Comprehend AI

~ Sneha P Pratap

This project is an AI-powered web application that generates reading comprehension questions based on user-input passages and selected grade levels (1–12). Using Groq's Llama language model, the system produces diverse question types—such as multiple-choice, true/false, and short answers—tailored to the reading ability of each grade. Built with a FastAPI backend and a React frontend, the tool combines prompt engineering and LLM capabilities to support automated, curriculum-aligned assessment creation.

1. System Architecture

The system is built as a modular full-stack application using:

- **Frontend**: React with a clean CSS UI for passage input, grade selection, and results display.
- **Backend**: FastAPI in Python handles request routing and calls the Groq API.
- **LLM API**: Groq's `llama-3.1-8b-instant` model is used for its balance of speed and quality.
- **LLM Prompting**: A custom prompt is constructed dynamically based on user grade level and passage input.

2. Flow of Operation

1. User enters a passage and selects a grade level (1–12).
2. Backend builds a prompt with this info and sends it to Groq API.
3. Groq responds with question-answer pairs based on passage.
4. The result is cleaned (e.g., `**` removed) and displayed on the frontend.

3. Prompt Engineering Strategy

Prompts are explicitly constructed to:

- Instruct the model to behave like a subject matter teacher.
- Specify the number and type of questions (MCQ, True/False, Short Answer).
- Request an answer key after each question.
- Adapt tone and complexity to the target grade level.

Example of prompt:

```
You are an expert teacher creating reading comprehension questions for Grade {grade_level}.
Passage:
\"\"\"{passage}\"\"\"

Generate at least 5 questions of diverse types (multiple-choice, short answer, true/false).
Provide an answer key too.
Questions should be relevant, clear, and grade-appropriate.
Format:
Q1. [question]
A1. [answer]
Q2. ...
```

4. Key Design Decisions

- **FastAPI** for async capabilities and easy CORS integration.
- **React** for fast UI rendering and user interaction.
- **Groq** chosen over OpenAI for faster performance and simple API.
- **CSS Gradients** in frontend for engaging UX (blue + purple palette).
- **Markdown-cleaning** to remove symbols like **.

5. Outcomes

- Fully functional application supporting input passages and live LLM-based question generation.
- Demonstrated adaptability of LLM across education levels.
- Successfully met assignment goals, including API integration, frontend UI, and backend orchestration.