Hiring Task

AI-Powered Mail Web Application

Build a mail app where the AI assistant controls the UI

Duration: **5 Calendar Days**   •   Difficulty: **Intermediate – Advanced**

Overview

Build a **mail web application** with an integrated AI assistant that can **control the UI programmatically** — composing emails, navigating between views, displaying filtered results, and interacting with the interface on the user's behalf.

The assistant doesn't just answer questions — it **paints the UI, fills forms, and executes actions** when instructed by the user through natural language.

What to Build

1. Mail Client

A functional email client connected to a real mail provider (Google, Microsoft, or any provider of your choice):

- **Inbox** — List of received emails (sender, subject, preview, date). Click to read.
- **Sent** — List of sent emails. Click to read.
- **Compose** — Write and send an email (To, Subject, Body at minimum).
- **Email Detail** — Read the full content of an email.

2. Real-Time Mail Sync

New emails should appear in the inbox without a manual refresh. Use push notifications from your chosen provider (e.g. Pub/Sub, webhooks, subscriptions — whatever fits your stack).

3. AI Assistant — The Core of This Task

The app must have an **assistant panel** (sidebar, popup, or embedded — your design choice) that can **control the UI through natural language**. This is the primary evaluation area.

The assistant must be able to:

Compose & Send

> **User:** *"Send an email to john@example.com with subject 'Meeting Tomorrow' and body 'Let's meet at 3pm'"*

- The compose view opens

- The fields visibly fill in (the user can see it happening)
- The user clicks Send, or the assistant sends it (either approach is fine)

Search & Display

> **User:** *"Show me emails from the last 10 days"*

> **User:** *"Find the email from Sarah about the project update"*

- The assistant queries emails with the right filters
- **The main UI updates** to show the results (not just a text response in the chat)

Navigate & Open

> **User:** *"Open the latest email from David"*

- The assistant navigates to and displays that specific email in the detail view

Context Awareness

> **User:** *"Reply to this" (while reading an email)*

- The assistant knows which email is open and pre-fills a reply

Filters via Assistant

> **User:** *"Show only unread emails from this week"*

- The inbox filters accordingly

4. Filters

Basic filtering — by date range, sender, keyword, read/unread status. Should work both through:

- **UI controls** (dropdowns, date pickers, etc.)
- **Assistant commands** (natural language)

What We're Evaluating

#	Criteria	Weight	Priority
1	Mail integration works — send and receive real emails	**20%**	High
2	Inbox and Sent views display real data	**15%**	High
3	Compose and send works via UI	**10%**	Medium
4	**Assistant can compose/fill the email form via natural language**	**20%**	**Critical**
5	**Assistant can search/filter and update the main UI with results**	**15%**	**Critical**
6	Assistant is context-aware (knows current view, open email, etc.)	**10%**	High
7	Real-time mail sync (no manual refresh)	**10%**	Medium

Bonus

Criteria	Points
Reply/forward via assistant	**+5**
Assistant asks for confirmation before sending (human-in-the-loop)	**+5**
Rich UI rendering in the assistant panel (email previews, not just text)	**+5**
Thread / conversation view	**+3**
Polished UI / dark mode	**+2**
Tests	**+3**
Deployed live demo	**+2**

Tech Stack

Your choice. Use whatever frameworks, libraries, and languages you're most productive with.

A few references that may be useful (not required):

- **CopilotKit** — Framework for building AI assistants that control UI (docs.copilotkit.ai)
- **Google Gmail API** — developers.google.com/gmail/api
- **Microsoft Graph API** — learn.microsoft.com/en-us/graph/api/resources/mail-api-overview

Deliverables

1. GitHub Repository (Private)

Create a **private repository** and invite the following collaborators:

- **giri-mt**
- **adarsh-processity**

- Clean commit history showing progression
- Working README with setup instructions

2. README Must Include

- How to set it up and run it locally
- Architecture decisions and trade-offs you made
- Screenshots or a short video demo showing the assistant controlling the UI
- What you'd improve with more time

Submission

Submit within **5 calendar days** of receiving this task.

What We're Really Looking For

1. **The AI controls the UI** — This is not a chatbot. When the user says "send an email to X", the compose form visibly fills up. When they say "show last week's emails", the inbox updates. The assistant is a co-pilot that drives the interface.

2. **Clean architecture** — Clear separation between mail service, UI, and AI integration. Easy to follow, easy to extend.

3. **API integration** — You can connect to real external services (OAuth, mail APIs, push notifications).

4. **Pragmatic engineering** — Smart trade-offs, clear docs, sensible error handling. Not over-engineered, not under-built.

Good luck — we're excited to see what you build.

Questions? Just ask. Asking smart questions is a positive signal.