AI-LMS Final Report

Junaid Jamshed (25100210), Jawad Saeed (2510009), Muhammad Ahmad (25100076), Huraira Anwer (25100314), Ibrahim Farrukh (25100227)

Problem Statement

This product focuses on the educational technology domain, specifically through the development of an **AI-powered Learning Management System (LMS)**. In this domain, there are several pain points: students often face a lack of summarized content for exam preparation, while instructors waste their valuable time manually grading assignments and quizzes.

Related Work

Recent advancements in educational technology have explored the integration of large language models (LLMs) into learning environments:

- **Lee et al. (2023)** present a vision for the **multimodality of AI in education**, discussing its potential role in progressing toward artificial general intelligence [arXiv].
- **Xu et al. (2024)** provide a comprehensive **survey on the use of LLMs in education**, outlining applications, challenges, and future directions [arXiv].
- **Spriggs et al. (2025)** introduce an **adaptive LMS powered by integrated LLMs**, highlighting personalization in learning experiences [arXiv].
- **Chu et al. (2025)** review the latest developments in **LLM agents for education**, focusing on their practical implementations and educational benefits [arXiv].
- **Mzwri et al. (2025)** propose **Dynamic Course Content Integration (DCCI)** in the Ask ME Assistant, bridging LMS platforms with generative AI for seamless course content access and interaction [axXiv].

Design Approach

System Architecture

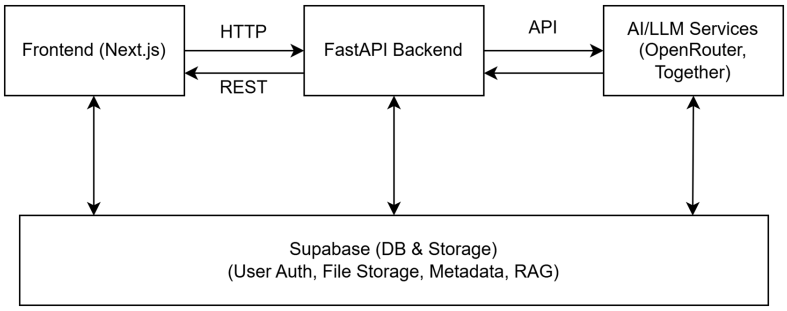

Image 1: System Architecture Overview

Our AI LMS operates as a web-based application built using a full-stack architecture. Here is a brief overview of the major components of our application:

- **Next JS:** Frontend framework providing us with Static and server-side rendering capabilities with React
- **FastAPI:** A light Python-based backend allowing us to integrate our AI-based features with ease
- **Supabase:** SQL-based database providing us with user authentication and storage buckets for storing course-related materials

- **Langchain/LangGraph:** Langchain provides us with the functionality to implement the RAG framework, while LangGraph acts as an orchestration framework to help design our agents for different features
- **Openrouter:** Unified LLM interface allowing us to use deployed LLMs through API calls
- **Together.ai:** Hosted a finetuned llama3-8b (finetuned using Unsloth) in a serverless manner on a verification feedback corpus (self-generated) by uploading .safetensors and .configs

LLM Integration

Role of LLMs

LLMs are an integral part of the application, with integration into all key features to promote personalized and automated learning. All of our features make use of **Llama 4 Maverick,** which is a multimodal model having 17 billion parameters and operates on a **Mixture of Experts** architecture.

Students have access to multiple AI-powered features, which include **Lecture Summarization**, **Flashcard Generation**, **Practice Question Generation,** and a **Talk to your Lecture Chatbot**. For each of these features, students have the ability to select specific lectures to provide additional context to the LLM.

Instructors have access to various AI-powered features, which include **Quiz** (theoretical assessment) and **Assignment** (practical / coding assessment) **Generation.** Instructors can choose to send specific lectures to provide additional context to the LLM. The generated quizzes are evaluated by LLM-as-judge and improved iteratively on metrics like coherence, clarity, depth, and structure.

Moreover, instructors have access to AI-powered grading features, which can be used to optimize their workflows.

Integration Architecture

The integration architecture of each of the proposed features is highlighted as follows:

- **Summarization:** The Summarization feature involves taking input from the user in the form of a prompt, where the user can specify any specific guidelines for focusing on specific topics. This is followed by the selection of the specific lectures for summarization, and once selected, a request is sent to the backend, where the corresponding text is extracted from the files and preprocessed. Once done, the preprocessed text alongside the user prompt and added to the summarization prompt, which features detailed instructions and guidelines on summary generation as well as guardrails to ensure that the generated summarization is relevant.
- **Flashcard Generation:** The process for the generation of flashcards is the same as that of summarization, with the difference of the prompt that is used for querying the LLM. It contains guardrails and further instructions on the format of the generated flashcards in a question-and-answer format. This is crucial since all generated flashcards are saved in student's private storage buckets and can be viewed whenever required, which requires parsing on the generated flashcards to display on the frontend.
- **Practice Question Generation:** Practice question generation follows an agent-based workflow where users can provide a difficulty level in addition to a prompt and relevant lectures to generate practice questions of varying levels of difficulty. Additionally, the workflow involved use of Retrieval Augmented Generation for the creation of contextualized prompts and retrieval of relevant documents in the course's storage bucket to generate practice questions.
- **Talk to Your Lecture Chatbot:** This feature allows you to select one or multiple lectures which are retrieved using RAG and queried through a chatbot interface. The system intelligently fetches relevant chunks from the selected lecture content and uses a language model to answer your questions, enabling a conversational way to review and understand lecture material.
- **Assignment/Quiz Generation:** Assignment / quiz generation follows an agentic workflow where users can provide a prompt and relevant lectures to generate both theoretical and coding assessments. The workflow leverages RAG for creation of contextualized prompts and retrieving relevant documents from course's storage bucket. Parsing techniques are applied to convert to PDF in case of a quiz, and JSON (later used as .IPYNB - notebook format) in case of a coding assignment. The generated quizzes are evaluated by LLM-as-judge and improved iteratively on metrics like coherence, clarity, depth, and structure. The

generated assignments are evaluated with human-in-the-loop for feedback and suggestions for improvements.

- **Assignment/Quiz Grading:** The grading process for assignments and quizzes follows a systematic and unified workflow. Once a quiz or assignment is generated by the instructor or administrator, it becomes available to students under the respective submission tab within the course interface. Students upload their submissions through this interface, after which the files are made accessible on the instructor/admin dashboard under the corresponding grading tab. Instructors have the flexibility to evaluate each submission individually. When the instructor initiates the grading process, both the student's submission and the instructor-approved mark scheme, originally generated by a large language model (LLM), are sent to the backend. Various parsing techniques are employed to convert the submission files (PDF or notebook format) into structured JSON. The extracted answers are then evaluated using an LLM, which compares them against the expected responses in the mark scheme and assigns scores based on correctness and semantic similarity. Upon completion of this process, two files are generated: a marks file (plain text) and a detailed feedback report (PDF). These are displayed on the instructor side for each student submission. On the student side, individuals can view only their respective marks and feedback, ensuring personalized and private performance insights.
- **Answer Key and Rubric Generation:** This feature enables instructors or administrators to generate high-quality mark schemes for quizzes and assignments. The instructor uploads the relevant quiz or assignment document, which is then parsed and processed on the backend. The parsed content is passed to an LLM along with an extensive system prompt that outlines the expected answer format, grading rubric, and evaluation criteria for each type of question. The LLM first produces an initial draft of the mark scheme. This draft is then reviewed by the model itself, acting in an evaluative capacity, against predefined metrics such as completeness, clarity, and logical soundness. If the initial mark scheme meets the required standards across all evaluation dimensions, a final, detailed rubric is generated and returned to the instructor. This rubric is subsequently used during the grading phase to ensure consistent and accurate evaluation of student submissions.

Experiment Design

Retrieval Augmented Generation

Evaluation Metrics

- **Query Response Time(s):** Average time taken to retrieve relevant content and generate a response
- **Retrieval Relevance(%):** Percentage of retrieved lecture content chunks that are relevant to the user's prompt or query.

Validation Strategies

- **User Testing:** Conduct user testing with a small group of students and instructors to test the relevance and usefulness of RAG-generated outputs. Manifests in the features using it.
- **Functional Testing:** Verified that the RAG pipeline correctly retrieves and processes lecture content from Supabase storage buckets, and embeddings are correctly generated on upload and placed in the relevant tables.

Summarization

Evaluation Metrics

- **Relevance Score(1-5):** User rating on how well the generated summary covers key topics
- **Time Saved(minutes):** Measured in minutes against making summarizations manually

Validation Strategies

- **User Testing:** Gather feedback from target users to see if generated summaries are relevant and useful
- **Functional Testing:** Checking if the generated summaries are correctly converted and uploaded to Supabase's file storage

Flashcard Generation

Evaluation Metrics

- **Correctness of Generated Q&As(%):** % of flashcards that have correct questions and answers
- **Format Accuracy(%):** % of flashcards that follow the correct Q&A format

Validation Strategies

- **User Testing:** Gather feedback from target users alongside manual review of reference material to check if the generated flashcards are correct or not
- **Functional Testing:** Checking if the generated flashcards are correctly parsed and displayed from the Resources tab

Assignment/Quiz Generation

Evaluation Metrics

- **Correctness of Generated Q&As(%):** % of flashcards that have correct questions and answers
- **Format Accuracy(%):** % of flashcards that follow the correct Q&A format
- **Quiz Content Coherence + Clarity + Relevance (score):** Score assigned to all quiz questions based on these metrics by the LLM
- **Assignment Content Coherence + Clarity + Relevance (feedback):** Feedback and suggestions for improvements/changes provided by humans

Validation Strategies

- **User Testing:** Manual review of reference material to check if the generated quizzes and assignments are correct or not
- **Functional Testing:** Checking if the generated assessments are correctly parsed and displayed from the Resources tab

Assignment/Quiz Grading

Evaluation Metrics

- **Code quality, readability, implementation, and completeness:** LLM evaluates the assignment submitted based on these metrics and grades it out of the percentages assigned to each
- **Instructor's contentment in generated feedback:** The instructor can grade a particular assignment/quiz and delete the feedback generated if they aren't satisfied with it

Validation Strategies

- **User Testing: The** Instructor can review the feedback and marks and delete it, and can be prompted to grade again if they aren't happy about the grading before releasing it to students
- **Functional Testing:** Checking if the generated summaries are correctly converted and uploaded to Supabase's file storage

Mark Scheme/Rubric Generation

Evaluation Metrics

- **LLM as a Judge:** LLM evaluates its own mark scheme based on completeness, logic, and clarity, and gives a score that is compared against a threshold

Validation Strategies

- **Error Testing:** Making sure that the mark scheme is not generated if it falls below the criteria and the instructor has to prompt the LLM to do it again
- **Functional Testing:** Checking if the generated mark schemes are correctly converted and uploaded to Supabase's file storage

Result Analysis

Summarization

Strengths

- Summaries were clear, concise, and aligned with prompt guidelines, scoring an average relevance of 4.
- Saved users 30–45 minutes on average compared to manual note-taking.

Weaknesses

- Some summaries lacked depth for technical lectures.
- Oversimplification occurred, especially when multiple lectures were summarized by chaining shorter summaries.

Flashcard Generation

Strengths

- Generated flashcard JSON files followed the correct Question Answer format 100% of the time, allowing for smooth integration with the frontend side
- Testing with a small sample of users revealed that they appreciated the feature that allowed for quick revision and self-testing to check their preparation

Weaknesses

- Manual review of generated Q&As revealed an error rate of 10%, where the model hallucinated in scenarios where relevant information was not present in the context

Assignment/Quiz Generation

Strengths

- Generated assessments followed the correct PDF/JSON formats 100% of the time, allowing for smooth integration with the frontend side
- Testing with a small sample of users revealed that they appreciated the feature that saved about 1-2 hrs as compared to manually creating the assessments.

Weaknesses

- When a relevant lecture wasn't added and the prompt wasn't specific enough, it would generate an assessment on an irrelevant topic. Example: with the prompt *'tokens'* and no lecture added, it may generate an assessment JSON Web Tokens instead of LLM tokens and embeddings.
- Prompt engineering skills required at instructor side - quality of the prompt and specification of topics affects quality of generated content significantly (especially during regeneration with feedback).

Assignment/Quiz Grading

Strengths

- Provides detailed, personalized feedback highlighting weak areas and improvement suggestions.
- Automates and standardizes grading, saving time and enhancing consistency compared to manual grading.

Weaknesses

- Feedback quality depends on the LLM; weaker models may generate vague responses.
- No "Grade All" feature yet, as individual grading yields better accuracy. Future updates could explore scalable group evaluation strategies.

Mark Scheme/Rubric Generation

Strengths

- Comprehensive mark schemes include correct and alternative answers with grading notes, enhancing clarity and consistency, especially valuable for reasoning-based questions.
- For coding assignments, rubrics outline marking criteria without revealing code, balancing transparency and academic integrity.

Weaknesses

- Some rubrics lacked completeness or detailed grading notes. Quality can improve with additional evaluation metrics or a more advanced LLM. The formatting of the code/task blocks in the mark scheme for assignment is a bit out of place. It will be made better in future iterations

Team Member	Work Done	Total Percentage Contribution
Junaid Jamshed	Assignment Mark scheme and rubric generation, Assignment Grading, Assignment Submissions, Quiz Submissions	20%
Jawad Saeed	Repository Set Up, Material UI Integration, Schema Design, Supabase Auth Integration, User Hierarchy Design, Course Creation Functionality, Summarization implemented, Flashcard Generation implemented	20%
Muhammad Ahmad	RAG and Embedding Generation on Upload Functionality, Joint contribution in Quiz and Assignment Generation, Chat with your lectures feature, fine-tuning hosting	20%
Huraira Anwer	Assignment and Quiz Generation, Agentic workflows for both, extensive prompt testing, initial fine-tuning attempts	20%
Ibrahim Farrukh	Assignment/Quiz Mark scheme and rubric generation, Assignment/Quiz Grading, Assignment/Quiz Submissions	20%