Likh diye theSmartSDLC – AI-Enhanced Software Development Lifecycle

Team ID: LTVIP2025TMID29288

Team Size: 4

Team Leader: Anamika Kumari Chouhan (22P31A0575)

Team member: Anushka Rajput (22P31A0578)

Team member: Nitin Kumar (22P31A05A9)

Team member: Vaishnavi Verma (22P31A05B8)

College: Aditya College of Engineering & Technology (Surampalem, A.P.)

1. INTRODUCTION:

1.1 Project Overview:

SmartSDLC is a full-stack, AI-driven platform that transforms the traditional Software Development Lifecycle (SDLC) by integrating Natural Language Processing (NLP), generative AI, and intelligent automation into each phase of development. Built using Python, Streamlit for the frontend, and optionally FastAPI for backend services, it leverages advanced AI models from Hugging Face or IBM Watsonx.

The platform allows users to upload unstructured requirement documents, which are automatically classified into SDLC phases and converted into structured user stories. It can generate clean, production-ready code from natural language prompts, create automated test cases, fix buggy code, and summarize complex logic into human-readable documentation. A built-in AI chatbot provides real-time support, helping users with SDLC-related queries. Overall, SmartSDLC reduces manual effort, accelerates development, and enhances software quality through intelligent automation.

Core Objectives:

- Automate multiple SDLC stages using generative AI
- Enable seamless translation of natural language into code, test cases, and documentation
- Provide real-time chatbot support to assist users with SDLC-related queries
- Improve software quality and development speed through AI-enhanced workflows

Scenarios:

Scenario 1: Requirement Upload & Classification

Purpose: This scenario automates the initial phase of the software lifecycle — gathering and organizing requirements.

How it works:

- The user uploads a PDF file that contains raw, unstructured requirements (like business goals or client expectations).
- The backend extracts text from the PDF using a library called PyMuPDF (also called fitz).
- AI processes each sentence and classifies it into categories like:
    - Requirements
    - Design
    - Development
    - Testing
    - Deployment
- These are displayed as structured user stories, which help in planning and assigning tasks.

Why it's useful: It saves time by replacing manual requirement analysis and helps developers better understand what needs to be built.

Scenario 2: AI Code Generator

Purpose: To quickly turn user stories or plain-text instructions into working code.

How it works:

- The user types a prompt like "Create a login system using Python and Flask."
- The prompt is sent to the Hugging Face or Watsonx generative model.
- The AI returns relevant, clean, and formatted code.
- The frontend displays the output with syntax highlighting for readability.

Why it's useful: Speeds up development, especially for repetitive or boilerplate code, and is helpful for junior developers or prototyping.

Scenario 3: Bug Fixer

Purpose: To help developers identify and fix coding errors without manually debugging.

How it works:

- The user pastes a snippet of buggy code (e.g., Python, JavaScript).
- The AI model analyzes the code and finds syntax or logical errors.
- It returns a corrected version, sometimes with explanations.

Why it's useful: Reduces the time spent on debugging and helps new developers learn best practices by example.

Scenario 4: Test Case Generator

Purpose: To automate the creation of test cases for validating code logic.

How it works:

- The user provides a code snippet or functional description (like "a function that calculates the area of a circle").
- The AI generates unit tests using frameworks such as unittest or pytest (for Python).
- These tests can be directly copied into the codebase for validation.

Why it's useful: Ensures better test coverage, promotes quality assurance, and saves time writing tests manually.

Scenario 5: Code Summarizer

Purpose: To generate human-readable explanations of source code.

How it works:

- The user submits a block of code.
- The AI reads the code and outputs a plain-English summary.
- It explains what the code does, what its inputs and outputs are, and how it might be used.

Why it's useful: Helpful for documentation, code reviews, or when onboarding new developers to understand existing logic.

Scenario 6: Floating AI Chatbot Assistant

Purpose: To assist users in real-time with SDLC questions and navigation.

How it works:

- A chatbot (using LangChain + Hugging Face or Watsonx) is embedded in the frontend.
- The user can ask questions like:
    - "How do I write a unit test?"
    - "What are the phases of SDLC?"
    - "What does this error mean?"
- The chatbot understands the question and gives an informative, AI-generated response.

Why it's useful: Acts like a virtual mentor or project assistant. Very helpful for new developers or non-technical users who need quick answers.

1.2 Purpose:

The purpose of the **SmartSDLC – AI-Enhanced Software Development Lifecycle** project is to **automate and optimize key phases of the software development lifecycle using AI technologies**, specifically IBM Watsonx Granite models and LangChain. It aims to:

1. **Reduce manual effort** in requirement analysis, code generation, testing, bug fixing, and documentation.
2. **Improve developer productivity and code quality** through AI-assisted features.
3. **Accelerate SDLC workflows** with faster and more intelligent tooling.
4. **Enable non-technical users** to interact with SDLC tasks using a simple Streamlit-based interface.
5. **Demonstrate the power of AI integration** in modern software engineering using cloud-native, modular architecture.

---

2. IDEATION PHASE:

2.1 Problem Statement:

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love.

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

I am	Describe customer with 3-4 key characteristics - who are they?	Describe the customer and their attributes here
I'm trying to	List their outcome or "job" the care about - what are they trying to achieve?	List the thing they are trying to achieve here
but	Describe what problems or barriers stand in the way – what bothers them most?	Describe the problems or barriers that get in the way here
because	Enter the "root cause" of why the problem or barrier exists – what needs to be solved?	Describe the reason the problems or barriers exist
which makes me feel	Describe the emotions from the customer's point of view – how does it impact them emotionally?	Describe the emotions the result from experiencing the problems or barriers

Example:

Customer Problem Statement (PS)

PS-1

I am a software developer in an agile development team.

