Document Management API Documentation

Overview

The Document Management API provides endpoints for creating, updating, retrieving, and acknowledging documents within a multi-tenant system. It includes version control to retain and label previous document versions when updates are made, ensuring old files remain accessible.

Key Components

Models

- **Document**: Stores the latest document metadata, including `title`, `file_url`, `file_path`, `file_type`, `file_size`, `version`, `tenant_id`, `uploaded_by_id`, `updated_by_id`, `uploaded_at`, `updated_at`, `expiring_date`, `status`, and `document_number`.
- **DocumentVersion**: Stores version-specific data for each document, including `version`, `file_url`, `file_path`, `file_type`, `file_size`, `created_at`, and `created_by_id`. Linked to `Document` via a foreign key.
- **DocumentAcknowledgment**: Tracks user acknowledgments of documents, including `document`, `user`, `tenant`, and `acknowledged_at`.

Serializers

- **DocumentSerializer**: Handles serialization for `Document`, including file validation (PDF/image, <10MB), tenant validation, and user metadata fetching from an external auth service. Manages creation and updates with version control.
- **DocumentVersionSerializer**: Serializes `DocumentVersion` records, including `created_by` user details.

- **DocumentAcknowledgmentSerializer**: Serializes acknowledgment data, ensuring tenant and user validation.

Views

- **DocumentListCreateView**: Handles `GET` (list documents) and `POST` (create new document) requests.
- **DocumentDetailView**: Handles `GET` (retrieve document), `PATCH` (update document), and `DELETE` (delete document) requests.
- **DocumentVersionListView**: Handles `GET` requests to retrieve all versions of a document.
- **DocumentAcknowledgeView**: Handles `POST` requests to acknowledge a document.

Endpoints

1. List/Create Documents

- **URL**: `/documents/`
- **Methods**:
  - **GET**: Lists all documents for the authenticated tenant.
  - **POST**: Creates a new document with an uploaded file.
- **Permissions**: Requires `IsAuthenticated` and `IsAdminUser`.
- **Behavior**:
  - Creates a new `Document` with `version=1`.
  - Uploads file to storage with `_v1` appended (e.g., `policy_v1.pdf`).
  - Creates a corresponding `DocumentVersion` record.
- **Response**:
  - `GET`: `200 OK` with list of documents.
  - `POST`: `201 Created` with document data, or `400 Bad Request` for validation errors.

2. Document Details

- **URL**: `/documents/<id>/`

- **Methods**:
  - **GET**: Retrieves a specific document.
  - **PATCH**: Updates a document, optionally replacing the file.
  - **DELETE**: Deletes a document.
- **Permissions**: Requires `IsAuthenticated` and `IsAdminUser`.
- **Behavior (PATCH)**:
  - Saves current file details to a new `DocumentVersion` record.
  - Increments `version` and uploads new file with `_v<version>` (e.g., `policy_v2.pdf`).
  - Updates `Document` with new file details and creates a new `DocumentVersion`.
- **Response**:
  - `GET`: `200 OK` with document data.
  - `PATCH`: `200 OK` with updated document data.
  - `DELETE`: `204 No Content`.

3. List Document Versions

- **URL**: `/documents/<document_id>/versions/`
- **Method**: `GET`
- **Permissions**: Requires `IsAuthenticated` and `IsAdminUser`.
- **Behavior**: Retrieves all `DocumentVersion` records for a document, including file URLs and version numbers.
- **Response**: `200 OK` with list of versions (e.g., `[{version: 1, file_url: ".../policy_v1.pdf"}, {version: 2, file_url: ".../policy_v2.pdf"}]`).

4. Acknowledge Document

- **URL**: `/documents/<document_id>/acknowledge/`
- **Method**: `POST`
- **Permissions**: Requires `IsAuthenticated`.
- **Behavior**: Creates a `DocumentAcknowledgment` record if the user hasn't acknowledged the document.
- **Response**: `201 Created` with acknowledgment data, or `400 Bad Request` if already acknowledged.

Version Control

- **Creation**: New documents start with `version=1`. A `DocumentVersion` record is created with the file details.
- **Update**: When updating with a new file:
  - The current file is saved as a `DocumentVersion` with the current `version`.
  - The `version` is incremented.
  - The new file is uploaded with `_v<version>` in the name.
  - A new `DocumentVersion` record is created for the updated file.
- **Retrieval**: Use `/documents/<document_id>/versions/` to access all versions, each labeled with its version number and file URL.

Notes

- **File Storage**: Files are stored with versioned names (e.g., `policy_v1.pdf`, `policy_v2.pdf`) to ensure old versions remain accessible. No automatic deletion of old files occurs.
- **Acknowledgments**: Updates to documents may require re-acknowledgment by users (not implemented in current code).
- **Database**: Ensure migrations are applied for the `DocumentVersion` model. Index `DocumentVersion.document` and `DocumentVersion.version` for performance.
- **Storage Costs**: Retaining all versions increases storage usage. Consider cleanup policies for old versions if needed.