ONDOKUZ MAYIS ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

MOBİL PROGRAMLAMA PROJE ÖDEVİ

Hasta-Doktor Chat Uygulaması

Berkin AKER

Kadir Emre ÖZER

Danışman

Doktor Öğretim Üyesi İsmai İŞERİ

OCAK, 2022

ÖZET

Sağlık sektörüne yönelik olarak bir tele tıp uygulaması geliştirdik.

Sistemimizin backend(arkayüz) tarafını Node.js ve Mysql veri tabanı, frontend(önyüz) olarakta flutter kullanarak dizayn ettik.

Uygulamamız başlatıldığında bizi login sayfası karşılıyor eğer kullanıcının(hastanın) bir hesabı yok ise Kayıt ol butonu ile Kayıt sayfasına yönlendirilip olduktan sonra Login sayfasına dönüp oluşturmuş olduğu kullanıcı adı ve şifre ile uygulamaya giriş yapabiliyor ve giriş yaptıktan sonra karşısına ana bilim dallarının isimleri hemen altında da doktor arayabileceği bir arama çubuğu, arama çubuğunun altında da doktor listesi yer almakta.

Hasta istediği doktoru aratıp veya direkt olarak listeden seçip seçtiği doktora şikayetini mesaj olarak iletebilmekte.Doktor hastadan gelen mesajı görebilmekte ve o kişiye cevap olarak mesaj yazabilmektedir.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

ÇİZELGELER LİSTESİ

BÖLÜM: I

GİRİŞ

1

AMAÇ

Mobil Cihazlar, Mobil platformları daha iyi anlamak Mobil Sistemler için Kullanıcı Arayüzü Geliştirme ve backend(arkayüz) geliştirme konusunda bilgilerimizi pekiştirmek.Socket kullanarak server ve client arasında iki yönlü iletişimin mesajlaşma sistemi tasarlanarak pratiğe dökülmesi.

BÖLÜM: II

MATERYAL

2

UYGULAMA MİMARİSİ

2.1 *Flutter*

Flutter kullanarak önyüz tasarımımızı ve get post gibi http işlemlerimizi gerçekleştirdik.

2.2 *Node JS*

Node js tarafımızda server'imizi oluşturup get post gibi işlemlere cevaplar dönderdik.Localde server'imizi başlatıp Mysql veri tabanımıza gerekli sorguları yazdık. Mesajlaşma kısmımızı ise Node js içerisinde bulunan paket olan socket paketini kullanarak gerçekleştirdik.

2.3 *Mysql*

BÖLÜM: III

YÖNTEM

3

PROJEMIZDE KULLANIĞIMIZ KODLARIN AÇIKLANMASI

Projemizde doktor ve hastalar arasında bir chat uygulaması gerçekleştirdik. Projemizin backend(arkayüz)'inde ilk olarak npm init yazarak package.json dosyamızı oluşturduk daha sonra body-parser, express, http, mysql, pm2, socket.io socket.io-stream paketlerini npm install mysql veya npm install socket.io şeklinde yazarak projemizin backend'inde kullanacağımız paketleri yükledik ve devamında server.js dosyamızı oluşturarak backend tarafımızda başlangıcımızı yapmış olduk.server.js klasörünü incelersek npm install yazarak yüklediğimiz paketleri require ediyoruz.Projemizde bir chat uygulaması gerçekleştirceğimizden Socket kullanacağız.

Socket ile HTTP'nin farkından kısaca bahsetmek gerekirse

HTTP, bir istek-yanıt protokolüdür, yani bir istemci bir sunucudan bir veri ister ve karşılığında sunucu bu talebe bir yanıt gönderir. Bunun dezavantajı, gönderilen her mesaj için yeni bir istek-yanıt verilmesidir.

Socket browser(client) ve server arasında gerçek zamanlı, çift yönlü ve olay tabanlı iletişimi sağlayan bir kitaplıktır.

server.js'e geri dönersek http modülünü kullanarak bir HTTP sunucusu oluşturduk parametre olarak app ve Socket.IO modülünün bir örneğini oluşturduk argüman olarak server verdik.require işlemlerinden sonra ilk yapcağımız iş veri tabanı bağlantısını sağlamak.

```
9   var mysql = require('mysql');
10  var db = mysql.createConnection({
11      connectionLimit: 100,
12      host: 'localhost',
13      user: 'root',
14      password: '',
15      database: 'mobilfinal0',
16      debug:   false
17  })
18
```

Şekil 1: Veri Tabanı bağlantısını yaptığımız kısım

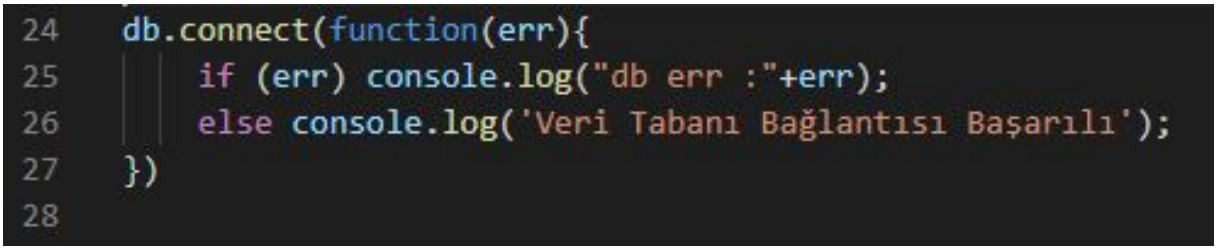
3.1 *Kodların Açıklanması*

Öncelikle server.js içerisindeki kodları açıklamaya geçmeden ön bilgi olarak sıkça kullandığımız metotlardan ve terimlerden bahsedelim.

Projemize dahil ettiğimiz paketlerde yer alan mysql paketinin içinde yer alan createConnection metodu ile bir MySQL bağlantısı oluşturduk. Metot parametre olarak MySQL bağlantısı ile ilgili gerekli bilgilerin bulunduğu bir nesne alır. Bağlantı bilgilerinde host, user(kullanıcı adı),

password(kullanıcı şifre), database(veri tabanı adı) değerlerini kullandık.

Bu Metot geri dönüş değeri olarak bağlantı bilgileri, bağlantı kurma, sorgulama yapma vb. metot ve özelliklere sahip bir nesne döndürür.Bağlantının sağlanabilmesi için döndürülen nesne içerisinde yer alan connect metodu kullanılır.



Şekil 2: connect metodunu kullandığımız kısım

connect metodu içerisinde eğer error alırsak console.log("db err :"+err) yazdığımız için konsolda ilgili erroru yazdırıyoruz eğer ki error almazsak konsola Veri Tabanına Bağlantısı Başarılı yazdırıyoruz ve bağlantımızın başarılı olduğunu teyit etmiş oluyoruz.Sonuç olarak bağlantı bilgileri düzenlenip kodlar çalıştırıldığında konsolda bu iki koşuldan birini göreceğiz.

Node.js ile MySQL sorgulama için query metodu kullanılır.Metodun ilk parametresi SQL sorgusunu, ikinci parametre sorgu sonucuyla işlem yapmak için geri bildirim fonksiyonu alır. Geri bildirim fonksiyonu parametre olarak hata, sonuç ve tablo alanları ile ilgili bilgilerin tutulduğu err, results veya rows parametrelerini alabilir.

Sunucumuza(server'imize) bir istemci(client) bağlandığında connection olayı tetiklenir.Tetiklenen olayları yakalamak için kullanılan on metodunu kullanıyoruz.emit ile'de belirtilen event'e veri'mizi gönderebiliriz.

Genel olarak server tarafımızda sıkça kullancağımız terimleri metotları açıkladık.Şimdi kodların açıklamasına geçelim.

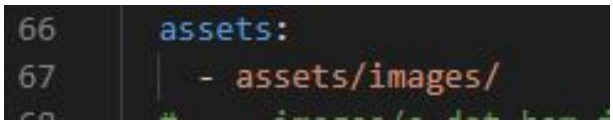
Öncelikle Flutter tarafında kullancağımız paketleri pubspec.yaml içerisinde tanımlıyoruz. pubspec.yaml dosyasından bahsetmemiz gerekirse

```
29 dependencies:
30   flutter:
31     sdk: flutter
32
33
34   # The following adds the Cu
35   # Use with the CupertinoIc
36   cupertino_icons: ^1.0.2
37   socket_io_client: ^1.0.2
38   http: ^0.13.4
39   fluttertoast: ^8.0.8
40   flutter_icons: ^1.1.0
41
```

Şekil 3: pupspec.yaml projeye ettiğimiz paketler

pubspec.yaml paketi projemizin ismi, versiyonu, tanımı bilgilerini içerebilir. Genel olarak projenizin bağlı olduğu bilgileri içerir.Üstteki görselde de görüldüğü gibi socket io client, http, fluttertoast ve flutter icons paketlerini dahil ediyoruz.Paketlerden bahsedersek socket io client socket bağlantımız için. http paketiyle birlikte API uzantılarına istek atmak veya kendi kurduğunuz sunucunun içerisine CRUD işlemleri yapmak için kullanacağız.flutter

toast package ile de ekranda Toast mesajı göstereceğiz.Toast mesajı belirli bir işlem sonucu bilgilendirme yapan kısa süreli mesajlardır.Bu Toast mesajını login ve register kısmında kullanıcının kullanıcı adı ve şifresini girerken yaptığı hataları söylemek için kullandık.

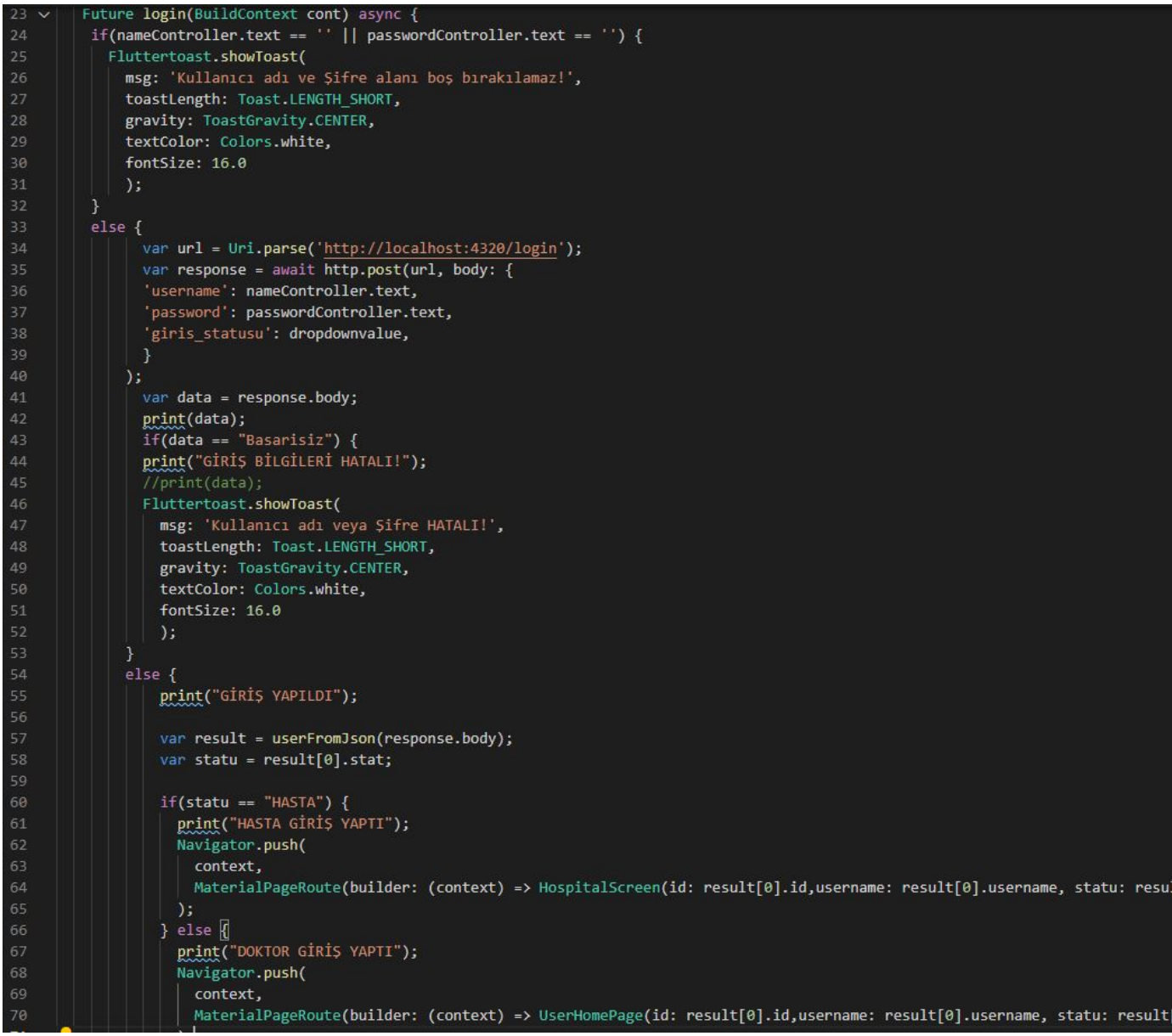


Şekil 4: assets yolunun pupspec.yaml'da tanımlanması

Uygulamamızda kullancağımız resimler için bir assets dosyası oluşturduk ve assets dosyası içinede images klasörü oluşturduk kullanacağımız resimleri bu klasör içerisine attık.Üstteki görselde görüldüğü gibi yine pubspec.yaml dosyasında assets yolunu flutter'a tanımladık.

