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

PROJE KONUSU: HAVALİMANI BAGAJ KONTROL SİSTEMİ

ÖĞRENCİ ADI: GÜREL BİLGİN, ALİ AKSOY

ÖĞRENCİ NUMARASI: 220502041, 210501009

DERS SORUMLUSU: ERAY DURSUN, CANDİDE ÖZTÜRK

TARİH: 09.05.2025

1 GİRİŞ

Bu projenin temel amacı, bir havalimanındaki güvenlik kontrol noktalarında gerçekleşen bagaj tarama sürecini yazılım temelli bir simülasyonla modellemektir. Günümüzde güvenlik, havalimanlarında en öncelikli konulardan biridir. Yolcuların ve bagajlarının güvenli bir şekilde kontrol edilmesi, hem insan hatalarını azaltmak hem de süreci hızlandırmak açısından büyük önem taşır. Gerçek dünyada işleyen bu sistemin, algoritmik olarak nasıl yürütülebileceğini anlamak, hem yazılım geliştirme açısından hem de sistem analizi bakımından faydalı bir deneyim sunar.

Simülasyon sayesinde, bir yolcunun güvenlik alanına gelişiyle başlayan ve bagajının taramadan geçerek kontrol edilmesine kadar olan tüm adımlar, yazılım ortamında modellenmiştir. Bu süreçte kuyruk (queue), yığın (stack) ve tek yönlü bağlı liste (single linked list) gibi veri yapıları kullanılarak yolcu akışı, bagajların kontrolü ve sistemin genel işleyişi kurgulanmıştır. Proje ayrıca, algoritmaların gerçek dünyadaki süreçleri nasıl temsil edebileceğini ve yazılım geliştirme sürecinde hangi yapısal yaklaşımların kullanılabileceğini göstermeyi amaçlamaktadır.

2 GEREKSİNİM ANALİZİ

Fonksiyonel Gereksinimler

Bu projede, havalimanı güvenlik kontrol sürecinin yazılım temelli bir simülasyonu geliştirilmiştir. Sistem, yolcuların ve bagajlarının belirli algoritmalarla analiz edilmesini ve kara liste kontrolünün yapılmasını sağlamaktadır. Aşağıda sistemin temel işlevsel gereksinimleri açıklanmıştır:

1. Yolcu ve Bagaj Oluşturma

- Sistem, rastgele ID'ye sahip yeni bir yolcu üretir.
- Her yolcuya ait bagajlar rastgele sayıda ve rastgele eşya içeriği ile oluşturulur.
- Yolcu ve bagaj bilgileri kuyruk veri yapısına (FIFO) eklenir.

2. Kuyruktan Yolcu Alma ve Güvenlik Taraması Başlatma

- Sistem, kuyruktaki ilk yolcuyu alarak güvenlik taramasını başlatır.
- Bu işlem sırasında yolcunun kimliği kara listeyle kontrol edilir.
- Yolcunun bagajları birer birer taranır.

3. Kara Liste Kontrolü

- Yolcunun ID'si, tek yönlü bağlı listeyle temsil edilen kara listedeki ID'lerle karşılaştırılır.
- Eşleşme olması durumunda sistem özel işlem yapar ve kullanıcıyı uyarır.

4. Bagaj Tarama

- Yolcunun bagajı stack (yığın) yapısına eklenir.
- Eşyalar LIFO (Son Giren İlk Çıkar) mantığıyla kontrol edilir.
- Her eşya, sistemde tanımlı olan tehlikeli eşya listesiyle karşılaştırılır.

5. Tehlikeli Eşya Tespiti

- Tarama sırasında tehlikeli bir eşya tespit edilirse sistem bunu kullanıcıya bildirir.
- Uyarı mesajı ve log kaydı oluşturulur.

- Bu yolcunun geçişi engellenebilir veya raporlanabilir.

6. Simülasyon Raporu Oluşturma

- Simülasyon tamamlandığında, tüm işlem adımlarına dair özet rapor sunulur.
- Raporda kaç yolcu kontrol edildiği, kaç tanesinde tehlikeli eşya bulunduğu ve kaçının kara listede olduğu gibi bilgiler yer alır.

