AI-Based Idea Filtering Solution for Deep Funding (MVP)

Deep Labs AI-Based Idea Filtering Pod

July 14, 2025

1. Project Objective

The goal is to develop a Minimum Viable Product (MVP) for a LLM-powered idea filtering system within 50 hours, including project management. The system will:

- Store ideas in Google Firestore upon submission.
- Process ideas via an API using OpenAI's GPT-4 for scoring (impact, feasibility, alignment), feed- back generation, and categorization.
- Detects duplicates using OpenAI embeddings and FAISS, flagging ideas with cosine similarity > 0.9.
- Host the API using Django REST Framework for public access.
- Provide a simple UI for idea submission.
- Visualize trends in Google Data Studio.
- Reduce manual review workload by over 60% while maintaining human oversight.

2. Problem Statement

Deep Funding faces challenges in scaling proposal evaluation:

- Resource Drain: 70% of reviewer time spent on initial screening.
- Inconsistent Prioritization: Subjective bias in early-stage filtering.
- Duplication Overlook: 15-20% redundant proposals per funding round.
- Feedback Delays: Weeks-long wait times for applicant feedback.
- Strategic Blind Spots: No systematic way to identify trend gaps or innovation hotspots.

3. Project Scope & Methodology

a. Scope

i. **Input:** Unstructured idea submissions (text) + metadata.

ii. **Output:**
   - ➢ Priority-ranked idea list with scores (impact, feasibility, alignment).
   - ➢ Automated feedback reports.
   - ➢ Categorization tags (e.g., Technology, Healthcare).
   - ➢ Duplicate detection flags and similar idea references.

iii. **Integration:** Django REST API with Firestore and Google Data Studio.

b. Methodology

i. **Database**: Google Firestore to store idea text, submission date, scores, feedback, category, dupli- cate flag, similar ideas, and embeddings.
ii. **AI Processing**: Integrated into the POST API using OpenAI GPT-4 for scoring, feedback, and categorization; OpenAI embeddings and FAISS for duplicate detection.
iii. **Backend**: Django REST Framework for API hosting.
iv. **Frontend**: Simple Django template for idea submission.
v. **Visualization**: Google Data Studio for real-time trend analysis.

4. Key Technical Components

a. **Firestore**: Stores idea data and metadata.
b. **OpenAI API**: GPT-4 for evaluation and embeddings for similarity search.
c. **FAISS**: Efficient similarity search for duplicate detection.
d. **Django REST Framework**: Hosts API endpoints for submission and retrieval.
e. **Google Data Studio**: Visualizes idea metrics (e.g., category distribution, duplicate count).

5. Team Member Contributions

a. Developer Responsibilities

i. Set up Firestore schema and Django project.

ii. Integrate OpenAI APIs for scoring, feedback, categorization, and embeddings.
iii. Implement FAISS for duplicate detection.
iv. Develop API endpoints (POST /submit-idea, GET /ideas).
v. Create a simple Django template for idea submission.
vi. Connect Firestore to Google Data Studio for visualization.

b. Key Deliverables

i. Firestore database with idea metadata.
ii. Django REST API with integrated AI processing.
iii. Basic submission UI.
iv. Google Data Studio dashboard.

c. Quality Assurance Responsibilities

i. Validate GPT outputs for fairness and consistency.
ii. Test API endpoints and AI processing logic.
iii. Verify dashboard accuracy.

d. Key Deliverables

i. Test reports for API and AI functionality.
ii. Validation of duplicate detection (90% recall target).
iii. Dashboard accuracy checks.

6. Milestones

Phase	Activities
Database Setup	Configure Firestore schema.
AI Integration	Develop API with OpenAI and FAISS processing.

Backend Development	Build Django REST API.
UI Development	Create a submission interface.
Data Visualization	Set up Google Data Studio dashboard.
Testing	Validate API, AI, and visualization.
Project Management	Plan and monitor progress.