IS 3920 – Individual Project on Business Solutions

Software Requirements Specification

for

AI Quiz Generator with RAG-Based Question Generation and Class Management

Prepared by B.M.M. Amri [225007N]

Department of Interdisciplinary Studies

Faculty of Information Technology

University of Moratuwa

2026-01-10

Student's Declaration

I hereby declare that this report is my original work and has not been submitted, in whole or in part, for any degree or diploma at any university or other institution of higher learning. All information derived from the work of others, whether published or unpublished, has been properly acknowledged in the text, and a complete list of references has been provided.

2026/01/18	[illegible]
…………………………	……………………………………
Date	Signature of Student

Supervisors' declaration

I hereby declare that I have reviewed this project and find it adequate in both scope and quality.

1. Name of Supervisor:

   Designation:

   Date:

   Signature:

   Any further comments:

2. Name of Supervisor: Ms. Methma Samaranayake

   Designation: Technical Lead - WSO

   Date: 22/01/2026

   Signature: MethmaSamaranayake

   Any further comments:

Table of Contents

List of Figures

Revision History

Name	Date	Reason for Changes	Version

Chapter 1

Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to define the functional and non-functional requirements for the "AI Quiz Generator with RAG-Based Generation and Class Management" system. This document covers the system's scope, architectural design, user characteristics, and specific feature requirements to guide the development and testing phases.

1.2 Document Conventions

This document follows IEEE 830-1998 standards for SRS documentation. Bold text is used for emphasis, and the term "system" refers to the AI Quiz Generator platform. Requirements are prioritized as High, Medium, or Low.

1.3 Intended Audience and Reading Suggestions

- **Project Supervisors & Mentors:** To verify that the proposed solution meets the academic and technical standards required for IS 3920.
- **Developers:** To understand the specific implementation details regarding the RAG pipeline, database schema, and frontend logic.
- **Testers:** To derive test cases for validation of functional requirements like quiz generation and live hosting.

1.4 Product Scope

1.4.1. Problem in brief

Educators currently spend significant time manually creating quizzes from various learning materials, reducing time for interactive teaching. Existing tools (e.g., Moodle, Google Forms) often require manual input, lack support for diverse file types, or do not offer personalized quiz generation for students. There is a need for a scalable, AI-driven system that automates this process using multiple content formats

1.4.2. Aim and Objectives

Aim: To develop a system for automated quiz generation from multiple content sources using Retrieval-Augmented Generation (RAG) and AI, enabling effective class management and student practice.

Objectives:

- Implement AI-based question generation from PDFs, Word, PPT, images, websites, and YouTube.
- Support multiple question types: MCQ, short answer, fill-in-the-blank, and essay.
- Allow teachers to create classes, upload materials, and track student performance.
- Enable students to generate personal practice quizzes and participate in live, Kahoot-style sessions.
- Design a scalable vector database solution (ChromaDB) for efficient content retrieval.

1.5 References

Standards

1. **IEEE Computer Society**, "IEEE Recommended Practice for Software Requirements Specifications," IEEE Std 830-1998.

Technical Documentation & Frameworks

2. **Google AI for Developers**, "Gemini API Documentation," [Online]. Available: https://ai.google.dev/docs.

3. **LangChain**, "LangChain Python Documentation," [Online]. Available: https://python.langchain.com/docs/get_started/introduction.

4. **Chroma**, "ChromaDB Documentation," [Online]. Available: https://docs.trychroma.com/.

5. **Vercel**, "Next.js Documentation," [Online]. Available: https://nextjs.org/docs.

6. **Tiangolo**, "FastAPI Documentation," [Online]. Available: https://fastapi.tiangolo.com/.

7. **PostgreSQL Global Development Group**, "PostgreSQL Documentation," [Online]. Available: https://www.postgresql.org/docs/.

Chapter 2

Overall description

2.1 Product Perspective

The AI Quiz Generator is a web-based platform that operates as a standalone system but integrates with external AI services. It utilizes a **RAG (Retrieval-Augmented Generation)** pipeline to process user-uploaded content.

- **Frontend:** Next.js (React) for a responsive user interface.
- **Backend:** FastAPI (Python) for API handling and orchestration.
- **AI/ML Core:** LangChain for logic orchestration and Gemini 2.5 Flash for LLM-based generation.
- **Data Storage:** PostgreSQL for user/class data and ChromaDB for vector embeddings.

