perplexity

# Next, you will design all API Routes and Endpoints

Excellent. Based on the established business logic and the database schemas, here is the comprehensive API design for the Enterprise Agentic Workflow Engine (EAWE).

The API is designed following RESTful principles, is secured with role-based access control (RBAC), and is inherently multi-tenant. All endpoints, unless otherwise noted, are scoped to the authenticated user's tenant ID to ensure data isolation.

**Base URL:** `/api/v1`

### 1. Authentication API

Handles user authentication and session management.

- **Endpoint:** `POST /auth/login`
  - **Description:** Authenticates a user and returns a JSON Web Token (JWT) for subsequent API calls.
  - **Request Body:**

```
{
  "email": "user@example.com",
  "password": "user_password"
}
```

  - **Response (200 OK):**

```
{
  "accessToken": "ey...",
  "user": {
    "id": "usr_...",
    "email": "user@example.com",
    "role": "OPERATOR"
  }
}
```

### 2. Workflow Definitions API

CRUD operations for workflow blueprints created in the Workflow Studio.

- **Required Role:** `DESIGNER`, `ADMINISTRATOR`
- **Endpoint:** `POST /workflows/definitions`
  - **Description:** Creates a new workflow definition.
  - **Request Body:**

```
{
  "name": "Quarterly Financial Report Generation",
  "description": "Pulls data from SAP and Salesforce to generate the QBR deck.",
  "definitionGraph": { ... } // JSON object representing the workflow nodes and e
}
```

  - **Response (201 Created):** Returns the newly created `WorkflowDefinition` object.
- **Endpoint:** `GET /workflows/definitions`
  - **Description:** Retrieves a paginated list of all workflow definitions for the tenant.
  - **Response (200 OK):** An array of `WorkflowDefinition` objects.
- **Endpoint:** `GET /workflows/definitions/{definitionId}`
  - **Description:** Retrieves a single workflow definition by its ID.
  - **Response (200 OK):** A single `WorkflowDefinition` object.
- **Endpoint:** `PUT /workflows/definitions/{definitionId}`
  - **Description:** Updates an existing workflow definition (e.g., change its name or update the graph).
  - **Request Body:** Similar to the `POST` request.
  - **Response (200 OK):** The updated `WorkflowDefinition` object.
- **Endpoint:** `DELETE /workflows/definitions/{definitionId}`
  - **Description:** Archives a workflow definition. A soft delete is performed.
  - **Response (204 No Content):**

### 3. Workflow Runs API

Endpoints for triggering and monitoring live workflow instances.

- **Required Role:** `OPERATOR`, `DESIGNER`, `ADMINISTRATOR`
- **Endpoint:** `POST /workflows/runs`
  - **Description:** Triggers a new run of a specified workflow definition. This is the primary endpoint for starting an automation process.
  - **Request Body:**

```
{
  "workflowDefinitionId": "def_...",
  "payload": { // Initial data to kick off the workflow
    "quarter": "Q3",
    "year": 2025
  }
}
```

  - **Response (202 Accepted):** Returns the newly created `WorkflowRun` object with a `PENDING` status.
- **Endpoint:** `GET /workflows/runs`

- **Description:** Retrieves a paginated list of all workflow runs for the tenant. Can be filtered by status (e.g., `?status=FAILED`).
- **Response (200 OK):** An array of `WorkflowRun` objects.

- **Endpoint:** `GET /workflows/runs/{runId}`
  - **Description:** Retrieves the current state and details of a specific workflow run.
  - **Response (200 OK):** A single `WorkflowRun` object.
- **Endpoint:** `GET /workflows/runs/{runId}/audit-logs`
  - **Description:** Retrieves the detailed, immutable audit trail for a specific workflow run.
  - **Response (200 OK):** An array of `AuditLog` objects.

### 4. Human-in-the-Loop (HITL) API

Manages tasks that require human intervention and approval.

