SDE Assignment – AI-Powered RFP Management System

1. Context

Many companies run procurement through **Requests for Proposal (RFPs)**: they define what they want to buy, email that out to multiple vendors, collect responses (often messy emails and attachments), and then someone manually compares all the quotes to decide who to award the work to.

This process is:

- Slow and error-prone
- Full of unstructured data (emails, PDFs, free-form descriptions)
- Repetitive: similar RFPs, similar evaluations, similar comparisons

We want to explore how you would design and build an **AI-powered RFP management system** that streamlines this workflow end-to-end.

---

2. Problem Statement

Design and implement a **single-user web application** that helps a procurement manager:

1. **Create RFPs**
    a. A user should be able to describe what they want to buy in **natural language**.
    b. Your system should turn this into a **structured representation** of an RFP that can be reused throughout the workflow (for sending to vendors, storing in a database, comparison, etc.).
    c. Example chat interaction for RFP creation: *"I need to procure laptops and monitors for our new office. Budget is $50,000 total. Need delivery within 30 days. We need 20 laptops with 16GB RAM and 15 monitors 27-inch. Payment terms should be net 30, and we need at least 1 year warranty."*
2. **Manage vendors and send RFPs**
    a. Maintain some form of **vendor master data** (who the vendors are and how to contact them).
    b. Allow the user to choose which vendors to send an RFP to.
    c. Send the RFP to vendors via **email**.
3. **Receive and interpret vendor responses**
    a. Support **inbound email**, where vendors reply with their proposals.
    b. Vendor responses can be **messy**: free-form text, tables, maybe attachments, etc.
    c. Use **AI** to extract the important details from these responses (e.g. prices, terms, conditions) and update your system automatically, without the user manually keying in each number.
4. **Compare proposals and recommend a vendor**
    a. For a given RFP, show how different vendors compare (e.g. pricing, terms, completeness of response).

b. Use **AI** to help the user evaluate: summaries, scores, or recommendations are all acceptable.
c. The system should help the user answer: *"Which vendor should I go with, and why?"*

You are **free to choose**:

- How you structure an "RFP" internally (fields, schema, etc.).
- How the **UI/UX** looks (chat-like, forms, wizards, dashboards, etc.).
- How you design the **scoring / comparison logic**.
- Which AI provider / model(s) you use and how you prompt them.

We care much more about **your decisions and reasoning** than about matching any specific reference implementation.

---

3. Requirements & Expectations

Functional (high-level)

At a minimum, your solution should demonstrate:

- A way to create and view **structured RFPs** driven by **natural language input**
- Basic **vendor management** (store vendors, pick them for an RFP)
- **Email sending** of RFPs to selected vendors
- **Email receiving** of at least one vendor response, and **automatic parsing** of that response with AI
- A **comparison view** for proposals for a given RFP, with some form of AI-assisted evaluation or recommendation

How you achieve each of these is up to you.

Technology

- Use a **modern web stack**.
  - Suggested: angular (frontend) and Django (backend),
- Use a **database** of your choice to persist data (MongoDB,).
- Integrate with a **real email system** for send + receive. This can be via SMTP, IMAP, or any email API/service.

AI Integration

Use an LLM (OpenAI) for **at least**:

- Turning natural language procurement needs into some structured representation of an RFP
- Parsing vendor responses (email body and/or attachments) into structured data
- Helping with proposal comparison and/or recommendations (e.g. explanations, summaries, scores)

You decide where and how heavily to use AI, and how you design prompts / flows.

Non-Goals (Out of Scope)

You do **not** need to implement:

- User authentication / multi-tenant support
- Real-time collaboration
- Email tracking (opens, clicks, etc.)
- A full RFP edit lifecycle with approvals, versioning, etc.

Focus on a solid, end-to-end **single-user** experience.

What We're Evaluating

We will look at:

- **Problem understanding & modeling** – how you model RFPs, vendors, proposals, and their relationships
- **Architecture & code quality** – structure, separation of concerns, naming, error handling
- **API & data design** – clarity and consistency of your APIs and data models
- **AI integration** – how thoughtfully and effectively you use AI, not just "call an API"
- **UX** – can a user actually complete the workflow without confusion?
- **Assumptions & reasoning** – are your assumptions reasonable and clearly documented?

---

4. Deliverables

GitHub Repository

- Public repo with clear structure (e.g. `/frontend`, `/backend`).
- `.env.example` listing all required environment variables (no secrets).

README (Mandatory)

Include at least:

1. **Project Setup**
    a. Prerequisites (Node version, DB, API keys).
    b. Install steps (frontend & backend).
    c. How to configure email sending/receiving.
    d. How to run everything locally.
    e. Any seed data or initial scripts.
2. **Tech Stack**
    a. Frontend, backend, DB, AI provider, email solution, and key libraries.
3. **API Documentation**
    a. Main endpoints: method + path, request body/params, example success & error responses.
4. **Decisions & Assumptions**
    a. Key design decisions (models, flows, scoring, etc.).
    b. Assumptions you made (about emails, formats, limitations, etc.).

5. **AI Tools Usage**
    a. Which AI tools you used while building (Copilot, ChatGPT, Claude, Cursor, etc.).
    b. What they helped with (boilerplate, debugging, design, parsing ideas, etc.).
    c. Any notable prompts/approaches.
    d. What you learned or changed because of these tools.

A