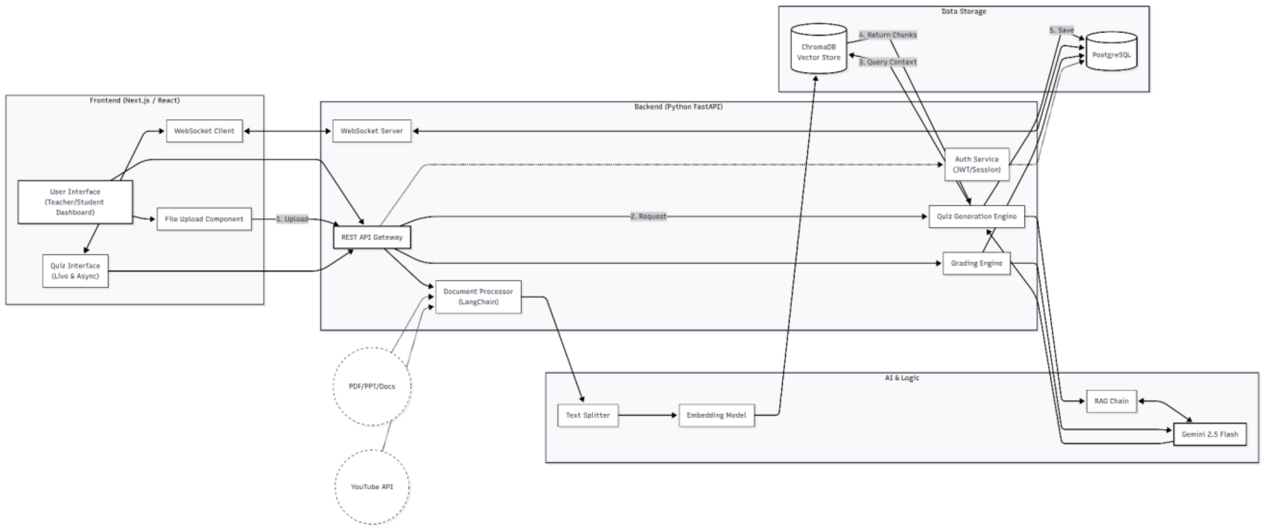


Figure 1 - Organizational Architecture

2.2 Product Functions

- **User Authentication:** Secure registration and login for teachers and students.
- **Content Processing:** Ingestion of documents (PDF, DOCX, PPT), OCR for images, and transcript extraction for YouTube videos.
- **Quiz Generation:** Automatic creation of questions based on selected materials and difficulty levels.
- **Classroom Management:** Creation of classes, enrollment via code, and assignment of mandatory quizzes.
- **Live Quiz Mode:** Real-time, synchronized quiz hosting with live leaderboards.

2.3 User Classes and Characteristics

Teacher:

- **Responsibilities:** Create classes, upload course materials, host live quizzes, and monitor class performance.
- **Technical Skill:** Moderate; comfortable with web applications and uploading files.

Student:

- **Responsibilities:** Join classes, take assigned quizzes, generate self-study quizzes, and view personal progress.
- **Technical Skill:** Basic; requires an intuitive interface for mobile or desktop use.

2.4 Operating Environment

- **Client Side:** Any modern web browser (Chrome, Firefox, Edge, Safari) on Desktop or Mobile.
- **Server Side:** Cloud-hosted environment (e.g., AWS/Vercel) capable of running Docker containers for FastAPI and Postgres.
- **Network:** High-speed internet connection required for real-time WebSocket communication (Live Quiz) and API calls.

2.5 Design and Implementation Constraints

- **API Costs:** Usage of LLMs (Gemini) and embedding models must be optimized to stay within budget/free tier limits.
- **Latency:** Real-time quizzes require low latency (<500ms) for state synchronization via WebSockets.

2.6 User Documentation

The system will provide comprehensive documentation to assist both Teachers and Students in navigating the platform. Documentation will be available in two formats: **Interactive In-App Walkthroughs** for first-time users and downloadable **PDF User Manuals** for detailed reference.

2.6.1. Teacher Guide

The Teacher Guide will serve as a comprehensive manual for classroom management and content creation. It will cover the following key modules:

- **Onboarding & Class Setup:** Step-by-step instructions on creating a new class, generating unique class codes, and managing student enrollments.
- **Material Upload & RAG Processing:** Guidelines on supported file formats (PDF, DOCX, PPT) and best practices for uploading clear images or YouTube links to ensure high-quality AI question generation.
- **Quiz Generation Workflow:** A tutorial on using the "Generate Quiz" feature, including how to select source materials, define difficulty levels, and choose question types (MCQ, Essay, etc.).
- **Live Quiz Hosting:** Instructions for launching a "Kahoot-style" live session, managing the lobby, controlling question flow, and interpreting the real-time leaderboard.
- **Analytics & Grading:** Explanations of how to interpret AI-generated feedback for essay questions and how to export class performance reports (CSV/PDF).

2.6.2. Student Guide

The Student Guide will be a concise handbook focused on learning and assessment activities:

- **Joining a Class:** Simple instructions on entering a class code to access shared materials and assignments.
- **Taking Quizzes:** A guide on the quiz interface, including how to answer different question types, track time limits, and submit responses before deadlines.
- **Personal Practice:** A tutorial on how to upload personal notes or select class materials to generate self-study practice quizzes without teacher intervention.
- **Performance Tracking:** How to view past quiz results, understand AI feedback on written answers, and track progress over time.

2.6.3. In-App Contextual Help

To reduce reliance on external manuals, the application will include:

- **Tooltips:** Hover-over text explaining complex settings (e.g., "RAG Processing Status," "Vector Embedding").
- **Error Resolution:** Clear, non-technical explanations and troubleshooting steps if a file upload fails or a file type is unsupported

2.7 Assumptions and Dependencies

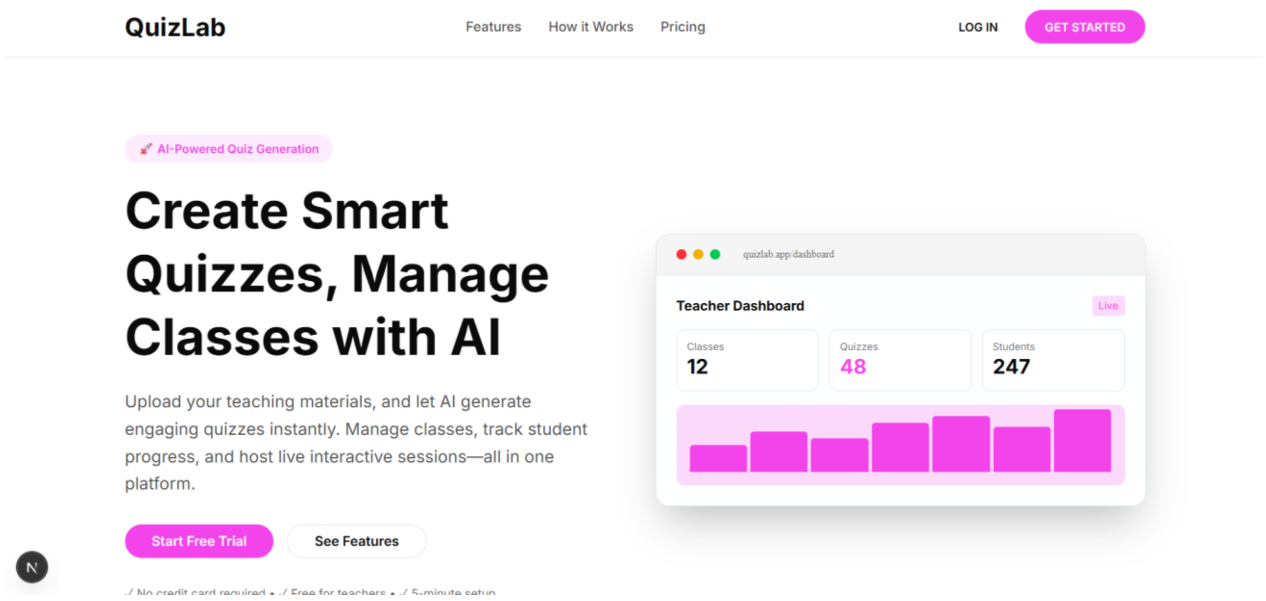
- **Assumption:** Users will upload legible documents (not corrupted or extremely low-resolution images).
- **Dependency:** The system relies on the availability of the Google Gemini API and YouTube Data API for core functionality.

