



YILDIZ TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ
BÖLÜMÜ

BLM3722
YAZILIM MÜHENDİSLİĞİ DERSİ
DÖNEM PROJESİ

Hazırlayanlar:

Sevda KARAHAN

Beyza FINDIK

Hira Nur MORCA

Sümeyye GÜLDEMİR

Elif İNCE

KODLAR:

<https://drive.google.com/file/d/1cXzyEJwd7ymCE1iYZxYiIbiEMLQBVkaB/view?usp=sharing>

İÇİNDEKİLER

| | |
|---|-----------|
| 1. ÖN İNCELEME VE FİZİBİLİTE ANALİZİ | 4 |
| 1.1. Ekip Şeması (Roller) | 4 |
| 1.2. Risk Tablosu | 4 |
| 1.3. Toplantı Raporları | 5 |
| 1.4. Fizibilite Raporu | 6 |
| 1.4.1. Teknik Fizibilite | 7 |
| 1.4.2. Zaman Fizibilitesi | 8 |
| 1.4.2.a. Gantt Şeması | 8 |
| 1.4.3. Sosyal Fizibilite | 8 |
| 1.4.4. Ekonomik Fizibilite | 9 |
| 1.4.5. Yönetim Fizibilitesi | 9 |
| 1.4.6. Yasal Fizibilite | 9 |
| 2. SİSTEM ANALİZİ | 10 |
| 2.1. Taslak Veri Akış Diyagramı | 10 |
| 3. TASARIM | 10 |
| 3.1. Use-Case Diyagram | 10 |
| 3.2. UML Şeması | 11 |
| 3.3. Kullanım Senaryoları | 11 |
| 3.4. Aktivite Diyagramları | 14 |
| 3.5. Sequence Diyagramları | 16 |
| 3.6. Durum Diyagramı | 16 |
| 4. GERÇEKLEŞTİRME | 17 |
| 5. TEST | 20 |
| 5.1. Test Fonksiyonları | 20 |
| 5.2. Test Çıktıları | 27 |

Problem:

Bir el sanatları kursu işletmecisi olarak, çeşitli el sanatları kursları veren müşteri firmamız, mevcut iş süreçlerimizi optimize etmek ve verimliliği artırmak amacıyla bir yazılım çözümüne ihtiyaç duymaktadır. Mevcut süreçlerde, kurs programları oluşturma, öğretmen yönetimi, kursiyer kayıtları ve ödemeler gibi birçok işlem manuel olarak gerçekleştirilmektedir. Bu durum hem zaman alıcıdır hem de hata riskini artırmaktadır.

Ayrıca, kurs programlarını yönetirken öğretmenlerin ve kursiyerlerin gereksinimlerini dikkate almak ve esneklik sağlamak önemlidir. Örneğin, farklı öğretmenlerin farklı saatlerde çalışabilmesi, kursların hafta içi veya hafta sonu düzenlenebilmesi gibi durumlar göz önünde bulundurulmalıdır.

Bununla birlikte, kursiyerlerin kayıtları, kurs ödemeleri ve katılım durumlarının izlenmesi ve yönetilmesi de önemlidir. Kursiyerlerin kayıtları güncel ve doğru olmalıdır ve ödemelerin takibi kolay ve güvenilir bir şekilde yapılmalıdır.

Bu nedenle, mevcut süreçleri daha etkin hale getirmek, veri izleme ve raporlama işlemlerini iyileştirmek ve iş akışlarını otomatikleştirmek için bir yazılım çözümü geliştirilmesi gerekmektedir. Bu yazılım çözümü, kurs programlarının oluşturulmasını, öğretmen yönetimini, kursiyer kayıtlarını ve ödemeleri kolaylaştıracak ve izlenebilirlik sağlayacaktır.

Çözüm:

Mevcut sorunları çözmek ve iş verimliliğini artırmak için, bir el sanatları kursu otomasyon sistemi geliştirilmesi planlanmaktadır. Bu sistem, kurs programlarının oluşturulmasını, öğretmen yönetimini, kursiyer kayıtlarını ve ödemeleri kolaylaştıracak ve izlenebilirlik sağlayacaktır. Yazılım çözümü, aşağıdaki ana bileşenleri içerecektir:

1. **Kurs Programı Yönetimi:** Kurs programlarının oluşturulması, güncellenmesi ve yönetilmesini sağlayacak bir arayüz sağlanacaktır. Personel, kurs içeriğini ve zamanlamasını kolayca düzenleyebilecek ve güncelleyebilecektir.
2. **Öğretmen Yönetimi:** Öğretmenlerin bilgileri ve çalışma saatleri sisteme kaydedilecek ve kurs programlarına atanabilecektir. Bu sayede, uygun öğretmenler kurs programlarında görevlendirilecek ve öğretmenlerin maaş bilgileri yönetilebilecektir.
3. **Kursiyer Yönetimi:** Kursiyerlerin kayıtları ve ödemeleri kolayca takip edilebilecek ve yönetilebilecektir. Kursiyerlerin bilgileri güncel tutulacak ve ödeme işlemleri otomatikleştirilecektir.

Bu çözüm, iş süreçlerini optimize etmek, veri izleme ve raporlama işlemlerini iyileştirmek ve iş akışlarını otomatikleştirmek için tasarlanmıştır. Böylelikle, el sanatları kurslarını daha etkin bir şekilde yönetmek ve müşteri memnuniyetini artırmak hedeflenmektedir.

1. ÖN İNCELEME VE FİZİBİLİTE ANALİZİ

1.1.Ekip Şeması (Roller):



1.2. Risk Tablosu

| Risk | Olasılık | Etki | Önlem |
|---|----------|--------|---|
| Öğretmenlerin zamanının uyumsuzluğu | Yüksek | Yüksek | Öğretmenlerin boş zamanlarını belirlemek için anket yapıp buna göre bir program oluşturmak. |
| Kursiyer talebinde ani değişiklikler | Orta | Yüksek | Esnek fiyatlandırma politikası ve kurs programlarını ayarlamak için anketler. |
| Teknik sorunlar | Orta | Orta | Yazılım geliştirme sürecinde sık sık testler yapılması ve yedekleme planları oluşturmak. |
| Güvenlik açıkları | Düşük | Yüksek | Güvenlik protokolleri ve şifreleme yöntemleri kullanmak, düzenli güvenlik kontrolleri yapmak. |
| Kursiyer ve öğretmen bilgilerinin gizliliği | Orta | Yüksek | Veri gizliliği politikaları oluşturmak ve uygulamak, sadece yetkili personelin erişimine izin vermek. |
| Rekabet | Orta | Orta | Piyasa araştırması, benzersiz satış teknikleri. |

1.3. Toplantı Raporları:

1. Toplantı: Analiz Birimi Toplantısı

Tarih: 15/03/2024

Gündem:

- Proje gereksinimlerinin analizi.
- Kullanıcı ihtiyaçlarının belirlenmesi.
- Varolan iş süreçlerinin analizi.
- Analiz sürecinin planlanması.

Özet:

Toplantıda, proje gereksinimlerinin analizi için gereken adımlar ve yöntemler tartışıldı. Kullanıcı ihtiyaçlarının belirlenmesi ve var olan iş süreçlerinin analizi için bir plan oluşturuldu. Analiz ekibi, projenin kapsamını ve hedeflerini netleştirmek için ilgili kişilerle görüşmeler yapacak ve gereksinimleri belirleyecektir.

