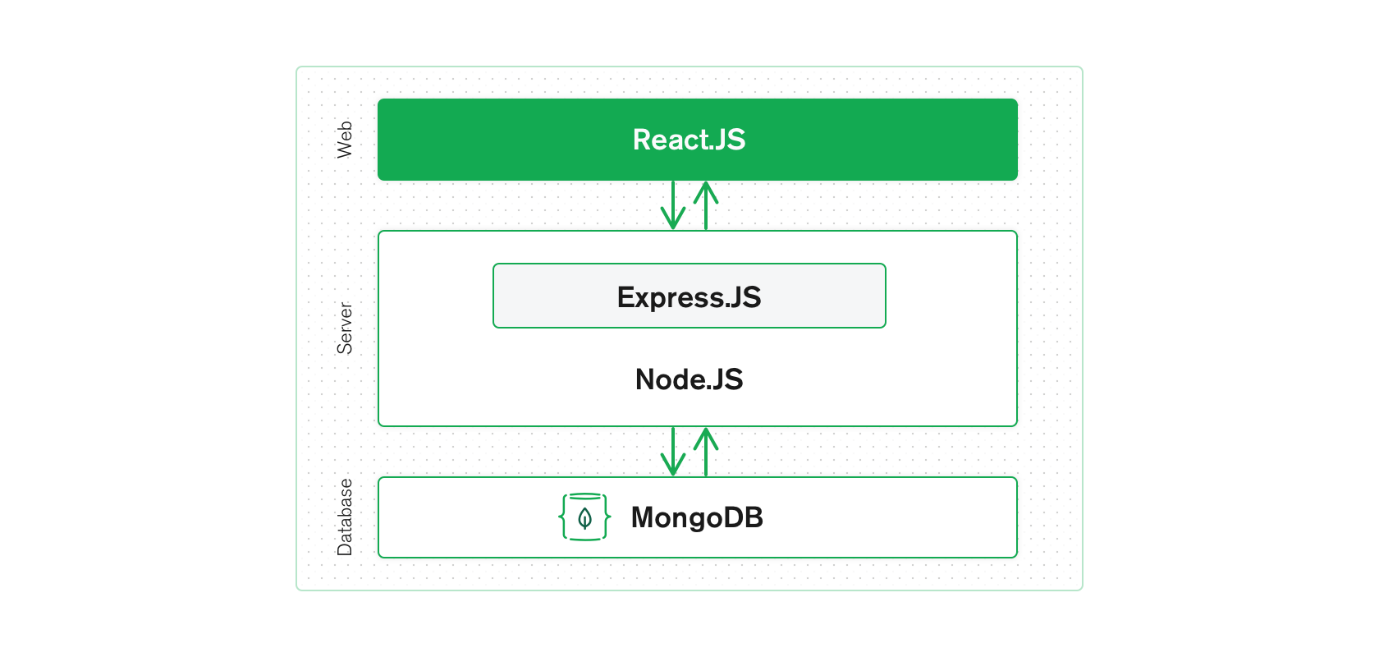
**Gmail Like Web Application**

**Introduction**

In this Project, I will be making a Web Application which will function just like a Mail App, hence the name. This Project will be made in four major technologies which are: MongoDB, Express.js, React, Nodejs, making this a MERN Project. This project will consist of a proper mailing feature with which you will be able to send any mail with any type of attachment you wish for, i.e. jpg, png, xlsx, pdf, etc. You can also mark any mail as starred, and view the starred mails in a different section of it’s own. Similarly, you have sections like “Sent”, where you can view all the mails that you send to another user. “Draft” section consists of all the draft mails, where you can select any mail and continue right where you left off. “Bin” section consists of all the mails which you deleted. You can also click on “Compose” to write a mail and either save it to a draft by cancelling it, or send it to the respective user you typed by clicking on the “Send” button. You can also view your account, update information about your current profile, such as profile picture, username, password, age. You also have your contacts where you can add a contact by providing it a nickname and an email, and also delete a particular contact.

**Technologies Used:**

* **MongoDB:** A NoSQL database for efficient and scalable data storage.
* **Express.js:** Serving as the backend framework, Express.js enables the creation of a powerful and flexible API.
* **React:** The frontend is developed using React, allowing for a dynamic and responsive user interface.
* **Node.js:** As the runtime environment, Node.js ensures efficient server-side execution.

**Key Node Packages Utilized**

* **bcrypt:** Employed for password hashing, ensuring the security of user credentials.
* **jsonwebtoken:** Facilitates tokenization for user authentication and session management.
* **mongoose:** Used to establish a connection between the application and the MongoDB database.
* **nodemailer:** Enables seamless email communication, allowing users to send and receive emails.
* **axios:** Integrated into the React frontend for making API requests to the backend, facilitating CRUD operations.

**Database Structure**

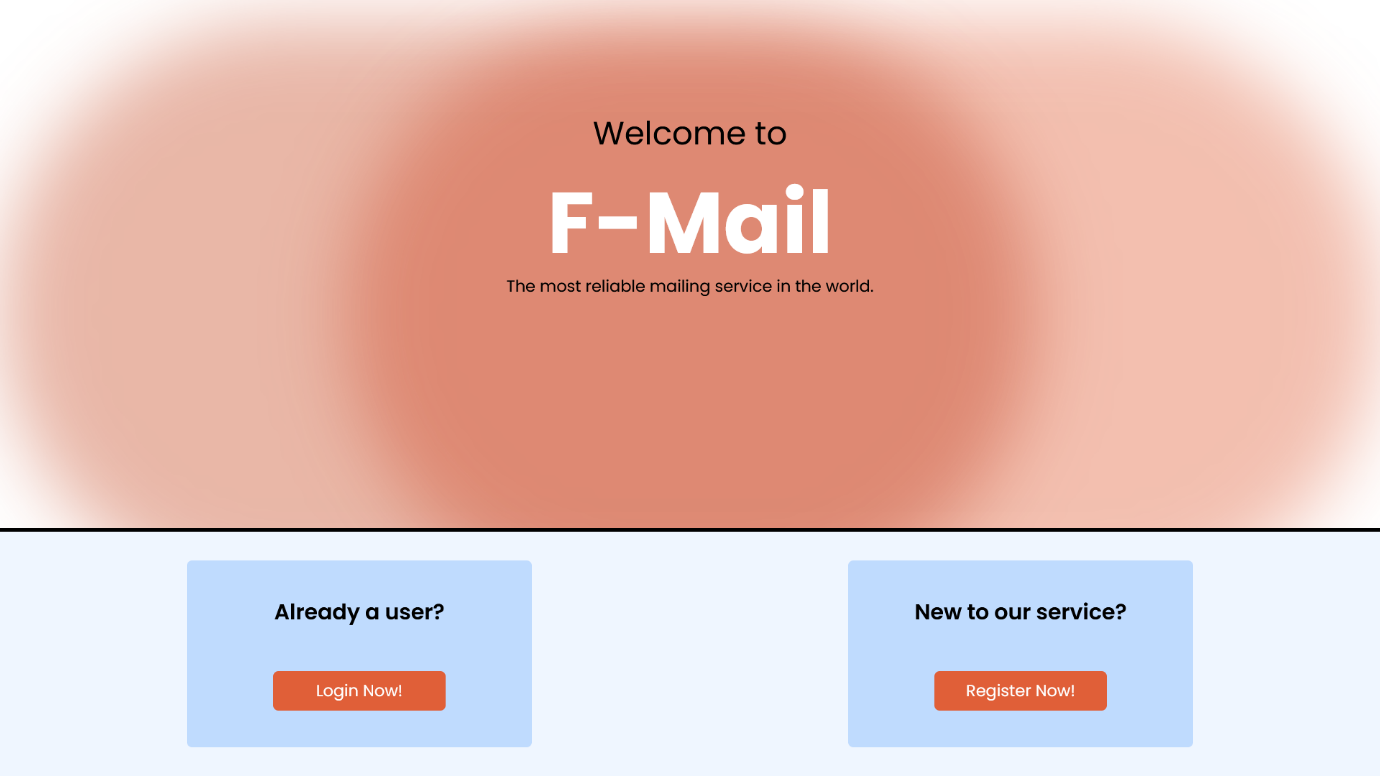
The application utilizes MongoDB to manage data efficiently. Three distinct collections have been implemented:

1. **Users Collection:** Handles user-related information, including authentication details.
2. **Mails Collection:** Manages email content, providing a structured storage system for sent and received emails.
3. **Contacts Collection:** Stores contact details, streamlining the process of composing and sending emails.

**Overview**

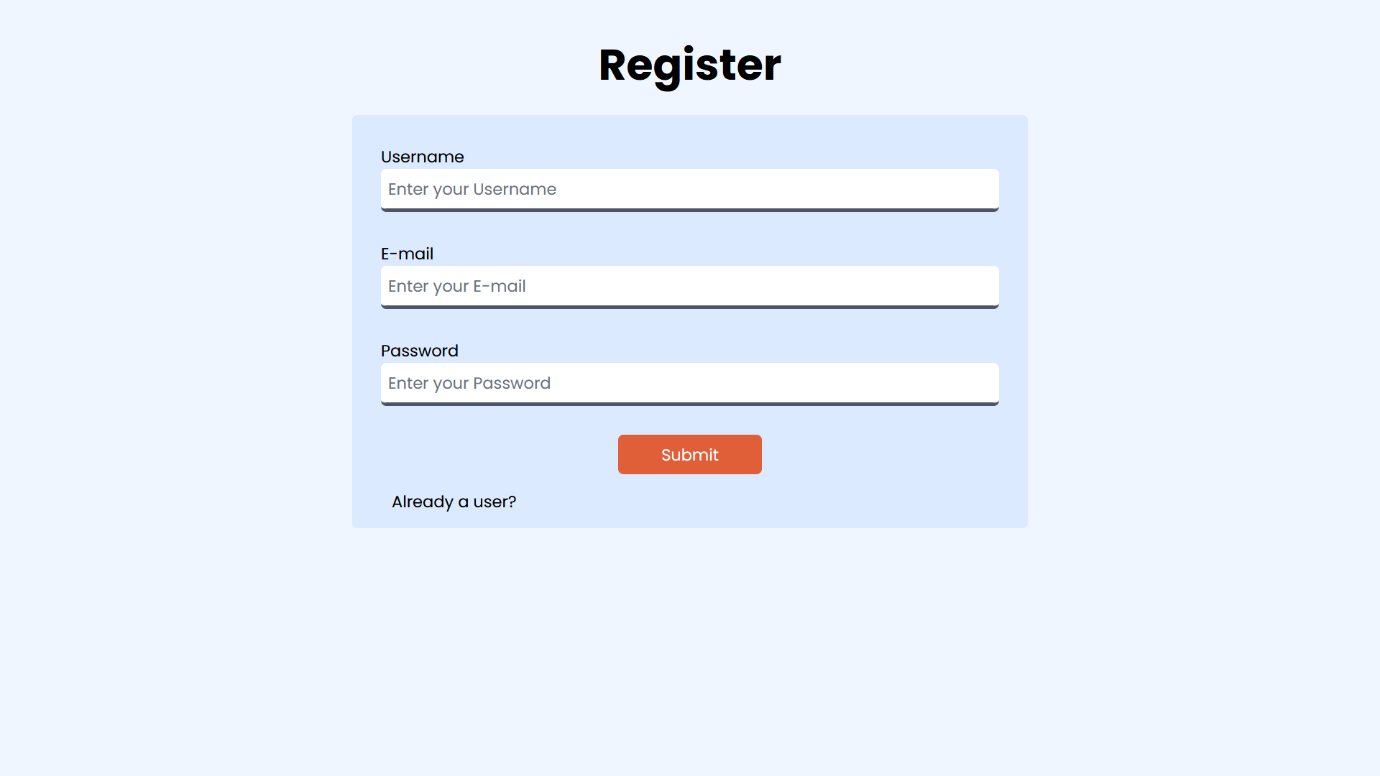
As mentioned earlier, I have used technologies which are: MongoDB, Express.js, React, Nodejs, making this a MERN Project. Apart from this, I have installed and used different node packages like bcrypt for hashing the password, jsonwebtoken for tokenization and maintaining a user session, mongoose for connecting to my MongoDB database, nodemailer for sending the mails to any user. And in React, I have use axios for making different requests to my backend for performing CRUD operations on my database. I have made 3 different collections in my database, one is for handling the users, another is for handling mails, and the last one is for handling the contacts. For styling the whole application, I have used Tailwind CSS which is a utility-first CSS framework and makes styling pages simpler.

**Welcoming Page**

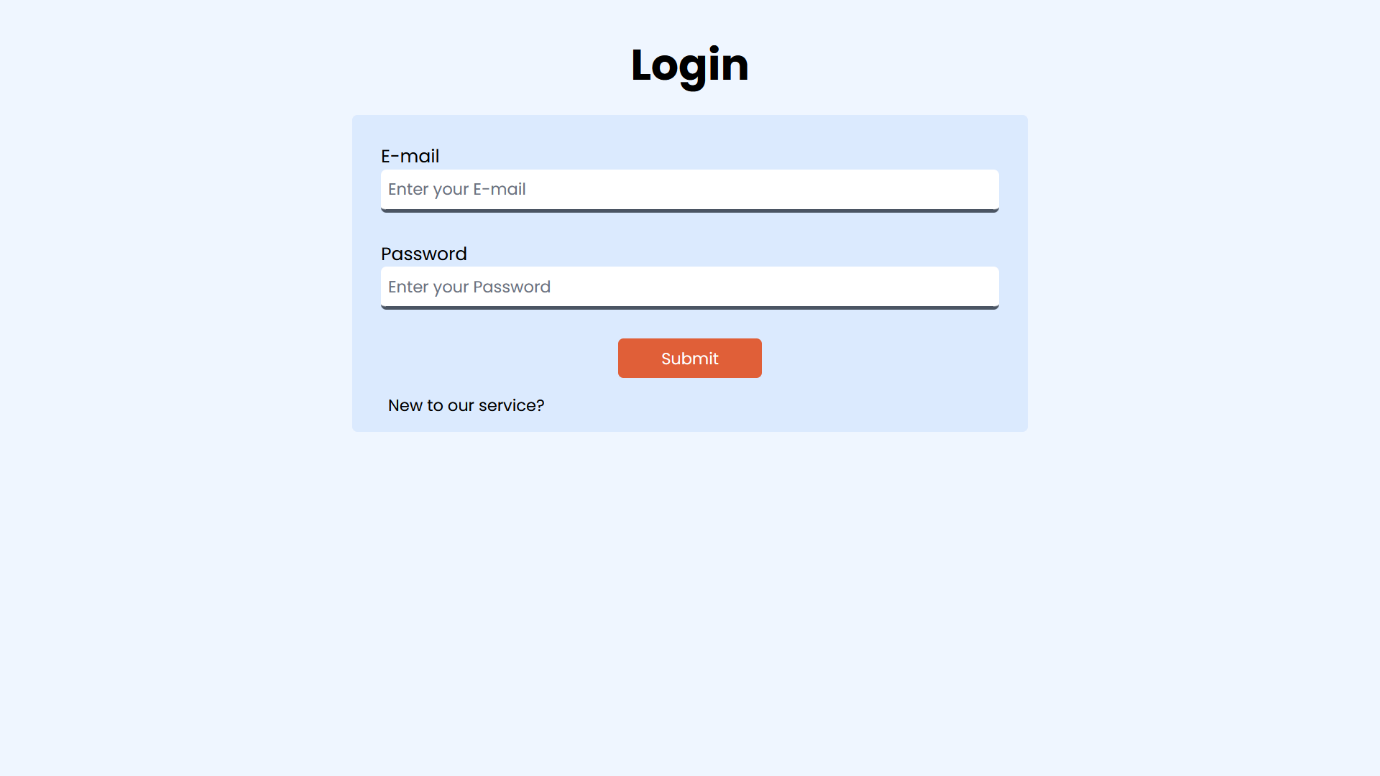


This project keeps the “Welcome” page as simple and as user friendly as possible. You are given two different choices based on whether you are a user or not. While clicking on the two buttons labelled “Login Now!” and “Register Now!” you are taken to their respective pages.

