T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 401 BİLGİSAYAR MÜHENDİSLİĞİ TASARIMI

Gerçek Zamanlı Verilerle CPU Zamanlama Algoritmalarının Görsel Simülasyonu

B221210588 – YOUNES RAHEBI

Bölüm : BİLGİSAYAR MÜHENDİSLİĞİ
Danışman : Doç.Dr. ABDULLAH SEVİN

2025-2026 Güz Dönemi

ÖNSÖZ

Bu bitirme çalışmasında, “Gerçek Zamanlı Verilerle CPU Zamanlama Algoritmalarının Görsel Simülasyonu” başlığı altında, işletim sistemlerinde derslerde teorik olarak ele alınan CPU zamanlama algoritmaları; gerçek süreç verileri, grafiksel kullanıcı arayüzü ve Gantt şemaları ile bütünleştirilerek uygulamalı bir şekilde incelenmiştir. FCFS, SJF, Priority ve Round Robin algoritmalarının, gerçek zamanlı veriler üzerinden simülasyonu gerçekleştirilmiş; CPU kullanım oranı, throughput, ortalama bekleme ve dönüş süreleri gibi performans ölçütleri yorumlanmıştır.

Bu çalışmanın, hem CPU zamanlama konusunun daha somut ve anlaşılır hâle getirilmesine hem de benzer ders ve projelerde eğitim amaçlı bir araç olarak kullanılmasına katkı sağlaması hedeflenmektedir.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

CPU	: Merkezi İşlem Birimi
OS	: İşletim Sistemi
GUI	: Grafiksel Kullanıcı Arayüzü
PID	: Süreç Kimliği
FIFO	: İlk Giren İlk Çıkar
FCFS	: İlk Gelen İlk Hizmet Alır
SJF	: En Kısa İş İlk
EDF	: En Erken Son Teslim Tarihi Önce
IDLE	: Boşta Durum
KPI	: Anahtar Performans Göstergesi
TAT	: Dönüş Süresi
WT	: Bekleme Süresi
EXE	: Çalıştırılabilir Dosya
PNG	: Portable Network Graphics
CSV	: Virgülle Ayrılmış Değerler
PDF	: Taşınabilir Belge Biçimi
AI	: Yapay Zekâ
ML	: Makine Öğrenmesi

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar Kelimeler: CPU zamanlama, Gantt şeması, işletim sistemi, grafiksel kullanıcı arayüzü, performans analizi.

Bu Tasarım çalışmasının amacı, işletim sistemlerinde kullanılan CPU zamanlama algoritmalarını gerçek zamanlı süreç verileri ile simüle ederek, bu algoritmaların davranışlarını görsel ve etkileşimli bir biçimde inceleyebilmektir. Çalışmanın kapsamında, Windows işletim sisteminde çalışan bir masaüstü uygulaması geliştirilmiş; sistemden psutil kütüphanesi ile çekilen süreç bilgileri kullanılarak [5] FCFS, önişlemeli SJF, önişlemeli Öncelik ve Round Robin algoritmaları uygulanmıştır. Uygulama, her algoritma için elde edilen çizelgeyi Gantt şeması üzerinde göstermekte; CPU kullanım oranı, throughput, ortalama dönüş süresi ve ortalama bekleme süresi gibi performans ölçütlerini hesaplayarak kullanıcıya sunmaktadır.

Geliştirilen grafiksel kullanıcı arayüzü, süreçlerin tablo üzerinde incelenebilmesini, belirli alanların elle düzenlenerek farklı senaryolar oluşturulmasını ve Gantt şemasının yakınlaştırma, kaydırma gibi etkileşimli özelliklerle detaylı olarak analiz edilmesini sağlamaktadır. Proje, PyInstaller kullanılarak tek bir çalıştırılabilir (EXE) dosya hâline getirilmiş [6] ve ek bir kurulum gerektirmeden kullanılabilir duruma getirilmiştir. Elde edilen sonuçlar, farklı zamanlama algoritmalarının aynı süreç kümesi üzerinde nasıl farklı performans değerleri ürettiğini somut olarak ortaya koymakta ve çalışmayı, CPU zamanlama konusunun öğretiminde kullanılabilecek pratik bir eğitim aracı hâline getirmektedir.

BÖLÜM 1. GİRİŞ

1.1. Projenin Amacı ve Kapsamı

1.1.1. Proje Amacı

Bu proje, işletim sistemlerinde kullanılan CPU çizelgeleme algoritmalarını gerçek sistem süreçleri üzerinde simüle eden, görsel ve interaktif bir masaüstü uygulaması geliştirmeyi amaçlamaktadır.

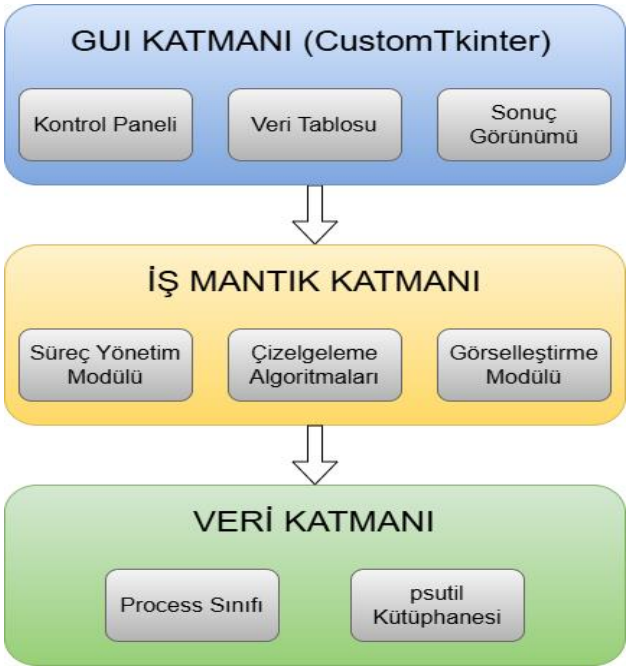
1.1.2. Proje Kapsamı

- Gerçek zamanlı sistem süreçlerinin otomatik olarak çekilmesi
- 4 farklı CPU çizelgeleme algoritmasının implementasyonu
- İnteraktif Gantt Chart görselleştirmesi
- Performans metriklerinin analizi
- Kullanıcı dostu GUI arayüzü

1.2. SİSTEM MİMARİSİ

1.2.1. Uygulamanın Genel Mimarisi

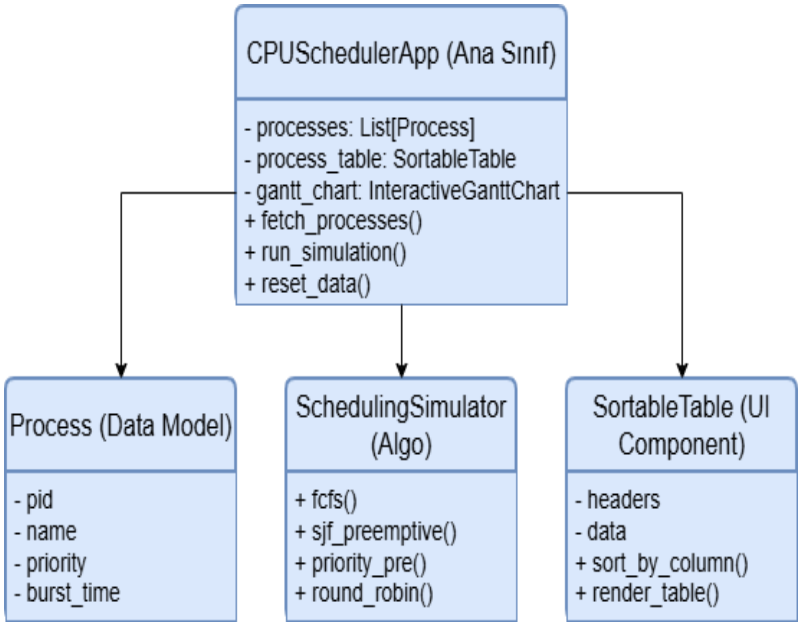
Uygulamanın genel mimarisinde, gerçek zamanlı süreç verilerini toplayan modül, CPU zamanlama algoritmalarını çalıştıran çekirdek ve sonuçları Gantt şeması ile metrikler şeklinde sunan grafiksel arayüz katmanı arasındaki veri akışı gösterilmektedir.



Şekil 1.1. Genel Mimari Şeması

1.2.2. Uygulama Bileşenlerinin Diyagramı

Bu diyagramda, Process sınıfı, zamanlama algoritmaları, Gantt çizim bileşeni, performans metriklerini hesaplayan modül ve kullanıcı arayüzü gibi ana bileşenler ile bunlar arasındaki ilişkiler ve etkileşimler bütüncül olarak sunulmaktadır.