2. Toplantı: Tasarım Birimi Toplantısı

Tarih: 21/03/2024

Gündem:

- Analiz sonuçlarına dayalı tasarım gereksinimlerinin belirlenmesi.
- Sistem mimarisi ve veritabanı tasarımı.
- Kullanıcı arayüzü tasarımı.
- Tasarım sürecinin planlanması.

Özet:

Toplantıda, analiz sonuçlarına dayalı olarak sistem mimarisi, veritabanı tasarımı ve kullanıcı arayüzü tasarımı ele alındı. Tasarım ekibi, gereksinimleri karşılamak için uygun bir sistem mimarisi oluşturacak ve kullanıcıların kolayca kullanabileceği bir arayüz tasarlayacaktır.

3. Toplantı: Proje Bitimi Toplantısı

Tarih: 26/04/2024

Gündem:

- Proje hedeflerinin değerlendirilmesi.

- Gerçekleştirilen işlerin gözden geçirilmesi.
- Projenin başarı kriterlerinin değerlendirilmesi.
- Sonraki adımların belirlenmesi.

Özet:

Toplantıda, proje hedeflerinin ve başarı kriterlerinin değerlendirilmesi yapıldı. Gerçekleştirilen işlerin gözden geçirilmesi ve projenin sonuçlarının müşteri temsilcisiyle birlikte değerlendirilmesi sağlandı. Sonraki adımlar ve proje kapanışıyla ilgili planlar belirlendi.

1.4. Fizibilite Raporu

1.4.1. Teknik Fizibilite:

a. Fizibilite Matrisi:

| FİZİBİLİTELER | Senaryo 1 | Senaryo 2 | Senaryo 3 |
|----------------------------|---|--|--|
| Teknik Fizibilite | Sistemi geliştirmek için işletim sistemi MacOS, programlama dili olarak Python seçilecektir. Teknik anlamda oldukça yeterli olan bu senaryoda aksayabilecek fonksiyonlar bulunabilir, Veritabanı olarak açık kaynak, yüksek performans ve ücretsiz hizmet sunması dolayısıyla PostgreSQL seçilmiştir. | Sistemi geliştirmek için işletim sistemi Linux, programlama dili olarak Java kullanılacaktır. Çalışanların tecrübesi olmasına rağmen Seçilen bu dilde bir sistemin gerçekleşmesi ve kaldırılması nispeten fazla zaman alacaktır. Sistemin veritabanı olarak MySQL hizmeti sunması adına PlanetScale kullanılacaktır, çeşitli ödeme planları mevcuttur. | Windows işletim sisteminde Flutter kullanılarak sistem gerçekleştirilecektir. Flutter'ın kullanım kolaylığı sağlamanın yanı sıra oldukça hızlı sonuçlar alınması sistemi geliştirme aşamasında bir artıdır. Sistemin veritabanı olarak SQLite kullanılmaktadır, fonksiyonları sebebiyle bu seçenek oldukça avantajlıdır. |
| Puanlama | 20 | 25 | 35 |
| Ekonomik Fizibilite | MacOS ve Python alanında sistem gerçekleştirebilecek bilgi ve deneyime sahip olmayan ekip üyeleri nedeniyle 2000TL eğitim masrafı vardır. Bunun dışında ekibin MacOS'a sahip araç gereçler satın alması gerekecektir. Gerekli lisanslar ve ürünlerle birlikte 4 ay için toplamda 200.000TL | Linux işletim sistemi açık kaynak olduğundan dolayı ücretli değildir. Bu senaryoda tercih edilen PlanetScale servisinde hizmet ücretleri 30USD-600USD arası değişmektedir. 5 aylık geliştirme süreci için masraf 10.000TL olarak belirlenmiştir. | Windows işletim sistemi lisans ücreti ödenmesi gerekmektedir. SQLite için herhangi bir ücret ödemesi yapılması gerekmemektedir. Bu senaryoda ödenmesi gereken toplam maliyet 40.000TL'dir. |
| | ödenek ayrılması gerekmektedir. | | |
| Puanlama | 10 | 15 | 10 |

| Zaman Fizibilitesi | 4-5 Ay | 4-5 Ay | 3-4 Ay |
|---------------------------|---|--|--|
| Puanlama | 20 | 20 | 25 |
| Yasal Fizibilite | Apple'ın lisans sözleşmesine göre, macOS sadece Apple donanımları üzerinde çalışabilir ve diğer bilgisayarlara yüklenmesi yasal olarak mümkün değildir. | Linux açık kaynak işletim sistemi olduğundan dolayı tamamen yasal bir şekilde kullanılabilir. Özel izinler konusunda diğer senaryolara göre daha avantajlıdır ve güvenlidir. | Windows kullanımı için satın alınan lisanslar, farklı kullanım sebepleri için farklı versiyonlar sunmaktadır ve versiyonlara göre alınması gereken özel izinler değişiklik gösterebilir, bu yüzden dikkatli olunmalıdır. |
| Puanlama | 5 | 15 | 10 |
| Sonuç | 55 | 75 | 80 |

Fizibilite matrisindeki puanlamalar, karar verme sürecinde öncelikler göz önünde bulundurularak yapılmıştır. Matriste bulunan üç senaryo içinden, verilen puanlar göz önünde bulundurularak 3. senaryo uygun görülmüş ve projenin bu senaryoya göre gerçekleşmesi uygun görülmüştür.

Sistemin gerçekleşmesi sürecinde, işletim sistemi olarak oldukça yaygın bir kullanıma sahip olan Windows işletim sistemi seçilmiştir. Ekip, Flutter ile sistem gerçekleştirme konusunda yeterli bilgiye sahiptir ve bu framework'un seçilmesinde en önemli faktör budur. Böylelikle geliştirme süreci daha hızlı ve sorunsuz şekilde ilerlemiştir. Veritabanı olarak seçilen SQLite, Flutter ile beraber oldukça sık kullanılan bir veritabanı yönetim sistemidir. Flutter ile entegresinin kolay olması ve maliyeti düşük olmasından dolayı oldukça avantajlı bir opsiyon durumundadır.

Yazılım fizibilitesi kapsamında proje geliştirme sürecinde kullanılacak destek elemanları şu şekildedir:

Kullanılan programlama dili ve framework: Flutter

Kullanılan Veri Tabanı: SQLite

İşletim Sistemi: Windows

Geliştirme Ortamı: VSCode

Donanım Gereksinimleri:

- Huawei, Mate 14, 512 GB SSD, 16 GB RAM
- Asus Zenbook, 1TB SSD, 16 GB RAM
- Lenovo IdeaPad 3, 512GB SSD, 8 GB RAM
- Asus Vivobook 15 256 GB SSD, 8 GB RAM

1.4.4. Ekonomik Fizibilite

Personel Giderleri:

- Proje Yöneticisi: 100\$ *42 gün
- Sistem Analisti: 100\$ * 42 gün
- Arayüz Geliştirici: 100\$ * 42 gün
- Veri Tabanı Geliştirici: 100\$ * 42 gün
- Uygulama Programlayıcı: 100\$ * 42 gün

Ek Giderler:

Yazılım-Program Giderleri: 30.00\$/ay (Server)

Toplam Maliyet:

Personel Giderleri: 500\$ * 42 = 21.000\$

Ek giderler: 30.00\$/ay

1.4.5. Yönetim Fizibilitesi

Öncelikle eğitici kurumlar, bu sistem sayesinde verdiği kursları daha iyi yönetebileceklerdir. Kursiyerlerin seçtiği eğitimler sistem üzerinden görüntülenebildiği için, hangilerinin daha çok talep gördüğü, hangilerinin katılıma açık olduğu gibi bilgiler sistemde oldukça rahat gözlemlenebilecektir. Böylece kurs yönetimi kurslarını daha iyi yönetebilecek ve kursiyerlerin ihtiyaçlarını karşılayabilmek için gerekli önlemleri alabileceklerdir.

Bu sistem aynı zamanda kurs yönetimine zaman ve maliyet tasarrufu sağlamaktadır. Kursların katılım durumları, eğitim ücretleri gibi bilgilerin otomatik olarak takip edilmesi, yönetimin bu işleri manuel olarak yapmasına gerek kalmadan verimli bir şekilde yönetmelerine olanak sağlar.

1.4.6. Yasal Fizibilite

Sistemimiz kullanıcıların isim, soy isim, e-posta adresi ve parola gibi özel bilgilerinin korunmasını garanti eden bir sistemdir. 6698 Kişisel Verileri Koruma Kanunu'na ve yasal gerekliliklere uygun olarak faaliyet göstermektedir.

Kullanıcı bilgileri, veri tabanında güvenli bir şekilde saklanmaktadır. Kullanıcı sisteme giriş yaparken verdiği bilgiler veri tabanından kontrol edilmektedir.

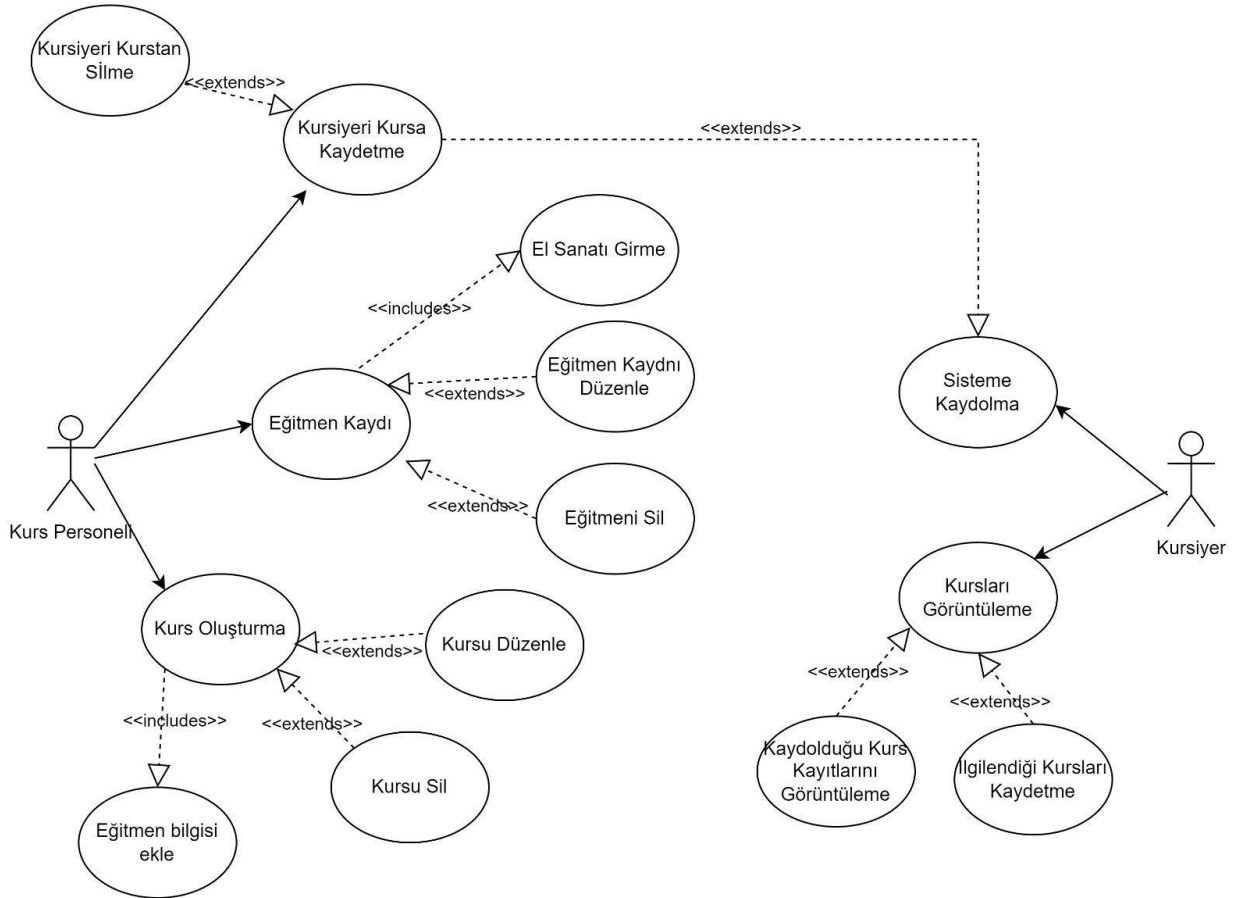
2. SİSTEM ANALİZİ

2.1. Taslak Veri Akış Diyagramı

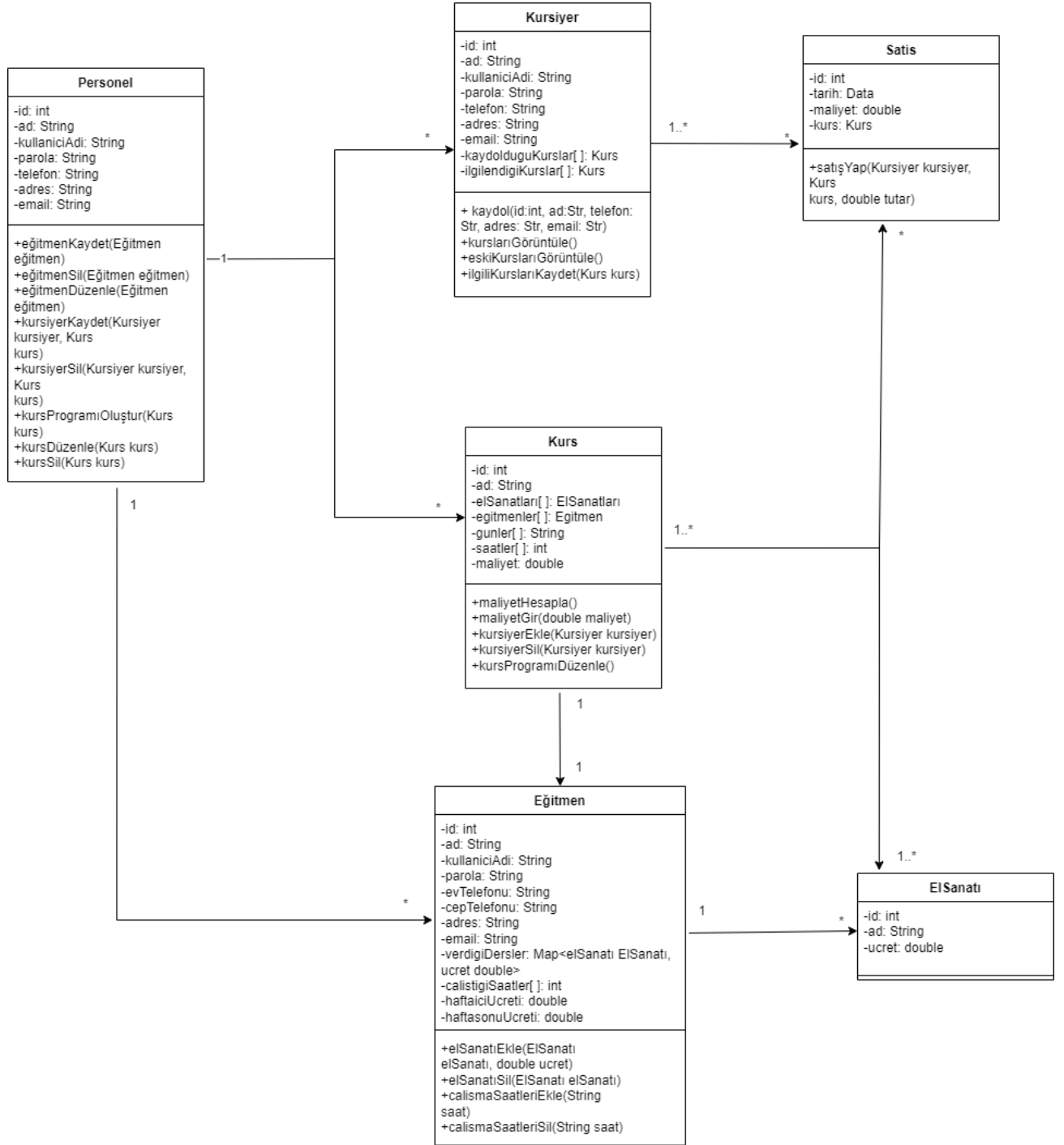


3. TASARIM

3.1. Use-Case Diyagramı



3.2. UML Şeması



3.3. Kullanım Senaryoları

1. Kursiyerin Sisteme Kaydı:

- **Senaryo:** Kursiyer, uygulama aracılığıyla sisteme kaydolar.
- **Adımlar:**
 1. Kursiyer platforma kaydolmak için uygulamayı ziyaret eder.

2. Kayıt formunu doldurur.
3. Sistem kursiyeri sisteme kaydeder.
4. Kayıt işlemi tamamlandıktan sonra kursiyer, kişisel bilgilerini sistemden görüntüleyebilir.
5. Kursiyer sistemden açılan kursları görüntüleyebilir ve ilgisini çeken kursları kaydedebilir.

○ **Alternatif Akış:**

2a. Kursiyerin sistemde kaydı var.

1. Sistem uyarı mesajı döndürür.

2. Kursiyerin Kursa Kaydolması:

- **Senaryo:** Kursiyer, kurs merkezine giderek ilgilendiği kursa kaydolur.

○ **Adımlar:**

1. Kursiyer kursa kaydolmak için kurs merkezini ziyaret eder.
2. Personel, kursiyerin öğrenmek istediği el sanatlarını, uygun zamanını ve ödeyebileceği bütçeyi sisteme girer, uygun kurslar listelenir.
3. Kursiyer listelenen kursları gözden geçirir ve uygun bir kursu seçer.
4. Personel sistemden ilgili kurs kayıt işlemlerini gerçekleştirir.
5. Kursiyer kart veya nakit ile ödemesini gerçekleştirir.
6. Sistem satış bilgilerini kaydeder.
7. Kayıt işlemi tamamlandıktan sonra kursiyer, kurs kayıtlarını sistemden görüntülenebilir.

○ **Alternatif Akış:**

3a. Kursiyerin istediği el sanatlarını içeren kurslar mevcut değil.

1. Kursiyerin ilgisini çekebilecek başka kurslara yönlendirilmesi yapılır.

3. Kurs Programı Oluşturma:

- **Senaryo:** Personel, yeni bir kurs programı oluşturur.

○ **Adımlar:**

1. Personel, yönetici paneline giriş yapar.
2. Yeni bir kurs programı oluşturmak için "Kurs Ekle" seçeneğine tıklar.
3. Kursun içeriğinde hangi el sanatlarının olacağını belirler.
4. Kursun hafta içi mi hafta sonu mu olacağını belirler.
5. Kursu verebilecek uygun eğitmenler içerisinde uygun öğretmeni kursa atar.
6. Sistem girilen kurslara ve öğretmenlere göre maliyeti hesaplayıp gösterir.
7. Programı onaylar ve yayımlar.

○ **Alternatif Akış:**

5a. Uygun öğretmen yoksa

1. Personel öğretmenlerle konuşur ve kursu verebilecek bir öğretmen bulursa sistemden öğretmenin bilgilerini düzenler ve kursa atamasını yapar.

6a. Kursun ücreti değiştirilmek istendi.

1. Personel önerilen tutar haricinde kendi belirlediği ücreti de girebilir.

4. Öğretmen Kaydı:

- **Senaryo:** Personel yeni bir öğretmeni platforma kaydeder.
- **Adımlar:**
 1. Kursta eğitim vermesi kararlaştırılan öğretmenlerle iletişime geçilir.
 2. Kişisel bilgileri personel tarafından sisteme işlenir.
 3. Çalışabileceği saat aralıklarını seçilir.
 4. Verebileceği dersleri seçer ve hizmet bedelini belirtir.
 5. Öğretmenin kayıt işlemi tamamlanır.
 6. Yeni bir kurs açılacağı zaman öğretmenin uygunluğu ve yetkinliğine göre kursa eğitmen olarak atanır.

- **Alternatif Akış:**

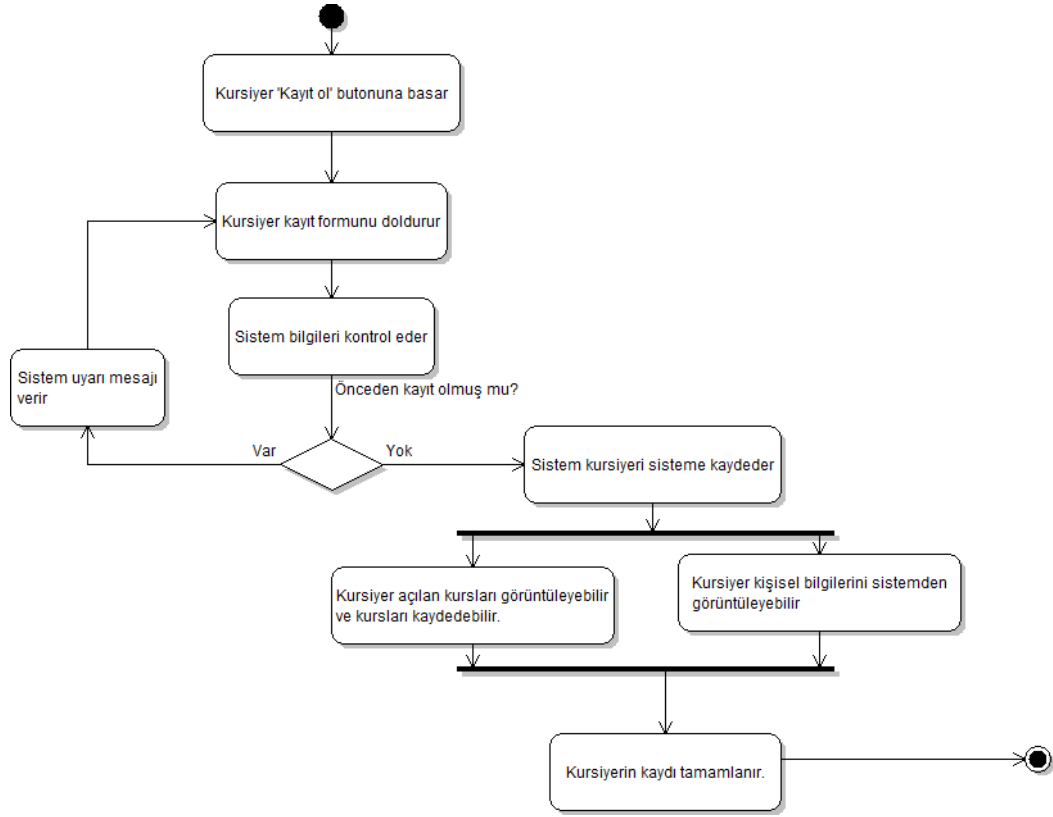
3a. Öğretmen daha sonra çalışmak istediği saatleri değiştirmek ister.

1. Personel, öğretmen kaydını günceller.
2. Öğretmen bir kursa atanmış ise kurs kaydını günceller, saatlerini değiştirir.

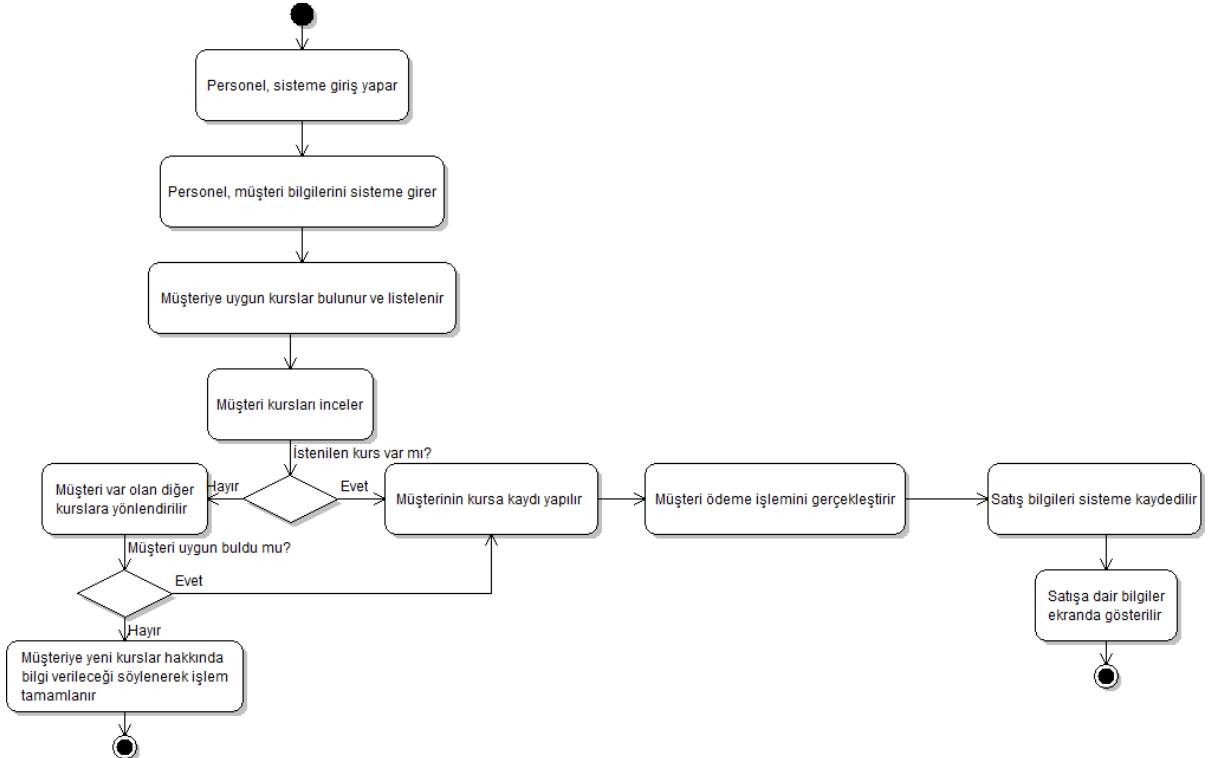
4a. Eğitimci daha sonra bazı dersleri vermekten vazgeçer.

3. Personel, öğretmen kaydını günceller.
4. Öğretmen bir kursa atanmış ise kurs kaydını günceller, başka bir öğretmen atar.

