**Project Report Format**

**1. INTRODUCTION**

**1.1 Project Overview**

Smart SDLC is an AI-powered platform that enhances the traditional Software Development Lifecycle (SDLC) by integrating Natural Language Processing (NLP) and Large Language Models (LLMs). It automates various phases of SDLC such as requirement analysis, code generation, bug fixing, test case creation, and documentation, thereby accelerating the development process and improving productivity.

**1.2 Purpose**

The purpose of this project is to minimize human intervention in the software development lifecycle by utilizing AI to interpret user requirements, generate and test code, provide documentation, and support developers through a chatbot assistant. This leads to faster delivery, improved software quality, and higher team efficiency.

**2. IDEATION PHASE**

**2.1 Problem Statement**

Traditional software development is time-consuming, error-prone, and heavily manual. Smart SDLC solves this by leveraging AI to automate key SDLC activities, allowing teams to build reliable software faster.

**2.2 Empathy Map Canvas**

The empathy map revealed that developers often feel overwhelmed by repetitive coding tasks, unclear requirements, and tight deadlines. They hear about AI-powered tools and wish to reduce their workload and focus more on innovation.

**2.3 Brainstorming**

We brainstormed ideas like AI requirement classification, code generation, bug fixing, and auto-documentation. Ideas were grouped and prioritized based on feasibility and impact.

**3. REQUIREMENT ANALYSIS**

**3.1 Customer Journey Map**

Users interact with Smart SDLC via Gradio UI to upload requirements, generate code, fix bugs, and retrieve documentation. They receive real-time responses and feedback.

**3.2 Solution Requirement**

Functional: Requirement upload, code generation, bug fixing, test case generation, documentation, chatbot support.  
Non-Functional: Usability, security, scalability, performance, reliability.

**3.3 Data Flow Diagram**

DFD Level 0 and 1 illustrate how requirement data flows from the user to various AI modules like code generation and testing, and is returned as downloadable output.

**3.4 Technology Stack**

* **Frontend:** Gradio (Python-based UI)
* **Backend:** Python, FastAPI
* **AI Models:** IBM Watsonx, Hugging Face Transformers
* **Database:** SQLite / MySQL
* **Cloud:** IBM Cloud, Docker, Kubernetes

**4. PROJECT DESIGN**

**4.1 Problem-Solution Fit**

We identified core problems (manual work, unclear requirements) and mapped them to AI-driven features (automated code/test generation, summarization).

**4.2 Proposed Solution**

A modular system with a user-friendly UI where requirements can be uploaded or typed. AI modules handle classification, code generation, bug fixing, and test case generation.

**4.3 Solution Architecture**

Three-tier architecture: UI (Gradio) → Application Logic (Python APIs) → AI Model Integration → Database/Storage. Deployed using Docker on IBM Cloud.

**5. PROJECT PLANNING & SCHEDULING**

**5.1 Project Planning**

4 sprints were conducted:

* Sprint 1: Requirement Upload & Analysis
* Sprint 2: Code Generation
* Sprint 3: Bug Fixing & Testing
* Sprint 4: Documentation & Chatbot

**6. FUNCTIONAL AND PERFORMANCE TESTING**

**6.1 Performance Testing**

* Code generation response time: ∼2 seconds
* Bug fixing and test case generation: ∼2.5 seconds
* File upload and classification: <3 seconds

**7. RESULTS**

**7.1 Output Screenshots**

(Included in appendix: requirement upload, generated code, test cases, chatbot interaction)

**8. ADVANTAGES & DISADVANTAGES**

**Advantages:**

* Automates repetitive tasks
* Speeds up development
* Reduces bugs and manual effort

**Disadvantages:**

* Accuracy depends on AI model quality
* Requires high compute for large datasets

**9. CONCLUSION**

Smart SDLC successfully demonstrates how AI can transform the software development lifecycle by automating key processes, reducing errors, and improving delivery speed.

**10. FUTURE SCOPE**

* Add multi-language code support
* Integrate CI/CD pipeline automation
* Advanced chatbot with voice interface
* Inbuilt project management dashboard

**11. APPENDIX**

* **Source Code:** [GitHub Link Placeholder]
* **Dataset Link:** [Dataset Placeholder]
* **GitHub & Project Demo:** [Demo Link Placeholder]