Şekil 1.2. Bileşen Diyagramı

1.3. CPU Çizelgeleme Algoritmaları

1.3.1. CPU Zamanlama Algoritmalarının Karşılaştırılması

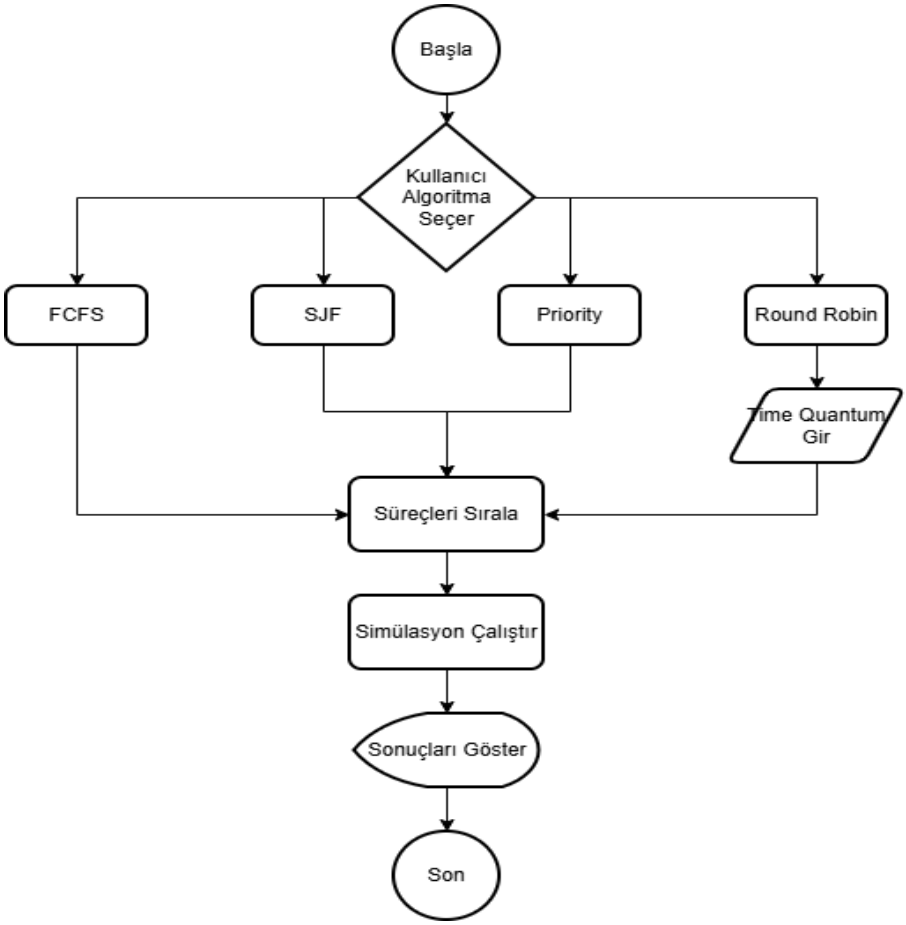
Tabloda FCFS, önişlemeli SJF, önişlemeli Öncelik ve Round Robin algoritmaları; önalım durumu, temel çalışma prensibi, güçlü ve zayıf yönleri ile hangi senaryolarda daha uygun oldukları açısından yan yana karşılaştırılmaktadır [1, 7].

Tablo 1.1. Algoritma Karşılaştırma Tablosu

Algoritma	Tip	Önalım	Zaman Karmaşıklığı	Kullanım Alanı
FCFS	Non-preemptive	Hayır	O(n)	Basit sistemler
SJF	Preemptive	Evet	$O(n^2)$	Batch işlemler
Priority	Preemptive	Evet	$O(n^2)$	Gerçek zamanlı sistemler
Round Robin	Preemptive	Evet	O(n)	Time-sharing sistemler

1.3.2. Algoritma Seçim Sürecinin Akış Şeması

Akış şemasında, sistemin yanıt süresi, adalet, öncelik gereksinimleri ve süreç özellikleri göz önüne alınarak hangi CPU zamanlama algoritmasının tercih edileceğine yönelik adım adım karar süreci gösterilmektedir.



Şekil 1.3. Algoritma Seçim Akış Şeması

1.4. RİSK ANALİZİ

1.4.1. Proje Risk Matrisinin Gösterimi

Risk matrisinde, proje boyunca ortaya çıkabilecek teknik ve yönetimsel riskler olasılık ve etki düzeylerine göre konumlandırılmakta, böylece hangi risklerin öncelikli olarak ele alınması gerektiği görsel olarak vurgulanmaktadır.

Tablo 1.2. Risk Matrisi Tablosu

Risk	Olasılık	Etki	Öncelik	Azaltma Stratejisi
Süreç erişim izni sorunu	Orta	Yüksek	Yüksek	Try-catch blokları, izin kontrolü
GUI donma	Yüksek	Yüksek	Kritik	Threading kullanımı
Platform uyumsuzluğu	Düşük	Orta	Orta	Çapraz platform test
Performans sorunları	Orta	Orta	Orta	Süreç sayısı sınırlama

BÖLÜM 2. GUI, VERİ MODELİ VE SÜREÇ ÇEKME

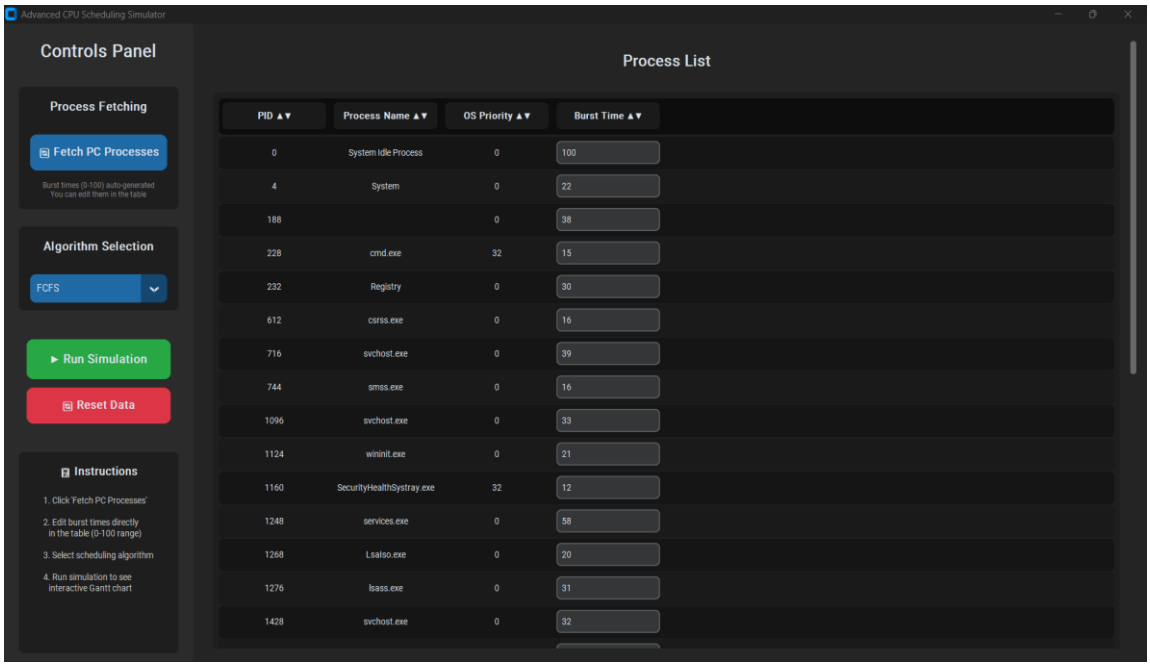
2.1. Aşama 2 Özeti

Bu aşamada projenin temel GUI yapısı ve veri yönetimi bileşenleri geliştirilmiştir:

- CustomTkinter ile modern GUI arayüzü oluşturuldu
- İki kolonlu responsive layout tasarlandı
- Süreç çekme modülü (psutil) implementasyonu
- Sıralanabilir tablo bileşeni geliştirildi
- Loading animasyon sistemi eklendi
- Process veri modeli oluşturuldu

2.2. GUI Mimarisi

Ana pencere, solda sabit genişlikte bir Kontrol Paneli, sağda ise dinamik genişlikte, dikey kaydırma destekli Veri ve Sonuç Paneli olacak şekilde iki kolonlu bir yapı olarak tasarlanmıştır [4]. Kontrol Paneli üzerinden süreç çekme, algoritma seçimi ve simülasyon kontrolü yapılırken, sağ panelde sistemden alınan süreçler modern, sıralanabilir bir tablo içinde görüntülenmektedir.

