İLERİ İNTERNET PROGRAMLAMA FİNAL PROJESİ

BUDGET MANAGER

Seray Özdemir 202013709021

metin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu**Register Sayfası**

Register sayfasında kullanıcı kaydı için bir form bulunmaktadır.

Kullanıcının adı, e-posta ve şifre bilgilerini girmesi beklenmektedir.

createUserWithEmailAndPassword fonksiyonu kullanılarak Firebase üzerinde yeni bir kullanıcı oluşturulmaktadır.

Kullanıcı oluşturulduktan sonra, kullanıcının profil bilgileri güncellenmekte ve ardından kullanıcı /dashboard sayfasına yönlendirilmektedir.

Hata durumunda uygun geri bildirim sağlanmaktadır.

metin, ekran görüntüsü, yazı tipi, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu**Login Sayfası**

Login.tsx dosyasında giriş için bir form bulunmaktadır.

Kullanıcının e-posta ve şifre bilgilerini girmesi beklenmektedir.

signInWithEmailAndPassword fonksiyonu kullanılarak Firebase üzerinde kullanıcı girişi yapılmaktadır.

Giriş başarılı olduğunda, kullanıcı /dashboard sayfasına yönlendirilmektedir.

Hata durumunda uygun geri bildirim sağlanmaktadır.

metin, ekran görüntüsü, dikdörtgen, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Dashboard Sayfası**

Dashboard sayfasında kullanıcının giriş yaptıktan sonra göreceği bir sayfa bulunmaktadır.

Sayfa içerisinde kullanıcı karşılama mesajı, gelir, gider, bakiye, işlem ve işlem geçmişi bulunmakta.

İşlem kısmından başlık, tutar ve gelir mi yoksa gider mi olup olmadığı seçilip kullanıcının giriş yaptığı değerlere göre gelir, gider ve bakiye değeri güncellenir.

Sayfaların ön yüzü style.scss dosyasında sass ile yazılmıştır.

**Balance.jsx**

**useEffect:** bileşenin her render işleminden sonra çalışan bir fonksiyonu çağırır. Bu sayede asenkron işlemleri gerçekleştirebiliyoruz.

**transactionsRef ve fetchTransactions:** Firebase Realtime Database kullanılarak transactions adlı bir referans oluşturuyor. Daha sonra onValue fonksiyonu kullanılarak bu referansa bir "değer değişikliği" dinleyicisi ekleniyor. Yani, transactionsRef üzerindeki veri her değiştiğinde, içerideki callback fonksiyonu çalıştırılıyor.

**Callback Fonksiyonu:** Bu fonksiyon, veritabanındaki işlemleri alır işlemleri bir diziye çevirir ve bu diziyi setTransactions fonksiyonu aracılığıyla bileşenin state'ine atar. Daha sonra, bu işlemleri filtreleyerek gelir ve gideri hesaplar. Gelir ve gideri de ayrı state'lerde saklanan setTotalIncome ve setTotalExpense fonksiyonları aracılığıyla bileşenin state'lerine atar.

**Expense.jsx**

**metin, elektronik donanım, ekran görüntüsü, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturulduuseState transactions:** Bu state, Firebase veritabanından çekilen işlemleri tutar. İlk değeri boş bir dizi olan useState hook'u ile oluşturulur.

**totalExpense:** Bu state, toplam gider miktarını tutar. İlk değeri 0 olan useState ile oluşturulur.

**useEffect:** useEffect, bileşen yüklendiğinde çalışır. Bu hook içinde, Firebase Realtime Database'den transactions adlı veritabanı tablosundan veri çekme işlemi gerçekleştirilir.

ref fonksiyonu ile transactions tablosuna referans oluşturulur. onValue fonksiyonu kullanılarak bu referansa bir "değer değişikliği" dinleyicisi eklenir. Veri değiştiğinde, snapshot üzerinden veriyi çeker ve ilgili state'leri günceller. Bu işlemler sadece bir kere çalışsın diye dependency array olarak boş bir dizi ([]) kullanılır.

**Return ve Cleanup:** return bloğu içinde, temizlik işlemleri yapılır. Bu durumda, fetchTransactions adlı fonksiyon ile veri çekme işlemi durdurulur. Bellek sızıntısını önlemek için kullanılır.

**History.jsx**

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**useState transactions:** Bu state, Firebase veritabanından çekilen işlemleri tutar. İlk değeri boş bir dizi olan useState hook'u ile oluşturulur.

**useEffect:** bileşen yüklendiğinde çalışır. Bu hook içinde, fetchData fonksiyonu çağrılarak Firebase Realtime Database'den transactions adlı veritabanı tablosundan veri çekme işlemi gerçekleştirilir.Dependency array olarak boş bir dizi kullanıldığı için, bu işlem sadece bir kez gerçekleşir.

**fetchData Fonksiyonu:** fetchData fonksiyonu, ref fonksiyonu ile transactions tablosuna referans oluşturur. onValue fonksiyonu kullanılarak bu referansa bir "değer değişikliği" dinleyicisi eklenir. Veri değiştiğinde, snapshot üzerinden veriyi çeker ve ilgili state'i günceller.

**handleDeleteButtonClick Fonksiyonu:** bir işlemin silinmesi durumunda çağrılır. ref fonksiyonu ile silinecek işlemin referansı oluşturulur. remove fonksiyonu kullanılarak Firebase'den veri silinir. Silme işlemi başarılı olduğunda, işlem state'ten kaldırılır ve onTransactionDeleted callback fonksiyonu çağrılarak silinen işlemi bildirir. Hata durumunda, hata konsola yazılır.

**metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturulduIncome.jsx**

**useEffect :** bileşen yüklendiğinde çalışır. Dependency array olarak boş bir dizi ([]) kullanıldığı için, bu işlem sadece bir kez gerçekleşir.

**Firebase Veritabanı Referansı Oluşturma:** ref fonksiyonu kullanılarak Firebase veritabanındaki transactions referansı oluşturur.

**Veri Çekme ve İşlemler:** Dinleyici fonksiyonu içinde, snapshot üzerinden veriyi çeker .Veri varsa, bu veriyi bir diziye çevirir ve setTransactions fonksiyonu aracılığıyla bileşenin transactions state'ine kaydeder.

**Toplam Geliri Hesaplama:** transactionList dizisi üzerinde filter ve reduce fonksiyonları kullanılarak gelir işlemlerinden toplam gelir hesaplanır. Hesaplanan gelir, setTotalIncome fonksiyonu aracılığıyla bileşenin totalIncome state'ine kaydedilir.

**metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturulduTransaction.jsx**

**useState:**

title: İşlem başlığını (title) tutan state.

amount: İşlem miktarını (amount) tutan state.

type: İşlem türünü (type) tutan state. Başlangıçta "income" olarak ayarlanmıştır.

**handleAddTransaction Fonksiyonu:** Kullanıcının eklediği yeni bir işlemi Firebase Realtime Database'e kaydetmek için kullanılır. ref fonksiyonu kullanılarak transactions adlı veritabanı tablosuna referans oluşturulur. push fonksiyonu ile yeni işlem objesi veritabanına eklenir. Eklenen işlemin referansı alınır. Eklenen işlemin referansından anahtar (key) alınır ve bu anahtar, yeni işlemin bir benzersiz kimliği olarak kullanılır. Yeni işlemin bilgileri ve anahtarı bir nesne olarak onTransactionAdded callback fonksiyonuna iletilir.

**Loglama ve Hata Kontrolü:**Eklenen işlemin anahtarı alındıysa, bu anahtar ve işlem bilgileri konsola yazılır. Anahtar alınamazsa, bir hata mesajı konsola yazılır.

Bu sayfa kullanıcının eklediği yeni bir işlemi Firebase Realtime Database'e kaydetmek için kullanılır. Yeni işlem bilgileri ve işlemin bir kimliği olan anahtar, onTransactionAdded callback fonksiyonuna iletilir. Bu sayede işlem eklenirken aynı zamanda kullanıcı arayüzü üzerinde de güncelleme yapılır.

**Dashboard.tsx**

metin, ekran görüntüsü, yazılım, işletim sistemi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**useState:**

totalIncome: Toplam geliri tutan state.

totalExpense: Toplam gideri tutan state.

totalBalance: Toplam bakiyeyi tutan state.

transactions: İşlem geçmişini tutan state.

**handleTransactionAdded Fonksiyonu:** Transaction bileşeninden gelen yeni bir işlemi işler. Eğer eklenen işlem bir gelirse, totalIncome state'ini günceller. Eğer eklenen işlem bir gider ise, totalExpense state'ini ve totalIncome state'ini günceller. totalBalance state'ini günceller.

**handleTransactionDeleted Fonksiyonu:** History bileşeninden gelen bir işlemin silinmesini işler. Eğer silinen işlem bir gelirse, totalIncome state'ini günceller. Eğer silinen işlem bir gider ise, totalExpense state'ini ve totalIncome state'ini günceller. totalBalance state'ini günceller. Silinen işlem, transactions state'inden filtrelenir.

**Render:** Bileşenin render fonksiyonu içinde, kullanıcı arayüzü oluşturulur. Welcome bileşeni, kullanıcıya hoş geldin mesajını içeren bir bileşendir. Balance, Income, ve Expense bileşenleri, kullanıcının toplam bakiye, toplam gelir ve toplam gider bilgilerini gösterir. Transaction bileşeni, kullanıcının yeni işlemler eklemesine olanak tanır ve eklenen işlemleri işlemek için handleTransactionAdded callback fonksiyonunu kullanır. History bileşeni, kullanıcının işlem geçmişini görüntülemesine olanak tanır ve silinen işlemleri işlemek için handleTransactionDeleted callback fonksiyonunu kullanır.

**metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturulduLogin.tsx**

**useState:**

email: Kullanıcının giriş yaparken kullanacağı e-posta adresini tutan state.

password: Kullanıcının giriş yaparken kullanacağı şifreyi tutan state.

err: Giriş işlemi sırasında bir hata olup olmadığını belirten boolean bir state.

errorMessage: Hata durumunda kullanıcıya gösterilecek hata mesajını tutan state.

**useNavigate :** sayfa geçişleri ve yönlendirme işlemlerini gerçekleştirmek için kullanılır.

**handleLogin Fonksiyonu:** formun gönderilmesini (submit) ele alır ve kullanıcının giriş yapma işlemlerini gerçekleştirir.

**e.preventDefault():** Boş form gönderimini engellemek için kullanılır.

**signInWithEmailAndPassword:** Firebase Authentication servisinin signInWithEmailAndPassword fonksiyonu ile kullanıcının e-posta ve şifresi ile giriş yapma işlemi gerçekleştirilir. Eğer giriş başarılı ise, navigate fonksiyonu kullanılarak kullanıcı "/dashboard" sayfasına yönlendirilir. Eğer bir hata oluşursa, err state'i true olarak ayarlanır ve hata mesajı (errorMessage) ilgili hata durumuna göre ayarlanır. Hata durumları console'a yazdırılır.

**metin, elektronik donanım, ekran görüntüsü, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturulduRegister.tsx**

**useState:**

err: Kayıt işlemi sırasında bir hata olup olmadığını belirten boolean bir state.

loading: Kayıt işlemi sırasında yüklenme durumunu belirten boolean bir state.

errorMessage: Hata durumunda kullanıcıya gösterilecek hata mesajını tutan state.

**useNavigate :** useNavigate, sayfa geçişleri ve yönlendirme işlemlerini gerçekleştirmek için kullanılır.

**handleSubmit Fonksiyonu:** kayıt formunun gönderilmesini ele alır ve kullanıcının yeni bir hesap oluşturmasını gerçekleştirir.

**e.preventDefault():** Boş form gönderimini engellemek için kullanılır.

Kullanıcının girdiği ad, e-posta ve şifre değerlerini alır. **createUserWithEmailAndPassword:** Firebase Authentication servisinin createUserWithEmailAndPassword fonksiyonu ile kullanıcının e-posta ve şifresi ile hesap oluşturma işlemi gerçekleştirilir.

**updateProfile:** Oluşturulan kullanıcı profiline ad bilgisini eklemek için kullanılır. Oluşturulan kullanıcı ve güncellenen profil bilgileri konsola yazdırılır. Kullanıcı başarıyla oluşturulduktan sonra, navigate fonksiyonu kullanılarak kullanıcı "/dashboard" sayfasına yönlendirilir. Eğer bir hata oluşursa, err state'i true olarak ayarlanır ve hata mesajı (errorMessage) ilgili hata durumuna göre ayarlanır. Hata durumu console'a yazdırılır. Loading state'i, kayıt işlemi tamamlandıktan sonra false olarak ayarlanır.

**Render:** Bileşenin render fonksiyonu içinde, kullanıcı arayüzü oluşturulur. Kullanıcıdan ad, e-posta ve şifre bilgilerini girmesi istenen bir form bulunur. Kayıt butonu kullanıcının kayıt işlemini başlatır. Bu buton, loading state'ine bağlı olarak etkin veya etkisiz hale getirilir. Eğer bir hata oluşursa, hata mesajı kullanıcıya gösterilir. Zaten bir hesabı varsa, kullanıcıyı giriş yapma sayfasına yönlendiren bir bağlantı bulunur.