DealMover Case Study

Assignment

Recreate a simplified DealMover workflow: **extract key financial values from a 10-K and display them in a grid**.

Part 1 — Backend (Django)

- Build an endpoint /api/extract/ that accepts:
  - A **PDF upload** (file, required).
  - A **period_end_date** (optional, YYYY-MM-DD).
- Parse the PDF to extract:
  - **Revenue**
  - **Cost of sales**
- Normalize values:
  - Remove $ symbols and spaces.
  - Convert (X) to -X.

```
Return JSON:
{
 "period_end_date": "2024-12-31",
 "results": {
  "revenue": "350018",
  "cos": "146306"
 }
}
```

Part 2 — Frontend (React)

- Build a small React component that:
  - Lets the user upload a PDF and (optionally) enter a period end date.
  - Sends the data to your backend.
  - Displays results in a **spreadsheet-like table** with two rows: Revenue and Gross Profit.

Part 3 — Stretch Goals (Optional)

- Extract one more field (e.g., **Operating Income**).
- Add **tests** for your parser (pytest).
- Show a **header row** in the grid with the statement period.

Submission Rules

/backend (Django app)

/frontend (React app)

README.md (instructions)

AI_USAGE.md (prompts/tools disclosure)

DECISIONS.md (assumptions & trade-offs)

tests/ (if added)

AI Use Policy

- You may use ChatGPT, Copilot, or similar tools.
- You must disclose:
    - Which tools you used
    - Your exact prompts
    - Where you accepted/rejected outputs
- Place this in AI_USAGE.md.

Commit History

- Use multiple commits (not one giant commit).
- Write short, meaningful messages.

Deliverables

- Working backend + frontend.
- Ability to run locally with pip install -r requirements.txt and npm install && npm run dev.
- Documentation of assumptions in DECISIONS.md.

Directory Tree Example

```
dealmover-intern-case/
├── backend/
│   ├── manage.py
│   ├── requirements.txt
│   ├── dealmover_case/
│   │   ├── __init__.py
│   │   ├── settings.py
│   │   ├── urls.py
│   │   └── wsgi.py
│   └── core/
│       ├── __init__.py
│       ├── apps.py
│       ├── urls.py
│       └── views.py
└── frontend/
    ├── index.html
    ├── package.json
    ├── tsconfig.json
    ├── vite.config.ts
    └── src/
        ├── main.tsx
        ├── App.tsx
        └── components/
            └── ResultsGrid.tsx
```

Live Session

Extend the Parser

- Add one new field (e.g., Operating Income).
- Handle a label variation (e.g., “Total Revenues” or “Revenues” instead of “Consolidated Revenues”).

Explain Your Code

- Walk line-by-line through _extract_values_from_text.
- Explain trade-offs (regex vs table parsing, assumptions about formatting).

Evaluation Criteria

Criteria	Weight	Description
Understanding & Reasoning	**25%**	Do you understand your own solution and trade-offs?
Parser Quality	**25%**	Does it handle negatives, label variants, basic edge cases?
Tests & Validation	**15%**	Did you add tests or checks for robustness?
Frontend Integration	**10%**	Clean flow from upload → extract → display.
Code Quality (15%)	**15%**	Readability, naming, structure.
Live Adaptability	**10%**	Can you extend/fix code quickly without AI help?