

# How to Build a Chat App with Firebase

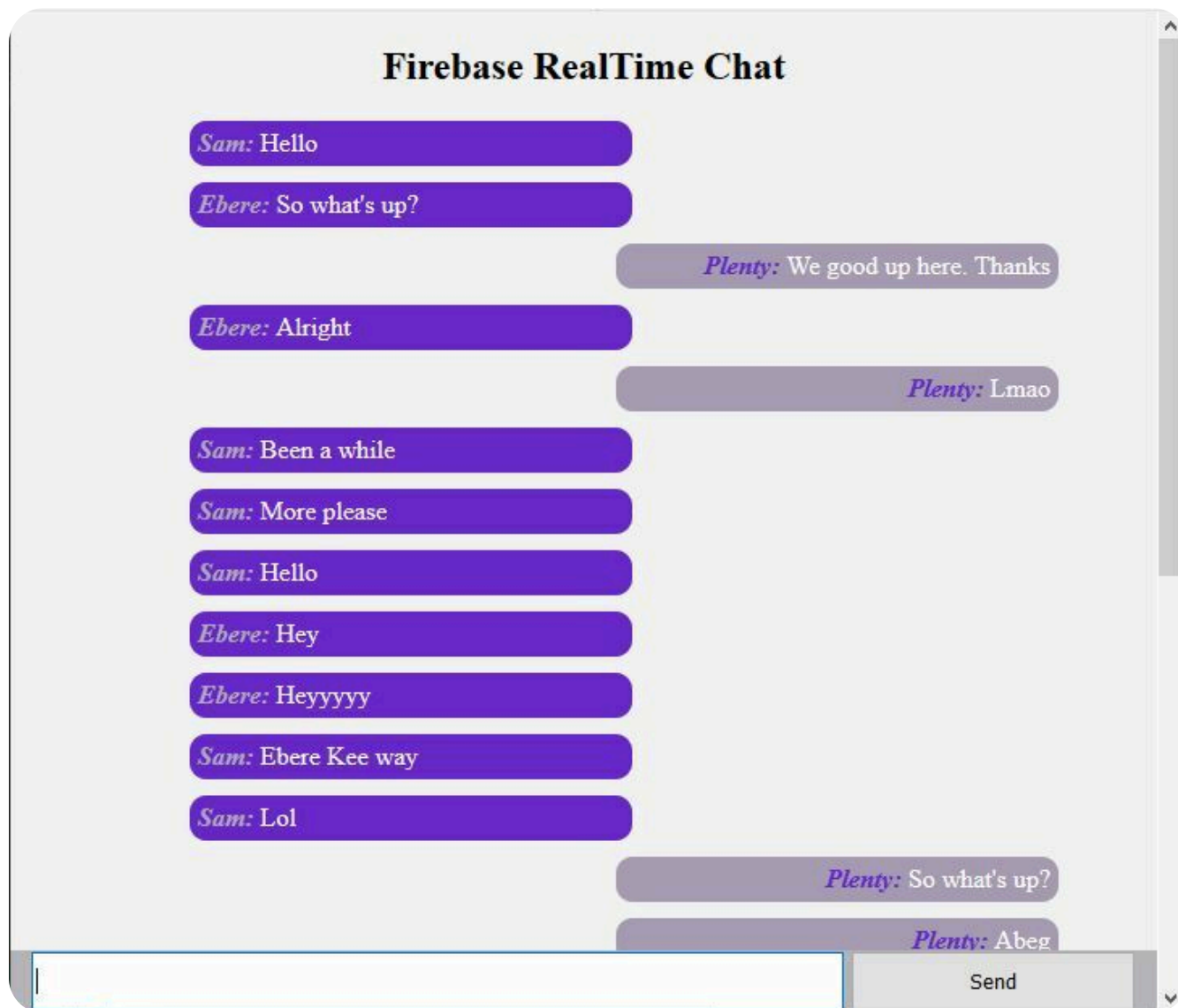
This tutorial will show you how to build your own chat application using Firebase's Realtime Database.

Njoku Samson Ebere • Mar 22, 2021



Have you thought about how difficult it is to build a chat application, showing you messages as soon as they are sent? In this tutorial we will do just that. I will guide you through building a web based chat application using Firebase as a realtime database and some simple HTML, CSS and Javascript for the interface. You can [view the source code](#) for the application on Github.

Note: If you're looking for a chat application but don't want to build your own, you should consider CometChat. Our robust suite of cloud, mobile & video solutions, ranging from simple drag-and-drop plugins to UI kits, APIs and fully customizable SDKs, plus a host of unique ready-to-use extensions, will seamlessly integrate into your website & apps, quickly and securely; saving you countless hours & resources, and dramatically growing engagement on your website. [Give us a try for free today!](#)



## What You Will Need

You will need a basic understanding of HTML, CSS and JavaScript as well as a google account to use with Firebase.

## What is Firebase?

**Firebase** is a back-end platform provided by Google for building full-stack applications. It provides programmers with authentication options, storage, databases, hosting, A/B testing and other services.

Firebase helps you to focus on developing the front-end of your applications while it does the hidden jobs for you.

[Log in](#)[Schedule a demo](#)

## Setting Up Firebase

Firebase offers two types of databases (Cloud Firestore and Realtime Database). For this tutorial, you will be using a Realtime Database.

You will need to create a new Firebase project:

- Go to the [Firebase Console](#)

- Create a new project

## Create a Realtime Database

In the sidebar on the left

- Open the **'Build'** menu

- Click on **'Realtime Database'** and then **'Create Database'**



Project Overview



Product categories

Build



Authentication



App Check



Firestore Database



Realtime Database



Extensions

NEW



Storage



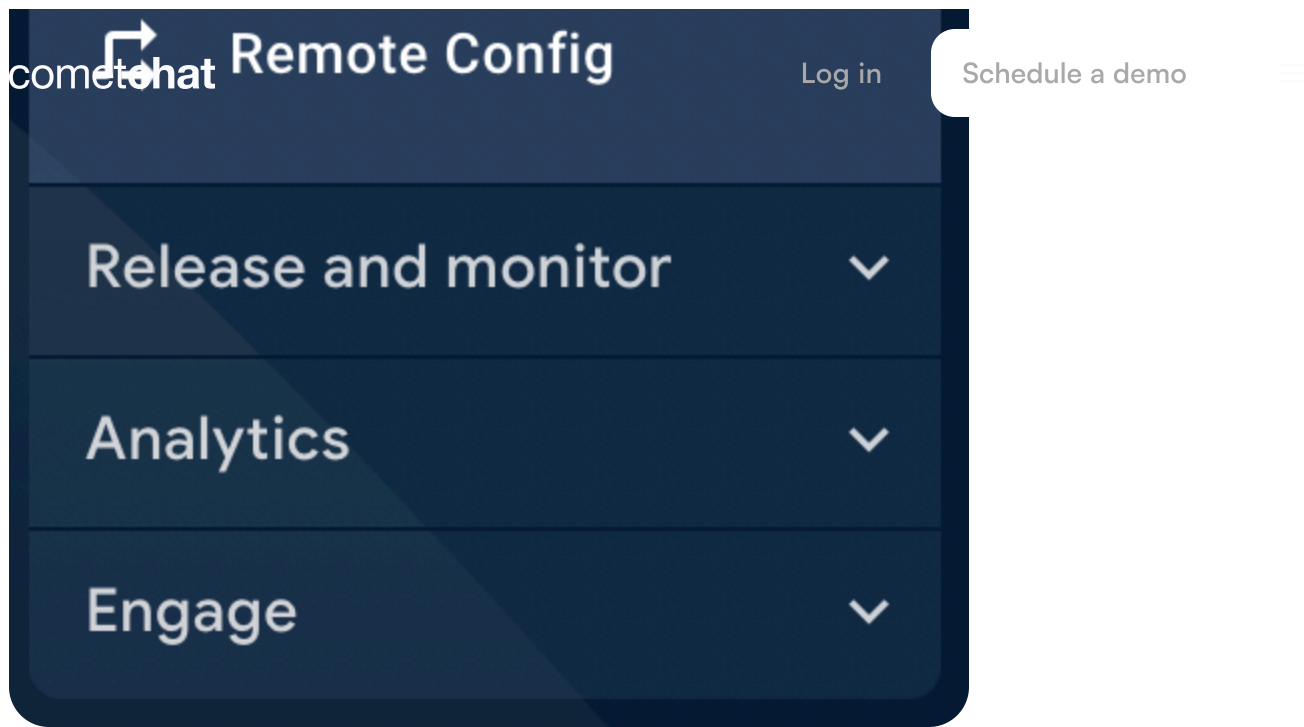
Hosting



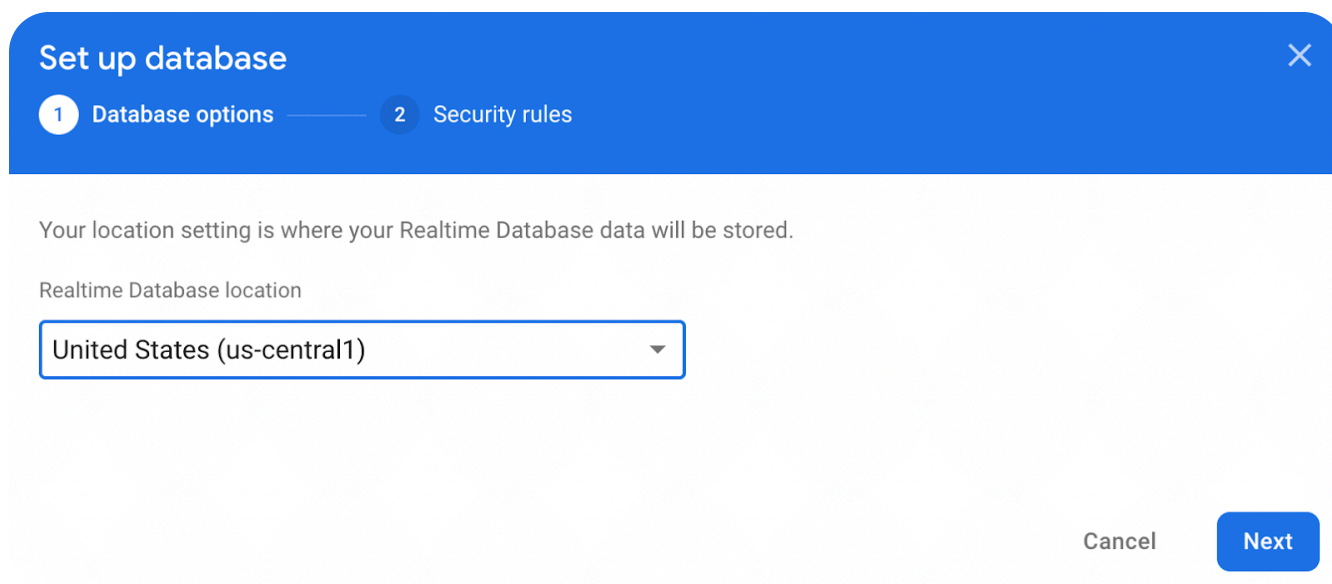
Functions



Machine Learning



After that, select the region you want to use



Finally, start the database in 'test mode'

Set up database  
cometchat

1 Database options

2 Security rules

Log in

Schedule a demo

×

☰

Once you have defined your data structure, **you will have to write rules to secure your data.**

[Learn more](#) 



#### Start in **locked mode**

Your data is private by default. Client read/write access will only be granted as specified by your security rules.



#### Start in **test mode**

Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

```
{
  "rules": {
    ".read": "now < 1670457600000", // 2022-12-8
    ".write": "now < 1670457600000", // 2022-12-8
  }
}
```



The default security rules for test mode allow anyone with your database reference to view, edit and delete all data in your database for the next 30 days

Cancel

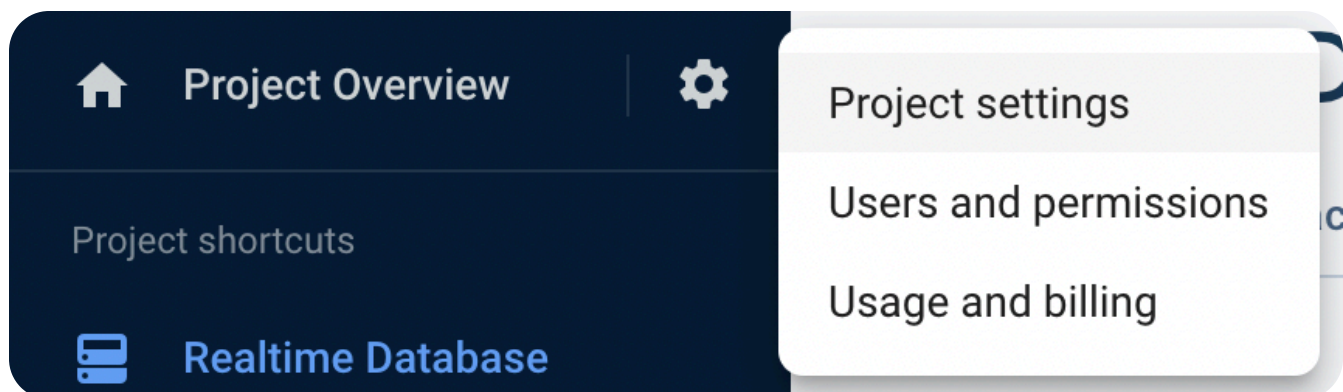
Enable

## Get Your Configuration Details

Once your database is setup, the only thing left to do is to get your configuration details. You will use these in order to authenticate your application with the Firebase service.

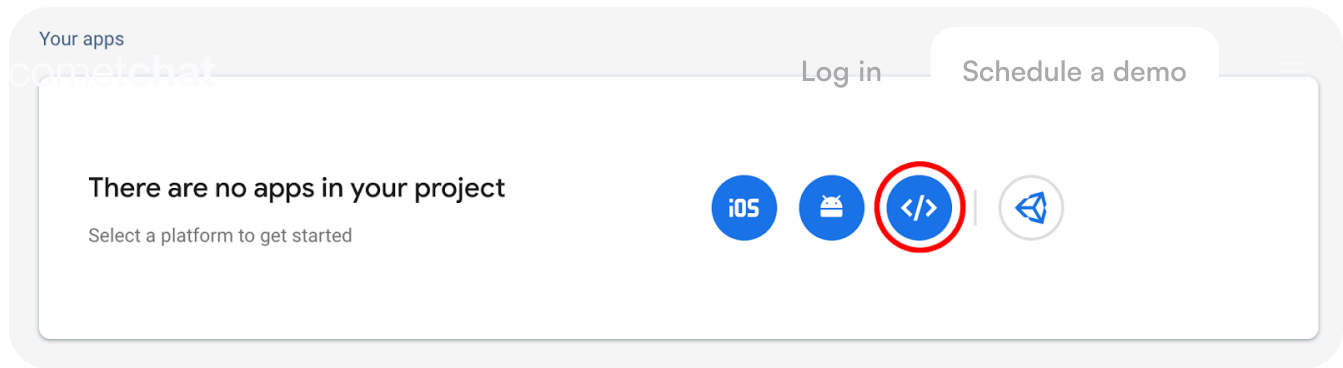
You can find your configuration details in the Firebase console:

- Click the Gear icon next to 'Project Overview'
- Choose 'Project Settings'



Under 'Your apps', select 'Web app':





Next you need to register your application:

- Enter a name for your application
- Click on **'Register App'**

You can ignore the next screen and click on **'Continue to Console'**

Your application's config values will then be visible in the **'Your apps'** section