I'm trying to translate client requirements into structured user stories, generate initial code, write test cases, and fix bugs rapidly.

But the current software development process is heavily manual, time-consuming, and often error-prone.

Because there is no unified, intelligent system that automates these SDLC phases efficiently using AI.

Which makes me feel overloaded, inefficient, and frustrated due to repetitive and tedious tasks.

PS-2

I am a product owner or non-technical stakeholder.
I'm trying to communicate business goals and requirements clearly to the development team and get timely feedback.
But technical jargon and complex SDLC processes make it hard to track how requirements are implemented.
Because traditional tools don't allow me to interact with or validate SDLC outputs in a user-friendly way.
Which makes me feel disconnected from the development process and unsure of product alignment.

2.2 Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

Example:

User Role:

Software Developer / Project Manager involved in agile software development.

---

1. Says

- "I spend too much time writing boilerplate code."
- "Debugging takes forever, especially for legacy code."
- "I wish there was a tool that could automate test writing."
- "Turning client requirements into technical documents is exhausting."
- "We're always under pressure to release faster."

---

2. Thinks

- "What if I make a mistake while converting vague requirements into code?"
- "AI could really help us accelerate development if integrated well."
- "Manual processes slow us down and introduce human errors."
- "I need something intelligent that understands both business language and code."

---

3. Does

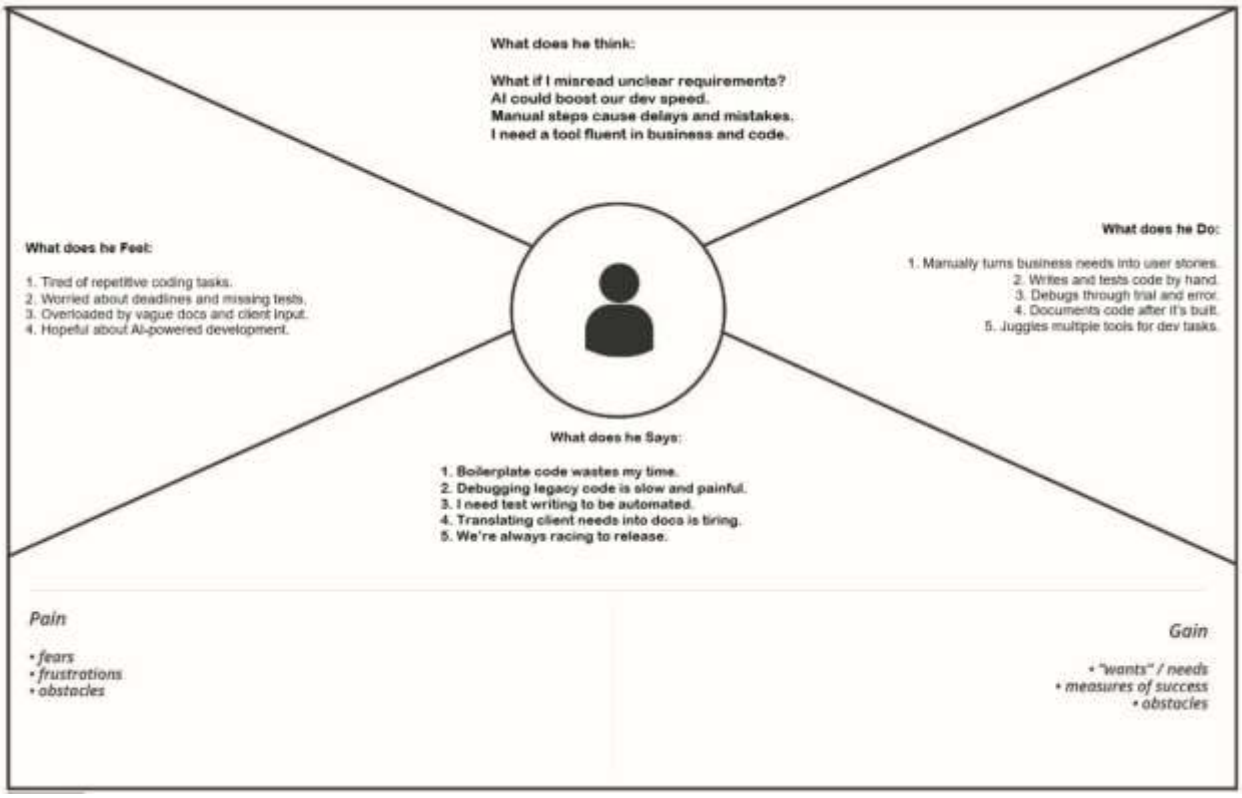
- Convert business requirements into user stories.
- Writes and tests code manually.
- Debugs errors through trial and error.
- Documents software modules post-development.
- Uses multiple tools for communication, coding, and testing.

---

4. Feels

- Frustrated with repetitive and mundane coding tasks.
- Stressed about timelines and incomplete test coverage.
- Overwhelmed by long documents and vague client instructions.
- Curious and hopeful about AI-enhanced development tools.

- **Diagram**



2.3 Brainstorming:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

2.3.1: Team Gathering, Collaboration and Select the Problem Statement

Selected Problem Statement:
There is no unified AI-powered solution to automate and simplify the Software Development Lifecycle (SDLC), which causes delays, inefficiencies, and a steep learning curve for both developers and non-technical stakeholders. This leads to slow delivery, communication gaps, and increased development effort.

