**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**

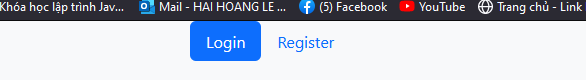
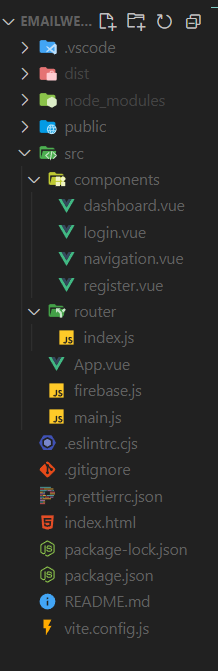
* VueJS CLI with Eslint, Prettier and Vue Router
* Axios library to make http request
* Boostrap framework
* PHP and php mailer library and xampp
* Firebase
* IDE: Visual Studio Code

**Knowledge Gaps and Solutions**

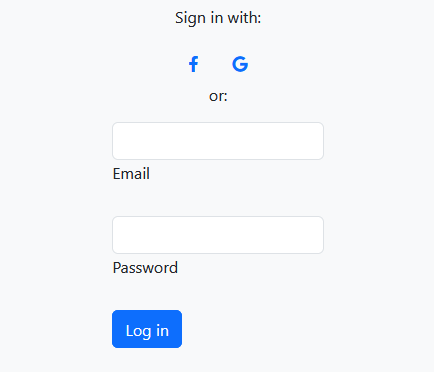
Gap1: Web Application structure

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.