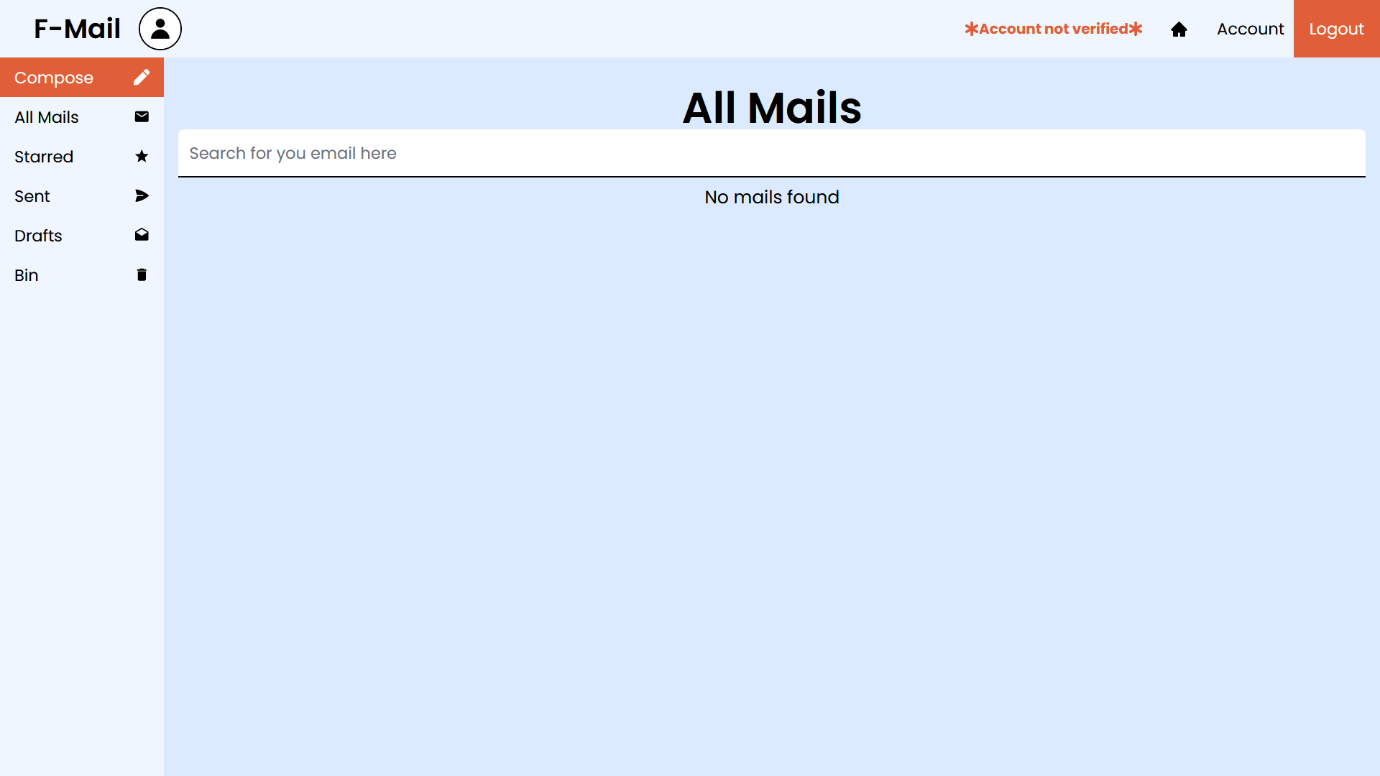
**Register Page**



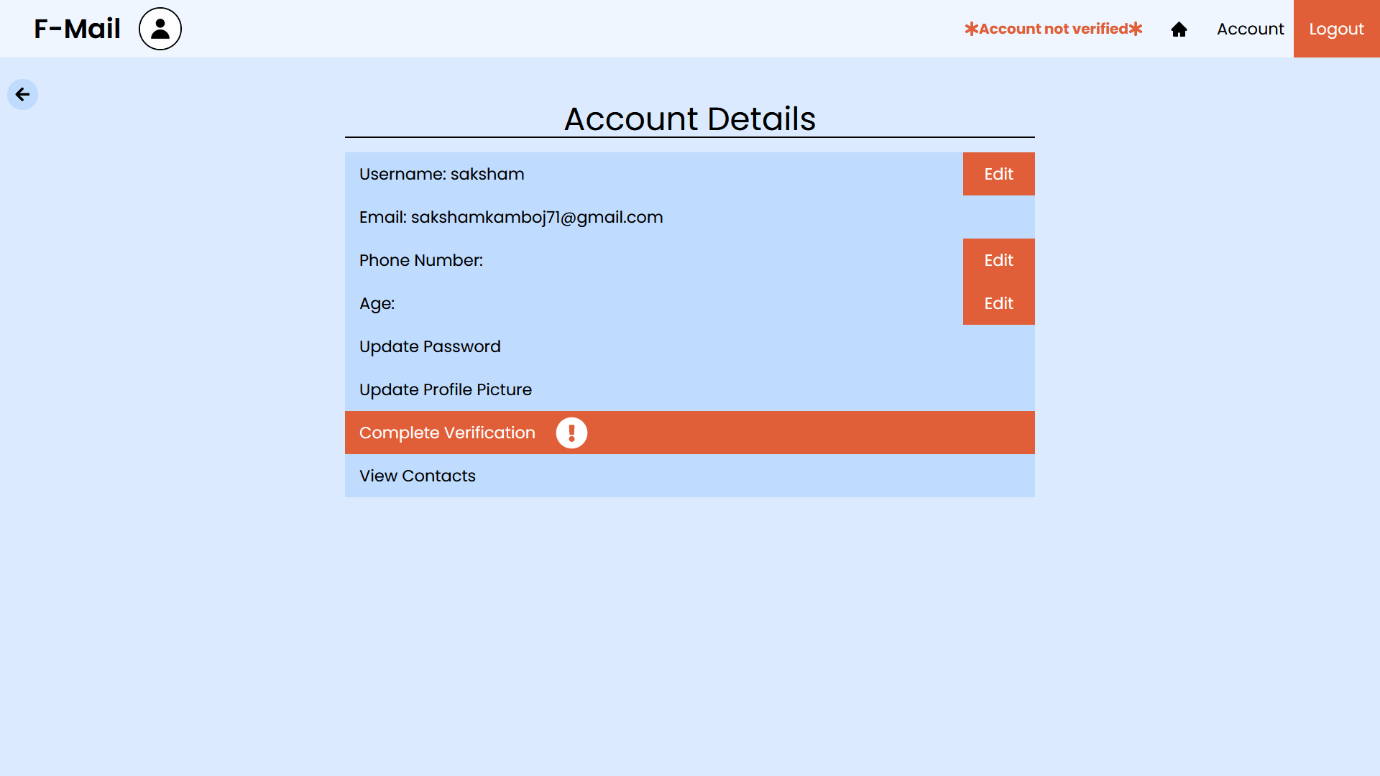
Again, keeping the registration process as simple as possible, you are to provide a particular username to your liking. The only condition being that your username must be unique, which is checked if it is present in the database or not. If the username or the mail already exists in the database, means that the information you have entered is already registered. If there is any error, it will be displayed at the bottom of the form, and you will not be able to submit the form. Once it is clear of errors and then if submitted, the data will be sent to the backend and it will create a new User Model and upload it to the “users” collection in the database.

**Login Page**

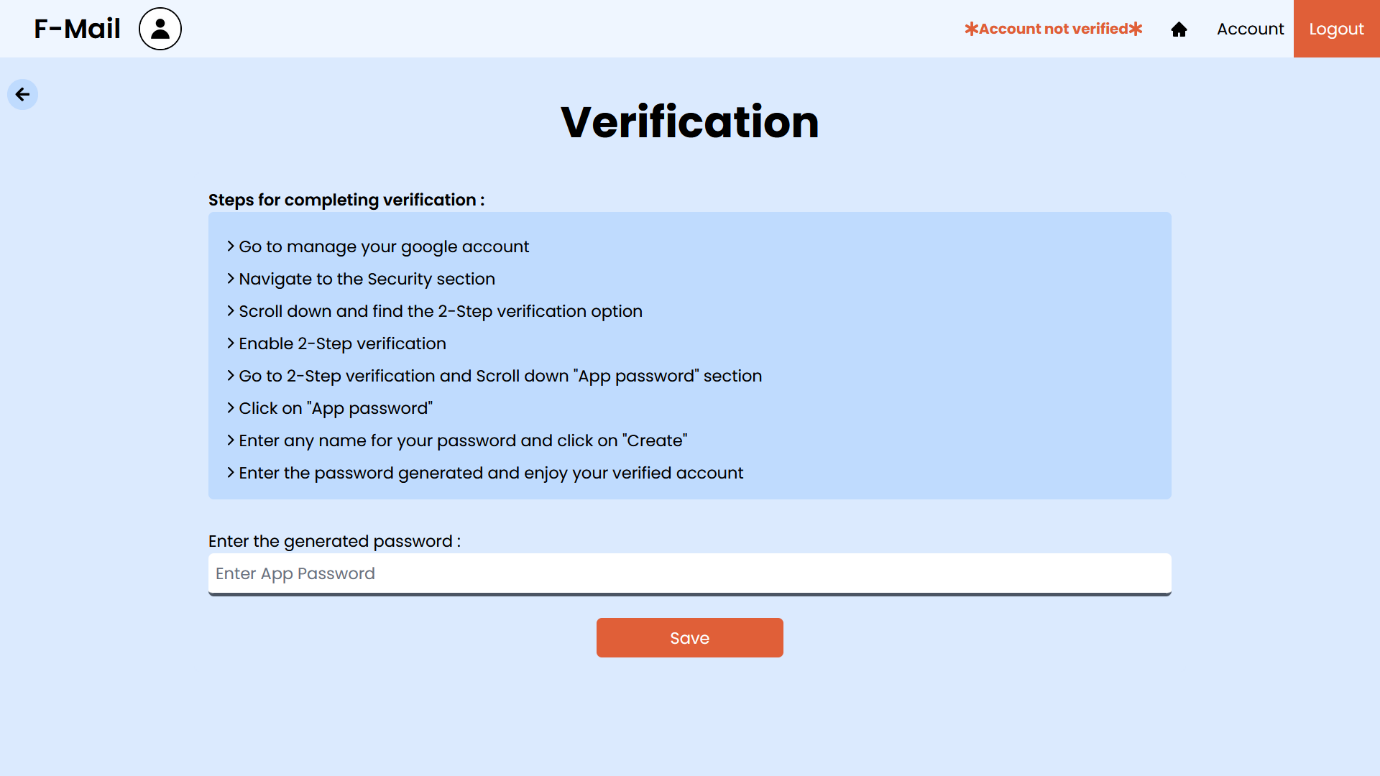
Once you are registered, you will be redirected to the login page where you can go ahead and log into the application. Once you fill your details, they are checked in the backend. If any error, it will display it at the bottom of the form. If there are no errors, means the email and the password match, and you will be logged in. The login process is simple. Your user ID which is created in the database, goes on to the jsonwebtoken, which then creates a particular token based on a secret key. The token is then stored in the local storage. You can set the token to expire in a particular time, which will automatically log you out of the application once the timer is reached.



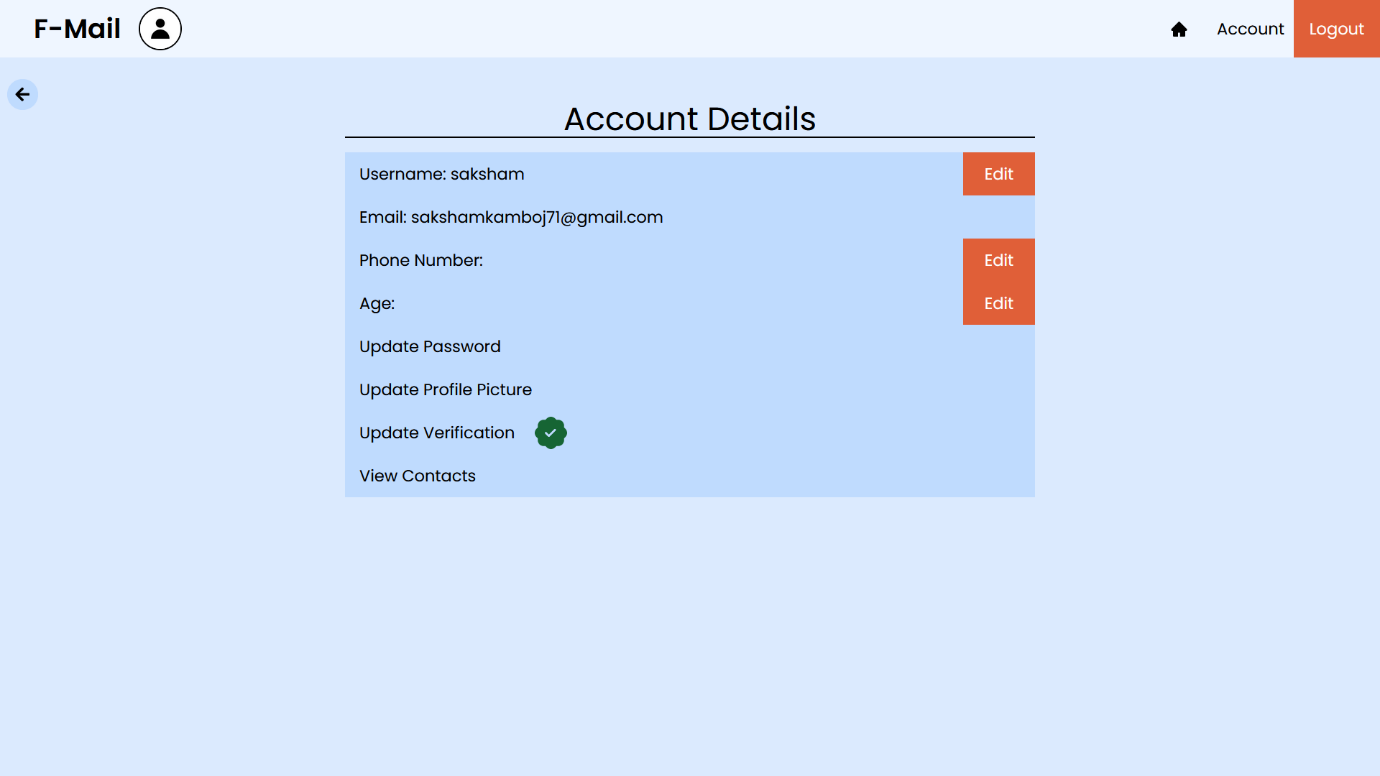
You will be then redirected to the home page, which is also the place where you can view all your mails. Currently the database is empty, so we will come back to this page once we fill the database. First let us see at a problem showing up in our Navbar. If you look at the navbar, you will look at the application name and the user profile picture, which is set to the default because we did not provide it with a particular picture. But at the right side of the navbar, we will see an error that says “Account not verified”. You must be wondering what it means. It means basically that you have not yet verified the account obviously. But in this case, verification means that you will have to verify it by putting in your app password from your official mail account. As this application is basically a platform, you will still have to provide nodemailer an authentication to send a mail. On the right to the error, we see the “Account” tab, and also the “Logout” button. On clicking on the logout button. We will just delete the token and the user ID from the local storage of the browser, and you will be navigated back to the “Welcome” page. Let us first verify our account, as if we try to compose a mail and send it, it will throw an error that our account it not yet verified, and basically none of the features will work, so the verification of the account comes first.



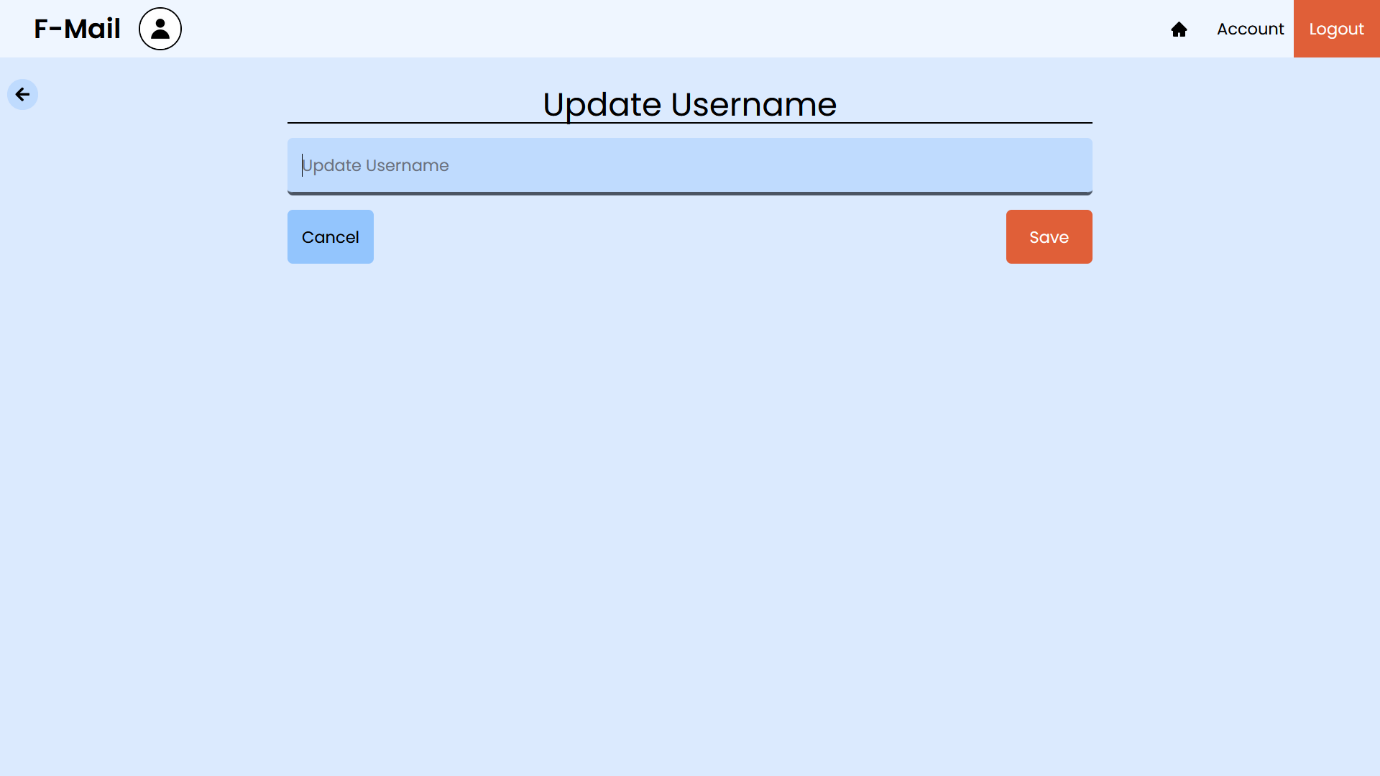
This is the “Account” page. You can see all the information here and all the update and edit buttons. The information of a particular user is fetched from the database by the backend and is stored as an object. From that object, all the information is then displayed, If there is no field to display, then it remains empty. But first our concern is the verification. As you can also see, the “Complete Verification” button is highlighted. Let us navigate there by clicking on it.



