



BİTİRME PROJESİ RAPORU

GezBot

Bilgisair

Harun Serkan Metin

Hüseyin Pekkan

Arda Erol

Rana Demir

BİLGİSAYAR MÜHENDİSLİĞİ VE YAPAY ZEKA
MÜHENDİSLİĞİ BÖLÜMÜ

İçindekiler

1.PROJENİN AMACI	3
2.HEDEF KULLANICI KİTLESİ.....	3
3.KULLANICI SENARYOLARI & GEREKSİNİM ANALİZİ	3
3.1. Kullanıcı Senaryoları	3
3.2. Gereksinim Analizi.....	4
4.SİSTEM FONKSİYONLARI.....	5
5.SİSTEM MİMARİSİ.....	10
6.SİSTEM UNSURLARININ İLİŞKİLERİ	12
7.SİSTEM UNSURLARININ DETAYLI AÇIKLAMASI	14
7.1Gezbot Flutter Yapısı.....	15
7.2Firebase Database Yapısı.....	21
7.3Gezbot FastAPI Backend Yapısı.....	25
8.TAKIM İÇERİSİNDEKİ İŞ PAYLAŞIMI.....	27
• Harun Serkan Metin	27
• Hüseyin Pekkan	27
• Arda Erol	28
• Rana Demir	29
9. REFERANSLAR.....	29
10.FİNAL DEMO VIDEO LİNKİ	29

1. PROJENİN AMACI

Geliştirmiş olduğumuz GezBot uygulaması, bir seyahat uygulaması olup insanların seyahatleri için en iyi ve uygun seçenekleri, onların belirttiği tercihlere bağlı olarak yapmayı amaçlar.

2. HEDEF KULLANICI KİTLESİ

Yalnız Türkiye değil tüm dünya genelinde de seyahat etmekten keyif alan çok sayıda insan var. Fakat gerek bütçelerinin eşit olmaması gerek ilgi alanlarının farklı olması gibi sebepler nedeniyle bu insanların yapacağı seyahat planları da birbirinden oldukça farklı seyredebiliyor. Fakat bu planlar ne kadar birbirinden bağımsız olsa da bu grupta sayılan insanların ortak paydada bulunduğu bir nokta var: Kendilerine her yönden en uygun planı yapabilmek. İşte bu yüzden kişiye özel önerilere duyulan ihtiyaç gün yüzüne çıkıyor. Bu noktada gezBot devreye girerek pek çok kritere uyumu gözetiyor. Tatil planı yapacak kişinin nereden nereye gideceğinin yanı sıra hangi ulaşım türünü tercih ettiği, ayırabileceği bütçesi, seyahatinde ana amacın ne olduğu, spesifik konaklama tercihleri gibi kişinin özel ilgi alanları da göz önünde bulundurularak kullanıcı memnuniyeti hedeflenmektedir.

Bu proje için belirlenen kitle; seyahat planları yapmada bütçesi sebebiyle güçlük çeken fakat belli ölçüde yapabilecek durumda olan, kalınacak yerleri veya ulaşımın nasıl sağlanacağı konusunda araştırmalar yapmakta zorlanan veya yeterince zaman ayıramayacak olan, enteresan ve eşsiz zevklere sahip fakat buna uyan mekânlar bulmakta zorluk çeken, başka insanların nerelere gittiğine bakıp kendi seyahatlerinde bundan ilham almak isteyen, kendi aracıyla belirli bir lokasyona doğru gittiği yol üstünde ilgi alanına yönelik yerleri ziyaret etmeyi planlarına dahil eden insanlardır. Kısacası, kendi zevklerine erişebileceği ve pek çok kısıtlamayı dahil ederek herhangi bir seyahat gerçekleştirmek isteyen kişilerdir.

3. KULLANICI SENARYOLARI & GEREKSİNİM ANALİZİ

3.1. Kullanıcı Senaryoları

Aniden evinden başka bir şehre gidip oranın yöresel lezzetlerini tatmak, giderken yol üstündeki tarihi yerleri gezmek ve varacağı şehirde birkaç gün konaklamak isteyen biri, projenin muhtemel kullanıcısıdır.

- Kullanıcı olarak, herhangi bir arkadaşımın gittiği yeri görürsem oraya gitmek isteyebilirim. Arkadaşım da gezBot uygulamasını kullanıyorsa onu takip edip seyahatlerine bakabilirim. Böylece nereye gideceğimi bilemediğim bir durumda bana fikir vermiş olur.
- Kullanıcı olarak, temelde isteyeceğim şey, varış noktası belirlemek olur. Bu sayede navigasyon şeklinde işleyecek bir yol tarifi alıp nereden gideceğimi görmüş olurum.
- Kullanıcı olarak, varacağım yere hangi vasıtayla (özel araç, uçak, otobüs veya tren gibi) gitmem gerektiğini seçerim. Böylece neyle gideceğime bağlı olarak ileriki adımlar için uygulama bana yardımcı olabilir. Örneğin bu, benim şahsi aracımса gidilecek rotanın belirlenmesi sağlanır.

- Kullanıcı olarak, bir sonraki adımda bütçemi belirtirim. Yüksek miktarda para harcamak istemeyebilirim ve bunun için de ulaşım ücretim, konaklamaya ödeyeceğim ücret ve yiyecek-içecek masrafı da dahil olmak üzere tüm giderlerimin belli bir seviyeden aşağıda olmasını isteyebilirim. Uygulama da bana bunu sağlayarak masraflarımı minimum düzeye indirebilir.
- Kullanıcı olarak, gideceğim yerde asıl aktivitenin ne olacağını belirtmek adına seyahat amacımı söyleyebilirim. Böylece uygulama, bana orada gideceğim mekânların tipini (bu durumda lokanta gibi yemek yenecek yerler ağırlıkta olacaktır) benim için belirler ve bana opsiyonlar sunabilir.
- Kullanıcı olarak, bu süreçte kalmak isteyeceğim konaklama alanının nasıl bir yer olacağı seçimini yaparım. Bu sayede lüks veya bütçe dostu veya farklı (Airbnb gibi) yerlerde kendi seçimime uygun olarak birçok opsiyon görebilirim ve seçimim haricindeki seçeneklere bakmakla vakit kaybetmem.
- Kullanıcı olarak, seyahatimin ana planlaması haricinde yapmak istediğim ekstra aktivite ve yaşamak istediğim farklı tecrübeler varsa bunlar için isteğimi ifade ederek planımın daha da spesifikleşmesini sağlayabilirim.
- Kullanıcı olarak, çeşitli gıdalar tüketmek için gittiğim mekânlarda beslenme tercihim seçerek özellikle bazı tatlarla karşı kesin bir sınıırım varsa bu sınırı koruma fırsatı yakalarım. Çünkü uygulama, bu seçenekleri göz önünde bulundurarak bana hiç önermemesi gereken yerler varsa bunları önermez. Örneğin vejetaryen birine et lokantası önermek gibi bir hataya düşmez.
- Kullanıcı olarak, seyahatimi yalnız mı yoksa başka kişilerle mi gerçekleştireceğimi cevapladığım takdirde yine yer önerilerine bu tercihlerim şeffaf bir şekilde yansıtacaktır.
- Kullanıcı olarak, tüm seçimlerim ve isteklerim haricinde belirtmem gereken özel durumlar (yürüme engelli olup buna uygun yerler bulmak gibi), daha spesifik tercihler varsa bunları belirterek tamamen kendime odaklanabilmeyi sağlarım.
- Kullanıcı olarak, oluşturduğum bir seyahat planı hakkında sorum olduğunda bunları sorup belli konularda bilgi alabilmek için seyahatim üzerinden mesajlaşma bölümünü açabilir, botla sohbet edebilirim. Böylece daha bilinçli bir şekilde kaliteli vakit geçirebilirim.
- Kullanıcı olarak, bu deneyimim haricinde başka bir deneyim yaşamak istersem varış noktalarını arayarak farklı konumlar bulabilir, buralara şans verebilirim.

3.2. Gereksinim Analizi

Projede ele alınan sorunun çözümü, kişiye özel seyahat planlarının detaylandırılması ve ardından önerilen konaklama ve mekanların belirli kriterlere göre değerlendirilmesidir. Bu kritik süreç, seyahat türünün belirlenmesiyle başlar ve ardından sunulan cevaplar, yapay zeka modelleri aracılığıyla hem arama kriterlerinin daraltılmasında hem de istenen seyahatin şekillendirilmesinde kullanılır.

Kullanıcının tercihlerine göre özelleştirilmiş bir gezi planı oluşturulmadan önce, kullanıcıya yöneltilen sorular sayesinde gezinin niteliği açıkça belirlenir. Bu sorular, yapay zeka tarafından

işlenerek kullanıcının isteklerine uygun seyahat önerileri oluşturulmasını sağlar. Bu süreçte, kullanıcının istemediği mekanların ve otellerin elemesi de yapılır ve daha iyi sorguların çıkarılabilmesi için yapılan işlemler kaydedilir ve modelin daha fazla veriyle güçlendirilmesi için geri besleme yapılır.

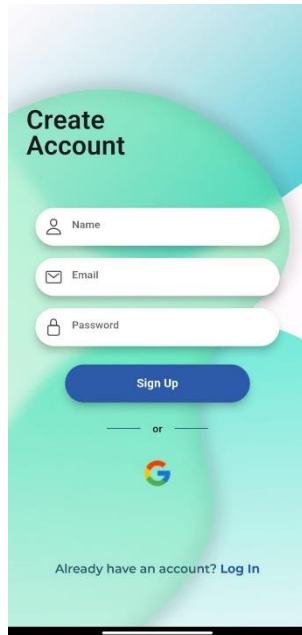
Önerilen otel ve mekanların değerlendirilmesi için belirlenen kriterler, kullanıcının tercihlerine, bütçesine ve seyahat amacına göre şekillenir. Bu kriterler, kullanıcının memnuniyetini en üst düzeye çıkarmak için dikkatle belirlenir ve yapay zeka algoritmaları tarafından sürekli olarak optimize edilir. Bu sayede, kullanıcının seyahat deneyimi kişiselleştirilir ve beklentilerini karşılayacak en iyi seçenekler sunulur.

Sonuç olarak, gezBot'un kullanıcılarına özel ve tatmin edici seyahat deneyimleri sunabilmesi için yapay zekâ modellerinin doğru şekilde kullanılması ve sürekli olarak geliştirilmesi önemlidir. Bu sayede, kullanıcılar ihtiyaçlarına uygun en iyi seçeneklere kolayca erişebilir ve keyifli bir seyahat deneyimi yaşayabilirler.

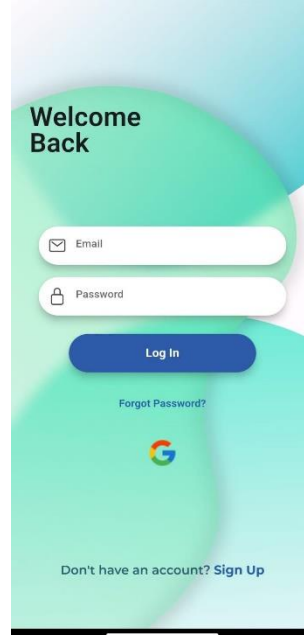
4. SİSTEM FONKSİYONLARI

Uygulama açıldığında kayıtlı bir hesap olmadığı durumda, kullanıcıyı karşılayan sayfalar şunlardır:

1. **Hesap Oluşturma Sayfası:** Kullanıcının yeni bir hesap oluşturabileceği alan.
2. **Oturum Açma Sayfası:** Mevcut bir hesapla oturum açma imkânı sunan alan.

