

SAKARYA ÜNİVERSİTESİ
BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

Ders Adı: Mobil Uygulama Geliştirme

Ders Sorumlusu: Öğr.Gör. AHMET ŞANSLI

Uygulama Adı: Stock Market App

Hazırlayan: Ömer Tufan Ayhan – B191210012

Amaç

Stock Market App, kullanıcıların kripto varlıklarını takip etmelerini, portföy oluşturmalarını ve bu varlıkların fiyatlarını izlemelerini sağlayan bir mobil uygulamadır. Temel amaçları şunlardır:

- Kullanıcı dostu bir arayüz üzerinden geniş kapsamlı kripto varlık listesi sunmak.
- Kullanıcıların kişisel portföylerini oluşturmalarına ve yönetmelerine olanak sağlamak.
- Güvenli kimlik doğrulama ve yönetici-kullanıcı rolleriyle farklı yetkilendirmeler sunmak.

Teknolojiler

Stock Market App, React Native kullanılarak geliştirilmiştir. Firebase gibi bulut tabanlı hizmetler kimlik doğrulama ve veritabanı yönetimi için kullanılmıştır. React Native, uygulamanın hem iOS hem de Android platformlarında sorunsuz çalışmasını sağlamıştır.

Öneriler ve İyileştirmeler

Uygulamanın daha da geliştirilmesi için öneriler şunlardır:

- Daha fazla kripto varlık verisini entegre etmek.
- Kullanıcıların portföylerini daha detaylı analiz edebilecekleri grafiksel gösterimler eklemek.

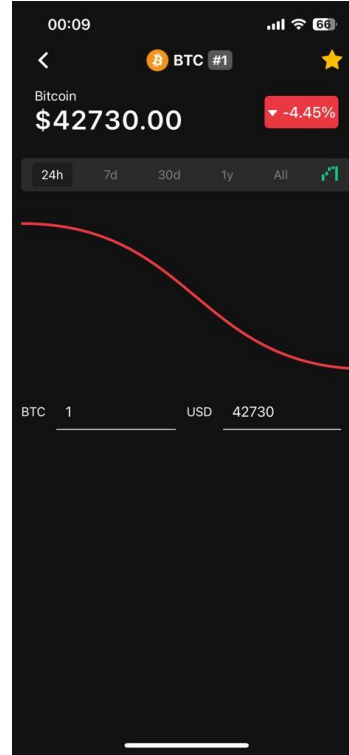
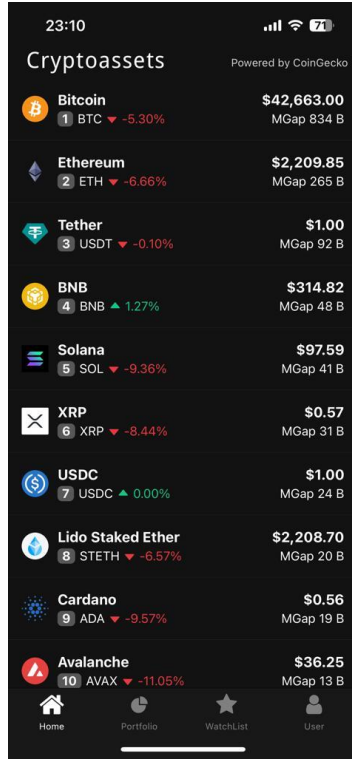
Sonuç

Stock Market App, kullanıcı dostu arayüzü ve çeşitli özellikleriyle kripto varlık takibi konusunda kullanıcılara önemli bir kolaylık sunmaktadır. Geliştirme sürecinde elde edilen geri bildirimler doğrultusunda yapılacak iyileştirmelerle uygulamanın kullanıcı tabanını genişletmek ve daha kapsamlı hale getirmek hedeflenmektedir.