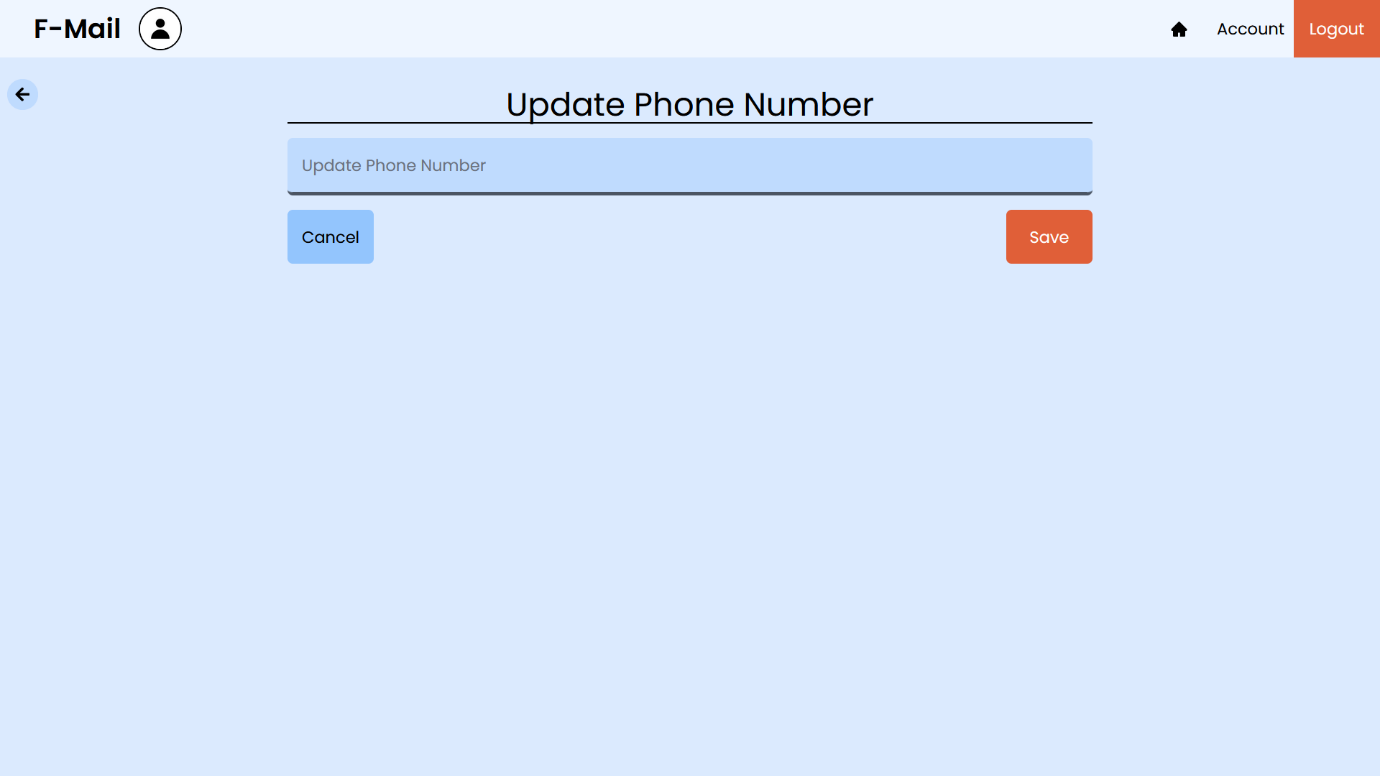
In the verification page, there are multiple steps listed on how you can verify your account. By completing them you get your respective “App password”. Fill it in and your account is verified. Let us navigate back to the account page after filling in the app password.



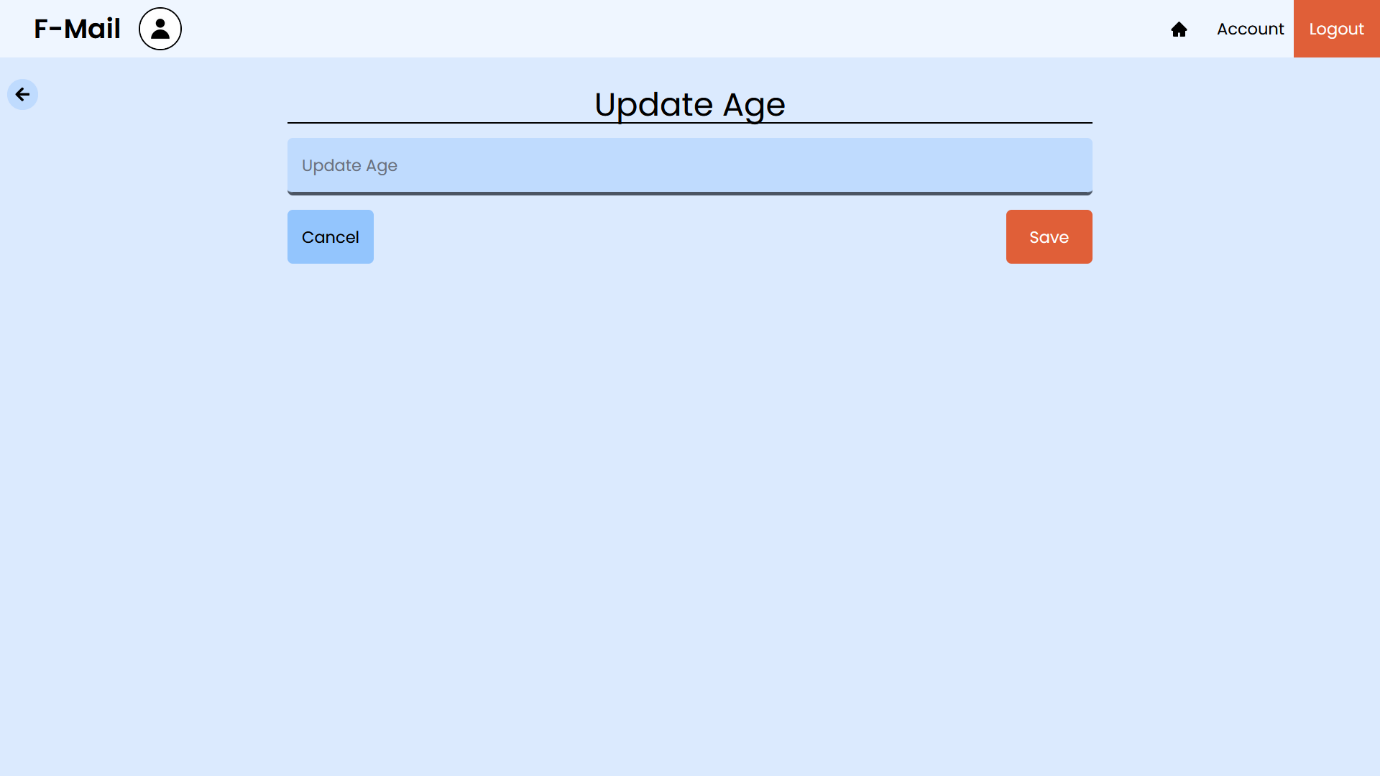
The error on the navbar is gone now, and the “Complete Verification” is is also gone and is replaced by “Update Verification” if you entered the wrong app password. Now let us look at updating the information by clicking on each button.



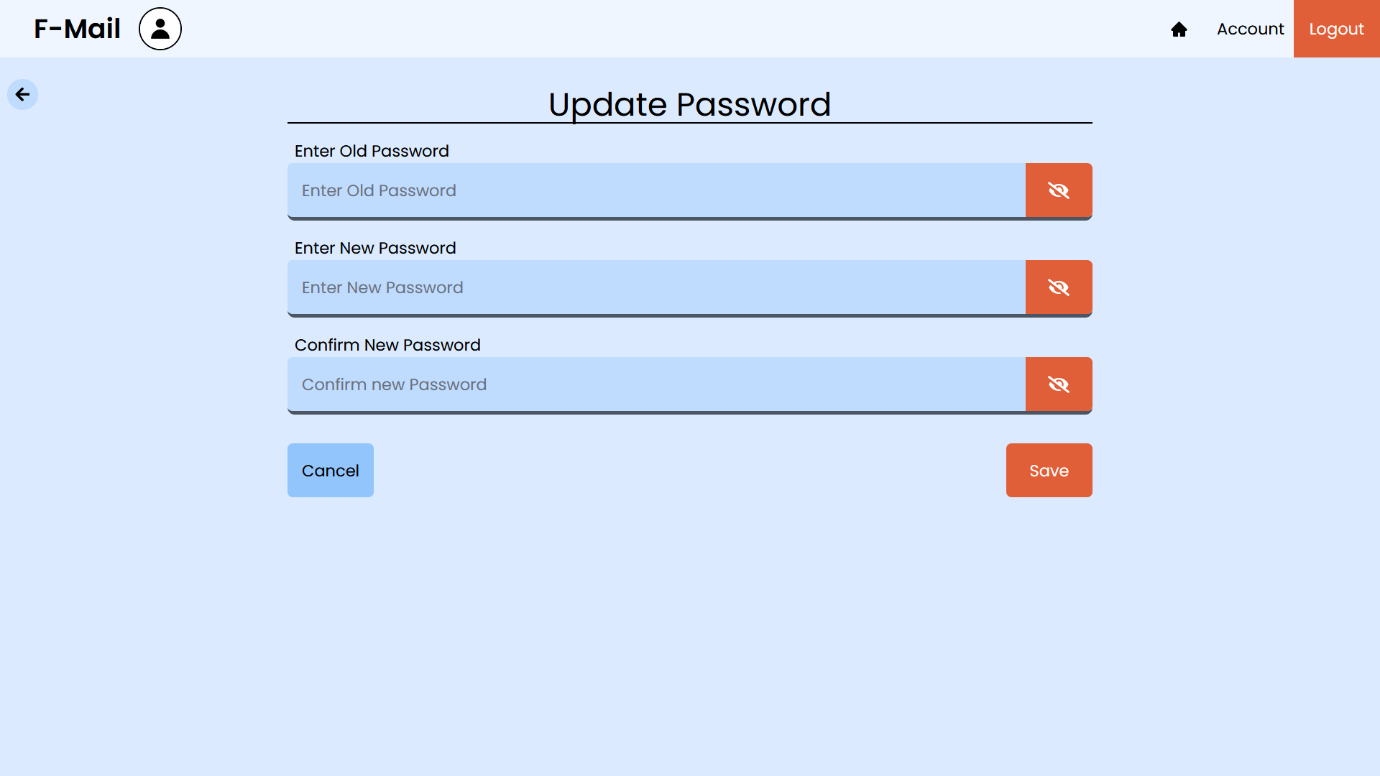
The Update Username page is pretty simple. You just fill in your new username, it will check if it already exists or not, and then if the username does not exist in the database, it will update it.



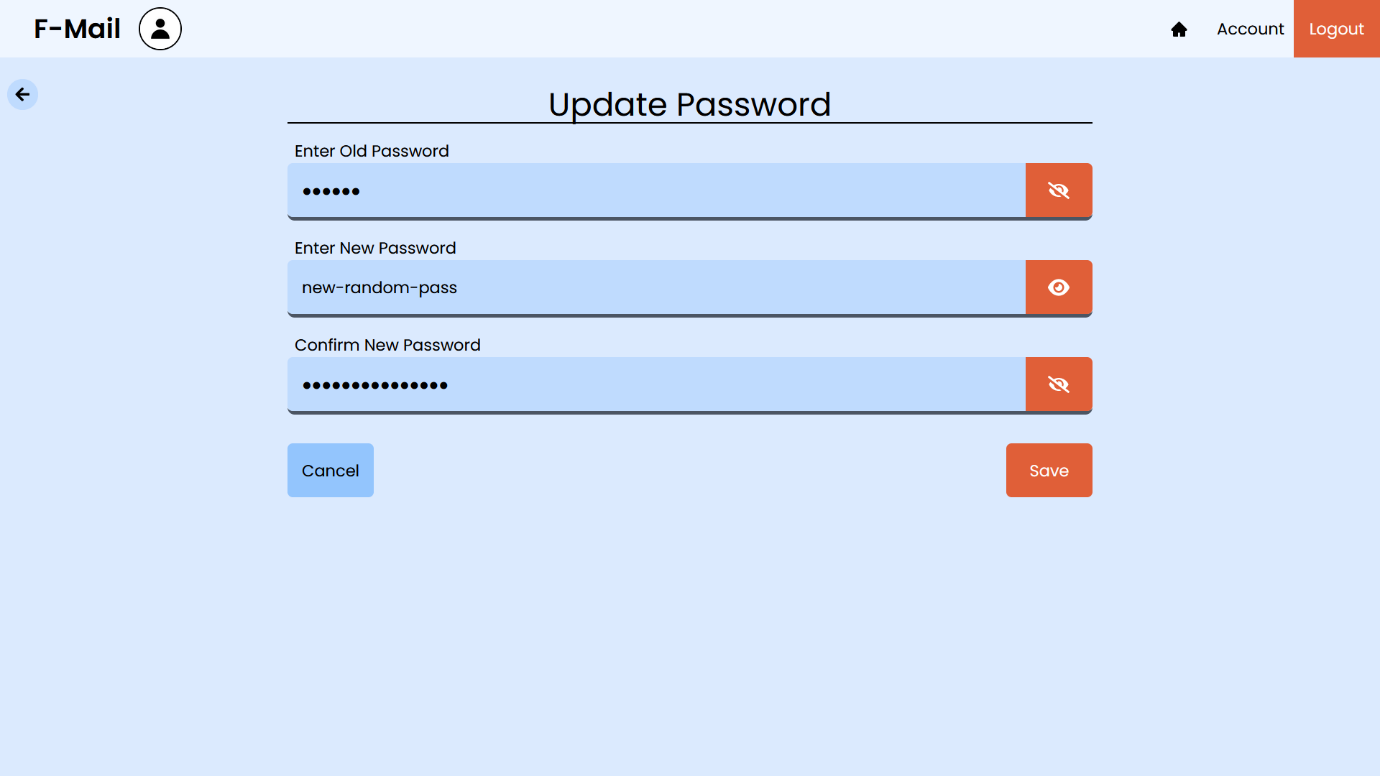
Updating the phone number is also simple, The field of the phone number just cannot be null or empty. The error will be displayed the same way as the update username page. The limit of the phone number is strictly 10 numbers, neither more nor less.

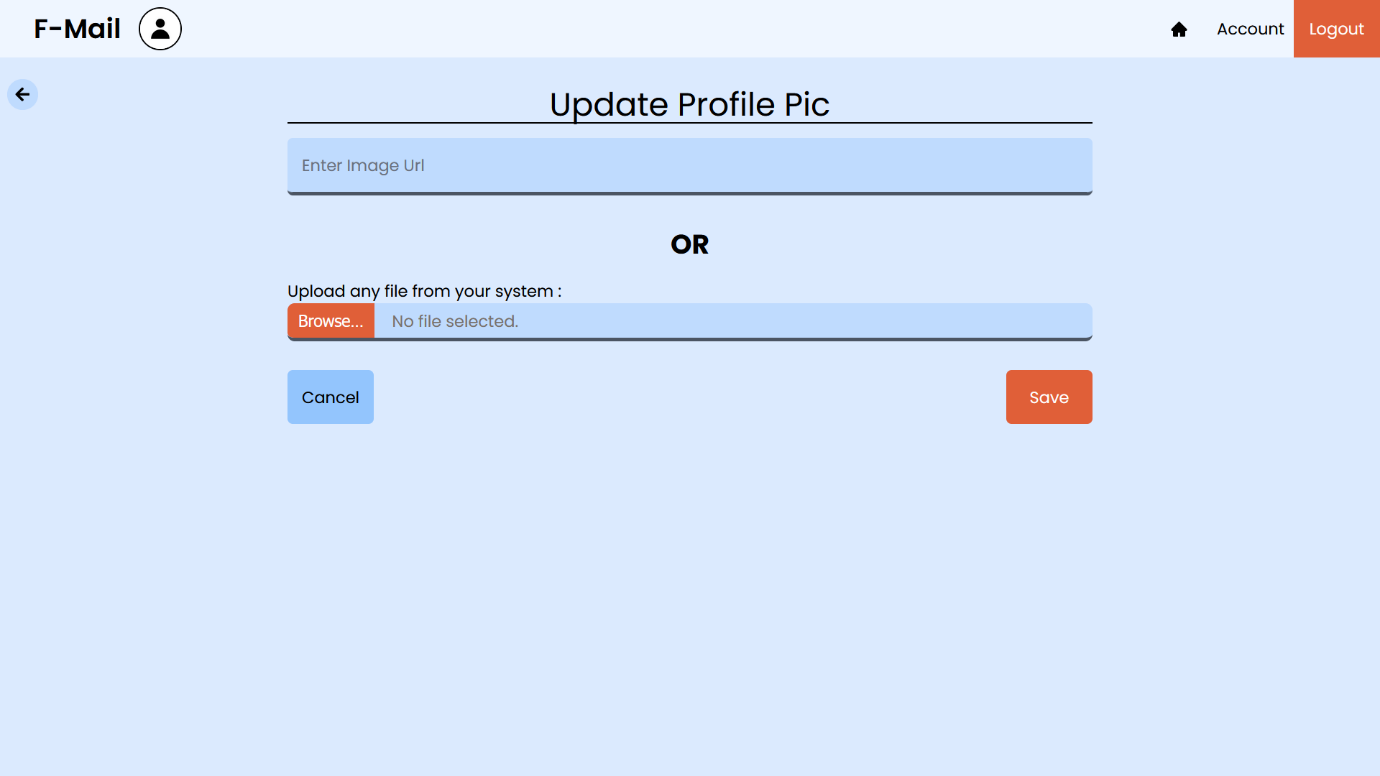


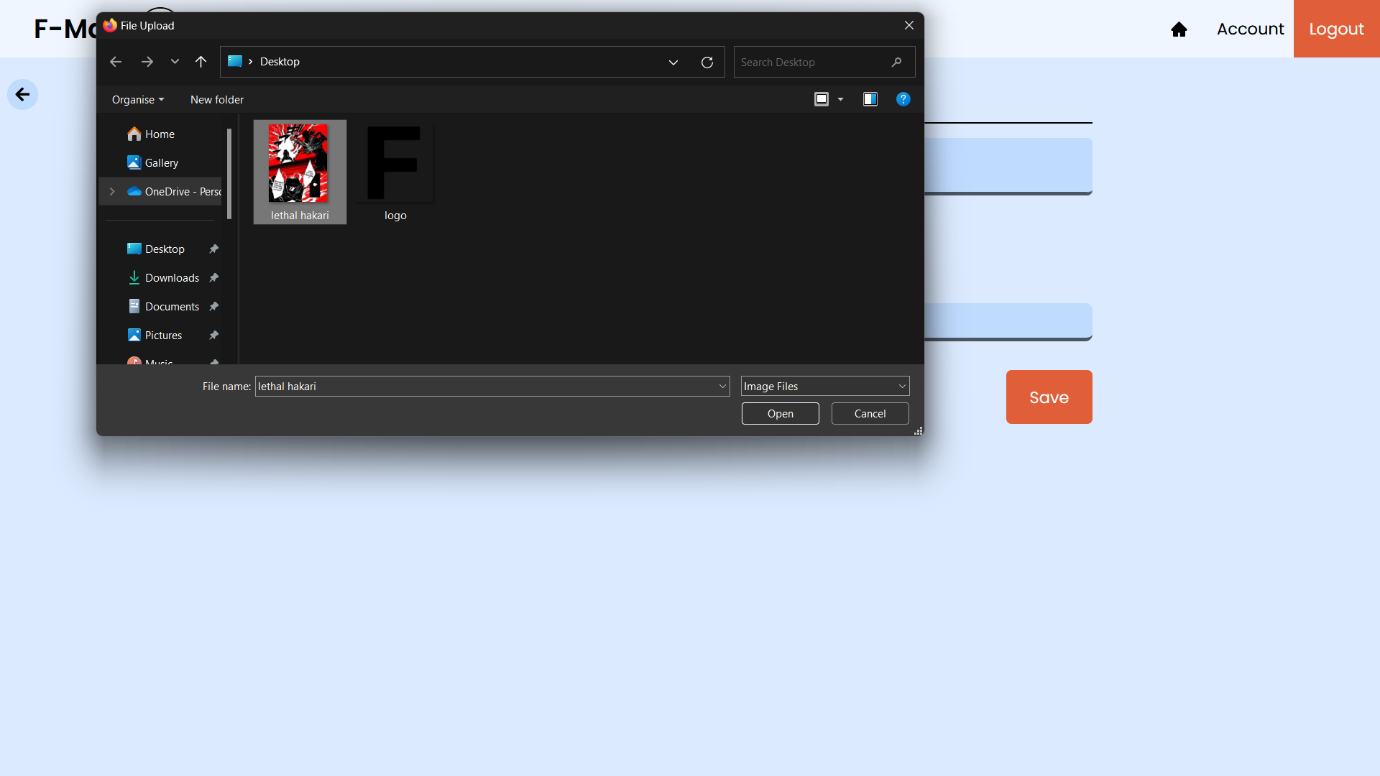
This is the update age page. It will update the age section in the accounts tab.

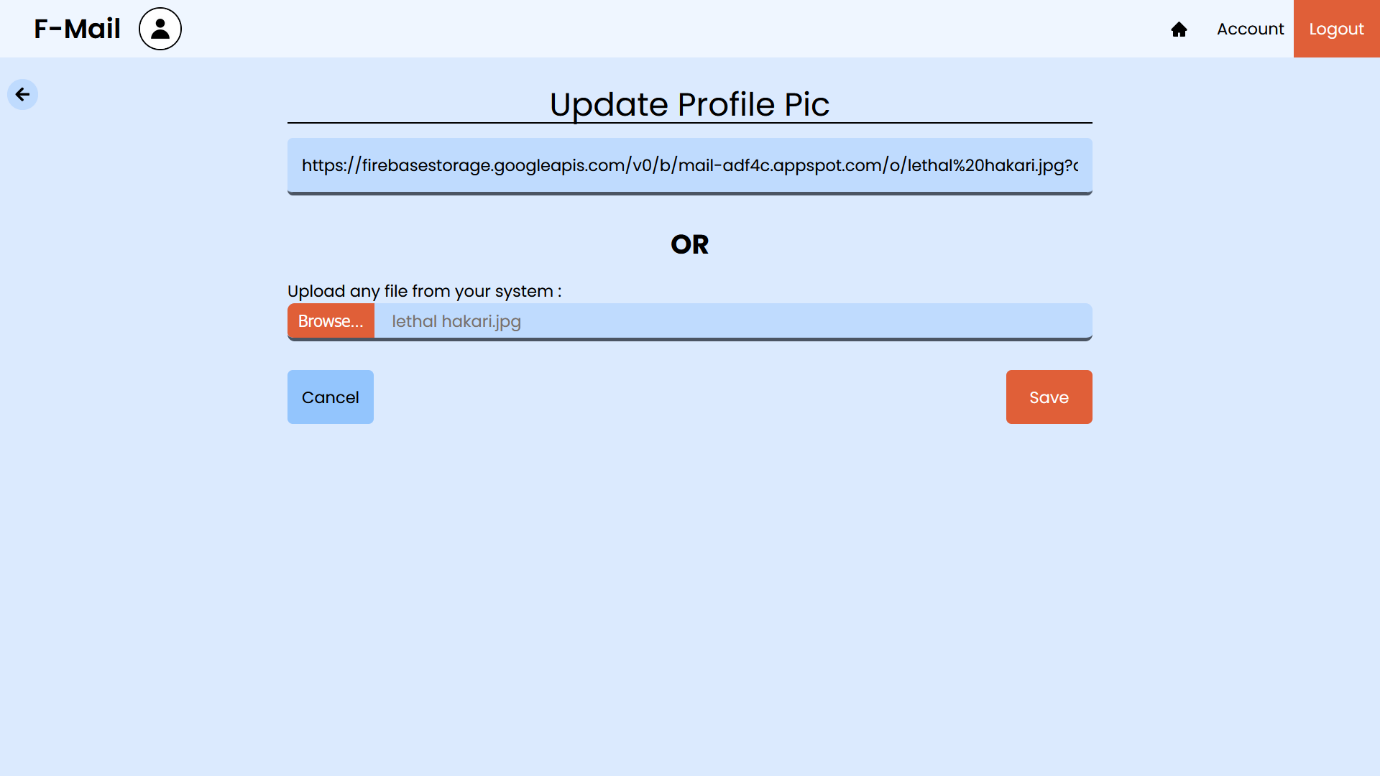


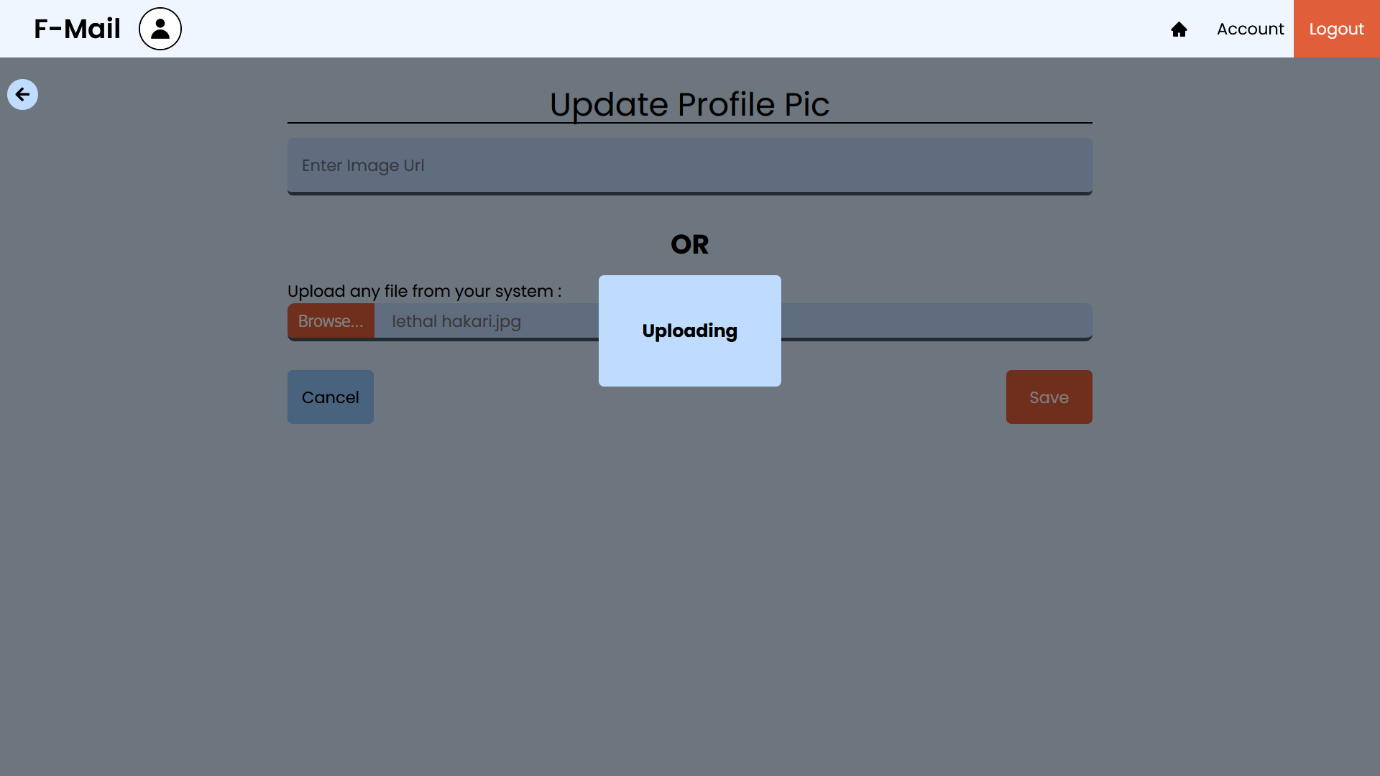
The “Update Password” section is used to update the current password of the user. Once you enter the old password of the user, you will have to enter the new password and confirm it as well. If and only if the old password matches, you will be able to change the password and that too if the new and the confirm passwords match. If you are not sure what you write in the fields, you can make it visible by clicking on the eye icon on the right side of the input fields. The errors that occur on this page are displayed on the bottom side of the form. Let me show you what it looks like when you make the input field visible.

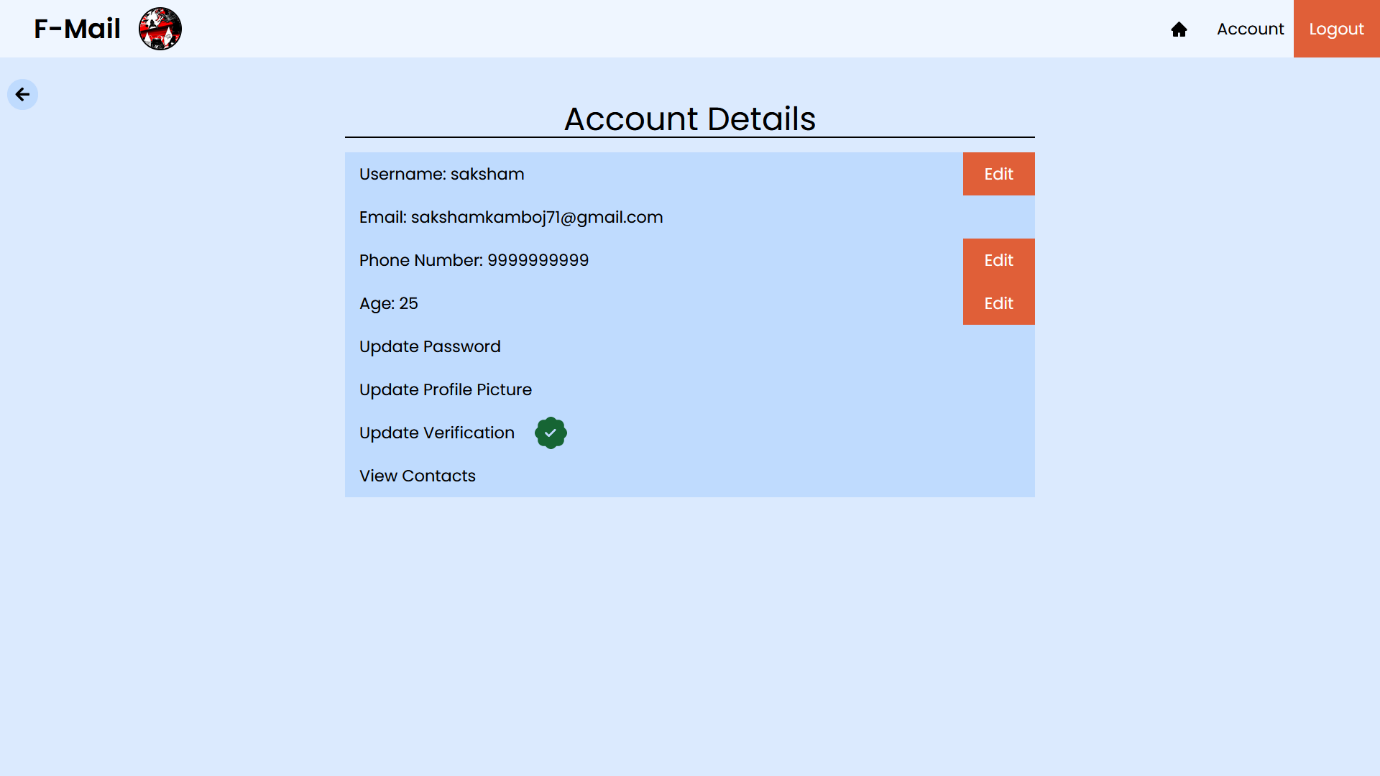




This is the “Update Profile Pic” page. There are two options in front of you. One is the simple image URL option, another is the browse option, which strictly inputs any image from your device which is less than 1MB in size. To handle and store the images, I have used firebase for the storage of the images. Once you upload any image in the input field, you receive a popup until the image is finished uploading.­­­­ Once the image is finished uploading in the firebase cloud, it will return the image’s URL and fill the “image URL” section of itself. Then you can simply click on save and it will update the user profile picture. The image URL feature is basically copy pasting the image from the web, but down below is the whole process of uploading the images.

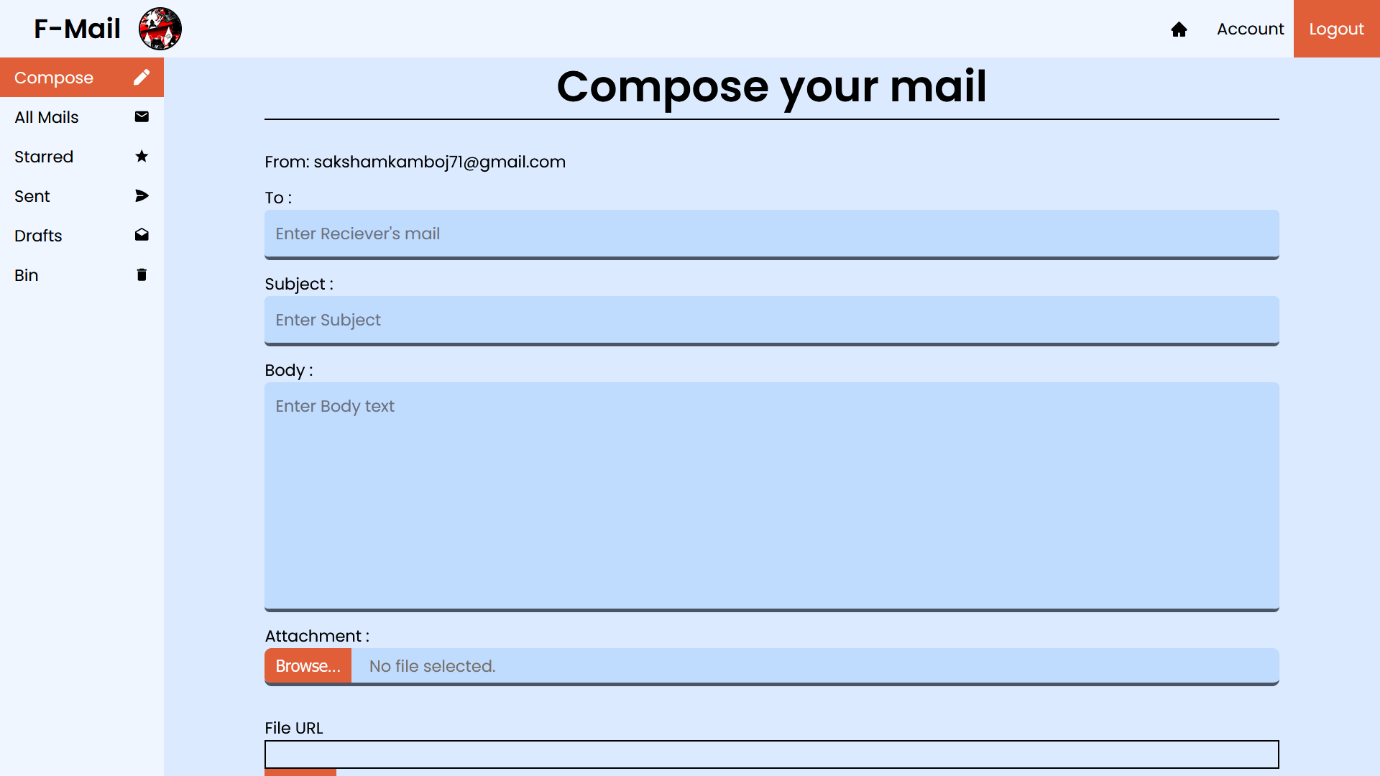


First you click on “browse” and select the image you want to upload. Then you receive a popup saying “Uploading”. It takes a few seconds and once it is done uploading, It will be stored in the firebase and then we will receive the image URL automatically in the URL field. And once you click on “save” button, it will save the following URL in the database of the particular user, the same functionality will be for updating the profile pic of the user.

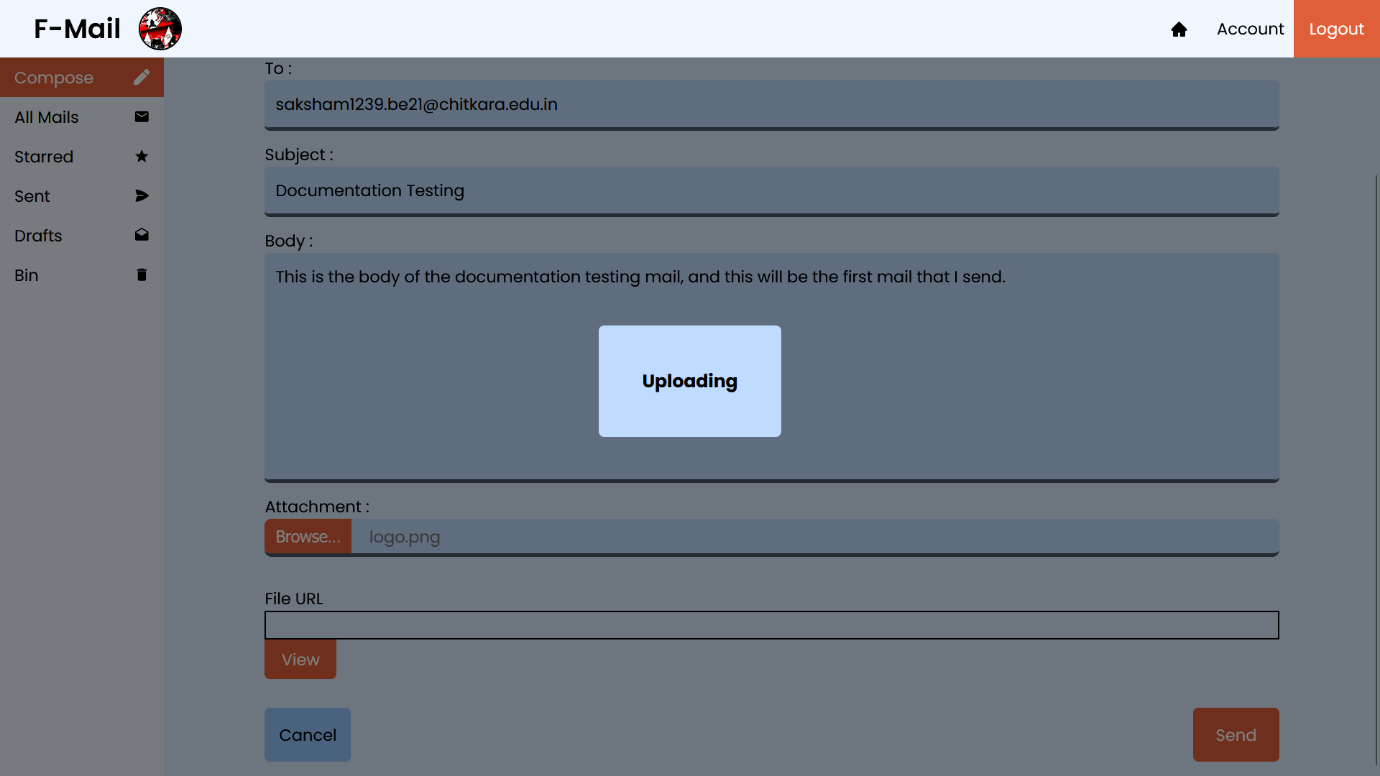


Here are the new updated details, these are not the real details, but only some random data input. The phone number and age are not compulsory to send a mail, so they are optional. You can also see that the profile pic also got updated. The “View Contacts” button will be talked about later in this documentation.

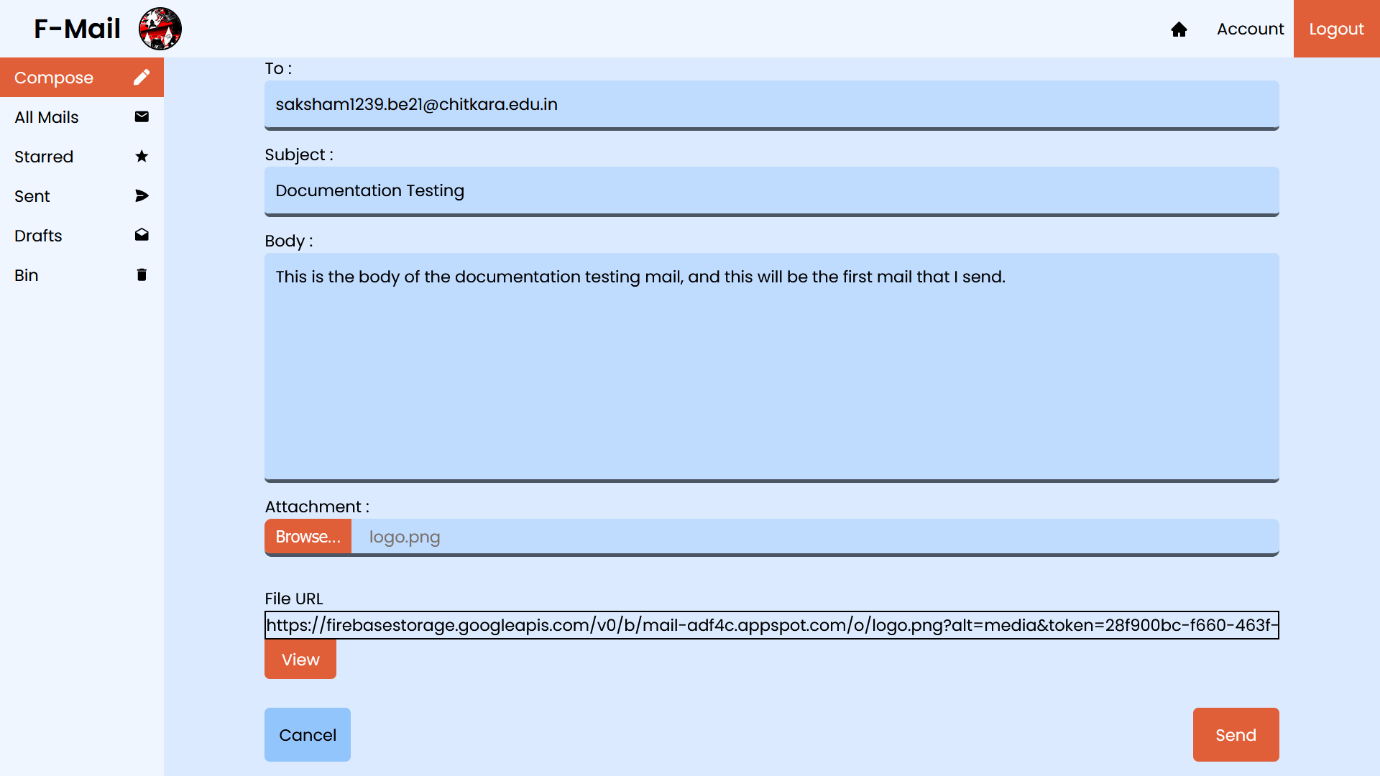
Now, the only thing left to explore is the whole “Mail” section.

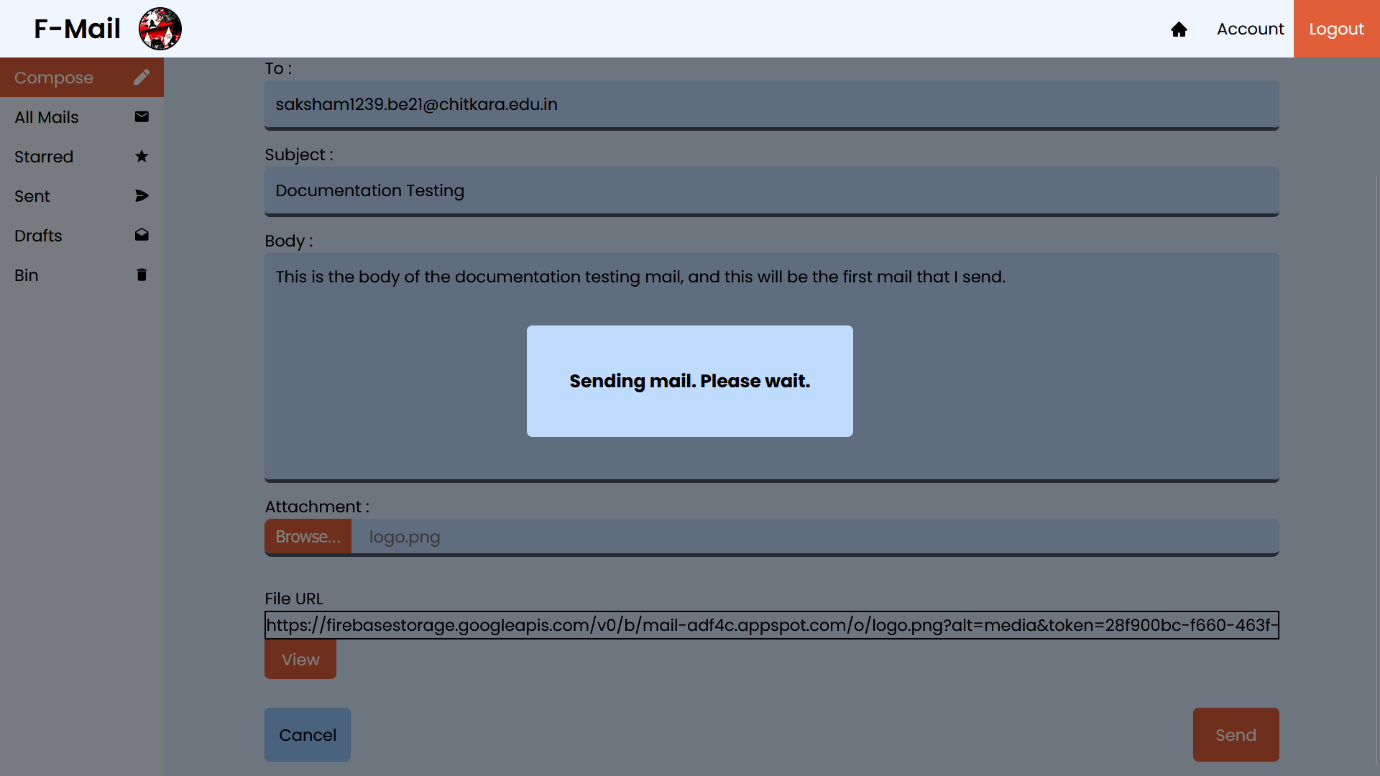


The above image is the screenshot of the “Compose” section. You have to fill the “to” field which is the sender’s mail address. The “subject” and the “body” field are not compulsory to fill. You also have an option to send one attachment with your mail. The attachment works the same as the “Update Profile Pic” page. You browse for the attachment, but this time, you can select any file from your device. It will be uploaded to firebase and then you will receive the file URL. You can even click on the “View” icon to view the uploaded attachment, it will open the attachment in a new window for you to view it easily. Let me show the whole process and send one mail to another user with a random attachment.

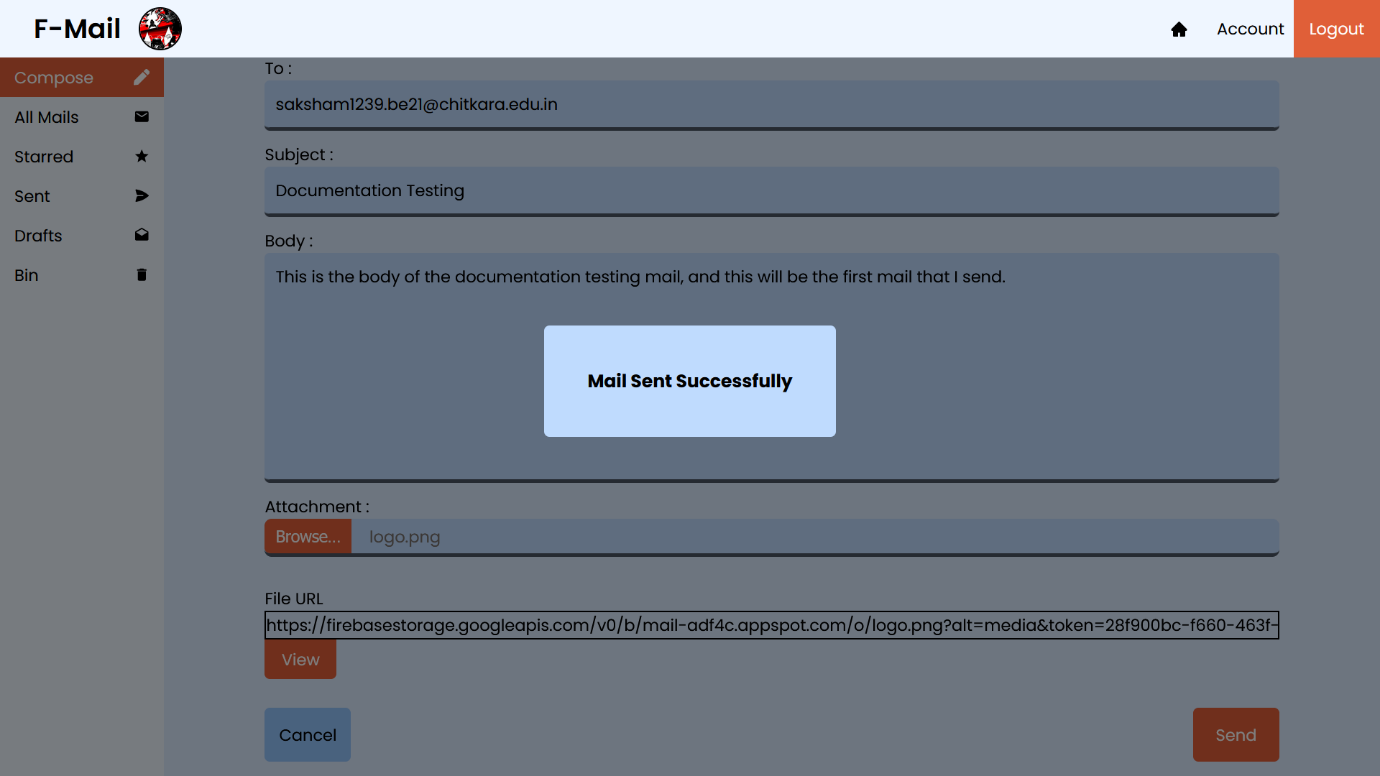


After selecting the attachment, It will take some time to upload it to the firebase, and once the uploading is finished, it will show the URL of the image in the “File URL” section. You can click on the view button to view the attachment on a new window. Once finished uploading, The “File URL” section will be filled and now your mail is ready to be send to the user you entered. Now just simply click on “Send” button to send the mail.



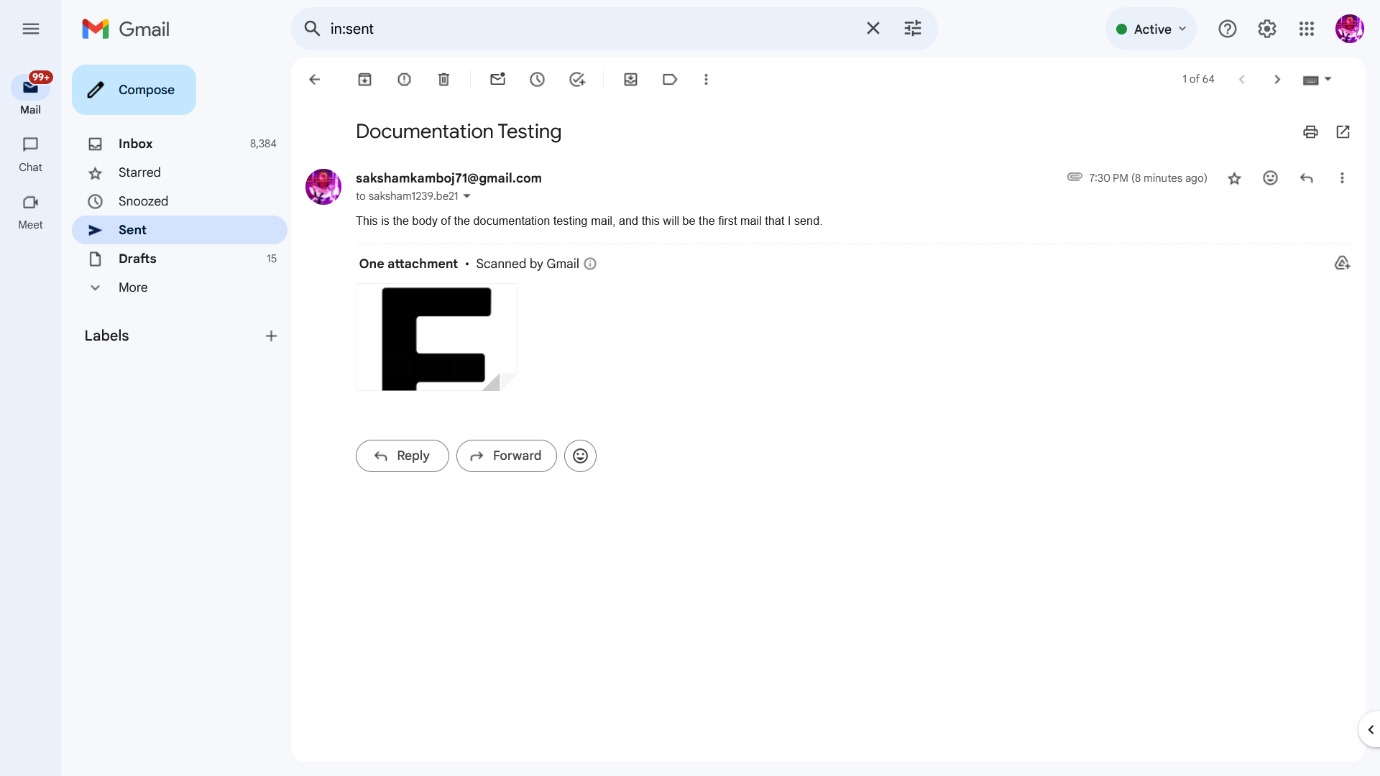


Once you click on “Send”, you will receive a popup saying that is is sending the mail, it takes some time as the mail will be sent through an SMTP server via nodemailer. When the mail is sent, you will be greeted with a message saying “Mail Sent Successfully”. And if there is any error in sending the mail, it will be displayed in the same way as the successful message.

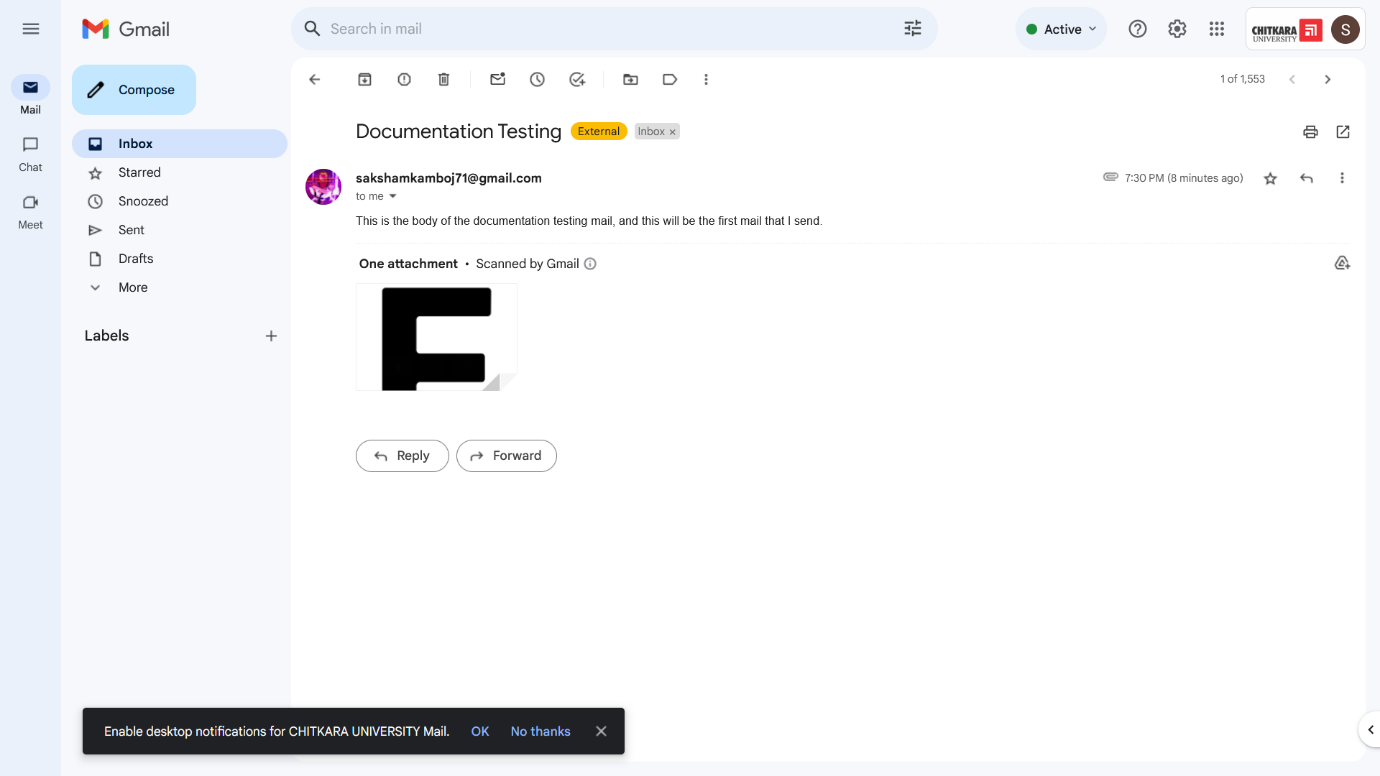




Now as you can see, the “All Mails” section consists of one field, and that is the mail we just sent. But before going any further into the project, let me first show you that the mail is successfully sent from the user, and also received by another user.

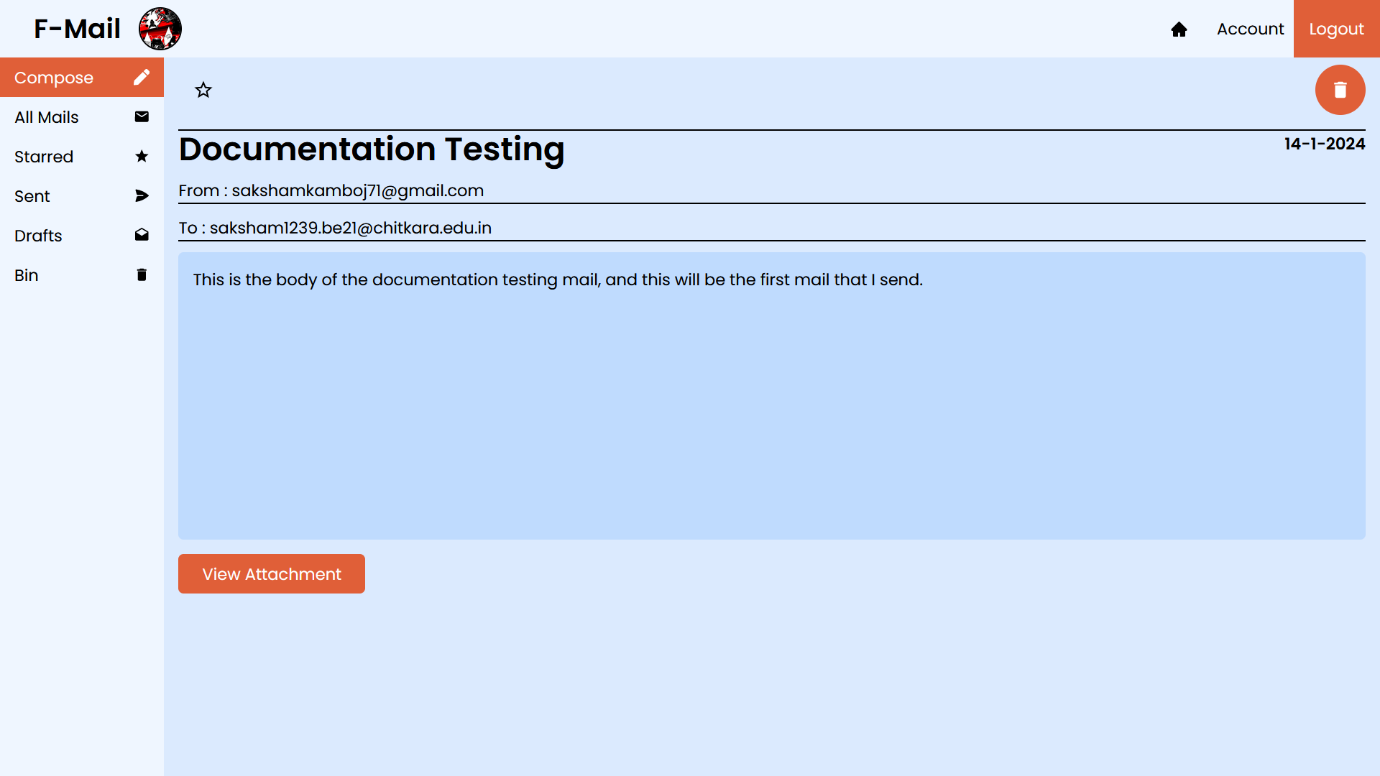


The above image is the Sender’s account on gmail.

The below image is the Receiver’s account on gmail.

This just shows that any mail we write to any user will go through nodemailer and be received by another person, no matter who he/she is and where he/she is.

Moving ahead, The mail that is being displayed in the “All mails” page of my website, is fully functional and clicking on it will navigate me to the “View-Mails” page.



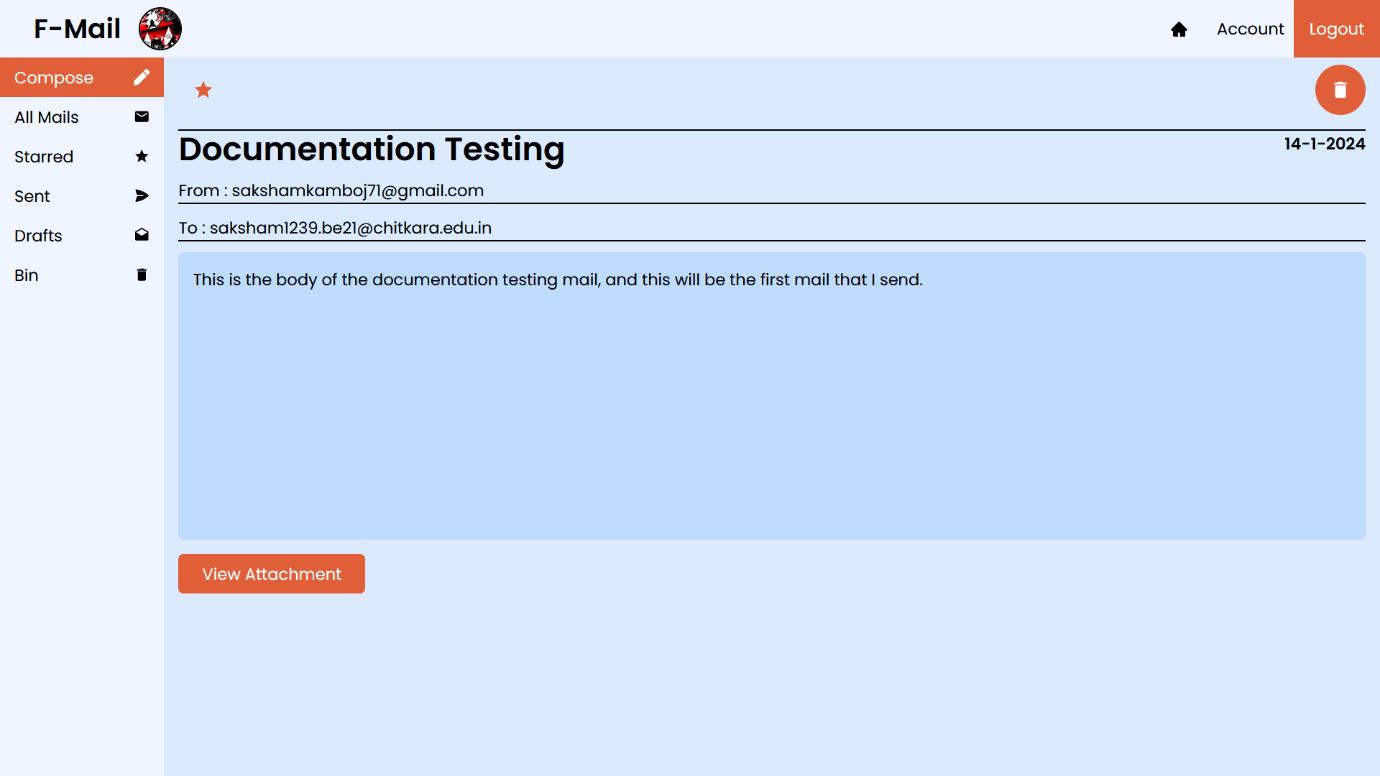
This is the mail we just sent. It shows the Subject of the mail in bold letters, the date on the right side of the subject, The sender’s mail address, and also the receiver’s mail address. Then it shows the body of the mail and also a button which says “View Attachment”. On clicking on that button we can view the attachment we sent in this particular mail.

On the top of the page, we can also see two different icons, one is for starring the mail as important, another is for completely deleting the mail. Deleting any mail will just move the mail into the “bin”. From where we can view it again and manually remove it from the database.

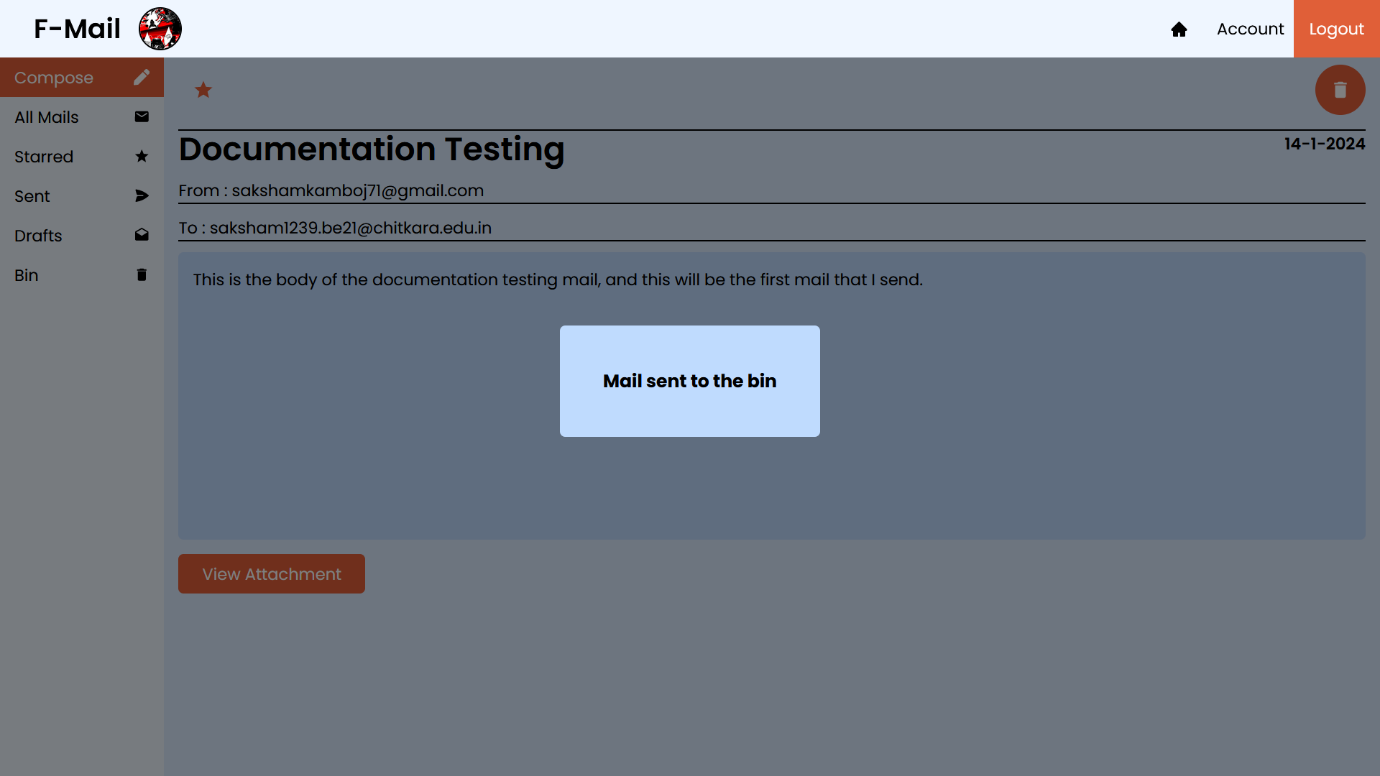
On Starring the mail, the star icon will turn orange, signifying that the mail is starred, and it will also be visible in the “Starred Mails” section.

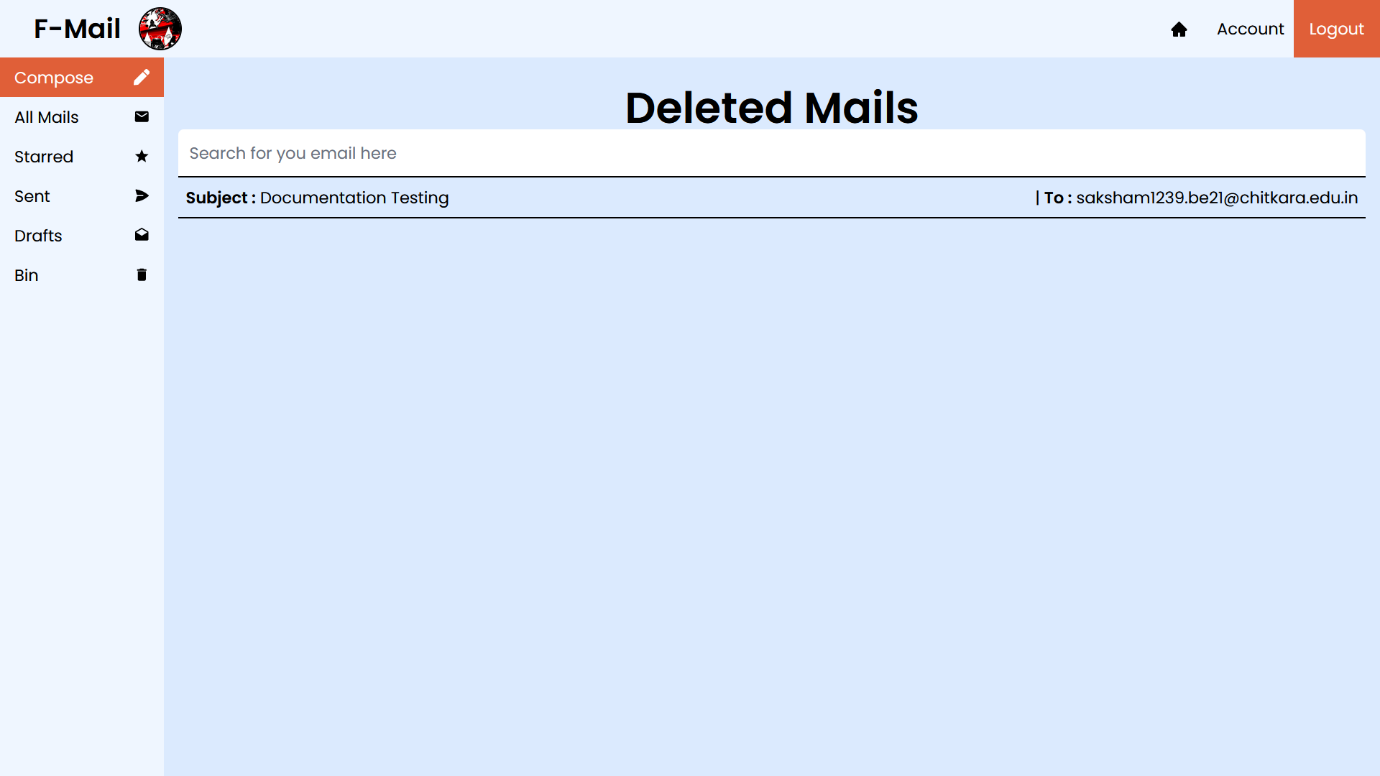
On deleting the mail, it will go into the “Bin”. And all the deleted mails will be visible in the “Bin” section.





