Plan de Proyecto: Tutor Inteligente Personalizado (MVP con Desarrollo a Medida)

Plan de Proyecto: Tutor Inteligente Personalizado (MVP con Desarrollo a Medida)

Objetivo General: Desarrollar un Producto Mínimo Viable (MVP) de un sistema de tutoría inteligente, con prompts ocultos, acceso mediante clave, botones de acción rápida para guía del alumno e identificación de ejercicios para registro de corrección, utilizando un enfoque de desarrollo a medida con Python/Flask para una prueba piloto inicial.

Fase 0: Preparación y Definición

Paso 0.1: Definición del Alcance del MVP

Descripción: Aclarar qué funcionalidades mínimas e indispensables tendrá la versión inicial para la prueba piloto.

Tareas:

Confirmar el tema(s) matemático(s) inicial a cubrir. Confirmado: Matemáticas y Física para secundaria/bachillerato.

Definir el tipo de interacción IA (chat simple, múltiples opciones, etc.). Confirmado: Chat con botones de acción rápida e identificación de ejercicios.

Establecer la cantidad de alumnos para la prueba piloto. Confirmado: 10 alumnos, escalable.

Especificar los datos mínimos a registrar (ej. interacciones, corrección). Confirmado: Interacciones, corrección, nivel de dificultad de los ejercicios, grado de avance.

Estado: Completado

Paso 0.2: Herramientas y Entorno de Desarrollo

Descripción: Asegurar que se tienen las herramientas básicas instaladas y configuradas.

Tareas:

Instalar Python 3.8+ (si no está).

Instalar un editor de código (VS Code, Sublime Text, etc.).

Crear cuenta en OpenAI y obtener la API Key.

Estado: completado

Nuevas consideraciones para la implementación:

1. **Para los "Botones de Acción Rápida**":

Diseño del Prompt: El prompt ahora necesitará instruir al IA no solo qué decir, sino también qué "sugerencias" de acciones puede ofrecer después de ciertas interacciones. Podríamos usar un formato específico en la respuesta del IA.

Ejemplo: El IA responde: "Aquí tienes un ejemplo. ¿Quieres [Otro Ejemplo] o [Ponme un Ejercicio]?"

Tu frontend parsearía la respuesta del IA, detectaría estos [...] y los convertiría en botones clicables.

Frontend: El JavaScript del chat necesitará la lógica para detectar estos "botones" en la respuesta del IA y renderizarlos dinámicamente.

Backend: Cuando el alumno haga clic en un botón, tu frontend enviaría ese texto (Otro Ejemplo, Ponme un Ejercicio) como el siguiente mensaje del alumno al backend, que a su vez se lo pasa al IA. El IA, al ver ese "mensaje", sabrá cómo responder.

2. Para la "Identificación de Ejercicios":

Diseño del Prompt: Necesitaremos que el IA encierre los ejercicios con tokens específicos.

Ejemplo: "Aquí tienes un ejercicio: [EJERCICIO_INICIO] Resuelve la ecuación: 2x + 5 = 11 [EJERCICIO_FIN]."

Tu backend detectará [EJERCICIO_INICIO] y [EJERCICIO_FIN] en la respuesta del IA.

Backend: La función get_ai_response (o una función posterior que procese la respuesta del IA) en tu backend necesitará lógica para:

Detectar estos tokens.

Marcar la sesión como "esperando_respuesta_ejercicio" cuando se detecta un [EJERCICIO_INICIO].

Cuando llegue la siguiente respuesta del alumno, si la sesión está marcada como "esperando_respuesta_ejercicio", tu backend sabrá que esa respuesta es un intento de solución al ejercicio.

Crucial: Necesitarás una forma de que el IA evalúe si la respuesta del alumno es correcta o no. Esto puede hacerse pidiéndole al IA que, después del token [EJERCICIO_FIN], también incluya la [SOLUCION_OCULTA: 3] o que tú le pidas al IA una evaluación separada de la respuesta del alumno al ejercicio. La primera opción es más limpia para el MVP.

Base de Datos (Actualización): La tabla de interacciones o una nueva tabla exercises necesitará campos para registrar:

exercise_text

expected_answer (si el IA la proporciona en el prompt o la calcula)

student_answer

is_correct

difficulty_level

session_id

¡Vamos al siguiente paso!

Fase 0: Preparación y Definición

Paso 0.2: Herramientas y Entorno de Desarrollo

Descripción: Asegurar que se tienen las herramientas básicas instaladas y configuradas.

Tareas:

Instalar Python 3.8+ (si no está).

Instalar un editor de código (VS Code, Sublime Text, etc.).

Crear cuenta en OpenAI y obtener la API Key.

