COIMBATORE INSTITUTE OF TECHNOLOGY

AN AUTONOMOUS INSTITUTE AFFILIATED TO ANNA UNIVERSITY

CAT II - FULL STACK APPLICATION DEVELOPMENT LABORATORY

PROJECT DOCUMENTATION

Project title : Mail Service Application using Full Stack

Date of Submission : 28.10.2024

Submitted by: 71762231021 - Guhasri R

71762231023 - Jeevan N

CONTENT

Hedwig Mail service

I . Objective of the Project :

- The primary objective of this project is to create a web-based email client that replicates the core functionalities of Gmail, providing users with a familiar and user-friendly interface.
- Named after Harry Potter's owl, Hedwig, this project is designed to offer essential email services such as composing, sending, and managing emails, along with additional features like starred emails and group categorization.
- The goal is to ensure that users can manage their emails effectively and efficiently within a minimalistic, easy-to-navigate interface that prioritizes productivity and usability.

II . Scope of the Project

The Hedwig email project is structured to focus on both backend and frontend components, creating a complete web application that functions as an independent email client. The scope includes:

- **Frontend Development:**

  A React-based application structured in a modular fashion. The frontend comprises various components that handle email-related actions such as composing, sending, managing drafts, and grouping emails. Users interact

with the frontend through a clean and intuitive user interface, making it easy to perform common email tasks.

- **Backend Development:**

  The backend is developed using Node.js and Express, which provides RESTful APIs to manage email data. It handles the business logic for saving, updating, and retrieving emails, drafts, and groups, along with user authentication and authorization. The backend also manages MongoDB integration to ensure persistent storage for emails and user data.

- **Security Measures:**

  The application uses environment variables in a .env file to store sensitive information such as MongoDB URLs and server port numbers, safeguarding data from exposure.

III . Problem Statement

- In today's digital landscape, traditional email clients are often cluttered with advanced features, making them complicated for everyday users who only need essential functionalities.

- This complexity can lead to user frustration, especially when trying to accomplish basic tasks quickly.

- Hedwig addresses this issue by offering a streamlined, minimal interface with all necessary email functionalities, allowing users to focus on essential actions such as composing and organizing emails.

- The project further aims to improve user experience by implementing a Groups feature, where emails can be categorized into custom groups for easier management.

- The addition of a starred section also enhances productivity by enabling users to quickly access important messages.

IV . Project Description

The Hedwig Gmail clone is designed to serve as an email client with a similar look and feel to popular email services, providing users with the core features they expect from an email client. The project is divided into several key components and features that contribute to its functionality and ease of use:

1. Compose and Send - The compose mail feature allows users to create emails with essential fields such as CC, BCC, and attachments. The rich text editor enables users to format their emails easily, while validation ensures that only valid emails are sent.

2. Drafts Management - Users can save incomplete emails as drafts, which they can access later to review, edit, or send. Drafts are stored in MongoDB, ensuring that data persists even after the user logs out or exits the application.

3. Starred Emails - Users can mark emails as 'starred' to easily access important messages. This feature improves productivity by reducing the time needed to locate priority emails.

4. Bin Management - Deleted emails are stored in a bin, allowing users to recover messages they may have accidentally deleted. Emails in the bin can either be restored or permanently deleted, giving users control over their data.

5. Groups Feature - This feature enables users to categorize emails under custom groups, facilitating better organization. For example, users might create groups like 'Work,' 'Family,' or 'Friends,' which helps in quickly locating relevant emails.

6. Backend and Frontend Setup - The frontend and backend are set up separately. To start the backend, users run `node server.js`, which initializes the Express server and connects to MongoDB. The frontend can be launched with `npm start` under the 'gmail' directory, providing users with an interactive interface to manage their emails.

V . Backend Structure

The backend folder consists of the following:

- **Models**: Contains Mongoose schemas such as Drafts.js, Email.js, Group.js, etc., to define data structures.

- **Routes:** Each feature has its route file, like allMails.js for viewing all emails, auth.js for authentication, and drafts.js for managing draft emails.

- **Data and Configurations:** The backend includes a CountryCodes.json file under 'data' for reference and an .env file to store database URLs and port numbers securely.

