Software Requirement Specification Document for Automated Question Generator

Youssef Alamrousy, Hassan Reda, Mohamed Ahmed, Hagar Hussam
Supervised by: Dr. Osama Talaat, Lina Bassel

February 16, 2024

Table 1: Document version history

Version	Date	Reason for Change
1.0	7-Jan-2024	SRS First version's specifications are defined.

GitHub: https://github.com/YoussefAmrousy/MoodleAutomatedQuestionGenerator

Contents

Abstract

The planned project seeks to revolutionize test production inside educational platforms by employing modern automation and Natural Language Processing (NLP) approaches. The solution supports the seamless integration of Moodle content, allowing instructors to easily develop various questions, customize question types, and preview the questions for evaluation. Security mechanisms enable restricted access based on Moodle user roles, and the user-friendly interface improves usability. Performance optimization, compatibility testing, and scalability concerns contribute to a dependable and efficient system. The project's goals include speeding test development, lowering educator effort, providing greater customization, assuring a user-friendly interface, and utilizing advanced NLP-driven analysis for contextually relevant and diverse questions. Implementing these standards not only overcomes the issues associated with traditional question formulation, but also lays the way for a more efficient, personalized, and complex evaluation process.

1 Introduction

1.1 Purpose of this document

This Software Requirements Specification (SRS) document's main goal is to outline and clarify the requirements needed to develop the Automated Exam Generation Project. This project aims to determine the number of questions needed for each exam and to speed up the process of creating different sets of questions with varying degrees of difficulty. The project also intends to facilitate the creation of diverse question sets with varying degrees of difficulty to speed up and streamline the question creation process.

1.2 Scope of this document

This Software Requirements Specification (SRS) document's scope is to investigate systems comparable to a platform for automated exam generation. It seeks to give a thorough overview and define the parameters of the Automated Exam Generation system design. This document requires a definition of the platform's goals, the types of users it is intended for, and a detailed list of both functional and non-functional requirements. It will also cover data architecture, and design constraints, and provide an application's basic class diagram. This document will also provide a structured timeline for the Automated Exam Generation system's development phases and clarify possible operational scenarios.

1.3 Business Context

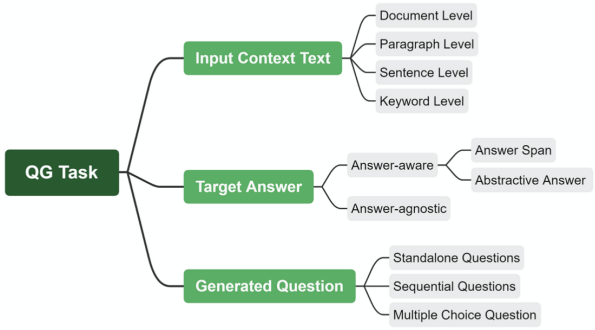
The goal of this project is to create an add-on module for our e-learning platform that will help students, instructors, and teaching assistants (TAs). This innovative addition will improve the efficacy and efficiency of assessment procedures by optimizing the process of creating automated exams. The tool's natural interface will enable professors and TAs to create comprehensive exams with simplicity and give students a structured, engaging assessment environment. Beyond our current scope, this add-on may find wide popularity with other educational organizations as a useful tool for automating test creation in their E-learning platforms. Universities can easily integrate this

functionality due to its adaptability, which will improve the examination process for educational organizations outside of their university.

2 Similar Systems

2.1 Academic

The purpose of this paper is to provide an overview of question creation from natural language literature.[1] It presents a thorough taxonomy of question-generating tasks and analyses existing models across several dimensions to obtain insights for future development. The article addresses the problems and advances in generating natural and relevant questions from various input forms, with an emphasis on natural language text. It also covers several hot subjects in question generation, such as diversified question generation, question generation pre-training, question generation with higher cognitive levels, and question generation for information searching. The study gives an excellent overview of question creation from natural language text, however, certain parts might be improved. For example, while it covers varied question production and the need for higher-level cognitive inquiries, it might use more specific examples and case studies to demonstrate these themes. Furthermore, the article might dig deeper into the real-world uses of question creation, such as in educational settings or conversational AI systems. Furthermore, a more in-depth exploration of the limits and problems of existing question-generating models, as well as potential ethical issues associated with the usage of produced questions, would be desirable. These enhancements would give a more complete and practical knowledge of questions generated from natural language text.



Computational intelligence techniques are attracting more and more attention in NLP and text analysis applications [2]. As per the research paper, the AQG framework employs a blend of methods, such as the Named-entity recognition (NER) parser and Super-sense tagging (SST), to recognize the primary verb in a sentence and categorize it as a "Wh"-question, such as "Who", "What", "Why", and so on. Additionally, the framework uses a rating module user interface, which asks the user to score the generated questions from 0 to 10, to assist the model in producing better questions. The primary finding of the study is that the AQG (Automatic Quiz Generation) framework that was created was able to produce high-caliber quiz questions that were on par with those created by teaching professionals. The good quality of automatically generated questions was indicated by the study's finding that students could not tell the difference between questions that were created manually and those that were generated automatically. The study does, however, also imply that further improvement of the question generation is necessary. The model is then trained using the ranking questions. The framework's drawbacks are also covered in the research paper, including how hard it is to come up with "Wh"-clause queries because so much domain expertise is needed and how much training is necessary. In general, the AQG framework may automate the process of creating quiz questions, saving teachers time and effort while also raising the caliber of the questions that are produced.

