AI Code Analysis & Generator - Documentation

This document provides an overview and explanation of the Python code implementing an AI-powered requirement analysis and code generation tool using Gradio, Hugging Face Transformers, and PyPDF2.

# 1. Overview

The code creates a web-based interface where users can upload a PDF or input text describing software requirements, and then the system analyzes these requirements or generates code based on them.

# 2. Libraries Used

* gradio – For creating the web-based user interface.
* torch – For running the machine learning model.
* transformers – For loading and using the IBM Granite language model.
* PyPDF2 – For extracting text from uploaded PDF files.
* io – For handling file input/output.

# 3. Model Loading

The IBM Granite model (ibm-granite/granite-3.2-2b-instruct) is loaded using Hugging Face's transformers library. The model and tokenizer are configured to run on GPU if available or CPU otherwise. Padding tokens are also handled.

# 4. Functions

## 4.1 generate\_response(prompt, max\_length)

Generates a response from the AI model based on a given prompt. It tokenizes the input, sends it to the model, and decodes the generated output.

## 4.2 extract\_text\_from\_pdf(pdf\_file)

Extracts text content from an uploaded PDF file using PyPDF2. Returns the extracted text as a string.

## 4.3 requirement\_analysis(pdf\_file, prompt\_text)

Performs requirement analysis either on uploaded PDF content or on manually entered text. It classifies the requirements into functional requirements, non-functional requirements, and technical specifications using the AI model.

## 4.4 code\_generation(prompt, language)

Generates code snippets in a specified programming language based on the given requirements prompt.

# 5. User Interface

The Gradio interface uses Tabs for two main features: Code Analysis and Code Generation.

## 5.1 Code Analysis Tab

Allows the user to upload a PDF or input text describing requirements. Once the 'Analyze' button is clicked, the requirements are analyzed and displayed in a text box.

## 5.2 Code Generation Tab

Allows the user to enter requirements and choose a programming language. Once the 'Generate Code' button is clicked, the AI generates code in the selected language.

# 6. App Launch

Finally, the Gradio app is launched with app.launch(share=True), enabling public sharing of the interface.

# 7. Summary

This code combines natural language processing, PDF parsing, and a web interface to analyze requirements and automatically generate code, demonstrating an AI-powered development assistant.