T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SAĞLIKLI SPOR İÇİN İZLEME SİSTEMİ MODELİNİN HAZIRLANMASI

YÜKSEK LİSANS TEZİ

PARVIZ ABBASOV

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Ahmed BABANLI

Mart. 2015

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SAĞLIKLI SPOR İÇİN İZLEME SİSTEMİ MODELİNİN HAZIRLANMASI

YÜKSEK LİSANS TEZİ

PARVIZ ABBASOV

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Ahmed BABANLI

Mart. 2015

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜ

Yüksek Lisans Tez Onay Belgesi

Enstitümüz Bilgisayar Mühendisliği Ana Bilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı **Y1213.010003** numaralı öğrencisi **Parviz ABBASOV** 'nun **"SAĞLIKLI SPOR AKTİVİTESİ İÇİN İZLEME SİSTEMİ MODELİNİN HAZIRLANMASI"** adlı tez çalışması Enstitümüz Yönetim Kurulunun 04.02.2015 tarih ve 2015/02 sayılı kararıyla oluşturulan jüri tarafından oy birliği ile Tezli Yüksek Lisans tezi olarak kabul edilmiştir.

Öğretim Üyesi Adı Soyadı	**İmzası**
Tez Savunma Tarihi :12/03/2015	
1)Tez Danışmanı: Prof. Dr. Ahmad BABANLI	.....
2) Jüri Üyesi : Yrd. Doç. Dr. Vassilla ABDULOVA	.....
3) Jüri Üyesi : Yrd. Doç. Dr. Metin ZONTUL	.....

Not: Öğrencinin Tez savunmasında **Başarılı** olması halinde bu form **imzalanacaktır**. Aksi halde geçersizdir.

Yemin Metni

Yüksek Lisans tezi olarak sunduğum "Sağlıklı Spor İzleme Sistemi Modelinin Hazırlanması" adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya'da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (03/02/2015)

Parviz ABBASOV /

ÖNSÖZ

Bu tez çalışmamda beni yönlendiren ve bana yardımcı olan değerli hocam Prof. Dr. Sayın Ahmed BABANLI'ya ve eğitimim boyunca emeyi geçen tüm hocalarıma teşekkür eder, saygılarımı sunarım.

Şubat 2015 YL Öğrencisi Parviz ABBASOV

İÇİNDEKİLER

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

SAĞLIKLI SPOR İÇİN İZLEME SİSTEMİ MODELİNİN HAZIRLANMASI

ÖZET

Fiziksel aktivite, her yaşta sağlığa yararlıdır. Dünya Sağlık Örgütü'ne göre, fiziksel aktivite yetersizliği %17 kalp hastalıkları ve diabete, % 10 meme kanseri ve kolon kanserine neden oluyor. Sık ve düzenli fiziksel egzersiz, bağışıklık sistemini güçlendirir ve kalp hastalığı, kardiyovasküler hastalıklar ve obezite gibi "refah hastalıkları"nı önlemeye yardımcı olur. Üstelik ruh sağlığı gelişimi ve zihinsel süreçler üzerinde olumlu etkiler bırakır. Fiziksel akivite zihinsel sağlık sorunlarına neden olan, kortizol duzeyini de azaltır.

Herkes egzersizden eşit yararlanamıyor. Herkesin fiziksel aktiviteye çok farklı bireysel tepkileri vardır: Mesela, orta derecede aktiviteye bazıları normal, bazıları da çift oksijen alımı ile tepki verebilir. Sağlık kontrolsüz, dinlenmesiz ve aşırı egzersiz yapmak zararlı olabilir. İnsan bünyesinin kaldıracağı spor seviyesini aşmak sağlık için ciddi problemler yaratabilir. Bunun için kişinin mevcut fiziksel ve sağlık durumunun spor öncesi, spor zamanı kontrol ve analiz edilmesi bireyin daha sağlıklı ve doğru şekilde spor yapmasına yardımcı olucaktır.

Bu tez çalışmasının en önemli amacı fiziksel aktivite zamanı bireyinlerin sağlık bilgilerinin kolayca kontrol edilebilen sağlık izleme sistemi geliştirmektir. Bu tezde 2 ana amaç hedeflenmiştir: Birinci amaç fiziksel aktivite öncesi ve aktivite zamanı bireyin sağlık bilgilerini otomatik olarak izlemek, kaydetmek ve yaşanabilecek sağlık risklerini otomatik olarak alarmla danışmanı ve bireyi uyararak önlemekdir. Diğer amaç ise kaydedilen bilgilerle, sağlık ve spor danışmanına devamlı olarak her aktivite gününde bireyin gerçekleştirdiği fiziksel aktivite yoğunluğunu daha doğru ve kanıtsal belirlemesi, analiz etmesi ve bu prosedür sonucunda bireye daha da uygun çalışma programı çıkarması için yardımcı olmaktır.

Anahtar Kelimeler: *Spor İzleme Sistemi, Teknoloji ile Sağlıklı Spor, Biyometrik Sensör.*

PREPARATION OF THE MODEL OF HEALTHY SPORT MONITORING SYSTEM

ABSTRACT

Physical activity is beneficial to health at all ages. Frequent and regular physical exercise boosts the immune system and helps to prevent of heart disease, cardiovascular disease and such as obesity "welfare diseases". In addition, it leaves a positive effect on health and mental processes. It reduces the level of cortisol level, which causes the mental health problems.

Everybody does not get equal benefit from exercise. All have different individual responses to physical activity. For example, some of them can react normal to regular activity, some of them can react with double oxygen uptake. Having extreme exercise with uncontrolled health and without rest can be harmfull for human life.

The working out more than body endurance can cause to serious health problems. For this reason, the measuring and analyzing of current physical and health status of individuals can help them to engage with right and healthy sport.

The aim of this thesis is to desing the e-health system that can easily controll an individual’s health status during the physical activity.

This thesis is designed in two main objectives: First is automatically to monitor, record an individual's health status before and during physical activity and in the case of risk automatically to warn the consultant and individual to prevent health harm. Other is a helping the health and fitness consultants to do more accurate and probative analyzing and making appropriate work out plans for individuals with recorded information.

Keywords: *Sport Monitoring System, Healthy Sport with Technology, Biometric Sensor.*

1. GİRİŞ

Düzenli, doğru ve güvenli fiziksel aktivite bir bireyin sağlığını koruması ve hatta iyileştirmesi için gereklidir. Kontrollü ve sağlıklı sporun yararları daha detaylı olarak aşağıda belirtilmiştir.

Kontrollü ve sağlıklı spor -

- kalp-damar sistemini güçlendirir;
- kan kolestrol seviyesine olumlu efekti vardır;
- hipertansiyon sıklığını azaltabilir ve kan basıncını normalleştirir;
- kanseri önlemeye yardımcı olur;
- vücut ağırlığını düzenlemeye yardımcı olur;
- duruşu geliştirir ve ortopedik bozuklukları azaltmaya yardımcı olur;
- dayanıklılık, kas gücü, esneklik ve denge, ayrıca eklem ve kemik yapısı gelişimi sağlar;
- kan pıhtılaşma bozukluklarını azaltır, kandaki oksijen miktarını artırır;
- ve kan pıhtısı oluşumunu engeller [1];
- endişe, huzursuzluk, uyku sorunları ve depresyon riskini azaltır;
- yaşam kalitesini artırır;

Fiziksel aktivite zamanı sağlık kontrolü neden yapılmalı?

- mevcut sağlık koşullarını, risklerini veya sınırlamalarını belirlemek için;
- mevcut fitness düzeyini değerlendirmek için;
- spor için fitness hedefleri, ilgi ve motivasyonları belirlemek için;
- uygun antrenman seçenekleri belirlemek için;
- ilerlemeyi izlemek ve programın başarısını değerlendirmek için; yöntemler kurmak için bireyle fitness danışmanı arasında uygun beklentileri kurmak için ;

Fiziksel aktivitemizi izleyebilen, gelişmemize ve sağlıklı düzgün spor yapmamıza yardımcı olan sağlık ve fitness uzmanları vardır [2]. Günümüzde fitness salonlarında

çalışan spor ve sağlık uzmanları bireylerin belli bir kriterilerine göre çalışma programları sunuyorlar. Lakin bazen bu programlar bazı kişiler için çok hafif diğerleri için daha zor olabiliyor. Bu durumda bireyler harcadıkları zamanı iyi değerlendiremiyor. Herkese özel, spesifik ve daha uygun çalışma programları çıkarmak için herkesin özgü tüm sağlık ve psikolojik bilgileri hemen-hemen her spor öncesi, spor zamanı ve sonrası izlenmelidir ve bir sonrakı çalışma programı bu bilgilere dayanarak hazırlanmalıdır. Bu sistemin bir diğer avantajı ise antrenman zamanında bireyde herhangi bir negativ etki oluşumu algılandığı zaman alarm öterek kişiye ve spor uzmanına anında haber verilmesidir. Tüm bu prosedürün spor uzmanı geleneksel yöntemlerle gerçekleştirmesi neredeyse imkansızdır.

Geleneksel yöntemlerden ayrılıp teknolojik yöntemlerle bu problemler daha az zaman içerisinde, daha doğru sonuça yaklaşarak çözülebilir ve bu yöntemle sağlık ve spor uzmanlarının bireyi daha doğru şekilde yönlendirmesi söz konusu olabilir. Bu tez çalışmasında ilgili problemler ele alınmış ve bir spor izleme sistemi modeli geliştirilmiştir.

Bölüm 2'de spor alanında kullanılan tıbbi uygulama örnekleri ve onların özellikleri incelenmiştir. Buraya dahildir : Nike + iPod sistemi, Endomondo Spor İzleme Sistemi. Onların çalışma özellikleri ve eksik yönleri gösterilmiştir.

Bölüm 3'de Arduino tabanlı sağlık platformu açıklanmıştır. Bu platformda kullanılmakta olan sağlık sensörleri (Nabız, Kandakı Oksijen, Vücut Isısı ve s.) hakkında bilgi verilmiş, çalışma prensibi, bileşenleri ve C++ dilinde yazılmış kaynak kodları gösterilmiştir.

Bölüm 4'de Spor İzleme Modeli için Ms Sql Server oluşturulmuş veritabanının yapısı açıklanmış, tabloların ve onlar arasında olan ilişkilerin özellikleri gösterilmiştir.

Bölüm 5'de oluşturulmuş Spor İzleme Sistemin'de sistem kullanıcı arayüzlerinin tasarımı, verilerin kontrolü ve işlenmesi modülü, haberleşme modülü ve sistemin genel testi yapılmıştır.

Sonuç bölümünde bu sistemin avantajları, kullanım alanları ve gelecek için perespektifleri gösterilmiştir.

2. SPOR ALANINDA KULLANILAN TIBBİ UYGULAMALAR

2.1. Spor Uygulamalarda Kullanılan Kablosuz Biyometrik Sistemler

Son zamanlarda, tıbbi uygulamalar ve spor uygulamaları için kablosuz biometrik sistemlere ilgi hızla artmaktadır. Biyometrik sensörler insanların günlük hayat ,fiziksel aktivite veya hastalık durumlarında sağlık bilgilerini uzaktan, gerçek zamanlı olarak kontrol etmesinde yardımcı olabilir. Şuan sağlık hizmetleri piyasası WiFi ve diğer kablosuz LAN teknolojileri için en hızlı büyüyen piyasalar arasındadır [3]. Bugünün giyilebilir kablosuz sensör teknolojileriyle öngörülen fiziksel aktivite zamanı yaşamsal bulguları kesintisiz izleme sistemi kurma imkanını sağlamaktadır.

2.2. Spor Alanında Kullanılan Tıbbi Uygulama Örnekleri

2.2.1. Nike + iPod sistemi

Nike ve Iphone tarafından hazırlanan "Personal Trainer" isimli yenilikçi bir projedir. Bu projede iphone veya ipod, ipod+ nike sensörü (adım sayar sensör), sensör ve ipod (iphone) alıcısı kullanılmıştır ve bu sisteme ek olarak nabız (pulse) sensörü de kullanılabilir. Bu sistemin kullanıcılara sunduğu seçenekler bunlardır:

- Zaman hedefli çalışma.
- Mesafe hedefli çalışma.
- Kalori yakma hedefli çalışma.

Bu seçeneklerin hepsi adım sayar sensör yardımıyla gerçekleştiriliyor. Bu uygulama sayesinde insanlar spor aktivitesi zamanı kendisinin ne kadar kalori harcadığını, ne kadar yol katetdiğini, nabzını ipod veya iphone cihazı kullanarak izleyebilir.

2.2.2. Endomondo spor izleme sistemi

Fitness çalışması uygulamasıdır. Bu uygulamanın amacı cep telefonunuzu kişisel antrenörünüze çevirmektir. Bu uygulama kullanıcıya, bulunduğu lokasyonu, bulunduğu arazinin coğrafik durumunu, hedef mesafesi ve kalan mesafeyi, hedef zamanı ve kalan süreyi, ortalama hızını, harcadığı kaloriyi, maksimal ve ortalama kalp atış bilgilerini izlemeyebilme özelliğini sunuyor. Bu projede akıllı telefon, adımsayar sensör ve nabız (pulse) sensörü kullanılmıştır.

2.3. Spor alanında kullanılan uygulamaların eksik yönleri

Günümüzde spor uygulamalarının en önemli açığı bireyin fiziksel aktivite öncesi tüm sağlık bilgilerinin otomatik ve hızlı kontrol edilip fiziksel aktivite yapmaya uygun olup olmadığının belirlenmemesidir. Diğer yandan ise egzersiz zamanı kesintisiz ve gerçek zamanlı tüm sağlık bilgileri kontrolünün yapılmamasıdır. Bireylerin fiziksel aktivite öcesi ve aktivite zamanı kontrollerinin sağlanması spor ve sağlık danışmanına bireylerin zamanlarını daha doğru şekilde değerlendirmesini, sağlıklı ve güvenli fiziksel aktivite yapmalarında yakından yardımcı olmasını sağlayacaktır.

Araştırmasını yaptığım projenin diğer projelerden farkı spor yapan bireyin sağlık durumunun fiziksel aktivite zamanı 9 tane biyometrik sensör yardımıyla sürekli uzaktan izlenerek sağlık riski oluşması durumunu minimuma indirilmesi ve her hangi bir risk durumunda alarm sensörüyle spor ve sağlık danışmanının ve bireyin uyarılmasıdır. Diğer yandan ise spor ve sağlık danışmanına bireye özgü çalışma programı hazırlaması için gereken tüm sağlık bilgilerinin sağlanmasıdır. Bu bilgileri analiz ederek bireyin daha sağlıklı ve uygun spor yapması ve zamanını daha iyi değerlendirmesine yardımcı olunabilir.

Kan basıncı kontrolü amacı - Fiziksel aktivite yoğunluğu ön hipertansiyon ve hipotansiyon kan basıncını azaltır. Yüksek kan basıncı spor zamanı tehlikeli olup, kalp krizi, felç, böbrek hasarı, omuz ve sırt, göğüs ağrısına yol açabilir [4,5]. Söz konusu durumun yaşanmaması için antrenman öncesinde herkes için kan basıncı kontrolü yapılması çok önemlidir. Yüksek kan basıncına sahip insanlar güçlü egzersiz antrenmanından kaçınmalıdırlar [6].

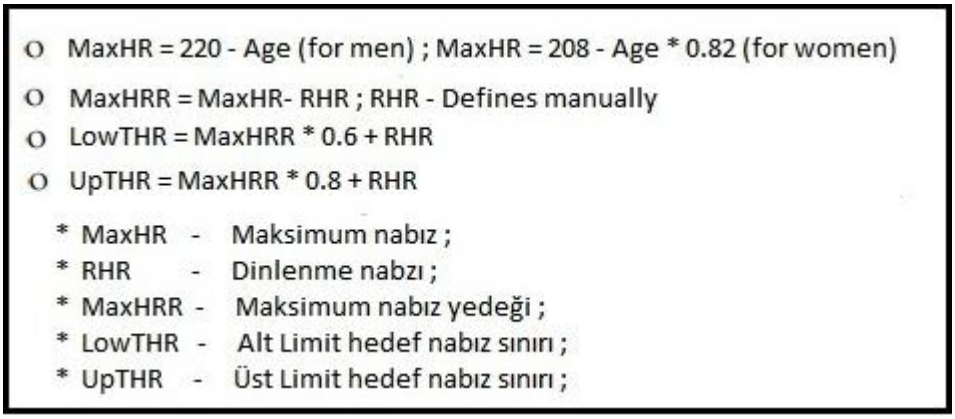
Kan şekeri kontrolü amacı - Egzersiz bazı insanlarda insulin duyarlılığını, insülin bağımsız glukoz alımını ve kas glukoz kullanımını artırıyor. Vücudunda bu gibi etkiler ortaya çıkan aktif kişilerde hipoglisemi riski daha çoktur. Düzenli aktivite, sağlıklı beslenme ve ideal kiloda olmak kan şekeri hastalığından korur [7]. En iyi sağlıklı yaşam için, uzmanlar haftada 150 dakika orta yoğunlukta fiziksel aktivite yapılmasını öneriyor. Kan şekeri seviyesine göre kurallar - (mmol / L) milligram başına litre veya (mg / dL) milimol başına desilitre olarak ölçülür. Egzersiz öncesi 100 - 250 mg/dL aralığı (5.6 -13.9 mmol/L) güvenli bir kan şekeri aralığıdır [8]. Uygulama arka planda glucometre sensöründen fiziksel aktivite öncesi aldığı bilgiyi sağlık kurumları tarafından belirlenen aralıklara göre analiz edecektir ve eğer bu

bilgiler aralıklara uygun gelmezse otomatik olarak uyarı verip, kişinin şuan fiziksel aktivite yapmasının sağlık durumuna göre uygun olmadığını bildirecektir.

ECG kontrolü amacı - Kalp dört odadan oluşan bir kas pompadır. Atrium denilen iki üst ve Vertrikül (karıncık) denilen iki alt odadan oluşmaktadır. Doğal elektrik sistemi kalp kasının kanı kalpten akciğerlere ve tüm vücuda pompalamasını sağlıyor. Elektrokardiyogram (ECG veya EKG) kalbin elektriksel aktivitesini denetleyen bir testtir.

Spor aktivitesi boyu ECG testi uygulandığı zaman spor ve sağlık danışmanı uzaktan kişinin kalbinin egzersiz yükünü idare etdip edemediğini kontrol edebilir ve aşırı yorgunluğu, fazla egzersiz yapıldığı zaman tespit edip önlemini alabilir [9,10].

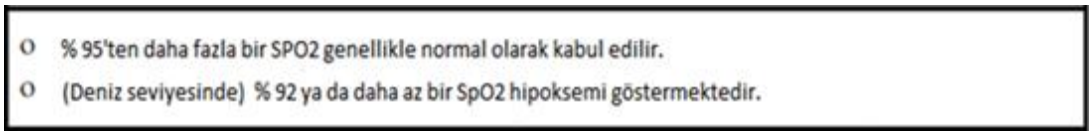
Nabız kontrolü amacı - Ter her zaman egzersiz yoğunluğunun en iyi göstergesi olmayabilir. Bunun için, aktivite zamanı kalbi izlememiz gerekir. Egzersiz sırasında kaydedilen nabızlar kalbinin ne kadar yoğun çalıştığını göstermektedir [11]. Spor ve sağlık danışmanı Nabız (pulse) senörü yardımıyla bilgisayar veya akıllı telefon üzerinden fiziksel aktivite zamanı kişilerin kalp atış hızı bilgilerini izleyebilir. Uygulama ise arka kısımda en çok kullanılan Karvonen metoduyla (Şekil 2.1) kişiye göre özel hedef nabız bölgesini belirleyecek ve egzersiz sırasında elde edilen bilgileri özel hedef nabız bölgesine göre analiz ederek her hangi bir limit aşımı sırasında spor ve sağlık danışmanı ve kişiye sinyal yardımıyla uyarı verecektir. Ayrıca bu bilgiler uzmana kişinin kalbinin çalışma performansını analiz etmesine ve kişiye özel bir çalışma programı çıkarmasına ve ne kadar yoğunlukta spor yapması gerektiğini belirlemesine de yardımcı olabilir [11].



Şekil 2.1 : Karvonen Formulü. Nabız Ölçümlerinin Değerlendirilmesi

Kandakı oksijen kontrolü amacı - Vücut ısısı, nabız, solunum, kan basıncı ve son zamanlarda yeni eklenen oksijen satürasyonu insan hayatında en önemli sağlık faktörlerdir. Kandakı oksijen satürasyonu pulse oksimetre sensörü yardımıyla ölçülmektedir.

Uygulama arka planda pulse oksimetre sensöründen gelen bilgilerle kişinin kanındakı oksijen yüzdesini kontrol altında tutar ve bu bilgiler sağlık kurumları tarafından belirlenen intervaller (Şekil 2.2) dışına çıktığında alarm öterek fitness danışmanına ve kişiye (sinyal sensörü yardımıyla) uyarı verir [12, 13].



Şekil 2.2 : Kandakı Oksijen Ölçümünün Değerlendirilmesi

EMG kontrolü amacı - elektromyogram (EMG) dinlenme ve fiziksel aktivite sırasındaki kasların elektriksel aktivitesini ölçer. Sinyaller tıbbi anormallikleri ve kas aktivasyon seviyyesini, tespit etmek için analiz edilebilir.

Galvanic deri tepkisi kontrolü amacı - Galvanik deri tepkisi olarak bilinen deri iletkenliği, cildin nem seviyyesine gore elektriksel iletkenliği ölçmek için bir yöntemdir. Ter bezleri sempatik sinir sistemi tarafından kontrol edilir ve güçlü fiziolojik değişiklikler, stress zamanı ciltin elektriksel dirençini değişebiliyor. Terleme düzeyi yükseldikçe derinin elektrik direnci düşüyor.

