



T.C.

**Kocaeli Sağlık ve Teknoloji Üniversitesi
Mühendislik ve Doğa Bilimleri Fakültesi**
Yazılım Mühendisliği
Proje 2

Ders: Programlama Lab 2

Proje: Bagaj Güvenlik Simülatörü

Hazırlayanlar: 230502054 Muhammed Emir Sarı,

230502054 Sude Çakır

Öğretim Görevlisi: Dr. Öğr. Üyesi Fulya Akdeniz

Ödev son teslim tarihi: 09.05.25 Ödev teslim tarihi: 09.05.25

Bagaj Güvenlik Simülatörü

1. GİRİŞ

1.1 Projenin Amacı

Havalimanlarında bagaj yönetimi, yolcu memnuniyetini ve operasyonel verimliliği doğrudan etkileyen kritik süreçlerden biridir. Bu projenin temel amacı, havaalanındaki bagaj işleme sürecini yazılımsal olarak modelleyip bir simülasyon sistemi oluşturmaktır. Yazılım, yolcuların havalimanına girişinden başlayarak bagajların teslim alınması, sıralanması, taşınması ve yönlendirilmesi gibi adımları veri yapıları ve nesne yönelimli yaklaşımalar ile simüle eder.

Simülasyon sayesinde, gerçek dünyadaki operasyonların küçük ölçekte modellenmesi ve test edilmesi mümkün hale gelmiştir. Ayrıca kullanıcı arayüzü ile etkileşim sağlanarak sistemin durumu görsel olarak izlenebilir kılmıştır.

Projede Gerçekleştirilen Hedefler:

- Yolcu ve bagaj ilişkilerinin nesne yönelimli yapılarla modellenmesi
- Kuyruk, yiğin ve bağlı liste veri yapılarının uygun yerlerde kullanılması
- Olasılık temelli bagaj kaybı ve işlem gecikmesi senaryolarının modellenmesi
- PyQt5 ile sade, işlevsel ve görsel bir kullanıcı arayüzü oluşturulması
- Yazılımın modüler yapıda geliştirilerek bakım ve genişletilebilirliğinin sağlanması

2. GEREKSİNİM ANALİZİ

2.1 Arayüz Gereksinimleri

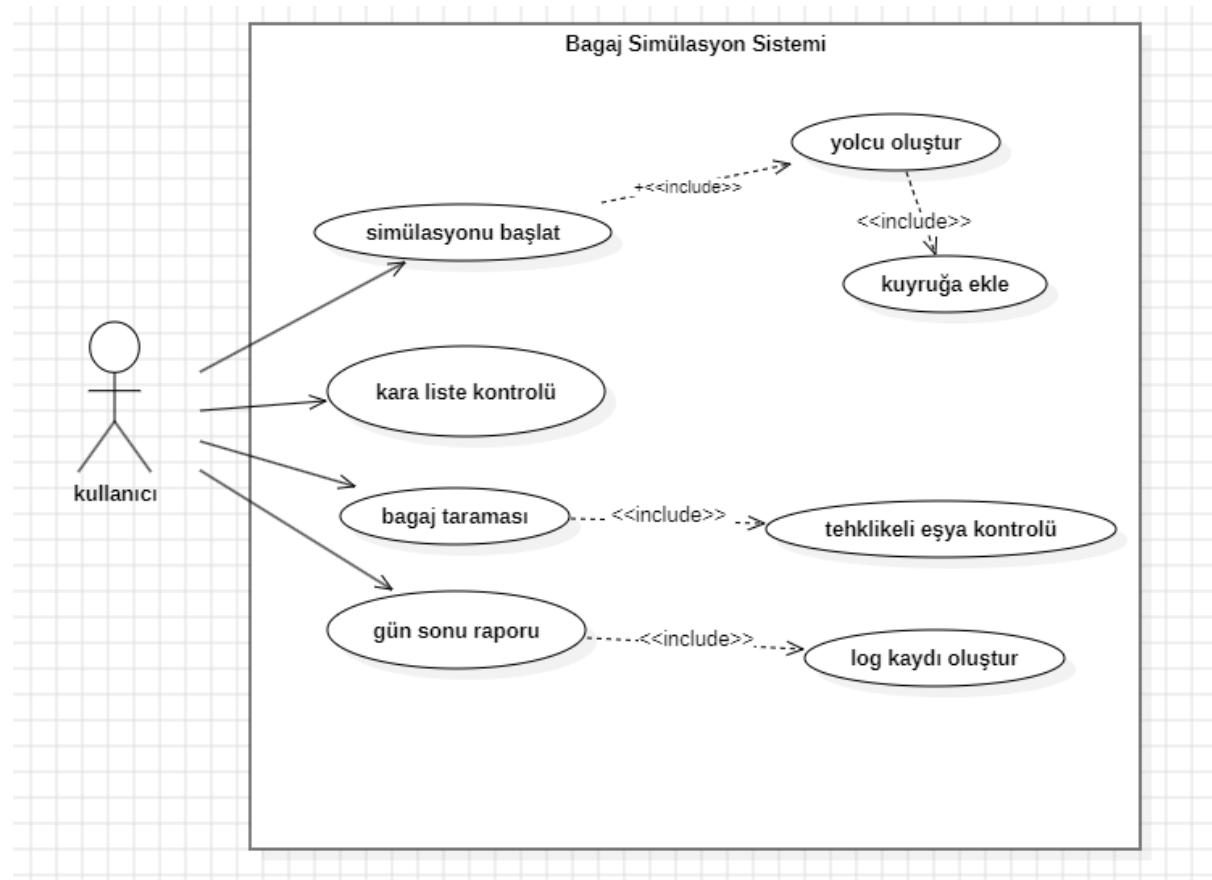
- Kullanıcı Arayüzü Özellikleri:**
 - “Başlat” ve “Durdur” düğmeleri ile simülasyon kontrolü
 - Yolcu ve bagaj bilgilerini listeleyen panel
 - Olay günlüğü ile anlık durum takibi
 - Minimalist ve işlevsel PyQt5 temelli tasarım
- Donanım Gereksinimleri:**

- Python 3.10+ kurulu herhangi bir bilgisayar
- PyQt5 kütüphanesi yüklü olması
- Standart işlemci ve RAM yeterlidir

2.2 Fonksiyonel Gereksinimler

- Yolcu nesnelerinin rastgele oluşturulması ve bagajla ilişkilendirilmesi
- FIFO mantığıyla bagajların sıraya alınması (queue)
- Öncelikli işlem gerektiren bagajlar için LIFO yaklaşımı (stack)
- Bagajların bağlı liste içinde işlenmesi ve geçmişinin tutulması
- Bagaj kaybı vb. durumların olasılıklu şekilde modellenmesi (olasilik.py)
- Simülasyon akışının arayüz üzerinden başlatılması ve durdurulması

2.3 Use-Case Diyagramı



3. TASARIM

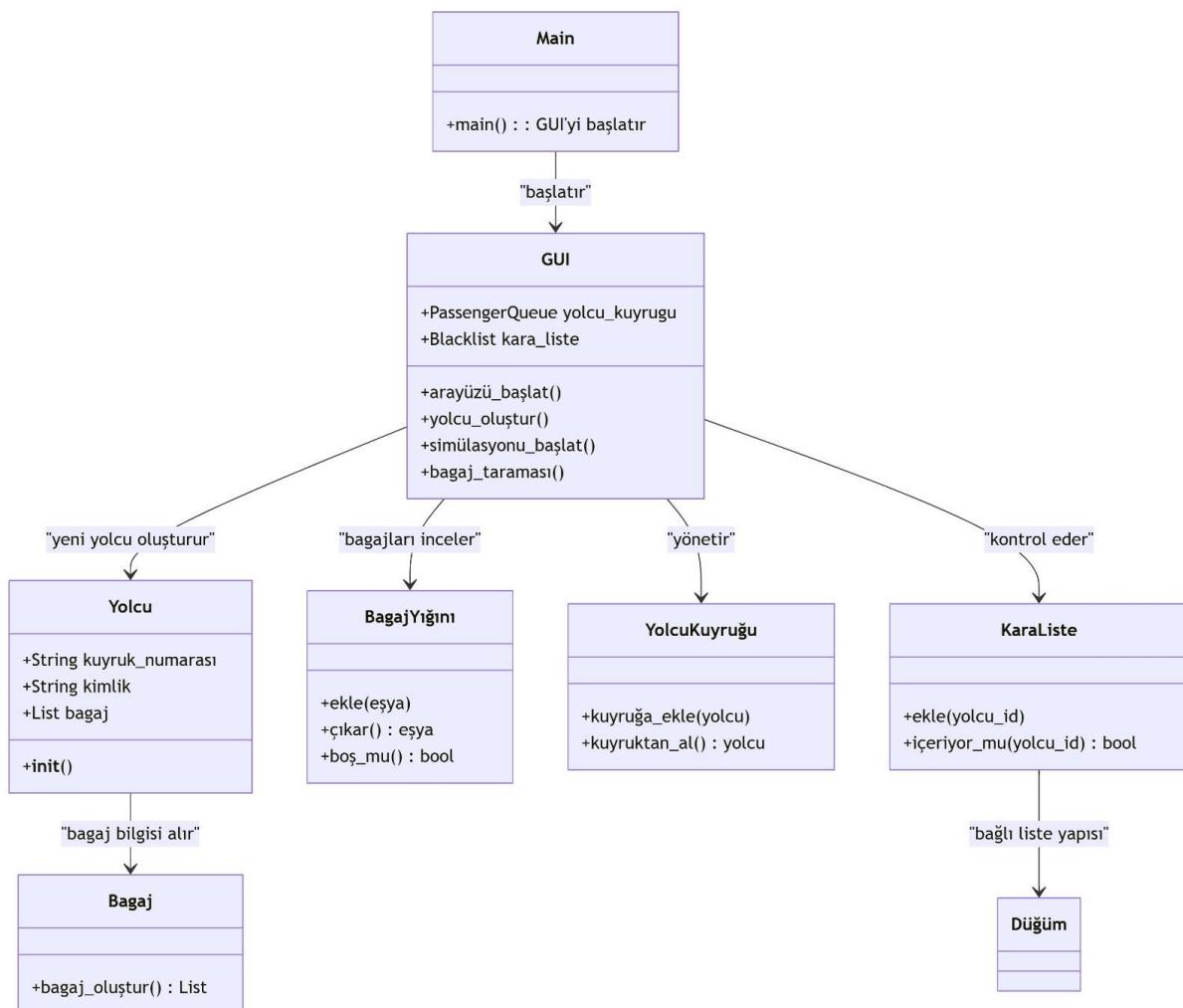
3.1 Mimari Tasarım

Projede temel yazılım mühendisliği prensiplerine bağlı kalınarak modüler bir mimari kurgulanmıştır:

- `models/`: Yolcu ve bagaj sınıfları ile temel veri yapıları
- `utils/`: Olasılık hesaplamalarının yapıldığı destek modülü
- `gui.py`: Arayüz bileşenlerinin oluşturulup gösterildiği PyQt5 ekranı
- `main.py`: Simülasyonun başlatıldığı ve modüller arası bağlantının sağlandığı ana dosya

Yapının temel avantajı, her modülün tek bir sorumluluğa sahip olması ve birim testlerin kolayca yapılabilmesidir.

MODÜL DİYAGRAMI:



3.2 Kullanılan Teknolojiler

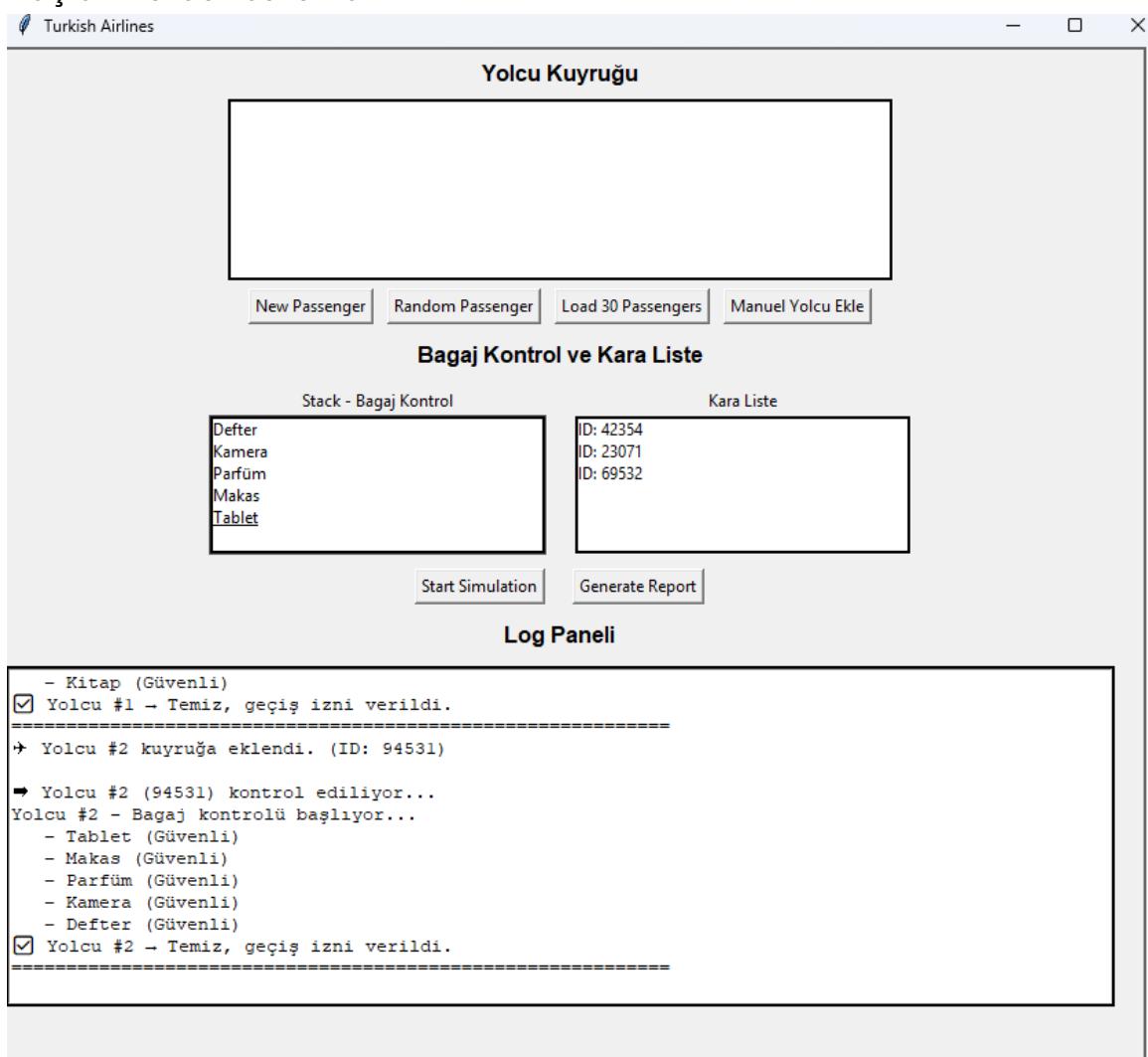
- **Programlama Dili:** Python 3.10
- **Kütüphaneler:** PyQt5 (arayüz), random (rastgelelik), time, threading
- **Dosya Yapısı:** Katmanlı ve modüler Python dizin yapısı

3.3 Veritabanı Tasarımı

Bu projede veritabanı kullanılmamıştır. Tüm işlemler çalışma zamanında (runtime) bellekte gerçekleşmektedir.

3.4 Kullanıcı Arayüzü Tasarımı

- Başlat/Durdur butonları



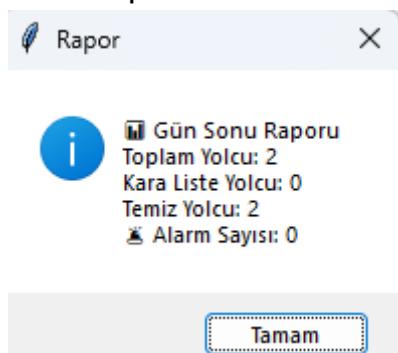
- Yolcu ve bagaj bilgilerini gösteren bilgi ekranı

```

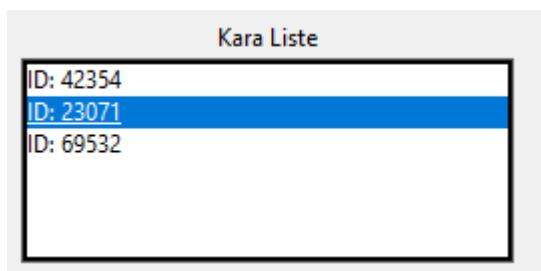
- Defter (Güvenli)
 Yolcu #2 → Temiz, geçiş izni verildi.
=====
✉ Rastgele yolcu eklendi: Yolcu #782 (ID: 54078)

→ Yolcu #782 (54078) kontrol ediliyor...
Yolcu #782 - Bagaj kontrolü başlıyor...
- Parfüm (Güvenli)
- Makas (Güvenli)
- Su şişesi (Güvenli)
- Defter (Güvenli)
- Tablet (Güvenli)
 Yolcu #782 → Temiz, geçiş izni verildi.
=====
```

