AI Assisted IDE

Overview

This is an application where a user can generate a full project from an AI prompt, run it behind the scenes, and interact with the user interface. When the user wants to modify the code, he selects an element in the interface such as a button and asks the AI to modify it. the system provides the AI with the relevant code context. The AI will then return an updated code snippet, and the system will apply the changes to the user's local project codebase. To achieve this, we will be indexing the entire codebase, perform semantic search, and allow us to fetch the right piece of code whenever needed. The relevant tools are, CocoIndex and CodeIndexer(zilliz).

Required Tools

CodeIndexer : An open-source tool to index your codebase into a vector database for semantic searches. It is nodejs based, it chunks code smartly by functions using Abstract Syntax Trees (AST) and uses local models like Ollama or Open AI's free tier apis for embeddings.

CocoIndex: An open source tool to index codebase . It has multi-language support but requires Python dependencies.
React and Vite: For the IDE's UI and running generated projects in an iframe for live previews.

FAISS/Pinecone: Vector databases for local or cloud storage for storing vector embeddings(index).

Ollama or Hugging Face Models: Free, local AI models for embedding code.

JavaScript parsers to apply AI-generated code changes accurately.

Workflow of the System

The process begins when the user provides an initial prompt such as "build me an ecommerce app in React and Node.js." The AI generates a complete project codebase which is stored locally. The folders and files, contents will be generated based on the response that will contain folder structure . Next, the codebase is indexed using a tool such as CocoIndex or CodeIndexer. Indexing basically means, creating a structured representation of codebase into mathematical representation called vectors that is easily searchable later on. It is like a map to the entire codebase. We can use FAISS or any cloud vector db to store the vector embeddings.The indexers such as Cocoindex or Codeindexer will scan all the files, breaks them into meaningful parts such as functions or components, and stores them as vectors inside the database. For indexing, metadata will also be used for storing the exact context of the code. Cocoindex or Codeindexer uses metadata for indexing on their own, it can be used later for updating the local codebase.

Once the user is interacting with the running application, they may select a UI element such as a button. The system identifies which part of the code corresponds to this element by using the index. It can retrieve the relevant code or function from the index that was previously created. When the user provides a new instruction, for example "change this button's color to red," the system sends this prompt along with the index and the relevant code snippet to the AI model. The AI then modifies the snippet accordingly and returns an updated version. Finally, the system replaces the old code in the local project inside the exact component or file with the updated one and re-runs the application so the user can immediately see the effect of the change in the interface.