



**ToGather Mobil Uygulaması**

**BİTİRME PROJESİ 1. ARA RAPOR**

Bilgisayar Mühendisliği Bölümü

**DANIŞMAN**

Doç. Dr. Önder Demir

**ÖNSÖZ**

Bu belge, ToGather Mobil Uygulaması projemizin belirli bir aşamasını kapsayan **ara rapor** niteliğindedir. Proje sürecinde şimdiye kadar elde ettiğimiz gelişmeleri, karşılaştığımız sorunları ve bu sorunlara yönelik çözüm yaklaşımlarımızı sistematik bir şekilde sunmayı amaçlamaktadır.

Mobil uygulama geliştirme süreci boyunca hem teknik bilgi birikimimizi artırma hem de ekip içi işbirliği ve proje yönetimi konularında önemli deneyimler edinme fırsatı bulduk. Özellikle yazılım mimarisi, kullanıcı arayüzü tasarımı, veri yönetimi ve hata ayıklama konularında pratik bilgiler edinerek, teorik bilgimizi uygulama sürecine aktarma imkânı elde ettik.

Bu ara rapor, projenin genel hedefleri doğrultusunda geldiğimiz noktayı değerlendirmek ve bundan sonraki geliştirme sürecine ışık tutmak amacıyla hazırlanmıştır. Sürecin devamında edineceğimiz yeni bilgiler ve deneyimlerle projemizi daha da geliştirerek başarılı bir final ürün ortaya koymayı hedeflemekteyiz.

.

**İÇİNDEKİLER**

[**1.**](#_ythaxr7te570) **GİRİŞ 1**

[**1.1.**](#_7moo9rbgq3d) **Proje Konusu Ve Amacı 1**

[**1.2.**](#_7moo9rbgq3d) **Proje Kapsamı 2**

[**1.3.**](#_7moo9rbgq3d) **Kullanılan Yöntem ve Teknolojiler 3**

**2**[**.**](#_kng5gde7c5s) **GELİŞTİRME SÜRECİNE DAİR BULGULAR 5**

**2**[**.1.**](#_63qsmftri2in) **Öğrenilen Yeni Teknolojiler Ve Yaklaşımlar 5**

**2**[**.2.**](#_63qsmftri2in) **Karşılan Sorunlar Ve Çözümler 6**

**3**[**.**](#_rucfmxu5z7r) **MEVCUT DURUM VE DEĞERLENDİRME 8**

**3**[**.1.**](#_63qsmftri2in) **Tamamlanan Kısımlar 8**

**3.2.****Devam Eden Ve Planlanan Geliştirmeler 11**

**ÖZET**

**ToGather Mobil Uygulaması**

ToGather, insanların bir araya gelmesini kolaylaştırmak amacıyla geliştirilen bir mobil uygulamadır. Kullanıcılar uygulama üzerinden gruplar oluşturabilir, bu gruplara katılabilir ve ortak buluşma noktaları belirleyebilir. Harita üzerinden konum paylaşımı yapılabilir ve daha önce gerçekleşen buluşmalar geçmişte görüntülenebilir. Uygulama, sosyal etkileşimi kolaylaştırmayı ve planlama sürecini herkes için daha pratik hâle getirmeyi amaçlamaktadır.

**Mart, 2025 Öğrenciler**

**Oğuz EVİRGEN**

**Enes EKŞİ**

**ABSTRACT**

**ToGather Mobile Application**

ToGather is a mobile application developed to make it easier for people to come together. Through the app, users can create groups, join existing ones, and decide on common meeting points. Location sharing is available via an integrated map, and past gatherings can be viewed in a history section. The main goal of the app is to simplify social interactions and make the planning process more practical for everyone.

