**GENAI Code Assistant**

## Project Topic

SmartSDLC – AI-Enhanced Software Development Life Cycle

## Tools Used

* Python
* Flask
* HTML
* CSS
* JavaScript
* Hugging Face Transformers
* IBM Cloud API

## Features

1. Code Generator  
2. Debugger  
3. Explainer  
4. Optimizer

## Project Description

The GENAI Code Assistant is a cutting-edge tool developed as a part of the SmartSDLC initiative, aimed at enhancing the software development lifecycle using advanced Generative AI capabilities. This project empowers developers with intelligent tools that automate, explain, debug, and optimize code efficiently, boosting both productivity and code quality.

## Development Process (4 Weeks)

### Week 1: Planning and Setup

During the first week, the focus was on understanding the goals of SmartSDLC and identifying the features to be built. Research was conducted on existing AI-based coding tools. The project environment was set up using Flask for the backend and basic HTML/CSS/JS for the frontend. The UI design and structure were planned for simplicity and usability.

### Week 2: Integrating Hugging Face Models

In the second week, APIs from Hugging Face were explored to find suitable models for natural language code generation and explanation. The 'Granite' and other large language models were tested for tasks like code generation and explanation. Integration of these models into the Flask backend was achieved, allowing text prompts to generate relevant code snippets.

### Week 3: IBM Cloud Integration and Feature Expansion

Week three focused on enhancing the assistant by integrating IBM Cloud services for reliable and scalable AI inference. API keys were obtained and securely configured using environment variables. Additional features like code debugging and optimization were added, leveraging model outputs to identify bugs and suggest improvements.

### Week 4: Testing, Optimization, and Final Deployment

In the final week, the project underwent extensive testing. Various use-cases were simulated to ensure the assistant responded accurately. Optimizations were made to handle edge cases and improve model response times. Finally, the app was polished with a user-friendly interface and deployed locally for demonstration.