İSTANBUL ÜNİVERSİTESİ
CERRAHPAŞA
MÜHENDİSLİK FAKÜLTESİ

İstanbul Üniversitesi-Cerrahpaşa

Mühendislik Fakültesi

STAJ DEFTERİ

İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
MÜHENDİSLİK FAKÜLTESİ

Staj Yapan Öğrencinin

Adı	ERMAND	Soyadı	HARUNI
Öğrenci No.	1316200111	Bölümü	Elektrik- Elektronik Mühendisliği
Eğitim-Öğretim yılı	2023-2024	Telefon No.	05394111175
		e-posta	ermand.haruni@ogr.iuc.edu.tr
İkametgâh adresi	SELİMPAŞA MAH. 3006 SK. NO:4 İÇ KAPI NO:2 SİLİVRİ/İSTANBUL		

Staj Yapılan İşyeri Yeri ve Staj Tarihi

Adı	Bilişim Bilgisayar Hizmetleri Anonim Şirketi				
Adresi	Kısıklı Mahallesi Aydınoğlu Sokak No:21 K.Çamlıca Üsküdar/İstanbul				
Üretim/Hizmet Alanı	Bankacılık kartlı ödeme sistemleri yazılım geliştirme				
Telefon No.	0216 521 14 14		Faks No.	0216 491 91 80	
e-posta	contact@banksoft.com.tr		Web adresi	www.banksoft.com.tr	
Staja başlama tarihi	01/07/2024	Bitiş tarihi	12/08/2024	Süresi (gün)	30
Staj türü (varsa)	Zorunlu				

STAJ DEFTERİNİN KULLANILMASINDA DİKKAT EDİLECEK HUSUSLAR

I. **Yapılan iş ile ilgili açıklamalar:**

Her sayfanın başlık kısmı doldurulacaktır.

İş konusu hakkında özet bilgi verilecektir.

Yapılan iş teknik bir rapor şeklinde yazılacaktır; stajyer ile iş yeri yetkilisi tarafından staj defterinin her sayfası imzalanacaktır.

II. **Staj defterinin kabul edilmesi ile ilgili açıklamalar:**

Defterin ilk ve son sayfasındaki staj tablosu ile iş yeri tabloları staj yapılan iş yerinin resmi mührü ile mühürlenmiş veya kaşelenmiş, ilgili yerler yetkili tarafından doldurularak imzalanmış olacaktır.

Belgeler belirtilen adrese kayıtlı olarak (kargo, taahhütlü posta, vb.) postalanır ya da kapalı zarfta, zarf üzeri imzalı, kaşeli ve/veya mühürlü olarak stajyer öğrenciye verilir.

Adres: İ.Ü.C. Mühendislik Fakültesi Dekanlığı Avcılar Kampüsü-İstanbul

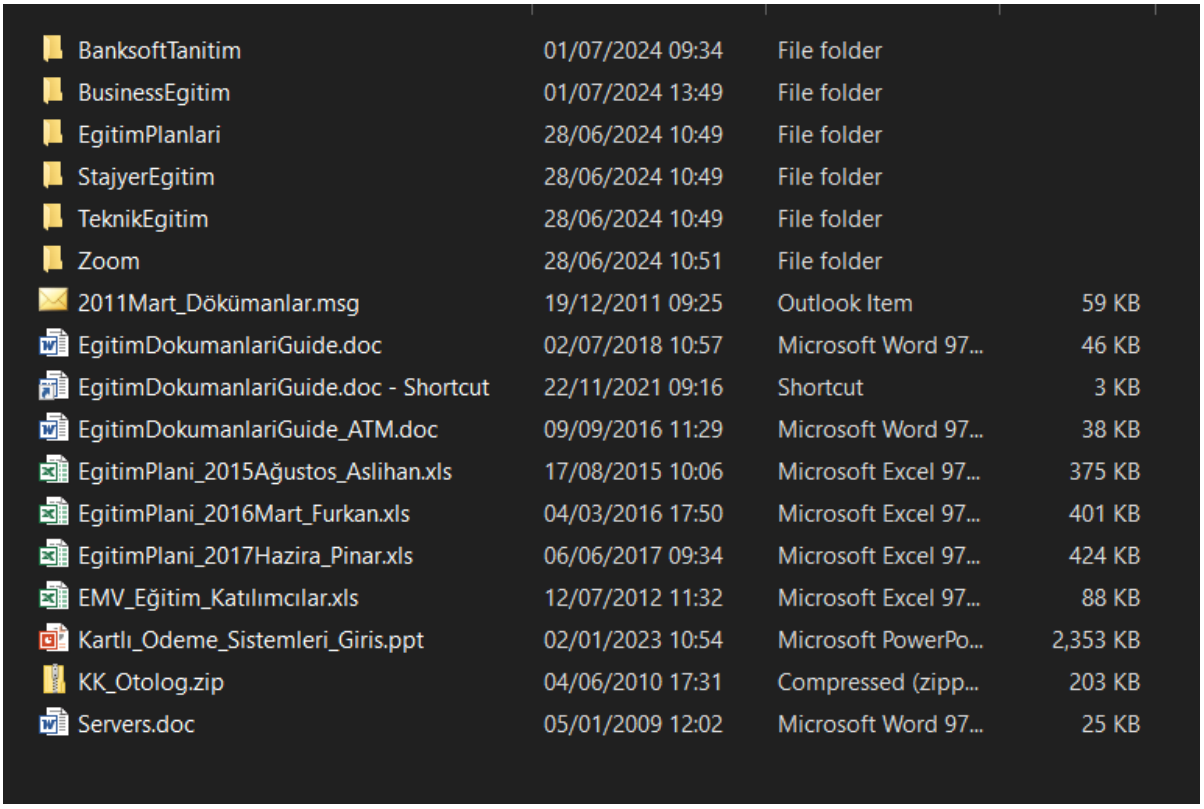
Telefon: (212) 473 70 70 **Faks**: (212) 473 71 82

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:01/07/2024

Yapılan iş: Banksoft şirketini ve çözümlerini tanıtan dökümanların okunması.

Staj sürecinin ilk aşamasında, sorumluluklarımı ve projelerimi içeren kapsamlı bir program tanıtıldı. Bu tanıtımda, staj boyunca neler yapacağım ve hangi adımları takip edeceğim detaylıca açıklandı. Programın ilk adımında, Banksoft şirketinin kuruluş tarihi, faaliyet alanları, ve ödeme sistemleri üzerine ürettiği çözümlerle ilgili bilgilendirici bir sunum incelendi. Bu sunum, şirketin nasıl kurulduğunu, hangi vizyon ve misyonla yola çıktığını ve ödeme sistemleri alanında nasıl bir uzmanlık geliştirdiğini anlamam için oldukça faydalıydı. Ayrıca, Banksoft'un hangi bankalarla çalıştığı, bu bankalara nasıl çözümler sunduğu ve sektördeki yerini nasıl güçlendirdiği üzerinde de duruldu.

Sunumun ardından, şirketin yazılım geliştirme sürecinde kullanılan yazım ve isimlendirme kuralları ile ekran tasarlama standartlarına dair bir doküman üzerinde çalıştım. Bu doküman, kod yazımında nasıl bir disiplinin takip edildiğini, hangi konvansiyonların kullanıldığını ve bu kuralların yazılımın sürdürülebilirliği açısından ne kadar önemli olduğunu vurguluyordu. Aynı zamanda, kullanıcı arayüzlerinin tasarımında hangi standartların uygulandığını, kullanıcı deneyimini iyileştirmek için nelere dikkat edildiğini ve tasarım sürecinde hangi adımların izlendiğini öğrenmiş oldum.



Resim-1: Dökümanlar

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:02/07/2024

Yapılan iş: SQL sorgu dilinin öğrenilmesi ve pratik yapılması

Yöneticimin hazırladığı SQL eğitimi videosunu izledikten sonra, SQL diline dair temel ve ileri düzeydeki komutları öğrenmek amacıyla https://www.w3schools.com/sql/ üzerinden çalışmalar yapmaya başladım. Bu süreçte, öncelikle veritabanı sorgulama ve veri yönetimi açısından en önemli komutlar olan SELECT, INSERT, UPDATE, ve DELETE komutlarını öğrendim. Bu komutlar, veritabanı üzerinde veri çekme, yeni veri ekleme, mevcut veriyi güncelleme ve silme işlemlerini gerçekleştirmemi sağladı.

Temel komutları öğrendikten sonra, veritabanı tabloları arasındaki ilişkileri anlamamı sağlayan INNER JOIN ve LEFT JOIN gibi join komutlarına geçtim. Bu komutlar, farklı tablolar arasında veri çekme işlemlerini yaparken hangi verilerin eşleştirileceğini belirlememi sağladı. Ardından, veriler üzerinde toplu işlemler gerçekleştiren MIN, MAX, SUM, ve COUNT gibi aggregate fonksiyonları inceledim. Bu fonksiyonlar, veri kümeleri üzerinde minimum ve maksimum değerleri bulma, toplam alma ve belirli kriterlere göre satır sayısını hesaplama gibi işlemleri gerçekleştirmeme olanak tanıdı.

Teorik bilgilerimi pratiğe dökmek amacıyla, Microsoft SQL Server üzerinde örnek bir veritabanı tablosu oluşturdum. Bu tablo üzerinde, öğrendiğim SQL komutlarını kullanarak çeşitli sorgular yazdım ve bu komutları gerçek veri üzerinde test ettim. Bu pratikler, SQL dilini daha iyi anlamamı ve öğrendiklerimi pekiştirmemi sağladı.

select2.sql - BSS02...SOFT\Stajyer (446))

```sql
-- ORDER BY --
-- Belli bir sütuna göre sort işlemi yapmaya sağlar
SELECT * FROM Products ORDER BY Price;

-- DESC Descending sıralama yapar (büyükten küçüğe), ASC as
SELECT * FROM Products ORDER BY Price DESC;

-- Birden fazla sütun ile sıralanabilir
SELECT * FROM Customers
ORDER BY Country ASC, CustomerName DESC;


-- SELECT TOP ilk 3 ünü alır
SELECT TOP 3 * FROM Customers;

-- AND operatorü conditionlar için
SELECT *
FROM ermand_tbl1
WHERE Country = 'Spain' AND CustomerName LIKE 'G%';

-- OR operatörü
SELECT *
FROM Customers
WHERE Country = 'Germany' OR Country = 'Spain'

-- NOT operatörü
-- if !(Country == "Spain")
SELECT * FROM Customers
```

Resim-2: Yazdığım SQL sorguları

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:03/07/2024

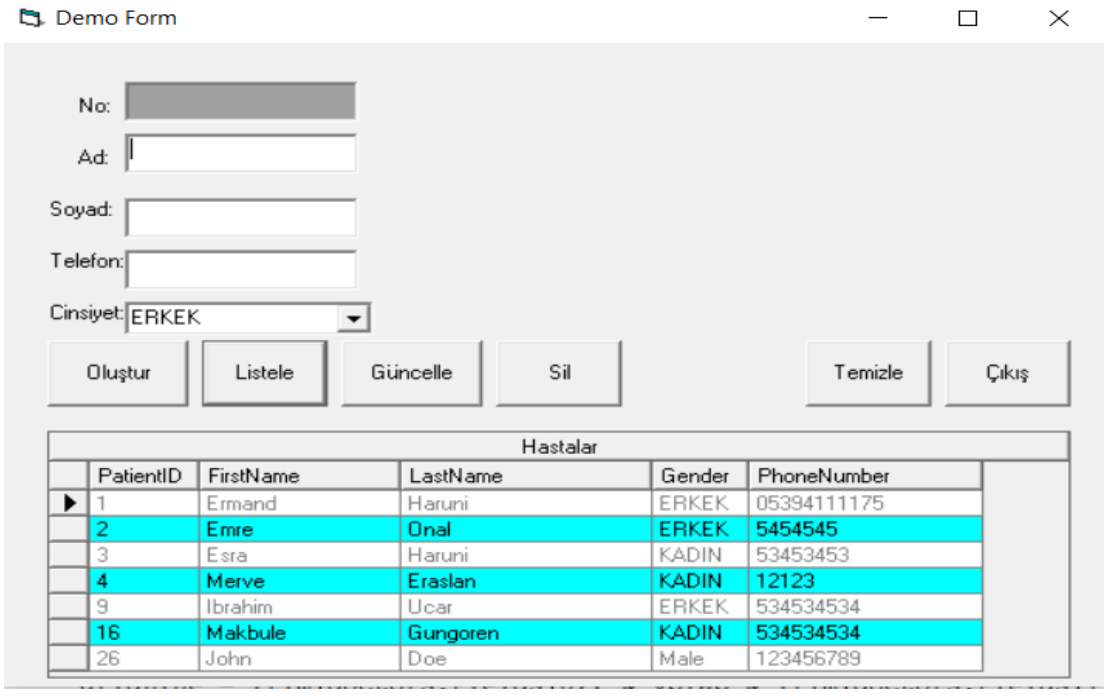
Yapılan iş: Visual Basic kullanarak hastane randevu formu tasarlanması

Yöneticimin hazırladığı Visual Basic eğitim videosunu izledikten sonra, bu yeni bilgileri pratiğe dökmek amacıyla benden istenen Visual Basic projesini gerçekleştirmeye başladım. İlk olarak, projenin temelini oluşturacak veritabanı yapısını kurmak için CREATE TABLE komutlarını kullanarak üç adet tablo oluşturdum: Patients, Doctors, ve Appointments. Bu tablolar, projenin ana veri yapısını oluşturdu ve hasta, doktor ve randevu bilgilerini yönetmek için gereken temel altyapıyı sağladı.

Veritabanı yapılandırmasını tamamladıktan sonra, Visual Basic üzerinde dört adet form tasarladım: frmMain, frmDoctors, frmPatients, ve frmAppointments. Bu formlar, kullanıcının veritabanı ile etkileşime geçmesini sağlayacak kullanıcı arayüzlerini temsil ediyordu. Her bir form, belirli bir tabloya yönelik işlemleri gerçekleştirmek için tasarlandı.

Visual Basic üzerinde ADO (ActiveX Data Objects) bağlantılarını kurduktan sonra, her formda ilgili tabloya ekleme, silme ve güncelleme işlemlerini gerçekleştirecek butonlar ekledim. Bu butonlar sayesinde kullanıcılar, veritabanındaki hasta, doktor ve randevu kayıtlarını kolayca yönetebileceklerdi. Ek olarak, verileri tablolar halinde listelemek için SSDB Grid komponentini kullandım. Bu komponent, veritabanındaki bilgilerin görsel olarak temsil edilmesine olanak tanıdı ve kullanıcıların mevcut verileri kolayca görüntülemesini sağladı.

Sonuç olarak, bu projeyle hem Visual Basic'teki temel veri tabanı işlemlerini öğrenmiş oldum hem de bir yazılım projesinde kullanıcı arayüzü tasarımı ve veri yönetimi arasındaki ilişkiyi daha iyi anladım. Bu deneyim, Visual Basic ve ADO kullanarak nasıl etkili bir şekilde veri tabanı ile etkileşime geçileceğini kavramama yardımcı oldu.



Resim-3: Hastalar formu

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:04/07/2024

Yapılan iş: Winfows Form üzerinde hastane randevu sistemini yazma

Visual Basic'te oluşturduğum form projesini tamamladıktan sonra, aynı projeyi C# Windows Forms üzerinde de yapmaya başladım. Projeye, dört adet form oluşturarak başladım ve bu formlara gerekli textBox ve butonları ekledim. Kod tekrarından kaçınmak ve projeyi daha modüler hale getirmek için DbManager adında bir sınıf oluşturdum. Bu sınıf içinde, veritabanı işlemlerini daha düzenli ve yeniden kullanılabilir hale getirmek amacıyla iki temel fonksiyon geliştirdim: ExecuteQuery ve ExecuteCommand.

ExecuteQuery fonksiyonunu, veritabanından veri okuma işlemleri için kullanıyorum. Genellikle SELECT sorguları için bu fonksiyonu tercih ediyorum, çünkü SQL sorgusunu çalıştırarak sonuçları bir DataTable gibi bir veri yapısında döndürüyor. Bu da verilerin kullanıcı arayüzünde görüntülenmesini veya raporlar oluşturulmasını sağlıyor.

Diğer yandan, ExecuteCommand fonksiyonunu ise veritabanında veri değiştiren işlemler için kullanıyorum. INSERT, UPDATE, DELETE gibi komutları çalıştırmak için ideal olan bu fonksiyon, veritabanına yeni veri eklemek, mevcut veriyi güncellemek ya da silmek gibi işlemler için kullanılıyor. Genellikle bir sonuç döndürmez, ancak istenirse etkilenen satır sayısı gibi bir bilgi döndürebiliyor.

Veritabanı bağlantısını daha esnek ve yönetilebilir hale getirmek amacıyla, ConnectionString bilgisini App.config dosyasından almayı öğrendim. Bu sayede, bağlantı bilgilerini merkezi bir yerden yönetebilir hale geldim. Projenin ilerleyen aşamalarında, DbManager sınıfını SQL Server'daki Stored Procedure'leri kullanabilecek şekilde güncelledim, böylece veritabanı işlemlerini daha güvenli ve verimli bir şekilde gerçekleştirebileceğim bir altyapı oluşturmuş oldum.

```
public class DbManager
{
    private readonly string connectionString;

    public DbManager()
    {
        this.connectionString = ConfigurationManager.ConnectionStrings["AppointmentConnString"].ConnectionString;
    }

    public SqlDataReader ExecuteReader(string query, List<SqlParameter> parameters = null)
    {
        SqlConnection conn = new SqlConnection();
        conn.ConnectionString = connectionString;
        conn.Open();
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = query;
        if (parameters != null)
        {
            cmd.CommandType = CommandType.StoredProcedure;
            foreach (var param in parameters)
            {
                cmd.Parameters.Add(param);
            }
        }
        else
        {
            cmd.CommandType = CommandType.Text;
        }
        SqlDataReader reader = cmd.ExecuteReader(CommandBehavior.CloseConnection);
        return reader;
    }

    public void ExecuteCommand(string query, List<SqlParameter> parameters = null)
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        using (SqlCommand command = new SqlCommand(query, connection))
        {
            if (parameters != null)
            {
                foreach (var param in parameters)
                {
                    command.Parameters.Add(param);
                }
            }

            connection.Open();
            command.ExecuteNonQuery();
        }
    }
}
```

Resim-4:DbManager sınıfı

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:05/07/2024

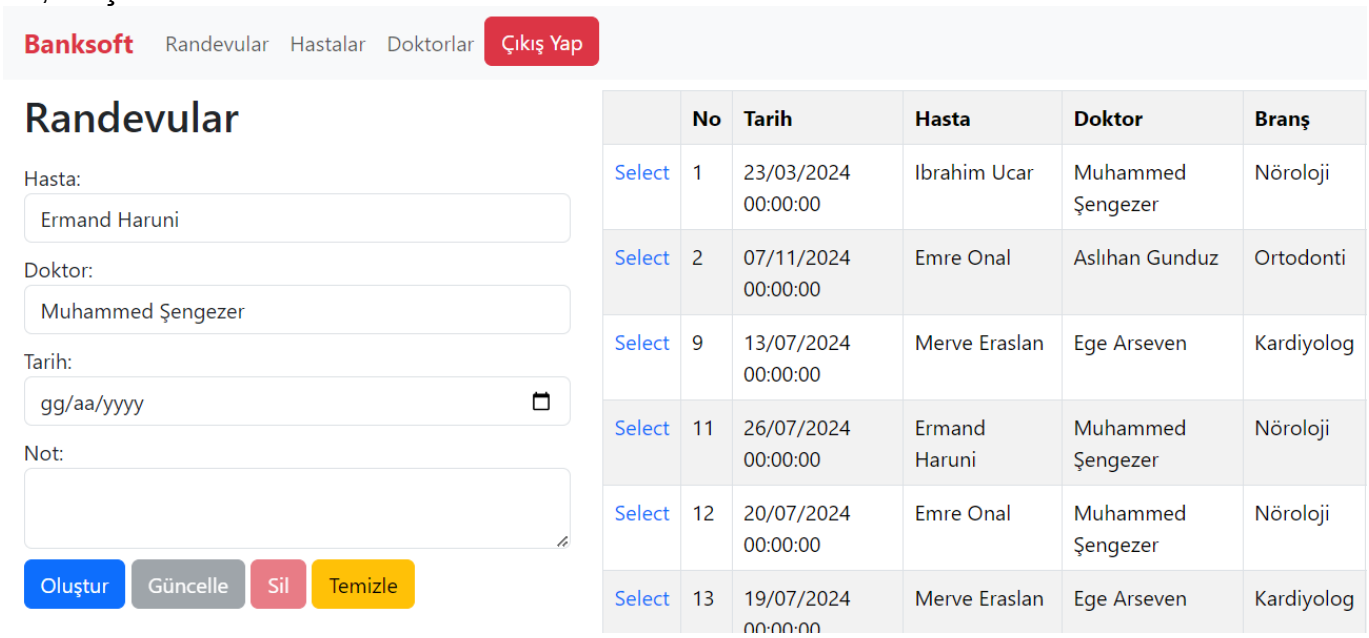
Yapılan iş: ASP.NET Web Form üzerinde hastane randevu sistemini yazmak

Windows Forms uygulamasını başarıyla tamamladıktan sonra, aynı işlevselliğe sahip bir uygulamayı ASP.NET Web Forms üzerinde nasıl geliştirebileceğimi araştırmaya başladım. Web Forms, farklı bir platform olduğu için, öncelikle arayüz tasarımını nasıl daha hızlı ve verimli bir şekilde yapabileceğimi düşündüm. Bu amaçla, CSS ve JavaScript ile zaman kaybetmemek için Bootstrap kütüphanesini kullanmaya karar verdim. Bootstrap, duyarlı ve modern arayüzler oluşturmayı kolaylaştırdığı için, benim için ideal bir çözüm oldu.

ASP.NET Web Forms'un sunduğu <asp:Button>, <asp:TextBox>, ve <asp:GridView> gibi temel componentleri kullanarak, kullanıcı arayüzünü hızlı bir şekilde oluşturdum. Bu componentler, hem veri girişi hem de veri görüntüleme işlemlerini kolayca gerçekleştirmeme olanak tanıdı.

Uygulamada üç ana sayfa oluşturdum: Patients.aspx, Appointments.aspx, ve Doctors.aspx. Bu sayfalarda, daha önce Windows Forms uygulamamda geliştirdiğim DbManager sınıfını tekrar kullanarak veri tabanı işlemlerini kolayca entegre ettim. Bu sınıf, veritabanı işlemlerini soyutlayarak, kod tekrarından kaçınmama ve projeyi daha modüler bir hale getirmeme yardımcı oldu. Ekleme, silme, güncelleme ve listeleme işlemlerini bu sayfalara entegre etmek, DbManager sınıfı sayesinde oldukça hızlı ve sorunsuz bir şekilde gerçekleşti.

Bu temel işlevleri tamamladıktan sonra, uygulamada kullanıcı girişi ve yetkilendirme işlemlerini nasıl gerçekleştirebileceğimi araştırdım. Authentication katmanı, kullanıcıların güvenli bir şekilde giriş yapabilmesi ve yetkilendirilmiş işlemleri gerçekleştirebilmesi için kritik bir unsur. Bu konuda yaptığım araştırmalar sayesinde, ASP.NET Web Forms uygulamamı kullanıcı doğrulama ve yetkilendirme ile nasıl daha güvenli hale getirebileceğimi öğrendim ve projeyi bu bilgilerle güçlendirmeyi planladım. Günü, bu şekilde tamamladım.



Resim-5: Randevular Web Sayfası

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

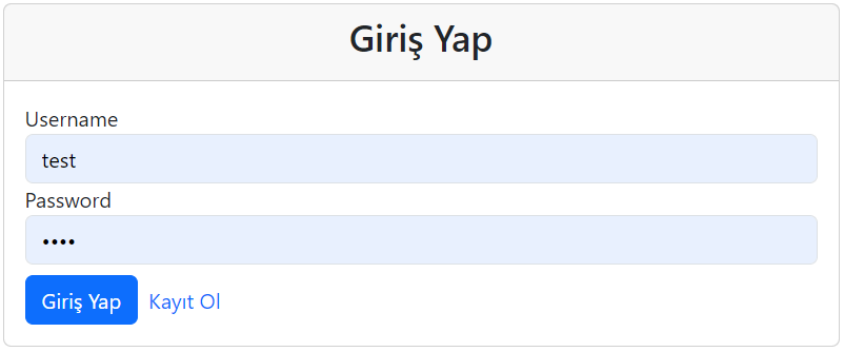
İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:08/07/2024

Yapılan iş: ASP.NET Web Forms üzerinde Authentication katmanının oluşturulması

Bugün ASP.NET Web Forms uygulamamda Authentication katmanını oluşturdum. Bu süreçte, kullanıcıların giriş yapabilmesi ve yeni hesap oluşturabilmesi için iki temel sayfa tasarladım: Login.aspx ve Register.aspx. Kullanıcıların bilgilerini güvenli bir şekilde saklayabilmek amacıyla, veritabanında Users adında bir tablo oluşturdum. Bu tabloyu kullanıcı adı ve şifreleri içerecek şekilde oluşturdum.

