SmartSDLC: AI-Powered Requirement Analysis & Code Generator

Introduction :

Traditional Software Development Life Cycle(SDLC)modelsinvolvemanualrequirement gathering, documentation, and coding, which are often time-consuming and prone to errors. With advancements in Artificial Intelligence (AI) and Large Language Models (LLMs), it is possible to automate requirement analysis, code generation, and documentation to make SDLC smarter, faster, and more reliable.

Smart SDLC leverages IBM Granite LLM, Gradio, and PyPDF2 to create an intelligent development assistant. The system extracts requirements from documents or user prompts,classifiesthemintofunctional,non-functional,and technical specifications,and generates working code in multiple programming languages.

Project Title :

Smart SDLC : AI-Powered Software Requirement Analyzer & Auto Code Generator

Team Leader : R RAJASRI
Team Members : K SAMYUKTHA
Team Members : E SURUTHIKA
Team Members : M SWETHA
Team Members : A SWETHIKA

Project Overview :

The Smart SDLC system is designed to helpd evelopers,students,andorganizations automate early stages of SDLC.
It provides:
- Requirement Analysis – from PDFs or manual input.
- Automatic Classification – into functional, non-functional, and technical requirements.
- Code Generation – in multiple programming languages (Python, Java, C++, etc.).
- InteractiveUI – using Gradio for real-time interaction.

Purpose :

- Reduce manual effort in requirement analysis.
- Generate quick boilerplate or prototype code.
- Provide real-time analysis and structured documentation.
- Act as an AI assistant for developers and students.

Features :

1. PDF Requirement Extraction
2. Requirement Analysis (Functional/Non-Functional/Technical)
3. Code Generation in multiple languages
4. Interactive Gradio Interface
5. AI-Driven structured responses

Architecture :

- Frontend: Gradio UI
- Backend: PyTorch + HuggingFace Transformers
- LLM: IBM Granite
- PDF Processing: PyPDF2

Setup Instructions :

Prerequisites :
- Python 3.9+
- Pip / Conda
- Torch + Transformers
- Gradio
- PyPDF2

Installation :
```
git clone
cd smart-sdlc
pip install -r requirements.txt
```

Run the app :
```
python smartsdlc.py
```

Folder Structure :

```
smartsdlc/
■■■smartsdlc.py
■■■requirements.txt
■■■docs/
■■■outputs/
```

Running the Application :

1. Launch the app: python smartsdlc.py
2. Open the Gradio link in browser.
3. Use 'Requirement Analysis' tab for structured requirement outputs.
4. Use 'Code Generation' tab for multi-language code generation.

API Documentation :

- generate_response(prompt): Generates AI-based response
- extract_text_from_pdf(file): Extracts text from PDFs
- requirement_analysis(pdf, prompt): Classifies requirements
- code_generation(prompt, language): Generates code

Known Issues :

- May generate generic code for vague requirements
- Large PDFs may cause memory issues

- Dependent on IBM Granite availability

Future Enhancements :

- Database integration for requirement storage
- Multi-language documentation generation
- Deployment-ready code templates
- UML diagram generation
- Project management dashboards

Objectives &Scope :

Objectives :
- Automate early SDLC phases
- Enhance productivity
- Provide intelligent requirement-to-code mapping

Scope :
- Requirement Analysis
- Code Generation
- Documentation Assistance

Use Case Scenarios :

- Student Project: Upload requirements □Get code skeleton
- Startup MVP: Generate prototype features
- Documentation: Convert raw requirements into structured docs

Technology Stack :

- Frontend: Gradio
- Backend: PyTorch + HuggingFace Transformers
- AI/LLM: IBM Granite
- PDF Processing: PyPDF2
- Supported Languages: Python, JavaScript, Java, C++, etc.

Performance Metrics :

- API response time <1s
- Requirement classification accuracy
- Code correctness via sample test cases

Limitations :

- Not a replacement for manual SDLC
- AI code may need debugging
- Requires internet for LLM

References :

- IBM Granite LLM Documentation
- HuggingFace Transformers Docs
- Gradio Developer Guide
- PyPDF2 Documentation

Conclusion :

The Smart SDLC Project demonstrates how AI can revolutionize the software development life cycle by automating requirement analysis,classification,and code generation.
While not are placement for developers,it serves as a powerful assistant,savingtime, reducing errors, and accelerating development.

With enhancements like UML generation,multilingual support,andreal-timemonitoring, Smart SDLC can evolve into a comprehensive AI-driven project assistant.