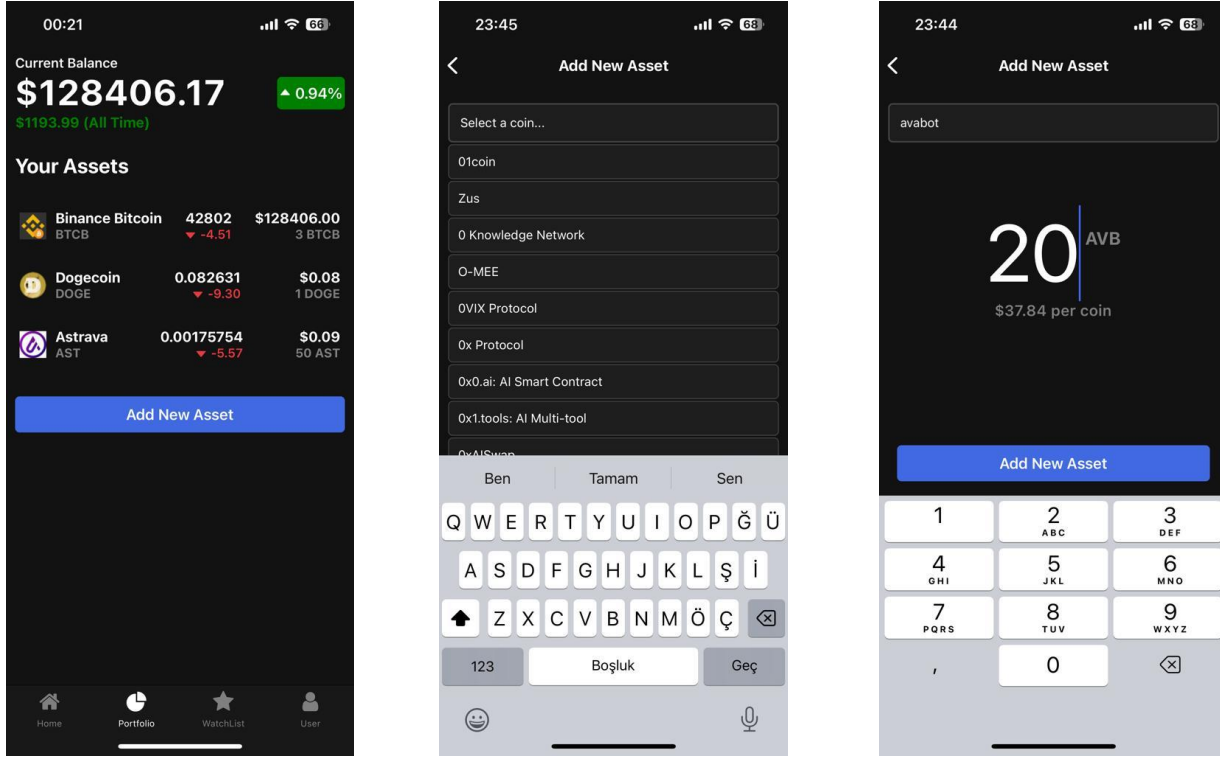
- [***Geliştirme Sırasında Alınan Hata için Açılan Stackoverflow açılan başlık***](#)

Uygulama Özellikleri

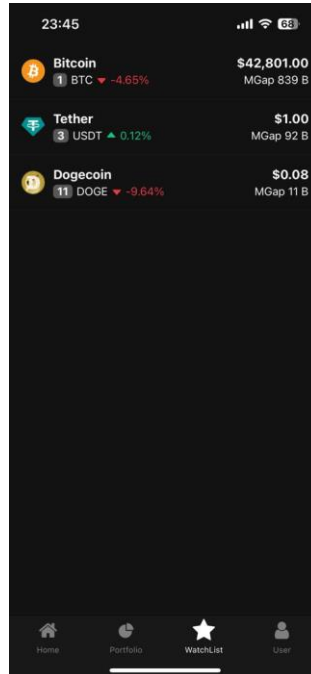
- Kripto Varlık Listesi:** Uygulama, farklı kripto varlıklarının anlık fiyatlarını ve değişimlerini kullanıcılara grafikler ile çeşitli tarih seçeneklerinde sunar.



