# SmartSDLC – AI-Enchanted Software Development Lifecycle

| | |
|---|---|
| Team Leader | E. Priyadharshini |
| Team Members | S. Priyadharshini, S. Sangavi, S. Sangeetha, S. Rethika |

## 1. Introduction

Project Title: SmartSDLC – AI-Enchanted Software Development Lifecycle This project focuses on enhancing the traditional Software Development Lifecycle (SDLC) by integrating Artificial Intelligence (AI) to improve efficiency, accuracy, and automation across various phases of software engineering.

## 2. Project Overview

Purpose: The purpose of SmartSDLC is to optimize software development by automating repetitive tasks, predicting issues early, and supporting decision-making with AI-powered insights. This reduces cost, increases speed, and ensures higher software quality. Features: • AI-based Requirement Analysis • Automated Code Generation & Review • Bug Prediction & Anomaly Detection • Test Case Automation & Optimization • Project Progress Forecasting • Documentation Summarization

## 3. Architecture

Frontend (Streamlit/React): Provides an interactive dashboard for requirement inputs, progress tracking, and report visualization. Backend (FastAPI/Django): Hosts APIs for AI-driven modules such as bug prediction, code review, and test automation. AI/ML Modules: • NLP for Requirement Analysis • ML Models for Bug Prediction • Automated Code Review using LLMs • Test Case Generation & Optimization Database: Stores project data, requirements, test cases, and logs. Integration: Supports CI/CD tools like Jenkins, GitHub, and Docker for seamless development.

## 4. Setup Instructions

Prerequisites: • Python 3.9+ • pip and virtual environment tools • API keys for AI/ML modules • Access to GitHub/Jenkins for CI/CD integration Installation Process: 1. Clone the repository 2. Install dependencies from requirements.txt 3. Configure environment variables (.env file) 4. Run backend server 5. Launch frontend dashboard 6. Upload project data and start interacting

## 5. API Documentation

Available APIs: • POST /analyze-requirements – Extracts requirements from uploaded documents • POST /generate-code – Generates code snippets from specifications • POST /predict-bugs – Predicts bugs from codebase • POST /generate-tests – Creates test cases automatically • GET /project-forecast – Provides progress insights and predictions

## 6. Testing

Testing Phases: • Unit Testing – For individual AI models and utility scripts • Integration Testing – For combined modules (code review + bug prediction) • API Testing – Using Postman & Swagger • Manual Testing – User interface and overall workflow • Edge Cases – Handling incomplete requirements, ambiguous inputs

## 7. Future Enhancements

• AI-based Project Management Assistant • Enhanced Natural Language Requirement Gathering • Predictive Maintenance for Large Codebases • Automated Deployment with AI-optimized CI/CD pipelines