ACİL SERVİS YÖNETİM SİSTEMİ

Proje Analiz Dokümanı

1. Giriş

1.1 Amaç

Bu doküman, Acil Servis Yönetim Sistemi'nin teknik analizini içermektedir. Dokümanın amacı, sistemin temel bileşenlerini, veri yapılarını, algoritmalarını ve işlevselliğini detaylı bir şekilde açıklamaktır. Bu analiz, geliştiricilerin sistemi anlamasını, bakımını ve gelecekteki geliştirmelerini kolaylaştırmayı hedeflemektedir.

1.2 Kapsam

Bu analiz dokümanı, Acil Servis Yönetim Sistemi'nin C programlama dili ile geliştirilmiş kaynak kodunu kapsamaktadır. Sistem, hasta kabulü, triyaj değerlendirmesi, hasta önceliklendirme, muayene yönetimi, reçete düzenleme ve HL7 mesajlaşma gibi temel acil servis işlemlerini içermektedir.

1.3 Tanımlar ve Kısaltmalar

Terim/Kısaltma	Açıklama
Triyaj	Acil servise gelen hastaların durumlarının ciddiyetine göre değerlendirilip önceliklendirilmesi işlemi
HL7	Health Level Seven, sağlık bilgi sistemleri arasında veri alışverişini standardize eden bir protokol
Queue	Kuyruk veri yapısı, FIFO (ilk giren ilk çıkar) prensibi ile çalışır
Stack	Yığın veri yapısı, LIFO (son giren ilk çıkar) prensibi ile çalışır
SLA	Service Level Agreement (Hizmet Seviyesi Anlaşması)

2. Sistem Mimarisi

2.1 Veri Yapıları

Sistem aşağıdaki temel veri yapılarını kullanmaktadır:

Hasta Yapısı (struct Hasta)

- **tcKimlikNo:** Hastanın TC kimlik numarası (string)
- **hastaNumarasi:** Acil servise giriş sırasında atanan numara (string)
- **ad/soyad:** Hasta adı/soyadı (string)
- **dogumTarihi:** YYYY-MM-DD formatında doğum tarihi (string)
- **cinsiyet:** Cinsiyet bilgisi (karakter)
- **adres:** Hasta adresi (string)
- **telefon:** İletişim numarası (string)
- **OncelikPuani:** Triyaj sonucu belirlenen öncelik puanı (integer)

Triyaj Bilgisi Yapısı (struct TriyajBilgisi)

- **aciliyetSeviyesi:** 1-5 arası aciliyet seviyesi (1 en acil, 5 en az acil) (integer)
- **sikayet:** Hastanın acil servise başvuru şikayeti (string)
- **hayatiTehlikesi:** 'E' veya 'H' (karakter)
- **atesDegeri:** Vücut sıcaklığı (float)
- **gelisTarihi/gelisSaati:** Acil servise geliş zamanı (string)
- **color:** Aciliyet seviyesi renk kodu (string)

İlaç Yapısı (struct Ilac)

- **ilacAdi:** İlaç adı (string)
- **doz:** Kullanılacak doz (string)
- **kullanimSekli:** İlacın kullanım talimatları (string)
- **adet:** Reçetelenecek ilaç miktarı (integer)

Reçete Yapısı (struct Recete)

- **receteNo:** Otomatik üretilen reçete numarası (string)
- **hastaId:** Reçete yazılan hastanın TC Kimlik numarası (string)
- **doktorId:** Reçeteyi yazan doktor kimliği (string)
- **tarih:** Reçete tarihi (string)
- **ilaclar:** Reçetedeki ilaçların listesi (Ilac pointer)
- **ilacSayisi:** Reçetedeki ilaç sayısı (integer)

Hasta Kuyruğu (struct HastaQueue)

- **front:** Kuyruğun başındaki hasta (HastaNode pointer)
- **rear:** Kuyruğun sonundaki hasta (HastaNode pointer)
- **size:** Kuyruktaki hasta sayısı (integer)

Reçete Yığını (struct ReceteStack)

- **top:** Yığının üstündeki reçete (ReceteNode pointer)
- **size:** Yığındaki reçete sayısı (integer)

HL7 Mesajı (union HL7Message)

- **rawHL7:** Ham HL7 mesaj formatı (string)
- **hl7Format:** Bölümlere ayrılmış HL7 mesaj formatı (struct)

2.2 İşlevsel Bileşenler

Sistemin temel işlevsel bileşenleri şunlardır:

1. **Hasta Kabul Modülü**
    a. Hasta bilgilerinin girilmesi
    b. Triyaj bilgilerinin alınması
    c. Öncelikli hasta kuyruğuna ekleme
2. **Triyaj Modülü**
    a. Hasta şikayetlerine göre aciliyet değerlendirmesi
    b. Aciliyet seviyesine göre renk kodu atama
3. **Hasta Yönetim Modülü**
    a. Hasta listesini görüntüleme
    b. Öncelikli hasta seçimi
    c. Hasta kaydının silinmesi

4. **Reçete Modülü**
    a. Reçete oluşturma
    b. İlaç bilgilerinin girilmesi
    c. Reçete saklama ve görüntüleme

5. **HL7 Entegrasyon Modülü**
    a. Hasta bilgilerinin HL7 formatına dönüştürülmesi
    b. HL7 mesajı gönderimi
    c. HL7 mesajından hasta bilgisi çıkarımı

2.3 Akış Şeması

Sistemin genel akış şeması aşağıdaki gibidir:

1. Hasta acil servise gelir
2. Hasta bilgileri ve triyaj değerlendirmesi yapılır
3. Triyaj algoritması aciliyet seviyesini belirler
4. Hasta öncelik puanına göre kuyruğa yerleştirilir
5. En yüksek öncelikli hasta muayeneye alınır
6. Muayene sonrası reçete düzenlenir (gerekirse)
7. Hasta kaydı kuyruktan silinir
8. HL7 mesajları ile diğer sistemlerle entegrasyon sağlanır

3. Fonksiyonel Gereksinimler

3.1 Hasta Kaydı ve Kabulü

Sistem, hasta kaydı oluşturma ve hastayı acil servis kuyruğuna alma işlevini sağlamaktadır. `hastaKabul()` fonksiyonu bu işlemi gerçekleştirir.

Özellikler:

- Hasta temel demografik bilgilerinin (TC, ad, soyad, doğum tarihi, cinsiyet, vb.) kaydı
- Otomatik hasta numarası atama (`hastaNumarasiCounter` değişkeni ile)
- Triyaj bilgilerinin alınması ve değerlendirilmesi
- Öncelik puanı hesaplanarak kuyruğa yerleştirme

3.2 Triyaj Değerlendirmesi

Sistem, `TriyajBelirleme()` fonksiyonu ile hastanın aciliyet seviyesini belirlemektedir.

Triyaj Kriterleri:

1. **Kırmızı (Seviye 1):** Hayati tehlike, şok belirtileri, bilinç kaybı, ciddi kafa travması, ani felç, kalp durması, masif kanama, vb.
2. **Turuncu (Seviye 2):** Yoğun kanama, şiddetli karın ağrısı, görme kaybı, şiddetli yanıklar, göğüs ağrısı, vb.
3. **Sarı (Seviye 3):** Orta şiddette ağrı, kırık şüphesi, hafif yanık, tekrarlayan kusma, şiddetli baş ağrısı, vb.
4. **Yeşil (Seviye 4):** Hafif burkulma, soğuk algınlığı, hafif kas ağrısı, kusma olmadan bulantı, vb.
5. **Mavi (Seviye 5):** Acil olmayan durumlar

Özel Değerlendirmeler:

- Ateş 39°C üzerindeyse aciliyet seviyesi yükseltilir
- Hayati tehlike olduğu belirtilirse otomatik olarak Seviye 1 (Kırmızı) atanır

3.3 Hasta Sıralaması ve Önceliklendirme

Sistem, hastaları öncelik puanına göre sıralamaktadır. Öncelik puanı şu formül ile hesaplanır:

```
öncelikPuanı = (6 - triyaj.aciliyetSeviyesi) * 20
```

Bu formül, en acil hastaların (Seviye 1) en yüksek puanı (100) almasını, en az acil hastaların (Seviye 5) en düşük puanı (20) almasını sağlar.

Kuyruk Yapısı:

- Hastalar öncelik puanına göre azalan sırada (yüksek öncelikli hasta önce) dizilir
- Her yeni hasta, öncelik puanına göre kuyruğun uygun konumuna yerleştirilir
- Böylece klasik FIFO kuyruğu yerine öncelikli kuyruk uygulanmış olur

3.4 Muayene İşlemleri

Sistem, kuyruktaki en öncelikli hastayı muayeneye alma ve muayene sonrası hastayı kuyruktan çıkarma işlevlerini sağlar.

İşlemler:

- `hastaGetir()` fonksiyonu ile en yüksek öncelikli hasta bilgilerine erişim
- Muayene sonrası hasta kaydının silinmesi (`hastaSil()` fonksiyonu)
- Muayene sonrası gerekirse reçete düzenleme seçeneği

3.5 Reçete Yönetimi

Sistem, muayene sonrasında reçete oluşturma ve saklama işlevini sağlar.

Özellikler:

- Otomatik reçete numarası üretimi (`receteNoUret()` fonksiyonu)
- Birden fazla ilaç bilgisinin kaydı
- Reçetelerin yığın (stack) veri yapısında saklanması
- Reçetelerin listelenmesi ve görüntülenmesi

3.6 HL7 Entegrasyonu

Sistem, hasta bilgilerini HL7 standardına uygun formatta dönüştürebilmektedir.

Özellikler:

- Hasta ve triyaj bilgilerinden HL7 mesajı oluşturma
- HL7 mesajının bölümlere ayrılmış yapısı (messageHeader, patientInfo, observationInfo, vb.)
- HL7 mesajından hasta bilgisi çıkarımı

4. Teknik Detaylar

4.1 Veri Yapıları Analizi

Bağlı Liste Yapısı

Sistem, hastalar için tek yönlü bağlı liste kullanmaktadır (HastaNode yapısı). Bu yapı, dinamik hasta ekleme ve çıkarma işlemlerini etkin bir şekilde yönetmeyi sağlar. Her düğüm, hasta bilgilerini ve triyaj bilgilerini içermektedir.

Öncelikli Kuyruk İmplementasyonu

HastaQueue yapısı, klasik bir FIFO kuyruğu yerine öncelik puanına göre sıralama yapan özel bir kuyruk uygulamasıdır. Yeni hasta eklendiğinde, öncelik değerine göre kuyruğun doğru konumuna yerleştirilir, böylece en yüksek öncelikli hasta her zaman kuyruğun başında bulunur.

Yığın (Stack) Yapısı

Reçeteler için kullanılan ReceteStack yapısı, LIFO (son giren ilk çıkar) prensibine göre çalışır. Bu, en son oluşturulan reçetenin en üstte olmasını sağlar, bu da tipik olarak en son işlemlerin daha hızlı erişilebilir olması gereken sağlık sistemleri için uygundur.

Union Kullanımı

HL7 mesajları için union veri yapısı kullanılmıştır. Bu, aynı bellek alanını hem ham HL7 mesajı hem de yapılandırılmış formatta kullanabilmeyi sağlar, böylece bellek kullanımı optimize edilir.

4.2 Algoritma Analizi

Triyaj Algoritması

Triyaj algoritması (`TriyajBelirleme()` fonksiyonu), aşağıdaki kriterleri kullanarak aciliyet seviyesini belirlemektedir:

1. **Öncelikli Kontrol:** Hayati tehlike durumu (E/H)
2. **Anahtar Kelime Analizi:** Hasta şikayetlerinde belirli anahtar kelimelerin varlığı
3. **Ateş Değerlendirmesi:** 39°C üzerindeki ateş durumunda aciliyet seviyesinin yükseltilmesi

Algoritmanın zaman karmaşıklığı O(n) şeklindedir, burada n şikayet metninin uzunluğudur.

Öncelikli Hasta Yerleştirme Algoritması

`hastaKabul()` fonksiyonundaki hasta yerleştirme algoritması, bağlı listeyi dolaşarak hastayı öncelik puanına göre doğru konuma yerleştirir. Bu algoritmanın zaman karmaşıklığı ortalama durumda O(n)'dir, burada n kuyruktaki hasta sayısıdır.

Reçete Numarası Üretme Algoritması

`receteNoUret()` fonksiyonu, tarih bilgisi ve rastgele sayı kombinasyonunu kullanarak benzersiz reçete numaraları üretir. Bu, çakışma olasılığını minimuma indiren basit ancak etkili bir yöntemdir.

4.3 Bellek Yönetimi

Dinamik Bellek Tahsisi

Sistem, çeşitli yapılar için dinamik bellek tahsisi yapmaktadır:

- Hasta düğümleri için (`malloc(sizeof(HastaNode))`)
- Reçete düğümleri için (`malloc(sizeof(ReceteNode))`)
- İlaç dizileri için (`malloc(sizeof(Ilac) * ilacSayisi)`)

Bellek Sızıntısı Önleme

Sistem, bellek sızıntılarını önlemek için kapsamlı önlemler içermektedir:

- `freeHastaQueue()` fonksiyonu hasta kuyruğunda ayrılan tüm belleği serbest bırakır
- `freeReceteStack()` fonksiyonu reçete yığınında ayrılan tüm belleği serbest bırakır
- Reçete içindeki ilaç dizileri (`ilaclar`) için ayrı bellek serbest bırakma işlemi

Bellek Hata Kontrolü

Sistem, bellek tahsisi başarısız olduğunda uygun hata mesajları ve çıkış stratejisi içermektedir:

```
if(yeniNode == NULL) {
    fprintf(stderr, "Bellek tahsisi basarisiz!\n");
    exit(EXIT_FAILURE);
}
```

5. Test Senaryoları

5.1 Hasta Kabul ve Triyaj Testleri

Test No	Senaryo	Beklenen Sonuç
T1-01	Hayati tehlikesi olan hasta kaydı	Aciliyet Seviyesi: 1 (Kırmızı), Öncelik Puanı: 100
T1-02	"Ciddi kafa travması" şikayeti olan hasta	Aciliyet Seviyesi: 1 (Kırmızı), Öncelik Puanı: 100
T1-03	"Orta şiddette ağrı" şikayeti, ateş 39.5°C	Aciliyet Seviyesi: 1 (Kırmızı), Öncelik Puanı: 100
T1-04	"Hafif baş ağrısı" şikayeti, ateş 36.5°C	Aciliyet Seviyesi: 4 (Yeşil), Öncelik Puanı: 40

5.2 Öncelikli Kuyruk Testleri

Test No	Senaryo	Beklenen Sonuç
T2-01	Ardışık 3 hasta (öncelik puanları: 40, 80, 60)	Kuyruk: [80, 60, 40]
T2-02	Aynı öncelik puanlı hastalar	Geliş sırasına göre sıralanmalı
T2-03	En yüksek öncelikli hastanın muayene sonrası silinmesi	Kuyruğun başı bir sonraki en yüksek öncelikli hasta olmalı

5.3 Reçete Yönetimi Testleri

Test No	Senaryo	Beklenen Sonuç
T3-01	Tek ilaçlı reçete oluşturma	Reçete stack'ine eklenmeli ve benzersiz reçete no içermeli
T3-02	Çoklu ilaçlı reçete oluşturma	Tüm ilaçlar reçeteye dahil edilmeli
T3-03	Reçete listesi yazdırma	Tüm reçeteler en yeniden en eskiye doğru listelenmeli

5.4 HL7 Entegrasyon Testleri

Test No	Senaryo	Beklenen Sonuç
T4- 01	Hasta bilgilerinden HL7 mesajı oluşturma	Geçerli HL7 formatında mesaj oluşturulmalı
T4- 02	HL7 mesajından hasta bilgisi çıkarma	Hasta bilgileri doğru şekilde çıkarılmalı

6. Performans Değerlendirmesi

6.1 Zaman Karmaşıklığı Analizi

İşlem	Karmaşıklık	Açıklama
Hasta Kabul	O(n)	n: Kuyruktaki hasta sayısı
Triyaj Değerlendirme	O(m)	m: Şikayet metninin uzunluğu
Hasta Getirme	O(1)	Kuyruğun başındaki hastaya doğrudan erişim
Hasta Silme	O(1)	Kuyruğun başındaki hastayı silme
Reçete Oluşturma	O(1)	Stack'e push işlemi
Reçete Listeleme	O(n)	n: Reçete sayısı

6.2 Bellek Kullanımı Analizi

Veri Yapısı	Yaklaşık Bellek Kullanımı	Optimizasyon Durumu
Hasta Düğümü	~700 bayt	Orta seviye optimizasyon
Reçete Düğümü	~100 bayt + (ilacSayisi * 180 bayt)	Dinamik boyut
HL7 Mesajı	~1000 bayt	Sabit boyut (MAX_HL7_MESSAGE)

6.3 Ölçeklenebilirlik Değerlendirmesi

Mevcut yapı, küçük-orta ölçekli acil servisler için yeterli olacaktır. Büyük ölçekli acil servisler için aşağıdaki performans iyileştirmeleri düşünülebilir:

- Daha verimli öncelikli kuyruk yapısı (heap tabanlı)
- Veritabanı entegrasyonu ile kalıcı veri saklama
- Multi-threading desteği ile paralel işlem yapabilme

7. Öneriler ve İyileştirmeler

7.1 Kod İyileştirmeleri

1. **Kod Modülaritesi:** Bazı fonksiyonlar (özellikle `menuGoster()`) çok uzundur ve birden fazla işlev içermektedir. Bunlar daha küçük, özel amaçlı fonksiyonlara bölünebilir.
2. **Bellek Optimizasyonu:** Bazı string alanları gereksiz yere büyük olabilir (örn. `hastaNumarasi[1000]`). Bu alanların boyutları optimize edilebilir.

7.2 Fonksiyonel İyileştirmeler

1. **Arama Fonksiyonu:** TC kimlik numarası veya hasta numarası ile hasta arama fonksiyonu eklenebilir.
2. **İstatistik Modülü:** Günlük/aylık hasta sayıları, ortalama bekleme süreleri gibi istatistikler tutan bir modül eklenebilir.
3. **Kullanıcı Kimlik Doğrulama:** Sistem üzerinde işlem yapabilecek kullanıcılar için kimlik doğrulama mekanizması eklenebilir.
4. **İlaç Veritabanı:** İlaç bilgilerinin otomatik tamamlanması için bir veritabanı entegrasyonu eklenebilir.
5. **Raporlama Modülü:** Çeşitli parametrelere göre raporlar oluşturabilen bir modül eklenebilir.

7.3 Teknik İyileştirmeler

1. **Veritabanı Entegrasyonu:** Verilerin kalıcı olarak saklanabilmesi için SQLite veya MySQL gibi bir veritabanı entegrasyonu eklenebilir.
2. **Grafik Kullanıcı Arayüzü:** Konsol tabanlı arayüz yerine GTK+ veya Qt gibi kütüphaneler kullanılarak grafik kullanıcı arayüzü geliştirilebilir.
3. **Ağ Yetenekleri:** Sistem, gerçek HL7 mesajlarını TCP/IP üzerinden gönderip alabilecek şekilde geliştirilebilir.
4. **Çoklu Dil Desteği:** Sistem arayüzü için çoklu dil desteği eklenebilir.

8. Programın Akış Diyagramı