Akıllı Etkinlik Planlama Platformu

220201073

1. Sude Nur Opan
Bilgisayar Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
sudeopann@gmail.com

220201060

2. Çağla Gök
Bilgisayar Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
caglagok369@gmail.com

Özetçe—
Kullanıcıların etkinlikler oluşturabileceği, katılabileceği ve etkinlikler etrafında sosyal etkileşim kurabilecekleri bir web tabanlı Akıllı Etkinlik Planlama Platformu geliştirmeyi hedeflemektedir. Kullanıcılar kişiselleştirilmiş etkinlik önerileri alacak, sohbet edebilecek ve etkinlikleri harita üzerinden takip edebileceklerdir

I. GİRİŞ

C# dilinde yazdığımız bu projeyi Visual Studio Model - View-Controller(MVC)'da geliştirdik .Rota için mapbox api kullandık .Admin için arayüzde grafik göstermek içinde chart.js kütüphansini fake data için de Bogus kütüphanesini kullandık.Projemizde Katmanlı mimari kullandık entitylerimiz ,viewmodel ve dto classlarımız Model katmanında ;Data katmanınımda projemizde kullandığımız veri tabanımız için Ef core kullandık modellerde yazdığımız entitylermizi tablolarını bu applicationdbcontext classda belirledik bu şekilde tek bir katmanda data ayarlaması yapıp service katmanında yazdığımız metotları çağırarak metotlarımızda bu classı çağırarak tekrarı önlemiş olduk en son da formlarımızda servicelerde yer alan kendi sql kodlarımızda yazdığımız metotlarımızda proje gerekliliklerini gerçekleştirdik. View Katmanında da kullanıcı dostu bir arayüz sağlamaya çalıştık.
Projemizin ana isterleri şunlardır:
1. Kullanıcı kayıt, giriş, şifre sıfırlama işlemleri ve profil güncelleme.
2. Etkinlik oluşturma, güncelleme, silme ve katılım sağlama işlevleri.
3. Kural tabanlı kişiselleştirilmiş öneri sistemi.
4. Etkinlikler arasında zaman çakışmalarını tespit etme sistemi.
5. Mesajlaşma işlevselliği kullanımı
6. Etkinliklerin harita üzerinde gösterilmesi ve en uygun rotaların önerilmesi.
7. Kullanıcıların etkinliklere katılımı üzerinden puan ve başarı kazanma, profil sayfasında gösterimi.
8. Kullanıcı ve etkinlik yönetimi için admin yetkileri; etkinlikleri onaylama ve düzenleme.
9. Arayüz ve tasarım yapısının oluşturulması
10. Kullanıcı doğrulama, yetkilendirme, veri güvenliği ve şifreleme yöntemleri.

II. YÖNTEM VE İLERLEYİŞ

Projemizi çalıştırdığımızda mainde yer alan DI diğer classlarda enjekte ettiğim servisler çalışıyor .Https de çalıştırdığım projemde bizi Anasayfa ekranı bizi karşılıyor ilk kullanıcı ise kaydolup kullanıcı adı, şifre, e-posta, konum, ilgi alanları, ad, soyad, doğum tarihi, cinsiyet, e-posta adresi, telefon numarası ve profil fotoğrafı bilgilerini dolduruyor fotoğrafları url şeklinde sakladık veri tabanında doğrulama için eposta girildikten sonra butona basarak kullanıcı girdiği mail adresine 6 haneli bir onay kodu gönderiliyor ilgi alanları çoklu seçebilir ve ilgi alanları veri tabanında ayrı bir tablo olarak tutuluyor .EmailService çalışma mantığı şu şekilde Sınıf, e-posta ile ilgili işlemleri gerçekleştiren metotlar içerir. İlk metot olan SendResetPasswordEmail, şifre sıfırlama e-postasını göndermek için kullanılır. Bu metot, alıcının e-posta adresini ve şifre sıfırlama bağlantısını parametre olarak alır. Gövde kısmında, şifre sıfırlama bağlantısı yer alır. Alıcının e-posta adresi eklenir ve e-posta asenkron olarak gönderilir. Herhangi bir hata oluşursa, hata mesajı konsola yazdırılır ve hata fırlatılır.İkinci metot olan SendConfirmationEmail, hesap onayı e-postasını göndermek için kullanılır. Bu metot da alıcının e-posta adresini ve onay kodunu parametre olarak alır. SMTP istemcisi oluşturularak bağlantı ayarları yapılır. E-posta mesajı, burada konu ve gövde bilgileri ile oluşturulur. Gövde kısmında, onay kodu vurgulanır. Alıcının e-posta adresi eklenir ve e-posta asenkron olarak gönderilir. Hata durumunda, hata mesajı konsola yazdırılır ve hata fırlatılır.

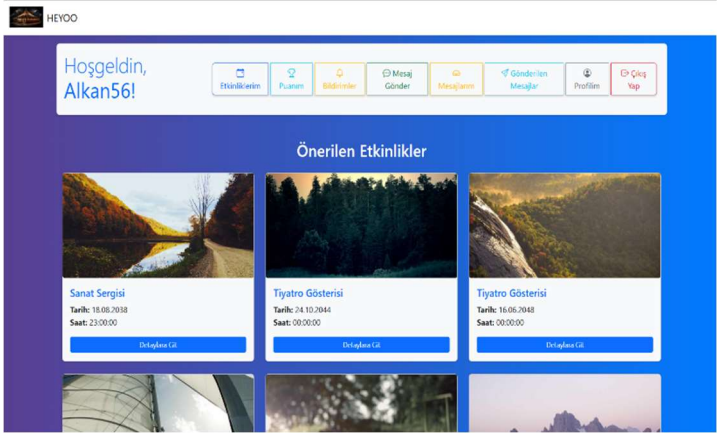
Bu sınıf, kullanıcı etkileşimlerini geliştirmek için etkili bir yöntem sunar; örneğin, kullanıcıların şifrelerini sıfırlamalarına ve hesaplarını onaylamalarına olanak tanır. Kullanılan e-posta adresi ve şifre, doğrudan koda yerleştirilmiştir

Register() metottun ad eğer kullanıcı girişi başarılı ise şifre alınır ve identity uygun şekilde şifrelenip db ye atılır ve user rolü UserManager atılır sonrasında ilgi alanları veri tabanında gönderilir.Kullanıcı kendi konumunu string olarak konum atar backend tarafında düzenlernir İlk olarak, GetNearbyEventsAsync metodu kullanıcıdan alınan enlem ve boylam bilgileriyle çalışmaktadır. Bu metod, veritabanındaki etkinlikleri asenkron bir şekilde çeker ve her bir etkinliğin konumunu kullanıcının konumuyla karşılaştırarak mesafesini hesaplar. Eğer hesaplanan mesafe 100 kilometre veya daha az

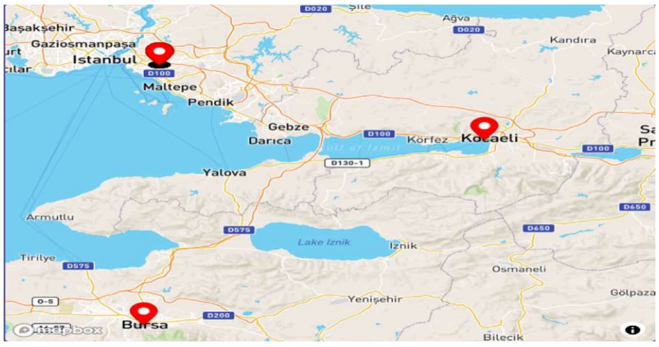
ise, bu etkinlik bir NearbyEventViewModel nesnesi olarak oluşturulup bir listeye eklenir. Bu nesne, etkinliğin adını, konumunu, mesafesini ve coğrafi koordinatlarını içermektedir. Sonuç olarak, bu liste döndürülerek kullanıcıya yakın olan etkinlikler sunulmaktadır.

Kodda yer alan ikinci fonksiyon, CalculateDistance, iki coğrafi nokta arasındaki mesafeyi hesaplamak için kullanılmaktadır. Bu metod, iki noktanın enlem ve boylam bilgilerini alarak, Haversine formülünü kullanarak mesafeyi hesaplar. Dünya'nın yarıçapı 6371 kilometre olarak tanımlanmış ve bu değer hesaplamalarda kullanılmıştır. Mesafe hesaplandıktan sonra, kullanıcı konumuna yakın etkinliklerin belirlenmesinde önemli bir rol oynamaktadır.

Ayrıca, ToRadians adlı bir yardımcı metod da bulunmaktadır. Bu metod, derece cinsinden verilen bir açıyı radyan cinsine dönüştürmektedir. Böylece, mesafe hesaplamaları için gerekli olan matematiksel işlemler doğru bir şekilde gerçekleştirilebilmektedir.



Kullanıcı kaydı başarılı ise Login sayfasına yönlendirilir kullanıcı kendi girişini yaptıktan sonra UserHubArea() ile view yönlendirilir .Anasayda kullanıcı ilk kaydolmuşsa ilgi alanlarına göre eğer önceden katılmış bir etkinliği var ise katılımlarına göre öneri sunulur ilk ilgi alanın tablısndan kategoriidler alınır Katılımı mevcut ise de katilims tablosunda etkinlikıdlere göre ve ilgi alanın tablosunda ilgilere göre öneri sunulur. EtkinlikÖneri service metotduna göre belirli bir kullanıcı için önerilen etkinliklerin listesini oluşturur. İlk olarak, kullanıcının ilgi alanları belirlenir. Bunun için, kullanıcının ilgi duyduğu kategoriler IlgıAlanları tablosundan alınır ve bir listeye dönüştürülür. Ardından, kullanıcının daha önce katıldığı etkinliklerin kategorileri sorgulanır. Bu işlem için Katilimis ve Etkinlikler tabloları birleştirilerek, katıldığı etkinliklerin kategorileri seçilir ve tekrarlı değerler kaldırılarak benzersiz bir liste elde edilir. Kullanıcı eğer beğendiği önerlerden tercih etmek isterse katıl a basar eğer tarih saat konum a göre başka bir etkinliği ile çakışıp çakışmadığı sorgusu yapılır çakışır ise alternatif kendi seçtiği kategorinin kategorId si ve etkinlik tarihi ve konuma göre alternatif sunulup önüne getirilir. Rota etkinlik ilk öneri bakama için açtığında ürün nerde olduğu rotada gözükür ve yakındaki etkinliklerede kullanıcı bakabilir çakışma yaşanırsa da alternatif önerilenlerden seçim sağlayabilir ve seçtikten sonra da katılım belgesi indirilir. Önerilenlerden seçtiği aynı sayfa içerisinde ayrıca o etkinliğe ait yorumları görevbilir bu yorumları Mesaj modelinde EtkinlikId ile nagivation propert ile bir foreign key bağlantısı sağladık.MesajConrolerdaki MesajService içerisinde EtkinlikMesajlarım() kısımanda kullanıcılar etkinliğe yorum yapabilir ve diğer kullanıcılar da bunu görür eğer etkinliğe yorum yapanları ParentId ile de cevaplabilir. Bir nevi form gibi bir ortam oluşturduk. Kullanıcı



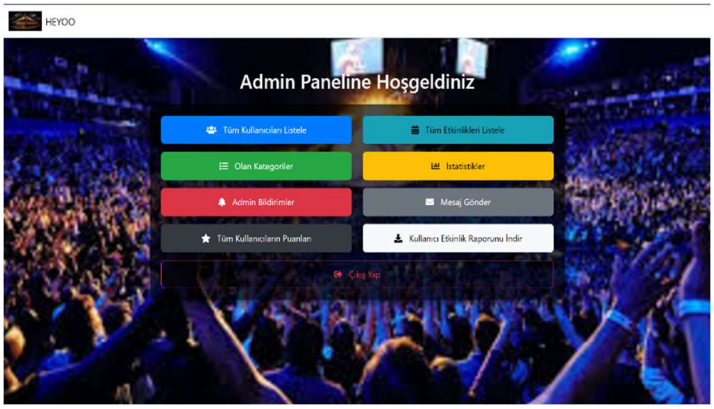
eğer etkinliğe katılmak isterse rota butona basıp araba bisiklet yaya olarak olarak uzaklık hesaplamasına bakabilir bu işlemi de ücretsiz olarak eriştiğimi Mapbox ile gerçekleştirdik mapbox kullanıcın girdiği string olarak konum değerini lng lat olarak koordinatlarını alır bu koordinatlar haritada yerleştirmenizi sağlar ayrıca İlçe bilgisi de girersek konumuzda gösterebiliriz. Kullanıcı konumu ve etkinlik konumu arasındaki mesafede Rota hesaplaması ve görünü viewda c# kodlarını gömerek sağladık. UserHubarea ya geri döndükten sonra başka kullanıcılara Usernameleri girerek onlara özel mesaj gönderebiliriz. Eğer mesaj göndermişsek gönderilen mesajlardan mesajlarımızı görebiliriz gelen mesajları da gelem mesajlar butonunda görebilriz viewları Mesaj viewında contoler içerisinde post ve get isteklerini yerleştirdik daha temiz bir kod elde etmek içinde MesajService metotlarını ekledik burada mesajları kullanıcı göndermemiz için Mesaj modelini kullandık tüm metotlarımız async olarak tanımladık .Profil sayfasında kullanıcı bilgileri mevcut isterse UpdateUser() metotdu ile bu kodu ile bilgilerini güncelleyebilir aynı şekilde ilgi alanın güncellemesi de yapabilir .Userhubare da Puan kısmında kullanıcın puan bilgileri mevcut bu puan hesaplaması PuanHesaplayiciService deki yer alan metotların EtkinlikContolerdaki create postuna HesaplaKatilimPuan: Kullanıcının katıldığı etkinlikleri sorgular. Eğer kullanıcı yalnızca bir etkinliğe katıldıysa 20 puan verir; eğer birden fazla etkinliğe katıldıysa, katıldığı etkinlik sayısı kadar 10 puan (her etkinlik için) hesaplarDönüş Değeri: Kullanıcının katılım puanını (int) döndürür.HesaplaOlusturmaPuanKullanıcının oluşturduğu etkinlikleri sorgular.Her bir oluşturulan etkinlik için 15 puan hesaplarDönüş Değeri: Kullanıcının oluşturma puanını (int) döndürür. HesaplaBonusPuan:Kullanıcının oluşturduğu etkinliklerde belirli bir kategoriye göre bonus puan hesaplar (örneğin, kategori ID'si 1 olan etkinlikler için).Her bonus etkinlik için 5 puan verir. Dönüş Değeri: Kullanıcının bonus puanını (int) döndürür. HesaplaToplamPuan:Üç puan türünü (katılım, oluşturma ve bonus puan) hesaplar. Hesaplanan puanları toplar ve toplam puanı döndürür. Dönüş Değeri: Kullanıcının toplam puanını (int) döndürür. KaydetPuan:Kullanıcının kazandığı puanı veritabanına kaydetmek için kullanılır.KullaniciID, PuanDegeri ve KazanilanTarih içeren yeni bir Puan nesnesi oluşturur.Bu yeni puan kaydını veritabanına ekler ve değişiklikleri kaydeder.Katıla da eklediğim metıolarla puan hesaplanıp kaydedilir.Ayrıca etkinlikte yer Delete metotunda silme işlmei olursa puanı düşülmesi sağlandı. Kullanıcın bildirimlerde adminin kullanıcın oluşturduğu etkinlikleri red ve onaylmasına göre bildiri iletilmesi sağlandı. Bildirimler tablosuna kaydedilip ilgili metot altına yerleştirildi.Bildirim Service GetKullaniciBildirimlerAsync metodu, belirli bir kullanıcıya ait bildirimleri alırken, GetAdminBildirimlerAsync metodu ise sadece admin bildirimlerini sorgulamak için kullanılır. Ayrıca, AddBildirimAsync metodu, belirli bir etkinlikle ilgili bildirim oluşturarak, ilgili bilgileri içeren bir mesaj oluşturur ve bu bildirimi veritabanına kaydeder. Metod, etkinlik ve kullanıcı

bilgilerini alarak, bildirim mesajını dinamik olarak oluşturur. Bildirim silme işlemi için DeleteBildirimAsync metodu bulunmaktadır. Bu metod, belirtilen bir bildirim kimliğine sahip bildirimi bulur ve eğer varsa onu veritabanından siler. Bu sınıf, bildirim yönetimi için temel işlevleri sağlamakta ve kullanıcıların bildirimlerini takip etmelerine, yöneticilerin ise admin bildirimlerini görüntülemelerine olanak tanımaktadır. CheckEtkinlikConflict adlı metod, yeni bir etkinliğin var olan etkinliklerle çakışıp çakışmadığını kontrol etmekte kullanılır. Öncelikle, mevcut kullanıcının adı alınır. Eğer kullanıcı adı bulunamazsa, bir yetkisizlik hatası fırlatılır. Daha sonra, kullanıcının daha önce oluşturduğu etkinlikler, belirtilen tarihteki mevcut etkinliklerle karşılaştırılır. Her etkinlik için etkinliğin bitiş saati, yeni etkinliğin başlangıç saati ile karşılaştırılarak çakışma durumu kontrol edilir. Eğer yeni etkinlik, mevcut bir etkinlik ile çakışıyorsa, true değeri döner; çakışma yoksa false döner.

CreateEtkinlik metodu ise, yeni bir etkinlik oluşturmak için kullanılmaktadır. İlk olarak, yeni etkinlik bilgilerini içeren bir EtkinlikDto nesnesi oluşturulur. Daha sonra, CheckEtkinlikConflict metodu çağrılarak çakışma kontrolü yapılır. Eğer çakışma varsa, bu metot false döner ve etkinlik oluşturulmaz. Çakışma yoksa, yeni etkinlik _context.Etkinlikler koleksiyonuna eklenir ve veritabanına kaydedilir. İşlem başarılı olursa true değeri döner.

Bu sınıf, etkinlik yönetimi uygulamalarında kullanıcıların çakışan etkinlikler oluşturmasını engellemek amacıyla oldukça faydalıdır. Örneğin, bir kullanıcı belirli bir tarihe iki etkinlik eklemeye çalıştığında, bu sınıf mevcut etkinlikleri kontrol ederek bu durumu önleyebilir.

Tüm etkinliklerde ise etkinlikleri silebilir düzenleyebilir ve etkinliğin detaylarını ve hangi kullanıcın o etkinliği oluşturduğunu görebilir. Etkinliğe ait yorumları da görebilir. Admin etkinlikleri onaylama ve reddetme hakkına sahiptir tüm etkinliklerde sayfasında gerçekleştirebilir. Bildirim olarak da kullanıcıya reddetme ya da onaylama bildirim olarak anlık iletilir. Tüm kullanıcıların puanlarını görüntüleyebilir. Admin kategori ekleyebilir silebilir ve düzenleyebilir. Admin kullanıcıların kaç etkinliğe oluşturup katıldığını belge olarak indirip inceleyebilir. StatisticsService, kullanıcı ve etkinliklerle ilgili istatistikleri hesaplamak için kullanılan bir servis sınıfıdır. Bu sınıf, çeşitli yöntemler içerir. İlk olarak, GetUserCountByGender metodu, kullanıcıları cinsiyete göre gruplar ve her cinsiyetteki kullanıcı sayısını döndürür. Sonuç, bir Dictionary<string, int> formatında döner; anahtarlar cinsiyetleri (örneğin, "Erkek" ve "Kadın"), değerler ise o cinsiyetteki kullanıcı sayısını temsil eder.Diğer bir metot olan GetEventCountByCategory, etkinlikleri kategoriye göre gruplar ve her kategorideki etkinlik sayısını döndürür. Bu metodun çıktısı da bir Dictionary<int, int> formatındadır; anahtarlar kategori ID'leri, değerler ise o kategorideki etkinlik sayısını temsil eder. GetUserCountByAgeGroup metodu, kullanıcıları yaş gruplarına göre gruplar ve her yaş grubundaki kullanıcı sayısını döndürür. Kullanıcının doğum tarihine dayanarak yaş hesaplaması yapılır ve sonuç yine bir Dictionary<int, int> formatında döner; anahtarlar yaş grubu, değerler ise o yaş grubundaki kullanıcı sayısını temsil eder. Son olarak, GetUserPoints metodu, her kullanıcının toplam puanını hesaplar. Kullanıcı adı ve toplam puan çiftlerini döndürerek, sonuç bir Dictionary<string, int> formatında sunulur; anahtarlar kullanıcı adları, değerler ise toplam puanlarıdır. StatisticsController ise, kullanıcıdan gelen istekleri yönetir ve StatisticsService sınıfını kullanarak istatistik verilerini toplar ve görüntüler. Index metodu, kullanıcıdan gelen istek üzerine cinsiyet, kategori, yaş grubu ve kullanıcı puanları gibi istatistikleri hesaplar. Bu verileri bir model nesnesine ekler ve ilgili görünüme döner.GetUserEventReport metodu, kullanıcı etkinlik raporunu oluşturur. Kullanıcıların oluşturduğu etkinlik sayısını, kategori ID'sini ve katıldıkları etkinlik sayısını gruplar ve rapor olarak döndürür. Bu işlem, LINQ kullanarak kullanıcı etkinliklerini gruplamak ve ilgili verileri toplamak için yapılır. GenerateUserEventReport metodu, kullanıcı etkinlik raporunu PDF formatında oluşturur. Text kütüphanesini kullanarak bir PDF dosyası oluşturur ve rapor verilerini bu dosyaya ekler. Son olarak, DownloadUserEventReport metodu, kullanıcı etkinlik raporunu oluşturur ve ardından oluşturulan PDF dosyasını kullanıcıya indirilmek üzere sunar. Bu metod, dosya varlığını kontrol eder ve uygun bir yanıt döner. ApplicationDbcontext çalışma biçimi



ApplicationDbContext Sınıfı: ApplicationDbContext sınıfı, IdentityDbContext<User> sınıfından türemektedir. Bu yapı, kullanıcı kimlik bilgilerini depolamak ve yönetmek için gereken tabloların otomatik olarak oluşturulmasını sağlar. Sınıfın yapıcı metodu, veritabanı seçeneklerini alarak üst sınıfın yapıcısına iletir.

GenerateFakeData metodu, belirtilen sayıda sahte kullanıcı oluşturur. Faker kütüphanesini kullanarak kullanıcı bilgilerini rastgele üretir. Kullanıcılar için isim, soyisim, e-posta, telefon numarası, doğum tarihi ve cinsiyet gibi bilgiler oluşturulur. Ayrıca, kullanıcıların konumları belirli ilçelere atanır**.**
Kullanıcı Rolleri

Kullanıcılar oluşturulduktan sonra, User adıyla bir rol tanımlanır. Bu rol, oluşturulan kullanıcılara atanarak, uygulamanın erişim kontrolü ve kullanıcı yönetimi işlevlerini destekler. Rol ataması sonrası, kullanıcıların başarıyla oluşturulup oluşturulmadığı kontrol edilir. Başarı durumuna göre kullanıcı bilgileri loglanır veya hata mesajları döndürülür. CreateRandomInterestAreasForUser: Bu metod, her bir kullanıcıya rastgele ilgi alanları atar. Belirli varsayılan kategoriler (örneğin, sanat, spor, teknoloji) oluşturulur ve bu kategoriler veritabanına eklenir. Her kullanıcıya üç tane rastgele ilgi alanı atanır, bu sayede kullanıcıların ilgi duyabileceği etkinliklerin belirlenmesi sağlanır.

CreateRandomEventsForUser: Kullanıcının ilgi alanlarına uygun etkinlikler oluşturur. Her ilgi alanı için etkinlik adları, açıklamaları ve tarihleri rastgele belirlenir. Bu etkinlikler, kullanıcıların ilgi alanlarıyla örtüşecek şekilde tasarlanır, böylece kullanıcılar için anlamlı içerikler sunulur.AddParticipantForInterest: Bu metod, kullanıcıların ilgi alanlarına göre etkinliklere katılımcı olarak eklenmesini

sağlar. Kullanıcılara atanan ilgi alanları üzerinden rastgele etkinlikler seçilir ve her etkinliğe kullanıcı katılımcı olarak eklenir. Bu sayede, etkinliklerin dinamik bir şekilde kullanıcılar tarafından sahiplenilmesi sağlanır.GenerateFakePuan: Bu metod, veritabanındaki tüm kullanıcılar için sahte puanlar oluşturur. Kullanıcılara belirli bir puan aralığından rastgele puan değerleri atanır. Oluşturulan bu puanlar, kullanıcıların etkinliklere katılımını ve ilgi alanlarını destekleyici unsurlar olarak veritabanına kaydedilir.CreateRandomCommentsForEvent: Bu metod, etkinliklere rastgele yorumlar ekler. Tüm kullanıcılar arasından rastgele seçilen kullanıcılar, belirli etkinlikler hakkında geri bildirimde bulunur. Bu, etkinliklerin etkileşimini artırır ve topluluk geri bildirimini teşvik eder.

A. Deneysel Sonuçlar

Projemizi ilk .Net Core api de backend ;react de de front yazdık ama sonrasında backend tarafında cookie kullanımı front tarafında çok fazla hata alımına sebebiyet verdi. O yüzden Mvc projesine geçilfdi.Public private bir erişim konrolu hata alındığında 404 hastası aldığım Dı hatası aldığımı ve yetkilendirmemde problem olduğunu düşündüm fakat metodu diğer metotlara private tanımladığım için erişim sorunu yaşadım.

B. Sonuç

Projemizde çoğu istere uygun çalışıyor.

Başla

Fonksiyon: Register(UserRegisterModel model)
Yeni bir kullanıcı bilgisi alınır.
Kullanıcı bilgileri ve şifre doğrulanır.
Kullanıcının konumu için koordinatlar hesaplanır.
Giriş yapan kullanıcı için uygun etkinlik önerileri getirilir.Eğer öneriler bulunamazsa:
Hiçbir etkinlik önerisi bulunamadı." mesajı gösterilir.
Eğer öneriler varsa, bu etkinlikler listelenir.
OneriController İşlevleri
Kullanıcının ilgi alanları ve katıldığı etkinliklere göre öneriler oluştur,
Önerileri tarih sırasına ve katılım sıklığına göre sırala
EtkinlikController İşlevleri
Fonksiyon: Index()
Kullanıcı tarafından oluşturulan etkinlikleri ve katıldığı etkinlikleri getirListeyi DataGrid'de göster
Fonksiyon: Rota(etkinlikId)
Kullanıcı ve etkinlik konum bilgilerini al
Kullanıcıya rota detaylarını göster
Fonksiyon: Katil(etkinlikId)
Kullanıcıyı etkinliğe katılmak üzere ekle
Eğer zaman çakışması varsa:
Alternatif etkinlikler öner
Etkinlik bilgileri ile PDF katılım belgesi oluştur ve indir
Fonksiyon: Create(Post, yeniEtkinlik)
Yeni etkinlik bilgilerini veritabanına kaydet
Eğer başarılıysa:
"Etkinlik oluşturuldu." mesajını göster
Aksi durumda:
"Oluşturma başarısız." mesajını göster
BildirimController İşlevleri
Fonksiyon: KullaniciBildirimler()
Kullanıcının bildirimlerini listele
Bildirimleri görünümde göster
Fonksiyon: AdminBildirimler()
Admin
için tüm bildirimleri listele
Fonksiyon: AdminRegister()
Yeni bir Admin kullanıcı oluştur
Kullanıcıyı veritabanına kaydet
Fonksiyon: TumKullanicilar()
Tüm kullanıcıları listele (Admin hariç)
Eğer kullanıcı yoksa:
"Kullanıcı bulunamadı." mesajını göster
Kullanıcıları DataGrid'de listele
Fonksiyon: KullaniciDetay(id)
Kullanıcıyı ve ilgili etkinlikleri veritabanından getir
Kullanıcı detaylarını görünümde göster

Bİttİ

Kaynaklar

[1] https://www.gencayyildiz.com/blog/
[2] https://dotnet.microsoft.com/en-us/learn
[3] https://www.youtube.com/watch?v=hYv6BM2fWd8&t=3205s
[4] https://www.mapbox.com/maps