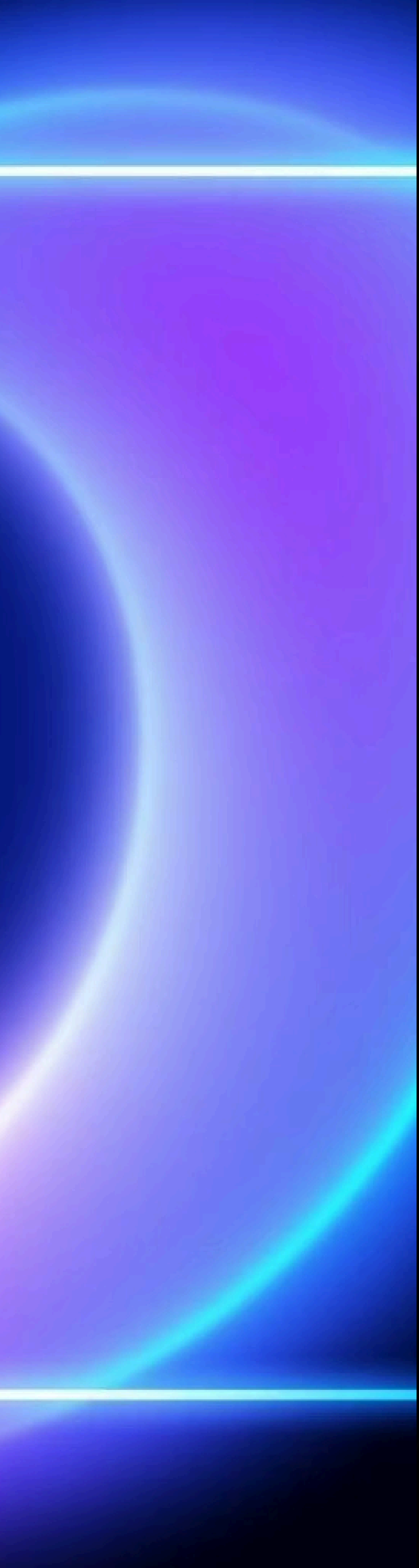


SmartSDLC: AI-Enhanced Software Development Lifecycle

SmartSDLC is an innovative AI-powered assistant designed to revolutionise the Software Development Lifecycle (SDLC). By leveraging advanced natural language processing and large language models, this tool streamlines critical stages from requirement analysis to code generation, enhancing efficiency and productivity for developers and analysts alike. This document outlines the project's core functionalities, system requirements, and future vision.

Our Dedicated Team

Meet the talented individuals behind SmartSDLC, a collaborative effort to bring AI innovation to the Software Development Lifecycle. Our team is committed to delivering a robust and intuitive solution that addresses key challenges in modern software engineering.

RUBAN D

Team Leader

JAGATH LAL K K

Team Member

RANJETH P

Team Member

HARINI V

Team Member

Team ID: NM2025TMID03731

Team Size: 4

SmartSDLC: System Requirements & Essential Packages

To ensure optimal performance and functionality of the SmartSDLC AI assistant, specific hardware and software configurations are recommended. Adhering to these guidelines will provide the best user experience. Additionally, several key Python packages are utilised to power its core features, enabling advanced AI capabilities and a responsive user interface.

System Requirements

Hardware (Minimum)

- CPU: Dual-core processor
- RAM: 4 GB
- Storage: 2 GB free space
- GPU: Not required (CPU-based operation)

Hardware (Recommended)

- CPU: Quad-core or better
- RAM: 8 GB or more
- Storage: 4 GB free space
- GPU: NVIDIA GPU with CUDA support (for faster model inference)

Software

- Operating System: Windows 10+, macOS, or modern Linux distribution
- Python Version: Python 3.8 or higher
- Web Browser: Chrome, Firefox, Edge, or any modern browser (for Gradio interface)

Python Packages Used

These fundamental Python packages underpin SmartSDLC's functionality, enabling everything from AI model execution to user interface rendering and document processing.

- **transformers:** Essential for loading and utilising the powerful IBM Granite model.
- **torch:** Provides the robust PyTorch backend necessary to efficiently run the AI model.
- **gradio:** Used to create the intuitive and user-friendly web interface for seamless interaction.
- **PyPDF2:** Crucial for reading and extracting text content from various PDF files, enabling document analysis.

SmartSDLC: AI-Enhanced Software Development Lifecycle

SmartSDLC is an AI-powered assistant designed to significantly enhance and automate key stages of the Software Development Lifecycle (SDLC). It leverages cutting-edge natural language processing (NLP) technologies and large language models (LLMs) to streamline development workflows. The tool analyses software requirements, extracts structured documentation, and generates clean, syntactically valid code across multiple programming languages, all within an interactive, user-friendly interface built with Gradio.

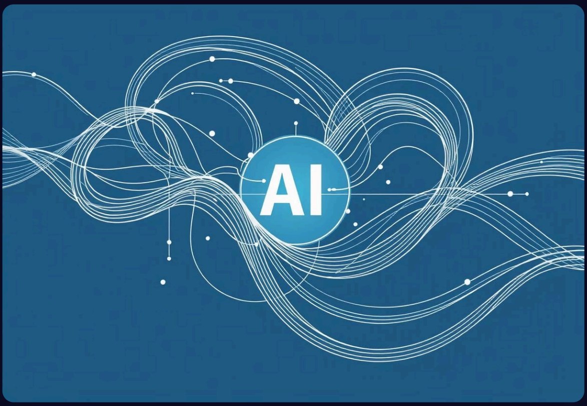
Key Features

Requirement Analysis (Text/PDF)

Upload PDF documents or manually input requirements. The AI intelligently extracts and organises functional, non-functional requirements, and technical specifications, providing a structured overview.

