**Internship Report**

**On**

**SmartSDLC: Intelligent Document-Driven SDLC Assistant**

**At Smart Bridge**

**Submitted by:**  
 **Team ID:** LTVIP2025TMID31783  
 **Team Size:** 4

**Team Leader:** Rayapudi D V K Ramya Sr

**Team Member:** P Harika  
 **Team Member:** Paleti Ram Prasad  
 **Team Member:** Pedda Giridha

**Mentor:**  
 J Prasanth Kumar

**1. INTRODUCTION**

**1.1 Project Overview**

SmartSDLC is an AI-powered platform designed to intelligently classify, analyze, and manage software requirement documents according to Software Development Life Cycle (SDLC) phases. By leveraging Large Language Models (LLMs) such as IBM Granite, and integrating it with a Python-based Gradio UI, SmartSDLC streamlines requirement analysis, mapping, and traceability across SDLC phases.

The system extracts requirements from documents (e.g., PDF) and automatically maps them to phases like Planning, Design, Implementation, Testing, and Maintenance. SmartSDLC reduces manual effort, increases traceability accuracy, and supports early error detection.

Accessible via a user-friendly Gradio interface in Google Colab, the system helps developers, analysts, and students automate SDLC phase classification, making the documentation process faster and more efficient.

**1.2 Purpose**

* To automate SDLC requirement classification from documents using NLP.
* To support accurate mapping of requirements to SDLC phases.
* To simplify traceability and document review in software engineering.
* To reduce human error and accelerate early-stage analysis.

**2. IDEATION PHASE**

**2.1 Problem Statement**

Manual SDLC documentation is time-consuming, error-prone, and often lacks standardization. Teams spend significant time analyzing requirement documents, identifying phases, and maintaining traceability matrices. There is a need for a tool that automates document analysis and phase classification to improve productivity and accuracy in software development.

**2.2 Empathy Map Canvas**

**Says:** “It takes hours to classify requirements into SDLC phases.”  
**Thinks:** “Did I miss any requirement?”  
**Does:** Spends time manually annotating documents.  
**Feels:** Tired, confused, and unsure.

**2.3 Brainstorming**

Key ideas generated:

* LLM-based SDLC phase prediction
* PDF document parsing and summarization
* Gradio UI for interactive use
* Session tracking and result export

**3. PROJECT DESIGN**

**3.1 Problem-Solution Fit**

SmartSDLC solves the manual inefficiencies of requirement phase classification by offering an automated system powered by IBM Granite. It reads documents, classifies content into SDLC phases, and presents output in a clear, exportable format.

**3.2 Proposed Solution**

* Upload requirement documents (PDF)
* Extract and classify statements into SDLC phases using LLM
* Display classified requirements through Gradio
* Allow export of results (CSV or PDF)

**3.3 Solution Architecture**

* **Frontend:** Gradio (Python UI)
* **Backend:** Python, FitZ for PDF parsing, Transformers for inference
* **AI Model:** IBM Granite (granite-3.3-2b-instruct)
* **Hosting:** Google Colab

**3.4 Project Workspace Status**

* Overall Progress: 100% Complete
* Model Integration: Complete
* UI Development: Complete

**4. PROJECT PLANNING & SCHEDULING**

|  |  |  |
| --- | --- | --- |
| **Phase** | **Week** | **Activities** |
| Research & Requirement | Week 1 | Problem study, technology exploration |
| UI & Model Setup | Week 2 | Gradio UI setup, IBM Granite integration |
| Document Parser Module | Week 3 | FitZ setup for PDF extraction |
| Inference & Classification | Week 4 | Mapping to SDLC phases using transformer model |
| Testing & Deployment | Week 5 | Functional testing, demo, final review |

**5. FUNCTIONAL & PERFORMANCE TESTING**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Area | Description | Observations | Status |
| File Upload | Upload of PDFs | Upload works as expected | ✅ Working |
| Requirement Extraction | Text extracted from PDFs | Extracts clean text | ✅ Working |
| Phase Classification | Sentences mapped to SDLC phases | Accurate and relevant | ✅ Working |
| Gradio UI | UI display and user flow | Responsive and intuitive | ✅ Working |
| Output Export | Results exported to CSV | Successfully downloadable | ✅ Working |

**6. RESULTS**

SmartSDLC successfully:

* Parses requirement documents (PDFs)
* Uses AI to classify each line into an SDLC phase
* Displays results interactively
* Supports export and future enhancements

**7. ADVANTAGES & DISADVANTAGES**

**Advantages:**

* Saves time and effort for analysts
* Reduces errors in traceability
* User-friendly interface via Gradio
* Cloud-compatible (Colab/Streamlit)

**Disadvantages:**

* Requires stable internet connection
* Model may need fine-tuning for complex domain-specific docs
* Currently supports English-only content

**8. FUTURE SCOPE**

* Add support for document formats like DOCX
* Enable batch processing of multiple files
* Visualize traceability matrices
* Add voice input/output for accessibility
* Store classified results in cloud database (Firebase, MongoDB)

**9. APPENDIX**

* Source Code: [GitHub link to be added]
* Demo Video: [Demo video link if available]
* Colab Link: [Colab notebook link]

**10. CONCLUSION**

SmartSDLC demonstrates how LLMs can enhance documentation analysis and software engineering efficiency. By reducing manual effort and improving traceability accuracy, it provides a step forward in automating parts of the SDLC process, benefiting both industry professionals and students alike.