Lab: Build a React Native Chat App with Firebase

Estimated time needed: 45 minutes

Welcome to this hands-on lab, where you will create a chat app named **Buddy**, using React Native that connects to Firebase for authentication, storage, and database. Users can browse and chat with others who have signed into the app. This project will help you familiarize yourselves with creating a user-friendly interface, managing data with Firebase, creating indexes for the collections and implementing user authentication.

In this app, you will create components for:

- A Sign Up screen where users can sign up with email and password.
- A Log In screen where registered users can sign up.
- A tabbed screen with three tabs, which is only for authenticated users.
    - One tab for the screen showing the list of users
    - One for the screen where you can chat
    - One for the screen with settings, where users can upload or change the avatars.

To do this, for the backend, you will implement Firebase functions to:

- Add new chats to Firestore.
- Add new avatar URLs for the users in a new collection in Firestore.
- Fetch all the users who have signed up, other than the user, and display them on the home screen.
- Upload pics for avatar into Firebase storage on the settings tab.

Prerequisites

- You need a Google account to create a Firebase database, which will be used to create the Firebase project.

    If you don't have a Google account, please visit http://google.com to create one.

- You will need a valid credit card to sign up for the Firebase billing to use the free tier for Firebase storage. You will not be charged. The credit card is for verification purposes and to avoid misuse.

- You need an Expo account as you will be using it to create signed React Native mobile apps for Android and iPhone.

    If you don't have and Expo account, please visit https://expo.dev/ to create one. Please note that you have a limited number of builds with Expo that are within the free tier.

Objectives

After completing this lab, you will be able to:

- Create a React Native chat app integrated with authentication using Firebase storage and Firestore database.
- Evaluate user credentials using Firebase authentication in your React.
- Apply indexes on Firestore Databases to retrieve data based on more than one field.

Clone and set up your Expo app

The app has been created for you. You will need to clone it, install the packages, and set it up to run.

1. Clone the repository by running the following code in the terminal.

    ```
    git clone https://github.com/ibm-developer-skills-network/reactnative_chatapp.git
    ```

2. Change to the React app directory.

    ```
    cd reactnative_chatapp
    ```

3. This React app has been created by running `npx create-expo-app reactnative_chatapp --template blank`. The following files have been created in the root directory:
    - `firebase.js`: You will set the configuration based on the Firebase app you create.
    - `App.js`: The content of this file for the chat app has been changed. Unregistered users are rendered with the Sign Up or Log in screen. The chat app uses the `Tab` navigator to provide three tabs for authenticated users.
    - `SignUpScreen.js`: This file is rendered initially when the user opens the file. It allows the user to sign up with an email and password, one of the authentication mechanisms available in Firebase.
    - `LoginScreen.js`: This file contains the page that will be rendered for signed up users to log in.
    - `useAuthentication.js`: This file contains the code that handles the app's user management.
    - `ListUsers.js`: This file contains the app's home page. This page lists all the users who have signed up. This page is accessible only when the user logs in. To do this, you will create `Authentication` using Firebase.
    - `ChatScreen.js`: This file is designed to chat with one of the users you choose from. Please note that his screen is reachable only when you click the user you want to chat with, and all the chat conversations between the user logged in and the user chosen to chat with will be displayed. This requires using an index, which you will be creating in Firebase.
    - `SettingsScreen.js`: This file contains the code that allows the user to add or update an avatar image.

Please review the code in each of these files and make sure you understand it before proceeding.

4. The `package.json` contains all the dependencies required for the app to run. Run the following command to install the packages needed to run the app.

```
npm install
```

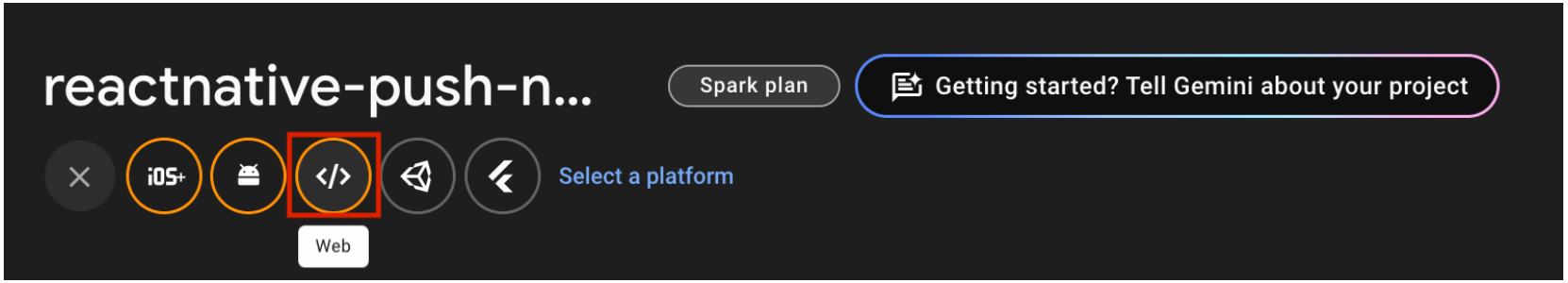
You cannot run the web app yet as the Firebase configuration is not set up.

Create a Firebase project and an app

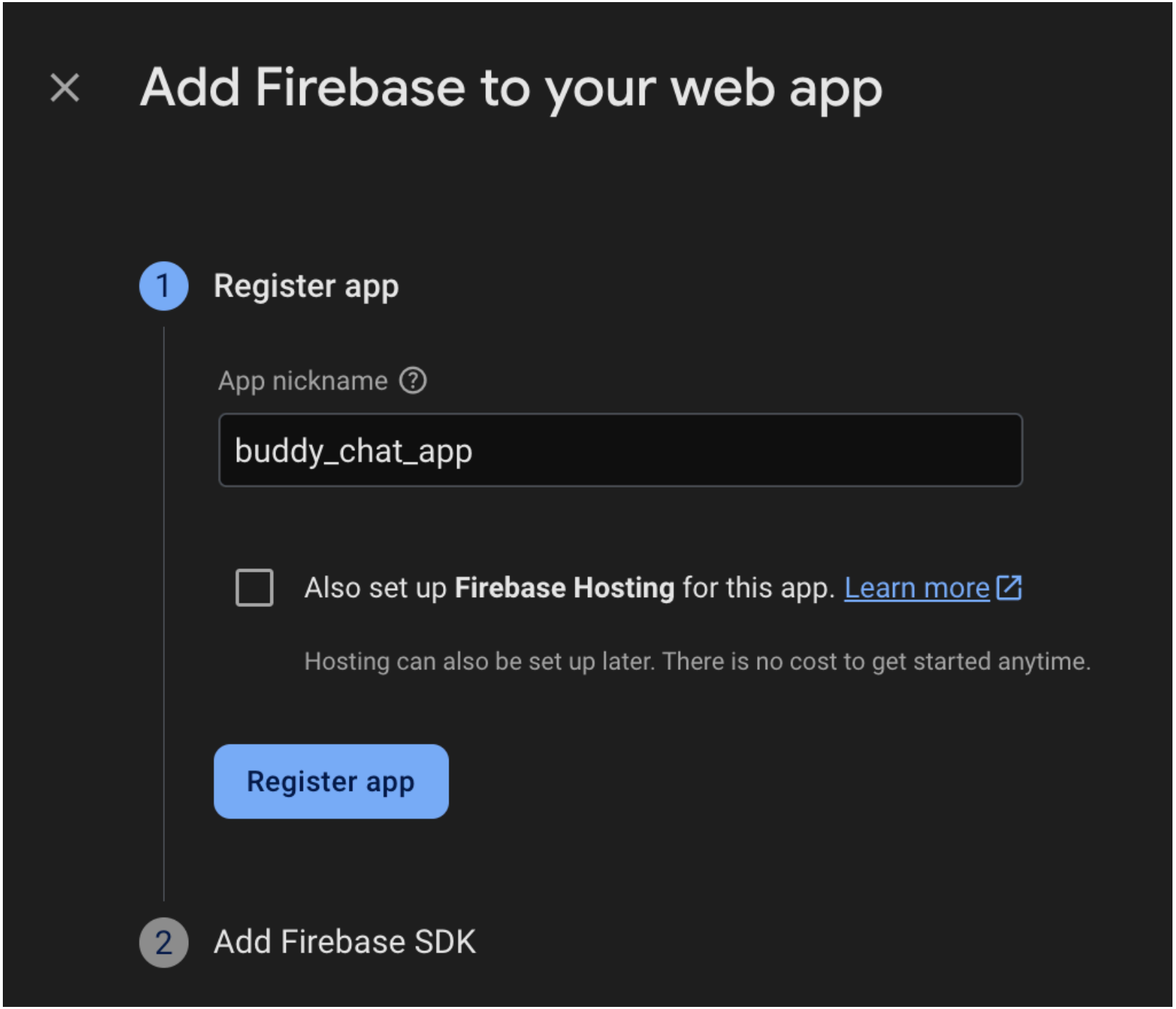
1. Go to Firebase Console and open your existing project or create a new project.

   If you use the same project that you used for the previous labs, you can reuse the already created authentication, storage, and database.

2. Select the `Web` icon to create a new web app in the project dashboard.



3. Give the app a name, such as `buddy_chat_app`, and select `Register app`.


Add Firebase to your web app
1
Register app
App nickname
buddy_chat_app
Also set up Firebase Hosting for this app. Learn more
Hosting can also be set up later. There is no cost to get started anytime.
Register app
2
Add Firebase SDK

4. Firebase generates the configuration for you to add to the `firebase.js`, where placeholders have been provided. Click the copy icon to copy the content onto your file. Make sure you retain only the configuration part from the copied code and remove the rest of the code.

Register app

2 Add Firebase SDK

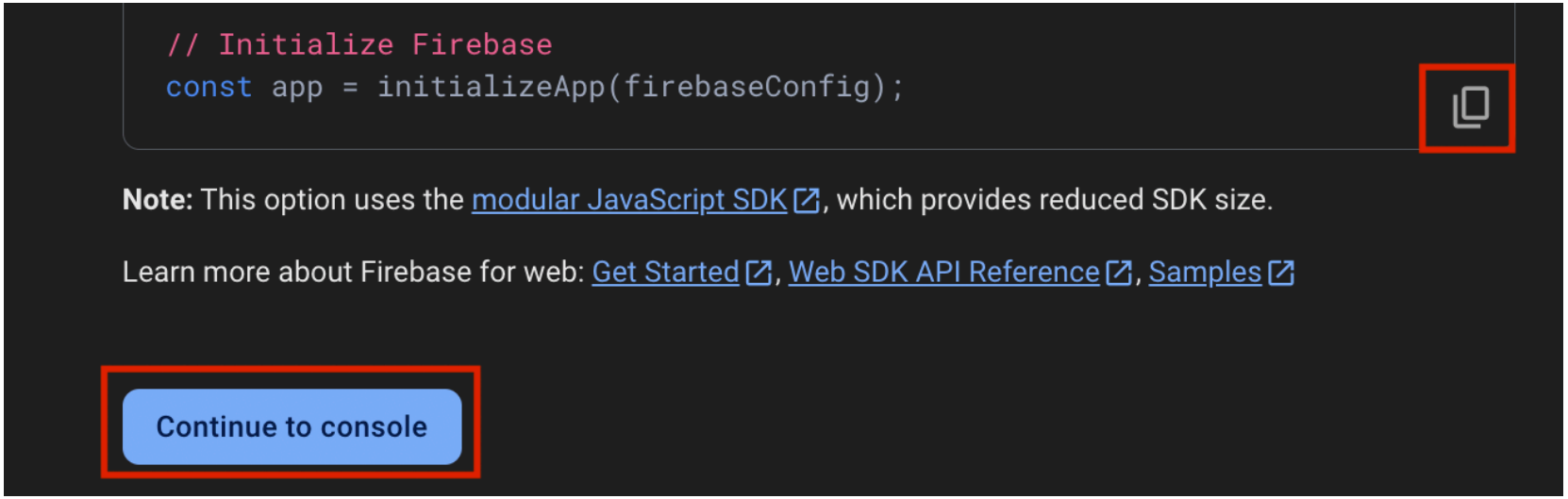
Use npm Use a <script> tag

If you're already using npm and a module bundler such as webpack or Rollup, you can run the following command to install the latest SDK (Learn more):

```
$ npm install firebase
```

Then, initialize Firebase and begin using the SDKs for the products you'd like to use.