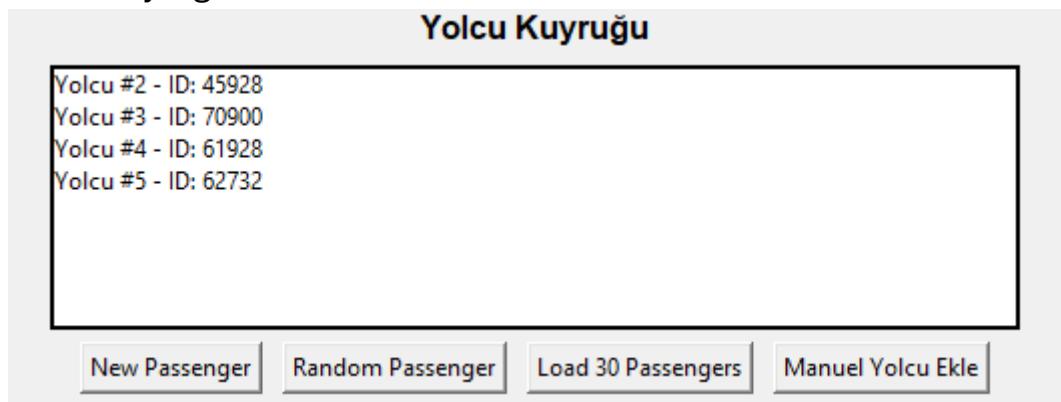
- Durum penceresi ile anlık olay günlüğü



- Arayüz `gui.py` içinde tanımlanmıştır, simülasyon `main.py` çalıştırılarak başlatılır
- Kara liste:



- Yolcu kuyruğu:



- Bagaj kontrol:



4. UYGULAMA

4.1 Kodlanan Bileşenlerin Açıklamaları

- `yolcu.py`: Yolcu kimliği, ismi ve bagaj sayısı gibi özelliklere sahip sınıf

```
import random
from models.bagaj import generate_bagaj

5 usages
class Passenger:
    def __init__(self, sira, yolcu_id=None):
        self.queue_number = f"Yolcu #{sira}"
        self.id = yolcu_id if yolcu_id else self._generate_id()
        self.bagaj = generate_bagaj()

    1 usage
    def _generate_id(self):
        return str(random.randint(a: 10000, b: 99999))

    def __str__(self):
        return f"{self.queue_number} - ID: {self.id}"
```

- `bagaj.py`: Bagaj nesnesi; ağırlık, ID, yolcu ilişkisi gibi bilgiler içerir

```
import random

temiz_esyalar = ["Laptop", "Kitap", "Su şisesi", "Makas", "Şarj Cihazı", "Parfüm", "Kamera", "Defter", "Tablet"]
tehlikeli_esyalar = ["Bıçak", "Silah", "Patlayıcı", "Çakı", "El Bombası"]

4 usages
def generate_bagaj():
    bagaj = random.sample(temiz_esyalar, k=random.randint(a: 4, b: 6))
    if random.random() < 0.1:
        bagaj.append(random.choice(tehlikeli_esyalar))
    random.shuffle(bagaj)
    return bagaj
```

- queue.py: FIFO sıralamasıyla bagajların teslim sürecine alınmasını sağlar

```
from collections import deque
💡
2 usages
class PassengerQueue:
    def __init__(self):
        self.queue = deque()

    3 usages
    def enqueue(self, passenger):
        self.queue.append(passenger)

    1 usage
    def dequeue(self):
        return self.queue.popleft() if self.queue else None

    1 usage
    def is_empty(self):
        return len(self.queue) == 0
```

- `stack.py`: Acele işlemler için bagajları LIFO yapısında işler

```
2 usages
class BaggageStack:
    def __init__(self):
        self.stack = []

1 usage
def push(self, item):
    self.stack.append(item)

1 usage
def pop(self):
    return self.stack.pop() if self.stack else None

2 usages
def is_empty(self):
    return len(self.stack) == 0

6 usages (6 dynamic)
def size(self):
    return len(self.stack)

def peek(self):
    return self.stack[-1] if not self.is_empty() else None
```

- linkedlist.py: Tüm işlem geçmişini bağlı liste ile takip eder

```

class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

2 usages
class Blacklist:
    def __init__(self):
        self.head = None

2 usages
def add(self, yolcu_id):
    yeni = Node(yolcu_id)
    yeni.next = self.head
    self.head = yeni

2 usages
def contains(self, yolcu_id):
    current = self.head
    while current:
        if current.data == yolcu_id:
            return True
        current = current.next
    return False

