Prompt Library - Product Requirements Document

Executive Summary

Prompt Library is a purpose-built, pure prompt management system designed for teams building AI-powered applications. The platform serves as a centralized repository for organizing, versioning, and managing prompts across multiple LLM models. Unlike general AI development platforms, Prompt Library focuses exclusively on prompt lifecycle management—creation, organization, versioning, tagging, status tracking, and model usage tracking.

The system enables teams to eliminate prompt chaos, maintain consistency across projects, and streamline prompt deployment while preserving full traceability of changes over time.

Product Vision

Vision Statement: Enable teams to manage their prompt inventory as a first-class asset with the same rigor and discipline applied to software code.

Core Philosophy: Prompts are not code, but they require code-like discipline. Prompt Library brings order to prompt management through structured organization, versioning, and governance.

1. Product Overview

1.1 What is Prompt Library?

Prompt Library is a **pure prompt management system** —a standalone platform focused exclusively on the creation, organization, versioning, and deployment of prompts. The system does not include:

- LLM inference capabilities
- Playground for testing (testing orchestrated via API integrations)
- Monitoring/analytics
- AI chat interfaces
- Code editors or development tools

1.2 Core Purpose

Prompt Library solves three critical problems:

1. **Chaos Prevention** – Eliminates scattered, unversioned prompts across spreadsheets, documents, and Slack messages
2. **Consistency** – Provides a single source of truth for prompts across teams and projects

3. **Governance** – Enables version control, status management, and audit trails for compliance

2. Key Features & Requirements

2.1 Prompt Organization

Tags System

- Every prompt can have multiple tags for flexible categorization
- Tag examples: `customer-service`, `code-generation`, `content-writing`, `data-analysis`, `experiment`
- Tags are not hierarchical—flat structure for maximum flexibility
- Predefined tag suggestions with custom tag creation allowed
- Search and filter by tags

Purpose Classification

- Each prompt must define its **purpose** (what the prompt is designed to do)
- Purpose examples:
  - `Email Reply Generation`
  - `Code Review Analysis`
  - `Customer Sentiment Analysis`
  - `Blog Post Outline`
  - `Data Transformation`
- Structured dropdown with ability to add custom purposes
- Enables grouping of related prompts by business function

Metadata Fields

- Prompt Name (required, descriptive)
- Description (optional, full context and use cases)
- Author (automatically captured)
- Created Date (automatic)
- Last Modified Date (automatic)
- Creator/Owner information

2.2 Prompt Content Storage

Prompt Structure

- Full prompt text storage (system prompt + user prompt optional separation)
- Support for template variables (e.g., `{{user_input}}`, `{{context}}`)
- Optional: System prompt field (separate from main prompt)
- Raw text storage without interpretation

Prompt Composition (Optional Enhancement)

- Support for modular prompt structure: Context → Task → Instructions → Examples → Primer
- Allows composition of complex prompts from building blocks
- Each section independently versionable

2.3 Version Control (Semantic Versioning)

Version Format: `X.Y.Z` (Major.Minor.Patch)

Versioning Rules

- **Major (X)**: Structural or fundamental prompt changes that significantly alter behavior
- **Minor (Y)**: New features, enhanced context, additional parameters
- **Patch (Z)**: Bug fixes, typo corrections, minor tweaks

Version Management

- Every change to a prompt automatically creates a new version
- Version history preserved indefinitely
- Ability to view full change history with timestamps
- Side-by-side comparison of any two versions
- One-click rollback to previous versions
- Version descriptions/changelogs (why the change was made)

Version Metadata

- Version number
- Date created
- Author of the change
- Change description
- Status at time of creation

2.4 Status Management

Prompt Lifecycle States

- **Draft** - Prompt is in development, not yet ready for use
- **In Review** - Prompt submitted for team review/approval
- **Active** - Approved and live, ready for production use
- **Archived** - Deprecated or superseded, kept for historical reference
- **Deprecated** - No longer recommended, replaced by newer version
- **Testing** - Under evaluation with real models/data

Status Transitions

- Draft → In Review → Active
- Active → Deprecated → Archived
- Active → Testing (for iteration)
- Testing → Active (after validation)

Status Rules

- Only Active prompts can be deployed to production
- Status changes require audit trail
- Comments/notes on status transitions

2.5 Usage (LLM/Model Tracking)

Model Association
Each prompt explicitly tracks which LLM models and providers it's designed for:

Supported Model Families

- OpenAI: `gpt-4`, `gpt-4-turbo`, `gpt-3.5-turbo`
- Anthropic: `claude-opus`, `claude-sonnet`, `claude-haiku`
- Google: `gemini-pro`, `gemini-pro-vision`
- Mistral: `mistral-large`, `mistral-medium`
- Open Source: `llama-2`, `mistral-7b`, etc.
- Meta: `llama-3`, `llama-3.1`
- Custom/Private models

Usage Configuration Per Version

- Applicable models (multi-select)
- Suggested temperature/sampling parameters
- Token limits/constraints
- Special notes (e.g., "requires function calling", "supports vision")
- Model-specific optimizations

Usage Metadata

- Last tested date
- Test results summary
- Known limitations per model

2.6 Additional Core Features

Search & Discovery

- Full-text search across prompt names, descriptions, content
- Filter by: Tags, Purpose, Status, Models, Author, Date Range
- Saved search filters
- Advanced search syntax

Collaboration Features

- Prompt ownership/assignment
- Comments on versions
- Review workflows (optional approval before status change)
- Activity log showing who changed what and when

API & Integration

- REST API to retrieve prompts by ID or query
- SDK/client libraries (Python, Node.js, etc.)
- Ability to programmatically fetch active versions
- Webhook support for integrations

Export & Backup

- Export individual prompts (JSON, Markdown, Plain Text)
- Bulk export of prompt library
- Version history export

Import & Migration

- Bulk import prompts from CSV/JSON
- Migration tools for existing prompt repositories
- Duplicate detection

3. User Personas & Use Cases

3.1 Primary Users

Prompt Engineers

- Create and iterate on prompts
- Version and test different variations
- Need rapid iteration cycles
- Key Need: Version control + model comparison

Team Leads / Prompt Managers

- Oversee prompt library health
- Approve prompts for production
- Maintain standards and consistency
- Key Need: Status workflows + governance

Application Developers

- Integrate prompts into applications
- Fetch prompts via API
- Need reliability and version pinning
- Key Need: API stability + version guarantees

Product Managers

- Understand what prompts exist
- Track prompt purposes and coverage
- Monitor usage patterns
- Key Need: Discovery + analytics

3.2 Use Cases

1. **Centralized Prompt Repository**: Team stores all 50+ prompts in Prompt Library instead of scattered documents
2. **A/B Testing Variants**: Create v1.0 and v1.1 of the same prompt, test both, track performance differences
3. **Model Migration**: When a new model releases (e.g., GPT-4 Turbo), easily tag prompts as compatible and version them
4. **Compliance Auditing**: View complete version history showing who changed what prompt and when

5. **Onboarding**: New team members discover available prompts via centralized library with clear purposes
6. **Prompt Reuse**: Teams reuse proven prompts across different projects
7. **Status Management**: Track which prompts are ready for production vs experimental
8. **Cross-Team Consistency**: Marketing and Support teams use the same prompt templates

4. Technical Requirements

4.1 Data Model

Prompt Entity

Field	Type	Notes
id	UUID	Unique identifier
name	String	Required, descriptive name
description	String	Optional, full context
purpose	String	Required, from predefined or custom
tags	Array[String]	Multiple tags, searchable
status	Enum	Draft | In Review | Active | Testing | Deprecated | Archived
owner	String	User ID of owner
created_at	DateTime	Automatic
updated_at	DateTime	Automatic
current_version	String	Reference to active version

Table 1: Prompt Entity Structure

PromptVersion Entity

Field	Type	Notes
id	UUID	Unique version ID
prompt_id	UUID	Reference to parent prompt
version_number	String	Format: X.Y.Z
content	Text	Full prompt text
system_prompt	Text	Optional, separate system prompt
status	Enum	Status at version creation
author	String	User ID who created version
created_at	DateTime	Automatic
change_description	String	Why this version was created
models	Array[String]	Compatible models
model_config	Object	Temperature, max_tokens, etc.
previous_version_id	UUID	Link to prior version

Table 2: PromptVersion Entity Structure

Tag & Purpose Entities

Field	Type	Notes
Tag Entity		
id	UUID	Unique tag ID
name	String	Tag text
usage_count	Integer	How many prompts use this tag
created_at	DateTime	Automatic
Purpose Entity		
id	UUID	Unique purpose ID
name	String	Purpose text
description	String	What this purpose means
usage_count	Integer	How many prompts

Table 3: Tag and Purpose Entities

4.2 API Specification

Core Endpoints

- **GET /api/prompts** - List all prompts with filters
- **POST /api/prompts** - Create new prompt
- **GET /api/prompts/{id}** - Retrieve prompt latest version
- **PUT /api/prompts/{id}** - Update prompt (creates new version)
- **DELETE /api/prompts/{id}** - Soft delete (archive)

- **GET /api/prompts/{id}/versions** - List all versions
- **GET /api/prompts/{id}/versions/{version}** - Get specific version
- **POST /api/prompts/{id}/versions/{version}/rollback** - Restore version
- **GET /api/tags** - List all tags
- **GET /api/purposes** - List all purposes
- **POST /api/prompts/{id}/status** - Change prompt status

Query Parameters

- `filter[tags]` - Filter by tags (comma-separated)
- `filter[purpose]` - Filter by purpose
- `filter[status]` - Filter by status
- `filter[models]` - Filter by compatible models
- `sort` - Sort by created_at, updated_at, name
- `search` - Full-text search

Response Format (JSON)

Success Response	HTTP 200/201
Error Response	HTTP 400/401/403/404/500 with error details

Table 4: Response Format

4.3 Architecture Requirements

Database

- PostgreSQL with JSONB support for flexible metadata
- Full-text search indexing
- Version history retained indefinitely
- Audit logging of all changes

Authentication & Authorization

- API key authentication for programmatic access
- User authentication (OAuth 2.0 or similar) for UI
- Role-based access control (Owner, Editor, Viewer)
- Team-based permissions

Infrastructure

- Scalable backend (Node.js, Python, or similar)
- CDN for static assets if applicable
- High availability setup
- Backup and disaster recovery

5. Non-Functional Requirements

5.1 Performance

- API response time: < 200ms for standard queries
- Search across 1000+ prompts: < 500ms
- Version history retrieval: < 100ms

5.2 Scalability

- Support 10,000+ prompts
- Support 100+ team members concurrently
- Version history without performance degradation

5.3 Security

- HTTPS/TLS encryption in transit
- Encrypted storage at rest (optional)
- API rate limiting
- GDPR compliance for user data
- Audit logging of all operations

5.4 Reliability

- 99.9% uptime SLA
- Automated backups (daily minimum)
- Disaster recovery plan with RTO < 4 hours

6. User Interface / Experience

6.1 Main Views

Prompt Library Dashboard

- Grid or list view of all prompts
- Quick filters (Status, Purpose, Tags, Models)
- Search bar
- "Create New Prompt" button
- Bulk actions

Prompt Detail View

- Prompt name, description, metadata
- Current status with change history
- Tags and purpose
- Associated models and configuration
- Current version content
- Version history timeline
- Comments/activity log

Version Comparison View

- Side-by-side comparison of any two versions

- Highlighted differences
- Metadata comparison (status, author, date)
- Ability to restore older version

Version History View

- Timeline of all versions
- Version number, author, date
- Change description
- Status at each version
- Quick access to rollback

Prompt Editor

- Clean text editor for prompt content
- Optional system prompt field
- Template variable highlighting
- Model selector (multi-select)
- Model configuration inputs
- Version description input
- Save as new version with changelog

6.2 Search & Discovery

- Full-text search with autocomplete
- Advanced filters (Tag, Purpose, Status, Models, Date Range)
- Saved searches
- Recently viewed prompts
- Popular prompts by team

6.3 Collaboration

- Comments on versions
- @mentions for team members
- Activity feed
- Approval workflow (if enabled)

7. Integration Points

7.1 External Integrations

LLM Testing Tools

- Integration with prompt testing platforms (PromptLayer, Helicone, etc.)
- Export prompts to testing tools
- Import test results back

Version Control

- Optional Git integration for version control alongside code
- Ability to commit prompts to Git repositories

CI/CD Pipelines

- Webhook support for prompt updates
- Trigger evaluations on version change
- Status updates based on test results

Team Communication

- Slack notifications for status changes
- Email notifications for reviews

7.2 SDK / Client Libraries

- Python SDK for accessing prompts
- JavaScript/Node.js SDK
- Documentation for building additional SDKs
- Example integrations

8. Success Metrics

8.1 Adoption Metrics

- Number of active users
- Number of prompts created/managed
- API call volume
- Monthly active teams

8.2 Quality Metrics

- Average time to approve prompt versions
- Number of prompt rollbacks
- Version history completeness
- Search effectiveness

8.3 Business Metrics

- User satisfaction (NPS)
- Feature usage distribution
- Integration adoption rate
- Enterprise customer success rate

9. Roadmap & Phasing

Phase 1: MVP (Weeks 1-8)

- Core prompt CRUD operations
- Semantic versioning (X.Y.Z)
- Status management (Draft, Active, Archived)
- Tags and Purpose classification
- Basic search and filtering
- REST API with basic endpoints
- Web UI dashboard

Phase 2: Collaboration (Weeks 9-12)

- Version comparison view
- Comments and activity log
- Status approval workflow
- Team-based permissions
- Webhook support

Phase 3: Model Management (Weeks 13-16)

- Model association and configuration
- Model-specific parameter tracking
- Compatibility matrix
- Model migration workflows

Phase 4: Advanced Features (Weeks 17+)

- Prompt composition (modular sections)
- Advanced search and filtering
- Performance analytics
- Integration marketplace
- Mobile app
- SDKs (Python, JavaScript)

10. Out of Scope

Explicitly NOT included:

- LLM inference or execution
- Chat interface or playground
- Monitoring and analytics
- Model training or fine-tuning
- Prompt optimization/auto-suggestion
- Security scanning or guardrails
- Cost tracking or billing

These features may be addressed through integrations but are not core to Prompt Library.

11. Success Criteria

The project is successful when:

1. ✓ Teams can store 100% of their prompts in Prompt Library (no scattered documents)
2. ✓ Every prompt change is tracked with full version history
3. ✓ Prompts can be tagged, filtered, and discovered within seconds
4. ✓ Status management prevents accidental production deployments
5. ✓ Developers can programmatically fetch prompts via API
6. ✓ Teams report improved consistency and reduced prompt duplication
7. ✓ Onboarding of new team members is 50% faster due to centralized discovery

12. Assumptions & Dependencies

Assumptions

- Teams have 10-100+ prompts requiring management
- Multiple models/providers are in use across projects
- Version control discipline is valued
- Prompts are text-based (no multimodal content initially)

Dependencies

- Database infrastructure (PostgreSQL or similar)
- Authentication system
- Optional: Integration platforms (Slack, GitHub, etc.)

Document Version: 1.0
Last Updated: January 7, 2026
Status: Ready for Development