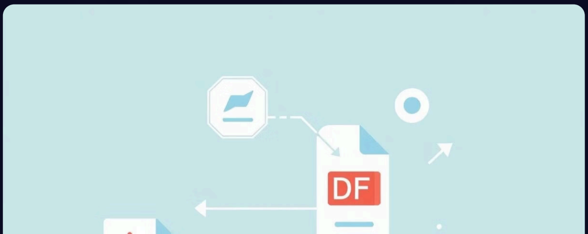
Automatic Code Generation

Generate runnable code (Python, JavaScript, Java, etc.) directly from natural language prompts. For Python, it includes syntax validation and sandboxed execution for safety and reliability.



PDF Parsing & Chunking

Utilises PyPDF2 for efficient extraction and chunking of content from uploaded requirement documents, preparing them for detailed AI analysis and processing.



Custom Gradio Interface

Features a stylish, responsive, and dark-themed user interface with organised tabs, ensuring clarity and an optimal user experience for all functionalities.

The Future of SDLC: Accessible, Efficient, Intelligent

SmartSDLC not only demonstrates the immediate benefits of AI in software development but also lays the groundwork for future advancements. It exemplifies how AI can transform the entire SDLC, making it more intuitive, powerful, and adaptable to evolving industry needs.

AI-Assisted Development

Showcasing how artificial intelligence can significantly assist in bridging the gap between high-level requirements and executable code, fostering innovation.

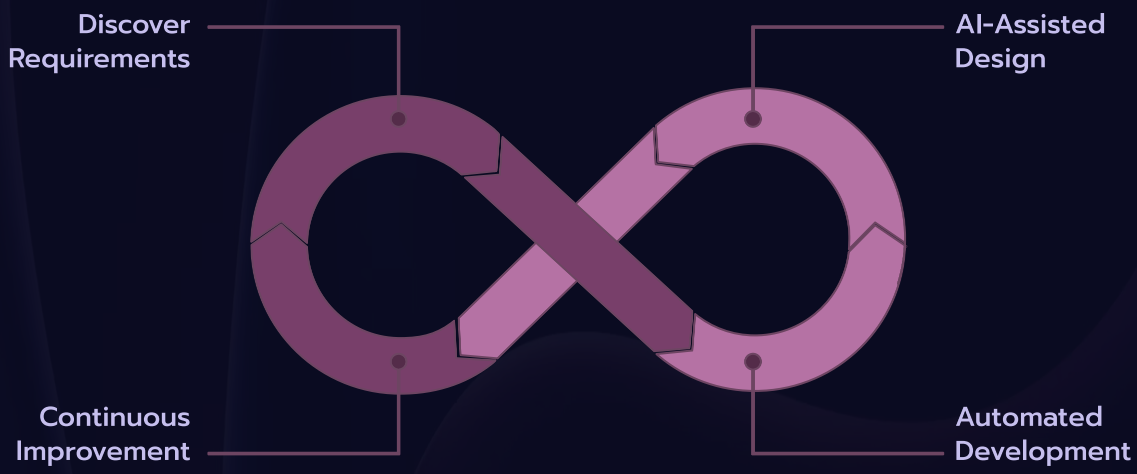
Enhanced Accessibility

Making the complex software development process more accessible to a wider range of users, including non-technical stakeholders, through intuitive design.

Scalable Architecture

Designed with a modular and scalable architecture, allowing for future expansion across the full spectrum of the SDLC, ensuring long-term viability.

This project paves the way towards smarter, AI-assisted software engineering, promising continuous innovation and improved outcomes in the realm of software creation. It represents a significant step towards a more automated and intelligent development future.



🔮 SmartSDLC: Future Enhancements

The SmartSDLC project is continuously evolving, with several exciting enhancements planned to further expand its capabilities and utility within the Software Development Lifecycle. These future developments aim to provide even more comprehensive support and automation, ensuring SmartSDLC remains at the forefront of AI-assisted development.

Support More File Types

Expand input capabilities to include .docx and .txt files, utilising libraries like python-docx for broader compatibility and data ingestion.

Add Full SDLC Phases

Integrate additional SDLC stages such as design, testing, and deployment, creating dedicated tabs within the Gradio interface for a complete workflow.

Safe Code Execution

Implement a secure, isolated environment for code execution, potentially using Docker or other safe Python methods to ensure integrity and safety.

Generate Test Cases

Automate the creation of comprehensive test cases for generated code, leveraging advanced AI prompts and Python's unittest module for robust validation.

API Integration

Enable seamless connection with external development tools like GitHub or JIRA by building a robust REST API, fostering ecosystem compatibility.

Multi-language Input

Allow requirements to be accepted in various languages, integrating translation tools such as the Google Translate API for global accessibility.

Add Dashboard

Introduce a comprehensive dashboard to display analysis results, includi
metrics and requirement coverage, using chart libraries like Plotly for visual insights.