#### Firebase SDK snippet

☐ Automatic <sup>?</sup> ☐ CDN <sup>?</sup> ☒ Config <sup>?</sup>

Copy and paste these scripts into the bottom of your `<body>` tag, but before you use any Firebase services:

```
const firebaseConfig = {
  apiKey: XXXXXXXXXXXXXXXXXXXX,
  authDomain: XXXXXXXXXXXXXXXXXXXX,
  databaseURL: XXXXXXXXXXXXXXXXXXXX,
  projectId: XXXXXXXXXXXXXXXXXXXX,
  storageBucket: XXXXXXXXXXXXXXXXXXXX,
  messagingSenderId: XXXXXXXXXXXXXXXXXXXX,
  appId: XXXXXXXXXXXXXXXXXXXX,
  measurementId: XXXXXXXXXXXXXXXXXXXX
};
```



## Project Setup

Your project will consist of 3 files, create a folder with the 3 files named:

- index.html
- index.css
- index.js

cometchat

[Log in](#)[Schedule a demo](#)

## Laying The Foundation

The index.html file will contain the HTML for your application as well as links to the index.css, and index.js files, as well as the Firebase SDK.

Add the following code to your **index.html** file:

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width,
6        initial-scale=1" />
7      <title>Firebase RealTime Chat</title>
8      <link rel="stylesheet" href="./index.css">
9    </head>
10   <body>
11     <!-- scripts -->
12     <script
13       src="https://www.gstatic.com/firebasejs/8.2.1/firebase-
14       app.js"></script>
15     <script
16       src="https://www.gstatic.com/firebasejs/8.2.1/firebase-
17       database.js"></script>
18     <script src="index.js"></script>
19   </body>
20 </html>
```

Notice that the script tags for the Firebase SDKs and your index.js file are included in the **<body>** element. These load the Firebase SDK and make it available to your application. The index.css file is also included in the **<head>** element and will be used to style the application.

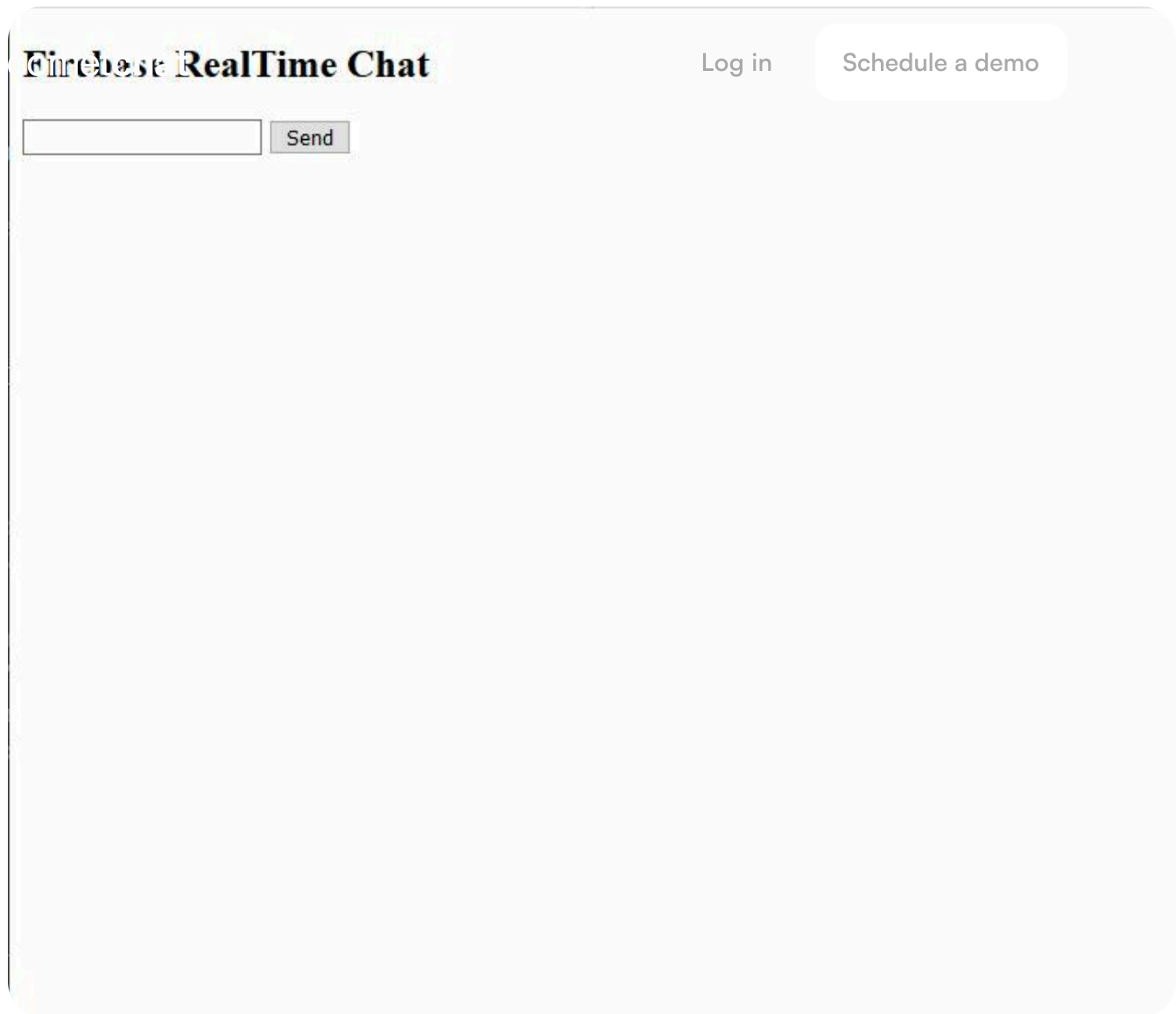
Next, Add the following content to the **<body>** element above the **<script>** tags:



```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width,
6      initial-scale=1" />
7      <title>Firebase RealTime Chat</title>
8      <link rel="stylesheet" href="./index.css">
9    </head>
10   <body>
11     <header>
12       <h2>Firebase RealTime Chat</h2>
13     </header>
14     <div id="chat">
15       <!-- messages will display here -->
16       <ul id="messages"></ul>
17
18       <!-- form to send message -->
19       <form id="message-form">
20         <input id="message-input" type="text" />
21         <button id="message-btn" type="submit">Send</button>
22       </form>
23     </div>
24     <!-- scripts -->
25     <script
26       src="https://www.gstatic.com/firebasejs/8.2.1/firebase-
27       app.js"></script>
28     <script
29       src="https://www.gstatic.com/firebasejs/8.2.1/firebase-
30       database.js"></script>
31     <script src="index.js"></script>
32   </body>
33 </html>
```

In the HTML above, you created a *<header>* with the name of your app. The *<ul>* element will be used to display messages loaded from the server while the *<form>* element will enable users to submit their messages.

If you open your application in a browser now, it should look something like this:



## Adding Functionality

You now need to open the `index.js` file. This is where we will build out the functionality of our application step by step.

## Initializing the Application

You need to tell your application how to connect to the Realtime Database we created and initialize the Firebase application.

Copy the *firebaseConfig* for your application from the Firebase Project Console and paste it into your `index.js` file. Then add the following lines of code:

```
1  var firebaseConfig = {  
2    apiKey: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",  
3    authDomain: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",  
4    databaseURL: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
```

```

5     projectId: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
6     storageBucket: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
7     messagingSenderId: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
8     appId: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
9   };
10
11   firebase.initializeApp(firebaseConfig);
12
13   const db = firebase.database();
14
15   const username = prompt("Please Tell Us Your Name");

```

## Sending Messages

In order to send messages, users will type into the input form provided and click on the 'Send' button. Your app will subscribe to the *submit* event when the 'Send' button is clicked and then call the *sendMessage()* function.

The *sendMessage()* function will:

- Prevent the default form behaviour
- Capture and store the message sent by the user
- Clear the input box
- Scroll to the bottom of the page where the new message will appear
- Add the message to the database

Edit your *index.js* file to contain the following code:

```

1   var firebaseConfig = {
2     apiKey: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
3     authDomain: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
4     databaseURL: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
5     projectId: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
6     storageBucket: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
7     messagingSenderId: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
8     appId: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
9   };

```

```
10
11 firebase.initializeApp(firebaseConfig);
12
13 const db = firebase.database();
14
15 const username = prompt("Please Tell Us Your Name");
16
17 function sendMessage(e) {
18     e.preventDefault();
19
20     // get values to be submitted
21     const timestamp = Date.now();
22     const messageInput = document.getElementById("message-
input");
23     const message = messageInput.value;
24
25     // clear the input box
26     messageInput.value = "";
27
28     //auto scroll to bottom
29     document
30         .getElementById("messages")
31         .scrollIntoView({ behavior: "smooth", block: "end",
inline: "nearest" });
32
33     // create db collection and send in the data
34     db.ref("messages/" + timestamp).set({
35         username,
36         message,
37     });
38 }
```

## Receiving Text Messages

When a message is sent by one user it needs to be received by the other users of the application.

Edit your index.js file to contain the following code:

```
1 var firebaseConfig = {
```

```
2     apiKey: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xx",
3     authDomain: "xxx-xxx-xxx-xxx-xxx-xxx-xx",
4     databaseURL: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
5     projectId: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
6     storageBucket: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
7     messagingSenderId: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
8     appId: "xxx-xxx-xxx-xxx-xxx-xxx-xxx-xxx",
9   };
10
11   firebase.initializeApp(firebaseConfig);
12
13   const db = firebase.database();
14
15   const username = prompt("Please Tell Us Your Name");
16
17   function sendMessage(e) {
18     e.preventDefault();
19
20     // get values to be submitted
21     const timestamp = Date.now();
22     const messageInput = document.getElementById("message-
input");
23     const message = messageInput.value;
24
25     // clear the input box
26     messageInput.value = "";
27
28     //auto scroll to bottom
29     document
30       .getElementById("messages")
31       .scrollIntoView({ behavior: "smooth", block: "end",
inline: "nearest" });
32
33     // create db collection and send in the data
34     db.ref("messages/" + timestamp).set({
35       username,
36       message,
37     });
38   }
39
```

```

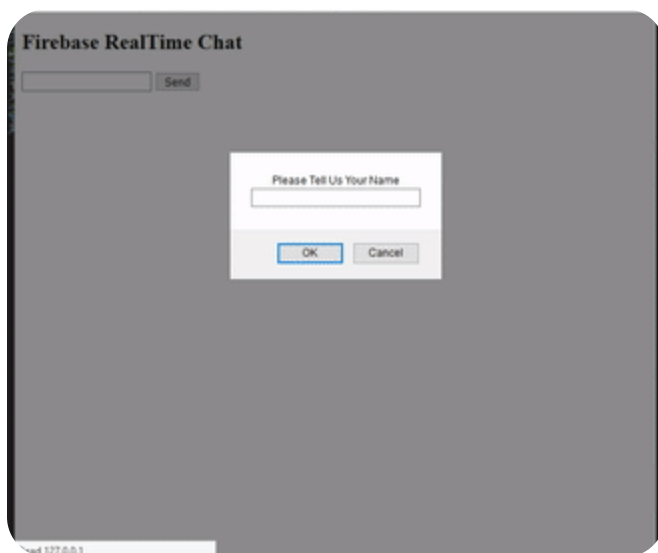
40  const fetchChat = db.ref("messages/");
41  fetchChat
42  fetchChat.on("child_added", function (snapshot) {
43    const messages = snapshot.val();
44    const message = `<li class=${
45      username === messages.username ? "sent" : "receive"
46    }><span>${messages.username}: </span>${messages.message}
    </li>`;
47    // append the message on the page
48    document.getElementById("messages").innerHTML += message;
49  });

```

In the code above, you subscribe to the database's *child\_added* event in order to be notified when new messages are added to the database. When a new message is added, it is appended to the `<ul>` element you created earlier as a new list item.

At this point, if you open your application in a browser it should be possible to:

- Specify a username
- Send and receive messages



## Chat App Demo Without CSS

## Styling The App

To style your application you need to add some CSS to the `index.css` file. Feel free to copy and paste the styles below to get you started:

```
1 body {
2     background-color: #f3f2f3;
3     margin: 0;
4     padding: 0;
5 }
6
7 header {
8     text-align: center;
9 }
10
11 #messages{
12     padding-bottom: 30%;
13 }
14
15 li {
16     list-style-type: none;
17     margin-bottom: 10px;
18     background-color: #6929ca;
19     padding: 5px;
20     border-radius: 10px;
21     color: white;
22     width: 50%;
23 }
24
25 li span {
26     font-style: italic;
27     font-weight: bolder;
28     color: #b5b0b9;
29 }
30
31 #chat {
32     width: 80%;
33     margin: auto;
34 }
35
36 #message-form {
37     text-align: center;
38     position: fixed;
39     left: 0;
40     bottom: 0;
```

[Log in](#)[Schedule a demo](#)



```
41     width: 100%;
42     background-color: #b7b5b9;
43 }
44
45 input {
46     width: 70%;
47     height: 30px;
48 }
49
50 button {
51     width: 25%;
52     height: 38px;
53 }
54
55 .sent {
56     text-align: right;
57     background-color: #a79cb3;
58     margin-left: 50%;
59 }
60
61 .sent span {
62     margin-left: 5px;
63     color: #6929ca;
64 }
```

## Conclusion

You have now built a simple realtime chat application using the Firebase SDK and Realtime Database as a datastore.

If you're looking for a robust application that's built to scale, consider CometChat. You can [create an account for free](#), and start building right away. Develop for as long as you need, and don't pay a thing until you're ready to scale.



**Njoku Samson Ebere**  
CometChat

He is a software engineer who is interested in building solutions to real world problems and teaching others about the things he knows. Something he

really enjoys doing aside writing codes and technical articles is building

[Log in](#)[Schedule a demo](#)

Share it with everyone!

 Facebook

 Twitter

 LinkedIn

## Try out CometChat in action

Experience CometChat's messaging with this interactive demo built with CometChat's UI kits and SDKs.

[Try the interactive demo](#)

cometchat

### Platform

Features

Chat & Messaging

Voice & Video Calls

Implementation

### Solutions

By use cases

Online Marketplaces

SaaS Businesses

Healthcare

Team Comms & Workflows

[UI Kits](#)[cometchat](#)[SDKs & APIs](#)[Widgets](#)[Sports & Gam'](#)[Log in](#)[Schedule a demo](#)[Online educat...](#)[Events & streaming](#)[Community & Social](#)[Dating](#)

## Developers

[Supported Technologies](#)[React Chat SDK](#)[React Native Chat SDK](#)[Android Chat SDK](#)[iOS Swift Chat SDK](#)[Flutter Chat SDK](#)[Angular Chat SDK](#)[Kotlin Chat SDK](#)[Vue Chat SDK](#)[Ionic Chat SDK](#)[PHP Chat Widget](#)[Wordpress Chat Widget](#)[Laravel Chat Widget](#)

## Resources

[Blog](#)[Tutorials](#)[Help Center](#)[Open-source Apps](#)[Product Status](#)[Comparison guides](#)[Best chat SDKs](#)[Best video calling SDKs](#)[Best sendbird alternative](#)[Best streamchat alternative](#)[Best pubnub alternative](#)

## Company

[Careers](#)[Partners](#)[Pricing](#)

Chat with us  
cometchat

Log in

Schedule a demo



2024 © CometChat

[Terms of service](#)

[Privacy Policy](#)

 [Facebook](#)

 [LinkedIn](#)

[Data Processing Addendum](#)

[Sub-processors List](#)

 [Youtube](#)

 [Twitter](#)

 [Github](#)