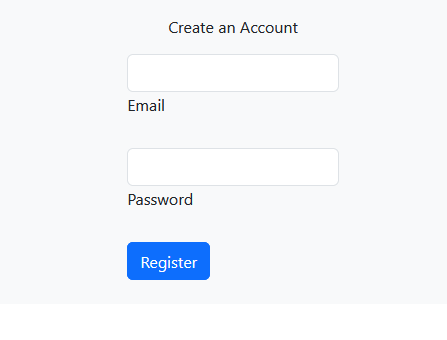
Project structure Navigation bar

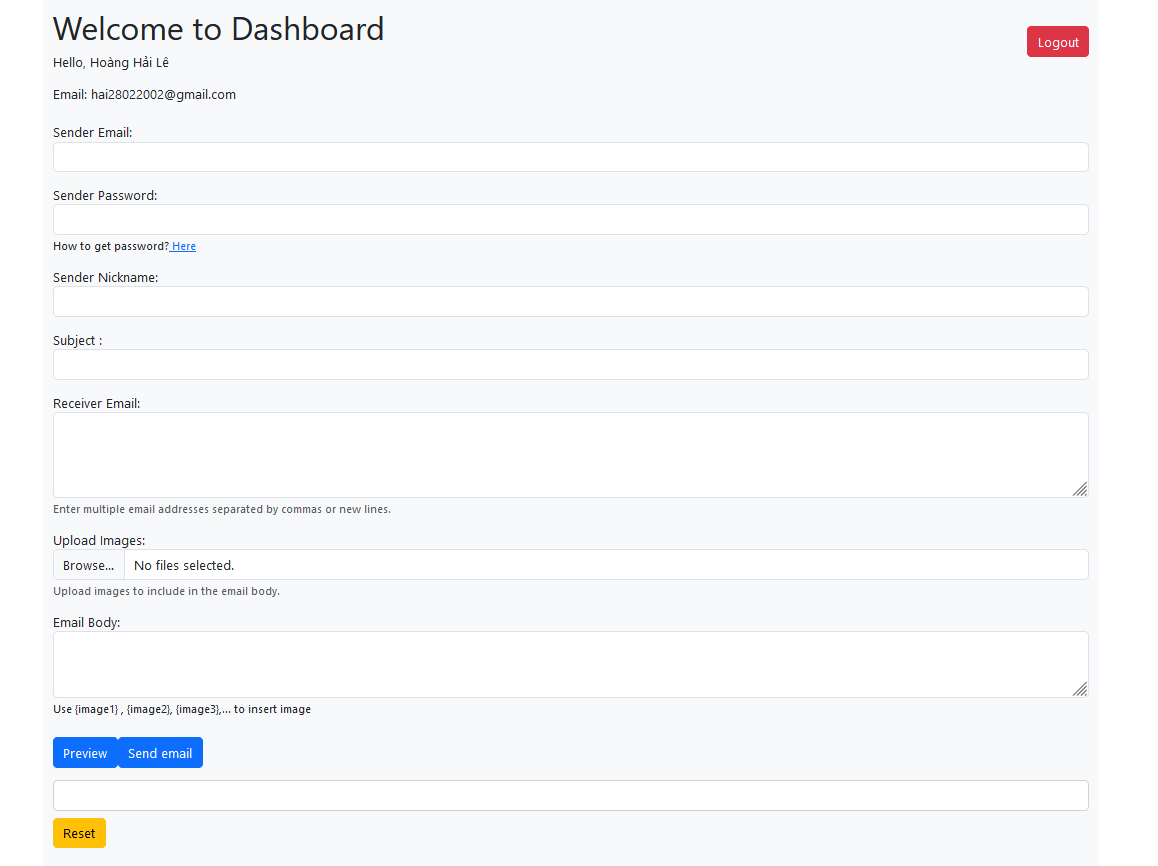
****

Login Page

****

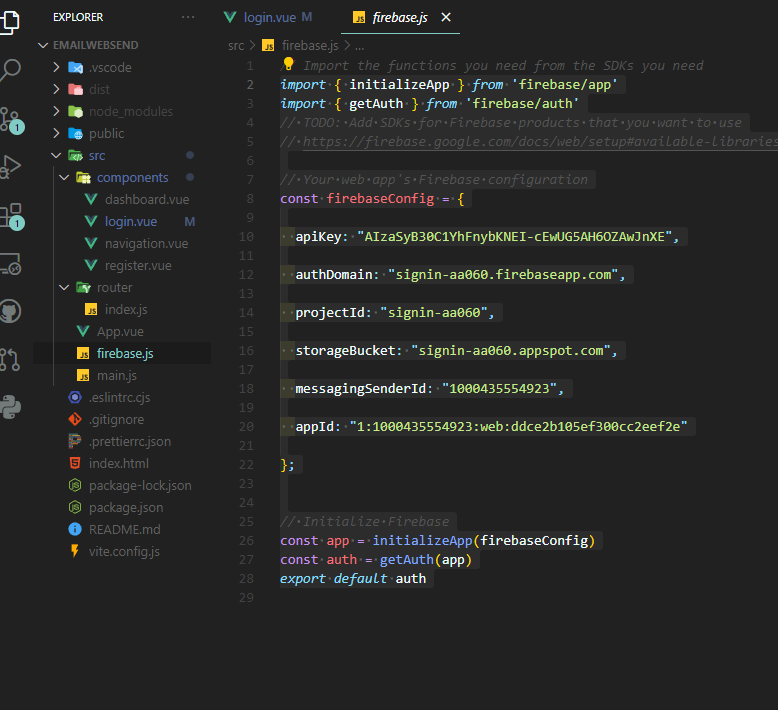
Register Page

****

** Dashboard**

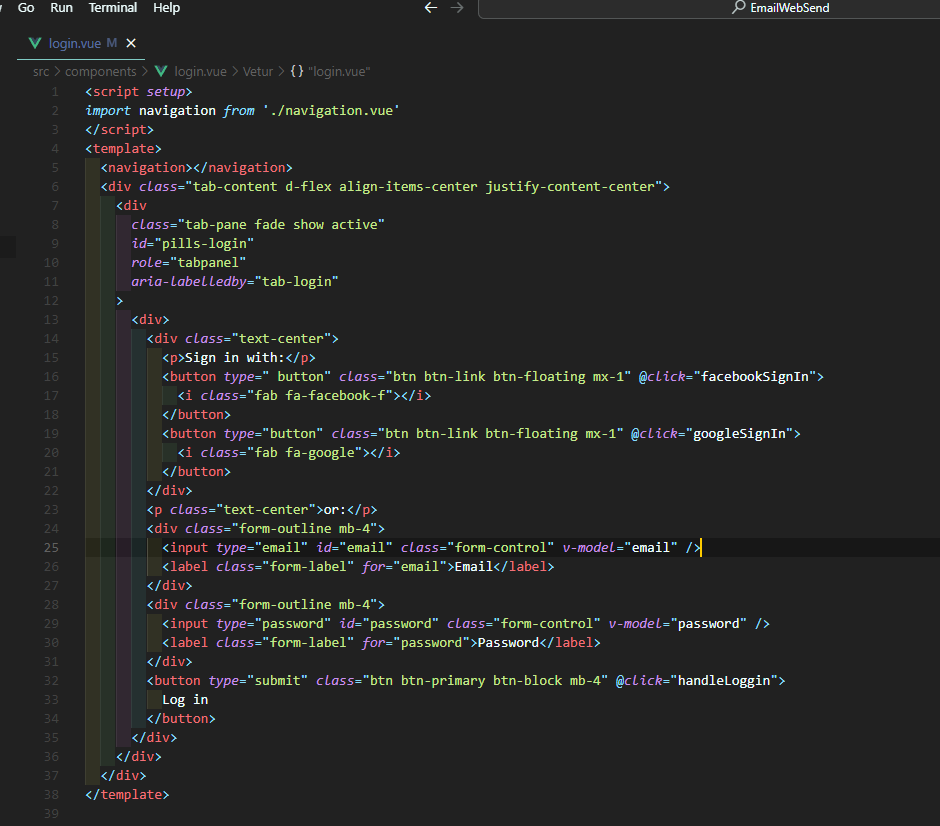
**Gap 2: Firebase**

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.