7. Olay Kaydı ve Geribildirim

- Gerçekleşen her işlem kullanıcıya anlık olarak bildirilir.
- Yolcunun kontrol durumu, eşya tarama adımları ve varsa tehlike durumları ekrana yazdırılır.
- Simülasyon boyunca yapılan işlemler günlük (log) şeklinde takip edilir.

8. Hazır Senaryo Verisi Yükleme (Opsiyonel)

- Test kolaylığı sağlamak adına örnek yolcu ve bagaj verilerini içeren bir hazır senaryo başlatılabilir.
- Bu sayede kullanıcı manuel giriş yapmadan sistem işleyişini test edebilir.

Arayüz Gereksinimleri

Projenin kullanıcı ile etkileşim kurmasını sağlayan arayüz, metin tabanlı (terminal) bir yapıda tasarlanmıştır. Kullanıcı, simülasyonu başlatmak, yolcu eklemek, kara liste kontrolü yapmak, bagaj taramasını başlatmak ve sonuçları görüntülemek gibi işlemleri klavye üzerinden gerçekleştirmektedir. Arayüzün kullanım kolaylığına, işlem sırasının netliğine ve geribildirimlerin anlaşılır olmasına özen gösterilmiştir.

Aşağıda simülasyon sürecinde kullanıcıya sunulan temel arayüz gereksinimleri açıklanmıştır:

1. Kontrol Paneli

- Kullanıcının simülasyonu başlatmasını sağlar (simulasyonu_baslat()).
- Yeni yolcu ve bagajların sisteme eklenmesini sağlar (yolcu_olustur()).
- Örnek verilerin hızlıca yüklenmesini sağlayan hazır senaryo komutu eklenmiştir.
- Simülasyon sonrası genel bir rapor alınmasını sağlar (rapor_goster()).

2. Yolcu Kuyruğu Paneli

- Sistemde bekleyen tüm yolcular FIFO sırasına göre listelenir.
- Her yolcunun ID bilgisi ve bagaj sayısı gösterilir.
- Kuyruktan sıradaki yolcu alındığında güncel liste ekrana yansıtılır.

3. Bagaj Yığını Paneli

- Güvenlik kontrolüne alınan yolcunun bagajındaki eşyalar LIFO yapısıyla görüntülenir.
- Stack yapısında kontrol edilen eşyalar sırayla ekrana yazdırılır.
- Tehlikeli eşya tespiti durumunda sistem alarm üretir.

4. Kara Liste Paneli

- Kara listedeki yolcuların ID bilgileri listelenir.
- Yeni gelen yolcu ile liste karşılaştırıldığında eşleşme varsa kullanıcıya uyarı mesajı verilir.

- Kara liste zamanla güncellenebilir, liste uçtan uca dolaşılarak kontrol yapılır.

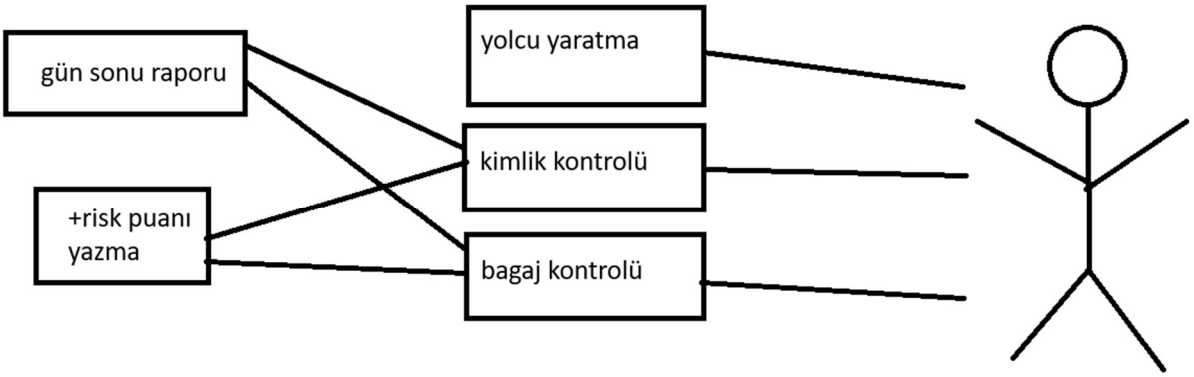
5. Olay Kayıt ve Geri Bildirim Paneli

- Gerçekleşen her adım, kullanıcının görebileceği şekilde ekrana yazılır (örneğin: "Yolcu 102 taranıyor.", "Tehlikeli eşya: bıçak").
- Kullanıcı yönlendirmeleri ve hata mesajları açık bir dille verilir.
- Simülasyon sonunda, işlem özeti ve genel sonuçlar listelenir.