GSR sensörü yardımıyla uzaktan bilgisayar veya akıllı mobil cihaz üzerinden spor ve sağlık danışmanları fiziksel aktivite zamanı kişilerin gerçek zamanlı fizyolojik tepkilerini izleyip, aşırı yüklenme ve fazla egzersiz yapılmasını önleyebilecek. Elde etdiği bilgilerle bir sonrakı gün için her kişiye özel ve daha uygun çalışma programı tasarlayabilicektir.

Vücut sıcaklığı kontrolü amacı - Aşırı ısı egzersizi vücutta meydana gelen kardiyovasküler, metabolik ve nöromüsküler değişikliklerемgöre egzersiz performansını sınırlayabilecek potansiyele sahiptir. Egzersiz sırasında, ter buharlaşmasına tepki olarak kan basıncı artar. Egzersiz vücut sıcaklığına ve susuzluk artışına neden olup, bu durum ağır egzersiz sırasında daha da artabilir [14]. Bu da kişide sıcak çarpmasıyla sonuçlanabilir [15]. Bu riskin yaşanmaması fiziksel aktivite zamanı vücut sıcaklığı sensör yardımıyla kontrol edilebilir ve her aşırı vücut

ısısı belirlendiğinde bu durum spor ve sağlık danışmanına ve bireye alarmla bildirilebilir.

Hava akışı kontrolü amacı - Anormal solunum oranları ve solunum hızı değişiklikleri önemli fizyolojik istikrarsızlıkların geniş bir göstergesidir, ve birçok durumda, solunum hızı bu istikrarsızlık önemli bir göstergelerinden biridir [16]. Maksimal oksijen tüketimi bireyin fiziksel aktiviteye uygunluğunu ve uzun süreli egzersiz sırasında dayanıklılık kapasitesini belirliyor [17].

Bu nedenle, insanın fiziksel aktivite zamanı, durumunun bir göstergesi olarak solunum hızını izlemede önemlidir. Hava akış sensörü, hipoksemi ve apneye karşı erken uyarı sağlayabilir.

3. ARDUINO TABANLI SAĞLIK SENSÖR PLATFORMU

3.1. Arduino Platformu

Arduino açık kaynak kodlu bir elektronik cihazdır. Bu da yazılımcılara bu platforma üzerinde yazılım geliştirmeye, maliyyeti az olan yenilikçi projeler yapmaya olanak sağlıyor. Arduino donanımında işlemciler Atmel markalı olup, C++ programlama dilini desteklemektedir. Arduino IDE editor yazdığımız programı derleyerek, karta yükleme işlemini yapmaktadır.

3.2. Arduino Sağlık Sensörleri

Arduino Sağlık Sensör Platformu, çeşitli biyometrik, tıbbi ve spor uygulamalarını gerçekleştirmemize imkan sağlıyor. Bu sağlık sensör platformu sağlık izleme sistemi geliştirilebilmesini sağlayan aşağıda isimleri listelenen 10 farklı sensör tipi sunmaktadır: (Şekil 3.1)

- Nabız, kandakı oksijen (SPO2)
- Hava akışı (Respiration)
- Vücut ısısı (Temperature)
- Elektrokardiyogram (ECG)
- Glükometre (Glucometer)
- Galvanik vücut tepkisi (Galvanik Skin Response)
- Kan basıncı (Blood Pressure)
- Patient position (Accelerometer)
- Kas/elektromyografi (EMG).

Şekil 3.1 : Arduino Tabanlı Sağlık Platformu

Bu sensörlerden gelen gerçek zamanlı ve kesintisiz bilgilerle yapılabilecek otomatik süreçlerden bazıları:

- Bir hastanın evinde uzaktan tedavi görmesi, tedavi sonuçlarının izlenmesi, sağlık bilgilerinin sürekli kontrolü;
- Fiziksel aktivite yapan bireyin, uzaktan sağlık durumunun sürekli izlenmesi, aktivite zamanı sağlık güvencesinde olması ve bu bilgilerin analiz edilip yeni çalışma programı hazırlanması;
- Yaşlı veya bakıma muhtaç bireylerin uzaktan sağlık durumunun denetim altına alınması;
- Sağlık düzeyini bozabilecek alanlarda çalışanların sağlık bilgilerinin denetim altına alınması;

3.2.1. Kandakı oksijen ve nabız sensörü (SpO2)

Nabız oksimetresi fonksiyonel hemoglobin oksijen satürasyonu ve nabzı (doygunluğunu) belirten bir noninvaziv yöntemdir. Bu teknoloji 1970'lerden beri mevcut olmasına karşın, son gelişmeler bu teknolojinin boyutunu ve maliyetini azaltmıştır. (Şekil 3.2) Bu sensör herhangi bir ortamda, fiziksel aktivite zamanı ,yoğun bakım, ameliyat, acil durumda vs. kullanılabilir.

Şekil 3.2 : Nabız ve Oksimetre Sensörü

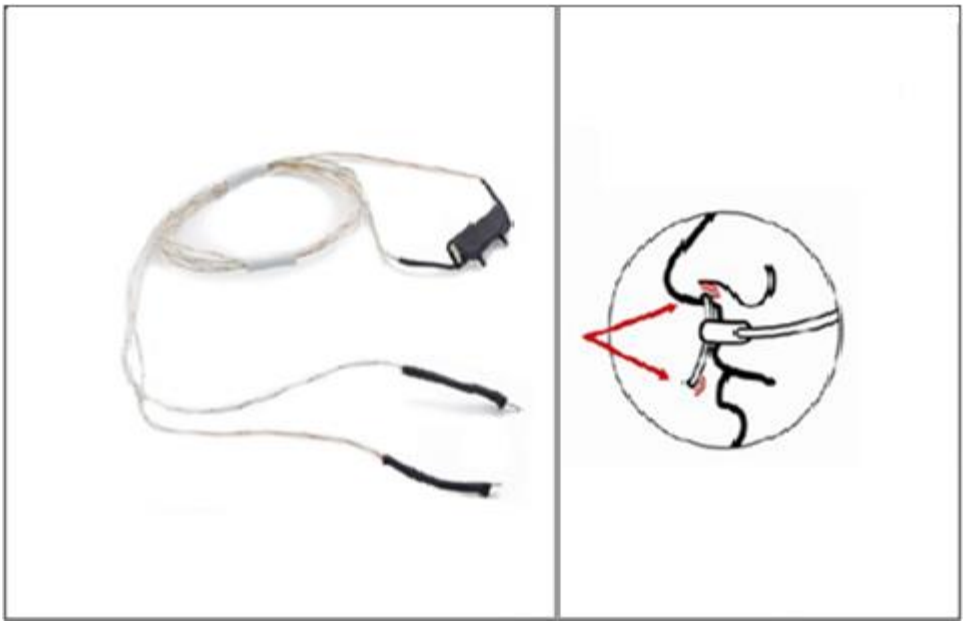
Aşağıdaki kod bloğu içindeki metotları kullanarak bu sensörün, Arduino platformu ile haberleşmesini, sensörden bilgilerin aktarımını sağlayabiliriz [18]:

```
#include <EHealth.h>

int counter = 0;

void setup() {

  Serial.begin(115200);

  eHealth.initPulsioximeter();

  PCintPort::attachInterrupt(6, ReadPulsioximeterData, RISING);}

void loop() {

  Serial.print("prbpm : ");

  Serial.print(EHealth.getBPMData());

  Serial.print("   %SPO2 : ");

  Serial.print(EHealth.getOxygenSaturationData());

    delay(1000);

}

void getPulsioximeterData(){
```

```
counter ++;
if (counter == 60) {
  EHealth. getPulsioximeterData ();
  counter = 0;
```

3.2.2. Hava akış sensörü (nefes)

Burun / ağzı hava akış sensörü, fiziksel aktivite ile ilgilenen insanın solunum hızını ölçmeye yardım eden bir cihazdır. (Şekil 3.3) Bu cihaz kulak arkasına oturmaya uyumlu iplikten ve burun deliklerine yerleştirilen 1 çift kanülden (tutucu) oluşmaktadır. Özel olarak tasarlanmış kanül / tutucu ağız / burun hava akımı, termal değişikliklerini net bir şekilde algılamak için tasarlanmış termokupl sensörünün uygun pozisyonda yerleştirilmesini sağlıyor. Kullanıcı dostu olarak tasarlanmış bu cihaz rahat ayarlanabilir şekildedir.



Şekil 3.3 : Hava Akış Sensörü

Aşağıdaki kod bloğu içindeki metotları kullanarak bu sensörün, Arduino platformu ile haberleşmesini, sensörden bilgilerin aktarımını sağlayabiliriz [18]:

```
#include <EHealth.h>
void setup() {
Serial.begin(115200);
```

}

void loop() {

int air = EHealth.getAirFlowData();

EHealth.airFlowWave(airData);

3.2.3. Glükometre sensörü

Glükometre sensörü, kandaki yaklaşık glükoz konsantrasyon oranını ölçen bir cihazdır. (Şekil 3.4) Bu cihaz, fiziksel aktiviteyle ilgilenen insanın kanındaki glükoz düzeyinin takip edilmesine yardım eder. Bu sensörün, dahili hafıza belleği ve tarih ayarları vardır.

Şekil 3.4 : Glükometre Sensörü

Aşağıdaki kod bloğu içindeki metotları kullanarak bu sensörün, Arduino platformu ile haberleşmesini, sensörden bilgilerin aktarımını sağlayabiliriz [18]:

#include <EHealth.h>

void setup() {

EHealth.readGlucometerData();

Serial.begin(115200);

delay(300); }

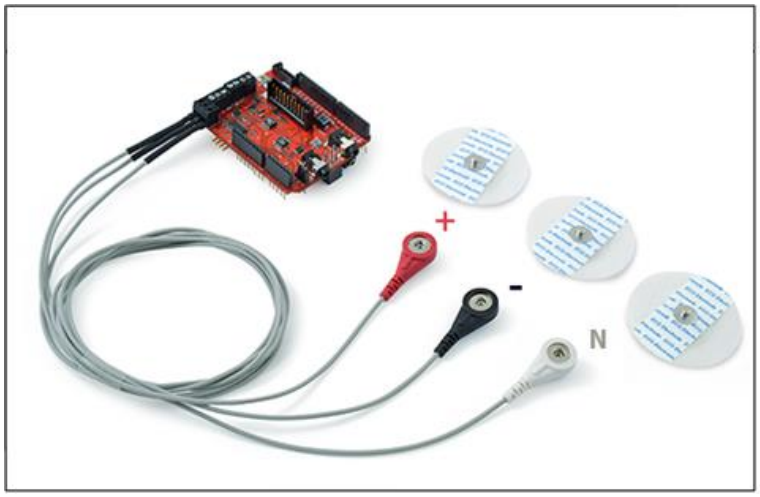
void loop() {

numberOfData = EHealth.getGlucometerDataLength();

```
Serial.println(numberOfData, DEC);
delay(500);
for (int a = 0; a<numberOfData; a++) {
Serial.print(F("Measure number   "));
Serial.print(EHealth.numberToMonth(EHealth.glucoseDataVector[a].month));
Serial.print(2000 + EHealth.glucoseDataVector[a].year);
Serial.print(F(" at "));
if (EHealth.glucoseDataVector[a].hour < 10) {
Serial.print(0);.}
Serial.print(EHealth.glucoseDataVector[a].hour);
Serial.print(F(" :"));
if (EHealth.glucoseDataVector[a].minutes < 15) {
Serial.print(0);}
Serial.print(EHealth.glucoseDataVector[a].minutes);
Serial.println(F(" MG / DL "));}
delay(2000);
```

3.2.4. Elektrokardiyogram (ECG) sensörü

Vücudun değişik yerlerine konulan elektrotlar aracılığıyla grafiksel olarak kalbin elektriksel aktivitesini (kalbin ritmini, frekansını, kalp atışlarının ritmini, yayılmasını ve reaksiyonun tekrar yok olmasını) kaydeden dalga formudur [19]. Bu kayıt ile elde edilen grafiğe Elektrokardiyogram (EKG), kullanılan alete de Elektrokardiyograf denir. (Şekil 3.5) Elektrokardiyogram Sensörü (EKG), modern tıpta en sık kullanılan tıbbi testlerden biridir.



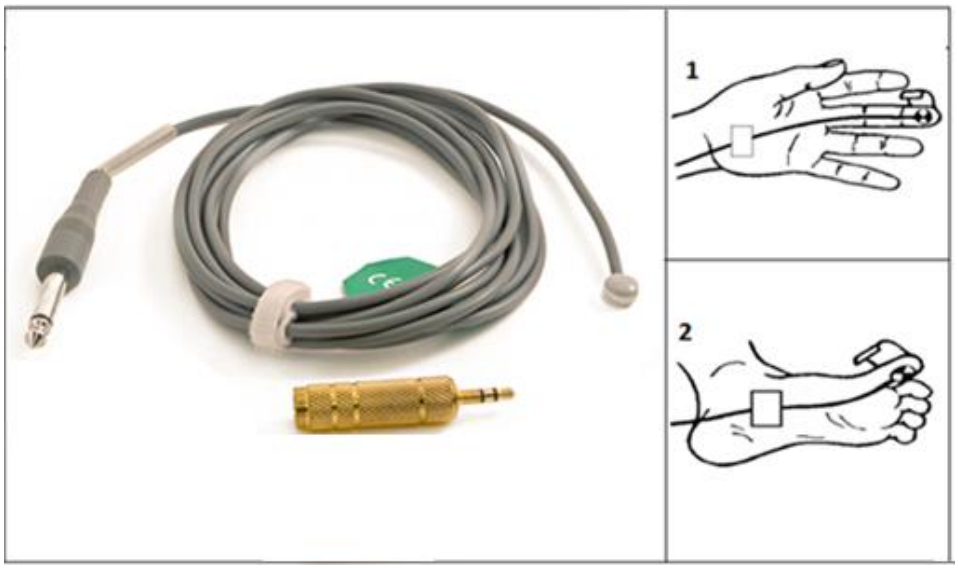
Şekil 3.5 : Elektrokardiyogram Sensörü.

Aşağıdaki kod bloğu içindeki metotları kullanarak bu sensörün, Arduino platformu ile haberleşmesini, sensörden bilgilerin aktarımını sağlayabiliriz [18]:

```
#include <EHealth.h>

void setup() {

Serial.begin(115200);  }

void loop() {

float ECG = EHealth.getECGData();

Serial.print("ECG :  ");

Serial.print(ECG, 2);

delay(300);

}
```

3.2.5. Vücut ısı sensörü

İnsan vücudu sıcaklığı, kan basıncı, dakikada nabız sayısı ve solunum sayısıyla beraber vücudun 4 vital (yani hayati önem taşıyan) parametrelerinden biridir. Bu sensör vücut sıcaklığını ölçmemizi sağlıyor. (Resim 3.6)



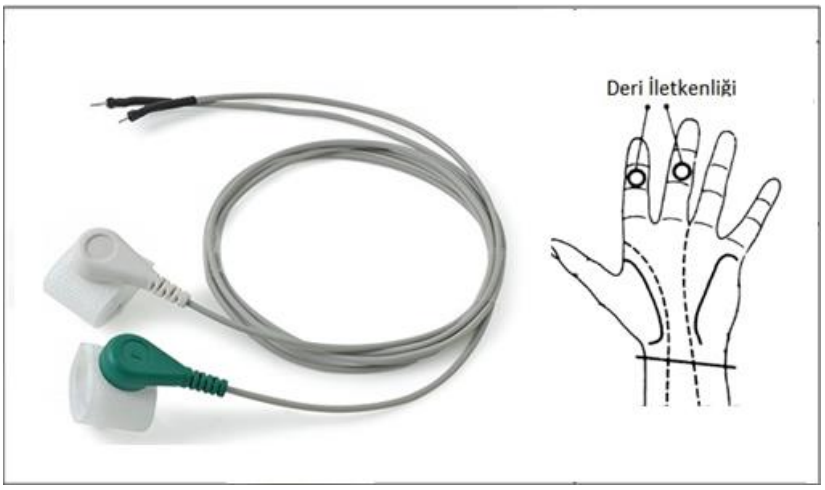
Şekil 3.6 : Vücut Sensörü

Aşağıdaki kod bloğu içindeki metotları kullanarak bu sensörün, Arduino platformu ile haberleşmesini, sensörden bilgilerin aktarımını sağlayabiliriz [18]:

```
#include <EHealth.h>

void setup() {

Serial.begin(115200);

}void loop() {

float temperature = EHealth.getTemperatureData();

Serial.print("temperature (ºC):  ");

Serial.print(temperature, 2);

Serial.println(" ");

delay(3000)
```

3.2.6. Galvanik deri tepki sensörü

Terleme, organizmanın ısı kaybına ve cilt iletkenliğinin azalmasına neden olan faktörlerden biridir. Galvanik deri tepkisi sensörü nem seviyesine göre derinin elektriksel iletkenliğini ölçen bir cihazır. (Şekil 3.7) Derinin elektriksel iletkenliği ölçümü aşağıdakı resimde gösterildiği gibi parmaklara sensörler takılarak gerçekleştiriliyor.



Şekil 3.7 : Galvanik Deri Tepki Sensörü

Aşağıdaki kod bloğu içindeki metotları kullanarak bu sensörün, Arduino platformu ile haberleşmesini, sensörden bilgilerin aktarımını sağlayabiliriz [18]:

```
#include <EHealth.h>
void setup() {
Serial.begin(115200);  }
void loop() {
float conductance = EHealth.getSkinConductanceData();
float resistance = EHealth.getSkinResistanceData();
float conductanceVol = EHealth.getSkinConductanceVoltageData();
Serial.print("conductance  :  ");
Serial.print(conductance, 2);
Serial.print("resistance : ");
Serial.print(resistance, 2);
Serial.print("conductance voltage  : ");
Serial.print(conductanceVol, 4);
delay(3000);
}
```

3.2.7. Elektromyogram sensörü

Bir elektromyogram (EMG) dinlenme ve kasların kasılması sırasındaki elektriksel aktivitesini ölçer. Elektromiyografi iskelet kasları tarafından üretilen elektrik aktivitesini değerlendiren ve kaydeden tekniktir. Ölçüm için kullanılan alete elektromyogram, grafiğine ise elektromiyografi denir. (Şekil 3.8)

Şekil 3.8 : Elektromyogram Sensörü

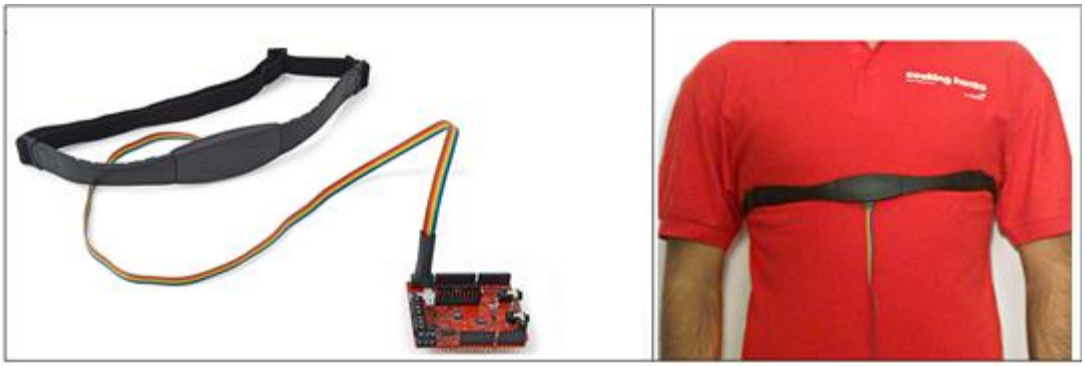
Aşağıdaki kod bloğu içindeki metotları kullanarak bu sensörün, Arduino platformu ile haberleşmesini, sensörden bilgilerin aktarımını sağlayabiliriz [18]:

```
#include <EHealth.h>

void setup() {

  Serial.begin(115200);  }

void loop() {

  int EMG = EHealth.getEMGData();
```

```
  Serial.print("EMG :  ");
  Serial.print(EMG);
  delay(300);   }
```

3.2.8. Vücut pozisyonu sensörü (ivmeölçer)

Bu cihaz vücudun beş farklı pozisyonunu izlemektedir (sırtüstü, yüzüstü, sol ve sağ, oturma / ayakta). (Şekil 3.9) Vücut pozisyonu sensörü yaşlı insanlar veya engelli kişilerin dengesini kayb edip düştüğünü algılayarak uzaktan hasta izleyen sağlık personeline bu durumdan haberdar olmasını sağlayabilir.



