



- 1.Ziad Hassan Mahmoud Mohamed 18010720
- 2.Abdelrahman Mohamed Ebrahim Mohamed Sakr 18010965
- 3.seif el din wael ali 18010832
- 4.mohamed mostafa badran 18011621

Report contents:

front end dependencies:page 2
user manual: page 4
front end components: page 12
project views: page 13
class uml page 15
design pattern page 16
save and load page 18

front end:

front end dependencies:

```
"name": "front",
"version": "0.1.0",
"private": true,
"scripts": {
  "serve": "vue-cli-service serve",
  "build": "vue-cli-service build",
  "lint": "vue-cli-service lint"
},
"dependencies": {
  "axios": "^0.21.1",
  "core-js": "^3.6.5",
  "sass-loader": "^10.1.0",
  "vue": "^3.0.0",
  "vue-router": "^4.0.0-0"
},
"devDependencies": {
  "@vue/cli-plugin-babel": "~4.5.0",
  "@vue/cli-plugin-eslint": "~4.5.0",
  "@vue/cli-plugin-router": "~4.5.0",
  "@vue/cli-service": "~4.5.0",
  "@vue/compiler-sfc": "^3.0.0",
  "@vue/eslint-config-prettier": "^6.0.0",
  "babel-eslint": "^10.1.0",
  "eslint": "^6.7.2",
  "eslint-plugin-prettier": "^3.1.3",
  "eslint-plugin-vue": "^7.0.0-0",
  "prettier": "^1.19.1"
},
```

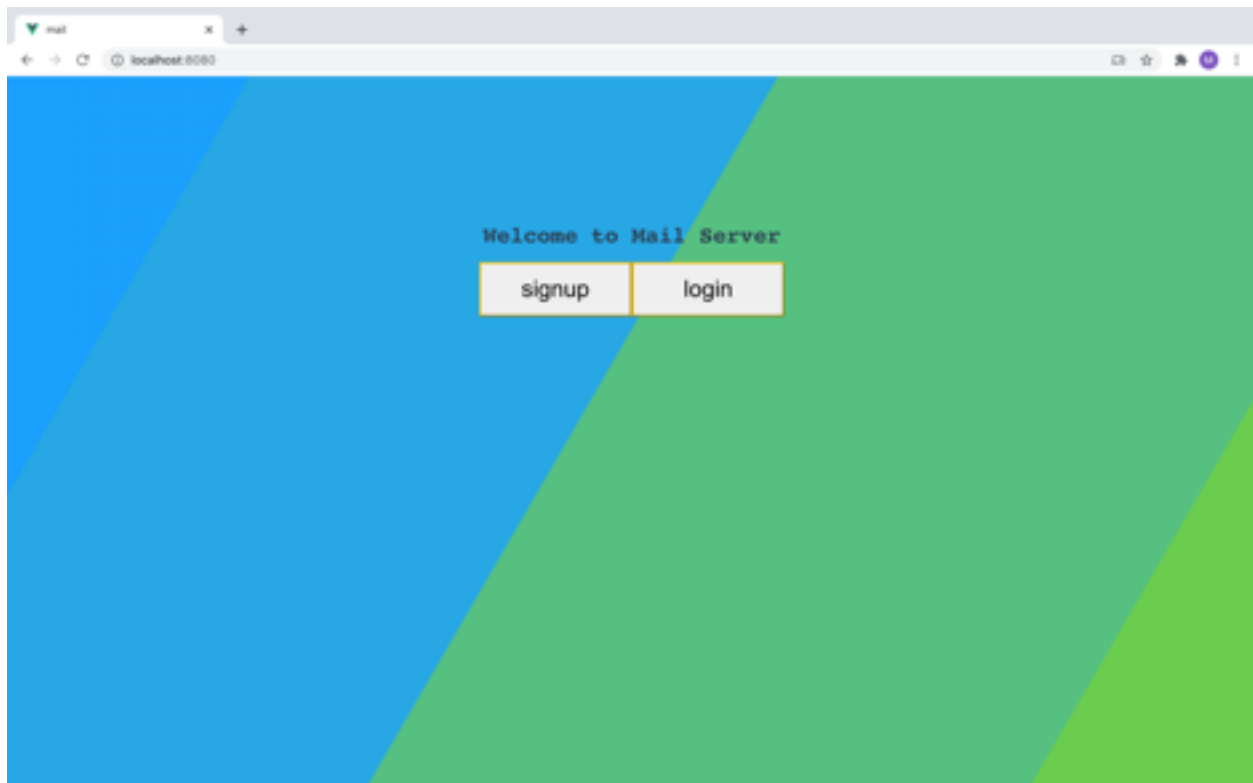
```
"eslintConfig": {
  "root": true,
  "env": {
    "node": true
  },
  "extends": [
    "plugin:vue/vue3-essential",
    "eslint:recommended",
    "@vue/prettier"
  ],
  "parserOptions": {
    "parser": "babel-eslint"
  },
  "rules": {}
},
"browserslist": [
  "> 1%",
  "last 2 versions",
  "not dead"
]
```

user manual:

to run front end make just make vue program with the required dependencies and add the sent src folder in the project instead of the made by the compiler then run npm serve and your <http://localhost:8080/> should have the project at it

