|  |  |  |
| --- | --- | --- |
|  | **T.C.**  **KASTAMONU ÜNİVERSİTESİ**  **MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ**  **BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ** |  |

**BİTİRME PROJESİ I**

**KONU**

Online Açık Arttırma Sistemi

**HAZIRLAYAN**

214410015 –Halil Öksüz

**DANIŞMAN**

Doç. Dr. Ekmel Çetin

Haziran - 2024

**KASTAMONU**

**T.C.**

**KASTAMONU ÜNİVERSİTESİ**

**MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**BİTİRME PROJESİ I**

**KONU**

Online Açık Arttırma Sistemi

**HAZIRLAYAN**

214410015 – Halil Öksüz

**DANIŞMAN**

Doç. Dr. Ekmel Çetin

Haziran - 2024

**KASTAMONU**

**…/…/ 2020**

**ETİK BEYAN**

Kastamonu Üniversitesi, Mühendislik ve Mimarlık Fakültesi, Bilgisayar Mühendisliği Bölümü, Mühendislik Tamamlama Programı, Tez Hazırlama Kılavuzu’nda yer alan kurallara uygun olarak hazırladığım bu çalışmada; proje içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi, tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu, proje çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi, kullanılan verilerde herhangi bir değişiklik yapmadığımı, bu projede sunduğum çalışmanın özgün olduğunu, bildirir; aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

**Öğrenci Numarası :** 214410015 **İmza**

**Adı Soyadı :** Halil Öksüz

**ÖZET**

**Bitime Projesi**

**Online Açık Arttırma Sistemi**

**Halil Öksüz**

**Kastamonu Üniversitesi**

**Bilgisayar Mühendisliği Bölümü**

**Bitirme Projesi Danışmanı:**

**[Dr. Öğr. Üyesi Murat MERİÇELLİ](https://unis.kastamonu.edu.tr/akademisyen/mmericelli)**

**Haziran 2024, 35 sayfa**

Teknolojinin gelişmesiyle birlikte insan yaşamında önemli bir yer edinen yapay zeka ve derin öğrenme gibi alanlar, pek çok alanda yenilikçi çözümler sunmaktadır. Bu yeniliklerden biri de online açık arttırma sistemleridir. Geleneksel açık arttırmaların dijital platformlara taşınmasıyla birlikte, bu sistemlerin kullanımı gün geçtikçe artmaktadır. Online açık arttırma sistemleri, hem alıcılar hem de satıcılar için geniş bir erişim sağlayarak, ticaretin daha şeffaf ve erişilebilir olmasını mümkün kılmaktadır.

Günümüzde online açık arttırma sistemlerinin gelişimi, pek çok teknolojik yenilikle desteklenmektedir. Bu yüzden temel bir online açık arttırma sistemi mobil uygulaması yapmak istedim.

**ABSTRACT**

**Senior Project**

**EMOTION RECOGNITION**

**Halil Öksüz**

**Kastamonu University**

**Faculty of Engineering and Architecture**

**Department of Computer Engineering**

**Project Advisor:**

**Assistant Professor Murat MERİÇELLİ, PhD.**

**June 2024, 35 pages**

With the advancement of technology, fields such as artificial intelligence and deep learning have gained significant importance in human life, offering innovative solutions across various domains. One of these innovations includes online auction systems. As traditional auctions move to digital platforms, the usage of these systems is steadily increasing. Online auction systems provide extensive access for both buyers and sellers, making trade more transparent and accessible.

The development of online auction systems today is supported by numerous technological advancements. Therefore, I wanted to create a basic mobile application for an online auction system.

**İÇİNDEKİLER SAYFA**

[**BİTİRME PROJESİ I** i](#_Toc170209886)

[**HAZIRLAYAN** i](#_Toc170209887)

[**DANIŞMAN** i](#_Toc170209888)

[**BİTİRME PROJESİ I** ii](#_Toc170209889)

[**HAZIRLAYAN** ii](#_Toc170209890)

[**DANIŞMAN** ii](#_Toc170209891)

[**ÖZET** iv](#_Toc170209892)

[**ABSTRACT** v](#_Toc170209893)

[**Halil Öksüz** v](#_Toc170209894)

[**Kastamonu University** v](#_Toc170209895)

[**Project Advisor:** v](#_Toc170209896)

[1. GİRİŞ ve TANITIM 7](#_Toc170209897)

[1.1. Sitemin Çalışma Mantığı 8](#_Toc170209898)

[2.1 Authentication Aşaması 9](#_Toc170209899)

[2.2 Profile aşaması 26](#_Toc170209900)

[2.3 Dashboard aşaması 33](#_Toc170209901)

[2.3 My auctions aşaması 38](#_Toc170209902)

[2.4 Won auctions aşaması 40](#_Toc170209903)

[2.4 Create post aşaması 42](#_Toc170209904)

[2.5 Auction aşaması 46](#_Toc170209905)

[PROJE ÇALIŞMASINDA KULLANILAN MATERYALLER 53](#_Toc170209906)

[3.1. Firebase 53](#_Toc170209907)

[3.2 Visual Studio Code 53](#_Toc170209908)

[3.6. Uygulamnın sayfalarına ait görseller 55](#_Toc170209909)

[4. SONUÇLAR VE ÖNERİLER 58](#_Toc170209910)

[**KAYNAKLAR** 63](#_Toc170209911)

[**ÖZGEÇMİŞ** 64](#_Toc170209912)

[**ADRES BİLGİLERİ** 64](#_Toc170209913)

# 1. GİRİŞ ve TANITIM

Online açık arttırma sistemlerinin mobil uygulamalar aracılığıyla erişilebilir hale gelmesi, kullanıcıların zaman ve mekandan bağımsız olarak katılımını sağlayarak esneklik sunar. Bu sayede potansiyel alıcılar ve satıcılar, her an her yerden ürünlerini alıcılarla buluşturabilir veya istedikleri ürünleri araştırıp satın alabilirler.

Ayrıca, mobil uygulamaların sağladığı kullanıcı dostu arayüzler ve hızlı erişim özellikleri sayesinde, kullanıcılar kolayca ürünleri inceleyebilir, teklif verebilir ve açık arttırmaları takip edebilirler. Bu da ticaretin daha dinamik ve interaktif bir hal almasını sağlar.

Geleneksel açık arttırmaların aksine, online platformlar üzerinden yapılan açık arttırmalar, daha geniş bir katılımcı kitlesine ulaşma imkanı sunar. Böylelikle alıcılar çeşitli ürünler arasından seçim yaparken, satıcılar da ürünlerini daha geniş bir potansiyel müşteri kitlesine sunma fırsatı bulurlar.

Son olarak, mobil uygulamaların sunduğu anlık bildirimler sayesinde kullanıcılar, takip ettikleri açık arttırmalar hakkında güncel bilgilere hızla erişebilirler. Bu da hem alıcıların hem de satıcıların zamanlarını verimli bir şekilde kullanmalarına yardımcı olur.

Kısacası, mobil uygulamaların online açık arttırma sistemlerine entegrasyonu, ticaretin daha kolay, erişilebilir ve etkili bir şekilde gerçekleştirilmesini sağlar, böylece küresel ticaretin ve insanların ticari etkileşimlerinin önündeki engelleri azaltır.

## 1.1. Sitemin Çalışma Mantığı

Projede flutter çerçevesi kullanılarak bir android mobil uygulama oluşturuyoruz. Flutter kullanmamızın sebebi çapraz platformlarda uygulamanın yapılmasını sağlamaktır. Çapraz platform sayesinde uygulamanın geliştirilmesinde şirketler için tasaruf sağlaması bakımından ilk temeli flutter ile atmak istedim.

Sistemde her kullanıcı kayıt – giriş ekranları var. Bu ekranlarda kişiler kayıt olabiliyor ve giriş olabiliyor. Kullanıcılar giriş yaptıktan sonra herkes anasayfaya ürünbaşlığını ve fotoğrafını koyarak pratik bir şekilde açık arttırmaya sunmak istediği ürünü paylaşabiliyor.

Anasayfada herkes tarafından paylaşılan açık arttırmalar ve bu açık arttırmanın ne zaman sonlanacağına dair countdown timer bulunmakta.

Açık arttırmada zaman 0 olduktan sonra en yüksek teklifi kim verdiyse o kişinin kazandığı açık arttırmalar sayfasına ürün konuluyor.

Uygulamaya kayıt olup giriş yapabilen herkes ürünlere teklifler verebiliyor.

Ürünleri açık arttırmaya sunmaktan vazgeçen kullanıcılar eğer isterlerse paylaştıkları açık arttırmaları silip yenisini paylaşabilirler. Açık arttırmaya ürün sunan kişiler aynı zamanda bir yandan başka açık arttırmalara çoklu şekilde teklif verebilir. Böylelikle zaman kaybının önüne geçmekle beraber kaçırılmaması gereken birden fazla ürün varsa hepsine aynı zamanda yetişilebiliyor.2. PROJE KONUSU HAKKINDA GENEL BİLGİLER

Kurulan sistem, folder structer architecture yapısına uygun olarak tasarlanmıştır. Böyle tasarlamamızın sebebi uygulamanın geliştirilmesinin basitleştirilmesi için tasarlanmıştır. Böylelikle uygulama her geçen zaman bir şirket çalışanları ile takım çalışması ile kolaylıkla geliştirilebilecek.

## 2.1 Authentication Aşaması

Bu aşama da kişileri kayıt etme, giriş yaptırma, kullanıcı bilgilerini girme gibi alanlar yer alıyor.

**1.** Bu aşamada sign\_in.dart sayfasında kullanıcılar firebase veritabanı ile giriş yapılması sağlanıyor.

**2.** sign\_up.dart sayfasında kullanıcıların firebase veritabanına eposta ve passwordlar gibi çeşitli kullanıcı bilgilerini kaydediyoruz.

**3.** sign\_up\_info.dart sayfasında ise kullanıcı isim , soyisim ve username bilgilerini alıyoruz. Bu bilgilerin alınmasının sebebi uygulamanın çeşitli sayfalarında uygulamanın işlevselliği ve göze hitap etmesi için gerekli oluyor.

**4.** auth\_controller.dart sayfasında ise auth\_repository sayfasındaki repositoryleri kullanıyoruz.

**5**.auth\_repository.dart sayfasında ise authentication ile ilgili firebasede çeşitli işlemlerin olduğu bir sayfadır. Bu sayfalara provider kullanarak tüm proje sayfalarında public olarak erişebiliyorum.

İşte sayfalara ait çeşitli görseller :

import 'dart:async';

import 'package:calisma3/core/constants/colors.dart';

import 'package:calisma3/core/constants/paths.dart';

import 'package:calisma3/data/services/audio\_player.dart';

import 'package:calisma3/data/repositories/auth\_controller.dart';

import 'package:calisma3/presentation/screens/sign\_up.dart';

import 'package:calisma3/presentation/screens/home.dart';

import 'package:calisma3/core/utils/sizes.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

class SignIn extends ConsumerStatefulWidget {

  const SignIn({super.key});

  @override

  ConsumerState<SignIn> createState() => \_SignInState();

}

class \_SignInState extends ConsumerState<SignIn> {

  final TextEditingController \_emailController = TextEditingController();

  final TextEditingController \_passwordController = TextEditingController();

  final AudioPlayerService \_audioPlayerService = AudioPlayerService();

  final \_formKey = GlobalKey<FormState>();

  bool \_rememberMe = false;

  @override

  void initState() {

    super.initState();

    \_loadUserData();

  }

  Future<void> \_loadUserData() async {

    final prefs = await SharedPreferences.getInstance();

    final email = prefs.getString('email');

    final password = prefs.getString('password');

    final rememberMe = email != null && password != null;

    if (rememberMe) {

      setState(() {

        \_emailController.text = email;

        \_passwordController.text = password;

        \_rememberMe = rememberMe;

      });

    }

  }

  Future<void> \_validateAndSignIn(BuildContext context, WidgetRef ref) async {

    if (\_formKey.currentState!.validate()) {

      ref

          .read(authControllerProvider)

          .signInWithEmailAndPassword(

              email: \_emailController.text,

              password: \_passwordController.text,

              rememberMe: \_rememberMe)

          .then((value) async {

        await \_audioPlayerService.playSound(audioImage);

        return Navigator.pushAndRemoveUntil(context,

            MaterialPageRoute(builder: (\_) => const Home()), (route) => false);

      });

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Stack(

        alignment: Alignment.bottomCenter,

        children: [

          //\* Wallpaper

          Container(

            height: MediaQuery.of(context).size.height,

            width: double.infinity,

            decoration: const BoxDecoration(

                image: DecorationImage(

                    image: AssetImage(signInImage), fit: BoxFit.cover)),

          ),

          //\* Sign in box

          AspectRatio(

            aspectRatio: 1,

            child: Container(

              padding: const EdgeInsets.all(15),

              decoration: const BoxDecoration(

                  color: containerColor,

                  borderRadius: BorderRadius.only(

                    topLeft: Radius.circular(20),

                    topRight: Radius.circular(20),

                  )),

              child: Form(

                  autovalidateMode: AutovalidateMode.onUserInteraction,

                  key: \_formKey,

                  child: SingleChildScrollView(

                    child: Column(

                      children: [

                        //^ Sign in text

                        Container(

                          alignment: Alignment.centerLeft,

                          padding: vertical(10),

                          child: const Text("Sign In",

                              style:

                                  TextStyle(color: titleColor, fontSize: 24)),

                        ),

                        //^ Email texformfield

                        Padding(

                          padding: vertical(5),

                          child: TextFormField(

                            controller: \_emailController,

                            validator: (value) {

                              if (value!.isEmpty) {

                                return 'Email is required';

                              }

                              return null;

                            },

                            decoration: InputDecoration(

                                labelText: 'Email',

                                border: OutlineInputBorder(

                                    borderSide:

                                        const BorderSide(color: borderColor),

                                    borderRadius: BorderRadius.circular(4))),

                          ),

                        ),

                        //^ Password textformfield

                        Padding(

                          padding: vertical(5),

                          child: TextFormField(

                            controller: \_passwordController,

                            validator: (value) {

                              if (value!.isEmpty) {

                                return 'Password is required';

                              }

                              return null;

                            },

                            obscureText: true,

                            decoration: InputDecoration(

                                labelText: 'Password',

                                border: OutlineInputBorder(

                                    borderSide:

                                        const BorderSide(color: borderColor),

                                    borderRadius: BorderRadius.circular(4))),

                          ),

                        ),

                        //^ Sign in button

                        Consumer(builder: (context, ref, child) {

                          return Padding(

                            padding: vertical(10),

                            child: MaterialButton(

                              minWidth: double.infinity,

                              onPressed: () => \_validateAndSignIn(context, ref),

                              shape: RoundedRectangleBorder(

                                  borderRadius: BorderRadius.circular(4)),

                              color: signInButtonColor,

                              child: Padding(

                                padding: vertical(10),

                                child: const Text("Sign In",

                                    style: TextStyle(color: containerColor)),

                              ),

                            ),

                          );

                        }),

                        //^ Remember Me

                        Row(

                          children: [

                            Checkbox(

                                value: \_rememberMe,

                                onChanged: (bool? value) {

                                  setState(() {

                                    \_rememberMe = value ?? false;

                                  });

                                }),

                            const Text(

                              "Remember Me",

                              style: TextStyle(color: titleColor),

                            )

                          ],

                        ),

                        //^ Don't haven't an account Sign Up text and textButton

                        Padding(

                          padding: vertical(10),

                          child: Row(

                            children: [

                              const Text(

                                "Don't haven't an account",

                                style: TextStyle(

                                    color: textButtonTextColor, fontSize: 14),

                              ),

                              TextButton(

                                onPressed: () async {

                                  Navigator.push(

                                      context,

                                      MaterialPageRoute(

                                          builder: (\_) => const SignUp()));

                                  await \_audioPlayerService

                                      .playSound(audioImage);

                                },

                                child: const Text(

                                  "Sign Up",

                                  style: TextStyle(

                                      color: signInButtonColor, fontSize: 12),

                                ),

                              )

                            ],

                          ),

                        )

                      ],

                    ),

                  )),

            ),

          ),

        ],

      ),

    );

  }

}

import 'package:calisma3/data/services/audio\_player.dart';

import 'package:calisma3/core/constants/colors.dart';

import 'package:calisma3/core/constants/paths.dart';

import 'package:calisma3/core/utils/sizes.dart';

import 'package:calisma3/data/repositories/auth\_controller.dart';

import 'package:calisma3/presentation/screens/sign\_in.dart';

import 'package:calisma3/presentation/screens/sign\_up\_info.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

class SignUp extends StatefulWidget {

  const SignUp({super.key});

  @override

  State<SignUp> createState() => \_SignUpState();

}

class \_SignUpState extends State<SignUp> {

  final TextEditingController \_emailController = TextEditingController();

  final TextEditingController \_passwordController = TextEditingController();

  final TextEditingController \_passwordAgainController =

      TextEditingController();

  final \_formKey = GlobalKey<FormState>();

  final AudioPlayerService \_audioPlayerService = AudioPlayerService();

  Future<void> \_validateAndSignUp(BuildContext context, WidgetRef ref) async {

    if (\_formKey.currentState!.validate()) {

      try {

        await ref.read(authControllerProvider).signUpWithEmailAndPassword(

              email: \_emailController.text,

              password: \_passwordController.text,

            );

        await \_audioPlayerService.playSound(audioImage);

        Navigator.pushReplacement(

          context,

          MaterialPageRoute(

            builder: (\_) => SignUpInfo(

              email: \_emailController.text,

              password: \_passwordController.text,

            ),

          ),

        );

      } catch (e) {

        if (e is FirebaseAuthException && e.code == 'email-already-in-use') {

          ScaffoldMessenger.of(context).removeCurrentSnackBar();

          ScaffoldMessenger.of(context).showSnackBar(

            const SnackBar(

              content: Text('This email is already registered'),

            ),

          );

        } else {

          ScaffoldMessenger.of(context).showSnackBar(

            const SnackBar(

              content: Text('An error occurred!'),

            ),

          );

        }

      }

    } else {

      ScaffoldMessenger.of(context).showSnackBar(

        const SnackBar(

          content: Text('Please enter values correctly'),

        ),

      );

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Stack(

        alignment: Alignment.bottomCenter,

        children: [

          //\* Wallpaper

          Container(

            height: MediaQuery.of(context).size.height,

            width: double.infinity,

            decoration: const BoxDecoration(

                image: DecorationImage(

                    image: AssetImage(signUpImage), fit: BoxFit.cover)),

          ),

          //\* Sign up box

          AspectRatio(

            aspectRatio: 0.90,

            child: Container(

              padding: paddingAll(15),

              decoration: const BoxDecoration(

                  color: containerColor,

                  borderRadius: BorderRadius.only(

                    topLeft: Radius.circular(20),

                    topRight: Radius.circular(20),

                  )),

              child: Form(

                autovalidateMode: AutovalidateMode

                    .onUserInteraction, // Sadece kullanıcı alanla etkileşime girdiği zaman form kullanılacak.

                key: \_formKey,

                child: SingleChildScrollView(

                  child: Column(

                    children: [

                      //^ Sign up text

                      Container(

                        alignment: Alignment.centerLeft,

                        padding: vertical(10),

                        child: const Text("Sign Up",

                            style: TextStyle(color: titleColor, fontSize: 24)),

                      ),

                      //^ Email texformfield

                      Padding(

                        padding: vertical(5),

                        child: TextFormField(

                          controller: \_emailController,

                          validator: (value) {

                            if (value!.isEmpty) {

                              return 'Email is required';

                            }

                            return null;

                          },

                          decoration: InputDecoration(

                              labelText: 'Email',

                              border: OutlineInputBorder(

                                  borderSide:

                                      const BorderSide(color: borderColor),

                                  borderRadius: BorderRadius.circular(4))),

                        ),

                      ),

                      //^ Password textformfield

                      Padding(

                        padding: vertical(5),

                        child: TextFormField(

                          controller: \_passwordController,

                          validator: (value) {

                            if (value!.isEmpty) {

                              return 'Password is required';

                            }

                            return null;

                          },

                          obscureText: true,

                          decoration: InputDecoration(

                              labelText: 'Password',

                              border: OutlineInputBorder(

                                  borderSide:

                                      const BorderSide(color: borderColor),

                                  borderRadius: BorderRadius.circular(4))),

                        ),

                      ),

                      //^ PasswordAgain textformfield

                      Padding(

                        padding: vertical(5),

                        child: TextFormField(

                          controller: \_passwordAgainController,

                          validator: (value) {

                            if (value! != \_passwordController.text) {

                              return 'passwords do not match';

                            }

                            return null;

                          },

                          obscureText: true,

                          decoration: InputDecoration(

                              labelText: 'Password Again',

                              border: OutlineInputBorder(

                                  borderSide:

                                      const BorderSide(color: borderColor),

                                  borderRadius: BorderRadius.circular(4))),

                        ),

                      ),

                      //^ Sign up button

                      Consumer(

                        builder: (context, ref, child) {

                          return Padding(

                            padding: vertical(10),

                            child: MaterialButton(

                              minWidth: double.infinity,

                              onPressed: () => \_validateAndSignUp(context, ref),

                              shape: RoundedRectangleBorder(

                                  borderRadius: BorderRadius.circular(4)),

                              color: continueButtonColor,

                              child: Padding(

                                padding: vertical(10),

                                child: const Text(

                                  "Sign Up",

                                  style: TextStyle(color: containerColor),

                                ),

                              ),

                            ),

                          );

                        },

                      ),

                      //^ Do you have an account? Sign Up text and textButton

                      Padding(

                        padding: vertical(10),

                        child: Row(

                          children: [

                            const Text(

                              "Do you have an account?",

                              style: TextStyle(

                                  color: textButtonTextColor, fontSize: 14),

                            ),

                            TextButton(

                              onPressed: () async {

                                Navigator.push(

                                    context,

                                    MaterialPageRoute(

                                        builder: (\_) => SignIn()));

                                await \_audioPlayerService.playSound(audioImage);

                              },

                              child: const Text(

                                "Sign In",

                                style: TextStyle(

                                    color: signInButtonColor, fontSize: 12),

                              ),

                            )

                          ],

                        ),

                      )

                    ],

                  ),

                ),

              ),

            ),

          )

        ],

      ),

    );

  }

}

import 'package:calisma3/data/services/audio\_player.dart';

import 'package:calisma3/core/constants/colors.dart';

import 'package:calisma3/core/constants/paths.dart';

import 'package:calisma3/core/utils/sizes.dart';

import 'package:calisma3/data/repositories/auth\_controller.dart';

import 'package:calisma3/presentation/screens/home.dart';

import 'package:calisma3/data/models/user\_model.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

class SignUpInfo extends StatefulWidget {

  const SignUpInfo({super.key, required this.email, required this.password});

  final String email;

  final String password;

  @override

  State<SignUpInfo> createState() => \_SignUpInfoState();

}

class \_SignUpInfoState extends State<SignUpInfo> {

  final TextEditingController \_nameController = TextEditingController();

  final TextEditingController \_surnameController = TextEditingController();

  final TextEditingController \_usernameController = TextEditingController();

  final AudioPlayerService \_audioPlayerService = AudioPlayerService();

  final \_formKey = GlobalKey<FormState>();

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Stack(

        alignment: Alignment.bottomCenter,

        children: [

          //\* Wallpaper

          Container(

            height: MediaQuery.of(context).size.height,

            width: double.infinity,

            decoration: const BoxDecoration(

                image: DecorationImage(

                    image: AssetImage(signUpInfoImage), fit: BoxFit.cover)),

          ),

          //\* Sign up box

          AspectRatio(

            aspectRatio: 1,

            child: Container(

              padding: const EdgeInsets.all(15),

              decoration: const BoxDecoration(

                  color: containerColor,

                  borderRadius: BorderRadius.only(

                    topLeft: Radius.circular(20),

                    topRight: Radius.circular(20),

                  )),

              child: Form(

                autovalidateMode: AutovalidateMode.onUserInteraction,

                key: \_formKey,

                child: SingleChildScrollView(

                  child: Column(

                    children: [

                      //^ SignUpInfo text

                      Container(

                        alignment: Alignment.centerLeft,

                        padding: vertical(10),

                        child: const Text("Sign Up Info",

                            style: TextStyle(color: titleColor, fontSize: 24)),

                      ),

                      //^ Name texformfield

                      Padding(

                        padding: vertical(5),

                        child: TextFormField(

                          controller: \_nameController,

                          validator: (value) {

                            if (value!.isEmpty) {

                              return 'Name is required';

                            }

                            return null;

                          },

                          decoration: InputDecoration(

                              labelText: 'Name',

                              border: OutlineInputBorder(

                                  borderSide:

                                      const BorderSide(color: borderColor),

                                  borderRadius: BorderRadius.circular(4))),

                        ),

                      ),

                      //^ Surname textformfield

                      Padding(

                        padding: vertical(5),

                        child: TextFormField(

                          controller: \_surnameController,

                          validator: (value) {

                            if (value!.isEmpty) {

                              return 'Surname is required';

                            }

                            return null;

                          },

                          decoration: InputDecoration(

                              labelText: 'Surname',

                              border: OutlineInputBorder(

                                  borderSide:

                                      const BorderSide(color: borderColor),

                                  borderRadius: BorderRadius.circular(4))),

                        ),

                      ),

                      //^ Username textformfield

                      Padding(

                        padding: vertical(5),

                        child: TextFormField(

                          controller: \_usernameController,

                          validator: (value) {

                            if (value!.isEmpty) {

                              return 'Username is required';

                            }

                            return null;

                          },

                          decoration: InputDecoration(

                              labelText: 'Username',

                              border: OutlineInputBorder(

                                  borderSide:

                                      const BorderSide(color: borderColor),

                                  borderRadius: BorderRadius.circular(4))),

                        ),

                      ),

                      //^ Continue button

                      Consumer(

                        builder: (context, ref, child) {

                          return Padding(

                            padding: vertical(10),

                            child: MaterialButton(

                              minWidth: double.infinity,

                              onPressed: () async {

                                if (\_formKey.currentState!.validate()) {

                                  UserModel userModel = UserModel(

                                      name: \_nameController.text,

                                      surname: \_surnameController.text,

                                      username: \_usernameController.text,

                                      email: widget.email,

                                      password: widget.password);

                                  ref

                                      .read(authControllerProvider)

                                      .storeUserInfoToFirebase(userModel)

                                      .then((value) async {

                                    await \_audioPlayerService

                                        .playSound(audioImage);

                                    return Navigator.pushAndRemoveUntil(

                                        context,

                                        MaterialPageRoute(

                                            builder: (\_) => const Home()),

                                        (route) => false);

                                  });

                                }

                              },

                              shape: RoundedRectangleBorder(

                                  borderRadius: BorderRadius.circular(4)),

                              color: continueButtonColor,

                              child: Padding(

                                padding: vertical(10),

                                child: const Text(

                                  "Continue",

                                  style: TextStyle(color: containerColor),

                                ),

                              ),

                            ),

                          );

                        },

                      ),

                    ],

                  ),

                ),

              ),

            ),

          )

        ],

      ),

    );

  }

}

import 'package:calisma3/data/repositories/auth\_repository.dart';

import 'package:calisma3/data/models/user\_model.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

final authControllerProvider = Provider<AuthController>(

  (ref) => AuthController(authRepository: ref.watch(authRepositoryProvider)),

);

class AuthController {

  final AuthRepository authRepository;

  AuthController({required this.authRepository});

  Future<void> signInWithEmailAndPassword({

    required String email,

    required String password,

    bool rememberMe = false,

  }) async {

    if (rememberMe) {

      final prefs = await SharedPreferences.getInstance();

      await prefs.setString('email', email);

      await prefs.setString('password', password);

    }

    return authRepository.signInWithEmailAndPassword(

      email: email,

      password: password,

    );

  }

  Future<void> signUpWithEmailAndPassword({

    required String email,

    required String password,

  }) async {

    return authRepository.signUpWithEmailAndPassword(

      email: email,

      password: password,

    );

  }

  Future<void> storeUserInfoToFirebase(UserModel userModel) async {

    return authRepository.storeUserInfoToFirebase(userModel);

  }

  Future<void> signOut() async {

    await authRepository.signOut();

    final prefs = await SharedPreferences.getInstance();

    await prefs.remove('email');

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import 'package:calisma3/data/models/user\_model.dart';

final authRepositoryProvider = Provider<AuthRepository>(

  (ref) => AuthRepository(

    auth: FirebaseAuth.instance,

    firebaseFirestore: FirebaseFirestore.instance,

  ),

);

class AuthRepository {

  final FirebaseAuth auth;

  final FirebaseFirestore firebaseFirestore;

  AuthRepository({required this.auth, required this.firebaseFirestore});

  Future<void> signInWithEmailAndPassword({

    required String email,

    required String password,

  }) async {

    await auth.signInWithEmailAndPassword(email: email, password: password);

  }

  Future<void> signUpWithEmailAndPassword({

    required String email,

    required String password,

  }) async {

    await auth.createUserWithEmailAndPassword(email: email, password: password);

  }

  Future<void> signOut() async {

    await auth.signOut();

  }

  Future<void> storeUserInfoToFirebase(UserModel userModel) async {

    userModel.profilePhoto ??=

        "https://static.vecteezy.com/system/resources/previews/036/280/650/original/default-avatar-profile-icon-social-media-user-image-gray-avatar-icon-blank-profile-silhouette-illustration-vector.jpg";

    userModel.uid = auth.currentUser!.uid;

    await firebaseFirestore

        .collection("users")

        .doc(userModel.uid)

        .set(userModel.toMap());

  }

  Future<void> deleteUser() async {

    final user = auth.currentUser;

    if (user != null) {

      await user.delete();

    }

  }

}

## 2.2 Profile aşaması

Bu aşama da kullanıcıların çeşitli popular sosyal medya platformlarında ki gibi profil bilgileri, profil fotoğrafı, profil düzenleme, çıkış yapma ve kullanıcı silme gibi çeşitli işlemler bulunuyor.

**1.** Edit profile kısmında kullanıcılar kayıt oldukları isimlerini, soyisimlerini, epostalarını, password işlemlerini, kullanıcı adlarını veya profil fotoğraflarını değiştirme imkanı sunuyor.

**2.** Edit profile yapılan değişikler tüm sayfalarda geçerli oluyor. Örneğin kullanıcı profil fotoğrafını değiştirir ve kullanıcı adını değiştirirse paylaştığı açık arttırmalarda da aynı şekilde ve teklif verdiği açık arttırmalarda da aynı şekilde otomatik olarak değişiyor.

**3.** Logout kısmında kullanıcının kendi hesabından çıkış yapıp sign\_in ekranına geçilmesi sağlanıyor

**4.** Delete user butonunda ise kullanıcı hesabını silmekle beraber firebaseden kullanıcıya ait tüm veriler siliniyor . Böylelikle gereksiz kayıtların depolanması gibi yükten kaçınılıyor.

İşte profile aşamasına ait detaylandırılmadan temeli verilen görseller

import 'package:calisma3/data/repositories/profile\_controller.dart';

import 'package:calisma3/presentation/widgets/delete\_user\_button\_profile.dart';

import 'package:calisma3/presentation/widgets/edit\_profile\_button\_profile.dart';

import 'package:calisma3/presentation/widgets/logout\_button\_profile.dart';

import 'package:calisma3/presentation/widgets/profile\_image\_profile.dart';

import 'package:calisma3/presentation/widgets/user\_fullname\_text\_profile.dart';

import 'package:calisma3/presentation/widgets/username\_text\_profile.dart';

import 'package:calisma3/data/models/user\_model.dart';

import 'package:calisma3/core/utils/sizes.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

class Profile extends ConsumerStatefulWidget {

  const Profile({super.key});

  @override

  ProfileState createState() => ProfileState();

}

class ProfileState extends ConsumerState<Profile> {

  late Future<UserModel> \_userFuture;

  @override

  void initState() {

    super.initState();

    \_userFuture = ref.read(profileControllerProvider).getUser();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: SafeArea(

        // Profile fotoğrafı taşmasın diyerekten safearea oluşturuyorum.

        child: Padding(

          padding: paddingAll(15),

          child: FutureBuilder<UserModel>(

            future: \_userFuture,

            builder: (context, snapshot) {

              if (snapshot.connectionState == ConnectionState.waiting) {

                return const Center(child: CircularProgressIndicator());

              } else if (snapshot.hasError) {

                return const Center(child: Text('An error occurred'));

              } else if (!snapshot.hasData) {

                return const Center(child: Text('No data available'));

              }

              final userModel = snapshot.data!;

              return Align(

                alignment: Alignment.center,

                child: Column(

                  mainAxisAlignment: MainAxisAlignment.center,

                  children: [

                    ProfileImageProfile(userModel: userModel),

                    UsernameTextProfile(username: userModel.username),

                    UserFullNameTextProfile(

                        name: userModel.name, surname: userModel.surname),

                    EditProfileButtonProfile(context: context),

                    LogoutButtonProfile(ref: ref, context: context),

                    DeleteUserButtonProfile(ref: ref, context: context),

                  ],

                ),

              );

            },

          ),

        ),

      ),

    );

  }

}

import 'dart:io';

import 'package:cached\_network\_image/cached\_network\_image.dart';

import 'package:calisma3/presentation/widgets/user\_info\_form\_profile\_edit.dart';

import 'package:calisma3/data/services/audio\_player.dart';

import 'package:calisma3/core/constants/colors.dart';

import 'package:calisma3/core/constants/paths.dart';

import 'package:calisma3/core/utils/sizes.dart';

import 'package:calisma3/data/repositories/profile\_controller.dart';

import 'package:calisma3/presentation/screens/home.dart';

import 'package:calisma3/data/models/post\_model.dart';

import 'package:calisma3/data/models/user\_model.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/foundation.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import 'package:image\_picker/image\_picker.dart';

class EditProfile extends ConsumerStatefulWidget {

  const EditProfile({super.key});

  @override

  ConsumerState<EditProfile> createState() => \_EditProfileState();

}

class \_EditProfileState extends ConsumerState<EditProfile> {

  final \_formKey = GlobalKey<FormState>();

  final AudioPlayerService \_audioPlayerService = AudioPlayerService();

  File? \_image;

  List<PostModel> posts = [];

  final TextEditingController \_nameController = TextEditingController();

  final TextEditingController \_lastNameController = TextEditingController();

  final TextEditingController \_userNameController = TextEditingController();

  final TextEditingController \_emailController = TextEditingController();

  final TextEditingController \_passwordController = TextEditingController();

  @override

  void initState() {

    super.initState();

    \_loadUserData();

    fetchPosts();

  }

  Future<void> \_loadUserData() async {

    final userModel = await ref.read(profileControllerProvider).getUser();

    setState(() {

      \_nameController.text = userModel.name;

      \_lastNameController.text = userModel.surname;

      \_userNameController.text = userModel.username;

      \_emailController.text = userModel.email;

      \_passwordController.text = userModel.password;

    });

  }

  Future<void> \_pickImage() async {

    final pickedFile =

        await ImagePicker().pickImage(source: ImageSource.gallery);

    if (pickedFile != null) {

      setState(() {

        \_image = File(pickedFile.path);

      });

    }

  }

  Future<void> fetchPosts() async {

    QuerySnapshot querySnapshot =

        await FirebaseFirestore.instance.collection("posts").get();

    List<PostModel> fetchedPosts = [];

    for (var doc in querySnapshot.docs) {

      fetchedPosts.add(PostModel.fromMap(doc.data() as Map<String, dynamic>));

    }

    setState(() {

      posts = fetchedPosts;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: const Text("Profil Düzenle")),

      body: buildBody(context),

    );

  }

  Widget buildBody(BuildContext context) {

    return SafeArea(

      child: FutureBuilder<UserModel>(

        future: ref.read(profileControllerProvider).getUser(),

        builder: (context, snapshot) {

          if (snapshot.connectionState == ConnectionState.waiting) {

            return const Center(child: CircularProgressIndicator());

          } else if (snapshot.hasError) {

            return const Center(child: Text('Bir hata oluştu'));

          } else if (!snapshot.hasData) {

            return const Center(child: Text('Veri bulunamadı'));

          }

          final userModel = snapshot.data!;

          return Padding(

            padding: paddingAll(15),

            child: Align(

              alignment: Alignment.center,

              child: Padding(

                padding: vertical(10),

                child: Form(

                  key: \_formKey,

                  child: Column(

                    children: [

                      //^ Profile Photo

                      Padding(

                        padding: vertical(10),

                        child: GestureDetector(

                          onTap: \_pickImage,

                          child: CircleAvatar(

                            backgroundColor: profilePhotoCircleColor,

                            radius: 70,

                            child: CircleAvatar(

                              radius: 68,

                              backgroundImage: \_image == null

                                  ? CachedNetworkImageProvider(

                                      userModel.profilePhoto!)

                                  : FileImage(\_image!) as ImageProvider,

                            ),

                          ),

                        ),

                      ),

                      //^ TextformField's

                      UserInfoFormProfileedit(

                        nameController: \_nameController,

                        lastNameController: \_lastNameController,

                        emailController: \_emailController,

                        passwordController: \_passwordController,

                        usernameController: \_userNameController,

                      ),

                      //^ Save Button

                      Padding(

                        padding: vertical(10),

                        child: MaterialButton(

                          minWidth: double.infinity,

                          onPressed: () async {

                            if (\_formKey.currentState!.validate()) {

                              userModel.name = \_nameController.text;

                              userModel.surname = \_lastNameController.text;

                              userModel.email = \_emailController.text;

                              userModel.username = \_userNameController.text;

                              userModel.password = \_passwordController.text;

                              try {

                                await ref

                                    .read(profileControllerProvider)

                                    .updateUser(

                                      userModel,

                                      posts,

                                      profilePhoto: \_image,

                                    );

                                // ignore: use\_build\_context\_synchronously

                                ScaffoldMessenger.of(context).showSnackBar(

                                  const SnackBar(

                                    content:

                                        Text('Profil başarıyla güncellendi'),

                                  ),

                                );

                                Navigator.pushAndRemoveUntil(

                                  // ignore: use\_build\_context\_synchronously

                                  context,

                                  MaterialPageRoute(

                                      builder: (\_) => const Home()),

                                  (route) => false,

                                );

                                await \_audioPlayerService.playSound(audioImage);

                              } catch (e, stackTrace) {

                                if (kDebugMode) {

                                  print('Hata: $e');

                                }

                                if (kDebugMode) {

                                  print('Hata izi: $stackTrace');

                                }

                                // ignore: use\_build\_context\_synchronously

                                ScaffoldMessenger.of(context).showSnackBar(

                                  SnackBar(

                                    content: Text(

                                        'Profil güncellenirken hata oluştu: $e'),

                                  ),

                                );

                              }

                            }

                          },

                          shape: RoundedRectangleBorder(

                            borderRadius: BorderRadius.circular(4),

                          ),

                          color: profilePhotoCircleColor,

                          child: Padding(

                            padding: vertical(10),

                            child: const Text(

                              "Kaydet",

                              style: TextStyle(color: containerColor),

                            ),

                          ),

                        ),

                      )

                    ],

                  ),

                ),

              ),

            ),

          );

        },

      ),

    );

  }

}

## 2.3 Dashboard aşaması

**1.** Bu aşamada dashboard.dart sayfasında uygulamanın anasayfasının olduğu yer oluyor.

**2.** dashboard sayfasında create button ve firebasedeki açık arttırmaya sunulan ürünler stack yapısı ile beraber yerleştiriliyor.

**3.** dashboard\_ controller sayfası dashboard\_reposityory sayfasındaki firebase de postlarla ilgili çeşitli işlemlerin yaptığı repositoryleri kullanmamızı sağlıyor.

**4.** dashboard\_ repository sayfasında ise postları yakalama, post silme, postu veri tabanına yerleştirme gibi çeşitli backed işlemleri gerçekleştiriliyor.

import 'package:calisma3/core/constants/paths.dart';

import 'package:calisma3/data/services/audio\_player.dart';

import 'package:calisma3/presentation/screens/auction.dart';

import 'package:calisma3/presentation/widgets/fetch\_data.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import 'package:flutter\_svg/svg.dart';

import 'package:calisma3/presentation/screens/create\_post.dart';

import 'package:calisma3/presentation/widgets/post\_card.dart';

import 'package:calisma3/data/models/post\_model.dart';

import 'package:calisma3/data/models/user\_model.dart';

class Dashboard extends ConsumerStatefulWidget {

  const Dashboard({super.key});

  @override

  ConsumerState<Dashboard> createState() => \_DashboardState();

}

class \_DashboardState extends ConsumerState<Dashboard> {

  final AudioPlayerService \_audioPlayerService = AudioPlayerService();

  List<PostModel> posts = [];

  UserModel userModel =

      UserModel(name: "", surname: "", username: "", email: "", password: "");

  @override

  void initState() {

    fetchPosts(ref, \_onPostsFetched);

    fetchCurrentUser(ref, \_onUserFetched);

    super.initState();

  }

  void \_onPostsFetched(List<PostModel> fetchedPosts) {

    setState(() {

      posts = fetchedPosts;

    });

  }

  void \_onUserFetched(UserModel fetchedUserModel) {

    setState(() {

      userModel = fetchedUserModel;

    });

  }

  @override

  void dispose() {

    \_audioPlayerService.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: buildBody(context),

    );

  }

  Widget buildBody(BuildContext context) {

    return Stack(

      alignment: Alignment.bottomCenter,

      children: [

        GridView.builder(

          gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(

            crossAxisCount: 1,

          ),

          itemCount: posts.length,

          itemBuilder: (BuildContext context, int index) {

            return PostCard(

              user: userModel,

              post: posts[index],

              onImageTap: () {

                Navigator.push(

                    context,

                    MaterialPageRoute(

                        builder: (\_) => Auction(

                              postModel: posts[index],

                              userModel: userModel,

                            )));

              },

            );

          },

        ),

        Positioned(

          bottom: 16.0,

          right: 16.0,

          child: IconButton(

            onPressed: () async {

              await \_audioPlayerService.playSound(audioAll);

              Navigator.push(

                  context, MaterialPageRoute(builder: (\_) => CreatePost()));

            },

            icon: SvgPicture.asset(plusIconSvg),

          ),

        ),

      ],

    );

  }

}

import 'package:calisma3/data/repositories/dashboard\_repository.dart';

import 'package:calisma3/data/models/post\_model.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

final dashboardControllerProvider = Provider((ref) => DashboardController(

    dashboardRepository: ref.watch(dashboardRepositoryProvider)));

class DashboardController {

  final DashboardRepository dashboardRepository;

  DashboardController({required this.dashboardRepository});

  Future<void> storePostInfoToFirebase(PostModel postModel) async {

    return dashboardRepository.storePostInfoToFirebase(postModel);

  }

  Future<List<PostModel>> fetchPosts() async {

    return dashboardRepository.fetchPosts();

  }

  Future<void> deletePost(PostModel postModel) async {

    return dashboardRepository.deletePost(postModel);

  }

}

import 'package:calisma3/data/models/post\_model.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

final dashboardRepositoryProvider = Provider((ref) => DashboardRepository(

    firebaseFirestore: FirebaseFirestore.instance,

    auth: FirebaseAuth.instance));

class DashboardRepository {

  final FirebaseFirestore firebaseFirestore;

  final FirebaseAuth auth;

  DashboardRepository({required this.firebaseFirestore, required this.auth});

  Future<void> storePostInfoToFirebase(PostModel postModel) async {

    postModel.userImage ??=

        "https://static.vecteezy.com/system/resources/previews/036/280/650/original/default-avatar-profile-icon-social-media-user-image-gray-avatar-icon-blank-profile-silhouette-illustration-vector.jpg";

    postModel.uid = auth.currentUser!.uid;

    await firebaseFirestore

        .collection("posts")

        .doc(postModel.id)

        .set(postModel.toMap());

  }

  Future<List<PostModel>> fetchPosts() async {

    QuerySnapshot querySnapshot =

        await FirebaseFirestore.instance.collection("posts").get();

    List<PostModel> fetchedPosts = [];

    for (var doc in querySnapshot.docs) {

      fetchedPosts.add(PostModel.fromMap(doc.data() as Map<String, dynamic>));

    }

    return fetchedPosts;

  }

  Future<void> deletePost(PostModel postModel) async {

    await firebaseFirestore.collection("posts").doc(postModel.id).delete();

  }

}

## 2.3 My auctions aşaması

Bu aşama da kullanıcıların paylaştığı açık arttırma gönderileri listeleniyor. Bunu e ticaret sitelerindeki geçmiş siparişlerim kısmına benzetebiliriz.

**1.** Bu aşamada tüm postları ve mevcut giriş yapan kullanıcıyı çekiyoruz.

**2.** Çekilen postların her bir idsine göre kontrol yapıp ona göre listelenmesi gereken postları listeliyoruz.

**3.** post.uid == usermodel.uid ise o zaman o postu myauctions sayfasında oluştur.

İşte my auctions aşamasına ait detaylandırılmadan temeli verilen görsel

import 'package:calisma3/presentation/widgets/post\_card.dart';

import 'package:calisma3/data/models/post\_model.dart';

import 'package:calisma3/data/models/user\_model.dart';

import 'package:calisma3/presentation/widgets/fetch\_data.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

class MyAuctions extends ConsumerStatefulWidget {

  const MyAuctions({super.key});

  @override

  ConsumerState<MyAuctions> createState() => \_MyAuctionsState();

}

class \_MyAuctionsState extends ConsumerState<MyAuctions> {

  List<PostModel> posts = [];

  UserModel usermodel =

      UserModel(name: "", surname: "", username: "", email: "", password: "");

  @override

  void initState() {

    super.initState();

    fetchPosts(ref, \_onPostsFetched);

    fetchCurrentUser(ref, \_onUserFetched);

  }

  void \_onPostsFetched(List<PostModel> fetchedPosts) {

    setState(() {

      posts = fetchedPosts;

    });

  }

  void \_onUserFetched(UserModel fetchedUserModel) {

    setState(() {

      usermodel = fetchedUserModel;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: \_buildBody(context),

    );

  }

  Widget \_buildBody(BuildContext context) {

    final userPosts = posts.where((post) => post.uid == usermodel.uid).toList();

    if (userPosts.isEmpty) {

      return const Center(child: Text("No Auctions Found"));

    }

    return GridView.builder(

        gridDelegate:

            const SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 1),

        itemCount: userPosts.length,

        itemBuilder: (context, index) {

          final post = userPosts[index];

          return PostCard(post: post, user: usermodel);

        });

  }

}

## 2.4 Won auctions aşaması

Bu aşama da kullanıcıların kazandığı açık arttırma gönderileri listeleniyor. Bunu e ticaret sitelerindeki geçmiş siparişlerim kısmına benzetebiliriz.

**1.** Bu aşamadada my auctions sayfasındaki gibi tüm postları ve mevcut giriş yapan kullanıcıyı çekiyoruz.

**2.** Ancak bu aşamada veritabanına post.offer kısmına kullanıcıların verdiği teklifin yanında kullanıcı adıyla beraber koyduğumdan dolayı ve kullanıcı adına göre postların won auctions sayfasında gösterilmesini istediğimden dolayı post.offerda çeşitli syntatic işlemlerle beraber parse işlemi yapıyorum ve buna göre listeliyorum.

İşte won auctions aşamasına ait detaylandırılmadan temeli verilen görsel

import 'package:calisma3/presentation/widgets/post\_card.dart';

import 'package:calisma3/data/models/post\_model.dart';

import 'package:calisma3/data/models/user\_model.dart';

import 'package:calisma3/presentation/widgets/fetch\_data.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

class WonAuctions extends ConsumerStatefulWidget {

  const WonAuctions({super.key});

  @override

  ConsumerState<WonAuctions> createState() => WonAuctionsState();

}

class WonAuctionsState extends ConsumerState<WonAuctions> {

  List<PostModel> posts = [];

  List<PostModel> wonPosts = [];

  UserModel usermodel =

      UserModel(name: "", surname: "", username: "", email: "", password: "");

  @override

  void initState() {

    super.initState();

    fetchPosts(ref, \_onPostsFetched);

    fetchCurrentUser(ref, \_onUserFetched);

  }

  void \_onPostsFetched(List<PostModel> fetchedPosts) {

    setState(() {

      posts = fetchedPosts;

    });

  }

  void \_onUserFetched(UserModel fetchedUserModel) {

    setState(() {

      usermodel = fetchedUserModel;

    });

  }

  List<PostModel> \_parseOffer() {

    for (var post in posts) {

      if (post.remainingTime == 0) {

        if (post.offer != null && post.offer != "0") {

          final parts = post.offer!.split(' \$');

          String postName = parts.first;

          if (usermodel.username == postName) {

            wonPosts.add(post);

          }

        }

      } else {

        print("Kardeşim sıkıntı burda");

      }

    }

    return wonPosts;

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: \_buildBody(context),

    );

  }

  Widget \_buildBody(BuildContext context) {

    final wonPosts = \_parseOffer();

    if (wonPosts.isEmpty) {

      return const Center(child: Text("No Won Auctions Found"));

    }

    return GridView.builder(

        gridDelegate:

            const SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 1),

        itemCount: wonPosts.length,

        itemBuilder: (context, index) {

          final post = wonPosts[index];

          return PostCard(post: post, user: usermodel);

        });

  }

}

## 2.4 Create post aşaması

Bu aşama da kullanıcılar yeni bir açık arttırma ekleyebiliyorlar. Bu aşamada kullanıcılar eklemek istedikleri açık arttırma görselini ve başlığını koyduktan sonra paylaşabiliyorlar.

**1.** Bu aşamada kullanıcı butona bastıktan sonra kullanıcının verdiği başlık ve görsele göre post oluşturuluyor. Bu postta veritabanına yerleştiriliyor.

**2.** Postların görselleri firebasenin storage alanında images klasöründe saklanıyor.

**3.** Postlar ise firebase firestorede posts çatısı altında saklanıyorlar.

İşte create post aşamasına ait detaylandırılmadan temeli verilen görsel

import 'dart:io';

import 'package:calisma3/core/constants/paths.dart';

import 'package:calisma3/data/services/audio\_player.dart';

import 'package:calisma3/data/repositories/dashboard\_controller.dart';

import 'package:calisma3/data/repositories/profile\_controller.dart';

import 'package:calisma3/presentation/screens/home.dart';

import 'package:calisma3/data/models/post\_model.dart';

import 'package:calisma3/data/services/storage\_service.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import 'package:image\_picker/image\_picker.dart';

class CreatePost extends ConsumerStatefulWidget {

  CreatePost({super.key});

  final AudioPlayerService \_audioPlayerService = AudioPlayerService();

  @override

  ConsumerState<CreatePost> createState() => \_CreatePostState();

}

class \_CreatePostState extends ConsumerState<CreatePost> {

  File? \_image;

  bool visible = false;

  late final PostModel postModel;

  final \_formKey = GlobalKey<FormState>();

  final TextEditingController \_contentTextController = TextEditingController();

  final StorageService \_storageService = StorageService();

  void onPressed() async {

    if (\_formKey.currentState!.validate() && \_image != null) {

      final user = await ref.read(profileControllerProvider).getUser();

      final imageUrl = await \_storageService.uploadImage(\_image!);

      final postId = FirebaseFirestore.instance.collection('posts').doc().id;

      postModel = PostModel(

          id: postId,

          uid: user.uid,

          username: user.username,

          userImage: user.profilePhoto,

          timestamp: Timestamp.now(),

          contentText: \_contentTextController.text,

          contentImage: imageUrl,

          remainingTime: 3600);

      visible = true;

      await ref

          .read(dashboardControllerProvider)

          .storePostInfoToFirebase(postModel)

          .then((\_) async {

        Navigator.pushAndRemoveUntil(

          context,

          MaterialPageRoute(builder: (\_) => const Home()),

          (route) => false,

        );

        await widget.\_audioPlayerService.playSound(audioAll);

      });

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text("Create Post"),

        backgroundColor: Colors.purple,

      ),

      body: Form(

        key: \_formKey,

        autovalidateMode: AutovalidateMode.onUserInteraction,

        child: Container(

          height: MediaQuery.of(context).size.height,

          decoration: const BoxDecoration(

            gradient: LinearGradient(

              colors: [Colors.purple, Colors.deepOrange],

              begin: Alignment.topLeft,

              end: Alignment.bottomRight,

            ),

          ),

          child: Padding(

            padding: const EdgeInsets.all(16),

            child: Column(

              children: [

                const SizedBox(height: 40),

                TextFormField(

                  controller: \_contentTextController,

                  textAlign: TextAlign.center,

                  maxLines: 1,

                  decoration: InputDecoration(

                    filled: true,

                    fillColor: Colors.white.withOpacity(0.8),

                    hintStyle: const TextStyle(fontWeight: FontWeight.bold),

                    border: OutlineInputBorder(

                      borderRadius: BorderRadius.circular(12.0),

                      borderSide: BorderSide.none,

                    ),

                    hintText: 'Please enter your title',

                  ),

                  validator: (value) {

                    if (value == null || value.isEmpty) {

                      return 'Content is required';

                    }

                    return null;

                  },

                ),

                const SizedBox(height: 16),

                GestureDetector(

                  onTap: () async {

                    final pickedFile = await ImagePicker()

                        .pickImage(source: ImageSource.gallery);

                    if (pickedFile != null) {

                      setState(() {

                        \_image = File(pickedFile.path);

                      });

                    }

                  },

                  child: Container(

                    height: 300,

                    width: double.infinity,

                    decoration: BoxDecoration(

                      color: Colors.white.withOpacity(0.8),

                      border: Border.all(color: Colors.grey),

                      borderRadius: BorderRadius.circular(12),

                      boxShadow: const [

                        BoxShadow(

                          color: Colors.black26,

                          offset: Offset(0, 4),

                          blurRadius: 8,

                        ),

                      ],

                    ),

                    child: \_image == null

                        ? const Center(

                            child: Text(

                              'Tap to select an image',

                              style: TextStyle(

                                color: Colors.black54,

                                fontWeight: FontWeight.bold,

                              ),

                            ),

                          )

                        : ClipRRect(

                            borderRadius: BorderRadius.circular(12),

                            child: Image.file(

                              \_image!,

                              fit: BoxFit.cover,

                            ),

                          ),

                  ),

                ),

                const SizedBox(height: 16),

                Consumer(

                  builder:

                      (BuildContext context, WidgetRef ref, Widget? child) {

                    return MaterialButton(

                      onPressed: onPressed,

                      minWidth: double.infinity,

                      shape: RoundedRectangleBorder(

                        borderRadius: BorderRadius.circular(12),

                      ),

                      color: Colors.deepOrange,

                      child: const Padding(

                        padding: EdgeInsets.symmetric(vertical: 16),

                        child: Text(

                          "Create",

                          style: TextStyle(

                            color: Colors.white,

                            fontWeight: FontWeight.bold,

                            fontSize: 18,

                          ),

                        ),

                      ),

                    );

                  },

                ),

                const SizedBox(height: 16),

                Visibility(

                  visible: visible,

                  child: const CircularProgressIndicator(),

                )

              ],

            ),

          ),

        ),

      ),

    );

  }

}

## 2.5 Auction aşaması

Bu aşama da ise kullanıcılar açık arttırmaya girdiği aşamadır.

**1.** Her açık arttırmanın kendi auction sayfası bulunmakta ve her açık arttırmada sayaç geriye doğru saymakta. Bu sayaçlar post oluşturulduktan sonra otomatik olarak başlıyor. Aynı zamanda firebase i etkili ve tasaruflu kullanabilmek için sürekli sayaç değerini kaydetmek yerine uygulamayı alta alındığında ve kapatıldığında uygulama daki timeların hepsi otomatik olarak firebasede güncelleniyor.

**2.** Kişiler burada teklifler veriyor ve en yüksek teklif anasayfa da kişinin kullanıcı adı ile beraber gösterilmektedir.

İşte görsel :

import 'package:calisma3/core/constants/paths.dart';

import 'package:calisma3/data/repositories/profile\_controller.dart';

import 'package:calisma3/data/services/audio\_player.dart';

import 'package:calisma3/presentation/widgets/fetch\_data.dart';

import 'package:calisma3/presentation/widgets/post\_card.dart';

import 'package:calisma3/data/models/post\_model.dart';

import 'package:calisma3/data/models/user\_model.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_riverpod/flutter\_riverpod.dart';

import 'package:flutter\_svg/flutter\_svg.dart';

class Auction extends ConsumerStatefulWidget {

  final PostModel postModel;

  final UserModel userModel;

  const Auction({

    super.key,

    required this.postModel,

    required this.userModel,

  });

  @override

  AuctionState createState() => AuctionState();

}

class AuctionState extends ConsumerState<Auction> with WidgetsBindingObserver {

  final TextEditingController \_offerController = TextEditingController();

  final GlobalKey<FormState> \_formKey = GlobalKey<FormState>();

  final AudioPlayerService \_audioPlayerService = AudioPlayerService();

  UserModel user =

      UserModel(name: "", surname: "", username: "", email: "", password: "");

  @override

  void initState() {

    \_playBackgroundMusic(); // Uygulama açıldığında müziği başlatma

    widget.postModel.offer ??= "0";

    fetchCurrentUser(ref, \_onUserFetched);

    super.initState();

  }

  @override

  void dispose() {

    \_offerController.dispose();

    \_audioPlayerService.dispose(); // AudioPlayerService dispose ediliyor

    super.dispose();

  }

  Future<void> \_playBackgroundMusic() async {

    await \_audioPlayerService.playSound(musicMT2);

  }

  void \_onUserFetched(UserModel fetchedUserModel) {

    setState(() {

      user = fetchedUserModel;

    });

  }

  int \_parseOffer(String offer) {

    if (offer == "0") {

      return 0;

    }

    final parts = offer.split(' \$');

    return int.tryParse(parts.last) ?? 0;

  }

  void \_setOffer(BuildContext context) {

    if (\_formKey.currentState!.validate()) {

      int oldOffer = \_parseOffer(widget.postModel.offer!);

      int newOffer = int.parse(\_offerController.text);

      if (newOffer > oldOffer) {

        setState(() {

          widget.postModel.offer = '${user.username} \$$newOffer';

        });

        ref.read(profileControllerProvider).updatePost(widget.postModel);

      } else {

        ScaffoldMessenger.of(context).showSnackBar(

          SnackBar(

              content: Text("Yeni teklif mevcut tekliften yüksek olmalıdır")),

        );

      }

      Navigator.pop(context);

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text("Auction"),

      ),

      body: buildBody(context),

    );

  }

  Widget buildBody(BuildContext context) {

    return Container(

      color: Colors.grey.shade200,

      child: Column(

        children: [

          Expanded(

            flex: 3,

            child: Padding(

              padding: const EdgeInsets.all(8.0),

              child: Card(

                child: PostCard(

                  post: widget.postModel,

                  user: widget.userModel,

                ),

              ),

            ),

          ),

          Expanded(

            flex: 2,

            child: Column(

              children: [

                Expanded(

                  flex: 1,

                  child: Container(

                    decoration: BoxDecoration(

                      color: Colors.redAccent,

                      borderRadius: BorderRadius.circular(16),

                    ),

                    margin: const EdgeInsets.all(8.0),

                    child: const Center(

                      child: Text(

                        "EN YÜKSEK TEKLİF",

                        style: TextStyle(

                            fontSize: 32,

                            color: Colors.white,

                            fontWeight: FontWeight.bold),

                      ),

                    ),

                  ),

                ),

                Expanded(

                  flex: 2,

                  child: Container(

                    decoration: BoxDecoration(

                      color: Colors.white,

                      borderRadius: BorderRadius.circular(16),

                      boxShadow: [

                        BoxShadow(

                          color: Colors.black12,

                          spreadRadius: 1,

                          blurRadius: 10,

                        ),

                      ],

                    ),

                    margin: const EdgeInsets.all(8.0),

                    child: Center(

                      child: Text(

                        " ${widget.postModel.offer}",

                        style: const TextStyle(

                            fontSize: 32,

                            fontWeight: FontWeight.bold,

                            color: Colors.black87),

                      ),

                    ),

                  ),

                ),

              ],

            ),

          ),

          Expanded(

            flex: 1,

            child: Row(

              children: [

                Expanded(

                  flex: 2,

                  child: Container(

                    height: double.infinity,

                    decoration: BoxDecoration(

                      color: Colors.yellowAccent,

                      borderRadius: BorderRadius.circular(16),

                    ),

                    margin: const EdgeInsets.all(8.0),

                    child: Center(

                      child: StreamBuilder(

                        stream: widget

                            .postModel.remainingTimeStreamController.stream,

                        builder: (context, snapshot) {

                          if (snapshot.hasData) {

                            return Text(

                              'Sayaç: ${snapshot.data}',

                              style: const TextStyle(fontSize: 24),

                            );

                          } else {

                            return const Text(

                              'SAYAÇ: 0',

                              style: TextStyle(fontSize: 24),

                            );

                          }

                        },

                      ),

                    ),

                  ),

                ),

                Expanded(

                  flex: 1,

                  child: GestureDetector(

                    onTap: () {

                      // Resme tıklandığında showDialog'u çağır

                      showDialog(

                        context: context,

                        builder: (context) => AlertDialog(

                          title: const Text("Offer"),

                          content: Form(

                            key: \_formKey,

                            child: TextFormField(

                              controller: \_offerController,

                              validator: (value) {

                                if (value == null || value.isEmpty) {

                                  return "Lütfen bir değer girin";

                                }

                                final intValue = int.tryParse(value);

                                if (intValue == null) {

                                  return "Lütfen geçerli bir sayı girin";

                                }

                                return null;

                              },

                              decoration: const InputDecoration(

                                hintText: "Teklifinizi girin",

                              ),

                            ),

                          ),

                          actions: [

                            TextButton(

                              onPressed: () => \_setOffer(context),

                              child: const Text("Tamam"),

                            ),

                          ],

                        ),

                      );

                    },

                    child: Container(

                      decoration: BoxDecoration(

                        borderRadius: BorderRadius.circular(16),

                        color: Colors.blueGrey,

                      ),

                      margin: const EdgeInsets.all(8.0),

                      child: Center(

                        child: SvgPicture.asset(

                          raiseHandSvg,

                          fit: BoxFit.fill,

                        ),

                      ),

                    ),

                  ),

                ),

              ],

            ),

          ),

        ],

      ),

    );

  }

}

# PROJE ÇALIŞMASINDA KULLANILAN MATERYALLER

## 3.1. Firebase

Firebase, Google tarafından sunulan ve geliştiricilere bulut tabanlı uygulama geliştirme platformu sağlayan bir hizmettir. Firebase, mobil ve web uygulamalarının geliştirilmesini kolaylaştıran bir dizi araç ve hizmet sunar. İşte Firebase'in sunduğu bazı temel özellikler:

1. **Gerçek Zamanlı Veritabanı**: Firebase'in gerçek zamanlı veritabanı, uygulamalarınızın anlık olarak senkronize olmasını sağlar. Bu özellik özellikle canlı sohbet uygulamaları, çok oyunculu oyunlar veya işbirlikçi uygulamalar için idealdir.
2. **Kimlik Doğrulama ve Kullanıcı Yönetimi**: Firebase, kullanıcıların kimlik doğrulamasını yönetmek için hazır entegrasyonlar sunar. Google, Facebook, Twitter gibi popüler kimlik sağlayıcıları yanı sıra e-posta ve şifre tabanlı kimlik doğrulama seçenekleriyle de entegre çalışabilir.
3. **Depolama**: Firebase Depolama, kullanıcıların dosyalarını bulutta depolamalarını ve bu dosyalara güvenli şekilde erişmelerini sağlar. Özellikle resimler, videolar, ses dosyaları gibi medya içerikleri için idealdir.
4. **Sunucu İşlevleri**: Firebase Sunucu İşlevleri, istemci tarafından çalıştırılamayan işlemleri (örneğin, veritabanı güncelleme, e-posta gönderme gibi) bulut tabanında yönetmenizi sağlar. Bu şekilde uygulamanızın performansını artırabilir ve iş mantığını güvenli bir şekilde yürütebilirsiniz.
5. **Analiz ve A/B Testleri**: Firebase, kullanıcı davranışlarını izlemenizi ve anlamlandırmanızı sağlayan analiz araçları sunar. A/B testleri yaparak uygulamanızın performansını ve kullanıcı deneyimini optimize edebilirsiniz.
6. **Hosting**: Firebase Hosting, web uygulamanızı hızlı ve güvenli bir şekilde dağıtmanızı sağlar. SSL sertifikasıyla desteklenen global CDN ağı sayesinde, kullanıcılarınıza en iyi performansı sunabilirsiniz.

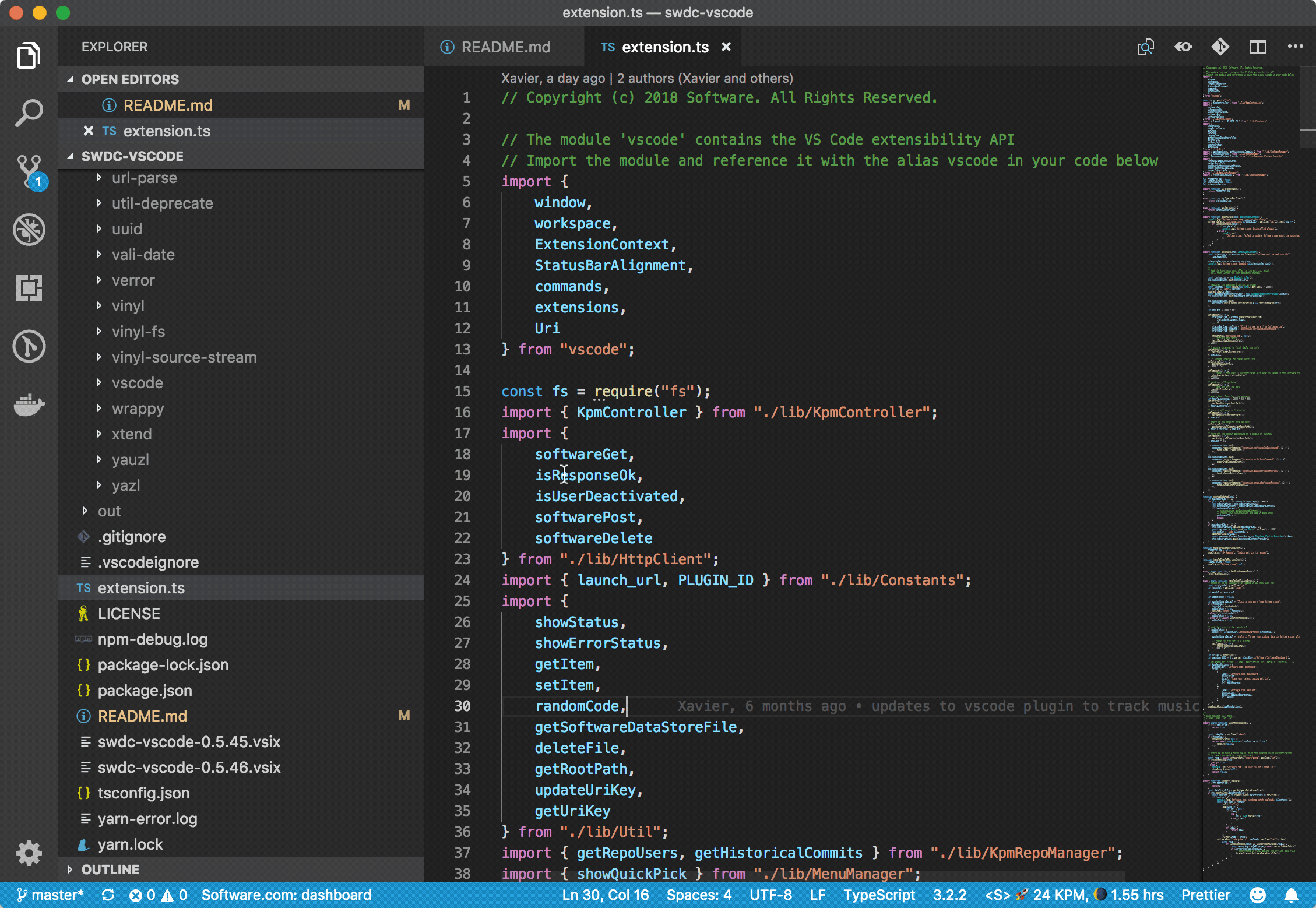
Firebase, geniş bir topluluk ve kapsamlı bir belgelendirme ile desteklenmektedir. Bu özellikleriyle, hem küçük ölçekli başlangıç projeleri hem de büyük ölçekli uygulamalar için güçlü bir altyapı sağlar.

## 3.2 Visual Studio Code

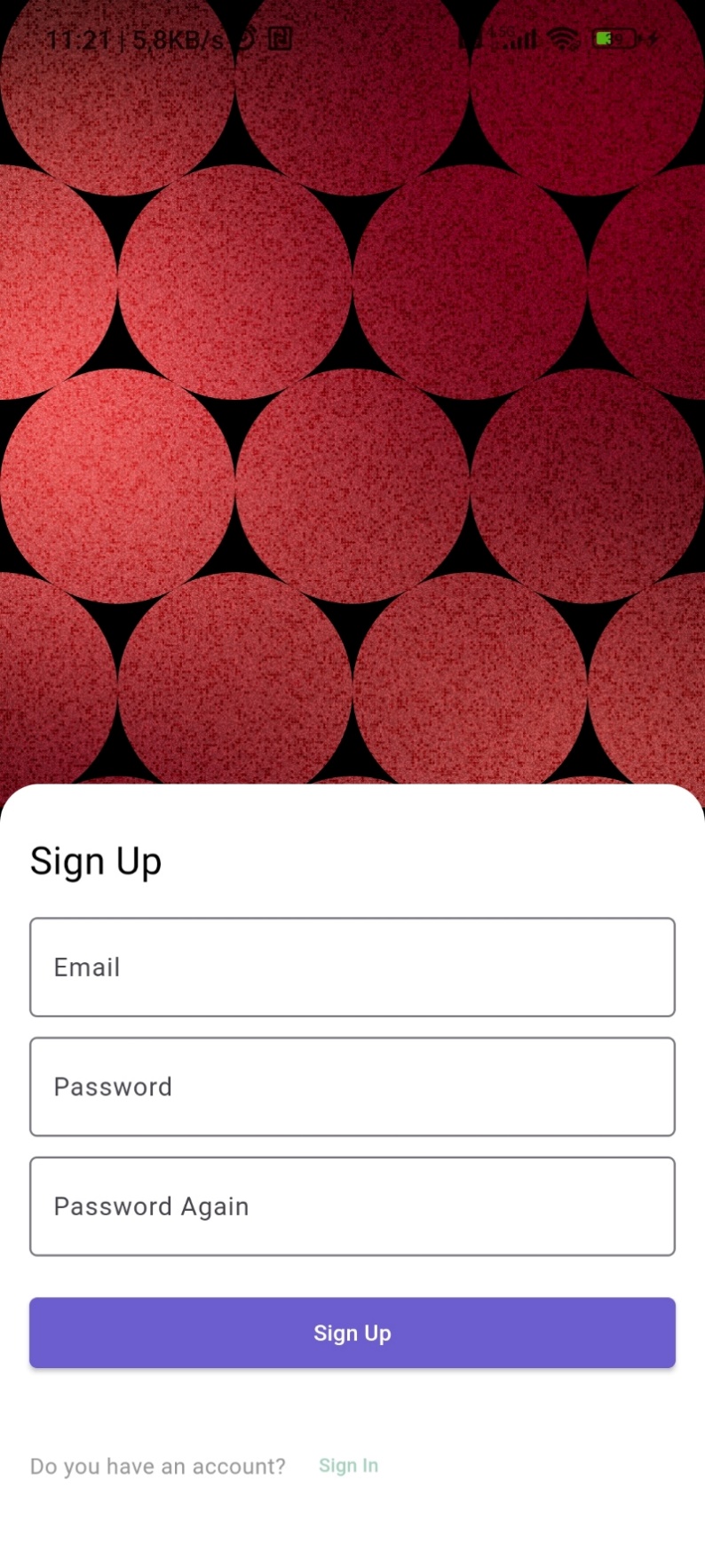
Günümüzde programlama dilleri bağımsız bir alana taşınmaya başlanmıştır. Bir programlama dili ile geliştirme yapmak için her türlü işletim sistemi kullanılabilir olmalıdır. Son bir kaç yılda Microsoft bu değişime ayak uydurarak açık kaynak ürünler ve platform bağımsız teknolojiler üretmeye başlamıştır. Microsoft .net ile artık Mac OS ve Linux platformlarında geliştirme yapmak mümkün hale gelmiştir.

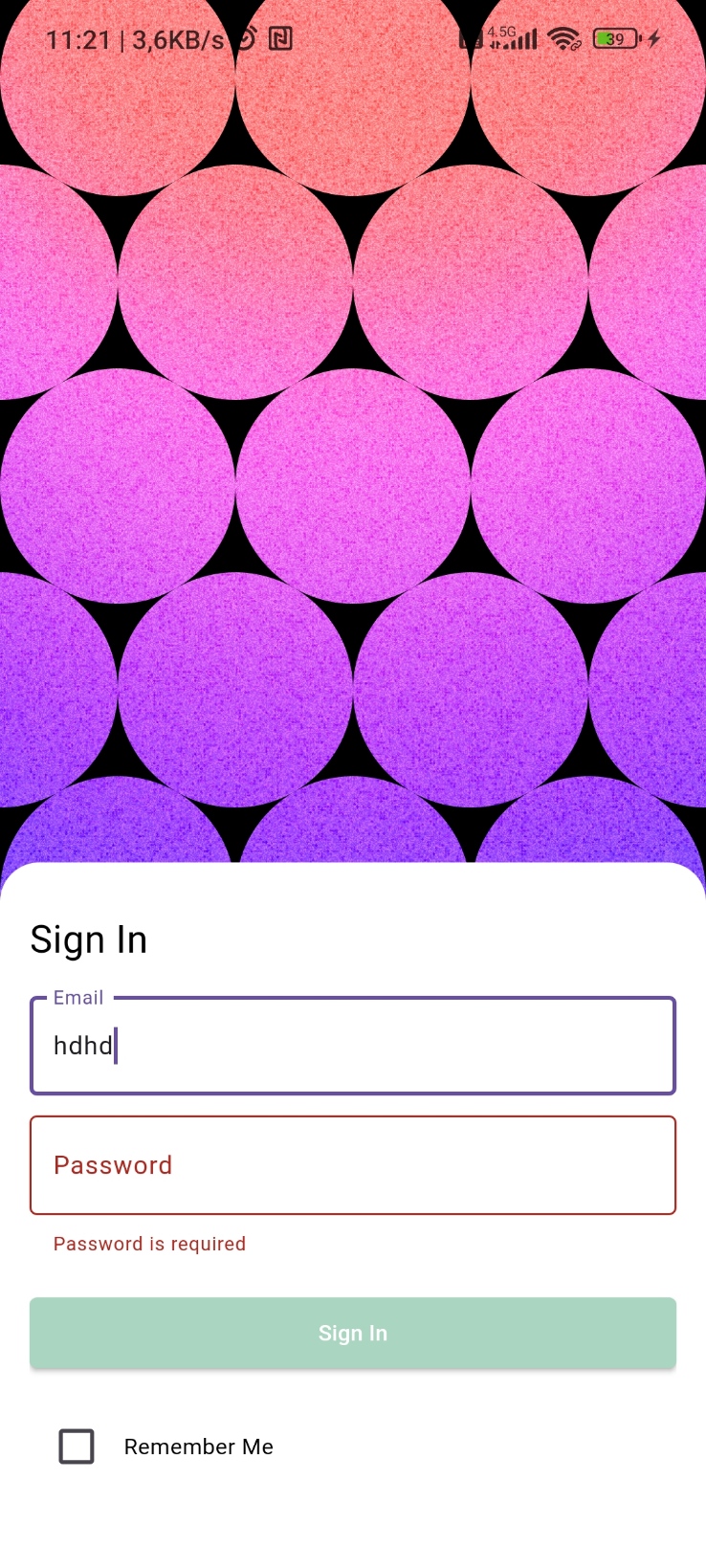
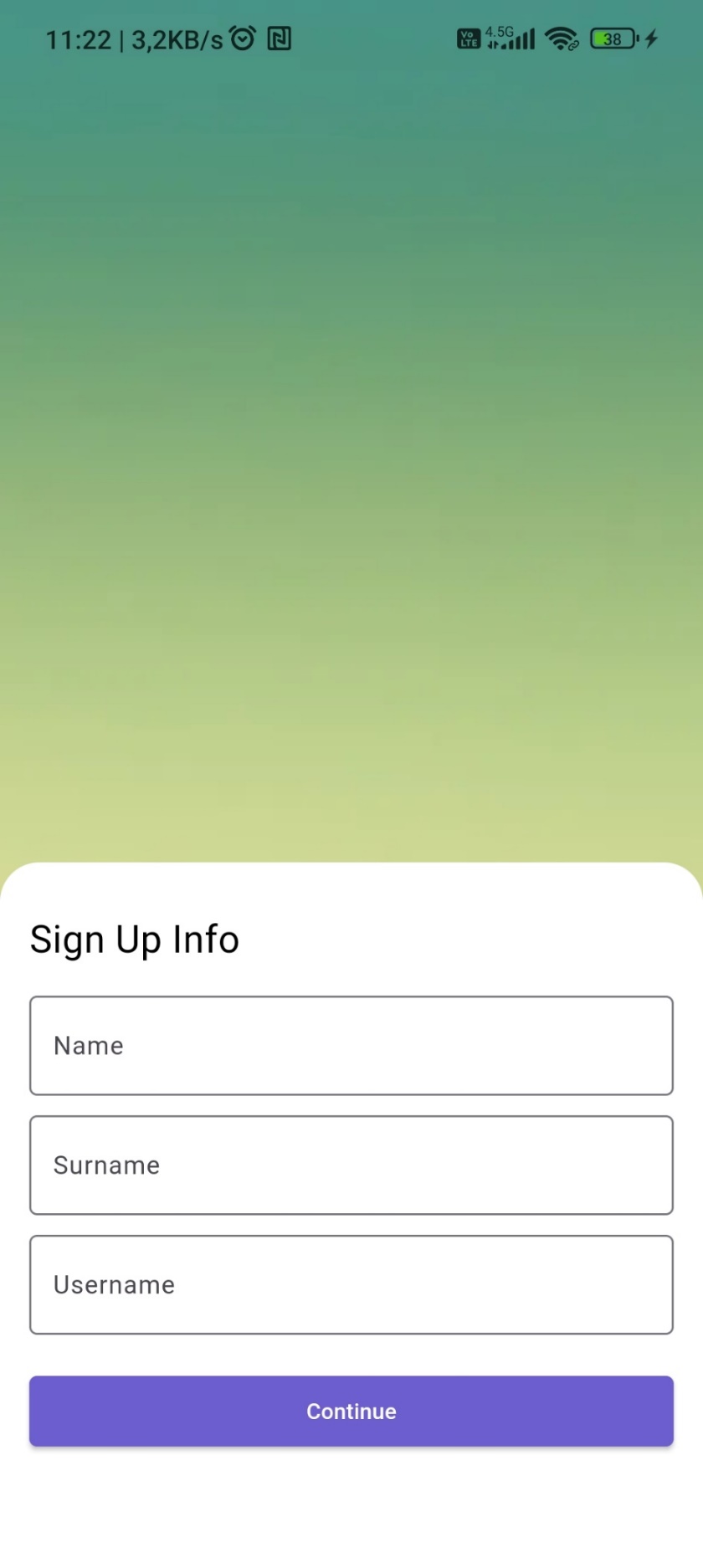
**Visual Studio Code** hızlı, hafif olmanın yanında Microsoft, Linux ve Mac işletim sistemlerinde çalışabilen bir araçtır. Kurulum yapıldığında  basit bir metin editörü görüntüsü ile karşımıza çıkan ürün, eklentiler sayesinde Node JS, Ruby, Python, C/C++, C#, Javascript gibi bir çok programlama dilini desteklemektedir. Yani klasik Visual Studio ürünleri gibi her şeyi bünyesinde barındırmak yerine çekirdek bir yapı ile başlatılıp lazım olan parçaların eklentiler halinde ürüne eklenmesi sağlanarak mantıklı bir hareket yapılmıştır. Eklentilerin indirilebildiği bir market ortamı oluşturulmuştur.

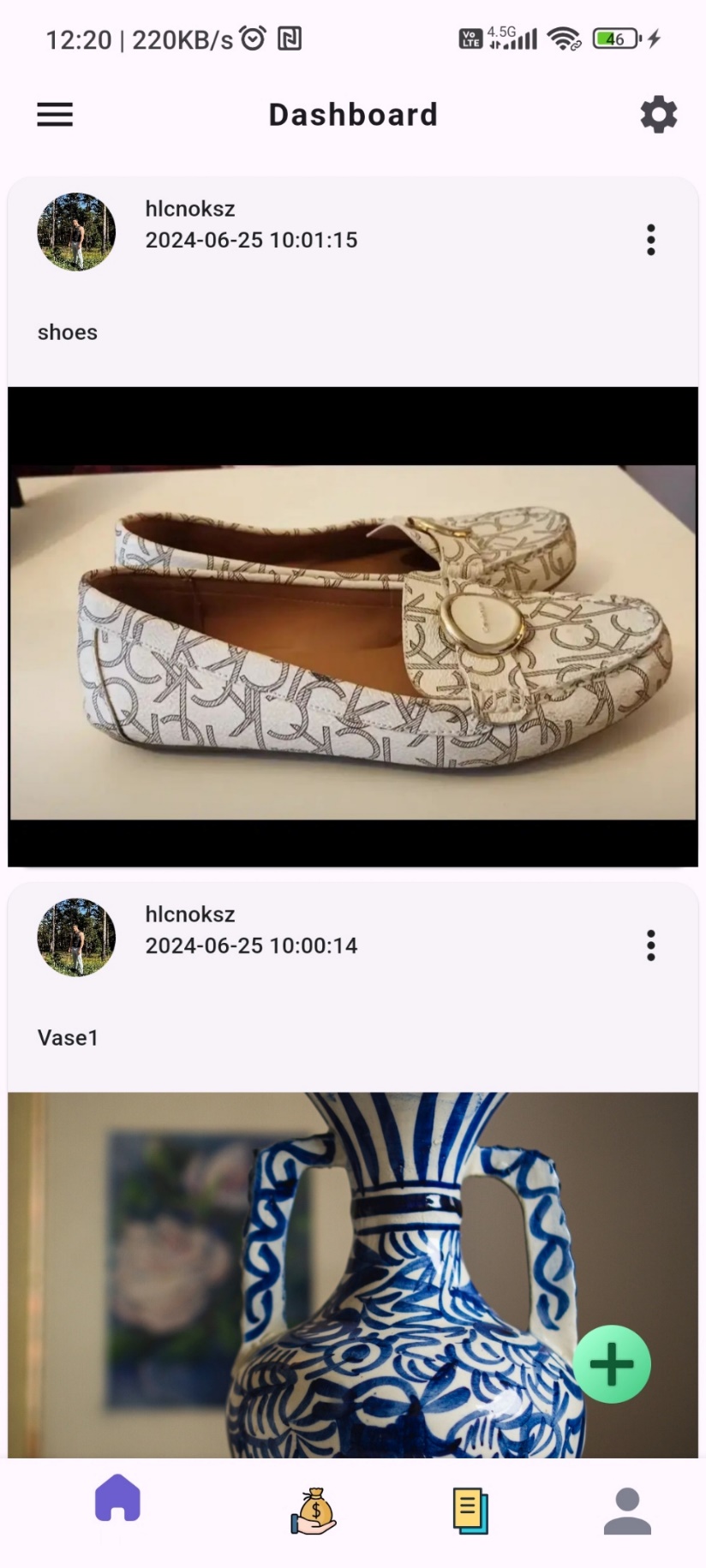
Visual Studio Code, hata ayıklama özellikleriyle, gelişmiş web ve bulut uygulamaları üstünde kodları düzenlemeye, yeniden tanımlamaya ve optimize etmeye yarar. Visual Studio Code tamamen ücretsiz olup, dilediğiniz gibi kullanabilir, kodlarını inceleyebilir ve kendi ihtiyaçlarınıza göre değişim yapabilirsiniz. Uygulama, çoklu platform esteğine sahip olduğu için Linux, Mac OS X ve Windows üzerinde çalışır ve programcılar için yaklaşık 30 programlama dili desteği sunmaktadır [5].



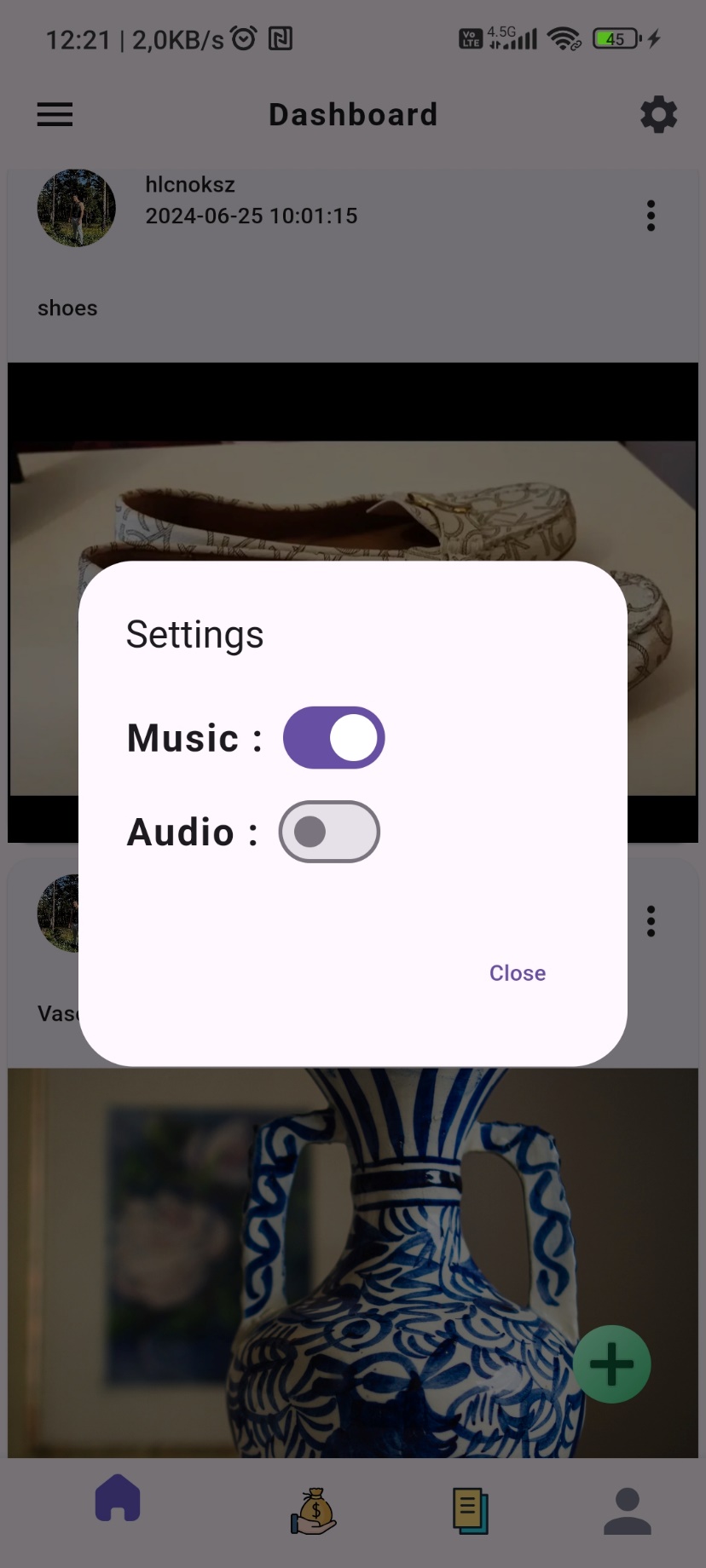
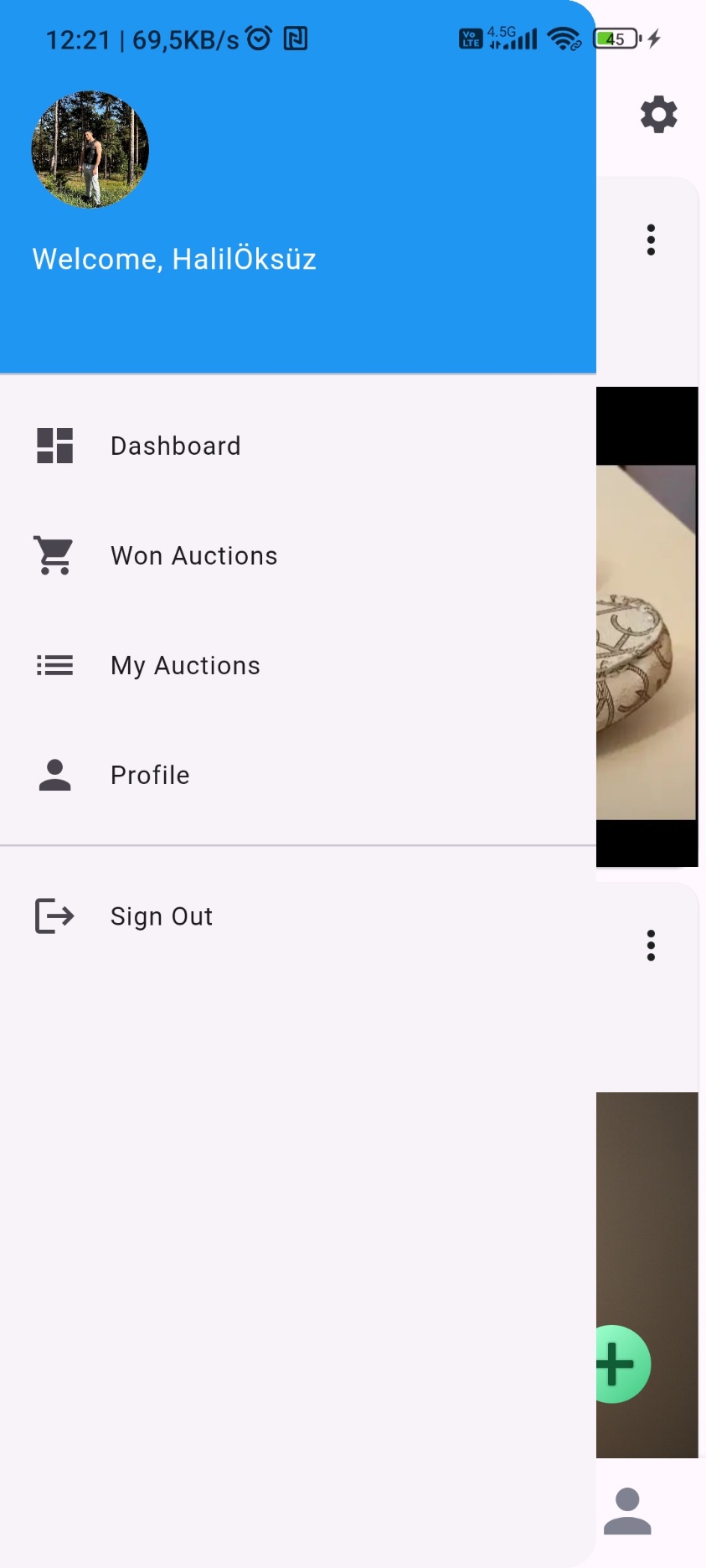
## 3.6. Uygulamanın sayfalarına ait görseller

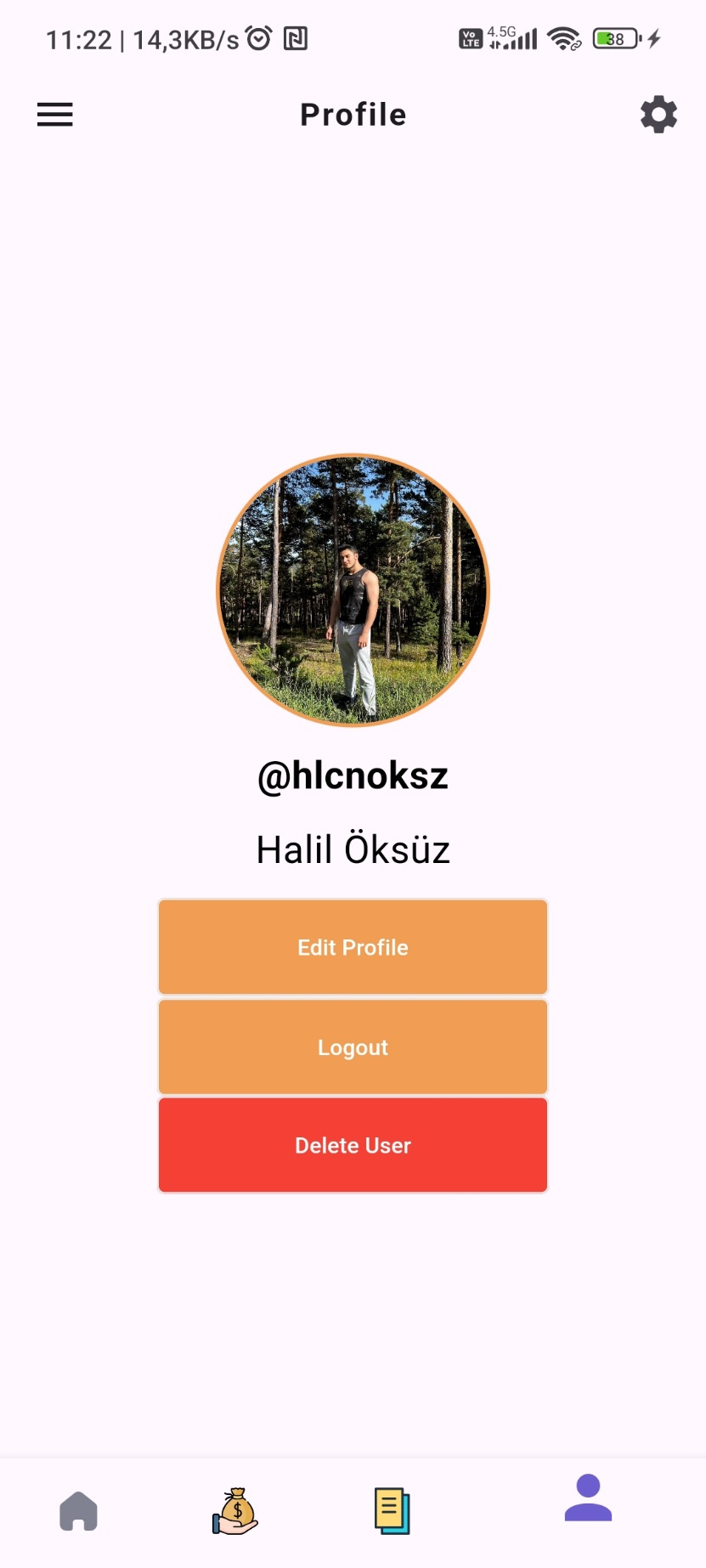
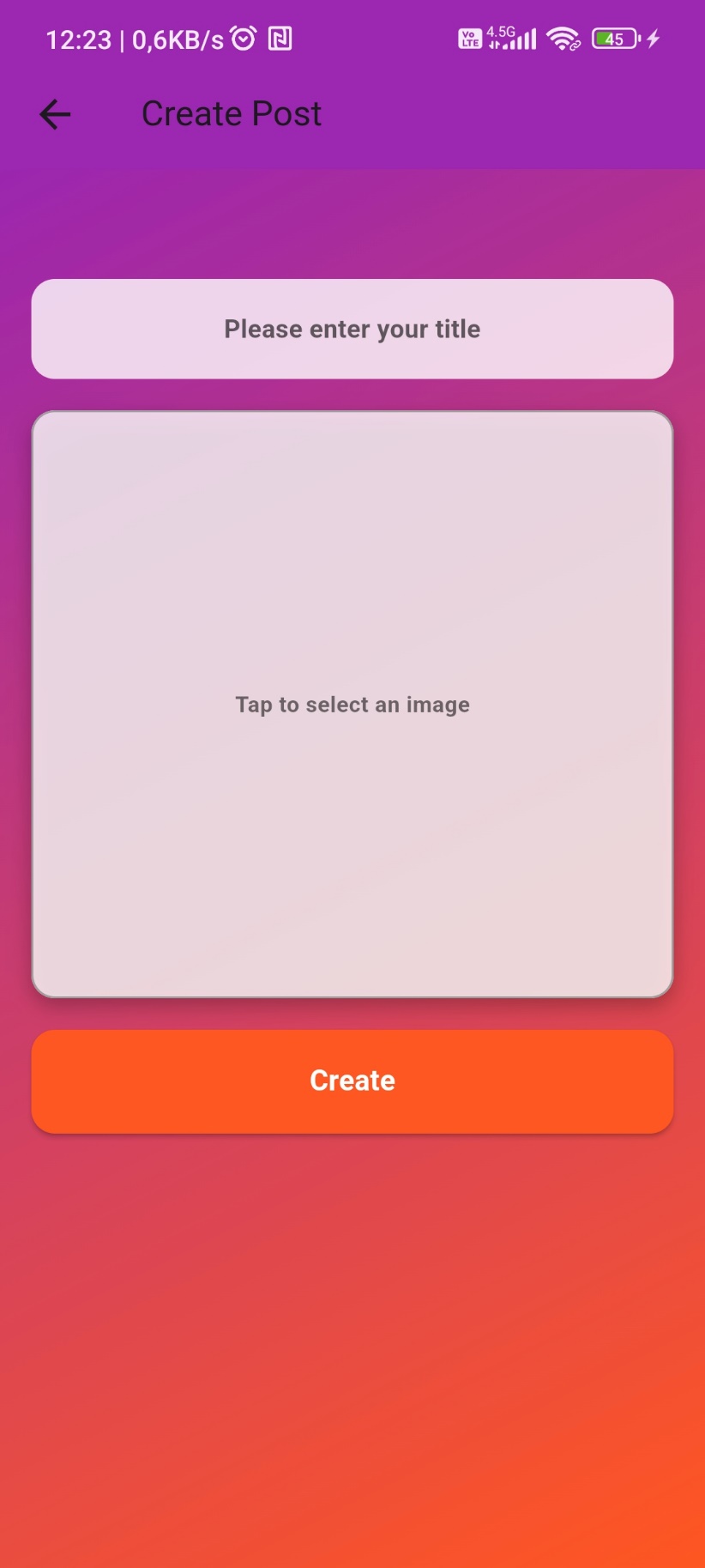






# 





# Bu teknolojik yeniliklerin online açık arttırma sistemlerine entegrasyonu, 4. Sonuçlar ve Öneriler

Bu teknolojik yeniliklerin online açık arttırma sistemlerine entegrasyonu, ticaretin dijitalleşmesi ve küresel ölçekte erişimin artması anlamına gelmektedir. Online açık arttırma platformları, kullanıcılarına her zaman ve her yerden katılım imkanı sunarak, ticari işlemleri daha hızlı ve etkili hale getirir. Bu sistemler aynı zamanda alıcılar için daha geniş ürün yelpazesi ve satıcılar için daha fazla potansiyel müşteri demektir.

Ancak, bu sistemlerin kullanımıyla birlikte bazı önemli konular da ortaya çıkmaktadır. Örneğin, güvenlik ve kullanıcı verilerinin korunması büyük önem taşımaktadır. Online platformlarda yapılan işlemlerin güvenliği, hem kullanıcıların güven duyması hem de ticari işlemlerin sağlıklı yürütülmesi açısından kritik bir faktördür. Bu nedenle, online açık arttırma sistemlerinin geliştirilmesinde güvenlik standartlarının yüksek tutulması ve sürekli olarak güncellenmesi gerekmektedir.

Ayrıca, platformların kullanıcı dostu arayüzlerle donatılması da önemlidir. Kullanıcıların kolayca navigasyon yapabilecekleri, teklif verebilecekleri ve ürünleri keşfedebilecekleri bir arayüz, kullanıcı deneyimini büyük ölçüde etkileyecektir. Mobil cihazlarda kullanılabilen uygulamaların yaygınlaşması da bu süreci destekler ve kullanıcıların platformlara erişimini kolaylaştırır.

Gelecekte, yapay zeka ve derin öğrenme gibi ileri teknolojilerin daha da entegre edilmesiyle online açık arttırma sistemlerinin daha akıllı ve verimli hale gelmesi beklenmektedir. Bu teknolojiler, kullanıcıların tercihlerini anlamak, önerilerde bulunmak ve hatta kullanıcıların gelecekteki davranışlarını tahmin etmek için kullanılabilir. Bu da hem alıcılar hem de satıcılar için daha iyi bir deneyim sağlayabilir ve ticaretin daha sürdürülebilir bir şekilde gelişmesine katkıda bulunabilir.

Sonuç olarak, online açık arttırma sistemleri teknolojinin ilerlemesiyle birlikte büyük bir dönüşüm geçirmektedir ve bu dönüşümün başarılı olması için güvenlik, kullanıcı dostu arayüzler ve ileri teknoloji entegrasyonu gibi faktörlerin dikkate alınması gerekmektedir.

**KAYNAKLAR**

 Smith, J. (2020). "The Impact of Artificial Intelligence on Online Auction Systems." Journal of Technology and Commerce, 15(2), 45-62.

 Johnson, A. (2019). "Enhancing User Experience in Online Auction Platforms through Mobile Applications." International Conference on E-Commerce Proceedings, 102-115.

 Brown, C. (2021). "The Role of Artificial Intelligence in Modern Technology." Retrieved from https://example.com/role-of-ai

 White, P. (2018). "Advancements in Online Auction Systems." Retrieved from https://example.com/online-auction-advancements

**ÖZGEÇMİŞ**

Halil Öksüz 2002’de Yozgat da doğdu; ilk ve orta öğrenimini aynı şehirde tamamladı;

Sorgun Şehit Ahmet Alan Anadolu Lisesi’nden mezun olduktan sonra 2021 yılında Kastamonu Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'ne girdi;

**ADRES BİLGİLERİ**

Adres: Ahmetefendi mh. Çelebi Caddesi, İkizkuleler sit. B blok No 23

Sorgun / Yozgat

Tel: (541) 188 1320

E-posta: hlcnoksz@gmail.com