InstaMail

Table of contents

• Introduction

InstaMail project is a comprehensive web-based email application designed to provide users with a modern, intuitive, and secure platform for managing their email communications. Built using **React** for the frontend, **Spring Boot** for the backend, and **MySQL** for data persistence, the application supports seamless integration between components and ensures efficient performance.

This project extends beyond basic email functionalities, offering a robust organizational system through customizable folders. Users can create, rename, delete, and organize emails into folders, providing flexibility and control over their inbox. Users can attach add attachements of any file to the email .It also includes a contact management system, enabling users to save recipient details and streamline the process of composing emails.

With **Server-Sent Events (SSE)**, the application ensurs users stay updated with new emails instantly. Advanced search and filtering options make it easy to locate specific emails or sort messages based on categories, keywords, or folders.

Security is a key focus of the application. Authentication is implemented using **JWT** for secure access, and passwords are hashed with **BCrypt** to safeguard user data.

Technology Stack

- **Frontend:** React
- **Backend:** Spring Boot
- **Database:** MySQL
- **Authentication:** JWT & BCrypt
- **Real-Time Communication:** SSE (Server-Sent Events)

How to Run the Application

Prerequisites

Before starting, ensure the following tools are installed and configured:

1. **Backend Requirements:**
   - Java Development Kit (JDK) 11 or later
   - Maven
   - MySQL Server
2. **Frontend Requirements:**
   - Node.js (version 16.x or later)
   - npm (comes with Node.js)

1. Clone the Repository

1. Open a terminal or command prompt.
2. Clone the repository:
   ```
   git clone https://github.com/omarzydan610/InstaMail
   ```
3. Navigate to the project folder:
   ```
   cd InstaMail
   ```

2. Set Up the Backend (Spring Boot)

Step 2.1: Configure the Database

1. Start your MySQL server.
2. Open your MySQL client and create a database:
   ```
   CREATE DATABASE InstaMail;
   ```
3. Update the application.properties file located in src/main/resources:
   ```
   spring.datasource.url=jdbc:mysql://localhost:3306/InstaMail
   spring.datasource.username=root
   spring.datasource.password=instamail
   spring.jpa.hibernate.ddl-auto=update
   spring.jpa.show-sql=true
   spring.jpa.properties.hibernate.format_sql=true
   ```

Step 2.2: Start BackEnd Server

1. Navigate to the backend directory:
   ```
   cd instamail-backend
   ```
2. Run java application from run button

3. Set Up the FrontEnd (React)

1. Navigate to the frontend directory:
   ```
   cd instamail-frontend
   ```
2. Run frontend server
   ```
   npm install
   npm start
   ```

- Used Design Patterns

1. Singleton Pattern

The **Singleton** pattern ensures that only one instance of a class is created throughout the application. In this project, it is applied to manage the configuration and initialization of the Spring Boot application context, ensuring that critical services such as database connections are only instantiated once, improving efficiency and reducing overhead.

2. Proxy Pattern

The **Proxy** pattern is used for controlling access to certain functionalities, specifically for limiting login attempts. It helps prevent brute force attacks by restricting the number of login attempts in a given time frame. If a user exceeds the allowed attempts, the proxy limits further login attempts, enhancing security.

3. Strategy Pattern

The **Strategy** pattern is applied to the email sorting functionality. Users can select different sorting algorithms (e.g., by date, sender, or subject), and the strategy pattern allows the system to dynamically change the sorting behavior without altering the core logic of the email management system.

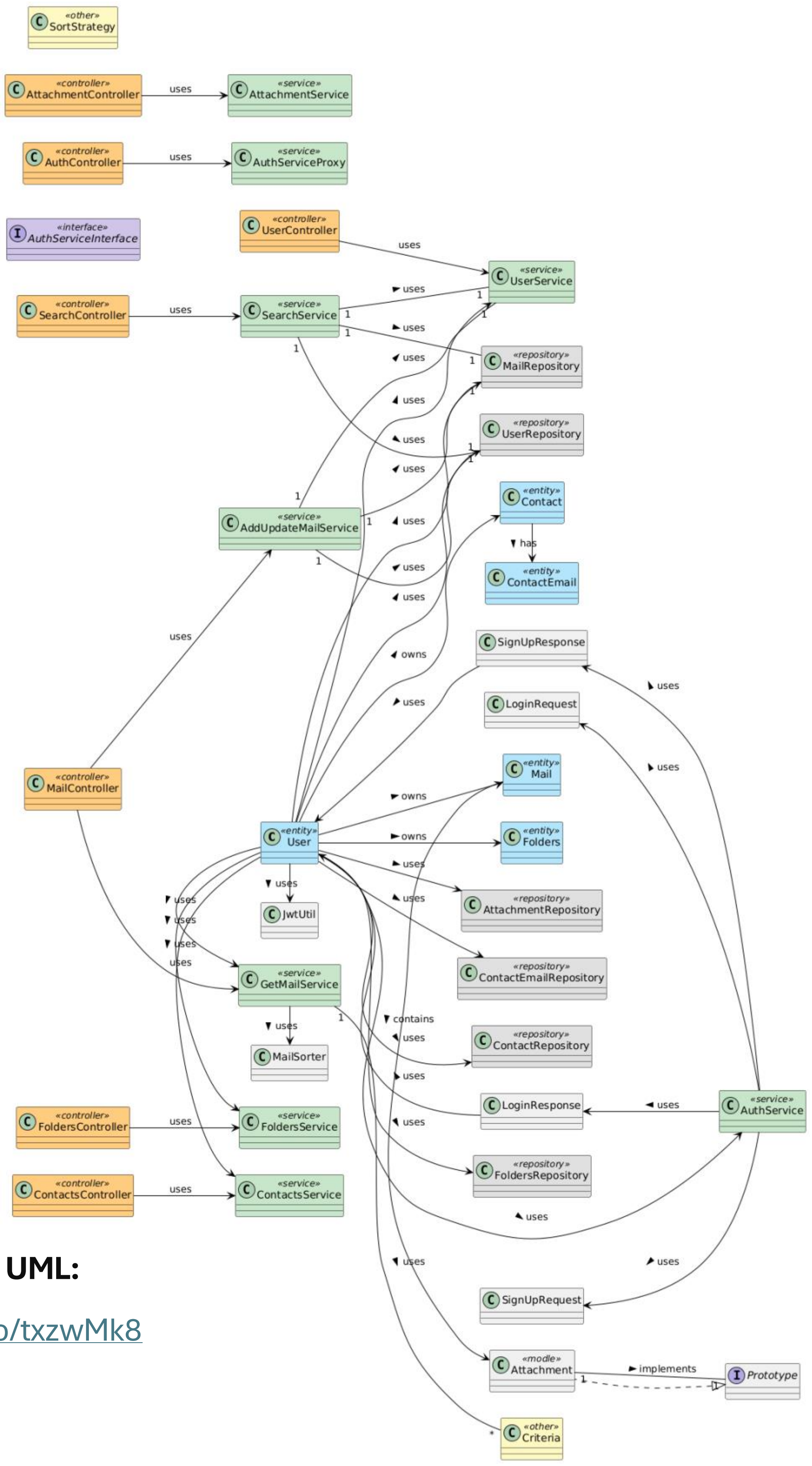
4. Filter Pattern

The **Filter** pattern is utilized for filtering emails based on various criteria, such as category (Inbox, Sent, Draft, Trash). This allows users to organize and find emails quickly without modifying the core email data, making the system more flexible and efficient.

