SmartSDLC – AI-Enhanced Software Development Lifecycle

Category: Generative Al with IBM

Prepared by :Huwaida

Irfana

Anisha

Asmitha banu

Skills Required:
18M Cloud, BM Watson

Project Description:

‘The SmartSDLC projectis an Al-enhanced platform designed to automate and streamline the Software Development Lifecycle (SDLC) using advanced technologies like IBM Watson, FastAPl, LangChain, and Streamilt. It integrates generative Al to handle key SDLC phases, including requirement analysis, code generation, test case creation, bug fixing, and documentation. The platform features 2 user-friendly interface that allows Users to upload PDFs, generate structured requirements, and ‘transform natural language prompts into functional code. It also includes an ‘Al-powered chatbot for real-time assistance and support. The backend, built with FastAPl, efficiently processes API requests, while the frontend, developed using Streamit, offers a visually appealing and interactive dashboard. The system is ‘modular, scalable, and secure, featuring a robust authentication mechanism and seamless integration between Al ‘models and user inputs. Deployment is streamlined with local hosting via Uvicorn and Streamlit, with comprehensive API documentation available through Swagger UI. By leveraging Al ina practical and efficient manner, SmartSDLC. significantly reduces manual workioad, accelerates development processes, and enhances software quality, making it an invaluable tool for developers and project teams.

‘Scenario 1: Requirement Upload and Classification

Requirement Upload and Classification, the platform simplifies the complex task of requirement gathering by allowing users to upload POF documents containing raw, unstructured text. The backend extracts content using PyMuPDF and leverages IBM Watsonx’s Granite-208 ‘Al model to classify each sentence into specific SDLC phases such as Requirements, Design, Development, Testing, or Deployment. These classified inputs are then transformed into structured user stories, enabling clear planning and traceability. The frontend displays this output in an ‘organized, readable format grouped by phase, significantly improving clarity and saving manual effort.

‘Scenario 2: Al Code Generator

Al Code Generator, addresses the development phase, where developers an input natural language prompts or structured user stories. These prompts ‘are sent to the Watsonx model, which generates contextually relevant, production-ready code. This reduces the time needed for boilerplate or prototype creation and enhances coding efficiency. The code is. presented in clean, syntax-highlighted format on the frontend, ready for use or further enhancement.

Scenario 3: Bug Fixer

Bug Fixer, the platform supports debugging by accepting code snippets in languages such as Python or JavaScript. Upon receiving the buggy code, the Watsonx Al analyzes it for both syntactical and logical errors and. returns an optimized version. This not only assists developers in identifying mistakes without extensive manual reviews but also provides immediate, corrected code directly in the frontend for comparison.

ChatGPT

You said:

Smart SDLC software development life cycle for example abstract

ChatGPT said:

Here’s an example Abstract for a Smart SDLC (Software Development Life Cycle) document:

Abstract

The Smart Software Development Life Cycle (Smart SDLC) is a modernized framework that enhances traditional SDLC models by integrating automation, agility, and intelligent decision-making tools. Unlike conventional approaches, Smart SDLC leverages advanced technologies such as Artificial Intelligence (AI), DevOps practices, and continuous integration/continuous deployment (CI/CD) pipelines to improve efficiency, quality, and adaptability in software projects.

This approach emphasizes smart project planning, automated testing, real-time monitoring, and predictive analysis to minimize risks and reduce development costs. By combining Agile, DevOps, and AI-driven practices, Smart SDLC ensures faster delivery, improved collaboration among stakeholders, and higher software reliability. It is especially suited for dynamic environments where requirements evolve rapidly, making it a practical methodology for modern software engineering projects.

Abstract

The Smart Software Development Life Cycle (Smart SDLC) is an advanced methodology that improves upon traditional SDLC models by incorporating agility, automation, and intelligent technologies. It integrates Artificial Intelligence (AI), DevOps, and Continuous Integration/Continuous Deployment (CI/CD) to optimize efficiency, minimize errors, and reduce costs. This smart approach ensures faster delivery, better adaptability to changing requirements, and high-quality outcomes, making it suitable for modern software projects.

Introduction

Software development has evolved significantly over the years, requiring methodologies that can keep pace with rapid technological advancements and dynamic business needs. Traditional SDLC models such as Waterfall and Spiral often face limitations in flexibility and speed. To overcome these challenges, Smart SDLC introduces innovation by merging Agile principles, DevOps culture, and intelligent tools. It emphasizes automation in testing, deployment, and monitoring, while enabling collaboration among stakeholders. As a result, Smart SDLC provides a structured yet adaptive framework for delivering reliable software solutions.

Objectives

The main objectives of Smart SDLC are:

To enhance the efficiency and reliability of software development.

To integrate automation for testing, deployment, and maintenance.

To support continuous delivery and quick adaptation to changes.

To minimize risks and errors through predictive analytics and monitoring.

To improve collaboration among developers, testers, and stakeholders.

To ensure high-quality software with reduced cost and time.

Phases of Smart SDLC

Requirement Analysis

Collecting requirements through smart tools (e.g., AI-based requirement gathering, user feedback analysis).

Prioritizing requirements with stakeholder collaboration.

System Design

Designing architecture using modeling tools.

Ensuring scalability, security, and adaptability in design.

Smart Development

Coding with integrated development environments (IDEs) enhanced by AI.

Automated code review and error detection.

Version control and collaborative development using platforms like GitHub/GitLab.

Testing (Automated & Continuous)

Automated test scripts for faster validation.

Continuous testing integrated with CI/CD pipelines.

Smart tools for detecting vulnerabilities and performance issues.

Deployment

Automated deployment using DevOps practices.

Continuous Integration/Continuous Deployment (CI/CD) for faster delivery.

Cloud-based deployment for scalability.

Monitoring & Maintenance

Real-time monitoring with AI-driven analytics.

Predictive maintenance to prevent failures.

Regular updates based on feedback and changing requirements.

Conclusion

The Smart SDLC is a powerful evolution of the traditional software development lifecycle, addressing the challenges of flexibility, speed, and quality in today’s dynamic environment. By combining automation, AI, and DevOps practices, it ensures efficient development, reliable performance, and customer satisfaction. Organizations adopting Smart SDLC can expect reduced costs, faster time-to-market, and improved collaboration, making it an essential methodology for future-ready software engineering.

Output

The implementation of Smart SDLC results in:

Faster software development and delivery cycles.

High-quality, secure, and reliable applications.

Reduced development cost and time.

Better collaboration among teams and stakeholders.

Improved adaptability to evolving customer and market needs.

Predictive insights for proactive decision-making.