6. Girdi Doğrulama

- Kullanıcıdan alınan komutlar veya veriler kontrol edilir, yanlış girişlerde bilgilendirici uyarı mesajı verilir.
- Gerekli yerlerde tekrar giriş talep edilir.

2.1 Use-Case Diyagramı



3 TASARIM

3.1 Mimari Tasarım

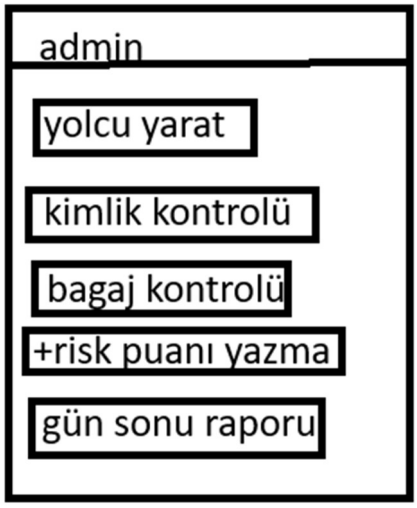
Bu projede geliştirilen yazılım, havalimanı güvenlik kontrol sürecini yazılım temelli bir simülasyonla modellemek amacıyla tasarlanmıştır. Uygulamanın genel mimarisi, her biri belirli bir veri yapısını temsil eden farklı bileşenlerden oluşmaktadır. Bu yapı sayesinde sistem hem görsel olarak takip edilebilir hale getirilmiş hem de yazılım geliştirmenin temel ilkeleri olan modülerlik ve sadelik korunmuştur. Mimari şu ana bileşenlerden oluşur:

1. Yolcu Kuyruğu (Queue)

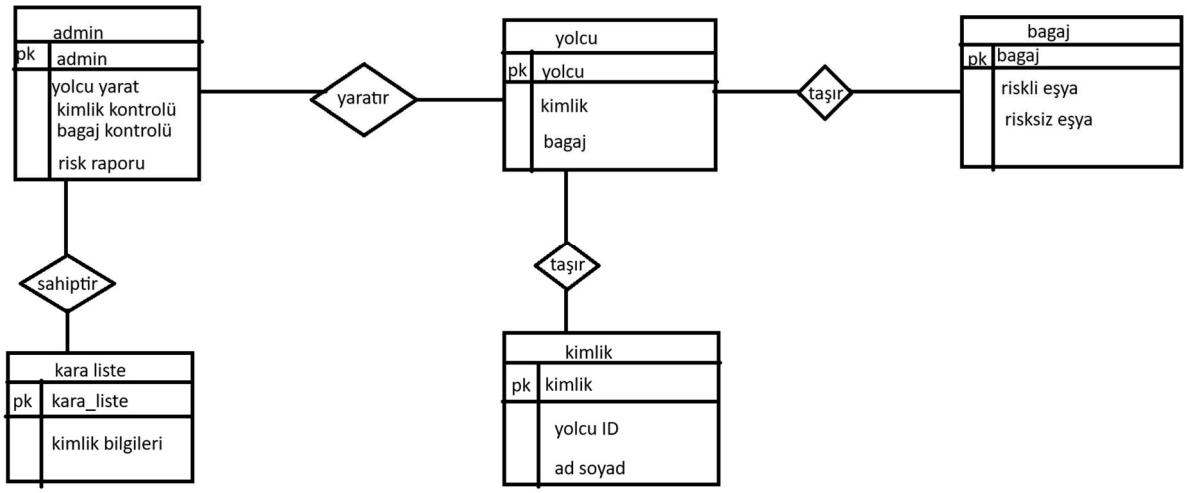
Güvenlik noktasına gelen yolcular, FIFO (First In, First Out) mantığıyla kuyruğa alınır. Bu yapı, gerçek hayattaki sıraya girme durumunu yazılım ortamına taşır. Yolcuların ID bilgisi ve taşıdıkları bagajlar bu noktada sisteme dahil edilir.

