# **RFP\_Solution.zip**

This document contains the comprehensive technology solution for automating the generation of new Request for Proposal (RFP) templates for your mutual fund organization, powered by AI-driven ingestion and analysis of example RFPs.

## **Table of Contents**

1. [README.md](#a3fi19n90vvk)
2. [architecture.md](#leog6sw36taw)
   * [1. Solution Overview](#fkxdu4dg3shv)
   * [2. Technical Architecture Description](#sibwciceqpyf)
   * [3. Knowledge Base Structure](#b844zdrkott9)
   * [4. AI Integration Strategy](#nsf21cet2qya)
3. [prompts.md](#bx18srfu27wx)
   * [OpenAI Prompts for Example RFP Data Extraction](#4tbn9taml07d)
     + [General Extraction Instructions](#x5ox89m79gdt)
     + [Prompt 1: Extract Executive Summary & Project Overview](#nem274mqbiug)
     + [Prompt 2: Extract Background & Context of RFP Issuer](#togruz1oyeuk)
     + [Prompt 3: Extract Objectives & Goals of the RFP](#j71xe9fs3jv9)
     + [Prompt 4: Extract Scope of Work & Deliverables](#dbs0izhqpd1a)
     + [Prompt 5: Extract Specific Requirements & Questions (Categorized)](#ahxymnw4cn7j)
     + [Prompt 6: Extract Evaluation Criteria & Selection Process](#iinfxnp32hu3)
     + [Prompt 7: Extract Timelines & Submission Instructions](#u8wc10tzo5ru)
     + [Prompt 8: Extract Commercials & Pricing Structure](#g2ffqxpm4bvo)
   * [OpenAI Prompts for New RFP Template Generation](#2wn5rvs77f7s)
     + [Prompt A: Generate New RFP Executive Summary](#3f128w5t1t1b)
     + [Prompt B: Generate New RFP Background & Purpose](#ucjitdyy9q1x)
     + [Prompt C: Generate New RFP Scope of Work](#1pkj19qep8af)
     + [Prompt D: Generate New RFP Requirements Section (Synthesized Q&A)](#c5jsg28meujb)
     + [Prompt E: Generate New RFP Evaluation Criteria](#ntvy199x0weh)
     + [Prompt F: Generate New RFP Timeline & Submission](#sps8vs8y2ct7)
     + [Prompt G: Generate New RFP Commercials & Pricing Instructions](#yzgld2iof0gx)
4. [diagrams.md](#gq0363hjfy1t)
   * [1. Solution Diagram](#5tb5xdgtzmb0)
   * [2. Architecture Diagram](#bgx312o5g8f3)
   * [3. Sequence Diagram: Example RFP Ingestion & Data Extraction](#290do9ogh229)
   * [4. Sequence Diagram: New RFP Template Generation](#zbth1xcdfq7n)

## **README.md**

This document outlines the "AI RFP Generator" solution, designed to streamline and enhance the creation of new Request for Proposal (RFP) templates for your mutual fund organization. The system enables users to upload a set of 5-7 example RFPs, from which it extracts common sections, questions, and requirements. This intelligence is then used by AI to synthesize a comprehensive and tailored new RFP template, ready for your organization to issue to potential vendors or service providers.

**Key Features:**

* **Example RFP Ingestion:** Upload multiple (5-7) example RFPs (DOCX/PDF) for automated parsing and detailed data extraction.
* **AI-Powered Extraction:** OpenAI models extract key sections, questions, objectives, risk/compliance requirements, and other data points from the uploaded example RFPs.
* **Intelligent Knowledge Base:** A structured SQLite database stores extracted information, creating a repository of common RFP elements.
* **New RFP Template Generation:** Synthesizes extracted patterns and content from the example RFPs to generate a new, customized RFP template (DOCX format) for your organization to issue.
* **Compliance & Consistency:** Helps ensure that newly generated RFPs include standard sections and a professional tone based on learned examples.
* **Human-in-the-Loop:** Allows for human review and refinement of extracted data and generated RFP content, ensuring accuracy and alignment with organizational needs.

**Technology Stack:**

* **Frontend (UI):** React
* **Backend (API):** FastAPI
* **Database:** SQLite (for structured data storage)
* **File Storage:** AWS S3 (for uploaded RFPs and generated templates)
* **AI Integration:** OpenAI API (gpt-4o recommended)
* **Document Processing:** python-docx for DOCX parsing/generation, PyMuPDF for PDF parsing, python-docx-template for DOCX templating.

**How to Navigate This Document:**

* **architecture.md**: Describes the high-level solution, technical components, and data structures with specified technologies.
* **prompts.md**: Provides detailed, regressive OpenAI prompts for both data extraction from example RFPs and content generation for new RFP templates.
* **diagrams.md**: Contains visual representations (Mermaid diagrams) of the system's flow and components, including interaction with the new technologies.

**Important Note:** This document represents a conceptual design. Implementing this solution requires a development team and adherence to all relevant data privacy, security, and financial regulations (e.g., SEBI guidelines).

