Alexandria University
Faculty of Engineering
Computer and Systems Engineering

Lab 4

A simple web-based email program

Name	***ID***
Mariam Mohamed Ahmed	***19017274***
Yara Hossam Abdulaziz	***19016871***
Nourhan Ahmed Arafa	***19016812***
Nada Mohamed Ibrahim	***19016782***

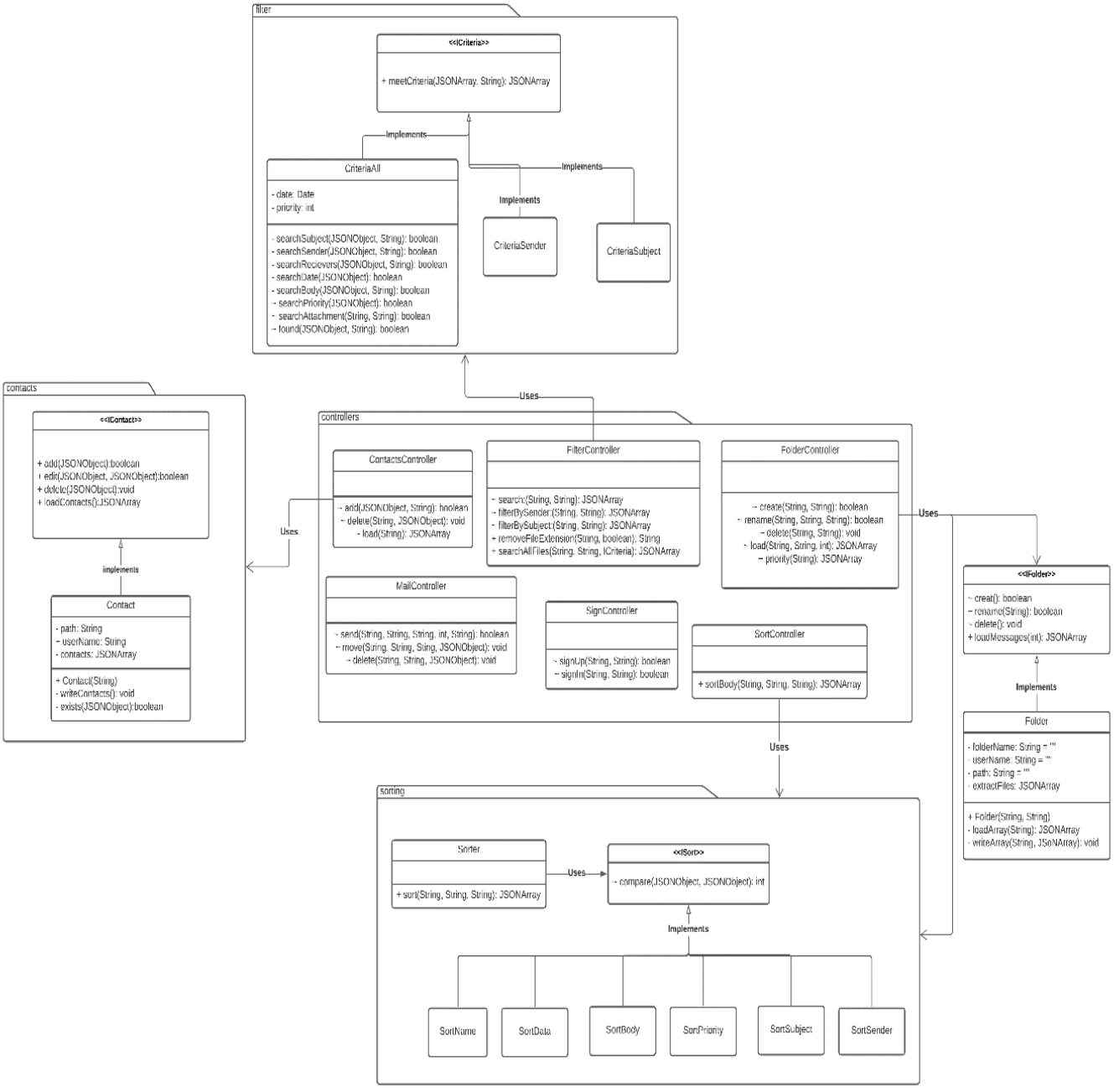
How to run the code

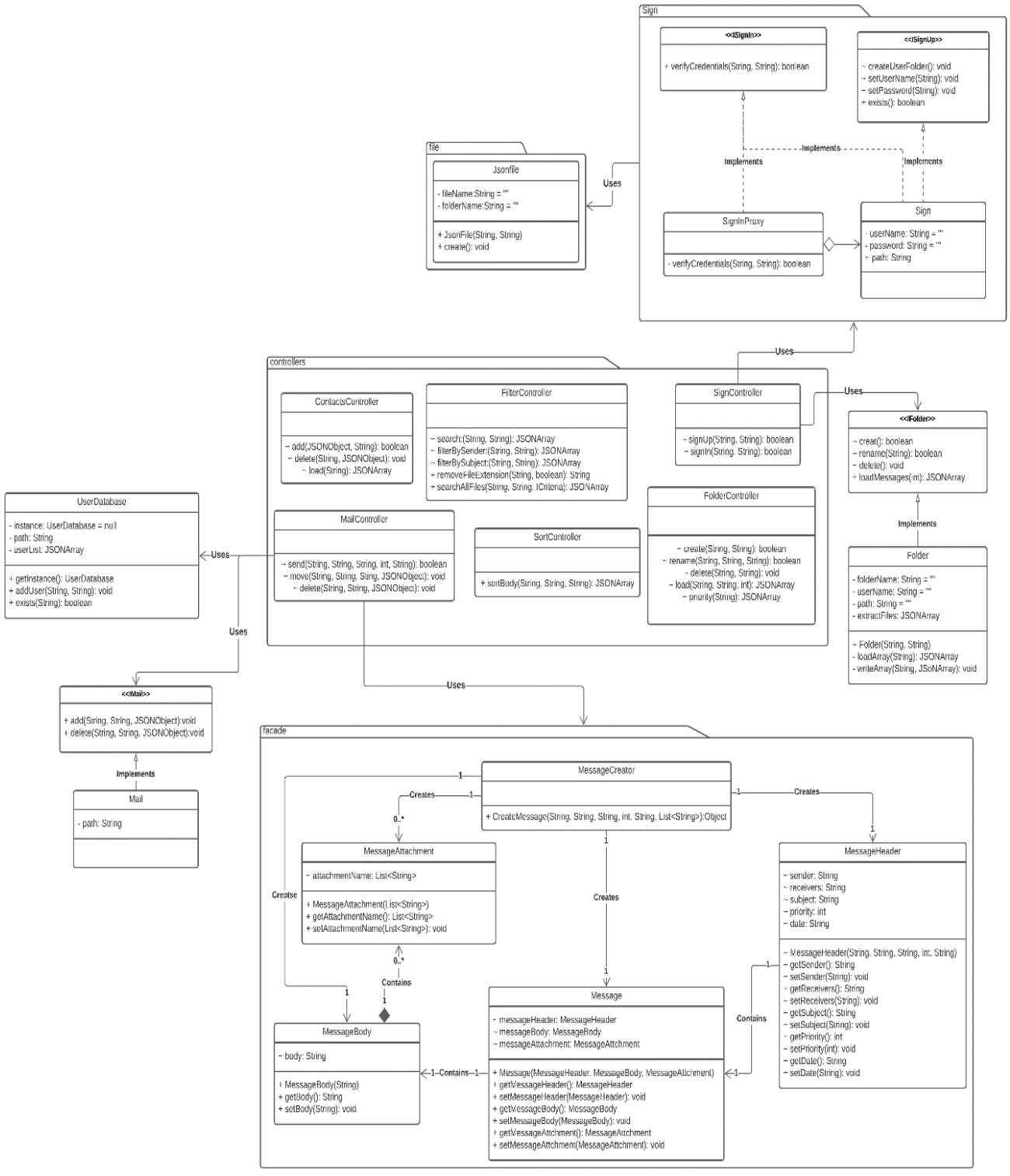
You need to import some packages, components, and decorators to run the program in a full phase.

