

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
KARACABEY MESLEK YÜKSEKOKULU

PROJE ADI

Kütüphane Otomasyonu

Projeyi Hazırlayan

Elif Tuana Aykırı - 262284043

Proje Danışmanı

Öğr. Gör. Koray ÇOŞKUN

Bilgisayar Teknolojileri Bölümü

Bilgisayar Programcılığı Programı

Haziran, 2024

Bursa

ÖZET

Bu proje, C# programlama dili ve MSSQL veri tabanı kullanılarak geliştirilmiş bir kütüphane otomasyon sistemidir. Projede, hem yönetici hem de kullanıcı girişi yapılabilmektedir.

Projede, kullanıcıların (üye) ve yöneticilerin (admin) giriş yapabilmesini sağlayan bir ekran var. Giriş yapamayan kullanıcılar, üye kayıt formuyla yeni bir hesap oluşturabilir. Yönetici, kütüphaneye yeni kitap ekleyebilir ve kütüphanede bulunan kitapları güncelleyebilir. Ayrıca yeni üyeler ve adminler ekleyebilir, kayıtlı üyelerin ve adminlerin bilgilerini güncelleyebilir veya silebilir. Admin, kitapları emanet olarak verebilir ve geri alabilir.

Kullanıcılar, hesaplarına giriş yaparak üzerindeki kitapları görüntüleyebilir ya da profil bilgilerini güncelleyebilir. Ancak, emanet aldığı kitabın iade süresi dolduysa kullanıcı cezalı sayılır ve sisteme giriş yapamaz. Emanet süresi uzatılmadığı sürece hesaplarına erişim sağlayamazlar.

Projemin genel olarak amacı, kitap emanet verme ve emanet alma işlemlerini daha kolay ve daha hızlı bir şekilde yapılmasını sağlamak.

İÇİNDEKİLER

1. GİRİŞ	4
2. DETAYLI GELİŞME	4
2.1 Planlama	4
2.2 Çözümleme / Analiz	5
2.3 Tasarım	5
2.4 Gerçekleştirim / Kodlama	6
2.5 Bakım / Destek	6
3. SONUÇ	6
4. KAYNAKÇA	7
5. UYGULAMAYA AİT EKРАН GÖRÜNTÜLERİ	8
6. UYGULAMAYA AİT KODLAR	14
6.1 Veri Tabanı Tablolarının Oluşturulması	14
6.2 T-SQL Kodlarının Oluşturulması	15
6.3 C# Kodlarının Oluşturulması	17

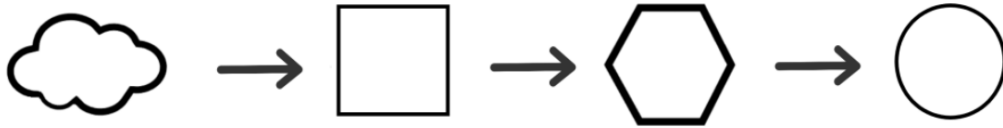
1. GİRİŞ

Bu proje, C# programlama dili ve MSSQL veri tabanı kullanılarak geliştirilmiştir. Projede çağlayan model kullanılmıştır. Projenin konusu, kütüphanelerdeki kitap ve üye yönetimi işlemlerini daha hızlı, güvenilir ve verimli hale getirmektir. Projenin amacı kütüphane personelinin iş yükünü azaltmak, işlemlerde zaman tasarrufu sağlamak ve veri yönetimini kolaylaştırmaktır. Kullanıcılar, sisteme kayıt olup giriş yaparak profillerini güncelleyebilir, üzerlerindeki emanet kitapları görüntüleyebilir. Adminler ise kitap ekleme, güncelleme işlemleri yapabilir ve kullanıcı - admin ekleme, güncelleme ve silme gibi işlemleri gerçekleştirebilir.

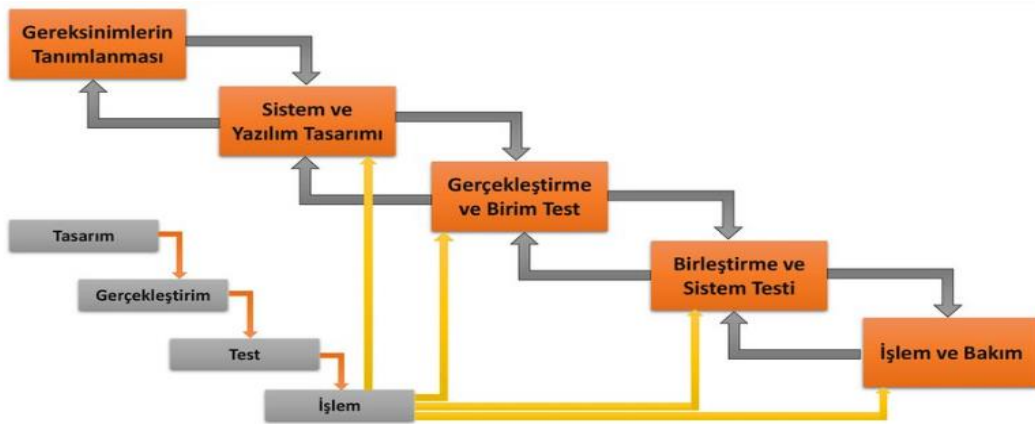
2. DETAYLI GELİŞME

2.1 Planlama

Projenin ilk aşamasında, kütüphane otomasyon sisteminin gereksinimleri belirlendi. Projede çağlayan model kullanılmasına karar verildi. İnternet üzerinden benzer projeleri inceleyerek temel olarak projeye neler eklenmesi gerektiği tespit edildi. Bu aşamada, kullanıcıların ve adminlerin hangi işlemleri yapması gerektiği detaylandırıldı ve kağıt prototip üzerinde belirlendi. Kullanıcılar için kayıt olma, giriş yapma, profil güncelleme ve emanet kitapları görüntüleme işlemleri eklenmesi gerektiğine karar verilirken, adminler için kitap ekleme, güncelleme, kullanıcı - admin ekleme, güncelleme ve silme, emanet verme ve alma işlemleri eklenmesine karar verildi.

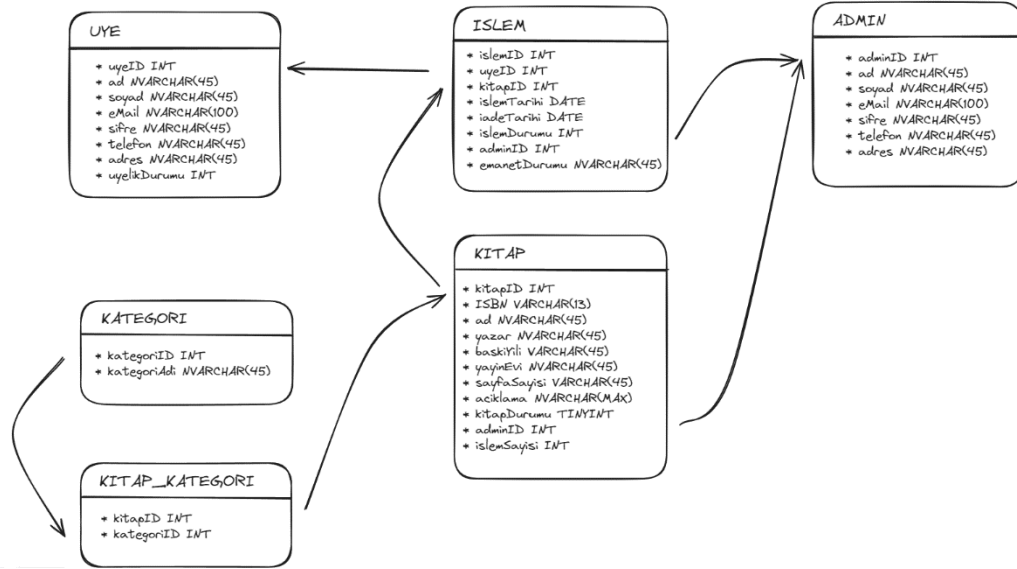


Çağlayan Modeli



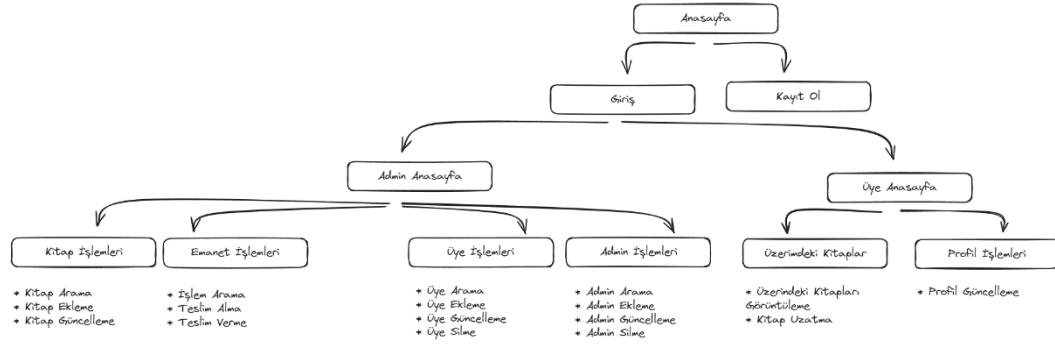
2.2 Çözümleme / Analiz

Projede neler olması gerektiği belirlendikten sonra veri tabanı tasarımı yapıldı. Kütüphane otomasyonu için gerekli olan kitaplar, kullanıcılar, adminler ve emanet işlemleri gibi tablolar oluşturuldu. Her tablonun alanları ve veri tipleri belirlendi. Veri tabanı şemaları çizilerek tablolar arasındaki ilişkiler ve veri tabanı tablolarında neler olması gerektiği belirlendi. Örneğin, kitap tablosunda kitap adı, yazar, basılı yılı, yayınevi, sayfa sayısı gibi bilgiler yer alırken, üye tablosunda ad, soyad, e-mail, şifre, telefon, adres gibi bilgiler bulunmaktaydı. Ayrıca, işlem tablosunda kitap ve üye idleri ile işlem ve iade tarihleri gibi bilgiler yer aldı. Bu analizler, veri tabanının düzgün ve verimli bir şekilde çalışmasını sağladı.



2.3 Tasarım

Sistem için gerekli formların ve kullanıcı arayüzlerinin tasarımı yapıldı. Hangi formda nelerin olması gerektiği kağıt prototip aracılığı ile belirlendi. Her formun içerdiği bilgiler ve kullanılacak kontroller belirlenerek taslaklar oluşturuldu. Ardından, prototipe bağlı kalarak Windows Form üzerinden formların tasarımları yapıldı.



2.4 Gerçekleştirim / Kodlama

Form tasarımlarının tamamlanmasının ardından, MSSQL veri tabanı ile form dosyaları arasında bağlantı kuruldu. Formların çalışır hale gelmesi için gerekli kodlar yazılarak, kullanıcıların ve adminlerin belirlenen işlemleri yapabilmesi sağlandı. Örneğin, kullanıcıların kayıt olabilmesi için gerekli kodlar yazıldı ve bu kodlar veri tabanına yeni kullanıcı bilgilerini ekledi. Aynı şekilde, adminlerin yeni kitap eklemesi için gerekli kodlar yazılarak, kitap bilgilerinin veri tabanına eklenmesi sağlandı. Ayrıca, kullanıcıların ve adminlerin giriş yapabilmesi için kimlik doğrulama işlemleri kodlandı.

2.5 Bakım / Destek

Projenin geliştirilme süreci tamamlandıktan sonra hatasız çalışmasını sağlamak için testler yapıldı. Kullanıcı ve admin olarak yapılan testler sonucunda tespit edilen hatalar giderildi. Ayrıca, sistemin ileride karşılaşılabileceği olası sorunları çözmek ve yeni özellikler eklemek için bakım ve destek planları hazırlandı. Bu planlar, kullanıcı geri bildirimlerini dikkate alarak sistemin sürekli güncellenmesini ve geliştirilmesini içermektedir.

3. SONUÇ

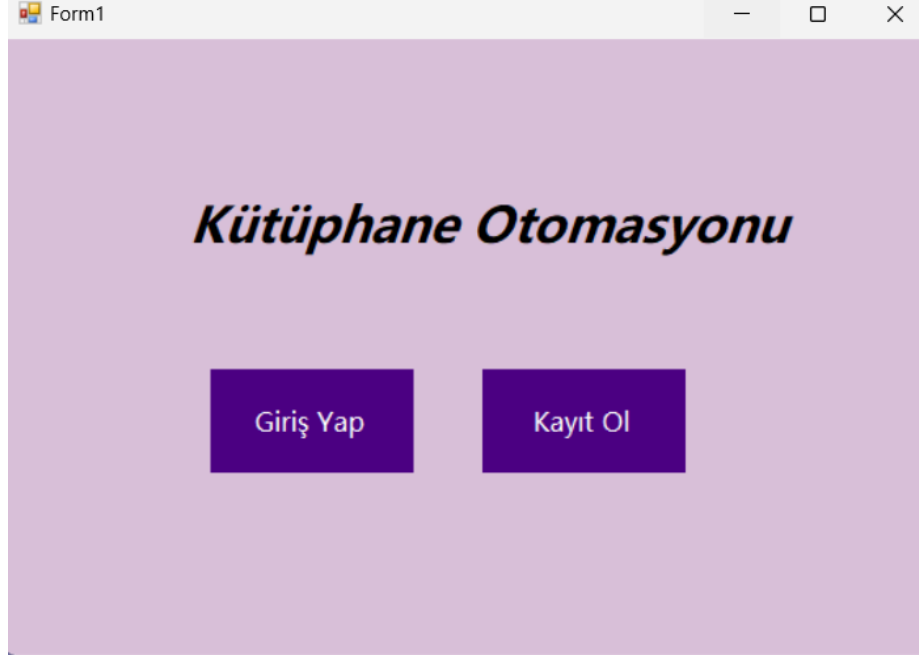
Kütüphane otomasyon sistemi, planlamalara bağlı olarak geliştirildi. Kullanıcılar ve yöneticiler için daha kullanılabilir bir ortam oluşturuldu. Süreç, planlama, çözümleme/analiz, tasarım, gerçekleştirim/kodlama ve bakım ve destek aşamalarına uygun olarak ilerletildi. Proje tamamlandıktan sonra gerekli testler yapıldı ve projenin düzgün olarak çalışıp çalışmadığı kontrol edildi.

