Appendence

Landing Page

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Messaging;
namespace pinnotedeskup
{
   public partial class giris : Form
   {
      public giris()
      {
         InitializeComponent();
      }
      private void button1_Click(object sender, EventArgs e)
      {
         kayit_ekran_2 formm2 = new kayit_ekran_2();
         formm2.Show();
         this.Hide();
      }
      private void button2_Click(object sender, EventArgs e)
      {
         hesaba_giris_ekranı_3 formm3 = new hesaba_giris_ekranı_3();
         formm3.Show();
         this.Hide();
      }
   }
}
```

Register Page

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static pinnotedeskup.Program;
using System.Text.RegularExpressions;
using System.Data.SqlClient;
namespace pinnotedeskup
{
   public partial class kayit_ekran_2 : Form
   {
      //DEĞİŞKEN ATAMALARI
      string pas_ctrl = @"^(?=.*[0-9])(?=.*[\W_])(?!.*\s).{8,}$";
      string tel_ctrl = @"^(?:\+?90|0)?[ -]?\d{3}[ -]?\d{3}[ -]?\d{4}$";
      bool name_ctrl = true;
      bool surname_ctrl = true;
      bool email_ctrl = true;
      bool password_ctrl = true;
      bool telnum_ctrl = true;
      bool birth_ctrl = true;
      private DateTime dateTimePicker1_Value;
```

```
    SqlConnection baglanti = new SqlConnection("Data Source = SEDEN\\SQLEXPRESS; Initial
Catalog = PinNoteDB; Integrated Security = True; Encrypt=False;");
    public kayit_ekran_2()
    {
      InitializeComponent();
    }
    private void button1_Click(object sender, EventArgs e)
    {
      if (string.IsNullOrWhiteSpace(name_box.Text))
      {
        MessageBox.Show("İsim boş bırakılamaz");
        name_box.Focus();
        return;
      }
      if (!Regex.IsMatch(name_box.Text, @"^[a-zA-Z]+\s*[a-zA-Z]*$"))
      {
        MessageBox.Show("İsim özel karakter veya boşluk içeremez");
        name_box.Text = "";
        name_box.Focus();
        return;
      }
      // Soyisim kontrolü
      if (string.IsNullOrWhiteSpace(surname_box.Text))
      {
        MessageBox.Show("Soyisim boş bırakılamaz");
        surname_box.Focus();
        return;
      }
      if (!Regex.IsMatch(surname_box.Text, @"^[a-zA-Z]+\s*[a-zA-Z]*$"))
      {
        MessageBox.Show("Soyisim özel karakter ve boşluk içeremez");
        surname_box.Text = "";
        surname_box.Focus();
        return;
      }
      // Şifre kontrolü
      if (string.IsNullOrWhiteSpace(pass_box.Text))
      {
        MessageBox.Show("Şifre boş bırakılamaz");
        pass_box.Focus();
        return;
      }
      if (!Regex.IsMatch(pass_box.Text, pas_ctrl))
      {
        MessageBox.Show("Parola uygun değil");
        pass_box.Text = "";
        pass_box.Focus();
        return;
      }
      // E-posta kontrolü
      if (string.IsNullOrWhiteSpace(mail_box.Text))
      {
        MessageBox.Show("E-posta boş bırakılamaz");
        mail_box.Focus();
        return;
      }
      //girilen veriler uygunsa db kontrolü yap
      if (name_ctrl && surname_ctrl && email_ctrl && password_ctrl && telnum_ctrl &&
birth_ctrl)
      {
```

```
            //mail db da mevcut mu
            string sorgu = "SELECT COUNT(*) FROM USER_DATA WHERE E_MAIL = @Email";
            SqlCommand command = new SqlCommand(sorgu, baglanti);
            command.Parameters.AddWithValue("@Email", mail_box.Text);
            try
            {
               baglanti.Open();
               int count = (int)command.ExecuteScalar();
               if (count > 0)
               {
                  email_ctrl = false;
                  MessageBox.Show("mail daha önceden kullanılmış");
                  name_box.Text = ""; mail_box.Focus();
               }
               else
               {
                  email_ctrl = true;
               }
            }
            catch (Exception ex)
            {
               MessageBox.Show("Error: " + ex.Message);
            }
            //tel no db mevcut mu
            string sorgu2 = "SELECT COUNT(*) FROM USER_DATA WHERE PHONE = @Phone";
            SqlCommand command1 = new SqlCommand(sorgu2, baglanti);
            command1.Parameters.AddWithValue("@Phone", phone_num_box.Text);
            try
            {
               baglanti.Open();
               int count = (int)command1.ExecuteScalar();

               if (count > 0)
               {
                  telnum_ctrl = false;
                  MessageBox.Show("telefon numarası daha önceden kullanılmış");
                  phone_num_box.Text = "";
                  phone_num_box.Focus();
               }
               else
               {
                  if (Regex.IsMatch(phone_num_box.Text, tel_ctrl)) { telnum_ctrl = true; }//tel no özel
karakter içeriyo mu
                  else { MessageBox.Show("Phone number is incorrect or missing");
phone_num_box.Text = ""; phone_num_box.Focus(); }
               }
            }
            catch (Exception ex)
            {
               Console.WriteLine("Error: " + ex.Message);
            }
            //birth kontrol
            if (dateTimePicker1_Value < DateTime.Today) { birth_ctrl = true; }
            else
            {
               MessageBox.Show("Date of birth cannot be in the future tense", "Invalid birth date");
            }
            //bilgilerin uygun olduğu durumda kayıt oluştur ve ana sayfaya yönlendir
            if (name_ctrl && surname_ctrl && email_ctrl && password_ctrl && telnum_ctrl &&
birth_ctrl)
```

```
            {
                user_data.Name = name_box.Text;
                user_data.Surname = surname_box.Text;
                user_data.Email = mail_box.Text;
                user_data.Password = pass_box.Text;
                user_data.PhoneNumber = phone_num_box.Text;
                user_data.BirthDate = birth.Value;
                //tabloya kullanıcı bilgilerini ekle
                string insertdata = "INSERT INTO USER_DATA (NAME, SURNAME, MAIL, PHONE,
BIRTH_DATE, REGISTER_DATE) VALUES (@NAME, @SURNAME, @MAIL, @PHONE,
@BIRTH, @REGISTER)";
                using (SqlCommand command2 = new SqlCommand(insertdata, baglanti))
                {
                    command2.Parameters.AddWithValue("@NAME", user_data.Name);
                    command2.Parameters.AddWithValue("@SURNAME", user_data.Surname);
                    command2.Parameters.AddWithValue("@MAIL", user_data.Email);
                    command2.Parameters.AddWithValue("@PHONE", user_data.PhoneNumber);
                    command2.Parameters.AddWithValue("@BIRTH", user_data.BirthDate);
                    command2.Parameters.AddWithValue("@REGISTER", DateTime.Now);
                    command2.ExecuteNonQuery();
                }
                //TABLOYA EKLENDİKTEN SONRA ID BİLGİSİNİ ÇEK
                string idcekme = "SELECT ID FROM USER_DATA WHERE MAIL = @Email";
                using (SqlCommand command3 = new SqlCommand(idcekme, baglanti))
                {
                    command3.Parameters.AddWithValue("@Email", user_data.Email);
                    user_data.Id = Convert.ToInt32(command3.ExecuteScalar());
                }
                //MessageBox.Show(Convert.ToString(user_data.Id));
                //tabloya password ekleme
                string insertpassword = "INSERT INTO USER_PASSWORD (ID, PASSWORD,
LAST_UPDATE) VALUES (@ID, @PASSWORD, @LAST_UPDATE)";
                using (SqlCommand command3 = new SqlCommand(insertpassword, baglanti))
                {
                    command3.Parameters.AddWithValue("@ID", user_data.Id);
                    command3.Parameters.AddWithValue("@PASSWORD", user_data.Password);
                    command3.Parameters.AddWithValue("@LAST_UPDATE", DateTime.Now);
                    command3.ExecuteNonQuery();
                }
                //yeni kullanıcı için tabloları oluştur

                // Tabloları oluştur

                string note_table = $"NOTES_{user_data.Id}";
                string create_note_table = $"CREATE TABLE {note_table} ({"NOTE_ID INT
PRIMARY KEY, EDIT BIT"})";
                using (SqlCommand command4 = new SqlCommand(create_note_table, baglanti))
                {
                    command4.ExecuteNonQuery();
                }

                string reminder_table = $"REMINDER_{user_data.Id}";
                string create_reminder_table = $"CREATE TABLE {reminder_table} ({"REMINDER_ID
INT PRIMARY KEY, REMINDER_EDIT BIT"})";
                using (SqlCommand command5 = new SqlCommand(create_reminder_table, baglanti))
                {
                    command5.ExecuteNonQuery();
                }
                string friends_table = $"FRIENDS_{user_data.Id}";
```

```
                string create_friends_table = $"CREATE TABLE {friends_table} ({"FriendID INT,
AddedDate DATETIME"})";
                using (SqlCommand command6 = new SqlCommand(create_friends_table, baglanti))
                {
                    command6.ExecuteNonQuery();
                }
                string requests_table = $"REQUESTS_{user_data.Id}";
                string create_requests_table = $"CREATE TABLE {requests_table} ({"RequesterID INT,
RequestDate DATETIME"})";
                using (SqlCommand command7 = new SqlCommand(create_requests_table, baglanti))
                {
                    command7.ExecuteNonQuery();
                }
                //ilk not ve hatırlatıcıyı ekle
                string firstnote_table = $"NOTES_{user_data.Id}";
                string firstnoteadd = $"INSERT {firstnote_table} (NOTE_ID, EDIT) VALUES
(1000003,0)";
                using (SqlCommand command8 = new SqlCommand(firstnoteadd, baglanti))
                {
                    command8.ExecuteNonQuery();
                }
                string firstreminder_table = $"REMINDER_{user_data.Id}";
                string firstreminderadd = $"INSERT {firstreminder_table} (REMINDER_ID,
REMINDER_EDIT)VALUES (1000,0)";
                using (SqlCommand command9 = new SqlCommand(firstreminderadd, baglanti))
                {
                    command9.ExecuteNonQuery();
                }
                string notify_table = $"NOTIFY_{user_data.Id}";
                string notifytable = $@"CREATE TABLE {notify_table} (BILDIRIM_ID INT
IDENTITY(1,1) PRIMARY KEY, PAYLASAN_ID INT NOT NULL, BILDIRIM_DATE
DATETIME NOT NULL, PAYLASILAN_ID INT NOT NULL, PAYLASILAN_TUR CHAR(1)
NOT NULL)";
                using (SqlCommand command10 = new SqlCommand(notifytable, baglanti))
                {
                    command10.ExecuteNonQuery();
                }
                baglanti.Close();
                homepage homepage = new homepage();
                homepage.Show();
                this.Hide();
            }
        }
    }
  }
}
```