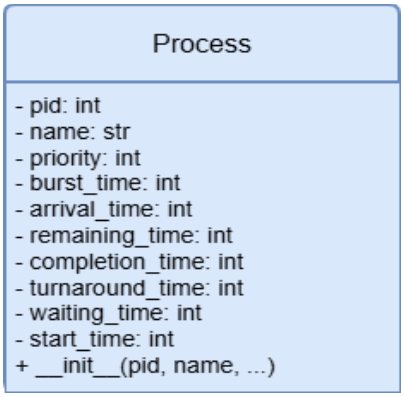


Şekil 2.1. Uygulamanın ana penceresi ve iki kolonlu GUI yerleşimi

2.3. Veri Modeli

2.3.1. Process Sınıf Yapısı

Process sınıfı; her sürecin kimliğini (pid), adını (name), önceliğini (priority) ve CPU çalışma süresini (burst_time) saklar. Simülasyon sırasında oluşan remaining_time, start_time, completion_time, turnaround_time ve waiting_time alanları ise algoritma ilerledikçe güncellenen zamanlama metriklerini temsil eder.



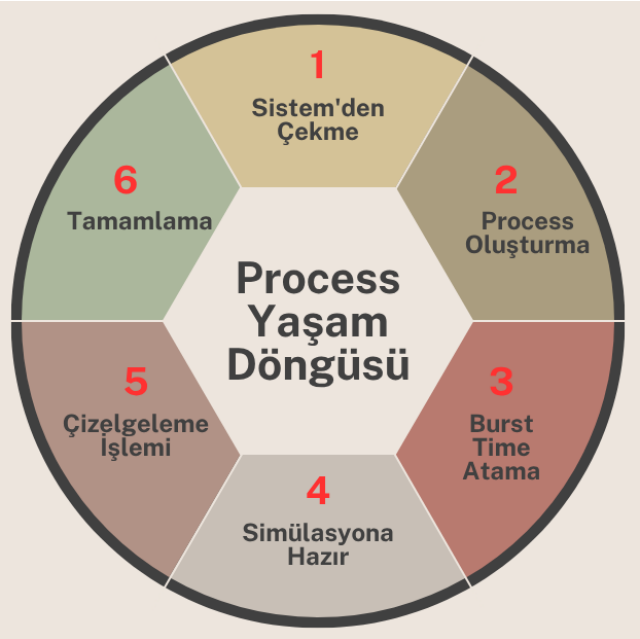
Şekil 2.2. Process sınıfının alanlarını gösteren veri modeli

2.3.2. Process Yaşam Döngüsü

Process yaşam döngüsü, bir sürecin simülasyondaki yolculuğunu adım adım tanımlar:

1. **Sistem'den Çekme:** psutil ile işletim sisteminden süreç listesi alınır [5].
2. **Process Oluşturma:** Her kayıt için bir Process nesnesi oluşturulur, pid, name ve priority atanır.
3. **Burst Time Atama:** CPU kullanımı ve rastgele bileşenler kullanılarak 0-100 aralığında burst time üretilir (gerekirse kullanıcı tarafından düzenlenir).
4. **Simülasyona Hazır**: Tüm süreçler oluşturulup değerleri belirlendikten sonra seçilen algoritma ile simülasyona geçmeye hazır hale gelir.
5. **Çizelgeleme İşlemi**: Seçilen çizelgeleme algoritması çalıştırılır, süreçlerin remaining_time ve ilgili zamanlayıcı değerleri güncellenir.

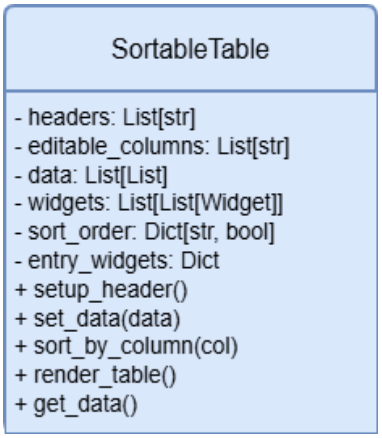
6. **Tamamlama**: Her süreç için completion_time, turnaround_time ve waiting_time hesaplanarak sonuçlar tablo ve grafikler üzerinden kullanıcıya gösterilir.



Şekil 2.3. Process yaşam döngüsünün adımlarını gösteren akış diyagramı

2.4. Sıralanabilir Tablo Bileşeni

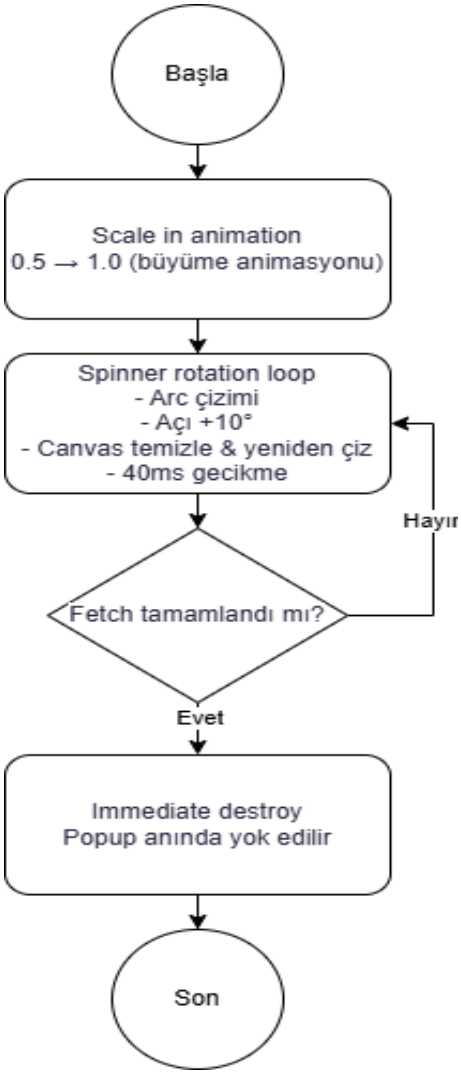
SortableTable sınıfı, süreç listesini GUI üzerinde sıralanabilir ve gerektiğinde düzenlenebilir bir tablo olarak göstermek için tasarlanmıştır. Sınıf; sütun başlıklarını, düzenlenebilir sütunları, tablo verisini ve hücrelere karşılık gelen widget referanslarını tutar. Kullanıcı sütun başlıklarına tıkladığında sort_by_column() metodu çağrılır, veri artan/azalan şekilde yeniden sıralanır ve render_table() ile tablo güncellenir; set_data() ve get_data() ise tabloya veri aktarma ve güncel veriyi geri okuma işlemlerinde kullanılır.



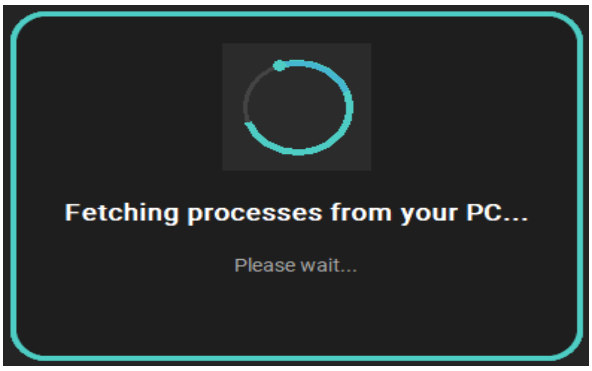
Şekil 2.4. Sıralanabilir tablo alanlarını gösteren veri modeli

2.5. Loading Animasyon Sistemi

Süreçler sistemden çekilirken kullanıcıya görsel geri bildirim sağlamak için, ortasında dairesel spinner bulunan bir loading popup tasarlanmıştır. Popup, önce 0.5 → 1.0 ölçekli bir "scale-in" animasyonu ile ekrana gelir, ardından sürekli dönen bir spinner animasyonu başlar. Bu döngüde arc çizimi yapılır, açı her adımda +10° artırılır, canvas temizlenip yeniden çizilir ve yaklaşık 40 ms gecikme ile akıcı bir hareket elde edilir. Arka plandaki fetch işlemi tamamlandığında animasyon hemen durdurulur ve popup immediate destroy ile anında kapatılarak kullanıcı ana arayüze geri döndürülür.



