

Student 1: Karim Tarek Ibrahim (20011112)

Student 2: youssef Hossam Aboelwafa (20012263)

Student 3: Nagui Mostafa Nagui (20012069)

Student 4: Abdullah Mohamed Taman (20010906)

Assignment 4 report: (C-Mail)



Description:

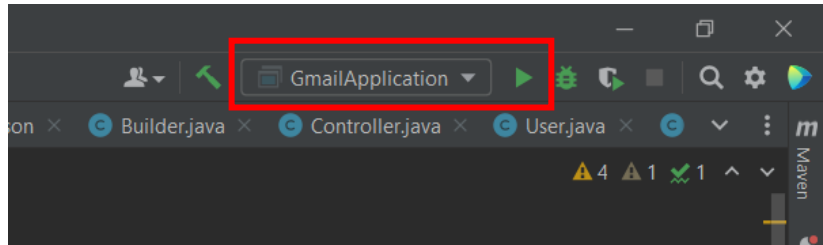
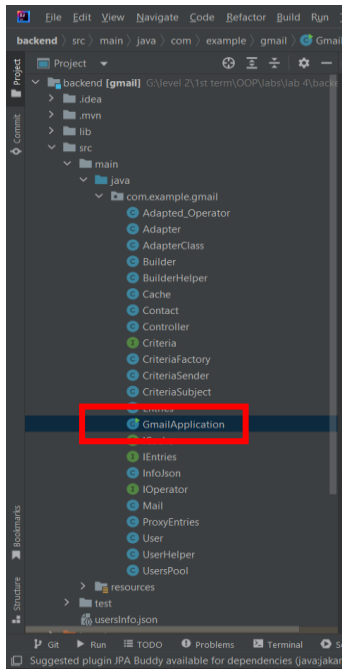
- Our project is an **object-oriented** model for a mail-server including the manipulation of mails, attachments, and contacts.
- We have applied OOP concepts that represent our model.
- Our project covers the following mail manipulation:
 - **Composing** new message.
 - **Inbox** Folder.
 - **Contacts** folder.
 - **Drafts** folder.
 - **Sent Mails** Folder.
 - **User Folders** (Adding, Renaming, Deleting).
 - **Trash** Folder.
- **Filters:** to filter mails according to subject or sender.
- **Searching** and **Sorting** based on different attributes (Date, Sender, Receivers, priority, Subject).

❖ This project is developed using **Java Spring Boot** and **Angular**

How to Run?

First step:

We run the backend from any IDE that supports JAVA (e.g., IntelliJ)



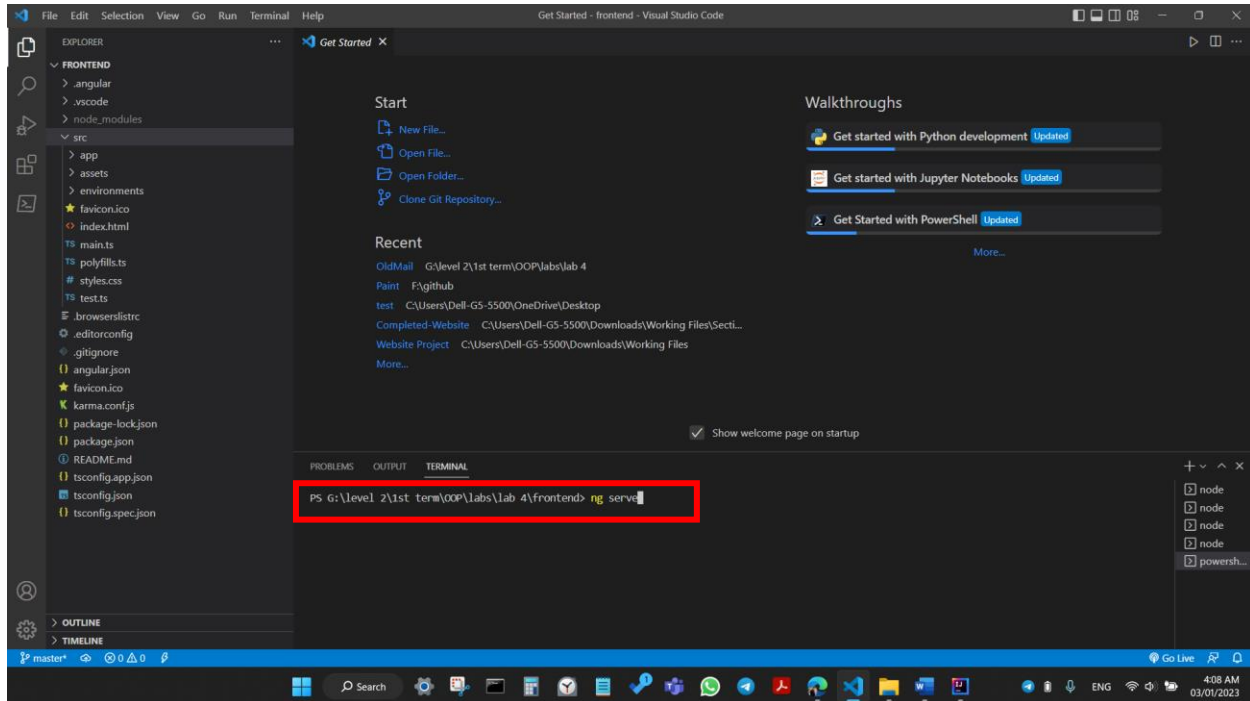
1. Select ("GmailApplication")

2. Run

Second step:

We run the frontend folder using vs code IDE

Run `ng serve` for a dev server. Navigate to `http://localhost:4200/`.
The application will automatically reload if you change any of the source files.

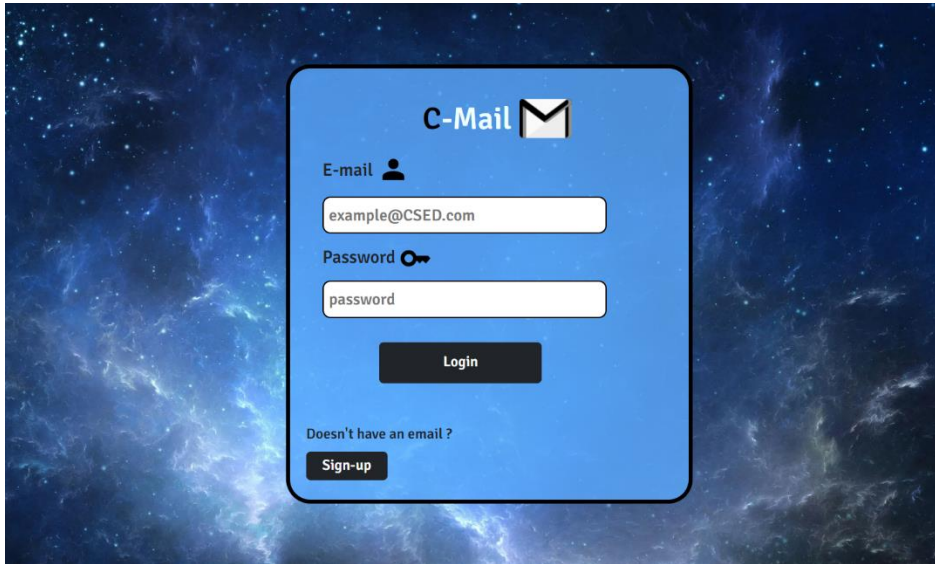


Assumptions:

- The user refreshes the page to show new content using only the refresh button implemented and not the browser refresh button
 - The attachments will only be downloaded if the user enters a correct path (a folder path) otherwise the attachments will not be downloaded
 - The user sends mails to other users signed up in the system
-

C-Mail Guideline:

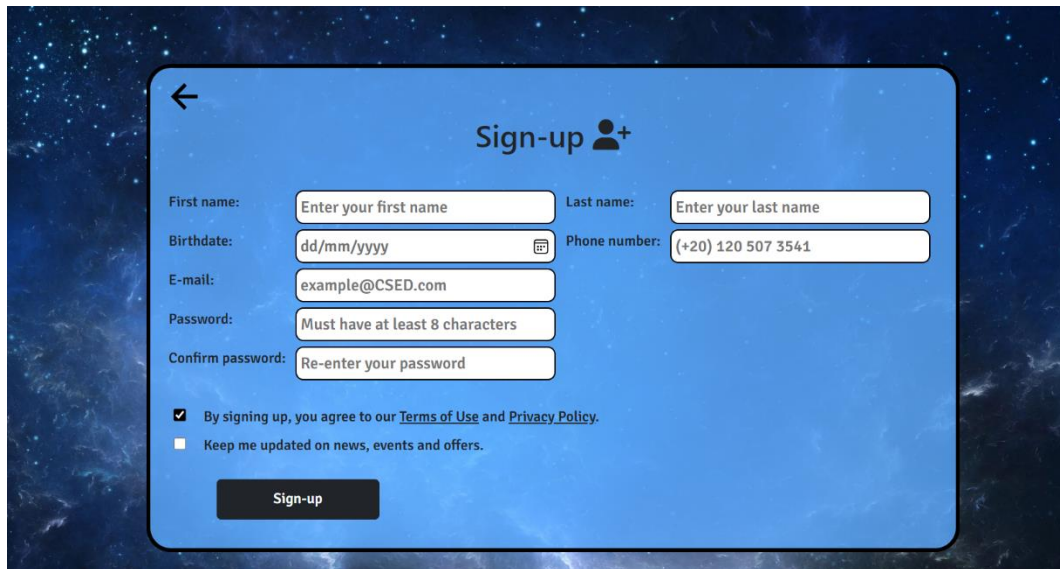
❖ Login page



For already registered users and it requires:

- E-mail
 - password
-

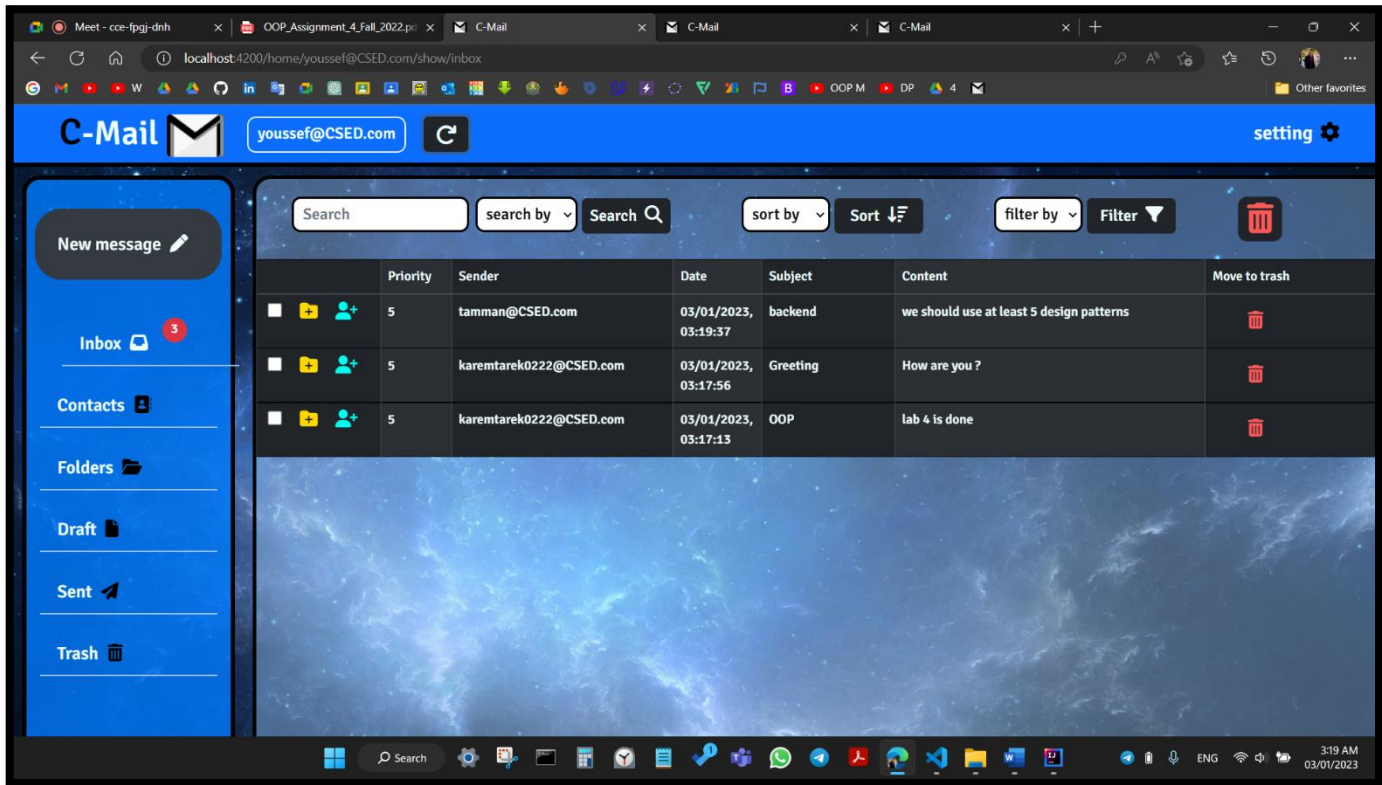
❖ Sign-up page

A screenshot of a mobile application's sign-up page. The page has a dark blue background with a starry space pattern. A white rounded rectangle contains the sign-up form. At the top left of the form is a back arrow. The title 'Sign-up' is followed by a person icon with a plus sign. The form includes fields for 'First name' (placeholder: 'Enter your first name'), 'Last name' (placeholder: 'Enter your last name'), 'Birthdate' (placeholder: 'dd/mm/yyyy' with a calendar icon), 'Phone number' (placeholder: '(+20) 120 507 3541'), 'E-mail' (placeholder: 'example@CSED.com'), 'Password' (placeholder: 'Must have at least 8 characters'), and 'Confirm password' (placeholder: 'Re-enter your password'). Below the fields are two checkboxes: the first is checked and labeled 'By signing up, you agree to our [Terms of Use](#) and [Privacy Policy](#).'; the second is unchecked and labeled 'Keep me updated on news, events and offers.' At the bottom is a black 'Sign-up' button.

For new users on C-Mail system and it requires:

- First name
 - Last name
 - Birthdate
 - E-mail
 - Phone number
 - Password
 - Confirmed password
 - Agreement to terms of use and privacy policy
-

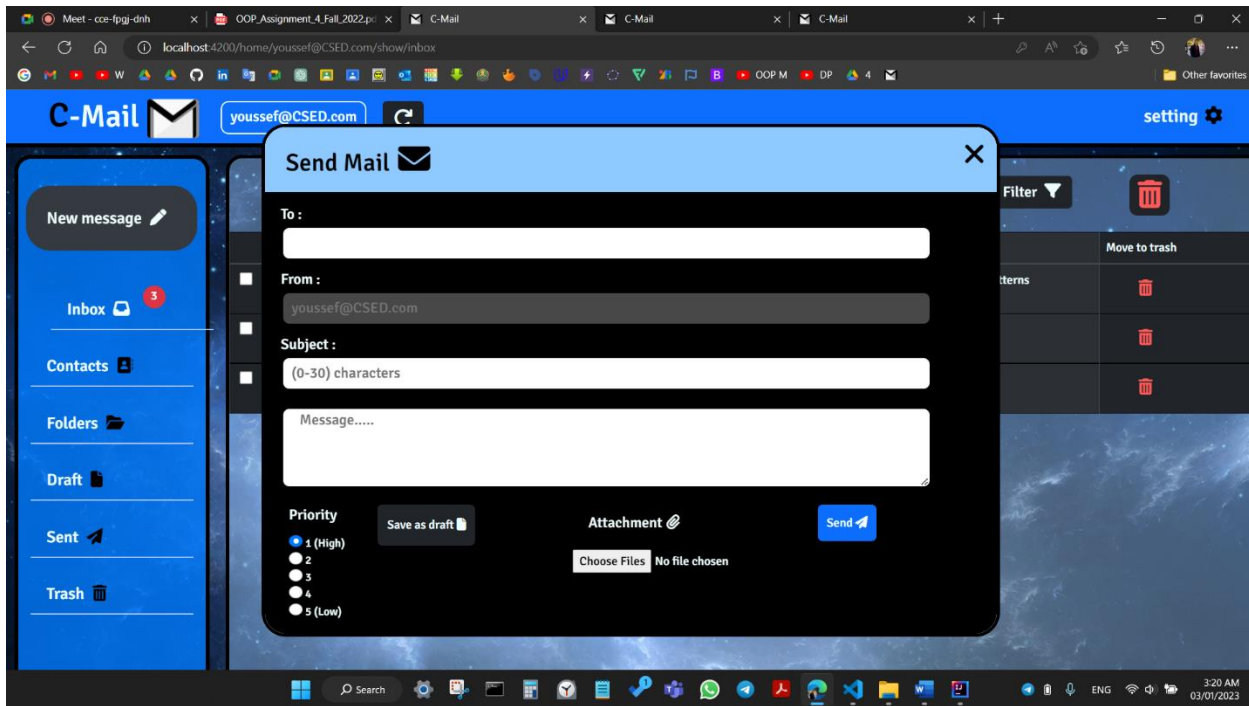
❖ Home page



The home page is divided into 3 main sections:

- 1) The title bar which includes:
 - a. The email of the current user
 - b. The refresh button
 - c. The setting option
- 2) The side bar which includes composing a new message and the following folders:
 - a. Inbox
 - b. Contacts
 - c. User folders
 - d. Draft
 - e. Sent mails
 - f. Trash
- 3) The main section that presents the selected folder and which opens the inbox folder by default.

❖ New message



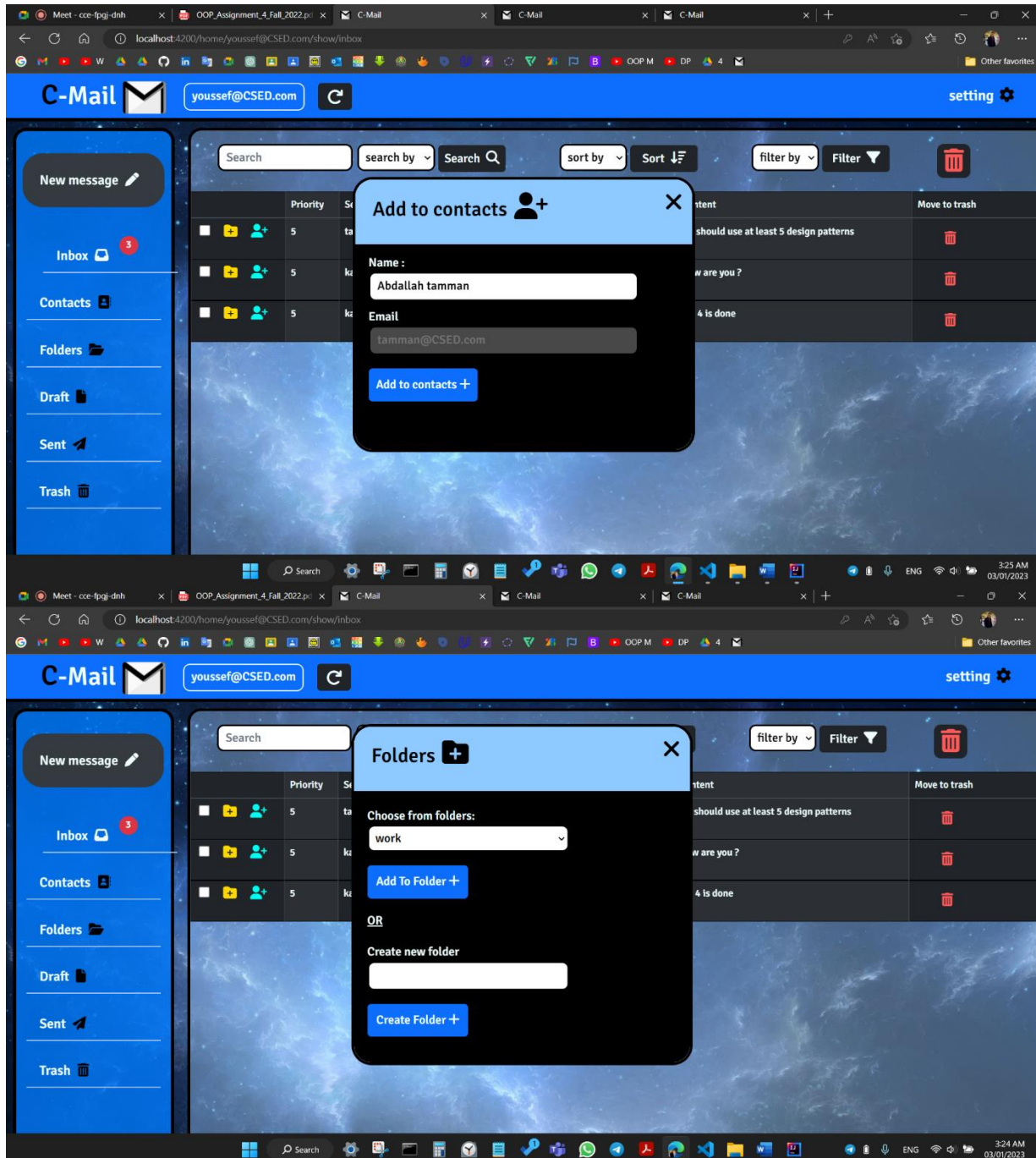
- User could send a new message to another email or multiple emails with a comma (,) then write the subject and the content
- User could select the priority of his message beginning with 1(high) to 5(low)
- User could attach a file or multiple files with his email

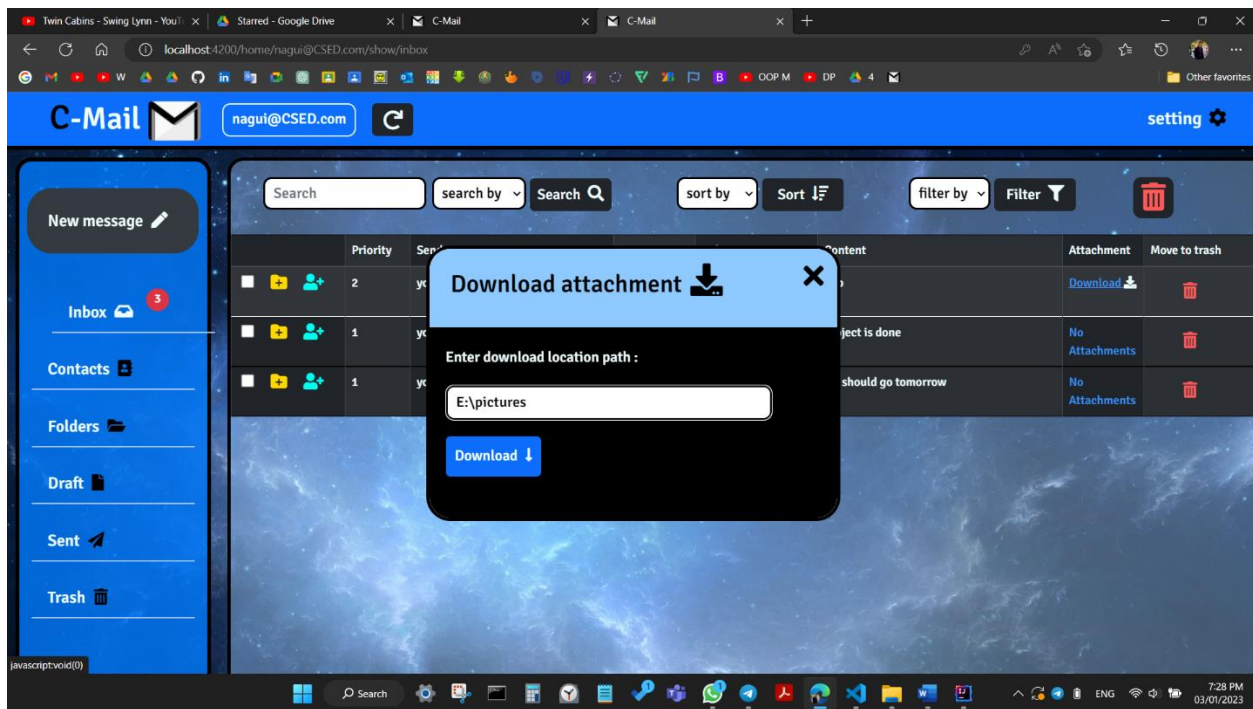
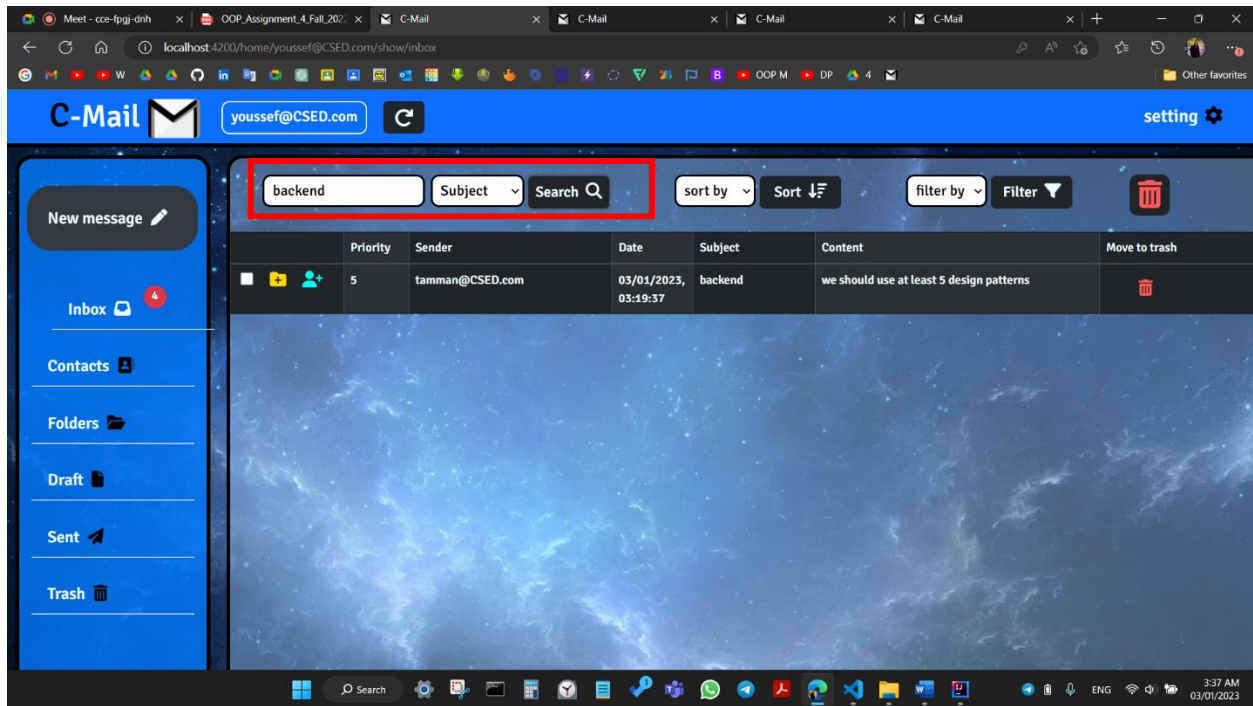
Finally, he could send his email or keep it in drafts for later

❖ Inbox

In the inbox user could:

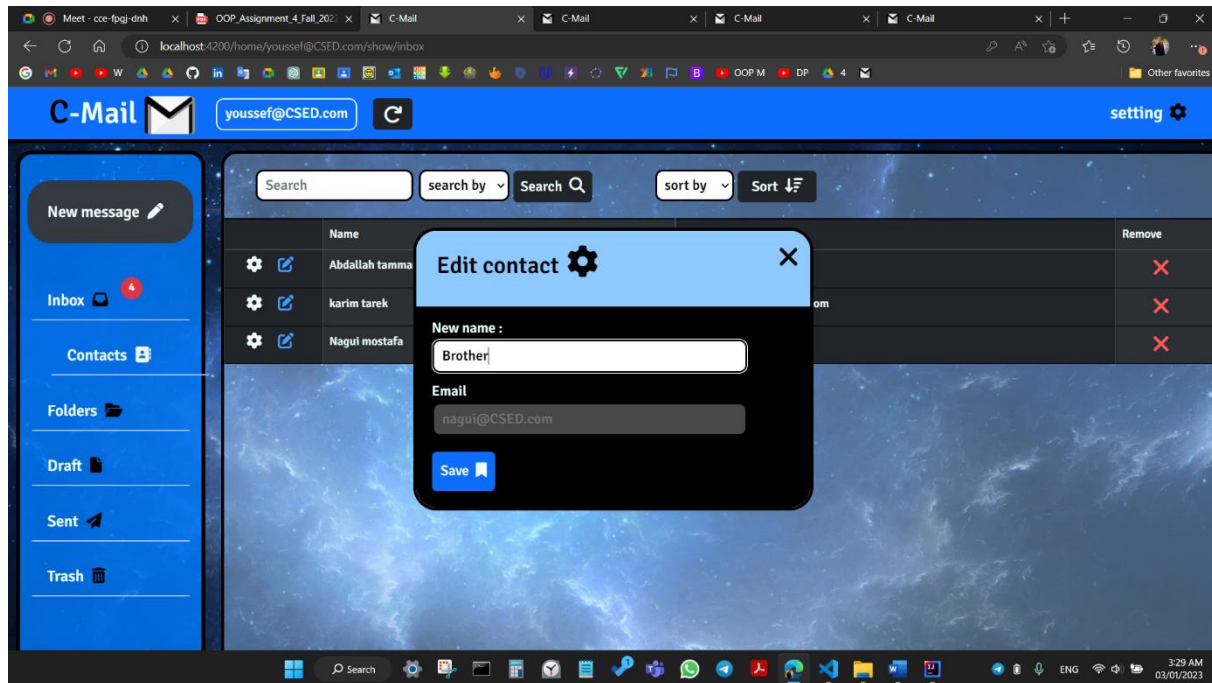
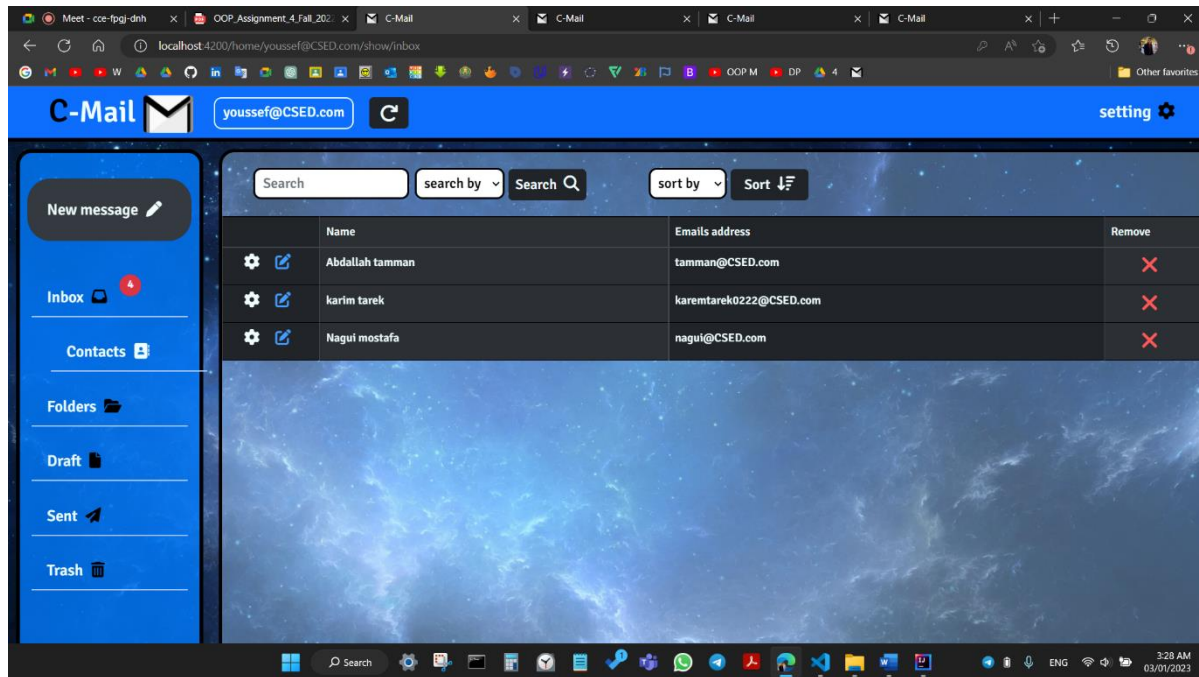
- Delete an email or select multiple emails and delete them all
- Add sender to contacts
- Add an email to user folders
- Search, sort and filter emails
- Download attachment





❖ Contacts

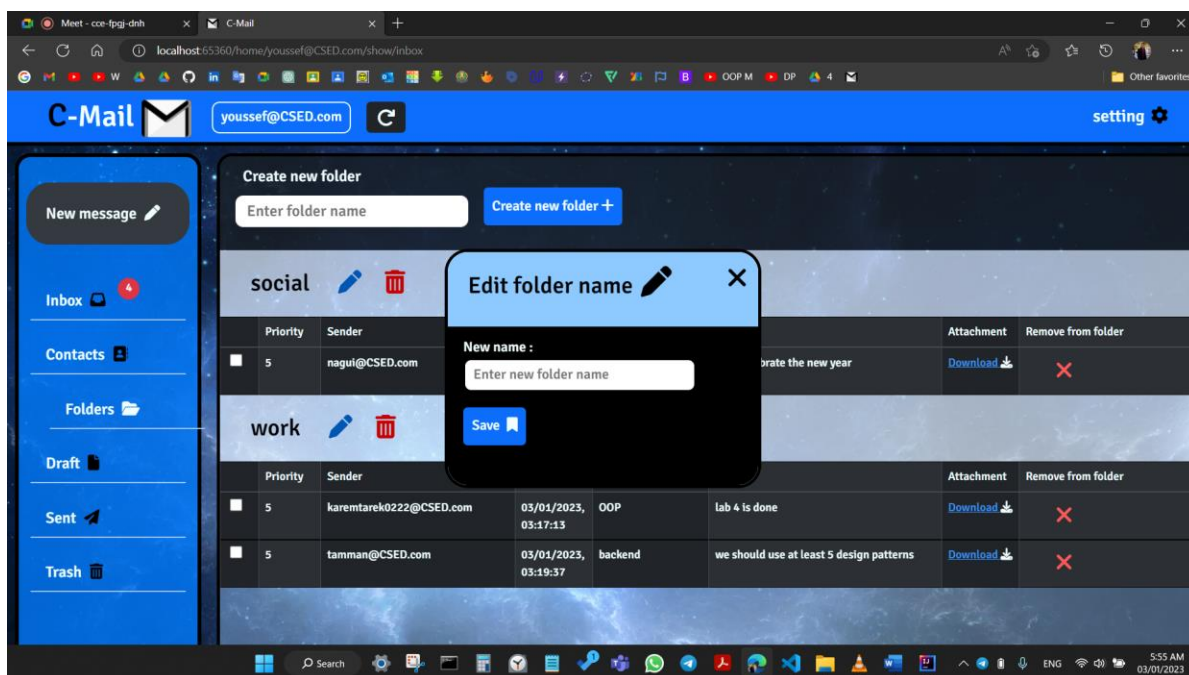
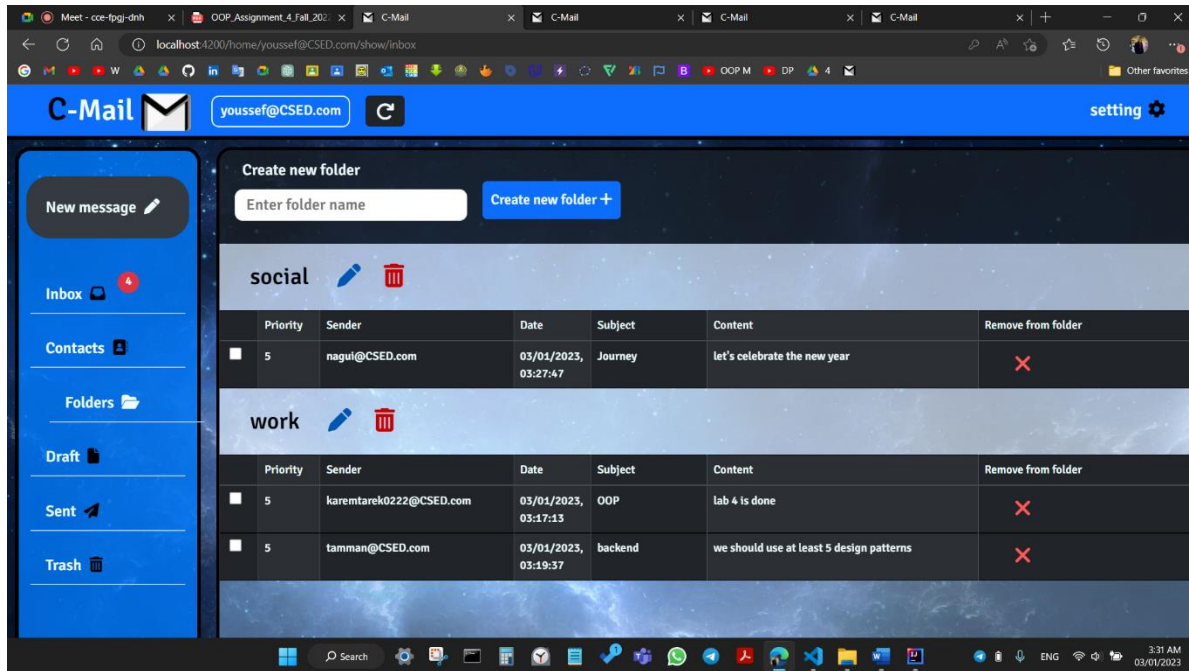
we can find in contacts all senders that we have added.
In addition to that we can send message to them and change their names