**March, 2025 Students**

**Oğuz EVİRGEN**

**Enes EKŞİ**

**KISALTMALAR**

**GMS :** Google Maps SDK

**CI/CD :** Continuous Integration / Continuous Deployment

**MVI :** Model–View–Intent

**DB**  : Database

**UI** : User Interface

**CRUD :** Create, Read, Update, Delete

**ŞEKİL LİSTESİ**

[**Şekil 1.1** Harita üzerinde kullanıcıların konumuna uygun avatarlarının bulunduğu mobil arayüz tasarımı.](#_gi4rp3xvou82)

## 1. GİRİŞ

### 1.1. Proje Konusu ve Amacı

Bu proje kapsamında geliştirilen **ToGather**, kullanıcıların grup oluşturarak ortak buluşma planları yapabildiği, harita üzerinden konum seçimi gerçekleştirebildiği ve buluşma geçmişlerini takip edebildiği, Android tabanlı bir mobil uygulamadır. ToGather uygulaması, bireysel kullanıcıların yanı sıra ilerleyen aşamalarda **daha geniş çaplı sosyal organizasyonlara** da ev sahipliği yapabilecek şekilde ölçeklenebilir olarak tasarlanmıştır.

Uygulamanın temel amacı, kullanıcıların sosyal etkileşimlerini organize etmelerini kolaylaştırmak ve ortak karar alma süreçlerinde yaşanan karışıklıkları ortadan kaldırmaktır. Bu kapsamda kullanıcılar, kendi gruplarını oluşturabilir, grup üyeleriyle birlikte buluşma noktası belirleyebilir ve planlamalarını harita entegrasyonu üzerinden kolaylıkla gerçekleştirebilir.

**ToGather**, yalnızca bireyler arası etkileşimle sınırlı kalmayıp, aşağıdaki gelişmiş özellikleri de desteklemeyi hedeflemektedir:

* **Daha büyük çaplı etkinlik organizasyonları**: Topluluklar, kulüpler veya işletmelerin geniş katılımlı etkinlikler planlayabileceği organizasyon modülü,
* **Konum bazlı kampanya ve fırsatlar**: Uygulama üzerinden kullanıcıların bulundukları konuma yakın **k**afeler, restoranlar veya etkinlik mekânlarının özel kampanyalarını görüntüleyebilmesi ve bu kampanyalara kolayca katılım sağlayabilmesi,
* **İşletmelere özel tanıtım alanları**: İşletmelerin uygulama üzerinden kullanıcı kitlesine ulaşabileceği, promosyon veya duyurular yapabileceği yapılar,
* **Etkinlik daveti ve katılım sistemi**: Kullanıcıların davet gönderip alabileceği, katılım durumu belirtme sistemlerinin desteklenmesi.

Bu genişletilebilir yapı sayesinde **ToGather** yalnızca bir buluşma planlama uygulaması değil, aynı zamanda sosyal etkileşimi teşvik eden, kullanıcıları çevresel fırsatlarla buluşturan ve yerel işletmeleri dijital ortama taşıyan bir platform haline gelecektir.

Uygulamanın ilk fazında, temel işlevsellik olan grup oluşturma, harita entegrasyonu ve buluşma geçmişi yönetimi üzerinde çalışılmıştır. Ancak tüm yazılım mimarisi, gelecekte bu vizyonun gerçekleşebilmesine olanak sağlayacak şekilde esnek ve sürdürülebilir olarak kurgulanmıştır.

### 1.2. Kapsam

Modern yaşamda bireylerin sosyal etkileşimlerini planlamak ve sürdürmek, zaman ve mekân koordinasyonu açısından çeşitli zorluklar barındırmaktadır. Özellikle arkadaş grupları, kulüpler, topluluklar veya öğrenci grupları gibi küçük grupların bir araya gelme süreçlerinde; tüm üyelerin uygunluk durumu, buluşma yeri seçimi ve bu bilgilerin paylaşılması ciddi zaman kaybına ve karışıklığa neden olabilmektedir.

ToGather, bu soruna yönelik pratik ve kullanıcı dostu bir çözüm sunmaktadır. Özellikle konum bazlı buluşma noktası belirleme, grup oluşturma, bildirimler, etkinlik geçmişi ve konum izinlerine bağlı harita takibi gibi işlevleriyle, kullanıcıların daha hızlı ve etkili bir şekilde plan yapmalarını sağlar. Ayrıca mobil platformda geliştirilmiş olması, uygulamanın taşınabilirliğini artırmakta ve kullanıcıların istedikleri an uygulamaya erişmelerine olanak tanımaktadır.