Şekil 2.5. Süreç Çekme İşleminin İş Akışı ve Threading Yapısını Gösteren Akış Diyagramı



Şekil 2.6. Süreç Çekme İşleminde Gösterilen Loading Penceresi ve Spinner Animasyonu

2.6. Süreç Çekme Modülü

2.6.1 Burst Time Hesaplama ve Özelleştirilebilirlik

Bu projede burst_time değeri, psutil ile okunan CPU kullanım yüzdesine dayalı olarak otomatik üretilmektedir. Her süreç için aşağıdaki formül kullanılmaktadır:

$$burst\_time = min(100, max(0, base\_burst + cpu\_influence + randomization)))$$

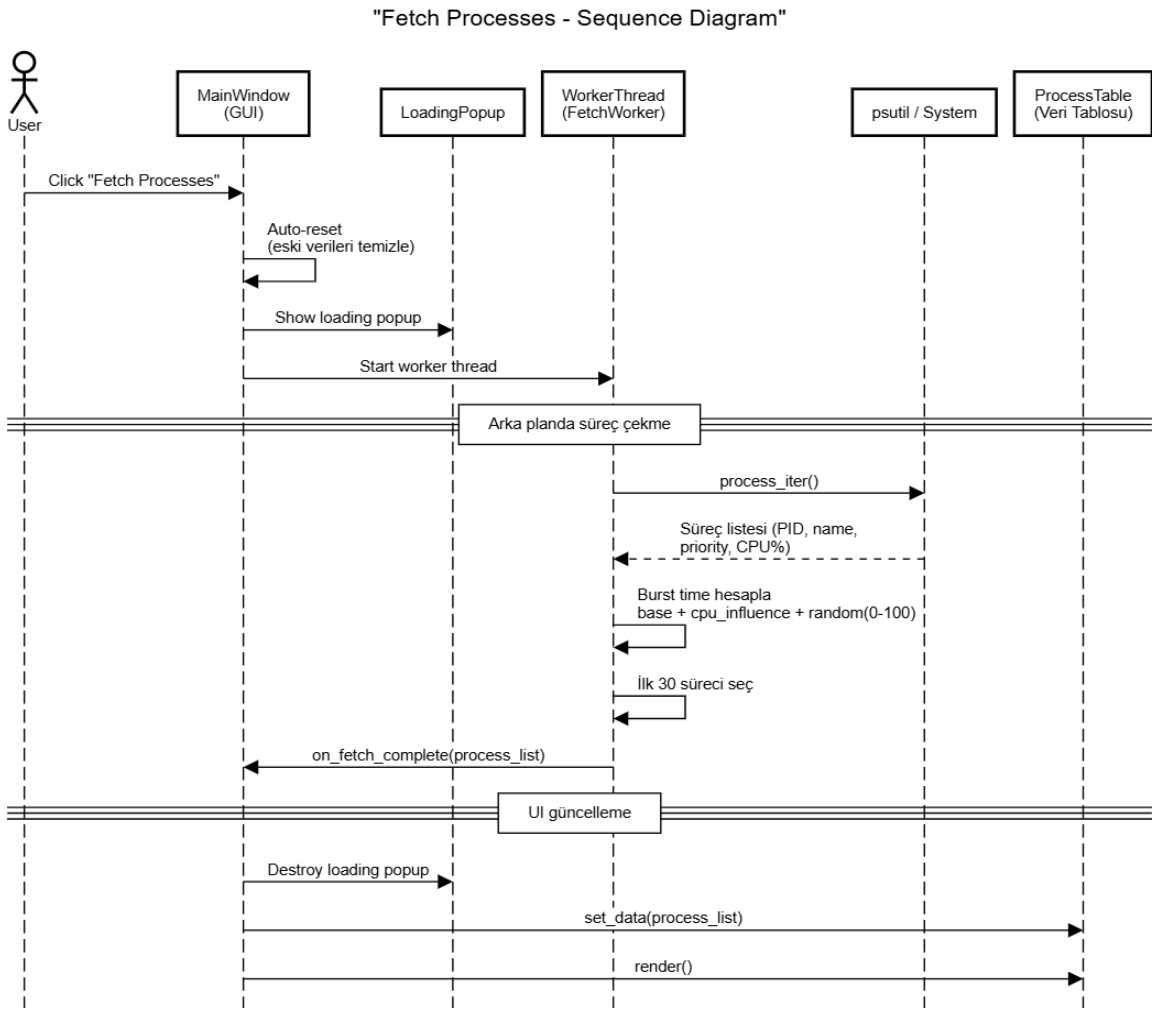
Burada:

- $base\_burst$: $random(5, 50)$
- $cpu\_influence$: $cpu\_percent\ /\ 2$
- $randomization$: $random(-10, 10)$

Bu sayede her süreç için 0-100 aralığında, hem gerçek CPU kullanımını yansıtan hem de randomization terimi sayesinde her çalıştırmada küçük farklılıklar içeren bir burst_time değeri üretilir. Kullanıcı arayüzünde Burst Time sütunu düzenlenebilir bırakıldığı için, kullanıcı dilerse bu otomatik değeri değiştirerek kendi test senaryolarını da oluşturabilmektedir.

2.6.2. Threading ve psutil Entegrasyonu (Sequence Diagram)

Bu Sequence Diagram, kullanıcının “Fetch Processes” butonuna tıklamasıyla başlayan süreci adım adım göstermektedir. Kullanıcı etkileşimi sonrasında MainWindow (GUI) eski verileri temizler, LoadingPopup bileşenini açar ve süreçleri arka planda toplamak için bir WorkerThread başlatır. Worker thread, psutil/System üzerinden süreç bilgilerini (PID, ad, öncelik, CPU yüzdesi) okur [5], burst time değerlerini hesaplayıp ilk 30 süreci seçer ve bu veriyi tamamlandığında tekrar GUI’ye iletir. Son aşamada GUI, loading popup’ını kapatarak ProcessTable bileşenini günceller ve elde edilen süreç listesini kullanıcıya gösterir.



Şekil 2.7. “Fetch Processes” işlemi için threading ve psutil entegrasyonunu gösteren sequence diyagramı

BÖLÜM 3. ALGORİTMALAR, GANTT CHART VE KPI'LAR

3.1. Aşama 3 Özeti

- **4 Çizelgeleme Algoritması Implementasyonu:**
  - FCFS (First Come First Serve)
  - SJF (Shortest Job First - Preemptive)
  - Priority Scheduling (Preemptive)
  - Round Robin
- **İnteraktif Gantt Chart:**
  - Zoom in/out özelliği
  - Pan (sürükleme) desteği
  - Scrollbar ile kaydırma
  - Dinamik legend gösterimi
- **Performans Metrikleri (KPI):**
  - CPU Utilization
  - Throughput
  - Average Turnaround Time
  - Average Waiting Time
- **Sonuç Görselleştirme:**
  - Detaylı sonuç tablosu
  - Sıralanabilir metrikler
  - Renkli Gantt Chart

3.2. Çizelgeleme Algoritmaları

3.2.1. FCFS (First Come First Serve)

En basit CPU çizelgeleme algoritmasıdır. Süreçler geliş sırasına göre işlenir [3].
Özellikler:

- Non-preemptive (önalımsız)
- FIFO (First In First Out) mantığı
- Convoy effect riski

3.2.2. SJF - Preemptive (Shortest Job First)

En kısa işlem süresine sahip süreç öncelikli çalıştırılır. Preemptive versiyonda, daha kısa süreç gelirse çalışan süreç durdurulur [1].
Özellikler:

- Preemptive (önalımlı)
- Optimal ortalama bekleme süresi
- Starvation riski (uzun süreçler)

3.2.3. Priority Scheduling (Preemptive)

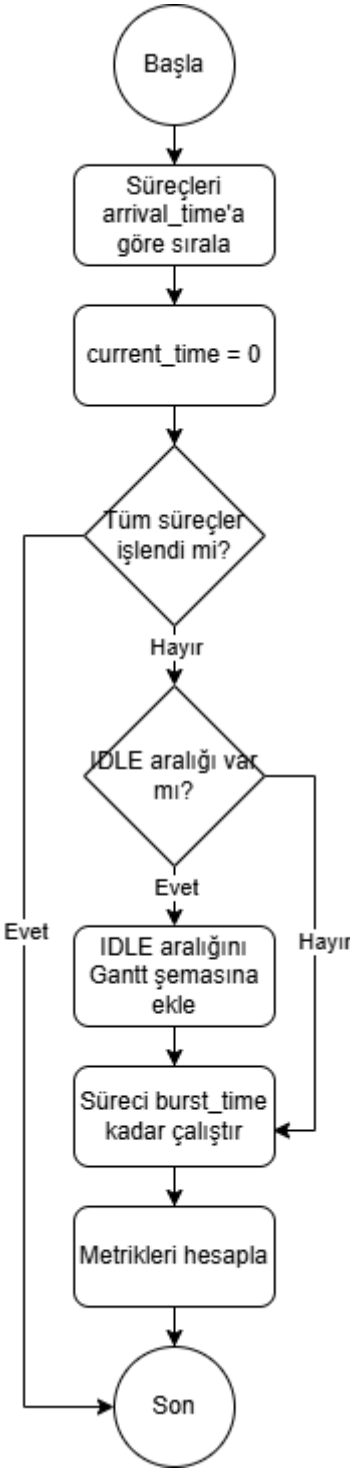
Her sürecin bir öncelik değeri vardır. En yüksek öncelikli süreç (düşük sayı = yüksek öncelik) çalıştırılır.
Özellikler:

- Preemptive
- Priority değeri OS'den alınır
- Priority inversion riski
- Aging ile starvation önlenebilir

