Project Documentation

SmartSDLC

1.Introduction

Project Title: SmartSDLC – AI Enhanced Software Development Life Cycle

Team Leader:

SELVAPRAKASH J

Members:

1.vignesh p

2.Rashidul J

3.Thuyavan K

4.Ashik S

2. Project Overview Purpose:

The purpose of SmartSDLC is to improve and automate the traditional Software Development Life Cycle (SDLC) using Artificial Intelligence, automation, and visualization tools. The system integrates requirement analysis, UML diagram generation, automated code suggestions, testing, and project documentation into a single platform.

SmartSDLC reduces manual workload, accelerates project delivery, and ensures higher accuracy in each development stage. It can be used by students, developers, and organizations as a smart assistant for managing end-to-end software projects.

Features:

Requirement Analysis Module:

Extracts requirements from user inputs/documents.

Converts requirements into structured forms.

Automated UML Generator:

Creates UML diagrams such as Use Case, Class, Activity, and Sequence.

AI-Powered Code Generator:

Generates initial code templates from requirements and design.

Testing Module:

Produces test cases and validates outputs.

Supports anomaly detection for debugging.

Documentation Generator:

Produces structured project documentation automatically.

History & Version Tracking:

Maintains a record of project phases, diagrams, and documents.

Notification & Alerts:

Provides task deadlines, project updates, and reminders.

3.Architecture:

Frontend (Streamlit/Gradio):

Interactive interface for entering requirements, generating diagrams, testing, and downloading reports.

Backend (FastAPI/Python):

Provides APIs for requirement processing, UML generation, code creation, and test execution.

AI/ML Integration:

NLP models for requirement analysis.

ML for anomaly detection and testing assistance.

Database (SQLite/MySQL/PostgreSQL):

Stores requirement data, UML diagrams, testing results, and history logs.

Version Control (GitHub Integration):

Tracks progress, commits, and documentation changes.

4.Setup Instructions:

Prerequisites:

Python 3.9+

Virtual environment (venv)

Libraries: streamlit, gradio, matplotlib, scikit-learn, transformers

Database server (SQLite/MySQL/PostgreSQL)

GitHub account for version tracking Installation

Steps:

1. Clone the SmartSDLC repository.

2. Install dependencies with pip install -r requirements.txt.

3. Configure .env file for database/API keys if needed.

4. Start backend API server (uvicorn main:app --reload).

5. Run frontend dashboard (streamlit run app.py).

5.Folder Structure:

app/ – Backend code and APIs. ui/ – Frontend

interface files. uml_generator.py – Generates

UML diagrams.

requirement_analyzer.py – Requirement extraction logic.

code_generator.py – Produces code templates.

test_module.py – Automated testing and anomaly detection.

doc_generator.py – Documentation creator.

6.Running the Application:

Launch backend server.

Run frontend UI.

Input project requirements or upload documents.

Generate UML diagrams, code, and test cases.

Download reports and track project history.

7.API Documentation:

POST /analyze-requirements → Extracts requirements.

POST /generate-uml → Produces UML diagrams.

POST /generate-code → Creates code templates.

POST /run-tests → Runs test cases.

GET /get-history → Retrieves saved history.

8.Authentication:

Supports token-based authentication (JWT).

Role-based access: Admin, Developer, Tester.

Future: OAuth2 integration and user session tracking.

9.User Interface:

Sidebar navigation with modules.

Tabs for Requirement Analysis, UML, Code, Testing, Documentation.

Real-time updates and notifications.

Export options for PDF/Doc reports.

10.Testing:

Unit Testing: Validates requirement analyzer, UML generator, and code generator.

API Testing: Verified using Swagger/Postman.

Manual Testing: Performed on frontend interface.

Edge Cases: Handles incomplete/incorrect input gracefully.

11.Known Issues:

UML diagrams for very large projects may need manual adjustment.

Limited support for highly specific programming frameworks.

12.Future Enhancements:

AI-assisted deployment automation.

Integration with Jira/Trello for project task management.

Mobile-friendly interface.

Support for multi-language documentation.

13.Conclusion:

SmartSDLC offers an AI-powered solution for automating and optimizing the Software Development Life Cycle. By integrating requirement analysis, UML generation, code creation, testing, and documentation, it reduces errors and saves significant development time.