



T.C

Celal Bayar Üniversitesi

Kırkağaç Meslek Yüksekokulu

Bilgisayar Programcılığı Bölümü

PERSONEL VARDİYA OTOMASYON PROGRAMI

(Sistem Analiz ve Tasarım Dersi)

Hazırlayanlar

221809079 Sıdıka DİKER

221809029 Mesut YILDIZ

221809051 Mustafa YANIK

221809061 Batuhan ASLITÜRK

221809067 Gülcan BİLEM

221809099 Gaye SAY

Danışman

Öğr. Gör. Murat ALBAYRAK

Manisa

2023

İçindekiler

1. Giriş.....	3
2. Materyal ve Yöntem.....	3
3. Sistemin Tanıtımı	3
4. Hazırlanan Uygulamanın Tanıtımı.....	4
4.1. Giriş Ekranı.....	4
4.2. Yönetici Ekranı.....	5
4.3. Personel Ekleme Ekranı	5
4.4. Personel Silme Ekranı	6
4.5. Personel Güncelleme Ekranı.....	7
4.6. Unvan İşlemleri Ekranı.....	7
4.7. Nöbet İşlemleri Ekranı	8
4.8. Nöbet Ekleme Ekranı	9
4.9. Nöbet Silme Ekranı	9
4.10. Nöbet Güncelleme Ekranı.....	10
4.11. Çıktı Alma Ekranı.....	10
4.12. Kullanıcı Ekranı	11
Kullanıcı Kılavuzu – Nasıl Kurulur?	11
Ekler.....	13
Kaynak Kodlar	14
Kaynakça.....	48

1. Giriş

1.1. Konunun Önemi

Bir nöbet programı, belirli bir zaman dilimindeki görevlerin, işlerin veya sorumlulukların belirli kişilere atanması ve düzenlenmesi için kullanılır. Özellikle sağlık sektörü, güvenlik sektörü, perakende ve endüstriyel alanlarda kullanılan nöbet programları, belirli zaman dilimlerindeki operasyonların sorunsuz ve düzenli bir şekilde yürütülmesini sağlar.

1.1. Çalışmanın Önemi ve Amacı

Vardiya otomasyon programları, işletmelerdeki vardiya planlamasını kolaylaştırarak personel yönetimini optimize eder, zaman ve kaynakların daha etkin kullanılmasını sağlar.

1.2. Çalışmanın Kapsamı

Bu çalışma, personel vardiya planlaması ve nöbetlerin otomatik oluşturulması üzerine yoğunlaşırken, aynı zamanda izin dönemlerini de göz önünde bulundurarak vardiya düzenlemelerini yapma amacını taşır.

2. Materyal ve Yöntem

2.1. Verilerin Elde Edilmesinde İzlenen Yöntem

Vardiya otomasyonu için veriler genellikle personel bilgileri, çalışma takvimi, izin durumu gibi kaynaklardan elde edilir. Bu bilgiler, veri tabanlarından alınarak işlenir, analiz edilir ve ardından vardiya planlaması için kullanılır. Veri toplama süreci genellikle yazılım araçları veya veri girişi yoluyla gerçekleştirilir.

3. Sistemin Tanıtımı

3.1. Faaliyet Alanı

Genellikle işletmelerde veya kurumlarda çalışan personellerin nöbetlerini oluşturmayı, düzenlemeyi ve izinlerini takip etmeyi amaçlayan bir otomasyon sistemini temsil ediyor.

3.2. Programın Analizi

Personel Bilgileri: Program, personel bilgilerini içeren bir veri tabanı kullanıyor. Bu, her personelin isim, kimlik numarası veya benzersiz bir tanımlayıcı ve diğer bilgilerini saklamak için kullanılır.

Nöbet Atama ve Oluşturma: Belirli tarih aralıkları için nöbet oluşturmayı ve personeller arasında bu nöbetleri dağıtmayı sağlar. Bu, vardiya planlaması ve nöbet atama işlevselliği sağlar.

İzin Yönetimi: Personellerin izin günlerini takip etmek için bir mekanizma içerir. İzin günleri, nöbet atama işlemleri sırasında dikkate alınır.

Raporlama: Program, oluşturulan nöbet listesini görüntülemek ve dışa aktarmak için bir raporlama aracı içerir.

4. Hazırlanan Uygulamanın Tanıtımı

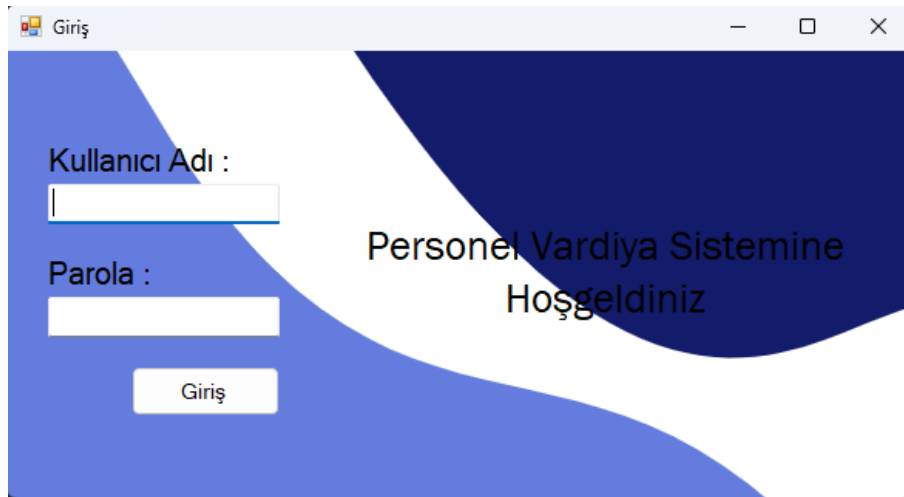
4.1. Giriş Ekranı

Giriş ekranı, programın güvenlik ve yetkilendirme açısından kullanıcıları doğrulamak ve belirli erişim izinleri sağlamak için kullanılır. Genellikle şu amaçlar için kullanılır:

Kullanıcı Doğrulama: Kullanıcıların kimliklerini doğrulamak için kullanılır. Kullanıcı adı ve şifre gibi kimlik bilgileriyle giriş yapma işlemi burada gerçekleşir.

Erişim Kontrolü: Program içindeki farklı özelliklere, fonksiyonlara veya verilere erişim yetkilerini belirlemek için kullanılır. Örneğin, bir yönetici veya personel farklı izinlere sahip olabilir.

Güvenlik: Programın veri tabanındaki hassas bilgileri korumak için kullanılır. Kullanıcıların sadece yetkilendirildikleri alanlara erişmesi sağlanır.



Şekil 1-Personel Vardiya Otomasyonu Giriş Ekranı

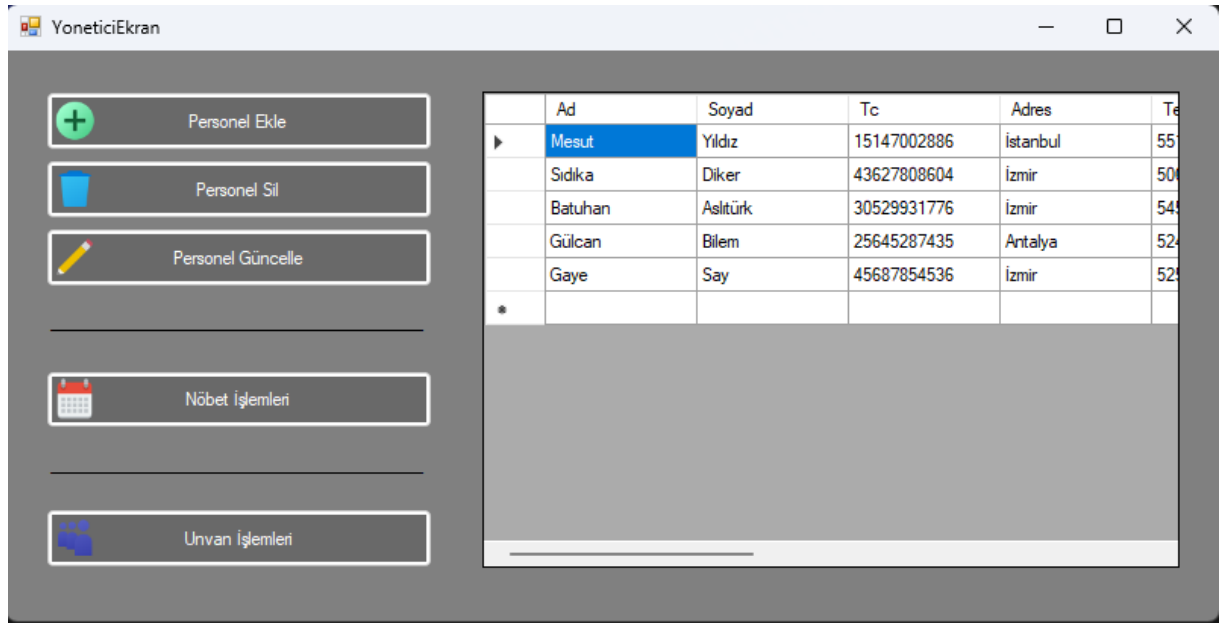
4.2. Yönetici Ekranı

Bu yönetici ekranı genellikle bir uygulamanın yönetim arayüzünü barındırır. Yönetici veya yetkilendirilmiş kullanıcılar, genellikle birkaç temel işlevi gerçekleştirmek için bu tür bir ekranı kullanırlar:

Personel Yönetimi: Yeni personel eklemek, mevcut personelleri düzenlemek, silmek veya personel verilerini güncellemek gibi işlemler bu ekran aracılığıyla yapılabilir.

Unvan veya Departman Yönetimi: Bu ekran, unvanlar, departmanlar veya pozisyonlar gibi işlemleri yönetmeyi sağlayabilir. Yeni unvanlar eklemek, mevcut unvanları güncellemek veya silmek için kullanılabilir.

Nöbet Planlama: Nöbetleri düzenlemek, personeller arasında nöbet ataması yapmak veya nöbet programını oluşturmak için bu ekran kullanılabilir.



Şekil 2- Programın Yönetici Ekranı

4.3. Personel Ekleme Ekranı

Personel ekleme ekranı, yeni personel kaydı oluşturmak için tasarlanmış bir arayüzdür. Bu ekran genellikle şu işlevleri içerir:

Personel Bilgileri: Yeni bir personel kaydı oluştururken veya mevcut bir personelin bilgilerini güncellerken, genellikle kişisel bilgiler (Ad, soyadı, TC kimlik numarası, adres, telefon numarası, e-posta adresi vb.) girilir.

Pozisyon Bilgisi: Sicil numarası, kadro, unvan gibi çalıştığı pozisyonla ilgili bilgiler.

Giriş Bilgileri: Kullanıcı adı, şifre gibi giriş bilgileri (sistem kullanımı için).

PersonekEkle

Adı :

Mail :

Soyadı :

Sicil No :

Tc :

Unvan :

Adres :

Kadro :

Telefon :

Şifre :

KAYDET

Şekil 3- Programın Personel Ekleme Ekranı

4.4. Personel Silme Ekranı

Personel silme ekranı, mevcut personel kayıtlarının sistemden kaldırılmasına olanak tanır.

Silme İşlemi: Seçilen personelin kaydı bu ekran aracılığıyla veritabanından kaldırılabilir.

PersonelSil

	personelid	Ad	Soyad	Tc	Adres	Telefon	Mail	Sicil No
▶	1004	Mesut	Yıldız	15147002886	İstanbul	5511670231	mst12yldz@gmail...	9029
	1012	Sıdika	Diker	43627808604	İzmir	5060678497	sdkdkr@gmail.com	9079
	1014	Batuhan	Aslıtürk	30529931776	İzmir	5453478463	basliturk48@ma...	9061
	1016	Gülcan	Bilem	25645287435	Antalya	5246578540	glcnblm@gmail.c...	9067
	2016	Gaye	Say	45687854536	İzmir	5258547869	gysy@gmail.com	9099
*								

Kayıt Sil

Şekil 4- Programın Personel Silme Ekranı

4.5. Personel Güncelleme Ekranı

Personel Düzenleme ekranı, mevcut personel kayıtlarının güncellenmesine olanak tanır.