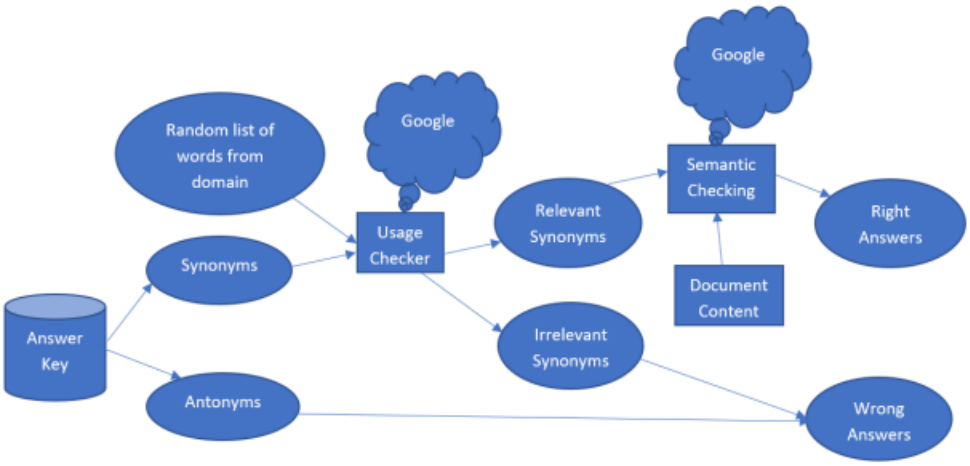
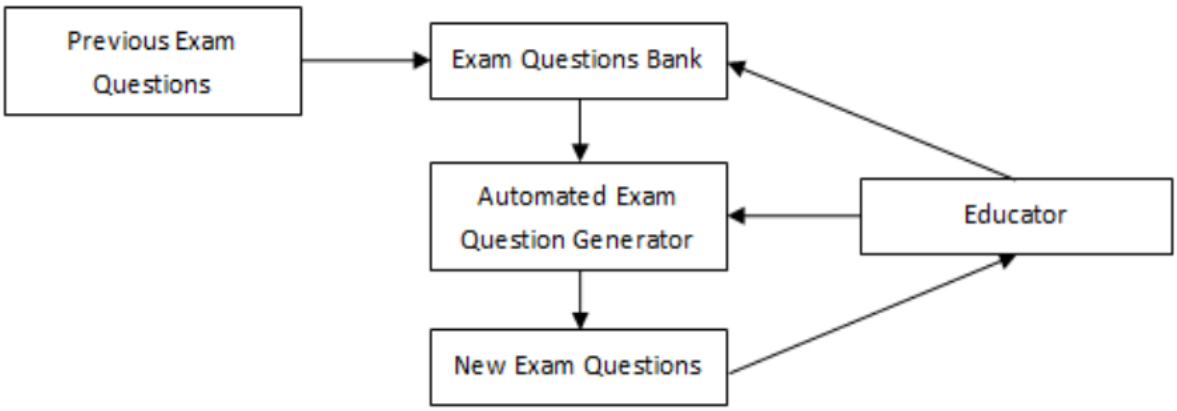


Fig. 5. Approach for generating distractors

The purpose of this research is to ease the educators' work in the process of preparation exam questions [3]. The time-consuming process of creating exam questions places a heavy strain on educators. The study's goal is to reduce the time teachers must spend preparing exam questions, freeing up more time for them to focus on lesson plans and improving their instructional strategies. The researchers developed an Automated Exam Question Generator using a Genetic Algorithm to solve the problem of burden on educators in preparing exam questions manually. The generator can produce questions that evaluate different levels of learners based on Bloom's cognitive domains. The generator was tested on two courses of the degree program. The researchers used the Automated Exam Question Generator to create a prototype that included 500 sample questions. Various numbers of chapters were chosen for each test case. The automated exam generator can extend to be used for any type of exam questions and it can be used for preparation of quiz or test questions. I don't find anything to criticize in this paper as the paper authors provided the data used in this research paper and how they managed to gather this data and they provided us with the main results.



2.2 Business Applications

The landscape of automated question generation and educational technology, **Quizgecko** joins other notable systems such as **ExamSoft**, **ProProfs** Quiz Maker, and Canvas Quizzes. **Quizgecko** distinguishes itself as an all-in-one platform, allowing educators to effortlessly create and distribute quizzes, tests, and exams in minutes. Its unique feature set includes intelligent question generation from provided text, the ability to create quizzes directly from Word, PDF, or Google Docs for premium subscribers, and AI-driven grading for short-answer questions. **Quizgecko** goes further by providing learning tips and feedback, adding an extra layer of educational support. Together with other similar systems, **Quizgecko** contributes to streamlining assessment processes and enhancing the efficiency of educational practices.

3 System Description

3.1 Problem Statement

The primary issue is time-consuming, as we previously mentioned in the proposal. Exam question creation is an exhausting task that begins with the preparation of the lectures that are included and ends with the extraction of the questions' content from these lectures. The question type is another issue. Some questions should not be simple for students to answer; instead, they should be more difficult and tailored to each student's needs. However, some instructors with less expertise may find it difficult to generate questions of this type. It could be more difficult for instructors to choose the difficulty level for each question while constructing the test because it shouldn't be extremely simple or complex.

3.2 System Overview

3.2.1 User Login

The user needs to have an authenticated account on Moodle by the institution. This system is only available to specific roles such as Teaching Assistants, Professors, Instructors, and admins.

3.2.2 Lecture Upload

Specific file types will be accepted with a maximum file size to ensure that the operation goes smoothly as expected. The English language is the only acceptable lecture type. The system isn't responsible if any other language is entered, it's not responsible to check for the language so it won't produce the expected output.

3.2.3 File to Text Conversion

The file is converted to text so it can be used later in the question generator module.

3.2.4 Text Extraction Module

AI Models will be used to remove unwanted data such as lecture titles and subtitles, hyperlinks, and references. Once this process is done, the analyzed text is produced containing lecture content only.

