Final Report – SmartSDLC

INTRODUCTION

1.1 Project Overview

SmartSDLC is an AI-powered platform that automates and optimizes key phases of the Software Development Lifecycle (SDLC). It integrates natural language processing (NLP) and generative AI to assist in requirement classification, code generation, bug fixing, test case creation, and code summarization, while providing SDLC-related chatbot support.

1.2 Purpose

To reduce manual effort, accelerate delivery timelines, and improve code quality by automating repetitive SDLC tasks using AI.

IDEATION PHASE

2.1 Problem Statement

Software development teams waste significant time on repetitive SDLC tasks, leading to delays and inefficiencies. SmartSDLC solves this by automating these tasks using AI.

2.2 Empathy Map Canvas

- **Thinks/Feels**: Wants to focus on core logic, frustrated by repetitive tasks
- **Says/Does**: Asks for automation tools, experiments with code gen plugins
- **Hears**: Industry pushing for AI productivity tools
- **Sees**: Growing backlog, limited AI tool adoption

2.3 Brainstorming

- Automate requirement classification
- Generate code from plain English
- Auto-fix Python bugs
- Generate pytest cases
- Summarize code
- Provide an SDLC chatbot

REQUIREMENT ANALYSIS

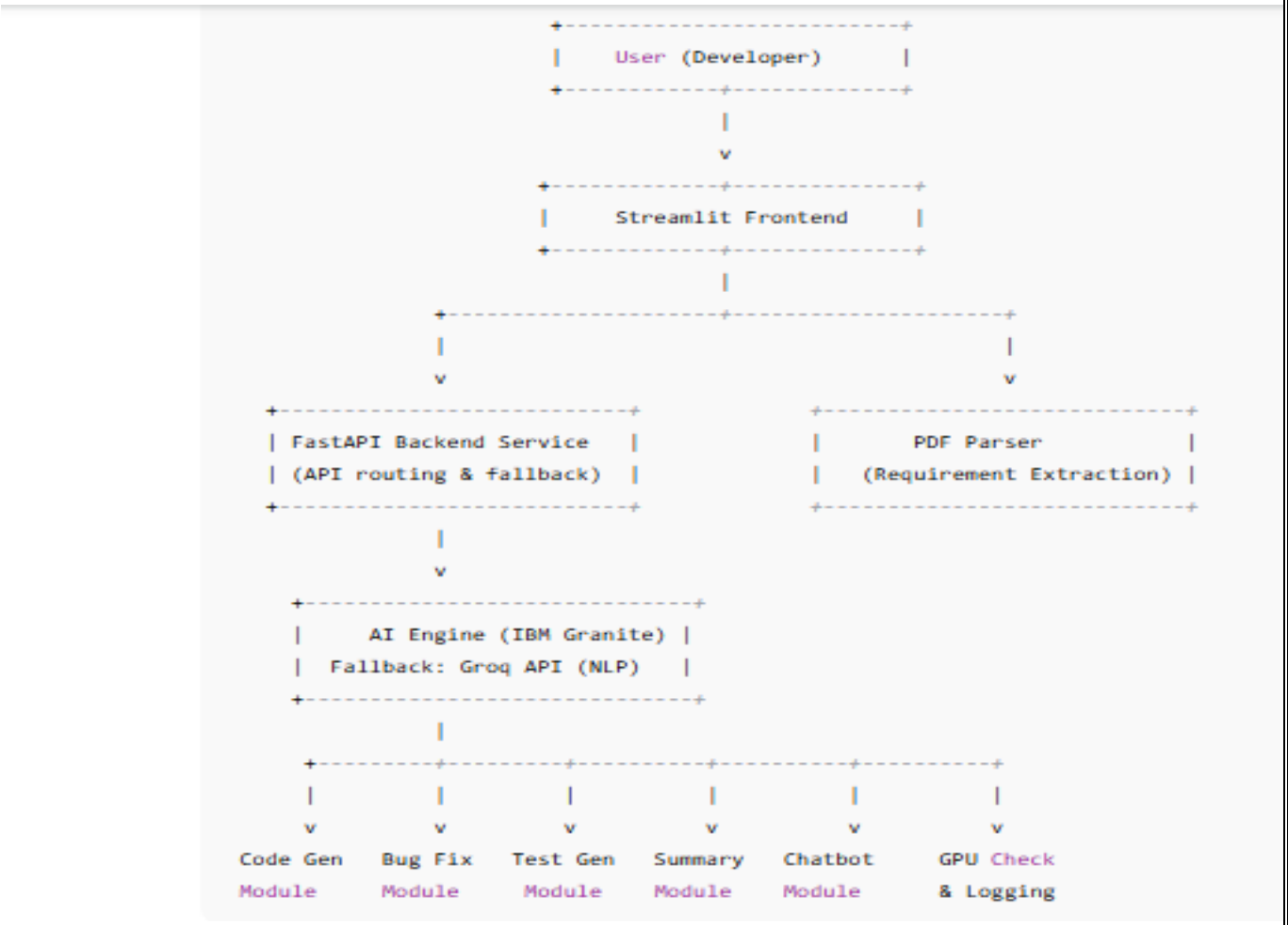
3.1 Customer Journey Map

Users discover SmartSDLC, upload requirements, generate code/tests, interact with chatbot, and gain productivity.

3.2 Solution Requirement

- Extract requirements from text/PDF
- Generate code + tests
- Fix bugs
- Summarize code intent
- Fallback API when local model fails

3.3 Data Flow Diagram



3.4 Technology Stack

- **Frontend**: Streamlit
- **Backend**: FastAPI
- **AI Models**: Transformers, IBM Granite 3.3, Hugging Face, Groq API (fallback)
- **Others**: PyMuPDF, dotenv, torch

PROJECT DESIGN

4.1 Problem Solution Fit

Meets the need for faster, automated SDLC workflows.

4.2 Proposed Solution

An end-to-end AI platform to automate SDLC tasks.

4.3 Solution Architecture

- Streamlit UI → FastAPI backend → AI model engine → External APIs fallback

PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Planned across 4 sprints: Ideation → MVP build → Testing → Enhancements

FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

- Response time under 3 seconds for code generation
- Smooth handling of concurrent API calls
- Stable large PDF uploads

ADVANTAGES & DISADVANTAGES

Advantages:

- Saves developer time
- Reduces human error
- Flexible with fallback API

Disadvantages:

- Limited to Python currently
- May need fine-tuning for complex requirements

CONCLUSION

SmartSDLC demonstrates how AI can effectively automate SDLC tasks, offering productivity gains and improved quality.

FUTURE SCOPE

- Multi-language code generation
- Customizable test scenarios
- Persistent user sessions

APPENDIX

- **Source Code**:

  Link: https://github.com/TejaSri128/SmartSDLC