Personel Seçme: Mevcut personel listesinden güncellenecek personel seçilir.

Bilgi Güncelleme: Formdaki alanlar üzerinde yapılan değişikliklerle personel bilgileri güncellenir.

personelId	Ad	Soyad	Tc
1004	Mesut	Yıldız	15147002886
1012	Sıdıka	Diker	43627808604
1014	Batuhan	Aslıtürk	30529931776
1016	Gülcan	Bilem	25645287435
2016	Gaye	Say	45687854536

Şekil 5- Programın Personel Düzenleme Kısmı

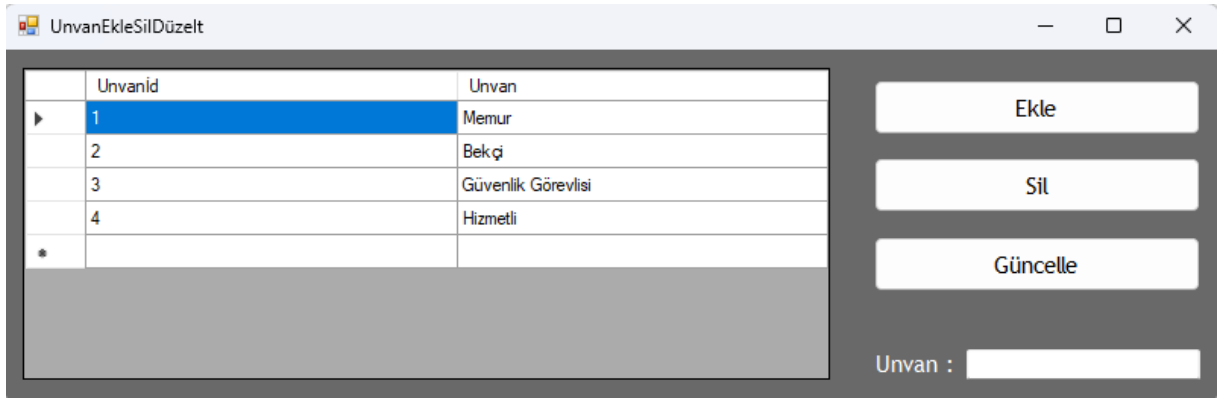
4.6. Unvan İşlemleri Ekranı

Unvan işlemleri ekranı, personel yönetimiyle ilgili olarak unvanların eklendiği, silindiği veya güncellendiği bir arayüz sunar.

Unvan Ekleme: Mevcut unvanlara yeni unvanlar eklemek için kullanılır. Yeni bir unvan girilir ve eklenir.

Unvan Silme: Var olan unvanlardan gereksiz veya artık kullanılmayanları silmek için kullanılır. Silinecek unvan seçilir ve silme işlemi gerçekleştirilir.

Unvan Güncelleme: Var olan unvanların isimlerini veya detaylarını değiştirmek için kullanılır. Güncellenmek istenen unvan seçilir, yeni bilgiler girilir ve güncelleme işlemi yapılır.



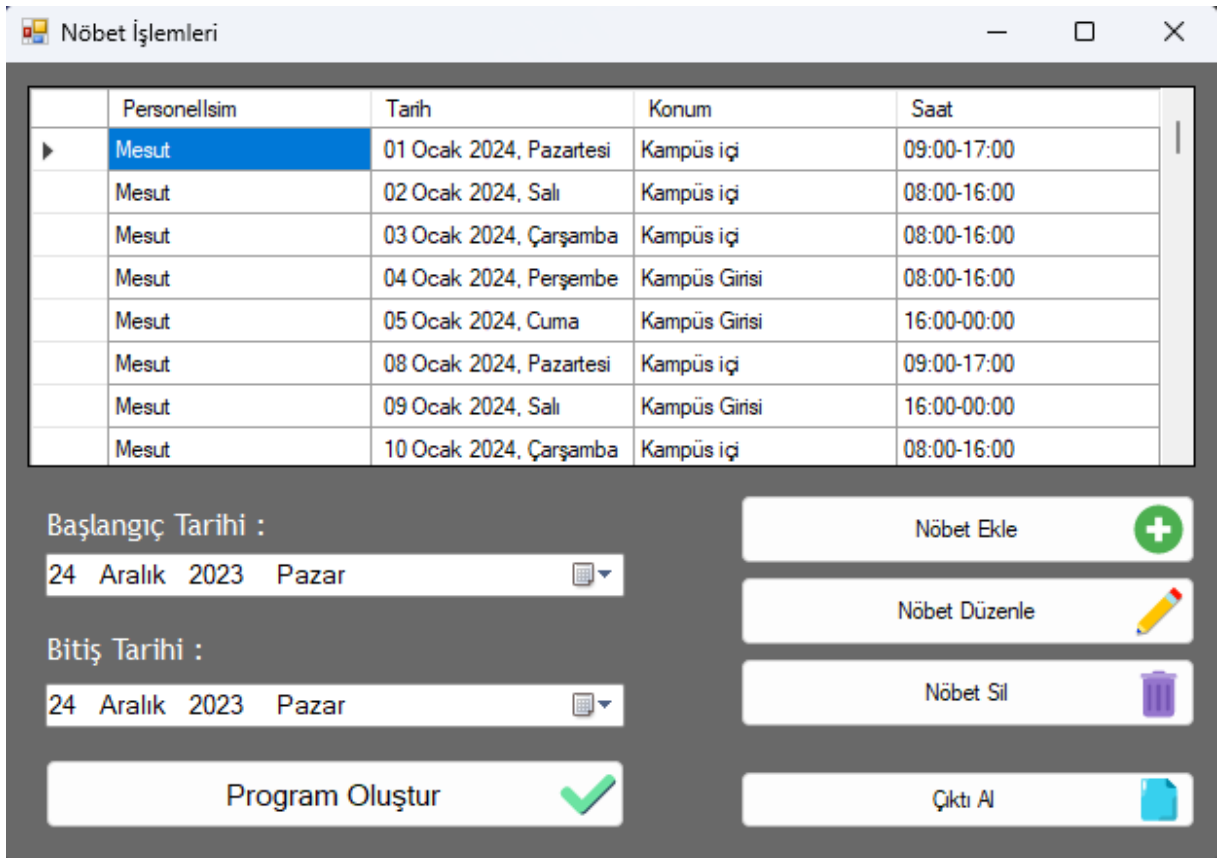
Şekil 6- Programın Unvan İşlemleri Ekranı

4.7. Nöbet İşlemleri Ekranı

Bu programın "Nöbet İşlemleri" ekranı, bir kurumun vardiya planlamasını yapmak ve yönetmek için tasarlanmıştır.

Nöbet Programı Oluşturma: Belirli bir tarih aralığı için vardiya programını otomatik oluşturur. Bu, çalışanların belirli tarihlerde hangi saatlerde hangi konumda (örneğin, kampüs içi veya kampüs girişi) görev alacaklarını belirler.

Veri Görüntüleme: Oluşturulan vardiya programını, atanan nöbetleri ve ilgili çalışanları detaylı olarak görüntüleyebilir. Bu genellikle bir tablo veya liste üzerinden yapılır.



Şekil 7- Programın Nöbet İşlemleri Ekranı

4.8. Nöbet Ekleme Ekranı

Bu programdaki "Nöbet Ekleme Ekranı", bir çalışana veya personelin belirli bir tarihte hangi konumda, saatte nöbet tutacağını sisteme kaydetmek için tasarlanmıştır. Genellikle aşağıdaki işlevleri yerine getirir:

Personel Seçme: Çalışanların isimleri veya kimlik bilgileri bir listede gösterilir. Bu ekranda, nöbet ataması yapılacak olan personel seçilir.

Tarih Seçme: Nöbet atanacak tarih belirlenir. Bu, hangi tarihte nöbetin gerçekleşeceğini belirlemek için kullanılır.

Konum ve Saat Seçme: Nöbetin hangi konumda (örneğin, kampüs içi veya kampüs girişi gibi) ve hangi saat diliminde (örneğin, sabah veya öğleden sonra) gerçekleşeceği seçilir.

	nobetId	personel	tarih	konum	saat
▶	8786	1004	01 Ocak 2024, ...	Kampüs içi	09:00
	8790	1004	02 Ocak 2024, ...	Kampüs içi	08:00
	8795	1004	03 Ocak 2024, ...	Kampüs içi	08:00
	8798	1004	04 Ocak 2024, ...	Kampüs Girişi	08:00
	8804	1004	05 Ocak 2024, ...	Kampüs Girişi	16:00
	8821	1004	08 Ocak 2024, ...	Kampüs içi	09:00
	8824	1004	09 Ocak 2024, ...	Kampüs Girişi	16:00

Personel : Mesut
Tarih : 10 Aralık 2023 Pazar
Konum :
Saat :
Vardiya Oluştur

Şekil 8- Programın Nöbet Ekle Ekranı

4.9. Nöbet Silme Ekranı

Bu programdaki "Nöbet Silme Ekranı", bir yönetici tarafından kullanılır

Belirli Bir Personelin Belirli Bir Nöbetini Silme: Bu ekran, belirli bir personelin belirli bir tarihte tuttuğu nöbeti sistemden kaldırmak için kullanılır.

Tüm Bir Personelin Nöbetlerini Silme: Ayrıca, bu ekran belirli bir çalışana ait tüm nöbet kayıtlarını silmek için kullanılabilir.

	nobetId	personel	tarih	konum	saat
▶	8786	1004	01 Ocak 20...	Kampüs içi	09:00-17:00
	8790	1004	02 Ocak 20...	Kampüs içi	08:00-16:00
	8795	1004	03 Ocak 20...	Kampüs içi	08:00-16:00
	8798	1004	04 Ocak 20...	Kampüs Girişi	08:00-16:00
	8804	1004	05 Ocak 20...	Kampüs Girişi	16:00-00:00
	8821	1004	08 Ocak 20...	Kampüs içi	09:00-17:00
	8824	1004	09 Ocak 20...	Kampüs Girişi	16:00-00:00
	8830	1004	10 Ocak 20...	Kampüs içi	08:00-16:00
	8832	1004	11 Ocak 20...	Kampüs Girişi	00:00-08:00

Personel : Mesut
Nöbet Sil
Bütün Nöbetleri Sil

Şekil 9 - Programın Nöbet Silme Ekranı

4.10. Nöbet Güncelleme Ekranı

"Nöbet Düzenleme ve Güncelleme Ekranı", yönetici veya yetkili tarafından kullanılır ve aşağıdaki işlemlere sahiptir:

Belirli Bir Nöbeti Düzenleme: Bu ekran, belirli bir personelin belirli bir tarihte tuttuğu nöbetin bilgilerini düzenlemek için kullanılır. Örneğin, nöbetin yapıldığı konumu veya saati değiştirme, nöbetin tutulduğu tarih ve kişi gibi bilgileri güncelleme amacıyla kullanılabilir.

nobetId	personel	tarih	konum	saat
8786	1004	01 Ocak 202...	Kampüs içi	09:00-17:00
8790	1004	02 Ocak 202...	Kampüs içi	08:00-16:00
8795	1004	03 Ocak 202...	Kampüs içi	08:00-16:00
8798	1004	04 Ocak 202...	Kampüs Girişi	08:00-16:00
8804	1004	05 Ocak 202...	Kampüs Girişi	16:00-00:00
8821	1004	08 Ocak 202...	Kampüs içi	09:00-17:00
8824	1004	09 Ocak 202...	Kampüs Girişi	16:00-00:00
8830	1004	10 Ocak 202...	Kampüs içi	08:00-16:00

Şekil 10- Programın Nöbet Güncelleme Ekranı

4.11. Çıktı Alma Ekranı

Bu programdaki "Çıktı Alma Ekranı" belirli koşullara göre veri tabanından çekilen bilgilerin bir rapor haline getirilmesi ve PDF formatında dışa aktarılmasını sağlar. Özellikle personel vardiyaları, belirli tarih aralıklarında çalışma bilgileri gibi verilerin raporlanması için tasarlanmıştır.