VI . Frontend Structure

The frontend includes a set of React components in the 'src/components' directory, each handling a specific part of the application:

- **AllMails.js:** Displays all received emails for the user.

- **BinEmails.js:** Shows emails that have been moved to the bin for potential recovery or deletion.

- **ComposeMail.js:** Provides the interface for composing new emails, with fields for To, CC, BCC, and subject.

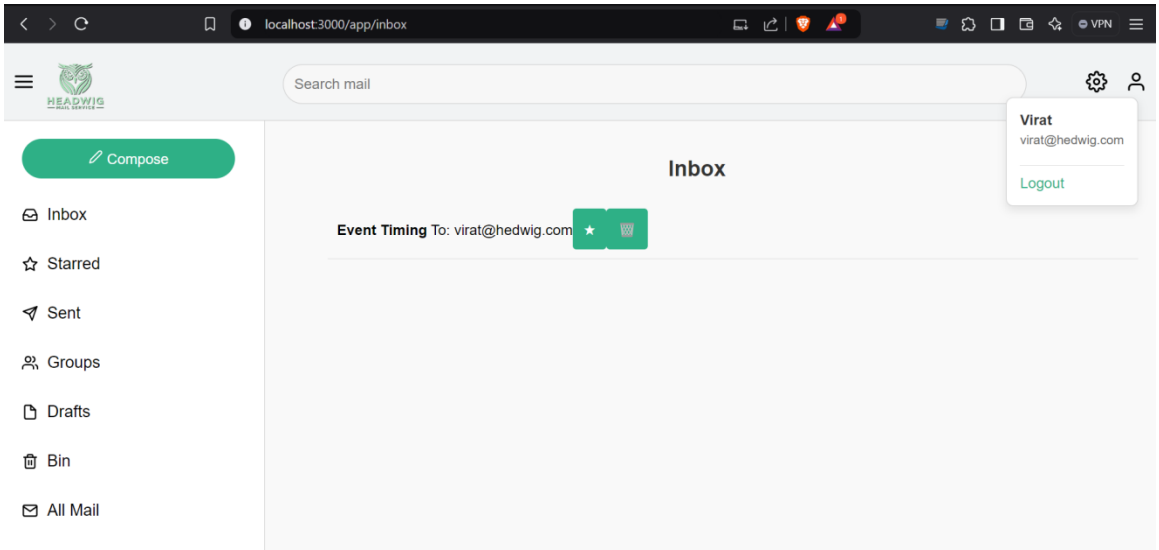
- **DraftsList.js:** Displays a list of saved drafts, allowing users to open and edit drafts as needed.

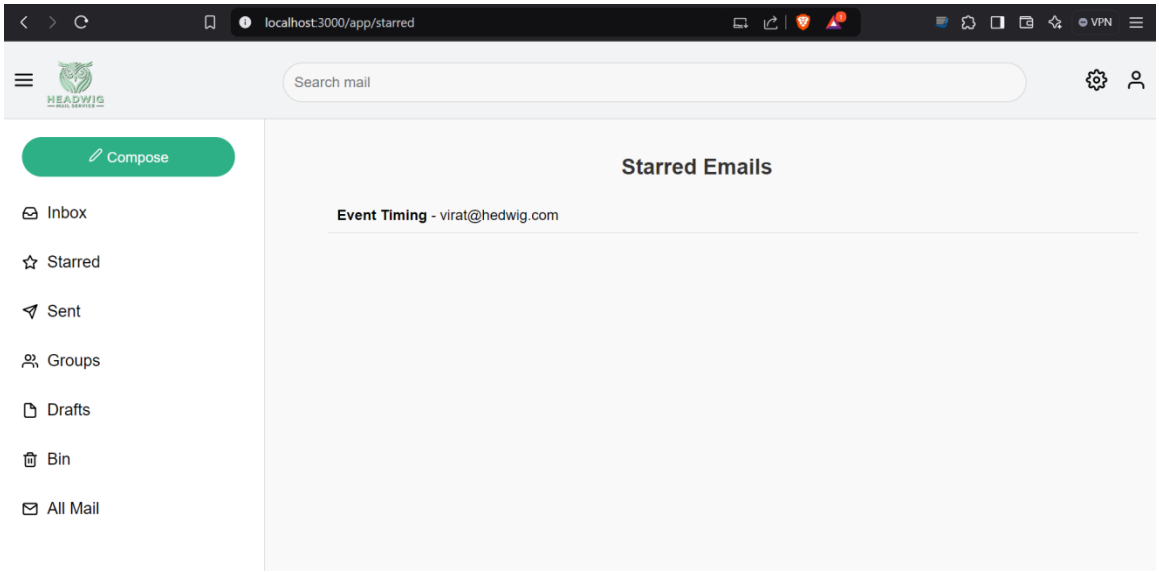
- **Settings.js:** Allows users to configure personal settings related to the email service.