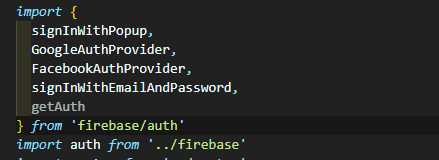
****

**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.



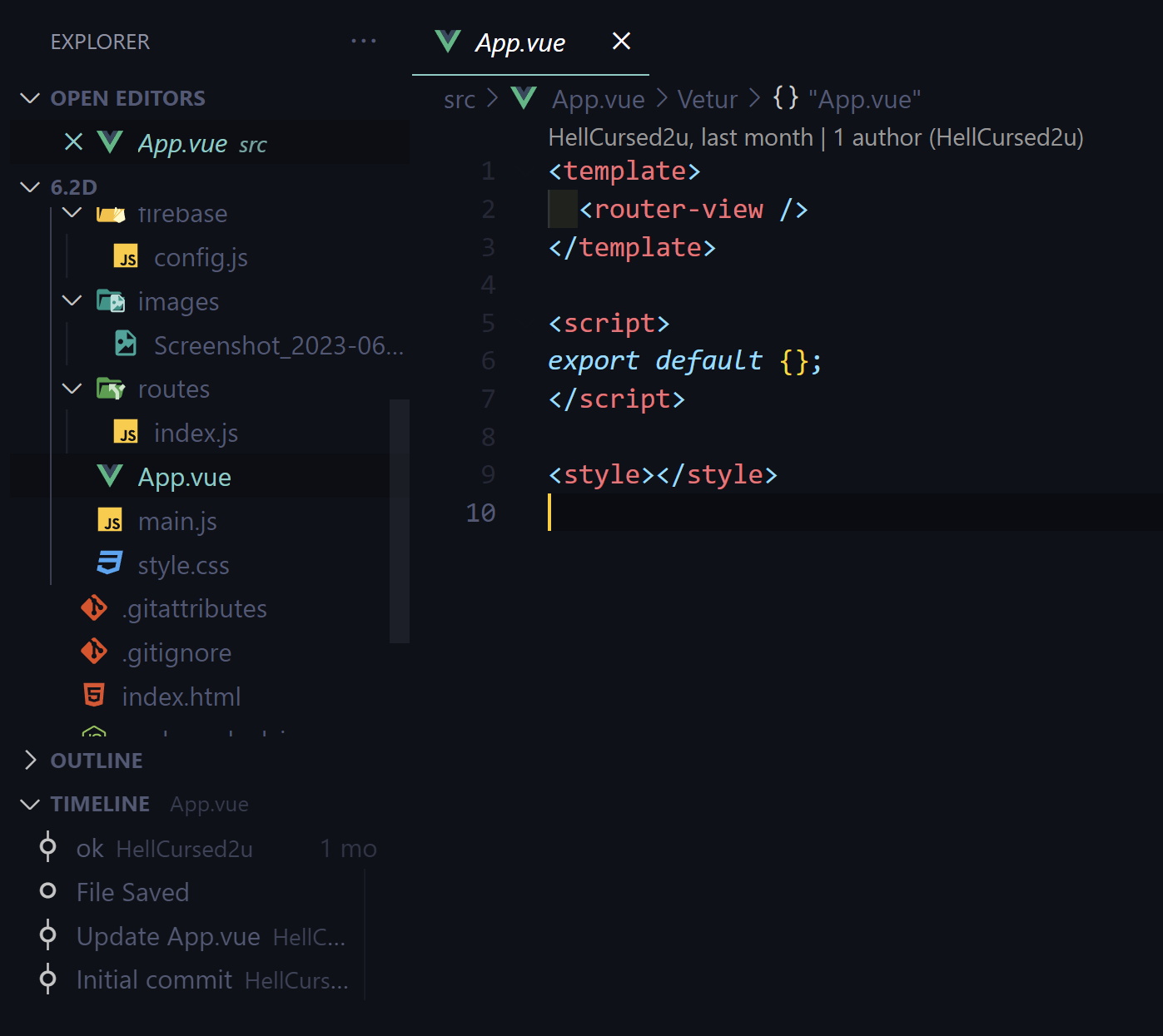
The login page will use the vue router to render the navigation.vue to implement the navigation bar. With the vue router we can make it more reusable component. There are two icons include which are Facebook icon, and Google icon. We used the @click for both icon so user can click it. The facebookSignIn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



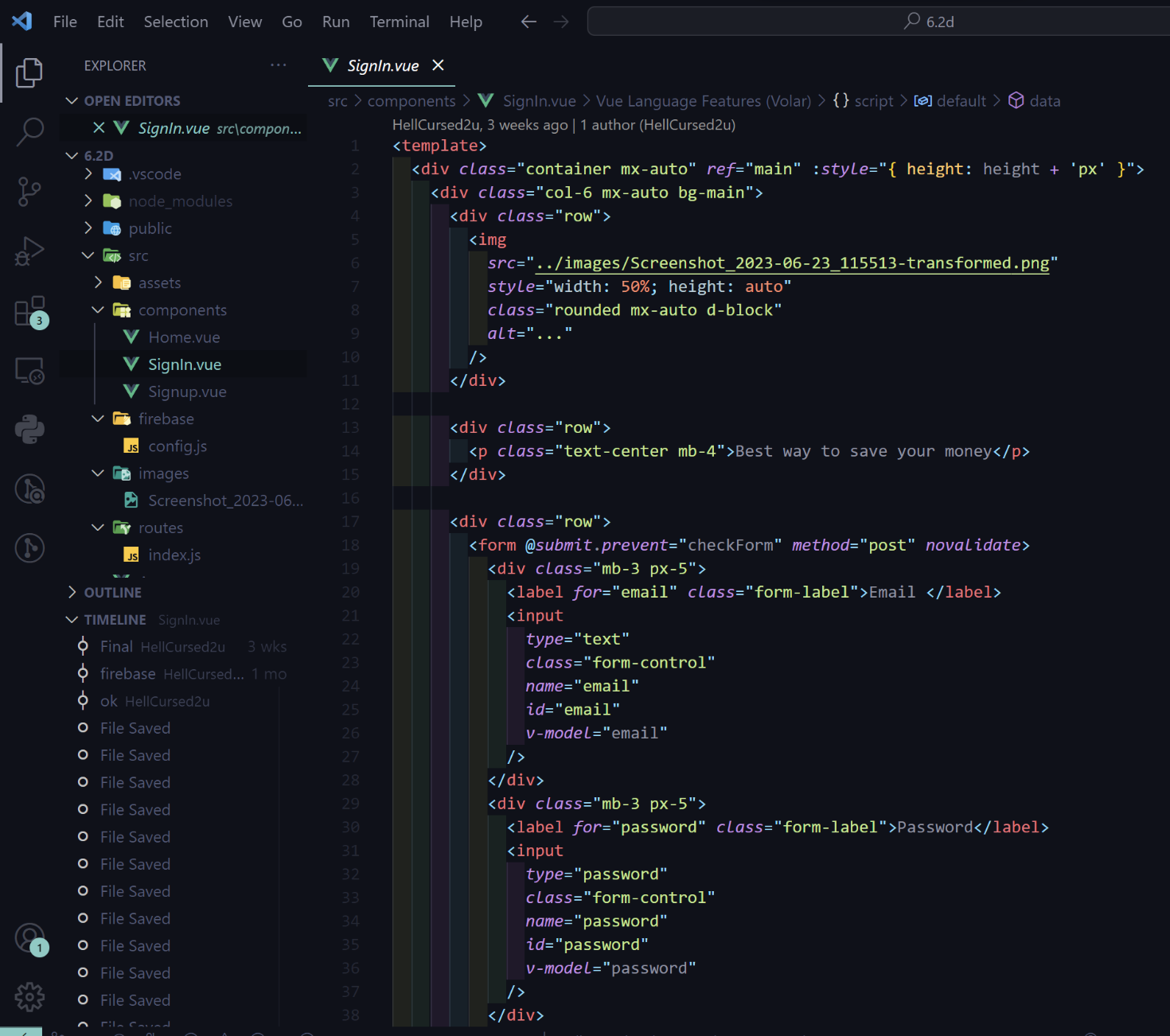
To use the login with google or facebook

**Gap 3: Components**

The purpose of this web application is used for authenticate. We will have three components, one for the app component which will display all child components by using Vue router

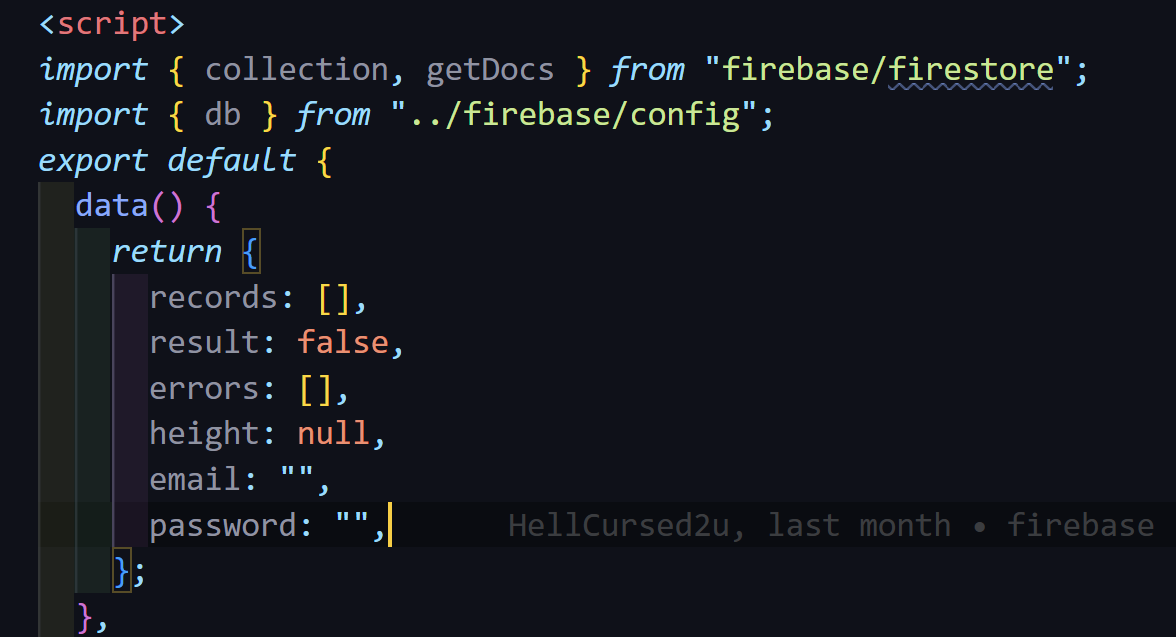


- 2 -

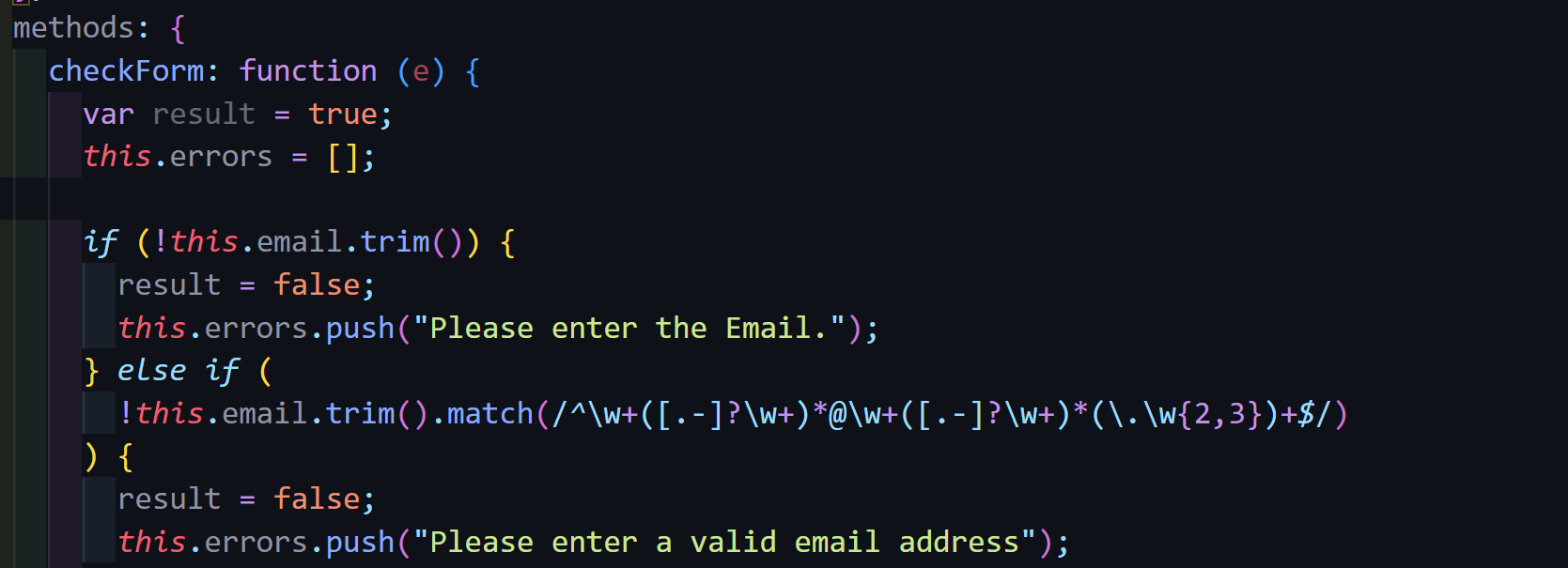


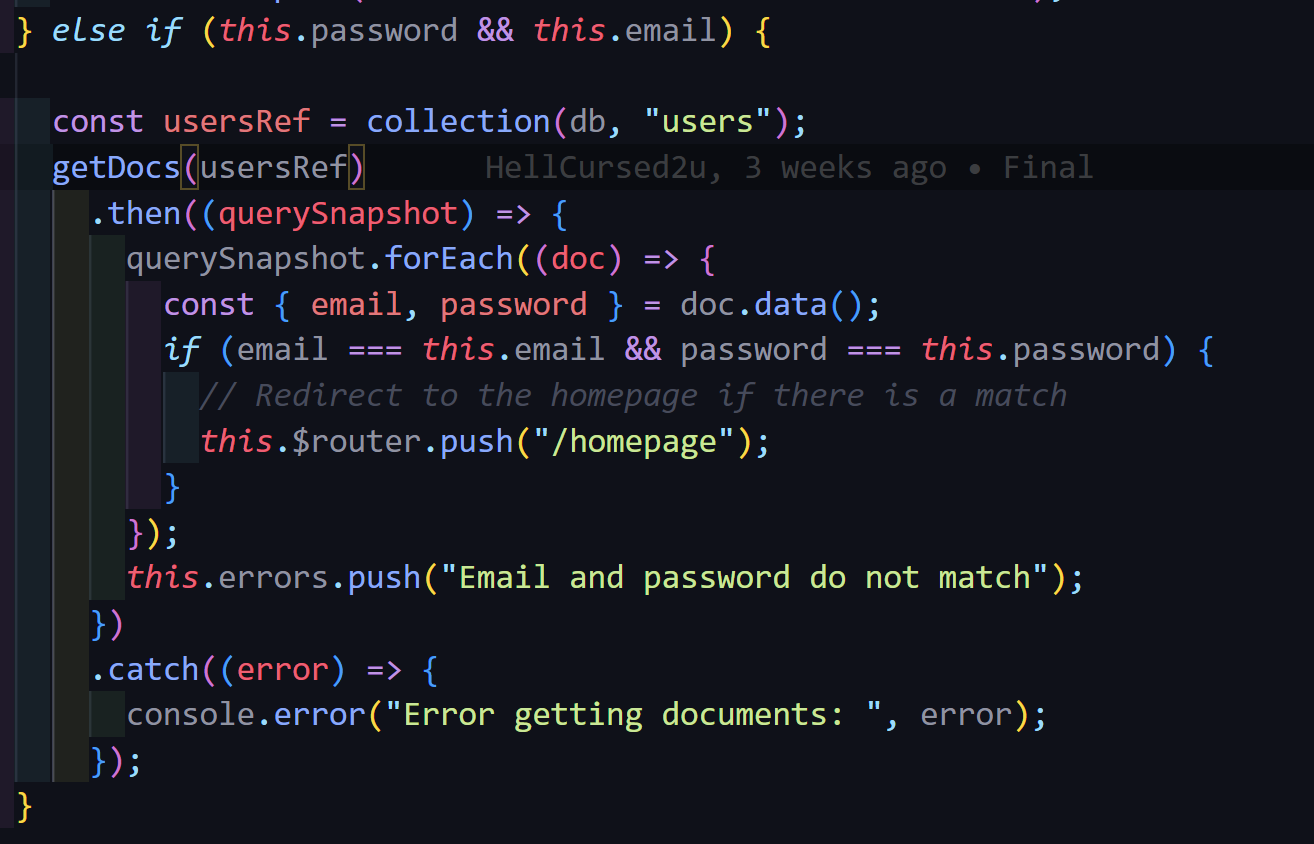
The Signin component

The SignIn Component will contain a header image an a form for user to loggin information

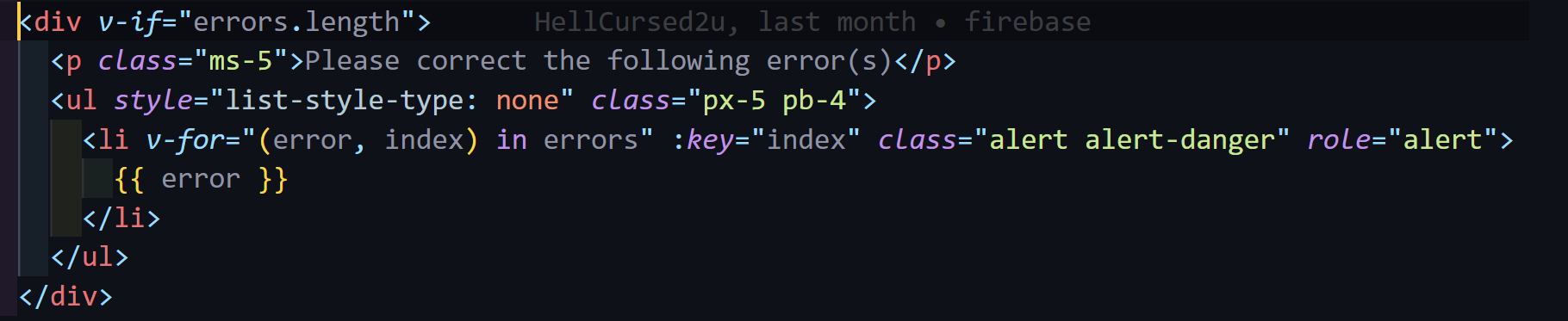


The password, email are used to create two way binding with the input user from the form. We will use this variables for data validation and compare with the data on firebase. All the errors from user input are stored in the errors array.

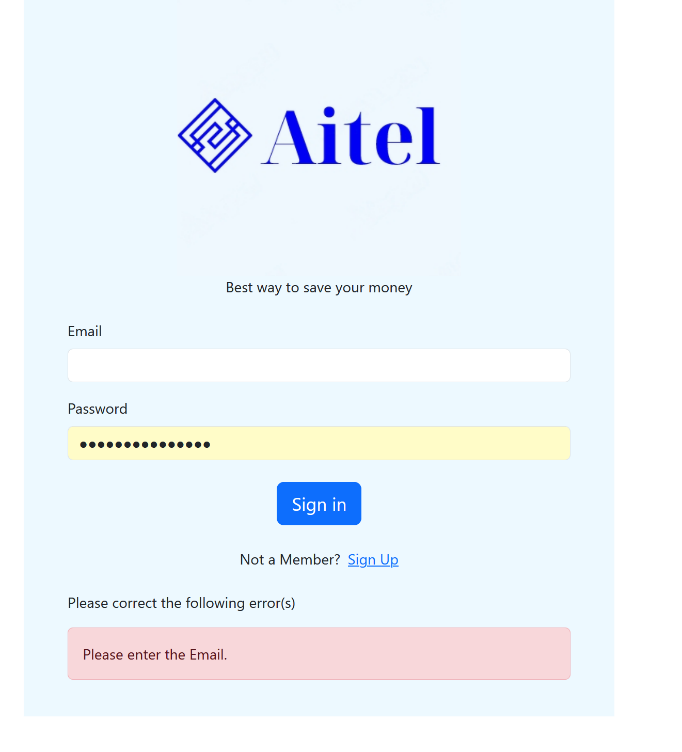
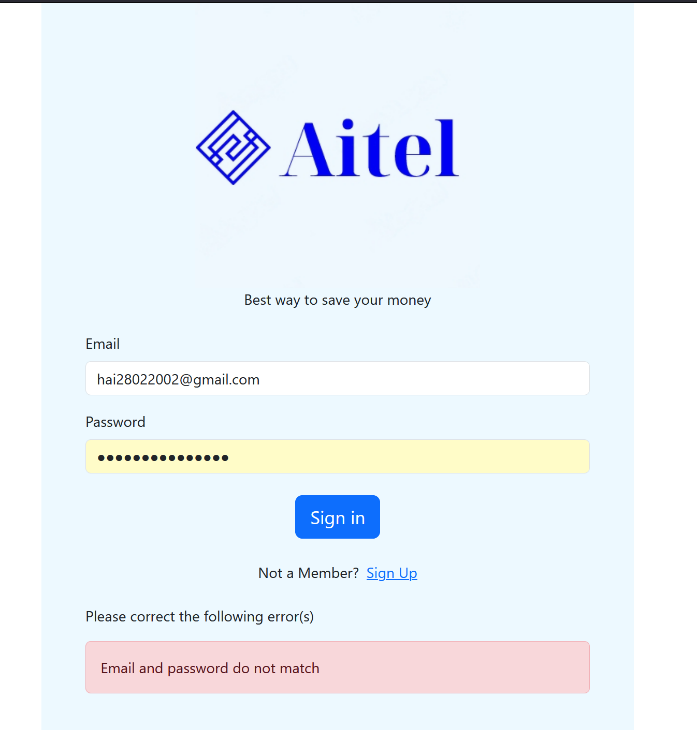
We add a click listener (@click) to the submit input in the form to execute the checkform function. This function used to check if the email is empty or not, if it is not empty, it will check the email in the right format by using regular expression. If it not fit the condition, push() method is used to add the error string tin the errors array.



This code is querying a Firestore database for a collection of users and checking if there is a match between the given email and password. The code query each doc in firestore which is a row of user and compare the email and password with the variables email and password from user input. If there is a match, it redirects to the homepage, and if there isn't, it adds an error message to an errors array.