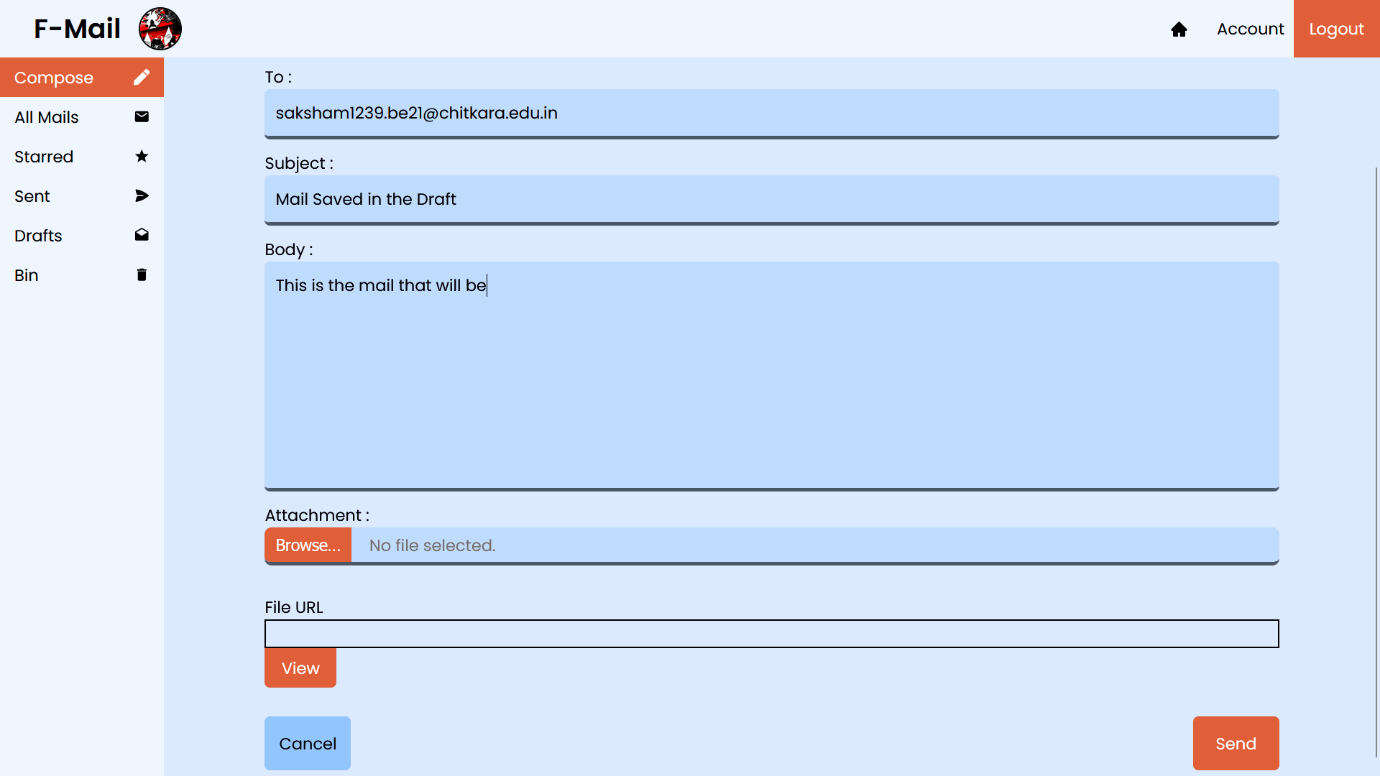
The star in the above mail turned orange, which tells us that this mail is “Starred”. You can also un-star the mail and it will be removed from the “Starred” section.



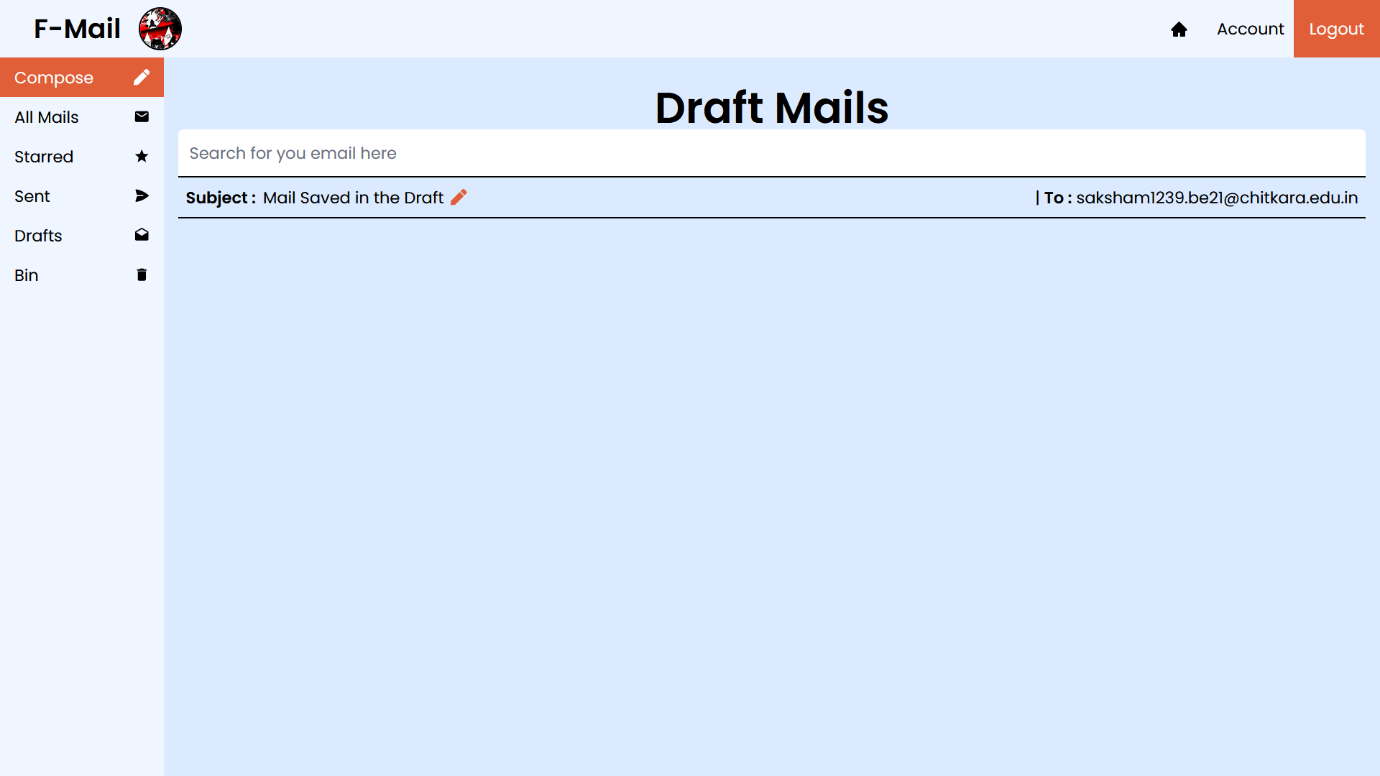


You can view the deleted mails too. And to permanently delete it, you just have to delete it once again. Then it will be removed from the database as well.

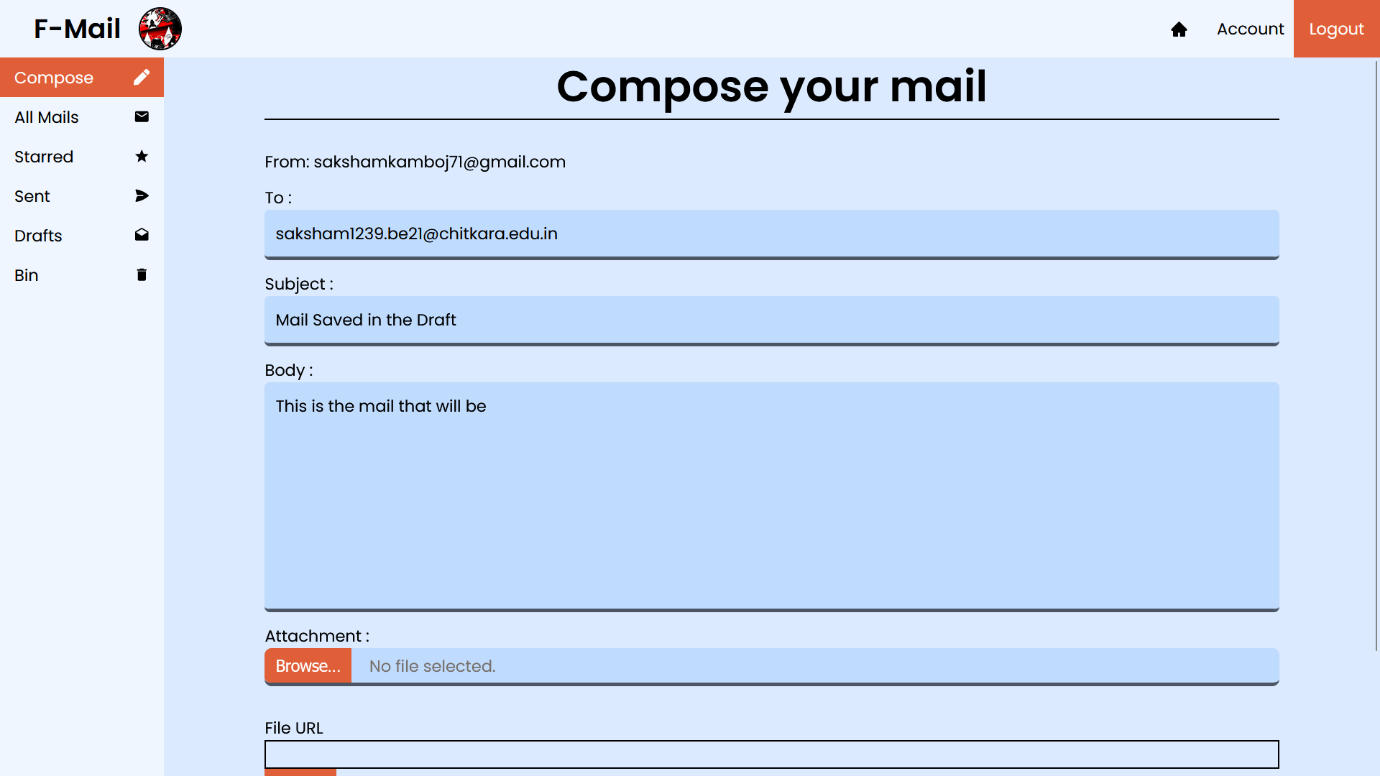
Now, To know about drafts, let us first write another mail, leave it half empty, and then click on cancel.



This is all I am writing and then clicking on the “Cancel” button. It will then navigate us to the “Draft Mails” page.



The mail we wrote is saved in drafts. Now the beautiful thing about this feature is that the mail is not currently sent to the user. Instead, the mail is saved in the database with a tag of “draft”. Now what’s best about this is, that whenever we click on any draft mail, we can continue writing the mail straight from where we left. The drafted mail displays a “Pen” icon.



We are navigated to compose page whenever we click on any draft mail. From here we can continue writing our mail. Cancelling the draft will not save anything. Deleting the draft sends it to the bin and we cannot edit it again. Sending the drafted mail after completing it will send the mail to the receiver’s address and save it in the database.

The mail features are now finished, so let me demonstrate the working of the search bar. For that, first let me send some demo mails in the database.



Now if we want to search for a specific mail, all we have to do it just type in the search-box. It does not matter if there are any uppercases, as it will just convert the whole string in the search-box into lowercase and then by using the filter array method, all the mails which include that particular string are displayed. For example now, let us search for “covid”.

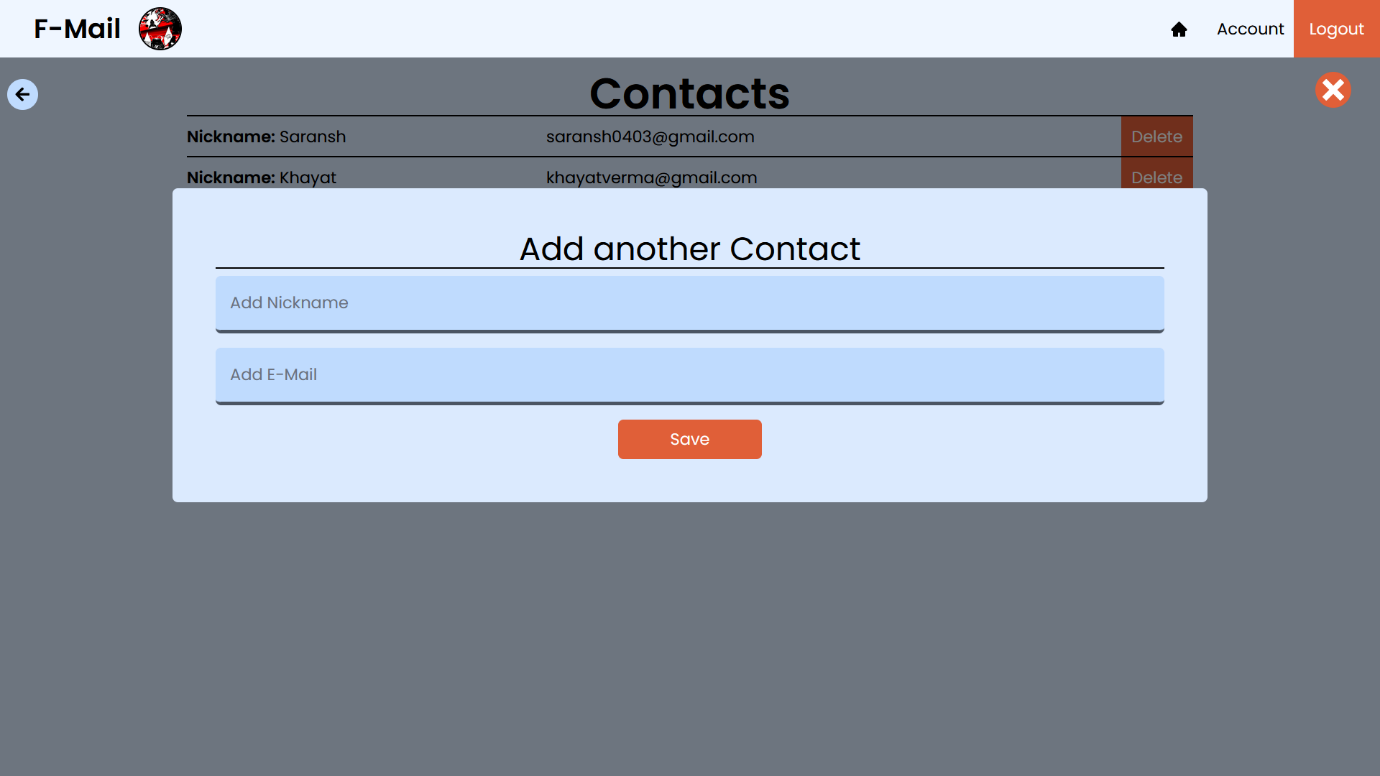


The other mails are not deleted, it is just showing the mails with the word written in the search-box. Once we delete the word in the search-box, all the mails will be displayed again. This functionality remains the same for each and every mail section.

Now let us come back and talk about the Contacts page. For the contacts, I have made a separate collection, where I will store the nicknames, and the corresponding emails. Not much information is needed to be store the contact, as the other details are not very useful for the particular user.



This is the contact page and I have filled some random data into the database. But to add another data in the database, you simply have to click on the **“+”** icon on the top right of the page. Clicking on it will open up a window.



You can just type the nickname and the mail address corresponding to that person and you will then see the contact added on the contacts page. And similarly you can also delete any specific contact you wish for by simply clicking on the delete button on that particular contact.