Bu proje aynı zamanda, Android tabanlı uygulama geliştirme süreci boyunca öğrencilerin yazılım mimarisi, kullanıcı arayüzü tasarımı, veri yönetimi ve üçüncü parti API entegrasyonları konularında uygulamalı bilgi ve beceri kazanmalarını sağlamaktadır. Bu yönüyle ToGather, yalnızca teknik bir ürün değil, aynı zamanda öğrenme temelli bir gelişim projesi olarak da değerlendirilmektedir.

### 

### 1.3. KULLANILAN YÖNTEM VE TEKNOLOJİLER

ToGather mobil uygulaması, güncel yazılım geliştirme yaklaşımları ve modern teknolojiler temel alınarak iki ana bileşen üzerinde yapılandırılmıştır: **Android (mobil istemci)** ve **Backend (sunucu tarafı)**. Her iki katman da yüksek okunabilirlik, sürdürülebilirlik, test edilebilirlik ve genişletilebilirlik prensipleri doğrultusunda geliştirilmiştir.

#### 1.3.1. Android Tarafı (Mobil Uygulama)

Android uygulaması, **Kotlin** programlama dili ile geliştirilmekte olup, güncel mimari yaklaşımlardan biri olan **MVI (Model–View–Intent)** yapısı temel alınmıştır. MVI mimarisi sayesinde kullanıcı arayüzü ile veri katmanı arasındaki ilişki tek yönlü veri akışıyla yönetilmekte ve UI durumları (State) merkezi bir yapı üzerinden kontrol edilmektedir. Bu yaklaşım, özellikle hata ayıklama, test edilebilirlik ve yeniden kullanılabilirlik açısından önemli avantajlar sağlamaktadır.

Mobil uygulama geliştirme sürecinde kullanılan başlıca teknolojiler ve kütüphaneler şunlardır:

* **Hilt:** Dependency Injection (bağımlılık enjeksiyonu) yapısını sağlamak için kullanılmıştır. Uygulamanın modülerliğini ve test edilebilirliğini artırmaktadır.
* **Retrofit:** API isteklerinin yönetilmesi ve RESTful servislerle haberleşme için tercih edilmiştir.
* **Room:** Yerel veritabanı işlemlerinin yönetilmesinde kullanılır. Uygulama içerisinde bazı verilerin çevrimdışı erişilebilirliği için tercih edilmektedir.
* **Google Maps SDK (GMS):** Harita gösterimi ve konum tabanlı işlemler (buluşma noktası belirleme vb.) için entegre edilmiştir.
* **Firebase:** Kimlik doğrulama, bildirim gönderimi ve temel veri saklama işlemleri için arka plan servis olarak kullanılmaktadır.
* **Coroutines & StateFlow:** Asenkron işlemler ve UI durum yönetimi için Kotlin’in coroutine yapısı ile birlikte StateFlow kullanılmaktadır.

#### 1.3.2. Backend Tarafı

Uygulamanın Backend tarafı, **Java 21** ile yazılmış ve **Spring Boot 3.4.4** çerçevesi ile yapılandırılmıştır. Backend sistemi, mobil istemciden gelen talepleri karşılamak, veri yönetimini gerçekleştirmek ve kullanıcı işlemlerini (grup oluşturma, buluşma geçmişi kaydı, kampanya entegrasyonu vb.) yürütmek üzere tasarlanmıştır.

Kullanılan başlıca Backend tarafı teknolojiler şunlardır:

* **Spring Boot 3.4.4:** Mikroservis mimarisi prensiplerine uygun, hafif ve genişletilebilir bir backend altyapısı sağlar.
* **Docker:** Uygulamanın konteynerize edilmesi sayesinde platform bağımsız olarak taşınabilirlik ve kolay dağıtım sağlanmıştır.
* **Spring Security:** API katmanında temel kimlik doğrulama ve yetkilendirme işlemleri için kullanılmaktadır.
* **Deployment Teknolojileri:** Projenin ilerleyen aşamalarında **CI/CD (Continuous Integration / Continuous Deployment)** süreçleri entegre edilerek, otomatik test ve dağıtım süreçleri hayata geçirilecektir. Bulut ortamına deploy için çeşitli alternatifler değerlendirilmektedir (örneğin: AWS, Google Cloud, Railway.app vb.).
* **Veritabanı:** Verilerin kalıcı olarak saklanması için ilişkisel veritabanı sistemleri tercih edilmektedir. Bu kapsamda PostgreSQL veritabanı ile çalışılmıştır.

## 2. GELİŞTİRME SÜRECİNE DAİR BULGULAR

### 2.1. Öğrenilen Yeni Teknolojiler ve Yaklaşımlar

ToGather projesi kapsamında yürütülen geliştirme süreci, sadece uygulama ortaya koyma amacıyla değil, aynı zamanda teknik bilgi birikimini artırma ve yeni teknolojileri uygulamalı olarak öğrenme açısından da oldukça verimli olmuştur. Proje boyunca hem mobil hem de backend tarafında birçok yeni teknolojiyle doğrudan çalışma fırsatı yakalanmıştır. Bu süreçte edinilen başlıca kazanımlar aşağıda iki ana başlık altında sunulmuştur.

#### 2.1.1. Android Tarafında Edinilen Bilgi ve Beceriler

Mobil uygulama geliştirme sürecinde daha önce temel düzeyde bilgi sahibi olunan bazı konular derinlemesine öğrenilmiş ve yeni teknolojiler ilk kez deneyimlenmiştir:

* **MVI (Model-View-Intent) Mimarisi**: Proje süreci boyunca MVI mimarisi uygulamalı şekilde kullanılarak bu yapının temel bileşenleri olan ViewState, Intent ve Reducer gibi kavramlar detaylı olarak kavranmıştır. Tek yönlü veri akışının nasıl sağlandığı, kullanıcı arayüzü durumlarının nasıl yönetildiği ve geri dönülebilir UI durumları oluşturmanın önemi gibi konularda pratik bilgi edinilmiştir.
* **Clean Code Prensipleri:** Kodun okunabilirliği, sürdürülebilirliği ve test edilebilirliği açısından önem arz eden temiz kod prensipleri, modüler yapı, fonksiyonel ayrım, sadeleştirme, anlamlı isimlendirme ve bağımlılıkların netleştirilmesi gibi kriterler üzerinden uygulamalı olarak öğrenilmiştir.
* **Google Maps SDK (GMS) Kullanımı:** İlk kez aktif bir harita bileşeniyle çalışılmış; harita gösterimi, kullanıcı konumu alma, marker yerleştirme ve konum izinlerinin yönetimi gibi fonksiyonlar üzerinden GMS API'lerinin nasıl entegre edildiği ve yönetildiği öğrenilmiştir.

#### 2.1.2. Backend Tarafında Edinilen Bilgi ve Beceriler

Sunucu tarafında yapılan geliştirmeler sayesinde backend teknolojileriyle ilgili birçok yeni kavram ve yapı ilk kez deneyim edilmiştir. Özellikle büyük ölçekli ve üretime uygun bir sunucu uygulamasının nasıl kurgulanacağına dair değerli deneyimler elde edilmiştir:

* **CI/CD Süreçleri:** Uygulamanın geliştirme ve dağıtım süreçlerinin otomatikleştirilmesi için kullanılan sürekli entegrasyon ve sürekli dağıtım (CI/CD) prensipleri öğrenilmiştir. Bu sayede hem kod kalitesi hem de üretim ortamına geçiş süreçleri daha güvenli ve hızlı hâle gelmektedir.
* **Container ve Sunucu Mantığı:** Docker kullanımı ile birlikte uygulamanın platform bağımsız olarak çalıştırılabilmesi sağlanmış, container mimarisi ile yazılımın taşınabilirliği ve dağıtımı konusunda pratik bilgi edinilmiştir. Uygulamanın fiziksel bir makine yerine izole bir ortamda çalışmasının avantajları doğrudan deneyimlenmiştir.
* **Core Katman ve Bağımlılık Yönetimi:** Spring Boot çerçevesinde projeyi modüler yapıda kurmak amacıyla core katmanlarının ayrılması, servis, veri ve yapı katmanlarının izole edilmesi ve farklı bileşenlerin birbiriyle olan bağımlılıklarının doğru şekilde yönetilmesi konularında tecrübe kazanılmıştır. Bu sayede daha okunabilir, test edilebilir ve genişletilebilir bir backend yapısı oluşturulmuştur.

