**Task 6.3HD: Custom Web Application**

**Github Repository: https://github.com/HaileInnoTech/EmailComposeWeb**

**Website url :** **https://emailcompose.hailecv.site/**

*This is a web application that use the SMTP server of google for user to create a marketing email*

*to many clients. The application is implemented by VueJS, boostrap and PHP for the backend process.*

Goals:

* The web application is built using the Vue CLI and utilizes VueJS and Bootstrap for the user interface.
* Users can log in using their Facebook, Google, or registered account credentials, and their information will be securely stored on Firebase using the Authentication feature.
* Users can compose and send emails to multiple clients either using traditional methods or by using HTML and CSS.
* Users can upload images and insert them into their emails to create visually engaging content.
* Users can logout when they are already login
* The application includes a preview button that allows users to preview their email content before sending it to clients.
* When the user clicks the send button, the application uses Axios to send a POST request to the server to initiate the mailing process.
* The server is implemented using PHP scripts, which receive the data from the form, process it, and send the email to the clients.
* Overall, the web application provides users with a user-friendly and secure platform for creating and sending marketing emails to multiple clients, using modern technologies such as VueJS, Bootstrap, Firebase, and Axios.

**Tools and Resources Used**

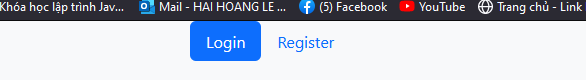
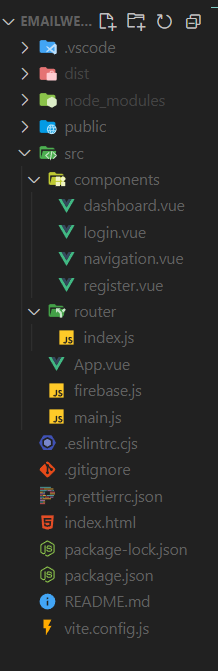
* VueJS CLI with Eslint, Prettier and Vue Router
* Axios library
* Boostrap framework
* PHP mailer library
* Firebase
* Xampp (localhost)
* IDE: Visual Studio Code

**Knowledge Gaps and Solutions**

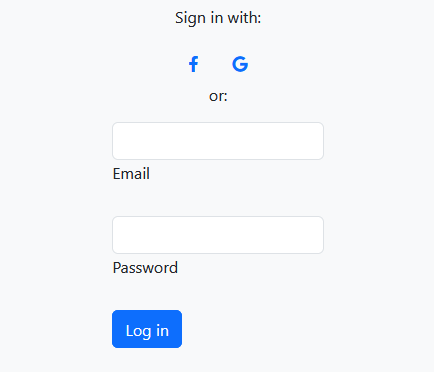
**Gap1: Web Application structure**

This app is use Vuejs to implement a Single page application so we will have many components and use Vue router for easy control the website.