Log In Page

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Security.Policy;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static pinnotedeskup.Program;
namespace pinnotedeskup
{
```

```
    public partial class hesaba_giris_ekranı_3 : Form
    {
        /*kullanicidan bilgileri alma*/
        string password="";
        string mail="";
        /*SQL Control*/
        string user_ID;
        string user_psw;
        /*eslesme durumunu kontrol etme*/
        bool isim;
        bool sifre;
        /*hata mesajı icerigi*/
        string warning_message;
        /*SQL baglantisi*/
        bool psw;
        SqlConnection baglanti = new SqlConnection("Data Source = SEDEN\\SQLEXPRESS; Initial Catalog
= PinNoteDB; Integrated Security = True; Encrypt=False;");
        /*DB SADECE  OKUMA MODUNDA AÇ!!!!!!!!!!*/
        public hesaba_giris_ekranı_3()
        {
            InitializeComponent();
            mailbox.Focus();
        }

        private void textBox3_TextChanged(object sender, EventArgs e)
        {
            mailbox.Focus();
            mail= mailbox.Text;
        }
        private void textBox4_TextChanged(object sender, EventArgs e)
        {
            password= textBox4.Text;

        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            baglanti.Open();
            if (mail.Length > 0 && password.Length > 0)
            {
                /*mail control*/

                SqlCommand mail_ctrl = new SqlCommand("SELECT ID FROM USER_DATA WHERE MAIL
= @Mail", baglanti);
                mail_ctrl.Parameters.AddWithValue("@Mail", mail);
                SqlDataReader sqlDataReader_mail = mail_ctrl.ExecuteReader();

                if (sqlDataReader_mail.Read())
                {
                    user_data.Id = Convert.ToInt32(sqlDataReader_mail["ID"]);
                }
                else
                {
                    //hata msj goster
                    MessageBox.Show("wrong mail!");
                    mailbox.Focus();
                    return;
                }
                sqlDataReader_mail.Close();

                /*password control*/
```

```
        SqlCommand psw_ctrl = new SqlCommand("SELECT PASSWORD FROM USER_PASSWORD
WHERE ID = @UserID", baglanti);
        psw_ctrl.Parameters.AddWithValue("@UserID", user_data.Id);
        SqlDataReader sqlDataReader_psw = psw_ctrl.ExecuteReader();

        if (sqlDataReader_psw.Read())
        {
           user_psw = sqlDataReader_psw["PASSWORD"].ToString();//eğer değer varsa atama yapacak

        }
        sqlDataReader_psw.Close();

        /*giris durumu*/
        if (user_psw == password)
        {
           try
           {
              // SQL komutu
              string sql = "SELECT NAME, SURNAME, PHONE, MAIL, BIRTH_DATE, PHOTO
FROM USER_DATA WHERE ID = @id";

              using (SqlCommand vericekme = new SqlCommand(sql, baglanti))
              {
                 // Parametre ekleme
                 vericekme.Parameters.AddWithValue("@id", user_data.Id);

                 using (SqlDataReader reader = vericekme.ExecuteReader())
                 {
                    if (reader.Read())
                    {
                       user_data.Name = reader["NAME"].ToString();
                       user_data.Surname = reader["SURNAME"].ToString();
                       user_data.PhoneNumber = reader["PHONE"].ToString();
                       user_data.Email = reader["MAIL"].ToString();
                       user_data.BirthDate = (DateTime)reader["BIRTH_DATE"];

                       // Fotoğraf okuma, varsa
                       if (reader["PHOTO"] != DBNull.Value)
                       {
                          byte[] photoArray = (byte[])reader["PHOTO"];
                          // Fotoğrafı işlemek için gerekli kodlar burada olabilir
                       }
                    }
                    else
                    {
                       MessageBox.Show("No data found.");
                    }
                 }
              }
           }
           catch (Exception ex)
           {
              MessageBox.Show("An error occurred: " + ex.Message);
           }                  /*homepage yonlendir*/
           homepage homepage = new homepage();
           homepage.Show();
           this.Hide();
        }
        else
        {
           //uyari mesaji goster ve yonlendir
           MessageBox.Show("Wrong password!");
           textBox4.Text = "";
```

```
                textBox4.Focus();
            }
        }
        else
        {
            MessageBox.Show("mail or password can not be empty");
        }
        baglanti.Close();
    }

    private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
    {
        dogrulama_kodu_4 dogrulama = new dogrulama_kodu_4();
        dogrulama.Show();
        this.Hide();
    }
  }

}
```

Verify Page

```
using System;
using System.Data.SqlClient;
using System.Net;
using System.Net.Mail;
using System.Text;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using static pinnotedeskup.Program;

namespace pinnotedeskup
{
  public partial class dogrulama_kodu_4 : Form
  {
    string mail;
    string verify_code;
    string verify_check;

    Random random = new Random();
    public dogrulama_kodu_4()
    {
      InitializeComponent();
    }
    private void button2_Click(object sender, EventArgs e)
    {
      verify_code = verify_maker(6); // 6 karakterlik doğrulama kodu oluştur
      mail =mail_box.Text;
      //email adresi db de mevcut mu kontrol et
      SqlConnection baglanti = new SqlConnection("Data Source = SEDEN\\SQLEXPRESS; Initial
Catalog = PinNoteDB; Integrated Security = True; Encrypt=False;");
      baglanti.Open();
      SqlCommand mail_ctrl = new SqlCommand("SELECT ID FROM USER_DATA WHERE MAIL =
@Mail", baglanti);
      mail_ctrl.Parameters.AddWithValue("@Mail", mail);
      SqlDataReader sqlDataReader_mail = mail_ctrl.ExecuteReader();
      if (sqlDataReader_mail.Read())
      {
        // Doğrulama kodunu yolla
        SendEmail(verify_code, mail_box.Text);
      }
      else
      {
        //hata msj goster
        MessageBox.Show("wrong mail!");
```

```
            }
            sqlDataReader_mail.Close();
        }
        public static string verify_maker(int verify_code_len)
        {
            const string verify_char =
"1234567890qwertyuopasdfghjklizxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM@*-+/!?.-_&%$";
            StringBuilder sb = new StringBuilder();
            Random random = new Random();
            for (int i = 0; i < verify_code_len; i++)
            {
                int idx = random.Next(verify_char.Length);
                sb.Append(verify_char[idx]);}
            return sb.ToString();}

        private void SendEmail(string verify_code, string user_mail)
        {
            string subjectEmail = "Verify Code";
            string bodyEmail = "Your verify code for reset password: " + verify_code;
            using (SmtpClient smtp = new SmtpClient("smtp.outlook.com")) // SMTP sunucusu
            {
                smtp.Port = 587; // SMTP port numarası
                smtp.Credentials = new NetworkCredential("pinnote_app@hotmail.com", "XqWchh6s!Y"); // E-
posta hesabı bilgileri
                smtp.EnableSsl = true; // SSL/TLS
                // E-posta oluştur
                MailMessage mailMessage = new MailMessage("pinnote_app@hotmail.com", user_mail)
                {
                    Subject = subjectEmail,
                    Body = bodyEmail
                };
                try
                {
                    smtp.Send(mailMessage);
                    MessageBox.Show("Mail sent successfully");
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error sending email: " + ex.Message);
                }
            }
        }
        private void verify_code_box_TextChanged(object sender, EventArgs e)
        {
            verify_check = verify_code_box.Text;
        }
        private void button3_Click(object sender, EventArgs e)
        {
            if (verify_code == verify_check)
            {
                sifre_sifirlama_5 sifre_Sifirlama_5 = new sifre_sifirlama_5();
                sifre_Sifirlama_5.Show();
                this.Hide();
            }
        }
    }
}
```

Password Reset

```
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 pinnotedeskup.Program;
namespace pinnotedeskup
{
   public partial class sifre_sifirlama_5 : Form
   {
      public sifre_sifirlama_5()
      {
         InitializeComponent();
      }
      private void psw_button_Click(object sender, EventArgs e)
      {

         if(new_psw1.Text == new_psw2.Text)
         {
            //password içeriğinin uygunluğunu kontrol et
            //db de password olarak psw1 ata
            SqlConnection baglanti = new SqlConnection("Data Source = SEDEN\\SQLEXPRESS; Initial
Catalog = PinNoteDB; Integrated Security = True; Encrypt=False;");
            baglanti.Open();
            SqlCommand VeriGuncelle = new SqlCommand("update  PASSWORD_TABLE
PASSWORD=@password,LAST_UPDATE=@lastupdate where ID=@id", baglanti);
            VeriGuncelle.Parameters.AddWithValue("@id", user_data.Id);
            VeriGuncelle.Parameters.AddWithValue("@password", new_psw1.Text);
            VeriGuncelle.Parameters.AddWithValue("@lastupdate", DateTime.Now);
            baglanti.Close();
            MessageBox.Show("password güncellendi");
         }
         else
         {
            MessageBox.Show("Passwordlar eslesmiyor");
            new_psw1.Text = "";
            new_psw2.Text = "";
            new_psw1.Focus();
         }
         giris giris = new giris();
         giris.Show();
         this.Hide();
      }
   }
}
```

Homepage

```
using pinnotedeskup.wgt;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static pinnotedeskup.Program;
namespace pinnotedeskup
{
   public partial class homepage : Form
   {
```

```
        SqlConnection baglanti = new SqlConnection("Data Source = SEDEN\\SQLEXPRESS; Initial Catalog
= PinNoteDB; Integrated Security = True; Encrypt=False;");
        public homepage()
        {
            InitializeComponent();
            //MessageBox.Show(user_data.Name + user_data.Email);
            //note
            baglanti.Open();
            string table_name = "NOTES_" + user_data.Id;
            List<int> note_id_list = new List<int>();
            List<bool> note_edit_list = new List<bool>();
            //database veri çekme(note)
            notetablewriter();
            //database veri çekme (reminder)
            remindertablewriter();
            //db name çekme
            using (SqlCommand name_ctrl = new SqlCommand("SELECT NAME FROM USER_DATA
WHERE ID = @id", baglanti))
            {
                name_ctrl.Parameters.AddWithValue("@id", user_data.Id);
                using (SqlDataReader sqlDataReader_nameread = name_ctrl.ExecuteReader())
                {
                    while (sqlDataReader_nameread.Read())
                    {
                        string alinan_veri = Convert.ToString(sqlDataReader_nameread["NAME"]);
                        label2.Text = "Merhaba " + alinan_veri + "!";
                    }
                }
            }
            //takvim verisi
            List<int> reminderIds = new List<int>();
            string reminderTable = $"REMINDER_{user_data.Id}";
            using (SqlCommand cmd = new SqlCommand($"SELECT REMINDER_ID FROM
{reminderTable}", baglanti))
            {
                using (SqlDataReader reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        reminderIds.Add(reader.GetInt32(0));
                    }
                }
            }
            foreach (int reminderId in reminderIds)
            {
                using (SqlCommand innerCmd = new SqlCommand("SELECT REMINDER_DATE FROM
REMINDERS WHERE REMINDER_ID = @id", baglanti))
                {
                    innerCmd.Parameters.AddWithValue("@id", reminderId);
                    using (SqlDataReader innerReader = innerCmd.ExecuteReader())
                    {
                        while (innerReader.Read())
                        {
                            DateTime reminderDate = innerReader.GetDateTime(0);
                            monthCalendar1.AddBoldedDate(reminderDate);
                        }
                    }
                }
            }
            monthCalendar1.UpdateBoldedDates();
            //masaustu bildirim
            foreach (int reminderId in reminderIds)
```