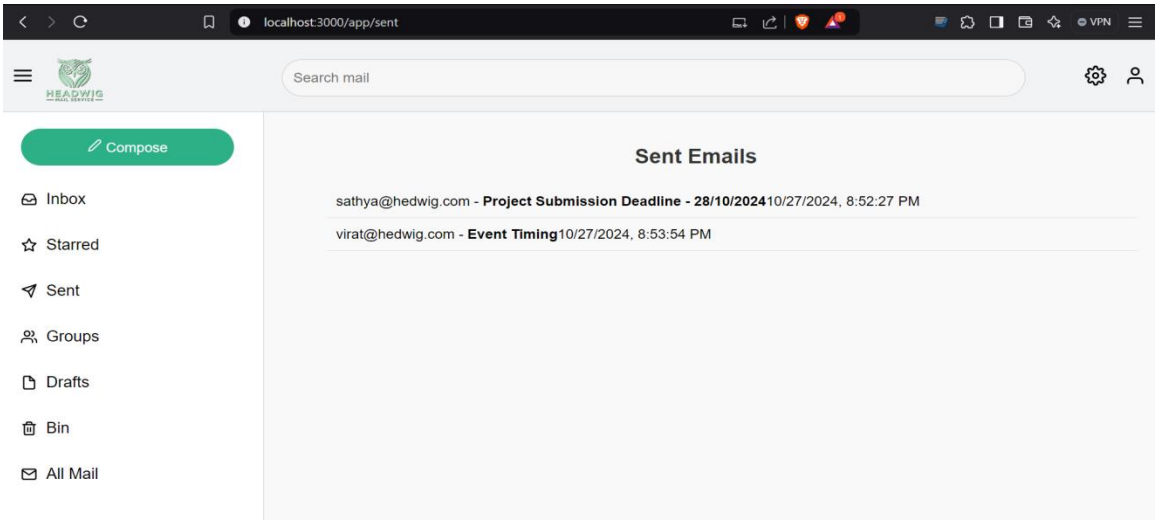
- **Sidebar.js:** Contains links to various sections such as Inbox, Sent, Drafts, and Starred emails for easy navigation.

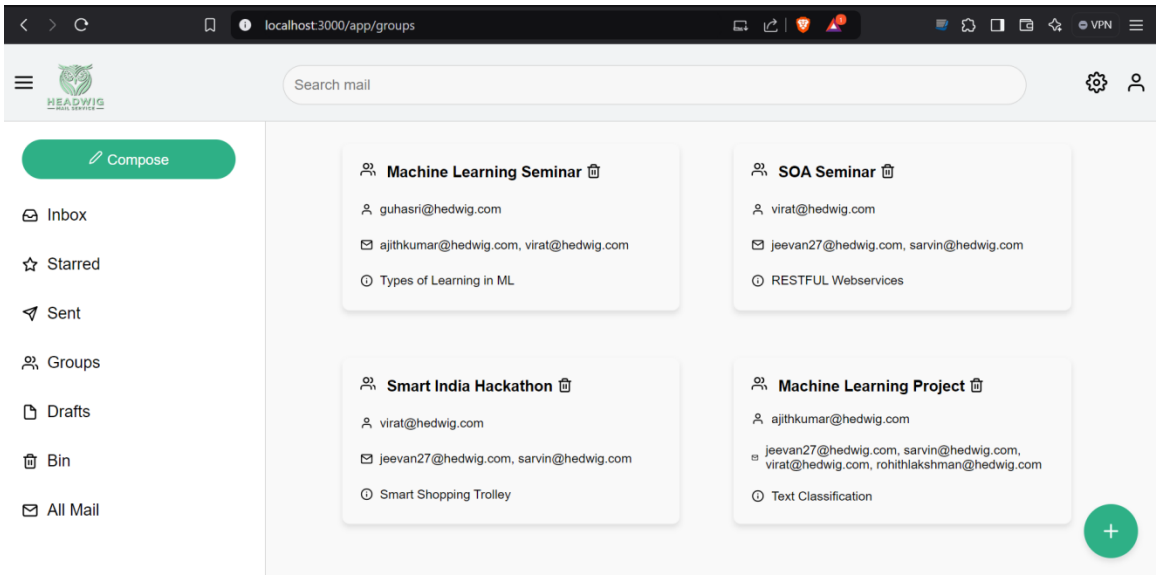
VII . Class Diagram :

