T.C.
ERCİYES ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

BERBER RANDEVU SİSTEMİ UYGULAMASI

Hazırlayan

Burak Uncel
1030510157
Furkan Kalender
1030510199

Danışman
Dr. Öğr. Üyesi Fehim KÖYLÜ

Bilgisayar Mühendisliği
Bitirme Ödevi

HAZİRAN 2024
KAYSERİ

"Berber Randevu Sistemi Uygulaması" adlı bu çalışma, jürimiz tarafından Erciyes Üniversitesi Bilgisayar Mühendisliği Bölümünde Bitirme Ödevi olarak kabul edilmiştir.

14/06/2024

JÜRİ :

Danışman : Dr. Öğr. Üyesi Fehim KÖYLÜ

Üye :

Üye :

ONAY :

Yukarıdaki imzaların, adı geçen öğretim elemanlarına ait olduğunu onaylarım.

.... / .... /20...
Prof. Dr. Veysel ASLANTAŞ
Bilgisayar Müh. Bölüm
Başkanı

ÖNSÖZ / TEŞEKKÜR

Bu tez, günümüzün önemli esnaflarından olan berberlerle daha sağlıklı bir iletişim kurmayı ve hem müşterilerin hem de berberlerin daha yüksek verim almasını amaçlamaktadır. Tezi oluştururken, karşılaştığımız zorlukları ve çabaları sürekli bir öğrenme ve gelişim fırsatı olarak ele aldık. Bu çalışma, bilgiye aç olan herkese, özellikle mobil uygulama alanına ilgi duyan araştırmacılara, bu alandaki bilgi birikimini artırmak ve ileriye taşımak için bir kaynak sunmayı amaçlamaktadır. İlk olarak, bu projenin başından sonuna kadar beni yönlendiren danışmanım Sayın Dr. Öğr. Üyesi Fehim KÖYLÜ hocama içtenlikle teşekkür ederiz. Değerli bilgileri, deneyimleri ve yönlendirmeleri sayesinde bu çalışma daha anlamlı hale geldi.

BERBER RANDEVU SİSTEMİ UYGULAMASI

Burak Uncel
Erciyes Üniversitesi, Bilgisayar Mühendisliği
Danışman : Dr. Öğr. Üyesi Fehim KÖYLÜ

ÖZET

Bu çalışma, Swift programlama dili ve SwiftUI kullanılarak bir berber randevu sistemi geliştirmeyi amaçlamaktadır. Berberlerin randevularını ve müşterilerin randevu almasını kolaylaştırmak için bir mobil uygulama tasarlanmıştır. Bu proje Swift UI'ın güçlü Frontend kasını kullanarak, iOS cihazlar için hızlı, optimize edilmiş ve cross-platform'a göre çok daha güvenilir bir şekilde compile ederek, kullanıcıya güzel bir deneyim yaşatacaktır. Cihaz lokal depolama teknolojileri ile kullanıcı bilgilerini kaydederek, kullanıcının tekrar tekrar giriş yaparak yorulmasını engellemektedir. Firebase teknolojisi kullanarak, kullanıcı verilerinin güvenli ve optimize bir şekilde depolanmasını, daha sonrasında ise diğer kullanıcılar ile rahat bir şekilde paylaşmasına basamak oluşturmaktadır. Çalışmamızın temel amacı önemli esnaflarımızdan olan berberler ile müşterilerin kordine bir şekilde maksimum verim alması ve zaman kazanmasıdır.

Anahtar Kelimeler: Swift, Randevu sistemi, Firebase, Lokal depolama.

BARBER APPOINTMENT SYSTEM APPLICATION

Burak Uncel
Erciyes University, Computer Engineering
Supervisor: (Associate Dr.) Fehim KÖYLÜ

ABSTRACT

This study aims to develop a barber appointment system using Swift programming language and SwiftUI. A mobile application has been designed to facilitate barbers' appointments and customers' appointment bookings. By utilizing the powerful Frontend capabilities of Swift UI, the project compiles quickly and optimally for iOS devices, providing users with a beautiful experience. Storing user data locally on the device prevents users from repeatedly logging in, thus reducing user fatigue. By using Firebase technology, user data is securely and efficiently stored, enabling easy sharing with other users later on. The primary goal of our project is to maximize efficiency and time savings for both barbers, who are important members of our community, and their customers.
Keywords: Swift, Appointment system, Firebase, Local storage.

İÇİNDEKİLER

BERBER RANDEVU SİSTEMİ UYGULAMASI

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

KISALTMALAR

GPS	:	Global Positioning System
CPU	:	Central Process Unit
iOS	:	iPhone Operating System
SDK	:	Software Developement Kit
API	:	Application Programming Interface
FCM	:	FireBase Cloud Messaging
$HTTPS$	:	Secure Hyper Text Transfer Protocol

GİRİŞ

Günümüzde teknolojinin hızlı bir şekilde gelişmesi, geleneksel işletmelerin de dijital dönüşümüne neden olmuştur. Berber dükkanları da bu dönüşümden nasibini almış ve teknolojinin sunduğu fırsatlardan yararlanmaya başlamıştır. Geleneksel randevu sistemlerinin yerini artık dijital berber randevu sistemleri almaktadır. Bu sistemler, berberlerin randevularını yönetmelerini ve müşterilerin kolayca randevu alabilmelerini sağlamaktadır.

Bu çalışma, SWİFT programlama dilini ve SwiftUI framework'ünü kullanarak bir berber randevu sistemi geliştirmeyi amaçlamaktadır. SWİFT, özellikle iOS ve macOS uygulamaları için geliştirilmiş güçlü ve kullanıcı dostu bir programlama dilidir. SwiftUI ise, Apple'ın kullanıcı arayüzü geliştirme kitaplığıdır ve deklaratif bir yaklaşım sunarak arayüz tasarımını daha basit ve verimli hale getirir.

Projenin temel hedefi, berberlerin randevu yönetimini kolaylaştırmak ve müşterilerin randevu alma sürecini daha kullanıcı dostu hale getirmektir. Bu amaç doğrultusunda, müşterilerin mobil uygulama üzerinden kolayca randevu alabilmesi ve berberlerin randevu takvimlerini yönetebilmesi sağlanacaktır. Ayrıca, Firebase gibi platformlar kullanılarak verilerin güvenli bir şekilde depolanması ve kullanıcıların verimli bir şekilde yönetilmesi sağlanacaktır.

SWİFT UI ile geliştirilen bu berber randevu sistemi projesi, geleneksel randevu sistemlerine kıyasla daha hızlı, daha kullanıcı dostu ve daha güvenilir bir deneyim sunmayı hedeflemektedir. Bu sayede, hem berberlerin iş süreçlerini iyileştirmek hem de müşterilere daha iyi bir hizmet sunmak mümkün olacaktır. //

Literatür Özeti

Berberlik sektörü, insanların kişisel bakım ihtiyaçlarını karşılamak üzere uzun bir

geçmişe sahip olan önemli bir sektördür. Ancak, bu sektör, teknolojinin hızla değişen doğası ve müşteri beklentilerindeki değişikliklerle birlikte sürekli olarak dönüşmektedir. Geleneksel berber dükkanları, müşterilerin randevu alma ve bekleme sürelerini azaltmak, randevu yönetimini kolaylaştırmak ve genel müşteri deneyimini iyileştirmek için yeni ve yenilikçi çözümlere ihtiyaç duymaktadır.

Son yıllarda, dijital berber randevu sistemleri bu ihtiyacı karşılamak üzere popüler hale gelmiştir. Bu sistemler, berberlerin randevu takvimlerini dijital ortama taşıyarak randevu yönetimini basitleştirirken, müşterilere de randevu alma ve yönetme süreçlerini kolaylaştırmaktadır. Mobil uygulamalar aracılığıyla randevu almak ve randevu tarihlerini değiştirmek gibi işlemler artık daha hızlı ve daha kolaydır.

