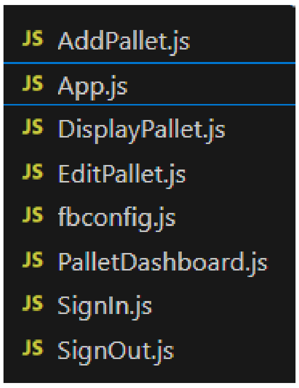
Using Firebase – The CS385 Shipping Comany

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?

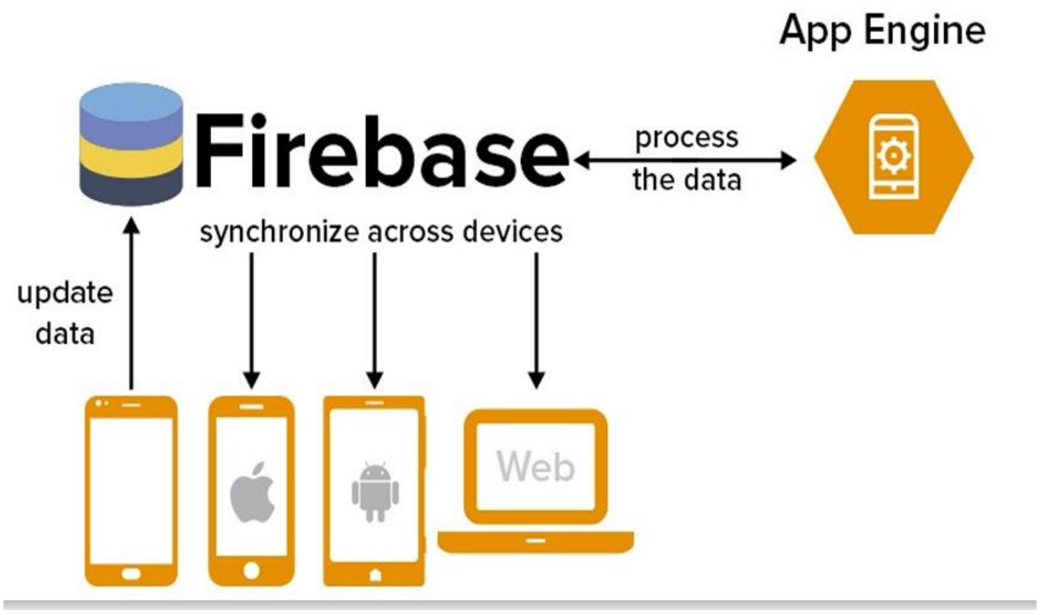
Authorization: What are you allowed to do?

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**.

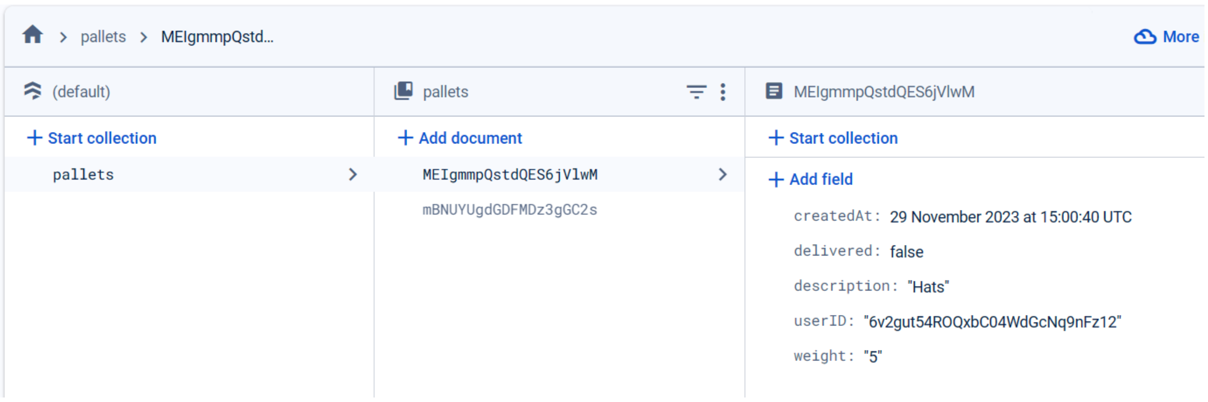
Each document (object) contains **a set of key-value pairs.** All documents (objects) must be stored in collections. **Collections and documents** are created **implicitly** in Cloud Firestore.