## **architecture.md**

### **1. Solution Overview**

The "AI RFP Generator" system has two primary phases:

* **Phase 1: Example RFP Ingestion & Intelligence Gathering:** Users upload a collection of 5-7 example RFPs (these are RFPs that *your organization has received* or *industry standard RFPs* you want to learn from). An automated pipeline extracts key information (executive summary, firm history, objectives, scope, requirements, evaluation criteria, timelines, commercials, etc.) from each using OpenAI. This extracted data is stored in a central SQLite knowledge base.
* **Phase 2: New RFP Template Generation:** Based on the aggregated and analyzed data from the ingested example RFPs, the system synthesizes a new, comprehensive RFP template. This template is designed for *your organization to issue* to potential vendors or service providers, leveraging the common patterns, question types, and structural elements learned from the examples.

### **2. Technical Architecture Description**

The solution follows a modular architecture leveraging the specified technologies.

* **Client-Side (Web UI):**
  + **Purpose:** User interaction for uploading example RFPs, initiating new RFP template generation, and viewing/downloading generated documents.
  + **Technologies:** **React**. Provides intuitive forms, file upload widgets, progress indicators, and dynamic displays for generated content.
* **Backend API Service:**
  + **Purpose:** The central hub handling all requests, orchestrating workflows, interacting with AI, database, and storage.
  + **Technologies:** **FastAPI** (Python). Chosen for its high performance, ease of use, and automatic API documentation (Swagger/OpenAPI).
  + **Endpoints:**
    - POST /upload\_example\_rfp: Handles receiving single/multiple file uploads.
    - GET /list\_example\_rfps: Lists previously uploaded example RFPs.
    - POST /generate\_new\_rfp\_template: Triggers the synthesis and generation of a new RFP template based on selected example RFPs.
    - GET /download\_rfp\_template/{template\_id}: Allows downloading of generated RFP templates.
    - GET /health: Basic health check.
* **Document Parsing Service:**
  + **Purpose:** Converts uploaded .docx and .pdf files into raw, extractable text. This can be a module within the FastAPI service or a separate microservice.
  + **Technologies:** Python with libraries:
    - python-docx for .docx files.
    - PyMuPDF (or fitz) for robust .pdf text extraction. (For scanned PDFs, integrate an OCR solution like Tesseract or a cloud OCR API before text extraction).
* **AI Extraction Service:**
  + **Purpose:** Interfaces with OpenAI to extract structured information from unstructured text based on specific prompts.
  + **Technologies:** Python with openai library. Manages prompt construction, API calls, error handling, and response parsing. Handles document chunking for large inputs if necessary.
* **Knowledge Base Service (Database):**
  + **Purpose:** Stores all structured data about your organization (e.g., standard boilerplate text for your RFPs) and, critically, the extracted data points from the ingested example RFPs.
  + **Technologies:** **SQLite**. Chosen for its simplicity, zero-configuration setup, and suitability for prototyping or small-to-medium scale deployments where a single-file database is sufficient. (Note: For high concurrency or large data volumes, a more robust database like PostgreSQL would be recommended for a production environment).
  + **Schemas:**
    - OrganizationBoilerplate: Stores reusable text snippets for your own RFPs (e.g., standard introduction, legal disclaimers).
    - ExampleRFPExtractions: Stores the structured JSON output from the AI extraction for each uploaded example RFP.
      * rfp\_id (PK, UUID), original\_filename, upload\_date, raw\_text\_content, extracted\_executive\_summary\_json, extracted\_background\_json, extracted\_objectives\_json, extracted\_scope\_json, extracted\_requirements\_json, extracted\_evaluation\_json, extracted\_timelines\_json, extracted\_commercials\_json.
    - CommonRFPQuestions: Aggregates and standardizes questions extracted from multiple example RFPs.
      * question\_id (PK, UUID), category, standardized\_question\_text, frequency\_count, example\_sources (list of rfp\_ids), last\_updated\_date.
* **Document Generation Service:**
  + **Purpose:** Populates Jinja2 templates within .docx files with synthesized data to create the new RFP template.
  + **Technologies:** Python with python-docx-template (which combines python-docx and Jinja2).
* **File Storage:**
  + **Purpose:** Securely stores uploaded raw example RFPs and the generated new RFP templates.
  + **Technologies:** **AWS S3**. Provides highly scalable, durable, and secure cloud object storage.

### **3. Knowledge Base Structure**

The ExampleRFPExtractions and CommonRFPQuestions tables are crucial for this solution.