```
        {
            using (SqlCommand innerCmd = new SqlCommand("SELECT
REMINDER_DATE,REMINDER_NAME, REMINDER_BODY FROM REMINDERS WHERE
REMINDER_ID = @id", baglanti))
            {
                innerCmd.Parameters.AddWithValue("@id", reminderId);
                using (SqlDataReader innerReader = innerCmd.ExecuteReader())
                {
                    while (innerReader.Read())
                    {
                        DateTime reminderDate = innerReader.GetDateTime(0);
                        string reminderName = innerReader.GetString(1);
                        string reminderbody = innerReader.GetString(2);
                        // Şu anki zaman ile hatırlatıcı tarihi arasındaki fark
                        TimeSpan difference = reminderDate - DateTime.Now;
                        // Eğer fark 0'a eşitse bildirim göster
                        if (difference == TimeSpan.Zero)
                        {
                            // Hatırlatıcı zamanı geldiğinde bildirim göster
                            DialogResult result = MessageBox.Show(reminderName, reminderbody,
MessageBoxButtons.OKCancel);
                            if (result == DialogResult.OK)
                            {
                                // "OK" butonuna tıklanırsa yapılacak işlemler
                                // Örneğin, başka bir formu açabilirsiniz
                                reminder_olusturma_10 rem = new reminder_olusturma_10(reminderId);
                                rem.Show();
                                this.Hide();
                            }
                            else if (result == DialogResult.Cancel)
                            {

                            }
                        }
                    }
                }
            }
        }
    }
    //yan menu
    private void pictureBox2_Click(object sender, EventArgs e)
    {
        //home
        homepage homepage = new homepage();
        homepage.Show();
        this.Hide();
    }
    private void pictureBox4_Click(object sender, EventArgs e)
    {
        //notlar
        notes notes = new notes();
        notes.Show();
        this.Hide();
    }
    private void pictureBox3_Click(object sender, EventArgs e)
    {
        //hatırlatıcı
        reminder_liste reminder_Liste = new reminder_liste();
        reminder_Liste.Show();
        this.Hide();
    }
    private void pictureBox6_Click(object sender, EventArgs e)
    {
```

```
        //arkadaslar
        Friends friends = new Friends();
        friends.Show();
        this.Hide();
    }
    private void pictureBox7_Click(object sender, EventArgs e)
    {
        //profil
        profile profile = new profile();
        profile.Show();
        this.Hide();
    }
    private void pictureBox1_Click(object sender, EventArgs e)
    {
        //home
        homepage homepage1 = new homepage();
        homepage1.Show();
        this.Hide();
    }
    //
    private void button1_Click_1(object sender, EventArgs e)
    {
        //bildirimler
    }
    private void remindergrid_CellContentClick(object sender, DataGridViewCellEventArgs e)
    {
        if (remindergrid.Columns[e.ColumnIndex] is DataGridViewButtonColumn && e.RowIndex >= 0)
        {
            try
            {
                int reminderIdColumnIndex = remindergrid.Columns["ReminderID"].Index;
                if
(!int.TryParse(remindergrid.Rows[e.RowIndex].Cells[reminderIdColumnIndex].Value.ToString(), out int
reminderId))
                {
                    MessageBox.Show("Reminder ID could not be parsed.");
                    return;
                }
                if (remindergrid.Columns[e.ColumnIndex].Name == "rem_edit")
                {
                    EditReminder(reminderId);

                }
                else if (remindergrid.Columns[e.ColumnIndex].Name == "delete_rem")
                {
                    DeleteReminder(reminderId);
                    remindertablewriter();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("An error occurred: " + ex.Message);
            }
        }
    }
    private void EditReminder(int reminderId)
    {
        reminder_olusturma_10 reminder_Olusturma_10 = new reminder_olusturma_10(reminderId);
        reminder_Olusturma_10.Show();
        this.Hide();
    }
    private void DeleteReminder(int reminderId)
    {
```

```
            string tablename = $"REMINDER_{user_data.Id}";
            string sqlsorgustr = $"DELETE FROM {tablename} WHERE REMINDER_ID = @delete";
            try
            {
                using (SqlCommand cmd = new SqlCommand(sqlsorgustr, baglanti))
                {
                    cmd.Parameters.AddWithValue("@delete", reminderId);
                    cmd.ExecuteNonQuery();
                }
                this.Refresh();
                MessageBox.Show("Reminder deleted successfully.");
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Failed to delete reminder: {ex.Message}");
            }
        }
        private void notegrid_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {
            if (notegrid.Columns[e.ColumnIndex] is DataGridViewButtonColumn && e.RowIndex >= 0)
            {
                try
                {
                    int noteIdColumnIndex = notegrid.Columns["NoteID"].Index;
                    if (!int.TryParse(notegrid.Rows[e.RowIndex].Cells[noteIdColumnIndex].Value.ToString(), out
int noteId))
                    {
                        MessageBox.Show("Note ID could not be parsed.");
                        return;
                    }

                    if (notegrid.Columns[e.ColumnIndex].Name == "edit")
                    {
                        EditNote(noteId);
                        this.Refresh();
                    }
                    else if (notegrid.Columns[e.ColumnIndex].Name == "delete")
                    {
                        DeleteNote(noteId);
                        notetablewriter();
                        this.Refresh();
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("An error occurred: " + ex.Message);
                }
            }
        }
        private void EditNote(int noteId)
        {
            newnote newNoteForm = new newnote(noteId);
            newNoteForm.Show();
            this.Hide();
        }
        private void DeleteNote(int noteId)
        {
            string tablename = $"NOTES_{user_data.Id}";
            string sqlsorgustr = $"DELETE FROM {tablename} WHERE NOTE_ID = @delete";
            try
            {
                using (SqlCommand cmd = new SqlCommand(sqlsorgustr, baglanti))
                {
```

```
                cmd.Parameters.AddWithValue("@delete", noteId);
                cmd.ExecuteNonQuery();
                MessageBox.Show("Note deleted successfully.");
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Failed to delete note: {ex.Message}");
        }
    }
    private void notetablewriter ()
    {
        string notetablename = "NOTES_"+user_data.Id;
        List<int> note_id_list = new List<int>();
        List<bool> note_edit_list = new List<bool>();
        using (SqlCommand noteid_ctrl = new SqlCommand($"SELECT NOTE_ID, EDIT FROM
{notetablename}", baglanti))
        {
            using (SqlDataReader sqlDataReader_noteid = noteid_ctrl.ExecuteReader())
            {
                while (sqlDataReader_noteid.Read())
                {
                    note_id_list.Add(sqlDataReader_noteid.GetInt32(0));
                    note_edit_list.Add(sqlDataReader_noteid.GetBoolean(1));
                }
                sqlDataReader_noteid.Close();
            }
        }
        //note tablosundan verileri çekme
        foreach (int note_id in note_id_list)
        {
            SqlCommand table_read = new SqlCommand("SELECT NOTE_TITLE, NOTE_BODY,
UPDATE_TIME FROM NOTES WHERE NOTEID = @id", baglanti);
            table_read.Parameters.AddWithValue("@id", note_id);
            SqlDataReader sqlDataReader_tableread = table_read.ExecuteReader();
            while (sqlDataReader_tableread.Read())
            {
                string title = sqlDataReader_tableread.GetString(0);
                string body = sqlDataReader_tableread.GetString(1);
                DateTime update = sqlDataReader_tableread.GetDateTime(2);
                notegrid.Rows.Add(note_id, title, body, update, "Edit", "Del");
            }
            sqlDataReader_tableread.Close();
        }
    }
    private void remindertablewriter ()
    {
        //reminder
        string reminder_name = "REMINDER_" + user_data.Id;
        List<int> reminder_id = new List<int>();
        List<bool> reminder_edit = new List<bool>();
        using (SqlCommand reminderid_ctrl = new SqlCommand($"SELECT REMINDER_ID,
REMINDER_EDIT FROM {reminder_name}", baglanti))
        {
            using (SqlDataReader sqlDataReader_reminderid = reminderid_ctrl.ExecuteReader())
            {
                while (sqlDataReader_reminderid.Read())
                {
                    reminder_id.Add(sqlDataReader_reminderid.GetInt32(0));
                    reminder_edit.Add(sqlDataReader_reminderid.GetBoolean(1));
                }
            }
        }
```

```
            //reminder tablosundan verileri çekme
            foreach (int i in reminder_id)
            {
                SqlCommand table_read = new SqlCommand("SELECT REMINDER_NAME,
REMINDER_BODY, REMINDER_DATE FROM REMINDERS WHERE REMINDER_ID = @id",
baglanti);
                table_read.Parameters.AddWithValue("@id", i);
                using (SqlDataReader reader = table_read.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        string title = reader.GetString(0);
                        string body = reader.GetString(1);
                        DateTime hatirlatma = reader.GetDateTime(2);
                        remindergrid.Rows.Add(i, title, body, hatirlatma, "Edit", "Del");
                    }
                }

            }
        }
        private void button1_Click(object sender, EventArgs e)
        {
            newnote newnote = new newnote(null);
            newnote.Show();
            this.Hide();
        }
        private void button2_Click(object sender, EventArgs e)
        {
            reminder_olusturma_10 reminder_Olusturma_10 = new reminder_olusturma_10(null);
            reminder_Olusturma_10 .Show();
            this .Hide();
        }
        private void pictureBox5_Click(object sender, EventArgs e)
        {
            IsteklerEkran isteklerEkran = new IsteklerEkran();
            isteklerEkran.Show();
        }
    }
}
```

Notes List

```
using Microsoft.Graph.Models;
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 pinnotedeskup.Program;
namespace pinnotedeskup
{
    public partial class notes : Form
    {
        SqlConnection baglanti = new SqlConnection("Data Source = SEDEN\\SQLEXPRESS; Initial Catalog
= PinNoteDB; Integrated Security = True; Encrypt=False;");
        public notes()
        {
            InitializeComponent();
            string table_name = "NOTES_" + user_data.Id;
            List<int> note_id = new List<int>();
```