Veri Seçme ve Listeleme: Personel ismi ve belirlenen tarih aralıkları kullanıcı tarafından seçilerek, bu kriterlere göre veritabanından veriler çekilir. Bu seçilen veriler daha sonra bir DataGridView üzerinde listelenir.

Rapor Oluşturma: DataGridView'deki görüntülenen veriler, seçilen sütun başlıklarıyla birlikte PDF formatında bir rapora dönüştürülür.

PDF Olarak Dışa Aktarma: Rapor, PDF formatında kaydedilmesi için kullanıcıya bir dosya kaydetme iletişim kutusu (SaveFileDialog) sağlar.

Personel :
Başlangıç Tarihi : 24 Aralık 2023 Pazar
Bitiş Tarihi : 24 Aralık 2023 Pazar

Listele Çıktı Al

4.12. Kullanıcı Ekranı

Bu programın "Kullanıcı Personel Ekranı" belirli bir kullanıcının oturum açtığı veya kimlik doğrulaması yaptığı bir ekranı temsil eder.

Kullanıcıya Özgü Verileri Listeleme: Program, kullanıcının kimliğiyle eşleşen personelin çalışma bilgilerini veritabanından çeker. Özellikle bu ekranda, belirli bir personelin vardiyaları, çalışma saatleri, vardiya konumları gibi bilgiler görüntülenebilir.

PDF Rapor Alma: Kullanıcının vardiyalarını veya çalışma bilgilerini PDF raporu olarak dışa aktarır. Kullanıcı, DataGridView üzerinde görüntülenen bilgileri PDF olarak kaydedebilir.



	Personelsim	Tarih	Konum	Saat
▶	Mesut	01 Ocak 2024, Pazartesi	Kampüs içi	09:00-17:00
	Mesut	02 Ocak 2024, Salı	Kampüs içi	08:00-16:00
	Mesut	03 Ocak 2024, Çarşamba	Kampüs içi	08:00-16:00
	Mesut	04 Ocak 2024, Perşembe	Kampüs Girişi	08:00-16:00
	Mesut	05 Ocak 2024, Cuma	Kampüs Girişi	16:00-00:00
	Mesut	08 Ocak 2024, Pazartesi	Kampüs içi	09:00-17:00
	Mesut	09 Ocak 2024, Salı	Kampüs Girişi	16:00-00:00
	Mesut	10 Ocak 2024, Çarşamba	Kampüs içi	08:00-16:00
	Mesut	11 Ocak 2024, Perşembe	Kampüs Girişi	00:00-08:00
	Mesut	12 Ocak 2024, Cuma	Kampüs Girişi	08:00-16:00

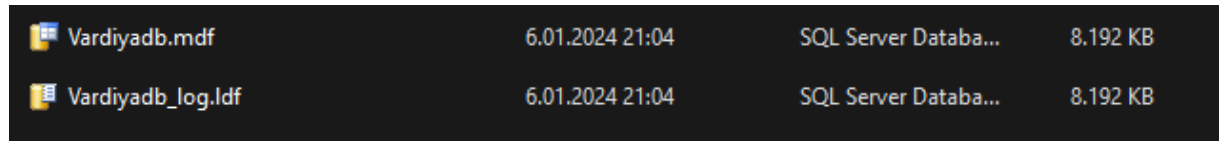
Çıktı Al

Şekil 11- Programın Kullanıcı Ekranı

Kullanıcı Kılavuzu – Nasıl Kurulur?

Personel vardiya otomasyon programını aşağıdaki adımları uygulayarak bilgisayarınıza kurabilirsiniz

İlk Olarak;

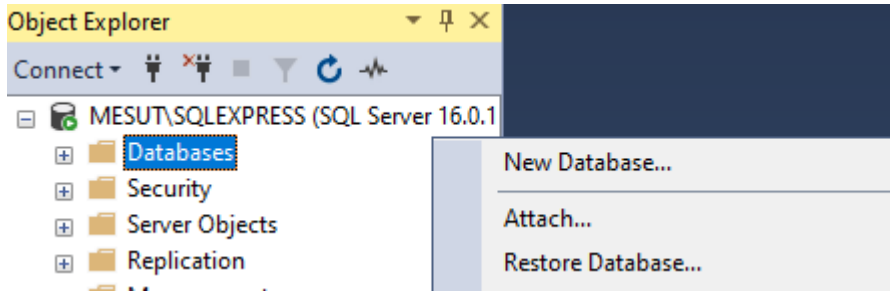


Vardiadb.mdf	6.01.2024 21:04	SQL Server Databa...	8.192 KB
Vardiadb_log.ldf	6.01.2024 21:04	SQL Server Databa...	8.192 KB

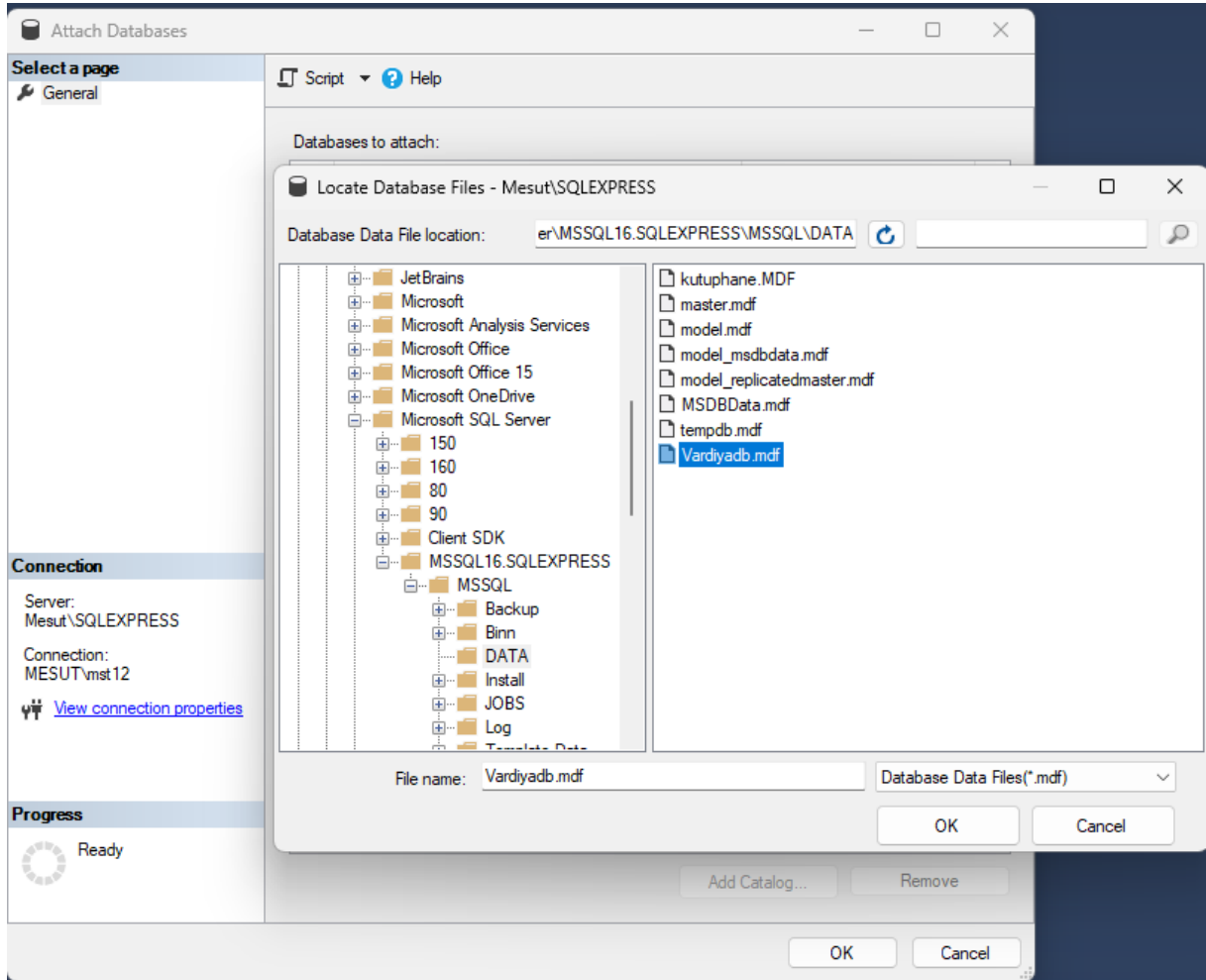
Size verdiğimiz kurulum dosyalarındaki vardiya database dosyalarını mssql içerisine yüklemeniz gerekecektir.

...\\Program Files\\Microsoft SQL Server\\MSSQL16.SQLEXPRESS\\MSSQL\\DATA yoluna gerekli dosyaları atmalısınız

Ardından Sql server management studioyu açalım



Buradan Databases sağ tıklayıp Attach diyelim. Karşımıza çıkan ekranda Add butonuna tıklayıp DATA klasörüne attığımız database dosyasını buluyoruz ve seçip veritabanımızı ekliyoruz.



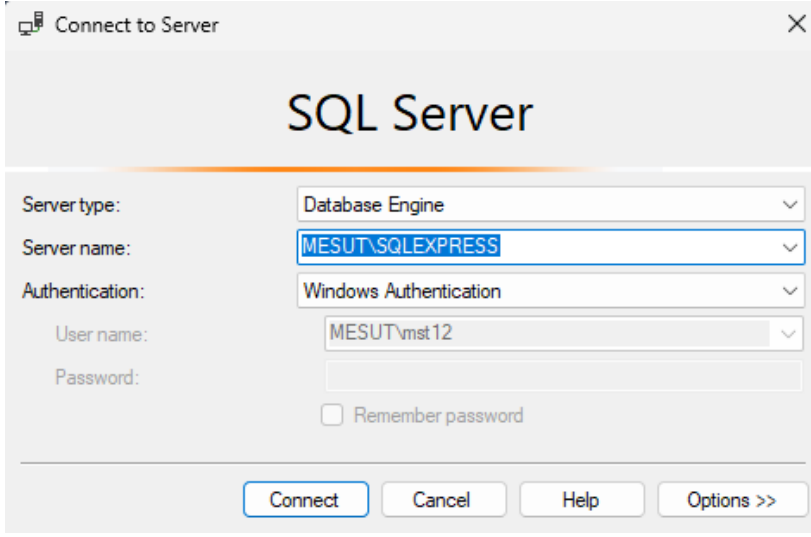
Ardından Size Vermiş Olduğumuz



Dosyasını bilgisayarınızın C: Sürücüsüne Atalım. Metin Dosyası içerisindeki bilgisayar sql adresi kendi bilgisayar ismimizi yazıyoruz

DATA SOURCE=Bilgisayar Sql Adresi; INITIAL CATALOG=Vardiyadb; INTEGRATED SECURITY = TRUE

