T.C
KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ

PROJE KONUSU:BAGAJ GÜVENLİK SİMÜLATORU

ÖĞRENCİ ADI:YERDINAT ALIKHAN,ZEYNEP ÇALIK
ÖĞRENCİ NUMARASI:220502050,220501008

DERS SORUMLUSU:
PROF. DR./DR. ÖĞR. ÜYESİ FULYA AKDENİZ

TARİH:07.05.2025

1 GİRİŞ

Projenin amacı

Bu proje, havalimanı güvenlik kontrol noktalarında gerçekleşen bagaj tarama ve yolcu risk analizi süreçlerini **yazılımsal bir simülasyon** ile modellemeyi amaçlamaktadır. Temel hedef, gerçek dünya senaryolarını kullanarak:

- **Veri yapılarının** (kuyruk, yığın, bağlı liste) pratik uygulamalarını göstermek,
- **Olasılık tabanlı karar mekanizmalarını** entegre etmek,
- **Etkileşimli bir grafik arayüz** (GUI) ile simülasyonun adımlarını görselleştirmek,
- **Test odaklı geliştirme** ile sistemin güvenilirliğini sağlamaktır.

Projede Gerçekleştirilmesi Beklenenler

1. **Veri Yapılarının Uygulanması:**
    - **Kuyruk (Queue):** Yolcuların FIFO (İlk Giren İlk Çıkar) mantığıyla sıralanması.
    - **Yığın (Stack):** Şüpheli bagajların LIFO (Son Giren İlk Çıkar) prensibiyle incelenmesi.
    - **Bağlı Liste (Linked List):** Kara listedeki yolcu ID'lerinin etkin şekilde saklanması ve taranması.
2. **Algoritmik İşlevler:**
    - **Rastgele Bagaj Oluşturma:** Her yolcuya 5-10 arası eşya atanması ve %10 olasılıkla tehlikeli eşya eklenmesi.
    - **Kara Liste Kontrolü:** Bağlı liste üzerinde lineer arama ile riskli yolcuların tespiti.
    - **Tehlikeli Eşya Tespiti:** Statik bir listeden yasaklı eşyaların kontrol edilmesi.
3. **Grafiksel Arayüz (GUI) Geliştirme:**
    - **Paneller:** Kuyruk, yığın, kara liste ve log bileşenlerinin gerçek zamanlı görselleştirilmesi.
    - **Etkileşim:** Kullanıcının yeni yolcu ekleme, simülasyonu adımlama ve rapor alma işlemlerini yönetebilmesi.
    - **Görsel Geri Bildirim:** Alarm durumunda yığın panelinin renk değiştirmesi, kara listede vurgulama.
4. **Test ve Doğrulama:**
    - **Birim Testleri:** Yolcu oluşturma, kara liste arama, tehlikeli eşya tespiti gibi kritik fonksiyonların test edilmesi.

- **Entegrasyon Testleri:** Tüm simülasyon adımlarının senaryo bazlı çalıştırılması (örneğin, kara listeli bir yolcunun yakalanması).
- **Raporlama:** İstatistiklerin CSV dosyasına aktarılması ve GUI üzerinde özetlenmesi.

5. **Belgelendirme ve Kullanıcı Deneyimi:**
   - **Kod Açıklamaları:** Fonksiyonların ve sınıfların detaylı dokümantasyonu.
   - **Kullanıcı Kılavuzu:** GUI'nin nasıl kullanılacağını açıklayan basit talimatlar.

2 GEREKSİNİM ANALİZİ

Arayüz gereksinimleri

1 Paneller ve Görsel Bileşenler:

QueuePanel (Yolcu Kuyruğu):

FIFO mantığıyla sıralanan yolcuları listeleyen bir Listbox.

Yolcu ID'leri ve bagaj bilgileri görüntülenmeli.

StackPanel (Şüpheli Bagaj Yığını):

LIFO mantığıyla tehlikeli eşyaları gösteren bir Listbox.

Alarm durumunda panel kırmızı renge dönüşmeli.

LinkedListPanel (Kara Liste):

Kara listedeki yolcu ID'lerini bağlı liste yapısıyla görselleştiren bir Listbox.