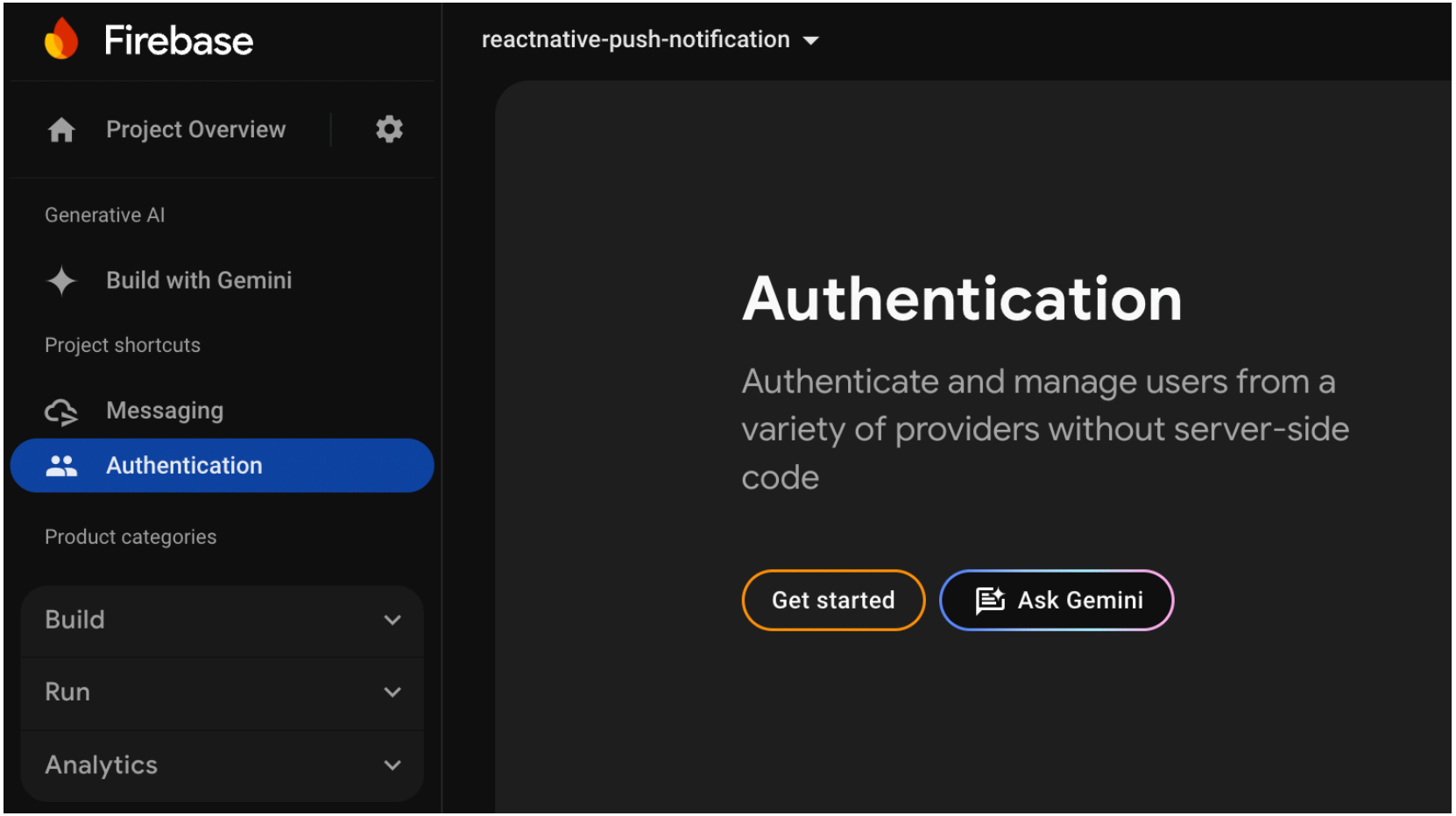
```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyByvHyTPvwSD5iI9gFMeKk89eCoL3P__Ss",
  authDomain: "reactnative-push-notific-e871b.firebaseapp.com",
  projectId: "reactnative-push-notific-e871b",
  storageBucket: "reactnative-push-notific-e871b.appspot.com",
  messagingSenderId: "98427882719",
  appId: "1:98427882719:web:8a17a006cb531468eaed6a"
};
```



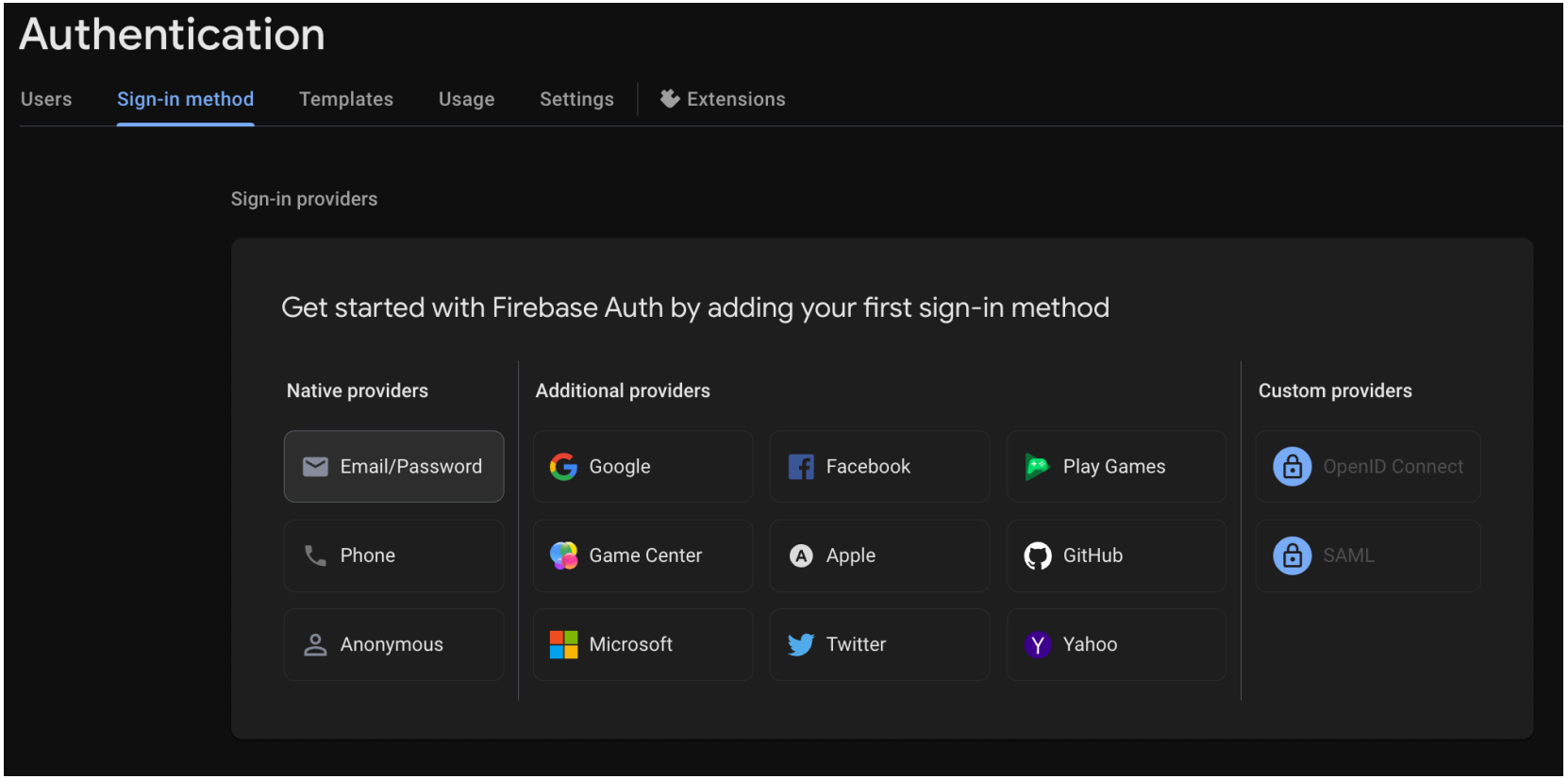
Add authentication in Firebase

If you have done other labs where you have enabled Email/Password authentication in Firebase, you can skip this step.

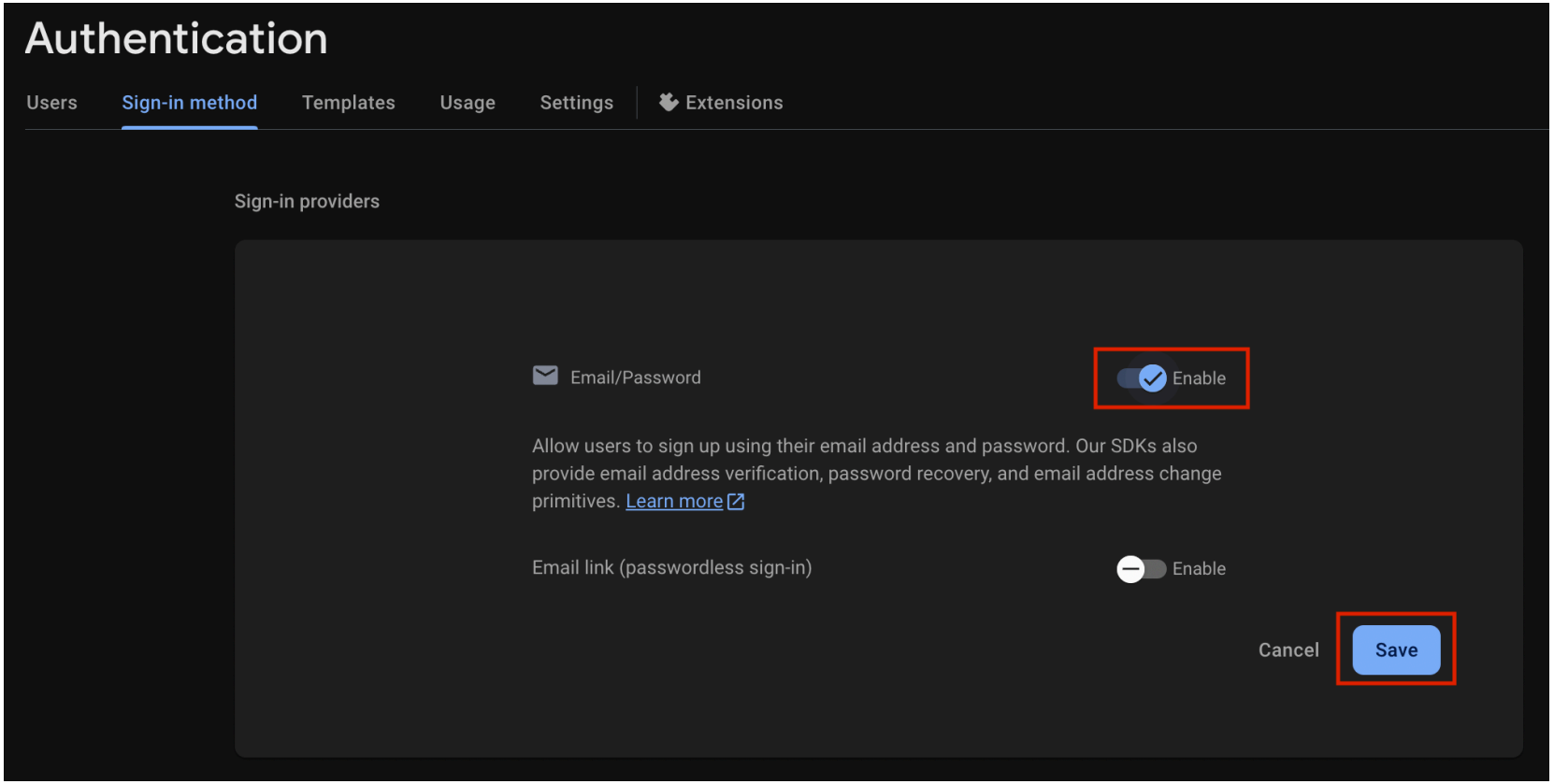
1. Select `Authentication` under `Build` on the left, then select the `Get started` button.



2. You will choose `Email/Password` authentication.



3. Enable `Email/Password` sign up, allowing users to sign up with their email and password and select `Save`.



You have now added authentication to your app.

4. Return to the Cloud IDE and run the app with the following command.

```
npm run web
```

Chat App

5. If the configuration has been set right, you will see the app's sign up page and be able to sign up using email and password.

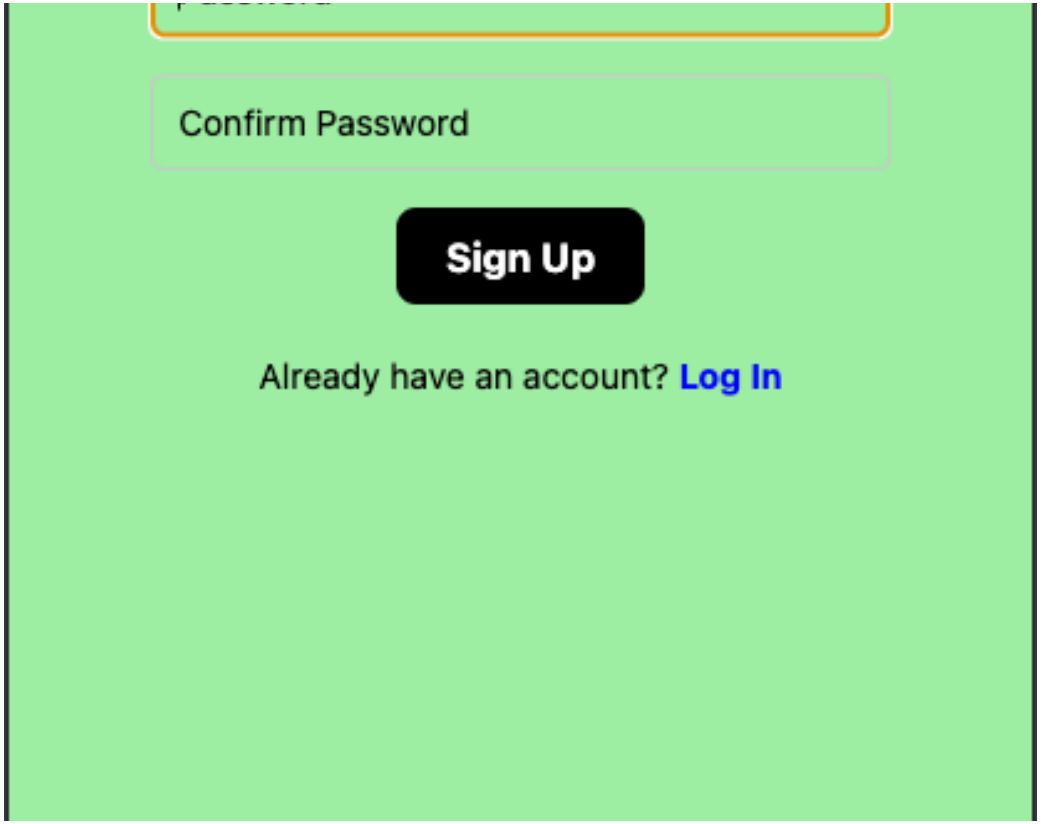
   If you have users signed up already for other labs and are reusing the project, you can use the same login credentials.