5. Prototype Pattern

The **Prototype** pattern is used for sending emails to multiple recipients. Instead of creating a new email object for each recipient, the system clones the original email using the prototype pattern, ensuring that each email is sent efficiently while maintaining consistency across all recipients.

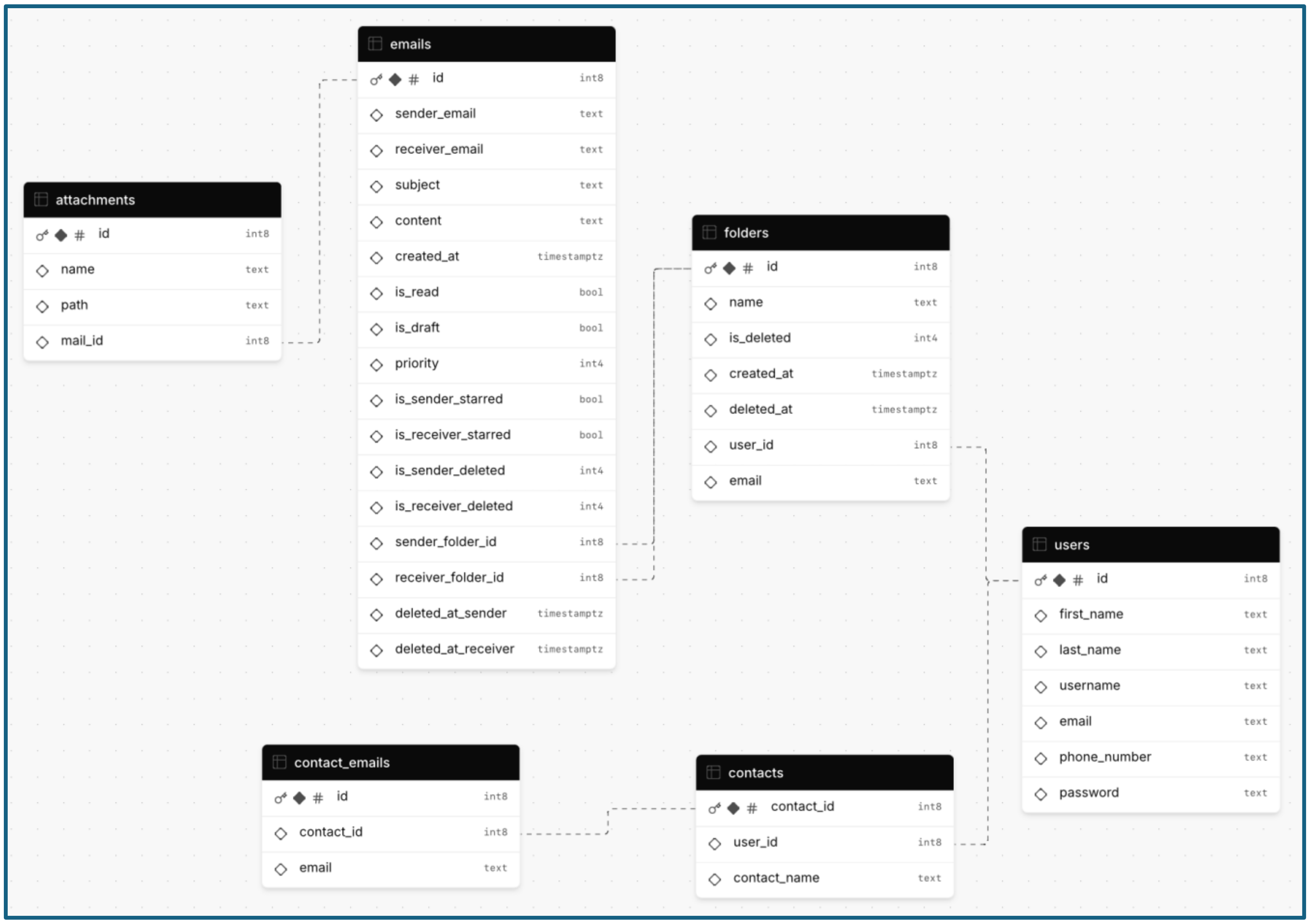
- UML Design


«other» SortStrategy
«controller» AttachmentController
uses
«service» AttachmentService
«controller» AuthController
«service» AuthServiceProxy
«interface» AuthServiceInterface
«controller» UserController
«service» UserService
«controller» SearchController
«service» SearchService
«repository» MailRepository
«repository» UserRepository
«service» AddUpdateMailService
«entity» Contact
has
«entity» ContactEmail
SignUpResponse
LoginRequest
«controller» MailController
«entity» Mail
«entity» User
owns
«entity» Folders
JwtUtil
«repository» AttachmentRepository
«repository» ContactEmailRepository
«service» GetMailService
MailSorter
contains
«repository» ContactRepository
LoginResponse
«service» AuthService
«controller» FoldersController
«service» FoldersService
«repository» FoldersRepository
«controller» ContactsController
«service» ContactsService
SignUpRequest
«model» Attachment
implements
Prototype
«other» Criteria

For Detailed UML:

https://ibb.co/txzwMk8

- Data Base Design



emails	
id	int8
sender_email	text
receiver_email	text
subject	text
content	text
created_at	timestamptz
is_read	bool
is_draft	bool
priority	int4
is_sender_starred	bool
is_receiver_starred	bool
is_sender_deleted	int4
is_receiver_deleted	int4
sender_folder_id	int8
receiver_folder_id	int8
deleted_at_sender	timestamptz
deleted_at_receiver	timestamptz

attachments	
id	int8
name	text
path	text
mail_id	int8

folders	
id	int8
name	text
is_deleted	int4
created_at	timestamptz
deleted_at	timestamptz
user_id	int8
email	text

users	
id	int8
first_name	text
last_name	text
username	text
email	text
phone_number	text
password	text

contact_emails	
id	int8
contact_id	int8
email	text

contacts	
contact_id	int8
user_id	int8
contact_name	text

Project Workflow

1. User Authentication

- Users log in with their credentials.
- The system validates credentials using **BCrypt** for hashed password verification and issues a **JWT** for secure session management.