```

- olasilik.py: Rastgelelik esasına dayalı bagaj kaybı, bekleme gibi durumları üretir

```

yasaklı_esyalar = ["bıçak", "çakı", "silah", "patlayıcı", "el bombası"]
💡
2 usages
def esya_tehlikeli_mi(esya):
    return esya.lower() in yasaklı_esyalar

```

- `gui.py`: PyQt5 arayüzüünü içerir, kullanıcı etkileşimlerini yönetir

```

import pygame
import tkinter as tk
from tkinter import messagebox, Scrollbar, Canvas, Frame
from models.stack import BaggageStack
from models.yolcu import Passenger
from models.queue import PassengerQueue
from models.linkedlist import Blacklist
from utils.olasilik import esya_tehlikeli_mi

# === GUI Global ===
yolcu_kuyruku_gui = PassengerQueue()
kara_liste_gui = Blacklist()
kara_liste_idler = ["42354", "23071", "69532"]

for kid in kara_liste_idler:
    kara_liste_gui.add(kid)

yolcu_sayaci = 1
toplam_yolcu = 0
kara_liste_yolcu = 0
temiz_yolcu = 0
alarm_sayisi = 0

16 usages
def log_yaz(mesaj):
    log_text.config(state='normal')
    log_text.insert(tk.END, mesaj + "\n")
    log_text.see(tk.END)
    log_text.config(state='disabled')

2 usages
def yolcu_olustur():
    global yolcu_sayaci, toplam_yolcu
    yolcu = Passenger(sira=yolcu_sayaci)
    yolcu_sayaci += 1
    toplam_yolcu += 1
    yolcu_kuyruku_gui.enqueue(yolcu)
    queue_listbox.insert(tk.END, *elements: str(yolcu))

```

- `main.py`: Simülasyonun başlangıç noktası, tüm modülleri yönetir

4.2 Görev Dağılımı

bu projedeki görev dağılımı rapor muhammed emir sarı
proje yapımınınında çoğunuğu sude çakır tarafından geliştirilmiştir.

4.3 Karşılaşılan Zorluklar ve Çözüm Yöntemleri

- PyQt5 arayüzünde arka plan işlemleri yönetmek başlangıçta sorun yarattı → QTimer ve threading ile paralel işlemler kontrol altına alındı
- Veri yapıları arası veri transferinde senkronizasyon sorunları oluştu → Nesne geçişlerinde referans takibi ile çözüldü
- Olasılık hesaplamalarının dengesiz sonuçlar üretmesi → Daha dengeli dağılımlar için random.choices parametreli kullanıldı

4.4 Proje İsterlerine Göre Eksik Yönler

- Bagaj konveyörünün görsel temsili yer almamaktadır
- Verilerin kalıcı olarak kayıt altına alınması (örneğin log dosyası) henüz uygulanmamıştır
- Yolcu sayısı veya bagaj miktarı gibi kullanıcı ayarları GUI üzerinden değiştirilememektedir

5. TEST VE DOĞRULAMA

5.1 Yazılımın Test Süreci

- Her sınıfın ayrı ayrı test edilmesiyle başlamıştır
- Queue ve Stack yapılarına yönelik ekleme-çıkarma işlemleri doğrulanmıştır
- Bagajların doğru yolcuya atanıp atanmadığı kontrol edilmiştir
- GUI, başlat/durdur tepkileri ve bilgi güncellemleri açısından test edilmiştir
- olasilik.py, farklı senaryolarla rastgelelik istikrarı açısından test edilmiştir

5.2 Yazılımın Doğrulanması

Sorunsuz Çalışan Bileşenler:

- Veri yapılarının tümü
- Olasılık modülü
- Arayüz ve kullanıcı etkileşimleri
- Simülasyon başlangıcı ve akışı

Geliştirilmeye Açık Bileşenler:

- Kalıcı veri kayıt sistemi
- GUI'de dinamik kullanıcı ayarları
- İleri düzey istatistiksel analiz (örneğin simülasyon sonunda toplam bekleme süresi)

SONUÇ

Geliştirilen bu simülasyon yazılımı, havaalanı bagaj yönetim sürecinin temel yapısını başarıyla modellemekte ve işlemleri anlaşılabilir bir biçimde kullanıcıya sunmaktadır. Veri yapılarının yerinde kullanımı ve modüler mimari sayesinde sistem esnek, test edilebilir ve genişletilebilir hale gelmiştir. Bu çalışma, hem yazılım mühendisliği prensiplerini pekiştirmeye hem de gerçek dünya problemlerinin dijital ortamda modellenmesine dair önemli bir örnek oluşturmaktadır.