SignUp

Let's start Buddying

Email

Password

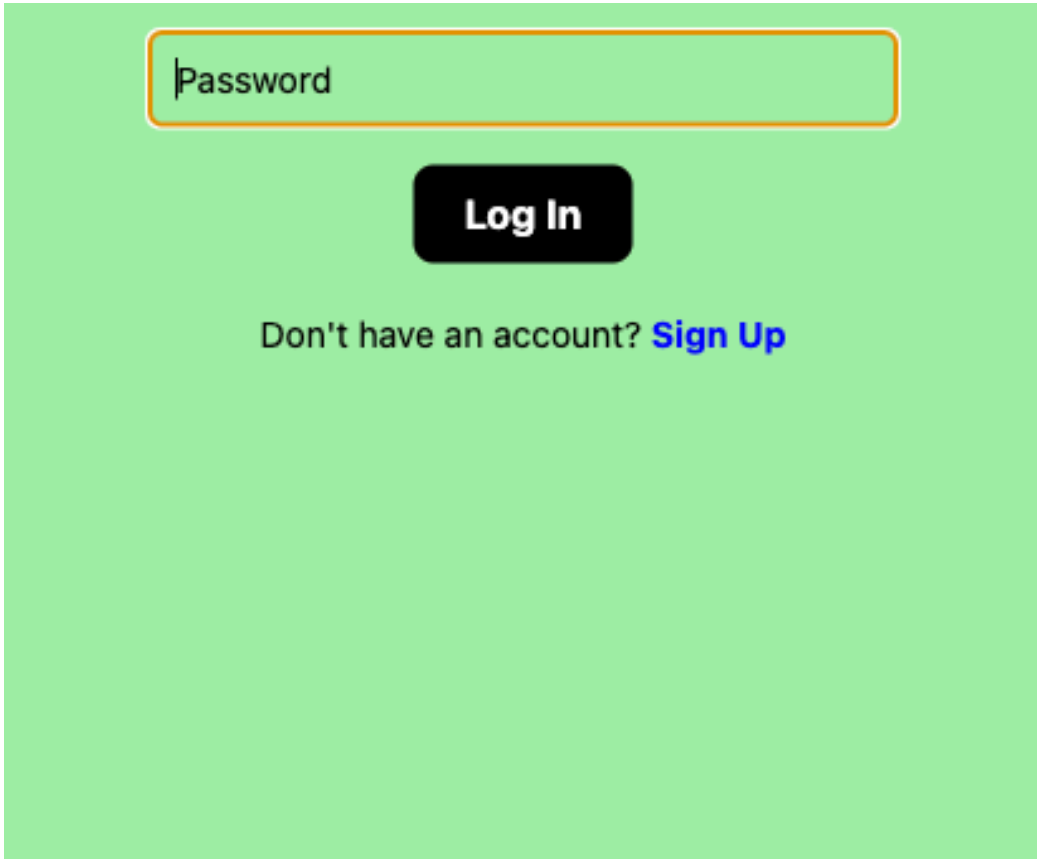


6. If you choose to log in with the existing credentials that you created in one of the previous labs of the same project, you can click the `Log In` link on the sign up page to go to the log in screen.

Login

Log In

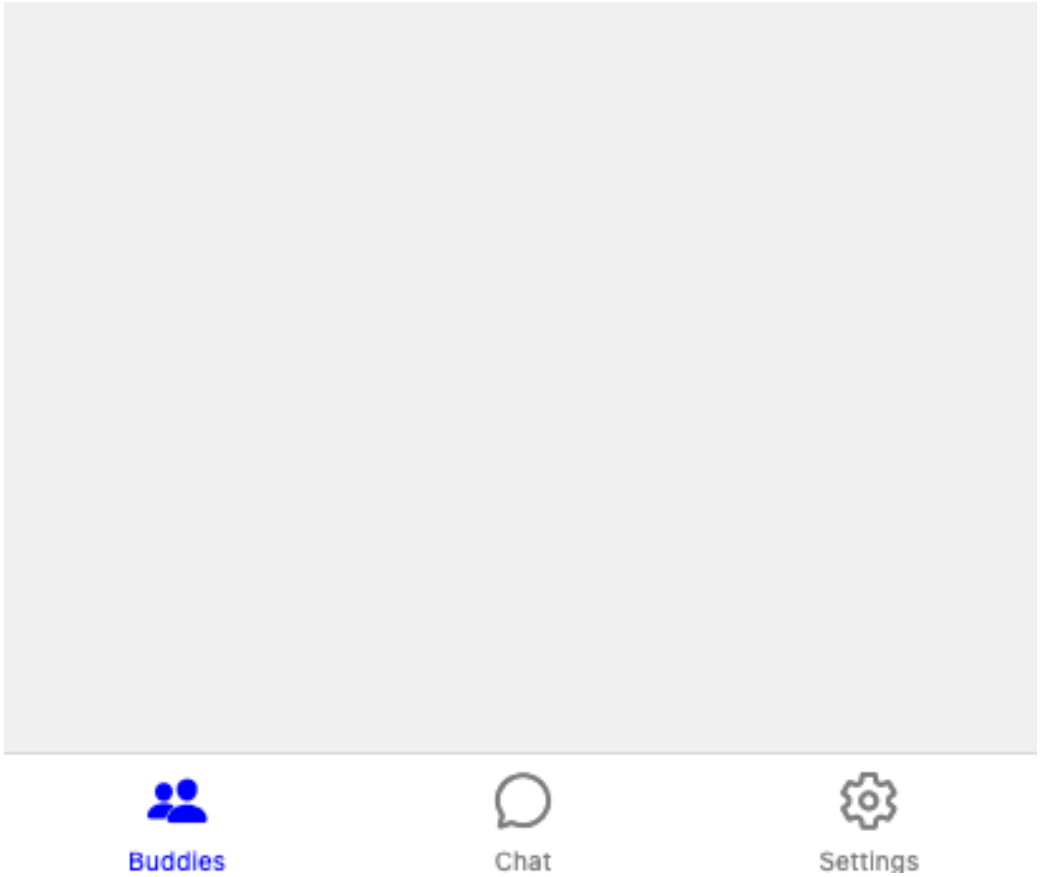
Email



7. Once you sign up or log in, you will be taken to the home screen, where you can see the three tabs at the bottom or log out. You cannot chat or upload the avatar yet on the settings tab as you need to create the resources for this purpose in Firebase.

If the user doesn't have an avatar, you will see the default, randomly generated image from this URL.

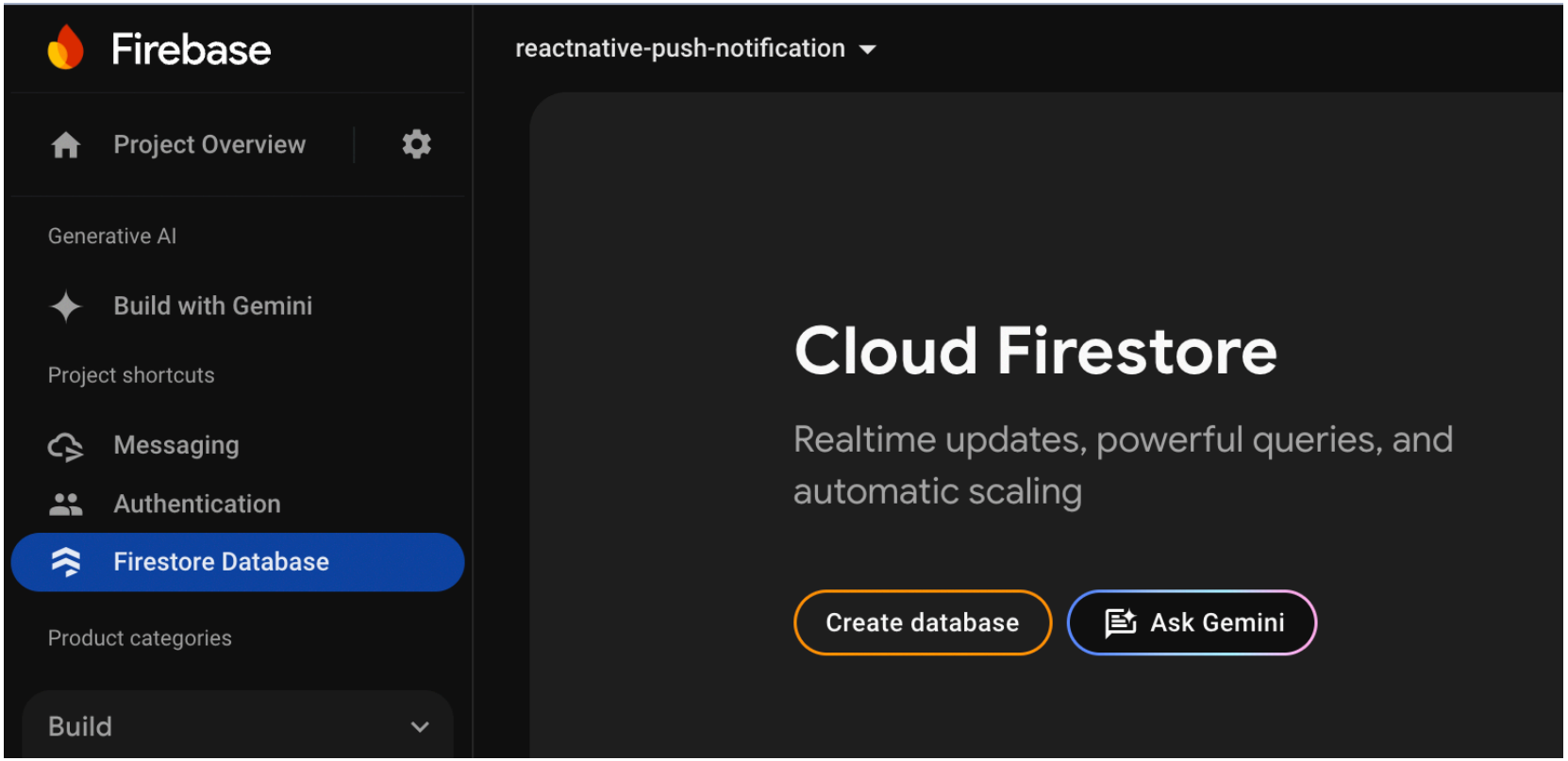

Buddies



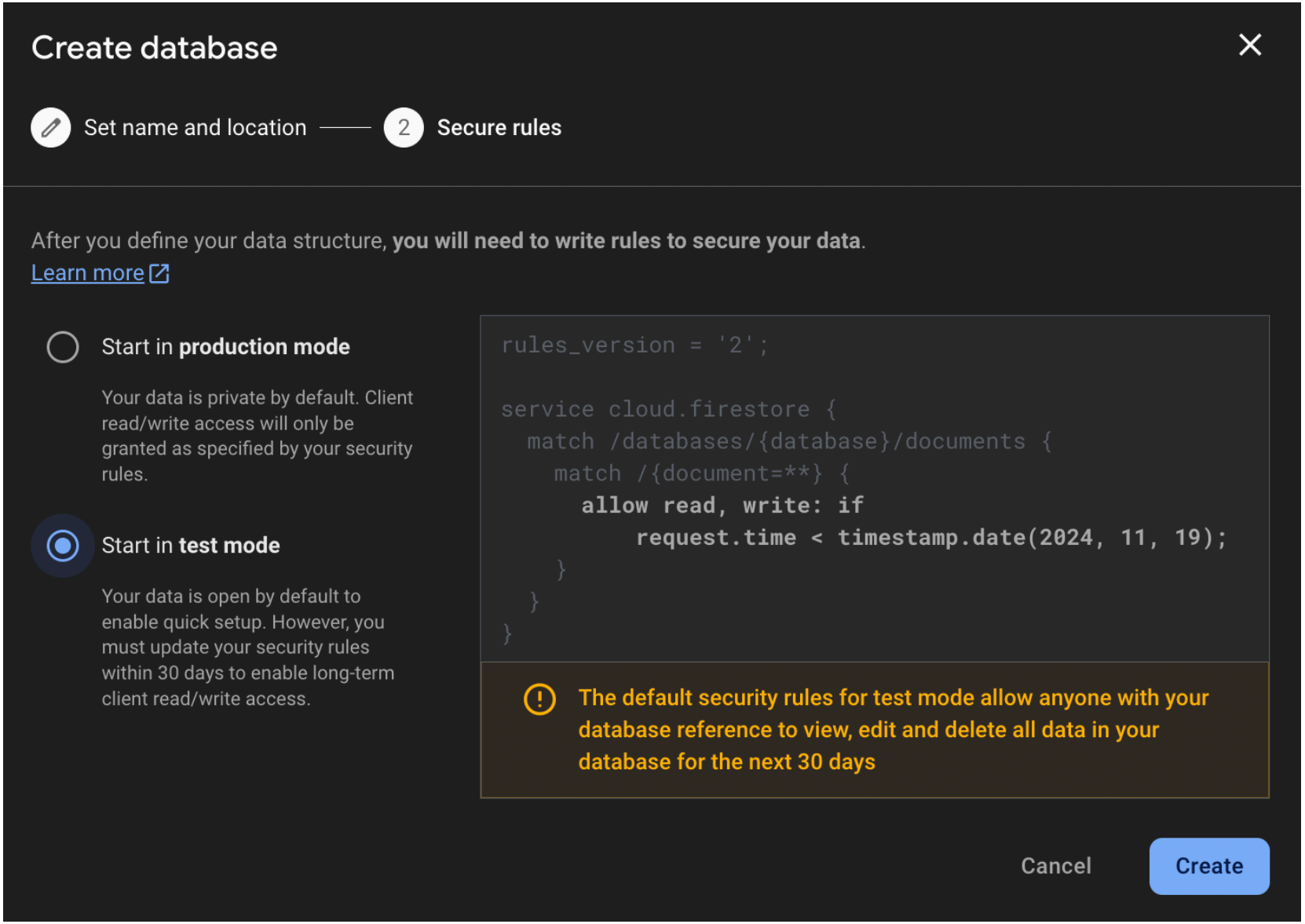
Add database and storage on Firebase

If you already have a database created for your project from the previous labs, you can skip the first three steps and go to step 4, where you will create two new collections.

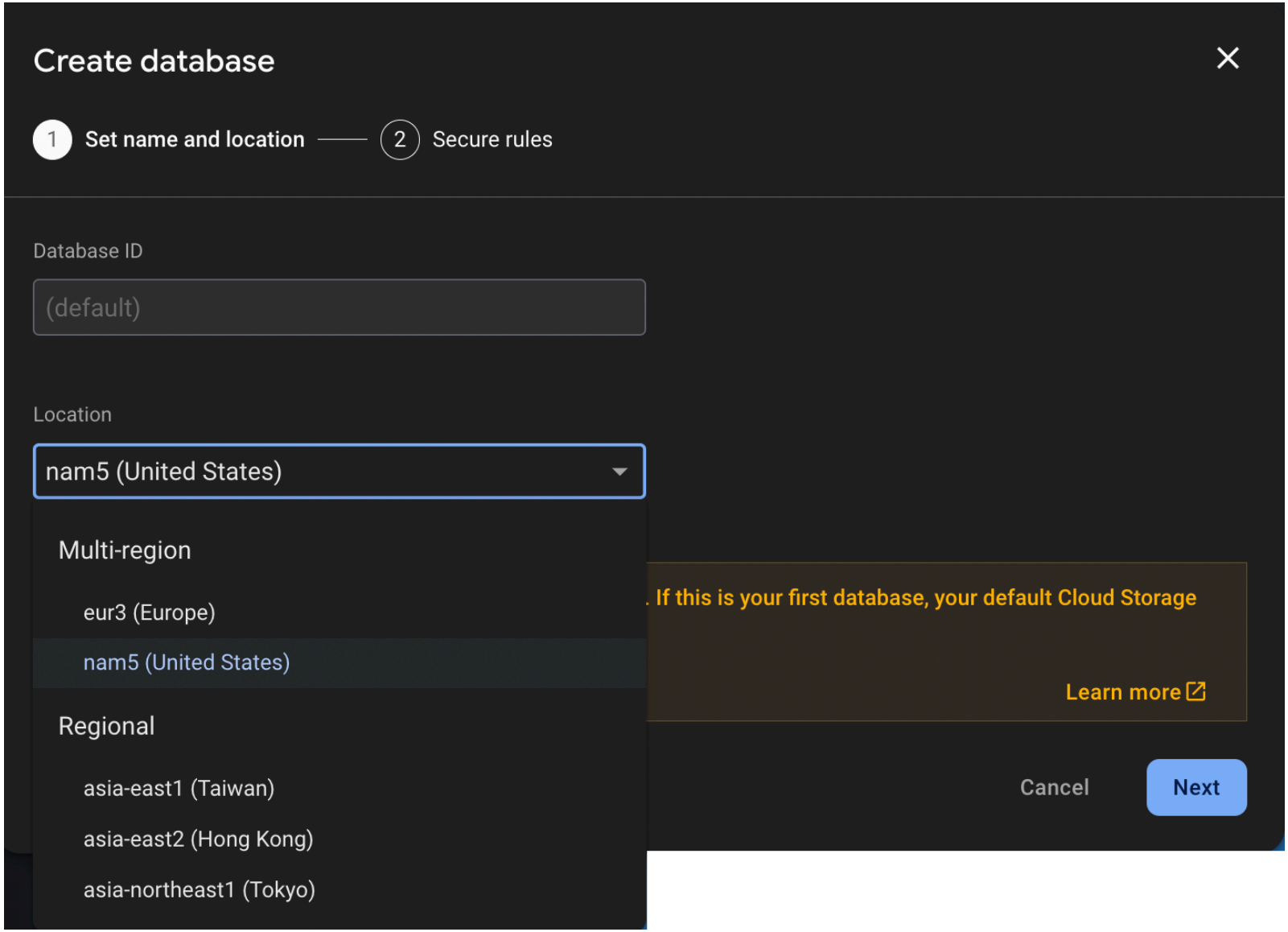
1. From the left menu on the `Firestore Database` console, under `Build`, choose `Database` and select the `Create database` button.



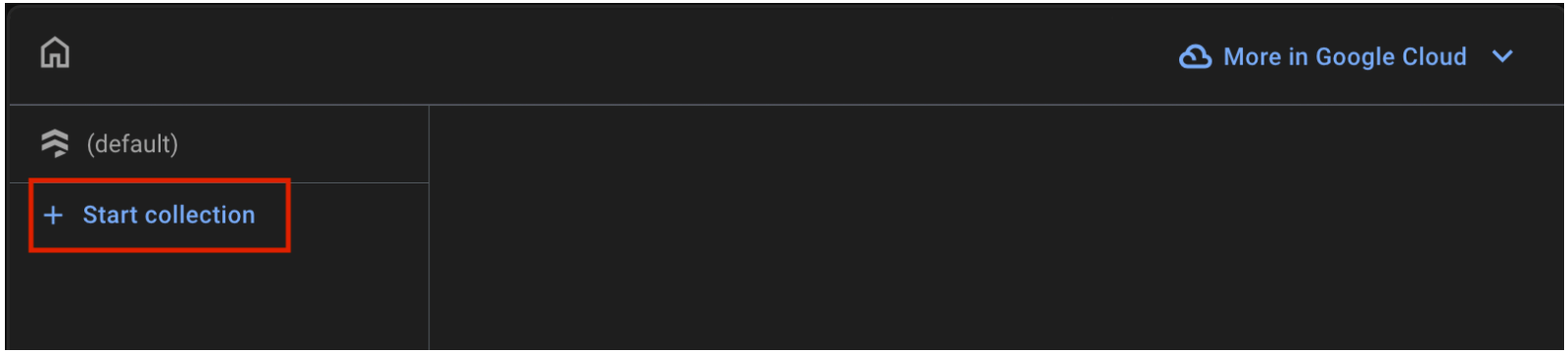
2. Choose to create a database in test mode.



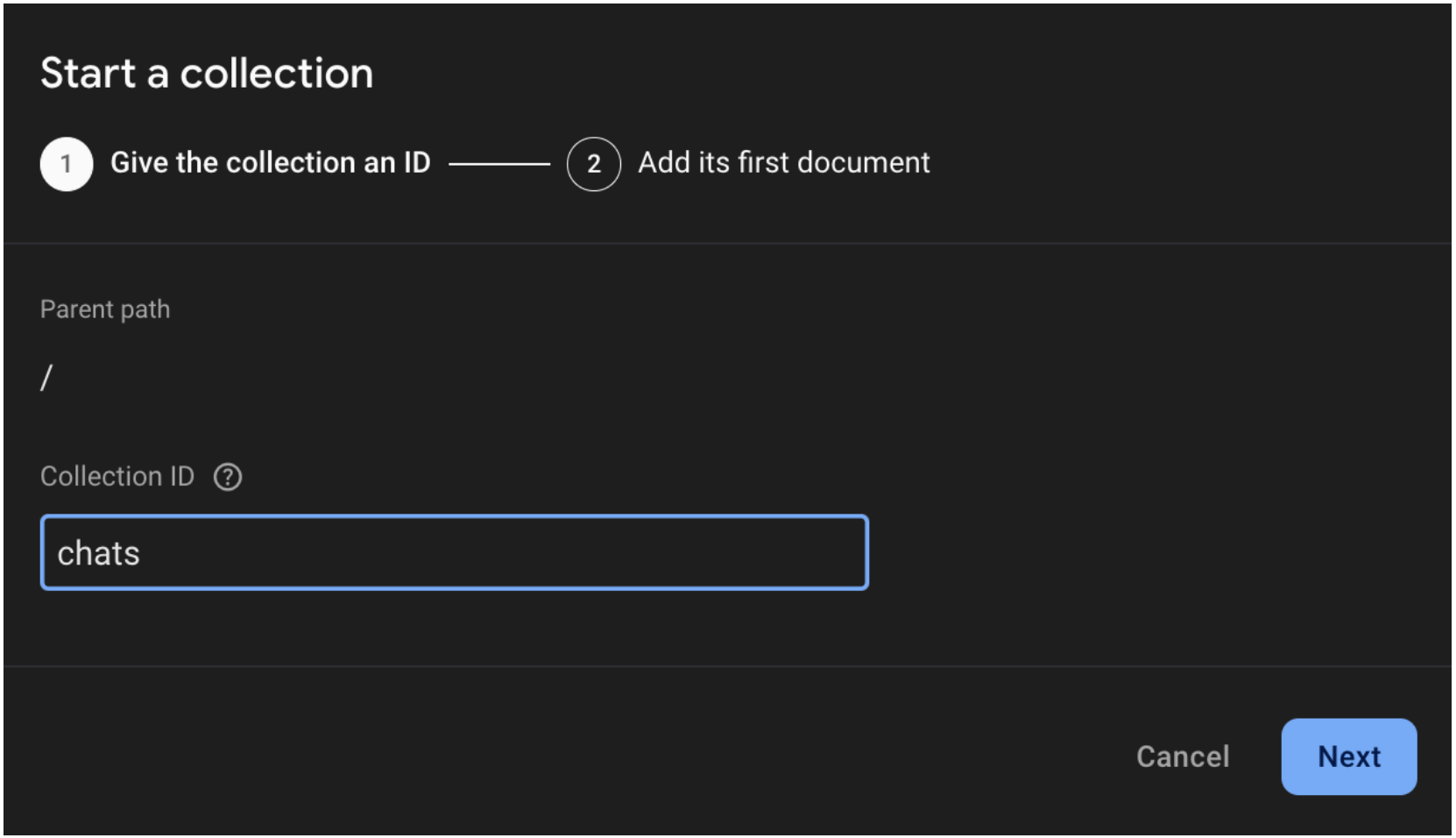
3. Choose the region where you want to create the database and press `Next`.



