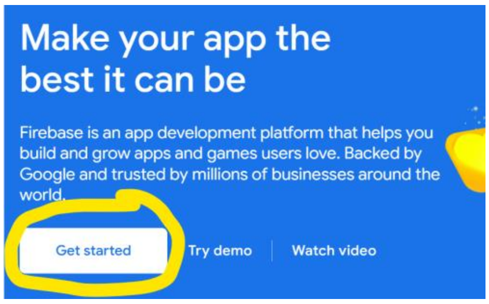
# Table of Contents

# PART 1 - How to run this project?

## Step 1: Register Firebase Account

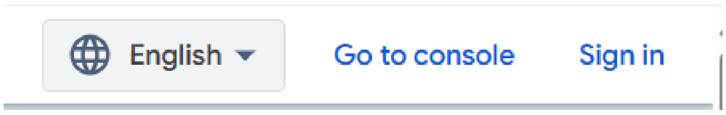
Open that link to create a Firebase Account https://firebase.google.com/

Click "**Get Started**" and register with your Google Account


Make your app the best it can be

Firebase is an app development platform that helps you build and grow apps and games users love. Backed by Google and trusted by millions of businesses around the world.

You can also switch Language on the top-right corner.

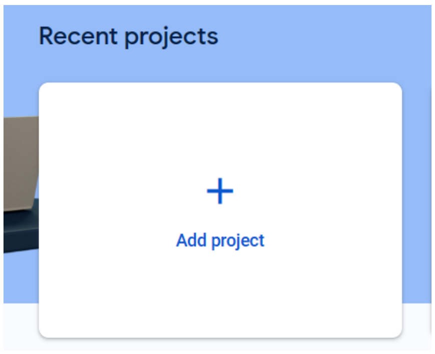


You can switch Language on the top-right corner.

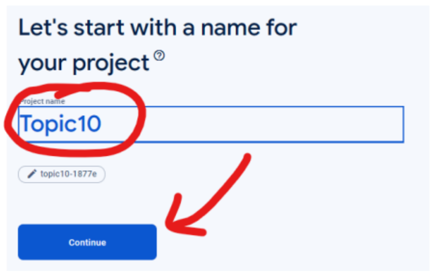
## Step 2: Create Firebase Projects

After register your account, you can see a Dashboard.

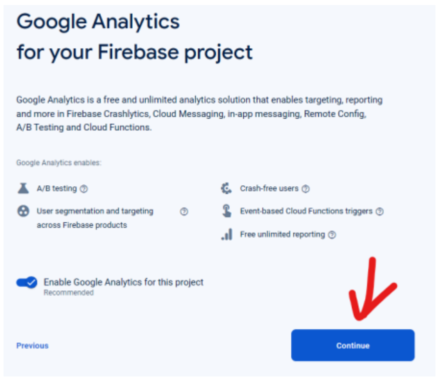
Click **"Add Project"** to create a Firebase Project.



1– Type your project name


Let's start with a name for your project

2– Continue



Google Analytics
for your Firebase project

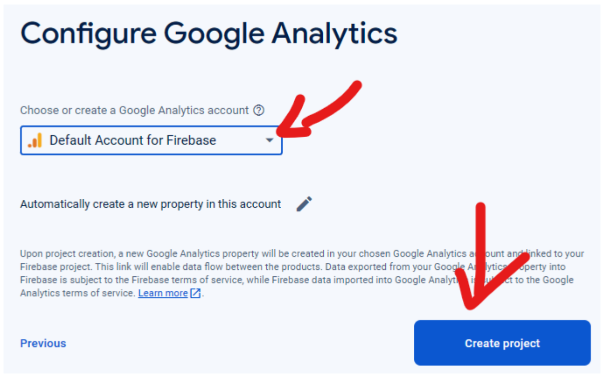
Google Analytics is a free and unlimited analytics solution that enables targeting, reporting and more in Firebase Crashlytics, Cloud Messaging, in-app messaging, Remote Config, A/B Testing and Cloud Functions.

Google Analytics enables:

- A/B testing
- User segmentation and targeting across Firebase products
- Crash-free users
- Event-based Cloud Functions triggers
- Free unlimited reporting

Enable Google Analytics for this project
Recommended

3– Select **Default Account for Firebase** and **Create Project**



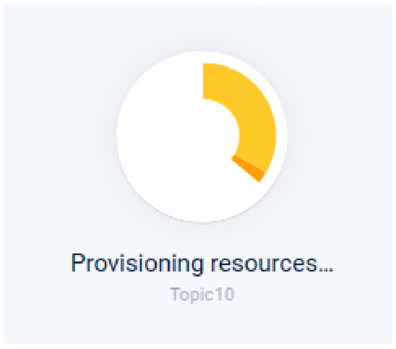
Configure Google Analytics

Choose or create a Google Analytics account: Default Account for Firebase

Automatically create a new property in this account

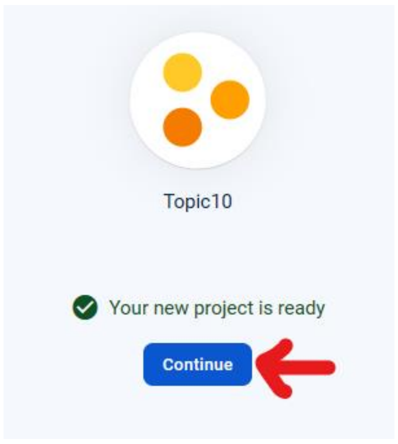
Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. Learn more.

4– Wait



Provisioning resources...
Topic10

5– Project created completely.

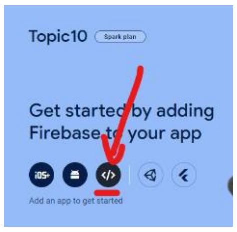


Topic10

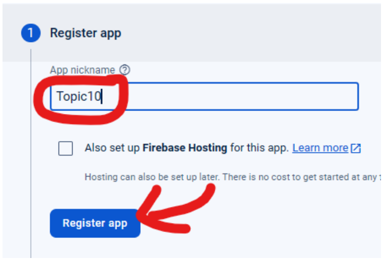
Your new project is ready

## Step 3: Add Firebase to App

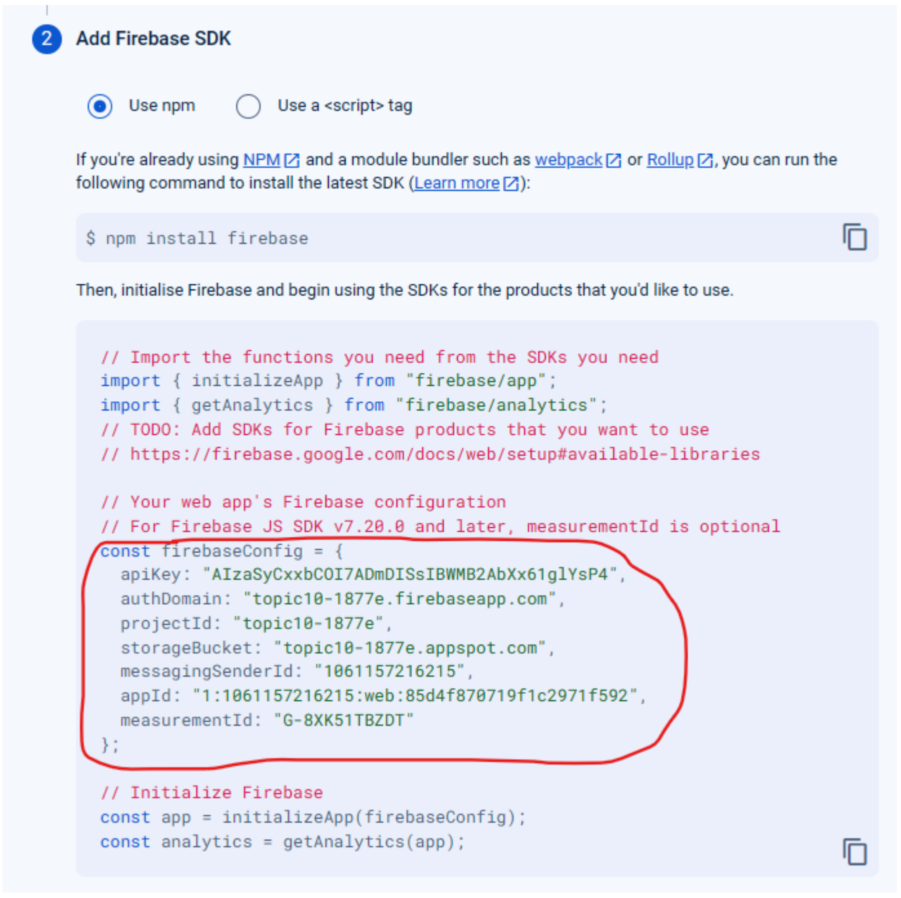
1- Add Firebase to Web App


Topic10
Spark plan
Get started by adding Firebase to your app
Add an app to get started

2- Set **"Nick Name"** and **register App**.


1 Register app
App nickname
Topic10
Also set up Firebase Hosting for this app. Learn more
Hosting can also be set up later. There is no cost to get started at any
Register app

3- Copy the **firebaseConfig** in your TXT file. You will use it later.



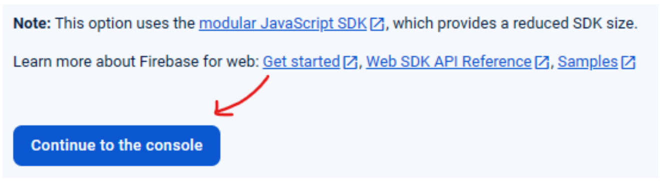
If you're already using NPM and a module bundler such as webpack or Rollup, you can run the following command to install the latest SDK (Learn more):

```
$ npm install firebase
```

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

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// 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
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyCxxbCOI7ADmDISsIBWMB2AbXx61glYsP4",
  authDomain: "topic10-1877e.firebaseapp.com",
  projectId: "topic10-1877e",
  storageBucket: "topic10-1877e.appspot.com",
  messagingSenderId: "1061157216215",
  appId: "1:1061157216215:web:85d4f870719f1c2971f592",
  measurementId: "G-8XK51TBZDT"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

4- Continue to the console!

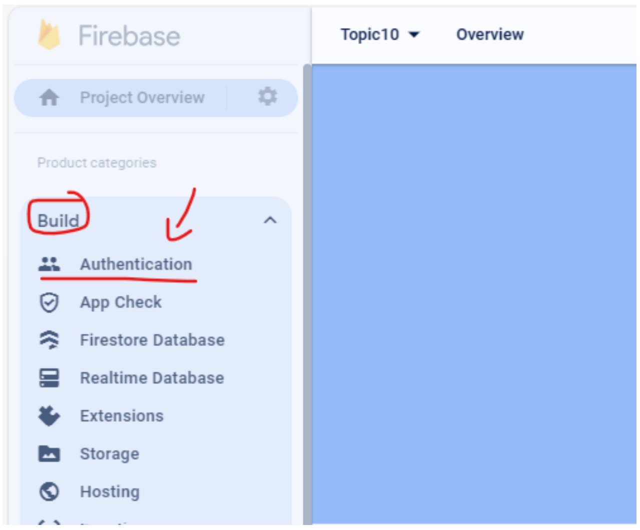


**Note:** This option uses the modular JavaScript SDK, which provides a reduced SDK size.

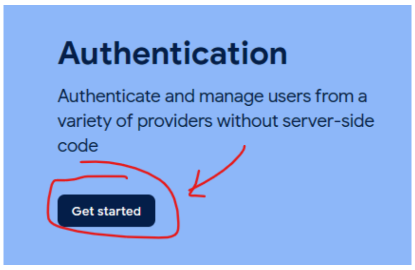
Learn more about Firebase for web: Get started, Web SDK API Reference, Samples

## Step 4: Set User Authentication

1– In the top–left menu (Product categories), select **Build –> Authentication**



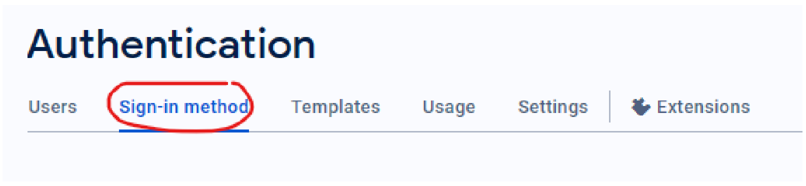
2– Get Started



Authentication

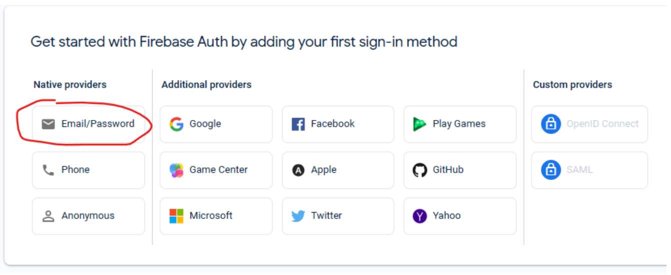
Authenticate and manage users from a variety of providers without server-side code

3– Click Sign–in Method



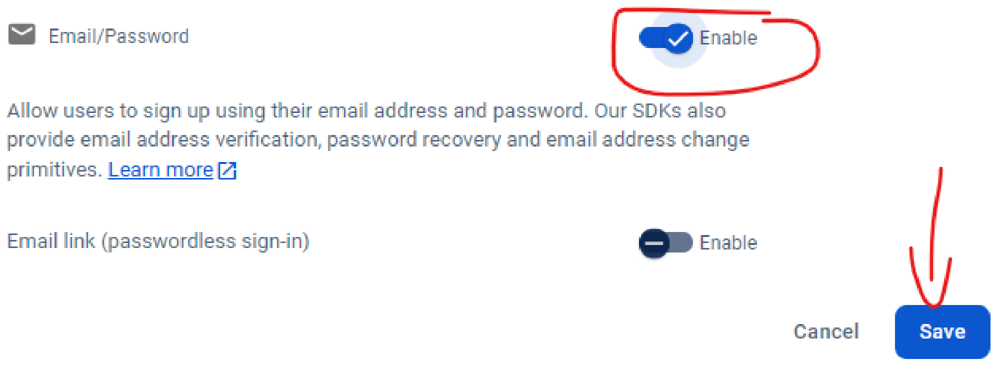
4– There are many sign–in methods.

In this project, we used the easiest one – **Email/Password**



Get started with Firebase Auth by adding your first sign-in method

5- Enable that method and save.


Email/Password

Enable

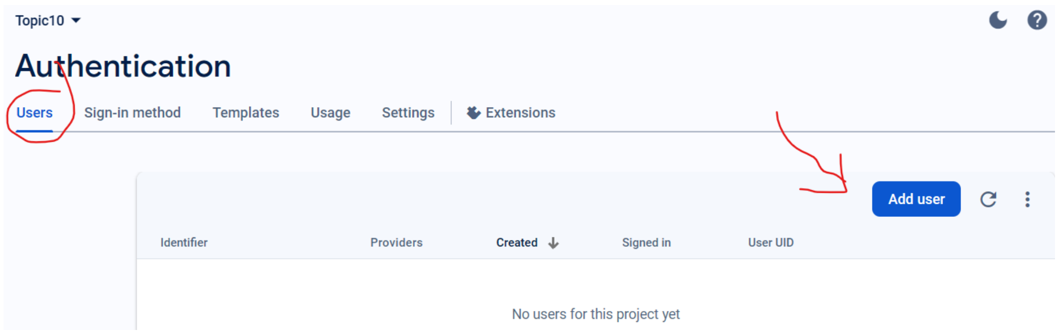
Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery and email address change primitives. Learn more

Email link (passwordless sign-in)

Enable

Cancel Save

6- Click **"Users"** in the menu, then **Add user.**


Topic10

Authentication

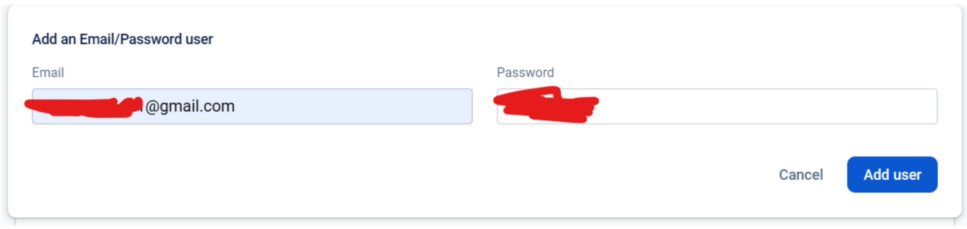
Users Sign-in method Templates Usage Settings Extensions

Add user

| Identifier | Providers | Created | Signed in | User UID |
|---|---|---|---|---|

No users for this project yet

7- Set the **email** and **password** in Authentication, then **Add User**


Add an Email/Password user

Email

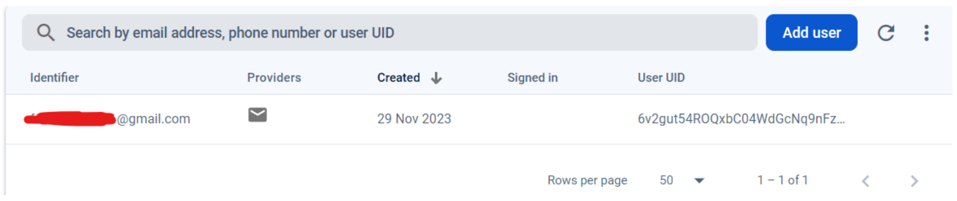
[redacted]@gmail.com

Password

[redacted]

Cancel Add user

8- You can see that **User** is Added in Firebase Authentication


Search by email address, phone number or user UID

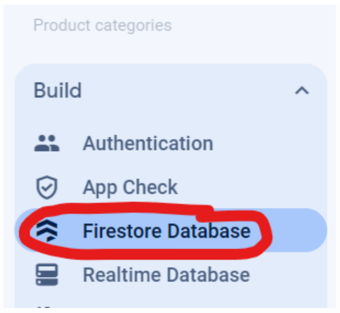
Add user

| Identifier | Providers | Created | Signed in | User UID |
|---|---|---|---|---|
| [redacted]@gmail.com | ✉ | 29 Nov 2023 | | 6v2gut54ROQxbC04WdGcNq9nFz... |

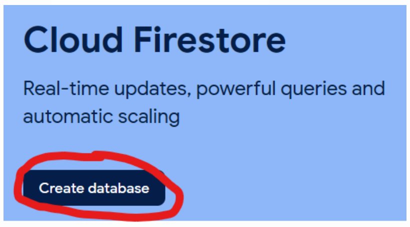
Rows per page 50 1 – 1 of 1

## Step 5: Check Firestore Database

1- In the top-left menu (Product categories), select **Build -> Firestore Database**



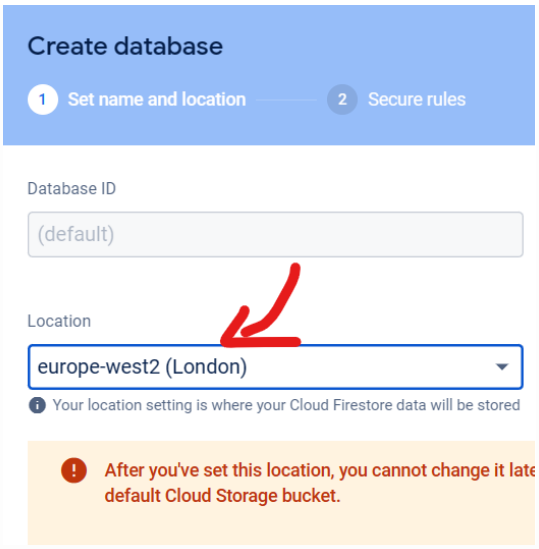
2- Create database


Cloud Firestore

Real-time updates, powerful queries and automatic scaling

3- Set the Firebase location

I select **Europe-west2(London),** because that server is closest to where I live.


Create database

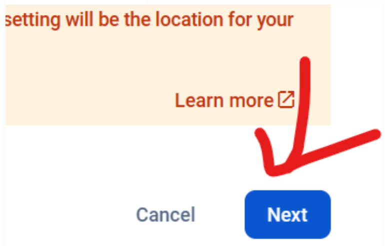
Database ID: (default)

Location: europe-west2 (London)

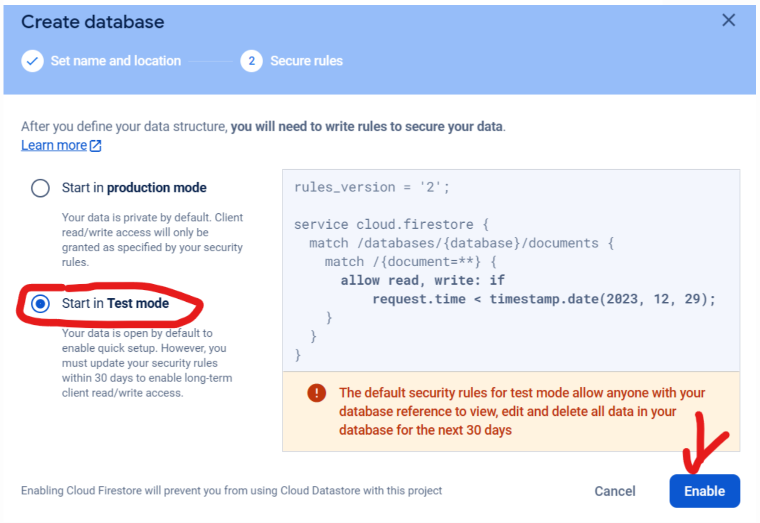
Your location setting is where your Cloud Firestore data will be stored

After you've set this location, you cannot change it late
default Cloud Storage bucket.

4- Click **Next**


setting will be the location for your

5- Select Rules – **Start in Test mode**, then **Enable.**



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.

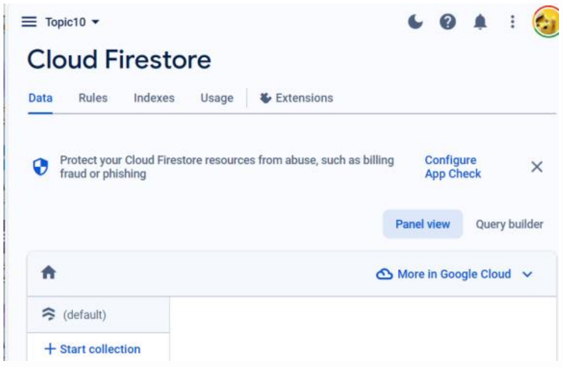
Start in **Test mode** [✓]
Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if
          request.time < timestamp.date(2023, 12, 29);
    }
  }
}
```

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

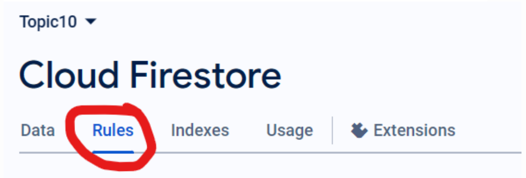
Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project

6- Firestore Database created successfully

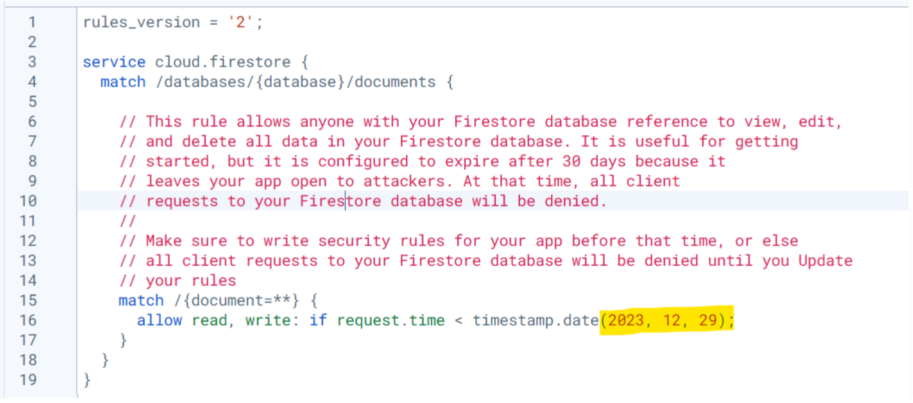


Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing

**7-** Pay attention to the **Security Rules!**



If the date is expired, you cannot use that Firebase App anymore. You must ensure that **the effective date of Firebase App should be after the current date.** ust enYou can edit the effective date in the code.



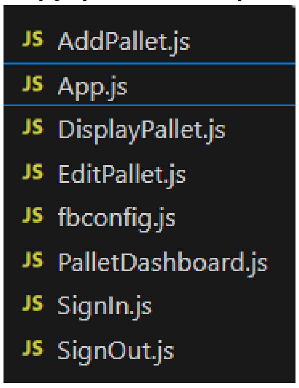
```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {

    // This rule allows anyone with your Firestore database reference to view, edit,
    // and delete all data in your Firestore database. It is useful for getting
    // started, but it is configured to expire after 30 days because it
    // leaves your app open to attackers. At that time, all client
    // requests to your Firestore database will be denied.
    //
    // Make sure to write security rules for your app before that time, or else
    // all client requests to your Firestore database will be denied until you Update
    // your rules
    match /{document=**} {
      allow read, write: if request.time < timestamp.date(2023, 12, 29);
    }
  }
}
```

## Step 6: Install and config React Project

1– Build the React Framework

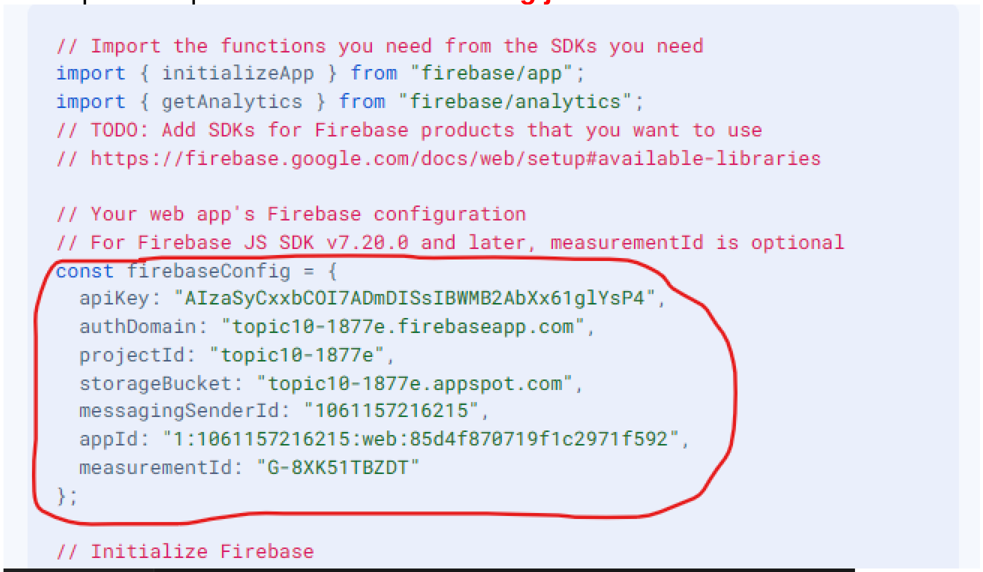
Use VS Code to open a new folder then type
**npx create-react-app client**
VS Code will create a React Framework for this project.

2– **Copy, paste and replace** the 8 files in this project.

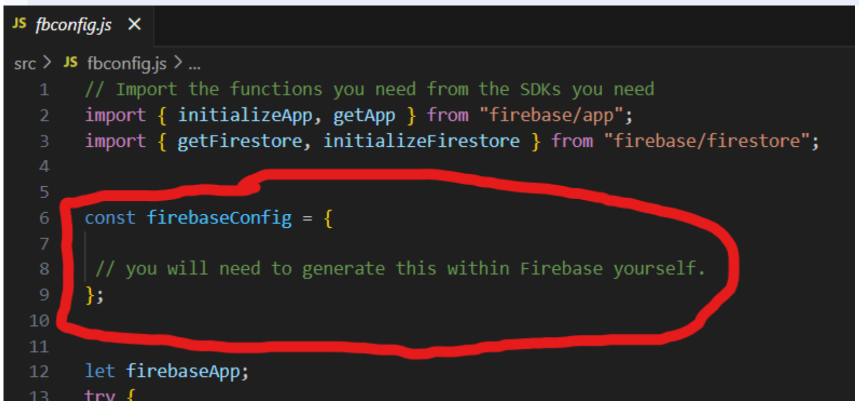


```
JS AddPallet.js
JS App.js
JS DisplayPallet.js
JS EditPallet.js
JS fbconfig.js
JS PalletDashboard.js
JS SignIn.js
JS SignOut.js
```

3– Set firebaseConfig.
In the firebase website, copy the String in **firebaseConfig**, and then paste it into the correspondent place in the JS file **fbconfig.js**



```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// 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
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyCxxbCOI7ADmDISsIBWMB2AbXx61glYsP4",
  authDomain: "topic10-1877e.firebaseapp.com",
  projectId: "topic10-1877e",
  storageBucket: "topic10-1877e.appspot.com",
  messagingSenderId: "1061157216215",
  appId: "1:1061157216215:web:85d4f870719f1c2971f592",
  measurementId: "G-8XK51TBZDT"
};

// Initialize Firebase
```

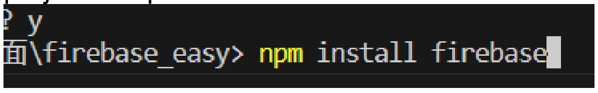


```
JS fbconfig.js ×
src > JS fbconfig.js > ...
// Import the functions you need from the SDKs you need
import { initializeApp, getApp } from "firebase/app";
import { getFirestore, initializeFirestore } from "firebase/firestore";


const firebaseConfig = {

  // you will need to generate this within Firebase yourself.
};


let firebaseApp;
try {
```

4- Install the dependency

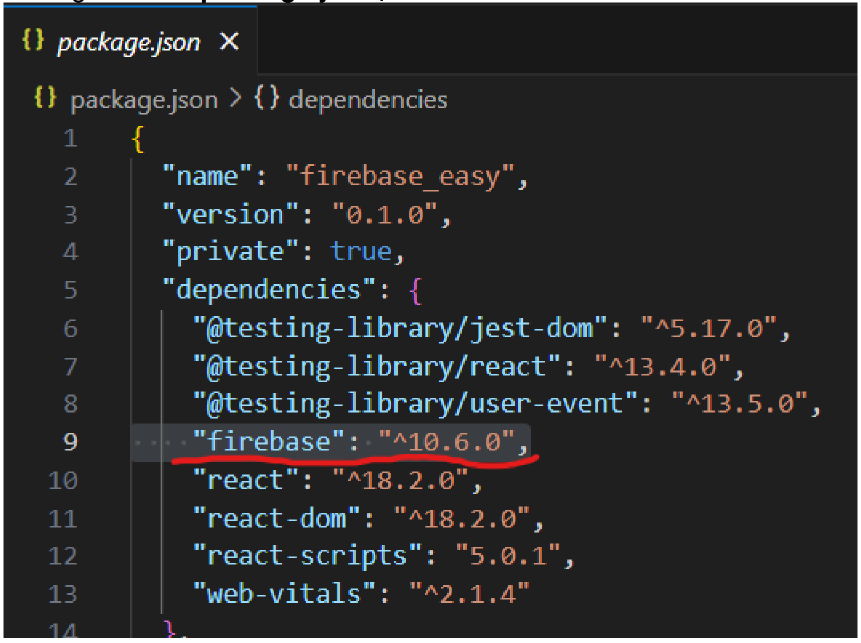
Remember to instlal the dependency of Firebase, otherwise your program will not work. There are two ways to install the dependency

1) Type **npm install firebase** in the terminal, it will install the latest stable version in your project's dependencies.



```
? y
面\firebase_easy> npm install firebase
```

2) Change the file **package.json,** and add the latest stable version of firebase.

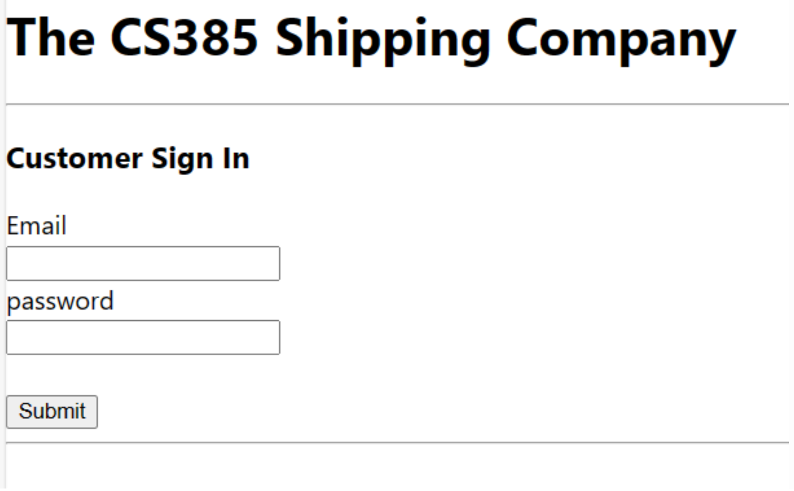


```
{
  "name": "firebase_easy",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "firebase": "^10.6.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
```

## Step 7: Run this project and test

1– Run the project

After you finished Step 6, you can run this project and test the CRUD now.
Type **npm start** in the terminal, the project will run at http://localhost:3000/



The CS385 Shipping Company

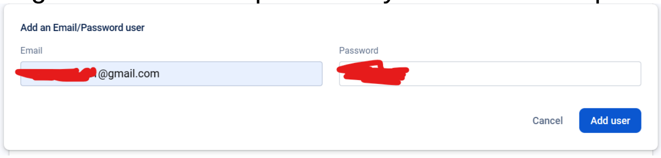
Customer Sign In

Email

password

## 2– Login

Login with the email/password you set in the step of Authentication.

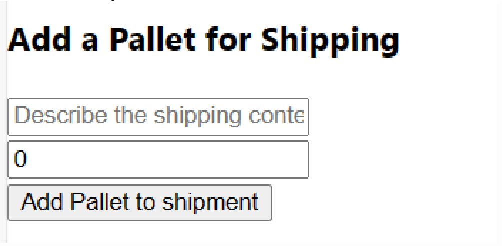

Add an Email/Password user

Email: [redacted]@gmail.com

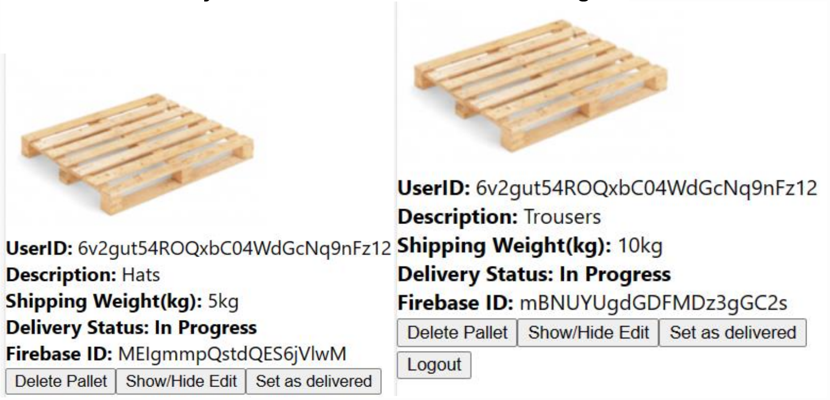
Password: [redacted]

## 3- Add Pallet

Add two pallets, set the Pallet name and weight.

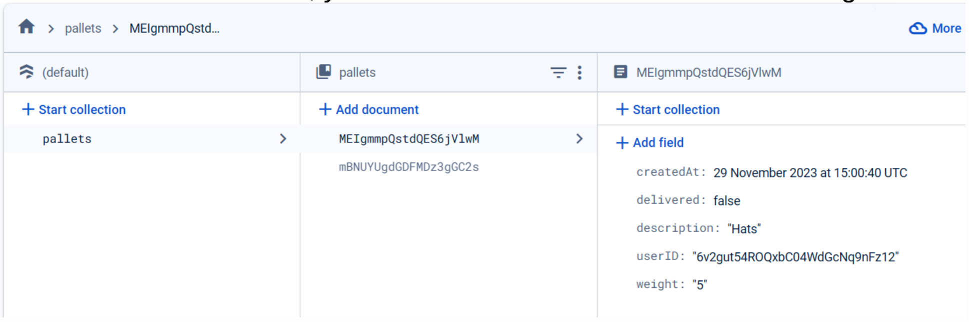

**Add a Pallet for Shipping**

You can see they could be seen in the WebPages.


**UserID:** 6v2gut54ROQxbC04WdGcNq9nFz12
**Description:** Hats
**Shipping Weight(kg):** 5kg
**Delivery Status: In Progress**
**Firebase ID:** MEIgmmpQstdQES6jVlwM

**UserID:** 6v2gut54ROQxbC04WdGcNq9nFz12
**Description:** Trousers
**Shipping Weight(kg):** 10kg
**Delivery Status: In Progress**
**Firebase ID:** mBNUYUgdGDFMDz3gGC2s

Go to Firestore Database, you can also see the collections has changed.



pallets: MEIgmmpQstdQES6jVlwM, mBNUYUgdGDFMDz3gGC2s

MEIgmmpQstdQES6jVlwM
```
createdAt: 29 November 2023 at 15:00:40 UTC
delivered: false
description: "Hats"
userID: "6v2gut54ROQxbC04WdGcNq9nFz12"
weight: "5"
```

The new **collection** pallets was added, it has two **documents**, each document has the **creationTime, deliveredStatus, description, userID and weight**.

## 4- To be continued

Try to play this project for fun.
You can also edit the information, delete the pallet or set it as delivered.
The data in Firestore database will be changed as well, because the **React App is now closely connected with Firebase Datatabse.**
That is how the CRUD works in the full-stack application.

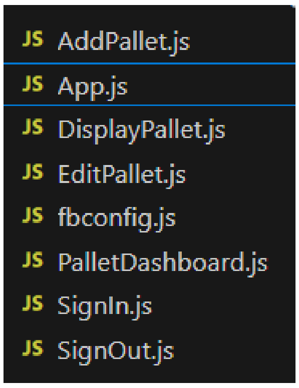
# PART 2 – Project Architecture

## Using Firebase – The CS385 Shipping Comany

Firebase is free and mobile-app focused. You don't need to write your own SERVER or backend, because Firebase provides the whole solution. The work is seamlessly with JSON objects, and Firebase provides you with a full C-R-U-D application functionality set.

## 1. Firebase App Structure

There are 8 files in this App.



JS AddPallet.js
JS App.js
JS DisplayPallet.js
JS EditPallet.js
JS fbconfig.js
JS PalletDashboard.js
JS SignIn.js
JS SignOut.js

- **Part A:** Consider **the use of Firebase as a means of providing persistent storage for an application** (without any authentication or user management).
- **Part B:** Consider how to use **Google Firebase as an API for providing authentication functionality** for any React application.
- **Part C:** **Combination:** Use the code in Part B to offer authentication and user data management for the application in Part A.

## 2. No-SQL database

Firebase allows us to **synchronize data continuously** across all users of our app.

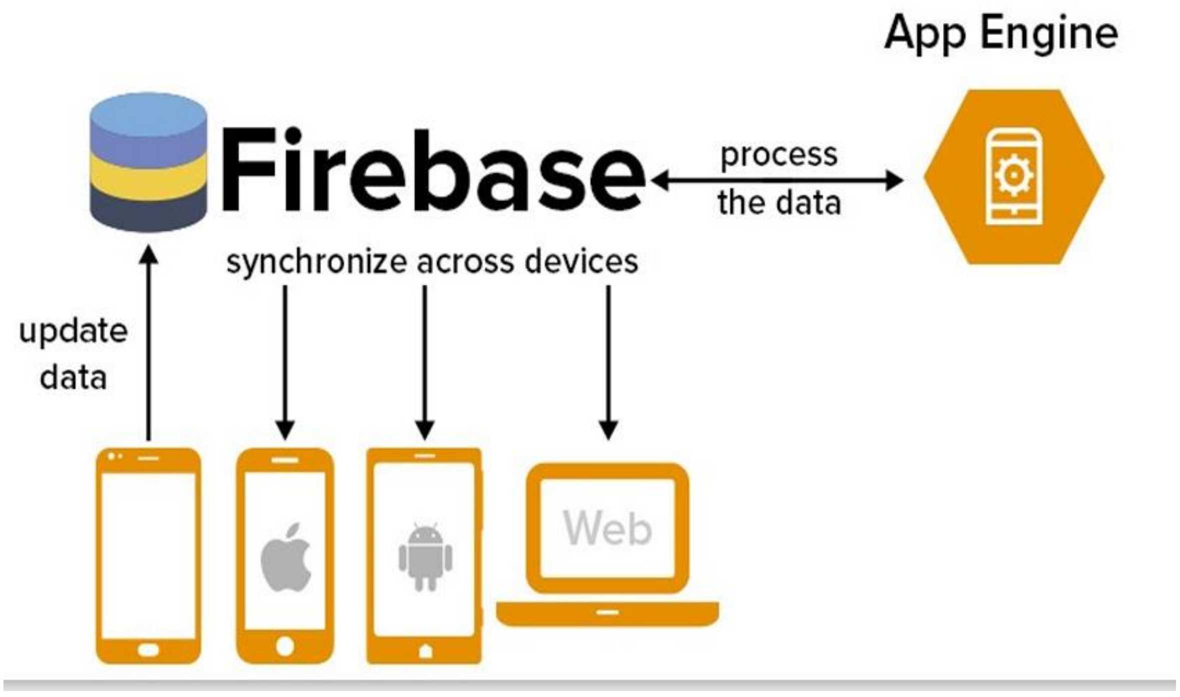
**Authentication:** Who can use this App?

**Authorization:** What are you allowed to do?

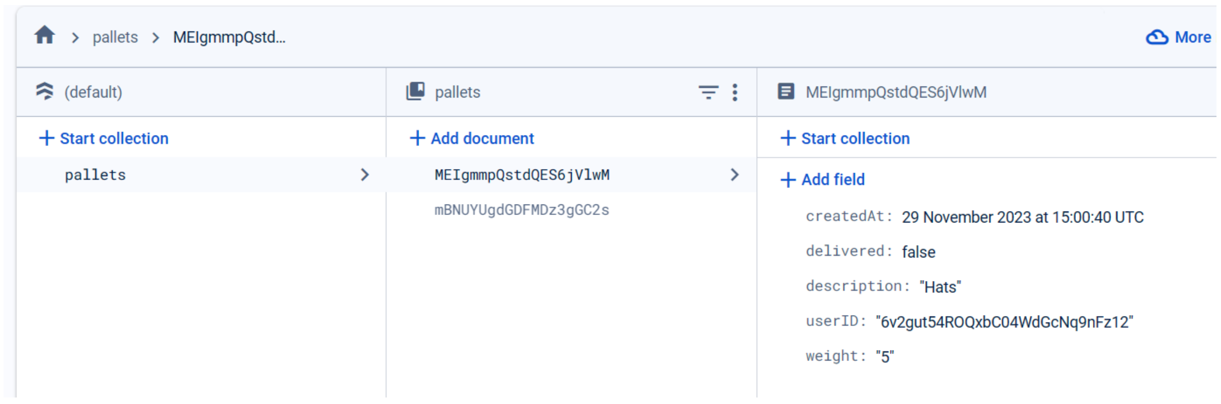
Firebase is No-SQL, document-oriented database. Unlike a SQL database, there is no tables or rows, instead, you store data (JSON objects) in **documents**, which are organized into **collections**.

Each document (object) contains **a set of key-value pairs.** All documents (objects) must be stored in collections. **Collections and documents** are created **implicitly** in Cloud Firestore.

In No-SQL database, we can insert JSON directly. If we want to add data in SQL database, we must write SQL Language.



## 3. Firestore Database



```
pallets
MEIgmmpQstdQES6jVlwM
mBNUYUgdGDFMDz3gGC2s
```

createdAt: 29 November 2023 at 15:00:40 UTC
delivered: false
description: "Hats"
userID: "6v2gut54ROQxbC04WdGcNq9nFz12"
weight: "5"

**Pallets –** Collection of Documents.

**2 Documents** – In the middle column

**5 Properties** – Document object, **key-value pairs**

# 4. Part A – Pallet Dashboard

In this part, we need to allow a user to

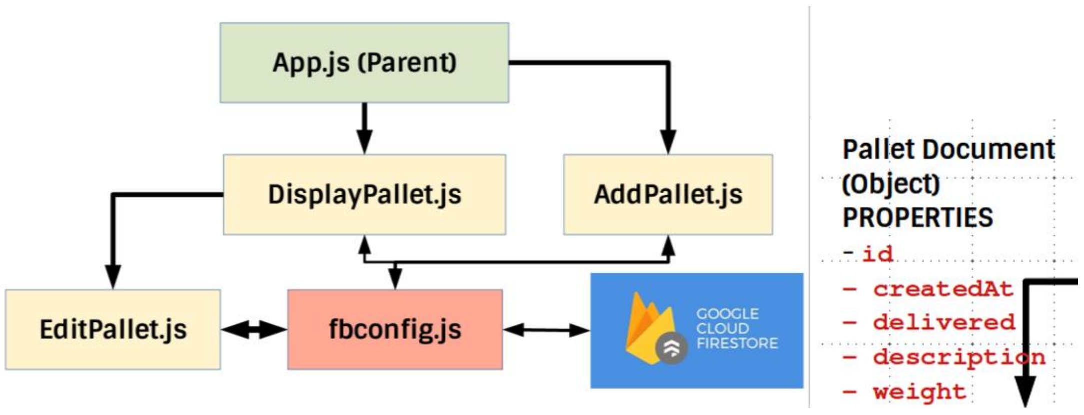
- **Add a pallet for shipping (add to our Firestore database)** – Allow the user to specify the goods on the pallet and the overall weight (KG).
- **Display all pallets** in our warehouse**.**
- **Delete a pallet** – remove this pallet from our Firestore database
- **Edit a pallet** – allow the user to change the details for an existing pallet in our Firestore database.

At this stage the shipping pallet will have four properties

- **Description** – String – a short description of the contents of the pallet.
- **Weight** – the approximate total weight of the pallet in kilograms.
- **CreatedAt** – this will be timestamp when the pallet document is stored in Firestore.
- **Delivered** – this is a boolean value to indicate if the pallet has been delivered.

Our entire application was developed around **the Pallet document (object)** – see below. The application is essentially driven by the movement of this object in and out of Firestore.

**Overall Component Schematic**



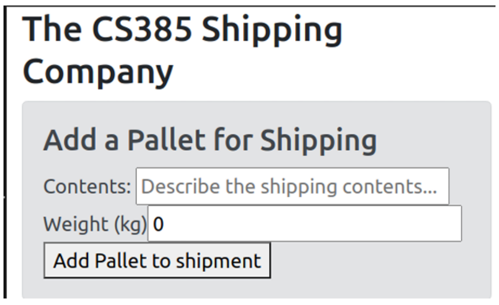
The **App.js** component is the parent. The other components are all considered as child components. The **fbconfig.js** file provides the configuration information to allow the components to access the Firestore database for this application.

## AddPallet.js

In this component, we must

- Obtain the **description** from the user.
- Obtain the **weight (KG)** of the pallet from the user.

When user presses the "App Pallet to shipment" button, the component should insert this document(object) in our firestore database.


The CS385 Shipping Company

Add a Pallet for Shipping

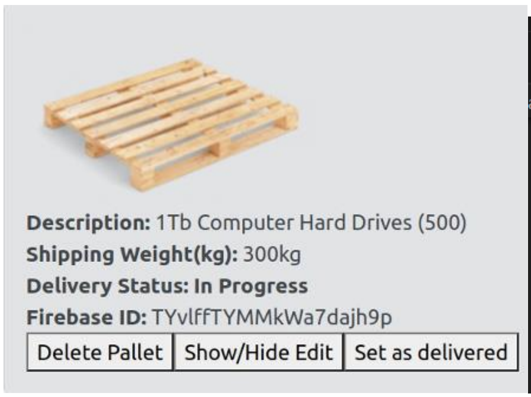
Contents: Describe the shipping contents...

Weight (kg) 0

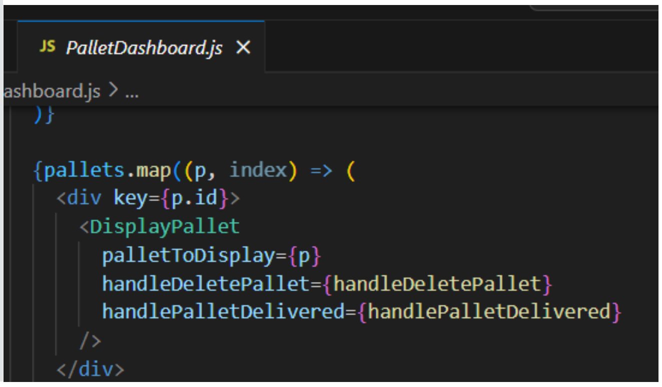
Add Pallet to shipment

## PalletDashboard.js

We use this component to display ALL documents in the "pallets" collection.


**Description:** 1Tb Computer Hard Drives (500)
**Shipping Weight(kg):** 300kg
**Delivery Status:** In Progress
**Firebase ID:** TYvlfFTYMMkWa7dajh9p

Delete Pallet | Show/Hide Edit | Set as delivered


PalletDashboard.js

```
ashboard.js > ...
  )}

  {pallets.map((p, index) => (
    <div key={p.id}>
      <DisplayPallet
        palletToDisplay={p}
        handleDeletePallet={handleDeletePallet}
        handlePalletDelivered={handlePalletDelivered}
      />
    </div>
```

*Use map function to render each **DisplayPallets.j**s one by one.*

- **Query** the Firestore database.
- **Retrieve** all of the documents within the "pallets" collection.
- Using a **map function** to display or render each document(object) within this collection

## DisplayPallet.js

This component is each Pallets rendered from ***PalletDashboard.js.***

Each Pallet has 5 properties:

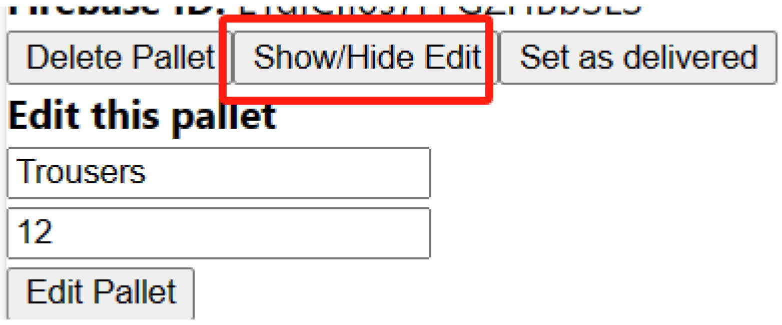
- **UserID:**
- **Description**
- **Shipping Weight(kg)**
- **Delivery Status**
- **Firebase ID:**

Each Pallet also has 3 functions:

- **Delete Pallet** button.
- **Set as Delivered** button
- **Edit Pallet button** – *this will call a child Component (EditPallet.js)*

## EditPallet.js

According to the **PalletID (Document ID)**, keep the deliveryStatus, creationTime and userID as the same, then change the **description and weight.**



When you click this button **"Show/Hide Edit"**, you will see a popup under the Pallet. The default value is the previous value of that Pallet, you can **edit the description and weight** of that Pallet, then send it back to firebase.

However, after you click **"Set as delivered"**, you cannot edit the Pallet anymore, that button has been disabled.

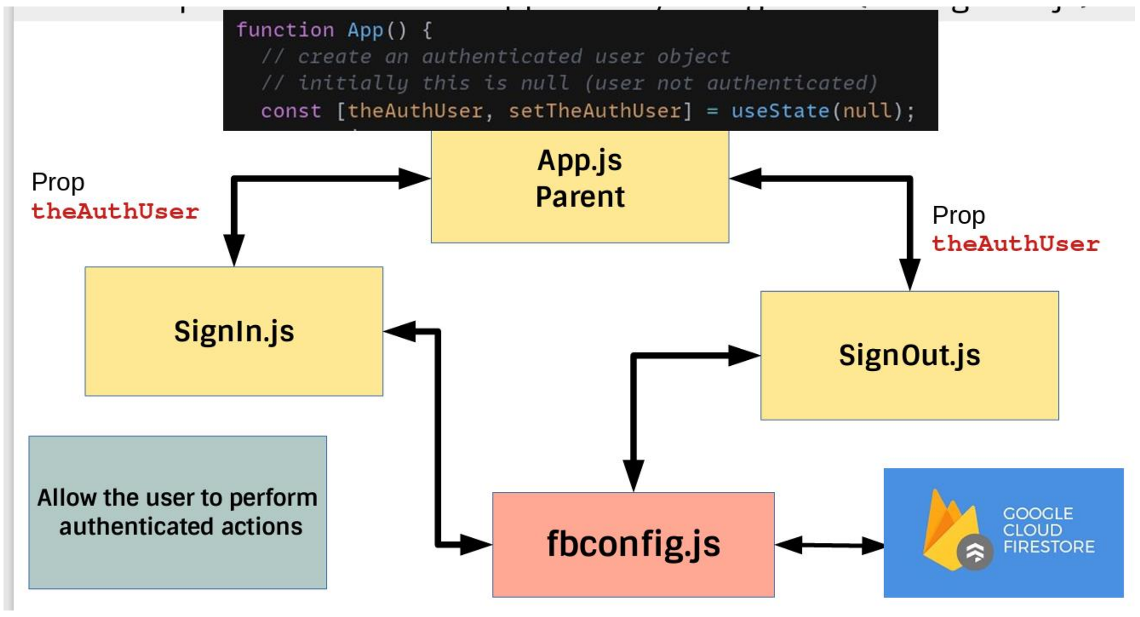
# 5. Part B - Authentication

**When a user authenticates in an application, your application software code has access to a special object containing authentication details about the user. This is the authenticated user.**

By using prop ***theAuthUser*** your software code can **determine if the user is authenticated (logged in)** or not. **When the user has logged out (not authenticated) then this object is null.**

The complication (solved by Firebase) is to provide a way to establish if a user is authenticated or not.

**Overall Component Schematic**



```
function App() {
  // create an authenticated user object
  // initially this is null (user not authenticated)
  const [theAuthUser, setTheAuthUser] = useState(null);
```

The App component (Parent) can offer the user the **opportunity to authenticate (via SignIn.js)**.

When the user is authenticated they can then be presented with the opportunity to **sign out (via SignOut.js) .** We use this Part to realize the function of **Log in and Log out** this application.

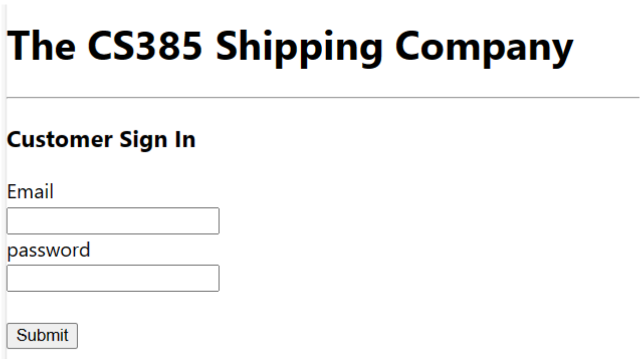
## App.js

The first Page when the Program runs. It will show the Login Page first.

This app used conditional rendering by the props **theAuthUser.**

**If the user is authenticated, then App.js will show PalletDashboard.js and signOut.js.**

## SignIn.js



The CS385 Shipping Company

Customer Sign In

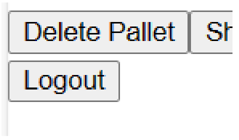
Email

password

**If the user fails to authenticate, the App.js will keep showing SignIn.js.**

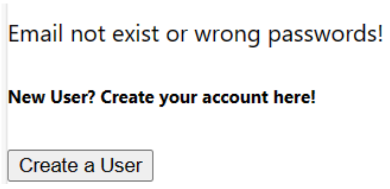
The props **theAuthUser** must have value if we want to go to the Dashboard home page.

## SignOut.js



After the user click **"Log Out"** button. The prop **theAuthUser** will be set to null, then **App.js** will render **Signin.js** again.

## SignUp.js



Email not exist or wrong passwords!

New User? Create your account here!

If the user fails to **Sign Up.** This Component will provide a **button** for the user to **sign up a new account** with the email and password he input before.

## 6. Part C – Combination

The App component (Parent) can offer the user the **opportunity to authenticate (via SignIn.js)**.

When the user is authenticated they can then be presented with the opportunity to **sign out (via SignOut.js)**

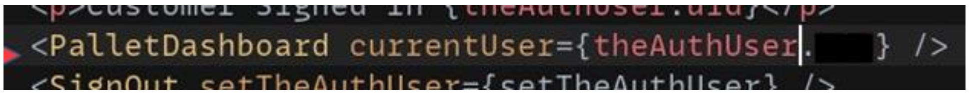
After we finished Part A and Part B, we need to use the **Authentication from Part B** to control access to **Pallet Dashboard from Part A.**

Informally, when **a user l**ogons to our CS385 Shipping Company Ltd application, they can **CRUD Pallet** documents (objects) AKA Shipments. **Users will be only able to see their own documents (objects).**

Users will be able to **AddPallet, EditPallet, DeletePallet** and **SetAsDelivered** – but **only for the Pallet documents that they own.**

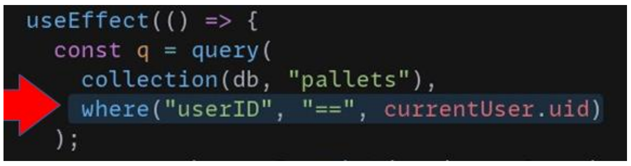
**STEPS**

1) Change App.js in Part A into PalletDashboard.js



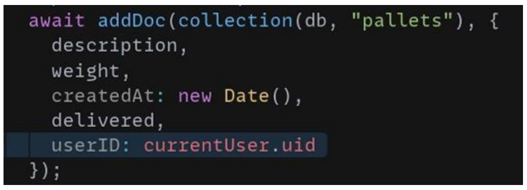
```
<p>Customer signed in {theAuthUser.uid}</p>
<PalletDashboard currentUser={theAuthUser.[redacted]} />
<SignOut setTheAuthUser={setTheAuthUser} />
```

2) Select Collection by **userID.**

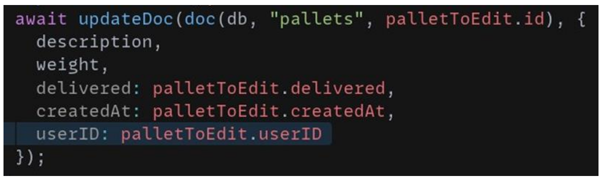


```
useEffect(() => {
  const q = query(
    collection(db, "pallets"),
    where("userID", "==", currentUser.uid)
  );
```

3) When adding / editing Pallets, add the **userID** as well.

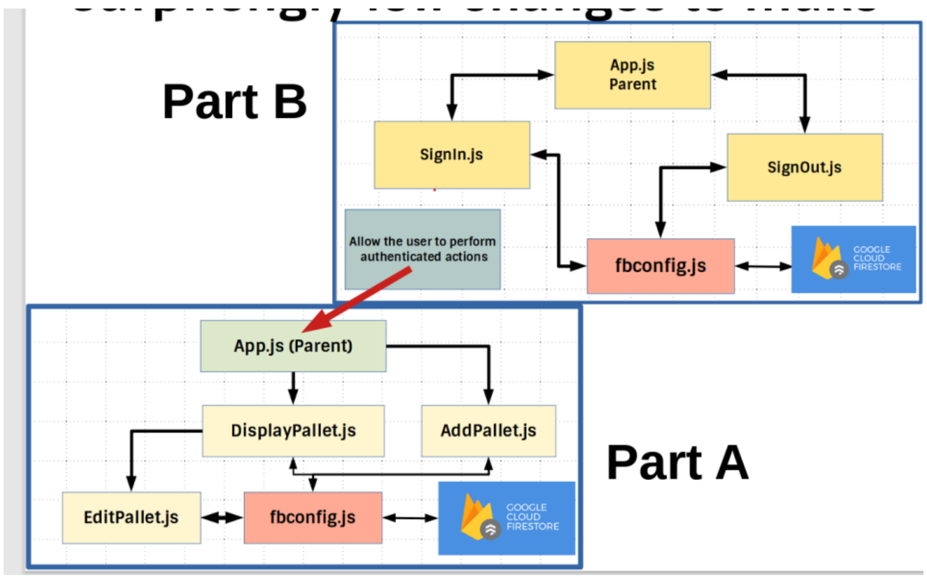


```
await addDoc(collection(db, "pallets"), {
  description,
  weight,
  createdAt: new Date(),
  delivered,
  userID: currentUser.uid
});
```



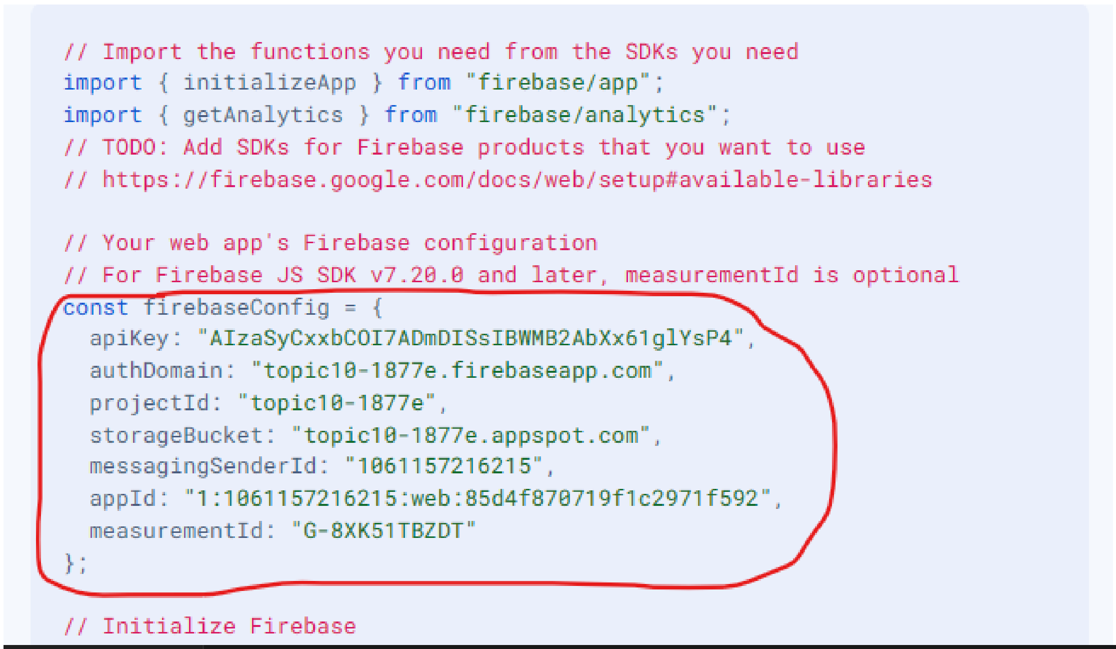
```
await updateDoc(doc(db, "pallets", palletToEdit.id), {
  description,
  weight,
  delivered: palletToEdit.delivered,
  createdAt: palletToEdit.createdAt,
  userID: palletToEdit.userID
});
```

4) Done!

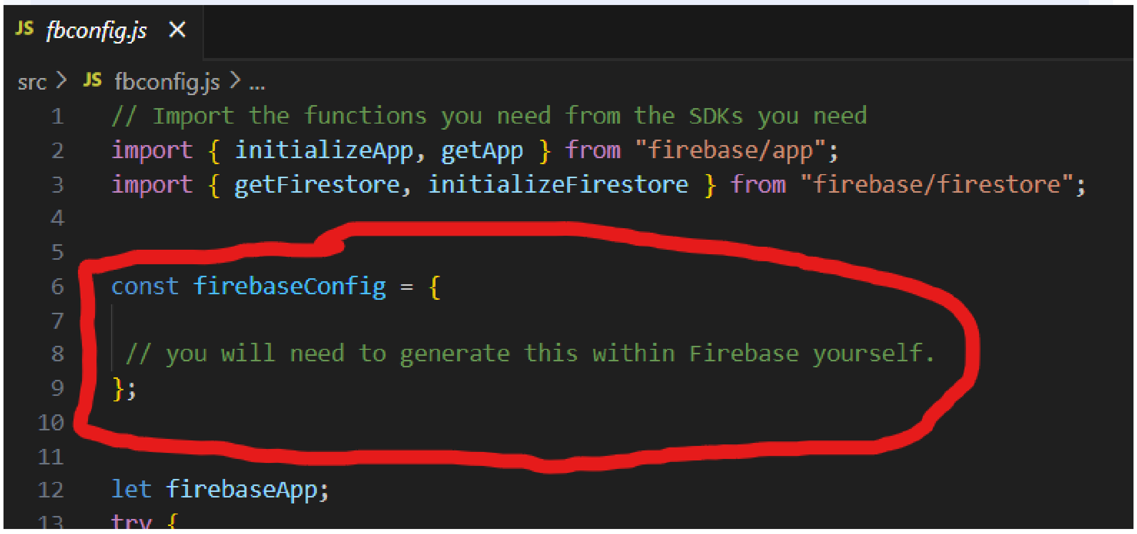


## fbconfig.js

In the firebase website, copy the String in **firebaseConfig**, and then paste it into the correspondent place in the JS file **fbconfig.js.**

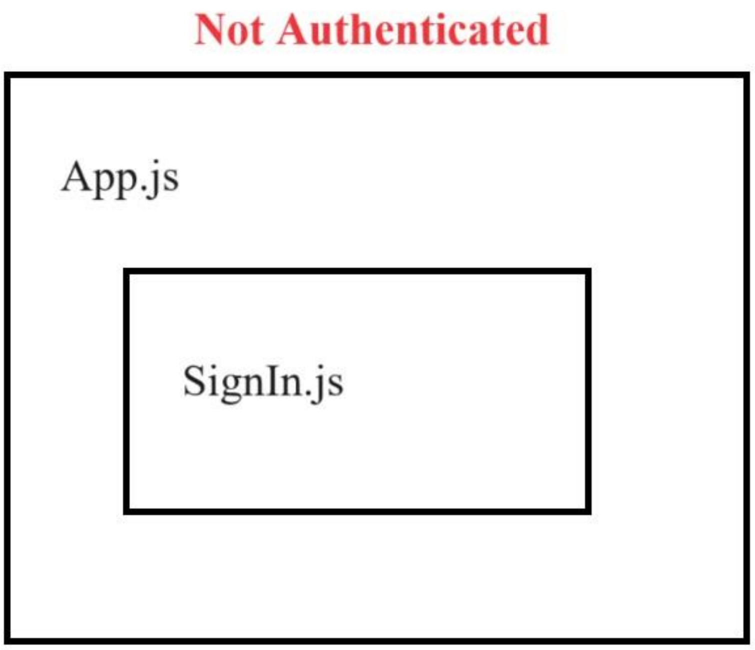


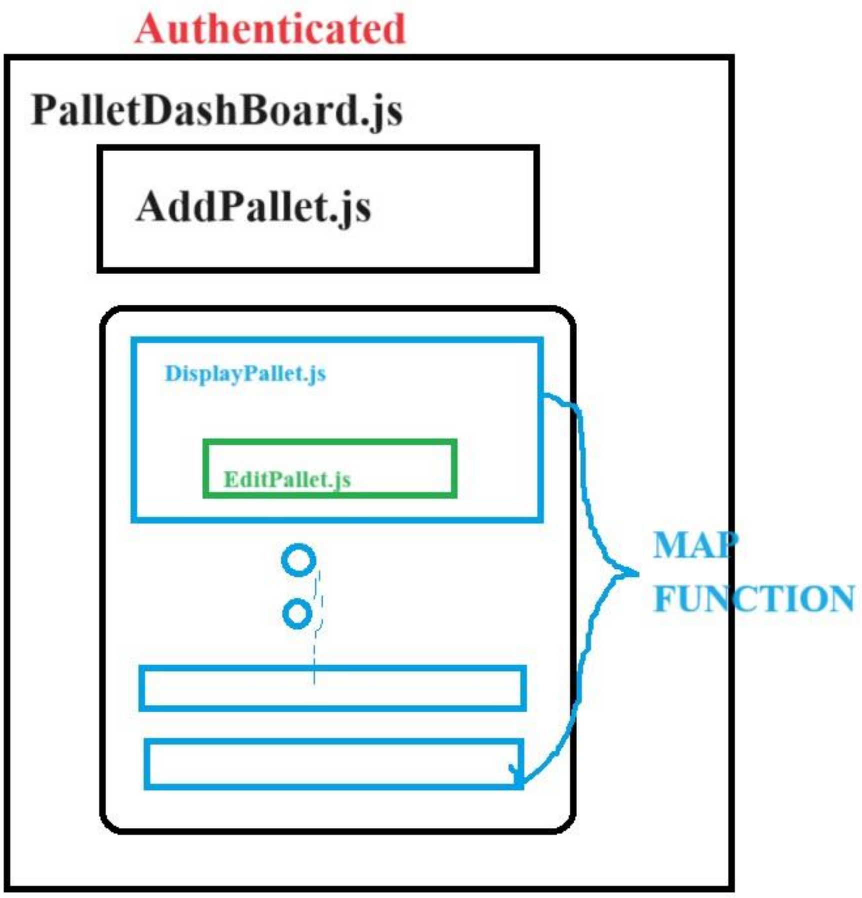
```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyCxxbCOI7ADmDISsIBWMB2AbXx61glYsP4",
  authDomain: "topic10-1877e.firebaseapp.com",
  projectId: "topic10-1877e",
  storageBucket: "topic10-1877e.appspot.com",
  messagingSenderId: "1061157216215",
  appId: "1:1061157216215:web:85d4f870719f1c2971f592",
  measurementId: "G-8XK51TBZDT"
};

// Initialize Firebase
```



```
// Import the functions you need from the SDKs you need
import { initializeApp, getApp } from "firebase/app";
import { getFirestore, initializeFirestore } from "firebase/firestore";


const firebaseConfig = {

  // you will need to generate this within Firebase yourself.
};


let firebaseApp;
try {
```

## 7. Overall Schematic





# PART 3 – Firebase Function

## CS385 Firebase Function

This is a very easy project that realised the full-stack function of C-R-U-D, by using Firebase and React. There are only 8 JavaScript files and 1 PNG file in this project.

The reason why we chose Firebase to connect with React is because we **don't need** to set up **Backend** program.

In this part, we need to allow a user to

- **CREATE - Add a pallet for shipping (add to our Firestore database)** – Allow the user to specify the goods on the pallet and the overall weight (KG).
- **READ - Display all pallets** in our warehouse**.**
- **UPDATE - Edit a pallet** – allow the user to change the details for an existing pallet in our Firestore database.
- **DELETE - Delete a pallet** – remove this pallet from our Firestore database

The following section explains the function that firebase used in each component.

## App.js

The first page of this program, it depends whether **theAuthUser** has logged in successfully or not.

It determines the next page to display by whether the user has authenticaded.

**Success => PalletDashboard + SignOut**

**Failed => Signin.js**

Authentication status is failed by default. **useState(null)**

## fbconfig.js

### `initializeApp(firebaseConfig)`

Initializes the **Firebase Application instance** according to the parameters defined by firebaseConfig,

### `getApp()`,

Get the Firebase application instance in your project.

After we have an application instance, we can access **Firebase authentication**, **database operations, storage**, and other services.

```
const db = initializeFirestore(firebaseApp,
  {
      experimentalForceLongPolling : true
  });
```

`initializeFirestore(firebaseApp)` 根据获取好的 firebase **应用程序实例**，初始化 Firebase 的**实时数据库服务**。这样 React 中的 Firebase 应用程序能够使用实时数据库服务。

Initialize Firebase's **real-time database service** according to the acquired **Firebase Application instance**. This allows the Firebase application in React to use the real-time database service.

### `experimentalForceLongPolling`

Forces the use of **long polling** as the transport method for communicating with the **Firestore server.**

Long polling is a technique for implementing real-time communication, which simulates real-time by means of a continuous connection between the client and the server. Typically, the Firebase SDK automatically selects the best transport mechanism (such as **WebSockets**) based on the environment and browser support. However, WebSockets may not work due to network limitations or environment-specific restrictions in some cases. In those cases, experimentalForceLongPolling can be used to force the connection between React and Firestore Server.

## Signin.js

### `getAuth(firebaseApp)`

Gets an **instance** of the **Authentication** service, passing an initialized Firebase application -- **firebaseApp**, as an argument.

### `e.preventDefault()`

Many elements and events have default behavior in the browser. For example, forms often try to submit to the server when the user clicks the submit button. Disabling the default submit behavior of a form gives you complete control over the event handling. This method is useful for handling link click events, form submit events, and so on.

### `signInWithEmailAndPassword(auth, email, password)`

This is the function used in the **Firebase Authentication Service** for **user login via email and password**。

There are three parameters that needs to be passed:

- **auth:** An **instance of the authentication service**, which can be obtained via the **getAuth** method.
- **email:** user's email
- **password:** user's password

If the authentication **succeeds**, the callback function in the **then()** receives a **UserCredential** object which contains the user's information.

If the authentication fails, the callback function in the catch() receives an object which contains an **error message.**

### `createUserWithEmailAndPassword(auth, email, password)`

This is the function used in the **Firebase Authentication Service** for **user registration via email and password**。

There are three parameters that needs to be passed:

- **auth:** An **instance of the authentication service**, which can be obtained via the **getAuth** method.
- **email:** user's email
- **password:** user's password

## SignOut.js

`SignOut()`

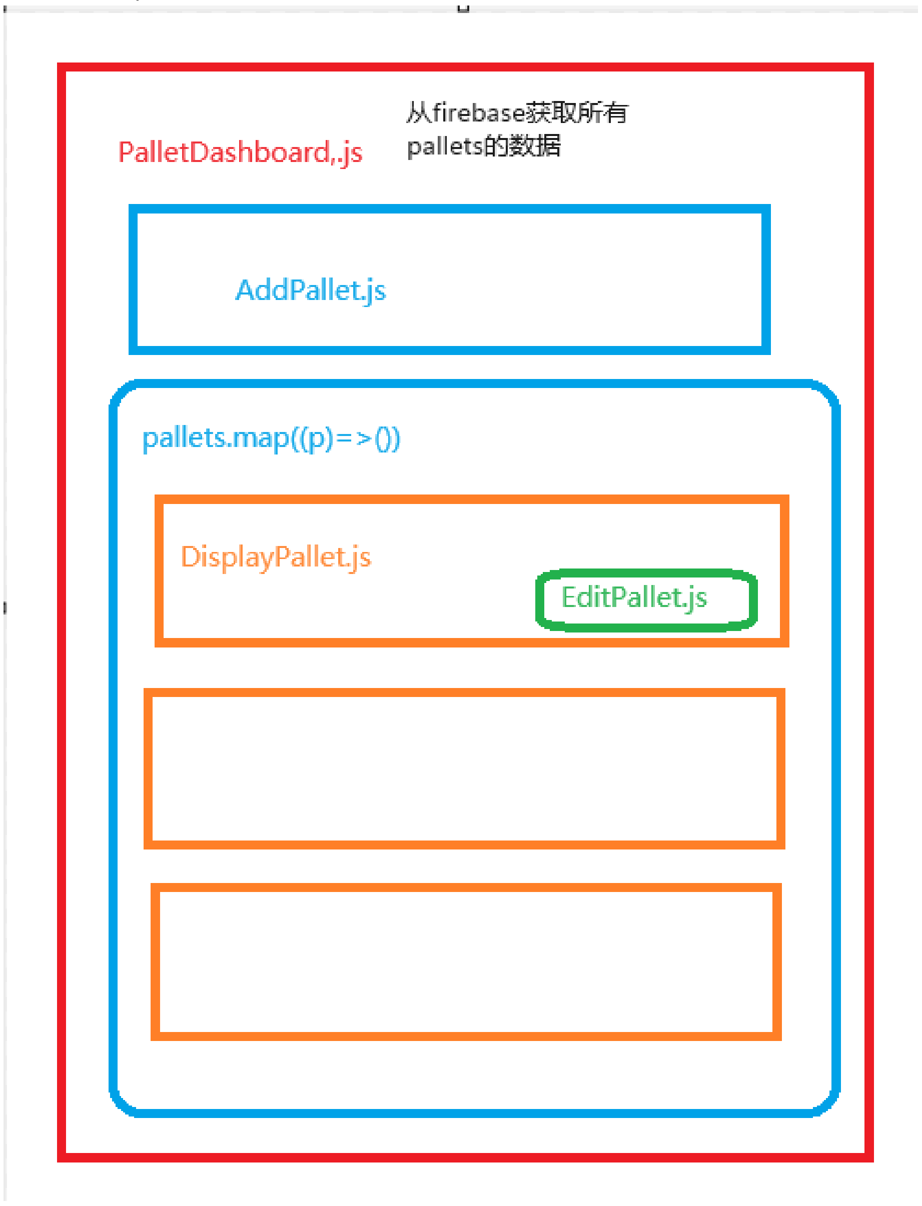
This function is used to **log out the current use**r in the Firebase Authentication Service. It serves to **terminate the current user's session** and log them out of the application.

## SignUp.js

It has a button to invoke **createUserWithEmailAndPassword** in **SignIn.js**

## PalletDashboard.js

Get all the pallets information from Firestore Database.

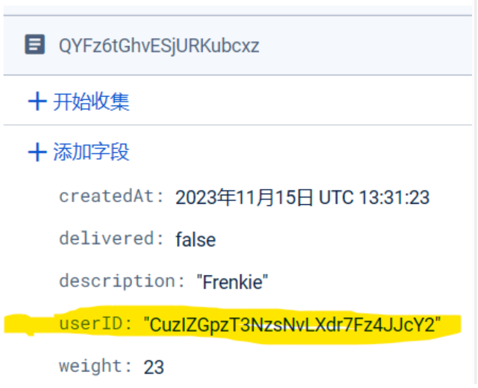


## `collection(query, (snapshot)=>{})`

```
const q = query(
    collection(db, "pallets"),
    where("userID", "==", currentUser.uid)
  );
```

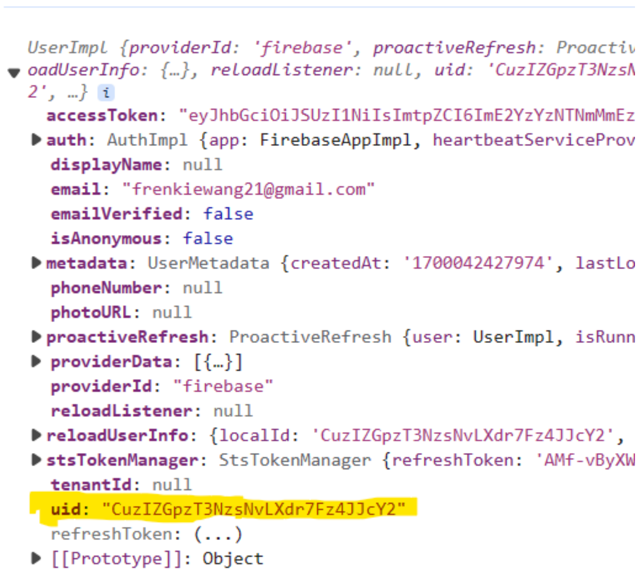
In Firebase's Firestore, the query function is used to create and execute queries against collections in your Firestore database.

It the above code, Firebase will find a **database** named **"db"** with the **collection** named **"pallets"**, and it must comply with the user login authentication.



```
QYFz6tGhvESjURKubcxz
+ 开始收集
+ 添加字段
  createdAt: 2023年11月15日 UTC 13:31:23
  delivered: false
  description: "Frenkie"
  userID: "CuzIZGpzT3NzsNvLXdr7Fz4JJcY2"
  weight: 23
```

**db.userID**



```
UserImpl {providerId: 'firebase', proactiveRefresh: Proactiv
▼ oadUserInfo: {…}, reloadListener: null, uid: 'CuzIZGpzT3NzsN
2', …} i
   accessToken: "eyJhbGciOiJSUzI1NiIsImtpZCI6ImE2YzYzNTNmMmEz
 ▶ auth: AuthImpl {app: FirebaseAppImpl, heartbeatServiceProv
   displayName: null
   email: "frenkiewang21@gmail.com"
   emailVerified: false
   isAnonymous: false
 ▶ metadata: UserMetadata {createdAt: '1700042427974', lastLo
   phoneNumber: null
   photoURL: null
 ▶ proactiveRefresh: ProactiveRefresh {user: UserImpl, isRunn
 ▶ providerData: [{…}]
   providerId: "firebase"
   reloadListener: null
 ▶ reloadUserInfo: {localId: 'CuzIZGpzT3NzsNvLXdr7Fz4JJcY2',
 ▶ stsTokenManager: StsTokenManager {refreshToken: 'AMf-vByXW
   tenantId: null
   uid: "CuzIZGpzT3NzsNvLXdr7Fz4JJcY2"
   refreshToken: (...)
 ▶ [[Prototype]]: Object
```

**currentUser.uid**

## `onsnapshot()`

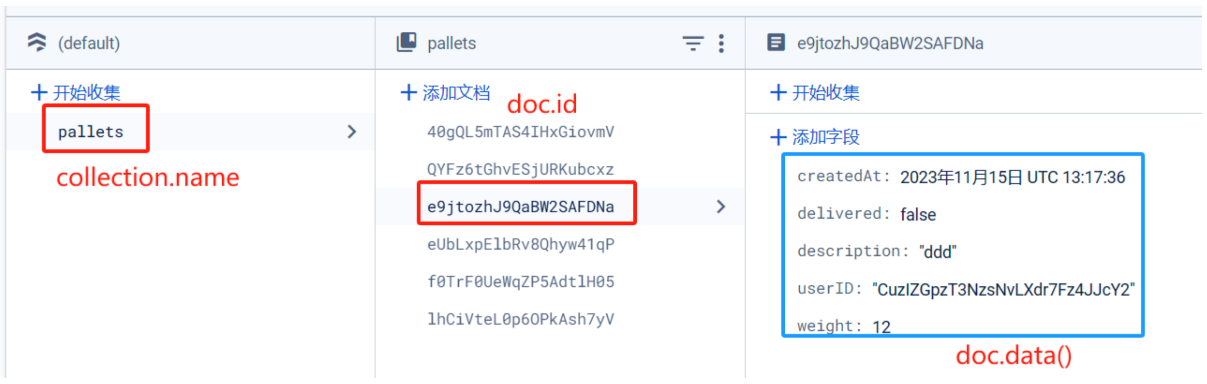
It is the function used by Firebase to listen to document and collection changes in real time. **If a document/collection is changed (created, updated or deleted)**, the listener is triggered and exposed to the corresponding change data.

```
querySnapshot.forEach((doc) => {
        palletsArrayFromFirebase.push({ ...doc.data(), id: doc.id });
```

- **doc.data()** - the content of the document
- **doc.id** - the id of the document

***Collection name***

- ***document.id***
- - ***document content***



(default)
+ 开始收集
pallets

pallets
+ 添加文档
40gQL5mTAS4IHxGiovmV
QYFz6tGhvESjURKubcxz
e9jtozhJ9QaBW2SAFDNa
eUbLxpElbRv8Qhyw41qP
f0TrF0UeWqZP5AdtlH05
lhCiVteL0p60PkAsh7yV

e9jtozhJ9QaBW2SAFDNa
+ 开始收集
+ 添加字段
createdAt: 2023年11月15日 UTC 13:17:36
delivered: false
description: "ddd"
userID: "CuzIZGpzT3NzsNvLXdr7Fz4JJcY2"
weight: 12

## DisplayPallet.js

### deleteDoc(doc(db, collection, id))

```
deleteDoc(doc(db, "pallets", id));
```

This function is a call that **removes the specified document** from the "pallets" collection **according to the ID of the document**.

- **db** - represents a Firebase database instance.
- "**pallets"** - the name of the collection.
- **id** - the **ID** of the document to be deleted

### updateDoc(doc(db, collection, id),newData)

```
    await updateDoc(doc(db, "pallets", docu.id), {
      delivered: !docu.delivered
    });
```

This is a function used to update the data of the **specified document**. According to **docu.id,** the updated data includes:

**description：** remain the same

**weight：** remain the same

**delivered：** change the data according to the input

**createdAt：** remain the same

**userID：** remain the same

The function of **updateDoc** can only update some of the fields while leaving others unchanged.

## AddPallet.js

### `form onSubmit={handleSubmit}>`

A <form> is an HTML element used to create a form that can contain input fields, buttons, and other elements. When the user fills out the form and submits it, data can be sent or certain actions can be triggered.

- **onSubmit:**

**onSubmit** is an event listener property used to catch form submission events.

The **onSubmit event** is triggered when the user submits the form (usually by clicking the submit button or pressing the enter key).

- **handleSubmit:**

handleSubmit is a function that defines the action that should be performed when a form is submitted.In React, this function is typically used to process form data, such as validating input, sending data to the server, or updating the application state.

In React, form elements are typically bound to the component's **State**, which makes it easy to collect and manage user input data. The handleSubmit function accesses the **State** data and uses it when the user submits the form.

### `addDoc(collection(db, "pallets"), newData)`

It is a function in Firebase to **add a new document** in the specified collection. There are 2 parameters to be passed:

- **db** - represents a Firebase database instance.
- **"pallets**" – nane of the collection.
- **newData – an object contains the data that needs to be stored in the new document.**

**Auto-generated Document ID:** When a document is added using **addDoc**, Firestore automatically generates a unique ID for the new document, as opposed to manually specifying the document ID (such as setDoc).

**Aynchronours function:** addDoc is an asynchronous function that returns a Promise. you can use the .then() and .catch() methods to handle the success or failure of the add operation.

## EditPallet.js

```
await updateDoc(doc(db, "pallets", palletToEdit.id), {
  description,
  weight,
  delivered: palletToEdit.delivered,
  createdAt: palletToEdit.createdAt,
  userID: palletToEdit.userID
});
```

This code uses the Firebase Firestore service and shows how to asynchronously update a specific document in the Firestore database:

1) **Asynchronous update operation:** the keyword **“await”** is used to wait for an asynchronous operation to complete. This means that the code waits for the updateDoc function to complete the update operation before continuing with the code that follows. It is often used to handle time-consuming network requests such as database operations.

2) **doc(db, "pallets", palletToEdit.id)**

- **db** - represents a Firebase database instance.
- "**pallets"** - the name of the collection.
- **palletToEdit.id** - the **ID** of the document to be updated
- **newData** - The object to be updated and the key in the object to be updated. If the key is specified in newData, this function will assign the new value to that key in the object. If the key is not specified in newData, the value of the key remains the same.

3) **Update data:** This is a function used to update the data of the specified document. According to palletToEdit.id, the updated data includes:

**description：** change the data according to the input

**weight：** change the data according to the input

**delivered：** remain the same

**createdAt：** remain the same

**userID：** remain the same

The function of **updateDoc** can only update some of the fields while leaving others unchanged.