**ExampleRFPExtractions Table Schema:**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| rfp\_id | TEXT (UUID) | Primary Key. Unique identifier for the uploaded example RFP. |
| original\_filename | TEXT | The original name of the file uploaded by the user. |
| upload\_date | TEXT (ISO 8601) | Timestamp of when the RFP was uploaded. |
| raw\_text\_content | TEXT | The full raw text extracted from the uploaded RFP. |
| extracted\_executive\_summary\_json | JSON | Structured JSON output from Prompt 1 for this specific example RFP. |
| extracted\_background\_json | JSON | Structured JSON output from Prompt 2 for this specific example RFP. |
| extracted\_objectives\_json | JSON | Structured JSON output from Prompt 3 for this specific example RFP. |
| extracted\_scope\_json | JSON | Structured JSON output from Prompt 4 for this specific example RFP. |
| extracted\_requirements\_json | JSON | Structured JSON output from Prompt 5 for this specific example RFP (list of categorized questions/requirements). |
| extracted\_evaluation\_json | JSON | Structured JSON output from Prompt 6 for this specific example RFP. |
| extracted\_timelines\_json | JSON | Structured JSON output from Prompt 7 for this specific example RFP. |
| extracted\_commercials\_json | JSON | Structured JSON output from Prompt 8 for this specific example RFP. |

**CommonRFPQuestions Table Schema:**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| question\_id | TEXT (UUID) | Primary Key. Unique identifier for the standardized question. |
| category | TEXT | General category of the question (e.g., 'Investment Strategy', 'Compliance', 'Reporting'). |
| standardized\_question\_text | TEXT | A cleaned, standardized version of the question, to group similar questions together (e.g., "Describe your risk management process."). |
| frequency\_count | INTEGER | A count of how many times this (or a very similar) question appeared across all ingested example RFPs. |
| example\_sources | TEXT (JSON list) | A JSON array of rfp\_ids from ExampleRFPExtractions where this question was found. |
| last\_updated\_date | TEXT (ISO 8601) | Timestamp of when this question was last updated or encountered. |

### **4. AI Integration Strategy**

The AI integration uses OpenAI's gpt-4o or gpt-4 for two distinct phases, with precise prompt engineering for structured output.

1. **Regressive Information Extraction (RFP Ingestion Phase):**
   * Each of the 5-7 uploaded example RFPs is processed individually.
   * A series of chained prompts (Prompt 1 to Prompt 8) are executed. Each prompt focuses on extracting a specific section or type of data, building context and ensuring comprehensive coverage.
   * temperature is kept very low (e.g., 0.1) to prioritize factual extraction and minimize hallucination.
   * The output of each prompt is strictly JSON, which is then parsed and stored in the ExampleRFPExtractions table.
   * A post-processing step (can also use AI) analyzes the question\_text fields from extracted\_requirements\_json across all ingested RFPs to populate/update the CommonRFPQuestions table, identifying common themes and standardizing phrasing.
2. **Synthesis and New RFP Template Generation (New RFP Template Generation Phase):**
   * When the user requests a new RFP template, the system aggregates relevant information from the ExampleRFPExtractions and CommonRFPQuestions tables.
   * A new set of prompts (Prompt A to Prompt G) are used to generate each section of the new RFP.
   * These prompts guide the AI to synthesize common elements, best practices, and frequently asked questions/requirements into coherent and well-structured prose for the new RFP.
   * temperature can be slightly higher (e.g., 0.5) to allow for good prose generation while still maintaining factual grounding.

**Chunking Strategy:** For very large RFPs, the Document Parsing Service will split the raw text. The AI Extraction Service will then process these chunks sequentially, ensuring each prompt has enough context to perform its extraction. For synthesis (generation), the AI can be given a summary of the common extracted points to generate larger sections.

## **prompts.md**

Here are the detailed OpenAI prompts for both the extraction from example RFPs and the generation of new RFP templates. All prompts target JSON output for structured data.

### **OpenAI Prompts for Example RFP Data Extraction**

*(Used in the* ***RFP Ingestion Process*** *to learn about the structure and content of prospective clients' RFPs)*

#### **General Extraction Instructions**

System Message (for all extraction prompts):

You are an expert financial analyst specializing in Request for Proposals (RFPs). Your task is to accurately extract very specific information from the provided RFP text. Adhere strictly to the requested JSON output format for each prompt. If information for a requested key is not explicitly found or is not applicable, set its value to 'null' or an empty array '[]' as appropriate for the JSON structure, but do not omit the key.

**Temperature Setting:** temperature=0.1 (for high accuracy and low creativity/hallucination)

#### **Prompt 1: Extract Executive Summary & Project Overview**

**User Message:**

Extract the executive summary or the primary overview section from the following RFP text. This section usually outlines the core purpose of the RFP, the client's (RFP issuer's) primary needs, and the overall scope of the services or project they are seeking. Also, identify if the client has mentioned any specific initial project scope, estimated budget, or key overarching goals in this overview.  
  
Output should be a JSON object with the following keys:  
- `section\_title`: (STRING) The exact title of the section if identifiable (e.g., "Executive Summary", "Introduction"). If not found, set to "Overview".  
- `summary\_text`: (STRING) The full text of the executive summary or a synthesized overview of the RFP's main purpose if no distinct summary is present.  
- `project\_scope\_overview`: (STRING) A brief description of the services or project being sought, as outlined in this introductory section.  
- `estimated\_budget\_mentioned`: (BOOLEAN) True if a budget amount or range is explicitly mentioned, False otherwise.  
- `budget\_amount`: (STRING, nullable) The specific budget amount or range mentioned (e.g., "$100,000 - $150,000", "INR 5 Crores"), otherwise null.  
- `overarching\_goals`: (ARRAY of STRING) A list of high-level goals or desired outcomes the RFP aims to achieve (e.g., "enhance operational efficiency", "select new investment manager").  
  