- Front-End (run on localhost 4200)
  - HttpClient: to allow sending requests to the server.
  - Injectable: to mark a class as available to be provided and injected as a dependency.
  - FormModule: to allow extracting inputs from forms.
  - Routing: interpret a browser URL as an instruction to navigate to a client-generated view.
  - Partitioning: provide navigation between paged information.
- Back-End (run on localhost 8080)
  - org.springframework.boot.SpringApplication
  - org.springframework.boot.autoconfigure.SpringBootApplication: to allow running spring application.
  - org.json.simple.JSONArray: to allow dealing with arrays in form of json arrays.
  - org.json.simple.JSONObject: to allow dealing with the objects in form of json objects.
  - org.json.simple.parser.JSONParser: for enabling parsing values.
  - org.json.simple.parser.ParseException: for using methods to handle invalis JSON and parsing errors.
  - java.io.FileReader, java.io.FileWriter: to read and write a stream of characters from and to files.
  - import java.io.IOException: for handling any exceptions.

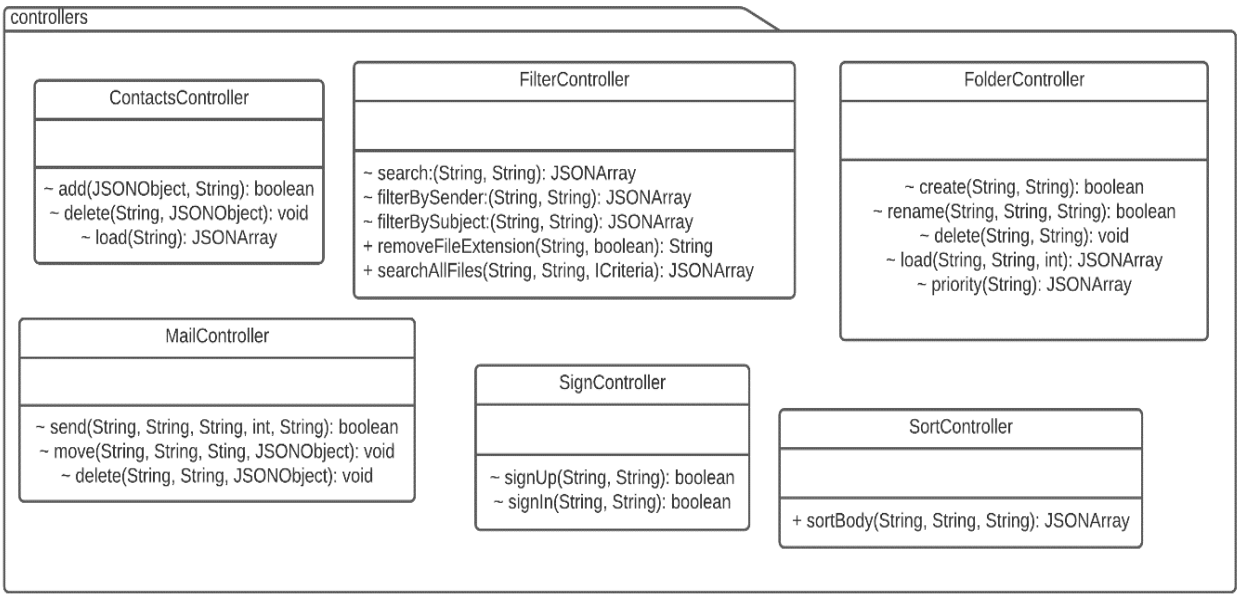
UML Diagrams

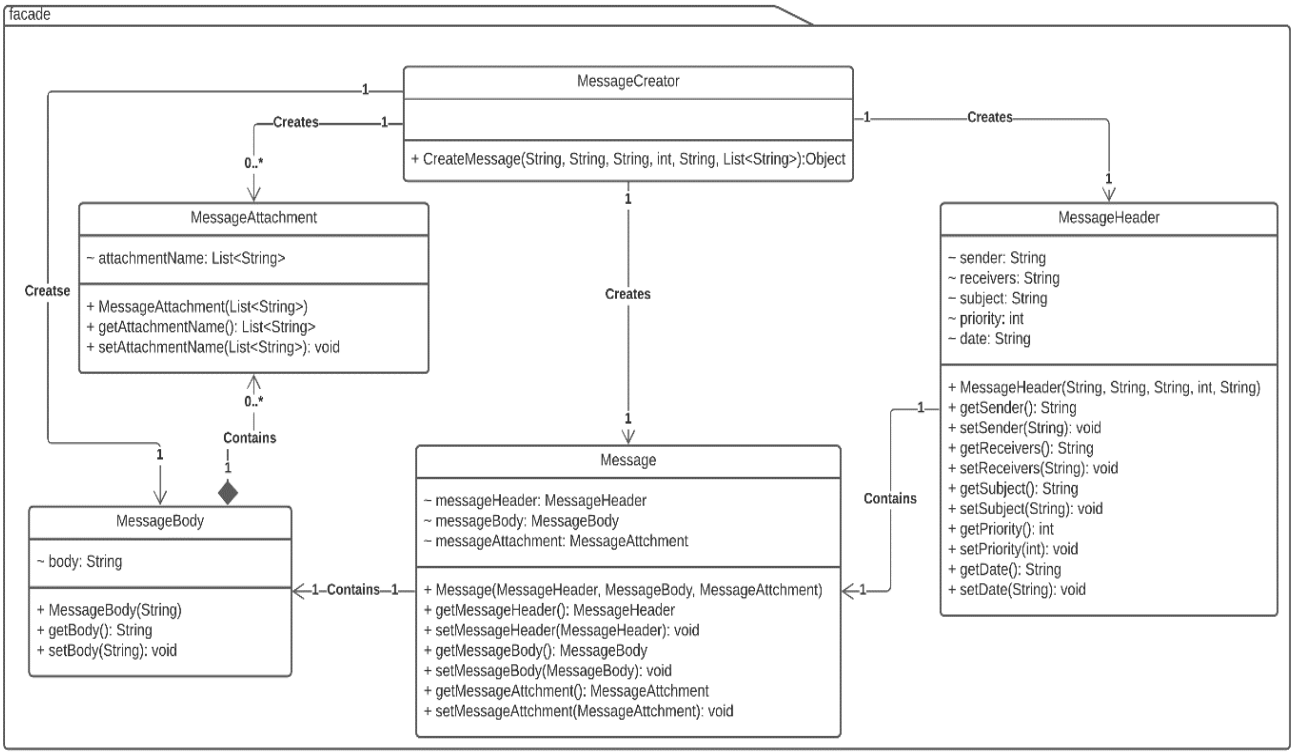
- Back-end class diagram:
  Full Diagram:



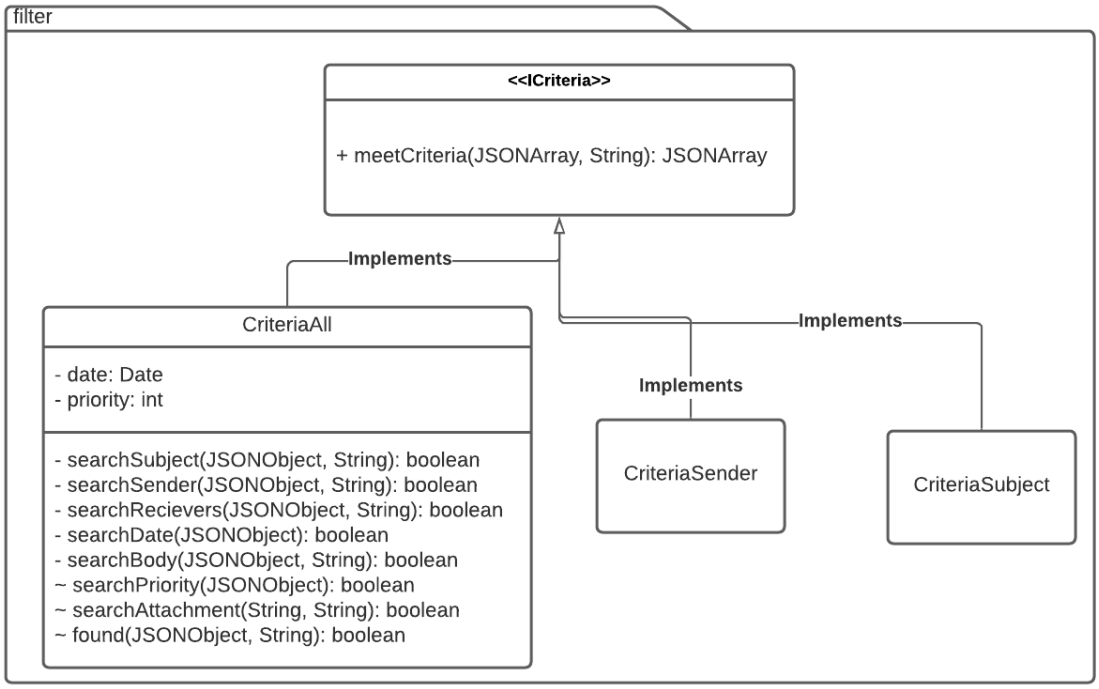

Sign
<<ISignIn>>
+ verifyCredentials(String, String): boolean
<<ISignUp>>
~ createUserFolder(): void
~ setUserName(String): void
~ setPassword(String): void
+ exists(): boolean
Implements
Implements
Implements
file
Jsonfile
- fileName:String = ""
- folderName:String = ""
+ JsonFile(String, String)
+ create(): void
Uses
SignInProxy
- verifyCredentials(String, String): boolean
Sign
- userName: String = ""
- password: String = ""
~ path: String
Uses
controllers
ContactsController
– add(JSONObject, String): boolean
– delete(String, JSONObject): void
~ load(String): JSONArray
FilterController
– search:(String, String): JSONArray
~ filterBySender:(String, String): JSONArray
~ filterBySubject:(String, String): JSONArray
+ removeFileExtension(String, boolean): String
+ searchAllFiles(String, String, ICriteria): JSONArray
SignController
~ signUp(String, String): boolean
– signIn(String, String): boolean
Uses
<<IFolder>>
~ creat(): boolean
