Prompt Management System - Low Level Design Document

|  |  |
| --- | --- |
| Field Details | |
| **Document Title** | Prompt Management System - Low Level Design |
| **Version** | 1.0 |
| **Date** | January 2025 |
| **Status** | Draft |
| **System** | Prompt Management API |
| C C | |

# System Overview

## Purpose

The Prompt Management System is a API service designed to manage AI prompts with versioning, real-time collaboration, and soft deletion capabilities. It provides a centralized platform for teams to create, version, and manage prompts used in AI applications.

## Scope

This document covers the low-level implementation details of the Prompt Management System, including:

 API endpoint specifications  Database schema design

 Error handling mechanisms  Security considerations

## Technology Stack

 **Backend Framework:** FastAPI (Python)

 **Database:** Google Cloud Firestore (NoSQL)

 **Authentication:** Firebase Admin SDK

 **API Documentation:** Auto-generated via FastAPI

 **Deployment:** ASGI server (Uvicorn)

# 2. System Architecture

## High-Level Architecture

FastAPI Application

┌─────────────────────────┐

│ API Routes │ Models │

│ Validation │ Logging │

└─────────────────────────┘

↓

Database Layer

┌─────────────────────────┐

│ Firestore Client │

│ Firebase Admin SDK │

└─────────────────────────┘

↓

Google Cloud Firestore

┌─────────────────────────┐

│ prompts Collection │

│ └─versions Subcollection│

└─────────────────────────┘

## Component Breakdown

### API Layer Components

python

*# File Structure*

project-root/

├── main.py *# FastAPI application and route handlers*

├── models.py *# Pydantic models for request/response validation*

├── firestore\_client.py *# Database connection and configuration*

├── config.py *# Application configuration (optional)*

├── utils.py *# Utility functions (optional)*

├── requirements.txt *# Python dependencies*

└── serviceAccountKey.json *# Firebase service account credentials*

### Request Flow

* + - 1. Client Request → FastAPI Router → Pydantic Validation
      2. Route Handler → Business Logic → Firestore Transaction
      3. Database Operation → Response Formation → Client Response

🗄

# Database Design

## Firestore Database Schema

### Collection:

prompts

javascript

*// Document Structure*

{

*// Document ID: {prompt\_title} (e.g., "customer-service-bot")* title: string, *// "customer-service-bot"* latestVersion: number, *// 3*

createdAt: timestamp, *// "2025-01-15T10:30:00Z"* updatedAt: timestamp, *// "2025-01-18T09:15:00Z"* isActive: boolean, *// true*

deactivatedAt: timestamp, *// null (only set if deactivated)* reactivatedAt: timestamp, *// null (only set if reactivated)* metadata: {

createdBy: string, *// "*[*user@company.com*](mailto:user@company.com)*"*

lastModifiedBy: string, *// "*[*user@company.com*](mailto:user@company.com)*"*

tags: array, *// ["customer-service", "production"]*

category: string *// "chatbot"*

}

}

# API Specification

## API Endpoints

### Create Prompt

POST /prompts/

Content-Type: application/json

Request Body:

{

"title": "string",

"content": "string", "config": {

"temperature": 0.7,

"max\_tokens": 150

}

}

Response (201 Created):

{

"message": "Prompt saved successfully", "version": 1,

"title": "string"

}

Errors:

* 400 Bad Request: Invalid input data
* 500 Internal Server Error: Database operation failed

### List Active Prompts

yaml

GET /prompts/

Response (200 OK): [

{

"id": "string",

"title": "string", "latestVersion": 1,

"createdAt": "2025-01-15T10:30:00Z",

"updatedAt": "2025-01-15T10:30:00Z",

"isActive": true

}

]

Errors:

- 500 Internal Server Error: Database query failed

### Get Prompt with Latest Version

yaml

GET /prompts/{title}

Path Parameters:

- title: string (required)

Response (200 OK):

{

"title": "string", "latestVersion": 1,

"createdAt": "2025-01-15T10:30:00Z", "updatedAt": "2025-01-15T10:30:00Z",

"isActive": true, "latestVersionData": {

"version": 1, "content": "string", "config": {...},

"timestamp": "2025-01-15T10:30:00Z"

}

}

Errors:

* 404 Not Found: Prompt not found or inactive
* 500 Internal Server Error: Database operation failed

### Get Prompt Versions

yaml

GET /prompts/{title}/versions

Path Parameters:

- title: string (required)

Response (200 OK): [

{

"version": 1, "content": "string", "config": {...},

"timestamp": "2025-01-15T10:30:00Z",

"isActive": true

}

]

Errors:

* 404 Not Found: Prompt not found
* 500 Internal Server Error: Database query failed

### Get Specific Version

yaml

GET /prompts/{title}/versions/{version}

Path Parameters:

* title: string (required)
* version: integer (required)

Response (200 OK):

{

"version": 1, "content": "string", "config": {...},

"timestamp": "2025-01-15T10:30:00Z",

"isActive": true

}

Errors:

* 404 Not Found: Prompt or version not found
* 500 Internal Server Error: Database operation failed

### Deactivate Prompt

yaml

PATCH /prompts/{title}/deactivate

Path Parameters:

- title: string (required)

Response (200 OK):

{

"message": "Prompt 'title' has been deactivated (can be reactivated if needed)"

}

Errors:

* 404 Not Found: Prompt not found
* 500 Internal Server Error: Database operation failed

### Reactivate Prompt

yaml

PATCH /prompts/{title}/activate

Path Parameters:

- title: string (required)

Response (200 OK):

{

"message": "Prompt 'title' has been reactivated"

}

Errors:

* 404 Not Found: Prompt not found
* 500 Internal Server Error: Database operation failed

### Admin: List All Prompts

yaml

GET /prompts/admin/all

Response (200 OK): [

{

"id": "string",

"title": "string", "latestVersion": 1,

"createdAt": "2025-01-15T10:30:00Z", "updatedAt": "2025-01-15T10:30:00Z",

"isActive": true

}

]

Errors:

- 500 Internal Server Error: Database query failed

### Health Check

yaml

GET /

Response (200 OK):

{

"message": "Prompt Management API is running", "version": "1.0.0",

"features": [...]

}