Final Report

1. INTRODUCTION

1.1 Project Overview

This project focuses on integrating Generative AI into the Software Development Life Cycle (SDLC) to enhance productivity, streamline tasks, and automate documentation. The system leverages AI to support planning, designing, testing, and reporting phases of software projects.

1.2 Purpose

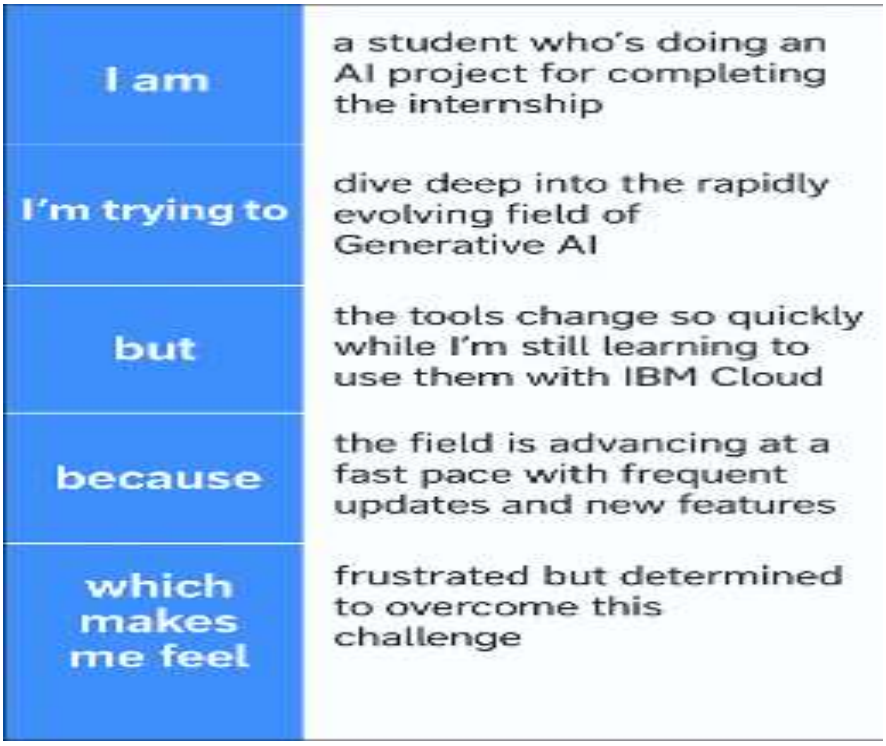
The purpose of this project is to explore how Generative AI can augment traditional SDLC workflows, reduce manual workload, and increase efficiency in software engineering processes.

