SMART SDLC – AI – ENHANCED SOFTWARE DEVELOPMENT LIFECYCLE

1.Introduction

Team Leader : DIVYA DHARSHINI.R

Team Member : KAVYA.R

Team Member : PAVITHRA.R

Team Member : THEJA SREE.P

Team Member :THILAGAVATHY.S

2. Project Overview

- **Purpose**:The purpose of this project is to build an AI-powered tool that analyzes software requirement documents (PDFs or text input) and automatically generates code in different programming languages.
- It helps developers quickly transform requirements into structured formats and working code, reducing manual effort and speeding up the software development lifecycle.

Features:

Requirement Analysis

***Key Point*:** Structured analysis

***Functionality*:** Extracts functional, non-functional, and technical requirements.

Code Generation

***Key Point*:** AI-powered code writing

***Functionality*:** Generates code in Python, JavaScript, Java, C++, and more.

PDF Support

***Key Point*:** Document input flexibility

***Functionality*:** Extracts text from PDF files using PyPDF2.

Interactive UI (Gradio)

***Key Point*:** User-friendly interface

***Functionality*:** Provides a tabbed interface with sections for analysis and code generation.

3. Architecture

- **Frontend (Gradio)**:
  Provides an easy-to-use web UI with tabs for *Code Analysis* and *Code Generation*. Supports PDF uploads, text inputs, and real-time outputs.
- **Backend (Transformers + PyTorch)**:
  Uses IBM Granite LLM for analyzing requirements and generating code. Runs on CPU or GPU (if available) for efficient inference.
- **PDF Processing (PyPDF2)**:
  Extracts and processes text from uploaded documents for requirement analysis.

4. Setup Instructions

Prerequisites:

- Python 3.9 or later
- pip & virtual environment tools
- Required libraries: torch, transformers, gradio, PyPDF2

Installation Process:

1. Clone the repository.
2. Install dependencies:
3. pip install -r requirements.txt
4. Run the project:
5. python smartsdlc.py
6. A Gradio shareable link will open for usage.

5. Folder Structure

```
project/
│
├── smartsdlc.py   # Main project script
├── requirements.txt (optional)
└── (future modules can be added here)
```

6. Running the Application

- Launch the script with python smartsdlc.py.
- Access the Gradio UI in your browser.

Navigate between:

Code Analysis Tab → Upload PDF or type requirements → Get structured analysis.

Code Generation Tab → Describe requirement → Select language → Generate AI code.

7. API / Function Documentation

generate_response(prompt, max_length) – Sends a prompt to Granite LLM and returns response.

extract_text_from_pdf(pdf_file) – Reads and extracts text from PDF.

requirement_analysis(pdf_file, prompt_text) – Organizes requirements into categories.

code_generation(prompt, language) – Generates code in the specified language.

8. Authentication

(Current version runs in open environment. Future updates may add:)

- API Key / Token Authentication
- Role-based access control
- User session history tracking

9. User Interface

Sidebar with tabs:

Code Analysis → PDF upload / text input → Requirement breakdown.

Code Generation → Requirement input → Select language → Generated code output.

Real-time outputs: Textboxes display results instantly.

10. Testing

Unit Testing: Verified PDF text extraction and LLM response functions.

API Testing: Checked Gradio UI interactions with functions.

Manual Testing: Uploaded various PDFs and generated code outputs.

Edge Case Handling: Empty PDFs, large text inputs, invalid file formats.

11. Screenshots

12. Known Issues

- Large PDFs may take longer to process.
- Code generation accuracy depends on clarity of requirements.

13. Future Enhancements

- Multi-file project code generation.
- Direct code export (ZIP file).
- GitHub auto-commit integration.
- User authentication and session history.