PROLAB-II PROJE RAPORU

Tuğba YILMAZ ve Ceyda YURDASUCU

Ulaşım Rota Planlama Sistemi

1.0. ÖZET

Bu projede, Kocaeli Üniversitesi PROLAB-II dersi kapsamında İzmit şehir içi ulaşım rota planlama sistemi geliştirilmiştir. Projenin temel amacı, kullanıcıların belirttiği başlangıç ve hedef noktaları arasında en etkin ulaşım rotalarını hesaplamak ve bu rotaları kullanıcıya anlaşılır biçimde sunmaktır. Bu doğrultuda proje, otobüs, tramvay ve taksi gibi farklı ulaşım modlarını desteklemekte ve kullanıcının tercihine göre en hızlı, en ekonomik veya özel senaryolara uygun rotaları hesaplamaktadır. Sistem ayrıca, kullanıcının yürüyüş mesafesini ve taksi kullanımını optimize etmekte, farklı kullanıcı tiplerine göre (öğrenci, yaşlı vb.) indirim hesaplamalarını gerçekleştirmektedir.

Projede kullanılan ulaşım verileri JSON formatında hazırlanmış ve özel olarak oluşturulan sınıflar aracılığıyla işlenmiştir. Veriler, OOP prensipleri çerçevesinde yapılandırılarak sistemin verimli çalışması sağlanmıştır. Algoritma olarak temel Dijkstra algoritması kullanılmış, ayrıca süre ve ücrete yönelik optimizasyon sağlayan varyantlar geliştirilmiştir.

Kullanıcı arayüzü WinForms ile geliştirilmiş olup, Leaflet.js kütüphanesi aracılığıyla dinamik harita gösterimi entegre edilmiştir. Kullanıcılar, farklı ulaşım seçenekleri ve detaylı rota bilgileri ile etkin bir kullanıcı deneyimine sahip olmaktadır. Sistem, gerek arayüz tasarımı gerek algoritma performansı açısından kullanıcıların şehir içi ulaşım deneyimlerini iyileştirmeye yönelik kapsamlı bir çözüm sunmaktadır.

1.0.1. Verilerin JSON Dosyasından Okunması

Projenin ilk aşamasında İzmit şehrine ait ulaşım verileri JSON formatında hazırlanarak okunmuştur. Durakların konum bilgileri, ulaşım süreleri, mesafeleri ve ücret bilgileri JSON dosyasından okunup uygun veri yapılarına dönüştürülmüştür.

1.0.2. Veri Yapısının Oluşturulması

Okunan veriler OOP prensiplerine göre Konum, Durak, UlasimVerisi ve benzeri sınıflar halinde yapılandırılarak tutulmuştur. Bu yapılandırma, verilerin sonraki işlemlerde kolay ve verimli kullanılmasını sağlamıştır.

1.0.3. Algoritmalar ve İşlemler

Projede temel olarak Dijkstra algoritması kullanılmıştır. Ayrıca süreye göre en hızlı rota, ücrete göre en ucuz rota gibi özel durumlar için Dijkstra algoritmasının varyasyonları geliştirilmiştir.

1.0.3.1. En Kısa ve En Ucuz Rota Hesaplama

Başlangıç ve hedef konumlar arasında en hızlı ve en ucuz rotalar hesaplanmıştır. Bu hesaplamalar özel olarak optimize edilen algoritmalar ile gerçekleştirilmiştir.

1.0.3.2. Yürüyüş ve Taksi Karar Mantığı

Başlangıç ve hedef noktasına 2 km'den uzak mesafelerde taksi, daha yakın mesafelerde ise yürüyüş kararları otomatik olarak uygulanmış, bu işlemler kullanıcıya detaylı olarak açıklanmıştır.

1.0.3.3. Kullanıcı Tipine Göre İndirim Hesaplamaları

Kullanıcıların öğrenci veya yaşlı gibi farklı tiplerde olması durumunda ücretlerde indirim uygulanmıştır.

2.0. GİRİŞ

İzmit şehir içi ulaşımında vatandaşların karşılaştığı temel sorunlardan biri, uygun rotanın hızlı ve ekonomik biçimde belirlenmesidir. Proje, kullanıcıların belirtilen iki nokta arasındaki ulaşım alternatiflerini değerlendirmelerini kolaylaştırmak ve en iyi seçeneği hızlıca sunmak amacıyla tasarlanmıştır. Bu bağlamda kullanıcıların farklı kriterlere göre (hız, maliyet, konfor) rota seçimi yapabilmeleri sağlanmıştır. Projenin geliştirilme sürecinde, SOLID prensiplerine ve tasarım desenlerine dikkat edilerek sürdürülebilir ve modüler bir yapı oluşturulmuştur. Kullanılan algoritmalar ve veri yapıları, uygulamanın performansını artırmak ve verilerin etkin kullanımını sağlamak için titizlikle seçilmiştir.

