Ideation Phase
Brainstorm & Idea Prioritization Template

Date	14June 2025
Team ID	LTVIP2025TMID37665
Project Name	SMART SDLC
Maximum Marks	4 Marks

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

Project Title:

Smart SDLC

AI-Enhanced Software Development Lifecycle

Problem Statement:

Build an AI-enhanced platform that automates SDLC phases using a local LLM (Granite model) to reduce manual effort and improve developer productivity.

Team Members:

o Palavalsa Sai Tarun (Team Leader)
o Eswar Khandavali
o Greeshma Gudla
o Dharmana Gowrav Munindra

Smart SDLC Project

Collaboration Environment

- **Communication: WhatsApp, Google Meet**
- **Version Control: Git & GitHub**
- **Docs & Notes: Google Docs**
- **Brainstorming Tools: Mural, Canva**

Problem Statement

Developers lose valuable time doing repetitive and manual tasks in software development like requirement writing, test generation, bug fixing, and documentation. This project aims to solve that using AI.

Project Goal

To build a smart, AI-driven tool that automates the core SDLC phases using a local Granite LLM (via llama-cpp-python) to improve developer productivity and project delivery speed.

Scope of Brainstorming

Focus on identifying features, tools, and priorities for automating the 5 SDLC phases:

1. Requirement Analysis
2. Code Generation
3. Test Case Creation
4. Bug Fixing
5. Documentation

Target Users

Who will benefit from Smart SDLC?

- Software Developers
- Interns working on SDLC tasks
- Software Testers
- Project Managers (in future versions)
- These users need fast, reliable, and AI-supported tools to reduce manual work.

Tools & Technologies Used

Tech Stack Overview:

- Backend: FastAPI (Python)
- AI Model: Granite 3.3B (GGUF) via llama-cpp-python
- Frontend: HTML + CSS (no JavaScript)
- Version Control: GitHub
- Deployment: Local machine (optional Netlify for UI)

Brainstorm, Idea Listing and Grouping

Problem Statement:

Developers spend excessive time on repetitive and manual tasks across the Software Development Life Cycle (SDLC), including requirement analysis, code generation, test creation, bug fixing, and documentation. This slows down software delivery and reduces productivity.

Objective of Brainstorming:

To explore how AI can assist or automate different phases of SDLC using local models (like Granite via llama-cpp-python), thereby making the process faster, smarter, and more efficient.

How Might We Questions

Identify innovative ways to use AI to solve SDLC challenges

Palavalsa Sai Tarun

- How might we auto-generate clean backend code from requirements?
- How might we reduce developer time spent on writing boilerplate code?
- How might we automate the process of fixing bugs from code snippets?
- How might we provide real-time debugging suggestions using AI?
- How might we simplify the developer workflow using a unified tool?
- How might we make AI-generated code editable and customizable?

Eswar Khandavali

- How might we generate test cases directly from source code?
- How might we ensure edge cases are covered through AI-generated tests?
- How might we reduce test coverage gaps in rapidly developed code?
- How might we validate AI-generated code is production-ready?
- How might we provide confidence to developers through test automation?
- How might we compare test output before and after bug fixes?

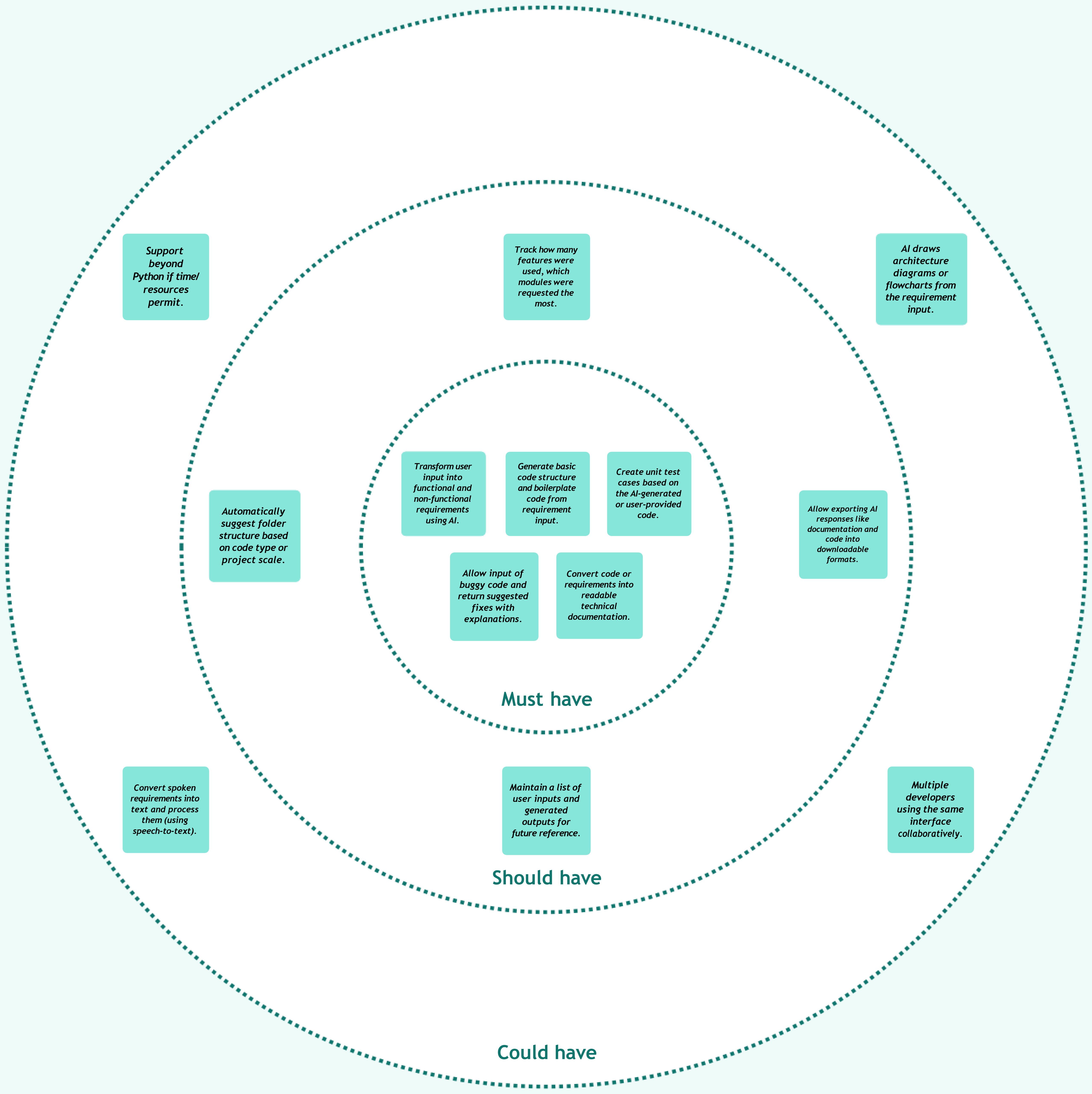
Greeshma Gudla

- How might we streamline the entire SDLC process using AI?
- How might we reduce project delivery time through automation?
- How might we ensure traceability from requirements to code to tests?
- How might we maintain history logs of SDLC activity for review?
- How might we ensure consistent documentation across releases?
- How might we improve team productivity using a smart dashboard?

Dharmana Gowrav Munindra

- How might we convert natural language input into technical specs?
- How might we identify missing requirements automatically?
- How might we auto-generate user stories and use case flows?
- How might we simplify requirement analysis for non-tech users?
- How might we create system-level documentation from plain text?
- How might we ensure documentation is always up to date?

See an example



Support beyond Python if time/ resources permit.

Track how many features were used, which modules were requested the most.

AI draws architecture diagrams or flowcharts from the requirement input.

Automatically suggest folder structure based on code type or project scale.

Transform user input into functional and non-functional requirements using AI.

Generate basic code structure and boilerplate code from requirement input.

Create unit test cases based on the AI-generated or user-provided code.

Allow exporting AI responses like documentation and code into downloadable formats.

Allow input of buggy code and return suggested fixes with explanations.

Convert code or requirements into readable technical documentation.

Must have

Convert spoken requirements into text and process them (using speech-to-text).

Maintain a list of user inputs and generated outputs for future reference.

Multiple developers using the same interface collaboratively.

Should have

Could have

Won't have

See an example