

Beykent Üniversitesi  
Yazılım Mühendisliği Bölümü  
Yazılım Mühendisliği Tasarım Projesi

Ara Sınav  
2022 - Bahar

# Sport with AI (SAI)

## **Grup Elemanları:**

Hilal Coşkun - 1803013016  
Muhammed Furkan Gülşen - 1803013010  
Yunus Taha Yılmaz - 1803013027

# İÇİNDEKİLER

<b>1 Sport with AI</b>	<b>5</b>
1 a. Projenin Açıklaması	5
1 b. Proje Ekibi	5
1 c. Projenin Aşamaları	6
<b>2 Dönem İçinde Yapılan İşler</b>	<b>7</b>
2 a. Arka Uç Geliştirme (Muhammed Furkan Gülşen)	7
1. Uygulama ve Gizlilik Politikası	7
2. Giriş yap ve Kaydol	8
3. Sosyal Medya	14
4. Kullanıcı	17
5. Egzersiz	19
Ön Uç Geliştirme - Admin Sayfası (Muhammed Furkan Gülşen)	22
2.b Mobil Uygulama (Hilal Coşkun)	32
Sosyal Medya Text(Yazı) Paylaşım Ekranı	32
Sosyal Medya Image(Resim) Paylaşım Ekranı	33
Sosyal Medya Pool(Anket) Paylaşım Ekranı	34
Sosyal Medya Ana Akış Ekranı	35
Sosyal Medya Image(Resim) Paylaşım Kartı	37
Sosyal Medya Pool(Anket) Paylaşım Kartı	38
Sosyal Medya Text(Yazı) Paylaşım Kartı	39
Sosyal Medya Kişisel Profil Ekranı	40
Sosyal Medya Comment(Yorum) Paylaşma Ve Listeleme Ekranı	41
Navigation Bar(Yönlendirme Çubuğu)	42
AppBar(Uygulama Çubuğu) Gösterim Alanı	43
LeaderBoard(Lider Sıralaması) Ekranı	44
Stats(İstatistik) Ekranı	46
İstatistik Verileri Görüntüleme Çizelgesi	47
Score(Skor) ve Steps(Adım) Verileri Görüntüleme Kartları	48
Data Modelleri	50
Sosyal Medya Modeli	50
İstatistik Çizelge Modeli	50
User(Kullanıcı) Modeli	51
Sosyal Medya Yorum Modeli	52
Sosyal Medya Servisleri (Arka Uç Bağlantı)	53
Tüm sosyal medya paylaşımları görüntüleme	53
Kullanıcıya ait filtrelenmiş tüm sosyal medya paylaşımlarını görüntüleme	53
Kullanıcıya ait tüm istatistikleri görüntüleme	54
Kullanıcının sosyal medyada text(yazı) paylaşımı	54
Kullanıcının sosyal medyada pool(anket) paylaşımı	55

Kullanıcının sosyal medyada paylaşılmış olan pool(ankete) oy vermesi	56
Kullanıcının sosyal medyada paylaşılan gönderileri beğenmesi	56
Kullanıcının sosyal medyada paylaşılan gönderiye yorum yapması	57
Sosyal medya paylaşımına ait tüm yorumların görüntülenmesi	58
Kullanıcıya ait tüm bilgilerin getirilmesi	58
<b>2 c. Mobil Uygulama (Yunus Taha Yılmaz)</b>	<b>60</b>
Welcome Ekranı	60
Splash Ekranı	61
Sayfaların Yapımında Kullanılan SWAI Widgetleri	62
Custom Dropdown Widgeti	62
Image Option Select Widgeti	63
Date Picker Widgeti	64
Mini Calendar Widgeti	65
SWAIPasswordTextField	66
Pie Chart Widgeti	67
SWAITextField Widgeti	68
Sign Up Ekranı	69
Tell Us More Ekraneları	70
Boy-Kilo	70
Cinsiyet-Doğum Tarihi	71
Hedef	72
Aktivasyon Kodu Ekranı	73
Login Ekranı	74
Settings Ekraneları	76
Resim Değiştirme Kısmı	77
Edit Profile Ekranı	78
Change Password Ekranı	79
Privacy Policy Ekranı	80
Activity Ekraneları	81
Weekly Goal Ekranı	83
Activity Day Ekranı	84
Rest Ekranı	85
Data Modelleri	86
Token	86
Activation Code	86
Sign Up	87
Data Servisleri	88
Log In	88
Sign Up	89
<b>3 Sonuç</b>	<b>91</b>
3 a. Başarı ile tamamlanan işler	91
3 b. Başarı ile tamamlanamayan işler	91
3 c. Kişilerin görevlerini yerine getirme durumu	91

# 1 Sport with AI

## 1 a. Projenin Açıklaması

Sport with AI mobil uygulaması sayesinde, spor yapıp sağlıklı kalmak isteyen insanlar, başka bir uygulamaya gerek kalmadan tek bir platform üzerinde toplanıyor. Uygulamamız içerisinde yer alan sosyal medya bölümü ile kullanıcılar arasında etkileşim artırılıyor, lig sıralaması ile kullanıcılar arasında rekabet duygusu oluşturulup spor aktivitesini sürekli yapma alışkanlığı kazandırılıyor, yapay zeka ile spor yapma özelliği sayesinde de kullanıcılar spor hareketlerini daha eğlenceli ve daha doğru bir şekilde yapabiliyor. Bu ve buna benzer özellikler sayesinde uygulamamız diğer uygulamalardan daha kapsamlı kullanım alanına sahiptir.

Bahar döneminde projenin ön uç ve arka uç geliştirmelerinin önemli kısmı tamamlanıp final dönemi planlaması ekipçe oluşturulup görev dağılımı yapılmıştır.

## 1 b. Proje Ekibi

Proje ekibi aşağıdaki tabloda görev detayı ile birlikte verilmiştir.

*Tablo1: Proje ekibi.*

No	Numara	Ad Soyad	Projedeki Görev Tanımı
1	1803013027	Yunus Taha Yılmaz	* Mobil uygulama tasarımını yapmak * Flutter teknolojisi kullanarak mobil uygulamayı geliştirmek * Mobil uygulamanın testleri yapmak
2	1803013016	Hilal Coşkun	* Mobil uygulama tasarımını yapmak * Flutter teknolojisi kullanarak mobil uygulamayı geliştirmek * Mobil uygulamanın testleri yapmak
3	1803013010	Muhammed Furkan Gülşen	* Arka Uç geliştirmek * Sunucu ve DevOps sürecini yönetmek * Yapay zeka servisini geliştirmesinin yapılması * Admin panelinin geliştirmek * Mikroservis yapısını geliştirmek

### **1 c. Projenin Aşamaları**

Proje aşamaları ve detayları alt kısımda bulunan tablo aracılığıyla incelenebilir.

*Tablo2: Proje aşamaları.*

No	İş Paketi Adı	Dönem	İçeriği
1	Mobil Uygulama Tasarımı	01.02.2022 - 13.02.2022	Figma kullanılarak mobil uygulama tasarımının yeniden dizayn edilmesi ve renklendirilmesi tamamlandı.
2	Mobil Uygulama Ön Uç Geliştirmesi	14.02.2022 - 08.04.2022	Flutter kullanılarak kullanıcı odaklı uygulama geliştirildi. Uygulama içerisinde kullanıcıların aktivite, sosyal medya, istatistik vb. özellikler kullanılabilecek ekranlar oluşturuldu..
3	Arka Uç Geliştirmesi	14.02.2022 - 08.04.2022	NestJS (NodeJS kütüphanesi) ile frontend tarafında gelecek isteklerin kodları tamamlandı. Yapay zeka sunucusu ile entegrasyon sağlandı.
4	Bulut Entegrasyonu	05.03.2022 - 17.03.2022	AWS kullanılarak Arka Uç ve PostgreSQL bulut üzerinde kısmen çalışabilir hale getirildi.
5	Web Ön Uç Geliştirmesi	16.03.2022 - 02.04.2022	Uygulama içerisinde yer alan sosyal medya paylaşımlarını, spor aktivitelerini vb tüm işlemleri yönetebileceğii Admin sayfasının yaratılması.

## 2 Dönem İçinde Yapılan İşler

### 2 a. Arka Uç Geliştirme (Muhammed Furkan Gülşen)

Arka uç geliştirilmesi en popüler NodeJS frameworklerinden biri olan NestJS ile geliştirilmiştir. Veritabanı olarak PostgreSQL kullanılmıştır. Arka uç ve veritabanı arasındaki bağlantı ise hard-coded yerine en popüler ORM kütüphanelerinden biri olan Prisma ile geliştirilmiştir. Yapay zeka servisi Python ile geliştirilmiş olup, ana arka uç servisi ile arasındaki iletişim RabbitMQ teknolojisi ile sağlanmıştır. Tüm bu servis ve veritabanı AWS sunucu üzerinde çalıştırılıp Docker container sistemi ile kontrol edilmektedir.

#### 1. Uygulama ve Gizlilik Politikası

Bu bölümde uygulamanın açılması için gerekli olan modeli ve versiyonu geliştirilmiştir.

```
model App {
    id          String      @id @unique @default(uuid())
    version     Int         @unique
    versionName String      @unique
    creationDate DateTime?  @default(now())
    isActivity  Boolean
    privacyPolicy PrivacyPolicy?
}
```

schema.prisma

Yukarıdaki verilerin anlamları şu şekildedir:

- **version:** Uygulamanın versiyonu (örn: v1.1 , v2.5 ...)
- **versionName:** Uygulamanın ilgili versiyonuna verdığımız isim (örn: sprintVersion)
- **creationDate:** Versiyonun oluşturulma tarihi.
- **isActivity:** Bu sadece bir App modeli için True değerini alır. Kullanımında olan versiyonu gösterir.
- **privacyPolicy:** Versiyonun bağlı olduğu (one-to-one) gizlilik politikası.

```
model PrivacyPolicy {
    id          String      @id @unique @default(uuid())
    content     String
    creationDate DateTime? @default(now())
    app_id     String      @unique
    app        App         @relation(fields: [app_id], references: [id], onDelete: Cascade)
}
```

schema.prisma

Yukarıdaki verilerin anlamları şu şekildedir:

- **content:** Gizlilik politikasının içeriği.
- **creationDate:** Oluşturulma tarihi.
- **app\_id:** Bağlı olduğu App modelinin id'si.
- **app:** Bağlı olduğu (one-to-one) App modeli.

```

1 import { Body, Controller, Get, Post } from '@nestjs/common';
2 import { AppService } from './app.service';
3 import { PrivacyPolicy } from './models/privacy-policy';
4 import { SaiApp } from './models/sai-app';
5
6 You, last month | 1 author (You)
7 @Controller()
8 export class AppController {
9   constructor(private readonly appService: AppService) {}
10
11   @Get()
12   checkServer(): string {
13     return this.appService.checkServer();
14   }
15
16   @Post('app')
17   createSaiApp(@Body() saiApp: SaiApp): void {
18     this.appService.createApp(saiApp);
19   }
20
21   @Post('privacy-policy')
22   createPrivacyPolicy(@Body() privacyPolicy: PrivacyPolicy): void {
23     this.appService.createPrivacyPolicy(privacyPolicy);
24   }
25
26   @Get('app')
27   async getSaiApp(): Promise<SaiApp> {
28     return await this.appService.getSaiApp();
29   }
30
31   @Get('privacy-policy')
32   async getPrivacyPolicy(): Promise<PrivacyPolicy> {
33     return await this.appService.getPrivacyPolicy();
34   }
35 }
```

app.controller

Yukarıdaki app.controller'da şu işlemler gerçekleşiyor:

- **Get.checkServer:** Sunucunun çalışıp çalışmadığı kontrol ediliyor.
- **Post.createSaiApp:** Yeni bir app modeli oluşturma isteği.
- **Post.privacyPolicy:** Yeni bir gizlilik politikası oluşturma isteği.
- **Get.app:** Aktif app modeli döndürür.
- **Get.privacyPolicy:** Aktif gizlilik politikasını döndürür.

## 2. Giriş yap ve Kaydol

Bu bölümde kullanıcının sisteme kayıt olması için gerekli olan geliştirmeler yapılmıştır.

```
model User {
    id          String      @id @unique @default(uuid())
    email       String      @unique
    username    String      @unique
    password    String
    token       String?    @unique
    registerDate DateTime? @default(now())
    isNotifyActive Boolean? @default(true)
    theFirstDayOfWeek String? @default("monday")
    trainingDays Int?     @default(3)
    isAccountActive Boolean   @default(false)
    profilePhoto String?

    personalInformation PersonalInformation?
    userHistories      UserHistory[]
    posts              Post[]
    postComments        PostComment[]
    userActivities     UserActivity[]
    UserAnsweredPool   UserAnsweredPool[]
}
```

schema.prisma

Yukarıda yer alan model kullanıcıya aittir. Burada yer alan verilerin anlamları şu şekildedir:

- **token:** 60 karakter uzunluğunda, kullanıcıya özgü oluşturulan, auterization kontrolü için gerekli olan veri
- **isNotifyActive:** Uygulama içeresine bildirim alıpmadığının verisini tutar.
- **trainingDays:** Haftada kaç gün spor yapacağı.
- **profilePhoto:** AWS S3'e kayttığımız profil fotoğrafının URL'i

```

21  @Controller('auth')
22  export class AuthController {
23      constructor(private authService: AuthService) {}
24
25      @Public()
26      @Post('sendActivationCode')
27      @HttpCode(HttpStatus.CREATED)
28      @ApiCreatedResponse({ type: User || HttpException })
29      sendActivationCode(
30          @Query('username') username: string,
31          @Query('email') email: string,
32      ): Promise<any> {
33          console.log(username);
34          console.log(email);
35
36          return this.authService.sendActivationCode(username, email);
37      }
38
39      @Public()
40      @Post('signup')
41      signup(@Body() dto: any): Promise<{ token: string } | HttpException> {
42          return this.authService.signup(dto);
43      }
44
45      @Public()
46      @Patch('signin')
47      @HttpCode(HttpStatus.OK)
48      @ApiCreatedResponse({ type: TokenData || HttpException })
49      signin(@Body() dto: SignInDto): Promise<TokenData | HttpException> {
50          return this.authService.signin(dto);
51      }
52
53      @UseGuards(AtGuard)
54      @Patch('logout')
55      @HttpCode(HttpStatus.OK)
56      logout(@Query() query: { userId: string }): Promise<any> {
57          return this.authService.logout(query.userId);
58      }
59

```

auth.controller.ts

Yukarıdaki auth.controller'da şu işlemler gerçekleşiyor:

#### **Post.sendActivationCode**

Kullanıcının malinin doğruluğunu kontrol etmek için input alanına girdiği maile 4 haneli aktivasyon kodunu gönderir. Bu işlem sahte hesapların oluşumunu engelleyecektir. AuthService'de sendActivationCode işlevi şu şekildedir:

```

29
30  async sendActivationCode(username: string, email: string): Promise<any> {
31    // check username
32    const usernameCheck: User = await this.prisma.user.findUnique({
33      where: { username: username },
34    });
35    const emailCheck: User = await this.prisma.user.findUnique({
36      where: { email: email },
37    });
38    if (usernameCheck === null) {
39      throw new HttpException(
40        'There is a user with this username. Please change your username.',
41        HttpStatus.CONFLICT,
42      );
43    } else if (emailCheck === null) {
44      throw new HttpException(
45        'There is a user with this email. Please change your email.',
46        HttpStatus.CONFLICT,
47      );
48    } else {
49      const activationCode = this.generateActivationCode();
50      this.mailerService
51        .sendMail({
52          to: email,
53          from: 'sportwithai@gmail.com',
54          subject: 'Testing Nest MailerModule ✓',
55          html: activation_code_email_template(activationCode),
56        })
57        .then((success) => {
58          console.log(success);
59        })
60        .catch((err) => {
61          console.log(err);
62        });
63
64      console.log('activationCode: ', activationCode);
65      return {
66        activationCode: activationCode,
67      };
68    }
69  }

```

auth.service.ts - sendActivationCode fonksiyonu

Bu kısımda eğer böyle bir kullanıcı yoksa hata mesajı dönüyor. Eğer varsa böyle bir maile sahip hesabın varlığını sorguluyor. Eğer yoksa mailin hatalı olduğunun hatasını döndürüyor. Eğer bu aşamaya da geçerse, 4 haneli bir aktivasyon kodu yaratılıp bir mail templatemin içeresine koyulup ilgili maile gönderiliyor. Mailin görüntüsü şu şekildedir:



**Hello there,**

If you enter the 4-digit activation code below into the input field within the application, your account will be securely saved to the system..

6290

*Sport With AI (SWAI)*

Aktivasyon Kodu Maili

**Post.signup**

Kullanıcının uygulama kayıt olması için oluşturulan Post endpointi. auth.service’te çalıştırılan işlevi:

```

71  async signup(
72    userDto: User | PersonalInformation | any,
73  ): Promise<{ token: string } | HttpException> {
74    const password = await this.hashPassword(userDto.password);
75    try {
76      const newUser: any = await this.prisma.user.create({
77        data: {
78          email: userDto.email,
79          username: userDto.username,
80          password,
81          personalInformation: {
82            create: {
83              birthDay: userDto.birthDay,
84              weight: userDto.weight,
85              height: userDto.height,
86              goal: userDto.goal,
87            },
88          },
89        },
90        include: {
91          personalInformation: true,
92        },
93      });
94      const tokenData = await this.getToken(newUser.id, newUser.email);
95      newUser.token = tokenData.token;
96      await this.updateToken(newUser.id, tokenData.token);
97      return { token: tokenData.token };
98    } catch (error) {
99      console.log('error: ', error);
100     return new HttpException(
101       'There is a user with this email address or username. Please enter different values.',
102       HttpStatus.CONFLICT,
103     );
104   }
105 }

```

auth.service.ts - signup fonksiyonu

Veritabanına kullanıcının şifresini olduğu gibi kaydetmiyor. Olduğu gibi bir String şeklinde kaydetmek kullanıcı gizliliği açısından hiç güvenli değil. Bu sebeple 74'ü satırda kullanıcının girdiği şifreyi kriptoloji yardımıyla şifreliyoruz. Örneğin 12 haneli bir şifre 140 haneli bir String haline geliyor. Bu sayede kullanıcı güvenliğini sağlamış oluyoruz.

Fonksiyonun devamında, oluşturulan kullanıcının id'si ve mailini baz alarak bir token oluşturuyoruz ve daha sonra bu token'ı kullanıcının veritabanına ekliyoruz.

Bir sorun olması halinde 100'ü satırda yer alan hata mesajını gönderiyoruz.

### **Post.signup**

Kullanıcısını mailini ve parolasını göndererek giriş yapılmasını sağlayan Patch endpointi.

```

async signin(dto: SignInDto): Promise<TokenData | HttpException> {
  let user: User;
  if (dto.email) {
    user = await this.prisma.user.findUnique({
      where: { email: dto.email },
    });
  } else if (dto.username) {
    user = await this.prisma.user.findUnique({
      where: { username: dto.username },
    });
  } else {
    return new HttpException(
      'There is a problem. Please try to login again later.',
      HttpStatus.NOT_FOUND,
    );
  }
}

```

auth.service.ts - signin fonksiyonu

### Post.logout

Kullanıcının uygulamadan çıkış yapmasını sağlayan Patch endpointi. Bu endpointte kullanıcının token verisi silinir. Bu sayede uygulama içerisinde authorization işlemi yapılmaz.

### 3. Sosyal Medya

Uygulama içerisinde yer alan sosyal medyada; yazı, anket ve görsel olarak üç farklı gönderi paylaşılabilir. Bunun haricinde paylaşılan içerik beğenilebiliyor ve yorum eklenebiliyor.

Sosyal medyada yer alan tüm postları çekmek için şu endpoint yaratıldı:

```

@Get('post/all')
@UseGuards(RtGuard)
getAllPost(@Req() request): Promise<PostI[] | HttpException> {
  return this.socialMediaService.getAllPosts(request.user.sub);
}

```

social-media.controller.ts

Sosyal medyada, kullanıcının kendi gönderilerini görmek için oluşturulan endpoint:

```

@Get('post/user/:userId')
@UseGuards(RtGuard)
getPostFromUser(
  @Param('userId') userId: string,
): Promise<PostI[] | HttpException> {
  if (!userId) {
    throw new HttpException(
      'There is a problem. Please try again later what you want to do..',
      HttpStatus.NOT_IMPLEMENTED,
    );
  }
  return this.socialMediaService.getPostFromUser(userId);
}

```

social-media.controller.ts

Paylaşılan gönderiyi silmek için oluşturulan endpoint (bu endpoint sadece adminler tarafından çalıştırılmaktadır):

```
@Delete('del/:postId')
@UseGuards(RtGuard)
deletePost(@Param('postId') postId: string): void {
  this.socialMediaService.deletePost(postId);
}
```

social-media.controller.ts

Yazı, görsel ve anket paylaşmak için yaratılan endpointler:

```
@Post('share/text')
@UseGuards(RtGuard)
shareText(@Body() postI: PostI): Promise<PostI | HttpException> {
  return this.socialMediaService.createPostText(postI);
}

@Post('share/image')
@UseGuards(RtGuard)
shareImage(
  @Req() request,
  @Body() body: any,
): Promise<PostI | HttpException> {
  return this.socialMediaService.createPostImage({
    userId: request.user.sub,
    content: body.content,
    imageURL: body.imageURL,
  });
}

@Post('upload/image')
@UseGuards(RtGuard)
@UseInterceptors(FileInterceptor('file'))
uploadImage(@UploadedFile('file') file) {
  return this.socialMediaService.uploadImage(file);
}

@Post('share/pool')
@UseGuards(RtGuard)
sharePool(@Body() postI: PostI): Promise<PostI | HttpException> {
  return this.socialMediaService.createPostPool(postI);
}
```

social-media.controller.ts

Burada 4 farklı endpoint yer almaktadır:

- **share/text:** Yazı olarak gönderi paylaşma endpointi..
- **share/image:** Görsel olarak gönderi paylaşma endpointi. Fakat burada öncelikle aşağısında yer alan upload/image endpointı çalıştırılacaktır. Bu endpoint ilgili fotoğrafı AWS S3'ün social-media klasörüne yükleyecektir. Ardından yüklenen lokasyonun URL'sini ön uc'a döndürecektr. Ön ucta alınan bu URL share/image'ın body kısmına eklenecektir.
- **share/pool:** anket paylaşma endpointi.

Yorum ile ilgili endpointler:

```
@Get('comments/:postId')
@UseGuards(RtGuard)
getPostComments(
  @Param('postId') postId: string,
): Promise<PostComment[] | HttpException> {
  if (!postId) {
    throw new HttpException(
      'There is a problem. Please try again later what you want to do..',
      HttpStatus.NOT_IMPLEMENTED,
    );
  }
  return this.socialMediaService.getPostComments(postId);
}

@Delete('comments/:commentId')
@UseGuards(RtGuard)
deletePostComments(@Param('commentId') commentId: string): void {
  if (!commentId) {
    throw new HttpException(
      'There is a problem. Please try again later what you want to do..',
      HttpStatus.NOT_IMPLEMENTED,
    );
  }
  this.socialMediaService.deleteComment(commentId);
}

@Get('comment/like')
@UseGuards(RtGuard)
likeForPostComment(
  @Query('postCommentId') postCommentId: string,
  @Query('userId') userId: string,
): void {
  if (!postCommentId || !userId) {
    throw new HttpException(
      'There is a problem. Please try again later what you want to do..',
      HttpStatus.NOT_IMPLEMENTED,
    );
  }
  this.socialMediaService.likeForPostComment(postCommentId, userId);
}
```

social-media.controller.ts

Burada 3 farklı endpoint yer almaktadır:

- **comments/postId:** İlgili post'a yorum yapma endpointi.
- **comment/commentId:** commentId değeri ile eşleşen yorumu silme endpointi.
- **comment/like:** commentId değeri ile eşleşen yorumu beğenme endpointi.

```
@Post('pool/answer')
@UseGuards(RtGuard)
answerPool(
  @Req() request,
  @Query('poolId') poolId: string,
  @Query('answer') answer: number,
): Promise<Pool | HttpException> {
  return this.socialMediaService.answerPool(poolId, answer, request.user.sub);
}
```

social-media.controller.ts

Kullanıcının ankete cevap vermesi sonucu çalıştırılan endpoint. Burada istenilen değerler anketin id'si ve cevabın index değeri (0,1,2).

#### 4. Kullanıcı

Kullanıcıları ön ucta yer alan listeye çekmek, kullanıcının bilgilerini değiştirmek, kullanıcıyı silmek vb. işlemleri gerçekleştiren bölüm.

```
21  @Get('user/:userId')
22  @UseGuards(RtGuard)
23  getUser(@Param('userId') userId: string): Promise<User | HttpException> {
24    return this.userService.getUser(userId);
25  }
26
27  @Patch('user/:userId')
28  @UseGuards(RtGuard)
29  updateUser(
30    @Param('userId') userId: string,
31    @Body() userBody: any,
32  ): Promise<User | HttpException> {
33    return this.userService.updateUserById(userId, userBody);
34  }
35
36  @Get('user/personal-information/:userId')
37  @UseGuards(RtGuard)
38  getUserPersonalInformation(
39    @Param('userId') userId: string,
40  ): Promise<PersonalInformation | HttpException> {
41    return this.userService.getUserPersonalInformation(userId);
42  }
43
44  @Post('user/personal-information/:userId')
45  @UseGuards(RtGuard)
46  createUserPersonalInformation(
47    @Param('userId') userId: string,
48    @Body() personalIBody: PersonalInformation,
49  ): Promise<PersonalInformation | HttpException> {
50    personalIBody.userId = userId;
51    return this.userService.createUserPersonalInformation(personalIBody);
52  }
53
54  @Patch('user/personal-information')
55  @UseGuards(RtGuard)
56  updateUserPersonalInformation(
57    @Body() personalIBody: PersonalInformation,
58  ): Promise<PersonalInformation | HttpException> {
59    return this.userService.updateUserPersonalInformation(personalIBody);
60  }
```

user.controller.ts

Burada 4 farklı endpoint yer almaktadır:

- **Get:user/:userId** : UserId değerine karşılık gelen kullanıcıyı döndürür.
- **Patch:user/:userId** : UserId değerine karşılık gelen verileri günceller.

- **Get:user/personal-information/:userId** : UserId değerine karşılık gelen kişisel bilgileri döndürür. Bu ön uç tarafında ayarlar kısmın içerisinde yer alan profilimi düzenle bölümünde yer alır.
- **Patch:user/personal-information** : UserId değerine karşılık gelen kullanıcının kişisel bilgilerini günceller.

```

62  @Post('user/history/:userId')
63  @UseGuards(RtGuard)
64  createUserHistory(
65    @Param('userId') userId: string,
66    @Body() userHistory: UserHistory,
67  ): Promise<UserHistory | HttpException> {
68    console.log('userId: ', userId);
69    console.log('userHistory: ', userHistory);
70    userHistory.userId = userId;
71    return this.userService.createUserHistory(userHistory);
72  }
73
74  @Get('user/history/:userId')
75  @UseGuards(RtGuard)
76  getUserHistory(
77    @Param('userId') userId: string,
78  ): Promise<UserHistory[] | HttpException> {
79    return this.userService.getUserHistory(userId);
80  }
81
82  @Get('users')
83  @UseGuards(RtGuard)
84  getAllUsers() {
85    return this.userService.getAllUsers();
86  }
87
88  @Post('user/add')
89  @UseGuards(RtGuard)
90  addUserFromAdmin(@Body() userBody: User): Promise<boolean> {
91    return this.userService.addUser(userBody);
92  }
93
94  @Delete('user/:userId')
95  @UseGuards(RtGuard)
96  deleteUser(@Param('userId') userId: string): void {
97    this.userService.deleteUser(userId.toString());
98  }
99
100 @Get('getmyprofile')
101 @UseGuards(RtGuard)
102 getMyProfile(@Req() request): Promise<any> {
103   return this.userService.getMyProfile(request.user.sub);
104 }
```

user.controller.ts

Burada 6 farklı endpoint yer almaktadır:

- **Post:user/history/:userId** : Kullanıcı bir değişiklik yaptığında çalıştırılacak endpoint. Bir nevi Loglama sistemi gibi.
- **Get:user/history/:userId** : Kullanıcıya ait işlem geçmişini (yaptığı değişiklikleri) döndürür.
- **Get:users** : Bütün kullanıcıları döndürür (sadece admin tarafından çalışıyor).
- **Post:user/add** : Yeni bir kullanıcı ekleme endpointı (sadece admin tarafından çalışıyor).
- **Delete:user/:userId** : UserId'e değerine karşılık gelen kullanıcıyı silme endpointı (sadece admin tarafından çalışıyor).
- **Get:getmyprofile** : Tokeni baz alarak o token'e ait kullanıcı verisi döndüren endpoint.