Estado: completado

Fase 1: Configuración del Entorno de Desarrollo

- **Paso 1.1: Inicializar el Proyecto Python**
    - **Descripción:** Crear la estructura base de carpetas y el entorno virtual para el proyecto.
    - **Tareas:**
        - Crear la carpeta principal del proyecto (ej. mi_tutor_ia/).
        - Abrir la terminal en esa carpeta.
        - Crear el entorno virtual: python -m venv venv.
        - Activar el entorno virtual.
    - **Estado:** completado

- **Paso 1.2: Instalar Dependencias del Backend**
    - **Descripción:** Instalar las librerías necesarias para el servidor web y la interacción con IA.
    - **Tareas:**
        - Instalar Flask: pip install Flask.
        - Instalar la librería de OpenAI: pip install openai.
        - Instalar python-dotenv para manejar variables de entorno: pip install python-dotenv.
        - Crear el archivo requirements.txt: pip freeze > requirements.txt.
    - **Estado:** Pendiente

Fase 2: Backend - Configuración Inicial y Base de Datos

- **Paso 2.1: Archivo de Configuración de Entorno**
    - **Descripción:** Configurar un archivo para almacenar la API Key de OpenAI de forma segura.
    - **Tareas:**
        - Crear un archivo .env en la raíz del proyecto.
        - Añadir OPENAI_API_KEY="tu_clave_aqui" al archivo.
        - Añadir .env a .gitignore (para no subirlo a repositorios públicos).
    - **Estado:** Pendiente
- **Paso 2.2: Configuración de la Base de Datos (SQLite)**
    - **Descripción:** Crear la base de datos y la tabla para almacenar los prompts personalizados y las claves de acceso.
    - **Tareas:**
        - Crear un archivo database.py.
        - Definir función init_db() para conectar a SQLite y crear la tabla prompts con columnas: id, student_email, topic, prompt_content, access_key, created_at.

        - Definir función add_prompt(student_email, topic, prompt_content, access_key).
        - Definir función get_prompt_by_key(access_key).
    - **Estado:** Pendiente
- **Paso 2.3: Aplicación Flask Principal (app.py)**
    - **Descripción:** Inicializar la aplicación Flask y conectar la base de datos.
    - **Tareas:**
        - Crear el archivo app.py.
        - Importar Flask y load_dotenv.
        - Inicializar la aplicación Flask.
        - Llamar a init_db() al iniciar la aplicación para asegurar que la base de datos y la tabla existan.
    - **Estado:** Pendiente

Fase 3: Backend - Funcionalidades Clave

- **Paso 3.1: Generación de Claves de Acceso**
    - **Descripción:** Crear una función para generar claves únicas y seguras.
    - **Tareas:**
        - En app.py (o en un módulo utils.py), definir generate_unique_access_key().
        - Asegurarse de que las claves generadas sean aleatorias y suficientemente largas.
    - **Estado:** Pendiente
- **Paso 3.2: Lógica de Interacción con OpenAI**
    - **Descripción:** Crear la función que envía el prompt combinado y el mensaje del alumno a la API de OpenAI.
    - **Tareas:**
        - En app.py, definir get_ai_response(system_prompt, user_message).

        - Usar openai.ChatCompletion.create con el parámetro messages correctamente estructurado.
        - Manejar posibles errores de la API.
    - **Estado:** Pendiente
- **Paso 3.3: Ruta para Crear y Guardar Prompts (Admin)**
    - **Descripción:** Crear una ruta sencilla para que tú puedas ingresar los datos del alumno, el prompt y generar una clave.
    - **Tareas:**
        - Crear una ruta /admin/create_prompt en app.py.
        - Implementar un formulario HTML (en templates/admin_create.html) para ingresar email, topic y prompt_content.
        - Al enviar el formulario (POST request), llamar a generate_unique_access_key() y add_prompt() de database.py.
        - Retornar la clave de acceso generada al usuario.
    - **Estado:** Pendiente

Fase 4: Frontend - Interfaz de Usuario Básica

- **Paso 4.1: Estructura de Plantillas HTML**
    - **Descripción:** Crear las carpetas y los archivos HTML principales.
    - **Tareas:**
        - Crear la carpeta templates/.
        - Crear templates/index.html (página de bienvenida/login con clave).
        - Crear templates/chat.html (interfaz del chat).
        - (Opcional) Crear templates/base.html para elementos comunes.
    - **Estado:** Pendiente
- **Paso 4.2: Página de Inicio (Input de Clave)**