2. Bagaj Yığını (Stack)
Her yolcunun bagajında bulunan eşyalar, güvenlik kontrolü sırasında LIFO (Last In, First Out) mantığıyla incelenir. Bagajın üst kısmına en son konulan eşya ilk kontrol edilir. Stack yapısı bu süreçte şüpheli madde taramasını birebir modellemeye uygundur.
3. Kara Liste (Linked List)
Güvenlik geçmişinde sorun yaşamış yolcuların bilgileri tek yönlü bağlı liste yapısında tutulur. Sisteme alınan her yeni yolcu, bu liste üzerinden kontrol edilir. Listede eşleşme varsa, sistem otomatik olarak uyarı verir ve özel işlem başlatılır.
4. Olay Günlüğü (Log)
Simülasyonun başından sonuna kadar gerçekleşen tüm işlemler, yolcu hareketleri, tarama sonuçları ve güvenlik alarmları bu kayıt sisteminde tutulur. Kullanıcı, süreci adım adım bu panel üzerinden izleyebilir ve gerektiğinde geriye dönük inceleme yapabilir.

Modül Diyagramı



ER Diyagramı:



3.2 Kullanılacak Teknolojiler

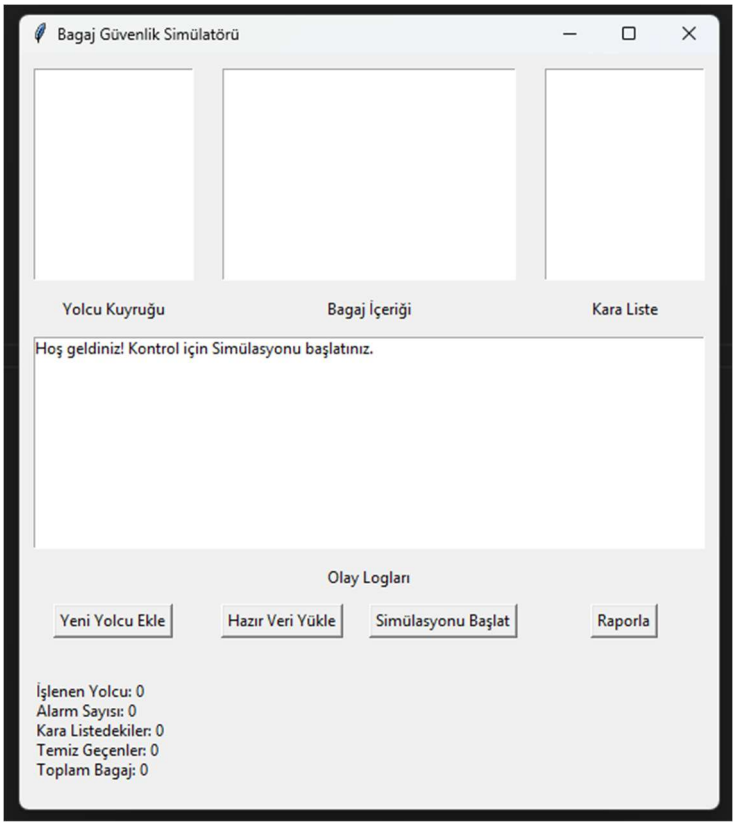
Python: Projenin tüm yazılım geliştirme süreci Python diliyle gerçekleştirilmiştir.

Nesne Yönelimli Programlama: Sınıf yapıları, kalıtım ve çok biçimlilik gibi OOP kavramları etkin olarak kullanılmıştır.