Paket ve assets tanımlamalarını anattık.Şimdi uygulama ilk açıldığında karşılaşcağımız login sayfasına geçelim.

LoginPage isminde bir Stateful widget oluşturuyoruz.Stateless ve StatefulWidget arasındaki farktan bahsedersek Bir widget stateful veya stateless'dır.Kullanıcı onunla etkileşim kurduğunda değişebilirse bu stateful'dur.Bu yüzden Stateful Widget tercih ediyoruz.Bu değişime örnek olarak Login sayfamızda kullancı aşağı açılan menüden giriş yaparken HASTA veya DOKTOR olduğunu belirtiyor dropdownButton(aşağı açılan menü)'dan seçtiği değer setState içerisinde güncelleniyor ve ekranda seçtiği değer görünüyor.Örneğin dropdownbutton'da HASTA seçiliyken DOKTOR'u seçtiği zaman setState metodu çalışıyor ve artık DOKTOR görünüyor.



```
Future login(BuildContext cont) async {
 if(nameController.text == '' || passwordController.text == '') {
   Fluttertoast.showToast(
     msg: 'Kullanıcı adı ve Şifre alanı boş bırakılamaz!',
     toastLength: Toast.LENGTH_SHORT,
     gravity: ToastGravity.CENTER,
     textColor: Colors.white,
     fontSize: 16.0
     );
 }
 else {
       var url = Uri.parse('http://localhost:4320/login');
       var response = await http.post(url, body: {
       'username': nameController.text,
       'password': passwordController.text,
       'giris_statusu': dropdownvalue,
       }
      );
       var data = response.body;
       print(data);
       if(data == "Basarisiz") {
       print("GİRİŞ BİLGİLERİ HATALI!");
       //print(data);
       Fluttertoast.showToast(
         msg: 'Kullanıcı adı veya Şifre HATALI!',
         toastLength: Toast.LENGTH_SHORT,
         gravity: ToastGravity.CENTER,
         textColor: Colors.white,
         fontSize: 16.0
         );
      }
      else {
          print("GİRİŞ YAPILDI");

          var result = userFromJson(response.body);
          var statu = result[0].stat;

          if(statu == "HASTA") {
            print("HASTA GİRİŞ YAPTI");
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => HospitalScreen(id: result[0].id,username: result[0].username, statu: resu
            );
          } else {
            print("DOKTOR GİRİŞ YAPTI");
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => UserHomePage(id: result[0].id,username: result[0].username, statu: result
```

Şekil 5: loginPage'deki login fonksiyonu

Şekil 5 görselindeki loginPage.dart dosyası içerisindeki login fonksiyonunu açıklarsak.İlk olarak Future tipini tercih etme sebebimiz asenkron kod yazmak için kullandığımız Future veri tipiyle ve async anahtar kelimesiyle birlikte HTTP istekleri, local'de veritabanı işlemleri ve CRUD operasyonları gerçekleştiriyoruz. Burada kurduğumuz if else yapsını açıklarsam en dıştaki ilk if bloğu kullanıcının login ekranında username(kullanıcı adı) veya password(parola)

kısmının boş olup olmadığını kontrol ediyor.Eğer ki iki alandan birisi boş ise önceki sayfalarda bahsettiğimiz toast mesajını kullanarak ekranın alt kısmında bir kaç saniyeliğine Kullanıcı adı ve Şifre alanı boş bırakılamaz diye bildirim yazdırıyoruz.

Eğer iki alanda boş değilse username ve password kısmına girilen ve dropdownbutton'dan seçilen değerleri server'imize http.post metodunu kullanarak http://localhost:4320/login url'sine post ediyoruz.

```
39  app.post("/login", function(req, res) {
40    console.log('post');
41    const username = req.body.username;
42    const password = req.body.password;
43    const giris_statusu = req.body.giris_statusu;
44    if(giris_statusu == "HASTA") {
45      db.query(" SELECT * FROM users WHERE username = '"+username+"' AND password = '"+password+"' ", function(err, re
46        if(err) throw err;
47
48        console.log('Query result: ', result.length);
49        var count = result.length;
50        console.log(result);
51
52        if(count >= 1){
53          //console.log(result);
54          res.send(result);
55        }
56        else {
57          res.send("Basarisiz")
58        }
59      });
60    } else {
61      db.query(" SELECT * FROM doktorlar WHERE username = '"+username+"' AND password = '"+password+"' ", function(err
62        if(err) throw err;
63
64        console.log('Query result: ', result.length);
65        var count = result.length;
66        console.log(result);
67
68        if(count >= 1){
69          //console.log(result);
70          res.send(result);
71        }
72        else {
73          res.send("Basarisiz")
74        }
75      });
76    }
77  });
```

Şekil 6: Server tarafında /login url'sine yapılan post'ların işlendiği kısım

Client(önyüz)'den gelen username password ve giris statusu(HASTA veya DOKTOR) verilerini kullanrak eğer ki giriş yapan kişi bir HASTA ise veri tabanı sorgumuzu users tablosuna yapıyoruz. users tablosu içerisinde hasta tarafından girilen username(kullanıcı adı) ve password(parola) varmı diye bakıyoruz eğer veri tabanında eşleşen bir username password ikilisi var ise veri tabanı bize sayı olarak bir adet result(sonuç) dönderecektir.Bunun kontrolünü count >=1 koşuluyla sağlıyoruz. Count>=1 ise veri tabanında girilen kullanıcı adı ve şifre ikilisi vardır demektir res.send(resut) ile client tarafına result'u yolluyoruz. Eğer veri tabanında eşleşen bir username password yok ise client tarafına res.send("Başarısız") mesajını gönderiyoruz.Şimdi client tarafına üstteki görseldeki login fonksiyonuna tekrar geri dönelim.

Server'da eşleşen username password bulunamaz ise sunucu yani server bize Başarısız mesajını gönderecekti. Eğer başarısız cevabı geldiyse yine Toast mesajını kullnarak ekranın altında bir kaç saniyeliğine kullanıcı adı veya şifre HATALI! yazısını kullanıcıya uyarı olarak yazdırıyoruz.

Server'dan Başarısız cevabı gelmez ise bir eşleşen username password bulunmuş demektir. Server bize girilen username ve passworda ait kişinin bilgilerini client'e result(sonuç) olarak gönderdi.Son olarak Giriş yapan kişinin HASTA veya DOKTOR olmasına göre farklı sayfalara Navigate(yönlendirme) işlemlerini bir if else bloğu içerisinde yapıyoruz. Yönlendirdiğimiz sayfaya giriş yapan kullanıcının id, username, statu ve cinsiyet bilgilerini parametre olarak veriyoruz. Register kısmında ise bu yaptığımız işlemlerden çok daha

basit olarak kişinin kullanıcı adı, şifre ve cinsiyet bilgisini alarak servera post ettikten sonra server'da direkt olarak aşağıdaki görselde görüldüğü üzere veri tabanı içerisine insert ediyoruz.

Şekil 7: Kayıt olan kullanıcının veri tabanına girdiği bilgilerin INSERT edilmesi

Login işlemlerinden sonra kullanıcı HASTA veya DOKTOR olmasına göre farklı sayfalara yönlendirildi.

```
Future callPerson(int id, String userName, String statu) async {

  try{
    final response = await http.get(Uri.parse('http://localhost:4320/fetchAllUsers/${statu}'));
      var result = userFromJson(response.body);
      //print(result[0].username);
      if(mounted)
      setState(() {
        counter = result.length;
        userResult = result;

      });
      return result;
    } catch(e) {
      print(e.toString());
    }
}
```

Şekil 8: callPerson fonksiyonu

Üstteki görsel'deki callPerson fonksiyonunu incelersek server'a giriş yapan kullanıcının statüsünü vererek bir http get işlemi yapıyoruz böylelikle giriş yapan kullanıcı HASTA ise servera HASTA parametresi DOKTOR ise DOKTOR parametresi gidiyor bu fonksiyonu hem DOKTOR'un login yaptıktan sonraki yönlendirildiği sayfada çağırıyoruz hem de HASTA'nın login yaptıktan sonra yönlendirildiği sayfada çağırıyoruz.Böylelikle HASTA giriş yaptığı zaman uygulamaya kayıtlı doktorları client üzerinde görmekte aynı şekilde DOKTOR giriş

yaptığı zaman da kayıtlı HASTA'ları client üzerinde görmekte.

Aşağıdaki görselde görüldüğü üzere eğer giriş yapan kişi HASTA ise veri tabanından doktorlar'ı çekip client'e gönderiyoruz.Giriş yapan kişi DOKTOR ise 'de users tablosundaki kullanıcıları yani HASTA statüsüne sahip olan kullanıcıları çekip client'e gönderiyoruz.

```
79  app.get("/fetchAllUsers/:statu", function(req, res) {
80  console.log(req.params.statu);
81  statu = req.params.statu;
82    if(statu == "HASTA") {
83      db.query("SELECT * FROM doktorlar WHERE statu = 'DOKTOR' ", function(err, result){
84        if(err) throw err;
85          res.send(result);
86          console.log(result)
87      });
88
89    } else {
90      db.query("SELECT * FROM users", function(err, result) {
91        if(err) throw err;
92        res.send(result);
93        console.log(result);
94      });
95    }
96  });
```

Şekil 9: callPerson fonksiyonu içerisinde istek atılan http.get işleminin server'da karşılandığı yer

HASTA login yaptıktan sonra gelen sayfada aynı kullanıcıları çektiğimiz gibi hatta daha basit bir şekilde hiç servera parametre göndermeye gerek kalmadan anabilimdalları tablosundan anabilim dallarını çekiyoruz ve client tarafında veri tabanından çekmiş olduğumuz ana bilim tablosu isimlerini ilgili görsellerin altında ekrana yazdırıyoruz.Son olarak bu oluşturmuş olduğumuz fonksiyonları initState içerisinde çağırıyoruz ki uygulama açıldığında

veriler çekilsin ve ekranda yerlerini alsın.

```
@override
void initState() {
    try{
    socket = IO.io('http://10.0.2.2:4320', <String, dynamic>{
        'transports': ['websocket'],
        'autoConnect': false,
      });

    socket.connect();

    socket.on("connect", (data) {
      print(socket.id);
      socket.emit('onlineHasta', widget.id);
    });

    socket.on('doktorOnline', (data){
      print(data);
      doktorOnlineId = data;
      print("doktorOnlineId ytazdiriliyor");

    });
  }
  catch (e) {
    print(e);
  }

```

Şekil 10: socket bağlantısı

Öncelikle HASTA login olduktan sonra gelen ekranı açıklayalım. try catch blokları içerisinde socket bağlantımızı gerçekleştiriyoruz socket bağlantısını yaparken 10.0.0.2 yani localhost ve iki noktandan sonra da port'umuzu 4320 olarak yazıyoruz çünkü serveri oluştururken 4320 olarak seçmiştik. socket.connect() ile socket bağantısını gerçekleştiriyoruz eğerki bir kullanıcı socket'e bağlanırsa socket.on("connect", (data)) ile bu olayı yakalayıp Debug konsolunda socket.id'sini yazdırıyoruz.

```
102     super.initState();
103     callPerson(widget.id, widget.username, widget.statu);
104     fetchAnaBilimDallari();
105     super.initState();
106     init();
107   }
108
```

Şekil 11: socket bağlantısı

Devamında üstteki initState içerisinde önceki sayfalarda açıkladığımız bazı gerekli verileri çekmemizi sağlayan fonksiyonları çağırıyoruz.