4. KAYNAKÇA

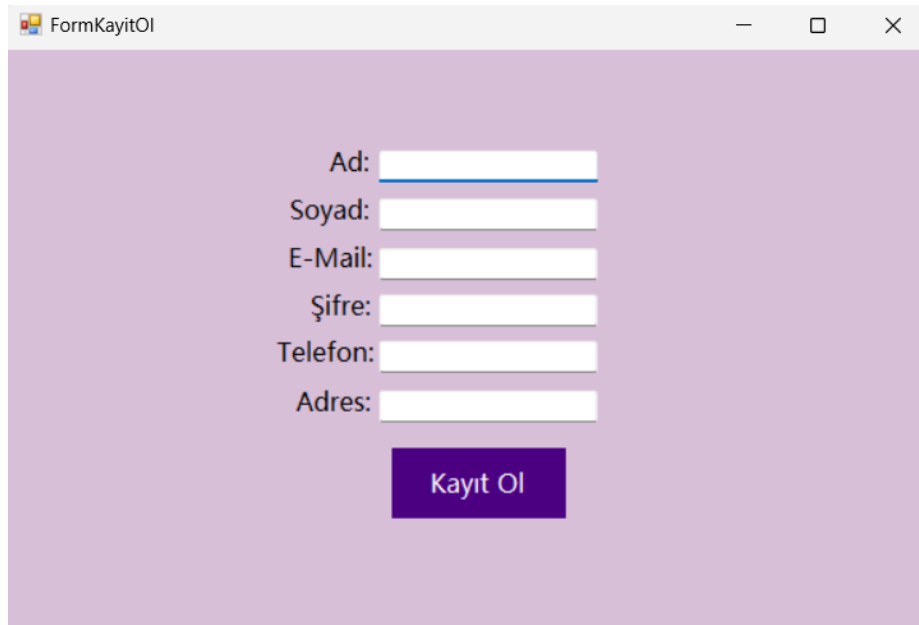
- <https://i0.wp.com/uzaykodu.com/wp-content/uploads/2021/10/evrimsel-model2.png?resize=1024%2C475&ssl=1>
- <https://slideplayer.biz.tr/slide/12386328/>
- https://excalidraw.com/#json=LwN9tgTDDkKKRXpdBZnqm,2vO79zXr6InJn_MVR_YlsPQ

5. UYGULAMAYA AİT EKRAN GÖRÜNTÜLERİ

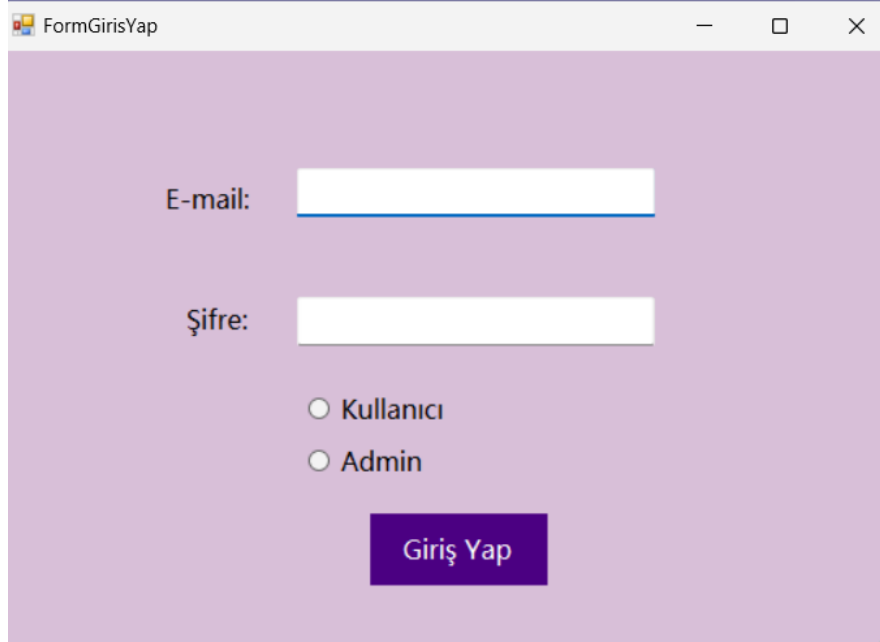
- **FormAnasayfa.cs:** Sistem ilk açıldığında görüntülenen formdur. Formda giriş yap ve kayıt ol butonları bulunmaktadır. Giriş yap butonuna tıklandığında FormGirisYap.cs formuna yönlendirir. Kayıt ol butonuna tıklandığında ise FormKayitOl.cs formuna yönlendirir.



- **FormKayitOl.cs:** Bu formda kullanıcı kendi bilgilerini girip kayıt ol butonuna bastığında kayıt olma işlemi gerçekleştirebilir. Herhangi bir hata durumunda gerekli hata mesajları gösterilir.

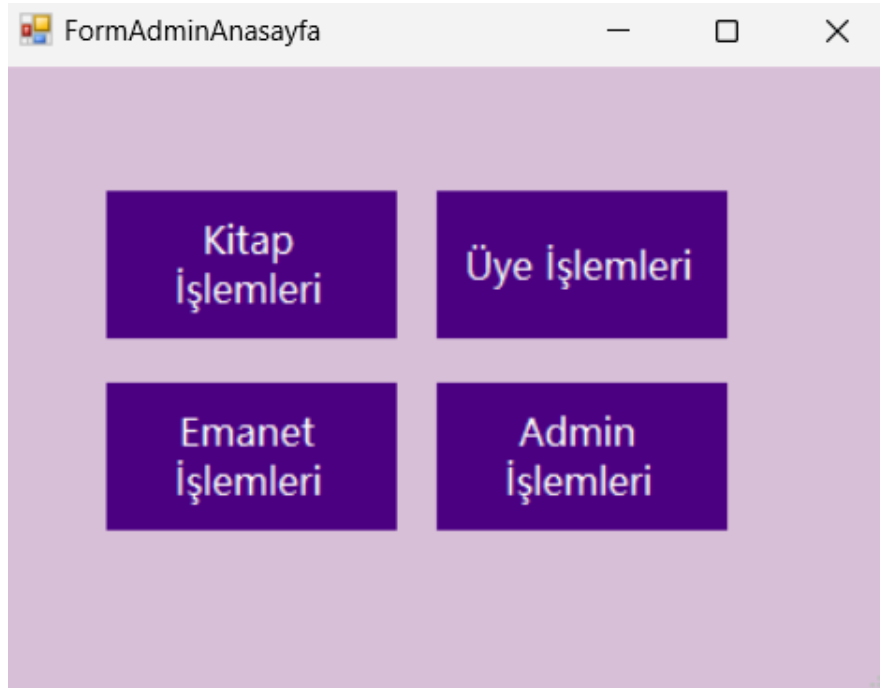


- **FormGirisYap.cs:** Kullanıcıların ve adminlerin seçimlerine göre gerekli tablolardan kontrollerini yaparak giriş bilgilerinin eşleşmesini kontrol eder. Hatalı giriş durumunda gerekli hata mesajları gösterilir. Admin girişinde FormAdminAnasayfa.cs formuna yönlendirilir. Kullanıcı girişinde ise FormUyeAnasayfa.cs formuna yönlendirilir.



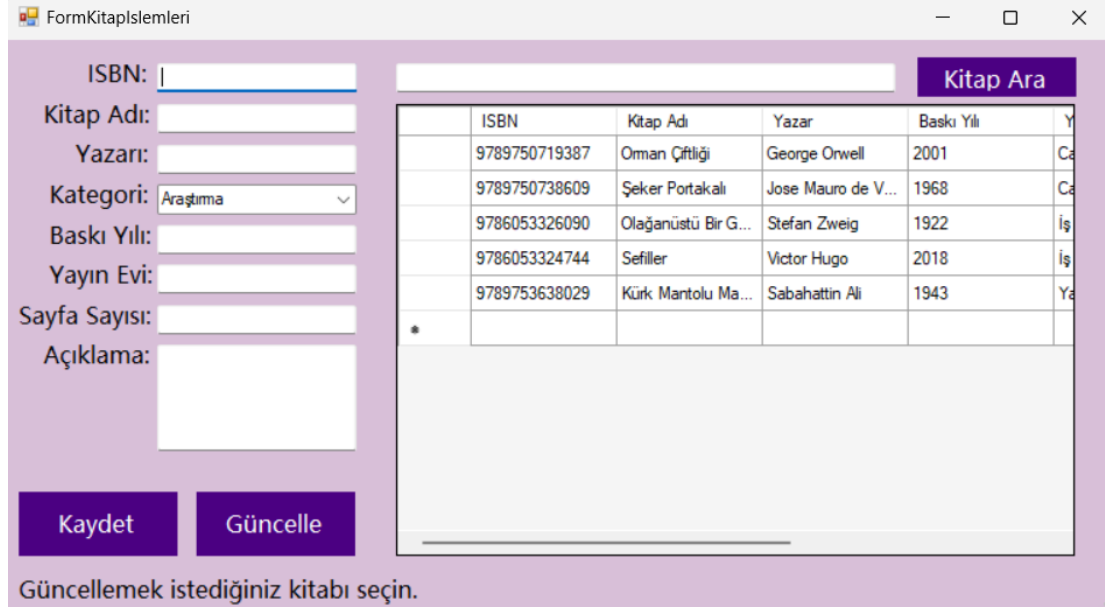
The screenshot shows a web application window titled "FormGirisYap". The background is a light purple color. It features two white input fields for "E-mail:" and "Şifre:". Below the password field, there are two radio buttons labeled "Kullanıcı" and "Admin". At the bottom center, there is a dark purple button with the text "Giriş Yap" in white.

- **FormAdminAnasayfa.cs:** Sisteme admin bilgileriyle giriş yapıldığında karşımıza çıkan ilk ekrandır. Menüdeki kitap işlemleri, üye işlemleri, emanet işlemleri, admin işlemleri butonlarına tıklandığında ilgili formlar açılır.



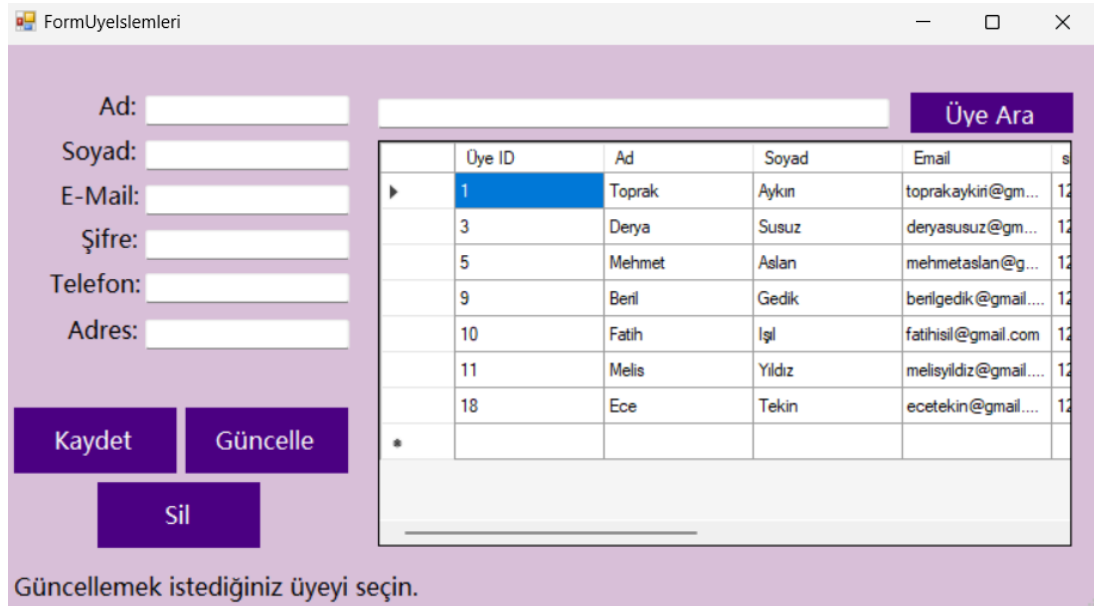
The screenshot shows a web application window titled "FormAdminAnasayfa". The background is a light purple color. It displays four dark purple buttons arranged in a 2x2 grid. The buttons are labeled "Kitap İşlemleri", "Üye İşlemleri", "Emanet İşlemleri", and "Admin İşlemleri" in white text.

- **FormKitapIslemleri.cs:** FormAdminAnasayfa.cs formunda kitap işlemleri butonuna basıldığında açılan formdur. Bu formda Kitap kaydetme ve güncelleme işlemleri yapılabilir. Aynı zamanda kitap ara butonu ile filtreleme işlemi yapılabilir.