**User Schema**

import mongoose from "mongoose";

const UserSchema = new mongoose.Schema({

  username: { type: String, required: true },

  email: { type: String, required: true, unique: true },

  password: { type: String, required: true },

  verification: { type: String },

  age: { type: Number },

  phone: { type: Number },

  pic: { type: String },

});

export const UserModel = mongoose.model("users", UserSchema);

**Mail Schema**

const MailSchema = new mongoose.Schema({

  userID: { type: String, required: true },

  to: { type: String, required: true },

  from: { type: String, required: true },

  subject: { type: String },

  text: { type: String },

  starred: { type: Boolean, required: true, default: false },

  bin: { type: Boolean, requried: true, default: false },

  type: { type: String, required: true },

  imgName: { type: String },

  imgUrl: { type: String },

  date: { type: String },

});

**Contact Schema**

const contactSchema = mongoose.Schema({

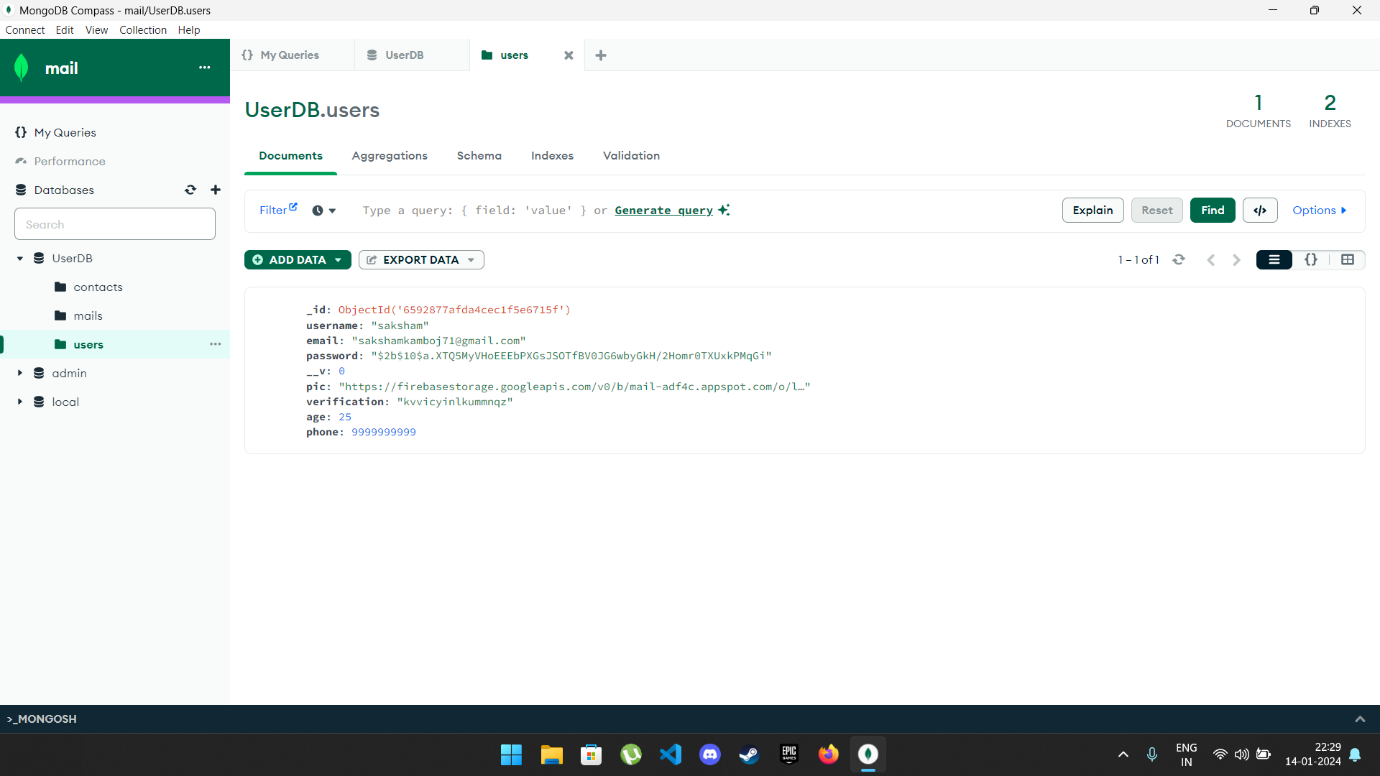
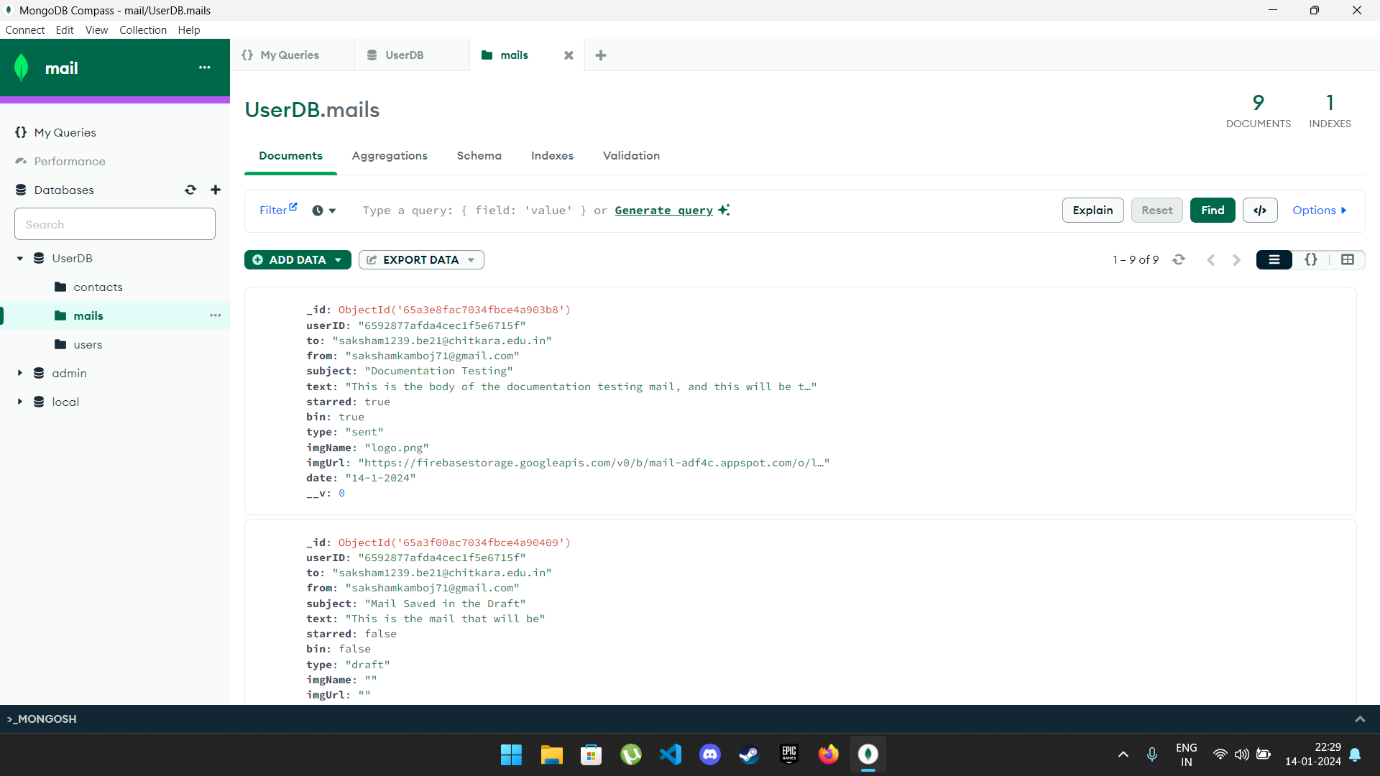
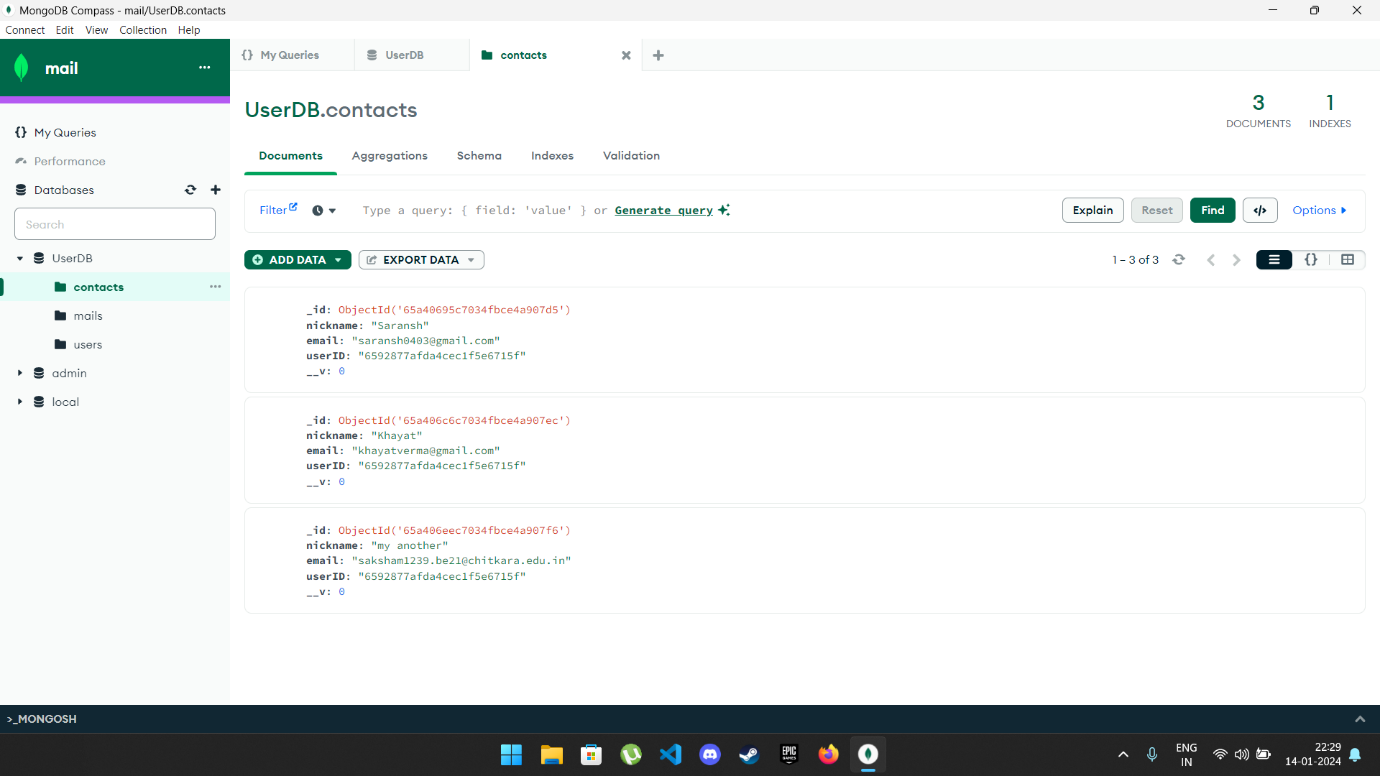
  nickname: { type: String, required: true },

  email: { type: String, required: true },

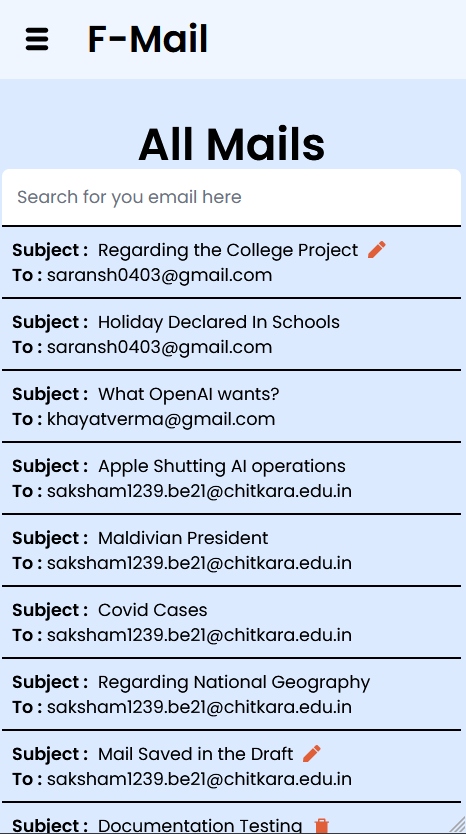
  userID: { type: String, required: true },

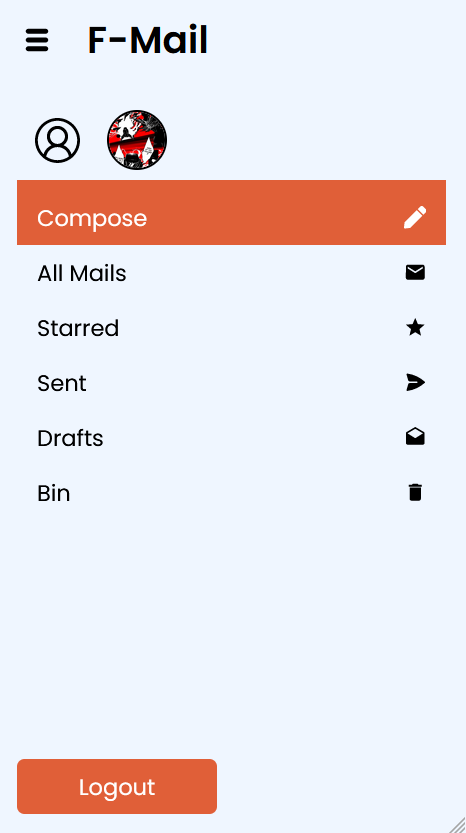
});

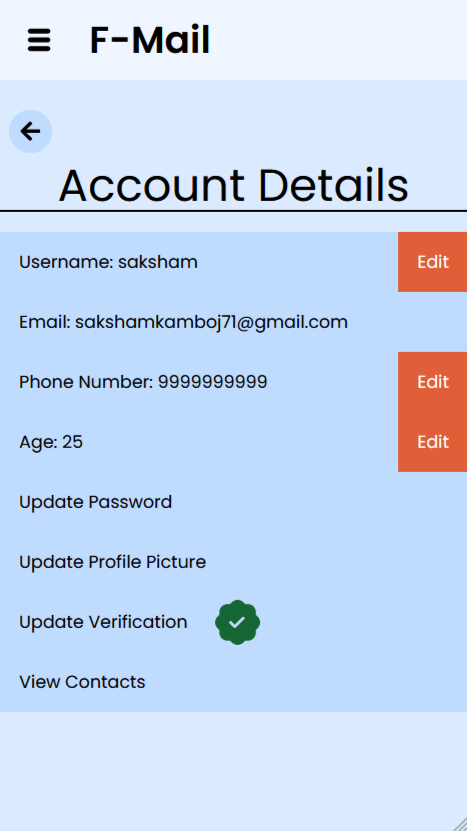
These below are my 3 different collections.

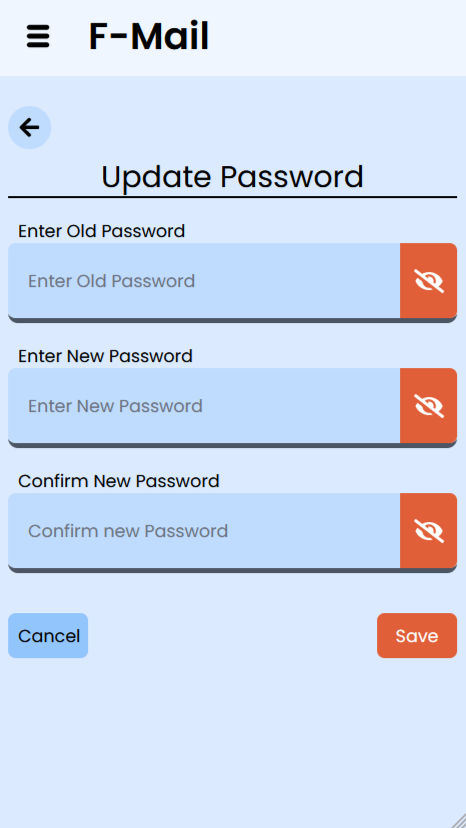


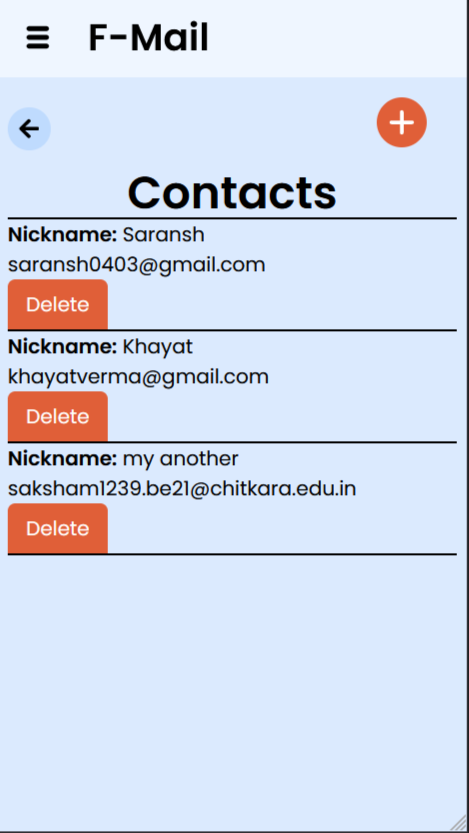
**RESPONSIVE DESIGN SNAPSHOTS**

****

****

****

****

****

****

**Future Scope**

* **Improving the Contact page:** Clicking on any particular contact will take you to the draft and automatically fill the “to” field.
* **Improving the animations:** The page currently lacks many different animations. Another way to improve the visual appeal of the page is by improving the animations. Animations
* **Two-Factor Authentication (2FA):** Strengthen user security by integrating 2FA for authentication.
* **More attachments:** Currently we can only add one attachment to the mail, but in future more attachments can be added to the mail.
* **Secure file handling:** The files currently are stored in firebase. It can be improved and a more secure way can be added to store the files.
* **Multi-delete mails:** A feature of selecting multiple mails and deleting them at once can also be added.
* **Receive mails:** Currently, nodemailer does not support receiving mails, but in future another package can be used for receiving the mails and displaying them on this platform.
* **Themes:** Different themes can also be added to the pages of the websites which can be changed according to the user’s preferences. This will improve the visual appeal of the website.
* **Notifications:** Currently there was not that need, but as an extra feature, notifications can also be added on sending mails.
* **Security:** The overall security of the website can also be improved. No matter how big a company is, there is always a risk of cyber-attacks, so security can always be improved.