HASTA statüsüne sahip kullanıcıların login yaptıktan sonra karşılaştığı ekran hospital screen body.dart dosyasındadır.Build kısmına geçelim

```
114   @override
115   Widget build(BuildContext context) {
116     Size size = MediaQuery.of(context).size;
117     var array2 = [];
118     return userResult == null ? Center(child: CircularProgressIndicator()) : HospitalBackground(
119       child: SingleChildScrollView(
120         child: Column(
121           crossAxisAlignment: CrossAxisAlignment.start,
122           mainAxisAlignment: MainAxisAlignment.start,
123           children: [
124
125             Padding(
126               padding: EdgeInsets.only(left:15,right: 25),
127               child: AppBar(
128                 backgroundColor: Colors.transparent,
129                 elevation: 0,
130                 leading: Icon(Icons.menu,color: Colors.black,size: 25,),
131                 actions: [
132                   GestureDetector(
133                     //onDoubleTap: (){Navigator.push(context,MaterialPageRoute(builder: (context){return ProfileScreen(userId: widget.userId)
134                     child: Container(
135                       child: CircleAvatar(
136                           child: widget.cinsiyet == "K" ? Image.asset("assets/images/hastawoman.png",alignment: Alignment.center,) : Image.as
137                         ), // Image.asset
138                       ), // CircleAvatar
139                       width: 75,
140                       height: 75,
141                       decoration: BoxDecoration(
142                         shape: BoxShape.circle,
143                         gradient: const LinearGradient(
144                           colors: [Colors.white10,Colors.white],
145                           stops: [0,1],
146                         ), // LinearGradient
147                       ), // BoxDecoration
148                     ), // Container
149                   ), // GestureDetector
150                 ],
151               ), // AppBar
152             ), // Padding
```

Şekil 12: hospital screen body.dart dosyasının build fonksiyonu

return userResut== null ? kısmı için oluşturduğumuz callPerson fonksiyonunu hatırlarsak userResult içerisine veri tabanından çekmiş olduğumuz Doktorların bilgilerini atamıştık aynı

şekilde callPerson fonksiyonu içerisinde counter değişkeni içerisine de server'dan gelen result'un(result: server'in bize cevap olarak gönderdiği veri tabanında bulunan doktorların bilgileri) result.length ile uzunluğunu yani kaç adet doktor bulunduğu değerini counter değişkenine atıyorduk.Uygulama açılırken milisaniyelikte olsa verileri çekme anı için bekleneceğinden eğer ki userResult null ise ekranda CircularProgressIndicator() ile yüklenme simgesi koyuyoruz ve userResult'a veri geldiği an userResult== null false olacağından widget'ımızı return ediyoruz.

İlk olarak kullanıcının cinsiyeti K (kadın) veya E (erkek) olmasına bağlı olarak Kadın kullanıcı ikonu veya erkek kullanıcı ikonlarını CircleAvatar widget'ı ile kullanıcının profil fotoğrafı olarak koyuyoruz.

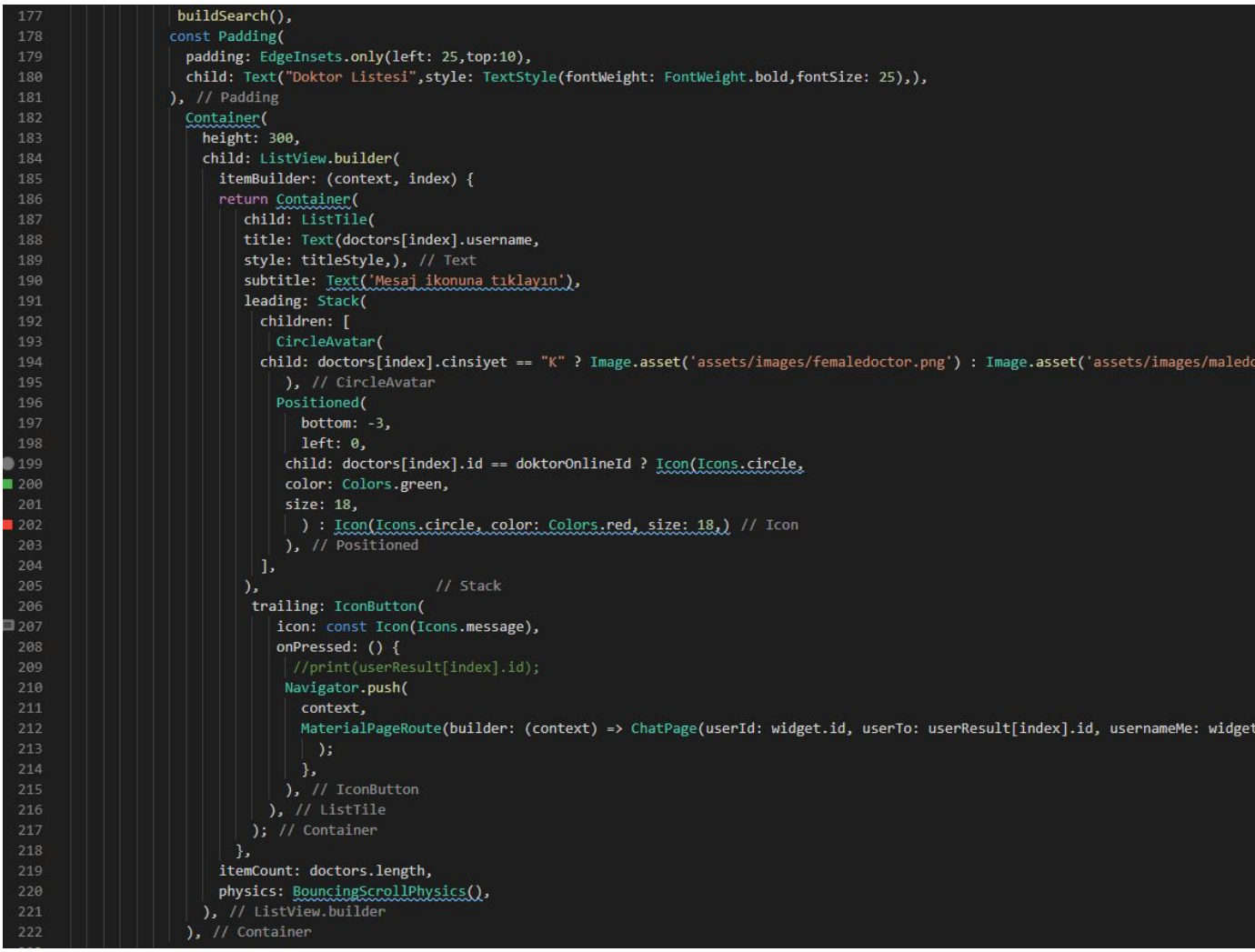
```
153                const Padding(
154                  padding: EdgeInsets.only(left: 25,top:15),
155                    child: Text("Kategoriler",style: TextStyle(fontWeight: FontWeight.bold,fontSize: 25),),
156                ), // Padding
157                Container(
158                  margin: EdgeInsets.only(top:10),
159                  width: size.width,
160                  height: 150,
161                  child: ListView.builder(
162                    itemCount: counter2,
163                    itemBuilder: (context, index) {
164
165                      return anabilimDaliResult == null ? Center(child: CircularProgressIndicator()) : Row(
166                          children: [
167                            MyContainer(ImgName: "${anabilimDaliResult[index].anaBilimDali.replaceAll(' ', '')}.png",ImgDesc: an
168                          ],
169                        ); // Row
170                    },
171                    physics: BouncingScrollPhysics(),
172                    scrollDirection: Axis.horizontal,
173
174
175                  ), // ListView.builder
176                ), // Container
177                 buildSearch(),
```

Şekil 13: hospital screen body.dart(HASTA'nın login olduktan sonra karşılaştığı ekran) dosyasının devamı

Üstteki görselde görüldüğü üzere Ana bilim dallarını bir Listview.builde yapısı kullanarak listeliyoruz.itemCount kısmı fetchAnaBilimDallari fonksiyonu içerisinde serverden gelen result'un length'ine göre yani anabilimdali sayısını counter2 değişkenine atamıştık burada da

o counter 2 değişkenini itemCount'a atıyoruz böylelikle her yeni ana bilim dalı eklendiğinde bu counter güncellenecek ve ekranda o sayıda içerik göreceğiz yapımızı dinamikleştirdik diyebiliriz. anabilimDaliResult== null ? kısmı yine verileri çekerken arada oluşan milisaniyelik zaman dilimi için ekranda yüklenme işareti gösteriyoruz ve veriler çekildikten sonra anabilimDaliResult== null false olacağından Row widget'ımızı return ediyoruz. Row widget'ının children property'si ile bizim özel olarak oluşturmuş olduğumuz MyContainer widget'ını kullanarak çektiğimiz anabilimDaliResult datalarını scrollDirection property'sinde Axis.horizontal kullanarak ekranda yatay olarak koyuyoruz.Sonda kullandığımız buildSearch fonksiyonu doktor arama işlemi için kullandığımız bir fonksiyon ilerki sayfalarda açıklanacaktır.

Aşağıdaki(bir sonraki sayfa) görselde(şekil 14) anabilimdallarını çekerken yaptığımız gibi ListView.builder kullanarak veri tabanından çektiğimiz doktorları ekrana dinamik(itemcount veri tabanına her yeni doktor eklendiğinde artıyor bu yüzden dinamik) olarak koyuyoruz.Doktorları default olarak vertical yani dikey olarak listeliyoruz.itemBuilder içerisinde doctors[index].cinsiyet ile o anki indexteki doktorun cinsiyetine göre kadın doktor ikonu veya erkek doktor ikonu koyuyoruz. Positioned widget'ı ile ikonların sol alt köşelerine gerekli koşul sorgulaması neticesinde yeşil(online) veya kırmızı(offline) ikonu koyuyoruz.ListTile widget'ının trailing property'sinde IconButton widget'ını kullanarak Icons.message ikonunu ListTile'nin sağ kısmına koyuyoruz ve bu ikona bir onpress özelliği veriyoruz. Kullanıcı ikona tıkladığı zaman tıkladığı ilgili doktor ile mesajlaşabileceği chat ekranına kendi id'si doktorun id'si kendi username'i ve statü parametreleriyle yönlendiriliyor. itemCount olarak veri tabanındaki doktor sayısını veren doctors.length'i kullanıyoruz böylelikle veri tabanına yeni doktor



```
177             buildSearch(),
178           const Padding(
179             padding: EdgeInsets.only(left: 25,top:10),
180             child: Text("Doktor Listesi",style: TextStyle(fontWeight: FontWeight.bold,fontSize: 25),),
181           ), // Padding
182             Container(
183               height: 300,
184               child: ListView.builder(
185                 itemBuilder: (context, index) {
186                 return Container(
187                    child: ListTile(
188                    title: Text(doctors[index].username,
189                    style: titleStyle,), // Text
190                    subtitle: Text('Mesaj ikonuna tıklayın'),
191                    leading: Stack(
192                      children: [
193                       CircleAvatar(
194                      child: doctors[index].cinsiyet == "K" ? Image.asset('assets/images/femaledoctor.png') : Image.asset('assets/images/maledo
195                        ), // CircleAvatar
196                       Positioned(
197                          bottom: -3,
198                          left: 0,
199                        child: doctors[index].id == doktorOnlineId ? Icon(Icons.circle,
200                        color: Colors.green,
201                        size: 18,
202                          ) : Icon(Icons.circle, color: Colors.red, size: 18,) // Icon
203                        ), // Positioned
204                      ],
205                    ),                    // Stack
206                     trailing: IconButton(
207                        icon: const Icon(Icons.message),
208                        onPressed: () {
209                         //print(userResult[index].id);
210                         Navigator.push(
211                          context,
212                          MaterialPageRoute(builder: (context) => ChatPage(userId: widget.id, userTo: userResult[index].id, usernameMe: widget
213                            );
214                          },
215                         ), // IconButton
216                       ), // ListTile
217                     ); // Container
218                   },
219                 itemCount: doctors.length,
220                 physics: BouncingScrollPhysics(),
221               ), // ListView.builder
222             ), // Container
```