### 2.2. Karşılaşılan Sorunlar ve Çözümler

#### 2.2.1. Tasarım Süreci ve Arayüz Oluşturma Problemleri

Projenin başlangıç aşamasında, kullanıcı arayüzü (UI) tasarımlarının hazırlanması sürecinde birtakım zorluklar yaşanmıştır. Özellikle Figma gibi profesyonel tasarım araçlarına hâkimiyet eksikliği, tutarlı ve kullanıcı dostu bir arayüz oluşturulmasını zorlaştırmıştır. Görsel uyumsuzluklar, boşluk ve hizalama problemleri ile sezgisel olmayan ekran akışları gibi sorunlarla karşılaşılmıştır.

Bu problemi çözmek adına, Google tarafından sunulan **Material Design** yönergeleri detaylı olarak incelenmiş ve Android Jetpack’in sunduğu **Material Components** aktif olarak kullanılmaya başlanmıştır. Bu sayede tasarım süreçleri daha sistematik ve standartlara uygun şekilde ilerlemiş; düğmeler, kartlar, formlar ve gezinme bileşenleri gibi birçok öğe için hazır çözümler uygulanarak modern ve tutarlı bir kullanıcı arayüzü elde edilmiştir.

#### 2.2.2. Backend ve Android Client Arasındaki Bağlantı Problemleri

Proje sürecinde mobil uygulama ile sunucu arasında veri alışverişi yapılması planlandığında, backend servislerinin henüz canlıya (production ortamına) alınmamış olması çeşitli entegrasyon sorunlarına neden olmuştur. Özellikle Android istemciden gelen taleplerin cevaplanamaması ve lokal ortamda çalışan backend servislerine mobil cihazdan doğrudan erişim sağlanamaması önemli bir engel oluşturmuştur.

Bu problem, backend uygulamasının **Docker** ortamında çalıştırılması ve hem yerel hem de mobil cihaz üzerinden erişilebilir hale getirilmesiyle aşılmıştır. Docker konteynerleri sayesinde uygulama bileşenleri izole bir ortamda çalıştırılmış; API testleri sağlıklı şekilde gerçekleştirilmiştir.

#### 2.2.3. Mimari Yapılanma ve Kod Organizasyonu Sorunları

Projenin ilk aşamalarında, Android uygulamasının mimari yapısını kurgularken **ekranlar arası veri paylaşımı**, **durum yönetimi**, **modülerlik** ve **bağımlılıkların yönetimi** konularında çeşitli mimari karmaşalarla karşılaşılmıştır. Uygulamanın büyüdükçe yönetilemez hale gelmemesi için sürdürülebilir ve temiz bir yapı kurma gerekliliği doğmuştur.

Bu bağlamda, farklı açık kaynak projeler ve kurumsal uygulama örnekleri incelenmiş; MVI mimarisi esas alınarak yeni bir yapılanma oluşturulmuştur. Kodun sorumluluklara göre ayrılması (presentation, domain, data), dependency injection altyapısının Hilt ile yönetilmesi ve durum yönetiminin StateFlow yapısıyla ele alınması sayesinde bu problemler aşılmıştır.

#### 2.2.4. Google Maps SDK Kullanımı ile İlgili Zorluklar

Uygulamanın önemli işlevlerinden biri olan harita tabanlı buluşma noktası belirleme sürecinde, **Google Maps SDK** entegrasyonu sırasında teknik zorluklarla karşılaşılmıştır. Özellikle konum izinlerinin yönetimi, harita üzerinde marker yerleştirme ve kullanıcıya en iyi deneyimi sunacak harita konumlamasının sağlanması konularında çeşitli denemeler yapılmıştır.