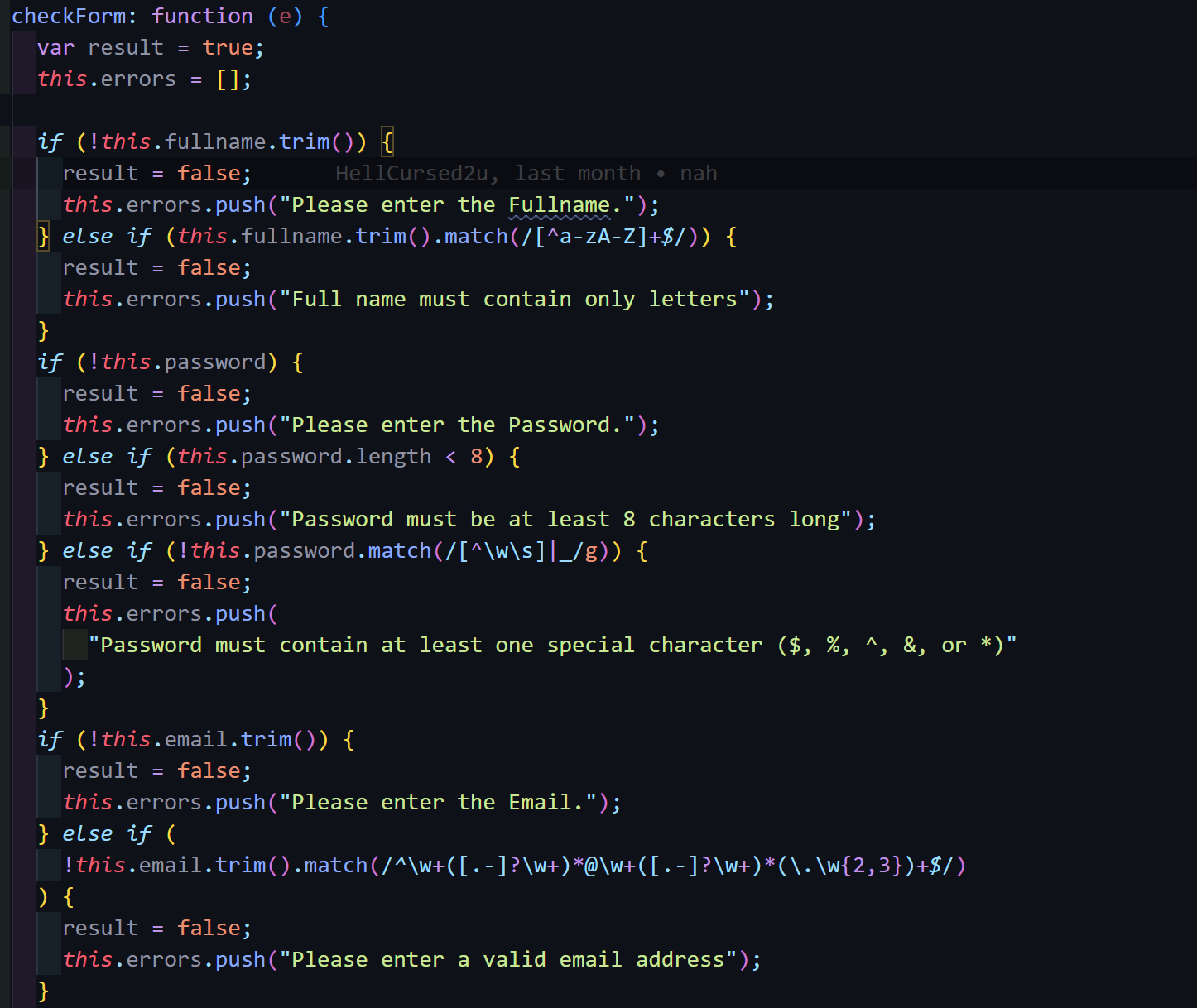
To display errors on the screen I use the v-if directive to check the length of errors array. If there is an error it will show the the error using v-for directive to display each error in the error array. We used Bootstrap for an easy deign



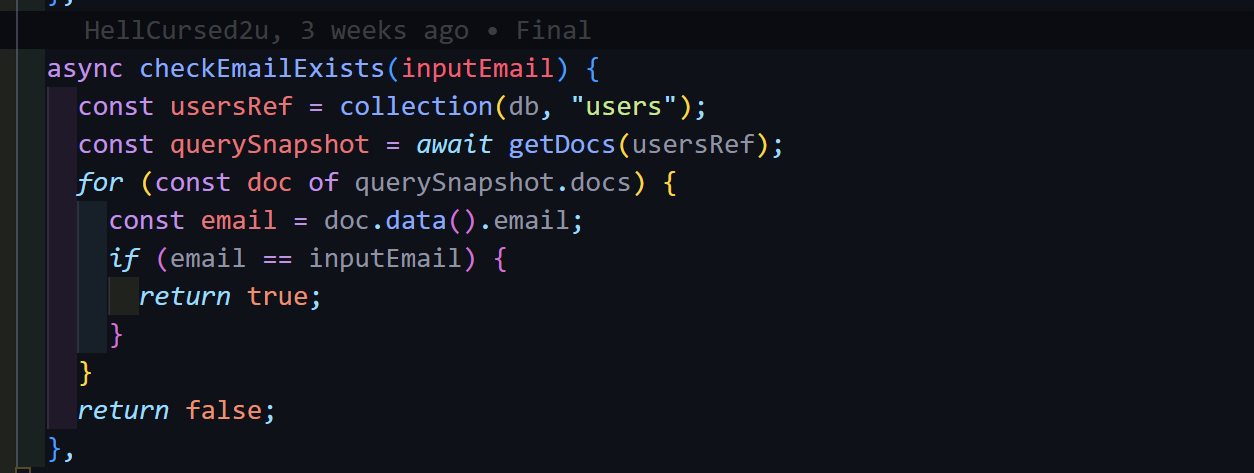
Register Page:

The step for the register page would mostly be the same. It contains a form for user to input, however, some steps must be configed as it is a register page.

The variables are v-model with the form input to have the two way binding.