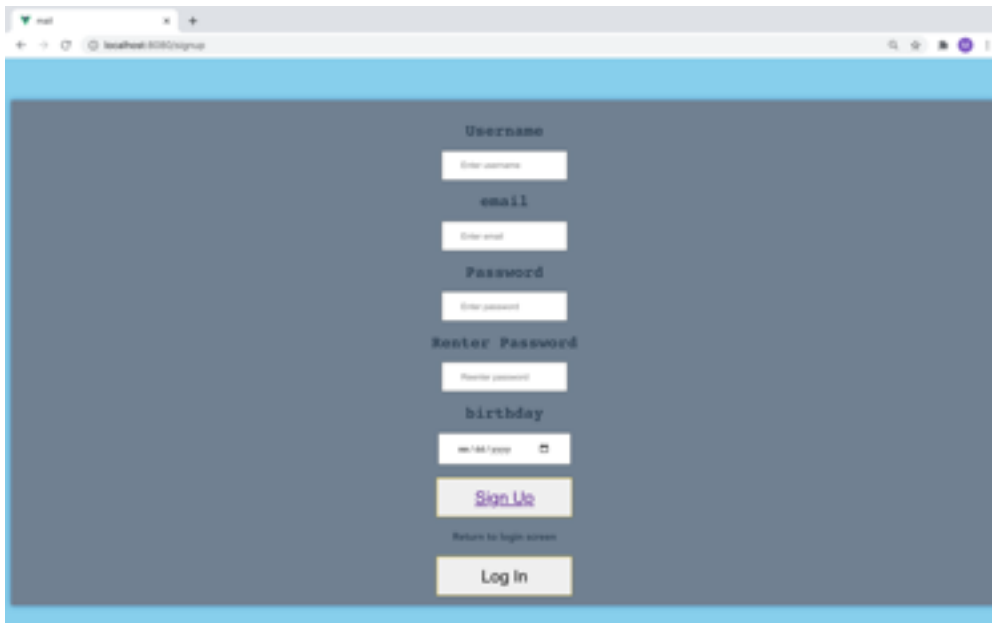
and make sure to run the sent backend too it own localhost and the program should run fine, open <http://localhost:8080/> and start using the mail server.

open <http://localhost:8080/>



at <http://localhost:8080/> the welcome screen should open and from there choose if you like to login with existing user or go to sign up to make a new user

at signup screen:



Username
Enter username

email
Enter email

Password
Enter password

Reenter Password
Reenter password

birthday
mm/dd/yyyy

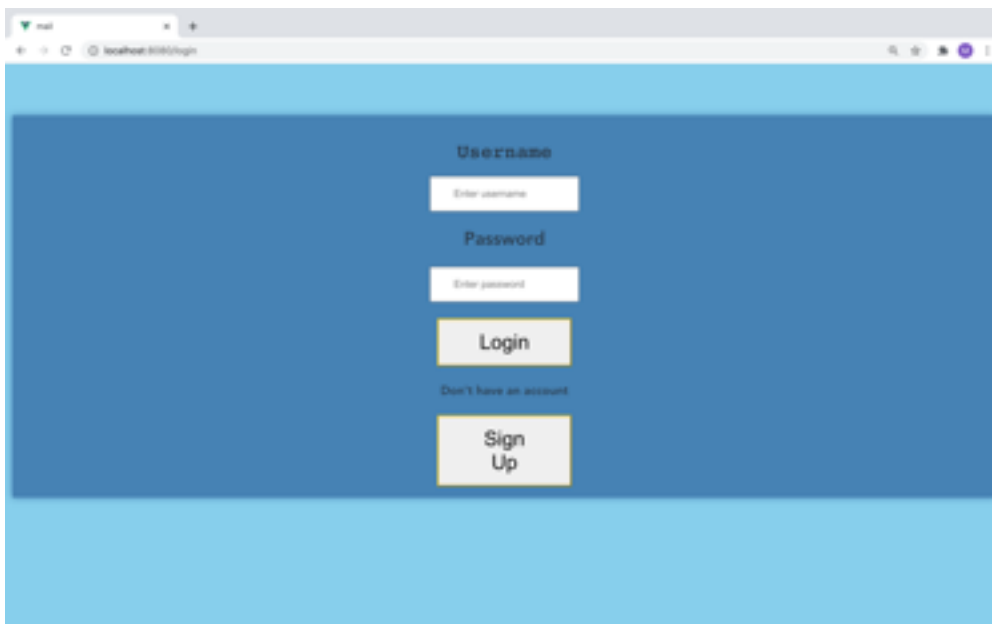
[Sign Up](#)

[Return to login screen](#)

[Log In](#)

to signup enter the required data the username and the email to be made and enter the password and reenter your password to make sure you entered it right

at login screen:



Username
Enter username

Password
Enter password

[Login](#)

[Don't have an account](#)

[Sign Up](#)

to login with existing user enter your email and password, then you will move to your mailbox.

mailbox:



at mailbox you can see

your username at the top at it right button to go to your contacts and button to logout and at the bottom of it you see the different mailboxes inbox sent trash and draft by clicking on any of them the backend will upload the mails in this box and show in in the same view without refreshing or going to a new page and under the mailboxes you see compose where it will make you go to /compose where you can compose a new mail and to the left delete button and under them filter and sort tools to help you find the mails you want.if you click in any email you will move to new page where only single mail is showed which you clicked in.

