Table of Contents

[1. PROJE KONUSU 2](#_Toc408072548)

[2. KOD ANLATIMI 3](#_Toc408072549)

[3. KULLANICI KILAVUZU 21](#_Toc408072550)

[3.1. Login Form 22](#_Toc408072551)

[3.2. Admin Form 22](#_Toc408072552)

[3.3. Doctor Form 28](#_Toc408072553)

[3.4. Staff Form 32](#_Toc408072554)

# PROJE KONUSU

Projemizin amacı bir hastanede yönetici, doktor ve çalışanlar tarafından kullanılabilecek ortak bir veritabanına erişip işlem yapılabilen bir program yapmaktır.

Programımızda kullanıcının doktor mu yönetici mi çalışan mı olduğu ana giriş ekranında anlaşılır. Bu ekrandan admin olarak giriş yapılarak , bölümlerin , doktorların ve çalışanların bilgileri görüntülenebilir , yeni çalışan ya da doktor eklenebilir, bilgiler üzerinden güncelleme yapılabilinir.

Programımıza doktor olarak giriş yapılırsa, kullanıcı adından hangi bölümde çalıştığı ve kim olduğu program tarafından anlaşılır ve hasta ekranı kullanıcı karşısına gelir. Buradan hastalar seçilerek, doktor tarafından daha geniş bilgiler, şikayetleri, yazılan ilaçlar vb. görüntülenebilir ya da doktor tarafından yeni testler, hasta şikayetleri, hastalık tanıları ve yazılan ilaç bilgileri girilerek kaydedilebilir.

Programımıza çalışan olarak giriş yapılırsa hastanaye yeni hasta kaydı eklemek ve hasta bilgilerini güncellemek gibi seçenekler elde edilir. Buradan bilgileri alınarak hastaneye kaydı yapılan hastalar daha sonra doktorlar tarafından görüntülenebilmektedir.

# KOD ANLATIMI

**DoctorPersonalInforForm sınıfı**

Bu sınıf doktorun kendi kişisel bilgilerini görüntülemesi ve değiştirmesini sağlar.

**Sınıf üyeleri:**

|  |
| --- |
| private HospitalDataContext database = new HospitalDataContext();  private String username; |

* ‘database’ nesnesi veritabanından yapılacak okuma ve yazma işlemleri için kullanırız.
* ‘username’ String tipinde ilgili doktorun kullanıcı adını tutmaktadır.
* Yapıcı fonksiyonda username’e atama yaptık. Bu class’ın nesnesinin yaratıldığı yerde username parametresi gönderilir yapıcı metoda.

**Metotlar:**

* **exitButton\_Click** metodu ilgili form’da Exit butonuna basıldığında çalışır. Geçerli olan bu form sınıfını kapatma işleminden sorumludur.

|  |
| --- |
| private void exitButton\_Click(object sender, EventArgs e)  {  this.Close();  } |

* **loadData()** metodu form’daki metin kutularına verilerin eklenmesini sağlar. Yukarda oluşturduğumuz ‘database’ veritabanı nesnesi ile işlem yaparız. ‘username’in belirtmiş olduğu kişiyi ilgili Linq sorgusu yardımıyla buluruz ve bulunan User kaydı, doctorBindingSource kaynağına atanarak metin kutularında gösterimi sağlanır.

|  |
| --- |
| private void loadData()  {  var doc = from user in database.Users  from doctor in user.Doctors  where user.User\_name == username  select doctor;  doctorBindingSource.DataSource = doc;  } |

* **DoctorPersonalInfoForm\_Load()** metodu form ilk yüklendiği an çalışır. loadData() metodunu çağırır.

|  |
| --- |
| private void DoctorPersonalInfoForm\_Load(object sender, EventArgs e)  {  loadData();  } |

* **saveButton\_Click** metodu saveButton’a tıklandığı zaman gerçekleşir. Eğer metin kutuları boşsa kaydetmeyi engeller ve uyarı mesajı gösterir. Metin kutuları doluysa değişikliklerin database üzerinde kayıtlanması için gerekli işlemleri yapar ve mesaj gösterir.

|  |
| --- |
| private void saveButton\_Click(object sender, EventArgs e)  {  if (doctor\_nameTextBox.Text == "" || doctor\_surnameTextBox.Text == "" || phoneTextBox.Text == "" || addressTextBox.Text == "")  MessageBox.Show("You must enter data to fields.", "Blank Field", MessageBoxButtons.OK, MessageBoxIcon.Error);  else  {  Validate(); *//Değişiklikleri uygular.*  doctorBindingSource.EndEdit(); *//Yazmayı sonlandırır.*  database.SubmitChanges(); *//Değişiklikleri doğrular.*  MessageBox.Show("Your changes have been saved successfully.", "Changed", MessageBoxButtons.OK);  }  } |

**InspectPatientForm sınıfı**

Bu sınıf doktorun bir hastayı muayene etmesini sağlar.

**Sınıf Üyeleri**

|  |
| --- |
| private HospitalDataContext database = new HospitalDataContext();  private int admissionID; |

* ‘database’ nesnesi veritabanından yapılacak okuma ve yazma işlemleri için kullanırız.
* ‘admissionID’ int tipinde, ilgili hastanın kayıt numarasını alır.
* Yapıcı fonksiyonda admissionID’ye değer atadık parametre olarak geleni.

**Metotlar**

* **loadData()** metodu formdaki ilgili metin kutuları ve check box’lara otomatik doldurmaları yapar. Verileri sorguyla database nesnesinin Patient\_Admissions tablosundan ‘admissionID’ye göre alır.

|  |
| --- |
| private void loadData()  {  patient\_AdmissionBindingSource.DataSource=  from patient in database.Patient\_Admissions  where patient.Patient\_ad\_id == admissionID  select patient;  } |

* **InspectPatientForm\_Load** metodu form ilk yüklendiğinde çalışır ve loadData() metodunu çağırır.
* **saveButton\_Click** metodu kullanıcının veri eklemesi veya değiştirmesi sonrası değişikliklerini kaydetmeyi sağlar.

|  |
| --- |
| private void saveButton\_Click(object sender, EventArgs e)  {  Validate();  patient\_AdmissionBindingSource.EndEdit();  database.SubmitChanges();  } |

* **closeButton\_Click** metodu geçerli form’u kapatmaya yarar. *(this.Close())*

**LoginForm sınıfı**

Bu sınıfın nesnesi program ilk çalıştığında oluşturulur ve kullanıcıya ilk bu form görünür.

**Sınıf Üyeleri**

* Veritabanı nesnesi olan ‘database’
* jobIndex int tipinde bir değişkendir. Hangi kullanıcı giriş yaptıysa o kullanıcının görevine göre bir index atanır.

**Metotlar**

* **ValidateLogin()** metodu kullanıcının username ve password metin kutularına girdiği verilere göre veritabanında öyle bir kullanıcının olup olmadığı bildirir ve aynı zamanda kullanıcı varsa o kullanıcının görevini belirten jobIndex’e değer ataması yapılır.

|  |
| --- |
| private bool ValidateLogin()  {  var inputLogin = from login in database.Users  where (login.Password == passwordTextBox.Text && login.User\_name == userNameTextBox.Text)  select login;  if (inputLogin.Count() != 0) *//Eğer öyle bir kullanıcı varsa*  {  if (inputLogin.First().Job.Equals("Admin"))  jobIndex = 0;  else if (inputLogin.First().Job.Equals("Doctor"))  jobIndex = 1;  else if (inputLogin.First().Job == "Staff")  jobIndex = 2;  return true;  }  else return false;  } |

* **loginButton\_Click** metodu girilen kullanıcı adı ve şifreye bir sonraki formu açmaya yarar. Eğer kullanıcı adı şifre yanlışsa hata mesajı verir.

|  |
| --- |
| private void loginButton\_Click(object sender, EventArgs e)  {  bool flag;  flag=ValidateLogin(); *//Kullanı adı şifrenin doğruluğu kontrol ediliyor*  if (flag == true)*//Eğer varsa öyle bir kullanıcı,göreve göre yeni form açar*  {  if (jobIndex == 0)  {  AdminForm admf = new AdminForm(userNameTextBox.Text);  admf.Show();  }  else if (jobIndex == 1)  {  DoctorForm docf = new DoctorForm(userNameTextBox.Text);  docf.Show();  }  else if (jobIndex == 2)  {  StaffForm stf = new StaffForm(userNameTextBox.Text);  stf.Show();  }  this.Hide(); *//Bu formu saklar*  }  else  {  MessageBox.Show("Username or password is wrong, please try again.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  } |

* **exitButton\_Click** metodu geçerli formu kapatır.

**StaffForm sınıfı**

Bu sınıfın nesnesi görevi Staff olan bir kullanıcının giriş yapmasının ardından yaratılır.

**Sınıf Üyeleri**

* ‘username’ String tipinde bir değişkendir. Giriş yapan Staff’ın kullanıcı adını alır yapıcı metot aracılığıyla.
* ‘database’ nesnesi veritabanı işlemleri için kullanılır.