## 5. Egzersiz

Bu bölümde uygulamanın temel hedefi olan egzersiz yapma endpointler yer almaktadır. Bu egzersizi kullanıcı ister yapay zeka ile birlikte ister normal bir spor uygulamasında ki gibi manuel olarak gerçekleştirebilir.

Bu alanda 4 farklı modelimiz var. Bu 4 modelin SQL modeli olarak gösterimi şu şekildedir:

```

138  model Workout {
139    id          String      @id @unique @default(uuid())
140    creationDate DateTime   @default(now())
141    name        String
142    kind        String
143    imageURL   String
144    information String
145    activityDays ActivityDay[]
146    userActivities UserActivity[]
147  }

```

schema.prisma - workout

Workout en büyük egzersiz modelimiz. Sırt egzersizi, omuz egzersizi, zayıflama egzersizi vb değerlerler alır. ActivityDays ile one-to-many ilişkisi vardır. ActivityDays dizisinin uzunluğu kadar workout günü vardır. ActivityDay modeli şu şekildedir:

```

149  model ActivityDay {
150    id          String      @id @unique @default(uuid())
151    creationDate DateTime   @default(now())
152    workoutId   String
153    workout     Workout     @relation(fields: [workoutId], references: [id], onDelete: Cascade)
154    day         Int
155    activityIds String[]
156  }

```

schema.prisma - ActivityDay

ActivityDay içerişine day (1,2,3 ...) ve aktiviteleri alır. Mekik, şınav gibi türler activityIds bölümüne eklenir. Activity modeli ise şu yapıdadır:

```

157
158   model Activity {
159     id           String  @id @unique @default(uuid())
160     creationDate DateTime @default(now())
161     name         String
162     imageURL    String
163     numberOfRepetitions Int
164     isAI         Boolean @default(false)
165   }

```

schema.prisma - Activity

**numberOfRepetitions** tekrar sayısını, **isAI** ise bu spor aktivitesinin yapay zeka ile yapılmış olup olmadığına verisini tutar.

```

26   @Get('workouts')
27   @UseGuards(RtGuard)
28   getWorkouts(): Promise<Workout[] | HttpException> {
29     return this.workoutService.getWorkouts();
30   }
31
32   @Get('workout/:id')
33   @UseGuards(RtGuard)
34   getWorkout(@Param('id') id: string): Promise<Workout | HttpException> {
35     return this.workoutService.getWorkout(id);
36   }
37
38   @Post('workout/upload/image')
39   @UseGuards(RtGuard)
40   @UseInterceptors(FileInterceptor('file'))
41   uploadImage(@UploadedFile('file') file) {
42     return this.workoutService.uploadCoverImage(file);
43   }
44
45   @Post('workout')
46   @UseGuards(RtGuard)
47   createUserPersonalInformation(
48     @Body() workoutBody: Workout,
49   ): Promise<Workout | HttpException> {
50     return this.workoutService.createWorkout(workoutBody);
51   }
52
53   @Patch('workout')
54   @UseGuards(RtGuard)
55   updateWorkout(
56     @Body() workoutBody: Workout,
57   ): Promise<Workout | HttpException> {
58     return this.workoutService.updateWorkout(workoutBody);
59   }
60
61   @Delete('workout/:id')
62   @UseGuards(RtGuard)
63   deleteWorkout(@Param('id') id: string): Promise<boolean> {
64     return this.workoutService.deleteWorkout(id);
65   }
66

```

workout.controller.ts

Burada 6 farklı endpoint yer almaktadır:

- **Get workouts** : Uygulama içerisinde yer alan tüm egzersizleri döndürür.

- **Get workout/:id** : Uygulama içerisinde id değerine sahip egzersizi döndürür.
- **Post workout/upload/image** : Egzersizin kapak fotoğrafını AWS S3'e yükler.
- **Post workout** : Yeni bir egzersiz oluşturur.
- **Patch workout** : Var olan egzersizi günceller.
- **Delete workout** : Var olan egzersizi siler.

Yukarıda yer alan Get workouts dışındaki tüm endpointler Adminler tarafından yapılabilmektedir.

```

101  @Get('activity/:id')
102  @UseGuards(RtGuard)
103  getActivity(@Param('id') id: string): Promise<Activity | HttpException> {
104    return this.workoutService.getActivity(id);
105  }
106
107  @Post('activity')
108  @UseGuards(RtGuard)
109  createActivity(
110    @Body() activityBody: Activity,
111 ): Promise<Activity | HttpException> {
112   return this.workoutService.createActivity(activityBody);
113 }
114
115  @Patch('activity')
116  @UseGuards(RtGuard)
117  updateActivity(
118    @Body() activityBody: Activity,
119 ): Promise<Activity | HttpException> {
120   return this.workoutService.updateActivity(activityBody);
121 }
122
123  @Delete('activity/:id')
124  @UseGuards(RtGuard)
125  deleteActivity(@Param('id') id: string): Promise<boolean> {
126    return this.workoutService.deleteActivity(id);
127  }
128

```

workout.controller.ts

Burada 4 farklı endpoint yer almaktadır:

- **Get activity/:id** : Uygulama içerisinde id değerine sahip aktiviteyi döndürür.
- **Post activity**: Yeni bir aktivite oluşturur.
- **Patch activity** : Var olan aktiviteyi günceller.
- **Delete activity**: Var olan aktiviteyi siler.

```

68  @Get('activity-day/:id')
69  @UseGuards(RtGuard)
70  getActivityDay(
71    @Param('id') id: string,
72  ): Promise<ActivityDay | HttpException> {
73    return this.workoutService.getActivityDay(id);
74  }
75
76  @Post('activity-day')
77  @UseGuards(RtGuard)
78  createActivityDay(
79    @Query('workoutId') workoutId: string,
80  ): Promise<ActivityDay | HttpException> {
81    return this.workoutService.createActivityDay(workoutId);
82  }
83
84  @Patch('activity-day')
85  @UseGuards(RtGuard)
86  updateActivityDay(
87    @Body() activityDayBody: ActivityDay,
88  ): Promise<ActivityDay | HttpException> {
89    return this.workoutService.updateActivityDay(activityDayBody);
90  }
91
92  @Delete('activity-day')
93  @UseGuards(RtGuard)
94  deleteActivityDay(
95    @Query('workoutId') workoutId: string,
96  ): Promise<HttpException> {
97    return this.workoutService.deleteActivityDay(workoutId);
98  }
99

```

workout.controller.ts

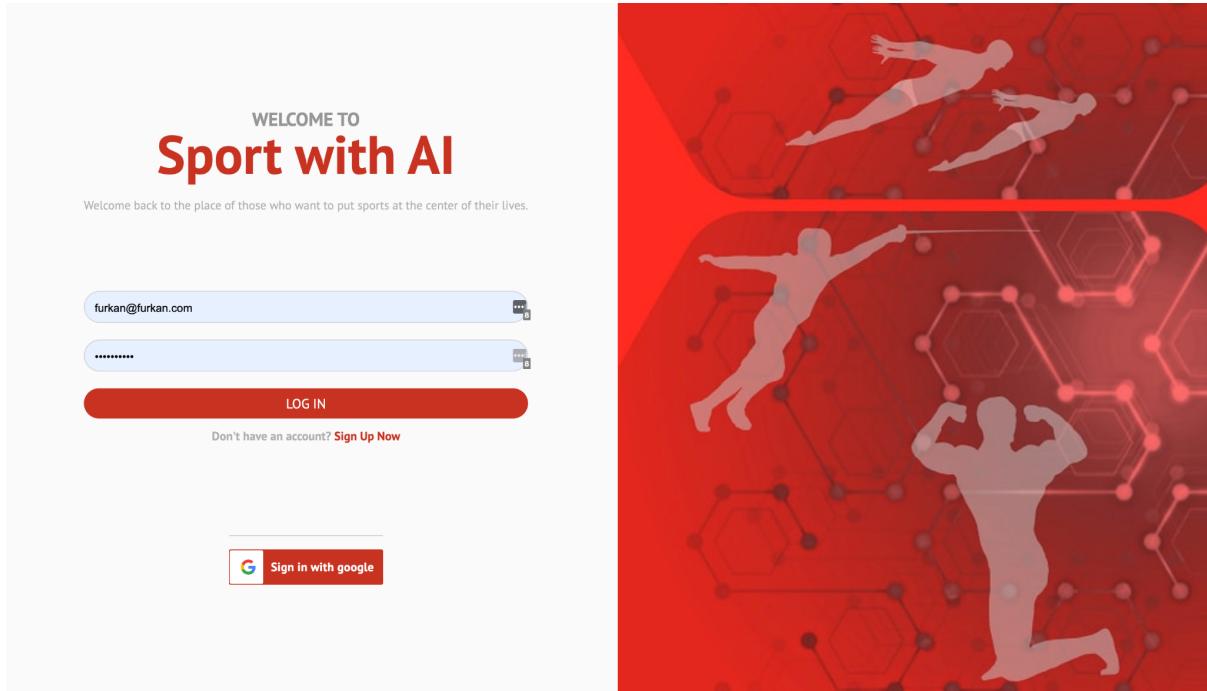
Burada 4 farklı endpoint yer almaktadır:

- **Get activity-day:id** : Uygulama içerisinde id değerine sahip aktiviteyi gününü döndürür.
- **Post activity-day**: Yeni bir aktivite günü oluşturur.
- **Patch activity-day** : Var olan aktivite gününü günceller.
- **Delete activity-day**: Var olan aktivite gününü siler.

## Ön Uç Geliştirme - Admin Sayfası (Muhammed Furkan Gülşen)

Adminlerin uygulama üzerinde tam kontrol sağlayabilmesi için bir Admin Dashboard geliştirilmiştir.

Uygulamanın giriş sayfası şu şekilde görünmektedir:



Ön Uç Giriş Ekranı

Bu giriş sayfası daha sonra kullanıcı giriş sayfası içinde kullanılacaktır.  
Uygulamaya giriş yapıldıktan sonra aşağıdaki sayfa açılacaktır.

The screenshot shows two main parts. On the left is the 'SAI Admin Dashboard' with a dark theme. It has a sidebar with 'Home', 'Social Media' (which is highlighted), 'Sport', and 'Users'. At the bottom is a red 'LOGOUT' button. On the right is a social media post from a user named 'furkan' dated 'Apr 7, 2022'. The post content is 'bu ikinci denemem' and includes a hand-drawn diagram illustrating a database schema with entities like 'Workout', 'Activity', 'Activity Day', and 'User Activity'. The diagram shows relationships between these entities, such as 'Workout' having many 'Activity' and 'Activity Day' having many 'User Activity'. Below the diagram is a photo of a person working out. At the bottom of the post are interaction icons for comments and likes.

Ön Uç Sosyal Medya Ekranı

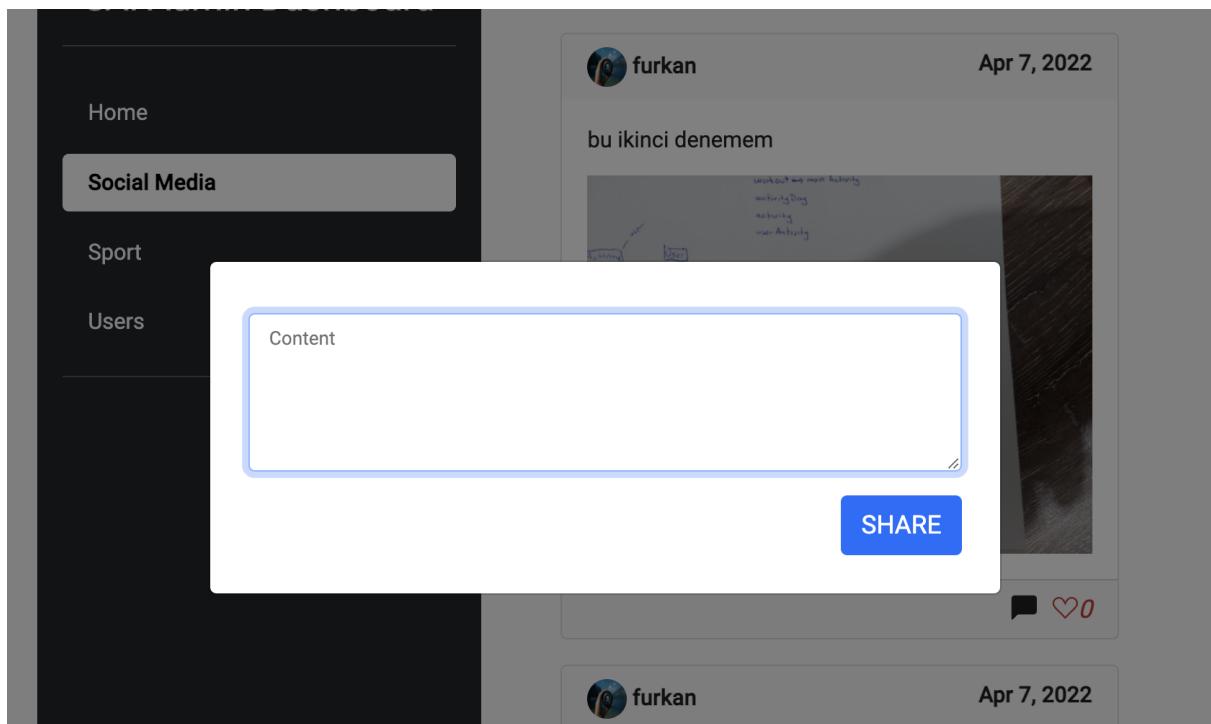
Sol kısmında bir sidebar menümüz vardır. Burada Admin yapmak istediği işleme göre gitmek istediği sayfayı seçebilmektedir. En alt bölümde yer alan logout butonu ile de uygulamadan çıkış yapabilmektedir.

Yukarıda yer alan ekran görüntüsü sosyal medya ekranına aittir. Burada Admin gönderileri takip edebilir, gönderi paylaşabilir, gönderiyi beğenebilir veya gönderiye yorum yapabilir.



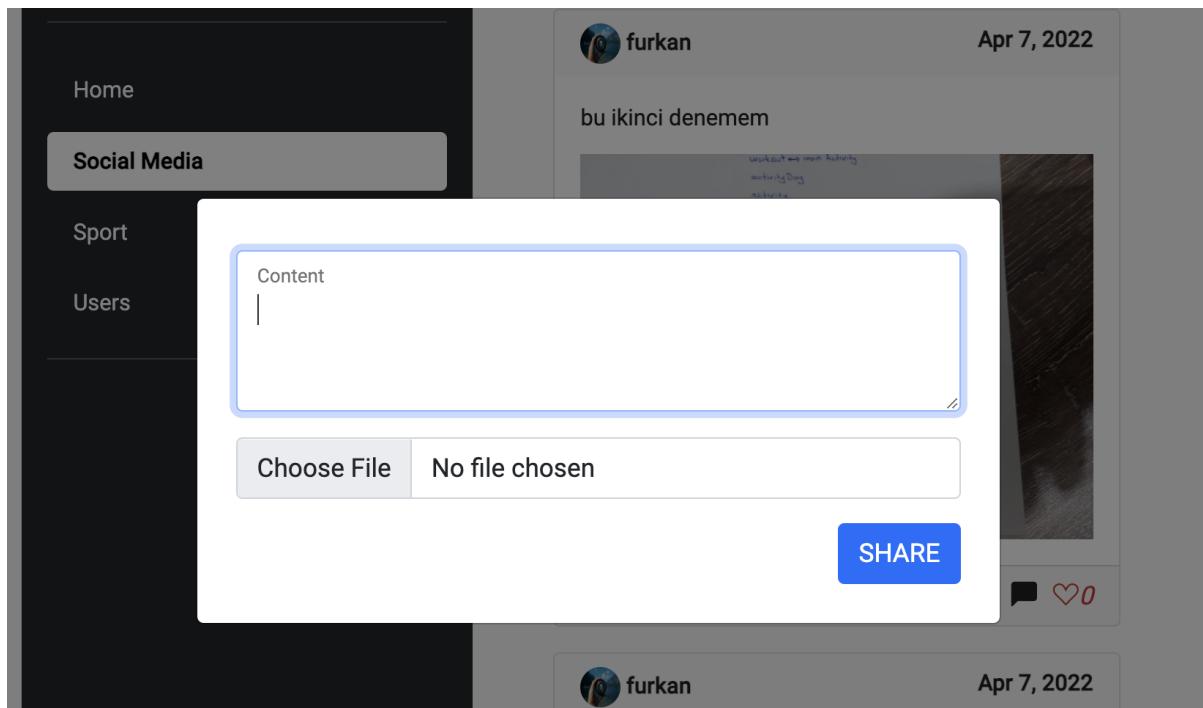
Ön Uç Gönderi Paylaşma Butonu

Yeni bir yazı gönderisi paylaşmak istenirse create post alanına tıklanılır. Bir görsel paylaşmak isteniyor ise görsel ikonuna, bir ankter paylaşmak isteniyor ise anket ikonuna tıklanılır.



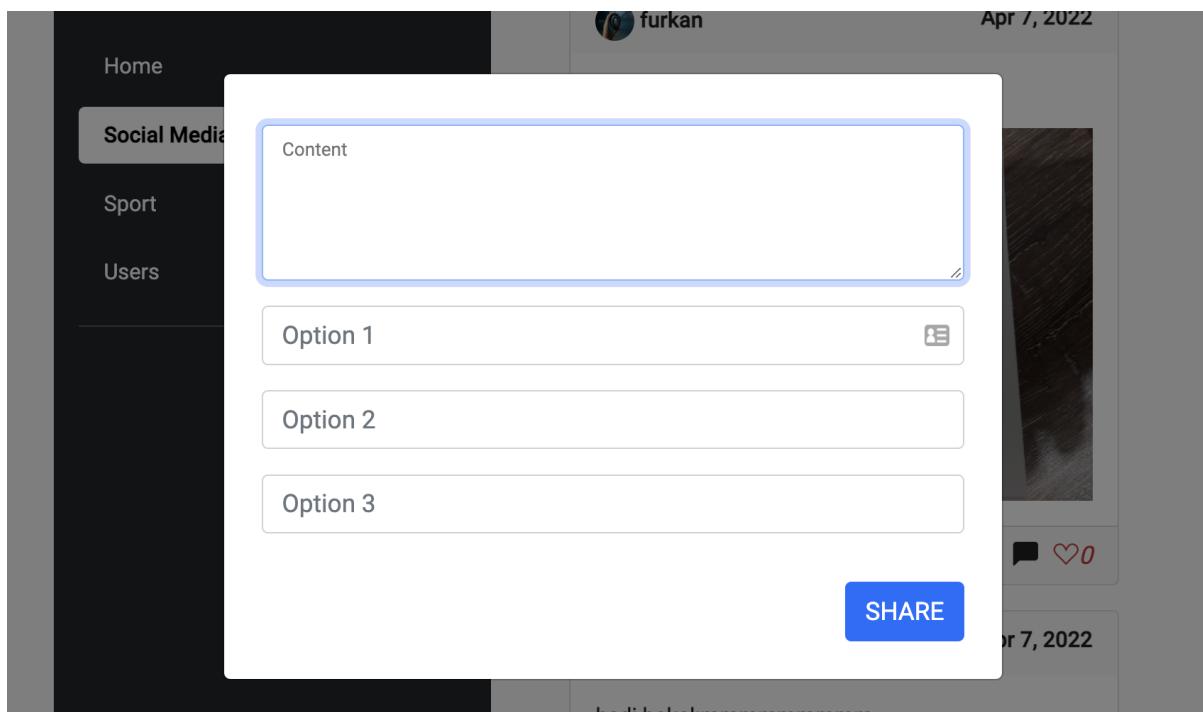
Ön Uç Yazı Gönderisi Paylaşma Açıılır Penceresi

Sosyal medyada bir yazı gönderisi paylaşmak için create post alanına tıklanırsa bir açılır pencere açılır. Burada kullanıcı paylaşmak istediği içeriği girip ardından sağ alt kısımda yer alan Share butonuna tıklayarak gönderisini sosyal medyada paylaşabilir.



Ön Uç Fotoğraf Gönderisi Paylaşma Açıılır Penceresi

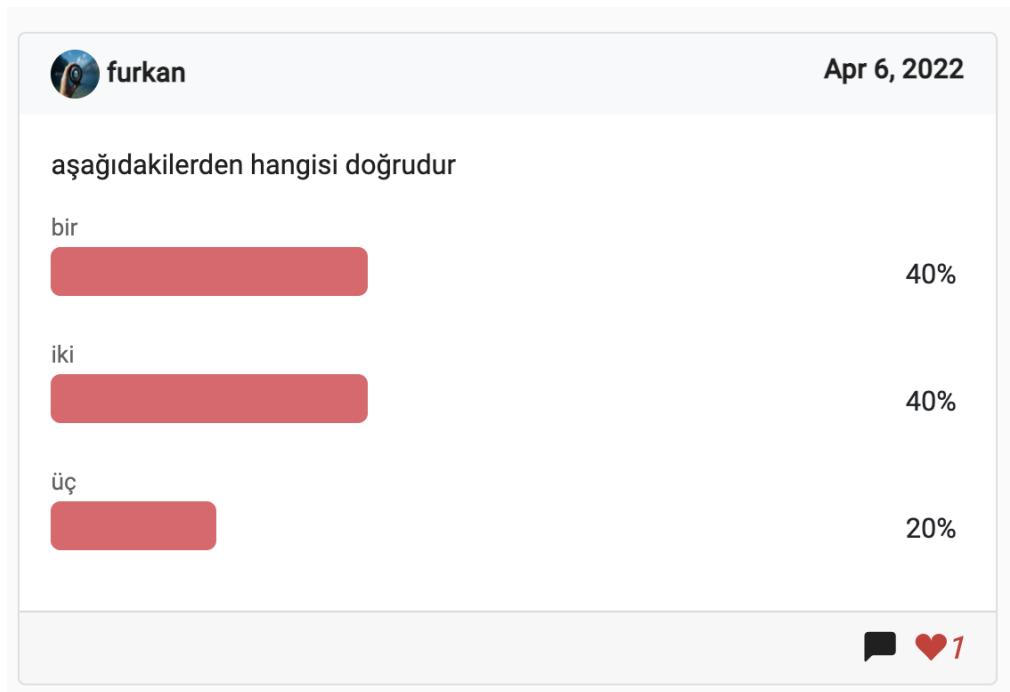
Sosyal medyada bir görsel gönderi paylaşmak için görsel ikon alanına tıklanırsa bir açılır pencere açılır. Burada kullanıcı paylaşmak istediği içeriği girip ve paylaşmak istediği görseli seçtikten sonra sağ alt kısmında yer alan Share butonuna tıklayarak gönderisini sosyal medyada paylaşabilir.



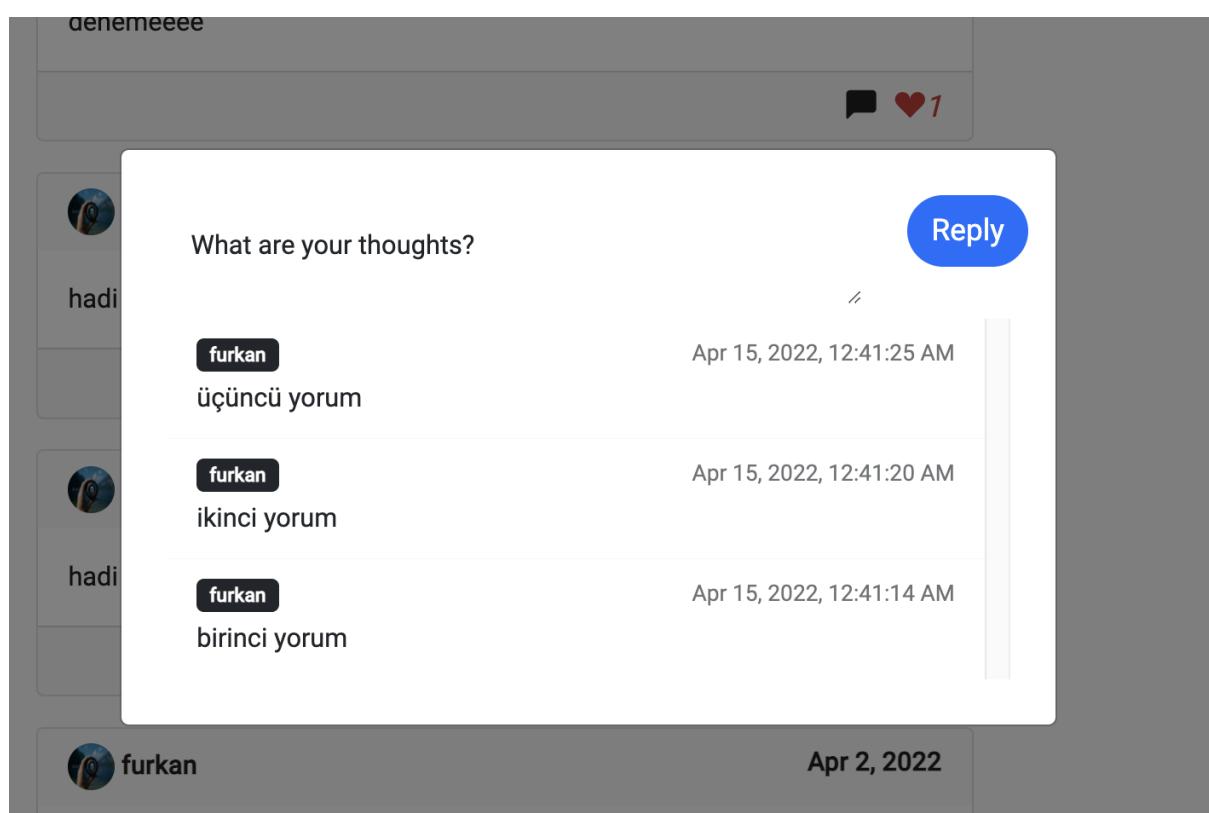
Ön Uç Anket Gönderisi Paylaşma Açıılır Penceresi

Sosyal medyada bir anket paylaşmak istenirse anket ikonuna tıklanır. Anket ikonuna tıklandıktan sonra kullanıcının önüne content ve optionların olduğu bir açılır pencere açılır. Burada kullanıcı

İçeriği girip ardından 3 seçenekten girdikten sonra sağ alt alanda yer alan Share butonuna tıklayarak anketi sosyal medyada paylaşabilir. Anket'e oy verildikten sonra görüntüsü şu şekilde olacaktır:

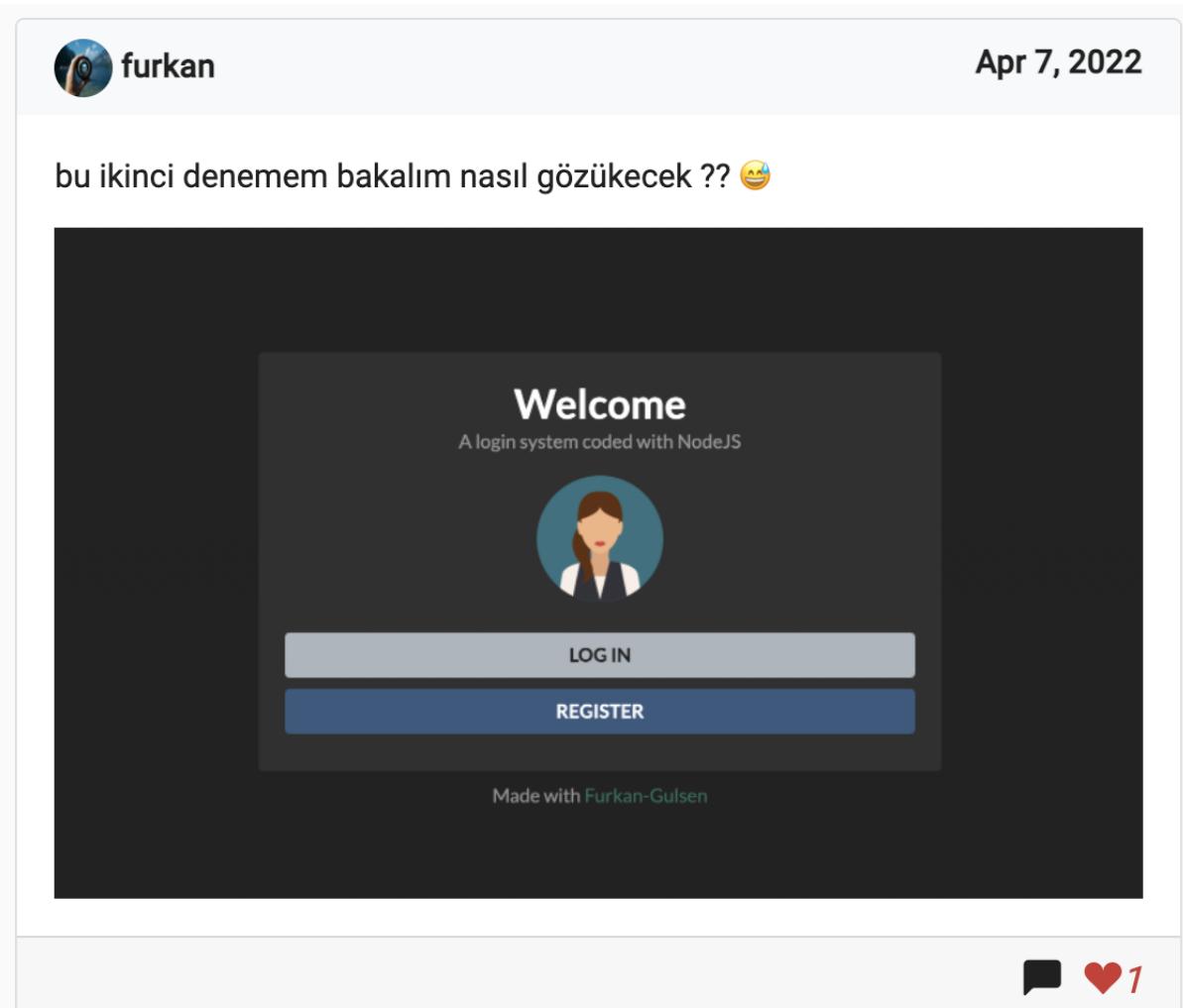


Ön Uç Sosyal Medya Anket Gönderisi



Ön Uç Yazı Yorum Açıılır Penceresi

Bir gönderide yorum yapmak istenirse, gönderi içerisinde yer alan yorum ikonuna tıklanır. Yorum ikonuna tıklandıktan sonra yukarıda yer alan açılıp pencere kullanıcının önüne gelir. Burada o gönderide ait yapılmış yorumları görebilir, üst kısmında yer alan input alanı ile de yeni bir yorum yapabilir.



Ön Uç Sosyal Medya GörSEL Gönderisi

Yukarıda sosyal medyada yer alan bir görsel gönderisini görülmektedir. Sağ alt kısmda yorum ikonu ve kalp butonu vardır. Kullanıcı kalp butonuna basarak gönderiyi beğenebilir, eğer beğendiyise beğenmesini geri alabilir.



**furkan**

Apr 2, 2022

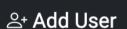
**Lorum Ipsum** is simply dummy text of the printing and typesetting industry. Lorum Ipsum has been the industry's standard dummy text ever since the 1500s,

### Ön Uç Sosyal Medya Yazı Gönderisi

Yukarıda sosyal medyada yer alan bir yazı gönderisini görülmektedir. Sağ alt kısımda yorum ikonu ve kalp butonu vardır. Kullanıcı kalp butonuna basarak gönderiyi beğenebilir, eğer beğendiysse beğenmesini geri alabilir.

### Users



Registration Date	Username	Email	Account Active	Notify Active	Action	Action
Mar 30, 2022	yunustaha3	yunustaha3@gmail.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Mar 30, 2022	yunustaha33	yunustaha214@gmail.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Mar 30, 2022	yunussafdsataha33	yunustahasffasa214s@gmail.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Mar 30, 2022	yunusafssafdsataha33	yunustasfahassffasa214s@gmail.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Apr 2, 2022	furkan	furkan@furkan.com	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		

### Ön Uç Kullanıcı Listesi

Bu sayfada Admin kullanıcıların tamamını görebilir ve yönetebilir. Sol üst kısımda yer alan Add User'a basarak yeni bir kullanıcı ekleyebilir. Bu kullanıcı aktivasyon koduna gerek olmaksızın sisteme direkt olarak kayıt olacaktır. Admin olarak sınırsız bir yetki mevcuttur bu sayfada. Kullanıcılar tablo şeklinde görülmektedir. Kullanıcıyı silmek için kırmızı butona, kullanıcıya dair bilgileri değiştirmek için sarı butona tıklanılır.

### Add User

Email

Username

The First Day Of Week  
Monday

Training Days  
3

Password

is notify activate:

is account active:

**LOG IN**

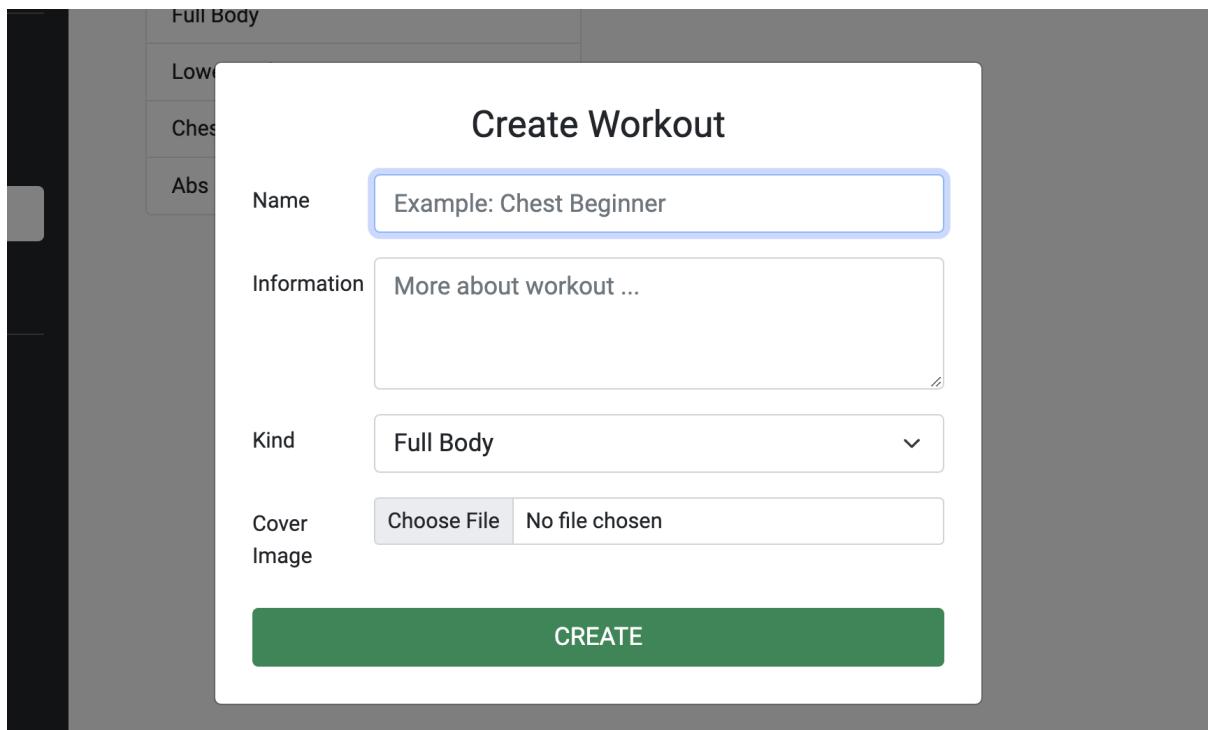
Ön Uç Yazı Kullanıcı Ekleme Açılsır Penceresi

Sağ üste yer alan Add User butonuna tıklandıktan sonra Admin'in bu açılır pencere açılacaktır. Burada gerekli değerleri girerek, Admin direkt olarak sisteme bir kullanıcı ekleyebilmektedir.

Workout	+	Full Body	⚙️	-trash	+
Full Body		1			
Lower Body		2			
Chest		3			
Abs Beginner					

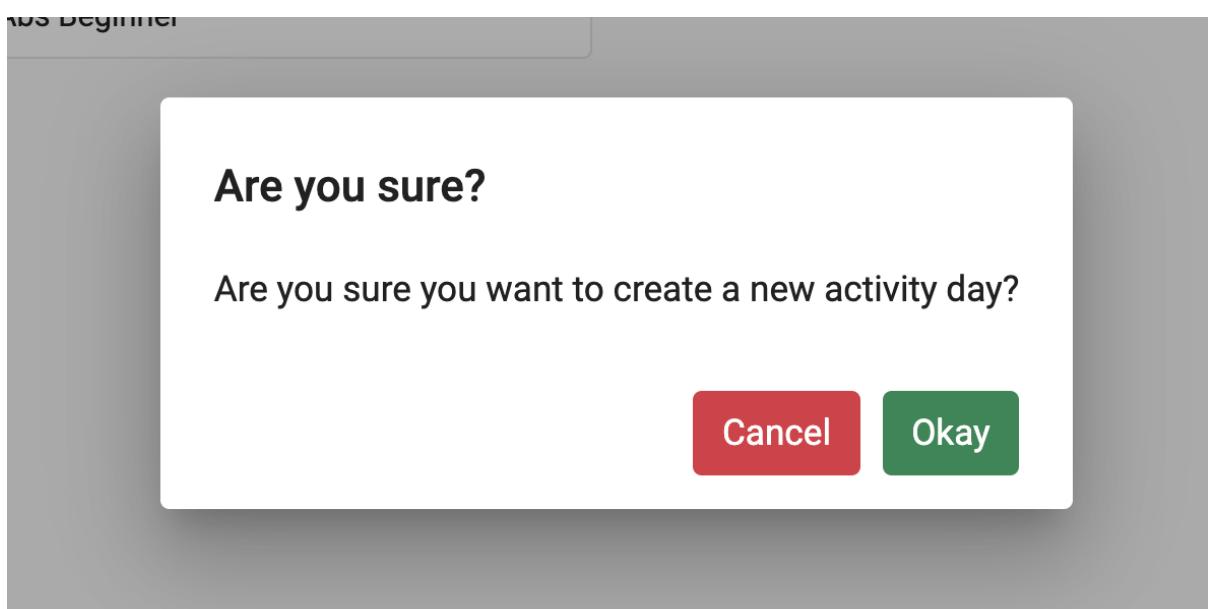
Ön Uç Spor Sayfası

Yukarıda yer alan sayfada, Admin Workout'un sağında bulunan + butonuna tıklayarak yeni bir workout oluşturabilir. Full Body yazılı kısmın sağında yer alan yeşil butona basarak seçilmiş olan workout'a gün ekleyebilir, kırmızı tuşa basarak gün çıkartabilir. Sarı butona basılarak ise seçilmiş olan workout düzenlenebilir.



Ön Uç Egzersiz Ekleme Açılsır Penceresi

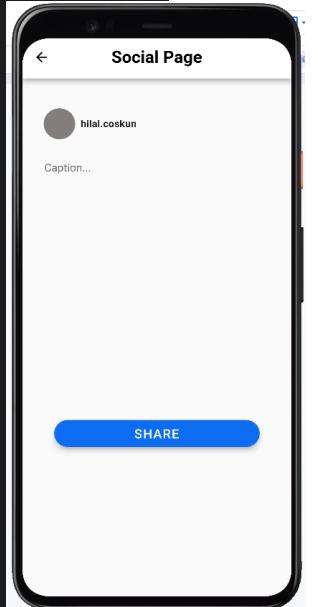
Yukarıdaki yer alan açılır pencerede Admin yeni bir workout oluşturabilir. Workout'ın ismini, ayrıntılı bilgisini, türünü ve en sonda da kapak fotoğrafını girdikten sonra Create butonuna basılması, yeni bir workout oluşturmak için yeterlidir.



Ön Uç İşlem Onaylama Açılsır Penceresi

Full Body'nin yanında yer alan yeşil butona tıklanılırsa yukarıda yer alan emin misin? açılır penceresi açılacaktır. Bu sayede seçilmiş olan Workout'ta yanlışlıkla gün ekleyip/çıkarmanın önüne geçilmiş olacaktır.

## 2.b Mobil Uygulama (Hilal Coşkun)



```
52     ),
53     style: TextStyle(
54       fontWeight: FontWeight.bold, fontSize: 15), // TextStyle
55   ), // Text
56   ],
57   ), // Column
58   ),
59   ) // Expanded
60   ],
61   ), // Row
62   ), // Container
63   Padding(
64     padding: const EdgeInsets.only(left: 5.0, top: 10),
65     child: Row(
66       children: [
67         Expanded(
68           child: Padding(
69             padding: EdgeInsets.only(right: 15.0),
70             child: TextField(
71               controller: textController,
72               style: TextStyle(fontWeight: FontWeight.w400),
73               autofocus: true,
74               cursorColor: Colors.blue,
75               maxLines: 7,
76               decoration: InputDecoration(
77                 hintText: 'Caption...',
78                 focusedBorder: OutlineInputBorder(
79                   borderSide: BorderSide(
80                     color: Color.fromRGBO(0, 238, 232, 232),
81                     width: 4)), // BorderSide // OutlineInputBorder
82                 enabledBorder: OutlineInputBorder(
83                   borderSide: BorderSide(width: 4))), // OutlineInputBorder
84               ),
85             ),
86           ),
87         ],
88       ),
89     ),
89   ),
90   Padding(
91     padding: const EdgeInsets.only(top: 230, right: 15),
92     child: Container(

```

Mobil Uygulama Yazı Gönderi Paylaşım Ekranı - Kodları

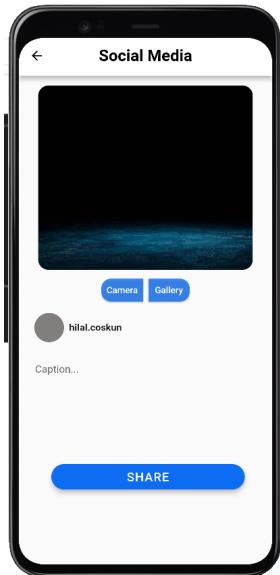
### 1. Sosyal Medya Text(Yazı) Paylaşım Ekranı

Kullanıcı sosyal medyada yazı paylaşımı yapmak istediği Feed Screen(AKİŞ ekranı) da sağ alt kısımda bulunan butondan yazı kısmına tıklayarak bu sayfaya geçiş yapar. Bu sayfayı oluşturma aşamasında kullanıcının metin gireceği alan TextField widget kullanılarak oluşturuldu. Metin içeriği özellikleri ise InputDecoration ile tanımlandı. Share butonu ise FloatingActionButton ile gerekli özellikler verilerek oluşturuldu. Kullanıcının gereklili alanları girdiği bilgiler FloatingActionButton'a ait olan onPressed özelliği ile Backend tarafına istek atılır.

## 2. Sosyal Medya Image(Resim) Paylaşım Ekranı

```
Padding(
  padding: const EdgeInsets.only(top: 15, left: 30, right: 30),
  child: SizedBox(
    height: MediaQuery.of(context).size.height * 0.35,
    width: 400,
    child: ClipRRect(
      borderRadius: BorderRadius.circular(16),
      child: image != null
        ? ImageCard(
            image: image!,
            onClicked: (source) => {pickImage(source)} // ImageCard
        : Image.network(
            'https://cdn.create.vista.com/api/media/small/232751862/stock-photo-dark-b
            fit: BoxFit.cover,
        ), // Image.network
    ), // ClipRRect
  ), // SizedBox
), // Padding
ButtonTheme(
  child: ButtonBar(
    alignment: MainAxisAlignment.center,
    children: <Widget>[
      RaisedButton(
        onPressed: () => pickImage(ImageSource.camera),
        child: const Text('Camera'),
        color: const Color.fromRGBO(193, 12, 108, 242),
        textColor: Colors.white,
        shape: const RoundedRectangleBorder(
          borderRadius: BorderRadius.only(
            bottomLeft: Radius.circular(15.0),
            topLeft: Radius.circular(15.0), // BorderRadius.only
          ), // RoundedRectangleBorder
        ), // RaisedButton
      RaisedButton(
        onPressed: () => pickImage(ImageSource.gallery),
        child: const Text('Gallery'),
        color: const Color.fromRGBO(193, 12, 108, 242),
        textColor: Colors.white,
        //color: Colors.white,
        shape: const RoundedRectangleBorder(
          borderRadius: BorderRadius.only(
            bottomRight: Radius.circular(15.0),
            topRight: Radius.circular(15.0), // BorderRadius.only
          ), // RoundedRectangleBorder
        ), // RaisedButton
    ],
  ),
)
```

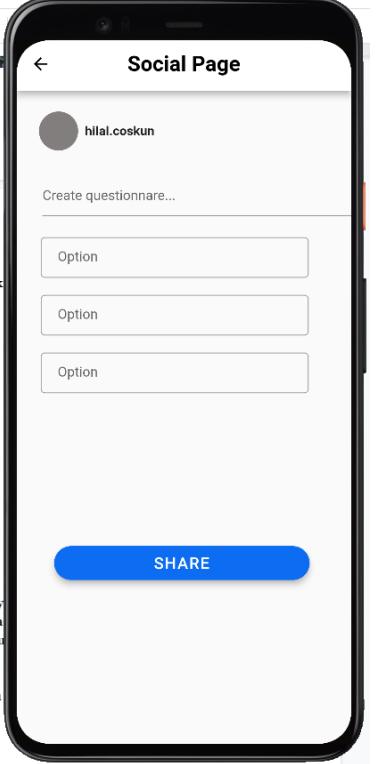
Mobil Uygulama Resim Gönderi Paylaşım Ekranı - Kodları



Kullanıcı sosyal medyada resim paylaşımı yapmak istediğiinde Feed Screen(Akış ekranı) da sağ alt kısımda bulunan butondan resim kısmasına tıklayarak bu sayfaya geçiş yapar. Bu sayfayı oluşturma aşamasında kullanıcının resim eklemesi için iki seçenek sunuyoruz. Bu seçenekler galeri ve kameralıdır. Kullanıcı resmi galeriden seçmek isterse galeri butonuna tıklayarak galeriye girer ve paylaşmak istediği resmi seçer. Eğer fotoğrafı kameraladan çekmek isterse kamera butonuna tıklar ve kamera açılır ardından fotoğrafı çekerek tamam butonuna tıklar. Kullanıcı tekrar bu sayfaya yönlendirilir ve resim pickImage özelliği ile sayfada görüntülenir. Bahsettiğim bu fotoğraf ve görsel ayarlamaları Flutter'a ait ImagePicker özelliği kullanılmıştır. Galeriye ImageSource.gallery ve kameralaya ise Image.camera ile ulaşılır. Daha sonra metin gireceği alan TextField widget kullanılarak oluşturuldu. Metin içeriği özellikleri ise InputDecoration ile tanımlandı. Share butonu ise FloatingActionButton ile gerekli özellikler verilerek oluşturuldu. Kullanıcının gerekli alanları

girdiği bilgiler FloatingActionButton'a ait olan onPressed özelliği ile Backend tarafına istek atılır.

### 3. Sosyal Medya Pool(Anket) Paylaşım Ekranı



```
), // SizedBox
    ), // Padding
    Padding(
      padding: const EdgeInsets.only(left: 18.0, right: 50, top: 25),
      child: Form(
        child: Column(
          children: [
            QuestionnaireFormCard(controller: options1Controller),
            const SizedBox(
              height: 20,
            ), // SizedBox
            QuestionnaireFormCard(
              controller: options2Controller,
            ), // QuestionnaireFormCard
            const SizedBox(
              height: 20,
            ), // SizedBox
            QuestionnaireFormCard(
              controller: options3Controller,
            ), // QuestionnaireFormCard
          ],
        ),
      ), // Column
    ), // Form
  ), // Padding
  Padding(
    padding: const EdgeInsets.only(top: 180, right: 15),
    child: Container(
      width: 300,
      height: 40,
      child: FloatingActionButton.extended(
        backgroundColor: Color.fromRGBO(255, 12, 108, 242),
        foregroundColor: Colors.white,
        child: Text('Share'),
      ),
    ),
  ),
}

Widget build(BuildContext context) {
  return Container(
    child: TextFormField(
      controller: controller,
      decoration: InputDecoration(
        contentPadding: EdgeInsets.fromLTRB(20.0, 10.0, 20.0, 10.0),
        hintText: 'Option',
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(5),
          borderSide: const BorderSide(
            color: Colors.blue,
          ),
        ),
        focusedBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(5),
          borderSide: const BorderSide(
            color: Color.fromRGBO(228, 33, 149, 243),
          ),
        ),
      ),
    ),
  );
}
```

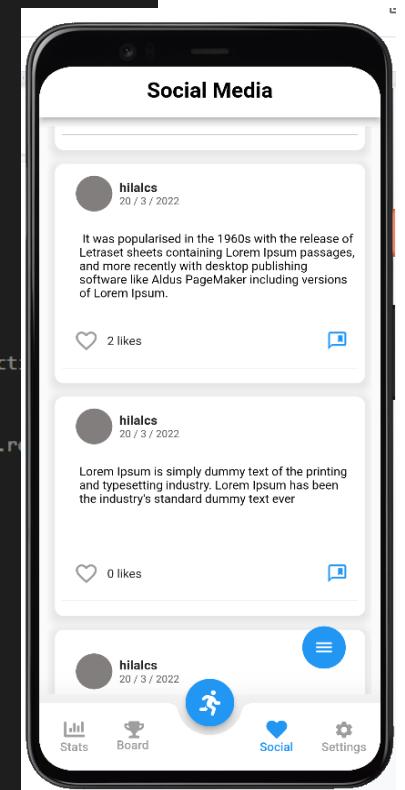
Mobil Uygulama Anket Gönderi Paylaşım Ekranı - Kodları

Kullanıcı sosyal medyada anket paylaşımı yapmak istediği Feed Screen (Akış ekranı) da sağ alt kısmında bulunan butondan anket kısmına tıklayarak bu sayfaya geçiş yapar. Bu sayfayı oluşturma aşamasında kullanıcının anket seçeneklerini gireceği alanları QuestionnaireFormCard widget'ı olarak oluşturuldu. Bu widget da verilerin girileceği yeri TextFormField ile oluşturuldu. Gerekli tasarımsal özellikler InputDecoration ve OutlineInputBorder ile ayarlandı.

Oluşturulan widget'ı yukarıdaki sayfada kullanılabilmesi için QuestionnaireFormCard olarak çağrılmış controller ile içerisinde bulunan değer alınabilir hale getirilir. Ayrıca seçeneklerin üst kısmında bulunan açıklama alanı ise TextField widget kullanılarak oluşturuldu. Metin içeriği özellikleri ise InputDecoration ile tanımlandı. Share butonu ise FloatingActionButton ile gerekli özellikler verilerek oluşturuldu. Kullanıcının gerekli alanları girdiği bilgiler FloatingActionButton'a ait olan onPressed özelliği ile Backend tarafına istek atılır.

#### 4. Sosyal Medya Ana Akış Ekranı

```
215 Positioned(      hilal-coskun, last month • create social-media ...
216   right: 30,
217   bottom: 30,
218   child: Stack(
219     alignment: Alignment.bottomRight,
220     children: <Widget>[
221       IgnorePointer(
222         child: Container(
223           color: Colors.transparent,
224           height: 150.0,
225           width: 150.0,
226         ), // Container
227       ), // IgnorePointer
228       Transform.translate(
229         offset: Offset.fromDirection(getRadiansFromDegree(270),
230           degOneTranslationAnimation.value * 100), // Offset.fromDirection
231         child: Transform(
232           transform: Matrix4.rotationZ(
233             getRadiansFromDegree(rotationAnimation.value)) // Matrix4.rotationZ
234             ..scale(degOneTranslationAnimation.value),
235           alignment: Alignment.center,
236           child: CircularButton(
237             width: 45,
238             height: 45,
239             color: Colors.blue,
240             icon: Icon(
241               Icons.image,
242               color: Colors.white,
243             ), // Icon
244             onClick: () {
245               print(dateTime);
246               print('First Button');
247               Navigator.push(
248                 context,
249                 MaterialPageRoute(
250                   builder: (context) => ShareImageScreen())); // MaterialPageRoute
```

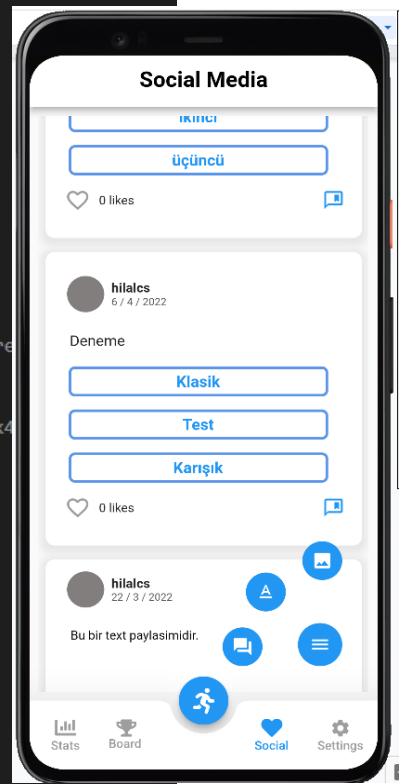


Mobil Uygulama Sosyal Medya Ana Akış Ekranı - Kodları

```

215 Positioned(
216   right: 30,
217   bottom: 30,
218   child: Stack(
219     alignment: Alignment.bottomRight,
220     children: <Widget>[
221       IgnorePointer(
222         child: Container(
223           color: Colors.transparent,
224           height: 150.0,
225           width: 150.0,
226         ), // Container
227       ), // IgnorePointer
228       Transform.translate(
229         offset: Offset.fromDirection(getRadiansFromDegree(270),
230           degOneTranslationAnimation.value * 100), // Offset.fromDirection
231         child: Transform(
232           transform: Matrix4.rotationZ(
233             getRadiansFromDegree(rotationAnimation.value)) // Matrix4
234             ..scale(degOneTranslationAnimation.value),
235           alignment: Alignment.center,
236           child: CircularButton(
237             width: 45,
238             height: 45,
239             color: Colors.blue,
240             icon: Icon(
241               Icons.image,
242               color: Colors.white,
243             ), // Icon
244             onClick: () {
245               print(dateTime);
246               print('First Button');
247               Navigator.push(
248                 context,
249                 MaterialPageRoute(
250                   builder: (context) => ShareImageScreen())); // MaterialPageRoute

```

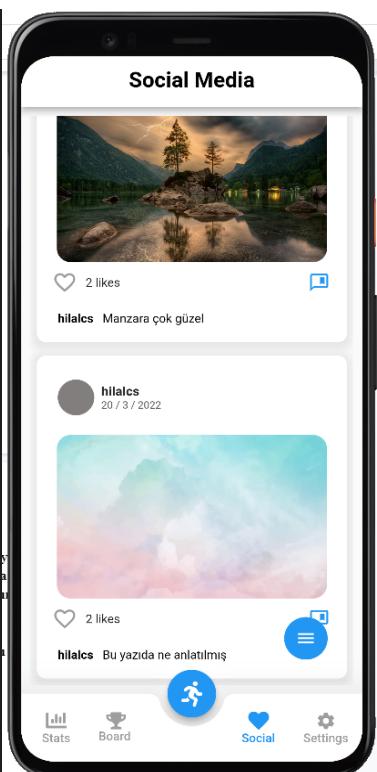


```

@Override
void initState() {
  animationController =
    AnimationController(vsync: this, duration: Duration(milliseconds: 25));
  degOneTranslationAnimation = TweenSequence([
    TweenSequenceItem<double>(
      tween: Tween<double>(begin: 0.0, end: 1.2), weight: 75.0), // TweenSequenceItem
    TweenSequenceItem<double>(
      tween: Tween<double>(begin: 1.2, end: 1.0), weight: 25.0), // TweenSequenceItem
  ]).animate(animationController); // TweenSequence
  degTwoTranslationAnimation = TweenSequence([
    TweenSequenceItem<double>(
      tween: Tween<double>(begin: 0.0, end: 1.4), weight: 55.0), // TweenSequenceItem
    TweenSequenceItem<double>(
      tween: Tween<double>(begin: 1.4, end: 1.0), weight: 45.0), // TweenSequenceItem
  ]).animate(animationController); // TweenSequence
  degThreeTranslationAnimation = TweenSequence([
    TweenSequenceItem<double>(
      tween: Tween<double>(begin: 0.0, end: 1.75), weight: 35.0), // TweenSequenceItem
    TweenSequenceItem<double>(
      tween: Tween<double>(begin: 1.75, end: 1.0), weight: 65.0), // TweenSequenceItem
  ]).animate(animationController); // TweenSequence

  rotationAnimation = Tween<double>(begin: 180.0, end: 0.0).animate(
    CurvedAnimation(parent: animationController, curve: Curves.easeOut));
  super.initState();
  animationController.addListener(() {
    setState(() {});
  });
}

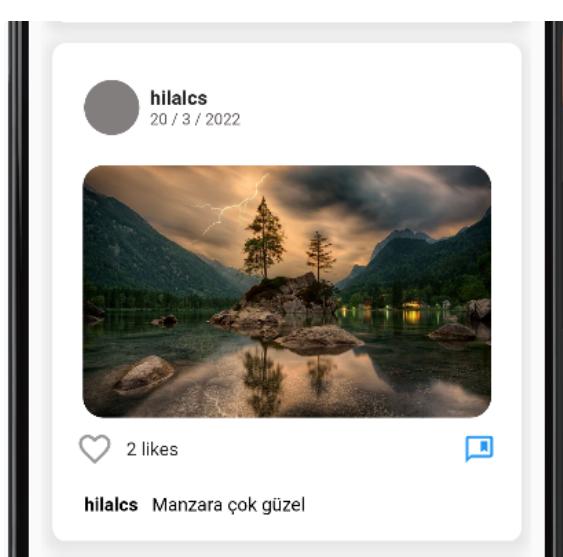
```



Kullanıcı sosyal medya alanına NavigationBar dan sosyal medya ikonuna tıklayarak geçiş yapar. Kullanıcıyı bu ekran karşılar. Bu ekranda sosyal medyada yapılan tüm paylaşımalar yer alır. Bu paylaşımalar Backend tarafından GET isteği atılarak veriler alınır. Alınan veriler ListViewBuilder ile List yapısında sayfada oluşturuldu. Ayrıca ekranda kullanıcının paylaşım yapabilmesi için ekranın sağ alt kısmında CircularButton bulunur. Bu butona kullanıcı tıkladığında Transform özelliği ile açılabilir üç buton ile karşılaşır. Bu üç buton içerisinde metin, resim, anket paylaşımı için görseller bulunmaktadır. Açılmış kapanır özelliğini Matrix4.rotationZ özelliği ile oluşturuldu.

#### **4.1. Sosyal Medya Image(Resim) Paylaşım Kartı**

```
245     DefaultTextStyle(  
246         style: Theme.of(context)  
247             .textTheme  
248                 .subtitle2!  
249             .copyWith(fontWeight: FontWeight.w800),  
250         child: Padding(  
251             padding: const EdgeInsets.only(top: 0),  
252             child: Text(  
253                 widget.likeCount.toString() + ' likes',  
254                 style: Theme.of(context).textTheme.bodyText2,  
255             ), // Text  
256         )), // Padding // DefaultTextStyle  
257     ],  
258 ), // Row  
259 Padding(  
260     padding: const EdgeInsets.only(left: 0),  
261     child: IconButton(  
262         onPressed: () =>  
263             Navigator.of(context).push(MaterialPageRoute(  
264                 builder: (context) => CommentScreen(  
265                     postID: widget.postId,  
266                     username: widget.username,  
267                     )), // CommentScreen // MaterialPageRoute  
268         icon: const Icon(  
269             Icons.comment_bank_outlined,  
270             color: Colors.blue,  
271             size: 25,  
272         ), // Icon  
273     ), // IconButton  
274     ), // Padding  
275 ],
```

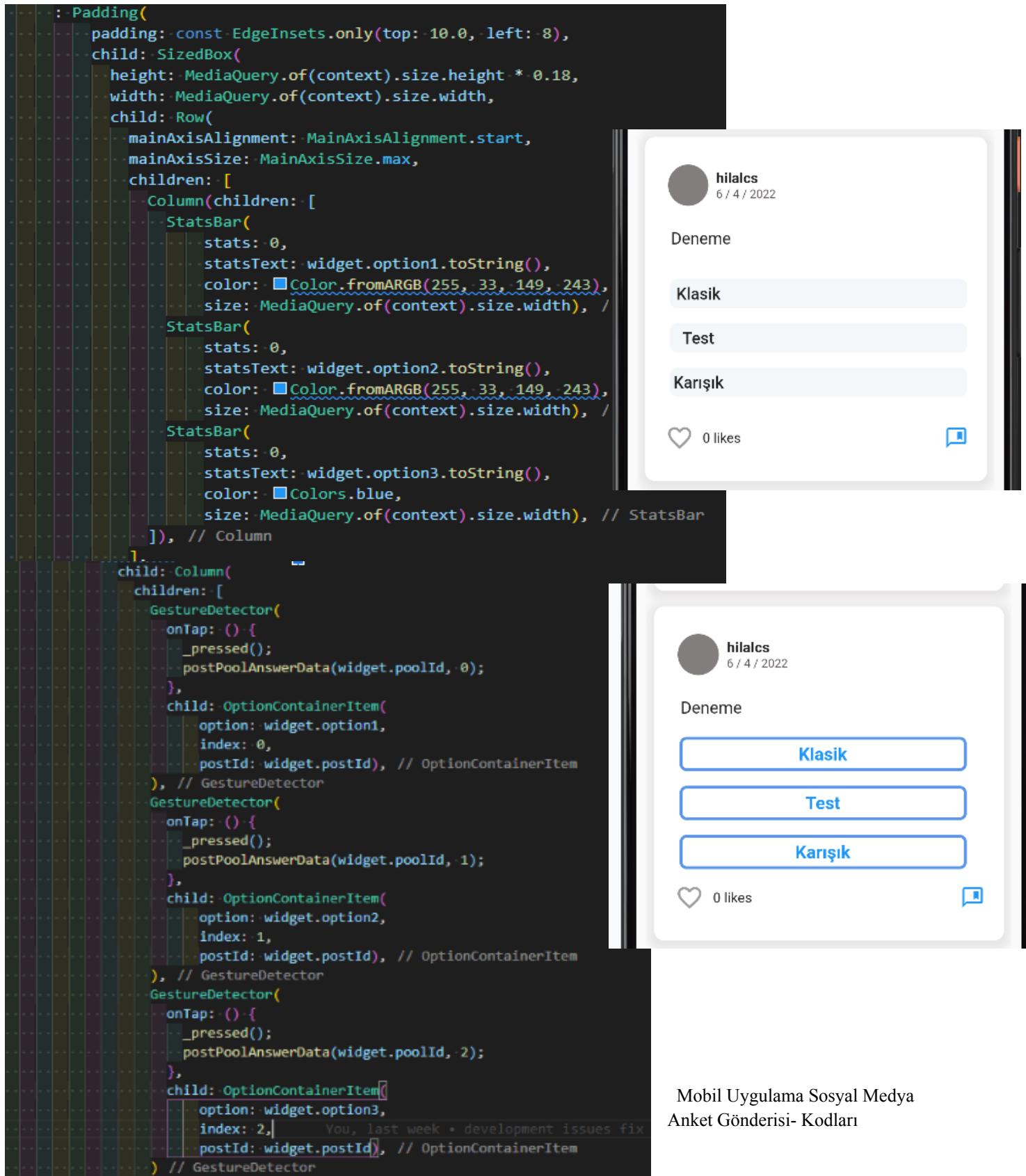


Mobil Uygulama Sosyal Medya Resim Gonderisi- Kodlari

Sosyal medyada yapılan paylaşım sonrasında paylaşımı görüntülenmek istenildiğinde veya Feed Screen (Akış Ekranı)'nda bulunan paylaşımları görüntülenmek istenildiğinde resim kullanılarak yapılan paylaşımlar yukarıda bulunan tasarım ile kullanıcıya sunulur. Paylaşımı Container yapısı ile çerçevelenip resim ise Image.network ile ayarlanmıştır. Container yapısına BoxShadow ile gölgelendirme verilip ayrıca

Stack yapısı da kullanılarak tüm nesnelerin Container içinde bulunması sağlanmıştır. Row özelliği kullanılarak kalp ikonu, beğenme sayısı ve yorum ikonu sıralanmıştır. Ayrıca kullanıcı yorum ikonu MaterialPageRoute ile paylaşımı ait yorumların bulunduğu sayfaya yönlendirilir.

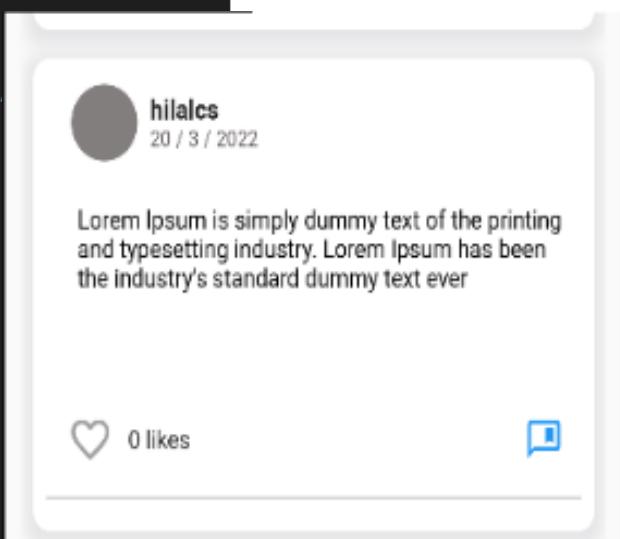
#### 4.2. Sosyal Medya Pool(Anket) Paylaşım Kartı



```
: Padding(
  padding: const EdgeInsets.only(top: 10.0, left: 8),
  child: SizedBox(
    height: MediaQuery.of(context).size.height * 0.18,
    width: MediaQuery.of(context).size.width,
    child: Row(
      mainAxisAlignment: MainAxisAlignment.start,
      mainAxisSize: MainAxisSize.max,
      children: [
        Column(children: [
          StatsBar(
            stats: 0,
            statsText: widget.option1.toString(),
            color: Color.fromRGBO(255, 33, 149, 243),
            size: MediaQuery.of(context).size.width),
          StatsBar(
            stats: 0,
            statsText: widget.option2.toString(),
            color: Color.fromRGBO(255, 33, 149, 243),
            size: MediaQuery.of(context).size.width),
          StatsBar(
            stats: 0,
            statsText: widget.option3.toString(),
            color: Colors.blue,
            size: MediaQuery.of(context).size.width), // StatsBar
        ]),
        // Column
      ],
      child: Column(
        children: [
          GestureDetector(
            onTap: () {
              _pressed();
              postPoolAnswerData(widget.poolId, 0);
            },
            child: OptionContainerItem(
              option: widget.option1,
              index: 0,
              postId: widget.postId), // OptionContainerItem
          ), // GestureDetector
          GestureDetector(
            onTap: () {
              _pressed();
              postPoolAnswerData(widget.poolId, 1);
            },
            child: OptionContainerItem(
              option: widget.option2,
              index: 1,
              postId: widget.postId), // OptionContainerItem
          ), // GestureDetector
          GestureDetector(
            onTap: () {
              _pressed();
              postPoolAnswerData(widget.poolId, 2);
            },
            child: OptionContainerItem(
              option: widget.option3,
              index: 2, You, last week + development issues fix
              postId: widget.postId), // OptionContainerItem
          ), // GestureDetector
        ],
      ),
    ],
  ),
)
```

Uygulamanın sosyal medya kısmında paylaşılan tüm anketler yukarıda solda bulunan görsel şeklinde Feed Screen (Akiş Ekrani) ‘nda görüntülenir. Bu tasarım oluşturulurken seçenekleri barındırmak için OptionContainerItem widget’ı oluşturuldu. Bu widget da BoxDecoration ile tasarıımı zenginleştirildi ve seçenek, seçenekin index numarası taşınır. Bir paylaşım içerisinde üç adet OptionContainerItem’a yer verir. Kullanıcı anket seçeneklerinden herhangi birine oy vermek istediği seçenekin bulunduğu alanı tıklaması yeterlidir. Bu özelliği GestureDetector ile sağlandı. Kullanıcı tıklama olayında sonra ankette seçenekler arası oy yüzdeleri ile karşılaşır. Bu özellik ise StatsBar ve LinearPercentIndicator ile oluşturuldu.

#### 4.3. Sosyal Medya Text(Yazı) Paylaşım Kartı



```

child: Row(children: [
  CircleAvatar(
    radius: 21,
    backgroundColor: Color.fromRGBO(255, 131, 126, 126),
  ), // CircleAvatar
  Expanded(
    child: Padding(
      padding: const EdgeInsets.only(left: 8),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.min,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            username,
            style: TextStyle(
              fontWeight: FontWeight.bold, fontSize: 15), // TextStyle
          ), // Text
          Text(
            lastSeen,
            style: TextStyle(
              fontSize: 12,
              color: Color.fromRGBO(255, 94, 86, 86)),
          ) // Text
        ],
      ), // Column
    ), // Padding
  ) // Expanded
]), // Row
), // Container
PostText(
  content: content,
  likeCount: likeCount,
  username: username,
  postId: postId,
), // PostText
Divider(
  height: 1,
  thickness: 1,
  color: Color.fromRGBO(255, 131, 126, 126),
)

```

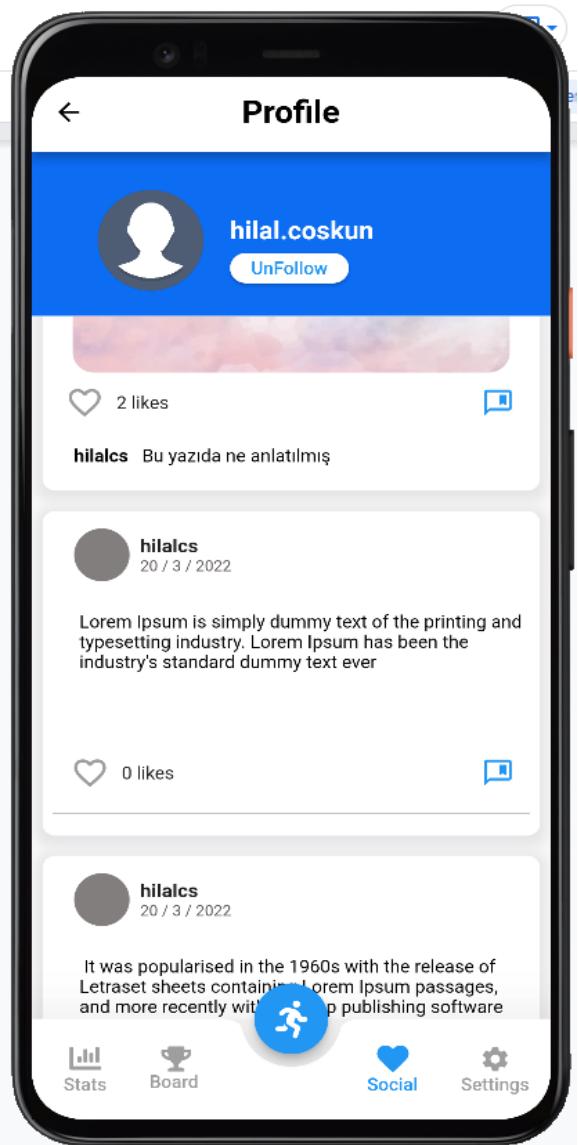
Mobil Uygulama Sosyal Medya Yazı Gönderisi- Kodları

Sosyal medyada yapılan paylaşım sonrasında paylaşımı görüntülenmek istenildiğinde veya Feed Screen (Akiş Ekrani)’nda bulunan paylaşımları görüntülenmek istenildiğinde yazı kullanılarak yapılan paylaşımlar yukarıda bulunan tasarım ile kullanıcıya sunulur. Paylaşımı Container yapısı ile çerçevelenerek ayarlanmıştır. Yaazı metni RichText ve TextSpan ile biçimlendirilerek ayarlanmıştır. Container yapısına BoxShadow ile gölgelendirme verilip ayrıca Stack yapısı da kullanılarak tüm

nesnelerin Container içinde bulunması sağlanmıştır. Row özelliği kullanılarak kalp ikonu, beğenme sayısı ve yorum ikonu sıralanmıştır. Ayrıca kullanıcı yorum ikonu MaterialPageRoute ile paylaşma ait yorumların bulunduğu sayfaya yönlendirilir.

## 5. Sosyal Medya Kişisel Profil Ekranı

```
child: ListView.builder(  
    itemCount: postList.length,  
    itemBuilder: (context, index) {  
      return ((postList[index].postType == 'text')  
        ? PostTextCard(  
            content:  
              postList[index].content.toString(),  
            image:  
              postList[index].postImage.toString(),  
            lastSeen: DateTime.parse(  
              postList[index].creationDate!  
                .day  
                .toString() +  
              ' / ' +  
              DateTime.parse(  
                postList[index].creationDate!  
                  .month  
                  .toString() +  
              ' / ' +  
              DateTime.parse(  
                postList[index].creationDate!  
                  .year  
                  .toString(),  
            username: postList[index]  
              .user!  
              .username  
              .toString(),  
            likeCount:  
              postList[index].usersWhoLike!.length,  
            postId: postList[index].id.toString(),  
        ) // PostTextCard  
        : (postList[index].postType == 'image')
```

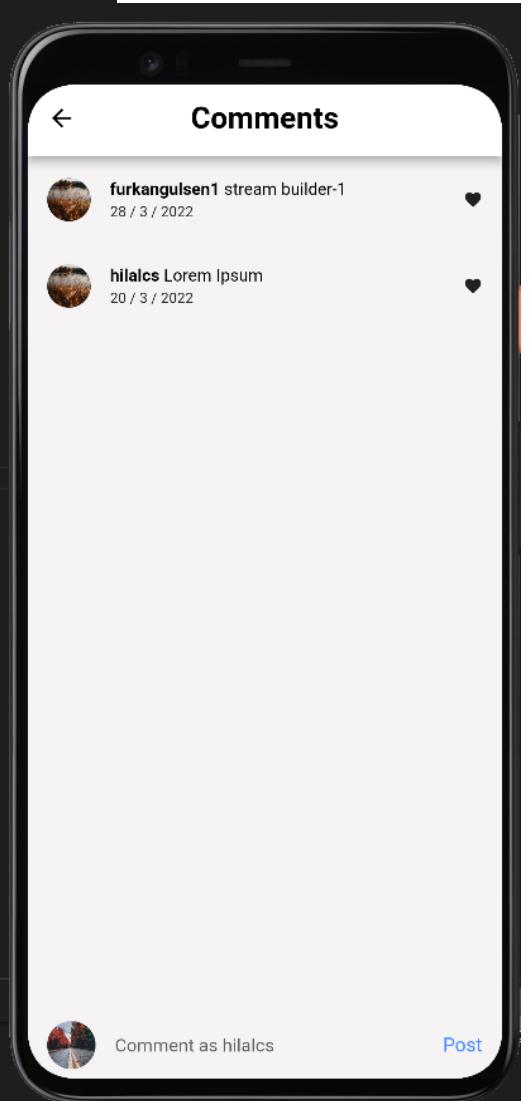


Mobil Uygulama Sosyal Medya Kullanıcı Profili - Kodları

Kullanıcı önceki dönemlerde kendi paylaştığı gönderileri görüntülemek istediginde sosyal medya alanından bu sayfaya geçiş yaptığında kendine ait önceki gönderileri görüntüler. Bu ekranı oluşturan Ana Akış Ekranı yapısı ile aynı olmasına dikkat edildi. Bu ekranın Akış Ekranından tek farkı kullanıcı filtreli olarak oluşturulmuş olmasıdır.

## 6. Sosyal Medya Comment(Yorum) Paylaşım ve Listeleme Ekranı

```
Expanded(  
    child: StreamBuilder<List<Comment>>(  
        You, 2 weeks ago • update like bu  
        stream: getComment(widget.postId.toString()),  
        builder: (context, snapshot) {  
            if (snapshot.hasData) {  
                var commentList = snapshot.data!;  
                return ListView.builder(  
                    itemCount: (commentList.length),  
                    itemBuilder: (context, index) {  
                        return (CommentCard(  
                            comment: commentList[index].comment.toString(),  
                            createDate: DateTime.parse(commentList[index].creationDate).toString(),  
                            day: DateTime.parse(commentList[index].creationDate).day.toString(),  
                            month: DateTime.parse(commentList[index].creationDate).month.toString(),  
                            year: DateTime.parse(commentList[index].creationDate).year.toString(),  
                            username: commentList[index].commentUser.username.toString()); // CommentCard  
                    }); // ListView.builder  
            } else if (snapshot.hasError) {  
                return Text(snapshot.error.toString());  
            } else {  
                return CircularProgressIndicator();  
            }  
        }, // StreamBuilder // Expanded
```

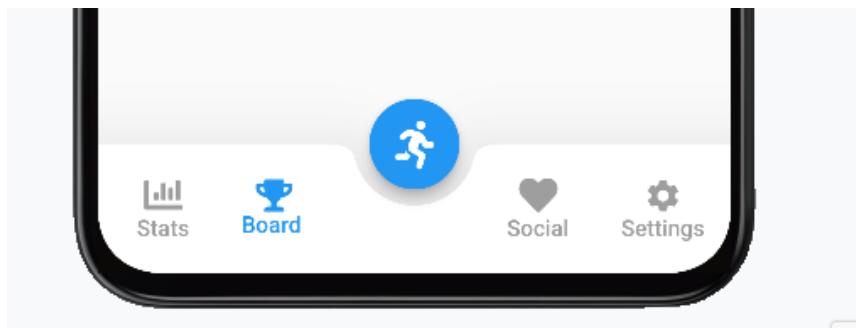


Mobil Uygulama Sosyal Medya Gönderiye ait Yorum Ekranı- Kodları

Kullanıcı sosyal medyada paylaşılan gönderilere ait yorumları görüntülemek ve yorum yazmak istediğiinde, paylaşılan gönderilerin sağ altında bulunan yorum ikonuna tıklayarak bu ekran geçiş yapar. Bu ekranda ise o gönderiye ait tüm yorumlar listelenir. Alt kısmda ise kullanıcı yorum yapmak istediğiinde boş olan kısma yorumu yazıp POST butonuna tıkladığında yorum paylaşmış olur. Kullanıcı sayfayı yenilemeden paylaştığı yorumu görüntüler bu özelliği StreamBuilder ile yapıldı. Tüm yorumlar ListViewBuilder ile listelendi.

## 7. Navigation Bar(Yönlendirme Çubuğu)

```
class _HomePageState extends State<HomePage> {
  int currentTab = 0;
  final List<Widget> screens = [
    SettingsPage(),
    FeedScreen(),
    ActivityHomePage(),
    SWAIHomePage(),
    StatsPage()      You, last week • development issue
  ];
  final PageStorageBucket bucket = PageStorageBucket();
  Widget currentScreen = SWAIHomePage();
}
```



Mobil Uygulama Yönlendirme Bölümü (Navigation Bar)- Kodları

Kullanıcı sayfalar arasında rahatlıkla geçiş yapabilmesi için önce PageStorage ile sayfalar List olarak oluşturuldu. Sayfalara index numaraları verilerek geçiş aşamalandırıldı. NavigationBar da her sayfayı temsil etmesi için FloatingActionButton içerisinde MaterialButton eklenerek onPressed ile sayfaların index numaraları ve sayfa yönlendirmeleri sağlandı. NavigationBar ile Ayarlar Ekranı, Sosyal Medya Akış Ekranı, Aktivite Ekranı, İstatistik Ekranı ve Egzersiz Geçmiş Ekranına geçiş yapılabilir. Kullanıcı uygulamaya giriş yaptığından ana sayfa olarak Aktivite Ekranı ile karşılaşacaktır.

