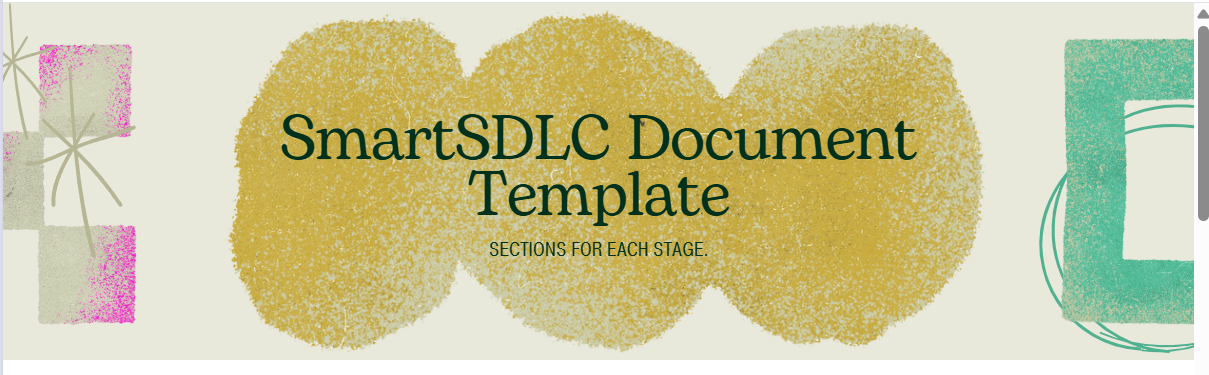
1.INTRODUCTION

1.1 Project Overview:

SmartSDLC is an AI-enhanced platform designed to automate and accelerate key phases of the Software Development Lifecycle (SDLC). It uses IBM Watsonx, FastAPI, LangChain, and Streamlit to streamline tasks like requirement analysis, code generation, testing, bug fixing, and documentation.



1.2 Purpose:

The purpose of SmartSDLC is to minimize human effort and time in software development by integrating generative AI. It aims to provide intelligent tools that enhance developer productivity, reduce errors, and automate repetitive tasks.



2. IDEATION PHASE

2.1 Problem Statement

**Customer Problem Statement Template:**

To understand the customer's point of view, it's essential to define a clear problem statement that highlights the real challenges faced during the traditional Software Development Lifecycle (SDLC). Project managers and developers often struggle with inefficient workflows, manual processes, lack of intelligent tools, and poor visibility across stages of development. These issues lead to delays, reduced productivity, and compromised software quality. By identifying these pain points, the SmartSDLC project aims to create AI-powered solutions that automate repetitive tasks, provide real-time insights, and enhance overall coordination. A well-articulated customer problem statement helps teams stay focused on what truly matters to users, allowing them to build intelligent, responsive experiences that improve the efficiency and success of software projects.

Graphical user interface, text, application, email

Description automatically generated

**Example:**

Chart, treemap chart

Description automatically generated

| **Problem Statement (PS)** | **I am (Customer)** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| --- | --- | --- | --- | --- | --- |
| PS-1 | a project manager responsible for overseeing multiple software development projects across various teams. | ensure that each stage of the Software Development Lifecycle (SDLC)-from planning to deployment is efficient, well-coordinated, and completed on time. | I face constant challenges due to lack of automation, scattered tools, inconsistent reporting, and manual tracking of tasks and progress. | traditional SDLC processes rely heavily on human input and disconnected systems, making it difficult to maintain real-time visibility and control over the entire workflow. | stressed, overwhelmed, and concerned about delays, cost overruns, and missed quality benchmarks. |
| PS-2 | a software developer working on tight sprint cycles with high expectations for productivity and code quality. | deliver well-tested, maintainable code quickly, while keeping up with dynamic changes in requirements. | I spend excessive time on repetitive coding tasks, manual debugging, and understanding unclear or outdated documentation. | my development environment lacks AI-powered tools that could assist with code generation, testing, and intelligent recommenda-tions. | frustrated, inefficient, and under continuous pressure to meet deadlines without compromising quality. |

2.2 Empathy Map Canvas

This empathy map illustrates the thoughts, pains, goals, and behavior of a developer who wants to write clean code faster. By understanding their frustrations (e.g., manual bug fixing, unclear requirements) and hopes (automation, better tools), we gain insights into their needs. This helps us design solutions that truly align with the user’s experience and expectations.

Reference: <https://www.mural.co/templates/empathy-map-canvas>

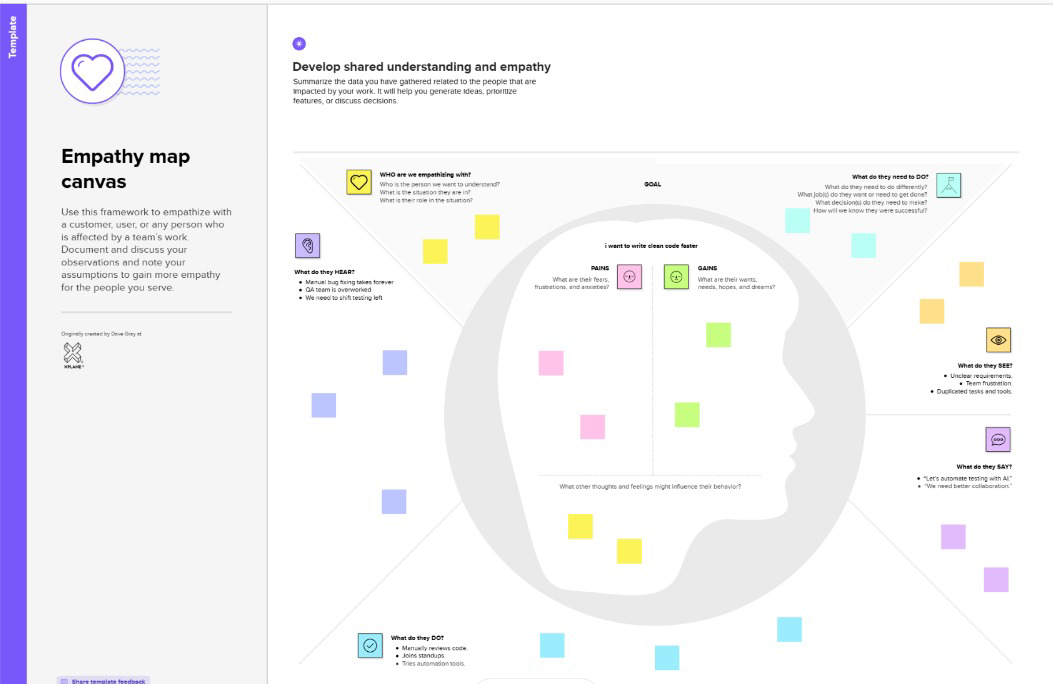
**S – Scope:**  
Define the problem.

**M – Model:**  
Create a user-centered model.

**A – Analyze:**  
Understand root causes.

**R – Redesign:**  
Propose solutions.

**T – Transform:**  
Implement and test solutions.



2.3 Brainstorming

**Brainstorm & Idea Prioritization Template:**  
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.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

In the ideation phase of the SmartSDLC project, the team collaboratively identified the key problem statement: how to automate and optimize core phases of the Software Development Life Cycle (SDLC) using AI technologies like IBM Watsonx and LangChain. During brainstorming, ideas were listed and grouped around core functionalities such as AI-powered requirement analysis, multilingual code generation, test case creation, bug fixing, code documentation, chatbot assistance, feedback collection, and GitHub integration. These ideas were then prioritized based on their impact and implementation effort. High-priority items included automated requirement classification from PDFs, AI-driven code generation. Features like chatbot guidance, GitHub workflow automation, and PDF export were marked for medium or future-phase development. This process laid the foundation for building a modular, AI-enhanced SDLC automation platform with practical, high-value features prioritized for early development.

**Step-1: Team Gathering, Collaboration and Select the Problem Statement**

Graphical user interface, application

Description automatically generated

**Step-2: Brainstorm, Idea Listing and Grouping**Graphical user interface, treemap chart

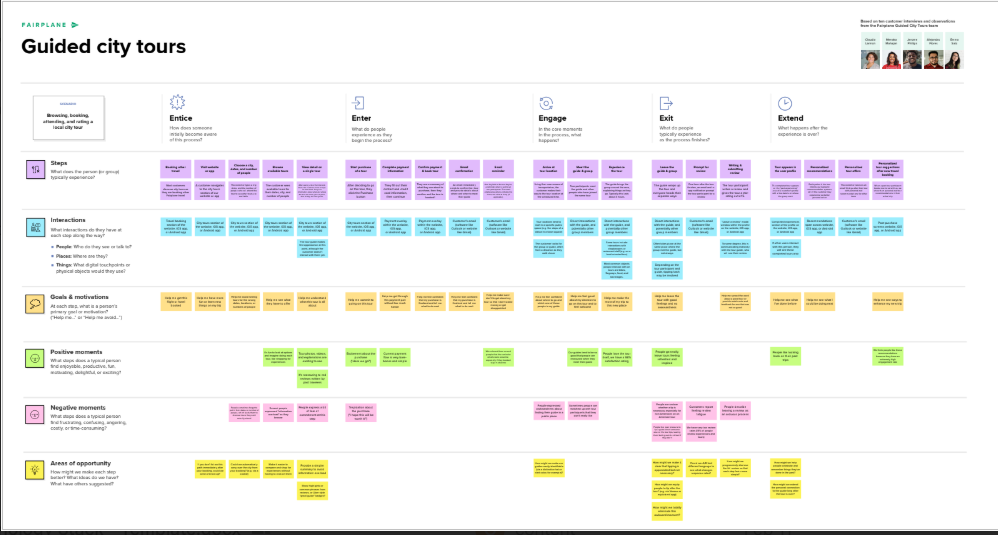
Description automatically generated

**Step-3: Idea Prioritization**

Diagram

Description automatically generated

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map:

3.2 Solution Requirement:

Functional Requirements:

Following are the functional requirements of the proposed solution.

| **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 Code Assistant | Auto-suggest code snippets using context Support for multiple languages (Python, Java, JS) |
| FR-4 | Requirement Analysis Automation | Upload and parse requirements document Generate user stories Tag entities using NLP |
| FR-5 | Project Dashboard | View tasks, commits, test status, and AI insights |
| FR-6 | Voice-to-Task Automation | Convert spoken inputs to tasks using STT |

**Non-functional Requirements:**

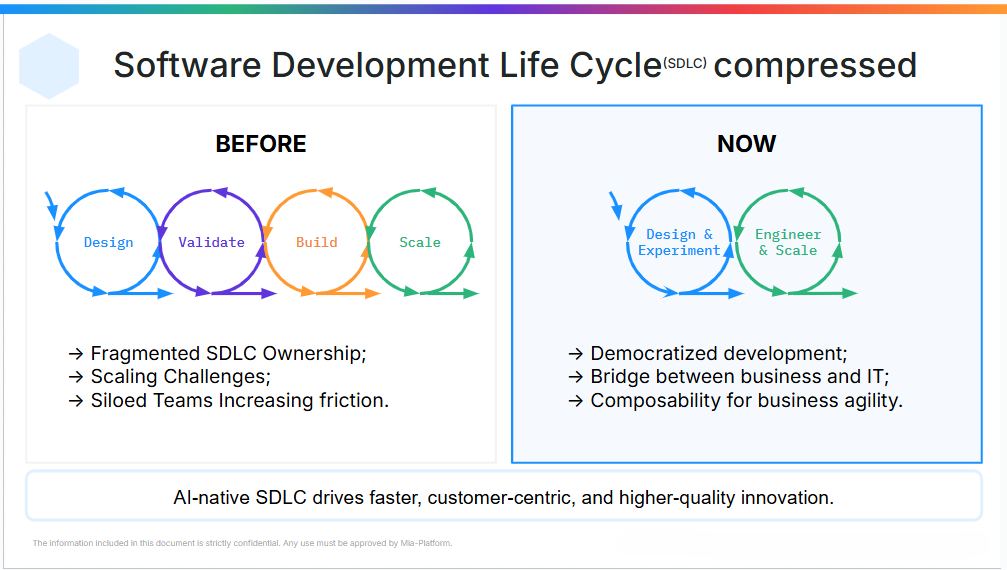
Following are the non-functional requirements of the proposed solution

| **FR No.** | **Non-Functional Requirement** | **Description** |
| --- | --- | --- |
| NFR-1 | **Usability** | User-friendly dashboard with minimal learning curve for developers |
| NFR-2 | **Security** | Role-based access control, OAuth2 login, HTTPS, JWT authentication |
| NFR-3 | **Reliability** | Redundant architecture, backup services, fault-tolerant components |
| NFR-4 | **Performance** | Capable of handling 100+ concurrent users; AI services respond within 2s |
| NFR-5 | **Availability** | 99.9% uptime via cloud deployment with distributed services and load balancing |
| NFR-6 | **Scalability** | Microservices architecture and Kubernetes orchestration for horizontal scaling |

3.3 Data Flow Diagram (DFD):

**Data Flow Diagrams:**

The Data Flow Diagram (DFD) for the SmartSDLC – AI-Enhanced Software Development Lifecycle project illustrates how data moves across various components of the system. At its core, the system begins with user interactions such as registration and task input through a web interface or voice input. These inputs are processed by AI modules—such as a requirement analyzer, code assistant, and task generator—which utilize services like IBM Watson STT and AI models for automation. The processed data flows into storage systems like IBM Cloudant and GitHub repositories, while project status, analytics, and code suggestions are presented back to users through a unified dashboard. External APIs like Jira and GitHub support real-time syncing and version control. The DFD clearly visualizes how SmartSDLC integrates AI to enhance software development workflows, from planning and coding to deployment and monitoring.

**Example:** 

Diagram, timeline

Description automatically generated

**User Stories**

Use the below template to list all the user stories for the product.

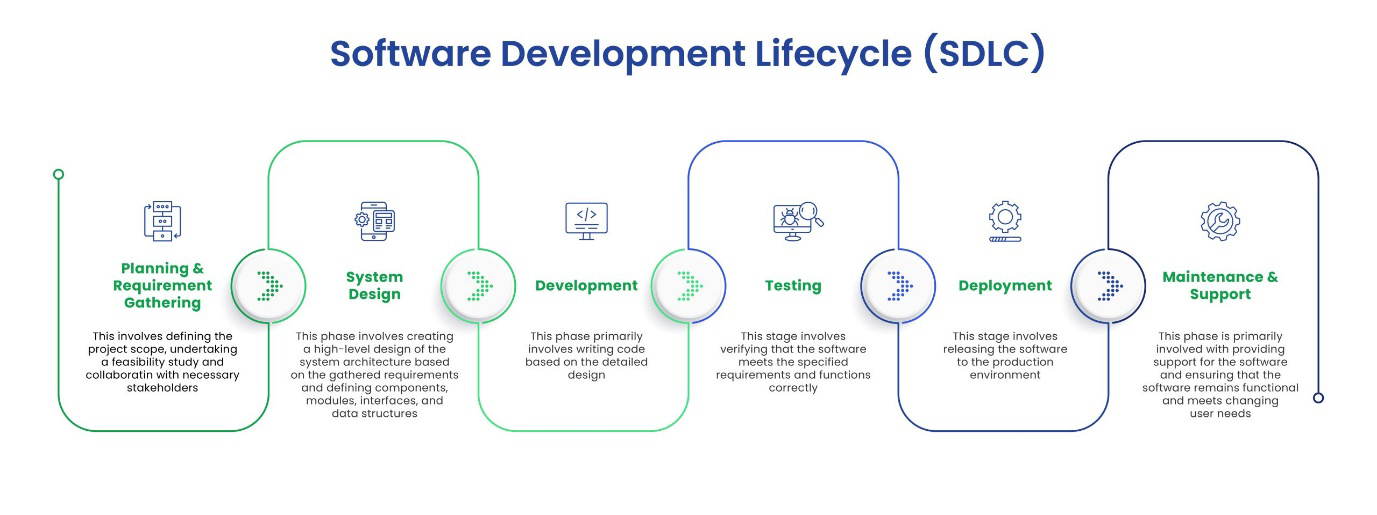
| **User Type** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Acceptance criteria** | **Priority** | **Release** |
| --- | --- | --- | --- | --- | --- | --- |
| Customer (Mobile user) | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | I can access my account / dashboard | High | Sprint-1 |
|  |  | USN-2 | As a user, I will receive a confirmation email once I have registered for the application. | I can receive confirmation email & click confirm | High | Sprint-1 |
|  |  | USN-3 | As a user, I can register for the application through Facebook. | I can register & access the dashboard with Facebook Login | Low | Sprint-2 |
|  |  | USN-4 | As a user, I can register for the application through Gmail. | I can register & access the dashboard using Gmail login | Medium | Sprint-1 |
|  | Login | USN-5 | As a user, I can log into the application by entering email & password. | I am redirected to the dashboard after successful login | High | Sprint-1 |
|  | Dashboard | USN-6 | As a user, I can view my current tasks and project progress on a dashboard. | Dashboard shows task list, status updates, and recommendations | High | Sprint-2 |
| Customer (Web user) | Task Management | USN-7 | As a user, I can create, edit, and delete development tasks. | Tasks reflect correctly in project tracker | High | Sprint-2 |
| Customer Care Executive | Support Ticket Management | USN-8 | As a support executive, I can view and respond to user queries and tickets. | I can respond and mark tickets as resolved | Medium | Sprint-3 |
| Administrator | User & Role Management | USN-9 | As an admin, I can add, edit, and remove users and assign roles. | User roles are correctly applied and updated in the system | High | Sprint-1 |
|  | System Monitoring | USN-10 | As an admin, I can monitor AI module performance and service health. | System shows uptime and error logs for AI services | High | Sprint-3 |
|  | Integration with GitHub & Jira | USN-11 | As an admin, I can configure and manage integrations with GitHub and Jira. | Sync with GitHub and Jira is functional and logs changes | Medium | Sprint-3 |
|  | AI Code Assistance | USN-12 | As a user, I can receive AI-generated code suggestions for selected tasks. | AI provides relevant suggestions within the IDE or dashboard | High | Sprint-3 |

3.4 Technology Stack:

**Technical Architecture:**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

**Example:** **AI-Augmented SmartSDLC for Agile Software Teams**   
**Reference:** <https://aws.amazon.com/blogs/apn/transforming-the-software-development-lifecycle-sdlc-with-generative-ai/>



**Table-1 : Components & Technologies:**

| **S.No** | **Component** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | User Interface | Web dashboard for managing projects and viewing AI insights | React.js, HTML5, CSS3 |
|  | Application Logic-1 | AI-based requirement analysis and document parsing | Python (Flask), SpaCy, NLTK |
|  | Application Logic-2 | Transcribe voice meetings for task logging | IBM Watson Speech to Text |
|  | Application Logic-3 | Chatbot support for answering SDLC queries | IBM Watson Assistant |
|  | Database | Stores user data, project metadata, task logs | MongoDB (NoSQL), MySQL |
|  | Cloud Database | Cloud-hosted database for real-time syncing | IBM Cloudant |
|  | File Storage | Code files, generated reports, documentation | IBM Block Storage, Local Filesystem |
|  | External API-1 | GitHub integration for CI/CD & code analysis | GitHub REST API |
|  | External API-2 | Integration with Jira for agile boards and ticketing | Jira API |
|  | Machine Learning Model | Predict bugs, generate code suggestions, and estimate effort | TensorFlow, OpenAI Codex, Scikit-learn |
|  | Infrastructure (Server / Cloud) | Cloud-native deployment with CI/CD pipeline | Kubernetes, Docker, IBM Cloud Foundry, Jenkins |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | Frameworks used for backend, frontend, and ML | Flask, React.js, TensorFlow, Kubernetes |
|  | Security Implementations | Role-based access, data encryption, secure APIs | JWT, SHA-256, OAuth2.0, HTTPS, IAM Policies |
|  | Scalable Architecture | Microservice-based deployment for each SDLC phase | Kubernetes, Docker |
|  | Availability | Ensured with replicated services and cloud load balancer | NGINX, IBM Cloud Load Balancer, Multi-Zone Setup |
|  | Performance | Use of Redis for caching, Celery for background tasks, CDN for static files | Redis, Celery, Cloudflare CDN |

**References:**

<https://developer.ibm.com/patterns/ai-powered-devops/>

<https://www.ibm.com/cloud/cloudant>

<https://www.ibm.com/cloud/watson-speech-to-text>

<https://docs.github.com/en/rest>

<https://developer.atlassian.com/cloud/jira/platform/rest/v3/>

<https://c4model.com/>

<https://www.ibm.com/cloud/architecture>

1. PROJECT DESIGN

4.1 Problem Solution Fit

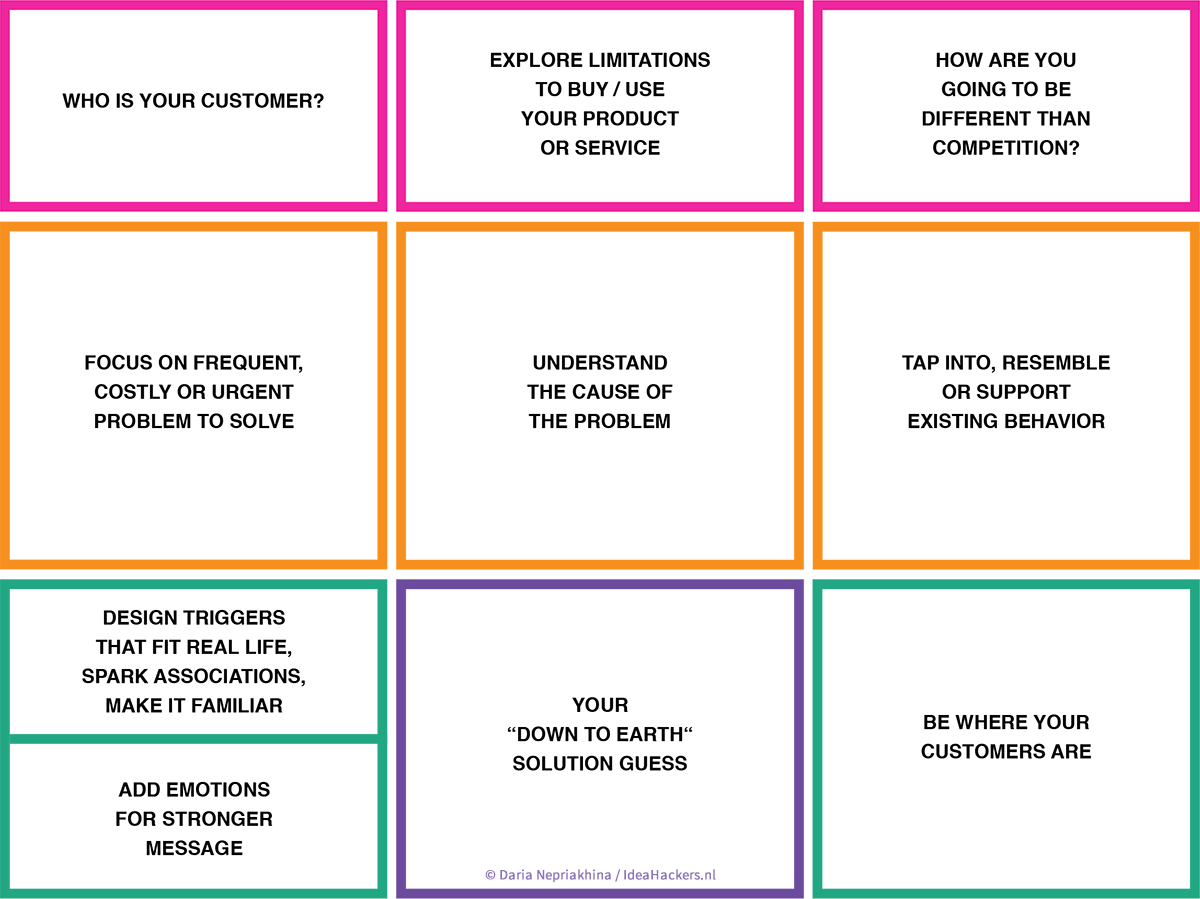
**Problem – Solution Fit Template:**

The SmartSDLC framework addresses the inefficiencies of traditional Software Development Lifecycles when applied to AI and machine learning projects. Conventional SDLC models are not optimized for iterative data workflows, frequent model tuning, or continuous deployment. SmartSDLC introduces automation, intelligence, and agility at each phase—data collection, preprocessing, model building, testing, and deployment—allowing faster feedback, adaptive development, and improved reliability. By integrating MLOps practices with agile sprint planning, it ensures efficient use of resources, reduces time-to-market, and aligns better with the evolving needs of modern AI-driven businesses.

**Purpose:**

* Solve complex ML development issues with a structured and intelligent workflow.
* Increase adoption by aligning with existing agile and DevOps behaviors.
* Sharpen communication between developers, data scientists, and stakeholders.
* Improve delivery speed and product quality with continuous feedback loops.
* Build trust and reliability through real-time testing and monitoring.
* Enhance scalability and flexibility of AI product development.

**Template:**



References:

1. <https://www.ideahackers.network/problem-solution-fit-canvas/>
2. <https://aws.amazon.com/blogs/apn/transforming-the-software-development-lifecycle-sdlc-with-generative-ai/>

4.2 Proposed Solution:

**Proposed Solution Template:**

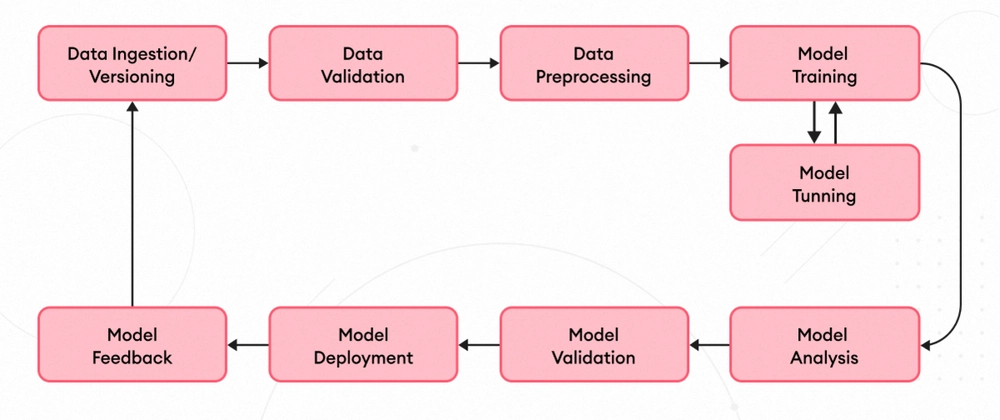
Project team shall fill the following information in the proposed solution template.

| **S.No.** | **Parameter** | **Description** |
| --- | --- | --- |
|  | Problem Statement (Problem to be solved) | Traditional software development life cycles are often inefficient or inflexible for machine learning (ML) projects, leading to delays, inconsistencies, and lack of automation. There is a need for a smarter, agile, and data-driven development model to handle ML pipelines efficiently from data collection to deployment. |
|  | Idea / Solution description | Our proposed solution introduces a Smart SDLC framework tailored for ML projects. It integrates agile sprint planning, automated data preprocessing, scalable model training, and web deployment using Flask. The solution ensures modularity, traceability, and rapid iteration across all ML development stages. |
|  | Novelty / Uniqueness | Unlike traditional SDLC models, our Smart SDLC combines principles of MLOps and agile methodology, enabling automation, sprint-based planning, and real-time feedback loops across the entire ML workflow. This fusion of DevOps and ML pipelines is rarely implemented in academic-level projects. |
|  | Social Impact / Customer Satisfaction | This solution improves the quality, speed, and reliability of deploying ML applications, leading to faster innovation cycles in sectors like healthcare, education, and e-governance. It also ensures that users experience more accurate and responsive applications through continuous integration and feedback. |
|  | Business Model (Revenue Model) | The solution can be offered as a SaaS (Software as a Service) toolkit for startups and enterprises working on AI projects. Additional revenue could be generated via customization, integration services, or cloud-based deployment support. |
|  | Scalability of the Solution | The Smart SDLC framework is highly scalable—it supports containerized deployment using tools like Docker, CI/CD pipelines, and cloud integration, making it adaptable for large-scale enterprise ML workflows or multiple project teams. |

4.3 Solution Architecture:

The solution architecture for our project outlines a structured approach to developing a machine learning application, starting from data collection and preprocessing to model building and deployment. Data is initially gathered from structured sources and processed using Python libraries like Pandas and Scikit-learn to handle missing and categorical values. The cleaned data is then used to train and test machine learning models, ensuring accurate predictions. Finally, the trained model is integrated into a user-friendly web interface using Flask and HTML for real-time interaction. This architecture ensures a seamless data flow, modular development, and scalable deployment, aligning with the project goals and agile sprint structure. The design is inspired by real-world architecture patterns, such as those presented in AWS's clinical voice application framework.

**Example - Solution Architecture Diagram:**



*Figure 1: Smart SDLC: End-to-End Machine Learning Architecture*

**Reference:** <https://medium.com/@mark.southworth98/utilising-ai-ml-to-improve-the-software-development-lifecycle-b0b6fa961cf6>

1. PROJECT PLANNING & SCHEDULING

5.1 Project Planning:

**Product Backlog, Sprint Schedule, and Estimation (4 Marks)**

Use the below template to create product backlog and sprint schedule

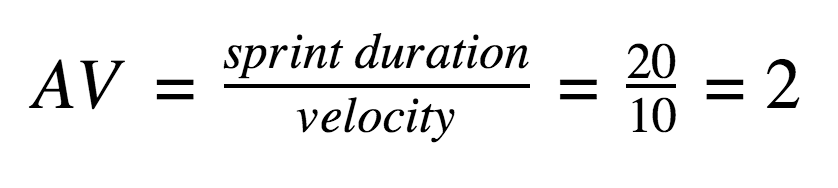
| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | 2 | High | Alice, John |
| Sprint-1 |  | USN-2 | As a user, I will receive confirmation email once I have registered for the application | 1 | High | Alice |
| Sprint-2 |  | USN-3 | As a user, I can register for the application through Facebook | 2 | Low | Alice |
| Sprint-1 |  | USN-4 | As a user, I can register for the application through Gmail | 2 | Medium | John |
| Sprint-1 | Login | USN-5 | As a user, I can log into the application by entering email & password | 1 | High | Bob |
| Sprint-2 | Dashboard | USN-6 | As a user, I can view upcoming events on the dashboard. | 3 | High | Alice, Bob |
| Sprint-3 | Event Registration | USN-7 | As a user, I can register for an event and receive a confirmation message. | 3 | High | Alice, Bob |
| Sprint-4 | Admin Dashboard | USN-8 | As an admin, I can create, update, and delete events. | 5 | High | John |

**Project Tracker, Velocity & Burndown Chart: (4 Marks)**

| **Sprint** | **Total Story Points** | **Duration** | **Sprint Start Date** | **Sprint End Date (Planned)** | **Story Points Completed (as on Planned End Date)** | **Sprint Release Date (Actual)** |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | 20 | 6 Days | 01 May 2025 | 06 May 2025 | 20 | 06 May 2025 |
| Sprint-2 | 20 | 6 Days | 07 May 2025 | 13 May 2025 | 20 | 13 May 2025 |
| Sprint-3 | 20 | 6 Days | 14 May 2025 | 20 May 2025 | 20 | 20 May 2025 |
| Sprint-4 | 20 | 6 Days | 21 May 2025 | 27 May 2025 | 20 | 27 May 2025 |

**Velocity:**

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let’s calculate the team’s average velocity (AV) per iteration unit (story points per day)



**Burndown Chart:**

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile [software development](https://www.visual-paradigm.com/scrum/what-is-agile-software-development/) methodologies such as [Scrum](https://www.visual-paradigm.com/scrum/scrum-in-3-minutes/). However, burn down charts can be applied to any project containing measurable progress over time.

[**https://www.visual-paradigm.com/scrum/scrum-burndown-chart/**](https://www.visual-paradigm.com/scrum/scrum-burndown-chart/)

[**https://www.atlassian.com/agile/tutorials/burndown-charts**](https://www.atlassian.com/agile/tutorials/burndown-charts)

**Reference:**

[**https://www.atlassian.com/agile/project-management**](https://www.atlassian.com/agile/project-management)

[**https://www.atlassian.com/agile/tutorials/how-to-do-scrum-with-jira-software**](https://www.atlassian.com/agile/tutorials/how-to-do-scrum-with-jira-software)

[**https://www.atlassian.com/agile/tutorials/epics**](https://www.atlassian.com/agile/tutorials/epics)

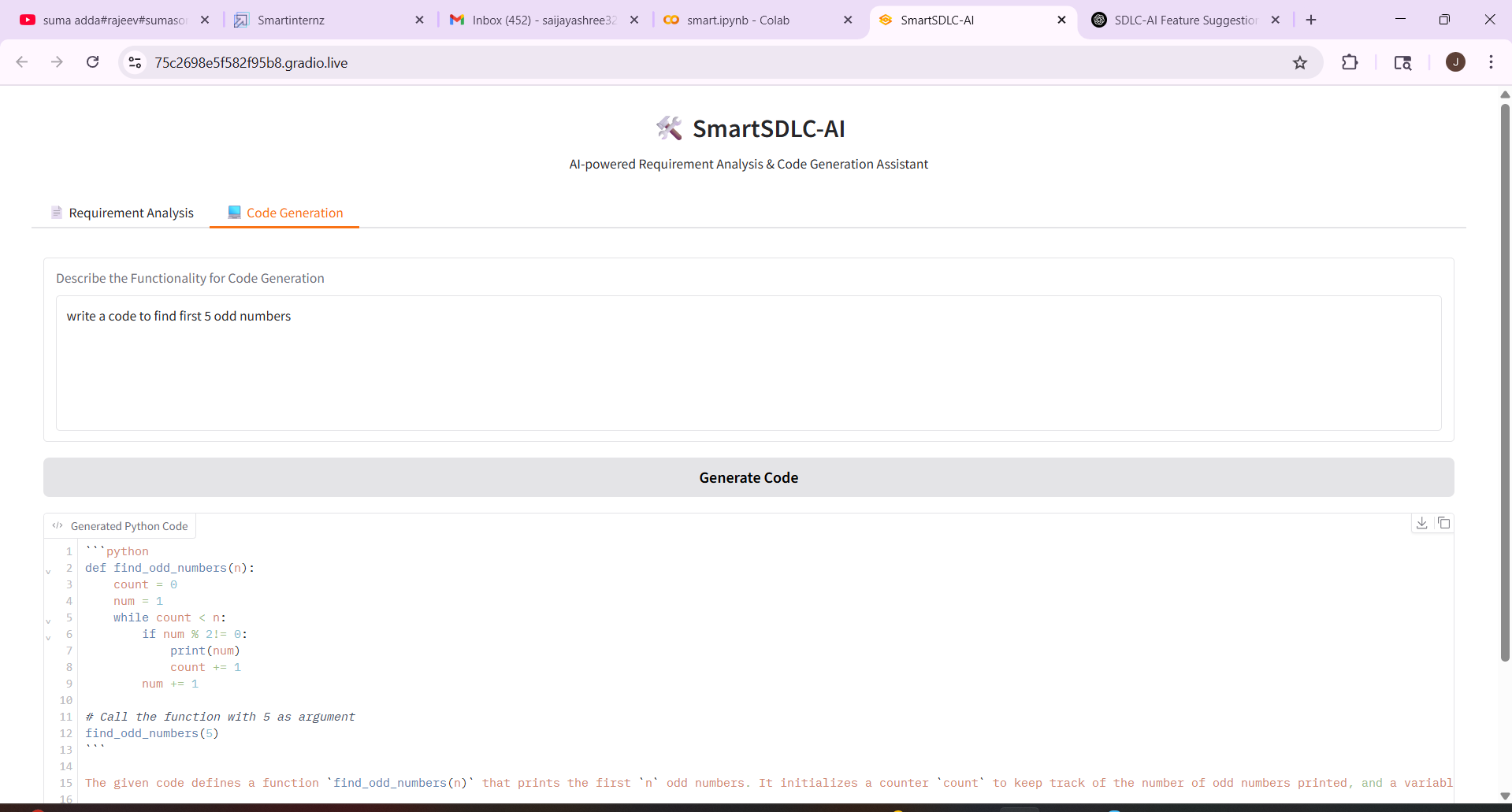
6.FUNCTIONAL AND PERFORMANCE TESTING

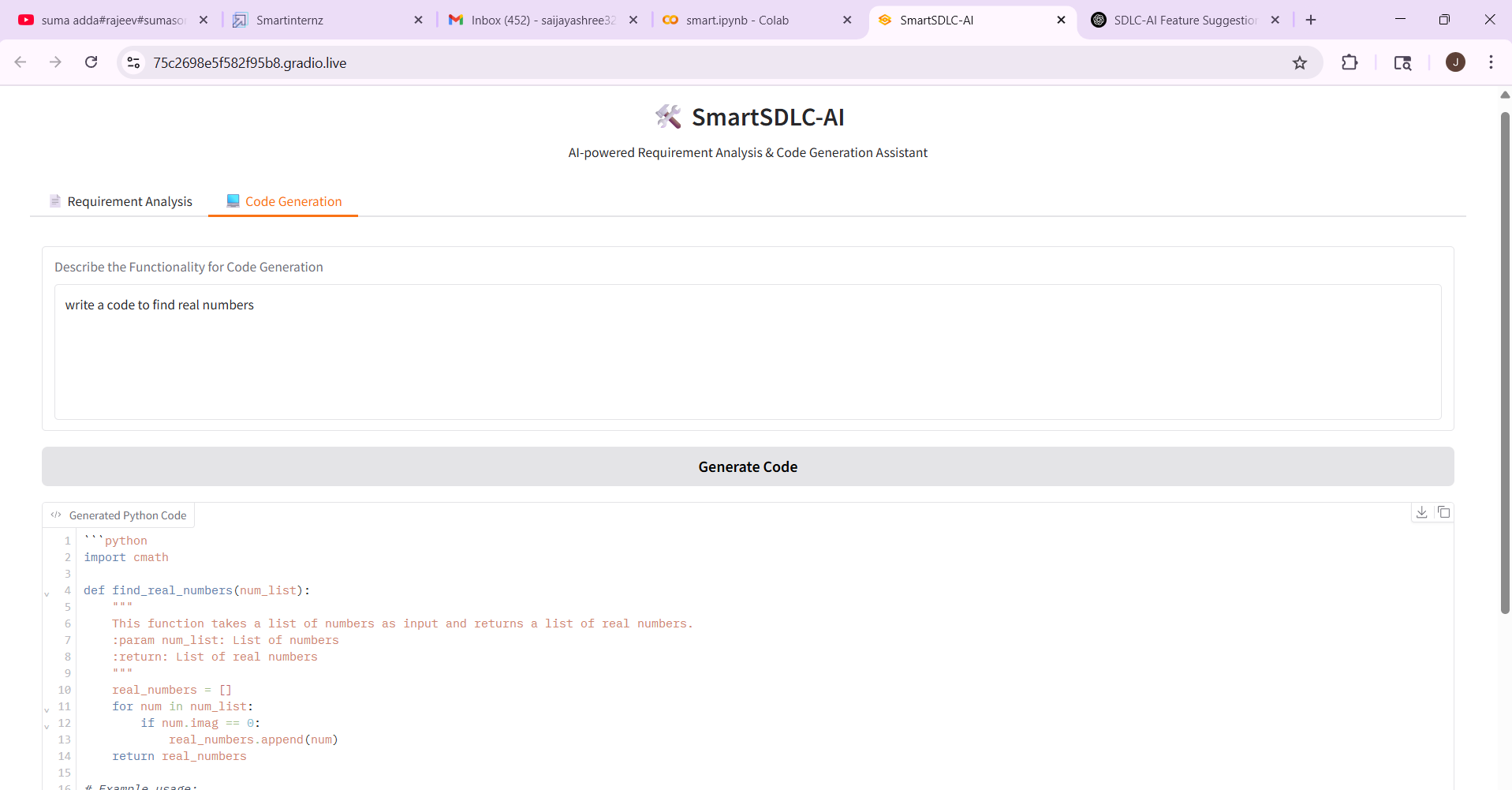
6.1 Performance Testing:

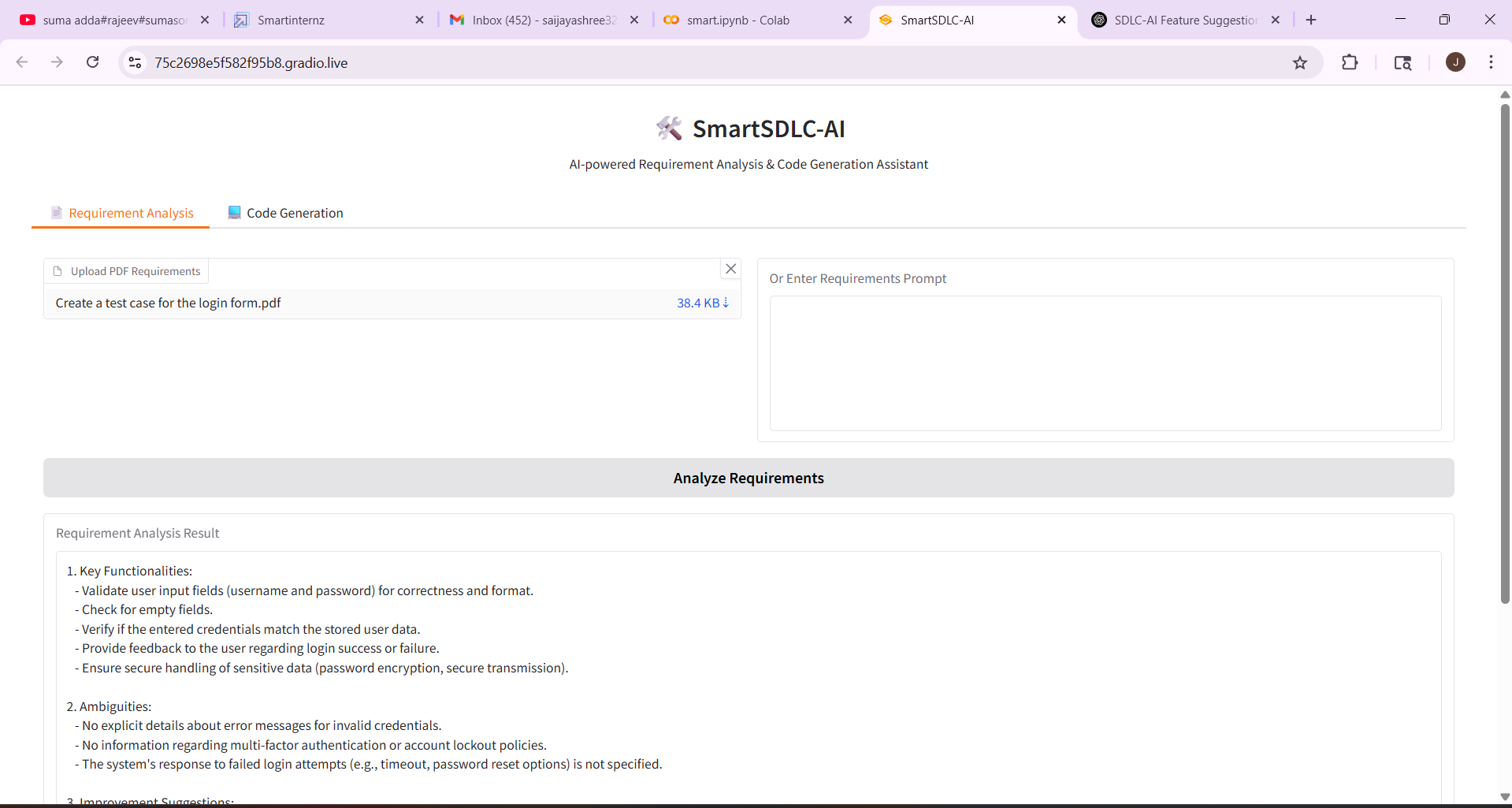
**Test Scenarios & Results**

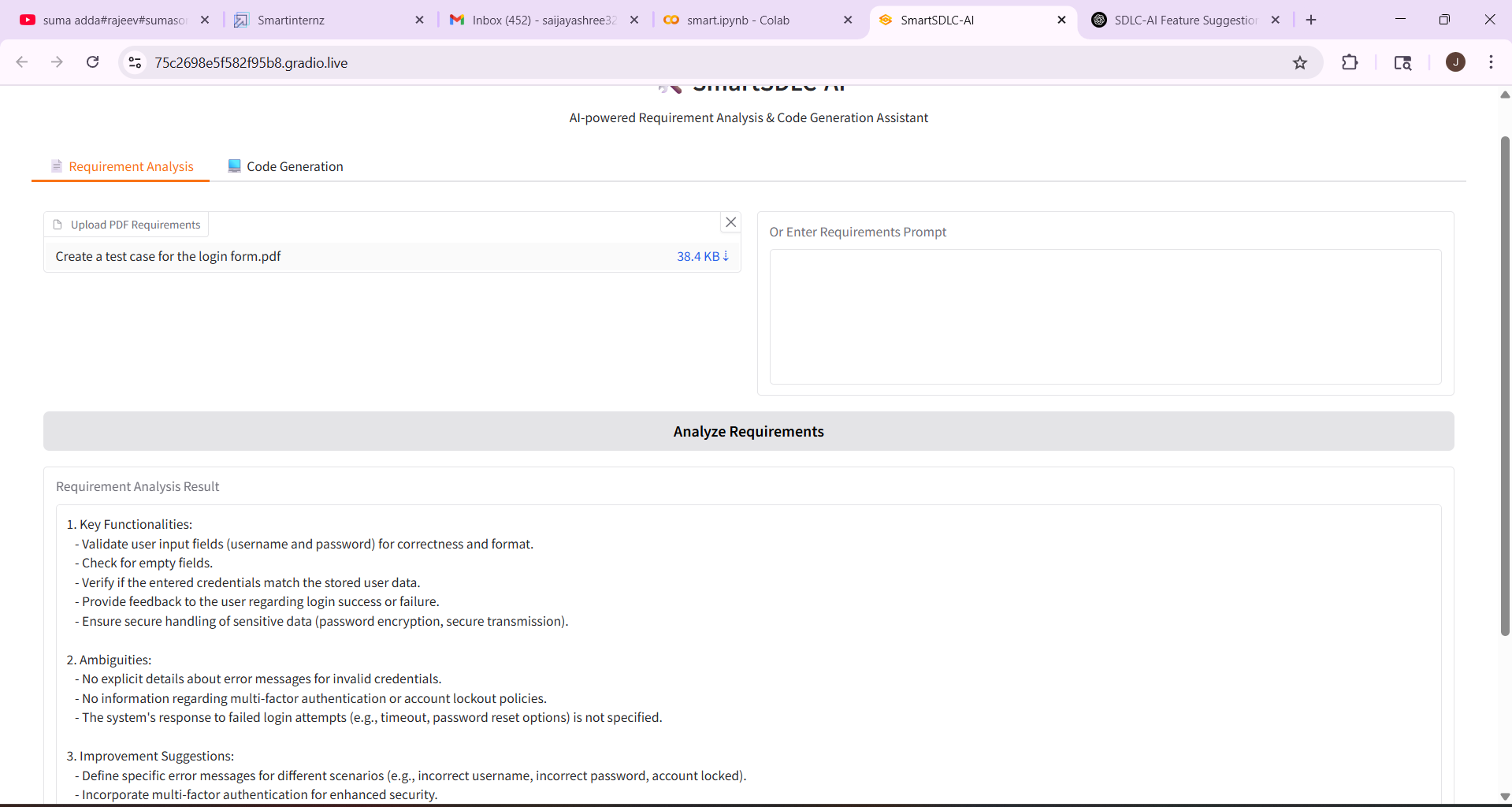
| **Test Case ID** | **Scenario (What to test)** | **Test Steps (How to test)** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| --- | --- | --- | --- | --- | --- |
| **FT-01** | Sprint and Task Input Validation | Enter valid and invalid names for sprints and tasks | Accepts valid text; displays errors for blank/invalid entries |  |  |
| **FT-02** | Story Point Range Validation | Input story points | error shown for out-of-range values |  |  |
| **FT-03** | Velocity Calculation Accuracy | Add multiple sprints and verify velocity calculation | Total story points / total sprints is calculated correctly |  |  |
| **FT-04** | Flask App Form Submission | Submit task and sprint data via form on Flask web app | Form submits successfully and data is processed |  |  |
| **PT-01** | Sprint Summary Generation Time | Measure the time it takes to generate a sprint report | Output should appear in under 3 seconds |  |  |
| **PT-02** | Flask App Load Performance | Access multiple pages of the app (home, sprint, velocity) rapidly | Pages should load without significant delay |  |  |
| **PT-03** | Concurrent Access Stability | Simulate 5 users submitting data at once | App handles all requests without crash or delay |  |  |

**7.RESULTS**

**7.1 Output Screenshots**







8. ADVANTAGES & DISADVANTAGES

**Advantages:**

* Reduces manual SDLC effort
* Faster and consistent output
* Easy to use with no technical dependency

**Disadvantages:**

* Dependent on model accuracy
* Limited by API quotas and prompt length
* Cannot fully replace human validation yet

1. **CONCLUSION**

SmartSDLC successfully automates key software development tasks using IBM Watsonx and LangChain. It improves speed, reduces errors, and provides intelligent assistance for developers through a modular, easy-to-use platform.

1. **FUTURE SCOPE**

1.CI/CD Integration

2.Multi-user collaboration features

3.GitHub version control support

4.Cloud deployment and fine-tuning with custom AI models

11. **APPENDIX**

Source Code(if any):

pip install transformers torch gradio accelerate bitsandbytes PyPDF2

# Imports

import gradio as gr

import torch

from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline

import PyPDF2

# SmartSDLC-AI Core Class

class SmartSDLC\_AI:

    def \_\_init\_\_(self):

        self.model\_name = "ibm-granite/granite-3.3-2b-instruct"

        self.tokenizer = None

        self.model = None

        self.pipeline = None

        self.load\_model()

    def load\_model(self):

        try:

            print("🔄 Loading AI model...")

            self.tokenizer = AutoTokenizer.from\_pretrained(self.model\_name, trust\_remote\_code=True)

            self.model = AutoModelForCausalLM.from\_pretrained(

                self.model\_name,

                torch\_dtype=torch.float16,

                device\_map="auto",

                trust\_remote\_code=True,

            )

            self.pipeline = pipeline(

                "text-generation",

                model=self.model,

                tokenizer=self.tokenizer,

                max\_length=1024,

                temperature=0.7,

                do\_sample=True,

                pad\_token\_id=self.tokenizer.eos\_token\_id

            )

            print("AI model loaded.")

        except Exception as e:

            print(f" Error: {e}")

            print("Falling back to DialoGPT-medium...")

            fallback\_model = "microsoft/DialoGPT-medium"

            self.tokenizer = AutoTokenizer.from\_pretrained(fallback\_model)

            self.model = AutoModelForCausalLM.from\_pretrained(fallback\_model)

            self.pipeline = pipeline(

                "text-generation",

                model=self.model,

                tokenizer=self.tokenizer,

                max\_length=1024,

                temperature=0.7,

                do\_sample=True,

                pad\_token\_id=self.tokenizer.eos\_token\_id

            )

            print(" Fallback model loaded.")

    def analyze\_requirements(self, text):

        prompt = f"You are a software requirement analysis assistant. Analyze the following requirements and list key functionalities, ambiguities, and improvement suggestions:\n\n{text}\n\nResponse:"

        response = self.pipeline(prompt)

        result = response[0]['generated\_text'].split("Response:")[-1].strip()

        return result

    def generate\_code(self, description):

        prompt = f"You are a software code generation assistant. Based on the following description, generate Python code:\n\n{description}\n\nCode:"

        response = self.pipeline(prompt)

        result = response[0]['generated\_text'].split("Code:")[-1].strip()

        return result

# PDF Text Extraction Function

def extract\_text\_from\_pdf(file\_obj):

    reader = PyPDF2.PdfReader(file\_obj)

    text = ""

    for page in reader.pages:

        text += page.extract\_text() or ""

    return text

# Gradio Interface Builder

def create\_gradio\_interface():

    with gr.Blocks(title="SmartSDLC-AI") as app:

        gr.HTML("<h1 style='text-align:center;'>🛠️ SmartSDLC-AI</h1><p style='text-align:center;'>AI-powered Requirement Analysis & Code Generation Assistant</p>")

        with gr.Tabs():

            # Requirement Analysis Tab

            with gr.Tab(" Requirement Analysis"):

                with gr.Row():

                    pdf\_input = gr.File(label="Upload PDF Requirements")

                    text\_input = gr.Textbox(label="Or Enter Requirements Prompt", lines=6)

                analyze\_btn = gr.Button("Analyze Requirements")

                analysis\_output = gr.Textbox(label="Requirement Analysis Result", lines=12)

            # Code Generation Tab

            with gr.Tab(" Code Generation"):

                code\_desc\_input = gr.Textbox(label="Describe the Functionality for Code Generation", lines=6)

                generate\_code\_btn = gr.Button("Generate Code")

                code\_output = gr.Code(label="Generated Python Code", language="python")

        # Requirement Analysis Function

        def handle\_analysis(pdf\_file, prompt\_text):

            if pdf\_file:

                text = extract\_text\_from\_pdf(pdf\_file)

            elif prompt\_text.strip():

                text = prompt\_text

            else:

                return "❗ Please upload a PDF or enter requirement text."

            result = smart\_sdlc\_ai.analyze\_requirements(text)

            return result

        analyze\_btn.click(fn=handle\_analysis, inputs=[pdf\_input, text\_input], outputs=analysis\_output)

        # Code Generation Function

        def handle\_code\_generation(desc):

            if not desc.strip():

                return "❗ Please enter a description for code generation."

            result = smart\_sdlc\_ai.generate\_code(desc)

            return result

        generate\_code\_btn.click(fn=handle\_code\_generation, inputs=code\_desc\_input, outputs=code\_output)

    gr.HTML("<p style='text-align:center; color:gray;'>⚙️ Powered by IBM Granite AI | SmartSDLC-AI for Modern Development</p>")

    return app

# 🚀 Run Application

if \_\_name\_\_ == "\_\_main\_\_":

    print("🚀 SmartSDLC-AI Initializing...")

    smart\_sdlc\_ai = SmartSDLC\_AI()

    iface = create\_gradio\_interface()

    print(" Launching SmartSDLC-AI with public link...")

    iface.launch(share=True)

GitHub link: <https://github.com/RupasreeGunigari766/Smart_sdlc>

Demo link: <https://drive.google.com/file/d/1FCjcVgRR5Q7gKYVBATj3kPDlSm2HSrlL/view?usp=sharing>