Media Library Platform with Background Processing

Project Objective

Build a secure backend platform where users can register, log in, and upload files (images, audio, video, PDF, docs). The platform must allow users to manage, search, and categorize files; automatically extract and store basic metadata; process uploads in the background (using Celery/RQ/async tasks); and expose a complete REST API for frontend consumption.

Core Features

User Authentication

- User registration, login, and JWT-based authentication (Django Rest Framework or FastAPI).
- Password reset functionality.

Media Upload and Management

- Authenticated users can upload multiple file types (image, video, audio, pdf, doc).
- Files stored on local disk with path stored in the DB.
- Save file metadata (filename, size, type, extension, upload timestamp, owner).
- Each file can be given a category/tag (e.g., “work”, “personal”, “projectX”).
- Support renaming, deleting, and re-categorizing files.

Background Metadata Extraction

- When a file is uploaded, extract metadata (using a background worker):
- - Images: width, height, format (use Pillow).
- - Audio/Video: duration, format, basic codec info (use ffprobe or mediainfo).
- - PDF/Docs: page count (use PyPDF2 or similar).
- Store extracted metadata in the database.
- User can query for upload status and metadata (processing should not block upload API).

RESTful API

- CRUD endpoints for users, files, categories/tags.
- API for listing/searching/filtering files by type, category, owner, date, etc.
- API endpoint to download files.
- Return metadata as part of file details.

Admin Panel

- Provide admin interface to view/manage users, files, and categories (Django admin or minimal FastAPI-admin).

Reporting & Export

- Endpoint to export a filtered list of files and their metadata as a CSV report.

Security & Validation

- Strong file validation (allowed extensions, size limit, duplicate detection).
- Handle edge cases (corrupt files, failed metadata extraction).
- API rate limiting or throttling for uploads.

Testing & Documentation

- Unit tests for major modules.
- API docs using Swagger/OpenAPI or detailed markdown.
- Well-structured README with setup instructions.

Bonus Features

- Implement user activity logs (file uploaded, deleted, renamed).
- Allow sharing files between users (with permission control).
- Tag-based file search and recommendations.
- (Optional) Simple dashboard (API endpoint) for per-user file statistics (count by type/category).

Evaluation Criteria

- Clean, modular code and architecture.
- Secure, robust, and well-documented API.
- Proper async/background processing for metadata.
- Comprehensive testing and documentation.
- Git usage and commit hygiene.

Submission

Public GitHub repo with clear instructions.
(Optional) Short demo video/screenshots.