

Project Report

1. INTRODUCTION

1.1 Project Overview

The SmartSDLC project introduces an AI-Enhanced Software Development Lifecycle platform designed to streamline and optimize various phases of software development using generative Artificial Intelligence. This platform aims to address common pain points faced by development teams, including ambiguous requirements, inefficient code generation and debugging, laborious testing, and difficulties in understanding complex codebases. By integrating several AI-powered tools into a single, cohesive application, SmartSDLC seeks to improve efficiency, code quality, and overall team productivity.

1.2 Purpose

The purpose of this project is to demonstrate the practical application of generative AI to enhance the traditional Software Development Lifecycle. SmartSDLC provides intelligent assistance that automates repetitive tasks, offers on-demand expertise, and facilitates better understanding and collaboration, ultimately leading to faster delivery of higher-quality software and increased job satisfaction for development professionals.

2. IDEATION PHASE

2.1 Problem Statement

Software development teams are often hampered by challenges such as unclear requirements, time-consuming manual coding and debugging, inefficient test case creation, difficulty comprehending complex code, fragmented tooling, and a lack of immediate knowledge support. These issues contribute to project delays, increased costs, and compromised software quality. SmartSDLC directly tackles these problems as detailed in the "SmartSDLC: Customer Problem Statements" Canvas, identifying six core pain points including Requirement Ambiguity, Inefficient Code Development & Debugging, Manual & Incomplete Testing, Difficulty Understanding Complex/Legacy Code, Lack of Immediate SDLC Knowledge/Support, and Fragmented Tooling & Inefficient Workflow.

2.2 Empathy Map Canvas

An Empathy Map was utilized to gain a deep understanding of our target users, primarily the Software Developer, but also Project Managers, QA Engineers, and Team Leads. The "SmartSDLC: Empathy Map Canvas (Software Developer)" details their "Think & Feel," "Hear," "See," and "Say & Do" aspects, alongside their "Pains" (e.g., frustrating debugging, time-consuming manual tasks, lack of quick knowledge) and "Gains" (e.g., reduced debugging time, accelerated code generation, on-demand AI assistance) that SmartSDLC aims to deliver. This empathy-driven approach ensured the solution directly addresses real-world user challenges.

2.3 Brainstorming

The brainstorming process for SmartSDLC focused on generating a wide array of AI-powered solutions to the identified problem statements. Ideas were initially generated freely and then grouped into logical categories such as "Code Generation & Optimization," "Debugging & Error Resolution," "Code Comprehension & Knowledge Transfer," and "Testing Automation." These grouped ideas directly informed the core features of SmartSDLC. Prioritization was then performed on an Importance vs. Feasibility matrix to select the most

impactful and viable features for initial development, as documented in the "SmartSDLC: Brainstorming & Idea Prioritization Template (Adapted)" Canvas.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey map

3.2 Solution Requirement

The functional requirements of SmartSDLC include core capabilities such as:

- **Requirement Classification:** PDF upload, SDLC phase classification, text extraction.
- **AI Code Generation:** Natural language prompt input, Python code output.
- **Automated Bug Fixing:** Buggy code input, fixed code output.
- **Smart Test Case Generation:** Code/requirement input, test case generation.
- **Code Summarization:** Code snippet input, human-readable summary.
- **Interactive AI Chatbot Assistant:** Chatbot UI, user message input, AI response.

3.3 Data Flow Diagram

The SmartSDLC system's data flow is conceptually represented by a DFD Level 0, as outlined in the "SmartSDLC: Data Flow Diagrams & User Stories" Canvas. At a high level, a "User" provides "Text/PDF Input" to the "SmartSDLC System." The system processes this input, interacts with an external "AI Model (e.g., ibm-granite)" for inference, and then returns "AI Responses/Code/Summary" or "Classification Results" back to the user. This diagram illustrates the high-level information flows and interactions between the user, the system, and the core AI component.