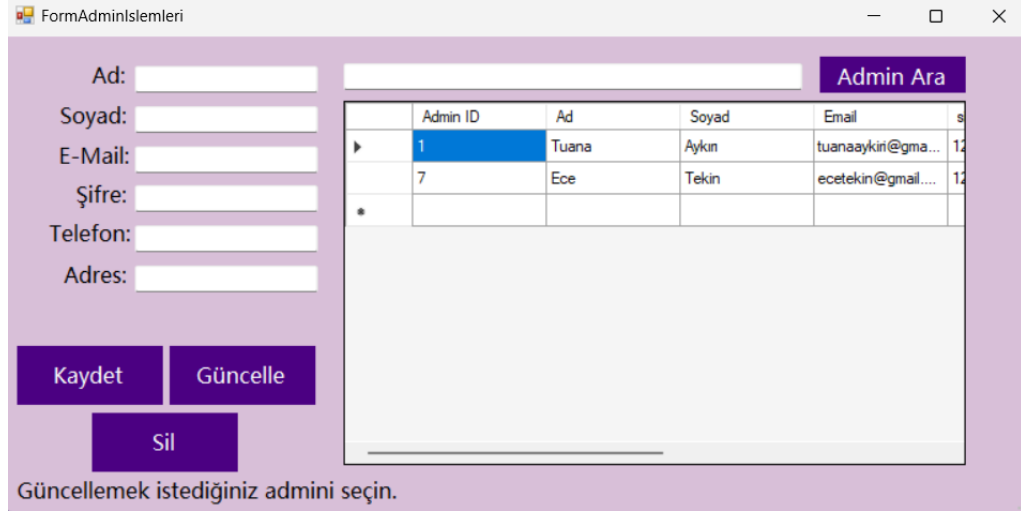
ISBN	Kitap Adı	Yazar	Baskı Yılı	Y
9789750719387	Oman Çiftliği	George Orwell	2001	Ce
9789750738609	Şeker Portakalı	Jose Mauro de V...	1968	Ce
9786053326090	Olağanüstü Bir G...	Stefan Zweig	1922	İş
9786053324744	Sefiller	Victor Hugo	2018	İş
9789753638029	Kürk Mantolu Ma...	Sabahattin Ali	1943	Ye

- **FormUyeIslemleri.cs:** FormAdminAnasayfa.cs formunda üye işlemleri butonuna tıklandığında açılan formdur. Bu formda üye ekleme güncelleme ve silme işlemleri yapılabilir. Aynı zamanda üye ara butonu ile filtreleme işlemi yapılabilir.



Üye ID	Ad	Soyad	Email	s
1	Toprak	Aykın	toprakaykiri@gm...	12
3	Derya	Susuz	deryasusuz@gm...	12
5	Mehmet	Aslan	mehmetaslan@g...	12
9	Benli	Gedik	benligedik@gmail....	12
10	Fatih	İşıl	fatihisil@gmail.com	12
11	Melis	Yıldız	melisyildiz@gmail....	12
18	Ece	Tekin	ecetekin@gmail....	12

- **FormAdminIslemleri.cs:** FormAdminAnasayfa.cs formunda admin işlemleri butonuna tıklandığında açılan formdur. Bu formda admin ekleme güncelleme ve silme işlemleri yapılabilir. Aynı zamanda üye ara butonu ile filtreleme işlemi yapılabilir.



FormAdminIslemleri

Ad:

Soyad:

E-Mail:

Şifre:

Telefon:

Adres:

Kaydet Güncelle

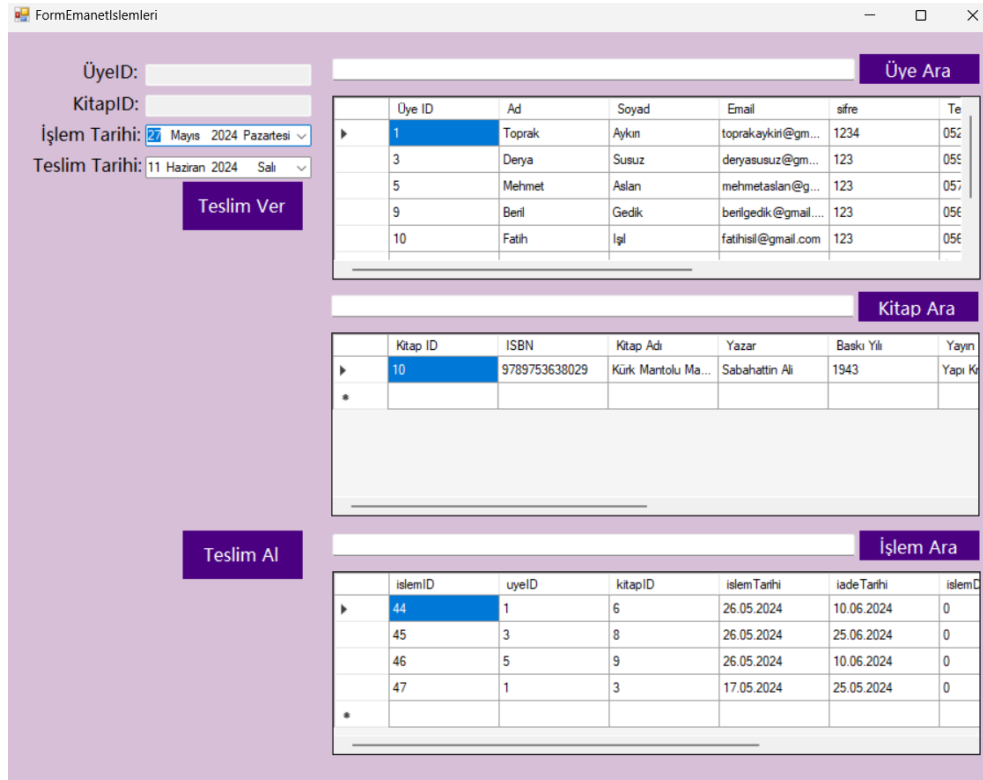
Sil

Admin Ara

Admin ID	Ad	Soyad	Email
1	Tuana	Aykın	tuanaaykin@gma...
7	Ece	Tekin	ecetekin@gmail....

Güncellemek istediğiniz admini seçin.

- **FormEmanetIslemleri.cs:** FormAdminAnasayfa.cs formunda admin işlemleri butonuna tıklandığında açılan formdur. Bu formda üye ara butonu ile istenilen üye listelenebilir. Kitap ara butonu ile istenilen kitap listelenebilir. Üye tablosundan seçilen üyeye kitap tablosundan seçilen kitap teslim verilebilir veya işlem tablosundan seçilen emanetteki kitap geri teslim alınabilir.



FormEmanetIslemleri

ÜyeID:

KitapID:

İşlem Tarihi: 26 Mayıs 2024 Pazartesi

Teslim Tarihi: 11 Haziran 2024 Salı

Teslim Ver

Üye Ara

Üye ID	Ad	Soyad	Email	Şifre	Te
1	Toprak	Aykın	toprakaykin@gm...	1234	052
3	Derya	Susuz	deryasusuz@gm...	123	055
5	Mehmet	Aslan	mehmetaslan@g...	123	057
9	Beril	Gedik	berilgedik@gmail...	123	056
10	Fatih	İşil	fatihisil@gmail.com	123	056

Kitap Ara

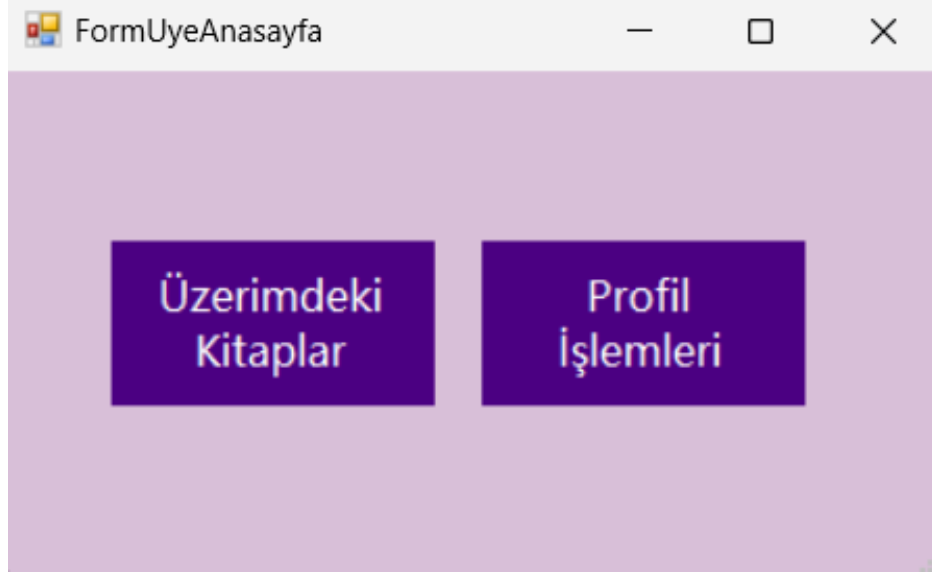
Kitap ID	ISBN	Kitap Adı	Yazar	Baskı Yılı	Yayın
10	9789753638029	Kürk Mantolu Ma...	Sabahattin Ali	1943	Yayı

İşlem Ara

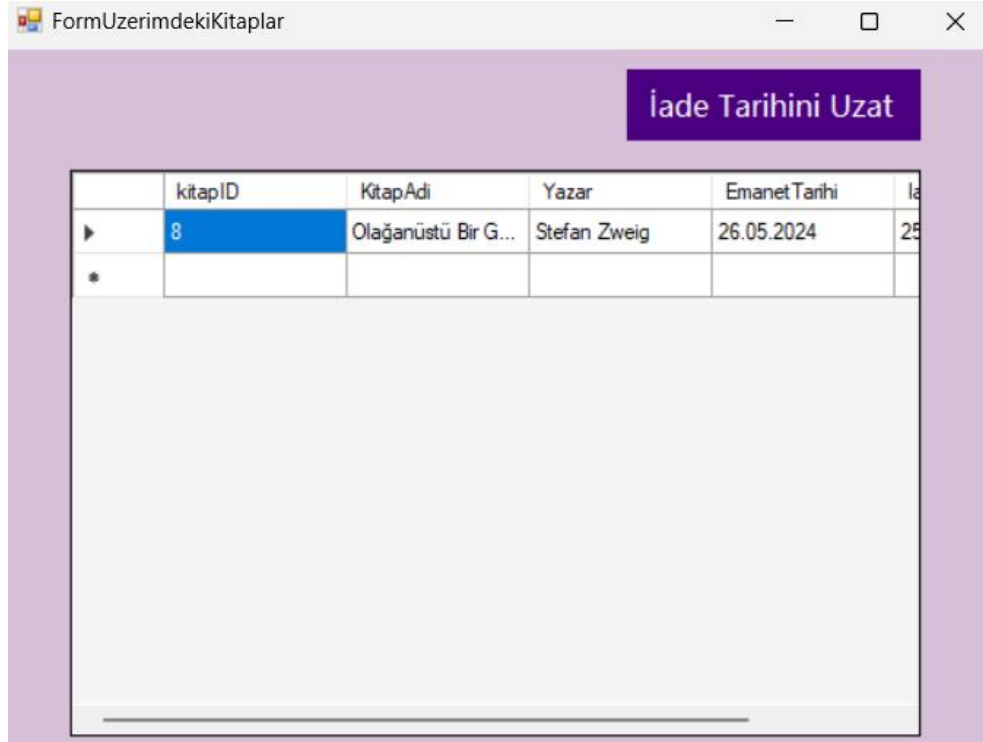
İşlemID	uyeID	kitapID	İşlem Tarihi	İade Tarihi	İşlemD
44	1	6	26.05.2024	10.06.2024	0
45	3	8	26.05.2024	25.06.2024	0
46	5	9	26.05.2024	10.06.2024	0
47	1	3	17.05.2024	25.05.2024	0

Teslim Al

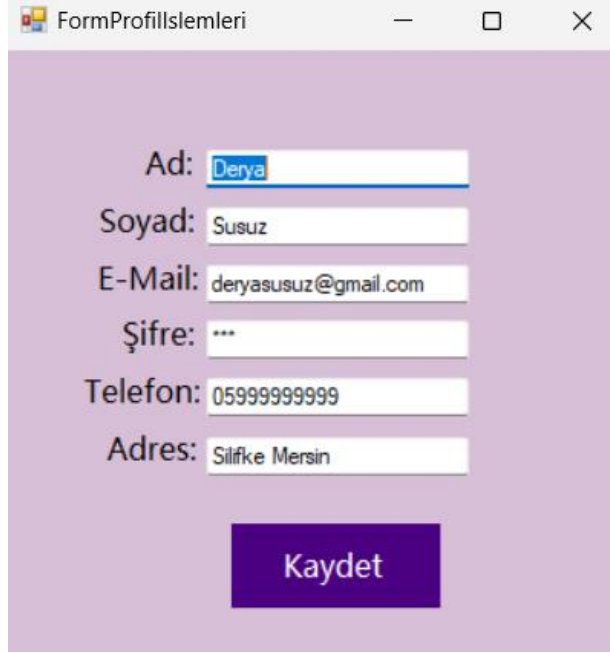
- **FormUyeAnasayfa.cs:** Sisteme admin bilgileriyle giriş yapıldığında karşımıza çıkan ilk ekrandır. Ekrandaki üzerimdeki kitaplar ve profil işlemleri butonlarına tıklandığında ilgili formlar açılır.