SWİFT programlama dili ve SwiftUI framework'ü, bu tür dijital berber randevu sistemlerinin geliştirilmesinde sıklıkla kullanılan araçlardır. Bu teknolojiler, gelişmiş performans, kullanıcı dostu arayüzler ve kolay entegrasyon gibi avantajlar sunar. Ayrıca, Firebase gibi bulut tabanlı veritabanı hizmetleri kullanılarak verilerin güvenli bir şekilde saklanması ve senkronize edilmesi sağlanır.

Berberlik sektöründeki dijital dönüşüm, SWİFT ve SwiftUI gibi modern teknolojilerin kullanımıyla birlikte giderek hız kazanmaktadır. Bu teknolojiler, geleneksel berber dükkanlarının rekabet gücünü artırmak ve müşteri memnuniyetini artırmak için önemli bir rol oynamaktadır. Ayrıca, bu tür dijital çözümlerin kullanılması, berberlerin iş süreçlerini optimize etmelerine ve daha verimli hale getirmelerine yardımcı olur.

Sonuç olarak, SWİFT ve SwiftUI gibi modern teknolojilerin kullanımıyla geliştirilen dijital berber randevu sistemleri, sektördeki geleneksel uygulamalara kıyasla önemli avantajlar sunmaktadır. Bu sistemler, berberlerin işlerini daha verimli bir şekilde yönetmelerine ve müşterilere daha iyi bir hizmet sunmalarına yardımcı olacaktır.

Çalışmanın Amacı

Bu çalışmasının amaçları şu şekilde sıralanabilir:

Bu çalışma, geleneksel berberlik sektörünü dijitalleştirme çabalarının bir parçası

olarak SWİFT programlama dili ve SwiftUI framework'ünü kullanarak bir berber randevu sistemi geliştirmeyi amaçlamaktadır. Berberlik sektörü, yüzyıllardır insanların kişisel bakım ihtiyaçlarını karşılamak için önemli bir rol oynamıştır. Ancak, teknolojinin hızlı bir şekilde ilerlemesiyle birlikte, geleneksel randevu sistemleri ve bekleme uygulamaları artık yetersiz kalmaktadır. Bu nedenle, bu çalışma, berberlerin iş süreçlerini modernize etmek ve müşterilere daha iyi bir deneyim sunmak için yeni bir yaklaşım sunmayı hedeflemektedir. Bu çalışmanın öncelikli amacı, berberlerin randevu yönetimini kolaylaştırmak ve müşterilere daha iyi bir hizmet sunmak için bir mobil uygulama geliştirmektir. SWİFT ve SwiftUI gibi modern teknolojilerin kullanımıyla geliştirilen bu mobil uygulama, berberlerin randevu takvimlerini dijital ortama taşıyarak randevu süreçlerini optimize etmeyi amaçlamaktadır. Ayrıca, müşterilerin kolayca randevu alabilmesi, randevu tarihlerini değiştirebilmesi ve berberlerle iletişim kurabilmesi için kullanıcı dostu bir arayüz sunmaktadır. Bu çalışmanın ikinci amacı, geliştirilen berber randevu sisteminin kullanıcıların gereksinimlerini karşılamasını sağlamaktır. Bu nedenle, kullanıcı ara yüzü tasarımı ve kullanılabilirliği büyük önem taşımaktadır. Kullanıcıların rahatlıkla randevu alabilmesi ve takvimlerini yönetebilmesi için uygulamanın sezgisel bir şekilde tasarlanması gerekmektedir. Son olarak, bu çalışma, berberlik sektöründe dijital dönüşümün önemini vurgulamak ve bu dönüşümü hızlandırmak için bir model oluşturmayı amaçlamaktadır. Geliştirilen berber randevu sistemi, berberlerin iş süreçlerini optimize etmelerine ve müşterilere daha iyi bir hizmet sunmalarına yardımcı olacak potansiyele sahiptir. Ayrıca, sektördeki diğer işletmelere benzer dijital çözümler geliştirme konusunda ilham verebilir.

Tezin Organizasyonu

Bu çalışmanın düzenlemesi şu şekildedir.

1.Bölümde, çalışmanın genel amacı ve kapsamı belirtilmiştir.Mobil teknolojilerinin özelliklerine genel bir giriş yapılmıştır. Araştırmanın motivasyonu ve hedefleri burada açıklanmıştır.

2.Bölümde, SwiftUi, Lokal depolama ve Firebase'e üzerine odaklanılmıştır. Bu ana

başlıkların detaylı açıklaması bu bölümde açıklanmıştır. Ayrıca uygulamamızın detaylı bir şekilde açılaması bu bölümde ele alınmıştır.

3.Bölümde, çalışmanın genel bulgularını ve elde edilen sonuçları özetlenmiştir. Bulguların literatürle ilişkilendirilmesi, çalışmanın katkıları ve gelecekteki araştırmalara yönelik öneriler bu bölümde yer almaktadır

1. BÖLÜM

MOBİL CİHAZLARDA YAZILIM PLATFORMLARI

Mobil cihazlar günümüzde en çok kullandığımız teknolojidir. Bundan dolayı yapacağımız yeni projelerin özellikle mobil tarafında çalışması çok önemlidir. Hem kullanıcı arayüzü olarak hem de kullanıcının deneyimi olarak web uygulamalarındansa mobil uygulamalar daha çok tercih edilmektedir. Bundan dolayı günümüzde mobil uygulamalar çıkarmak çok önemlidir. Projemiz her ne kadar berber randevu sistemi olsa da aslında buradaki ana hedefimiz mobil kitleyi yakalayarak onlara güzel bir tecrübe yaşatmaktır.

1.1. Native Teknolojiler

Mobil teknolojinin hızlı evrimi, günlük yaşantımızda mobil uygulamaların rolünü önemli ölçüde artırmıştır. Bu talebin artmasıyla birlikte, geliştiriciler daha etkili ve verimli uygulamalar oluşturma yollarını aramışlardır. Bu noktada, native mobil uygulamalar ön plana çıkan güçlü çözümlerden biri olmuştur. Bu uygulamalar, mükemmel performans ve üstün kullanıcı deneyimi sunarak mobil dünyada kendilerine özgü bir yer edinmişlerdir.

Native mobil uygulamalar, belirli bir işletim sistemi için özel olarak geliştirilen yazılımlardır. iOS için Objective-C veya Swift, Android için ise Java veya Kotlin gibi platforma özgü programlama dilleri kullanılarak oluşturulurlar. Bu uygulamalar, cihazın donanımına doğrudan erişebilir ve işletim sisteminin sunduğu tüm özellikleri en iyi şekilde kullanabilirler. Bu erişim, uygulamaların daha hızlı çalışmasını, daha iyi bir kullanıcı deneyimi sunmasını ve cihazın yerel özelliklerinden, örneğin kamerası, mikrofonu veya GPS'si gibi özelliklerden tam olarak yararlanmasını sağlar.

1.1.1. Avantajları

1.1.1.1. Yüksek Performans

Native mobil uygulamalar, işletim sistemiyle doğrudan etkileşimde bulundukları için daha hızlı ve verimli bir performans sunarlar.

1.1.1.2. Daha İyi Kullanıcı Deneyimi

Kullanıcı arayüzü, işletim sisteminin yerel kullanıcı arayüzü bileşenlerini kullanarak oluşturulur; bu da kullanıcılara tanıdık bir deneyim sunar.

1.1.1.3. Yüksek Performans

Cihazın yerel özelliklerine tam erişim, geliştiricilere daha kapsamlı ve zengin özellikler sunma olanağı tanır.

1.1.2. Mobil Uygulama Geliştirme Süreci

Mobil uygulama geliştirme süreci, bir fikrin somut bir uygulamaya dönüştürülmesinden, nihai kullanıcılar tarafından kullanılabilir hale getirilmesine kadar uzanan kapsamlı bir serüvendir. Bu süreç, fikrin ilk ortaya çıkışından testlerin ve dağıtımın gerçekleştirildiği son aşamalara kadar çeşitli aşamalardan oluşur.