2.3.2: Ideas Generated and Grouped by Category:

Idea No.	Idea Description	Group/Category
1	Build a platform that accepts unstructured PDF requirements and classifies them automatically	Requirement Analysis
2	Generate production-ready code from natural language descriptions	Code Generation
3	Create test cases using AI based on given code	Testing Automation
4	Implement AI-based bug fixing system	Debugging/Bug Resolution
5	Summarize and document code using generative AI	Documentation
6	Add a floating AI chatbot to assist with SDLC concepts and platform usage	Chatbot Assistant
7	Collect user feedback to improve AI outputs	Feedback Loop
8	Integrate GitHub for pushing AI-generated code, tracking issues, and syncing documentation	DevOps/GitHub Integration

2.3.3: Prioritization

Priority	Idea Description	Reason for Priority
High	Requirement classification from uploaded PDFs using AI	Solves initial bottleneck and saves planning time
High	Code generation from user stories using Hugging Face model	Automates a core SDLC task with high impact
High	Test case generation for provided code	Ensures quality and accelerates testing
Medium	Bug fixing using AI to correct logical/syntactical code errors	Reduces debugging time
Medium	Floating AI chatbot using LangChain to guide users	Enhances accessibility for all user types

Medium	Code summarization and documentation	Supports maintainability and team onboarding
Low	GitHub integration for syncing generated outputs	Useful for teams, not critical in MVP phase
Low	Feedback system for improving AI responses over time	More valuable in long-term iterations

---

3. REQUIREMENT ANALYSIS:

3.1 Customer Journey Map:

Stage	User Goal	Touchpoints	User Actions	Pain Points	AI-Driven Solution
Awareness	Discover a tool to speed up SDLC tasks	Social media, GitHub, College/Company demos	Learns about SmartSDLC features	Unsure about AI's usefulness in coding	Interactive chatbot explains features and benefits
Onboarding	Register and access the platform	Streamlit Web UI, Login Page	Signs up via form, Google, or LinkedIn	Complex onboarding, time-consuming setup	Simple sign-up, instant access to dashboard
Requirement Input	Upload requirements for processing	"Upload PDF" page	Uploads raw requirements in PDF	Unstructured inputs, hard to extract useful data	AI extracts text and classifies it into SDLC phases
Development	Generate code and test cases	Code Generator, Test Generator	Inputs prompt/user story, receives code and tests	Writing boilerplate code, lack of test coverage	AI generates clean code and test cases in seconds
Debugging	Fix errors in buggy code	Bug Fixer Page	Submits code snippets with bugs	Identifying and fixing bugs is slow	AI auto-fixes code and provides clean version

Documentation	Create summaries and technical docs	Code Summarizer Page	Uploads or selects code blocks	Writing docs is time-consuming	AI generates clear summaries and documentation
Feedback Loop	Provide feedback or request improvements	Feedback Page	Rates output, suggests improvements	Frustration if output is inaccurate	Feedback helps improve model prompts and responses
Integration	Save or share work via GitHub	GitHub Integration Page	Pushes code, opens issues	Manual Git tasks are tedious	AI automates GitHub interactions
Retention	Reuse platform for future projects	Home Page, Saved Sessions	Logs in again for new SDLC tasks	Difficulty reusing old sessions	Platform saves session history and learns user preferences

3.2 Solution Requirements (Functional & Non-functional):

- **Functional Requirements:**

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIn
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	AI-Powered Requirement Analysis	Upload PDF of requirements Extract and classify into SDLC phases Generate user stories
FR-4	Multilingual Code Generation	Generate Python or other language code from user stories Generate boilerplate code

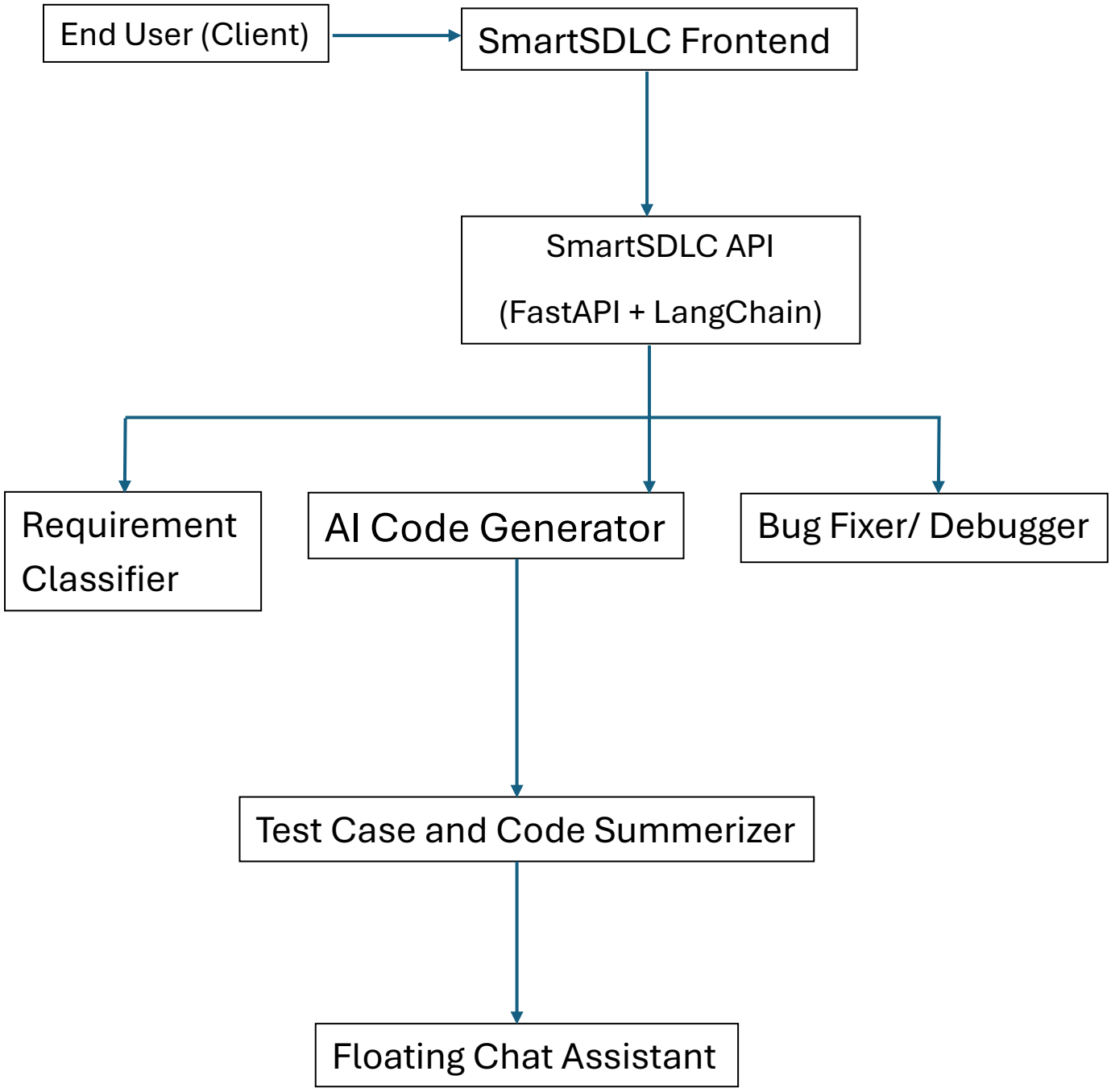
FR-5	Test Case Generation	Create unit and integration tests from code or prompts
FR-6	Bug Detection and Auto-Fixing	Identify and fix bugs in user-submitted code Return corrected code
FR-7	Code Summarization and Documentation	Summarize uploaded/generated code Generate technical documentation
FR-8	Interactive AI Chatbot	Chat interface for development help and suggestions
FR-9	Feedback Collection	Capture user feedback on AI output and usability
FR-10	GitHub Integration	Push generated code to GitHub Create issues and sync documentation