- **Required Role:** `OPERATOR`, `DESIGNER`, `ADMINISTRATOR`
- **Endpoint:** `GET /hitl/tasks`
  - **Description:** Retrieves a list of all HITL tasks assigned to the authenticated user.
  - **Response (200 OK):** An array of `HITLTask` objects.
- **Endpoint:** `POST /hitl/tasks/{taskId}/approve`
  - **Description:** Approves a pending task, allowing the associated workflow to continue.
  - **Request Body:** (Optional)

    ```
    { "comment": "Looks good to me." }
    ```

  - **Response (200 OK):** The updated `HITLTask` object with `APPROVED` status.
- **Endpoint:** `POST /hitl/tasks/{taskId}/reject`
  - **Description:** Rejects a pending task, halting the associated workflow.
  - **Request Body:**

    ```
    { "rejectionReason": "The invoice amount does not match the purchase order." } //
    ```

  - **Response (200 OK):** The updated `HITLTask` object with `REJECTED` status.
- **Endpoint:** `POST /hitl/tasks/{taskId}/edit-and-approve`
  - **Description:** Allows a user to correct data processed by an agent and then approve it. This is a critical feedback mechanism for the learning loop.
  - **Request Body:**

    ```
    {
      "humanCorrectedOutput": { // The corrected data object
        "invoiceTotal": 1599.99
    ```

```
  }
}
```

- **Response (200 OK):** The updated `HITLTask` object with `EDITED_AND_APPROVED` status.

### 5. Conversational & Retrieval API

The interface for semantic search and conversational interaction.

- **Required Role:** `OPERATOR`, `DESIGNER`, `ADMINISTRATOR`
- **Endpoint:** `POST /query`
  - **Description:** The central endpoint for submitting a natural language query (text-based). The system will perform RAG to generate a synthesized answer.
  - **Request Body:**

```
{
  "query": "Find all proposals for Acme Corp from last quarter and summarize thei
}
```

  - **Response (200 OK):**

```
{
  "answer": "I found two proposals for Acme Corp from last quarter. Proposal #123
  "sources": [
    { "documentId": "doc_...", "fileName": "Proposal_123.pdf", "sourceUri": "..."
  ]
}
```

- **Endpoint:** `WS /query/voice`
  - **Description:** A WebSocket endpoint for real-time, streaming voice-to-text and response. The client streams audio data, and the server (using NVIDIA Riva) streams back transcribed text and the final synthesized answer.

### 6. Document Ingestion API

Manages data sources for the vector database.

- **Required Role:** `DESIGNER`, `ADMINISTRATOR`
- **Endpoint:** `POST /documents/ingest`
  - **Description:** Initiates the ingestion process for a new data source. The server will asynchronously process the source, chunk its contents, generate embeddings, and store them in Qdrant.
  - **Request Body:**

```
{
  "sourceType": "SHAREPOINT_FOLDER",
  "sourceUri": "https://tenant.sharepoint.com/sites/Finance/Shared%20Documents/In
}
```

    - **Response (202 Accepted):** Returns the newly created `Document` entry with an `INDEXING` status.
- **Endpoint:** `GET /documents`
    - **Description:** Lists all indexed data sources and their current status.
    - **Response (200 OK):** An array of `Document` objects.
- **Endpoint:** `DELETE /documents/{documentId}`
    - **Description:** Removes a document and all its associated vector embeddings from the system.
    - **Response (204 No Content):**

### 7. Credentials Management API

Securely manages credentials for third-party integrations.

- **Required Role:** `ADMINISTRATOR`
- **Endpoint:** `POST /integrations/credentials`
    - **Description:** Adds a new encrypted credential to the vault.
    - **Request Body:**

```
{
  "serviceName": "SALESFORCE",
  "type": "OAUTH2",
  "encryptedValue": "..." // Value is encrypted on the client-side or applicatior
}
```

    - **Response (201 Created):** Returns the credential metadata (ID, serviceName, type).
- **Endpoint:** `GET /integrations/credentials`
    - **Description:** Lists all configured credentials for the tenant **without** returning the secret values.
    - **Response (200 OK):** An array of credential metadata objects.
- **Endpoint:** `DELETE /integrations/credentials/{credentialId}`
    - **Description:** Deletes a credential from the vault.
    - **Response (204 No Content):**

⁂