Eşleşen ID'ler sarı renkle vurgulanmalı.

LogPanel (Olay Kayıtları):

Tüm simülasyon adımlarını kronolojik olarak kaydeden kaydırılabilir bir metin alanı.

Log seviyelerine göre renkli etiketler (info: siyah, warning: turuncu,

danger: kırmızı).

ControlPanel (Kontrol Butonları):

"Yeni Yolcu Ekle", "Simülasyon Adımı Çalıştır", "Tüm Kuyruğu İşle", "Yığını Tara", "Rapor Göster" butonları.

2 Etkileşim ve İşlevsellik:

Yeni Yolcu Ekleme:

Rastgele bagajlı bir yolcu oluşturur ve kuyruğa ekler.

QueuePanel otomatik güncellenir.

Simülasyon Adımı:

Kuyruktan bir yolcu alır, kara liste ve bagaj kontrolü yapar.

Sonuçlar LogPanel'e yazılır, StackPanel renk değiştirir.

Tam Simülasyon:

Tüm kuyruğu otomatik işler, adımlar arasında 0.1 saniye bekleme ekler.

Görsel Geri Bildirim:

Tehlikeli eşya tespitinde StackPanel kırmızıya döner.

Kara liste eşleşmesinde ilgili ID sarı renkle vurgulanır.

3 **Kullanıcı Deneyimi:**

Basit ve Sezgisel Tasarım:

Paneller net bir şekilde ayrılmış, butonlar anlaşılır etiketlere sahip.

Gerçek Zamanlı Güncelleme:

Her işlem sonrası ilgili paneller anında yenilenir.

Hata Yönetimi:

Boş kuyruk/yığın durumunda kullanıcıya bilgilendirme mesajı gösterilir.

Fonksiyonel gereksinimler

1. Yolcu ve Bagaj Yönetimi

- **Yolcu Oluşturma:**
  - Rastgele bir yolcu ID'si (örneğin: `Yolcu #23`) oluşturulmalı.
  - Her yolcuya **5-10 adet** rastgele eşya atanmalı.
  - Bagajın **%10 olasılıkla** en az bir tehlikeli eşya içermesi sağlanmalı.
  - Oluşturulan yolcu, kuyruk (Queue) yapısına otomatik eklenmeli.
- **Kuyruk İşlemleri:**
  - Yolcular **FIFO (İlk Giren İlk Çıkar)** mantığıyla işlenmeli.
  - Kuyruk boşken işlem yapılmaya çalışıldığında kullanıcıya uyarı verilmeli.

2. Güvenlik Kontrol Süreçleri

- **Kara Liste Kontrolü:**
  - Her yolcunun ID'si, bağlı liste (LinkedList) üzerinde **lineer arama** ile taranmalı.
  - Eşleşme durumunda yolcuya "yüksek risk" etiketi eklenmeli ve bagajı doğrudan yığına (Stack) gönderilmeli.
- **Bagaj Tarama:**
  - Bagajdaki her eşya, `esya_tehlikeli_mi()` fonksiyonu ile kontrol edilmeli.
  - Tehlikeli eşya tespit edilirse:
    - Bagajın tamamı yığına (Stack) aktarılmalı.
    - GUI'de **StackPanel** kırmızı renkle vurgulanmalı.
    - Log paneline **ALARM** mesajı eklenmeli.
- **Yığın (Stack) İncelemesi:**
  - Yığındaki eşyalar **LIFO (Son Giren İlk Çıkar)** mantığıyla tek tek çıkarılmalı.
  - Çıkarılan her eşya, tehlikeli olup olmadığına göre loglanmalı.

3. Raporlama ve İstatistikler

- **Gün Sonu Raporu:**
  - Toplam işlenen yolcu sayısı.
  - Alarm verilen bagaj sayısı.
  - Kara listede yakalanan yolcu sayısı.

- Temiz geçiş yapan yolcu sayısı.
- Rapor, **CSV dosyasına** otomatik kaydedilmeli ve GUI'de görüntülenmeli.