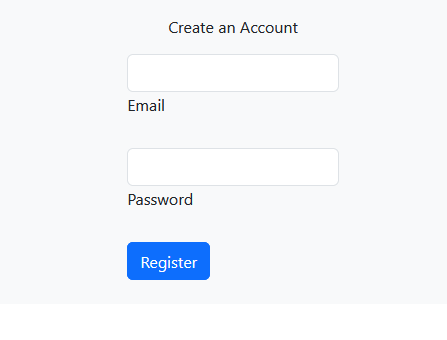
Project structure Navigation bar

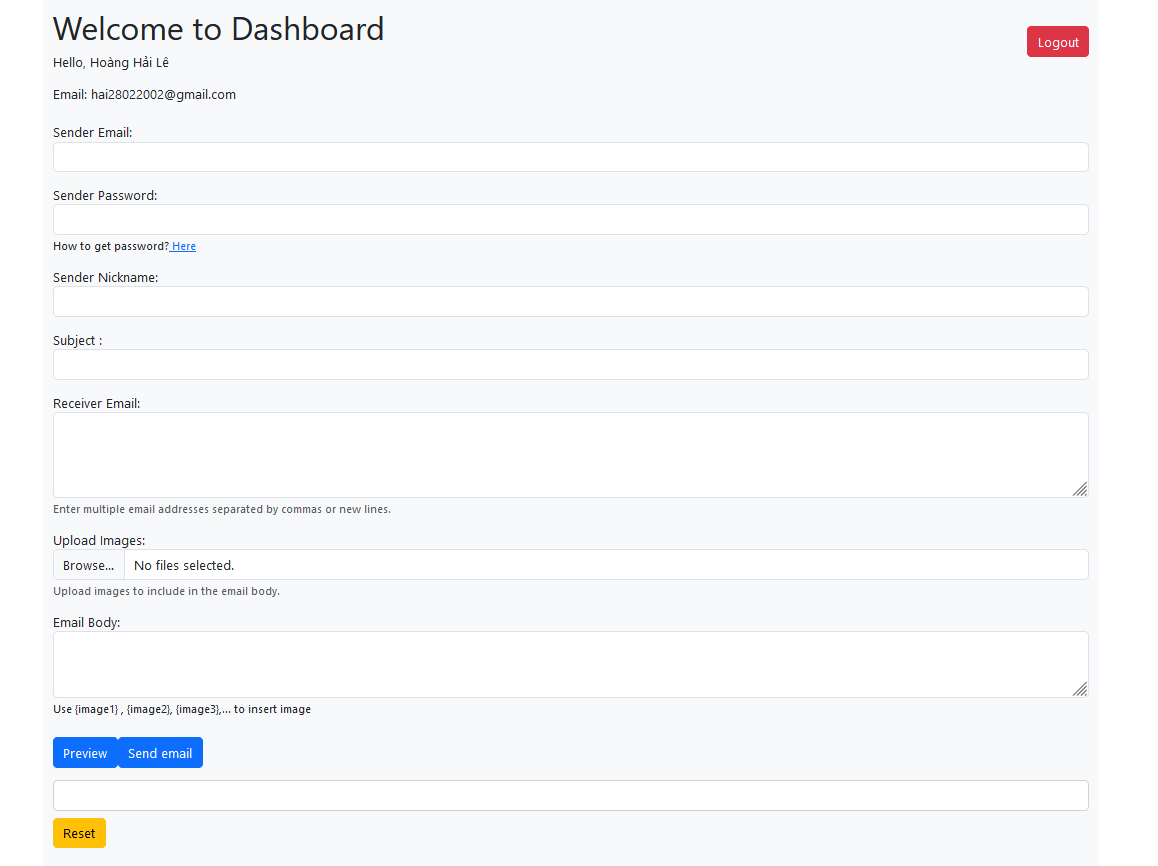
****

Login Page

****

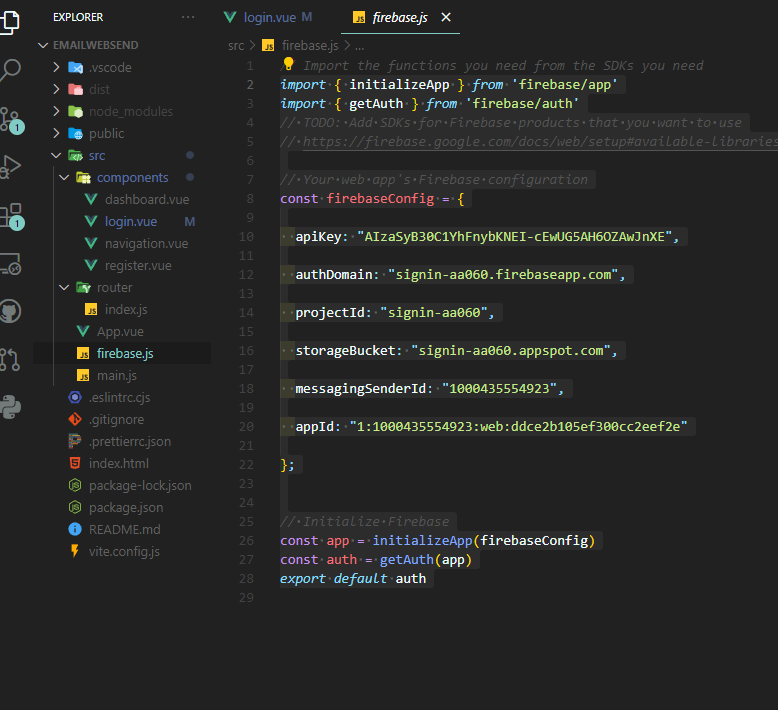
Register Page

****

**** Dashboard

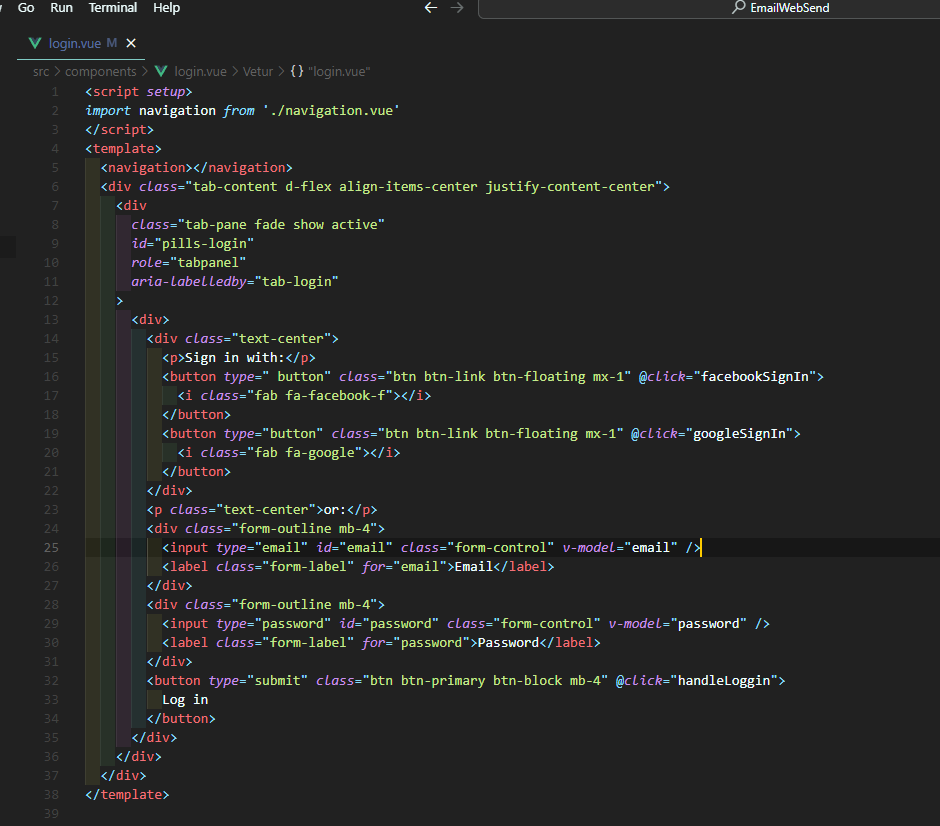
**Gap 2: Firebase**

In order to use Firebase, we had to create a firebase.js file and import the firebaseconfig provide by firebase. Then we have to initializes the Firebase app with the configuration object using the 'initializeApp' function, and assigns the resulting app instance to the 'app' constant. The 'getAuth' function is then called with the 'app' instance as a parameter to retrieve an instance of the Firebase Authentication service, and this instance is assigned to the 'auth' constant. Finally, the 'auth' constant is exported as the default export of the module, which means it can be imported into other modules as needed to access the Firebase Authentication service.

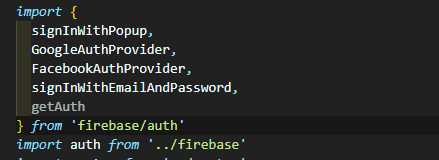
****

**Gap 3: Login Page**

In this assignment, we developed a better authentication system for the user, now with Firebase Authentication feature, we can implemented more choices for user to login the application, in this case, user can login with google or facebook, or normal way.



The login page will use the vue router to render the navigation.vue to implement the navigation bar. With the vue router we can make it more reusable component. There are two icons include which are Facebook icon, and Google icon. We used the @click for both icon so user can click it. The facebookSignIn or google SignIn methods will be executed.



To use the login with google or Facebook we can import the methods from the firebase and create of that object to pass in the auth object imported from the firebase.js. Then use the method to popup a new window for user to login based on their choices.

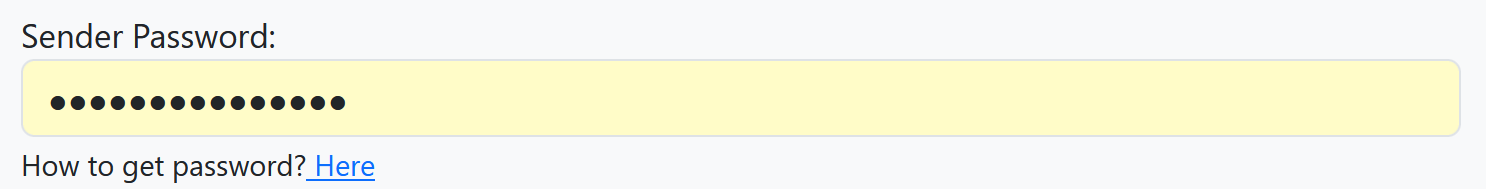
**Gap 4: Register Page**

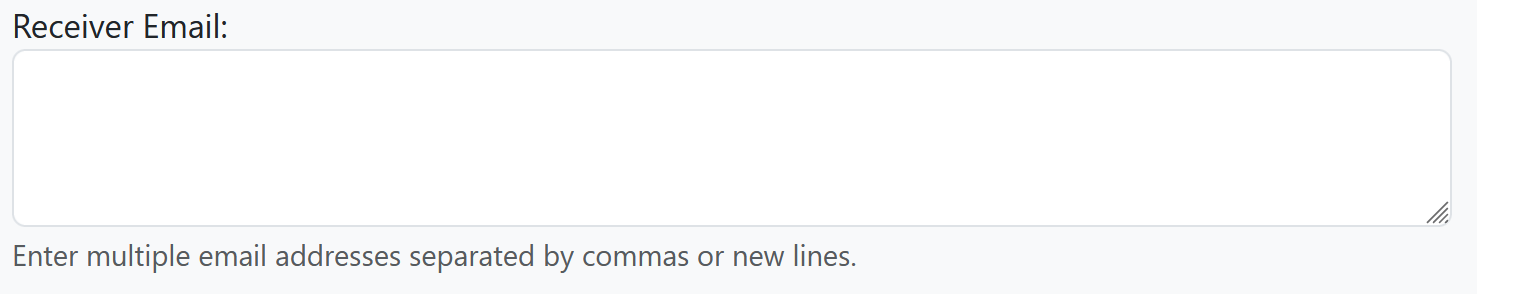
****

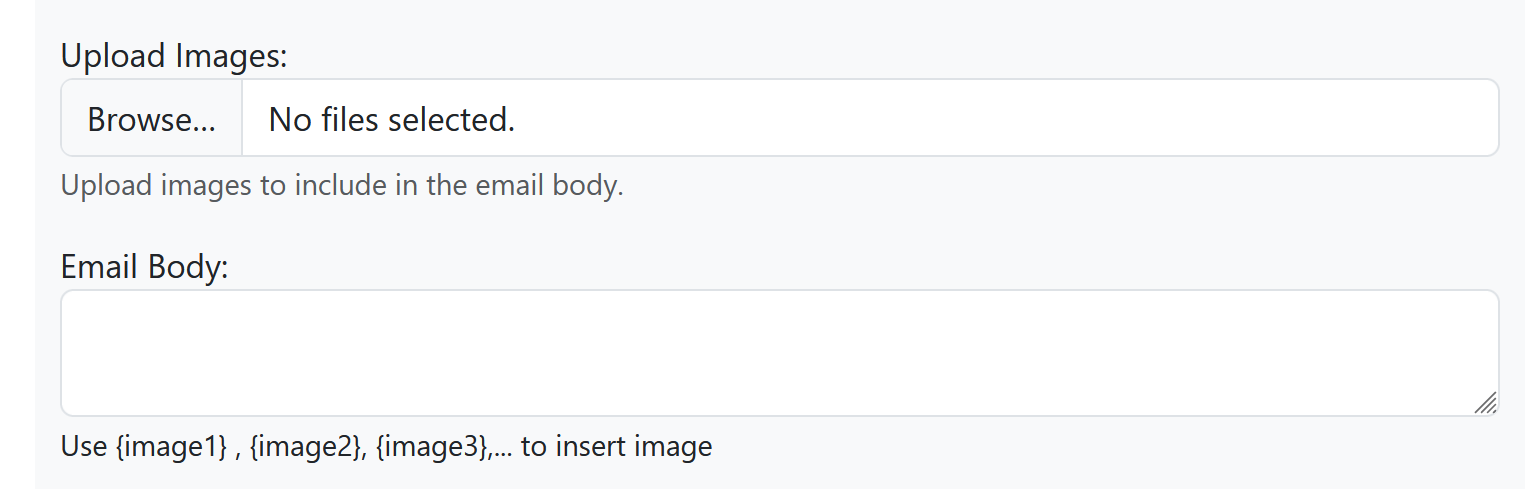
Using the same concept, for the register page we included the form as will as the method createUserWithEmailAndPassword() that receives the email as well as password for creating an account 

**Gap 5: Dashboard**

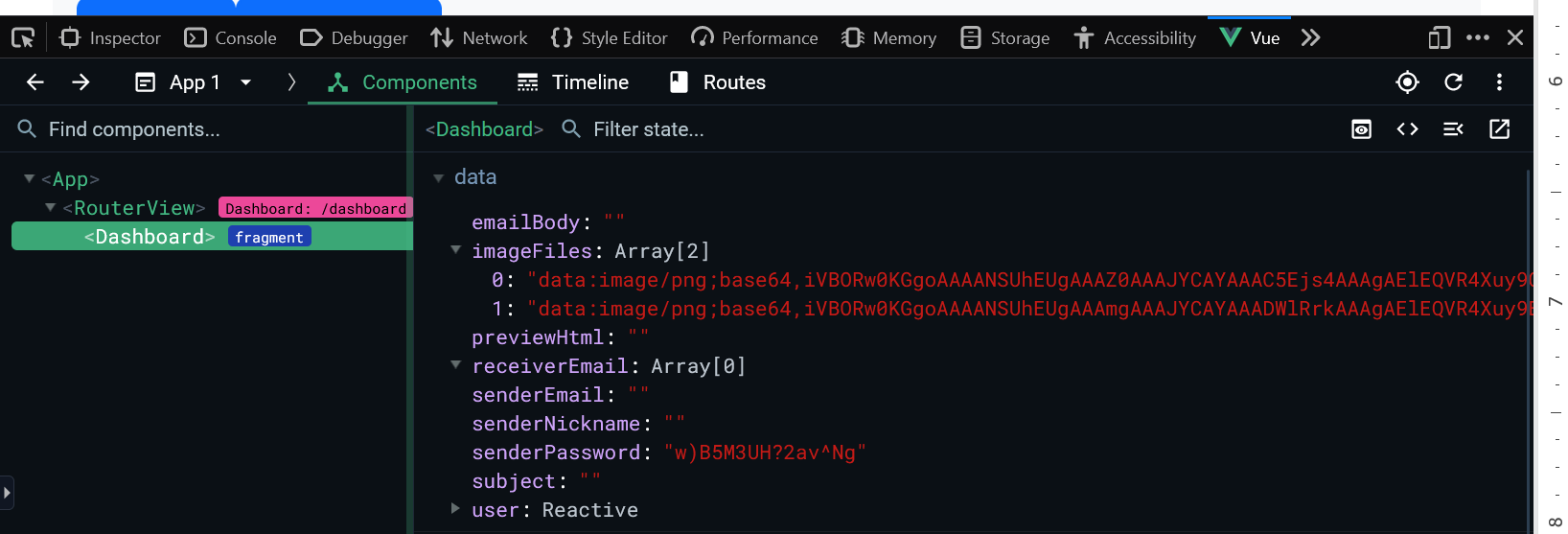
In order to send an Email we need a form that will send post request to a php script to process the data. We will need the senderEmail, senderPassword (this is the secure password for gmail app so user can send an Email form third party application) as required. We also include an hyperlink so user can see the instruction to find it



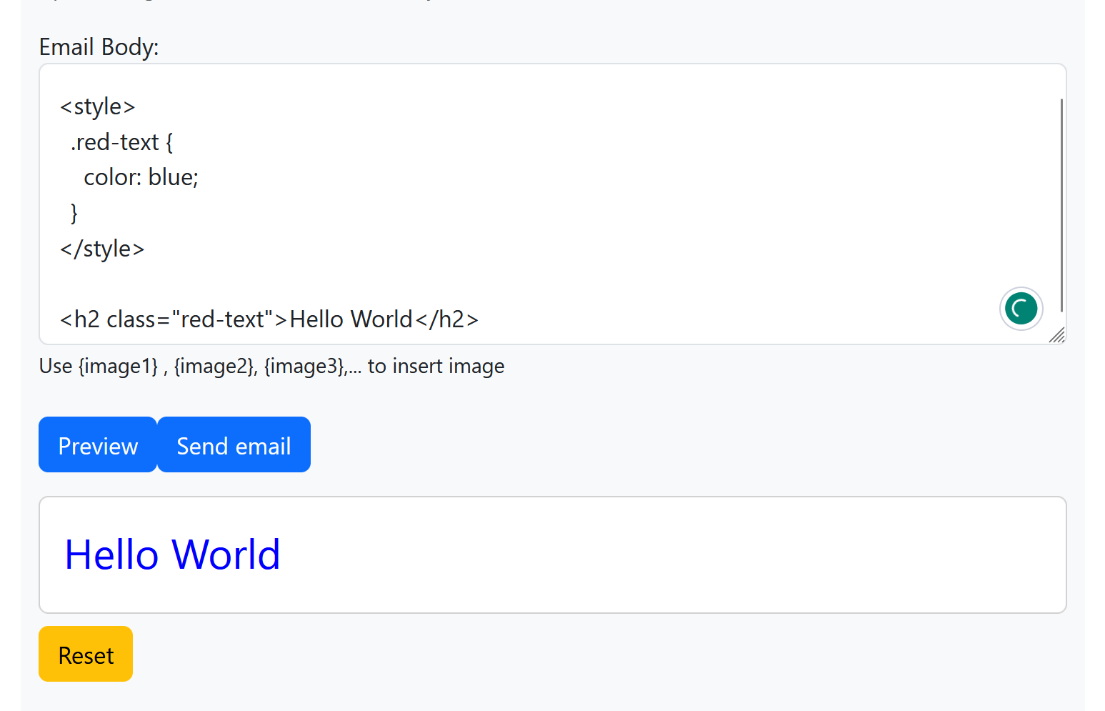
Gmail service also support to send your email with a nickname, user an leave it as a blank or put in an nickname for the mail. The subject email and receiver email as an array. User can input one or many email separate by a comma. The receiverEmail will be stored in an array.

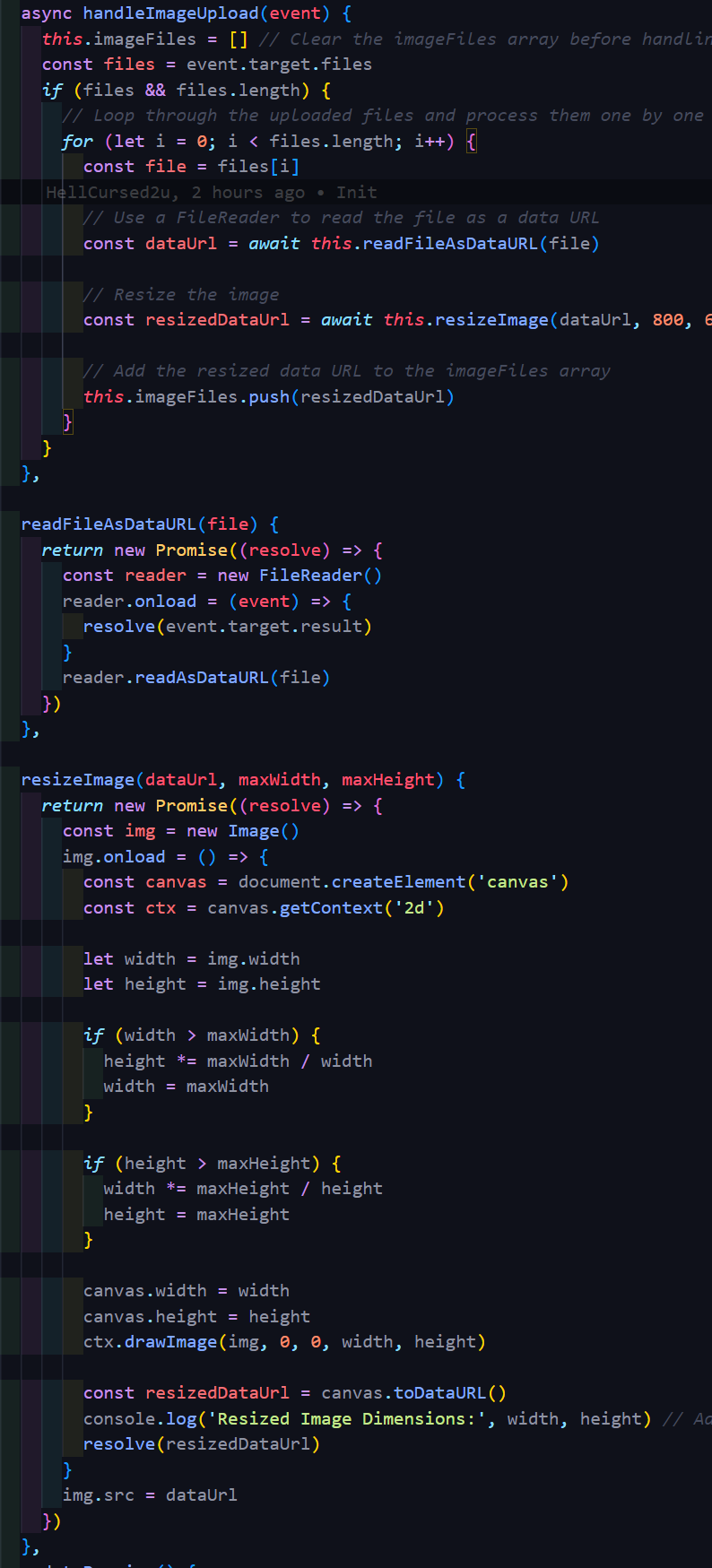


As a commercial marketing email, images should be embedded in the email body, users and import one or more images in the email body. The images will be stored in an array under base64.



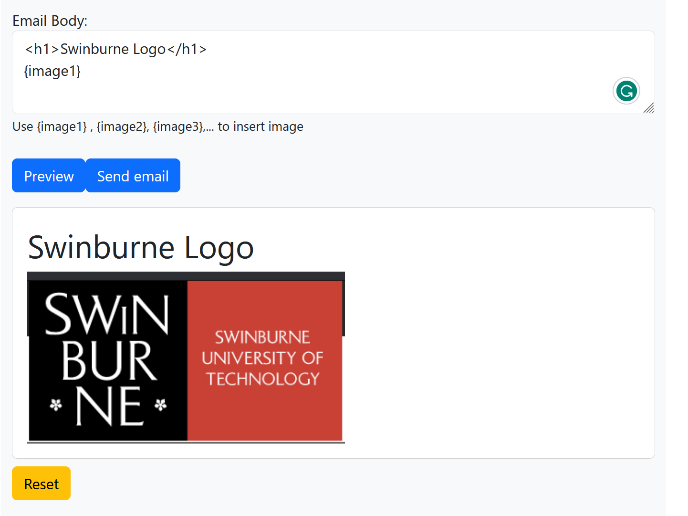
For the emailBody, use can compose with normal text or an coding email for a better design. A preview button is set to @click to pass in the emailBody to the v-html directive which allows you to bind a data property or an expression that contains HTML markup to an element and have the HTML rendered as actual content instead of treating it as plain text.

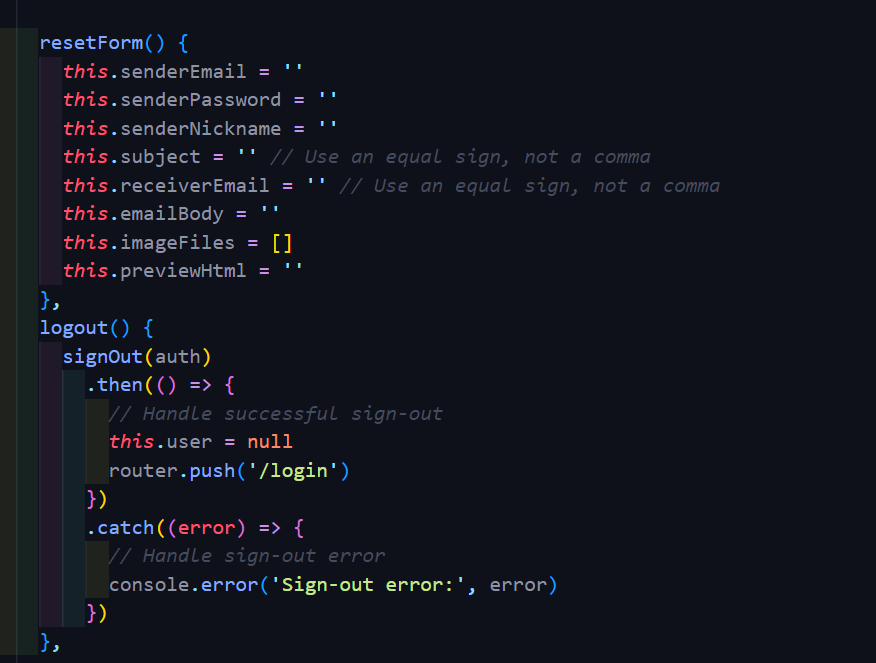




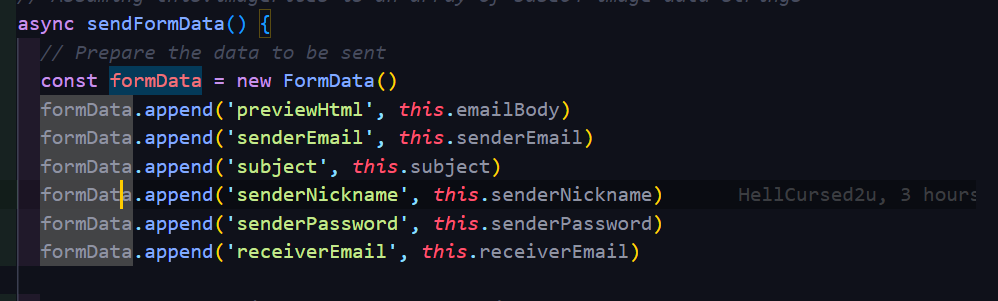
When an image file is selected, the handleImageUpload function is triggered. It clears the imageFiles array and sequentially processes each uploaded file. The readFileAsDataURL function reads the file as a data URL using the FileReader API. The resizeImage function resizes the image by creating a canvas element, adjusting the dimensions, and converting it back to a data URL. The updatePreview function updates the email body preview by replacing {imageX} placeholders with <img> tags containing the uploaded images. Remaining placeholders without uploaded images are removed. The resulting HTML is stored in the previewHtml variable. Overall, this code enables image uploads, resizing, and dynamic preview updates in the email composition feature.



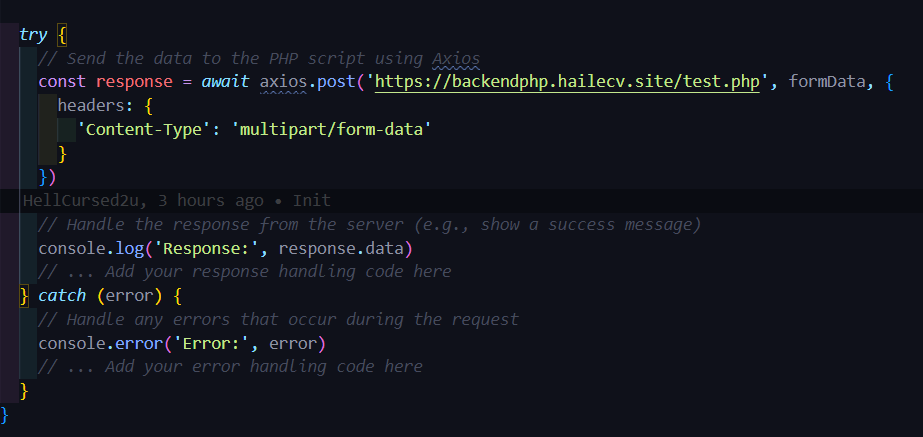




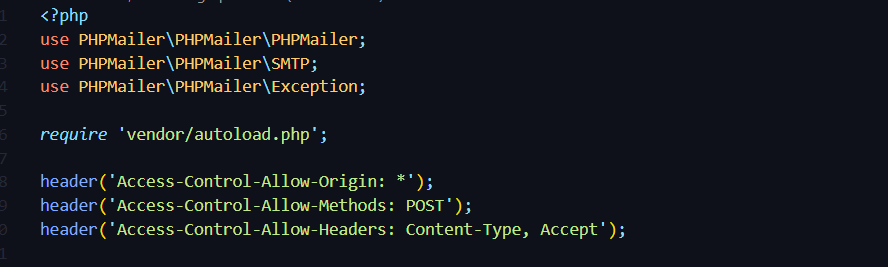
We added the @click event to the reset button which enables user to rest everything and a logout button that call the signOut method of auth object imported from the firbase.js. It sets the user to null and route to the login page.

To send all the information to backend server side. We create a formData that store all form input and use Axios to post request to the API to send the gmail.

For the images we will change each images in the imagesFiles array which are in base64 to blob (which is binary image data). It continues to split into smaller chunks and converted into unit8array which is a typed array in JavaScript that represents an array of 8-bit unsigned integers. It is commonly used to work with binary data, such as image data. The unit8array creates a blob object. Finally, the blob is post appended to the formData in the images array with the name in the order of image{index+1}.png

When we all have the necessary data, we use the axios library to post request to the server side with the header type of form data which includes the form data as well as the file upload script.