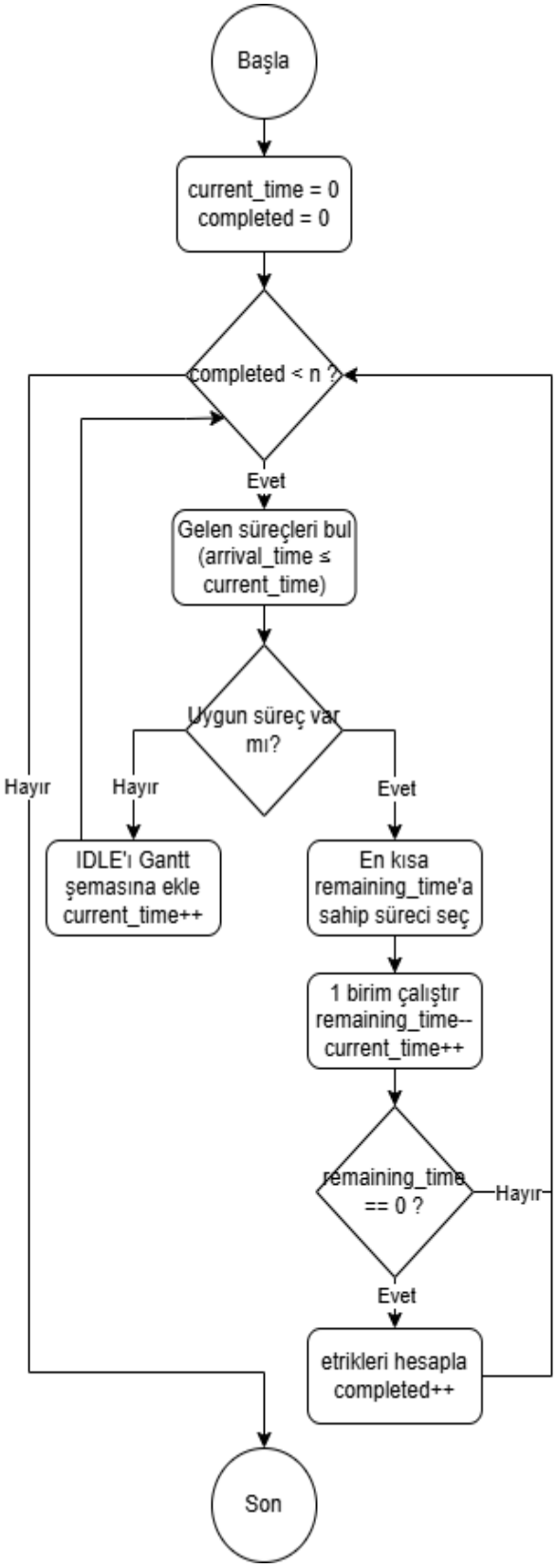
3.2.4 Round Robin

Her süreç belirli bir zaman dilimi (quantum) boyunca çalışır. Süre dolunca sıraya geri eklenir [2].
Özellikler:

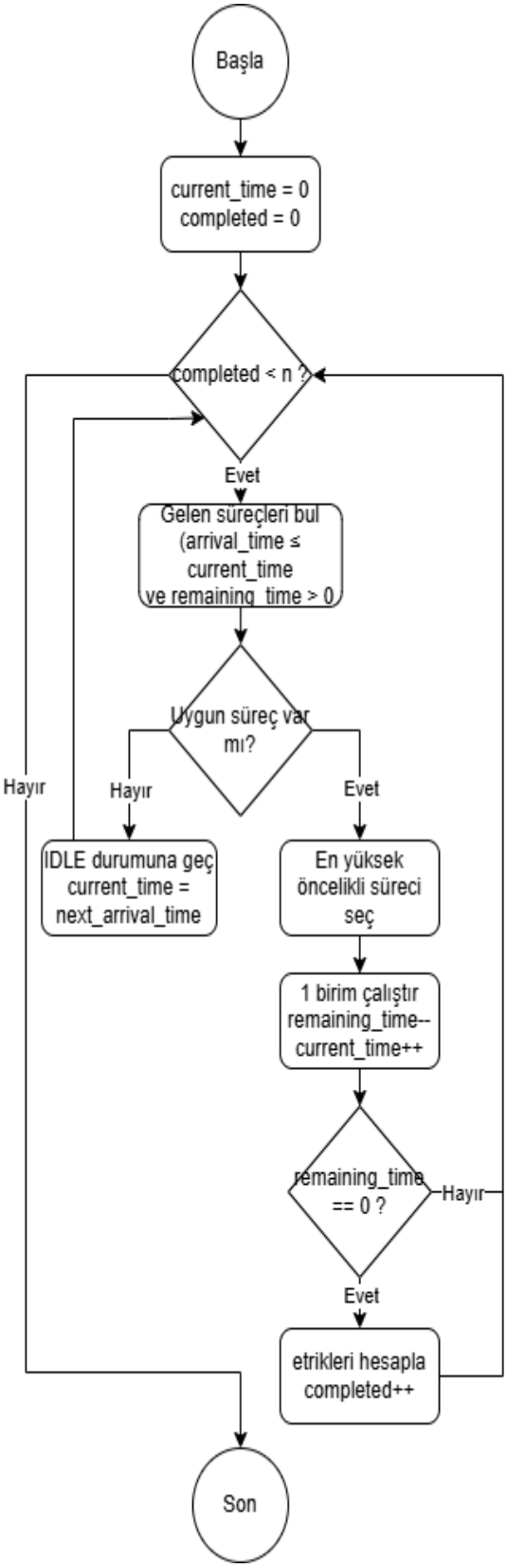
- Preemptive
- Time-sharing sistemler için ideal
- Fair (adil) dağılım
- Quantum seçimi kritik



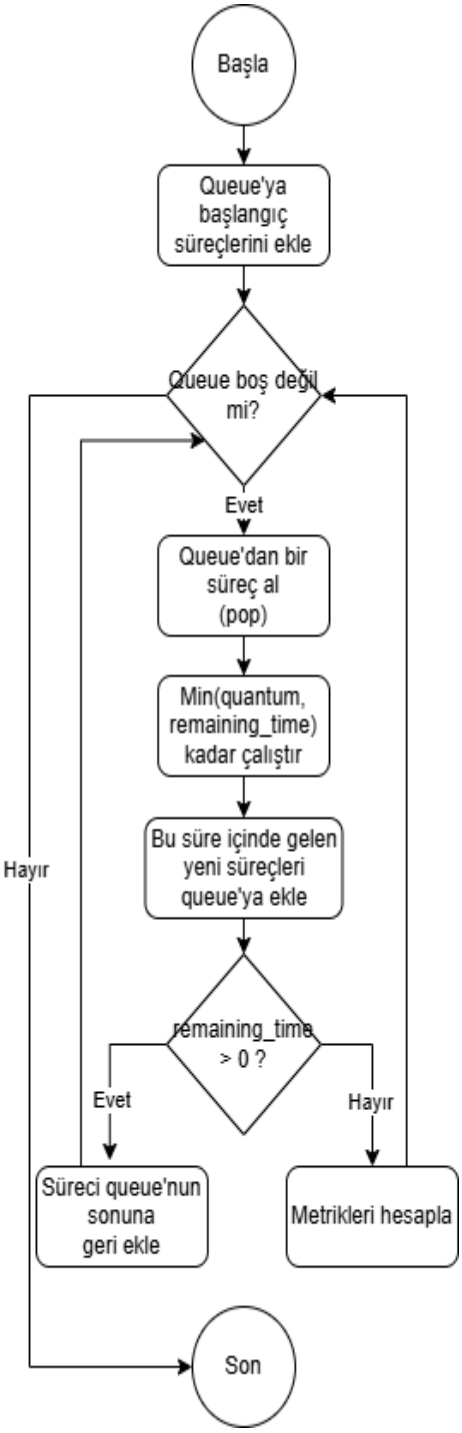
Şekil 3.1. FCFS (First Come First Serve)



Şekil 3.2. SJF - Preemptive (Shortest Job First)



Şekil 3.3. Priority Scheduling (Preemptive)



Şekil 3.4. Round Robin

3.3. İnteraktif Gantt Chart

3.3.1. Gantt Chart Mimarisi

İnteraktif Gantt Chart, simülasyon sonuçlarını görsel zaman çizelgesi olarak sunar. Üç ana katmandan oluşur:

Üst Katman - Kontrol Paneli:

- Başlık ve zoom kontrolleri (Zoom In, Zoom Out, Reset View)
- Kullanıcı bu butonlarla grafiği büyütüp küçültebilir

Orta Katman - Canvas Çizim Alanı:

- Tüm süreç barları bu alanda çizilir
- Her süreç farklı renkle gösterilir
- IDLE zamanları gri renkte belirtilir
- Zaman işaretleri ve süre etiketleri eklenir

Alt Katman - Navigasyon:

- Yatay scrollbar ile uzun zaman çizelgeleri kaydırılabilir
- Scrollbar, görünür alanı temsil eder

Legend (Açıklama Alanı):

- Tüm süreçlerin renk kodları gösterilir
- Scrollable (kaydırılabilir) yapıdadır
- Her sürecin PID'si ve rengi eşleştirilir

3.3.2. Zoom Mekanizması

Zoom Seviyesi Kavramı: Grafik başlangıçta 1.0 zoom seviyesindedir (normal boyut). Kullanıcı her "Zoom In" tıkladığında zoom seviyesi 1.3 ile çarpılır (yani %30 büyür). "Zoom Out" ile 1.3'e bölünür (%30 küçülür).

