REACHINBOX

RAZOR SHARP COLD E-MAIL OUTREACH TOOL

Assignment - Associate Backend Engineer

About Us:

ReachInbox is transforming cold outreach with our revolutionary AI-driven platform. Our all-in-one solution empowers businesses to effortlessly find, enrich, and engage high-intent leads through multi-channel outreach on Twitter, LinkedIn, email, and phone. With just a single prompt, ReachInbox springs into action, prospecting and verifying leads, crafting personalized sequences, and notifying businesses of responsive prospects. Imagine being part of an AI-powered growth team that consistently generates top-tier leads. ReachInbox is more than a tool; it's your growth partner.

We are looking for passionate and innovative individuals to join our team and help us continue to redefine the future of lead generation and business growth.

Assignment - **Build a Feature-Rich Onebox for Emails**

Problem Statement

We are looking for the best candidates who can build a highly functional **onebox email aggregator** with advanced features, similar to **Reachinbox**. Your task is to

create a backend and frontend system that synchronizes multiple IMAP email accounts in real-time and provides a seamless, searchable, and AI-powered experience.

Your submission will be judged based on the number of features you successfully implement. We will maintain a **leaderboard on GitHub** to track progress, ranking submissions based on feature completion and quality.

Requirements & Features

For the **Backend Engineering assignment**, you will begin by building and showcasing the listed features using Postman. If you're able to successfully complete point 5, you will then integrate and display all the features on the frontend. Achieving this will demonstrate your ability to work end-to-end. Lastly, completing point 6 will secure you a direct invitation to the final interview.

Use Language: Typescript, Node.js runtime.

1. Real-Time Email Synchronization

- Sync multiple **IMAP accounts** in real-time - minimum 2
- Fetch **at least the last 30 days** of emails
- Use **persistent IMAP connections (IDLE mode)** for real-time updates (**No cron jobs!**).

2. Searchable Storage using Elasticsearch

- Store emails in a **locally hosted Elasticsearch** instance (use Docker).
- Implement indexing to **make emails searchable**.
- Support filtering by **folder & account**.

3. AI-Based Email Categorization

- Implement an AI model to categorize emails into the following labels:
    - **Interested**
    - **Meeting Booked**
    - **Not Interested**

    - **Spam**
    - **Out of Office**

4. Slack & Webhook Integration

- Send **Slack notifications** for every new **Interested** email.
- Trigger **webhooks** (use webhook.site as the webhook URL) for external automation when an email is marked as **Interested**.

5. Frontend Interface

- Build a **simple UI** to display emails, filter by folder/account, and show AI categorization.
- Basic **email search functionality** powered by Elasticsearch.

6. AI-Powered Suggested Replies (Direct invitation to final interview)

- Store the **product and outreach agenda** in a **vector database**.
- Use **RAG (Retrieval-Augmented Generation)** with any LLM to suggest replies.
- Example:
    - **Training data**: "I am applying for a job position. If the lead is interested, share the meeting booking link: https://cal.com/example"
    - **Email received**:*"Hi, Your resume has been shortlisted. When will be a good time for you to attend the technical interview?"*
    - **AI Reply Suggestion**:*"Thank you for shortlisting my profile! I'm available for a technical interview. You can book a slot here: https://cal.com/example"*

How and Where to submit the assignment:

1. **Create a private GitHub repository** with your implementation.
2. **Grant access to the user**: Mitrajit

3. **Push your code and update the README** with setup instructions, architecture details, and feature implementation.
4. **Provide a demo video** showcasing the functionalities. (Do not exceed 5 mins)
5. Fill this form with relevant links and details - Assignment Submission - https://forms.gle/DqF27M4Sw1dJsf4j6

Evaluation Criteria

1. **Feature Completion** – The number of features implemented.
2. **Code Quality & Scalability** – Clean, modular, and well-documented code.
3. **Real-Time Performance** – Efficient IMAP sync (no polling!).
4. **AI Accuracy** – Performance of email categorization and suggested replies.
5. **UX & UI** – Frontend usability and smooth user experience.
6. **Bonus Points** for additional features or optimizations.

Deadline for Task Completion:

You have a maximum of 48 hours to complete the task. Receiving this assignment means you're already ahead of many candidates. Good luck!

Note: Do not submit a plagiarized assignment. All GitHub code will be thoroughly reviewed, and any evidence of plagiarism will result in the assignment being rejected.

Thank you!