CS385 Learning Note文本

描述已自动生成文本

描述已自动生成图示

描述已自动生成图示

描述已自动生成图示

描述已自动生成

文本

描述已自动生成

# App.js

登录的第一个页面

根据theAuthUser是否登录验证成功，决定接下来显示的页面

**验证成功 => PalletDashboard + SignOut**

**验证失败 => Signin.js**

默认为验证失败。**useState(null)**

# fbconfig.js

**initializeApp(firebaseConfig)** 根据firebaseConfig定义好的参数，初始化 Firebase **应用程序实例**。

**getApp()**，**获取**到项目中Firebase的 **应用程序实例.**

得到应用程序实例以后，可以访问 Firebase 的身份验证、数据库操作、存储和其他一些服务.

const db = initializeFirestore(firebaseApp,

  {

      experimentalForceLongPolling : true

  });

**initializeApp(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。

**experimentalForceLongPolling**

强制使用**长轮询（long polling）**作为与 Firestore 服务器通信的传输机制。

长轮询是一种实现实时通信的技术，它通过**客户端与服务器之间的持续连接**来模拟实时性。

通常，**Firebase SDK 会根据环境和浏览器支持自动选择最佳的传输机制（如 WebSockets）。**然而，在某些情况下，由于网络限制或特定环境的限制，WebSockets 可能不可用。在这种情况下，可以使用 experimentalForceLongPolling 来强制使用长轮询。

# Signin.js

**getAuth(firebaseApp)**

获取**身份验证（Authentication）服务实例**，传递一个初始化过的 Firebase **应用程序实例** firebaseApp 作为参数。

**e.preventDefault()**  **取消事件的默认行为**

在浏览器中，许多元素和事件都有默认的行为。例如，当用户点击提交按钮时，表单通常会尝试提交到服务器。阻止了表单的默认提交行为，可以让你完全控制事件的处理逻辑，而不受浏览器默认行为的影响。这个方法对于处理链接的点击事件、表单提交事件等都非常有用。

**signInWithEmailAndPassword(auth, email, password)**

是 Firebase 身份验证服务中用于通过**电子邮件和密码**进行用户登录的方法。这个方法需要传递三个参数：

**auth: 身份验证服务的实例**，可以通过 getAuth 方法获取。

**email: 用户的电子邮件地址**。

**password: 用户的密码**。

如果登录**成功**，then 函数中的回调将获得一个包含**用户信息**的 **UserCredential** 对象；

如果登录**失败**，catch 函数中的回调将接收一个包含**错误信息的对象**。

# Signout.js

**SignOut()**

signOut() 是 Firebase 身份验证服务中用于**登出（注销）当前用户**的方法。它的作用是终止当前用户的会话并将其从应用程序中注销。

# PalletDashboard.js

图形用户界面

描述已自动生成

**collection(query, (snapshot)=>{})**

 collection(db, "pallets"),

Firebase寻找名为db的数据库，在数据库中寻找名为pallets的表名

**onsnapshot()**

Firebase用于实时监听文档和集合变化的方法。如果文档/集合发生变化（创建、更新或者删除），监听器会被触发，并接触到相应的变化数据。

querySnapshot.forEach((doc) => {

        palletsArrayFromFirebase.push({ ...doc.data(), id: doc.id });

doc.data() 表格元素的内容，doc.id 表格元素的id

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 文本

描述已自动生成

const q = query(

      collection(db, "pallets"),

      where("userID", "==", currentUser.uid)

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成db.userID

图形用户界面, 文本, 应用程序

描述已自动生成 currentUser.uid

# DisplayPallet.js

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

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

是一个用来删除指定文档的函数调用，它会从 "pallets" 集合中删除ID为 id 的文档

- db 表示对Firebase数据库实例。

- "pallets" 是集合的名字。

- id 是要删除的文档ID

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

    await updateDoc(doc(db, "pallets", docu.id), {

      delivered: !docu.delivered

    });

用来更新数据库中特定文档的函数调用。这个语句通常用于**修改或添加文档中的字段，而不是完全替换整个文档。**

- db 表示对Firebase数据库实例。

-"pallets" 是集合的名字。

- id 是要删除的文档ID

-newData 包含要更新的对象和新值的对象

*要更新的字段名称：字段的新值*

# AddPallet.js

**form onSubmit={handleSubmit}>**

**<**form> 是一个HTML元素，用于创建一个表单，可以包含输入字段、按钮等元素。

用户填写表单并提交时，可以发送数据或触发某些动作。

**事件处理器 onSubmit：**

onSubmit 是一个事件监听属性，用于捕捉表单的提交事件。

当用户提交表单（通常是通过**点击提交按钮或按下回车键**）时，会触发 onSubmit 事件。

**处理函数 handleSubmit：**

handleSubmit 是一个函数，定义了当表单提交时应该执行的动作。

在React中，这个函数通常用于处理表单数据，例如验证输入、发送数据到服务器或更新应用状态。

重要的是，handleSubmit 函数中通常会包含 **event.preventDefault()** 方法。

这个方法阻止了表单的默认提交行为（即将数据发送到页面的URL）

，这样开发者就可以完全控制提交过程，包括异步提交数据到服务器。

**React 和状态管理：**

在React中，表单元素通常与组件的状态（state）绑定，这使得可以轻松地收集和管理用户输入的数据。

handleSubmit 函数可以访问这些状态数据，并在用户提交表单时使用它们。

**addDoc(collection(db, "pallets"), newData)**

addDoc 是一个Firestore方法，用于在指定的集合中添加一个新文档。

需要传递两个参数：一个是集合的引用，另一个是要添加的新文档的数据（newData）。

-db 是对Firestore数据库的引用。

-"pallets" 是集合的名称。

-newData 是一个对象，包含了你想在新文档中存储的数据。

**自动生成的文档ID：**当使用 addDoc 添加文档时，Firestore会自动为新文档生成一个唯一的ID。这与手动指定文档ID（例如使用 setDoc 方法）不同。

**异步操作**：addDoc 是一个异步函数，它返回一个Promise。你可以使用 .then() 和 .catch() 方法来处理添加操作的成功或失败情况。

用途：这个方法在需要向数据库中添加新数据时非常有用，例如用户注册时添加用户信息，或者添加新的订单信息等场景。

# EditPallet.js

    await updateDoc(doc(db, "pallets", palletToEdit.id), {

      description,

      weight,

      delivered: palletToEdit.delivered,

      createdAt: palletToEdit.createdAt,

      userID: palletToEdit.userID

    });

这段JavaScript代码是使用了Firebase Firestore服务的一个示例，它展示了如何异步更新Firestore数据库中的一个特定文档。以下是代码的详细解释和作用：

**1）异步更新操作**：await 关键字用于等待一个异步操作的完成。这意味着代码会等待 updateDoc 函数完成更新操作后再继续执行后面的代码。这是JavaScript的异步编程的一部分，通常用于**处理耗时的网络请求**，如数据库操作。

**2）文档引用：doc(db, "pallets", palletToEdit.id)** 创建了一个指向Firebase Firestore数据库中特定文档的引用。在这里：

- db 是对Firestore数据库的引用。

- "pallets" 是集合的名称。

- palletToEdit.id 是需要更新的文档的ID（唯一标识符）。

**3）更新数据：**updateDoc 是Firestore的一个方法，用于**更新指定文档的数据。**在这段代码中，它被用来更新 palletToEdit.id 对应的文档，更新的数据包括：

**description：新的描述字符串。**

**weight：新的重量值。**

**delivered：保持 palletToEdit.delivered 的当前值。**

**createdAt：保持 palletToEdit.createdAt 的当前值。**

**userID：保持 palletToEdit.userID 的当前值。**

这表明代码的意图是**更新部分字段，而保持其他字段不变。**

用途：这种更新操作通常用于修改数据库中的现有记录。例如，如果你正在开发一个物流系统，你可能需要更新货物的描述或重量，但同时保留其创建时间和用户ID等信息不变。

这段代码是一个很好的例子，展示了如何在现代JavaScript应用程序中使用Firebase来进行高效的数据管理和更新。使用await确保了代码的可读性和操作的正确性，特别是在涉及数据库交互的场景中。