3.4 Technology Stack

SmartSDLC is built on a robust and scalable technology stack:

- **Frontend:** HTML, CSS (Tailwind CSS), JavaScript for a responsive web application.
- **Backend:** Python with FastAPI for high-performance API services.
- **AI Models:** The core intelligence is provided by the ibm-granite/granite-3.3-2b-instruct Large Language Model, integrated using the Hugging Face Transformers library.
- **PDF Processing:** PyMuPDF is used for efficient text extraction from PDF documents.
- **Deployment:** Developed and demonstrated using Google Colab, uvicorn, for local development and public access, with future plans for cloud deployment on Kubernetes/Docker for scalability.

4. PROJECT DESIGN

4.1 Problem Solution Fit

The "SmartSDLC: Problem – Solution Fit Template" Canvas comprehensively maps each identified customer problem (e.g., Requirement Ambiguity, Inefficient Code Development) to a specific SmartSDLC solution (e.g., Intelligent Requirement Classification, AI Code

Generator). This mapping clearly demonstrates how each feature of SmartSDLC directly addresses and mitigates a particular pain point for software development professionals, ensuring the project delivers genuine value and solves real-world problems.

4.2 Proposed Solution

The proposed solution, SmartSDLC, is an AI-enhanced platform that centralizes key SDLC functionalities. It aims to automate repetitive tasks, provide intelligent assistance, and improve overall workflow efficiency. Its integrated suite of tools includes Requirement Classification, AI Code Generation, Automated Bug Fixing, Smart Test Case Generation, Code Summarizer, and an Interactive AI Chatbot Assistant.

4.3 Solution Architecture

The solution architecture of SmartSDLC follows a client-server model. The web-based frontend interacts with a Python FastAPI backend. The backend orchestrates calls to the [ibm-granite](#) AI model (hosted externally via Hugging Face) and utilizes PyMuPDF for PDF processing. This layered architecture ensures a clear separation of concerns, facilitating independent development and scaling of components. The conceptual DFD Level 0 and detailed component descriptions in the "SmartSDLC: Solution Architecture Diagram" and "SmartSDLC: Technology Stack (Architecture & Stack)" Canvas illustrate these interactions and the flow of data.

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

The SmartSDLC project followed an agile methodology, leveraging sprints for iterative development. Key agile concepts such as Epics, User Stories, and Story Points were utilized for planning and estimation. The project was broken down into manageable 5-day sprints, with a target velocity of 8 story points per sprint. A detailed Product Backlog outlining functional requirements and their associated user stories, along with a Sprint Schedule and estimation, is provided in the "SmartSDLC: Project Planning Template" Canvas. The "SmartSDLC: Planning Logic (Adapted)" Canvas further details the agile concepts and provides an example of velocity calculation based on the project's sprint data.

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

Functional and performance testing for SmartSDLC focused on validating the correctness and efficiency of each AI-powered feature. Functional tests covered areas like PDF upload and requirement classification, accurate code generation and bug fixing, relevant test case generation, precise code summarization, and responsive chatbot interactions. Performance tests aimed to ensure acceptable response times for AI model inferences and smooth handling of file uploads under various conditions, as outlined in the "SmartSDLC: Functional &