- **FormUzerimdekiKitaplar.cs:** FormUyeAnasayfa.cs formunda üzerimdeki kitaplar butonuna basıldığında açılan formdur. Bu formda kullanıcı üzerindeki kitapları görüntüleyebilir kitabı aldığı ve geri vermesi gerektiği tarihi görebilir. İade tarihini uzat butonu ile seçili olan kitabın iade tarihini 15 gün uzatabilir.



- **FormProfilIslemleri.cs:** FormUyeAnasayfa.cs formunda profil işlemleri butonuna basıldığında açılan formdur. Bu formda kullanıcı kayıtlı bilgilerini görebilir ve güncelleyebilir.



The screenshot shows a web form titled "FormProfilIslemleri" with a light purple background. The form contains several input fields for user profile information, each with a label and a text box. The fields are: "Ad:" with the value "Derya", "Soyad:" with the value "Susuz", "E-Mail:" with the value "deryasusuz@gmail.com", "Şifre:" with the value "...", "Telefon:" with the value "05999999999", and "Adres:" with the value "Silfke Mersin". Below the input fields is a large purple button labeled "Kaydet".

Field	Value
Ad:	Derya
Soyad:	Susuz
E-Mail:	deryasusuz@gmail.com
Şifre:	...
Telefon:	05999999999
Adres:	Silfke Mersin

Kaydet

6. UYGULAMAYA AİT KODLAR

6.1 Veri Tabanı Tablolarının Oluşturulması

- Admin tablosunun oluşturulması:

```
CREATE TABLE admin (  
    adminID INT PRIMARY KEY IDENTITY(1,1),  
    ad NVARCHAR(45) NOT NULL,  
    soyad NVARCHAR(45) NOT NULL,  
    eMail NVARCHAR(100) NOT NULL UNIQUE,  
    sifre NVARCHAR(45) NOT NULL DEFAULT '123',  
    telefon NVARCHAR(45),  
    adres NVARCHAR(45),  
    normalizedEmail NVARCHAR(255)  
);
```

- Üye tablosunun oluşturulması:

```
CREATE TABLE uye (  
    uyeID INT PRIMARY KEY IDENTITY(1,1),  
    ad NVARCHAR(45) NOT NULL,  
    soyad NVARCHAR(45) NOT NULL,  
    eMail NVARCHAR(100) NOT NULL UNIQUE,  
    sifre NVARCHAR(45) NOT NULL DEFAULT '123',  
    telefon NVARCHAR(45),  
    adres NVARCHAR(45),  
    uyelikDurumu INT DEFAULT 1,  
    normalizedEmail NVARCHAR(255)  
);
```

- Kitap tablosunun oluşturulması:

```
CREATE TABLE kitap (  
    kitapID INT PRIMARY KEY IDENTITY(1,1),  
    ISBN VARCHAR(13) NOT NULL UNIQUE,  
    ad NVARCHAR(45) NOT NULL,  
    yazar NVARCHAR(45) NOT NULL,  
    baskiYili VARCHAR(45),  
    yayinEvi NVARCHAR(45),  
    sayfaSayisi VARCHAR(45),  
    aciklama NVARCHAR(MAX),  
    kitapDurumu TINYINT NOT NULL DEFAULT 1,  
    adminID INT,  
    islemSayisi INT DEFAULT 0,  
    CONSTRAINT fk_kitap_adminID FOREIGN KEY (adminID) REFERENCES admin (adminID)  
);
```

- Kategori tablosunun oluşturulması:

```
CREATE TABLE kategori (  
    kategoriID INT PRIMARY KEY IDENTITY(1,1),  
    kategoriAdi NVARCHAR(45) NOT NULL UNIQUE  
);
```

- **Kitap kategori tablosunun oluşturulması:**

```
CREATE TABLE kitap_kategori (  
    kitapID INT NOT NULL,  
    kategoriID INT NOT NULL,  
    PRIMARY KEY (kitapID, kategoriID),  
    CONSTRAINT FK_kategori FOREIGN KEY (kategoriID) REFERENCES kategori (kategoriID),  
    CONSTRAINT FK_kitap FOREIGN KEY (kitapID) REFERENCES kitap (kitapID)  
);
```

- **İşlem tablosunun oluşturulması:**

```
CREATE TABLE islem (  
    islemID INT NOT NULL IDENTITY(1,1),  
    uyeID INT NOT NULL,  
    kitapID INT NOT NULL,  
    islemTarihi DATE,  
    iadeTarihi DATE,  
    islemDurumu INT NOT NULL DEFAULT 0,  
    adminID INT,  
    emanetDurumu NVARCHAR(45) DEFAULT 'RAFTA',  
    PRIMARY KEY (islemID, uyeID, kitapID),  
    CONSTRAINT fk_kitapID FOREIGN KEY (kitapID) REFERENCES kitap (kitapID),  
    CONSTRAINT fk_uyeID FOREIGN KEY (uyeID) REFERENCES uye (uyeID),  
    CONSTRAINT fk_admin FOREIGN KEY (adminID) REFERENCES admin (adminID)  
);
```

6.2 T-SQL Kodlarının Oluşturulması

- **Giriş kontrol prosedürü:**

```
CREATE PROCEDURE sp_GirisKontrol  
    @Email NVARCHAR(50),  
    @Sifre NVARCHAR(50)  
AS  
BEGIN  
    SELECT 'Admin' AS Rol  
    FROM admin  
    WHERE normalizedEmail = dbo.NormalizeEmail(@Email) AND sifre = @Sifre  
  
    UNION  
  
    SELECT 'Kullanici' AS Rol  
    FROM uye  
    WHERE normalizedEmail = dbo.NormalizeEmail(@Email) AND sifre = @Sifre  
END;
```

- **Teslim verme prosedürü:**

```
CREATE PROCEDURE TeslimVer  
    @uyeID INT,  
    @kitapID INT,  
    @islemTarihi DATE,  
    @iadeTarihi DATE,  
    @adminID INT  
AS  
BEGIN  
    -- İşlem tablosuna yeni kayıt ekle  
    INSERT INTO islem (uyeID, kitapID, islemTarihi, iadeTarihi, adminID, emanetDurumu)  
    VALUES (@uyeID, @kitapID, @islemTarihi, @iadeTarihi, @adminID, 'EMANETTE');  
  
    -- Kitap tablosundaki kitapDurumu'nu güncelle
```

```
UPDATE kitap SET kitapDurumu = 0 WHERE kitapID = @kitapID;
END;
```

- **Teslim alma prosedürü:**

```
CREATE PROCEDURE TeslimAl
    @islemID INT,
    @kitapID INT
AS
BEGIN
    -- İşlem tablosundan ilgili kaydı sil
    DELETE FROM islem WHERE islemID = @islemID;

    -- Kitap tablosundaki kitapDurumu'nu güncelle
    UPDATE kitap SET kitapDurumu = 1 WHERE kitapID = @kitapID;
END;
```

- **Email normalizasyon fonksiyonu:**

```
CREATE FUNCTION dbo.NormalizeEmail(@email NVARCHAR(255))
RETURNS NVARCHAR(255)
AS
BEGIN
    DECLARE @normalizedEmail NVARCHAR(255);

    -- Küçük harflere dönüştürme
    SET @normalizedEmail = LOWER(@email);

    -- Gereksiz boşlukları kaldırma
    SET @normalizedEmail = LTRIM(RTRIM(@normalizedEmail));

    RETURN @normalizedEmail;
END;
```

- **Email normalizasyon triggerı:**

- Admin tablosu için:

```
CREATE TRIGGER trg_Admin_Insert
ON admin
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE admin
    SET normalizedEmail = dbo.NormalizeEmail(i.eMail)
    FROM inserted i
    WHERE admin.adminID = i.adminID;
END;
```

- Üye tablosu için:

```
CREATE TRIGGER trg_Uye_Insert
ON uye
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE uye
    SET normalizedEmail = dbo.NormalizeEmail(i.eMail)
```



```
FROM inserted i
WHERE uye.uyeID = i.uyeID;
END;
```

6.3 C# Kodlarının Oluşturulması

- **Sql ile bağlantı kurma:**

```
private string _connectionString;
private static ClassSql _instance;

public ClassSql(string connectionString)
{
    _connectionString = connectionString;
}

public static ClassSql GetInstance()
{
    if (_instance == null)
    {
        _instance = new ClassSql("Data Source=LAPTOP-0MDR6GEA\\SQLEXPRESS;Initial
Catalog=KutuphaneOtomasyonu;Integrated Security=True;");
    }
    return _instance;
}
```

- **Tutulacak verilerin tanımlanması:**

```
public static class ClassVeriler
{
    public static int AdminID { get; set; }
    public static int UyeID { get; set; }
}
```

- **Anasayfa formu**

- **FormAnasayfa.cs:**

```
public partial class FormAnasayfa : Form
{
    public FormAnasayfa()
    {
        InitializeComponent();
    }

    private void girisGitBtn_Click(object sender, EventArgs e)
    {
        FormGirisYap giris = new FormGirisYap();
        giris.ShowDialog();
    }

    private void kayitOlBtn_Click(object sender, EventArgs e)
    {
        FormKayitOl kayit = new FormKayitOl();
        kayit.ShowDialog();
    }
}
```

- **Kayıt ol formu**

- **ClassSql:**

```
public bool KayitOl(string tableName, string ad, string soyad, string email, string
sifre, string telefon, string adres)
{
    if (TelefonVarMi(telefon))
    {
        MessageBox.Show("Bu telefon numarası zaten kayıtlı.");
        return false;
    }

    if (EmailVarMi(email))
    {
        MessageBox.Show("Bu e-posta adresi zaten kayıtlı.");
        return false;
    }

    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = $"INSERT INTO {tableName} (ad, soyad, eMail, sifre, telefon,
adres) VALUES (@Ad, @Soyad, @Email, @Sifre, @Telefon, @Adres)";
        using (var command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Ad", ad);
            command.Parameters.AddWithValue("@Soyad", soyad);
            command.Parameters.AddWithValue("@Email", email);
            command.Parameters.AddWithValue("@Sifre", sifre);
            command.Parameters.AddWithValue("@Telefon", telefon);
            command.Parameters.AddWithValue("@Adres", adres);

            int rowsAffected = command.ExecuteNonQuery();
            return rowsAffected > 0;
        }
    }
}

public bool BosKontrolu(params TextBox[] textBoxes)
{
    foreach (var textBox in textBoxes)
    {
        if (string.IsNullOrWhiteSpace(textBox.Text))
        {
            MessageBox.Show($"{textBox.Name} boş bırakılamaz.", "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return false;
        }
    }
    return true;
}

public bool TelefonVarMi(string telefon)
{
    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "SELECT COUNT(*) FROM uye WHERE telefon = @Telefon";
        using (var command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Telefon", telefon);
            int count = (int)command.ExecuteScalar();
            return count > 0;
        }
    }
}
```

```

public bool EmailVarMi(string email)
{
    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "SELECT COUNT(*) FROM uye WHERE eMail = @Email";
        using (var command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Email", email);
            int count = (int)command.ExecuteScalar();
            return count > 0;
        }
    }
}

```

- **FormKayitOl.cs:**

```

public partial class FormKayitOl : Form
{
    private ClassSql classSql;
    public FormKayitOl()
    {
        InitializeComponent();
        classSql = ClassSql.GetInstance();
    }

    private void kayitOlBtn_Click(object sender, EventArgs e)
    {
        string ad = adTxt.Text;
        string soyad = soyadTxt.Text;
        string email = emailTxt.Text;
        string sifre = sifreTxt.Text;
        string telefon = telefonTxt.Text;
        string adres = adresTxt.Text;

        ClassSql sqlInstance = ClassSql.GetInstance();

        if (!sqlInstance.BosKontrolu(adTxt, soyadTxt, emailTxt, sifreTxt,
telefonTxt, adresTxt))
        {
            return;
        }

        bool kayitBasarili = sqlInstance.KayitOl("uye", ad, soyad, email, sifre,
telefon, adres);

        if (kayitBasarili)
        {
            MessageBox.Show("Kayıt işlemi başarıyla gerçekleştirildi.", "Başarılı",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            this.Close();
        }
        else
        {
            MessageBox.Show("Kayıt işlemi sırasında bir hata oluştu. Lütfen tekrar
deneyin.", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

```

- Giriş Yap Formu

- ClassSql:

```
public string GirisKontrol(string email, string sifre)
{
    string rol = string.Empty;
    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        using (var command = new SqlCommand("sp_GirisKontrol", connection))
        {
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.AddWithValue("@Email", email);
            command.Parameters.AddWithValue("@Sifre", sifre);

            using (SqlDataReader reader = command.ExecuteReader())
            {
                if (reader.Read())
                {
                    rol = reader["Rol"].ToString();
                }
                else
                {
                    throw new Exception("Kullanıcı veya şifre hatalı.");
                }
            }
        }
    }
    return rol;
}

public int GetAdminIDByEmail(string email)
{
    int adminID = 0;
    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "SELECT adminID FROM admin WHERE Email = @Email";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Email", email);

            using (SqlDataReader reader = command.ExecuteReader())
            {
                if (reader.Read())
                {
                    adminID = reader.GetInt32(0);
                }
            }
        }
    }
    return adminID;
}

public int GetUyeIDByEmail(string email)
{
    int uyeID = 0;
    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "SELECT uyeID FROM uye WHERE Email = @Email";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Email", email);
```

```

        using (SqlDataReader reader = command.ExecuteReader())
        {
            if (reader.Read())
            {
                uyeID = reader.GetInt32(0);
            }
        }
    }
    return uyeID;
}

```

- **FormGirisYap.cs:**

```

public partial class FormGirisYap : Form
{
    private ClassSql classSql;
    public FormGirisYap()
    {
        InitializeComponent();
        classSql = ClassSql.GetInstance();
    }

    private void girisBtn_Click(object sender, EventArgs e)
    {
        try
        {
            string email = emailTxt.Text;
            string sifre = sifreTxt.Text;
            bool isAdminSelected = radioAdmin.Checked;
            bool isUserSelected = radioKullanici.Checked;

            ClassSql classSql = ClassSql.GetInstance();
            string rol = classSql.GirisKontrol(email, sifre);

            if (isAdminSelected && rol != "Admin")
            {
                MessageBox.Show("Yanlış seçim yaptınız. Lütfen tekrar deneyiniz.");
                return;
            }

            if (isUserSelected && rol != "Kullanici")
            {
                MessageBox.Show("Yanlış seçim yaptınız. Lütfen tekrar deneyiniz.");
                return;
            }
            if (rol == "Admin")
            {
                int adminID = classSql.GetAdminIDByEmail(email);
                ClassVeriler.AdminID = adminID;

                FormAdminAnasayfa formAdminAnasayfa = new FormAdminAnasayfa();
                formAdminAnasayfa.ShowDialog();
                this.Hide();
            }
            else if (rol == "Kullanici")
            {
                int uyeID = classSql.GetUyeIDByEmail(email);
                ClassVeriler.UyeID = uyeID;

                if (classSql.UyeIadeEdilmemisKitabiVarMi(uyeID))
                {
                    MessageBox.Show("İade etmeniz gereken bir kitabınız  
bulunmaktadır. Kitap iadesi yapılana kadar hesabınıza giriş yapamazsınız.");
                    return;
                }
            }
            else

```

```

        {
            FormUyeAnasayfa formUyeAnasayfa = new FormUyeAnasayfa();
            formUyeAnasayfa.ShowDialog();
            this.Hide();
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}
}

```

- **Admin Anasayfa Formu**

- **FormAdminAnasayfa.cs:**

```

public partial class FormAdminAnasayfa : Form
{
    public FormAdminAnasayfa()
    {
        InitializeComponent();
    }

    private void kitapIslemBtn_Click(object sender, EventArgs e)
    {
        FormKitapIslemleri kitapIslemleri = new FormKitapIslemleri();
        kitapIslemleri.ShowDialog();
    }

    private void uyeIslemBtn_Click(object sender, EventArgs e)
    {
        FormUyeIslemleri uyeIslemleri = new FormUyeIslemleri();
        uyeIslemleri.ShowDialog();
    }

    private void emanetIslemBtn_Click(object sender, EventArgs e)
    {
        FormEmanetIslemleri emanetIslemleri = new FormEmanetIslemleri();
        emanetIslemleri.ShowDialog();
    }

    private void adminIslemBtn_Click(object sender, EventArgs e)
    {
        FormAdminIslemleri adminIslemleri = new FormAdminIslemleri();
        adminIslemleri.ShowDialog();
    }
}

```

- **Kitap İşlemleri Formu**

- **ClassSql:**

```

public bool BosKontrolu(params TextBox[] textBoxes)
{
    foreach (var textBox in textBoxes)
    {
        if (string.IsNullOrWhiteSpace(textBox.Text))
        {

```

```

        MessageBox.Show($"{textBox.Name} boş bırakılamaz.", "Hata",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
    return true;
}

public DataTable TumKitaplarıGetir()
{
    DataTable dataTable = new DataTable();

    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        string query = @"
        SELECT k.kitapID, k.ISBN, k.ad, k.yazar, k.baskiYili, k.yayinEvi, k.sayfaSayisi,
        k.aciklama, kk.kategoriID
        FROM kitap k
        LEFT JOIN kitap_kategori kk ON k.kitapID = kk.kitapID";

        SqlCommand command = new SqlCommand(query, connection);
        SqlDataAdapter adapter = new SqlDataAdapter(command);
        adapter.Fill(dataTable);
    }

    return dataTable;
}

public DataTable TumKategorileriGetir()
{
    DataTable dataTable = new DataTable();

    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        string query = "SELECT * FROM kategori";
        SqlCommand command = new SqlCommand(query, connection);
        SqlDataAdapter adapter = new SqlDataAdapter(command);
        adapter.Fill(dataTable);
    }

    return dataTable;
}

public DataTable FiltreleKitap(string anahtarKelime)
{
    DataTable dataTable = new DataTable();
    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "SELECT * FROM kitap WHERE ISBN LIKE @AnahtarKelime OR Ad
        LIKE @AnahtarKelime OR yazar LIKE @AnahtarKelime ";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@AnahtarKelime", "%" + anahtarKelime +
            "%");

            try
            {
                SqlDataAdapter adapter = new SqlDataAdapter(command);
                adapter.Fill(dataTable);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Filtreleme sırasında bir hata oluştu:
                {ex.Message}");
            }
        }

        return dataTable;
    }
}

```

```

public bool KitapEkle(string isbn, string kitapAdi, string yazar, string baskiYili,
string yayinEvi, string sayfaSayisi, string aciklama, int kategoriID)
{
    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        SqlTransaction transaction = connection.BeginTransaction();

        try
        {
            // ISBN kontrolü
            string isbnCheckQuery = "SELECT COUNT(*) FROM kitap WHERE ISBN = @ISBN";
            using (SqlCommand isbnCheckCommand = new SqlCommand(isbnCheckQuery,
connection, transaction))
            {
                isbnCheckCommand.Parameters.AddWithValue("@ISBN", isbn);
                int count = (int)isbnCheckCommand.ExecuteScalar();
                if (count > 0)
                {
                    MessageBox.Show("Aynı ISBN numarası ile kayıt yapılamaz.");
                    return false;
                }
            }

            string kitapQuery = "INSERT INTO kitap (ISBN, ad, yazar, baskiYili,
yayinEvi, sayfaSayisi, aciklama, kitapDurumu, adminID) " +
"OUTPUT INSERTED.kitapID " +
"VALUES (@ISBN, @Ad, @Yazar, @BaskiYili, @YayinEvi,
@SayfaSayisi, @Aciklama, 1, 1)";

            int kitapID;
            using (SqlCommand kitapCommand = new SqlCommand(kitapQuery, connection,
transaction))
            {
                kitapCommand.Parameters.AddWithValue("@ISBN", isbn);
                kitapCommand.Parameters.AddWithValue("@Ad", kitapAdi);
                kitapCommand.Parameters.AddWithValue("@Yazar", yazar);
                kitapCommand.Parameters.AddWithValue("@BaskiYili", baskiYili);
                kitapCommand.Parameters.AddWithValue("@YayinEvi", yayinEvi);
                kitapCommand.Parameters.AddWithValue("@SayfaSayisi", sayfaSayisi);
                kitapCommand.Parameters.AddWithValue("@Aciklama", aciklama);

                kitapID = (int)kitapCommand.ExecuteScalar();
            }

            string kitapKategoriQuery = "INSERT INTO kitap_kategori (kitapID,
kategoriID) VALUES (@KitapID, @KategoriID)";

            using (SqlCommand kitapKategoriCommand = new
SqlCommand(kitapKategoriQuery, connection, transaction))
            {
                kitapKategoriCommand.Parameters.AddWithValue("@KitapID", kitapID);
                kitapKategoriCommand.Parameters.AddWithValue("@KategoriID",
kategoriID);

                kitapKategoriCommand.ExecuteNonQuery();
            }

            transaction.Commit();
            return true;
        }
        catch (Exception ex)
        {
            transaction.Rollback();
            MessageBox.Show($"Kitap eklenirken bir hata oluştu: {ex.Message}");
            return false;
        }
    }
}

public bool KitapGuncelle(int kitapID, string isbn, string kitapAdi, string yazar,
string baskiYili, string yayinEvi, string sayfaSayisi, string aciklama, int
kategoriID)

```



```

{
    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        SqlTransaction transaction = connection.BeginTransaction();

        try
        {
            string kitapQuery = "UPDATE kitap SET ISBN = @ISBN, ad = @Ad, yazar = @Yazar, baskiYili = @BaskiYili, yayinEvi = @YayinEvi, sayfaSayisi = @SayfaSayisi, aciklama = @Aciklama WHERE kitapID = @KitapID";

            using (SqlCommand kitapCommand = new SqlCommand(kitapQuery, connection, transaction))
            {
                kitapCommand.Parameters.AddWithValue("@KitapID", kitapID);
                kitapCommand.Parameters.AddWithValue("@ISBN", isbn);
                kitapCommand.Parameters.AddWithValue("@Ad", kitapAdi);
                kitapCommand.Parameters.AddWithValue("@Yazar", yazar);
                kitapCommand.Parameters.AddWithValue("@BaskiYili", baskiYili);
                kitapCommand.Parameters.AddWithValue("@YayinEvi", yayinEvi);
                kitapCommand.Parameters.AddWithValue("@SayfaSayisi", sayfaSayisi);
                kitapCommand.Parameters.AddWithValue("@Aciklama", aciklama);

                kitapCommand.ExecuteNonQuery();

                string kitapKategoriQuery = "UPDATE kitap_kategori SET kategoriID = @KategoriID WHERE kitapID = @KitapID";

                using (SqlCommand kitapKategoriCommand = new SqlCommand(kitapKategoriQuery, connection, transaction))
                {
                    kitapKategoriCommand.Parameters.AddWithValue("@KitapID", kitapID);
                    kitapKategoriCommand.Parameters.AddWithValue("@KategoriID", kategoriID);
                    kitapKategoriCommand.ExecuteNonQuery();
                }
                transaction.Commit();
                return true;
            }
            catch (Exception ex)
            {
                transaction.Rollback();
                MessageBox.Show($"Kitap güncellenirken bir hata oluştu: {ex.Message}");
                return false;
            }
        }
    }
}

```

- FormKitapIslemleri.cs

```

public partial class FormKitapIslemleri : Form
{
    private ClassSql classSql;
    public FormKitapIslemleri()
    {
        InitializeComponent();
        classSql = ClassSql.GetInstance();
    }

    private void FormKitapIslemleri_Load(object sender, EventArgs e)
    {
        DataTable dataTable = classSql.TumKitaplariGetir();
        dataGridView1.DataSource = dataTable;
    }
}

```

```

dataGridView1.Columns["kitapID"].Visible = false;
dataGridView1.Columns["ISBN"].HeaderText = "ISBN";
dataGridView1.Columns["ad"].HeaderText = "Kitap Adı";
dataGridView1.Columns["yazar"].HeaderText = "Yazar";
dataGridView1.Columns["baskiYili"].HeaderText = "Baskı Yılı";
dataGridView1.Columns["yayinEvi"].HeaderText = "Yayın Evi";
dataGridView1.Columns["sayfaSayisi"].HeaderText = "Sayfa Sayısı";
dataGridView1.Columns["aciklama"].HeaderText = "Açıklama";
dataGridView1.Columns["kategoriID"].Visible = false;

DataTable kategoriler = classSql.TumKategorileriGetir();
kategoriComboBox.DataSource = kategoriler;
kategoriComboBox.DisplayMember = "kategoriAdi";
kategoriComboBox.ValueMember = "kategoriID";
}

private void araBtn_Click(object sender, EventArgs e)
{
    string anahtarKelime = kitapAraTxt.Text.Trim();
    ClassSql classSql = ClassSql.GetInstance();
    DataTable sonuclar = classSql.FiltreleKitap(anahtarKelime);
    dataGridView1.DataSource = sonuclar;
}

private void KaydetBtn_Click(object sender, EventArgs e)
{
    string isbn = ISBNTxt.Text.Trim();
    string kitapAdi = adTxt.Text.Trim();
    string yazar = yazarTxt.Text.Trim();
    string baskiYili = baskiYiliTxt.Text.Trim();
    string yayinEvi = yayinEviTxt.Text.Trim();
    string sayfaSayisi = sayfaSayisiTxt.Text.Trim();
    string aciklama = aciklamaTxt.Text.Trim();
    int kategoriID = Convert.ToInt32(kategoriComboBox.SelectedValue);

    ClassSql classSql = ClassSql.GetInstance();

    if (!classSql.BosKontrolu(ISBNTxt, adTxt, yazarTxt, baskiYiliTxt,
        yayinEviTxt, sayfaSayisiTxt, aciklamaTxt))
    {
        return;
    }

    bool kitapEklendi = classSql.KitapEkle(isbn, kitapAdi, yazar, baskiYili,
        yayinEvi, sayfaSayisi, aciklama, kategoriID);

    if (kitapEklendi)
    {
        MessageBox.Show("Kitap başarıyla eklendi.");
        DataTable kitaplar = classSql.TumKitaplariGetir();
        dataGridView1.DataSource = kitaplar;
    }
    else
    {
        MessageBox.Show("Kitap eklenirken bir hata oluştu.");
    }
}

private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = dataGridView1.Rows[e.RowIndex];
        ISBNTxt.Text = row.Cells["ISBN"].Value.ToString();
        adTxt.Text = row.Cells["ad"].Value.ToString();
        yazarTxt.Text = row.Cells["yazar"].Value.ToString();
        baskiYiliTxt.Text = row.Cells["baskiYili"].Value.ToString();
        yayinEviTxt.Text = row.Cells["yayinEvi"].Value.ToString();
        sayfaSayisiTxt.Text = row.Cells["sayfaSayisi"].Value.ToString();
        aciklamaTxt.Text = row.Cells["aciklama"].Value.ToString();
    }
}

