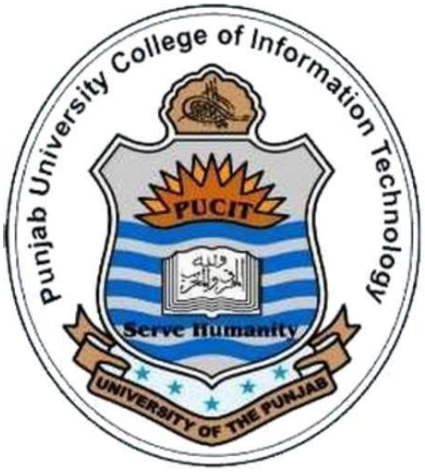


Final Year Project

AI-Powered Assessment Platform: ExAmigo

Proposal By:

Name	Roll Number
Abdul Moeed	BSDSF21A004
Syed Hamza	BSDSF21A019
Iqra Haris	BSDSF21A047

Advisor: Dr. Adnan Abid

Department of Data Science

University of the Punjab

Lahore, Pakistan

(2021-2025)

Table of Contents

ExAmigo

Introduction:

In today's fast-paced educational landscape, teachers and students alike face challenges in balancing the demands of teaching, learning, and assessment. Traditional methods of quiz creation and grading can be time-consuming and inefficient, often leaving little room for personalization or innovation. Recognizing these challenges, **ExAmigo** was conceived as a transformative solution to revolutionize the teaching and learning experience.

ExAmigo is not just another web-based application; it is a vision realized through cutting-edge technology and user-centric design. The platform is built to empower educators and learners with tools that make education smarter, faster, and more engaging. Designed to bridge the gap between traditional teaching methods and modern technological advancements, ExAmigo combines the power of automation, artificial intelligence, and seamless user interfaces to create a dynamic educational ecosystem.

Motivation:

The traditional methods of creating and grading quizzes in educational institutions are often time-consuming, repetitive, and prone to errors. Teachers spend countless hours crafting questions, evaluating answers, and providing feedback, leaving them with less time to focus on personalized instruction and student engagement. Similarly, students preparing for competitive exams like MDCAT or attempting to master theoretical subjects struggle to find tailored resources and efficient ways to assess their progress. This gap between effort and outcomes in education inspired the creation of **ExAmigo**.

The motivation behind ExAmigo lies in the vision of making education more accessible, efficient, and interactive through technology. By leveraging the power of artificial intelligence and automation, the system addresses the challenges faced by both teachers and students, streamlining the quiz creation, grading, and feedback process. ExAmigo empowers educators to focus on their passion for teaching while enabling students to achieve their academic goals with tools designed to meet their specific learning needs.

Example:

Imagine a teacher, Sarah, who has been struggling to create quizzes for her students while juggling lesson planning, grading, and administrative tasks. She spends hours crafting questions, grading papers, and providing feedback, leaving her exhausted and with little time to focus on improving her teaching methods. On the other hand, there's Ali, a student preparing for his

medical entrance exam, MDCAT. He finds it challenging to evaluate his readiness for the test, as existing tools lack the customization he needs to generate quizzes tailored to his weak areas. ExAmigo was designed with users like Sarah and Ali in mind. By automating repetitive tasks like quiz generation and grading, ExAmigo enables teachers to focus on teaching and mentoring while giving students like Ali a personalized and efficient way to prepare for their exams.

Problem Statement:

The traditional process of quiz creation and evaluation, especially in theoretical subjects, is time-consuming and inefficient. Students need personalized and frequent quizzes to better prepare for exams, but manual quiz generation can be inconvenient. Furthermore, current digital tools often fall short in catering to subjective evaluations. The unmet needs are:

- Lack of automated quiz generation for theoretical subjects.
- Absence of a system that evaluates and grades quizzes, including MCQs, true/false, short questions, and optionally, subjective papers.
- Inefficient manual efforts required from teachers for quiz generation.

Who needs it?

- **Students:** Students need continuous assessment tools to track their progress and prepare for exams effectively.
- **Teachers:** Teachers need tools that automate quiz creation and grading to save time and provide consistent feedback to students.
- **Educational Institutions:** Schools and universities looking for scalable solutions to automate assessments will benefit from this application.

Objectives:

- Develop a user-friendly web-based platform for students and teachers to access quiz generation and assessment features.
- Implement dynamic quiz generation for multiple question types, including MCQs, true/false, short questions, and descriptive questions.

- Automate the evaluation and grading of quizzes, focusing on theoretical subjects, particularly for MDCAT preparation and computer-based courses.
- Provide real-time feedback to students based on their quiz performance to enhance their learning experience.
- Enable optional subjective evaluation for descriptive questions, allowing teachers to manually grade and provide insights.
- Create a custom dataset to support building a unique model for quiz generation and evaluation, enhancing the system's accuracy and efficiency.

Scope:

- The application will support two user roles: student and teacher.
- Procedural generation of quizzes in various formats, with MCQs primarily focused on MDCAT preparation.
- Automatic evaluation and grading for MCQs, true/false, and short questions.
- Descriptive questions will be generated for computer-based courses, with subjective evaluation optional and manually handled by the teacher.
- The focus is on theoretical subjects, specifically targeting exam preparation for medical and computer-based courses.

Limitation:

- The system will not cover practical or technical subjects.
- The platform will be designed for exam preparation and will not support real-time classroom engagement.
- Subjective answers are evaluated based on keywords, context, completeness and model answers, which might not fully capture drawbacks in student responses

LITERATURE REVIEW

Related Work:

Existing Quiz Systems:

Platforms like Google Forms, Quizizz, and Kahoot! allow teachers to create quizzes and automate grading for objective questions (MCQs, true/false). However, these systems primarily focus on objective-type questions and lack support for subjective evaluation.

Disadvantages:

- Limited to objective questions.
- No procedural quiz generation.
- Inadequate for subjective, theory-based subjects.

1. ## Automated Grading Systems:

Tools like **EdX** and **Grammarly** use AI for grading essays and short answers, but they struggle with understanding scientific, complex responses in theoretical subjects, often requiring human intervention. **Disadvantages**:

- Poor accuracy in grading scientific answers.
- Requires manual oversight for content evaluation.

3. ## Virtual University Quiz Systems:

The **Virtual University of Pakistan (VUP)** offers a comprehensive system for quiz generation, mainly using automated processes to generate quizzes for students across various theoretical courses. The system focuses on objective questions such as MCQs and true/false but also includes subjective questions, which require manual grading by instructors. However, VUP's system lacks dynamic quiz generation features based on real-time data or adaptive learning.

Disadvantages:

- Limited procedural quiz generation.
- Manual grading for subjective questions.

Gap Analysis:

Current systems are inadequate for generating and grading quizzes for theoretical subjects, especially with scientific and technological questions. Our project fills this gap by providing:

- Automated generation and evaluation of MCQs, true/false, short answer, and subjective questions.
- AI-based evaluation designed specifically for theoretical subjects, improving accuracy and reducing manual effort.

2. Project Proposal/Goal:

Project Title: Automated Self Assessment

Group Leader: Abdul Moeed

Project Members:

Name	Registration#	Email Address	Signature
Abdul Moeed	BSDSF21A004	Bsdsf21a004@pucit.edu.pk	
Hamza Faiz	BSDSF21A019	Bsdsf21a019@pucit.edu.pk	
Iqra Haris	BSDSF21A047	Bsdsf21a047@pucit.edu.pk	

Project Goal:

To develop a web-based application that dynamically generates quizzes and evaluates them, primarily focusing on MDCAT preparation and computer-based courses, enhancing exam readiness for students and streamlining assessment for teachers.

Objectives:

Sr. #	
1.	Develop a user-friendly web-based platform for students and teachers to access quiz generation and assessment features.
2.	Implement dynamic quiz generation for multiple question types, including MCQs, true/false, short questions, and descriptive questions.
3.	Automate the evaluation and grading of quizzes, focusing on theoretical subjects, particularly for MDCAT preparation and computer-based courses.

4.	Provide real-time feedback to students based on their quiz performance to enhance their learning experience.
5.	Enable optional subjective evaluation for descriptive questions, allowing teachers to manually grade and provide insights.
6.	Create a custom dataset to support building a unique model for quiz generation and evaluation, enhancing the system's accuracy and efficiency.

Project Success criteria: Successful implementation of a web-based application that dynamically generates and evaluates quizzes, achieving 75% to 80% user satisfaction and accuracy in assessments.

Assumptions:

- Users possess basic computer skills and internet access to utilize the application effectively.
- The context matching between two given texts is possible, enabling accurate evaluation and grading of quizzes.

Risk and Obstacles:

- Challenges in accurately matching students' answers based on four measures: Concreteness, Completeness, Accuracy, and Relevancy, which may impact the evaluation process.

Organization Address: Department of Data Science, University of the Punjab, Lahore, Pakistan.

Target End Users: Students, Teachers, Educational Institutes.

Suggested Project Supervisor: Prof. Dr. Adnan Abid

Approved By: Prof. Dr. Adnan Abid

Date: October 3, 2024

Key Features:

1. A Seamless Start: User Registration and Role-Based Access

ExAmigo ensures a smooth onboarding experience for users through a secure registration process. Teachers and students can sign up by entering their basic details and choosing their respective roles. Once logged in, the system dynamically adjusts its interface to present features specific to the user's role:

- **Teachers** gain access to tools for creating quizzes.
- **Students** are equipped with resources to self-assess their knowledge and track their progress.

The system incorporates advanced security features like hashed password storage, role-based access controls, and session timeout mechanisms to ensure data protection and privacy.

2. Study Material Upload and Parsing

One of the standout features of ExAmigo is its ability to process study materials uploaded by teachers. Whether it's a PDF, DOCX, or other supported formats, the system intelligently parses the content to extract relevant information. This extracted content serves as the foundation for generating quizzes, ensuring that all questions align with the intended topics. Teachers no longer have to worry about manually drafting quizzes from scratch, as ExAmigo simplifies the process with unparalleled efficiency.

3. AI-Driven Quiz Generation

The heart of ExAmigo lies in its quiz-generation engine. Teachers can create quizzes by specifying the topic, difficulty level, and question types (e.g., MCQs, true/false, short questions, and descriptive). The system uses AI algorithms to generate procedurally crafted quizzes tailored to the provided study material.

Students can also generate practice quizzes for self-assessment, allowing them to test their understanding and identify areas for improvement. This feature is particularly beneficial for MDCAT and computer science courses, where students need focused preparation.

4. Automated Grading and Feedback

ExAmigo's grading system eliminates the manual effort required to evaluate quizzes. For objective questions, grading is instant and accurate, providing students with their scores immediately after submission. For descriptive questions, the system leverages natural language processing (NLP) to analyze responses, ensuring that grading is fair and consistent.

Detailed feedback is provided, including a breakdown of scores and suggestions for improvement. This feature is invaluable for students aiming to enhance their knowledg.

5. AI Chatbot: The On-Demand Tutor

ExAmigo also features an AI-powered chatbot designed to assist users with queries and provide real-time support. The chatbot can:

- Summarize study materials.
- Explain complex concepts in simple terms.
- Offer guidance on how to use the platform effectively.

By acting as a virtual assistant, the chatbot ensures that users are never left without help, making learning and teaching more accessible and efficient.

6. Progress Monitoring and Recommendations

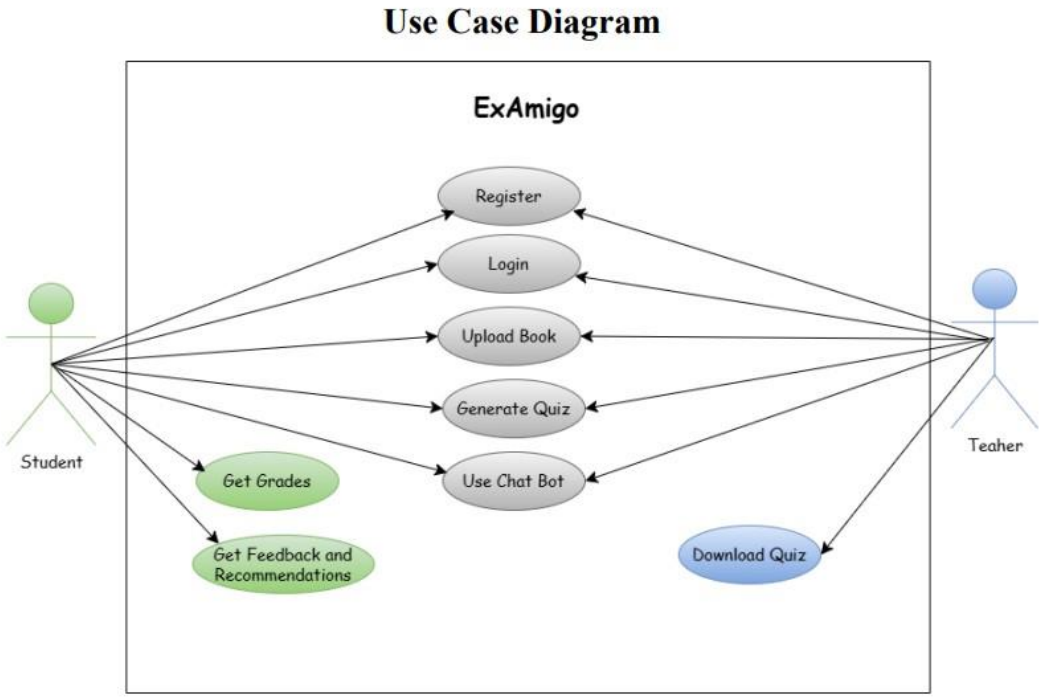
Both students and teachers benefit from ExAmigo's advanced analytics module. Students can view detailed progress reports highlighting their strengths and weaknesses.

The system's recommendation engine suggests areas for improvement based on quiz results, helping users make data-driven decisions for better outcomes.

7. Scalability and Deployment

ExAmigo is designed with scalability in mind, making it suitable for classrooms of any size. The application is deployed in a Dockerized environment, ensuring smooth operations even with high user loads. The system supports simultaneous activities, such as quiz generation, grading, and data uploads, without compromising performance or data integrity.

USE CASE DIAGRAM



Structure of **Use Case Diagram**:

1. **Actors**:
    - Student
    - Teacher
2. **Use Cases**:
    - Registration
    - Login
    - Upload Book
    - Query Interaction (Chatbot)
    - Generate Quiz
    - Grade Quiz
    - View Quiz Performance Report (Feedback and Recommendations)
    - Download Quiz

3. **Relationships**:

    - Both **Student** and **Teacher** will have associations with almost all use cases.

