BRD Analysis - 2025-08-13

Okay, here’s a breakdown of the provided API documentation, categorized for clarity and focusing on key information.

I. Overview

- **Project:** IDFC AI Platform
- **Purpose:** A platform for AI model management, application invocation, and guardrail enforcement.
- **Architecture:** Uses MongoDB as the primary database, leverages JWT authentication, and integrates with an external chat API.

II. API Endpoints & Functionality

A. Authentication

- **Endpoint:** `/api/v1/auth/login`
- **Method:** POST
- **Description:** Authenticates users.
- **Request:** JSON payload containing `username` and `password`.
- **Response:** JWT token, `access_token`, `token_type` (bearer), and `user_id` with assigned role.
- **Status Codes:** 200 (Success), 401 (Invalid credentials), 400 (Invalid JSON)

B. Model Management APIs

- **Endpoint:** `/api/v1/models/all_models`
- **Method:** GET
- **Description:** Retrieves all available AI models from MongoDB.
- **Headers:** `Authorization: Bearer `
- **Response:** Array of model objects, including metadata like `id`, `name`, `type`, `domain`, `pipeline`, and `status`.
- **Endpoint:** `/api/v1/models/provision`
- **Method:** POST
- **Description:** Creates and provisions new AI model instances (Admin only).
- **Authorization:** Requires a valid JWT token.

• **Request:** JSON payload containing model details like `id`, `name`, `type`, `domain`, `pipeline`, and `status`.

• **Response:** Provisioned model object with database ID, success on 200. 403 for non-admins, 409 if model ID exists.

C. Application Management APIs

• **Endpoint:** `/api/v1/apps/register`

• **Method:** POST

• **Description:** Registers new applications linked to AI models (Admin/Developer only).

• **Request:** JSON payload with application details (id, name, type, domain, pipeline, status). Auto-generates unique app IDs.

• **Response:** Provisioned app object with database ID.

• **Endpoint:** `/api/v1/apps/{app_id}/invoke`

• **Method:** POST

• **Description:** Executes application, forwarding requests to an external chat API.

• **Request:** Multi-part form data allowing for the submission of text content, model selection, file uploads (PDF, XLSX, JPG, PNG, MP4), and metadata like input type (text, pdf, etc.).

• **Request Headers:** `Content-Type: multipart/form-data`, `Authorization: Bearer `.

• **Response:** Response from the external chat API, including `request_id`, `timestamp`, and an output object containing the result of the execution.

• **Endpoint:** `/api/v1/guardrails/{user_id}`

• **Method:** GET

• **Description:** Retrieves all flagged responses for a specific user across all applications (Admin only).

• **Response:** Array of flagged response objects, each with `content` and `guardrail_info` (including the reason for flagging).

III. Supporting Information

• **Data Storage Architecture:** MongoDB

• **Security Features:** JWT authentication, bcrypt password hashing, input validation.

• **Error Format:** JSON with `detail` field.

• **File Upload Specs:** Maximum size (50MB), supported formats, automatic file type detection.

• **Documentation:** Swagger UI and ReDoc endpoints available for interactive API documentation.

IV. Quick Start & Setup

- **Login:** `curl -X POST "http://localhost:8001/api/v1/auth/login" ...`
- **Export Token:** `export TOKEN="your_access_token_here"`
- **Server Run:** `python -m uvicorn app.main:app --host 0.0.0.0 --port 8001 --reload`

- Key takeaways:** This API appears to be a well-structured system designed for managing AI models and their application invocation while incorporating safeguards (guardrails) to prevent misuse. The use of JWT authentication and detailed logging makes it suitable for a production environment.

Do you have a specific question about this documentation you'd like me to answer or a particular aspect you’d like me to elaborate on?