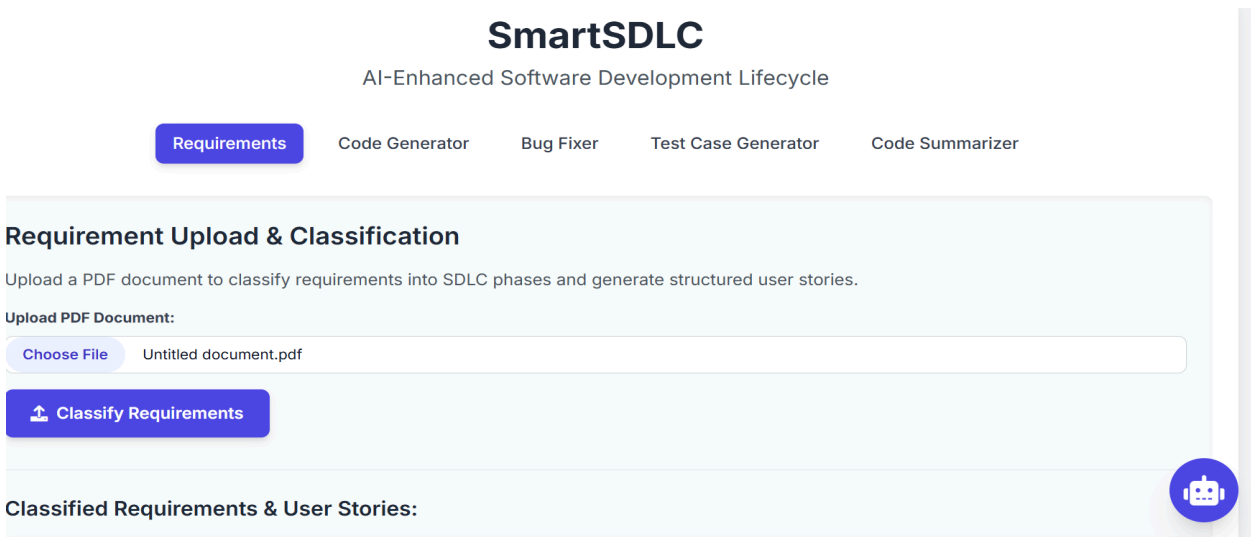
Performance Testing Template" Canvas. Key performance indicators included AI response times (aiming for under 15-20 seconds) and API stability under moderate load.

7. RESULTS

7.1 Output Screenshots

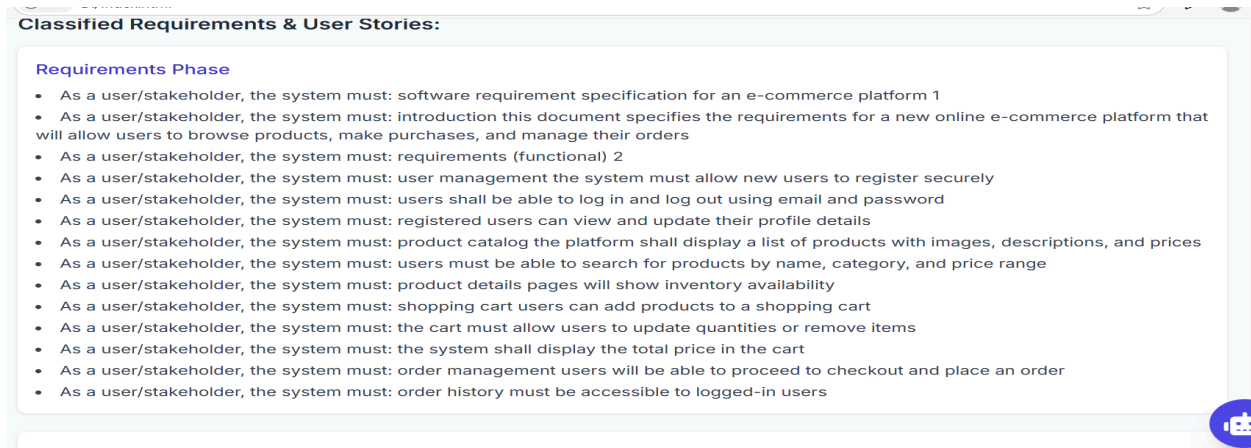
1. Intelligent Requirement Classification:

Input:



Description: This feature allows users to upload unstructured text documents (primarily PDFs) containing project requirements. The system then processes this text, identifies key requirement sentences, and classifies them into standard SDLC phases (e.g., Requirements, Design, Development, Testing, Deployment, Other). It can also generate structured user stories from the extracted information.

Output:



2. AI Code Generator:

Input:

SmartSDLC

AI-Enhanced Software Development Lifecycle

RequirementsCode GeneratorBug FixerTest Case GeneratorCode Summarizer

AI Code Generator

Input natural language prompts or structured user stories to generate production-ready code.