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

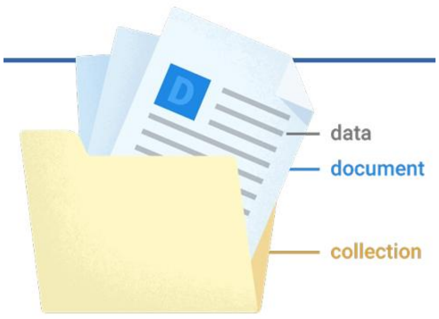
MEIgmmpQstdQES6jVlwM
mBNUYUgdGDFMDz3gGC2s

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

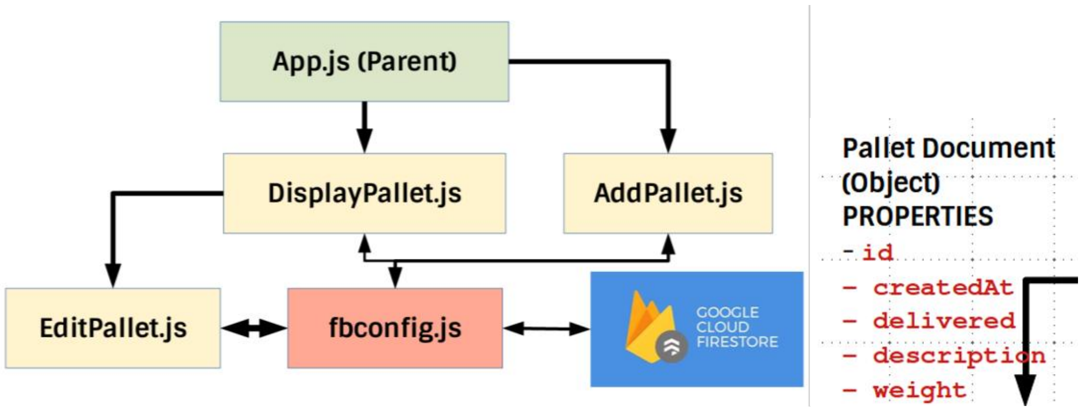
- **Add a pallet for shipping (add to our Firestore database)** – Allow the user to specify the goods on the pallet and the overall weight (KG).
- **Display all pallets** in our warehouse**.**
- **Delete a pallet** – remove this pallet from our Firestore database
- **Edit a pallet** – allow the user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



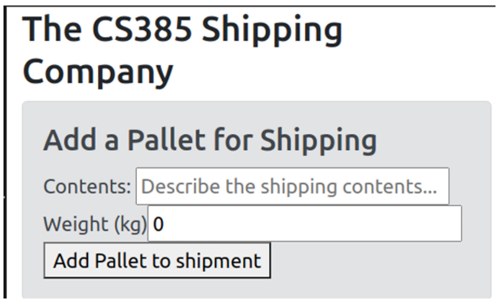
The **App.js** component is the parent. The other components are all considered as child components. The **fbconfig.js** file provides the configuration information to allow the components to access the Firestore database for this application.

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

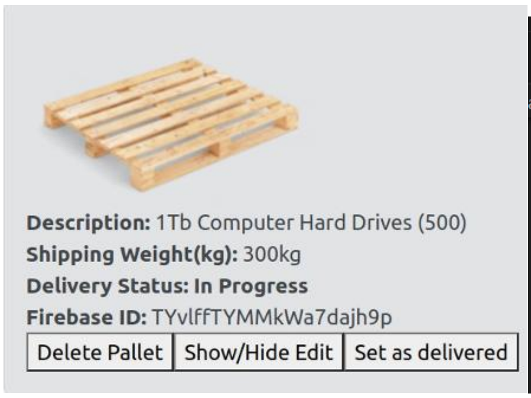
Add a Pallet for Shipping

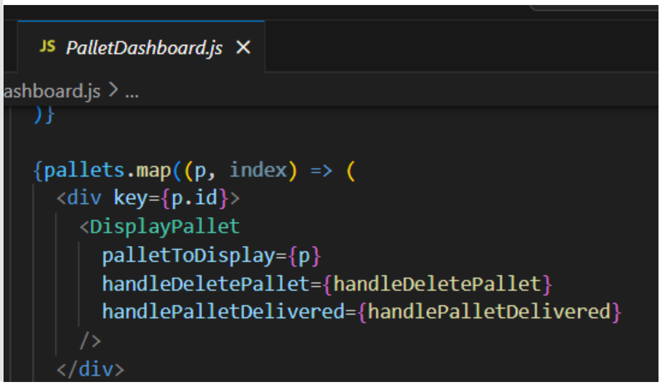
Contents: Describe the shipping contents...

Weight (kg) 0

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



```
)}

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

*Use map function to render each **DisplayPallets.js** 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:

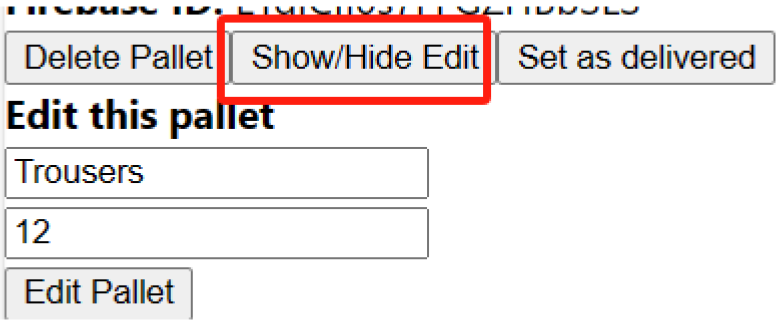
- **UserID:**
- **Description**
- **Shipping Weight(kg)**
- **Delivery Status**
- **Firebase ID:**

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.

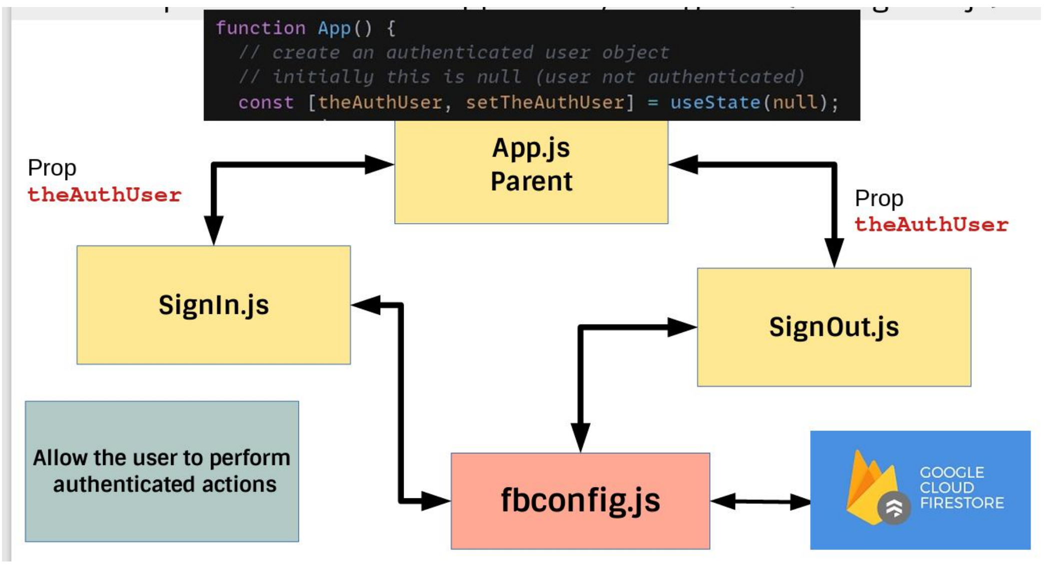
5. Part B - Authentication

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.

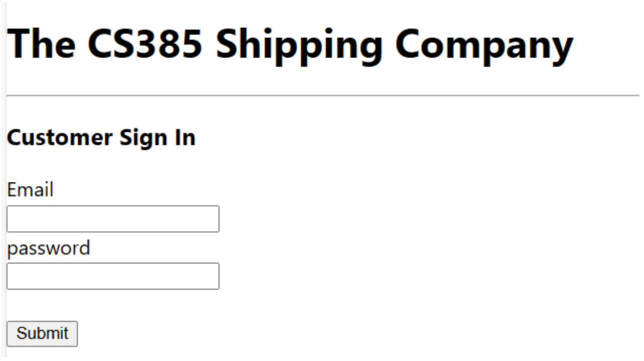
App.js

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

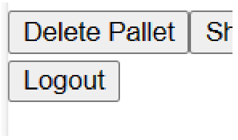
Email

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.

Further job to do:

Implement a function to allow users to sign up for an account or to have their password reset.

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)**

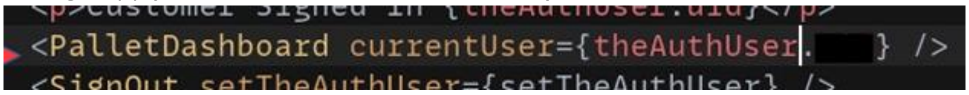
After we finished Part A and Part B, we need to use the **Authentication from Part B** to control access to **Pallet Dashboard from Part A.**

Informally, when **a user** logons 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.**

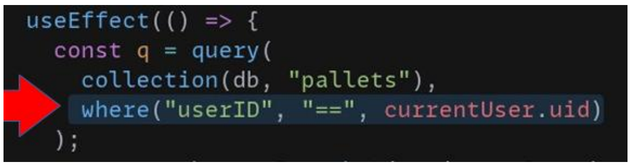
STEPS

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



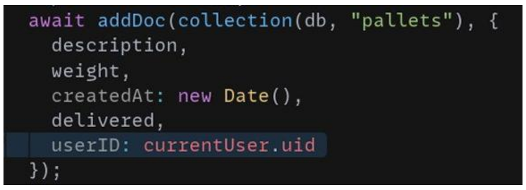
```
<p>Customer Signed in {theAuthUser.uid}</p>
<PalletDashboard currentUser={theAuthUser.[redacted]} />
<SignOut setTheAuthUser={setTheAuthUser} />
```

2) Select Collection by **userID.**

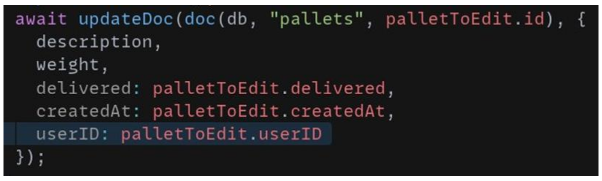


