

**Title:** Technical Product Requirements Document (PRD): Blogger Plus (B+) **Version:** 1.0 (Alpha Strategic) **Date:** January 21, 2026 **Target Audience:** Engineering Leads, System Architects, Product Managers

**Keywords:** Google Cloud Run, Vertex AI RAG, Spanner Graph, Workspace Add-ons, Gemini 1.5 Pro, SaaS Architecture, System Design, Google AdSense API

## 1. Executive Summary & Developer Mandate

**The Objective:** We are not building a blog; we are building the **Operating System for the Creator Economy**. The business goal is a \$10 Trillion valuation. Your technical mandate is to build a platform capable of **engineering virality** through predictive AI, not just hosting text.

**The "Brutally Honest" Technical Reality:** We cannot achieve this scale with a standard LAMP stack or by wrapping the ChatGPT API.

- **The Trap:** If you build a monolithic app, we will die at 10 million users.
- **The Requirement:** You must utilize **Google Cloud Run** for serverless scale and **Spanner Graph** for the social layer.
- **The AI Pivot:** "Content generation" is a commodity. Our differentiator is **Vertex AI RAG** (Retrieval-Augmented Generation) which grounds content in user research to create high-authority outputs.

## 2. High-Level System Architecture

**Core Philosophy:** Serverless-First, Event-Driven, Graph-Based.

- **Compute: Google Cloud Run.** It offers the flexibility of containers with the simplicity of serverless, allowing us to scale to zero when idle and burst to millions of requests instantly.
  - *Decision:* Do not use App Engine Standard; Cloud Run provides necessary container flexibility for our custom AI agents.
- **Database (The Dual-Core Strategy):**
  - **Transactional: Cloud SQL (PostgreSQL)** for user accounts, subscriptions, and financial ledgers.
  - **Social & Relationship: Spanner Graph.** We need purpose-built graph capabilities compatible with SQL to map 100M+ users and their interest intersections. *Warning: Do not use Firestore for the complex social graph; it will fragment at scale.*
- **Storage: Google Cloud Storage** buckets, strictly integrated with **Google Drive API** for user asset management.

## 3. Module Specifications & Technical Implementation

### 3.1 Module A: Workspace Integration (The "Sticky" Layer)

- **Objective:** Users must draft in Docs and publish to B+ without a distinct "export" step.
- **Tech Stack:** Google Apps Script, Workspace Add-ons API.
- **Manifest Strategy:** Define a `addOns.common` section in the manifest to ensure the tool persists across Drive, Docs, and Sheets.
- **Scopes:**
  - <https://www.googleapis.com/auth/documents.readonly> (Content Fetching)

- <https://www.googleapis.com/auth/drive.file> (Asset Linking)

### 3.2 Module B: Gemini Predictive Engine (The "Virality" Layer)

- **Objective:** Score content potential *before* publishing.
- **Model:** Gemini 1.5 Pro via Vertex AI.
- **Constraint Management:**
  - **Rate Limits:** We must utilize **Tier 2** or higher paid quotas. The Free Tier (5 RPM) is insufficient for anything beyond "Hello World" testing.
  - **Token Optimization:** Use the Batch API for non-urgent predictive scoring to handle up to 300M tokens/day.
- **Implementation:**
  - *Input:* Blog Draft + Historical Trend Data (from Spanner).
  - *Process:* Vertex AI Agent analyzes sentiment, structure, and keyword density against "viral" clusters.
  - *Output:* A 0-100 "Virality Score" with specific rewrite suggestions.

### 3.3 Module C: Knowledge Retrieval (RAG / NotebookLM)

- **Objective:** "Deep Research" capabilities that cite user's own notes.
- **Architecture:** Vertex AI RAG Engine.
- **Workflow:**
  1. **Ingestion:** User uploads PDFs/Docs.
  2. **Chunking:** Vertex AI automatically chunks and vectorizes this data.
  3. **Retrieval:** When writing, the AI retrieves relevant chunks to "ground" the generation, reducing hallucinations.
- **Vector DB:** Use **Vertex AI Vector Search** for low-latency retrieval of millions of embeddings.

### 3.4 Module D: Social & Community (The "Moat")

- **Objective:** High-density interest graphs.
- **Data Model:** **Spanner Graph**.
  - *Why?* It allows querying complex relationships (e.g., "Find all users who liked posts about 'AI' and follow 'TechCrunch'") using ISO GQL (Graph Query Language) alongside standard SQL.
- **Real-Time Sync:** Use **Firebase Realtime Database** or **Firestore** strictly for the *ephemeral* chat layer, syncing back to Spanner for permanence.

## 4. Security & Compliance (Google Standards)

- **Identity:** **Google Identity Services (OAuth 2.0)**. No custom password handling.
- **IAM Policy:** Enforce "Least Privilege". Service accounts for Cloud Run must not have generic Editor access; scope strictly to specific Cloud SQL instances and Storage buckets.
- **Visibility:** Implement **Google Cloud Security Command Center** to detect misconfigurations (e.g., public buckets) immediately.

## 5. Monetization Integration

- **AdSense:** Use the **AdSense Management API** for "Embedded Connect".
- **Flow:**
  1. User authenticates via OAuth.
  2. Use `accounts.list` to fetch their Ad Clients.
  3. Inject ad units dynamically into the Single Page Application (SPA) structure.

## 6. Development Roadmap & Risks

### Phase 1: Foundation (Months 1-3)

- Setup Google Cloud Organization & Billing.
- Deploy Spanner Graph schema.
- Build Workspace Add-on "Hello World".

### Phase 2: The Intelligence (Months 4-6)

- Train Vertex AI agents on "Viral" datasets.
- Implement RAG pipeline with Vector Search.

### Phase 3: Scale (Months 7-9)

- Load test Cloud Run with 1M concurrent simulated users.
- Launch Beta to select creators.

### Critical Risks (The "Mirror"):

- **Cost Blowout:** Vertex AI RAG can get expensive. Monitor "Vector Search" node hours closely.
- **Latency:** RAG retrieval adds latency. Cache common queries aggressively.
- **Platform Risk:** If we rely too heavily on specific Workspace UI hacks, a Google UI update could break the editor. Stick to the official **Add-ons Card Service** layout.

## 7. Verified References (Source Catalogue)

### Strategic Context

1. Strategic Business Plan: Blogger Plus .pdf (Internal)
2. Blogger Plus Logo Assets (Internal)

**Cloud Infrastructure** 3. Cloud Run vs App Engine Comparison 4. Cloud Run Autoscaling Documentation 5. Cloud Run Pricing & TCO

**AI & Machine Learning** 6. Vertex AI RAG Engine Implementation Guide 7. Vertex AI Vector Search Architecture 8. Gemini API Rate Limits (2025 Guide) 9. Gemini Batch API Limits & Token Pricing 10. NotebookLM Deep Research & Source Types 11. RAG Grounding & Citation Mechanisms

**Data & Social Graph** 12. Spanner Graph & ISO GQL Documentation 13. Firestore vs. Spanner for Scalability 14. Vector Database Use Cases (Google Cloud)

**Workspace & Integration** 15. Google Workspace Add-ons Manifests 16. Workspace Add-ons Configuration 17. AdSense Embedded Connect for Platforms 18. AdSense Management API Quickstart

**Security** 19. Google Cloud Security Best Practices 20. IAM & Least Privilege on GCP

**AI Disclosure:** This PRD was generated using **Gemini 2.5 Flash**. The AI acted as a Lead Systems Architect, synthesizing internal strategic requirements with live Google Cloud technical documentation to ensure architectural validity and scalability.

- 

**Next Step:** Would you like me to generate the **YAML Manifest file** for the Workspace Add-on or the **SQL Schema** for the Spanner Graph user database?