Project Title

AI-Enhanced Software Development Lifecycle (AI-SDLC)

1. Introduction

Project Title: AI-Enhanced Software Development Lifecycle

Team Members:

1) S.VIDHYALAKSHMI

2)S.RANJANI

3)V.RUPIKAA

4)K.VAISHNAVI

2. Project Overview

• Purpose:

The purpose of the AI-Enhanced Software Development Lifecycle (AI-SDLC) is to revolutionize

traditional software engineering processes by embedding artificial intelligence at every phase

of the SDLC. By integrating advanced AI models for planning, coding, testing, deployment, and

monitoring, this platform aims to increase development efficiency, reduce human error, and

enable predictive software maintenance.

It empowers teams with tools like intelligent code generation, automated test case writing,

real-time defect detection, and performance monitoring powered by AI/ML. The system offers

role-specific insights, making it a strategic assistant for developers, testers, project managers,

and DevOps engineers.

• Features:

AI-Powered Requirement Analysis

Key Point: NLP-driven extraction of user needs

Functionality: Converts user stories, emails, and documents into formal requirements using

LLMs.

Smart Code Generator

Key Point: Auto-generates boilerplate or module code

Functionality: Utilizes AI models (e.g., CodeGen, Codex) to write functions from structured

prompts.

AI Test Writer

Key Point: Auto-generates test cases

Functionality: Analyzes source code and creates unit and integration test cases using AI.

Bug and Anomaly Detector

Key Point: Identifies runtime bugs and logical errors

Functionality: Integrates ML-based anomaly detection and static analysis tools.

Predictive DevOps Dashboard

Key Point: Forecasts system reliability

Functionality: Uses time-series models to predict downtimes, deployment issues, or

performance drops.

Conversational Assistant

Key Point: DevOps and code assistant

Functionality: AI chatbot for answering tech stack questions, explaining code, and suggesting

improvements.

3. System Architecture

Frontend (Streamlit):

An interactive web dashboard built with Streamlit offering multi-tab support:

Requirements Upload

Code Generation Interface

Test Generator

Bug/Anomaly Viewer

Monitoring Dashboard

Uses streamlit-option-menu for sidebar navigation and modular pages.

Backend (FastAPI):

FastAPI is used for serving REST endpoints:

Requirement parsing

Code and test generation

Bug detection

Model inference

All endpoints are async for better performance. Swagger UI is enabled for documentation.

LLM Integration (OpenAI/IBM Watsonx):

Uses Codex, IBM Watsonx, or open-source Code LLMs to handle code and test generation,

documentation summarization, and explanations.

AI Models:

NER Models: For extracting requirement entities

CodeGen/Codex: For generating code snippets

TestBERT: For writing test cases

Isolation Forest / LSTM: For bug and anomaly detection in logs

ARIMA / Prophet: For predictive DevOps metrics

4. Setup Instructions

Prerequisites:

Python 3.9+

Virtual environment tools

API keys (OpenAI/IBM Watsonx, Hugging Face)

Docker (optional)

Git

Installation Steps:

Clone the repository

Create and activate a virtual environment

Run: pip install -r requirements.txt

Create .env file with API credentials

Start backend: uvicorn app.main:app --reload

Launch frontend: streamlit run ui/dashboard.py

Interact with features via the dashboard

5. Folder Structure

ai\_sdlc/

├── app/

│ ├── api/

│ │ ├── requirements.py

│ │ ├── codegen.py

│ │ ├── testing.py

│ │ ├── bugs.py

│ │ └── monitor.py

│ ├── models/

│ │ └── ml\_models.py

│ ├── utils/

│ │ └── helpers.py

│ └── main.py

├── ui/

│ ├── dashboard.py

│ ├── pages/

│ │ ├── Requirements.py

│ │ ├── CodeGen.py

│ │ ├── Tests.py

│ │ ├── Monitor.py

│ │ └── Assistant.py

├── README.md

├── .env

└── requirements.txt

6. Running the Application

Run the FastAPI backend server

Launch the Streamlit frontend

Navigate through tabs:

Upload requirement documents

Generate code or tests

View performance forecasts or anomalies

All operations trigger real-time responses via API

7. API Documentation

Available endpoints include:

Endpoint MethodDescription

/parse-requirements POST Extracts structured requirements from plain text

/generate-code POST Returns Python code for described modules

/generate-testsPOST Returns unit tests for uploaded code

/detect-bugsPOST Analyzes code/log files for bugs or anomalies

/forecast-performance GET Predicts system health metrics

/chat POST AI assistant to answer SDLC-related queries

Swagger UI: http://localhost:8000/docs

8. Authentication

For demo purposes, the system is open. Production setups can include:

JWT-based user authentication

Role-based access (admin, developer, tester)

API key usage for third-party services

Integration with enterprise SSO (OAuth2)

9. User Interface

Key UI features:

Sidebar navigation

Tabbed layout for each SDLC phase

AI assistant for code/test Q&A

Real-time chart updates (monitoring)

Code and test viewer with syntax highlighting

Export capability for test cases and code snippets

10. Testing

Testing methodology includes:

Unit Tests: Model functions and API logic

API Tests: Swagger and Postman validations

Mock Testing: With dummy requirement sets

Error Handling: Invalid inputs, timeouts, unsupported formats

CI/CD Compatible: Includes GitHub Actions workflow for automated testing

11. Screenshots

΋ Placeholder for dashboard, chat assistant, code generator, test viewer, etc.

(Add images or embed them in the actual document/presentation.)

12. Known Issues

Occasional hallucination in code generation from vague prompts

Limited performance for very large documents (optimize chunking)

Anomaly detection requires labeled datasets for best results

Chat assistant sometimes lacks context continuity

13. Future Enhancements

Integration with GitHub/GitLab for auto-pull and commit

CI pipeline generator using AI

Enhanced user analytics and insights

AI-assisted refactoring recommendations

Auto-documentation generation from source code

Multilingual support for non-English development teams

Model fine-tuning for domain-specific SDLCs (healthcare, finance, etc.)