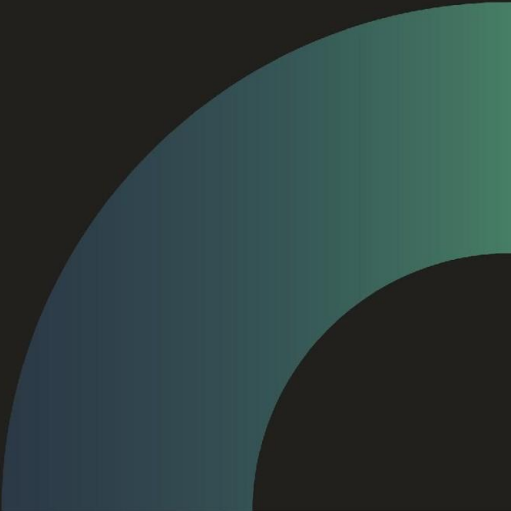


Full Stack Web Development

Supplementary material Firebase

Outline

-
- Social media login with Firebase
 - Push notification with Firebase
- 

Google Firestore is a Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firestore provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.



Firestore offers a number of services, including:

- Analytics – Google Analytics for Firestore offers free, unlimited reporting on as many as 500 separate events. Analytics presents data about user behavior in iOS and Android apps, enabling better decision-making about improving performance and app marketing.
- Authentication – Firestore Authentication makes it easy for developers to build secure authentication systems and enhances the sign-in and onboarding experience for users. This feature offers a complete identity solution, supporting email and password accounts, phone auth, as well as Google, Facebook, GitHub, Twitter login and more.
- Cloud messaging – Firestore Cloud Messaging (FCM) is a cross-platform messaging tool that lets companies reliably receive and deliver messages on iOS, Android and the web at no cost.
- Realtime database – the Firestore Realtime Database is a cloud-hosted NoSQL database that enables data to be stored and synced between users in real time. The data is synced across all clients in real time and is still available when an app goes offline.



- Crashlytics – Firebase Crashlytics is a real-time crash reporter that helps developers track, prioritize and fix stability issues that reduce the quality of their apps. With crashlytics, developers spend less time organizing and troubleshooting crashes and more time building features for their apps.
- Performance – Firebase Performance Monitoring service gives developers insight into the performance characteristics of their iOS and Android apps to help them determine where and when the performance of their apps can be improved.
- Test lab – Firebase Test Lab is a cloud-based app-testing infrastructure. With one operation, developers can test their iOS or Android apps across a variety of devices and device configurations. They can see the results, including videos, screenshots and logs, in the Firebase console.

Social Media Login with Firebase

Setting up Firebase – navigate your browser to Firebase Console. Make sure you are logged into your Google account.

Click on Add project and you should be presented with the following screen:

× Create a project (Step 1 of 3)

Let's start with a name for your project[?]

Project name

materi-pwdk

✎ materi-pwdk

☒ I confirm that I will use Firebase exclusively for purposes relating to my trade, business, craft, or profession.

Continue

× Create a project (Step 2 of 2)

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.

Google Analytics enables:

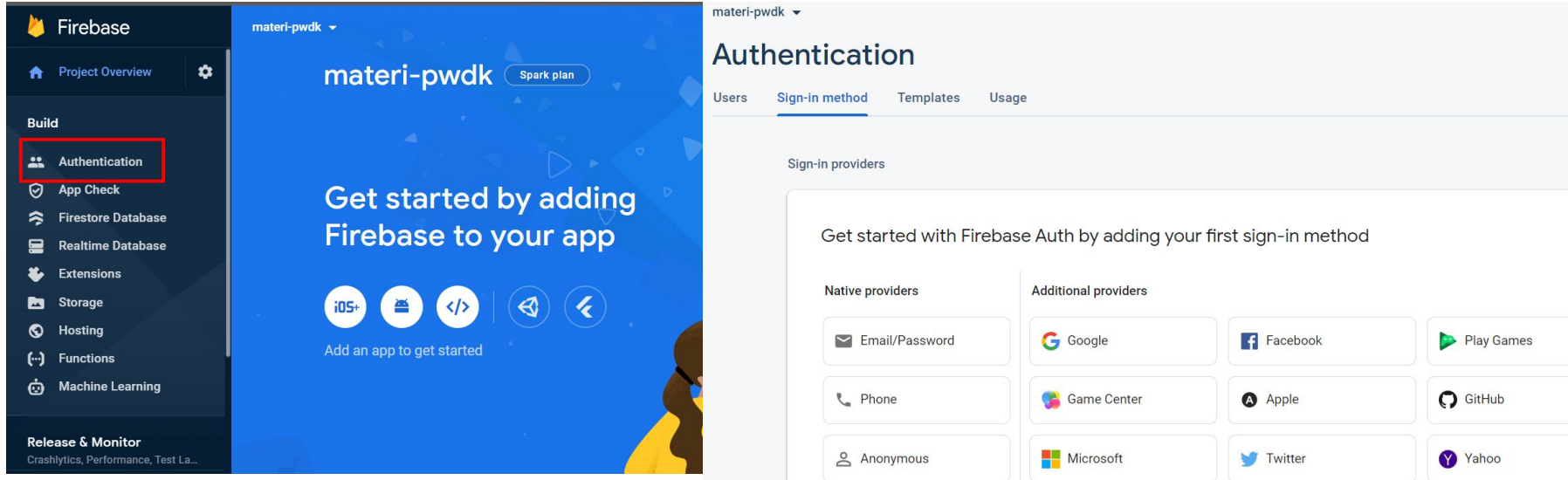
- × A/B testing[?]
- × User segmentation & targeting across Firebase products[?]
- × Crash-free users[?]
- × Event-based Cloud Functions triggers[?]
- × Free unlimited reporting[?]

☒ **Enable Google Analytics for this project**
Recommended

[Previous](#) **Create project**

Social Media Login with Firebase

Click on Authentication on the sidebar and click on Get Started to enable the module. Now you will be presented with various authentication options:





The screenshot displays the Firebase console interface. On the left, the 'Build' sidebar is visible, with 'Authentication' highlighted by a red rectangle. The main content area shows the 'materi-pwdk' project dashboard with a 'Get started by adding Firebase to your app' message. To the right, the 'Authentication' section is open, showing the 'Sign-in method' tab. Under 'Sign-in providers', there is a heading 'Get started with Firebase Auth by adding your first sign-in method'. Below this, two columns of provider options are shown: 'Native providers' and 'Additional providers'.

Native providers	Additional providers		
Email/Password	Google	Facebook	Play Games
Phone	Game Center	Apple	GitHub
Anonymous	Microsoft	Twitter	Yahoo


Social Media Login with Firebase


Login through Email & Password, click on enable email/password and hit save button to applies the changes

Sign-in providers

 Email/Password  Enable

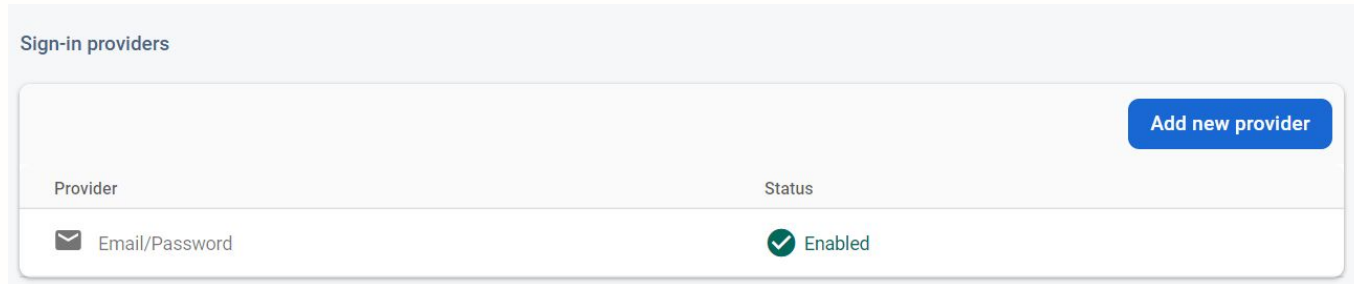
Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives. [Learn more](#)

Email link (passwordless sign-in)  Enable

Cancel 

Social Media Login with Firebase

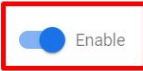
As you can see, we already enable sign-in methods using email and password. Now lets click on **add new provider** button to add another method. In this case lets click on google



Social Media Login with Firebase



Google




Google sign-in is automatically configured on your connected Apple and web apps. To set up Google sign-in for your Android apps, you need to add the [SHA1 fingerprint](#) for each app on your [Project Settings](#).