```

```

        if (row.Cells["kategoriID"].Value != DBNull.Value)
        {
            kategoriComboBox.SelectedValue = row.Cells["kategoriID"].Value;
        }
        else
        {
            kategoriComboBox.SelectedIndex = -1;
        }
    }
}

private void guncelleBtn_Click(object sender, EventArgs e)
{
    string isbn = ISBNTxt.Text.Trim();
    string kitapAdi = adTxt.Text.Trim();
    string yazar = yazarTxt.Text.Trim();
    string baskiYili = baskiYiliTxt.Text.Trim();
    string yayinEvi = yayinEviTxt.Text.Trim();
    string sayfaSayisi = sayfaSayisiTxt.Text.Trim();
    string aciklama = aciklamaTxt.Text.Trim();
    int kategoriID = Convert.ToInt32(kategoriComboBox.SelectedValue);

    DataGridViewRow selectedRow = dataGridView1.CurrentRow;
    int kitapID = Convert.ToInt32(selectedRow.Cells["kitapID"].Value);

    ClassSql classSql = ClassSql.GetInstance();

    bool kitapGuncellendi = classSql.KitapGuncelle(kitapID, isbn, kitapAdi,
yazar, baskiYili, yayinEvi, sayfaSayisi, aciklama, kategoriID);

    if (kitapGuncellendi)
    {
        MessageBox.Show("Kitap başarıyla güncellendi.");
        DataTable kitaplar = classSql.TumKitaplarıGetir();
        dataGridView1.DataSource = kitaplar;
    }
    else
    {
        MessageBox.Show("Kitap güncellenirken bir hata oluştu.");
    }
}
}

```

• Üye İşlemleri Formu

- ClassSql:

```

public bool BosKontrolu(params TextBox[] textBoxes)
{
    foreach (var textBox in textBoxes)
    {
        if (string.IsNullOrEmpty(textBox.Text))
        {
            MessageBox.Show($"{textBox.Name} boş bırakılamaz.", "Hata",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return false;
        }
    }
    return true;
}

public DataTable TumUyeleriGetir()
{
    DataTable dataTable = new DataTable();

    using (SqlConnection connection = new SqlConnection(_connectionString))
    {

```

```

        string query = "SELECT * FROM uye";
        SqlCommand command = new SqlCommand(query, connection);
        SqlDataAdapter adapter = new SqlDataAdapter(command);
        adapter.Fill(dataTable);
    }

    return dataTable;
}

public DataTable FiltreleUye(string anahtarKelime)
{
    DataTable dataTable = new DataTable(); // Filtrelenmiş verileri saklamak için
    bir DataTable oluştur

    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "SELECT * FROM uye WHERE Ad LIKE @AnahtarKelime OR Soyad LIKE
@AnahtarKelime";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@AnahtarKelime", "%" + anahtarKelime +
"%");

            try
            {
                SqlDataAdapter adapter = new SqlDataAdapter(command);
                adapter.Fill(dataTable);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Filtreleme sırasında bir hata oluştu:
{ex.Message}");
            }
        }
    }

    return dataTable;
}

public bool UyeEkle(string ad, string soyad, string email, string telefon, string
adres)
{
    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string checkQuery = "SELECT COUNT(*) FROM uye WHERE email = @Email";

        using (SqlCommand checkCommand = new SqlCommand(checkQuery, connection))
        {
            checkCommand.Parameters.AddWithValue("@Email", email);
            int count = (int)checkCommand.ExecuteScalar();

            if (count > 0)
            {
                return false;
            }
        }

        string query = "INSERT INTO uye (ad, soyad, email, telefon, adres) VALUES
(@Ad, @Soyad, @Email, @Telefon, @Adres)";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Ad", ad);
            command.Parameters.AddWithValue("@Soyad", soyad);
            command.Parameters.AddWithValue("@Email", email);
            command.Parameters.AddWithValue("@Telefon", telefon);
            command.Parameters.AddWithValue("@Adres", adres);
        }
    }
}

```

```

        try
        {
            command.ExecuteNonQuery();
            return true;
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Üye eklenirken bir hata oluştu: {ex.Message}",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return false;
        }
    }
}

public bool UyeGuncelle(int uyeID, string ad, string soyad, string email, string
telefon, string adres)
{
    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "UPDATE uye SET ad = @Ad, soyad = @Soyad, email = @Email,
telefon = @Telefon, adres = @Adres WHERE uyeID = @UyeID";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@UyeID", uyeID);
            command.Parameters.AddWithValue("@Ad", ad);
            command.Parameters.AddWithValue("@Soyad", soyad);
            command.Parameters.AddWithValue("@Email", email);
            command.Parameters.AddWithValue("@Telefon", telefon);
            command.Parameters.AddWithValue("@Adres", adres);

            try
            {
                command.ExecuteNonQuery();
                return true;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Üye güncellenirken bir hata oluştu:
{ex.Message}");
                return false;
            }
        }
    }
}

public bool UyeSil(int uyeID)
{
    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "DELETE FROM uye WHERE uyeID = @UyeID";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@UyeID", uyeID);

            try
            {
                command.ExecuteNonQuery();
                return true;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Üye silinirken bir hata oluştu: {ex.Message}");
                return false;
            }
        }
    }
}

```

- **FormUyeAnasayfa.cs:**

```
public partial class FormUyeIslemleri : Form
{
    private ClassSql classSql;
    public FormUyeIslemleri()
    {
        InitializeComponent();
        classSql = ClassSql.GetInstance();
    }

    private void FormUyeIslemleri_Load(object sender, EventArgs e)
    {
        DataTable dataTable = classSql.TumUyeleriGetir();
        dataGridView1.DataSource = dataTable;
        dataGridView1.Columns["uyeID"].HeaderText = "Üye ID";
        dataGridView1.Columns["ad"].HeaderText = "Ad";
        dataGridView1.Columns["soyad"].HeaderText = "Soyad";
        dataGridView1.Columns["eMail"].HeaderText = "Email";
        dataGridView1.Columns["telefon"].HeaderText = "Telefon";
        dataGridView1.Columns["adres"].HeaderText = "Adres";
    }

    private void araBtn_Click(object sender, EventArgs e)
    {
        string anahtarKelime = uyeAraTxt.Text.Trim();

        ClassSql classSql = ClassSql.GetInstance();

        DataTable sonuclar = classSql.FiltreleUye(anahtarKelime);

        dataGridView1.DataSource = sonuclar;
    }

    private void kaydetBtn_Click(object sender, EventArgs e)
    {
        string ad = adTxt.Text.Trim();
        string soyad = soyadTxt.Text.Trim();
        string email = mailTxt.Text.Trim();
        string telefon = telefonTxt.Text.Trim();
        string adres = adresTxt.Text.Trim();

        ClassSql classSql = ClassSql.GetInstance();

        if (!classSql.BosKontrolu(adTxt, soyadTxt, mailTxt, sifreTxt,
        telefonTxt, adresTxt))
        {
            return;
        }

        bool basarili = classSql.UyeEkle(ad, soyad, email, telefon, adres);

        if (basarili)
        {
            MessageBox.Show("Üye başarıyla eklendi.");
            DataTable uyeler = classSql.TumUyeleriGetir();
            dataGridView1.DataSource = uyeler;
        }
        else
        {
            MessageBox.Show("Bu e-posta adresiyle kayıtlı kullanıcı
            bulunmaktadır.", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

```

private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = dataGridView1.Rows[e.RowIndex];

        adTxt.Text = row.Cells["ad"].Value.ToString();
        soyadTxt.Text = row.Cells["soyad"].Value.ToString();
        mailTxt.Text = row.Cells["email"].Value.ToString();
        telefonTxt.Text = row.Cells["telefon"].Value.ToString();
        adresTxt.Text = row.Cells["adres"].Value.ToString();
    }
}

private void guncelleBtn_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        DataGridViewRow row = dataGridView1.SelectedRows[0];
        int uyeID = Convert.ToInt32(row.Cells["uyeID"].Value);

        string ad = adTxt.Text.Trim();
        string soyad = soyadTxt.Text.Trim();
        string email = mailTxt.Text.Trim();
        string telefon = telefonTxt.Text.Trim();
        string adres = adresTxt.Text.Trim();

        bool basarili = classSql.UyeGuncelle(uyeID, ad, soyad, email,
telefon, adres);

        if (basarili)
        {
            MessageBox.Show("Üye başarıyla güncellendi.");
            DataTable uyeler = classSql.TumUyeleriGetir();
            dataGridView1.DataSource = uyeler;
        }
        else
        {
            MessageBox.Show("Üye güncellenirken bir hata oluştu.");
        }
    }
    else
    {
        MessageBox.Show("Lütfen güncellemek istediğiniz üyeyi seçin.");
    }
}

private void SilBtn_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        DataGridViewRow row = dataGridView1.SelectedRows[0];
        int uyeID = Convert.ToInt32(row.Cells["uyeID"].Value);

        DialogResult dialogResult = MessageBox.Show("Bu üyeyi silmek
istediğinizden emin misiniz?", "Üye Sil", MessageBoxButtons.YesNo);
        if (dialogResult == DialogResult.Yes)
        {
            bool basarili = classSql.UyeSil(uyeID);

            if (basarili)
            {
                MessageBox.Show("Üye başarıyla silindi.");
                DataTable uyeler = classSql.TumUyeleriGetir();
                dataGridView1.DataSource = uyeler;
            }
            else
            {
                MessageBox.Show("Üye silinirken bir hata oluştu.");
            }
        }
    }
}

```

```

    }
    else
    {
        MessageBox.Show("Lütfen silmek istediğiniz üyeyi seçin.");
    }
}
}

```

- **Emanet İşlemleri Formu**

- **ClassSql:**

```

public DataTable FiltreleUye(string anahtarKelime)
{
    DataTable dataTable = new DataTable(); // Filtrelenmiş verileri saklamak için
    bir DataTable oluştur

    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "SELECT * FROM uye WHERE Ad LIKE @AnahtarKelime OR Soyad LIKE
@AnahtarKelime";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@AnahtarKelime", "%" + anahtarKelime +
"%");

            try
            {
                SqlDataAdapter adapter = new SqlDataAdapter(command);
                adapter.Fill(dataTable);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Filtreleme sırasında bir hata oluştu:
{ex.Message}");
            }
        }
    }

    return dataTable;
}

```

```

public DataTable FiltreleKitap(string anahtarKelime)
{
    DataTable dataTable = new DataTable();

    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "SELECT * FROM kitap WHERE ISBN LIKE @AnahtarKelime OR Ad
LIKE @AnahtarKelime OR yazar LIKE @AnahtarKelime ";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@AnahtarKelime", "%" + anahtarKelime +
"%");

            try
            {
                SqlDataAdapter adapter = new SqlDataAdapter(command);
                adapter.Fill(dataTable);
            }
        }
    }
}

```



```

        catch (Exception ex)
        {
            MessageBox.Show($"Filtreleme sırasında bir hata oluştu:
{ex.Message}");
        }
    }

    return dataTable;
}

public DataTable FiltreleIslem(string anahtarKelime)
{
    DataTable dataTable = new DataTable();

    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "SELECT * FROM islem WHERE emanetDurumu LIKE @AnahtarKelime
OR islemID LIKE @AnahtarKelime";
        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@AnahtarKelime", "%" + anahtarKelime +
"%");

            try
            {
                SqlDataAdapter adapter = new SqlDataAdapter(command);
                adapter.Fill(dataTable);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Filtreleme sırasında bir hata oluştu:
{ex.Message}");
            }
        }

        return dataTable;
    }
}

public DataTable TumIslemleriGetir()
{
    DataTable dataTable = new DataTable();
    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        string query = "SELECT * FROM islem";
        SqlCommand command = new SqlCommand(query, connection);
        SqlDataAdapter adapter = new SqlDataAdapter(command);
        adapter.Fill(dataTable);
    }

    return dataTable;
}

public bool TeslimVer(int uyeID, int kitapID, DateTime islemTarihi, DateTime
iadeTarihi, int adminID)
{
    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        using (SqlCommand command = new SqlCommand("TeslimVer", connection))
        {
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.AddWithValue("@uyeID", uyeID);
            command.Parameters.AddWithValue("@kitapID", kitapID);
            command.Parameters.AddWithValue("@islemTarihi", islemTarihi);
            command.Parameters.AddWithValue("@iadeTarihi", iadeTarihi);
            command.Parameters.AddWithValue("@adminID", adminID);
        }
    }
}

```

```

        try
        {
            command.ExecuteNonQuery();
            return true;
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Teslim verilirken bir hata oluştu: {ex.Message}");
            return false;
        }
    }
}

public bool TeslimAl(int islemID, int kitapID)
{
    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        using (SqlCommand command = new SqlCommand("TeslimAl", connection))
        {
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.AddWithValue("@islemID", islemID);
            command.Parameters.AddWithValue("@kitapID", kitapID);

            try
            {
                command.ExecuteNonQuery();
                return true;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Teslim alınırken bir hata oluştu: {ex.Message}");
                return false;
            }
        }
    }
}

public DataTable TumRaftakiKitaplariGetir()
{
    DataTable dataTable = new DataTable();

    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        string query = "SELECT * FROM kitap WHERE kitapDurumu = 1"; // kitapDurumu 1
        // olan kitaplar rafta demek
        SqlCommand command = new SqlCommand(query, connection);
        SqlDataAdapter adapter = new SqlDataAdapter(command);
        adapter.Fill(dataTable);
    }

    return dataTable;
}

```

- FormEmanetIslemleri.cs:

```

public partial class FormEmanetIslemleri : Form
{
    private ClassSql classSql;
    public FormEmanetIslemleri()
    {
        InitializeComponent();
        classSql = ClassSql.GetInstance();
    }
    private void FormEmanetIslemleri_Load(object sender, EventArgs e)
    {
        kitapIDTxt.Enabled = false;
    }
}

```

```

uyeIDTxt.Enabled = false;
DataTable dataTable = classSql.TumUyeleriGetir();
DataTable dataTable2 = classSql.TumRaftakiKitaplariGetir();
DataTable dataTable3 = classSql.TumIslemleriGetir();

// DataGridView'e tüm üyeleri yükler
dataGridView1.DataSource = dataTable;
dataGridView2.DataSource = dataTable2;
dataGridView3.DataSource = dataTable3;

dateTimePicker1.Value = DateTime.Today;
dateTimePicker2.Value = DateTime.Today.AddDays(15);

//üye
dataGridView1.Columns["uyeID"].HeaderText = "Üye ID";
dataGridView1.Columns["ad"].HeaderText = "Ad";
dataGridView1.Columns["soyad"].HeaderText = "Soyad";
dataGridView1.Columns["eMail"].HeaderText = "Email";
dataGridView1.Columns["telefon"].HeaderText = "Telefon";
dataGridView1.Columns["adres"].HeaderText = "Adres";

//kitap
dataGridView2.Columns["kitapID"].HeaderText = "Kitap ID";
dataGridView2.Columns["ISBN"].HeaderText = "ISBN";
dataGridView2.Columns["ad"].HeaderText = "Kitap Adı";
dataGridView2.Columns["yazar"].HeaderText = "Yazar";
dataGridView2.Columns["baskiYili"].HeaderText = "Baskı Yılı";
dataGridView2.Columns["yayinEvi"].HeaderText = "Yayın Evi";
dataGridView2.Columns["sayfaSayisi"].HeaderText = "Sayfa Sayısı";
dataGridView2.Columns["aciklama"].HeaderText = "Açıklama";
}

private void uyeAraBtn_Click(object sender, EventArgs e)
{
    string anahtarKelime = uyeAraTxt.Text.Trim();

    ClassSql classSql = ClassSql.GetInstance();

    DataTable sonuclar = classSql.FiltreleUye(anahtarKelime);

    dataGridView1.DataSource = sonuclar;
}

private void kitapAraBtn_Click(object sender, EventArgs e)
{
    string anahtarKelime = kitapAraTxt.Text.Trim();
    ClassSql classSql = ClassSql.GetInstance();
    DataTable sonuclar = classSql.FiltreleKitap(anahtarKelime);
    dataGridView2.DataSource = sonuclar;
}

private void islemAraBtn_Click(object sender, EventArgs e)
{
    string anahtarKelime = islemAraTxt.Text.Trim();
    ClassSql classSql = ClassSql.GetInstance();
    DataTable sonuclar = classSql.FiltreleIslem(anahtarKelime);
    dataGridView3.DataSource = sonuclar;
}

private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = dataGridView1.Rows[e.RowIndex];
        uyeIDTxt.Text = row.Cells["uyeID"].Value.ToString();
    }
}

private void dataGridView2_CellClick(object sender,
DataGridViewCellEventArgs e)
{

```

```

        if (e.RowIndex >= 0)
        {
            DataGridViewRow row = dataGridView2.Rows[e.RowIndex];
            kitapIDTxt.Text = row.Cells["kitapID"].Value.ToString();
        }
    }

    private void teslimVerBtn_Click(object sender, EventArgs e)
    {
        if (dataGridView2.SelectedRows.Count > 0)
        {
            int uyeID = int.Parse(uyeIDTxt.Text);
            int kitapID =
Convert.ToInt32(dataGridView2.SelectedRows[0].Cells["kitapID"].Value);
            DateTime islemTarihi = dateTimePicker1.Value;
            DateTime iadeTarihi = dateTimePicker2.Value;
            int adminID = ClassVeriler.AdminID;

            ClassSql classSql = ClassSql.GetInstance();
            bool basarili = classSql.TeslimVer(uyeID, kitapID, islemTarihi,
iadeTarihi, adminID);

            if (basarili)
            {
                MessageBox.Show("Kitap başarıyla emanet verildi.");
                dataGridView2.Rows.RemoveAt(dataGridView2.SelectedRows[0].Index);
                DataTable islemTablosu = classSql.TumIslemleriGetir();
                dataGridView3.DataSource = islemTablosu;
            }
            else
            {
                MessageBox.Show("Kitap emanet verilirken bir hata oluştu.");
            }
        }
        else
        {
            MessageBox.Show("Lütfen emanet vermek istediğiniz kitabı seçin.");
        }
    }

    private void teslimAlBtn_Click(object sender, EventArgs e)
    {
        if (dataGridView3.SelectedRows.Count > 0)
        {
            int islemID =
Convert.ToInt32(dataGridView3.SelectedRows[0].Cells["islemID"].Value);
            int kitapID =
Convert.ToInt32(dataGridView3.SelectedRows[0].Cells["kitapID"].Value);

            ClassSql classSql = ClassSql.GetInstance();
            bool basarili = classSql.TeslimAl(islemID, kitapID);

            if (basarili)
            {
                MessageBox.Show("Kitap başarıyla geri alındı.");

                dataGridView3.Rows.RemoveAt(dataGridView3.SelectedRows[0].Index);
                DataTable kitapTablosu = classSql.TumRaftakiKitaplarıGetir();
                dataGridView2.DataSource = kitapTablosu;
            }
            else
            {
                MessageBox.Show("Kitap geri alınırken bir hata oluştu.");
            }
        }
        else
        {
            MessageBox.Show("Lütfen geri almak istediğiniz işlemi seçin.");
        }
    }
}

```

- **Admin İşlemleri Formu**

- **ClassSql:**

```
public bool BosKontrolu(params TextBox[] textBoxes)
{
    foreach (var textBox in textBoxes)
    {
        if (string.IsNullOrEmpty(textBox.Text))
        {
            MessageBox.Show($"{textBox.Name} boş bırakılamaz.", "Hata",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
            return false;
        }
    }
    return true;
}

public DataTable TumAdminleriGetir()
{
    DataTable dataTable = new DataTable();

    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        string query = "SELECT * FROM admin";
        SqlCommand command = new SqlCommand(query, connection);
        SqlDataAdapter adapter = new SqlDataAdapter(command);
        adapter.Fill(dataTable);
    }

    return dataTable;
}

public DataTable FiltreleAdmin(string anahtarKelime)
{
    DataTable dataTable = new DataTable();

    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "SELECT * FROM admin WHERE Ad LIKE @AnahtarKelime OR Soyad LIKE @AnahtarKelime";
        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@AnahtarKelime", "%" + anahtarKelime + "%");

            try
            {
                SqlDataAdapter adapter = new SqlDataAdapter(command);
                adapter.Fill(dataTable);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Filtreleme sırasında bir hata oluştu: {ex.Message}");
            }
        }
    }

    return dataTable;
}

public bool AdminEkle(string ad, string soyad, string email, string telefon, string adres)
{
    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        connection.Open();
```

```

string checkQuery = "SELECT COUNT(*) FROM admin WHERE email = @Email";
using (SqlCommand checkCommand = new SqlCommand(checkQuery, connection))
{
    checkCommand.Parameters.AddWithValue("@Email", email);
    int count = (int)checkCommand.ExecuteScalar();

    if (count > 0)
    {
        return false;
    }
}

string query = "INSERT INTO admin (ad, soyad, email, telefon, adres) VALUES
(@Ad, @Soyad, @Email, @Telefon, @Adres)";

using (SqlCommand command = new SqlCommand(query, connection))
{
    command.Parameters.AddWithValue("@Ad", ad);
    command.Parameters.AddWithValue("@Soyad", soyad);
    command.Parameters.AddWithValue("@Email", email);
    command.Parameters.AddWithValue("@Telefon", telefon);
    command.Parameters.AddWithValue("@Adres", adres);

    try
    {
        command.ExecuteNonQuery();
        return true;
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Admin eklenirken bir hata oluştu: {ex.Message}",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
}
}

public bool AdminGuncelle(int adminID, string ad, string soyad, string email, string
telefon, string adres)
{
    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "UPDATE admin SET ad = @Ad, soyad = @Soyad, email = @Email,
telefon = @Telefon, adres = @Adres WHERE adminID = @AdminID";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@AdminID", adminID);
            command.Parameters.AddWithValue("@Ad", ad);
            command.Parameters.AddWithValue("@Soyad", soyad);
            command.Parameters.AddWithValue("@Email", email);
            command.Parameters.AddWithValue("@Telefon", telefon);
            command.Parameters.AddWithValue("@Adres", adres);

            try
            {
                command.ExecuteNonQuery();
                return true;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Admin güncellenirken bir hata oluştu:
{ex.Message}");
                return false;
            }
        }
    }
}

```

```

public bool AdminSil(int adminID)
{
    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "DELETE FROM admin WHERE adminID = @AdminID";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@AdminID", adminID);

            try
            {
                command.ExecuteNonQuery();
                return true;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Admin silinirken bir hata oluştu: {ex.Message}");
                return false;
            }
        }
    }
}

```

- FormAdminIslemleri.cs:

```

public partial class FormAdminIslemleri : Form
{
    private ClassSql classSql;
    public FormAdminIslemleri()
    {
        InitializeComponent();
        classSql = ClassSql.GetInstance();
    }

    private void FormAdminIslemleri_Load(object sender, EventArgs e)
    {
        DataTable dataTable = classSql.TumAdminleriGetir();
        dataGridView1.DataSource = dataTable;
        dataGridView1.Columns["adminID"].HeaderText = "Admin ID";
        dataGridView1.Columns["ad"].HeaderText = "Ad";
        dataGridView1.Columns["soyad"].HeaderText = "Soyad";
        dataGridView1.Columns["eMail"].HeaderText = "Email";
        dataGridView1.Columns["telefon"].HeaderText = "Telefon";
        dataGridView1.Columns["adres"].HeaderText = "Adres";
    }

    private void araBtn_Click(object sender, EventArgs e)
    {
        string anahtarKelime = adminAraTxt.Text.Trim();

        ClassSql classSql = ClassSql.GetInstance();

        DataTable sonuclar = classSql.FiltreleAdmin(anahtarKelime);

        dataGridView1.DataSource = sonuclar;
    }

    private void kaydetBtn_Click(object sender, EventArgs e)
    {
        string ad = adTxt.Text.Trim();
        string soyad = soyadTxt.Text.Trim();
        string email = mailTxt.Text.Trim();
        string telefon = telefonTxt.Text.Trim();
        string adres = adresTxt.Text.Trim();

        ClassSql classSql = ClassSql.GetInstance();
    }
}

```

```

        if (!classSql.BosKontrolu(adTxt, soyadTxt, mailTxt, sifreTxt, telefonTxt,
adresTxt))
        {
            return;
        }

        bool basarili = classSql.AdminEkle(ad, soyad, email, telefon, adres);

        if (basarili)
        {
            MessageBox.Show("Admin başarıyla eklendi.");
            DataTable adminler = classSql.TumAdminleriGetir();
            dataGridView1.DataSource = adminler;
        }
        else
        {
            MessageBox.Show("Bu e-posta adresiyle kayıtlı kullanıcı bulunmaktadır.",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        if (e.RowIndex >= 0)
        {
            DataGridViewRow row = dataGridView1.Rows[e.RowIndex];

            adTxt.Text = row.Cells["ad"].Value.ToString();
            soyadTxt.Text = row.Cells["soyad"].Value.ToString();
            mailTxt.Text = row.Cells["email"].Value.ToString();
            telefonTxt.Text = row.Cells["telefon"].Value.ToString();
            adresTxt.Text = row.Cells["adres"].Value.ToString();
        }
    }

    private void guncelleBtn_Click(object sender, EventArgs e)
    {
        if (dataGridView1.SelectedRows.Count > 0)
        {
            DataGridViewRow row = dataGridView1.SelectedRows[0];
            int adminID = Convert.ToInt32(row.Cells["adminID"].Value);

            string ad = adTxt.Text.Trim();
            string soyad = soyadTxt.Text.Trim();
            string email = mailTxt.Text.Trim();
            string telefon = telefonTxt.Text.Trim();
            string adres = adresTxt.Text.Trim();

            bool basarili = classSql.AdminGuncelle(adminID, ad, soyad, email,
telefon, adres);

            if (basarili)
            {
                MessageBox.Show("Admin başarıyla güncellendi.");
                DataTable adminler = classSql.TumAdminleriGetir();
                dataGridView1.DataSource = adminler;
            }
            else
            {
                MessageBox.Show("Admin güncellenirken bir hata oluştu.");
            }
        }
        else
        {
            MessageBox.Show("Lütfen güncellemek istediğiniz admini seçin.");
        }
    }

    private void SilBtn_Click(object sender, EventArgs e)
    {
        if (dataGridView1.SelectedRows.Count > 0)

```



```

{
    DataGridViewRow row = dataGridView1.SelectedRows[0];
    int adminID = Convert.ToInt32(row.Cells["adminID"].Value);

