# Copilot 140

## 1. Introduction

Copilot 140 is a Python-based AI tool designed to assist developers in generating and executing code snippets directly from a web interface. It leverages the Gemini API for generating code based on user prompts and integrates with JDoodle API to execute the generated code in various programming languages. The project was developed using Streamlit, which provides a user-friendly interface for interacting with the AI model and executing code.

## 2. Project Setup

### 2.1 Environment Configuration

* Programming Language: Python
* Web Framework: Streamlit
* APIs Integrated:
  + Gemini API: Used for generating code snippets based on user input.
  + JDoodle API: Used for executing the generated code in selected programming languages.

### 2.2 Dependencies

```bash  
pip install streamlit requests google-generativeai  
```3. Core Functionality

### 3.1 Gemini API Integration

The Gemini API is integrated into Copilot 140 to generate code snippets based on user prompts. Users input a natural language prompt describing the code they need, and the API returns the corresponding code snippet.  
  
API Configuration: The Gemini API key is set up using environment variables for security.  
Code Generation: A dedicated function `generate\_code(prompt)` is implemented to send the user's prompt to the Gemini API and retrieve the generated code.

### 3.2 JDoodle API Integration

The JDoodle API allows Copilot 140 to execute the generated code in different programming languages like Python, Java, and C.

* Execution Function: The `run\_code\_with\_jdoodle(code, language, version)` function sends the generated code to JDoodle for execution and returns the output.
* Language Support: A mapping is created to handle different language versions supported by JDoodle.

### 3.3 User Interface (UI)

The UI is built using Streamlit, providing a clean and intuitive interface for users to interact with Copilot 140.

Control Sets: Users can create multiple control sets, each consisting of:

* A dropdown for selecting the programming language.
* A text area for entering the prompt.
* A 'Generate' button to create the code snippet.
* A text area for modifying the generated code.
* A 'Run' button to execute the code and display the output.

### 3.4 Dynamic Controls

Copilot 140 allows users to dynamically add and remove control set, providing flexibility in managing multiple code generation tasks.  
Add Cell: Adds a new code cell for generating and running code.  
Delete Cell: Removes a selected cell.

## 4. User Interaction Flow

1. Select Language: Users choose the programming language for code generation.  
2. Enter Prompt: Users provide a descriptive prompt in natural language.  
3. Generate Code: The 'Generate' button triggers code generation via Gemini API.  
4. Modify Code (Optional): Users can modify the generated code in the text area.  
5. Run Code: The 'Run' button executes the code via JDoodle API, and the output is displayed below the code box.

## 5. Advantages

* Time Efficiency: Quickly generates and executes code, saving developers time on routine tasks.
* User-Friendly Interface: Provides an intuitive UI with dynamic controls for easy task management.
* Language Flexibility: Supports multiple programming languages with customizable code generation.
* Enhanced Productivity: Integrates code generation and execution in one platform, streamlining development.
* Learning Tool: Acts as an educational resource, offering immediate feedback for learning and debugging.

**6.Conclusion**

Copilot 140 is a robust and interactive code generation and execution tool designed to enhance developer productivity. With its integration of advanced AI models and cloud-based execution environments, it provides a seamless experience from code ideation to execution.