AI -DRIVEN QUIZ APPLICATION WITH VIDEO LEARNING

Swapnil Sanjay Pawar Rohan Jaiswal Priyanka Saxena Shreenath Gandhi

1. Project Overview

This project is designed as a web-based application that facilitates learning by allowing professors to upload educational videos and automatically generate quizzes based on the video content using an AI model. The primary stakeholders of this project are professors and students.

Professors have the ability to upload educational videos, generate quizzes, and view students' results. Students, on the other hand, can watch the videos and take quizzes, with real-time feedback on their performance. This system reduces the workload on educators by automating quiz generation using the Gemini AI model and provides an efficient learning environment for students.

2. Objective

The main objective of this project is to develop a platform where professors can:
- Upload educational content (videos).
- Automatically generate multiple-choice questions (MCQs) from the video content using the Gemini AI model.
- Track student progress and view quiz results.

For students, the platform provides:
- A repository of educational videos for viewing.
- Quizzes generated from the video content to assess understanding.
- Real-time feedback on their performance, including score and answer explanations.

3. Technologies Used

To achieve the project's objectives, the following technologies were used:

- Backend Development (PHP): Handles user authentication (professors and students), video uploads, and quiz display.

- Database (MySQL): Stores user details, video metadata, quizzes, and students' quiz results.
- AI Model (Gemini AI): The core of the project is the AI-powered quiz generation, using the Gemini AI model. This model analyzes the content of the videos and generates quizzes based on it.
- Frontend Development (HTML, CSS, Bootstrap): Provides an intuitive and responsive user interface for both professors and students.
- Flask (Python Framework): Integrates the Python-based Gemini AI model with the PHP-based backend to automate quiz generation.
- XAMPP: A local development environment that includes Apache, MySQL, and PHP, which was used for hosting the application locally during development.

4. System Features

4.1 Professor Section
- Login/Registration: Professors can register and log into the system securely.
- Upload Video Content: Professors can upload educational videos, which are stored in the system along with their metadata.
- AI-Generated Quizzes: After uploading a video, the professor can use the Gemini AI model to generate quizzes automatically. The AI model extracts important information from the video and generates relevant questions, ensuring that the quizzes are dynamic and relevant to the content.
- View Quiz Results: Professors can track the performance of their students by viewing their quiz results. This provides insights into student understanding and allows the professor to offer additional support if needed.

4.2 Student Section
- Login/Registration: Students can create accounts and log into the system to access educational content.
- Video Viewing: Students have access to the videos uploaded by professors. They can browse through available videos and select one to watch.
- Quiz After Viewing: After watching a video, students are required to take a quiz. The quiz is automatically generated using the AI model and assesses the student’s understanding of the video content.
- Real-Time Feedback: Upon quiz submission, students receive immediate feedback on their performance. This includes their score, which questions they answered correctly or incorrectly, and explanations for each answer.

5. Methodology

The development of the project followed a structured methodology to ensure seamless integration of various components such as AI, backend, and frontend systems. Below are the steps followed during the development:

5.1 Requirement Gathering and Analysis
The project began with identifying the core features that would meet the needs of both professors and students. Key functionalities such as video uploads, AI-based quiz generation, and real-time feedback were highlighted as priorities.

5.2 System Design
The system was designed with two main sections: one for professors and one for students. The backend needed to support two user roles and handle video management, quiz generation, and result tracking. The frontend was designed to be user-friendly, ensuring that both students and professors could easily interact with the system.

The system also required an AI model for quiz generation, leading to the integration of the Gemini AI model. This influenced the choice of technologies, especially the use of Flask for the Python-based AI model, alongside PHP for the backend.

5.3 Technology Integration
- Backend (PHP and MySQL): The PHP backend was responsible for user management, video uploads, and data retrieval from the MySQL database. The database stored essential data such as user credentials, video metadata, quizzes, and results.

- AI Integration (Python and Gemini AI): The AI component was built using Python and Flask. When a professor uploads a video, a request is made to the AI model (via Flask), which processes the video content and generates quiz questions. This quiz data is then sent back to the PHP backend for storage in the MySQL database.

- Frontend (HTML, CSS, and Bootstrap): A clean and intuitive user interface was developed using HTML, CSS, and Bootstrap. The frontend allows easy navigation for students and professors, ensuring that the system is user-friendly.

5.4 AI-Generated Quizzes
The core innovation in this project was the use of the Gemini AI model to generate quizzes. When a professor uploads a video, the content is sent to the Gemini AI model, which processes the text and video to extract key points. Based on this, the AI generates a set of multiple-choice questions (MCQs) designed to test a student’s understanding of the content.

This automation reduces the professor's workload, ensuring that quizzes are consistent and aligned with the video material. It also personalizes learning for students, as the AI can generate different sets of questions each time.

5.5 Quiz Result Management
Once a student submits their quiz, the system evaluates their responses against the correct answers and provides real-time feedback. The results, along with detailed explanations for each question, are displayed immediately. This feedback loop is crucial for reinforcing learning and allowing students to understand their mistakes.

On the professor's side, they can view a summary of student performance for each quiz. This allows them to track overall progress and identify students who may need additional support.

6. Deployment

The application was developed in a local environment using XAMPP, which provides an integrated stack of Apache, MySQL, and PHP. Here's how the deployment process was structured:

- Setting Up the Local Environment: XAMPP was used to simulate a production environment. The PHP files were placed in the `htdocs` directory, and the MySQL database was set up using `phpMyAdmin`.

- Database Setup: The database `quiz_app` was created, with tables for users, videos, quizzes, and results. Sample data was inserted to test functionality during the development process.

- Running the AI Model: The Flask-based Python server that handles AI quiz generation was run alongside the PHP backend. API calls were made between the two components to ensure real-time quiz generation.

- Testing: Both sections of the application (professor and student) were rigorously tested to ensure functionality, especially around AI quiz generation and result feedback.

7. Challenges Faced

AI Model Integration-
Understanding how to properly integrate the Gemini AI model with the PHP backend using Python was a challenge. This required bridging two different technologies (PHP and Python) and ensuring seamless communication between them using Flask.

Security-

Implementing secure user authentication was a priority. Password hashing and session management were introduced to safeguard user data. However, scaling these security measures for real-world usage (such as multi-factor authentication) could be further improved.

Dynamic Quiz Generation-
Another challenge was ensuring that the AI-generated quizzes were sufficiently challenging and diverse. While the Gemini AI model was used to extract relevant questions, fine-tuning the complexity and variety of the questions is an area for future improvement.

8. Future Enhancements

The project provides a solid foundation but can be further enhanced with the following features:
- Advanced AI Quiz Generation: Improve the sophistication of the quiz-generation algorithm by fine-tuning the AI model to offer more nuanced questions and difficulty levels.
- More Secure Authentication: Implement features such as two-factor authentication and enhanced password security measures.
- Scalability: Deploy the system on cloud-based platforms for larger-scale usage, making it accessible to multiple institutions.
- Additional Quiz Formats: Incorporate different types of quizzes such as fill-in-the-blank, true/false, or short-answer questions to diversify assessments.

9. Conclusion

The project successfully demonstrates how AI can be used to automate quiz generation, making learning and assessments more efficient. The system reduces the professor's workload and provides students with personalized assessments. Through the integration of video content, AI-driven quizzes, and real-time feedback, this project offers a dynamic and modern learning platform. With further improvements in scalability and security, this platform could serve as a robust educational tool for institutions and educators worldwide. This concludes the final report on the quiz application project. The integration of AI and traditional learning methods shows the potential of technology in transforming education.