Business Requirements Document (BRD)

1. Executive Summary

As Generative AI adoption accelerates across learning design, instructional engineering, and content development, organizations face growing challenges in **prompt reuse, consistency, quality control, and lifecycle management**. Teams frequently recreate similar prompts, introduce subtle variations without governance, and lack a trusted, queryable repository aligned to instructional design frameworks such as **ADDIE (Analyze, Design, Develop, Implement, Evaluate)**.

This initiative introduces a **phased Prompt Management System** that enables:

- Structured capture of prompts aligned to ADDIE
- Human and automated vetting for quality and similarity
- A **Vertex AI Agent** that allows users to *ask for prompts by ADDIE phase*
- Progressive automation from pilot to full production

The system begins with lightweight tooling (Google Forms, Sheets, human review) and evolves into a **fully automated, agent-driven, BigQuery-backed platform** with semantic similarity, versioning, and governance.

2. Problem Statement

Current Challenges

- Prompts are created ad hoc and stored inconsistently (docs, chats, notebooks).
- High duplication of “similar sounding” prompts with no visibility.
- No standardized framework alignment (e.g., ADDIE).
- No quality gate or testing before reuse.
- No agent-friendly, authoritative data source for prompt retrieval.

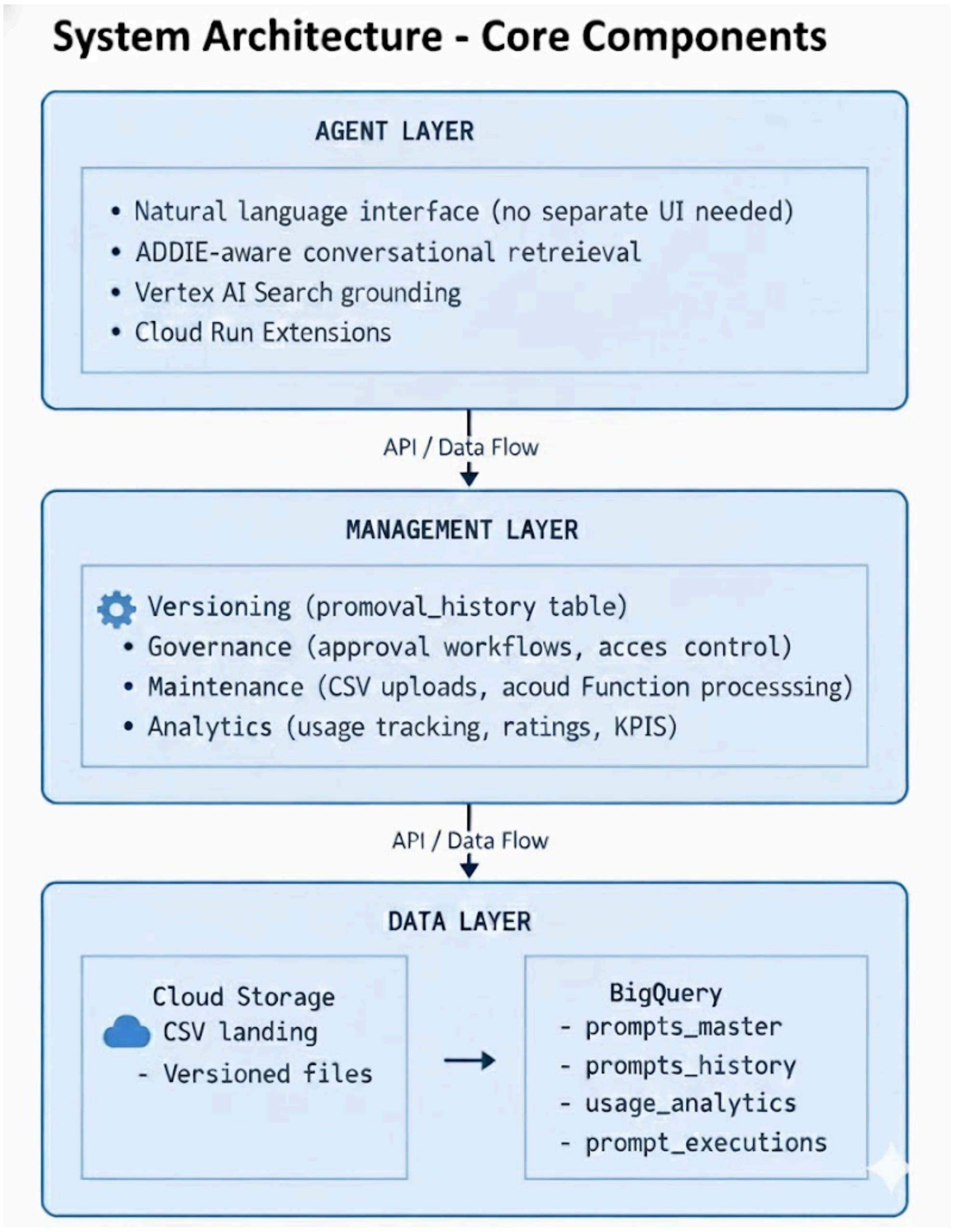
Business Impact

- Reduced productivity due to prompt re-creation.
- Inconsistent outputs across teams and courses.
- Increased risk of low-quality or misaligned prompts being reused.
- Lack of trust in AI-assisted workflows.

3. Business Objectives

1. **Standardize prompt creation** using the ADDIE framework.
2. **Reduce duplication** through similarity detection.
3. **Improve quality and consistency** via review and testing.
4. **Enable natural-language prompt retrieval** via a Vertex AI Agent.
5. **Progressively automate** with minimal disruption to users.

4. Agent-Centric Prompt Management Architecture


System Architecture - Core Components
AGENT LAYER
• Natural language interface (no separate UI needed)
• ADDIE-aware conversational retreieval
• Vertex AI Search grounding
• Cloud Run Extensions
API / Data Flow
MANAGEMENT LAYER
Versioning (promoval_history table)
• Governance (approval workflows, acces control)
• Maintenance (CSV uploads, acoud Function processsing)