3.2.5 Question Generator

Two Inputs are entered, Question type and number of questions. The question generator AI model is going to use this data with the analyzed text using the Transformers module, NLP module, and Evaluation metrics so the questions can be generated successfully. Starting with the Transformers module, It's a model from hugging face which consists of pre-trained models that help in text generation. Evaluation metrics are used to develop metrics and measure the quality of generated questions and answers.

3.2.6 Question Bank

The generated questions will be stored in the question bank. Every course will have its question bank to avoid any conflicts between different questions.

3.2.7 Difficulty

Once the questions are generated and stored in the question bank, the professor can specify the difficulty (Easy/Medium/Hard) for the questions needed and the questions that match the criteria will be outputted.

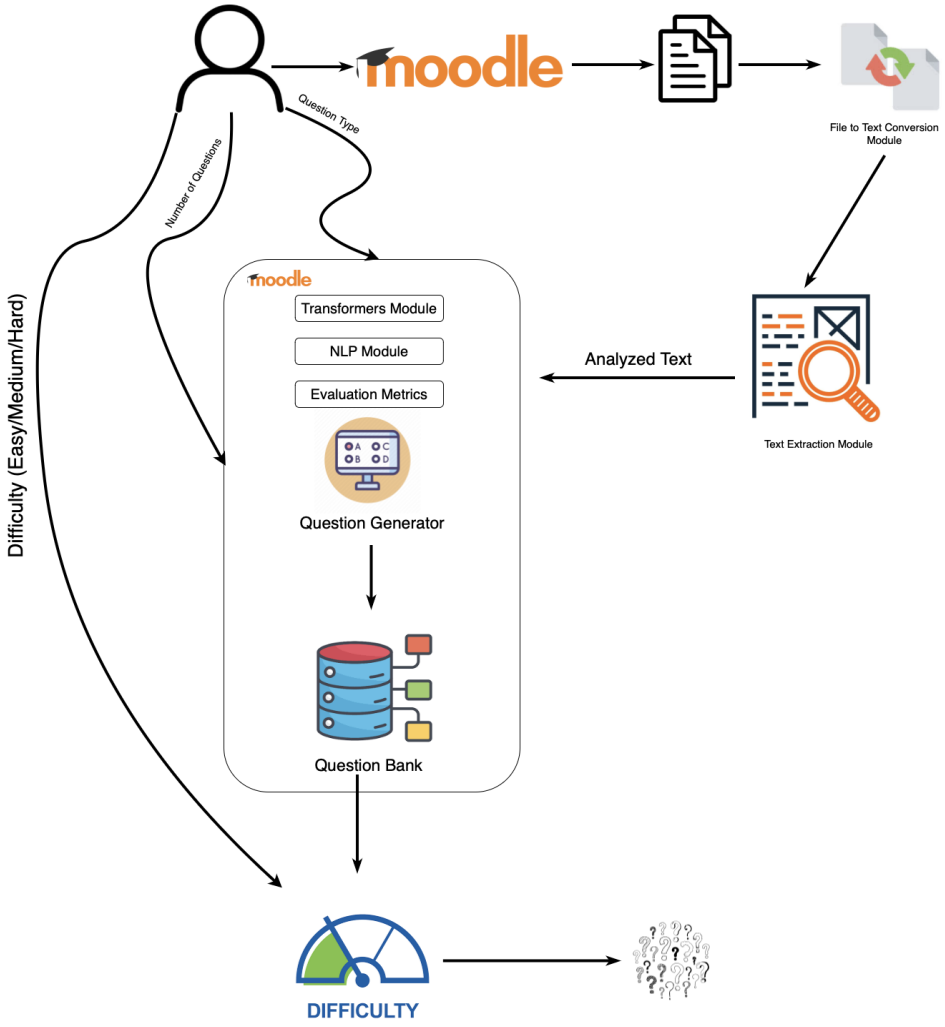
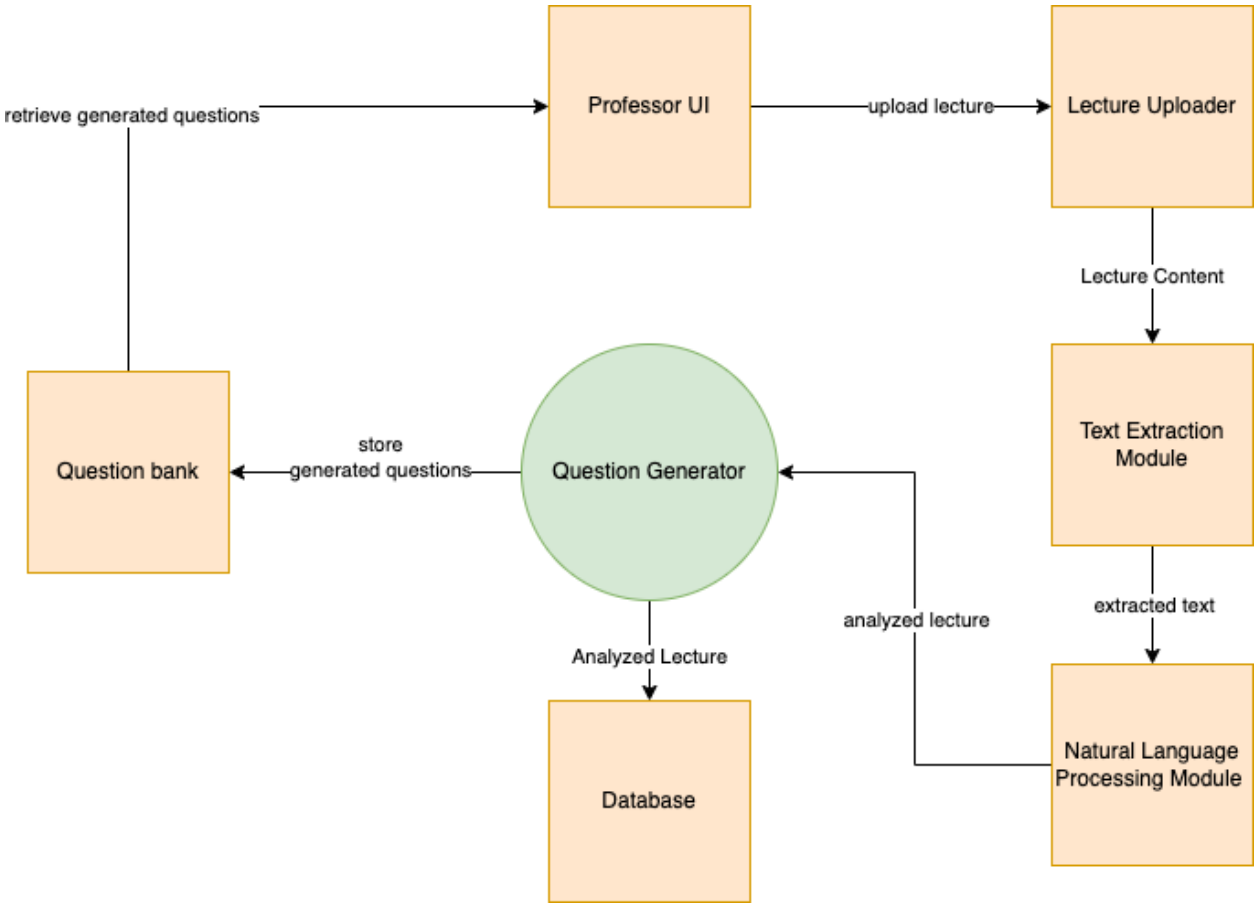