**Metotlar**

* **exitToolStripMenuItem1\_Click** metodu Menu Strip: User Account > *Exit* butonuna basıldığında gerçekleşir ve uygulamayı sonlandırmaya yarar. *(Application.Exit())*
* **logOffToolStripMenuItem\_Click** metodu Menu Strip: User Account > Log Off butonuna basıldığında gerçekleşir ve geçerli formu sonlandırır ve LoginForm nesnesi yaratır.

|  |
| --- |
| private void logOffToolStripMenuItem\_Click(object sender, EventArgs e)  {  this.Close();  LoginForm logf = new LoginForm();  logf.Show();  } |

* **aboutToolStripMenuItem\_Click** metodu Menu Strip: Help > About butonuna basıldığında gerçekleşir ve mesaj kutusu şeklinde program hakkında bilgi verir.
* **StaffForm\_Load** metodu form ilk yüklendiğinde gerçekleşir. ‘timer’ nesnesinin ayarlamalarını yapar ve başlatır. Aynı zamanda geçerli kullanıcının ad-soyadını veritabanından username değişkeni yardımıyla çeker ve ilgili label’e yazar bu ad-soyadı.

|  |
| --- |
| private void StaffForm\_Load(object sender, EventArgs e)  {  timer1.Interval = 1000;  timer1.Start();  var staff= from user in database.Users  from stf in user.Staffs  where user.User\_name == username  select stf;  helloLabel.Text = "Hello, " + staff.First().Name + " " + staff.First().Surname; } |

* **timer1\_Tick** metodu ilgili label’lere şu anki gün ve zamanı atar.

|  |
| --- |
| private void timer1\_Tick(object sender, EventArgs e)  {  dateLabel.Text = DateTime.Now.ToLongDateString();  hourLabel.Text = DateTime.Now.ToLongTimeString();  } |

* **exitButton\_Click** metodu uygulamayı kapatır. *(Application.Exit())*
* **addPatientButton\_Click** ve **updateButton\_Click** metotları Staff’ın yeni hasta eklemek istediğinde veya hasta bilgilerini değiştirmek istediğinde addPatientButton’una veya updatePatientButton’a bastığında gerçekleşir ve yeni form için nesne yaratır.

|  |
| --- |
| private void addPatientButton\_Click(object sender, EventArgs e)  {  AddPatientForm apf = new AddPatientForm();  apf.ShowDialog();  }    private void updateButton\_Click(object sender, EventArgs e)  {  UpdatePatientForm upf = new UpdatePatientForm();  upf.ShowDialog();  } |

* **changeUserSettingsToolStripMenuItem1\_Click** metodu geçerli Staff’ın kullanıcı bilgilerini değiştirmesi için User Account > Change User Settings’e tıkladığında gerçekleşir ve ilgili yeni formun nesnesini yaratıp görünür yapar.

|  |
| --- |
| private void changeUserSettingsToolStripMenuItem1\_Click(object sender, EventArgs e)  {  UserSettings suf = new UserSettings(username);  suf.ShowDialog();  } |

* **personalInfoSettingsToolStripMenuItem\_Click** metodu geçerli Staff’ın kişisel bilgilerini değiştirmesi için User Account > My Personal Info’ya tıkladığında gerçekleşir ve ilgili yeni formun nesnesini yaratıp görünür yapar.

|  |
| --- |
| private void personalInfoSettingsToolStripMenuItem\_Click(object sender, EventArgs e)  {  StaffPersonalInfoForm spif = new StaffPersonalInfoForm(username);  spif.ShowDialog() } |

**StaffInfoExtendedForm sınıfı**

Bu sınıfın nesnesi admin’in genel Staff Bilgilerini gördüğü ekrandan herhangi bir staff’ın tüm bilgilerini görmesini istediğinde yaratılır.

**Sınıf Üyeleri**

* ‘database’ nesnesi veritabanı işlemleri için kullanılmaktadır.
* ‘staffID’ değişkeni ise geçerli staffın id numarasını tutmaktadır, o kişinin bilgilerine erişim için.

**Metotlar**

* **staffsBindingNavigatorSaveItem\_Click** metodu Save icon’una basıldığında gerçekleşir ve o anki haliyle formu veritabanına kaydeder.

|  |
| --- |
| private void staffsBindingNavigatorSaveItem\_Click(object sender, EventArgs e)  {  Validate();  staffsBindingSource.EndEdit();  database.SubmitChanges();  } |

* **StaffInfoExtendedForm\_Load** metodu form ilk yüklendiğinde gerçekleşir. İlgili kaynağın verilerine veritabanından bilgileri çeker.

|  |
| --- |
| private void StaffInfoExtendedForm\_Load(object sender, EventArgs e)  {  staffsBindingSource.DataSource = from user in database.Users  from staff in user.Staffs  where staff.Staff\_id == staffID  select staff;  } |

* **closeButton\_Click** metodu ilgili formu kapatır. *(this.Close())*
* **bindingNavigatorDeleteItem\_Click** metodu bilgileri açık olan Staff’ı veritabanından siler. Aynı zamanda User tablosundan da gerekli silmeyi gerçekleştirir.

|  |
| --- |
| private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)  {  User deletedUser = (from user in database.Users  from staff in user.Staffs  where staff.Staff\_id == staffID  select user).First();  database.Users.DeleteOnSubmit(deletedUser);  database.SubmitChanges();  } |

**StaffInfoForm sınıfı**

Bu sınıfı nesnesi admin tarafından tüm staff’ların görüntülenmesi istediğinde yaratılır.

**Sınıf Üyeleri**

* ‘database’ nesnesi veritabanı işlemleri için gereklidir.

**Metotlar**

* **StaffInfoForm\_Load** metodu form ilk yüklendiğinde gerçekleşir. İlgili data grid’in kaynağına veritabanından verileri ekler.

|  |
| --- |
| private void StaffInfoForm\_Load(object sender, EventArgs e)  {  staffBindingSource.DataSource = from staff in database.Staffs  orderby staff.Name,staff.Surname  select staff;  } |

* **refreshButton\_Click** netodu herhangi bir değişiklik sonra data grid view’in verilerinin yenilenmesini sağlar. Load metodunu çağırır. *(this.StaffInfoForm\_Load(sender, e);)*
* **closeButton\_Click** metodu geçerli formu kapatır. *(this.Close())*
* **staffDataGridView\_CellDoubleClick** metodu data grid view’de herhangi bir satırın çift tıklandığı zaman yeni formun açılması için gereken nesne ve komutları gerçekleştirmektedir.

|  |
| --- |
| private void staffDataGridView\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)  {  StaffInfoExtendedForm sief = new StaffInfoExtendedForm((int)staffDataGridView[0,staffDataGridView.CurrentRow.Index].Value);  sief.ShowDialog();  } |

**StaffPersonalInfoForm sınıfı**

Bu sınıfın nesnesi ilgili staff’ın kendi kişisel bilgilerini görüntüleyip değiştirmek istediğinde yaratılır.

**Sınıf Üyeleri**

* ‘database’ nesnesi veritabanı işlemleri için kullanılır.
* ‘username’ String tipindeki değişken ilgili Staff’ın kullanıcı adını yapıcı metot ile alır.

**Metotlar**

* **exitButton\_Click** metodu exit butonuna basıldığında gerçekleşir ve geçerli formu kapatır*.(this.Close())*
* **loadData()** metodu veritabanından ilgili kullanıcının bilgilerini username değişkeni aracılığıyla çeker ve kaynağa atar.

|  |
| --- |
| private void loadData()  {  staffBindingSource.DataSource = from user in database.Users  from staff in user.Staffs  where user.User\_name == username  select staff;  } |

* **StaffPersonalInfoForm\_Load** metodu form ilk yüklendiğinde çalışır ve loadData() metodunu çağırır.
* **saveButton\_Click** metodu Save butonuna basıldığında gerçekleşir. Metin kutuları boş değilse veritabanına ilgili değişiklikleri kaydeder, metin kutularından boş olan varsa uyarı mesajı verir.

|  |
| --- |
| private void saveButton\_Click(object sender, EventArgs e)  {  if (nameTextBox.Text == "" || surnameTextBox.Text == "" || phoneTextBox.Text == "" || addressTextBox.Text == "")  MessageBox.Show("You must enter data to fields.", "Blank Field", MessageBoxButtons.OK, MessageBoxIcon.Error);  else  {  Validate();  staffBindingSource.EndEdit();  database.SubmitChanges();  MessageBox.Show("Your changes have been saved successfully.", "Changed", MessageBoxButtons.OK);  }  } |

**UpdatePatientForm sınıfı**

Bu sınıfın nesnesi Staff’ın bir hastanın bilgilerini güncellemek istediğinde yaratılır.

**Sınıf Üyeleri**

* ‘database’ nesnesi veritabanı işlemleri için kullanılır.

**Metotlar**

* **closeButton\_Click** geçerli formun kapatılmasını sağlar*. (this.Close())*
* **refreshPatients()** metodu veri kaynağına veritabanından verileri çeker. İlk veriye gider. Ve de arama kutusunu temizler.

|  |
| --- |
| private void refreshPatients()  {  patientBindingSource.DataSource = from patient in database.Patients  orderby patient.Patient\_id  select patient;  patientBindingSource.MoveFirst();  ssnTextBox.Clear();  } |

* UpdatePatientForm\_Load metodu form ilk yüklendiğinde gerçekleşir. refreshPatients() metodunu çağırır.
* **patientBindingNavigatorSaveItem\_Click** metodu Save Butonuna basıldığında gerçekleşir. İlgili değişikleri veritabanına kaydeder. Ve kaynak verilerine yeniler.

|  |
| --- |
| private void patientBindingNavigatorSaveItem\_Click(object sender, EventArgs e)  {  Validate();  patientBindingSource.EndEdit();  database.SubmitChanges();  refreshPatients();  } |

* **findButton\_Click** metodu Click butonuna tıklandığında gerçekleşir. ssnTextBox adlı metin kutusundaki numaraya göre ilgili hastanın kaydını veritabanından alır.

|  |
| --- |
| private void findButton\_Click(object sender, EventArgs e)  {  patientBindingSource.DataSource = from patient in database.Patients  where patient.Ssn == ssnTextBox.Text  select patient;  } |

* **browseButton\_Click** metodu Browse butonuna tıklandığında gerçekleşir ve refreshPatients() metodunu çağırır.

