Final Specification: AI Proposal Maker API

This document outlines the design for an AI-powered API that generates proposal drafts. It synthesizes information from a bidder's profile and a specific tender's details to automatically complete the required proposal format.

1. User Identification: Why `bidder_id` is Not in the Request Format

You are correct that the system must identify the bidder to fetch the correct profile. However, for security reasons, this is not done by sending a `bidder_id` in the request body.

Instead, the bidder is identified using a secure authentication token sent in the request header. This is a standard, secure practice that prevents users from impersonating one another. Your own API documentation for the Dynamic Bidder Profile already establishes this pattern by stating that authentication is "Required (to identify the user)"[1111].

The process is as follows:

1. The user sends a request with their secure token in the header.
2. The server validates the token and extracts the `auth_user_id`[2].
3. This server-verified `auth_user_id` is then used to query the `users` collection and securely retrieve the correct bidder's profile data[3333].

2. Understanding the `proposalFormat` for Your Requirements

The

`proposalFormat` is a field within the scraped tender data that acts as a template for the proposal that needs to be submitted[444]. Based on the example in your

`Tender Scraping.pdf`, it is an array of objects, where each object defines a specific question or requirement for the proposal[555].

Here is a breakdown of its structure based on your provided data:

- **`section`**: A string used to group related questions, such as "A: Company Profile" or "B: Technical Proposal"[666666666].
- **`questionId`**: A unique string identifier for each question, like "A1" or "B2"[777777777].
- **`questionText`**: The actual text of the question presented to the bidder, for example,

"Describe your proposed technical approach and methodology."[888888888].

- **responseType**: A string indicating the expected format of the answer, such as "file_upload", "text_long", or "gantt_chart_upload"[999999999].
- **isRequired**: A boolean value (`true` or `false`) that indicates whether the question must be answered[1010101010101010].

The AI's task is to generate an appropriate answer for each object in this `proposalFormat` array.

3. API Endpoint Specification

- **API Name**: `POST /api/v1/proposals/generate`
- **Method**: `POST`
- **Authentication**: Required[11111111]. The bidder's identity is determined from the authentication token.

4. API Process Flow

1. **Receive Request**: The API receives a request to the `POST /api/v1/proposals/generate` endpoint with a valid authentication token in the header.
2. **Authenticate and Identify**: The server validates the token and extracts the `auth_user_id`[12].
3. **Fetch Bidder Profile**: Using the `auth_user_id`, the server queries the `users` collection to retrieve the bidder's `profile_data`[13131313].
4. **Fetch Tender Details**: Using the `tender_id` from the request body, the server queries the `tenders` collection to get the full tender document, including `tenderDetails` and `proposalFormat`[141414].
5. **Construct AI Prompt**: The backend constructs a detailed prompt for the Gemini API, providing the tender information as context and the bidder's profile as the source for answers.
6. **Call Gemini API**: The server sends the prompt to the Gemini API.
7. **Format and Return**: The API receives the AI-generated answers, formats them into a final JSON structure, and returns the completed proposal draft.

5. Input and Output Formats

Input Format (Request Body)

The request body only needs the identifier for the target tender.

JSON

```
{
  "tender_id": "tend_f4a8b1c9"
}
```

Output Format (Success 200 OK)

The output provides the generated proposal, mirroring the `proposalFormat` structure but with an added `generated_answer` field for each question.

JSON


```
{
  "status": "success",
  "data": {
    "bidder_user_id": "ObjectId('...')",
    "tender_id": "tend_f4a8b1c9",
    "tender_title": "Construction of Smart City Command Centre",
    "generated_proposal": [
      {
        "section": "A: Company Profile",
        "question_id": "A1",
        "question_text": "Provide your company's registration certificate.",
        "response_type": "file_upload",
        "is_required": true,
        "generated_answer": {
          "source_field": "profile_data.company_registration_docs",
          "explanation": "This requirement directly maps to the 'company_registration_docs' field in the user's profile.",
          "value": "file_id_or_link_from_profile"
        }
      },
      {
        "section": "B: Technical Proposal",
        "question_id": "B1",
        "question_text": "Describe your proposed technical approach and methodology.",
        "response_type": "text_long",
        "is_required": true,
        "generated_answer": {
          "source_field": null,
          "explanation": "Generated by synthesizing the bidder's specialization in 'Buildings' with the tender's summary about a 'Smart City Command Centre'.",
          "value": "Leveraging our experience in SOFTWARE_DEVELOPMENT and specialization in constructing Commercial Office buildings, our technical approach for the Smart City Command Centre will involve a phased design-build methodology."
```

```
      }
    }
  ]
 }
}
```

6. Gemini API Integration and Prompt Engineering

To ensure accurate results, provide the data to Gemini in a structured prompt.

Suggested Prompt Structure:

Plaintext

```
You are an expert proposal writer. Your task is to fill out a proposal based on the provided Bidder Profile and Tender Details. Analyze the Tender Details to understand the requirements, especially the 'procurementSummary' [cite: 350] and 'eligibilityRequirements'[cite: 362]. Using the Bidder's Profile, answer each question in the 'Proposal Format'. Return ONLY a single valid JSON object containing one key: "proposal_answers". This key should hold an array where each object has "question_id" and "generated_answer".

**1. Tender Details (Context):**
```json
{
 "tenderDetails": {
  "title": "Construction of Smart City Command Centre",
  "procurementSummary": "This project involves the design, development, and construction of a centralized command centre for smart city operations..."
 },
 "eligibilityRequirements": [
  {
   "type": "Financial",
   "description": "Average annual turnover of at least INR 20 Crores in the last 3 financial years."
  }
 ]
}
```

2. Bidder Profile (Source of Answers):

JSON

```
{
 "bidder_type": "COMPANY",
 "profile_data": {
  "company_name": "Innovate Inc.",
  "company_registration_id": "U74999DL2025PTC123456",
  "avg_annual_turnover": 25,
  "project_specialization": ["Buildings"],
```
```

```
    "primary_industry": "SOFTWARE_DEVELOPMENT"
  }
}
```

3. Proposal Format (The Template to Fill)[15]:

JSON

```
[
  {
    "question_id": "A1",
    "question_text": "Provide your company's registration certificate.",
    "response_type": "file_upload"
  },
  {
    "question_id": "B1",
    "question_text": "Describe your proposed technical approach and methodology.",
    "response_type": "text_long"
  }
]
```