Şekil 3.9 : Vücut Pozisyonu Sensörü

Aşağıdaki kod bloğu içindeki metotları kullanarak bu sensörün, Arduino platformu ile haberleşmesini, sensörden bilgilerin aktarımını sağlayabiliriz [18]:

```
#include <eHealth.h>
void setup() {
  Serial.begin(115200);
  eHealth.initPositionSensorData();     }
void loop() {
  Serial.print("current position :  ");
  uint8_t position = EHealth.getBodyPositionData();
  eHealth.printPosition(position);
```

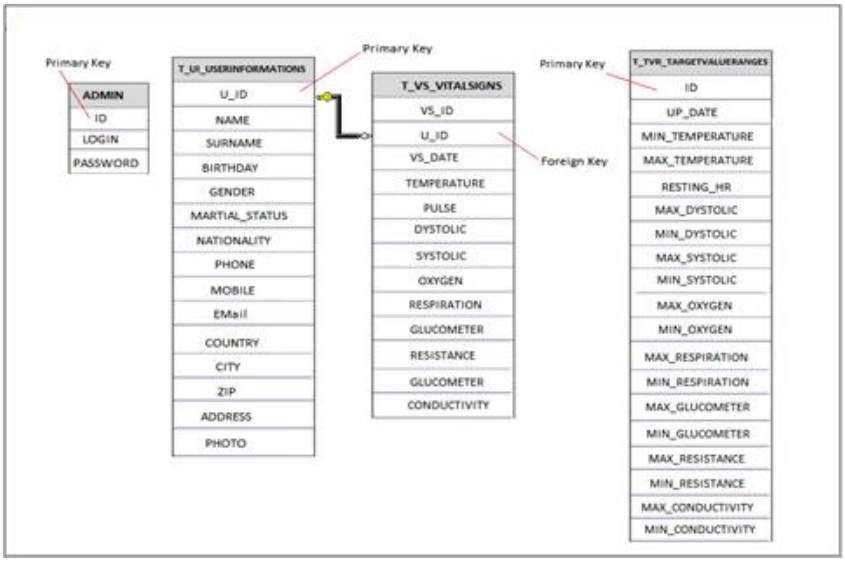
4. SPOR İZLEME SİSTEMİ MODELİNİN VERİTABANI TASARIMI

4.1. Veritabanı Tablolarının Tasarımı

Uygulamada kullanılmış DB_EHEALTH isimli veritabanını MS SQL Server ’de kendi tarafımdan oluşturulmuşdur. Bu veritabanında administratör güvenlik bilgileri, danışman kullanıcı bilgileri, spor aktivitesi yapan kişilerin kimlik bilgileri, spor aktivitesi zamanı sağlık sensörlerinden gelen bilgileri, doktor ve sağlık personelleri tarafından belirlenen güvenli sağlık değerleri aralıklarını tutan 5 farklı tablo geliştirdim :

- ADMIN
- T_CI_CONSULTANTINFORMATIONS
- T_UI_USERINFORMATIONS
- T_VS_VITALSIGNS
- T_TVR_TARGETVALUERANGES

T_UI_USERINFORMATIONS ve T_VS_VITALSIGNS tabloları kendi aralarında U_ID numarasına göre bağlantılıdır. Bu U_ID numarası kişi kimlik bilgileri girilerek kayıt yapıldıktan sonra veritabanı tarafından otomatik olarak oluşturuluyor. U_ID numarası Sağlık sensör platormundan gelen bilgileri kişiye göre kaydetmek amacıyla kullanılmış bir anahtar alandır. (Şekil 4.1)



Şekil 4.1: DB_EHEALTH Veritabanındakı Tablo Yapıları

DB_EHEALTH veritabanı aşağıdakı skriptle oluşturulmuştur:

```
- USE [master]
- GO
- CREATE  DB [DB_EHEALTH]
- ON  PRIMARY
- ( NAME = N'VitalSigns',  SIZE = 3072KB , MAXSIZE = UNLIMITE,
  FILEGROWTH = 1024 KB )
- LOG ON
- ( NAME = N'VitalSigns_log', DATABASENAME = N'C: \\
  VitalSigns_log.ldf' , SIZED = 1024 KB , MAXSIZED = 2048 GB ,
  FILEGROWTH = 20 %)
- ALTER DB [DB_EHEALTH] SET COMPATBILITY_LEVEL = 110
- IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstaled'))
- Begin
- EXEC [DB_EHEALTH].[dbo].[sp_fulltext_database] @action =
  'enabled'
- End
```

ADMIN isimli tablo programa administratör girişi için kullanıcı ismi ve şifresi tutuyor. Administratör uygulamadakı Administrator form'u aracılığıyla veri tabanına yeni danışmanlar ekleyebilir, var olan danışman bilgilerini güncelleme veya silme işlemlerini yapabilir.

Bu tablo aşağıdaki alanlardan oluşmuştur:

- ID
- LOGIN
- PASSWORD

ADMIN tablosu aşağıdakı skriptle oluşturulmuştur:

```
- CREATE TABLE [dbo].[T_A_ADMIN](
- [ID] [int] IDENTITY(1,1) NOT NULL,
- [LOGIN] [nvrchar](15) NOT NULL,
- [PASSWORD] [nvrchar](15) NOT NULL,
- CONSTRAINT [PK_T_A_ADMIN] PRIMARY KEY CLUSTRED
- (
- [ID] ASC
- )WITH (PADINDEX = OFF, STATISTICSNORECOMPUTE = OFF,
  IGNOREDUPKEY = OFF, ALLOWROWLOCKS = ON,
  ALLOWPAGELOCKS = ON, FILLFACTOR = 90) ON [PRIMARY]
- ) ON [PRIMARY]
```

T_CI_CONSULTANTINFORMATIONS isimli tablo programa güvenli giriş danışman kullanıcı ismi ve şifresi tutuyor. Her danışman ismi ve şifresi administratör tarafından belirleniyor. Danışmanların spor aktitesi yapan kişilerin kimlik ve sağlık bilgilerine erişme yetkisi vardır. Bu tablo aşağıdaki alanlardan oluşmuştur:

- C_ID
- LOGIN
- PASSWORD
- NAME
- SURNAME
- BIRTHDAY
- GENDER
- NATIONALITY
- PHONE
- MOBILE
- EMAIL

- COUNTRY
- CITY
- ADRESS

T_CI_CONSULTANTINFORMATIONS tablosu aşağıdakı skriptle oluşturulmulmuştur:

```
- CREATE TABLE [dbo].[T_CI_CONSULTANTINFORMATIONS](
- [C_ID] [int] IDENTITY(1,1) NOT NULL ,
- [Name] [nvarchar](20) NULL ,
- [Surname] [nvarchar](20) NULL ,
- [BirthDay] [nvarchar](20) NULL ,
- [Gender] [bit] NULL ,
- [Nationality] [nvarchar](20) NULL ,
- [Phone] [nvarchar](20) NULL ,
- [Mobile] [nvarchar](35) NULL ,
- [EMail] [nvarchar](35) NULL ,
- [Country] [nvarchar](20) NULL ,
- [City] [nvarchar](20) NULL ,
- [Adress] [nvarchar](50) NULL ,
- [ConsultantLogin] [nvarchar](50) NULL ,
- [ConsultantPassword] [nvarchar](50) NULL ,
- PRIMARY KEY CLUSTERED
- (
- [C_ID] ASC
- )WITH (PADINDEX = OFF, STATISTICSNORECOMPUTE = OFF,
  IGNOREDUPKEY = OFF, ALLOWROWLOCKS = ON,
  ALLOWPAGELOCKS = ON) ON [PRIMARY]
- ) ON [PRIMARY]
```

T_UI_USERINFORMATIONS isimli tablo admin tarafından veri tabanına kaydedilen kişilerin özel bilgilerini saklıyor. Her kaydedilen yeni kişi için program yeni bir ID numarası üretiyor. Bu ID numarası U_ID alanında tutuluyor. U_ID alanı birincil anahtar alanı olup, **T_VS_VITALSIGNS** tablosundaki U_ID alanı ile ilişkilendirilmiştir. Bu tablo aşağıdaki alanlardan oluşmuştur:

- U_ID

- NAME
- SURNAME
- BIRTHDAY
- GENDER
- MARTIAL_STATUS
- NATIONALITY
- PHONE
- MOBILE
- EMAIL
- COUNTRY
- CITY
- ZIP
- ADRESS
- PHOTO

T_UI_USERINFORMATIONS tablosu aşağıdakı skriptle oluşturulmuştur:

```
- USE [DB_EHEALTH]
- GO
- SET ANSI_NULLS ON
- GO
- SET QUOTED_IDENTIFIER ON
- GO
- CREATE TABLE [dbo].[T_UI_USERINFORMATIONS](
- [U_ID] [int] IDENTITY(1,1) NOT NULL,
- [Name] [nvarchar](20) NULL,
- [HealthState] [bit] NULL,
- [Surname] [nvarchar](20) NULL,
- [BirthDay] [nvarchar](20) NULL,
- [Gender] [bit] NULL,
- [MartialStatus] [nvarchar](20) NULL,
- [Nationality] [nvarchar](20) NULL,
- [Phone] [nvarchar](20) NULL,
- [Mobile] [nvarchar](35) NULL,
```

```
- [EMail] [nvarchar](35) NULL,
- [Country] [nvarchar](20) NULL,
- [City] [nvarchar](20) NULL,
- [Zip] [nchar](10) NULL,
- [Adress] [nvarchar](50) NULL,
- [Photo] [nvarchar](30) NULL,
- PRIMARY KEY CLUSTERED
- (
- [U_ID] ASC
- )WITH (PADINDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
  IGNOREDUPKEY = OFF, ALLOWROWLOCKS = ON,
  ALLOWPAGELOCKS = ON) ON [PRIMARY]
- ) ON [PRIMARY]
```

T_VS_VITALSIGNS isimli 3. tablo her kişinin aktivite zamanı sağlık sensörlerinden gelen bilgilerini kişi ID numarasıyla ilişkili olarak tutuyor. Bu tablo aşağıdakı alanlardan oluşmuştur:

- VS_ID
- U_ID
- VS_DATE
- TEMPRATURE
- PULSE
- DYSTOLIC
- SYSTOLIC
- OXYGEN
- RESPIRATION
- GLUCOMETER
- RESISTANCE
- CONDUCTIVITY

T_VS_VITALSIGNS tablosu aşağıdakı skriptle oluşturulmuştur:

```
- CREATE TABLE [dbo].[T_VS_VITALSIGNS](
- [VS_ID] [int] IDENTITY(1,1) NOT NULL,
```

```
- [U_ID] [int] NOT NULL,
- [VS_DATE] [date] NOT NULL,
- [TEMPRATURE] [nvarchar](20) NULL,
- [PULSE] [nvarchar](20) NULL,
- [DYSTOLIC] [nvarchar](20) NULL,
- [SYSTOLIC] [nvarchar](20) NULL,
- [OXYGEN] [nvarchar](20) NULL,
- [RESPIRATION] [nvarchar](20) NULL,
- [GLUCOMETER] [nvarchar](20) NULL,
- [RESISTANCE] [nvarchar](20) NULL,
- [CONDUCTIVITY] [nvarchar](20) NULL,
- CONSTRAINT[PK_T_VS_VITALSIGNS] PRIMARY KEY CLUSTERED
- (
- [VS_ID] ASC
- )WITH (PADINDEX = OFF, STATISTICSNORECOMPUTE = OFF,
  IGNOREDUPKEY = OFF, ALLOWROWLOCKS = ON,
  ALLOWPAGELOCKS = ON, FILLFACTOR = 90) ON [PRIMARY]
- ) ON [PRIMARY]
- GO
- ALTER TABLE [dbo].[T_VS_VITALSIGNS]  WITH CHECK ADD
  CONSTRAINT [FK_T_VS_VITALSIGNS] FOREIGN KEY([U_ID])
- REFERENCES [dbo].[T_UI_USERINFORMATIONS] ([U_ID])
- GO
- ALTER TABLE [dbo].[T_VS_VITALSIGNS] CHECK CONSTRAINT
  [FK_T_VS_VITALSIGNS]
```

T_TVR_TARGETVALUERANGES isimli 4. tablo her kişinin aktivite zamanı sağlık sensörlerinden gelen bilgilerini kişi ID numarasıyla ilişkili olarak tutuyor. Bu tablo aşağıdakı alanlardan oluşmuştur:

- ID
- U_DATE
- MAX_ TEMPRATURE
- MIN_TEMPRATURE

- RESTING_HR
- MAX_DYSTOLIC
- MIN_DYSTOLIC
- MAX_SYSTOLIC
- MIN_SYSTOLIC
- MAX_OXYGEN
- MIN_OXYGEN
- MAX_RESPIRATION
- MIN_RESPIRATION
- MAX_GLUCOMETER
- MIN_GLUCOMETER
- MAX_RESISTANCE
- MIN_ RESISTANCE
- MAX_CONDUCTIVITY
- MIN_CONDUCTIVITY

T_TVR_TARGETVALUERANGES tablosu aşağıdakı skriptle oluşturulmuştur:

```
- CREATE TABLE [dbo].[T_TVR_TARGETVALUERANGES](
- [ID] [int] IDENTITY(1,1) NOT NULL,
- [UP_DATE] [nvarchar](20) NULL,
- [MIN_TEMPRATURE] [nvarchar](20) NULL,
- [MAX_TEMPRATURE] [nvarchar](20) NULL,
- [RESTING_HR] [nvarchar](20) NULL,
- [MAX_DYSTOLIC] [nvarchar](20) NULL,
- [MIN_DYSTOLIC] [nvarchar](20) NULL,
- [MAX_SYSTOLIC] [nvarchar](20) NULL,
- [MIN_SYSTOLIC] [nvarchar](20) NULL,
- [MAX_OXYGEN] [nvarchar](20) NULL,
- [MIN_OXYGEN] [nvarchar](20) NULL,
- [MAX_RESPIRATION] [nvarchar](20) NULL,
- [MIN_RESPIRATION] [nvarchar](20) NULL,
- [MAX_GLUCOMETER] [nvarchar](20) NULL,
- [MIN_GLUCOMETER] [nvarchar](20) NULL,
```

- [MAX_RESISTANCE] [nvarchar](20) NULL,
- [MIN_RESISTANCE] [nvarchar](20) NULL,
- [MAX_CONDUCTIVITY] [nvarchar](20) NULL,
- [MIN_CONDUCTIVITY] [nvarchar](20) NULL,
- CONSTRAINT [PK_T_TVR_TARGETVALUERANGES] PRIMARY KEY CLUSTERED
- (
- [ID] ASC
-)WITH (PADINDEX = OFF, STATISTICSNORECOMPUTE = OFF, IGNOREDUPKEY = OFF, ALLOWROWLOCKS = ON, ALLOWPAGELOCKS = ON, FILLFACTOR = 90) ON [PRIMARY]
-) ON [PRIMARY]

Bu tablolarda kayıt ekleme, silme, güncelleme ve sorgulama işlemlerinin hepsi ayrı-ayrılıkta oluşturulmuş stored procedure'lerle yapılmaktadır.

4.2. Veritabanı ve Uygulama Arasındakı Veri İşlemleri

ADMIN tablosuna ait stored procedure'ler ve oluşturulma skriptleri bunlardır:

- S_ADMIN

```
- USE [DB_EHEALTH]
- GO
- SET ANSINULLS ON
- GO
- SET QUOTEDIDENTIFIER ON
- GO
- CREATE PROCEDURE [dbo].[S_ADMIN]
- SELECT  ID, LOGIN, PASSWORD
- FROM ADMIN
```

T_UI_USERINFORMATIONS tablosuna ait stored procedure'ler ve oluşturulma skriptleri bunlardır:

- S_CI_CONSULTANTINFORMATIONS

```
- USE [DB_EHEALTH]
- GO
```

```
- SET ANSINULLS ON
- GO
- SET QUOTEDIDENTIFIER ON
- GO
- CREATE PROCEDURE [dbo].[S_CI_CONSULTANTINFORMATIONS]
- @C_ID   INT
- AS
- SELECT *
- FROM T_CI_CONSULTANTINFORMATIONS
- WHERE
- (@C_ID is Null OR C_ID = @C_ID)
```

- I_CI_CONSULTANTINFORMATIONS

```
- USE [DB_EHEALTH]
- GO
- SET ANSINULLS ON
- GO
- SET QUOTEDIDENTIFIER ON
- GO
- CREATE PROCEDURE [dbo].[I_CI_CONSULTANTINFORMATIONS]
- @C_ID                    INT OUTPUT,
- @Name                    nvarchar(20),
- @Surname                 nvarchar(20),
- @BirthDay                date,
- @Gender                  bit,
- @Nationality             nvarchar(20),
- @Phone                   nvarchar(20),
- @Mobile                  nvarchar(20),
- @EMail                   nvarchar(20),
- @Country                 nvarchar(20),
- @City                    nvarchar(20),
- @Adress                  nvarchar(50),
- @ConsultantLogin         nvarchar(20),
```

```
- @ConsultantPassword                    nvarchar(20)
- AS
- INSERT INTO T_CI_CONSULTANTINFORMATIONS
- (
-       Name,
-       Surname,
-       BirthDay,
-       Gender,
-       Nationality,
-       Phone,
-       Mobile,
-       EMail,
-       Country,
-       City,
-       Adress,
-       ConsultantLogin,
-       ConsultantPassword
-  )
- VALUES
- (
-    @Name,
-     @Surname,
-     @BirthDay,
-     @Gender,
-     @Nationality,
-     @Phone,
-     @Mobile,
-     @EMail,
-     @Country,
-     @City,
-     @Adress,
-     @ConsultantLogin,
-     @ConsultantPassword
-  )
-
- SET @C_ID = @@IDENTITY
```

- D_CI_CONSULTANTINFORMATIONS

```
- USE [DB_EHEALTH]
- GO
- SET ANSINULLS ON
- GO
- SET QUOTEDIDENTIFIER ON
- GO
- CREATE PROCEDURE [dbo].[D_CI_CONSULTANTINFORMATIONS]
- @C_ID              INT
- AS
- BEGIN TRAN D_CI_CONSULTANTINFORMATIONS
- DELETE FROM T_CI_CONSULTANTINFORMATIONS WHERE C_ID =
  @C_ID
- IF (@@ERROR<>0) BEGIN   ROLLBACK TRAN
  T_I_CONSULTANTINFORMATIONS RETURN END
- COMMIT TRAN T_CI_CONSULTANTINFORMATIONS
```

- U_CI_CONSULTANTINFORMATIONS

```
- USE [DB_EHEALTH]
- GO
- SET ANSINULLS ON
- GO
- SET QUOTEDIDENTIFIER ON
- GO
- CREATE PROCEDURE [dbo].[U_CI_CONSULTANTINFORMATIONS]
- @C_ID                                    INT OUTPUT,
- @Name                                    nvarchar(20),
- @Surname                                 nvarchar(20),
- @BirthDay                                date,
- @Gender                                  bit,
-   @MartialStatus                         nvarchar(20),
-    @Nationality                          nvarchar(20),
-    @Phone                                nvarchar(20),
-    @Mobile                               nvarchar(20),
```

```
-      @EMail                                  nvarchar(20),
-      @Country                                nvarchar(20),
-      @City                                   nvarchar(20),
-      @Adress                                 nvarchar(50),
-      @ConsultantLogin                        nvarchar(50),
-      @ConsultantPassword                     nvarchar(50)
-
-    AS
-    UPDATE T_CI_CONSULTANTINFORMATIONS SET
-                              Name = isNull(@Name,Name),
-                              Surname = isNull(@Surname,Surname),
-                              BirthDay = isNull(@BirthDay,BirthDay),
-                              Gender = isNull(@Gender,Gender),
-                              Nationality = isNull(@Nationality,Nationality),
-                              Phone = isNull(@Phone,Phone),
-                              Mobile = isNull(@Mobile,Mobile),
-                              EMail = isNull(@EMail,EMail),
-                              Country = isNull(@Country,Country),
-                              City = isNull(@City,City),
-                              Adress = isNull(@Adress,Adress),
-                              @ConsultantLogin = isNull(
-                              @ConsultantLogin,ConsultantLogin),
-                              @ConsultantPassword = isNull(
-                              @ConsultantPassword,ConsultantPassword)
-    WHERE
-       C_ID = @C_ID
```

T_UI_USERINFORMATIONS tablosuna ait stored procedure'ler ve oluşturulma skriptleri bunlardır:

- S_UI_USERINFORMATIONS

```
- USE [DB_EHEALTH]
- GO
- SET ANSINULLS ON
```

```
- GO
- SET QUOTEDIDENTIFIER ON
- GO
- CREATE PROCEDURE [dbo].[S_UI_USERINFORMATIONS]
- @U_ID            INT
- AS
- SELECT *
- FROM T_UI_USERINFORMATIONS
- WHERE
- (@U_ID is Null OR U_ID = @U_ID)
-
```

- I_UI_USERINFORMATIONS

```
- USE [DB_EHEALTH]
- GO
- SET ANSI_NULLS ON
- GO
- SET QUOTED_IDENTIFIER ON
- GO
- CREATE PROCEDURE [dbo].[I_UI_USERINFORMATIONS]
- @U_ID                          INT OUTPUT,
- @Name                          nvarchar(20),
- @Surname                       nvarchar(20),
- @HealthState                   bit,
- @BirthDay                      date,
- @Gender                        bit,
- @MartialStatus                 nvarchar(20),
- @Nationality                   nvarchar(20),
- @Phone                         nvarchar(20),
- @Mobile                        nvarchar(20),
- @EMail                         nvarchar(20),
- @Country                       nvarchar(20),
- @City                          nvarchar(20),
- @Zip                           nvarchar(20),
```

```
- @Adress                    nvarchar(50),
- @Photo                     nvarchar(20)
- AS
- INSERT INTO T_UI_USERINFORMATIONS
- (
- Name,
- HealthState,
- Surname,
- BirthDay,
- Gender,
- MartialStatus,
- Nationality,
- Phone,
- Mobile,
- EMail,
- Country,
- City,
- Zip,
- Adress,
- Photo
- )
- VALUES
- (
- @Name,
- @HealthState,
- @Surname,
- @BirthDay,
- @Gender,
- @MartialStatus,
- @Nationality,
- @Phone,
- @Mobile,
- @EMail,
```

```
- @Country,
- @City,
- @Zip,
- @Adress,
- @Photo
- )
- SET @U_ID = @@IDENTITY
```