## 8. AppBar(Uygulama Çubuğu) Gösterim Alanı

```
// To implement PreferredSizeWidget
Size get preferredSize => const Size.fromHeight(60);
}

You, last week | 2 authors (You and others)
class _AppBarCardState extends State<AppBarCard> {
  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.only(top: 0.0),
      child: new AppBar(
        backgroundColor: Colors.white,
        elevation: 6,
        title: new Text(
          widget.title,
          style: TextStyle(
            color: Colors.black,
            fontWeight: FontWeight.bold,
            fontSize: 25,
          ), // TextStyle
        ), // Text
        centerTitle: true,
        leading: widget.leftButton,
      ), // AppBar
    ); // Padding
  }
}
```

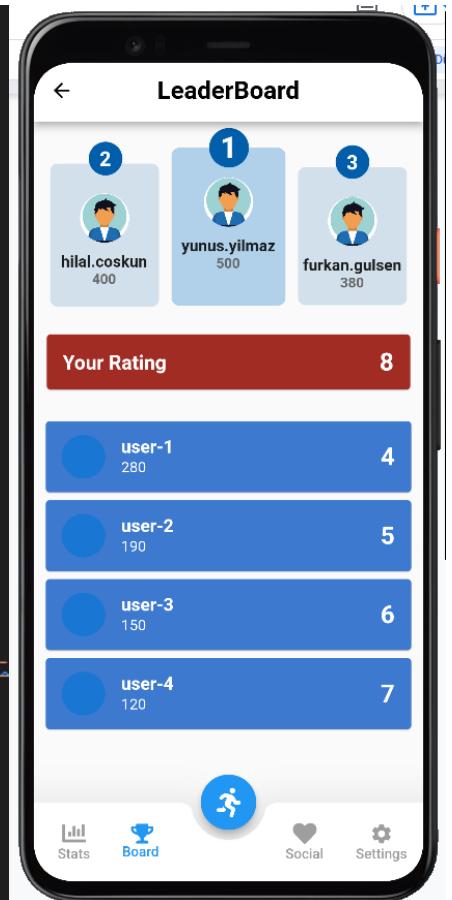


Mobil Uygulama Sayfa Tanıtım Başlık Alanı (AppBar)- Kodları

Kullanıcı uygulamayı kullandığı sırada hangi sayfada olduğunu anlaması için her sayfaya AppBar adı verilen açıklama alanları eklendi. NavigationBar ile sayfalar arası geçiş yapıldığında açılan sayfadan sayfa içerisinde bulunan yönlendirme ile bir diğer başka sayfaya geçiş yaptığından tekrar geri dönebilmesi Navigator.Pop ile sağlandı.

## 9. LeaderBoard(Lider Sıralaması) Ekranı

```
children: [
  Stack(
    children: [
      Padding(
        padding: const EdgeInsets.only(top: 42.0, left: 10),
        child: ClipRRect(
          borderRadius: BorderRadius.circular(8),
          child: Container(
            width: 110,
            height: 145,
            child: Container(
              color: Color.fromARGB(197, 198, 216, 231),
              child: Column(
                children: [
                  SizedBox(
                    height: 30,
                  ), // SizedBox
                  CircleAvatar(
                    radius: 25,
                    backgroundImage: NetworkImage(
                      'https://cdn2.vectorstock.com/i/1000x1000/20/76/man-'
                    ), // CircleAvatar
                  ),
                  SizedBox(
                    height: 10,
                  ), // SizedBox
                  Text(
                    /*Widget.username*/ 'hilal.coskun',
                    style: TextStyle(
                      fontWeight: FontWeight.bold,
                      fontSize: 16,
                    ), // TextStyle
                  ), // Text
                  Text(
                    /*Widget.score*/ '400',
                    style: TextStyle(
                      fontSize: 14,
                      color: Color.fromARGB(255, 104, 103, 103),
                    ), // TextStyle
                  ),
                ],
              ),
            ),
          ),
        ),
      ),
    ],
  ),
]
```



```
LeaderListItem(
    image: '',
    rowNumber: "7",
    score: "150",
    username: "user-4",
), // LeaderListItem
Expanded(
    child: FutureBuilder(
        future: getPostApi(),
        builder: (context, snapshot){
            if(!snapshot.hasData){
                return Text('Loading');
            }else{
                var leaderList;
                return ListView.builder(
                    itemCount: leaderList.length,
                    itemBuilder: (context, index){
                        return (
                            LeaderListItem(
                                score: leaderList[index].score.toString(),
                                image: leaderList[index].image.toString(),
                                rowNumber: leaderList[index].id.toString(),
                                username: leaderList[index].username.toString(),
                            ) // LeaderListItem
                        );
                    }
                ); // ListView.builder
            }
        }
    ) // FutureBuilder
)
```

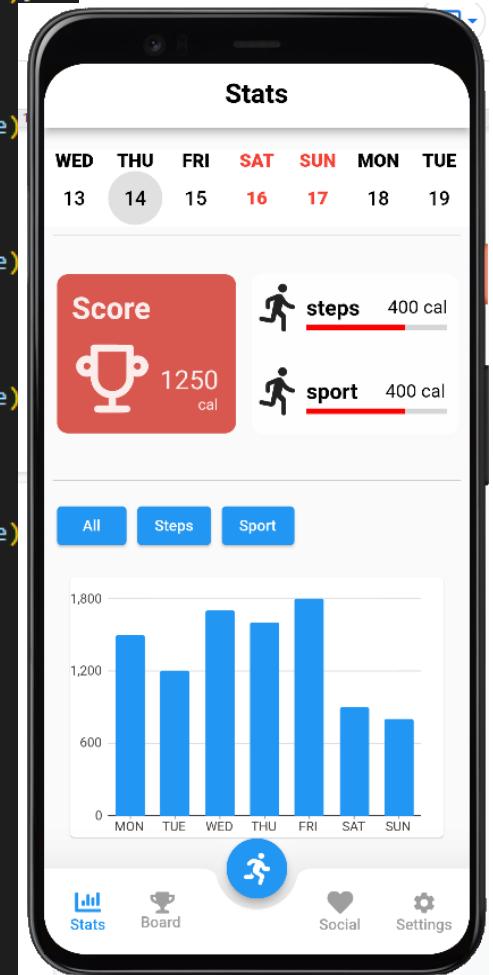
Mobil Uygulama Lider Sıralaması Ekranı - Kodları

Kullanıcı gerçekleştirdiği aktiviteler sonucunda belli istatistikler yaratır. Bu istatistikler ile kullanıcılar skor kazanır ve skorlar ile kullanıcılar kendi aralarında liderlik sıralaması oluşur. Bu ekranda üst kısımda bulunan kartlar Stack yapısı kullanılmıştır. Tüm bileşenlerin bir Container da bulunması için. Your Rating yazan kısımda kullanıcının kendi sıralaması yer alır.

## 10. Stats(İstatistik) Ekranı

```
barColor: charts.ColorUtil.fromDartColor(Colors.blue),
calory: 1200,
day: "TUE"),
StatsGraphModel(
barColor: charts.ColorUtil.fromDartColor(Colors.blue),
calory: 1700,
day: "WED"),
StatsGraphModel(
barColor: charts.ColorUtil.fromDartColor(Colors.blue),
calory: 1600,
day: "THU"),
StatsGraphModel(
barColor: charts.ColorUtil.fromDartColor(Colors.blue),
calory: 1800,
day: "FRI"),
StatsGraphModel(
barColor: charts.ColorUtil.fromDartColor(Colors.blue),
calory: 900,
day: "SAT"),
StatsGraphModel(
barColor: charts.ColorUtil.fromDartColor(Colors.blue),
calory: 800,
day: "SUN"),
];
}

@Override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBarCard(
title: 'Stats',
), // AppBarCard
body: Center(
child: Padding(
padding: const EdgeInsets.only(top: 20.0),
child: GraphicCard(
data: data,
userId: widget.userId,
), // GraphicCard
), // Padding
)); // Center // Scaffold
}
```



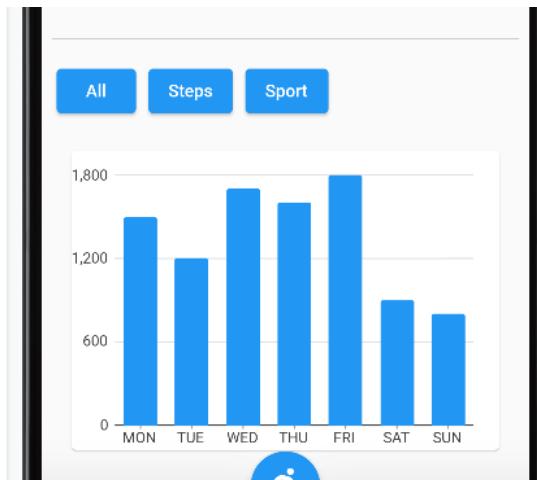
Mobil Uygulama İstatistik Ekranı - Kodları

Kullanıcı geçmiş tüm istatistiklerini görüntülemek istediğiinde Stats Ekranına gelerek görüntüler. Bu ekranda gün içinde attığı adım sayısı, yaptığı aktivite sayesinde yaktığı kalori miktarını ve bu verilerin istatistik grafiği görüntülenir. Grafik oluşturulurken GraphicCard widget kullanılmıştır. GraphicCard içerisinde Backend ten alınan tüm verilerin charts.Series ile grafiği oluşturulur. Ayrıca widgetların kodları bir sonraki görsellerde bulunmaktadır.

## 10.1. İstatistik Verileri Görüntüleme Çizelgesi

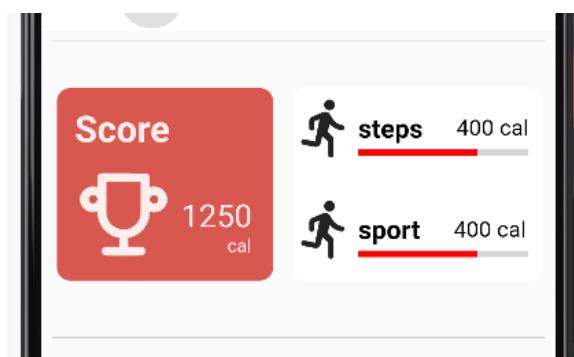
```
Widget build(BuildContext context) {
  List<charts.Series<StatsGraphModel, String>> series = [
    charts.Series(
      id: "GraphicStats",
      data: widget.data,
      domainFn: (StatsGraphModel series, _) => series.day!,
      measureFn: (StatsGraphModel series, _) => series.calory,
      colorFn: (StatsGraphModel series, _) => series.barColor!,
    )
  ];
  return Column(children: [
    MiniCalendar(
      controller: _dateController,
    ), // MiniCalendar
    Padding(
      padding: const EdgeInsets.only(left: 8.0, right: 8.0),
      child: Divider(color: Color.fromRGBO(255, 163, 161, 161)),
    ), // Padding
    Padding(
      padding: const EdgeInsets.only(top: 20.0, bottom: 18, left: 4),
      child: Row(
        children: [ScoreCard(), StepsCard()],
      ), // Row
    ), // Padding
    SizedBox(height: 10),
    Padding([
      You, 4 weeks ago • leaderboard-stats page ...
    ], padding: const EdgeInsets.only(left: 8.0, right: 8.0),
    child: Divider(color: Color.fromRGBO(255, 163, 161, 161)),
  ], // Padding
    Padding(
      padding: const EdgeInsets.only(left: 12.0, top: 10),
      child: Row(
        children: [
          ElevatedButton(
            onPressed: () {
              getAllStatistic(widget.userId);
            },
          ),
        ],
      ),
    ),
  ],
);
```

Mobil Uygulama İstatistik Ekranına ait Grafik Widget - Kodları



## 10.2. Score(Skor) ve Steps(Adım) Verileri Görüntüleme Kartları

```
Padding(  
    padding: const EdgeInsets.only(top: 15.0, left: 10),  
    child: Column(  
        children: [  
            Text(  
                'Score',  
                style: TextStyle(  
                    color: Color.fromARGB(228, 255, 255, 255),  
                    fontWeight: FontWeight.bold,  
                    fontSize: 28), // TextStyle  
            ), // Text  
            Icon(  
                TablerIcons.trophy,  
                color: Color.fromARGB(228, 255, 255, 255),  
                size: 80,  
            ) // Icon  
        ],  
    ), // Column  
, // Padding  
Padding(  
    padding: const EdgeInsets.only(top: 85, left: 5),  
    child: Column(  
        crossAxisAlignment: CrossAxisAlignment.end,  
        children: [  
            Text(  
                widget.score, You, 1 second ago • Uncommitted  
                style: TextStyle(  
                    color: Color.fromARGB(228, 255, 255, 255),  
                    fontSize: 24), // TextStyle  
            ), // Text  
            Text(  
                'cal',  
                style: TextStyle(  
                    color: Color.fromARGB(228, 255, 255, 255),  
                ), // TextStyle  
            ), // Text
```



Mobil Uygulama İstatistik Ekranına ait Skor ve Adım Widget

Mobil Uygulama İstatistik Ekranına ait Skor ve Adım Widget - Kodları

## 11. Data Modelleri

### **11.1. Sosyal Medya Modeli**

Kullanıcı uygulamanın sosyal medya alanında paylaşım yapmak için bu modeli kullanır. Bu modeli backend tarafından gelen JSON verilerine Dart diline çevirerek kullanıcının kullanılması için hazır hale getirilir. Kodlarda görünen kısımda Post Model de id, userId, creationDate, content, postType, usersWhoLike bulunur. Ayrıca PostImage, Pool, PostComment in de kendilerine ait data modelleri de bulunur. İlerleyen aşamalarda anlatılacaktır.

```

PostModel.fromJson(Map<String, dynamic> json) {
    id = json['id'];
    userId = json['userId'];
    creationDate = json['creationDate'];
    content = json['content'];
    postType = json['postType'];
    pool = json['pool'] != null ? new Pool.fromJson(json['pool']) : null;
    usersWhoLike = json['usersWhoLike'].cast<String>();
    postImage = json['postImage'] != null
        ? new PostImage.fromJson(json['postImage'])
        : null;
    user = json['user'] != null ? new User.fromJson(json['user']) : null;
    if (json['postComments'] != null) {
        postComments = <PostComments>[];
        json['postComments'].forEach((v) {
            postComments!.add(new PostComments.fromJson(v));
        });
    }
}

Map<String, dynamic> toJson() {
    final Map<String, dynamic> data = new Map<String, dynamic>();
    data['id'] = this.id;
    data['userId'] = this.userId;
    data['creationDate'] = this.creationDate;
    data['content'] = this.content;
    data['postType'] = this.postType;
    data['usersWhoLike'] = this.usersWhoLike;
    if (this.postComments != null) {
        data['postComments'] = this.postComments?.map((v) => v.toJson()).toList();
    }
    if (this.pool != null) {
        data['pool'] = this.pool.toJson();
    }
}

```

Mobil Uygulama Sosyal Medya Json to Dart Modeli

## 11.2. İstatistik Çizelge Modeli

Kullanıcı aktivite geçmişinde olan verileri görüntülemesi için Ön uç kısmında Arka Uçtan bağımsız olarak StatsGraphModel adında model yaratıldı. Bu kısım JSON verilerini Dart diline çevrilerek oluşturuldu. Bu model içerisinde

- day: İstatistiklerin ait olduğu gün kavramı
- calory: Kullanıcının o gün içerisinde yakmış olduğu kalori miktarı
- barColor: İstatistik sütununun rengi

```
1 import 'package:charts_flutter/flutter.dart' as charts;
2
3 You, 4 weeks ago | 1 author (You)
4 class StatsGraphModel { You, 4 weeks ago • leaderboard-stats page ...
5   final String? day;
6   final int? calory;
7   final charts.Color? barColor;
8
9   StatsGraphModel(
10     {this.day, this.barColor, this.calory});
11 }
```

Mobil Uygulama İstatistik Çizelge Json to Dart Modeli

### 11.3. User(Kullanıcı) Modeli

Kullanıcının kişisel bilgilerini sayfada görüntüleyebilmek için Ön uç kısmında User adında model yaratıldı. Bu kısım JSON verilerini Dart diline çevrilerek oluşturuldu. Bu model içerisinde Arka Uç kısmında olan tüm değişkenler bulunur.

```
1 User.fromJson(Map<String, dynamic> json) {
2   id = json['id'];
3   email = json['email'];
4   username = json['username'];
5   password = json['password'];
6   token = json['token'];
7   registerDate = json['registerDate'];
8   isNotifyActive = json['isNotifyActive'];
9   theFirstDayOfWeek = json['theFirstDayOfWeek'];
10  trainingDays = json['trainingDays'];
11  isAccountActive = json['isAccountActive'];
12  profilePhoto = json['profilePhoto'];
13 }
14
15 Map<String, dynamic> toJson() {
16   final Map<String, dynamic> data = new Map<String, dynamic>();
17   data['id'] = this.id;
18   data['email'] = this.email;
19   data['username'] = this.username;
20   data['password'] = this.password;
21   data['token'] = this.token;
22   data['registerDate'] = this.registerDate;
23   data['isNotifyActive'] = this.isNotifyActive;
24   data['theFirstDayOfWeek'] = this.theFirstDayOfWeek;
25   data['trainingDays'] = this.trainingDays;
26   data['isAccountActive'] = this.isAccountActive;
27   data['profilePhoto'] = this.profilePhoto;
28 }
```

Mobil Uygulama User Json to Dart Modeli

### 11.4. Sosyal Medya Yorum Modeli

Kullanıcı uygulamanın sosyal medya alanında paylaşılara yorum yapmak için bu modeli kullanır. Bu modeli backend tarafından gelen JSON verilerine Dart diline çevirerek kullanıcının kullanılması için hazır hale getirilir. Bu model önceki modellerde anlatılan Post Model'e bağlıdır. PostComment

yapısında id, creationDate, postId, commentUserId, comment ve ayrıca yorum yapacak kullanıcıya ait CommentUser adında User modeline bağlı bir model kullanılır.

```
18     . . . this.usersWhoLike,
19     . . . this.commentUser);
20
21     PostComment.fromJson(Map json) {
22         id = json['id'];
23         creationDate = json['creationDate'];
24         postId = json['postId'];
25         commentUserId = json['commentUserId'];
26         comment = json['comment'];
27
28         commentUser = json['commentUser'] != null
29             ? new CommentUser.fromJson(json['commentUser'])
30             : null;
31     }
32
33     Map toJson() {
34         final Map<String, dynamic> data = new Map();
35         data['id'] = this.id;
36         data['creationDate'] = this.creationDate;
37         data['postId'] = this.postId;
38         data['commentUserId'] = this.commentUserId;
39         data['comment'] = this.comment;
40         if (this.commentUser != null) {
41             data['commentUser'] = this.commentUser!.toJson();
42         }
43         return data;
44     }
```

Mobil Uygulama Sosyal Medya Yorum Json to Dart Modeli

## 12. Sosyal Medya Servisleri (Arka Uç Bağlantı)

### 12.1. Tüm sosyal medya paylaşımları görüntüleme

Sosyal Medya kısmında tüm kullanıcıların yaptıkları paylaşımıları Feed Screen (Sosyal Medya Akış Ekranı)'da görüntüleyebilmek için Backend tarafında yaratılan endpointlere request atarak tüm verileri Backend tarafından alırız.

Response kısmında class özelinde global olarak oluşturulan URL değişkenine post/all ve token'ı kullanırız. Tüm verileri List yapısında alır ve ListViewBuilder yapısı ile ekranda listeleriz.

```

Future<List<PostModel>> getPostList() async {
    try {
        var response = await Dio().get("$URL/post/all",
            options: new Options(
                headers: {HttpHeaders.authorizationHeader: 'Bearer ${token}'});
        List<PostModel> _postModel = [];
        if (response.statusCode == 200) {
            _postModel =
                (response.data as List).map((e) => PostModel.fromJson(e)).toList();
        }
        return _postModel;
    } on DioError catch (e) {
        return Future.error(e.message);
    }
}

```

Mobil Uygulama Servisleri getPostList() fonksiyonu

## 12.2. Kullanıcıya ait filtrelenmiş tüm sosyal medya paylaşımlarını görüntüleme

Sosyal Medya kısmında kullanıcı özelinde yapılan paylaşımları Social Media Profile Screen (Sosyal Medya Profil Akış Ekranı)'da görüntüleyebilmek için Backend tarafından yaratılan endpointlere request atarak tüm verileri Backend tarafından alırız.

Response kısmında class özelinde global olarak oluşturulan queryString yapısında URL değişkenine post/user/\$userId ve token'ı kullanırız. Tüm verileri List yapısında alır ve ListViewBuilder yapısı ile ekranda listeleriz.

```

Future<List<PostModel>> getUserPostList(userId) async {
    try {
        var response = await Dio().get("$URL/post/user/$userId",
            options: new Options(
                headers: {HttpHeaders.authorizationHeader: 'Bearer ${token}'});
        List<PostModel> _postModel = [];
        if (response.statusCode == 200) {
            _postModel =
                (response.data as List).map((e) => PostModel.fromJson(e)).toList();
        }
        return _postModel;
    } on DioError catch (e) {
        return Future.error(e.message);
    }
}

```

Sosyal Medya Servisleri getUserPostList() fonksiyonu

## 12.3. Kullanıcıya ait tüm istatistikleri görüntüleme

Istatistik sayfasında kullanıcı özelinde yapılan tüm aktiviteler sonucunda elde edilen bilgiler ve hesaplamalar sonucunda ortaya çıkan verileri Stats Screen(Istatistik Ekranı)'da görüntüleyebilmek için Backend tarafından yaratılan endpointlere request atarak tüm verileri Backend tarafından alırız.

Response kısmında class özelinde global olarak oluşturulmuş queryString yapısında URL değişkenine user/history/\$userId ve token'ı kullanırız. Tüm verileri alarak sütun barlarda grafik yapısında kullanıcının görüntülemesi için hazır ederiz.

```
//step.sports.calories
Future<int> getAllStatistic(userId) async {
  try {
    var response = await Dio().get("$URL/user/history/:$userId",
        options: new Options(
            headers: {HttpHeaders.authorizationHeader: 'Bearer ${token}'}) );
    var allStatistic;
    if (response.statusCode == 200) {
      allStatistic = response.data;
    }
    return allStatistic;
  } on DioError catch (e) {
    return Future.error(e.message);
  }
}
```

Sosyal Medya Servisleri getAllStatistic() fonksiyonu

#### 12.4. Kullanıcının sosyal medyada text(yazı) paylaşımı

Kullanıcı sosyal medyada yazı veya metin paylaşmak istediğiinde Akış ekranında sağ alt kısmında bulunan paylaşım ikonuna tıklayarak metin paylaşma ekranına giderek gereken verileri girdikten sonra Share butonuna tıkladıktan sonra bu istek Backend tarafında yaratılan endpointlere request atarak tüm verileri Backend tarafına göndeririz.

Data'nın içerisinde;

content: Kullanıcının girdiği metin

userId: Kullanıcının ait unique Id

postType: Kullanıcının paylaştığı post tipi

Response kısmında data da eklenecek class özelinde global olarak oluşturulan queryString yapısında URL değişkenine /share/text ve token'ı kullanız.

```
Future postData(username, uid, content) async {
  var data = {"content": content, "userId": uid, "postType": "text"};
  var response = await Dio().post("$URL/share/text",
      data: data,
      options: new Options(
          headers: {HttpHeaders.authorizationHeader: 'Bearer ${token}'}) );
  return response;
}
```

Sosyal Medya Servisleri postData() fonksiyonu

## 12.5. Kullanıcının sosyal medyada pool(anket) paylaşımı

Kullanıcı sosyal medyada anket paylaşmak istediğiinde Akış ekranında sağ alt kısımda bulunan paylaşım ikonuna tıklayarak anket paylaşma ekranına giderek gereken verileri girdikten sonra Share butonuna tıkladıktan sonra bu istek Backend tarafında yaratılan endpointlere request atarak tüm verileri Backend tarafına göndeririz.

Data nın içerisinde;

- content: Kullanıcının girdiği metin
- userId: Kullanıcının ait unique Id
- postType: Kullanıcının paylaştığı post tipi
- pool: Kullanıcının anket seçeneklerinin kapsar

Response kısmında data da eklenerek class özelinde global olarak oluşturulan queryString yapısında URL değişkenine /share/pool ve token'ı kullanırız.

```
Future<postPoolData(content,option1,option2,option3,uid,) async {  
    List _options = [option1, option2, option3];  
    var data = {  
        "userId": uid,  
        "content": content,  
        "postType": "pool",  
        "pool": {"options": _options},  
    };  
  
    var response = await Dio().post("$URL/share/pool",  
        data: data,  
        options: new Options()  
            ..headers = {HttpHeaders.authorizationHeader: 'Bearer ${token}'});  
  
    return response;  
}
```

Sosyal Medya Servisleri postPoolData() fonksiyonu

## 12.6. Kullanıcının sosyal medyada paylaşılmış olan pool(ankete) oy vermesi

Kullanıcı sosyal medyada paylaşılan herhangi bir ankete oy vermek istediğiinde anket paylaşımında yer alan üç seçenekten birisine tıkladıktan sonra bu istek Backend tarafında yaratılan endpointlere request atarak tüm verileri Backend tarafına göndeririz.

Data nın içerisinde;

- poolId:** Anket paylaşımına ait unique Id
- answer:** Kullanıcının oy verdiği seçeneğin index numarası

Response kısmında data da eklenerek class özelinde global olarak oluşturulan queryString yapısında URL değişkenine /pool/answer?poolId=\$poolId&answer=\$index ve token'ı kullanırız.

```
Future postPoolAnswerData(poolId, index) async {
  try {
    var response = await Dio().post(
      "$URL/pool/answer?poolId=$poolId&answer=$index",
      options: new Options(
        headers: {HttpHeaders.authorizationHeader: 'Bearer ${token}'});
  } catch (e) {
    print(e.toString());
  }
}
```

Sosyal Medya Servisleri postPoolAnswerData() fonksiyonu

## 12.7. Kullanıcının sosyal medyada paylaşılan gönderileri beğenmesi

Kullanıcı sosyal medyada paylaşılan herhangi bir paylaşımı beğenmek istediğiinde paylaşımın sol alt kısmında bulunan kalp ikonuna tıkladıktan sonra bu istek Backend tarafında yaratılan endpointlere request atarak tüm verileri Backend tarafına göndeririz.

Data nın içerisinde;

**postId:** Paylaşımı ait unique Id

**userId:** Kullanıcının ait unique Id

Response kısmında data da eklerek class özeline global olarak oluşturulan queryString yapısında URL değişkenine /post/like?userId=\$userId&postId=\$postId ve token'ı kullanırız.

```
Future likePost(String postId, String userId) async {
  try {
    var response = await Dio().post(
      "$URL/post/like?userId=$userId&postId=$postId",
      options: new Options(
        headers: {HttpHeaders.authorizationHeader: 'Bearer ${token}'});
  } catch (e) {
    print(e.toString());
  }
}
```

Sosyal Medya Servisleri likePost() fonksiyonu

## 12.8. Kullanıcının sosyal medyada paylaşılan gönderide yorum yapması

Kullanıcı sosyal medyada paylaşılan herhangi bir paylaşımına yorum yapmak istediği zaman paylaşımın sağ alt kısmında bulunan yorum ikonuna tıkladıktan sonra kullanıcı yorumlarının bulunduğu sayfaya yönlendirilir bu sayfada kullanıcı istenen alanlara gereken verileri girdikten sonra POST butonuna tıkladıktan sonra bu istek Backend tarafında yaratılan endpointlere request atarak tüm verileri Backend tarafına göndeririz.

Data'nın içerisinde;

- postId: Paylaşımıza ait unique Id
- comment: Kullanıcının yaptığı yorum içeriği
- commentUserId: Kullanıcının ait unique Id

Response kısmında data da eklenerek class özelinde global olarak oluşturulan queryString yapısında URL değişkenine /share/comment ve token'ı kullanırız.

```
Future postComment(String comment, postId, commentUserId) async {
  var data = {
    "comment": comment,
    "postId": "3a36d663-84b9-41a9-ba72-fedb10c25bae",
    "commentUserId": commentUserId
  };
  try {
    if (comment.isNotEmpty) {
      var response = await Dio().post("$URL/share/comment",
        data: data,
        options: new Options(
          headers: {HttpHeaders.authorizationHeader: 'Bearer ${token}'})
    )
  } catch (e) {
    print(e.toString());
  }
}
```

Sosyal Medya Servisleri postComment() fonksiyonu

## 12.9. Sosyal medya paylaşımına ait tüm yorumların görüntülenmesi

Kullanıcı sosyal medyada paylaşılan herhangi bir paylaşımı ait yorumlar görüntülemek istediği zaman paylaşımın sağ alt tarafında bulunan yorum ikonuna tıkladıktan sonra bu istek Backend tarafında yaratılan endpointlere request atarak tüm verileri Backend tarafına göndeririz.

Data'nın içerisinde;

- postId: Paylaşımıza ait unique Id

Response kısmında data da eklenerek class özelinde global olarak oluşturulan queryString yapısında URL değişkenine /comments/\${postId} ve token'ı kullanırız. Tüm verileri List yapısında alır ve ListViewBuilder yapısı ile ekranda listeleriz.

```

Stream<List<dynamic>?>? getComment(String postId) async* {
  try {
    var response = await Dio().get("$URL/comments/${postId}",
      options: new Options(
        headers: {HttpHeaders.authorizationHeader: 'Bearer ${token}'});
    List<PostComment> _postComment = [];
    if (response.statusCode == 200) {
      _postComment =
          (response.data as List).map((e) => PostComment.fromJson(e)).toList();
    }
    yield _postComment;
  } catch (e) {
    print(e);
  }
}

```

Sosyal Medya Servisleri getComment() fonksiyonu

## 12.10. Kullanıcıya ait tüm bilgilerin getirilmesi

Kullanıcıya ait sosyal medyada bulunan profil sayfasında bilgilerinin görüntülenmesi için kullanıcı o sayfaya giriş yaptığında Backend tarafında yaratılan endpointlere request atarak tüm verileri Backend tarafına göndeririz.

Data'nın içerisinde;

userId: Kullanıcıya ait unique Id

Response kısmında data da eklenerek class özeline global olarak oluşturulan queryString yapısında URL değişkenine /user/\$userId ve token'ı kullanırız. Gelen tüm verileri istediğimiz formatta kullanıma hazır hale getirerek kullanırız.

```

Future<String> getUserInformation(userId) async {
  try {
    var response = await Dio().get("http://10.0.2.2:3000/api/user/${userId}",
      options: new Options(
        headers: {HttpHeaders.authorizationHeader: 'Bearer ${token}'});
    var _userInformation;
    print(response.data);
    if (response.statusCode == 200) {
      _userInformation = response.data;
    }
    return _userInformation;
  } on DioError catch (e) {
    return Future.error(e.message);
  }
}

```

Sosyal Medya Servisleri getUserInformation() fonksiyonu

## **2 c. Mobil Uygulama (Yunus Taha Yılmaz**

Bu kısımdan dönem içerisinde mobil uygulama geliştirmeleri bulunmaktadır. Yapılan geliştirmeler başlıca Dart dili ve Flutter teknolojisi aracılığı ile Visual Studio Code üzerinde yazılmıştır ve Servis mimarisinde Flutter Dio kütüphanesinden yararlanılmıştır.

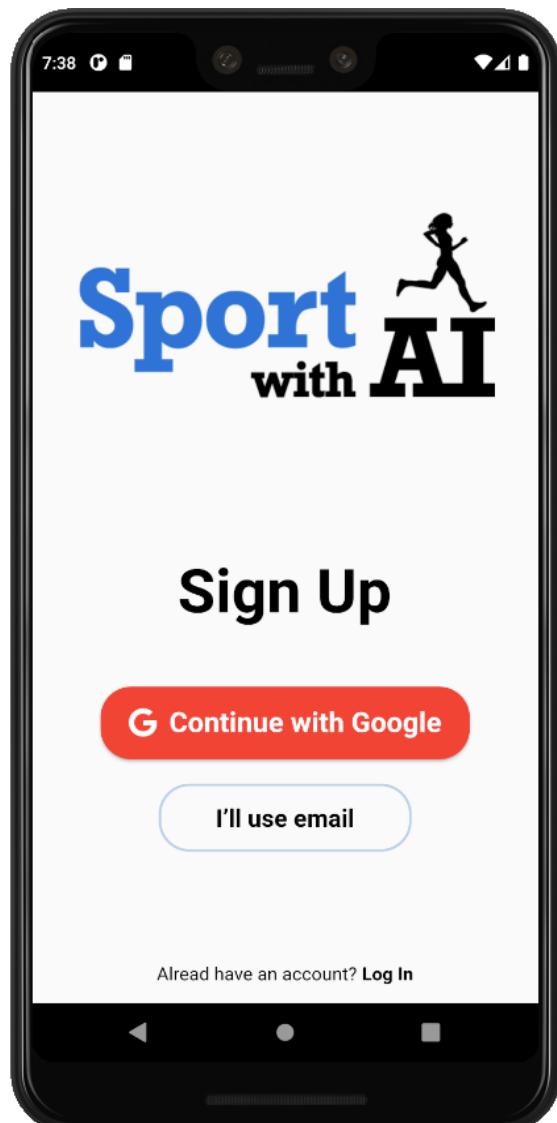
Uygulama testleri ve aşağıdaki görselleri Android Studio Emulator ile gerçekleştirilmiştir.

Geliştirilen yapılar görseller, açıklamalar ve kodlar aracılığıyla aşağı kısımda incelenebilir.

### **1. Welcome Ekranı**

Uygulamanın ilk açılışında veya Splash ekranında token kontrolu yapıldıktan sonra eğer token yoksa bu sayfa açılır.

Google ile veya Email ile giriş veya kayıt sayfalarına kullanıcılar bu sayfa aracılığıyla yönlendirilir.



Welcome Ekranı Gösterimi

## 2. Splash Ekranı

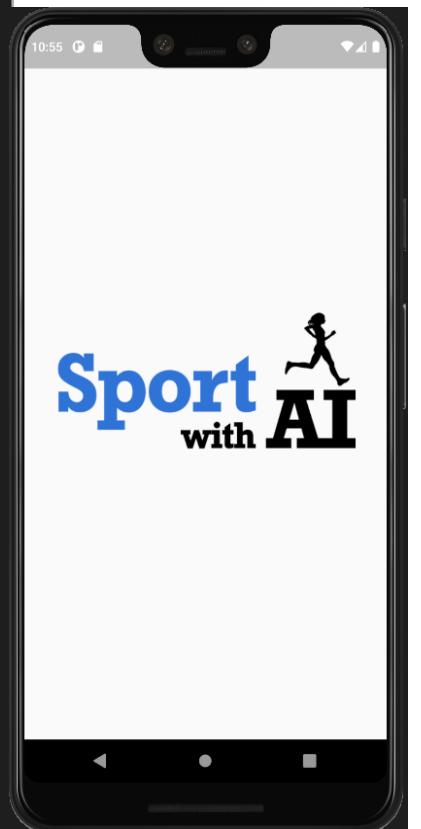
Kullanıcıları uygulamanın açılışında token kontrolü sırasında karşılayan ekranıdır.

Daha önce giriş yapılmışsa ana sayfaya, eğer giriş yapılmamışsa Welcome sayfasına yönlendirir.

```
Future<bool> _checkAuth() async {
    try {
        String? accessKey = await SWAISecureStorage().readSecureData("token");
        if (accessKey == null) {
            return false;
        } else {
            return true;
        }
    } catch (e) {
        print("auth error: ${e}");
        return false;
    }
}

@Override
void initState() {
    Future.delayed(const Duration(seconds: 2), () {
        _checkAuth().then((value) => {
            if (value == true)
            {
                Navigator.of(context)
                    .push(MaterialPageRoute(builder: (context) => HomePage()));
            }
            else
            {
                Navigator.of(context).push(
                    MaterialPageRoute(builder: (context) => WelcomePage()));
            }
        });
    });
    super.initState();
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        body: Center(
            child: Padding(
                padding: EdgeInsets.fromLTRB(35, 0, 35, 0),
                child: Image.asset(
                    "assets/images/logo.png",
                ),
            ),
        );
    );
}
```



Splash Kodları ve Uygulama Ekranı

### 3. Sayfaların Yapımında Kullanılan SWAI Widgetleri

Kodları yazarken tekrar eden yapıları önleyen ve esnek bir şekilde kullanılabilen komponentler yaparak anlaşılabilir, tek bir yerden düzenlenebilir ve performanslı bir yapı oluşturmayı amaçlamıştır.

Aşağıda ekranlar yapılrken kullanılan komponentler kodları ve ekran görüntüleri ile bulunmaktadır.

#### 3.1. Custom Dropdown Widgeti

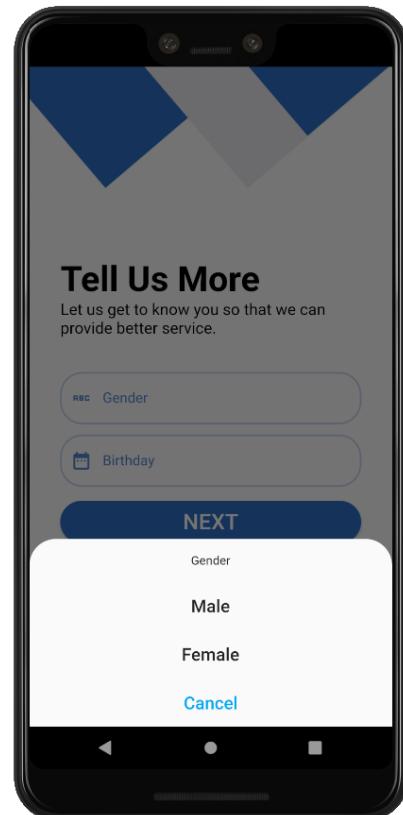
Tıklandığında genel olarak belirli seçeneklerden seçim yapılabilen bir komponenttir.

```
_getBody() {  
    return widget.dropdownItems.map(  
        (e) => BottomSheetAction(  
            title: Text(e),  
            onPressed: () {  
                setState(() => {_controller.text = e, widget.selectedData(e)});  
                Navigator.of(context, rootNavigator: true).pop();  
            },  
        ),  
    );  
}  
  
@override  
Widget build(BuildContext context) {  
    return TextField(  
        controller: _controller,  
        readOnly: true,  
        enableInteractiveSelection: true,  
        onTap: () => {  
            showAdaptiveActionSheet(  
                context: context,  
                title: Text(widget.label),  
                androidBorderRadius: 30,  
                actions: <BottomSheetAction>[..._getBody()],  
                cancelAction: CancelAction(  
                    onPressed: () => {  
                        Navigator.of(context, rootNavigator: true).pop(),  
                    },  
                    title: const Text('Cancel'),  
                ),  
            ),  
            decoration: InputDecoration(  
                prefixIcon: Icon(  
                    Icons.abc_outlined,  
                    color: SWAIColors.blue.withOpacity(0.8),  
                ),  
                focusedBorder: const OutlineInputBorder(  
                    borderRadius: BorderRadius.all(Radius.circular(25)),  
                    borderSide: BorderSide(width: 1.5, color: SWAIColors.blue),  
                ),  
                enabledBorder: OutlineInputBorder(  
                    borderSide: BorderSide(  
                        color: SWAIColors.blue.withOpacity(0.3), width: 1.5),  
                        borderRadius: const BorderRadius.all(Radius.circular(25))),  
                labelText: widget.label,  
                labelStyle: TextStyle(color: SWAIColors.blue.withOpacity(0.8))),  
            );  
    }  
}
```

Hedef, cinsiyet vb. Kısımlarda kullanılmıştır.

Parametreleri aşağıdaki gibidir:  
**selectedData:** Seçilen seçeneği dönderen callback fonksiyonudur.  
**dropdownItems:** Gösterilmek istenilen seçenekler bu parametre aracılığıyla gönderilir.

**Label:** Dropdown'a açıklayıcı bir başlık yazısı vermemizi sağlar.



Dropdown Widgeti Kodları ve Uygulama Ekranı

### 3.2. Image Option Select Widgeti

Fotoğraf değiştirme, kaldırma ve ekleme işlemlerinde kullanılan yapıdır.

```
_getBody() {  
    return _dropdownItems.map(  
        (e) => BottomSheetAction(  
            leading: Padding(  
                padding: const EdgeInsets.only(left: 18),  
                child: e == "Choose from gallery"  
                    ? const Icon(Icons.photo)  
                    : e == "Shot on camera"  
                        ? const Icon(Icons.camera_alt)  
                        : const Icon(  
                            Icons.delete,  
                            color: SWAICOLORS.red,  
                        ),  
            ),  
        ),  
        title: e == "Delete"  
            ? Text(  
                e,  
                style: const TextStyle(color: SWAICOLORS.red  
            ),  
            )  
            : Text(e),  
        onPressed: () {  
            e == "Choose from gallery"  
                ? pickImage(ImageSource.gallery)  
                : e == "Shot on camera"  
                    ? pickImage(ImageSource.camera)  
                    : widget.image(null);  
            Navigator.of(context, rootNavigator: true).pop();  
        },  
    );  
}  
  
@override  
Widget build(BuildContext context) {  
    return InkWell(  
        onTap: (() => {  
            showAdaptiveActionSheet(  
                context: context,  
                androidBorderRadius: 30,  
                actions: <BottomSheetAction>[..._getBody()],  
            )  
        }),  
        child: widget.child,  
    );  
}
```

Tetiklendiğinde al kısından açılan ve çıkan seçeneklere tıklanabilir bir yapıdır.

Parametreleri aşağıdaki gibidir:

**child:** Tıklanacak ikon görüntü vb. yapı girilir. Bu yapıya tıklandığında widget tetiklenir.  
**image:** Seçilen fotoğrafın yolunu almaya yarayan bir call back fonksiyon parametresidir.

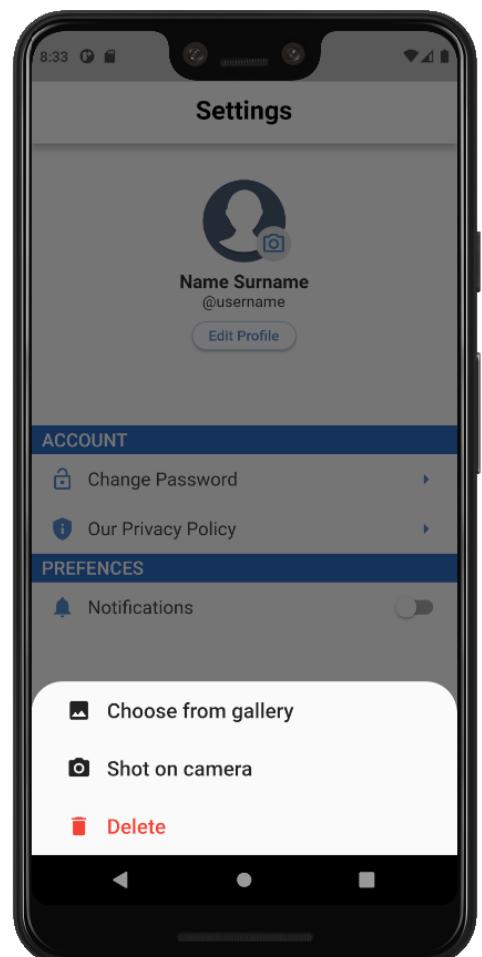


Image Option Select Widgeti

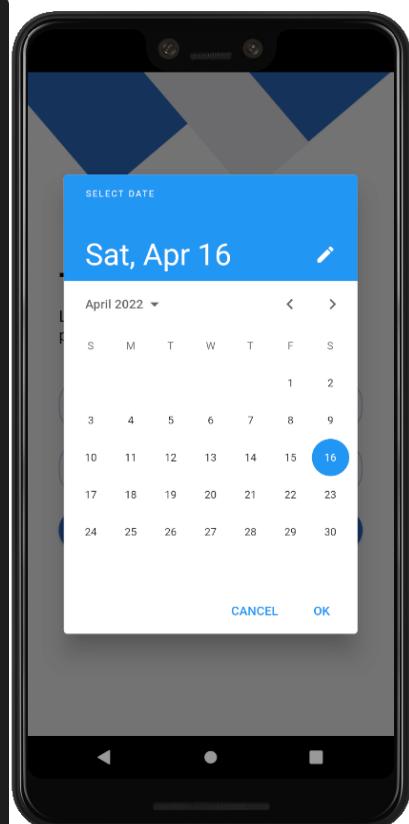
### 3.3. Date Picker Widgeti

Kullanıcıların doğum günü veya herhangi bir tarih seçebildiği takvim yapısında bir widget. Uygulamada kullanıcılarından doğum tarihlerini almak amacıyla kullanılır. Parametreleri aşağıdaki gibidir:

**selectedData:** Seçilen tarihi döndüren bir callback fonksiyonudur.

**Label :** Date Picker'a açıklayıcı bir başlık yazısı vermemizi sağlar.

```
return DateTimePicker(  
  decoration: InputDecoration(  
    prefixIcon: Icon(  
      Icons.date_range,  
      color: SWAIColors.blue.withOpacity(0.8),  
>,  
  focusedBorder: OutlineInputBorder(  
    borderRadius: const BorderRadius.all(Radius.circular(25)),  
    borderSide:  
      BorderSide(width: 1.5, color: SWAIColors.blue.withOpacity(0.8)),  
>,  
  enabledBorder: OutlineInputBorder(  
    borderSide: BorderSide(  
      color: SWAIColors.blue.withOpacity(0.3), width: 1.5),  
    borderRadius: const BorderRadius.all(Radius.circular(25))),  
  labelText: widget.label,  
  labelStyle: TextStyle(color: SWAIColors.blue.withOpacity(0.8))),  
  initialValue: '',  
  onChanged: (val) => setState(() => {  
    widget.selectedDate(val),  
  }),  
  firstDate: DateTime(2000),  
  lastDate: DateTime(2100),  
>);
```



Date Picker Kodları ve Uygulama Ekranı

### 3.4. Mini Calendar Widget'i

Genelde ilgili sayfanın üst kısmında bulunan kullanıcının haftalık sporlarını aktivitelerini görebileceği minik bir takvimdir. Activity ve Stats sayfalarına kullanılır. Parametresi aşağıdaki gibidir:

**controller** Seçilen tarih kullanılan sayfadaki controller'a bu parametre aracılığıyla gönderilir.

```
return CalendarWeek(  
  controller: widget.controller,  
  height: 72,  
  showMonth: true,  
  minDate: DateTime.now().add(  
    const Duration(days: -365),  
  ),  
  maxDate: DateTime.now().add(  
    const Duration(days: 365),  
  ),  
  onDatePressed: (DateTime datetime) {  
    // Do something  
  },  
  onDateLongPressed: (DateTime datetime) {  
    // Do something  
  },  
  onWeekChanged: () {  
    // Do something  
  },  
  marginDayOfWeek: const EdgeInsets.only(),  
  weekendsStyle: const TextStyle(  
    color: Colors.red, fontWeight: FontWeight.w800, fontSize: 17),  
  todayBackgroundColor: Colors.black12,  
  todayDateStyle: const TextStyle(  
    color: SWAIColors.black, fontWeight: FontWeight.w500, fontSize: 18),  
  pressedDateBackgroundColor: SWAIColors.orange,  
  pressedDateStyle: const TextStyle(  
    color: Colors.white, fontWeight: FontWeight.w700, fontSize: 18),  
  dayOfWeekStyle: const TextStyle(  
    color: SWAIColors.black, fontWeight: FontWeight.w800, fontSize: 17),  
  dateStyle: const TextStyle(  
    color: SWAIColors.black, fontWeight: FontWeight.w500, fontSize: 18),  
  monthViewBuilder: (DateTime time) => Align(),  
);
```



Mini Calendar Kodları ve Kullanım Görüntüsü

### 3.5. SWAIPasswordTextField

Login, Sign Up, Change Password sayfalarında kullanılan bu widget genel olarak kullanıcıların şifrelerini girerken gizleyip göstermelerine yarayan bir icon button içerir.

Parametreleri aşağıdaki gibidir:

**controller:** girilen veri kullanılan sayfadaki controller'a bu parametre aracılığıyla gönderilir.

**label:** Inputta default olarak buraya girilen yazı gösterilir.

**icon:** Inputun sol kısmındaki ikonun adı buraya girilir.

**validator.** Şifre kontrol bilgilerinin(karakter sayısı vb.) girildiği parametredir.

```
return TextFormField(
  autofocus: false,
  validator: (value) => widget.validator!(value),
  controller: widget.controller,
  obscureText: !_passwordVisible,
  decoration: InputDecoration(
    suffixIcon: GestureDetector(
      onTap: () => setState(() => _passwordVisible = !_passwordVisible),
      child: Padding(
        padding: const EdgeInsets.only(right: 15),
        child: Row(
          mainAxisAlignment: MainAxisAlignment.end,
          mainAxisSize: MainAxisSize.min,
          children: [
            SizedBox(
              height: 50,
              child: VerticalDivider(
                color: SWAIColors.blue.withOpacity(0.3), width: 20),
            ),
            Icon(
              _passwordVisible ? Icons.visibility : Icons.visibility_off,
              color: SWAIColors.blue.withOpacity(0.8),
            ),
          ],
        ),
      ),
    ),
    prefixIcon: Icon(
      widget.icon,
      color: SWAIColors.blue.withOpacity(0.8),
    ),
    focusedBorder: const OutlineInputBorder(
      borderRadius: BorderRadius.all(Radius.circular(25)),
      borderSide: BorderSide(width: 1.5, color: SWAIColors.blue),
    ),
    enabledBorder: OutlineInputBorder(
      borderSide:
        BorderSide(color: SWAIColors.blue.withOpacity(0.3), width: 2),
      borderRadius: const BorderRadius.all(Radius.circular(25))),
    border: const OutlineInputBorder(
      borderRadius: BorderRadius.all(Radius.circular(25))),
    labelText: widget.label,
    labelStyle: TextStyle(color: SWAIColors.blue.withOpacity(0.8)),
  );
);
```



SWAIPasswordTextField Widgeti Kodları ve Kullanım Görüntüsü

### 3.6. Pie Chart Widget'i

Anlık aktivite durumunun yuvarlak şema halinde göründüğü bir widgettir.

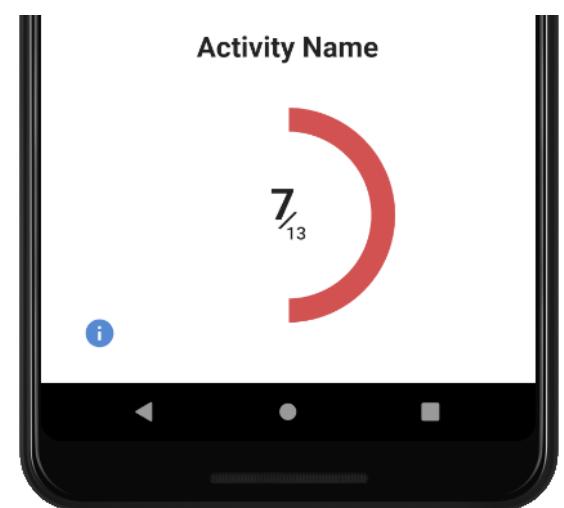
AI ile spot aktivite sayfasında kullanılır

Parametreleri aşağıdaki gibidir:

**measure:** 0 ile 100 arasında bu parametreye girilen sayı grafik olarak gösterilir.

**label:** Grafigin orta kısmında bulunan yapı bu parametreye girilir.

```
DChartPie(  
    data: [  
        {'domain': '1', 'measure': widget.measure  
    },  
        {'domain': '2', 'measure': othersDatas},  
        {'domain': '3', 'measure': othersDatas},  
        {'domain': '4', 'measure': othersDatas},  
        {'domain': '5', 'measure': othersDatas},  
        {'domain': '6', 'measure': othersDatas},  
        {'domain': '7', 'measure': othersDatas},  
        {'domain': '8', 'measure': othersDatas},  
        {'domain': '9', 'measure': othersDatas},  
        {'domain': '10', 'measure': othersDatas},  
    ],  
    labelPosition: PieLabelPosition.inside,  
    labelPadding: 0,  
    labelColor: Colors.white,  
    fillColor: (pieData, index) {  
        switch (pieData['domain']) {  
            case '1':  
                return SWAIColors.logout;  
            default:  
                return SWAIColors.white;  
        }  
    },  
    showLabelLine: false,  
    pieLabel: (pieData, index) {  
        return "";  
    },  
    donutWidth: 20),
```



Pie Chart Widget'i Kodları ve Uygulama Ekranı

### 3.7. SWAITextField Widgeti

```
return TextFormField(
  autofocus: false,
  validator: (value) => widget.validator!(value),
  keyboardType: widget.keyboardType,
  controller: widget.controller,
  decoration: InputDecoration(
    suffixIcon: widget.suffixIcon != null || widget.suffixText != null
      ? Padding(
          padding: const EdgeInsets.only(right: 15),
          child: Row(
            mainAxisAlignment: MainAxisAlignment.end,
            children: [
              SizedBox(
                height: 50,
                child: VerticalDivider(
                  color: SWAIColors.blue.withOpacity(0.3),
                  width: 20),
              ),
              if (widget.suffixIcon != null)
                Icon(
                  widget.suffixIcon,
                  color: SWAIColors.blue.withOpacity(0.8),
                ),
              if (widget.suffixText != null)
                SizedBox(
                  width: 30,
                  child: Text(
                    widget.suffixText!,
                    style: TextStyle(
                      fontWeight: FontWeight.w600,
                      fontSize: 18,
                      color: SWAIColors.blue.withOpacity(0.8)),
                  ),
                ),
            ],
          ),
        )
      : null,
  prefixIcon: Icon(
    widget.icon,
    color: SWAIColors.blue.withOpacity(0.8),
  ),
  focusedBorder: const OutlineInputBorder(
    borderRadius: BorderRadius.all(Radius.circular(25)),
    borderSide: BorderSide(width: 1.5, color: SWAIColors.blue),
  ),
  enabledBorder: OutlineInputBorder(
    borderSide: BorderSide(
      color: SWAIColors.blue.withOpacity(0.3), width: 1.5),
    borderRadius: const BorderRadius.all(Radius.circular(25))),
  border: const OutlineInputBorder(
    borderRadius: BorderRadius.all(Radius.circular(25))),
  labelText: widget.label,
  labelStyle: TextStyle(color: SWAIColors.blue.withOpacity(0.8)),
);
```

Uygulamadaki Mail, Boy, Kilo, Kullanıcı Adı vb. gibi yazı gerektiren yerlerde bu widget kullanılmaktadır.

Parametreleri aşağıdaki gibidir:

**controller:** girilen veri kullanılan sayfadaki controller'a bu parametre aracılığıyla gönderilir.

**label:** İputta default olarak buraya girilen yazı gösterilir.

**icon:** İputun sol kısmındaki ikonun adı buraya girilir.

**suffixIcon:** İputun sağ kısmındaki ikonun adı buraya girilir.

**suffixText:** İputun sağ kısmındaki yazı buraya girilir.

**keyboardType:** klavye tipinin girildiği parametredir.

**validator:** İputa girilen verilerin belirli şartlara göre submit edilmesini sağlayan parametredir.



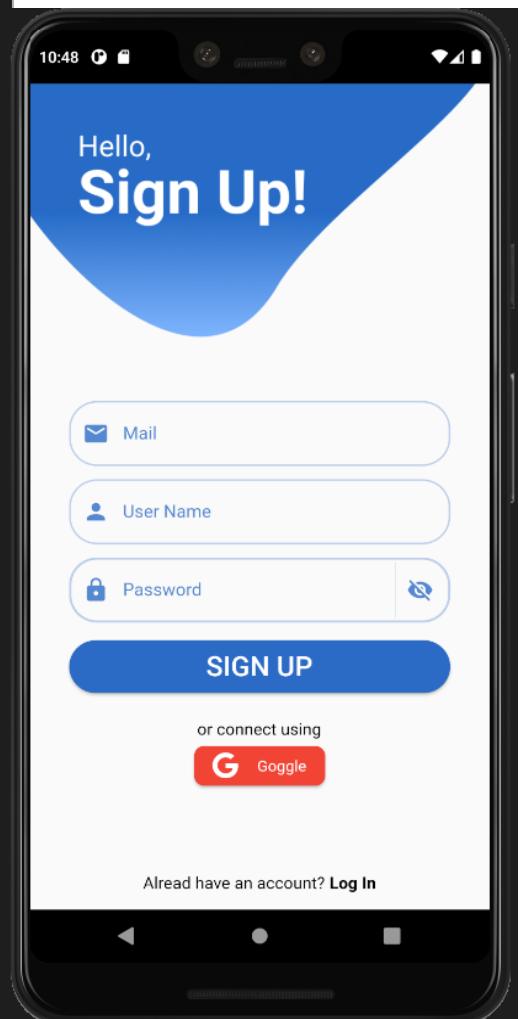
#### 4. Sign Up Ekranı

Kullanıcıların Google ile veya Mail ile kayıt olabildikleri ekranıdır.

```
SWAITextField(  
    controller: _mailController,  
    label: "Mail",  
    icon: Icons.mail_rounded),  
SizedBox(  
    height: size.height * .015,  
>,  
SWAITextField(  
    controller: _nameController,  
    label: "User Name",  
    icon: Icons.person,  
>,  
SizedBox(  
    height: size.height * .015,  
>,  
SWAIPasswordTextField(  
    _passwordController,  
    "Password",  
    Icons.lock,  
>,  
SizedBox(  
    height: size.height * .02,  
>,  
ElevatedButton(  
    style: ButtonStyle(  
        backgroundColor:  
            MaterialStateProperty.all(SWAIColors.blue),  
        minimumSize: MaterialStateProperty.all(Size(  
            size.width,  
            size.height * .06,  
>),  
        shape: MaterialStateProperty.all<  
            RoundedRectangleBorder>(  
            RoundedRectangleBorder(  
                borderRadius: BorderRadius.circular(25.0),  
>),  
        textStyle: MaterialStateProperty.all(  
            const TextStyle(fontSize: 25))),  
    child: const Text(  
        'SIGN UP',  
        style: TextStyle(fontWeight: FontWeight.w600),  
>,  
    onPressed: () {  
        SWAILocalStorage('signup')  
            .addItemsToLocalStorage("signup_info", {  
                'email': '${_mailController.text}',  
                'username': '${_nameController.text}',  
                'password': '${_passwordController.text}'  
>);  
        Navigator.of(context).push(MaterialPageRoute(  
            builder: (context) => TellUsMorePage()));  
>,  
>);
```

Welcome sayfası ile  
kullanıcılar buraya  
yönlendirilir.

**SWAIPasswordTextField** ve  
**SWAITextField** widgetleri  
kullanılarak hazırlanmıştır.



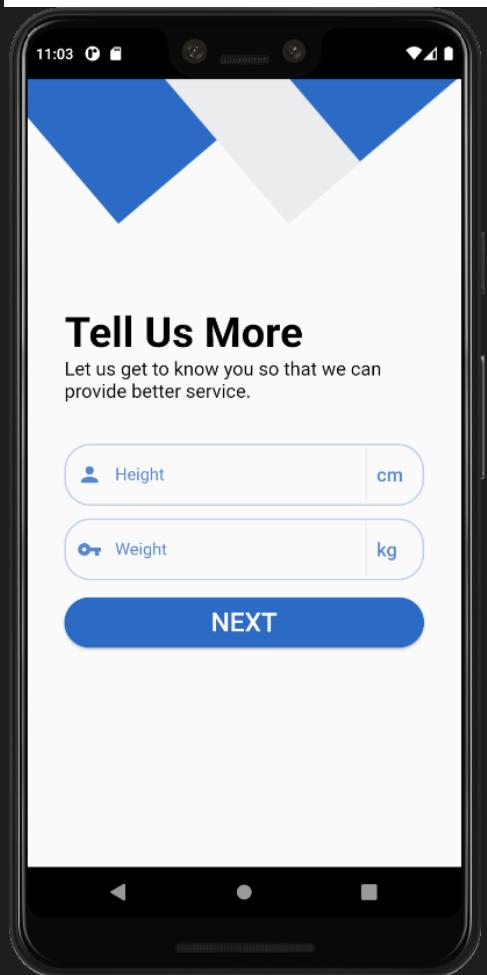
#### 4.1. Tell Us More - Boy Kilo

```
SWAITTextField(  
    controller: _heightController,  
    label: "Height",  
    icon: Icons.person,  
    keyboardType: TextInputType.number,  
    suffixText: "cm"),  
SizedBox(  
    height: size.height * .015,  
,  
SWAITTextField(  
    controller: _weightController,  
    label: "Weight",  
    icon: Icons.vpn_key,  
    keyboardType: TextInputType.number,  
    suffixText: "kg"),  
SizedBox(  
    height: size.height * .02,  
,  
ElevatedButton(  
    style: ButtonStyle(  
        backgroundColor:  
            MaterialStateProperty.all(SWAIColors.blue),  
        minimumSize: MaterialStateProperty.all(Size(  
            size.width,  
            size.height * .06,  
        )),  
        shape:  
            MaterialStateProperty.all<RoundedRectangleBorder>(  
                RoundedRectangleBorder(  
                    borderRadius: BorderRadius.circular(25.0),  
                )),  
        textStyle:  
            MaterialStateProperty.all(TextStyle(fontSize: 25))),  
    child: const Text(  
        'NEXT',  
        style: TextStyle(fontWeight: FontWeight.w600),  
,  
    onPressed: () {  
        SWAILocalStorage('signup')  
            .addItemsToLocalStorage("signup_info", {  
                'height': int.parse(_heightController.text),  
                'weight': int.parse(_weightController.text),  
            });  
  
        Navigator.of(context).push(MaterialPageRoute(  
            builder: (context) => TellUsSecondPage()));  
,  
,
```

Kullanıcının boy ve kilo bilgileri bu ekran aracılığı ile alınır.

NEXT butonuna tıklanıldığında ilgili veriler uygulama localstorageye kayıt edilir.

**SWAITTextField** widgeti kullanılarak yapılmıştır.



Height-Weight Kodları ve Uygulama Ekranı

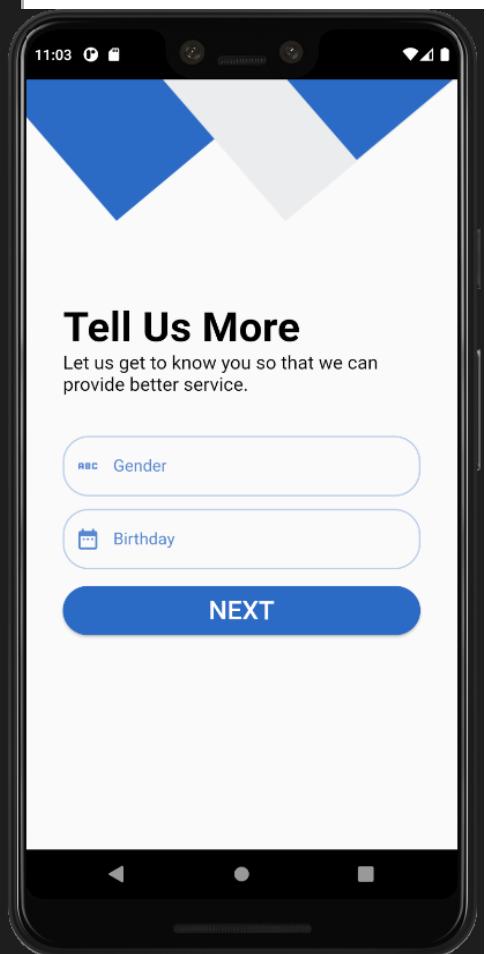
#### 4.2. Tell Us More - Cinsiyet Doğum Tarihi

Kullanıcıdan cinsiyet ve doğum tarihi almak için yapılan ekran bu sekildedir.

```
CustomDropdown(  
    selectedData: (e) => _selectedData = e,  
    dropdownItems: _dropdownItems,  
    label: "Gender"),  
SizedBox(  
    height: size.height * .015,  
>),  
DateTimePicker(  
    decoration: InputDecoration(  
        prefixIcon: Icon(  
            Icons.date_range,  
            color: SWAIColors.blue.withOpacity(0.8),  
>),  
    focusedBorder: OutlineInputBorder(  
        borderRadius: BorderRadius.all(Radius.circular(25)),  
        borderSide: BorderSide(  
            width: 1.5,  
            color: SWAIColors.blue.withOpacity(0.8)),  
>),  
    enabledBorder: OutlineInputBorder(  
        borderSide: BorderSide(  
            color: SWAIColors.blue.withOpacity(0.3),  
            width: 1.5),  
        borderRadius:  
            const BorderRadius.all(Radius.circular(25))),  
    labelText: "Birthday",  
    labelStyle:  
        TextStyle(color: SWAIColors.blue.withOpacity(0.8))),  
    initialValue: '',  
    onChanged: (val) => setState(() => {  
        _birthdayDate = val,  
    }),  
    firstDate: DateTime(2000),  
    lastDate: DateTime(2100),  
>),
```

NEXT butonuna  
tiklanıldığında ilgili veriler  
uygulama localstorageye  
kayıt edilir.

Date Picker ve Custom  
Dropdown widgetleri  
kullanılarak yapılmıştır.



### 4.3. Tell Us More - Hedef

Kullanıcılarından belirli bir haftalık hedef belirlemelerini istediğimiz ekrandır. Bu sayfadaki Complete butonuna tıklanmasıının ardından kullanıcıya kayıt işlemini tamamlaması için bir aktivasyon kodu gönderilir.

**Complete** butonuna tıklandığında kullanıcının Kayıt ve Tell Us More sayfalarına girdiği veriler uygulamanın localstoragede tutulur.

**Custom Dropdown** widgeti kullanılarak yapılmıştır.

```
CustomDropdown(
    selectedData: (e) => _selectedData = e,
    dropdownItems: _dropdownItems,
    label: "Your Goal"),
SizedBox(
    height: size.height * .02,
),
ElevatedButton(
    style: ButtonStyle(
        backgroundColor:
            MaterialStateProperty.all(SWAIColors.blue),
        minimumSize: MaterialStateProperty.all(Size(
            size.width,
            size.height * .06,
        )),
        shape:
            MaterialStateProperty.all<RoundedRectangleBorder>(
                RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(25.0),
                )),
        textStyle:
            MaterialStateProperty.all(TextStyle(fontSize: 25)),
    child: const Text(
        'COMPLETE',
        style: TextStyle(fontWeight: FontWeight.w600),
    ),
    onPressed: () {
        SWAILocalStorage('signup')
            .addItemsToLocalStorage("signup_info", {
            'goal': _selectedData,
        });
        sendActivationCode(SWAILocalStorage('signup')
            .getItemFromLocalStorage("signup_info"));
        print(
            "getItemFromLocalStorage: ${SWAILocalStorage('signup').getItemFromLocalStorage("signup_info")}");
        Navigator.of(context).push(MaterialPageRoute(
            builder: (context) => SignUpActivationPage()));
    },
),
```

Goal Kodları ve Uygulama Ekranı

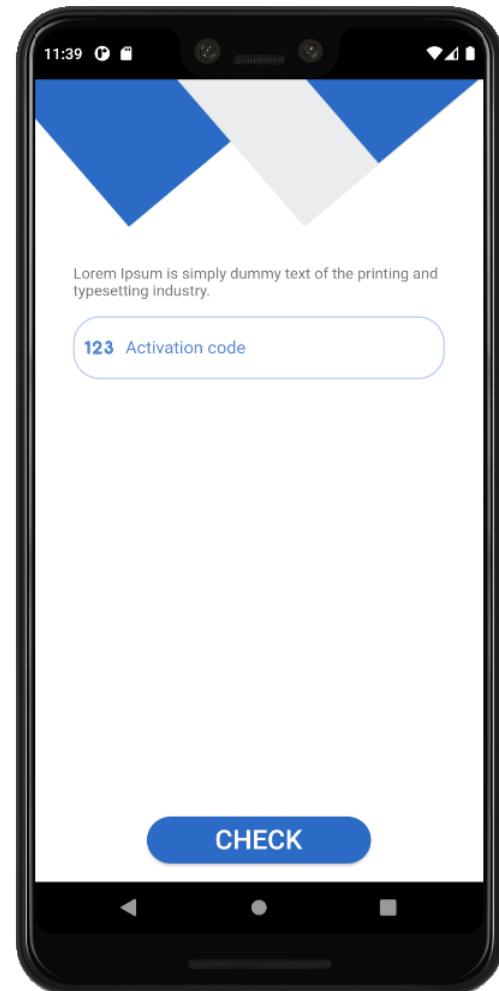
#### 4.4. Aktivasyon Kodu Ekranı

```
Padding(
  padding: const EdgeInsets.fromLTRB(35, 35, 35, 15),
  child: Column(
    children: [
      Text(
        "Lorem Ipsum is simply dummy text of the printing and typesetting industry. "
        ,
        style: TextStyle(
          color: SWAIColors.grey.shade600),
      ),
      SizedBox(
        height: size.height * .02,
      ),
      SWAITextField(
        controller: _activationController,
        label: "Activation code",
        keyboardType: TextInputType.number,
        icon: Typicons.sort_numeric,
      ),
    ],
  ),
),
ElevatedButton(
  style: ButtonStyle(
    backgroundColor:
      MaterialStateProperty.all(SWAIColors.blue),
    minimumSize: MaterialStateProperty.all(Size(
      size.width * .5,
      size.height * .055,
    )),
    shape: MaterialStateProperty.all<
      RoundedRectangleBorder>(RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(25.0),
    )),
    textStyle: MaterialStateProperty.all(
      const TextStyle(fontSize: 25)),
  ),
  child: const Text(
    'CHECK',
    style: TextStyle(fontWeight: FontWeight.w600),
  ),
  onPressed: () {
    SWAISecureStorage()
      .readSecureData("activation_code")
      .then((value) => {
        if (value == _activationController.text)
        {
          signUp(),
          print("İçerdemə: ${value}"),
          Navigator.of(context).push(
            MaterialPageRoute(
              builder: (context) =>
                HomePage()))
        }
        else
        {
          print("Aktivasyon Kodları farklı"),
        }
      });
  },
),
```

Uygulamaya kaydın tamamlanması için kullanıcı mailine gönderilen kodun girildiği ekrandır.

Maile gelen kodla inputa girilen kodun eşleşmesi durumunda kullanıcı kaydı tamamlanır ve ana sayfaya yönlendirilir.

**Check** butonuna tıklandığında tetiklenen fonksiyonu sayesinde aktivasyon kodunun doğruluğu kontrol ediliyor eğer doğru ise servisler aracılığıyla arka uç kışma kullanıcının kaydı gerçekleştiriliyor. **SWAITextField** widgeti kullanılarak yapılmıştır.



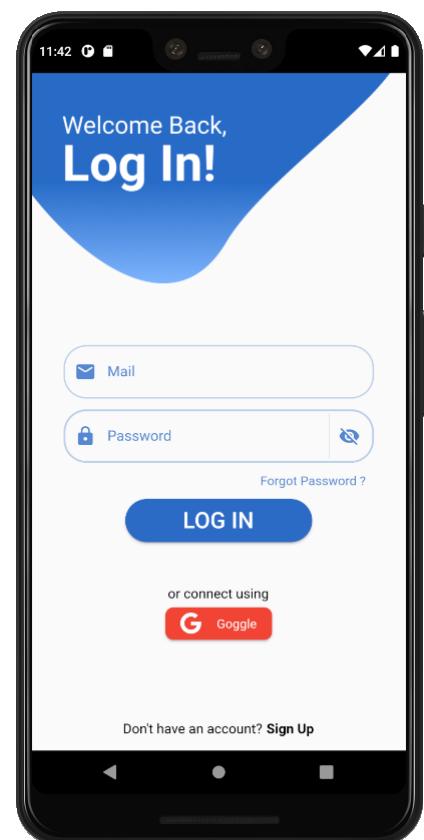
## 5. Login Ekranı

Daha önce uygulamaya kayıt olan kullanıcıların uygulamaya giriş yapmasını sağlayan ekran aşağıdaki gibidir.

Google ile veya Mail aracılığıyla giriş yapılabilir.

```
SWAITextField(  
    validator: (email) {  
        if (isValidEmail(email)) {  
            return null;  
        } else {  
            return 'Please enter a valid email address';  
        }  
    },  
    controller: _mailController,  
    label: "Mail",  
    icon: Icons.mail_rounded),  
SizedBox(  
    height: size.height * .015,  
)  
SWAIPasswordTextField(  
    _passwordController,  
    "Password",  
    Icons.lock,  
    validator: (password) {  
        if (isValidPassword(password)) {  
            return null;  
        } else {  
            return 'Please enter a valid password';  
        }  
    },  
)
```

**SWAITextField** ve  
**SWAIPassword**  
**TextField** widgetleri  
kullanılarak  
yapılmıştır.



Login Kodları ve Uygulama Ekranı

```
ElevatedButton(  
    style: ButtonStyle(  
        backgroundColor: MaterialStateProperty.all(  
            SWAIColors.blue),  
        minimumSize: MaterialStateProperty.all(Size(  
            size.width * .5,  
            size.height * .06,  
        )),  
        shape: MaterialStateProperty.all<  
            RoundedRectangleBorder>(  
            RoundedRectangleBorder(  
                borderRadius: BorderRadius.circular(25.0),  
            )),  
        textStyle: MaterialStateProperty.all(  
            TextStyle(fontSize: 25))),  
    child: const Text(  
        'LOG IN',  
        style: TextStyle(fontWeight: FontWeight.w600),  
    ),  
    onPressed: () {  
        //bu inputtaki focus'u kaldırıyor.  
        FocusScope.of(context).unfocus();  
        if (formGlobalKey.currentState!.validate()) {  
            formGlobalKey.currentState?.save();  
            _onLoading();  
            print(_mailController.text + "\n");  
            print(_passwordController.text);  
            logIn(_mailController.text,  
                  _passwordController.text)  
                .then((value) => {  
                    if (value != null)  
                    {  
                        Navigator.of(context).push(  
                            MaterialPageRoute(  
                                builder: (context) =>  
                                    HomePage()))  
                    }  
                    else  
                    {  
                        print("ÇIKIŞ"),  
                        //loading snipped'ini kapatmak için.  
                        Navigator.of(context).pop()  
                    }  
                });  
        }  
    },  
,
```

**Log In** butonuna tıklandığında tetiklenen submit fonksiyonu sayesinde servisler aracılığıyla arka uç kısmından kullanıcının kayıtlı olup olmadığı kontrol ediliyor.

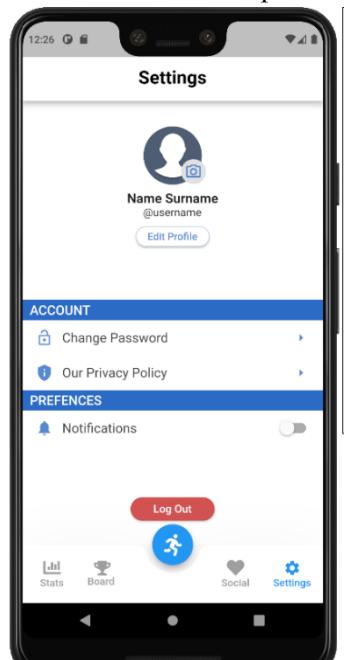


## 6. Settings Ekranları

Uygulamamızın “Setting” ana sayfa ekranı aşağıdaki şekildedir.

Kod satırlarındaki turuncu yazınlara bakılarak hangi kısımların kodları olduğu anlaşılabilir.

Kullanıcılar bu ekran aracılığıyla şifre profil bilgileri düzenleme, görsel değiştirme şifre değiştirme, bildirimleri kapatıp açma, uygulamadan çıkış işlemlerini yapabilir ve gizlilik politikasını okuyabilir.



```
TextButton(
  style: TextButton.styleFrom(),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
      Row(
        children: [
          Icon(
            Icons.privacy_tip_sharp,
            color: SWAIColors.blue.withOpacity(0.8),
          ),
          SizedBox(
            width: size.width * .03,
          ),
          Text(
            "Our Privacy Policy",
            style: TextStyle(
              fontSize: 18,
              color: SWAICOLORS.GREY.shade800,
              fontWeight: FontWeight.normal),
          ),
        ],
      ),
      Icon(
        Icons.arrow_right,
        color: SWAICOLORS.BLUE.withOpacity(0.8),
      ),
    ],
  ),
  onPressed: () {
    Navigator.of(context).push(
      MaterialPageRoute(
        builder: (context) =>
          PrivacyPolicyPage()));
  },
),
```

```
Container(
  alignment: Alignment.centerLeft,
  width: size.width,
  height: size.height * .035,
  color: SWAICOLORS.BLUE,
  child: const Padding(
    padding: EdgeInsets.only(left: 10),
    child: Text(
      "ACCOUNT",
      style: TextStyle(
        fontSize: 18,
        fontWeight: FontWeight.w600,
        color: SWAICOLORS.WHITE),
    ),
  ),
),
child: Row(
  mainAxisSize: MainAxisSize.max,
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  children: [
    Row(
      children: [
        Icon(
          Icons.notifications,
          color: SWAICOLORS.BLUE.withOpacity(0.8),
        ),
        SizedBox(
          width: size.width * .03,
        ),
        Text(
          "Notifications",
          style: TextStyle(
            fontSize: 18,
            color: SWAICOLORS.GREY.shade800,
            fontWeight: FontWeight.normal),
        ),
      ],
    ),
    Switch(
      activeColor: SWAICOLORS.BLUE.withOpacity(0.8),
      value: _isSwitched,
      onChanged: (value) {
        setState(() {
          _isSwitched = value;
          print(_isSwitched);
        });
      },
      activeTrackColor:
        SWAICOLORS.BLUE.withOpacity(0.3),
    ),
  ],
),
```

```
ElevatedButton(
  style: ElevatedButton.styleFrom(
    minimumSize: Size(size.width * .2, size.height * .035),
    primary: SWAICOLORS.WHITE,
    shape: RoundedRectangleBorder(
      borderRadius: const BorderRadius.all(
        Radius.circular(30),
      ),
      side: BorderSide(
        color: SWAICOLORS.BLUE.withOpacity(0.3))),
  ),
  child: Text(
    "Edit Profile",
    style: TextStyle(
      color: SWAICOLORS.BLUE.withOpacity(0.8),
      fontSize: 14),
  ),
  onPressed: () => {
    Navigator.of(context).push(MaterialPageRoute(
      builder: (context) => EditProfilePage())),
  },
),
TextButton(
  style: TextButton.styleFrom(),
  child: Row(
    mainAxisSize: MainAxisSize.max,
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
      Row(
        children: [
          Icon(
            Icons.lock_open,
            color: SWAICOLORS.BLUE.withOpacity(0.8),
          ),
          SizedBox(
            width: size.width * .03,
          ),
          Text(
            "Change Password",
            style: TextStyle(
              fontSize: 18,
              color: SWAICOLORS.GREY.shade800,
              fontWeight: FontWeight.normal),
          ),
        ],
      ),
      Icon(
        Icons.arrow_right,
        color: SWAICOLORS.BLUE.withOpacity(0.8),
      ),
    ],
  ),
  onPressed: () {
    Navigator.of(context).push(
      MaterialPageRoute(
        builder: (context) =>
          ChangePasswordPage()));
  },
),
```

Settings Ana Sayfa Kodları ve Uygulama Görüntüsü

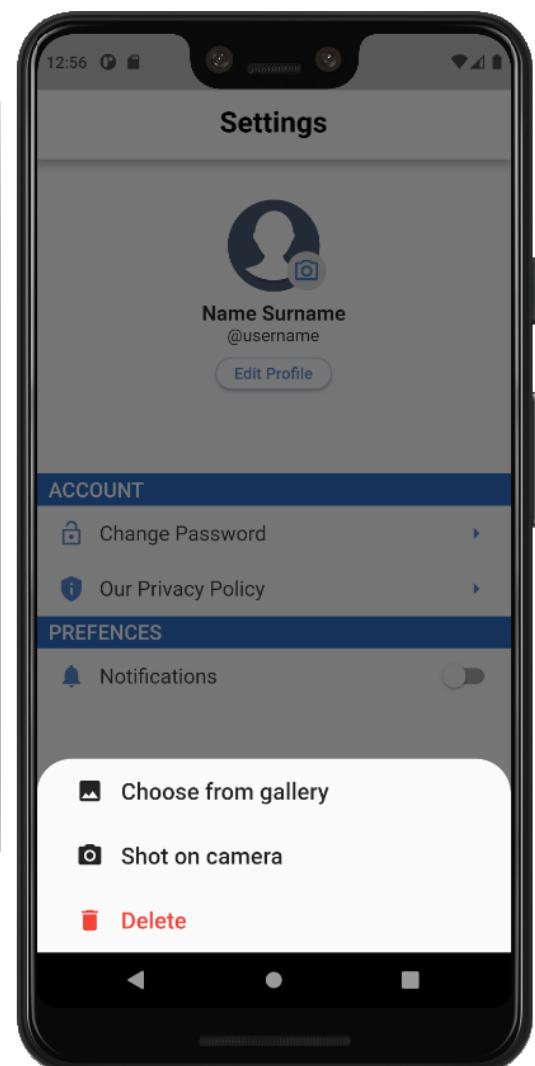
## 6.1. Resim Değiştirme Kısmı

Avatarın sağ altındaki kamera ikonuna tıklandığında resim değiştirme, kaldırma ve ekleme seçenekleri olan menü açılır.

Kullanıcılar bu menu aracılığıyla uygulamada kullanılan fotoğraflarını istedikleri şekilde ayarlayabilir.

**Image Option Select** widgeti kullanılarak yapılmıştır.

```
child: ImageOptionSelect(  
  child: Container(  
    decoration: BoxDecoration(  
      color: SWAICOLORS.grey.shade300,  
      borderRadius:  
        const BorderRadius.all(Radius.circular(25)),  
    ),  
    child: Padding(  
      padding: const EdgeInsets.all(5.0),  
      child: Icon(  
        Icons.camera_alt_outlined,  
        color: SWAICOLORS.blue.withOpacity(0.8),  
      ),  
    ),  
    image: (e) => {  
      setState(() => {_image = e})  
    },  
  ),  
,
```



Setting Resim Değiştirme Kodları ve Uygulama Ekranı

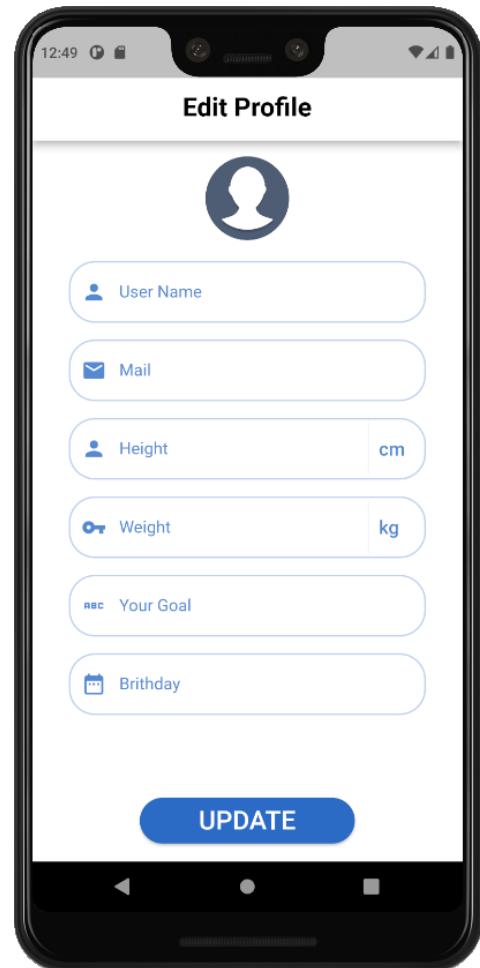
## 6.2. Edit Profile Ekranı

```
SWAITextField(  
    controller: _nameController,  
    label: "User Name",  
    icon: Icons.person),  
SizedBox(  
    height: size.height * .020,  
)  
SWAITextField(  
    controller: _mailController,  
    label: "Mail",  
    icon: Icons.mail_rounded),  
SizedBox(  
    height: size.height * .020,  
)  
SWAITextField(  
    controller: _heightController,  
    label: "Height",  
    icon: Icons.person,  
    keyboardType: TextInputType.number,  
    suffixText: "cm"),  
SizedBox(  
    height: size.height * .020,  
)  
SWAITextField(  
    controller: _weightController,  
    label: "Weight",  
    icon: Icons.vpn_key,  
    keyboardType: TextInputType.number,  
    suffixText: "kg"),  
,  
SizedBox(  
    height: size.height * .020,  
)  
CustomDropdown(  
    selectedData: (e) => _selectedData = e,  
    dropdownItems: _dropdownItems,  
    label: "Your Goal"),  
SizedBox(  
    height: size.height * .020,  
)  
SWAIDatePicker(  
    selectedDate: (e) => _selectedDate = e,  
    label: 'Brithday',  
)
```

Kayıt sırasında Tell Us More sayfalarında girilen bilgiler bu sayfa aracılığıyla değiştirilebilir.