```
useEffect(() => {
  const q = query(
    collection(db, "pallets"),
    where("userID", "==", currentUser.uid)
  );
```

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

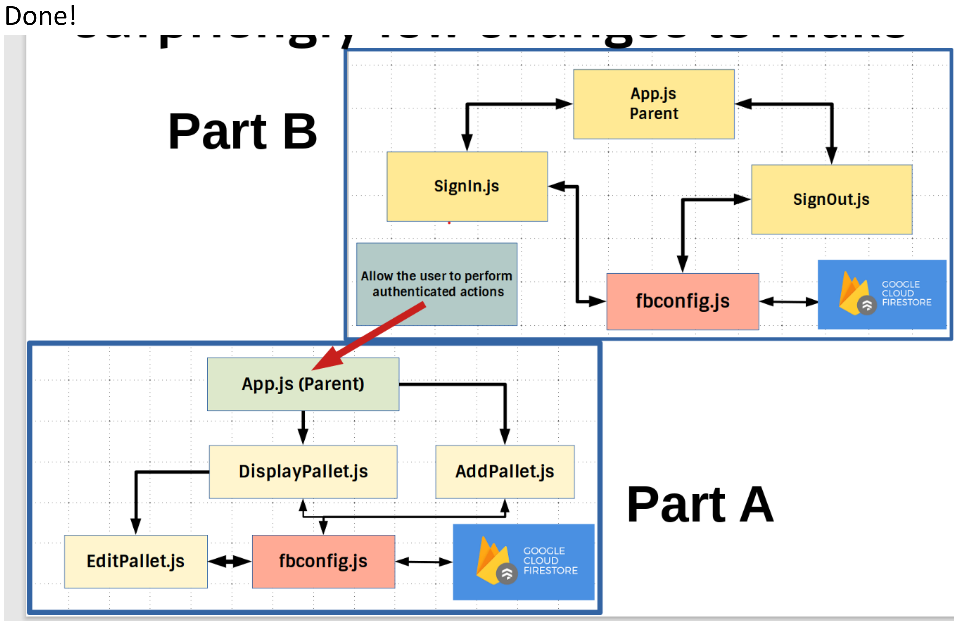