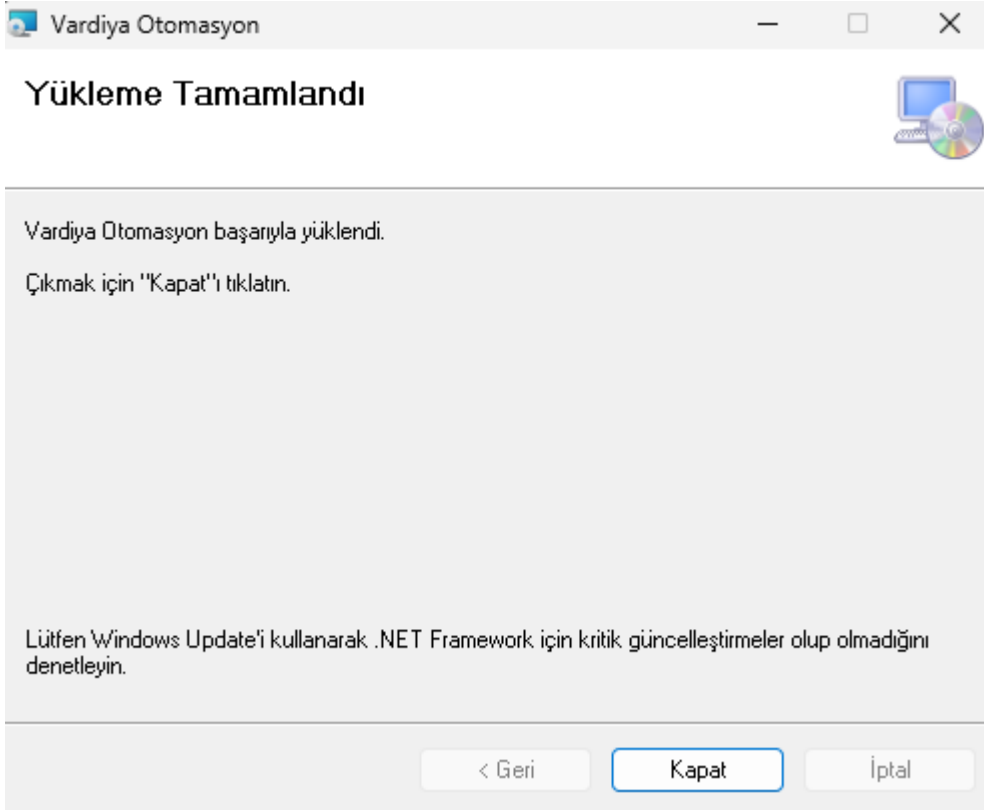
Bunu Management Studioya girerken açılan connect ekranından öğrenebilirsiniz



Geriye bir tek setup dosyasını çalıştırıp kurmak kalıyor

setup.exe	6.01.2024 16:28	Uygulama	568 KB
-----------	-----------------	----------	--------

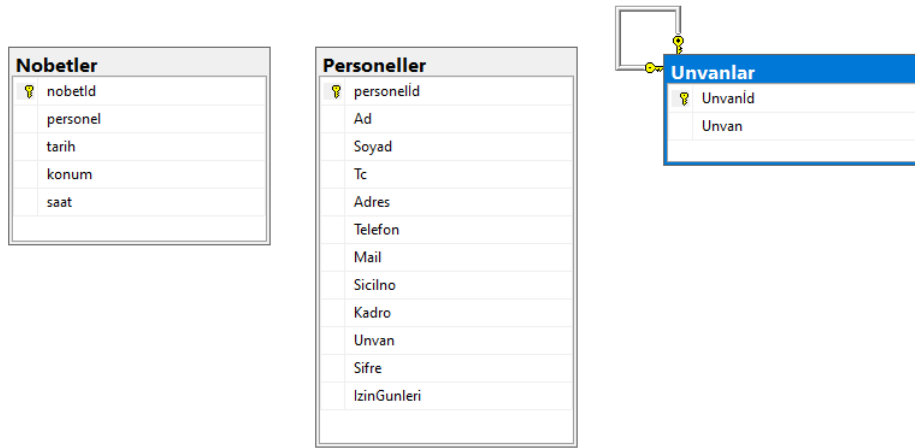
Çalıştırıp programın kurulacağı yeri seçip tamamlayalım



Böylelikle programı masaüstüne gelen kısayoldan bilgisayarınızda kullanmaya başlayabilirsiniz. :)

Ekler

Programın Data Diyagramı



Kaynak Kodlar

1.Giriş Formu Kod Bloğu

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Common;
using System.Data.SqlClient;
using System.Diagnostics.Eventing.Reader;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static Vardiya_Otomasyon_Programı.Program;

namespace Vardiya_Otomasyon_Programı
{
    public partial class Giriş : Form
    {
        public Giriş()
        {
            InitializeComponent();

            bağlantı bgl = new bağlantı();

            private void btnGiris_Click(object sender, EventArgs e) //Personel ve
            Admin kullanıcı adı parola kontrolü
            {
                if (txtKulAd.Text == "" || txtParola.Text == "") //Boş İşe Hata
                ver
                {
                    MessageBox.Show("Boş Alanları Doldurunuz!");
                }
                else if (txtKulAd.Text == "admin" && txtParola.Text == "admin")
                //doğruysa Yönetici Paneline aktar
                {
                    YoneticiEkran yoneticiEkran = new YoneticiEkran();
                }
            }
        }
    }
}

```

```

        yoneticiciEkran.Show();
        this.Hide();
    }
    else
    {
        try
        {
            using (SqlConnection connection = new
SqlConnection(bgl.Adres))
            {
                SqlCommand sqlCommand = new SqlCommand("SELECT * FROM
Personeller WHERE Tc = @Tc and Sifre = @Sifre ", connection);
                sqlCommand.Parameters.AddWithValue("@Tc", txtKulAd.Text);
                sqlCommand.Parameters.AddWithValue("@Sifre",
txtParola.Text);
                SqlDataAdapter sqlDataAdapter = new
SqlDataAdapter(sqlCommand);
                DataTable dataTable = new DataTable();

                sqlDataAdapter.Fill(dataTable);

                if (dataTable.Rows.Count > 0)           //Girilen Tc İle
Şifre Doğruysa Kullanıcı Paneline Yönlendir
                {
                    MessageBox.Show("Giriş Başarılı");
                    string tc = txtKulAd.Text;
                    KullanıcıEkran kullaniciEkran = new
KullanıcıEkran(tc);

                    kullaniciEkran.Show();
                    this.Hide();
                }
                else
                {
                    MessageBox.Show("Tc Veya Şifre Yanlış!");
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("" + ex);
        }
    }
}

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

private void Giris_KeyPress(object sender, KeyPressEventArgs e) //Enter
Tuşuna Basıldığında Giriş Tuşuna bas
{
    if (e.KeyChar == (char)Keys.Enter)
    {
        btnGiris.PerformClick();
        e.Handled = true;
    }
}
}
}

```

2. Yönetici Panel Formu Kod Bloğu

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static Vardiya_Otomasyon_Programı.Program;

namespace Vardiya_Otomasyon_Programı
{
    public partial class YoneticiEkran : Form
    {
        public YoneticiEkran()
        {
            InitializeComponent();

            Image originalImage = Properties.Resources.Plus;
            Image resizedImage = new Bitmap(originalImage, new Size(30, 30));
            button1.Image = resizedImage;

            Image originalImage2 = Properties.Resources.Delete;
            Image resizedImage2 = new Bitmap(originalImage2, new Size(30, 30));
            button2.Image = resizedImage2;

            Image originalImage3 = Properties.Resources.Edit;
            Image resizedImage3 = new Bitmap(originalImage3, new Size(30, 30));
            //İconları ayarlama
            button3.Image = resizedImage3;

            Image originalImage4 = Properties.Resources.calendar_icon;
            Image resizedImage4= new Bitmap(originalImage4, new Size(30, 30));
            button4.Image = resizedImage4;

            Image originalImage5 = Properties.Resources.MySpace;
            Image resizedImage5 = new Bitmap(originalImage5, new Size(30, 30));
            button5.Image = resizedImage5;
        }

        bağlantı bgl = new bağlantı();

        void VeriGetir() //Sql Personeller Tablosunu Çağır
        {
            using (SqlConnection connection = new SqlConnection(bgl.Adres))
            {
                connection.Open();

                SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT Ad,
Soyad, Tc, Adres, Telefon, Mail, Sicilno, Kadro, Unvan, Sifre, IzinGunleri FROM
Personeller", connection);

                DataTable tablo = new DataTable();

                dataAdapter.Fill(tablo);

                dataGridView1.DataSource = tablo;

                connection.Close();
            }
        }
    }
}
```



```

    }
}

private void YoneticiEkran_Load(object sender, EventArgs e) //Method
Çalıştır (sayfa açıldığında direkt yüklenir)
{
    VeriGetir();
}

e) private void YoneticiEkran_FormClosed(object sender, FormClosedEventArgs
//Form Kapatıldığında
{
    Giris giris = new Giris();
    giris.Show();
}

private void button1_Click(object sender, EventArgs e) //Personel Ekle
Kısına Yönlendir
{
    PersonekEkle personekEkle = new PersonekEkle();
    personekEkle.Show();
    this.Hide();
}

private void button2_Click(object sender, EventArgs e) //Personel Sil
Kısına Yönlendir
{
    PersonelSil personelSil = new PersonelSil();
    personelSil.Show();
    this.Hide();
}

private void button3_Click(object sender, EventArgs e) //Personel
Bilgileri Güncelleme Kısmına Yönlendir
{
    PersonelDüzenle personelDüzenle = new PersonelDüzenle();
    personelDüzenle.Show();
    this.Hide();
}

private void YoneticiEkran_FormClosed_1(object sender,
FormClosedEventArgs e) //Form Kapatıldığında Giriş Ekranına Yönlendir
{
    Giris giris = new Giris();
    giris.Show();
}

private void button5_Click(object sender, EventArgs e) //Unvan İşlemleri
Kısına Yönlendir
{
    UnvanEkleSilDüzeltil unvanEkleSilDüzeltil = new UnvanEkleSilDüzeltil();
    unvanEkleSilDüzeltil.Show();
    this.Hide();
}

private void button4_Click(object sender, EventArgs e) //Nöbet
Düzenle Formuna Yönlendir
{
    Nöbetİşlemleri nöbetDüzenle = new Nöbetİşlemleri();
}

```

```

        nöbetDüzenle.Show();
        this.Hide();
    }
}

```

3. Personel Ekle Formu Kod Bloğu

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Diagnostics.Eventing.Reader;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static Vardiya_Otomasyon_Programı.Program;

namespace Vardiya_Otomasyon_Programı
{
    public partial class PersonekEkle : Form
    {
        public PersonekEkle()
        {
            InitializeComponent();

            bağlantı bgl = new bağlantı();

            void VeriGetir() //ComboBox'a Unvan Verilerini Getirme
            {
                using (SqlConnection connection = new SqlConnection(bgl.Adres))
                {
                    using (SqlCommand sqlCommand = new SqlCommand("SELECT Unvan FROM
Unvanlar", connection))
                    {
                        connection.Open();

                        using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
                        {
                            while (sqlDataReader.Read())
                            {
                                cmbUnvan.Items.Add(sqlDataReader.GetValue(0));
                            }
                        }
                        connection.Close();
                    }
                }
            }

            int sonEklenenPersonelId = -1;

            private void btnKaydet_Click(object sender, EventArgs e) //Personel
Bilgileri Girildikten Sonra Kaydet
            {
                if (txtAd.Text == "" || txtSoyad.Text == "" || txtTc.Text == "" ||
txtAdres.Text == "" || txtTel.Text == "" || txtSicil.Text == "" || txtMail.Text

```