RFP Text:  
---  
[Full text of the RFP document or relevant introductory pages]  
---

#### **Prompt 2: Extract Background & Context of RFP Issuer**

**User Message:**

From the following RFP text, extract information pertaining to the RFP issuer's (client's) background, history, and organizational context. This includes details about their establishment, mission, key milestones, and general operational or business scope.  
  
Output should be a JSON object with the following keys:  
- `section\_title`: (STRING) The exact title of the section if identifiable (e.g., "About Us", "Company Background"). If not found, set to "Organization Background".  
- `firm\_name`: (STRING) The full legal name of the RFP issuing organization.  
- `establishment\_details`: (STRING) When the organization was established and any notable founding details (e.g., "Established in 1985", "Founded in 1990 as a non-profit").  
- `mission\_and\_values`: (STRING) The organization's mission, vision, or core values as stated.  
- `key\_milestones`: (ARRAY of STRING) A brief list of significant organizational achievements, historical points, or evolution stages.  
- `organizational\_structure\_type`: (STRING) A summary of their structure or type (e.g., "non-profit endowment", "government pension fund", "private corporation", "educational institution").  
- `current\_aum\_or\_financial\_size`: (STRING, nullable) Their approximate assets under management, total financial size, or relevant budget/revenue figures, if mentioned (e.g., "$500 million AUM", "Annual budget of $20M").  
- `relevant\_industry\_context`: (STRING) Any specific industry or sector context relevant to their operations.  
  
RFP Text:  
---  
[Full text of the RFP document]  
---

#### **Prompt 3: Extract Objectives & Goals of the RFP**

**User Message:**

From the following RFP text, extract the detailed objectives and specific goals that the RFP issuing organization aims to achieve by procuring the services outlined in this RFP. These are typically the specific outcomes they expect from the selected provider.  
  
Output should be a JSON object with the following keys:  
- `section\_title`: (STRING) The exact title of the section if identifiable (e.g., "Objectives", "Goals and Outcomes"). If not found, set to "RFP Objectives".  
- `primary\_objectives\_list`: (ARRAY of STRING) A list of the main objectives of this specific RFP (e.g., "improve investment performance", "implement a new CRM system", "enhance cybersecurity posture").  
- `quantitative\_goals`: (ARRAY of STRING) Any measurable or quantitative targets (e.g., "achieve 8% annualized return", "reduce operational costs by 15%", "complete implementation within 6 months").  
- `qualitative\_goals`: (ARRAY of STRING) Any qualitative outcomes desired (e.g., "strengthen vendor relationship", "improve data accuracy", "enhance user experience").  
- `reason\_for\_issuing\_rfp\_detailed`: (STRING) A more detailed explanation of why they are seeking this service/investment now (e.g., "due to growth of fund", "in response to new regulatory requirements", "to replace existing outdated system").  
  
RFP Text:  
---  
[Full text of the RFP document]  
---

#### **Prompt 4: Extract Scope of Work & Deliverables**

**User Message:**

Extract the detailed Scope of Work (SOW) and specific deliverables expected from the successful vendor/provider, as described in the following RFP text. This section outlines the tasks, responsibilities, and outputs required.  
  
Output should be a JSON object with the following keys:  
- `section\_title`: (STRING) The exact title of the section if identifiable (e.g., "Scope of Work", "Services Required"). If not found, set to "Scope of Work".  
- `scope\_summary`: (STRING) A comprehensive summary of the entire scope of work.  
- `key\_tasks\_responsibilities`: (ARRAY of STRING) A list of the main tasks and responsibilities assigned to the vendor.  
- `expected\_deliverables`: (ARRAY of STRING) A list of concrete outputs, reports, systems, or services expected (e.g., "monthly performance reports", "implemented software module", "annual compliance review").  
- `client\_responsibilities\_mentioned`: (ARRAY of STRING) Any responsibilities or inputs expected from the RFP issuing client (e.g., "provide necessary data access", "allocate internal project manager").  
- `geographic\_scope`: (STRING, nullable) Any mention of geographical or operational boundaries (e.g., "India-specific", "global operations", "servicing all branches").  
  
RFP Text:  
---  
[Full text of the RFP document]  
---

#### **Prompt 5: Extract Specific Requirements & Questions (Categorized)**

**User Message:**

From the following RFP text, extract all specific requirements, detailed questions, and criteria that a prospective vendor must address in their proposal. Categorize these items logically.  
  