    - The **System** plays a central role in managing the process of grading, feedback, and generating quizzes. It also processes the upload and handles the download functionalities.

Detailed explanation of how the **Use Case Diagram** along with the interactions between the actors (student, teacher, and the system) and the system's various functionalities.

Actors:

1. **Student**

    - This actor is a primary user who wants to interact with the system for educational purposes. They will mainly use the system to upload study material, generate quizzes, take quizzes, receive feedback, and download quizzes for future use.

2. **Teacher**

    - This actor also interacts with the system, but their role is more focused on creating and generating quizzes and content uploading.

Use Cases:

1. Login (Student, Teacher):

Description: Both the student and teacher actors must log into the system before accessing any features. The login process ensures that the system knows the role of the user and provides them with role-specific access to content and functionalities.

Interaction:

- Students and teachers provide credentials (email/username and password).
- The system authenticates the credentials and grants access based on the user's role.
- Once logged in, students can access features related to quiz-taking and performance tracking, while teachers can access quiz creation and content uploading.

2. **Upload Book (Student, Teacher):**

Description: Both students and teachers can upload study material (PDF, DOCX) to the system. The system will parse the material and generate a table of contents for reference.

Interaction:

- The user selects the file they want to upload.
- The system validates the file type (PDF/DOCX) and checks the size.
- Once the file is uploaded, the system extracts the content and makes it available for use in the quiz generation process.

3. **Query Interaction (Chatbot) (Student, Teacher):**

Description: After uploading the study material, both students and teachers can interact with a chatbot to ask questions related to the uploaded content. The chatbot can answer factual questions, provide summaries, and explain concepts.

Interaction:

- The user types a question related to the study material (e.g., "What is the definition of X?").
- The system processes the query and responds with an appropriate answer, a summary, or an explanation, depending on the nature of the query.

4. **Generate Quiz (Student, Teacher):**

Description: After interacting with the chatbot (or if no queries are needed), the user can generate a quiz based on the uploaded content. The system provides a form where the user can specify quiz details, such as the number of MCQs, short questions, topics, and difficulty levels.

Interaction:

- The user selects the type of questions (MCQs, short questions, true/false, etc.), defines the number of questions, and selects the topic(s) they want to focus on.
- The system then generates the quiz dynamically based on the provided information. It assembles the questions, applies the difficulty levels, and formats the quiz.
- For students, the quiz is displayed on the screen, and they can start solving it.

5. Grade Quiz (System):

Description: After the student takes the quiz, the system automatically grades the quiz based on predefined criteria for each question type (MCQs, True/False, Short Answer, Descriptive).

Interaction:

- The student submits the quiz after completing it.
- The system evaluates each answer (MCQs and True/False are graded instantly, while short answers are evaluated based on predefined keywords or phrases).
- For descriptive questions, the system uses context-based evaluation to assess the answer.
- The grading system generates a score and detailed feedback for the student.

6. **View Quiz Performance Report (Student):**

   Description: After grading, the student can view a detailed report that provides feedback on their performance, including accuracy, speed, and areas for improvement.

   Interaction:

   - The system generates a performance report that shows the total score, accuracy percentage, and speed of completion.
   - The system also provides a visual representation (charts/graphs) showing progress over time and areas where the student needs improvement.

7. **Download Quiz (Student, Teacher):**

   Description: After the quiz is generated, both students and teachers have the option to download the quiz in different formats (PDF, DOCX, TXT) for offline use or to share with others.

   Interaction:

   - The user clicks on the download icon/button.
   - A modal window appears, asking the user to select the format they want to download the quiz in (PDF, DOCX, TXT).
   - The user selects the format and specifies the location on their device where the file should be saved.
   - The system prepares the file, ensures that the layout is neat and printable, and allows the user to download the file within a few seconds.

Relationships between Use Cases:

- **Login** is the first step that enables all other interactions. Without logging in, users (students or teachers) cannot proceed to any of the other actions (uploading books, generating quizzes, grading, etc.).

- **Upload Book** leads into **Generate Quiz** because the uploaded study material is the base content for quiz generation. Once a book is uploaded, the system can parse it and allow quiz creation based on the content.

- **Query Interaction** (Chatbot) happens optionally after the book is uploaded, allowing users to interact with the system for more detailed explanations or information before proceeding to quiz generation.

- **Generate Quiz** is the main functionality that both students and teachers will frequently interact with, but with different goals: students take quizzes, and teachers generate quizzes for students.
- **Grade Quiz** is automatically triggered when the student submits a quiz for grading. The system processes and evaluates the answers to provide results.

- **View Quiz Performance Report** is available for students after their quiz is graded. It allows them to see detailed insights into their performance.

- **Download Quiz** is an option for both students and teachers to save or share quizzes for future use.

User Story 1

User Registration and Authentication

The **User Registration and Authentication** module is the foundational component of the ExAmigo system, designed to ensure secure and personalized access for students and teachers. This module facilitates the seamless onboarding of new users by providing distinct registration and authentication workflows tailored to their respective roles.

The registration process allows users to sign up using separate forms for teachers and students, ensuring that the system captures role-specific details effectively. The module incorporates robust validation mechanisms to verify user inputs, such as email format, username uniqueness, and password strength. Once registered, users can log in securely using their credentials, gaining access to a personalized dashboard that caters to their unique requirements.

Security is a key focus of this module. Passwords are hashed and stored securely to protect user data, and access to role-specific features is restricted to ensure unauthorized users cannot exploit the system. Additionally, the module supports simultaneous registration and login requests for up to 100 users, ensuring scalability.

This module ensures a smooth user experience, with registration and login attempts processed within 1-3 seconds. By laying a secure and efficient foundation, the User Registration and Authentication module enables all other functionalities of ExAmigo to operate seamlessly.

Screens:

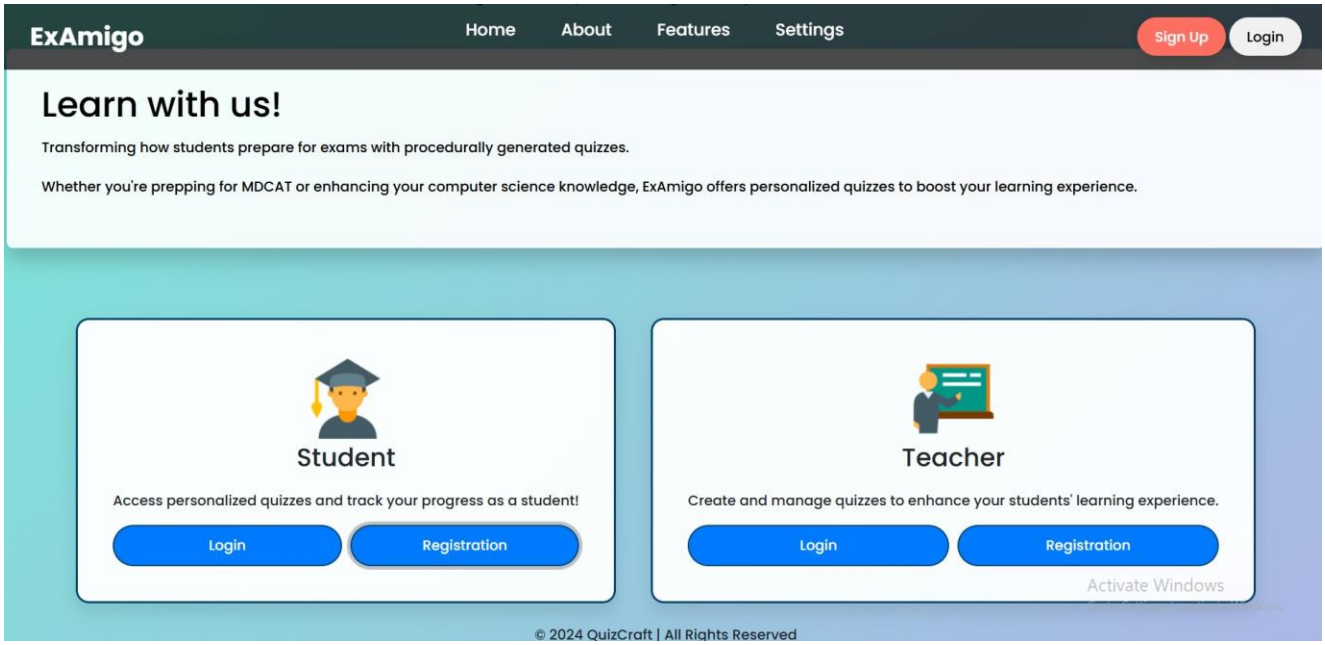


Fig 1.1: Screen with student and teacher module

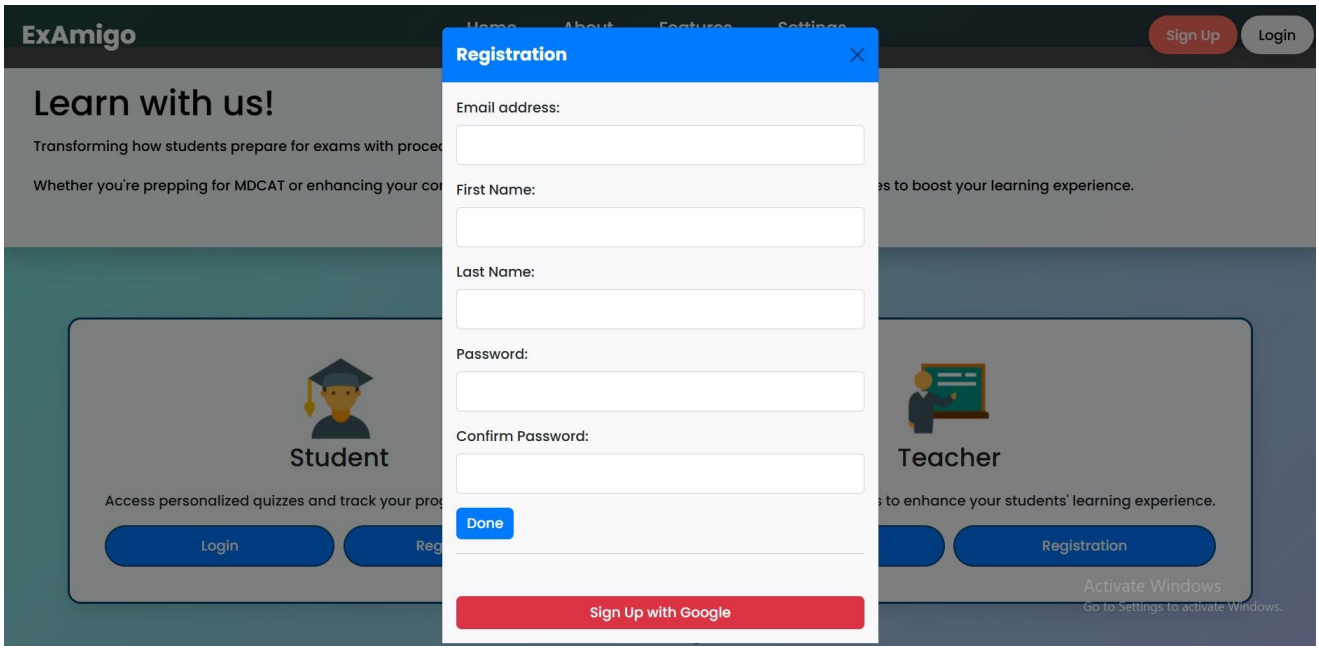


Fig 1.2: Screen with registration form

These screens shows the registration process in ExAmigo. It is simple and role-specific, with two main modules: **Teacher** and **Student**. Both roles use a shared registration form, ensuring consistency while directing users to the appropriate features.

Users start by selecting their module (**Teacher** or **Student**) via the respective registration button. After filling out the form with details like name, email, username, and password, the system registers them accordingly.

- If a user selects the **Teacher module** and completes the registration form, their account is created with teacher-specific permissions. Upon successful registration, the system redirects them to a personalized dashboard tailored to teachers. This dashboard provides functionalities such as quiz creation.
- Conversely, if a user selects the **Student module** and completes the registration form, their account is registered with student-specific permissions. Post-registration, they are guided to a dashboard designed for students, offering features such as quiz participation, performance reports, and interactive chatbot assistance.

Acceptance Criteria:

- The system should allow students and teachers to register via separate registration forms.
- The system should validate the inputs, such as email format, username availability, and password strength.
- Upon successful registration, the user should be redirected to the appropriate dashboard (student or teacher).
- The system should allow secure login and logout functionalities.
- Registration and login attempts should take no more than 2-3 seconds.
- The system should support at least 100 simultaneous registrations.
- Passwords should be securely hashed and stored in the database.
- Unauthorized access to role-specific features should be restricted.

Failure Conditions:

- If the email or username is already taken, the system should display a clear error message indicating the issue.
- If the password is weak, the system should prompt the user to select a stronger password.
- If the registration fails due to a network error or other issue, the system should inform the user with a friendly error message and prompt them to try again.
- If the user provides incorrect credentials during login, they should be prompted with an invalid login message and should not be granted access to the system.
- If the system fails to validate the input fields (e.g., email format or password strength), the registration process should halt and an appropriate error message should be displayed.

User Story 2

Login/Logout Module

The **Login/Logout Module** ensures that registered users, whether they are students or teachers, can securely log in and log out of the platform. When a user attempts to log in, the system validates the provided login credentials, such as username or email and password. This process should be quick, with authentication occurring within 3-5 seconds, ensuring a seamless experience for the users. The system must handle multiple simultaneous login requests (at least 100 users) without any delays or failures.

Once logged in, users will have access to their personalized functionalities depending on their role, either student or teacher. In addition, the system ensures that users' sessions are automatically terminated after a specified period of inactivity, typically 30 minutes, to reduce the risk of unauthorized access if the user forgets to log out.

When it comes to logging out, the system ensures that all active sessions are closed and the user is safely logged out. The logout process must also be secure, ensuring that no data is left exposed after the user exits.

To further enhance security, the system limits failed login attempts. If a user exceeds the defined number of unsuccessful login attempts (e.g., five attempts), the system will lock the account temporarily to prevent brute-force attacks. This helps to safeguard user accounts from unauthorized access.

Lastly, to protect user data, passwords are stored securely using advanced hashing algorithms, making it impossible to retrieve the actual password from the system. All of these features combined make the login/logout process both secure and efficient for all users.

Screens:

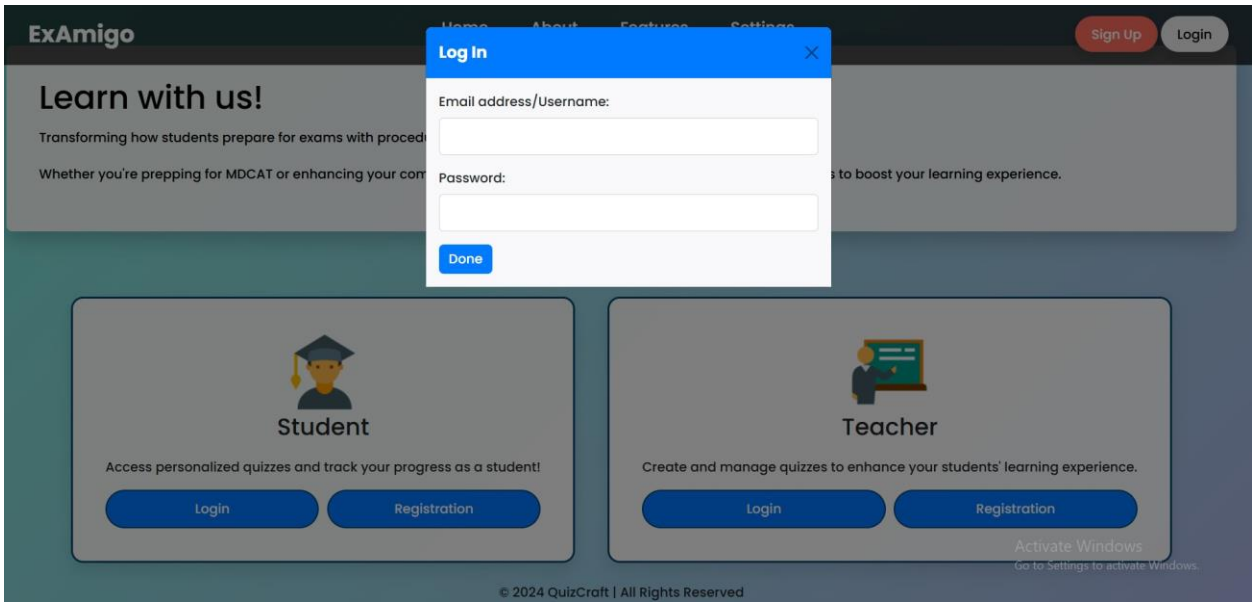


Fig 2.1: Login Modal

The login process begins with the user selecting their role, either as a **Teacher** or a **Student**, through a modal that appears when the user clicks the **Login** button. This modal presents two distinct options: one for the Teacher login and another for the Student login. Both modules share the same login form but are customized based on the user's selection.

Once the user chooses their role, they proceed to fill out the login form by entering their **username/email** and **password**. After submitting the credentials, the system performs authentication. If the login is successful, the user is directed to their respective dashboard based on the selected role:

- **Teacher**: If a teacher logs in, they are granted access to the teacher-specific functionalities such as generating quizzes.
- **Student**: If a student logs in, they gain access to the student-specific functionalities, including taking quizzes, viewing grades, and receiving performance feedback.

In the event of incorrect credentials, the system will display an error message prompting the user to re-enter their information. Additionally, if the user has exceeded the maximum number of failed login attempts, the account will be temporarily locked to enhance security.

This login process ensures that users are securely authenticated and directed to the correct set of functionalities, based on their selected role, providing a personalized experience.

Acceptance Criteria:

1. **Authentication Speed**: The system must authenticate users within 3-5 seconds after submitting login credentials.
2. **Simultaneous Login Requests**: The system should handle login attempts from at least 100 users simultaneously without any degradation in performance.
3. **Session Expiration**: User sessions should automatically expire after 30 minutes of inactivity, ensuring that unauthorized users cannot access the system if the user forgets to log out.
4. **Failed Login Attempts**: The system must limit failed login attempts to a maximum of 5. If a user exceeds the limit, the account should be temporarily locked to prevent brute-force attacks.
5. **Secure Password Storage**: User passwords must be securely hashed using encryption techniques (e.g., bcrypt, SHA-256) to prevent unauthorized access to user credentials.

Failure Conditions:

1. **Slow Authentication**: If the system takes longer than 5 seconds to authenticate a user, this could indicate performance issues or server bottlenecks. In this case, users may experience delays when logging in or out.
2. **Simultaneous Login Failure**: If the system cannot handle simultaneous login requests from at least 100 users, there could be errors or delays in the login process, causing users to be unable to access the system.
3. **Session Persistence**: If user sessions do not expire after 30 minutes of inactivity, it could lead to security vulnerabilities, where unauthorized users can access the platform if a legitimate user forgets to log out.
4. **Account Lockout Failure**: If the system does not lock accounts after five failed login attempts, malicious actors could repeatedly attempt to guess passwords, risking unauthorized access to the platform.
5. **Insecure Password Storage**: If passwords are not securely hashed or encrypted, they could be exposed in the event of a data breach, compromising user security and privacy.

User Story 3

Upload Book Module

The **Upload Book Module** allows registered users, whether students or teachers, to upload study materials for generating quizzes. Users can upload files in common formats like PDF or DOCX, making it convenient to work with diverse study resources.

The system ensures a smooth upload experience with support for file sizes up to 50MB, completing uploads in under 3 seconds. It automatically processes the uploaded content to extract relevant information required for quiz generation, ensuring accuracy and reliability. If a user attempts to upload an unsupported format or an oversized file, the system immediately displays an error message, guiding them to make corrections. This feature can handle up to 50 simultaneous uploads, ensuring efficiency and scalability.

Screens:

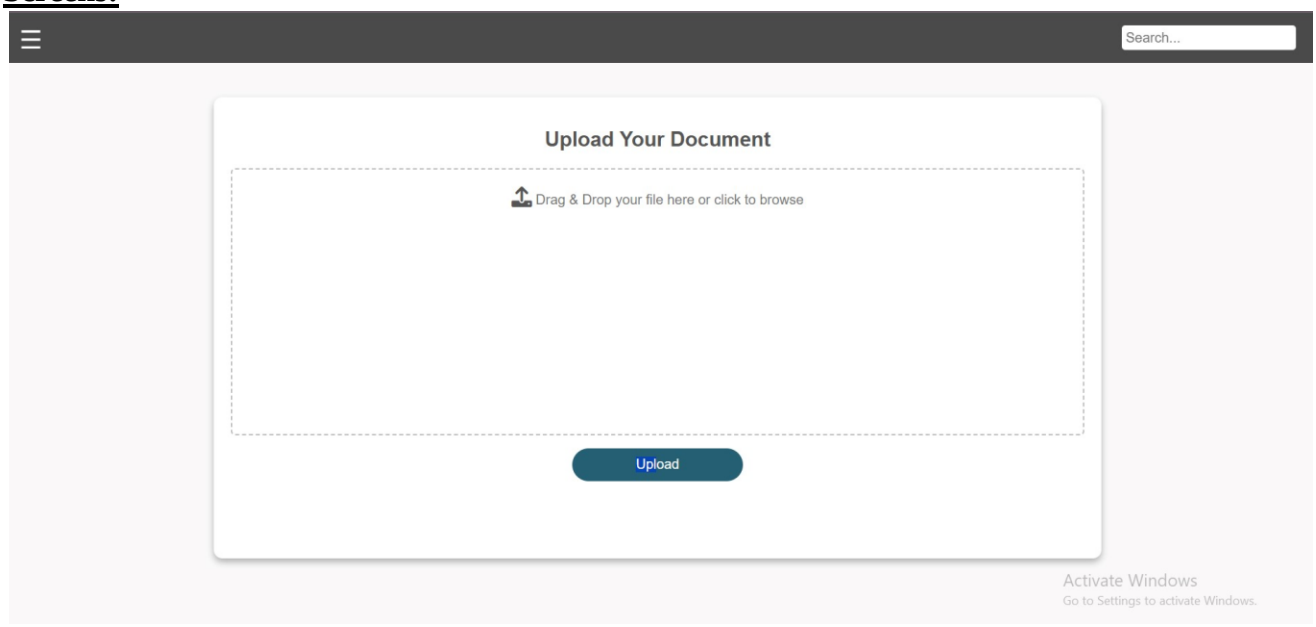


Fig 3.1: Upload Document

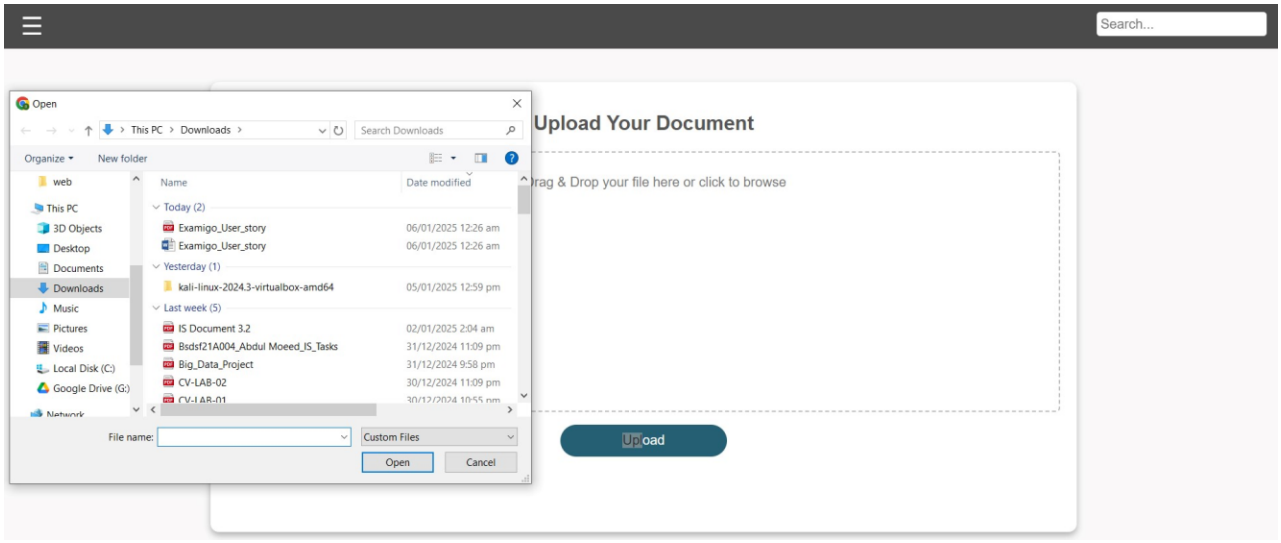


Fig 3.2: Choose File from system to upload

These screens shows the uploading content process. When a registered user (student or teacher) wants to upload a study material, they go to the **Upload Document**. After clicking the **Upload** button, they can choose a PDF or DOCX file from their device. If the file is compatible and within the 50MB size limit, the system processes it and extracts relevant content for quiz generation. If there's an issue, such as an unsupported file format or an oversized file, the system alerts the user with an error message. Once the upload is successful, the material is ready for generating quizzes. This process ensures a smooth and quick experience for users to upload study materials efficiently.

Acceptance Criteria:

- The system should support file uploads in PDF and DOCX formats.
- The upload process should complete in less than 3 seconds for a 50MB file.
- The system should be able to accurately parse and extract the content from uploaded books to generate quizzes.
- The system should display a clear error message for unsupported file formats or oversized files (e.g., greater than 50MB).
- Users should be able to upload books or study material with ease, without any issues on the backend or interface.

Failure Conditions:

- If the file is not in a supported format (PDF, DOCX), the system should reject the file and notify the user of the issue.
- If the file exceeds the upload limit (e.g., 50MB), the system should display an error message and not proceed with the upload.
- If the upload process takes longer than 3 seconds for a 50MB file, the system should alert the user with a "Server Timeout" or "Slow Upload" message.
- If the content extraction fails due to file corruption or unreadable format, the system should inform the user and prompt them to try uploading a different file.

User Story 4

Generate Quiz Module

In the **Generate Quiz Module**, both teachers and students can dynamically create quizzes based on their selected subjects and desired question types. The system allows users to specify various parameters, such as the subject, topic, difficulty level, and number of questions. The quiz can include multiple-choice questions (MCQs), true/false questions, short answer questions, and optional descriptive questions.

For teachers, the system provides the option to generate quizzes that they can download for later use. For students, the system allows them to not only generate the quiz but also solve and submit it. Additionally, students can download their quizzes for future reference or offline use.

Once the quiz is generated, the system ensures that the process takes less than 3 seconds, providing a fast and efficient experience. Users can also save the generated quizzes for later use, making it convenient for both students and teachers. The system guarantees that at least three question types are included, ensuring variety in the quizzes.

Screens:

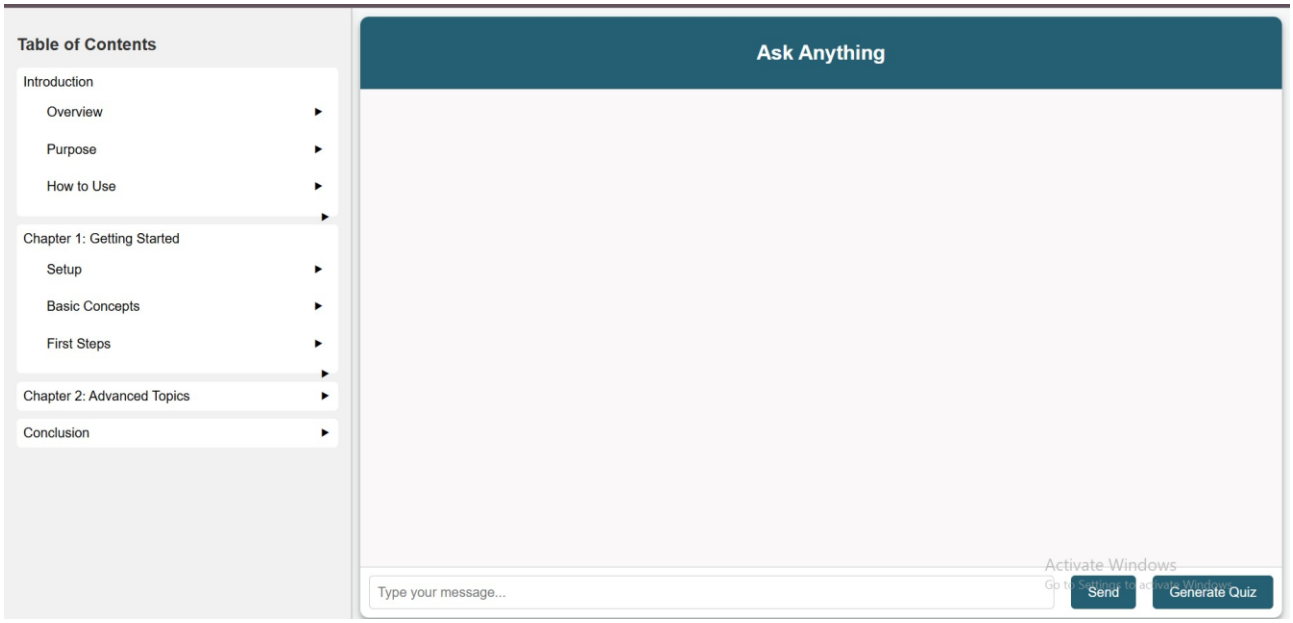


Fig 4.1: Table of content and AI-powered Chatbot

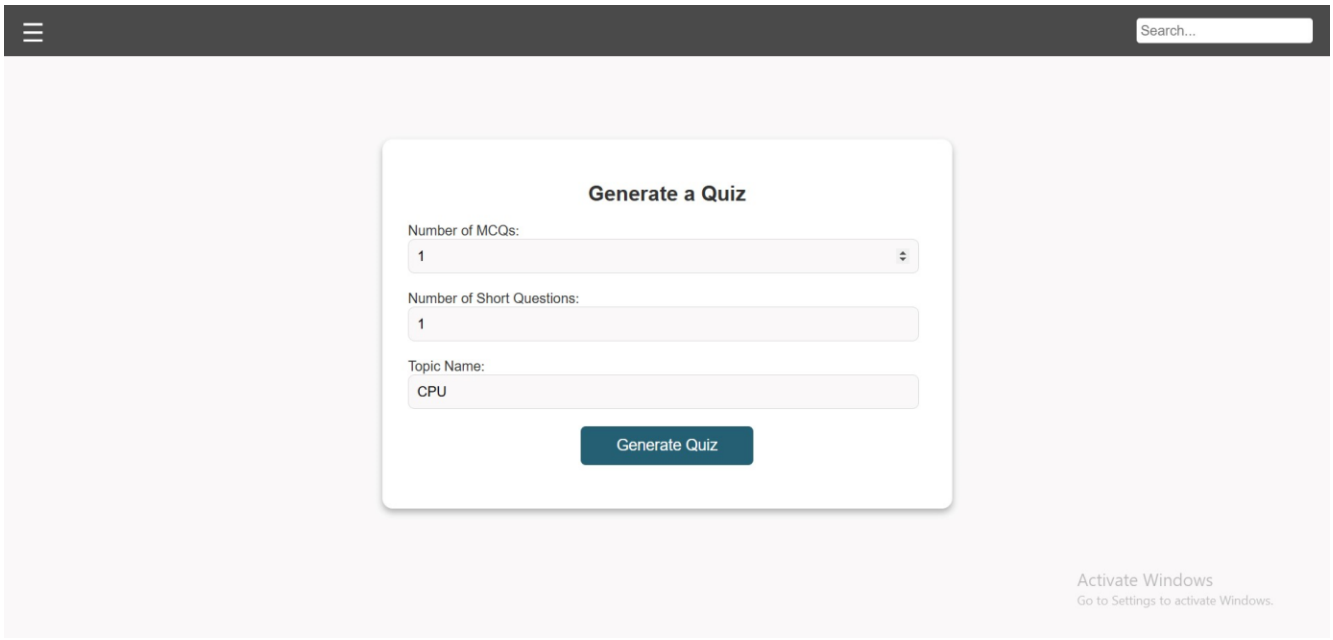


Fig 4.2: Form to generate a quiz

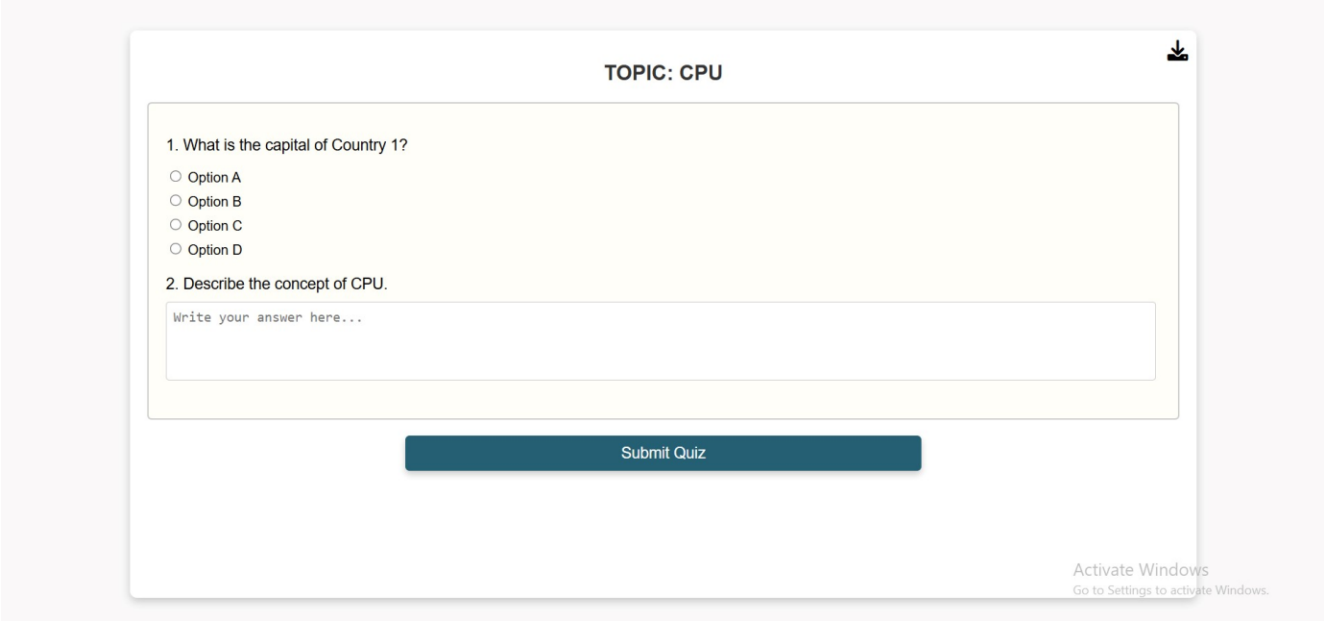


Fig 4.3: Generated quiz

In the **Upload Book and Quiz Generation Module**, the process begins when the user uploads a PDF or DOCX file containing study material, such as a book or other educational content. Once the file is uploaded, the system automatically generates a table of contents from the content. This table of contents serves as a guide for both students and teachers, helping them navigate the material efficiently.

To make the learning experience even more interactive, the system includes an AI-powered **chatbot** that assists users in understanding the uploaded material. The chatbot can answer a variety of queries, providing assistance on factual questions, offering summaries of chapters or sections, and explaining key concepts in the material. This feature is designed to enhance the learning experience by offering immediate support, making it easier for students and teachers to comprehend complex topics and find answers quickly.

If a user does not have any specific questions or queries, they can skip directly to the **Quiz Generation** process. Once the user decides to generate a quiz, they can simply click on the "Generate Quiz" button. A form will appear, allowing the user to specify the number of MCQs (Multiple Choice Questions), short answer questions, and the topics they want to include in the quiz. After filling out the form with the desired parameters, the user can generate a quiz based on the selected information.

This quiz generation process is designed to be quick and easy, giving users the flexibility to create quizzes tailored to their specific study needs. The system ensures that the generated

quizzes are based on the content uploaded and that they reflect the user's preferences for question types and topics. The ease of access to quizzes, along with the support of the chatbot, allows for a seamless, efficient learning experience for both students and teachers.

Acceptance Criteria:

- The system should allow both students and teachers to select quiz topics, question types (MCQs, true/false, short questions, descriptive), and difficulty levels.
- The system should dynamically generate a quiz based on the user's specified parameters.
- Quizzes should be generated in under 3 seconds for optimal performance.
- The generated quiz should include a mix of MCQs, true/false questions, short questions, and optionally, descriptive questions.
- The system should allow users to download or save the quiz for later use.

Failure Conditions:

- If the system fails to generate the quiz within 3 seconds, it should display an error message like "Quiz Generation Timeout."
- If the user fails to select valid or required options (e.g., question count, topic), the system should display an error message and prompt the user to correct the input.
- If the quiz generation logic encounters an error (e.g., invalid parameters), the system should inform the user and give them the option to retry with corrected information.
- If the quiz includes incorrectly formatted questions or issues in question logic, the system should highlight the problem and allow the user to fix the issue.

User Story 5

Grading Module

In the **Automatic Quiz Grading Module**, students benefit from an efficient and quick grading system. Once a quiz is completed, the system automatically grades the submitted answers. For **MCQs (Multiple Choice Questions)** and **True/False** questions, the system provides instant feedback and evaluation, completing the grading process in just 2-3 seconds. This ensures that students receive prompt results, allowing them to quickly identify areas of improvement.

For **short answer questions**, the system evaluates responses based on predefined criteria and answers, ensuring that the evaluation is consistent and accurate. In cases of **descriptive questions**, the system performs automated grading by analyzing the answers using predefined keywords and contextual understanding. While the grading accuracy for descriptive answers is

targeted at 80%, the system continuously refines its evaluation mechanism to ensure fairness and precision.
After grading, the system generates a **detailed performance report** for each student, summarizing the results of the quiz and providing insights into their strengths and areas for improvement. This report not only includes the scores for individual questions but also gives an overview of how well the student performed overall, helping them track their progress. The automatic grading system is designed to save time for both students and teachers, enabling a smooth and efficient learning and evaluation process.

Screens:

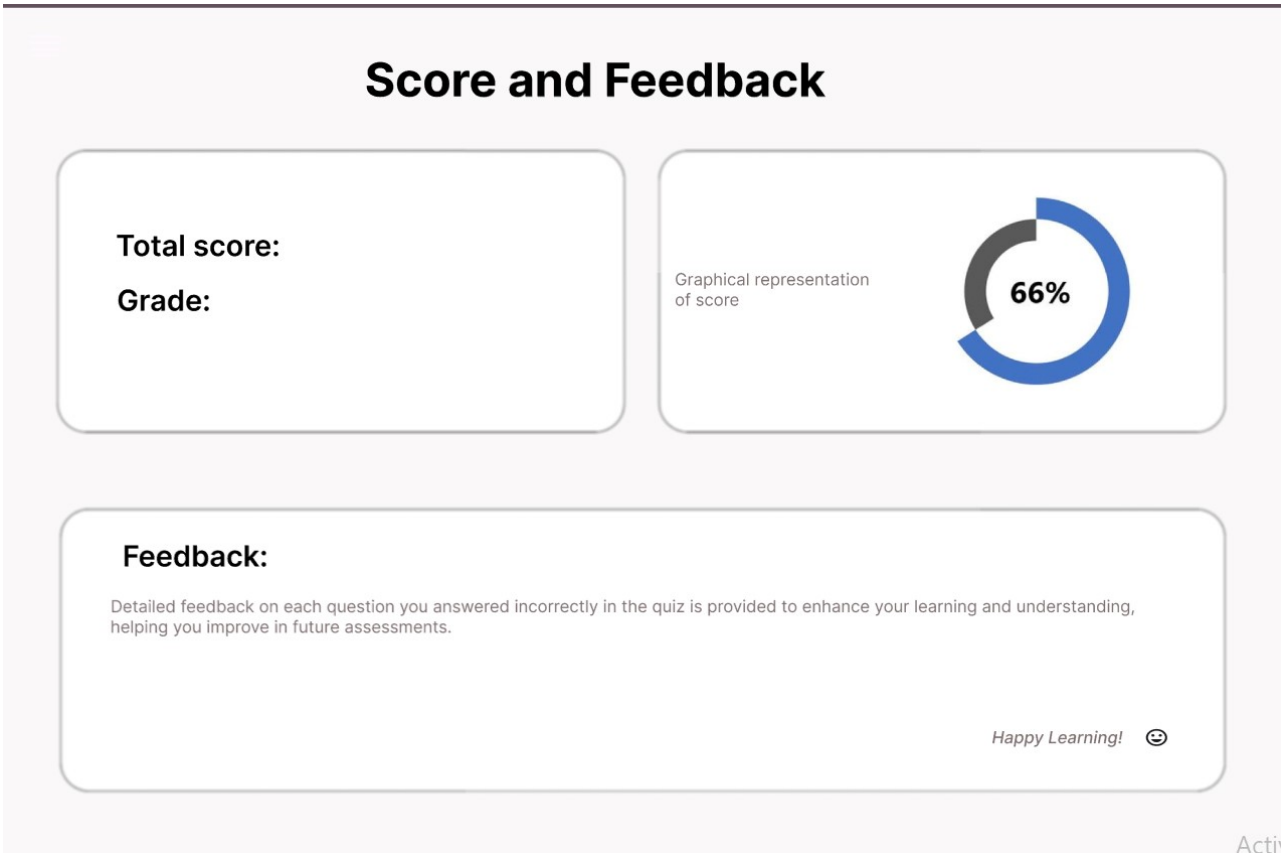


Fig 5.1: Grading Module

In the **Grading Module**, once a student completes their quiz, a comprehensive grading screen is generated to provide instant feedback and a clear summary of their performance. This screen will display the **total score** achieved by the student, offering a clear indication of how well they have done overall.

Alongside the total score, the system presents a **visual representation** of the student's score. This could be in the form of graphs, such as bar charts or pie charts, which visually break down the student's performance on different sections of the quiz. For example, the chart might show how many correct answers were given in **MCQs**, how well the student performed in **true/false questions**, and the accuracy of their responses in **short answer** and **descriptive questions**. This visual feedback makes it easy for the student to quickly understand their strengths and areas where they need improvement.

The **detailed score breakdown** is accompanied by comments or suggestions, providing constructive feedback that can help the student in their future learning endeavors. If the quiz contains **descriptive questions**, the system also gives an analysis of the key terms or concepts identified in the answers and the corresponding score based on predefined criteria.

The grading screen is designed to be both **informative and user-friendly**, offering a seamless experience for students to review their quiz performance without any delays. This feature is particularly beneficial for students who want to track their progress over time and identify areas they can work on for improvement.

Acceptance Criteria:

- The system should grade MCQs, true/false, and short questions automatically in under 2-3 seconds.
- The system should evaluate descriptive answers based on predefined keywords and context with at least 80% accuracy.
- The graded quizzes should include a detailed performance report, highlighting the score and areas for improvement.
- Users should be able to see a visual representation of their quiz score.

Failure Conditions:

- If grading fails due to system errors, the user should be informed with an error message like "Grading Failed. Please Try Again."
- If the system’s evaluation of descriptive answers does not meet 80% accuracy, it should prompt the user to review the criteria or consider revising the grading system.
- If there’s an issue with rendering the performance report, the system should notify the user and provide alternative options like retrying or downloading the results.
- If the grading process takes longer than 3 seconds for MCQs or true/false questions, the system should notify the user of a performance issue.

User Story 6

Feedback and Recommendation Module

The **Feedback and Recommendation Module** is designed to help students monitor their academic progress and identify areas for improvement. After completing a quiz, the system generates a **detailed performance summary**, which includes key metrics such as **accuracy**, **speed**, and areas that need further attention. This feedback allows students to understand not only how many questions they answered correctly, but also how quickly they completed the quiz, and where they might need more practice.

The performance report is generated within **5-10 seconds** after the quiz evaluation, ensuring a fast response time. The system highlights areas where the student performed well and also flags areas where their performance was lacking, providing a clear view of their strengths and weaknesses. For instance, if a student struggled with a particular topic or question type, it would be explicitly noted in the report.

To make the feedback more actionable, the system includes **progress charts** that visually track the student's performance over time. These charts are interactive and allow students to easily compare their progress across different quizzes or topics, helping them see trends in their learning journey. For example, if a student's accuracy has been improving steadily in **MCQs** but needs improvement in **descriptive answers**, they can quickly see these insights through a clear visual interface.

In addition to performance summaries and progress tracking, the system provides **personalized recommendations** based on the student's performance. If the system detects that a student's accuracy is below **60%** in any section or topic, it will suggest targeted areas for improvement. These recommendations may include **study resources**, specific **practice quizzes**, or tips on how to better prepare for future quizzes, giving students a roadmap to enhance their learning and achieve better results.

Overall, this module empowers students with the tools and insights they need to continuously improve and tailor their study habits based on real-time feedback.

Screens:

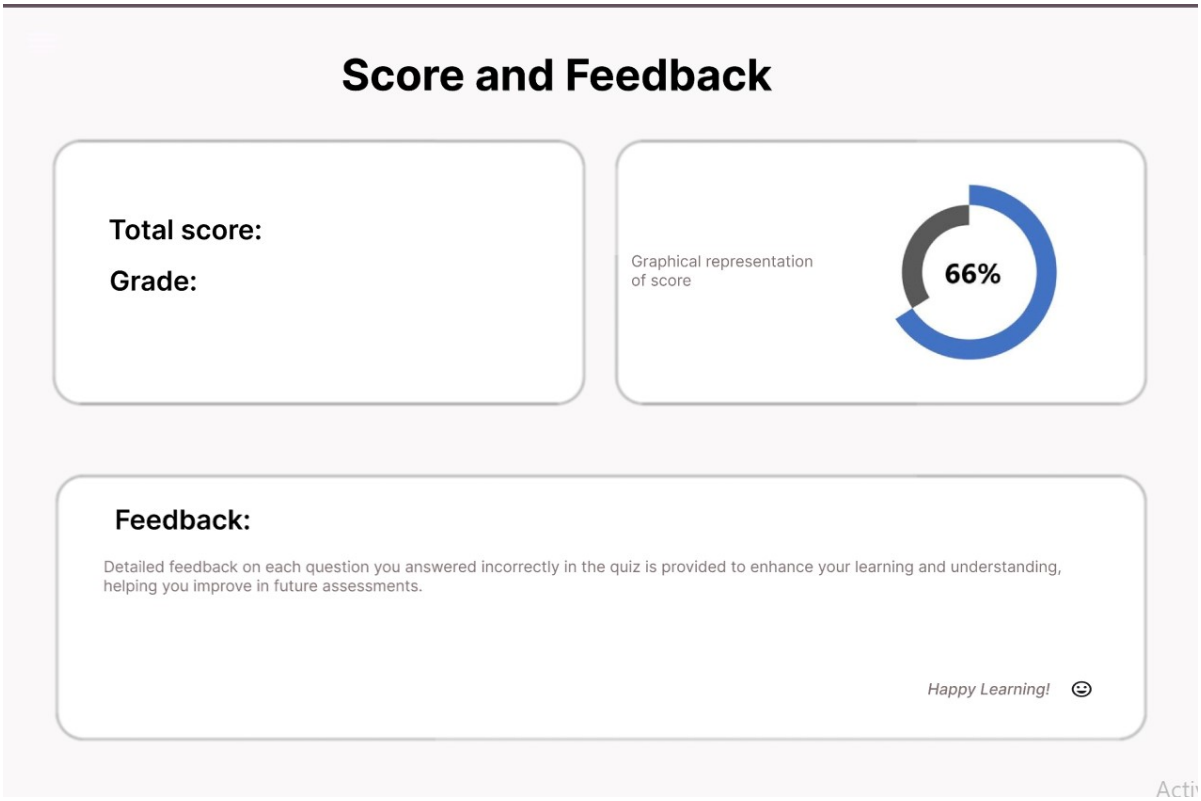


Fig 6.1: Feedback and recommendation

After completing a quiz, the student will be directed to the **Grading Module**, where the results of the quiz are presented in a detailed and easy-to-understand format. Upon landing on the grading screen, the student is immediately greeted with a **total score** feedback, showing the overall performance on the quiz. This score provides a clear picture of how well the student performed, summarizing the results in a single, easily interpretable number.

In addition to the score, the grading module includes a **visual representation** of the student's performance, such as bar charts or pie graphs, allowing them to instantly grasp key insights into their results. These visual elements help break down the quiz performance into specific categories, such as accuracy, speed, and subject/topic performance. For example, a bar chart could show the number of correct versus incorrect answers across different question types, such as MCQs, true/false, and short questions. These visuals offer a quick and interactive way for students to see how they performed in various sections of the quiz.

But the grading screen doesn't stop at just showing the total score and visual feedback. The system will also link this data to the **Feedback and Recommendation Module**, offering detailed analysis based on the student's performance. The student will see a breakdown of their

strengths and weaknesses, and the system will suggest areas for improvement based on their quiz results. If a student scored poorly on specific topics, they might receive recommendations for additional study materials, practice quizzes, or tips to improve their performance.
For example, if a student struggled with multiple-choice questions, the system might recommend reviewing certain topics or taking more quizzes of the same type. If their speed was slower than average, the system might suggest practicing timed quizzes or taking shorter quizzes to improve their response time.

All of this is presented in a seamless and interactive format, ensuring that students are not only aware of their score but are also given clear direction on how to improve. The grading screen becomes not only a place for results but also a launchpad for **personalized feedback** and **continuous improvement**, encouraging students to track their progress over time and stay motivated to reach their learning goals.

This integrated approach allows the student to not only review their performance but also gain actionable insights that can guide their study strategies moving forward. The system aims to support each student's unique learning journey by offering data-driven recommendations based on real-time performance, making it a truly personalized educational experience.

Acceptance Criteria:

- After quiz completion and evaluation, the system should generate detailed performance reports, including accuracy, speed, and areas for improvement.
- The performance reports should be generated in under 5-10 seconds.
- The system should display progress over time using graphs and visualizations.
- The system should suggest areas where improvement is needed (e.g., topics with accuracy below 60%).

Failure Conditions:

- If the report generation fails, the system should display a message like "Error Generating Feedback" and provide the user with options to retry or contact support.
- If the system encounters a problem with generating graphs or visual representations, it should alert the user and offer an alternative display method.
- If the recommendation engine fails to generate suggestions for areas of improvement, the system should suggest a general "Review All Topics" recommendation until the issue is resolved.

User Story 7

Download Quiz Module

In this **Download Quiz Module**, both teachers and students are provided with the ability to download the quizzes they have generated, offering a convenient way to access quizzes offline and share them for future use.

Teacher's Perspective:
As a teacher, once the quiz is generated, they will be presented with an option to download the quiz in multiple formats, such as **PDF** or **DOCX**. After selecting the preferred format, the system will ensure the quiz layout is clean, neat, and printable, making it ready to be used in an offline setting or shared with students. Teachers can quickly download and distribute the quiz without any technical issues or formatting errors. This ensures a smooth transition from online quiz creation to offline usage.

The system ensures that all elements of the quiz — from the questions to any accompanying instructions — are properly formatted, with appropriate spacing and alignment. Whether the teacher intends to print out physical copies of the quiz or distribute it digitally to students, the downloaded file will maintain the same high-quality formatting as displayed in the online interface.

Student's Perspective:
For students, once the quiz has been generated, they too will have the option to download the quiz for future reference or offline use. This feature is particularly useful for students who prefer to work offline, need to review the quiz later, or wish to keep a copy of the quiz for further study. The system will ensure that the layout remains neat and structured, making it easy for students to read, answer, and refer to the quiz whenever necessary.

The system also guarantees that the downloaded quizzes will maintain consistent formatting across multiple download formats like **PDF** and **DOCX**. Whether they are reviewing the quiz questions later on or sharing it with their peers, the downloaded version will be as user-friendly and readable as the one presented in the system.

Download Process:
Once the teacher or student requests to download the quiz, the system will process the request and generate the file in the selected format within **1-3 seconds**. The downloaded file will include all the related questions, formatted clearly and correctly. The download process is efficient, ensuring that the user can access their quiz with minimal delay. Whether the format is for print or for future digital review, the system guarantees that the quiz will be properly structured, with all elements included in the final file.

This feature enhances the flexibility and usability of the quiz generation system, allowing both teachers and students to have control over how they use and distribute the quizzes. By offering

the ability to download quizzes in commonly used file formats, the system helps ensure that both students and teachers have a smooth and seamless experience with the tool, whether they are online or offline.

Screens:

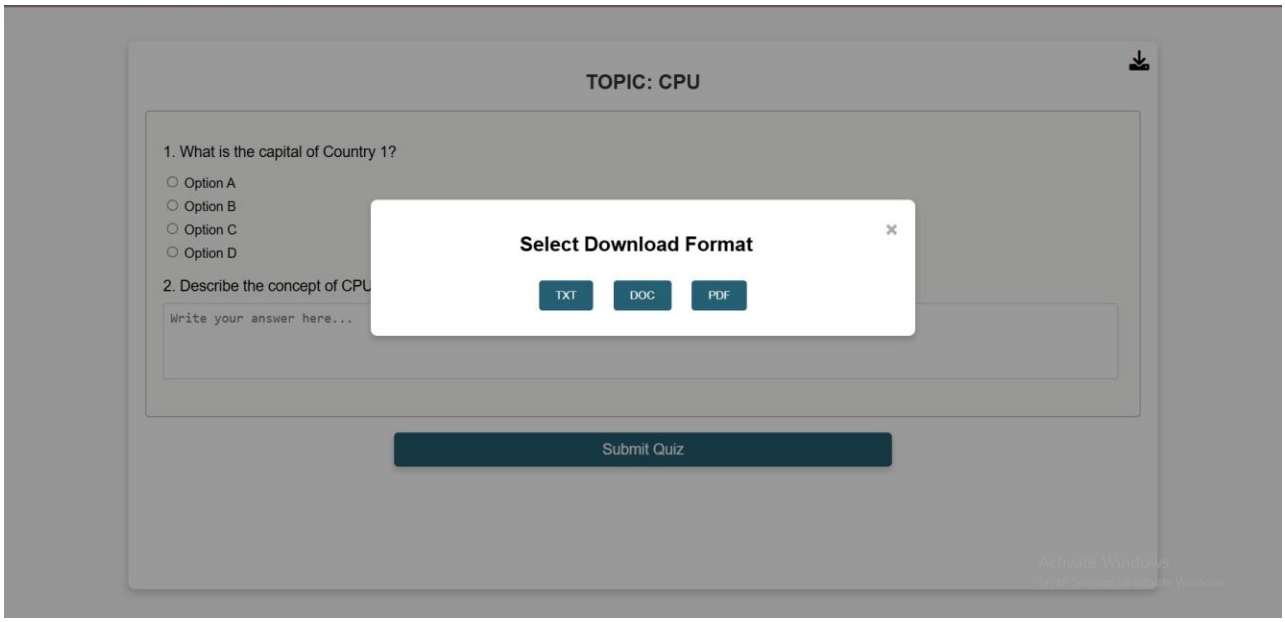


Fig 7.1: Select Download format

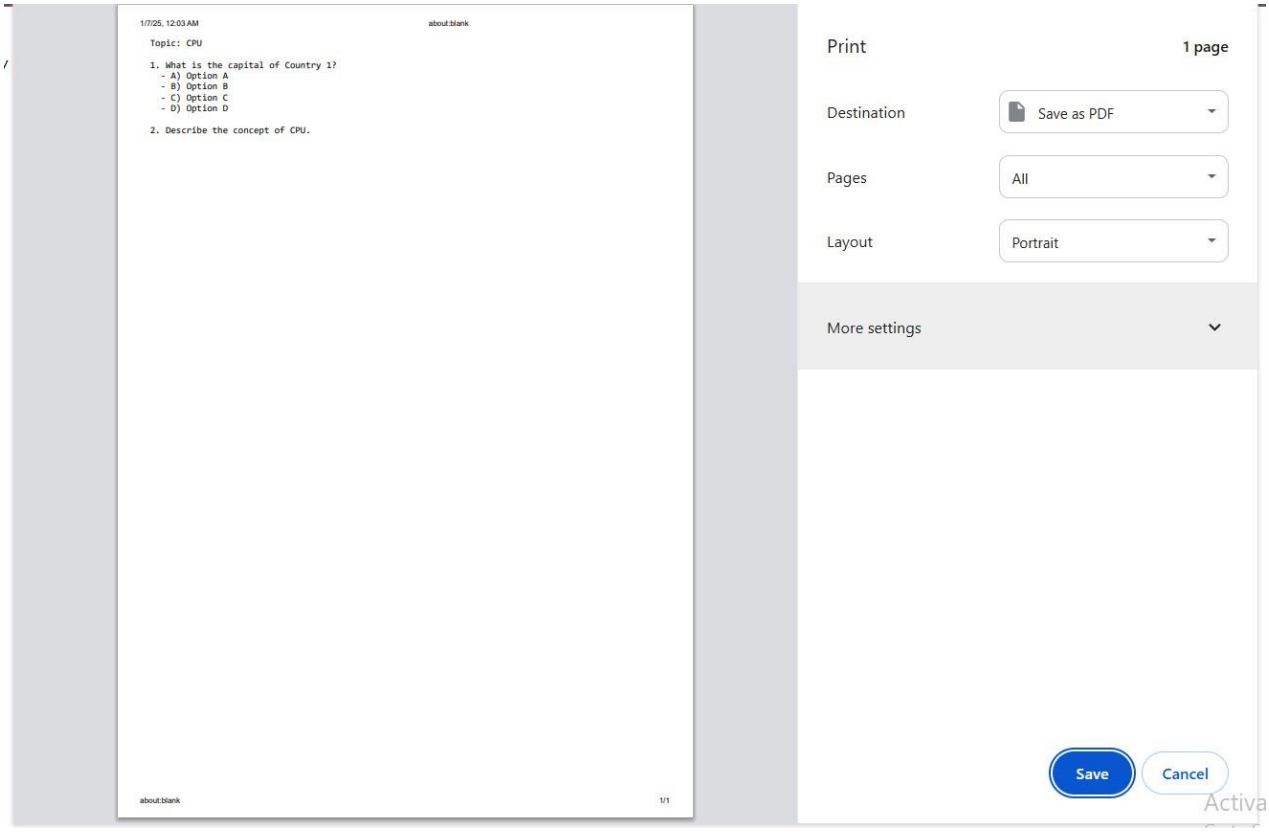


Fig 7.2: Details needed to downloading quiz

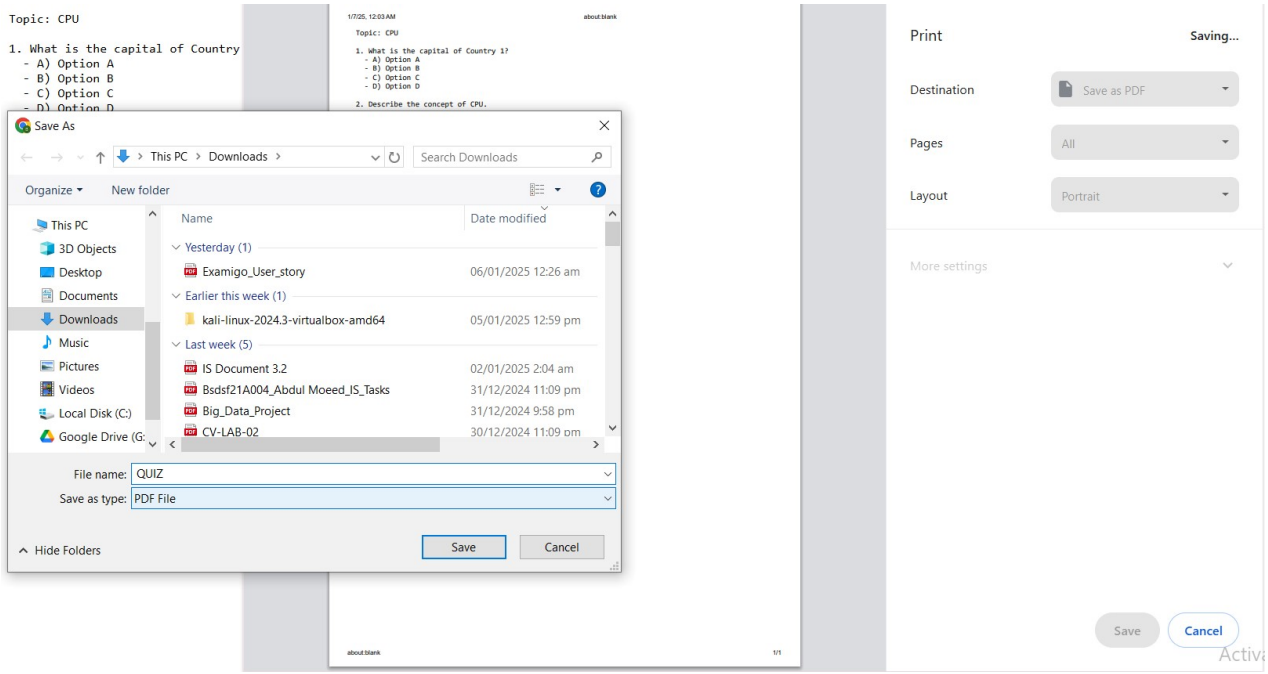


Fig 7.3: Choose Download location

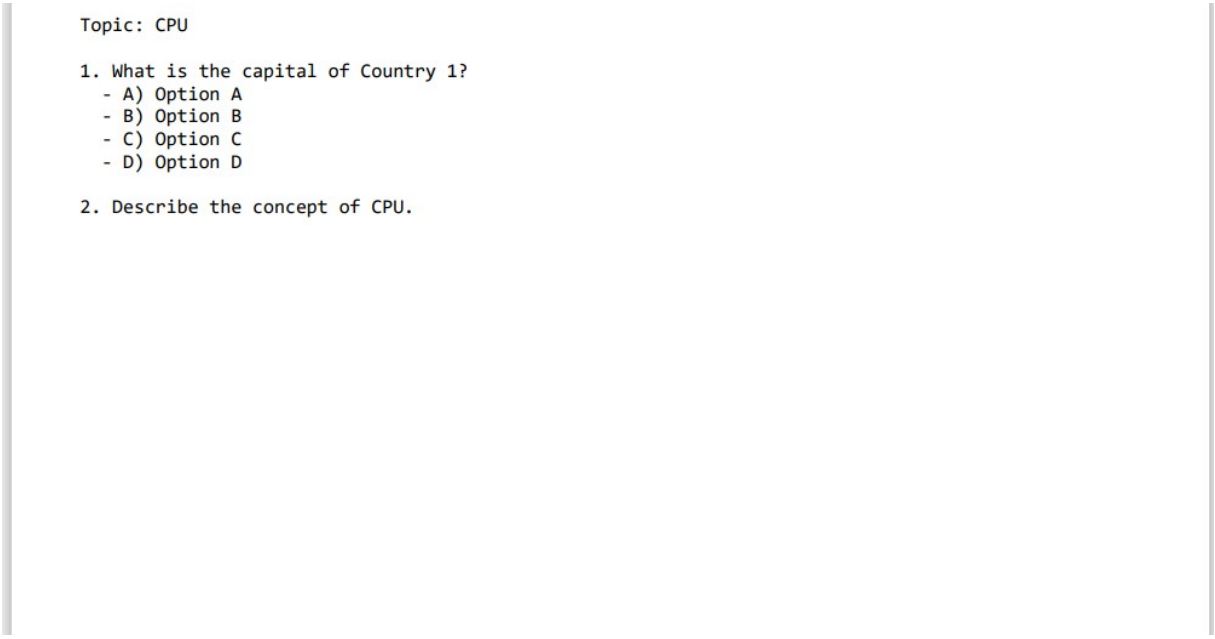


Fig 7.4: Downloaded Quiz

When a user (either a student or teacher) decides to download a quiz, the process begins when they click on the **download icon** displayed on the screen after quiz generation. Upon clicking the icon, a modal will pop up, prompting the user to choose the **download format** they prefer.

The available formats are **PDF**, **DOC**, and **TXT**, offering flexibility depending on the user's needs — whether they require a file for printing, further editing, or simply viewing in a text format.

Once the user selects the desired format, the modal will then proceed to ask for further details regarding **where** they want to save the downloaded file on their system. The modal will include a file path field, enabling the user to either accept the default download location or choose a different folder or directory within their local system.

The process flow for this download feature is as follows:

1. **Click on the Download Icon**: The user clicks the download icon after the quiz has been generated, initiating the download process.

2. **Modal Pop-Up**: A modal appears on the screen, displaying options for the user to select the format in which they wish to download the quiz. The available options are **PDF**, **DOC**, and **TXT**. Each format corresponds to a different use case:
    - **PDF** for neat, printable layouts. ○ **DOC** for further editing or customizations.
    - **TXT** for simple text-based storage or review.

3. **Choose Download Location**: After selecting the format, the modal will prompt the user to specify the location where they would like to save the quiz file on their system. A file explorer dialog may appear, allowing the user to navigate to their preferred folder or directory, or they can simply confirm the default download location.

4. **Download File**: Once the user has made the format and location selections, they click the **Download** button. The system will process the request and generate the quiz in the chosen format. The file will be saved to the designated location within **1-3 seconds**, and the user will be notified of a successful download.

By implementing this process in the user interface, the system ensures that both students and teachers have complete control over the format and location of the files they download. This feature enhances the user experience, providing flexibility and convenience for managing quizzes offline. The clear, step-by-step flow allows users to easily download quizzes in the format they prefer, whether for later use, printing, or editing.

Acceptance Criteria:

- The system should allow users to download generated quizzes in multiple formats (PDF, DOCX, TXT).
- The download process should be completed within 1-3 seconds.
- The downloaded quiz should maintain proper formatting (questions, answers, sections).
- The user should be able to choose the location on their system to save the downloaded file.

Failure Conditions:

- If the download format is not supported, the system should display an error message.
- If the user cannot download the quiz (due to browser issues, file permission errors, etc.), the system should notify the user with an appropriate error message like "Download Failed."
- If the file is downloaded but the formatting is incorrect (misaligned text, missing questions), the system should allow the user to report the issue for further refinement.

ERD (Entity-Relationship Diagram) Structure:

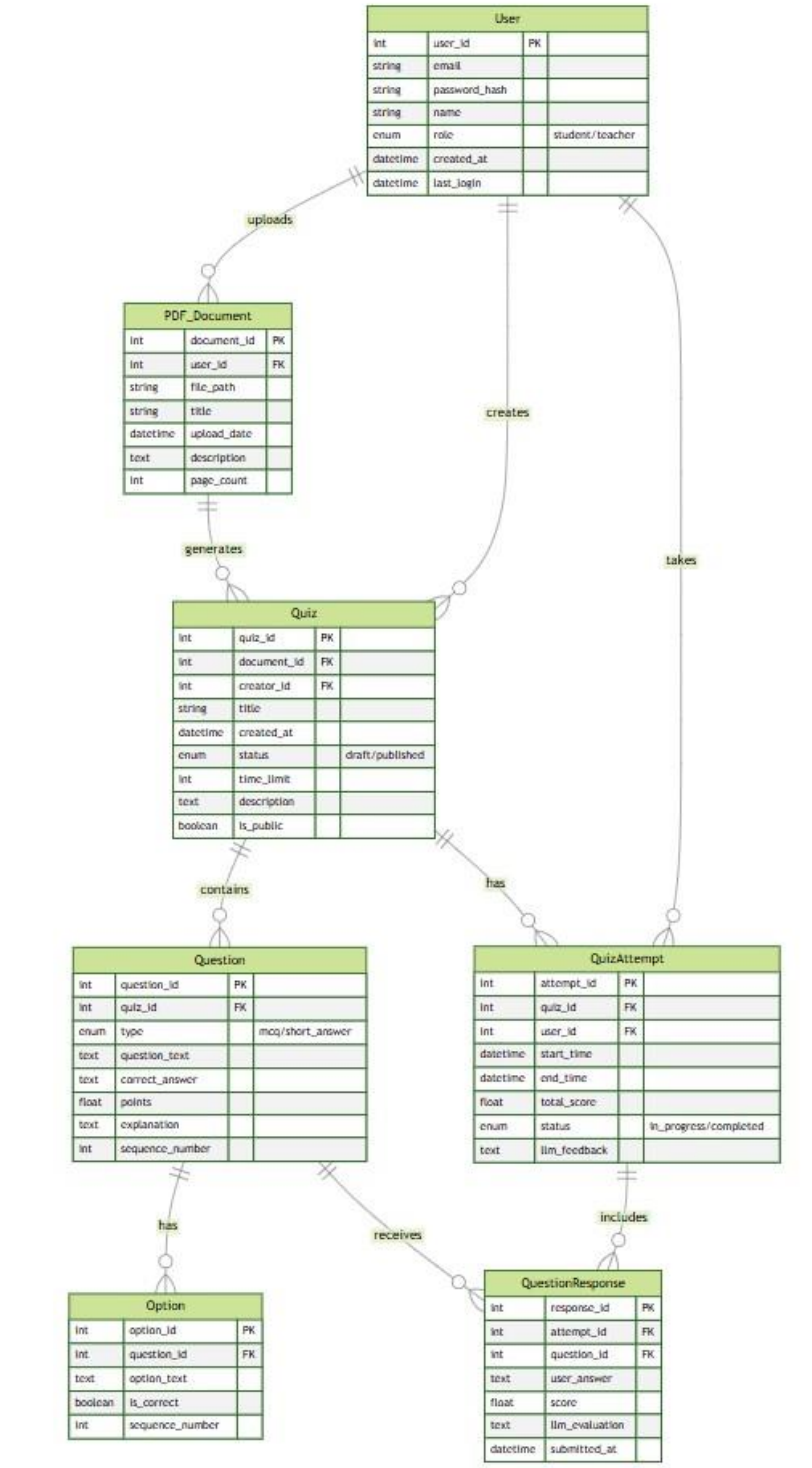


Fig: Database Design for ExAmigo

ERD (Entity-Relationship Diagram) Structure

This design represents a **structured and scalable** way to manage document-driven quiz creation, automatic grading, and detailed feedback. The relationships between entities (e.g., **Quiz** having multiple **Questions**, **QuizAttempt** containing **QuestionResponse**) ensure that all aspects of quiz generation, attempt tracking, and grading are properly handled within the database.

Database Design for ExAmigo

ExaAmigo is designed to automate the process of quiz creation, attempt tracking, grading, and providing feedback to students. It leverages uploaded PDF documents to generate quizzes, evaluates student responses, and provides feedback. The database is structured to support these functionalities, and the entities outlined below play a vital role in making the system scalable, efficient, and user-friendly.

Entities and Relationships

1. User Entity

The **User** entity represents both **students** and **teachers** who interact with the system. Teachers upload PDF documents, create quizzes, and grade quizzes, while students take quizzes and track their performance.

Attributes:

- **user_id (PK):** A unique identifier for each user in the system.
- **email:** The email address used for communication and login.
- **password_hash:** A securely stored hashed password for authentication.
- **name:** Full name of the user (used to personalize the experience).
- **role:** Defines whether the user is a **"student"** or **"teacher"**. This ensures role-based access to system features.
- **created_at:** Timestamp indicating when the account was created.
- **last_login:** Timestamp indicating the last time the user logged into the system.

Relationships:

- A **User** can upload multiple **PDF_Document** entries (if they are a teacher).
- A **User** (either student or teacher) can create multiple **Quiz** entries.
- A **User** (student) can take multiple **QuizAttempt** entries.

2. PDF_Document Entity

The **PDF_Document** entity stores the details of PDF files uploaded by **teachers** for quiz generation. These documents serve as the base for creating quizzes, as the system extracts relevant content for question generation.

Attributes:

- **document_id (PK):** A unique identifier for each document uploaded.
- **user_id (FK):** References the **user_id** of the teacher who uploaded the document.
- **file_path:** The path on the server where the document is stored.
- **title:** The title of the document, helping users identify the document.
- **upload_date:** The date and time the document was uploaded.
- **description:** A short description or summary of the document.
- **page_count:** The number of pages in the document, used to help with document management.

Relationships:

- A **PDF_Document** can be used to generate multiple **Quiz** entries.

3. Quiz Entity

The **Quiz** entity stores details about the quizzes that are created by **teachers** from the uploaded **PDF_Document** files. Each quiz is based on a specific document and can have a collection of questions, settings, and results.

Attributes:

- **quiz_id (PK):** A unique identifier for each quiz created.
- **document_id (FK):** References the **document_id** of the **PDF_Document** that the quiz is based on.
- **creator_id (FK):** References the **user_id** of the teacher who created the quiz.
- **title:** The title of the quiz.
- **created_at:** Timestamp of when the quiz was created.
- **status:** Defines whether the quiz is in a **"draft"** state or **"published"** and ready for students to take.
- **time_limit:** The time limit (in minutes) set for completing the quiz.
- **description:** A brief description or instructions for the quiz.
- **is_public:** A flag to indicate whether the quiz is accessible by all students or restricted to certain groups.

Relationships:

- A **Quiz** contains multiple **Question** entries.
- A **Quiz** can have multiple **QuizAttempt** entries associated with students.

4. Question Entity

The **Question** entity defines the individual questions that are included in a **Quiz**. Each question has an associated type (e.g., multiple-choice or short answer), and correct answers are also stored for grading.

Attributes:

- **question_id (PK):** A unique identifier for each question.
- **quiz_id (FK):** References the **quiz_id** of the related quiz.
- **type:** Specifies the type of the question (e.g., **"mcq"** for multiple-choice or **"short_answer"** for short-answer questions).
- **question_text:** The actual text of the question that is presented to students.
- **correct_answer:** The correct answer for this question. For MCQs, it's an option, and for short answers, it's a textual response.
- **points:** The number of points awarded for a correct answer.
- **explanation:** Provides an explanation or rationale for why the correct answer is correct.
- **sequence_number:** The order in which the question appears within the quiz.

Relationships:

- A **Question** has multiple **Option** entries for MCQs.
- A **Question** receives multiple **QuestionResponse** entries, which store the student's responses.

5. Option Entity

The **Option** entity is used to store the different answer choices for **MCQ** (Multiple Choice Questions). Each MCQ can have multiple options, one of which is correct.

Attributes:

- **option_id (PK):** A unique identifier for each option.
- **question_id (FK):** References the **question_id** of the associated **Question**.
- **option_text:** The text of the option.
- **is_correct:** A boolean flag indicating if the option is correct (true/false).
- **sequence_number:** The order in which the option appears in the list of answer choices for the question.

Relationships:

- Each **Option** is associated with a single **Question**, specifically MCQs.

6. QuizAttempt Entity

The **QuizAttempt** entity tracks each time a **student** attempts to complete a **Quiz**. This entity helps to record the attempt details such as the time taken and the final score.

Attributes:

- **attempt_id (PK):** A unique identifier for each quiz attempt.
- **quiz_id (FK):** References the **quiz_id** of the quiz being attempted.
- **user_id (FK):** References the **user_id** of the student who is taking the quiz.
- **start_time:** The timestamp indicating when the quiz attempt started.
- **end_time:** The timestamp indicating when the quiz attempt ended.
- **total_score:** The total score that the student earned on the quiz.
- **status:** The status of the attempt (**"in_progress"** or **"completed"**).
- **llm_feedback:** Feedback provided by the LLM (Language Model) after evaluating the student's responses.

Relationships:

- A **QuizAttempt** includes multiple **QuestionResponse** entries, which store the student's responses to individual questions in the quiz.

7. QuestionResponse Entity

The **QuestionResponse** entity stores the responses submitted by students during their quiz attempts. It allows for tracking of individual question answers, scores, and LLM-based feedback.

Attributes:

- **response_id (PK):** A unique identifier for each response.
- **attempt_id (FK):** References the **attempt_id** of the associated quiz attempt.
- **question_id (FK):** References the **question_id** of the question being answered.
- **user_answer:** The answer provided by the student.
- **score:** The score awarded for the response based on correctness.
- **llm_evaluation:** Evaluation or feedback provided by the LLM (Language Model) on the student's answer.
- **submitted_at:** The timestamp of when the response was submitted.

Relationships:

- A **QuestionResponse** belongs to one **QuizAttempt** and one **Question**.
- A **QuestionResponse** stores a student's answer for a given **Question**.

Technologies Used in ExAmigo

The ExAmigo relies on a variety of technologies for different components of the system, ranging from database management to document processing, quiz generation, and the use of large language models (LLMs) for grading and feedback.

1. Database Technology

Relational Database Management System (RDBMS):

- **Technology**: MySQL, PostgreSQL, or SQLite
- **Purpose**: The relational database stores structured data, such as user profiles, documents, quizzes, questions, answers, and quiz attempts.

Advantages:

- **Structured data**: Relational models with tables and foreign keys make it easier to enforce relationships (e.g., users to quizzes, quizzes to questions).
- **ACID compliance**: Ensures consistency and reliability during database operations, such as quiz attempts and grading.
- **SQL queries**: Ability to retrieve, insert, update, and delete records with complex queries and joins for various operations.

Why not NoSQL:

- **NoSQL databases** (e.g., MongoDB) would not be ideal for this project since the relationships are predefined and structured (e.g., users taking quizzes, questions within quizzes). An RDBMS is a better fit for scenarios requiring complex joins and transactional integrity.

2. Document Processing & Upload

Technology:

- **PDF Processing**: Python libraries like **PyPDF2**, **pdfminer.six**, or **PyMuPDF** can be used to extract text from uploaded PDF documents for quiz generation.

- **File Storage**:
    - **Local Storage**: Saving PDFs on a local file server.
    - **Cloud Storage**: Using **Amazon S3**, **Google Cloud Storage**, or **Azure Blob Storage** for scalable document storage.
    - **Database**: Store file metadata (e.g., title, description, upload date) and file path, while the actual PDF is stored separately.

Why not Use:

- **Traditional File Systems** for handling large volumes of documents can lead to management issues. Cloud services provide more flexibility in scalability and security.

3. Web Development (User Interface)

Technology:

- **Frontend**:
    - **HTML/CSS/JavaScript** for creating the structure and style of the pages.
    - **React.js** or **Vue.js**: For building dynamic user interfaces that interact with backend services for creating quizzes and attempting them.
    - **Bootstrap**: For responsive, mobile-first design.
- **Backend**:
    - **Flask** or **Django** (Python-based): To build the REST APIs or the full-fledged web application. Flask is lighter and easier for smaller apps, while Django provides more built-in features (authentication, admin panel, etc.).
    - **Node.js** with Express: If JavaScript is preferred for backend development.

4. LLMs for Automatic Grading

Technology: Large Language Models (LLMs) like GPT-3/4, BERT, and T5 have revolutionized the way automatic grading and feedback are generated. Here's a comparative analysis:

1. **GPT-3/4 (Generative Pre-trained Transformer) Key**

Features:

- **Capabilities**:
  - Can generate, summarize, or classify text based on context.
  - It is well-suited for grading **short-answer questions**, as it can understand free-form responses, provide feedback, and even generate explanations.
  - Fine-tuned models (like **GPT-4** or specialized variants) can be trained to align with specific subject areas (e.g., Computer Science, English).
- **Integration**:
  - Uses REST APIs provided by OpenAI to integrate into the system. It can process answers and provide evaluation and feedback.
- **Advantages**:
  - **High accuracy in language understanding**. ○ **Generates contextual feedback** based on the content of the student's answer.
  - Highly scalable and able to process large numbers of requests.
- **Limitations**:
  - **Cost**: GPT-3/4 requires API calls, which could be expensive depending on usage.
  - **Longer Responses**: Can sometimes generate lengthy or irrelevant responses, requiring post-processing or filtering. ○ **Bias**: Prone to generating biased responses based on the training data.

Use Case: Ideal for evaluating short-answer or descriptive answers and providing detailed feedback, especially in subjects where language understanding and explanation are critical.

2. **BERT (Bidirectional Encoder Representations from Transformers) Key Features:**

- **Capabilities**:
  - Designed for understanding the context in bidirectional text and is better for tasks like **question answering** or **classification**.

    - Unlike GPT, BERT works by reading the entire sequence of words in both directions, which helps in understanding context better.
- **Integration**: Available through platforms like **Hugging Face** for easy access and fine-tuning.

- **Advantages**:
    - Best suited for **discrete question-answering tasks** where the goal is to match answers to predefined solutions (like multiple-choice or factual short-answer).
    - More efficient in resource usage for classification tasks.
- **Limitations**:
    - Not suitable for generating long-form feedback or explanations.
    - Less effective for generating human-like, contextual feedback in descriptive answers compared to GPT.

Use Case: Ideal for **MCQs** or factual answers where evaluation is more about matching the response to a correct or incorrect option.

3. T5 (Text-to-Text Transfer Transformer) Key

Features:

- **Capabilities**:
    - T5 treats every NLP task as a text-to-text task (e.g., summarization, translation, question answering, etc.).
    - It's well-suited for tasks that involve transforming one type of text into another, such as transforming an answer into a grade or feedback.

- **Advantages**:
    - Extremely flexible and can be adapted for various NLP tasks with minimal modification.
    - Strong performance on tasks like **question answering**, **summarization**, and **text generation**.

- **Limitations**:
    - **Larger Models** can be computationally expensive.
    - **Fine-tuning required**: Like GPT, T5 models need to be fine-tuned on your specific dataset to get optimal results.

Use Case: Useful when there's a need for **text transformation tasks** (e.g., converting answers into grades, summarizing or simplifying answers).

5. Comparative Analysis:

Model	Use Case	Strengths	Limitations	Recommended Use
GPT-3/4	Short-answer grading, feedback generation, content creation	Natural language understanding, context-based feedback	Expensive, may generate irrelevant text	Descriptive answers, feedback generation
BERT	MCQ and fact-based question answering	High accuracy on classification tasks	Does not generate explanations	MCQs, short-answer with factual responses
T5	Text transformation (grading, summarization)	Very versatile, strong on various NLP tasks	Needs fine-tuning, computationally heavy	Grading, transforming answers into grades

6. Other Technologies:

Technology:

- **Authentication and Authorization**: OAuth, JWT for secure user login and role-based access (students vs. teachers).
- **Containerization and Deployment**:
    - **Docker**: For isolating components and ensuring consistency in environments.
    - **Kubernetes**: For orchestration, especially when scaling services.
- **CI/CD Tools**: Jenkins, GitLab CI for automating testing and deployment pipelines.
- **Cloud Infrastructure**: AWS, GCP, or Azure to scale the system in the cloud for handling large-scale operations (e.g., document uploads, quiz creation).

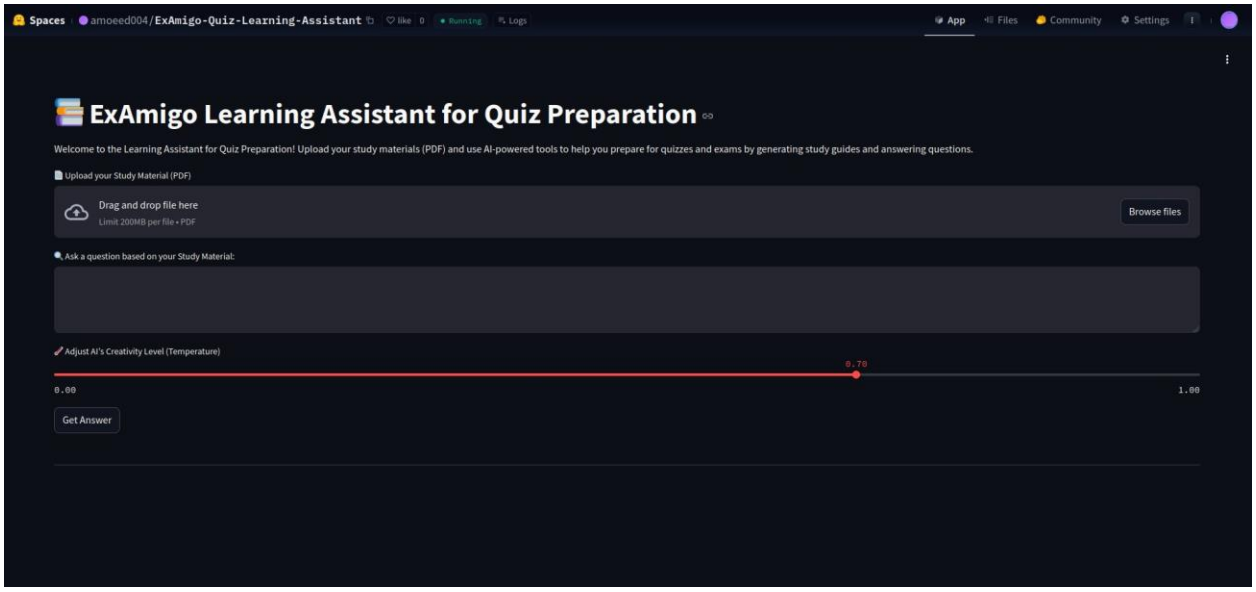
Minimum Viable Product (MVP)

1. Chat Bot for Question Preparation

The Chat Bot for Question Preparation is an AI-powered learning assistant designed to help students efficiently prepare for exams by providing tailored guidance on question preparation. Unlike traditional question generators, this chatbot focuses on assisting students in understanding key topics, answering queries, and reviewing important concepts based on uploaded study materials. The chatbot engages users in real-time, helping them explore relevant questions and review critical points for enhanced retention and understanding.

Deployment: The chatbot has been deployed on Hugging Face as part of the Examigo Learning platform, offering seamless interaction and instant feedback. With its user-friendly interface, students can access personalized assistance, ensuring a dynamic and supportive study experience.