```

== "" || txtSifre.Text == "" || cmbKadro.SelectedItem == null ||
cmbUnvan.SelectedItem == null)
{
    MessageBox.Show("Boş Alanları Doldurunuz!"); //Boş
değer varsa uyar
}
else if (cmbKadro.SelectedItem.ToString() == "Memur" &&
listBoxİzinGunleri.SelectedItems.Count > 2)
{
    MessageBox.Show("Memur için maksimum 2 izin günü seçilebilir!");
//seçilen kadro Memur ve izin günü 2 den fazlaysa uyar
}
else if (cmbKadro.SelectedItem.ToString() == "İşçi" &&
listBoxİzinGunleri.SelectedItems.Count > 1)
{
    MessageBox.Show("İşçi için maksimum 1 izin günü seçilebilir!");
//seçilen kadro işçi ve izin günü 1 den fazlaysa uyar
}
else
{
    using (SqlConnection connection = new SqlConnection(bgl.Adres))
//Girdiğimiz Tcyi veritabanında sorgula
    {
        using (SqlCommand komut = new SqlCommand("SELECT * FROM
Personeller WHERE Tc = '" + txtTc.Text + "'", connection))
        {
            connection.Open();

            using (SqlDataReader oku = komut.ExecuteReader()) //Veri
tabanında Zaten Olan Personeli Kontrol Etme
            {
                if (oku.Read())
                {
                    MessageBox.Show("Bu Kullanıcı Sistemde Kayıtlı!",
this.ProductName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                }
                else
                {
                    oku.Close();

                    using (SqlCommand sqlCommand = new
SqlCommand("INSERT INTO Personeller (Ad, Soyad, Tc, Adres, Telefon, Mail,
Sicilno, Kadro, Unvan, Sifre) VALUES (@Ad, @Soyad, @Tc, @Adres, @Tel, @Mail,
@Sicil, @Kadro, @Unvan, @Sifre)", connection))
                    {
                        sqlCommand.Parameters.AddWithValue("@Ad",
txtAd.Text); //Girdiğimiz değerleri Sql'e Ekleme
                        sqlCommand.Parameters.AddWithValue("@Soyad",
txtSoyad.Text);
                        sqlCommand.Parameters.AddWithValue("@Tc",
txtTc.Text);
                        sqlCommand.Parameters.AddWithValue("@Adres",
txtAdres.Text);
                        sqlCommand.Parameters.AddWithValue("@Tel",
txtTel.Text);
                        sqlCommand.Parameters.AddWithValue("@Mail",
txtMail.Text);
                        sqlCommand.Parameters.AddWithValue("@Sicil",
txtSicil.Text);
                        sqlCommand.Parameters.AddWithValue("@Kadro",
cmbKadro.SelectedItem.ToString());
                        sqlCommand.Parameters.AddWithValue("@Unvan",
cmbUnvan.SelectedItem.ToString());
                    }
                }
            }
        }
    }
}

```

```

txtSifre.Text);

sqlCommand.Parameters.AddWithValue("@Sifre",

sqlCommand.ExecuteNonQuery();
}

using (SqlCommand idCommand = new
SqlCommand("SELECT IDENT_CURRENT('Personeller') AS SonEklenenId", connection))
//Son Eklenen Personelin İdsini Al
{
    sonEklenenPersonelId =
Convert.ToInt32(idCommand.ExecuteScalar());
    connection.Close();
}

}

if (listBoxİzinGunleri.SelectedItems.Count > 0)
//İzin Günleri 0 dan fazlaysa İzin gününü veya günlerini veritabanına ekleme
{
    string izinGunleri = string.Join(", ",
listBoxİzinGunleri.SelectedItems.Cast<string>());

    connection.Open();

    string updateQuery = "UPDATE Personeller SET
IzinGunleri = COALESCE(IzinGunleri + ', ', '') + @IzinGunleri WHERE personelId =
@PersonelId"; //İzin günlerini güncelleme

using (SqlCommand izinEkleKomut = new
SqlCommand(updateQuery, connection))
{
    izinEkleKomut.Parameters.AddWithValue("@PersonelId", sonEklenenPersonelId);
    izinEkleKomut.Parameters.AddWithValue("@IzinGunleri", izinGunleri);

    izinEkleKomut.ExecuteNonQuery();
}

    connection.Close();
}

    MessageBox.Show("Personel eklendi!",
this.ProductName, MessageBoxButtons.OK, MessageBoxIcon.Information);
}
    connection.Close();
}
}
}

private void PersonekEkle_Load(object sender, EventArgs e) //Sayfa
Açıldığında Gerçekleşecekler
{
    VeriGetir();
}

```

```

e)     private void PersonekEkle_FormClosed(object sender, FormClosedEventArgs
        //Sayfa Kapatıldığında Gerçekleşecekler
        {
            YoneticisiEkran yoneticisiEkran = new YoneticisiEkran();
            yoneticisiEkran.Show();
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static Vardiya_Otomasyon_Programı.Program;

namespace Vardiya_Otomasyon_Programı
{
    public partial class PersonelDüzenle : Form
    {
        public PersonelDüzenle()
        {
            InitializeComponent();

            bağlantı bgl = new bağlantı();

            void VeriGetir() //Personeller Tablosunu Getir
            {
                using (SqlConnection connection = new SqlConnection(bgl.Adres))
                {
                    connection.Open();

                    SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT * FROM
Personeller", connection);

                    DataTable tablo = new DataTable();

                    dataAdapter.Fill(tablo);

                    dataGridView1.DataSource = tablo;

                    connection.Close();
                }

            }

            private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e) // Güncellemek istediğim Personele Tıkladığımda
Personel Bilgilerini Doldur
            {
                txtAd.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();
                txtSoyad.Text =
dataGridView1.CurrentRow.Cells[2].Value.ToString();
                txtTc.Text = dataGridView1.CurrentRow.Cells[3].Value.ToString();
                txtAdres.Text =
dataGridView1.CurrentRow.Cells[4].Value.ToString();
            }
        }
    }
}

```

```

        txtTel.Text = dataGridView1.CurrentRow.Cells[5].Value.ToString();
        txtMail.Text =
dataGridView1.CurrentRow.Cells[6].Value.ToString();
        txtSicil.Text =
dataGridView1.CurrentRow.Cells[7].Value.ToString();
        cmbKadro.Text =
dataGridView1.CurrentRow.Cells[8].Value.ToString();
        cmbUnvan.Text =
dataGridView1.CurrentRow.Cells[9].Value.ToString();
        txtSifre.Text =
dataGridView1.CurrentRow.Cells[10].Value.ToString();

    }

    private void btnGuncelle_Click(object sender, EventArgs e)
// Seçtiğim Personeli Güncelle
    {
        var personelId =
Convert.ToInt32(dataGridView1.CurrentRow.Cells[0].Value.ToString()); //Seçtiğim
Personelin id'sini integer'a Dönüştür

        using (SqlConnection connection = new SqlConnection(bgl.Adres))
        {
            using (SqlCommand sqlCommand = new SqlCommand("UPDATE Personeller
SET Ad = '" + txtAd.Text + "', Soyad = '" + txtSoyad.Text + "', Tc = '" +
txtTc.Text + "', Adres = '" + txtAdres.Text + "', Telefon = '" + txtTel.Text +
"', Mail = '" + txtMail.Text + "', SicilNo = '" + txtSicil.Text + "', Kadro = '"
+ cmbKadro.Text + "', Unvan = '" + cmbUnvan.Text + "', Sifre = '" + txtSifre.Text
+ "' WHERE personelId = '" + personelId + "'", connection)) //Sql Update Komutu
            {
                connection.Open();
                sqlCommand.ExecuteNonQuery();
                MessageBox.Show("Personel güncellendi!", this.ProductName,
MessageBoxButtons.OK, MessageBoxIcon.Information);
                connection.Close();
                VeriGetir();
            }
        }
    }

    private void PersonelDüzenle_Load(object sender, EventArgs e)
//Sayfa Açıldığında Hangi İşlemler gerçekleşeceği
    {
        VeriGetir();

        using (SqlConnection connection = new SqlConnection(bgl.Adres))
//Veritabanındaki Ünvanları Comboboxa çağır
        {
            using (SqlCommand sqlCommand = new SqlCommand("SELECT Unvan FROM
Unvanlar", connection))
            {
                connection.Open();

                using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
                {
                    while (sqlDataReader.Read())
                    {
                        cmbUnvan.Items.Add(sqlDataReader.GetValue(0)); //Her
birini döngüyle ekle
                    }
                }
                connection.Close();
            }
        }
    }

```

```

        }
    }

    private void PersonelDüzenle_FormClosed(object sender,
FormClosedEventArgs e) //Form Kapatıldığında Yönetici Ekranına Yönlendir
    {
        YoneticisiEkran yoneticisi = new YoneticisiEkran();
        yoneticisi.Show();
    }
}

```

5. Personel Sil Form Ekranı

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static Vardiya_Otomasyon_Programı.Program;

namespace Vardiya_Otomasyon_Programı
{
    public partial class PersonelSil : Form
    {
        public PersonelSil()
        {
            InitializeComponent();

            bağlantı bgl = new bağlantı();

            void VeriGetir() //Personeller Tablosunu Getir
            {
                using (SqlConnection connection = new SqlConnection(bgl.Adres))
                {
                    connection.Open();

                    SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT * FROM
Personeller", connection);

                    DataTable tablo = new DataTable();

                    dataAdapter.Fill(tablo);

                    dataGridView1.DataSource = tablo;

                    connection.Close();
                }
            }

            private void button1_Click(object sender, EventArgs e) //
Seçtiğim Personelin Kaydını Sil
            {
                var personelId =
Convert.ToInt32(dataGridView1.CurrentRow.Cells[0].Value.ToString());

                using (SqlConnection connection = new SqlConnection(bgl.Adres))
                {

```

```

        using (SqlCommand sqlCommand = new SqlCommand("DELETE FROM
Personeller WHERE personelİd = '" + personelId + "'", connection))
        {
            connection.Open();
            sqlCommand.ExecuteNonQuery();
            MessageBox.Show("Personel silindi!", this.ProductName,
MessageBoxButtons.OK, MessageBoxIcon.Information);
            connection.Close();
            VeriGetir();
        }
    }

    private void PersonelSil_Load(object sender, EventArgs e) //Sayfa
Açıldığında Gerçekleşecekler
    {
        VeriGetir();
    }

    private void PersonelSil_FormClosed(object sender, FormClosedEventArgs e)
//Sayfa Kapatıldığında Gerçekleşecekler
    {
        YoneticıEkran yoneticıEkran = new YoneticıEkran();
        yoneticıEkran.Show();
    }
}

```

6. Unvan İşlemleri Kod Bloğu

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static Vardiya_Otomasyon_Programı.Program;

namespace Vardiya_Otomasyon_Programı
{
    public partial class UnvanEkleSilDüzeltil : Form
    {
        public UnvanEkleSilDüzeltil()
        {
            InitializeComponent();

            bağlantı bgl = new bağlantı();

            void VeriGetir() //Unvanlar Tablosunu Getirme
            {
                using (SqlConnection connection = new SqlConnection(bgl.Adres))
                {
                    connection.Open();

                    SqlDataAdapter dataAdapter = new SqlDataAdapter("SELECT * FROM
Unvanlar", connection);

                    DataTable tablo = new DataTable();

```



```

        dataAdapter.Fill(tablo);

        dataGridView1.DataSource = tablo;

        connection.Close();
    }
}

private void UnvanEkleSilDüzeltil_Load(object sender, EventArgs e)
//Sayfa Açıldığında Gerçekleşecekler
{
    VeriGetir();
}

private void UnvanEkleSilDüzeltil_FormClosed(object sender,
FormClosedEventArgs e) //Form Kapatıldığında Yönetici Ekranına Yönlendir
{
    YoneticisiEkran yoneticisiEkran = new YoneticisiEkran();
    yoneticisiEkran.Show();
}

private void btnEkle_Click(object sender, EventArgs e) //Unvan Ekle
İşlemi
{
    using (SqlConnection connection = new SqlConnection(bgl.Adres))
    {
        using (SqlCommand sqlCommand = new SqlCommand("INSERT INTO
Unvanlar VALUES ('" + txtUnvan.Text + "')", connection))
        {
            connection.Open();

            sqlCommand.ExecuteNonQuery();

            MessageBox.Show("Ünvan eklendi!", this.ProductName,
MessageBoxButtons.OK, MessageBoxIcon.Information);

            connection.Close();

            VeriGetir();
        }
    }
}

private void btnSil_Click(object sender, EventArgs e) //Unvan Sil İşlemi
{
    using (SqlConnection connection = new SqlConnection(bgl.Adres))
    {
        using (SqlCommand sqlCommand = new SqlCommand("DELETE FROM
Unvanlar WHERE UnvanId = '" + dataGridView1.CurrentRow.Cells[0].Value.ToString()
+ "'", connection))
        {
            connection.Open();
            sqlCommand.ExecuteNonQuery();
            MessageBox.Show("Ünvan Silindi!", this.ProductName,
MessageBoxButtons.OK, MessageBoxIcon.Information);
            connection.Close();
            VeriGetir();
        }
    }
}

private void btnGuncelle_Click(object sender, EventArgs e) //Unvan
Güncelle İşlemi

```

```

        {
            var UnvanId =
Convert.ToInt32(dataGridView1.CurrentRow.Cells[0].Value.ToString());

            using (SqlConnection connection = new SqlConnection(bgl.Adres))
            {
                using (SqlCommand sqlCommand = new SqlCommand("UPDATE Unvanlar
SET Unvan = '" + txtUnvan.Text + "' WHERE UnvanId ='" + UnvanId + "'",
connection))
                {
                    connection.Open();
                    sqlCommand.ExecuteNonQuery();
                    MessageBox.Show("Ünvan Güncellendi!", this.ProductName,
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    connection.Close();
                    VeriGetir();
                }
            }
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e) //Tıkladığımız Satırın Ünvanını Textboxa
doldurma
        {
            txtUnvan.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();
        }
    }
}

```

7. Nöbet İşlemleri Formu Kod Bloğu

```

using iTextSharp.text.pdf;
using iTextSharp.text;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using System.Globalization;
using static Vardiya_Otomasyon_Programı.Program;

namespace Vardiya_Otomasyon_Programı
{
    public partial class Nöbetİşlemleri : Form
    {
        public Nöbetİşlemleri()
        {
            InitializeComponent();

            bağlantı bgl = new bağlantı();

```

```

        List<int> GetPersonelIDs()           //Veri Tabanındaki Personellerin
        Personelİdlerini Alan Ve Lisleyen Metho
        {
            List<int> personelIDs = new List<int>();

            string query = "SELECT Personelİd FROM Personeller;";           //
        Personel tablosundan PersonelID'leri getiren SQL sorgusu

            using (SqlConnection connection = new SqlConnection(bgl.Adres))
            {
                SqlCommand command = new SqlCommand(query, connection);
                connection.Open();

                SqlDataReader reader = command.ExecuteReader();

                while (reader.Read())
                {
                    int personelID = reader.GetInt32(0);           // Eğer
        PersonelID INT ise
                    personelIDs.Add(personelID);
                }

                reader.Close();
            }

            return personelIDs;
        }

    private List<DateTime> GetDatesBetween(DateTime startDate, DateTime
    endDate)           //İki Tarih Arasını Alan ve Bu Tarihleri Listeleyen Method
    {
        TimeSpan fark = endDate - startDate;
        int gunSayisi = (int)fark.TotalDays;

        List<DateTime> tarihListesi = new List<DateTime>();
        for (int i = 0; i <= gunSayisi; i++)
        {
            DateTime tarih = startDate.AddDays(i);
            tarihListesi.Add(tarih);
        }

        return tarihListesi;
    }

    private bool TarihVarMi(DateTime tarih)           //Veri Tabanında Oluşturulan
    Nöbet Programında ki Tarih Var mı Yok mu Kontrol Eden Method
    {
        string turkceTarih = tarih.ToString(" dd MMMM yyyy, dddd", new
        CultureInfo("tr-TR"));

        using (SqlConnection connection = new SqlConnection(bgl.Adres))
        {
            connection.Open();

            string query = "SELECT COUNT(*) FROM Nobetler WHERE Tarih =
        @Tarih";

            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@Tarih", turkceTarih);
            }
        }
    }

```

```

        int count = (int)command.ExecuteScalar();

        return count > 0;
    }
}

private void SilIzinGunlerineGoreNobetler(DateTime baslangicTarihi,
DateTime bitisTarihi) //Nöbet Listesi Oluştuktan Sonra Personelin
//İzin Günü Hangisiyse Onu Silen Method
{
    List<DateTime> tarihListesi = GetDatesBetween(baslangicTarihi,
bitisTarihi);

    foreach (DateTime tarih in tarihListesi)
    {
        string turkceTarih = tarih.ToString(" dd MMMM yyyy, dddd", new
CultureInfo("tr-TR"));

        foreach (int personelID in GetPersonelIDs())
        {
            using (SqlConnection connection = new
SqlConnection(bgl.Adres))
            {
                connection.Open();

                string izinGunuSorgusu = "SELECT IzinGunleri FROM
Personeller WHERE PersonelId = @PersonelID";

                using (SqlCommand izinCommand = new
SqlCommand(izinGunuSorgusu, connection))
                {
                    izinCommand.Parameters.AddWithValue("@PersonelID",
personelID);

                    string izinGunleri =
izinCommand.ExecuteScalar()?.ToString();

                    if (!string.IsNullOrEmpty(izinGunleri))
                    {
                        string[] izinGunlerListesi =
izinGunleri.Split(',');

                        foreach (string izinGun in izinGunlerListesi)
                        {
                            string turkceIzinGun = izinGun.Trim();

                            // Veritabanındaki izin günleri ile
tarihlerin gün adını karşılaştır

                            if (tarih.ToString("dddd", new
CultureInfo("tr-TR")).Equals(turkceIzinGun, StringComparison.OrdinalIgnoreCase))
                            {
                                string query = "DELETE FROM Nobetler
WHERE personel = @PersonelID AND tarih = @Tarih";
                                using (SqlCommand deleteCommand = new
SqlCommand(query, connection))
                                {
                                    deleteCommand.Parameters.AddWithValue("@PersonelID", personelID);

```

```

deleteCommand.Parameters.AddWithValue("@Tarih", turkceTarih);

        deleteCommand.ExecuteNonQuery();
    }
}
}
}
}
connection.Close();
}
}
}

private void btnAta_Click(object sender, EventArgs e) //Nöbet Programı
Oluştur Butonu
{
    DateTime baslangicTarihi = dateTimePicker1.Value;
    DateTime bitisTarihi = dateTimePicker2.Value;

    List<DateTime> tarihListesi = GetDatesBetween(baslangicTarihi,
bitisTarihi);

    Random random = new Random();

    string[] saatDilimleriKampusİci = { "08:00-16:00", "09:00-17:00" };
    string[] saatDilimleriKampusGiris = { "00:00-08:00", "08:00-16:00",
"16:00-00:00" };
    string[] konumlar = { "Kampüs Giris", "Kampüs içi" };

    List<int> personelIDs = GetPersonelIDs(); // Örnek olarak personel
ID'lerini alın

    foreach (DateTime tarih in tarihListesi) // Personellere Nöbet
Atayan Döngü
    {
        if (TarihVarMi(tarih))
        {
            MessageBox.Show("Seçtiğiniz Tarih veya Tarihlerde Zaten Nöbet
Bulunmakta", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            break;
        }

        personelIDs = personelIDs.OrderBy(x => random.Next()).ToList();
// Liste içindeki personelleri karıştır

        int index = 0; // Liste içindeki personel indexini takip
etmek için

        string turkceTarih = tarih.ToString(" dd MMMM yyyy, dddd", new
CultureInfo("tr-TR"));

        foreach (string saatDilimi in saatDilimleriKampusGiris)
//Kampüs Girişi Vardiyalarını Nöbet Tablosuna Ekleyen Döngü
        {
            if (index >= personelIDs.Count) // Personel sayısı saat
dilimi sayısından azsa sıfırdan başla
                index = 0;

```

```

        string randomKonum = konumlar[0];

        using (SqlConnection connection = new
SqlConnection(bgl.Adres))
        {
            connection.Open();

            string query = "INSERT INTO Nobetler (personel, Tarih,
Konum, Saat) VALUES (@PersonelID, @Tarih, @Konum, @Saat)";

            using (SqlCommand command = new SqlCommand(query,
connection))
            {
                int currentPersonelID = personelIDs[index];
                command.Parameters.AddWithValue("@PersonelId",
currentPersonelID);
                turkceTarih);
                randomKonum);

                command.Parameters.AddWithValue("@Tarih",
                command.Parameters.AddWithValue("@Konum",
                command.Parameters.AddWithValue("@Saat", saatDilimi);

                command.ExecuteNonQuery();

                connection.Close();
            }

            index++; // Bir sonraki saat dilimine geçmek için index'i
artır
        }
    }

    foreach (string saatDilimi in saatDilimleriKampusİci)
//Kampüs İçi Vardiyalarını Nöbet Tablosuna Ekleyen Döngü
    {
        if (index >= personelIDs.Count)
            index = 0;

        string randomKonum = konumlar[1]; // Kampüs Girişi için
atama yapılacak

        using (SqlConnection connection = new
SqlConnection(bgl.Adres))
        {
            connection.Open();

            string query = "INSERT INTO Nobetler (personel, Tarih,
Konum, Saat) VALUES (@PersonelID, @Tarih, @Konum, @Saat)";

            using (SqlCommand command = new SqlCommand(query,
connection))
            {
                int currentPersonelID = personelIDs[index];
                command.Parameters.AddWithValue("@PersonelId",
currentPersonelID);
                turkceTarih);
                randomKonum);

                command.Parameters.AddWithValue("@Tarih",
                command.Parameters.AddWithValue("@Konum",
                command.Parameters.AddWithValue("@Saat", saatDilimi);

```

```

        command.ExecuteNonQuery();

        connection.Close();
    }

    index++;
}

SilIzinGunlerineGoreNobetler(baslangicTarihi, bitisTarihi);
}
VeriGetir();           //Güncellenmiş Verileri Göster
}

private void button2_Click(object sender, EventArgs e)           //Nöbet
Düzenle Formuna Yönlendir
{
    NöbetDüzenle nöbetDüzenle = new NöbetDüzenle();
    nöbetDüzenle.Show();
    this.Hide();
}

private void button3_Click(object sender, EventArgs e)           //Nöbet Silme
Formuna Yönlendir
{
    NobetSil nobetSil = new NobetSil();
    nobetSil.Show();
    this.Hide();
}

private void Nöbetİşlemleri_FormClosed(object sender, FormClosedEventArgs
e)           //Form Kapatıldığında Ne Gerçekleşecek
{
    YoneticisiEkran yoneticisiEkran = new YoneticisiEkran();
    yoneticisiEkran.Show();
    this.Hide();
}

private void button1_Click_1(object sender, EventArgs e)           //Nöbet
Ekleme Formuna Yönlendir
{
    NöbetEkle nöbetEkle = new NöbetEkle();
    nöbetEkle.Show();
    this.Hide();
}

void VeriGetir()           //Nöbet Listesini datagridviewe çağıran method
{
    using (SqlConnection connection = new SqlConnection(bgl.Adres))
    {
        using (SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT
p.Ad AS PersonelIsim, n.Tarih,n.Konum,n.Saat FROM Nobetler n INNER JOIN
Personeller p ON n.personel = p.personelId", connection))
        {
            using (DataTable dataTable = new DataTable())
            {
                connection.Open();

                sqlDataAdapter.Fill(dataTable);

                dataGridView1.DataSource = dataTable;
            }
        }
    }
}

```

```

        connection.Close();
    }
}

private void Nöbetİşlemleri_Load(object sender, EventArgs e)
//Sayfa Yüklendiğinde Yani Açıldığında Tabloyu ve Gerekli Bilgileri Çağır
{
    VeriGetir();
}

private void button4_Click(object sender, EventArgs e)
{
    CıktıAl cıktıAl = new CıktıAl();
    cıktıAl.Show();
}
}
}

```

8.Nöbet Ekle Formu Kod Bloğu

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static Vardiya_Otomasyon_Programı.Program;

namespace Vardiya_Otomasyon_Programı
{
    public partial class NöbetEkle : Form
    {
        public NöbetEkle()
        {
            InitializeComponent();

            bağlantı bgl = new bağlantı();

            void VeriGetir()
            {
                var personelId = 0;

                using (SqlConnection connection = new SqlConnection(bgl.Adres))
                {
                    using (SqlCommand sqlCommand = new SqlCommand("SELECT personelİd
FROM Personeller WHERE Ad = '" + cmbPersonel.Text + "'", connection))
                    {
                        connection.Open();

                        using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
                        {

