**Project Design Phase**

**Problem – Solution Fit Template**

|  |  |
| --- | --- |
| Date | 15 February 2025 |
| Team ID | LTVIP2025TMID32102 |
| Project Name | SmartzSDLC – AI-Enhanced Software Development Lifecycle |
| Maximum Marks | 2 Marks |

**Problem – Solution Fit Template:**

**Offline**: Whiteboard/design sessions, pair programming, team meetings.

• Relying on individual scripts or manual documentation.

• Low‑code/no‑code tools (e.g. Quixy) for parts of SDLC.

• Generic large language models (ChatGPT, GPT-4) for snippets.

• Limited budget for commercial code assistants.  
• No in-house AI/ML infrastructure or expertise.  
• Security concerns with third-party tools and code privacy.

• Software developers, QA engineers, CTOs, tech leads, and product managers in small to medium enterprises who spend extensive time on repetitive SDLC tasks.

• They prioritize productivity and consistency in project delivery.

• Requirement gathering is unclear, manual, and error‑prone.

• Boilerplate code, test case writing, documentation – all time-consuming.

• Bugs take too long to identify and fix.

• Explaining code to new team members is difficult, hurting onboarding.

• SDLC phases (requirements → design → code → test → docs) are disjointed, inefficient, and error‑prone due to manual workflows.  
• Inconsistent output quality due to human variance in writing and documentation.

• Frequent back-and-forth on Slack or email for requirement specs.  
• Hold standups and debug sessions for simple script clarifications.  
• Use templates to manually create user stories, code, tests, docs.

**Online**: Slack, GitHub issues, Jira, StackOverflow, internal wikis.

**SmartSDLC** – An AI‑powered end-to-end SDLC assistant that enables:

PDF upload to classify requirements → generates user stories automatically

AI code generation, bug-fixing, test cases, and human-readable documentation

Conversational AI assistant for real-time SDLC guidance

Built with IBM Granite 3.3B model via Hugging Face; fast, easy, and secure

• Tight deadlines or quality gates cause frustration.

• Frequent merging delays due to missing test cases or unclear requirements.

• Frustration from repetitive, manual tasks.  
• Burnout from poor onboarding and debugging cycles.