4. Select `Start collection` to create a collection.



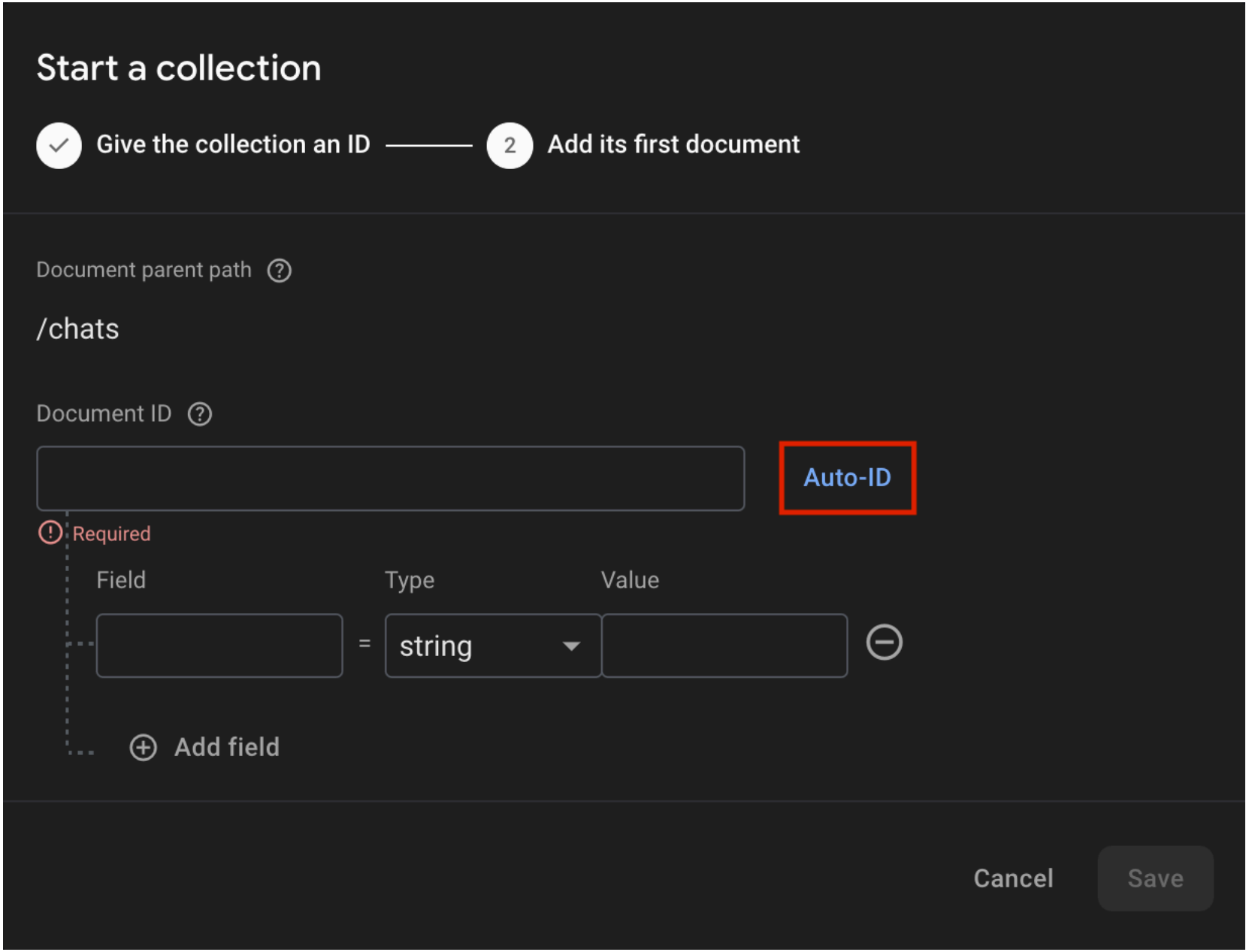
5. Name the first collection as `chats` in your database. This is to maintain the details such as the sender, receiver, chat text, and date. Click `Next`



The `chats` collection has now been initiated for creation.

6. It prompts you to add the first document. Click `Auto-Id` to generate an id.

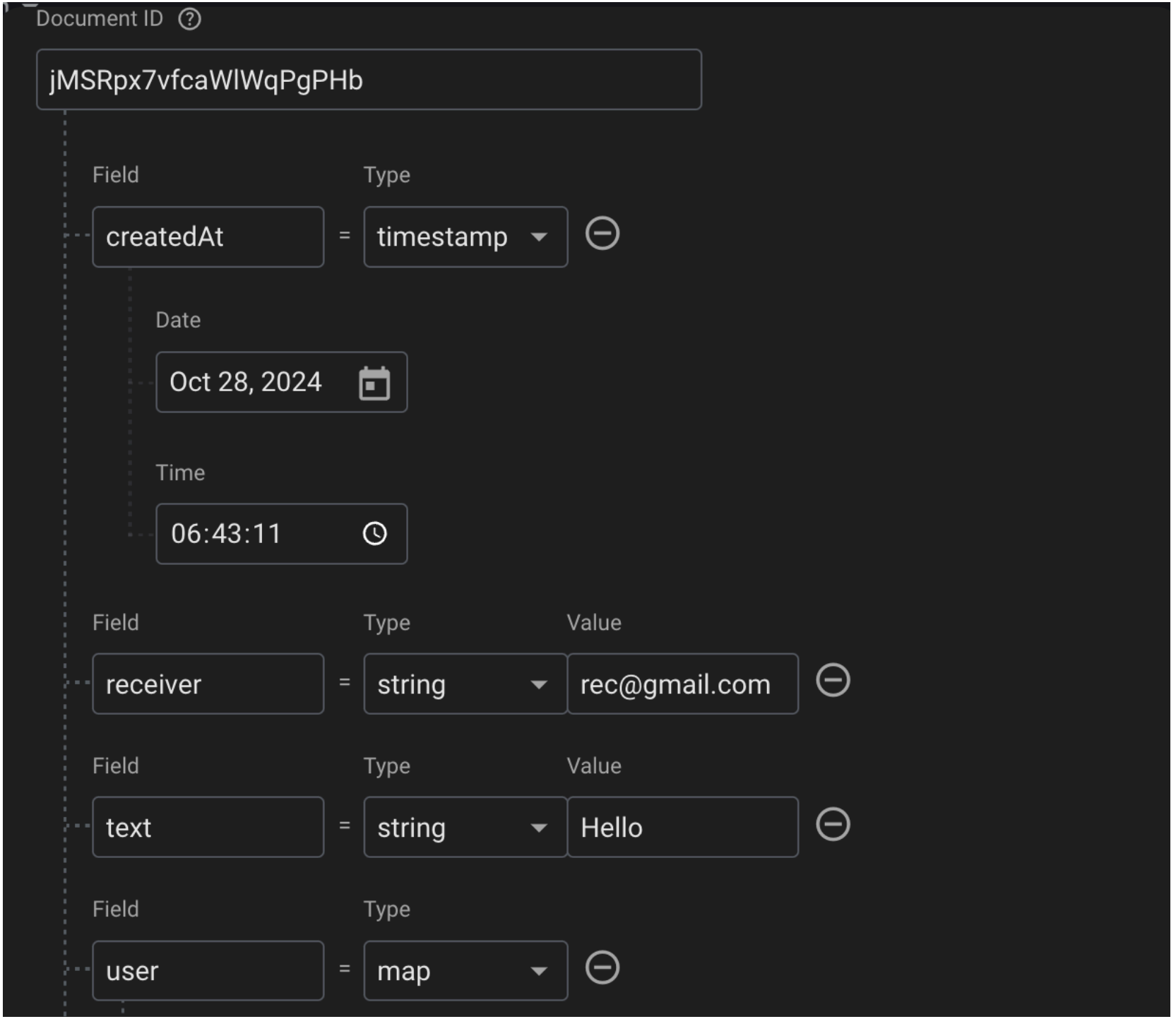
   You will add a dummy document with all the required fields to the `chats` collection. This will help you in creating an index.

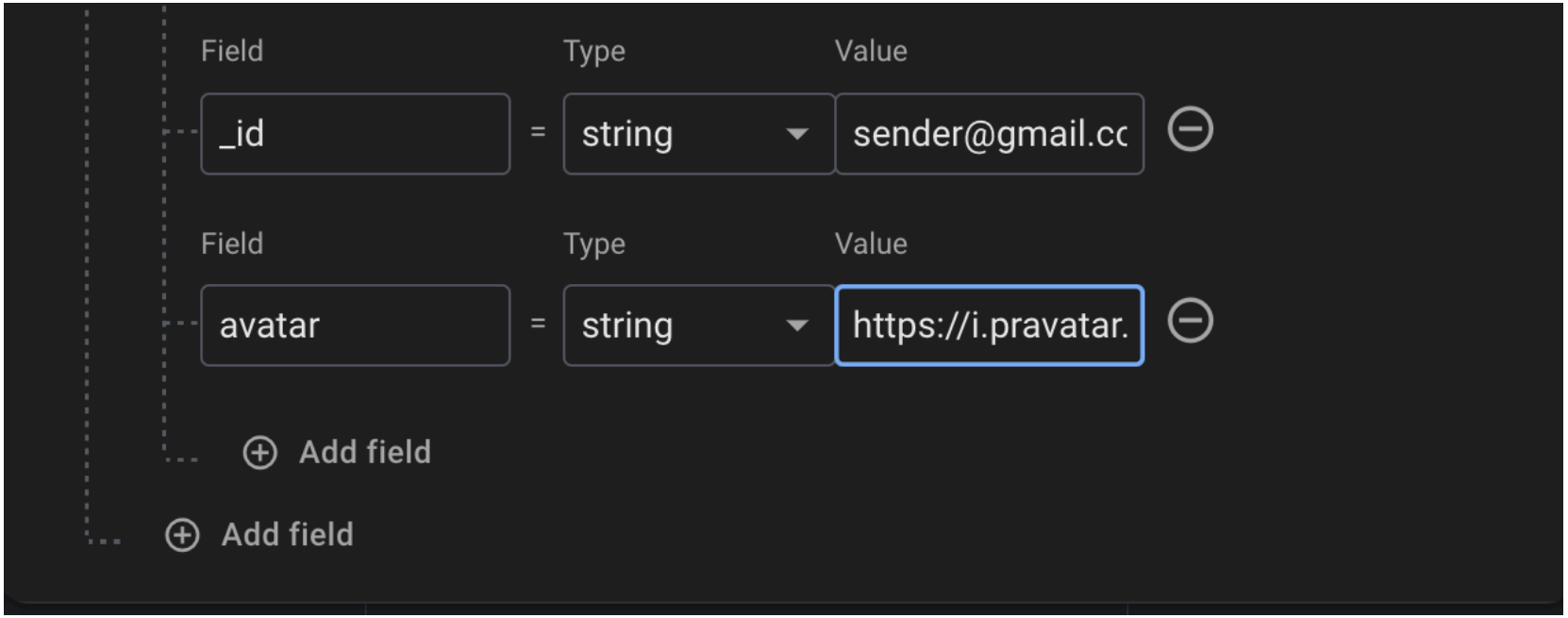


7. Add the other fields with dummy values.

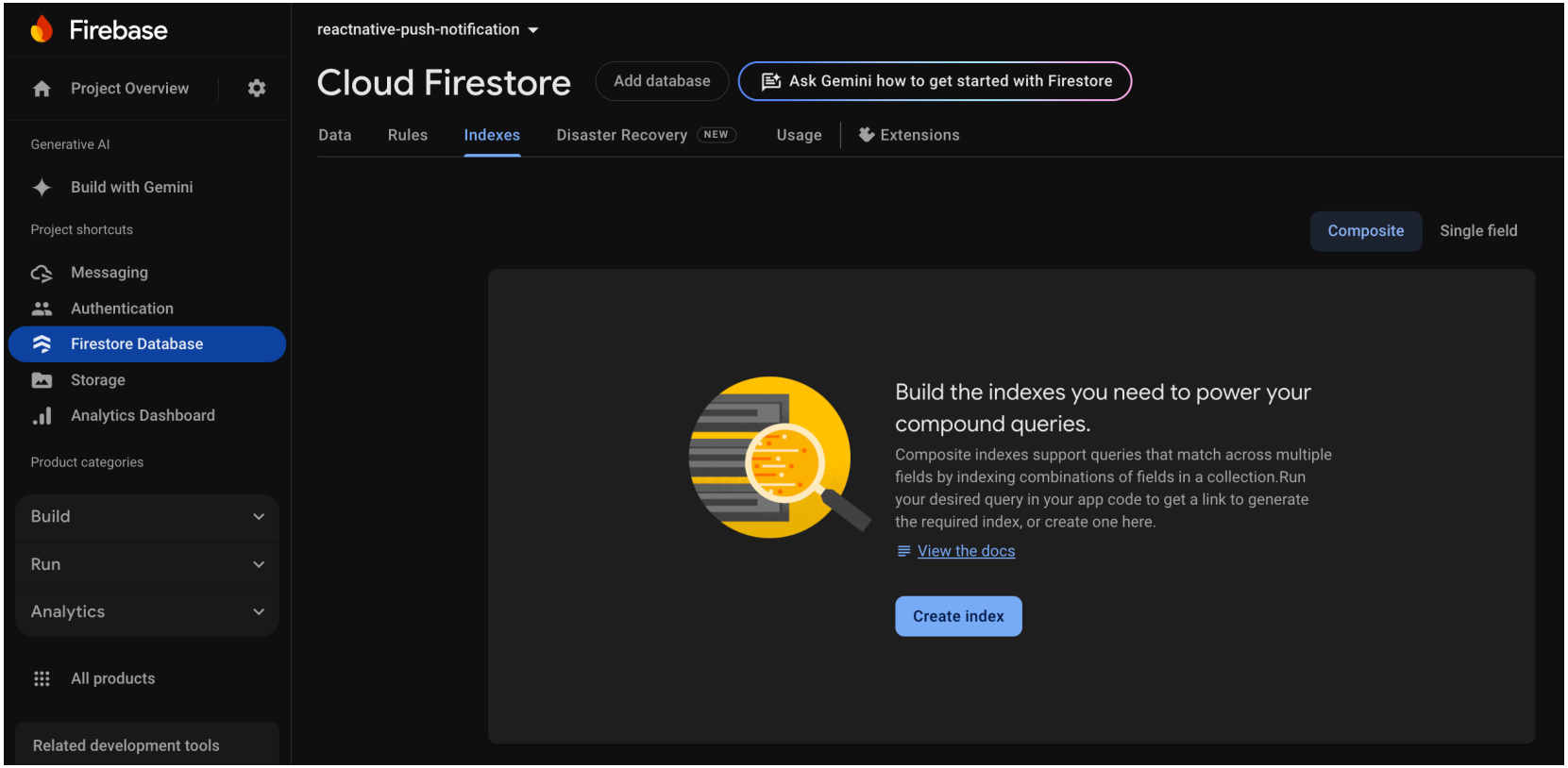
- createAt (timestamp)
- receiver (string)

- text (string)
- user (map)
    - _id (string)
    - avatar (string)


Document ID
jMSRpx7vfcaWlWqPgPHb
Field
Type
createdAt
=
timestamp
Date
Oct 28, 2024
Time
06:43:11
Field
Type
Value
receiver
=
string
rec@gmail.com
Field
Type
Value
text
=
string
Hello
Field
Type
user
=
map



8. On the chat screen, you want to render only the chat between the sender and receiver. Based on these details, you will need to create two indexes for your collection. Go to your Firebase console. Choose `Firestore Database` from the left menu. Go to the `indexes tab` and click `Create index`.



9. Create an index with the sender in ascending order using the field `user._id` and the date and time of chat in descending order using the field `createdAt`.

Create a composite index

Cloud Firestore uses composite indexes for compound queries not already supported by single field indexes (ex: combining equality and range operators).

Collection ID

chats

Fields to index

At least two fields required*

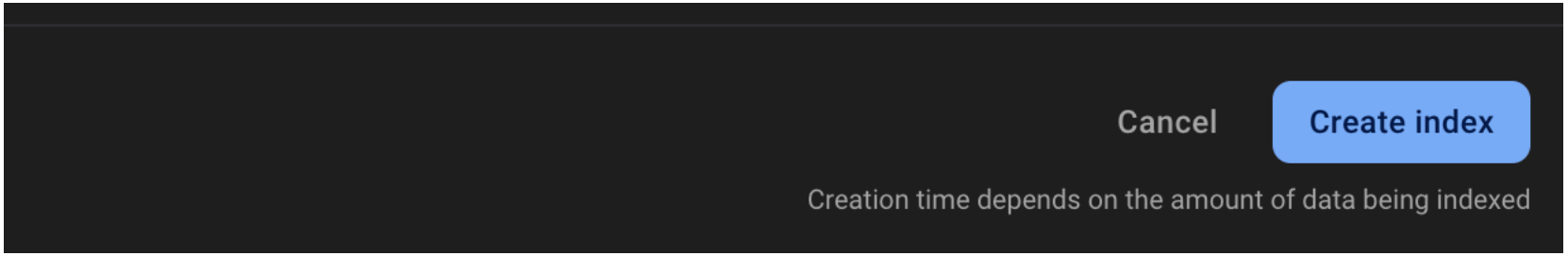
1	user._id	Ascending
2	createdAt	Descending

Add field

Query scopes

Collection

For queries within a specific collection path



10. Create an index with ascending order of the receiver using the field `receiver` and descending order of the date and time of chat using the field `createdAt`.

Create a composite index

Cloud Firestore uses composite indexes for compound queries not already supported by single field indexes (ex: combining equality and range operators).

Collection ID

chats

Fields to index

At least two fields required*

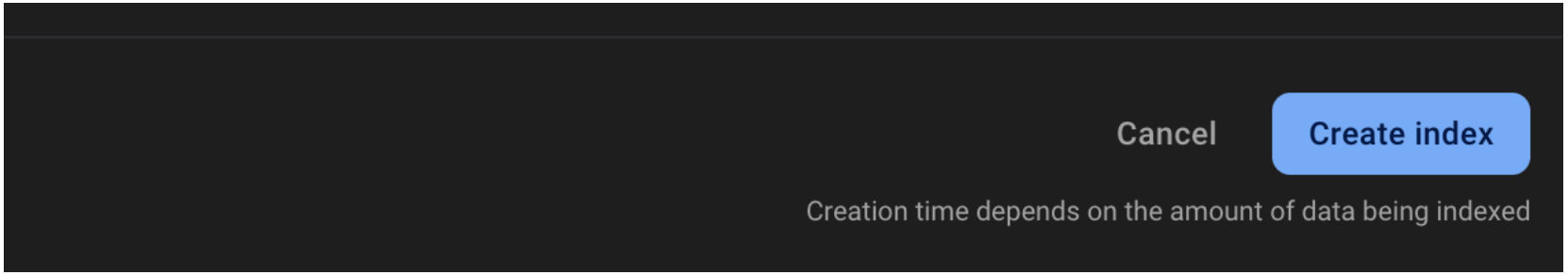
1	receiver	Ascending
2	createdAt	Descending

Add field

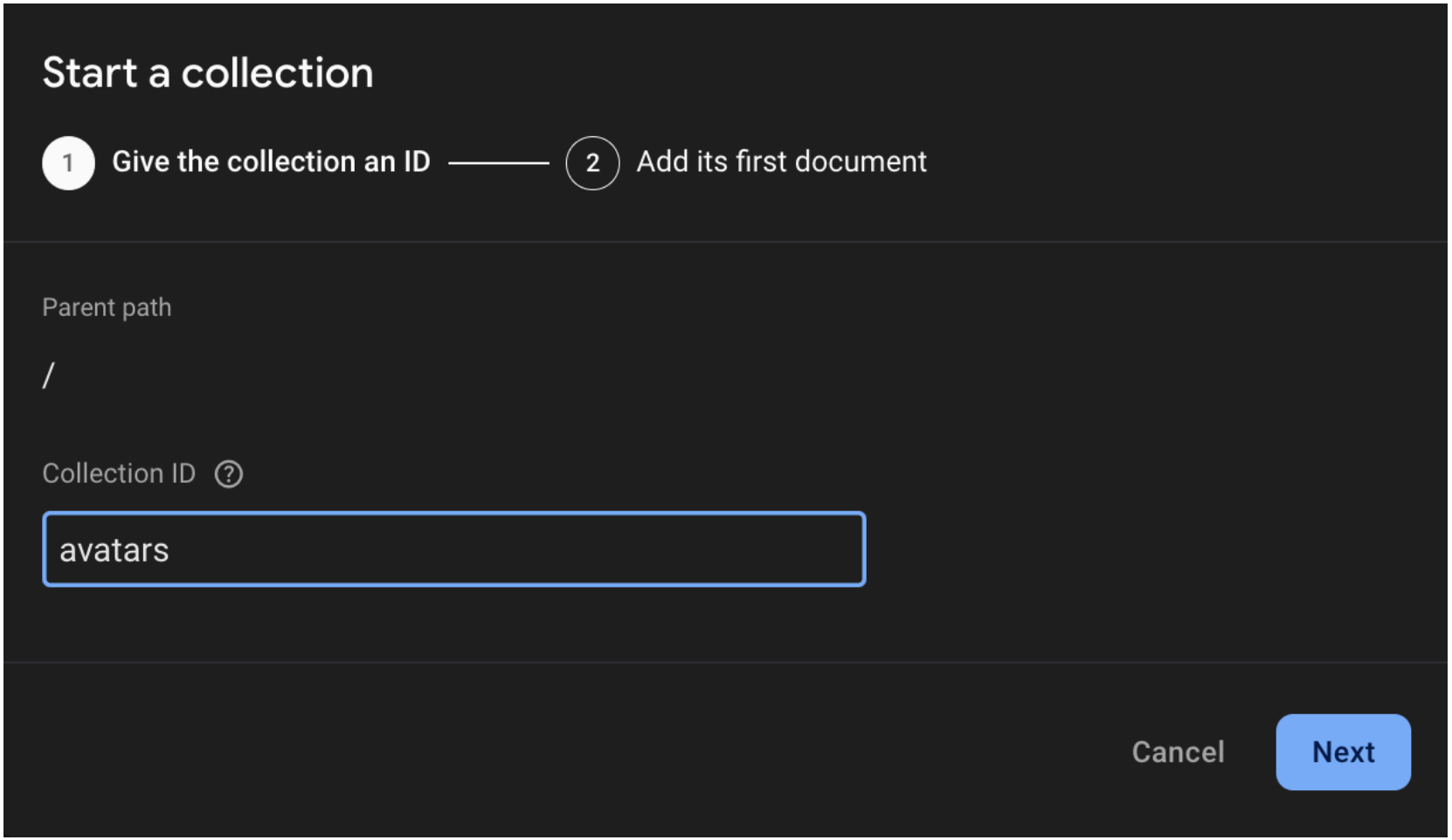
Query scopes

Collection

For queries within a specific collection path



11. Name the second collection as `avatars` in your database. This will maintain the image URL of the avatars and add dummy data with two string fields, `email` and `avatar`.

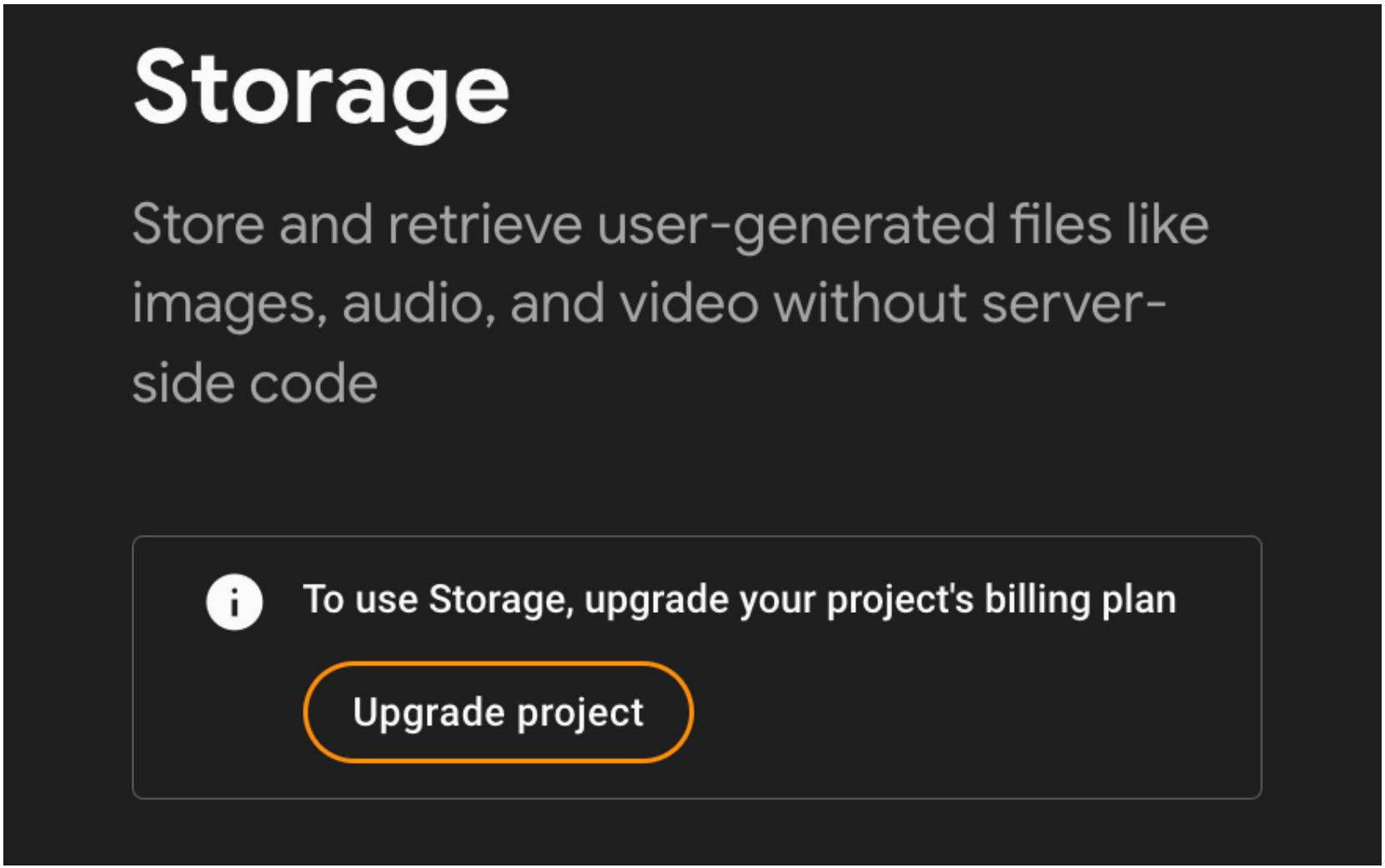


The `avatars` collection has now been created.

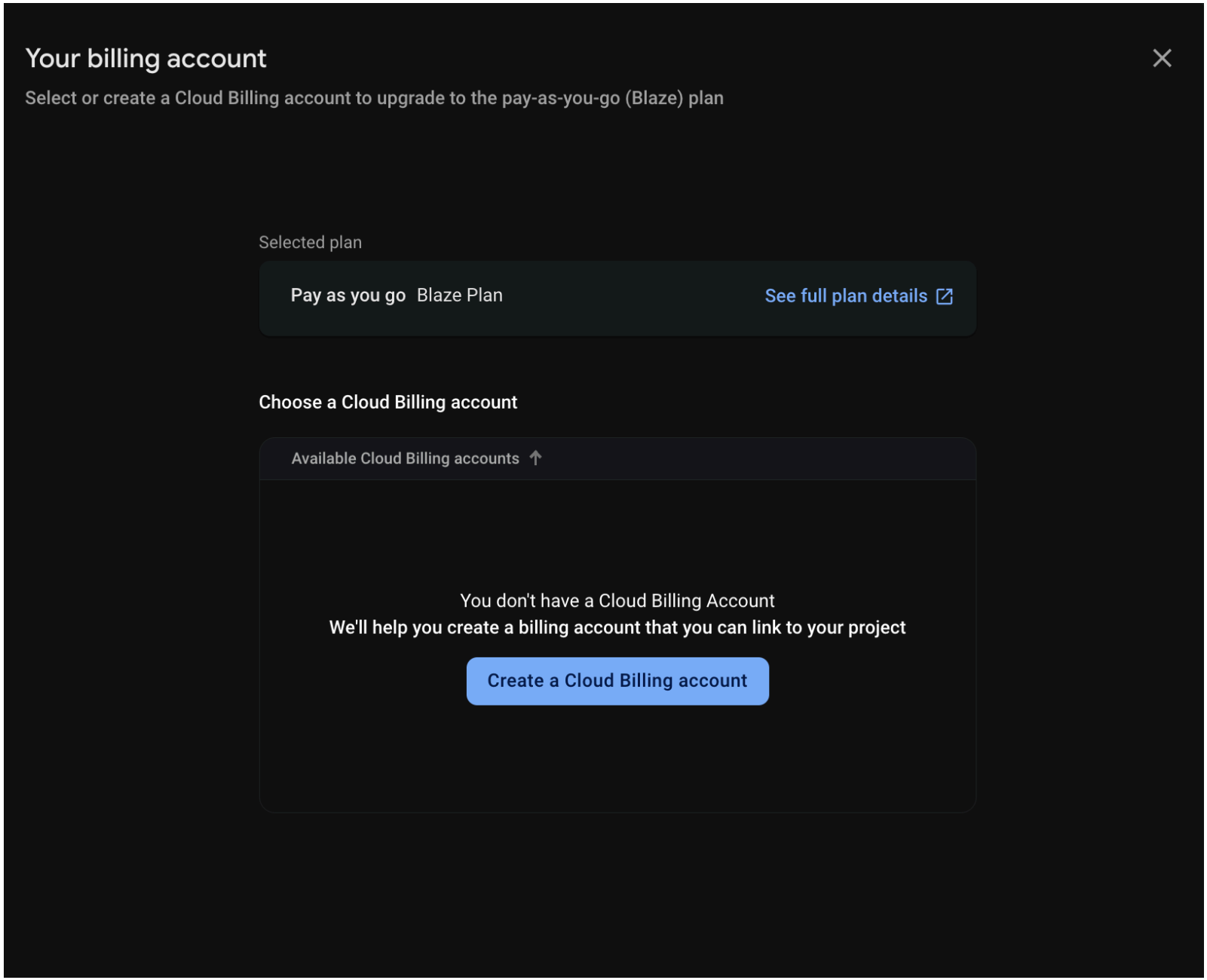
Create Firebase Storage (Optional)

If you are reusing the project from previous labs and you already have storage created, you can skip the first four steps and proceed to step 4.

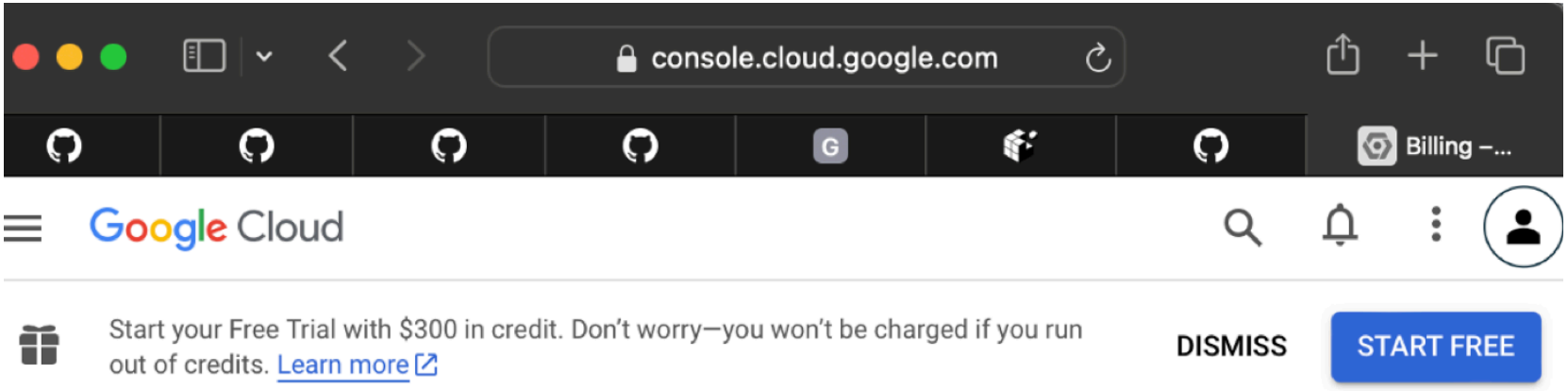
1. From the left menu on the Firebase console, under `Build` choose `Storage`. If you have already created storage for your project previously, skip the next three steps.
2. You will see a upgrade recommendation screen as below. Click on `Upgrade Project` . Google Firebase provides you with $300 credit that can be used over 90 days. You can find more details about the same in this link.



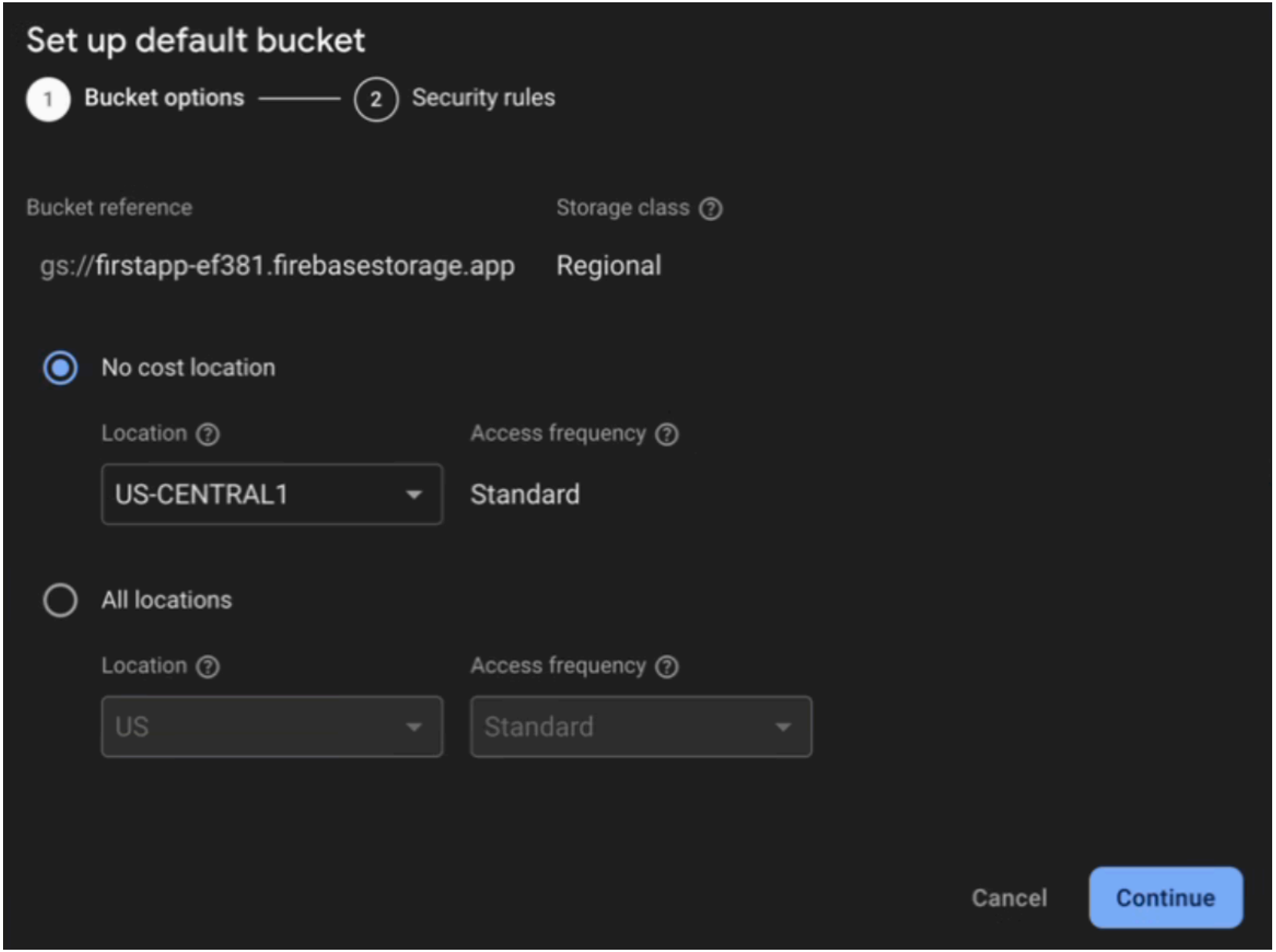
3. View the billing details by clicking on the link and then click `Create a Cloud Billing account`.



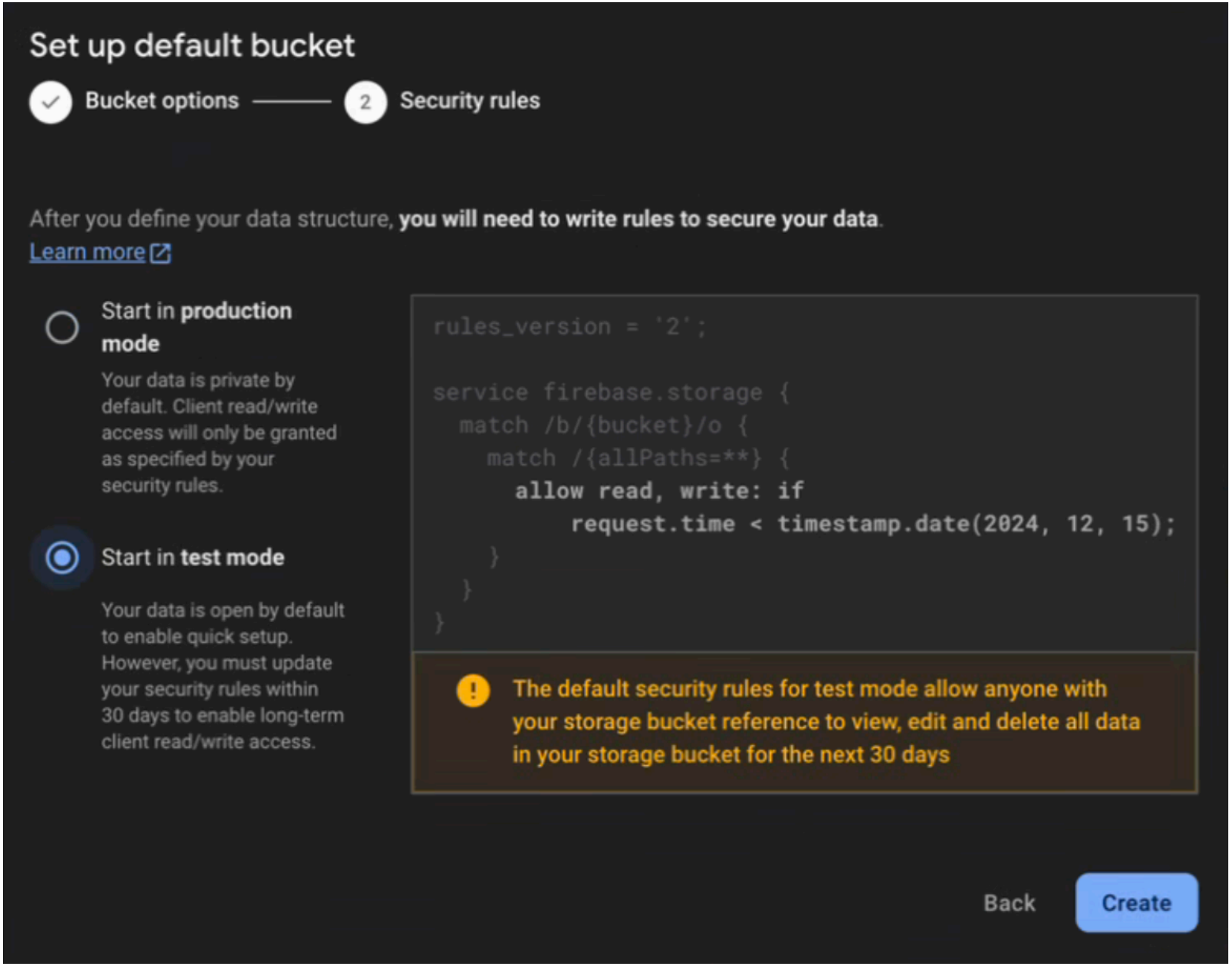
4. Click `Start Free` and follow the instructions. You will need a credit card to sign into this. But you will not be charged unless you manually authenticate it.



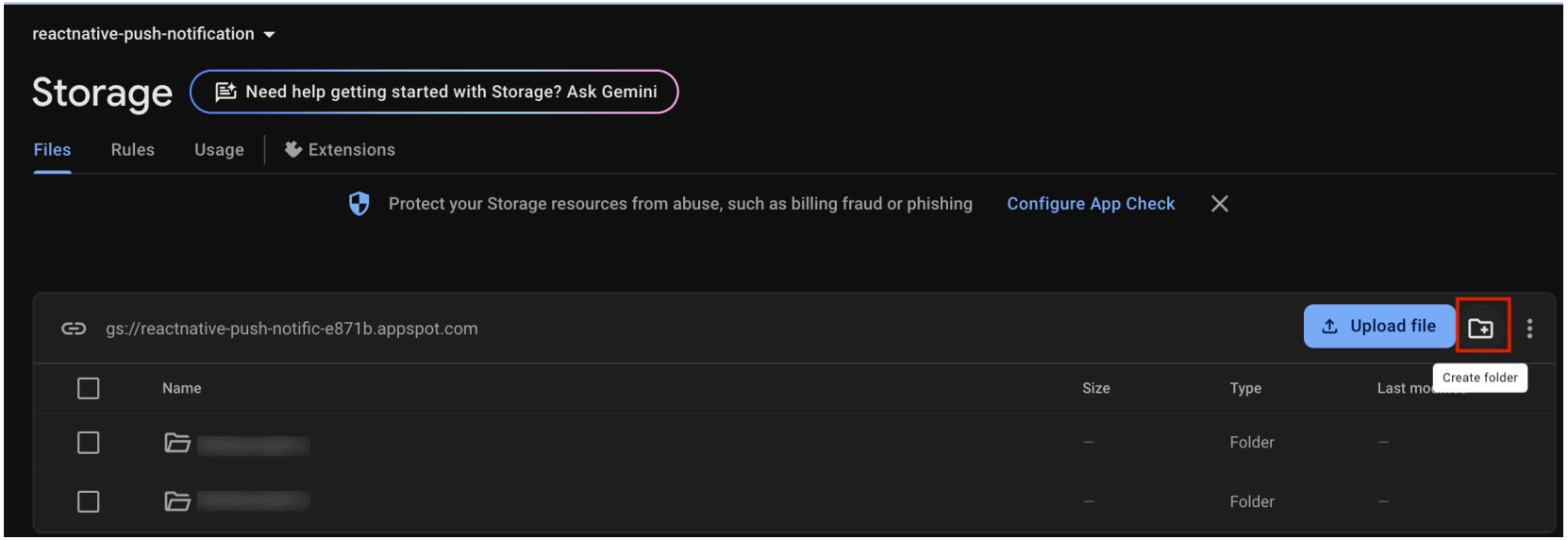
5. Allow the location to be the default chosen location based on where your Google account is created.



6. You can then choose to create the storage in test mode and click `Create`.



7. You can click the `Create folder` icon to create a new folder named `profile_pics`, where the avatar images will be stored.



If you decide to create storage, please use `SettingsScreen_withstorage.js` instead of `SettingsScreen.js`.
Now that your database and storage have been created, you can run and test the app thoroughly.

Test the app

1. Run your app from the command prompt in web mode using the following command. Once the app is rendered on the web page, right-click, choose **Inspect Element** (on Chrome), and select the mobile mode.

   ```
   npm run web
   ```

Chat App

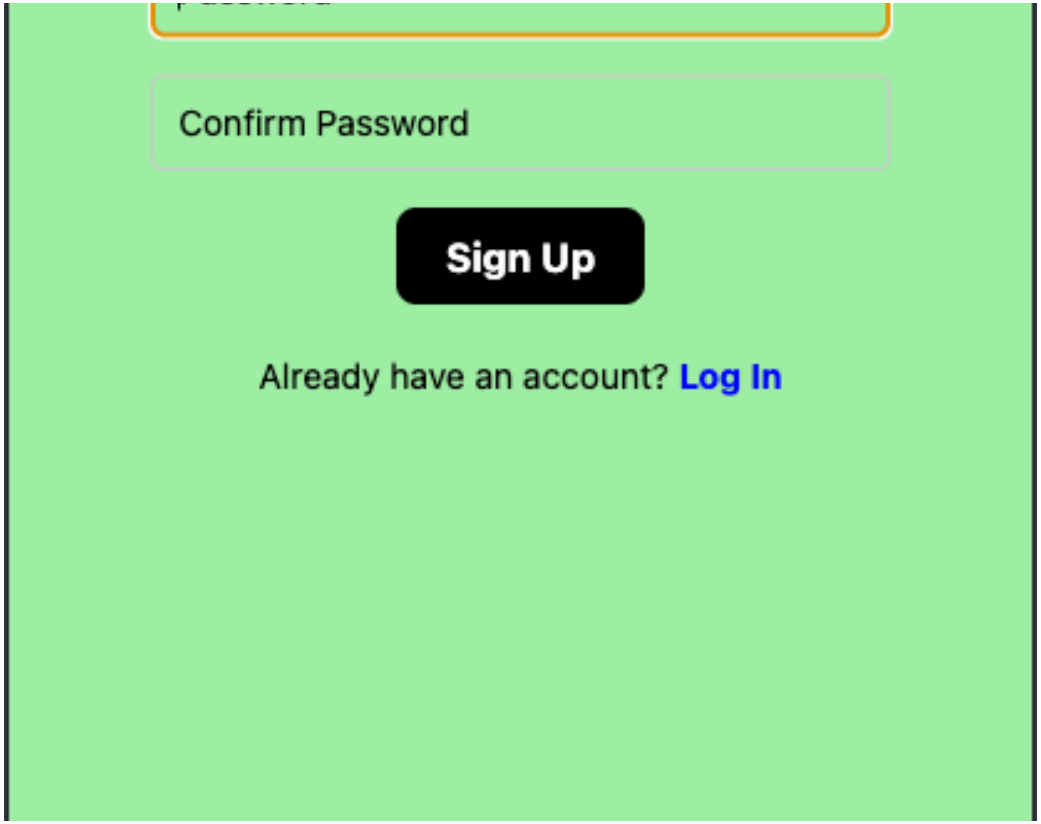
2. If your log in session is not retained, you will see the Sign Up screen where you can click `Log In` to log in.

SignUp

Let's start Buddying

Email

Password

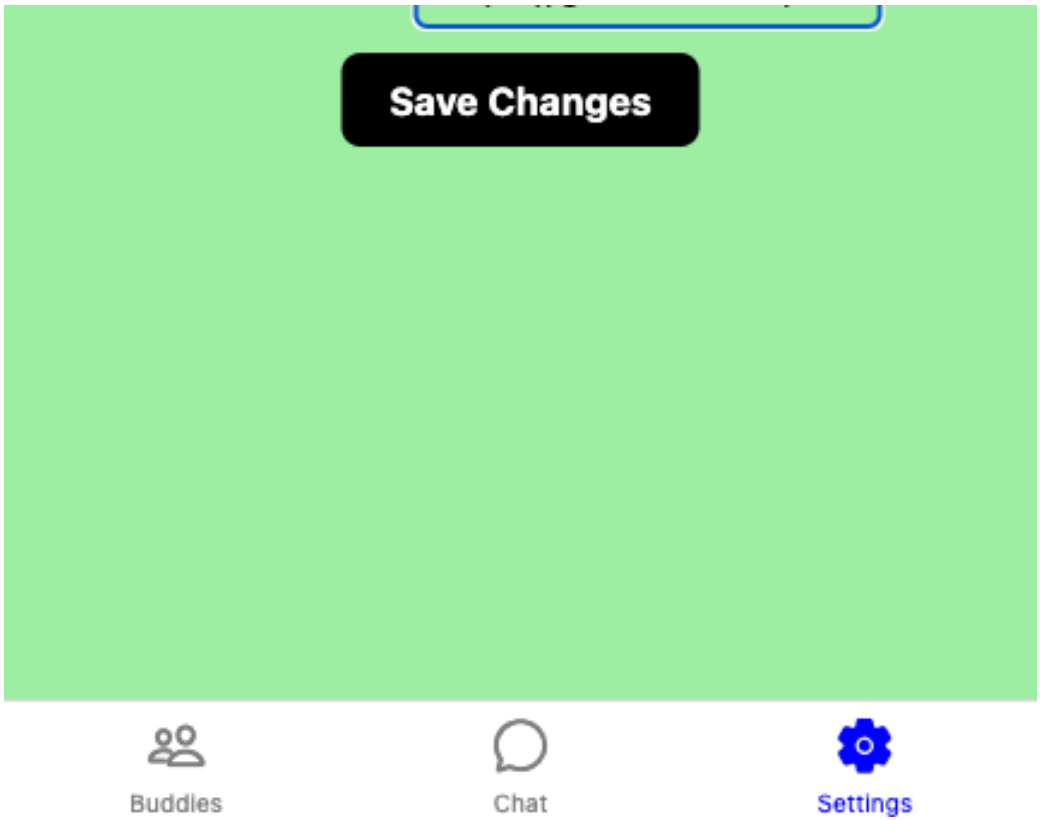


3. Choose the `Settings` tab at the bottom to add or update the avatar image. To get avatar URL you can use any online utility such as https://vinicius73.github.io/gravatar-url-generator/#/.

Settings

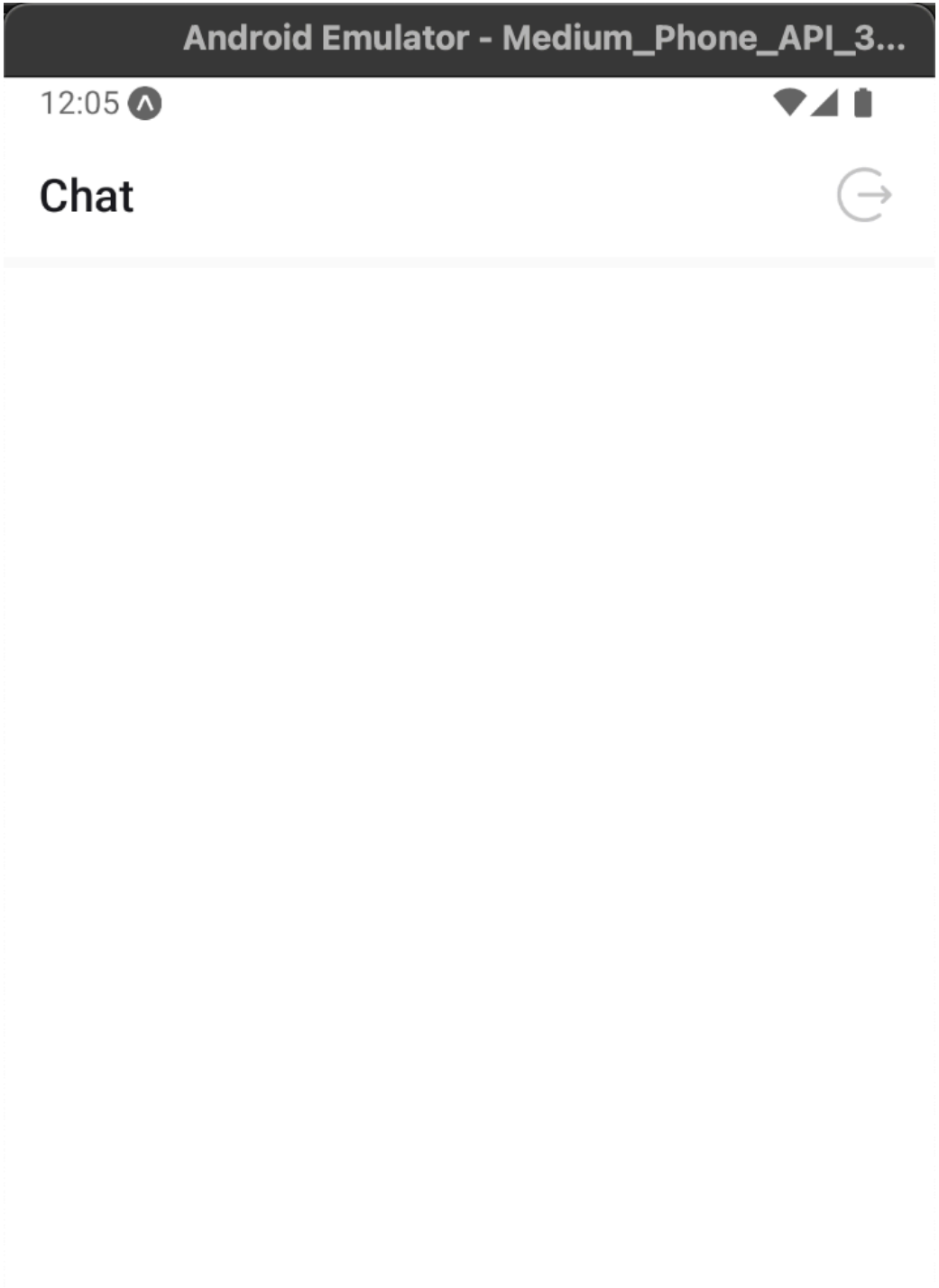
Change the avatar!

Avatar URL https://gravatar.com/ava



4. Once the avatar is added or saved, you will be taken back to the buddies screen where you can see it updated.

5. Choose a buddy to chat with, you can see the exchange of messages on the chat window, as in the image below.


Android Emulator - Medium_Phone_API_3...
12:05
Chat

Practice exercise

1. Scan the QR code on the terminal and use `Expo` on your phone to run the app.
2. Create an Android app on Firebase inside the same project. On the terminal, generate the `apk` for the app. Change the configuration file `firebase.js` as required.

   You will need to have builds left for the month in your expo account, to do this.
3. Once the QR code is generated, install and run the app on a device. Log in using one of the same users you created with the web app, and you should see your avatar and chat conversations.

Conclusion

Congratulations on completing this lab! You have now learned how to create a chat React native app with authentication, database, and storage using Firebase.

Author(s)

Lavanya T S