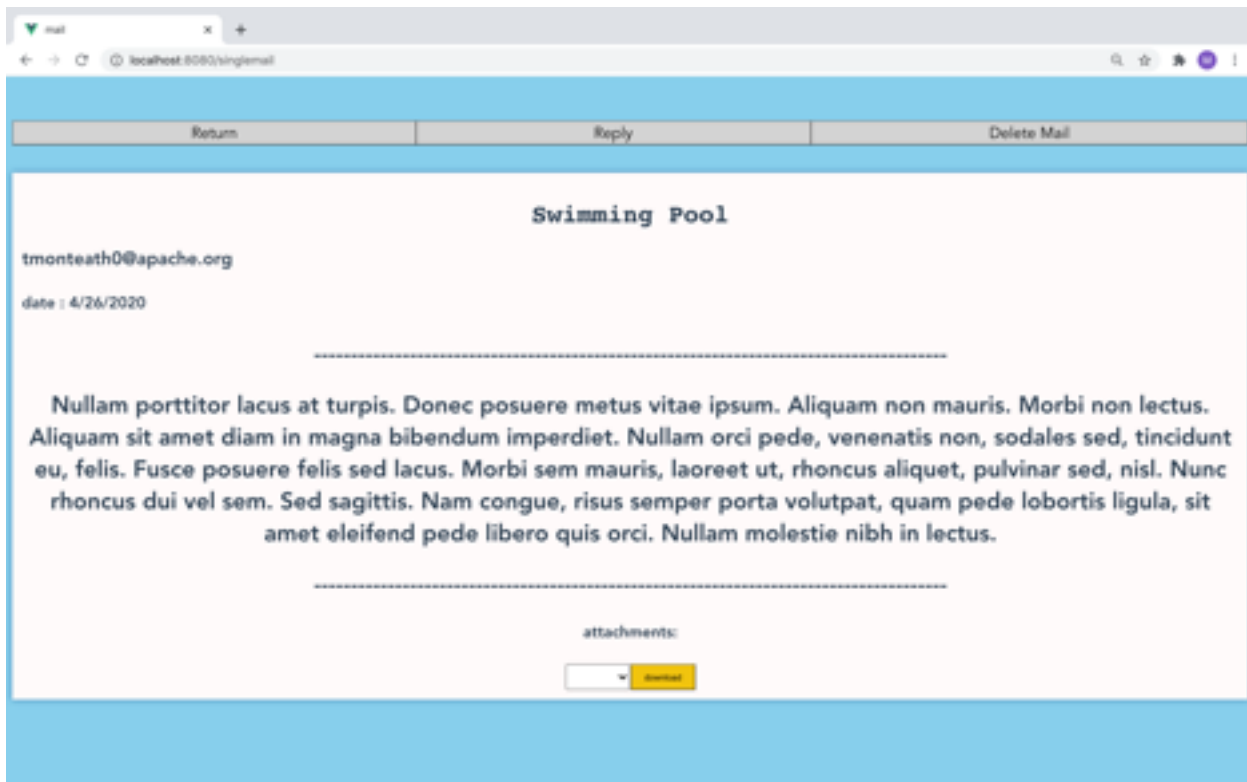
to delete emails select the email you want by checking the checkbox in the bottom of each mail and then click delete button at the right of compose button.

at mailbox you can click compose at the top of the page to create a new mail.
compose:

The screenshot displays a web browser window with the address bar showing 'localhost:8080/compose'. The page features a light blue header bar with three buttons: 'Send', 'Save To Draft', and 'Delete'. Below the header, the email composition interface is set against a light pink background. It includes a 'To:' field with the placeholder 'Enter the email', a 'Title:' field with the placeholder 'Enter title', and a large text area for the email body with the placeholder 'Enter the mail'. At the bottom of the form, there is a green button labeled 'Choose File' with a file icon, and a green button labeled 'Remove Selected attachment'.

you choose if the email is high priority and then you enter email addresses you want the mail to then you write title and the mail and then if you want to add attachments you can use the file reader in the bottom and you can remove added attachment too if you want.
after finishing writing the mail you can send it using the send button at the top or save it in the drafts or you can delete it and return to mailbox and deleting all what you added.

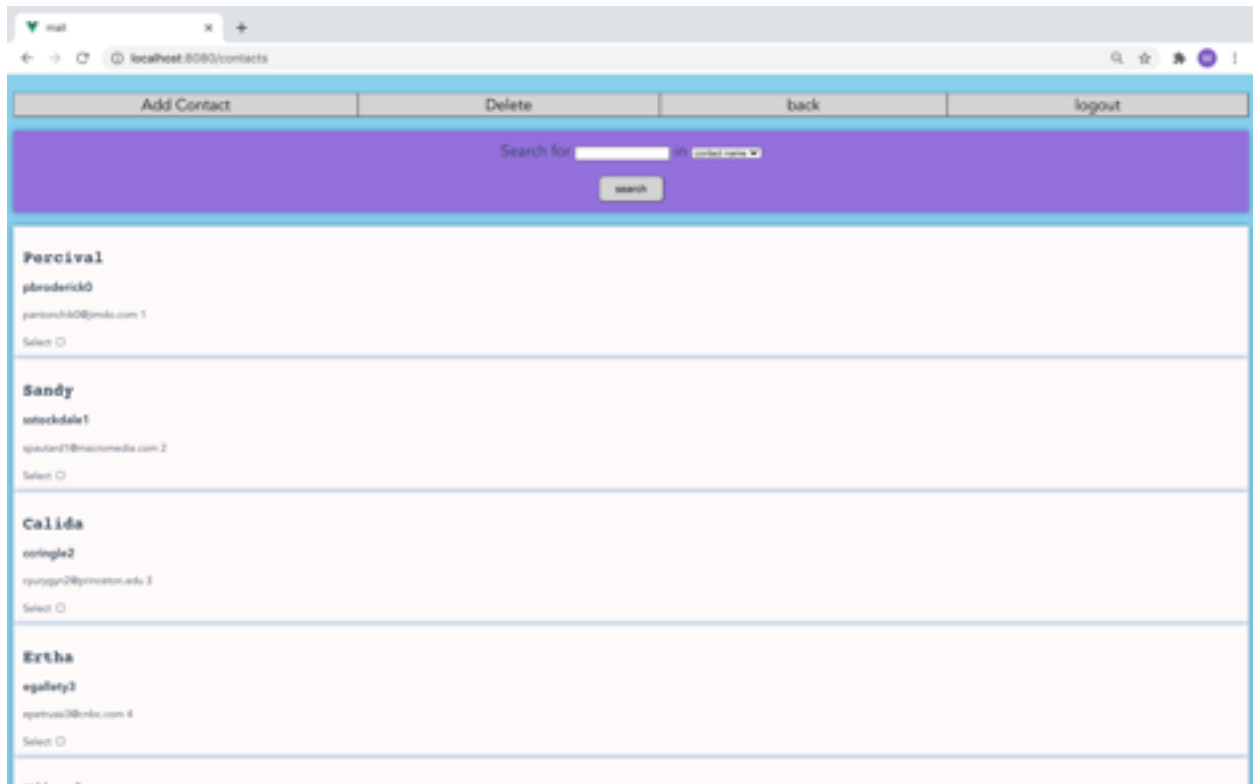
if you clicked at any mail in the mailbox it will open single mail view.
single mail:



when opening a single mail it open in a new page with larger front to read it better also you can download the attachments if there is any in the mail like you can do to in the mailbox.

in the single mail top there is 3 buttons at the top the first is return to return to mailbox and the second is reply where it where take you to compose with email of the user in the mail as the email to send to the new mail and the last button is delete mail which will delete the mail you opened

if you clicked contacts at the top of the mailbox next to your username you will open your contacts.
contacts:



at contacts page you will find at the top 4 buttons and under them filter tool to filter through your contacts and under it your contacts if you clicked on any of them it will open the contact in new page where you can see it data and send mail to it.

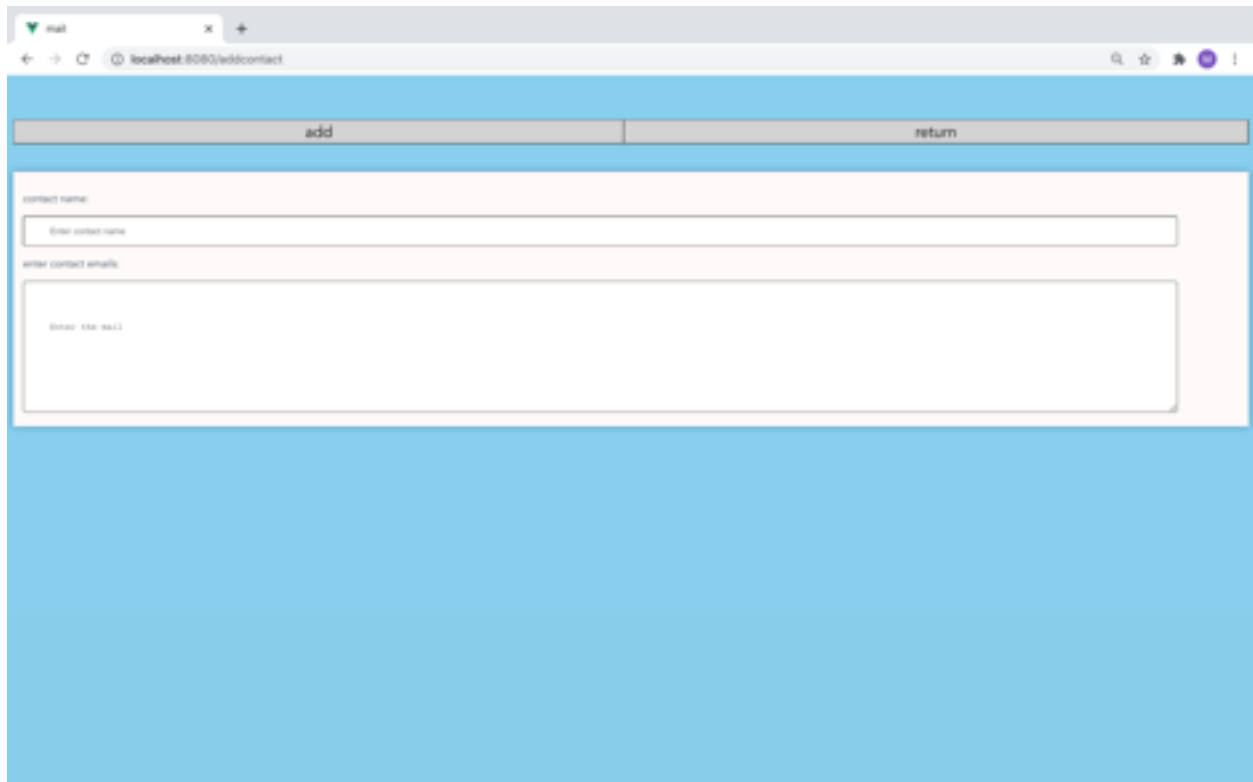
the 4 buttons at the top of the pages are:

- 1.add contact : it will move you to a new a page where you can add a new contact.
- 2.delete: button to delete selected contacts.
- 3.back to return to your mailbox.
- 4.logout: to log out of your email.

to delete contacts select the email you want by checking the checkbox in the bottom of each contact and then click delete button at the right of add contact button.

if you clicked at add contact at the top of contacts page it will take you to /addcontact where you can add a new contact:

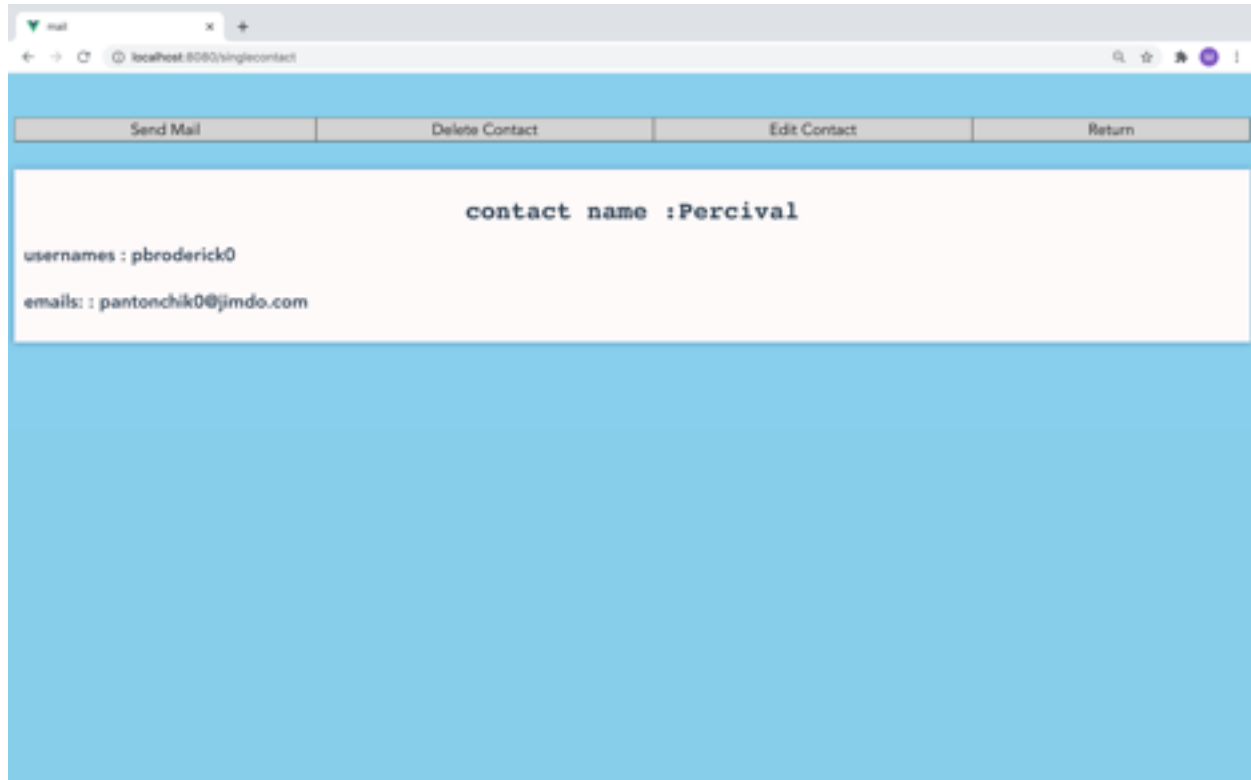
add contact:



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/addcontact'. The page has a light blue header bar containing two buttons: 'add' and 'return'. Below the header is a form with a light pink background. The form contains two text input fields. The first field is labeled 'contact name:' and the second is labeled 'enter contact email:'. The browser's address bar also shows a search icon, a star icon, and a purple icon.

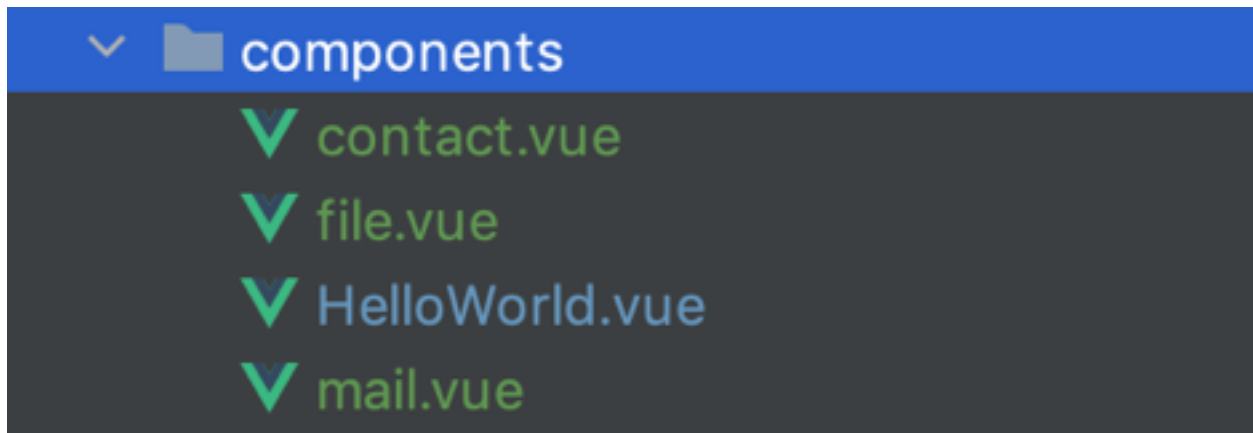
at add contact you add the contact name you want for the emails you will add and then add the emails that will be in the contacts.
at the top there is 2 buttons the first is add to add the contact after filling the data you want in the text boxes and the second is return to return to contacts with adding the new contact so the data you added will be removed without saving it.

if you click at any contact at contacts you will move to a new page where it show you this single contact.
single contact:



at single contact you will see the data of the contact you clicked on with 4 buttons at top the first button is send mail which if you clicked on it will move to compose to write a new mail with the emails in the contact added in the email sent to, the second button is delete is which delete the contact you opened, the third button is edit which will make you edit if the contact data by moving to /addcontact and making you add any changes you like in the contacts and after making them press add to save it or return to undo the changes made and the last button is return which will make you return to contacts.