2. IDEATION PHASE

2.1 Problem Statement

Ideation Phase

Define the Problem Statements

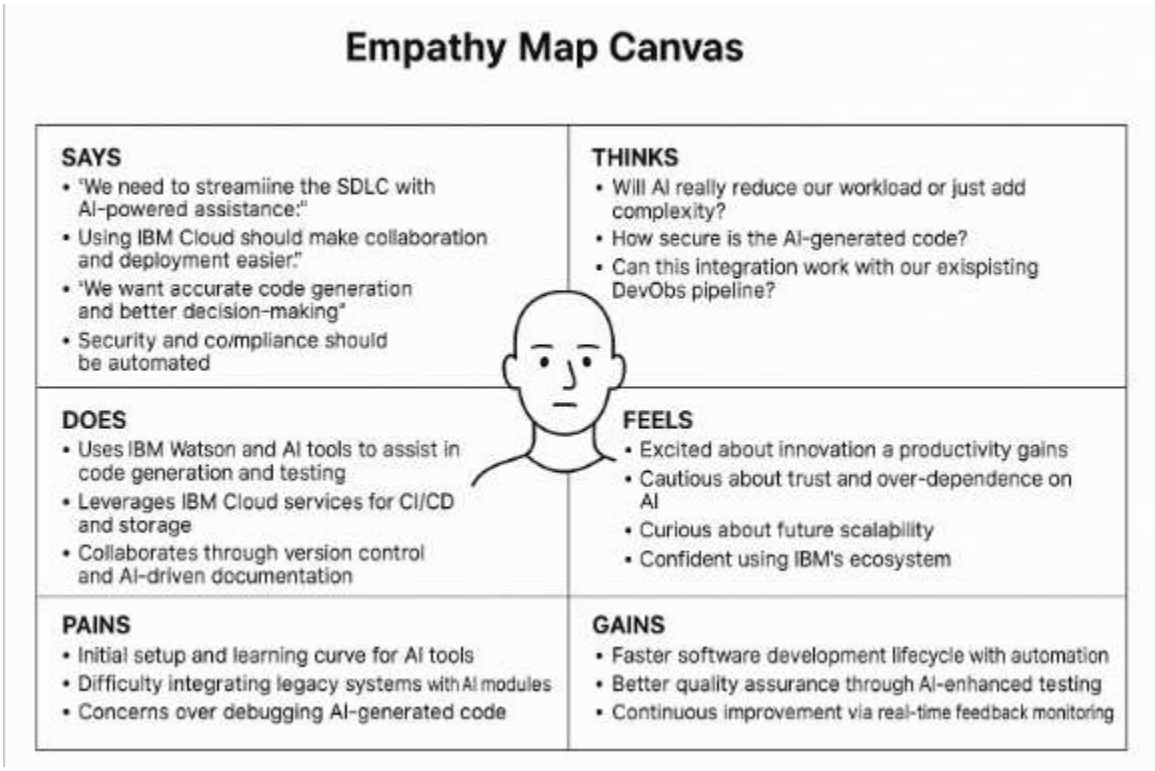


2.2 Empathy Map Canvas

Ideation Phase

Empathize & Discover

Empathy Map Canvas:

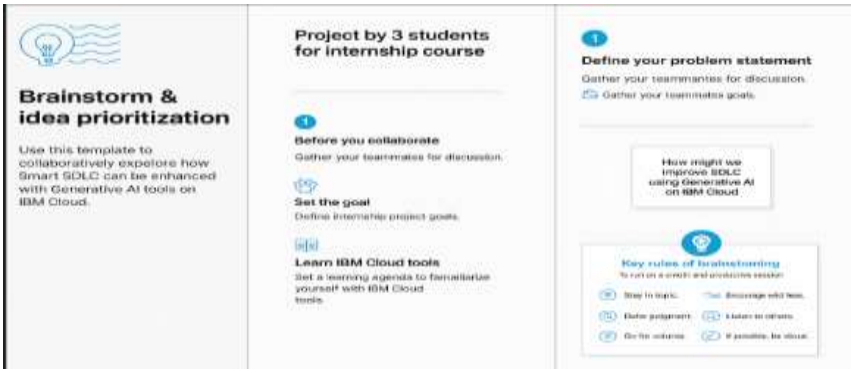


2.3 Brainstorming

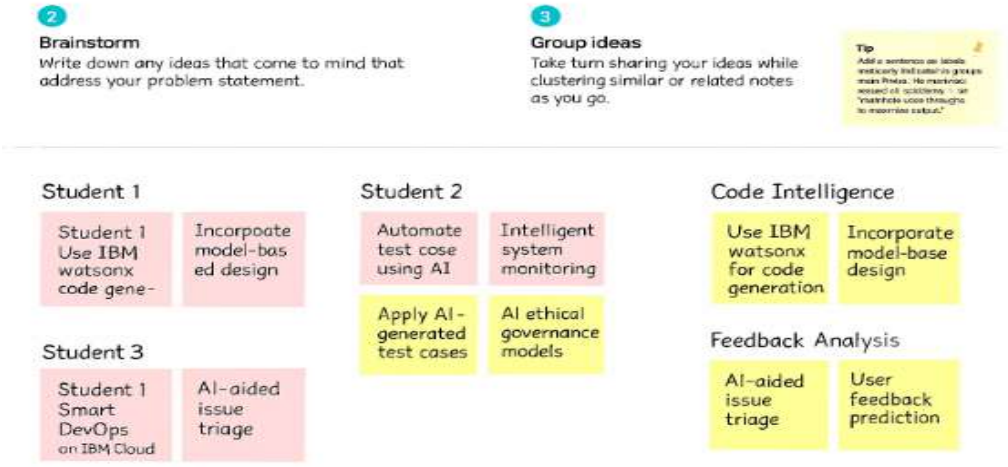
Ideation Phase

Brainstorm & Idea Prioritization Template

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



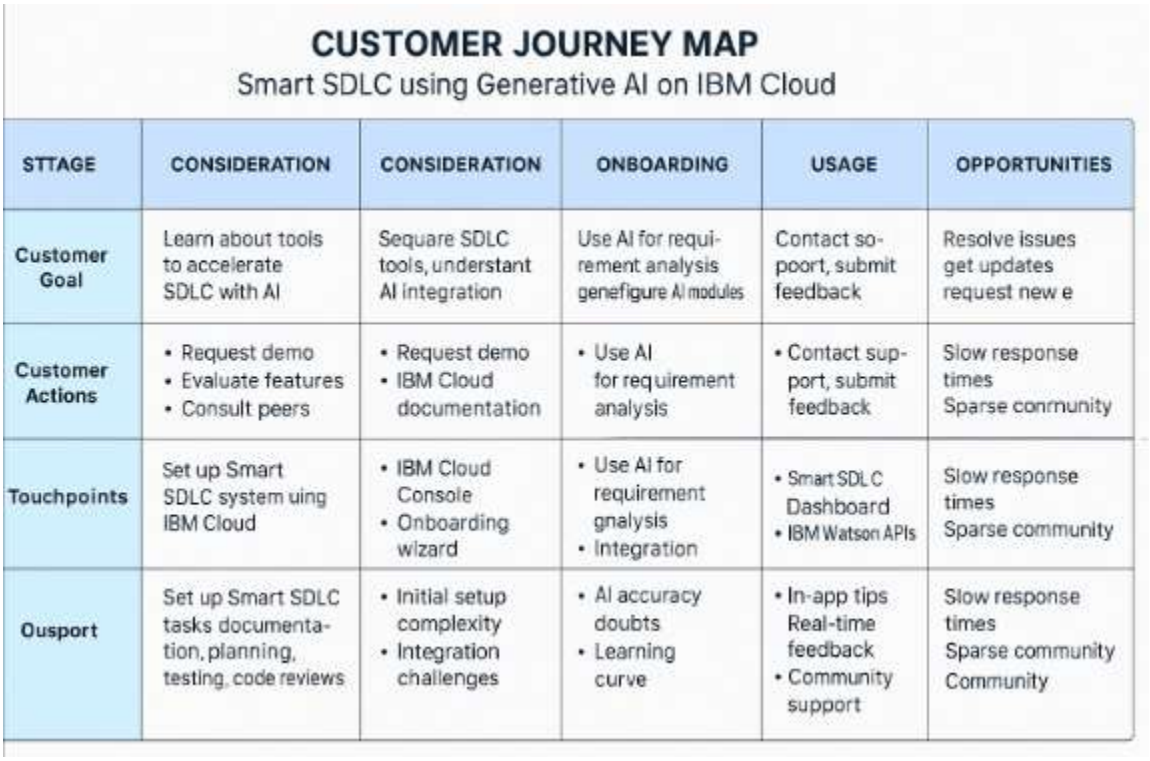
Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization

3. REQUIREMENT ANALYSIS

3.1 Customer Journey map



CUSTOMER JOURNEY MAP

Smart SDLC using Generative AI on IBM Cloud

STTAGE	CONSIDERATION	CONSIDERATION	ONBOARDING	USAGE	OPPORTUNITIES
Customer Goal	Learn about tools to accelerate SDLC with AI	Sequare SDLC tools, understant AI integration	Use AI for requi-rement analysis genefigure AI modules	Contact so-poort, submit feedback	Resolve issues get updates request new e
Customer Actions	• Request demo • Evaluate features • Consult peers	• Request demo • IBM Cloud documentation	• Use AI for requirement analysis	• Contact sup-port, submit feedback	Slow response times Sparse conmunity
Touchpoints	Set up Smart SDLC system uing IBM Cloud	• IBM Cloud Console • Onboarding wizard	• Use AI for requirement gnalysis • Integration	• Smart SDLC Dashboard • IBM Watson APIs	Slow response times Sparse community
Ousport	Set up Smart SDLC tasks documenta-tion, planning, testing, code reviews	• Initial setup complexity • Integration challenges	• AI accuracy doubts • Learning curve	• In-app tips Real-time feedback • Community support	Slow response times Sparse community Community