2. Email Composition and Sending

- Users can compose an email, select single or multiple recipients, and send the email.
- The **Prototype Pattern** is used to clone the email object for each recipient, ensuring efficient handling of multi-recipient emails.

3. Email Organization

- Users can create, rename, delete, and organize emails into custom folders.
- Emails can be moved between folders, providing flexibility in email management.

4. Contact Management

- Users can save recipient information as contacts for faster access when composing new emails.

5. Real-Time Notifications

- The application uses **Server-Sent Events (SSE)** to notify users in real-time about incoming emails without requiring page reloads.

6. Email Filtering and Searching

- Users can filter emails based on categories (Inbox, Sent, Trash) or folders using the **Filter Pattern**.
- A search functionality allows users to locate specific emails by keywords or other criteria.

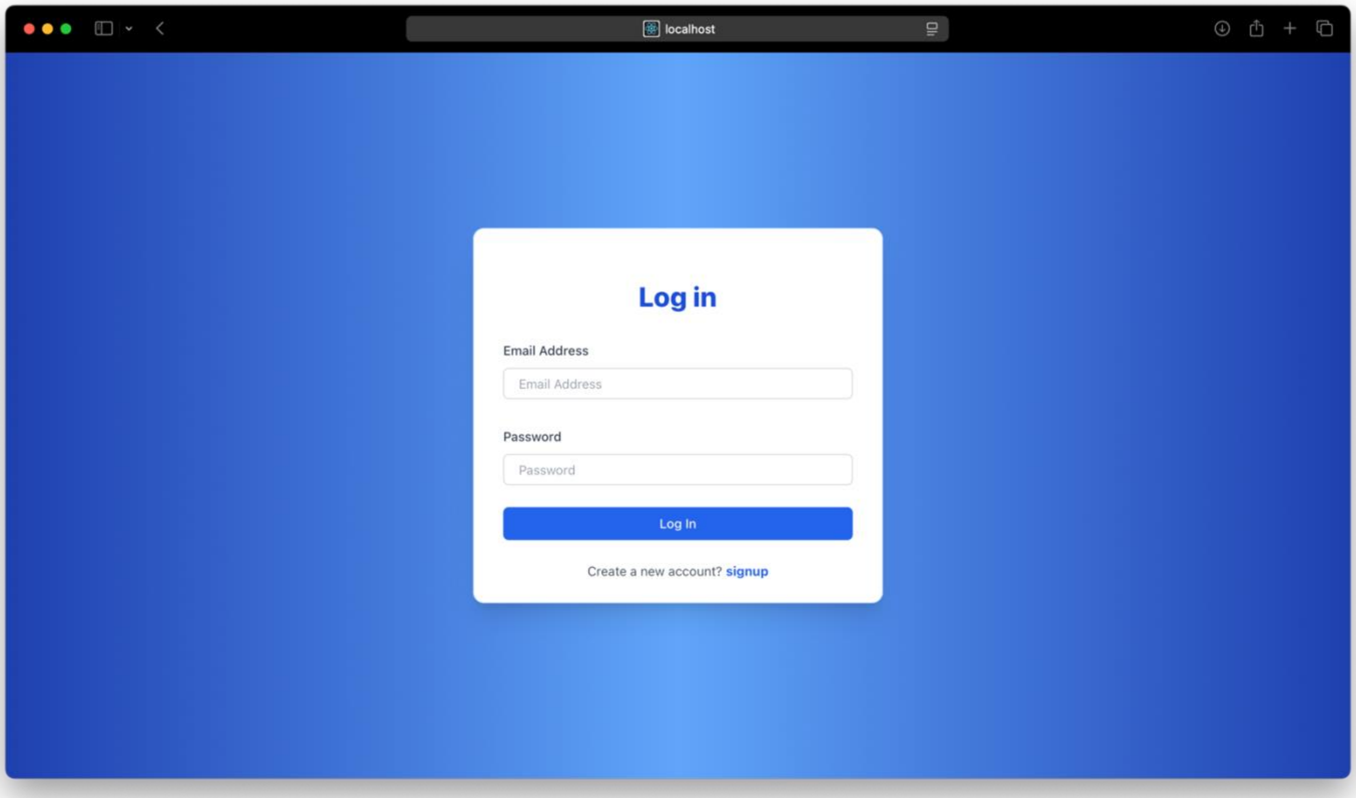
7. Email Sorting

- Emails can be sorted dynamically by attributes such as date, sender, or subject.
- The **Strategy Pattern** facilitates easy switching between different sorting algorithms.

8. Rate Limiting for Login Attempts

- To prevent brute force attacks, the **Proxy Pattern** enforces a limit on the number of login attempts allowed within a specified time frame.