Şekil 14: Doktorlar'ın ListView.builder ile listelenmesi

eklenince otomatik olarak listemize yer almakta.Genel olarak toparlarsak bu sayfada yaptığımız cinsiyete göre Hastanın profil ikonunun koyulması orta bölümde anabilim dallarının veritabanından çekilip yerleştirilmesi. search bar koyulup searh bar'ın altında'da doktorların veri tabanından çekilip yine bazı koşullara göre resimlernin online offline durumlarının kararlaştırılıp listelenmesi ve son olarak ilgili doktorun yanındaki chat ikonuna tıklanılığında tıklayan hasta ile tıklanılan doktor arasındaki mesajlaşma ekranına gidilmesi.

```
Widget build(BuildContext context) {
    //print(widget.id);
    //print(widget.username);
  return userResult == null ? Center(child: CircularProgressIndicator()) : Scaffold(
     appBar: AppBar(
       title: widget.statu == 'HASTA' ? Text('Doktor Listesi') : Text('Hasta Listesi'),
       backgroundColor: Colors.teal,
       ), // AppBar
       body: Column(
          children: [
            buildSearch(),
             Expanded(
                child: ListView.builder(
               physics: const BouncingScrollPhysics(),
               itemCount: users.length,
               itemBuilder: (context, index) {
                 return ListTile(
                   title: Text(users[index].username,
                   style: titleStyle,), // Text
                   subtitle: Text('Mesaj ikonuna tıklayın'),
                   leading: Stack(
                     children: [
                      CircleAvatar(
                       child: users[index].cinsiyet == "K" ? Image.asset('assets/images/hastawoman.png') : Image.asset('assets/images/hastamal
                       backgroundColor: Colors.black54,
                      ), // CircleAvatar
                      Positioned( bottom: -3,
                          left: 0,
                        child: users[index].id == hastaOnlineId ? Icon(Icons.circle,
                        color: Colors.green,
                        size: 18,
                          ) : Icon(Icons.circle, color: Colors.red, size: 18,)), // Icon // Positioned
                    ],

                    ), // Stack
                     trailing: IconButton(
                       icon: const Icon(Icons.message),
                       onPressed: () {
                         //print(userResult[index].id);
                        Navigator.push(
                          context,
                          MaterialPageRoute(builder: (context) => ChatPage(userId: widget.id, userTo: userResult[index].id, usernameMe: widget
                        );
                       },
                    ), // IconButton
                  ); // ListTile
                }), // ListView.builder
              ), // Expanded
```

Şekil 15

Üstteki görsele de bakarsak Doktor tarafından giriş yapılıncada en üstte bir searchbar(arama çubuğu) devamında aynı mantıkla veritabanından doktorlar değil hastalar çekildi bir ListView.builder ile Kadın veya Erkek olmasına göre ikonlarını alıp online offline durumları koşul ile kararlaştırılıp ekrana koyuldu.ListTile'nin trailing property'sinde IconButton Widget'ı kullandık ve en sağa mesaj ikonu koyduk bu ikonun onPress özelliğine tıklanıldığında ise aynı önceki kısımlarda anlattığımız hastaların tıkladığı doktorla mesajlaşmaya başlayacağı ekrana

yönlenmesi gibi burada da doktor ilgili hastanın mesaj ikonuna basıp o hastadan gelen mesajları görebileceği chat ekranına kendi id'si tıkladığı hastanın id'si kendi username'i, statüsü ile yönlendirilmekte ve bu chat ekranından ilgili hastadan gelen mesaja cevap verebilmekte.(NOT: Kullanıcıların mesajlaşmak için yönlendirildiği ChatPage widget'i detaylı olarak bir sonraki paragrafta açıklanmaya başlıyor.)

ChatPage widget'ımızı(Şekil 16) stateful yapmamızın sebebi ekranın sürekli güncellenmesi(yeni mesaj geldiğinde eski mesajların yukarıya kayması vs.) yani kullanıcı ile bir etkileşim olduğundan statefulWidget olarak tercih ediyoruz.Görselde(şekil 16) de görüldüğü gibi socket bağlantımızı yaptıktan sonra ilk yaptığımız şey eski mesajlaşmaları veri tabanından çekmek olacak bunun için 'fetchOldMesagges' eventine UserChat Modelini de kullanarak socket.emit ediyoruz. Server tarafında fetchOldMessagesi bekleyen client.on('fetchOldMessages' içeriği şekil 17'de verilmiştir.)server.js tarafında boş bir array oluşturuyoruz ve data.statu yani veritabanındaki mesajın statüsüne göre HASTA ise mesajı gönderen kişi messages(mesajlar) tablosundan eski konuşulan mesajları çekerken kendi user id 'miz userTo yani karşıdaki kişinin id'si ve statü'sü HASTA olan mesajları çekiyoruz veya karşıdan gelen mesajlar içinde tam tersi olarak userid userto yani karşıdaki kişinin idsi ve userto user id yani kendi id'miz statü olarakta DOKTOR statüsü olan mesajları çekiyoruz böylelikle örneğin id'si 1 olan hasta ile id'si 2 olan doktor arasındaki mesajlar için ilk sorguda 1den2 ye hasta statüsü + 2den1 e doktor statüsü olan mesajları çekmiş oluyoruz.(NOT: UYGULAMAYA GİRİŞ YAPAN KİŞİNİN STATÜSÜ HASTA İSE BU GEÇERLİ.)Eğer uygulamaya giriş yapan kişinin statüsü DOKTOR ise else bloğuna gireceğiz ve bu sefer sorguda kendi idmizden userto ya

```
class _ChatPageState extends State<ChatPage> {

  final TextEditingController _messageController = TextEditingController();
  final ScrollController _scrollController = ScrollController();
  final List<UserChat> _messages = [];

   void setStateIfMounted(f) {
    if (mounted) setState(f);
  }

  late IO.Socket socket;

  @override
  void initState() {
     try{
      print("test");
      socket = IO.io('http://10.0.2.2:4320', <String, dynamic>{
          'transports': ['websocket'],
          'autoConnect': false,
        });

      socket.connect();

      socket.emit('fetchOldMessages', UserChat(
                                        userId: widget.userId,
                                        userTo: widget.userTo,
                                        message: '',
                                        statu: widget.statu,
                                        //widget.username
                                       ).toJson());

      socket.on("fetchMessages", (data) {
          var c = data.length;
          print(c);

          for(int i = 0; i < c; i++){
          var oldMessages = UserChat.fromJson(data[i]);
          print(oldMessages.message);
          setStateIfMounted(() {
            _messages.add(oldMessages);
          });
          //print(oldMessages.message);
          }
        });
```

Şekil 16: ChatPage extends StatefulWidget

giden sorguda statü dokto veya karşıdan gelen mesajlar içinde statü hasta olmalı çünkü doktor olarak giriş yaptıysak gelen mesajlar hastalardan gelecek. Böylelikle doktorlarında her bir

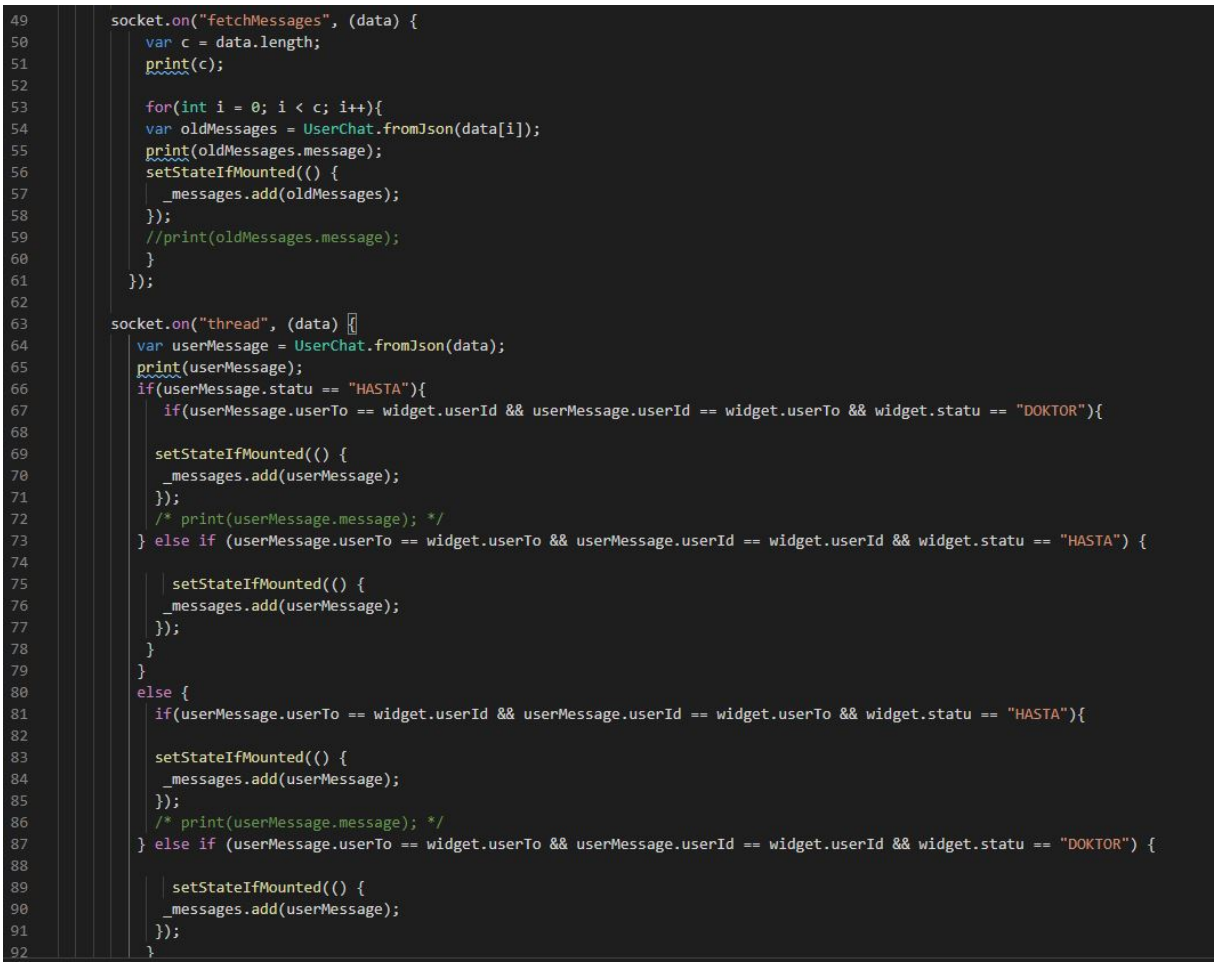
hastasıyla olan bire bir konuşmalarını çekmiş olduk.Çektiğimiz her bir mesajı(row) array içerisine push ediyoruz ve bu array'i fetchMessages event'ine emit ediyoruz.

```
client.on('fetchOldMessages', function(data) {
  console.log("test");
  var array = [];
  console.log(data);
  if(data.statu == "HASTA") {
    db.query("SELECT user_id, user_to, message, statu FROM `messages` WHERE (user_id='"+data.user_id+"' AND user_to='"+data.user_to+"' AND statu='HASTA') OR (user_
      if(err) throw err;
      for(var i = 0; i < rows.length; i++) {
          var row = rows[i];
          array.push(row);
          console.log(row);
      }
      console.log(array);
      client.emit("fetchMessages", array);
      client.emit("sonMesaj", array[rows.length - 1]);
    });
  } else {
    db.query("SELECT user_id, user_to, message, statu FROM `messages` WHERE (user_id='"+data.user_id+"' AND user_to='"+data.user_to+"' AND statu='DOKTOR') OR (user
      if(err) throw err;
      for(var i = 0; i < rows.length; i++) {
          var row = rows[i];
          array.push(row);
      }
      console.log(array);
      client.emit("fetchMessages", array);
    });
  }

})
```

Şekil 17: fetchOldMessages server tarafı(server.js)

Aşağıdaki görselde(şekil 18) önceki sayfada anlattığımız fetchMessages eventini dinliyoruz serverden gelen eski mesajlaşmaları UserChat modeli içerisinde bulundan fromJson metodu her bir data'yı ile json'dan UserChat tipine çeviriyoruz. Ve oluşturmuş olduğumuz messages listesine ekliyoruz.Böylelikle örnek vermek gerekise artık HASTA statüsüne sahip id'si 3 olan kişi ile DOKTOR statüsüne sahip id'si 5 olan kişi arasındaki konuşmalar kendi aralarında çekildi yani herkes'in mesajları 1e1 olarak çekildi ve mesajlar arayüze eklendi.

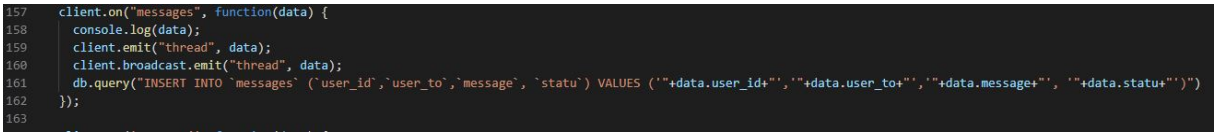
şekil18'e devam edersek thread event'i dinlenmekte bu event message eventine emit yapıldığı zaman çalışmakta öncelikle message eventine yapılan emit kısmına bakalım.Şekil 19'da görüldüğü üzere kullanıcı bir mesaj girdiğinde trim metodu ile trimlenip(boşlukların kaldırılması) messages event'ine(server'deki) gerekli parametreler(UserChat modelini kullanarak)



```
socket.on("fetchMessages", (data) {
    var c = data.length;
    print(c);

    for(int i = 0; i < c; i++){
    var oldMessages = UserChat.fromJson(data[i]);
    print(oldMessages.message);
    setStateIfMounted(() {
      _messages.add(oldMessages);
    });
    //print(oldMessages.message);
    }
  });

socket.on("thread", (data) {
  var userMessage = UserChat.fromJson(data);
  print(userMessage);
  if(userMessage.statu == "HASTA"){
     if(userMessage.userTo == widget.userId && userMessage.userId == widget.userTo && widget.statu == "DOKTOR"){

    setStateIfMounted(() {
     _messages.add(userMessage);
    });
    /* print(userMessage.message); */
  } else if (userMessage.userTo == widget.userTo && userMessage.userId == widget.userId && widget.statu == "HASTA") {

      setStateIfMounted(() {
     _messages.add(userMessage);
    });
   }
  }
  else {
    if(userMessage.userTo == widget.userId && userMessage.userId == widget.userTo && widget.statu == "HASTA"){

    setStateIfMounted(() {
     _messages.add(userMessage);
    });
    /* print(userMessage.message); */
  } else if (userMessage.userTo == widget.userTo && userMessage.userId == widget.userId && widget.statu == "DOKTOR") {

      setStateIfMounted(() {
     _messages.add(userMessage);
    });
   }
```

Şekil 18: client fetchMessages event'inin ve thread event'inin dinlenmesi

ile emit ediyoruz. Kullanıcı mesajı gönder butonuna bastıktan sonrada messagecontroller'ı clear ile mesaj girilen yeri temizliyoruz.



```
client.on("messages", function(data) {
  console.log(data);
  client.emit("thread", data);
  client.broadcast.emit("thread", data);
  db.query("INSERT INTO `messages` (`user_id`,`user_to`,`message`, `statu`) VALUES ('"+data.user_id+"','"+data.user_to+"','"+data.message+"', '"+data.statu+"')")
});
```

Şekil 19: girilen mesajın server'da karşılanması

Şekil 19'da görüldüğü üzere yeni girilen mesajı kimden kime mesajı ve statü'sünü veri tabanında messages tablosuna INSERT ediyoruz.Böylelikle girilen mesajları sürekli veri tabanında depoluyoruz.Bu girilen mesajı thread event'ine emit ediyoruz.

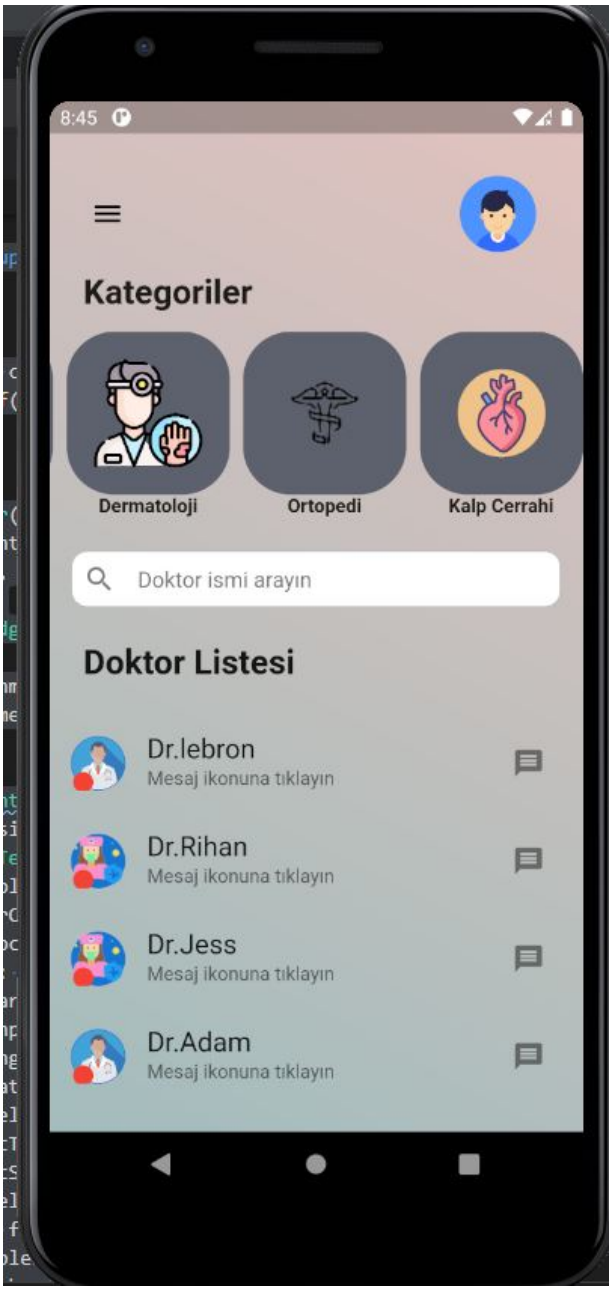
Şekil 18'de thread event'ini dinliyoruz burada eğer mesaj bir HASTA tarafından atıldıysa doktor tarafında userTo'da hasta id'si userId'de hastanın kendi id'si ve statü doktor ise messages listemize ekliyoruz. karşı taraf içinde user to userto ve user id'side kendi user'di ve statüsünde mesaj HASTA statülü bir kişi tarafından gönderildiğinden hasta ise messages içerisine ekliyoruz. Eğer mesajı gönderen DOKTOR statüsüne sahip biriysede statü kısmı tam tersi şekilde gerisi aynı şekilde gerçekleşmekte. Bı anlattığımıza bir örnek vermek gerekirse Hasta statüsüne sahip kişi 1 den 2 ye doktor veya 2 den 1'e doktor olan mesajları messages'e ekler böylelikle kişiler arası 1 e 1 konuşma sağlamış olduk.

```
        children: <Widget>[
          Column(
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: _messages.map((message) {
                //print(message);
                return ChatBubble(
                  message: message.message,
                  isMe: message.userId == widget.userId,
                ); // ChatBubble
              }).toList()), // Column
        ], // <Widget>[]
      ), // Column
    ); // Padding
  }
```

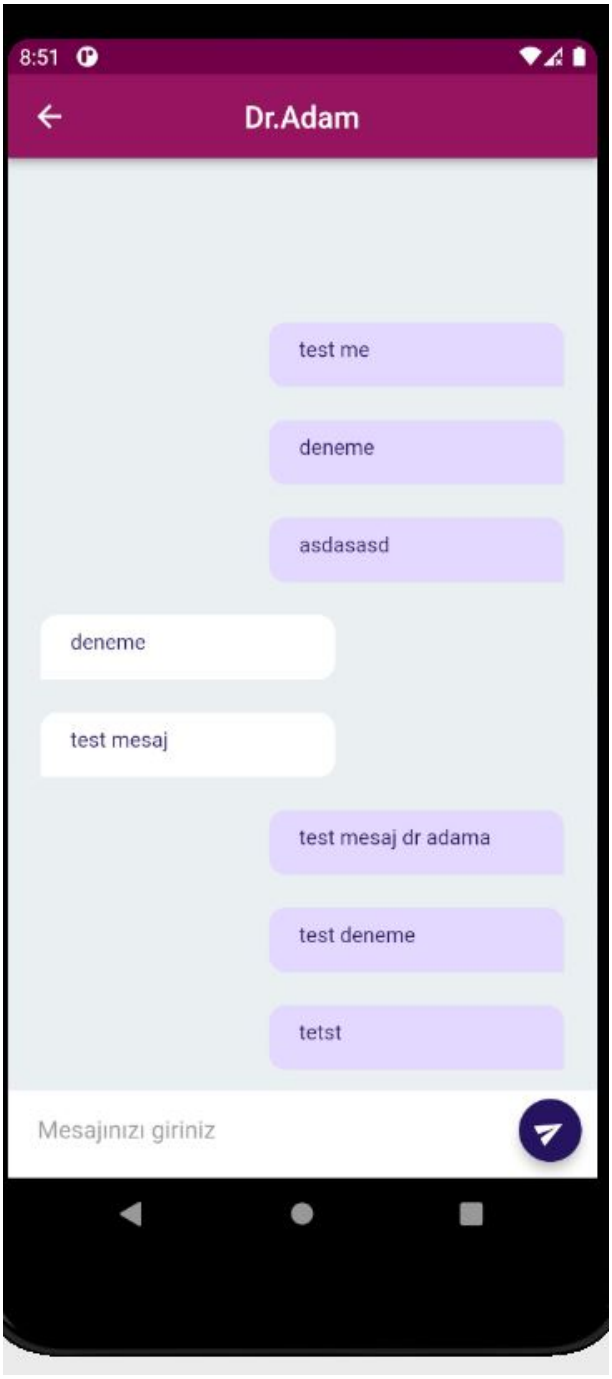
Şekil 20: ChatBubble widget

Bir diğer önemli kısım şekil 20'de ChatBubble yani mesaj baloncuğuna gönderilen parametre olan isMe parametresidir bu parametre eğer uygulamaya giriş yapan kişinin id'si kendi id'si ise eşitlik true olacak ve mesaj sağ tarafta görünecek.false ise yani karşıdan gelen bir mesaj ise gelen mesaj solda görünecek.

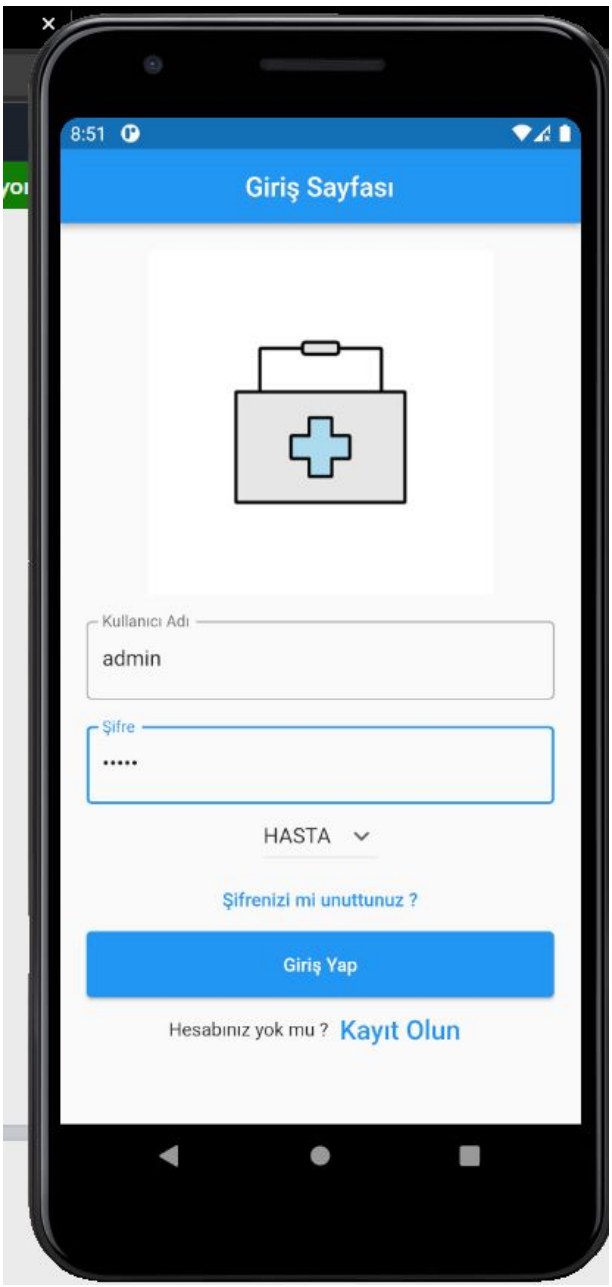
EKRAN GÖRÜNTÜLERİ



Şekil 21: Ekran görüntüleri



Şekil 22: Ekran görüntüleri



Şekil 23: Ekran görüntüleri

BÖLÜM: IV

SONUÇ

4

SONUÇ

HASTA ve DOKTOR arasında kişiden kişiye(1e1) bir mesajlaşma uygulaması ve arayüzü geliştirdik.Hastalar şikayetlerini istediği doktoru seçip mesaj olarak iletebilmekte ve doktorlar'da hastalardan gelen mesajları okuyup cevap verebilmektedir.

K A Y N A K Ç A

KAYNAK KODLAR İLK FLUTTER KODLARI GELİYOR

DEVAMINDA NODEJS KODLARI

——————api klasörü—————— doctors-api.dart import ’dart:convert’; import ’package:client/models/doktor.dart’; import ’package:http/http.dart’ as http;

class DoctorsApi static Future<List<Doktor» getDoctors(String query) async final url= Uri.parse(’http://localhost:4320/getSearchDoctors’); final response= await http.get(url); final List doctors= json.decode(response.body);

print(doctors); return doctors.map((json)=> Doktor.fromJson(json)).where((doktor) final doctorName= doktor.username.toLowerCase(); final anaBilimDali= doktor.anaBilimId.toString(); final searchLower= query.toLowerCase();

print(searchLower); return doctorName.contains(searchLower);).toList();

users-api.dart

import ’package:http/http.dart’ as http; import ’dart:convert’; import ’package:client/models/user.dart’;

class UsersApi static Future<List<User» getUsers(String query) async final url= Uri.parse(’http://localhost: final response= await http.get(url); final List users= json.decode(response.body);

print(users); return users.map((json)=> User.fromJson(json)).where((user) final userName= user.username.toLowerCase();

final searchLower= query.toLowerCase();

print(searchLower); return userName.contains(searchLower);).toList();

models klasörü anabilimdali.dart import ’dart:convert’;

List<AnaBilimDali> anaBilimDaliFromJson(String str)=> List<AnaBilimDali>.from(json.decode(str).map(

AnaBilimDali.anaBilimDalifromJson(x)));

String anaBilimDaliToJson(List<AnaBilimDali> data)=> json.encode(List<dynamic>.from(data.map((x)=>

x.anaBilimDalitoJson())));

class AnaBilimDali AnaBilimDali(required this.id, required this.anaBilimDali,);

late final int id; late final String anaBilimDali;

AnaBilimDali.anaBilimDalifromJson(Map<String, dynamic> json) id= json['id']; anaBilimDali=

json['anaBilimDali'];

Map<String, dynamic> anaBilimDalitoJson() final $_{d}ata =< String, dynamic >;_{d}ata['id'] =$

$id;_{d}ata['anaBilimDali'] = anaBilimDali; return_{d}ata;$

doktor.dart import 'dart:convert'; /* Doktor fromJson(String str)=> Doktor.fromJson(json.decode(str));

String toJson(Doktor data)=> json.encode(data.toJson()); */ class Doktor

final int id; final String username; final String password; final String cinsiyet; final String

statu; final int anaBilimId;

const Doktor (required this.id, required this.username, required this.password, required

this.cinsiyet, required this.statu, required this.anaBilimId,);

factory Doktor.fromJson(Map<String, dynamic> json)=> Doktor(id: json['id'], username:

json['username'], password: json['password'], cinsiyet: json['cinsiyet'], statu: json['statu'],

anaBilimId: json['anaBilimId'],);

Map<String, dynamic> toJson()=> 'id' : id, 'username': username, 'password': password,

'cinsiyet': cinsiyet, 'statu': statu, 'anaBilimId': anaBilimId, ; user.dart import 'dart:convert';

import 'package:http/http.dart' as http; Future<User> fetchUser(int userId) async final

response= await http.get(Uri.parse('http://localhost:3000/profile/userId'));

if (response.statusCode== 200) //print(response.body); // If the server did return a 200 OK response, // then parse the JSON. final jsonresponse= json.decode(response.body); return User.fromJson(jsonresponse[0]); else // If the server did not return a 200 OK response, // then throw an exception. throw Exception('Failed to load User');

List<User> userFromJson(String str)=> List<User>.from(json.decode(str).map((x)=> User.fromJson(x)));

String userToJson(List<User> data)=> json.encode(List<dynamic>.from(data.map((x)=> x.toJson())));

class User User(required this.id, required this.username, required this.password, required this.cinsiyet, required this.stat,); late final int id; late final String username; late final String password; late final String cinsiyet; late final String stat;

User.fromJson(Map<String, dynamic> json) id= json['id']; username= json['username']; password= json['password']; cinsiyet= json['cinsiyet']; stat= json['statu'];

Map<String, dynamic> toJson() final $_data =< String, dynamic >; _data['id'] = id; _data['username'] = username; _data['password'] = password; _data['cinsiyet'] = cinsiyet; _data['statu' stat; return_data;$

userChat.dart

import 'dart:convert';

UserChat fromJson(String str)=> UserChat.fromJson(json.decode(str));

String toJson(UserChat data)=> json.encode(data.toJson());

class UserChat UserChat(required this.userId, required this.userTo, required this.message, required this.statu,);

late final int userId; late final int userTo; late final String message; late final String statu;

UserChat.fromJson(Map<String, dynamic> json) userId= json['user$_id$']; $userTo = json['user_to']; message$ $json['message']; statu = json['statu'];$

Map<String, dynamic> toJson() final $_{d}ata =< String, dynamic >;_{d}ata['user_{i}d'] = userId;_{d}ata['user_{t}o'] = userTo;_{d}ata['message'] = message;_{d}ata['statu'] = statu; return_{d}ata;$

screens klasörü chatPage.dart import 'package:flutter/material.dart'; import 'package:client/models/UserCha

import 'package:socket$_{i}o_{c}lient/socket_{i}o_{c}lient.dart'asIO;$

class ChatPage extends StatefulWidget final int userId; final int userTo; final String usernameMe; final String usernameTo; final String statu;

ChatPage(Key? key, required this.userId, required this.userTo, required this.usernameMe, required this.usernameTo, required this.statu) : super(key: key);

@override $_{C}hatPageStatecreateState() =>_{C} hatPageState();$

class $_{C}hatPageStateextendsState < ChatPage >$

final TextEditingController $_{m}essageController = TextEditingController(); finalScrollController_{s}crol ScrollController(); finalList < UserChat >_{m} essages = [];$

void setStateIfMounted(f) if (mounted) setState(f);

late IO.Socket socket;

@override void initState() try print("test"); socket= IO.io('http://10.0.2.2:4320', <String, dynamic> 'transports': ['websocket'], 'autoConnect': false,);

socket.connect();

socket.emit('fetchOldMessages', UserChat(userId: widget.userId, userTo: widget.userTo, message: '', statu: widget.statu, //widget.username).toJson());

socket.on("fetchMessages", (data) var c= data.length; print(c);

for(int i= 0; i < c; i++) var oldMessages= UserChat.fromJson(data[i]); print(oldMessages.message); setStateIfMounted(() $_{m}essages.add(oldMessages);); //print(oldMessages.message););$

socket.on("thread", (data) var userMessage= UserChat.fromJson(data); print(userMessage);

if(userMessage.statu== "HASTA") if(userMessage.userTo== widget.userId userMessage.userId==

widget.userTo widget.statu== "DOKTOR")

setStateIfMounted(() $_m essages.add(userMessage);$); / $*$ $print(userMessage.message);*/elseif(user$

$widget.userTouserMessage.userId == widget.userIdwidget.statu == "HASTA")$

setStateIfMounted(() $_m essages.add(userMessage);$); $elseif(userMessage.userTo == widget.userIdu$

setStateIfMounted(() $_m essages.add(userMessage);$); / $*$ $print(userMessage.message);*/elseif(user$

$widget.userTouserMessage.userId == widget.userIdwidget.statu == "DOKTOR")$

setStateIfMounted(() $_m essages.add(userMessage);$);

);

catch (e) print(e);

super.initState();

@override Widget build(BuildContext context) print(widget.userId); print(widget.userTo);

print(widget.usernameMe); print(widget.usernameTo); print(widget.statu);

Size size= MediaQuery.of(context).size; return Scaffold(appBar: AppBar(centerTitle:

true, title: Text(widget.usernameTo), backgroundColor: const Color(0xFF971160)), body:

SafeArea(child: Container(color: const Color(0xFFEAEFF2), child: Column(mainAxisAlignment:

MainAxisAlignment.start, children: <Widget>[Flexible(child: MediaQuery.removePadding(

context: context, removeTop: true, child: ListView.builder(controller: $_s crollController, physics$:

$constBouncingScrollPhysics(), reverse :_m essages.isEmpty?false : true, itemCount :$

$1, shrinkWrap : false, itemBuilder : (BuildContextcontext, intindex)returnPadding(padding : constE$

$Colors.white, padding : constEdgeInsets.only(bottom : 10, left : 20, right : 10, top :$

$5), child : Row(children :< Widget > [Flexible(child : Container(child : TextField(minLines :$

$1, maxLines : 5, controller :_m essageController, textCapitalization : TextCapitalization.sentences, dec$

$const InputDecoration.collapsed(hintText : "Mesajßnßzßgiriniz", hintStyle : TextStyle(color :$
$Colors.grey,),),),),), SizedBox(height : 43, width : 42, child : FloatingActionButton(backgroundCol$
$constColor(0xFF271160), onPressed : ()asyncif(_messageController.text.trim().isNotEmpty)Stringm$

socket.emit("messages", UserChat(userId: widget.userId, userTo: widget.userTo, message: message, statu: widget.statu,) .toJson());

$_messageController.clear();, mini : true, child : Transform.rotate(angle : 5.79449, child :$
$constIcon(Icons.send, size : 20)),),),],),)],),),),);$

class ChatBubble extends StatelessWidget final bool isMe; final String message;

ChatBubble(Key? key, required this.message, this.isMe= true,); @override Widget build(BuildContext context) Size size= MediaQuery.of(context).size; return Container(margin: const EdgeInsets.symmetric(horizontal: 12, vertical: 6), child: Column(mainAxisAlignment: isMe ? MainAxisAlignment.end : MainAxisAlignment.start, crossAxisAlignment: isMe ? CrossAxisAlignment.end : CrossAxisAlignment.start, children: <Widget>[Container(padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 10), margin: const EdgeInsets.symmetric(vertical 5.0), constraints: BoxConstraints(maxWidth: size.width * .5), decoration: BoxDecoration(color: isMe ? const Color(0xFFE3D8FF) : const Color(0xFFFFFFFF), borderRadius: isMe ? const BorderRadius.only(topRight: Radius.circular(11), topLeft: Radius.circular(11), bottomRight: Radius.circular(0), bottomLeft: Radius.circular(11),) : const BorderRadius.only(topRight: Radius.circular(11), topLeft: Radius.circular(11), bottomRight: Radius.circular(11), bottomLeft: Radius.circular(0),),), child: Column(crossAxisAlignment: CrossAxisAlignment.start, mainAxisSize: MainAxisSize.min, children: <Widget>[Text(message, textAlign: TextAlign.start, softWrap: true, style: const TextStyle(color: Color(0xFF2E1963), fontSize: 14),), Align(alignment: Alignment.centerRight, child: Padding(padding: const EdgeInsets.only(top: 7), /*

child: Text("Kimden: " + username, textAlign: TextAlign.end, style: const TextStyle(color: Color(0xFF594097), fontSize: 12),), */),)],),),],),);

hospital-screen-background.dart import 'package:flutter/material.dart';

class HospitalBackground extends StatelessWidget final Widget child;

const HospitalBackground(Key? key,required this.child) : super(key: key); @override Widget build(BuildContext context) Size size= MediaQuery.of(context).size; return Container(width: size.width, height: size.height, child: Stack(alignment: Alignment.center, children: [child,],),); hospital-screen-body.dart import 'package:client/models/anaBilimDali.dart'; import 'package:client/models/doktor.dart'; import 'package:client/models/user.dart'; import 'package:client/models/onlineList.dart'; import 'package:client/screens/chatPage.dart'; import 'package:client/widgets/searchbar.dart'; import 'package:flutter/material.dart'; import '../common$_{w}idgets.dart$

$http/http.dart'ashttp; import'package : socket_{i}o_{c}lient/socket_{i}o_{c}lient.dart'asIO; import'../constants.d$

$client/api/doctors_{a}pi.dart';$

class Body extends StatefulWidget final int id; final String username; final String statu; final String cinsiyet;

Body(Key? key, required this.id, required this.username, required this.statu, required this.cinsiyet) : super(key: key);

@override State<Body> createState()=> $_{B}odyState();$

class $_{B}odyStateextendsState < Body > int?counter; int?counter2; varuserResult; varanabilimDaliR$

Future callPerson(int id, String userName, String statu) async

try final response= await http.get(Uri.parse('http://localhost:4320/fetchAllUsers/$statu'$)); $varresult =$

$userFromJson(response.body); //print(result[0].username); if(mounted)setState(()counter = result$

Future fetchAnaBilimDallari() async try

final response2= await http.get(Uri.parse('http://localhost:4320/fetchAnaBilimDallari'));

print("2132131"); print(response2); print(response2.body); var result2= anaBilimDaliFromJson(response2.bod

print(result2[0].anaBilimDali); setState(() counter2= result2.length; anabilimDaliResult=

result2;); catch (e) print(e.toString());

late IO.Socket socket;

@override void initState() try socket= IO.io('http://10.0.2.2:4320', <String, dynamic>

'transports': ['websocket'], 'autoConnect': false,);

socket.connect();

socket.on("connect", (data) print(socket.id); socket.emit('onlineHasta', widget.id););

socket.on('doktorOnline', (data) print(data); doktorOnlineId= data; print("doktorOnlineId

ytazdiriliyor");

); catch (e) print(e);

super.initState(); callPerson(widget.id, widget.username, widget.statu); fetchAnaBilimDallari();

super.initState(); init();

Future init() async final doctors= await DoctorsApi.getDoctors(query); setState(()=>

this.doctors= doctors);

@override Widget build(BuildContext context) Size size= MediaQuery.of(context).size;

var array2= []; return userResult== null ? Center(child: CircularProgressIndicator()) : HospitalBackground(

child: SingleChildScrollView(child: Column(crossAxisAlignment: CrossAxisAlignment.start,

mainAxisAlignment: MainAxisAlignment.start, children: [

Padding(padding: EdgeInsets.only(left:15,right: 25), child: AppBar(backgroundColor:

Colors.transparent, elevation: 0, leading: Icon(Icons.menu,color: Colors.black,size: 25,),

actions: [GestureDetector(//onDoubleTap: ()Navigator.push(context,MaterialPageRoute(builder:

(context)return ProfileScreen(userId: widget.userId);));, child: Container(child: CircleAvatar(

child: widget.cinsiyet== "K" ? Image.asset("assets/images/hastawoman.png",alignment: Alignment.center,) : Image.asset("assets/images/hastamale.png",alignment: Alignment.center,),), width: 75, height: 75, decoration: BoxDecoration(shape: BoxShape.circle, gradient: const LinearGradient(colors: [Colors.white10,Colors.white], stops: [0,1],),),),),],),), const Padding(padding: EdgeInsets.only(left: 25,top:15), child: Text("Kategoriler",style: TextStyle(fontWeight: FontWeight.bold,fontSize: 25),),), Container(margin: EdgeInsets.only(top:10), width: size.width, height: 150, child: ListView.builder(itemCount: counter2, itemBuilder: (context, index)

return anabilimDaliResult== null ? Center(child: CircularProgressIndicator()) : Row(children: [MyContainer(ImgName: "*anabilimDaliResult*[*index*].*anaBilimDali.replaceAll*('','').*png*", *Im anabilimDaliResult*[*index*].*anaBilimDali*,),],);, *physics* : *BouncingScrollPhysics*(), *scrollDirection* : *Axis.horizontal*,

),), buildSearch(), const Padding(padding: EdgeInsets.only(left: 25,top:10), child: Text("Doktor Listesi",style: TextStyle(fontWeight: FontWeight.bold,fontSize: 25),),), Container(height: 300, child: ListView.builder(itemBuilder: (context, index) return Container(child: ListTile(title: Text(doctors[index].username, style: titleStyle,), subtitle: Text('Mesaj ikonuna tıklayın'), leading: Stack(children: [CircleAvatar(child: doctors[index].cinsiyet== "K" ? Image.asset('assets/images/fem Image.asset('assets/images/maledoctor.png'),), Positioned(bottom: -3, left: 0, child: doctors[index].id== doktorOnlineId ? Icon(Icons.circle, color: Colors.green, size: 18,) : Icon(Icons.circle, color: Colors.red, size: 18,)),],), trailing: IconButton(icon: const Icon(Icons.message), onPressed: () //print(userResult[index].id); Navigator.push(context, MaterialPageRoute(builder: (context)=> ChatPage(userId: widget.id, userTo: userResult[index].id, usernameMe: widget.username, usernameTo: userResult[index].username, statu: widget.statu,))); ,),),); , itemCount: doctors.length, physics: BouncingScrollPhysics(),),),

],),),); Widget buildSearch()=> SearchWidget(text: query, hintText: 'Doktor ismi arayın', onChanged: searchDoctor,); Future searchDoctor(String query) async final doctors= await DoctorsApi.getDoctors(query); print("denemelesa1212312312321332"); if(!mounted) return; setState(() this.query= query; this.doctors= doctors;); hospital.screen.dart import 'package:client/screens/hospital$_{s}creen_{b}ody.dart'; import' package : flutter / material.dart';

class HospitalScreen extends StatelessWidget final int id; final String username; final String statu; final String cinsiyet; const HospitalScreen(Key? key, required this.id, required this.username, required this.statu, required this.cinsiyet) : super(key: key);

@override Widget build(BuildContext context)

return MaterialApp(debugShowCheckedModeBanner: false, home: Scaffold(body: Container(decoration: BoxDecoration(gradient: LinearGradient(colors: [Color(0xffE2C2BE), Color(0xFFA3BDBD),], begin: Alignment.topRight, end: Alignment.bottomLeft,),), child: Body(id: id, username: username, statu: statu, cinsiyet: cinsiyet,)),),); loginPage.dart import 'package:client/models/user.dart'; import 'package:client/screens/hospital$_{s}creen.dart'; import' package : client / screens / registerPage.dart'; i client / screens / userMainPage.dart'; import' package : flutter / material.dart'; import' package : http / http.dart' ashttp; import' package : fluttertoast / fluttertoast.dart';

class LoginPage extends StatefulWidget

const LoginPage(Key? key) : super(key: key);

@override State<LoginPage> createState()=> $_{L}oginPageState();$

class $_{L}oginPageStateextendsState < LoginPage > TextEditingControllernameController = TextE$

Future login(BuildContext cont) async if(nameController.text== '' || passwordController.text== '') Fluttertoast.showToast(msg: 'Kullanıcı adı ve Şifre alanı boş bırakılamaz!', toastLength: Toast.LENGTH$_{S}HORT, gravity : ToastGravity.CENTER, textColor : Colors.white, fontSize :$ 16.0); $elsevarurl = Uri.parse('http : / / localhost : 4320 / login'); varresponse = awaithttp.post(url, b$

var result= userFromJson(response.body); var statu= result[0].stat;

if(statu== "HASTA") print("HASTA GİRİŞ YAPTI"); Navigator.push(context, MaterialPageRoute(builder: (context)=> HospitalScreen(id: result[0].id,username: result[0].username, statu: result[0].stat, cinsiyet: result[0].cinsiyet,)) //UserHomePage); else print("DOKTOR GİRİŞ YAPTI"); Navigator.push(context, MaterialPageRoute(builder: (context)=> UserHomePage(id: result[0].id,username: result[0].username, statu: result[0].stat,))); var items= ['HASTA', 'DOKTOR']; @override Widget build(BuildContext context) return Scaffold(appBar: AppBar(title: Text('Giriş Sayfası'), centerTitle: true,), body: Padding(padding: EdgeInsets.all(10), child: ListView(children: <Widget>[Container(alignment: Alignment.center, padding: EdgeInsets.all(10), height: 280, child: Image.asset('assets/images/login-logo.jpg')), Container(padding: EdgeInsets.all(10), child: TextField(controller: nameController, decoration: InputDecoration(border: OutlineInputBorder(), labelText: 'Kullanıcı Adı',),),), Container(padding: EdgeInsets.fromLTRB(10, 10, 10, 0), child: TextField(obscureText: true, controller: passwordController, decoration: InputDecoration(border: OutlineInputBorder(), labelText: 'Şifre',),),), Center(child: DropdownButton(value: dropdownvalue, icon: const Icon(Icons.keyboard$_{a}rrow_{d}own), //$Arraylistofitemsitems$: $items.map((String items)returnDropdownMenuItem(value : items, child : Text(items),);).toList(), /$ $(String?newValue)setState(()dropdownvalue = newValue!;);,),),

TextButton(onPressed: ()

, child: Text('Şifrenizi mi unuttunuz ?'),), Container(height: 50, padding: EdgeInsets.fromLTRB(10, 0, 10, 0), child: ElevatedButton(child: Text('Giriş Yap'), onPressed: () async login(context); if(nameController.text.trim().isNotEmpty passwordController.text.trim().isNotEmpty) String message= nameController.text.trim() + passwordController.text.trim(); print(message); ,)),

Container(child: Row(children: <Widget>[Text('Hesabınız yok mu ?'), TextButton(child: Text('Kayıt Olun', style: TextStyle(fontSize: 20),), onPressed: () Navigator.push(

context, MaterialPageRoute(builder: (context)=> RegisterPage())); ,)], mainAxisAlignment: MainAxisAlignment.center,))],))); registerPage.dart import 'package:flutter/material.dart'; import 'package:http/http.dart' as http; import 'package:fluttertoast/fluttertoast.dart'; import 'package:client/models/user.dart';

class RegisterPage extends StatefulWidget RegisterPage(Key? key) : super(key: key);

@override $_R egisterPageStatecreateState() =>_R egisterPageState();$

class $_R egisterPageStateextendsState < RegisterPage > TextEditingControllernameController = T$

Future register(BuildContext cont) async

if(nameController.text== '' || passwordController.text== '') Fluttertoast.showToast(msg: 'Kullanıcı adı ve Şifre alanı boş bırakılamaz!', toastLength: Toast.LENGTH$_S HORT, gravity$: $ToastGravity.CENTER, textColor : Colors.white, fontSize : 16.0); elsevarurl = Uri.parse('http : //l$

var data= response.body; print(data);

print("KAYIT BAŞARILI");

var result= userFromJson(response.body); var statu= result[0].stat;

if(statu== "HASTA") print("HASTA KAYIT OLDU"); /* Navigator.push(context, MaterialPageRoute(build (context)=> HospitalScreen(id: result[0].id,username: result[0].username, statu: result[0].stat, cinsiyet: result[0].cinsiyet,)) //UserHomePage); */ else print("DOKTOR KAYIT OLDU"); /* Navigator.push(context, MaterialPageRoute(builder: (context)=> UserHomePage(id: result[0].id,username: result[0].username, statu: result[0].stat,))); */

var items= ['HASTA', 'DOKTOR']; var items2= ['K', 'E']; @override Widget build(BuildContext context) return Scaffold(appBar: AppBar(title: Text('Kayıt Sayfası'), centerTitle: true,), body: Padding(padding: EdgeInsets.all(10), child: ListView(children: <Widget>[Container(alignment: Alignment.center, padding: EdgeInsets.all(10), height: 280, child: Image.asset('assets/images/login-logo.jpg')), Container(padding: EdgeInsets.all(10), child: TextField(controller: nameController,

decoration: InputDecoration(border: OutlineInputBorder(), labelText: 'Kullanıcı Adı',),),), Container(padding: EdgeInsets.fromLTRB(10, 10, 10, 0), child: TextField(obscureText: true, controller: passwordController, decoration: InputDecoration(border: OutlineInputBorder(), labelText: 'Şifre',),),), Row(mainAxisAlignment: MainAxisAlignment.center, children: [DropdownButton(value: dropdownvalue, icon: const Icon(Icons.keyboard$_{a}rrow_{d}own$), //$Arraylistofitems$ $items.map((Stringitems)returnDropdownMenuItem(value : items, child : Text(items),);).toList(), /$ $(String?newValue)setState(()dropdownvalue = newValue!;;), SizedBox(width : 30,), DropdownB$ $dropdownvalue2, icon : constIcon(Icons.keyboard_arrow_down), //Arraylistofitemsitems :$ $items2.map((Stringitems2)returnDropdownMenuItem(value : items2, child : Text(items2),);).toLis$ $(String?newValue)setState(()dropdownvalue2 = newValue!;;),],), TextButton(onPressed :$ $()//forgotpasswordscreen, child : Text('ifrenizimiunuttunuz?'),), Container(height :$ $50, padding : EdgeInsets.fromLTRB(10, 0, 10, 0), child : ElevatedButton(child : Text('KayßtOl'), onP$ $()asyncregister(context); if(nameController.text.trim().isNotEmptypasswordController.text.trim().$ $client/models/user.dart'; import'package : client/screens/chatPage.dart'; import'package :$ $flutter/material.dart'; import'package : http/http.dart'ashttp; import'package : socket_io_client/socke$ $client/api/users_api.dart'; import'package : client/widgets/searchbar.dart';$

class UserHomePage extends StatefulWidget final int id; final String username; final String statu; UserHomePage(Key? key, required this.id, required this.username, required this.statu) : super(key: key);