3.0. YÖNTEM

3.1. Verilerin Hazırlanması

Proje kapsamında kullanılan veri seti, İzmit şehir içi ulaşımına ait durak bilgileri, mesafe, süre ve ücret verilerinden oluşmaktadır. Bu veriler, JSON dosyası aracılığıyla sisteme aktarılmıştır. JSON formatının tercih edilme sebebi, verilerin hızlı ve etkin bir şekilde okunup işlenebilmesi ve gerektiğinde kolayca güncellenebilmesidir.

3.2. Veri Modelleme

Veriler, Nesne Yönelimli Programlama (OOP) prensipleri doğrultusunda yapılandırılmıştır. Bu amaçla Konum, Durak, UlasimVerisi gibi temel veri sınıfları oluşturulmuştur. Ayrıca rota hesaplama işlemleri için IRotaHesaplayici gibi interface'ler ve Strategy Pattern uygulanarak farklı senaryolar için optimize edilmiş hesaplama sınıfları geliştirilmiştir. Bu yaklaşım, kodun genişletilebilirliğini ve bakım kolaylığını artırmıştır.

3.3. Algoritma Uygulamaları

Proje içerisinde temel algoritma olarak Dijkstra algoritması kullanılmıştır. Ancak kullanıcı ihtiyaçları doğrultusunda süre ve maliyet optimizasyonlarına yönelik özel Dijkstra algoritma varyasyonları geliştirilmiştir. Yürüyüş veya taksi gibi ulaşım kararları için ise mesafe tabanlı hesaplamalar yapılmıştır.

3.4. Görselleştirme

Uygulama içerisinde rotaların görselleştirilmesi amacıyla Leaflet.js harita kütüphanesi kullanılmıştır. Kullanıcıya sunulan rota bilgileri HTML formatında detaylı bir şekilde hazırlanmış ve kullanıcı dostu bir arayüzle sunulmuştur.

3.5. Kullanılan Araçlar ve Teknolojiler

- Programlama dili: C# (.NET 8.0)
- Kullanıcı arayüzü: WinForms
- Harita entegrasyonu: HTML, JavaScript (Leaflet.js)
- Veri formatı: JSON

4.0. DENEYSEL SONUÇLAR

Geliştirilen sistem, kapsamlı test süreçlerinden geçirilmiştir. Test sonuçlarında farklı rota hesaplama stratejileri kullanılarak yapılan hesaplamaların tutarlı ve doğru olduğu görülmüştür. Özellikle kullanıcı tiplerine göre yapılan indirim uygulamaları ve mesafe bazlı yürüyüş/taksi kararlarının kullanıcı deneyimini olumlu etkilediği gözlemlenmiştir. Ayrıca, sistemin farklı zamanlarda ve yoğunluklarda çeşitli kullanıcı senaryolarına etkin cevap verdiği testlerle doğrulanmıştır.

5.0. SONUÇ

Bu proje ile İzmit şehir içi ulaşımında rota planlama problemlerine güçlü ve etkili çözümler üretilmiştir. Kullanıcıların farklı ihtiyaçlarına yönelik geliştirilen algoritmalar, kullanıcı dostu arayüz ve detaylı görselleştirme ile birleşerek kullanıcı memnuniyetini üst seviyeye taşımıştır. Sistemin başarılı çalışması, gelecekte daha büyük ölçekte veri entegrasyonu, gerçek zamanlı trafik bilgileri kullanımı ve mobil platform desteği gibi geliştirmelerle daha da zenginleştirilebilir ve yaygınlaştırılabilir.

6.0. ÖĞRENCİ KATKILARI

Ceyda YURDASUCU ve Tuğba YILMAZ

Bu proje kapsamında her iki öğrenci de yazılım geliştirme sürecinin tüm aşamalarında aktif olarak görev almıştır. Projenin analiz, tasarım, kodlama, test ve dökümantasyon süreçlerine eşit katkı sağlamışlardır. Öğrenciler, .NET 8.0 platformunda C# dili ile geliştirilen WinForms tabanlı kullanıcı arayüzünü birlikte planlamış ve uygulamıştır. Harita entegrasyonu, JSON veri yapısının çözümlemesi, veri modellemesi (OOP yapısı ile Durak, Konum, Rota vb. sınıfların tasarımı), rota hesaplama algoritmalarının (Dijkstra algoritması ile) uygulanması ve farklı rota stratejilerinin geliştirilmesinde birlikte çalışmışlardır.

Ayrıca proje içerisinde Strategy, Factory ve Template Method gibi yazılım tasarım desenleri başarılı bir şekilde kullanılmış; bu desenlerin projeye uygun şekilde entegre edilmesi ve SOLID prensiplerine uygun kod geliştirme süreçleri de yine her iki öğrenci tarafından ortaklaşa yürütülmüştür. Kullanıcı tipine göre ücretlendirme, ödeme yöntemlerinin yönetimi (Nakit, Kredi Kartı, KentKart), rota adımlarının detaylı açıklamalarla gösterimi, alternatif rotaların karşılaştırılması ve harita üzerinde dinamik çizimi gibi gelişmiş özellikler de birlikte geliştirilmiştir.

Proje boyunca ortaya çıkan teknik sorunların çözümünde birlikte fikir üretilmiş, kod yapısının sürdürülebilir ve genişletilebilir olmasına özellikle özen gösterilmiştir. Geliştirilen sistem, yalnızca teknik açıdan değil, aynı zamanda kullanıcı deneyimi ve görsel arayüz açısından da işlevsel ve anlaşılır olacak şekilde tasarlanmıştır. Tüm bu süreçlerde her iki öğrenci aktif görev almış, görev paylaşımı dengeli şekilde yapılmış ve proje eşit sorumluluklarla başarıyla tamamlanmıştır.

7.0. PSEUDOCODE

1.0. Gerekli Sınıfları Tanımla
1.1. Konum Sınıfı (Enlem, Boylam)

- Metotlar: MesafeHesapla()

1.2. Durak Sınıfı (Durak Adı, Durak Id, Konum Bilgisi)

- Metotlar: DurakBilgisiGetir()

1.3. UlasimVerisi Sınıfı (Duraklar, Mesafeler, Süreler, Ücretler)

- Metotlar: VeriYukle()

1.4. RotaHesaplayici Sınıfları (IRotaHesaplayici interface'ini implemente eden sınıflar)

- Metotlar: RotaHesapla()

1.5. DijkstraAlgoritmasi Sınıfı (Graf, Duraklar)

- Metotlar: EnKisaYoluBul(), EnUcuzYoluBul()

2.0. Ana Sınıfı Tanımla 2.1. RotaPlanlamaUygulamasi Sınıfı

- Metotlar: UygulamaBaslat(), HaritaGoster(), RotaDetaylariniGoster()

3.0. Uygulamanın Başlangıç Methodunu Tanımla
3.1. Ana Uygulama Başlangıç Methodu:

- WinForms başlatılır.

4.0. Uygulama Başlangıç Methodu
4.1. Ana Pencereyi Oluştur:

- Harita gösterimi alanı.
- Kullanıcı giriş alanları.
- Rota seçenekleri düğmeleri.

4.2. Düğmelere İşlev Ekle:

- Rota hesaplama başlat.
- Rota seçeneklerini (En hızlı, En ucuz, Sadece taksi vb.) seç.

5.0. Ulaşım Bağlantılarını Kur
5.1. JSON Dosyasını Oku:

- Ulaşım bilgilerini analiz et. - Durak ve bağlantı bilgilerini veri yapılarına dönüştür.

6.0. Algoritmaları Uygula
6.1. En hızlı rota:

- Kullanıcıdan başlangıç ve hedef konumu al.
- Dijkstra algoritmasını süre bazında çalıştır.
- Sonuçları görselleştir.

6.2. En ucuz rota:

- Kullanıcıdan başlangıç ve hedef konumu al.
- Dijkstra algoritmasını ücret bazında çalıştır.
- Sonuçları görselleştir.

6.3. Sadece taksi:

- Başlangıç ve hedef arasında direkt taksi rotası hesapla.
- Sonuçları görselleştir.

7.0. Kullanıcı Tipine Göre Ücret Hesapla

- Kullanıcı tipini al (normal, öğrenci, yaşlı vb.).
- Tipine göre rota ücretini yeniden hesapla ve güncelle.

8.0. Rota Bilgisini Görselleştir

- Başlangıç ve hedef noktaları için marker ekle.
- Durakları marker olarak ekle.
- Rotayı yürüyüş (yeşil), taksi (turuncu), toplu taşıma (mavi) çizgileri ile göster.

9.0. Sonuçları Kullanıcıya Göster

- Rota adımlarını detaylarıyla HTML olarak hazırla.
- Kullanıcıya rota sonuçlarını açıkça sun.

10.0. Uygulamayı Kapat
10.1. Uygulama Kapatıldığında:

- Tüm veri yapılarını temizle.
- Harita ve arayüz bileşenlerini sıfırla.
- Sistem kaynaklarını serbest bırak.

8.0 Gelişmiş Senaryolar ve Gelecekteki Genişletmeler

● Yeni Ulaşım Yöntemleri (Elektrikli Scooter vb.) Sisteme Nasıl Eklenebilir?

Sisteme yeni bir ulaşım yöntemi (örneğin elektrikli scooter) eklendiğinde, öncelikle bu ulaşım türüne ilişkin veriler JSON dosyasına eklenmelidir. Scooter'a özel durak bilgileri, ortalama hız, ücretlendirme gibi verilerle birlikte sistemde bu türü temsil edecek bir rota hesaplayıcı sınıf (örneğin ScooterRotaHesaplayici) tanımlanmalıdır. Bu sınıf IRotaHesaplayici arayüzünü implemente etmeli ve RotaHesaplayiciBase sınıfından türemelidir. Sistem SOLID prensiplerine uygun geliştirildiğinden mevcut algoritmalarda değişiklik yapılmadan genişletme mümkündür.

● Otonom Taksi Gibi Yeni Taşıma Araçları Sisteme Nasıl Dahil Edilir?

Otonom taksi gibi araçlar, mevcut TaksiRotaHesaplayici sınıfı temel alınarak genişletilebilir. Otonom taksiye özgü özellikler (örneğin, farklı ücretlendirme, trafik dışı rota tercihleri vb.) içeren bir OtonomTaksiRotaHesaplayici sınıfı oluşturularak bu yeni tür sisteme entegre edilebilir. JSON verisi bu türü içerecek şekilde genişletilmeli ve RotaHesaplayiciFactory sınıfı içinde uygun şekilde yönlendirilmelidir.

● Hangi Fonksiyonlar Değiştirilmeli veya Genişletilmelidir?

Yeni ulaşım türlerinin eklenmesi durumunda genellikle mevcut fonksiyonlar değiştirilmeden

sistem genişletilebilir. Ancak genişletilecek başlıca alanlar şunlardır: - JSON veri işleyici (VeriYukle metodu) - IRotaHesaplayici arayüzünü implemente eden yeni sınıf - RotaHesaplayiciFactory - Form1.cs içinde kullanıcı seçim alanı Mevcut Dijkstra algoritmaları ve veri sınıfları değişmeden yeniden kullanılabilir.

● Açık/Kapalı Prensibine Uygun Genişletme Nasıl Sağlanır?

Open/Closed Principle, bir modülün geliştirmeye açık ancak değişikliğe kapalı olmasını önerir. Bu projede bu prensip IRotaHesaplayici arayüzü ve RotaHesaplayiciBase sınıfı ile uygulanmıştır. Yeni bir ulaşım türü eklenmek istendiğinde yalnızca yeni bir sınıf oluşturularak mevcut yapılar korunur. Bu sayede sistemin kararlılığı bozulmadan yeni özellikler sisteme entegre edilir.

● 65 Yaş Üzeri Kullanıcılar İçin Ücretsiz Seyahat Hakkı Nasıl Sınırlandırılır?

Proje kapsamında yaşlı bireyler için ücretsiz seyahat hakkının 20 ile sınırlandırılması gerektiğinde Kullanici sınıfına ‘GunlukSeyahatSayisi’ ve ‘Yas’ özellikleri eklenmelidir. Kullanıcının yaşı 65 ve üzeriyse ve seyahat sayısı 20’yi geçtiyse sistem normal ücretlendirme yapmalıdır. Bu kontrol, ücret hesaplamasının yapıldığı sınıflarda uygulanmalıdır.

● Bu Kısıtlama Kodda Nasıl Uygulanmalıdır? Hangi Sınıflar Etkilenir?

Bu sınırlama için aşağıdaki sınıflar ve fonksiyonlar etkilenir: - Kullanici sınıfı: yaş ve günlük seyahat bilgileri tutulur. - RotaHesaplayiciBase: ücret hesaplama mantığı güncellenir. - Form1.cs: kullanıcı bilgileri arayüzden alınır ve rota oluşturma öncesinde kontrol edilir. Her rota hesaplaması sonrası seyahat sayısı bir artırılarak sistem dinamik şekilde güncellenir.

9.0. KAYNAKÇA

1. Microsoft. (2024). .NET 8.0 Documentation. Erişim adresi: https://learn.microsoft.com/en-us/dotnet/
2. Microsoft. (2024). WinForms Documentation. Erişim adresi: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/
3. Newtonsoft.Json Kütüphanesi. (2024). Erişim adresi: https://www.newtonsoft.com/json
4. Leaflet.js Harita Kütüphanesi. (2024). Erişim adresi: https://leafletjs.com
5. OpenAI ChatGPT, Yazılım Geliştirme Desteği ve Danışmanlık. (2024). Erişim adresi: https://openai.com