**UserSettings sınıfı**

Bu sınıfın nesnesi Doktor veya Staff’ın kullanıcı bilgilerini değiştirmek veya görüntülemek istediğinde gerçekleşir.

**Sınıf Üyeleri**

* ‘database’ nesnesi veritabanı işlemleri için kullanılır.
* ‘username’ String değişkeni ilgili kullanıcının kullanıcı adını tutar ve yapıcı metot aracılığıyla değer atanır.

**Metotlar**

* **StaffUserSettings\_Load** metodu formun ilk yüklenmesinde gerçekleşir ve loadData() metodunu çağırır.
* **loadData()** metodu ilgili kullanıcının bilgilerini kaynağa yükler. ‘username’ değişkenine göre veritabanından ilgili kaydı çeker.

|  |
| --- |
| private void loadData()  {  userBindingSource.DataSource = from user in database.Users  where user.User\_name == username  select user;  } |

* **saveButton\_Click** metodu Save butonuna basıldığında çalışır. İlgili değişiklikler sonrası veritabanına bilgilerin kaydedilmesini sağlar. Password veya Username sahaları boşsa uyarı mesajı verir. Doluysa eğer değişiklikleri veritabanına kaydeder.

|  |
| --- |
| private void saveButton\_Click(object sender, EventArgs e)  {  if(passwordTextBox.Text=="")  MessageBox.Show("Password field cannot be blank.", "Blank Field", MessageBoxButtons.OK, MessageBoxIcon.Error);  else if (user\_nameTextBox.Text == "")  MessageBox.Show("Username field cannot be blank.", "Blank Field", MessageBoxButtons.OK, MessageBoxIcon.Error);  else  {  Validate();  userBindingSource.EndEdit();  database.SubmitChanges();  loadData();  MessageBox.Show("Your changes have been saved successfully.","Changed",MessageBoxButtons.OK);  passwordTextBox.UseSystemPasswordChar = true;  }  } |

* **exitButton\_Click** metodu ilgili form’da Exit butonuna basıldığında çalışır. Geçerli olan bu form sınıfını kapatma işleminden sorumludur. *(this.Close())*
* **passwordTextBox\_Enter** metodu Password metin kutusunun karakter görünümünü değiştirir.

|  |
| --- |
| private void passwordTextBox\_Enter(object sender, EventArgs e)  {  passwordTextBox.UseSystemPasswordChar = false;  } |

**AddDoctorForm sınıfı**

* **AddDoctorForm():**AddDoctorForm classının yapıcı methodudur.
* **continueButton\_Click** **(object sender, EventArgs e):** AddDoctorForm’undaki continue butonunun click eventini oluşturan methoddur. passwordTextBox ile passwordTextBox2 nin içeriklerini kontrol ederek karşılaştırır ve yanlış şifre girilmesini ya da boş şifre girilmesini engeller.Eğer girilen değerler doğru ise formu 2.tab sekmesine geçirir.
* **AddDoctorForm\_Load** **(object sender, EventArgs e):** Tabda 2.sekmeye geçildiği zaman comboBox’ta department adlarının gözükmesi için database’in Department entitysinden Dept\_name’i çeker.
* **saveButton\_Click (object sender, EventArgs e):** Save yazan butona basınca yapılacak işlerin olduğu methoddur.Eğer eksik bilgi girersek hata yazısı çıkar.Girmezsek , Doctor ve User entityleri için Doctor ve User tipinde iki nesne üretilir ve textBox’lara girilen değerler bu nesnelerin attributelerine atanır.Daha sonra User tipindeki nesne ‘InsertOnSubmit’ methodu ile databasein User tablosuna eklenir.Daha sonra eklenen bu usera database tarafından verilen user\_id databaseden okunur ve Doctor tipindeki nesneye gerekli olan attribute değerleri atanır.Daha sonra Doctor tipindeki nesne InsertOnSubmit methodu ile databasein Doctor tablosuna eklenir ve bu ekleme işleminden sonra eklemenin başarılı olduğuna dair MessageBox gösterilir. Bu messageBox’ta doktor eklemeye devam etmek isteyip istemediği sorulur ve istiyorsa tablar sıfırlanır . İstemiyorsa form sonlandırılır.
* **previousButton\_Click** **(object sender, EventArgs e) :** Önceki taba dönmeyi sağlayan   
  methoddur.tabControl.SelectedIndex’e 0 değerini atayarak ilk taba dönmeyi sağlar.
* **tabControl\_SelectedIndexChanged (object sender, EventArgs e):** Tabın hangi sekmede olduğunu anlayan methoddur.İlk tabda password değerleri ile username istenilen şekilde girilmeden ikinci sekmeye geçilmesini önler.
* **closeButton\_Click (object sender, EventArgs e):** Form sekmesini kapatmaya yarayan butondur.Basıldığı zaman bir önceki form sayfasına dönülür.
* **close2Button\_Click** **(object sender, EventArgs e):** Form sekmesini kapatmaya yarayan butondur.Basıldığı zaman bir önceki form sayfasına dönülür.

**AddPatientForm sınıfı**

* **AddPatientForm():** Yapıcı methoddur.
* **saveButton\_Click** **(object sender, EventArgs e):** Save butonuna basınca yapılacak olan işler bu methodda yapılır. Gerekli sahalar boş bırakıldıysa ya da eksik girildiyse MessageBox ile hata yazısı çıkar.Eğer bütün sahalar gerektiği şekilde girildi ise , ve daha önceden böyle bir hasta hastaneye gelmedi ise Patient tipinde bir patient nesnesi oluşturulur ve hastanın nüfus cüzdanı bilgileri textBoxlara girilen değerleri bu nesnenin bileşenlerine atar.Daha sonra InsertOnSubmit methodu aracılığı ile databasein Patient tablosuna bu nesne eklenir ve databasedeki değişiklikler kaydedilir. Daha sonra Patient\_Admission tipinde patientAdm nesnesi oluşturulur ve patientAdm nesnesine diğer tablolarla bağlantı kurabileceği bilgiler olan patient\_id , dept\_id , doctor\_id databaseden okunarak atanır.Daha sonra bu nesne InsertOnSubmit methodu ile database’in Patient\_Admission tablosuna eklenir ve databasedeki değişiklikler kaydedilir.Yeni bir hasta eklemek isteyip istemediği kullanıcıya sorulur ve gelen cevap evet ise textBoxlardaki değerler temizlenir ; hayır ise form kapatılır.
* **exitButton\_Click** **(object sender, EventArgs e):** Exit butonuna basınca form’u kapatır.
* **AddPatientForm\_Load** **(object sender, EventArgs e):** Department adlarını gösteren comboBox ile seçilen departmanlardaki doktorlar databaseden okunur.
* **checkButton\_Click** **(object sender, EventArgs e):**Check butonuna basınca yapılacak işlemleri içeren methoddur. Hastanın Ssn numarası databaseden okunur ve textBox ‘a girilen değer ile karşılaştırılır.Eğer hasta daha önce hastaneye geldi ise isExist değişkenine true , gelmediyse false atanır.
* **deptComboBox\_SelectedIndexChanged** **(object sender, EventArgs e):**deptComboBox’ının eventidir.Bu comboboxtan department seçilir ve seçilen departmandaki doktorlar databaseden çekilerek doctorComboBox’ına atanır.

**AddStaffForm sınıfı**

* **AddStaffForm():**Yapıcı methoddur.
* **continueButton\_Click (object sender, EventArgs e):** AddStaffForm’undaki continue butonunun click eventini oluşturan methoddur. passwordTextBox ile passwordTextBox2 nin içeriklerini kontrol ederek karşılaştırır ve yanlış şifre girilmesini ya da boş şifre girilmesini engeller. Eğer girilen değerler doğru ise formu 2.tab sekmesine geçirir.
* **saveButton\_Click** **(object sender, EventArgs e):** Save yazan butona basınca yapılacak işlerin olduğu methoddur. Eğer eksik bilgi girersek hata yazısı çıkar.Girmezsek , Staff ve User entityleri için Staff ve User tipinde iki nesne üretilir ve textBox’lara girilen değerler bu nesnelerin attributelerine atanır.Daha sonra User tipindeki nesne ‘InsertOnSubmit’ methodu ile databasein User tablosuna eklenir.Daha sonra eklenen bu usera database tarafından verilen user\_id databaseden okunur ve Staff tipindeki nesneye gerekli olan attribute değerleri atanır.Daha sonra Staff tipindeki nesne InsertOnSubmit methodu ile databasein Staff tablosuna eklenir ve bu ekleme işleminden sonra eklemenin başarılı olduğuna dair MessageBox gösterilir. Bu messageBox’ta staff eklemeye devam etmek isteyip istemediği sorulur ve istiyorsa tablar sıfırlanır . İstemiyorsa form sonlandırılır.
* **tabControl\_SelectedIndexChanged** **(object sender, EventArgs e):** Tabın hangi sekmede olduğunu anlayan methoddur. İlk tabda password değerleri ile username istenilen şekilde girilmeden ikinci sekmeye geçilmesini önler.
* **closeButton\_Click (object sender, EventArgs e):** Form sekmesini kapatmaya yarayan butondur.Basıldığı zaman bir önceki form sayfasına dönülür.
* **close2Button\_Click (object sender, EventArgs e):** Form sekmesini kapatmaya yarayan butondur.Basıldığı zaman bir önceki form sayfasına dönülür.