```



```

        while (sqlDataReader.Read())
        {
            personelId =
Convert.ToInt32(sqlDataReader.GetValue(0).ToString());
        }
        connection.Close();
    }
}

using (SqlConnection connection = new SqlConnection(bgl.Adres))
{
    using (SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT
* FROM Nobetler WHERE personel = '" + personelId + "'", connection))
    {
        using (DataTable dataTable = new DataTable())
        {
            connection.Open();

            sqlDataAdapter.Fill(dataTable);

            dataGridView1.DataSource = dataTable;

            connection.Close();
        }
    }
}

private void NöbetEkle_Load_1(object sender, EventArgs e)
{
    using (SqlConnection connection = new SqlConnection(bgl.Adres))
    {
        using (SqlCommand sqlCommand = new SqlCommand("SELECT Ad FROM
Personeller", connection))
        {
            connection.Open();

            using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
            {
                while (sqlDataReader.Read())
                {
                    cmbPersonel.Items.Add(sqlDataReader.GetValue(0));
                }
            }
            connection.Close();
        }
    }
}

private void NöbetEkle_FormClosed(object sender, FormClosedEventArgs e)
{
    Nöbetİşlemleri nöbetİşlemleri = new Nöbetİşlemleri();
    nöbetİşlemleri.Show();
    this.Hide();
}

private void cmbPersonel_SelectedIndexChanged_1(object sender, EventArgs
e)
{
    var personelId = 0;

```

```

        using (SqlConnection connection = new SqlConnection(bgl.Adres))
        {
            using (SqlCommand sqlCommand = new SqlCommand("SELECT personelId
FROM Personeller WHERE Ad = '" + cmbPersonel.Text + "'", connection))
            {
                connection.Open();

                using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
                {
                    while (sqlDataReader.Read())
                    {
                        personelId =
Convert.ToInt32(sqlDataReader.GetValue(0).ToString());
                    }

                    connection.Close();
                }
            }

            using (SqlConnection connection = new SqlConnection(bgl.Adres))
            {
                using (SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT
* FROM Nobetler WHERE personel = '" + personelId + "'", connection))
                {
                    using (DataTable dataTable = new DataTable())
                    {
                        connection.Open();

                        sqlDataAdapter.Fill(dataTable);

                        dataGridView1.DataSource = dataTable;

                        connection.Close();
                    }
                }
            }
        }

private void btnOlustur_Click_1(object sender, EventArgs e)
{
    var personel = 0;

    using (SqlConnection connection = new SqlConnection(bgl.Adres))
    {
        using (SqlCommand sqlCommand = new SqlCommand("SELECT personelId
FROM Personeller WHERE Ad = '" + cmbPersonel.Text + "'", connection))
        {
            connection.Open();

            using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
            {
                while (sqlDataReader.Read())
                {
                    personel =
Convert.ToInt32(sqlDataReader.GetValue(0).ToString());
                }
            }

            connection.Close();
        }
    }
}

```

```

    }

    var tarih = dtpTarih.Value.Date;

    string turkceTarih = tarih.ToString(" dd MMMM yyyy, dddd", new
CultureInfo("tr-TR"));

    if (cmbPersonel.SelectedItem == null || cmbKonum.SelectedItem == null
|| cmbSaat.SelectedItem == null)
    {
        MessageBox.Show("Boş Alanları Doldurunuz!");
    }
    else
    {
        using (SqlConnection connection = new SqlConnection(bgl.Adres))
        {
            using (SqlCommand komut = new SqlCommand("SELECT * FROM
Nobetler WHERE Tarih = '" + tarih + "' AND personel = '" + personel + "'",
connection))
            {
                connection.Open();

                using (SqlDataReader oku = komut.ExecuteReader()) //Veri
tabanında Zaten Olan Tarihi Kontrol Etme
                {
                    if (oku.Read())
                    {
                        MessageBox.Show("Bu Tarihte Zaten Nöbet
Kayıtlı!", this.ProductName, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    }
                    else
                    {
                        oku.Close();

                        using (SqlCommand sqlCommand = new
SqlCommand("INSERT INTO Nobetler VALUES (@personel,@tarih,@konum,@saat)",
connection))
                        {
                            sqlCommand.Parameters.AddWithValue("@personel", personel);
                            sqlCommand.Parameters.AddWithValue("@tarih",
turkceTarih);
                            sqlCommand.Parameters.AddWithValue("@konum",
cmbKonum.Text);
                            sqlCommand.Parameters.AddWithValue("@saat",
cmbSaat.Text);

                            sqlCommand.ExecuteNonQuery();

                            MessageBox.Show("Nöbet eklendi!",
this.ProductName, MessageBoxButtons.OK, MessageBoxIcon.Information);

                            connection.Close();

                            VeriGetir();
                        }
                    }
                }
                connection.Close();
            }
        }
    }
}

```

```

    }
}

```

9. Nöbet Düzenle Form Kod Bloğu

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static Vardiya_Otomasyon_Programı.Program;

namespace Vardiya_Otomasyon_Programı
{
    public partial class NöbetDüzenle : Form
    {
        public NöbetDüzenle()
        {
            InitializeComponent();

            bağlantı bgl = new bağlantı();

            void VeriGetir() //Güncellenen Verileri Getiren Method
            {
                var personelId = 0;

                using (SqlConnection connection = new SqlConnection(bgl.Adres))
                //Seçilen Personelin ID'sini Bulma
                {
                    using (SqlCommand sqlCommand = new SqlCommand("SELECT personelId
FROM Personeller WHERE Ad = '" + cmbPersonel.Text + "'", connection))
                    {
                        connection.Open();

                        using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
                        {
                            while (sqlDataReader.Read())
                            {
                                personelId =
Convert.ToInt32(sqlDataReader.GetValue(0).ToString());
                            }
                        }

                        connection.Close();
                    }
                }

                using (SqlConnection connection = new SqlConnection(bgl.Adres))
                //Seçilen personelin Nöbet listesini Getir
                {
                    using (SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT
* FROM Nobetler WHERE personel = '" + personelId + "'", connection))
                    {
                        using (DataTable dataTable = new DataTable())
                        {

```

```

        connection.Open();

        sqlDataAdapter.Fill(dataTable);

        dataGridView1.DataSource = dataTable;

        connection.Close();
    }
}

private void NöbetDüzenle_Load(object sender, EventArgs e)
//Sayfa Açıldığında neler gerçekleşsin
{
    using (SqlConnection connection = new SqlConnection(bgl.Adres))
    //Personel isimlerini comboboxa getirmes
    {
        using (SqlCommand sqlCommand = new SqlCommand("SELECT Ad FROM
Personeller", connection))
        {
            connection.Open();

            using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
            {
                while (sqlDataReader.Read())
                {
                    cmbPersonel.Items.Add(sqlDataReader.GetValue(0));
                }
            }
            connection.Close();
        }
    }
}

private void cmbPersonel_SelectedIndexChanged(object sender, EventArgs e)
//comboboxtaki personel seçildiğinde gerçekleşenler
{
    var personelId = 0;

    using (SqlConnection connection = new SqlConnection(bgl.Adres))
    //seçtiğimiz personelin İDSini bulma
    {
        using (SqlCommand sqlCommand = new SqlCommand("SELECT personelİd
FROM Personeller WHERE Ad = '" + cmbPersonel.Text + "'", connection))
        {
            connection.Open();

            using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
            {
                while (sqlDataReader.Read())
                {
                    personelId =
Convert.ToInt32(sqlDataReader.GetValue(0).ToString());
                }
            }
            connection.Close();
        }
    }
}

```

```

        using (SqlConnection connection = new SqlConnection(bgl.Adres))
//Seçtiğimiz personelin Nöbet Listesini Getir
        {
            using (SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT
* FROM Nobetler WHERE personel = '" + personelId + "'", connection))
            {
                using (DataTable dataTable = new DataTable())
                {
                    connection.Open();

                    sqlDataAdapter.Fill(dataTable);

                    dataGridView1.DataSource = dataTable;

                    connection.Close();
                }
            }
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e) //Nöbet Günü Seçildiğinde Konum ve saati
otomatik doldur
        {
            cmbKonum.Text = dataGridView1.CurrentRow.Cells[3].Value.ToString();
            cmbSaat.Text = dataGridView1.CurrentRow.Cells[4].Value.ToString();
        }

        private void btnGuncelle_Click(object sender, EventArgs e)
//Güncelle butonu
        {
            var nobetid =
Convert.ToInt32(dataGridView1.CurrentRow.Cells[0].Value.ToString());

            using (SqlConnection connection = new SqlConnection(bgl.Adres))
//seçilen Konum ve saati Güncelleyen sql sorgusu
            {
                using (SqlCommand sqlCommand = new SqlCommand("UPDATE Nobetler
SET konum = '" + cmbKonum.Text + "', saat = '" + cmbSaat.Text + "' WHERE nobetId
= '" + nobetid + "'", connection)) //Sql Update Komutu
                {
                    connection.Open();
                    sqlCommand.ExecuteNonQuery();
                    MessageBox.Show("Nöbet Güncellendi!", this.ProductName,
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    connection.Close();
                    VeriGetir();
                }
            }
        }

        private void NöbetDüzenle_FormClosed(object sender, FormClosedEventArgs
e) //Form Kapatıldığında Ne Gerçekleşsin
        {
            Nöbetİşlemleri nöbetİşlemleri = new Nöbetİşlemleri();
            nöbetİşlemleri.Show();
            this.Hide();
        }
    }
}

```

10. Nöbet Sil Formu Kod Bloğu

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static Vardiya_Otomasyon_Programı.Program;

namespace Vardiya_Otomasyon_Programı
{
    public partial class NobetSil : Form
    {
        public NobetSil()
        {
            InitializeComponent();

            bağlantı bgl = new bağlantı();

            void VeriGetir() //Yenilenen Listeyi Getiren Method
            {
                var personelId = 0;

                using (SqlConnection connection = new SqlConnection(bgl.Adres))
                //comboboxta seçilen personelin id'sini Alma
                {
                    using (SqlCommand sqlCommand = new SqlCommand("SELECT personelİd
FROM Personeller WHERE Ad = '" + cmbPersonel.Text + "'", connection))
                    {
                        connection.Open();

                        using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
                        {
                            while (sqlDataReader.Read())
                            {
                                personelId =
Convert.ToInt32(sqlDataReader.GetValue(0).ToString());
                            }
                        }

                        connection.Close();
                    }
                }

                using (SqlConnection connection = new SqlConnection(bgl.Adres))
                //Personel İd'sini Aldıktan Sonra Bu idyi Kullanarak o personelin nöbet listesini
                çekmek
                {
                    using (SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT
* FROM Nobetler WHERE personel = '" + personelId + "'", connection))
                    {
                        using (DataTable dataTable = new DataTable())
                        {
                            connection.Open();

                            sqlDataAdapter.Fill(dataTable);
                        }
                    }
                }
            }
        }
    }
}
```

```

        dataGridView1.DataSource = dataTable;

        connection.Close();
    }
}

private void button1_Click(object sender, EventArgs e) //Seçilen Nöbetin
Nöbet idsini Alıp 0 Nöbeti Silen Buton
{
    var nobetid =
Convert.ToInt32(dataGridView1.CurrentRow.Cells[0].Value.ToString());

    using (SqlConnection connection = new SqlConnection(bgl.Adres))
    {
        using (SqlCommand sqlCommand = new SqlCommand("DELETE FROM
Nobetler WHERE nobetId = '" + nobetid + "'", connection))
        {
            connection.Open();
            sqlCommand.ExecuteNonQuery();
            MessageBox.Show("Nöbet Silindi!", this.ProductName,
MessageBoxButtons.OK, MessageBoxIcon.Information);
            connection.Close();
            VeriGetir(); //Yenilenen Verileri Getiren Method
        }
    }
}

private void NobetSil_Load(object sender, EventArgs e) //Combobox'a
Personelleri Getirme
{
    using (SqlConnection connection = new SqlConnection(bgl.Adres))
    {
        using (SqlCommand sqlCommand = new SqlCommand("SELECT Ad FROM
Personeller", connection))
        {
            connection.Open();

            using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
            {
                while (sqlDataReader.Read())
                {
                    cmbPersonel.Items.Add(sqlDataReader.GetValue(0));
                }
            }

            connection.Close();
        }
    }
}

private void cmbPersonel_SelectedIndexChanged(object sender, EventArgs e)
//comboboxta seçilen personelin id'sini Alma
{
    var personelId = 0;

    using (SqlConnection connection = new SqlConnection(bgl.Adres))
    {