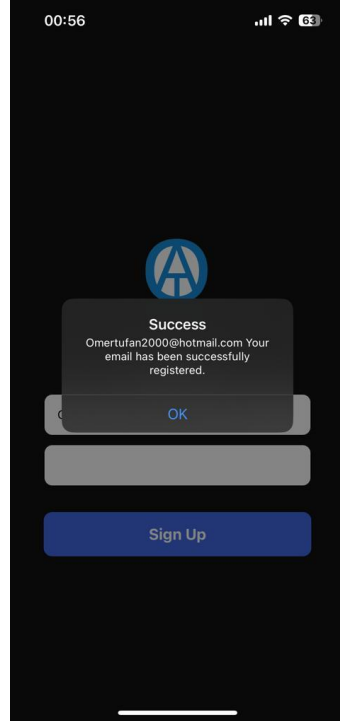
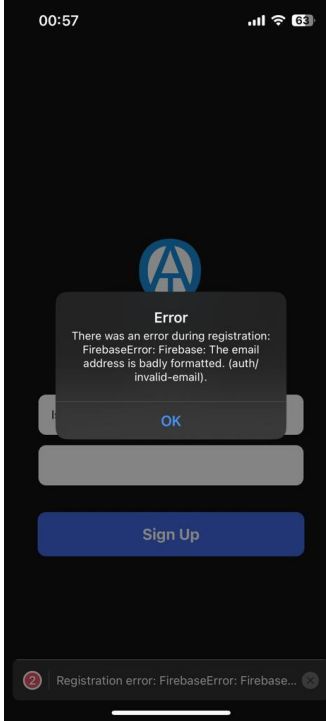
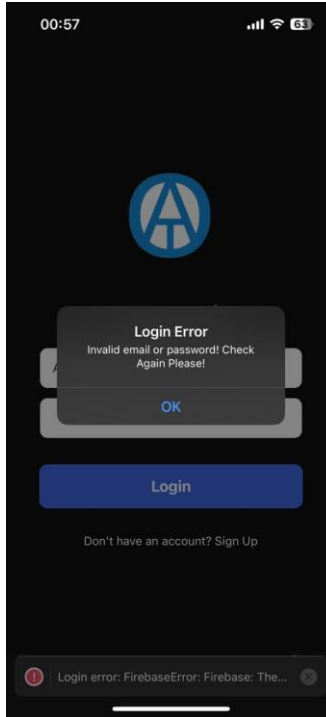
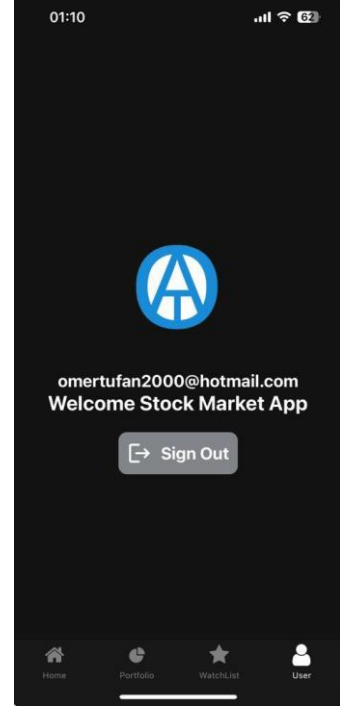
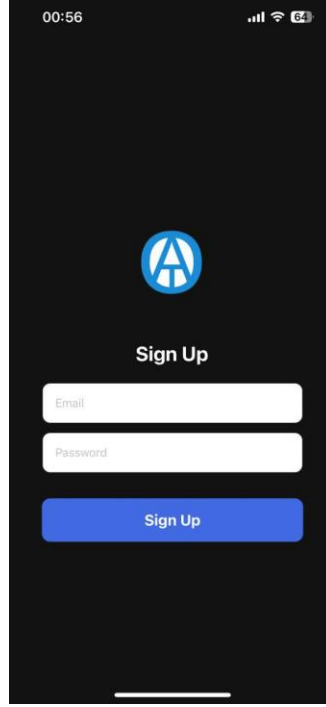
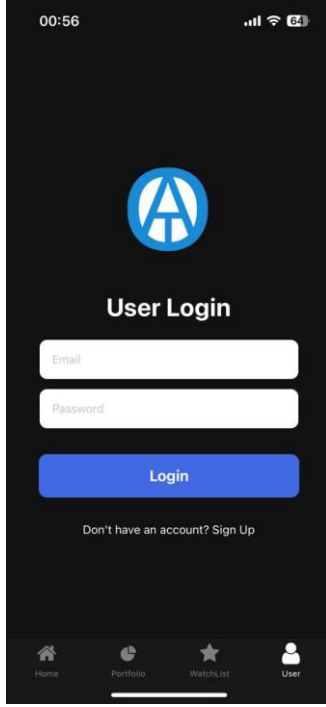
2. Portföy Oluşturma ve İzleme: Kullanıcılar, uygulama üzerinden kendi portföylerini oluşturabilir ve bu varlıkların performansını takip edebilir.



3. İzleme Listeleri: Kullanıcılar favori varlıklarını izleme listelerine ekleyerek daha kolay takip edebilirler.



4. Kimlik Doğrulama ve Yetkilendirme: Uygulama, güvenli giriş sağlamak ve farklı kullanıcı rolleri arasında ayrım yapmak için kimlik doğrulama özellikleri sunar.



Teknik Dokümantasyon:

Firestore Config File: (<https://firebase.google.com/docs/auth>) kendi dokümantasyonunun takip edilmesi önem teşkil etmektedir.

```
JS firebase.js x
src > services > JS firebase.js > ...
1  import firebase from "firebase/compat/app";
2  import "firebase/compat/auth";
3  import "firebase/compat/firestore";
4
5  // Your web app's Firebase configuration
6  const firebaseConfig = {
7    apiKey: "AIzaSyAPU2EcSwfF0uLssHJUMIA9rCiTEA10iKo",
8    authDomain: "stock-market-app-auth.firebaseio.com",
9    projectId: "stock-market-app-auth",
10   storageBucket: "stock-market-app-auth.appspot.com",
11   messagingSenderId: "469569718811",
12   appId: "1:469569718811:web:b2481385af93be06a9c6be"
13 };
14
15 // Initialize Firebase
16 let app;
17 if (firebase.apps.length === 0) {
18   app = firebase.initializeApp(firebaseConfig);
19 } else {
20   app = firebase.app();
21 }
22
23 const auth = firebase.auth();
24
25 export { auth };
26
```

Flatlist Kullanım:

Kullanıcı her 50+1 veriye geldiğinde kaynakların devamı yüklenmektedir.

```
JS index.jsx x
src > screens > HomeScreen > JS index.jsx > [0] HomeScreen
60  </View>
61  <FlatList
62    data={coins}
63    renderItem={({ item }) => <CoinItem marketCoin={item} />}
64    onEndReached={() => fetchCoins(coins.length / 50 + 1)}
65    refreshControl={
66      <RefreshControl
67        refreshing={loading}
68        tintColor="white"
69        onRefresh={refetchCoins}
70      />
71    }
72  />
73 </View>
74 ];
75 };
76
77 export default HomeScreen;
78
```

Authentication Kullanım:

```
const UserScreen = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [userEmail, setUserEmail] = useState(""); // Kullanıcının e-posta adresi
  const navigation = useNavigation();
  const [initializing, setInitializing] = useState(true);

  function onAuthStateChanged(user) {
    if (user) {
      setUserEmail(user.email); // Şuanki kullanıcının e-posta adresini al
    }
    if (!initializing) setInitializing(false);
  }

  useEffect(() => {
    const subscriber = auth.onAuthStateChanged(onAuthStateChanged);
    return subscriber;
  }, []);

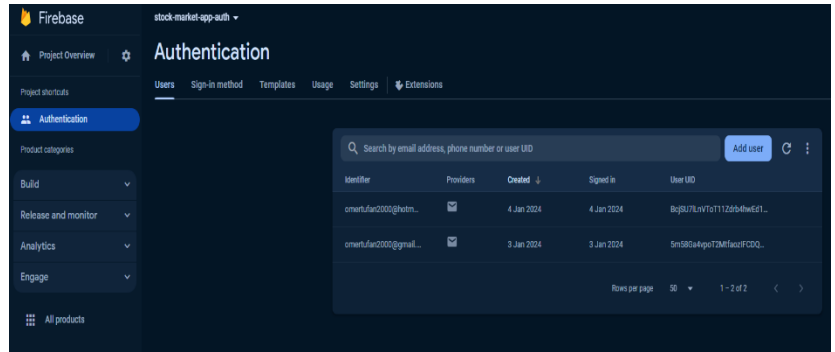
  const handleLogout = async () => {
    try {
      await auth.signOut();
      setUserEmail(""); // Kullanıcı oturumunu kapatırken kullanıcısı bilgisini temizle
      await AsyncStorage.removeItem("userEmail"); // AsyncStorage'den kullanıcısı bilgisini kaldır
    } catch (error) {
      console.error("Logout error:", error);
    }
  };

  const handleLogin = async () => {
    try {
      const userCredential = await auth.signInWithEmailAndPassword(
        email,
        password
      );
      const user = userCredential.user;
      console.log("Successful login:", user);
      setUserEmail(user.email); // Şuanki kullanıcının e-posta adresini al
      await AsyncStorage.setItem("userEmail", user.email);
    } catch (error) {
      console.error("Login error:", error);
      Alert.alert(
        "Login Error",
        "Invalid email or password! Check Again Please!"
      );
    }
  };

  const handleSignUpNavigation = () => {
    navigation.navigate("SignUpScreen");
  };

  if (!initializing) return null;
};
```

src/LoginScreen/index.jsx



The screenshot shows the Firebase Authentication console for a project named 'stock-market-app-auth'. The 'Users' tab is selected, displaying a table of users. The table has columns for Identifier, Providers, Created, Signed in, and User UID. Two users are listed: one created on 4 Jan 2024 and another on 3 Jan 2024. The interface includes a search bar, an 'Add user' button, and a sidebar with navigation options like Project Overview, Authentication, Build, Release and monitor, Analytics, Engage, and All products.

Identifier	Providers	Created	Signed in	User UID
omwrfar2000@ghetm...	📧	4 Jan 2024	4 Jan 2024	8c9d77b1v7ts1126b9dwe6t1...
omwrfar2000@gmail...	📧	3 Jan 2024	3 Jan 2024	5m585a6pet7AMfsczFCDD...

Firestore Authentication ait görüntü

Recoil Kullanım:

Kripto para portföyü takibi için, kullanıcıların yerel depoda (localStorage) saklanan kripto para birimi kimlikleri ile API'den güncel verilerin alınmasıyla Portfolio ekranında takip edilebiliyor.

```
JS PortfolioAssets.js X
src > atoms > JS PortfolioAssets.js > [0] allPortfolioBoughtAssetsFromAPI > [0] get > [0] boughtAssets > [0] boughtPortfolioAssets.map() callback
1 import { atom, selector } from 'recoil';
2 import AsyncStorage from '@react-native-async-storage/async-storage';
3 import { getWatchlistedCoins } from '../services/requests';
4
5 export const allPortfolioBoughtAssets = selector({
6   key: 'allPortfolioBoughtAssets',
7   get: async () => {
8     const jsonValue = await AsyncStorage.getItem('@portfolio_coins')
9     return jsonValue != null ? JSON.parse(jsonValue) : []
10  }
11 })
12
13 export const allPortfolioBoughtAssetsFromAPI = selector({
14   key: 'allPortfolioBoughtAssetsFromAPI',
15   get: async ({get}) => {
16     const boughtPortfolioAssets = get(allPortfolioBoughtAssetsInStorage)
17     const portfolioAssetsMarketData = await getWatchlistedCoins(1, boughtPortfolioAssets.map((portfolioAsset) => portfolioAsset.id).join(','))
18
19     const boughtAssets = boughtPortfolioAssets.map((boughtAsset) => {
20       const portfolioAsset = portfolioAssetsMarketData.filter((item) => boughtAsset.id === item.id)[0]
21       return {
22         ...boughtAsset,
23         currentPrice: portfolioAsset.current_price,
24         priceChangePercentage: portfolioAsset.price_change_percentage_24h
25       }
26     })
27
28     return boughtAssets.sort((item1, item2) => (item1.quantityBought * item1.currentPrice) < (item2.quantityBought * item2.currentPrice))
29   }
30 })
31
32 export const allPortfolioAssets = atom({
33   key: 'allPortfolioAssets',
34   default: allPortfolioBoughtAssetsFromAPI
35 })
36
37 export const allPortfolioBoughtAssetsInStorage = atom({
38   key: 'allPortfolioBoughtAssetsInStorage',
39   default: allPortfolioBoughtAssets,
40 })
```