Login sayfasında, kullanıcının girdiği kullanıcı adı ve şifreyi veritabanında saklanan bilgilerle karşılaştırdım. Kullanıcı adı ve şifre doğruysa, başarılı bir oturum açma işlemi için bir AuthenticationCookie oluşturdum ve bu çerezi kullanıcıya sundum. Çıkış yap butonu ekleyerek, kullanıcıların çıkış işlemi yapabilmesi için cookie'yi silmelerini sağladım. Bu, oturum kapatma işlemini güvenli bir şekilde gerçekleştirdi.

Ayrıca, veritabanında şifrelerin güvenliğini artırmak amacıyla, kullanıcı şifrelerini kayıt işlemi sırasında AES algoritması ile şifreleyerek veritabanında sakladım. Bu yöntem, şifrelerin yetkisiz erişimlere karşı korunmasını sağladı ve güvenliği önemli ölçüde artırdı.



Resim-6: Giriş Yapma Sayfası

```
protected void btnLogin_Click(object sender, EventArgs e)
{
    try
    {
        // AES anahtarınızı ve IV'nizi buradan alabilirsiniz.
        string aesKey = "Your_AES_Key"; // AES anahtarı
        string aesIV = "Your_AES_IV"; // AES IV

        // Parolayı AES ile şifrele
        string encryptedPassword = EncryptPassword(txtPassword.Text, aesKey, aesIV);

        List<SqlParameter> parameters = new List<SqlParameter>()
        {
            new SqlParameter("@Username", txtUsername.Text),
            new SqlParameter("@Password", encryptedPassword)
        };

        DataTable result = dbManager.ExecuteQuery("sp_AuthenticateUser", parameters);
        if (result.Rows.Count > 0)
        {
            FormsAuthentication.SetAuthCookie(txtUsername.Text, false);
            string redirectUrl = FormsAuthentication.GetRedirectUrl(txtUsername.Text, false);
            Response.Redirect(redirectUrl);
        }
        else
        {
            ClientScript.RegisterStartupScript(this.GetType(), "alert", "alert('Invalid username or password');", true);
        }
    }
    catch (Exception ex)
    {
        ClientScript.RegisterStartupScript(this.GetType(), "alert", string.Format("alert('{0}');", ex.Message), true);
    }
}
```

Resim-7: Giriş Yapma ve şifreleme fonksiyonu

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:09/07/2024

Yapılan iş: Şifrelerin hashlenerek veri tabanında tutulması ve web service kullanımı

AES şifrelemesinden sonra, şifre güvenliğini daha da artırmak amacıyla SHA256 hash algoritmasını kullanarak şifreleri hashleyip veritabanında sakladım. SHA256, kriptografik olarak daha güçlü bir algoritma olduğundan, bu yöntemle şifrelerin güvenliğini maksimum seviyeye çıkardım. SHA256 kullanarak şifreleri hashlemek, şifrelerin yetkisiz erişimlere karşı daha iyi korunmasını sağladı çünkü hash algoritmaları geri dönüşümsüz olduğu için orijinal şifrelerin geri elde edilmesini zorlaştırır.

Web Form kısmını tamamladıktan sonra, ASP.NET Web Service projesini oluşturdum. Bu projede, daha önce gerçekleştirdiğim veritabanı işlemlerini web servisleri aracılığıyla yapmaya başladım. .asmx dosyalarını kullanarak, web servis üzerinde çeşitli fonksiyonlar oluşturduk. Bu sayede, yerel ağ üzerindeki diğer kullanıcılar, bu web servis aracılığıyla veritabanı işlemlerine erişim sağlayabilir hale geldi. Web servisleri, veritabanı işlemlerini merkezi bir noktadan yönetmeye olanak tanır ve uygulamanın erişilebilirliğini artırır, çünkü ağ üzerindeki herkes bu servislere erişebilir ve veritabanı işlemlerini gerçekleştirebilir.

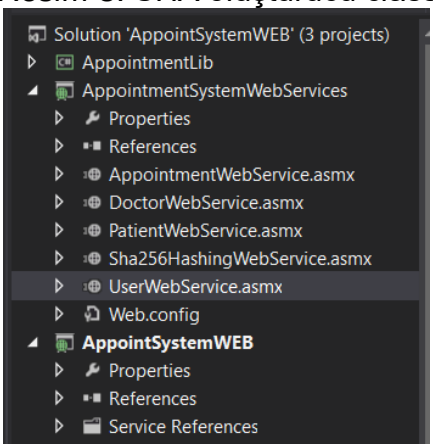
Bu iki aşama ile hem veritabanı güvenliğini arttırdım hem de sistemin erişilebilirliğini kolaylaştırdım. SHA256 ile şifre güvenliğini artırarak kullanıcı verilerini korudum, web servisleri ile de veritabanı işlemlerine erişim sağlayan merkezi bir yapı oluşturmuş oldum.

```
protected void btnRegister_Click(object sender, EventArgs e)
{
    try
    {
        string hashedPassword = Sha256HashGenerator.Generate(txtPassword.Text);
        List<SqlParameter> parameters = new List<SqlParameter>()
        {
            new SqlParameter("@Username", txtUsername.Text),
            new SqlParameter("@Password", hashedPassword),
        };
        dbManager.ExecuteCommand("sp_InsertUser", parameters);
        Response.Redirect("Login.aspx");

    }
    catch (Exception ex)
    {
        ClientScript.RegisterStartupScript(this.GetType(), "alert", string.Format("alert('{0}');", ex.Message), true);
    }
}
```

ClientScriptManager Page.ClientScript
(The documentation cache is still being constructed. Please try again in a few seconds.)

Resim-8: SHA oluşturucu class



Resim-9:Web Service'lerin oluşturulması

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:10/07/2024

Yapılan iş: XML oluşturma ve XML'den veritabanına aktarma.

Web Service projesini tamamladıktan sonra, C# dilinde XML dosyaları oluşturan veya okuyan bir program geliştirdim. Bu programın ilk aşamasında, bir DataTable objesindeki verileri XML formatına dönüştüren bir uygulama yazdım. Bunun için DataTable sınıfının WriteXml yöntemini kullandım. Bu yöntem, DataTable içindeki verilerin XML dosyası olarak kaydedilmesini sağladı.
İkinci aşamada ise, var olan bir XML dosyasındaki kayıtları okuyup veritabanına kaydeden bir program yazdım. XML dosyasını okumak için DataSet sınıfının ReadXml yöntemini kullandım. Bu yöntem, XML dosyasındaki verileri bir DataSet objesine dönüştürdü ve ardından bu DataSet objesini DataTable objesine dönüştürdüm. Son olarak, DbManager sınıfının ExecuteCommand yöntemini kullanarak, bu DataTable objesindeki kayıtları veritabanındaki ilgili tabloya kaydettim. Bu program, verilerin XML formatında saklanması ve daha sonra veritabanına aktarılması işlemlerini verimli bir yapmamı sağladı.

```
class Program
{
    static void Main(string[] args)
    {
        StoredProcedureToXml("Doctors" ,"sp_GetAllDoctors");
        Console.ReadLine();
    }

    static void StoredProcedureToXml(string tableName, string procedure, List<SqlParameter> parameters = null)
    {
        string projectDirectory = Directory.GetParent(Directory.GetCurrentDirectory()).Parent.Parent.FullName;

        string xmlFilePath = Path.Combine(projectDirectory, String.Format("XmlFiles/{0}", tableName + ".xml"));
        DbManager dbManager = new DbManager();
        DataTable dataTable = dbManager.ExecuteQuery(procedure, parameters);
        dataTable.TableName = tableName;

        dataTable.WriteXml(xmlFilePath);
        Console.WriteLine(String.Format("{0} başarıyla XML dosyasına yazıldı.", procedure));
    }
}
```

Resim-10:Veritabanından gelen tabloyu XML'e aktaran kod

```
class Program
{
    static void Main(string[] args)
    {
        string tableName = "Doctors";
        string fileName = tableName + ".xml";
        string projectDirectory = Directory.GetParent(Directory.GetCurrentDirectory()).Parent.Parent.FullName;
        string xmlFilePath = Path.Combine(projectDirectory, String.Format("XmlFiles/{0}", fileName));
        string spName = "sp_InsertDoctor";

        DataSet dataSet = new DataSet();
        dataSet.ReadXml(xmlFilePath);

        DbManager dbManager = new DbManager();
        DataTable dataTable = dataSet.Tables[tableName];
        foreach (DataRow row in dataTable.Rows)
        {
            List<SqlParameter> parameters = new List<SqlParameter>()
            {
                new SqlParameter("@FirstName", row["FirstName"] ),
                new SqlParameter("@LastName", row["
                new SqlParameter("@PhoneNumber", ro
                new SqlParameter("@Email", row["Ema
                new SqlParameter("@Branch", row["Branch"] ),
            };
            dbManager.ExecuteCommand(spName, parameters);
        }
        Console.WriteLine("Veriler başarıyla veritabanına eklendi.");
        Console.ReadKey();
    }
}
```

class System.String
(The documentation cache is still being constructed. Please try again in a few sec

Resim-11:XML dosyasındaki kayıtları veritabanına aktaran kod

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:11/07/2024

Yapılan iş: C# ile birbirleriyle anlık haberleşen soket uygulaması yazma

C# üzerinde gerçek zamanlı bir soket uygulaması geliştirmek için öncelikle C#'taki Socket kütüphanesi ile ilgili detaylı bir araştırma yaptım. Bu araştırmamda soket programlamanın temel prensiplerini ve nasıl uygulama geliştirebileceğimi öğrendim.

Ardından, soket uygulamamı geliştirmek için SocketServer adında bir proje oluşturup, bu projede 43665 portunu kullanarak bir endpoint tanımladım. Bu endpoint, soket uygulamamın diğer soketlerle iletişim kuracağı noktayı belirliyor.

Proje kapsamında, Socket sınıfından bir nesne oluşturup, bu nesneye tanımladığım endpoint'i bağlamak için Bind metodunu kullandım. Bu işlem, soketimi belirli bir IP adresi ve port numarası ile ilişkilendirerek dinlemeye başlamasını sağladı.

Sonraki adımda, Listen fonksiyonunu kullanarak soketimin diğer soketlerden gelen bağlantı taleplerini dinlemeye başladım. Listen fonksiyonu, gelen bağlantı isteklerini kabul etmeye ve yönetmeye hazır hale getirdi.

Bağlantıları dinleme aşamasında, sürekli olarak bir bağlantı talebi beklemek için bir sonsuz döngü (while) kurdum. Bu döngü içinde Accept fonksiyonunu kullanarak bir soketin bağlanmasını bekledim. Bir soket bağlandığında, bu soketten gelen mesajları almak ve ekrana yazdırmak için Console.WriteLine fonksiyonunu kullandım.

Bu projede soketler arasında gerçek zamanlı veri iletişimi sağlamak ve gelen mesajları ekranda görüntülemek için temel bir soket sunucusu oluşturmuş oldum.

```
Socket listener = new Socket(ip.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
try
{
    // bağlantı beklenecek olan endpointi sokete bağla
    listener.Bind(localEp);

    // serverın dinleme işlemini başlat
    listener.Listen(maxListenerCount);
    while (true)
    {
        Console.WriteLine("Waiting for a connection.....");
        // bağlantı isteği geldiğinde kabul et
        Socket handler = listener.Accept();
        Console.WriteLine("------ New Socket Connected ------");
        data = null;
        // end of file görene kadar clientı dinlemeye devam et
        while (true)
        {
            int byteRec = handler.Receive(buffer);
            data = Encoding.ASCII.GetString(buffer, 0, byteRec);
            Console.WriteLine("Client: " + data);
            // gelen mesajda EOF varsa döngüyü bitir
            if (data.IndexOf("<EOF>") > -1)
            {
                handler.Shutdown(SocketShutdown.Both);
                handler.Close();
                break;
            }
            Console.Write("Server: ");
            string msgStr = Console.ReadLine();
            byte[] msg = Encoding.ASCII.GetBytes(msgStr);
            handler.Send(msg);
        }
    }
}
```

Resim-12:Soketin server kodları

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:12/07/2024

Yapılan iş: Multithreaded Socket Server yazılması

Dün oluşturduğum soket projesini multithreaded destekleyecek şekilde geliştirmek için araştırma yaptım. Bu araştırmanın ardından, C#'taki Thread kütüphanesini kullanarak, aynı anda birden fazla client'ın bağlanıp birbirleriyle iletişim kurabileceği bir sohbet uygulaması geliştirdim. Her client için yeni bir thread oluşturup, her bağlantının kendi iş parçacığında yönetilmesini sağladım. Bu sayede, birden fazla client aynı anda sunucuya bağlanıp sohbet edebildi.