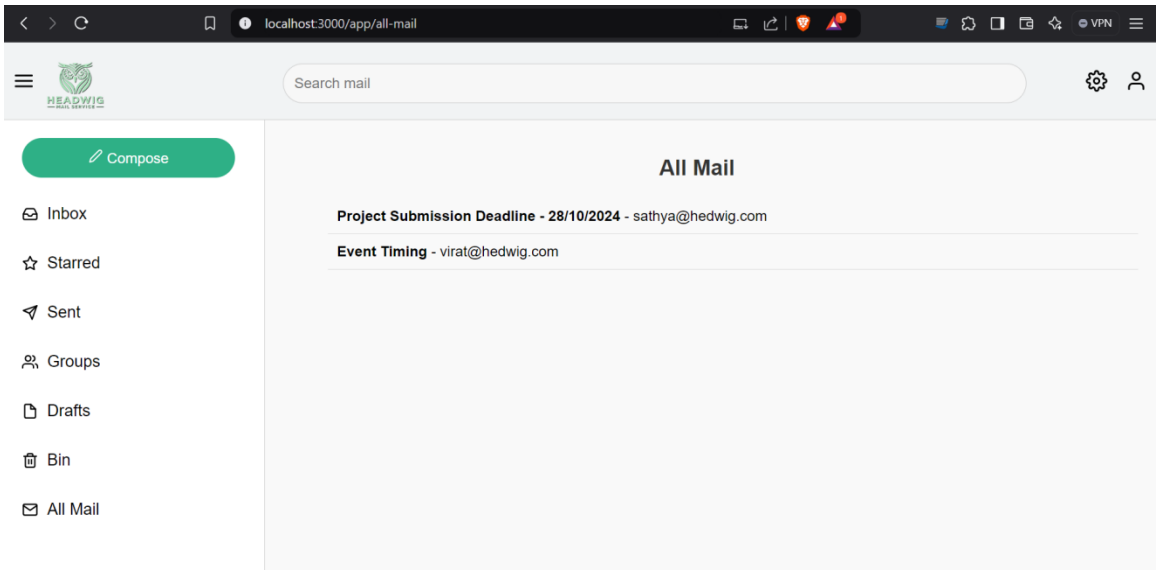
VIII . Application screenshot:

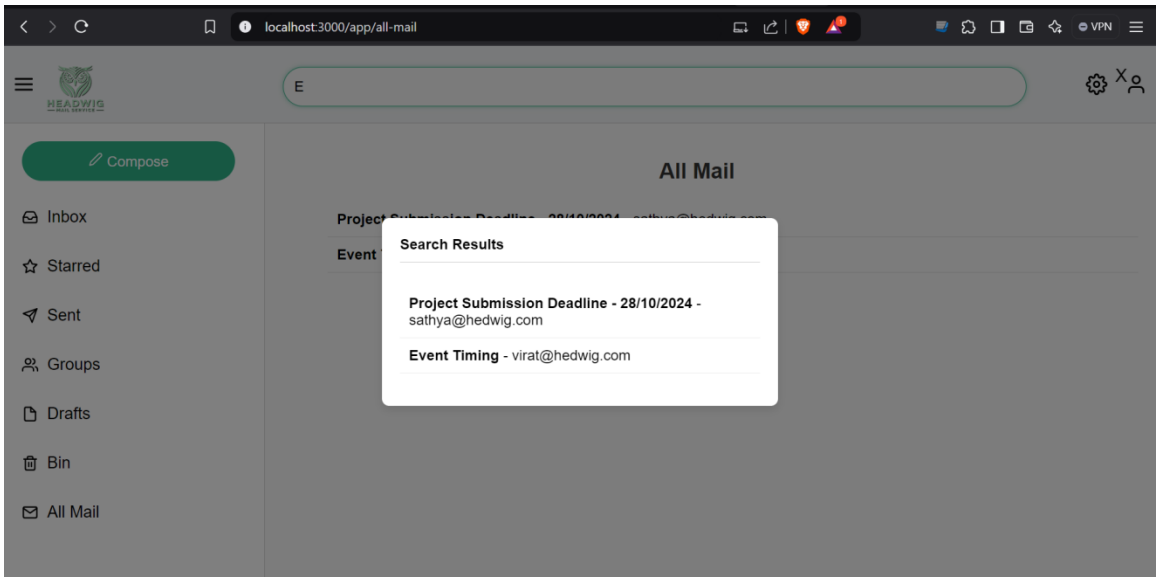


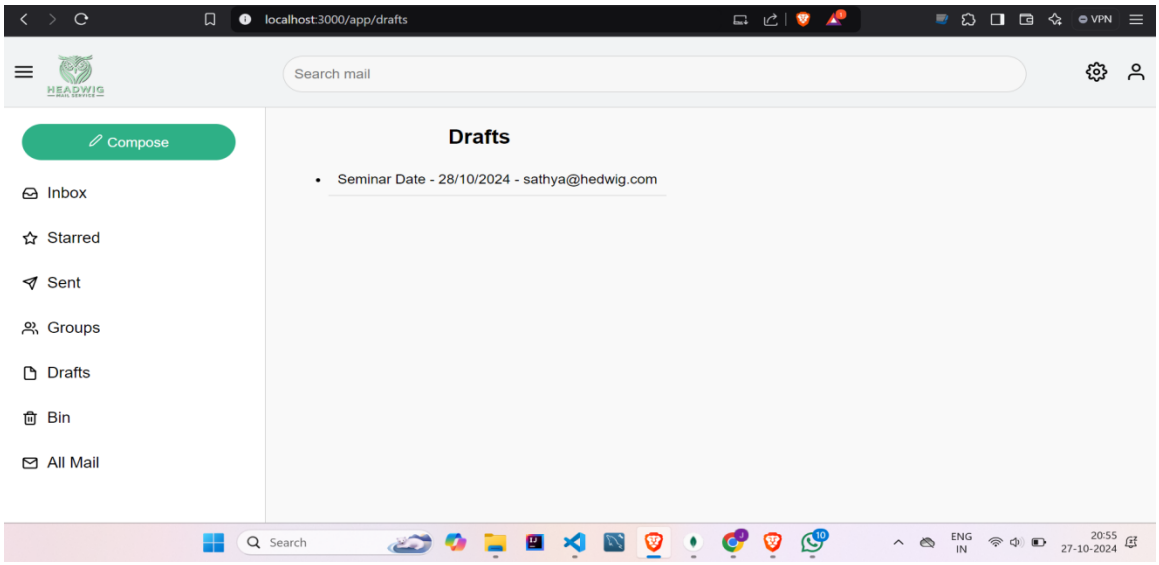

localhost:3000/app/starred
VPN
HEADWIG
Search mail
Compose
Inbox
Starred
Sent
Groups
Drafts
Bin
All Mail
Starred Emails
Event Timing - virat@hedwig.com

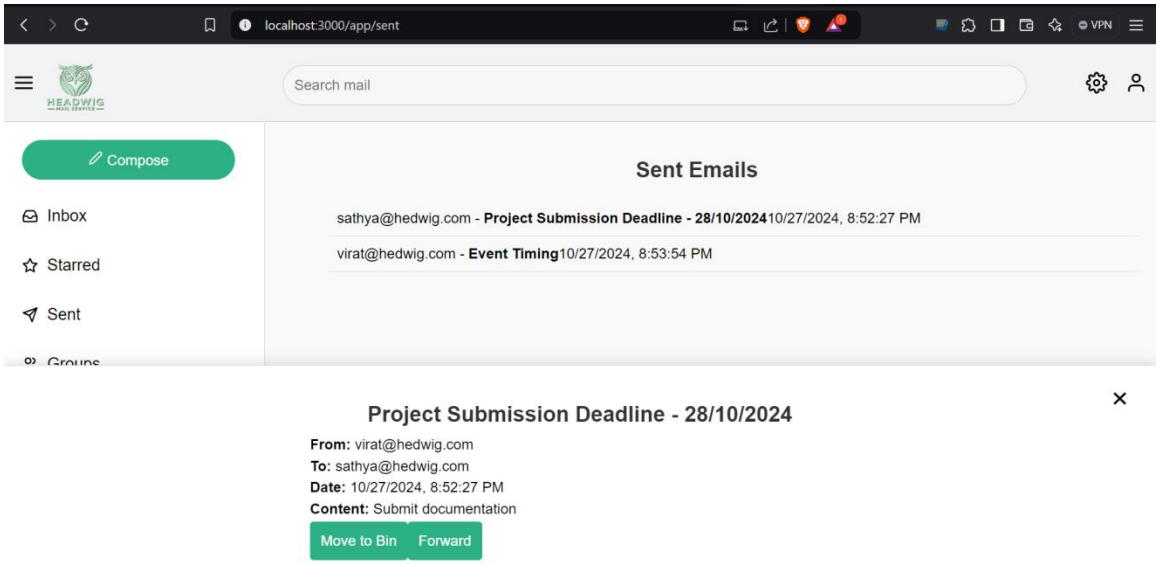

localhost:3000/app/sent
VPN
HEADWIG
Search mail
Compose
Inbox
Starred
Sent
Groups
Drafts
Bin
All Mail
Sent Emails
sathya@hedwig.com - Project Submission Deadline - 28/10/202410/27/2024, 8:52:27 PM
virat@hedwig.com - Event Timing10/27/2024, 8:53:54 PM


localhost:3000/app/groups
VPN
HEADWIG
Search mail
Compose
Inbox
Starred
Sent
Groups
Drafts
Bin
All Mail
Machine Learning Seminar
guhasri@hedwig.com
ajithkumar@hedwig.com, virat@hedwig.com
Types of Learning in ML
SOA Seminar
virat@hedwig.com
jeevan27@hedwig.com, sarvin@hedwig.com
RESTFUL Webservices
Smart India Hackathon
virat@hedwig.com
jeevan27@hedwig.com, sarvin@hedwig.com
Smart Shopping Trolley
Machine Learning Project
ajithkumar@hedwig.com
jeevan27@hedwig.com, sarvin@hedwig.com, virat@hedwig.com, rohithlakshman@hedwig.com
Text Classification


localhost:3000/app/all-mail
VPN
HEADWIG
Search mail
Compose
Inbox
Starred
Sent
Groups
Drafts
Bin
All Mail
All Mail
Project Submission Deadline - 28/10/2024 - sathya@hedwig.com
Event Timing - virat@hedwig.com


localhost:3000/app/all-mail
VPN
E
HEADWIG
Compose
Inbox
Starred
Sent
Groups
Drafts
Bin
All Mail
All Mail
Search Results
Project Submission Deadline - 28/10/2024 - sathya@hedwig.com
Event Timing - virat@hedwig.com


localhost:3000/app/drafts
VPN
Search mail
HEADWIG
Compose
Inbox
Starred
Sent
Groups
Drafts
Bin
All Mail
Drafts
Seminar Date - 28/10/2024 - sathya@hedwig.com
Search
ENG
IN
20:55
27-10-2024



IX . Conclusion:

The Hedwig email client effectively combines essential email functionalities within a straightforward interface. By focusing on simplicity and user-centric design, it aims to serve as a practical alternative to traditional email clients. Hedwig enhances productivity through features like drafts management, starred messages, and custom groups. This project demonstrates how a balanced combination of frontend and backend technologies can lead to a powerful, user-friendly application. The inclusion of MongoDB ensures that data is stored securely and remains accessible, providing a robust foundation for expanding the application’s capabilities in the future.