

Project Design Phase Problem – Solution Fit Template

| | |
|---------------|--|
| Date | 25 June 2025 |
| Team ID | LTVIP2025TMID38248 |
| Project Name | SmartSDLC - AI-Enhanced Software Development Lifecycle |
| Maximum Marks | 2 Marks |

Problem – Solution Fit Template:

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 touch-points 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 in order to improve it for your target group.

Template:

| Problem-Solution fit canvas 2.0 | | | Purpose / Vision | |
|---------------------------------|---|---|--|--|
| Define CS, fit into CC | <div>1. CUSTOMER SEGMENT(S)</div> <div><ul style="list-style-type: none">Software Development Professionals and Teams.Specifically: Project Managers, Software Developers (Frontend/Backend), QA Engineers, DevOps Engineers, New Team Members, Junior Developers, Team Leads.</div> | <div>6. CUSTOMER CONSTRAINTS</div> <div><ul style="list-style-type: none">Time & Deadlines: Pressure to deliver quickly limits time for manual, thorough work.Budget: Constraints on investing in multiple, expensive specialized tools or custom integrations.Existing Toolchain Lock-in: Difficulty integrating a new solution with deeply embedded legacy tools/processes.<p>Security Concerns: Hesitation to adopt cloud-based or AI tools due to data privacy or intellectual property worries.</p><p>Technical Complexity: Certain problems (e.g., highly complex code refactoring) are difficult to automate reliably.</p></div> | <div>5. AVAILABLE SOLUTIONS</div> <div><p>Manual Processes:</p><ul style="list-style-type: none">Pros: Full human control.Cons: Time-consuming, error-prone, inconsistent, not scalable, relies on individual expertise.<p>Traditional IDEs/Debuggers:</p><ul style="list-style-type: none">Pros: Essential for development.Cons: Lack intelligent automation for code generation, bug fixing suggestions, or deep code summarization.</div> | Explore AS, differentiate |
| | <div>2. JOBS-TO-BE-DONE / PROBLEMS</div> <div><ol style="list-style-type: none">Requirement Ambiguity:Inefficient Code Development & DebuggingManual & Incomplete Testing <div>4. Difficulty Understanding Complex/Legacy Code:</div><div><ol style="list-style-type: none">Lack of Immediate SDLC Knowledge/Support:Fragmented Tooling & Inefficient Workflow</div></div> | <div>9. PROBLEM ROOT CAUSE</div> <div><ul style="list-style-type: none">Increasing Software Complexity: Modern applications are inherently complex, making manual processes inefficient and error-prone.Rapid Development Cycles: Market demands force faster delivery, often at the expense of thoroughness or documentation.Human Cognitive Limits: The sheer volume of information (requirements, code, tests) exceeds human capacity for manual processing and correlation.Siloed Knowledge & Tooling: Lack of integration between different SDLC phases and tools leads to inefficiencies, inconsistencies, and knowledge gaps.</div> | <div>7. BEHAVIOUR</div> <div><ul style="list-style-type: none">Manually analyzes requirement documents, often highlighting and annotating.Writes code line-by-line, often from scratch or by adapting existing boilerplate.Uses print statements, debuggers, and logs to trace and identify bugs.Manually drafts test cases based on requirements or code logic.Spends time tracing code flow, reading function signatures, and dissecting modules to understand functionality.Interrupts colleagues for quick questions or code explanations.Switches between multiple applications: IDE, browser (for docs/search), project management tool, communication app.Attends numerous meetings for status updates, planning, and clarification.</div> | Focus on JBP, tap into BE, understand PC |
| Identify strong TR & EM | <div>3. TRIGGERS</div> <div><ul style="list-style-type: none">Project Deadlines & Scope Changes: Pressure to deliver quickly or adjust to new requirements.Bug Reports: Identification of defects in development or production.Onboarding New Team Members: Need for rapid code comprehension and SDLC knowledge transfer.</div> | <div>10. YOUR SOLUTION</div> <div><ul style="list-style-type: none">Intelligent Requirement Classification: Automates requirement analysis.AI Code Generator: Accelerates code writing.Automated Bug Fixer: Expedites bug resolution.Smart Test Case Generator: Enhances test coverage.</div> | <div>8. CHANNELS of BEHAVIOUR</div> <div><ul style="list-style-type: none">Integrated Development Environments (IDEs) (VS Code, IntelliJ, PyCharm)Version Control Systems (Git, GitHub, GitLab, Bitbucket)Project Management / Tooling Systems (Jira, Asana, Trello)Team Communication Platforms (Slack, Microsoft Teams) <ul style="list-style-type: none">Internal Wiki/Documentation Portals (Confluence, SharePoint)Online Development Forums and Q&A sites (Stack Overflow)Email and Calendar for meetings.Company-specific internal tools and dashboards.</div> | Extract online & offline CH of BE |
| | <div>4. EMOTIONS: BEFORE / AFTER</div> <div><ul style="list-style-type: none">Before (Facing the Problem): Frustrated, Anxious, Overwhelmed, Inefficient, Pressured, Concerned, Stuck, Limited, Burned out, Lost, Insecure.After (Using SmartSDLC Solution): Confident, Efficient, Empowered, Productive, Relieved, In Control, Streamlined, Knowledgeable, Assured, Delighted.</div> | | | |