---

- **Non-functional Requirements:**

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	Intuitive UI using Streamlit with simple navigation and user-friendly workflow
NFR-2	Security	Secure user authentication; protect API keys and user data
NFR-3	Reliability	Ensure consistent AI response quality; fallback handling in case of failures
NFR-4	Performance	Fast response times for code generation and bug fixing
NFR-5	Availability	Local deployment availability with future support for cloud deployment
NFR-6	Scalability	Modular backend structure supports future scale-out with additional services

3.3 Data Flow Diagram (DFD) Level 0:



User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
Developer	Requirement Classification	USN-1	As a developer, I can upload a PDF file of raw requirements and see them classified.	I can see clearly grouped requirements by SDLC phase.	High	Sprint-1

Developer	AI Code Generation	USN-2	As a developer, I can input a natural language prompt and receive production-ready code.	I receive working, syntax-highlighted code on the frontend.	High	Sprint-1
QA Engineer	Test Case Generation	USN-3	As a tester, I can input code and get auto-generated test cases.	I see test cases using unittest or pytest formats.	Medium	Sprint-2
Developer	Bug Fixer	USN-4	As a developer, I can submit buggy code and get the fixed version with explanation.	I get a corrected version of my code.	High	Sprint-2
Developer	Code Summarization	USN-5	As a developer, I can get human-readable summaries of any code snippet.	I receive a readable explanation of code functions.	Medium	Sprint-2
End User (Any Role)	Floating AI Chatbot Assistant	USN-6	As a user, I can ask SDLC-related queries to an AI chatbot.	I receive accurate, contextual SDLC answers in real-time.	High	Sprint-2

3.4 Technology Stack (Architecture & Stack):

A. Technical Architecture Overview:

The architecture integrates:

- **Frontend (Streamlit)** for user interaction.
- **Backend (FastAPI)** for routing and logic handling.
- **IBM Watsonx (Granite Models)** and **LangChain** for AI capabilities.
- **External services** like GitHub for code repository actions.
- **PDF processing** via PyMuPDF and feedback logging via file/DB.

B. Components & Technologies Used:

S.No	Component	Description	Technology
1	User Interface	Web UI interface for interacting with SmartSDLC features	Streamlit (Python), HTML/CSS
2	Application Logic-1	SDLC automation logic (PDF parsing, prompt generation, AI query, etc.)	Python (FastAPI), PyMuPDF
3	Application Logic-2	AI-based code generation, summarization, bug fixing	IBM Watsonx Granite Models
4	Application Logic-3	AI chatbot for SDLC guidance and Q&A	LangChain, Watsonx AI
5	Database	Stores user feedback, login info, session details	JSON/File-based (or SQLite optional)
6	Cloud Database	For future cloud persistence of data (optional)	IBM Cloudant or IBM DB2
7	File Storage	Requirement documents and generated outputs	Local Filesystem
8	External API-1	AI Foundation Model API (IBM Watsonx)	IBM Watsonx APIs
9	External API-2	GitHub Integration for repo operations	GitHub REST APIs
10	Machine Learning Model	NLP + Code Generation + Classification	granite-13b-chat-v1, granite-20b-code-instruct
11	Infrastructure	Runs locally, but scalable to cloud deployments	Local Server (Uvicorn), IBM Cloud

---

C. Application Characteristics:

S.No	Characteristics	Description	Technology
1	Open-Source Frameworks	Frameworks used for full-stack development	FastAPI, Streamlit, LangChain, PyMuPDF
2	Security Implementations	Hashed passwords, API key management using environment variables (.env)	bcrypt, dotenv, OAuth (planned), HTTPS
3	Scalable Architecture	Microservices with clear routing and modular services	FastAPI modular routers, service layers
4	Availability	Accessible locally, cloud-compatible with CI/CD and Docker in future plans	Uvicorn, IBM Cloud Foundry (future)
5	Performance	Fast inference via optimized AI calls, async FastAPI routes, lightweight UI	Async FastAPI, Caching (planned), Streamlit

4. PROJECT DESIGN:

4.1 Problem – Solution Fit:

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why.