Code Prompt / User Story:

Python program to find the factorial of a number

Generate Code

Description: This powerful tool enables developers to generate functional code snippets or full functions by simply providing a natural language prompt or a user story. It significantly accelerates the coding process for boilerplate, common algorithms, or initial feature implementations.

Output:

Generate Code

Generated Code:

```
def factorial(n):  
    if n < 0:  
        return "Factorial is not defined for negative numbers"  
    elif n == 0:  
        return 1  
    else:  
        result = 1  
        for i in range(1, n + 1):  
            result *= i  
        return result
```

3. Automated Bug Fixer:

Input:

Requirements

Code Generator

Bug Fixer

Test Case Generator

Code Summarizer


Bug Fixer

Input a buggy code snippet (Python/JavaScript) to get an optimized and corrected version.

Buggy Code Snippet:

```
def sum(a,b):  
    return a+b
```

Fix Bug



Description: Designed to assist developers in quickly identifying and correcting errors. Users can paste a buggy code snippet, and the AI analyzes it to suggest and provide an optimized, corrected version, reducing manual debugging time and improving code quality.

Output:

Fixed Code:

Original Code:

```
def sum(a,b):  
    return a+b
```

Corrected Code:

```
python  
def add(a, b):  
    return a + b
```

4. Smart Test Case Generator:

Input:

AI-Enhanced Software Development Lifecycle

RequirementsCode GeneratorBug FixerTest Case GeneratorCode Summarizer


Test Case Generator

Provide functional code or a requirement to generate suitable test cases.

Code / Requirement for Test Cases:

Python function to find whether a number is prime or not.

Generate Test Cases



Description: This feature automates the creation of test cases. Developers or QA engineers can provide either functional code or a software requirement, and the AI will generate relevant unit or integration test cases, often adhering to specific testing frameworks like unittest or pytest.


Output:

```
class TestIsPrime(unittest.TestCase):

    def test_is_prime_positive_integers(self):
        self.assertTrue(is_prime(2))
        self.assertTrue(is_prime(3))
        self.assertTrue(is_prime(5))
        self.assertTrue(is_prime(7))
        self.assertTrue(is_prime(11))
        self.assertFalse(is_prime(4))
        self.assertFalse(is_prime(6))
        self.assertFalse(is_prime(8))
        self.assertFalse(is_prime(9))
        self.assertFalse(is_prime(10))

    def test_is_prime_zero_and_negative(self):
        self.assertFalse(is_prime(0))
        self.assertFalse(is_prime(-1))
        self.assertFalse(is_prime(-2))

if __name__ == '__main__':
    unittest.main()
```



5. Code Summarizer:

Input:

SmartSDLC

AI-Enhanced Software Development Lifecycle

RequirementsCode GeneratorBug FixerTest Case GeneratorCode Summarizer

Code Summarizer

Input any source code snippet or module to get a human-readable explanation.

Code Snippet to Summarize:

```
def fibonacci_iterative(n_terms):  
    """  
    Generates the Fibonacci series up to n_terms using an iterative approach.  
    """  
    if n_terms <= 0:  
        print("Please enter a positive integer.")  
        return  
    elif n_terms == 1:  
        print("Fibonacci Series:")  
        print(0)  
        return
```

Summarize Code

Description: To enhance code comprehension and knowledge transfer, this tool provides human-readable explanations and summaries of any given code snippet or module. It's invaluable for new team members, code reviews, or understanding complex legacy codebases.