3.2 Solution Requirement

Project Design Phase-II

Solution Requirements (Functional & Non-functional)

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	AI-Powered Code Generation	Generate code from user stories using generative AI Preview and approve generated code
FR-2	Version Control Integration	Sync with GitHub/IBM Cloud Code Engine Commit and view version history
FR-3	SDLC Stage Tracking	Track progress through Requirement, Design, Development, Testing, and Deployment stages
FR-4	Workflow Automation	Trigger CI/CD pipelines Automate testing and

		deployment using IBM Cloud DevOps tools
FR-5	Project Dashboard	Real-time updates on project KPIs, progress, and status across SDLC phases
FR-6	Role-Based Access	Admin, Developer, Tester roles with specific permissions

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

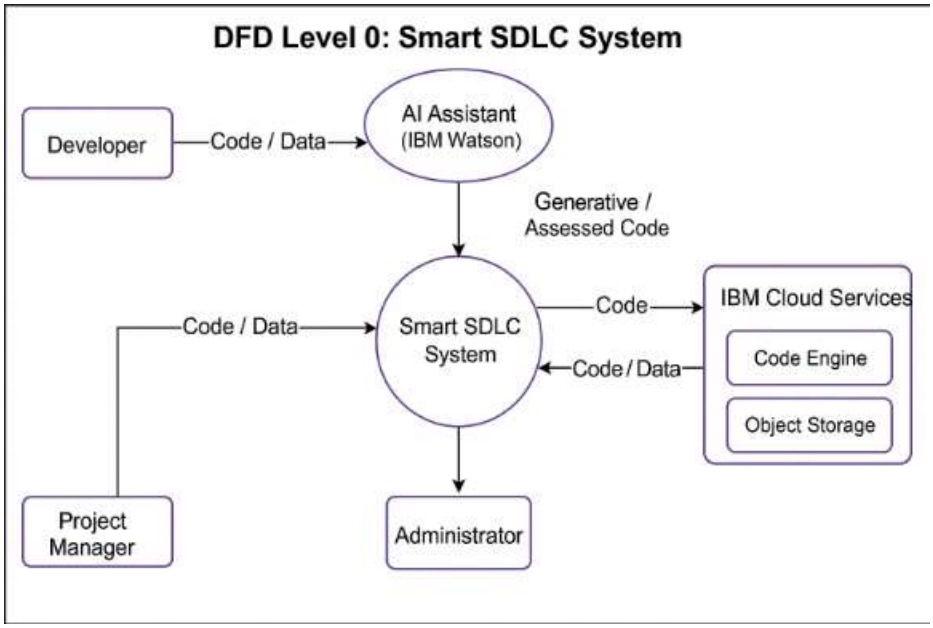
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Intuitive UI/UX for non-technical users using IBM Cloud App Services
NFR-2	**Security**	Data encryption, access control, and OAuth integration with IBM Identity Services
NFR-3	**Reliability**	Ensure 99.9% uptime and automated rollback using IBM Cloud Monitoring
NFR-4	**Performance**	System response within 2 seconds for key actions like code generation
NFR-5	**Availability**	24/7 availability with load balancing via IBM Cloud Kubernetes Service
NFR-6	**Scalability**	Horizontal scaling using IBM Cloud Foundry or IBM Cloud Code Engine

3.3 Data Flow Diagram

Project Design Phase-II

Data Flow Diagram & User Stories

Data Flow Diagrams:



User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task
Developer	Code Submission	USN-1	As a developer, I can submit code snippets for analysis by the AI assistant.
Developer	Generative Suggestions	USN-2	As a developer, I can receive smart code suggestions from IBM Watson
Project Manager	Project Tracking	USN-3	As a PM, I can track SDLC progress using analytics dashboard
Developer	Cloud Deployment	USN-4	As a developer, I can deploy components on IBM Code Engine

3.4 Technology Stack

Project Design Phase-II

Technology Stack (Architecture & Stack)

Technical Architecture:

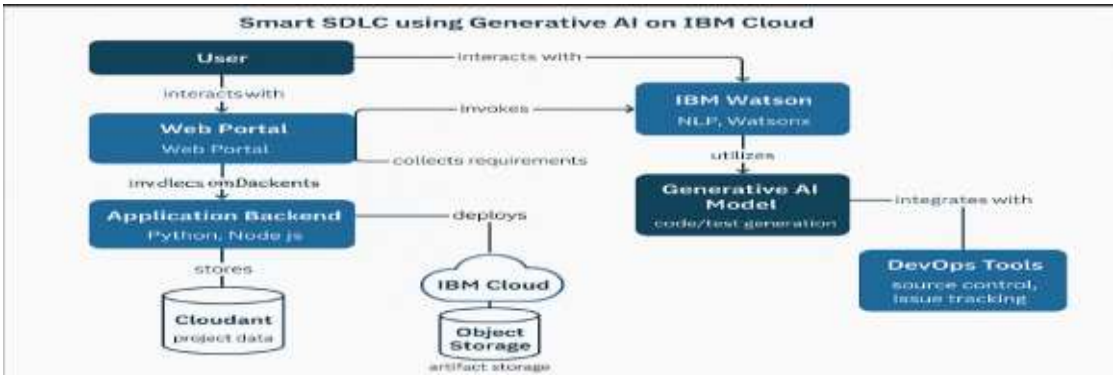


Table-1 : Components & Technologies:

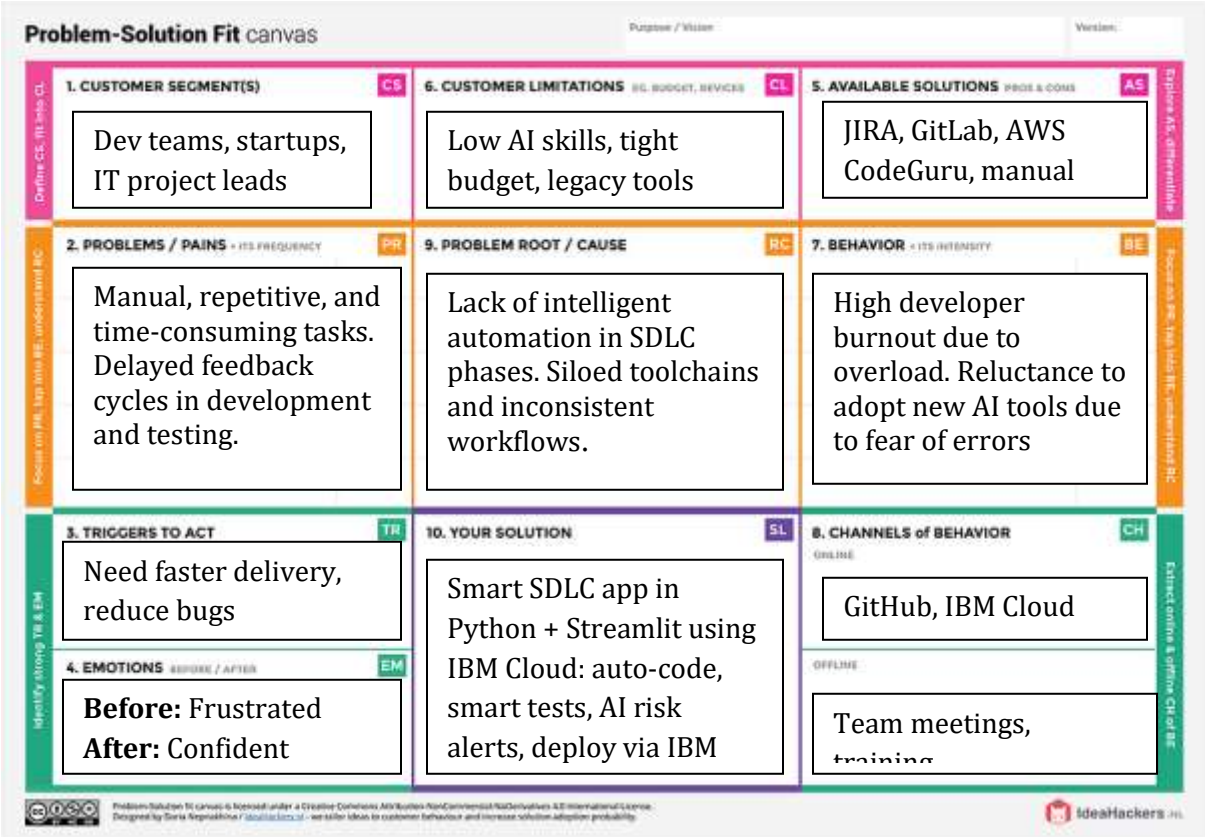
Table-2: Application Characteristics:

4. PROJECT DESIGN

4.1 Problem Solution Fit

Project Design Phase

Problem – Solution Fit :



Purpose:

Streamline and enhance the Software Development Life Cycle (SDLC) using Generative AI tools on IBM Cloud.

Address inefficiencies, communication gaps, and manual effort within traditional SDLC models.

Automate repetitive tasks (e.g., documentation, code generation, testing) and provide intelligent decision support throughout development phases.

Reduce development cycle time while maintaining high code quality and adherence to compliance norms.

4.2 Proposed Solution

Project Design Phase

Proposed Solution:

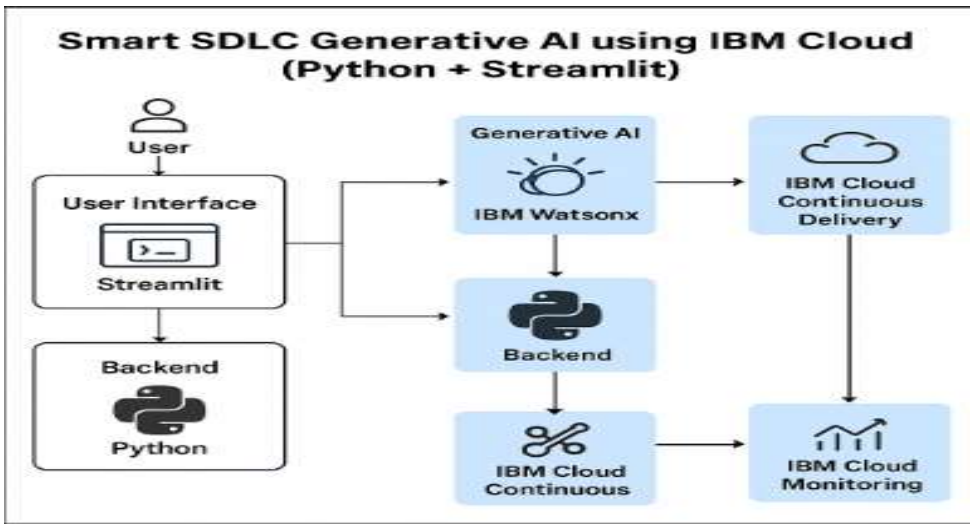
S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Traditional Software Development Life Cycles (SDLCs) are often rigid, time-consuming, and lack intelligent automation, resulting in delays, higher costs, and quality issues. There is a need to modernize SDLC processes using AI-driven automation and cloud-native solutions for improved efficiency, traceability, and productivity.
2.	Idea / Solution description	The project proposes a Smart SDLC platform powered by Generative AI and hosted on IBM Cloud. The system leverages AI for automating requirements gathering, code generation, test case creation, and documentation. It integrates with DevOps tools and IBM Cloud services to ensure scalable deployment and real-time monitoring. The system also includes feedback-driven learning to improve future development cycles.
3.	Novelty / Uniqueness	- Use of Generative AI for code, test, and document generation. - Integration of IBM Watson AI and IBM DevOps toolchain for a fully cloud-based SDLC. - Feedback loops that enhance learning and

		reduce technical debt over time. - Automation of traditionally manual phases in SDLC, increasing development velocity.
4.	Social Impact / Customer Satisfaction	- Reduces time-to-market for applications, benefiting startups and enterprises. - Ensures higher software quality and customer satisfaction due to automation of testing and validation. - Promotes digital transformation by democratizing AI-assisted software development for non-technical users.