```



```

        using (SqlCommand sqlCommand = new SqlCommand("SELECT personelİd
FROM Personeller WHERE Ad = '" + cmbPersonel.Text + "'", connection))
        {
            connection.Open();

            using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
            {
                while (sqlDataReader.Read())
                {
                    personelİd =
Convert.ToInt32(sqlDataReader.GetValue(0).ToString());
                }

                connection.Close();
            }
        }

        using (SqlConnection connection = new SqlConnection(bgl.Adres))
//Personel İd'sini Aldıktan Sonra Bu idyi Kullanarak o personelin nöbet listesini
çekmek
        {
            using (SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT
* FROM Nobetler WHERE personel = '" + personelİd + "'", connection))
            {
                using (DataTable dataTable = new DataTable())
                {
                    connection.Open();

                    sqlDataAdapter.Fill(dataTable);

                    dataGridView1.DataSource = dataTable;

                    connection.Close();
                }
            }
        }

        private void NobetSil_FormClosed(object sender, FormClosedEventArgs e)
//Form Kapatıldığında Nöbet işlemleri Formuna geç
        {
            Nöbetİşlemleri nöbetİşlemleri = new Nöbetİşlemleri();
            nöbetİşlemleri.Show();
            this.Hide();
        }

        private void button2_Click(object sender, EventArgs e) //Seçilen
Personelin Bütün Nöbetlerini Silen Buton
        {
            var personel =
Convert.ToInt32(dataGridView1.CurrentRow.Cells[1].Value.ToString());

            using (SqlConnection connection = new SqlConnection(bgl.Adres))
            {
                using (SqlCommand sqlCommand = new SqlCommand("DELETE FROM
Nobetler WHERE personel = '" + personel + "'", connection))
                {
                    connection.Open();
                    sqlCommand.ExecuteNonQuery();
                    MessageBox.Show("Nöbetler Silindi!", this.ProductName,
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }
        }

```

```

        connection.Close();
        VeriGetir();
    }
}
}
}
}

```

11. Çıktı Al Formu Kod Bloğu

```

using iTextSharp.text.pdf;
using iTextSharp.text;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Globalization;
using static Vardiya_Otomasyon_Programı.Program;

namespace Vardiya_Otomasyon_Programı
{
    public partial class CıktıAl : Form
    {
        public CıktıAl()
        {
            InitializeComponent();

            bağlantı bgl = new bağlantı();

            private void Listele_Click(object sender, EventArgs e) //Nöbet
Programını Listeleyen buton
            {
                var personel = 0;

                using (SqlConnection connection = new SqlConnection(bgl.Adres))
//Seçilen Personelin İdsini Alma
                {
                    using (SqlCommand sqlCommand = new SqlCommand("SELECT personelİd
FROM Personeller WHERE Ad = '" + cmbPersonel.Text + "'", connection))
                    {
                        connection.Open();

                        using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
                        {
                            while (sqlDataReader.Read())
                            {
                                personel =
Convert.ToInt32(sqlDataReader.GetValue(0).ToString());
                            }

                            connection.Close();
                        }
                    }
                }
            }
        }
    }
}

```

```

    }

    DateTime baslangicTarihi = dtpBaslangic.Value; // Başlangıç tarihi
    DateTime bitisTarihi = dtpBitis.Value; // Bitiş tarihi

    string baslangicStr = baslangicTarihi.ToString(" dd MMMM yyyy, dddd",
new System.Globalization.CultureInfo("tr-TR")); //Tarihleri Türkçeye çevirme
    string bitisStr = bitisTarihi.ToString(" dd MMMM yyyy, dddd", new
System.Globalization.CultureInfo("tr-TR"));

    // Veritabanından nöbetleri çekmek için SQL sorgusu
    string query = @"SELECT p.Ad AS PersonelAdi, n.Tarih, n.Konum, n.Saat
FROM Nobetler n INNER JOIN Personeller p ON n.personel = p.personelId WHERE
n.personel = @PersonelID AND n.Tarih BETWEEN @BaslangicTarihi AND @BitisTarihi";

    using (SqlConnection connection = new SqlConnection(bgl.Adres))
    {
        using (SqlCommand command = new SqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@PersonelID", personel);
            command.Parameters.AddWithValue("@BaslangicTarihi",
baslangicStr);
            command.Parameters.AddWithValue("@BitisTarihi", bitisStr);

            using (SqlDataAdapter adapter = new SqlDataAdapter(command))
            {
                DataTable dataTable = new DataTable();
                adapter.Fill(dataTable);

                dataGridView1.DataSource = dataTable; // DataGridView'e
verileri yükle
            }
        }
    }

    private void button1_Click(object sender, EventArgs e) //Çıktı
Almaya Yarayan Buton
    {
        if (dataGridView1.Rows.Count > 0)
        {
            SaveFileDialog save = new SaveFileDialog();
            save.Filter = "PDF (*.pdf)|*.pdf";
            save.FileName = "Result.pdf";
            bool ErrorMessage = false;
            if (save.ShowDialog() == DialogResult.OK)
            {
                if (File.Exists(save.FileName))
                {
                    try
                    {
                        File.Delete(save.FileName);
                    }
                    catch (Exception ex)
                    {
                        ErrorMessage = true;
                        MessageBox.Show("Veriler diske yazılamıyor!" +
ex.Message);
                    }
                }
            }
        }
    }

```

```

        if (!ErrorMessage)
        {
            try
            {
                PdfPTable pTable = new
PdfPTable(dataGridView1.Columns.Count);
                pTable.DefaultCell.Padding = 2;
                pTable.WidthPercentage = 100;
                pTable.HorizontalAlignment = Element.ALIGN_LEFT;
                foreach (DataGridViewColumn col in
dataGridView1.Columns)
                {
                    PdfPCell pCell = new PdfPCell(new
Phrase(col.HeaderText));
                    pTable.AddCell(pCell);
                }
                foreach (DataGridViewRow viewRow in
dataGridView1.Rows)
                {
                    foreach (DataGridViewCell dcell in viewRow.Cells)
                    {
                        if (dcell.Value != null) // Null kontrolü
                        {
                            pTable.AddCell(dcell.Value.ToString());
                        }
                        else
                        {
                            pTable.AddCell(""); // Boş hücreye
varsayılan değer ekleme
                        }
                    }
                }
            }
            using (FileStream fileStream = new
FileStream(save.FileName, FileMode.Create))
            {
                Document document = new Document(PageSize.A4, 8f,
16f, 16f, 8f);

                PdfWriter.GetInstance(document, fileStream);
                document.Open();
                document.Add(pTable);
                document.Close();
                fileStream.Close();
            }
            MessageBox.Show("Yazdırma başarılı!",
this.ProductName, MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Veriler aktarılırken hata oluştu!" +
ex.Message);
        }
    }
}
else
{
    MessageBox.Show("Kayıt bulunamadı!", this.ProductName,
MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
}

private void CıktıAl_Load(object sender, EventArgs e) //Sayfa
Açıldığında gerçekleştirenler
{

```

```

        using (SqlConnection connection = new SqlConnection(bgl.Adres))
//Personelleri Comboboxa Ekleme
        {
            using (SqlCommand sqlCommand = new SqlCommand("SELECT Ad FROM
Personeller", connection))
            {
                connection.Open();

                using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
                {
                    while (sqlDataReader.Read())
                    {
                        cmbPersonel.Items.Add(sqlDataReader.GetValue(0));
                    }
                }

                connection.Close();
            }
        }
    }
}

```

12. Kullanıcı Ekranı Formu Kod Bloğu

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Drawing.Printing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;
using iTextSharp.text;
using iTextSharp.text.pdf;
using static Vardiya_Otomasyon_Programı.Program;

namespace Vardiya_Otomasyon_Programı
{
    public partial class KullanıcıEkran : Form
    {
        public KullanıcıEkran(string tc)
        {
            InitializeComponent();
            this.tc = tc; //Giris Ekranından Tcyi çekiyoruz
        }

        bağlantı bgl = new bağlantı();

        private void KullanıcıEkran_FormClosed(object sender, FormClosedEventArgs
e)
        {
            //Form Kapatıldığında Neler Gerçekleşsin
            {
                Giris giris = new Giris();
                giris.Show();
            }
        }
    }
}

```

```

public string tc = "";
private void KullanıcıEkran_Load(object sender, EventArgs e)
{
    var personelId = 0;

    using (SqlConnection connection = new SqlConnection(bgl.Adres))
    //Giriş Ekranında Tcsi Alınan Personelin ID'sini Bulma
    {
        using (SqlCommand sqlCommand = new SqlCommand("SELECT personelİd
FROM Personeller WHERE Tc = '" + tc + "'", connection))
        {
            connection.Open();

            using (SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader())
            {
                while (sqlDataReader.Read())
                {
                    personelId =
Convert.ToInt32(sqlDataReader.GetValue(0).ToString());
                }
            }

            connection.Close();
        }
    }

    using (SqlConnection connection = new SqlConnection(bgl.Adres))
    // Tc'ye ait nöbetleri listele
    {
        using (SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT
p.Ad AS Personelİsim, n.Tarih,n.Konum,n.Saat FROM Nobetler n INNER JOIN
Personeller p ON n.personel = p.personelİd WHERE personel = '" + personelId +
"'", connection))
        {
            using (DataTable dataTable = new DataTable())
            {
                connection.Open();

                sqlDataAdapter.Fill(dataTable);

                dataGridView1.DataSource = dataTable;

                connection.Close();
            }
        }
    }
}

private void btnCıktı_Click_1(object sender, EventArgs e)
//Listenin Çıktısını alma
{
    if (dataGridView1.Rows.Count > 0)
    {
        SaveFileDialog save = new SaveFileDialog();
        save.Filter = "PDF (*.pdf)|*.pdf";
        save.FileName = "Result.pdf";
        bool ErrorMessage = false;
        if (save.ShowDialog() == DialogResult.OK)
        {
            if (File.Exists(save.FileName))

```

```

        {
            try
            {
                File.Delete(save.FileName);
            }
            catch (Exception ex)
            {
                ErrorMessage = true;
                MessageBox.Show("Veriler diske yazılamıyor!" +
ex.Message);
            }
        }
        if (!ErrorMessage)
        {
            try
            {
                PdfPTable pTable = new
PdfPTable(dataGridView1.Columns.Count);
                pTable.DefaultCell.Padding = 2;
                pTable.WidthPercentage = 100;
                pTable.HorizontalAlignment = Element.ALIGN_LEFT;
                foreach (DataGridViewColumn col in
dataGridView1.Columns)
                {
                    PdfPCell pCell = new PdfPCell(new
Phrase(col.HeaderText));
                    pTable.AddCell(pCell);
                }
                foreach (DataGridViewRow viewRow in
dataGridView1.Rows)
                {
                    foreach (DataGridViewCell dcell in viewRow.Cells)
                    {
                        if (dcell.Value != null) // Null kontrolü
                        {
                            pTable.AddCell(dcell.Value.ToString());
                        }
                        else
                        {
                            pTable.AddCell(""); // Boş hücreye
varsayılan değer ekleme
                        }
                    }
                }
                using (FileStream fileStream = new
FileStream(save.FileName, FileMode.Create))
                {
                    Document document = new Document(PageSize.A4, 8f,
16f, 16f, 8f);
                    PdfWriter.GetInstance(document, fileStream);
                    document.Open();
                    document.Add(pTable);
                    document.Close();
                    fileStream.Close();
                }
                MessageBox.Show("Yazdırma başarılı!",
this.ProductName, MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Veriler aktarılırken hata oluştu!" +
ex.Message);
            }
        }
    }
}

```

```
        }  
    }  
    else  
    {  
        MessageBox.Show("Kayıt bulunamadı!", this.ProductName,  
        MessageBoxButtons.OK, MessageBoxIcon.Warning);  
    }  
}  
}
```

Kaynakça

<https://stackoverflow.com>

<https://www.codeproject.com/Lounge.aspx>

<https://github.com>