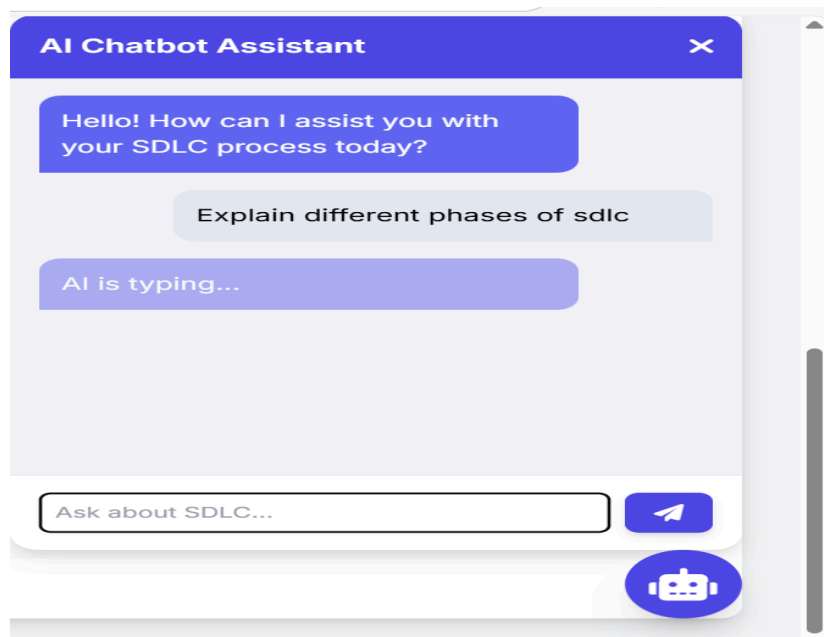
Output:

Code Summary:

The provided Python function `fibonacci_iterative(n_terms)` generates the first `n_terms` of the Fibonacci series using an iterative method. The Fibonacci series is a sequence of numbers where each number is the sum of the two preceding ones, usually starting with 0 and 1. This function checks for valid input (positive integers only), prints the initial terms manually, and then uses a loop to calculate subsequent terms until it reaches the desired count (`n_terms`). It's useful for generating the first few terms of the Fibonacci series without computing all terms upfront, which can be efficient for larger values of `n`. Explanation: 1. Function definition: - The function takes one argument, `n_terms`, which specifies the number of Fibonacci terms to generate. 2. Input validation: - It checks if `n_terms` is less than or equal to zero; if so, it prompts the user for a positive integer and returns early. - If `n_tabs` equals 1, it prints the first term directly as 0 and terminates. 3. Initialization: - Variables `a` and `b` are initialized to the first two Fibonacci numbers, 0 and 1 respectively. - The first two terms are printed explicitly before entering the loop. 4. Iteration and calculation: - A for-loop runs from 2 to `n_terms`, calculating each term by adding the previous two (`a` and `b`), storing the result in `next_term`, and updating `a` and `b` accordingly. - Inside the loop, each calculated term is printed followed by a space. 5. Output formatting: - After printing all terms, the function prints a newline character (`print()` with no arguments) to move to the next line in the output. Use Cases: - Education: Demonstrating the Fibonacci series concept and iteration. - Coding practice: Helping beginners understand loops, variable initialization, and function handling in Python. - Applications requiring Fibonacci sequence generation: Such as algorithms, simulations, or problem-solving tasks that involve Fibonacci sequences.

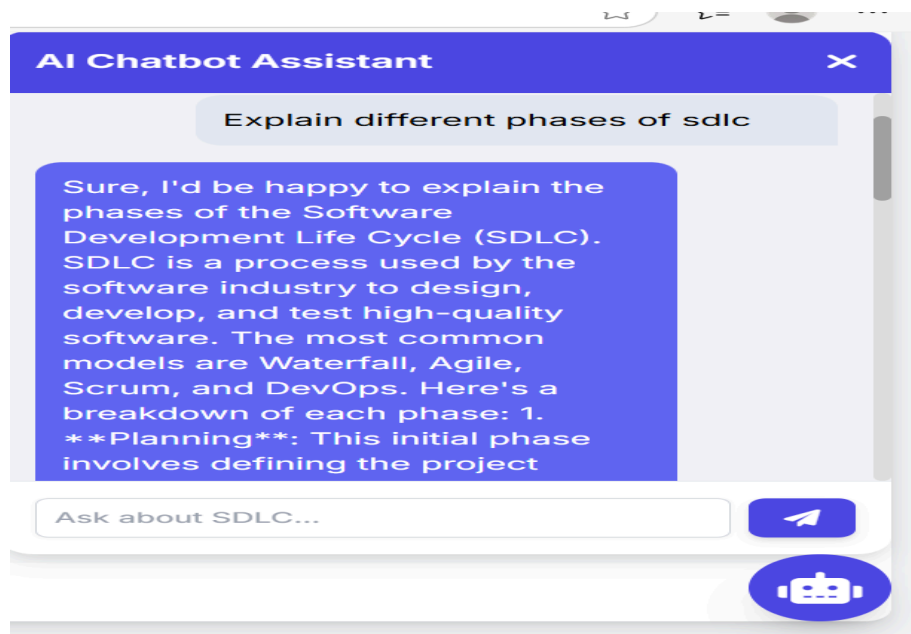
6. Interactive AI Chatbot Assistant:

Input:



Description: A conversational AI assistant available on-demand to answer questions related to the Software Development Lifecycle. It can provide information on methodologies (Agile, Scrum), best practices, specific tools, or general programming concepts, acting as an instant knowledge base.

Output:



8. ADVANTAGES & DISADVANTAGES

Advantages:

- **Increased Productivity:** Automates repetitive tasks like boilerplate code generation, test case creation, and initial debugging, freeing developers to focus on complex problem-solving.
- **Improved Code Quality:** AI-driven bug fixing and comprehensive test case generation lead to more reliable and robust software.
- **Faster Time-to-Market:** Accelerates various SDLC phases, from requirements understanding to coding and testing.
- **Enhanced Knowledge Transfer:** Code summarization and the AI chatbot democratize access to project-specific and general SDLC knowledge, aiding onboarding and continuous learning.

Disadvantages:

- **AI Dependency:** Reliance on external AI models introduces potential latency, cost, and availability concerns.
- **Model Hallucinations/Accuracy:** Generative AI can sometimes produce incorrect or non-optimal code/text, requiring human oversight and validation.
- **Data Privacy/Security Concerns:** Handling sensitive code or requirement documents for AI processing requires robust security measures (though not fully implemented in this prototype).
- **Integration Complexity:** Integrating AI into existing, often diverse, SDLC toolchains can be complex.

9. CONCLUSION:

SmartSDLC successfully demonstrates the transformative potential of integrating generative AI across the Software Development Lifecycle. By intelligently automating and assisting in key areas such as requirement classification, code generation, bug fixing, test case generation, code summarization, and providing an interactive chatbot, the platform significantly enhances developer productivity, improves software quality, and streamlines development workflows. This project serves as a compelling proof-of-concept for the next generation of intelligent software development tools, built with a robust FastAPI backend and a responsive HTML/CSS/JS frontend, powered by the ibm-granite AI model.

10. FUTURE SCOPE

The future scope for SmartSDLC is extensive and includes:

- **Advanced AI Capabilities:** Integrating more powerful LLMs, incorporating multimodal inputs (e.g., design mockups), and enabling automated code refactoring.

- **Broader Language Support:** Expanding code generation and analysis beyond Python to other popular programming languages.
- **Deeper Tool Integration:** Seamless integration with popular IDEs (VS Code, IntelliJ), CI/CD pipelines (Jenkins, GitLab CI), and project management tools (Jira, Azure DevOps).
- **User Management & Collaboration:** Implementing robust authentication, authorization, and team collaboration features.
- **Custom Model Fine-tuning:** Allowing organizations to fine-tune AI models on their proprietary codebases for more tailored results.
- **Enhanced Security:** Implementing enterprise-grade security, data encryption, and access controls.
- **Comprehensive Reporting & Analytics:** Providing dashboards for tracking SDLC metrics and AI tool usage.
- **Self-Hosting Options:** Offering solutions for on-premise or private cloud deployment for enhanced data control.

11. APPENDIX

- **GitHub Repository Link:**
<https://github.com/Madhuri-Karnam/SmartSDLC---AI-Enhanced-Software-Development-Lifecycle>
- **Project Demo Link:**
<https://drive.google.com/file/d/1xgeYCrtmhoBJwuw1YLYb5UlvFUFgm0yE/view?usp=sharing>