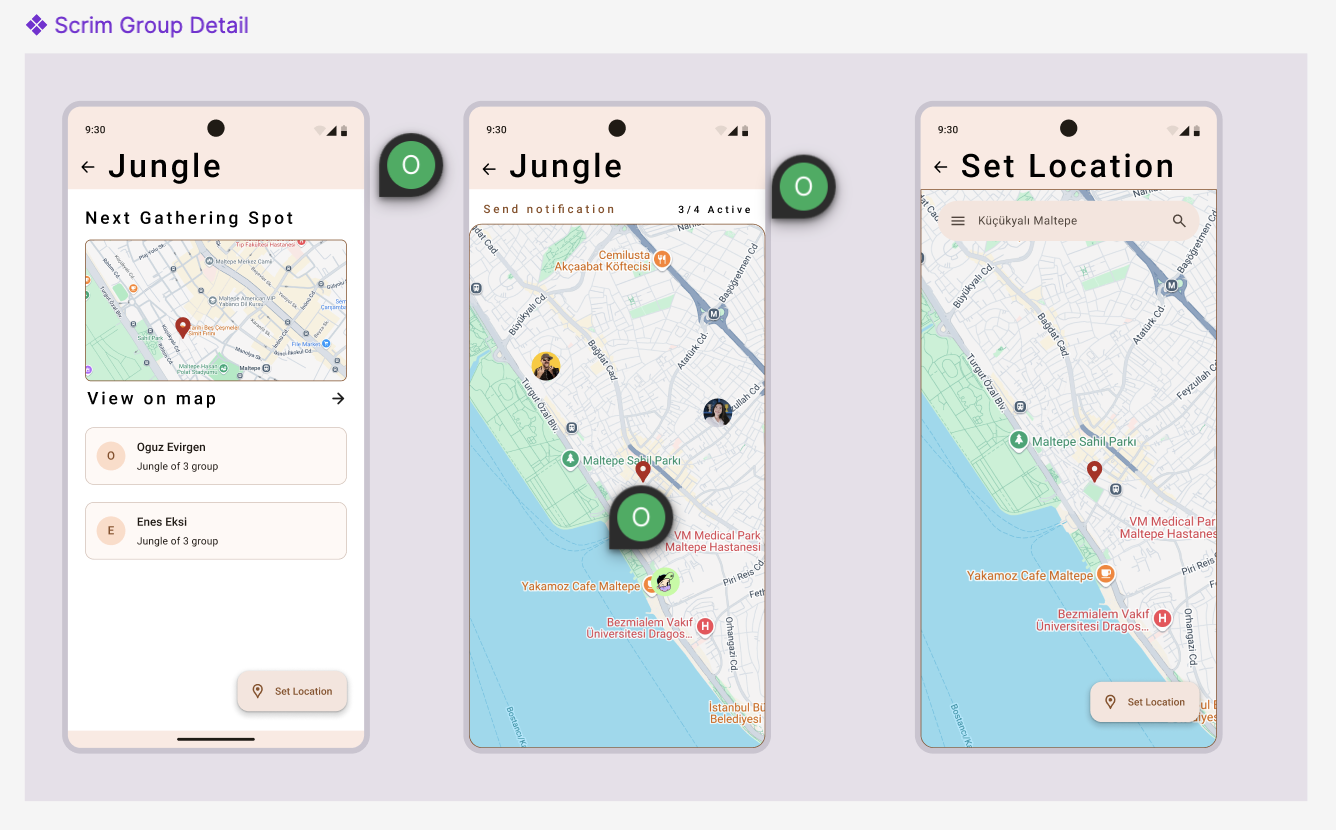
Bu sorunlar halen proje kapsamında üzerinde çalışılmakta olan bir alan olup, daha stabil ve kullanıcı dostu bir harita deneyimi oluşturmak amacıyla araştırmalar ve denemeler devam etmektedir. Bu bağlamda, Google Maps’in resmi dökümantasyonu, Stack Overflow tartışmaları ve çeşitli açık kaynak projelerden faydalanılmaktadır.

## 3. **MEVCUT DURUM VE DEĞERLENDİRME**

### 3.1. Tamamlanan Kısımlar

#### 3.1.1. Arayüz Tasarımı ve Kullanıcı Deneyimi

Uygulamanın arayüz tasarım süreci, kullanıcı deneyimini ön planda tutacak şekilde **Figma** aracı kullanılarak yürütülmüştür. Başlangıçta tasarıma yönelik deneyim eksikliği nedeniyle bazı zorluklar yaşansa da, bu süreçte Google’ın sunduğu **Material Design** ilkeleri incelenmiş ve rehber alınarak tasarım süreci yapılandırılmıştır.



**Şekil 1.1** [Harita üzerinde kullanıcıların konumuna uygun avatarlarının bulunduğu mobil arayüz tasarımı](#_gi4rp3xvou82)

Material Design sistemine ait **hazır bileşenler (components)** kullanılarak oluşturulan ekranlarda tutarlı ve modern bir kullanıcı arayüzü elde edilmiştir. Örneğin, kullanıcı avatarları, harita kartları, konum belirleme düğmeleri ve gezinme unsurları gibi öğeler, Material Design bileşenleriyle inşa edilmiştir. Şekil 1.1’de paylaşılan ekran görsellerinde de görüldüğü üzere, grup detay ekranı, harita ekranı ve konum belirleme ekranı gibi temel UI bileşenleri tamamlanmıştır. Bu sayede kullanıcılar buluşma yerlerini görebilmekte, üyeleri listeleyebilmekte ve konum belirleme işlemlerini gerçekleştirebilmektedir.

#### 3.1.2. Backend Geliştirmeleri

Sunucu tarafında geliştirilen modüller ve sistem altyapısı şu şekilde özetlenebilir:

* **Veritabanı Tasarımı (DB Design):** Kullanıcılar, gruplar, buluşma noktaları ve bildirim sistemini kapsayan temel veritabanı tabloları tasarlanmış ve ilişkisel model kurulmuştur.
* **Base Mimari Yapı:** Spring Boot çatısı altında sürdürülebilir ve modüler bir backend mimarisi kurulmuştur.
* **Database Bağlantıları:** Veritabanı işlemleri için JPA/Hibernate tabanlı yapı kurulmuş, repository katmanları oluşturulmuştur.
* **CI/CD Süreçleri:** Kodun sürümlenmesi, test edilmesi ve otomatik dağıtımı için temel continuous integration/deployment altyapısı oluşturulmuştur.
* **Kimlik Doğrulama ve Yetkilendirme:** JWT tabanlı **Authentication** ve kullanıcı bazlı **Authorization** mekanizmaları devreye alınmıştır.
* **CRUD Operasyonları:** Kullanıcı, grup ve buluşma noktaları gibi temel varlıklar için CRUD işlemlerini kapsayan REST servisleri geliştirilmiştir.
* **Exception Yönetimi:** Sunucu taraflı hata kontrolü için özelleştirilmiş **global exception handler** yapısı oluşturulmuştur.
* **Logger Yapısı:** Uygulamanın hata ve işlem kayıtlarının izlenebilmesi için merkezi bir **logger sistemi** yapılandırılmıştır.

#### 3.1.3. Android (Client) Geliştirmeleri

Mobil uygulama tarafında tamamlanan başlıca yapılar şu şekildedir:

* **Base Mimarinin Kurulması:** Uygulama, **MVI (Model–View–Intent)** mimarisi üzerine kurulmuştur. ViewState, Event, Intent ve Reducer yapılarıyla modüler ve test edilebilir bir akış oluşturulmuştur.
* **Modern Android Kütüphaneleri:** Hilt (DI), Retrofit (API), Room (yerel veri), Coil (görsel yükleme), Coroutine & StateFlow (durum yönetimi) gibi güncel kütüphaneler projeye entegre edilmiştir.
* **Harita İşlevleri:** Google Maps SDK entegrasyonu sağlanarak kullanıcı konumu, marker gösterimi ve konum seçimi gibi özellikler başarıyla devreye alınmıştır.
* **Session Management:** Kullanıcının oturum yönetimi ve verisinin güvenli taşınması için temel session yapıları kurgulanmıştır.
* **Önbellekleme Mekanizmaları:** Yerel verilerin hızlı erişimi için temel **cache** yapıları (örneğin Room kullanımı) devreye alınmış ve ileriye dönük genişletilebilir hale getirilmiştir.

### 3.2. Devam Eden veya Planlanan Geliştirmeler

#### 3.2.1. Android Tarafında Devam Eden Geliştirmeler

* **Harita İşlevlerinin Tamamlanması:** Google Maps SDK kullanımı temel düzeyde entegre edilmiş olup, kullanıcı konumunu izleme, harita üzerinde dinamik marker ekleme ve grup üyelerinin konumlarının eşzamanlı olarak gösterilmesi gibi işlevlerin eksiksiz ve hatasız şekilde çalışması için aktif olarak geliştirme yapılmaktadır. Ayrıca harita bileşenlerinin kullanıcı dostu hâle getirilmesi, etkileşimlerin akıcı olması ve mobil performansın korunması için çeşitli optimizasyonlar planlanmaktadır.
* **Session Management ve Local Cache Yapılarının Genişletilmesi:** Uygulamanın daha stabil çalışması için oturum yönetimi (session) sistemi güçlendirilecek ve kullanıcı bazlı önbellekleme (caching) stratejileri uygulanacaktır. Bu sayede ağ bağlantısı kopsa dahi, belirli verilerin yerel olarak erişilebilir olması sağlanacak ve kullanıcı deneyimi kesintisiz hâle getirilecektir.
* **Performans Refaktörleri:** Uygulama büyüdükçe, kod yapısının sürdürülebilir kalabilmesi için refactoring (yeniden yapılandırma) çalışmaları planlanmaktadır. Özellikle UI bileşenleri, ViewModel yapıları, veri akış mekanizmaları ve dependency injection (bağımlılık enjeksiyonu) bölümleri sadeleştirilerek geliştirilebilir ve okunabilir bir kod tabanı oluşturulacaktır.

#### 3.2.2. Backend Tarafında Devam Eden Geliştirmeler

* **Sunucuya Deployment (Canlı Yayın) İşlemleri:** Projenin üretim ortamına taşınabilmesi adına, dockerize edilmiş backend servislerinin bir bulut sağlayıcıya kalıcı olarak deploy edilmesi planlanmaktadır. Bu işlem tamamlandığında, Android uygulaması doğrudan gerçek sunucular üzerinden hizmet alacak ve uygulama, dış dünyaya açık hâle gelecektir.
* **CI/CD Pipeline Kurulumu(Sunucu Bazlı):**
* **Bildirim ve Mail Servislerinin Kurulması:** Kullanıcıların uygulama içerisindeki hareketleri ile ilişkilendirilmiş olarak anlık bildirimler (push notification) ve e-posta bildirimleri gönderebilmek adına, Firebase Cloud Messaging (FCM) ve SMTP tabanlı e-mail servisleri entegre edilmektedir. Bu sistem, özellikle grup etkinlikleri, buluşma hatırlatmaları veya kampanya duyuruları gibi kritik kullanıcı etkileşimlerini destekleyecektir.
* **Kafka Kurulumu ve Asenkron İletişim Altyapısı:** Projenin ilerleyen fazlarında mikroservis mimarisine evrilebilmesi adına, mesajlaşma altyapısı olarak **Apache Kafka** kurulması hedeflenmektedir. Kafka, servisler arası iletişimi asenkron hâle getirerek sistemin ölçeklenebilirliğini ve esnekliğini artıracaktır. Bu yapı özellikle yüksek kullanıcı yoğunluğuna sahip senaryolarda sistemin dayanıklılığını güçlendirecektir.
* **Eksik Sınıf ve Servislerin Tamamlanması:** Backend tarafında yer alan bazı servisler (örneğin kullanıcı ayar yönetimi, grup filtreleme, geçmiş detaylandırma servisleri) ve Android istemci tarafında henüz entegre edilmemiş veri modelleri ve DTO yapıları tamamlanacaktır. Android ve Backend ekipleri bu eksik yapıların ortak olarak tamamlanması üzerine çalışmalar yürütmektedir.
* **Performans ve Kod Kalitesi İyileştirmeleri:** Kodun daha sade, anlaşılır ve bakımı kolay bir hâle gelmesi için hem istemci hem de sunucu tarafında refactoring süreci devam etmektedir. Özellikle redundant kodların temizlenmesi, servis katmanlarının sadeleştirilmesi ve ortak bileşenlerin soyutlanması gibi adımlar uygulanacaktır.