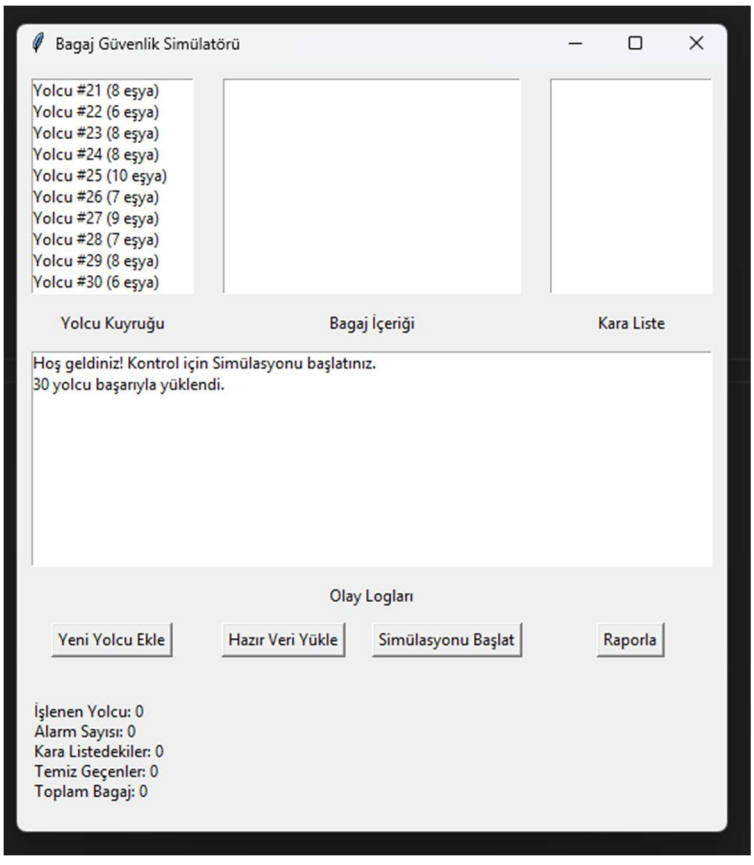
Veri Yapıları: queue, stack, linked list gibi temel veri yapıları projeye özel olarak sınıflandırılarak kullanılmıştır.

3.3 KULLANICI ARAYÜZÜ TASARIMI

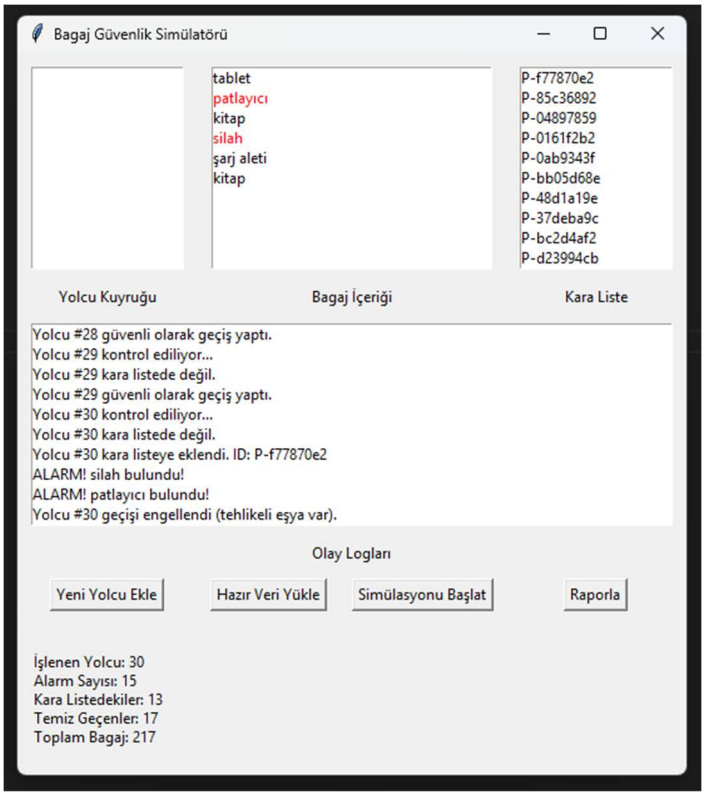
Genel görünüm:



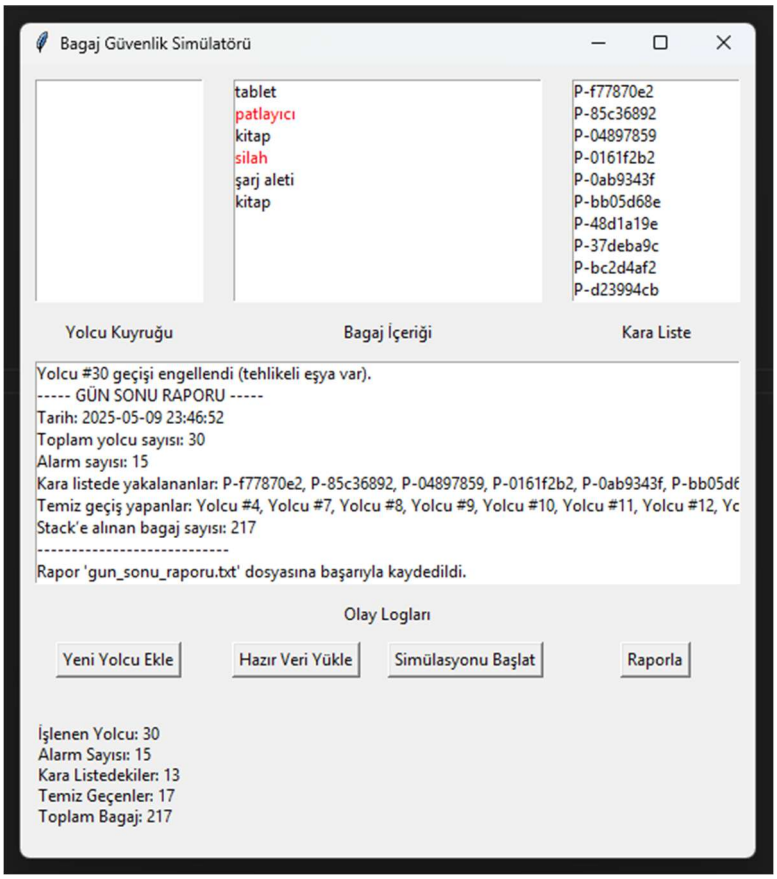
Hazır veri yükleme:



Simülasyon görüntüsü:



Raporlama:



4 UYGULAMA

4.1 Kodlanan Bileşenlerin Açıklamaları

Bu projede, Python programlama dili kullanılarak terminal tabanlı bir simülasyon yazılımı geliştirilmiştir. Simülasyon, havalimanı güvenlik kontrol süreçlerini modellemek amacıyla farklı veri yapılarının (kuyruk, yığın ve tek yönlü bağlı liste) entegrasyonuyla oluşturulmuştur. Kodlanan temel bileşenler ve görevleri aşağıda açıklanmıştır:

1. Yolcu ve Bagaj Oluşturma

- **yolcu_olustur()**: Rastgele yolcu kimlik numarası ve eşyalar üretir. Üretilen yolcu ve eşyaları kuyruk veri yapısına ekler. Her yolcunun en az bir bagajı ve bu bagajda rastgele oluşturulmuş eşyalar yer alır. Kuyruk sıralamasıyla işlemler başlatılır.

2. Simülasyon Akış Kontrolü

- **simulasyonu_baslat()**: Kuyruktaki ilk yolcuyu alarak güvenlik taramasını başlatır. Yolcunun bagajı yığına (stack) gönderilir ve içerdiği eşyalar kontrol edilir. Aynı zamanda yolcunun kara listede olup olmadığı bağlı liste üzerinden denetlenir.

3. Kara Liste Kontrolü

- **kara_listede_mi(id)**: Tek yönlü bağlı liste kullanılarak daha önceden sisteme kayıtlı olan şüpheli yolcuların kimlikleriyle karşılaştırma yapar. Eğer eşleşme varsa, yolcuya özel işlem uygulanır.

4. Bagaj Tarama İşlemleri

- **bagaj_tarama(yolcu)**: Yolcunun bagajındaki eşyaları teker teker yığının en üstüne yerleştirir. Bu işlem sırasında tüm eşyaların içerikleri kontrol edilmek üzere hazır hale getirilir.