Projemi tamamladıktan sonra, ThreadPool kütüphanesi hakkında da araştırma yaptım. ThreadPool, iş parçacıklarını daha verimli bir şekilde yönetir ve kullanır, bu da performans iyileştirmeleri sağlar. Bu bilgiyi doğrulamak için bir örnek uygulama yaparak, normal iş parçacıkları ile ThreadPool kullanarak yapılan işlemlerin tamamlanma sürelerini karşılaştırdım. Elde ettiğim sonuçlara göre, ThreadPool kullanarak yapılan işlemlerin tamamlanma süresinin daha kısa olduğunu gözlemledim. Bu sayede ThreadPool'un iş parçacığı thread yönetimini optimize ederek, daha hızlı ve verimli işlem yapılmasını sağladığını öğrendim.

```
class Server
{
    static Socket listenerSocket;
    static List<ClientManager> clients;

    static void Main(string[] args)
    {
        Console.WriteLine("Starting server");

        var hostName = Dns.GetHostName();
        IPHostEntry ipHost = Dns.GetHostEntry(hostName);
        IPAddress ip = ipHost.AddressList[0];
        IPEndPoint localEp = new IPEndPoint(ip, 4242);

        clients = new List<ClientManager>();
        listenerSocket = new Socket(ip.AddressFamily,SocketType.Stream ,ProtocolType.Tcp);
        listenerSocket.Bind(localEp);
        Thread listenerThread = new Thread(ListenThread);
        listenerThread.Start();
        Console.ReadLine();
    }
```

Resim-13:Multithreaded socket server kodları

```
static void Main(string[] args)
{
    Console.Write("Enter your name:");
    name = Console.ReadLine();

    A:Console.Clear();
    var hostName = Dns.GetHostName();
    IPHostEntry ipHost = Dns.GetHostEntry(hostName);
    IPAddress ip = ipHost.AddressList[0];
    IPEndPoint serverEp = new IPEndPoint(ip, 4242);

    socket = new Socket(ip.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

    try
    {
        socket.Connect(serverEp);
    }
    catch (Exception e)
    {
        Console.WriteLine("Could not connect to host: " + e.Message);
        Thread.Sleep(1000);
        goto A;
    }

    Thread t = new Thread(ReceiveData);
    t.Start();

    while (true)
    {
        string input = Console.ReadLine();

        Packet p = new Packet(PacketType.Chat, id);
        p.data.Add(name);
        p.data.Add(input);
        socket.Send(p.ToBytes());
        Thread.Sleep(300);
    }
}
```

Resim-14: Client kodları

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:16/07/2024

Yapılan iş: C'de ODBC kütüphanesi ve DLL kullanımı.

C dilinde veritabanı işlemlerini yönetmek amacıyla ODBC (Open Database Connectivity) kütüphanesini kullanarak bir konsol uygulaması geliştirdim. Bu proje kapsamında, öncelikle veritabanına bağlanmak için gerekli olan değişken referanslarını oluşturmak üzere SQLAllocHandle fonksiyonlarını kullandım. Bu fonksiyonlar, veritabanı bağlantısını başlatmak için gerekli olan handle'ları (işlem tanıtıcıları) oluşturmamı sağladı.

Sonrasında, veritabanı bağlantısını kurmak için SQLDriverConnect fonksiyonunu kullandım. Bu fonksiyon, verdiğim Connection String aracılığıyla veritabanı ile bağlantı kurmama imkan tanıdı. Bağlantı dizesi, veritabanının konumunu ve erişim bilgilerini içeren bir dizedir ve bu sayede veritabanına erişim sağladım.

Veritabanı üzerinde SQL sorguları çalıştırmak için SQLExecDirect fonksiyonunu kullandım. Bu fonksiyon, SQL komutlarını doğrudan çalıştırmamı sağladı. Bu sayede, Patients tablosundaki kayıtları sorguladım ve elde ettiğim verileri ekrana yazdırdım. Bu işlem, veritabanındaki verileri görüntüleyebilmemi sağladı ve ekleme, silme ve güncelleme gibi temel veritabanı işlemlerini gerçekleştirebildim.

Proje tamamlandıktan sonra, C dilinde başka bir projeden elde ettiğimiz bir DLL (Dynamic Link Library) dosyasının içindeki fonksiyonları kullanmak amacıyla LoadLibrary fonksiyonunu kullandım. LoadLibrary, DLL dosyasını dinamik olarak yüklememe olanak tanıdı ve bu dosyanın içindeki belirli bir fonksiyonu uygulama içerisinde kullanmamı sağladı. Bu işlem, DLL dosyasında bulunan işlevleri başka projelerde de kullanabilmeme olanak tanıdı ve kodun yeniden kullanılabilirliğini artırdı.

```c
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <iostream>

#ifndef DBOPERATIONS_H
#define DBOPERATIONS_H


typedef struct s_patient
{
    SQLINTEGER patient_id;
    SQLCHAR first_name[256];
    SQLCHAR last_name[256];
    SQLCHAR gender[256];
    SQLCHAR phone_number[256];
} t_patient;


void    connectToDatabase(SQLWCHAR* connString, SQLHENV& hEnv, SQLHDBC& hDbc);
void    printPatients(SQLHDBC& hDbc);
void    addPatient(t_patient patient, SQLHDBC& hDbc);

#endif
```

Resim-15: dboperations.h header dosyası

```c
void  connectToDatabase(SQLWCHAR* connString, SQLHENV& hEnv, SQLHDBC& hDbc){
    SQLRETURN ret;
    // Environment Handle oluşturma
    ret = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &hEnv);
    checkError(ret, SQL_HANDLE_ENV, hEnv, "Error allocating environment handle");

    ret = SQLSetEnvAttr(hEnv, SQL_ATTR_ODBC_VERSION, (void*)SQL_OV_ODBC3, 0);
    checkError(ret, SQL_HANDLE_ENV, hEnv, "Error setting ODBC version");

    // Connection başlatma
    ret = SQLAllocHandle(SQL_HANDLE_DBC, hEnv, &hDbc);
    checkError(ret, SQL_HANDLE_DBC, hDbc, "Error allocating connection handle");


    // Bağlanma
    ret = SQLDriverConnect(hDbc, NULL, (SQLWCHAR*)connString, SQL_NTS, NULL, 0, NULL, SQL_DRIVER_NOPROMPT);
    checkError(ret, SQL_HANDLE_DBC, hDbc, "Error connecting to database");
}

void      printPatients(SQLHDBC& hDbc)
{
    SQLRETURN ret;
    // Statement handle, queryler için
    SQLHSTMT hStmt = NULL;

    SQLWCHAR sqlQuery[] = L"SELECT * FROM Patients";

    //Statement handle olusturma
    ret = SQLAllocHandle(SQL_HANDLE_STMT, hDbc, &hStmt);
    checkError(ret, SQL_HANDLE_STMT, hStmt, "Error allocating statement handle");

    ret = SQLExecDirect(hStmt, (SQLWCHAR*)sqlQuery, SQL_NTS);
    checkError(ret, SQL_HANDLE_STMT, hStmt, "Error executing SQL query");
```

Resim-16:Header İmplementasyonu

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:17/07/2024

Yapılan iş: Scrum Excellerinin yüklenebilmesi için WinForm projesi.

Yöneticim tarafından verilen eğitim programını tamamladıktan sonra, Banksoft şirketinde günlük olarak çalışanların hangi Workflow üzerinde ne kadar efor sarf ettiğini gösteren Scrum Excel dosyalarının veritabanına kaydedilmesi ve daha sonra bir web sitesi üzerinden rapor halinde görüntülenmesi görevini aldım.
Projenin işleyişini tasarladıktan sonra, ilk adım olarak SQL Server'da ehScrums adında bir tablo oluşturdum. Bu tablo, Excel dosyalarındaki verilerin saklanacağı temel yapıyı sağladı.
Sonrasında, bir Windows Form projesi geliştirerek, Excel dosyalarını yükleyip içindeki verileri ehScrums tablosuna aktarmak için bir uygulama yaptım. Bu uygulamada, OpenFileDialog bileşenini kullanarak kullanıcıların Excel dosyasını seçmesini sağladım. Seçilen dosyanın içindeki satırları okuyup, uygun biçimde ehScrums tablosuna aktardım. Bu sayede, Excel dosyasındaki verilerin veritabanında düzenli bir şekilde saklanmasını ve ileride web sitesi üzerinden rapor olarak görüntülenebilmesini sağladım.

```
if exists (select 1 from sys.tables where name = 'ehScrums')
    drop table ehScrums
go
create table ehScrums (
    ScrumID int identity(1,1) PRIMARY KEY,
    TeamCode varchar(255),
    Filename  varchar(MAX),
    ScrumStartDate date,
    WFNo int,
    Bank varchar(MAX),
    Subject varchar(MAX),
    Employee varchar(MAX),
    Description varchar(MAX),
    Priority decimal(8,2),
    Status varchar(MAX),
    Start decimal(8,2),
    Completed decimal(8,2),
    Date date,
    InsertDate datetime
)
```

Resim-17: ehScrums tablosunu oluşturma

```
private void btnChoose_Click(object sender, EventArgs e)
{
    openFileDialog = new OpenFileDialog();

    openFileDialog.Filter = "Excel Files (*.xls;*.xlsx)|*.xls;*.xlsx|All Files (*.*)|*.*";

    openFileDialog.Multiselect = false;

    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        string selectedFileName = Path.GetFileName(openFileDialog.FileName);
        lblFilename.Text = selectedFileName;
        UpdateStatus(true);
    }
}

private void btnInsert_Click(object sender, EventArgs e)
{
    try
    {
        // TODO: Dosyanın excel dosyası olup olmadığını kontrol et
        string selectedFilePath = openFileDialog.FileName;
        scrumManager.InsertScrumsFromExcelFile(selectedFilePath);
        UpdateStatus(false);
        MessageBox.Show("Kayıtlar başarıyla eklendi!");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
}
```

Resim-18:Dosya seçme durumunda veritabanına yükleyen kod

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:18/07/2024

Yapılan iş: ScrumProcessTask projesinin yazılması

Ertesi gün projeyi yöneticime gösterdikten sonra, projede bazı değişiklikler yapma kararı aldık. Bu değişiklikler doğrultusunda, veritabanına kayıt işlemlerinin bir Windows Form uygulaması yerine, belirli bir klasör altındaki tüm Excel dosyalarını tarayan ve bu dosyaları işleyen bir Taskman uygulaması olarak yapılması gerektiğine karar verdik. Yeni iş akışı, Excel dosyalarını bulup veritabanına kaydettikten sonra, başarılı işlemler için dosyaları "OKUNDU" adlı bir klasöre, hata ile karşılaşılan dosyalar için ise "HATA" adlı bir klasöre taşımayı içeriyordu.

Bu yeni gereksinimleri karşılamak için projeyi güncellemeye başladım. İlk olarak, System.IO kütüphanesini kullanarak, belirtilen klasör altındaki Excel dosyalarını arayıp bulmak için bir fonksiyon geliştirdim. Bu fonksiyonun adı GetExcelFilesUnderDirectory oldu. Fonksiyonun amacı, belirtilen dizin içindeki tüm Excel dosyalarını taramak ve bu dosyaları bir listeye ekleyip geri döndürmekti.

Fonksiyon, .xls ve .xlsx uzantılı dosyaları aramak için iki farklı dosya uzantısını arıyor. Ayrıca, geçici dosyaları (genellikle Excel'in açılışında oluşturulan ve "~$" ile başlayan dosyalar) hariç tutmak için bir kontrol ekledim. Bulunan dosyaları bir HashSet içinde topladım; HashSet'i dosya isimlerinde büyük-küçük harf farkını göz ardı ederek, dosya isimlerinde çakışma yaşanmasını engellemek ve sadece benzersiz dosyaları listede tutmak için kullandım.

```
public class FileManager
{

    public List<string> GetExcelFilesUnderDirectory(string dirPath)
    {
        string[] excelExtensions = { "*.xls", "*.xlsx" };
        // çakışan dosyaları silmek için
        HashSet<string> excelFiles = new HashSet<string>(StringComparer.OrdinalIgnoreCase);

        foreach (string ext in excelExtensions)
        {
            var files = Directory.GetFiles(dirPath, ext, SearchOption.TopDirectoryOnly);

            if (files != null)
            {
                foreach (string file in files)
                {
                    // Geçici dosyaları alma
                    if (!Path.GetFileName(file).StartsWith("~$"))
                    {
                        Logger.Log2TraceFile("Dosya bulundu: " + Path.GetFileName(file));
                        excelFiles.Add(file);
                    }
                }
            }
        }
        return new List<string>(excelFiles);
    }
```

Resim-19:GetExcelFilesUnderDirectory fonksiyonu

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:19/07/2024

Yapılan iş: Excel tablosundaki kayıtların veritabanına aktarılması

Yaptığım projede, dosya listesinden dönen her dosya yolunu işlemek üzere bir foreach döngüsü kullanarak, her dosya adını InsertScrumsFromExcelFile fonksiyonuna gönderdim. Bu fonksiyon, Interop.Excel kütüphanesini kullanarak belirttiğim dosya üzerinden bir Excel uygulaması açtı ve bu uygulama içinde bulunan tüm Worksheet (çalışma sayfalarını) tek tek işledi.
Fonksiyon, Excel dosyasındaki her çalışma sayfasını bir döngü ile geçerek işledi. Çalışma sayfasındaki satırları, belirli bir satır aralığında veritabanına kaydetti. Veritabanına kayıt işlemi sırasında, harcanan eforu hesaplamak için, önceki gün ile bugünün iş günü arasındaki farkı hesapladım. Bu fark, o gün harcanan eforu belirlemek için kullanıldı ve veritabanına kaydedildi.
Bu işlem, Excel dosyalarındaki verilerin doğru bir şekilde işlenmesini ve veritabanında saklanmasını sağladı. Her dosya için bu işlemi gerçekleştirdikten sonra, veritabanındaki bilgiler güncel kalmış oldu ve raporlama amacıyla kullanılabilir hale geldi.

```
// worksheetleri tek tek okuyarak veritabanına ekle
foreach (Worksheet worksheet in sheets)
{
    if (worksheet.Name == "Grafik" || worksheet.Name == "Chart")
        continue;

    Console.WriteLine(Path.GetFileNameWithoutExtension(filePath) + " -> " + "Processing: " + worksheet.Name);
    Console.WriteLine("---------------------------------");
    try
    {
        this.DeleteIfDateExists(worksheet, scrumConfig, teamCode);
    }
    catch (Exception ex)
    {
        Logger.Log2ErrorFile(string.Format("ScrumProcessTask->ScrumManager->DeleteIfDateExists(): Filename:{0} Sheet:{1} Exception:{2}",
                                           filename, worksheet.Name, ex.ToString()));
        throw ex;
    }

    int rowCount = worksheet.UsedRange.Rows.Count;

    for (int row = scrumConfig.DataStartRowNo; row <= rowCount; row++)
        InsertScrumsToDb(worksheet, scrumConfig, row, filename);
    Console.WriteLine("---------------------------------");
}
```

Resim-20:InsertScrumsFromExcelFile fonksiyonu

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:22/07/2024

Yapılan iş: Scrumlar için web sitesinin oluşturulması.

Veritabanına kaydetme işlemini tamamladıktan sonra, kaydedilmiş verilerin gözlemlenebilmesi için ScrumsWeb adında bir Web Form projesi oluşturdum. Bu projede, kullanıcıların verileri etkili bir şekilde görüntüleyebilmesi ve etkileşimde bulunabilmesi için modern ve şık bir arayüz tasarladım. Arayüz tasarımında, hızla uyumlu ve estetik bir görünüm elde etmek için Bootstrap kütüphanesini kullandım.

Veritabanındaki ehScrums tablosundaki verileri web arayüzünde görüntülemek amacıyla Handsontable adında bir JavaScript kütüphanesi entegrasyonu gerçekleştirdim. Handsontable, kullanıcıların verileri filtrelemesine, sıralamasına ve düzenlemesine olanak tanıyan güçlü bir tablo komponentidir.

Veri görselleştirme işlemi için önce verileri JSON formatına dönüştürdüm. JSON, verilerin yapılandırılmış bir biçimde temsil edilmesini sağlayarak, Handsontable kütüphanesi ile uyumlu hale getirir. JSON stringini Handsontable'ın data özelliğine bağlayarak, verilerin tabloda dinamik ve etkileşimli bir şekilde görüntülenmesini sağladım.

```
function initializeHandsontable(jsonData) {
    var data = jsonData

    var colHeaders = Object.keys(jsonData[0]);

    var columns = colHeaders.map(function (header) {
        var column = { data: header };
        if (header == "Açıklama" || header == "Description")
            column.width = 200;
        return column;
    });

    var container = document.getElementById('gridContainer');

    var hot = new Handsontable(container, {
        data: data,
        colHeaders: colHeaders,
        columns: columns,
        height: 500,
        width: '100%',
        afterRender: function () {
            updateStatistics(this);
        },
        search:true,
        filters: true,
        dropdownMenu: ['alignment', '---------', 'filter_by_condition', 'filter_by_value', 'filter_action_bar'],
        contextMenu: false,
        readOnly: true,
        columnSorting: true,
        hiddenColumns: true,
        stretchH: 'all',
        licenseKey: 'non-commercial-and-evaluation',
    });
```

Resim-21:Handsontable’ın enjekte edilmesi

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:23/07/2024

Yapılan iş: ehScrums tablosunun düzenlenmesi ve ehScrumConfigs tablosunun oluşturulması.

Yaptığım projeyi yöneticime sunduktan sonra, projede önemli bir değişiklik yapma kararı aldık ve yeni bir veritabanı tasarımı oluşturduk. Yöneticim, Banksoft'un bankalarla iş birliğinin nasıl işlediğini ve anlaşmaların sarf edilen efor üzerinden yapıldığını detaylı bir şekilde açıkladı. Bu açıklamalar doğrultusunda, yeni tasarım gereği ehScrums tablosuna EkipKodu adında bir sütun eklemeye karar verdik. Bu sütunun değeri, ilgili Excel dosyasının adındaki ilk kelime olarak belirlenecek.
Ayrıca, Excel dosyasında hangi sütun numarasının hangi veri alanına karşılık geleceğini belirtmek amacıyla ehScrumConfigs adında yeni bir tablo oluşturduk. Bu yeni tablo, her ekip koduna göre Excel dosyasındaki sütunların hangi veri değerlerine karşılık geldiğini tanımlayacak. Bu tasarım sayesinde, Excel dosyasındaki sütunlar ve verilerin karşılığı ekip koduna bağlı olarak dinamik bir şekilde ayarlanabilecek, böylece farklı ekipler için özelleştirilmiş veri yönetimi sağlanmış olacak.

```
BSS...SOFT\Stajyer (724))   scrumtablecreation...SOFT\Stajyer (735))   select2.sql - BSS02...SOFT
if exists (select 1 from sys.tables where name = 'ehScrums')
    drop table ehScrums
go
create table ehScrums (
    ScrumID int identity(1,1) PRIMARY KEY,
    TeamCode char(20),
    Filename  char(100),
    ScrumStartDate date,
    WFNo int,
    BankCode char(20),
    Subject char(255),
    EmployeeCode char(20),
    Description char(1020),
    Priority decimal(8,2),
    Status char(255),
    Start decimal(8,2),
    Completed decimal(8,2),
    WorkDate date   data type decimal(8, 2)
    InsertDate datetime
)
```

Resim-22:ehScrums tablosunun güncellenmesi

```
figs.sql...SOFT\Stajyer (742))   ehScrums.sql - BSS...SOFT\Stajyer (724))   scrumtablecreation...SOFT\Stajyer (
if exists (select 1 from sys.tables where name = 'ehScrumConfigs')
    drop table ehScrumConfigs
go
create table ehScrumConfigs (
    ScrumConfigID int identity(1,1) PRIMARY KEY,
    TeamCode char(20) not null,
    WFNoColNo int not null,
    BankColNo int not null,
    SubjectColNo int not null,
    EmployeeColNo int not null,
    DescriptionColNo int not null,
    PriorityColNo int not null,
    StatusColNo int not null,
    StartColNo int not null,
    DateStartColNo int not null,
    DataStartRowNo int not null,
    InsertDate datetime
)
go
```

Resim-23:ehScrumConfigs tablosunun oluşturulması

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:24/07/2024

Yapılan iş: Excel'de sondan başa doğru okumak için gerekli değişiklikleri yaptım ve hataları düzelttim

Bir önceki gün yapılan değişikliklerin üstüne son olarak excel sheetlerinin geçmiştün günümüze doğru okunması ve daha önceden aynı güne ait bir kayıt bulunması halinde bulunan kayıtların silinip yerine güncel kayıtların kaydedilmesi değişikliğine gittik. Bunların üstüne yazdığım kodların algoritmasını yaptığım değişikliklere göre değiştirdim ve veritabanına kayıt işlemini tamamladım ve test aşamasına geçtim. Test aşamasında birçok hata ile karşılaştım bazı tarihlerde white space olduğu için tarih olarak görülmemesi veya tarihin yanına TATİL yazıldığı için programın kayıtların tamamını düzgün kaydetmemesi gibi ve bu hatalı GetExcelDate HandleDoubleValues gibi fonksiyonlar yazarak excel verilerinin işlenirken hata vermemesini sağladım ve günümün geri kalanında bunun gibi diğer hataları düzelttim.

```
// Sheets koleksiyonunu tersine çevir
List<Worksheet> sheets = new List<Worksheet>();
foreach (Worksheet sheet in workbook.Worksheets)
    sheets.Add(sheet);
sheets.Reverse();

// worksheetleri tek tek okuyarak veritabanına ekle
foreach (Worksheet worksheet in sheets)
{
    if (worksheet.Name == "Grafik" || worksheet.Name == "Chart")
        continue;

    Console.WriteLine(Path.GetFileNameWithoutExtension(filePath) + " -> " + "Processing: " + workshe
    Console.WriteLine("---------------------------------");
    try
    {
        this.DeleteIfDateExists(worksheet, scrumConfig, teamCode);
    }
    catch (Exception ex)
    {
        Logger.Log2ErrorFile(string.Format("ScrumProcessTask->ScrumManager->DeleteIfDateExists(): Fi
                                           filename, worksheet.Name, ex.ToString()));
        throw ex;
    }

    int rowCount = worksheet.UsedRange.Rows.Count;

    for (int row = scrumConfig.DataStartRowNo; row <= rowCount; row++)
        InsertScrumsToDb(worksheet, scrumConfig, row, filename);
    Console.WriteLine("---------------------------------");
}
```

Resim-24:Excel sondan başa doğru çevrilmesi ve okunması

```
public static DateTime? GetExcelDate(dynamic col)
{
    DateTime? date = null;
    try
    {
        date = (DateTime)col;
    }
    catch (Exception ex)
    {
        try
        {
            string dateStr = Regex.Replace((string)col, @"\s+", "");
            date = Convert.ToDateTime(dateStr);
        }
        catch (Exception ex2)
        {
            date = null;
        }

    }
    return date;
}
```