❖ Folders

In User folders we can find emails that we added from inbox

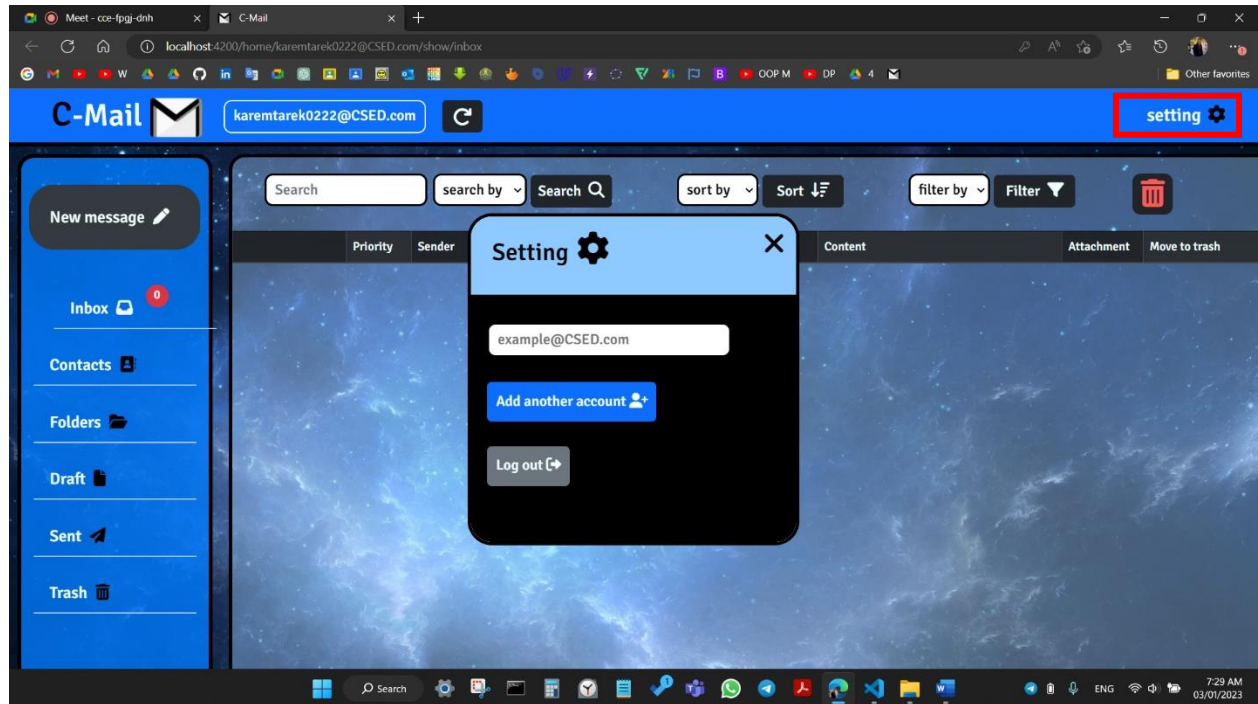
We can rename or delete every folder



❖ Setting

From setting we have 2 choices:

- 1) Add another account for the same user
- 2) Log out



Design Decisions:

- ❖ We used the required priority queue to handle the messages' priority show. We also used array lists instead of a priority queue for each attribute for the default. So, when the client wants to show the messages according to the priority we give him this priority queue, if he wants to show default, we return the array list. All the sorting stuff is handled in the default mode using the array list.
 - ❖ All the work regarding user is encapsulated in the user class, when you need to do something to a user you need to use the service that the user class provides.
-

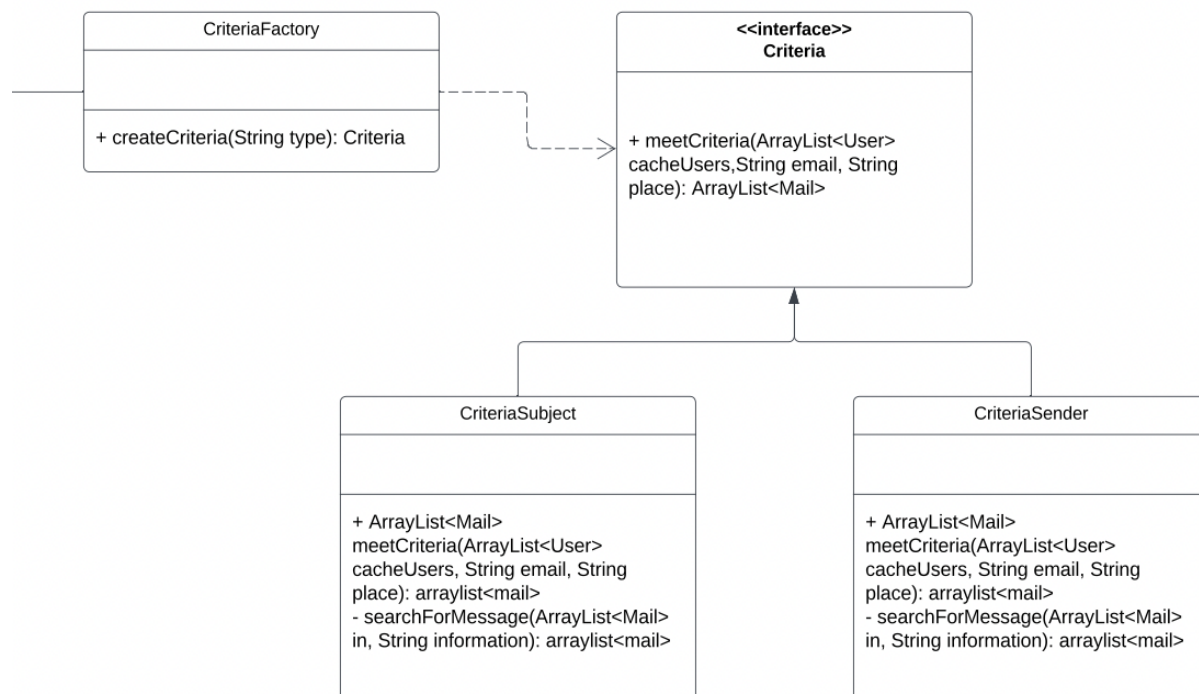
How did we implement the Design patterns?

1.Filter:

Criteria interface and concrete classes implementing this interface were created to filter list of *Mail* objects. *Operator* class uses *Criteria* Factory class to create *Criteria* objects to filter List of *Mail* objects based on Senders and Subjects.

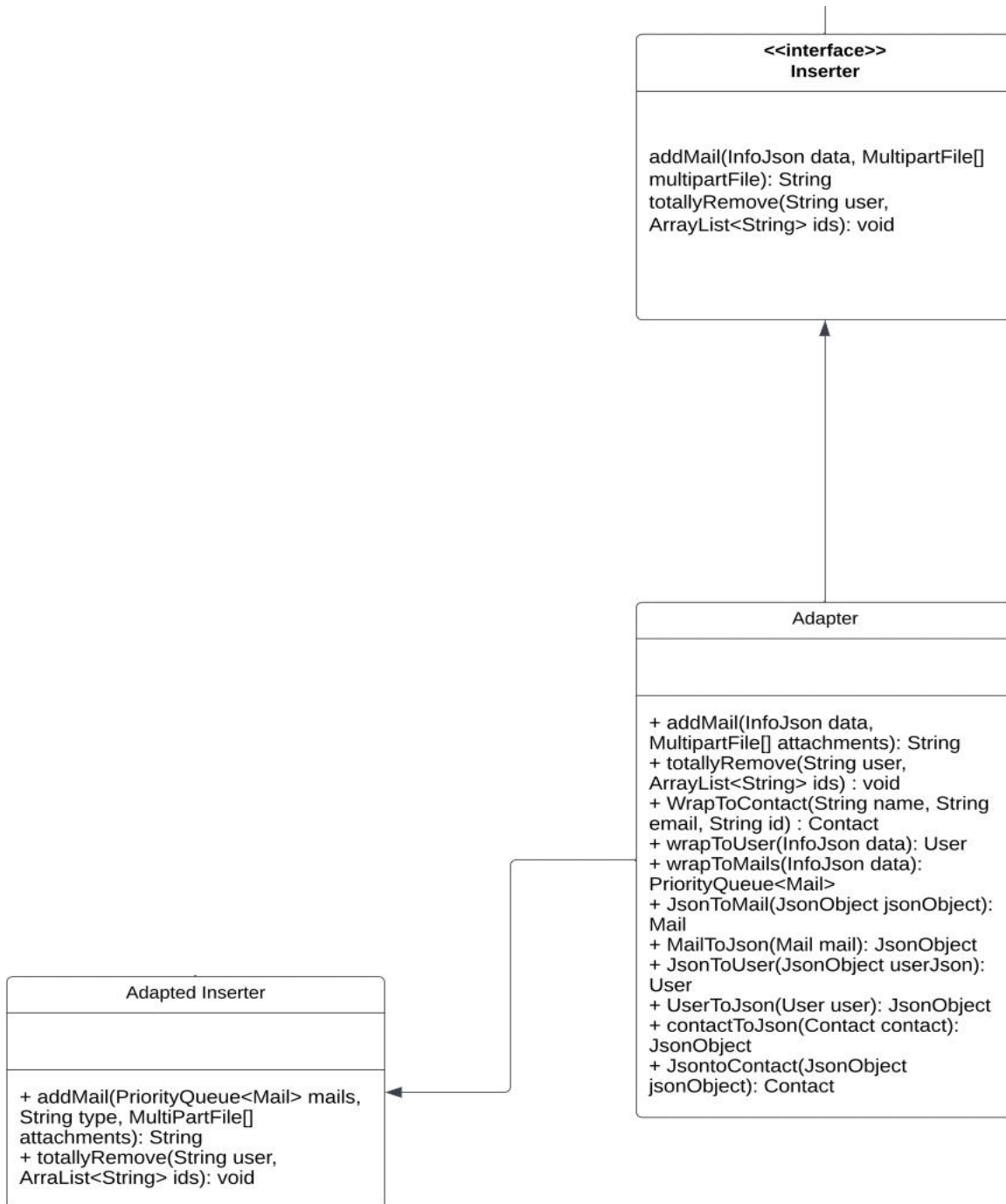
2.Factory:

CriteriaFactory class to create a specific *Criteria* object (*CriteriaSender* or *CriteriaSubject*) based on the type required



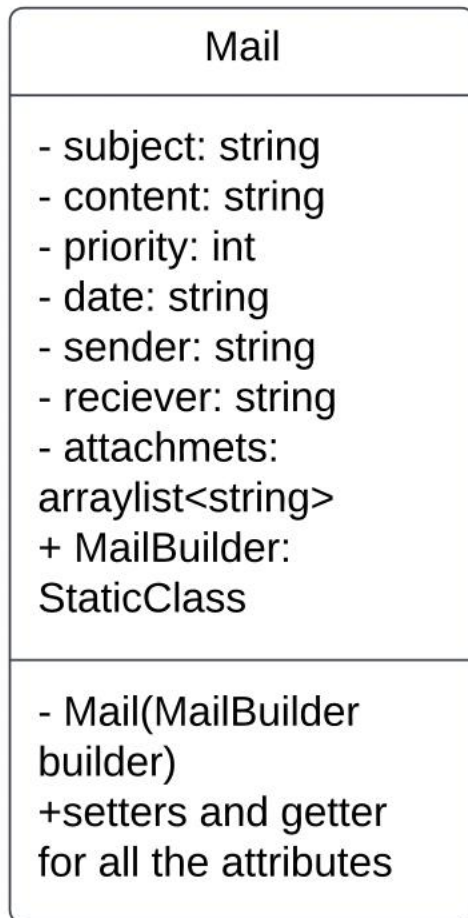
3.Adapter:

Class Adapter implementing the interface Inserter is used to convert the objects received in form of InfoJson to The required form (User , Mail) to be able to perform specific operations on them in the Adapted Inserter class



4.Builder:

A Mail builder class was implemented as an inner static class in the Mail class and is used to build Immutable Mail objects with required and optional attributes.

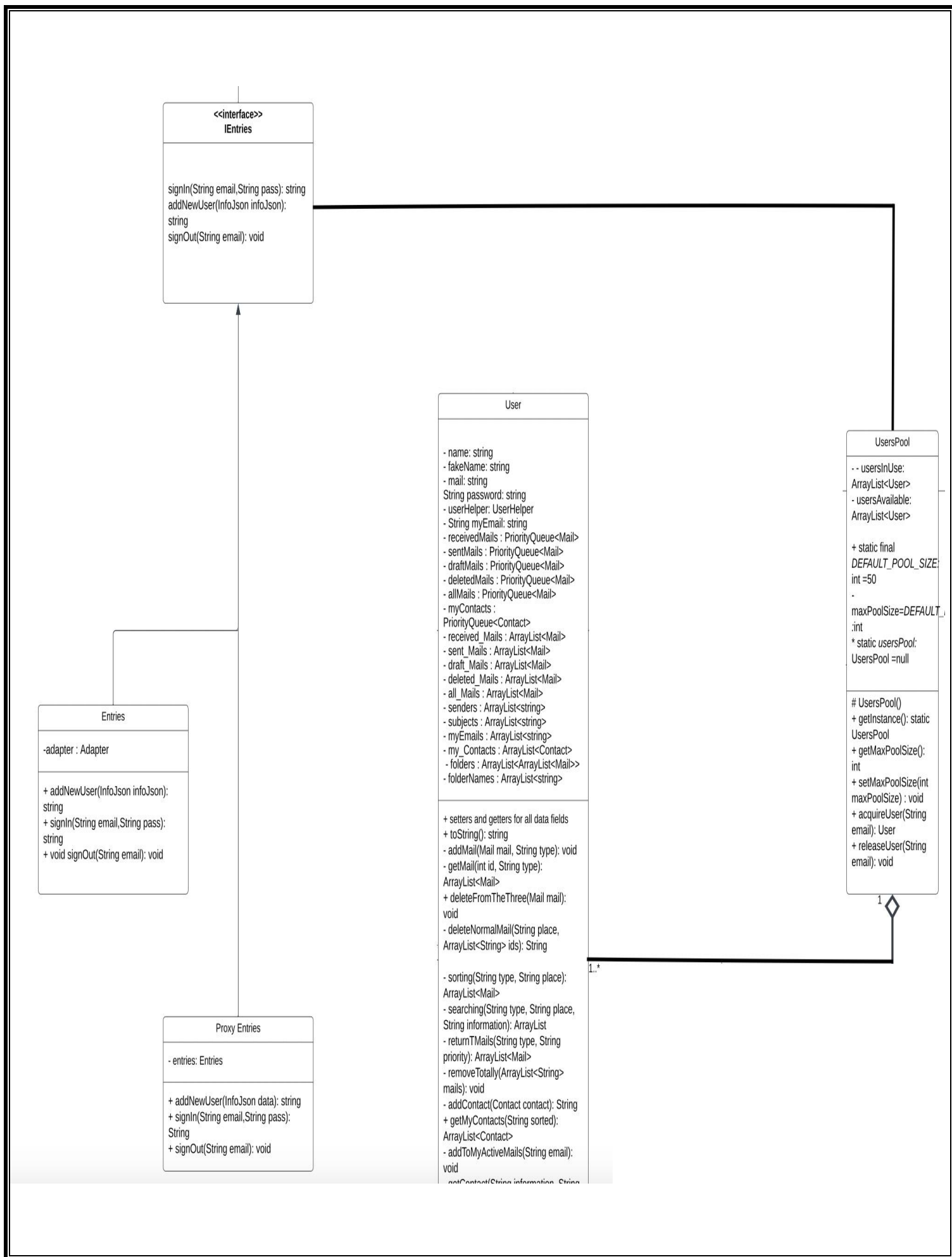


5.Proxy

Proxy Entries is a **Protection Proxy (Access Control)** class implementing the interface **IEntries** responsible of the **Sign in** and **Sign-Up** operations. It is used to control access of the users to the system

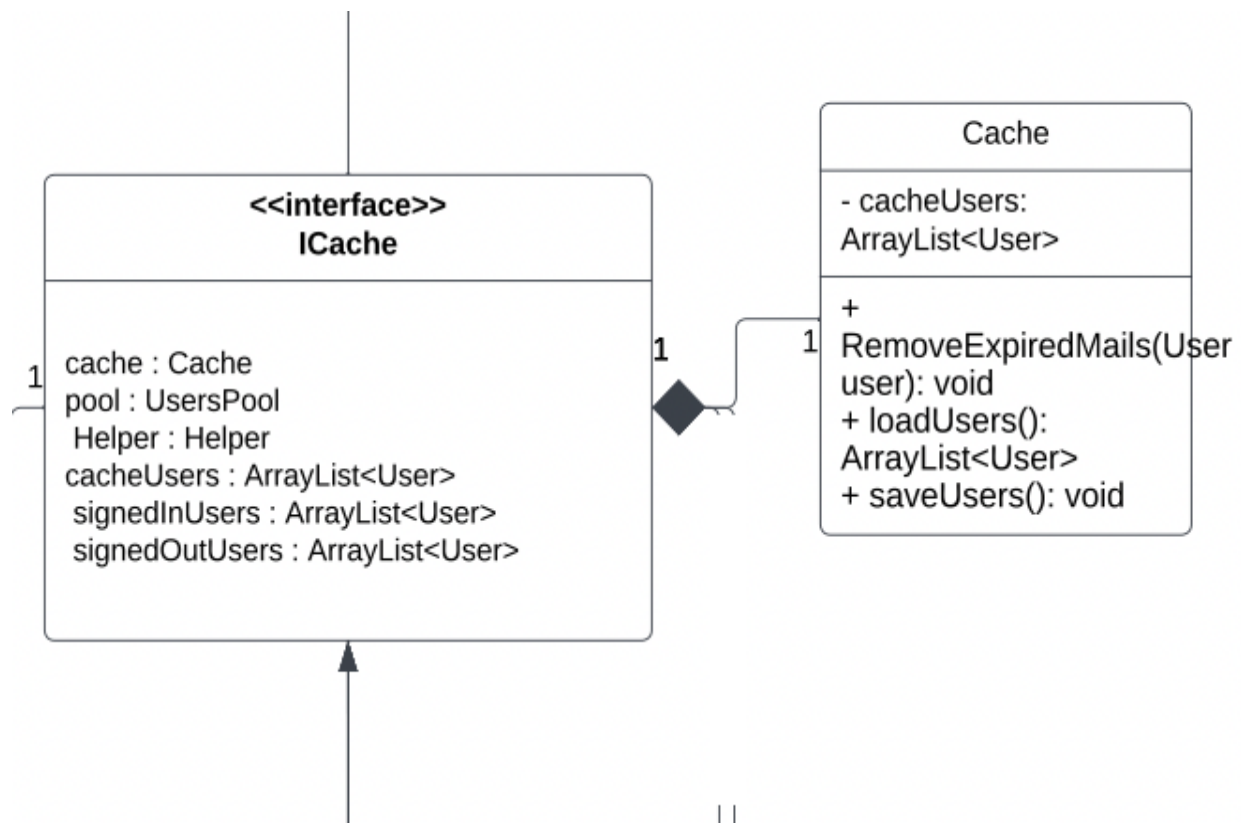
6.Object Pool

A **Users Pool Class** is implemented. It contains a list of available and in use **Users** objects to limits the maximum number of **Users Objects** that can be created to make sure performance is not affected



7.Singleton:

A singleton class Cache is implemented. It is responsible of the save and load operations where the attributes of the single Cache Objects work as a database



A descriptive UML class diagram for the project:

https://drive.google.com/drive/folders/15a_FNRQDDKIYEEYa4TdSt4U2P4JDUyo4q?usp=share_link