• Analytics (usage tracking, ratings, KPIS)
API / Data Flow
DATA LAYER
Cloud Storage
CSV landing
- Versioned files
BigQuery
- prompts_master
- prompts_history
- usage_analytics
- prompt_executions

System Architecture - Detailed Layer Explanation

AGENT LAYER (Top Layer)

What it does: This is our **user-facing intelligence and interface** - the layer that curriculum developers actually interact with.

Components:

1. Natural language interface (no separate UI needed)

- **What:** Gemini Enterprise (Agentspace) provides a chat interface
- **Why:** No need to build React/HTML/CSS - Google provides the conversational UI
- **Benefit:** Zero frontend development, works on web + mobile automatically
- **User experience:** Developers ask questions like "Find me prompts for learner analysis" instead of navigating menus

2. ADDIE-aware conversational retrieval

- **What:** The agent understands ADDIE framework (Analysis, Design, Development, Implementation, Evaluation)
- **Why:** Prompts are organized by ADDIE phases, so agent needs domain knowledge
- **Benefit:** Agent can suggest "You're in Analysis phase, next try Design prompts for objectives"
- **Example:** User says "I need to analyze learners" → Agent knows to search Analysis phase

3. Vertex AI Search grounding

- **What:** Semantic search powered by vector embeddings
- **Why:** Keyword search misses meaning - "learner personas" and "audience analysis" are semantically similar
- **Benefit:** Finds relevant prompts even if exact keywords don't match
- **How it works:** our query "analyze audience" finds prompts about "learner personas" because they mean similar things

4. Cloud Run Extensions

- **What:** Three microservices the agent calls (query_prompts, execute_prompt, log_analytics)
- **Why:** Agent needs to DO things beyond just chat - execute prompts via Gemini, track usage, filter searches
- **Benefit:** Agent can "call APIs" to get structured data or perform actions
- **Example:** When user says "Execute this prompt," agent calls execute_prompt extension which runs Gemini API

Why this layer exists:

Traditional systems need separate layers for UI and intelligence. Gemini Enterprise **combines both -** it's smart (ADDIE-aware) AND it's the interface (chat). This eliminates months of frontend development.

MANAGEMENT LAYER (Middle Layer)

What it does: Handles the **lifecycle, quality, and governance** of our prompt library.