- **esya_tehlikeli_mi(eşya)**: Eşyanın sistemde tanımlı olan tehlikeli/yasaklı eşyalar listesinde olup olmadığını kontrol eder. Örneğin: bıçak, patlayıcı, yanıcı madde vs. gibi öğeler tespit edilirse bu bilgi simülasyon raporuna yansıtılır.

- **stack_taramasi()**: Yığındaki tüm eşyalar sırayla çıkarılır ve her bir eşyanın güvenlik açısından riskli olup olmadığı analiz edilir. Bu süreç LIFO prensibiyle yürütülür.

5. Raporlama ve Özet Bilgi

- **rapor_goster()**: Simülasyon tamamlandığında toplam yolcu sayısı, tehlikeli eşya taşıyan yolcu sayısı, kara listedeki yolcular gibi istatistiksel verileri kullanıcıya sunar. Sistem üzerinde test edilmiş işlemlerin genel özeti bu fonksiyon aracılığıyla alınır.

Karşılaşılan Zorluklar ve Çözüm Yöntemleri

Projeyi Python diliyle geliştirirken karşılaştığımız başlıca zorluklardan biri, farklı veri yapılarının (kuyruk, yığın, bağlı liste) aynı sistem içinde senkron bir şekilde çalışmasını sağlamak oldu. Her veri yapısının kendine özgü çalışma mantığı olduğundan, simülasyon akışını bozmadan bu yapılar arası geçişleri kontrol etmek başlangıçta karmaşık hale geldi. Bu sorunu, her veri yapısını kapsayan modüler sınıflar oluşturarak ve bu sınıfları birbirleriyle sade arayüzler üzerinden ilişkilendirerek çözdük.

TEST VE DOĞRULAMA

Geliştirdiğimiz bagaj tarama simülasyonunda testler ağırlıklı olarak manuel yöntemlerle gerçekleştirilmiştir. Her bir veri yapısı (kuyruk, yığın, bağlı liste) için ayrı ayrı test senaryoları uygulanarak sistemin beklenen şekilde çalışıp çalışmadığı gözlemlenmiştir.

Kuyruk veri yapısı üzerinde yapılan testlerde, yolcuların sıraya girme ve kontrol noktasına çağrılma sıraları takip edilerek FIFO (First-In, First-Out) prensibinin doğru şekilde işlediği kontrol edilmiştir. Yığın kullanılarak yapılan bagaj kontrol simülasyonunda, tarama işlemi biten bagajların LIFO (Last-In, First-Out) sırasıyla sistemden çıkartıldığı test edilmiştir. Ayrıca bağlı liste yapısı kullanılarak kontrol noktaları arasında bagajların taşınması sırasında bağlantıların doğru güncellenip güncellenmediği kontrol edilmiştir.

Her işlem sonrasında sistem çıktıları dikkatle incelenmiş ve beklenen senaryoya uygunluk doğrulanmıştır. Özellikle veri yapıları arasında geçişlerin olduğu bölümlerde çeşitli uç senaryolar denenerek simülasyonun dayanıklılığı test edilmiştir. Örneğin, çok sayıda yolcunun art arda sisteme alınması veya yığın boşken çıkartma yapılmaya çalışılması gibi durumlarda sistemin kararlı şekilde çalışması sağlanmıştır.

5 Görev Dağılımı

Projenin tüm aşamalarında genel olarak beraber çalışılmış olup proje bitimi kod incelemesi topluca yapılmıştır. Rapor son kez beraber düzenlenmiştir.

Gürel Bilgin:

- Kullanıcı arayüzünün geliştirilmesi ve menü sisteminin oluşturulması
- Manuel kontrol işlemlerinin kodlanması
- Simülasyon sonunda sistem çıktılarının raporlanması

Ali Aksoy:

- Simülasyon mimarisinin planlanması ve sınıf yapılarının tasarımı
- Kuyruk, yığın ve bağlı liste veri yapılarının Python ile kodlanması
- Simülasyon yöneticisi (controller) sınıfının yazımı

Grup Üyeleri GitHub Hesapları:
Gürel Bilgin: https://github.com/GurelBilgin

Ali Aksoy: https://github.com/aaksoy581