```
        List<bool> edit = new List<bool>();
        //database veri çekme(reminderid)
        baglanti.Open();
        SqlCommand id_ctrl = new SqlCommand($"SELECT Note_ID, EDIT FROM {table_name}",
baglanti);
        SqlDataReader sqlDataReader_noteid = id_ctrl.ExecuteReader();
        while (sqlDataReader_noteid.Read())
        {
            note_id.Add(sqlDataReader_noteid.GetInt32(0));
            edit.Add(sqlDataReader_noteid.GetBoolean(1));
        }
        sqlDataReader_noteid.Close();
        //note tablosundan verileri çekme
        foreach (int id in note_id)
        {
            string query = "SELECT NOTE_TITLE, NOTE_BODY, UPDATE_TIME, IMP_LEVEL,
CREATER_ID FROM NOTES WHERE NOTEID = @NoteId";
            SqlCommand table_read = new SqlCommand(query, baglanti);
            table_read.Parameters.AddWithValue("@NoteId", id);
            SqlDataReader sqlDataReader_tableread = table_read.ExecuteReader();
            string title = "", body = "", creater = "";
            DateTime update = DateTime.Now;
            int imp = 0, createrid = 0;
            if (sqlDataReader_tableread.Read())
            {
                title = sqlDataReader_tableread.GetString(0);
                body = sqlDataReader_tableread.GetString(1);
                update = sqlDataReader_tableread.GetDateTime(2);
                imp = sqlDataReader_tableread.GetInt32(3);
                createrid = sqlDataReader_tableread.GetInt32(4);
            }
            sqlDataReader_tableread.Close();
            if (createrid != 0)
            {
                SqlCommand creater_find = new SqlCommand("SELECT NAME, SURNAME FROM
USER_DATA WHERE ID=@CreatorId", baglanti);
                creater_find.Parameters.AddWithValue("@CreatorId", createrid);

                SqlDataReader read = creater_find.ExecuteReader();
                if (read.Read())
                {
                    string creatername = read.GetString(0);
                    string creatersurname = read.GetString(1);
                    creater = creatername + " " + creatersurname;
                }
                read.Close();
            }
            dataGridView.Rows.Add(id, title, body, update, imp, creater,"Edit","Del");
        }
    }
    private void button1_Click_1(object sender, EventArgs e)
    {
        //oluşturma sayfasına yönlendir
        newnote newnote = new newnote(null);
        newnote.Show();
        this.Close();
    }
    private void pictureBox2_Click_1(object sender, EventArgs e)
    {
        homepage homepage2 = new homepage();
        homepage2.Show();
        this.Hide();
    }
```

```
        private void pictureBox4_Click(object sender, EventArgs e)
        {
            this.Controls.Clear();
            this.InitializeComponent();
        }
        private void pictureBox3_Click_1(object sender, EventArgs e)
        {
            reminder_liste reminder_Liste = new reminder_liste();
            reminder_Liste.Show();
            this.Hide();
        }
        private void pictureBox6_Click_1(object sender, EventArgs e)
        {
            Friends friends = new Friends();
            friends.Show();
            this.Hide();
        }
        private void pictureBox7_Click_1(object sender, EventArgs e)
        {
            profile profile2 = new profile();
            profile2.Show();
            this.Hide();
        }
        private void dataGridView_CellContentClick_1(object sender, DataGridViewCellEventArgs e)
        {
            if (dataGridView.Columns[e.ColumnIndex] is DataGridViewButtonColumn && e.RowIndex >= 0)
            {
                int noteIdColumnIndex = 0;
                int noteId = Convert.ToInt32(dataGridView.Rows[e.RowIndex].Cells[noteIdColumnIndex].Value);
                if (dataGridView.Columns[e.ColumnIndex].Name == "edit")
                {
                    // Düzenleme butonu işlemi
                    newnote newNoteForm = new newnote(noteId);
                    newNoteForm.Show();
                    this.Hide();
                }
                else if (dataGridView.Columns[e.ColumnIndex].Name == "delete")
                {
                    // Silme butonu işlemi
                    string deltablename = "NOTES_" + user_data.Id;
                    try
                    {
                        using (SqlCommand delnote = new SqlCommand($"DELETE FROM {deltablename}
WHERE NOTE_ID=@notid", baglanti))
                        {
                            delnote.Parameters.AddWithValue("@notid", noteId);
                            delnote.ExecuteNonQuery();
                            MessageBox.Show("Note deleted successfully.");
                        }
                        this.Refresh();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Error deleting note: " + ex.Message);
                    }
                }
            }
        }

    }
}
```

Create Note

```
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 pinnotedeskup.Program;
namespace pinnotedeskup
{
   public partial class newnote : Form
   {
      int onemdrc = 1;
      int? note_id=null;
      int newnote_id;
      Dictionary<int, bool> paylasim = new Dictionary<int, bool>();
      SqlConnection baglanti = new SqlConnection("Data Source = SEDEN\\SQLEXPRESS; Initial Catalog
= PinNoteDB; Integrated Security = True; Encrypt=False;");
      public newnote(int? noteid)
      {
         InitializeComponent();
         if (noteid!=null)
         {
            note_id = noteid.Value;
            //not bilgilerini ekrana yazdir
            string sql = $"SELECT NOTE_TITLE, NOTE_BODY, IMP_LEVEL FROM NOTES WHERE
NOTEID=@id ";
            using (SqlCommand sqlCommand = new SqlCommand(sql, baglanti))
            {
               baglanti.Open();
               sqlCommand.Parameters.Add("@id", SqlDbType.Int).Value = note_id.Value;
               SqlDataReader sqlDataReader_noteread = sqlCommand.ExecuteReader();
               while (sqlDataReader_noteread.Read())
               {
                  note_title_box.Text = sqlDataReader_noteread.GetString(0);
                  note_body.Text = sqlDataReader_noteread.GetString(1);
                  int imp = sqlDataReader_noteread.GetInt32(2);
                  if (imp == 1) { comboBox1.SelectedIndex = 0; }
                  else if (imp == 2) { comboBox1.SelectedIndex = 1; }
                  else if (imp == 3) { comboBox1.SelectedIndex = 2; }
                  else if (imp == 4) { comboBox1.SelectedIndex = 3; }
                  else if (imp == 5) { comboBox1.SelectedIndex = 4; }
               }
               baglanti.Close();
            }
         }
         else
         {
            note_body.Text = "";
            note_title_box.Text = "";
         }
      }
      private void button1_Click(object sender, EventArgs e)
      {
         int onemdrc = Convert.ToInt32(comboBox1.SelectedValue);
         if (note_id!=null)
         {
            //değişiklikleri kaydet
```

```
            if (!string.IsNullOrWhiteSpace(note_title_box.Text)
&& !string.IsNullOrWhiteSpace(note_body.Text))
            {
                string updateSql = "UPDATE NOTES SET NOTE_TITLE = @NoteTitle, NOTE_BODY =
@NoteBody, IMP_LEVEL = @ImpLevel, UPDATE_TIME = @UpdateTime WHERE NOTEID =
@NoteId;";
                using (SqlCommand command = new SqlCommand(updateSql, baglanti))
                {
                    // Parametreleri ayarla
                    command.Parameters.AddWithValue("@NoteId", note_id); // Güncellenecek notun ID'si
                    command.Parameters.AddWithValue("@NoteTitle", note_title_box.Text);
                    command.Parameters.AddWithValue("@NoteBody", note_body.Text);
                    command.Parameters.AddWithValue("@ImpLevel", onemdrc); // Önem derecesi
                    command.Parameters.AddWithValue("@UpdateTime", DateTime.Now); // Güncelleme
zamanı
                    baglanti.Open();
                    command.ExecuteNonQuery();
                    baglanti.Close();
                    MessageBox.Show("Degisiklikler kaydedildi.");
                    homepage homepage = new homepage();
                    homepage.Show();
                    this.Hide();
                }
            }
        }
        else
        {
            if (!string.IsNullOrWhiteSpace(note_title_box.Text)
&& !string.IsNullOrWhiteSpace(note_body.Text))
            {
                baglanti.Open();
                //notu notlar tablosuna ekler
                int impLevel = Convert.ToInt32(comboBox1.SelectedValue);
                string insertnote = @"INSERT INTO NOTES (NOTE_TITLE, NOTE_BODY, IMP_LEVEL,
UPDATE_TIME, CREATER_ID) VALUES (@Notetitle, @Notebody, @Imp_level, @Updatetime,
@Createrid);
                SELECT SCOPE_IDENTITY();"; // son identitiy değeri alır
                using (SqlCommand command = new SqlCommand(insertnote, baglanti))
                {
                    command.Parameters.AddWithValue("@Notetitle", note_title_box.Text);
                    command.Parameters.AddWithValue("@Notebody", note_body.Text);
                    command.Parameters.AddWithValue("@Imp_level", impLevel);
                    command.Parameters.AddWithValue("@Updatetime", DateTime.Now);
                    command.Parameters.AddWithValue("@Createrid", user_data.Id);
                    newnote_id = Convert.ToInt32(command.ExecuteScalar());
                }
                //notu kullanıcının tablosuna ekler
                string noteidadd = "NOTES_"+user_data.Id;
                string insertnoteadd = $"INSERT INTO {noteidadd} (NOTE_ID, EDIT) VALUES (@noteid,
@edit)";
                using (SqlCommand sqlnote = new SqlCommand(insertnoteadd, baglanti))
                {
                    sqlnote.Parameters.AddWithValue("@noteid", newnote_id);
                    sqlnote.Parameters.AddWithValue("@edit", true);
                    sqlnote.ExecuteNonQuery();
                    MessageBox.Show("Saved!");
                    homepage homepage1 = new homepage();
                    homepage1.Show();
                    this.Hide();
                }
                baglanti.Close();
            }
```