Native mobil uygulamalar geliştirmek için öncelikle kendi kimliğinizi belirlemeniz gerekiyor. Örnek vermek gerekirse; Siz Turkcell TV+ mobil uygulamasını geliştiriyorsunuz ve Iphone 13 piyasaya sürülürken cihaza yeni özellikler eklenmiş. Ekran boyutu değişmiş, ekrana bir çentik atılmış veya kaldırılmış, GPS'e algılama sistemi değişmiş vs. bu gibi durumlarda geliştirilen uygulamanız için kritik bir öneme sahip ise cihazda gerçekleştirilen güncellemelerin uygulamanızdaki etkisini anında verebilmeniz gerekir. [1]

1.1.2.1. Fikir ve Araştırma

Her şey bir fikirle başlar. Bu fikrin pazarda ne kadar geçerli olduğunu anlamak için kapsamlı bir pazar ve hedef kitle araştırması yapılması gerekir.

1.1.2.2. Fikir ve Araştırma

1.1.2.3. Planlama

Projenin kapsamını tanımlama, gerekli teknoloji ve araçları seçme ve bir zaman çizelgesi oluşturma aşaması, projenin temel altyapısını oluşturur.

1.1.2.4. Tasarım

Bu evre, kullanıcı arayüzü ve kullanıcı deneyiminin (UX) tasarımını içerir. Prototipler ve mockup'lar bu aşamada hazırlanır.

1.1.2.5. Geliştirme

Kodlama aşaması, native mobil uygulama geliştirme sürecinin temel adımlarından biridir. Bu aşamada, platforma özgü dil ve araçlar kullanılarak uygulamanın geliştirilmesi gerçekleştirilir. Ön ve arka uç geliştirme bu aşamanın önemli parçalarını oluşturur.

1.1.2.6. Test

Uygulama, farklı test aşamalarından geçirilir: birim testleri, entegrasyon testleri, performans testleri ve kullanılabilirlik testleri. Bu testler, uygulamanın beklenen performansı sergileyip sergilemediğini kontrol etmek ve sorunsuz bir şekilde çalıştığından emin olmak için son derece önemlidir.

1.1.2.7. Dağıtım ve Bakım

Uygulama, Google Play Store veya Apple App Store gibi platformlarda yayınlanır. Yayınlandıktan sonra, uygulama düzenli olarak güncellemeler ve bakım gerektirir. Bu güncellemeler, uygulamanın performansını ve güvenliğini sağlamak, yeni

özellikler eklemek ve kullanıcı geri bildirimlerine dayalı iyileştirmeler yapmak için önemlidir.

Mobil uygulama geliştirme süreci, titiz planlama ve detaylı çalışma gerektiren karmaşık bir süreçtir. Native mobil uygulamaların geliştirilmesi, bu sürecin özel bir parçasıdır ve uygulamanın başarısında kritik bir rol oynar. Bu süreç, uygulamanın performansından kullanıcı deneyimine kadar birçok faktörü doğrudan etkiler.

1.1.3. Uygulama Performansının Önemi

Uygulama performansı, kullanıcıların bir mobil uygulama ile etkileşim kalitesini doğrudan etkileyen kritik bir faktördür. Yüksek performanslı bir uygulama, hızlı yanıt süreleri ve sorunsuz bir kullanıcı deneyimi sunar, bu da kullanıcı memnuniyetini artırır ve uygulamanın başarısını artırır. Özellikle, native mobil uygulamalar, donanım kaynaklarına doğrudan erişim sayesinde üstün performans sergiler.

1.1.3.1. Anında Tepki

Native mobil uygulamalar, kullanıcıların eylemlerine anında yanıt verir. Bu özellik, özellikle oyunlar ve yoğun etkileşimli uygulamalar için büyük önem taşır.

1.1.3.2. Düşük Gecikme Süreleri

Veri işleme ve ekran yenilemeleri, native uygulamalarda daha hızlı gerçekleşir, bu da kullanıcıların beklemesini minimuma indirir.

1.1.3.3. Kaynak Optimizasyonu

Native uygulamalar, cihazın CPU ve belleğini daha verimli kullanır, böylece daha az pil tüketimi ve daha iyi bir genel performans sağlar. Bu avantajlar, native mobil uygulamaların kullanıcılar tarafından tercih edilme nedenlerinden bazılarıdır ve uygulama geliştiriciler için performansın neden öncelikli olduğunu

açıklar niteliktedir.

1.1.4. Kullanıcı Deneyimi ve Tasarım

Kullanıcı deneyimi, bir uygulamanın başarısında kritik bir rol oynar. Üstün bir kullanıcı deneyimi ve tasarım, kullanıcıların uygulamayı kolayca kullanmasına ve anlamasına yardımcı olur. Native mobil uygulamalar, platforma özgü olarak optimize edilmiş kullanıcı arayüzleriyle bu deneyimi sunma konusunda benzersiz avantajlara sahiptir.

1.1.4.1. Doğal Arayüz Elemanları

Native uygulamalar, işletim sistemi tarafından sağlanan kullanıcı arayüzü bileşenlerini kullanarak, kullanıcılara zaten aşina oldukları bir deneyim sunar.

1.1.4.2. Dokunmatik Jestler

Ekran boyutu ve yönlendirmesine dinamik olarak uyum sağlayabilen arayüzler, kullanıcılara kesintisiz bir deneyim sunar ve kullanım kolaylığı sağlar.

1.1.4.3. Adaptif Tasarım

Ekran boyutu ve yönlendirmesine dinamik olarak uyum sağlayabilen arayüzler, kullanıcılara sürekli bir deneyim sunar.

Uygulama tasarımı, kullanıcıların uygulama ile etkileşim kalitesini doğrudan etkileyen kritik bir faktördür. Bu nedenle, geliştiriciler kullanıcıların gereksinimlerini ve beklentilerini karşılayacak tasarımlar oluşturmalıdır.

1.1.5. Cross-Platform Uygulamalar ile Karşılaştırma

Native mobil uygulamalar ile cross-platform uygulamalar arasında tercih yapmak, geliştiricilerin karşılaştığı önemli bir karar noktasıdır. Her iki yaklaşımın da kendine özgü avantajları ve dezavantajları bulunmaktadır, ancak bazı temel farklar açıkça

belirginleştirilebilir.

Native mobil uygulamalar, özellikle performans ve kullanıcı deneyimi açısından önemli avantajlar sunar. Ancak, cross-platform geliştirme, bazı durumlarda daha hızlı geliştirme süreçleri ve daha düşük maliyetler sağlayabilir. Geliştiricilerin, projenin gereksinimlerine ve hedeflerine göre en uygun yaklaşımı seçmeleri önemlidir.

1.1.6. Test ve Dağıtım

Native Mobil Uygulama geliştirme sürecinin kritik aşamalarından biri, test ve dağıtımdır. Bu aşama, uygulamanın son kullanıcılar tarafından sorunsuz bir şekilde kullanılabilmesi için hayati öneme sahiptir. Test süreci, uygulamanın farklı cihazlarda ve işletim sistemlerinde beklenen performansı gösterip göstermediğini doğrular. Dağıtım ise uygulamanın son kullanıcılara ulaştırılmasını sağlar.

1.1.7. Test Süreci

1.1.7.1. Birim Testleri

Uygulamanın en küçük birimlerinin test edilmesi.

1.1.7.2. Entegrasyon Testleri

Bu testler, uygulamanın farklı modüllerinin birbiriyle uyumlu bir şekilde çalışıp çalışmadığını kontrol eder.

1.1.7.3. Kullanılabilirlik Testleri

Kullanıcı deneyimi odaklı testler, uygulamanın kullanım kolaylığını değerlendirir ve kullanıcıların uygulama içindeki etkileşimlerini analiz eder.

1.1.7.4. Performans Testleri

Uygulamanın yük altında nasıl performans gösterdiğini ölçen testler gerçekleştirilir.

1.1.8. Pazarlama Stratejisi ve Uygulama Analitiği

Native Mobil Uygulamanın başarısı, sadece teknik mükemmellikle değil, aynı zamanda etkili bir pazarlama stratejisi ve kapsamlı uygulama analitiği kullanımıyla da belirlenir. Pazarlama stratejisi, uygulamanın hedef kitlesine ulaşmasını ve kullanıcı tabanını genişletmesini sağlar. Uygulama analitiği ise kullanıcı davranışlarını anlamak ve uygulamayı bu yönde iyileştirmek için gereklidir.

1.1.9. Mobil Cihazlarda İçerik Yönetimi

Mobil cihazlarda içerik yönetimi, kullanıcıların uygulama içindeki bilgilere kolayca erişebilmesi ve etkileşimde bulunabilmesi için kritik bir unsurdur. Native Mobil Uygulama geliştiricileri için, içerik yönetim sistemi (CMS), uygulama içi içeriği düzenli olarak güncellemek ve kullanıcıların ilgisini çekmek için kullanılan güçlü bir araçtır.

Test ve dağıtım, etkili bir pazarlama stratejisi, uygulama analitiği kullanımı ve mobil cihazlarda içerik yönetimi, Native Mobil Uygulama geliştiricileri için çok önemlidir. Bu unsurlar, yalnızca teknik performansı değil, aynı zamanda kullanıcı deneyimini ve uygulamanın pazar yerindeki konumunu da doğrudan etkiler.

1.2. Cross-Platform Uygulama Geliştirme

Çapraz platform mobil uygulamalar, çeşitli işletim sistemleriyle uyumlu olan ve tek bir kod tabanı üzerinden çalışabilen mobil uygulama türleridir. iOS ve Android gibi farklı platformlarla uyumlu olan bu uygulamalar, tek bir geliştirme süreciyle her iki platformda da çalışabilir ve bu nedenle çift platform olarak da adlandırılırlar. Bu uygulama sistemleri, ürün ve hizmetlerin daha hızlı bir şekilde piyasaya sürülmesine yardımcı olur ve geniş bir kitleye ulaşma potansiyeli nedeniyle etkili bir seçenektirler. Birçok işletme native uygulamaları tercih etse de, native mobil uygulamalar sadece belirli bir işletim sistemi için geliştirildiği için yüksek maliyetlerle karşılaşabilirler.

Çapraz platform uygulamaları ise bu sorunu aşarak işletmelere daha ekonomik ve verimli bir mobil uygulama seçeneği sunarlar.

Çapraz platformlar sayesinde, farklı işletim sistemleriyle uyumlu uygulamaların geliştirilmesi artık mümkün hale gelmiştir. Bu durum, 2020 yılında 8 milyar Dolar değerinde olan hibrit platform geliştirme pazarının hızla büyümesine neden olmuştur. Doğru araç ve kaynak kullanımıyla, çapraz platform uygulamaları native bir uygulama kadar etkili olabilirler. İşletmeler, çoklu platform teknolojilerini kullanarak mobil uygulamanın toplam maliyetini azaltabilir ve aynı anda farklı platformlarda varlık göstererek yüksek verimlilik sağlayabilirler. Böylece, kaliteden ödün vermeden geniş bir kitleye ulaşabilirler.

1.2.1. Cross Platform Kullanım Avantajları Nelerdir?

Multi platform uygulamalar, pazara hızlı bir şekilde girmek isteyen ve düşük bütçesi olan işletmeler için oldukça avantajlıdır. Bu uygulamalar geliştirme süreleri açısından oldukça hızlıdır ve native uygulamalar kadar kullanışlıdır. Bu nedenle bu uygulamalar düşük bütçeli işletmeler tarafından talep görür.

1.2.2. Cross platform avantajları şu şekilde sıralanabilir;

Eğer tüm mobil işletim sistemlerine yönelik native bir uygulama geliştirmeyi hedeflerseniz, bütçenizin bu yönde önemli bir kısmını ayırmanız gerekebilir. Ancak cross-platform uygulamalar, bu maliyeti önemli ölçüde düşürerek daha düşük bir bütçeyle native uygulamalar kadar etkili bir mobil uygulama geliştirme fırsatı sunar.

Cross-platform uygulamalar sayesinde, native uygulamalarla aynı kullanıcı deneyimini sağlayarak mobil uygulama geliştirebilirsiniz. Bu uygulamalar, native uygulamaların özelliklerini koruyarak işletmenize avantaj sağlar.

Eğer pazarı hızlı bir şekilde penetrasyon etmek ve zamanı kısıtlı bir ortamda faaliyet göstermek istiyorsanız, cross-platform çözümler size farklı işletim sistemlerindeki kullanıcılara hızlı bir şekilde ulaşma imkanı sunar. Native uygulamaların geliştirme süreci daha uzun olabilir, ancak cross-platform çözümler ile zamanı daha verimli

kullanabilirsiniz.

Deneyimli bir mobil uygulama geliştirme ekibiyle işbirliği yaparak, mobil uygulama geliştirme sürecindeki yönetim risklerini azaltabilirsiniz.

Cross-platform uygulamalar sayesinde, kısa sürede geniş bir hedef kitleye ulaşma şansını elde edebilirsiniz. Bu uygulamalar, farklı işletim sistemlerine yönelik geliştirme işlemlerini birleştirerek hedef kitlenizi genişletmenize yardımcı olur.

Markanızı küresel ölçekte yeni pazarlara taşıyarak çapraz platform uygulamaları aracılığıyla geniş bir kitleye ulaşabilirsiniz. Bu uygulamalar, farklı işletim sistemlerine yönelik geliştirme sürecini birleştirerek markanızın pazara girişini hızlandırır.

Her iki mobil işletim sisteminde de yer alarak uygulama gelirlerinizi artırabilirsiniz.

1.2.3. Cross Platformda Kullanabildiğimiz Teknolojiler

1.2.3.1. React Native

2015 yılında Facebook tarafından tanıtılan React Native, mobil uygulama geliştirme alanında öne çıkan bir teknolojidir. Bu teknolojinin dikkate değer özellikleri arasında kodların yeniden kullanılabilirliği ve hızlı geliştirme süresi bulunmaktadır. React Native, geliştiricilere JavaScript programlama dilini kullanarak native, yani yerel uygulamalara benzer bir kullanıcı arayüzü sunma imkanı verir. Bu özellikleriyle, geliştiricilerin mobil uygulamalarını hızlı ve etkili bir şekilde oluşturmasına olanak tanır.

1.2.3.2. Xamarin

2011 yılında kurulan Xamarin, çapraz platform geliştirme teknolojilerinden biridir. Başlangıçta bağımsız bir şekilde faaliyet gösteren Xamarin, daha sonra Microsoft tarafından satın alınarak Microsoft bünyesinde çalışmaya başlamıştır. Bu teknoloji, C programlama dilini kullanarak mobil uygulamalar geliştirmek için açık kaynaklı bir çözüm sunar. Bu sayede, mobil uygulamalar Android ve iOS gibi birçok farklı

platformda çalışabilir hale gelir.

1.2.3.3. Flutter

Flutter, Google tarafından oluşturulan açık kaynaklı bir UI yazılım geliştirme kitidir. Android, iOS, Windows, Mac, Linux ve web için uygulamalar geliştirmek için kullanılıyor. Flutter'ın ilk sürümü "Sky" olarak biliniyordu ve Android işletim sisteminde çalışıyordu. Flutter ilk olarak 2015 Dart geliştirici zirvesinde tanıtıldı. Tanıtımında sabit bir şekilde saniyede 120 FPS çalışan uygulamalar geliştirilebileceği belirtildi. 4 Aralık 2018'de Flutter 1.0, Flutter Live etkinliğinde ilk "kararlı sürüm" olarak yayımlandı. 11 Aralık 2019'da Flutter Interactive etkinliğinde Flutter 1.12 yayımlandı. [2]

2. BÖLÜM

PROJEDEKİ YAZILIM TEKNOLOJİLERİ

Berber Randevu Projesinde native platform tercih edilmiştir. Bunun nedeni daha optimize ve hızlı çıktılar almasından dolayı. Ayrıca Cross platforma göre çok daha kapmsamlı operasyonlar ile geniş bir dünya sunmasındandır.

Verilerin yönetimi için Firebase kullanılmıştır. Bunun sebebi hiç bir backend operasyonu yapmadan kullanıma hazır şekilde sunulmuş bir yapı sunmasıdır. Hiç bir backend kodu yazmadan projemizin verilerini servis tarafına aktarmış bulunmaktayız.

Kullanıcın bilgilerini kullanabilmek için her cihaza özgü bir depolama uygulanmıştır. Bundan dolayı kullanıcın kendine özgü işlemlerini çok daha hızlı ve kolay şekilde yapmasına olanak sağlanmıştır.

2.1. Swift-UI

2014 yılında Apple tarafından duyurulan Swift, zaman içinde evrim geçirerek dünyanın en dikkat çekici ve güçlü programlama dillerinden biri haline gelmiştir.

Bugün, Swift sadece bir güncelleme değil, daha büyük bir değişimin de habercisidir; bu değişimin adı SwiftUI'dir. SwiftUI, kullanıcı arayüzlerini oluşturmak için yeni bir yaklaşım sunar. SwiftUI ile birlikte, geleneksel emir tabanlı programlamadan daha bildirimsel bir programlamaya doğru bir geçiş yaşanmaktadır. Bu iki yaklaşım arasındaki farka kısaca değinmek gerekirse, emir tabanlı programlamada "nasıl" sorusu üzerinde durulurken, bildirimsel programlamada "ne" olacağına odaklanılır. Bu durumda, programın daha fazla bilgi sahibi olması ve programcının isteklerine

göre hareket etmesi gerekmektedir.

Swift programalama dilinden önce apple tüm cihazlarında Objective-C isminde bir programlama dilini kullanmaktaydı. 1970' lerde Smalltalk ismi verilen bir programlama dili geliştirilmekteydi. Bu programlama dilinin en büyük özelliği o dönemde devrimsel bir gelişme olan nesnel yapıda olmasıydı. 1980' lere gelindiğinde procedural bir dil olan C dili iki geliştirici ile nesnel bir yapıya kavuşturulmak için Smalltalk programlama dili ile birleştirildi. Bunun sonucunda Objective-C dili doğdu. [3]

Bildirimsel programlamanın ışığında, uygulamanın nasıl olacağına dair cevapları küçük modüllerden oluşan bir yapıda sunarak daha modüler bir uygulama yapısına erişileceği öngörülmektedir. Apple, bu büyük değişimi zorunlu kılmamış, SwiftUI içinde UIKit kullanımını mümkün kılarak geliştiricilere daha esnek bir geçiş imkanı sunmuştur.

SwiftUI, uzun süredir devam eden Storyboard veya kod ile tasarım tartışmalarına son verecek yenilikçi bir yaklaşım sunmaktadır.

Ayrıca, Apple ürünleri için cross-platform bir arayüz tasarımı oluşturmak artık çok daha kolay hale gelmiştir.

SwiftUI'da, bileşenler görünüm olarak adlandırılır ve View protokolü ile tanımlanır. Bu protokolün zorunlu bir özelliği olan "body", bir View döndürür. Bu özellik, özyinelemeli bir yapı oluşturarak görünüm içerisinde görünüm yapılarının kolayca uygulanabilmesine imkan tanır. Bu da SwiftUI'nin esnekliğini ve kullanım kolaylığını artırır.

XCode 'da Apple Inc firması tarafından apple ürünleri için yazılım geliştirilmesi için yayınlanmış bir IDE dir. XCode ile özellikle Objective-C ya da Swift dili kullanılarak yazılım geliştirilebilir. [4]

2.1.1. Kütüphaneler, çalışma zamanı ve geliştirme

Swift, Objective-C ile aynı çalışma zamanı sistemini kullanır; ancak iOS 7 veya

macOS 10.9 ve sonrası sürümleri gerektirir. Swift ve Objective-C kodları, C ve C++ dahil olmak üzere aynı program içinde uzantılarla kullanılabilir. Ancak, C++ kodları doğrudan Swift içinde kullanılamaz. Bu durumda, bir Objective-C veya C wrapper Swift ile C++ arasında bir köprü oluşturmalıdır.

2.1.2. Bellek yönetimi

Swift, bellek yönetimi için Otomatik Referans Sayıcı kullanırken, Objective-C'de bellek yönetimi el ile yapılması gerekiyordu. 2011 yılında, bellek tahsis etme ve serbest bırakma işlemlerini basitleştirmek amacıyla ARC tanıtıldı.

ARC'nin olası sorunlarından biri, iki farklı örneğin birbirine güçlü bir referans döngüsü ile bağlandığı durumlarda, birbirlerinin referanslarını hiçbir zaman bırakmayacak olmalarıdır. Ancak, Swift bu tür durumlardan kaçınmak için "weak" ve "unowned" anahtar kelimelerini sağlar. Bu, güçlü referans döngülerini önler ve bellek sızıntılarını engeller.

2.1.3. Objective-C'den farkları

Swift, ifadelerin noktalı virgül (;) ile bitmesini zorunlu kılmaz ve bir satırda birden çok ifade kullanımına izin verir.

Başlık dosyaları kullanılmaz.

Swift, otomatik tür algılama özelliğini kullanır.

Jenerik programlama desteği vardır.

Fonksiyonlar birinci sınıf nesnelerdir.

Stringler Unicode'u tamamıyla destekler ve birçok Unicode karakterini tanımlamak ve operatör olarak kullanmak mümkündür.

2.2. Firebase

Google tarafından sürekli olarak geliştirilen ve çeşitli ihtiyaçları karşılayabilme

iddiasında olan firebase, ücretsiz bir platform olarak öne çıkıyor. Mobil uygulama geliştiricilerinin ihtiyaç duyduğu kontrol paneli ve kullanıcı veri deposu gibi temel unsurları ücretsiz olarak sunuyor. Günümüzde uygulamaların farklı cihazlardan aynı veriye erişme talebi artıyor. Firebase, tam olarak bu ihtiyaca cevap veriyor. Uygulamaların kullanıcı verilerini yönetme, analiz etme, kullanıcı oturumlarını tutma, bildirimler gönderme gibi işlemleri kolaylıkla gerçekleştirmek için geliştiricilere güçlü bir yönetim paneli sağlıyor. [5]

2011 yılında Andrew Lee ve James Tamplin tarafından ENVOLVE adlı bir şirket olarak kurulan Firebase, başlangıçta gerçek zamanlı veri tabanı sağlayan bir API olarak hizmet veriyordu. Ancak, 2014 yılında Google tarafından satın alınmasıyla birlikte, Firebase çok daha geniş kapsamlı bir platform haline geldi. Mayıs 2016'dan itibaren, Firebase geliştiricilere bir dizi yeni özellikle birlikte sunulmaya başladı.

Firebase, geliştiricilere uygulama oluşturma, yayınlama ve etkileşim alanlarında çeşitli araçlar ve hizmetler sunar.

2.2.1. Bulut Firestore

Cloud Firestore, Firebase ve Google Cloud'dan mobil, web ve sunucu geliştirme için esnek ve ölçeklenebilir bir veritabanıdır. Firebase Realtime Database gibi, veriler gerçek zamanlı dinleyiciler aracılığıyla sunulanların içerdiği tutarlar ve web için çevrimdışı destek sunar, böylece geliştiricilere ağ gecikmesinden veya internet bağlantısından bağımsız olarak çalışabilen duyarlı uygulamalar geliştirme imkanı tanır. Cloud Firestore, diğer Firebase ve Google Cloud ürünleriyle sorunsuz bir şekilde entegre olabilir, bu da geliştiricilerin verimliliğini artırır ve uygulamalarını daha kapsamlı hale getirir.

2.2.2. Firebase Kimlik Doğrulaması (Doğrulama)

Firebase Authentication, uygulamalarda kullanıcıların etkinliğini sağlamak için arka uç hizmetleri, kullanımı kolay SDK'lar ve hazır kullanıcı arayüzü kitaplıkları sunar. Parolalar, telefon numaraları, Google, Facebook ve Twitter gibi popüler birleşik kimlik sağlayıcıları ve daha fazlasını kullanarak kimlik doğrulamasını sağlar.

Firebase Authentication, diğer Firebase hizmetleriyle sıkı bir şekilde entegre olur ve OAuth 2.0 ve OpenID Connect gibi endüstri standartlarını kullanır, bu da özel bir arka uç ile kolayca entegre olabilir.

2.2.3. FireBase Realtime DataBase (Gerçek Zamanlı Veritabanı)

Firebase Realtime Database, bulutta barındırılan bir veritabanıdır. Veriler JSON formatında depolanır ve gerçek zamanlı olarak yüklenir. iOS, Android ve JavaScript SDK'larıyla platformlar arası uygulamalar oluşturulduğunda, tüm kullanıcılar bir Realtime Database örneğini paylaşır ve en son verilerle otomatik olarak güncellenirler.

2.2.4. FireBase Makine Öğrenimi

Firebase Machine Learning, Google'ın güçlü makine öğrenimi uzmanlığını Android ve iOS uygulamalarına yönelik kullanımı kolay bir paket içinde sunar. Makine öğrenimi konusunda yeterli bilgiye sahip olmasanız bile, gereken işlevselliği sadece birkaç satır kodla uygulayabilirsiniz. Ayrıca, kapsamlı bir makine öğrenimi geliştiricisiyseniz, Firebase Machine Learning, mobil uygulamalarınızda özel TensorFlow Lite modellerinizi kullanmanıza yardımcı olan kullanışlı API'ler sağlar.

2.2.5. FireBase Bulut Depolama (Bulut Depolama)

Firebase Cloud Storage, güçlü, basit ve maliyet açısından uygun bir nesne depolama hizmetidir, Google'ın bakış açısından tasarlanmıştır. Firebase SDK'ları, ağ kalitesinden bağımsız olarak Firebase uygulamaları için dosya yükleme ve indirme işlemlerine Google güvenliği ekler. Görüntü, ses, video veya diğer kullanıcı tarafından oluşturulan içerik parçalarını depolamak için SDK'lar kullanılabilir. Sunucu tarafında, aynı dosyaların değiştirilmesi için Google Cloud Storage (Google Bulut Depolama) kullanılır. Firebase storage, sunucunuza yüklenmesi gereken detaları kontrol etmenizi sağlamak amacıyla kendi güvenlik kuralları ile birlikte

gelir. [6]

2.2.6. Bulut İşlevleri

Firebase Cloud Functions, Firebase özellikleri ve HTTPS istekleri tarafından tetiklenen olaylara yanıt olarak arka uç kodunu otomatik olarak çalıştırmak için bir çerçeve sağlar. JavaScript veya TypeScript kodları Google Cloud'da depolanır ve yönetilen bir şekilde çalıştırılır. Bu, geliştiricilerin kendi sunucularını yönetmelerine veya ölçeklendirmelerine gerek olmadan, uygulamalarının işlevselliğini genişletmelerine olanak tanır.

2.2.7. Barındırma

Firebase Hosting, geliştiriciler için üretim düzeyinde web içeriği barındırma sağlar. Tek bir komutla, web uygulamaları hızla dağıtılabilir ve hem statik hem de dinamik içerik, global bir içerik sunulabilirliği ile sunulabilir.

2.2.8. Firebase Crashlytics

Firebase Crashlytics, uygulama yazılımının kararlılık sorunlarını genişletmeye, parlaklığı artırmaya ve aygıtlara yardımcı olmaya yönelik hafif ve gerçek zamanlı bir kilitlenme raporudur. Crashlytics, çökmeleri gruplayarak ve bunlara yol açanları vurgulayarak sorunu giderme süresinden tasarruf sağlar. Ayrıca, belirli bir çökmenin kaç kullanıcıyı etkilediğini belirler ve sorunun ciddiyeti arttığında uyarılar gönderir. Hangi kod bölümlerinin çökmelere neden olduğu ayrıca gösterilir.

2.2.9. Firebase Test Laboratuvarı

Firebase Test Lab, bulut tabanlı bir uygulama test platformudur. Tek bir işlemle Android veya iOS uygulamaları geniş bir cihaz yelpazesi ve cihaz sonuçları üzerinde test edilebilir ve sonuçlar (günlükler, videolar ve ekran görüntüleri dahil) Firebase konsolunda görüntülenebilir.

2.2.10. Firebase Uygulama Süreci

Firebase Uygulama Dağıtımı, güvenilir test kullanıcılarına uygulamanın sorunsuz bir şekilde dağıtımını sağlar. Uygulamalar test kullanıcılarının cihazlarına hızlı bir şekilde aktarılarak daha erken ve sık geri bildirimlerin alınmasını sağlar.

2.2.11. FireBase Performans Takibi (Performans İzleme)

Firebase Performance Monitoring, iOS, Android ve web uygulamalarının performans özellikleri hakkında fikir edinilmesine yardımcı olan bir hizmettir.

Uygulamalardan performans parçalarının toplanması için Performance Monitoring Yazılım Geliştirme Kiti kullanılır, ardından bu veriler Firebase konsolunda incelenip analiz edilebilir. Performans İzleme, işleyişin nerede ve ne zaman geliştirilebileceğinin anlaşılmasına yardımcı olur, böylece bu bilgilerin performans parametrelerinin iyileştirilmesi için kullanılabilir.

2.2.12. FireBase Uzaktan Yapılandırma (Uzak Yapılandırma)

Firebase Remote Config, kullanıcıların bir uygulama güncellemesini indirmesini gerektirmeden sunulmasını ve görünümün değiştirilmesine olanak sağlayan bir bulut hizmetidir. Remote Config kullanılırken, uygulamanın davranışını ve görünümünü kontrol eden uygulama içi değerler oluşturulur. Ayrıca, tüm uygulama kullanıcıları veya kullanıcı tabanının segmentleri için uygulama içi varsayılan değerleri geçersiz kılmak için Firebase konsolu veya Remote Config arka uç API'leri kullanılabilir.

2.2.13. Google Analytics

Analytics, Firebase özellikleriyle entegre olur ve geliştiricilere Firebase Yazılım Geliştirme Kitini kullanarak tanımlayabilecekleri 500'e kadar farklı etkinlik için sınırsız raporlama sağlar. Analytics raporları, kullanıcı davranışlarını net bir şekilde anlamaya yardımcı olur, bu da uygulama pazarlaması ve performans dağılımları hakkında beklenen sonuçların hazırlanmasını kolaylaştırır.

2.2.14. FireBase Tahminleri (Tahminler)

Firebase Predictions, kullanıcıların tahmin edilen performanslarına göre dinamik kullanıcı segmentleri oluşturmak için geliştiricilerin analiz sistemlerini makine öğrenimi modelleriyle entegre eder. Bu tahminler; Firebase Remote Config, Bildirim Oluşturucu, Firebase Uygulama İçi Mesajlaşma ve A/B Testi gibi özelliklerle otomatik olarak kullanılabilir. Ayrıca, daha fazla analiz yapmak veya üçüncü taraf araçlara aktarmak için sunulan tahmin verileri BigQuery'e aktarılabilir.

Tek seferlik mesajlar veya yinelenen kampanyalar için Bildirim Oluşturucusu ile birlikte Tahminler de mevcuttur. Örneğin, uygulamayı kullanmayı bırakacak kullanıcıların otomatik olarak bir bildirim alması gibi senaryolar uygulanabilir.

2.2.15. Firebase A/B Testi

Firebase Predictions, kullanıcıların tahminlenen performanslarına göre dinamik kullanıcı katmanları oluşturmak için geliştiricilerin analiz sistemlerini makine öğrenimi modelleriyle birleştirir. Bu tahminler; Firebase Remote Config, Bildirim Oluşturucu, Firebase Uygulama İçi Mesajlaşma ve A/B Testi gibi özelliklerle otomatik olarak entegre edilir. Ayrıca, daha kapsamlı analizler yapmak veya üçüncü taraf araçlara aktarmak için sunulan tahmin verileri BigQuery'e aktarılabilir.

Tek seferlik mesajlar veya yinelenen kampanyalar için Bildirim Oluşturucu ile birlikte Tahminler de kullanılabilir. Örneğin, uygulamayı kullanmayı bırakacak kullanıcıların otomatik olarak bir bildirim alması gibi senaryolar bu özelliklerle yönetilebilir.

2.2.16. FireBase Cloud Messaging (Bulut Mesajlaşma)

Firebase Cloud Messaging , platformlar arası mesajlaşma çözümü olarak hiçbir ücret ödemeden güvenilir bir şekilde mesajlaşma imkanı sunar.

FCM kullanılarak, bir görüntü uygulamasına yeni e-posta veya diğer verilerde saklananların bildirilebileceği bildirimler gönderilebilir. Kullanıcıların etkileşimde bulunması ve bağlı kalması için bildirim gönderimleri kolaylıkla gerçekleştirilebilir.

Anlık mesajlaşma gibi senaryolar için, bir mesajda uygulamaya aktarılan yükler 4KB'ye kadar olabilir.

2.2.17. Uygulama İçi Mesajlaşma

Firebase Uygulama İçi Mesajlaşma, hedefli ve içerik odaklı mesajlar göndererek kullanıcıların uygulama içinde daha fazla etkileşimde bulunmalarını teşvik eder. Bu mesajlar, kullanıcıların uygulama özelliklerini kullanarak belirli eylemlerde bulunmalarını teşvik etmek için gönderilebilir. Örneğin, kullanıcıların abone olmalarını, video izlemelerini, bir seviyeyi tamamlamalarını veya bir öğe satın almalarını teşvik etmek için uygulama içi mesajlar kullanılabilir. Mesajlar özel olarak kartlar, posterler, modeller veya resimler şeklinde olabilir ve tetikleyicilerle belirlenen zamanlarda kullanıcıların en çok fayda sağlayacakları zamanlarda görüntülenir.

2.2.18. Firebase Dinamik Bağlantılar (Dinamik Bağlantılar)

Firebase Dynamic Links, çeşitli platformlarda hızlı bir şekilde çalışan ve istenilen şekilde yönlendirilebilen bağlantılardır.

Bu dinamik bağlantılar, kullanıcılar için en iyi deneyimi sağlamak için mevcut platforma uygun şekilde açılır. Örneğin, bir kullanıcı iOS veya Android'de bir dinamik bağlantıya tıklarsa, içerik doğrudan yerel uygulamada açılabilir. Aynı bağlantıyı kişisel bir tarayıcıda açarsa, web içeriği tam olarak görüntülenebilir.

Bunun yanı sıra, dinamik bağlantılar indirme işlemleri arasında da çalışır: Bir kullanıcı iOS veya Android'de bir dinamik bağlantıyı açarsa ve ilgili uygulama yüklü değilse, kullanıcıya uygulamayı indirme seçeneği sunulabilir. Uygulama yüklendikten sonra, kullanıcı doğrudan bağlantıya erişebilir.

2.2.19. Firebase Uzantıları

Firebase Uzantıları, uygulamaların hızlı bir şekilde dağıtılmasına yardımcı olurken başlangıçta karşılaşılan zorlukları çözme sürecini hızlandırır.

Bir Firebase Uzantısı yüklendikten sonra, belirli bir görev veya olayları gerçekleştirmek için HTTPS isteklerine, Cloud Scheduler olaylarına veya diğer Firebase ürünlerinden gelen etkinliklere yanıt verebilir. Bu, geliştiricilerin özelleştirilmiş işlevsellikler eklemelerine ve uygulamalarını daha da geliştirmelerine olanak tanır.

2.2.20. Firebase Uygulama Dizine Ekleme

Firebase Uygulama Dizine Ekleme, uygulamayı Google Arama'ya entegre etmenin bir yoludur. Kullanıcılar uygulamayı yüklediklerinde, doğrudan aradıkları içeriğe erişebilirler. Uygulama Dizine Ekleme, genel içeriği kullanıcıların cihazlarında bulmalarına yardımcı olur, böylece kullanıcılar hızlı bir şekilde ihtiyaç duydukları bilgilere ulaşabilirler. Ayrıca, henüz indirmediği uygulamaları keşfetmelerine yardımcı olmak için otomatik tamamlama önerileri sunar. Kullanıcılar ilgili sorguları arattığında, uygulamalarını indirmeleri için bir yükleme kartı görüntülenebilir.

2.3. TEKNOLOJİLERİN PROJEYE UYARLANMASI

2.3.1. Swift-Ui

Projenin front-end kısmı swift ui teknolojisi kullanılarak yapıldı. Mimari olarak, model,view, view model mimarisi kullanılmıştır. Assetler dışarıdan projeye tanımlanarak görsel, yazı ve dış kaynaklar kullanılmıştır. View kısmını sadece arayüz sınıfları içerirken, bütün logic işlemler view model katmanında halledilmiştir. Serviste veriler use case abstract katmanında çağırılıp extension ile tanımlanmıştır. Projede kullanıcı hesabı yok ise Kayıt ol buton ile kayıt olup daha sonrasında ana ekrana girmiştir. Ana ekranda bottom tab bar kullanılmıştır. Bu sayede kullanıcı kolay bir şekilde sayfalar arasında geçiş yapabilmektedir. Kullanıcı ilanları görebilir Favorilerine ekleyebilir ve randevu alabilir. Ayrıca kullanıcı bilgilerini görüntülebilir ve düzenleyebilir.

Projenin çoklu dil desteğini sağlayabilmesi için localization yapılmıştır.tanımlanmış tr ve en dosyalarında ilgili key bilgileri ile cihazın seçili olduğu dil bilgisine göre

otomatik olarak kullanıcıya gösterilmektedir.

Kodlar nesne tabanlı programlamaya uygun olarak yazılmıştır.

Her bir bileşen ayrı sınıflara bölünerek sadece kendi işini yapacak şekilde ayrılmıştır.

Randevu sistemi, önce firebaseden ilanın id'sine göre tüm randevuları getirir. Daha sonra tüm zaman dilimlerinden daha önce randevu yapılmış saatleri filtreleyerek ui tarafına gönderilir. Bu işlemlerin tamamı View model kısmında yapılmaktadır. View kısmında gelen saat bilgilerine göre erilebilir saatler kullanıcın tercihine sunulur. Bu sayede daha önceden randevu alınmış saat bir daha alınamaz hale gelir.

Tüm veriler önce view model katmanında çekilir, daha sonrasında state veriler yüklenerek view katmanına gönderilir. View katmanı gelen veriyi kullanıcıya sunacak şekilde çizer.

Projece, swift dilinin gücünü verimli bir şekilde kullanarak oluşturulmuştur.

2.3.2. Firebase

Önceikle bir firebase hesabı açılıp daha sonrasında projenin bundle id'si ile uygulamaya bağlanmıştır.

2.3.2.1. Firebase Auth

Firebasenin ilk olarak kimlik doğrulama teknolojisi kullanılmıştır. İşleyiş şu şekildedir. Kullanıcı öncelikle e-mailini daha sonrasında şifresini girer. Girilen bilgiler view model katmanında firebase'ye gönderilir. Firebase Auth kendi içerisinde böyle bir hesap var mı kontrol eder. Daha sonrasında eğer hesap varsa, bulunan hesap ile şifre bilgilerini eşleştirir. Eğer her koşul doğru ise geriye True statement döner.

Eğer kullanıcın bir kaydı yoksa kullancı kayıt ol kısmına gelir. Kullanıcıdan bilgileri alındıktan sonra ilk olarak Firebase Auth Kısmına yönelendirilir. Burada böyle bir hesap var mı kontrol edilir. Eğer böyle bir hesap yoksa kayıt işlemi yapılır.

2.3.2.2. Firebase Firestore

Kullanıcın kaydı yapıldıktan sonra diğer bilgileri(adres telefon gibi) bunların kaydının tutulduğu ve firebasenin geniş imkanlar tanıdığı firebase firestore katmanına gönderilir. Kullanıcını bilgileri başarı ile tanımlanır. Kimlik doğrulama tanımlandıktan sonra kullanıcının tüm bilgileri Firebase firestore katmanından çekilir.

Kullanıcı bilgilerinin yanı sıra, İlanlar, Randevular gibi tüm bilgiler firebase firestore katmanında depolanır. Bu sayede kullanıcılar tüm ilanları ve randevuları başarılı ve hızlı bir şekilde anlık olarak görüntüleyebilir.

2.3.2.3. Firebase Storage

Kullanıcın json verilirinin tutulmasının yanı sıra bir de görsel verilerinin de tutulması gerekiyor. Ayrıca ilanların daha estetik ve kullanıcıya ilan edebilmesi için, Salon resimlerinin de alınması gerekiyordu. Bu kısımda firebase storage güçlü kasları ile bu sorun da ortadan kalkıyor. Kullanıcıdan alınan görsel uygun formata çevirildikten sonra firebase storageye gönderilir. Ve bu bilgileri geri çekebilmek çok daha kolay olacaktır.

2.3.3. Local Storage

Kullanıcının uygulamayı her kapatıp tekrar açtığında, kimlik doğrulama yapmasının önüne geçebilmek için swift-ui'ın yerel depolama teknolojisine ihtiyaç duyulmuştur. Kullanıcı giriş yaptıktan sonra giriş bilgileri yerel depolamaya kaydedilir. Daha sonrasında uygulama açıldığında önce yerel depolama alanına bakılır, eğer kullanıcı bilgileri varsa bu bilgiler ile giriş yapılmaya çalışılır. Eğer giriş başarılı olursa kullanıcıyı Ana ekrana atar, ama şifre değişikliği veya daha farklı bir işlem yapılmışsa, giriş başarısız olur ve Giriş ekranına atılır.

Kullanıcı çıkış yapmak istediğinde bu bilgiler yerel depolama alanından silinir.

3. BÖLÜM

TARTIŞMA, SONUÇ ve ÖNERİLER

3.1. Tartışma, Sonuç ve Öneriler

Swift-Ui teknolojisini ilk defa bu proje ile deneyimledik. Swift ui'ı kullanmamızda bize avantaj sağlayan en büyük özelliği, Apple teknolojisinin tüm kasları, açık bir şekilde sunulmuştu. Bu sayede istediğimiz operasyonları araştırarak kolaylıkla yapabildik. Back-end tarafında firebase teknolojisi ile tüm servis tarafını kolaylıkla halledebildik. Google tarafında geliştirlen bu teknoloji, developerların kendi başına iş yapmasını çok kolaylaştırmaktadır. Biz de bu sayede bu projeyi tamamlayabildik.

Tamamladığımız bu proje ile, berberler ve müşteriler arasındaki gerginliklerin ortadan kalkması beklenmektedir. İş, okul ve tüm hayat koşuşturması arasında boşuna kaybedilen her vakit biraz daha canımızı sıkmaktadır. Kuaförlerin randevu sistemi hali hazırda günümüzde çok kullanılmakta, bu sayede müşteriler vakit kaybı yaşamadan başarılı bir şekilde işlemlerini halledebilmektedir. İş berberlere geldiğinde, sadece işlerinde büyük cirolara sahip berberlerin sadece randevu sistemi olduğunu gözlemledik. Sadece büyük ciro yapan berber dükkanların aksine her berber bir randevu sistemini hak ediyordu. Ayrıca pahalı berberlere gidemeyen kullanıcların da zamanının boşa gitmemesi gerekiyordu. Bu yüzden tüm halkın kullanabileceği ve kolaylıkla erişebileceği bir proje için adımlarımızı attık. Berber randevu sistemi sayesinde ne müşteriler zaman kaybedecekti, ne de berberler bu sistemden mahrum kalacaktı. Berber randevu sistemi ile başarılı bir şekilde bu sorunun üstesinden gelebildik.

Projemizin eksik kaldığı noktaların başında geniş kullanıcı trafiğinin kullanılabilmesi

için firebase yerine gerçek bir back-end'e bağlanarak, daha hızlı ve erişilebilirlik olarak daha kolay hale getirilmesi gerekiyor. Ne kadar firebase bize çok olanak sağlasa bile lojik işlemlerin hepsini front-end tarafında yapmak zorunda kaldık. Bu da tersine mühendislikle uygulamanın kodlarına girilip kendi isteğine göre işlem yapılabilirliği sağlamaktadır. Yani kötü niyetli insanların eline geçip kurallar ihlal edip sistem bozulabilir. Bu lojik işlemler back-end tarafında işlenip ona göre bize çıktı üretebilirse, kötü niyetli insanların eline geçse bile herhangi bir sorun oluşmayacaktır. Bu sayede hem trafiği kontrol altına alabilir hem de daha güvenilir sistem kurulabilir.

KAYNAKLAR

1. Yazanlar, G., 2022, Cross-Platform ve Geleceği: Native Programlama, https://gelecegiyazanlar.turkcell.com.tr/blog/cross-platform-ve-gelecegi-native-programl

2. Vikipedi.

3. Erol, I.E., 2019. SWIFT İLE İOS UYGULAMA GELİŞTİRME: YENİ BAŞLAYANLAR İÇİN, İstanbul Aydın Üniversitesi Yayınları.

4. Erol, I.E., 2019. SWIFT İLE İOS UYGULAMA GELİŞTİRME: YENİ BAŞLAYANLAR İÇİN, İstanbul Aydın Üniversitesi Yayınları.

5. BLOG, T., 2024, Firebase Nedir?, https://blog.tekhnelogos.com/firebase-nedir-ne-ise-yarar/.

6. TalentGrid, 2023, Firebase Nedir?, https://talentgrid.io/tr/firebase-nedir/.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı, Soyadı	:	Burak Uncel
Uyruğu	:	Türkiye (T.C.)
Doğum Tarihi ve Yeri	:	22.01.2000 - ŞIRNAK
Telefon	:	0 545 849 43 29
Belgegeçer	:	0 352 555 88 77
E-posta	:	1030510157@erciyes.edu.tr
Adres	:	Yafes Mah. Çınar Sok. 73200, Cizre ŞIRNAK TÜRKİYE

EĞİTİM

Derece	**Kurum**	**Mez.Yılı**
Lise	Cizre Fen Lisesi, Cizre	2018
Ortaokul	Kaymakam Mümin Heybet Ortaokulu, Cizre	2014

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı, Soyadı	:	Furkan Kalender
Uyruğu	:	Türkiye (T.C.)
Doğum Tarihi ve Yeri	:	11.02.2001 - AFŞİN
Telefon	:	0 553 020 47 52
Belgegeçer	:	0 352 555 88 77
E-posta	:	1030510199@erciyes.edu.tr
Adres	:	MEVLANA Mah. KARASU Cd. 38200, Talas KAYSERİ TÜRKİYE

EĞİTİM

Derece	**Kurum**	**Mez.Yılı**
Lise	Afşin Anadolu Lisesi, AFŞİN	2019
Ortaokul	TEAŞ Ortaokulu, AFŞİN	2015