Components:

1. Versioning (prompts_history table)

- **What:** Every prompt change creates a new version; old versions never deleted
- **Why:** You need to know "what did this prompt look like 6 months ago?" or rollback bad changes
- **Benefit:** Complete audit trail, can revert to any previous version
- **Example:** Prompt ANLYS-LRN-001 v1 → v2 → v3. If v3 is bad, revert to v2
- **Technical:** prompts_history table stores every version with timestamp, author, change notes

2. Governance (approval workflows, access control)

- **What:** Rules about who can add/edit prompts and whether changes need approval
- **Why:** Can't let everyone publish prompts without review - quality control needed
- **Benefit:** Maintains prompt library quality, prevents unauthorized changes
- **Example:** Junior developer proposes new prompt → Senior reviews → Approves → Goes live
- **Technical:** IAM roles control who can upload CSVs, future: approval workflows in system

3. Maintenance (CSV uploads, Cloud Function processing)

- **What:** How new prompts get into the system

- **Why:** Need a simple way for non-developers to add prompts (CSV is universal)
- **Benefit:** Upload CSV → Automatic processing → Prompts available in minutes
- **Technical:**
    - User uploads `new_prompts.csv` to Cloud Storage
    - Cloud Function triggers automatically
    - Validates CSV, generates embeddings, loads to BigQuery
    - Updates Vertex AI Search index

4. Analytics (usage tracking, ratings, KPIs)

- **What:** Track which prompts are used, how often, ratings, execution success
- **Why:** Need to know "which prompts are most valuable?" and "which need improvement?"
- **Benefit:** Data-driven decisions on prompt quality
- **Example:** Prompt rated 2/5 stars with 0 usage → deprecate or improve
- **Technical:** Every usage logged to usage_analytics table, aggregated for dashboards

Why this layer exists:

Without management, our prompt library becomes chaotic - no versions, no quality control, no visibility into what works. This layer ensures **sustainable, governed, high-quality** prompt library.

DATA LAYER (Bottom Layer)

What it does: Persistent storage - the single source of truth for all prompts and usage data.

Two Storage Systems (and why both):

Cloud Storage (Left box)

What it stores:

- CSV files (raw prompt data)
- Versioned copies of all uploads

What it does:

1. **CSV landing zone:** Where users upload new prompts
2. **File versioning:** Keeps old versions of CSV files for 30 days
3. **Trigger point:** File upload triggers Cloud Function

Why we need it:

- **Version control friendly:** CSV files can be committed to Git
- **User-friendly:** Anyone can edit in Excel/Google Sheets
- **Backup:** Original files preserved even after BigQuery import
- **Event trigger:** Cloud Storage events start processing pipeline

Example flow:

User exports from Excel → uploads new_prompts.csv →
GCS stores file → triggers Cloud Function → processes CSV

BigQuery (Right box)

What it stores:

- `prompts_master` - Current version of all active prompts
- `prompts_history` - Every version ever created (audit trail)
- `usage_analytics` - Every time someone uses a prompt
- `prompt_executions` - Every time someone executes a prompt via Gemini

What it does:

1. **System of Record:** Canonical, queryable database
2. **Analytics engine:** Fast SQL queries for dashboards
3. **Search source:** Vertex AI Search pulls from here
4. **API backend:** Extensions query these tables

Why we need it:

- **Analytics optimized:** Designed for OLAP queries (aggregations, trends, reports)
- **Serverless:** No database servers to manage
- **Scalable:** Handles kilobytes to petabytes automatically
- **SQL familiar:** Everyone knows SQL, easy to query
- **Time travel:** Built-in 7-day rollback for accidents

Example queries:

```
-- Find most popular prompts
SELECT prompt_id, COUNT(*) as usage
FROM usage_analytics
GROUP BY prompt_id
ORDER BY usage DESC;

-- Average rating by ADDIE phase
SELECT addie_phase, AVG(avg_rating)
```

```
FROM prompts_master
GROUP BY addie_phase;
```

Arrow between them (→):

Represents the **data flow**: Cloud Storage → Cloud Function → BigQuery

Process:

1. CSV uploaded to Cloud Storage
2. Cloud Function reads CSV
3. Generates embeddings for semantic search
4. Writes to BigQuery (both prompts_master and prompts_history)
5. Syncs to Vertex AI Search for semantic queries

Why TWO storage systems?