Components:



we made 4 components for the frontend :

1.contact:

component used to show contacts in contacts list view, it handle the style, script and template for the contacts in the contacts list.

2.file:

component used in make the file reader and selector that help the front end in handling the add attachment and removing attachment when compose a new mail.

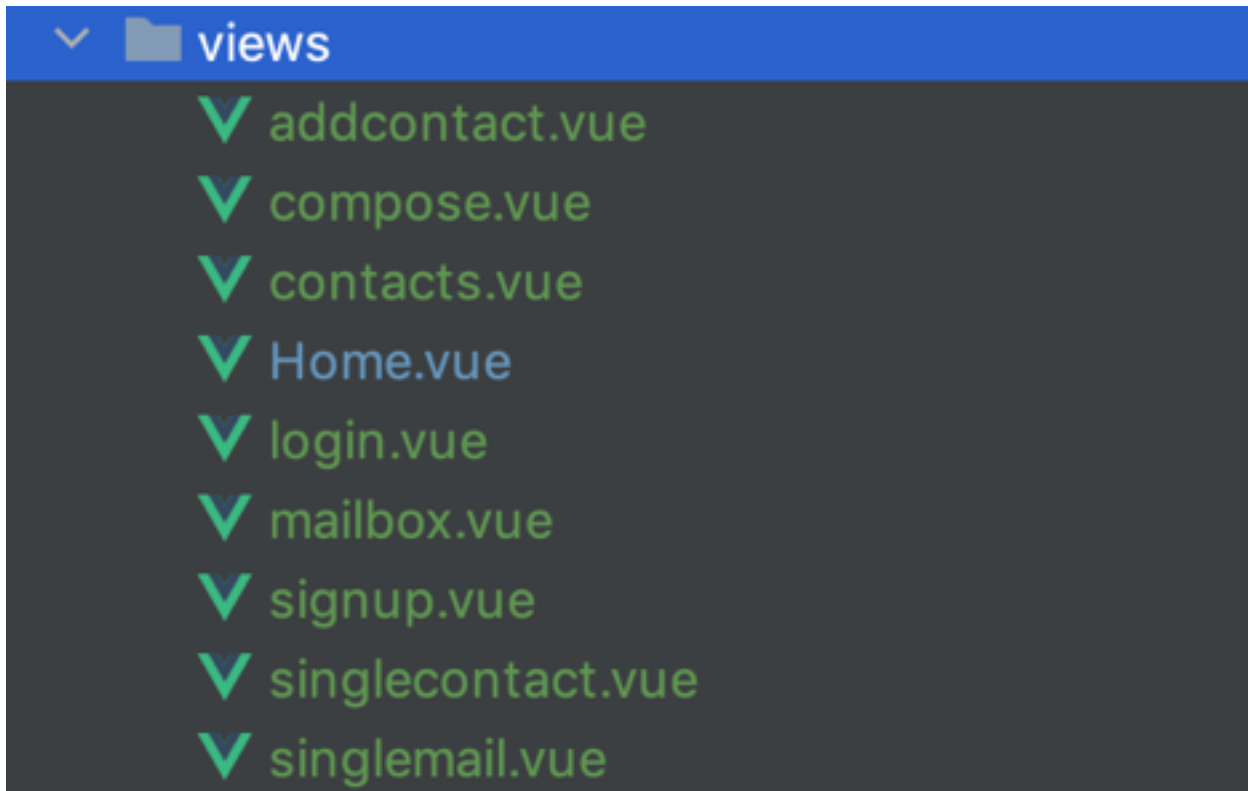
3.HelloWorld:

component used when open the main localhost and it act as welcome screen for the user where it help him go to login or to signup.

4.mail:

component used to show mails in mailbox view, it handle the style, script and template for the mails in the mailbox.

Views:



we made 9 views:

1.home:

it is the main and first view to open at <http://localhost:8080/> it show hello world component.

2.login:

the view shown when logging in it handle the login process in the front end as shown in page 4 in the user manual.

3.signup:

the view shown when signing up in it handle the login process in the front end as shown in page 4 in the user manual.

4.mailbox:

it is view when you login or signup that show you the mailbox of your email with all all other as mentioned in page 5 in the manual.

5.compose:

the view which opens when you write a new mail as shown in page 6 in the manual.

6.single mail:

the view that opens when opening a single mail as shown in page 7.

7.contacts:

the view that open when opening your contact as shown in page in page 8.

8.add contact:

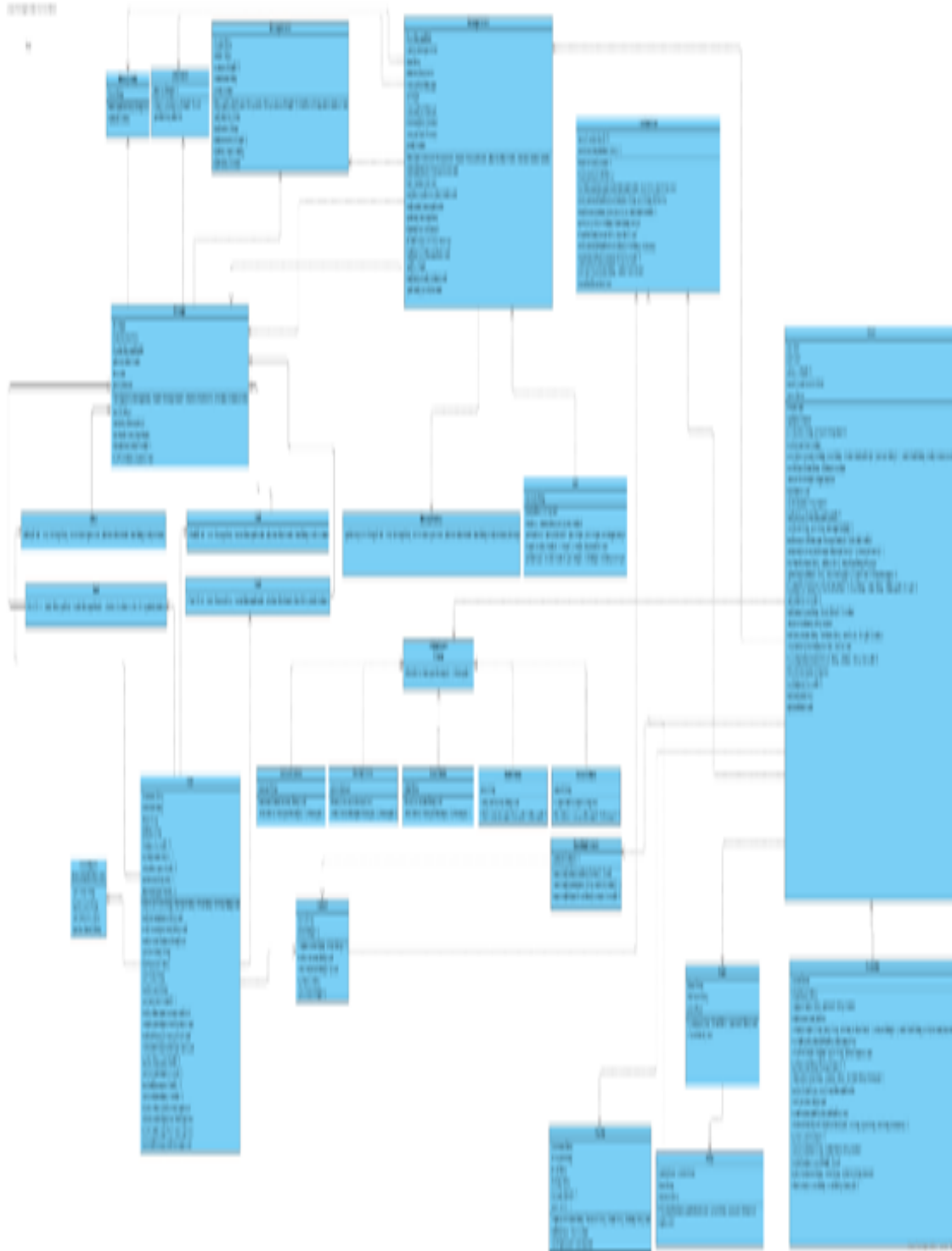
the view that open when making a new contact as shown in page in page 9 in the user manual.

10.single mail:

the view that opens when opening a single contact as shown in page 10.

Backend:

Class uml:



THE IMPLEMENTED DESIGN PATTERNS:

1-Filter Design Pattern:

We have implemented this design pattern in searching for different attributes in message like (Date, Sender, Receivers, Importance, Subject, Body, Attachments) all searching classes use interface Criteria to implement the filter method which take List of Messages then filter it using certain attribute then return the Another List contains the filtered Messages

2-Factory Design Pattern:

We used this design pattern in creating instances of Messages extended Classes “Sent , Draft , Inbox , trash”, in Message Creator in which we implement the Façade design pattern will we explain it ,so we create this instances without expose to the logic of creation.

3-Singleton Design Pattern:

We use this design pattern in Server Class in which we create only one instance of it making the Controller Class changing the Same instance every time it called.

4-Read-Only-Interface Design Pattern:

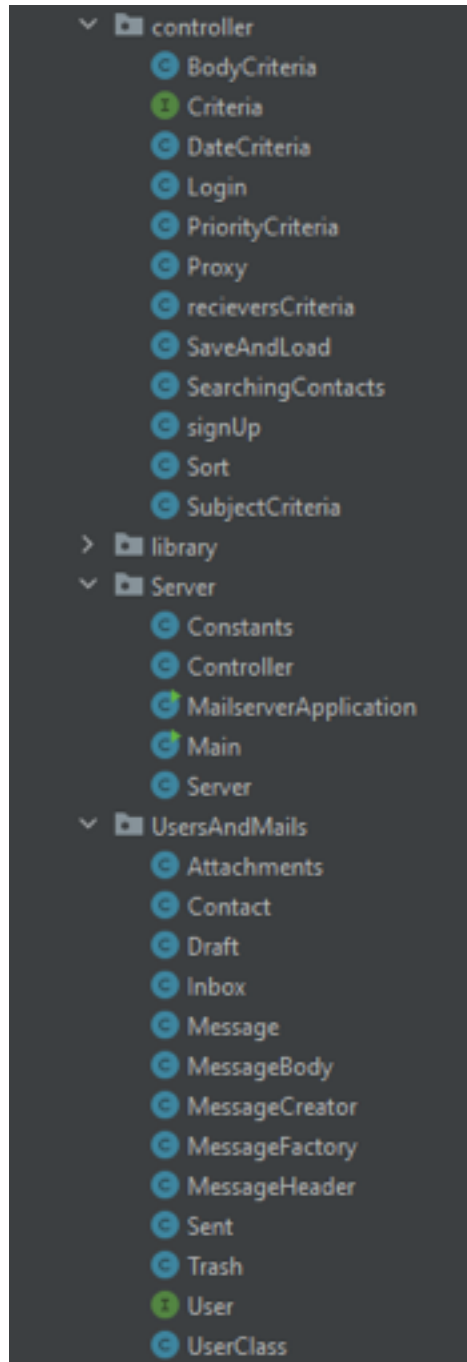
We used this design pattern in User class to allow some class only get attributes of the user object without modifying them and other classes not .