– rename(String): boolean
– delete(): void
+ loadMessages(int): JSONArray
Implements
Folder
- folderName: String = ""
- userName: String = ""
- path: String = ""
- extractFiles: JSONArray
+ Folder(String, String)
- loadArray(String): JSONArray
- writeArray(String, JSoNArray): void
UserDatabase
- instance: UserDatabase = null
- path: String
- userList: JSONArray
+ getInstance(): UserDatabase
+ addUser(String, String): void
+ exists(String): boolean
Uses
MailController
– send(String, String, String, int, String): boolean
– move(String, String, Sting, JSONObjec): void
~ delete(String, String, JSONObject): void
SortController
+ sortBody(String, String, String): JSONArray
FolderController
– create(String, String): boolean
~ rename(String, String, String): boolean
~ delete(String, String): void
– load(String, String, int): JSONArray
– priority(String): JSONArray
Uses
Uses
<<IMail>>
+ add(String, String, JSONObject):void
+ delete(String, String, JSONObject):void
Implements
Mail
- path: String
facade
MessageCreator
+ CreateMessage(String, String, String, int, String, List<String>):Object
1
Creates
1
0..*
Creates
1
Creates
MessageAttachment
~ attachmentName: List<String>
+ MessageAttachment(List<String>)
+ getAttachmentName(): List<String>
+ setAttachmentName(List<String>): void
Creates
Creates
MessageHeader
– sender: String
~ receivers: String
~ subject: String
– priority: int
– date: String
– MessageHeader(String, String, String, int, String)
– getSender(): String
– setSender(String): void
– getReceivers(): String
– setReceivers(String): void
– getSubject(): String
– setSubject(String): void
– getPriority(): int
– setPriority(int): void
– getDate(): String
– setDate(String): void
0..*
Contains
1
1
Contains
1
1
MessageBody
~ body: String
+ MessageBody(String)
+ getBody(): String
+ setBody(String): void
1–Contains–1
Message
- messageHeader: MessageHeader
~ messageBody: MessageBody
– messageAttachment: MessageAttchment
+ Message(MessageHeader, MessageBody, MessageAttchment)
+ getMessageHeader(): MessageHeader
+ setMessageHeader(MessageHeader): void
+ getMessageBody(): MessageBody
+ setMessageBody(MessageBody): void
+ getMessageAttchment(): MessageAttchment
+ setMessageAttchment(MessageAttchment): void
Contains
1
1

Separate Packages:

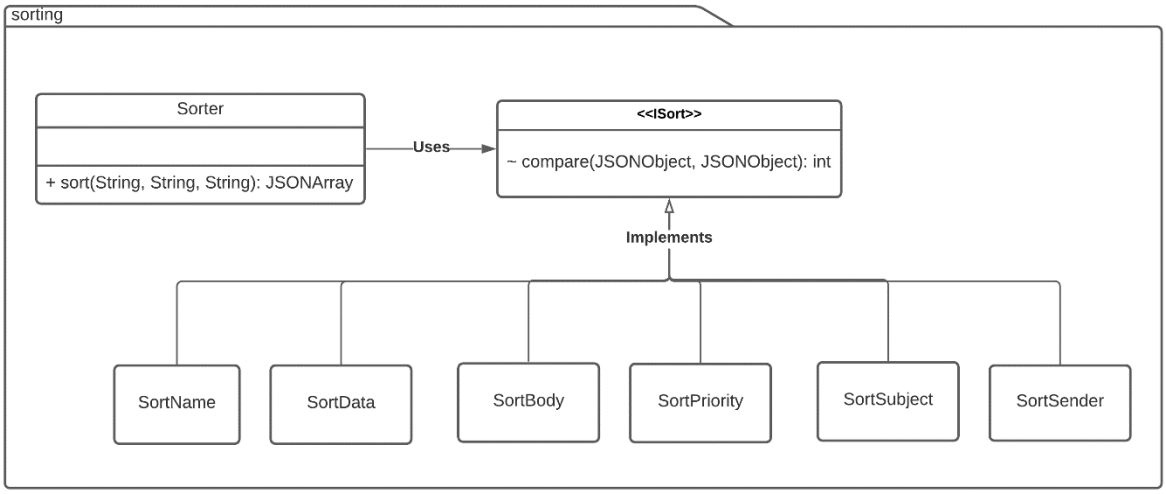




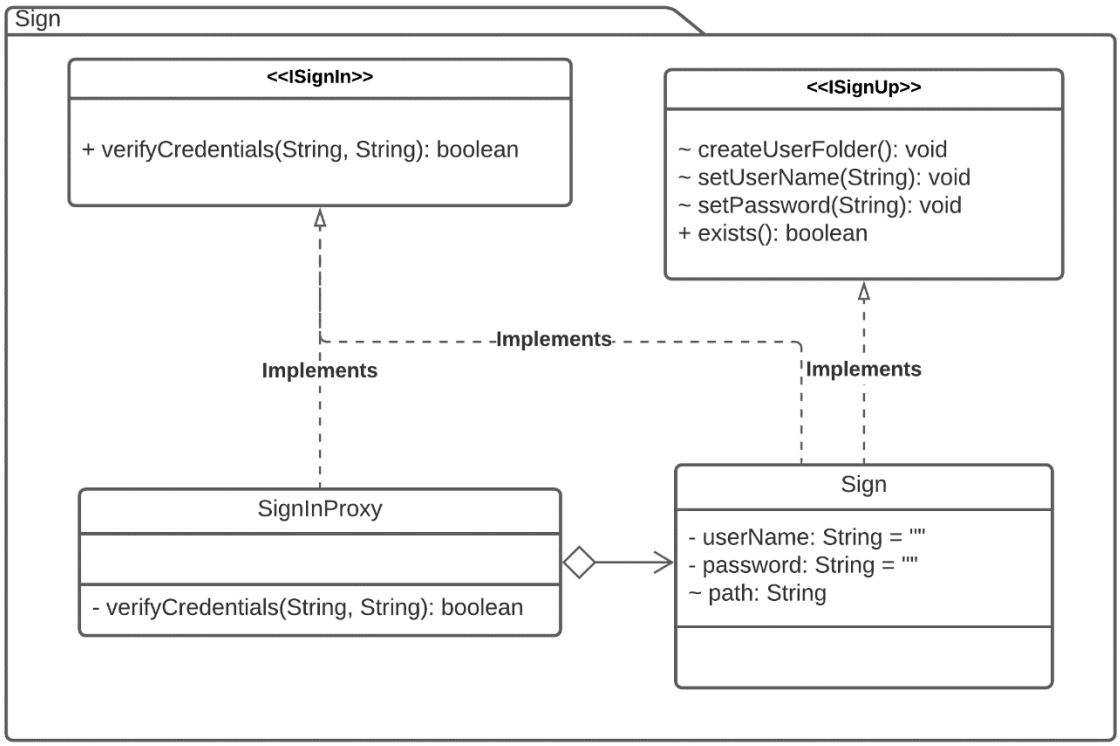
Filter DP:



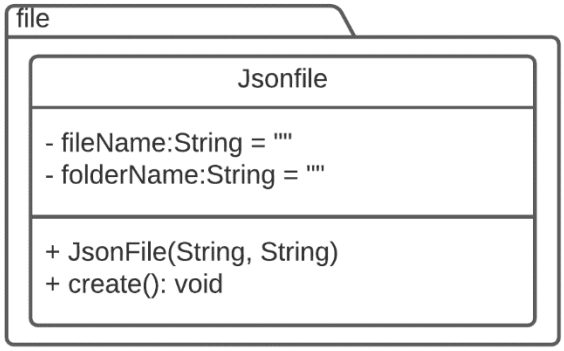
Sorting:



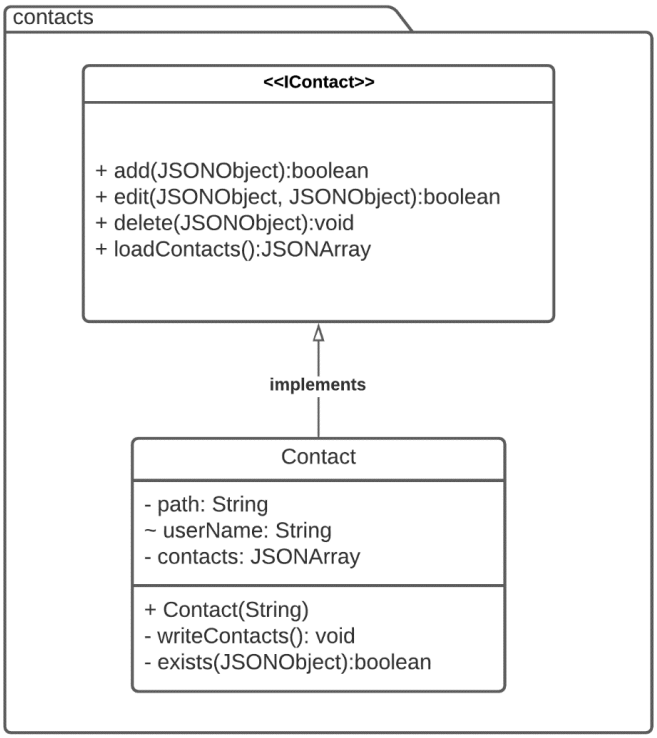
Proxy DP:



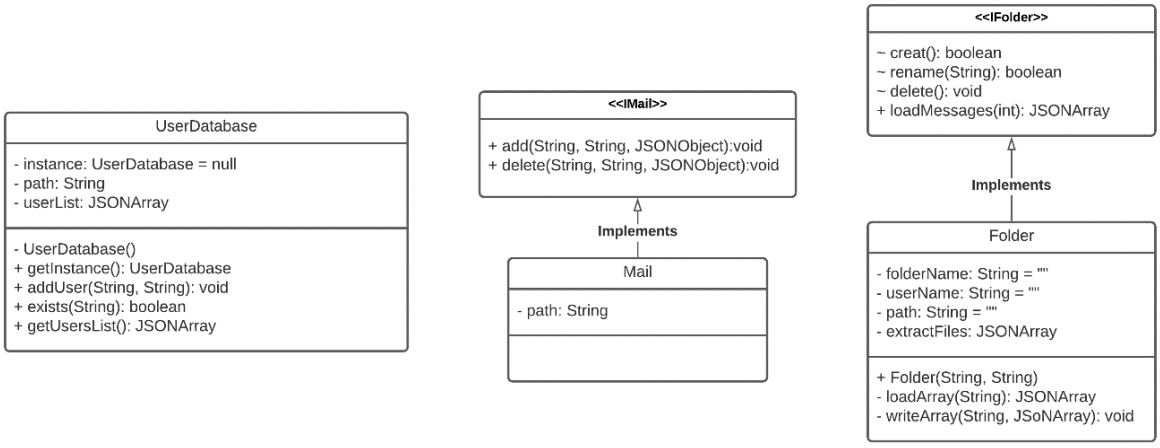
JSON Configuration:



Contacts:



User Database:



Design Patterns

- For the Back-End, We used five design patterns:
  - Façade design pattern:
    Used to create and send email messages.
  - Filter design pattern:
    Used to filter emails according to different criteria
  - Proxy design pattern:
    Used to perform verification before the request gets through the original object.
  - Singleton design pattern:

    Single database object shared by different parts of the program.

Fundamental DP:

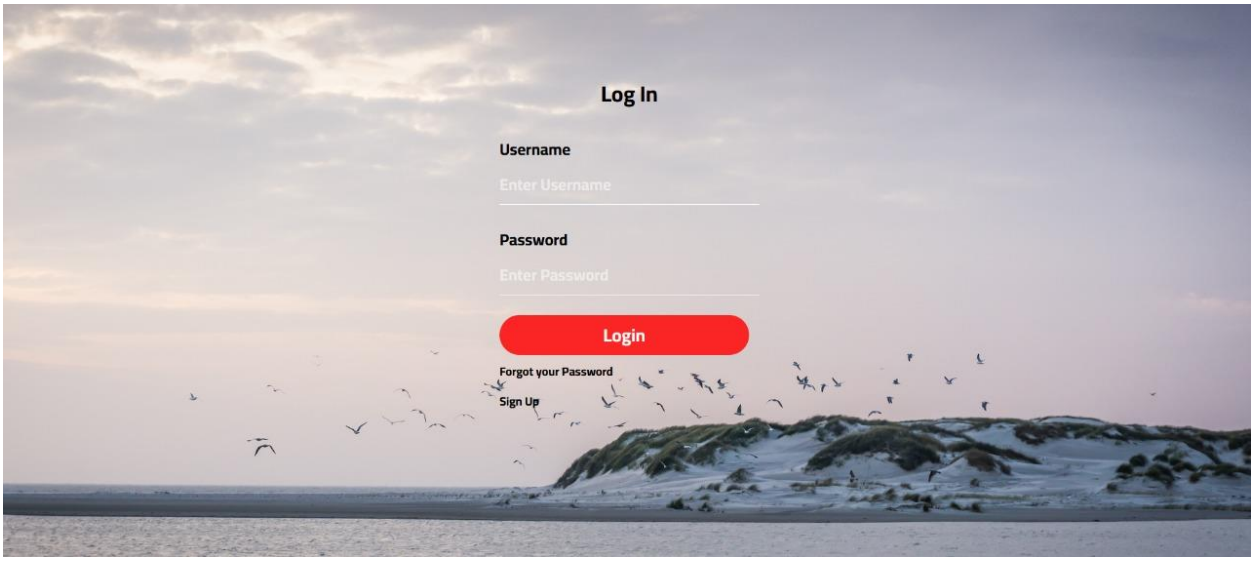
- Interface: Keep classes that uses data and services provided by instances of other classes independent of those classes by having those instances through an interface.
- MVC

Design Decisions

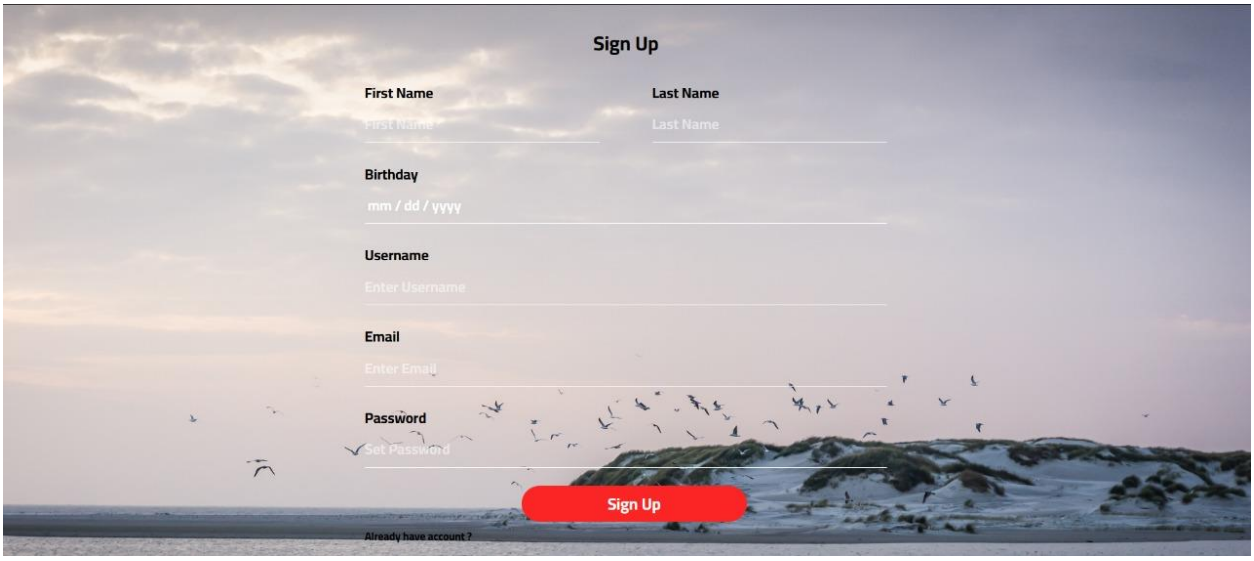
1. Filtering directs the results to a new user folder.
2. It’s prohibited to have 2 contacts or folders with the same name.
3. Emails are retrieved using pagination.
4. Any file type of attachment is supported.
5. Trash messages are auto deleted after 30 days.