Below is the attachment of MVP:



2. Chat Bot for MCQs Preparation

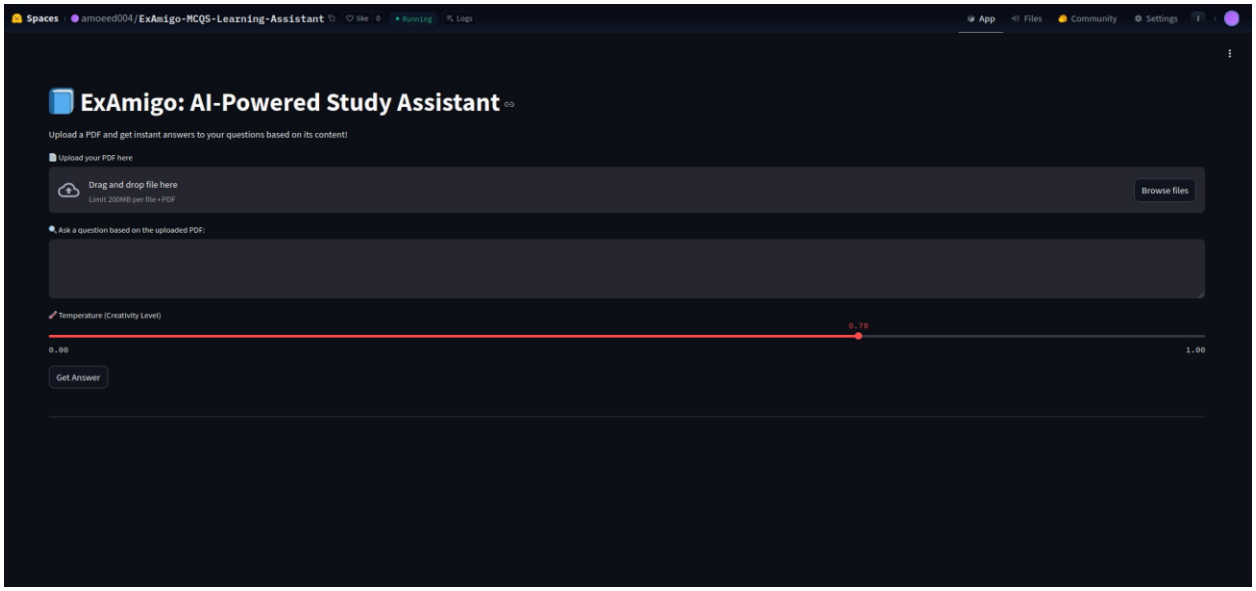
Chat Bot for Question Preparation

The Chat Bot for Question Preparation is an AI-powered learning assistant designed to help students efficiently prepare for exams by providing tailored guidance on question preparation. Unlike traditional question generators, this chatbot focuses on assisting students in understanding key topics, answering queries, and reviewing important concepts based on

uploaded study materials. The chatbot engages users in real-time, helping them explore relevant questions and review critical points for enhanced retention and understanding.

Deployment: The chatbot has been deployed on Hugging Face as part of the Examigo Learning platform, offering seamless interaction and instant feedback. With its user-friendly interface, students can access personalized assistance, ensuring a dynamic and supportive study experience.

Below is the attachment of MVP:



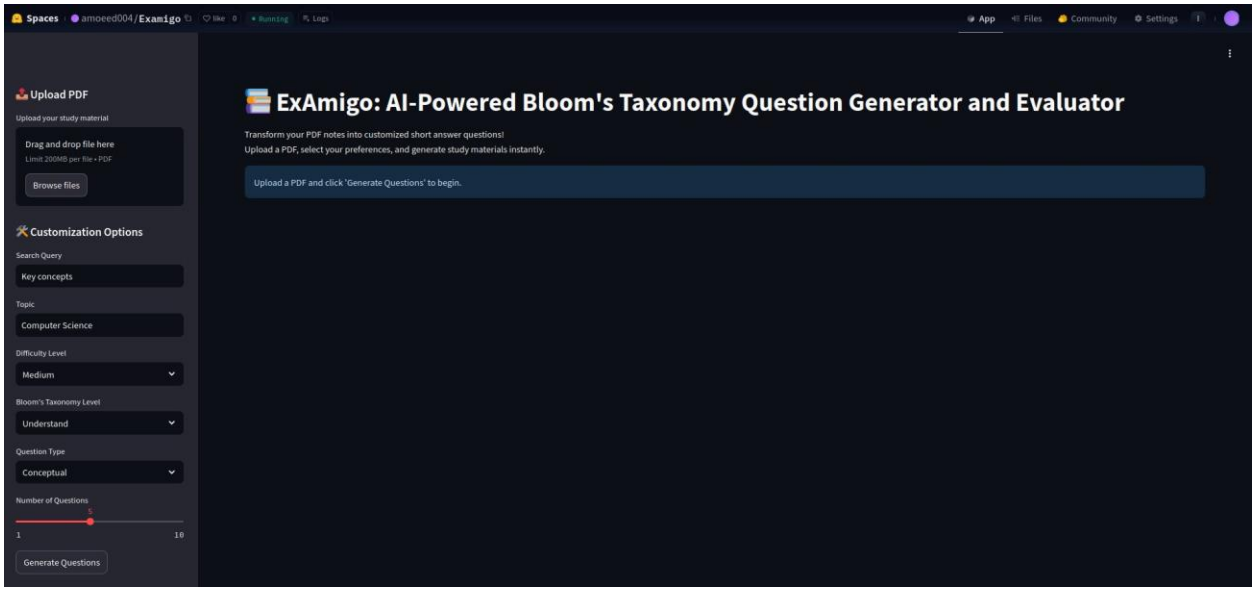
3. Question Generator

The Question Generator is an advanced AI tool designed to assist students in preparing for exams by automatically generating a variety of questions based on uploaded study materials. Leveraging sophisticated natural language processing, short-answer questions aligned with the content, helping students test their knowledge and reinforce their understanding. This tool adapts to different topics and formats, making it a versatile study companion.

Deployment: The Question Generator has been deployed on Hugging Face as part of the Examigo Learning platform, providing a scalable and accessible solution for students seeking a

structured approach to exam preparation. Through this deployment, users can generate personalized quizzes that cater to their specific learning needs.

Below is the attachment of MVP:

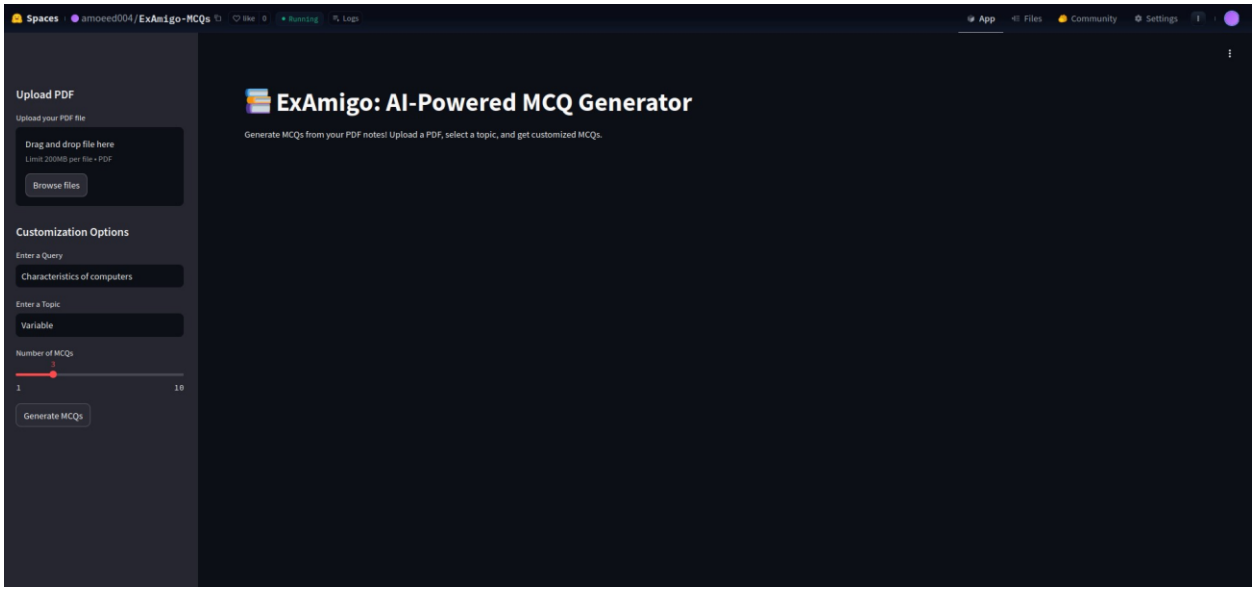


4. MCQs Generator

The MCQs Generator is an AI-powered tool designed to help students enhance their exam preparation by automatically generating multiple-choice questions (MCQs) based on the content of uploaded study materials. With the ability to generate MCQs in varying difficulty levels, this tool enables students to test their knowledge, identify weak areas, and improve their understanding of key concepts. It provides a dynamic way for learners to practice and evaluate their readiness for exams.

Deployment: The MCQs Generator is deployed on Hugging Face as part of the Examigo Learning platform, offering a convenient and scalable solution for generating customized MCQs. This deployment allows students to instantly create quizzes tailored to their learning objectives, promoting effective and focused revision.

Below is the attachment of MVP:



Documentation of MVP

Examigo: AI-Powered Educational Assessment Generator

Overview:

Examigo is a comprehensive Streamlit-based web application that harnesses artificial intelligence to generate and evaluate educational assessments. The system provides two primary assessment types: Multiple Choice Questions (MCQs) and Short Answer Questions based on Bloom's Taxonomy. By processing PDF documents, Examigo creates customized questions and provides interactive interfaces for both question generation and answer evaluation.

Features

The application provides a rich set of features for educational assessment:

Multiple Choice Questions (MCQs)

The MCQ generation system creates structured multiple-choice questions with four options each. These questions are automatically generated from uploaded PDF content and include correct answer identification. The system processes the document context to ensure questions are relevant and accurately reflect the source material.

Short Answer Questions

The short answer component generates questions based on Bloom's

Taxonomy levels, allowing for deeper assessment of student understanding. These questions span various cognitive levels from basic recall to complex analysis and evaluation.

Common Features

Both assessment types share sophisticated document processing capabilities and an intuitive user interface. The system includes PDF processing, context-aware question generation, and session management for a seamless user experience.

Technical Architecture

Core Components

The application is built on several key technological components:

1. **Document Processing Engine** The system processes documents through multiple stages:

- PDF loading and parsing through PyPDFLoader
- Intelligent document chunking using **RecursiveCharacterTextSplitter**
- Vector embedding creation with HuggingFaceEmbeddings
- Similarity search implementation using FAISS

2. **Question Generation System** The question generation pipeline includes:

- Integration with OpenAI's GPT models for content generation
- Context-aware question formulation using vector similarity search
- Structured output formatting for both MCQs and short answer questions
- Automatic answer generation for evaluation purposes

3. **User Interface Layer** The interface provides:

- PDF document upload functionality
- Customizable question generation parameters
- Interactive quiz taking experience
- Real-time answer evaluation and feedback

Configuration

Environment Variables

The system requires proper configuration of: - OPENAI_API_KEY: Authentication key for OpenAI API access

System Constants

The application uses carefully tuned constants for optimal performance:

- CHUNK_SIZE = 500 # Characters per document chunk
- CHUNK_OVERLAP = 50 # Overlap between consecutive chunks
- MODEL_NAME = "all-MiniLM-L6-v2" # Embedding model
- OPENAI_MODEL = "gpt-3.5-turbo"# Language model
- TEMPERATURE = 0.7 # Creative variation in generation

Core Functions

Document Processing

The document processing pipeline consists of several specialized functions

load_pdf(pdf_path)

Handles PDF document loading and initial parsing.

Parameters:

- pdf_path (str): File system path to the PDF document **Returns:** List of processed

document objects containing parsed content **split_documents(documents, chunk_size,**

chunk_overlap) Performs intelligent document chunking for optimal processing.

Parameters:

- documents (list): Document objects to be processed
- chunk_size (int): Character length of each chunk
- chunk_overlap (int): Overlap between chunks

Returns: List of Document objects representing processed chunks

Question Generation

The system provides two distinct question generation pathways:

MCQ Generation generate_mcqs

(llm, relevant_chunks, topic, num_mcqs)

Creates multiple choice questions with four options each.

Parameters:

- llm: Language model instance
- relevant_chunks: Context chunks from the document
- topic (str): Subject area for questions
- num_mcqs (int): Number of questions to generate **Returns:** Formatted text

containing structured MCQs **parse_mcqs(mcqs_text)**

Processes generated MCQ text into structured format.

Parameters:

- mcqs_text (str): Raw generated MCQ content

Returns: List of dictionaries containing parsed MCQs

Short Answer Generation

The short answer generation system incorporates Bloom's Taxonomy levels and provides detailed evaluation capabilities.

Vector Store Operations create_vector_store(chunks)

Creates searchable vector representations of document content.

Parameters: chunks: Processed document chunks

Returns: FAISS vector store instance

retrieve_chunks(vector_store, query, k=3)

Performs similarity search for relevant content.

Parameters: vector_store: FAISS instance - query (str): Search query - k (int): Number of results to retrieve **User Interface**

MCQ Interface

The MCQ interface provides: PDF upload functionality - Topic and query specification Number of questions selection - Interactive question display -Automatic scoring and feedback

Session Management

The application maintains several session states: mcqs: Stores generated multiple choice questions - answers: Tracks user responses - quiz_started: Manages quiz state uploaded_file_path: Tracks document location

Error Handling

The system implements comprehensive error handling for: - PDF processing errors - Question generation failures - Parsing issues - Session state management - User input validation

Dependencies

The application relies on several key libraries:

- streamlit: Web interface framework - langchain: Language model integration
- PyPDF: PDF processing
- FAISS: Vector similarity search
- HuggingFace Transformers: Text embeddings
- OpenAI GPT: Question generation
- python-dotenv: Environment management
- re: Regular expression processing

Usage Instructions

1. Upload PDF through the sidebar interface

2. Configure generation parameters:

- For MCQs: Specify topic and number of questions
- For short answers: Select Bloom's level and difficulty

3. Generate questions using the dedicated button

4. Complete the assessment interactively

5. Submit for immediate feedback and scoring

Performance Considerations

- The system is optimized through:
- Efficient document chunking (500-character chunks with 50-character overlap)
- Minimum of top-3 relevant chunks for context
- Balanced temperature setting (0.7) for reliable generation
- Session state management for smooth user experience

The documentation provides a comprehensive overview of both MCQ and short answer functionality while maintaining clear organization and detailed explanations of each component.