```
            else { MessageBox.Show("Title or body can not be empty"); }
        }
        //tabloya ekleme
        foreach (var kvp in paylasim)
        {
            string tablenamearkds = "NOTES_" + kvp.Key;
            using (SqlCommand arkdsnot = new SqlCommand($"INSERT INTO {tablenamearkds}
(NOTE_ID, EDIT) VALUES (@noteid, @edit)", baglanti))
            {
                arkdsnot.Parameters.AddWithValue("@noteid", kvp.Key);
                arkdsnot.Parameters.AddWithValue("@edit", kvp.Value);

                baglanti.Open();
                arkdsnot.ExecuteNonQuery();
                baglanti.Close();
            }
        }
    }
    private void pictureBox2_Click(object sender, EventArgs e)
    {
        homepage homepage = new homepage();
        homepage.Show();
        this.Hide();
    }
    private void pictureBox4_Click(object sender, EventArgs e)
    {
        notes notes = new notes();
        notes.Show();
        this.Hide();
    }
    private void pictureBox3_Click(object sender, EventArgs e)
    {
        reminder_liste reminder_Liste = new reminder_liste();
        reminder_Liste.Show();
        this.Hide();
    }
    private void pictureBox6_Click(object sender, EventArgs e)
    {
        Friends friends = new Friends();
        friends.Show();
        this.Hide();
    }
    private void pictureBox7_Click(object sender, EventArgs e)
    {
        profile profile = new profile();
        profile.Show();
        this.Hide();
    }
    private void button2_Click(object sender, EventArgs e)
    {
        int friendctrl = Convert.ToInt32(textBox1.Text);
        bool editper = checkBox1.Checked;
        List<int> friends = new List<int>();
        baglanti.Open();
        string friendstable = "FRIENDS_" + user_data.Id;
        //arkadaşlar listesini çek
        using (SqlCommand sqlfriend = new SqlCommand($"SELECT FriendID FROM
{friendstable}",baglanti))
        {
            friends.Add(Convert.ToInt32(sqlfriend.ExecuteScalar()));
        }
        //yazılan id listede var mı kontrol et
        bool eklimi = friends.Contains(friendctrl);
```

```
            string name = string.Empty;
            string surname = string.Empty;
            if (eklimi)
            {
                //isim verisi çekme
                using (SqlCommand namecekme = new SqlCommand("SELECT NAME, SURNAME FROM
USER_DATA WHERE ID=@id", baglanti))
                {
                    namecekme.Parameters.AddWithValue ("@id", friendctrl);
                    SqlDataReader sqlDataReader_name = namecekme.ExecuteReader();
                    while (sqlDataReader_name.Read())
                    {
                        name = sqlDataReader_name.GetString(0);
                        surname = sqlDataReader_name.GetString(1);
                    }
                }
                paylasim.Add(friendctrl,editper);
                MessageBox.Show("Not paylaşıldı!");
            }
            else
            {
                MessageBox.Show("Arkadaş listenizde bulunmamaktadır.");
            }
            baglanti.Close();
        }
    }
}
```

Reminder List

```
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 pinnotedeskup.Program;
namespace pinnotedeskup
{
    public partial class reminder_liste : Form
    {
        SqlConnection baglanti = new SqlConnection("Data Source = SEDEN\\SQLEXPRESS; Initial Catalog
= PinNoteDB; Integrated Security = True; Encrypt=False;");
        public reminder_liste()
        {
            InitializeComponent();
            string reminderName = "REMINDER_" + user_data.Id;
            List<int> reminderIds = new List<int>();
            List<bool> reminderEdits = new List<bool>();
            baglanti.Open();
            // ID ve edit bilgilerini çekme
            using (SqlCommand command = new SqlCommand($"SELECT REMINDER_ID,
REMINDER_EDIT FROM {reminderName}", baglanti))
            {
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        reminderIds.Add(reader.GetInt32(0));
                        reminderEdits.Add(reader.GetBoolean(1));
                    }
```

```
            }
        }
        // Detayları çekme
        for (int i = 0; i < reminderIds.Count; i++)
        {
            using (SqlCommand detailCommand = new SqlCommand("SELECT REMINDER_NAME,
REMINDER_BODY, REMINDER_IMPLVL, REMINDER_DATE, REMINDER_CREATER FROM
REMINDERS WHERE REMINDER_ID = @id", baglanti))
            {
                detailCommand.Parameters.AddWithValue("@id", reminderIds[i]);
                using (SqlDataReader detailReader = detailCommand.ExecuteReader())
                {
                    while (detailReader.Read())
                    {
                        string title = detailReader.GetString(0);
                        string body = detailReader.GetString(1);
                        int implvl = detailReader.GetInt32(2);
                        DateTime reminderDate = detailReader.GetDateTime(3);
                        int createrId = detailReader.GetInt32(4);
                        dataGridView.Rows.Add(reminderIds[i], title, body, reminderDate, implvl, createrId,
reminderEdits[i], "Del");
                    }
                }
            }
        }
    }
    private void button1_Click_1(object sender, EventArgs e)
    {
        reminder_olusturma_10 newreminder_ = new reminder_olusturma_10(null);
        newreminder_.Show();
        this.Hide();
    }
    private void pictureBox2_Click(object sender, EventArgs e)
    {
        homepage homepage = new homepage();
        homepage.Show();
        this.Hide();
    }
    private void pictureBox4_Click(object sender, EventArgs e)
    {
        notes notes = new notes();
        notes.Show();
        this.Hide();
    }
    private void pictureBox6_Click(object sender, EventArgs e)
    {
        Friends friends = new Friends();
        friends.Show();
        this.Hide();
    }
    private void pictureBox7_Click(object sender, EventArgs e)
    {
        profile profile = new profile();
        profile.Show();
        this.Hide();
    }
    private void dataGridView_CellContentClick(object sender, DataGridViewCellEventArgs e)
    {
        if (dataGridView.Columns[e.ColumnIndex] is DataGridViewButtonColumn && e.RowIndex >= 0)
        {
            // Sütun isimlerine göre işlem türünü kontrol et
            if (dataGridView.Columns[e.ColumnIndex].Name == "edit")
            {
```