**AdminChangePassword sınıfı**

* **AdminChangePassword** **(object sender, EventArgs e):** Yapıcı methoddur.Bu method içinde daha önceki pencerelerde admin tarafından girilmiş olan username bu class içindekiusername değişkenine atanır.
* **AdminChangePassword\_Load** **(object sender, EventArgs e):** username değeri usernameTextBox’ının text kısmına atanır fakat bu text kısmı değiştirilmeye açık değildir.
* **exitButton\_Click(object sender, EventArgs e):** Exit butonuna basılınca yapılacak işlerin olduğu methoddur.Bu method çalıştığı zaman , AdminChangePassword formu kapanır.
* **saveButton\_Click\_1(object sender, EventArgs e):** Save butonuna basınca yapılacak işlerin olduğu methoddur.Bu formu sadece admin açabileceği için adminin user bilgileri databaseden username ile okunur.Daha sonra adminin önceki password değeri ile şuandaki password değeri bu okunan satır aracılığı ile karşılaştırılır.Eğer iki değer aynı programın başında admin olduğu anlaşılır ve User tablosunda adminin bilgilerinin olduğu satırda adminin passworduna newPasswordTextBox’ının textindeki password değeri atanır ve veritabanına kaydedilir.Daha sonra kaydedildiğine dair bir mesaj ekranda gösterilir.Eğer adminin kontrol sırasında girdiği password değeri ile User’dan okunan password değeri aynı değilse programı kullanan kişinin admin olmadığı düşünülerek , şifreyi yanlış girdiğine dair hata verilir.

**AdminForm sınıfı**

* **AdminForm(String username):**Yapıcı methoddur.Adminin login sırasında girdiği username parametre olarak alınır ve AdminForm classı içindeki username değişkenine atanır.
* **AdminForm\_Load(object sender, EventArgs e):**AdminForm çalıştırılınca oluşturulacak şeylerin olduğu methoddur.Formda bulunan timer1 adlı timer çalıştırılır.
* **timer1\_Tick(object sender, EventArgs e):**Timer1’in çalışması boyunca yapılacak işlerin olduğu methoddur.dateLabel değişkeninin textine makinenin tarihi , hourLabel değişkenin textine ise makinenin saati atanır.
* **exitButton\_Click(object sender, EventArgs e):**Exit butonu basılınca olacak olayların olduğu methoddur.Bu butona basınca program çalışmayı durdurur ve kapanır.
* **exitToolStripMenuItem\_Click(object sender, EventArgs e):** Menudeki exit butonu basılınca olacak olayların olduğu methoddur.Bu butona basınca program çalışmayı durdurur ve kapanır.
* **logOffToolStripMenuItem\_Click(object sender, EventArgs e):** Menude User Account sekmesindeki Log Off yazısına basınca olacak olayların olduğu methoddur. Buraya basınca AdminForm kapanır ve LoginForm geri açılır.
* **aboutToolStripMenuItem\_Click(object sender, EventArgs e):** Menude Help sekmesindeki About yazısına basınca olacak olayların olduğu methoddur. Buraya tıklanınca bu programın hangi amaçla yazıldığını belirten bir MessageBox ekranda belirir.
* **addDoctorButton\_Click(object sender, EventArgs e):** addDoctorButton’una basılınca yapılacak olayların olduğu methoddur. Buraya tıklanınca AddDoctorForm tipinde bir nesne yani form oluşturulur ve bu nesne dialog şeklinde açılır.
* **addStaffButton\_Click(object sender, EventArgs e):** addStaffButton’una basılınca yapılacak olayların olduğu methoddur. Buraya tıklanınca AddStaffForm tipinde bir nesne yani form oluşturulur ve bu nesne dialog şeklinde açılır.
* **changePasswordToolStripMenuItem\_Click(object sender, EventArgs e):** Menude User Account sekmesindeki Change Password yazısına tıklanınca yapılacak olayların olduğu methoddur. Buraya tıklanınca AdminChangePassword tipinde bir nesne yani form oluşturulur ve bu nesne dialog şeklinde açılır.
* **doctorsInfoToolStripMenuItem\_Click(object sender, EventArgs e):** Menude Informations sekmesindeki Doctors Info yazısına tıklanınca yapılacak olayların olduğu methoddur. Buraya tıklanınca DoctorInfo tipinde bir nesne yani form oluşturulur ve bu nesne dialog şeklinde açılır.
* **staffInfoToolStripMenuItem\_Click(object sender, EventArgs e):** Menude Informations sekmesindeki Staff Info yazısına tıklanınca yapılacak olayların olduğu methoddur. Buraya tıklanınca StaffInfoForm tipinde bir nesne yani form oluşturulur ve bu nesne dialog şeklinde açılır.
* **departmentInfoToolStripMenuItem\_Click(object sender, EventArgs e):** Menude Informations sekmesindeki Department Info yazısına tıklanınca yapılacak olayların olduğu methoddur. Buraya tıklanınca DeparmentInfoForm tipinde bir nesne yani form oluşturulur ve bu nesne dialog şeklinde açılır.

**DepartmentInfoForm sınıfı**

* **DepartmentInfoForm():**Yapıcı methoddur.
* **count():**Hangi departmanda kaç tane doktor olduğunu bulan methoddur. var tipinde doctorCounts değişkeni tanımlanır ve veritabanından Department ve Doctor tablolarına join işlemi yapılarak hangi departmanda kaç doktor olduğu bilgisi çekilir.Daha sonra bu bilgi dataGridView’a atanır.
* **DepartmentInfoForm\_Load(object sender, EventArgs e):** Bu method bu DepartmentInfoForm çalıştığı zaman devreye girer ve count methodunu çağırır.
* **closeButton\_Click(object sender, EventArgs e):**Çarpı işareti şeklinde gözüken , closeButton ‘a tıklanınca olacak olayların olduğu methoddur. Buraya tıklayınca DepartmentInfoForm kapanır.

**DoctorForm sınıfı**

* **DoctorForm(String username):**Yapıcı methoddur.Login ekranında girilen username i parametre olarak alır ve class içindeki username değişkenine atar.
* **aboutToolStripMenuItem\_Click(object sender, EventArgs e):** Menude Help sekmesindeki About yazısına basınca olacak olayların olduğu methoddur. Buraya tıklanınca bu programın hangi amaçla yazıldığını belirten bir MessageBox ekranda belirir.
* **DoctorForm\_Load(object sender, EventArgs e):**DoctorForm çalıştırılınca yüklenecek bilgilerin olduğu methoddur.timer çalıştırılır ve username aracılığı ile veritabanından hangi doktorun sistemi kullanıdığı anlaşılır ve bu doktorun adı ve soyadı ve merhaba yazısı helloLabel’in text kısmına yazılır.
* **timer1\_Tick(object sender, EventArgs e):** Timer1’in çalışması boyunca yapılacak işlerin olduğu methoddur.dateLabel değişkeninin textine makinenin tarihi , hourLabel değişkenin textine ise makinenin saati atanır.
* **allPatients():** var tipinde patient değişkeni tanımlanır ve değişkene User , Doctor ve Patient\_Admissions tablolarının join işlemi sonucu databaseden okunan hasta bilgileri atanır ve daha sonra bu değişkende dataGridView’e atanır.
* **todaysPatients():** var tipinde patient değişkeni tanımlanır ve bu değişkene User , Doctor ve Patient\_Admission tablolarının join işlemi ile tarih kısmı pcnin o günkü tarihine eşit olan hastalar bilgileriyle beraber atanır.Daha sonra bu değişken dataGridView’e atanır.
* **inspectButton\_Click (object sender, EventArgs e):**Üzerinden Inspect yazan butona tıklanınca yapılacak işlemlerin olduğu methoddur.DatagridView’dan hangi satırın seçildiği currentRow ile okunur ve bu seçilen hasta InspectPatientForm tipinde nesne oluştururken parametre olarak atanır.Bu nesne yani form da daha sonra dialog şeklinde açılır.
* **todayToolStripMenuItem\_Click(object sender, EventArgs e):**Menude Patients sekmesindeki Today’s Patients yazısına tıklayınca yapılacak işlemlerin olduğu methoddur.Buraya tıklayınca todaysPatients methodunu çağırır.
* **allToolStripMenuItem\_Click(object sender, EventArgs e):** Menude Patients sekmesindeki All Patients yazısına tıklayınca yapılacak işlemlerin olduğu methoddur. Buraya tıklayınca allPatients methodunu çağırır.
* **exitButton\_Click(object sender, EventArgs e):** Exit butonu basılınca olacak olayların olduğu methoddur.Bu butona basınca program çalışmayı durdurur ve kapanır.
* **changeUserSettingToolStripMenuItem\_Click(object sender, EventArgs e):**Menude User Account’da Change User Settings yazısına tıklayınca olacak olayların olduğu methoddur. UserSettings tipinde username’i parametre olarak alan bir nesne yani form oluşturulur ve bu form dialog şeklinde açılır.
* **myPersonalInformationsToolStripMenuItem\_Click(object sender, EventArgs e):** Menude User Account’da My Personal Informations yazısına tıklayınca olacak olayların olduğu methoddur. DoctorPersonalInfoForm tipinde username’i parametre olarak alan bir nesne yani form oluşturulur ve bu form dialog şeklinde açılır.

