**SMART SDLC-AI ENHANCED SOFTWARE DEVELOPMENT LIFECYCLE**

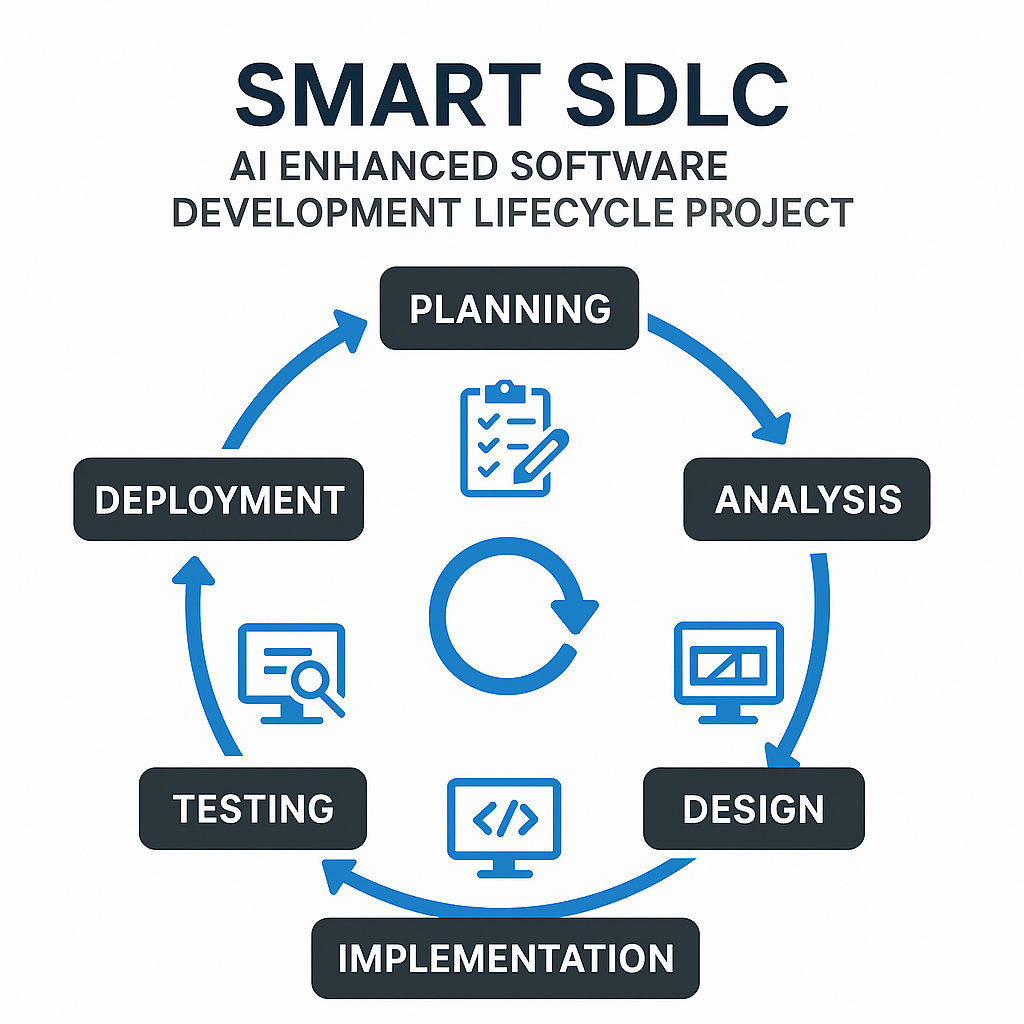
PROJECT DESCRIPTION

**1.Introduction**

* The Smart SDLC – AI Enhanced Software Development Lifecycle project leverages AI and Natural Language Processing (NLP) to automate and streamline different phases of the software development lifecycle (SDLC). Using IBM Granite’s LLM, the system can analyze requirement documents (e.g., PDFs) or user-written text, extract functional and non-functional requirements, and generate code snippets in multiple programming languages.
* This reduces manual effort in requirement analysis and code creation, helping teams accelerate software delivery, minimize errors, and improve productivity.

The system then performs:

* **Requirement Analysis** – Extracts and categorizes requirements into **functional, non-functional, and technical requirements**.
* **Code Generation** – Automatically generates code snippets in multiple programming languages (Python, Java, JavaScript, etc.) based on requirements.

****

**2.OBJECTIVES**

 **Automating Requirement Analysis**

* gathering. The code uses **PyPDF2** to read PDF files and extracts requirement text.
* Then it prompts the **IBM Granite LLM** to analyze the requirements and classify them into:
  + Functional Requirements
  + Non-Functional Requirements
  + Technical Specifications
* This reduces manual effort in requirement

**** **AI-Powered Code Generation**

* The code allows users to describe what they want (e.g., *“build a login system”*).
* The AI model then generates **code snippets** in the programming language selected from the dropdown (Python, Java, C++, etc.).
* This helps developers quickly move from requirements to working prototypes.

 **User-Friendly Interface**

* The project uses **Gradio Blocks & Tabs** to create a simple and interactive UI.
* Tabs separate functionalities:
  + **Code Analysis Tab** → Upload requirements and analyze them.
  + **Code Generation Tab** → Generate code in multiple languages.
* This makes it easy for non-technical users to interact with AI.
  + Upload a **PDF** with software requirements.
  + Or type requirements directly into a **textbox**.

 **Integration of Modern AI Models**

* The project integrates **IBM Granite LLM** through Hugging Face.
* Uses **transformers and PyTorch** to process natural language.
* The objective is to demonstrate how **LLMs can assist software engineers** in real-world SDLC tasks.

**3.ARCHITECTURE**

**1. Input Layer**

* **Requirement Input:**
  + Upload PDF file (handled by PyPDF2)
  + Or type requirements manually (via Textbox)
* **User Choices:**
  + Select programming language for code generation.

**2. Processing Layer**

* **Text Extraction Module:**
  + Extracts text from uploaded PDF using PyPDF2**.**
* **Prompt Engineering Module:**
  + Creates structured prompts for requirement analysis.
  + Creates structured prompts for code generation.

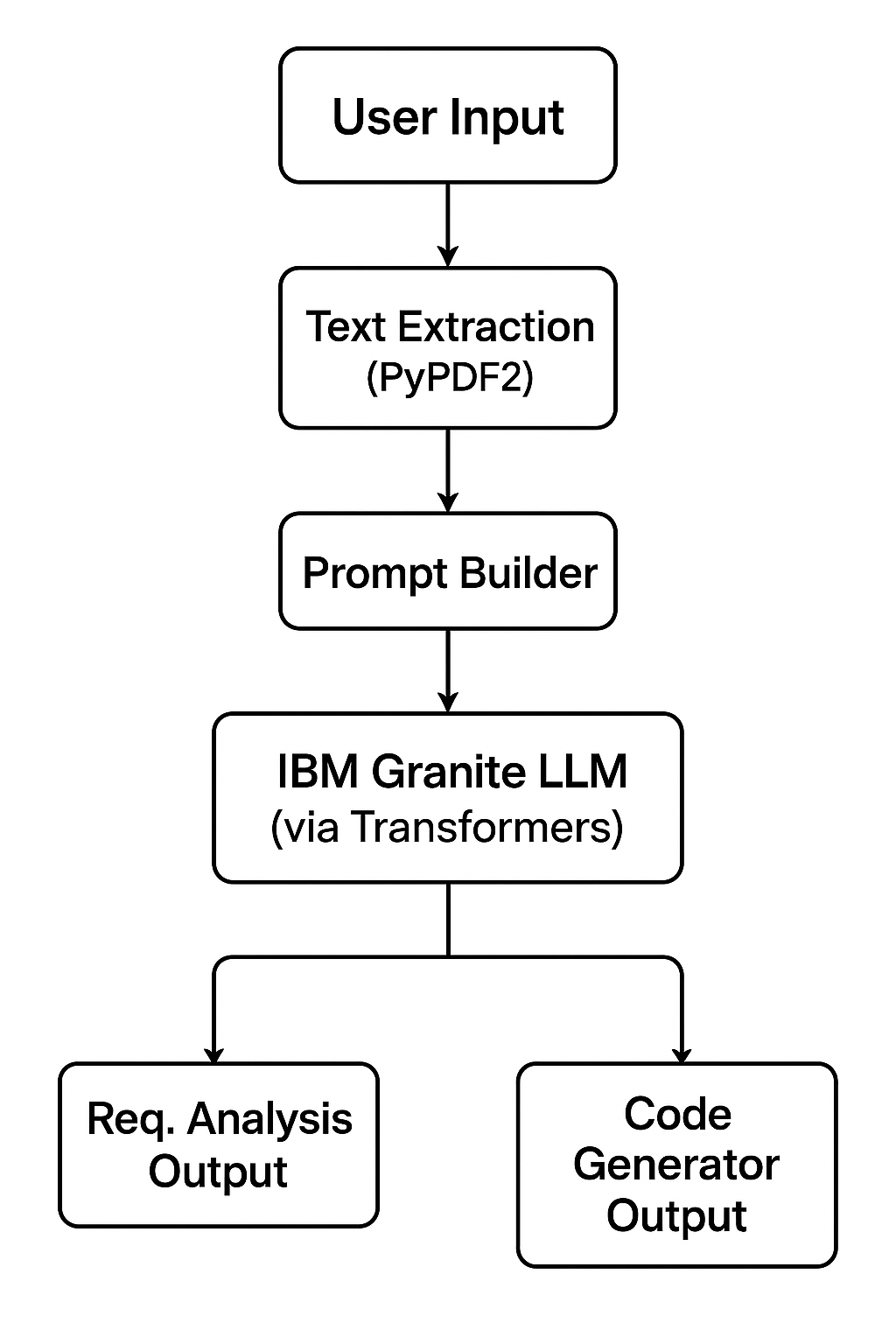
**3. AI Model Layer**

* **Tokenizer:** Converts text into tokens for the model**.**
* **LLM (IBM Granite 3.2 2B Instruct):**
  + Performs requirement classification (functional, non-functional, technical).
  + Generates code snippets based on input.
* **Torch Backend (CPU/GPU):** Handles computation.

**4. Output Layer**

* **Requirement Analysis Output:**
  + Displays categorized requirements in a textbox.
* **Code Generation Output:**
  + Displays generated code in selected programming language.
* **Gradio UI: Provides two tabs:**
  + Code Analysis Tab (Requirement Analysis)
  + Code Generation Tab (Code Creation)

The diagram shows how the system processes user input through different stages. Users upload a PDF or enter text, which is extracted and converted into prompts. The IBM Granite LLM then analyzes requirements or generates code, and the outputs are displayed in the Gradio interface.

** SYSTEM ARCHITECTURE DIAGRAM**