Purpose:

- Solve complex problems in a way that fits the state of your customers.
- Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behavior.
- Sharpen your communication and marketing strategy with the right triggers and messaging.
- Increase contact with your company by finding the right problem-behavior fit and building trust by solving frequent annoyances, or urgent or costly problems.
- Understand the existing situation to improve it for your target group.

Diagram:

Customer Segment

Software developers, project managers, and QA engineers working in agile software development teams. These are the users who directly interact with the SDLC and feel its pain points most acutely. They need a tool that simplifies their workflow and reduces manual effort.

Problem

Traditional Software Development Life Cycle (SDLC) processes are often time-consuming, highly manual, and lack comprehensive automation. This leads to delays, increased costs, and developer frustration. The lack of integration between various SDLC stages hinders overall team productivity and efficiency.

Existing Solutions

Currently, agile teams rely on a collection of individual tools for each stage of the SDLC. Examples include Jira for issue tracking, VS Code for coding, and Postman for API testing. These tools require significant manual effort, constant tool switching, and often lack seamless integration, creating bottlenecks and inefficiencies.

Key Challenges

Agile teams face several critical challenges throughout the SDLC:

- **Converting Requirements into Actionable Tasks:** Translating user stories and requirements into concrete development tasks is often a manual and error-prone process.
- **Writing Comprehensive Test Cases:** Generating effective test cases requires a deep understanding of the software and can be tedious and time-consuming.
- **Efficient Debugging:** Identifying and resolving bugs can be a lengthy process, especially when relying on manual debugging techniques.
- **Thorough Documentation:** Creating and maintaining accurate documentation is often neglected due to time constraints, leading to knowledge gaps and maintainability issues.

Unique Solution: SmartSDLC

SmartSDLC is an AI-powered, full-stack platform designed to automate and streamline SDLC tasks. It provides a unified, integrated environment that leverages AI to assist with requirements analysis, test case generation, debugging, and documentation. By automating these key areas, SmartSDLC significantly reduces manual effort, accelerates development cycles, and improves software quality.

Behavioral Insight

Software developers and agile teams prefer integrated and automated tools that seamlessly fit into their existing workflows. They are more likely to adopt solutions that minimize context switching, reduce repetitive tasks, and enhance collaboration. An aversion to manual processes and a strong desire for efficiency drive their tool choices.

Why It Works

SmartSDLC effectively addresses the problem-solution fit by:

- **AI-Powered Automation:** SmartSDLC leverages AI algorithms to automate critical SDLC tasks, significantly reducing manual effort and accelerating development cycles.
- **Streamlined Workflow:** The integrated platform eliminates the need for constant tool switching, creating a seamless and efficient workflow.
- **Improved Collaboration:** SmartSDLC enhances collaboration by providing a centralized platform for managing requirements, code, tests, and documentation.
- **Matching Developer Habits:** By automating tasks and providing intelligent assistance, SmartSDLC aligns with developers' preferred working styles and reduces frustration.

In essence, SmartSDLC solves the pain points of traditional SDLC by seamlessly integrating with and automating existing developer workflows, rather than trying to replace them. It improves efficiency and reduces friction.

4.2 Proposed Solution:

S.No.	Parameter	Description
1	**Problem Statement**	Traditional SDLC processes involve multiple tools and extensive manual work in requirement analysis, coding, testing, and documentation, leading to delays, errors, and inefficiencies in software development.
2	**Idea / Solution Description**	SmartSDLC is a full-stack, AI-powered platform that automates the software development lifecycle using IBM Watsonx and LangChain. It supports automated requirement classification, code generation, bug fixing, test case generation, code summarization, and AI-based help.
3	**Novelty / Uniqueness**	Unlike traditional systems, SmartSDLC combines multiple SDLC functions into a unified platform driven by large language models. It offers contextual code generation, intelligent bug resolution, and seamless test case creation from natural language.
4	**Social Impact / Customer Satisfaction**	The platform empowers both technical and non-technical users, boosts development team productivity, reduces release cycles, improves accuracy, and enhances collaboration between stakeholders.
5	**Business Model (Revenue Model)**	Freemium model: Basic functionalities free for individuals and small teams; paid plans for enterprises with features like usage-based APIs, chatbot support, and integration with DevOps pipelines.
6	**Scalability of the Solution**	Built on scalable backend architecture and cloud-compatible services, SmartSDLC can serve individual developers to large organizations, supporting continuous upgrades and integration with CI/CD systems.

4.3 Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

SmartSDLC – Solution Architecture

SmartSDLC is a cloud-integrated, AI-powered platform that leverages modular microservices to automate core phases of the Software Development Lifecycle (SDLC). It uses Watsonx large language models for intelligent processing and LangChain for interactive chatbot capabilities. The architecture follows a layered pattern for scalability, maintainability, and modularity.

Key Components:

1. Frontend: A React-based user interface that allows interaction with modules such as requirement upload, AI code generation, test generation, and the chatbot assistant.

2. Backend (FastAPI): Serves as the middleware to route user inputs, handle authentication, and invoke specific microservices.

3. AI Services:

- Watsonx (Granite-20B): Powers the NLP understanding for requirement classification, code generation, test case suggestion, summarization, and debugging.
- LangChain: Handles contextual prompt routing and real-time chatbot interaction.

