

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

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret 'HF_TOKEN' does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(
tokenizer_config.json: 8.88k/? [00:00<00:00, 649kB/s]
vocab.json: 777k/? [00:00<00:00, 28.5MB/s]
merges.txt: 442k/? [00:00<00:00, 26.7MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 87.3MB/s]
added_tokens.json: 100% ██████████ 87.0/87.0 [00:00<00:00, 6.62kB/s]
special_tokens_map.json: 100% ██████████ 701/701 [00:00<00:00, 61.5kB/s]
config.json: 100% ██████████ 786/786 [00:00<00:00, 83.2kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 2.94MB/s]
Fetching 2 files: 100% ██████████ 2/2 [01:19<00:00, 79.55s/it]
model-00002-of-00002.safetensors: 100% ██████████ 67.1M/67.1M [00:01<00:00, 18.7MB/s]
model-00001-of-00002.safetensors: 100% ██████████ 5.00G/5.00G [01:18<00:00, 104MB/s]
Loading checkpoint shards: 100% ██████████ 2/2 [00:17<00:00, 7.17s/it]
generation_config.json: 100% ██████████ 137/137 [00:00<00:00, 16.9kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://19308b09126c21a5e8.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces
```



No interface is running right now

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.