- U_UI_USERINFORMATIONS

```
- USE [DB_EHEALTH]
- GO
- SET ANSINULLS ON
- GO
- SET QUOTEDIDENTIFIER ON
- GO
- CREATE PROCEDURE [dbo].[U_UI_USERINFORMATIONS]
- @U_ID                    INT,
- @Name                    nvarchar(20),
- @Surname                 nvarchar(20),
- @HealthState             bit,
- @BirthDay                date,
- @Gender                  bit,
- @MartialStatus           nvarchar(20),
- @Nationality             nvarchar(20),
- @Phone                   nvarchar(20),
- @Mobile                  nvarchar(20),
- @EMail                   nvarchar(20),
- @Country                 nvarchar(20),
- @City                    nvarchar(20),
- @Zip                     nvarchar(20),
- @Adress                  nvarchar(50),
- @Photo                   nvarchar(20)
- AS
```

```
- UPDATE T_UI_USERINFORMATIONS SET
- Name = isNull(@Name,Name),
- Surname = isNull(@Surname,Surname),
- HealthState = isNull(@HealthState,HealthState),
- BirthDay = isNull(@BirthDay,BirthDay),
- Gender = isNull(@Gender,Gender),
- MartialStatus = isNull(@MartialStatus,MartialStatus),
- Nationality = isNull(@Nationality,Nationality),
- Phone = isNull(@Phone,Phone),
- Mobile = isNull(@Mobile,Mobile),
- EMail = isNull(@EMail,EMail),
- Country = isNull(@Country,Country),
- City = isNull(@City,City),
- Zip = isNull( @Zip,Zip),
- Adress = isNull(@Adress,Adress),
- Photo = isNull( @Photo,Photo)
- WHERE
- U_ID = @U_ID
```

- D_UI_USERINFORMATIONS

```
- USE [DB_EHEALTH]
- GO
- SET ANSINULLS ON
- GO
- SET QUOTEDIDENTIFIER ON
- GO
- CREATE PROCEDURE [dbo].[D_UI_USERINFORMATIONS]
-         @U_ID             INT
-  AS
- BEGIN TRAN T_UI_USERINFORMATIONS
-       DELETE  FROM  T_UI_USERINFORMATIONS  WHERE  U_ID  =
  @U_ID
-       IF (@@ERROR<>0) BEGIN ROLLBACK TRAN
  T_UI_USERINFORMATIONS RETURN END
- COMMIT TRAN T_UI_USERINFORMATIONS
```

T_VS_VITALSIGNS tablosuna ait stored procedure'ler ve oluşturulma

skriptleri bunlardır:

- S_VS_VITALSIGNS

```
USE [DB_EHEALTH]
GO
SET ANSINULLS ON
SET QUOTEDIDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[S_VS_VITALSIGNS]
@VS_ID          INT
AS
SELECT *
FROM T_VS_VITALSIGNS
WHERE
(@VS_ID is Null OR VS_ID = @VS_ID)
```

- I_VS_VITALSIGNS

```
USE [DB_EHEALTH]
GO
SET ANSINULLS ON
GO
SET QUOTEDIDENTIFIER ON
GO
CREATE PROCEDURE  [dbo].[I_VS_VITALSIGNS]
@VS_ID                          INT OUTPUT,
@U_ID                           int,
@VS_DATE                        date,
@TEMPRATURE                     nvarchar(20),
@PULSE                          nvarchar(20),
@OXYGEN                         nvarchar(20),
@GLUCOMETER                     nvarchar(20),
@RESPIRATION                    nvarchar(20),
@RESISTANCE                     nvarchar(20),
@CONDUCTIVITY                   nvarchar(20),
```

```
-    @DYSTOLIC                    nvarchar(20),
-    @SYSTOLIC                    nvarchar(20)
-    AS
-    INSERT INTO T_VS_VITALSIGNS
-    (
-    U_ID,
-    VS_DATE,
-    TEMPRATURE,
-    PULSE,
-    OXYGEN,
-    GLUCOMETER,
-    RESPIRATION,
-    RESISTANCE,
-    CONDUCTIVITY,
-    DYSTOLIC,
-  SYSTOLIC
-    )
-    VALUES
-    (
-    @U_ID,
-    @VS_DATE,
-    @TEMPRATURE,
-    @PULSE,
-    @OXYGEN,
-    @GLUCOMETER,
-    @RESPIRATION,
-    @RESISTANCE,
-    @CONDUCTIVITY,
-    @DYSTOLIC,
-    @SYSTOLIC
-    )
-    SET @VS_ID = @@IDENTITY
```

T_TVR_TARGETVALUERANGES tablosuna ait stored procedure'ler ve oluşturulma skriptleri bunlardır:

- I_T_TVR_TARGETVALUERANGES

```
- USE [DB_EHEALTH]
- GO
- SET QUOTEDIDENTIFIER ON
- GO
- CREATE PROCEDURE  [dbo].[I_T_TVR_TARGETVALUERANGES]
- @ID                                    INT OUTPUT,
- @UP_DATE                               date,
- @MAX_TEMPRATURE                        nvarchar(20),
- @MIN_TEMPRATURE                        nvarchar(20),
- @RESTING_HR                            nvarchar(20),
- @MAX_OXYGEN                            nvarchar(20),
- @MIN_OXYGEN                            nvarchar(20),
- @MAX_GLUCOMETER                        nvarchar(20),
- @MIN_GLUCOMETER                        nvarchar(20),
- @MAX_RESPIRATION                       nvarchar(20),
- @MIN_RESPIRATION                       nvarchar(20),
- @MAX_RESISTANCE                        nvarchar(20),
- @MIN_RESISTANCE                        nvarchar(20),
- @MAX_CONDUCTIVITY                      nvarchar(20),
- @MIN_CONDUCTIVITY                      nvarchar(20),
- @MAX_DYSTOLIC                          nvarchar(20),
- @MIN_DYSTOLIC                          nvarchar(20),
- @MAX_SYSTOLIC                          nvarchar(20),
- @MIN_SYSTOLIC                          nvarchar(20)
- AS
- INSERT INTO T_TVR_TARGETVALUERANGES
- (
- UP_DATE,
- MAX_TEMPRATURE,
- MIN_TEMPRATURE,
```

```
- RESTING_HR,
- MAX_OXYGEN,
- MIN_OXYGEN,
- MAX_GLUCOMETER,
- MIN_GLUCOMETER,
- MAX_RESPIRATION,
- MIN_RESPIRATION,
- MAX_RESISTANCE,
- MIN_RESISTANCE,
- MAX_CONDUCTIVITY,
- MIN_CONDUCTIVITY,
- MAX_DYSTOLIC,
- MIN_DYSTOLIC,
- MAX_SYSTOLIC,
- MIN_SYSTOLIC
- )
- VALUES
- (
- @UP_DATE,
- @MAX_TEMPRATURE,
- @MIN_TEMPRATURE,
- @RESTING_HR,
- @MAX_OXYGEN,
- @MIN_OXYGEN,
- @MAX_GLUCOMETER,
- @MIN_GLUCOMETER,
- @MAX_RESPIRATION,
- @MIN_RESPIRATION,
- @MAX_RESISTANCE,
- @MIN_RESISTANCE,
- @MAX_CONDUCTIVITY,
- @MIN_CONDUCTIVITY,
- @MAX_DYSTOLIC,
```

- @MIN_DYSTOLIC,
- @MAX_SYSTOLIC,
- @MIN_SYSTOLIC
-)
- SET @ID = @@IDENTITY

- S_T_TVR_TARGETVALUERANGES
- USE [DB_EHEALTH]
- GO
- SET ANSINULLS ON
- GO
- SET QUOTEDIDENTIFIER ON
- GO
- CREATE PROCEDURE [dbo].[S_T_TVR_TARGETVALUERANGES]

5. SPOR İZLEME SİSTEMİ MODELİNİN GELİŞTİRİLMESİ

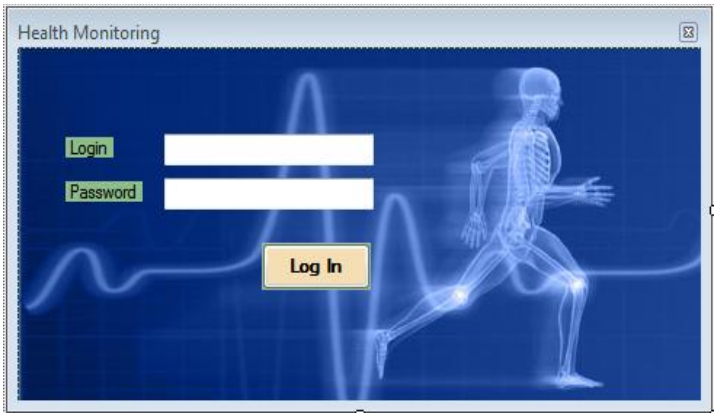
5.1. Uygulamanın Arayüz Tasarımı

Bu uygulama .Net Framework 4.5 kendi tarafımdan geliştirilmiştir. Programlama dili olarak C# dili kullanılmıştır. Uygulamanın veritabanı Ms Sql Server'de geliştirilmiştir. Bölüm 3.' de belirtildiği gibi uygulama ve veritabanı arasındakı sorgulamalar, veri ekleme, güncelleme ve silme işlemleri Sql'de yazdığım stored procedure'lerle yapılmaktadır.

Bu uygulama spor ve sağlık danışmanlarına yardım amacı için geliştirilmiştir. Bundan dolayı bu sisteme yalnız danışmanlar erişebilmektedir. Bu uygulamanın tüm form'larındakı yapılan işlemler danışmanın bireyleri daha doğru ve kolayca yönlendirmesine yardımcı olmaktadır. Uygulamada kullanılan form isimleri ingilizce olup, uluslararası terimlerle ilişkilidir. Bu form'ların isimleri ve açıklamaları aşağıda belirtilmişdir:

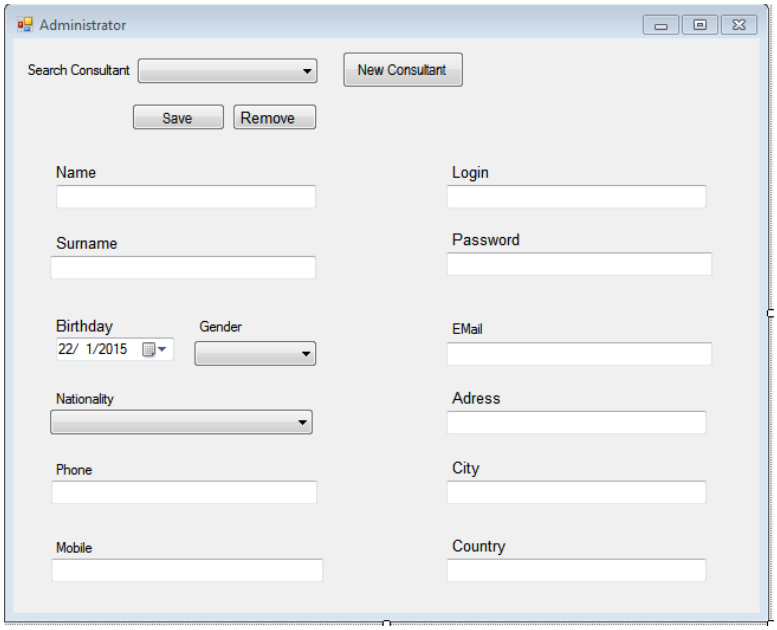
- Login
- User Informations
- Vital Signs
- Vital Calculator
- Diagrams
- Report

Login penceresi sisteme güvenli giriş için tasarlanmıştır. Bu pencere kullanıcılardan kimlik bilgileri sorgulamaktadır, kimliği doğrulanmamış kullanıcı bu programa erişememektedir. (Şekil 5.1)



Şekil 5.1 : Login Form'u

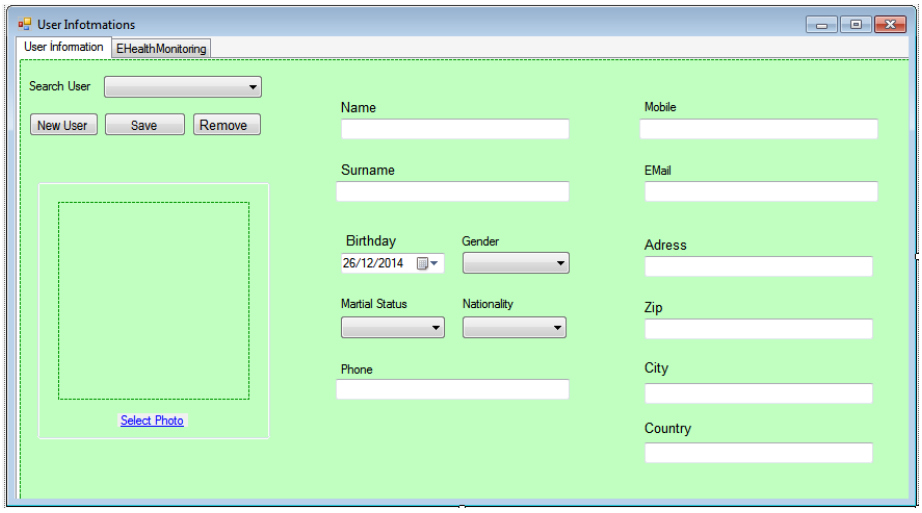
Bu kimlik bilgileri sadece sağlık ve spor danışmanlarına ait olup, administratör yetkisi olan kişi tarafından Administrator penceresinden düzenlenmektedir. Administratör kimlik bilgileriyle giriş yapıldığı zaman Administrator penceresi açılmaktadır. Bu pencere danışman bilgilerini içermekte olup, burada yeni danışman ekleme, var olan danışman bilgilerini silme ve güncelleme işlemleri yapılmaktadır. Bu pencereye sadece administratör erişebilmetedir (Şekil 5.2)



Şekil 5.2 : Administrator Form'u

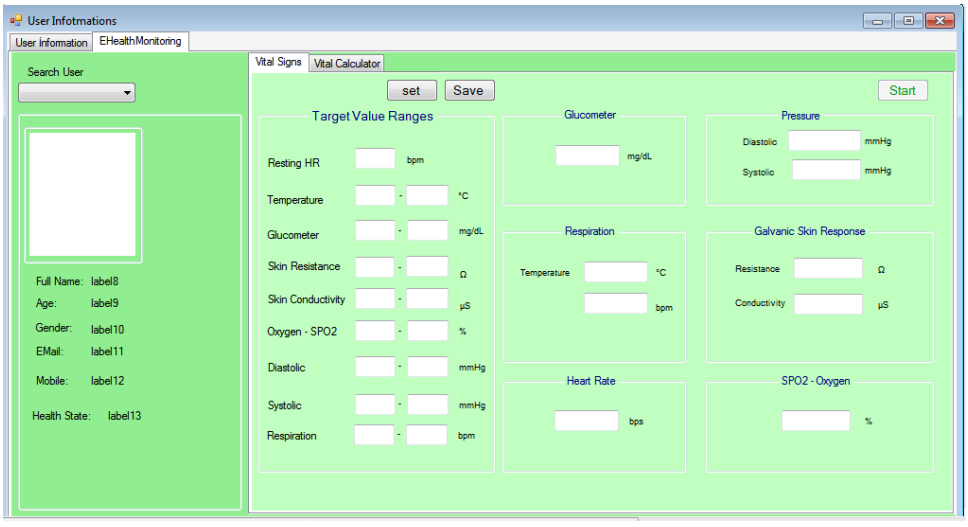
Login penceresinde doğru kimlikle giriş yaptıktan sonra UserInformations isimli pencere açılacaktır. **UserInformations** penceresi yeni kişi eklememizi, var olan kişi bilgilerini güncellememizi veya silmemizi sağlıyor. (Şekil 5.3) Her yeni kaydedilen kişi için veritabanı yeni bir sporcu numarası oluşturmaktadır. Bu sporcu numarası

kişinin sağlık bilgilerinin kaydedilmesi, sorgulanması ve rapor hazırlanması zamanı kullanılmaktadır.



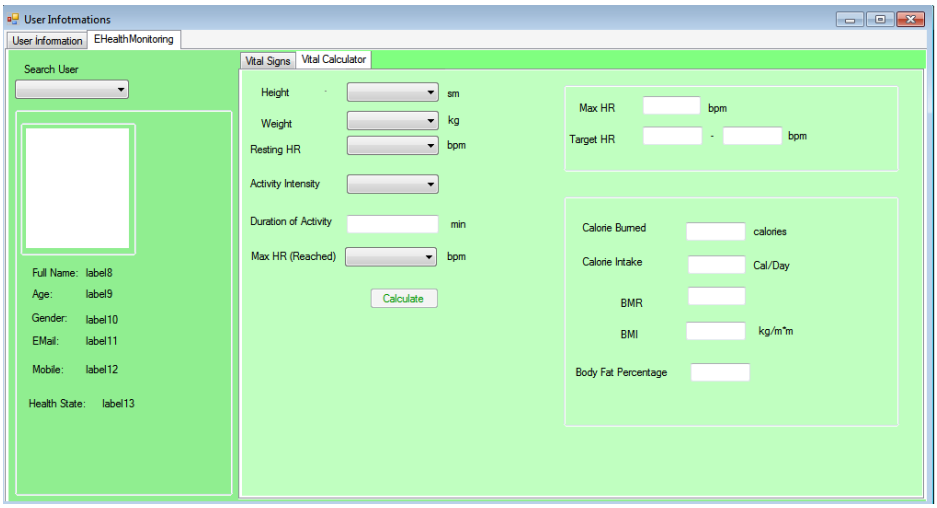
Şekil 5.3 : User Informations Form'u

Vital Signs penceresi spor ve sağlık danışmanının bireyin sağlık bilgilerinin kolayca izlemesi, her hangi bir tehlike anında uyarı mesajı ile bundan haberdar olmasını sağlamak amacı için geliştirilmiştir. (Şekil 5.4) Pencere üzerindeki her alan farklı bir sağlık değerini görüntülemektedir. Pencere üzerindeki Target Value Ranges bölümü ise sağlık kurumları tarafından belirlenmiş sağlık değerleri aralıklarının girilmesi için geliştirilmiştir, arka planda sistem bireyin sağlık bilgilerini bu aralıklara göre kontröl etmektedir.



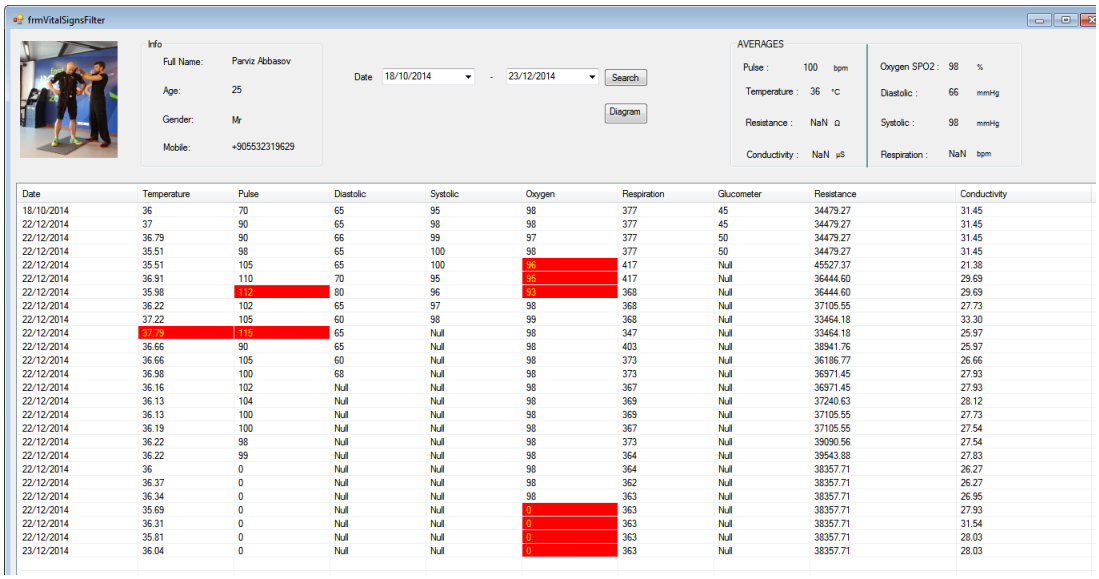
Şekil 5.4 : Vital Signs Form'u

Vital Calculator penceresi, bireyin spor aktivitesi zamanı harcadığı kaloriyi, günlük kalori gereksinim miktarını, vücut kitle indeksini, vücut yağ oranını, güvenli egzersiz nabız aralığını hesaplayan alanlardan oluşmuştur. (Şekil 5.5)



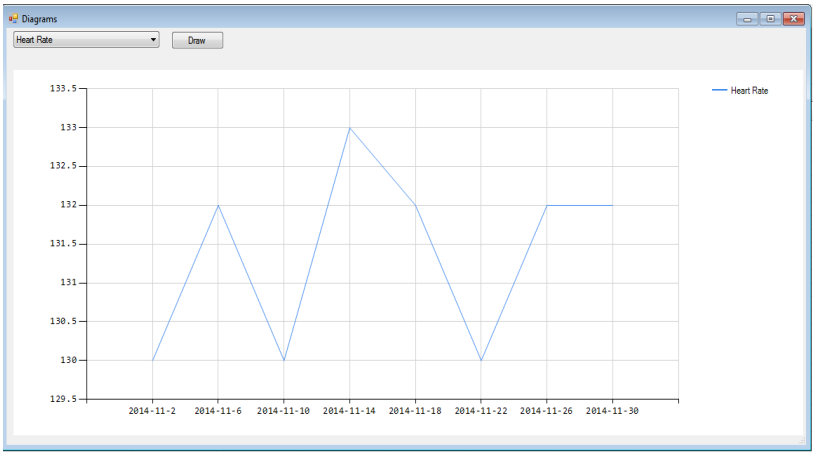
Şekil 5.5 : Vital Calculator Form'u

Raport penceresinde danışman 2 tarih aralığı seçerek veritabanına önceden kaydedilmiş bireyin sağlık bilgilerini otomatik listeleyebilir. Bu listede güvenli sağlık aralıklarını aşmış sağlık değerleri sistem tarafından otomatik kırmızı renkle belirtilmiştir. Pencerenin Averages kısmında ise seçilmiş tarih aralıklarındaki sağlık değerlerinin her birinin ortalaması hesaplanarak gösterilmiştir. (Şekil 5.6) Danışman penceredeki Diagrams düğmesine tıklayarak Diagrams penceresini açabilir.



Şekil 5.6 : Rapor Form'u Örneği

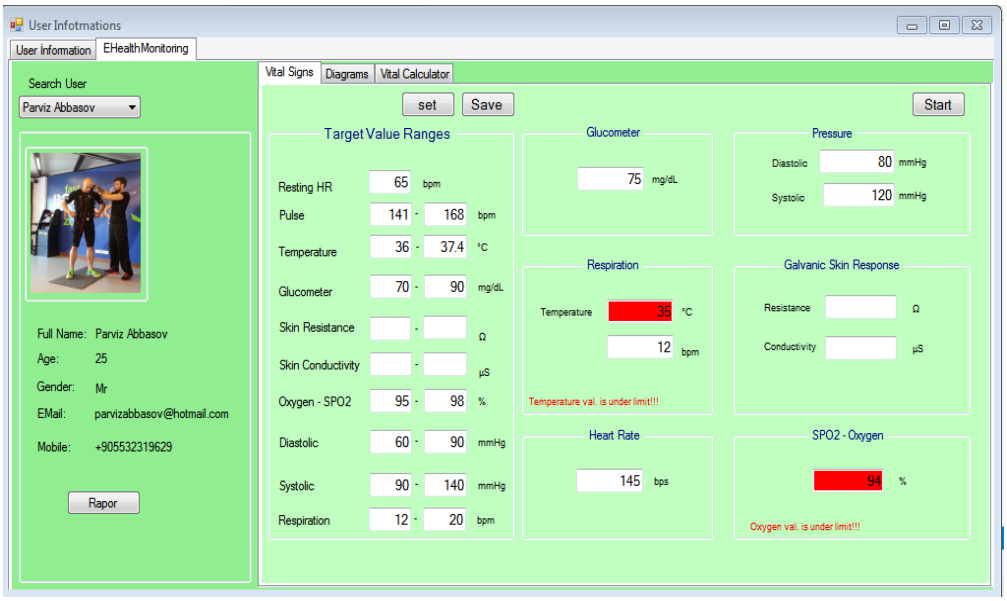
Diagrams penceresi bireyin spor aktivitesi zamanı günlük sağlık değerlerinin ortalamasını hesaplayarak bir aylık diagramını çiziyor. Bu diagramlar yardımıyla spor ve sağlık danışmanı bireyin bir ay içerisinde sağlık değerlerinin her birinin günlük çalışma programına göre nasıl değiştiğini görüp, analiz edebilir ve bireye daha uygun çalışma programı hazırlayabilir. (Şekil 5.7)



Şekil 5.7: Diagrams Form'u Örneği

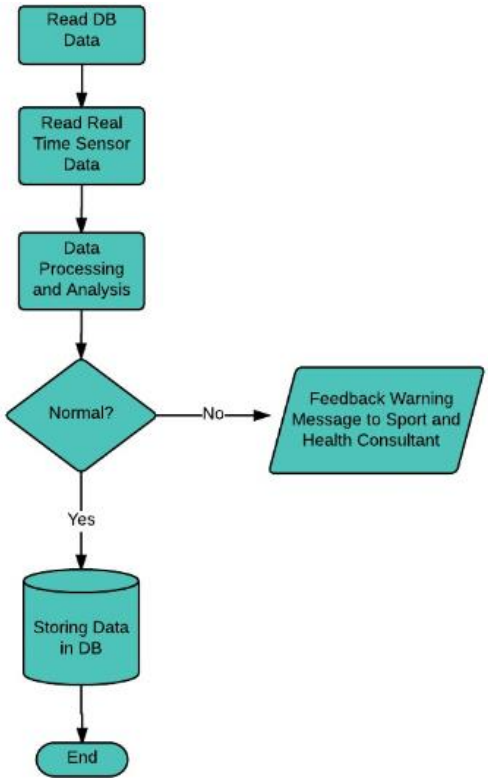
5.2. Veri Kontrol ve İşleme Modülü

Veri kontrölü ve işleme modülü sağlık izleme sisteminin en önemli modülüdür. Bu modül bireyin üzerinde sağlık sensörleriyle spor aktivitesi yaptığı zaman sağlık bilgilerinin nasıl değiştiğini arka planda kontrol ediyor. Ön planda ise spor danışmanı bu bilgileri uygulamanın Vital Signs penceresinde izleyebiliyor. (Şekil 5.8) Bu modül için geliştirilmiş algoritma elde edilen sağlık bilgilerinin güvenli olup olmadığını tespit etmek için sağlık kurumları tarafından önceden belirlenmiş olan sınırlara göre kıyaslama yapıyor. Bu bilgilerden her hangi biri veya hepsinin bu aralıklar dışında olduğu tespit edildiğinde spor ve sağlık danışmanına otomatik olarak, anında uyarı mesajı veya mesajları veriliyor. Bu işlemin testi yapılmış olup, uyarı mesajının örneği Resim 20.'de gösterilmiştir. Bu testde vücut ısısı ve kandakı oksijen miktarı değerlerinde limit aşımı tespit edilmiştir. Bu limit aşımı uyarısı mesaj olarak danışmana Vital Signs penceresinden iletilmiştir.



Şekil 5.8 : Vital Signs Form'dakı Uyarı Mesajı Örneği

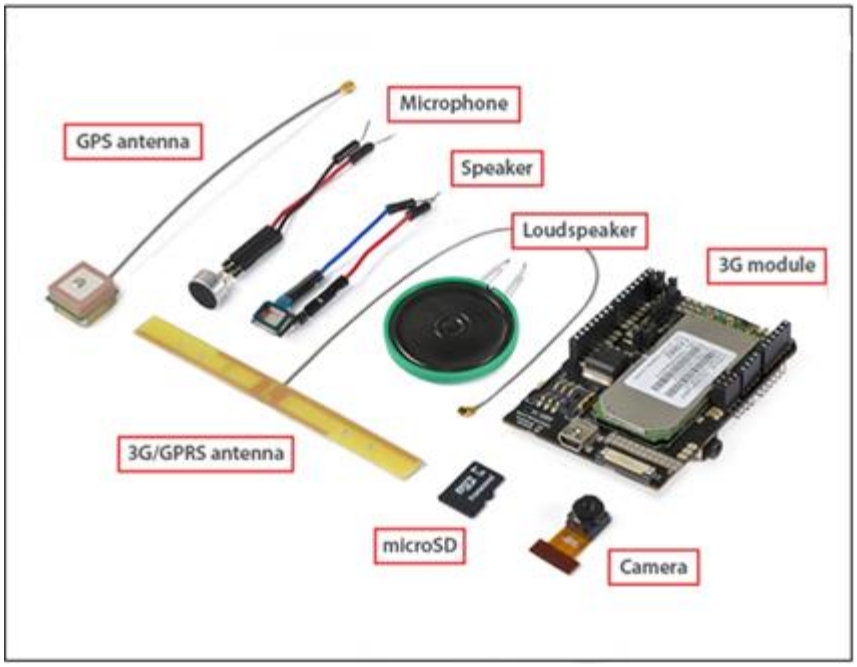
Tüm işlemlerden sonra sensörden gelen bilgilerin hepsi veritabanına kaydedilerek tutuluyor. Bu süreçler Şekil 5.9'dakı akış diyagramına göre yapılmaktadır.



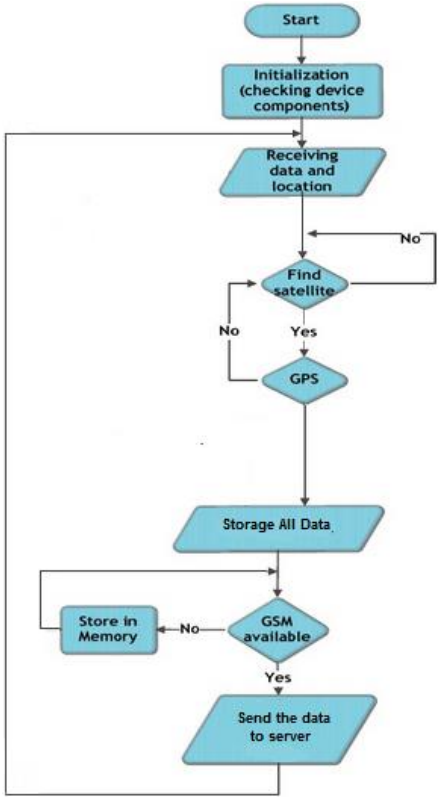
Şekil 5.9 : Veri Kontrol ve İşleme Modülü Akış Diagramı

5.3. Spor İzleme Sistemi Modelinin Haberleşme Modülü

Bu modül arduino tabanlı 3G internet platformu kullanarak sağlık sensör platformu ve uygulama arasındakı haberleşmeni sağlamak için amaçlanmıştır.(Şekil 5.10) Bu platform internet bağlantısı ve haberleşmeni yüksek hızlı WCDMA ve HSPA hücresel ağlar kullanarak gerçekleştiriyor. Sağlık Sensör platformuyla uygulama arasındaki 3G internet haberleşmesi Şekil 5.11.'deki akış diyagramı göre yapılmaktadır.



Şekil 5.10 : 3G Internet Bağlantısı Platformu



Şekil 5.11: 3G Haberleşne Modülü Akış Diagramı

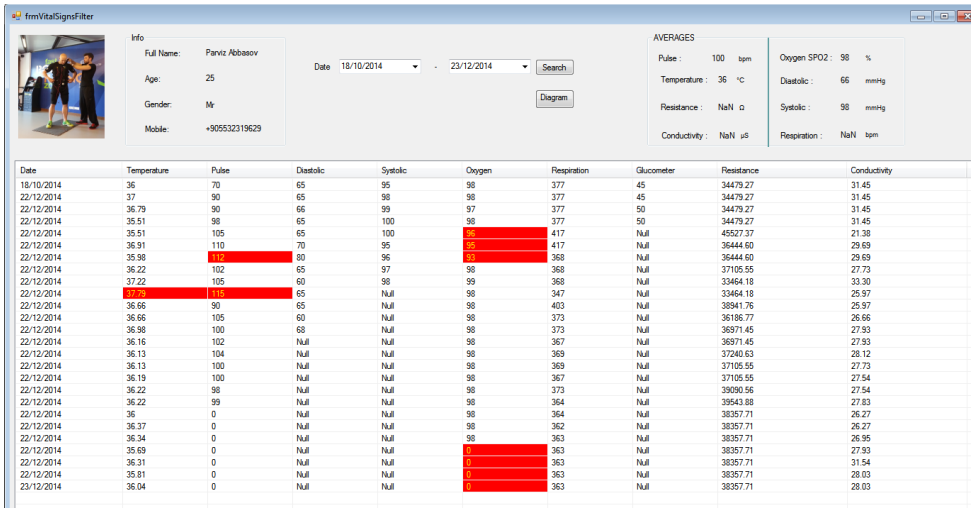
5.4. Spor İzleme Sistemi Modelinin Testi

Bölüm 2'de bahs edilen günümüzdeki spor uygulamaları (Nike+ipod, endomondo ve s.) sadece kullanıcı için olup adım sayar ve nabız sensörüyle kullanıcıya ne kadar kalori yaktiği, kaç kilometre koştuğu ve bu gibi bir takım bilgiler sunmaktadır. Bu programların en büyük açığı bireyin sağlık durumunun yeterince kontrol edilmemesidir. Diğer yandan ise sağlık ve spor danışmanının bireye önerdiği çalışma programının onun sağlık bilgilerine nasıl etki etdiğini izleyememesidir. Bu durumda spor danışmanı bireylere özel çalışma programları geliştirmekte zorluklar çekebilir. Bunun sonucunda birey sağlık sorunu yaşayabilir ve yahud amacına ulaşmakta zorlanabilir ve zamanını iyi değerlendiremez.

Geliştirmiş olduğum spor izleme sistemi modeli spor ve sağlık danışmanlarına yardım amacı taşımaktadır. Bu sistemi günümüzdeki var olan spor uygulamalarına entegrasyon ederek insanların sağlıklı yaşamasını daha da kolaylaştırabiliriz. Bu sistemle spor aktivitesi zamanı biryandan kullanıcı kendi mobil telefonu veya tabletinde sağlık bilgilerini izleyerken diğer yandan ise sağlık ve spor danışmanının kontrolü altında olucaktır. Programın diğer amaçlarından biri de spor ve sağlık

danışmanlarına bireyleri spor aktivitesi zamanı herhangi bir sağlık riski yaşanmasını önlemesi veya anında mudahele etmesine yardımcı olmaktır. Bununla yanaşı danışmana bireye önerdiği çalışma programı zamanı bu programda olan egzersizlerin her birinin bireyin sağlık durumunu nasıl etkilediğini, vücutun pisikolojik ve fiziksel tepkilerini izleyebilmesini sağlamaktır. Bununlada danışman yapılan egzersizin süresini ve ağırlığını kişiye göre değiştirebilir. Böyle bir sistemle insanlar kendine uygun çalışma programlarını yapıcak, zamanlarını iyi değerlendiricek ve kendilerini sağlık güvencesi altında hissedicekler. Bu sistemin bir diğer avantajı ise bireyler mekandan yerden bağımsız şekilde istedikleri yerde spor yapabilecek ve 3g internet bağlantısı ve gps yardımıyla spor danışmanları onları uzaktan kolayca control edip güvence altında tutabilecekler. Kaydedilen bilgiler sağlık danışmanına sistem tarafından otomatik sunulmaktadır. Danışman bu bilgilere Rapor form'unda olan 2 tarih aralığını seçerek otomatik liste halinde ulaşabilir. Bu listede sağlık kurumları tarafından belirlenmiş aralıkları aşan bilgiler kırmızı renkle belirtilmektedir. Diagramlar form'unda ise bireyin 1 ay içerisinde sağlık değerlerinin değişme grafikini görebilir.

Spor izleme sistemi modelinin 1 ay içerisinde 4 gün aralıklarla spor zamanı testi yapılmıştır. Bu test zamanı sensörlerden gelen tüm sağlık bilgileri veritabanına sürekli kaydedilmiştir. Bu kaydedilen bilgiler Resim 5.12.'deki Rapor form'unda listelenmiştir. Bu sağlık değerlerinin hepsi veritabanına kaydedilmeden önce sürekli olarak sağlık kurumları tarafından belirlenmiştir sağlık değerleri aralıklarına göre sistem tarafından otomatik kontrol edilmiştir. Sağlık kurumları tarafından belirlenen bu değerler Çizelge 5.1'deki tabloda belirtilmiştir.[20, 21, 22] Resim 5.13.'deki tablodakı rapor sistem tarafından 1 ay içerisinde kaydedilen her bir sağlık değerinin günlük ortalaması hesaplanarak hazırlanmıştır. Diagrams form'unda bu tablodakı bilgiler kullanılarak her bir sağlık değerine göre system tarafından otomatik olarak diagramlar çizilmiştir. (Şekil 5.13, 5.14, 5.15, 5.16, 5.17, 5.18)



Date	Temperature	Pulse	Diastolic	Systolic	Oxygen	Respiration	Glucometer	Resistance	Conductivity
18/10/2014	36	70	65	95	98	377	45	34479.27	31.45
22/12/2014	37	90	65	98	98	377	45	34479.27	31.45
22/12/2014	36.79	90	66	99	97	377	50	34479.27	31.45
22/12/2014	35.51	98	65	100	98	377	50	34479.27	31.45
22/12/2014	35.51	105	65	100	96	417	Null	45527.37	21.38
22/12/2014	36.91	110	70	95	95	417	Null	36444.60	29.69
22/12/2014	35.98	112	80	96	93	368	Null	36444.60	29.69
22/12/2014	36.22	102	65	97	98	368	Null	37105.55	27.73
22/12/2014	37.22	105	60	98	99	368	Null	33464.18	33.30
22/12/2014	37.79	115	65	Null	98	347	Null	33464.18	25.97
22/12/2014	36.66	90	65	Null	98	403	Null	38941.76	25.97
22/12/2014	36.66	105	60	Null	98	373	Null	36186.77	26.66
22/12/2014	36.98	100	68	Null	98	373	Null	36971.45	27.93
22/12/2014	36.16	102	Null	Null	98	367	Null	36971.45	27.93
22/12/2014	36.13	104	Null	Null	98	369	Null	37240.63	28.12
22/12/2014	36.13	100	Null	Null	98	369	Null	37105.55	27.73
22/12/2014	36.19	100	Null	Null	98	367	Null	37105.55	27.54
22/12/2014	36.22	98	Null	Null	98	373	Null	39090.56	27.54
22/12/2014	36.22	99	Null	Null	98	364	Null	39543.88	27.83
22/12/2014	36	0	Null	Null	98	364	Null	38357.71	26.27
22/12/2014	36.37	0	Null	Null	98	362	Null	38357.71	26.27
22/12/2014	36.34	0	Null	Null	98	363	Null	38357.71	26.95
22/12/2014	35.69	0	Null	Null	0	363	Null	38357.71	27.93
22/12/2014	36.31	0	Null	Null	0	363	Null	38357.71	31.54
22/12/2014	35.81	0	Null	Null	0	363	Null	38357.71	28.03
23/12/2014	36.04	0	Null	Null	0	363	Null	38357.71	28.03

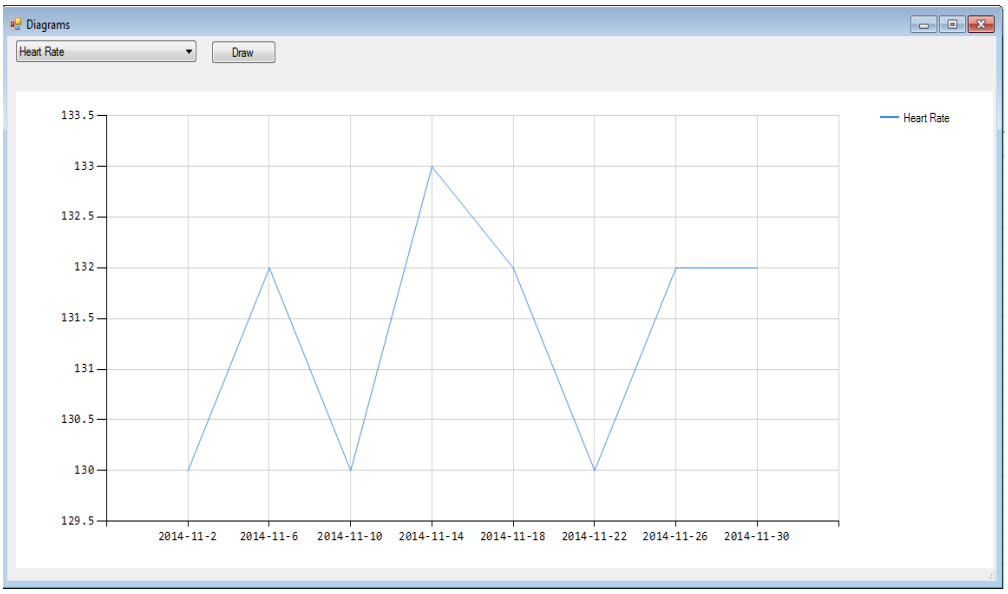
Şekil 5.12 : Rapor Form'u Örneği

Çizelge 5.1: Güvenli Sağlık Değerleri Aralığı

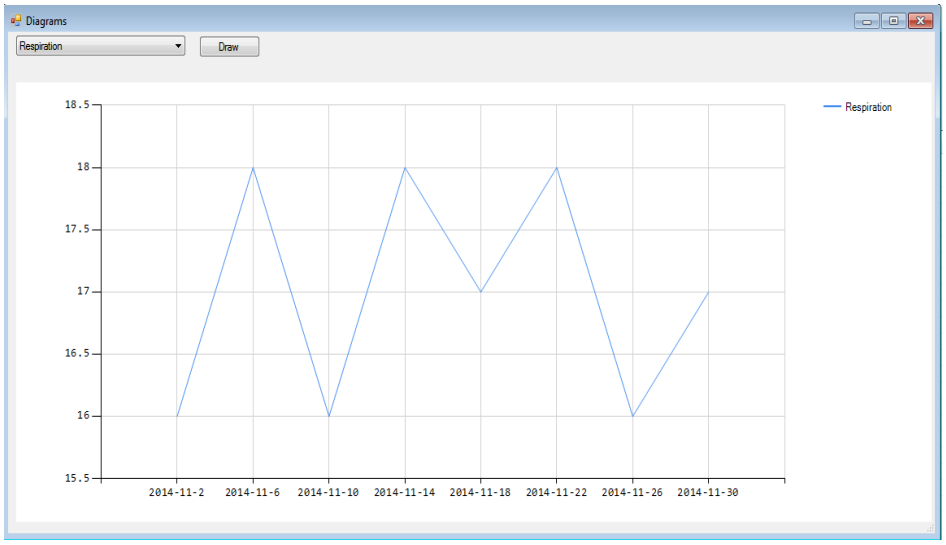
Temperatur	36 - 37,4C
Nabız	141 - 168 bpm
Diyastolik Kan Basıncı	60 - 90 mmHg
Sistolik Kan Basıncı	90 - 140 mmHg
Hava Akışı (nefes)	12 - 20 bpm
Kandakı Oksijen	95 - 98 %