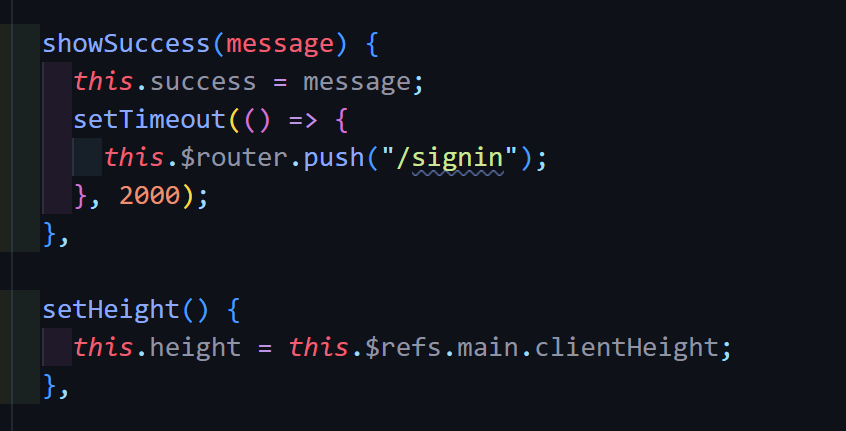


For the data validation, we used the same methods as the login page by using regular expression and store all the error in an array.

We implemented a function that will check if the user input email is exists or not, if email is already exists it return true, else retune false.



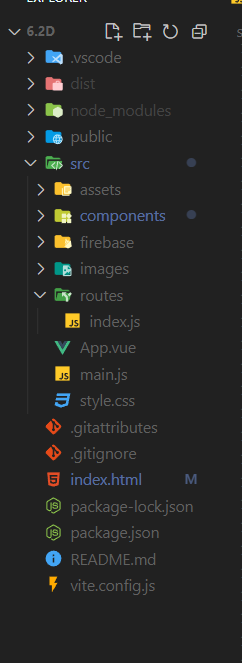
If all the volition is pass, We create a formdata, includes the fullname, email and password to insert into the Firestore. If the function emailExists returns true so it will push the error “Email is already exist” else, it will use the addDoc method provided by Firebase to inset the data in the database.

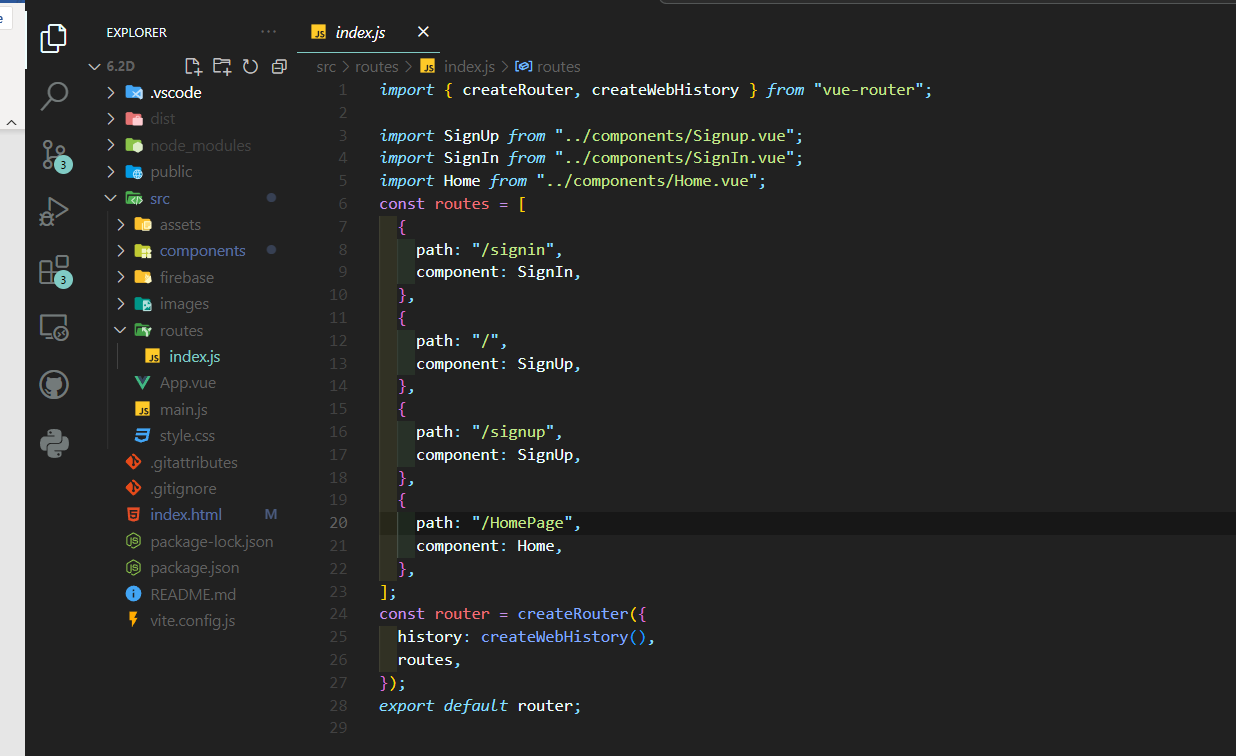


If the data successfully added, a success message will be shown and router will push to the lon in page after 2 second.

**Gap 4: Vue Router**

We also implemented the Vue Router to our web application. This the file project, we created a folder name routes and create an index.js to initial the router object.

****



It begins by importing the necessary functions from the vue-router package: createRouter and createWebHistory. The code then imports the component files for SignUp, SignIn, and Home. The routes array is defined next, specifying the different routes in the application. Each route object contains a path property indicating the URL path and a component property specifying the corresponding Vue component to be rendered. In this case, the SignIn component is associated with the /signin path, SignUp with both / and /signup paths, and Home with /HomePage. The createRouter function is called to create a new router instance with the specified routes and history mode set to use clean URLs. Finally, the router object is exported as the default export of the module. This configuration allows for navigation between different routes in the Vue.js application, rendering the appropriate components based on the accessed URL paths.