5-Façade Design Pattern:

We use it to make this class implement all the logic and communicate the all set of the related objects which are the message contents we call this class Message Creator which have access to Message header, body , attachments .

6-Proxy Design Pattern:

We use this design pattern for as protection Proxy as we does not access the Client expect in case we match his Email and Password as one we have in our database so make him pass this request to class Login then to our server Class to deal with his Data



Save And Load :

In Save And Load we use the power of **GSON** that enables us to change any class to json and save this in a json file without dealing with anything except the class that help us in save and load messages and dealing with messages easily.

Example of dealing with Gson to convert Class :

```
169 ////////////////add contact function////////////////////////
170 public void AddContact(String userEmail,Contact contact) throws IOException, ParseException
171     JSONArray arrayOfContact;//to get previous contact
172     if(readJsonArrayOfPreviousContact(userEmail)!=null) {
173         arrayOfContact=readJsonArrayOfPreviousContact(userEmail);//get previous contact
174     }else {
175         arrayOfContact=new JSONArray(); //new arraylist if now found previos contact
176     }
177 ////////////////change class to json////////////////////////
178     Gson gson=new Gson();
179     String json= gson.toJson(contact);
180     JSONObject userJson2=new JSONObject();
181     userJson2.put(Constants.CONTACTS,json);
182     arrayOfContact.add(userJson2);/////add json object to the array list
183 ////////////////write new contact to file////////////////////////
184     FileWriter fileWriter=new FileWriter(Constants.DATABASE_PATH+userEmail+"//"+Constant
185     fileWriter.write(arrayOfContact.toJSONString());
186     fileWriter.flush();
187     fileWriter.close();
188 }
189 ////////////////make first directory////////////////////////
190 public void makeFirstDirectory() throws IOException {
```

we dealing with this part by composite it to three parts :

first part users and their information.

second part messages :

third part is Contacts of the users :

folders structure of saving and loading procedural :

first we save users in directorydata_base//Accounts//index.json

```
1 [{"user":{"username":"asad","password":"12345","Email":{"emailID":"asad@","Birthday":{"2020-12-24"}}},
2 {"user":{"username":"omar","password":"456","Email":{"emailID":"omar@","Birthday":{"2020-12-20"}}}]
```

we save messages and contact after user sign up with folder name is Email of user and make the following folders

Name	Date modified	Type	Size
Accounts	T-T-/1T/T7 µ ·~·-1	File folder	
omar@	T-T-/1T/T7 µ ·~·-T	File folder	
oef@	T-T-/1T/T7 µ ·~·-1	File folder	

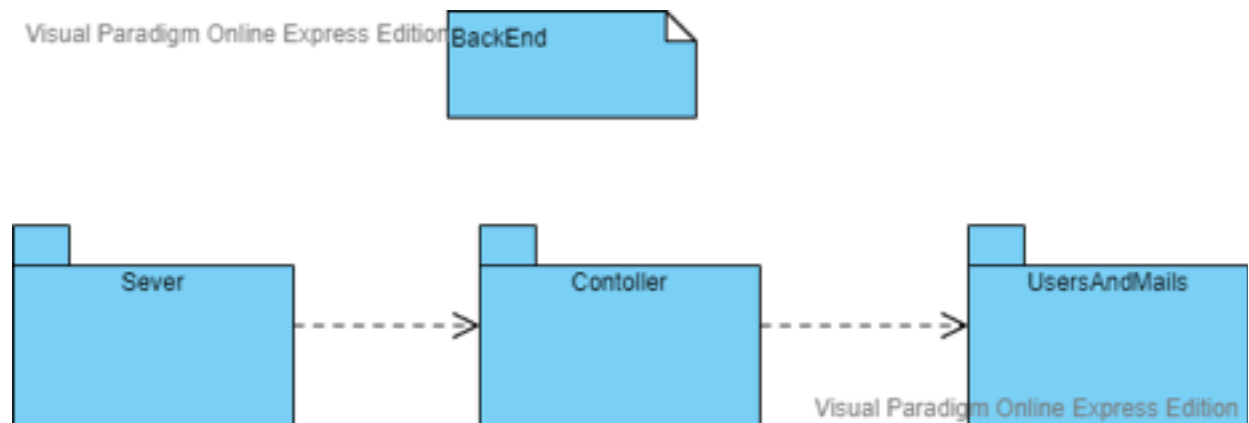
name	Date modified	Type	Size
Attachments	T-T-/1T/T7 µ ·~·-T	File folder	
Contacts	T-T-/1T/T7 µ ·~·-T	File folder	
Draft	T-T-/1T/T7 µ ·~·-T	File folder	
Inbox	T-T-/1T/T7 µ ·~·-T	File folder	
Sent	T-T-/1T/T7 µ ·~·-T	File folder	
Trash	T-T-/1T/T7 µ ·~·-T	File folder	

we save attachments L in folder Attachments but we save the directory in messages
save attachments of draft
save attachments of sent
save attachments of inbox

name	Date modified	Type	Size
AttachmentsDrafts	T-T-/1T/T7 µ ·~·-T	File folder	
Inbox	T-T-/1T/T7 µ ·~·-T	File folder	
sent	T-T-/1T/T7 µ ·~·-T	File folder	

Notice about sort :

we use in the sort mechanism **Binary sort** using data structure we **implement last year using** double linked list and stack we choose it because it will be the most efficient



As it is clear our backend is consist of three packages

1. Server: which is responsible of sending and receiving requests
2. Controller: which responsible of all logic in the program as searching , sorting , creating ,....
3. UserAndMails :which is responsible for containing all basic objects for our Program