Mail Server: Xmail

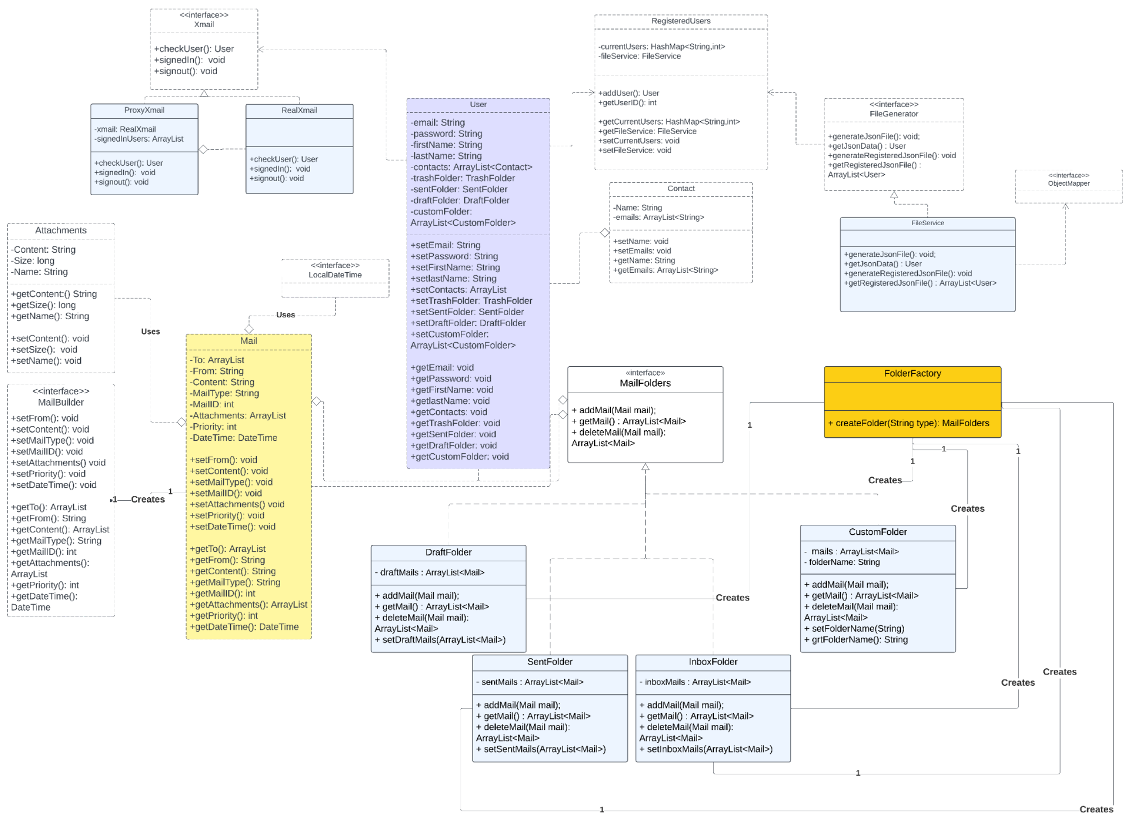
Team Members:

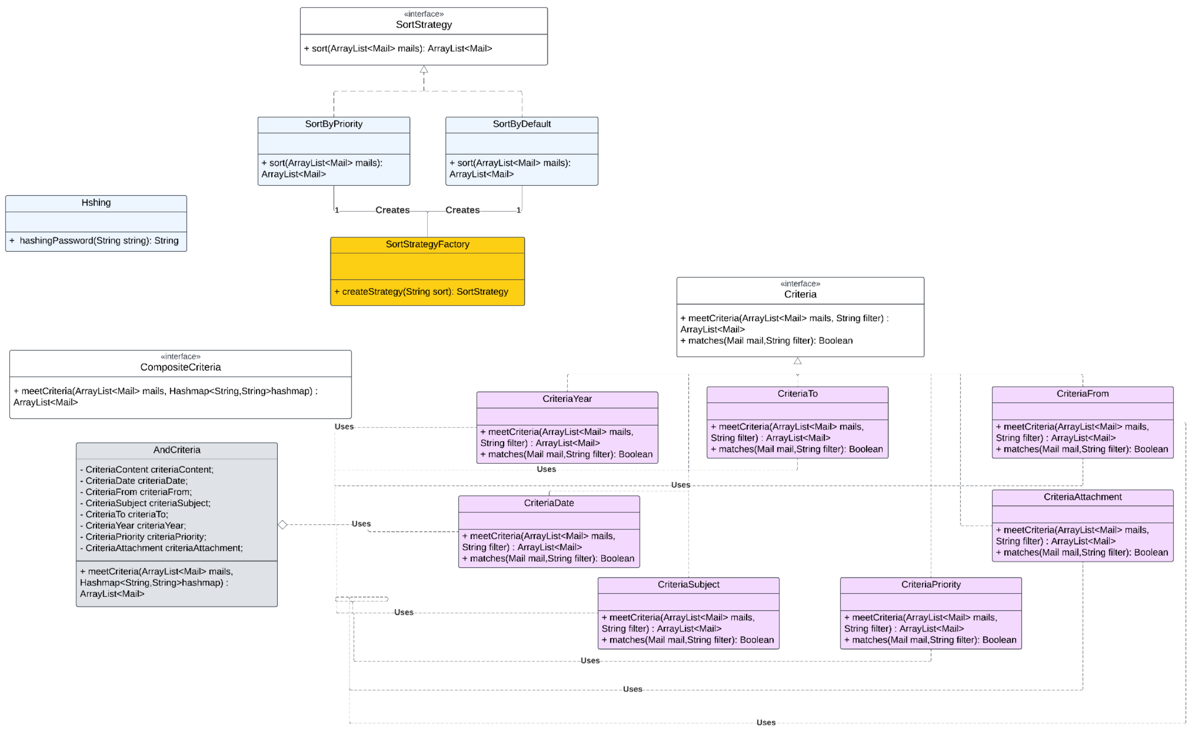
1. **Rowan Gamal Ahmed Elkhouly** - 21010539
2. **Yomna Yasser Sobhy** - 21011566
3. **Yousef Mahmoud Mohamed Ahmed** - 21011630
4. **Mustafa Mohamed Elkaranshawy** - 21011375

Steps of running the code:

- The backend and the frontend files should both be installed with all the included dependencies and packages
- File **.env** should be created in the spring-boot folder with your machine paths variables, you can use **.env.example file** for illustration
- The backend file should run first as we made it run on a default port of 8080
- The frontend file should be run afterwards, and the local host is opened in the browser

UML diagrams:




«interface» SortStrategy
+ sort(ArrayList<Mail> mails): ArrayList<Mail>
SortByPriority
+ sort(ArrayList<Mail> mails): ArrayList<Mail>
SortByDefault
+ sort(ArrayList<Mail> mails): ArrayList<Mail>
Creates
Creates
1
1
SortStrategyFactory
+ createStrategy(String sort): SortStrategy
Hshing
+ hashingPassword(String string): String
«interface» Criteria
+ meetCriteria(ArrayList<Mail> mails, String filter) : ArrayList<Mail>
+ matches(Mail mail,String filter): Boolean
«interface» CompositeCriteria
+ meetCriteria(ArrayList<Mail> mails, Hashmap<String,String>hashmap) : ArrayList<Mail>
AndCriteria
- CriteriaContent criteriaContent;
- CriteriaDate criteriaDate;
- CriteriaFrom criteriaFrom;
- CriteriaSubject criteriaSubject;
- CriteriaTo criteriaTo;
- CriteriaYear criteriaYear;
- CriteriaPriority criteriaPriority;
- CriteriaAttachment criteriaAttachment;
+ meetCriteria(ArrayList<Mail> mails, Hashmap<String,String>hashmap) : ArrayList<Mail>
CriteriaYear
CriteriaTo
CriteriaFrom
CriteriaDate
CriteriaAttachment
CriteriaSubject
CriteriaPriority
+ meetCriteria(ArrayList<Mail> mails, String filter) : ArrayList<Mail>
+ matches(Mail mail,String filter): Boolean
Uses

For higher Resolution:

- https://lucid.app/lucidchart/c98149b5-887b-4cd2-9554-f13417fee89f/edit?viewport_loc=-2767%2C-1516%2C3345%2C1547%2C0_0&invitationId=inv_cbfda2c8-cc0a-4757-a9e5-1ff885bddae3
- https://lucid.app/lucidchart/97dc0250-11c8-4720-a4e0-37e474d89d88/edit?view_items=X68Zu67QspdL&invitationId=inv_8c358e62-df4a-4d60-b9de-c68f2c0561e8

DESIGN PATTERNS

Design Patterns Applied:

1. **Factory (in Folder Creation)**
2. **Singleton**
3. **Builder**
4. **Filter**
5. **Factory (in Strategy Sort)**
6. **Proxy**
7. **Strategy**

How We Applied Required Design Patterns:

1. **Factory (in Folder Creation)**

Since the user can have multiple folders, we needed a Creational design pattern that creates objects upon runtime, because we don’t exactly know the type of class needed.

2. **Singleton**
   The singleton pattern ensures that a class has only one instance and provides a global point of access to that instance.
   We used singleton in all folders’ creation for every single user, since per user we want to ensure one folder is not instantiated every time the user wants to add an email to it.
3. **Builder**
   The builder pattern is used to construct complex objects, allowing for the creation of different configurations of a mail object without the need for a large number of constructors or complex parameter lists.
   This pattern is especially used for mail class, when a mail object has optional parameters, attachments, or varying formats, providing a clear and flexible way to create different types of mail instances while improving readability and maintainability in the code.
4. **Filter**
   The filter design pattern is used to enable the efficient and flexible filtering of objects based on **various criteria** without modifying the underlying object structure.
   In the context of filtering an array of mail objects, this pattern allows users to apply different filters (such as sender, date, subject, etc.) dynamically and combine them as needed, providing a modular and extensible way to manage complex filtering requirements without altering the core logic of the mail objects or the filtering process.
   The filter also helped us specifically in using its Criteria for general **search** in our mail server
5. **Factory (in Strategy Sort)**
   Factory for the creation of Sort class which sorts the list of mails.
6. **Proxy**
   Proxy for **Xmail** to prevent everyone from having access to the system unless he is already signed in.
7. **Strategy**
   The strategy design pattern is used in the context of sorting an array of mail objects to define a family of algorithms (strategies) for sorting and making them interchangeable. This allows the client code to dynamically select and apply a sorting strategy based on priorities or default criteria without modifying the sorting logic in the mail objects themselves.

Code Ideation and Data Structures Used:

- **Database Structure and Design Decisions:**
  a. We came up with the optimal solution of saving every user in a separate JSON file with a path name relevant to its ID
  b. How to get ID? We save **registered user as keys in a hashmap** and their ID values.
  c. Whenever a user logs in, we fetch its Key in O(1) through hashmap, and by directly accessing its file path, we object map the JSON file into our currentuser object in Service class

- **Version Control Used:**
  a. Git (Used git ignore for environment variables" paths")

- **Security and Privacy:**
  a. We used Hashing of "SHA-256" for hashing password before saving it in our JSON files.

- Any time/space complexities were taken into consideration

- **Data Structures Used:**
  a. Arraylists for mails in every folder
  b. Arraylist of type Attachments in Mail Class
  c. Arraylist of Custom Folders
  d. Arraylist of Contacts
  e. Hashmap for user retrieval
  f. Hashmap for filtering based on which criteria
  g. PriorityQueue for sorting according to Priority

- **Java Interfaces Implemented**
  a. Comparable (in contacts for sorting, in mails for priority sorting)
  b. Serializable (for easy deserialization)

Xmail Features:

- User authentication and authorization:
    - Sign Up
    - Sign In / Log in
- Mail Features:
    - Composing mails
    - Sending mails
    - Drafting mails
    - Trashing mails
- Folder Features
    - Having mails organised in folders
    - Creating Custom Folders
    - Editing, adding, moving any mail from one folder to another (Including our default primary folders)
    - Sorting mails (<u>in any folder:</u> Default / Priority)
    - Trash is auto-deleted after 30 days
    - Restore from Trash to the folder it was trashed from
- Usability of our system for user
    - Bulk move
    - Bulk Trash
    - Bulk restore
- Contacts Features:
    - Creating Contacts
    - Renaming, adding emails to the same contact, and editing contacts
    - Sorting Contacts (Default / Alphabetically)
- Filter and Search Functionality
    - Search without determining any filter:
        - **Searches in all of our criteria: (Date, Sender, Receivers, Priority, Subject, Content, Attachments, Year)**
        - **Search is not static, any part of the string would still work**
    - Filtering:
        - **Single Filtering/ Multi-filtering** on all of our criteria: (Date, Sender, Receivers, Priority, Subject, Content, Attachments, Year)
        - Allowing of multi filtering (AND criteria)
        - Users can add filters as much as they want, all would be filtered accordingly
        - Bulk move/select filtered mails and move to/create a new folder
- User Interface Features:
    - Routing
    - Pagination in every folder
    - Scroll bar for extra terms in the sidebar
    - Icons for delete, restore, etc.

    - More discussed in design decisions.
- Website User handling:
    - By the use of caching, we can allow users to use our system at the same time (but on different browsers of course)

Bonus Features:

- Restore for all types of mails
- Bulk restore
- Search in all fields
- Search by letter, word, term, or any co-string
- Multi and single-filtering
- *Extra* Year filtering

Design Decisions:

- Sidebar Placement:
  The sidebar is positioned on the left side of the window.
  Facilitates easy navigation for users to identify their current page and access folders quickly.
- Compose Option:
  The "Compose" option is placed on top of the sidebar for easy access.
- Compose Area Features:
  Compose area has minimized (-) and close (x) options.
  Users can minimize, close, and maximize the compose box as needed.
- Attachments Icon:
  An attachment icon is provided for users to upload files from any location.
- Priority Dropbox:
  A priority dropdown is included, ***marked with ** to emphasize its importance before sending***.
- To Field Validation:
  ** is displayed in the "To" field, indicating it must be filled correctly.
  Error messages are shown for any mistakes made during composition and sending.
- Draft Editing:
  Compose textarea opens when clicking on a draft message for users to continue editing.
- Create Folder Option:
  Users can create custom folders by clicking on the labels bar and entering a unique name.
  Checks are implemented to prevent the repetition of custom folder names.

- Message Handling Icons:
    Each message has a delete icon for quick removal.
    The trash page includes a restore icon for recovering deleted messages.
- Custom Folder Management:
    Rename and delete icons are provided for each custom folder created by the user.
- Folder Deletion Confirmation:
    An alert is triggered when attempting to delete a folder, ensuring the user confirms the permanent action.
- Default Inbox Display:
    The inbox folder is displayed by default, notifying users of new messages upon opening the app.
- Search and Filter Bar:
    A search bar is placed at the top of the page for easy message retrieval.
    A filter icon allows users to narrow down messages based on specific criteria.
- Filter Options:
    Messages can be filtered by various criteria such as to, from, subject, content, attachments, name, date, and year.
- Select All Option:
    A select all box is provided to select all messages on a page for bulk actions (move, trash, restore).
- Move Box:
    A “move” box displays custom folders for users to choose where to move a bulk of messages.
- Sort Options:
    Users can choose the order in which messages are displayed (default - newest to oldest, or by priority).
- Page Navigation Symbols:
    Symbols are added to navigate between pages within a folder for an organized user experience.
- Contacts Page Button:
    A button on top of the contacts page allows users to access it easily.
    A sorting dropdown is included for arranging contacts alphabetically.
- Multiple Emails for a Contact:
    Users can add more than one email for the same contact.
- Contact Management:
    Editing and deleting contacts is facilitated through options accessed by clicking on dots next to contact info.

Xmail User Guide 101:

System in a nutshell:

- ➢ The Xmail platform requires users to either sign up as new users or log in if they already have an account. Upon signing in, users encounter a sidebar featuring categories (inbox, sent, draft, trash) and folders. Within each category, emails are displayed by default in chronological order, with the option to prioritize them. Each email on the home page shows details such as sender, recipient, date, subject, and priority.
- ➢ Opening an email reveals all its details on a full screen, including sender, recipient, subject, email content, and attachments. To compose a new email, users click the compose button in the sidebar, where they can add various details. However, specifying recipients and priority is mandatory for sending, and optional for saving as a draft. Users can edit drafts, save as new drafts, or send them if they meet the necessary requirements.
- ➢ Deleting emails is possible through the trash icon, and multiple emails can be selected for actions like moving to a folder or deletion. Emails in the trash can be restored to their original folders or permanently deleted. Users can create, edit, or remove folders/labels. The platform also provides access to contacts through a dedicated button, allowing users to add, remove, and edit contacts as needed.

By-book Guide of Internal Functionalities:

Registration:

- Users of our web application must be logged in to use our service, so filling all the required sections in the signup form is a must.
- An age of more than 18 is required in the date of birth section.
- **All emails must end with "@Xmail.com"** and be **a unique email** (if the email is repeated an indication of that appears).
- If the user already has an account, he can directly log in by entering email and password in the login form.
- The logout button at the top left of the sidebar returns the user to the home page and closes his account

Compose:

- To send a message, the user chooses the compose button that opens the sections of the message to be filled.
- The message must contain at least 1 valid Xmail (that ends with @Xmail.com) to be sent to.
- The user must choose a priority for the message as an indication to the receiver how important the message is (with 1 being the highest priority).
- Other sections of the message are optional.

- Two red stars next to the input field not filled or filled incorrectly.
- For entering multiple receivers, comma-separated Xmails are entered in the "to" section.
- To add one or multiple attachments to the message, please use the small clip button at the left bottom section of the compose box.
- The messages that the user sent are found in the sent section in the side bar so that the user can view.

Drafts:

- If you made a message and didn't want to send it, you can save it as a draft using the "draft" button in the compose section.
- Draft messages are found in the draft section which is accessed by pressing the "draft" from the sidebar.
- By pressing on a message in the draft section, the compose box appears with the old data held by that message; now you can modify it again and send it, send it as it is (if required fields were filed), or save it again as a draft with or without editing (if edited and saved as draft, the old data in the message is overwritten so no new draft is created).

Inbox:

- Received messages are found in the inbox section in the sidebar.
- Newly received messages appear after refreshing the website ONLY.
- In the inbox and all other message sections, there is a box that allows the user to sort the messages by priority or by the date of message creation (default).