4. Document Processor: Uses PyMuPDF for extracting text from uploaded PDF requirement documents.

5. Database (MongoDB/PostgreSQL): Stores classified requirements, code outputs, test cases, and user sessions.

6. API Gateway & Load Balancer: Ensures scalable API access and high availability.

7. Cloud Integration (Optional): Deployable on AWS/Azure with serverless functions and CI/CD pipelines.

Example - Solution Architecture Diagram:

Figure 1: Architecture and data flow of Software Development Lifecycle

5. PROJECT PLANNING AND SCHEDULING:

5.1 Project Planning Phase

Project Planning (Product Backlog, Sprint Planning, Stories, Story points)

Sprint	Feature	User Story Number	User Story/ Task	Story Points	Priority	Team Members
Sprint-1	Requirement Classification	USN-01	As a user, I can upload a PDF of requirements and have it classified automatically by SDLC phase.	3	High	Member 1
Sprint-1	AI Code Generation	USN-02	As a user, I can input a prompt and get production-ready code using AI.	5	High	Member 2
Sprint-1	Documentation UI Setup	USN-07	As a user, I can view the classified outputs in a structured and readable format.	2	Medium	Member 3
Sprint-1	FastAPI Backend Setup	USN-08	Set up backend routes for requirement classification and AI prompts.	2	Medium	Member 4
Sprint-2	Test Case Generator	USN-03	As a user, I can enter a code snippet or feature description and get autogenerated test cases.	4	Medium	Member 1
Sprint-2	Bug Fixer	USN-04	As a user, I can paste code with bugs and get an AI-fixed version of the same.	4	High	Member 2
Sprint-2	Code Summarizer	USN-05	As a user, I can paste any code and get a human-readable explanation.	2	Medium	Member 3
Sprint-2	Floating AI Chatbot	USN-06	As a user, I can interact with a floating AI chatbot to get SDLC help.	4	High	Member 4

Workload Distribution (Total Story Points):

Member	Assigned Features	Total Story Points
Member 1	USN-01, USN-03	3 + 4 = 7
Member 2	USN-02, USN-04	5 + 4 = 9
Member 3	USN-07, USN-05	2 + 2 = 4
Member 4	USN-08, USN-06	2 + 4 = 6

This keeps the overall effort balanced. You can replace the Member 1/2/3/4 with actual names.

Project Tracker, Velocity & Burndown Chart

Project Duration: 20 June 2025 – 27 June 2025
Total Story Points: 24
Team Members: 4
Average Velocity: 24/8=3

Progress Tracker Table (Suggested Format)

Date	Planned Remaining Points	Actual Remaining Points	Comments
20 June 2025	24	24	Project Initiation
21 June 2025	21	21	Requirement module started
22 June 2025	18	18	Code Generator module initiated
23 June 2025	15	15	Initial testing modules added
24 June 2025	12	12	Bug fixer integrated
25 June 2025	9	8	Code summarizer almost complete
26 June 2025	6	5	AI Chatbot integrated
27 June 2025	3	0	Final deployment & documentation

Burndown Chart Description

- The Planned Line drops steadily from 24 → 0 in 8 days.
- The Actual Line closely follows planned progress.
- Slight improvement seen on 25–27 June due to efficient task execution.

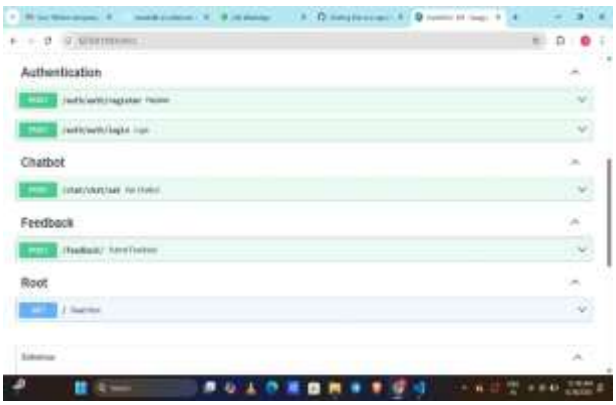
6. PERFORMANCE TESTING:

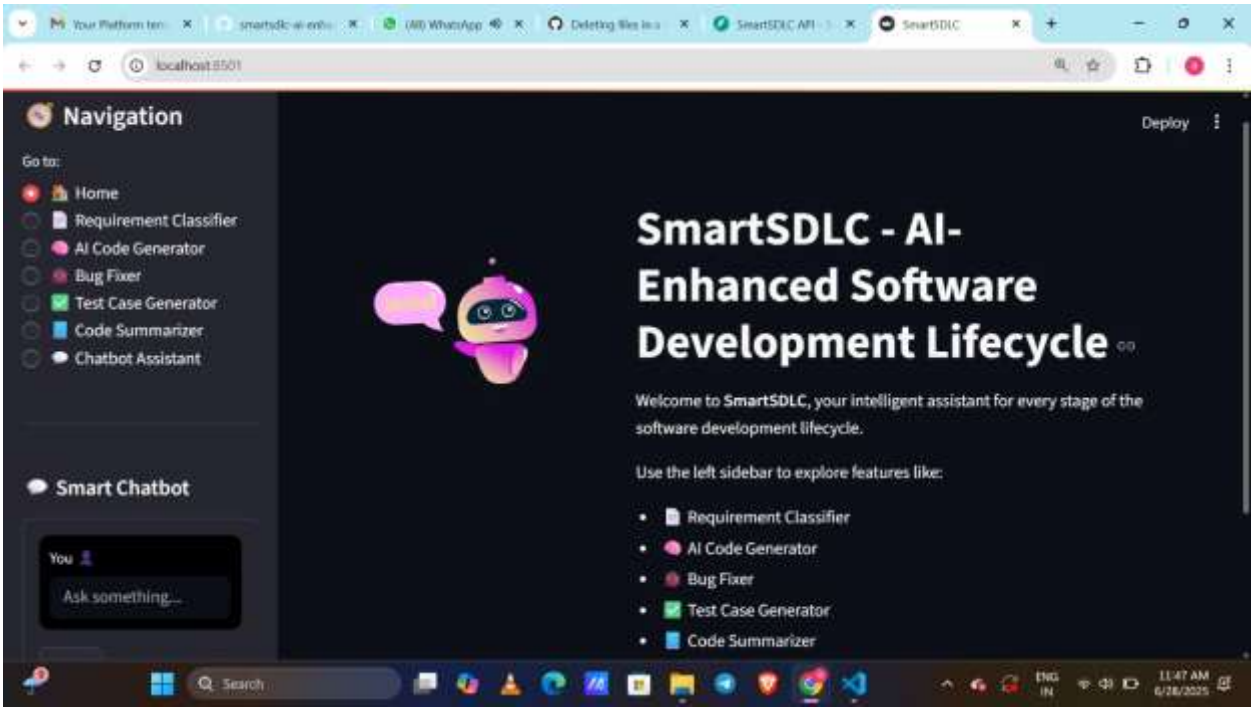
Test Scenarios & Results (based on SmartSDLC features)