Figure 1: System Overview

3.3 System Scope

- **Question Generation:** Our system is designed to automatically generate exam questions by extracting the content of lectures supposedly (PDF, PPTX, or docx) within a learning management system (Moodle).
- **Customization:** Educators can customize the exams' structure to their liking, by specifying the quantity and the types of questions, along with preferred difficulty levels.
- **Integration with LMS:** Our system seamlessly integrates with the Moodle LMS and works as a plug-in for facilitating the extraction of lecture content and the delivery of the generated questions.
- **Natural Language Processing-Driven Analysis:** Our system analyses the lecture material using natural language processing (NLP) techniques to generate questions that are appropriate for the given context as well as assigning more appropriate difficulty levels.

3.4 System Context



3.5 Objectives

- **Streamlining and simplifying** the process of exam creation for educators by automating the generation of varied and contextually relevant questions based on lecture content.
- **Time efficiency and workload reduction:** By automating the traditionally exhausting process of creating questions, our system saves instructors a significant amount of time and lessens their workload.
- **Enhanced Customization:** empowers educators with the ability to personalize exams, they can effortlessly specify the number and types of questions, as well as preferred difficulty level.
- **User-friendly Interface:** To make the UI easy to use for educators, make it intuitive and user-friendly. The purpose of this goal is to improve the user experience as a whole.
- **Advanced NLP-driven Analysis:** Utilise cutting-edge Natural Language Processing (NLP) methods to analyze lecture content in a sophisticated manner. To promote a more complex and efficient assessment process, this seeks to provide questions that not only accurately reflect the context of the subject matter but also assign appropriate degrees of difficulty.

3.6 User Characteristics

- Teachers with different levels of technical proficiency use the system to generate test questions automatically.
- Teachers must be familiar with Moodle LMS and the process of creating an online exam.
- Students use the system's output as the automated-generated exam is available for the students to take.

4 Functional Requirements

4.1 System Functions

- **ID: 01** The professor will upload the lectures to the system.
- **ID: 02** The professor should be able to define the number of questions to be generated.
- **ID: 03** The professor should be able to choose the difficulty of the questions.
- **ID: 04** The professor can choose the type of question (MCQ, Essay, T/F).
- **ID: 05** The professor will have a preview of the generated questions for review.
- **ID: 06** The professor should be able to remove a certain question from the preview section.
- **ID: 07** The professor can save the questions in the question bank.
- **ID: 08** The professor can view the question bank.
- **ID: 09** The professor can delete questions from the question bank.
- **ID: 10** The professor will have the option to filter questions in the question bank by difficulty.
- **ID: 11** The professor will be able to search for questions by lecture title in the question bank.
- **ID: 12** The professor can edit the question and its model answer.
- **ID: 13** The system will accept certain types of files.
- **ID: 14** The system will accept files until a certain size limit.
- **ID: 15** The system will generate questions with the model answers.
- **ID: 16** The system will store questions with their model answers in the question bank.

Figure 2: Professor Use Case

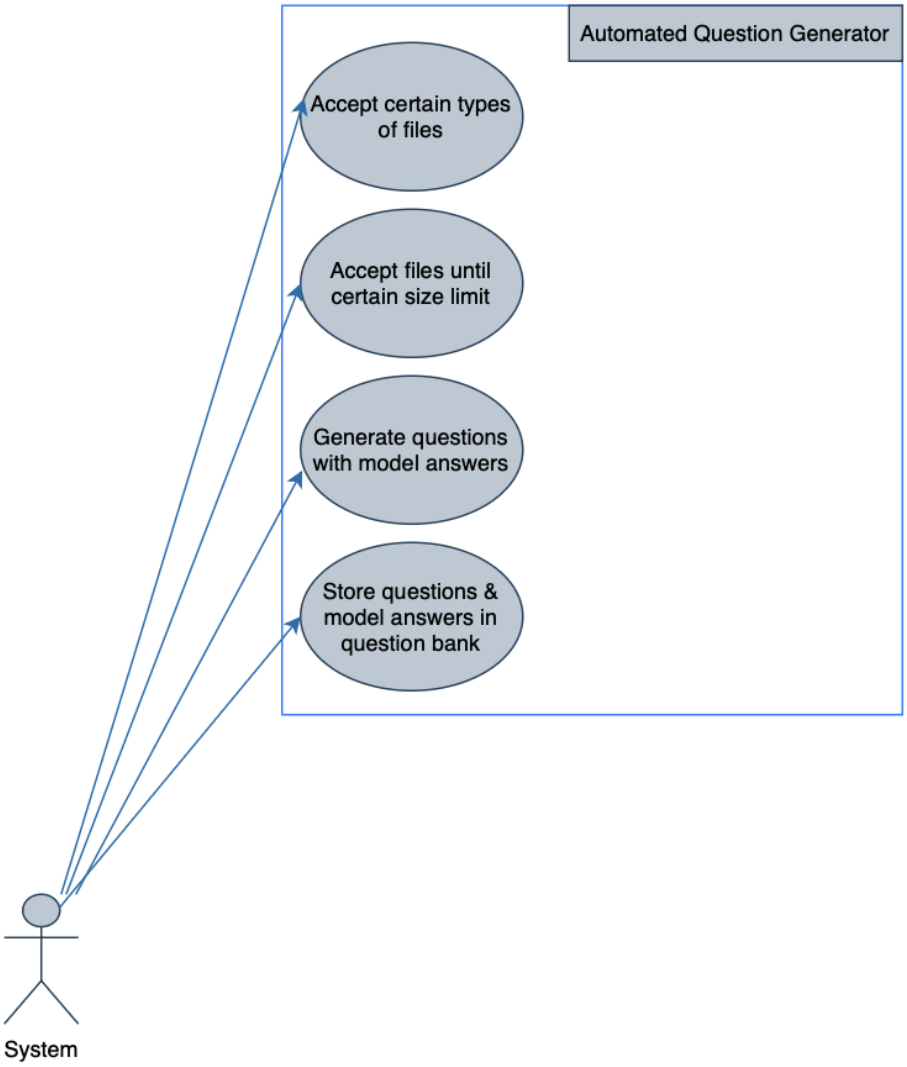


Figure 3: System Use Case

4.2 Detailed Functional Specification

Table 2: Upload lecture Function Description

Name	Upload Lecture
Code	ID:01
Priority	High
Critical	To give the application data to process and classify.
Description	The function is responsible for uploading the lecture to the system to get processed and classified.
Input	File
Output	Boolean: true if a file is accepted.
Pre-condition	Educator having already uploaded lectures on Moodle
Post-condition	The file is processed
Dependency	depends on the presence of a lecture to upload.
Risk	No lecture uploaded

Table 3: Choose the type of question Function Description

Name	Choose the type of question
Code	ID:04
Priority	Medium
Critical	To give the application specs for the question to process and classify.
Description	The function is responsible for providing the type of question needed to get processed and classified (MCQ, Essay, T/F).
Input	Text
Output	Text: A generated question of the same type
Pre-condition	N/A
Post-condition	The question is generated and passed to be reviewed
Dependency	depends on ID: 01
Risk	N/A

Table 4: Save questions to question bank Function Description

Name	Save Questions in Question bank
Code	ID:07
Priority	High
Critical	Save the question for further use.
Description	The function is responsible for saving the generated question in the question bank.
Input	Text: Question
Output	Text: Message to confirm saving.
Pre-condition	Having a question bank.
Post-condition	The question is saved.
Dependency	Depends on the presence of a question bank.
Risk	No Question Bank

Table 5: Search by Lecture Function Description

Name	Search by lecture title
Code	ID:11
Priority	low
Critical	To reduce the educator's time instead of manual searching.
Description	The function is responsible for searching in the question bank with the lecture's title.
Input	Text: title
Output	Text: questions based on the lecture's title.
Pre-conditions	Some questions are added to the question bank.
Post-conditions	Questions related to the specific lecture are selected.
Dependency	depends on the presence of questions in the question bank with the storing of the lecture's title.
Risk	No questions in the question bank.

Table 6: Generate questions with Model answers Function Description

Name	Generate questions with Model answers
Code	ID:15
Priority	High
Critical	To give the application a data to process and classify.
Description	The function is responsible for generating questions with model answers based on previous specifications.
Input	Text: Quantity, type, difficulty.
Output	Text: Questions and model answers.
Pre-condition	Question specifications are entered.
Post-condition	Questions with their model answers are generated.
Dependency	depends on ID: 02, 03, 04
Risk	N/A

5 Design Constraints

- Only the professor role will have access to our system.
- The system will require internet connection to work.
- The system will only accept certain types of files (PDF, PPTX, docx) for lecture content.
- The system will have a maximum allowable size for uploaded files.
- The system will only accept English language material.
- The system will support three types of questions (MCQ, Essay, T/F).
- The system must integrate advanced Natural Language Processing (NLP) techniques for analyzing lecture content.

5.1 Standards Compliance

- The system follows standards set by Moodle for seamless integration, ensuring compatibility and efficient data exchange.
- The system follows industry-standard security practices to protect against unauthorized access and data breaches.

5.2 Other Constraints as appropriate

- The project must be completed within a specified timeframe, considering academic schedules and deadlines.

- The system must operate within the legal and ethical boundaries of educational technology and assessment practices.

6 Non-functional Requirements

6.0.1 Security

Restrict access to the system according to Moodle user roles. Ensuring that users can view question banks related to current courses enrolled in.