**DoctorInfo sınıfı**

* **DoctorInfo():**Yapıcı methoddur.
* **DoctorInfo\_Load(object sender, EventArgs e):**DoctorInfo formu çalıştırıldığında çalışan methoddur.Doktorların isim ve soyisim bilgilerini veritabanındaki Doctor tablosundan çeker ve doctorBindingSource’a atar.
* **doctorDataGridView\_CellDoubleClick(object sender, DataGridViewCellEventArgs e):** doctorDataGirdView’e çift tıklama yapılınca çalışan methoddur.doctorDataGridView’dan o satırdaki doktor bilgilerini okur ve DoctorInfoExtendedForm tipinde nesne yani form oluşturup bu değeri parametre olarak DoctorInfoExtendedForm’a gönderir ve oluşan nesneyi dialog şeklinde açar.
* **closeButton\_Click(object sender, EventArgs e):** Çarpı işareti şeklinde gözüken, closeButton ‘a tıklanınca olacak olayların olduğu methoddur. Buraya tıklayınca DoctorInfo formu kapanır.
* **refreshButton\_Click(object sender, EventArgs e):** Geri dönüşüm işaretine benzeyen butona yani refreshButton’a basılınca olacak olayların olduğu methoddur. Buraya tıklayınca DoctorInfo\_Load methodu son hali ile geri döner.

**DoctorInfoExtendedForm sınıfı**

* **DoctorInfoExtendedForm(int doctorID):** Yapıcı methoddur. doctor\_id’yi parametre olarak alır ve bunu bu classın içinde değişken olarak tanımlanan doctorID’ye atar.
* **DoctorInfoExtendedForm\_Load(object sender, EventArgs e):**DoctorInfoForm çalıştırıldığı zaman yapılacak işlerin olduğu methoddur.Öncelikle var tipinde doc değişkeni tanımlanır ve veritabanından doctorID aracılığı ile Doctor ve User tablolarındaki bilgileri çekilerek doctorsBindingSource a bu değişken üzerinden atanır. Daha sonra deptComboBox’a Departman adları atanır.
* **doctorsBindingNavigatorSaveItem\_Click(object sender, EventArgs e):**Disket şeklinde duran save ikonuna tıklayınca yapılacak işlemlerin olduğu methoddur.Öncelikle bütün işlemler veritabanına kaydedilir ve veritabanında formun parametre olarak aldığı doctorID numarasında bir doktor olup olmadığına bakılır.Eğer böyle bir doktor varsa Doctor tipinde myDoctor nesnesi oluşturulur ve comboBox’tan seçilen departmanın bilgileri databaseden okunur ve myDoctor’ın department attributelerine atanır ve veritabanına kaydedilir.Daha sonra User ve Doctor tablolarının join işlemi sonucu elde edilen ve Doctor\_id’si doctorID’ye eşit olan doktorun bilgileri doctorsBindingSource’a atanır.
* **closeButton\_Click(object sender, EventArgs e):** Çarpı işareti şeklinde gözüken, closeButton ‘a tıklanınca olacak olayların olduğu methoddur. Buraya tıklayınca DoctorInfoExtendedForm kapanır.
* **bindingNavigatorDeleteItem\_Click(object sender, EventArgs e):**Silme işaretine yani Delete iconuna tıklanınca yapılacak işlemlerin olduğu methoddur. User ve Doctor tablolarının join işlemi sonucu , Doctor\_id’si doctorID’ye eşit olan doktor bulunur ve bilgileri ‘DeleteOnSubmit’ ile User tablosundan silinir.Daha sonra veritabanında değişiklikler kaydedilir.Eğer böyle bir doktor yoksa exception handling ile runtime hatanın önüne geçilir.

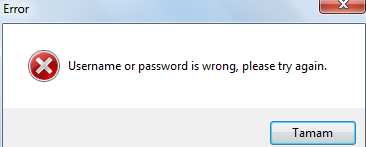
# KULLANICI KILAVUZU

* Başlangıç olarak veritabanında kayıtlı olan kullanıcıların kullanıcı adı – şifre – görev tablosu aşağıdaki gibidir. Bu verilerle programa giriş yapabiliriz.

|  |  |  |
| --- | --- | --- |
| **USERNAME** | **PASSWORD** | **JOB** |
| admin | admin | Admin |
| staff1 | staff1 | Staff |
| urodoc1 | urodoc1 | Doctor |
| carddoc1 | carddoc1 | Doctor |
| neurodoc1 | neurodoc1 | Doctor |
| entdoc1 | entdoc1 | Doctor |
| entdoc1 | entdoc1 | Doctor |
| gastrodoc1 | gastrodoc1 | Doctor |
| gsdoc1 | gsdoc1 | Doctor |
| gyndoc1 | gyndoc1 | Doctor |
| neurodoc2 | neurodoc2 | Doctor |
| oncodoc1 | oncodoc1 | Doctor |
| opthadoc1 | opthadoc1 | Doctor |
| orthodoc1 | orthodoc1 | Doctor |
| psydoc1 | psydoc1 | Doctor |
| urodoc2 | urodoc2 | Doctor |
| phydoc1 | phydoc1 | Doctor |
| gsdoc2 | gsdoc2 | Doctor |
| entdoc2 | entdoc2 | Doctor |
| entdoc3 | entdoc3 | Doctor |

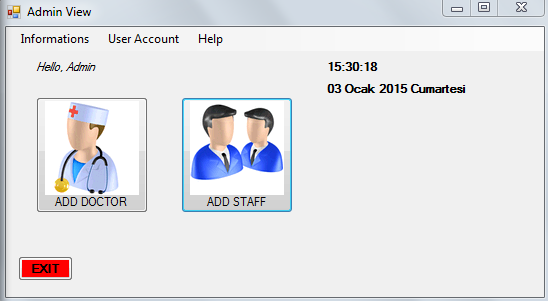
## Login Form

* Program ilk açıldığında gelen ekrandır(Resim 1). Admin, Doktor veya Staff’tan herhangi birisi kendi kullanıcı adı ve şifrelerini girerek **LOGIN** butonuna basarak sisteme giriş yapabilirler. Eğer yanlış bir kullanıcı adı – şifre girerseler hata mesajı alırlar(Resim 2). **EXIT** butonuna basarak programı kapatabilirler.

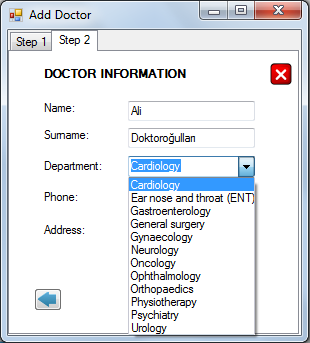
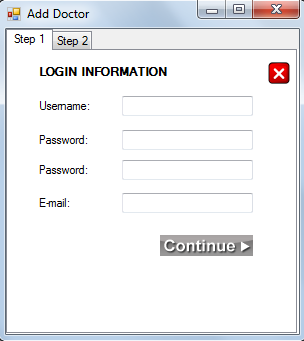
*\*Resim 2*  
*\*Resim 1*

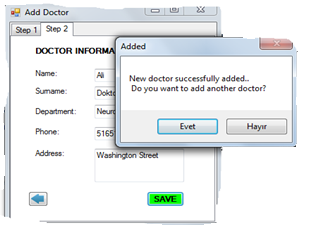
## Admin Form

* Login Form’dan admin olarak giriş yapıldığı zaman aşağıdaki gibi bir admin penceresi ekrana gelir(Resim 3).

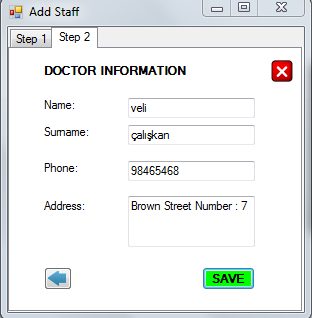
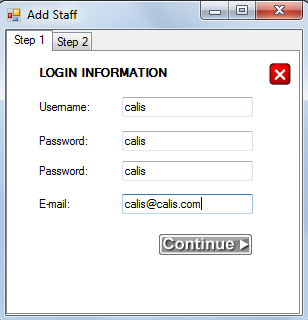
  
*\*Resim 3*

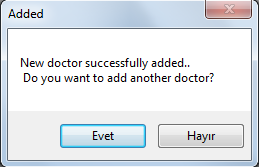
* Görüldüğü üzere formda bir sürü sekme vardır.Doktor eklemek için **ADD DOCTOR** butonuna , çalışan eklemek için **ADD STAFF** butonuna tıklamak gereklidir.
* Aşağıda **ADD DOCTOR** butonuna basılınca gelen pencere gözükmektedir (Resim 4). Pencereye eklemek istediğimiz doktorun bilgilerini girebiliriz. Step1’deki bilgileri girmeden step ikiye geçilememektedir. Bilgiler girildikten sonra Save tuşuna basarak bilgiler kaydedilir (Resim 6).