Bu bilgilerin hepsinin tek bir yerden değiştirilmesi sağlanarak kullanıcıya kullanım kolaylığı sunmak amaçlanmıştır.

**SWAITextField, Custom Dropdown ve Date Picker** widgetleri aracılığıyla oluşturulmuştur.



Setting Edit Pofile Kodları ve Uygulama Görüntüsü

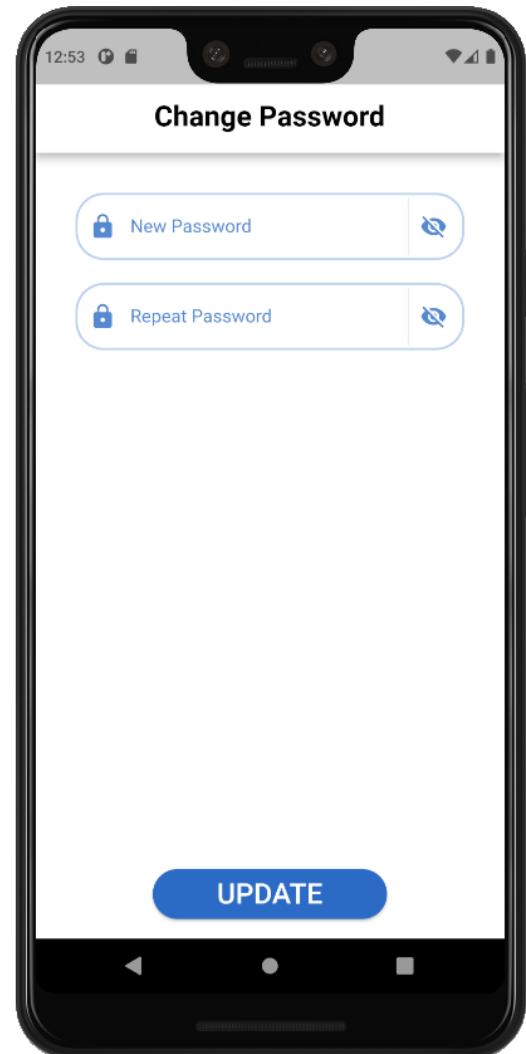
### 6.3. Change Password Ekranı

Kullanıcıların şifre değiştirmek istediklerinde ayarlar aracılığıyla bu sayfaya gelerek kolaylıkla şifrelerini değiştirmelerine yarayan bir ekrandır.

```
child: Column(  
    mainAxisAlignment: MainAxisAlignment.max,  
    mainAxisAlignment: MainAxisAlignment.spaceBetween,  
    children: [  
        Column(  
            children: [  
                SWAIPasswordTextField(  
                    _newPasswordController, "New Password",  
                    Icons.lock),  
                SizedBox(  
                    height: size.height * .025,  
                ),  
                SWAIPasswordTextField(  
                    _repeatPasswordController,  
                    "Repeat Password", Icons.lock),  
            ],  
        ),  
        ElevatedButton(  
            style: ButtonStyle(  
                backgroundColor:  
                    MaterialStateProperty.all(SWAIColors.  
blue),  
                minimumSize: MaterialStateProperty.all(Size(  
                    size.width * .5,  
                    size.height * .055,  
                )),  
                shape:  
                    MaterialStateProperty.all<  
RoundedRectangleBorder>(  
                    RoundedRectangleBorder(  
                        borderRadius: BorderRadius.circular(25.0),  
                    ),  
                    textStyle: MaterialStateProperty.all(  
                        const TextStyle(fontSize: 25)),  
                    child: const Text(  
                        'UPDATE',  
                        style: TextStyle(fontWeight: FontWeight.w600),  
                    ),  
                    onPressed: () {  
                        Navigator.of(context).push(MaterialPageRoute(  
                            builder: (context) => ActivationPage()));  
                    },  
                ),  
            ],  
        ),  
    ],  
)
```

Gereksiz kod tekrarını önlemek ve az kod ile çok iş yapmayı amaçladığımız için bu sayfa yapımında

**SWAIPasswordTextField** widgetinden faydalaniılmıştır.



Settings Change Password Kodları ve Uygulama Ekranı

#### 6.4. Privacy Policy Ekranı

Gizlilik politikamızın kullanıcılar tarafından görülebilmesini sağlayan ekrandır. Herhangi bir özelliği yoktur sadece kullanıcıları bilgilendirmek amacıyla yapılmıştır.

```
body: Container(  
    child: Padding(  
        padding: EdgeInsets.fromLTRB(10, 15, 10, 0),  
        child: Text(  
  
"Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean feugiat non justo in cursus. Ut a pulvinar velit. Duis in neque eget turpis maximus condimentum. Nullam eleifend lacinia nisl ut egestas. Proin eget enim et magna fermentum laoreet e get sit amet dui. Proin euismod sagittis mauris i n euismod. Lorem ipsum dolor sit amet, consectetur adipiscing elit."  
  
,  
        style: TextStyle(  
            color: SWAIColors.black,  
            fontSize: 18,  
            fontWeight: FontWeight.normal),  
        ),  
    ),  
);
```



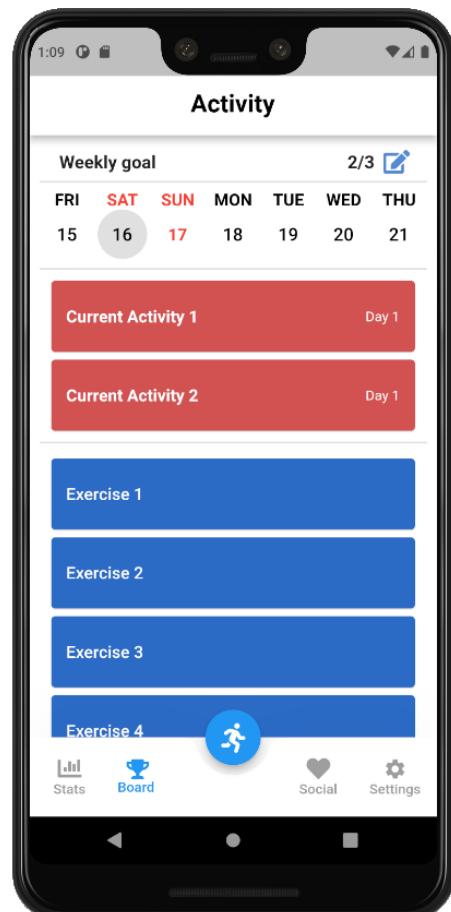
Setting Privacy Policy Kodları ve Uygulama Ekranı

## 7. Activity Ekranları

Kullanıcıların hedef belirlemelerini, aktivitelerine devam etmelerini, egzersiz başlatabilmelerini amaçlayan ve takvim aracılığıyla aktivitelerini daha kolay planlamalarını sağlayan ekranlardır.

Yapımında **Mini Calendar** widgetinden yararlanılmıştır.

```
child: Row(
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  children: [
    const Text(
      "Weekly goal",
      style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
    ),
    Row(
      children: [
        Text(
          "${_selectedDaysCount}/${_totalDays}",
          style: const TextStyle(
            fontSize: 18, fontWeight: FontWeight.bold),
        ),
        SizedBox(
          width: size.width * .02,
        ),
        InkWell(
          onTap: () {
            Navigator.of(context).push(MaterialPageRoute(
              builder: (context) => WeeklyGoalPage()));
            print(
              '_dateController: ${_dateController.selectedDate.day}/${_dateController.selectedDate.month}/${_dateController.selectedDate.year}');
          },
          child: IconTheme(
            data: IconThemeData(
              color: SWAIColors.blue.withOpacity(0.8)),
            child: const Icon(
              FontAwesome5.edit,
            ),
          ),
        ),
      ],
    ),
  ],
),
```



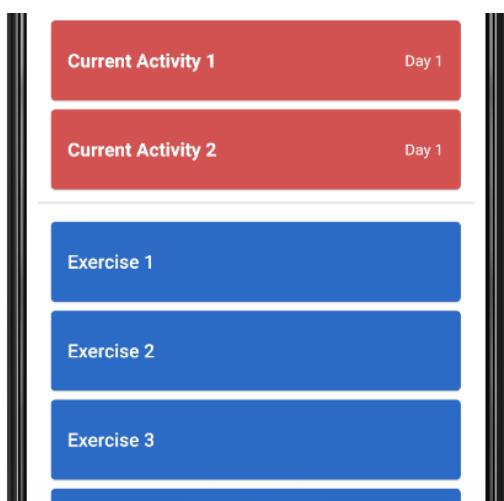
```
Minicalendar(
  controller: _dateController,
```

Activity Ana Sayfa Kodları ve Uygulama Ekranı

```

child: ListView.builder(
  itemCount: _status.length,
  itemBuilder: (context, index) {
    return Padding(
      padding: const EdgeInsets.symmetric(horizontal: 8),
      child: InkWell(
        onTap: () => {},
        child: SizedBox(
          height: size.height * .1,
          child: Card(
            color: SWAIColors.blue,
            child: Padding(
              padding: const EdgeInsets.only(
                left: 15, right: 15),
              child: Row(
                mainAxisAlignment:
                  MainAxisAlignment.spaceBetween,
                children: [
                  Text(
                    "Exercise ${index + 1}",
                    style: const TextStyle(
                      fontSize: 17,
                      color: SWAIColors.white,
                      fontWeight: FontWeight.w600),
                  ),
                  ],
                ),
              )),
            ),
          );
        },
      );
    });
  },
);

```



```

return ListView.builder(
  itemCount: _status.length,
  itemBuilder: (context, index) {

    return Padding(
      padding: const EdgeInsets.symmetric(horizontal: 8),
      child: InkWell(
        onTap: () => {},
        child: SizedBox(
          height: size.height * .1,
          child: Card(
            color: SWAIColors.logout,
            child: Padding(
              padding: const EdgeInsets.only(
                left: 15, right: 15),
              child: Row(
                mainAxisAlignment:
                  MainAxisAlignment.spaceBetween,
                children: [
                  Text(
                    "Current Activity ${index + 1}",
                    style: const TextStyle(
                      fontSize: 17,
                      color: SWAIColors.white,
                      fontWeight: FontWeight.w700),
                  ),
                  Text(
                    "Day 1",
                    style: TextStyle(
                      color: SWAIColors.white,
                    ),
                  ),
                ],
              )));
  );
});

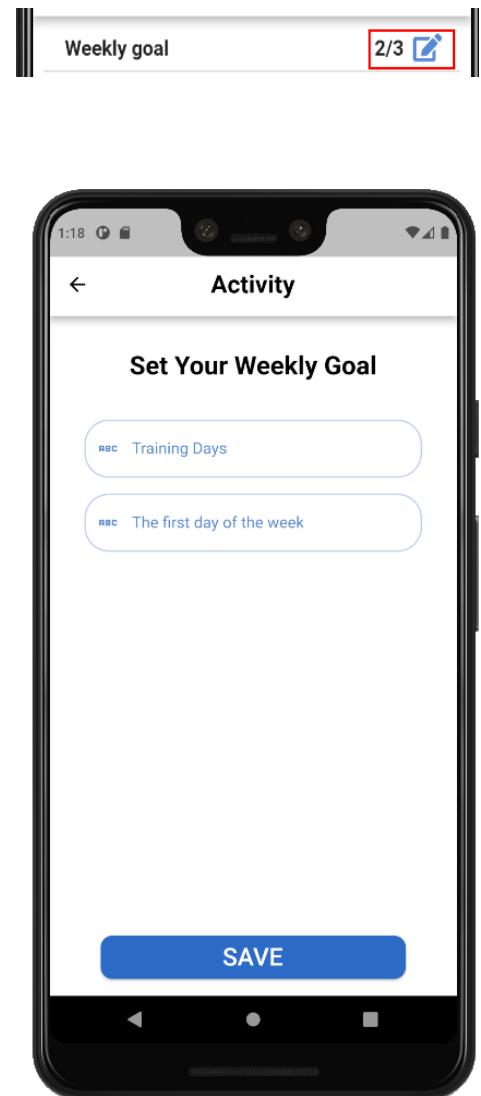
```

## 7.1. Weekly Goal Ekranı

Haftalık hedef ve set belirlenmesini sağlayan ekrandır. Activity sayfasındaki sağ üstte görünen ikona tıklandığında açılır.

```
CustomDropdown(
    selectedData: (e) => _selectedTrainingDay = e,
    dropdownItems: _trainingDays,
    label: "Training Days"),
SizedBox(
    height: size.height * .02,
),
CustomDropdown(
    selectedData: (e) => _selectedWeekDay = e,
    dropdownItems: _weekDays,
    label: "The first day of the week"),
],
),
],
),
ElevatedButton(
    style: ButtonStyle(
        backgroundColor:
            MaterialStateProperty.all(SWAIColors.blue),
        minimumSize: MaterialStateProperty.all(Size(
            size.width * .75,
            size.height * .055,
        )),
        shape:
            MaterialStateProperty.all<RoundedRectangleBorder>(
                RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(12.0),
                )),
        textStyle: MaterialStateProperty.all(
            const TextStyle(fontSize: 25)),
    child: const Text(
        'SAVE',
        style: TextStyle(fontWeight: FontWeight.w600),
    ),
    onPressed: () {
        print("_selectedTrainingDay: ${_selectedTrainingDay}");
        print("_selectedWeekDay: ${_selectedWeekDay}");
    },
),
),
```

**Custom Dropdown**  
widgetinden yapımında  
yararlanılmıştır.

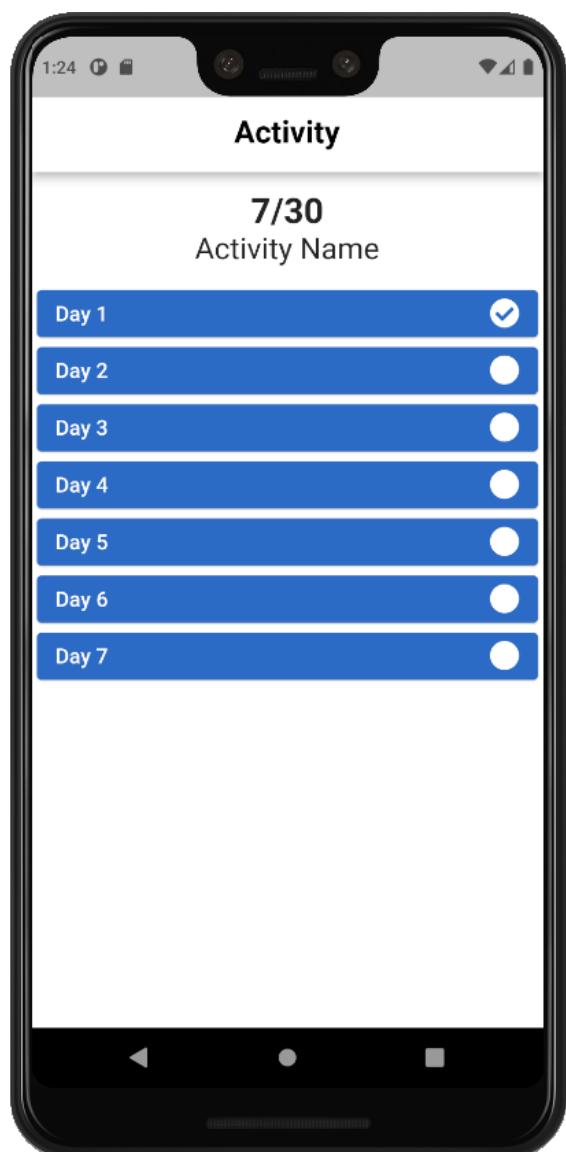


## 7.2. Activity Day Ekranı

İlgili egzersizin üzerine tıklandığında açılan ekrandır. Kişinin egzersizini tamamlama durumunu günlere bağlı olarak bu sayfa aracılığıyla görebilir.

Activity ana sayfasındaki Mini Calendar ve bu sayfadaki haftalık günlerin tamamlanma durumu yapısı ile kullanıcılar aktivitelerini daha kolay planlayıp uygulamaları amaçlanmıştır.

```
child: ListView.builder(  
  itemCount: _status.length,  
  itemBuilder: (context, index) {  
    return SizedBox(  
      height: constraints.maxHeight * .075,  
      child: Card(  
        color: SWAIColors.blue,  
        child: Padding(  
          padding: const EdgeInsets.only(  
            left: 15, right: 15),  
          child: Row(  
            mainAxisAlignment:  
              MainAxisAlignment.spaceBetween,  
            children: [  
              Text(  
                "Day ${index + 1}",  
                style: const TextStyle(  
                  fontSize: 17,  
                  color: SWAIColors.white,  
                  fontWeight: FontWeight.w600),  
              ),  
              Icon(  
                _status[index]  
                  ? FontAwesome5.check_circle  
                  : FontAwesome5.circle,  
                color: SWAIColors.white,  
              ),  
            ],  
          ),  
        )),  
      );  
    },  
  ),  
);
```



Activity Day Sayfası ve Uygulama Ekranı

### 7.3. Rest Ekranı

Aktivite set aralarında kullanıcının belirli süre ara vermesini amaçlayan ekrandır. Bir `_startTimer` fonksiyonu aracılığıyla ekran açıldığında otomatik olarak geri sayım başlar.

```
Row(  
  mainAxisAlignment: MainAxisAlignment.max,  
  mainAxisSize: MainAxisSize.max,  
  crossAxisAlignment: CrossAxisAlignment.baseline,  
  textBaseline: TextBaseline.alphabetic,  
  children: [  
    Text(  
      "00:${_counter}",  
      style: const TextStyle(  
        fontWeight: FontWeight.bold, fontSize: 56),  
    ),  
    const Text(  
      "sec",  
      style: TextStyle(fontSize: 36),  
    ),  
  ],  
,
```

```
initState() {  
  _startTimer();  
}  
  
int _counter = 30;  
  
late Timer _timer;  
  
void _startTimer() {  
  _timer = Timer.periodic(  
    const Duration(seconds: 1),  
    (timer) {  
      setState(  
        () {  
          _counter--;  
          if (_counter == 0) {  
            timer.cancel();  
          }  
        },  
      );  
    },  
  );  
}
```



Activity Rest Sayfası Kodları ve Uygulama Ekranı

## 8. Data Modelleri

Uygulama verilerini arka uç kısımdan getirirken verileri json yapısından çıkarmak için kullanılan Data Modellerini aşağıda bulabilirsiniz.

### 8.1. Token

Kayıt ve giriş sonrasında arka uç yapıdan servisler aracılığıyla Activation Code getirildiğinde onu çözmek için kullanılan model.

```
class Token {  
    String? token;  
  
    Token({this.token});  
  
    Token.fromJson(Map<String, dynamic> json) {  
        token = json['token'];  
    }  
  
    Map<String, dynamic> toJson() {  
        final Map<String, dynamic> data = new Map<String, dynamic>();  
        data['token'] = this.token;  
        return data;  
    }  
}
```

Token Class Modeli Kodları

### 8.2. Activation Code

Kayıt sonrasında veya kod gerektiren işlemler sonrasında arka uç yapıdan servisler aracılığıyla Activation Code getirildiğinde onu çözmek için kullanılan model.

```
class ActivationCode {  
    int? activationCode;  
  
    ActivationCode({this.activationCode});  
  
    ActivationCode.fromJson(Map<String, dynamic> json) {  
        activationCode = json['activationCode'];  
    }  
  
    Map<String, dynamic> toJson() {  
        final Map<String, dynamic> data = new Map<String, dynamic>();  
        data['activationCode'] = this.activationCode;  
        return data;  
    }  
}
```

ActivationCode Class Modeli Kodları

### 8.3. Sign Up

Arka uç yapıdan kayıt işlemi sonrası servislere gelen verileri çözmekte kullanılan model.

```
class SignUp {      You, 2 weeks ago • added activity page changes. ..
  String? email;
  String? password;
  String? username;
  int? activationCode;
  String? token;

  signUp(
    {this.email,
     this.password,
     this.username,
     this.activationCode,
     this.token});

  SignUp.fromJson(Map<String, dynamic> json) {
    email = json['email'];
    password = json['password'];
    username = json['username'];
    activationCode = json['activationCode'];
    token = json['token'];
  }

  Map<String, dynamic> toJson() {
    final Map<String, dynamic> data = new Map<String, dynamic>();
    data['email'] = this.email;
    data['password'] = this.password;
    data['username'] = this.username;
    data['activationCode'] = this.activationCode;
    data['token'] = this.token;
    return data;
  }
}
```

SignUp Class Modeli Kodları

## 9. Data Servisleri

İşlenmiş verileri arka uç yapıya gönderme ve ardından gerekli verileri çekme işlemlerinde kullanılan servis yapıları bu kısımda bulunmaktadır.

Servis yapılarını geliştirirken Flutter **Dio** kütüphanesinden yararlanılmıştır.

### 9.1. Log In

Login işleminde mail ve password yapısına uygun veriler gönderildiyse, veriler bu servis aracılığıyla arak uç kayıt yapısına gönderiliyor.

**logIn** fonksiyonu aracılığıyla kullanılır.

Bu fonksiyon **email** ve **password** parametrelerini alır.

Eğer arka uç yapıda kayıt işlemi başarılı oldusaya dönen tokeni şifreleyerek

**SWAISecureStorage** yapısı ile localstorage ye kaydedilir.

```
Future<dynamic> logIn(String email, String password) async {
  try {
    Response response = await Dio().patch(
      "${SWAIServices.URL}/signin",
      data: {"email": email, "password": password},
    );

    Token resData = Token.fromJson(response.data);
    SWAISecureStorage().writeSecureData("token", resData.token.toString());

    SWAISecureStorage()
      .readSecureData("token")
      .then((value) => print("Token: ${value}"));

    return resData.token;
  } on DioError catch (e) {
    Future.error(e.message);
    return null;
  }
}
```

logIn Servis Fonksiyonu Kodları

## 9.2. Sign Up

Sign Up sayfasında uygun veriler gönderildiyse ve aktivasyon kodu doğru girildiyse veriler bu servis aracılığıyla arak uç kayıt yapısına gönderiliyor.

**sendActivationCode** fonksiyonu aracılığıyla aktivasyon kodu işlemleri yapılıyor.

Bu yapı kişinin mailin gerçek sahibi olup olmadığını kontrol amaçlı kullanılır.

**data** parametresini alır ve bu parametre aracılığıyla arka uç yapıya kullanıcının email ve kullanıcı adı gönderilir.

Ardından arka uç yapıdan aktivasyon kodu döner ve kullanıcının girdiği kod ile bu kod karşılaştırılması için **SWAISecureStorage** yapısı ile localstorage ye kayıt edilir.

```
Future<ActivationCode> sendActivationCode(dynamic data) async {
  try {
    Response response = await Dio()
      .post("${SWAIServices.URL}/sendActivationCode", queryParameters: {
        "email": data['email'],
        "username": data['username'],
      });

    ActivationCode resData = ActivationCode.fromJson(response.data);

    if (resData.activationCode != null) {
      print("signUp: ${resData.activationCode}");

      SWAISecureStorage().writeSecureData(
        "activation_code", resData.activationCode.toString());
    } else {
      //toast message olacak.
      print("Activation kodu dömmüyör");
    }

    return resData;
  } on DioError catch (e) {
    return Future.error(e.message);
  }
}
```

sendActivationCode Servis Fonksiyonu Kodları

**signUp** fonksiyonu aracılığıyla kayıt işlemleri yapılıyor.  
localStorage den kullanıcının girdiği bilgiler çekilerek arka uç yapıya gönderilir.  
Ardından arka uç tarafından token döner ve bu token authentication işlemlerinde  
kullanılmak üzere şifrelenerek **SWAISecureStorage** yapısı ile localStorageye kayıt  
edilir.

```
Future<Token> signUp() async {
  try {
    Response response = await Dio().post(
      "${SWAIServices.URL}/signup",
      data: SWAILocalStorage("signup").getItemFromLocalStorage('signup_info'),
    );
    Token resData = Token.fromJson(response.data);
    SWAISecureStorage().writeSecureData("token", resData.token.toString());

    SWAISecureStorage()
      .readSecureData("token")
      .then((value) => print("Token: ${value}"));

    return resData;
  } on DioError catch (e) {
    return Future.error(e.message);
  }
}
```

signUp Servis Fonksiyonu Kodları

### **3 Sonuç**

#### **3 a. Başarı ile tamamlanan işler**

Ara dönem içerisinde planlanan ve başarı ile tamamlanan işleri sırası ile aşağıdaki tabloda bulabilirsiniz.

*Tablo3: Başarı ile tamamlanan işler.*

No	Başarı ile tamamlanan işler
1	Mobil ön uç sayfa tasarımları kodlanması.
2	Mobil ön uç localhost işlemleri.
3	Token kontrolü ve yapısı.
4	Sosyal medya, istatistik, kullanıcı, kayıt, giriş, token ve aktivasyon kodu işlemlerinin servis ve model yapıları.
5	Sosyal medya, kayıt ve giriş sayfalarının veritabanı ilişkilendirilmesi ve uygulanması.

#### **3 b. Başarı ile tamamlanamayan işler**

Ara dönem içerisinde planlanan ve başarı ile tamamlanamayan işler sırası ile aşağıda verilmiştir.

*Tablo4: Başarı ile tamamlanamayan işler.*

No	Başarı ile tamamlanamayan
1	Yapay zeka servisi tamamlanmadı - finale kadar tamamlanacak.
2	Tüm sistem AWS sistemi üzerine entegre olmadı - finale kadar tamamlanacak.
3	Anket seçeneğinin 3 seçenekten fazla olması (3 seçenekle sınırlandırıldı)

#### **3 c. Kişilerin görevlerini yerine getirme durumu**

Bitirme projesi ekibi görevi yerine getirme durumu gerekli bilgiler ile birlikte alt kısımdaki tabloda verilmiştir.

*Tablo5: Kişilerin görevlerini yerine getirme durumu.*

No	Numara	Ad Soyad	Projedeki Görevini Yerine Getirme Durumu
1	1803013027	Yunus Taha Yılmaz	%100
2	1803013016	Hilal Coşkun	%100
3	1803013010	Muhammed Furkan Gülsen	%100