Aspect	Cloud Storage	BigQuery
Purpose	File landing, backup	Structured queries, analytics
Format	Raw CSV files	SQL tables with schema
Access	File operations	SQL queries
Best for	Uploads, versions	Searching, reporting
Persistence	90 days (lifecycle)	Indefinite

Why not just one?

- **Cloud Storage alone:** Can't efficiently query "show me all Design phase prompts with rating >4"
- **BigQuery alone:** Users can't easily upload data (no CSV → BigQuery direct path without code)
- **Together:** Best of both - easy uploads (GCS) + powerful queries (BigQuery)

Data Flow Example (End-to-End)

Let's trace what happens when a curriculum developer uses the system:

Scenario 1: Adding a new prompt

1. Developer creates CSV with new prompt in Excel
   ↓
2. Uploads to Cloud Storage bucket
   ↓ (automatic trigger)
3. Cloud Function wakes up:
   - Reads CSV
   - Validates format
   - Generates embedding (768-dimension vector)
   - Determines if new or update (version check)
   ↓
4. Writes to BigQuery:
   - prompts_master (current version)
   - prompts_history (audit trail)
   ↓
5. Syncs to Vertex AI Search
   ↓
6. ✅ Prompt now searchable in Gemini agent

Time: 30-60 seconds from upload to availability

Scenario 2: Finding and executing a prompt

1. Developer asks agent: "Find me a learner persona prompt"
   ↓
2. AGENT LAYER:
   - Gemini understands request
   - Calls Vertex AI Search (semantic search)
   - Vertex AI Search queries BigQuery for ADDIE phase=Analysis
   ↓
3. Agent presents results: "Found ANLYS-LRN-001 (4.5 stars, used 47 times)"
   ↓
4. Developer: "Execute it with topic=Cloud Security"
   ↓
5. AGENT LAYER:
   - Calls execute_prompt extension (Cloud Run)
   ↓
6. MANAGEMENT LAYER:
   - Extension retrieves prompt from BigQuery
   - Fills placeholders with user inputs
   - Calls Gemini API to generate content
   ↓
7. DATA LAYER:

```
   - Logs execution to prompt_executions table
  ↓
8. Agent shows generated persona
  ↓
9. Developer: "Rate it 5 stars"
  ↓
10. MANAGEMENT LAYER:
    - Calls log_analytics extension
   ↓
11. DATA LAYER:
    - Writes to usage_analytics table
    - Updates avg_rating in prompts_master
```

Time: 3-5 seconds from request to generated content

Why This Architecture?

Design Principles:

1. Separation of Concerns

- **Agent Layer:** User interaction ONLY
- **Management Layer:** Business logic ONLY
- **Data Layer:** Storage ONLY

Benefit: Can change storage (switch from BigQuery to Spanner) without touching agent code.

2. Event-Driven

- CSV upload → automatic processing (no manual trigger)
- Agent request → automatic extension call
- No polling, no cron jobs

Benefit: Instant response, zero idle resources.

3. Serverless

- No servers to manage anywhere
- Auto-scales from 0 to 1000s of users
- Pay only when used

Benefit: Zero ops burden, predictable costs.

4. Managed Services

- Gemini Enterprise = managed agent + UI
- Vertex AI Search = managed vector search
- BigQuery = managed database
- Cloud Storage = managed file storage

Benefit: Google handles availability, scaling, backups, security patches.

Cost Breakdown (Why Each Layer Costs What It Does)

For 100 users making 10 queries/day:

Layer	Service	Monthly Cost	Why?
Agent	Vertex AI Search	$60	30K semantic queries × $0.002/query
Agent	Gemini API	$10	2K executions × $0.005/request
Management	Cloud Functions	$0.04	100 CSV uploads × $0.0004/invocation
Management	Cloud Run	$0.30	500K extension calls × minimal cost
Data	BigQuery Storage	$0.20	10GB × $0.02/GB/month
Data	BigQuery Queries	$0.25	50GB scanned × $0.005/GB
Data	Cloud Storage	$0.10	5GB × $0.02/GB/month
Total		**~$71/month**	

Key insight: 84% of cost is Agent Layer (Vertex AI Search) because that's where the intelligence is. Data layer is dirt cheap (<2% of cost).

Key Takeaways

1. **Agent Layer = Intelligence + Interface** (Gemini does both, eliminating traditional UI layer)

2. **Management Layer = Governance + Processing** (keeps library high-quality and maintainable)

3. **Data Layer = Dual Storage Strategy**
    - Cloud Storage for easy uploads + file versioning
    - BigQuery for powerful queries + analytics
4. **Data flows down** (upload → process → store) and **queries flow up** (data → logic → agent → user)
5. **Total architecture cost = ~$71/month** for 100 users, with 84% spent on intelligence (search + AI)

This architecture is **production-ready** yet **simple enough to maintain** - no complex Kubernetes, no microservices sprawl, just three clean layers with clear responsibilities.

8. Pilot Phase (Phase 1)

Phase 1 Goal

Validate the **end-to-end value** of structured prompt capture and agent-based retrieval with **human-in-the-loop controls**.

Phase 1 Architecture (Conceptual)

Prompt Capture

- Google Forms used to submit:
    - Prompt text
    - ADDIE phase
    - Intended use
    - Tags
    - Author

Intermediate Review

- Responses stored in **Google Sheets**
- Human reviewer:
    - Checks similarity manually
    - Reviews quality and alignment
    - Approves or rejects

Approved Prompt Storage

- Approved prompts exported to **Google Cloud Storage (GCS)**
- Each prompt stored as a structured JSON or Markdown file

Agent Consumption

- Vertex AI Agent Builder configured with:
  - **Cloud Storage as a data source**
- Agent answers:
  "Give me a Design-phase prompt for learning objectives"

Key Characteristics

- Manual similarity detection
- Manual approval
- Simple governance
- Fast to deploy and iterate

---

9. Architectural Overview – Full Production Phase (Phase 2)

Phase 2 Goal

Achieve **near-perfect automation** while preserving quality, governance, and trust.

Phase 2 Architecture (Conceptual)

Prompt Capture

- UI or Agent-assisted submission
- Prompts stored directly in **BigQuery (Prompt Backlog)**

Automated Similarity Detection

- Prompt embeddings generated using Vertex AI
- Similarity search performed against existing prompts
- Threshold-based duplicate detection

Testing & Vetting

- Automated prompt tests (rubrics, format checks)
- Results stored in BigQuery

Lifecycle Management

- Versioning
- Status transitions (Draft → Testing → Approved → Published)

Authoritative Data Source

- **BigQuery Prompt Registry**
- Only approved, active prompts exposed

Agent Consumption

- Vertex AI Agent:
    - Uses **Prompt Registry** (BigQuery or exported index)
    - Retrieves prompts by:
        - ADDIE phase
        - Intent
        - Tags
    - Explains selection rationale

---

10. Data Sources by Phase

Phase	Data Source	Purpose
Pilot	Google Sheets	Capture + review
Pilot	Google Cloud Storage	Agent data source
Production	BigQuery (Backlog)	Draft & review
Production	BigQuery (Registry)	Published prompts
Production	Vertex Embeddings	Similarity search

---

11. Vertex AI Agent Responsibilities

Agent Capabilities

- Interpret user intent
- Enforce ADDIE phase filtering
- Retrieve only approved prompts
- Explain why a prompt was selected

- Ask clarifying questions if needed

Example Agent Interaction

User: “I need a Design-phase prompt to create learning objectives.”

Agent:

- Filters prompts where `ADDIE = Design`
- Ranks by semantic relevance
- Returns top prompt + explanation

---

12. Success Metrics

Pilot Phase

- % of prompts reused
- Reviewer satisfaction
- Time to find a prompt

Production Phase

- Reduction in duplicate prompts
- Test pass rates
- Agent retrieval precision
- User satisfaction with agent responses

---

13. Risks & Mitigations

Risk	Mitigation
Low-quality prompts	Human review (Phase 1)
Over-automation too early	Phased rollout
Agent hallucination	Authoritative registry only
User trust	Explainable retrieval

---

14. Roadmap Summary

Phase	Description
Phase 1	Google Forms → Sheets → Human Review → GCS → Agent
Phase 2	BigQuery → Similarity → Testing → Registry → Agent

15. Conclusion

This phased approach balances **speed, trust, and scalability**, allowing teams to immediately benefit from structured prompt reuse while building toward a **fully automated, agent-driven prompt platform** aligned to the ADDIE framework and Google Cloud best practices.