## API Specifications

**1. API Name:** GlobalTender API

2. Authentication:

Authentication will be handled via an API key. The client must include the key in the request header.

- **Header Name:** X-API-Key
- **Example:** X-API-Key: your_secret_api_key_here

**3. Request Process: GET vs. POST**

We will support both GET for simple queries and POST for more complex, multi-parameter searches.

---

**A. GET Request: /v1/tenders**

This method is ideal for quick lookups and simple filtering.

**Endpoint:** https://api.yourdomain.com/v1/tenders

**Method:** GET

**Query Parameters:**

| Parameter | Type | Description | Example |
|---|---|---|---|
| country | String | ISO 3166-1 alpha-2 country code. | IN, US, SG |
| state | String | State, province, or region. | Tamil Nadu, California |
| keywords | String | Comma-separated keywords to search in title/summary. | software,construction |

| | | | |
|---|---|---|---|
| tenderId | String | Fetch a specific tender by its unique ID. | tend_1a4b2c |
| min_value | Integer | Minimum tender value. | 100000 |
| currency | String | ISO 4217 currency code. | USD, INR |
| page | Integer | Page number for pagination. Default: 1. | 2 |
| limit | Integer | Number of results per page. Default: 10, Max: 100. | 25 |

Example GET Request:

https://api.yourdomain.com/v1/tenders?country=IN&state=Tamil Nadu&keywords=solar,power&limit=5

---

**B. POST Request: `/v1/tenders/search`**

This method is better for complex queries with multiple criteria that might exceed URL length limits.

**Endpoint:** https://api.yourdomain.com/v1/tenders/search

**Method:** POST

Request Body (JSON):

The body allows for more complex filtering, such as including and excluding keywords.

JSON

```
{
  "filters": {
    "location": {
```

```
    "country": ["IN", "AE"],
    "state": ["Tamil Nadu", "Dubai"]
   },
   "keywords": {
    "include": ["infrastructure", "IT"],
    "exclude": ["maintenance"]
   },
   "dateRange": {
    "closingDateStart": "2025-09-01T00:00:00Z",
    "closingDateEnd": "2025-10-31T23:59:59Z"
   },
   "valueRange": {
    "min": 50000,
    "max": 1000000,
    "currency": "USD"
   }
  },
  "pagination": {
   "page": 1,
   "limit": 20
  }
}
```

### API Output Format (JSON)

The response for both `GET` and `POST` requests will follow this structure.

JSON

```
{
  "status": "success",
  "results": 2,
  "pagination": {
    "currentPage": 1,
    "totalPages": 5,
    "totalResults": 50,
    "limit": 10
  },
  "data": [
    {
      "tenderId": "tend_f4a8b1c9",
      "sourceUrl": "https://example.gov.in/tenders/2025/infra/001",
      "scrapedTimestamp": "2025-08-29T16:30:00Z",
      "country": "IN",
      "state": "Tamil Nadu",
      "region": "Chennai",
      "tenderDetails": {
        "referenceNumber": "TN/INFRA/2025/001",
        "title": "Construction of Smart City Command Centre",
```

```
    "issuingAuthority": "Greater Chennai Corporation",
    "procurementSummary": "This project involves the design, development, and construction of a centralized command centre for smart city operations...",
    "category": ["Infrastructure", "Construction", "IT"],
    "tenderValue": 50000000,
    "currency": "INR",
    "dates": {
     "publishedDate": "2025-08-15T12:00:00Z",
     "clarificationEndDate": "2025-09-05T17:00:00Z",
     "closingDate": "2025-09-30T15:00:00Z",
     "openingDate": "2025-10-01T11:00:00Z"
    }
   },
   "eligibilityRequirements": [
    {
     "type": "Financial",
     "description": "Average annual turnover of at least INR 20 Crores in the last 3 financial years."
    },
    {
     "type": "Experience",
     "description": "Must have completed at least two similar projects of value not less than INR 30 Crores each."
    },
    {
     "type": "Certification",
     "description": "ISO 9001:2015 certified."
    }
   ],
   "proposalFormat": [
    {
     "section": "A: Company Profile",
     "questionId": "A1",
     "questionText": "Provide your company's registration certificate.",
     "responseType": "file_upload",
     "isRequired": true
    },
    {
     "section": "B: Technical Proposal",
     "questionId": "B1",
     "questionText": "Describe your proposed technical approach and methodology.",
     "responseType": "text_long",
     "isRequired": true
    },
    {
     "section": "B: Technical Proposal",
     "questionId": "B2",
     "questionText": "Provide project timelines and key milestones.",
     "responseType": "gantt_chart_upload",
     "isRequired": true
    }
```

```
    ],
    "unstructuredData": {
      "section_4_payment_terms": "Payments will be made in three installments: 40% on
mobilization, 40% on successful completion of milestone 2, and 20% on final project handover...",
      "annex_c_special_conditions": "All bidders must adhere to local labor laws as specified in
document ref: LLC2022. The contractor is responsible for all site safety measures."
    },
    "vectorEmbedding": null
  }
 ]
}
```