```
await addDoc(collection(db, "pallets"), {
  description,
  weight,
  createdAt: new Date(),
  delivered,
  userID: currentUser.uid
});
```



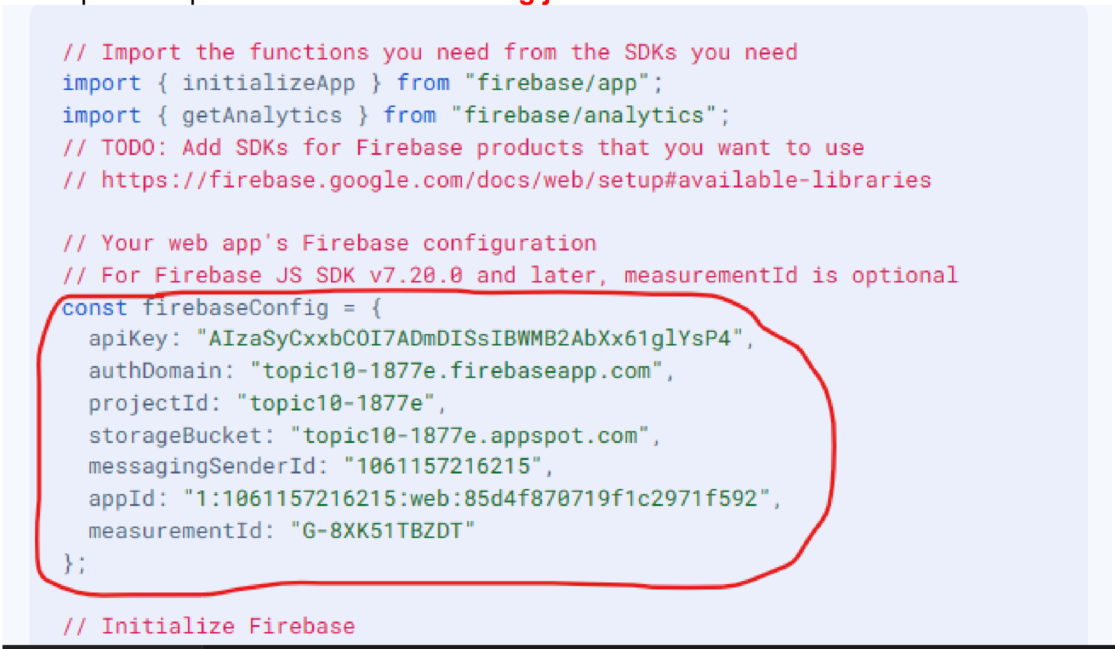
```
await updateDoc(doc(db, "pallets", palletToEdit.id), {
  description,
  weight,
  delivered: palletToEdit.delivered,
  createdAt: palletToEdit.createdAt,
  userID: palletToEdit.userID
});
```

4) Done!

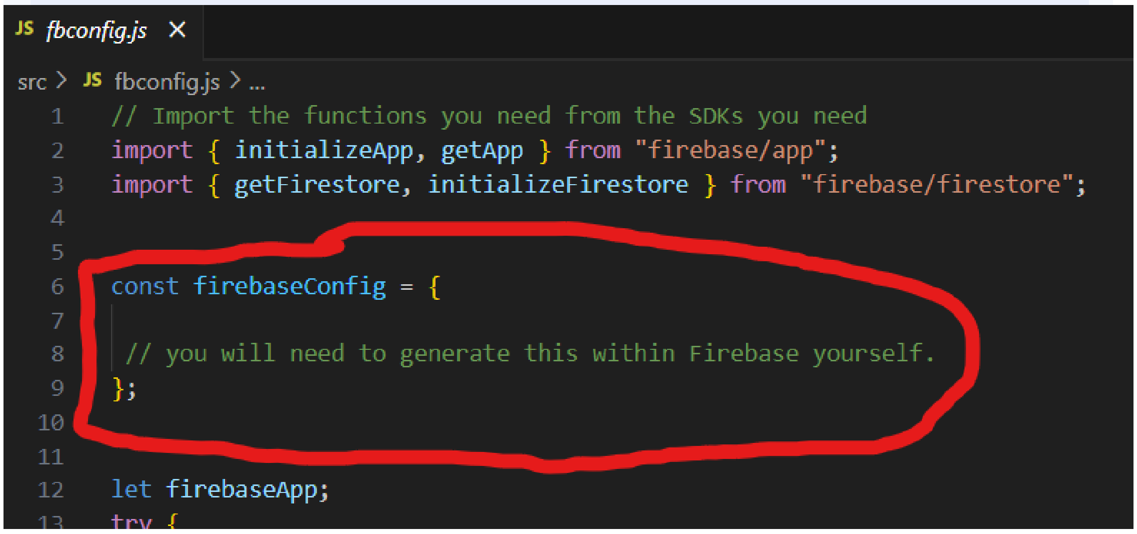


fbconfig.js

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
```