Nasıl Çalışır:

1. Kullanıcı Zoom In'e tıklar
2. Zoom seviyesi 1.0'dan 1.3'e çıkar
3. Chart genişliği (1200px × 1.3 = 1560px) olur
4. Tüm elementler yeniden pozisyonlanır

5. Scroll alanı otomatik güncellenir

Reset View: Zoom seviyesini 1.0'a, scroll pozisyonunu başa döndürür.

3.3.3. Canvas Çizim Sistemi

Gantt Bar Çizim Süreci:

Her bir gantt segmenti (süreç bloğu) şu adımlarla çizilir:

1. **Pozisyon Hesaplama:**
    - Başlangıç zamanı (start) ve bitiş zamanı (end) alınır
    - Bu değerler, canvas üzerindeki x koordinatlarına dönüştürülür
    - Scale faktörü kullanılır: x = margin + (time × scale)
2. **Gölge Efekti:**
    - Ana bardan 3 piksel sağda ve aşağıda siyah bir dikdörtgen çizilir
    - Bu, 3D derinlik hissi verir
3. **Ana Bar:**
    - Renkli dikdörtgen çizilir (süreç rengi)
    - Beyaz kenarlık eklenir (3px kalınlık)
    - Bar yüksekliği 80 pikseldir
4. **Label (Etiket):**
    - Bar ortasına süreç kimliği yazılır (örn: "P1234")
    - Beyaz, kalın yazı karakteri kullanılır
5. **Süre Bilgisi:**
    - Label'ın altına süre yazılır (örn: "(5u)")
    - Daha küçük font, gri renk
6. **Zaman İşaretleri:**
    - Her barın altına başlangıç zamanı yazılır
    - Son barın sonuna toplam süre yazılır
    - Küçük çizgi işaretleri (tick marks) eklenir

3.3.4. Renk Paleti Sistemi

15 Farklı Renk: Uygulama 15 farklı, birbirinden ayırt edilebilir renk kullanır. Renkler önceden tanımlıdır ve canlı tonlardadır (kırmızı, mavi, turkuaz, turuncu vb.).
Renk Atama Mantığı: Her süreç PID'sine göre bir renk alır. PID değeri 15'e bölünür ve kalan sayı (modulo) renk indexi olarak kullanılır. Bu sayede:

- Aynı süreç her zaman aynı rengi alır
- Renkler dengeli dağılır
- 15'ten fazla süreç varsa renkler tekrar eder

IDLE Rengi: IDLE zamanları her zaman koyu gri (#3A3A3A) ile gösterilir.

3.3.5. İnteraktif Özellikler

Pan (Kaydırma): Kullanıcı mouse ile canvas üzerine tıklayıp sürükleyebilir. Tıklama noktası kaydedilir, hareket mesafesi hesaplanır ve scroll pozisyonu güncellenir.
Mouse Wheel: Fare tekerleği ile yatay kaydırma yapılabilir. Her tekerlek hareketi scroll pozisyonunu değiştirir.
Scrollbar: Alt kısımdaki scrollbar ile manuel kaydırma mümkündür. Scrollbar boyutu, görünür alanın toplam alana oranını gösterir.

3.3.6. Grid ve Zaman Ekseni

Grid Çizgileri: Okunaklığı artırmak için dikey grid çizgileri eklenir. Bu çizgiler:

- Her 5-10 time unit'te bir çizilir
- Kesikli çizgi şeklindedir
- Açık gri renktedir
- Arka planda kalır (süreç barlarının altında)

Zaman Ekseni:

- Alt kısımda yatay bir çizgi vardır
- Belirli aralıklarla zaman değerleri yazılır
- Küçük tick işaretleri önemli noktaları gösterir

3.4. Performans Metrikleri (KPI)

3.4.1. KPI Formülleri

3.4.1.1. CPU Utilization (CPU Kullanımı)

Tanım: CPU'nun toplam zaman içinde ne kadar çalıştığını gösterir. Yüzde olarak ifade edilir.
Hesaplama Mantığı: Toplam çalışma süresinden IDLE (boş) süre çıkarılır. Kalan süre, toplam süreye bölünür ve 100 ile çarpılır [1].
Formül Açıklaması:

- Toplam Süre: Simülasyonun başından sonuna kadar geçen zaman
- IDLE Süre: Gantt Chart'taki tüm IDLE segmentlerinin toplamı
- Çalışma Süresi: Toplam - IDLE
- Kullanım: (Çalışma / Toplam) × 100

Örnek Hesaplama:

- Toplam süre: 50 birim
- IDLE süre: 5 birim
- Çalışma: 50 - 5 = 45 birim
- CPU Kullanımı: (45 / 50) × 100 = %90

İdeal Değer: %90'ın üzeri mükemmel kabul edilir. %100 teorik olarak en iyisidir ancak gerçek sistemlerde ulaşılamaz.

3.4.1.2. Throughput (İş Çıktısı)

Tanım: Birim zamanda tamamlanan süreç sayısını gösterir. Sistem verimliliğinin temel göstergesidir.
Hesaplama Mantığı: Tamamlanan toplam süreç sayısı, toplam geçen süreye bölünür.
Formül Açıklaması:

- Throughput = Tamamlanan Süreç Sayısı / Toplam Süre

Örnek Hesaplama:

- Tamamlanan süreç: 10 adet

- Toplam süre: 50 birim
- Throughput: 10 / 50 = 0.2 süreç/birim
- Yani her 5 birimde 1 süreç tamamlanıyor

İdeal Değer: Yüksek olması tercih edilir. Değer, sistem ve süreç özelliklerine göre değişir. Farklı algoritmaları karşılaştırırken önemlidir.

3.4.1.3. Turnaround Time (Tamamlanma Süresi)

Tanım: Bir sürecin sistemde geçirdiği toplam süredir. Gelişten (arrival) tamamlanmaya (completion) kadar geçen zaman [7].

Her Süreç İçin Hesaplama: Tamamlanma zamanından geliş zamanı çıkarılır.

Formül:

$$Turnaround\ Time\ =\ Completion\ Time\ -\ Arrival\ Time$$

Ortalama Hesaplama: Tüm süreçlerin turnaround time'ları toplanır ve süreç sayısına bölünür.

Örnek:

$$P1: Tamamlanma = 10, Geliş = 0\ \rightarrow\ TAT = 10$$

$$P2: Tamamlanma = 15, Geliş = 2\ \rightarrow\ TAT = 13$$

$$P3: Tamamlanma = 20, Geliş = 5\ \rightarrow\ TAT = 15$$

$$Ortalama\ TAT: (10 + 13 + 15)\ /\ 3\ =\ 12.67\ birim$$

İdeal Değer: Düşük olması tercih edilir. Kullanıcı perspektifinden önemlidir çünkü işin ne kadar sürede bittiğini gösterir.

Ne Anlama Gelir: Bir süreç sisteme girdiğinde ortalama 12.67 birim sonra tamamlanıyor demektir.

3.4.1.4. Waiting Time (Bekleme Süresi)

Tanım: Bir sürecin CPU'yu beklerken geçirdiği süredir. Turnaround time'dan burst time çıkarılarak bulunur.

Her Süreç İçin Hesaplama: Turnaround time'dan burst time çıkarılır. Kalan süre, sürecin hazır kuyruğunda beklediği süredir.

Formül:

- Waiting Time = Turnaround Time - Burst Time

Ortalama Hesaplama: Tüm süreçlerin waiting time'ları toplanır ve süreç sayısına bölünür.

Örnek:

- P1: TAT=10, Burst=5 → WT=5
- P2: TAT=13, Burst=4 → WT=9
- P3: TAT=15, Burst=6 → WT=9
- Ortalama WT: (5+9+9) / 3 = 7.67 birim

İdeal Değer: Düşük olması tercih edilir. Optimal scheduling'in temel hedefidir.

Algoritma Kalitesi: Düşük waiting time, algoritmanın iyi çalıştığını gösterir. SJF genellikle en düşük waiting time'ı sağlar.

3.4.2. KPI Görselleştirme

Kart Düzeni: Her KPI, ayrı bir kart içinde gösterilir. Kartlar yan yana 1×4 düzeninde dizilir (tek satır, 4 sütun).