Chapter 3

External Interface Requirements

3.1 User Interfaces

- **Dashboard:** A clean, card-based layout for navigating between "My Classes," "Create Quiz," and "Library."
- **Quiz Interface:** Distraction-free view for taking quizzes with a timer and progress bar.
- **Live Host Screen:** Large, high-contrast display of questions and leaderboards for classroom projection.

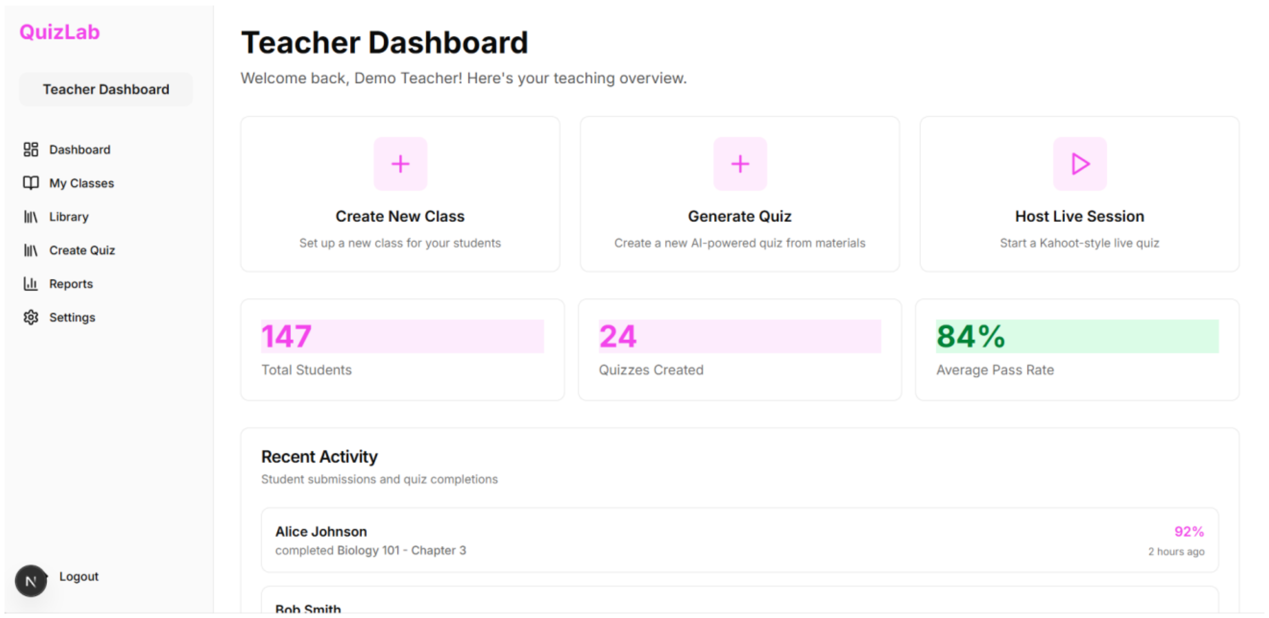


Create Smart Quizzes, Manage Classes with AI

Upload your teaching materials, and let AI generate engaging quizzes instantly. Manage classes, track student progress, and host live interactive sessions—all in one platform.

✓ No credit card required • ✓ Free for teachers • ✓ 5-minute setup

Figure 2 - Home Page



QuizLab

Teacher Dashboard

- Dashboard
- My Classes
- Library
- Create Quiz
- Reports
- Settings

Logout

Teacher Dashboard

Welcome back, Demo Teacher! Here's your teaching overview.

