# AI-Powered Quiz Generation and Evaluation Platform – Project Report

**Student Names:**

**Rohit M. Ghosarwadkar (24P0620005), Seamus Rodrigues (24P0620011)**

**MSc AI, Part 1, Goa University**

## 1. Introduction

In the rapidly evolving educational landscape, Artificial Intelligence (AI) is transforming how instructors create, deliver and assess learning materials. This project presents an **AI-driven quiz generation and evaluation platform** that automates time-consuming assessment tasks, delivers instant feedback to students, and empowers educators with actionable insights.

## 2. Problem Statement

Educators encounter several pain-points during the assessment lifecycle:

- **Time-Consuming Quiz Creation** – Manually extracting questions from study material is labor-intensive.
- **Delayed Feedback** – Traditional grading postpones feedback, limiting timely interventions.
- **Lack of Personalization** – One-size-fits-all quizzes cannot accommodate diverse learner needs.
- **Fragmented Toolchain** – Teachers juggle multiple apps for uploading material, generating quizzes, administering tests and grading.

A unified, AI-enabled solution is required to automate and personalize every stage of the assessment workflow.

## 3. Objectives

1. Automatically generate high-quality quiz questions from uploaded course content.
2. Provide educators with an interface to curate the generated questions.
3. Deliver quizzes to students and evaluate responses in real time.
4. Offer immediate, individualized feedback and analytics to improve learning outcomes.

## 4. Approach

The platform follows a multi-stage pipeline:

1. **PDF Upload & Processing**

- Educators upload study material in PDF format.
- The backend extracts text, splits it into coherent sentence chunks and embeds each chunk as a vector for semantic search.

2. **Question Generation**

- Large Language Models (LLMs) analyse the vectorised content and propose diverse, thought-provoking questions.
- Questions are returned in a structured JSON schema.

3. **Question Selection**

- Through the frontend, teachers review, edit and select questions to assemble a quiz tailored to their syllabus.

4. **Quiz Administration**

- Finalised quizzes are published to students via a responsive web interface (Next.js).

5. **Automated Evaluation**

- Student answers are batch-evaluated by an LLM, returning a verdict (`correct` / `incorrect`) with a concise justification.

6. **Feedback & Analytics**

- Results are persisted and visualised for educators to track performance and identify learning gaps.

## 5. Retrieval-Augmented Generation (RAG)

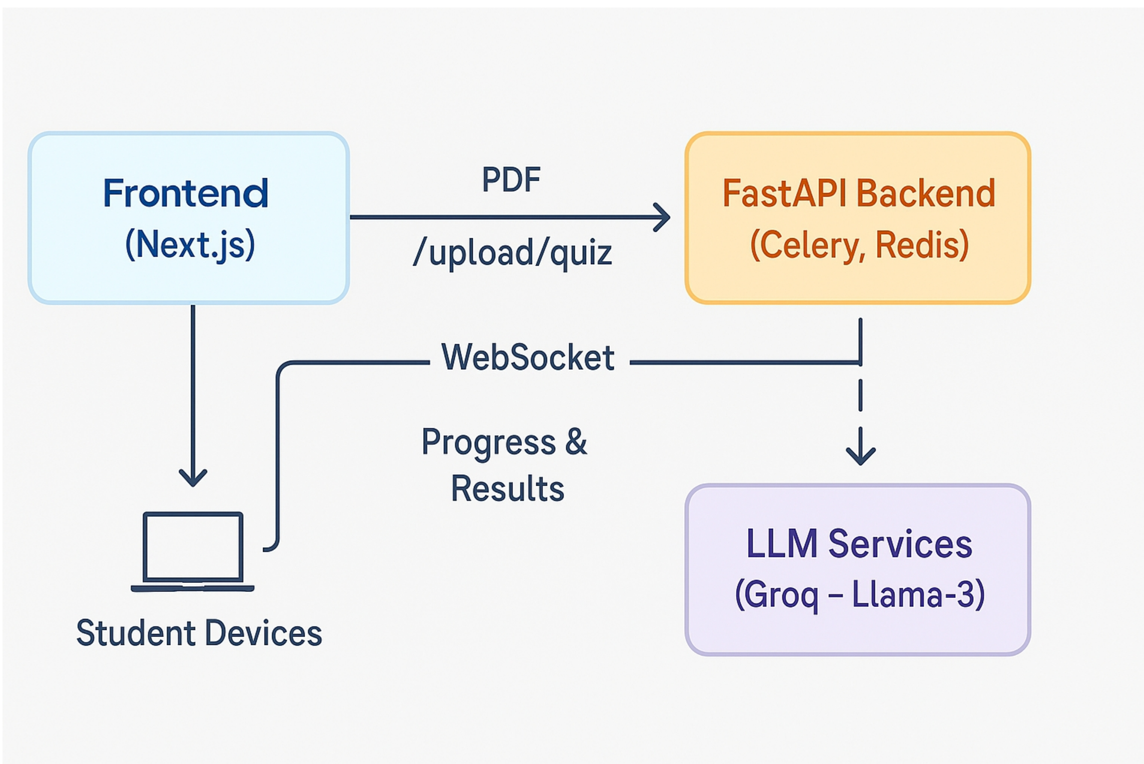
The platform employs Retrieval-Augmented Generation (RAG) to generate contextually relevant quiz questions from uploaded course content. RAG combines a retriever, which identifies relevant document chunks using semantic search over vectorized text, with a generator (in this case, Llama-3) to produce high-quality questions. This approach leverages external knowledge (the uploaded PDFs) without modifying the pre-trained model's weights, ensuring flexibility and scalability across diverse educational content.

**Comparison with Fine-Tuning:**

- **RAG Advantages:**
  - **Contextual Flexibility:** RAG dynamically retrieves relevant content at inference time, enabling the model to adapt to new material without retraining.
  - **Cost Efficiency:** Unlike fine-tuning, which requires computationally expensive retraining and labeled datasets, RAG relies on embeddings and a pre-trained model, reducing setup time and costs.
  - **Generalization:** RAG avoids overfitting to specific domains, making it ideal for varied course materials.
- **Fine-Tuning Advantages:**
  - **Domain Specialization:** Fine-tuning optimizes a model for specific tasks or domains, potentially improving performance on highly specialized content.
  - **Reduced Latency:** Fine-tuned models may require less runtime computation, as they do not rely on external retrieval.
- **Trade-Offs:**
  - Fine-tuning demands significant computational resources and curated datasets, which may not be feasible for educators with diverse or frequently updated materials.
  - RAG introduces retrieval latency and depends on the quality of the vector store, but it excels in scenarios requiring broad, dynamic knowledge.

**Implementation in the Platform:** The platform uses RAG by vectorizing PDF content into a Weaviate vector store, where sentence-level chunks are embedded using a local encoder. During question generation, the system queries the vector store to retrieve relevant chunks, which are then fed to Llama-3 to produce JSON-formatted questions. This approach ensures questions are tightly aligned with the source material, enhancing relevance and pedagogical value. RAG was chosen over fine-tuning due to its ability to handle diverse PDFs without requiring model retraining, aligning with the platform's goal of minimizing educator effort.

## 6. System Architecture




## 6. Component Details

| Component | Source Module | Summary |
|---|---|---|
| File Upload | file_op.py | Receives PDF via /uploadfile, saves to temp storage, enqueues Celery task, and streams progress back over WebSocket. |
| PDF | pdf_vectorizer.py | Loads PDFs, splits into sentence-level chunks |

| | | |
|---|---|---|
| Vectorizer | | (SentenceTextSplitter), returns list for embedding. |
| Vector Operations | vector_operations.py | Creates collections, embeds chunks with a local encoder, and stores in Weaviate. Publishes progress to Redis. |
| Question Generation | generate_questions.py | Iteratively queries Llama-3 with grouped chunks to produce JSON-formatted questions. |
| Evaluation Service | evaluate.py | Accepts batches of Q-A pairs, invokes Llama-3, parses structured correctness/justification JSON. |
| API Integration | app.py | Exposes /gen-questions and /evaluate-batch endpoints and wires routers. |

## 7. Technology Stack

- **Languages:** Python 3.11, TypeScript / JavaScript ES2023
- **Frameworks & Libraries:** FastAPI, Celery, Next.js, React, LangChain, Transformers
- **Orchestration & Messaging:** Redis, Weaviate
- **LLM Provider:** Groq Cloud (Llama-3-70B-Versatile)
- **DevOps:** Docker, GitHub Actions, Vercel

## 8. Challenges & Mitigations

1. **PDF Parsing Accuracy** – Utilised `pymupdf4llm` and fallback OCR to handle varied layouts.
2. **Question Relevance** – Iteratively refined prompts and used smaller context chunks (≈ 200-300 tokens) to balance API cost with contextual fidelity.
3. **Scalability** – Horizontal Celery workers and Redis streams decouple long-running tasks.
4. **User Experience** – Conducted usability tests; adopted a card-based quiz builder and progress indicators for transparency.
5. **Chunking Strategy Trade-offs**: When processing large documents, we needed to divide the text into smaller segments or "chunks" to comply with the token limits of LLMs. Selecting the optimal chunk size was critical:
    - **Small Chunks (100–300 tokens)**: Using smaller chunks ensured each input remained well within the LLM's context window, minimizing truncation risk and improving precision in question generation. However, it significantly increased the number of API calls to the LLM, leading to higher latency and higher operational costs.

- **Large Chunks (400–600 tokens)**: Slightly larger chunks helped reduce the number of API calls, improving system throughput and lowering the overall cost. However, larger chunks increased the chances of losing context boundaries between different concepts, which sometimes affected the quality and relevance of generated questions.

## 9. Reference Images

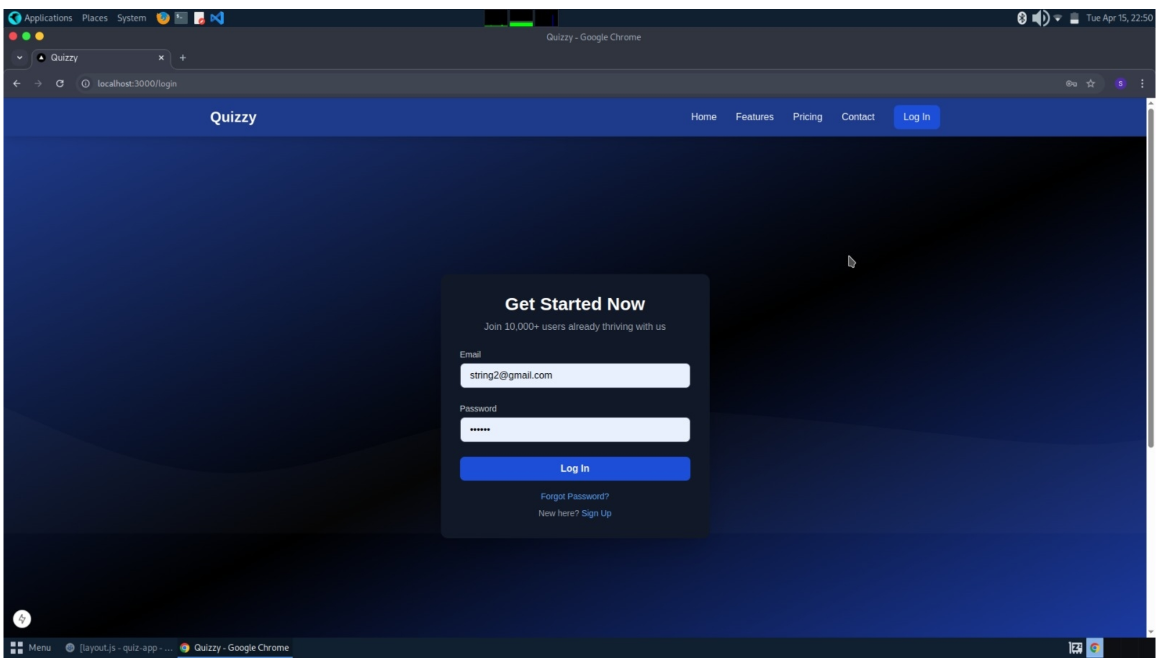



**Figure shows login page**

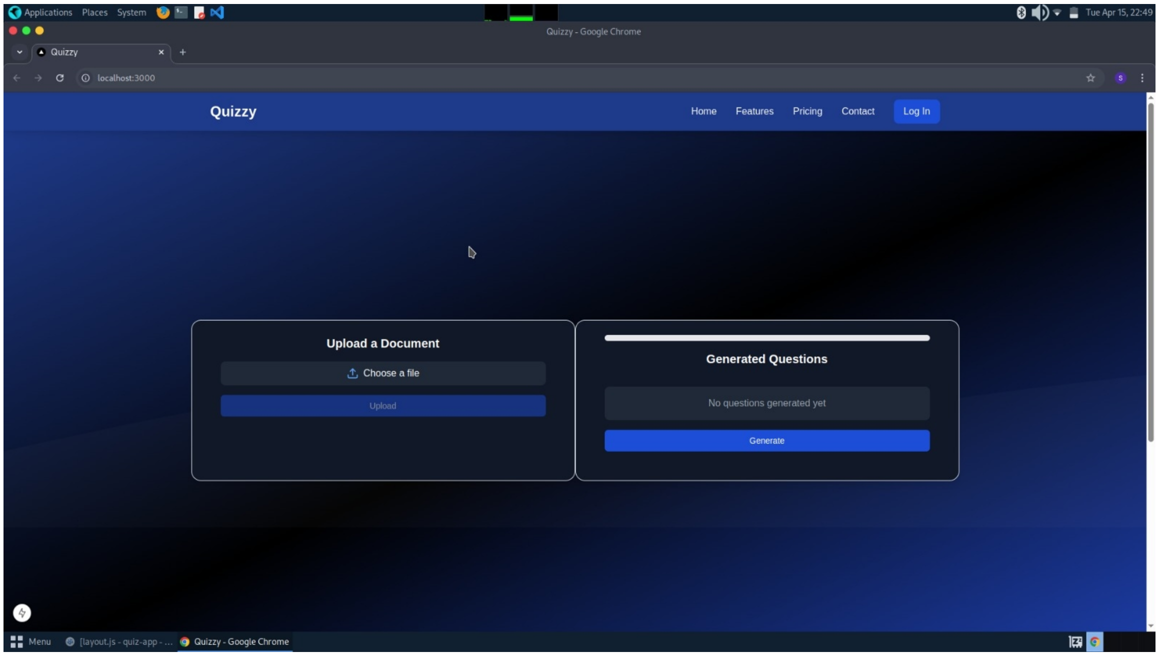



**Figure shows upload pdf option**

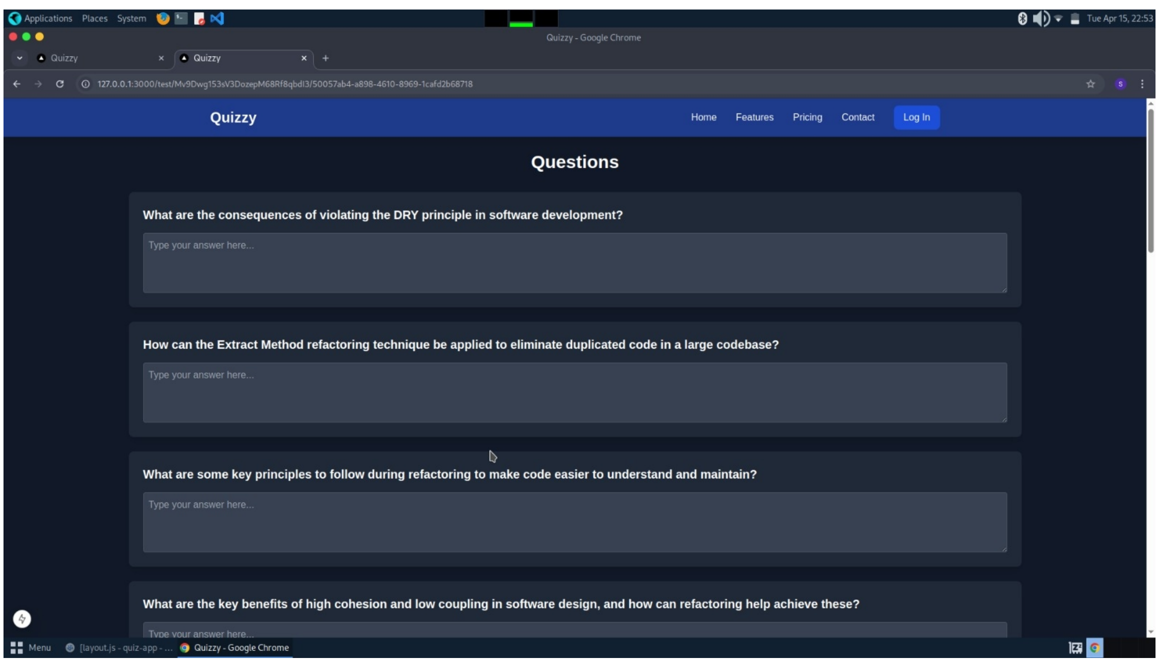



**Figure shows quiz page**

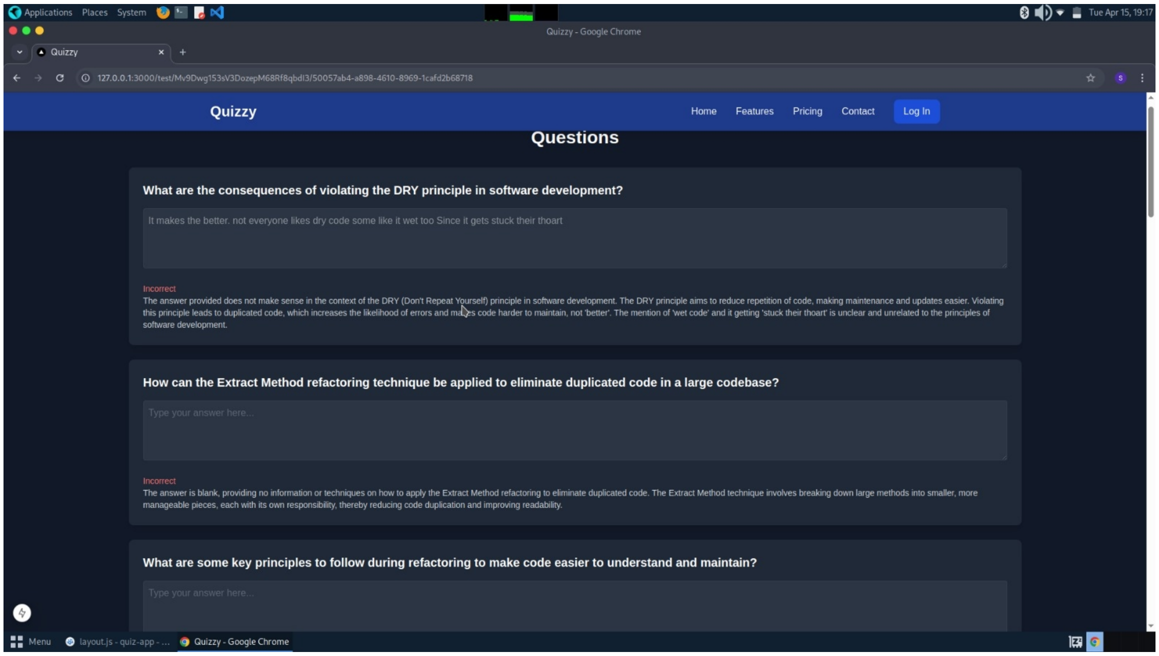



**Figure shows evaluation**

## 10. Future Work

- **Multilingual Support** – Incorporate translation layer for non-English content.
- **Mobile Apps** – Flutter-based companion apps for iOS & Android.
- **Advanced Analytics** – Adaptive learning dashboards & cohort analysis.
- **LMS Integration** – LTI-compatible modules for Moodle, Canvas, Blackboard.

## 11. Conclusion

The proposed platform consolidates quiz creation, delivery and evaluation into a seamless, AI-augmented workflow. By **reducing manual effort** and delivering **instant, personalized feedback**, it enables educators to focus on instruction while fostering a more engaging learning environment.