Kart İçeriği:

1. **Başlık:** Metrik adı (örn: "CPU Utilization")
2. **Değer:** Büyük fontla, renkli gösterim
3. **Birim:** Yüzde, süreç/birim, veya time units

Renk Kodlama:

- CPU Utilization: Yeşil (#28a745) - iyi performans
- Throughput: Mavi (#4ECDC4) - verimlilik
- Avg Turnaround: Turuncu (#FFA07A) - dikkat
- Avg Waiting: Mor (#BB8FCE) – optimizasyon

BÖLÜM 4. EXE,SINIRLAMALAR VE GELECEK ÇALIŞMALAR

4.1. Aşama 4 Özeti

Bu aşamada projenin final testi, optimizasyonu, dağıtımı ve dokümantasyonu tamamlanmıştır:

EXE Dosyası Oluşturma:

- PyInstaller ile tek dosya (--onefile) oluşturuldu
- GUI modda (--windowed) çalışma sağlandı
- Özel ikon (--icon) eklendi
- Dosya boyutu: 28 MB (standalone, kurulum gerektirmez)

Proje Sınırlamaları:

- 30 süreç limiti
- 4 algoritma (daha fazla eklenebilir)
- Sadece Windows EXE
- Export özellikleri yok

Gelecek Önerileri:

- Kısa Vadeli: Tooltips, export, keyboard shortcuts
- Orta Vadeli: Comparison mode, yeni algoritmalar
- Uzun Vadeli: Web versiyonu, mobile app

4.2. Exe Dosyası Oluşturma Ve Dağıtım Kalitesine

Kullanılan Komut:

```
pyinstaller --onefile --windowed --icon="icon.ico"
CPUSchedulingSimulator.py
```

Komut Parametreleri Açıklaması

--onefile: Tüm bağımlılıkları tek bir .exe dosyasında toplar. Kullanıcı sadece bu dosyayı çalıştırabilir, ek kurulum gerekmez [6].

Avantajları:

- Kolay dağıtım (tek dosya paylaşımı)
- Kullanıcı karışıklığı yok

- Profesyonel görünüm

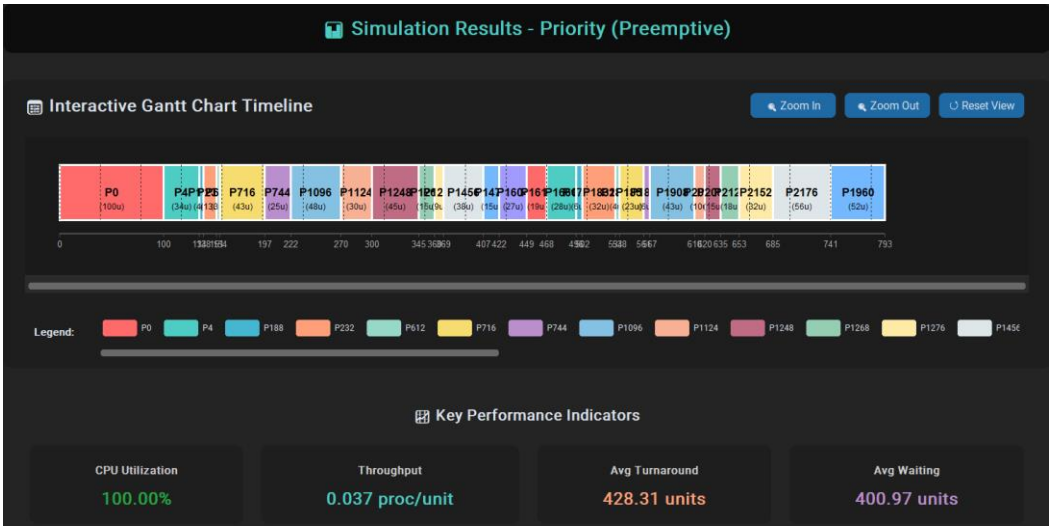
Dezavantajları:

- İlk açılış biraz yavaş (~1-2 saniye)
- Dosya boyutu büyük (~30 MB)

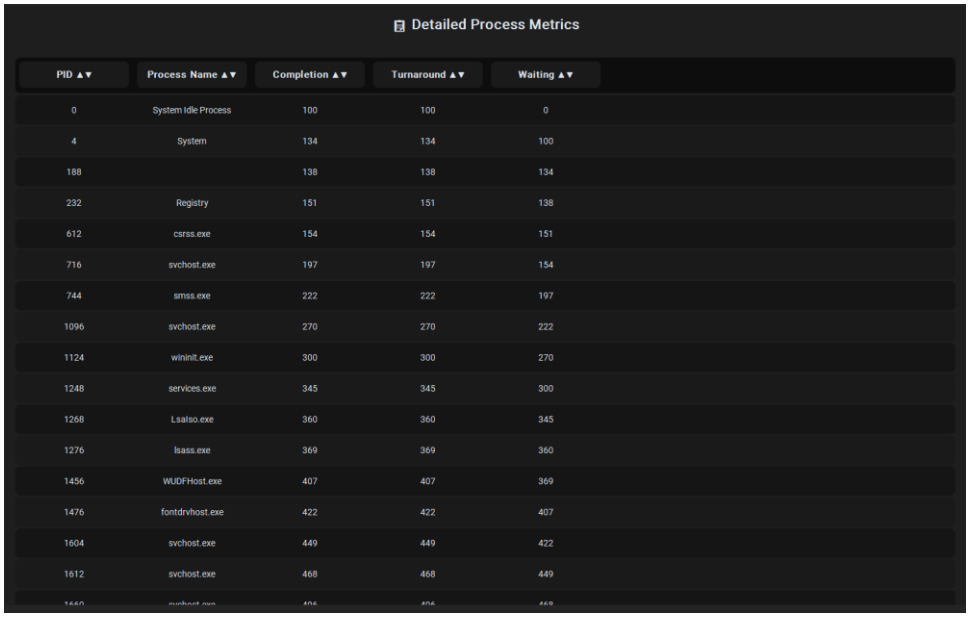
--windowed: Konsol penceresi açılmadan çalışır. Sadece GUI görünür, arka planda terminal penceresi olmaz.

Neden Önemli: Desktop uygulamaları için profesyonel görünüm sağlar. Kullanıcı sadece uygulama penceresini görür.

--icon="icon.ico": Uygulama ikonu belirlenir. Windows'ta .exe dosyasının ve çalışan uygulamanın ikonu bu olur.



Şekil 4.1. Etkileşimli Gantt Zaman Çizelgesi ve Temel Performans Göstergeleri



Şekil 4.2. Ayrıntılı Süreç Metrikleri Tablosu

4.3. Proje Kısıtları Ve Sınırlamalar

Teknik Sınırlamalar:

1. **30 Süreç Limiti:** Performans için sınırlandı, genişletilebilir
2. **Windows Odaklı:** EXE sadece Windows için, diğer platformlar kaynak koddan
3. **Gerçek Zamanlı Değil:** Statik simülasyon, canlı izleme yok
4. **Tek Kullanıcı:** Multi-user desteği yok

Fonksiyonel Sınırlamalar:

1. **4 Algoritma:** Daha fazla eklenebilir (Multilevel Queue eksik)
2. **Export Yok:** Gantt chart PNG/PDF export özelliği yok
3. **Karşılaştırma Modu Yok:** 2+ algoritma yan yana gösterim yok
4. **Dil Desteği:** Sadece İngilizce arayüz

4.4. Öneriler Ve Gelecekteki Çalışmalar

4.4.1. Kısa Vadeli Öneriler

Kullanıcı Deneyimi İyileştirmeleri:

1. **Tooltips Ekleme:**
    - Her buton üzerine gelince açıklama göster
    - Algoritma seçiminde kısa açıklamalar
    - KPI metriklerinin anlamını göster
2. **Keyboard Shortcuts:**
    - Ctrl+F: Fetch processes
    - Ctrl+R: Run simulation
    - Ctrl+X: Reset
3. **Hata Mesajlarını İyileştirme:**
    - Daha açıklayıcı mesajlar
    - Çözüm önerileri ekle
    - Dialog yerine notification sistemi

Export Özellikleri:

1. **Gantt Chart Export:**

- PNG formatında kaydetme
- Yüksek çözünürlük seçeneği
- Otomatik dosya adı (algoritma + tarih)

2. **Sonuç Raporu Export:**
    - PDF rapor oluşturma
    - Tüm metrikleri içeren
    - Gantt chart dahil
3. **CSV Export:**
    - Süreç listesi
    - Sonuç metrikleri
    - Excel uyumlu

4.4.2. Orta Vadeli Öneriler

Yeni Özellikler:

1. **Comparison Mode (Karşılaştırma Modu):**
    - 2 veya daha fazla algoritmayı yan yana çalıştır
    - Gantt chart'ları alt alta göster
    - KPI'ları karşılaştır
    - Hangi algoritma ne zaman daha iyi?
2. **Custom Process Entry:**
    - Kullanıcı manuel süreç girebilsin
    - Burst time, arrival time, priority belirleyebilsin
    - Önceden hazır senaryolar (templates)
    - Test senaryoları oluşturma
3. **Statistics Dashboard:**
    - Detaylı istatistikler
    - Histogram: Waiting time dağılımı
    - Scatter plot: Burst vs Waiting
    - Algorithm efficiency scorer

Algoritma Eklentileri:

1. **Multilevel Queue Scheduling:**

- 3 farklı kuyruk seviyesi
- Foreground, Background ayrımı
- Priority inheritance

2. **Multilevel Feedback Queue:**
    - Dinamik öncelik değişimi
    - Aging mekanizması
    - Adaptive time quantum
3. **Earliest Deadline First (EDF):**
    - Real-time scheduling
    - Deadline tanımı
    - Miss rate hesaplama

4.4.3. Uzun Vadeli Öneriler

Platform Genişletme:

1. **Web-Based Version:**
    - React veya Vue.js frontend
    - Python Flask/FastAPI backend
    - Online kullanım, kurulum yok
    - Tarayıcıdan erişim
2. **Mobile Application:**
    - Android ve iOS
    - React Native veya Flutter
    - Basitleştirilmiş arayüz
    - Educational focus
3. **Cloud Integration:**
    - Kullanıcı hesapları
    - Simülasyon sonuçlarını kaydetme
    - Paylaşım özellikleri
    - Leaderboard (en iyi schedulers)

AI ve Makine Öğrenmesi:

1. **Smart Scheduling Recommendation:**

- Süreç özelliklerine göre algoritma önerisi
- ML modeli ile tahmin
- "Bu süreçler için SJF en uygun" gibi

2. **Pattern Recognition:**
    - Süreç davranış analizi
    - Benzer süreçleri gruplama
    - Anomali tespiti

4.4.4. Eğitim ve Akademik Öneriler

Ders Entegrasyonu:

1. **Lab Modülü Hazırlama:**
    - 4 haftalık lab planı
    - Her hafta bir algoritma
    - Öğrenci ödevleri
    - Değerlendirme rubriği
2. **Quiz ve Testler:**
    - Uygulama içi quiz sistemi
    - Algoritma bilgisi testi
    - Sonuç yorumlama soruları
    - Otomatik puanlama
3. **Öğretmen Paneli:**
    - Öğrenci sonuçlarını izleme
    - İstatistiksel analiz
    - Sınıf performansı raporu

Araştırma Olanakları:

1. **Veri Toplama (Opsiyonel):**
    - Kullanıcı izni ile anonim istatistik
    - Hangi algoritmalar daha çok tercih ediliyor?
    - Ortalama kullanım süresi
    - Araştırma makalesi potansiyeli
2. **Benchmark Dataset:**

- Standart test senaryoları
- Algoritma karşılaştırma için
- Akademik yayınlarda referans

BÖLÜM 5. SONUÇLAR VE ÖNERİLER

Bu bitirme çalışmasında, gerçek zamanlı süreç verilerini kullanarak FCFS, önişlemeli SJF, önişlemeli Öncelik ve Round Robin algoritmalarını simüle eden bir CPU zamanlama aracı geliştirilmiştir. Uygulama, etkileşimli Gantt şeması, ayrıntılı süreç tablosu ve temel performans göstergeleri (CPU kullanım oranı, throughput, ortalama dönüş ve bekleme süreleri) ile farklı algoritmaların aynı süreç kümesi üzerindeki davranışlarını görsel ve anlaşılır biçimde ortaya koymaktadır. Elde edilen sonuçlar, algoritmaların özellikle bekleme ve dönüş süreleri açısından birbirlerinden belirgin şekilde ayrıldığını ve geliştirilen yazılımın eğitim amaçlı kullanılabilir bir simülatör niteliği taşıdığını göstermektedir.

Çalışmanın devamında, ek zamanlama algoritmalarının (örneğin çok seviyeli kuyruk ve geri beslemeli kuyruk) sisteme dâhil edilmesi, birden fazla algoritmanın aynı anda karşılaştırılabildiği bir modun eklenmesi ve Gantt şeması ile metriklerin dışa aktarılabilmesi (PNG, CSV, PDF) önerilmektedir. Ayrıca, projenin web veya mobil ortama taşınması ve farklı süreç profilleri için uygun algoritma seçimini destekleyecek basit makine öğrenmesi tabanlı öneri mekanizmalarının eklenmesi, hem kullanım alanını genişletecek hem de çalışmanın akademik katkısını artıracaktır.

KAYNAKLAR

[1] Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th Edition). Wiley.

[2] Tanenbaum, A. S., & Bos, H. (2014). Modern Operating Systems (4th Edition). Pearson.

[3] Stallings, W. (2018). Operating Systems: Internals and Design Principles (9th Edition). Pearson.

[4] Tomschimansky, T. (2024). CustomTkinter Documentation. https://customtkinter.tomschimansky.com/

[5] Rodola, G. (2024). psutil Documentation. https://psutil.readthedocs.io/

[6] PyInstaller Development Team. (2024). PyInstaller Manual. https://pyinstaller.org/

[7] GeeksforGeeks. (2024). "CPU Scheduling Algorithms". https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/

ÖZGEÇMİŞ

Younes Rahebi, 16.09.2000 tarihinde Gonbad-e Kavus'ta doğdu. 2022 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'nde lisans eğitimine başlamış olup, eğitimine son sınıf öğrencisi olarak devam etmektedir. Lisans sürecinde veri bilimi ve makine öğrenmesi alanlarına yönelmiş, Python ile veri analizi ve model geliştirme üzerine çeşitli akademik ve kişisel projeler yürütmüştür. 2024-2025 yılları arasında Kadir Has Üniversitesi'nde TÜBİTAK destekli bir projede araştırma stajyeri olarak görev almış; literatür taraması, teknik dokümantasyon ve makale hazırlık çalışmalarına katkı sunmuştur. Ayrıca Artes Solution firmasında yazılım geliştirme ve donanım/IoT stajları yaparak .NET Core MVC tabanlı web uygulamaları geliştirmiş, Arduino tabanlı prototipler ve sensör tabanlı gömülü sistemler üzerinde çalışmıştır.

BSM 401 BİLGİSAYAR MÜHENDİSLİĞİ TASARIMI
DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI

KONU :
ÖĞRENCİLER (Öğrenci No/AD/SOYAD):

Değerlendirme Konusu	İstenenler	Not Aralığı	Not
Yazılı Çalışma			
Çalışma klavuza uygun olarak hazırlanmış mı?	x	0-5	
Teknik Yönden			
Problemin tanımı yapılmış mı?	x	0-5	
Geliştirilecek yazılımın/donanımın mimarisini içeren blok şeması (yazılımlar için veri akış şeması (dfd) da olabilir) çizilerek açıklanmış mı?			
Blok şemadaki birimler arasındaki bilgi akışına ait model/gösterim var mı?			
Yazılımın gereksinim listesi oluşturulmuş mu?			
Kullanılan/kullanılması düşünülen araçlar/teknolojiler anlatılmış mı?			
Donanımların programlanması/konfigürasyonu için yazılım gereksinimleri belirtilmiş mi?			
UML ile modelleme yapılmış mı?			
Veritabanları kullanılmış ise kavramsal model çıkarılmış mı? (Varlık ilişki modeli, noSQL kavramsal modelleri v.b.)			
Projeye yönelik iş-zaman çizelgesi çıkarılarak maliyet analizi yapılmış mı?			
Donanım bileşenlerinin maliyet analizi (prototip-adetli seri üretim vb.) çıkarılmış mı?			
Donanım için gerekli enerji analizi (minimum-uyku-aktif-maksimum) yapılmış mı?			
Grup çalışmalarında grup üyelerinin görev tanımları verilmiş mi (iş-zaman çizelgesinde belirtilebilir)?			
Sürüm denetim sistemi (Version Control System; Git, Subversion v.s.) kullanılmış mı?			
Sistemin genel testi için uygulanan metotlar ve iyileştirme süreçlerinin dökümü verilmiş mi?			
Yazılımın sızma testi yapılmış mı?			
Performans testi yapılmış mı?			
Tasarımın uygulamasında ortaya çıkan uyumsuzluklar ve aksaklıklar belirtilerek çözüm yöntemleri tartışılmış mı?			
Yapılan işlerin zorluk derecesi?	x	0-25	
Sözlü Sınav			
Yapılan sunum başarılı mı?	x	0-5	
Soruları yanıtlama yetkinliği?	x	0-20	
Devam Durumu			
Öğrenci dönem içerisindeki raporlarını düzenli olarak hazırladı mı?	x	0-5	
Diğer Maddeler			
Toplam			

DANIŞMAN :
DANIŞMAN İMZASI: