Product Charter: Sarathi (Prompt Builder)

This charter defines the scope and strategy for the initial version of the Prompt Builder application. Our primary objective is to solve the core problem of prompt disorganization and accessibility for non-technical teams. This product will be a standalone web application, designed with future API integrations in mind.

1. User Intent: What do users want?

- **As a non-technical user,** I want a simple, structured place to write and save my prompts so I don't lose my work and can track what works.
- **As a team member,** I want to stop asking my colleagues for "that one prompt for summarizing reports" and instead easily find, use, and improve upon a proven, high-quality version from a shared team space.
- **As** a product **owner,** I want to create a central, living repository of best-practice prompts to ensure my team uses our AI tools consistently and effectively, with quality improving over time.

2. Primary Goal (The Core Problem)

To deliver an intelligent and dynamic ecosystem for prompts that guides users in crafting effective prompts and allows teams to seamlessly save, share, discover, and continuously improve them through collaboration.

3. Sub-goals (How We'll Achieve the Primary Goal)

- **Guide Prompt Creation:** Introduce an AI-assisted workflow that helps users transform a business problem into a well-structured and effective prompt.
- **Foster a Collaborative Ecosystem:** Introduce features like version history, ratings, and upvotes to enable teams to collectively refine and identify the best prompts.
- **Establish a Standard:** Implement a clean and mandatory structure for every prompt to ensure all shared prompts are useful and easy to understand.
- **Create a "Single Source of Truth":** Eliminate prompt loss and duplication by providing one central place for teams to store their collective knowledge.
- **Enable Effortless Discovery:** Build proactive and manual search/filtering systems so users can find relevant prompts in seconds.
- **Foster Psychological Safety:** Clearly separate a user's Private Library (a safe space for drafting and experimentation) from the Shared Team Library (a place for polished, useful contributions).

4. Underspecification (Intentionally Out of Scope for the Initial Version)

- **Formal Evaluation:** We will not implement an automated system for running prompts and judging their effectiveness. The "quality" of a prompt is based on implicit team trust and user ratings.
- **Full API Implementation:** While the system will be *designed* for future API abstraction, a publicly consumable API for integration with other tools will not be built in the initial

version.

5. Optimization (Strategic Trade-offs for the Initial Version)

- **Collaborative Intelligence over Initial Simplicity:** We are prioritizing the creation of a rich, collaborative environment from the start. We accept the added complexity of features like versioning and ratings as a necessary trade-off to foster a culture of continuous improvement and collective ownership.
- **Structure over Unstructured Flexibility:** We will enforce the completion of key metadata fields when creating a prompt. This slight friction during creation is a deliberate trade-off to ensure the high value and searchability of the shared library later.

6. Key User Journeys

User Journey 1 — Create and Add New Prompt in Library

- **Step 1: User Login:** User logs into the system or signs up if it's their first visit.
- **Step** 2: Request New Generate **Prompt:** The user provides the initial context for the prompt, including the use case details and other context via Various Tags i.e. project, user persona, desired output format, and purpose (e.g., Research or Marketing)
- **Step 3: Generate Prompt and Validate Before Save:** The system generates a draft which user can validates the generated response, and edits the prompt as needed. They must define the prompt's type as Private or Public, chooses a target LLM and mandatory tags for categorization before save.
- **Step 4: Save:** The system validates that the title is unique and all required fields are complete before saving the prompt.

User Journey 2 — System Recommendations for Best Prompt

- **Context:** A user is unaware of a relevant best prompt and the system provides recommendations.
- **Step 1: User Login:** User logs into the system.
- **Step 2: Type Prompt:** User begins typing a prompt into the input field.
- **Step 3: System Suggests Relevant Prompts:** The system recommends existing prompts based on relevance tags, likes/dislikes, and usage count.
- **Step 4: User Selection:** The user can either continue typing their own prompt or choose one from the recommendation list.
- **Step 5: Insert Chosen Prompt:** If a recommendation is selected, the system inserts the entire prompt into the input window.
- **Step 6: Edit & Execute:** The user may edit the inserted prompt before executing it.

User Journey 3 — Use an Existing Prompt from the Library

- **Context:** A user prefers to use a known prompt directly from the library.
- **Step 1: User Login:** User logs into the system.
- **Step 2: Access Library:** User clicks the "Library" button to browse existing prompts.
- **Step** 3: System Displays Prompt **Library:** The system shows a list of available Private and Public prompts, which can be searched and filtered. The list is sorted by selected tags,

likes/dislikes, and popularity, with Private prompts prioritized in case of an equal match.
- **Step 4: User Selection:** The user chooses a prompt from the library list.
- **Step 5: Insert Prompt:** The system inserts the entire selected prompt into the chat window.
- **Step 6: Edit & Execute:** The user can edit the prompt before executing it.

User Journey 4 — Iterate and Improve a Shared Prompt

- **Context:** A user finds a valuable prompt in the shared library but sees an opportunity to make it even better.
- **Step 1: Discover and Evaluate:** The user finds and uses a public prompt from the library (as in Journey 3). They note that the output is good, but could be improved (e.g., by requesting a specific format like JSON).
- **Step 2: Initiate New Version:** The user navigates to the prompt's detail page and clicks an "Improve" or "Create New Version" button.
- **Step 3: Edit and Document Changes:** The system opens the prompt editor, pre-populated with the content from the previous version. The user makes their desired changes to the prompt text. They add a brief "Version Note" explaining the improvement (e.g., "Modified to output in valid JSON format").
- **Step 4: Save as New Version:** The user saves their work. The prompt is now updated, and the previous version is accessible in the "Version History" tab, along with the new version note.
- **Step 5: Community Validation:** Over time, other team members begin to use the new, improved version. It accumulates more upvotes than the older version, naturally becoming the new standard for the team.

User Journey 5 — Review prompt performance and Manage Contributions

- **Context:** A user who has previously shared prompts wants to track their impact
- **Step 1: Navigate to Contributions:** The user logs in and navigates to their personal dashboard or a "My Contributions" page.
- **Step** 2: Review **Prompt Performance:** The system displays a list of prompts the user has created, showing key metrics like usage count and the number of upvotes for each. This allows the user to see which of their contributions are most valuable to the team.

User Journey 6 — Provide Feedback on a Shared Prompt

- **Context:** A user wants to provide direct feedback on a prompt they found in the shared library.
- **Step 1: Find a Colleague's Prompt:** The user finds a useful prompt in the shared library that was created by a team member.
- **Step 2: Upvote:** After using the prompt, the user navigates back to its detail page. They click the "upvote" button to endorse its quality.
- **Step** 3: See **Immediate Impact:** The system instantly updates the prompt's upvote count. This feedback becomes visible to all other users, helping the best prompts become more discoverable.

User Journey 7 — Archive an Outdated Shared Prompt

- **Context:** A prompt author realizes their shared prompt is no longer relevant, either because the project it was for has ended or a LLM version it best worked do not exists anymore.
- **Step 1: Locate the Prompt:** The user navigates to their "My Contributions" page and finds the shared prompt they wish to archive (Inactive).
- **Step 2: Initiate Archival:** On the prompt's detail page, the user clicks an "Archive" button.
- **Step 3: System Confirmation:** The system displays a confirmation message, explaining that archiving will remove the prompt from active search results and recommendations but will not delete it, preserving its history.
- **Step 4: Confirm Action:** The user confirms they want to proceed.
- **Step 5: Prompt is Archived:** The system flags the prompt as "Archived." It no longer appears in the main library view but can still be found by its author in an "Archived" section of their dashboard. This keeps the main library clean and relevant.

User Journey 8 — Manage a Retired LLM Version

- **Context:** An underlying LLM version (e.g., Gemini 2.5 Pro) is being discontinued by its provider, and the system needs to handle this gracefully.
- **Step 1: Admin Configures Fallback:** A system administrator navigates to an admin panel. They pre-emptively configure a fallback mapping (e.g., map all "Gemini 2.5 Pro" prompts to "Gemini 2.6 Pro-latest").
- **Step 2: Admin Discontinues LLM:** The administrator officially marks the "Gemini 2.5 Pro" version as discontinued in the system.
- **Step 3: System Notifies Owners:** The system automatically triggers a notification (e.g., via email) and on ‘My Contributions’ page to all users who own prompts that were using the now-retired LLM. The notification informs them of the change and the need for re-validation.
- **Step 4: Prompts Marked for Review:** Simultaneously, all affected prompts are flagged with a status like "Needs Validation" and are temporarily removed from being considered "production ready." They may be deprioritized or removed in search results as per Admin settings.
- **Step** 5: **Owner Re-validates:** The prompt owner receives the notification, opens their affected prompt, and Re-tests it with the new fallback LLM. They make any necessary edits as a revalidation process.