Create New Class — Set up a new class for your students

Generate Quiz — Create a new AI-powered quiz from materials

Host Live Session — Start a Kahoot-style live quiz

147 Total Students

24 Quizzes Created

84% Average Pass Rate

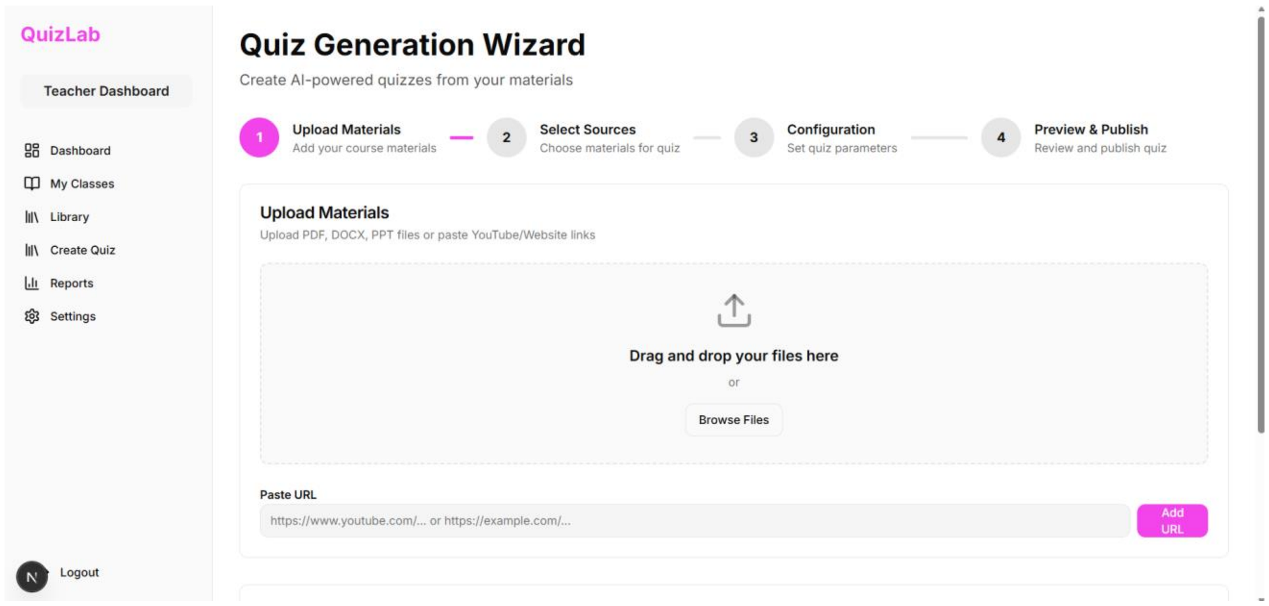
Recent Activity

Student submissions and quiz completions

Alice Johnson — completed Biology 101 - Chapter 3 — 92% — 2 hours ago

Bob Smith

Figure 3 - Teacher's Dashboard



QuizLab

Teacher Dashboard

- Dashboard
- My Classes
- Library
- Create Quiz
- Reports
- Settings

Logout

Quiz Generation Wizard

Create AI-powered quizzes from your materials

1 Upload Materials — Add your course materials
2 Select Sources — Choose materials for quiz
3 Configuration — Set quiz parameters
4 Preview & Publish — Review and publish quiz

Upload Materials

Upload PDF, DOCX, PPT files or paste YouTube/Website links

Drag and drop your files here

or

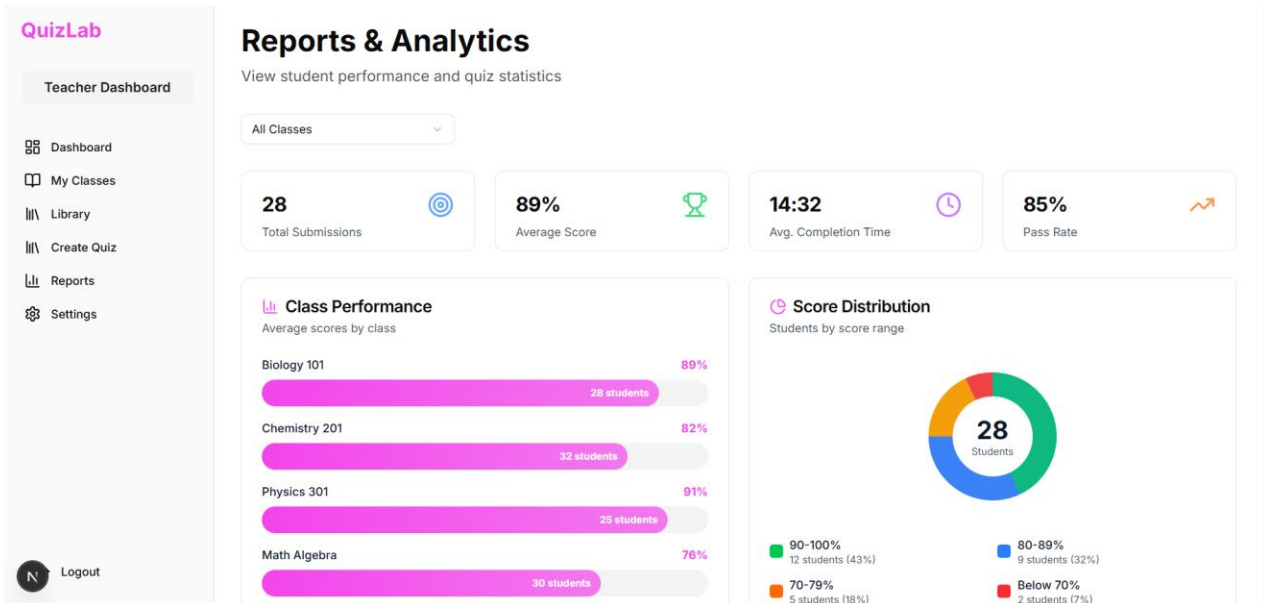
Browse Files

Paste URL

https://www.youtube.com/... or https://example.com/...

Add URL

Figure 4 - Quiz Generator Page



Reports & Analytics

View student performance and quiz statistics

28 Total Submissions

89% Average Score

14:32 Avg. Completion Time

85% Pass Rate

Class Performance

Average scores by class

Class	Average	Students
Biology 101	89%	28 students
Chemistry 201	82%	32 students
Physics 301	91%	25 students
Math Algebra	76%	30 students

Score Distribution

Students by score range

28 Students

Range	Students
🟩 90-100%	12 students (43%)
🟦 80-89%	9 students (32%)
🟧 70-79%	5 students (18%)
🟥 Below 70%	2 students (7%)

Figure 5 - Reports & Analysis Page

3.2 Software Interfaces

- **Gemini 2.5 Flash API:** Used for text generation and semantic evaluation of answers.
- **ChromaDB:** Connects via internal API to store and retrieve vector embeddings of uploaded content.
- **LangChain:** Middleware to handle document loading, text splitting, and prompt engineering.

3.3 Communication Interfaces

- **REST API:** Used for standard CRUD operations (User profiles, Class management).
- **WebSockets:** Used for the "Live Quiz" module to push real-time updates (question state, leaderboard scores) to all connected clients.
- **JSON:** The primary format for data exchange between frontend and backend.

Chapter 4

System Features

4.1 Automated Quiz Generation (RAG)

4.1.1 Description and Priority

Priority: High (Core Feature)

Description: This feature enables the automatic generation of educational quizzes from unstructured learning materials uploaded by the user. By utilizing a Retrieval-Augmented Generation (RAG) pipeline, the system extracts text from diverse file formats (PDF, DOCX, PPT, Images, and YouTube URLs). The system then segments this text into semantic chunks, converts them into vector embeddings using ChromaDB, and retrieves relevant context to prompt the Large Language Model (Gemini 2.5 Flash). This ensures that the generated questions are strictly grounded in the provided course material, minimizing hallucinations and ensuring academic relevance.

4.1.2 Stimulus/Response Sequences

- **User Action:** The user navigates to the "Create Quiz" module and uploads a specific file (e.g., "Lecture_1_Notes.pdf") or pastes a YouTube URL.
- **System Response:** The system validates the file format and size. If valid, it displays a "Processing" status bar.
- **System Internal Action:** The backend uses LangChain document loaders to extract raw text. If the file is an image, OCR is triggered. If it is a video, the transcript is fetched via the YouTube API.
- **System Internal Action:** The text is split into chunks (e.g., 1000 characters with 200 character overlap) and converted into vector embeddings, which are stored in the ChromaDB vector store.
- **User Action:** The user selects configuration options: "Number of Questions" (e.g., 10), "Difficulty" (Medium), and "Question Types" (MCQ, Essay).
- **System Response:** The system retrieves the top-k most relevant text chunks from ChromaDB based on the configuration and sends a prompt to the Gemini LLM.
- **System Response:** The generated questions are parsed into JSON format and displayed to the user for preview and editing.

4.1.3 Functional Requirements

REQ-4.1.1: File Ingestion & Validation

- **REQ-4.1.1.1:** The system shall allow users to upload files with the following extensions: .pdf, .docx, .pptx, .txt, .png, .jpg, .jpeg.
- **REQ-4.1.1.2:** The system shall enforce a maximum file size limit of 25MB per document to ensure processing efficiency.
- **REQ-4.1.1.3:** The system shall validate that YouTube URLs are publicly accessible and contain a valid video ID before attempting transcript extraction.
- **REQ-4.1.1.4:** The system shall provide an error message if an uploaded file is corrupted or password-protected.

REQ-4.1.2: Content Processing (RAG Pipeline)

- **REQ-4.1.2.1:** The system shall utilize Optical Character Recognition (OCR) to extract text from image-based uploads or scanned PDFs.
- **REQ-4.1.2.2:** The system shall utilize the YouTube Data API to fetch captions/transcripts for valid video URLs.
- **REQ-4.1.2.3:** The system shall clean extracted text by removing non-printable characters and excessive whitespace before chunking.
- **REQ-4.1.2.4:** The system shall split the cleaned text into chunks of a configurable size (default: 1000 tokens) to optimize context retrieval.
- **REQ-4.1.2.5:** The system shall generate vector embeddings for each text chunk and store them in the ChromaDB collection associated with the specific Class or User ID.

REQ-4.1.3: Question Generation

- **REQ-4.1.3.1:** The system shall support the generation of the following question types: Multiple Choice Questions (MCQ) with 4 options, True/False, Fill-in-the-Blank, Short Answer, and Essay.
- **REQ-4.1.3.2:** The system shall allow the user to mix multiple question types in a single quiz generation request.
- **REQ-4.1.3.3:** The system shall ensure that for MCQs, one option is marked as correct and three options are plausible distractors derived from the content.
- **REQ-4.1.3.4:** The system shall allow users to select a difficulty level (Easy, Medium, Hard), which modulates the complexity of the prompts sent to the LLM.

REQ-4.1.4: Output & Editing

- **REQ-4.1.4.1:** The system shall present the generated questions in an editable interface, allowing the user to modify the question text, options, or correct answer.
- **REQ-4.1.4.2:** The system shall allow users to delete irrelevant or poor-quality questions before finalizing the quiz.
- **REQ-4.1.4.3:** The system shall allow users to save the final quiz to the "Class Library" or "Personal Library".

4.1.4 Exception Handling

- **ERR-4.1.1:** If the OCR engine fails to recognize text in an image (e.g., due to blur), the system shall flag the file as "Unprocessable" and suggest the user upload a clearer version.
- **ERR-4.1.2:** If the Gemini API returns a rate-limit error, the system shall implement an exponential backoff strategy and retry the request up to 3 times before notifying the user.
- **ERR-4.1.3:** If the uploaded content is insufficient to generate the requested number of questions (e.g., a 1-page PDF for 50 questions), the system shall generate the maximum possible unique questions and warn the user: "Content insufficient for requested quantity."

4.2 Classroom Management

4.2.1 Description and Priority

Priority: High (Core Feature)

Description: This feature provides the administrative backbone of the system, enabling Teachers to organize students into logical groups (Classes) for targeted content distribution. It replaces manual email lists or spreadsheet tracking. Teachers can create digital classrooms, generate secure access codes, and monitor enrollment. For Students, this module provides a unified "My Classes" dashboard where they can access all course-specific materials and pending assignments. The system ensures data isolation, meaning students in "Class A" cannot access materials or quizzes from "Class B" unless explicitly enrolled.

4.2.2 Stimulus/Response Sequences

- **User Action (Teacher):** The Teacher logs in and clicks "Create New Class," entering a name (e.g., "CS101 - Intro to AI") and description.
- **System Response:** The system creates a new database record and generates a unique 6-character alphanumeric Join Code (e.g., "X7-99-AB").
- **User Action (Student):** A Student logs in, navigates to "Join Class," and enters the code provided by the teacher.
- **System Internal Action:** The system validates the code against active classes. If valid, it creates an enrollment record linking the Student_ID to the Class_ID.
- **System Response:** The Student is immediately redirected to the Class Dashboard, where they can see the syllabus and available quizzes.
- **User Action (Teacher):** The Teacher views the "Students" tab to verify the new enrollment and can optionally remove unauthorized students.

4.2.3 Functional Requirements

REQ-4.2.1: Class Creation & Configuration

- **REQ-4.2.1.1:** The system shall allow Teachers to create an unlimited number of classes (subject to database storage limits).
- **REQ-4.2.1.2:** The system shall require a unique name for each class within the Teacher's specific account (e.g., a teacher cannot have two "CS101" classes, but two different teachers can).
- **REQ-4.2.1.3:** The system shall automatically generate a unique, case-insensitive 6-character Join Code for every new class to facilitate easy sharing.
- **REQ-4.2.1.4:** The system shall allow Teachers to "Archive" classes at the end of a semester, making them read-only but preserving all data.

REQ-4.2.2: Student Enrollment

- **REQ-4.2.2.1:** The system shall allow Students to join a class by entering a valid Join Code.
- **REQ-4.2.2.2:** The system shall prevent duplicate enrollments (i.e., a student cannot join the same class twice).
- **REQ-4.2.2.3:** The system shall allow Teachers to manually remove a student from a class, which will revoke the student's access to all class materials and quizzes.
- **REQ-4.2.2.4:** The system shall maintain an "Enrollment Timestamp" for auditing purposes.

REQ-4.2.3: Assignment Management

- **REQ-4.2.3.1:** The system shall allow Teachers to assign a generated quiz to one or specific classes.
- **REQ-4.2.3.2:** The system shall allow Teachers to set a "Mandatory" flag on quizzes. Mandatory quizzes must be highlighted in the Student's "Pending Tasks" list.
- **REQ-4.2.3.3:** The system shall allow Teachers to define a "Submission Deadline" (Date and Time). The system shall block attempts to submit the quiz after this timestamp (unless a grace period is configured).

4.2.4 Exception Handling

- **ERR-4.2.1:** If a Student enters an invalid Join Code, the system shall display a clear error message: "Class not found. Please check the code."
- **ERR-4.2.2:** If a Student attempts to join a class they are already enrolled in, the system shall redirect them to that Class Dashboard without creating a new record.

4.3 Live Quiz Hosting

4.3.1 Description and Priority

Priority: Medium (Differentiation Feature)

Description: The Live Quiz module transforms the platform into a synchronous engagement tool similar to Kahoot! or Quizizz. It allows a Teacher to host a quiz session where all students answer questions simultaneously on their own devices. This requires real-time state synchronization using **WebSockets**, ensuring that every student sees the same question at the same time. The feature includes a "Lobby" for waiting, a "Game" view for answering, and a "Leaderboard" that updates after every question to gamify the learning experience.

4.3.2 Stimulus/Response Sequences

- **User Action (Host):** The Teacher selects a quiz and clicks "Host Live."
- **System Response:** The system opens a WebSocket channel, creates a unique numeric Game PIN (e.g., "559922"), and displays the Lobby screen.
- **User Action (Student):** Students enter the Game PIN on their devices.
- **System Response:** The system connects the student via WebSocket to the specific game room and displays their nickname on the Host's Lobby screen.
- **User Action (Host):** The Teacher clicks "Start Game."
- **System Internal Action:** The server broadcasts the first question payload (Question text, options, time limit) to all connected clients simultaneously.
- **User Action (Student):** Students select an answer. The client sends the selection + timestamp to the server.
- **System Internal Action:** When the timer expires, the server calculates scores based on correctness and speed, then broadcasts the correct answer and updated leaderboard rankings.

4.3.3 Functional Requirements

REQ-4.3.1: Session Management (WebSockets)

- **REQ-4.3.1.1:** The system shall use the WebSocket protocol (via FastAPI Websockets) to maintain persistent, low-latency connections between the Host and all Student clients.
- **REQ-4.3.1.2:** The system shall generate a temporary, unique 6-digit Game PIN for each live session, valid only while the session is active.
- **REQ-4.3.1.3:** The system shall support a "Lobby State" where the Host can see a real-time count and list of names of joined players before starting the quiz.

REQ-4.3.2: Real-Time Gameplay

- **REQ-4.3.2.1:** The system shall broadcast quiz questions sequentially. Students must not be able to advance to the next question until the Host triggers it or the timer expires.
- **REQ-4.3.2.2:** The system shall enforce a server-side timer. Answers received after the timer expires (accounting for a configurable network latency buffer of ~500ms) shall be rejected.
- **REQ-4.3.2.3:** The system shall calculate points using a decay algorithm: Max points for immediate correct answers, decreasing linearly as time elapses.

REQ-4.3.3: Leaderboard & Feedback

- **REQ-4.3.3.1:** The system shall display an "Answer Summary" on the Host screen after every question, showing a bar chart of how many students chose each option.
- **REQ-4.3.3.2:** The system shall compute and display a "Top 5 Leaderboard" after every question, showing the student's Nickname and Total Score.
- **REQ-4.3.3.3:** The system shall provide immediate feedback on the Student's device (Green screen for Correct, Red for Incorrect) after the question round ends.

4.3.4 Exception Handling

- **ERR-4.3.1:** If a Student's internet connection drops, the system shall attempt to automatically reconnect the WebSocket. If reconnection succeeds within the same question window, the student can answer. If the question has ended, they skip to the current state.
- **ERR-4.3.2:** If the Host closes the browser tab, the system shall detect the disconnection and automatically end the session for all participants after a 30-second timeout.

4.4 Automatic Evaluation & Analytics

4.4.1 Description and Priority

Priority: High (Core Feature)

Description: This feature automates the grading process, drastically reducing teacher workload. While Multiple Choice Questions (MCQ) are graded using simple logic matching, the system differentiates itself by using **AI Semantic Analysis** for subjective questions (Short Answer, Essay). The system uses the LLM (Gemini) to compare the student's written response against the "Correct Answer" or the source material vectors, assigning a score based on semantic similarity rather than exact keyword matching. Additionally, this module aggregates data to provide actionable insights, such as identifying "weak topics" for a class.

4.4.2 Stimulus/Response Sequences

- **User Action:** A student submits a completed quiz containing both MCQs and an Essay question.
- **System Internal Action:**
- **Step A (MCQ):** The system compares the selected Option_ID with the stored Correct_Option_ID. Score is assigned immediately.
- **Step B (Essay):** The system constructs a prompt for the LLM: *"Compare the student's answer [Student_Text] with the reference answer [Reference_Text]. Rate correctness on a scale of 0-10 and provide brief constructive feedback."*

- **System Internal Action:** The LLM returns a JSON object containing the Score and Feedback_String.
- **System Response:** The system saves the total score to the database.
- **User Action:** The Teacher opens the "Reports" tab.
- **System Response:** The system renders a bar chart showing the grade distribution (e.g., how many got A, B, C) and a list of students who failed.

4.4.3 Functional Requirements

REQ-4.4.1: Objective Grading (MCQ/Fill-in-Blanks)

- **REQ-4.4.1.1:** The system shall grade Multiple Choice and True/False questions instantly upon submission with 100% accuracy based on the stored key.
- **REQ-4.4.1.2:** For Fill-in-the-Blank questions, the system shall support fuzzy matching (e.g., ignoring capitalization or minor spelling errors) if configured by the teacher.

REQ-4.4.2: AI Semantic Grading (Essays)

- **REQ-4.4.2.1:** The system shall use the Gemini API to evaluate open-ended text responses.
- **REQ-4.4.2.2:** The evaluation prompt sent to the AI must include the *Student's Answer*, the *Reference Material/Context*, and a *Rubric* (e.g., "Focus on keyword presence and logical flow").
- **REQ-4.4.2.3:** The system must generate and store a brief (1-2 sentence) text justification for the grade given by the AI, which is visible to the student (e.g., "Good mention of X, but you missed concept Y").
- **REQ-4.4.2.4:** The system shall allow Teachers to manually override the AI-assigned grade if they disagree with the evaluation.

REQ-4.4.3: Analytics & Reporting

- **REQ-4.4.3.1:** The system shall calculate aggregate metrics for each quiz: Average Score, Median Score, and Highest/Lowest Score.
- **REQ-4.4.3.2:** The system shall identify "Difficult Questions" by flagging questions where less than 40% of the class answered correctly.
- **REQ-4.4.3.3:** The system shall allow Teachers to export the full Gradebook for a class as a CSV file (compatible with Excel) or a PDF summary report.

4.4.4 Exception Handling

- **ERR-4.4.1:** If the AI service is unavailable during grading, the system shall mark the Essay questions as "Pending Grading" and queue them for processing when the service restores, while still showing the MCQ score immediately.

- **ERR-4.4.2:** If the AI feedback contains hallucinated or inappropriate content (detected via a safety filter), the system shall suppress the feedback text and flag the response for Teacher review.

Chapter 5

Other Nonfunctional Requirements

5.1 Performance Requirements

- **Quiz Generation Time:** Should not exceed 30 seconds for a standard 10-question quiz.
- **Live Quiz Latency:** Synchronization delay between host and student devices must be under 1 second.

5.2 Security Requirements

- **Data Privacy:** Student grades and personal data must be accessible only to authorized teachers.
- **Authentication:** Passwords must be hashed (e.g., bcrypt) before storage in PostgreSQL.

5.3 Software Quality Attributes

- **Scalability:** The architecture must support multiple concurrent classes running live quizzes without crashing.
- **Usability:** The interface should be intuitive enough that no user manual is required for basic tasks.

Chapter 6

Appendix A – Glossary

- **API (Application Programming Interface):** A set of protocols and tools that allows different software applications to communicate with each other. In this system, APIs are used to connect the frontend, backend, and external AI services.
- **ChromaDB:** An AI-native, open-source vector database used in this project to store and retrieve vector embeddings for efficient content retrieval.
- **Embeddings:** Numerical representations (vectors) of text that capture semantic meaning. The system converts uploaded course materials into embeddings to enable the AI to "understand" and search the content.
- **FastAPI:** A modern, high-performance web framework for building APIs with Python, used as the backend for this system.
- **Gemini 2.5 Flash:** The specific Large Language Model (LLM) developed by Google, utilized in this project for generating quiz questions and evaluating student answers.
- **LangChain:** An orchestration framework used to connect the LLM with external data sources (PDFs, websites) and manage the RAG pipeline.
- **LLM (Large Language Model):** A type of artificial intelligence algorithm that uses deep learning techniques and massive data sets to understand, summarize, generate, and predict new content.
- **Next.js:** A React-based web development framework used to build the user interface (frontend) of the application.
- **OCR (Optical Character Recognition):** Technology used to convert different types of documents, such as scanned paper documents, PDF files, or images captured by a digital camera, into editable and searchable data.
- **PostgreSQL:** The relational database management system used to store structured data such as user profiles, class details, and quiz scores.
- **RAG (Retrieval-Augmented Generation):** A technique that enhances the accuracy and reliability of generative AI models with facts fetched from external sources. In this project, it allows the AI to generate quizzes based specifically on the user's uploaded notes rather than general knowledge.
- **Semantic Analysis:** The process of understanding the meaning and interpretation of words, signs, and sentence structure. This system uses it to compare student essay answers against the source material for automatic evaluation.
- **Vector Database:** A type of database indexed for storing and retrieving high-dimensional vector embeddings, essential for the RAG architecture.
- **WebSocket:** A communication protocol that provides full-duplex communication channels over a single TCP connection, used in this project to support real-time interactivity for Live Quizzes.

Chapter 7

Appendix B – Analysis Models

7.1 Use Case Diagram

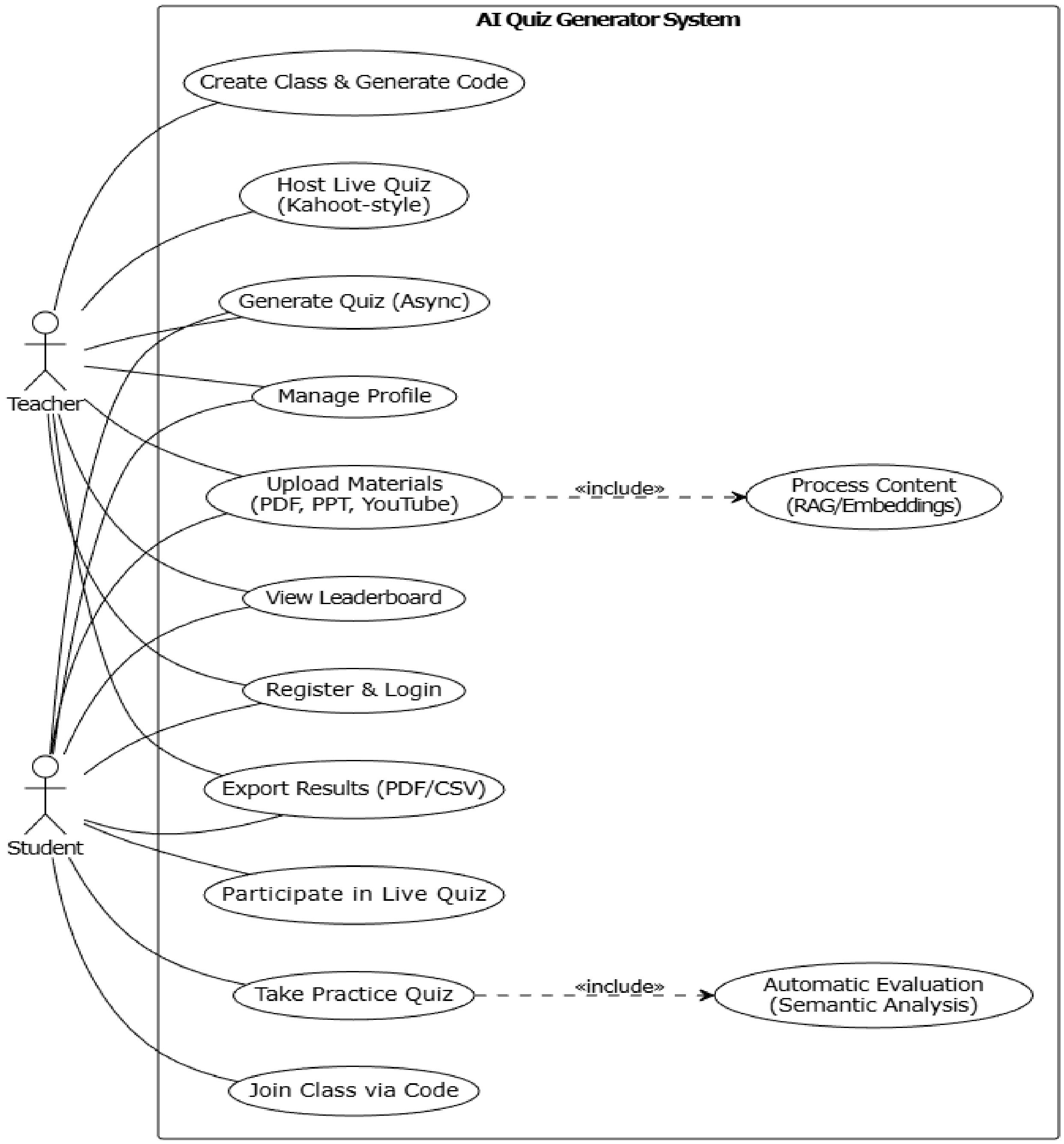


Figure 6 - Use Case Diagram

7.2 Class Diagram

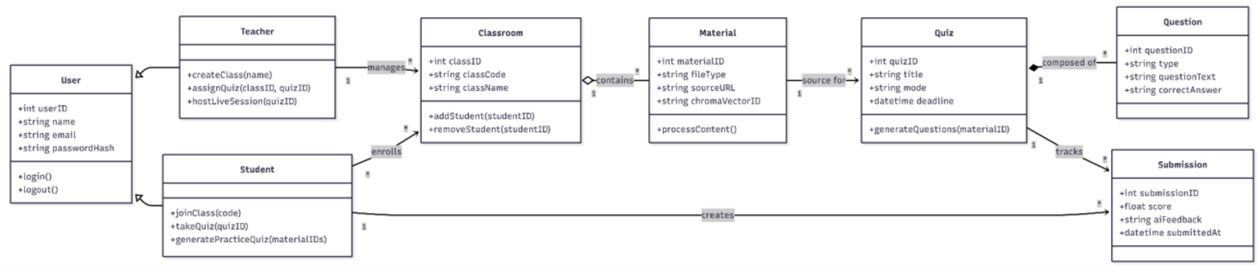


Figure 7 - Class Diagram

7.3 ER Diagram

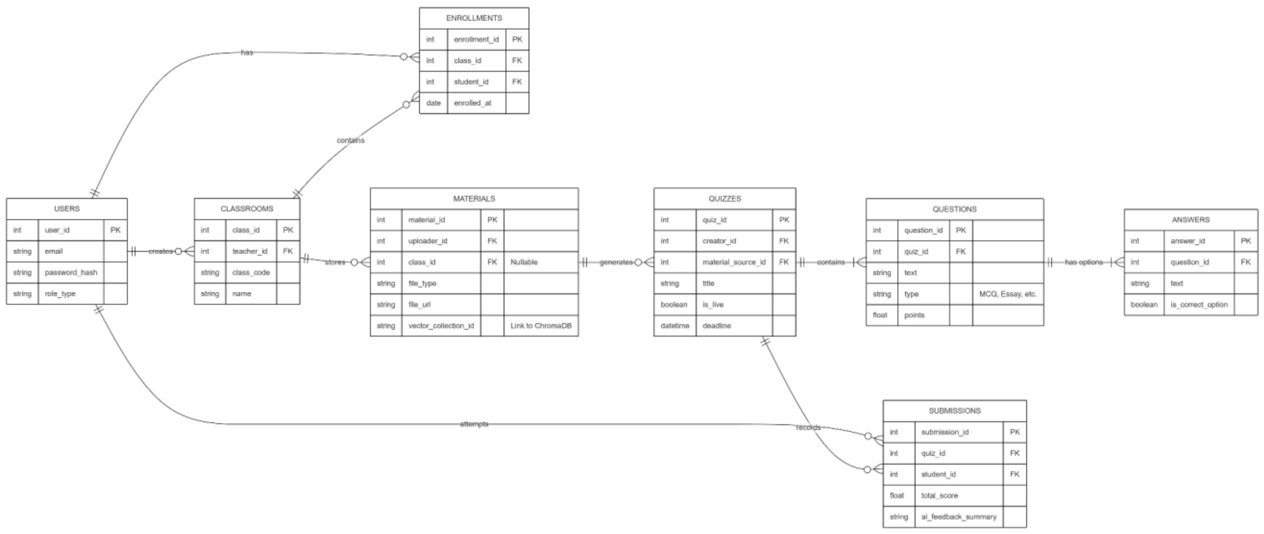


Figure 8 - ER Diagram