Output should be a JSON object with the following keys:  
- `section\_title`: (STRING) The exact title of the section if identifiable (e.g., "Vendor Requirements", "Technical Questions"). If not found, set to "Requirements and Questions".  
- `categorized\_requirements`: (ARRAY of OBJECTS) Each object represents a category of questions/requirements.  
 - `category\_name`: (STRING) The name of the category (e.g., "Investment Philosophy", "Technical Capabilities", "Client Service Model", "Risk Management", "Compliance & Regulatory", "Reporting & Data", "Personnel & Team", "Past Experience", "Pricing Structure", "Implementation & Support", "ESG Integration", "Cybersecurity").  
 - `questions`: (ARRAY of STRING) A list of specific questions or requirements under this category, as phrased in the RFP. If a requirement is not a question, rephrase it as a clear requirement statement.  
  
RFP Text:  
---  
[Full text of the RFP document]  
---

#### **Prompt 6: Extract Evaluation Criteria & Selection Process**

**User Message:**

Extract the criteria and process by which the RFP issuing organization will evaluate the submitted proposals and select a vendor. Look for weighting factors, evaluation stages (e.g., initial screening, presentations), and decision-making authority.  
  
Output should be a JSON object with the following keys:  
- `section\_title`: (STRING) The exact title of the section if identifiable (e.g., "Evaluation Criteria", "Selection Process"). If not found, set to "Proposal Evaluation".  
- `evaluation\_criteria\_summary`: (STRING) A summary of the overall evaluation approach.  
- `key\_criteria\_list`: (ARRAY of STRING) A list of the most important factors for evaluation (e.g., "Technical Competence", "Financial Proposal", "Experience", "Cultural Fit").  
- `weighting\_factors`: (STRING, nullable) Any explicit weighting or scoring breakdown mentioned (e.g., "Technical (40%), Commercial (30%), Experience (30%)").  
- `evaluation\_stages`: (ARRAY of STRING) A sequence of steps in the evaluation process (e.g., "Initial Review", "Shortlisting", "On-site Presentations", "Reference Checks", "Final Negotiation").  
- `decision\_makers\_mentioned`: (STRING, nullable) Any indication of who will make the final decision (e.g., "Investment Committee", "Procurement Team").  
  
RFP Text:  
---  
[Full text of the RFP document]  
---

#### **Prompt 7: Extract Timelines & Submission Instructions**

**User Message:**

Extract all important dates, deadlines, and specific instructions for submitting proposals from the following RFP text. This includes submission methods, contact persons for queries, and any required formats.  
  
Output should be a JSON object with the following keys:  
- `section\_title`: (STRING) The exact title of the section if identifiable (e.g., "RFP Schedule", "Submission Guidelines"). If not found, set to "RFP Timeline & Submission".  
- `key\_dates`: (ARRAY of OBJECTS) Each object representing a key date.  
 - `event`: (STRING) Name of the event (e.g., "RFP Release Date", "Question Submission Deadline", "Proposal Due Date", "Finalist Notification Date").  
 - `date`: (STRING) The specific date or deadline (e.g., "June 30, 2025", "within 2 weeks of release").  
- `submission\_method`: (STRING) How proposals should be submitted (e.g., "online portal", "email to designated address", "hard copy by mail").  
- `required\_format`: (ARRAY of STRING) Any specific formatting requirements (e.g., "PDF format", "max 50 pages", "separate technical and commercial proposals").  
- `contact\_person\_for\_queries`: (STRING, nullable) Name and contact details of the person for questions.  
- `pre\_bid\_conference\_details`: (STRING, nullable) Date, time, and location of any pre-bid conference.  
- `number\_of\_copies\_required`: (STRING, nullable) If hard copies are required, how many.  
  
RFP Text:  
---  
[Full text of the RFP document]  
---

#### **Prompt 8: Extract Commercials & Pricing Structure**

**User Message:**

Extract all information related to the commercial aspects, pricing models, and fee structures that the RFP issuer expects or requests from the vendor in their proposal.  
  
Output should be a JSON object with the following keys:  
- `section\_title`: (STRING) The exact title of the section if identifiable (e.g., "Pricing Proposal", "Commercial Terms"). If not found, set to "Commercials and Fees".  
- `pricing\_model\_requests`: (ARRAY of STRING) The types of pricing models requested (e.g., "fixed fee", "time and materials", "percentage of AUM", "performance-based fees").  
- `specific\_fee\_details\_requested`: (ARRAY of STRING) Any specific costs or charges they want itemized (e.g., "management fees", "custody fees", "implementation costs", "annual maintenance fees").  
- `payment\_terms\_expectations`: (STRING, nullable) Any preferred payment schedules or terms.  
- `invoicing\_requirements`: (STRING, nullable) Specifics about invoicing procedures.  
- `contract\_duration\_expected`: (STRING, nullable) The expected contract length.  
- `budget\_allocation\_breakdown\_request`: (ARRAY of STRING) If they request a breakdown of budget allocation.  
  
RFP Text:  
---  
[Full text of the RFP document]  
---

### **OpenAI Prompts for New RFP Template Generation**

*(Used in the* ***New RFP Template Generation Process*** *to synthesize a new RFP that your organization will issue)*

System Message (for all generation prompts):

You are a professional RFP document generator for [Your Mutual Fund Company Name]. Your task is to synthesize information from various example RFPs to create a coherent, comprehensive, and well-structured new RFP template that our organization will issue. Maintain a formal, clear, and direct tone. Ensure sections flow logically and all necessary information for a vendor response is requested. You will be provided with aggregated data extracted from 5-7 example RFPs.

**Temperature Setting:** temperature=0.5 (for balanced creativity in prose generation and adherence to structure)

#### **Prompt A: Generate New RFP Executive Summary**

**User Message:**

Generate the Executive Summary section for a new RFP template that our organization ([Your Mutual Fund Company Name]) will issue. This summary should clearly articulate the purpose of our RFP and the primary objective of the services we are seeking.  
  
Aggregated Project Overviews (from ExampleRFPExtractions.extracted\_executive\_summary\_json):  
---  
[Synthesized list of `project\_scope\_overview` from multiple example RFPs, highlighting common themes or the most comprehensive one]  
---  
  
Common Overarching Goals (from ExampleRFPExtractions.extracted\_executive\_summary\_json):  
---  
[Synthesized list of `overarching\_goals` from multiple example RFPs, showing common desired outcomes]  
---  
  
Our Organization's Context (from OrganizationBoilerplate or FirmInfo):  
- Name: [Your Mutual Fund Company Name]  
- Brief Mission/Focus: [Your company's mission/focus, e.g., "delivering superior returns and service to investors in India"]  
  
Synthesize these elements into a compelling executive summary for our new RFP.

#### **Prompt B: Generate New RFP Background & Purpose**

**User Message:**

Generate the "Background and Purpose" section for a new RFP template that our organization ([Your Mutual Fund Company Name]) will issue. This section should introduce our organization and explain why we are seeking proposals.  
  
Our Organization's Detailed Background (from OrganizationBoilerplate or FirmInfo):  
- Name: [Your Mutual Fund Company Name]  
- Establishment: [Your company's establishment details]  
- Mission/Values: [Your company's mission/values]  
- Business Context: [Brief overview of your company's business and relevance to this RFP]  
  
Aggregated Reasons for Issuing RFPs (from ExampleRFPExtractions.extracted\_objectives\_json.reason\_for\_issuing\_rfp\_detailed from multiple examples):  
---  
[Synthesized common reasons for issuing similar RFPs from your analyzed examples, e.g., "to enhance existing capabilities", "to meet evolving market demands", "to explore new technologies"]  
---  
  
Combine these to explain our organization and the strategic importance of this RFP.

#### **Prompt C: Generate New RFP Scope of Work**

**User Message:**

Generate the "Scope of Work" section for a new RFP template. This section should clearly define the services, tasks, and responsibilities we expect from the successful vendor/provider.  
  
Aggregated Scope Summaries (from ExampleRFPExtractions.extracted\_scope\_json.scope\_summary from multiple examples):  
---  
[Synthesized common themes or comprehensive summaries of scope from the example RFPs]  
---  
  
Common Key Tasks & Responsibilities (from ExampleRFPExtractions.extracted\_scope\_json.key\_tasks\_responsibilities from multiple examples):  
---  
[List of frequently occurring tasks/responsibilities requested in similar RFPs]  
---  
  
Common Expected Deliverables (from ExampleRFPExtractions.extracted\_scope\_json.expected\_deliverables from multiple examples):  
---  
[List of frequently requested deliverables from similar RFPs]  
---  
  
Synthesize these points to form a comprehensive Scope of Work section. Include a prompt for the vendor to outline their own proposed methodology.

#### **Prompt D: Generate New RFP Requirements Section (Synthesized Q&A)**

**User Message:**

Generate the "Requirements and Questions" section for a new RFP template. This section should present a comprehensive set of questions and requirements that we expect prospective vendors to address. Base these on the common questions identified across multiple example RFPs.  
  
Common RFP Questions (from CommonRFPQuestions table, prioritized by `frequency\_count` and category relevance to the new RFP's purpose):  
---  
[A curated list of categories and `standardized\_question\_text` from CommonRFPQuestions, e.g.,  
{  
 "category\_name": "Investment Philosophy",  
 "questions": ["Describe your investment philosophy and decision-making process.", "How do you manage portfolio concentration?"]  
},  
{  
 "category\_name": "Risk Management",  
 "questions": ["Detail your risk identification, measurement, and monitoring framework.", "How do you handle liquidity risk under stressed market conditions?"]  
},  
...  
]  
---  
  
Organize these questions logically under clear category headings. For each question, prompt the vendor to provide detailed responses. Include a general statement about requiring comprehensive answers.

#### **Prompt E: Generate New RFP Evaluation Criteria**

**User Message:**

Generate the "Evaluation Criteria and Selection Process" section for a new RFP template. This section should outline how we will assess vendor proposals and make our selection.  
  
Aggregated Evaluation Criteria (from ExampleRFPExtractions.extracted\_evaluation\_json from multiple examples):  
---  
[Synthesized common `key\_criteria\_list` and `weighting\_factors` from example RFPs, e.g., "Technical Competence (40%)", "Commercial Proposal (30%)", "Experience (20%)", "Client Service (10%)"]  
---  
  
Common Evaluation Stages (from ExampleRFPExtractions.extracted\_evaluation\_json.evaluation\_stages from multiple examples):  
---  
[Synthesized common evaluation stages, e.g., "Initial Screening", "Detailed Review", "Vendor Presentations", "Reference Checks", "Final Negotiation"]  
---  
  
Synthesize these points to explain our evaluation methodology, including criteria, weighting (if applicable), and the expected stages of the selection process.

#### **Prompt F: Generate New RFP Timeline & Submission Instructions**

**User Message:**

Generate the "RFP Timeline and Submission Instructions" section for a new RFP template. This section should provide all critical dates and guidelines for vendors to submit their proposals.  
  
Common Key Dates (from ExampleRFPExtractions.extracted\_timelines\_json.key\_dates from multiple examples):  
---  
[Synthesized common key event types and their typical sequence from example RFPs, e.g., "RFP Release Date", "Questions Due Date", "Proposal Submission Deadline", "Finalist Presentation Dates", "Anticipated Award Date"]  
---  
  
Common Submission Methods & Formats (from ExampleRFPExtractions.extracted\_timelines\_json from multiple examples):  
---  
[Synthesized common `submission\_method`, `required\_format`, and `number\_of\_copies\_required` from example RFPs]  
---  
  
Synthesize these to clearly outline the schedule and provide precise instructions on how to submit a compliant proposal.

#### **Prompt G: Generate New RFP Commercials & Pricing Instructions**

**User Message:**

Generate the "Commercial Proposal and Pricing Structure" section for a new RFP template. This section should instruct vendors on how to present their financial proposals and what cost details we require.  
  
Common Pricing Model Requests (from ExampleRFPExtractions.extracted\_commercials\_json.pricing\_model\_requests from multiple examples):  
---  
[Synthesized common `pricing\_model\_requests` (e.g., "fixed fee", "AUM-based", "hourly rates") from example RFPs]  
---  
  
Common Specific Fee Details Requested (from ExampleRFPExtractions.extracted\_commercials\_json.specific\_fee\_details\_requested from multiple examples):  
---  
[Synthesized common `specific\_fee\_details\_requested` (e.g., "implementation fees", "recurring service fees", "licensing costs") from example RFPs]  
---  
  
Common Payment Terms Expectations (from ExampleRFPExtractions.extracted\_commercials\_json.payment\_terms\_expectations from multiple examples):  
---  
[Synthesized common `payment\_terms\_expectations` (e.g., "quarterly in advance", "milestone-based payments") from example RFPs]  
---  
  
Instruct the vendor on the required pricing format, what costs to itemize, and any preferred payment terms.

## **diagrams.md**

### **1. Solution Diagram**

This diagram illustrates the two main workflows: **Example RFP Ingestion** and **New RFP Template Generation**.

graph TD  
 subgraph "Example RFP Ingestion Workflow"  
 A[User Uploads 5-7 Example RFPs] --> B{Document Parsing Service}  
 B --> C[Extracted Raw Text (per RFP)]  
 C -- For Each RFP --> D[AI Extraction Service (Prompts 1-8)]  
 D --> E{Structured Extracted Data (per RFP)}  
 E --> F[Knowledge Base Updater]  
 F --> G[SQLite Knowledge Base]  
 G -- Notification --> H[Human Review & Enrichment]  
 end  
  
 subgraph "New RFP Template Generation Workflow"  
 I[User Initiates New RFP Generation] --> J{FastAPI Backend Service}  
 J --> K[SQLite Knowledge Base]  
 K -- Aggregates & Synthesizes Data from ExampleRFPExtractions & CommonRFPQuestions --> J  
 J --> L[AI Content Generation/Synthesis (Prompts A-G)]  
 L --> M[Assembled Content for New RFP]  
 M --> N{Document Generation Service}  
 N --> O[Final New RFP Template (DOCX)]  
 O --> P[User Downloads Template]  
 end  
  
 G -- Aggregated Data & Common Patterns --> L

### **2. Architecture Diagram**

This diagram shows the major system components and their interactions, specifying the chosen technologies.

graph LR  
 subgraph "Client-Side (UI)"  
 UI[React Frontend]  
 end  
  
 subgraph "Backend Services (FastAPI)"  
 API\_GW(FastAPI App)  
 DOC\_PARSE(Document Parsing Module)  
 AI\_EXTRACT(AI Extraction Module)  
 DOC\_GEN(Document Generation Module)  
 end  
  
 subgraph "Data & Storage Layer"  
 KB(SQLite Database)  
 FILE\_STORE(AWS S3 Storage)  
 end  
  
 subgraph "External AI"  
 OPENAI[OpenAI API]  
 end  
  
 UI -- HTTP Requests (Upload, Generate, Download) --> API\_GW  
  
 API\_GW -- File Bytes --> FILE\_STORE  
 FILE\_STORE -- File Data --> DOC\_PARSE  
 DOC\_PARSE -- Extracted Text --> AI\_EXTRACT  
 AI\_EXTRACT -- API Calls --> OPENAI  
 OPENAI -- JSON Response --> AI\_EXTRACT  
 AI\_EXTRACT -- Writes Structured Data --> KB  
  
 API\_GW -- Reads/Writes Data --> KB  
 API\_GW -- Requests Document Generation --> DOC\_GEN  
 DOC\_GEN -- Reads Data --> KB  
 DOC\_GEN -- Generates .docx --> FILE\_STORE  
  
 FILE\_STORE -- Serves Uploaded/Generated Files --> UI  
  
 Human\_Review[Human Analyst/Content Team] -- Direct Interaction (Review/Curate) --> KB

### **3. Sequence Diagram: Example RFP Ingestion & Data Extraction**

This diagram details the sequence of events when example RFPs are uploaded for extraction.

sequenceDiagram  
 participant User  
 participant UI as React UI  
 participant API\_GW as FastAPI Backend  
 participant S3 as AWS S3 Storage  
 participant DocParse as Document Parser  
 participant AIExtract as AI Extraction Service  
 participant OpenAI as OpenAI API  
 participant DB as SQLite DB  
  
 User->>UI: Uploads 5-7 Example RFP Files (e.g., .docx, .pdf)  
 UI->>API\_GW: POST /upload\_example\_rfp (sends files)  
 API\_GW->>S3: Stores raw RFP files  
 API\_GW->>API\_GW: Generates unique rfp\_id for each file  
  
 loop For each uploaded RFP file  
 API\_GW->>DocParse: Request text extraction (file\_path)  
 DocParse->>S3: Reads raw file  
 DocParse-->>API\_GW: Returns raw text content  
 API\_GW->>DB: Stores raw\_text\_content in ExampleRFPExtractions (with rfp\_id)  
  
 loop For each Extraction Prompt (1 through 8)  
 API\_GW->>AIExtract: Send raw\_text\_content & Prompt N  
 AIExtract->>OpenAI: API Call (Prompt N, raw\_text)  
 OpenAI-->>AIExtract: Returns JSON extracted\_data\_N  
 AIExtract-->>API\_GW: Returns extracted\_data\_N  
 API\_GW->>DB: Updates ExampleRFPExtractions.extracted\_data\_N\_json for current rfp\_id  
 end  
  
 API\_GW->>API\_GW: Post-processing: Analyze extracted\_requirements\_json  
 API\_GW->>DB: Updates CommonRFPQuestions (standardize, count frequency)  
 end  
  
 API\_GW-->>UI: Confirms RFP ingestion complete  
 UI-->>User: Displays status / notification  
 User->>DB: (Human Analyst) Reviews & enriches CommonRFPQuestions (updates predefined\_answer, is\_reviewed\_by\_human)

### **4. Sequence Diagram: New RFP Template Generation**

This diagram details the sequence of events for generating a new RFP template.

sequenceDiagram  
 participant User  
 participant UI as React UI  
 participant API\_GW as FastAPI Backend  
 participant DB as SQLite DB  
 participant AIExtract as AI Extraction Service  
 participant OpenAI as OpenAI API  
 participant DocGen as Document Generation Service  
 participant S3 as AWS S3 Storage  
  
 User->>UI: Selects "Generate New RFP Template"  
 UI->>API\_GW: POST /generate\_new\_rfp\_template (optional: specify purpose/context)  
  
 API\_GW->>DB: Query ExampleRFPExtractions (to get all extracted data)  
 API\_GW->>DB: Query CommonRFPQuestions (to get common questions/patterns)  
 DB-->>API\_GW: Returns aggregated extracted data & common questions  
  
 API\_GW->>API\_GW: Synthesize aggregated data for AI prompts  
  
 loop For each Generation Prompt (A through G)  
 API\_GW->>AIExtract: Send aggregated\_data & Prompt X (e.g., Prompt A for Executive Summary)  
 AIExtract->>OpenAI: API Call (Prompt X, aggregated\_data)  
 OpenAI-->>AIExtract: Returns generated\_text\_X  
 AIExtract-->>API\_GW: Returns generated\_text\_X  
 API\_GW->>API\_GW: Stores generated\_text\_X in temporary structure  
 end  
  
 API\_GW->>DocGen: Request document assembly (all generated texts, template path)  
 DocGen->>DocGen: Populates Jinja2 template with generated content  
 DocGen->>S3: Saves final .docx template  
 S3-->>DocGen: Returns template file path/URL  
 DocGen-->>API\_GW: Returns template file path/URL  
  
 API\_GW-->>UI: Returns template download link  
 UI->>User: Provides link to download new RFP template  
 User->>S3: Downloads the generated .docx file