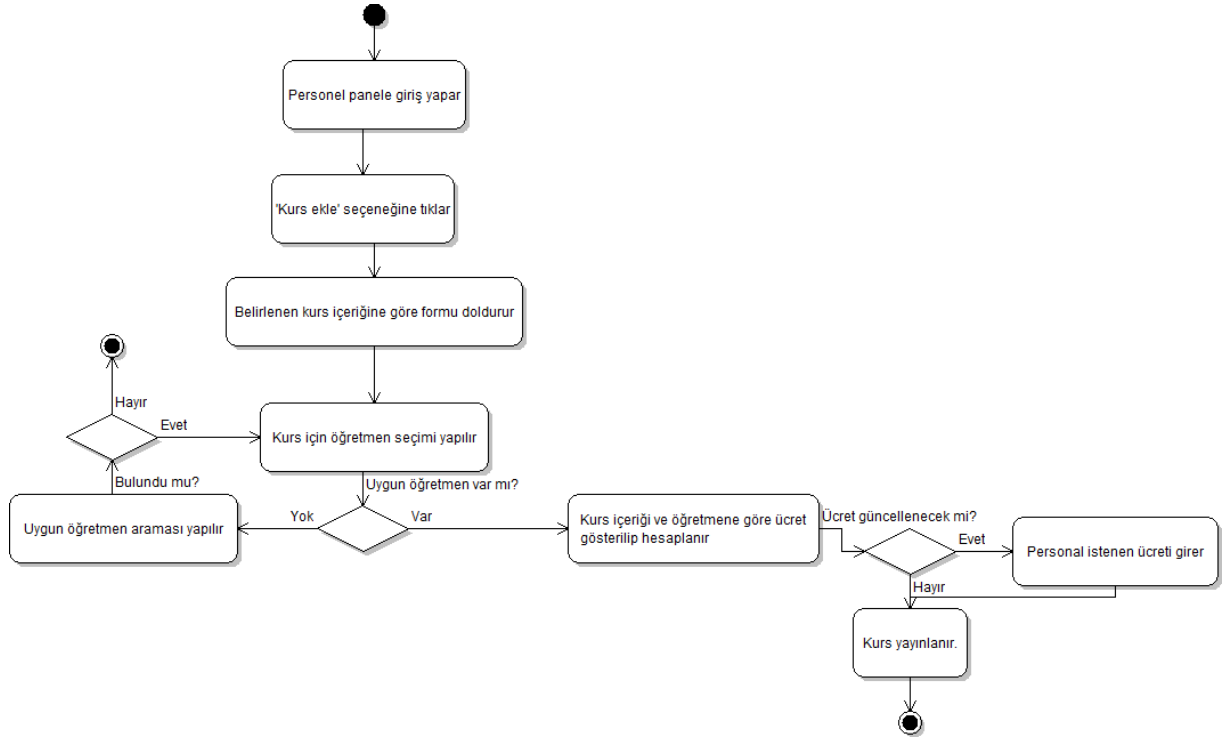
3.4 Aktivite Diyagramları



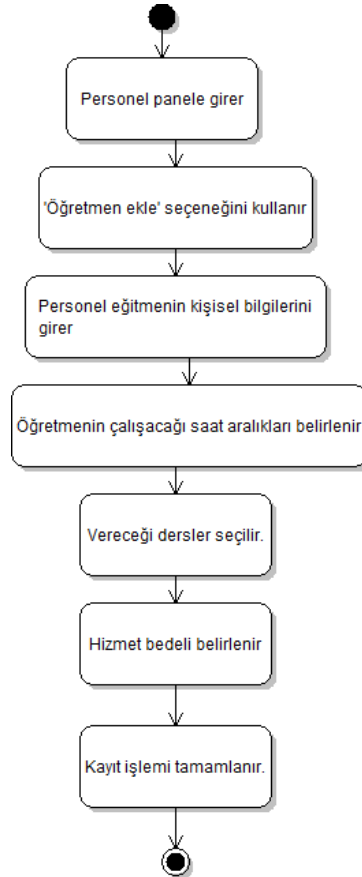
Kursiyerin kayıt olma sürecine ait aktivite diyagramı.



Personelin müşteriye kursa kaydetmesi sürecine ait aktivite diyagramı.

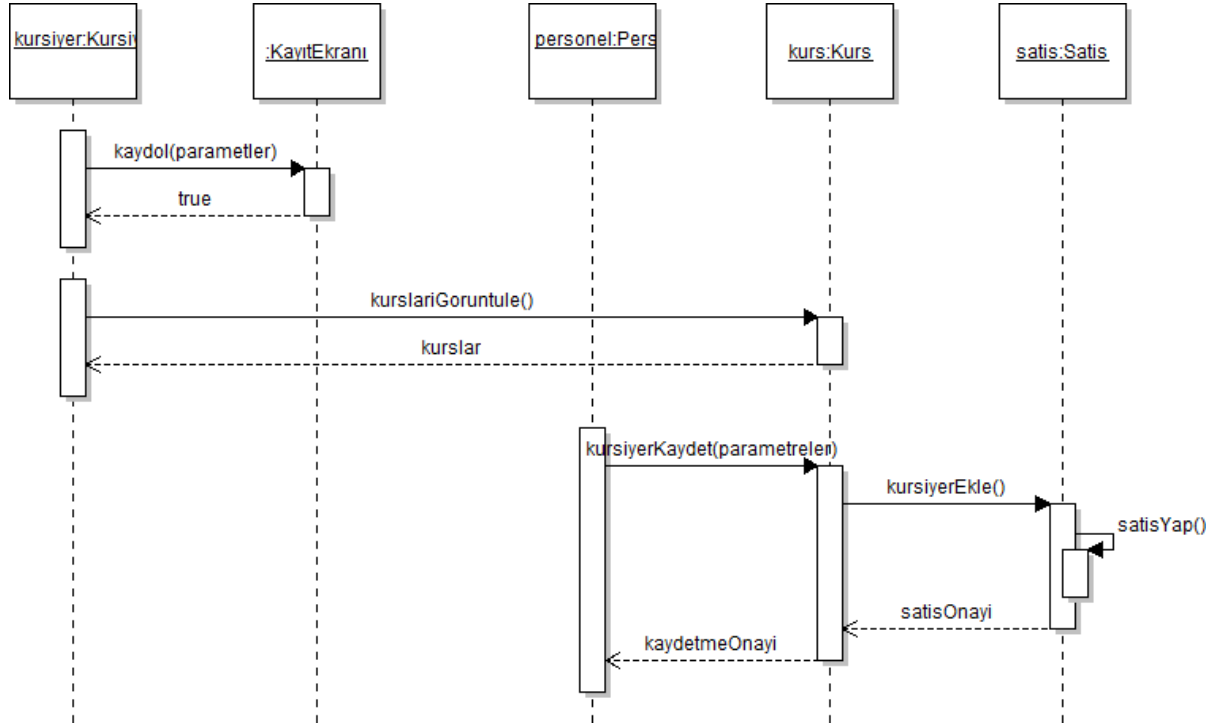


Personelin sisteme yeni kurs eklemesine ait sürecin aktivite diyagramı.



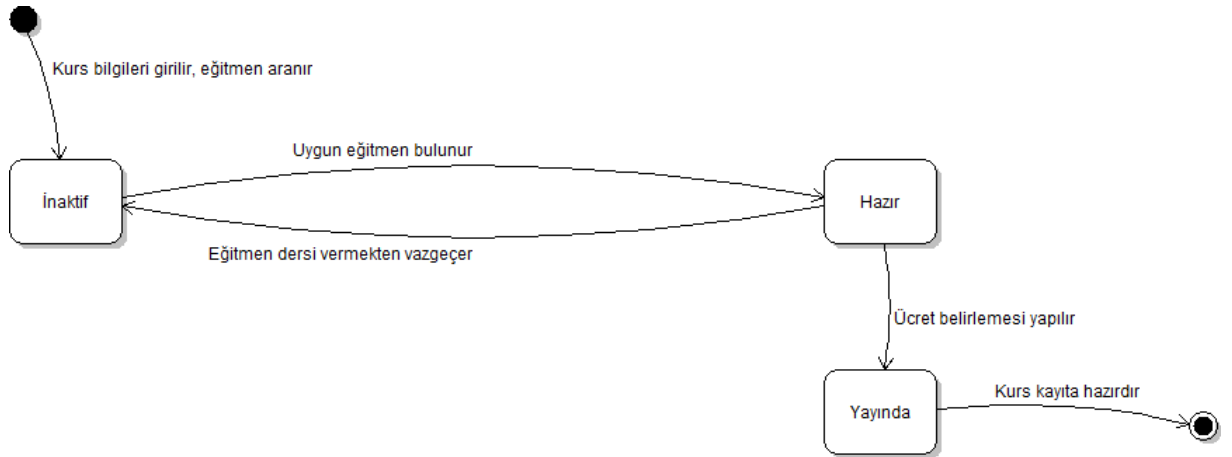
Personelin sisteme yeni öğretmen eklemesi sürecine ait aktivite diyagramı.

3.5 Sequence Diyagramları



Kursiyerin kaydının yapılması ve personel tarafından kursa kayıt edilmesine ait diyagram.

3.6 Durum Diyagramı



Kursun hazırlanması aşamasında kursa ait durumlar ve farklı senaryolara göre değişime ait durum diyagramı.

4. GERÇEKLEŞTİRME

Giriş Ekranı:



Kullanıcı uygulamaya giriş yaptığında karşısına giriş sayfası çıkar.