Resim-25:Tarih alma kısmındaki hataların arındırılması için yazılan fonksiyon

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:25/07/2024

Yapılan iş: AJAX istekleri için Generic Handler dosyalarının yazılması

Projeye, veritabanındaki ehScrumConfigs tablosunu görüntülemek ve yönetmek amacıyla bir ScrumConfigs.aspx sayfası ekledim. Bu sayfada, kullanıcıların tabloyu dinamik olarak görüp düzenleyebilmesi için Handsontable kütüphanesini kullandım. Handsontable, kullanıcıların verileri kolayca düzenleyebilmesini sağlayan güçlü bir tablo komponentidir.
Handsontable'ın sunduğu afterChange, afterInsertRow, ve beforeRemoveRow gibi hook fonksiyonlarını kullanarak, tablo üzerindeki veri değişikliklerini anlık olarak yakaladım. Bu sayede, kullanıcıların tablo üzerinde yaptığı ekleme, silme ve güncelleme işlemleri hemen veritabanına yansıdı.
Veri işlemlerini asenkron bir şekilde gerçekleştirmek için jQuery'nin AJAX fonksiyonlarını kullandım. AJAX, web sayfasını yeniden yüklemeden veri gönderip almayı sağlar, bu da kullanıcı deneyimini daha akıcı ve hızlı hale getirir. AJAX isteklerini işlemek için Generic Handler (.ashx) dosyalarını kullandım. Bu handler dosyaları, gelen AJAX isteklerini karşılayarak, kullanıcı tarafından talep edilen işlemi (silme, güncelleme, ekleme) yerine getirdi. Bu yapı, veritabanı ile olan etkileşimi dinamik ve verimli bir şekilde yönetmeme olanak sağladı.

```
public class InsertScrumConfigHandler : IHttpHandler
{

    public void ProcessRequest(HttpContext context)
    {
        context.Response.ContentType = "application/json";
        try
        {
            string jsonString;
            using (var reader = new StreamReader(context.Request.InputStream))
            {
                jsonString = reader.ReadToEnd();
            }
            ScrumsShared.Entities.ScrumConfig[] configs = JsonConvert.DeserializeObject<ScrumsShared.Entities.ScrumConfig[]>(jsonString);

            ScrumConfigManager configManager = new ScrumConfigManager();
            foreach (var config in configs)
            {
                configManager.InsertScrumConfig(config);
            }
            context.Response.StatusCode = 200;
            context.Response.Write("{\"status\":\"success\"}");
        }
        catch (Exception ex)
        {
            context.Response.StatusCode = 500;
            context.Response.Write("{\"status\":\"error\", \"message\":\"" + ex.Message + "\"}");
        }
    }

    public bool IsReusable
    {
        get { return false; }
    }
}
```

Resim-26:InsertScrumConfigHandler dosyası

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:26/07/2024

Yapılan iş: İlgililer ve Bankalar tablolarının oluşturulması

Banksoft şirketinin iş birliği yaptığı bankalarla olan süreçlerini ve nasıl çalıştığını öğrendikten sonra, proje üzerinde önemli bir değişiklik yapma kararı aldık. Bu değişiklikler doğrultusunda, veritabanını genişletmek ve raporlama yeteneklerini artırmak amacıyla iki yeni tablo eklemeye karar verdik: ehEmployees ve ehBanks.

İlk olarak, ehEmployees tablosu şirketin çalışan bilgilerini ve ehBanks tablosu ise bankalarla ilgili bilgileri saklayacak şekilde tasarladım. Bu tabloları, mevcut ehScrums tablosuyla INNER JOIN ederek raporlama işlemlerinde kullanılmak üzere oluşturdum. Bu INNER JOIN, raporlama sırasında her çalışanın ve bankanın ilgili Scrum verileriyle ilişkilendirilmesini sağlayacak.

Bu iki yeni tabloyu veritabanına ekledikten sonra, ASP.NET Web Form projesinde her biri için ayrı birer sayfa oluşturdum. Bu sayfaların her birini, ilgili tablodaki verilerin yönetilmesini sağlamak üzere tasarladım.

```
if exists (select 1 from sys.tables where name = 'ehBanks')
    drop table ehBanks
go

CREATE TABLE ehBanks (
    BankID INT PRIMARY KEY IDENTITY(1,1),
    BankCode char(20) NOT NULL,
    BankName char(255)
);
```

Resim-27: ehBanks tablosu

```
if exists (select 1 from sys.tables where name = 'ehEmployees')
    drop table ehEmployees
go

CREATE TABLE ehEmployees (
    EmployeeID INT PRIMARY KEY IDENTITY(1,1),
    EmployeeCode char(20) NOT NULL,
    EmployeeName char(100),
    EmployeeSurname char(100)
);
```

Resim-28: ehEmployees tablosu

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:29/07/2024

Yapılan iş: WF Süre Toplamları ve WF İlgili Toplamları raporlarının oluşturulması

Önceden planlanmış olan WF Süre Toplamları ve WF İlgili Toplamları raporları için gerekli SQL sorgularını hazırladım. Bu sorguları, dört ana filtre kullanarak ehScrums tablosundaki kayıtları daraltacak şekilde tasarladım: Başlangıç Tarihi, Bitiş Tarihi, Banka Kodu ve İlgili Kodu. Filtreleme işlemi, kullanıcıların raporlarda belirli bir zaman dilimi, banka ve ilgili kodlarına göre veri elde etmelerini sağlıyor.

Sorguları hazırladıktan sonra, bu sorgulara C# uygulamasından erişebilmek için iki ayrı Stored Procedure oluşturdum: rpsp_GetWfTimeReports ve rpsp_GetWfEmployeeReports. Bu Stored Procedure'ler, veritabanında tanımladığımız sorguları çalıştırarak gerekli rapor verilerini sağlıyor.

Sonrasında, Web Form projesindeki ScrumManager sınıfına iki fonksiyon ekleyerek bu prosedürleri çalıştırdım ve elde edilen verileri aldım. ScrumManager sınıfındaki fonksiyonlar, ilgili Stored Procedure'leri çağırarak veri çekme işlemini otomatikleştiriyor.

Elde edilen verileri, kullanıcı dostu bir arayüzde gösterebilmek için Handsontable kütüphanesini kullandım. Handsontable, verilerin filtrelenmesi ve sıralanması gibi işlevlerle kullanıcıya etkileşimli bir tablo sunuyor. Verileri JSON formatına dönüştürdükten sonra, Handsontable'ın data özelliğine bağlayarak tablo üzerinde gösterilmesini sağladım. Bu, verilerin dinamik bir şekilde görüntülenmesini ve kullanıcıların raporları daha etkili bir şekilde analiz etmelerini sağladı.

```
go
create PROC rpsp_GetWfEmployeeTimeReports
    @TarihBasla char(8) = '',       --YYYYAAGG, Bos: Hepsi
    @TarihBitir char(8) = '',       --YYYYAAGG, Bos: Hepsi
    @BankaKodu  char(20) = '',      --Bos: Hepsi
    @Ilgili     char(20) = ''       --Bos: Hepsi
AS
BEGIN
    SELECT
        WFNo 'WF No',
        ehScrums.EmployeeCode 'İlgili',
        SUM(Completed) 'Tamamlanan',
        MIN([WorkDate]) 'Başlangıç Tarihi',
        MAX([WorkDate]) 'Bitiş Tarihi'
    FROM
        ehScrums
    INNER JOIN ehEmployees ON
            ehScrums.EmployeeCode = ehEmployees.EmployeeCode
    INNER JOIN  ehBanks ON ehScrums.BankCode = ehBanks.BankCode

    WHERE WFNo IS NOT NULL

    and (@TarihBasla = '' or [WorkDate] >= @TarihBasla)
    and (@TarihBitir = '' or [WorkDate] <= @TarihBitir)
    and (@BankaKodu = '' or @BankaKodu = ehScrums.BankCode)
    and (@Ilgili = '' or @Ilgili = ehScrums.EmployeeCode)

    GROUP BY
        WFNo,
        ehScrums.EmployeeCode
    ORDER BY ehScrums.EmployeeCode,MIN([WorkDate])
END
```

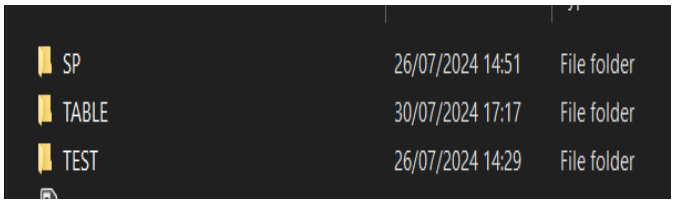
Resim-29: WF İlgili Toplamları rapor SP'si

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:30/07/2024

Yapılan iş: SQL scriptlerinin klasörlenmiş şekilde saklanması ve yeniden isimlendirmesi

SQL scriptlerini daha düzenli ve yönetilebilir hale getirmek için yaptığım bazı iyileştirmeler yaptım bunlar: Tüm tablo oluşturma ve stored procedure oluşturma kodlarını ayrı dosyalara yazdım ve bunları uygun klasörlere ayırdım. Stored procedure isimlerinin ön eklerini "scsp_" olarak düzenleyerek isimlendirme standardını sağladım. Ayrıca, bellek kullanımını optimize etmek amacıyla veritabanındaki veri tiplerini ALTER TABLE CONSTRAINT COLUMN kullanarak güncelledim; örneğin, varchar(MAX) yerine daha uygun bir karakter limiti olan char(20) gibi veri tiplerini kullandım.Bu değişiklikleri gerçekleştirdikten sonra, daha önce yazmış olduğum tüm projelerde bu yeni düzenlemelere uyum sağlamak için mevcut kodları gözden geçirdim ve gerekli güncellemeleri yaptım. Bu süreç, projelerimin daha performanslı ve düzenli olmasını sağladı.



Resim-30: SP'lerin saklandığı klasörler

Resim-31: SP klasörünün içeriği

```
if exists (select 1 from sys.tables where name = 'ehEmployees'
    drop table ehEmployees
go

CREATE TABLE ehEmployees (
    EmployeeID INT PRIMARY KEY IDENTITY(1,1),
    EmployeeCode char(20) NOT NULL,
    EmployeeName char(100),
    EmployeeSurname char(100)
);
```

Resim-32:Tabloların alanlarının uygun karakter limitleriyle değiştirilmesi

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:31/07/2024

Yapılan iş: Raporların filtrelenmesi ve web sitesinde listelenmesi

Raporlama işlevselliğini geliştirmek için WFTimeReports.aspx, WFEmployeeReports.aspx ve Default.aspx sayfalarına veri filtreleme özelliği ekledim. Her bir sayfada, kullanıcıların verileri daha detaylı bir şekilde inceleyebilmeleri için dört adet input alanı yer aldı: İlgili Kodu, Banka Kodu, Başlangıç Tarihi ve Bitiş Tarihi. Bu filtreler, kullanıcının raporları belirli kriterlere göre filtreleyebilmesini sağlıyor.
Kullanıcı, filtreleme butonuna bastığında, bu inputlardan alınan değerler dinamik olarak veritabanındaki üç farklı stored procedure'e parametre olarak gönderiliyor. Bu prosedürler, scsp_GetWfTimeReports, scsp_GetWfEmployeeReports ve scsp_GetAllScrumsFiltered isimleriyle tanımlanan işlemler, sırasıyla belirli zaman dilimindeki toplam süreleri, çalışan bazında verileri ve belirli kriterlere göre filtrelenmiş scrum kayıtlarını döndürüyor.
Stored procedure'ler, gelen parametrelere göre verileri filtreleyip sorgulama işlemini gerçekleştiriyor. Filtreleme işlemi tamamlandıktan sonra elde edilen veriler, ilgili rapor sayfalarına bağlanarak Handsontable kütüphanesi aracılığıyla tablo şeklinde sunuluyor. Bu yapı, kullanıcıların raporları daha anlamlı ve ihtiyaçlarına göre özelleştirilmiş bir biçimde görüntülemelerini sağladı, böylece raporların doğruluğu ve kullanılabilirliği artırıldı.

```
protected void btnFilter_Click(object sender, EventArgs e)
{
    try
    {
        string formattedStartDate = "";
        string formattedEndDate = "";
        string empCode = ddlEmployees.SelectedValue;
        string bankCode = ddlBanks.SelectedValue;
        if (!string.IsNullOrEmpty(txtStartDate.Text))
        {
            DateTime startDate = Convert.ToDateTime(txtStartDate.Text);
            formattedStartDate = startDate.ToString("yyyyMMdd");
        }

        if (!string.IsNullOrEmpty(txtEndDate.Text))
        {
            DateTime endDate = Convert.ToDateTime(txtEndDate.Text);
            formattedEndDate = endDate.ToString("yyyyMMdd");
        }
        LoadGridView(empCode, bankCode, formattedStartDate, formattedEndDate);
    }
    catch (Exception ex)
    {
        string script = string.Format("toastr.error('{0}');", ex.Message);
        toastrErrScript.Text = string.Format("<script type=\"text/javascript\">{0}</script>", script);
    }

}
```

Resim-33: Filtreleme işleminin kodları

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:01/08/2024

Yapılan iş: Rapor sayfalarında sütunların dinamik olarak alınması ve Web Form'ların klasörlenmesi

