



SMART AI ENHANCED SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

19.09.2025

1.INTRODUCTION



Project title : Smart AI Enhanced Software Development Life Cycle

Team Leader : AAKASH B

Team member : AKASHKUMAR S

Team member : ARI PRASATH J

Team member : KESAVAN G

Team member : LOKESH N

2. PROJECT OVERVIEW

Purpose:

The purpose of the Smart AI Enhanced SDLC project is to integrate Artificial Intelligence into every phase of the software development lifecycle to improve efficiency, accuracy, and adaptability. By leveraging AI-driven insights, predictive analytics, and automation, this system helps software teams deliver higher-quality products in less time.

AI assists in requirement analysis, design optimization, automated code generation, intelligent testing, defect prediction, and project management. The system also provides real-time recommendations, anomaly detection, and documentation support, helping developers, testers, and managers streamline their workflow.

Ultimately, this AI-powered SDLC enhances collaboration, reduces human error, accelerates delivery, and ensures that the final product aligns with customer expectations.



Features:

AI-Powered Requirement Analysis

Key Point: Automated requirement validation

Functionality: Analyzes business requirements, identifies ambiguities, and suggests improvements for clarity and completeness.

Intelligent Design Assistance

Key Point: Smart architecture recommendations

Functionality: Suggests optimal design patterns and system architectures based on project requirements and best practices.

Automated Code Generation

Key Point: Faster development cycles

Functionality: Generates clean, modular, and reusable code snippets in multiple languages from design specifications.



AI Testing & Bug Prediction

Key Point: Early defect detection

Functionality: Uses machine learning models to predict potential defects and automates test case generation for maximum coverage.

Project Risk Forecasting

Key Point: Risk management

Functionality: Predicts risks such as cost overruns, delays, and resource bottlenecks using historical project data.

Anomaly Detection

Key Point: Early warning system

Functionality: Identifies unusual project patterns in timelines, code changes, or team productivity to prevent issues before escalation.

Policy & Documentation Summarization



Key Point: Simplified project documentation

Functionality: Converts lengthy project documents into concise, AI-generated summaries for quick understanding.

AI-Driven Feedback Loop

Key Point: Continuous improvement

Functionality: Collects developer and tester feedback, analyzes it, and suggests workflow optimizations.

3. ARCHITECTURE


Frontend (Streamlit / Gradio):

Provides an interactive dashboard where project managers, developers, and testers can visualize AI insights, review forecasts, and interact with the system through natural language queries.

Backend (FastAPI):

Handles API requests for AI-driven analysis, forecasting, bug prediction, requirement analysis, and document summarization. Optimized for asynchronous performance and scalability.

LLM Integration (e.g., IBM Watsonx / OpenAI):



Large language models are used for requirement analysis, code generation, documentation summarization, and intelligent chat-based assistance.

Vector Database (Pinecone / FAISS):

Stores project documents, historical data, and embeddings for semantic search, enabling users to query project knowledge bases in natural language.

ML Modules (Forecasting & Defect Prediction):

Implements predictive models for risk management, anomaly detection, and defect forecasting using Scikit-learn, TensorFlow, and Pandas.

4. SETUP INSTRUCTIONS

Prerequisites:


Python 3.9 or later

pip and virtual environment tools

API keys for AI services (Watsonx / OpenAI / Pinecone)

Internet access for cloud-based processing

Installation Process:

- 
1. Clone the repository
 2. Install dependencies from requirements.txt
 3. Configure .env file with API keys and credentials
 4. Run backend server with FastAPI
 5. Launch frontend dashboard with Streamlit/Gradio
 6. Upload project documents and interact with AI modules

5. Folder Structure

app/ – Contains backend logic (APIs, models, services)
app/api/ – Modular API routes for requirements, testing, risk forecasting
ui/ – Frontend components (Streamlit/Gradio dashboards)
ai_llm.py – LLM integration for requirements & documentation
defect_predictor.py – Predicts potential software defects
risk_forecaster.py – Forecasts risks and project delays



doc_summarizer.py – Summarizes project documents

report_generator.py – Generates AI-driven project reports

6. RUNNING THE APPLICATION

1. Start FastAPI server to enable backend APIs
2. Launch Streamlit dashboard for web interface
3. Navigate using sidebar menus (requirements, testing, forecasting, reports)
4. Upload project documents or queries
5. Receive AI-generated insights, summaries, and risk predictions

