in the Post-Processing module for final output presentation.

As shown in the project network (Section 2.3), and the Gantt Chart (Section 2.4), before developing these subsystems planning and data preparation processes will be done in the first 2 weeks to be able to work on the subsystems properly. LLM subsystem is planned to be developed in 4 weeks, Full Stack subsystem is planned to be developed in 4-5 weeks. After developing the subsystems, they are going to be integrated with each other to achieve the functional and performance requirements in the integration part which is planned to be completed in 3 weeks. Then platform verification will be done in 2-3 weeks to observe the performance of the platform. Finally, we will prepare the final report to explain the details of the project.

4.2. Evaluation

In order to evaluate the proposal, it is necessary to address whether the functional and performance requirements have been fulfilled. The functional requirements, as stated in Functional Requirements (Section 1.2.1), include question generation for homework and exam generation, which refers to 1. Preparing personalized questions for students based on the selected course using LLM, 2. Finding the parameters that are needed for creating the corresponding exams and assignments (such as time, number of questions, etc.). The second functional requirement is Evaluation and Feedback which demands the platform to automatically evaluate and grade exams by utilizing the LLM. Additionally, the requirement states that the platform should be able to provide detailed feedback to students regarding their exams. The last functional requirement is managing the evaluations which asserts that the instructors should have the option to monitor student performance through a dashboard. Moreover, the system must be explainable, meaning that the evaluations made should be explained by the LLM and the instructors should be able to supervise these evaluations and manage the scores of the students. On the other hand, the performance requirements as stated in the Performance Requirements (Section 1.2.2), firstly include accuracy, demanding the system to be at least 90% accurate in question generation and evaluation.

The second performance requirement is time, which addresses the system's ability to generate at least one question in every five seconds and the evaluation results being provided within 5 seconds for most question types, with coding challenges taking up to 10 seconds. Scalability is the third performance requirement which means that the system should handle up to 1.000 concurrent users with low latency. The last performance requirement is availability, which addresses that the platform maintains 99.9% uptime during operational hours.

Various tests will be conducted to verify the LLM model's accuracy in generating and scoring questions. Generated questions and scoring will be evaluated by the experts to analyze the results. To check the interface usability, platform usability testing sessions are planned to be done where a group of students interact with the system. Thus, various bugs and problems can be observed in the early stages. Backend integration is also planned to be tested to ensure systems work without downtime or data inconsistency. In order to design the system with respect to the stated requirements, it is crucial to examine the experiments that will be performed to ensure that the system works as desired, and how the data will be collected and analyzed. Hopefully, the data may be provided by the advisors; Dr. Binnur Kurt and Dr. Fatih Kahraman from the database of Bahçeşehir University (past exam questions, answers, homework and assignments). As the provided data will be classified, it is important to take the necessary measurements to anonymize the data before utilizing it and prevent any possible data leaks. As this is not guaranteed, other data sources are considered as well. Questions and answers open source databases (such as MMMU: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI, which consists of multi-disciplinary college questions and answers [7]) will be utilized as well as AI-Generated data in case of lacking sufficient data.

5. SUMMARY AND CONCLUSION

In summary; throughout the capstone project, we focus on building a self assessment tool using the Large Language Model for educational purposes at Bahçeşehir University. To handle a high number of students, homeworks, exams and materials; we carefully planned the structure of the system to make it fair and reliable. The system that we designed automates the production and scoring of assignments and tests by exploiting LLMs' dynamic content generation and evaluation capabilities. By considering critical parts in the designing process including Backend part integration with cloud base system and user-friendly frontend, we aimed to build a robust LLM education system.

As a conclusion, we highlighted the importance of a fair automated educational system where assignments, grading, exams and all other educational materials are done by Large Language Models. By building an automated educational system, we aim to set more personalized learning to students with reducing workload on every instructor. After considering potential risks, we detailly planned our solutions and confirmed them after preliminary tests. We will do more explorations and work on security features and scalability to make them improved. We will finalize the testing protocols and start the full-scale integration with subsystems in the beginning upcoming semester. We put our effort into ensuring all components interact efficiently in the integration process.

ACKNOWLEDGEMENTS

We wish to thank our advisers Dr. Binnur Kurt and Dr. Fatih Kahraman for their help.

REFERENCES

1. C.-H. Chiang, W.-C. Chen, C.-Y. Kuan, C. Yang, and H.-Y. Lee, “Large Language Model as an Assignment Evaluator: Insights, Feedback, and Challenges in a 1000+ Student Course”, [Online]. Available: https://aclanthology.org/2024.emnlp-main.146.pdf [Accessed: Dec. 21, 2024].

2. Y.-T. Lin and Y.-N. Chen, “LLM-EVAL: Unified Multi-Dimensional Automatic Evaluation for Open-Domain Conversations with Large Language Models”, [Online]. Available: https://arxiv.org/pdf/2305.13711 [Accessed: May 23, 2023]

3. Y. Liu et al.,” Aligning with Human Judgement: The Role of Pairwise Preference in Large Language Model Evaluators”, [Online]. Available: http://arxiv.org/pdf/2403.16950 [Accessed: Aug. 10, 2024].

4. D. Hirunyasiri, D. R. Thomas, J. Lin, K. R. Koedinger, and V. Aleven, “Comparative Analysis of GPT-4 and Human Graders in Evaluating Praise Given to Students in Synthetic Dialogues”, [Online]. Available: https://arxiv.org/pdf/2307.02018 [Accessed: Jul. 05, 2023].

5. Z. Chu, Q. Ai, Y. Tu, H. Li, and Y. Liu, “PRE: A Peer Review Based Large Language Model Evaluator”, [Online]. Available: https://arxiv.org/pdf/2401.15641 [Accessed: Jun. 03, 2024].

6. K. D. Dunnell, T. Painter, A. Stoddard, and A. Lippman, “Latent Lab: Large Language Models for Knowledge Exploration”, [Online]. Available: https://arxiv.org/pdf/2311.13051 [Accessed: Nov. 21, 2023].

7. X. Yue *et al.*, “A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI” , [Online]. Available: https://mmmu-benchmark.github.io [Accessed: Sep. 05, 2024].

8. M. Fariz, S. Lazuardy, and D. Anggraini, “Modern Front End Web Architectures with React.Js and Next.Js,” *International Research Journal of Advanced Engineering and Science*, [Online]. Available: https://irjaes.com/wp-content/uploads/2022/02/IRJAES-V7N1P162Y22.pdf [Accessed: 2022].

9. R. Sawhney, *Beginning Azure Functions: Building Scalable and Serverless Apps*. 2019.

10. A. Luca, “POLITECNICO DI TORINO User Interface Development of a Modern Web Application Candidate Marzieh SOMI,” Available: https://webthesis.biblio.polito.it/secure/30076/1/tesi.pdf [Accessed: 2021].