```
            // Düzenleme işlemi
            int reminderId = Convert.ToInt32(dataGridView.Rows[e.RowIndex].Cells[0].Value);
            reminder_olusturma_10 reminder_Olusturma_10 = new reminder_olusturma_10(reminderId);
            reminder_Olusturma_10.Show();
            this.Hide();
        }
        else if (dataGridView.Columns[e.ColumnIndex].Name == "delete")
        {
            // Silme işlemi
            int reminderId = Convert.ToInt32(dataGridView.Rows[e.RowIndex].Cells[0].Value);
            string remdeletename = "REMINDER_" + user_data.Id;
            try
            {
                using (SqlCommand delreminder = new SqlCommand($"DELETE FROM {remdeletename}
WHERE REMINDER_ID=@remid", baglanti))
                {
                    delreminder.Parameters.AddWithValue("@remid", reminderId);
                    delreminder.ExecuteNonQuery();
                    MessageBox.Show("Reminder deleted successfully.");
                }
                this.Refresh();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error deleting reminder: " + ex.Message);
            }
        }
    }
  }
}
```

Reminder Create

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Net.Mail;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static pinnotedeskup.Program;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.Button;
using Microsoft.Graph.Models;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using System.Globalization;
namespace pinnotedeskup
{
  public partial class reminder_olusturma_10 : Form
  {
    int onemdrc=1;
    string mailaddress = string.Empty;
    int? reminderID = null;
    List<int> paylasilanlar = new List<int>();
    List<int> addedfriendsids = new List <int>();
    bool duzenleme;
    bool mail;
    List<int> friends = new List<int>();
    string mail_body;
```

```
        List<bool>mailyolla = new List<bool>();
        List<bool> duzenizin = new List<bool>();
        int index;
        SqlConnection baglanti = new SqlConnection("Data Source = SEDEN\\SQLEXPRESS; Initial Catalog
= PinNoteDB; Integrated Security = True; Encrypt=False;");
        public reminder_olusturma_10(int? reminderId)
        {
            baglanti.Open();
            InitializeComponent();
            //reminderid null değilse çalışacak
            if (reminderId != null)
            {
                reminderID = reminderId.Value;
                //not bilgilerini ekrana yazdir
                string sql = $"SELECT REMINDER_NAME, REMINDER_BODY, REMINDER_IMPLVL
FROM REMINDERS WHERE REMINDER_ID=@id ";
                using (SqlCommand sqlCommand = new SqlCommand(sql, baglanti))
                {
                    sqlCommand.Parameters.Add("@id", SqlDbType.Int).Value = reminderID.Value;
                    SqlDataReader sqlDataReader_reminder_read = sqlCommand.ExecuteReader();
                    while (sqlDataReader_reminder_read.Read())
                    {
                        reminder_title_box.Text = sqlDataReader_reminder_read.GetString(0);
                        reminder_box.Text = sqlDataReader_reminder_read.GetString(1);
                        int imp = sqlDataReader_reminder_read.GetInt32(2);

                        if (imp == 1) { comboBox1.SelectedIndex = 0; }
                        else if (imp == 2) { comboBox1.SelectedIndex = 1; }
                        else if (imp == 3) { comboBox1.SelectedIndex = 2; }
                        else if (imp == 4) { comboBox1.SelectedIndex = 3; }
                        else if (imp == 5) { comboBox1.SelectedIndex = 4; }
                    }
                    sqlDataReader_reminder_read.Close();
                }
            }
            else
            {
                reminder_box.Text = "";
                reminder_title_box.Text = "";
                reminderId = null;
                date.Value = DateTime.Now;
                saat.SelectedIndex = 9;
                comboBox1.SelectedIndex = 1;
            }
             //arkadaslar listesini yazdır
            string tablename = "FRIENDS_" + user_data.Id;
            using (SqlCommand arkadaslariyazdir = new SqlCommand($"SELECT FriendID FROM
{tablename}", baglanti))
            {
                SqlDataReader reader = arkadaslariyazdir.ExecuteReader();
                while (reader.Read())
                {
                    friends.Add(reader.GetInt32(0));  // FriendID değerlerini listeye ekle
                }
                reader.Close();
            }
            foreach (int id in friends)
            {
                using (SqlCommand listboxyazdir = new SqlCommand("SELECT NAME, SURNAME, MAIL
FROM USER_DATA WHERE ID=@id", baglanti))
                {
                    listboxyazdir.Parameters.AddWithValue("@id", id);
                    SqlDataReader arkadasisim = listboxyazdir.ExecuteReader();
```

```
                while (arkadasisim.Read())
                {
                    string isim = arkadasisim.GetString(0);
                    string soyisim = arkadasisim.GetString(1);
                    mailaddress = arkadasisim.GetString(2);
                    addfriend.Rows.Add(id, isim, soyisim,false,false,"Add"); // DataGridView'e veri ekler
                }
                arkadasisim.Close();
            }
        }
        baglanti.Close();
    }
    //add friend table
    private void addfriend_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)
    {
        int id = Convert.ToInt32(addfriend.Rows[e.RowIndex].Cells[0].Value);
        int index = 0;
        for (int i = 0; i<addedfriendsids.Count; i++)
        {
            if (addedfriendsids[i] == id)
            {
                index = i; break;
            }
        }
        if (e.RowIndex >= 0 && (e.ColumnIndex == addfriend.Columns["edit"].Index))
        {
           duzenleme = Convert.ToBoolean(addfriend.Rows[e.RowIndex].Cells["edit"].Value);
            duzenizin[index] = duzenleme;
        }
        if (e.ColumnIndex == addfriend.Columns["mail"].Index)
        {
           mail = Convert.ToBoolean(addfriend.Rows[e.RowIndex].Cells["mail"].Value);
            mailyolla[index] = duzenleme;
        }
    }
    private void addfriend_CellContentClick(object sender, DataGridViewCellEventArgs e)
    {
        if (e.RowIndex >= 0 && (e.ColumnIndex == addfriend.Columns["editcheck"].Index ||
e.ColumnIndex == addfriend.Columns["mail_bildirim"].Index))
        {
            addfriend.CommitEdit(DataGridViewDataErrorContexts.Commit);
        }
        // Tıklanan hücrenin buton olup olmadığını ve "add" isimli buton olup olmadığını kontrol et
        if (e.RowIndex >= 0 && e.ColumnIndex == addfriend.Columns["add"].Index)
        {
            try
            {
                int arkdsId = Convert.ToInt32(addfriend.Rows[e.RowIndex].Cells[0].Value);
                // Listede bu ID yoksa ekleme işlemini yap
                if (!addedfriendsids.Contains(arkdsId))
                {
                    string arkdsname = Convert.ToString(addfriend.Rows[e.RowIndex].Cells[1].Value);
                    string arkdssur = Convert.ToString(addfriend.Rows[e.RowIndex].Cells[2].Value);
                    bool edt = Convert.ToBoolean(addfriend.Rows[e.RowIndex].Cells[3].Value);
                    bool m = Convert.ToBoolean(addfriend.Rows[e.RowIndex].Cells[4].Value);
                    // Arkadaş bilgilerini yeni tabloya ekle
                    added_friend.Rows.Add(arkdsId, arkdsname, arkdssur, edt, m, "Add");
                    // ID'yi listeye ekle
                    addedfriendsids.Add(arkdsId) ;
                    duzenizin.Add(edt);
                    mailyolla.Add(m);
                    MessageBox.Show("Arkadaş eklendi: " + arkdsname + " " + arkdssur);
                }
```

```
            else
            {
                MessageBox.Show("Bu arkadaş zaten listeye eklenmiş.");
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Hata oluştu: " + ex.Message);
        }
    }
}
//added friend table
private void added_friend_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    int id = Convert.ToInt32(addfriend.Rows[e.RowIndex].Cells[0].Value);
    for (int i = 0; i < addedfriendsids.Count; i++)
    {
        if (addedfriendsids[i] == id)
        {
            index = i; break;
        }
    }
    // Tıklanan hücrenin buton olup olmadığını ve "del" isimli buton olup olmadığını kontrol et
    if (e.RowIndex >= 0 && e.ColumnIndex == added_friend.Columns["del"].Index)
    {
        try
        {
            addedfriendsids.Remove(addedfriendsids[index]);
            mailyolla.Remove(mailyolla[index]);
            duzenizin.Remove(duzenizin[index]);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Hata oluştu: " + ex.Message);
        }
    }
}
private void added_friend_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    int id = Convert.ToInt32(added_friend.Rows[e.RowIndex].Cells[0].Value);
    for (int i = 0; i < addedfriendsids.Count; i++)
    {
        if (addedfriendsids[i] == id)
        {
            index = i; break;
        }
    }
    if (e.RowIndex >= 0 && (e.ColumnIndex == added_friend.Columns["edit"].Index))
    {
        duzenleme = Convert.ToBoolean(added_friend.Rows[e.RowIndex].Cells["edit"].Value);
        duzenizin[index] = duzenleme;
    }
    if (e.ColumnIndex == added_friend.Columns["mail"].Index)
    {
        mail = Convert.ToBoolean(added_friend.Rows[e.RowIndex].Cells["mail"].Value);
        mailyolla[index] = duzenleme;
    }
}
//yan menu
private void pictureBox2_Click(object sender, EventArgs e)
{
    homepage homepage = new homepage();
    homepage.Show();
```

```
            this.Hide();
        }
        private void pictureBox4_Click(object sender, EventArgs e)
        {
            notes notes = new notes();
            notes.Show();
            this.Hide();
        }
        private void pictureBox3_Click(object sender, EventArgs e)
        {
            reminder_liste reminderlist = new reminder_liste();
            reminderlist.Show();
            this.Hide();
        }
        private void pictureBox6_Click(object sender, EventArgs e)
        {
            Friends friends = new Friends();
            friends.Show();
            this.Hide();
        }
        private void pictureBox7_Click(object sender, EventArgs e)
        {
            profile profile = new profile();
            profile.Show();
            this.Hide();
        }
        //kaydet ve mail yolla
        private void SendEmail(string paylasan, string paylasılanmailadresi, string mailbody)
        {
            baglanti.Open();
            string subjectEmail = paylasan + " share a reminder with you.";
            string bodyEmail = mailbody;
            using (SmtpClient smtp = new SmtpClient("smtp.outlook.com")) // SMTP sunucusu
            {
                smtp.Port = 587; // SMTP port numarası
                smtp.Credentials = new NetworkCredential("pinnote_app@hotmail.com", "XqWchh6s!Y"); // E-
posta hesabı bilgileri
                smtp.EnableSsl = true; // SSL/TLS
                // E-posta oluştur
                MailMessage mailMessage = new MailMessage("pinnote_app@hotmail.com",
paylasılanmailadresi)
                {
                    Subject = subjectEmail,
                    Body = bodyEmail
                };
                try
                {
                    smtp.Send(mailMessage);
                    MessageBox.Show("Mail sent successfully");
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error sending email: " + ex.Message);
                }
            }
            baglanti.Close();
        }
        private void button1_Click_2(object sender, EventArgs e)
        {
            baglanti.Open();
            int onemdrc = Convert.ToInt32(comboBox1.SelectedValue);
```

```
        if (reminderID != null)
        {
            //değişiklikleri kaydet
            if (!string.IsNullOrWhiteSpace(reminder_title_box.Text)
&& !string.IsNullOrWhiteSpace(reminder_box.Text))
            {
                string updateSql = @"UPDATE REMINDERS SET REMINDER_NAME = @remname,
REMINDER_BODY = @remBody, REMINDER_DATE=@remdate, REMINDER_IMPLVL = @ImpLevel
WHERE REMINDER_ID = @remId;";
                using (SqlCommand command = new SqlCommand(updateSql, baglanti))
                {
                    // parametreleri ayarla
                    command.Parameters.AddWithValue("@remId", reminderID);
                    command.Parameters.AddWithValue("@remname", reminder_title_box.Text);
                    command.Parameters.AddWithValue("@remBody", reminder_box.Text);
                    command.Parameters.AddWithValue("@ImpLevel", onemdrc);
                    command.Parameters.AddWithValue("@remdate", date.Value);
                    command.ExecuteNonQuery();
                }
            }
            baglanti.Close();
            MessageBox.Show("Reminder saved successfully.");
        }
        else
        {
            DateTime hatirlatma = date.Value;
            if (!string.IsNullOrWhiteSpace(reminder_title_box.Text)
&& !string.IsNullOrWhiteSpace(reminder_box.Text))
            {
                // Notları REMINDERS tablosuna ekle
                string insertnote = "INSERT INTO REMINDERS (REMINDER_NAME, REMINDER_BODY,
REMINDER_DATE, REMINDER_IMPLVL, REMINDER_CREATER) VALUES (@Notetitle,
@Notebody, @Reminderdate, @Imp_level, @Createrid)";
                using (SqlCommand command = new SqlCommand(insertnote, baglanti))
                {
                    command.Parameters.AddWithValue("@Notetitle", reminder_title_box.Text);
                    command.Parameters.AddWithValue("@Notebody", reminder_box.Text);
                    command.Parameters.AddWithValue("@Reminderdate", hatirlatma);
                    command.Parameters.AddWithValue("@Imp_level", onemdrc);
                    command.Parameters.AddWithValue("@Createrid", user_data.Id);
                    command.ExecuteNonQuery();
                }
                // En son eklenen Reminder'ın ID'sini al
                string getReminderIdQuery = "SELECT IDENT_CURRENT('REMINDERS')";  // Varsayılan
olarak son eklenen ID'yi döner
                int reminderID;
                using (SqlCommand sqlsorgu = new SqlCommand(getReminderIdQuery, baglanti))
                {
                    reminderID = Convert.ToInt32(sqlsorgu.ExecuteScalar()); // ID değerini döndür
                }
                // REMINDER_ID'yi kullanıcının REMINDER tablosuna ekle
                string note_table_name = "REMINDER_" + user_data.Id;
                string insert_note_id = $"INSERT INTO {note_table_name} (REMINDER_ID,
REMINDER_EDIT) VALUES (@Noteid, @Edit)";
                using (SqlCommand command1 = new SqlCommand(insert_note_id, baglanti))
                {
                    command1.Parameters.AddWithValue("@Noteid", reminderID);
                    command1.Parameters.AddWithValue("@Edit", 1);
                    command1.ExecuteNonQuery();
                }
                MessageBox.Show("Reminder saved successfully.");
                //paylasilan arkadaslara reminder ekleme
                try
```

```
            {
                int a = 0;
                foreach (int id in addedfriendsids)
                {
                    string note_table_name1 = $"REMINDER_{id}";
                    string insert_rem_id = $"INSERT INTO {note_table_name1} (REMINDER_ID,
REMINDER_EDIT) VALUES (@remid, @Edit)";
                    using (SqlCommand command1 = new SqlCommand(insert_rem_id, baglanti))
                    {
                        command1.Parameters.AddWithValue("@remid", reminderID);
                        command1.Parameters.AddWithValue("@Edit", duzenizin[a]);
                        command1.ExecuteNonQuery();
                    }
                    if (mailyolla[a])
                    {
                        string paylasan = user_data.Name + " " + user_data.Surname;
                        mail_body = paylasan + " shared a reminder with you.\n" + reminder_title_box.Text +
"\n" + reminder_box.Text + "\n" + date.Value;
                        SendEmail(paylasan, mailaddress, mail_body);
                    }
                    a++;
                }
                homepage homepage = new homepage();
                homepage.Show();
                this.Hide();
            }
            catch (Exception ex)
            {
                Console.WriteLine("An error occurred: " + ex.Message);
            }
        }
        else
        {
            MessageBox.Show("Title or body can not be empty");
        }
    }
    homepage homepage1 = new homepage();
    homepage1.Show();
    this.Hide();
  }
 }
}
```

Friends

```
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 Microsoft.Graph.Models;
using Microsoft.VisualBasic;
using pinnotedeskup.wgt;
using static System.Net.Mime.MediaTypeNames;
using static pinnotedeskup.Program;
namespace pinnotedeskup
{
  public partial class Friends : Form
  {
```

```
        SqlConnection baglanti = new SqlConnection("Data Source = SEDEN\\SQLEXPRESS; Initial Catalog
= PinNoteDB; Integrated Security = True; Encrypt=False;");
        public Friends()
        {
            InitializeComponent();
            baglanti.Open();
            //arkdas id çekme
            string table_name = "FRIENDS_" + user_data.Id;
            List<int> friendsids = new List<int>();
            using (SqlCommand friendslist = new SqlCommand($"SELECT FriendID FROM
{table_name}",baglanti))
            {
                try
                {
                    using(SqlDataReader friends_list =friendslist.ExecuteReader())
                    {
                        while (friends_list.Read())
                        {
                            int friendID = friends_list.GetInt32(0);
                            friendsids.Add(friendID);
                        }
                        friends_list.Close();
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error: " + ex.Message);
                }
            }
            //arkadas bilgilerini çekme
            foreach (int friendID in friendsids)
            {
                using (SqlCommand friendsinfo = new SqlCommand("SELECT NAME, SURNAME, PHOTO
FROM USER_DATA WHERE ID=@ID", baglanti))
                {
                    try
                    {
                        friendsinfo.Parameters.AddWithValue("@ID", friendID);
                        using (SqlDataReader sqlDataReader_friendsinfo = friendsinfo.ExecuteReader())
                        {
                            while (sqlDataReader_friendsinfo.Read())
                            {
                                string name = (sqlDataReader_friendsinfo.GetString(0));
                                string surname = (sqlDataReader_friendsinfo.GetString(1));
                                if (!sqlDataReader_friendsinfo.IsDBNull(2))
                                {
                                    byte[] imageData = (byte[])sqlDataReader_friendsinfo["PHOTO"];
                                    using (MemoryStream ms = new MemoryStream(imageData))
                                    {
                                        System.Drawing.Image profilephoto = System.Drawing.Image.FromStream(ms);
                                        dataGridView1.Rows.Add(profilephoto, name + " " + surname, friendID);
                                    }
                                }
                                else
                                {
                                    dataGridView1.Rows.Add(null, name + " " + surname, friendID);
                                }
                            }
                            sqlDataReader_friendsinfo.Close();
                        }
                    }
                    catch (Exception ex)
                    {
```

```
                MessageBox.Show("Error: " + ex.Message);
            }
        }
    }
}
private void dataGridView1_CellContentClick_1(object sender, DataGridViewCellEventArgs e)
{
    if (dataGridView1.Columns[e.ColumnIndex] is DataGridViewButtonColumn && e.RowIndex >= 0)
    {
        int idColumnIndex = 2;
        int id = Convert.ToInt32(dataGridView1.Rows[e.RowIndex].Cells[idColumnIndex].Value);
        friendprofile friendprofile = new friendprofile(id);
        friendprofile.Show();
        this.Hide();
    }
}
private void button1_Click(object sender, EventArgs e)
{
    //add friends
    string add_id = Interaction.InputBox("Add friends id:", "Add Friend!");
    if (!string.IsNullOrEmpty(add_id))
    {
        // ID'nin USER_DATA tablosunda mevcut mu
        string idvarmi = "SELECT COUNT(1) FROM USER_DATA WHERE ID = @add_id";
        using (SqlCommand checkid = new SqlCommand(idvarmi, baglanti))
        {
            checkid.Parameters.AddWithValue("@add_id", int.Parse(add_id));
            // Kullanıcı ID'sinin var olup olmadığını kontrol edin
            int userExists = (int)checkid.ExecuteScalar();
            if (userExists > 0)
            {
                string tablename = "FRIENDS_" + user_data.Id;
                string eklimi = $"SELECT COUNT(1) FROM {tablename} WHERE FriendID = @add_id";
                using (SqlCommand checkfriend = new SqlCommand(eklimi, baglanti))
                {
                    checkfriend.Parameters.AddWithValue("@add_id", int.Parse(add_id));
                    int ekli = (int)checkfriend.ExecuteScalar();
                    if(ekli <= 0)
                    {
                        // Kullanıcı ID'si mevcutsa arkadaşlık isteğini ekleyin
                        string table_name = "REQUESTS_" + add_id;
                        string insertid = "INSERT INTO " + table_name + " (RequesterID, RequestDate)
VALUES (@myid, @date)";
                        using (SqlCommand addfriend = new SqlCommand(insertid, baglanti))
                        {
                            // Parametreler oluşturulur ve değerleri atanır
                            addfriend.Parameters.AddWithValue("@myid", user_data.Id);
                            addfriend.Parameters.AddWithValue("@date", DateTime.Now);
                            try
                            {
                                addfriend.ExecuteNonQuery();
                                MessageBox.Show("Request sent!");
                            }
                            catch (SqlException ex)
                            {
                                MessageBox.Show("An error occurred while sending the request: " + ex.Message);
                            }
                        }
                    }
                    else
                    {
                        MessageBox.Show("This person is already on your list.");
                    }
```

```
                }
            }
            else
            {
                // Kullanıcı ID'si mevcut değilse hata mesajı göster
                MessageBox.Show("The specified user ID does not exist.");
            }
        }
    }
}
private void pictureBox2_Click_1(object sender, EventArgs e)
{
    homepage homepage = new homepage();
    homepage.Show();
    this.Hide();
}
private void button2_Click(object sender, EventArgs e)
{
    IsteklerEkran isteklerEkran = new IsteklerEkran();
    isteklerEkran.Show();
}
private void pictureBox4_Click_1(object sender, EventArgs e)
{
    notes notes = new notes(); notes.Show(); this.Hide();
}
private void pictureBox3_Click(object sender, EventArgs e)
{
    reminder_liste reminder_Liste = new reminder_liste();
    reminder_Liste.Show();
    this.Hide();
}
private void pictureBox6_Click(object sender, EventArgs e)
{
    Friends friends = new Friends();
    friends.Show();
    this.Hide();
}
private void pictureBox7_Click(object sender, EventArgs e)
{
    profile profile = new profile();
    profile.Show();
    this.Hide();
}
}
}
```

Friend Profile

```
using Microsoft.Graph.Models;
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 static pinnotedeskup.Program;
namespace pinnotedeskup
{
    public partial class friendprofile : Form
    {
```

```
        SqlConnection baglanti = new SqlConnection("Data Source = SEDEN\\SQLEXPRESS; Initial Catalog
= PinNoteDB; Integrated Security = True; Encrypt=False;");
        public friendprofile(int friendID)
        {
            InitializeComponent();
            baglanti.Open();
            //profil foto
            using (SqlCommand friendsinfo = new SqlCommand("SELECT NAME, SURNAME, MAIL,
PHOTO FROM USER_DATA WHERE ID=@ID", baglanti))
            {
                try
                {
                    friendsinfo.Parameters.AddWithValue("@ID", friendID);
                    using (SqlDataReader sqlDataReader_friendsinfo = friendsinfo.ExecuteReader())
                    {
                        while (sqlDataReader_friendsinfo.Read())
                        {
                            textBox1.Text=friendID.ToString();
                            textBox2.Text = (sqlDataReader_friendsinfo.GetString(0));
                            textBox3.Text = (sqlDataReader_friendsinfo.GetString(1));
                            textBox4.Text = sqlDataReader_friendsinfo.GetString(2);
                            if (!sqlDataReader_friendsinfo.IsDBNull(3))
                            {
                                byte[] imageData = (byte[])sqlDataReader_friendsinfo["PHOTO"];
                                using (MemoryStream ms = new MemoryStream(imageData))
                                {
                                    pictureBox1.Image = System.Drawing.Image.FromStream(ms);
                                }
                            }
                        }
                        sqlDataReader_friendsinfo.Close();
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error: " + ex.Message);
                }
            }
            baglanti.Close();
            /*********************************************************/
            //ortak notlar
            /*
            string mytable_name = "NOTES_" + user_data.Id;
            string friendtable_name = "NOTES_" + friendID;
            List<int> noteid1 = new List<int>();
            List<int> noteid2 = new List<int>();
            List<int> ortaknotes = new List<int>();
            using (SqlCommand noteid_ctrl = new SqlCommand($"SELECT NOTE_ID FROM
{mytable_name}", baglanti))
            {
                using (SqlDataReader sqlDataReader_noteid = noteid_ctrl.ExecuteReader())
                {
                    while (sqlDataReader_noteid.Read())
                    {
                        noteid1.Add(sqlDataReader_noteid.GetInt32(0));
                    }
                    sqlDataReader_noteid.Close();
                }
            }
            using (SqlCommand noteid_ctrl = new SqlCommand($"SELECT NOTE_ID FROM
{friendtable_name}", baglanti))
            {
                using (SqlDataReader sqlDataReader_noteid = noteid_ctrl.ExecuteReader())
```

```
            {
                while (sqlDataReader_noteid.Read())
                {
                    noteid2.Add(sqlDataReader_noteid.GetInt32(0));
                }
                sqlDataReader_noteid.Close();
            }
        }
        foreach (int noteid in noteid1)
        {
            if (noteid2.Contains(noteid))
            {
                ortaknotes.Add(noteid);
            }
        }
        //note tablosundan verileri çekme
        foreach (int note_id in ortaknotes)
        {
            SqlCommand table_read = new SqlCommand("SELECT NOTE_TITLE, NOTE_BODY,
UPDATE_TIME FROM NOTES WHERE NOTEID = @id", baglanti);
            table_read.Parameters.AddWithValue("@id", note_id);
            SqlDataReader sqlDataReader_tableread = table_read.ExecuteReader();
            while (sqlDataReader_tableread.Read())
            {
                string title = sqlDataReader_tableread.GetString(0);
                string body = sqlDataReader_tableread.GetString(1);
                DateTime update = sqlDataReader_tableread.GetDateTime(2);
                this.ortaknotes.Rows.Add(note_id, title, body, update);
                MessageBox.Show(Convert.ToString(note_id));
            }
            sqlDataReader_tableread.Close();
        }

        /*********************************************************/
        //ortak hatırlatıcılar
        /*
        string mytable_namerem = "REMINDER_" + user_data.Id;
        string friendtable_namerem = "REMINDER_" + friendID;
        List<int> remid1 = new List<int>();
        List<int> remid2 = new List<int>();
        List<int> ortakrems = new List<int>();

        using (SqlCommand noteid_ctrlrem = new SqlCommand($"SELECT REMINDER_ID FROM
{mytable_namerem}", baglanti))
        {
            using (SqlDataReader sqlDataReader_noteidrem = noteid_ctrlrem.ExecuteReader())
            {
                while (sqlDataReader_noteidrem.Read())
                {
                    remid1.Add(sqlDataReader_noteidrem.GetInt32(0));
                }
                sqlDataReader_noteidrem.Close();
            }
        }
        using (SqlCommand noteid_ctrlrem = new SqlCommand($"SELECT REMINDER_ID FROM
{friendtable_namerem}", baglanti))
        {
            using (SqlDataReader sqlDataReader_noteidrem = noteid_ctrlrem.ExecuteReader())
            {
                while (sqlDataReader_noteidrem.Read())
                {
                    remid2.Add(sqlDataReader_noteidrem.GetInt32(0));
                }
```

```
                sqlDataReader_noteidrem.Close();
            }
        }
        foreach (int id in remid1)
        {
            if (remid2.Contains(id))
            {
                ortakrems.Add(id);
            }
        }
        //note tablosundan verileri çekme
        foreach (int remid in ortakrems)
        {
            SqlCommand table_readrem = new SqlCommand("SELECT NOTE_TITLE, NOTE_BODY,
UPDATE_TIME FROM NOTES WHERE NOTEID = @id", baglanti);
            table_readrem.Parameters.AddWithValue("@id", remid);
            SqlDataReader sqlDataReader_tablereadrem = table_readrem.ExecuteReader();
            while (sqlDataReader_tablereadrem.Read())
            {
                string title = sqlDataReader_tablereadrem.GetString(0);
                string body = sqlDataReader_tablereadrem.GetString(1);
                DateTime update = sqlDataReader_tablereadrem.GetDateTime(2);
                ortakreminders.Rows.Add(remid, title, body, update);
            }
            sqlDataReader_tablereadrem.Close();
        }
        */
    }
    private void pictureBox2_Click(object sender, EventArgs e)
    {
        homepage homepage = new homepage();
        homepage.Show();
        this.Hide();
    }
    private void pictureBox4_Click(object sender, EventArgs e)
    {
        notes notes = new notes();
        notes.Show();
        this.Hide();
    }
    private void pictureBox3_Click(object sender, EventArgs e)
    {
        reminder_liste reminderlist = new reminder_liste();
        reminderlist.Show();
        this.Hide();
    }
    private void pictureBox6_Click(object sender, EventArgs e)
    {
        Friends friends = new Friends();
        friends.Show();
        this.Hide();
    }
    private void pictureBox7_Click(object sender, EventArgs e)
    {
        profile profile = new profile();
        profile.Show();
        this.Hide();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            string tableDeleteName = "FRIENDS_" + user_data.Id;
```

```
            string sql = $"DELETE FROM {tableDeleteName} WHERE FriendID = @id";
            baglanti.Open();
            using (SqlCommand delFriend = new SqlCommand(sql, baglanti))
            {
                // friend_id değişkeninin integer olduğunu varsayarak
                int id = Convert.ToInt32(friend_id.Text);
                // Parametreyi AddWithValue kullanarak ekleyin
                delFriend.Parameters.AddWithValue("@id", id);
                int rowsAffected = delFriend.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
                    MessageBox.Show("Deleted!");
                }
                else
                {
                    MessageBox.Show("No record found with the specified ID.");
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Bir hata oluştu: " + ex.Message);
        }
    }
  }
}
```

