T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 498 BİTİRME ÇALIŞMASI

YAPAY ZEKA DESTEKLİ SEYAHAT REHBERİ MOBİL UYGULAMASI

G191210020 – Serdar ARICI

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ
Tez Danışmanı : Prof. Dr. Nejat YUMUŞAK

2023-2024 Bahar Dönemi

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

YAPAY ZEKA DESTEKLİ SEYAHAT REHBERİ MOBİL UYGULAMASI

BSM 498 - BİTİRME ÇALIŞMASI

Serdar ARICI

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Bu tez .. / .. / … tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

…….	…….	…….
Jüri Başkanı	**Üye**	**Üye**

ÖNSÖZ

Günümüzde yapay zeka teknolojisinin hızlı ve etkili bir şekilde gelişmesiyle, bu alanda yapılan çalışmalar ve uygulamalar da hızla artmaktadır. Yapay zeka, birçok projede etkili sonuçlar sunmakta ve yazılımcılar için yapay zeka API'leri projelerinde kullanımı kolaylaştırmaktadır. Bu bağlamda, teknolojik işbirlikleri veri hızının arttırılması, iletişim maliyetlerinin düşürülmesi, veri güvenliğinin sağlanması ve müşteri memnuniyetinin arttırılması gibi konularda önemli katkılar sağlamaktadır.

Bu çalışma, "Yapay Zeka Destekli Seyahat Rehberi Mobil Uygulaması" başlıklı bitirme çalışması tezidir. Bu çalışma, mobil uygulama geliştirme alanında bir inceleme yapmak ve seyahat eden bireylerin ihtiyaçlarına yardımcı olmak amacıyla çeşitli yöntemlerle çözümler önermek doğrultusunda gerçekleştirilmiştir.

Bu çalışmanın hazırlanması sürecinde bana yardımcı olan aileme, dostlarıma ve değerli hocalarıma teşekkürü bir borç bilirim. Ayrıca, Prof. Dr. Nejat Yumuşak tarafından sağlanan rehberlik ve destek büyük önem taşımıştır. Kendisine teşekkür borçluyum; engin bilgisi ve değerli önerileri, bu çalışmayı tamamlamamda kilit bir rol oynamıştır.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

AOSP	: Android Open Source Project (Android Açık Kaynak Projesi)
API	: Application Programming Interface (Uygulama Programlama Arayüzü)
FAB	: Floating Action Button (Yüzen İşlem Butonu)
FCM	: Firebase Cloud Messaging (Firebase Bulut Mesajlaşma)
HTTP	: Hypertext Transfer Protocol (Hiper Metin Aktarım Protokolü)
IDE	: Integrated Development Environment (Entegre Geliştirme Ortamı)
JIT	: Just-In-Time (Anında Derleme)
JSON	: JavaScript Object Notation (JavaScript Nesne Gösterimi)
LBS	: Location-Based Services (Konum Tabanlı Hizmetler)
MVC	: Model-View-Controller (Model-Görünüm-Yönetici)
MVVM	: Model-View-ViewModel (Model-Görünüm-Görünüm Modeli)
NLP	: Natural Language Processing (Doğal Dil İşleme)
NoSQL	: Not Only SQL (Sadece SQL Değil)
OpenGL	: Open Graphics Library (Açık Grafik Kütüphanesi)
REST	: Representational State Transfer (Temrepresentasyonel Durum Transferi)
SCM	: Software Configuration Management (Yazılım Yapılandırma Yönetimi)
SDK	: Software Development Kit (Yazılım Geliştirme Kiti)
SoC	: System on Chip (Çip Üzerinde Sistem)
SOAP	: Simple Object Access Protocol (Basit Nesne Erişim Protokolü)
SSL	: Secure Sockets Layer (Güvenli Soket Katmanı)
UI	: Kullanıcı Arayüzü (User Interface)
URL	: Uniform Resource Locator (Birleşik Kaynak Yer Belirleyici)
UX	: User Experience (Kullanıcı Deneyimi)
VCS	: Version Control System (Sürüm Kontrol Sistem
XML	: eXtensible Markup Language (Genişletilebilir İşaretleme Dili)

ŞEKİLLER LİSTESİ

ÖZET

Anahtar kelimeler: Yapay Zeka, Mobil Uygulama, Android, Kotlin, Firebase, API

Teknolojinin hızlı gelişimiyle birlikte seyahat etme alışkanlıklarımızda önemli değişimler yaşanmaktadır. İnsanlar, yeni yerler keşfetmek ve farklı kültürleri deneyimlemek için seyahat etmeyi her zaman sevmişlerdir. Ancak, teknolojinin ilerlemesiyle birlikte seyahat deneyimleri daha konforlu, planlı ve bilgiye erişimi daha kolay hale gelmiştir.

Bu bağlamda, "Travel Guide" adlı mobil uygulama, seyahat etmeyi kolaylaştırmak ve kullanıcıların seyahat deneyimlerini zenginleştirmek amacıyla geliştirilmiştir. Kullanıcı dostu arayüzü ve kapsamlı özellikleriyle öne çıkan bu uygulama, seyahat tutkunlarının ihtiyaçlarını karşılamak için tasarlanmıştır.

Travel Guide, kullanıcıların gezdikleri yerler hakkında deneyimlerini paylaşmalarına olanak tanır ve diğer kullanıcılara rehberlik yapar. Kullanıcılar, kendi profillerinde paylaşımlarını görebilir ve düzenleyebilirler, böylece gezdikleri yerler hakkında bir tür seyahat defteri oluşturma imkanı bulurlar.

Uygulama, kullanıcıların seyahatlerini planlamalarına yardımcı olmak için bir dizi özellik sunar. Yapay zeka destekli çeviri araçları, dil engellerini aşmaya yardımcı olurken, chatbot gibi özellikler ise kullanıcıların her an yanlarında bir rehber bulmalarını sağlar.

Travel Guide'ın harita entegrasyonu, kullanıcıların bulundukları konum hakkında anlık bilgiye erişmelerini sağlar ve yollarını kolayca bulmalarına yardımcı olur. Ayrıca, ülkeler hakkında genel bilgilerin ve seyahat için gerekli olan tüm detayların yapay zeka destekli olarak sunulması, kullanıcıların bilgiye hızlı ve doğru bir şekilde ulaşmalarını sağlar.

Sonuç olarak, Travel Guide, seyahat etmeyi sevenler için vazgeçilmez bir yardımcı olmayı amaçlamaktadır. Kullanıcı dostu arayüzü ve kapsamlı özellikleriyle öne çıkan uygulama, seyahat deneyimlerini daha keyifli hale getirmek ve seyahat etmeyi herkes için daha erişilebilir kılmak için tasarlanmıştır.

BÖLÜM 1. GİRİŞ

Günümüzde teknolojinin hızla ilerlemesiyle birlikte, seyahat etme alışkanlıklarımız da büyük bir değişim geçirmektedir. İnsanlar, yeni yerleri görmek ve farklı kültürleri tanımak amacıyla seyahat etmeyi her zaman sevmişlerdir. Ancak, teknoloji gelişmeden önce seyahat etmek oldukça zorlu bir süreçti. Eski dönemlerde, insanlar ellerinde haritalar ve yerel halkın yardımıyla yol bulmak zorundaydılar. Günümüzde ise internet ve mobil uygulamalar sayesinde bu süreçler oldukça kolaylaştı. Seyahat etmeden önce gidilecek yer hakkında detaylı bilgilere ulaşmak ve seyahat sırasında ihtiyaç duyulan bilgilere anında erişmek mümkün hale geldi.

Mobil uygulamaların gelişmesiyle birlikte, seyahat deneyimleri daha da konforlu ve planlı bir hale geldi. Harita verilerinden, çeviri uygulamalarına, gezilecek yerler hakkında bilgi almaya kadar birçok işlem mobil uygulamalar aracılığıyla yapılabilir hale geldi. Özellikle yapay zeka teknolojisinin entegrasyonu, seyahat planlamasını ve bilgiye ulaşmayı daha hızlı ve etkili bir hale getirdi. Yapay zeka destekli uygulamalar, kullanıcıların internette uzun süre araştırma yapmadan ihtiyaç duydukları bilgilere hızla ulaşmalarını sağlıyor.

Bu bağlamda, Travel Guide adlı mobil uygulama projem, seyahat etmeyi kolaylaştırmak ve kullanıcıların seyahat deneyimlerini zenginleştirmek amacıyla geliştirildi. Travel Guide, kullanıcıların gezdikleri ve deneyim kazandıkları yerler hakkında gönderiler paylaşmalarını, diğer insanlara rehberlik etmelerini sağlıyor. Kullanıcılar, kendi profillerinde paylaşımlarını görebilir ve düzenleyebilirler, bu da onlara gezdikleri yerler hakkında bir tür seyahat defteri oluşturma imkanı tanır.

Bir yere seyahat etmek isteyen kullanıcılar, bu gönderilere göz atarak kendilerine bir rehber hazırlayabilir ve filtreleme özellikleri sayesinde gidecekleri yer ve kategoriye göre istedikleri bilgilere kolaylıkla ulaşabilirler. Ayrıca, uygulama kullanıcıları ülkelere ait genel bilgilere ve seyahat için gerekli bilgilere yapay zeka destekli bir

şekilde erişebilirler. Yapay zeka destekli bir chatbot da her zaman kullanıcıların yardımına hazırdır. Uygulamada bulunan harita özelliği ile kullanıcılar, uygulamadan çıkmadan istedikleri konum hakkında bilgiye kolaylıkla ulaşabilirler.

Travel Guide, kullanım kolaylığı, işlevselliği ve kullanıcı dostu arayüzü ile öne çıkmaktadır. Kullanıcıların her zaman yanlarında taşıyabilecekleri, kapsamlı bir seyahat rehberi niteliğindedir. Bu rapor, genel seyahat mobil uygulamaları eğilimlerine odaklanarak, Travel Guide'ın öne çıkan özelliklerini detaylı bir şekilde ele alacaktır.

1.1. Projenin Amacı

Travel Guide projesi, seyahat etmeyi sevenlerin ihtiyaçlarına cevap vermek ve seyahat deneyimlerini daha keyifli, planlı ve stressiz hale getirmek amacıyla geliştirilmiştir. Bu uygulama, kullanıcılara seyahat ederken ihtiyaç duyabilecekleri tüm bilgileri tek bir platformda sunmayı hedeflemektedir. Kullanıcıların yeni yerler keşfederken, bilgiye erişim sürecini kolaylaştırmak ve seyahatlerini daha verimli bir şekilde planlamalarına yardımcı olmak, projemizin temel amaçları arasındadır.

Travel Guide, kullanıcıların gezdikleri yerler hakkında deneyimlerini paylaşmalarına olanak tanıyarak, seyahat topluluğu içinde bilgi alışverişini teşvik etmektedir. Bu sayede, kullanıcılar başkalarının deneyimlerinden faydalanarak kendi seyahat planlarını oluşturabilirler. Aynı zamanda, kullanıcıların kendi seyahat defterlerini oluşturarak, gezdikleri yerler hakkında detaylı kayıtlar tutmalarını sağlamak da uygulamanın önemli bir hedefidir.

Uygulama, kullanıcıların seyahat ederken karşılaşabilecekleri dil engellerini aşmalarına yardımcı olacak çeviri araçları ve yapay zeka destekli chatbot ile donatılmıştır. Ayrıca, harita entegrasyonu sayesinde kullanıcılar, bulundukları yer hakkında anlık bilgiye erişebilir ve yollarını kolayca bulabilirler. Ülkeler hakkında genel bilgilerin ve seyahat için gerekli olan tüm detayların yapay zeka destekli olarak sunulması, kullanıcıların bilgiye hızlı ve doğru bir şekilde ulaşmalarını sağlamaktadır.

Travel Guide, seyahat etmeyi sevenler için vazgeçilmez bir yardımcı olmayı amaçlamaktadır. Kullanıcı dostu arayüzü, kapsamlı özellikleri ve işlevselliği ile kullanıcıların her an yanlarında taşıyabilecekleri pratik bir seyahat rehberi sunmaktadır. Projenin amacı, kullanıcıların seyahat deneyimlerini en üst düzeye çıkarmak ve seyahat etmeyi herkes için daha erişilebilir ve keyifli hale getirmektir.

BÖLÜM 2. MOBİL UYGULAMA

2.1. Mobil Uygulama Nedir?

Mobil uygulama, bir mobil cihazda (akıllı telefon, tablet, iPod vb.) çalışan ve bağımsız yazılımı destekleyen bir işletim sistemine sahip olan yazılım uygulamasıdır. Genellikle mobil işletim sisteminin sahibi tarafından işletilen uygulama dağıtım platformları aracılığıyla temin edilirler; örneğin Apple App Store, Google Play, Windows Phone Store ve BlackBerry App World. Mobil uygulamalar, mobil cihazda önceden yüklenmiş olabileceği gibi, kullanıcılar tarafından mobil uygulama mağazalarından veya İnternet üzerinden de indirilebilirler. Ayrıca, mobil uygulamalar genellikle kullanıcılara, genellikle masaüstü veya dizüstü bilgisayar üzerinden daha sık erişilen İnternet hizmetlerine bağlanmalarında veya taşınabilir cihazlarında İnternet'i kullanmalarını kolaylaştırmalarında yardımcı olur.[1] Bu uygulamalar, kullanıcı dostu arayüzleri ve hızlı erişim imkanları ile mobil cihaz kullanımını daha pratik hale getirir, aynı zamanda özelleştirilebilir seçenekler sunarak kullanıcıların ihtiyaçlarına daha uygun bir deneyim sunarlar.

2.2. Mobil Uygulamaların Geçmişi

Mobil uygulamaların geçmişi, mobil teknolojinin gelişimine paralel olarak önemli bir evrim geçirmiştir. İlk mobil uygulamalar, mobil cihazların başlangıç dönemlerinde genellikle sınırlı özelliklere sahip basit programlar olarak ortaya çıktı. 2000'li yılların başlarında, özellikle akıllı telefonların ve mobil işletim sistemlerinin yaygınlaşmasıyla birlikte, mobil uygulamalar daha karmaşık ve çeşitli hale geldi. Bu dönemde, uygulama geliştiricileri, kullanıcıların mobil cihazlarından daha fazla fayda sağlamalarını amaçlayan çeşitli uygulamalar geliştirmeye başladılar. 2010'lu yılların ortalarında ise Apple'ın App Store ve benzeri diğer uygulama mağazalarının açılmasıyla birlikte, mobil uygulamaların hızla popülerleştiği bir döneme giriş yapıldı. Bu platformlar, geliştiricilere uygulama dağıtımını kolaylaştırdıkları gibi,

kullanıcılara da geniş bir uygulama yelpazesi sunma imkanı tanıdı. Mobil uygulamaların bu tarihçesi, mobil teknolojinin kullanıcı deneyimini ve işlevselliğini kökten değiştiren bir süreci yansıtmaktadır.

2.3. Mobil Uygulama Geliştirme

Mobil uygulama geliştirme, günümüzde hızla evrilen teknolojinin getirdiği dinamik bir süreçtir. Bu süreç, çeşitli platformlarda çalışabilen, kullanıcı dostu ve işlevsel uygulamaların yaratılmasını amaçlar. Geliştirme süreci genellikle bir fikir veya ihtiyaçtan başlar ve ardından tasarım, yazılım geliştirme, test aşamalarıyla devam eder. İlk adım genellikle kapsamlı bir pazar araştırması ve kullanıcı ihtiyaçlarının analizi ile başlar. Tasarım aşamasında, kullanıcı arayüzü (UI) ve kullanıcı deneyimi (UX) ön planda tutularak uygulamanın estetik ve işlevselliği planlanır. Yazılım geliştirme aşamasında, programlama dilleri ve çeşitli geliştirme araçları kullanılarak uygulamanın kodlanması gerçekleştirilir. Test aşamasında ise uygulamanın farklı cihazlarda ve durumlarda sorunsuz çalışması sağlanır. Mobil uygulama geliştirme süreci, hızla değişen teknolojik trendlere ayak uydurmayı gerektirir ve geliştiricilere, kullanıcıların beklentilerini karşılayacak yaratıcı ve yenilikçi çözümler bulma fırsatı sunar. Bu sürecin başarılı olması, kullanıcıların etkileşimini maksimum düzeyde artırmak ve uygulamanın rekabet avantajını sürdürmek açısından kritiktir.

2.4. Mobil Uygulama Geliştirme Süreci

Mobil uygulama geliştirme süreci, genellikle bir dizi aşamadan oluşan karmaşık bir süreçtir. İlk olarak, projenin başında bir fikir veya ihtiyaç ortaya çıkar ve bu, pazar araştırması ve kullanıcı geri bildirimleriyle desteklenir. Bu aşamada, uygulamanın temel özellikleri, hedef kitlesi ve pazarlama stratejileri belirlenir. Ardından, tasarım aşamasında, kullanıcı arayüzü (UI) ve kullanıcı deneyimi (UX) ön planda tutularak, uygulamanın estetik ve kullanıcı dostu bir arayüze sahip olması sağlanır.

Yazılım geliştirme aşaması, belirlenen özelliklere uygun olarak uygulamanın kodlanmasıyla başlar. Bu aşamada, geliştiriciler genellikle belirli bir platform için uygun programlama dillerini ve çerçeveleri kullanarak uygulamanın temel

işlevselliğini oluştururlar. Geliştirme sürecinde, düzenli testler ve geri bildirim alınarak uygulamanın istikrarlı ve hatasız bir şekilde çalışması sağlanır.

Test aşamasında, uygulama farklı cihazlarda ve işletim sistemlerinde test edilir ve olası hatalar veya uyumsuzluklar giderilir. Bu aşama, kullanıcı deneyimini optimize etmek ve uygulamanın istikrarını sağlamak adına hayati bir rol oynar. Son olarak, uygulama dağıtım aşamasında, geliştirici tarafından seçilen platformlara (örneğin, App Store veya Google Play) uygulamanın sunulması ve kullanıcılara erişilebilir hale getirilmesi sağlanır. Bu süreç, mobil uygulamaların başarılı bir şekilde geliştirilmesini ve kullanıcılara etkili bir şekilde sunulmasını amaçlar.

2.5. Mobil Uygulamaların Seyahat ve Turizm Alanında Kullanılması

Turizme yönelik olarak geliştirilen mobil uygulamalar günümüzde akıllı telefonlar, tabletler ve akıllı saatler gibi çeşitli mobil cihazlarda kullanılabilmektedir. Akıllı telefonların yaygın kullanımının artması, seyahatleri destekleyici uygulamaların gelişiminde önemli bir etken olarak kabul edilmektedir. Mobil cihaz teknolojilerindeki sürekli ilerlemeler, sadece web sitelerini daha erişilebilir hale getirmekle kalmamakta, aynı zamanda mobil seyahat uygulamalarının da daha fazla ilgi görmesine neden olmaktadır. [2]

Mobil seyahat uygulamaları, gezginlere dünyayı keşfetme ve seyahatlerini planlama konusunda yardımcı olmak için çeşitli özellikler sunar. İşte mobil seyahat uygulamalarının sunduğu avantajlar ve popüler özellikler:

Konum Tabanlı Hizmetler (LBS): Mobil seyahat uygulamaları genellikle konum tabanlı hizmetleri kullanır. Bu, kullanıcıların bulundukları konuma göre öneriler almasını, yakındaki mekanları keşfetmesini ve rotalarını belirlemesini sağlar. Örneğin, restoranlar, oteller, müzeler ve diğer turistik yerler hakkında bilgi alabilirler.

Seyahat Planlama: Bu uygulamalar, kullanıcılara seyahatlerini planlamak için bir dizi araç sunar. Uçuş ve konaklama rezervasyonları yapma, rota oluşturma, etkinlikler planlama ve hava durumu tahmini gibi özelliklerle seyahat planlaması kolaylaşır.

Yerel Rehberlik: Mobil seyahat uygulamaları, kullanıcıların seyahat ettikleri yerlerdeki yerel kültürü ve etkinlikleri keşfetmelerine yardımcı olur. Yerel restoranlar, etkinlikler, festivaller ve diğer ilginç noktalar hakkında bilgi sunarlar.

Yol Tarifi ve Navigasyon: Seyahat edenler için yolda kaybolmak artık bir sorun değil. Mobil seyahat uygulamaları, kullanıcıların rotalarını belirlemesine, trafik durumu hakkında bilgi almasına ve navigasyon desteği sağlamasına yardımcı olur.

Sosyal Etkileşim: Bazı mobil seyahat uygulamaları, kullanıcıların seyahat deneyimlerini paylaşmalarına ve diğer gezginlerle etkileşimde bulunmalarına olanak tanır. Bu, seyahat önerileri almak, yerel rehberlik sağlamak ve yeni arkadaşlar edinmek için harika bir yol olabilir.

Ancak, mobil seyahat uygulamalarının bazı zorlukları da vardır. Örneğin, kullanıcıların güvenliği ve kişisel verilerinin korunması gibi endişeler bulunmaktadır. Ayrıca, her uygulamanın doğruluğu ve güncelliği konusunda güvenilir olup olmadığı da bir sorun olabilir.

Sonuç olarak, mobil seyahat uygulamaları, gezginlere seyahat deneyimlerini optimize etme ve dünya üzerindeki keşiflerini artırma fırsatı sunar. Ancak, doğru uygulamayı seçmek ve güvenlik önlemlerini almak önemlidir. Bu uygulamalar, modern seyahatçiler için vazgeçilmez bir araç haline gelmiştir ve gelecekte daha da yaygınlaşmaları beklenmektedir.

BÖLÜM 3. YAPAY ZEKA

Yapay zeka, bilgisayar sistemlerinin insan benzeri yetenekler kazanmasını sağlayan ve veri analizi, örüntü tanıma, karar verme gibi zeki davranışları gerçekleştirmek için kullanılan bir bilim dalıdır. Makine öğrenimi, doğal dil işleme ve görüntü işleme gibi alt alanlarıyla, verilerden öğrenme, dilin anlaşılması ve görüntülerin yorumlanması gibi karmaşık görevleri başarıyla yerine getirebilir. Bu teknoloji, otomasyon, sağlık, finans, ulaşım, e-ticaret ve daha birçok alanda kullanılarak insan hayatını önemli ölçüde etkilemekte ve geliştirmektedir. Örneğin, sağlık sektöründe yapay zeka, hastalık teşhisi, ilaç keşfi ve tedavi planlaması gibi alanlarda önemli katkılar sağlamaktadır.

3.1. Yapay Zekanın Geçmişi

Yapay zeka, insanlık tarihinin uzun bir sürecinde gelişen ve evrilen bir konsepttir. İnsanlar, antik çağlardan itibaren makinelerin insan benzeri zekaya sahip olabileceği fikrini düşünmüşlerdir. Ancak modern yapay zeka çalışmaları genellikle 20. yüzyılın ikinci yarısında başlamıştır. 1950'lerde, Alan Turing'in "Turing Testi" ile yapay zekanın temelleri atılmıştır. Daha sonra, 1956'da Dartmouth Konferansı'nda John McCarthy ve diğer araştırmacılar tarafından yapay zeka terimi resmen kullanılmıştır. 1950'ler ve 1960'lar boyunca, yapay zeka araştırmaları sembolik ve mantıksal yaklaşımlar üzerinde yoğunlaştı. Ancak, bu dönemde elde edilen ilerlemeler sınırlı kaldı. 1980'lerde ve 1990'larda, yapay zeka araştırmaları tekrar popülerlik kazandı ve özellikle uzman sistemler ve uzman sistem tabanlı yaklaşımlar ön plana çıktı. Günümüzde ise, büyük veri, derin öğrenme ve büyük hesaplama gücü gibi faktörlerle birlikte yapay zeka alanında büyük bir ivme yaşanmaktadır.

3.2. Yapay Zekanın Mobil Uygulamalarda Kullanımı

Yapay zeka, mobil uygulamalarda giderek daha yaygın bir şekilde kullanılmaktadır ve bu kullanım, mobil deneyimleri daha zengin, kişiselleştirilmiş ve verimli hale

getirmektedir. Mobil uygulamalarda yapay zeka, çeşitli alanlarda fayda sağlayabilir. Örneğin, sesli asistanlar (örneğin Siri, Google Assistant), kullanıcıların sesli komutlarını anlamak ve yanıtlamak için doğal dil işleme (NLP) tekniklerini kullanır. Görüntü işleme algoritmaları, fotoğraf ve videoları analiz ederek nesne tanıma, yüz tanıma ve etiketleme gibi işlevler sağlayarak kullanıcı deneyimini artırabilir. Makine öğrenimi, kullanıcı tercihlerini analiz ederek kişiselleştirilmiş içerik önerileri sunabilir veya uygulamanın performansını optimize edebilir. Ayrıca, yapay zeka destekli chatbotlar, müşteri hizmetleri süreçlerini otomatikleştirerek kullanıcıların sorularını yanıtlayabilir ve problemlerini çözebilir. Mobil uygulamalarda yapay zeka kullanımı, kullanıcıların ihtiyaçlarına daha hızlı ve etkili bir şekilde cevap verebilen, daha akıllı ve kullanıcı dostu uygulamaların geliştirilmesine olanak tanır. Bu da mobil uygulama pazarındaki rekabeti artırır ve kullanıcı memnuniyetini artırır.

3.3. Yapay Zekanın Önemi Ve Geleceği

Yapay zeka, tarihsel olarak çekişmeli bir konu olmuştur. Geliştiricilerin bazıları, bu teknolojinin muazzam bir ilerleme kaydettiğini ve insan hayatını önemli ölçüde dönüştürdüğünü savunurken, bazıları ise yapay zekanın potansiyel risklerini vurgulamıştır. Gelişen teknoloji, iş dünyasında önemli değişikliklere yol açmıştır. Teknolojik ilerlemeler, işgücü piyasasında dengesizliklere ve işlerin otomatikleştirilmesine yol açarken, bu durum bazıları tarafından endişe verici olarak algılanmaktadır. Önemli bir soru ise, Herbert Simon'un ifade ettiği gibi, "Makineler, bir insanın yapabileceği her işi yapabilir hale gelecek mi?" Bu soru, teknolojinin ilerlemesi ve insanların işgücündeki rolünün değişmesiyle ilgili önemli bir tartışma konusudur.[3]

Yapay zeka, günümüzde ve gelecekte insanlık için büyük öneme sahip olan bir teknolojidir. Bu teknoloji, birçok alanda devrim niteliğinde değişiklikler yapma potansiyeline sahiptir. Yapay zeka, otomasyon, veri analizi, sağlık, eğitim, ulaşım, güvenlik ve daha birçok sektördeki iş süreçlerini optimize etmek ve geliştirmek için kullanılabilir. Veri analizi ve makine öğrenimi sayesinde, büyük veri setlerinden anlamlı bilgiler çıkarabilir ve daha iyi kararlar alabiliriz. Sağlık sektöründe yapay zeka, hastalık teşhisi, ilaç keşfi ve tedavi planlaması gibi alanlarda önemli rol

oynamaktadır. Eğitimde, öğrenci performansını izlemek ve öğrenme süreçlerini kişiselleştirmek için kullanılabilir. Ulaşım sektöründe, sürücüsüz araçlar gibi yenilikçi çözümler geliştirmek için yapay zeka büyük bir potansiyele sahiptir. Yapay zeka teknolojileri, bu ve benzeri alanlarda hızla ilerlemekte ve gelişmektedir. Gelecekte yapay zeka, daha da yaygınlaşacak ve insan yaşamının her alanında daha fazla varlık gösterecektir. Ancak, bu teknolojinin etik ve güvenlik konularına dikkat edilmesi gerekmektedir. Gelecekte yapay zeka, insanlığın yaşam kalitesini artırmak ve daha sürdürülebilir bir dünya için önemli bir araç olmaya devam edecektir.

BÖLÜM 4. PROJE YAPISI VE TASARIMI

Projede MVVM (Model-View-ViewModel) mimarisi kullanılmıştır. Temiz kod ve sürdürülebilirlik prensiplerine dikkat edilmiştir. Proje düzeni, kodların amaçlarına göre farklı klasörlerde tutulmasıyla daha okunabilir ve yönetilebilir hale getirilmiştir. Model, repository, service, utils, viewmodel ve view gibi klasörler, ilgili proje dosyalarının düzenli bir şekilde yer almasını sağlamaktadır.

4.1. Model Katmanı

Bu katmanda uygulama veri modelleri yer alır. Her model, belirli bir veri tipini temsil eder ve uygulamanın çeşitli bölümlerinde kullanılır. Aşağıda bulunan model sınıfları, uygulamada kullanılan veri yapılarını temsil eder:

Country: Ülke verilerini temsil eden bir modeldir. Ülke adı, başkenti, nüfusu gibi bilgileri içerir.

Explore: Kullanıcıların paylaştığı seyahat deneyimlerini temsil eden bir modeldir. Başlık, açıklama, ülke gibi bilgileri içerir.

Profile: Kullanıcı profillerini temsil eden bir modeldir. Kullanıcı adı, e-posta, doğum tarihi gibi bilgileri içerir.

User: Uygulama kullanıcılarını temsil eden bir modeldir. Kullanıcı adı, e-posta, şifre gibi bilgileri içerir.

4.2. Repository Katmanı

Bu katmanda veri kaynaklarına erişim sağlayan sınıflar bulunur. Her bir repository sınıfı, belirli bir veri kaynağıyla (örneğin Firebase veritabanı, API çağrıları) iletişim kurar ve veri işleme işlemlerini gerçekleştirir. Örnek repository sınıfları şunlardır:

AuthRepository: Kullanıcı kimlik doğrulama işlemleriyle ilgili Firebase Auth servisiyle iletişim kurar. Kullanıcı kaydı, giriş, şifre sıfırlama gibi işlemleri yönetir.

ExploreRepository: Kullanıcıların seyahat deneyimleriyle ilgili Firebase veritabanıyla iletişim kurar. Keşif oluşturma, düzenleme, silme gibi işlemleri gerçekleştirir.

ProfileRepository: Kullanıcı profilleriyle ilgili Firebase veritabanıyla iletişim kurar. Profil oluşturma, düzenleme, silme gibi işlemleri gerçekleştirir.

CountryRepository: Ülke verileriyle ilgili API çağrıları yapar ve uygulamada kullanılan ülke bilgilerini yönetir.

4.3. ViewModel Katmanı

Bu katmanda kullanıcı arayüzü (UI) bileşenlerini temsil eden view model sınıfları yer alır. Her bir view model sınıfı, ilgili veri işleme işlemlerini gerçekleştirir ve view'a (fragment veya aktiviteye) gerekli verileri sağlar. Örnek view model sınıfları şunlardır:

AuthViewModel: Kullanıcı kimlik doğrulama işlemleriyle ilgili işlemleri yönetir. AuthRepository ile iletişim kurar ve kullanıcı kimlik doğrulama işlemlerini gerçekleştirir.

ExploreViewModel: Kullanıcıların seyahat deneyimleriyle ilgili işlemleri yönetir. ExploreRepository ile iletişim kurar ve seyahat deneyimleriyle ilgili veri işleme işlemlerini gerçekleştirir.

ProfileViewModel: Kullanıcı profilleriyle ilgili işlemleri yönetir. ProfileRepository ile iletişim kurar ve profil işlemlerini gerçekleştirir.

CountryViewModel: Ülke verileriyle ilgili işlemleri yönetir. CountryRepository ile iletişim kurar ve ülke verilerini işler.

4.4. Service Katmanı

Bu katmanda harici servislerle iletişim kurulur. Servislerin oluşturulması, yönetilmesi ve kullanılması bu katmanda gerçekleştirilir. Örnek servisler şunlardır:

CountriesAPI: Ülke verilerini çekmek için kullanılan bir API'yi temsil eder. Retrofit kütüphanesi kullanılarak API çağrıları gerçekleştirilir.

RetrofitClient: Retrofit kütüphanesinin yapılandırılması ve API çağrıları için temel yapıyı sağlar.

4.5. Utils Katmanı

Bu katmanda yardımcı işlevler ve araçlar yer alır. Genellikle tekrar kullanılabilir yardımcı sınıflar ve araçlar içerir. Örnekler şunlardır:

ApiUtils: API çağrılarını yapılandırmak ve oluşturmak için kullanılan yardımcı bir sınıftır.

DateUtils: Tarih ve saat işlemleri için yardımcı işlevler içeren bir sınıftır.

4.6. View Katmanı

Bu katmanda kullanıcı arayüzünün (UI) görüntülendiği bileşenler bulunur. Activity'ler, Fragment'ler ve XML dosyaları bu katmanda yer alır. Örnekler şunlardır:

AuthActivity: Kullanıcı kimlik doğrulama işlemlerini yöneten bir aktivitedir.

MainActivity: Uygulamanın ana aktivitesidir.

Diğer fragmentlar ve aktiviteler: Kullanıcı arayüzünün farklı bileşenlerini temsil eder ve ilgili view modellerle etkileşim kurarak veri işleme işlemlerini gerçekleştirir.

Bu yapı, uygulamanın farklı bileşenlerini ve bunlar arasındaki ilişkileri net bir şekilde tanımlar. Her katmanın belirli bir sorumluluğu vardır ve bu, kodun düzenli, okunabilir ve bakımı yapılabilir olmasını sağlar.

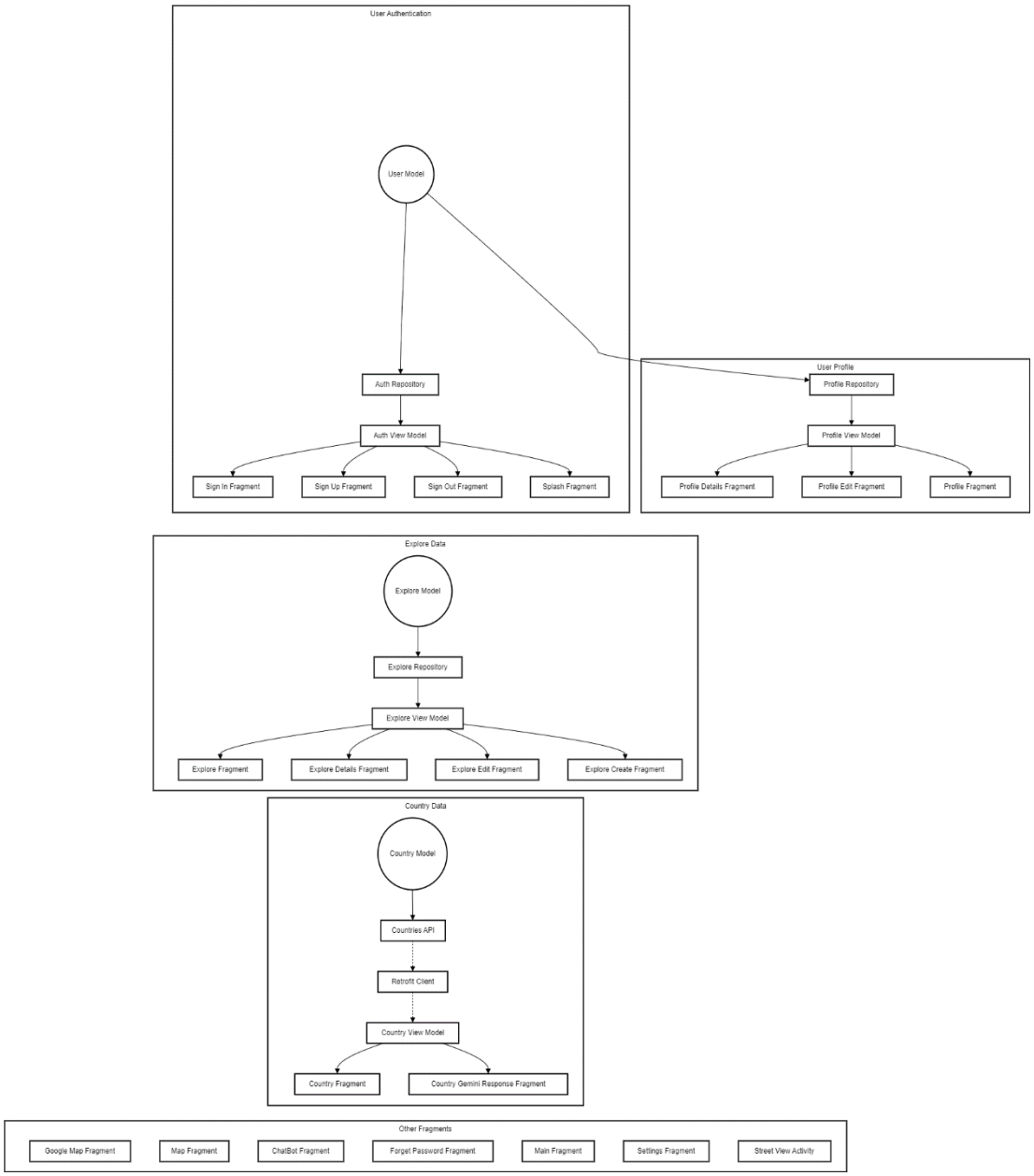
4.7. Veri Akış Şeması

Veri akış şeması, bir uygulamadaki bilgi akışını ve veri hareketlerini görselleştiren bir diyagramdır. Bu şema, uygulamanın içerdiği bileşenler arasındaki veri akışını ve ilişkileri gösterir. Genellikle farklı katmanlar arasındaki veri alışverişini, kullanıcı etkileşimlerini ve veritabanı işlemlerini açıklamak için kullanılır.

4.7.1. Uygulama içindeki veri akışı ve işleyiş

Kullanıcı etkileşimiyle başlayan bir işlem, View Katmanı'ndaki bir bileşende gerçekleşir. Bu etkileşim ViewModel'e iletilir. ViewModel, gerekli veri işlemlerini

gerçekleştirir ve Repository Katmanı'na erişir. Repository, veri kaynaklarından veri alır veya veri kaynaklarına veri gönderir. Alınan veya işlenen veri, ViewModel aracılığıyla View Katmanı'na iletilir. Sonuç, kullanıcı arayüzünde görüntülenir veya kullanıcıya geri bildirim olarak iletilir. Bu şekilde, her katmanın belirli bir sorumluluğu vardır ve veri akışı katmanlar arasında düzenli bir şekilde gerçekleşir.



Şekil 4.1. Uygulama veri akış şeması

BÖLÜM 5. PROJEDE KULLANILAN TEKNOLOJİLER

Projede, mobil uygulama geliştirmek için Android işletim sistemi tercih edilmiştir. Android Studio, resmi bir entegre geliştirme ortamı olarak kullanılarak Java ve Kotlin programlama dillerini desteklemiştir. Yazılım mimarisi olarak MVVM (Model-View-ViewModel) benimsenmiş ve yazılım dili olarak Kotlin tercih edilmiştir. Veritabanı yönetimi için Firebase kullanılmıştır. Firebase, authentication işlemleri için Firebase Authentication, veri depolama işlemleri için Realtime Database, büyük veriler için ise Firebase Storage kullanılmıştır. Projede API kullanımı yapılmıştır. Ülke bilgileri için restCountries API, yapay zeka entegrasyonu için ise Google Gemini API kullanılmıştır. API kullanımları için Retrofit ve JSON Converter kütüphaneleri tercih edilmiştir. Proje sürecinde kod versiyon kontrolü ve işbirliği için Git ve GitHub entegre edilmiştir. Ayrıca, kullanıcı tabanının geniş ve küresel olması göz önüne alınarak çoklu dil desteği projenin önemli bir özelliği olarak ele alınmıştır.

5.1. Mobil İşletim Sistemi

Mobil işletim sistemleri, taşınabilir cihazların, özellikle akıllı telefonların ve tabletlerin çalışmasını sağlayan temel yazılım platformlarıdır. Bu işletim sistemleri, cihazın temel fonksiyonlarını yönetir ve uygulama geliştirmeyi destekler. Örnek olarak, Android, iOS, Windows Phone ve BlackBerry işletim sistemleri yaygın olarak kullanılmaktadır. Her bir mobil işletim sistemi, benzersiz bir kullanıcı arayüzü, özellik seti ve güvenlik önlemleri sunar. Ayrıca, uygulama mağazalarını barındırarak kullanıcılara çeşitli uygulamalara erişim imkanı tanır. Mobil işletim sistemleri, cihazların performansını optimize ederek enerji verimliliği sağlar ve geniş bir uygulama ekosistemini destekleyerek kullanıcılara çeşitli deneyimler sunar. Bu işletim sistemleri, mobil teknolojinin yaygınlaşmasıyla birlikte sürekli olarak güncellenmekte ve geliştirilmektedir, böylece kullanıcılar en son teknolojik yeniliklerden yararlanabilirler.

Projemde Android işletim sistemini kullanmayı tercih ettim. Android, mobil uygulama geliştirme için oldukça popüler bir işletim sistemidir ve geniş bir kullanıcı kitlesine hitap etmektedir.

5.1.1. Android işletim sistemi

Android, mobil cihazlar için bir yazılım yığınıdır ve işletim sistemi, ara katman yazılımı ve temel uygulamaları içerir. Google Inc., yazılımın ilk geliştiricisi olan Android Inc.'i 2005 yılında satın almıştır. Android'in mobil işletim sistemi, modifiye edilmiş bir Linux çekirdeği üzerine kuruludur. Google ve Open Handset Alliance üyeleri, Android'in geliştirilmesi ve yayınlanması konusunda işbirliği yapmışlardır. Android'in bakımı ve daha fazla geliştirilmesi görevi Android Open Source Project (AOSP) tarafından üstlenilmiştir. Android işletim sistemi, dünya genelinde en çok satan akıllı telefon platformudur. Android, cihazların işlevselliğini genişleten uygulamaları yazan büyük bir geliştirici topluluğuna sahiptir. Şu anda Android için 150.000'den fazla uygulama bulunmaktadır. Android Market, Google tarafından işletilen çevrimiçi bir uygulama mağazasıdır, ancak uygulamalar üçüncü taraf sitelerden de indirilebilir. Geliştiriciler, genellikle Java dilinde yazıp cihazı Google tarafından geliştirilmiş Java kütüphaneleri aracılığıyla kontrol eder. Android dağıtımının 5 Kasım 2007'deki tanıtımı, Open Handset Alliance'ın kuruluşu ile birlikte duyuruldu. Bu, mobil cihazlar için açık standartları ilerletmeye adanmış 80 donanım, yazılım ve telekom şirketinden oluşan bir konsorsiyumdur. Google, Android kodunun büyük bir kısmını Apache Lisansı altında, ücretsiz yazılım ve açık kaynak lisansı olarak yayınlamıştır.

Android açık kaynak yazılım yığını, Java uygulamalarını içeren, Java tabanlı, nesne yönelimli bir uygulama çerçevesi üzerinde çalışan, JIT derlemesi içeren Dalvik sanal makinesi üzerinde çalışan bir yapıya sahiptir. C dilinde yazılmış kütüphaneler arasında yüzey yöneticisi, OpenCore medya çerçevesi, SQLite ilişkisel veritabanı yönetim sistemi, OpenGL ES 2.0 3D grafik API, WebKit düzen motoru, SGL grafik motoru, SSL ve Bionic libc yer almaktadır. Android işletim sistemi, Linux çekirdeği dahil olmak üzere yaklaşık 12 milyon satır kod içerir, bunun içinde 3 milyon satır

XML, 2.8 milyon satır C, 2.1 milyon satır Java ve 1.75 milyon satır C++ bulunmaktadır.[4]

5.2. Yazılım Geliştirme Ortamı

Yazılım geliştirme ortamı (IDE), bir yazılım geliştiricisinin yazılım üretmek için kullandığı bir araç veya yazılım paketidir. IDE'ler, yazılım geliştirme sürecini kolaylaştırmak ve hızlandırmak için geliştiricilere bir dizi araç ve özellik sunar. Bu özellikler genellikle bir metin düzenleyici, derleyici, hata ayıklayıcı, kod tamamlama, yapılandırma yöneticisi ve sürüm kontrol sistemi entegrasyonu gibi unsurları içerir. Ayrıca, IDE'ler genellikle projeleri organize etmek, kodu yönetmek ve işbirliği yapmak için kullanılan ek araçlar ve işlevler sunar. Önde gelen IDE'ler arasında Android Studio, Visual Studio, Eclipse, IntelliJ IDEA ve Xcode gibi platformlar bulunmaktadır. Bu yazılımlar, farklı programlama dilleri ve platformlar için özelleştirilmiş geliştirme deneyimleri sunarlar.

5.2.1. Android Studio

Android Studio, Android uygulama geliştirmek için kullanılan resmi entegre geliştirme ortamıdır. Google tarafından sağlanan bu geliştirme ortamı, geliştiricilere Android uygulamalarını kolayca tasarlama, kodlama ve test etme imkanı sunar. Android Studio, Java ve Kotlin gibi popüler programlama dillerini destekler ve kullanıcı dostu bir arayüzle gelir, böylece geliştirme sürecini daha verimli hale getirir. Zengin özellik seti, hata ayıklama araçları, derleme ve dağıtım süreçlerini yönetme yetenekleri, Android Studio'yu geliştiriciler için güçlü bir araç haline getirir. Ayrıca, Android Studio, Google'ın sunduğu güncel SDK'lar ve diğer geliştirme kaynaklarıyla entegre çalışarak, geliştiricilere en yeni Android platform özelliklerinden ve iyileştirmelerinden yararlanma fırsatı tanır. Android uygulama geliştirme sürecinde kullanılan bu geliştirme ortamı, profesyonel ve bağımsız geliştiricilerin Android platformunda etkileyici ve güncel uygulamalar oluşturmalarına olanak sağlar.

Şekil 5.1. Android Studio

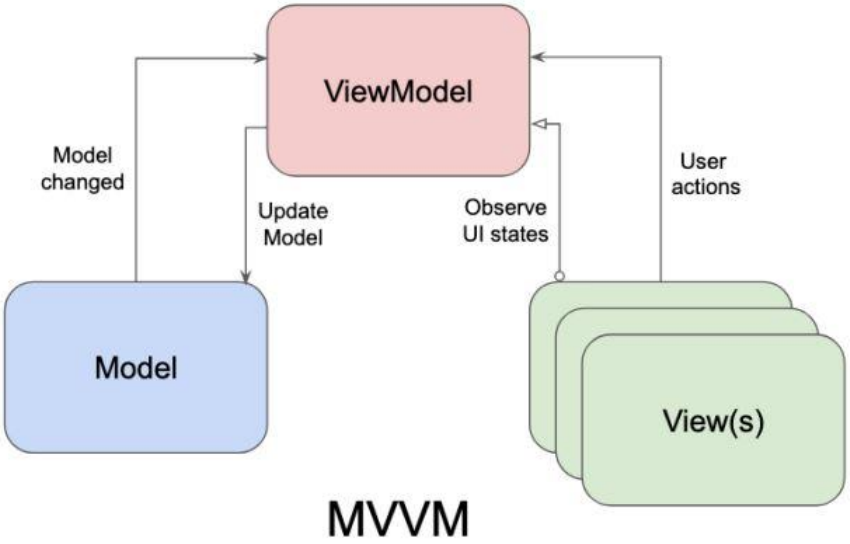
5.3. Yazılım Mimarisi

Yazılım mimarisi, bir yazılım sisteminin tasarımının temel yapı taşlarını ve bileşenlerini düzenleme ve organize etme sürecini ifade eder. Bu disiplin, bir yazılım projesinin genel yapılandırmasını belirlemeyi, modüler bir yaklaşım benimsemeyi ve sistemin parçalarının birbirleriyle etkileşimini düzenlemeyi amaçlar. Yazılım mimarisi, genellikle performans, güvenilirlik, genişletilebilirlik ve kullanılabilirlik gibi faktörleri göz önünde bulundurarak, sistem genelinde bir bütün olarak tutarlılık ve etkileşim sağlamayı hedefler. Bu, büyük ölçekli ve karmaşık yazılım projelerinde, geliştiricilerin daha iyi bir organizasyon ve daha sürdürülebilir bir kod tabanı oluşturmalarına yardımcı olur. Yazılım mimarisi, bir projenin gereksinimlerine uygun bir çerçeve oluşturarak, geliştirme ekibine rehberlik eder ve uzun vadeli bakımı ve genişletilmesini kolaylaştırır. Ayrıca, paylaşılan kaynakların etkili bir şekilde kullanılmasını sağlar ve yazılım geliştirmenin genel verimliliğini artırır.

5.3.1. MVVM (Model-View-ViewModel) Mimarisi

1980'lerde Model View ViewModel (MVVM) tasarım deseni, Smalltalk'ta MVC'nin sınırlamalarını aşmak ve bazı güçlü yönlerinden faydalanmak amacıyla tanıtıldı. Aslında, hem MVC hem de MVVM, Sorumlulukların Ayrılması (SoC) konseptini benimser ve bu da kod kalitesini arttırır. Ancak, MVVM, MVC'den daha fazla sorumluluğu ayırır, bu da karmaşıklığı azaltır ve test edilebilirlik ile yeniden kullanılabilirliği maksimize eder. MVVM, View objesinin sorumluluğunu MVC'deki Controller objesinin üstlendiği verileri hazırlama mantığına sahip yeni bir nesne olan

ViewModel nesnesini tanıttı. Ayrıca, MVC'deki View objeleri ve Controller objeleri bir çift olarak çalışırken, MVVM bunları tek bir nesne içinde birleştirip ona View objeleri adını verdi. Bu sayede, MVVM veri bağımsızlığını artırır ve uygulama mantığı kapsülleme. Son bir çalışma, MVVM tasarım desenini kullanarak kodu sıkıştırdığını ve esnek hale getirdiğini gösteriyor.[5]



Şekil 5.2. Model-View-ViewModel mimarisi

Projemde, yazılım geliştirmeyi daha etkili ve düzenli hale getirmek adına MVVM (Model-View-ViewModel) mimarisini benimsedim. Bu yapı, her bir görünüm (view) için ayrı bir viewmodel oluşturarak kullanıcı arayüzü ve iş mantığını daha modüler bir şekilde ayırmama olanak tanıdı. Ayrıca, veri kullanımı gerektiren kısımlarda model yapılarını oluşturarak, veri katmanını ve iş mantığını birbirinden bağımsız hale getirdim. Kodlarımın büyük bir kısmını repository sınıflarında yazarak, veri erişimini ve yönetimini tek bir noktada topladım. Bu sayede, kod tekrarını önledim, bakımı kolaylaştırdım ve kodun daha okunabilir ve sürdürülebilir olmasını sağladım. MVVM mimarisi, proje boyunca tutarlı bir yapı sağlayarak, kod karmaşıklığını azalttı ve geliştirme sürecini daha verimli kıldı. Bu yöntemle, yazılım geliştirmenin yanı sıra bakım ve genişletme süreçlerinde de daha etkili bir yaklaşım benimsemiş oldum.

5.4. Mobil Programlama Dilleri

Mobil programlama dilleri, taşınabilir cihazlar üzerinde çalışan uygulamaları geliştirmek için kullanılan programlama dilleridir. Mobil uygulama geliştirme sürecinde yaygın olarak kullanılan diller arasında Java, Kotlin, Swift, Objective-C ve C# bulunmaktadır. Android platformunda genellikle Java veya Kotlin tercih edilirken, iOS platformu Swift ve Objective-C'yi destekler. C# ise özellikle Microsoft'un Xamarin framework'ü aracılığıyla hem Android hem de iOS için uygulama geliştirmek amacıyla kullanılır. Bu diller, taşınabilir cihazlara özgü fonksiyonlar ve özelliklere uyumlu bir şekilde çalışabilen uygulamalar oluşturmak üzere tasarlanmıştır. Her bir dilin kendine özgü avantajları ve kullanım alanları bulunmaktadır; örneğin, Kotlin'in Android geliştirmesi için modern bir dil olması ve Swift'in iOS geliştirmesi için Apple tarafından önerilmesi gibi. Geliştiriciler, hedefledikleri platforma ve proje gereksinimlerine bağlı olarak uygun olan mobil programlama dilini seçerek, etkili ve performanslı uygulamalar geliştirebilirler.

Projemde Kotlin programlama dilini kullanmayı tercih ettim. Kotlin, özellikle Android uygulama geliştirmesi için tasarlanmış, modern, ifade edici ve güvenilir bir dil olarak bilinir. Java ile uyumlu olması ve aynı zamanda kendi başına bir programlama dili olarak kullanılabilmesi, Kotlin'in geliştiricilere esneklik ve verimlilik sağlamasına olanak tanır. Projenin gereksinimleri ve Kotlin'in temiz, anlaşılır syntax'ı, uygulama geliştirmesini daha etkili ve keyifli hale getirmeme yardımcı oldu.

5.4.1. Kotlin programalama dili

Kotlin, Temmuz 2011'de JetBrains tarafından tanıtılan bir nesne yönelimli ve statik tipli bir programlama dilidir. Google tarafından, Mart 2017'de Android geliştirmesi için resmi bir dil olarak tanıtıldı. Kotlin, Java Sanal Makinesi üzerinde çalışacak şekilde tasarlandı, bu da Kotlin ile oluşturulan uygulamaların Java bytecode'a derlendiği anlamına gelir, bu da Kotlin uygulamalarının Java çalışma ortamını destekleyen herhangi bir makinede çalışmasına olanak tanır.[6]

5.5. Veritabanı Yönetimi

Veritabanı yönetimi, bir kuruluşun veya bir bireyin verilerini organize etmek, saklamak, güncellemek ve korumak için kullanılan bir dizi süreci ve stratejiyi içerir. Veritabanı yönetimi, veritabanlarını oluşturma, bakımını yapma, yedekleme ve güvenliğini sağlama süreçlerini içerir. Veritabanı yöneticileri, verilerin düzenli ve etkili bir şekilde depolanmasını ve erişilmesini sağlamak için veritabanı sistemlerini yönetirler. Veritabanı tasarımı, verilerin mantıklı bir şekilde düzenlenmesi ve ilişkilendirilmesi sürecini içerirken, performans optimizasyonu, yedekleme stratejileri ve veri bütünlüğü de yönetimin önemli yönlerindendir. Bu süreçler, veritabanının güvenilir ve etkili bir şekilde çalışmasını sağlamak, veri kaybını önlemek ve kullanıcıların verilere güvenilir bir şekilde erişmelerini sağlamak için kritiktir. Veritabanı yönetimi, günümüzde birçok farklı veritabanı yönetim sistemini içerebilir ve kurumların büyüklüğüne ve ihtiyaçlarına göre çeşitlilik gösterir.

Veritabanı çeşitleri arasında ilişkisel, belgelik, grafik, nesne tabanlı ve zaman serisi gibi çeşitli türler bulunmaktadır. Ancak, projemde Firebase veritabanını tercih ettim. Firebase, geliştiricilere gerçek zamanlı veritabanı çözümleri sunan bulut tabanlı bir platformdur. Firebase'in sunduğu gerçek zamanlı veritabanı, mobil uygulamalarda kullanım için son derece uygun ve etkilidir. Bu veritabanı, uygulama verilerini senkronize etmek ve anlık güncellemeleri sağlamak için özel olarak tasarlanmıştır. Mobil uygulamalar için hızlı, güvenilir ve ölçeklenebilir bir veritabanı çözümü olan Firebase, uygulama geliştirme sürecini kolaylaştırır ve geliştiricilere zaman kazandırır. Ayrıca Firebase'in sunduğu diğer özellikler, kullanıcı kimlik doğrulama, bulut depolama, analitik ve bildirim gibi, geliştiricilere kapsamlı bir mobil uygulama geliştirme platformu sağlar. Bu nedenle, Firebase'in mobil uygulamalarda veritabanı çözümü olarak tercih edilmesi, uygulamanın performansını artırırken geliştirme sürecini de iyileştirir.

5.5.1. Firebase

Firebase, Google tarafından geliştirilen bir bulut tabanlı veritabanı yapısıdır. Firebase, büyük ve karmaşık verileri etkili bir şekilde depolamak ve hızlı sorgular

yapmak amacıyla oluşturulmuş doküman tabanlı bir NoSQL veritabanıdır. Firestore, esnek bir yapıya sahiptir, bu sayede veriler istenilen şemada tutulabilir. Veriler, doküman adını taşıyan yapılar içinde saklanır. Doküman topluluklarına ise koleksiyonlar adı verilir. Bu özellikleriyle Firestore, diğer NoSQL veritabanlarına benzerlik gösterse de, kendi içinde özelleştirilebilir ve kullanıcı dostu bir yapı sunar.[7]

Şekil 5.3. Firebase

5.5.1.1. Firebase Authentication

Uygulamada kullanıcı kayıt işlemleri için Firebase Authentication kullanılmaktadır. Kullanıcı kayıt yaptıktan sonra, Firebase Authentication tarafından sağlanan birinci sınıf güvenlik ve kimlik doğrulama hizmetinden yararlanılarak kullanıcının email ve şifre bilgileri güvenli bir şekilde saklanır. Firebase Authentication, kullanıcıları kimlik doğrulama, şifre sıfırlama ve giriş işlemleri için önceden yapılandırılmış işlevler sunar.

Uygulama, kullanıcıdan alınan email ve şifre bilgilerini Firebase Authentication aracılığıyla doğrulayarak kayıt işlemini tamamlar. Kullanıcının şifresinin en az 6 karakterden oluşması gerektiği ve doğru bir şekilde girilip girilmediği kontrol edilir. Kayıt işlemi başarıyla tamamlandığında, kullanıcının email ve şifre bilgileri Firebase Authentication tarafından güvenle saklanır ve giriş yapması için kullanılır.
Bu sayede uygulama, Firebase'in sunduğu güçlü kimlik doğrulama özellikleriyle kullanıcı verilerini güvenli bir şekilde yönetir ve uygulamanın genel güvenliğini sağlar.

Identifier	Providers	Created ↓	Signed in	User UID
elif.yilmaz92@gmail.com	✉	23 May 2024	23 May 2024	Hp6RJzzb4qR7gJHnHaRzmY...
sofia.garcia95@gmail.c...	✉	23 May 2024	23 May 2024	7TVxSyVdrGR6ts2KsitCfB6Eo...
michael.thompson87@...	✉	23 May 2024	23 May 2024	mkFnRwl1SPgCy8sO5juX6y99...
serdar@gmail.com	✉	22 May 2024	22 May 2024	DyOWvOeK6WSDOlr5u46BETI...

Şekil 5.4. Firebase kullanıcı kayıt tablosu

5.5.1.2. Firebase Realtime Database

Uygulamada, Firebase Realtime Database kullanılarak bazı veri kayıtları yapılmaktadır. Firebase Realtime Database, verilerin bulutta gerçek zamanlı olarak saklanmasını ve senkronize edilmesini sağlayan bir veritabanı hizmetidir. Bu hizmet, geliştiricilere veritabanı yönetimini kolaylaştırır ve hızlı veri güncellemeleri yapma imkanı sunar. Ayrıca, Firebase Realtime Database, çevrimdışı kullanım desteği sayesinde kullanıcıların internet bağlantısı olmadığında bile verilerini kullanabilmelerini sağlar. Veriler, internet bağlantısı sağlandığında otomatik olarak senkronize edilir.
Firebase Realtime Database'in avantajları arasında:

Gerçek Zamanlı Senkronizasyon: Veriler, tüm bağlı istemcilerde anlık olarak güncellenir.
Kolay Entegrasyon: Firebase SDK'ları ile kolayca entegre edilebilir ve kullanımı basittir.
Çevrimdışı Destek: Kullanıcılar çevrimdışı olduğunda bile veri ekleyip güncelleyebilir, bağlantı sağlandığında otomatik olarak senkronize edilir.
Yüksek Performans: Büyük miktarda veriyi hızlı ve verimli bir şekilde yönetir.

Projemde, Firebase Realtime Database'de iki ana bölüm bulunmaktadır: Explore ve Profil kısımları.

Profil Kısmı: Bu bölümde kullanıcı profillerinin kayıtları tutulmaktadır. Kullanıcı sisteme kayıt olduğunda, kullanıcının email, kullanıcı adı ve doğum tarihi gibi veriler

burada saklanır. Ayrıca, kullanıcılar daha sonra profillerini güncelleyerek profil fotoğrafı ve biyografi ekleyebilirler. Bu bilgiler de veritabanında anlık olarak güncellenmektedir.

Explore Kısmı: Bu bölümde ise kullanıcıların paylaşmış oldukları gönderiler kayıt altında tutulmaktadır. Paylaşılan gönderilerin başlık, ülke, mekan, kategori, detaylar, puanlama numarası, kullanıcı e-posta ve fotoğraf linki gibi bilgileri anlık olarak saklanmaktadır. Kullanıcı bir güncelleme veya silme işlemi yaptığında, veritabanı bu değişiklikleri gerçek zamanlı olarak yansıtır.

Bu yapı sayesinde, uygulama kullanıcılarına hızlı, güvenilir ve sürekli güncellenen bir veri yönetim sistemi sunulmaktadır. Firebase Realtime Database'in sağladığı avantajlar, uygulamanın kullanıcı deneyimini önemli ölçüde artırmaktadır.

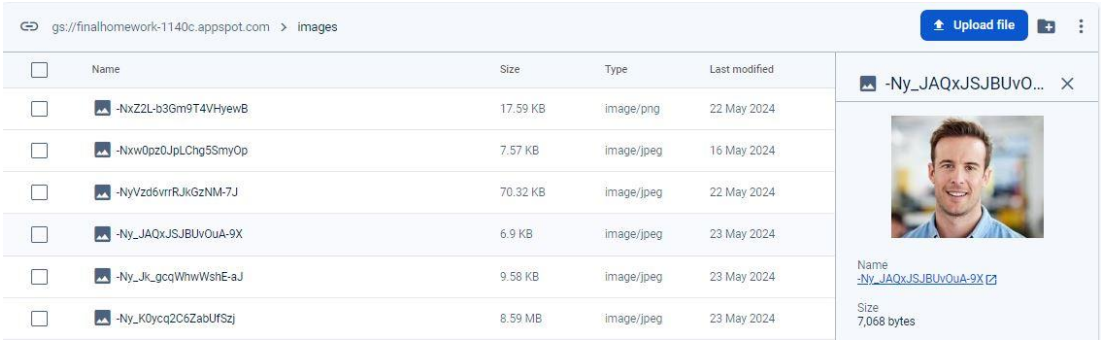
Şekil 5.5. Firebase realtime database veri kayıtları tablosu

5.5.1.3. Firebase Storage

Projede, daha büyük boyuttaki verileri saklamak için Firebase Storage kullanılmaktadır. Firebase Storage, fotoğraf ve video gibi büyük veri dosyalarının güvenli bir şekilde bulutta saklanmasını sağlar. Bu veritabanı, yüksek performanslı ve ölçeklenebilir bir yapı sunarak, geliştiricilere verilerini kolayca yönetme imkanı tanır.

Projemizde, profil fotoğrafları ve kullanıcıların paylaştıkları gönderilere ait fotoğrafları saklamak için Firebase Storage'dan faydalandım. Firebase Storage, kullanıcıların profil fotoğraflarını ve gönderi resimlerini güvenli ve verimli bir şekilde depolayarak, bu verilerin hızlı bir şekilde erişilebilir olmasını sağlar. Ayrıca, Firebase Storage ile entegre edilen güçlü güvenlik ve doğrulama özellikleri, verilerin yetkisiz erişimlere karşı korunmasına yardımcı olur.

Bu yapı, uygulamanın kullanıcı deneyimini iyileştirirken, büyük veri dosyalarının yönetimini ve depolanmasını da kolaylaştırmaktadır. Firebase Storage'ın sunduğu avantajlar sayesinde, kullanıcıların profil ve gönderi fotoğrafları güvenli bir şekilde saklanmakta ve ihtiyaç duyulduğunda hızlı bir şekilde erişilebilmektedir.



Şekil 5.6. Firebase storage fotoğraflar tablosu

5.6. API

API (Application Programming Interface), bir yazılım uygulamasının diğer yazılım uygulamaları ile iletişim kurmasını sağlayan bir arayüzdür. API'ler, farklı yazılım bileşenlerinin birlikte çalışmasını kolaylaştırır ve veri alışverişini standart bir formatta gerçekleştirir. Bu, geliştiricilerin karmaşık işlevleri ve hizmetleri uygulamalarına kolayca entegre etmelerine olanak tanır.

API'ler, çeşitli türlerde olabilir ve farklı protokoller kullanabilir. REST (Representational State Transfer) ve SOAP (Simple Object Access Protocol) en yaygın kullanılan API türlerindendir. REST API'leri, HTTP protokolü üzerinden

çalışır ve genellikle JSON (JavaScript Object Notation) formatında veri döner. Bu özellikleri sayesinde, REST API'leri mobil uygulamalar için ideal bir çözüm sunar.

5.6.1. API'lerin mobil uygulamalarda kullanımı

Mobil uygulamalarda API'ler, uygulamaların işlevselliğini ve kullanıcı deneyimini büyük ölçüde artırır. API'ler, üçüncü taraf hizmetlere erişim sağlamak, veri alışverişini gerçekleştirmek ve uygulamaların yeteneklerini genişletmek için kullanılır. Mobil uygulamalarda API kullanımının bazı yaygın alanları şunlardır:

Veri Entegrasyonu: API'ler, mobil uygulamaların harici veri kaynaklarına erişmesini sağlar. Örneğin, bir hava durumu uygulaması, güncel hava durumu verilerini bir hava durumu API'sinden alabilir.
Sosyal Medya Entegrasyonu: Sosyal medya API'leri, kullanıcıların Facebook, Twitter veya Instagram hesaplarına bağlanmasını ve bu platformlarda içerik paylaşmasını sağlar.
Harita ve Konum Servisleri: Google Maps API gibi harita ve konum servisleri, kullanıcıların harita üzerinde konum bilgilerini görmesini ve rota planlaması yapmasını sağlar.
Ödeme Sistemleri: Ödeme API'leri, mobil uygulamalara güvenli ödeme işlemleri eklenmesini sağlar. Örneğin, PayPal veya Stripe API'leri ile uygulamalara ödeme entegrasyonu yapılabilir.
Bildirim Servisleri: FCM (Firebase Cloud Messaging) gibi bildirim API'leri, uygulamaların kullanıcılarına anlık bildirimler göndermesine olanak tanır.

Yapay Zeka ve Makine Öğrenimi: Yapay zeka API'leri, mobil uygulamalara dil işleme, görüntü tanıma ve diğer yapay zeka özellikleri eklemeyi sağlar. Örneğin, Google Cloud Vision API, fotoğraflardaki nesneleri tanımlamak için kullanılabilir.

API'lerin mobil uygulamalarda kullanımı, geliştiricilere esneklik ve hız kazandırır. Bu sayede, uygulamalar daha işlevsel, kullanıcı dostu ve dinamik hale gelir. API entegrasyonu, mobil uygulamaların piyasaya daha hızlı sunulmasını sağlar ve

geliştirme sürecini kolaylaştırır. Ayrıca, API'ler sayesinde uygulamalar, sürekli olarak güncel ve doğru verilerle çalışabilir, bu da kullanıcı memnuniyetini artırır.

5.6.2. Projede kullanılan API'ler

Projemde çeşitli verileri ve işlevleri sağlamak için farklı API'lerden faydalandım. Ülke bilgilerini sağlamak için RestCountries API'yi kullandım. Yapay zeka destekli bir uygulama geliştirmek amacıyla, Google'ın yapay zeka aracı olan Google Gemini API'den faydalandım. Harita işlemleri ve konum bilgileri için ise Google Maps API ve Google Places API'yi entegre ettim. Bu API'ler, projenin işlevselliğini ve kullanıcı deneyimini büyük ölçüde artırmıştır.

5.6.2.1. Google gemini API

Google Gemini API, Google'ın yapay zeka ve makine öğrenimi hizmetleri sunan bir aracıdır. Bu API, uygulamalara yapay zeka yetenekleri eklemek için kullanılabilir. Projemde, yapay zeka destekli özellikler sunmak amacıyla Google Gemini API'yi entegre ettim. Bu sayede, kullanıcıların daha kişiselleştirilmiş ve akıllı bir deneyim yaşamalarını sağlamayı hedefledim. Ayrıca projeme bu api ile çalışan bir chatbot ekledim. Google Gemini API, kullanıcı verilerini analiz ederek, önerilerde bulunma ve çeşitli akıllı işlemleri gerçekleştirme gibi işlevler sunar.

Şekil 5.7. Google Gemini

5.6.2.2. Restcountries API

RestCountries API, dünya üzerindeki ülkeler hakkında çeşitli bilgileri sağlayan bir web servisidir. Bu API'yi kullanarak ülke adları, bayrakları, nüfusları, yüzölçümleri, dilleri, para birimleri ve daha birçok bilgiye erişim sağlanabilir. Projemde,

kullanıcıların seyahat planları yaparken ihtiyaç duyabilecekleri ülke bilgilerini sağlamak için RestCountries API'den yararlandım. Bu API, uygulamama kapsamlı ve güncel ülke bilgilerini kolayca entegre etmemi sağladı.

5.6.2.3. Google maps API

Google Maps API, harita ve coğrafi konum bilgileri sunan bir servistir. Bu API, dünya genelindeki haritaları ve konum bilgilerini uygulamalara entegre etmeyi sağlar. Projemde, kullanıcıların gezilecek yerleri bulmalarına ve bu yerler hakkında bilgi almalarına yardımcı olmak için Google Maps API'yi kullandım. Kullanıcılar, uygulama üzerinden harita üzerinde arama yapabilir ve belirli konumları inceleyebilir.

Şekil 5.8. Google Maps

5.6.2.4. Google places API

Google Places API, yerel işletmeler ve ilgi çekici noktalar hakkında bilgi sağlayan bir servistir. Bu API, kullanıcıların yakınlardaki restoranlar, kafeler, müzeler ve diğer ilgi çekici yerler hakkında bilgi almasını sağlar. Projemde, kullanıcıların seyahat ettikleri yerlerde keşfetmek istedikleri mekanları bulmalarına yardımcı olmak için Google Places API'yi kullandım. Kullanıcılar, uygulama üzerinden yakınlarındaki yerleri arayabilir, değerlendirmeleri okuyabilir ve mekanlar hakkında detaylı bilgilere erişebilir.

5.7. Uygulama Kütüphaneleri

Mobil uygulama geliştirme sürecinde, çeşitli işlevleri kolaylaştırmak ve geliştirme sürecini hızlandırmak için bir dizi kütüphane kullanılır. Uygulama kütüphanesi, belirli bir işlevi yerine getirmek için tasarlanmış kod parçaları veya modüller topluluğudur. Bu kütüphaneler, geliştiricilere tekrar eden işleri kolayca yapabilme imkanı sunar ve projeye hızla entegre edilebilecek önceden yazılmış kodlar içerir. Kütüphaneler, veri yönetiminden API entegrasyonuna, kullanıcı arayüzü geliştirmesinden veritabanı yönetimine kadar geniş bir yelpazede yardımcı olur.

5.7.1. Retrofit

Retrofit, Android uygulamalarında HTTP tabanlı web servislerine erişmek için kullanılan popüler ve güçlü bir kütüphanedir. Square Inc. tarafından geliştirilen Retrofit, RESTful API'lerle çalışmayı son derece kolay ve etkili hale getirir. Bu kütüphane, özellikle API isteklerini ve yanıtlarını yönetme sürecini basitleştirerek geliştiricilere büyük kolaylık sağlar.

Retrofit, basit ve anlaşılır bir API sunar. RESTful servislerle etkileşim kurmak için gereken kod miktarını minimuma indirir. Sadece birkaç satır kodla, bir web servisine GET, POST, PUT, DELETE gibi HTTP istekleri göndermek mümkündür. JSON veri formatını otomatik olarak Java nesnelerine dönüştürmek için Gson veya Moshi gibi JSON dönüştürücü kütüphanelerle entegre çalışabilir. Bu, JSON verilerini manuel olarak ayrıştırma ihtiyacını ortadan kaldırır ve hata riskini azaltır.

Retrofit, hem asenkron hem de senkron HTTP isteklerini destekler. Asenkron istekler, arka planda çalışır ve kullanıcı arayüzünü bloklamaz, böylece daha iyi bir kullanıcı deneyimi sağlar. Senkron istekler ise doğrudan çalıştırılır ve sonucu hemen döner. Ayrıca, URL'leri ve istek parametrelerini dinamik olarak belirlemeyi sağlar. Bu özellik, API uç noktalarının ve parametrelerinin kolayca yapılandırılabilmesine olanak tanır.

Retrofit, özel dönüştürücüler, araya giren katmanlar (interceptor) ve çeşitli adaptörlerle genişletilebilir. Bu, geliştiricilere kütüphaneyi ihtiyaçlarına göre özelleştirme esnekliği sunar. Örneğin, bir API'den kullanıcı bilgilerini çekmek için Retrofit ile sadece birkaç satır kod yazmanız yeterlidir. Bu şekilde, API entegrasyon süreçlerini çok daha verimli hale getirebilirsiniz.

Sonuç olarak, Retrofit, Android geliştiricileri için RESTful API'lerle çalışmayı son derece kolaylaştıran güçlü bir kütüphanedir. JSON dönüşümü, asenkron istekler, dinamik URL'ler ve genişletilebilirlik gibi özellikleriyle, geliştiricilerin daha hızlı ve hatasız bir şekilde web servislerine entegre olmasına yardımcı olur.

Ben de projemde Retrofit kütüphanesini kullanarak, API'leri kolay ve verimli bir şekilde entegre ettim. Retrofit'in sağladığı basit ve anlaşılır API sayesinde, RESTful servislerle iletişim kurmak için gereken kod miktarını minimuma indirdim. Bu sayede, ülke bilgilerini sağlamak için RestCountries API'yi, yapay zeka destekli işlemler için Google Gemini API'yi ve harita işlemleri için Google Maps API ile Google Places API'yi projeme hızlıca entegre edebildim. Retrofit'in JSON dönüşümü ve asenkron istek desteği, API entegrasyon sürecini oldukça kolaylaştırdı ve projeyi daha verimli hale getirdi.

5.7.2. Picasso

Picasso, Android platformunda resim yükleme ve önbelleğe alma işlemlerini kolaylaştıran popüler bir kütüphanedir. Bu kütüphane, Square tarafından geliştirilmiş olup, uygulamaların ağ üzerinden resim indirip göstermelerini verimli ve hafıza dostu bir şekilde yapmalarını sağlar.

Picasso, Android cihazların bellek yönetimiyle uyumlu olarak çalışır, bu da uygulamaların performansının artmasına yardımcı olur. Bellekte yer kaplamayan resimleri otomatik olarak temizler ve böylece bellek sızıntılarını önler. Disk önbelleğe alma özelliği sayesinde, aynı resimlerin tekrar tekrar indirilmesi gereksinimini azaltarak veri kullanımını düşürür ve uygulamanın daha hızlı tepki vermesini sağlar.

Picasso ayrıca yerel dosyalardan, kaynak dosyalardan ve uzak sunuculardan resim yükleyebilme yeteneği ile esnek bir çözüm sunar. Resimleri özelleştirme kapasitesi de yüksektir; döndürme, boyutlandırma ve kesme gibi işlemleri destekler.

Bu kütüphane, kullanım kolaylığı açısından da öne çıkar. Geliştiriciler, çok az kod ile hızlı ve etkili bir şekilde resim yükleme işlemleri gerçekleştirebilirler. Bu özellikleri ile Picasso, Android geliştirme sürecinde resim yönetimi için güçlü ve tercih edilen bir araçtır.

Projemde, Firebase veritabanından URL olarak çektiğim fotoğrafları, Picasso kütüphanesi sayesinde ImageView içerisinde kolayca gösterebildim. Profil fotoğrafları ve gönderi fotoğrafları gibi öğeler, veritabanından kolayca alınarak Picasso kullanılarak ekranlarda başarıyla gösterilmektedir. Bu süreç, uygulamanın kullanıcı arayüzünün daha dinamik ve etkileşimli olmasını sağlar, aynı zamanda veri çekme ve görsel gösterim işlemlerini sorunsuz bir şekilde hızlandırır.

5.8. Versiyon Kontrol Sistemi

Versiyon kontrol sistemi (VCS), yazılım geliştirme süreçlerindeki dosya değişikliklerini izleyen, yöneten ve organize eden bir araçtır. Geliştiricilerin eş zamanlı olarak aynı projede çalışmalarını sağlayan bu sistem, projenin farklı versiyonlarını saklamak, geçmiş değişiklikleri takip etmek ve gerektiğinde önceki sürümlere dönmek gibi önemli avantajlar sunar. VCS'nin kullanılması, yazılım geliştirme ekibinin işbirliğini artırır ve kod tabanındaki değişiklikleri düzenli bir şekilde yönetmelerine yardımcı olur. Popüler VCS araçları arasında Git, Mercurial ve Subversion bulunmaktadır. Git, dağıtık bir yapıya sahip olması ve hızlı performansıyla öne çıkar, bu nedenle birçok açık kaynak ve özel projede tercih edilen bir versiyon kontrol sistemi haline gelmiştir. VCS, yazılım geliştirme sürecinde güvenilirlik, işbirliği ve geçmiş değişikliklere erişim gibi kritik unsurları sağlayarak projelerin başarılı bir şekilde yönetilmesine katkıda bulunur.

Projemde Git versiyon kontrol sistemini tercih ettim. Git, dağıtık yapısı ve hızlı performansıyla öne çıkan bir versiyon kontrol sistemidir. Proje geliştirme sürecinde eş zamanlı çalışma, geçmiş değişikliklere erişim ve kod tabanının etkili yönetimi için Git'in esnek ve güvenilir yapısından faydalandım. Bu, kod değişikliklerini düzenli ve güvenilir bir şekilde yönetmeme olanak tanıdı.

5.8.1. Git ve github

Git, dağıtık bir versiyon kontrol sistemidir ve yazılım geliştirme süreçlerinde kullanılan popüler bir araçtır. Geliştiricilere projelerini izleme, değişiklikleri kaydetme, farklı sürümler arasında geçiş yapma ve işbirliği içinde çalışma imkanı tanır. Projelerin geçmişini, değişiklikleri ve kolları (branch) takip etmek için kullanılır.

GitHub, Git Kaynak Kodu Yönetimi'ni (SCM) kullanan popüler bir kod barındırma web sitesidir. Yazılım depolarını barındırmanın yanı sıra, GitHub, geliştirmenin sosyal yönlerini de içerir. GitHub kullanıcıları topluluk tarafından görülebilen profillere sahiptir ve kullanıcı eylemleri diğer topluluk üyeleri tarafından takip edilebilir ve izlenebilir. Kullanıcının kimliği ve eylemlerinin bu şekilde sosyal olarak entegre edilmesi, GitHub'a özgüdür.[8]

Projemde, versiyon kontrolü ve işbirliği için Git ve GitHub'u etkili bir şekilde kullandım. Proje sürecinde her bir önemli gelişmeyi ve kod değişikliğini bir Git deposunda kaydederek, projenin geçmişini izledim. Bu sayede, herhangi bir aşamada geri dönüp önceki versiyonlara erişebilir, hataları düzeltebilir ve geliştirmeleri takip edebilirim. Ayrıca GitHub üzerinde projenin ana deposunu oluşturarak, diğer geliştiricilere ve ilgili kişilere projenin gelişimini paylaşma ve katkıda bulunma imkanı sundum. Projemin Github linki:
https://github.com/serdararici/TravelGuideMobileApp

Şekil 5.9. Github

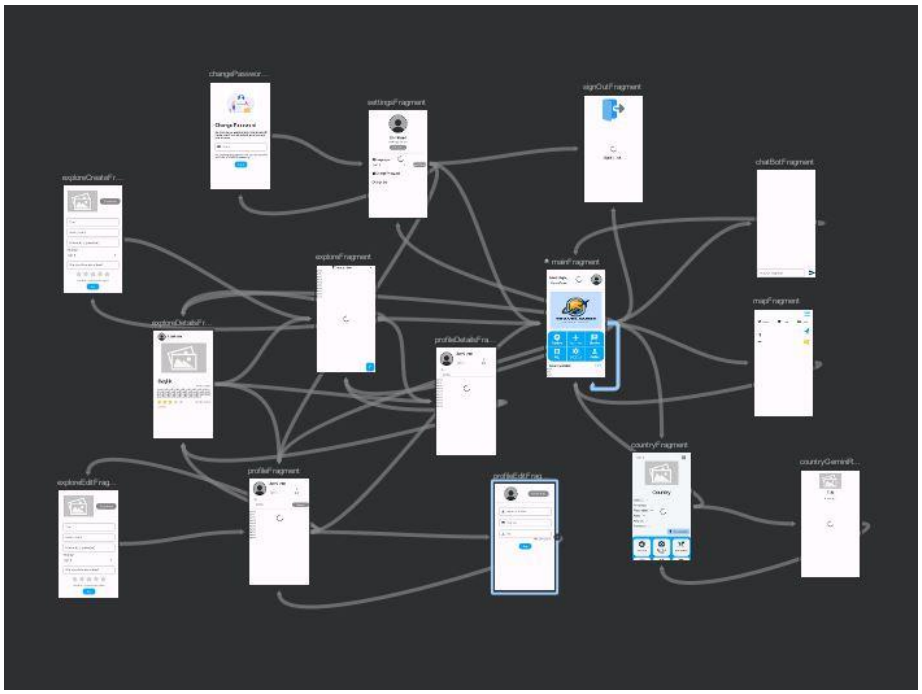
5.9. Çoklu Dil Desteği

Çoklu dil desteği, bir yazılım uygulamasının veya internet sitesinin farklı dillerdeki kullanıcılarına etkili bir şekilde hizmet verebilme yeteneğini ifade eder. Bu özellik, kullanıcıların tercih ettikleri dilde içerik görmelerine, iletişim kurmalarına ve uygulamayı kullanmalarına olanak tanır. Çoklu dil desteği, genellikle kullanıcı arayüzü, metin içeriği, hata mesajları ve diğer iletişim unsurlarını içerir. Bu, global ölçekte kullanılan uygulamalar, web siteleri veya yazılımlar için önemli bir özelliktir. Çoklu dil desteği, farklı kültürlerden ve dil gruplarından gelen kullanıcılar için daha çekici ve erişilebilir bir deneyim sunarak geniş bir kullanıcı kitlesine hitap etme avantajı sağlar. Ayrıca, bu özellik, işletmelerin ve geliştiricilerin ürünlerini veya hizmetlerini küresel pazarlara açma ve uluslararası düzeyde rekabet avantajı elde etme konusunda kritik bir rol oynar.

Ben de projemde bu konuya önem verdim. Uygulamam bir seyahat uygulaması olduğu için, dil desteği büyük önem taşır. Dünyanın her yerinden kullanıcıların uygulamayı kolaylıkla kullanabilmesi gerekmektedir. Şimdilik projemde İngilizce, Türkçe, Almanca, İspanyolca ve Fransızca olmak üzere beş dil desteği mevcuttur. İlerleyen zamanlarda bu dil seçeneklerini arttırmayı planlıyorum.

BÖLÜM 6. UYGULAMA TASARIMI

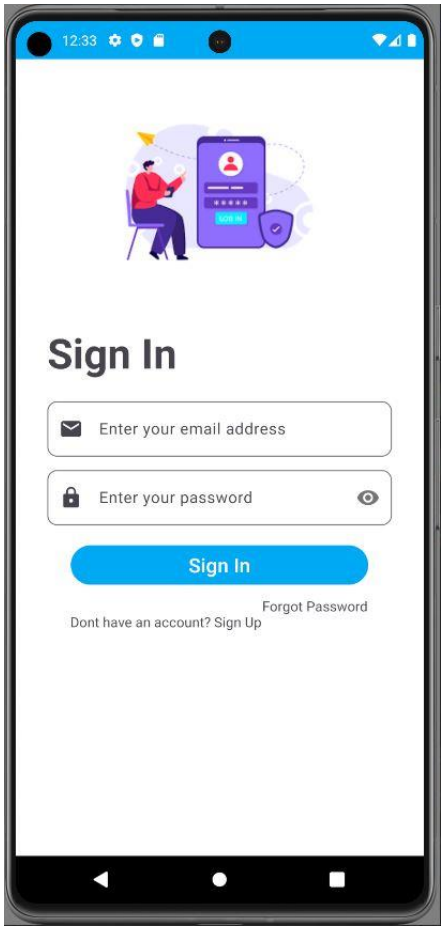
Uygulama tasarımı, kullanım kolaylığı ve işlevselliği ön planda tutarak dikkatle oluşturuldu. Gerekli tüm ekranlar, birbiriyle bağlantılı bir şekilde tasarlanmıştır ve bu tasarım Navigation Graph içinde Şekil 5.1'de gösterilmiştir.



Şekil 6.1. Uygulama tasarımı navigation graph gösterimi

6.1. Giriş Yap Ekranı

Uygulamanın giriş yap ekranında, kullanıcıdan kayıtlı bir e-posta adresi ve şifre istenir. Kullanıcı tarafından sağlanan bu bilgiler, Firebase Authentication üzerinden email ve şifre kontrolü için kullanılır. Eğer kullanıcının girdiği email ve şifre doğruysa, uygulama kullanıcı girişi yapar; yanlış ise hata mesajları ile kullanıcıya bilgilendirme yapılır. Ayrıca, bu ekran üzerinde yer alan "Şifremi Unuttum" seçeneği sayesinde, kullanıcı şifresini unuttuğunda, açılan ekranda girilen e-posta adresine şifre sıfırlama bağlantısı gönderilir. "Hesabın yok mu? Kayıt Ol." seçeneği ise kullanıcıyı kayıt olma ekranına yönlendirir, böylece yeni bir hesap oluşturabilir.



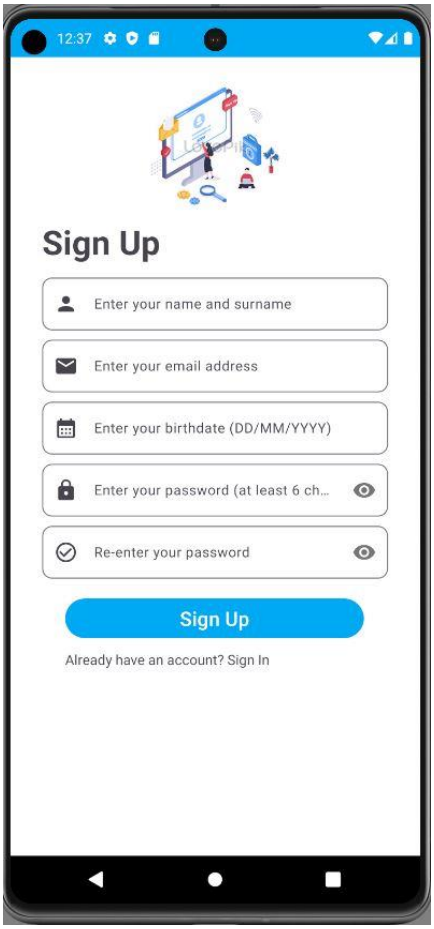
Şekil 6.2. Giriş yap sayfa tasarımı

6.2. Kayıt Ol Ekranı

Kayıt ol ekranı, kullanıcıdan isim, soyisim, e-posta, doğum tarihi ve şifre bilgilerini eksiksiz girmesini isteyen bir form içermektedir. Doğum tarihi alanına tıklandığında, kullanıcının tarih seçme işlemini daha kolay yapabilmesi için bir takvim görüntülenir. Şifre, en az 6 karakterden oluşmalıdır; bu gereksinim kullanıcıya şifre alanında belirtilmiştir. Ayrıca, şifrenin doğru girildiğini doğrulamak için "Şifrenizi Tekrar Girin" alanı bulunmaktadır.

Kullanıcı, tüm bilgileri eksiksiz girdiği takdirde "Kayıt Ol" butonuna tıkladığında Firebase Authentication işlemi başlatılır ve kullanıcının e-posta ve şifresi kaydedilir. Bu işlem sırasında aynı zamanda bir profil nesnesi oluşturulur ve buraya isim, soyisim, e-posta ve doğum tarihi gibi bilgiler aktarılır. Bu bilgiler, Firebase veritabanında profil kısmında saklanır.

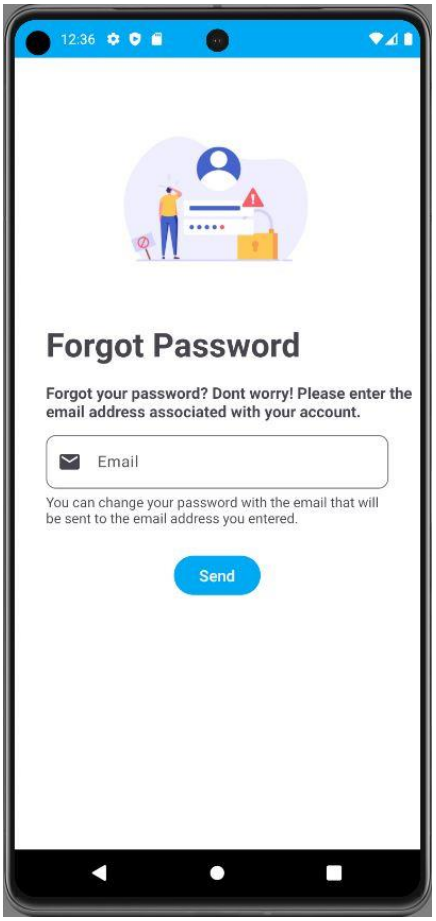
Eğer kullanıcı zaten bir hesaba sahipse, "Zaten bir hesabın var mı? Giriş Yap" bağlantısı aracılığıyla giriş yapma ekranına geri dönebilir. Bu sayede kullanıcılar, kolayca kayıt olabilir veya mevcut hesaplarına giriş yapabilirler.



Şekil 6.3. Kayıt ol sayfa tasarımı

6.3. Şifremi Unuttum Ekranı

Şifremi Unuttum ekranı, uygulamaya giriş yapmaya çalışan ve şifresini hatırlayamayan kullanıcılar için oluşturulmuştur. Kullanıcı, şifresini hatırlamıyorsa giriş yap ekranındayken "Şifremi Unuttum" seçeneğine tıklayarak bu ekrana ulaşabilir. Burada geçerli bir e-posta adresi girdikten sonra, kullanıcının e-posta adresini kontrol etmesi gerektiği bildirilir. Kullanıcı, e-posta adresine gelen mailden şifresini kolayca yenileyebilir.



Şekil 6.4. Şifremi unuttum sayfa tasarımı

6.4. Anasayfa Ekranı

Anasayfa ekranı, uygulamaya giriş yapıldığında kullanıcıların ilk karşılaştığı ekrandır. Bu nedenle, kullanıcıların ihtiyaç duyduğu pek çok bilgi burada gösterilmektedir. Sayfanın en üstünde, giriş yapan kullanıcının profil bilgileri veritabanından alınarak kullanıcının ismi "İyi günler [Kullanıcı adı]" şeklinde yazdırılır ve profil fotoğrafı görüntülenir. Alt kısımda, dünyanın çeşitli bölgelerinden şehir fotoğraflarının ve isimlerinin gösterildiği bir slider mevcuttur. Bu slider, kullanıcıya daha etkili bir tasarım sunmayı amaçlamaktadır. Sliderın altında, önemli sayfalara kolayca ulaşılabilmesi için bir menü bulunmaktadır. Menünün altında ise, son eklenen beş gönderinin yer aldığı bir kısım vardır. Burada, "Son Eklenenler" başlığı altında gönderiler gösterilir. Kullanıcı, "Tümünü Gör" butonuna tıklayarak keşfet sayfasına gidip tüm gönderileri de görebilir.

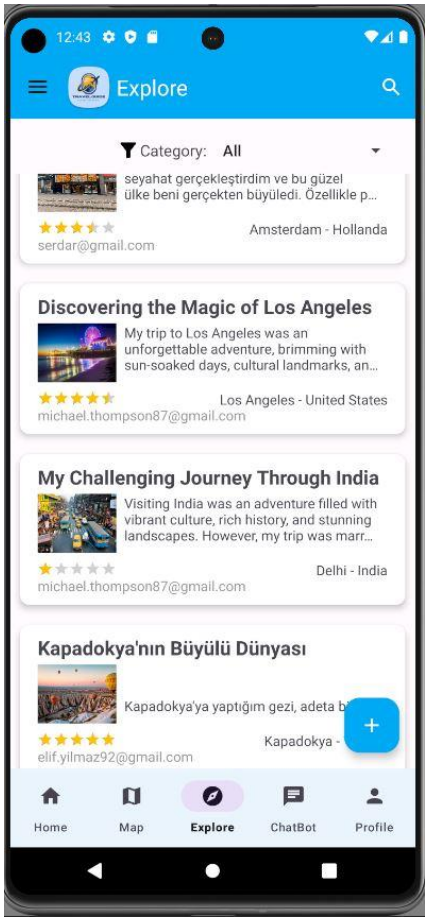
Şekil 6.5. Anasayfa sayfa tasarımı

6.5. Keşfet Ekranı

Keşfet ekranı, kullanıcıların paylaşmış oldukları gönderilerin gösterildiği ekrandır. Gönderiler üzerinde başlık, fotoğraf, detay yazısı, puanlama, kullanıcı maili ve ülke-mekan ismi yer almaktadır. Bu veriler Firebase Realtime veritabanından anlık olarak çekilmektedir. Sayfa açıldığında kullanıcı tüm gönderileri görebilir. Ayrıca, üst tarafta bulunan kategori seçeneği ile istenilen kategorideki gönderiler gösterilebilir. Kullanıcı üstteki spinner menüyü açtığında yedi seçenekle karşılaşır: Hepsi, Tarih ve Kültür, Ulaşım ve Konaklama, Yemek, Doğa ve Macera, Eğlence, Deneyim ve Tecrübe. Bu kategorilerden birini seçebilir.

Filtreleme işlemi için en üstte bulunan arama çubuğu da kullanılabilir. Bu arama çubuğu sayesinde kullanıcı, aradığı kelimeye göre başlıkta ya da ülke, mekan ismine göre eşleşen gönderileri görebilir. Gönderilerin üzerine tıklandığında detay sayfası

açılır ve kullanıcı gönderiyi daha detaylı inceleyebilir. Gönderilerin üzerinde kullanıcı mail hesapları da bulunmaktadır; bu hesaplara tıklanıldığında o kullanıcının profil sayfası görüntülenebilir. Son olarak, sayfanın alt sağ köşesinde bulunan FAB (Floating Action Button) butonuna tıklanarak kullanıcı yeni bir gönderi ekleme sayfasına yönlendirilir.

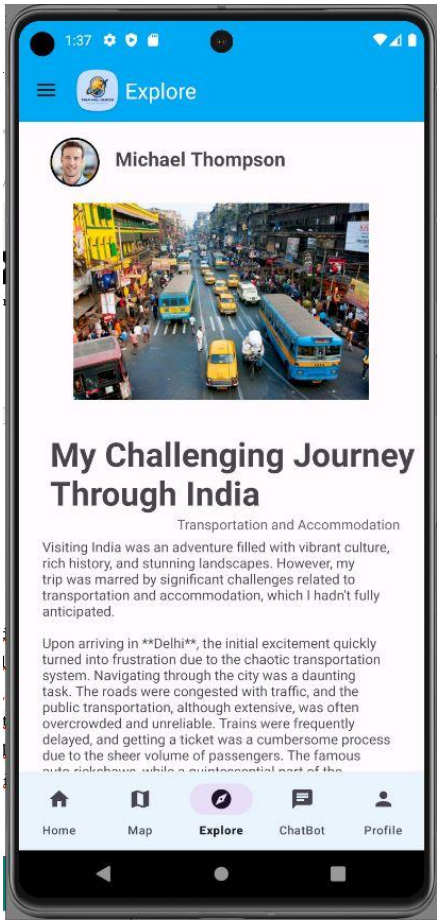


Şekil 6.6. Keşfet sayfa tasarımı

6.6. Gönderi Detay Ekranı

Gönderi detay ekranı, kullanıcıya gönderi ile ilgili daha fazla bilgi vermeye yarayan bir ekrandır. Seçilen gönderiye göre bu ekran açılır ve tüm bilgiler bu sayfada görülebilir. Sayfanın üst tarafında, gönderiyi paylaşan kullanıcının profil resmi ve kullanıcı adı yer almaktadır. Alt tarafta ise gönderinin görseli bulunmaktadır. Bu görselin altında gönderi başlığı, kategorisi ve detaylı yazısı yer almaktadır. En alt kısımda ise gönderinin rating puanı, oluşturulma tarihi ve ülke-mekan bilgisi yer

almaktadır. Bu şekilde, kullanıcılar seçilen gönderi hakkında kapsamlı bilgiye erişebilir.



Şekil 6.7. Gönderi detay sayfa tasarımı

6.7. Gönderi Ekleme Ekranı

Gönderi ekleme ekranı, kullanıcıların bir gönderi paylaşmak için gerekli bilgileri ekleyebilecekleri sayfadır. Bu sayfada, kullanıcı gönderi için bir fotoğraf seçebilir ve fotoğrafı paylaşımına ekleyebilir; ancak isterse fotoğraf eklemeyebilir. Ayrıca, gönderi ile ilgili bilgileri gerekli bölümlere yazabilir. Başlık, ülke adı, şehir veya mekan adı, paylaşım yapacağı yazı gerekli TextInput alanlarına girilir. Kategori seçimi için bir spinner bulunmakta olup, kullanıcı burada istediği kategoriyi kolaylıkla seçebilir. Alt kısımda, "Bu mekanı tavsiye eder misiniz?" yazısıyla birlikte bir rating bar bulunmaktadır. Burada kullanıcı, yaptığı paylaşım hakkında ne kadar tavsiye ettiğini belirterek gönderiye puan verebilir. Son olarak, kaydet butonuna

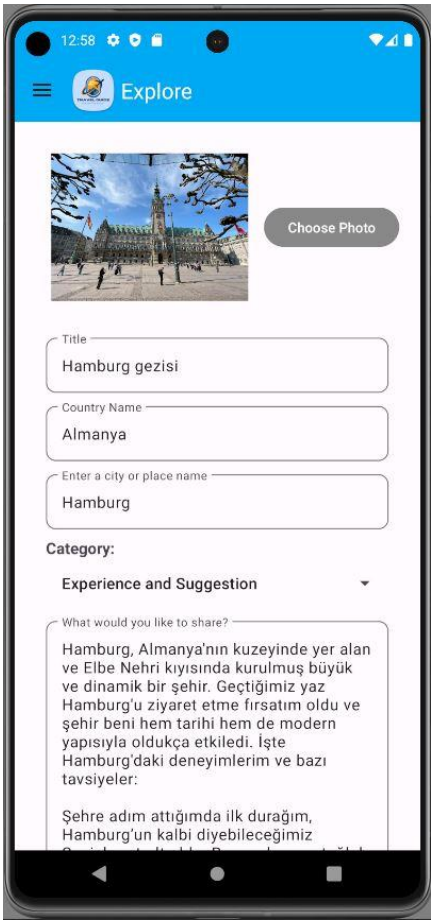
tıklayarak gönderisini başarıyla kaydedebilir. Bu şekilde, kullanıcılar kolayca gönderi paylaşabilir ve bilgilerini düzenleyebilir.

Şekil 6.8. Gönderi ekleme sayfa tasarımı

6.8. Gönderi Düzenleme Ekranı

Gönderi düzenleme ekranı, kullanıcıların paylaşmış oldukları gönderileri düzenlemek istedikleri takdirde kullanabilecekleri bir sayfadır. Kullanıcılar, kendi profil sayfalarından gönderilerin sağ tarafında bulunan menüden "Düzenle" seçeneğini seçerek bu sayfaya ulaşabilirler. Gönderi düzenleme ekranı, gönderi ekleme sayfası ile aynı tasarıma sahiptir. Bu sayfa açıldığında, gönderi ile ilgili bilgiler gösterilir ve bu bilgiler profil sayfasındaki seçilen gönderiden alınan verilerle doludur. Veriler, bundle ile bu fragmenta aktarılır, dolayısıyla veritabanından yeniden alınmaz ve sayfanın açılma süresi kısalır.

Sayfada kullanıcı, istediği bilgileri değiştirebilir. Yeni bir fotoğraf seçebilir, başlık, ülke adı, şehir veya mekan adı, kategori, paylaşmış olduğu yazı ve rating puanı gibi bilgileri güncelleyebilir. Kullanıcı, gerekli değişiklikleri yaptıktan sonra "Kaydet" butonuna tıkladığında değişiklikler kaydedilmiş olur. Bu sayede kullanıcılar, gönderilerini kolayca düzenleyebilir ve güncelleyebilir.



Şekil 6.9. Gönderi düzenleme sayfa tasarımı

6.9. Ülkeler Ekranı

Ülkeler ekranı, kullanıcıların seçtikleri ülke ile ilgili temel bilgileri ve seyahatlerinde ihtiyaç duyabilecekleri bilgileri göstermek amacıyla yapılmıştır. Bu sayfada, ülkeleri ve ülke bilgilerini internetten çekmek için RestCountries API kullanılmıştır. Bu API sayesinde tüm ülkelerin isimlerine ve bazı önemli bilgilere kolaylıkla ulaşılabilmektedir.

Sayfanın en üst kısmında bir spinner bulunmaktadır. Bu spinnerda API'den gelen ülke isimleri listelenir ve kullanıcı istediği ülkeyi kolaylıkla seçebilir. Ülke seçimi yapıldığında, seçilen ülkenin bayrağı, ülke ismi, başkent, dil, nüfus, alan, bölge ve para birimi bilgileri kullanıcıya gösterilir. Sayfada ayrıca bir "Google Haritalarda Göster" butonu yer almaktadır. Bu butona tıklandığında seçili ülke Google Haritalar'da açılabilir.

Sayfanın alt kısmında ise bir menü bulunmaktadır. Bu menüde, ülke hakkında bilgi edinebilmek için gerekli bazı kategoriler bulunmaktadır. Bu kategoriler arasında genel bakış, görülmesi gereken yerler, yemek kültürü, sanat ve tarih gibi toplam 12 kategori yer almaktadır. Kullanıcı, istediği kategoriyi seçerek seçtiği ülke ve kategori hakkında bilgi alabilir. Bu işlem Google Gemini API sayesinde yapılmaktadır. Seçilen ülke ve kategori bilgileri ile bir prompt oluşturularak Google Gemini API'ye gönderilir ve API buna göre bir cevap metni oluşturur. Bu metin, seçilen kategori butonuna tıklandığında açılan sayfada kullanıcıya gösterilir, böylece kullanıcı gerekli bilgiye kolaylıkla ulaşmış olur.

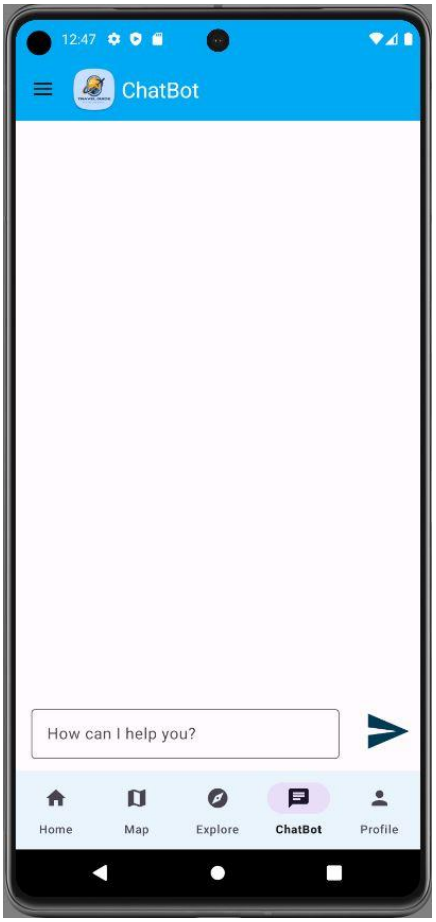
Şekil 6.10. Ülkeler sayfa tasarımı

6.10. ChatBot Ekranı

ChatBot ekranı, kullanıcıların ülkeler, seyahat ettikleri veya edecekleri yerler ya da herhangi bir konuda merak ettiklerini kolaylıkla ve hızlı bir şekilde öğrenmelerine olanak sağlamaktadır. Bu sayfa, Google Gemini API ile entegre edilerek oluşturulmuştur.

Sayfanın alt kısmında, "Nasıl yardımcı olabilirim?" yazısı bulunan bir text input alanı yer almaktadır. Kullanıcı, sayfada yer alan bu text input kısmına sormak istediği herhangi bir şeyi yazar ve sağ tarafta bulunan gönderme butonuna tıklayarak mesajını gönderir. Google Gemini API, bu mesajı alır ve buna uygun bir cevap oluşturarak geri gönderir. Bu cevap da ekranda gösterilir.

Kullanıcı hangi dilde bir mesaj girerse, cevap da o dilde gelmektedir. Sayfa, gerçekçi bir chat ekranı gibi tüm konuşmaları barındırır. Ancak sayfadan çıkıldıktan sonra bu konuşmalar silinir. Bu sayfa, uygulamanın içinde gerçekçi bir ChatBot sayfası olarak çalışmaktadır ve kullanıcılara kolaylık ve hız sağlamaktadır.



Şekil 6.11. Chatbot sayfa tasarımı

6.11. Harita Ekranı

Harita ekranı, kullanıcılara uygulamadan çıkmadan kolayca haritaya ulaşmaları, gezinebilmeleri, arama yapabilmeleri ve konum bulabilmeleri gibi özellikler için oluşturulmuştur. Bu harita sayfası, Google Maps API ile birlikte oluşturulmuştur. Ayrıca, yer arama özelliği için Google Places API de kullanılmıştır.

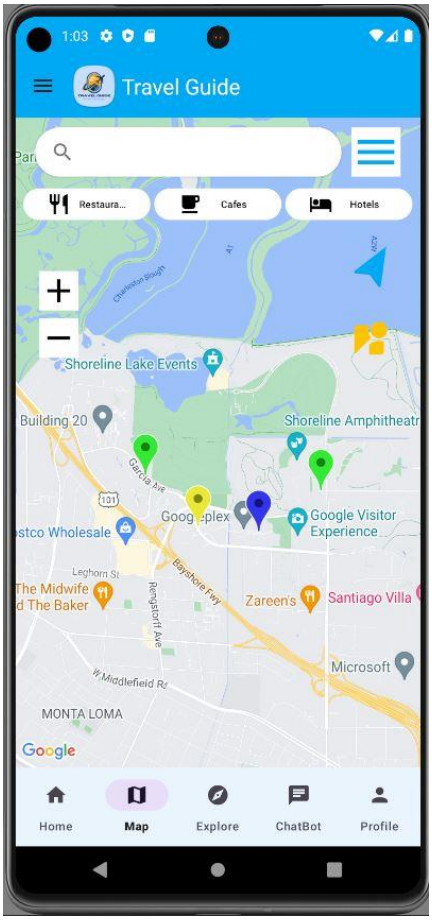
Sayfanın üst kısmında bir arama çubuğu bulunmakta ve kullanıcı buraya istediği bir yer adını girerek API sayesinde çıkan sonuçlardan seçimini yapabilir ve seçilen

bölgenin konumunu haritada görüntüleyebilir. Arama çubuğunun yan tarafında bulunan menüden harita görünümü değiştirilebilir. Menüde normal harita, hibrit harita, uydu haritası, arazi haritası gibi seçenekler bulunmaktadır.

Arama çubuğunun altında üç tane yanyana buton bulunmakta. Bu butonlar ile kullanıcı, konumuna göre yakınındaki restaurantlar, kafeler ve otelleri haritada görebilir. Sol tarafta bulunan artı ve eksi butonları ile harita yakınlaştırılıp uzaklaştırma işlemleri kolaylıkla yapılabilir. Sağ tarafta bulunan mavi navigasyon butonu sayesinde kullanıcı kendi konumunu görebilir. Bu butona tıklandığında önce kullanıcıdan konum izni istenir. Eğer daha önce bu izin verilmişse tekrar izin istenmez. Kullanıcı konum izni verdiğinde, harita kullanıcının konumuna zoom yapar.

Bu butonun hemen altındaki sarı sokak görünümü butonu ile birlikte kullanıcı seçeceği bir yerin sokak görünümüne ulaşabilir. Ancak, kullanıcı harita üzerinde bir nokta seçmemişse, bu kullanıcıya toast mesajı ile bilgilendirilir. Kullanıcı harita üzerinde bir yere dokunarak sarı bir işaret oluşturabilir. Bu işaret oluşturulduktan sonra sokak görünümü butonuna tıklandığında kullanıcı o noktanın sokak görünümüne ulaşabilir. Eğer seçilen noktada bir sokak görünümü mevcut değilse, açılan sayfa siyah ekran olarak kalacaktır.

Harita üzerindeki işaretler Hakkında Bilgi:
Mavi işaret: Kullanıcının konumunu gösterir.
Sarı işaret: Kullanıcı haritada bir noktaya tıkladığında çıkar ve sokak görünümü için kullanılır.
Yeşil işaret: Kullanıcı haritada bir noktaya uzun basılı tuttuğunda oluşur ve kullanıcının bir noktayı işaretlemesi için kullanılır. Bu nokta, kullanıcının istediği, seyahat etmek istediği veya beğendiği bir mekan gibi yerler olabilir. Bu işaret, kullanıcının telefonunda shared reference olarak saklanır ve haritada daima gözükür.
İşaretlerin üzerine tıklandığında işaret hakkında bir yazı gösterilir. Ayrıca, alt tarafta bir snackbar ile kullanıcıya bu işareti kaldırmak isteyip istemediği sorulur. Eğer kullanıcı "Evet" kısmına tıklarsa işaret haritadan silinir, aksi halde snackbar 5 saniye sonra yok olur.



Şekil 6.12. Harita sayfa tasarımı

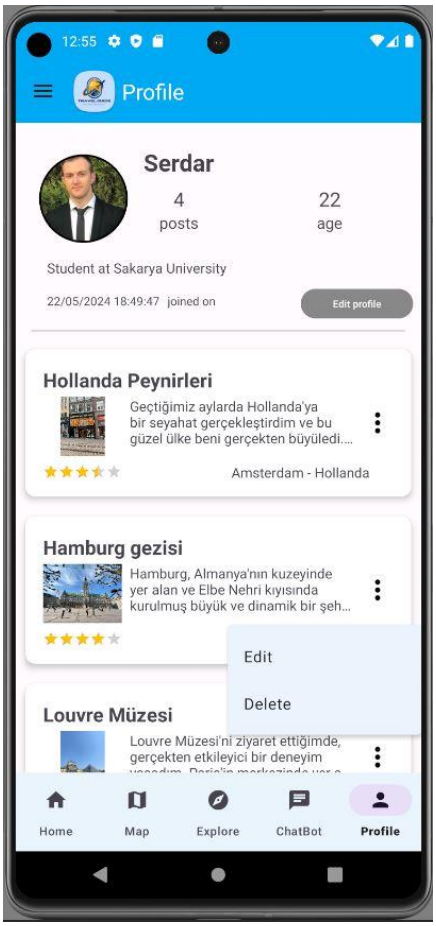
6.12. Profil Ekranı

Profil ekranı, kullanıcının profilini görüntüleyebilmesini sağlar. Bu sayfada kullanıcı profil bilgilerini ve eklemiş olduğu gönderileri görebilir.

Ekranın üst kısmında kullanıcının profil resmi ve ismi bulunur. Kullanıcının isminin altında, paylaştığı gönderi sayısını gösteren bir bölüm ve kullanıcının yaşını gösteren bir alan bulunur. Bu bilgilerin altında, kullanıcının biyografisini gösteren bir kısım ve profilin oluşturulduğu tarihi gösteren bir alan yer alır. Profil oluşturulduğu tarihin sağ tarafında "Profili Düzenle" isimli bir buton bulunur. Kullanıcı bu butonu kullanarak profilini düzenleme ekranına gidebilir.

Bu bilgilerin altında bir çizgi bulunur ve profil bilgileri bölümünü gönderilerin gösterildiği bölümden ayırır. Bunun altında bir RecyclerView bulunur ve burada

kullanıcının paylaşmış olduğu gönderiler listelenir. Bu gönderiler kullanıcının mail hesabına göre veritabanından çekilmektedir. Gönderilerin üzerine tıklandığında gönderi detay ekranı açılır ve gönderi burada görüntülenir. Ayrıca, gönderilerin sağ tarafında üç nokta menü çubuğu bulunur. Kullanıcı bu butona tıkladığında bir popup menü açılır ve "Sil" ve "Düzenle" seçenekleri bulunur. Sil seçeneği tıklanıldığında sayfanın altında bir snackbar açılır ve "Silmek istediğinize emin misiniz?" şeklinde bir mesaj gösterilir. "Evet" seçeneği seçilirse silme işlemi gerçekleşir. Bir seçim yapılmazsa, snackbar 5 saniye sonra kaybolur. Düzenle seçeneği seçildiğinde ise gönderi düzenleme sayfasına gidilir.



Şekil 6.13. Profil sayfa tasarımı

6.13. Profil Düzenleme Ekranı

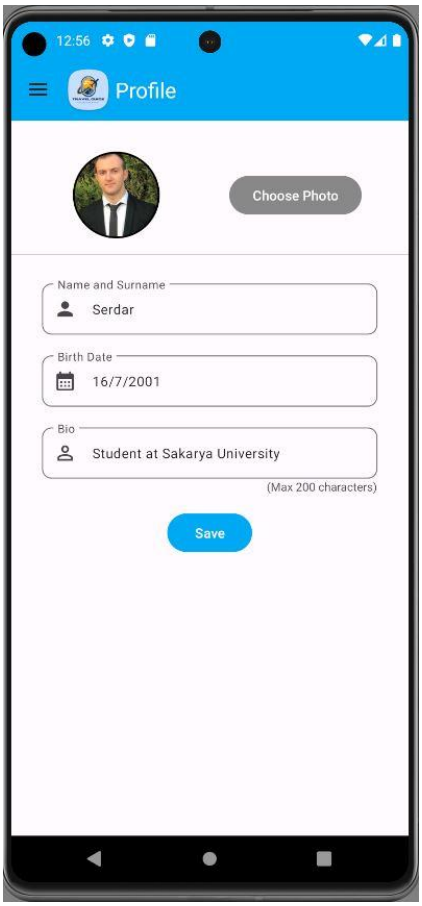
Profil düzenleme ekranı, kullanıcının profilini düzenlemesine olanak sağlar. Burada kullanıcı profil bilgilerini değiştirebilir. Profil ekranında "Profil Düzenle" butonuna

tıklandığında bu ekran açılır. Sayfa açıldığında kullanıcının tüm profil bilgileri sayfada gösterilir ve bu bilgiler profil sayfasından bundle ile alınmaktadır.

Bu sayfanın üst kısmında profil resmi gösterilir ve yanında "Fotoğraf Seç" yazılı bir buton bulunur. Kullanıcı bu butona tıklayarak cihazından bir fotoğraf ekleyebilir. Fotoğraf seçme işlemi tamamlandığında profil resmi güncellenir.

Profil düzenleme ekranının alt kısmında isim, doğum tarihi ve biyografi için text input alanları bulunur. Bu alanlar kullanılarak değişiklik yapılabilir. Biyografi alanı için en fazla 200 karakterlik bir sınırlama bulunmaktadır. Bu sınırlama, kullanıcıya biyografi text inputunun altında bir metin olarak gösterilir.

Bunların alt kısmında "Kaydet" yazılı bir buton bulunur. Değişiklikler yapıldıktan sonra kullanıcı bu butona basarak değişikliklerin başarıyla kaydedilmesini sağlayabilir ve profil ekranına geri dönebilir.



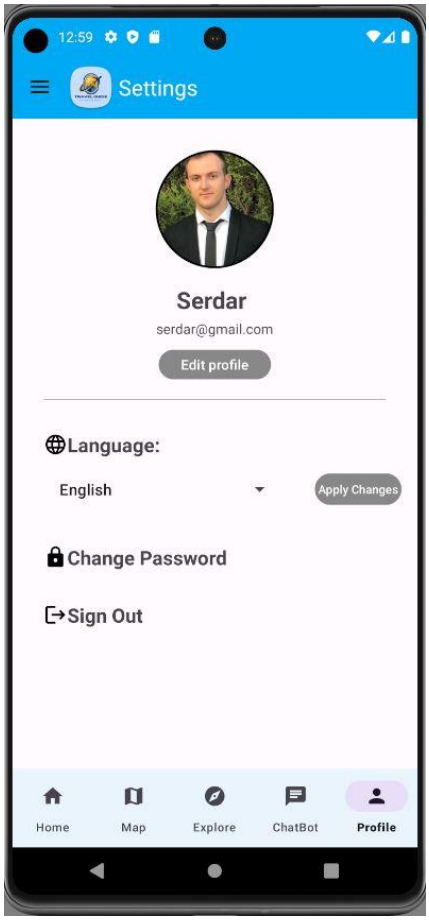
Şekil 6.14. Profil düzenleme sayfa tasarımı

6.14. Ayarlar Ekranı

Ayarlar ekranı, kullanıcının uygulama içerisinde bazı ayarları yapmasına olanak tanır. Bu sayfanın üst tarafında kullanıcının profil resmi, kullanıcı adı ve kullanıcı mail hesabı görüntülenir. Bunların altında "Profili Düzenle" yazılı bir buton bulunur ve bu butona tıklandığında profil sayfasına yönlendirilir.

Profil kısmının altında bir çizgi bulunur ve diğer ayarlar ile ayrılır. Bu çizginin altında "Dil" kısmı bulunur. Burada bir spinner bulunur ve spinner menü içerisinde 5 farklı dil seçeneği bulunur. Kullanıcı istediği dili seçerek sağ tarafta bulunan "Değişiklikleri Uygula" butonuna tıkladığında seçilen dil uygulamaya uygulanır ve ana sayfa açılır.
Dil seçimi kısmının altında "Şifre Değiştir" kısmı bulunur. Bu yazıya tıklandığında şifre değişikliği yapılabilmesi için bir ekran açılır. Bu ekranda kullanıcı mail hesabını text input içerisine yazar ve "Gönder" yazılı butona tıklayarak işlemi gerçekleştirir. Daha sonra mailine gelen mail ile şifre değişikliğini kolayca yapabilir.

Ayarlar ekranının en altında "Çıkış Yap" yazısı bulunur. Kullanıcı bu yazıya tıkladığında çıkış yapma ekranı açılır ve kullanıcı uygulamadan çıkış yapmış olur.



Şekil 6.15. Ayarlar sayfa tasarımı

6.15. Drawer Menu (Çekmece Menü) ve Bottom Navigation View (Alt Gezinim Görünümü)

Uygulamada hem sol tarafta bulunan "Drawer Menu" hem de alt tarafta konumlandırılmış "Bottom Navigation Bar" sayesinde kullanıcılar, istedikleri sayfalara hızlı ve kolay bir şekilde erişebilirler. Bu iki menü, kullanıcı dostu bir deneyim sunarak sayfa geçişlerini etkili bir şekilde sağlar.

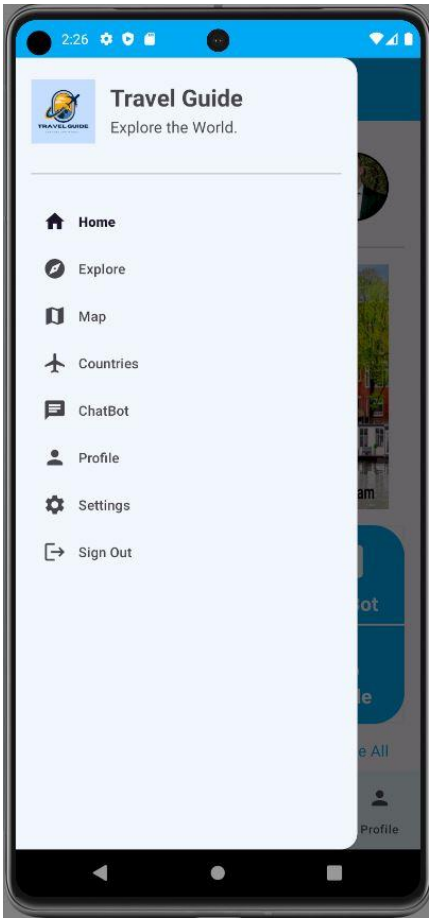
Drawer Menu:

Üst tarafta yer alan toolbar üzerindeki üç çizgi simgesine tıklayarak açılan Drawer Menu, geniş bir yelpazede seçenekler sunar. Kullanıcılar, ana sayfa, keşfet, harita, ülkeler, ChatBot, profil, ayarlar ve çıkış yap gibi seçenekleri görüntüleyebilir ve istedikleri sayfaya kolayca geçiş yapabilirler. Bu menü, uygulamanın tüm özelliklerine hızlı bir şekilde erişim sağlar.

Bottom Navigation Bar:

Alt kısımda bulunan Bottom Navigation Bar ise en sık kullanılan sayfaları içerir. Anasayfa, harita, keşfet, ChatBot ve profil gibi seçenekler, kullanıcıların hızlı bir şekilde istedikleri sayfaya geçiş yapmasını sağlar. Bu navigasyon barı, uygulamanın kullanımını daha da kolaylaştırarak, kullanıcıların en çok ihtiyaç duyduğu yerlere hızlı bir şekilde erişmelerini sağlar.

Bu düzenleme, kullanıcıların uygulamayı daha kolay ve etkili bir şekilde kullanmalarını sağlayarak, kullanıcı deneyimini önemli ölçüde arttırır. Hem Drawer Menu hem de Bottom Navigation Bar, kullanıcıların istedikleri sayfaya rahatlıkla geçiş yapmalarını sağlayarak, uygulamanın kullanımını daha keyifli hale getirir.



Şekil 6.16. Drawer menu (çekmece menu) tasarımı

BÖLÜM 7. SONUÇLAR VE ÖNERİLER

TravelGuide uygulaması, modern seyahat deneyimlerinin gereksinimlerini karşılamak üzere tasarlanmış bir mobil uygulamadır. Teknolojinin sunduğu imkanlarla birlikte, seyahat etme alışkanlıklarımızın hızla değiştiği bir dönemde, TravelGuide kullanıcıların seyahat deneyimlerini kolaylaştırmak ve zenginleştirmek için geliştirilmiştir.

Günümüzde seyahat etmek artık eskisi kadar zorlu değil. Eskiden haritalar ve yerel halkın yardımıyla yol bulmak gerekiyordu, ancak şimdi internet ve mobil uygulamalar sayesinde seyahat süreçleri çok daha kolay ve erişilebilir hale geldi. TravelGuide, bu gelişmeleri takip ederek, kullanıcıların seyahat öncesi planlamadan seyahat sırasında ihtiyaç duydukları bilgilere kadar her şeye erişimini kolaylaştırmak için tasarlanmıştır.

Mobil uygulamaların sunduğu geniş olanaklar sayesinde, seyahat deneyimleri daha planlı, konforlu ve bilgi dolu hale gelmiştir. TravelGuide, harita entegrasyonundan çeviri araçlarına kadar birçok özelliğiyle kullanıcıların seyahatlerini yönetmelerine yardımcı olurken, yapay zeka destekli chatbot ile de her zaman kullanıcıların yanında bulunmaktadır.

Projemizin amacı, seyahat etmeyi sevenlerin ihtiyaçlarına cevap vermek ve onların seyahat deneyimlerini en üst düzeye çıkarmaktır. TravelGuide, kullanıcı dostu arayüzü, kapsamlı özellikleri ve pratik kullanımıyla kullanıcıların vazgeçilmez bir seyahat rehberi olmayı hedeflemektedir.

Sonuç olarak, TravelGuide, teknolojinin sunduğu imkanlardan en iyi şekilde faydalanarak, seyahat etmeyi herkes için daha erişilebilir, keyifli ve bilgi dolu hale getirmeyi amaçlamaktadır.

KAYNAKLAR

[1] WANG, H. Y., LIAO, C., & YANG, L. H. (2013). What affects mobile application use? The roles of consumption values. International Journal of Marketing Studies, 5(2), 11.

[2] Ercan, F. (2021). Mobil Seyahat Uygulamaları ve Karadeniz Ereğli İçin Bir Mobil Gezi Rehberi Uygulama Önerisi: KdzEregli Travel Guide (KTG). International Journal of Contemporary Tourism Research, 5(1), 1-12.

[3] Vardi, M. Y. (2012). Artificial intelligence: past and future. Communications of the ACM, 55(1), 5-5.

[4] ALLIANCE, OPEN HANDSET. "Android (operating system)." Marketing 4.5 (2013).

[5] ALJAMEA, MARIAM, and MOHAMMAD ALKANDARI. "MMVMi: A validation model for MVC and MVVM design patterns in iOS applications." IAENG Int. J. Comput. Sci 45.3 (2018): 377-389.

[6] WU, WEI-LING, ET AL. "A Review of Apps for Programming: programming languages and making apps with apps." Scientific Phone Apps and Mobile Devices (2019).

[7] TOSUNOĞLU, E., & USTUN, A. B. (2021). Xamarin Çapraz-Platformu ile Gerçek Zamanlı Bulut Veri Tabanı iletişimi: Bütünleşik Akıllı Ev Sistemi. Avrupa Bilim Ve Teknoloji Dergisi(27), 658-664.

[8] PETERSON, KEVIN. "The github open source development process." url: http://kevinp. me/github-process-research/github-processresearch. pdf (visited on 05/11/2017) (2013).

ÖZGEÇMİŞ

Serdar Arıcı, 10.07.2001 de Sakarya'da doğdu. İlk, orta ve lise eğitimini Sakarya'da tamamladı. 2019 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'nü kazandı. 2023 yılında Mitrona Elektronik ve Yazılım Teknolojileri A.Ş. Şirketinde donanım stajını yapmıştır. Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü 4. Sınıf öğrencisidir.

BSM 498 BİTİRME ÇALIŞMASI DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI

KONU : Yapay Zeka Destekli Seyahat Rehberi Mobil Uygulaması
ÖĞRENCİLER (Öğrenci No/AD/SOYAD): G191210020/SERDAR/ARICI

Değerlendirme Konusu	İstenenler	Not Aralığı	Not
Yazılı Çalışma			
Çalışma klavuza uygun olarak hazırlanmış mı?	x	0-5	
Teknik Yönden			
Problemin tanımı yapılmış mı?	x	0-5	
Geliştirilecek yazılımın/donanımın mimarisini içeren blok şeması (yazılımlar için veri akış şeması (dfd) da olabilir) çizilerek açıklanmış mı?			
Blok şemadaki birimler arasındaki bilgi akışına ait model/gösterim var mı?			
Yazılımın gereksinim listesi oluşturulmuş mu?			
Kullanılan/kullanılması düşünülen araçlar/teknolojiler anlatılmış mı?			
Donanımların programlanması/konfigürasyonu için yazılım gereksinimleri belirtilmiş mi?			
UML ile modelleme yapılmış mı?			
Veritabanları kullanılmış ise kavramsal model çıkarılmış mı? (Varlık ilişki modeli, noSQL kavramsal modelleri v.b.)			
Projeye yönelik iş-zaman çizelgesi çıkarılarak maliyet analizi yapılmış mı?			
Donanım bileşenlerinin maliyet analizi (prototip-adetli seri üretim vb.) çıkarılmış mı?			
Donanım için gerekli enerji analizi (minimum-uyku-aktif-maksimum) yapılmış mı?			
Grup çalışmalarında grup üyelerinin görev tanımları verilmiş mi (iş-zaman çizelgesinde belirtilebilir)?			
Sürüm denetim sistemi (Version Control System; Git, Subversion v.s.) kullanılmış mı?			
Sistemin genel testi için uygulanan metotlar ve iyileştirme süreçlerinin dökümü verilmiş mi?			
Yazılımın sızma testi yapılmış mı?			
Performans testi yapılmış mı?			
Tasarımın uygulamasında ortaya çıkan uyumsuzluklar ve aksaklıklar belirtilerek çözüm yöntemleri tartışılmış mı?			
Yapılan işlerin zorluk derecesi?	x	0-25	
Sözlü Sınav			
Yapılan sunum başarılı mı?	x	0-5	
Soruları yanıtlama yetkinliği?	x	0-20	
Devam Durumu			
Öğrenci dönem içerisindeki raporlarını düzenli olarak hazırladı mı?	x	0-5	
Diğer Maddeler			
Toplam			

DANIŞMAN (JÜRİ ADINA): PROF. DR. NEJAT YUMUŞAK
DANIŞMAN İMZASI: