Available online at www.alphanumericjournal.com

alphanumeric journal

The Journal of Operations Research, Statistics, Econometrics and Management Information Systems

Volume 1, Issue 1, 2013

*2013.01.01.MIS.01*

# SAĞLIK SEKTÖRÜ İÇİN DÜŞÜK MALİYETLİ BİR MOBİL HASTA TAKİP SİSTEMİ ÖNERİSİ

Tunçhan CURA*

*İstanbul Üniversitesi, İşletme Fakültesi, Sayısal Yöntemler Anabilim Dalı, İstanbul*

**Özet**

Bu çalışmada özellikle orta büyüklükteki hastanelere yönelik bir mobil hasta takip sistemi önerilmiştir. Önerilen sistem oldukça düşük maliyetlidir ve böylece hastalara ilave bir külfet doğurmamaktadır. Bu sistem küçük bir uzman sistem de barındırmaktadır. Söz konusu sistemin kural tabanı ise bir ilişkisel veri tabanı yönetim sistemi kullanılarak oluşturulmuştur. Kural tabanının tamamıyla programcı ve programlama dilinden bağımsız olmasına özen gösterilmiştir. Bu esnek tasarım sayesinde program koduna hiç bir müdahalede bulunulmaksızın, kural tabanında güncellemeler yapılması mümkün olmaktadır.

*Anahtar Kelimeler: Mobil sağlık, Karar Destek Sistemleri*

*Jel Kodu: L86, I12*

**Abstract**

In this study, a mobile patient monitoring system is proposed especially for medium-sized hospitals. The proposed system is relatively low cost, and thus does not cause an additional burden on patients. This system contains a small expert system. The rule base of the system is built using a relational database management system. Rule base is kept completely independent of the programmer and the programming language. Thanks to this flexible design, it is possible to update rule base without an intervention to program code.

*Keywords: Mobil healthcare, Decission Support Systems.*

*Jel Code: L86, I12*

* *tunchan@istanbul.edu.tr*

## 1. GİRİŞ

Bu çalışma kişisel sağlık sektörüne yönelik bilişim teknolojilerinin mobil cihazlar üzerinde uygulanmasını ele almaktadır. Aslında hastaların sağlık durumlarının takip edilmesine yönelik yapılmış çalışmalar olsa da bunların bir çoğu maliyeti nispeten yüksek projelerdir. Örneğin Türkiye'de büyük bir hastane ile bir GSM Operatörü'nün birlikte sunduğu hizmet ile hastaların durumları takip edilmektedir. Fakat söz konusu uygulama hastalara ilave aylık ödeme yükü getirmektedir. Buna karşılık bu çalışmada önerilen yöntem bir hastane için sabit IP adresine sahip bir kişisel bilgisayar ile ortalama bir akıllı telefon maliyetinden öteye geçmemektedir. Böylece hastalardan da ilave ücret talep edilmesi gerekmemektedir.

Sağlık sektörü ve buna yönelik teknolojik çalışmalar son yılların hızla gelişen ve ilgi çeken konuları haline gelmiştir. Ayrıca mobil cihazlardaki hızlı gelişim ve bu aygıtların maliyetlerindeki düşüşler de hemen her alanda olduğu gibi sağlık sektörü uygulamalarında da araştırmacıların ve uygulamacıların dikkatini çekmeye başlamıştır (Katz ve Rice, 2009). Dünya genelinde artık çoğu sağlık cihazı, kişisel sağlık kayıtları ve tetkik yöntemleri taşınabilir hale gelmiştir (Katz ve Rice, 2009; Halteren vd., 2004).

Sağlık sektöründe hasta takibi son derece önemli bir konudur. Zira bazı hastaların sürekli kontrol altında tutulmaları gerekmektedir. Bu durum pek çok hastalık için hastaları sağlık kurumlarına bağımlı hale getirmiştir. Her ne kadar bir çok tahlil için evde bulundurulabilir düşük maliyetli cihazlar bulunsa da hasta durumunun cihazlardan elde edilen parametre değerlerine göre yorumlanması oldukça karmaşık olabilmektedir. Bu nedenle de hastalar sağlık kurumlarına bağımlı hale gelmekte ve böylece hareketlilikleri düşmektedir.

Yakın bir tarihe kadar çoğu mobil cihaz "aptal" (dump) makinalar olarak nitelenmekteydi. Bunlar tabiatıyla akıllı telefonların ilk sürümleriydi. Genellikle e-posta odaklıydılar ve bir çoğu kullanışlı dokunmatik ekrandan bile yoksundu. Genellikle özel kalemler ile ekranlarına yazılabilmekteydi (Charl ve LeRoux, 2011).

Fakat akıllı telefonlardaki son gelişmeler ve bunların sahip oldukları pek çok işlev yukarıda değinilmiş olan sorunlara da çözümler getirmiştir. iPhone ve Android telefonları gibi akıllı telefonlar geniş depolama kapasiteleri ve Wi-Fi veya 3G/4G ağlar ile gelişmiş bağlantı imkanları sağlamaktadırlar. Ayrıca bunlar klavye/sanal klavye, kamera, sayısal pusula, GPS alıcı ve akselerometre gibi arzu edilen işlevlerle gelmektedirler (Sun vd., 2010). Depolama kapasitesi ve bağlantı gücünün yanı sıra akıllı telefonlar kişisel bilgisayarlarla kıyaslanabilecek işlemci gücü ve iç bellek kapasitesine de sahiptirler (Weng vd., 2012).

Bir kaç yıl içerisinde ABD'deki İnternet kullanıcılarının yarıdan fazlasının İnternet erişimi için akıllı telefonları ve tablet bilgisayarları (mobil ağ odaklı bilgisayar) kullanacakları beklenmektedir. 2010'da 84 milyon Amerikalı İnternet'e mobil aygıtlar aracılığıyla erişmiştir ve bu rakamın 2014'e kadar iki katına çıkması beklenmektedir. Ayrıca bir araştırma işletmesi olan Gartner, kısa zaman içerisinde mobil telefonların kişisel bilgisayarların yerini alarak, insanların İnternet erişiminde en çok kullanacakları yöntem olacağını tahmin etmiştir. Günümüzde, İnternet'te gerçekleşen tüm aramanın yüzde beşini mobil cihazlar oluşturmaktadır. Fakat bu oranın 2016'da yüzde 23,5 olması beklenmektedir (Laudon ve Laudon, 2011).

## 2. ANDROID UYGULAMASI

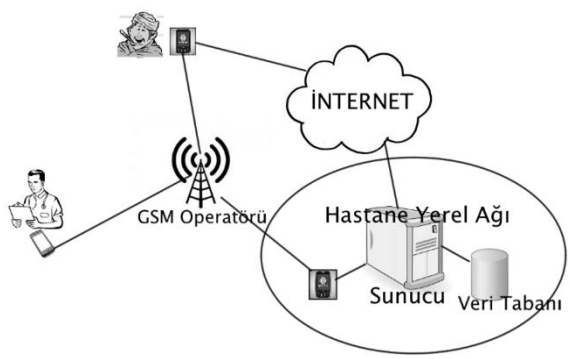
Bu çalışmada geliştirilen mobil hasta takip sisteminin mobil bileşenleri Android İşletim Sistemi üzerinde çalışmaktadır. Android, mobil cihazlar için Google tarafından geliştirilmiş bir işletim sistemidir (Android Developers, 2013). Bu işletim sisteminin tercih edilmesinin sebebi mobil cihaz pazarında en fazla kullanılan işletim sistemi olmasıdır. Buna göre pazarın yüzde 72.4'ünü Android elinde tutarken ikinci sırada gelen Apple'ın geliştirdiği iOS yüzde 13.9'luk paya sahiptir (Gartner Inc., 2013). En bilinen üretici markaların Android destekli çok çeşitli ürünleri bulunmaktadır.

Bunun da ötesinde kısım 2.2'de ele alınacak olan sunucu tarafındaki mobil uygulama, telefona gelen SMS'leri dinlemektedir. Her işletim sistemi gelen SMS'lerin dinlenmesine izin vermemektedir. Bu nedenle her ne kadar hasta tarafı için farklı işletim sistemlerine göre uygulama geliştirmek mümkün olsa da bu çalışmada önerildiği gibi sunucu tarafında bir mobil cihaz bulundurulması halinde, cihazda Android gibi gelen SMS'leri dinlemeye izin veren bir işletim sistemi koşturulması zorunludur.

Android Telefonlar'a uygulama geliştirmek için Google, İnternet'ten ücretsiz indirilebilen Android Yazılım Geliştirici Paketi'ni (Android SDK) sunmuştur. Android uygulaması geliştirmek için kullanılan dil Java'dır (Oracle, 2013). Uygulama geliştirenlerin neredeyse tamamı Bütünleşik Geliştirme Ortamı (IDE) olarak Eclipse'i (The Eclipse Corparation, 2013) kullanmaktadır. Bu IDE'ye Android Geliştirme Araçları (ADT) ve Android SDK da eklenmektedir. Böylece Android Telefonlar için rahatlıkla uygulama geliştirilebilmektedir (Weng vd., 2012).

Bu çalışma için tasarlanan sistem üç ana bileşenden meydana gelmektedir. Bunlardan ilki hastanın mobil cihazına yüklenen Android uygulamasıdır. Bu uygulama hastanın kendi yaptığı tahlil sonuçlarını girdiği ve bunları hastane sunucusuna gönderdiği yalın bir ekrandan ibarettir. İkincisi hastanenin yerel ağı ile hastane sunucusuna bağlı mobil cihaz üzerinde bulunan başka bir Android uygulamadır. Bu uygulama, hastanın tahlil sonuçlarını SMS ile göndermesi halinde değerleri alıp yerel ağ üzerinden sunucuya göndermektedir. Hasta, sonuçları İnternet aracılığıyla doğrudan sunucuya gönderme imkanına da sahiptir. Sunucudan yerel ağ aracılığıyla gelen mesajı hastanın doktoruna SMS ile göndermek bu uygulamanın ana görevidir. Üçüncü bileşen ise sunucudur. Sunucu bir kişisel bilgisayar olabilir. Sunucunun üzerindeki bir Java Servlet uygulaması hastadan gelen tahlil sonuçlarını değerlendirir ve hastanın doktoruna SMS ile değerlendirme sonucunu gönderir. Sunucu doğrudan SMS gönderemeyeceği için bunu yerel ağ ile kendisine bağlı Android uygulamasının yapmasını sağlar. Tüm yapı Şekil 1'de temsil edilmiştir. Önerilen uygulamanın çalışma biçimi özetle aşağıdaki gibidir:

1. Hasta tahlilini yanında bulundurduğu cihaz(lar) ile kendisi yapar.
2. Hasta kullandığı cihaz(lar)dan elde ettiği parametre değerlerini mobil cihazında yüklü uygulamaya girer.
3. Bu değerleri bağlantısı varsa İnternet, yoksa SMS ile sunucuya gönderir.
4. Değerler SMS ile gönderilirse; hastane sunucusuyla aynı yerel ağ içerisinde bulunan başka bir mobil cihazdaki uygulama, SMS mesajıyla gelen verileri sunucuda yer alan Java Servlet uygulamasına gönderir.
5. Sunucu gelen verileri bir uzman sistem içerisinde değerlendirir ve kural tabanında yer alan kurallara göre bir sonuç çıkarır. Bu sonuç şüphesiz ki kesin karar için yeterli olmayacaktır. Dolayısıyla kesin karar için hastanın doktoruna sonuç SMS ile gönderilir. Doktor böylece gerekiyorsa hastasıyla görüşerek kesin kararı verir.




Şekil 1. Hasta takibi uygulamasının ana unsurları

Şunu da belirtmek gerekir ki sunucunun SMS alması ve göndermesi için tek yol bunu bir mobil telefona bağlamak değildir. Alternatif yöntemlerden en sık karşılaşılanları sunucuya GSM modem takmak, bir Mobil Ağ Geçidi'nden hizmet almak veya sunucunun doğrudan Sinyalleşme Sistemi 7 (SS7) şebekesine bağlanmasıdır. Her bir alternatifin kendine özgü avantaj ve dezavantajları vardır. Bu durumda

takip edilen hasta sayısı da göz önünde bulundurularak bir tercih yapılmalıdır.

Sunucunun bir mobil cihaza bağlanması sonucunda karşılaşılan en önemli sorun cihaza bağlı olarak değişen şarj edilme ve nadiren de olsa kapatılıp tekrar açılma gereksinimleridir. Ayrıca çok yoğun mesajlaşmalarda bazı sorunlar da olabilir. Fakat bu çalışma orta büyüklükteki hastaneleri hedef aldığından söz konusu sorunun göz ardı edilebileceği düşünülmüştür. Daha fazla sayıda hastası olan hastaneler için sunucuyu SS7 şebekesine veya bir Mobil Ağ Geçidi'ne bağlamak düşünülebilir. Fakat özellikle SS7 şebekesinin kullanılması halinde maliyetler önemli ölçüde artacaktır.

### *2.1. Hasta Mobil Uygulaması*

Daha önce de değinildiği gibi hastaların sağlık kurumlarından bağımsız, kendi başlarına tahlillerini yapabilmesi bu çalışmanın ana hedefidir. Bu nedenle hastanın bazı cihazların yardımıyla yaptığı tahlillerden elde ettiği sonuçları,


Şekil 2. Hasta takip uygulaması

Veri paylaşımında kullanılan bazı standart biçimlendirme yöntemleri vardır. Bunların başında Uzatılmış İşaretleme Dili (XML) gelse de gerek kolay okunabilir oluşu, gerekse de nesneye dayalı dillerle daha uyumlu kullanılabilmesi sebepleriyle JavaScript Nesne Nütasyonu (JSON) son zamanlarda yaygın hale gelmiştir. Bu standarda göre küme parantezi içine nesne, köşeli parantez içine ise dizler yerleştirilir. Her bir özellik virgülle ayrılır. Nesnelerin dizi olmasının mümkün olduğu gibi, bir nesne özelliğinin de başka bir nesne olması veya dizi olması mümkündür. Özelliklerin isimleri ile değerleri arasına ":" işareti konularak birbirinden ayrılırlar. Ayrıca gerek özellik ismi gerekse de taşıdığı değer tırnak içerisinde yazılır. Aşağıda JSON biçimindeki veri örneği yer almaktadır:

{"BA":"5","EO":"3","hastano":"1","MO":"3"}

Buna göre 1 numaralı hastanın Bazofil Yüzdesi (BA) 5, Eozinofil Yüzdesi (EO) 3 ve Monosit Yüzdesi (MO) de 3 değerlerine sahiptir. Bu çalışmanın kapsamına tıbbi tahlil değerlerinin yorumlanması girmediğinden, daha detaylı tahlil parametreleri ele alınmamış ve sınamalarda bu kadarıyla yetinilmiştir.

Google'ın geliştirdiği ve Android uygulamalarında yararlanılabilen Gson kütüphanesi, verinin yukarıdaki biçime dönüştürülmesi veya bu biçimindeki verinin ayrıştırılması için kullanılabilir (Google Project Hosting, 2013).

Kullanıcı verileri İnternet ile göndermek isterse dokunduğu düğme tetiklenecek ve Şekil 3'te gösterilen kod çalıştırılacaktır. Aksi halde Şekil 4'te yer alan kod yürütülmektedir. uyari(String arg) metodu argümanında yer alan mesajı kullanıcıya göstermek için kullanılmaktadır.

```
((Button) findViewById(R.id.button1))
            .setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        String jSonMesaj = bilgiler();
        Handler mYonetici = new Handler() {
            public void handleMessage
            (android.os.Message msg) {
                String cevap = (String) msg.obj;
                uyari(cevap);
            };
        };
        Gonderici gonderici = new Gonderici(jSonMesaj,
                                            mYonetici);
        new Thread(gonderici).start();
    }
});
```

Şekil 3. Verilerin sunucuya İnternet ile gönderilmesi

Android uygulamalarının ana iş parçacığında (main thread) ağ bağlantısı kurulması engellenmiştir. Bu nedenle Runnable ara yüzünü uygulayan Gonderici sınıfından bir nesne ile ağ bağlantısı kurulmaktadır. TELNO sabit değişkeninde ise sunucuya bağlı olan mobil telefonun numarası bulunmaktadır.

Gson kütüphanesi sayesinde Şekil 5'te görüldüğü gibi kullanıcının girdiği veriler kolaylıkla JSON biçimine dönüştürülebilmektedir.

```
((Button) findViewById(R.id.button2))
            .setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        String sms = bilgiler();
        SmsManager smsyonetici =
                        SmsManager.getDefault();
        if (sms.length() <= 160)
            smsyonetici
            .sendTextMessage(TELNO, null, sms, null, null);
        else {
            ArrayList<String> cokluMesaj = smsyonetici
                                        .divideMessage(sms);
            smsyonetici.sendMultipartTextMessage(
                        TELNO, null, cokluMesaj, null, null);
        }
    }
});
```

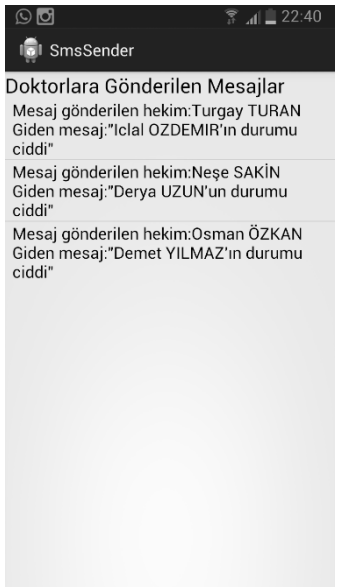
Şekil 4.Verilerin sunucuya SMS ile gönderilmesi

```
private String bilgiler() {
    Map<String, String> mesaj =
                            new HashMap<String, String>();
    mesaj.put("hastano", numara.getText().toString());
    mesaj.put("EO", EO.getText().toString());
    mesaj.put("MO", MO.getText().toString());
    mesaj.put("BA", BO.getText().toString());
    return new Gson().toJson(mesaj);
}
```

Şekil 5. Verilerin JSON biçimine dönüştürülmesi

### *2.2. Sunucu Mobil Uygulaması*

Hasta takip sisteminin bir mobil uygulaması daha vardır ve sunucu tarafındaki mobil cihaz üzerinde çalışır. Şekil 6'da görülen bu uygulamanın görevi hastalardan gelen SMS'leri ve sunucudan gelen mesajları dinlemektir. Hastalardan gelen SMS'leri okuyup sunucuya yerel ağ ile iletir.




Şekil 6. Sunucu tarafındaki mobil uygulama

Sunucunun hastaların durumunu bir uzman sistem yardımıyla değerlendirmesi ile elde edilen sonucun hastanın doktoruna iletilmesi için bu uygulamanın sunucudan gelen mesajları da dinlemesi gerekmektedir. Bu nedenle uygulama ana iş parçacığının yanı sıra söz konusu dinleme işlemlerini gerçekleştirmek için iki iş parçacığı daha barındırır. Bunlardan ilki Şekil 7'de görüldüğü gibi 6666 numaralı bağlantı noktasını (port) dinleyen Dinleyici sınıfından bir nesnedir. Bu nesne gelen mesajları Handler sınıfından mYonetici nesnesine göndermektedir.

```
Handler mYonetici = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        Dinleyici.jsonMesaj mesaj =
                            (Dinleyici.jsonMesaj) msg.obj;
        SmsManager smsyonetici = SmsManager.getDefault();
        String sms = "Sayın " + mesaj.doktor + " " +
mesaj.mesaj;
        if (sms.length() <= 160)
            smsyonetici.sendTextMessage(mesaj.telno,
                                null, sms, null, null);
        else {
            ArrayList<String> cokluMesaj =
                        smsyonetici.divideMessage(sms);
            smsyonetici.sendMultipartTextMessage(
                    mesaj.telno, null, cokluMesaj, null, null);
        }
```

```
        ogeListesi.add("Mesaj gönderilen hekim:" +
                mesaj.doktor + "\nGiden mesaj:\"" +
                                mesaj.mesaj + "\"");
        adaptor.notifyDataSetChanged();
    }
};
dinleyici = new Thread(new Dinleyici(6666, mYonetici));
dinleyici.start();
```

Şekil 7. Yerel ağdan gelen mesajların dinlenmesi

Böylece doktorlara hastalarıyla ilgili mesajlar iletilmektedir. ogeListesi ve adaptor nesneleri, ListView sınıfından bir nesnede doktorlara giden mesajları ekranda listelenmek için kullanılmaktadır.

Gelen JSON biçimindeki bir SMS, küme parantezi ile başlayacağından yalnızca bunların dikkate alınması gerekmektedir. Şekil 8'de görüldüğü gibi gelen mesaj eğer JSON biçiminde ise yerel ağ üzerinden sunucuya yönlendirilmektedir. Ardından bu mesajın "gelen kutusu" na eklenmesini önlemek için SMS dinleme işlemini gerçekleştiren BroadcastReceiver sınıfından nesnenin abortBroadcast() metodundan yararlanılmıştır.

```
Object[] pdus = (Object[]) bundle.get("pdus");
msgs = new SmsMessage[pdus.length];
String data = "";
for (int i = 0; i < msgs.length; i++) {
    msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
    data += msgs[i].getMessageBody().toString();
}
if (data.charAt(0) == '{') {
    Handler mYonetici = new Handler() {
        public void handleMessage(android.os.Message msg) {
            String cevap = (String) msg.obj;
            Toast.makeText(ctx, cevap, 10).show();
        };
    };
    gonderici con = new gonderici(data, mYonetici);
    new Thread(con).start();
    abortBroadcast();
}
```

Şekil 8. Gelen SMS'lerin dinlenmesi

Şekilde görülen Android uygulamasına girerek, İnternet üzerinden veya SMS ile hastane sunucusuna gönderecektir.


Şekil 2. Hasta takip uygulaması

Veri paylaşımında kullanılan bazı standart biçimlendirme yöntemleri vardır. Bunların başında Uzatılmış İşaretleme Dili (XML) gelse de gerek kolay okunabilir oluşu, gerekse de nesneye dayalı dillerle daha uyumlu kullanılabilmesi sebepleriyle JavaScript Nesne Nütasyonu (JSON) son zamanlarda yaygın hale gelmiştir. Bu standarda göre küme parantezi içine nesne, köşeli parantez içine ise dizler yerleştirilir. Her bir özellik virgülle ayrılır. Nesnelerin dizi olmasının mümkün olduğu gibi, bir nesne özelliğinin de başka bir nesne olması veya dizi olması mümkündür. Özelliklerin isimleri ile değerleri arasına ":" işareti konularak birbirinden ayrılırlar. Ayrıca gerek özellik ismi gerekse de taşıdığı değer tırnak içerisinde yazılır. Aşağıda JSON biçimindeki veri örneği yer almaktadır:

{"BA":"5","EO":"3","hastano":"1","MO":"3"}

Buna göre 1 numaralı hastanın Bazofil Yüzdesi (BA) 5, Eozinofil Yüzdesi (EO) 3 ve Monosit Yüzdesi (MO) de 3 değerlerine sahiptir. Bu çalışmanın kapsamına tıbbi tahlil değerlerinin yorumlanması girmediğinden, daha detaylı tahlil parametreleri ele alınmamış ve sınamalarda bu kadarıyla yetinilmiştir.

Google'ın geliştirdiği ve Android uygulamalarında yararlanılabilen Gson kütüphanesi, verinin yukarıdaki

biçime dönüştürülmesi veya bu biçimindeki verinin ayrıştırılması için kullanılabilir (Google Project Hosting, 2013).

Kullanıcı verileri İnternet ile göndermek isterse dokunduğu düğme tetiklenecek ve Şekil 3'te gösterilen kod çalıştırılacaktır. Aksi halde Şekil 4'te yer alan kod yürütülmektedir. uyari(String arg) metodu argümanında yer alan mesajı kullanıcıya göstermek için kullanılmaktadır.

```
((Button) findViewById(R.id.button1))
            .setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        String jSonMesaj = bilgiler();
        Handler mYonetici = new Handler() {
            public void handleMessage
            (android.os.Message msg) {
                String cevap = (String) msg.obj;
                uyari(cevap);
            };
        };
        Gonderici gonderici = new Gonderici(jSonMesaj,
                                            mYonetici);
        new Thread(gonderici).start();
    }
});
```

Şekil 3. Verilerin sunucuya İnternet ile gönderilmesi

Android uygulamalarının ana iş parçacığında (main thread) ağ bağlantısı kurulması engellenmiştir. Bu nedenle Runnable ara yüzünü uygulayan Gonderici sınıfından bir nesne ile ağ bağlantısı kurulmaktadır. TELNO sabit değişkeninde ise sunucuya bağlı olan mobil telefonun numarası bulunmaktadır.

Gson kütüphanesi sayesinde Şekil 5'te görüldüğü gibi kullanıcının girdiği veriler kolaylıkla JSON biçimine dönüştürülebilmektedir.

```
((Button) findViewById(R.id.button2))
            .setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        String sms = bilgiler();
        SmsManager smsyonetici =
                    SmsManager.getDefault();
        if (sms.length() <= 160)
            smsyonetici
            .sendTextMessage(TELNO, null, sms, null, null);
        else {
            ArrayList<String> cokluMesaj = smsyonetici
                                        .divideMessage(sms);
            smsyonetici.sendMultipartTextMessage(
                    TELNO, null, cokluMesaj, null, null);
        }
    }
});
```

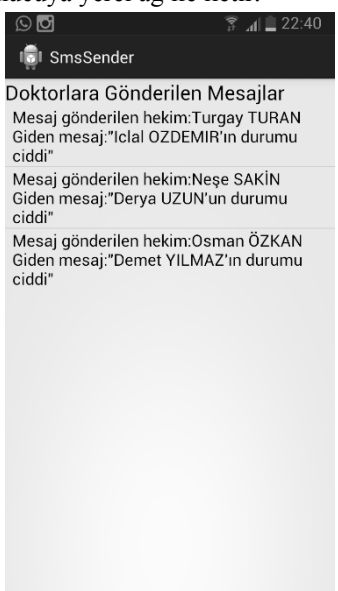
Şekil 4.Verilerin sunucuya SMS ile gönderilmesi

```
private String bilgiler() {
    Map<String, String> mesaj =
                        new HashMap<String, String>();
    mesaj.put("hastano", numara.getText().toString());
    mesaj.put("EO", EO.getText().toString());
    mesaj.put("MO", MO.getText().toString());
    mesaj.put("BA", BO.getText().toString());
    return new Gson().toJson(mesaj);
}
```

Şekil 5. Verilerin JSON biçimine dönüştürülmesi

### *2.3. Sunucu Mobil Uygulaması*

Hasta takip sisteminin bir mobil uygulaması daha vardır ve sunucu tarafındaki mobil cihaz üzerinde çalışır. Şekil 6'da görülen bu uygulamanın görevi hastalardan gelen SMS'leri ve sunucudan gelen mesajları dinlemektir. Hastalardan gelen SMS'leri okuyup sunucuya yerel ağ ile iletir.




Şekil 6. Sunucu tarafındaki mobil uygulama

Sunucunun hastaların durumunu bir uzman sistem yardımıyla değerlendirmesi ile elde edilen sonucun hastanın doktoruna iletilmesi için bu uygulamanın

sunucudan gelen mesajları da dinlemesi gerekmektedir. Bu nedenle uygulama ana iş parçacığının yanı sıra söz konusu dinleme işlemlerini gerçekleştirmek için iki iş parçacığı daha barındırır. Bunlardan ilki Şekil 7'de görüldüğü gibi 6666 numaralı bağlantı noktasını (port) dinleyen Dinleyici sınıfından bir nesnedir. Bu nesne gelen mesajları Handler sınıfından mYonetici nesnesine göndermektedir.

```
Handler mYonetici = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        Dinleyici.jsonMesaj mesaj =
                            (Dinleyici.jsonMesaj) msg.obj;
        SmsManager smsyonetici = SmsManager.getDefault();
        String sms = "Sayın " + mesaj.doktor + " " +
mesaj.mesaj;
        if (sms.length() <= 160)
            smsyonetici.sendTextMessage(mesaj.telno,
                                    null, sms, null, null);
        else {
            ArrayList<String> cokluMesaj =
                        smsyonetici.divideMessage(sms);
            smsyonetici.sendMultipartTextMessage(
                    mesaj.telno, null, cokluMesaj, null, null);
        }
        ogeListesi.add("Mesaj gönderilen hekim:" +
                mesaj.doktor + "\nGiden mesaj:\"" +
                                    mesaj.mesaj + "\"");
        adaptor.notifyDataSetChanged();
    }
};
dinleyici = new Thread(new Dinleyici(6666, mYonetici));
dinleyici.start();
```

Şekil 7. Yerel ağdan gelen mesajların dinlenmesi

Böylece doktorlara hastalarıyla ilgili mesajlar iletilmektedir. ogeListesi ve adaptor nesneleri, ListView sınıfından bir nesnede doktorlara giden mesajları ekranda listelenmek için kullanılmaktadır.

Gelen JSON biçimindeki bir SMS, küme parantezi ile başlayacağından yalnızca bunların dikkate alınması gerekmektedir. Şekil 8'de görüldüğü gibi gelen mesaj eğer JSON biçiminde ise yerel ağ üzerinden sunucuya yönlendirilmektedir. Ardından bu mesajın "gelen kutusu" na eklenmesini önlemek için SMS dinleme işlemini gerçekleştiren BroadcastReceiver sınıfından nesnenin abortBroadcast() metodundan yararlanılmıştır.

```
Object[] pdus = (Object[]) bundle.get("pdus");
msgs = new SmsMessage[pdus.length];
String data = "";
for (int i = 0; i < msgs.length; i++) {
    msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
    data += msgs[i].getMessageBody().toString();
}
if (data.charAt(0) == '{') {
    Handler mYonetici = new Handler() {
        public void handleMessage(android.os.Message msg) {
            String cevap = (String) msg.obj;
            Toast.makeText(ctx, cevap, 10).show();
        };
    };
    gonderici con = new gonderici(data, mYonetici);
    new Thread(con).start();
    abortBroadcast();
}
```

Şekil 8. Gelen SMS'lerin dinlenmesi

## 3. HASTA TAKİP SİSTEMİNİN SUNUCU TARAFI

Bu çalışmada önerilen sistemin sunucu tarafı iki önemli alt bileşenden oluşmaktadır. Bunlardan birincisi İnternet üzerinden iletilen mesajları dinleyen ve yanıtlayan Java Servlet uygulamasıyken diğeri bir veri tabanıdır. Bu veri tabanı hasta ve doktorlarla ilgili verilerin yanı sıra daha önce değinildiği gibi uygulamada yer alan uzman sistem için bir de kural tabanı (rule base) barındırmaktadır.

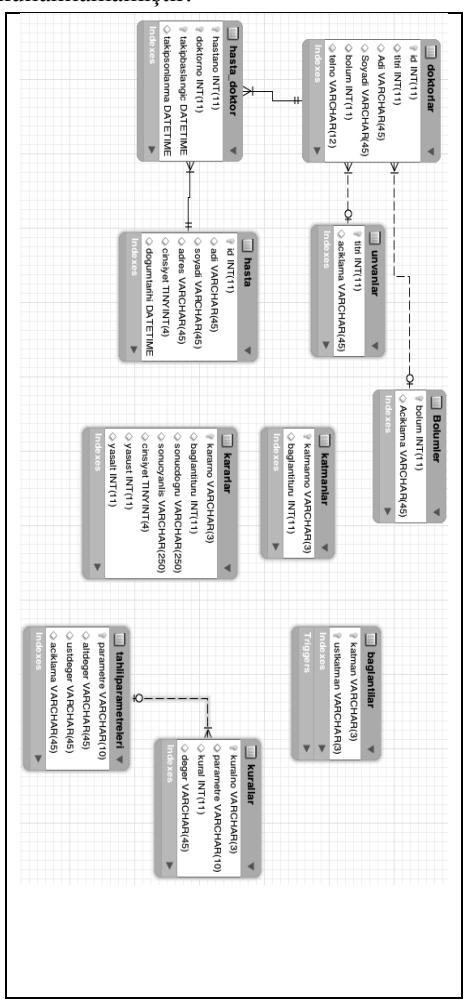
Tabiatıyla Servlet uygulamasının yürütülebilmesi için bir sunucu yazılımı gereklidir. Bunun için açık kaynak bir yazılım olan Apachi Tomcat 7 (The Apache Foundation, 2013) kullanılmıştır. Apachi Tomcat varsayılan olarak 8080 numaralı bağlantı noktasını dinler. Bunun değiştirilmesi mümkündür. Nitekim bu çalışmada kullanılan sunucu cihazda çakışmaları önlemek için 8081 olarak değiştirilmiştir.

Aslında yalnızca sunucu ile haberleşmek için Servlet uygulaması gerekli değildir. Fakat her mobil işletim sistemi sunduğu Uygulama Programlama Ara Yüzleri (API) sayesinde Birörnek Kaynak Konumlayıcı (URL) kullanarak sorunsuz ağ bağlantıları yapmayı oldukça kolaylaştırmaktadır.

Sunucuda hasta ve doktor verilerini bulunduran, aynı zamanda önerilen uzman sistem için kural tabanını barındıran veri tabanı yönetim sistemi için MySQL tercih edilmiştir. Bu veri tabanı yönetim

sistemi de ücretsiz olarak temin edilebilmektedir (Oracle Corporation and/or its affiliates, 2013).

Bu çalışmada tasarlanan ilişkisel veri tabanı 10 adet tablodan meydana gelmektedir. Bu tabloların beşi hasta ve doktorlarla ilgili kayıtları tutmak için oluşturulmuşken, kalan beş tablo ise kural tabanı olarak kullanılmaktadır. Varlık-ilişki diyagramı'da verilmiş olan veri tabanının dikkat edilirse yalnızca veri saklamak için kullanılan ilk beş tablosunda dışsal anahtar kullanılmışken, kural tabanında bir tablo haricinde kullanılamamıştır.




Şekil 9. Veri tabanı varlik-ilişki diyagramı

Bunun sebebi burada önerilmiş olan uzman sistem için tasarlanan kural tabanının yapısıdır. İyi tasarlanmış bir uzman sistemde kural tabanının kodlama gerektirmeden düzenlenebilmesi gerekir. Böylece program koduna dokunulmaksızın yalnızca kural tabanında yapılacak düzenlemelerle yeni kurallar eklenebilir, bazıları çıkarılabilir veya güncellenebilir. Ancak ilişkisel modele göre tasarlanmış bir veri tabanında bahsedilen özelliklere sahip bir kural tabanı oluşturmak da güçleşmektedir. Bu nedenle bu çalışmada ilişkisel modelin temel bazı kuralları esnetilerek dışsal anahtar–birincil anahtar referans bütünlük kontrolü, tetikleyiciler (trigger) yardımıyla sağlanmıştır. Bunun nedeni önerilen tasarımda baglantilar tablosunun kurallar, katmanlar ve kararlar tabloları arasında bağlantı sağlayan bir tablo oluşudur. Fakat söz konusu tabloda yer alan katman ve ustkatman kolonları kuralno, katmanno ve kararno kolonlarını referans almaktadır. Başka bir ifadeyle referans olan bir kolondaki değer bu iki kolona da verilebilmektedir. Ayrıca referans olan kolonların sayısı da birden fazladır. Dışsal anahtar aslında bir tür kısıt (constraint) olduğundan, tüm bu referans kolonların dışsal anahtar olarak gösterilmesi de veri girişinde istenilenden fazla bir kısıtlamaya neden olarak işlerin doğru yürütülmesini zorlaştırabilir. Ancak veri bütünlüğünün sağlanması son derece önemli bir husus olduğundan yine veri tabanında bu sorunun çözülmesi gerekir. Böyle çok nadir karşılaşılabilecek karmaşık referans durumlarına karşılık MySQL gibi bir çok veri tabanı yönetim sistemi tetikleyiciler ve fonksiyonlardan yararlanır. Şekil 10'da yer alan katman_kontrol(katman, ustkatman) fonksiyonu bütünlük kontrolünü yapmakta ve hata durumunda bir, aksi halde ise sıfır değerini geri döndürmektedir. Böylece yeni kayıt girişi için tanımlanmış bir tetikleyicide aşağıdaki gibi bir kontrol mekanizması rahatlıkla kurulabilir:

```
IF (katman_kontrol(NEW.katman,
                   NEW.ustkatman) = 1) THEN
    SIGNAL SQLSTATE '45000' SET
                MESSAGE_TEXT = 'Katman
                    baglanti hatasi
olustu';
END IF;
```

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION
    `katman_kontrol`(katman VARCHAR(3), ustkatman
    VARCHAR(3)) RETURNS tinyint(4)
BEGIN
DECLARE katmanno1 INT;
DECLARE katmanno2 INT;
DECLARE ustkatmanno1 INT;
DECLARE ustkatmanno2 INT;
SET katmanno1 = (SELECT count(*) FROM katmanlar
                        WHERE katmanno = katman);
SET katmanno2 = (SELECT count(*) FROM kurallar
                        WHERE kuralno = katman);
SET ustkatmanno1 = (SELECT count(*) FROM
          katmanlar WHERE katmanno = ustkatman);
SET ustkatmanno2 = (SELECT count(*) FROM kararlar
                        WHERE kararno = ustkatman);
IF (katmanno1 = 0 AND katmanno2 = 0) OR
     (ustkatmanno1 = 0 AND ustkatmanno2 = 0) THEN
   RETURN 1;
ELSE
   RETURN 0;
END IF;
END
```

Şekil 9. Kural tabanında veri bütünlüğünün sağlanması

Görüldüğü gibi baglanti tablosuna girilecek yeni bir kaydın karşılığı eğer yoksa katman_kontrol() fonksiyonu bir değerini döndürecek ve veri tabanı hata mesajı üreterek eylemi kesecektir.

Bu açıklamalardan sonra kural tabanında yer alan kurallardan yapılacak çıkarımların nasıl tespit edileceğine değinmek gerekir. Karar çıkarım mekanizması bu çalışmada üst üste yerleştirilmiş katmalar biçiminde düşünülmüştür. Dolayısıyla her karar için ortaya bir ağaç yapısı çıkmaktadır. Hiyerarşinin en altında temel kurallar yer alır. Örneğin BA parametresinin 0.973 değerinden küçük olması bir kuraldır. Kural tabanında eşitlik ve eşitsizlikleri temsil etmek için Tablo 1'de verilmiş olan değerler kullanılmıştır. Böylece bir numaralı kural, kurallar tablosunda (1, BA, -2, 0.973) biçiminde saklanmaktadır.

Tablo 1. Kuralların Temsili Değerleri

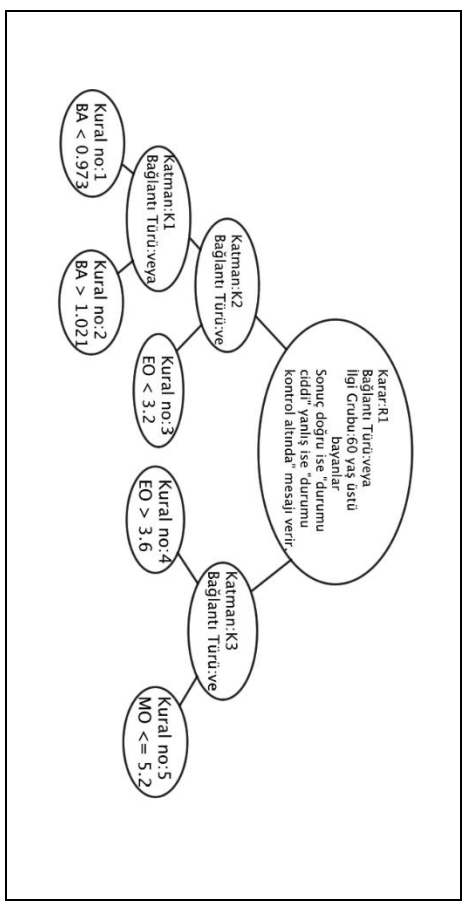
| Kural | Temsili Değer |
|---|---|
| Küçüktür(<) | -2 |
| Küçükeşittir(≤) | -1 |
| Eşittir(=) | 0 |
| Büyükeşittir(≥) | 1 |
| Büyüktür | 2 |

Yukarıda değinilen hiyerarşinin en alt basamağı kurallar ve bunun üzerinde de katmanlar yer almaktadır. Hiyerarşide yer alacak katman sayısı önermenin karmaşıklığıyla alakalıdır. Yeri gelmişken vurgulamak gerekir ki bir kural, bir katman ve hatta kararın kendisi birer önermedir. Kararı meydana getiren alt önermeler katmanlardan ve kurallardan meydana gelir. Bir kural tek başına önerme iken, katman tek başına önerme değildir. Zira katmanların amacı birden fazla kural ya da katmanı bir araya getirmektir. Tabiatıyla bu da "ve" veya "veya" mantık operatörleriyle sağlanır. Dolayısıyla her katmanın kuralları ve/veya katmanları bağlamak için bir baglantituru vardır. Bu değer veya için sıfır iken ve için birdir. Daha önce değinildiği gibi baglantilar tablosu hiyerarşiyi temsil etmek için kullanılmaktadır. Başka bir ifadeyle bu tabloya girilen (1, K1) ve (2, K1) satırları birinci ve ikinci kuralları K1 katmanına bağlamaktadır. Zira ustkatman kolonunda K1 yer almaktadır. Ayrıca K1 katmanının baglantituru sıfır olduğundan, birinci ve ikinci kuralların ikisinin birden sağlanması (doğru olması) durumunda bu katmanın (önermenin) doğru, aksi halde yanlış olacağı anlamını taşır. Kararlar ise katmanlara çok benzer. Ancak hiyerarşinin en üstünde yer alırlar. Dolayısıyla her bir kararın önermenin doğru veya yanlış olmasına göre bir sonucu vardır. Bu çalışmanın kapsamı dışında olduğundan tıbbi terimlere odaklanılmamış ve oldukça yalın terimler kullanılmıştır. Bu nedenle sınama amaçlı örneklerde karmaşık tıbbi ifadeler geçmemektedir. Örneğin R1 kararı önerme doğruysa sonucdogru kolonuna girilmiş olan mesajı, yanlışsa sonucyanlis kolonuna girilmiş olan mesajı verecektir. Eğer bu kolonlardan herhangi birisine değer girilmemiş ise doktora karşılık gelen durumda mesaj gönderilmesine gerek olmadığı anlaşılır. Zira uzman sistem bir sonuç üretmemektedir.

Tablo 2. Karar Çıkarım Hiyerarşisi

| Katman | Ustkatman |
|---|---|
| 1 | K1 |
| 2 | K1 |
| 3 | K2 |
| 4 | K3 |
| 5 | K3 |
| K1 | K2 |
| K2 | R1 |
| K3 | R1 |

Tablo 2'de verilmiş olan baglanti tablosuna göre tek bir karar vardır. Söz konusu karar R1 olarak kodlanmıştır. Zira kararlar tablosunda yalnızca bunun karşılığı bulunmaktadır. Bu tablonun karşılık geldiği hiyerarşi ise




Şekil 11. Karar çıkarım mekanizması

Şekil 11'de gösterilmektedir. Bu hiyerarşi matematiksel nütasyonla aşağıdaki gibi temsil edilebilir:

$$sonuc\begin{cases} R1 & cinsiyet\ 0 \wedge \\ \quad doğum\,tarihi < 01.01.1954; & \\ \varnothing & değilse; \end{cases}$$

$$R1\begin{cases} durumu\,ciddi\left[\begin{pmatrix} BA < 0.973 \vee \\ BA > 1.021 \end{pmatrix} \wedge EO < 3.2\right] \\ \vee (EO > 3.6 \wedge MO \le 5.2); \\ durumu\,kontrol\,altında\,değilse; \end{cases}$$

Görüldüğü gibi bu son derece esnek bir tasarımdır. Böylece kural ve karar değişimleri tamamen veri tabanı seviyesinde gerçekleştirilebilmektedir. Bunun daha rahat yapılabilmesi için sunucu tarafında başka bir uygulama geliştirilebilir. Böylece bir hastane personeli bile yeni kuralları sisteme girebilir.

Dikkat edilirse bu tasarım doktora birden fazla mesaj gönderilmesine izin vermektedir. Ayrıca bir kararın, başka bir kararın üst katmanı olması da mümkündür. Bu sayede son derece karmaşık ve detaylı bir karar çıkarım mekanizması oluşturulabilir.

Oluşturulan bu yapı Java Servlet uygulamasında, uygulama düzeyinde bir dizi nesnede tutulabilir ve gelen parametre değerleri hızla değerlendirilebilir. Bunun için bellekte Şekil 11'de görülen ağaç yapısı oluşturulmalıdır.

Ağaç yapısındaki her bir düğüm için KararDugumu isimli sınıftan nesneler oluşturulur. Bu sınıfta cocuk isimli bir özellik bildirilmiştir. Bu özelliğin türü KararDugumu dizisidir. Böylece her düğüm kendisine bağlı alt düğümleri bilmektedir. Bu sınıfa ait bir nesnenin sahip olduğu özellikler Şekil 12'de verilmiştir.

```
private String id, parametre, sonucDogru,
                sonucYanlis;
private int baglanti, kural, cinsiyet, yasAlt, yasUst;
private double deger;
private ArrayList<KararDugumu> cocuk;
```

Şekil 10. *KararDugumu* sınıfının özellikleri

Dikkat edilirse bazı düğümlerdeki bazı özellikler herhangi bir değer almayacaktır. Örneğin sonucDogru ve sonucYanlis özellikleri yalnızca karar katmanı için geçerlidir. Bunun gibi deger özelliği de yalnızca kurallar için geçerlidir. Yine de hepsini kapsayacak genel bir sınıfın oluşturulması, ağaç yapısının da daha rahat oluşturulması imkanını vermektedir. Uygulama seviyesinde yalnızca karar düğümleri mesaj üretmek için kullanılacaktır. Zira, her düğüm cocuk özelliği sayesinde kendisine bağlı alt düğümleri de bilmektedir. Bu bakımdan daha önce de değinildiği gibi her düğüm aslında önermedir ve doğru ya da yanlıştır. Dolaysıyla her önermenin sonucu nesneye gönderilen Şekil 13'deki kotrol(hastabilgisi) mesajıyla tespit edilebilir.

```
private boolean kontrol(Map hastaBilgisi) {
   if (deger != -1) {
      double val = Double.parseDouble(
           (String)hastaBilgisi.get(parametre));
      switch (kural) {
         case -2:
            if (val < deger) return true;
            else return false;
         case -1:
            if (val <= deger) return true;
            else return false;
         case 0:
            if (val == deger) return true;
            else return false;
         case 1:
            if (val >= deger) return true;
            else return false;
         case 2:
            if (val > deger) return true;
            else return false;
      }
      return false;
   }
   else {
      if (baglanti == 0) {
         for (int i = 0; i < cocuk.size(); i++)
            if (!getCocuk(i).kontrol(hastaBilgisi)) return false;
         return true;
      } else {
         for (int i = 0; i < cocuk.size(); i++)
            if (getCocuk(i).kontrol(hastaBilgisi)) return true;
         return false;
      }
   }
}
```

Şekil 11. Bir düğümdeki önermenin sonucu

Görüldüğü gibi düğümün kural düğümü olup olmadığı değer özelliğinden anlaşılmaktadır. Eğer özelliğe -1 değeri atanmışsa düğümün bir katman olduğu anlaşılır. hastaBilgisi ise hastanın mobil uygulaması ile gönderdiği verileri barındırır.

## 4. SİSTEMİN SINANMASI

Bu çalışmada önerilen sistemle literatürde tam olarak örtüşen başka bir çalışma bulunamadığından, yapılmış çalışmalardan elde edilen bulgularla buradakileri kıyaslamak mümkün olmamıştır. Bu nedenle bu kısımda yalnızca sınamalarda elde edilen sonuçlar sunulmuştur.

Uygulamanın sınanması için çift çekirdekli 3.00 GHz işlemci ve 2 GB iç belleğe sahip bir sunucu kullanılmıştır. Sunucuda Windows 7 Ultimate işletim sistemi koşturulmaktadır. Daha önce de değinildiği gibi sunucu üzerinde Apache Tomcat 7 ve MySQL 5.5.29 çalıştırılmaktadır. Sunucu tarafında bulunan mobil cihaz ise dört çekirdekli 1.4 GHz işlemciye sahiptir. Cihaz üzerinde Android 4.1.2 işletim sistemi koşturulmaktadır. Hasta uygulamasının sınanması için ise birbirinden farklı cihazlar kullanılmıştır. Hatta bazıları için iOS hasta takip uygulaması da geliştirilmiştir.

Sınamalar için toplam 60 deneme yapılmıştır. Bu denemelerde hasta uygulamasının farklı yerlerden çalıştırılmasına özen gösterilmiştir. Sunucu ve sunucuyla aynı yerel ağda bulunan mobil cihaz İstanbul–Bakırköy'de bulunmaktadır. Buna karşın farklı yerlerde gerek İnternet üzerinden, gerekse de SMS ile yapılan hasta veri gönderimlerinin hiç birisinde veri kaybı veya hata gerçekleşmemiştir. Bölgelere göre yapılan deneme sayısı, veri aktarım biçimi, saniye cinsinden ortalama veri aktarım süresi ($\mu$) ve bu aktarım sürelerinin standart sapması ($\sigma$) Tablo 3'te verilmiştir. İnternet'ten veri aktarımı iki ana gruba ayrılabilir. Bunlardan birisi hastanın verilerini Wi-Fi ile bağlı olduğu modem üzerinden göndermesiyken, diğeri de 3G aracılığıyla iletmesidir. İnternet ile iletim hızı ölçülürken, veri gönderilip sunucudan yanıt gelinceye kadar geçen süre dikkate alınmıştır. SMS ile iletim hızı ölçülürkense verinin cihazdan sunucu tarafındaki mobil aygıta ulaşana

kadar geçen süre ölçülmüştür. İnternet iletimi aynı cihaz üzerinde ölçüldüğünden daha hassastır.

Tablo 3'te görüldüğü gibi verilere göre farklı bölgelerde farklı iletim hızı varmış gibi bir anlam çıkmaktadır. Oysa tam aksine, bu tablo veri iletim hızının bölgeye bağımlı olmadığını göstermektedir.

Zira sunucuya en yakın bölgedeki iletim hızı en yavaş olanıdır. Antalya gibi sunucuya oldukça uzak bir yerle kıyaslandığında bile önemli ölçüde yavaş olduğu görülmektedir. Bunun temel sebebi Antalya'dan sisteme erişen cihazın HSPA+ teknolojisinden yararlanmasıdır. Avcılar'dan veri gönderen cihaz ise 10 Mbps indirme ve 1 Mbps yükleme hızına sahip bir Wi-Fi bağlantıdır. Böylece burada önerilen yöntemin yapılan sınamalara göre mesafeden etkilenmediği, yalnızca bağlantı teknolojisinden etkilendiği söylenebilir.

Tablo 3. Hasta Takip Sisteminin Sınama Sonuçları

| Bölge | D.Sayı | Aktarım | $\mu$ | $\sigma$ |
|---|---|---|---|---|
| Bakırköy | 10 | SMS | 4,747 | 0,439 |
| | 10 | İnternet | 2.782 | 1.292 |
| Avcılar | 10 | SMS | 5,2 | 0,991 |
| | 10 | İnternet | 0.125 | 0.048 |
| Antalya | 10 | SMS | 4.894 | 0.67 |
| | 10 | İnternet | 0.436 | 0.243 |

Yapılan sınamalarda herhangi bir hata ile karşılaşılmamıştır. Ayrıca veri aktarım süreleri İnternet için 1.115 sn ve SMS için 4.943 sn genel ortalama ile oldukça makuldür denebilir.

## 5. SONUÇ

Bu çalışmada özellikle orta büyüklükteki hastaneler için düşük maliyetli bir mobil hasta takip sistemi önerilmiştir. Kullanılan teknolojilerin hemen hepsi ücretsiz, bir çoğu da açık kaynaktır. Hastaların sunucuyla tercihlerine göre İnternet üzerinden veya SMS ile bağlantı kurmaları sağlanmıştır. Başlı başına farklı bir çalışma konusu olacağından sistemin güvenliği üzerine ilave bir araştırma yapılmamıştır.

Sınama sonuçları sistemin orta büyüklükteki hastanelerin hastalarına başarıyla hizmet verebileceğini göstermektedir. Ancak büyük hastaneler için mevcut sınamalara göre kesin bir çıkarımda bulunmak doğru olmayacaktır. Zira büyük hastaneler için yapılacak sınamanın çok daha kapsamlı olması gerekmektedir. Bu da sınama maliyetlerini oldukça yükseltecektir.

İleriki çalışmalarda hasta takip sisteminin güvenliğine odaklanılması düşünülmektedir. Ayrıca burada önerilen sistemin daha büyük hastaneler için de kullanılmasına yönelik araştırmalar yine ileriki çalışmaların konusu olacaktır.

## Kaynakça


1. Android Developers, (2013). http://developer.android.com/.
2. Charl A., LeRoux B., (2011). Web apps are cheaper to develop and deploy than native apps, but can they match the native user experience?, communications of the acm, 54(5), 49-53.
3. Gartner Inc., (2013).
4. http://www.gartner.com/newsroom/id/2237315.
5. Google Project Hosting, (2013).
6. http://code.google.com/p/google-gson/downloads/list/.
7. Halteren A.V., Bults R., Wac K., Konstantas D., Widya I., Dokovsky N., Koprinkov G., Jones V., Herzog R., (2004). Mobile Patient Monitoring: The MobiHealth System, The Journal on Information Technology in Healthcare 2(5), 365–373.
8. Katz J. E., Rice R. E., (2009). Public views of mobile medical devices and services: A US national survey of consumer sentiments towards RFID healthcare technology. International Journal of Medical Informatics 78, 104–114.
9. Laudon K. C., Laudon J. P., (2011).Yönetim Bilişim Sistemleri Dijital İşletmeyi Yönetme, Çev. Ed. Yozgat U., Nobel Yayıncılık.
10. Oracle Corporation and/or its affiliates, (2013). http://dev.mysql.com/.
11. Oracle, (2013). http://www.java.com/.

12. Sun F.S., Weng Y.H., Grigsby J., (2010). Smartphones for Geological Data Collection - an Android Phone Application. Eos 91(59).
13. The Apache Foundation, (2013). http://tomcat.apache.org/.
14. The Eclipse Corparation, (2013). http://www.eclipse.org/.
15. Weng Y.H., Sun F.S., Grigsby J.D., (2012). GeoTools: An android phone application in geology. Computers & Geosciences 44 24–30.