Çizelge 5.2 : Spor Zamanı Kaydedilen Sağlık Testi

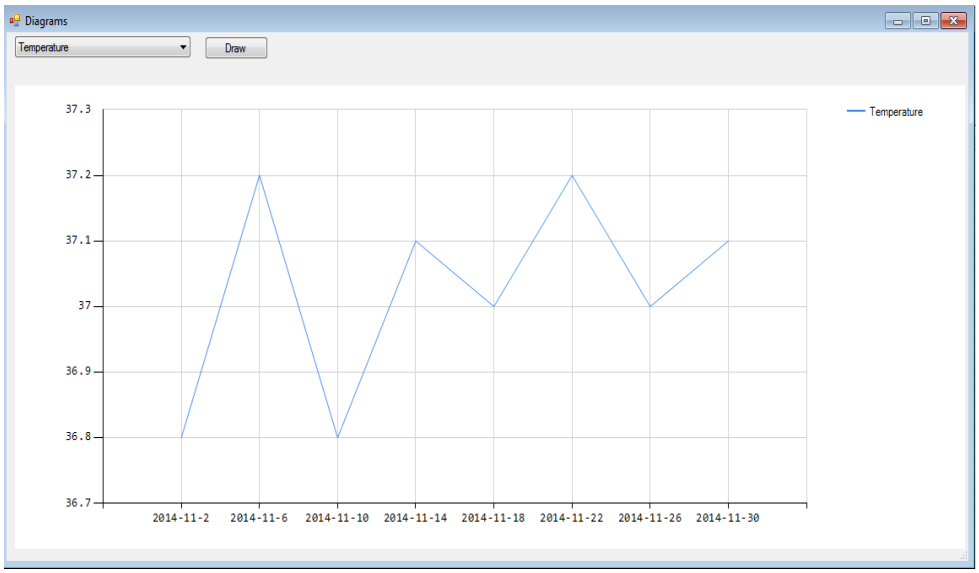
Tarih	Vüc. Isısı (ºC)	Nabız (bpm)	Hava Akışı (bpm)	Diyastolik Kan Bas. (mmHg)	Sistolik Kan Bas. (mmHg)	Kan. Oksijen (%)
02.11.2014	36.8	120	15	80	120	98
04.11.2014	37	126	15	80	120	98
06.11.2014	37.2	130	18	78	125	97
08.11.2014	37	132	17	80	125	98
10.11.2014	36.8	125	16	80	120	98
12.11.2014	36.9	125	15	78	122	98
14.11.2014	37.1	130	18	75	125	98
16.11.2014	37	128	17	80	120	97
18.11.2014	37	120	17	80	120	98
20.11.2014	36.8	115	16	79	122	97
22.11.2014	37.2	135	18	80	125	97
24.11.2014	37	130	15	80	120	98
26.11.2014	37	130	16	78	120	98
28.11.2014	37.1	132	17	78	122	97
30.11.2014	37.3	135	17	80	120	98



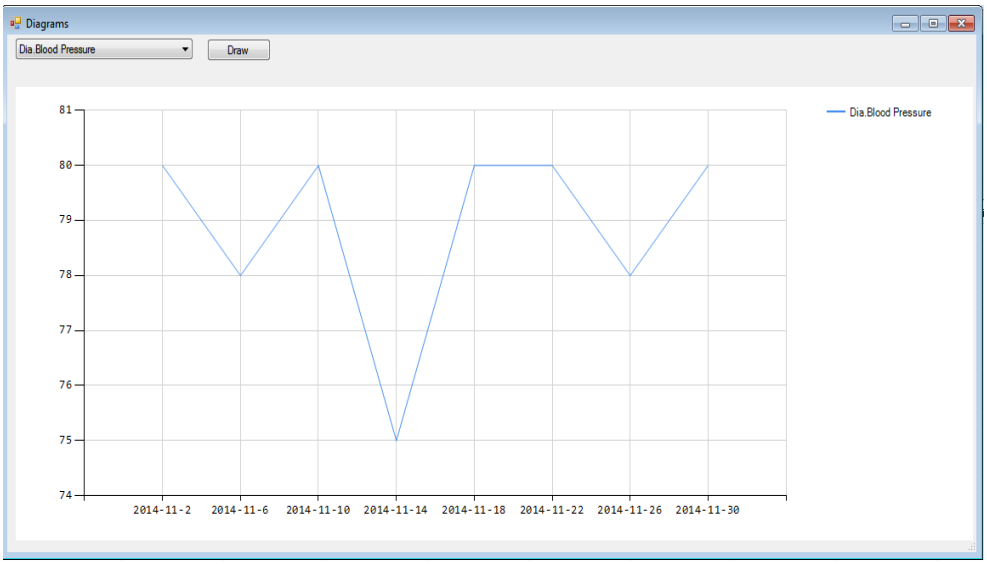
Şekil 5.13 : Nabız Diagramı Testi.



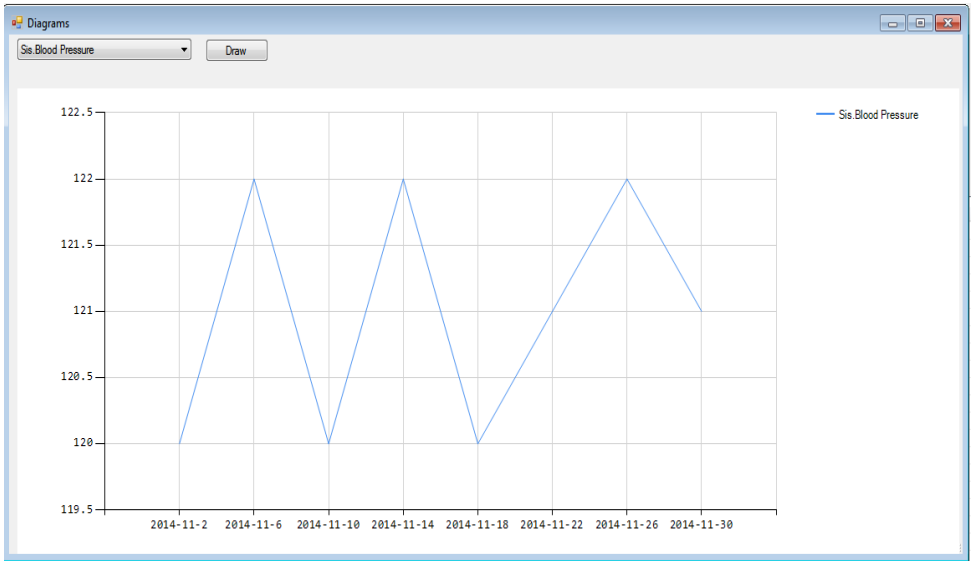
Şekil 5.14 : Hava Akışı Diagramı Testi.



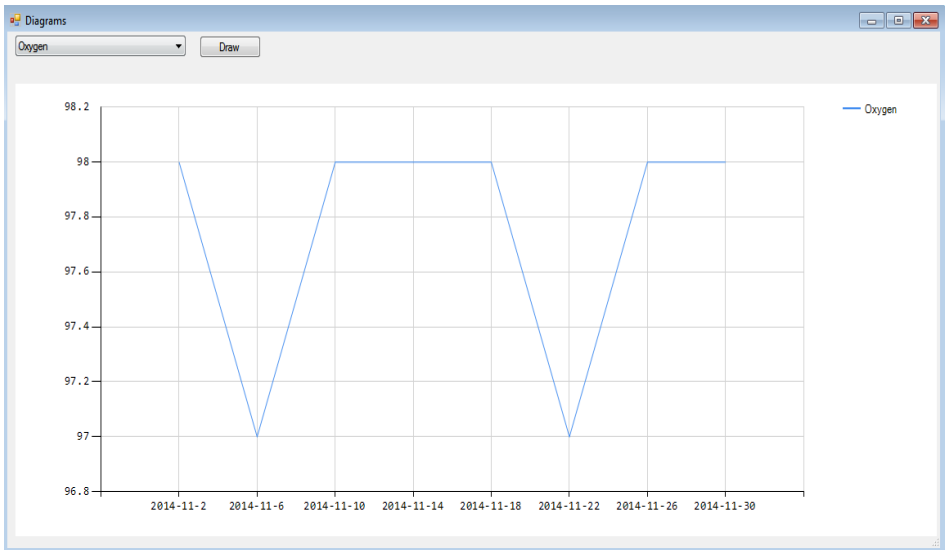
Şekil 5.15 : Vücut Isısı Diagramı Testi.



Şekil 5.16 : Diyastolik Kan Basıncı Diagramı Testi.



Şekil 5.17 : Sistolik Kan Basıncı Diagramı Testi.



Şekil 5.18 : Kandakı Oksijen Diagramı Testi.

6. SONUÇ

Bu tezin temel amacı, spor ve sağlık danışmanlarına, spor aktiviteleriyle ilgilenen kişilerin, sağlık bilgilerini analiz etmesine, sağlık durumlarını kolayca kontrol altında tutabilmesine ve daha uygun bir çalışma programı hazırlamalarına teknoloji desteğiyle yardımcı olmaktır.

Sağlık İzleme Sistemi Modeli sensör platformundan alınan verileri işleyip, sonrasında 2 seviyeye göre (normal, uyarı) sınıflandırarak karar verebilme özelliğine sahiptir ve bu uyarıyı kırmızı renkli bir mesajla Vital Signs form'undan sağlık ve spor danışmanına iletebilmektedir. Diğer yandan ise birey sağlık sensör platformu üzerinde olan alarm sensörü yardımıyla uyarabilmektedir.

Bu sistemde spor aktivitesi zamanı, kişi sağlık bilgilerinin önceden elektronik ortamda doktor ve diğer sağlık personellerinin belirlediği güvenli sağlık verileri aralıklarına göre kontrolü sağlanmaktadır. Sağlık İzleme Sistemi Modeli her spor gününün ardından, sağlık kontrolü için hastaneye gitme gereksinimini ve maliyetini minimuma indiriyor. Bu sistem günümüzde var olan spor uygulamalarına entegrasyon edilerek bu programları daha yararlı ve güvenli hale getirebilir.

Gelecekte insanların bu sistemle daha sağlıklı spor yapabilmelerini sağlamak için birçok geliştirilmeler yapılabilinir. İnsanın spor zamanı üzerinde daha rahat gezdirebileceği, giyebileceği ve daha az enerji tüketen biyometrik sensörler dizaynedilebilir.

KAYNAKLAR

[1] **Silberner, Joanne** (June 7, 2010). *"100 Years Ago, Exercise Was Blended Into Daily Life"*. npr.org. Retrieved 23 November 2010.

[2] **Council Of The Europe Union.** Council recommendation on promoting health-enhancing physical activity across sectors COUNCIL OF THE EUROPEAN UNION Brussels, 25 November 2013 (OR. en) 15575/13 LIMITE SPORT 93 SAN 424 EDUC 412 ENV 1001 TRANS 554

[3] **CasaDJ, Armstrong LE, Kenny GP, O'Connor FG, Huggins RA.** *Exertional Heat Stroke: New concepts regarding cause and care.* Curr. Sports Med. Rep. 2012;

[4] **Barry A. Franklin, Ph.D., FACSM (chair); Gary J. Balady, M.D., FACC, Kathy Berra, M.S.N., ANP; Neil F. Gordon, M.D., FACSM, and Michael L. Pollock, Ph.D., FACSM.** American College of Sport Medicine. Exercise for Persons with Cardiovascular Disease. American College of Sports Medicine.

[5] **American Heart Association.** *Understanding and Managing High Blood Pressure.* American Heart Association. (10.12.2014)

[6] **National Institutes Health (NIH) .** NIH Publication No. 14–5180 May 2014. *What I need to know about Physical Activity and Diabetes.*

[7] **National Institutes Health (NIH) .** NIH Publication No 14-5180. May 2014. *National Diabetes Information Clearinghouse.*

[8] **Exercise is a Medicine Australia Organisation.** *Exercise Is Medicine.* Australia Factsheet. Published May 2014 by Exercise Is Medicine Australia.

[9] **Evgeny Stankevich, Ilya Paramonov** Sport Training System Based on ECG Monitor. Yaroslavl State University Yaroslavl, Russia

[10] **Alan F. Smeaton#1, #2, Dermot Diamond#1, Philip Kelly#1, #2, Kieran Moran#3, King-Tong Lau#1, Deirdre Morris#1, Niall Moyna#3, Noel E. O'Connor#1, #2 and Kirk Zhang#1,** #2. #1Adaptive Information Cluster and #2Centre for Digital Video Processing #3

[11] **National Health Service (UK)** Center for Evidence-based Purchasing. *Project initiationdocument: Pulse oximeters*. 2009.

[12] **WONCA/ICC.** *Global Primary Care and Patient Education.* WONCA/ICC document : Clinical Use Of Pulse Oximetry Pocket Reference . (2010).

[13] **American Council on Exercise.** *Fit Facts***.** (15.12.2014)

[14] **BuonoMJ, Wall AJ.** *Effect of hypohydration on core temperature during exercise in temperate and hot environments.* Pflügers Arch.Eur.J.Physiol. 2000;440(3):476–480.

[15] **J.Wen #1, K.Inthavong #1, Z.F.Tian #1, J.Y.Tu #1, C.L.Xue #2 and C.G.Li #2.** 1 School of Aerospace,Mechanical and Manufacturing Engineering, RMIT University,Melbourne,VIC,3083,Australia 2School of Health Sciences,RMIT University, Melbourne,VIC,3083, Australia. *Airflow Patterns in Both Sides of a Realistic Human Nasal Cavity for Laminar and Turbulent Conditions.* 16th Australasian Fluid Mechanics Conference Crown Plaza, Gold Coast, Australia 2-7 December 2007.

[16] **Simon SB, Clark RA**. *Using pulse oximetry: a review of pulse oximetry use in acute care medical wards.* Clinical Effectiveness in Nursing, 6; 106-110. (2002)

[17] **NHS Wirral Community** WONCA/ICC (2010).Pocket Reference. *Clinical Procedure Procedure for pulse OXIMETRY/SPO2.*

[18] **Libelium**. *Guidance for e-Health Sensor Platform.* (10.11.2014)

[19] **Hale Kula. Cem Süer.** *Kısa Süreli Egzersizin Antrene Sporcularda deri iletkenliğine etkisi.* Sağlık Bilimleri Dergisi (Journal of Health Sciences) 15(2) 107-115, 2006

[20] **Joseph V. Stewart, M.D. Landes.** (2003) *Bioscience Vital Signs and Resuscitation.* Georgetown, Texas, USA

[21] **Minor, M.A., Minor, S.** *Patient Care Skills, 6th ed*. Pearson Prentice Hall: Upper Saddle River, NJ. (2006).

[22] **BuonoMJ, Wall AJ.** *Effect of hypohydration on core temperature during exercise in temperate and hot environments.* Pflügers Arch.Eur.J.Physiol. 2000;440(3):476–480.

EKLER

EK A: Uygulamanın Kaynak Kodları

```
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 System.Configuration;
using System.Net.Configuration;
namespace Fitness_Guide
{
   public partial class Form1 : Form
   {
      SqlConnection myConn = new SqlConnection("Data Source=ACER-PC\\SQLEXPRESS; Initial Catalog=DB_EHEALTH; User id=SA; Password=perviz1; Integrated Security=True");
      public Form1()
      {
         InitializeComponent();
```

```
        }
        private void btnLogIn_Click(object sender, EventArgs e)
        {
            bool verified = false;
            var selectCmd = new
SqlCommand("S_VerificationCONSULTANTINFORMATIONS", myConn);
            selectCmd.CommandType = CommandType.StoredProcedure;
            myConn.Open();
            SqlDataAdapter da = new SqlDataAdapter(selectCmd);
            SqlDataReader reader = selectCmd.ExecuteReader();
            while (reader.Read())
            {
                if (txtLogin.Text == (string)reader["ConsultantLogin"] &&
txtPassword.Text == (string)reader["ConsultantPassword"])
                {
                    verified = true;
                }           }
            reader.Close();
            myConn.Close();
            if (verified)
            {
                frmUserInformations frmUI = new frmUserInformations();
                frmUI.ShowDialog();
            }
            else
```

```
            if (txtLogin.Text == "parvizabbasov" && txtPassword.Text ==
"parviz123")
            {
                frmAdministrator frmAdmin = new frmAdministrator();
                frmAdmin.ShowDialog();
            }
        else
        {
            MessageBox.Show("Kullanıcı veya Şifre yalnış");
        }
      }
   }
}
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;
namespace Fitness_Guide
{
   public partial class frmAdministrator : Form
```

```
{
    SqlConnection myConn = new SqlConnection("Data Source=ACER-PC\\SQLEXPRESS; Initial Catalog=DB_EHEALTH; User id=SA; Password=perviz1; Integrated Security=True");
    public frmAdministrator()
    {
        InitializeComponent();
    }
    private void PopulateComboBox()
    {
        var selectCmd = new SqlCommand("S_VerificationCONSULTANTINFORMATIONS", myConn);
        selectCmd.CommandType = CommandType.StoredProcedure;
                myConn.Open();
        SqlDataAdapter da = new SqlDataAdapter(selectCmd);
        SqlDataReader reader = selectCmd.ExecuteReader();
        while (reader.Read())
        {
            lstConsultantID.Add((int)reader["C_ID"]);
            cmbConsultantID.Items.Add(reader["Name"] + " " + reader["Surname"]);
        }
        myConn.Close();
    }

    public static int cmbSelectedIndex;
    bool NewConsultant;
```

```
        List<int> lstConsultantID = new List<int>();
SqlCommand("S_CI_CONSULTANTINFORMATIONS", myConn);
            selectcmd.CommandType = CommandType.StoredProcedure;
            selectcmd.Parameters.AddWithValue("@C_ID",
        private void cmbConsultantID_SelectedIndexChanged(object sender,
EventArgs e)
        {
            NewConsultant = false;
            var selectcmd = new
SqlCommand("S_CI_CONSULTANTINFORMATIONS", myConn);
            selectcmd.CommandType = CommandType.StoredProcedure;
            selectcmd.Parameters.AddWithValue("@C_ID",
lstConsultantID[cmbConsultantID.SelectedIndex]);
            myConn.Open();
            SqlataAdapter da = new SqlDataAdapter(selectcmd);
            SqlDataReader reader = selectcmd.ExecuteReader();
            while (reader.Read())
            {
                txtConsultantName.Text = (string)reader["Name"];
                txtConsultantSurname.Text = (string)reader["Surname"];
                txtConsultantCity.Text = (string)reader["City"];
                txtConsultantCountry.Text = (string)(reader["Country"]);
                txtConsultantEMail.Text = (string)reader["EMail"];
                txtConsultantPhone.Text = (string)reader["Phone"];
                txtConsultantMobile.Text = (string)reader["Mobile"];
                cmbConsultantNationality.Text = (string)reader["Nationality"];
```

```
            cmbConsultantGender.Text = (bool)(reader["Gender"]) ? "Mr" : "Ms";
            txtConsultantLogin.Text = (string)reader["ConsultantLogin"];
            txtConsultantPassword.Text = (string)reader["ConsultantPassword"];

        }
        reader.Close();
        myConn.Close();
    }
    private void btnNewConsultant_Click(object sender, EventArgs e)
    {
        NewConsultant = true;
        ResetFields();
        cmbConsultantID.Items.Clear();
        PopulateComboBox();
    }
    public void ResetFields()
    {
        txtConsultantName.Clear();
        txtConsultantSurname.Clear();
        txtConsultantCity.Clear();
        txtConsultantEMail.Clear();
        txtConsultantPhone.Clear();
        txtConsultantMobile.Clear();
        cmbConsultantGender.SelectedIndex = 0;
        cmbConsultantNationality.SelectedIndex = 0;
        txtConsultantCountry.Clear();
```

```
            txtConsultantAdress.Clear();
        }
        public bool LoginPasswordControl(bool logpasscontroller)
        {
            if (txtConsultantLogin.Text == "" || txtConsultantPassword.Text == "")
            {
                logpasscontroller = false;
            }
            if (txtConsultantPassword.TextLength < 5 ||
txtConsultantPassword.TextLength > 12)
            {
                logpasscontroller = false;
            }
            if (txtConsultantLogin.TextLength < 4 || txtConsultantPassword.TextLength
> 14)
            {
                logpasscontroller = false;
            }
            else
            {
                logpasscontroller = true;
            }
            return logpasscontroller;
        }
        public bool RequirementControl(bool controller)
        {
```

```
        if (txtConsultantName.Text == "" || txtConsultantSurname.Text == "" ||
dtpConsultantBirthDay.Text == "" || cmbConsultantGender.Text == "")
        {
            controller = false;
        }
        else
        {
            controller = true;
        }
        return controller;
    }
    Boolean controllers;
    private void btnSaveConsultant_Click(object sender, EventArgs e)
    {
        if (!LoginPasswordControl(controllers))
        {
            MessageBox.Show("Please fill the valid Login and Password (Login and
Password charachters must be 5-12 charachters!");
        }
        if (!RequirementControl(controllers))
        {
            MessageBox.Show("Please fill in the fields you leave blank!");
        }
        else
        {
            bool ReturnValue;
```

```
            bool Gender;


            if (cmbConsultantGender.Text == "Mr")
               Gender = true;
            else
               Gender = false;
            SqlCommand cmd = new SqlCommand();
            if (NewConsultant)
            {
               cmd = new SqlCommand("I_CI_CONSULTANTINFORMATIONS",
myConn);
               cmd.ComandType = CommandType.StoredProcedure;
               SqlParaeter prmKOD = new SqlParameter("@C_ID", SqlDbType.Int,
4);                prmKOD.Direction = ParameerDirection.Output;
               cmd.Parameters.Add(prmKOD);
            }
            else
            {
               cmd = new SqlCommand("U_CI_CONSULTANTINFORMATIONS",
myConn);
               cmd.ComandType = CommandType.StoredProcedure;
               cmd.Parameters.AddWithValue("@C_ID",
lstConsultantID[cmbConsultantID.SelectedIndex]);
            }
            cmd.Parameters.AddWihValue("@Name", txtConsultantName.Text);
            cmd.Parameters.ddWithValue("@Surname", txtConsultantSurname.Text);
```

```
            cmd.Parameters.AddWithValue("@BirthDay",
dtpConsultantBirthDay.Text);
            cmd.Parameters.AddWithValue("@Gender", Gender);
            cmd.Parameters.AddWithValue("@Nationality",
cmbConsultantNationality.Text);
            cmd.Parameters.AddWithValue("@Phone", txtConsultantPhone.Text);
            cmd.Parameters.AddWithValue("@Mobile", txtConsultantMobile.Text);
            cmd.Parameters.AddWithValue("@EMil", txtConsultantEMail.Text);
            cmd.Parameters.AddWithValue("@Country", txtConsultantCountry.Text);
            cmd.Parameters.AddWithValue("@City", txtConsultantCity.Text);
            cmd.Parameters.AddWithValue("@Adress", txtConsultantAdress.Text);
            cmd.Parameters.AddWithValue("@ConsultantLogin",
txtConsultantLogin.Text);
            cmd.Parameters.AddWithValue("@CosultantPassword",
txtConsultantPassword.Text);
            myConn.Open();
            ReturnValue = Convert.ToBoolean(cmd.ExecuteNonQuery());
            myConn.Close();
            cmbConsultantID.Items.Clear();
            PopulateComboBox();

            if (NewConsultant)
                lblMsg.Text = (bool)ReturnValue ? "Cnsultant Created!" : "Error!";
            else
                lblMsg.Text = (bool)ReturnValue ? "Consultant Updated!" : "Error!";
        }
    }
```

```
        private void btnDeleteConsultant_Click(object sender, EventArgs e)
        {
            bool ReturnValue;
            var deleteCmd = new SqlCommand("D_UI_USERINFORMATIONS",
myConn);
            deleteCmd.CommandType = CommandType.StoredProcedure;
            deleteCmd.Parameters.AddWithValue("@U_ID",
lstConsultantID[cmbConsultantID.SelectedIndex]);
            myConn.Open();
            ReturnValue = Convert.ToBoolean(deleteCmd.ExecuteNonQuery());
            myConn.Close();
            lblMsg.Text = (bool)ReturnValue ? "Consultant Deleted!" : "Error!";
            ResetFields();
            cmbCnsultantID.Ites.Clear();
            PopulateComboBox();
        }
        private void frmAdministrator_Load(object sender, EventArgs e)
        {
            PopulateComboBox();
        }
    }
}
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.Configuration;
using System.Data.SqlClient;
using System.Windows.Forms.DataVisualization.Charting;
using System.IO.Ports;
using System.Threading;
namespace Fitness_Guide
{
   public partial class frmUserInformations : Form
   {
      SqlConnection myConn = new SqlConnection("Data Source=ACER-PC\\SQLEXPRESS; Initial Catalog=DB_EHEALTH; User id=SA; Password=perviz1; Integrated Security=True");
      public static SerialPort _serialport;
      public frmUserInformations()
      {
         InitializeComponent();
      }
      private delegate void SetTextDeleg(string text);
      private void frmUserInformations_Load(object sender, EventArgs e)
      {
         listBox1.Visble = false;
```

```
            btnStart.Enabled = false;
            btnCaculate.Enabled = false;
            PopulateComboBox();
            cmbGender.SelectedIndex = 0;
            cmbNationality.SelectedIndex = 0;
            cmbMartialStatus.SelectedIndex = 0;
        }
        List<int> lstUserID = new List<int>();
        private void cmbUserID_SelectedIndexChanged(object sender, EventArgs e)
        {
            cmbUserID2.SectedIndex = cmbUserID.SelectedIndex;

            NewUser = false;

            var selectcmd = new SqlCommand("S_UI_USERINFORMATIONS",
myConn);
            selectmd.CommandType = CommandType.StoredProcedure;
            selectcmd.Parameters.AddWithValue("@U_ID",
lstUserID[cmbUserID.SelectedIndex]);
            myConn.Open();
            SqlDataAdapter da = new SqlDataAdapter(selectcmd);
            SqlDataReader reader = selectcmd.ExecuteReader();
            while (reader.Read())
            {
                txtUserName.Text = (string)reader["Name"];
                txtUserSrname.Text = (string)reader["Surname"];
```

```
            txtCity.Text = (string)reader["City"];
            txtEMail.Text = (string)reader["EMail"];
            txtPhoe.Text = (string)reader["Phone"];
            txtMobile.Text = (string)reader["Mobile"];
            cmbMartialStatus.Text = (string)reader["MartialStatus"];
            cmbNationality.Text = (string)reader["Nationality"];
            cmbGender.Text = (bool)(reader["Gender"]) ? "Mr" : "Ms";
            txtCounry.Text = (string)(reader["Country"]);
            txtPostalCode.Text = (string)(reader["Zip"]);
            pictureBox1.Image = Image.FromFile((string)(reader["Photo"]));
            pictureBox2.Image = Image.FromFile((string)(reader["Photo"]));

            this.picturBox1.SizeMode = PictureBoxSizeMode.StretchImage;
            this.pictureBox2.SizeMode = PictureBoxSizeMode.StretchImage;
        }
        reader.Close();
        myConn.Close();
    }
    private void PopulateComboBox()
    {
        var selectCmd = new SqlCommand("S_UI_USERINFORMATIONS",
myConn);
        selectCmd.CommandType = CommandType.StoredProcedure;
        selectCmd.Parameters.AddWithValue("@U_ID", DBNull.Value);
        myConn.Open();
        SqlDataAdapter da = new SqlDataAdapter(selectCmd);
```

```
    SqlDataReader readr = selectCmd.ExecuteReader();


    while (reader.Read())
    {
        lstUrID.Add((int)reader["U_ID"]);
        cmbUserID.Items.Add(reader["Name"] + " " + reader["Surname"]);
        cmbUserID2.Items.Add(reader["Name"] + " " + reader["Surname"]);
    }
    reader.Close();
    myConn.Close();
}
private void btnReset_Click(object sender, EventArgs e)
{
    ResetFields();
}
public void ResetFields()
{
    txtCity.Clear();
    txtCountry.Clear();
    txtEMil.Clear();
    txtMobile.Clear();
    txtPhone.Clear();
    txtPostalCode.Clear();
    txtUseName.Clear();
    txtUserSurname.Clear();
    txtAress.Clear();
```

```
            cmbGender.SelectedIndex = 0;
            cmbNationality.SelectedIndex = 0;
            cmbMartlStatus.SelectedIndex = 0;
        }
        bool NewUser;
        private void btnNewUser_Click(object sender, EventArgs e)
        {           NewUser = true;
            ResetFields();
            cmbUserID.Items.Clear();
            PopulateComboBox();
        }
         public bool RequirementControl(bool controller)
        {
            if (txtUserName.Text == "" || txtUserSurname.Text == "" || dtpBirthDay.Text
== "" || cmbGender.Text == "" || ChosenPicRoute == "")
            {
                controller = false;
            }
            else
            {
                controller = true;
            }
            return controller;
        }
        private void btnSave_Click(object sender, EventArgs e)
        {
```

```
if (!RequirementControl(controllers))
{
    MessageBox.Show("Please fill in the fields you leave blank!");
}
else
{
    bool ReturnValue;
    bool Gender;
    if (cmbGender.Text == "Mr")
        Gender = true;
    else
        Gender = false;
    SqlCommand cmd = new SqlCommand();
    if (NewUser)
    {
        cmd = new SqlCommand("I_UI_USERINFORMATIONS", myConn);
        cmd.CommandType = CommandType.StoredProcedure;
        SqlParameter prmKOD = new SqlParameter("@U_ID", SqlDbType.Int,
        prmKOD.Direction = ParameterDirection.Output;
        cmd.Parameters.Add(prmKOD);
    }
    else
    {
        cmd = new SqlCommand("U_UI_USERINFORMATIONS", myConn);
        cmd.CommandType = CommandType.StoredProcedure;
```

```
                cmd.Parameters.AddWithValue("@U_ID",
lstUserID[cmbUserID.SelectedIndex]);


            }
            cmd.Parameters.AddWithValue("@Name", txtUserName.Text);
            cmd.Parameters.AddWithValue("@Surname", txtUserSurname.Text);
            cmd.Parameters.AddWithValue("@BirthDay", dtpBirthDay.Text);
            cmd.Parameters.AddWithValue("@Gender", Gender);
            cmd.Parameters.AddWithValue("@MartialStatus",
cmbMartialStatus.Text);
            cmd.Parameters.AddWithValue("@Nationality", cmbNationality.Text);
            cmd.Parameters.AddWithValue("@Phone", txtPhone.Text);
            cmd.Parameters.AddWithValue("@Mobile", txtMobile.Text);
            cmd.Parameters.AddWithValue("@EMail", txtEMail.Text);
            cmd.Parameters.AddWithValue("@Country", txtCountry.Text);
            cmd.Parameters.AddWithValue("@City", txtCity.Text);
            cmd.Parameters.AddWithValue("@Zip", txtPostalCode.Text);
            cmd.Parameters.AddWithValue("@Adress", txtAdress.Text);
            cmd.Parameters.AddWithValue("@Photo", ChosenPicRoute);
            myConn.Open();
            ReturnValue = Convert.ToBoolean(cmd.ExecuteNonQuery());
            myConn.Close();
            cmbUserID.Items.Clear();
            PopulateComboBox();
            if (NewUser)
                lblMsg.Text = (bool)ReturnValue ? "User Created!" : "Error!";
```

```
            else
                lblMsg.Text = (bool)ReturnValue ? "User Updated!" : "Error!";
        }
    }
    private void btnDelete_Click(object sender, EventArgs e)
    {
        bool ReturnValue;
        var deleteCmd = new SqlCommand("D_UI_USERINFORMATIONS",
myConn);
        deleteCmd.CommandType = CommandType.StoredProcedure;
        deleteCmd.Parameters.AddWithValue("@U_ID",
lstUserID[cmbUserID.SelectedIndex]);

        myConn.Open();
        ReturnValue = Convert.ToBoolean(deleteCmd.ExecuteNonQuery());
        myConn.Close();
        lblMsg.Text = (bool)ReturnValue ? "User Deleted!" : "Error!";
        ResetFields();
        cmbUserID.Items.Clear();
        PopulateComboBox();
    }
    public static int cmbSelectedIndex;
    private void cmbUserID2_SelectedIndexChanged(object sender, EventArgs e)
    {
        cmbUserID.SelectedIndex = cmbUserID2.SelectedIndex;
        cmbSelectedIndex = lstUserID[cmbUserID2.SelectedIndex];
```

```
            var selectcmd = new SqlCommand("S_UI_USERINFORMATIONS",
myConn);
            selectcmd.CommandType = CommandType.StoredProcedure;
            selectcmd.Parameters.AddWithValue("@U_ID",
lstUserID[cmbUserID2.SelectedIndex]);
            myConn.Open();
            SqlDataAdapter da = new SqlDataAdapter(selectcmd);
            string birthDay = "";
            SqlDataReader reader = selectcmd.ExecuteReader();
            while (reader.Read())
            {
                birthDay = (string)reader["BirthDay"];
                lbfFullName.Text = (string)reader["Name"] + " " +
(string)reader["Surname"];
                lbfEMail.Text = (string)reader["EMail"];
                lbfMobile.Text = (string)reader["Mobile"];
                lbfGender.Text = (bool)(reader["Gender"]) ? "Mr" : "Ms";
            }
            string YearOfBirth = birthDay.Substring(6);
            DateTime now = DateTime.Today;
            lbfAge.Text = (now.Year - int.Parse(YearOfBirth)).ToString();
            reader.Close();
            myConn.Close();
            btnStart.Enabled = true;
            btnCalculate.Enabled = true;
        }
```

```
public string Age(DateTime birthday)
{
   string YearOfBirth = lbfAge.Text.Substring(0, 6);
   DateTime now = DateTime.Today;
   int age = now.Year - int.Parse(YearOfBirth);
   if (now < birthday.AddYears(age)) age--;
   return age.ToString();
}
private void btnStart_Click(object sender, EventArgs e)
{
   ConfigureSerialPortConSettings();
}
private void ConfigureSerialPortConSettings()
{          try
   {
      string[] sports = new string[10];
      sports = SerialPort.GetPortNames();
      _serialport = new SerialPort("COM3", 115200, Parity.None, 8,
StopBits.One);
      _serialport.Handshake = Handshake.None;
      _serialport.ReadTimeout = 1;
      _serialport.WriteTimeout = 1;
      _serialport.Open();
      _serialport.DtrEnable = true;
      _serialport.Write("RESET");
      timer1.Start();
```

```
        }    catch (Exception e)
        {
            MessageBox.Show(e.ToString());
        }
    }
public void CalculateHearRate()
    {
        int MaxHR = 0;
        int Age = 0;
        int MaxHRR = 0;
        int RHR = 0;
        int LowTHR = 0;
        int UpTR = 0;
        if (txtRestingHR.Text == "")
            lblPulseControl.Text = "Please select the resting heart rate!";
        else
            RHR = Convert.ToInt32(txtRestingHR.Text);
        if (lbfAge.Text == "")
            MessageBox.Show("Please select the age!");
        else
                    if (lbfGender.Text == "Bay")
            MaxHR = 220 - Age; //Max HR for men
        else
            MaxHR = 208 - (Age * 82 / 100); //Max HR for women

        MaxHRR = MaxHR - RHR;
```

```
    LowTHR = MaxHRR * 6 / 10 + RHR;
    UpTHR = MaxHRR * 8 / 10 + RHR;
    Height = Convert.ToDouble(cmbHeight.Text);
    BMI = Weight / (Height / 100 * Height / 100);
    txtMaxHR.Text = MaxHR.ToString(); ;
    txtLowerTargetHR.Text = LowTHR.ToString();
    txtUpperTargetHR.Text = UpTHR.ToString();
}
public void CalculateBMI()
{
    double BMI = 0, Weight = 0, Height = 0;
    Weight = Convert.ToDouble(cmbWeight.Text);
    Height = Convert.ToDouble(cmbHeight.Text);
    BMI = Weight / (Height / 100 * Height / 100);
    txtBMI.Text = BMI.ToString();
    if (BMI <= 18.5)
        lbfBMI.Text = "Under Weight";
    lbfBMI.ForeColor = Color.Red;
    if (BMI > 18.5 && BMI < 24.9)
        lbfBMI.Text = "Normal Weight";
    lbfBMI.ForeColor = Color.Black;
    if (BMI > 30)
        lbfBMI.Text = "Obesity";
    lbfBMI.ForeColor = Color.Red;
}
public void CalculateCalorieBurn()
```

```
{       float weight;
    int age;
    int maxHRreached;
    float exerciseDuration;
    string CalorieBurned;
    maxHRreached = int.Parse(cmbMaxHRreached.Text);
    weight = float.Parse(cmbWeight.Text);
    age = int.Parse(lbfAge.Text);
    exerciseDuration = float.Parse(txtExerciseDuration.Text);
    if (lbfGender.Text == "Bay")
        CalorieBurned = (((-55.0969 + (0.6309 * maxHRreached) + (0.1988 *
weight) + (0.2017 * age)) / 4.184) * 60 * exerciseDuration).ToString();
    else
        CalorieBurned = (((-20.4022 + (0.4472 * maxHRreached) - (0.1263 *
weight) + (0.074 * age)) / 4.184) * 60 * exerciseDuration).ToString();
    txtCalorieBurned.Text = CalorieBurned;
}
public void CalculateBodyFatPercentage()
{           int age;
    float BMI;


    BMI = float.Parse(txtBMI.Text);
    age =int.Parse(lbfAge.Text);
    if (lbfGender.Text == "Bay")
        txtBodyFatPercentage.Text = ((1.20 * BMI) + (age * 0.23) - (1 * 10.8) -
5.4).ToString();
```

```
            else
                txtBodyFatPercentage.Text = ((1.20 * BMI) + (age * 0.23) - (0 * 10.8) -
5.4).ToString();
        }
        public void CalculateBMR()
        {
            float TargetBMR = 0; int age = 0; float height = 0; float weight = 0;
            float EstimatedCaloriIntake = 0;
                weight = float.Parse(cmbWeight.Text);
            height = float.Parse(cmbHeight.Text);
            age = int.Parse(lbfAge.Text);

            if (lbfGender.Text == "Bay")
                TargetBMR = (float)(655 + (9.6 * weight + (1.8 * height) - (4.7 * age)));
            else
                TargetBMR = (float)(655 + weight + height - age);
            if (cmbActivityIntensity.Text == "Sedentary")
                EstimatedCaloriIntake = (float)(TargetBMR * 1.2);
            if (cmbActivityIntensity.Text == "Lightly Active")
                EstimatedCaloriIntake = (float)(TargetBMR * 1.375);
           if (cmbActivityIntensity.Text == "Moderately Active")
                EstimatedCaloriIntake = (float)(TargetBMR * 1.55);
            if (cmbActivityIntensity.Text == "Very Active")
                EstimatedCaloriIntake = (float)(TargetBMR * 1.725);
            if (cmbActivityIntensity.Text == "Extra Active")
                EstimatedCaloriIntake = (float)(TargetBMR * 1.9);
```

```
    txtCalorieIntake.Text = EstimatedCaloriIntake.ToString();
    txtBMR.Text = TargetBMR.ToString();
}

private void btnCalculate_Click(object sender, EventArgs e)
{
    CalculateHearRate();
    CalculateCalorieBurn();
    CalculateBodyFatPercentage();
    CalculateBMR();
    CalculateBMI();
}
public string ChosenPicRoute;
public void UploadPhoto()
{
    ChosenPicRoute = string.Empty;
    openFileDialog1.InitialDirectory = "D:";
    openFileDialog1.Title = "Please Select Photo";
    openFileDialog1.FileName = "";
    openFileDialog1.Filter = "JPEG IMAGES|*.jpg|GIF IMAGES|*.gif";
    if (openFileDialog1.ShowDialog() == DialogResult.Cancel)
        MessageBox.Show("Operation Canceled!");
    else
        ChosenPicRoute = openFileDialog1.FileName;
    pictureBox1.Image = Image.FromFile(ChosenPicRoute);
    pictureBox2.Image = Image.FromFile(ChosenPicRoute);
```

```
            this.pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
            this.pictureBox2.SizeMode = PictureBoxSizeMode.StretchImage;
        }
        private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            UploadPhoto();
        }
        void sp_DataReceived(object sender, SerialDataReceivedEventArgs e)
        {
            Thread.Sleep(500);
            try
            {
                string data = _serialport.ReadLine();
                this.BeginInvoke(new SetTextDeleg(SensorDataReceived), new object[] {
data });
            }
            catch (TimeoutException)
            {
            }
        }
        private void SensorDataReceived(string data)
        {
            if (listBox1.Items.Count == 5)
            {
                if (listBox1.Items.Count == 5)
```

```
{   bool ReturnValue;
    SqlCommand cmd = new SqlCommand();
    cmd = new SqlCommand("I_VS_VITALSIGNS", myConn);
    cmd.CommandType = CommandType.StoredProcedure;
    SqlParameter prmKOD = new SqlParameter("@VS_ID",
SqlDbType.Int, 4);
    prmKOD.Direction = ParameterDirection.Output;
    cmd.Parameters.Add(prmKOD);

    cmd.Parameters.AddWithValue("@U_ID",
lstUserID[cmbUserID.SelectedIndex]);
    cmd.Parameters.AddWithValue("@VS_DATE", DateTime.Now);
    cmd.Parameters.AddWithValue("@TEMPRATURE",
txtTemprature.Text);
    cmd.Parameters.AddWithValue("@PULSE", txtPulse.Text);
    cmd.Parameters.AddWithValue("@OXYGEN", txtOxygen.Text);
    cmd.Parameters.AddWithValue("@GLUCOMETER", "0");
    cmd.Parameters.AddWithValue("@RESPIRATION",
txtRespiration.Text);
    cmd.Parameters.AddWithValue("@RESISTANCE",
txtResistance.Text);
    cmd.Parameters.AddWithValue("@CONDUCTIVITY",
txtConductivity.Text);
    cmd.Parameters.AddWithValue("@DYSTOLIC", txtDiastolic.Text);
    cmd.Parameters.AddWithValue("@SYSTOLIC", txtSystolic.Text);
    myConn.Open();
    ReturnValue = Convert.ToBoolean(cmd.ExecuteNonQuery());
```

```
            myConn.Close();
            listBox1.Items.Clear();
        }
    }
    listBox1.Items.Add(data);
    try
    {
        if (data.Substring(0, 8) == "Oxygen :")
            txtOxygen.Text = data.Substring(8);

        if (data.Subtring(0, 7) == "Pulse :")
            txtPulse.Text = data.Substring(7);
        if (data.Substring(0, 13) == "Temperature :")
            txtTemprature.Text = data.Substring(13);
        if (data.Sustring(0, 9) == "Airflow :")
            txtRespration.Text = data.Substring(9);

        if (data.Substring(0, 12) == "Resistance :")
            txtResisance.Text = data.Substring(12);
        if (data.Substring(0, 13) == "Conductance :")
            txtConductivity.Text = data.Substring(13);
    }
    catch { };
}
private void button3_Click(object sender, EventArgs e)
{     timer1.Start();
```

```
}     public string device;
 private void timer1_Tick(object sender, EventArgs e)
 {
    _serialport.DataReceived += new
SerialDataReceivedEvHandler(sp_DataReceived);


 }
 private void btnSet_Click(object sender, EventArgs e)
 {
    SetTargetRanges();
 }
public void SetTargetRanges()
 {
    var selectcmd = new
SqlCommand("S_T_TVR_TARGETVALUERANGES", myConn);
    selectcmd.CommanType = CommandType.StoredProcedure;
    myConn.open();


    SqlDataAdapter da = new SqlDataAdapter(selectcmd);
    SqlDataReader reader = selectcmd.ExecuteReader();
    while (reader.Read())
    {
       txtRestingHR.Text = (string)reader["REST_HR"];
       txtMaxTemperature.Text = (string)reader["MAX_TEMPRATURE"];
       txtMinTemperature.Text = (string)reader["MIN_TEMPRATURE"];
```

```
            txtMaxRespiration.Text = (string)reader["Max_Respiration"];
            txtMinRespiration.Text = (string)reader["Min_Respiration"];
            txtMaxSystolic.Text = (string)reader["MAX_SYSTOLIC"];
            txtMinSystolic.Text = (string)reader["MIN_SYSTOLIC"];
            txtMaxDiastolic.Text = (string)reader["MAX_DYSTOLIC"];
            txtMinDiastolic.Text = (string)reader["MIN_DYSTOLIC"];
            txtMaxOxygen.Text = (string)reader["MAX_OXYGEN"];
            txtMinOxygen.Text = (string)reader["M_OXYGEN"];
                }
          reader.Close();
           myConn.Close();
           btnStart.Enabled = true;
           btnCalculate.Enabled = true;
        }
        public void SaveTargetRanges()
        {
           bool ReturnValue;
           SqlCommand cmd = new SqlCommand();
           cmd = new SqlCommand("I_UI_USERINFORMATIONS", myConn);
           cmd.CommandType = CommandType.StoredProcedure;
           SqlParameter prmKOD = new SqlParameter("ID", SqlDbType.Int, 4);
           prmKOD.Direction = ParameterDirection.Output;
           cmd.Parameters.Add(prmKOD);
           cmd.Parameters.AddWithValue("@_DATE", DateTime.Now);
           cmd.Parameters.AddWithValue("@MAX_TEMPRATURE",
txtMaxTemperature.Text);
```

```
            cmd.Parameters.AddWithValue("@MIN_TEMPRATURE",
txtMinTemperature.Text);

            cmd.Parameters.AddWithValue("@RESTING_HR", txtRestingHR.Text);

            cmd.Parameters.AddWithValue("@MAX_DYSTOLIC",
txtMaxDiastolic.Text);

            cmd.Parameters.AddWithValue("@MIN_DYSTOLIC",
txtMinDiastolic.Text);

            cmd.Parameters.AddWithValue("@MAX_SYSTOLIC",
txtMaxSystolic.Text);

            cmd.Parameters.AddWithValue("@MIN_SYSTOLIC", txtMinSystolic.Text);

            cmd.Parameters.AddWithValue("@MAX_OXYGEN", txtMaxOxygen.Text);

            cmd.Parameters.AddWithValue("@MIN_OXYGEN", txtMinOxygen.Text);

            cmd.Parameters.AddWithValue("@MAX_RESPIRATION",
txtMaxRespiration.Text);

            cmd.Parameters.AddWithValue("@MIN_RESPIRATION",
txtMinRespiration.Text);

            cmd.Parameters.AddWithValue("@MAX_GLUCOMETER",
txtMaxGlucometer.Text);

            cmd.Parameters.AddWithValue("@MIN_GLUCOMETER",
txtMinGlucometer.Text);

            cmd.Parameters.AddWithValue("@MAX_RESISTANCE",
txtMaxResistance.Text);

            cmd.Parameters.AddWithValue("@MIN_RESISTANCE",
txtMinResistance.Text);

            cmd.Parameters.AddWithValue("@MAX_CONDUCTIVITY",
txtMaxConductivity.Text);

            cmd.Parameters.AddWithValue("@MIN_CONDUCTIVITY",
txtMinConductivity.Text);
```

```
    myConn.Open();
    ReturnValue = Convert.ToBoolean(cmd.ExecuteNonQuery());
    myConn.Close();
    lblTargetValueStatus.Text = (bool)ReturnValue ? "Saved!" : "Error!";
}
private void btnSaveTargetVal_Click(object sender, EventArgs e)
{
    SaveTargetRanges();
}       private void txtOxygen_TextChanged(object sender, EventArgs e)
{
    try
    {
        float oxygen;

        oxygen = float.Parse(txtOxygen.Text);

        if (oxygen > float.Parse(txtMaxOxygen.Text)) //min value 100 for ref
        {
            txtOxygen.BackColor = Color.Red;
            lblOxygenStatus.Text = "Oxygen val. is over limit!!!";
        }
        if (oxygen < float.Parse(txtMinOxygen.Text))  //max value 250 for ref
        {
            txtOxygen.BackColor = Color.Red;
            lblOxygenStatus.Text = "Oxygen val. is under limit!!!";
        }
```

```
        else
        {
            txtOxygen.BackColor = Color.Empty;
            lblOxygenStatus.Text = "";
        }
    }
    catch
    {
    }
}
private void txtPulse_TextChanged(object sender, EventArgs e)
{
    double Pulse = Convert.ToDouble(txtPulse.Text);

    if (Pulse > UpTHR && txtRestingHR.Text != "")
    {
        txtPulse.BackColor = Color.Red;
        lblPulseControl.Text = "Hear Rate val. is over target!";
    }
    if (Pulse < LowTHR && txtRestingHR.Text != "")
    {
        txtPulse.BackColor = Color.Red;
        lblPulseControl.Text = "Hear Rate val. is under target!";
    }           else
    {
        txtPulse.BackColor = Color.Empty;
```

```
        lblPulseControl.Text = "";
    }
}
int MaxHR = 0;
int MaxHRR = 0;
int RHR = 0;
int LowTHR = 0;
int UpTHR = 0;
public void CalculateTargetHeartRate()
{
    int Age = 0;

    if (txtRestingHR.Text == "")
        lblPulseControl.Text = "Please type the resting heart rate!";
    else
        RHR = Convert.ToInt32(txtRestingHR.Text);
    if (lbfAge.Text == "")
        MessageBox.Show("Please select the age!");
    else
        Age = Convert.ToInt32(lbfAge.Text);
    if (lbfGender.Text == "Mr")
        MaxHR = 220 - Age; //Max HR for men
    if (lbfGender.Text == "Ms")
        MaxHR = 208 - (Age * 82 / 100); //Max HR for women

    MaxHRR = MaxHR - RHR;
```

```
            LowTHR = MaxHRR * 6 / 10 + RHR;
            UpTHR = MaxHRR * 8 / 10 + RHR;
            if (txtMinPulse.Text == "" || txtMaxPulse.Text=="")
            {
                txtMinPulse.Text = LowTHR.ToString();
                txtMaxPulse.Text = UpTHR.ToString();
            }
        }
        private void txtDiastolic_TextChanged(object sender, EventArgs e)
        {
            float diastolicPressure;
            diastolicPressure = float.Parse(txtDiastolic.Text);

            if (diastolicPressure > float.Parse(txtMaxDiastolic.Text))  // Max
DiastolicPressure 120 for ref
            {
                txtDiastolic.BackColor = Color.Red;
                lblPressureStatus.Text = "Diastolic Pres. is over limit!";
            }
            if (diastolicPressure < float.Parse(txtMinDiastolic.Text))
            {
                txtDiastolic.BackColor = Color.Red;
                lblPressureStatus.Text = "Diastolic Pres. is under limit!";
            }
            else
            {
```

```
            txtDiastolic.BackColor = Color.Empty;
            lblPressureStatus.Text = "";
        }
    }
private void txtSystolic_TextChanged(object sender, EventArgs e)
    {
        float systolicPressure;
        systolicPressure = float.Parse(txtSystolic.Text);

        if (systolicPressure > float.Parse(txtMaxSystolic.Text))   //
maxSystolicPressure is 80 for ref.
        {
            txtSystolic.BackColor = Color.Red;
            lblPressureStatus.Text = "Systolic Pres. is over limit!";
        }
        if (systolicPressure < float.Parse(txtMinSystolic.Text))
        {
            txtSystolic.BackColor = Color.Red;
            lblPressureStatus.Text = "Systolic Pres. is under limit!";
        }
        else
        {
            txtSystolic.BackColor = Color.Empty;
            lblPressureStatus.Text = "";
        }
```

```
    }    private void txtTemprature_TextChanged(object sender, EventArgs e)
    {
      try
      {
        float temperature;
        temperature = float.Parse(txtTemprature.Text);
        if (temperature > float.Parse(txtMaxTemperature.Text)) //min value 100
for ref          {
          txtTemprature.BackColor = Color.Red;
          lblRespirationStatus.Text = "Temperature val. is over limit!!!";
        }
        if (temperature < float.Parse(txtMinTemperature.Text))  //max value 250
for ref          {
          txtTemprature.BackColor = Color.Red;
          lblRespirationStatus.Text = "Temperature val. is under limit!!!";
        }
        else
        {
          txtTemprature.BackColor = Color.Empty;
          lblRespirationStatus.Text = "";
        }
      }
      catch
      {
      }
    }
```

```
private void txtGlucometer_TextChanged(object sender, EventArgs e)
{
   try
   {
                    glucometer = float.Parse(txtGlucometer.Text);
     if (glucometer < float.Parse(txtMinGlucometer.Text)) //min value 100 for
      {
         txtGlucometer.BackColor = Color.Red;
         lblGlucometerStatus.Text = "Glucometer value is under limit!!!";
      }
      if (glucometer > float.Parse(txtMaxGlucometer.Text))
      {
         txtGlucometer.BackColor = Color.Red;
         lblGlucometerStatus.Text = "Glucometer value is over limit!!!";
      }
      else
      {
         txtGlucometer.BackColor = Color.Empty;
         lblGlucometerStatus.Text = "";
      }
   }
   catch
   {
   }
}
```

```
private void txtResistance_TextChanged(object sender, EventArgs e)
{
    try
    {
        float resistance;
        resistance = float.Parse(txtResistance.Text);
        if (resistance > float.Parse(txtMaxResistance.Text))
        {
            txtResistance.BackColor = Color.Red;
            lblGSRStatus.Text = "Resistance val. is over limit!";
        }
        if (resistance < float.Parse(txtMinResistance.Text))
        {
            txtResistance.BackColor = Color.Red;
            lblGSRStatus.Text = "Resistance val. is under limit!";
        }
        else
        {
            txtResistance.BackColor = Color.Empty;
            lblGSRStatus.Text = "";
        }
    }
    catch
    {


    }
```

```
}       private void txtConductivity_TextChanged(object sender, EventArgs e)
{
   try
   {
      float conductivity;
      conductivity = float.Parse(txtResistance.Text);

      if (conductivity > float.Parse(txtMaxConductivity.Text))
      {
         txtConductivity.BackColor = Color.Red;
         lblGSRStatus.Text = "Conductivity val. is over limit!";
      }
      if (conductivity < float.Parse(txtMinConductivity.Text))
      {
         txtConductivity.BackColor = Color.Red;
         lblGSRStatus.Text = "Conductivity val. is under limit!";
      }
      else
      {
         txtConductivity.BackColor = Color.Empty;
         lblGSRStatus.Text = "";
      }
   }
   catch
   {          }
}       private void txtRespiration_TextChanged(object sender, EventArgs e)
```

```
{    try
   {
      float respiration;

      respiration = float.Parse(txtRespiration.Text);

      if (respiration > float.Parse(txtMaxRespiration.Text))              {
         txtRespiration.BackColor = Color.Red;
         lblRespirationStatus.Text = "Respiration val. is over limit!!!";
      }
      if (respiration < float.Parse(txtMinRespiration.Text))  //max value 250 for
      {
         txtRespiration.BackColor = Color.Red;
         lblRespirationStatus.Text = "Respiration val. is under limit!!!";
      }
      else
      {
         txtRespiration.BackColor = Color.Empty;
         lblRespirationStatus.Text = "";
      }
   }
   catch
   {
   }
}
private void btnRapor_Click(object sender, EventArgs e)
```

```
        {       frmRapor frmRapor = new frmRapor();
            frmRapor.ShowDialog();
        }
    }
}
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;

namespace Fitness_Guide
{
    public partial class frmRapor : Form
    {
        SqlConnection myConn = new SqlConnection("Data Source=ACER-PC\\SQLEXPRESS; Initial Catalog=DB_EHEALTH; User id=SA; Password=perviz1; Integrated Security=True");

        public frmRapor()
        {       InitializeComponent();
```

```
}    public void ShowVitalSignHistory()
{
    listView1.Items.Clear();

    var selectcmd = new SqlCommand("S_VS_VITALSIGNS", myConn);
    selectcmd.CommandType = CommandType.StoredProcedure;
    selectcmd.Parameters.AddWithValue("@U_ID",
+frmUserInformations.cmbSelectedIndex);
    selectcmd.Parameters.AddWithValue("@VS_DATE1",
DateTime.Parse(cmbVS_DATE1.Text));
    selectcmd.Parameters.AddWithValue("@VS_DATE2",
DateTime.Parse(cmbVS_DATE2.Text));
    myConn.Open();
    SqlDataAdapter da = new SqlDataAdapter(selectcmd);
    SqlDataReader Reader = selectcmd.ExecuteReader();
    while (Reader.Read())
    {
        ListViewItem lv = new
ListViewItem(Reader.GetDateTime(0).ToShortDateString());

        lv.SubItems.Add(Reader.GetSqlString(1).ToString());
        lv.SubItems.Add(Reader.GetSqlString(2).ToString());
        lv.SubItems.Add(Reader.GetSqlString(3).ToString());
        lv.SubItems.Add(Reader.GetSqlString(4).ToString());
        lv.SubItems.Add(Reader.GetSqlString(5).ToString());
        lv.SubItems.Add(Reader.GetSqlString(6).ToString());
        lv.SubItems.Add(Reader.GetSqlString(7).ToString());
```

```
            lv.SubItems.Add(Reader.GetSqlString(8).ToString());
            lv.SubItems.Add(Reader.GetSqlString(9).ToString());
            listView1.Items.Add(lv);
        }
        Reader.Close();
        myConn.Close();
    }
     public void calculateAverage()
     {
        int counter1 = 0;
        int counter2 = 0;
        int counter3 = 0;
        int counter4 = 0;
        int counter5 = 0;
        int counter6 = 0;
        //  int counter7 = 0;
        int counter8 = 0;
        int counter9 = 0;

        double temperatureAverage = 0;
        double pulseAverage = 0;
        double diastolicAverage = 0;
        double systolicAverage = 0;
        double oxygenAverage = 0;
        double respirationAverage = 0;
        double glucometerAverage = 0;
```

```
double resistanceAverage = 0;
double conductivityAverage = 0;
int l = listView1.Items.Count;
for (int m = 1; m < 6; m++)
{               for (int i = 0; i < l; i++)
   {
      try
      {
         if (listView1.Items[i].SubItems[m].Text != "Null" &&
listView1.Items[i].SubItems[m].Text != "NULL" &&
listView1.Items[i].SubItems[m].Text != "" && listView1.Items[i].SubItems[m].Text
!= "0")
            {
               if (m == 1)
               {
                  temperatureAverage = temperatureAverage +
Convert.ToDouble(listView1.Items[i].SubItems[m].Text);
                  counter1++;
               }

               if (m == 2)
               {
                  pulseAverage = pulseAverage +
Convert.ToDouble(listView1.Items[i].SubItems[m].Text);
                  counter2++;
               }
               if (m == 3)
```

```
                {           diastolicAverage = diastolicAverage +
Convert.ToDouble(listView1.Items[i].SubItems[m].Text);
                    counter3++;
                }
                if (m == 4)
                {
                    systolicAverage = systolicAverage +
Convert.ToDouble(listView1.Items[i].SubItems[m].Text);
                    counter4++;
                }
                if (m == 5)
                {
                    oxygenAverage = oxygenAverage +
Convert.ToDouble(listView1.Items[i].SubItems[m].Text);
                    counter5++;
                }
                if (m == 6)
                {
                    respirationAverage = respirationAverage +
Convert.ToDouble(listView1.Items[i].SubItems[m].Text);
                    counter5++;
                }
                if (m == 7)
                {
                    glucometerAverage = glucometerAverage +
Convert.ToDouble(listView1.Items[i].SubItems[m].Text);
                    counter5++;
```

```
                    }       if (m == 8)
                    {
                        resistanceAverage = resistanceAverage +
Convert.ToDouble(listView1.Items[i].SubItems[m].Text);
                        counter5++;
                    }
                    if (m == 9)
                    {
                        conductivityAverage = conductivityAverage +
Convert.ToDouble(listView1.Items[i].SubItems[m].Text);
                        counter5++;
                    }
                }
            }
            catch
            { }
        }

    }
    lblAvgTemp.Text = Math.Round(temperatureAverage / counter1).ToString();
    lblAvgPulse.Text = Math.Round(pulseAverage / counter2).ToString();
    lblAvgDiastolic.Text = Math.Round(diastolicAverage / counter3).ToString();
    lblAvgSystolic.Text = Math.Round(systolicAverage / counter4).ToString();
    lblAvgOxygen.Text = Math.Round(oxygenAverage / counter5).ToString();
    lblAvgRespiration.Text = Math.Round(respirationAverage /
counter6).ToString();
```

```
        //lblAvgGlu.Text = Math.Round(glucometerAverage / counter7).ToString();
        lblAvgResistance.Text = Math.Round(resistanceAverage /
counter8).ToString();
        lblAvgConductivity.Text = Math.Round(conductivityAverage /
counter9).ToString();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        if (cmbVS_DATE1.Text == "" || cmbVS_DATE2.Text == "")
        {
            MessageBox.Show("Please choose the date");
        }
        else
        {
            ShowVitalSignHistory();

            calculateAverage();
            Color fcol;
            fcol = Color.Red;

            int t = listView1.Items.Count;

            for (int i = 0; i < t; i++)
            {
                try
```

```
        {      if ert.ToDouble(listView1.Items[i].SubItems[1].Text) > ,
            {
                listView1.Items[i].UseItemStyleForSubItems = false;
                listView1.Items[i].SubItems[1].ForeColor = Color.Yellow;
                listView1.Items[i].SubItems[1].BackColor = Color.Red;
            }
            if (Convert.ToDouble(listView1.Items[i].SubItems[2].Text) > 110)
            {
                listView1.Items[i].UseItemStyleForSubItems = false;
                listView1.Items[i].SubItems[2].BackColor = Color.Red;
                listView1.Items[i].SubItems[2].ForeColor = Color.Yellow;
            }
            if (Convert.ToDouble(listView1.Items[i].SubItems[5].Text) < 97)
            {
                listView1.Items[i].UseItemStyleForSubItems = false;
                listView1.Items[i].SubItems[5].ForeColor = Color.Yellow;
                listView1.Items[i].SubItems[5].BackColor = Color.Red;
            }
        }
        catch
        { }
      }
   }
}
private void frmVitalSignsFilter_Load(object sender, EventArgs e)
{
```

```
            cmbVS_DATE1.Items.Clear();
            cmbVS_DATE2.Items.Clear();
            listView1.Clear();
            listView1.GridLines = true;
            listView1.View = View.Details;
            listView1.Columns.Add("Date", 150);
            listView1.Columns.Add("Temperature", 120);
            listView1.Columns.Add("Pulse", 120);
            listView1.Columns.Add("Diastolic", 120);
            listView1.Columns.Add("Systolic", 120);
            listView1.Columns.Add("Oxygen", 120);
            listView1.Columns.Add("Respiration", 120);
            listView1.Columns.Add("Glucometer", 120);
            listView1.Columns.Add("Resistance", 120);
            listView1.Columns.Add("Conductivity", 120);
            var selectcmd = new SqlCommand("S_UI_USERINFORMATIONS",
myConn);
            selectcmd.CommandType = CommandType.StoredProcedure;
            selectcmd.Parameters.AddWithValue("@U_ID",
frmUserInformations.cmbSelectedIndex);
            myConn.Open();
            SqlDataAdapter da = new SqlDataAdapter(selectcmd);
            string birthDay = "";
            SqlDataReader reader = selectcmd.ExecuteReader();
            while (reader.Read())
            {
```

```
            birthDay = (string)reader["BirthDay"];
            lbfFullName.Text = (string)reader["Name"] + " " +
(string)reader["Surname"];
            lbfMobile.Text = (string)reader["Mobile"];
            lbfGender.Text = (bool)(reader["Gender"]) ? "Mr" : "Ms";
            pictureox1.Image = Image.FromFile((string)(reader["Photo"]));
            this.pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
        }
        string YearOfBirth = birthDay.Substring(6);

        DateTime now = DateTime.Today;
        lbfAge.Text = (now.Year - int.Parse(YearOfBirth)).ToString();
        //lbfAge.Text = "24";
        myConn.Close();
        FiterVitalSignDate();

    }
    public void FiterVitalSignDate()
    {
        string sql = "Select VS_DATE from T_VS_VITALSIGNS WHERE U_ID="
+ frmUserInformations.cmbSelectedIndex;
        myConn.Open();
        SqlCommand cmd = new SqlCommand(sql, myConn);
        SqlDataReader Reader = cmd.ExecuteReader();

        listView1.Items.Clear();
```

```
            string date = "";
            while (Reader.Read())
            {
                if (date != Reader.GetDateTime(0).ToString())
                {
                    date = Reader.GetDateTime(0).ToString();

cmbVS_DATE1.Items.Add(Reader.GetDateTime(0).ToShortDateString());

cmbVS_DATE2.Items.Add(Reader.GetDateTime(0).ToShortDateString());
                }


            }


            Reader.Close();
            myConn.Close();
        }


        public static int dayCounter;
        public static List<DateTime> lstOfDays = new List<DateTime>();
        private void button1_Click_1(object sender, EventArgs e)
        {
            foreach (String item in cmbVS_DATE1.Items)
            {
                lstOfDays.Add(Convert.ToDateTime(item));
            }
```

```
            dayCounter = 0;
            dayCounter = cmbVS_DATE1.Items.Count;
            frmDiagrams frmDiagrams = new frmDiagrams();
            frmDiagrams.ShowDialog();
        }
    }
}
```

ÖZGEÇMİŞ

Ad-Soyad : Parviz ABBASOV

Doğum Tarihi ve Yeri : Azerbaycan

E-Posta : parvizabbasov@hotmail.com

ÖĞRENİM DURUMU

- **Lisans** : 2012, Azerbaycan Devlet Neft Akademisi, Projelendirme Sistemlerinin Otomasyonu
- **Yüksek Lisans** : İstanbul Aydın Üniversitesi, Bilgisayar Mühendisliği