Web projesindeki raporlar bölümünde, tabloların gösteriminde stored procedure'lerdeki sütun değişikliklerinin anlık olarak tabloya yansıması için bazı kod değişiklikleri yaptım. Özellikle, JavaScript'teki Object.keys fonksiyonunu kullanarak gelen JSON verisinin sütun isimlerini bir dizi olarak aldım ve bu diziyi colHeaders özelliğine atadım. Bu yöntemle, sütun isimlerinin dinamik olarak güncellenmesini sağladım, böylece stored procedure'lerde yapılan değişiklikler tabloya anında yansıdı. Ayrıca, projenin düzenini iyileştirmek amacıyla, tüm .aspx dosyalarını aynı klasörde toplamak yerine, WebForms adında bir klasör oluşturdum. Bu klasörü Reports ve Parameters olmak üzere iki alt klasöre böldüm ve ilgili .aspx dosyalarını bu alt klasörlere yerleştirdim. Dosyalar arasındaki referansları güncellemek için ResolveUrl fonksiyonunu kullanarak doğru path düzenlemesi yaptım.

```
var data = jsonData

var colHeaders = Object.keys(jsonData[0]);

var columns = colHeaders.map(function (header) {
    var column = { data: header };
    if (header == "Açıklama" || header == "Description")
        column.width = 200;
    return column;
});

var container = document.getElementById('gridContainer');

var hot = new Handsontable(container, {
    data: data,
    colHeaders: colHeaders,
    columns: columns,
```

Resim-34: Sütunların dinamik olarak alınması

```
<link rel="stylesheet" href="<%= ResolveUrl("~/Content/Lib/bootstrap.min.css") %>">
<link rel="stylesheet" href="<%= ResolveUrl("~/Content/Lib/bootstrap-select.min.css") %>">
<link rel="stylesheet" href="<%= ResolveUrl("~/Content/Lib/handsontable.full.min.css") %>">
<script src="<%= ResolveUrl("~/Scripts/Lib/handsontable.full.min.js") %>"></script>
<script src="<%= ResolveUrl("~/Scripts/Lib/filesaver.min.js") %>"></script>
<script src="<%= ResolveUrl("~/Scripts/Lib/xlsx.full.min.js") %>"></script>
<link rel="stylesheet" href="<%= ResolveUrl("~/Content/Lib/toastr.min.css") %>">
<asp:ContentPlaceHolder runat="server" ID="HeadScriptsContent" />
```

Resim-35: ResolveUrl kullanımı

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:02/08/2024

Yapılan iş: Tamamlanan projenin Teknik Tasarım dökümanının hazırlanması

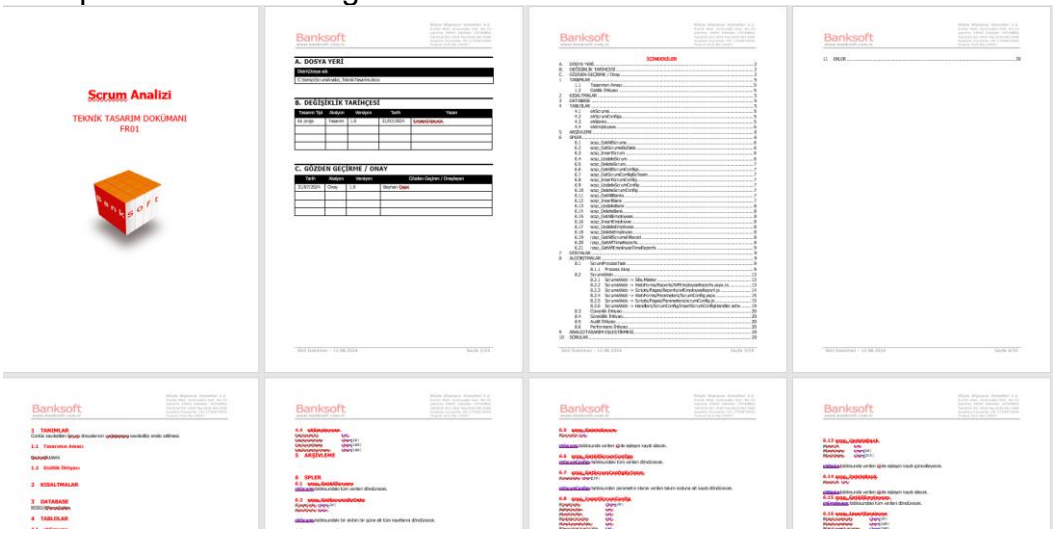
Scrum Analizi projesi için hazırladığım Teknik Tasarım dökümanına başlarken, öncelikle projenin amacını net bir şekilde belirttim. Bu bölümde, projenin neden geliştirildiğini, hangi problemleri çözmeyi hedeflediğini ve elde edilen sonuçların ne gibi faydalar sağlayacağını ayrıntılı bir şekilde açıkladım. Ardından, projede kullanılan veritabanı tablolarını detaylı bir şekilde anlattım. Bu tabloların her birinin hangi verileri barındırdığını, bu verilerin nasıl kullanılacağını ve projedeki diğer bileşenlerle nasıl etkileşime girdiğini örneklerle açıkladım.

Oluşturduğum stored procedure'lerin her birini ele alarak, bunların hangi işlemleri gerçekleştirdiğini, hangi parametreleri aldığını ve sonuçta hangi verileri döndürdüğünü teknik detaylarıyla beraber dökümanda belirttim. Bu sayede, bu stored procedure'leri kullanacak olan kişiler için kapsamlı bir referans noktası oluşturmuş oldum.

Proje boyunca yazdığım iki ayrı uygulamanın en kritik kod parçacıklarını dökümanda ayrı başlıklar altında topladım. Bu kod parçacıklarının işlevlerini açıklayarak, bu kodların projenin genel işleyişi içinde nasıl bir rol oynadığını detaylandırdım. Böylece, dökümanı inceleyenlerin kodun amacını ve nasıl çalıştığını net bir şekilde anlamasını sağladım.

Dökümanın içinde yer alan başlıkları ve alt başlıkları düzenli bir şekilde yapılandırarak, dökümanın başında yer alan Navigasyon Tablosunun doğru ve etkin bir şekilde çalışmasını sağladım. Bu, dökümanın farklı bölümleri arasında hızlı ve kolay bir şekilde gezinebilme imkanı tanıyarak, okuyucuların ihtiyaç duydukları bilgilere hızlıca ulaşabilmesini mümkün kıldı.

Teknik Tasarım dökümanını tamamladıktan sonra, yöneticim benden bu projeyi bir Zoom toplantısında sunmamı ve aynı zamanda ekran kaydı almamı istedi. Toplantıyı organize ederek, projeyi ayrıntılı bir şekilde sundum. Sunum sırasında, projenin teknik detaylarına dair soruları yanıtladım ve projeyle ilgilenecek olan diğer kişilerin projeyi daha iyi anlayabilmeleri için gerekli bilgileri verdim. Bu sunum ve döküman sayesinde, benden sonra projeyi devralacak kişilerin projeye hızlı bir şekilde adapte olabilmesini sağladım.



Resim-36: Teknik tasarım dökümanının ilk sayfaları

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:05/08/2024

Yapılan iş: Banksoft web sitesinin daha modern şekilde yeniden oluşturulmasına başlanması

Banksoft'un mevcut web sitesinin eski ve günümüz ihtiyaçlarına uygun olmaması nedeniyle, sitenin yeniden tasarlanmasına karar verildi. Bu proje için öncelikle uygun bir tema seçimi yapmam gerekiyordu. Modern ve kullanıcı dostu bir arayüz sunmak amacıyla, birçok farklı tema arasından araştırma yaptım ve sonunda https://themewagon.com/ adresinden ücretsiz bir Bootstrap teması seçmeye karar verdim. Temayı seçtikten sonra, temanın tüm kaynak kodlarını bilgisayarıma indirdim. Ardından, Banksoft için oluşturduğum ASP.NET Web Form projesine bu temayı entegre etmek için çalışmalara başladım. İlk olarak, projenin temel yapı taşı olan Site.master dosyasına odaklandım. Site.master, projenin tüm sayfalarına uygulanan genel layout'u kontrol eden dosya olduğu için, temanın HTML kodlarını ve CSS stil dosyalarını bu dosyaya ekleyerek projenin tüm sayfalarına şık bir görünüm kazandırdım.

Bu entegrasyon süreci, temanın HTML yapısını ASP.NET Web Forms ile uyumlu hale getirmek için bazı ince ayarları yapmamı gerektirdi. Örneğin, temadaki navigasyon menüsünü, footer alanını ve diğer kritik tasarım kısımlarını, dinamik olarak içeriklerin yüklendiği alanlar haline getirdim. Bu, sitenin kullanıcı deneyimini geliştirmek için gerekliydi.

CSS ve JavaScript dosyalarını doğru şekilde projeye dahil ettikten sonra, temanın tüm bileşenlerinin sorunsuz bir şekilde çalıştığından emin olmak için kapsamlı testler yaptım. Bu testleri, sitenin farklı tarayıcılarda nasıl göründüğünü, mobil cihazlarda nasıl davrandığını ve kullanıcı etkileşimlerini nasıl yönettiğini görmek için yaptım.

```
ter.cs   CardSolutions.aspx.cs   bootstrap.min.css   Company.aspx   Default.aspx   CardSolutions.aspx   Site.Master
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Banksoft</title>
    <meta content="width=device-width, initial-scale=1.0" name="viewport">
    <meta content="" name="keywords">
    <meta content="" name="description">

    <!-- Favicon -->
    <link href="/Content/img/favicon.ico" rel="icon">

    <!-- Google Web Fonts -->
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Open+Sans:wght@400;600&family=Ubuntu:wght@500;700&display=swap"
        rel="stylesheet">

    <!-- Icon Font Stylesheet -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.10.0/css/all.min.css" rel="stylesheet">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.4.1/font/bootstrap-icons.css" rel="stylesheet">

    <!-- Libraries Stylesheet -->
    <link href="/Scripts/lib/animate/animate.min.css" rel="stylesheet">
    <link href="/Scripts/lib/owlcarousel/assets/owl.carousel.min.css" rel="stylesheet">
    <script src="https://unpkg.com/typeit@8.8.4/dist/index.umd.js"></script>
    <asp:ContentPlaceHolder runat="server" ID="Head"></asp:ContentPlaceHolder>
    <!-- Customized Bootstrap Stylesheet -->
    <link href="/Content/css/bootstrap.min.css" rel="stylesheet">

    <!-- Template Stylesheet -->
    <link href="/Content/css/style.css" rel="stylesheet">
    <link href="Content/css/collapsible.css" rel="stylesheet" />
    <script src="/Scripts/js/collapsible.js"></script>
</head>
```

Resim-37: Kaynak kodların projenin layout kısmına koyulması

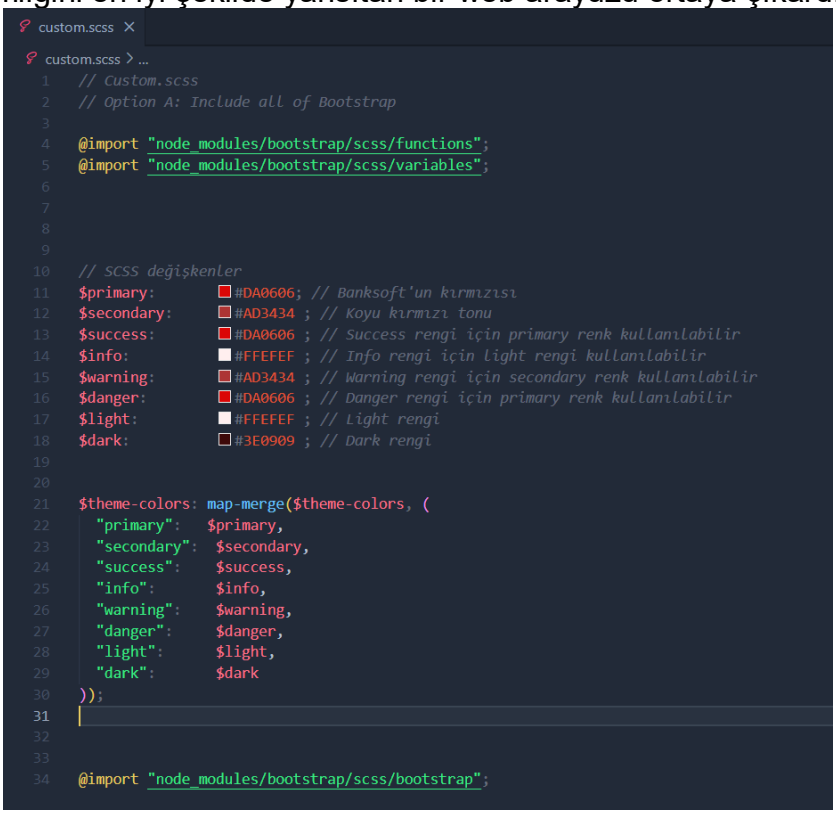
Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:06/08/2024

Yapılan iş: Web sitenin özelleştirilmesi

Öncelikle, Banksoft'un mevcut web sitesinin eski görünümünü yenilemek amacıyla, seçtiğim Bootstrap teması üzerinde çalışmalara başladım. Temanın ana rengi mavi olduğu için, bu renk Banksoft'un kurumsal renkleriyle uyuşmuyordu. Bu nedenle, temanın renk paletini değiştirmek için öncelikle Node.js'i bilgisayarıma kurdum. Node.js sayesinde, gerekli olan "bootstrap@5.0.0" ve "sass" kütüphanelerini indirdim. Bu kütüphaneler, Bootstrap'in SCSS dosyalarını düzenlememi ve derlememi sağladı.