6.0.2 Usability

The user interface should be intuitive and user-friendly.

6.0.3 Performance

Response time for question generation should be within acceptable limits. Load-balancing techniques are to be used to distribute multiple requests across multiple server instances.

6.0.4 Compatibility

Only supported file types can be uploaded to the lecture upload module.

6.0.5 Scalability

A huge well-optimized database is to be created to store all the question banks generated for each course.

6.0.6 Reliability

Design the system to continue working, in the presence of faults or errors. Ensuring consistent performance and availability upon varying conditions.

6.0.7 Maintainability

Specify the best code standards and practices such as SOLID Principles to guarantee long-term maintainability and easier bug fixes.

7 Data Design

We're going to have a dataset to train the model on the question difficulty. This dataset will contain (Easy, Medium, and Hard) questions inserted manually and the model will be trained several times to adapt the model to the format of the difficult or easy questions. Our project has a main database called question bank which is going to store all the questions generated with their model answers. Each question will have a unique ID. Each question will have an attribute in the database to store

the professor's name in case any problem happens. There will be a date attribute for each question showing the creation date.

8 Preliminary Object-Oriented Domain Analysis

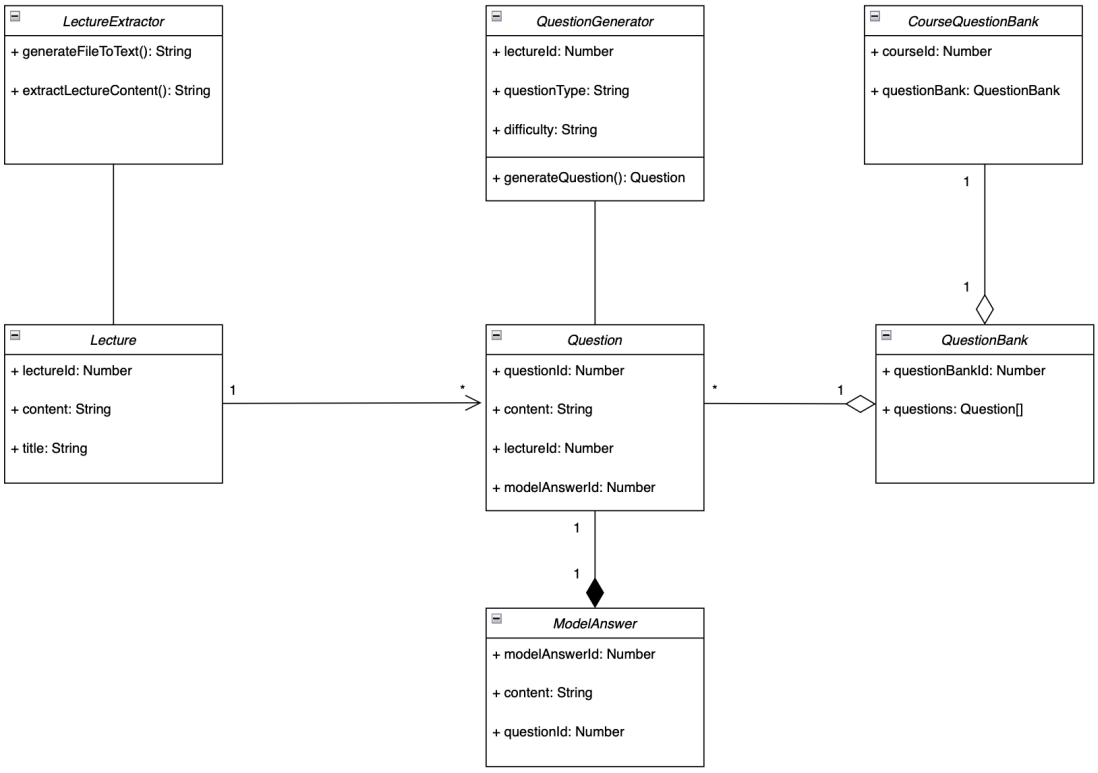


Figure 4: Class Diagram

9 Operational Scenarios

Scenario 1: An educator chooses a particular course from the moodle integrated Learning Management System (LMS) after logging into the system. Relevant material from recently held lectures in that subject is automatically extracted by the system. The Examiner modifies the format of the test by determining the quantity, types, and degrees of difficulty of the questions. After verification, the system analyses the material using Natural Language Processing (NLP) and produces a list of various, contextually appropriate questions. The questions are examined and approved by the educator to be used in future tests.

Scenario 2: To get ready for a test for a different subject, another educator signs in. This educator has preferences for more difficult questions and has a particular question structure. The produced questions are customized by the system based on these inputs. The Examiner reviews

the proposed questions, modifies, and completes the exam design. The system's versatility and customization capabilities are demonstrated by the way it makes sure the specified preferences are represented in the produced questions.