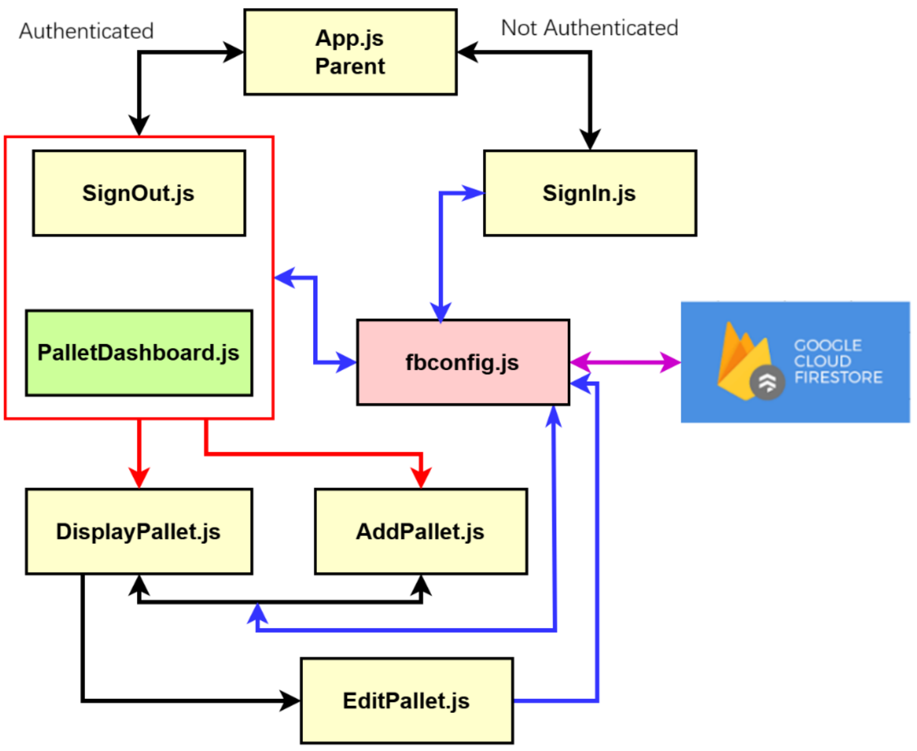


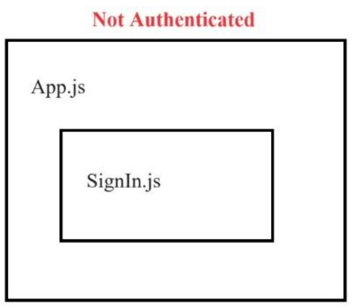
fbconfig.js ×

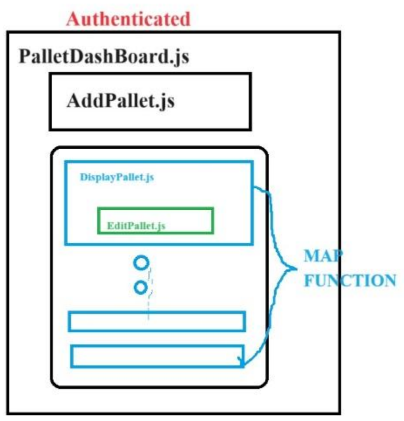
src > fbconfig.js > ...

```
// 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 {
```

7. Overall Schematic


Authenticated
App.js
Parent
Not Authenticated
SignOut.js
SignIn.js
PalletDashboard.js
fbconfig.js
GOOGLE
CLOUD
FIRESTORE
DisplayPallet.js
AddPallet.js
EditPallet.js


Not Authenticated
App.js
SignIn.js


Authenticated
PalletDashBoard.js
AddPallet.js
DisplayPallet.js
EditPallet.js
MAP
FUNCTION