4. Kullanıcı Arayüzü (GUI) İşlevleri

- **Gerçek Zamanlı Güncelleme:**
  - QueuePanel, StackPanel ve LinkedListPanel, her işlem sonrası anında güncellenmeli.
  - LogPanel'de tüm olaylar **kronolojik** ve **renkli etiketlerle** (info, warning, danger) listelenmeli.
- **Etkileşimli Kontroller:**
  - "Yeni Yolcu Ekle" butonu ile manuel yolcu eklenebilmeli.
  - "Tüm Kuyruğu İşle" butonu ile simülasyon otomatik tamamlanmalı.
  - "Rapor Göster" butonu ile istatistikler ayrı bir pencerede görüntülenmeli.

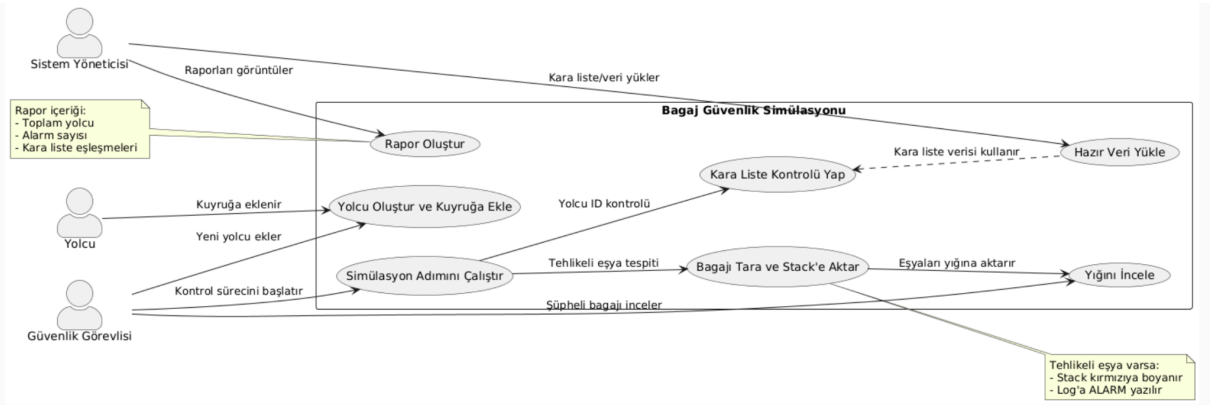
5. Test ve Doğrulama

- **Hazır Veri Yükleme:**
  - "Hazır Veri Yükle" butonu ile **30 yolcu** otomatik olarak kuyruğa eklenmeli.
- **Edge Case Testleri:**
  - Kara listede olmayan ancak tehlikeli eşya taşıyan yolcuların tespiti.
  - Boş kuyruk/yığın durumunda hata mesajlarının doğru çalışması.

6. Hata Yönetimi

- **Geçersiz Dosya Yükleme:**
  - Kara liste dosyası (`kara_liste.json`) bulunamazsa veya geçersizse, kullanıcıya uyarı verilmeli.
- **Beklenmeyen Veri Tipleri:**
  - Yolcu ID veya eşya listesi geçersiz formatta olduğunda sistem çökmemeli ve hata loglanmalı.