- Screen Shots from application



Log in

Email Address

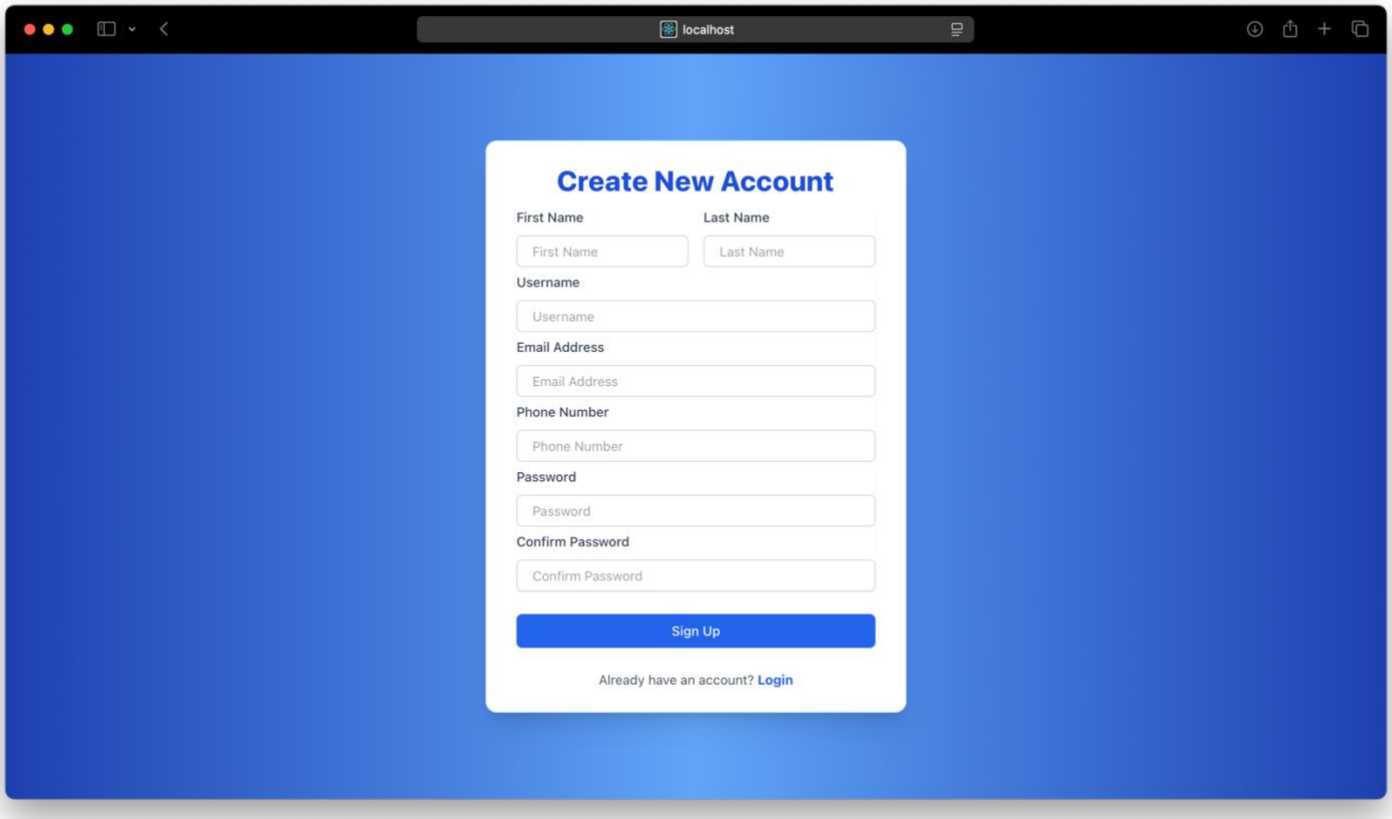
Email Address

Password

Password

Log In

Create a new account? signup



Create New Account

First Name

First Name

Last Name

Last Name

Username

Username

Email Address

Email Address

Phone Number

Phone Number

Password

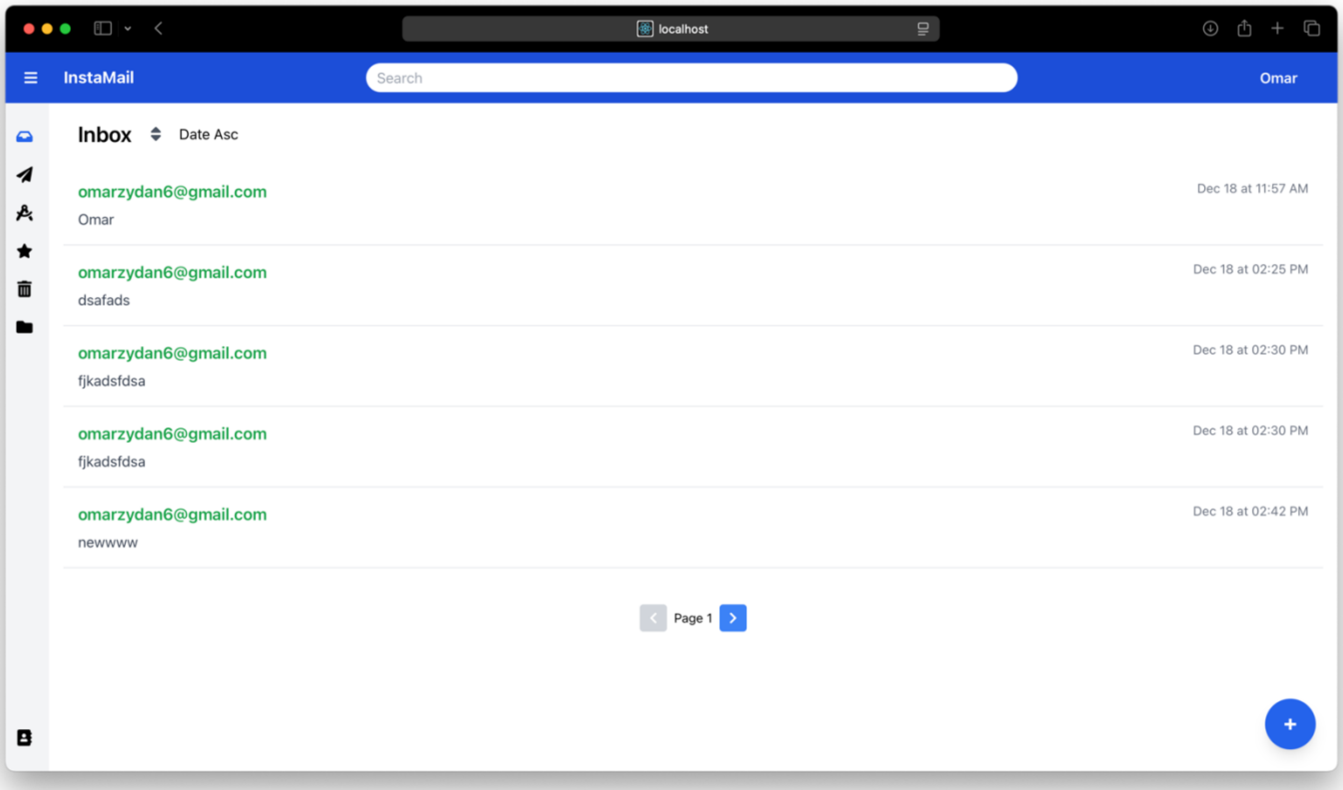
Password

Confirm Password