**Key Fields Explained:**

- `tenderId`: Your internal unique ID for this tender.
- `unstructuredData`: This is the critical field for data that doesn't fit the schema. You can dump raw text blocks or even HTML snippets here, keyed by their original section title if available. This preserves all data from the source.
- `vectorEmbedding`: This field is a placeholder, initially `null`. After the data is saved, a separate background process can take the content from `procurementSummary` and `unstructuredData`, generate a vector embedding using a model like Sentence-BERT or OpenAI's APIs, and store it here as an array of floats. This enables powerful semantic search capabilities later.

## MongoDB Schema Suggestion

MongoDB is an excellent choice due to its flexible schema, which aligns perfectly with this use case. A document in your `tenders` collection would mirror the JSON output structure.

**Collection:** `tenders`

JavaScript

```
{
 "_id": ObjectId("..."), // MongoDB's native ID
 "tenderId": "tend_f4a8b1c9", // Your unique, indexed ID
 "sourceUrl": "https://example.gov.in/tenders/2025/infra/001",
 "scrapedTimestamp": ISODate("2025-08-29T16:30:00Z"),
 "country": "IN",
 "state": "Tamil Nadu",
 // ... all other fields from the JSON output ...
 "tenderDetails": {
  // ... nested object ...
 },
 "eligibilityRequirements": [
  // ... array of objects ...
 ],
```

```
  "proposalFormat": [
    // ... array of objects ...
  ],
  "unstructuredData": {
    "section_4_payment_terms": "...",
    "annex_c_special_conditions": "..."
  },
  "vectorEmbedding": [0.0123, -0.2345, ..., 0.9876] // Or null if not processed yet
}
```

**Recommended Indexes:**

To ensure fast queries, you should create indexes on the following fields:

- `tenderId` (Unique Index)
- `country` and `state` (Compound Index: `{ "country": 1, "state": 1 }`)
- `tenderDetails.closingDate`
- `tenderDetails.category` (Multikey Index)
- For keyword search: A **text index** on `tenderDetails.title` and `tenderDetails.procurementSummary`.
- For vector search: A **vector index** on `vectorEmbedding` (supported by MongoDB Atlas and other vector databases).

## Web Scraping Suggestions

Since the core of your service is scraping, here are some strategic suggestions.

1. **Hybrid Scraping Approach (Structured + Unstructured):**
    - **Define a "Golden Schema":** Identify the most common fields across all sources (e.g., Title, Closing Date, Authority, Reference Number).
    - **Scrape for Structure First:** Write your scrapers (spiders) to specifically target these fields using CSS selectors or XPath.
    - **Fallback to Unstructured:** For any section of the tender document that you cannot parse into a structured field (like "Special Conditions", "Payment Terms"), grab the entire block of text/HTML for that section. Store this content in the `unstructuredData` object with a key derived from the section's heading. This ensures zero data loss.
2. **Technology Stack:**
    - **Python:** The industry standard for web scraping.
    - **Scrapy Framework:** Excellent for managing large-scale, concurrent scraping projects. It handles requests, proxy management, and data pipelines efficiently.
    - **Beautiful Soup / lxml:** For parsing HTML and XML content.
    - **Playwright / Selenium:** Essential for modern websites that rely heavily on JavaScript to load content. Use these when simple HTTP requests don't return the full data.

3. **Overcoming Anti-Scraping Measures:**
    - **Proxy Rotation:** Use a reliable rotating proxy service (with residential and datacenter IPs) to avoid IP-based blocking.
    - **User-Agent Spoofing:** Rotate user-agent strings to mimic different browsers and devices.
    - **Headless Browsers with Human-like Behavior:** When using Playwright/Selenium, introduce random delays and mouse movements to avoid bot detection.
    - **CAPTCHA Solving Services:** Integrate services like 2Captcha or Anti-CAPTCHA for sites that are protected by CAPTCHAs.
4. **The Vector Embedding Pipeline:**
    - **Step 1 - Scrape & Store:** Your scraper runs, populates the MongoDB document, and leaves `vectorEmbedding` as `null`.
    - **Step 2 - Trigger a Worker:** Use a message queue (like RabbitMQ or Kafka) or a database trigger (like MongoDB Change Streams) to signal that a new tender has been added.
    - **Step 3 - Process & Embed:** A separate worker process picks up the new `tenderId`. It reads the relevant text fields (`procurementSummary`, `title`, and values from `unstructuredData`). It then uses a pre-trained language model to convert this text into a numerical vector.
    - **Step 4 - Update:** The worker updates the MongoDB document, populating the `vectorEmbedding` field with the generated vector. This decouples the time-consuming ML process from the scraping process, making your system more robust.