- **Descripción:** Implementar la interfaz para que el alumno ingrese su clave de acceso.
- **Tareas:**
  - En app.py, crear la ruta / que renderice index.html.
  - Diseñar un formulario en index.html con un campo de texto para la clave.
  - Cuando se envía el formulario, redirigir a /chat/<access_key> si la clave es válida.
- **Estado:** Pendiente

- **Paso 4.3: Interfaz de Chat (Frontend JS)**
  - **Descripción:** Implementar la interfaz de chat donde el alumno interactúa con el IA.
  - **Tareas:**
    - En app.py, crear la ruta /chat/<access_key> que renderice chat.html.
    - Dentro de chat.html:
      - Un contenedor para los mensajes.
      - Un campo de entrada para el mensaje del alumno.
      - Un botón de "Enviar".
    - Crear la carpeta static/ y static/js/chat.js.
    - Escribir el JavaScript en chat.js para:
      - Manejar el evento de envío del formulario (o clic en el botón).
      - Obtener el mensaje del alumno.
      - Realizar una llamada fetch (POST) a un endpoint /api/chat en el backend (que crearás a continuación), pasando el mensaje del alumno y la access_key.
      - Actualizar la interfaz de usuario con la respuesta del IA.
  - **Estado:** Pendiente

Fase 5: Conexión Frontend-Backend y Lógica de Chat

- **Paso 5.1: Endpoint API para el Chat**
    - **Descripción:** Crear un endpoint en el backend que el frontend llamará para obtener respuestas del IA.
    - **Tareas:**
        - En app.py, crear la ruta /api/chat (POST).
        - Extraer la access_key y el user_message de la solicitud JSON.
        - Validar la access_key contra la base de datos para obtener el prompt_content asociado.
        - Llamar a get_ai_response(prompt_content, user_message).
        - Devolver la respuesta del IA como JSON al frontend.
    - **Estado:** Pendiente
- **Paso 5.2: Mejora de la Interfaz de Chat (Opcional MVP)**
    - **Descripción:** Añadir estilos básicos para que el chat sea más usable.
    - **Tareas:**
        - Crear static/css/style.css.
        - Añadir estilos para los mensajes, el campo de entrada, etc.
    - **Estado:** Pendiente

Fase 6: Prueba Local y Refinamiento del MVP

- **Paso 6.1: Pruebas Internas**
    - **Descripción:** Probar todas las funcionalidades en tu máquina local.
    - **Tareas:**
        - Generar varios prompts de prueba usando la interfaz /admin/create_prompt.
        - Acceder con las claves generadas y simular sesiones de tutoría.
        - Verificar que el IA se comporta según el prompt.

        - Identificar bugs o fallos de lógica.
    - **Estado:** Pendiente
- **Paso 6.2: Refinamiento de Prompts**
    - **Descripción:** Ajustar la redacción de los prompts para obtener el comportamiento deseado del IA.
    - **Tareas:**
        - Experimentar con diferentes formulaciones y ejemplos.
        - Asegurarse de que el IA no dé respuestas directas si no debe, etc.
    - **Estado:** Pendiente

Fase 7: Prueba Piloto Externa y Feedback

- **Paso 7.1: Despliegue para Prueba Piloto**
    - **Descripción:** Subir la aplicación a un servicio de hosting para que los alumnos de la prueba piloto puedan acceder.
    - **Tareas:**
        - Seleccionar un servicio de hosting de bajo costo (ej. Render, Heroku).
        - Seguir las instrucciones para desplegar una aplicación Flask.
        - Asegurar que la base de datos funcione correctamente en el hosting (cambiar a PostgreSQL si SQLite no es ideal para el hosting).
    - **Estado:** Pendiente
- **Paso 7.2: Reclutamiento y Lanzamiento de Prueba Piloto**
    - **Descripción:** Contactar a los alumnos seleccionados y darles acceso.
    - **Tareas:**
        - Enviar los emails con el enlace y la clave de acceso.
        - Proporcionar instrucciones claras de uso.
    - **Estado:** Pendiente

- **Paso 7.3: Recolección de Feedback**
    - **Descripción:** Reunir opiniones y datos de los participantes de la prueba.
    - **Tareas:**
        - Crear encuestas de satisfacción y usabilidad.
        - Programar entrevistas con algunos alumnos.
        - Monitorear el uso y las interacciones (si implementaste el registro en la DB).
    - **Estado:** Pendiente
- **Paso 7.4: Análisis y Conclusiones del MVP**
    - **Descripción:** Evaluar los resultados de la prueba piloto y decidir los siguientes pasos.
    - **Tareas:**
        - Analizar los datos y el feedback.
        - Identificar fortalezas y debilidades.
        - Proponer mejoras y funcionalidades para futuras iteraciones.
    - **Estado:** Pendiente