  
*\*Resim 4 \*Resim 5*

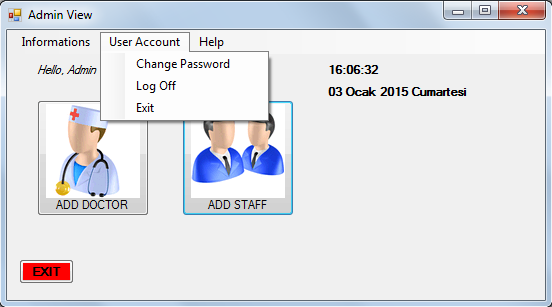
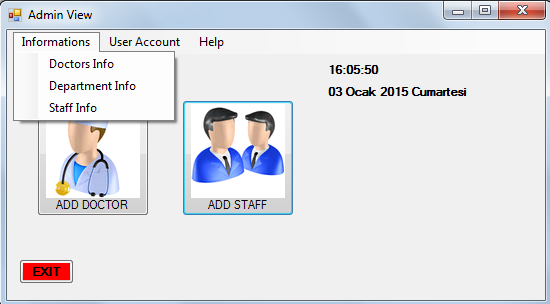
  
*\*Resim 6*

* Evet’e basarsak yeni bir doktor daha eklemek istediğimiz anlamına gelir ve aynı pencere bir daha açılır.Hayıra basarsak Admin View’a geri döneriz.Hayır bastığımızı düşünerek Admin View’dan devam edelim. Bu sefer çalışan eklemek için **ADD STAFF** butonuna basalım.Gelen pencere Add Doctor’a basınca gelen pencere ile aynı olup bilgileri girip çalışanı kaydedelim.Aşağıda bu işlemler anında programdan görüntüler bulunmaktadır (Resim 7-8-9).

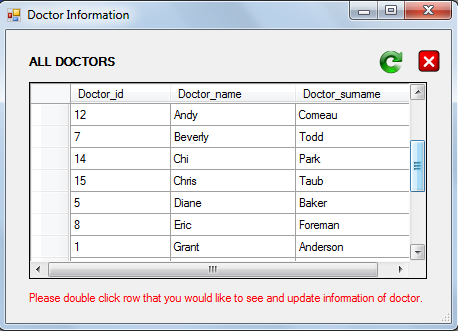
  
*\*Resim 7 \*Resim 8*

  
 *\*Resim 9*

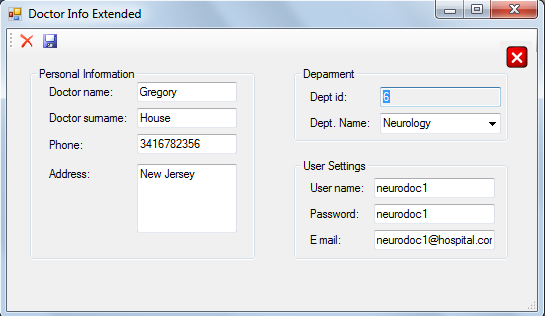
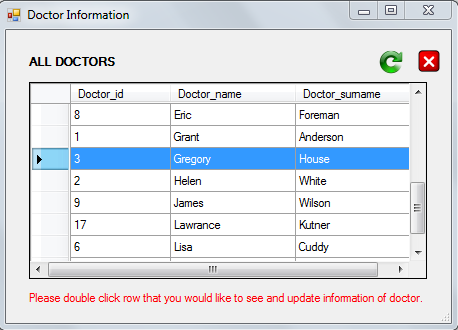
* Hayır’a basarak Admin View’a geri gelelim. Görüldüğü üzere Admin View’in üst tarafında menu vardır ve bu menu aracılığı ile çeşitli bilgileri elde edebiliriz(Resim 10).

  
 *\*Resim 10*

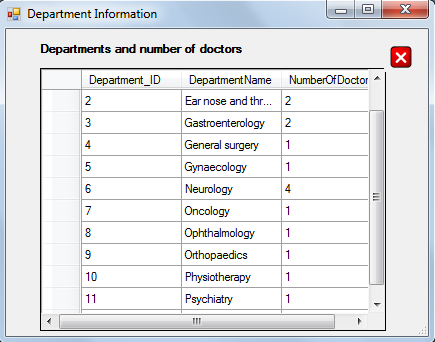
* Informations kısmından Doctor Info’ya tıklayarak gelen pencereye bakalım(Resim 11). Çarpıya tıklarsak bu form kapanır.Yeşil butona basarsak pencere kendini günceller.

  
*\*Resim 11*

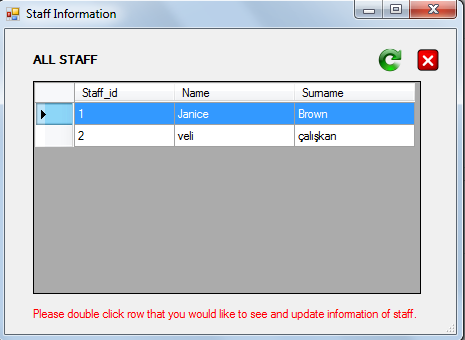
* Şimdi listeden doctor Gregory House’a çift tıklayarak Gregory House’un bilgilerini görüntüleyelim(Resim 12).

  
*\*Resim 12*

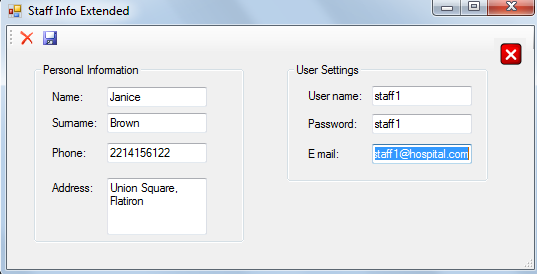
* Gelen pencereden görüldüğü üzere Dr.House’un kişisel bilgilerine ulaşılabiliyor.Buradan bu doktorun bilgilerini değiştirip soldaki save butonuna tıklayarak kaydedebiliriz ya da soldaki çarpı işaretine tıklayarak bu doktorun kayıtlarını silebiliriz.Eğer sağdaki çarpıya tıklarsak herhangi bir işlem yapılmadan bu pencere kapanır ve Doctor Information penceresine geri döneriz.Biz telefon numarasını değiştirip kaydedelim.Daha sonra sağ taraftaki çarpılara iki kere tıklayarak Admin View’a geri döneriz.
* Şimdi Informations kısmından Department Info ‘ya tıklayınca gelen pencereye bakalım(Resim 13). Hangi departman da kaç doktorun çalıştığı gözlenmektedir. Sağdaki çarpıya basarak pencereyi kapatabiliriz. Pencereyi kapatıp Admin View’a geri dönelim.

  
*\*Resim 13*

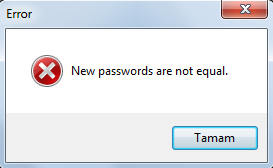
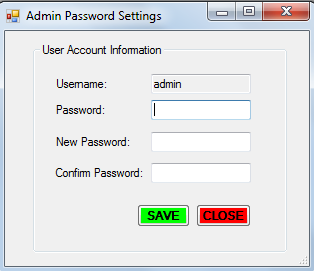
* Şimdi Informations kısmından Staff Info’ya tıklayarak gelen pencereyi inceleyelim(Resim 14). Görüldüğü üzere çalışanların listesi ile pencereyi güncelleme ve kapatma tuşları gözükmektedir. Ayrıca ADD STAFF butonuna tıklayınca eklediğimiz Veli Çalışkan isimli çalışanda burada gözükmektedir.

  
*\*Resim 14*

* Şimdi Janice Brown’a çift tıklayıp gelecek olan pencere bakalım(Resim 15). Aynı Doctor Info Extended penceresinde olduğu gibi tıkladığımız kişinin bilgileri gelmiştir.Bu sefer bir değişiklik yapmadan çıkalım.

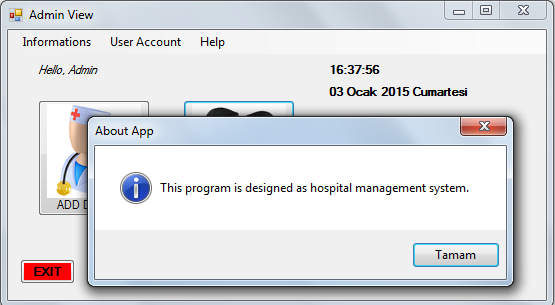
  
 *\*Resim 15*

* Şimdi User Account kısmından Change Password’e tıklayalım(Resim 16). Bu pencereden admin kendi parolasını değiştirebiliyor.Fakat bu işlemi yapabilmek için password kısmına eski parolasını , new password ve confirm password kısmına yeni parolasını girmesi gerekiyor. Şimdi confirm password u , new password’den farklı girerek **SAVE** tuşuna basarak kaydetmeye çalışalım. Görüldüğü üzere iki parolanın eşleşmediği hatasını veren pencere gelmiştir(Resim 17). Tamam butonuna basıp , herhangi bir işlem yapmamak üzere **CLOSE** butonuna basıp Admin View’a dönelim.