@override $_{U}serHomePageStatecreateState() =>_{U}$ $serHomePageState();$

class $_{U}serHomePageStateextendsState < UserHomePage > int?counter; varuserResult; Stringquer$

try final response= await http.get(Uri.parse('http://localhost:4320/fetchAllUsers/$statu'$)); $varresult =$ $userFromJson(response.body); //print(result[0].username); if(mounted)setState(()counter = result$); return result; catch(e) print(e.toString());

late IO.Socket socket;

@override void initState() try socket= IO.io('http://10.0.2.2:4320', <String, dynamic> 'transports': ['websocket'], 'autoConnect': false,);

socket.connect();

socket.on("connect", (data) print(socket.id); socket.emit('onlineDoktor', widget.id););

socket.on('hastaOnline', (data) print(data); hastaOnlineId= data;);

catch (e) print(e);

super.initState(); callPerson(widget.id, widget.username, widget.statu); init();

Future init() async final users= await UsersApi.getUsers(query); setState(()=> this.users= users);

@override Widget build(BuildContext context) //print(widget.id); //print(widget.username); return userResult== null ? Center(child: CircularProgressIndicator()) : Scaffold(appBar: AppBar(title: widget.statu== 'HASTA' ? Text('Doktor Listesi') : Text('Hasta Listesi'), backgroundColor: Colors.teal,), body: Column(children: [buildSearch(), Expanded(child: ListView.builder(physics: const BouncingScrollPhysics(), itemCount: users.length, itemBuilder: (context, index) return ListTile(title: Text(users[index].username, style: titleStyle,), subtitle: Text('Mesaj ikonuna tıklayın'), leading: Stack(children: [CircleAvatar(child: users[index].cinsiyet== "K" ? Image.asset('assets/images/hastawoman.png') : Image.asset('assets/images/hastamale.png'), backgroundColor: Colors.black54,), Positioned(bottom: -3, left: 0, child: users[index].id== hastaOnlineId ? Icon(Icons.circle, color: Colors.green, size: 18,) : Icon(Icons.circle, color: Colors.red, size: 18,)),],

), trailing: IconButton(icon: const Icon(Icons.message), onPressed: () //print(userResult[index].id); Navigator.push(context, MaterialPageRoute(builder: (context)=> ChatPage(userId: widget.id,

userTo: userResult[index].id, usernameMe: widget.username, usernameTo: userResult[index].username,

statu: widget.statu,))); ,),);),),],

), floatingActionButton: FloatingActionButton(child: Icon(Icons.refresh), backgroundColor:

Colors.teal, onPressed: () callPerson(widget.id, widget.username, widget.statu); ,), bottomNavigationBar:

Container(decoration: BoxDecoration(color: Colors.blueAccent,borderRadius: BorderRadius.circular(10.0),bo

Border.all(color: Colors.black, width: 2.0), gradient: LinearGradient(colors: [Colors.white60,

Colors.teal,]), boxShadow: [BoxShadow(color: Colors.grey , blurRadius: 2.0, offset:

Offset(2.0,2.0))]), child: Padding(padding: EdgeInsets.all(10.0), child: BottomAppBar(

color: Colors.transparent, child: Text('Giriş yapan doktor: ' + widget.username, style: userNameStyle,),

elevation: 0,

),),),);

Widget buildSearch()=> SearchWidget(text: query, hintText: 'Hasta ismi giriniz', onChanged:

searchUser,); Future searchUser(String query) async final users= await UsersApi.getUsers(query);

if(!mounted) return;

setState(() this.query= query; this.users= users;);

widgets klasörü import 'package:flutter/material.dart';

class SearchWidget extends StatefulWidget final String text; final ValueChanged<String>

onChanged; final String hintText;

const SearchWidget(Key? key, required this.text, required this.onChanged, required

this.hintText,) : super(key: key);

@override $_{S}earchWidgetStatecreateState() =>_{S} earchWidgetState();$

class $_{S}earchWidgetStateextendsState < SearchWidget > finalcontroller = TextEditingController($

@override Widget build(BuildContext context) final styleActive= TextStyle(color: Colors.black); final styleHint= TextStyle(color: Colors.black54); final style= widget.text.isEmpty ? styleHint : styleActive;

return Container(height: 42, margin: const EdgeInsets.fromLTRB(16, 16, 16, 16), decoration: BoxDecoration(borderRadius: BorderRadius.circular(12), color: Colors.white, border: Border.all(color: Colors.black26),), padding: const EdgeInsets.symmetric(horizontal: 8), child: TextField(controller: controller, decoration: InputDecoration(icon: Icon(Icons.search, color: style.color), suffixIcon: widget.text.isNotEmpty ? GestureDetector(child: Icon(Icons.close, color: style.color), onTap: () controller.clear(); widget.onChanged(''); FocusScope.of(context).requestFocus(FocusNode()); ,) : null, hintText: widget.hintText, hintStyle: style, border: InputBorder.none,), style: style, onChanged: widget.onChanged,),); lib klasörü commonswidget.dart import 'package:flutter/material.dart';

class MyTextField extends StatelessWidget final String hinttext; final String labeltext; final bool readonly; final TextEditingController controller;

const MyTextField(Key? key,required this.hinttext,required this.labeltext,required this.readonly, required this.controller) : super(key: key);

@override Widget build(BuildContext context) Size size= MediaQuery.of(context).size; return Container(margin: EdgeInsets.only(top:4,bottom: 4), padding: EdgeInsets.fromLTRB(0, 4, 0, 0), width: size.width*0.6, child: TextFormField(controller: controller, validator: (value) if (value== null || value.isEmpty) return 'This field cannot be left blank'; return null; , readOnly: readonly, decoration: InputDecoration(labelText: labeltext, hintText: hinttext, border: OutlineInputBorder(),),),); class MyContainer extends StatelessWidget final String ImgName; final String ImgDesc; const MyContainer(Key? key,required this.ImgName,required this.ImgDesc) : super(key: key);

@override Widget build(BuildContext context) Size size= MediaQuery.of(context).size;
return Container(margin: EdgeInsets.only(left:10,top: 5), child: Column(children: [ClipRRect(
borderRadius: BorderRadius.circular(35), child: Container(padding: EdgeInsets.all(20),
width: 120, color: Color(0xff5E616D), child: Image.asset("assets/images/*ImgName*", *alignment* :
Alignment.center,),),), *Text*(*"*ImgDesc",style: TextStyle(fontWeight: FontWeight.bold),),
],),); class MyAnimatedContainer extends StatelessWidget final String ImgName; final
String DocName; const MyAnimatedContainer(Key? key,required this.ImgName,required
this.DocName,) : super(key: key);
@override Widget build(BuildContext context) return AnimatedContainer(margin: EdgeInsets.only(bottom
alignment: Alignment.center, duration: Duration(seconds: 1), height: 70, child: Card(color:
Colors.grey, child: Row(mainAxisAlignment: MainAxisAlignment.start, children: [Container(
padding:EdgeInsets.only(left:10,top:5,bottom: 5,right: 10), child: Image.asset("assets/images/*ImgName*",),),
10,), *Text*(*"*DocName",style: TextStyle(fontSize: 17,fontWeight: FontWeight.bold),),],),),
); class MyFlatButton extends StatelessWidget final Widget screen; final String text;
const MyFlatButton(Key? key,required this.screen,required this.text) : super(key: key);
@override Widget build(BuildContext context) Size size= MediaQuery.of(context).size;
return Container(margin: EdgeInsets.symmetric(vertical: 6), width: size.width*0.6, decoration:
BoxDecoration(color: Colors.white70, borderRadius: BorderRadius.circular(20), boxShadow:
[BoxShadow(offset: Offset(0,10), blurRadius: 10, color: Colors.grey,),]), child: FlatButton(
onPressed: ()Navigator.push(context,MaterialPageRoute(builder: (context)return screen;));,
child: Text(text,style: TextStyle(),),),); constant.dart import 'package:flutter/material.dart';
const titleStyle= TextStyle(fontSize: 20);
const userNameStyle= TextStyle(color: Colors.black, fontSize: 25, decorationStyle:
TextDecorationStyle.wavy, decorationColor: Colors.teal);

main.dart import ’package:client/notmain.dart’; import ’package:client/screens/loginPage.dart’; import ’package:flutter/material.dart’;

void main() runApp(const MyApp());

class MyApp extends StatelessWidget const MyApp(Key? key) : super(key: key); @override Widget build(BuildContext context) return MaterialApp(debugShowCheckedModeBanner: false, title: ’Flutter Demo’, theme: ThemeData(primarySwatch: Colors.blue,), home: LoginPage(),);

class RegScreen extends StatelessWidget

TextEditingController myId= TextEditingController(); TextEditingController userTo= TextEditingController();

RegScreen(Key? key) : super(key: key);

@override Widget build(BuildContext context) var size= MediaQuery.of(context).size; return Scaffold(body: SafeArea(child: Container(color: const Color(0xFFEAEFF2), height: size.height, width: size.width, child: Container(margin: const EdgeInsets.symmetric(horizontal: 20, vertical: 60), child: Column(crossAxisAlignment: CrossAxisAlignment.center, mainAxisAlignment: MainAxisAlignment.start, children: [Flexible(child: Container(width: size.width * 0.80, child: TextField(controller: myId, cursorColor: Colors.black, autofocus: false, style: const TextStyle(fontSize: 18), keyboardType: TextInputType.text, textInputAction: TextInputAction.go, maxLength: 20, decoration: InputDecoration(labelText: ’Kendi idniz’, hintText: ’Giriniz’, hintStyle: const TextStyle(fontSize: 15), labelStyle: const TextStyle(fontSize: 15, color: const Color(0xFF271160)), enabledBorder: UnderlineInputBorder(borderSide: BorderSide(color: const Color(0xFF271160))), focusedBorder: UnderlineInputBorder(borderSide: BorderSide(color: const Color(0xFF271160))), disabledBorder: UnderlineInputBorder(borderSide: BorderSide(color: const Color(0xFF271160))),),),),), Flexible(child: Container(width: size.width * 0.80,

child: TextField(controller: userTo, cursorColor: Colors.black, autofocus: false, style: const TextStyle(fontSize: 18), keyboardType: TextInputType.text, textInputAction: TextInputAction.go, maxLength: 20, decoration: InputDecoration(labelText: 'Karşı kullanıcı idsi', hintText: 'Giriniz', hintStyle: const TextStyle(fontSize: 15), labelStyle: const TextStyle(fontSize: 15, color: const Color(0xFF271160)), enabledBorder: UnderlineInputBorder(borderSide: BorderSide(color: const Color(0xFF271160))), focusedBorder: UnderlineInputBorder(borderSide: BorderSide(color: const Color(0xFF271160))), disabledBorder: UnderlineInputBorder(borderSide: BorderSide(color: const Color(0xFF271160))),),),),), Container(margin: const EdgeInsets.symmetric(vertica 30), width: size.width * 0.80, height: 50, child: ElevatedButton(style: ElevatedButton.styleFrom(primary: const Color(0xFF271160)), onPressed: () /* print(myId.text); print(userTo.text); */ Navigator.push(context, MaterialPageRoute(builder: (context)=> NewApp(userId: myId.text, userTo: userTo.text,))); , child: Text('Start Chat', style: const TextStyle(fontSize: 17, color: Colors.white)),),),],),),),),);