**PROJECT TITLE:**

**SmartSDLC – AI-Enhanced Software Development Lifecycle**

**TEAM NAME:**

• Rayudu Yaswanth

• Velamala Chakradhar

• Veligatla Pavan Kumar

• Adapa Veera Ajay Kumar

**PHASE-1: BRAINSTORMING & IDEATION**

**OBJECTIVE:**

• Identify the problem statement.

• Define the purpose and impact of the project.

**KEY POINTS:**

**• PROBLEM STATEMENT:**

Traditional Software Development Lifecycle (SDLC) involves repetitive manual tasks in requirement gathering, development, testing, and documentation, leading to delays, inconsistencies, and reduced productivity.

**• PROPOSED SOLUTION:**

SmartSDLC is a full-stack, AI-powered platform that automates critical SDLC phases using Natural Language Processing (NLP) and Generative AI (IBM Watsonx Granite), transforming unstructured requirements into structured, actionable development assets.

**• TARGET USERS:**

Software development teams, project managers, business analysts, and startups aiming to accelerate development with minimal manual overhead.

**• EXPECTED OUTCOME:**

An intelligent system that automates requirement classification, code generation, bug fixing, test case generation, code summarization, and conversational assistance—significantly reducing development time and effort.

**PHASE-2: REQUIREMENT ANALYSIS**

**OBJECTIVE:**

• Define technical and functional requirements.

**KEY POINTS:**

**1. TECHNICAL REQUIREMENTS:**

* **Languages:** Python
* **Libraries/Tools:** Streamlit, IBM Watsonx Granite Model, PyMuPDF, LangChain
* **Platform:** IBM Cloud

**2. FUNCTIONAL REQUIREMENTS:**

* Upload and classify requirements
* AI-based code generation
* Bug detection and fixing
* Test case generation
* Code summarization
* Floating AI chatbot assistant

**3. CONSTRAINTS & CHALLENGES:**

* Managing large language model latency
* Ensuring security and consistency in code output
* Interpreting ambiguous user prompts accurately

**PHASE-3: PROJECT DESIGN**

**OBJECTIVE:**

• Create the architecture and user flow.

**KEY POINTS:**

**1. SYSTEM ARCHITECTURE DIAGRAM:**

* **Frontend:** Streamlit UI
* **Backend:** Python APIs with AI model integration (IBM Watsonx)
* **Data Flow:** User input → AI model processing → Output visualization

**2. USER FLOW:**

* Upload requirement → Classify into SDLC phases → Generate output (code/test/summary/etc.) → Display structured results

**3. UI/UX CONSIDERATIONS:**

* Clean section-based interface
* Phase-specific tabs
* AI chatbot for contextual assistance

**PHASE-4: PROJECT PLANNING (AGILE METHODOLOGIES)**

**OBJECTIVE:**

• Break down the tasks using Agile methodologies.

**KEY POINTS:**

**1. SPRINT PLANNING:**

* **Sprint 1:** UI structure and file upload feature
* **Sprint 2:** Requirement classification and code generation
* **Sprint 3:** Testing, chatbot integration, and final polishing

**2. TASK ALLOCATION:**

**Requirement Classifier:** Rayudu Yaswanth

**AI Code Generation:** Velamala Chakradhar

**Chatbot & Test Module:** Veligatla Pavan Kumar

**Documentation & Debugging:** Adapa Veera Ajay Kumar

**3. TIMELINE & MILESTONES:**

* **Week 1:** Core features & model integration
* **Week 2:** Module completion & test case generation
* **Week 3:** Chatbot, UI finalization, and testing

**PHASE-5: PROJECT DEVELOPMENT**

**OBJECTIVE:**

• Code the project and integrate components.

**KEY POINTS:**

**1. TECHNOLOGY STACK USED:**

* Python, Streamlit, IBM Watsonx Granite, PyMuPDF, LangChain

**2. DEVELOPMENT PROCESS:**

* Requirement upload handled using PyMuPDF
* Classification via Watsonx model
* Prompt-based code/test generation
* Output rendering in Streamlit interface

**3. CHALLENGES & FIXES:**

* **Issue:** Inconsistent AI responses → Fix: Prompt tuning and output validation
* **Issue:** File parsing errors → Fix: Added preprocessing layer for PDFs

**PHASE-6: FUNCTIONAL & PERFORMANCE TESTING**

**OBJECTIVE:**

• Ensure the project works as expected.

**KEY POINTS:**

* **Functional Testing:**  
  Verified all modules (requirement parsing, code/test generation, bug fixing) produce relevant and accurate outputs.
* **Performance Testing:**  
  Checked prompt latency and interface responsiveness under multiple test conditions.
* **Bug Fixes:**  
  Resolved formatting issues in output, enhanced chatbot accuracy, ensured model timeout handling