Demo Project Proposal

AI-Powered Developer Assistant

# Statement of Purpose

* + To build a cross-platform .NET MAUI application that integrates with a large language model (OpenAI API or Azure OpenAI) to act as a coding copilot for developers. The assistant will help generate, explain, and debug C#/.NET code snippets in a simple, chat-based interface.
  + This project demonstrates the potential of applying AI directly to developer workflows. It will serve as a foundation for a more advanced intelligent IDE companion as a final project.

Purpose:

* Provide developers with an AI-powered assistant that improves coding productivity.
* Simplify debugging by offering context-aware suggestions and explanations.
* Explore integration of modern AI services into .NET applications.
* Lay the groundwork for a scalable enterprise developer tool.

# Runtime Environment

* **Frontend:** .NET MAUI (cross-platform desktop/mobile)
* **Backend:**
  + OpenAI API or Azure OpenAI Service (for LLM responses)
  + Optional Azure Functions for backend orchestration
* **Database:** SQLite (for local chat history persistence)

# Information Needed

* **User Input:** Code snippets, error messages, or natural language questions.
* **LLM Response:** Suggestions, explanations, optimizations, or generated code.
* **Quick Action Prompts:** Predefined actions like *Explain Code*, *Fix Errors*, *Optimize Code*.

# Data to be Persisted (Data Base)

* Chat history (user queries + AI responses).
* Timestamps of interactions.
* (Optional) User preferences, such as model selection or theme.

# App Concerns

* **Latency:** API response times for LLM queries.
* **Context Size:** Token limits may restrict length of code snippets.
* **Error Handling:** Handle timeouts, API failures, or invalid responses.
* **Privacy:** Avoid storing sensitive code/data without user consent.
* **Cross-Platform Testing:** Ensuring consistency across Windows, macOS, and Android.

# User Interface Outline and Functional Flow

**Screen 1: Chat Interface**

* Text input box for code/questions
* Chat bubble responses (user + AI)
* Buttons for Quick Actions (Explain Code, Fix Errors, Optimize)

**Screen 2: History (Optional for Demo)**

* List of past queries/responses
* Ability to reload or delete sessions

**Flow:**

1. User launches app.
2. User enters question or pastes code.
3. App sends request to LLM API.
4. Response is displayed in chat interface.
5. Interaction is stored in SQLite.

# Special Features

**Demo (2 weeks):**

* Core chat interface with LLM integration.
* Quick Action buttons for specialized prompts.
* SQLite storage for chat history.

**Stretch (Final Project Potential):**

* GitHub repo integration (context-aware code analysis).
* Azure Functions for backend workflow logic.
* Voice-to-text support for mobile developers.
* Personalized assistant with memory (per-user context).

# Summary

This mini-project demonstrates how AI can enhance software development workflows by embedding an LLM into a cross-platform developer tool. The .NET MAUI app will serve as both a **demo showcase** and a **foundation for a larger final project**, positioning it as a strong portfolio piece for cloud-native and AI-integrated solutions.