SCSS dosyaları üzerinden temanın renk ayarlarına ulaştım ve mavi olan ana rengi Banksoft'un kırmızı kurumsal rengine uygun olacak şekilde değiştirdim. Renk değişikliklerini tamamladıktan sonra, Node.js üzerinde SCSS dosyalarını derleyerek yeni bir "bootstrap.css" dosyası oluşturdum. Bu CSS dosyası, artık Banksoft'un kurumsal kimliğine daha uygun olan kırmızı tonlarında bir tema sunuyordu. Bu süreçte, renk değişikliklerinin yanı sıra, temanın genel estetiğini ve uyumunu da gözden geçirerek Banksoft'un marka kimliğini en iyi şekilde yansıtan bir web arayüzü ortaya çıkardım.



Resim-38: Tema renklerinin değiştirilmesi

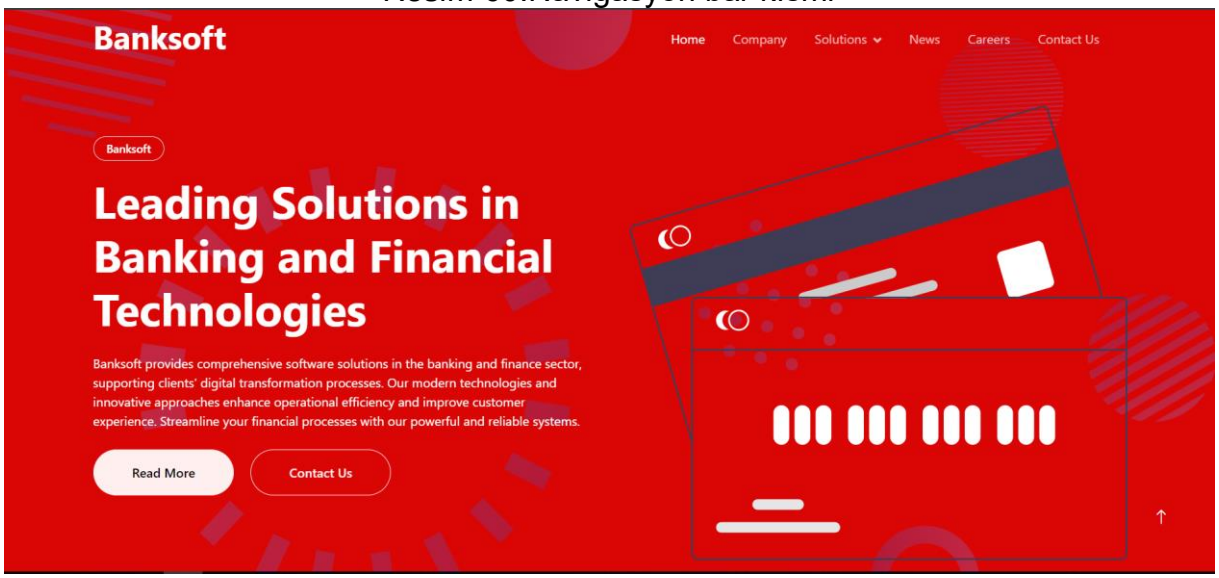
Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:07/08/2024

Yapılan iş: Yeni oluşturulan web sitenin navigasyon barının ve içeriğinin düzenlenmesi.

Renk temasını değiştirdikten sonra, navigasyon barını Banksoft'un kendi sitesindeki navigasyon barı ile aynı hale getirdim. Navigasyon barında menü seçeneklerini modern bir görünüm kazandıracak şekilde yeniden tasarladım. Menüde, kullanıcıların farklı raporlar ve parametreler arasında kolayca gezinebilmeleri için açılır menüler (dropdowns) ekledim. Ayrıca, navigasyon barının mobil uyumlu olması için bir açılır menü butonu da ekledim, böylece küçük ekranlarda menü öğeleri düzgün bir şekilde görüntülenebilir hale geldi. Menü öğelerinin her birini, Banksoft web sitesinin kullanıcı deneyimini iyileştirecek şekilde tasarladım, böylece kullanıcılar istedikleri sayfalara hızla ve kolayca erişebiliyor. Navigasyon barının tamamlanmasının ardından, sayfanın genel estetiğini iyileştirmek için uygun illüstrasyonlar ve modern içerik düzenlemeleri yaparak, web sitesinin daha şık ve profesyonel görünmesini sağladım.

Resim-60:Navigasyon bar kısmı



Resim-39: Ana sayfanın görüntüsü

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:08/08/2024

Yapılan iş: Web sitelerinin “subnavigation” kısmı için açılıp kapanabilir element eklemek.

Banksoft’un web sitesinde çok sayıda iç içe navigasyon işlemi olduğundan, bu durumu yönetilebilir hale getirmek için açılıp kapanabilir (collapsible) HTML elementleri oluşturdum. Bu yapıyı desteklemek amacıyla, CSS kodu kullanarak collapsible içeriklerin nasıl görüneceğini tanımladım.
Örneğin, .collapsible-content sınıfıyla, içeriğin başlangıçta görünmemesini, taşma durumlarında gizlenmesini ve içeriğe biraz sol dolgu eklenmesini sağladım. Ayrıca, açılır menü öğeleri için Font Awesome ikonları kullandım; bu ikonlar, menü öğelerinin yanına ekleyerek kullanıcıya daha iyi bir görsel geri bildirim sağlamak amacıyla eklendi.
.btn-collapsible sınıfı ile, açılır menü butonlarına ikonlar eklenirken, .active-collapse sınıfı ile aktif durumdaki öğelerin yazı rengi beyaz olarak ayarlandı ve bu durumdaki butonlar için de farklı bir ikon kullanıldı. Bu tasarım sayesinde, kullanıcılar iç içe navigasyonları kolayca açıp kapatabilir hale geldi ve bu sayede web sitesi daha kullanıcı dostu hale gelmiş oldu.

```css
.collapsible-content {
    display: none;
    overflow: hidden;
    padding-left:12px;
}

.collapsible-content .page-nav-item:before {
    border: none;
    content: "\f068";
    font-family: "Font Awesome 5 Free";
    font-weight: 900;
    vertical-align: middle;
    margin-right: 8px;
}

.btn-collapsible:before {
    border: none;
    content: "\f105";
    font-family: "Font Awesome 5 Free";
    font-weight: 900;
    vertical-align: middle;
    margin-right: 8px;
}

.active-collapse {
    color:white;
}

.active-collapse:before {
    border: none;
    content: "\f107";
    font-family: "Font Awesome 5 Free";
    font-weight: 900;
    vertical-align: middle;
    margin-right: 8px;
}

```

Resim-40:”collapsible.css” dosyası

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:09/08/2024

Yapılan iş: Mikroservis mimarisinin araştırılması

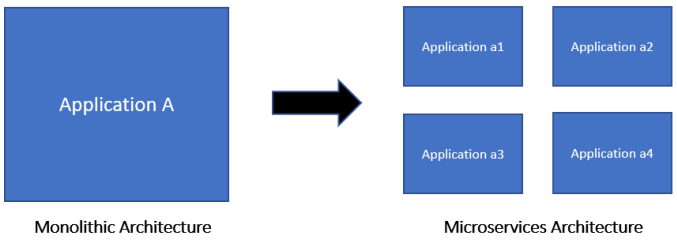
Yöneticimin isteği üzerine mikroservis mimarisi hakkında araştırma yaptım ve mikroservislerin özellikle büyük ve karmaşık sistemlerde nasıl avantajlar sağladığını inceledim. Mikroservislerin, bağımsız dağıtım ve geliştirme yeteneği sayesinde ekiplerin daha esnek ve hızlı çalışabilmesine olanak tanıdığını öğrendim. Ayrıca, farklı programlama dilleri ve teknolojiler kullanılarak geliştirilebilmeleri, her mikroservisin en uygun araçlarla inşa edilmesine imkan tanıyor. Bununla birlikte, mikroservisler, uygulama performansını artıran yatay ölçeklenebilirlik sunarken, hata yalıtımı sayesinde bir serviste meydana gelen problemin diğer servisleri etkilemesini engelliyor. Mikroservis üzerine yaptığım araştırmayı tamamladıktan sonra aşağıdaki slaytları hazırladım ve yöneticilerime bir zoom toplantısı ile sunarak günümü tamamladım.

Why Microservice Architecture?

1. **Code base is better understandable**
2. **Highly maintainable and testable - enables rapid and frequent development and deployment**
3. **Loosely coupled with other services so independently deployable**
4. **Improved fault isolation**
5. **Adapting new technologies is easier**
6. **The faster IDE makes developer more productive**

Resim-41: Mikroservislerin avantajları

What are Microservices?



Resim-42: Mikroservislerin görselleştirilmesi

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		

İş Yerinde Çalışılan Birim: Yazılım Geliştirme (Kredi Kartı) Sistemleri	Çalışma Tarihi:12/08/2024

Yapılan iş: BTK Akademi üzerinden “Algoritma ve Veri Yapıları İleri Seviye” kursunun alınması.

BTK Akademi’de yer alan “Algoritma ve Veri Yapıları İleri Seviye” kursunun ilk üç bölümünü tamamlayarak, bu alandaki bilgi ve becerilerimi önemli ölçüde geliştirdim. Kursun ilk bölümlerinde, algoritmalar hakkında temel kavramları derinlemesine öğrenme fırsatım oldu. Algoritma analizine yönelik olarak zaman karmaşıklığı, çalışma zamanı analizi gibi konular öğretildi. Bu süreçte, Big O notasyonu ve asimptotik notasyonlar gibi kritik analiz araçlarını da detaylı bir şekilde öğrendim. Bu kavramlar, algoritmaların verimliliğini ve performansını değerlendirmek için temel oluşturdu.
Bu teorik temeli sağlamlaştırdıktan sonra, kursun ilerleyen kısımlarında veri yapıları üzerinde çalışmaya başladım. İlk olarak, diziler ve koleksiyonlar gibi temel veri yapıları ele alındı. Bu yapıların nasıl çalıştığını öğrenmekle kalmayıp, aynı zamanda pratik uygulamalarla pekiştirdim. Özellikle, sıfırdan bir generic Array sınıfı oluşturarak, öğrendiğim kavramları gerçek dünya problemlerine uygulayabilme yeteneğimi geliştirdim. Bu uygulama, yalnızca teorik bilgilere sahip olmanın ötesine geçerek, yazılım geliştirme sürecinde karşılaşabileceğim gerçek zorluklara hazırlanmamı sağladı.

```
public class Array<T>: IEnumerable<T>, ICloneable
{
    private T[] innerList;
    public int Count { get; private set; }
    public int Capacity { get { return innerList.Length; } }

    public Array()
    {
        innerList = new T[2];
        Count = 0;
    }

    private void IncreaseCapacity()
    {
        var newArr = new T[Capacity * 2];
        //for (int i = 0; i < Count; i++)
        //{
        //  newArr[i] = innerList[i];
        //}
        System.Array.Copy(innerList, newArr, Capacity);
        innerList = newArr;
    }

    private void HalfCapacity()
    {
        if (Capacity > 2)
        {
            var half = new T[Capacity / 2];
            System.Array.Copy(innerList, half, Capacity / 2);
            innerList = half;
        }
    }

    public void Add(T item)
    {
        if (Count >= Capacity)
            this.IncreaseCapacity();
        innerList[Count] = item;
        Count++;
    }

    //foreach döngüsünü kullanabilmek için
    public IEnumerator<T> GetEnumerator()
    {
        return innerList.Take(Count).GetEnumerator();
    }

    System.Collections.IEnumerator System.Collections.IEnumerable.GetEnumerator()
    {
        return GetEnumerator();
    }

    public object Clone()
    {
        return this.MemberwiseClone();
    }
}
```

Resim-43: Array sınıfının kaynak kodları

Yapılan işin süresi	Stajyerin imzası	İş yeri yetkilisinin imzası
1 gün		