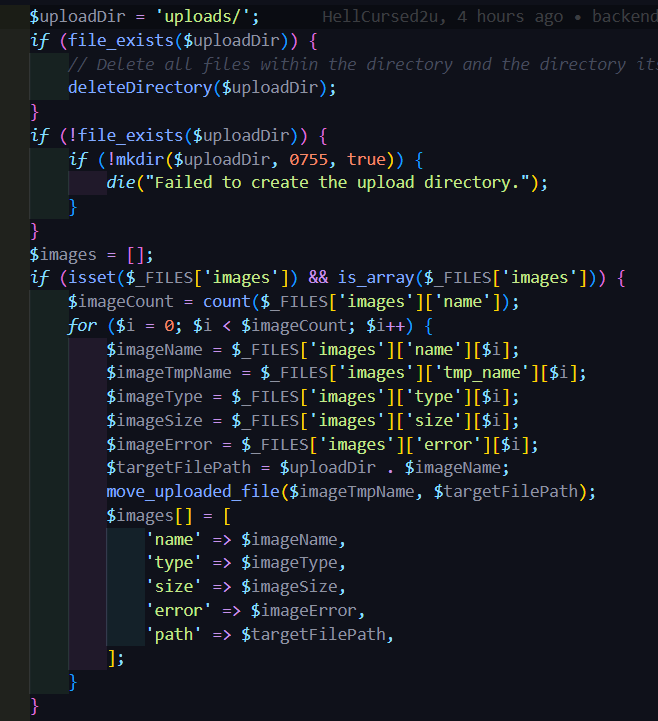
**Gap 6: Server side PHP script**

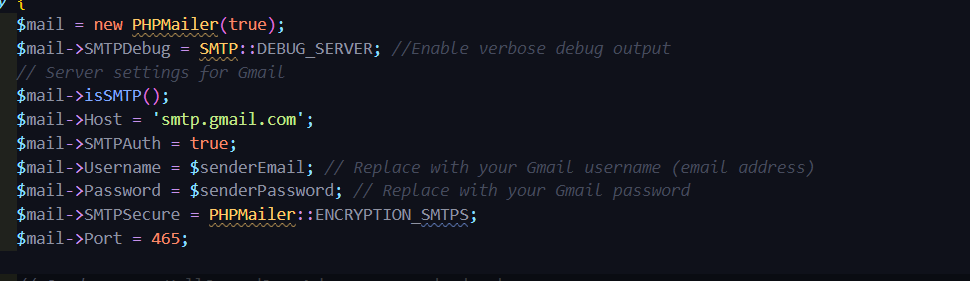
To perform the ability to send email, we will use the outsource library PHPMailer which allow to perform email send.

As a server which has a responsible for receiving Post request. We need to add some header that will allow the request into the server in this case, any domains can only do the post request to the server which has the head of Content-Type and Accept. If there is no header, it would cause the CORS errors

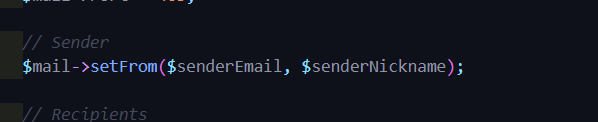


We will set the condition based on the global $\_SERVER variable. If there is a post request to the sever we will set each variables to the $\_POST with the coordinating name. We will create a folder to store the uploaded image if is not exist with a full permission. Then save each image in the images array with the move\_uploaded\_file() methods with the name and the paths.

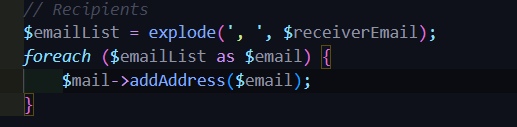


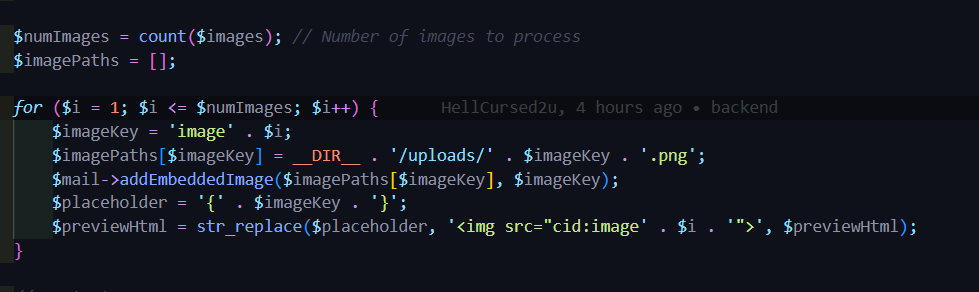


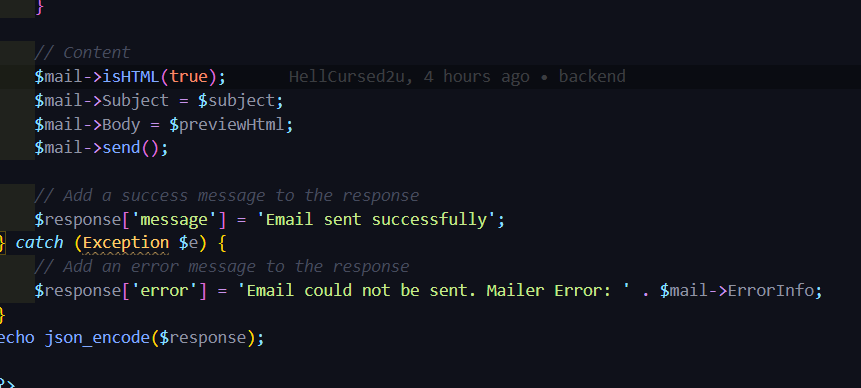
Now we can ready to set up for sending email. The host is through the smtp.gmail.com, the port is 465 with the username and password are from the user input.



The setFrom method can be used to set the nick name for the incoming email

From the array of emailList, we explode the comma to take out each email and use a foreach function to add the email address.



To send the email with embedded images inside, we will take the previewHTML that contains the {image1} and replace with the html of img format, additionally, we will use the addEmbeddedImage() function to embed the image in the email. Finally we can send the email with the formatted previewHTML