Use Case Description for Your Dashboard Application

Overview: This dashboard application features a REST backend and a simple UI frontend, with backen d documentation provided by OpenAPI. The application is designed to store simple data and files, whic h can be viewed and managed through the UI.

Key Functionalities:

1. **File Management**:
    - **Upload File**:
        - Endpoint: `POST /api/file`
        - Request Body: Accepts a binary file.
        - Response: Returns the uploaded file's ID.
    - **Retrieve All Files**:
        - Endpoint: `GET /api/file`
        - Response: Returns a list of all files stored in the system.
    - **Retrieve Specific File**:
        - Endpoint: `GET /api/file/{fileId}`
        - Parameters: `fileId` (string)
        - Response: Returns the content of the specified file.
    - **Delete Specific File**:
        - Endpoint: `DELETE /api/file/{fileId}`
        - Parameters: `fileId` (string)
        - Response: Confirms file deletion.
2. **Data Management**:
    - **Create New Data Entry**:
        - Endpoint: `POST /api/data/{domain}/{category}/{entry}`
        - Parameters: `domain`, `category`, `entry` (all strings)
        - Request Body: Data structure defined by `DataRequest`.
        - Response: Confirms data creation.
    - **Delete Data Entry**:

        - Endpoint: `DELETE /api/data/{domain}/{category}/{entry}`
        - Parameters: `domain`, `category`, `entry` (all strings)
        - Optional Query: `softDelete` (default `true`)
        - Response: Confirms data deletion.
    - **Retrieve All Data Entries**:
        - Endpoint: `GET /api/data`
        - Response: Returns a list of all data entries.
    - **Retrieve Domain-Specific Data**:
        - Endpoint: `GET /api/data/{domain}`
        - Parameters: `domain` (string)
        - Response: Returns a list of data entries for the specified domain.
    - **Delete Category**:
        - Endpoint: `DELETE /api/data/{domain}/{category}`
        - Parameters: `domain`, `category` (both strings)
        - Response: Confirms category deletion.
3. **Data Hierarchy Management**:
    - **Create Data at Different Levels**:
        - Multiple endpoints to create hierarchical data entries at various levels, specified by path parameters such as `level0`, `level1`, etc.

Example Scenarios:

1. **Uploading a File**:
    - A user uploads a new file using the UI, which triggers a `POST` request to `/api/file`. The backend stores the file and returns an ID for future reference.
2. **Viewing All Files**:
    - On the dashboard, a user clicks to view all files, which triggers a `GET` request to `/api/file`. The UI displays the list of files returned by the backend.
3. **Creating a Data Entry**:

- A user inputs new data into the form and submits it. This action triggers a `POST` request to `/api/data/{domain}/{category}/{entry}`, storing the data in the backe nd.

4. **Deleting a Specific Data Entry**:
    - From the UI, the user selects a data entry to delete. This sends a `DELETE` request to `/api/data/{domain}/{category}/{entry}`, which removes the data.

Use Case Description for Your Dashboard Application

Overview: Your application features a REST backend, complete with OpenAPI documentation, and a user-friendly UI frontend. This setup allows for efficient storage and retrieval of simple data and files. Users can interact with the data and files through the UI, ensuring a smooth and integrated experience.

Key Functionalities:

1. **File Management**:
    - **Upload File**: Users can upload files directly from the UI. When a file is uploaded, a `POST` request is sent to the `/api/file` endpoint. The backend processes this request, stores the file, and returns an identifier (ID) for the uploaded file. This ID can be used later to retrieve or manage the file.
    - **Retrieve All Files**: Users can view a list of all uploaded files. This is done through a `GET` request to the `/api/file` endpoint. The backend responds with a list of all stored files, allowing users to see what has been uploaded at a glance.
    - **Retrieve Specific File**: If a user wants to access the content of a specific file, they can do so via a `GET` request to the `/api/file/{fileId}` endpoint, where `fileId` is the identifier of the file they want to retrieve. The backend will return the file's content.
    - **Delete Specific File**: Users can delete a specific file using a `DELETE` request to the `/api/file/{fileId}` endpoint. This operation will remove the file from the storage, ensuring it is no longer accessible.
2. **Data Management**:
    - **Create New Data Entry**: Users can create new data entries by sending a `POST` request to the `/api/data/{domain}/{category}/{entry}` endpoint. The request includes the necessary parameters—`domain`, `category`, and `entry`—and the data to be stored. The backend processes this information and confirms the data entry creation.
    - **Delete Data Entry**: Users can delete a specific data entry by sending a `DELETE` request to the `/api/data/{domain}/{category}/{entry}` endpoint. This request requires the `domain`, `category`, and `entry` parameters. An optional `softDelete` query parameter can be included, with `true` being the default value, allowing for a less permanent deletion.
    - **Retrieve All Data Entries**: By sending a `GET` request to the `/api/data` endpoint, users can retrieve a list of all data entries stored in the system. This helps in keeping track of all available data.

- **Retrieve Domain-Specific Data**: Users can filter data based on the domain by sending a `GET` request to the `/api/data/{domain}` endpoint. This will return all data entrie s associated with the specified domain.
- **Delete Category**: To delete a category, users can send a `DELETE` request to the `/api/data/{domain}/{category}` endpoint, specifying the domain and categor y to be deleted. The backend will confirm the deletion.

3. **Data Hierarchy Management**:
    - **Create Data at Different Levels**: The application supports the creation of hierarchical d ata entries at multiple levels. Various endpoints allow users to specify different levels (e. g., `level0`, `level1`, etc.) within the path parameters. For example, creating data at `level0` would involve a `POST` request to the `/api/data/{domain}/{category}/{entry}/{level0}` endpoint, and similarly for other levels.

Example Scenarios:

1. **Uploading a File**:
    - A user wants to store a new document in the system. They navigate to the file upload sec tion in the UI, select their file, and click upload. This action triggers a `POST` request to `/api/file`. The backend receives the file, stores it, and returns an ID, which the user can use to access this file in the future.
2. **Viewing All Files**:
    - On the dashboard, a user clicks on a button to view all files. This action sends a `GET` re quest to `/api/file`. The backend returns a list of all files, which are then displayed o n the UI, allowing the user to see all available files.
3. **Creating a Data Entry**:
    - A user needs to add a new piece of data. They fill out the required fields in the UI and su bmit the form. This sends a `POST` request to `/api/data/{domain}/{category}/{entry}` with the provided data. The backend processes this request a nd stores the new data entry.
4. **Deleting a Specific Data Entry**:
    - From the UI, the user selects an existing data entry they wish to delete and confirms the deletion. This action sends a `DELETE` request to `/api/data/{domain}/{category}/{entry}`. The backend deletes the specified data entry, removing it fr om the storage.