Snapshots of the UI

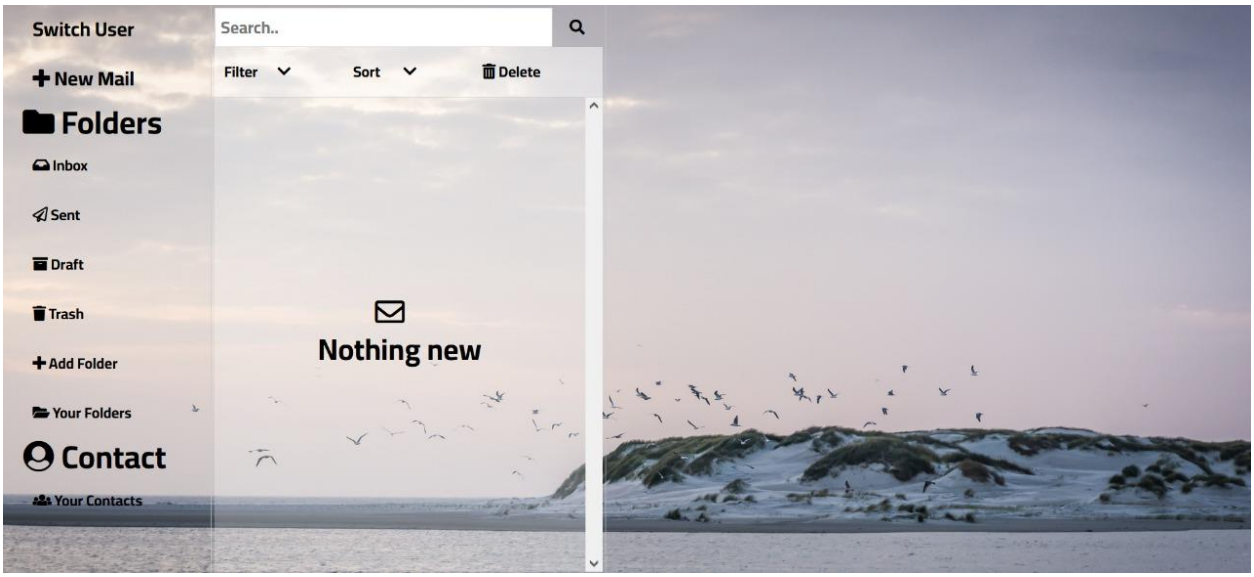
- The UI
  LogIn page



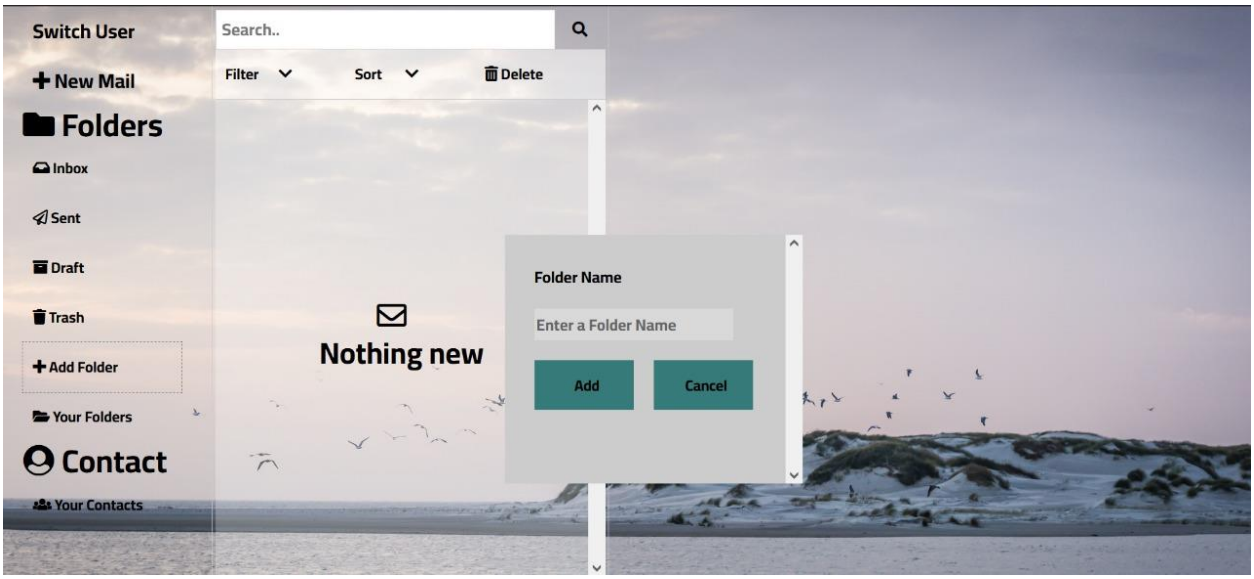
- Sign Up page



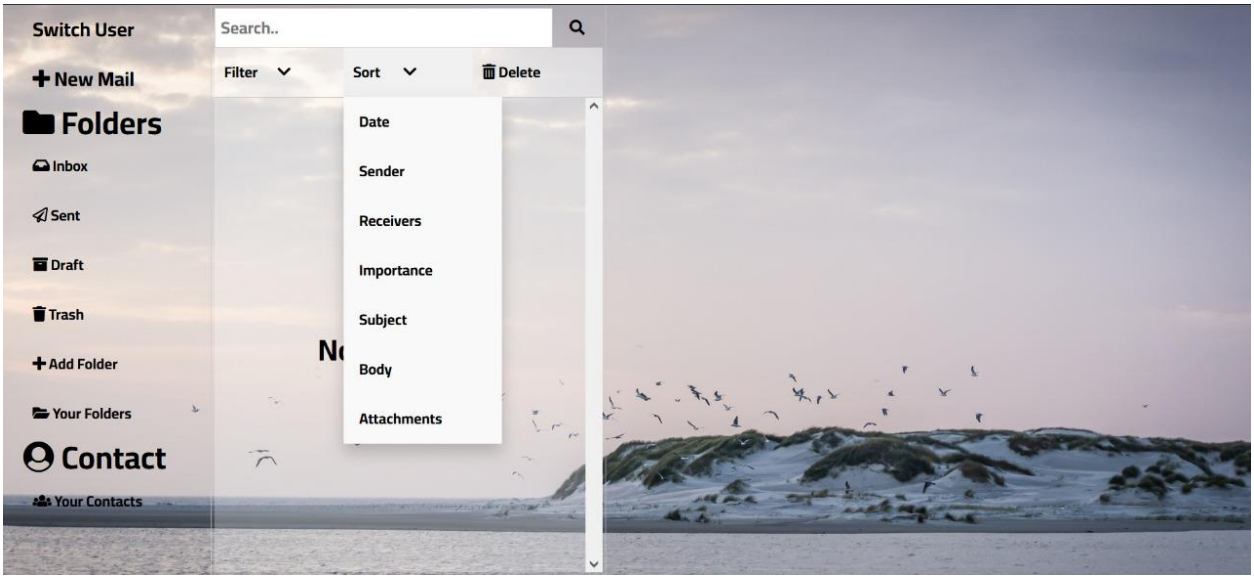
Main Page:



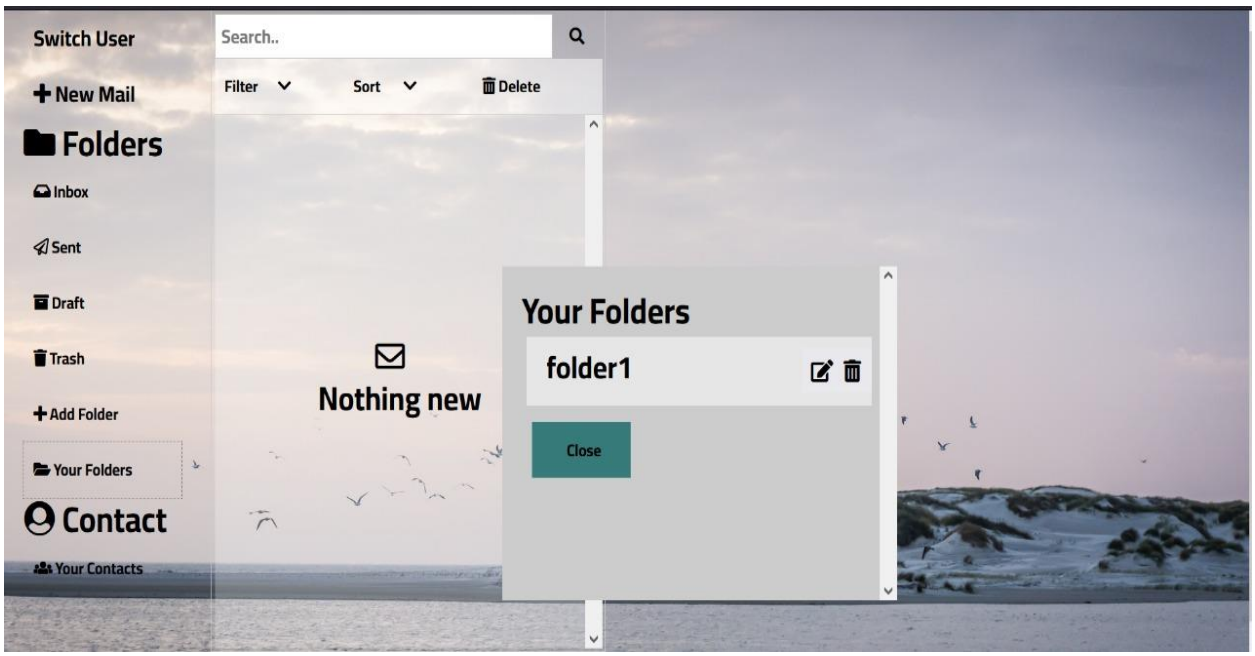
- Add folder



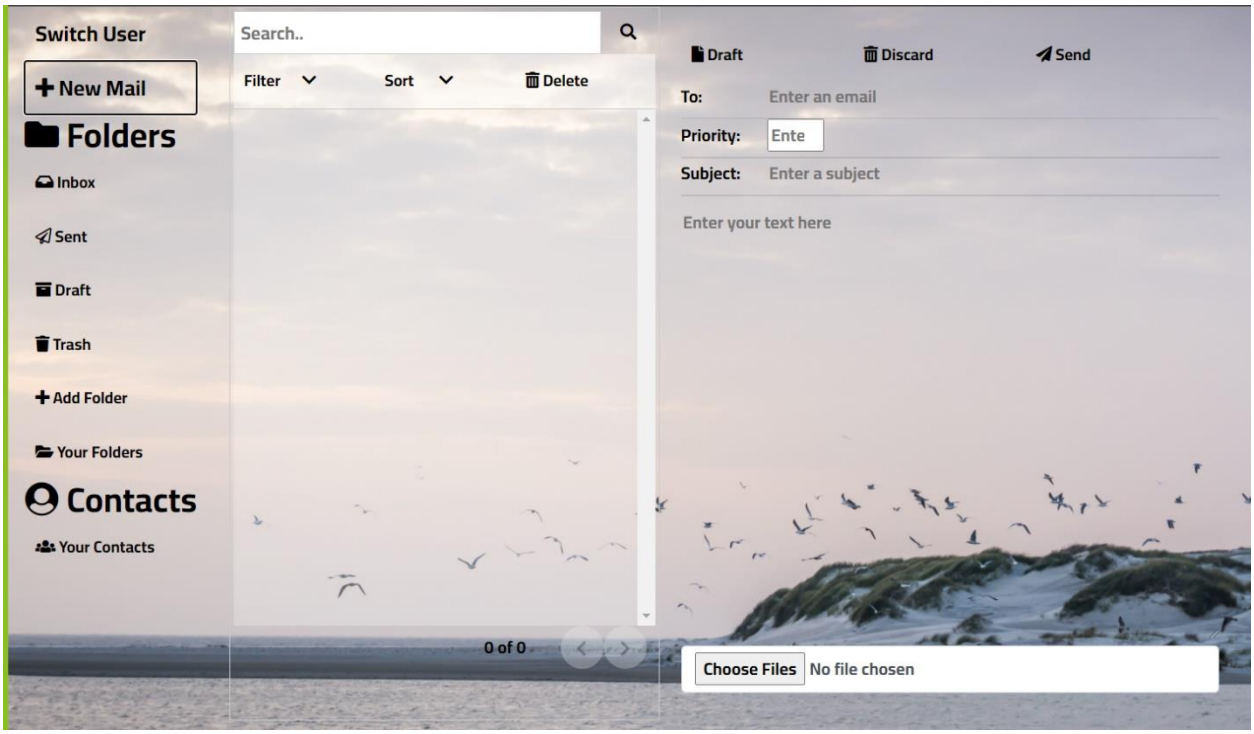
Sort:



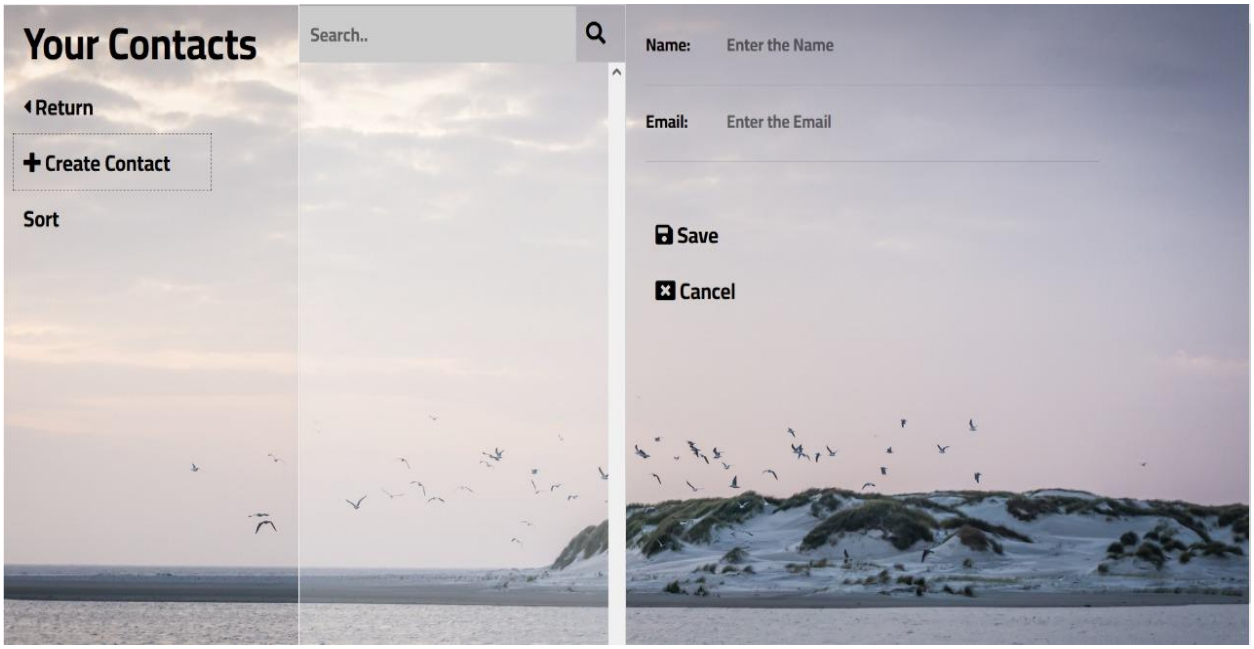
User Folders:



Compose



Contact:



User Guide:

1. Signup / login if the user exists.
2. The user will be directed to the main page.
3. Default (Inbox) and user folders will appear on the left panel.
4. You can compose a new message/draft from the New Mail button.
5. You can add a new folder by pressing Add folder button, then entering the folder name.
6. Filtering is according to sender/subject. After choosing a criteria, a dialog box will appear to enter the sender/subject criteria and folder name where the filter results will be directed to.
7. To view your contacts, press the Your Contacts button which will direct you to the contacts page where you can add, view, edit, delete any contact.
8. To bulk delete, you can either select all the elements to be deleted and press the delete button