10 Project Plan

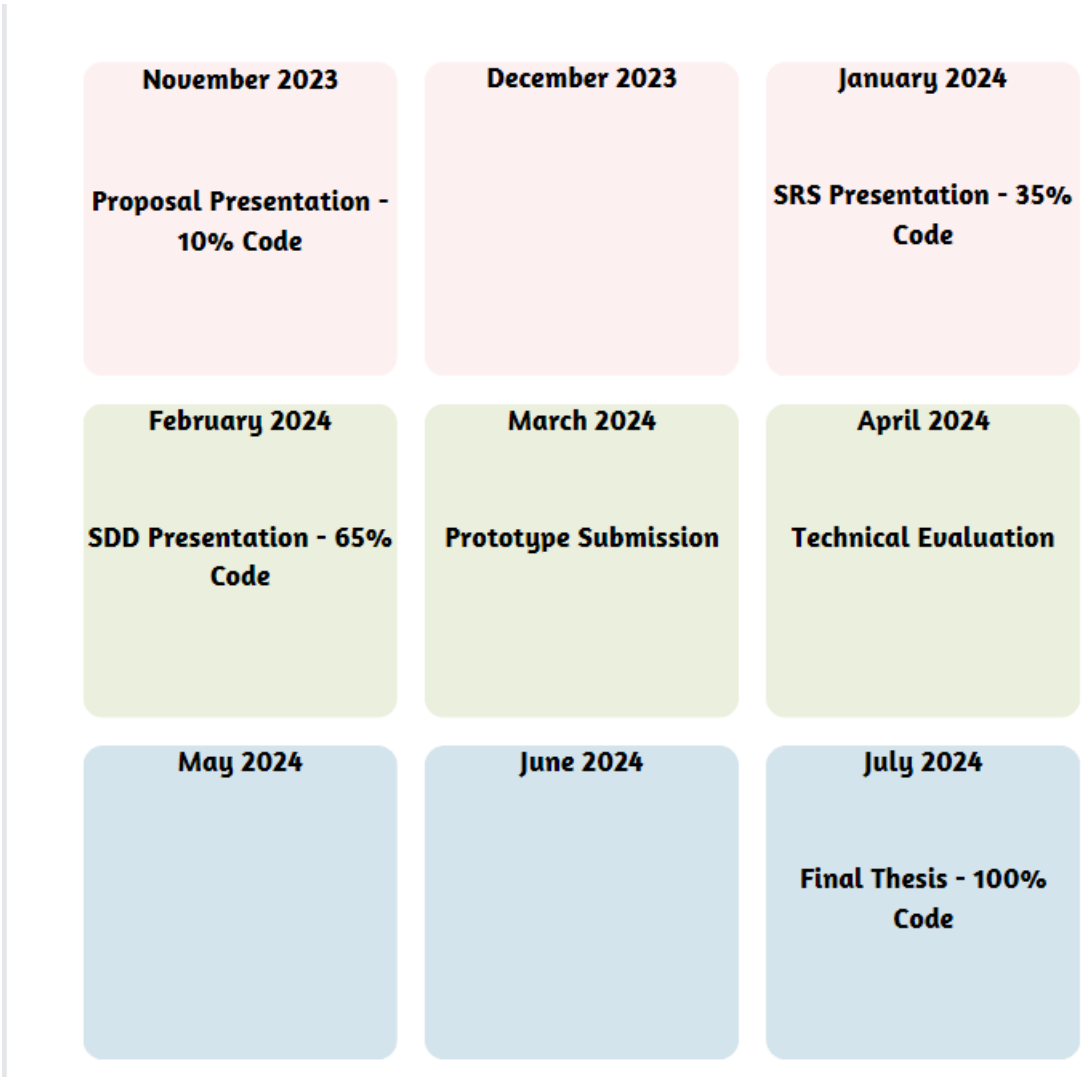


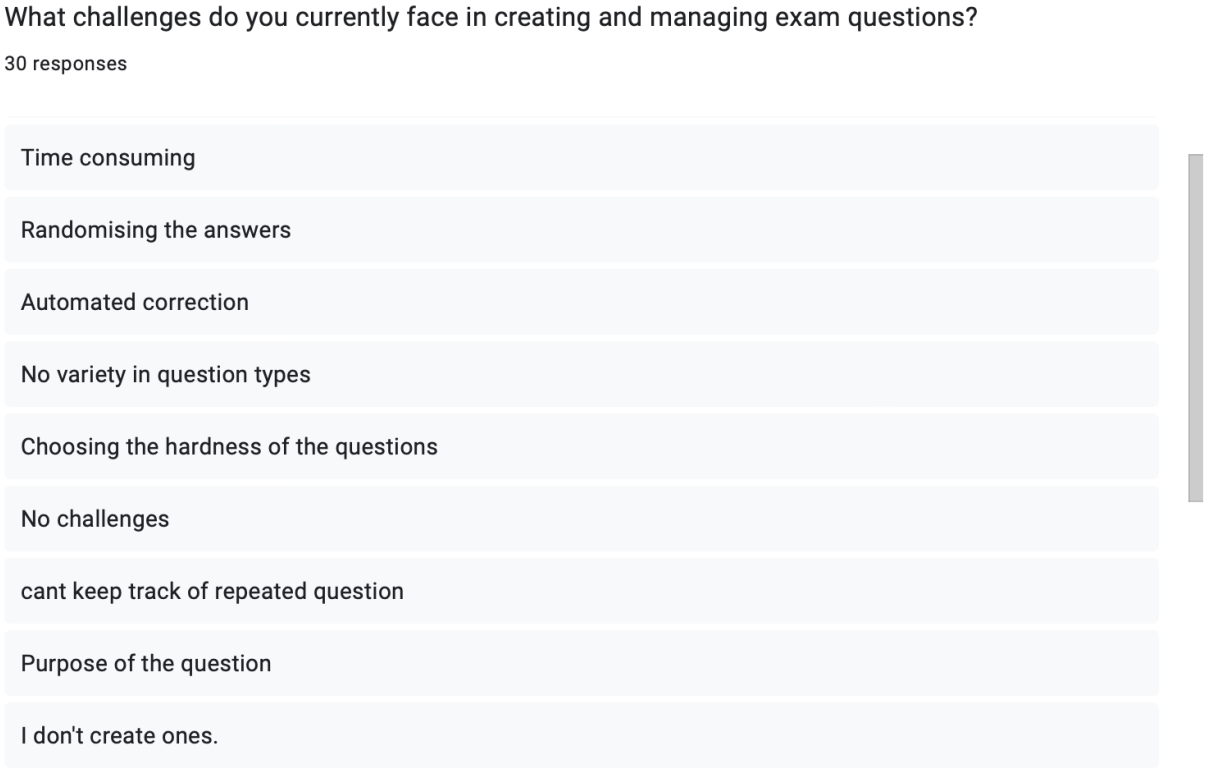
Figure 5: Time Plan

11 Appendices

11.1 Definitions, Acronyms, Abbreviations

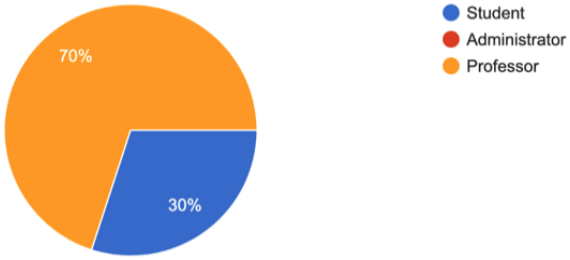
MCQ: Multiple-Choice Question - **T/F:** True/False
NLP: Natural Language Processing
LMS: Learning Management System
SRS: Software Requirements Specification
UI: User Interface
KPIs: Key Performance Indicators

11.2 Supportive Documents



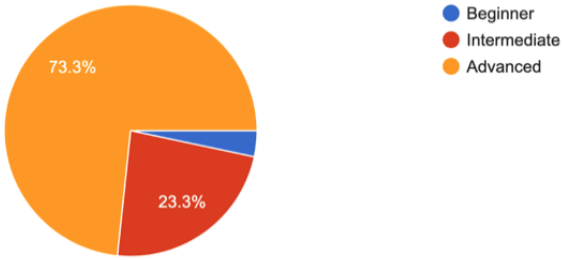
Role in Education

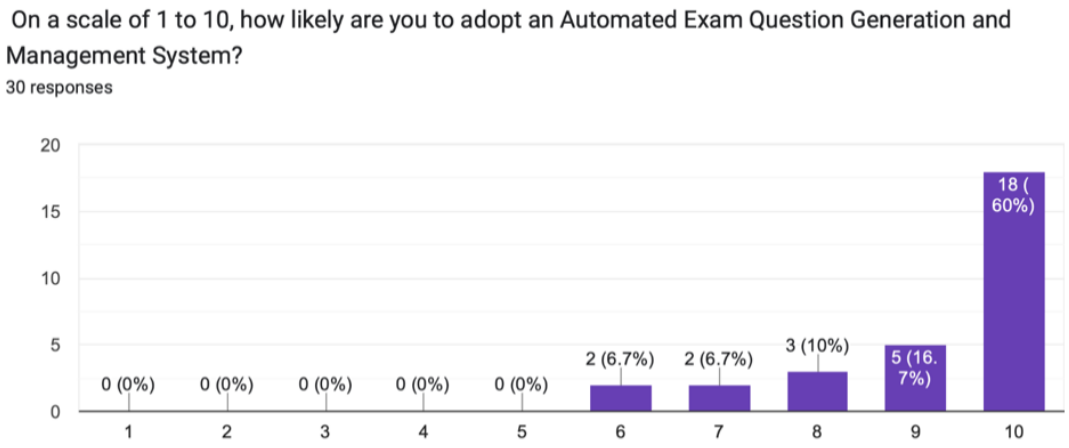
30 responses

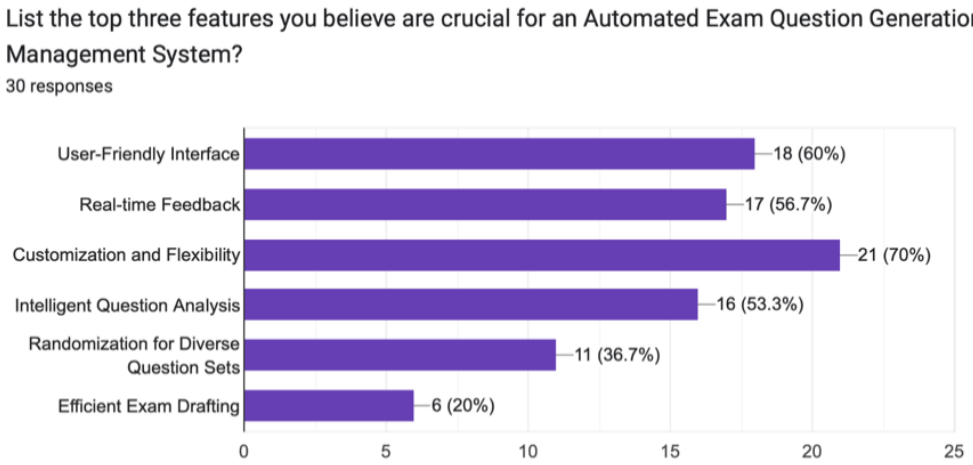


Experience with Moodle

30 responses







References

[1] Ruqing Zhang, Jiafeng Guo, Lu Chen, et al. "A Review on Question Generation from Natural Language Text". In: *ACM Trans. Inf. Syst.* 40.1 (Sept. 2021). ISSN: 1046-8188. DOI: 10.1145/3468889. URL: https://doi.org/10.1145/3468889.

[2] Akhil Killawala, Igor Khokhlov, and Leon Reznik. "Computational Intelligence Framework for Automatic Quiz Question Generation". In: *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. 2018, pp. 1–8. DOI: 10.1109/FUZZ-IEEE.2018.8491624.

[3] T. N. T. A. Rahim, Z. A. Aziz, R. H. A. Rauf, et al. "Automated exam question generator using genetic algorithm". In: *2017 IEEE Conference on e-Learning, e-Management and e-Services (IC3e)*. 2017. DOI: 10.1109/ic3e.2017.8409231.