    DialogResult dialogResult = MessageBox.Show("Bu admini silmek istediğinizden emin misiniz?", "Admin Sil", MessageBoxButtons.YesNo);
    if (dialogResult == DialogResult.Yes)
    {
        bool basarili = classSql.AdminSil(adminID);

        if (basarili)
        {
            MessageBox.Show("Admin başarıyla silindi.");
            DataTable adminler = classSql.TumAdminleriGetir();
            dataGridView1.DataSource = adminler;
        }
        else
        {
            MessageBox.Show("Admin silinirken bir hata oluştu.");
        }
    }
    else
    {
        MessageBox.Show("Lütfen silmek istediğiniz admini seçin.");
    }
}
}

```

• Üye Anasayfa Formu

- ClassSql:

```

public bool BosKontrolu(params TextBox[] textBoxes)
{
    foreach (var textBox in textBoxes)
    {
        if (string.IsNullOrEmpty(textBox.Text))
        {
            MessageBox.Show($"{textBox.Name} boş bırakılamaz.", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return false;
        }
    }
    return true;
}

public DataTable UzerimdekiKitaplariGetir(int uyeID)
{
    DataTable dt = new DataTable();
    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = @"
        SELECT k.kitapID, k.ad AS KitapAdi, k.yazar AS Yazar, i.islemTarihi AS
        EmanetTarihi, i.iadeTarihi AS IadeTarihi
        FROM islem i
        JOIN kitap k ON i.kitapID = k.kitapID
        WHERE i.uyeID = @uyeID AND i.emanetDurumu = 'EMANETTE'";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@uyeID", uyeID);
            using (SqlDataAdapter adapter = new SqlDataAdapter(command))
            {
                adapter.Fill(dt);
            }
        }
    }
}

```

```

    }
    }
    return dt;
}

public bool KitapIadeTarihiniGuncelle(int kitapID, DateTime yeniIadeTarihi)
{
    using (SqlConnection connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "UPDATE islem SET iadeTarihi = @YeniIadeTarihi WHERE kitapID
= @KitapID";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@YeniIadeTarihi", yeniIadeTarihi);
            command.Parameters.AddWithValue("@KitapID", kitapID);

            try
            {
                int rowsAffected = command.ExecuteNonQuery();
                return rowsAffected > 0;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Kitap iade tarihi güncellenirken bir hata oluştu:
{ex.Message}");
                return false;
            }
        }
    }
}

```

- FormUyeAnasayfa.cs:

```

public partial class FormUyeIslemleri : Form
{
    private ClassSql classSql;
    public FormUyeIslemleri()
    {
        InitializeComponent();
        classSql = ClassSql.GetInstance();
    }

    private void FormUyeIslemleri_Load(object sender, EventArgs e)
    {
        DataTable dataTable = classSql.TumUyeleriGetir();
        dataGridView1.DataSource = dataTable;
        dataGridView1.Columns["uyeID"].HeaderText = "Üye ID";
        dataGridView1.Columns["ad"].HeaderText = "Ad";
        dataGridView1.Columns["soyad"].HeaderText = "Soyad";
        dataGridView1.Columns["eMail"].HeaderText = "Email";
        dataGridView1.Columns["telefon"].HeaderText = "Telefon";
        dataGridView1.Columns["adres"].HeaderText = "Adres";
    }

    private void araBtn_Click(object sender, EventArgs e)
    {
        string anahtarKelime = uyeAraTxt.Text.Trim();

        ClassSql classSql = ClassSql.GetInstance();
        DataTable sonuclar = classSql.FiltreleUye(anahtarKelime);

        dataGridView1.DataSource = sonuclar;
    }

    private void kaydetBtn_Click(object sender, EventArgs e)

```

```

{
    string ad = adTxt.Text.Trim();
    string soyad = soyadTxt.Text.Trim();
    string email = mailTxt.Text.Trim();
    string telefon = telefonTxt.Text.Trim();
    string adres = adresTxt.Text.Trim();

    ClassSql classSql = ClassSql.GetInstance();

    if (!classSql.BosKontrolu(adTxt, soyadTxt, mailTxt, sifreTxt, telefonTxt,
adresTxt))
    {
        return;
    }

    bool basarili = classSql.UyeEkle(ad, soyad, email, telefon, adres);

    if (basarili)
    {
        MessageBox.Show("Üye başarıyla eklendi.");
        DataTable uyeler = classSql.TumUyeleriGetir();
        dataGridView1.DataSource = uyeler;
    }
    else
    {
        MessageBox.Show("Bu e-posta adresiyle kayıtlı kullanıcı bulunmaktadır.",
"Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
    {
        DataGridViewRow row = dataGridView1.Rows[e.RowIndex];

        adTxt.Text = row.Cells["ad"].Value.ToString();
        soyadTxt.Text = row.Cells["soyad"].Value.ToString();
        mailTxt.Text = row.Cells["email"].Value.ToString();
        telefonTxt.Text = row.Cells["telefon"].Value.ToString();
        adresTxt.Text = row.Cells["adres"].Value.ToString();
    }
}

private void guncelleBtn_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        DataGridViewRow row = dataGridView1.SelectedRows[0];
        int uyeID = Convert.ToInt32(row.Cells["uyeID"].Value);

        string ad = adTxt.Text.Trim();
        string soyad = soyadTxt.Text.Trim();
        string email = mailTxt.Text.Trim();
        string telefon = telefonTxt.Text.Trim();
        string adres = adresTxt.Text.Trim();

        bool basarili = classSql.UyeGuncelle(uyeID, ad, soyad, email, telefon,
adres);

        if (basarili)
        {
            MessageBox.Show("Üye başarıyla güncellendi.");
            DataTable uyeler = classSql.TumUyeleriGetir();
            dataGridView1.DataSource = uyeler;
        }
        else
        {
            MessageBox.Show("Üye güncellenirken bir hata oluştu.");
        }
    }
    else

```

```

        {
            MessageBox.Show("Lütfen güncellemek istediğiniz üyeyi seçin.");
        }
    }

    private void SilBtn_Click(object sender, EventArgs e)
    {
        if (dataGridView1.SelectedRows.Count > 0)
        {
            DataGridViewRow row = dataGridView1.SelectedRows[0];
            int uyeID = Convert.ToInt32(row.Cells["uyeID"].Value);

            DialogResult dialogResult = MessageBox.Show("Bu üyeyi silmek istediğinizden emin misiniz?", "Üye Sil", MessageBoxButtons.YesNo);
            if (dialogResult == DialogResult.Yes)
            {
                bool basarili = classSql.UyeSil(uyeID);

                if (basarili)
                {
                    MessageBox.Show("Üye başarıyla silindi.");
                    DataTable uyeler = classSql.TumUyeleriGetir();
                    dataGridView1.DataSource = uyeler;
                }
                else
                {
                    MessageBox.Show("Üye silinirken bir hata oluştu.");
                }
            }
        }
        else
        {
            MessageBox.Show("Lütfen silmek istediğiniz üyeyi seçin.");
        }
    }
}

```

- **Üzerimdeki Kitaplar Formu**

- **ClassSql:**

```

public DataTable GetUyeDataByID(int uyeID)
{
    DataTable dataTable = new DataTable();
    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "SELECT * FROM uye WHERE uyeID = @UyeID";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@UyeID", uyeID);

            using (SqlDataReader reader = command.ExecuteReader())
            {
                dataTable.Load(reader);
            }
        }
    }
    return dataTable;
}

public bool UyeBilgileriGuncelle(int uyeID, string ad, string soyad, string email, string sifre, string telefon, string adres)
{
    try

```

```

        {
            using (var connection = new SqlConnection(_connectionString))
            {
                connection.Open();
                string query = "UPDATE uye SET ad = @Ad, soyad = @Soyad, eMail = @Email, sifre = @Sifre, telefon = @Telefon, adres = @Adres WHERE uyeID = @UyeID";

                using (SqlCommand command = new SqlCommand(query, connection))
                {
                    command.Parameters.AddWithValue("@Ad", ad);
                    command.Parameters.AddWithValue("@Soyad", soyad);
                    command.Parameters.AddWithValue("@Email", email);
                    command.Parameters.AddWithValue("@Sifre", sifre);
                    command.Parameters.AddWithValue("@Telefon", telefon);
                    command.Parameters.AddWithValue("@Adres", adres);
                    command.Parameters.AddWithValue("@UyeID", uyeID);

                    int affectedRows = command.ExecuteNonQuery();
                    return affectedRows > 0;
                }
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Üye bilgileri güncellenirken bir hata oluştu: " +
ex.Message);
        return false;
    }
}

```

- **FormUzerimdekiKitaplar.cs:**

```

public partial class FormUzerimdekiKitaplar : Form
{
    private ClassSql classSql;
    private int uyeID;

    public FormUzerimdekiKitaplar(int uyeID)
    {
        InitializeComponent();
        classSql = ClassSql.GetInstance();
        this.uyeID = uyeID;
    }

    private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
    }

    private void FormUzerimdekiKitaplar_Load(object sender, EventArgs e)
    {
        DataTable uzerimdekiKitaplar = classSql.UzerimdekiKitaplariGetir(uyeID);
        dataGridView1.DataSource = uzerimdekiKitaplar;

        dataGridView1.AutoGenerateColumns = true;
    }

    private void kitapUzatBtn_Click(object sender, EventArgs e)
    {
        if (dataGridView1.SelectedRows.Count > 0)
        {
            DateTime iadeTarihi =
Convert.ToDateTime(dataGridView1.SelectedRows[0].Cells["IadeTarihi"].Value);

            DateTime yeniIadeTarihi = iadeTarihi.AddDays(15);

            dataGridView1.SelectedRows[0].Cells["IadeTarihi"].Value =
yeniIadeTarihi.ToString("yyyy-MM-dd");
        }
    }
}

```

```

        int kitapID =
Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["kitapID"].Value);

        ClassSql classSql = ClassSql.GetInstance();

        bool basarili = classSql.KitapIadeTarihiniGuncelle(kitapID,
yeniIadeTarihi);

        if (basarili)
        {
            MessageBox.Show("Kitap iade tarihi başarıyla güncellendi.");
        }
        else
        {
            MessageBox.Show("Kitap iade tarihi güncellenirken bir hata
oluşturdu.");
        }
    }
    else
    {
        MessageBox.Show("Lütfen bir kitap seçin.");
    }
}
}

```

• Profil İşlemleri Formu

- ClassSql:

```

public DataTable GetUyeDataByID(int uyeID)
{
    DataTable dataTable = new DataTable();
    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();
        string query = "SELECT * FROM uye WHERE uyeID = @UyeID";

        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@UyeID", uyeID);

            using (SqlDataReader reader = command.ExecuteReader())
            {
                dataTable.Load(reader);
            }
        }
    }
    return dataTable;
}

public bool UyeBilgileriGuncelle(int uyeID, string ad, string soyad, string email,
string sifre, string telefon, string adres)
{
    try
    {
        using (var connection = new SqlConnection(_connectionString))
        {
            connection.Open();
            string query = "UPDATE uye SET ad = @Ad, soyad = @Soyad, eMail = @Email,
sifre = @Sifre, telefon = @Telefon, adres = @Adres WHERE uyeID = @UyeID";

            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@Ad", ad);

```

```

        command.Parameters.AddWithValue("@Soyad", soyad);
        command.Parameters.AddWithValue("@Email", email);
        command.Parameters.AddWithValue("@Sifre", sifre);
        command.Parameters.AddWithValue("@Telefon", telefon);
        command.Parameters.AddWithValue("@Adres", adres);
        command.Parameters.AddWithValue("@UyeID", uyeID);

        int affectedRows = command.ExecuteNonQuery();
        return affectedRows > 0;
    }
}
}
}
catch (Exception ex)
{
    Console.WriteLine("Üye bilgileri güncellenirken bir hata oluştu: " +
ex.Message);
    return false;
}
}
}

```

- FormProfilIslemleri.cs:

```

public partial class FormProfilIslemleri : Form
{
    private ClassSql classSql;

    public FormProfilIslemleri()
    {
        InitializeComponent();
        classSql = ClassSql.GetInstance();
        LoadProfileData(ClassVeriler.UyeID);
    }

    private void FormProfilIslemleri_Load(object sender, EventArgs e)
    {
    }

    private void KaydetBtn_Click(object sender, EventArgs e)
    {
        try
        {
            string ad = adTxt.Text;
            string soyad = soyadTxt.Text;
            string email = mailTxt.Text;
            string sifre = sifreTxt.Text;
            string telefon = telefonTxt.Text;
            string adres = adresTxt.Text;

            bool basarili = classSql.UyeBilgileriGuncelle(ClassVeriler.UyeID, ad,
soyad, email, sifre, telefon, adres);
            if (basarili)
            {
                MessageBox.Show("Üye bilgileri başarıyla güncellendi.");
            }
            else
            {
                MessageBox.Show("Üye bilgileri güncellenirken bir hata oluştu.");
            }
        }
        catch { }
    }
}

```

```

    }
    catch (Exception ex)
    {
        MessageBox.Show("Üye bilgileri güncellenirken bir hata oluştu: " +
ex.Message);
    }
}

private void LoadProfileData(int uyeID)
{
    DataTable userData = classSql.GetUyeDataByID(uyeID);

    if (userData.Rows.Count > 0)
    {
        DataRow row = userData.Rows[0];
        adTxt.Text = row["ad"].ToString();
        soyadTxt.Text = row["soyad"].ToString();
        mailTxt.Text = row["email"].ToString();
        sifreTxt.Text = row["sifre"].ToString();
        telefonTxt.Text = row["telefon"].ToString();
        adresTxt.Text = row["adres"].ToString();
    }
}
}

```