Founding Engineer Technical Exercise

Proposal Ingestion → Invitation to Bid Automation

Context

Bridgeline Technologies is building AI-powered software to eliminate manual workflows in pre-construction. One of the most painful tasks for General Contractors is extracting subcontractor contact information from proposals (PDFs, spreadsheets, emails) and manually re-entering it into Invitation to Bid (ITB) systems.

This exercise simulates a real problem we solve.

Objective

Build a **simple web application** that:

1. Accepts a set of subcontractor proposals
2. Extracts key contact information
3. Automatically populates that data into an **Invitation to Bid** interface
4. Information about invitation to bid can be found here: https://en.wikipedia.org/wiki/Invitation_to_tender

The goal is not perfection—it's to demonstrate how you think, architect, and ship.

Input

You will be provided with:

- A folder of **mock subcontractor proposals** (mixed formats)
  - PDFs
  - Excel files
  - (Optional) plain text or email-style docs

Each proposal may contain:

- Company name
- Contact name
- Email address
- Phone number
- Trade / scope

Expect inconsistency and messiness—this is intentional.

Requirements

1. Proposal Upload

- A basic UI that allows users to upload one or more proposal files
- Files should be processed server-side

2. Data Extraction

- Extract the following fields where possible:
  - Company Name
  - Contact Name
  - Email
  - Phone
  - Trade / Scope
- You may use:
  - Rule-based parsing
  - OCR
  - LLMs
  - Or any hybrid approach
- Accuracy matters, but **transparency matters more**
  - If confidence is low, show it

3. Review & Edit Step

- Display extracted data in a simple table or form
- Allow the user to:
  - Edit fields
  - Confirm or reject entries
- This mirrors real pre-construction workflows

4. Invitation to Bid Creation

- Once confirmed, populate the data into a mock **Invitation to Bid** screen:
  - Subcontractor list
  - Contact info
  - Trade assignment
- No need to send real emails—just show the populated state

Technical Expectations

- Use any modern web stack you're comfortable with
- Clean, readable, production-minded code
- Reasonable data models
- Thoughtful error handling
- Clear README explaining:
    - Architecture choices
    - Tradeoffs
    - What you'd improve with more time

What We're Evaluating

We care less about polish and more about **how you think**:

- How you handle unstructured, real-world data
- Product intuition (what the user needs vs. what's technically interesting)
- Ability to make pragmatic tradeoffs
- Code clarity and extensibility
- How you communicate assumptions and limitations

Bonus (Optional)

- Confidence scoring on extracted fields
- Deduplication of subcontractors
- Handling multiple contacts per company
- Notes on how this would scale in production