SmartSDLC-AI-Enhancement Software Development Lifecycle

Project Documentation

---

1. Introduction

Project Title: SmartSDLC-AI-Enhancement Software Development Lifecycle

Team Members:

DHANASEKARAN R

DHINESH C

KUMARAN C

SUTHARSAN C

The SmartSDLC project introduces AI-driven enhancements into the traditional Software Development Lifecycle (SDLC). It provides intelligent assistance in requirements gathering, design, coding, testing, deployment, and maintenance phases to reduce errors, improve productivity, and accelerate software delivery.

---

2. Project Overview

Purpose

The main goal of SmartSDLC-AI is to integrate AI-powered automation and decision-making into the software development process. This ensures:

Faster development cycles

Higher code quality

Early bug detection

Smarter resource utilization

Improved collaboration

Features

AI-powered Requirement Analysis: Extracts and validates requirements using NLP.

Automated Architecture Suggestions: Provides architecture recommendations based on project scope.

Intelligent Code Generation: AI-assisted code generation and refactoring.

Test Case Automation: AI-generated unit, integration, and regression test cases.

Bug Prediction & Fix Suggestions: Predicts common coding issues and recommends fixes.

Deployment Optimization: Smart CI/CD pipeline integration.

Performance Monitoring: AI-driven monitoring with anomaly detection.

---

3. Architecture

Frontend: React.js / Angular (for UI)

Backend: Node.js / Django / Flask

Database: PostgreSQL / MongoDB

AI/ML Layer: Python (TensorFlow / PyTorch / scikit-learn)

DevOps Tools: Docker, Kubernetes, Jenkins/GitHub Actions

Cloud Deployment: AWS / Azure / GCP

A layered architecture is followed:

1. Presentation Layer (UI/UX)

2. Application Layer (APIs, business logic)

3. AI Layer (automation, predictions, enhancements)

4. Data Layer (databases, logs, knowledge base)

---

4. Setup Instructions

1. Clone the repository:
git clone https://github.com/Dhanasekaran760/SmartSDLC.git

cd ai-enhancement

2. Install dependencies:

npm install   # for frontend
pip install -r requirements.txt # for backend + AI modules

3. Configure environment variables (.env file).

4. Initialize the database:

npm run migrate

5. Start services:

npm start   # frontend
python manage.py runserver # backend

---

5. Folder Structure

SmartSDLC/
├── frontend/   # React/Angular code
├── backend/   # API + business logic
├── ai_engine/   # AI/ML models and scripts
├── database/   # DB migrations and seed data
├── tests/   # Unit & integration tests
├── docs/   # Documentation
├── config/   # Environment & configs
├── scripts/   # Automation scripts
└── README.md

---

6. Running the Application

Development mode:

npm run dev

Production mode (Docker):

docker-compose up --build

---

7. API Documentation

All endpoints are available at:
http://localhost:5000/api/docs (Swagger / Postman collection).

Example:

POST /api/requirements/analyze → AI-based requirement extraction.

POST /api/code/generate → Auto code generation.

POST /api/test/generate → Test case generation.

---

8. Authentication

JWT-based authentication

Role-based access:

Admin: Full access

Developer: Code + test generation

Tester: Bug prediction + test validation

Viewer: Read-only access

---

9. User Interface

Dashboard: Project status, AI insights, SDLC stage tracking

Requirements Panel: Requirement analysis results

Code Panel: AI-assisted coding and refactoring suggestions

Testing Panel: Auto-generated test cases and bug predictions

Deployment Panel: CI/CD monitoring

10. Testing

Unit Tests: PyTest / Jest

Integration Tests: Postman / Newman

Automated Testing: AI-generated test cases integrated into CI/CD

Performance Testing: JMeter / Locust

Security Testing: OWASP ZAP