CODE WIZARDS

GoFloww Atom Mail Challenge

AIM

To design and implement an AI-driven email assistant for GoFlow Mail that streamlines email communication by enabling intelligent auto-composition, smart response generation, and content refinement.
The solution will leverage past email conversations to understand context, adapt to individual communication styles, and generate accurate, context-aware replies — ultimately reducing the time spent on email management while enhancing communication quality and user satisfaction.

KEY FEATURES THAT WE WANT OUR PROJECT TO HAVE

✨ AI-based email auto-composition
📩 Smart reply generation using Gemini API
📚 Context awareness using RAG (Retrieval-Augmented Generation)
📥 Functional, responsive UI with filters, starring, read/unread
🔒 Privacy and security of user data
🧠 Learning and adapting to individual communication styles

DEVELOPMENT DONE

Frontend Development – Atom Mail:

We built Atom Mail, a responsive and modern web-based email platform using React.js and Tailwind CSS. The UI supports core functionalities like inbox navigation, mail filtering, starring, read/unread toggling, and batch actions via checkboxes.

Key additions:

- Login & Signup pages with form validation.
- Drag & drop to Starred, dynamic search, and improved mail controls.
- Enhanced UI/UX with responsive design and clean component layout.
- Modular component structure for scalability and reusability.
- Profile, Settings & Apps dropdowns integrated into the Navbar.

.

https://github.com/rohangarg-2006/HackFest

DEVELOPMENT DONE

Backend Development – Atom Mail

The backend is built using Node.js and Express.js, offering a scalable and secure REST API for managing email data, user authentication, and third-party integrations.

Key functionalities:

- Email send/receive system fully synced with MongoDB to ensure real-time data consistency
- JWT-based authentication for secure user access
- Integrated all the ML Modules into the website.
- MongoDB stores user data, emails, preferences, and conversation history
- Seamless integration between frontend, backend, and external services (e.g., Gemini API, email servers) via well-defined API endpoints

These capabilities enable robust communication, smart automation, and smooth interaction between all system components

https://github.com/rohangarg-2006/HackFest

DEVELOPMENT DONE

Machine Learning – Atom Mail

The ML module powers the intelligence behind Atom Mail, enabling smarter, faster, and more organized email interactions.

Key features:

- Auto-reply generation using Google's Gemini API, providing context-aware and personalized responses.
- Email summarization for quick overviews of long threads.
- Spam detection to filter out unwanted or suspicious emails.
- Email classification into Personal, Professional, and Spam categories for better organization.
- Seamlessly integrated with the backend to serve real-time insights and suggestions to the frontend.

These AI-driven features enhance productivity and automate routine email tasks for a smoother user experience.

https://github.com/rohangarg-2006/HackFest

FRONT END

STACK USED

- React.js: Core library for building a fast, interactive user interface with reusable components.
- Tailwind CSS: Utility-first CSS framework for responsive design and rapid UI development.
- React Router: For managing navigation between views like Inbox, Sent, Starred, and individual emails.

BACK END

STACK USED

1)Node.js:

JavaScript runtime used for building a fast and scalable server-side environment.

2)Express.js:

Lightweight and flexible Node.js web application framework for routing, middleware, and API handling.

3)MongoDB

Stores email data, user profiles, preferences, and conversation history in a flexible, document-based format.

ML

Aim:

To enable intelligent auto-composition of emails based on user context and past communication.

We have successfully integrated the Gemini API to generate AI-driven email responses.

Our immediate goal is to implement a Retrieval-Augmented Generation (RAG) pipeline. This will allow the system to access and utilize relevant past email conversations, enabling the AI to generate contextually accurate and personalized responses that align with the user's communication history and intent.

ML CHATBOT USING RAG

- RAG-powered chatbot for answering user queries using both stored data and generative AI
- We implemented the Retrieval-Augmented Generation (RAG) pipeline for our chatbot to enhance response accuracy using contextual data. However, we were unable to fully integrate RAG into the chatbot model due to the complexity of our existing MongoDB schema. Refactoring the database structure to support the integration required more time than we had available, and altering the schema risked affecting other critical components of the website.

INDIVIDUAL CONTRIBUTION

Robin: Backend
Suryansh Dixit: Backend
Mukul Mishra: FrontEnd
Rohan Garg: FrontEnd
Rudra Gupta: ML

THANK YOU