7. API DOCUMENTATION

POST /requirements/analyze – Validates and analyzes project requirements

POST /design/suggest – Suggests architectures/design patterns



POST /code/generate – Generates code from design inputs

POST /testcases/auto-generate – Creates test cases automatically

GET /forecast/risk – Returns project risk analysis

POST /feedback/submit – Stores developer/tester feedback for analysis

8. AUTHENTICATION

Swagger UI documentation for API testing

Authentication keys for secure endpoints


Role-based access control (developer, tester, manager)

9. USER INTERFACE

The interface is minimalist and functional, focusing on accessibility for non-technical users, such as city planners, sustainability coordinators, or concerned residents. It includes:

Sidebar with navigation: This allows users to easily switch between key functions like Energy Management, Waste Reduction, and Transportation.

KPI visualizations with summary cards: The dashboard presents key performance indicators (KPIs) in clear, actionable formats. For example, a card could show 'Current City Energy Consumption' in MWh or 'Daily Waste Diversion Rate' as a percentage, helping users quickly understand the city's sustainability progress.



Tabbed layouts for chat, eco tips, and forecasting: This design organizes the core features of the assistant. Users can interact with the AI chatbot for specific queries, explore proactive eco tips tailored to their city's data, and view forecasting models for future sustainability trends.

Real-time form handling: This ensures a smooth user experience when inputting data, such as reporting illegal dumping or submitting a proposal for a new green space.

PDF report download capability: Users can generate and download comprehensive reports on various metrics, like quarterly carbon emissions or waste management data, for presentations or further analysis.

10.TESTING

Testing was done in multiple phases to ensure the reliability and accuracy of the Sustainable Smart City Assistant in providing critical data and recommendations.

Unit Testing: Individual functions for prompt engineering were tested to ensure the AI chatbot provided relevant and accurate responses to sustainability-related queries. Utility scripts for data parsing and cleaning were also validated to guarantee data integrity.

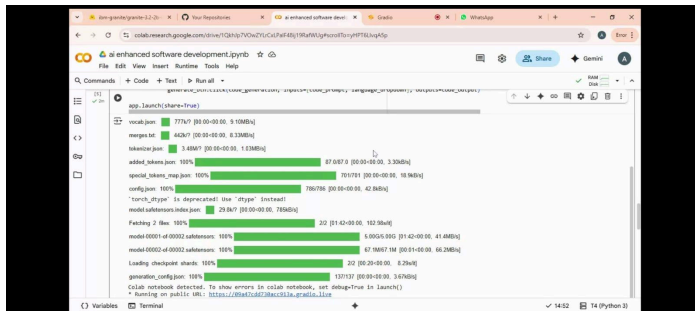
API Testing: Via Swagger UI, Postman, and custom test scripts, the system's integration with third-party APIs (e.g., for real-time traffic data or weather information) was rigorously tested to ensure seamless data exchange.

Manual Testing: The entire user journey was tested manually, including file uploads of city data, the consistency and accuracy of chat responses, and the visual presentation of forecasting outputs to ensure a reliable and cohesive user experience.

Edge Case Handling: The system was tested against a range of challenging scenarios, such as malformed input files (e.g., incorrectly formatted spreadsheets),

large data files that could strain the system, and invalid API keys, to ensure it handled errors gracefully without crashing.

11. SCREENSHOTS



12. KNOWN ISSUES

Limited accuracy for highly domain-specific requirements


Code generation may need manual refinement

Forecasting accuracy depends on availability of historical data

13. FUTURE ENHANCEMENTS

Integration with Jira/GitHub for live project tracking

AI-driven automated deployment pipeline (DevOps integration)



Support for multi-language code generation (Java, Python, C++)

Real-time collaborative assistant for agile teams