Profile

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using static pinnotedeskup.Program;
using System.IO;
namespace pinnotedeskup
{
  public partial class profile : Form
  {
    /*sql baglanti*/
    SqlConnection baglanti = new SqlConnection("Data Source = SEDEN\\SQLEXPRESS; Initial Catalog
= PinNoteDB; Integrated Security = True; Encrypt=False;");
    public profile()
    {
      InitializeComponent();
      user_id_box.Text = Convert.ToString(user_data.Id);
      try
      {
        // SQL komutu
        string sql = "SELECT NAME, SURNAME, PHONE, MAIL, BIRTH_DATE, PHOTO FROM
USER_DATA WHERE ID = @id";
        using (SqlCommand vericekme = new SqlCommand(sql, baglanti))
        {
          // Parametre ekleme
          vericekme.Parameters.AddWithValue("@id", user_data.Id);
          // Bağlantıyı aç
          baglanti.Open();
          using (SqlDataReader reader = vericekme.ExecuteReader())
          {
```

```
            if (reader.Read())
            {
                usr_name_box.Text = reader["NAME"].ToString();
                user_surname_box.Text = reader["SURNAME"].ToString();
                user_phonenum_box.Text = reader["PHONE"].ToString();
                user_mail_box.Text = reader["MAIL"].ToString();
                // Fotoğraf okuma, varsa
                if (!reader.IsDBNull(reader.GetOrdinal("PHOTO"))) // Burada doğru sütun adı kullanılıyor
                {
                    byte[] imageData = (byte[])reader["PHOTO"];
                    using (MemoryStream ms = new MemoryStream(imageData))
                    {
                        System.Drawing.Image profilephoto = System.Drawing.Image.FromStream(ms);
                        profile_pic_box.Image = profilephoto;
                    }
                }
                else
                {
                    profile_pic_box.Image = null; // veya varsayılan bir resim atayabilirsiniz
                }
            }
            else
            {
                MessageBox.Show("No user found with the provided ID.");
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("An error occurred: " + ex.Message);
    }
    finally
    {
        baglanti.Close(); // Bağlantıyı her durumda kapat
    }
}
private void pp_change_button_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Filter = "Image files (*.jpg, *.jpeg, *.png) | *.jpg; *.jpeg; *.png";
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        profile_pic_box.ImageLocation = openFileDialog.FileName;
        resim_textbox.Text = openFileDialog.FileName;
        SaveImageToDatabase(openFileDialog.FileName);
    }
}
private byte[] ImageToByteArray(string imagePath)
{
    using (var imageStream = new FileStream(imagePath, FileMode.Open, FileAccess.Read))
    {
        var binaryReader = new BinaryReader(imageStream);
        return binaryReader.ReadBytes((int)imageStream.Length);
    }
}
private void SaveImageToDatabase(string imagePath)
{
    byte[] imageBytes = ImageToByteArray(imagePath);

    string sql = "UPDATE USER_DATA SET PHOTO = @Photo WHERE ID = @Id";
    using (SqlCommand command = new SqlCommand(sql, baglanti))
    {
```

```
            command.Parameters.AddWithValue("@Photo", imageBytes);
            command.Parameters.AddWithValue("@Id", user_data.Id);
            baglanti.Open();
            int result = command.ExecuteNonQuery();
            if (result > 0)
            {
                MessageBox.Show("Profile photo updated successfully.");
            }
            else
            {
                MessageBox.Show("No record was updated.");
            }
        }
    }
    private void pictureBox2_Click(object sender, EventArgs e)
    {
        homepage homepage = new homepage();
        homepage.Show();
        this.Hide();
    }
    private void pictureBox4_Click(object sender, EventArgs e)
    {
        notes notes = new notes();
        notes.Show();
        this.Hide();
    }
    private void pictureBox3_Click(object sender, EventArgs e)
    {
        reminder_liste reminder_Liste = new reminder_liste();
        reminder_Liste.Show();
        this.Hide();
    }
    private void pictureBox6_Click(object sender, EventArgs e)
    {
        Friends friends = new Friends();
        friends.Show();
        this.Hide();
    }
    private void pictureBox7_Click(object sender, EventArgs e)
    {
        profile profile = new profile();
        profile.Show();
        this.Hide();
    }
  }
}
```

Requests

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Windows.Forms;
using static pinnotedeskup.Program;
namespace pinnotedeskup.wgt
{
  public partial class IsteklerEkran : Form
  {
    private DateTime lastCheck = DateTime.Now;
    SqlConnection baglanti = new SqlConnection("Data Source = SEDEN\\SQLEXPRESS; Initial Catalog
= PinNoteDB; Integrated Security = True; Encrypt=False;");
    public IsteklerEkran()
    {
      InitializeComponent();
      LoadRequests();
```

```
    }

    private void LoadRequests()
    {
        string tableName = "REQUESTS_" + user_data.Id;
        List<int> requestIds = new List<int>();
        baglanti.Open();
        // İstek ID'lerini çekme
        using (SqlCommand requestList = new SqlCommand($"SELECT RequesterID FROM {tableName}",
baglanti))
        {
            using (SqlDataReader reader = requestList.ExecuteReader())
            {
                while (reader.Read())
                {
                    requestIds.Add(reader.GetInt32(0));
                }
            }
        }
        // Kişi bilgilerini çekme
        foreach (int reqID in requestIds)
        {
            using (SqlCommand friendsInfo = new SqlCommand("SELECT NAME, SURNAME FROM
USER_DATA WHERE ID = @ID", baglanti))
            {
                friendsInfo.Parameters.AddWithValue("@ID", reqID);
                using (SqlDataReader reader = friendsInfo.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        string name = reader.GetString(0);
                        string surname = reader.GetString(1);
                        istek.Rows.Add(reqID, name, surname, "Accept", "Decline");
                    }
                }
            }
        }
        baglanti.Close();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        this.Close();
    }
    private void istek_CellContentClick_1(object sender, DataGridViewCellEventArgs e)
    {
        if (e.RowIndex >= 0 && istek.Columns[e.ColumnIndex] is DataGridViewButtonColumn)
        {
            int friendId = Convert.ToInt32(istek.Rows[e.RowIndex].Cells[0].Value);

            if (istek.Columns[e.ColumnIndex].Name == "accept")
            {
                AcceptRequest(friendId);
            }
            else if (istek.Columns[e.ColumnIndex].Name == "decline")
            {
                RemoveRequest(friendId);
            }
        }
    }
    private void AcceptRequest(int friendId)
    {
        string tableName = "FRIENDS_" + user_data.Id;
        baglanti.Open();
```

```
        using (SqlCommand command = new SqlCommand($"INSERT INTO {tableName} (FriendID,
AddedDate) VALUES (@friendId, @date)", baglanti))
        {
            command.Parameters.AddWithValue("@friendId", friendId);
            command.Parameters.AddWithValue("@date", DateTime.Now);
            command.ExecuteNonQuery();
        }
        RemoveRequest(friendId);
        baglanti.Close();
    }
    private void RemoveRequest(int friendId)
    {
        string tableName = "REQUESTS_" + user_data.Id;
        baglanti.Open();
        using (SqlCommand command = new SqlCommand($"DELETE FROM {tableName} WHERE
RequesterID = @friendId", baglanti))
        {
            command.Parameters.AddWithValue("@friendId", friendId);
            command.ExecuteNonQuery();
        }
        // DataGridView'dan satırı kaldırma
        foreach (DataGridViewRow row in istek.Rows)
        {
            if (Convert.ToInt32(row.Cells[0].Value) == friendId)
            {
                istek.Rows.Remove(row);
                break;
            }
        }
        baglanti.Close ();
    }
  }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace pinnotedeskup
{
  internal static class Program
  {
    /// <summary>
    /// Uygulamanın ana girdi noktası.
    /// </summary>
    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new giris());
    }
    public static class user_data
    {
        public static int Id { get; set; }
        public static string Name { get; set; }
        public static string Surname { get; set; }
        public static string Email { get; set; }
        public static string Password { get; set; }
        public static string PhoneNumber { get; set; }
        public static DateTime BirthDate { get; set; }
    }}}
```