Confirm Password

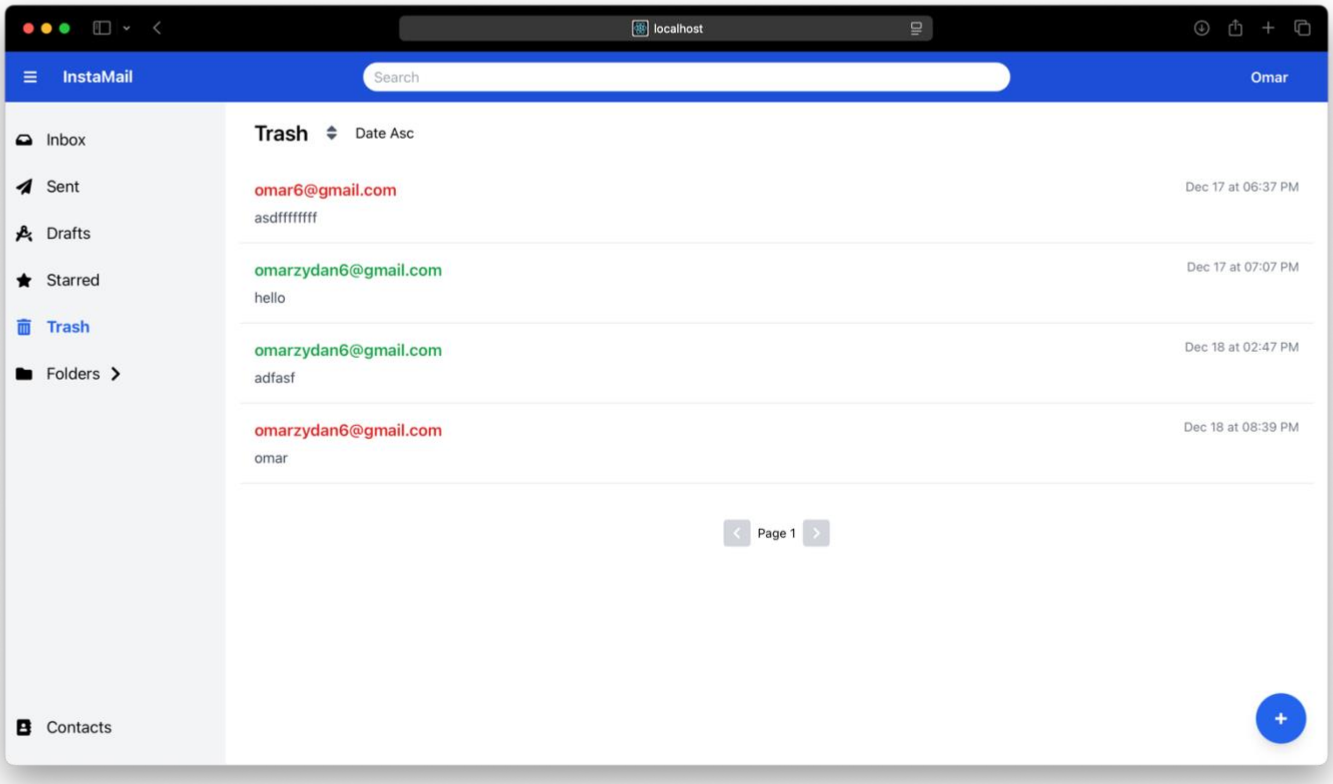
Sign Up

Already have an account? Login



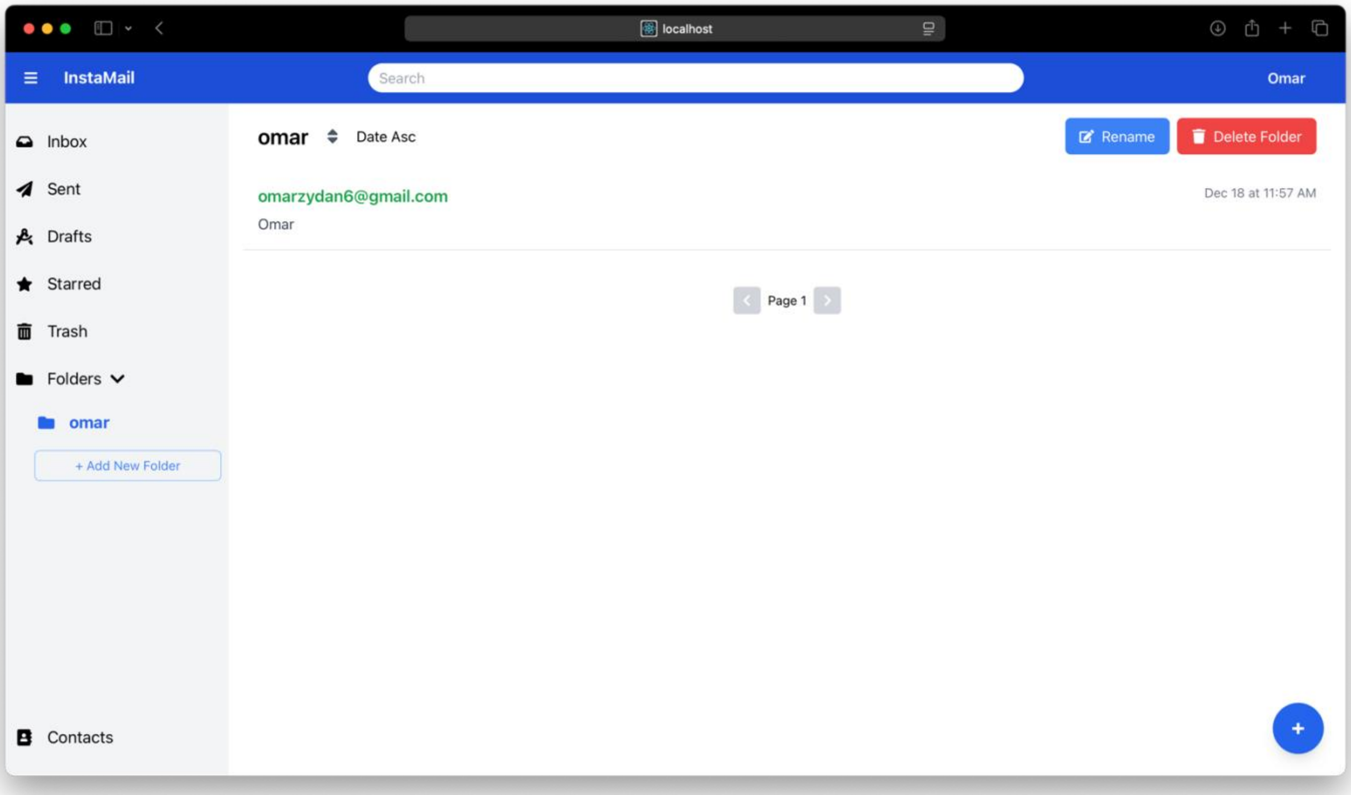
Inbox Date Asc

omarzydan6@gmail.com Omar	Dec 18 at 11:57 AM
omarzydan6@gmail.com dsafads	Dec 18 at 02:25 PM
omarzydan6@gmail.com fjkadsfdsa	Dec 18 at 02:30 PM
omarzydan6@gmail.com fjkadsfdsa	Dec 18 at 02:30 PM
omarzydan6@gmail.com newwww	Dec 18 at 02:42 PM



