1. Özet

Bu proje, Kocaeli ili İzmit ilçesi özelinde bir toplu taşıma ve taksi sistemi tasarlayarak, kullanıcının mevcut konumundan belirli bir hedef noktaya en uygun rota ile ulaşmasını sağlamaktadır. Rota belirleme sürecinde maliyet, süre, aktarma sayısı gibi faktörler dikkate alınarak kullanıcıya en verimli güzergâh önerilecektir. Sistem, Nesne Yönelimli Programlama (OOP) prensiplerine uygun olarak geliştirilmiştir. OOP prensiplerinin uygulanması, sistemin modülerlik, genişletilebilirlik ve sürdürülebilirlik açısından avantaj sağlamasını temin etmektedir.

2. Giriş

Modern kentlerde ulaşım sistemleri, büyüyen nüfus ve artan taşıma ihtiyacı nedeniyle karmaşıklaşmaktadır. Bu proje, İzmit ilçesinde toplu taşıma ve taksi hizmetlerini entegre ederek en uygun rota bulma problemini çözmeyi amaçlamaktadır. Projede, kullanıcıların en verimli ve ekonomik şekilde seyahat etmelerini sağlayacak algoritmalar geliştirilmiş, özellikle OOP prensipleri kullanılarak sistemin genişletilebilir ve modüler yapıda olması hedeflenmiştir.

3. Yöntem

Proje geliştirilirken aşağıdaki adımlar izlenmiştir:

- **Veri Toplama:** İzmit ilçesindeki toplu taşıma ve taksi sistemleri analiz edilmiştir.
- **Sistem Tasarımı:** OOP prensiplerine uygun şekilde sınıflar oluşturulmuştur.
    - **Soyut Sınıf:** Arac (tüm ulaşım araçlarının temelini oluşturan sınıf)
    - **Alt Sınıflar:** Otobüs, Minibüs, Taksi
    - **Arayüzler:** IRotaHesaplayici, IUcretHesaplayici
    - **Polimorfizm:** Farklı ulaşım araçlarının aynı metodu farklı şekilde gerçekleştirmesi
- **Algoritma Geliştirme:** En kısa yol hesaplaması için Dijkstra algoritması kullanılmıştır.

Algoritma kaba kodu:

1. Veri Yapılarını Başlat:
    - Mesafeler: Her durağa olan en kısa mesafeyi saklayan sözlük.
    - Onceki: Bir düğümün önceki durağını saklayan sözlük.
    - Adimlar: Rota adımlarını tutan sözlük.
    - ZiyaretEdilmemis: Henüz ziyaret edilmemiş düğümler listesi.
2. Başlangıç Duraklarını Ayarla:
    - Tüm duraklara başlangıçta sonsuz mesafe ata.
    - Başlangıç durağının mesafesini 0 yap.

3. Dijkstra Algoritması:
    - Mevcut Durağı Seç:
        - ZiyaretEdilmemis listesinden mesafesi en küçük olanı seç.
        - Eğer bu durak hedef duraksa çık.
    - Komşu Düğümleri Güncelle:
        - Mevcut durağın bağlı olduğu durakları kontrol et.
        - Eğer yeni hesaplanan mesafe önceki değerden küçükse, mesafeyi güncelle.
        - Onceki ve Adimlar sözlüklerini güncelle.
    - Aktarma Kontrolü:
        - Eğer durakta aktarma varsa ve uygun koşulları sağlıyorsa, mesafeyi güncelle.
4. En Kısa Rotayı Çıkart:
    - Onceki haritasını takip ederek rotayı oluştur.
5. Sonucu Döndür:
    - Rota adımları listesi ve hedef durağa olan en kısa mesafe döndürülür.

Teknik Sorular ve Cevapları

2.1 Yeni Ulaşım Yöntemleri (Elektrikli Scooter vb.) Sisteme Nasıl Eklenir?

Daha önce hiç kullanılmamış yeni bir ulaşım yöntemi (örneğin, elektrikli scooter) projeye eklenmek istendiğinde, sistemin genişletilebilir bir yapıya sahip olması gerekmektedir. Bunun için:

- Soyut "Arac" sınıfı oluşturulmalıdır Tüm ulaşım araçları (otobüs, taksi, metro, scooter vb.) bu sınıftan türemelidir.
- Yeni bir ulaşım yöntemi eklendiğinde, sadece "Arac" sınıfını miras alan yeni bir sınıf oluşturulmalı ve uygun metotlar uygulanmalıdır.
- Var olan kodlar değiştirilmeden, yeni ulaşım aracına uygun algoritmalar yazılarak sistem genişletilebilir.

2.2 Otonom Taksi ve Yeni Ulaşım Araçları İçin Gereken Değişiklikler

Otonom taksi gibi yeni bir ulaşım aracı sisteme eklenmek istenirse, aşağıdaki değişiklikler yapılmalıdır:

- "Taksi" sınıfı genişletilmeli veya yeni bir "OtonomTaksi" sınıfı türetilmelidir.
- Otonom taksilerin özel kuralları (örneğin, belirli bölgelerde çalışması, dinlenme süresi, ücretlendirme farkları) belirlenmelidir.
- Rota optimizasyon algoritması, otonom taksilerin güzergâh kurallarını destekleyecek şekilde güncellenmelidir.

2.3 Daha Önce Yazılmış Fonksiyonlardan Hangilerinde Değişiklik Yapılmalıdır?

Eğer sistem genişletilebilir bir yapıda oluşturulmuşsa, yeni ulaşım araçları eklemek için mevcut fonksiyonlarda büyük değişiklikler yapılması gerekmez. Ancak aşağıdaki bölümler etkilenebilir:

- Rota hesaplama fonksiyonları: Yeni ulaşım aracının hız, ücretlendirme ve güzergâh kuralları eklenmelidir.
- Ücret hesaplama fonksiyonları: Yeni aracın ücret tarifesi entegre edilmelidir.

- Araç seçim fonksiyonları: Kullanıcının tercih edebileceği yeni ulaşım araçları eklenmelidir.

2.4 Açık/Kapalı Prensibi (Open/Closed Principle) Kapsamında Yeni Araçlar Nasıl Eklenebilir?

Açık/Kapalı Prensibi'ne göre, mevcut sınıflar değiştirilmeden sistem genişletilebilir olmalıdır. Bunu sağlamak için:

- "Arac" adında soyut bir temel sınıf oluşturulmalıdır.
- Bu sınıfın alt sınıfları olarak farklı ulaşım araçları tanımlanmalıdır.
- Yeni bir ulaşım aracı eklemek için mevcut kodlar değiştirilmeden yalnızca yeni bir sınıf tanımlanmalıdır.
- Polimorfizm kullanılarak, mevcut metodlar yeni sınıflar için uyarlanabilir hale getirilmelidir.

2.5 65 Yaş ve Üzeri Ücretsiz Seyahat Hakkının 20 Seyahat ile Sınırlandırılması

Eğer 65 yaş üstü bireyler için ücretsiz seyahat hakkı 20 seyahat ile sınırlandırılacaksa, aşağıdaki değişiklikler yapılmalıdır:

- "Yolcu" sınıfına bir "yas" ve "ucretsizSeyahatSayisi" özelliği eklenmelidir.
- Ücret hesaplama fonksiyonuna bir kontrol eklenmelidir:
    - Eğer yolcunun yaşı 65 ve üzeriyse, "UcretsizSeyahatSayisi" kontrol edilmelidir.
    - Eğer bu sayı 20'yi aştıysa, normal ücretlendirme uygulanmalıdır.
    - Aksi takdirde, seyahat ücretsiz olarak işlenmelidir ve "UcretsizSeyahatSayisi" artırılmalıdır.

4. Deneysel Sonuçlar

Test senaryoları üzerinde yapılan analizlerde, sistemin farklı ulaşım seçeneklerini ve maliyetleri doğru bir şekilde hesapladığı gözlemlenmiştir. Kullanıcılara en hızlı ve en ekonomik rota sunulmuş, OOP prensipleri sayesinde yeni ulaşım araçlarını sisteme eklemek kolaylaştırılmıştır.

5. Sonuç

Bu proje, İzmit ilçesinde toplu taşıma ve taksi sistemlerinin verimli bir şekilde kullanılmasını sağlamak amacıyla geliştirilmiştir. OOP prensipleri sayesinde sistem modüler, genişletilebilir ve sürdürülebilir bir yapıda tasarlanmıştır. Gelecekte, otonom taşıma sistemleri gibi yeni teknolojiler kolayca entegre edilebilecektir.

6. Kaynakça

https://www.codeproject.com/Articles/1221034/Pathfinding-Algorithms-in-Csharp

Proje Görselleri