**Aim:** Implementation and integration with firebase backend and application deployment.

**Objective:**

To develop and deploy a robust application by leveraging Firebase backend services for authentication, database management, storage, and real-time updates, ensuring a seamless and scalable solution.

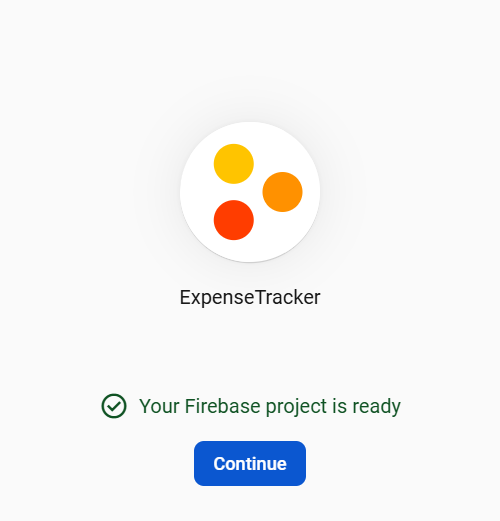
**Implementation:**

1. Firebase Backend Setup:
   * Configure Firebase project in the Firebase Console.
   * Enable necessary services: Authentication, Firestore/Realtime Database, Cloud Storage, and Hosting.
2. Authentication Integration:
   * Implement user authentication using Firebase Authentication with support for email/password, Google Sign-In, and other providers.
   * Secure authentication tokens for session management.
3. Database Management:
   * Design and structure Firestore/Realtime Database for efficient data storage and retrieval.
   * Implement security rules to control access based on user roles.
4. Cloud Storage Integration:
   * Use Firebase Cloud Storage to manage and serve user-generated content such as images or files.
   * Optimize upload/download workflows and ensure secure access.
5. Application Development:
   * Integrate Firebase SDKs into the application.
   * Use Firebase features for real-time updates, data synchronization, and push notifications.
6. Deployment:
   * Deploy the application using Firebase Hosting or other hosting platforms.
   * Ensure CI/CD pipelines are in place for continuous integration and deployment.

Implementation and integration with firebase backend and application deployment.

**1.1 Create Firebase Project**

1. Go to the Firebase Console.
2. Click on **Add Project** and follow the steps:
   * Provide a project name and select the country/region.
   * Disable Google Analytics for now (optional).

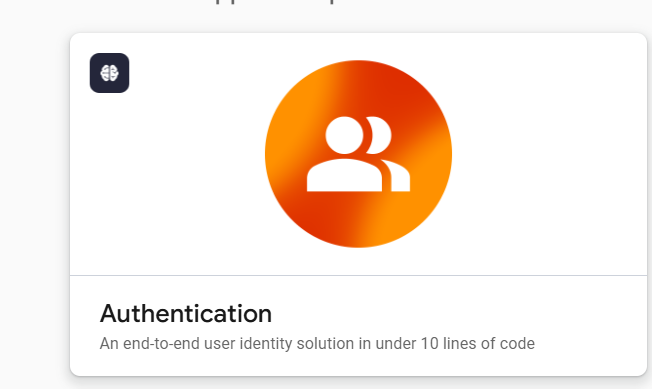


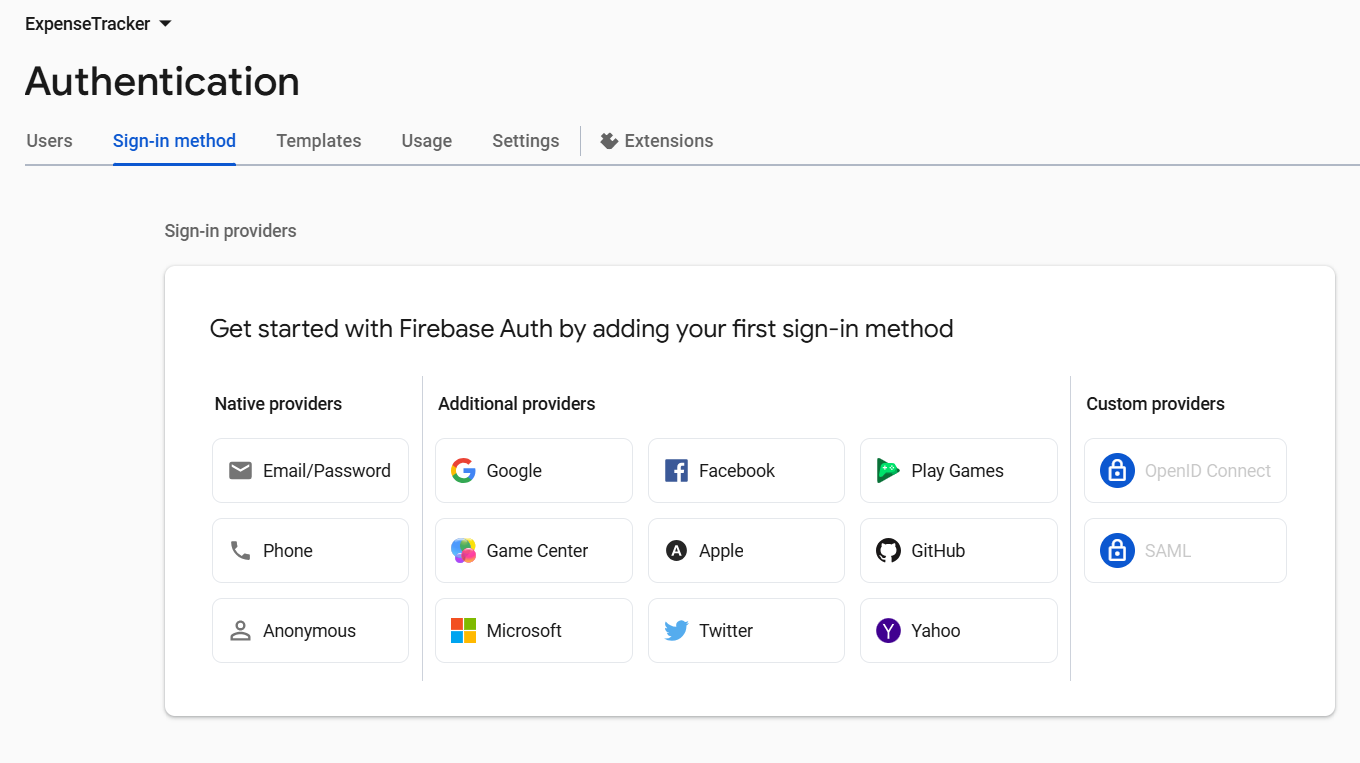
Sure! Here is a detailed step-by-step guide for developing the **Expense Tracker App with Firebase** using **.NET MAUI**.

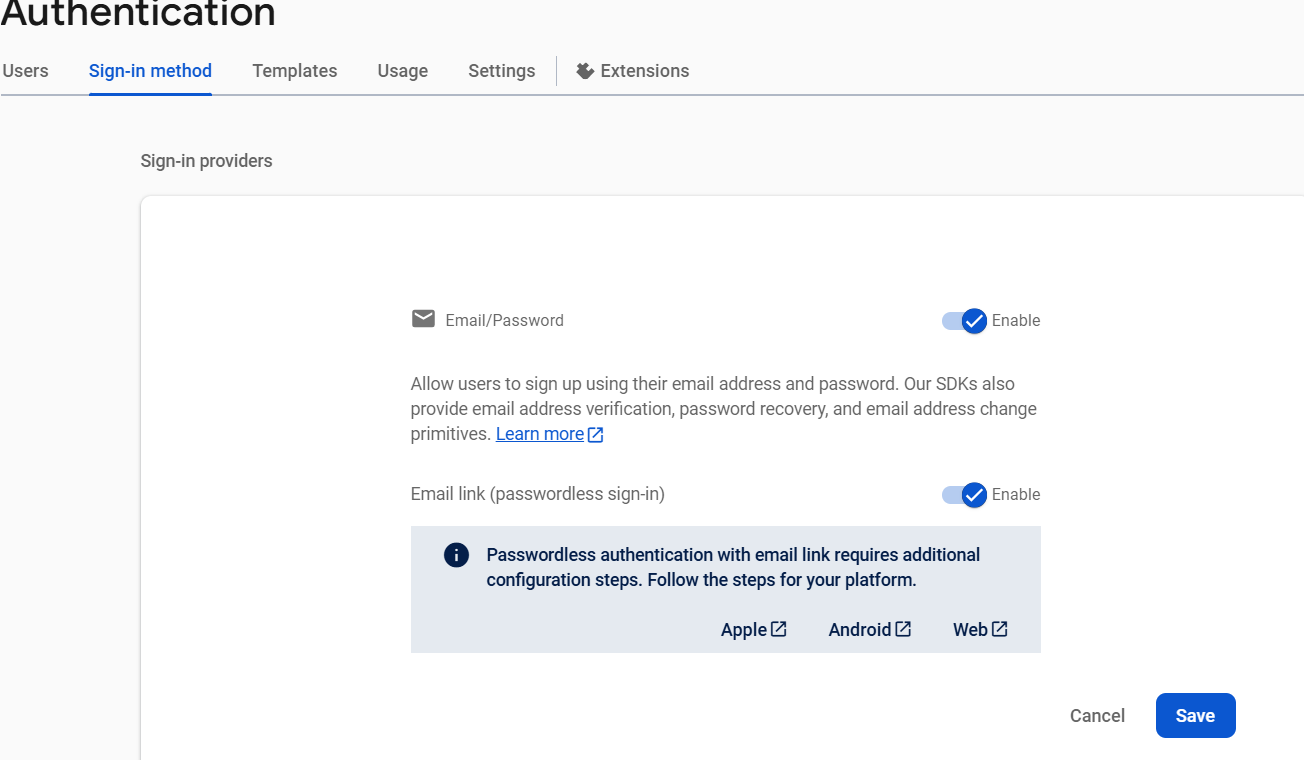
**Step 1: Setting Up Firebase Project**

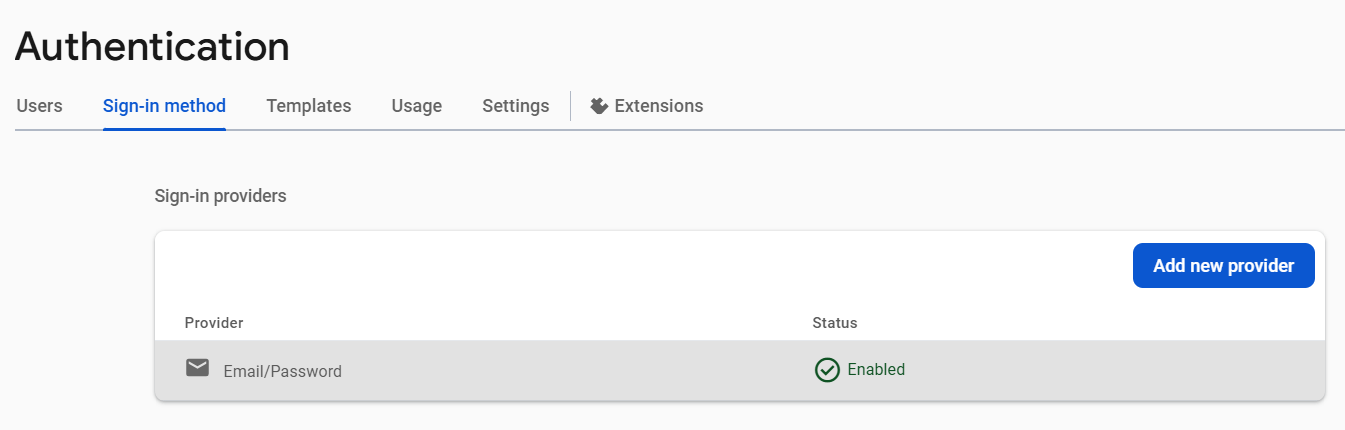
**1.2 Enable Firebase Authentication**

1. In the Firebase console, select **Authentication** from the left menu.
2. Under **Sign-in method**, enable **Email/Password** authentication



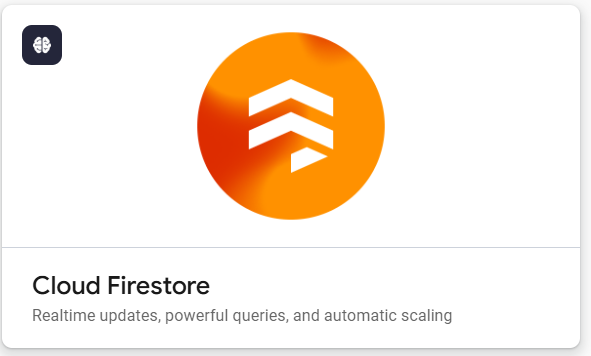


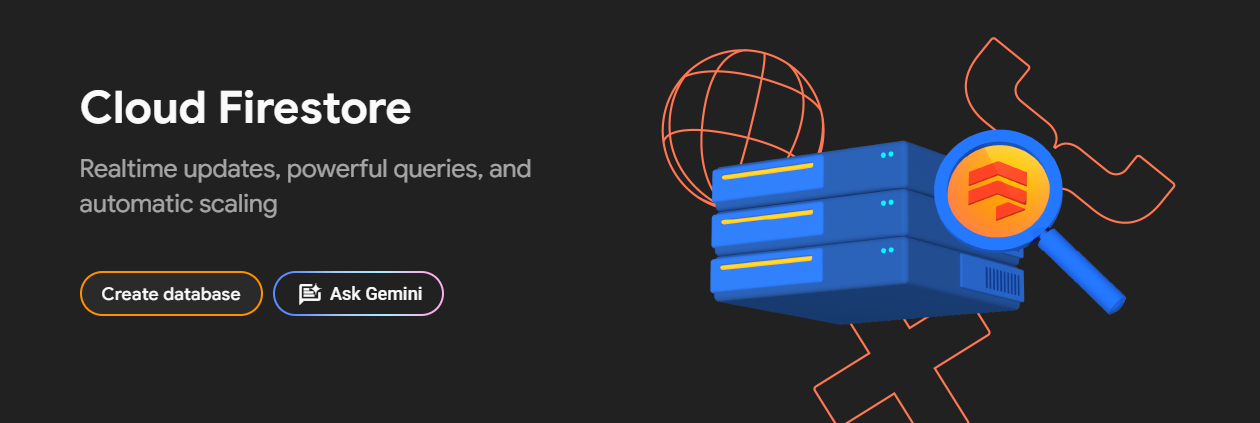


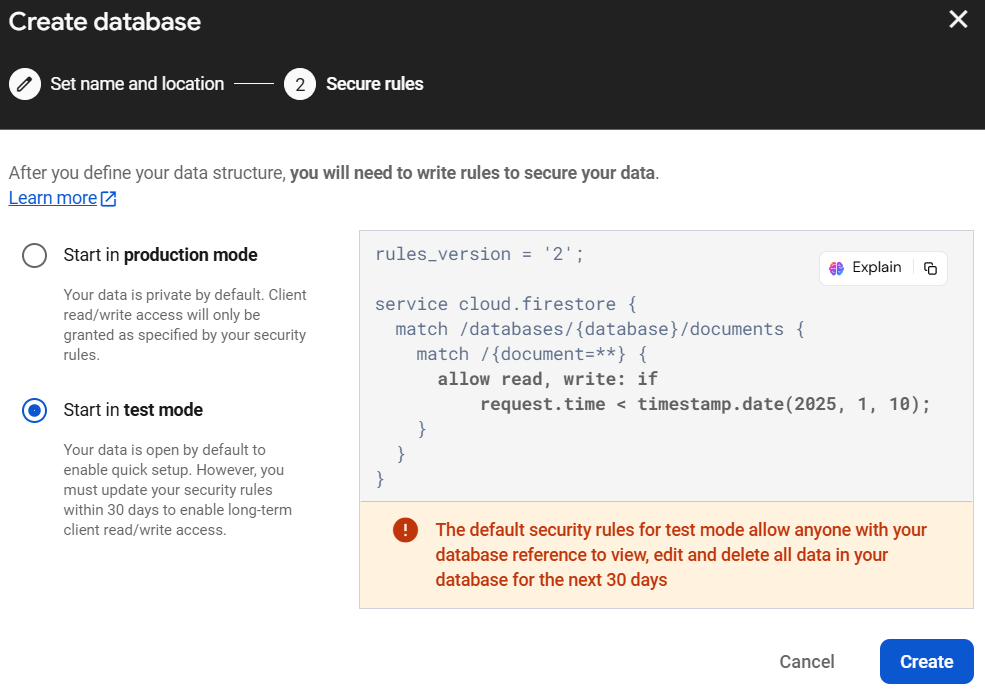


**1.3 Set Up Firebase Firestore**

1. In the Firebase console, go to **Firestore Database** from the left menu.
2. Click on **Create Database** and select **Start in test mode** (You can modify security rules later).
3. Select the region closest to your server location.







**1.4 Add Firebase SDK to .NET MAUI**

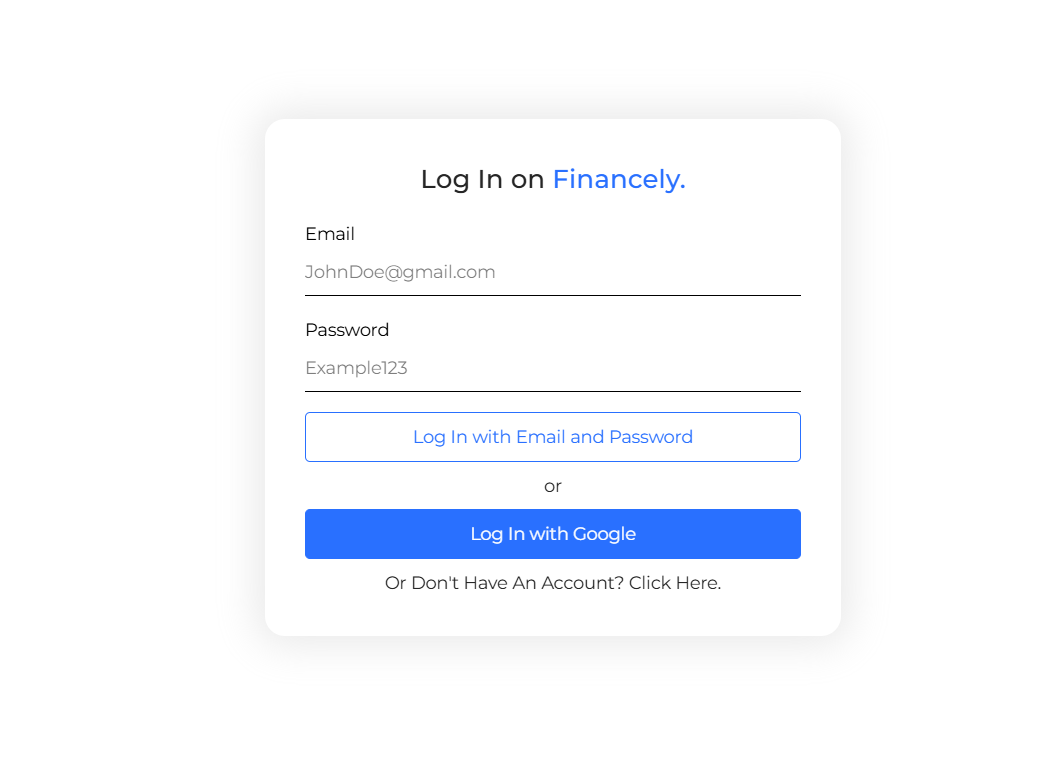
To integrate Firebase with your .NET MAUI project, you need to install Firebase SDK libraries.

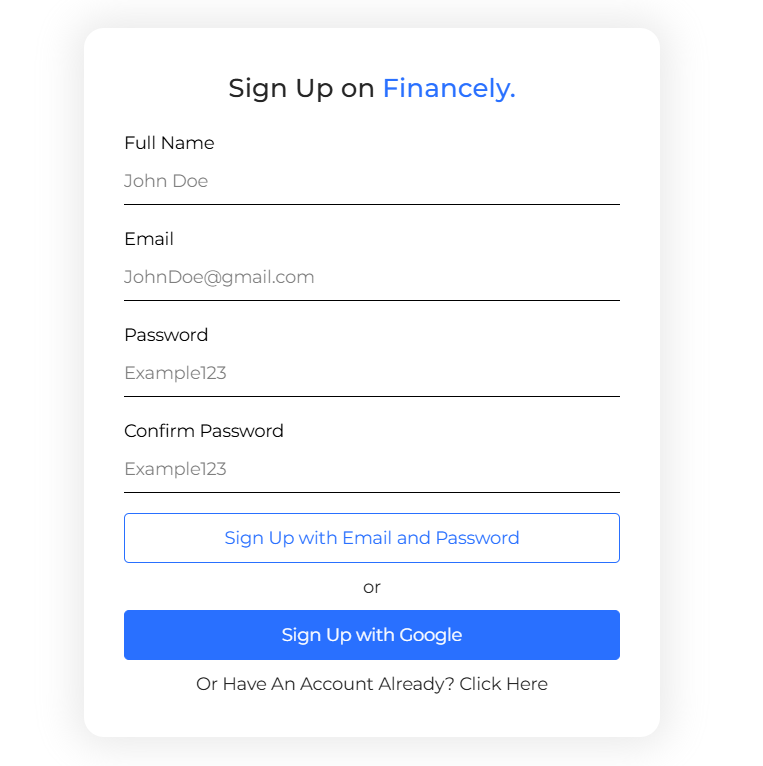
1. Open your **.NET MAUI** project in Visual Studio.
2. Go to **NuGet Package Manager** and install the following packages:
   * Firebase.Auth
   * Firebase.Firestore
   * Newtonsoft.Json

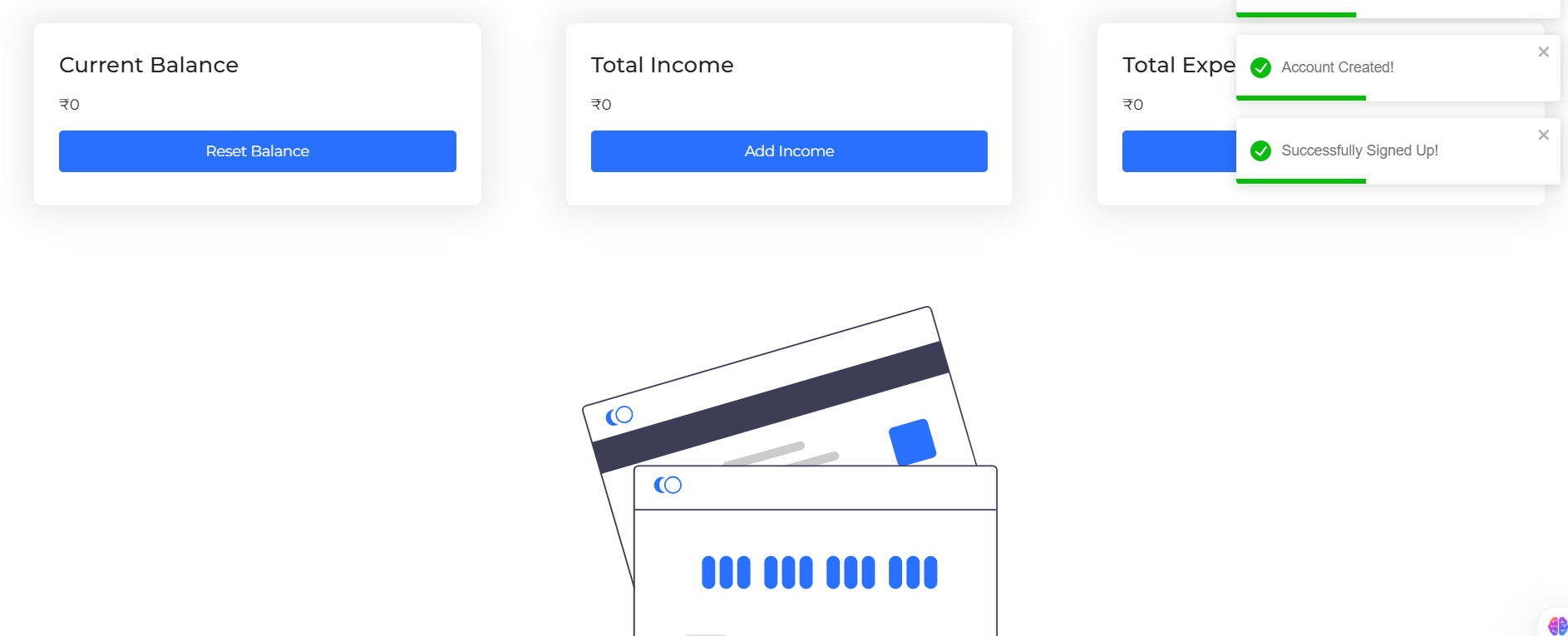
dotnet add package Firebase.Auth

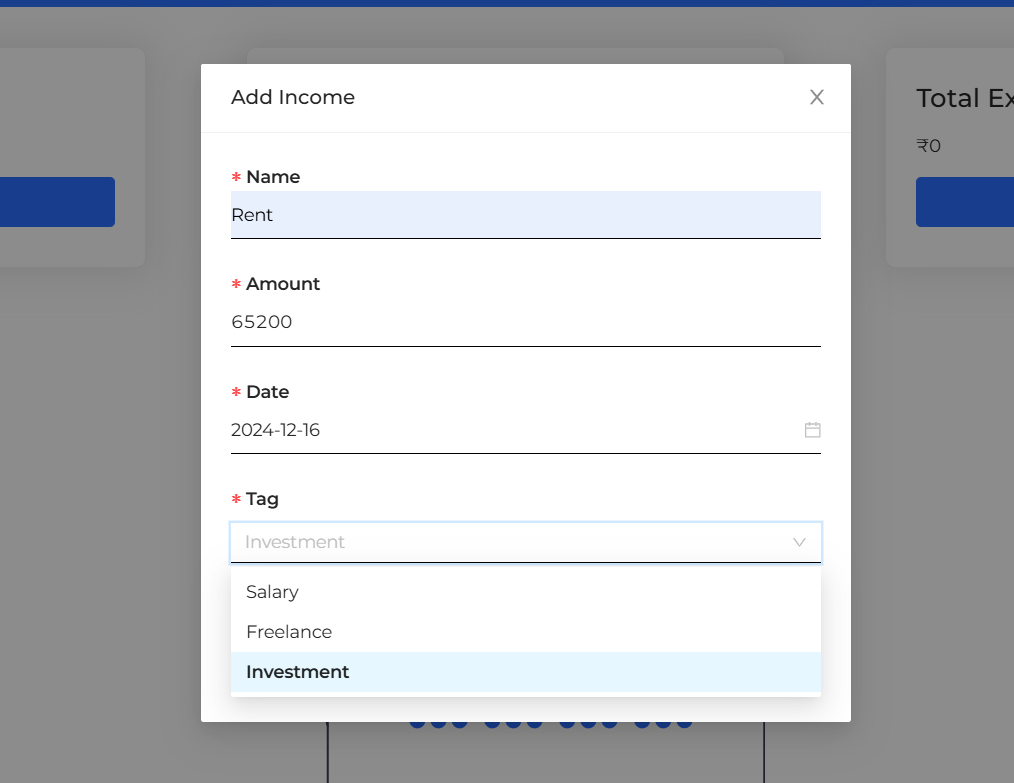
dotnet add package Firebase.Firestore

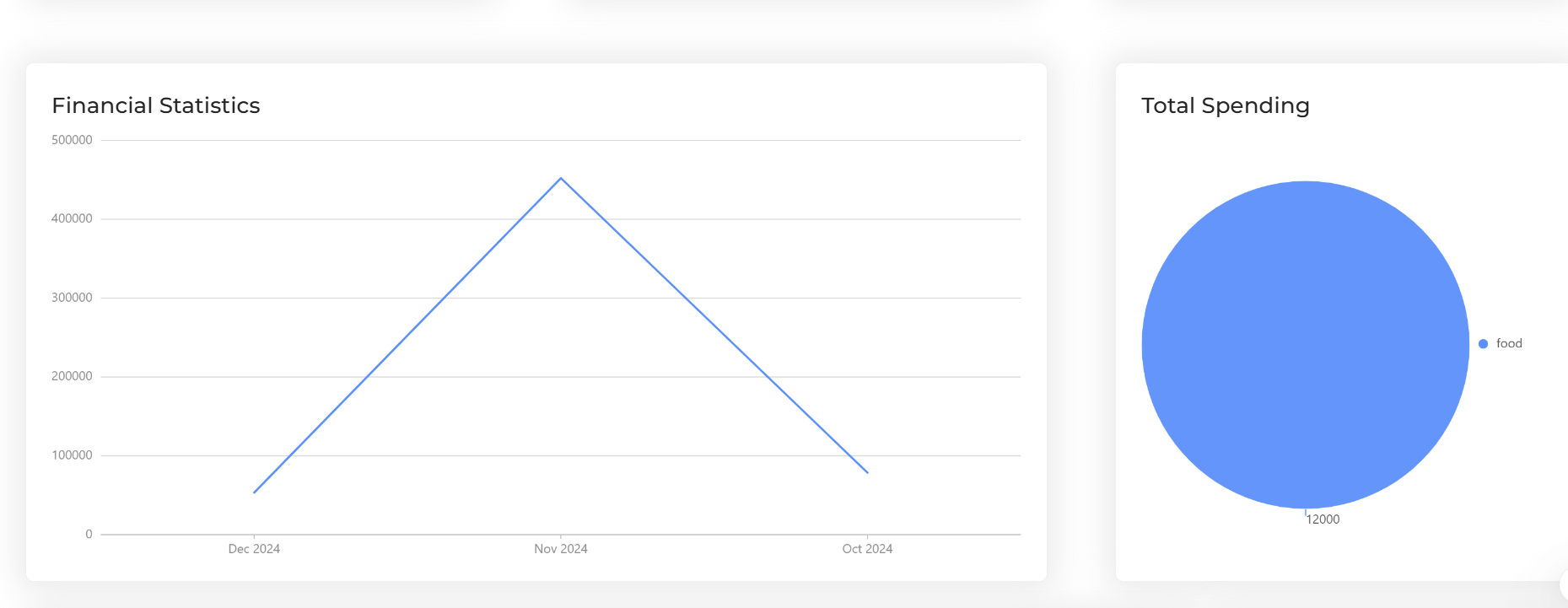
dotnet add package Newtonsoft.Json

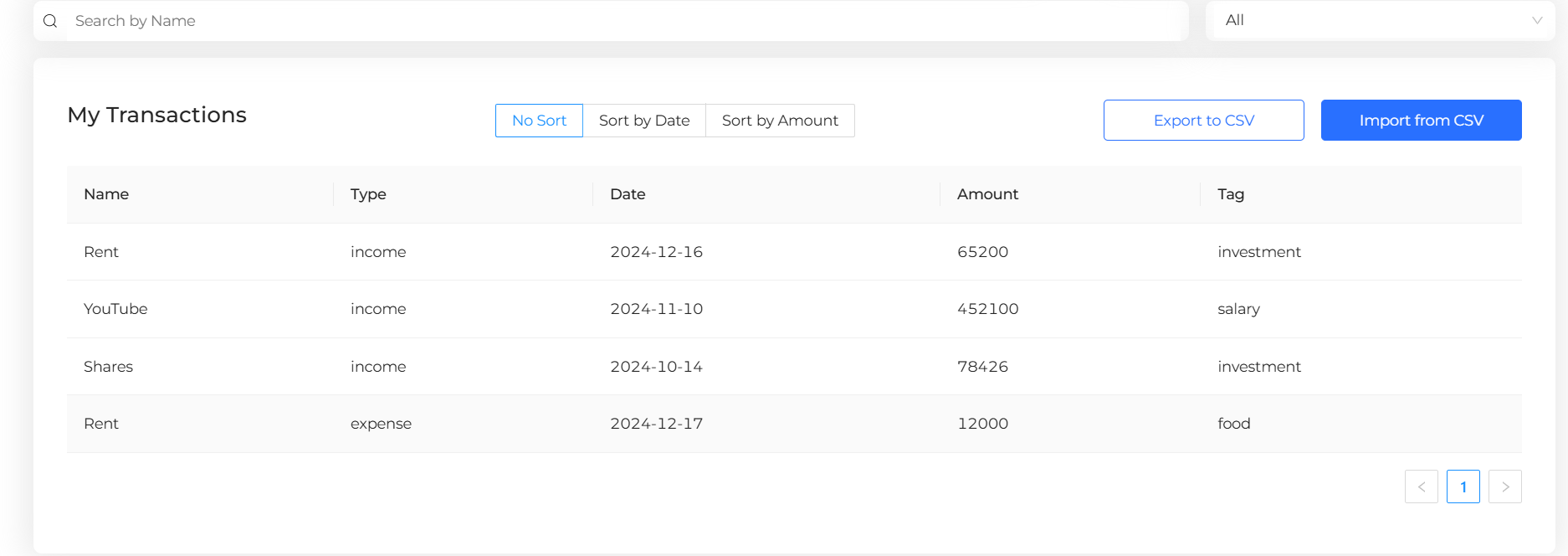
****

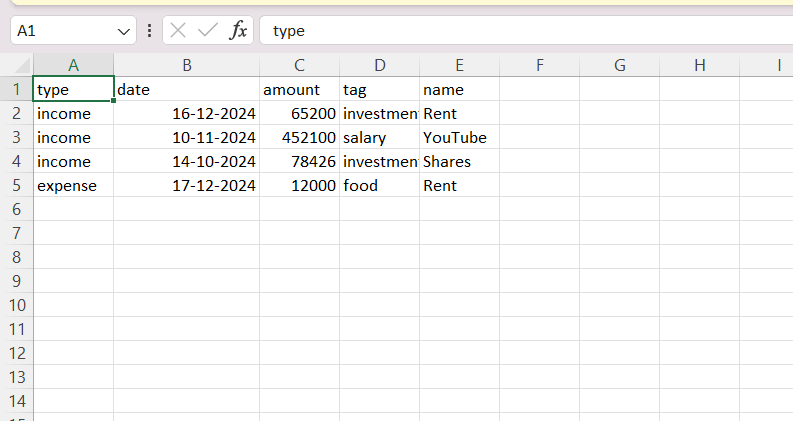
****

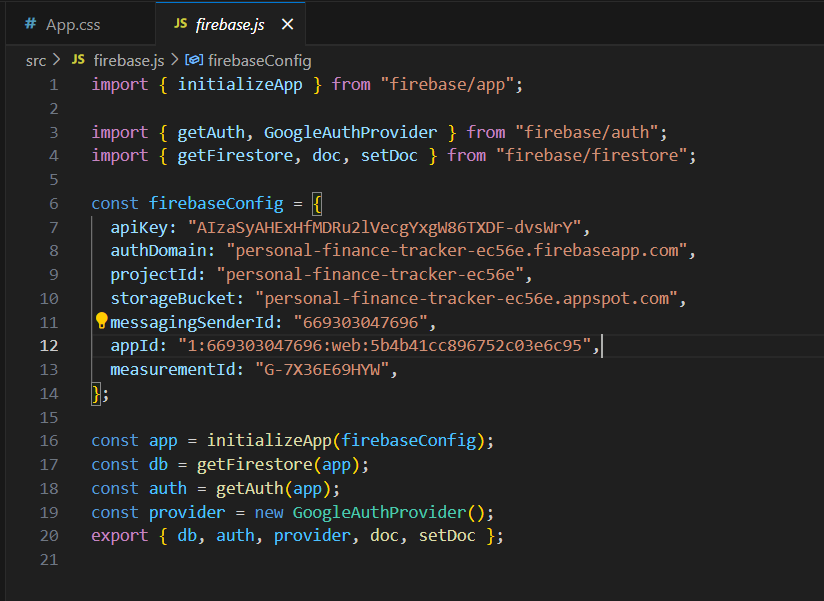
****

****

****

****

****

****

***FIREBASE DATA***

**Index.html**

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="utf-8" />*

*<link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />*

*<meta name="viewport" content="width=device-width, initial-scale=1" />*

*<meta name="theme-color" content="#000000" />*

*<meta*

*name="description"*

*content="Web site created using create-react-app"*

*/>*

*<link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />*

*<!--*

*manifest.json provides metadata used when your web app is installed on a*

*user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/*

*-->*

*<link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />*

*<!--*

*Notice the use of %PUBLIC\_URL% in the tags above.*

*It will be replaced with the URL of the `public` folder during the build.*

*Only files inside the `public` folder can be referenced from the HTML.*

*Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will*

*work correctly both with client-side routing and a non-root public URL.*

*Learn how to configure a non-root public URL by running `npm run build`.*

*-->*

*<title>React App</title>*

*</head>*

*<body>*

*<noscript>You need to enable JavaScript to run this app.</noscript>*

*<div id="root"></div>*

*<!--*

*This HTML file is a template.*

*If you open it directly in the browser, you will see an empty page.*

*You can add webfonts, meta tags, or analytics to this file.*

*The build step will place the bundled scripts into the <body> tag.*

*To begin the development, run `npm start` or `yarn start`.*

*To create a production bundle, use `npm run build` or `yarn build`.*

*-->*

*</body>*

*</html>*

**Index.js**

*import React, { useEffect } from "react";*

*import "./styles.css";*

*import { Link, useNavigate } from "react-router-dom";*

*import { useAuthState } from "react-firebase-hooks/auth";*

*import { auth } from "../../firebase";*

*import userSvg from "../../assets/user.svg";*

*function Header() {*

*const [user] = useAuthState(auth);*

*const navigate = useNavigate();*

*function logout() {*

*auth.signOut();*

*navigate("/");*

*}*

*useEffect(() => {*

*if (!user) {*

*navigate("/");*

*} else {*

*navigate("/dashboard");*

*}*

*}, [user, navigate]);*

*return (*

*<div className="navbar">*

*<p className="navbar-heading">Financly.</p>*

*{user ? (*

*<p className="navbar-link" onClick={logout}>*

*<span style={{ marginRight: "1rem" }}>*

*<img*

*src={user.photoURL ? user.photoURL : userSvg}*

*width={user.photoURL ? "32" : "24"}*

*style={{ borderRadius: "50%" }}*

*/>*

*</span>*

*Logout*

*</p>*

*) : (*

*<></>*

*)}*

*</div>*

*);*

*}*

*export default Header;*

**styles.css**

*.navbar {*

*position: sticky;*

*width: 100%;*

*padding: 0.75rem 1.5rem;*

*background-color: var(--theme);*

*display: flex;*

*justify-content: space-between;*

*align-items: center;*

*}*

*.navbar-heading {*

*color: #fff;*

*font-weight: 500;*

*margin: 0px;*

*font-size: 1.2rem;*

*}*

*.navbar-link {*

*color: #e3e3e3;*

*font-weight: 500;*

*margin: 0px;*

*font-size: 1rem;*

*cursor: pointer;*

*}*

*.navbar-link:hover {*

*color: #fff;*

*transition: all 0.3s;*

*}*

**TransactionSearch.js**

*import React, { useRef, useState } from "react";*

*import { Input, Table, Select, Radio } from "antd";*

*import { SearchOutlined } from "@ant-design/icons";*

*import search from "../assets/search.svg";*

*import { parse } from "papaparse";*

*import { toast } from "react-toastify";*

*import { useNavigate } from "react-router-dom";*

*const { Search } = Input;*

*const { Option } = Select;*

*const TransactionSearch = ({*

*transactions,*

*exportToCsv,*

*addTransaction,*

*fetchTransactions,*

*}) => {*

*const [searchTerm, setSearchTerm] = useState("");*

*const [selectedTag, setSelectedTag] = useState("");*

*const [typeFilter, setTypeFilter] = useState("");*

*const [sortKey, setSortKey] = useState("");*

*const fileInput = useRef();*

*function importFromCsv(event) {*

*event.preventDefault();*

*try {*

*parse(event.target.files[0], {*

*header: true,*

*complete: async function (results) {*

*// Now results.data is an array of objects representing your CSV rows*

*for (const transaction of results.data) {*

*// Write each transaction to Firebase, you can use the addTransaction function here*

*console.log("Transactions", transaction);*

*const newTransaction = {*

*...transaction,*

*amount: parseInt(transaction.amount),*

*};*

*await addTransaction(newTransaction, true);*

*}*

*},*

*});*

*toast.success("All Transactions Added");*

*fetchTransactions();*

*event.target.files = null;*

*} catch (e) {*

*toast.error(e.message);*

*}*

*}*

*const columns = [*

*{*

*title: "Name",*

*dataIndex: "name",*

*key: "name",*

*},*

*{*

*title: "Type",*

*dataIndex: "type",*

*key: "type",*

*},*

*{*

*title: "Date",*

*dataIndex: "date",*

*key: "date",*

*},*

*{*

*title: "Amount",*

*dataIndex: "amount",*

*key: "amount",*

*},*

*{*

*title: "Tag",*

*dataIndex: "tag",*

*key: "tag",*

*},*

*];*

*const filteredTransactions = transactions.filter((transaction) => {*

*const searchMatch = searchTerm*

*? transaction.name.toLowerCase().includes(searchTerm.toLowerCase())*

*: true;*

*const tagMatch = selectedTag ? transaction.tag === selectedTag : true;*

*const typeMatch = typeFilter ? transaction.type === typeFilter : true;*

*return searchMatch && tagMatch && typeMatch;*

*});*

*const sortedTransactions = [...filteredTransactions].sort((a, b) => {*

*if (sortKey === "date") {*

*return new Date(a.date) - new Date(b.date);*

*} else if (sortKey === "amount") {*

*return a.amount - b.amount;*

*} else {*

*return 0;*

*}*

*});*

*const dataSource = sortedTransactions.map((transaction, index) => ({*

*key: index,*

*...transaction,*

*}));*

*return (*

*<div*

*style={{*

*width: "100%",*

*padding: "0rem 2rem",*

*}}*

*>*

*<div*

*style={{*

*display: "flex",*

*justifyContent: "space-between",*

*gap: "1rem",*

*alignItems: "center",*

*marginBottom: "1rem",*

*}}*

*>*

*<div className="input-flex">*

*<img src={search} width="16" />*

*<input*

*placeholder="Search by Name"*

*onChange={(e) => setSearchTerm(e.target.value)}*

*/>*

*</div>*

*<Select*

*className="select-input"*

*onChange={(value) => setTypeFilter(value)}*

*value={typeFilter}*

*placeholder="Filter"*

*allowClear*

*>*

*<Option value="">All</Option>*

*<Option value="income">Income</Option>*

*<Option value="expense">Expense</Option>*

*</Select>*

*</div>*

*{/\* <Select*

*style={{ width: 200, marginRight: 10 }}*

*onChange={(value) => setSelectedTag(value)}*

*placeholder="Filter by tag"*

*allowClear*

*>*

*<Option value="food">Food</Option>*

*<Option value="education">Education</Option>*

*<Option value="office">Office</Option>*

*</Select> \*/}*

*<div className="my-table">*

*<div*

*style={{*

*display: "flex",*

*justifyContent: "space-between",*

*alignItems: "center",*

*width: "100%",*

*marginBottom: "1rem",*

*}}*

*>*

*<h2>My Transactions</h2>*

*<Radio.Group*

*className="input-radio"*

*onChange={(e) => setSortKey(e.target.value)}*

*value={sortKey}*

*>*

*<Radio.Button value="">No Sort</Radio.Button>*

*<Radio.Button value="date">Sort by Date</Radio.Button>*

*<Radio.Button value="amount">Sort by Amount</Radio.Button>*

*</Radio.Group>*

*<div*

*style={{*

*display: "flex",*

*justifyContent: "center",*

*gap: "1rem",*

*width: "400px",*

*}}*

*>*

*<button className="btn" onClick={exportToCsv}>*

*Export to CSV*

*</button>*

*<label for="file-csv" className="btn btn-blue">*

*Import from CSV*

*</label>*

*<input*

*onChange={importFromCsv}*

*id="file-csv"*

*type="file"*

*accept=".csv"*

*required*

*style={{ display: "none" }}*

*/>*

*</div>*

*</div>*

*<Table columns={columns} dataSource={dataSource} />*

*</div>*

*</div>*

*);*

*};*

*export default TransactionSearch;*

**firebase.js**

*import { initializeApp } from "firebase/app";*

*import { getAuth, GoogleAuthProvider } from "firebase/auth";*

*import { getFirestore, doc, setDoc } from "firebase/firestore";*

*const firebaseConfig = {*

*apiKey: "AIzaSyAHExHfMDRu2lVecgYxgW86TXDF-dvsWrY",*

*authDomain: "personal-finance-tracker-ec56e.firebaseapp.com",*

*projectId: "personal-finance-tracker-ec56e",*

*storageBucket: "personal-finance-tracker-ec56e.appspot.com",*

*messagingSenderId: "669303047696",*

*appId: "1:669303047696:web:5b4b41cc896752c03e6c95",*

*measurementId: "G-7X36E69HYW",*

*};*

*const app = initializeApp(firebaseConfig);*

*const db = getFirestore(app);*

*const auth = getAuth(app);*

*const provider = new GoogleAuthProvider();*

*export { db, auth, provider, doc, setDoc };*

**Signup.js**

*import React, { useState } from "react";*

*import { useNavigate } from "react-router-dom";*

*import { auth, provider, db } from "../firebase";*

*import {*

*createUserWithEmailAndPassword,*

*signInWithEmailAndPassword,*

*signInWithPopup,*

*} from "firebase/auth";*

*import { doc, getDoc, setDoc } from "firebase/firestore";*

*import Header from "./Header";*

*import { toast } from "react-toastify";*

*const SignUpSignIn = () => {*

*const [name, setName] = useState("");*

*const [email, setEmail] = useState("");*

*const [password, setPassword] = useState("");*

*const [confirmPassword, setConfirmPassword] = useState("");*

*const [loading, setLoading] = useState(false);*

*const [flag, setFlag] = useState(false);*

*const navigate = useNavigate();*

*const createUserDocument = async (user) => {*

*setLoading(true);*

*if (!user) return;*

*const userRef = doc(db, "users", user.uid);*

*const userData = await getDoc(userRef);*

*if (!userData.exists()) {*

*const { displayName, email, photoURL } = user;*

*const createdAt = new Date();*

*try {*

*await setDoc(userRef, {*

*name: displayName ? displayName : name,*

*email,*

*photoURL: photoURL ? photoURL : "",*

*createdAt,*

*});*

*toast.success("Account Created!");*

*setLoading(false);*

*} catch (error) {*

*toast.error(error.message);*

*console.error("Error creating user document: ", error);*

*setLoading(false);*

*}*

*}*

*};*

*const signUpWithEmail = async (e) => {*

*setLoading(true);*

*e.preventDefault();*

*try {*

*const result = await createUserWithEmailAndPassword(*

*auth,*

*email,*

*password*

*);*

*const user = result.user;*

*await createUserDocument(user);*

*toast.success("Successfully Signed Up!");*

*setLoading(false);*

*navigate("/dashboard");*

*} catch (error) {*

*toast.error(error.message);*

*console.error(*

*"Error signing up with email and password: ",*

*error.message*

*);*

*setLoading(false);*

*}*

*};*

*const signInWithEmail = async (e) => {*

*setLoading(true);*

*e.preventDefault();*

*try {*

*const result = await signInWithEmailAndPassword(auth, email, password);*

*const user = result.user;*

*navigate("/dashboard");*

*toast.success("Logged In Successfully!");*

*setLoading(false);*

*} catch (error) {*

*toast.error(error.message);*

*console.error(*

*"Error signing in with email and password: ",*

*error.message*

*);*

*setLoading(false);*

*}*

*};*

*const signInWithGoogle = async () => {*

*setLoading(true);*

*try {*

*const result = await signInWithPopup(auth, provider);*

*const user = result.user;*

*await createUserDocument(user);*

*toast.success("User Authenticated Successfully!");*

*setLoading(false);*

*navigate("/dashboard");*

*} catch (error) {*

*setLoading(false);*

*toast.error(error.message);*

*console.error("Error signing in with Google: ", error.message);*

*}*

*};*

*return (*

*<>*

*<Header />*

*<div className="wrapper">*

*{flag ? (*

*<div className="signup-signin-container">*

*<h2 style={{ textAlign: "center" }}>*

*Log In on <span className="blue-text">Financely.</span>*

*</h2>*

*<form onSubmit={signUpWithEmail}>*

*<div className="input-wrapper">*

*<p>Email</p>*

*<input*

*type="email"*

*placeholder="JohnDoe@gmail.com"*

*value={email}*

*onChange={(e) => setEmail(e.target.value)}*

*/>*

*</div>*

*<div className="input-wrapper">*

*<p>Password</p>*

*<input*

*type="password"*

*placeholder="Example123"*

*value={password}*

*onChange={(e) => setPassword(e.target.value)}*

*/>*

*</div>*

*<button*

*disabled={loading}*

*className="btn"*

*onClick={signInWithEmail}*

*>*

*{loading ? "Loading..." : " Log In with Email and Password"}*

*</button>*

*</form>*

*<p style={{ textAlign: "center", margin: 0 }}>or</p>*

*<button*

*disabled={loading}*

*className="btn btn-blue"*

*onClick={signInWithGoogle}*

*>*

*{loading ? "Loading..." : " Log In with Google"}*

*</button>*

*<p*

*onClick={() => setFlag(!flag)}*

*style={{*

*textAlign: "center",*

*marginBottom: 0,*

*marginTop: "0.5rem",*

*cursor: "pointer",*

*}}*

*>*

*Or Don't Have An Account? Click Here.*

*</p>*

*</div>*

*) : (*

*<div className="signup-signin-container">*

*<h2 style={{ textAlign: "center" }}>*

*Sign Up on <span className="blue-text">Financely.</span>*

*</h2>*

*<form onSubmit={signUpWithEmail}>*

*<div className="input-wrapper">*

*<p>Full Name</p>*

*<input*

*type="text"*

*placeholder="John Doe"*

*value={name}*

*onChange={(e) => setName(e.target.value)}*

*/>*

*</div>*

*<div className="input-wrapper">*

*<p>Email</p>*

*<input*

*type="email"*

*placeholder="JohnDoe@gmail.com"*

*value={email}*

*onChange={(e) => setEmail(e.target.value)}*

*/>*

*</div>*

*<div className="input-wrapper">*

*<p>Password</p>*

*<input*

*type="password"*

*placeholder="Example123"*

*value={password}*

*onChange={(e) => setPassword(e.target.value)}*

*/>*

*</div>*

*<div className="input-wrapper">*

*<p>Confirm Password</p>*

*<input*

*type="password"*

*placeholder="Example123"*

*value={confirmPassword}*

*onChange={(e) => setConfirmPassword(e.target.value)}*

*/>*

*</div>*

*<button type="submit" className="btn">*

*{loading ? "Loading..." : "Sign Up with Email and Password"}*

*</button>*

*</form>*

*<p style={{ textAlign: "center", margin: 0 }}>or</p>*

*<button*

*disabled={loading}*

*className="btn btn-blue"*

*onClick={signInWithGoogle}*

*>*

*{loading ? "Loading..." : "Sign Up with Google"}*

*</button>*

*<p*

*onClick={() => setFlag(!flag)}*

*style={{*

*textAlign: "center",*

*marginBottom: 0,*

*marginTop: "0.5rem",*

*cursor: "pointer",*

*}}*

*>*

*Or Have An Account Already? Click Here*

*</p>*

*{/\* <button onClick={signInWithEmail}>*

*Sign In with Email and Password*

*</button> \*/}*

*</div>*

*)}*

*</div>*

*</>*

*);*

*};*

*export default SignUpSignIn;*

**Conclusion:**

By implementing and integrating Firebase backend services, the application achieved secure and scalable data management, seamless authentication, and efficient real-time updates. Deployment through Firebase ensured a streamlined process with minimal downtime, enabling a reliable user experience.