Test Case ID	Scenario (What to test)	Test Steps (How to test)	Expected Result	Actual Result	Pass/Fail
FT-01	Text Input Validation (e.g., prompts, requirements)	Enter valid/invalid requirements or prompts	Valid inputs accepted, errors shown for invalid formats	As expected	Pass
FT-02	PDF Upload & Extraction	Upload requirement PDF with mixed content	Extracts structured text using PyMuPDF	As expected	Pass
FT-03	AI Code Generation	Input user story and trigger Watsonx model	Generates production-quality code	As expected	Pass

FT-04	Bug Fixer Functionality	Input buggy code and receive corrected output	Returns fixed version with explanation	As expected	Pass
FT-05	Test Case Generator	Input code block or requirement and click “Generate Tests”	Outputs valid unit/pytest test cases	As expected	Pass
FT-06	Code Summarizer	Input a source code block	Returns natural language summary of function/purpose	As expected	Pass
FT-07	Floating AI Chatbot (LangChain)	Ask questions like “What is unit testing?”	Returns helpful AI-based SDLC answers	As expected	Pass
Test Case ID	**Scenario (What to test)**	**Test Steps (How to test)**	**Expected Result**	**Actual Result**	**Pass/Fail**
PT-01	Content Generation Response Time	Measure time for Watsonx to return code/test after input	Should respond in under 3 seconds	2.4 seconds	Pass
PT-02	Concurrent API Load (Parallel Requests)	Simulate 10 simultaneous user requests	No significant slowdown or failure	As expected	Pass
PT-03	PDF Upload Stress Test	Upload 5–10 large requirement PDFs in succession	Handles uploads without crash	As expected	Pass
PT-04	Multi-Feature Usage Performance	Use code generation, test generator, summarizer simultaneously	System remains stable and responsive	As expected	Pass

7. RESULTS:





8. ADVANTAGES AND DISADVANTAGES:

Advantages of SmartSDLC:

1. **Automation of SDLC Tasks**: Automates key phases like requirement analysis, code generation, testing, bug fixing, and documentation using IBM Watsonx and LangChain.
2. **Time and Cost Efficiency**: Reduces manual effort, accelerates development time, and minimizes errors, making the development cycle faster and cost-effective.
3. **Natural Language Interaction**: Accepts user inputs in plain English and generates structured output, enabling even non-technical stakeholders to contribute.
4. **Modular & Scalable Architecture**: Clean separation of front end and backend with support for microservices ensures scalability and easy maintainability.
5. **AI-Powered Chatbot**: Offers 24x7 smart assistant to guide users throughout the SDLC process.
6. **PDF Requirement Extraction**: Extracts and classify SDLC tasks directly from uploaded documents for a seamless workflow.
7. **Seamless Integration**: Integration with GitHub allows continuous code updates, issue tracking, and workflow synchronization.

Disadvantages of SmartSDLC:

1. **Model Dependency**: Relies heavily on Watsonx Granite models—performance and availability are limited to IBM's cloud services.

2. **Data Privacy Concerns**: Handling sensitive code or business requirements may raise compliance and data privacy issues in enterprise use.
3. **Limited Offline Functionality**: Requires continuous internet and API access to Watsonx; not suitable for air-gapped environments.
4. **Learning Curve for Non-Developers**: While AI helps, initial setup (like virtual environments, API keys) might be technical for non-programmers.
5. **Inference Latency**: AI model response times can be affected by network delays or API load, especially for large PDF inputs or complex prompts.

9. CONCLUSION:

The **SmartSDLC** platform demonstrates how generative AI can revolutionize traditional software development by streamlining every major phase with minimal human intervention. From requirement analysis and code generation to test creation, bug fixing, and feedback, SmartSDLC enhances developer productivity, ensures accuracy, and fosters agile development practices. Its clean architecture, intuitive interface, and seamless AI integration make it a powerful tool for modern software engineering teams.

