**American AI Logistics: Engineering Technical Assessment**

Email Module Improvements

# Objective:

This assessment evaluates your ability to design and engineer a new feature for an AI-powered government contracting platform. Your solution should incorporate large language model (LLM) capabilities to intelligently parse email bodies and enhance user productivity through data-aware UI features. You’ll also be asked to demonstrate thoughtful software architecture, extensibility, and use of design patterns.

# Background:

Our product currently allows users to receive and respond to bid opportunities. Many of these come via unstructured email bodies and attachments. Users also maintain a supplier directory, but there’s no current way to auto-prefill or recommend suppliers when composing responses to bids.

We aim to improve two core capabilities:  
 1. Intelligent Email Parsing: Use LLMs to extract relevant fields (e.g., items requested, quantities, deadlines) from the email body and attachments.  
 2. Supplier Autofill in Emails: Enable smart supplier suggestions or autofill based on the items present in the parsed email and existing supplier catalog data.

We also want to see how you would design this in a scalable, maintainable way—especially considering that bid sources (emails, portals, etc.) may vary and evolve.

# Your Task:

Design and describe an engineering solution that:  
 1. Uses an LLM to read the email body and extract structured bid-related data such as items, quantities, due dates, and part numbers.  
 2. Matches these items against saved supplier data and enables autofill or dropdown suggestions for supplier names and contact details within the email compose UI.  
 3. Ensures that parsing works across multiple email formats (e.g., plain text, HTML, forwarded chains).  
 4. Can scale across different types of incoming bids (e.g., different agencies, formats, or vendor submission styles).  
 5. Follows solid architectural design practices that isolate bid source variability, making it easy to add or modify extraction logic for new sources without breaking other parts of the codebase.  
 6. Demonstrates good use of design patterns, object-oriented principles, and unit testability.

# Deliverables:

Submit a 2–4 page document including:  
 - System Design Overview: Architecture diagram and narrative description of how email parsing, supplier matching, and UI interaction will work.  
 - LLM Interaction Strategy: How will the email content be extracted, cleaned, and sent to the LLM? What’s the expected format of the returned structured data?  
 - Supplier Matching Logic: How will you efficiently and accurately match extracted items to existing suppliers? How will the matching be surfaced in the UI?  
 - Bid Source Abstraction Design: How will you design your code to accommodate multiple bid sources with potentially different field requirements and parsing logic? What class/interface structure would you use?  
 - Code Samples: Provide brief (but illustrative) pseudocode or real code examples, in the language of your choice, for:  
 - Parsing and classifying an email body  
 - Matching suppliers to bid line items  
 - An abstract interface or factory method pattern to support different bid sources  
 - Frontend Enhancement Mockup: Add a sketch or mockup of how the supplier autofill dropdown might look when composing an email. (Simple drawing or design tool output is acceptable.)

# Evaluation Criteria:

- Robustness and realism of the email parsing and supplier matching logic  
 - Elegance and extensibility of your architecture  
 - Demonstrated use of clean, testable, object-oriented design  
 - Consideration of scalability and ease of future feature integration  
 - UI/UX thoughtfulness in mockup

# Submission Instructions:

Please return your completed document in either Word or PDF format. You may use LLMs and code generation tools to assist your thinking or generate diagrams, but make sure the final work reflects your own engineering judgment and clarity.