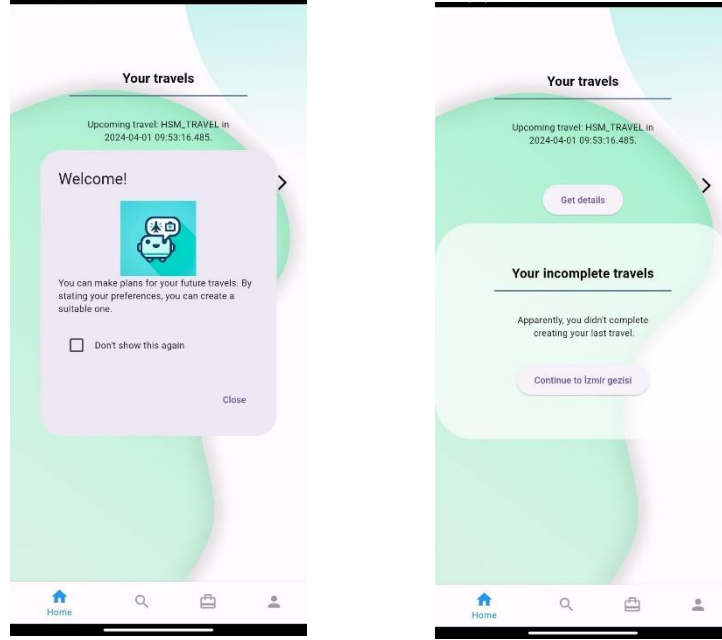


The 'Create Account' screen features a light blue background with a green circular graphic. It includes input fields for 'Name', 'Email', and 'Password', each with a corresponding icon (person, envelope, and lock). A blue 'Sign Up' button is positioned below the fields. Below the button, there is a link that says 'or' followed by the Google logo. At the bottom, a link reads 'Already have an account? Log In'.

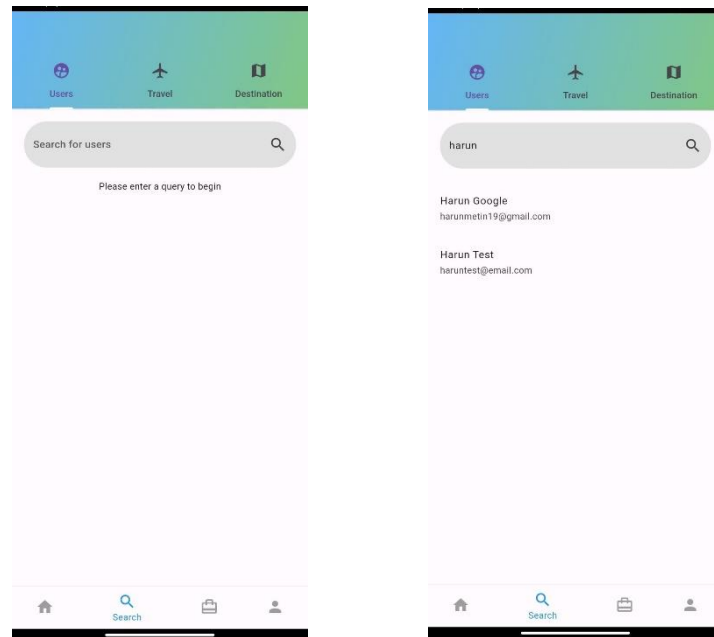


The 'Welcome Back' screen has a light blue background with a green circular graphic. It features input fields for 'Email' and 'Password', each with an icon (envelope and lock). A blue 'Log In' button is located below the fields. Below the button, there is a link that says 'Forgot Password?'. At the bottom, a link reads 'Don't have an account? Sign Up'.

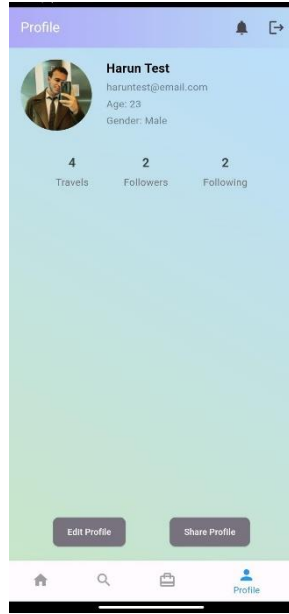
Giriş yapıldıktan sonra “Home Page” kısmı kullanıcıyı karşılar. Bu kısımdan yaklaşmakla olan seyahatlerini ya da tam olarak oluşturmadığı yani yarıda bıraktığı seyahatlerini görebilir.



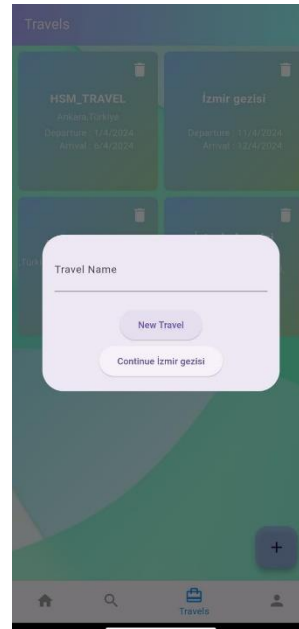
En alt kısımdaki navigasyon çubuğu kullanılarak bir diğer sekme olan Arama kısmına geçebilir, bu kısımda ise hem kullanıcı adına hem oluşturulan seyahat adına hem de varış noktasına göre arama yapılarak istenen sonuçlara ulaşılabilir.



Navigasyon çubuğu kullanılarak bir diğer sekme olan Profil kısmına geçebilir, bu kısımda kullanıcı profili görüntülenerek düzenleme yapılabilir, bekleyen arkadaşlık istekleri görüntülenebilir, takipçiler ve takip edilenler listelenebilir, kullanıcı hesabından çıkış yapılabilir.



Kullanıcı, Navigasyon çubuğu kullanılarak bir diğer sekme olan “Travels” kısmına geçebilir. Bu kısımda tüm seyahatler listelenmektedir istenirse sağ altta bulunan “+” butonundan yeni bir seyahat oluşturulabilir. Eğer oluşturulması yarım kalmış bir seyahat planı varsa direkt bu plandan devam edilebilmektedir.



Yeni seyahat yaratırken kullanıcıyı cevaplaması gereken 13 soru bulunmaktadır. Bunlar :

1. "Kalkış noktanız neresi?"
2. "İstediğiniz varış noktası nedir?"
3. "Gidiş tarihiniz nedir?"
4. "Dönüş tarihin ne?"
5. "Varış noktanıza nasıl seyahat etmeyi planlıyorsunuz?"
6. "Bu gezi için tahmini bütçeniz nedir?"
7. "Seyahatinizin asıl amacı nedir?"
8. "Belirli bir konaklama tercihiniz var mı?"
9. "Özellikle dahil etmek istediğiniz herhangi bir etkinlik veya deneyim var mı?"
10. "Herhangi bir diyet kısıtlamanız veya tercihiniz var mı?"
11. "Yalnız mı yoksa başkalarıyla mı seyahat ediyorsunuz? Başkalarıyla birlikteyseniz grubunuzda kaç kişi var?"
12. "Herhangi bir özel hizmete ihtiyacınız var mı? (ör. erişilebilirlik ihtiyaçları, çocuk dostu tesisler)"
13. "Konaklamanız sırasında gerçekleşen yerel etkinlikler veya aktivitelerle ilgili öneriler ister misiniz?"

1 of 13 questions

4 of 13 questions

5 of 13 questions

6 of 13 questions

7 of 13 questions

8 of 13 questions

9 of 13 questions

10 of 13 questions

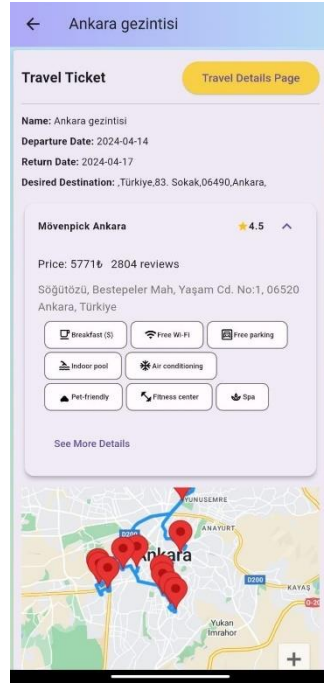
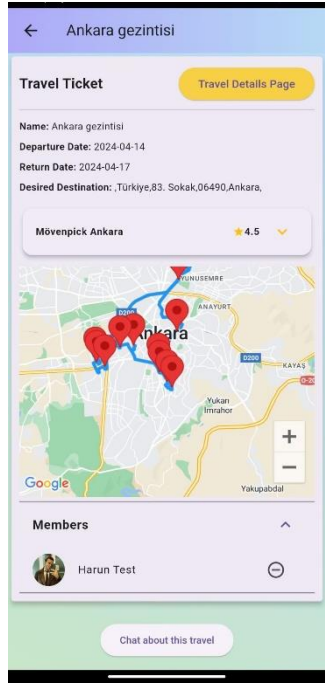
11 of 13 questions

12 of 13 questions

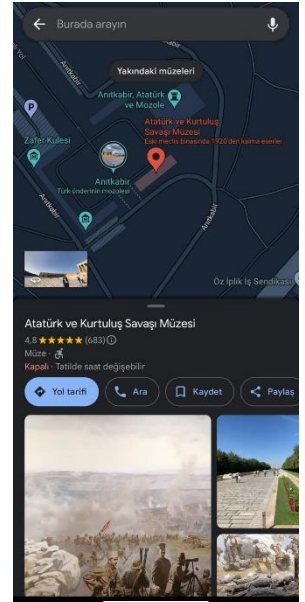
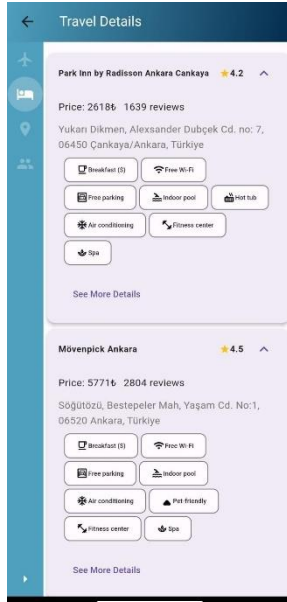
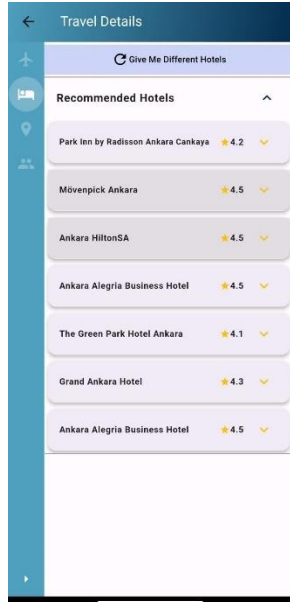
13 of 13 questions

Seyahat yaratıldıktan sonra kullanıcı için en iyi oteller ve mekanların bulunması 2-5 dakika almaktadır bu süre boyunca kullanıcının sayfada beklemesine gerek yoktur. Her sonuç bulunduktan sonra “Travel Details Page” kısmına eklenmektedir. “Travel Ticket Page” kısmında seyahat hakkında genel bilgiler bulunmaktadır.

Ayrıca kullanıcı için seçilmiş en uygun otel ve kullanıcı için özel hazırlanmış mekan önerileri Haritada işaretlenerek rotalanmış şekilde mevcuttur. Kullanıcı dilerse otel adının üstüne tıklayarak açılan kısımdan otel olanaklarına detaylıca ulaşabilir. Rezervasyon yapılmak istenirse “See More Details” butonuna tıklayarak otel sayfasına ulaşarak rezervasyon yapabilir.

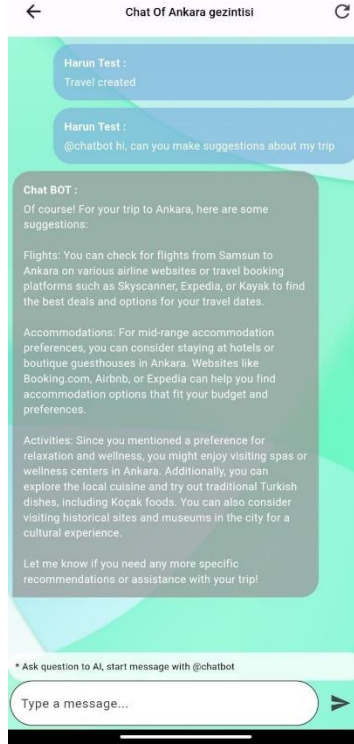


“Travel Details Page” butonuna tıklanarak belirtilen tarihler arasındaki uçak biletleri, konaklama imkanları ve mekanların listesi görüntülenebilir. Kullanıcı, her bir pencere için beğenmediği önerilerin başka bir öneriyle değiştirilmesini üstteki “Give Me Different” butonuyla isteyebilir. Ayrıca her mekanın üstüne



tıklayarak o mekanın konumunu ve kenardaki google maps iconuna tıklayarak yol tarifi alabilir ya da mekanın sayfasını inceleyebilir.

Kullanıcı isterse bu seyahatte bulunan diğer kişilerle mesajlaşabilir. Yada Chatbot ile mesajlaşarak bir problemle karşılaştığı zaman çözmeye çalışabilir.



5. SİSTEM MİMARİSİ

GezBot, kullanıcıların seyahatlerini keşfetmelerine, planlamalarına ve yönetmelerine olanak tanıyan güvenilir bir platform olarak hizmet vermektedir. FastAPI ile güçlendirilmiş sağlam bir arka uç ve Flutter kullanılarak geliştirilen sezgisel bir ön uç ile uygulama, kesintisiz bir kullanıcı deneyimi sunar. Bu rapor, kullanıcı yönetimi ve AI destekli öneri özelliğine odaklanarak sistemin temel bileşenlerinin bir özetini sunmaktadır.

5.1. Sistem Mimarisi

Flutter kullanılarak geliştirilen ön uç, Android, iOS ve web tarayıcıları da dahil olmak üzere çeşitli cihazlarda çok platformlu, duyarlı bir tasarım sunar. Bu sayede, kullanıcılar istedikleri cihaz üzerinden GezBot'a erişebilir ve seyahat planlarını kolayca oluşturabilirler. Ayrıca, kullanıcı dostu bir arayüz sayesinde, gezilerini kişiselleştirebilir ve tercihlerine göre öneriler alabilirler.

Kullanıcı Profili ve Etkileşim: Kullanıcılar, profillerini oluşturabilir ve diğer kullanıcılarla

etkileşime geçebilir. Bu özellik sayesinde, kullanıcılar seyahat deneyimlerini paylaşabilir, önerilerde bulunabilir ve birlikte seyahat etmek isteyen kişilerle iletişime geçebilirler. Bu, GezBot'un kullanıcıları arasında bir topluluk oluşturmasını sağlar.

Arka uç, FastAPI kullanılarak güçlendirilmiştir. HTTP isteklerini işlemek için FastAPI kullanır ve veri alımı, manipülasyonu ve depolanmasını etkin bir şekilde gerçekleştirir. Arka uç, kullanıcıların seyahat planlarını oluşturmaya ve önerilen mekanları keşfetmelerine olanak tanır.

- **AI Destekli Öneriler:** Google API ve özel olarak geliştirilmiş yapay zeka modelleri, kullanıcılara kişiselleştirilmiş seyahat önerileri sunar. Bu öneriler, kullanıcıların tercihlerine, bütçesine ve seyahat amacına göre şekillenir. Örneğin, kullanıcıların ilgi alanlarına göre önerilen etkinlikler, restoranlar ve konaklama seçenekleri sunulur.
- **Gerçek Zamanlı Veri:** Gerçek zamanlı seyahat verilerini almak için dış API'ler ve web kazıma fonksiyonları entegre edilmiştir. Bu sayede, kullanıcılar güncel bilgilere erişebilir ve seyahat planlarını daha iyi bir şekilde planlayabilirler.

5.2. Dizin Yapısı

lib/pages: Kullanıcı navigasyonunu sorunsuz hale getiren çeşitli Dart ekran dosyalarını barındırır.
lib/components: Uygulama genelinde görsel tutarlılığı artıran yeniden kullanılabilir UI bileşenlerini içerir.

lib/models: Uygulama durum yönetimi için veri yapılarını tanımlar.

lib/services: API iletişimi de dahil olmak üzere iş mantığını yönetir.

lib/utills: Kodun bakımını ve yeniden kullanılabilirliğini kolaylaştıran yardımcı fonksiyonları sağlar.

5.3. Python Modülleri

main.py: API uç noktalarını yönetir ve arka ucun omurgası görevini görür.

scrapper.py ve flight.py: Gerçek zamanlı seyahat verilerini almak için dış API'ler ve web kazıma fonksiyonlarını entegre eder.

models.py: Veri modellerini veritabanı şemalarına eşleyerek veri etkileşimini kolaylaştırır.

ai_main.py ve ai_models.py: Kişiselleştirilmiş önerileri güçlendiren AI algoritmalarını içerir.

Firestore ile Kullanıcı Yönetimi

Firestore, kullanıcı hesaplarını ve kimlik doğrulamayı yönetmek için kullanılır. Güvenli bir giriş sistemi sağlar ve kullanıcı profillerini ve seyahat girişlerini sürdürmek için Firestore veritabanı ile etkileşimde bulunur.

5.4. Firestore Özellikleri

Kimlik Doğrulama: Firestore, kullanıcı hesaplarını ve kimlik doğrulamayı yönetmek için kullanılır. Güvenli bir giriş sistemi sağlar ve kullanıcı profillerini ve seyahat girişlerini sürdürmek için Firestore veritabanı ile etkileşimde bulunur. Firestore, GezBot'un kullanıcılarına güvenli ve güvenilir bir platform sunmasını sağlar.

Firestore Veritabanı: Kullanıcı bilgilerini ve seyahat planlarını gerçek zamanlı olarak depolar ve

senkronize eder, veri tutarlılığını sağlar.

AI Destekli Arka Uç : Kullanıcı etkileşimini zenginleştirmek için tasarlanan AI bileşeni, seyahat seçenekleri önerir ve benzer ilgi alanlarına sahip kullanıcıları bulur.

5.5. AI İşlevleri

Seyahat Önerileri: Kullanıcı tercihlerini ve geçmiş seyahatleri analiz ederek kişiselleştirilmiş seyahat planları önerir.

Benzer Kullanıcı Eşleştirme: Topluluk atmosferini teşvik ederek, ilgi alanlarına göre kullanıcıları bulan makine öğrenimi algoritmalarını kullanır.

Güvenlik Hususları: API anahtarları ve kullanıcı kimlik bilgileri gibi hassas veriler, uygulama güvenliğine uygun şekilde ortam değişkenlerinde saklanır ve kod tabanına veya versiyon kontrol sistemine gömülmemiştir.

6. SİSTEM UNSURLARININ İLİŞKİLERİ

Sistem, veritabanı ve diğer modüller ile ayrı ayrı iki adet sunucuda çalışmaktadır. Veritabanı Firebase seçildiği için ayrıca kendimizin bir sunucu ayağa kaldırmasına gerek yoktur fakat Yapay Zekâ modeli veya mekanların getirilmesi, otellerin Web Scrapping yöntemiyle alınması gibi durumlar için Python dilinde yazılmış Fast API backendimizin ayağa kaldırılması gerekmektedir.

CRUD işlemleri ve Backend ile iletişim için Önyüz kısmı olan Flutter içinde Dart dilinde yazılmış 4 farklı servis bulunmaktadır. Bunlar:

- **Backend Service**: FastAPI backend ile Önyüzün iletişim kurmasını sağlayan katmandır,
- **Google Cloud Service**: Önyüzü içinde ufak çaplı Google Maps API kullanılması gereken yerlerde backende bağımlı olmadan Google places Api ye sorgu atabilen servistir,
- **Database Service**: Çok büyük bir yapıya sahip olan bu servis tüm Firebase Database işlemlerinin yapıldığı katmandır,
- **User Service**: User Auth işlemleri ve Google Hesabi ile giriş gibi işlemleri yapan servistir

FastAPI Backend işlevleri için 4 farklı yapı bulunmaktadır. Bunlar:

- **GptAPI**: Prompt oluşturma ve Google Api de aratılmak üzere filtreleri ve queryleri hazırlar
- **GoogleAPI**: Place Search kısmını 2 farklı yöntemle yapar sonra bunları birleştirerek database e yazar
- **HotelScraper**: Verilen girdiye göre Selenium kullanarak Google Travels sitesinden anlık Web verisi çekmeye yarar
- **FlightScraper**: Verilen girdiye göre Google Flights bilgilerini Web'ten anlık çekmeye yarar

Gezbot, Seyahat oluşturma işlemleri dışında kendi kendine çalışabilen mobil bir uygulamadır. Oluşturulmuş seyahatleri görmek, yeni mekân önerileri yapılmasını istemek, gezi rotasını

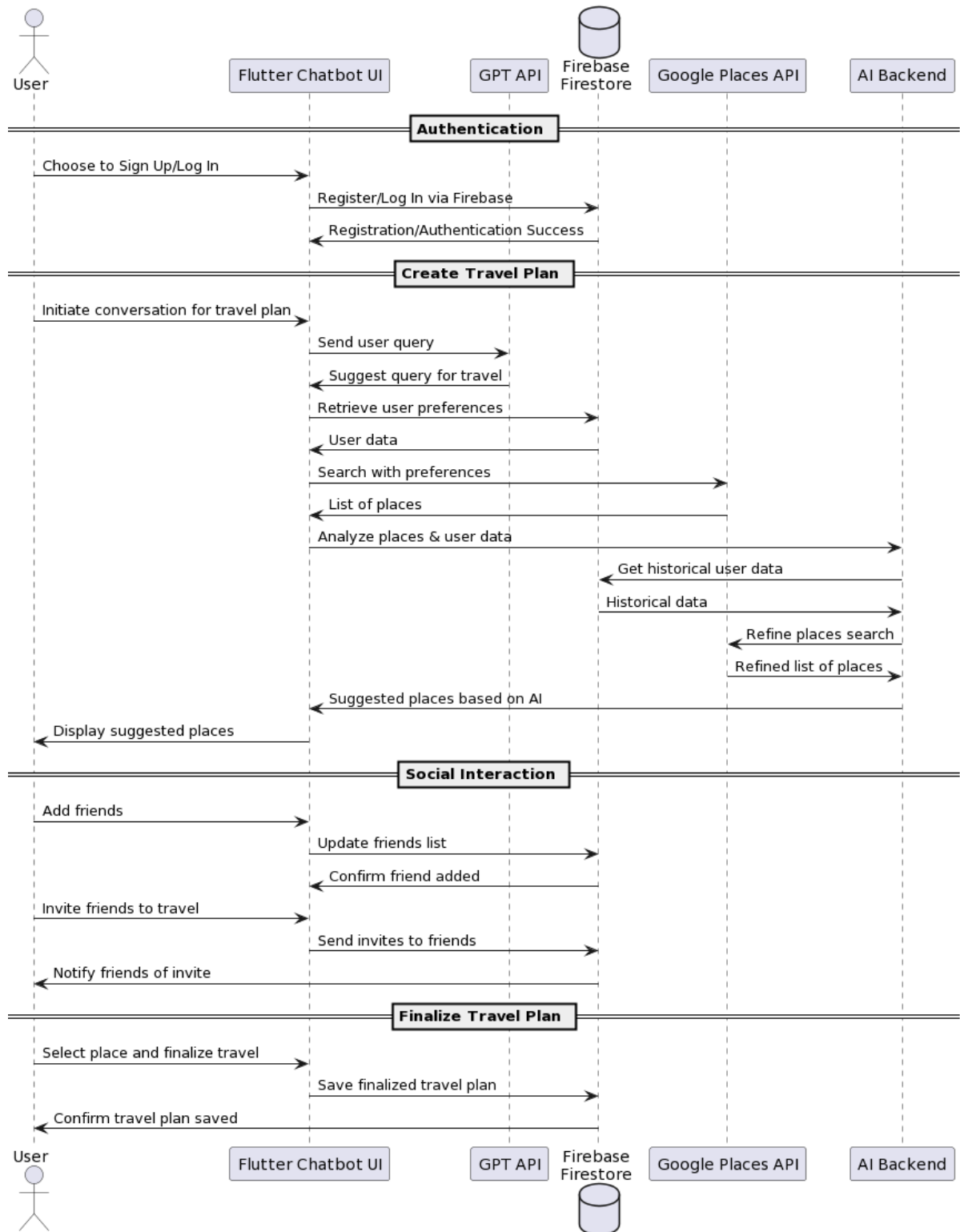
değiřtirmek, insanlarla mesajlařmak, Chatbotla yazıřmak iin ayrıca bir Backend'in ayađa kaldırılmasına ihtiya yoktur. Bu avantaj, Firebase tercih edilmesinin de temel nedenidir. Flutter iinde iyi bir řekilde tanımlanmıř ve kullanılmıř olan kendi “*Service*” kodlarımız bu karmařık iřlerin ođunu halletmektedir.

Seyahat oluřturma iřlemi, seyahat hakkında sorulan 13 tane sorunun hepsi tamamlanınca Backend server'a Frontend tarafından istek gider Bunlar; Find Flights, Find hotels ve Find Places istekleridir. Sırasıyla; Flights iin scrapping yapıldıktan sonra Veritabanına kaydedilir, Oteller iin scrapping yapıldıktan sonra Veritabanına kaydedilir, Google Places Api kullanılarak 2 farklı yolla belirtilen filtrelerle arama yapılır, yapılan aramalardan dnen mekân sonuları (yaklařık 400 farklı mekân) Yapay Zekâ modeline verilerek kullanıcıya en uygun olanın bulunması sađlanır. Sonra bu bulunan uygun mekanların 7 tanesi alınarak kullanıcıya gsterilir. Kullanıcı eđer sunulan yerleri beğenmediyse, mekanları yeniden getir diyebilir; bu durum tm dngy bařtan tetiklemek yerine kiřiye uygun mekanları seen AI modeli, yeniden farklı parametrelerle alıřtırılarak daha uygun sonular elde edilmeye alıřılır.

Gezbot, kullanıcıların seyahat planlarını kolayca oluřturabilecekleri, mekân nerileri alabilecekleri ve seyahat rotalarını kiřiselleřtirebilecekleri interaktif bir platform sunar. Seyahat planlama iřlemi, kullanıcıdan alınan girdilere dayanarak otomatikleřtirilmiř bir dizi sorgulama ve analizle gerekleřtirilir. Bu sre, uuř ve otel bilgilerinin toplanmasından, kullanıcıya uygun mekân nerilerinin yapay zeka modeli ile belirlenmesine kadar uzanır.

Sonu olarak, Gezbot, geliřmiř teknolojilerin ve kullanıcı merkezli tasarımın birleřimiyle, seyahat planlamayı kiřiselleřtirilmiř, etkileřimli ve zahmetsiz bir deneyime dnřtren yeniliki bir mobil uygulamadır. Bu sistem, kullanıcıların beklentilerine gre řekillenen ve srekli geliřen bir seyahat asistanı olarak hizmet verir.

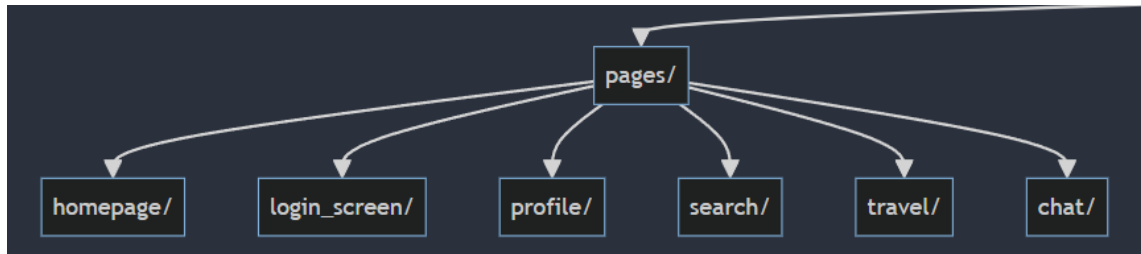
7. SİSTEM UNSURLARININ DETAYLI AÇIKLAMASI



7.1 Gezbot Flutter Yapısı

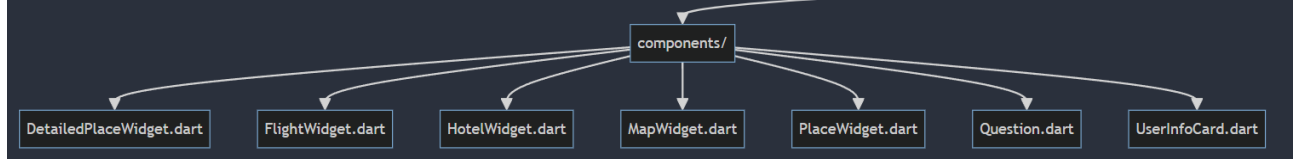


main.dart: Flutter'da genellikle main.dart dosyası olarak başlar ve uygulamanın köküdür. “main.dart” dosyası, gezBot uygulamasının başlangıç noktasıdır ve Flutter widget'ları, Firebase servisleri ve çevre değişkenlerinin başlatılmasını yönetir. Kullanıcı oturum açma durumuna göre yerel depolama üzerinden kontrol sağlar ve isLoggedIn durumuna göre kullanıcıyı ya giriş ekranına (LoginScreen) ya da ana sayfaya (HomePage) yönlendirir. Uygulama içi navigasyon için adlandırılmış rotalar tanımlar, bu sayede kullanıcılar arasında sohbet bilgileri gibi parametrelerle sayfalar arası geçiş yapılabilir. Bu yapı, uygulamanın genel akışını ve kullanıcı deneyimini temel alır.



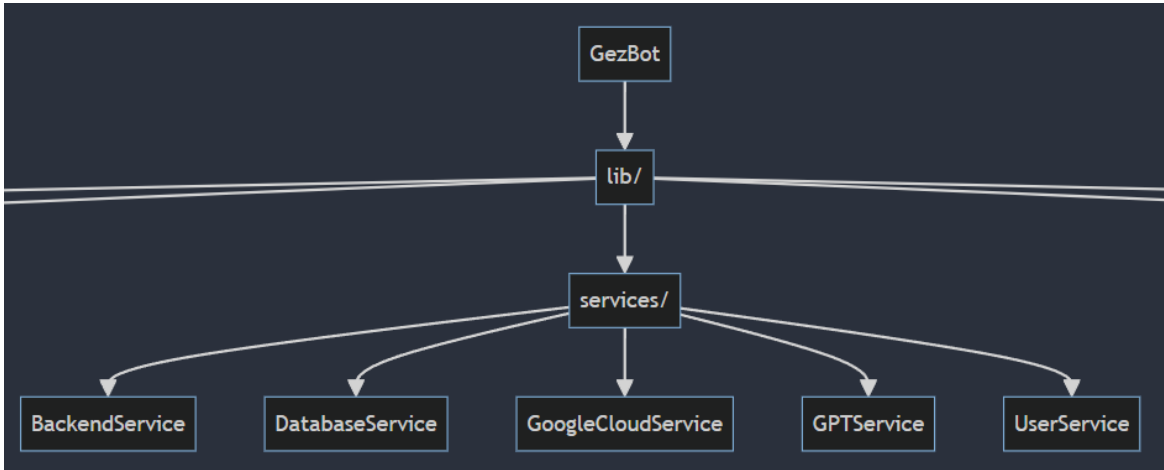
pages/: Bu dizin, uygulamanın farklı sayfalarını içeren Dart dosyalarını barındırır. Her bir sayfa, uygulamanın bir ekranını temsil eder ve genellikle bir veya birden fazla widget içerir.

- home/: Uygulamanın ana sayfasını tanımlar.
- login_screen/: Kullanıcıların giriş yapabileceği ekranı içerir.
- profile/: Kullanıcı profil bilgilerinin gösterildiği sayfa.
- search/: Seyahat veya kullanıcı aramalarının yapıldığı sayfa.
- travel/: Seyahat planlarının oluşturulduğu ve görüntülendiği sayfa.
- chat/: Kullanıcıların mesajlaşabildiği sohbet ekranı.



components/: Yeniden kullanılabilir kullanıcı arayüzü parçalarını içerir. Bu bileşenler, tutarlı bir kullanıcı deneyimi oluşturmak için uygulamanın çeşitli sayfalarında kullanılır.

- **DetailedPlaceWidget.dart:** Bir yer veya mekân hakkında ayrıntılı bilgileri gösteren bir widget. Kullanıcılara bir yer hakkında daha fazla içerik ve kontekst sağlar.
- **FlightWidget.dart:** Uçuş bilgilerini gösteren bir widget. Uçuş zamanları, fiyatları ve diğer uçuş detayları gibi bilgileri kullanıcıya sunar.
- **HotelWidget.dart:** Otel bilgilerini listeleyen bir widget. Otelin adı, yıldız derecesi, konum gibi bilgileri ve müsaitlik durumunu gösterir.
- **MapWidget.dart:** Harita üzerinde konumları gösteren bir widget. Kullanıcılara mekânların coğrafi yerini harita üzerinde gösterir.
- **PlaceWidget.dart:** Genel yer veya mekân bilgilerini gösteren basitleştirilmiş bir widget. Özet bilgileri ve temel işlevleri sağlar.
- **Question.dart:** Kullanıcıya sorular sunan bir widget. Kullanıcı etkileşimli bir şekilde sorulara cevap verebilir.
- **UserInfoCard.dart:** Kullanıcının profil bilgilerini gösteren bir kart widget'ı. Kullanıcı adı, fotoğraf gibi kişisel bilgilerin görüntülenmesine olanak tanır.



BackendService sınıfı: gezBot uygulamasının arka uç iletişimini yönetmek için Flutter'da yazılmış bir servistir. Bu servis, uçuşları ve otelleri bulmak ve önerilerde bulunmak için farklı API çağrıları yapar.

- **Sınıf İnşa Edici (Constructor)**
 - **BackendService:** Bu metod, bir Travel nesnesi alır ve GPT API anahtarını ve arka uç URL'sini çevre değişkenlerinden (dotenv) yükler.

- **getFlights:** Bu asenkron fonksiyon, kullanıcının seyahat etmek istediği başlangıç ve varış noktalarını, gidiş ve dönüş tarihlerini parametre olarak alır. Bu bilgilerle birlikte bir HTTP POST isteği yaparak uçuş bilgilerini arka uçtan alır. HTTP isteği başarılı olduğunda (HTTP 200 durum kodu aldığında), yanıtın gövdesini JSON formatında çözümleyip bir liste olarak döndürür. Eğer istek başarısız olursa bir hata fırlatılır.
- **findHotels:** Bu asenkron fonksiyon, GPT hizmetini kullanarak otel önerileri alır. GPT servisi, travel nesnesindeki verileri kullanarak önerilerde bulunur. Alınan JSON önerilerinde gerekli tüm alanlar doluysa (place, checkin, checkout, stars, hotel_types, hotel_options, adults, children), bir HTTP POST isteği yaparak bu bilgilerle otel arar. HTTP isteği başarılı olduğunda true döndürür, aksi takdirde hata fırlatılır.
- **findPlaces:** Bu metod, seyahatle ilgili yerler bulmak için arka uca HTTP POST isteği yapar. Bu istek TravelID'yi göndererek yapılır ve başarılı olduğunda true, başarısız olduğunda ise hata fırlatılır.

DatabaseService sınıfı: gezBot uygulamasının Firestore veritabanı ile etkileşimlerini yönetir. Bu servis, kullanıcı bilgileri, seyahat planları, otel ve yer önerileri gibi verilerin saklanması, sorgulanması ve güncellenmesi işlemlerini gerçekleştirir. Servis ayrıca, Google Cloud ve GPT servisleri ile entegrasyonu da içerir, böylece konum bilgileri dönüştürülebilir ve kullanıcılarla interaktif sohbetler gerçekleştirilebilir. Özetle, DatabaseService sınıfı gezBot uygulamasının veri yönetim omurgasını oluşturur ve uygulamanın dinamik ve etkileşimli özelliklerinin çoğunu destekleyen temel işlemleri sağlar.

- **Kullanıcı İşlemleri**
 - **getAllUsers:** Tüm kullanıcıları getirir.
 - **getUser:** Belirli bir kullanıcının bilgilerini getirir.
 - **setUserOptions, getUserOptions:** Kullanıcıların seçeneklerini (örneğin, seyahat tercihleri) saklar ve sorgular.
- **Seyahat Planı İşlemleri**
 - **createTravel:** Yeni bir seyahat planı oluşturur.
 - **getTravelOfUser:** Kullanıcının belirli bir seyahat planının detaylarını getirir.
 - **updateTravel:** Seyahat planını günceller.
 - **completeTravel:** Seyahat planını tamamlanmış olarak işaretler, otel ve yer bulma işlemlerini tetikler.
- **Chat İşlemleri**
 - **getUserAllChats, getChat:** Kullanıcının tüm sohbetlerini veya belirli bir sohbeti getirir.
 - **sendMessage:** Sohbeti yeni bir mesaj ekler.
- **Arkadaşlık İşlemleri**
 - **getFollowingsOfUser, getFollowersOfUser:** Kullanıcının takip ettikleri veya tarafından takip edilenleri listeler.
 - **sendFriendRequest, cancelFriendRequest, declineFriendRequest,**

acceptFriendRequest: Arkadaşlık istekleri gönderir, iptal eder, reddeder veya kabul eder.

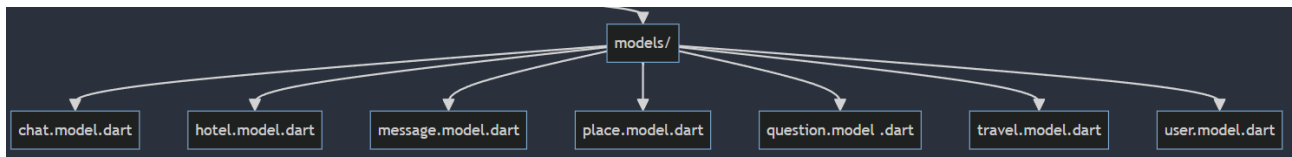
- Otel ve Yer Önerisi İşlemleri
 - getAllHotels, getFirstHotel, getRecommendedHotels, getDifferentHotels: Seyahat planına bağlı olarak otelleri sorgular ve önerir.
 - getAllPlaces, getRecommendedPlaces, getDifferentPlaces: Seyahat planına bağlı olarak yerleri sorgular ve önerir.
- Arama Fonksiyonları
 - searchByUserNameAndEmail, searchTravelsByUserNameAndEmail, searchTravelsByDestination: Kullanıcıları ve seyahat planlarını çeşitli kriterlere göre arar.

GoogleCloudService sınıfı: Google Cloud Platform'un çeşitli API'ları ile etkileşim kurmak için kullanılır. Bu servis, yer bilgileri, yer aramaları, rota bilgileri ve daha fazlası gibi çeşitli işlevleri yerine getirebilmektedir.

- Sınıf İnşa Edici (Constructor)
 - _apiKey: API anahtarı, çevre değişkenlerinden (dotenv) yüklenir. Bu anahtar, Google Cloud API isteklerini yetkilendirmek için kullanılır.
- Metodlar
 - _getFunction: GET istekleri için genel bir yardımcı metod. URL ve parametreler alır ve Google Cloud API'sine istek gönderir. İsteğe bağlı olarak, sonuçları sınırlamak için limit parametresi kullanılabilir.
 - _postFunction: POST istekleri için genel bir yardımcı metod. Belirli bir URL'ye ve verilen JSON gövdesine göre bir POST isteği yapar.
 - coordinatesToAddress: Belirli bir enlem ve boylam koordinat seti için adres bilgilerini döndürür. İsteğe bağlı olarak detaylı adres bilgileri döndürülebilir veya sadece şehir ve ülke gibi temel bilgiler döndürülebilir.
 - fetchPlacesNearby: Belirli bir konumun çevresindeki yerleri arar. Arama, belirli türlerdeki yerlerle sınırlanabilir ve belirli bir yarıçap içindeki yerler döndürülür.
 - fetchPlacesQuery: Bir sorgu metni kullanarak yerleri arar. İsteğe bağlı olarak, aramayı belirli türlerdeki yerlerle sınırlamak için types parametresi kullanılabilir.
 - fetchPlacesAllMethods: Hem metin sorgusu hem de konum bilgisine dayalı olarak yerleri arar. Bu metod, çeşitli arama kriterlerini birleştirerek daha kapsamlı sonuçlar elde etmek için kullanılır.
 - fetchPlaceDetails: Belirli bir yerin detaylı bilgilerini döndürür. Yer detayları, yerin kimliği (place_id) kullanılarak sorgulanır.
 - fetchCarRoute: İki nokta arasındaki rota bilgilerini döndürür. Rota, sürüş, yürüyüş vb. gibi belirli bir modda hesaplanabilir.

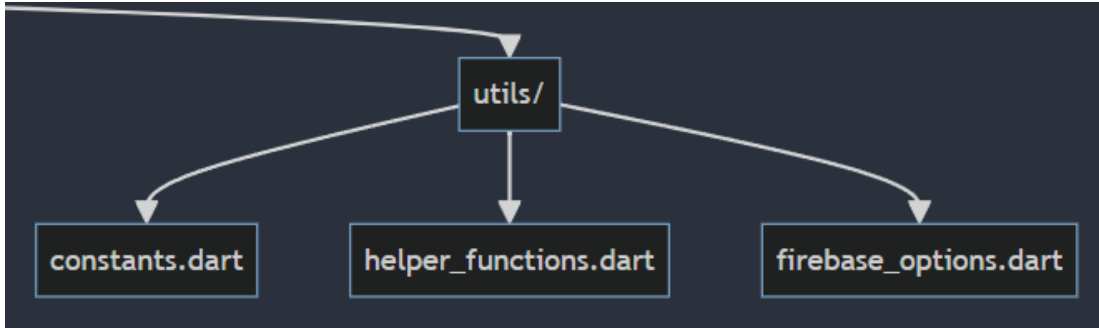
UserService sınıfı: kullanıcı işlemlerini yönetmek için Firebase Authentication, Cloud Firestore ve Firebase Storage servisleri ile etkileşime geçer. Bu servis, kullanıcı kaydı, oturum açma, profil güncellemeleri ve arkadaşlık durumu kontrolü gibi temel kullanıcı yönetimi işlevlerini sağlar.

- Kullanıcı Kaydı ve Oturum İşlemleri
 - registerUser: Email ve şifre kullanarak yeni bir kullanıcı kaydeder. Kullanıcı adı benzersiz olmalıdır. Başarılı kayıt sonrası kullanıcı bilgileri Firestore'a kaydedilir ve oturum açılır.
 - signInWithEmailAndPassword: Email ve şifre ile oturum açar.
 - signUpWithGoogle, signInWithGoogle: Google hesabı ile oturum açma veya kayıt olma işlemi yapar. İlk kez giriş yapan kullanıcılar için Firestore'a yeni bir kayıt oluşturulur.
- Kullanıcı Profili Güncellemeleri
 - updateUsername: Kullanıcının adını günceller.
 - updateUserProfilePhoto: Kullanıcı profil fotoğrafını günceller. Fotoğraf Firebase Storage'a yüklenir ve URL Firestore'a kaydedilir.
 - updateUserBirthDate: Kullanıcının doğum tarihini günceller.
 - updateGender: Kullanıcının cinsiyet bilgisini günceller.
- Yardımcı Metodlar
 - fetchAndStoreUserDetails: Belirli bir kullanıcının detaylarını Firestore'dan çeker ve SharedPreferences ile yerel depolamaya kaydeder.
 - fetchUserDetails: Belirli bir kullanıcının detaylarını döndürür.
 - checkRelationshipStatus: İki kullanıcı arasındaki ilişki durumunu kontrol eder (örn. arkadaşlar, arkadaşlık isteği gönderildi, vb.).
- Önemli Noktalar
 - Kullanıcı kaydı ve oturum açma işlemleri sırasında hata yönetimi için showErrorDialog fonksiyonu kullanılır.
 - Google oturum açma işlemi için GoogleSignIn paketi kullanılır.
 - Kullanıcı bilgilerinin güncellenmesi ve saklanması için FirebaseFirestore ve FirebaseStorage servisleri etkin şekilde kullanılır.
 - Uygulama içi navigasyon ve kullanıcı durumu yönetimi için SharedPreferences kullanılır.



Model dosyaları: gezBot uygulamasının veri yapılarını temsil eder ve her biri uygulamanın farklı veri türlerini saklamak için kullanılır.

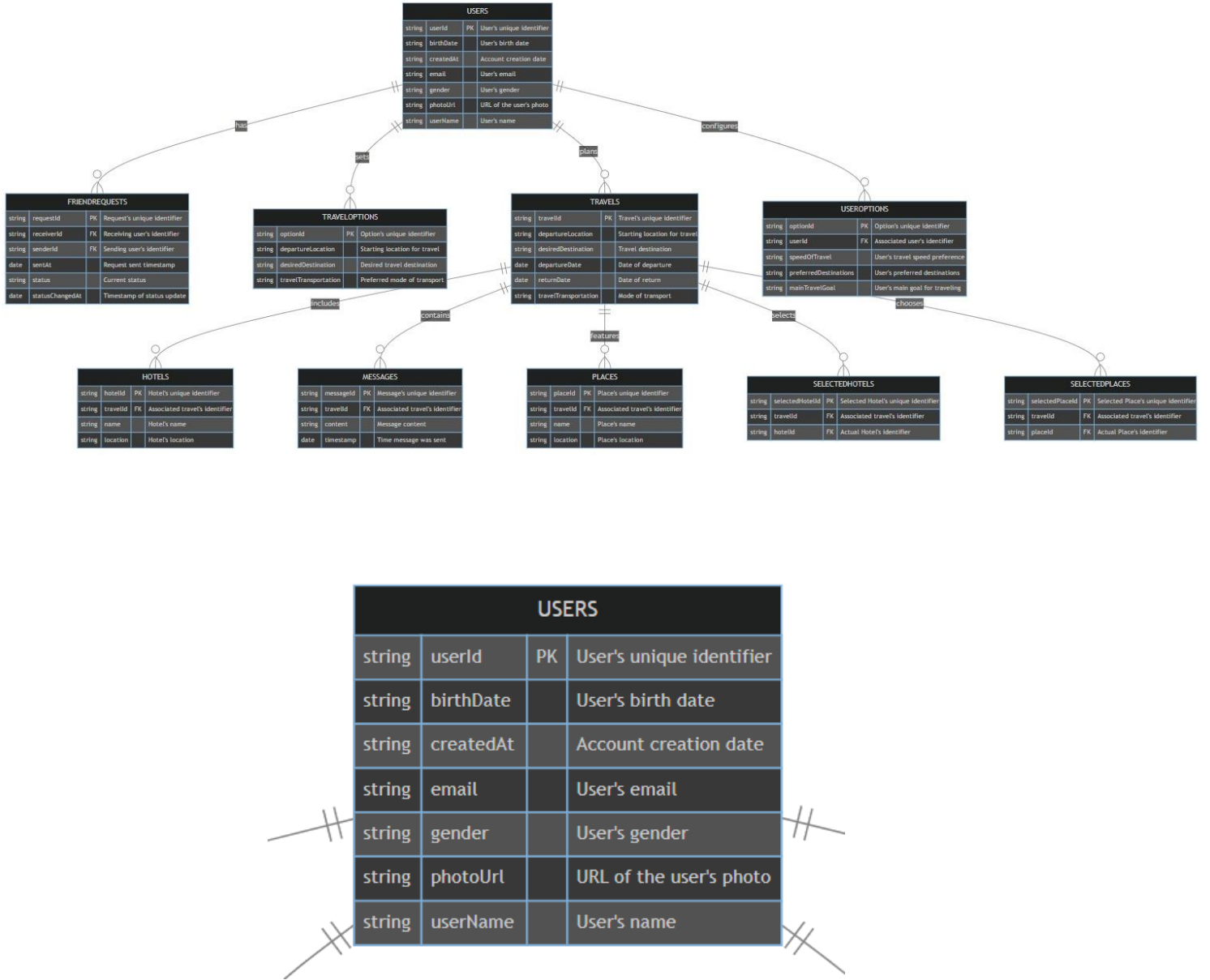
- chat.model.dart: Sohbetlerin veri yapısını tanımlar. Kullanıcılar arasında mesajlaşma işlevi için gerekli bilgileri, mesajların içeriği, gönderenin bilgisi gibi sohbet ile ilgili detayları içerir.
- hotel.model.dart: Otel verilerini saklar. Otelin adı, yıldız derecesi, fiyatı, konumu gibi otel ile ilgili özellikleri içeren veri yapısını tanımlar.
- message.model.dart: Mesajların veri yapısını tanımlar. Bir sohbet içindeki bireysel mesajlar için gönderenin kimliği, mesaj metni, zaman damgası gibi bilgileri saklar.
- place.model.dart: Mekân veya yerlerin veri yapısını tanımlar. Konum, açıklama, popülerlik derecesi gibi mekân hakkında bilgileri içerir.
- question.model.dart: Kullanıcılar tarafından yanıtlanacak soruların veri yapısını tanımlar. Soru metni ve olası cevapları içerir.
- travel.model.dart: Seyahatlerin veri yapısını tanımlar. Seyahatin adı, açıklaması, oluşturulma tarihi gibi seyahatle ilgili ayrıntıları saklar.
- user.model.dart: Kullanıcı bilgilerinin veri yapısını tanımlar. Kullanıcının adı, e-posta adresi, profil fotoğrafı gibi kullanıcı hakkındaki temel bilgileri içerir.



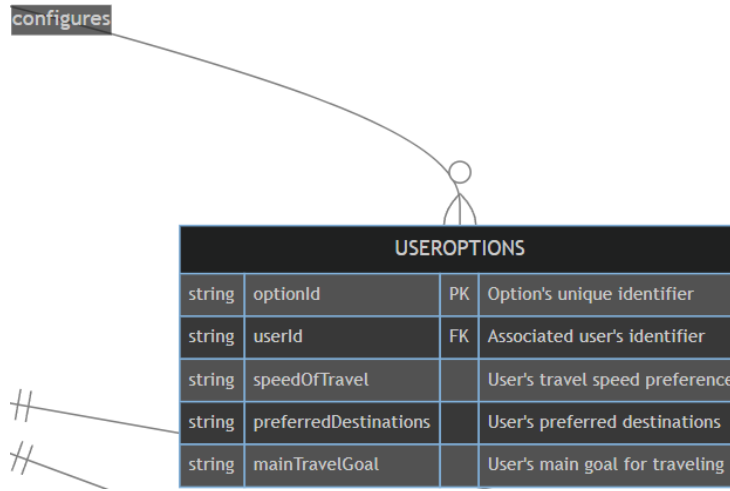
utils klasörü: gezBot uygulamasının yardımcı fonksiyonlarını ve yapılandırma dosyalarını içerir.

- constants.dart: Uygulama genelinde kullanılan sabit değerleri tanımlar. Örneğin, tasarım renkleri, metin stilleri veya uygulama içi sabit metinler gibi uygulamanın farklı yerlerinde tekrar tekrar kullanılması gereken değerleri içerir.
- helper_functions.dart: Uygulamanın çeşitli yerlerinde tekrar eden görevleri yerine getiren yardımcı fonksiyonları içerir. Bu, tarih formatlama, veri dönüşümü veya belirli hesaplamalar gibi genel amaçlı işlevleri içerir.
- firebase_options.dart: Firebase başlatma için gerekli olan yapılandırma ve seçeneklerin tanımlandığı dosyadır. Bu dosya, Firebase projesinin platform özel yapılandırmalarını içerir.

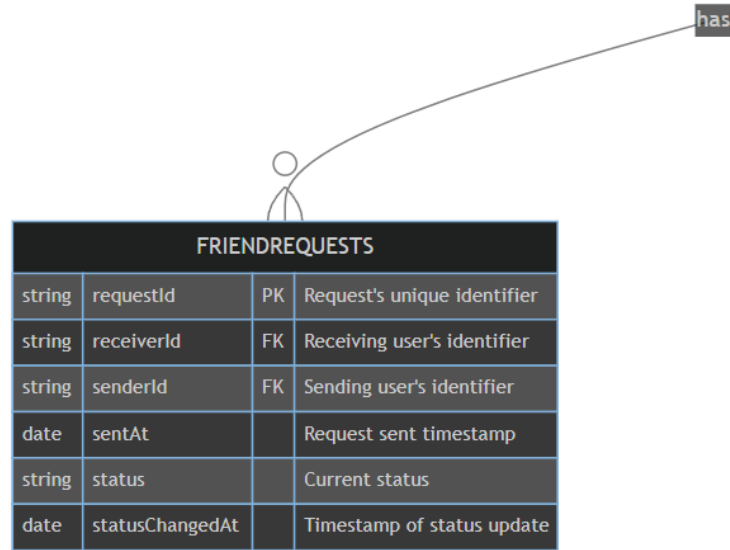
7.2 Firebase Database Yapısı



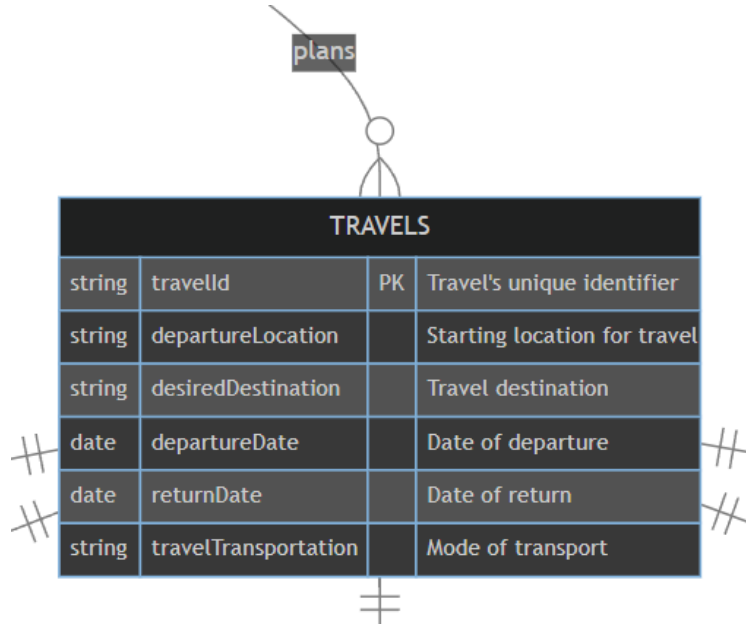
- **USERS:** Kullanıcıların kişisel bilgilerinin saklandığı modeldir. Her kullanıcı için benzersiz bir kimlik (`userId`), doğum tarihi (`birthDate`), hesap oluşturulma tarihi (`createDate`), e-posta adresi (`email`), cinsiyet (`gender`), profil fotoğrafının URL'si (`photoUrl`) ve kullanıcı adı (`userName`) saklanır.



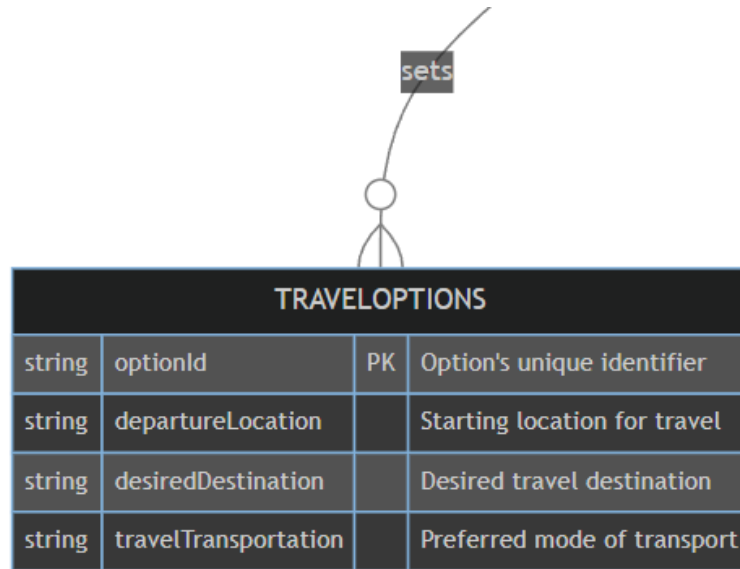
- **USEROPTIONS:** Kullanıcıların seyahat tercihlerini ifade eden seçeneklerin saklandığı modeldir. Seyahat hız tercihi (speedOfTravel), tercih edilen destinasyonlar (preferredDestinations), ve ana seyahat amacı (mainTravelGoal) gibi kullanıcı bazında tercihler yer alır.



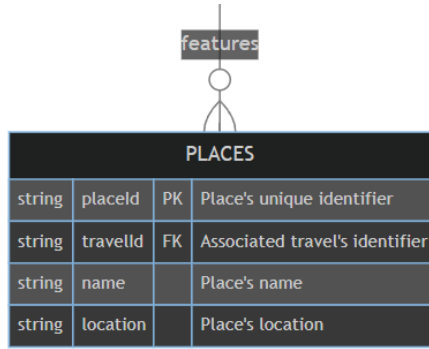
- **FRIENDREQUESTS:** Kullanıcılar arası arkadaşlık isteklerinin takip edildiği modeldir. Her istek için benzersiz bir kimlik (requestId), gönderen kullanıcının kimliği (senderId), alan kullanıcının kimliği (receiverId), istek gönderme zamanı (sentAt), güncel durum (status) ve durum güncellenme zamanı (statusChangedAt) bilgilerini içerir.



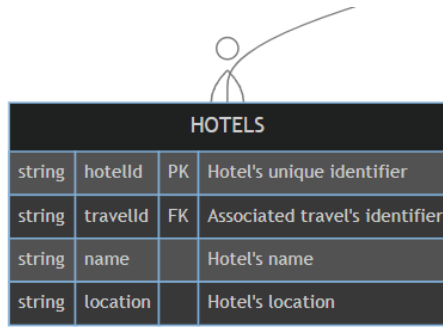
- **TRAVELS:** Kullanıcıların oluşturduğu seyahat planlarının saklandığı modeldir. Her seyahat için benzersiz bir kimlik (travelID), başlangıç yeri (departureLocation), varış yeri (desiredDestination), gidiş tarihi (departureDate), dönüş tarihi (returnDate) ve ulaşım türü (travelTransportation) gibi detaylar yer alır.



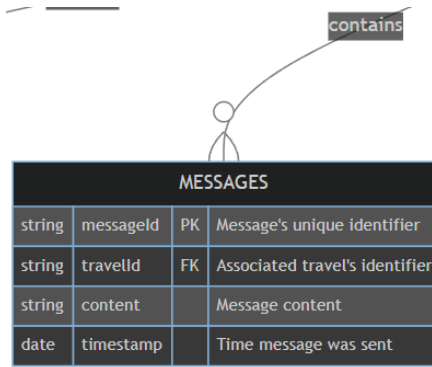
- **TRAVELOPTIONS:** Seyahatler sırasında kullanıcının tercih ettiği çeşitli seçeneklerin saklandığı modeldir. Her seçenek için benzersiz bir kimlik (optionId), tercih edilen varış yeri (desiredDestination), ve tercih edilen ulaşım modu (travelTransportation) gibi bilgileri içerir.



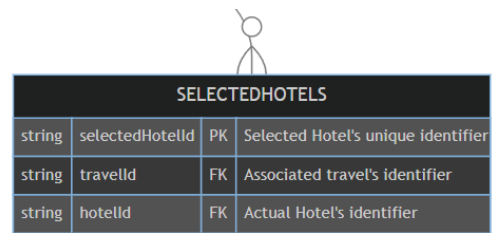
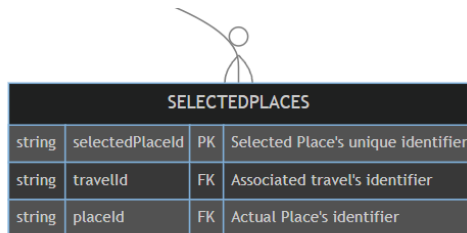
- **PLACES:** Seyahat sırasında ziyaret edilen veya edilecek yerlerin bilgilerinin saklandığı modeldir. Her yer için benzersiz bir kimlik (placeId), ve konum (location) bilgisi bulunur.



- **HOTELS:** Seyahat planlarına dahil edilen otellerin detaylarının saklandığı modeldir. Her otel için benzersiz bir kimlik (hotelId), otelin adı (name), ve konumu (location) gibi bilgiler tutulur.



- **MESSAGES:** Seyahat sırasında kullanıcılar arasında gönderilen mesajların detaylarının saklandığı modeldir. Her mesaj için benzersiz bir kimlik (messageId), ilişkili seyahatin kimliği (travelId), mesajın içeriği (content) ve zaman damgası (timestamp) yer alır.



- **SELECTEDHOTELS ve SELECTEDPLACES:** Kullanıcıların seyahatleri için seçtikleri oteller ve yerlerin saklandığı modellerdir. Seçilmiş her otel veya yer için seçilmiş otelin (selectedPlaceId) veya yerin benzersiz kimliği, ilişkili seyahatin kimliği (travelId) ve gerçek otelin (hotelId) veya yerin kimliği (placeId) bulunur.

7.3 Gezbot FastAPI Backend Yapısı

GET	/	Read Root	▼
POST	/find_places	Find Places	▼
POST	/query	Query	▼
GET	/travel_details	Travel Details	▼
GET	/travel_exist	Travel Exist	▼
POST	/flights	Flights	▼
POST	/find_hotels	Find Hotels	▼
POST	/create_car_route	Create Car Route	▼

Bu Python betiği, FastAPI kullanarak RESTful API'ler için bir backend oluşturur. Asenkron bir yapıya sahip olduğu için, birden çok isteği eş zamanlı olarak idare edebilir.

- **Ana özellikleri**
 - Firebase ve Firestore ile entegrasyon sağlar, veritabanı işlemleri için Firebase Admin SDK'sını kullanır.
 - GoogleApi, gpt_api ve Hotel_Api sınıflarını kullanarak dış API servislerini yönetir.
 - CORS (Cross-Origin Resource Sharing) ayarlarını yapılandırır, böylece API'nin farklı kaynaklardan güvenli bir şekilde çağrılmasını sağlar.
 - Uygulama uvicorn ile sunucu olarak çalışır, bu sayede main:app kullanılarak yeniden yüklemeler ile otomatik olarak yeniden başlatılabilir.
- **API Endpoints**
 - /: API'nin temel URL'sine bir GET isteği geldiğinde, servisin canlı olduğunu doğrular.
 - /find_places: Bir TravelModel nesnesi olarak, belirtilen destinasyon için ilgili yerleri bulur ve Firestore veritabanına kaydeder.
 - /query: Belirli bir sorguyu Google yerler aramasına gönderir ve sonuçları döndürür.
 - /travel_details: Belirtilen travelID'ye göre seyahat ayrıntılarını döndürür.
 - /travel_exist: Verilen travelID'nin var olup olmadığını kontrol eder.
 - /flights: Uçuş bilgilerini getirmek için üçüncü parti bir servis kullanır.
 - /find_hotels: HotelModel verisine göre otel bulur ve Firestore'a kaydeder.
 - /create_car_route: Belirtilen koordinatlar arası bir araç rotası oluşturur.
 - s
- **GPT API sınıfı:** OpenAI'nin GPT-3.5 modelini kullanarak özelleştirilmiş seyahat planları önerileri üretmek için tasarlanmış bir Python sınıfıdır. Bu sınıf, kullanıcıların seyahat ayrıntılarını ve tercihlerini temsil eden yapılandırılmış JSON girdisini işler ve belirtilen seyahat süresi boyunca ziyaret edilecek yerler ve yapılacak aktivitelerin kapsamlı bir listesini oluşturmak için tasarlanmıştır.

- `__call__gpt_api`: GPT-3.5 modeline HTTP POST isteği gönderen ve yanıt olarak önerileri alan özel bir metod.
- `Get_google_search_tags`: Kullanıcının seyahat verilerini ve kişisel tercihlerini alır, bu verileri GPT-3 modeline gönderir ve modelin ürettiği önerileri JSON olarak döndürür.
- **GoogleApi sınıfı**: Google Haritalar API'sini kullanarak çeşitli lokasyonlara ait bilgileri almak, yer detaylarını sorgulamak, adresleri koordinatlara çevirmek ve rota hesaplamaları yapmak için tasarlanmış bir Python sınıfıdır. Bu sınıfın metodları asenkron olarak tasarlanmıştır, bu da onları aynı anda birden fazla iş parçacığı veya proses tarafından çalıştırılabileceği anlamına gelir.
 - `__init__`: Kurucu fonksiyon, `config.json` dosyasından API anahtarını yükler ve sınıf için bir `API_KEY` özelliği oluşturur.
 - `get_function`: GET istekleri yaparak Google Maps API'den veri alır ve bu veriyi döndürür.
 - `post_function`: POST istekleri yaparak Google Maps API'ye veri gönderir ve yanıtı işler.
 - `save_as_json`: Aldığı veriyi JSON formatında dosyaya yazar.
 - `coordinates_to_address`: Verilen enlem ve boylam koordinatları için adres bilgisi getirir.
 - `__fetch_places_nearby`: Verilen lokasyon ve türler bazında yakındaki yerleri bulur.
 - `fetch_places_query`: Verilen sorgu cümlesiyle yerleri arar.
 - `fetch_places_all_methods`: Hem sorgu cümlesi hem de konum bilgisi kullanarak yerleri arar ve çakışmayan yerlerin birleşimini döndürür.
 - `fetch_place_details`: Belirli bir yerin detaylarını getirir.
 - `fetch_car_route`: İki nokta arasında rota hesaplaması yapar ve çeşitli parametreleri (seyahat modu, trafik modeli vb.) destekler.

default	^
POST /process_reviews/ Process Reviews	v
POST /find_similar_places_from_local/ Find Similar Places From Local	v
POST /process_and_find_similar_places/ Process And Find Similar Places	v
GET / Read Root	v

- **AI Backend**: Temelde, kullanıcı yorumları ve tercihleri üzerinden lokasyonlara ait semantik anlam taşıyan gömülü (embedded) vektörler oluşturur ve bu vektörleri kullanarak kullanıcıların tercihlerine en uygun lokasyonları bulur.
 - BertTokenizer ve BertModel kullanarak, BERT modelini "bert-base-multilingual-cased" versiyonu ile yükler ve tokenleştirme (metni tokenlere ayırma) ve modeli kullanarak gömülü vektörler (embeddings) üretir.
 - `compute_embedding` fonksiyonu, verilen bir metin için gömülü vektör hesaplar. Modelden çıkan çıktının ortalamasını alarak tek bir vektör haline getirir.
 - `/process_reviews/ endpoint'i`, gönderilen yorumların her birini işler ve ağırlıklı gömülü vektörler oluşturur. Bu vektörler, lokasyonların semantik anlamını temsil eder.
 - `/find_similar_places_from_local/ endpoint'i`, yerel veri setinden lokasyonların gömülü vektörlerini hesaplar ve kullanıcının tercihlerine göre en benzer lokasyonları bulur.
 - `find_similar_places` fonksiyonu, kullanıcının tercihlerini temsil eden özel bir gömülü vektör ve restoran vektörlerini kullanarak en benzer lokasyonları bulur. Benzerlik skoru, kosinüs benzerliği kullanılarak hesaplanır.
 - `/process_and_find_similar_places/ endpoint'i`, verilen yorumları işler ve kullanıcının tercihiye göre en benzer lokasyonları döndürür.

8. TAKIM İÇERİSİNDEKİ İŞ PAYLAŞIMI

• Harun Serkan Metin

Takım içi iş paylaşımı bağlamında, projede bir dizi farklı rol üstlendim ve her biri için kapsamlı çalışmalar yürüttüm. Öncelikle, Flutter ve Firebase entegrasyonunu gerçekleştirdim ve uygulamamız için hayati öneme sahip olan Database Service'i oluşturdum. Bu servis, uygulama içindeki tüm CRUD operasyonlarının merkezi yönetimini sağladı ve veritabanı işlemlerimizi etkili bir şekilde düzenledi. Ayrıca, backend tarafında RESTful API'ler için FastAPI kullanarak bir backend geliştirdim. Asenkron yapısı sayesinde, backendimiz birden fazla isteği eş zamanlı olarak işleyebildi, bu da uygulamamızın performansını artırdı.

Projede ayrıca, otel verilerini almak için Google API'larına güvenemediğimiz için kendi Detaylı Web Scraping aracımızı geliştirdik. Bu araç, otellerin fiyatları, rezervasyon bilgileri, olanakları ve daha fazlasını topladı, böylece kullanıcılarımız için güncel ve detaylı bilgiler sağladık. Google Places API'yi sıkça kullanarak, kullanıcıların kişiselleştirilmiş mekân sorgularını işlemek için HTTP isteklerini etkin bir şekilde yönettim. Bu sorguları geliştirmek için GPT APIs'ine sorgu atarak, kişisel tercihlere göre arama filtrelerini hem otel araması için hem de gezilecek mekanların aranması için optimize ettim. Bu sayede, kullanıcılarımızın daha özelleştirilmiş ve ilgi alanlarına uygun sonuçlar elde etmelerini sağladık.

Ayrıca, Google Maps API'nin farklı arama yöntemlerini birleştirerek, kullanıcılarımızın daha geniş bir sonuç yelpazesine erişimini sağladık ve kullanıcı deneyimini iyileştirdik.

Son olarak, Flutter kısmında Chat Page'i ve backend yapısını AI Chat bot entegrasyonu ile birlikte geliştirdim. Bu, kullanıcıların uygulama içinde etkileşimde bulunabilecekleri dinamik bir iletişim aracı sağladı. Ayrıca, UI ile ilgili büyük çaplı hataları tespit edip çözmekte diğer takım üyelerime destek oldum. Bu süreçte, takımın her aşamasında etkin bir rol oynadım ve proje başarısında önemli bir katkı sağladım. Bu şekilde, iş paylaşımı ve iş birliği sayesinde, projenin başarılı bir şekilde tamamlanmasını sağladık.

• Hüseyin Pekkan

Başlangıçta, kullanıcıların uygulamaya kolayca giriş yapabilmesi ve **kaydolabilmesi** için Flutter tabanlı arayüzler tasarladım. Bu işlem sırasında, geliştirdiğim sayfaları Firebase'in kimlik doğrulama sistemine entegre ettim, aynı zamanda kullanıcıların profillerini yönetebilecekleri, arkadaşlık istekleri gönderebilecekleri ve alabilecekleri bir profil sayfası oluşturdum.

Profil sayfasını ve ilgili işlevselliği geliştirirken, arkadaşlık isteklerini yönetme, profil bilgilerini düzenleme ve kullanıcı etkileşimlerini güçlendiren özellikleri uygulamaya döküm. Bu süreçte, veritabanı servisinde ekip arkadaşım Harun tarafından geliştirilen bazı metodları yeniden düzenledim ve ihtiyaç duyulan alanlarda eklemeler yaptım. User Service sınıfını, projenin gereksinimleri doğrultusunda geliştirdim.

Travel oluşturma sürecini basitleştirmek için dinamik bir form tasarladım. Kullanıcılar bu form üzerinden bilgilerini girerken verilerini veritabanına aktardım ve kullanıcıların daha sonra formu nerede bıraktıysa oradan devam edebilmelerini sağladım. Formda yer alan soruların Firestore'dan dinamik olarak çekilmesi özelliğini de başarıyla entegre ettim. Böylece, kullanıcıların seyahat planlama deneyimlerinin veritabanıyla sürekli senkronize olmasını sağladım.

Google Places API entegrasyonu ile, kullanıcıların uygulama üzerinden harita aracılığıyla konumlarını belirlemelerine olanak tanıdım. Bu kapsamda, seçilen başlangıç ve varış noktalarına bağlı olarak en kısa rota bilgisinin kullanıcıya sunulmasını sağlayan bir Map Widget geliştirdim. Bu widget, kullanıcılara önerilen yerler arasındaki en kısa rotayı göstermek için ekip arkadaşım Harun'un geliştirdiği algoritmayı da içeriyordu.

Yapay zeka entegrasyonu konusunda, kullanıcıların geçmiş tercihleri ve yerel mekan yorumlarına dayanarak kişiselleştirilmiş öneriler sunan bir sistem kurdum. Google Maps'te belirlenen ilgi çekici noktaları, kullanıcı tercihleri ve BERT dil modeli ile analiz ederek kullanıcıya sunulan öneriler, hem kullanıcının geçmiş tercihlerine hem de mekanların yorumlarına dayalı olarak hazırlandı. Bu entegrasyonu ve kullanıcı etkileşimleri için gerekli backend altyapısını FastAPI kullanarak ben geliştirdim.

Son olarak, kullanıcıların uçak bileti arama sürecini optimize etmek için, belirli koşullar altında en uygun uçuş seçeneklerini bulan bir web scraping kütüphanesi yazdım. Bu kütüphaneyi, projenin backend sunucusuyla tam olarak entegre etmek için çalışmalar yürüttüm.

- **Arda Erol**

Flutter UI geliştirmesi yaparak, uygulamamızın kullanıcı dostu ve etkileyici bir görünüme sahip olmasını sağladım. Kullanıcı arayüzü tasarımında estetik ve işlevsellik arasında denge kurarak, kullanıcıların uygulamayı kolayca kullanmasını sağladım.

UI/UX tasarımı konusunda dikkatli bir çalışma yürüterek, uygulamanın her detayını titizlikle ele aldım. Kullanıcı arayüzünde yer alan butonlar, formlar, sayfa geçişleri ve diğer bileşenlerin her birini özenle tasarladım ve kullanıcı deneyimini en üst düzeye çıkarmak için gereken düzenlemeleri yaptım. Chat bölümünün tasarımını oluştururken, kullanıcıların etkileşimli ve sorunsuz bir iletişim deneyimi yaşamalarını sağlamaya çalıştım. Ayrıca, backend tarafında, kullanıcı mesajlarını işlemek, depolamak ve iletmek için güvenilir ve verimli bir altyapı oluşturmak için çalıştım. Bu süreçte, kullanıcıların anlık iletişim kurmasını ve uygulamanın performansını artırmak için asenkron yapıları kullanarak verimli bir çözüm geliştirdim. Bu şekilde, hem kullanıcı deneyimini hem de uygulamanın teknik performansını en üst düzeye çıkarmak için çaba gösterdim.

Ayrıca, kullanıcıların seyahat tercihlerini belirlemelerine olanak tanıyan bir form oluşturmak için prompt engineering tekniklerini kullanarak, ChatGPT'nin anlayabileceği şekilde promptları optimize ettim. Ana sayfa ve profil sayfası için kullanıcı arayüzü kodlarını Flutter üzerinde yazarak, Firebase veritabanı servisleriyle etkileşim sağladım ve kullanıcıların seyahatleri ve profilleriyle ilgili bilgileri görüntüleyebilmelerini sağladım.

- **Rana Demir**

Uygulama giriş ve ana sayfadaki arka plan tasarımı, Figma uygulaması üstünde geliştirildi ve uygulamada gösterilmek üzere Flutter projesine aktarıldı. Bu tasarımın hareketlerini gösterecek şekilde dinamikler ayarlandı.

Kullanıcının yeni seyahat oluştururken yaptığı seçimler (nereden nereye, ne zaman, ne amaçla, başka bir aktivite arzusu olarak yoksa olmadan mı gideceği üstüne yapılanlar) ve en sonda bir prompt şeklinde diğer özel isteklerini belirtebildiği noktaya kadar olan tüm unsurların, ChatGPT'nin anlayacağı şekilde bir prompt'a dönüştürülmesi gerekti. Onun algılayabileceği şekilde prompt'un uygun hâle getirilmesi adına prompt engineering yapıldı. Bu görevi yerine getirmek adına kalıp şeklinde bir prompt metni oluşturuldu.

Ana sayfanın frontend kodları Flutter üzerinde yazıldı. Bu süreçte Firebase veri tabanına erişim için hazırlanmış olan servis kodları aracılığıyla veri tabanı kontrol edildi ve kullanıcı o ana kadar hiç seyahat oluşturmamışsa, oluşturmuş fakat belirli bir sebepten devam edememişse bunların ana sayfada gösterimi sağlandı. Bu gösterimler üzerinden seyahat sayfasına erişim sağlandı. Oluşturulmuş seyahatler var ise de bunların her biri arasında gezinerek bunlar hakkında bilgi alınabilecek 'Get details' butonuyla beraber bir alan oluşturuldu. Bu buton aracılığıyla da yine seyahat sayfasına erişim sağlandı.

Profil sayfasının frontend kodları Flutter üzerinde yazıldı. Yine Firebase veri tabanına ulaşmayı sağlayan servis kodları aracılığıyla kullanıcı bilgilerine erişim sağlandı ve kullanıcının sahip olduğu seyahat, takipçi ve takip edilen sayısı; aynı şekilde profil fotoğrafı, kullanıcı adı, mail adresi, yaş ve cinsiyet bilgisi gibi tüm bilgiler ekrana yansıtıldı. Gerektiğinde kullanıcı adı, yaş ve cinsiyet bilgilerini düzenlemek için bir profil düzenleme butonu; buna bağlı olarak da açılan pencere fonksiyonları ayarlandı.

9. REFERANSLAR

[1] <https://developers.google.com/maps/documentation/places/web-service/overview>

[2] <https://selenium-python.readthedocs.io/>

[3] <https://www.google.com/travel/hotels/>

[4] <https://www.google.com/travel/flights/>

10. FİNAL DEMO VIDEO LİNKİ

<https://youtu.be/1i7UiZpUXQw>