*\*Resim 16 \*Resim 17*

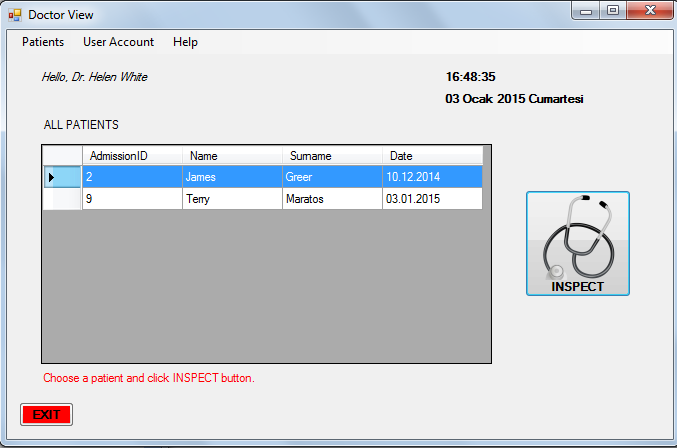
* Bu sefer Help kısmından About yazısına basalım(Resim 18). Görüldüğü üzere programın yazılış amacını belirten bir cümle ekrana gelmektedir. Tamam butonuna tıklayıp Admin View’a geri dönelim.

  
*\*Resim 18*

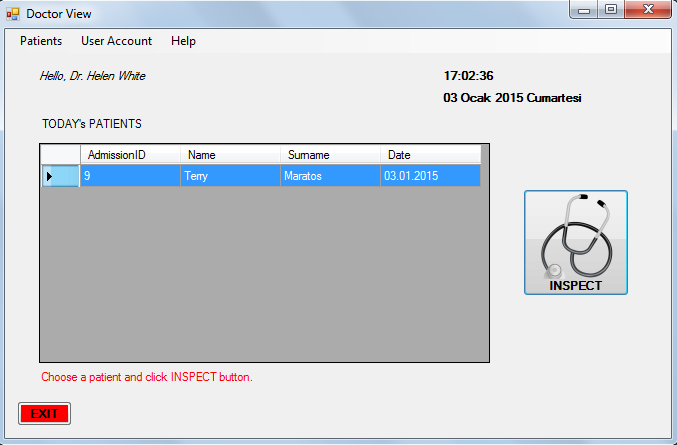
* Şimdi User Account’tan Log Off’a basarsak Login Form’a geri döneriz. Eğer Exit yazısına tıklarsak programımız komple kapanır.Biz Log Off’a basalım ve güvenli olarak çıkış yaparak Login penceresine geri dönelim.

## Doctor Form

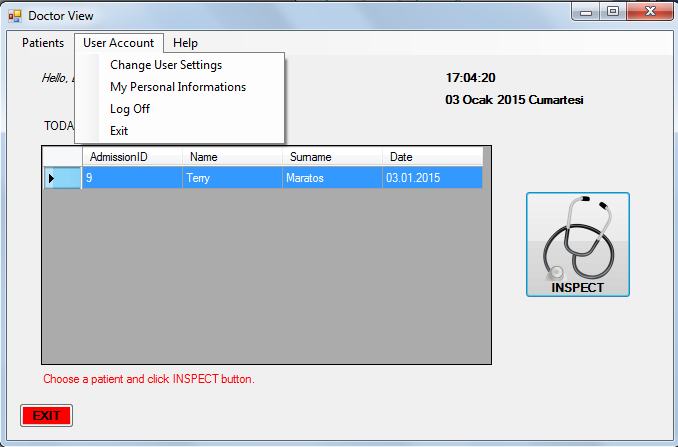
* Login’den doktor olarak girdikten sonra aşağıdaki gibi bir ekran karşımıza çıkmaktadır(Resim 19). Görüldüğü üzere gelen ekranda hasta listesi gözükmektedir.Gözüken bu liste Patients sekmesindeki All Patients ile aynıdır.

  
*\*Resim 19*

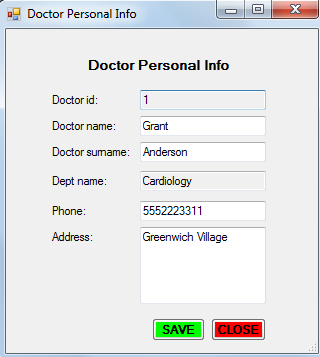
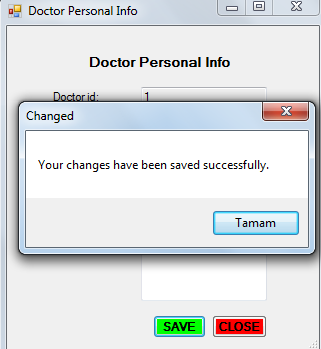
* Şimdi Patients kısmından Today’s patient kısmına tıklayalım ve bugün gelen hastaları görüntüleyelim(Resim 20).

  
*\*Resim 20*

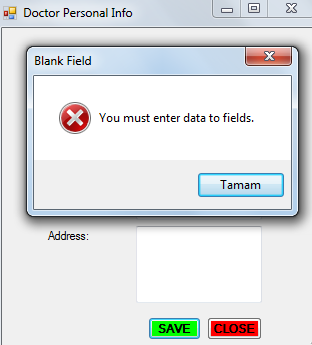
* Bu sefer başka bir menu sekmesi olan User Account’a tıklayalım.Aşağıdaki gibi bir görüntü gelmesi gerekmektedir(Resim 21). “My Personal Informations” ve “Change User Settings” sekmelerine tıklayalım.

*\*Resim 21*

* Doktorun kendi kişisel bilgilerini değiştirebileceği form *Resim 22*’de görüldüğü gibidir. Doktor kendi departmanı hariç adını, soyadını, telefon numarasını ve adresini değiştirip **SAVE** butonuna basarak bilgilerini hastane veritabanında kaydedebilir. Başarıyla kaydedildiğine dair bilgi mesajı *Resim 23*’deki gibi kullanıcıya gösterilir.

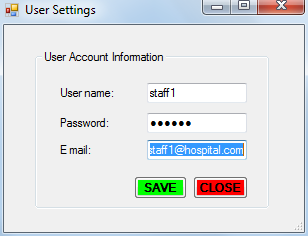
  
*\*Resim 22*  \**Resim 23*

* Eğer herhangi bir alan boşken **SAVE** butonuna basılırsa hata mesajı alınır. Bu hata mesajını *Resim 24*’de görebilirsiniz.

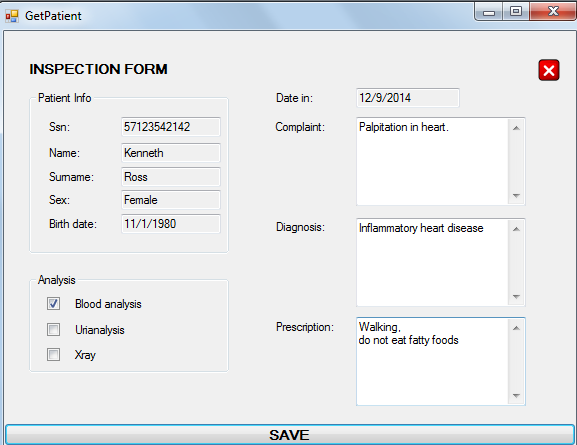


*\*Resim 24*

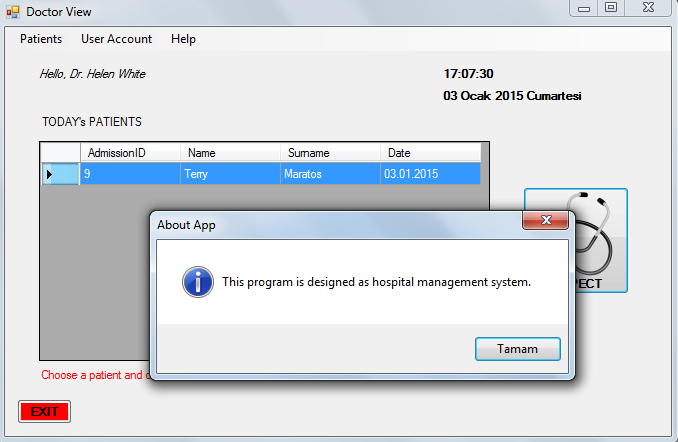
* **CLOSE** butonuna basarsanız bu form kapatılır.
* Doktor ya da Staff’ın sistemi giriş bilgilerini değiştirebileceği form aşağıdaki gibidir (Resim 25). Yeni bir kullanıcı adı, şifre veya mail girerek **SAVE** butonuna basılır ve veritabanında bilgiler güncellenir. **CLOSE** butonu ile ilgili form kapatılır.

  
*\*Resim 25*

* Şimdi de doktorun ana menüsünde var olan INSPECT butonuna basalım. Bir doktor bir hastayı seçtikten sonra muayene için gerekli işlemi yapınca bu form açılır (Resim 26). Doktor, gelen hastanın bilgilerinden şikayet, tanı ve reçete kısımları ile analiz kısımları olan kan tahlili, üre tahlili ve röntgen gibi bilgileri değiştirip güncelleyebilir. Aynı zamanda hastanın ad,soyad,doğum tarihi gibi kişisel bilgilerini görebilir ancak değişiklik yapamaz.
* Doktor gerekli kısımları doldurduktan sonra **SAVE** butonuna basarak muayene bilgilerini kaydeder. Ve **X** butonuna basarak ilgili formu kapatır.