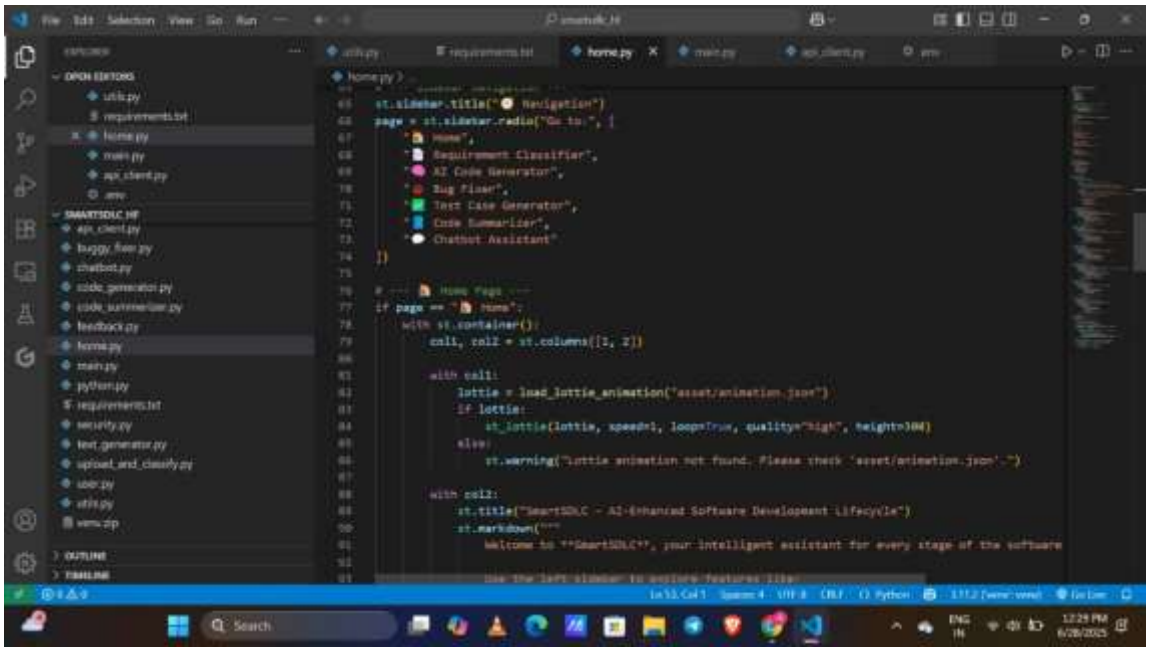
10. FUTURE SCOPE:

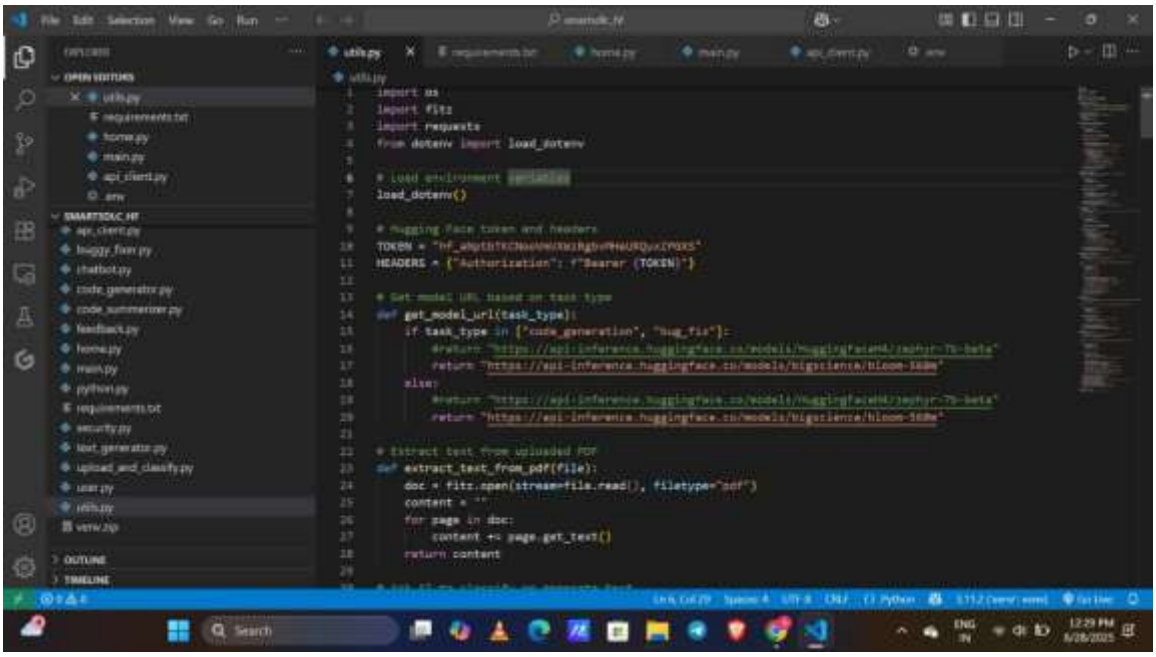
1. **CI/CD Integration**: Extend the platform to support CI/CD pipelines for automatic testing, build, and deployment.
2. **Team Collaboration Features**: Add multi-user collaboration modules for real-time project development and review.
3. **Custom Model Support**: Introduce plug-and-play support for other AI providers like OpenAI, Google Vertex AI, or local LLMs.
4. **Enhanced Chatbot Memory**: Upgrade the assistant to support contextual memory and deeper user history for intelligent suggestions.
5. **Cloud Deployment**: Move from local to hybrid or full cloud hosting (e.g., IBM Cloud Kubernetes, AWS EC2) for scalability and global access.

Voice-Based Interaction: Integrate speech-to-text functionality for voice-controlled SDLC operations.

11. APPENDIX:

- **Sample Code –**





- **GitHub Link–** https://github.com/Anamika1511/smartsdlc-ai-enhanced-software-development

- **Project Demo Link -** https://drive.google.com/file/d/11GH-cxySXtkzS8qID6EI8HiiZvCN15Zh/view?usp=drivesdk