Trash:

- Any message can be deleted by either the delete icon next to it or by selecting it and then pressing on the delete all icon at the top of the page.
- All deleted messages are found in the trash section in the sidebar.
- Restoring messages from the trash returns them to the folder they were deleted from.
- Deleting messages from trash deletes them PERMANENTLY.
- Draft messages that are sent to trash are still opened in the compose box but can't be sent (either save as a new draft or restore it to be able to send it).

Contacts:

- The user can save some emails with a name in the contacts section to keep them from being lost.

- Saved contact names must be unique so no two contacts can have the same name.
- Contact emails must be valid (ends with @Xmail.com).
- Each contact can have multiple emails (entered separated by commas).
- Contacts can be renamed or deleted, and these two options appear by pressing on the 3 dots on the right of the contact.
- There is a box that allows the user to sort the contacts alphabetically or by the date of contact creation (default).

Custom Folders (Labels):

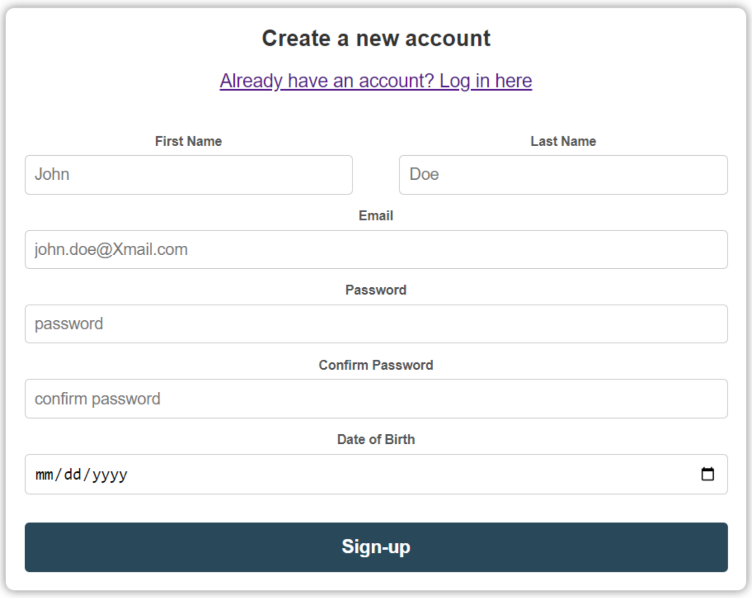
- The user can move messages to existing labels by the move button at the top of the page.
- Sent and inbox folders are the only ones that support moving to labels.
- The user can create a new label by pressing the + sign in the sidebar to put some messages in according to his choice.
- Messages put in the label can't return to sent or inbox; they can be either deleted or moved to another label.
- Deleting a custom folder deletes all the messages inside it so be careful.

Searching and Sorting:

- The search icon at the left of the input box at the top of the page is used to search all parts of the message of the entered word and returns all the matches.
- When the filter icon at the right of the input box at the top of the page is pressed, it opens multiple filter criteria to allow the user to filter the results of the section he is in as he pleases (if he is in the inbox section, only the inbox section is filtered).
- Note that the filter gets each message that all filter criteria are in it so if one is not in the message it's not selected.

SnapShots of System:

Sign up


Create a new account

Already have an account? Log in here

First Name	Last Name
John	Doe

Email

john.doe@Xmail.com

Password

password

Confirm Password

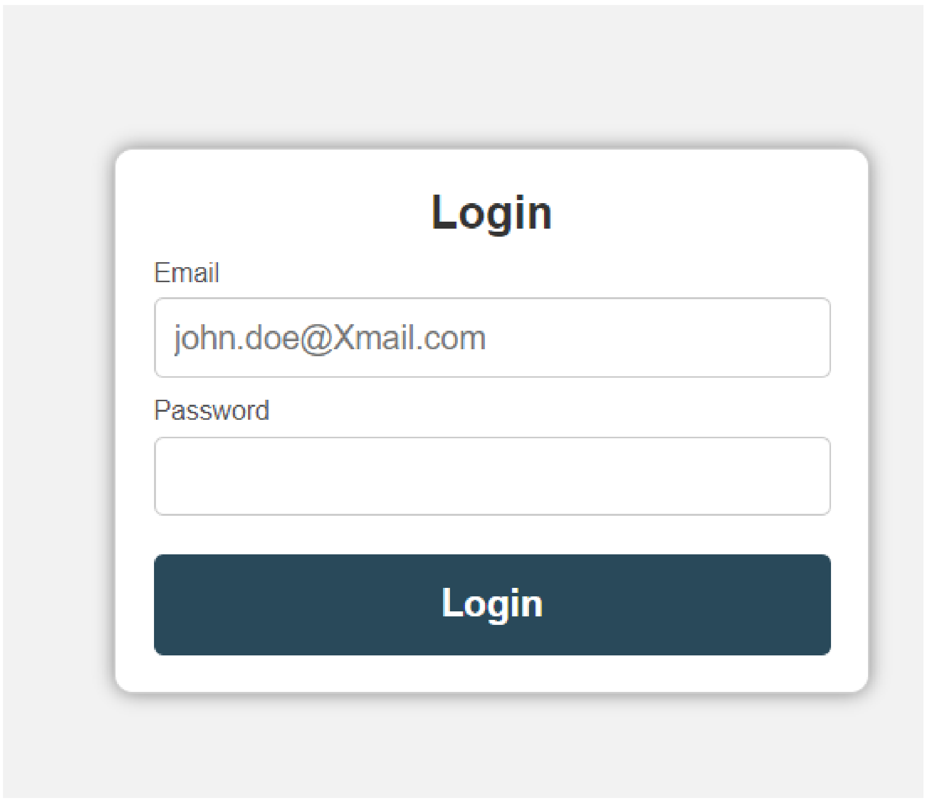
confirm password

Date of Birth

mm/dd/yyyy

Sign-up

Sign in


Login

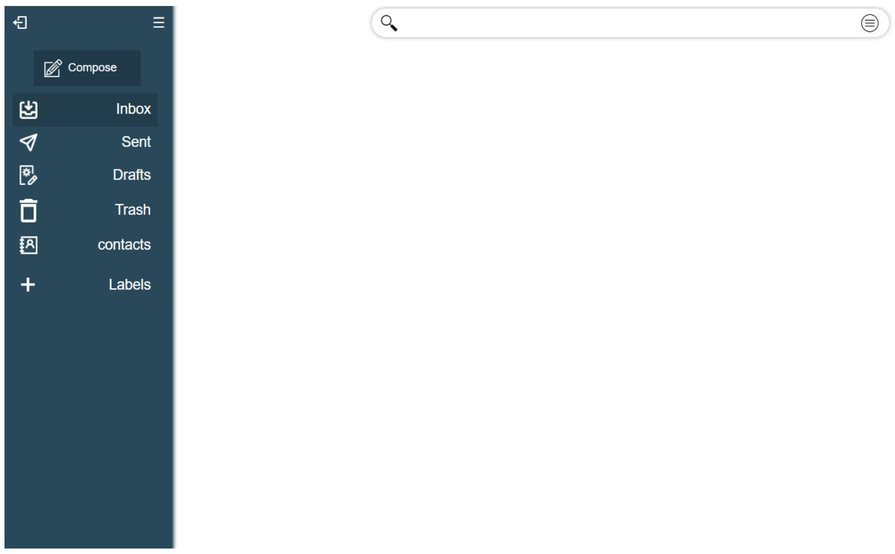
Email

john.doe@Xmail.com

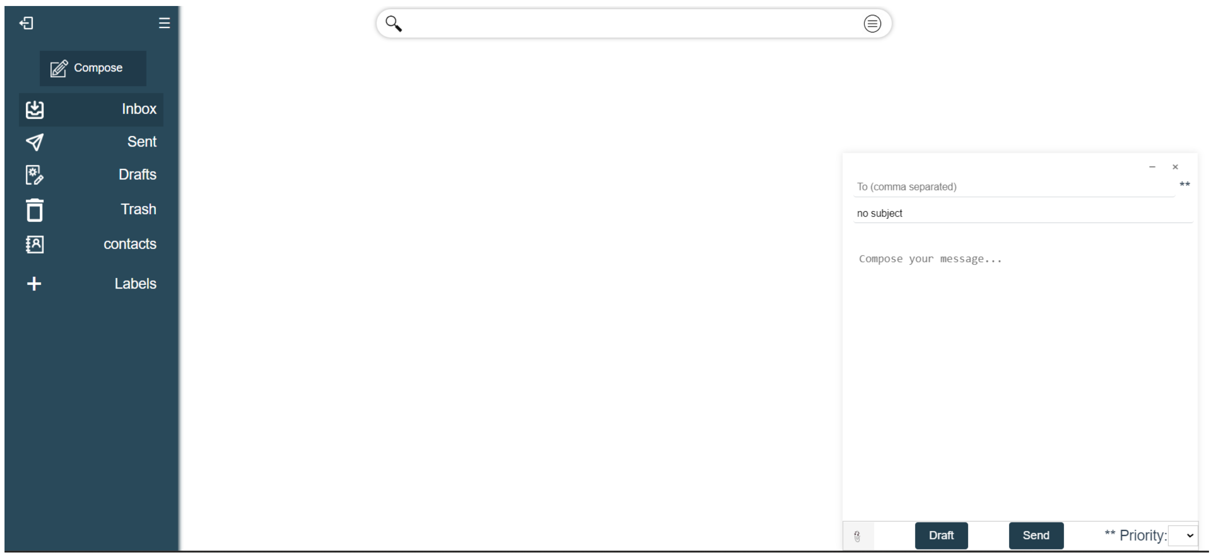
Password

Login

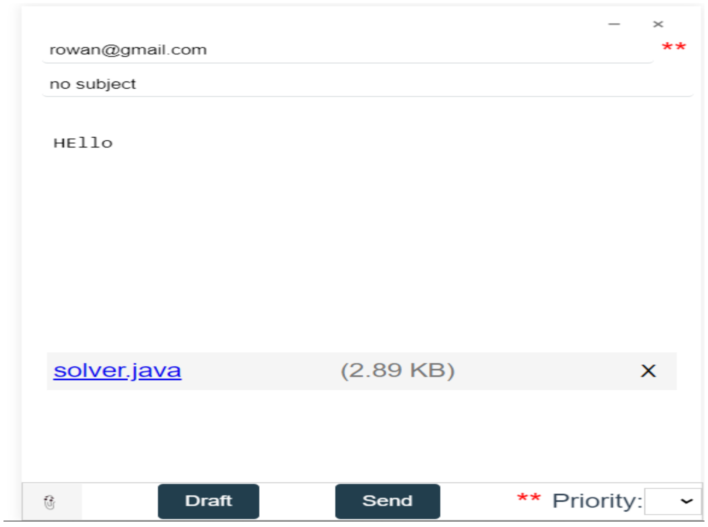
Home



Compose



Compose error

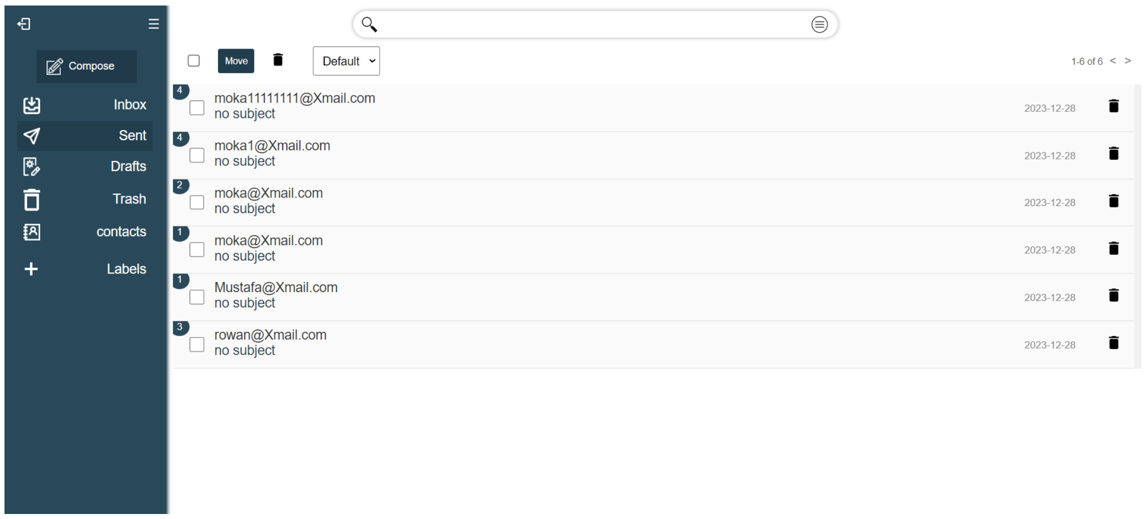

rowan@gmail.com **

no subject

HEllo

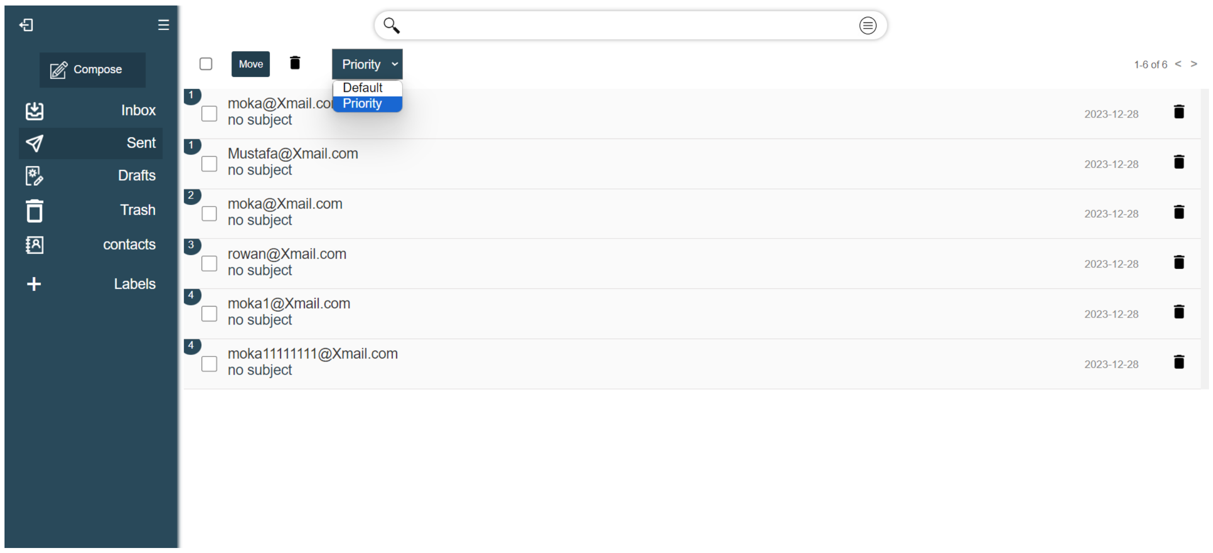
solver.java (2.89 KB)

Default sorting



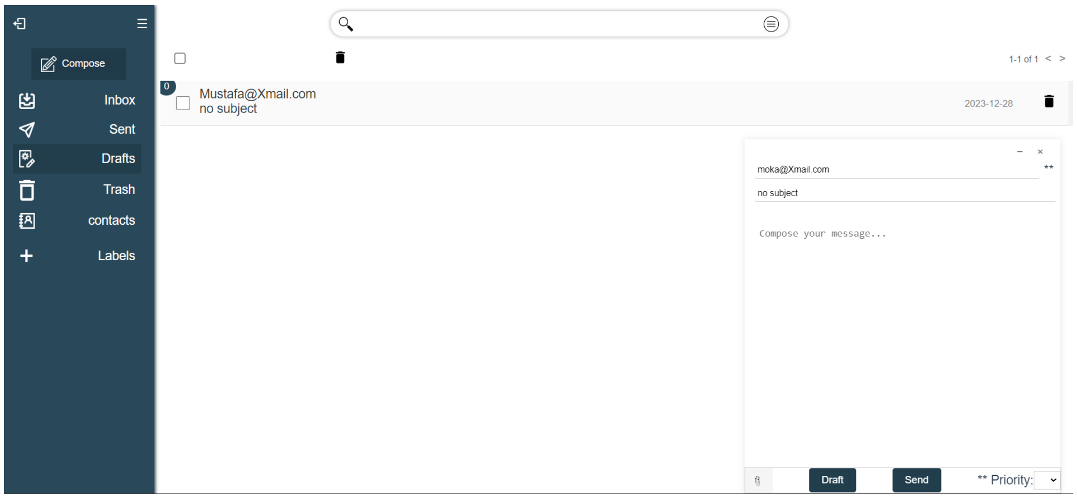
4	moka11111111@Xmail.com no subject	2023-12-28	
4	moka1@Xmail.com no subject	2023-12-28	
2	moka@Xmail.com no subject	2023-12-28	
1	moka@Xmail.com no subject	2023-12-28	
1	Mustafa@Xmail.com no subject	2023-12-28	
3	rowan@Xmail.com no subject	2023-12-28	

Priority Sort



1	moka@Xmail.co no subject	2023-12-28	
1	Mustafa@Xmail.com no subject	2023-12-28	
2	moka@Xmail.com no subject	2023-12-28	
3	rowan@Xmail.com no subject	2023-12-28	
4	moka1@Xmail.com no subject	2023-12-28	
4	moka11111111@Xmail.com no subject	2023-12-28	

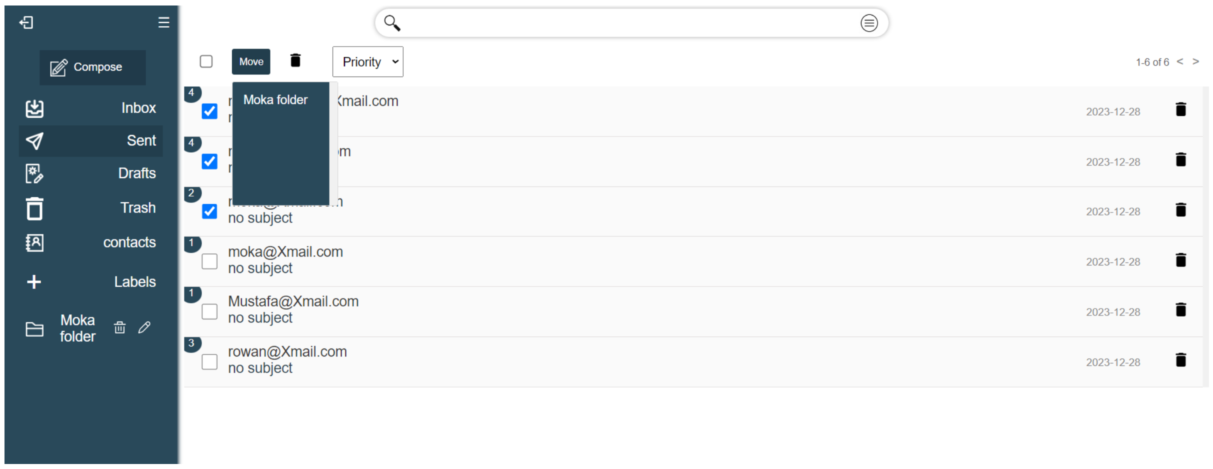
Draft folder



Mustafa@Xmail.com
no subject
2023-12-28

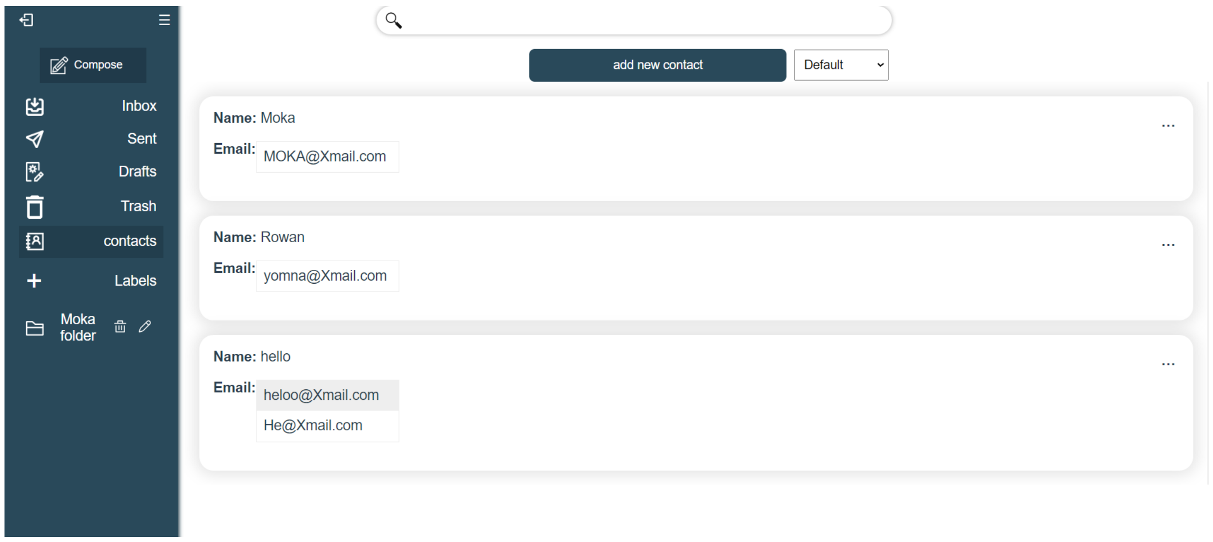
moka@Xmail.com
no subject
Compose your message...

Bulk Move



4	...mail.com	2023-12-28
4	...m	2023-12-28
2	...n no subject	2023-12-28
1	moka@Xmail.com no subject	2023-12-28
1	Mustafa@Xmail.com no subject	2023-12-28
3	rowan@Xmail.com no subject	2023-12-28

Contact

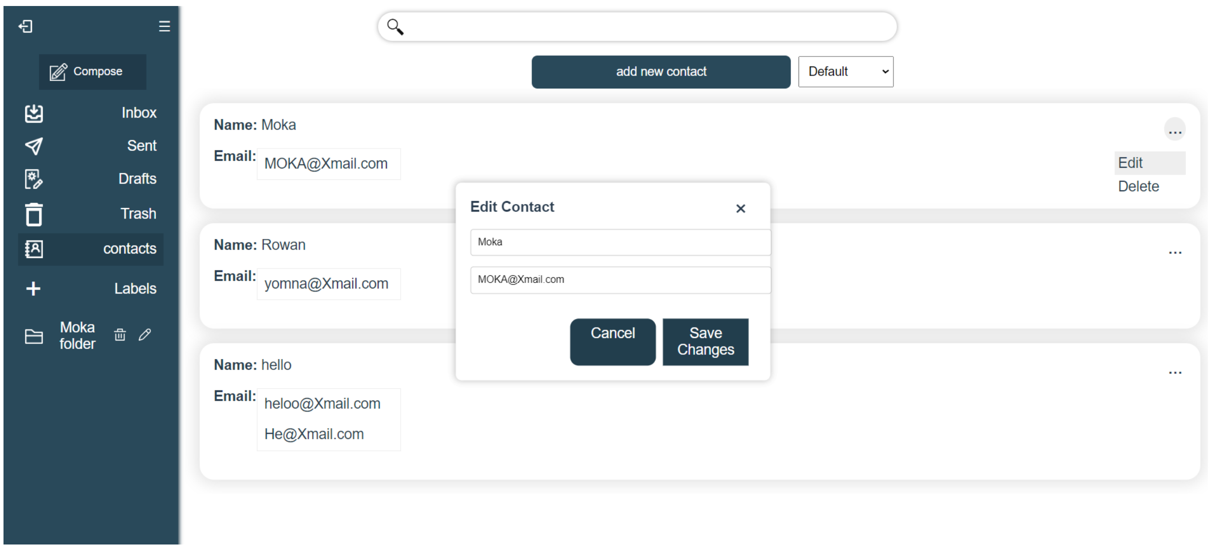


Name: Moka
Email: MOKA@Xmail.com

Name: Rowan
Email: yomna@Xmail.com

Name: hello
Email: heloo@Xmail.com
He@Xmail.com

Contact edit



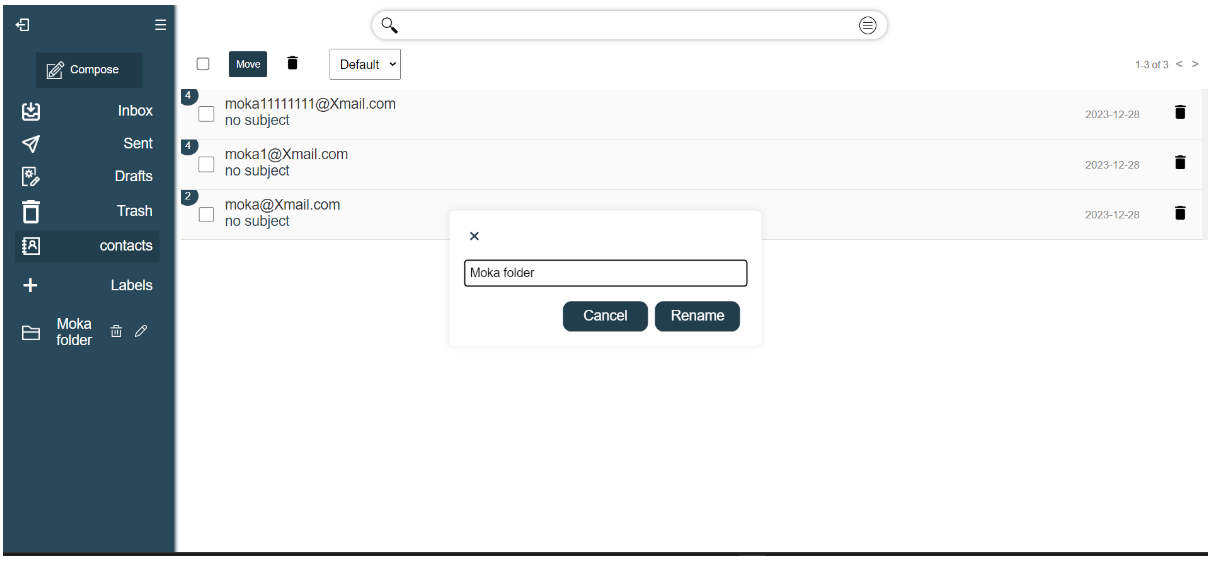
Name: Moka
Email: MOKA@Xmail.com

Name: Rowan
Email: yomna@Xmail.com

Name: hello
Email: heloo@Xmail.com
He@Xmail.com

Edit Contact
Moka
MOKA@Xmail.com

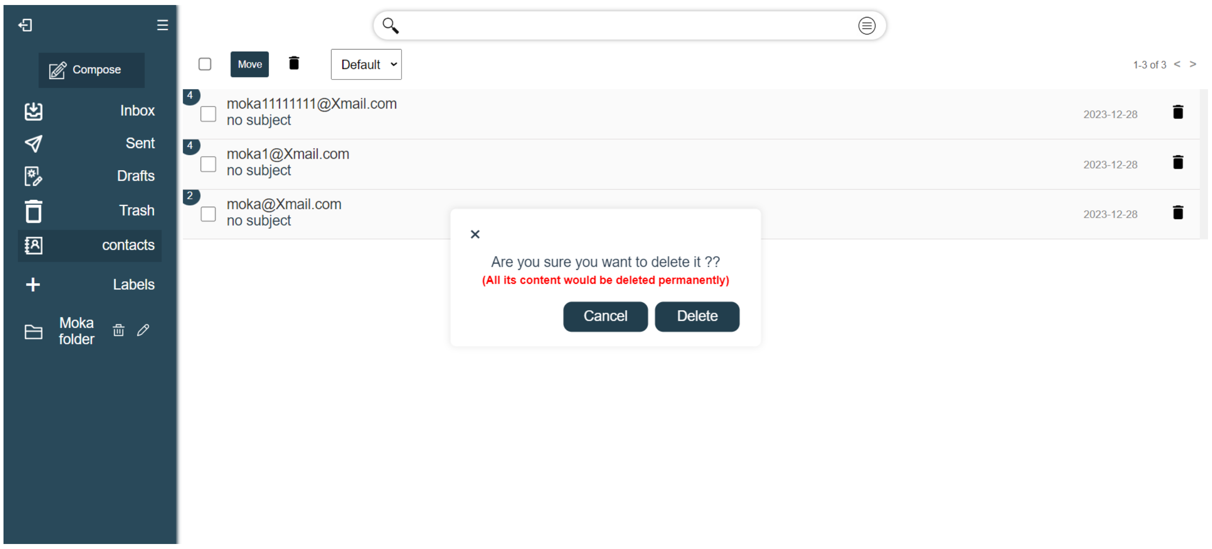
Rename folder



4 moka11111111@Xmail.com no subject 2023-12-28
4 moka1@Xmail.com no subject 2023-12-28
2 moka@Xmail.com no subject 2023-12-28

Moka folder

Delete Folder



moka11111111@Xmail.com
no subject 2023-12-28

moka1@Xmail.com
no subject 2023-12-28

moka@Xmail.com
no subject 2023-12-28

Are you sure you want to delete it ??
(All its content would be deleted permanently)

Cancel Delete

Illustrative Video:

- https://drive.google.com/drive/folders/1QjMPM0RfS2r68glnk2Euxby6p9IMW6Ah

Extra Notes/Tips For Dealing With JSON Files and Serialization Errors:

- Use ObjectMapper as it was the easiest to use for dealing with JSON in Java.
- Implement a Marker Interface: serializable (Important in the class that you save its object in a JSON file)
- Make sure your class and its attribute each have a constructor
- If a class doesn't you have to add an empty constructor (Otherwise retrieval/ mapping from a file to an object leads to errors)
- Make sure any object indicated in your code is initialized somewhere, (ex: an ArrayList/an Object in an attribute/etc.)
- Finally: JSON file errors can be confusing and frustrating, I would advise:
    - Open the file itself, check if it's valid JSON in any validator
    - Inspect the file, if there's an object that's put as NULL whereas you meant to use it later → That's probably what gets the serialization error
    - If there's sth/list that's empty → should be saved as [ ]
    - Calm down, Leave your laptop, Go for a walk, Pray to god and it'll be fine