Full-Stack Intern Assignment:

Mini Document Manager

Objective

Design and implement a minimal document manager that supports uploading, listing, searching, and downloading documents.

This assignment evaluates:

- API and data design
- Full-stack thinking (frontend + backend)
- Ability to explain tradeoffs and constraints

⏱️ Expected time: ~90 minutes
⚠️ We will not run your code locally

What You Need to Build (Minimum Scope)

1. Upload Documents

- Users can upload multiple documents in one action.
- Each document has:
  - Title (can default to filename)
  - File
- Backend API must support multiple-file upload.

Files may be stored locally (disk). Cloud storage is out of scope.

2. List Documents

Show a list of uploaded documents with:

- Title
- File size
- Upload date
- Download action

Backend must support:

- Pagination (`page`, `pageSize`)
- Sorting (by upload date)
- Text search on document title (simple "contains" search)

Frontend must include:

- Search input
- Sort control
- Pagination

3. Download Document

- User can download a document.
- Backend must return the file via a streaming response (do not load the full file into memory).

Out of Scope- Not Required

These will be evaluated via questions only:

- Authentication / OAuth
- Cloud storage (S3, GCS, etc.)
- Fuzzy search
- Bulk ZIP downloads
- File previews
- Background jobs / queues
- Antivirus scanning

Technical Expectations

- Clear separation between:
    - File storage (binary data)
    - Metadata storage (title, size, timestamps)
- Clean API contracts
- Basic validation and error handling
- Code should be readable and reasonably structured

Perfection is not expected. Clear thinking is.

APIs (Guidance Only)

You do not need to match these exactly, but equivalent functionality must exist:

- POST /documents
  Upload one or more documents
- GET /documents
  Supports page, pageSize, sortOrder, q (search)

- GET /documents/{id}/download
  Streaming download

Design Questions (Mandatory)

Add a section in your README titled “Design Questions” and answer briefly.

1. Multiple Uploads

How does your system handle uploading multiple documents?

- One request or many?
- Any limits or tradeoffs?

2. Streaming

Why is streaming important for upload/download?
What problems occur if the server loads the full file into memory?

3. Moving to S3

If files move to object storage (e.g., S3):

- What changes in your backend?
- Would the backend still handle file bytes?

4. Frontend UX

If you had more time:

- How would you add document preview?
- How would you show upload progress?

Submission Instructions

Send your submissions to internship@finfully.co

⚠️ We will NOT run your code

Your submission must be understandable without execution.

Please submit:

1. GitHub repository (or zip) containing frontend + backend code
2. Architecture diagram (PNG/PDF)
   - Show frontend, backend, APIs, storage
   - Include upload and download flow
3. Screen recording (5 minutes max)
   In one continuous recording, Demo the UI (upload, list, search, download)
   Send google drive or one drive link, don't send video file on email
4. README.md including:
   - Setup assumptions
   - Key tradeoffs due to time limit
   - Answers to Design Questions

Optional (nice to have, not required):

- Screenshots
- Tests (even one is fine)

Evaluation Criteria

We will evaluate based on:

- Clarity of system design
- Quality of API and data modeling
- Correctness of approach
- Ability to explain decisions
- Frontend state handling (loading, empty, error)

Completing fewer things well and explaining them clearly is better than incomplete breadth.

Final Note

Using AI tools is allowed.
However, you must be able to explain everything you submit.