⚙️ Update the [project-level setting](#) below to continue

Project public-facing name ⓘ

project-642799249883

Project support email ⓘ



Safelist client IDs from external projects (optional) ⓘ



Web SDK configuration ⓘ



Enable the switch button and set email address as project support email. Click on save button to keep the changes.

Cancel

Save

Social Media Login with Firebase

Create database

- 1 Secure rules for Cloud Firestore — 2 Set Cloud Firestore location

After you define your data structure, you will need to write rules to secure your data.

[Learn more](#)

☐ Start in **production 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_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if
        request.time < timestamp.date(2022, 7, 21);
    }
  }
}
```

! 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

Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project, notably from the associated App Engine app

Cancel

Next

Now, let's set up the database we are going to use Cloud Firestore. Click on Firestore Database on the sidebar and click on Create Database.

Remember to select Start in test mode. Production mode databases require a configuration of security rules, which is out of the scope of this tutorial.

Click Next. Select the region. This should completely set up your Cloud Firestore database.

Integrating Firebase into React app

Open your react app projects, and install this several packages.

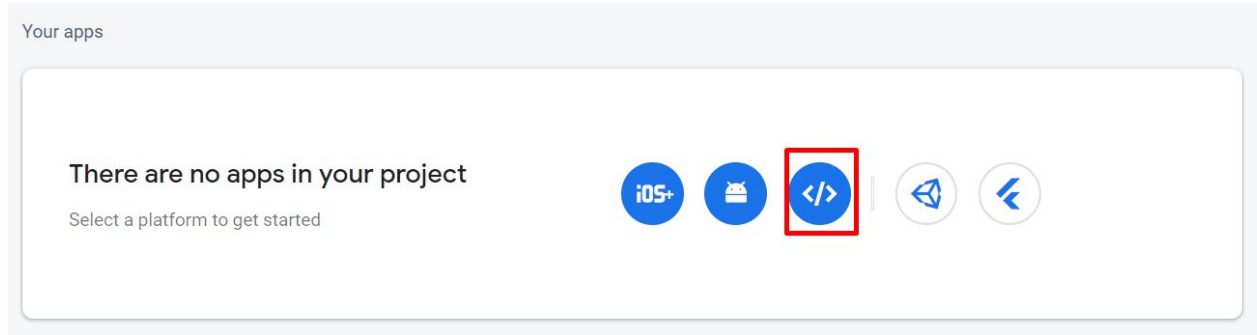
install:

`npm install firebase react-router-dom react-firebase-hooks`

we are installing firebase to communicate with Firebase services, and we are also installing react-router-dom to handle the routing of the application. We use react-firebase-hooks to manage the authentication state of the user.


Integrating Firebase into React app


Go to your Firebase Console dashboard, click on Project Settings, scroll down, and you should see something like this and click on the third icon from the left:





Integrating Firebase into React app


Enter the app name and click on Continue.

 Add Firebase to your web app

 Register app


 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:

```
$ npm install firebase
```



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

Integrating Firebase into React app

SDK setup and configuration

☒ npm  ☐ CDN  ☐ Config 

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:

```
$ 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 = {
```

Go back to the project settings and you should now see a config like this.

Copy the config. Create a new file in the src folder named firebase.js

Integrating Firebase into React app

Clone this repo that contain Login.js, Register.js, Dashboard.js component:

<https://github.com/hasianamin/firebase-auth.git>

Install all dependencies and start the projects.

Integrating Firebase into React app

```
src > JS firebase.js > ...
1  import { initializeApp } from "firebase/app";
2  import {
3      GoogleAuthProvider,
4      getAuth,
5      signInWithPopup,
6      signInWithEmailAndPassword,
7      createUserWithEmailAndPassword,
8      sendPasswordResetEmail,
9      signOut,
10 } from "firebase/auth";
11 import {
12     getFirestore,
13     query,
14     getDocs,
15     collection,
16     where,
17     addDoc,
18 } from "firebase/firestore";
```

Let's first checkout firebase.js.

Import firebase modules, since Firebase uses modular usage in v9

Integrating Firebase into React app

Now paste in the config we just copied. Let's initialize our app and services so that we can use Firebase throughout our app:

```
21  const firebaseConfig = {  
22    apiKey: 'YOUR_API_KEY',  
23    authDomain: 'YOUR_AUTH_DOMAIN',  
24    projectId: 'YOUR_PROJECT_ID',  
25    storageBucket: 'YOUR_STORAGE_BUCKET',  
26    messagingSenderId: 'YOUR_MESSAGING_SENDER_ID',  
27    appId: 'YOUR_APP_ID',  
28  };  
29  
30  const app = initializeApp(firebaseConfig);  
31  const auth = getAuth(app);  
32  const db = getFirestore(app);
```

Integrating Firebase into React app

```
31
32 const googleProvider = new GoogleAuthProvider();
33 const signInWithGoogle = async () => {
34   try {
35     const res = await signInWithPopup(auth, googleProvider);
36     const user = res.user;
37     const q = query(collection(db, "users"), where("uid", "==", user.uid));
38     const docs = await getDocs(q);
39     if (docs.docs.length === 0) {
40       await addDoc(collection(db, "users"), {
41         uid: user.uid,
42         name: user.displayName,
43         authProvider: "google",
44         email: user.email,
45       });
46     }
47   } catch (err) {
48     console.error(err);
49     alert(err.message);
50   }
51 };
```

We will be creating all important authentication-related functions in firebase.js itself. So first look at the Google Authentication function:

Integrating Firebase into React app

Now let's make a function for signing in using an email and password:

```
53   const loginWithEmailAndPassword = async (email, password) => {  
54     try {  
55       await signInWithEmailAndPassword(auth, email, password);  
56     } catch (err) {  
57       console.error(err);  
58       alert(err.message);  
59     }  
60   };
```

Integrating Firebase into React app

Now, let's create a function for registering a user with an email and password:

```
62  const registerWithEmailAndPassword = async (name, email, password) => {
63    try {
64      const res = await createUserWithEmailAndPassword(auth, email, password);
65      const user = res.user;
66      await addDoc(collection(db, "users"), {
67        uid: user.uid,
68        name,
69        authProvider: "local",
70        email,
71      });
72    } catch (err) {
73      console.error(err);
74      alert(err.message);
75    }
76  };
```

Integrating Firebase into React app

Create a function that will send a password reset link to an email address:

```
78   const sendPasswordReset = async (email) => {  
79     try {  
80       await sendPasswordResetEmail(auth, email);  
81       alert("Password reset link sent!");  
82     } catch (err) {  
83       console.error(err);  
84       alert(err.message);  
85     }  
86   };
```

Integrating Firebase into React app

Create the logout function:

```
89   const logout = () => {  
90     |   signOut(auth);  
91   };
```

Integrating Firebase into React app

Finally we export all the functions at the end of the code:

```
93   export {  
94     auth,  
95     db,  
96     signInWithGoogle,  
97     loginWithEmailAndPassword,  
98     registerWithEmailAndPassword,  
99     sendPasswordReset,  
100    logout,  
101  };
```


Integrating Firebase into React app

Checkout Register.js component for implementing registration using email and password

```
</div>  
<div>  
  <button  
    onClick={() => registerWithEmailAndPassword(name, email, password)}  
  >  
    Submit  
  </button>
```

Integrating Firebase into React app

```
const onLogin = async () => {
  try {
    const result = await logInWithEmailAndPassword(email, password);
    if (result === 'sign in success') {
      setOpenTab(3);
    }
  } catch (error) {
    console.log(error);
  }
};

const onLoginWithGoogle = async () => {
  try {
    const result = await signInWithGoogle();
    if (result === 'signin with google success') {
      setOpenTab(3);
    }
  } catch (error) {
    console.log(error);
  }
};
```

Checkout Login.js component for implementing login using email and password, and also signin and register using google account

Integrating Firebase into React app

Put those function into the button and put trigger onClick actions

```
</div>
<div>
  <button onClick={onLogin}>Submit</button>
</div>
<div>
  <button onClick={onLoginWithGoogle}>Signin with Google</button>
</div>
</div>
```

Integrating Firebase into React app

```
1 import { logout } from './firebase';
2 import React from 'react';
3 import './App.css';
4
5 function Dashboard({ setOpenTab }) {
6   const onLogout = () => {
7     const result = logout();
8     if (result === 'logout success') {
9       setOpenTab(2);
10    }
11  };
12
13  return (
14    <>
15      <h3>Dashboard</h3>
16      <h4>You are login</h4>
17      <div>
18        <button onClick={onLogout}>Log out</button>
19      </div>
20    </>
21  );
22 }
```

Finally look after Dashboard.js, here is how to logout from the sessions.

Push Notification With Firebase

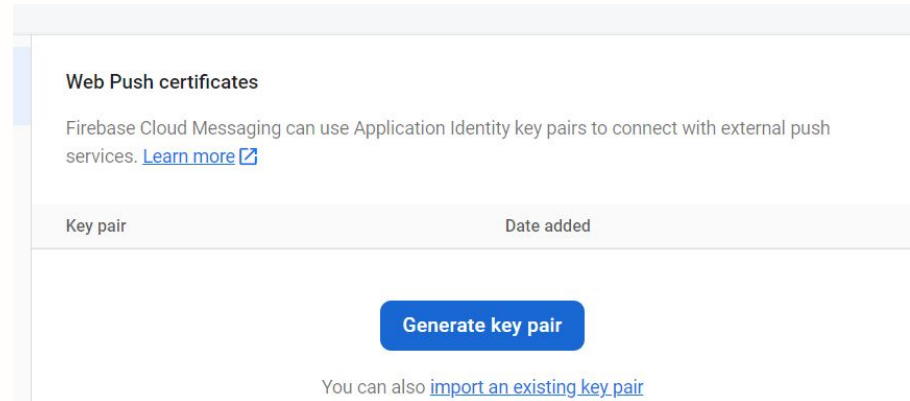
Since we already integrating our projects with firebase, lets integrate our projects to cloud messaging provide by firebase.

Generate a web push certificate key. Navigate to the cloud messaging tab for your project and scroll to the Web configuration section.

Under Web Push certificates, click on Generate key pair. Note the key that gets generated.

Go to:

https://console.firebase.google.com/u/0/project/_/settings/cloudmessaging



Push Notification With Firebase

Open up firebase.js and write down this code to import some methods provided by firebase to handling push notification.

```
19 | import { getMessaging, getToken, onMessage } from 'firebase/messaging';  
20 |
```

Push Notification With Firebase

Open up firebase.js and write down this code to import some methods provided by firebase to handling push notification.

```
19 | import { getMessaging, getToken, onMessage } from 'firebase/messaging';  
20 |
```

Use getMessaging method to access the messaging object from the firebase

```
30 | const app = initializeApp(firebaseConfig);  
31 | const auth = getAuth(app);  
32 | const db = getFirestore(app);  
33 | const messaging = getMessaging(app); You, 1  
34 |
```

Push Notification With Firebase

In order to send push notifications to the browser, we need to get permission from the user. It opens the “Enable notifications?” popup that you may have seen on other websites.

The way to initiate that request is by calling the **getToken** method Firebase provides. Before that, let's create function named as `handleGetToken` on `firebase.js`, which will keep track of whether we have access to the notifications


```
99  ✓ const handleGetToken = (setTokenFound) => {
100  ✓  return getToken(messaging, {
101      vapidKey: 'YOUR_VAPID_KEY',
102  })
103  ✓  .then((currentToken) => {
104  ✓      if (currentToken) {
105          console.log('current token for client: ', currentToken);
106          setTokenFound(true);
107          // Track the token -> client mapping, by sending to backend server
108          // show on the UI that permission is secured
109      } else {
110          console.log(
111              'No registration token available. Request permission to generate one.',
112          );
113          setTokenFound(false);
114          // shows on the UI that permission is required
115      }
116  })
117  ✓  .catch((err) => {
118      console.log('An error occurred while retrieving token. ', err);
119      // catch error while creating client token
120  });
121  };
```

Push Notification With Firebase

After create function to track notification permission, let's use that code in our App.js file by following this code:


```
10 |     const [isTokenFound, setTokenFound] = React.useState(false);  
11 |     handleGetToken(setTokenFound);  
12 |  
13 |     console.log(isTokenFound);
```


Import handleGetToken from firebase.js to use the function

```
6 | import { handleGetToken } from './firebase';
```

Push Notification With Firebase

When you allow the pop up notification, this will be shown in your screen:



Notification permission enabled 

Check out the terminal to see the logs:

```
4 current token for   firebase.js:106  
client:  
dCJs0utBtqT5gpuAXDmeRe:APA91bEugVroAw  
By2AegpfvGA81OL2aMQvty3C8WFjhVSXMQVIu  
NFGhhV2dvPzzKjaCr5_IOIpYYmvVmNUvcTNYl  
OL3Ztrs-  
ohlz3H8M89BFdC2uNdu27gjH1ncmm_RzWlZIR  
9YiLYZc
```

Configuring Message Listeners

After that we should add a listener to the incoming push notification that is directed towards the client.

We do that by adding a **firebase-messaging-sw.js** service worker file in the public folder, write down this line of code:

```
JS App.js M    JS firebase.js 1, M    JS firebase-messaging-sw.js 5, U X
public > JS firebase-messaging-sw.js > [🔗] firebaseConfig > 🔑 messagingSenderId
1  // Scripts for firebase and firebase messaging
2  importScripts(
3  |   'https://www.gstatic.com/firebasejs/9.0.0/firebase-app-compat.js',
4  |   )
5  importScripts(
6  |   'https://www.gstatic.com/firebasejs/9.0.0/firebase-messaging-compat.js',
7  |   )
8
```

Configuring Message Listeners

```
9 // Initialize the Firebase app in the service worker by passing the generated config
10 var firebaseConfig = {
11     apiKey: 'YOUR_API_KEY',
12     authDomain: 'YOUR_AUTH_DOMAIN',
13     projectId: 'YOUR_PROJECT_ID',
14     storageBucket: 'YOUR_STORAGE_BUCKET',
15     messagingSenderId: 'YOUR_MESSAGING_SENDER_ID',
16     appId: 'YOUR_APP_ID',
17 };
18
19 firebase.initializeApp(firebaseConfig);
20
```

Configuring Message Listeners

```
21 // Retrieve firebase messaging
22 const messaging = firebase.messaging();
23
24 messaging.onBackgroundMessage(function (payload) {
25     console.log('Received background message ', payload);
26
27     const notificationTitle = payload.notification.title;
28     const notificationOptions = {
29         body: payload.notification.body,
30     };
31
32     self.registration.showNotification(notificationTitle, notificationOptions);
33 });
```


Foreground Listener

To take care of cases in which the app is active in foreground, we need to add this code to the firebase.js file:

```
124   const onMessageListener = () =>
125     new Promise((resolve) => {
126       onMessage(messaging, (payload) => {
127         resolve(payload);
128       });
129     });
```

Export onMessageListener function.

Foreground Listener

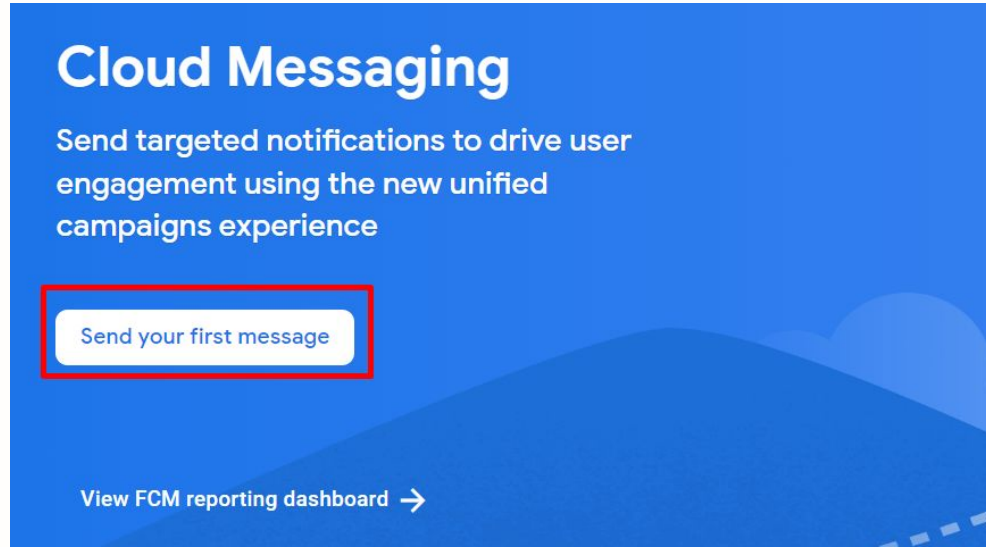
```
14  const onMessageListenerHandler = () => {  
15    try {  
16      const result = onMessageListener();  
17      console.log(result);  
18    } catch (error) {  
19      console.log(error);  
20    }  
21  };  
22  useEffect(() => {  
23    onMessageListenerHandler();  
24  }, []);
```

We also need to import this in App.js and add logic to create the notification out of the parsed payload, which looks something like this:

Testing Push Notifications

Open up your google firebase projects and go to cloud messaging sections.

On the home page, click on Send your first message. Enter the Notification title and Notification text. Then, under the Device preview section, click on Send test message.



Testing Push Notifications

Fill out the form and click on send test message under Device preview section

1

Notification

Notification title ?

This is pwdk testing notification

Notification text

Hello world, this is push notification from firebase

Notification image (optional) ?

Device preview

This preview provides a general idea of how your message will appear on a mobile device. Actual message rendering will vary depending on the device. Test with a real device for actual results.

Send test message

Testing Push Notifications

Copy and paste your token from web console, and put it on Add an FCM registration token and click on + button. Make sure the FCM token is checked, then click on Test.

Test on device



You can test this campaign by entering or selecting the [FCM registration tokens](#) of your development device below.

Add an FCM registration token

Recently Used ?

☒

fCj18IQ8Vs4ggmW22yvJhD:APA91bF350_Rgw3_YnGx8I5mdgVXxtS...

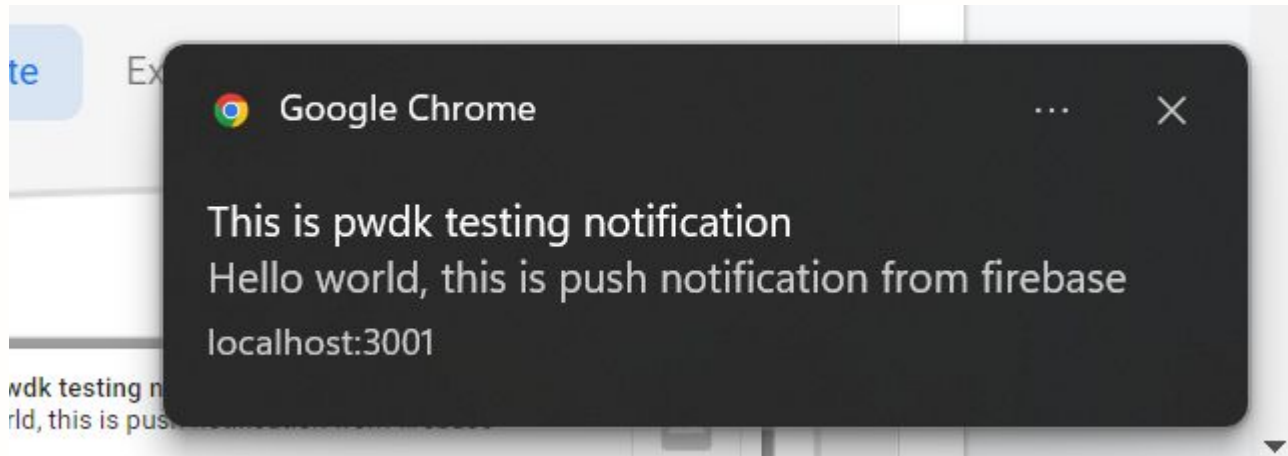


Cancel

Test

Testing Push Notifications

Notification successfully sent to your application.



Thank You!

