**BACK-END SETUP PROCESS**

Jonathan is going to go group-by-group to go over our setup status (frontend React app + GitHub Repo, backend GitHub Repo), and assisting with the backend setup.

He started with Cristiana going into the Backend folder in VS Code and run the following command:

**npm init –yes**

We used **axios** for our frontend API

We will be using **express** for the backend API

**npm i express @types/express cors @types/cors mongodb**

**Express** is for the API

**CORS** is so the front-end and back-end can talk securely

**MongoDB** acts as our back-end to store the info

**npm i -d concurrently**

Concurrently helps the backend communicate with Firebase

**Yes**

This makes a .gitignore for us

**firebase init**

Cristiana got the following error message because she didn’t have Firebase tools library installed

**zsh: command not found: firebase**

**npm i -g firebase-tools**

Because it is a Mac, she had to use **sudo npm i -g firebase-tools**

Tried to use **firebase init** after that, but still ran into an issue so went to **console.firebase.google** and setup a new project, following the steps involved.

The front-end and back-end of the app will live here in Firebase.

She then went back to VS Code to login to Firebase

**firebase login –interactive**

**interactive** tells the computer to open a prompt so she can login (say ‘Yes’ to allow it to pop up)

Once you login, go back into the terminal and use **firebase init**

Go down to Functions and hit Space, then Enter.

Select to use an existing Project and hit Enter

Select we want TypeScript

WE DO NOT WANT ESLint

Install Dependencies with NPM now? Yes

It will run through a series of installs and create a Functions folder. All our backend code will live inside here in the **src** folder. Any code outside **src** doesn’t exist as far as Firebase is concerned.

Go to the **tsconfig.json** and after line 9 put a comma and go to the next line

**“esModuleInterop”: true** (must change this from **false**)

**index.ts**

In the classroom GitHub Jonathan wrote a very basic server (computer that runs continuously and waits for someone to make a request with something like Express endpoints). This code says to create an App using Express and CORS and return with Express into JSON format.

The shoutouts aren’t necessary for a basic server, but he had it here.

Text

Description automatically generated with medium confidence

In the **Firebase Console** website you have to update the plan from Spark to Blaze so we can use **Functions**.

Go back to VS Code and in the **package.json** you need to add code to scripts:

“serve:dev”: “concurrently \“tsc --watch\” \“firebase emulators:start --only functions\“”,

If we want to run out back-end locally (like in Postman), the co

**npm serve:dev**

After this, we used **firebase deploy**

This can take a long time, so do not use CTRL+C to stop it because you could end up with a weird result.

With a **pipeline**, when you merge the code into main, then it will automatically **firebase deploy** to push the code up.

Back to Firebase Console and on the left you will see:

Build -> Functions

After this a URL will appear and then you hover over the HTTP to copy the address. This is the base URL for doing **API calls** to our **database**.

<https://us-central1-shakeemup-c22e5.cloudfunctions.net/api>

Until we have our endpoints setup will have to make the calls manually.

Pro tip is to not deploy off someone’s branch. Only deploy from main because we don’t want to push random branch code. Code only works so long as it is on the internet, not your computer.

Jakob owns the **front-end repo** and so had him open his VS Code and went to the folder that contains the hidden file **.git** (where our GitHub Repo exists).

But since Cristiana owns the Firebase Console Project, where Hosting occurs, he actually had her open a Terminal in the frontend folder and used:

**firebase init hosting**

Use an existing project

Y

Build

Y

Y – to build a pipeline; this will kick you over to GitHub so you can authorize Firebase

For which GitHub repository would you like to set up a GitHub workflow?

jsenyko/ShakeEmUps-FinalProject

Unfortunately, because Cristiana owns one and Jakob owns the other Jonathan was concerned we would have to do the **pipeline** manually since she isn’t the owner.

Set up the workflow to run a build script before every deploy? Yes

Set up automatic deployment? Yes

What is the name of the GitHub branch associated with your site’s live channel? Main

Notice how there is a **build** folder created in the **frontend** folder. So far, we have been building React apps in our personal computer. If a user had to download the entire React act code to use the App, then that would take a long time and suck. Instead, they download this optimized **build** which is faster. It appears in the **index.html** file. To use this, enter:

**npm run build**

This will add extra files to the **build** folder. This process will take a while as well. The **pipeline** will automatically update the **build** when new code is merged to the main, which is then pushed to Firebase.

**firebase deploy**

After this you should see your **Project URL**

Go back to Firebase and refresh the page and you will see the URL where your **App** appears on the web.

**shakemeup-c22e5.web.app**

When setting up tasks (say, in Trello), it is best practice to identify whether something is front-end or back-end to reduce confusion.