# PART 1 - How to run this project?

## **Step 1: Register Firebase Account**

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 应用程序, Word

描述已自动生成

图片包含 图形用户界面

描述已自动生成

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, 应用程序, 聊天或短信

描述已自动生成

图形用户界面, 文本, 应用程序

中度可信度描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

x

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

It will take some time to deploy

图形用户界面, 文本

描述已自动生成

Deploy completed

图形用户界面, 文本, 应用程序, 网站

描述已自动生成

图形用户界面, 文本, 应用程序

中度可信度描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图片包含 图形用户界面

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

## **Step 2: Create Firebase Projects**

After register your account, you can see a Dashboard.

Click **“Add Project”** to create a Firebase Project.

图形用户界面, 应用程序

描述已自动生成

1. Type your project name

图形用户界面, 应用程序

中度可信度描述已自动生成

1. Continue

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

1. Select **Default Account for Firebase** and **Create Project**

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

1. Wait

应用程序

低可信度描述已自动生成

1. Project created completely.

图片包含 文本

描述已自动生成

## **Step 3: Add Firebase to App**

1. Add Firebase to Web App

手机屏幕的截图

描述已自动生成

1. Set **“Nick Name”** and **register App**.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

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

图形用户界面, 文本, 应用程序

描述已自动生成

1. Continue to the console!

图形用户界面, 文本, 应用程序

描述已自动生成

## **Step 4: Set User Authentication**

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

图形用户界面, 应用程序

描述已自动生成

2- Get Started

图形用户界面, 应用程序

中度可信度描述已自动生成

3- Click Sign-in Method

图片包含 日程表

描述已自动生成

4- There are many sign-in methods.

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

图形用户界面, 网站

描述已自动生成

1. Enable that method and save.

图形用户界面, 文本, 应用程序

描述已自动生成

1. Click **“Users”** in the menu, then **Add user.**

图形用户界面, 文本, 应用程序

描述已自动生成

1. Set the **email** and **password** in Authentication, then **Add User**

图形用户界面, 图示

中度可信度描述已自动生成

1. You can see that **User** is Added in Firebase Authentication

图形用户界面, 文本, 应用程序

描述已自动生成

## **Step 5: Check Firestore Database**

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

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

1. Create database

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

1. Set the Firebase location

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

图形用户界面, 文本, 应用程序

描述已自动生成

1. Click **Next**

图形用户界面, 应用程序

描述已自动生成

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

图形用户界面, 应用程序

描述已自动生成

1. Firestore Database created successfully

图形用户界面, 文本, 应用程序

描述已自动生成

1. Pay attention to the **Security Rules!**

图形用户界面, 应用程序

描述已自动生成

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.

图形用户界面, 文本, 应用程序

描述已自动生成

## **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.

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

文本

中度可信度描述已自动生成

1. 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**

图形用户界面, 文本, 应用程序

描述已自动生成

文本

描述已自动生成

1. 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.

文本, 徽标

描述已自动生成

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

文本

描述已自动生成

## **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/>

图形用户界面, 应用程序

描述已自动生成

1. Login

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

图形用户界面, 图示

中度可信度描述已自动生成

1. Add Pallet

Add two pallets, set the Pallet name and weight.

图形用户界面, 文本, 应用程序

描述已自动生成

You can see they could be seen in the WebPages.

文本

描述已自动生成文本

中度可信度描述已自动生成

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

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

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

1. 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.

、

# **AddressModal.js**

import React from 'react';

import styled from 'styled-components';

function TherapistAddress(props){

    return(

        <Modal onClose={props.closeModal}>

        <h4>{`Address of ${props.selectedTherapist.firstName} ${props.selectedTherapist.surName}`}</h4>

        <p>{props.selectedTherapist.homeAddress.addressLine1}</p>

        <p>{props.selectedTherapist.homeAddress.addressLine2}</p>

        <p>{props.selectedTherapist.homeAddress.town}</p>

        <p>{props.selectedTherapist.homeAddress.countyCity}</p>

        <p>{props.selectedTherapist.homeAddress.eircode}</p>

      </Modal>

    )

}

function Modal({ onClose, children }) {

    return (

      <StyledModal>

        <ModalContent>

          <CloseButton onClick={onClose}>&times;</CloseButton>

          {children}

        </ModalContent>

      </StyledModal>

    );

  }

export default TherapistAddress;

**Modal组件 – 模态对话框**

**一种用户界面元素，用于在应用程序的主内容之上显示次要内容，通常是为了捕获用户的注意力或要求用户做出决定。Modal 通常包含文本、表单元素或其他交互内容，并且可以通过关闭按钮或完成某些操作来关闭。它用于实现诸如确认对话框、信息提示、表单输入等交互方式。React 中的 Modal 组件可以是自定义的，也可以使用各种现成的库和框架提供的 Modal 组件。**

**&times;**

**在HTML中，&times; 是一个字符实体，用于表示乘号（×）**

**CSS文件部分 – Styled Components**

const **StyledModal** = styled.div`

  position: fixed; // 使 Modal 具有固定定位，这意味着它不会随着页面滚动而移动，而是固定在屏幕上。

  top: 0;  left: 0;  right: 0;  bottom: 0; // 将 Modal 拉伸到整个视口，使其覆盖整个屏幕，创建了一个背景遮罩。

  background-color: rgba(0, 0, 0, 0.5); // 设置了半透明的黑色背景，以遮挡背后的内容，创建了一个模糊效果，提高了 Modal 的可读性。

  display: flex;  align-items: center;  justify-content: center; // 使 Modal 内容在垂直和水平方向上都居中对齐，确保 Modal 在屏幕中间显示。

  z-index: 1000; // 设置了层级，确保 Modal 在其他页面元素之上，不会被其他元素遮挡。

`;

const **ModalContent** = styled.div`

  background-color: #fff; // 为 Modal 内容添加白色背景，使其在半透明背景上醒目可见。

  padding: 20px; // 为 Modal 内容添加内边距，增加内容与边框之间的间距，提高美观性。

  border-radius: 5px; // Modal 内容添加圆角边框，使其看起来更加友好。

  width: 80%;  max-width: 400px; // 控制 Modal 内容的宽度，可以根据需要进行调整。

CloseButton:

  position: relative; // 关键属性，用于定位 CloseButton

`;

const **CloseButton** = styled.span`

  cursor: pointer; // 手型光标

  position: absolute; // 将Close Button的定位设置为 Modal Content的绝对位置，以便可以精确控制其位置。

  top: 0;  right: 0; // 让 X 位于 ModalContent的右上方

  padding: 10px; // 给予一些间距，使得 X 不会贴着边框

  font-size: 1.5em; // 设置关闭按钮的字体大小，使其更加易于点击。

  line-height: 1; //将行距设置为1，使X更加紧凑

`;

**当一个元素的 position 设置为 relative，它将成为其子元素定位的参照物，如果这些子元素的 position 被设置为 absolute。这意味着绝对定位的子元素将相对于这个相对定位的父元素进行定位，而不是相对于整个页面或最近的其他定位元素。**

styled-components 是一个流行的库，用于**在React应用中编写和管理CSS**。主要用途和特点包括：

**1）CSS-in-JS:** styled-component**能在JavaScript文件中编写CSS样式**。

**2）组件级别的样式封装:** 每个styled-component都是一个独立的组件，它的样式仅适用于该组件，不会影响到其他组件。这有助于避免样式冲突，并且使得组件更易于重用和维护。

**3）动态样式:** styled-components 允许**根据组件的props或状态动态地更改样式**。这为根据用户交互或应用程序状态改变外观提供了极大的灵活性。

**4）样式复用:** 可以很容易地通过创建可重用的样式化组件来**共享样式。**

**5）无需担心命名冲突:** 库在背后生成唯一的类名，因此开发者无需担心CSS类名冲突的问题。