Kullanıcı Kayıt:



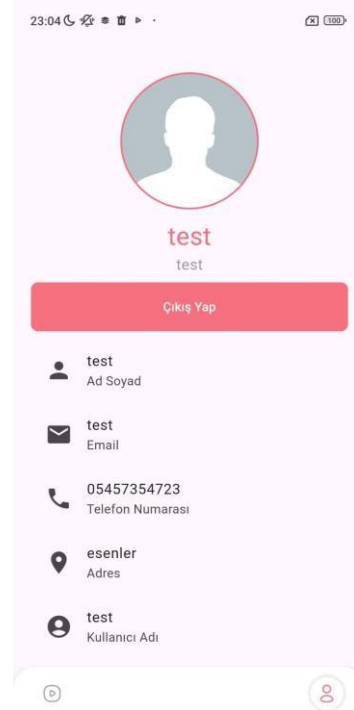
Kullanıcı kayıt ol tuşuna bastığında kayıt olma sayfasına yönlendirilir.

Kullanıcı Giriş:



Kullanıcı giriş yap butonuna bastığında giriş sayfasına yönlendirilir.

Kullanıcı Profili:



Kullanıcı giriş yaptığında profil sayfasına yönlendirilir.

Kullanıcının Ödediği Kurslar:



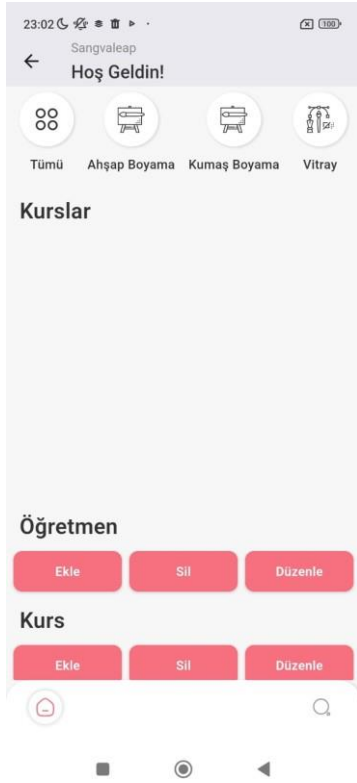
Kullanıcı
profilinden
daha önce
kaydolduğu
kursları
görüntüleyebilir.

Admin Girişi:



Admin, giriş
sayfasından
“admin girişi”
ile admin
sayfasına
yönlendirilir.
Admin giriş
yapar.

Admin İçin Kurs ve Öğretmen Aksiyonları:



Admin ana sayfasında var olan kursları görüntüleyebilir. Öğretmen ekle/sil/düzenleme ve kurs ekle/sil/düzenle. kursları işlemlerini gerçekleştirebilir.

Kurs Arama:

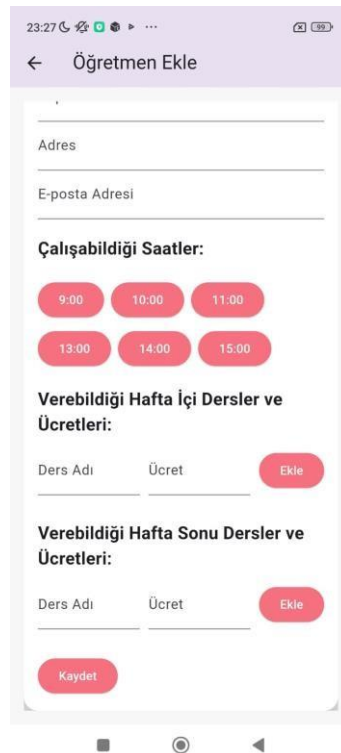


Admin, kursiyerin istediği günleri el sanatlarını ve bütçeyi girerek uygun listeler ve kursiyeri kaydeder.

Kurs Ekleme:

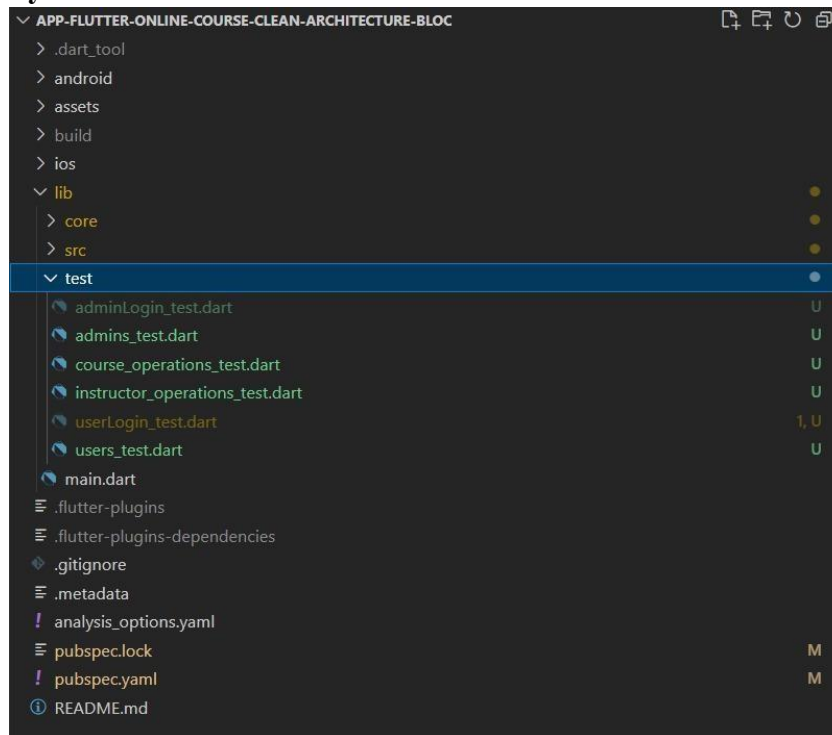


Öğretmen Ekleme:



5. TEST

5.1 Test Fonksiyonları



adminLogin_test.dart

```
import 'package:flutter_test/flutter_test.dart';
import 'package:online_course/core/services/database_helper.dart';

void main() {
  group('Admin Authentication Tests', () {
    late DatabaseHelper dbHelper;

    setUp(() {
      dbHelper = DatabaseHelper.instance;
    });

    test('Correct admin credentials should authenticate successfully', () async {
      // Define admin credentials
      String adminUsername = 'admin';
      String adminPassword = 'admin';

      // Attempt to authenticate with the admin credentials
      bool isAuthenticated = await dbHelper.authenticateAdmin(adminUsername,
adminPassword);
```

```

    // Expect authentication to be successful
    expect(isAuthenticated, true);
  });

  test('Incorrect admin credentials should not authenticate', () async {
    // Define incorrect admin credentials
    String wrongAdminUsername = 'admin';
    String wrongAdminPassword = 'wrongpassword';

    // Attempt to authenticate with the incorrect admin credentials
    bool isAuthenticated = await dbHelper.authenticateAdmin(wrongAdminUsername,
wrongAdminPassword);

    // Expect authentication to fail
    expect(isAuthenticated, false);
  });

  tearDown(() {
    // Clean up any database changes made during testing
    // (This might not be necessary if the tests are run in isolation)
  });
});
}

```

userLogin_test.dart

```

import 'package:flutter_test/flutter_test.dart';
import 'package:online_course/core/services/database_helper.dart';
import 'package:online_course/src/JSON/users.dart';

void main() {
  group('User Authentication Tests', () {
    late DatabaseHelper dbHelper;

    setUp(() {
      dbHelper = DatabaseHelper.instance;
    });

    test('Correct user credentials should authenticate successfully', () async {
      // Create a user with known credentials
      Users testUser = Users(userName: 'testuser', password: 'password123');

      // Insert the test user into the database
      await dbHelper.createUser(testUser);

      // Attempt to authenticate with the test user credentials
      bool isAuthenticated = await dbHelper.authenticate(testUser);
    });
  });
}

```

```

    // Expect authentication to be successful
    expect(isAuthenticated, true);
  });

  test('Incorrect user credentials should not authenticate', () async {
    // Create a user with known credentials
    Users testUser = Users(usrName: 'testuser', password: 'wrongpassword');

    // Attempt to authenticate with the test user credentials
    bool isAuthenticated = await dbHelper.authenticate(testUser);

    // Expect authentication to fail
    expect(isAuthenticated, false);
  });

  tearDown(() {
    // Clean up the test database by deleting the test user
    dbHelper.database.then((db) {
      db!.delete(
        'users',
        where: 'usrName = ?',
        whereArgs: ['testuser'],
      );
    });
  });
}

admins_test.dart

import 'package:flutter_test/flutter_test.dart';
import 'package:online_course/src/JSON/admins.dart';

void main() {
  group('Admins class tests', () {
    test('Admin object creation', () {
      final admin = Admins(
        userName: 'test_admin',
        password: 'test_password',
      );

      expect(admin.userName, 'test_admin');
      expect(admin.password, 'test_password');
    });

    test('Admin fromMap method', () {
      final adminMap = {
        "userName": "admin",

```

```

        "password": "admin123",
    };

    final admin = Admins.fromMap(adminMap);

    expect(admin.userName, "admin");
    expect(admin.password, "admin123");
  });

  test('Admin toMap method', () {
    final admin = Admins(
      userName: "admin",
      password: "admin123",
    );

    final adminMap = admin.toMap();

    expect(adminMap["userName"], "admin");
    expect(adminMap["password"], "admin123");
  });
}

users_test.dart

import 'package:flutter_test/flutter_test.dart';
import 'package:online_course/src/JSON/users.dart';

void main() {
  group('Users class tests', () {
    test('User object creation', () {
      final user = Users(
        usrName: 'test_user',
        password: 'test_password',
      );

      expect(user.usrName, 'test_user');
      expect(user.password, 'test_password');
    });

    test('User fromMap method', () {
      final userMap = {
        "usrId": 1,
        "fullName": "John Doe",
        "email": "john.doe@example.com",
        "phoneNumber": "1234567890",
        "address": "123 Main St",
        "usrName": "john_doe",
        "usrPassword": "password123",
      };

```

```

    };

    final user = Users.fromMap(userMap);

    expect(user.userId, 1);
    expect(user.fullName, "John Doe");
    expect(user.email, "john.doe@example.com");
    expect(user.phoneNumber, "1234567890");
    expect(user.address, "123 Main St");
    expect(user.userName, "john_doe");
    expect(user.password, "password123");
  });

  test('User toMap method', () {
    final user = Users(
      userId: 1,
      fullName: "John Doe",
      email: "john.doe@example.com",
      phoneNumber: "1234567890",
      address: "123 Main St",
      userName: "john_doe",
      password: "password123",
    );

    final userMap = user.toMap();

    expect(userMap["userId"], 1);
    expect(userMap["fullName"], "John Doe");
    expect(userMap["email"], "john.doe@example.com");
    expect(userMap["phoneNumber"], "1234567890");
    expect(userMap["address"], "123 Main St");
    expect(userMap["userName"], "john_doe");
    expect(userMap["usrPassword"], "password123");
  });
});
}

course_operations_test.dart

import 'package:flutter_test/flutter_test.dart';
import 'package:online_course/core/services/database_helper.dart';

void main() {
  group('Course Operations Tests', () {
    late DatabaseHelper dbHelper;

    setUp(() {
      dbHelper = DatabaseHelper.instance;
    });
  });
}

```



```

test('Inserting new course', () async {
  Map<String, dynamic> newCourse = {
    'instructorId': 1, // Örnek instructor ID'si
    'name': 'Sample Course',
    'description': 'This is a sample course.',
    'price': 100.0,
    'image': 'sample_course.jpg',
    'startDate': '2024-01-01',
    'endDate': '2024-12-31',
    'lessonIds': '1,2,3', // Örnek ders ID'leri
    'craftDays': 'Monday,Wednesday,Friday', // Örnek günler
  };

  int courseId = await dbHelper.insertCourse(newCourse);

  expect(courseId, isNotNull);
});

test('Deleting existing course', () async {
  // Öncelikle bir kurs ekleyelim
  Map<String, dynamic> newCourse = {
    'instructorId': 1, // Örnek instructor ID'si
    'name': 'Sample Course',
    'description': 'This is a sample course.',
    'price': 100.0,
    'image': 'sample_course.jpg',
    'startDate': '2024-01-01',
    'endDate': '2024-12-31',
    'lessonIds': '1,2,3', // Örnek ders ID'leri
    'craftDays': 'Monday,Wednesday,Friday', // Örnek günler
  };

  int courseId = await dbHelper.insertCourse(newCourse);

  // Eklene kursu silelim
  int rowsAffected = await dbHelper.deleteCourse(courseId);

  expect(rowsAffected, equals(1));
});
});

instructor_operations_test.dart

import 'package:flutter_test/flutter_test.dart';
import 'package:online_course/core/services/database_helper.dart';

void main() {
  group('Instructor Operations Tests', () {

```

```

late DatabaseHelper dbHelper;

setUp(() {
  dbHelper = DatabaseHelper.instance;
});

test('Inserting new instructor', () async {
  Map<String, dynamic> newInstructor = {
    'name': 'John Doe',
    'homePhone': '1234567890',
    'cellPhone': '0987654321',
    'address': '123 Main St',
    'email': 'john.doe@example.com',
  };

  int instructorId = await dbHelper.insertInstructor(newInstructor);

  expect(instructorId, isNotNull);
});

test('Deleting existing instructor', () async {
  // Öncelikle bir instructor ekleyelim
  Map<String, dynamic> newInstructor = {
    'name': 'John Doe',
    'homePhone': '1234567890',
    'cellPhone': '0987654321',
    'address': '123 Main St',
    'email': 'john.doe@example.com',
  };
  int instructorId = await dbHelper.insertInstructor(newInstructor);

  // Eklenen instructor'ü silelim
  int rowsAffected = await dbHelper.deleteInstructor(instructorId);

  expect(rowsAffected, equals(1));
});
}

```

5.2 Test Çıktıları

```
(base) PS C:\FlutterProject\new\app-flutter-online-course-clean-architecture-bloc> flutter test lib/test/adminLogin_test.dart
00:02 +2: All tests passed!
(base) PS C:\FlutterProject\new\app-flutter-online-course-clean-architecture-bloc> flutter test lib/test/userLogin_test.dart
00:05 +2: All tests passed!
(base) PS C:\FlutterProject\new\app-flutter-online-course-clean-architecture-bloc> flutter test lib/test/admins_test.dart
00:03 +3: All tests passed!
(base) PS C:\FlutterProject\new\app-flutter-online-course-clean-architecture-bloc> flutter test lib/test/users_test.dart
00:02 +3: All tests passed!
(base) PS C:\FlutterProject\new\app-flutter-online-course-clean-architecture-bloc> flutter test lib/test/course_operations_test.dart
00:03 +2: All tests passed!
(base) PS C:\FlutterProject\new\app-flutter-online-course-clean-architecture-bloc> flutter test lib/test/instructor_operations_test.dart
00:02 +2: All tests passed!
```