Use-Case diyagramı



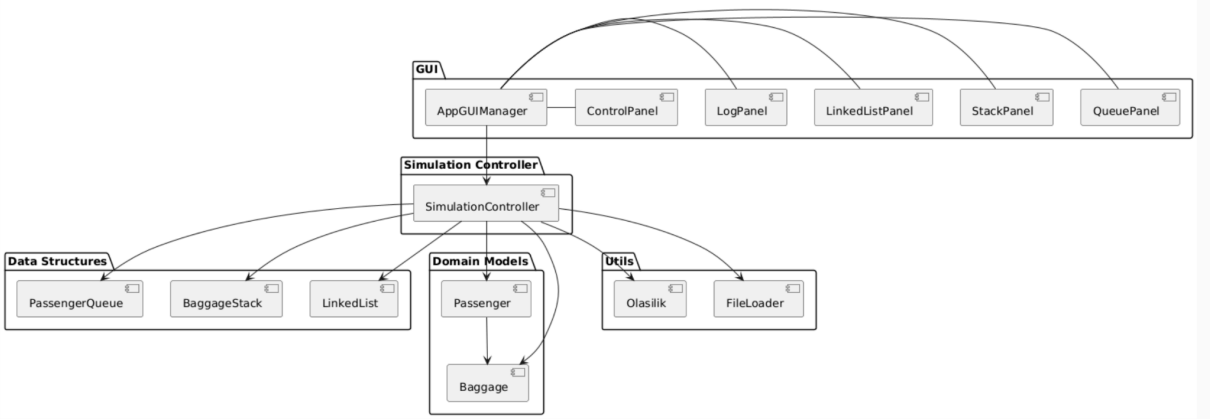
4 TASARIM

Mimari tasarım

1. Katmanlı Mimari

1. **Sunum Katmanı (GUI):**
    - Kullanıcı etkileşimini ve veri görselleştirmesini yönetir.
    - **Paneller:** QueuePanel, StackPanel, LinkedListPanel, LogPanel, ControlPanel.
    - **Teknoloji:** Tkinter kütüphanesi kullanılarak geliştirilmiştir.
2. **Uygulama Katmanı (Simülasyon Kontrolcüsü):**
    - İş mantığını ve simülasyon akışını yönetir.
    - **Ana Bileşen:** `SimulationController` sınıfı.
    - **İşlevler:** Yolcu işleme, kara liste kontrolü, bagaj tarama, raporlama.
3. **Veri Katmanı:**
    - Veri yapılarını ve domain modellerini içerir.
    - **Veri Yapıları:** `PassengerQueue` (kuyruk), `BaggageStack` (yığın), `LinkedList` (bağlı liste).
    - **Modeller:** `Passenger` (yolcu), `Baggage` (bagaj).
4. **Yardımcı Katman (Utils):**
    - Tekrar kullanılabilir yardımcı fonksiyonları barındırır.
    - **İçerik:** Rastgele bagaj oluşturma (`olasilik.py`), kara liste dosya yükleme (`load_blacklist_from_file`).

Modül diyagramı



Kullanılacak teknolojiler

Yazılım Dili

Proje, **Python 3.7**+ sürümleri ile geliştirilmiştir. Python'un tercih edilme nedenleri:

- **Hızlı Prototipleme:** Veri yapılarını ve GUI'yi hızla implemente etme imkanı.
- **Geniş Kütüphane Desteği:** Standart kütüphanelerle temel işlevlerin kolayca entegrasyonu.
- **Platform Bağımsızlık:** Windows, macOS ve Linux'ta sorunsuz çalışabilme.

Kullanılan Harici Kütüphaneler

Projede **ek bir harici kütüphane kullanılmamıştır**. Temel Python modülleriyle yetinilmiştir:

1. **Tkinter:** GUI geliştirmek için standart Python kütüphanesi.
2. **Random:** Rastgele eşya ve yolcu oluşturmak için.
3. **CSV:** Raporları CSV dosyasına yazmak için.
4. **JSON:** Kara liste verilerini dosyadan okumak için.
5. **Unittest:** Test senaryolarını yazmak için.

Diğer Teknolojiler ve Araçlar

1. **Veri Yapıları:**
    - Kuyruk (Queue), yığın (Stack), bağlı liste (Linked List) sıfırdan implemente edilmiştir.
2. **GUI Tasarımı:**
    - **Tkinter Widget'ları:** Listbox, Button, LabelFrame, ScrolledText.

5 UYGULAMA

Kodlanan bileşenlerin açıklamaları

1. Ana Sınıflar

a. `SimulationController` (main.py)

- **Amaç:** Tüm simülasyon sürecini yönetir.
- **İşlevler:**
    - `__init__()`: Veri yapılarını (kuyruk, yığın, bağlı liste) başlatır, istatistikleri sıfırlar.

    - yolcu_olustur_ve_ekle(): Rastgele bagajlı yolcu oluşturup kuyruğa ekler.
    - simulasyon_adimini_calistir(): Kuyruktan yolcu alır, kara liste ve bagaj kontrolü yapar.
    - kara_listede_mi(): Yolcunun kara listede olup olmadığını kontrol eder.
    - bagaj_tarama(): Bagajdaki eşyaları tarar, tehlikeliyse yığına aktarır.
    - stack_taramasi(): Yığındaki eşyaları LIFO ile çıkarıp loglar.
    - rapor_goster(): İstatistikleri CSV'ye yazar ve GUI'de gösterir.

b. Passenger **(models/yolcu.py)**

- **Amaç:** Yolcu bilgilerini ve bagajını temsil eder.
- **Özellikler:**
    - passenger_id: Benzersiz yolcu ID'si (örneğin: Yolcu #1).
    - baggage: Baggage sınıfından oluşturulan bagaj nesnesi.
    - is_risky: Yolcunun risk durumu (kara listede ise True).

c. Baggage **(models/bagaj.py)**

- **Amaç:** Bagajdaki eşyaları ve tehlikeli durumu yönetir.
- **Metodlar:**
    - has_dangerous_items(): Eşyaları tehlikeli listede kontrol eder.
    - get_items(): Bagajdaki tüm eşyaları döndürür.

---

2. Veri Yapıları

a. PassengerQueue **(models/queue.py)**

- **Amaç:** FIFO mantığıyla yolcu kuyruğunu yönetir.
- **Metodlar:**
    - enqueue(): Yolcuyu kuyruğa ekler.
    - dequeue(): Kuyruğun başındaki yolcuyu çıkarır.
    - is_empty(): Kuyruğun boş olup olmadığını kontrol eder.

b. BaggageStack **(models/stack.py)**

- **Amaç:** LIFO mantığıyla şüpheli eşyaları yönetir.
- **Metodlar:**
    - push(): Eşyayı yığına ekler.

    - pop(): Yığından en üstteki eşyayı çıkarır.

c. LinkedList **(models/linkedlist.py)**

- **Amaç:** Kara listedeki yolcu ID'lerini saklar.
- **Metodlar:**
    - append(): Yeni ID ekler.
    - search(): ID'yi bağlı listede arar.

---

3. Yardımcı Modüller

a. utils/olasilik.py

- **Amaç:** Rastgele veri üretme ve kontrol fonksiyonlarını içerir.
- **Fonksiyonlar:**
    - generate_random_baggage(): 5-10 arası rastgele eşya oluşturur (%10 tehlikeli).
    - esya_tehlikeli_mi(): Eşyanın yasaklı listede olup olmadığını kontrol eder.
    - load_blacklist_from_file(): kara_liste.json dosyasından kara liste ID'lerini yükler.

b. gui.py

- **Amaç:** Tkinter tabanlı GUI'yi yönetir.
- **Sınıf:** AppGUIManager
    - update_queue_panel(): Kuyruk panelini günceller.
    - update_stack_panel(): Yığın panelini ve alarm durumunu günceller.
    - log_message(): Log paneline renkli mesajlar ekler.
    - _run_full_simulation(): Tüm kuyruğu otomatik işler.

---

4. Test Bileşenleri

a. test_simulator.py

- **Amaç:** Birim testleriyle sistemin doğruluğunu kontrol eder.
- **Test Senaryoları:**

- test_passenger_creation(): Yolcu oluşturma ve kuyruğa ekleme testi.
- test_blacklist_check(): Kara liste arama testi.
- test_dangerous_item_detection(): Tehlikeli eşya tespiti testi.

Bileşen Etkileşimleri

1. **Simülasyon Başlatma:**
   - SimulationController, yolcu_olustur_ve_ekle() ile yolcu oluşturur ve PassengerQueue'ya ekler.
2. **Kara Liste Kontrolü:**
   - kara_listede_mi(), LinkedList üzerinde ID arar.
3. **Bagaj Tarama:**
   - bagaj_tarama(), Baggage nesnesinin has_dangerous_items() metodunu çağırır.
4. **Yığın İnceleme:**
   - stack_taramasi(), BaggageStack'ten eşyaları çıkarır ve loglar.
5. **GUI Senkronizasyon:**
   - AppGUIManager, her adımda panelleri SimulationController ile senkronize eder.