SmartSDLC-AI- Enhanced Software Development Lifecycle

Project Documentation

1. **Introducation:**
   - **Project Title:** SmartSDLC-AI-Enhanced Software Development Lifecycle
   - **Team Members:**
     - ❖ **GAYATHRI B**
     - ❖ **ABINAYA T**
     - ❖ **GOWTHAMI R**
     - ❖ **DHARSHINI V**

1. Project Overview

Purpose

The purpose of this project is to streamline and enhance the software development lifecycle (SDLC) using AI technologies. The system integrates a large language model (LLM), machine learning modules, and vector search to provide intelligent support across various stages of development including planning, coding, testing, and documentation.

Features

* AI-powered code generation and documentation

* Automated bug detection and suggestion

* Natural language queries for project search

* Intelligent task and sprint planning

* Real-time collaboration insights

* Version-aware AI assistance

* Smart code review using ML models

2. Architecture

Frontend

*** Framework:** React.js / Next.js

*** Features:** Dynamic UI rendering, real-time updates, LLM-powered chat interface, and visual task boards.

*** Communication:** Interacts with backend through REST/GraphQL APIs.

Backend

*** Framework:** Node.js / Express / FastAPI

*** Services:**

* API routing

* Task management logic

* Middleware for auth & rate-limiting

* **Database:** PostgreSQL / MongoDB for structured and unstructured data

LLM Integration

* **Provider:** OpenAI / HuggingFace / Local LLM (e.g., LLaMA)

* **Functions:**

    * Natural language understanding for prompts

    * Context-aware code and documentation generation

    * Dialogue memory & user history

Vector Search

* **Engine:** Pinecone / FAISS / Weaviate

* **Use Cases:**

    * Fast document/code snippet retrieval

    * Contextual matching of user queries

    * Embedding generation via sentence-transformers

ML Module

* **Model:** Custom-trained bug prediction/classification model

* **Tasks:**

    * Static code analysis

    * Pattern recognition in bug reports

    * Suggesting refactors or test cases

3. Setup Instructions

Prerequisites

* Node.js >= 16.x / Python >= 3.8

* Docker & Docker Compose (for deployment)

* PostgreSQL or MongoDB

* API key for LLM provider (e.g., OpenAI)

* Python virtual environment manager (optional)

Installation Process

This section explains how to install and run the AI Enhanced Software Development Lifecycle system on a local machine. This process is typically performed by the development or technical team.

4. Folder Structure

The project is organized into folders to keep each part of the system well-structured and easy to manage. Each folder has a specific purpose that supports the overall functioning of the AI Enhanced Software Development Lifecycle system.

Running the Application

- In development mode, the application is run in a way that allows the development team to:
- Test new features
- Find and fix bugs
- Improve design and functionality
- Make real-time changes and see updates immediately

Development Mode

The Development phase is where the actual coding takes place, turning design specifications into functional software. In an AI-enhanced SDLC, this phase is significantly optimized using AI-powered tools and techniques.

Production Mode

Once the application has been fully developed and tested, it is launched in Production Mode. This is the live version of the system that is used by actual users such as developers, project managers, or clients.

6. API Document

An API (Application Programming Interface) is a set of rules that allows different parts of the software system to communicate with each other. In this project, the API connects the frontend user interface with the backend system that processes data and runs AI features.

7. Authentication

* JWT-based token authentication

* Role-based access (Admin, Developer, Viewer)

* OAuth2 integration optional

8. User Interface

* **Dashboard:** Overview of active projects, tasks, and AI suggestions

* **Editor:** Real-time code input/output with LLM integration

* **Chat bot:** Conversational UI for AI assistance

* **Search:** Smart search bar with semantic search capability

9. Testing

* **Unit tests:** Pytest / Jest

* **Integration tests:** Postman / Supertest

* **ML model testing:** sklearn test suite

* **Code coverage:** >80%

10. **Screenshots**

11. Known Issues

* High latency on LLM responses during peak usage
* Vector search sometimes returns outdated snippets
* Limited language support (currently supports Python & JS)

12. Future Enhancements

* Multi-language support for code generation
* Self-hosted LLM models for privacy
* CI/CD integration for real-time code analysis
* AI-powered test case generation
* Plugin system for 3rd-party tools (Jira, GitHub, etc.)