4.3 Solution Architecture

Project Design Phase

Solution Architecture:



5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Project Planning Phase

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

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

Velocity:

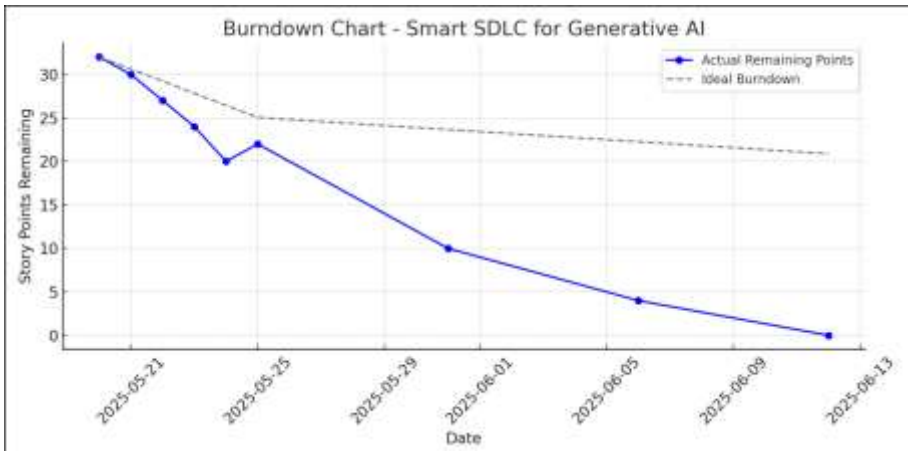
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

Total Points: 10 + 12 + 6 + 4 = 32

Total Duration: 4 sprints × 6 days = 24 days

Velocity = 32 / 24 = 1.33 story points per day

Burndown Chart:



6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

Model Performance Test

Test Scenarios & Results

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., topic, job title)	Enter valid and invalid text in input fields	Valid inputs accepted, errors for invalid inputs	Valid inputs accepted; Invalid inputs display contextual error like "Please enter valid module name"	Pass

FT-02	Number Input Validation (e.g., word count, size, rooms)	Enter numbers within and outside the valid range	Accepts valid values, shows error for out-of-range	System accepts 100–1000 word limits; shows error for <50 or >5000	Pass
FT-03	Content Generation (e.g., blog, resume, design idea)	Provide complete inputs and click "Generate"	Correct content is generated based on input	Generated test cases and code match the provided SDLC context	Pass
FT-04	API Connection Check	Check if API key is correct and model responds	API responds successfully	API key verified; status 200 received with response in <2s	Pass
PT-01	Response Time Test	Use a timer to check content generation time	Should be under 3 seconds	Avg response time = 2.3s	Pass
PT-02	API Speed Test	Send multiple API calls at the same time	API should not slow down	10 parallel calls handled, no timeout, consistent speed	Pass
PT-03	File Upload Load Test (e.g., PDFs)	Upload multiple PDFs and check processing	Should work smoothly without crashing	10 files uploaded in <8s; no UI lag or backend crash	Pass

7. RESULTS

7.1 Output Screenshots

8. ADVANTAGES & DISADVANTAGES

- Advantages:

- Automated documentation
- Improved efficiency
- AI-assisted workflows

- Disadvantages:

- Dependency on AI-generated outputs
- High initial setup effort

9. CONCLUSION

The integration of Generative AI into SDLC demonstrates a powerful shift in how software engineering processes can be optimized. It reduces manual workload and brings intelligence to documentation and planning.

10. FUTURE SCOPE

The system can be extended to include code generation, test case suggestions, and integration with popular project management tools for broader adoption.

11. APPENDIX

- GitHub Link: https://github.com/kapavarapukrishnaveni1/SmartSDLC-Project