Contract Summary Tool - Take-Home Assessment

Overview

In this take-home project, you'll be building an interface for a Patchbay user to be able to...

- Upload a document with record deal terms.
- See these key headline terms extracted into editable fields
- Save the newly created record deal as a form.

Attached is a sample contract which we briefly discussed in our conversation.

Data Model

These are some of the fields we use on our data models - use these as a guide to build out your fields and feel free to mock the data using the information provided.

```
{
  "active": true,
  "artist": "string",
  "book_number": "string",
  "credits": [],
  "instruments": [],
  "work_for_hire": false,
  "executed_contract": "string",
  "execution_date": "2024-12-19",
  "fee_amount": 0,
  "fee_currency": "USD",
  "fee_schedule": "100% Up Front",
  "kill_fee_amount": 0,
  "kill_fee_currency": "EUR",
  "masters": [],
  "payment_status": "Unpaid",
  "percent_recoupable": 0,
  "priority": 1,
  "producer": "string",
  "status": "Counterparty Action",
  "type": "Producer",
  "notes": "string",
  "original_file_name": "string",
  "start_date": "2024-12-19"
}
```

In your deliverables we'll be looking at how you make UI/UX decisions with ambiguity, while maintaining frontend principles (state management, passing props, etc) that create a smooth user experience.

Include a readme that explains how to run your code and briefly discuss your decisions and what you'd do to knock this feature out of the park if you had more time.

Core Features

1. Upload & Summary

- Users can upload a contract document (PDF/DOCX)
- System displays extracted key information:
  - Deal Type (e.g., Producer, Mixer, Record)
  - Artist Name
  - Fee Amount
  - Start Date
  - Work for Hire Status

2. Review & Edit

- Users can review the extracted information
- Edit any incorrect fields
- Form validation for required fields

3. Save

- Users can save the form as a newly created record deal.

Technical Requirements

Must Use..

- React
- TypeScript
- Form validation
- Error handling for null/undefined values

What to Mock

- Document processing/extraction - Just simulate a 2-second delay

- Backend API - Use local state

Deliverables

1. Working Application

- Source code in a GitHub repository
- Instructions to run locally

2. Demo Video

- A screen-recording of the application with you walking through core functionality, explaining some UI/UX decisions and what you'd improve with more time.

3. Read-me Document

Brief write-up explaining:

- Component structure
- State management approach
- Error handling strategy
- UI/UX decisions
- What you would improve with more time

Time Expectation

- 8-10 hours
- Focus on core functionality over perfect styling
- Document any shortcuts taken due to time constraints

Evaluation Criteria

- Code organization
- TypeScript usage
- Error handling
- Form validation
- UI/UX intuitiveness
- Documentation quality
- Effective communication in demo video