TASK: AI-Powered Assessment Tool

The goal is to create a web app that takes uploaded educational content (such as lecture notes, articles, or textbook excerpts) and converts it into an interactive assessment and tutoring workflow. The app should allow students to upload a document and then either generate an assessment (for example, a short essay task or a set of 20 multiple-choice questions) or engage in a tutoring session where follow-up questions about the content can be answered. In both cases, the system should automatically grade any submitted answers and provide feedback to the student.

Challenge Tasks

Candidates should implement the following core tasks:

1. **Document Upload and Content Generation:** The application should allow a user to upload an educational document (e.g., a PDF or text file). The system should then use a large language model to generate one of the following:

    1.1. An essay topic with clear guidelines and helping material asking the student to write a 300–400 word writing piece based on the content. The generated essay topic should be presented clearly in the user interface.

    1.2. a set of 20 multiple-choice questions (including answer choices, correct answers, an explanations). The generated quiz should be presented clearly in the user interface.

2. **Answer Submission and Automatic Grading:** The app should accept user responses to the generated assessment. For an essay, the user will submit a written answer; for MCQs, the user will select choices. The system must automatically evaluate these answers and provide immediate feedback or scores. Use an AI model to compare user answers against the expected answers or key points from the content, and display results indicating correct answers and guidance on incorrect ones.

3. **Tutoring Chatbot Interface:** Include a chat-based tutoring interface in the application. After an assessment, the user can ask follow-up questions about the uploaded content or about their answers. The chatbot, powered by an AI language model, should respond with helpful explanations, clarifications, or hints based on the material. The chat interface should be easy to use and display conversation history.

4. **Front-End Development:** Build a user-friendly front-end that ties all components together. A modern web framework (React or plain JS is preferred) should be used to create a clean interface. The UI must allow users to upload documents, view generated questions or essay prompts, submit answers, see grades/feedback, and interact with the tutoring chatbot. Ensure the design is intuitive and responsive.
5. **Containerization and Deployment:** Package the entire application for easy deployment. Provide Docker configurations (Dockerfiles) for each component and include setup for container orchestration (Kubernetes is preferred). The solution should demonstrate how the back-end services and front-end can run in containers, making it ready for cloud deployment.

Suggested Technologies (Optional)

Candidates may choose any technology stack they prefer. Suggested options include:

- **Backend/API:** Python with FastAPI or Flask (or any server-side framework)
- **Front-End:** React (preferred) or another modern JavaScript framework
- **AI/LLM:** Any large language model or AI API (for example, GPT-4.1-mini via OpenAI, or open-source models like llama or deepseek). If you do not have access, please use the following **api key** to access your preferred models via OpenRouter.
    - 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
- **Containerization:** Docker for building container images, and Kubernetes (or a similar platform) for orchestration

The above are only suggestions; feel free to use other libraries and tools as long as they meet the requirements.

Submission Guidelines

Please submit your solution with the following deliverables:

- A GitHub repository containing all source code for the project.
- A comprehensive **README** file that includes:
    - Setup and installation instructions for running your application.
    - A list of dependencies or requirements.
    - Explanations for any incomplete or unimplemented features.
- Ensure your code is well-organized and commented where necessary. Include any additional instructions needed to build and run the app.

Evaluation Criteria

Submissions will be evaluated based on:

- **Functionality:** How completely the solution meets the stated requirements and tasks.
- **Problem-Solving:** The effectiveness and creativity of your approach to each task.
- **Code Quality:** Readability, structure, and maintainability of your code.
- **Documentation:** Clarity of the README and inline comments; how well you explain your design and setup.
- **User Experience:** The usability and polish of the front-end interface (is it intuitive and user-friendly?).
- **Scope Management:** Appropriate feature scope and clear notes on any missing or unfinished parts, demonstrating good time management.

Disclaimer

All code, materials, and design provided by the candidate in response to this programming challenge are intended solely for the purposes of assessment and evaluation. None of the code or related content will be used for any commercial, production, or revenue-generating purposes.

Timeline

You have **72 hours** from the start of this challenge to complete and submit your solution. Please manage your time wisely to focus on core functionality.

Good luck!