*\*Resim 26*

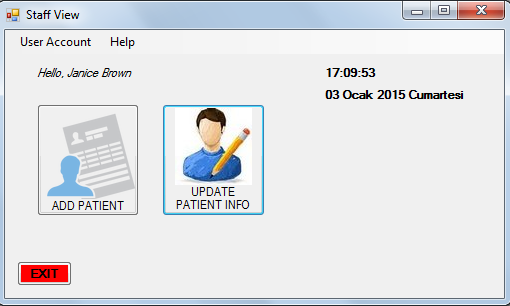
* Buradan Help kısmından about’a tıklayalım ve projenin yapılış amacını görelim(Resim 27).  
    
  

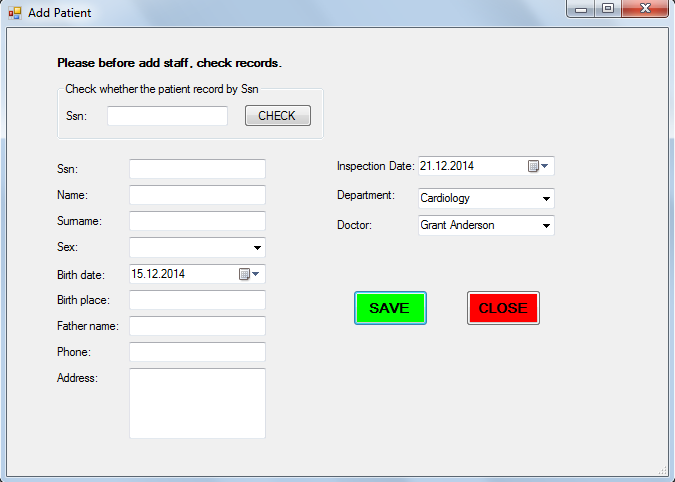
*\*Resim 27*

* Şimdi User Account’tan Log off yaparak çıkalım ve login form’a dönelim.

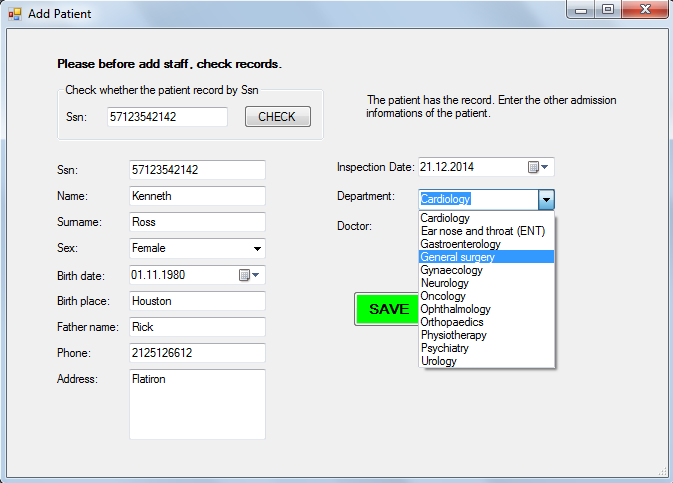
## Staff Form

* Login Form’dan çalışan olarak giriş yapılınca karşımıza çıkan ekran aşağıdaki gibidir(Resim 28). Bu ekranda güncel ve dinamik olan saat ve tarih, hoşgeldiniz mesajı içeren label mevcuttur. Ayrıca yeni bir hasta kaydı için **ADD PATIENT** butonu, varolan hastaların güncellenmesi için **UPDATE PATIENT INFO** butonu yer almaktadır.
* Bu ekrandan **ADD PATIENT** butonuna tıklayalım. Resim 29’daki gibi bir ekran karşımıza çıkmaktadır.

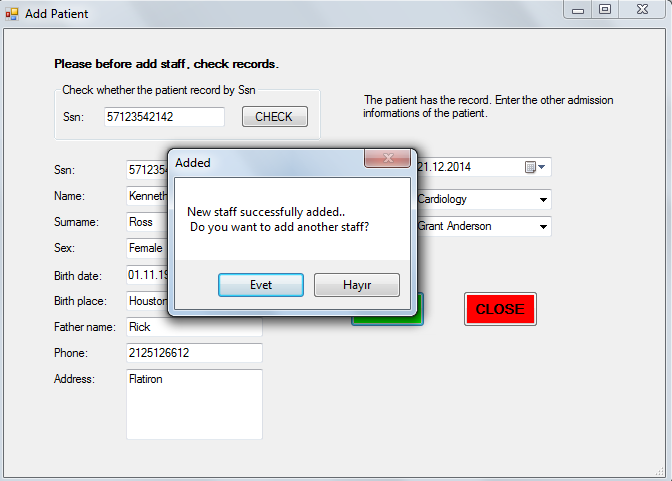
  
 *\*Resim 28*

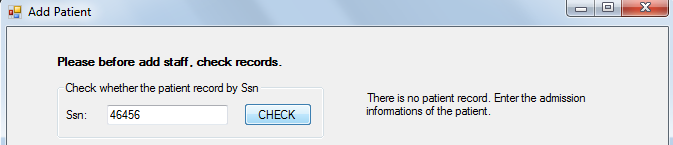
*\*Resim 29*

* Şimdi bu ekrandan daha önce hastanın hastanemize gelip gelmediğini kontrol etmek için Ssn numarasını girip **CHECK** tuşuna basalım. Görüldüğü üzere hastanın daha önceden gelmiş olduğu anlaşılıp nüfus bilgileri otomatik olarak ekrana gelmektedir.Ekranın sağ tarafından hastanın gitmek istediği bölümü seçip **SAVE** tuşuna basarak hasta kaydını gerçekleştirebiliriz.

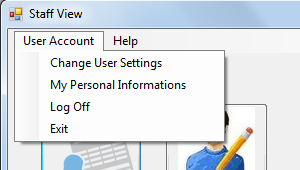
  
*\*Resim 30*

* Bunu yapınca karşımıza bir MessageBox gelmektedir(Resim 31). Evet butonuna tıklayıp yeni bir hasta daha eklemeyi deneyelim. Bu sefer olmayan bir hasta SSN’i girelim ve aşağıdaki gibi bir yazının penceremizde olduğuna dikkat edelim(Resim 32).

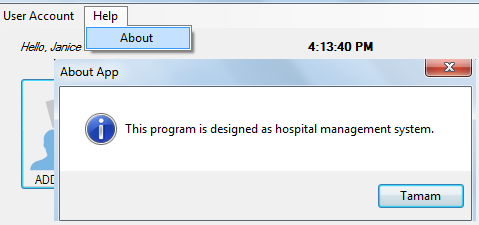
*\*Resim 31*

 *\*Resim 32*

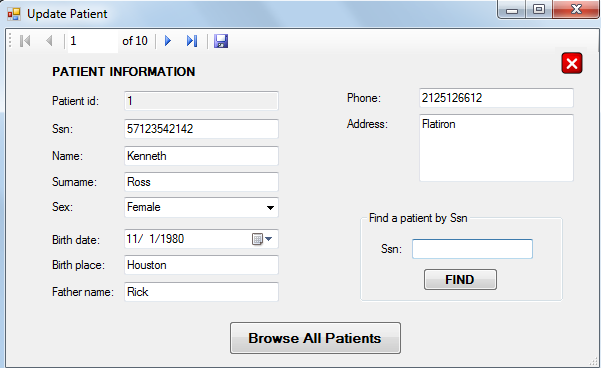
* Böyle bir durumda hasta bilgileri tek tek elle girilir ve SAVE’e basarak kaydedilir.Eğer çıkış yapmak istiyorsak CLOSE tuşuna basarak Staff View’a geri dönebiliriz.
* *User Account* menü item’ına basıldığında kullanıcının kendi kişisel bilgi ve sisteme giriş bilgilerini değiştirmesi için gereken menü item’ları (*Change User Settings* ve *My Personal Informations*) ve de bunlara ek olarak kullanıcının sistemden çıkışını sağlayan Log Off item’ı ile programın kapatılmasını sağlayan *Exit* item’ı bulunmaktadır (Resim 33).

 *\*Resim 33*

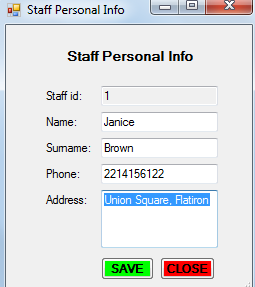
* *Help* menu item’ı ise program hakkında bilgi veren *About* item’ını içermektedir(Resim 34).

  
*\*Resim 34*

* Staff UPDATE PATIENT butonuna basarak hastane veritabanında kayıtlı hastaların kişisel bilgilerine erişip üzerinde değişiklik yapıp kaydedebileceği ekrana ulaşır (Resim 35). Oklar(|< <> >|) yardımıyla hastalara sırayla bakabilir. Aynı zamanda SSN numarası elinde olan bir hastayı ‘Find a patient by Ssn’ alanına girerek **FIND** butonuna basarak o hastanın bilgilerine ulaşabilir. Değişiklik yaptıktan sonra bilgilerin kaydedilmesini sağlamak için **Save** item’ına basabilir. Bir hastayı arattıktan sonra tekrar tüm hastalara sırayla bakmak için **Browse All Patients** butonuna basabilirsiniz. Formu kapatmak için **X**(Close) butonuna basabilirsiniz.

  
*\*Resim 35*

* Staff Menu’den “My Personal Informations” sekmesine tıklarsa kendi kişisel bilgilerini görüntüleyip değiştirebileceği ekrana ulaşır (Resim 36). Değişiklikleri yaptıktan sonra **SAVE** butonuna basarak kaydedebilir. **CLOSE** butonuna basarak ilgili formu kapatabilir.

  
*\*Resim 36*