Trash Date Asc

omar6@gmail.com asdfffffffff	Dec 17 at 06:37 PM
omarzydan6@gmail.com hello	Dec 17 at 07:07 PM
omarzydan6@gmail.com adfasf	Dec 18 at 02:47 PM
omarzydan6@gmail.com omar	Dec 18 at 08:39 PM



localhost

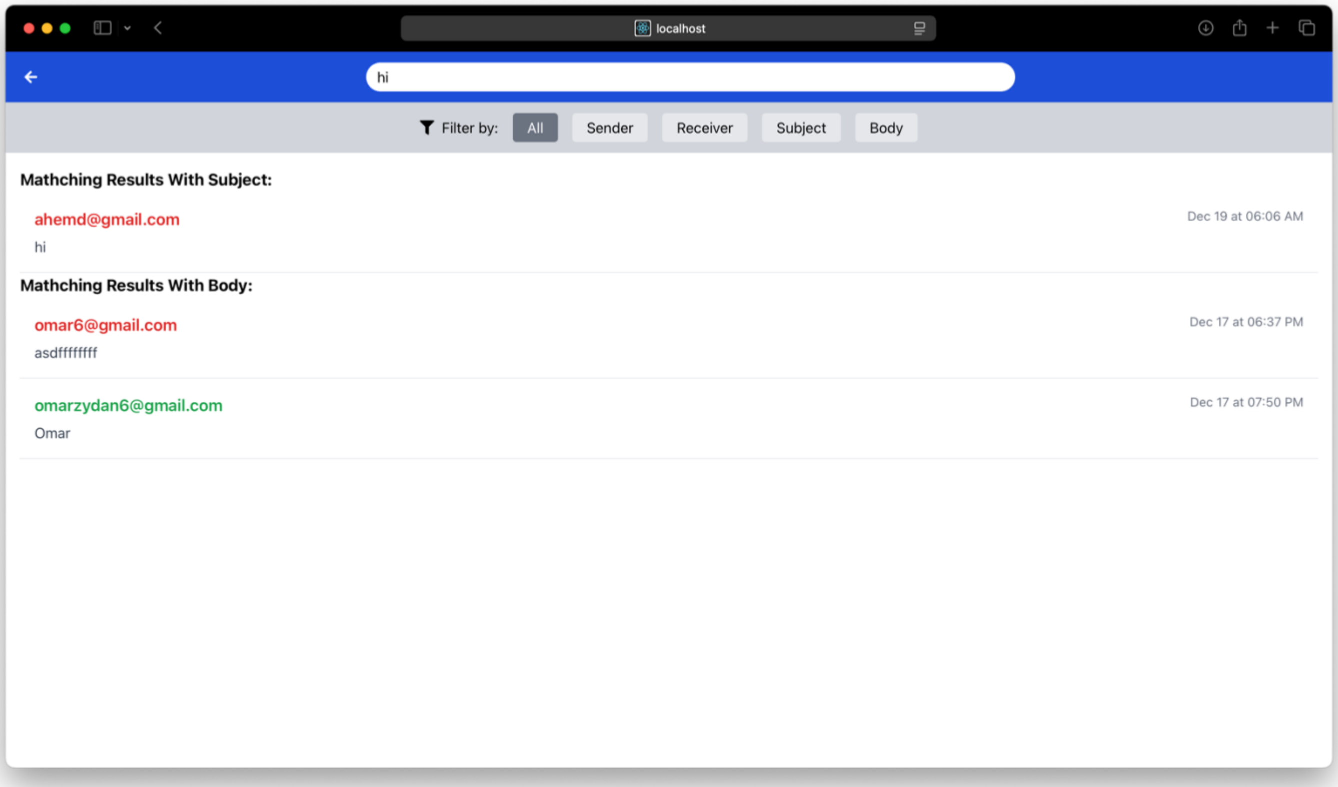
InstaMail
Search
Omar

Inbox
Sent
Drafts
Starred
Trash
Folders
omar
+ Add New Folder
Contacts

omar   Date Asc   Rename   Delete Folder

omarzydan6@gmail.com   Dec 18 at 11:57 AM
Omar

Page 1



localhost

hi

Filter by: All   Sender   Receiver   Subject   Body

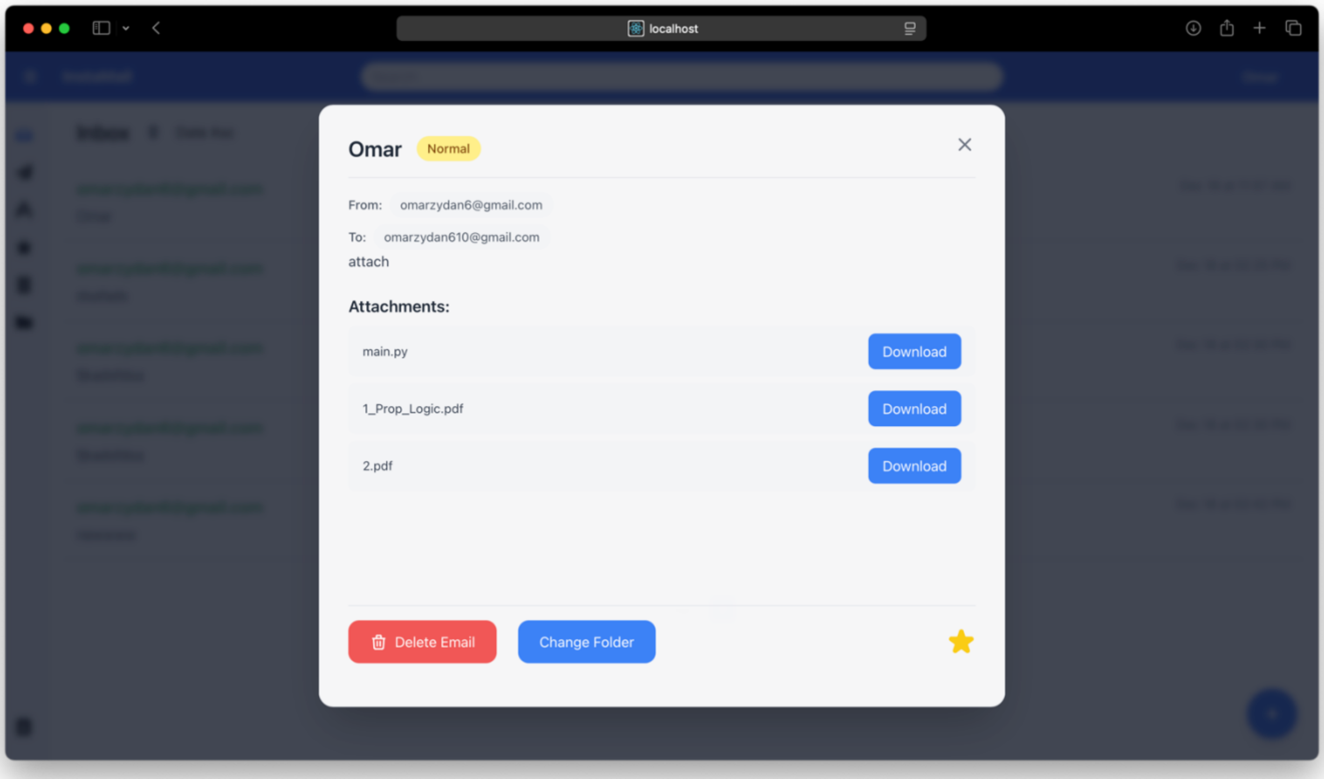
Mathching Results With Subject:

ahemd@gmail.com   Dec 19 at 06:06 AM
hi

Mathching Results With Body:

omar6@gmail.com   Dec 17 at 06:37 PM
asdffffffff

omarzydan6@gmail.com   Dec 17 at 07:50 PM
Omar


localhost

Omar Normal

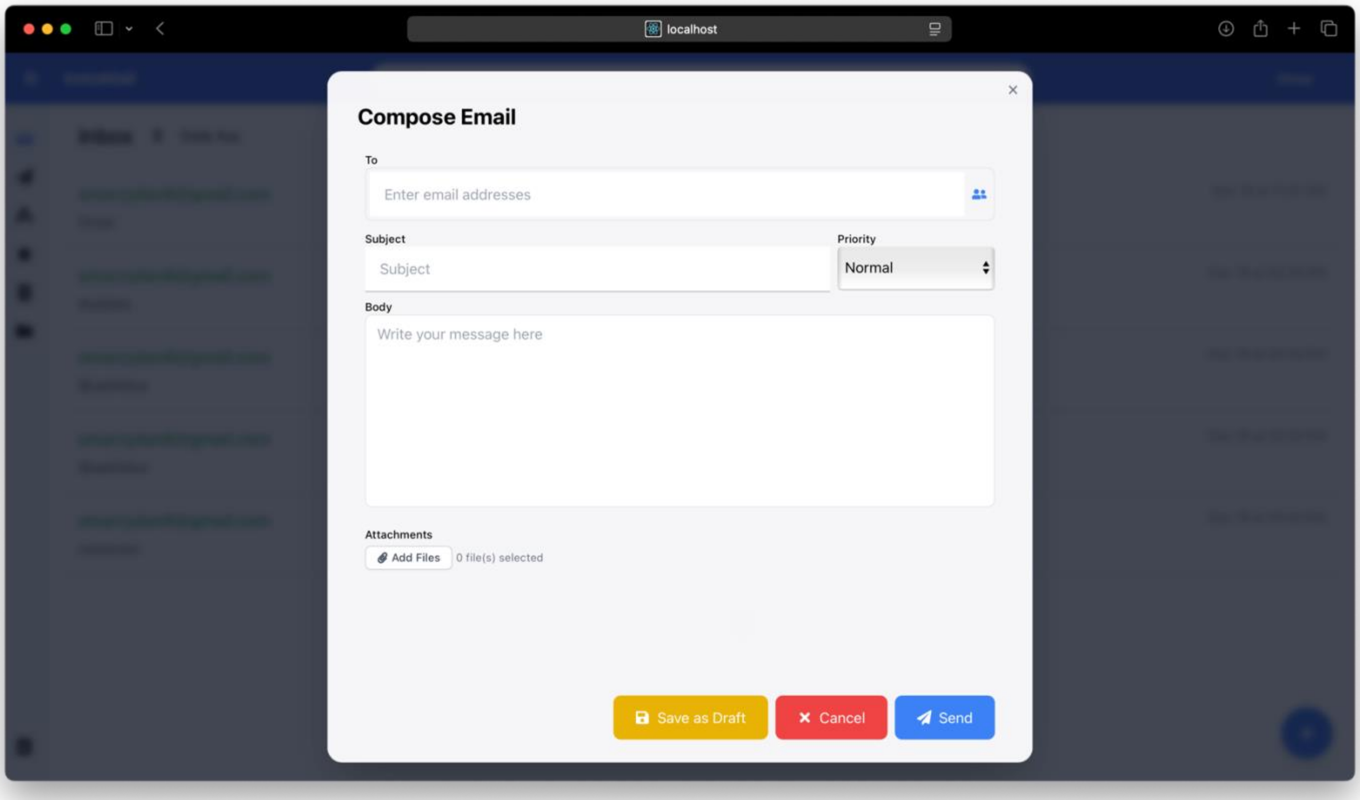
From: omarzydan6@gmail.com

To: omarzydan610@gmail.com

attach

Attachments:

main.py	Download
1_Prop_Logic.pdf	Download
2.pdf	Download


localhost

Compose Email

To

Enter email addresses

Subject

Subject

Priority

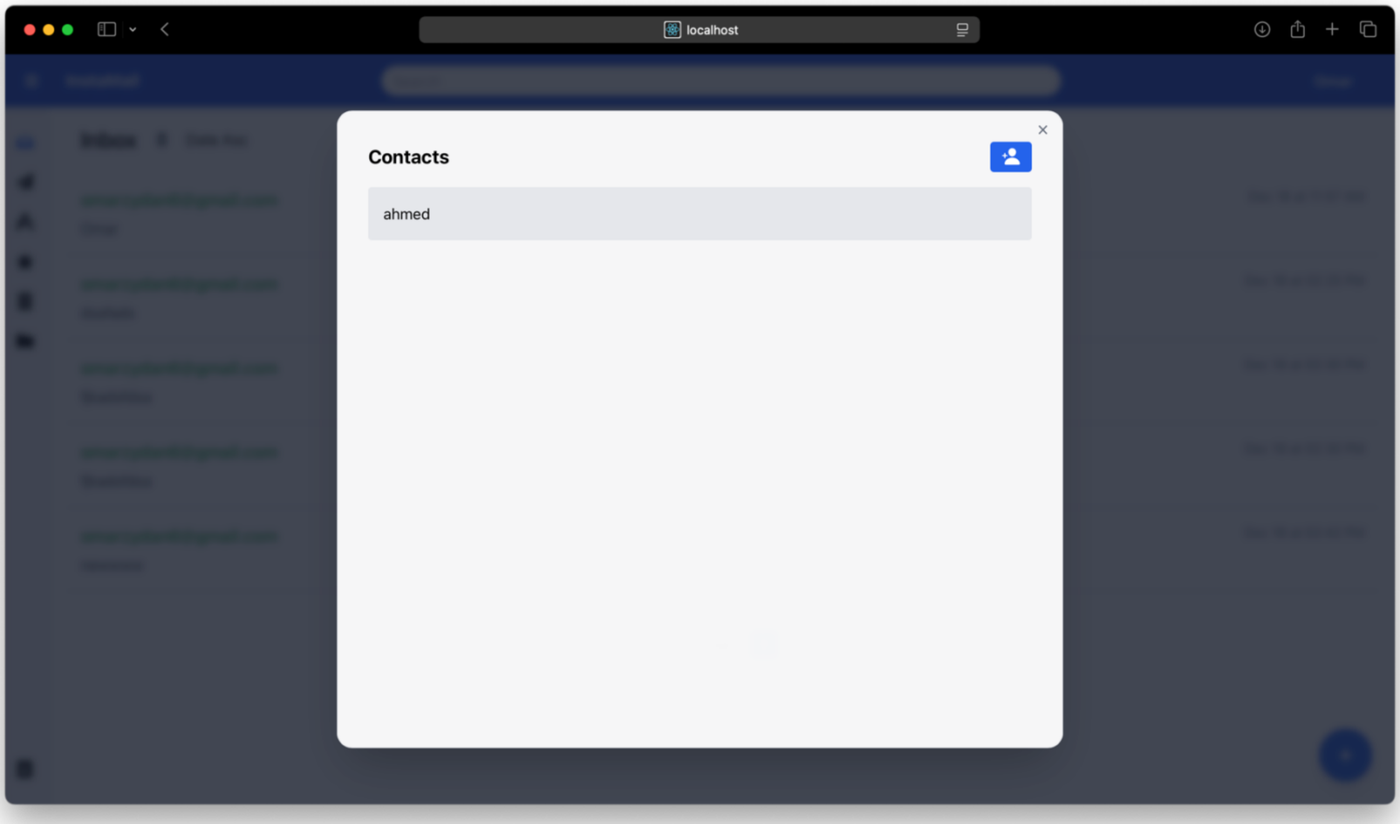
Normal

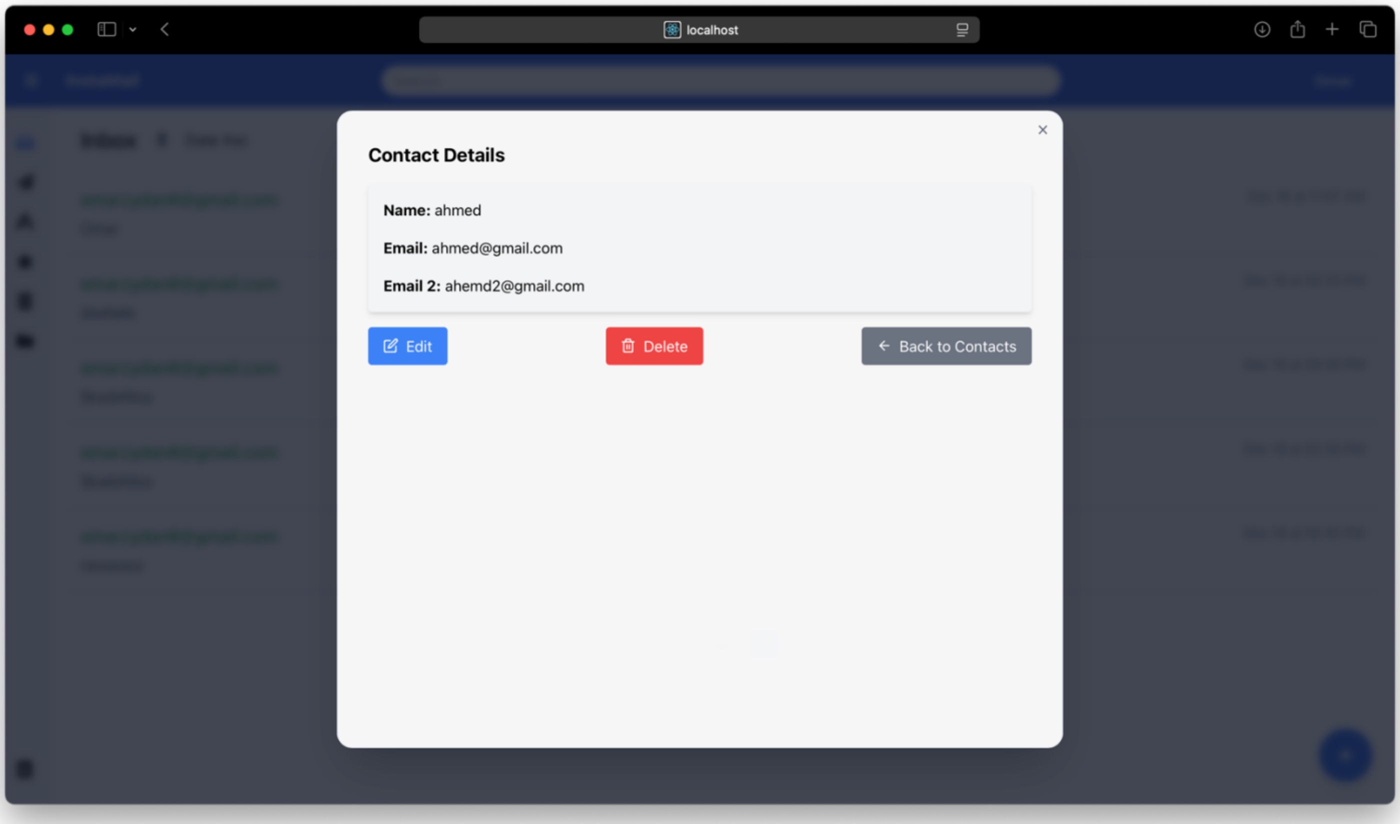
Body

Write your message here

Attachments

Add Files 0 file(s) selected





Name: ahmed

Email: ahmed@gmail.com

Email 2: ahemd2@gmail.com