**MOBİL UYGULAMA GELİŞTİRME**

**PROJE RAPORU**

**PROJENİN ADI:** CHITCHAT

**GRUP ÜYELERİ:**

415214 MELİKE NUR YILMAZ

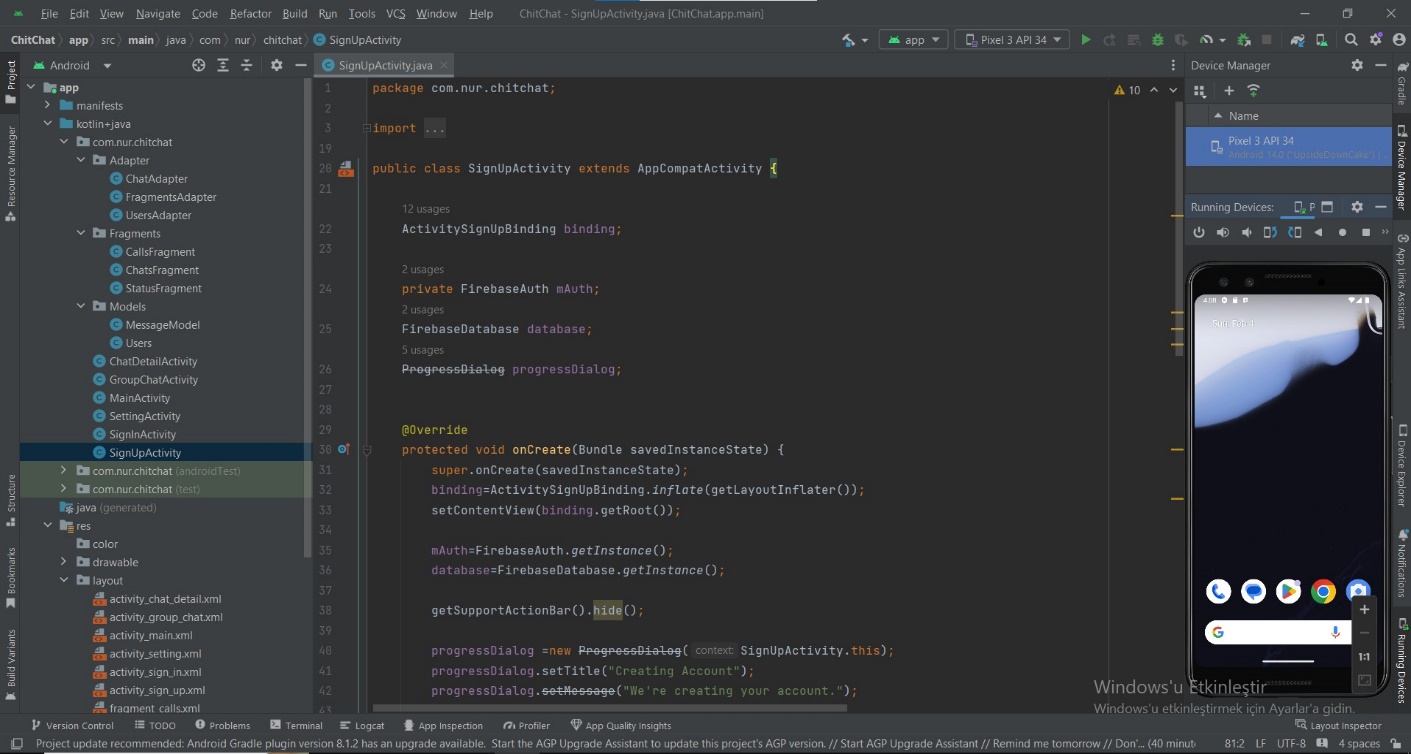
**İÇİNDEKİLER**

1. Proje Tanıtımı
2. Proje Detaylı Anlatımı
3. Kaynaklar
4. **Proje Tanıtımı**

Mobil uygulamaların günümüzde önemi oldukça fazladır. Her telefon uygulamasının kendine göre birçok özelliği bulunmaktadır. Hangi amaçla kullanılması gerektiği de önemlidir.

ChitChat uygulaması, internet üzerinden telefon kullanıcılarının birbiriyle iletişime geçmesini sağlayan bir mobil uygulamadır. ChitChat, mail hesabı ile kullanılabiliyor. Mail adresleri ile kayıt olan telefon sahipleri, hesabını aktifleştirebiliyor.  
  
ChitChat, listesinde kayıtlı olan kişilere internet üzerinden direkt mesaj atılabiliyor. Mesaj atılacak olan kişi ChitChat listesinden belirlenerek sohbet sayfası başlatılıyor. İstenilen dosya veya mesaj internet sayesinde gönderilebiliyor.

1. **Proje Detaylı Anlatımı**

* **Sign Up Design**

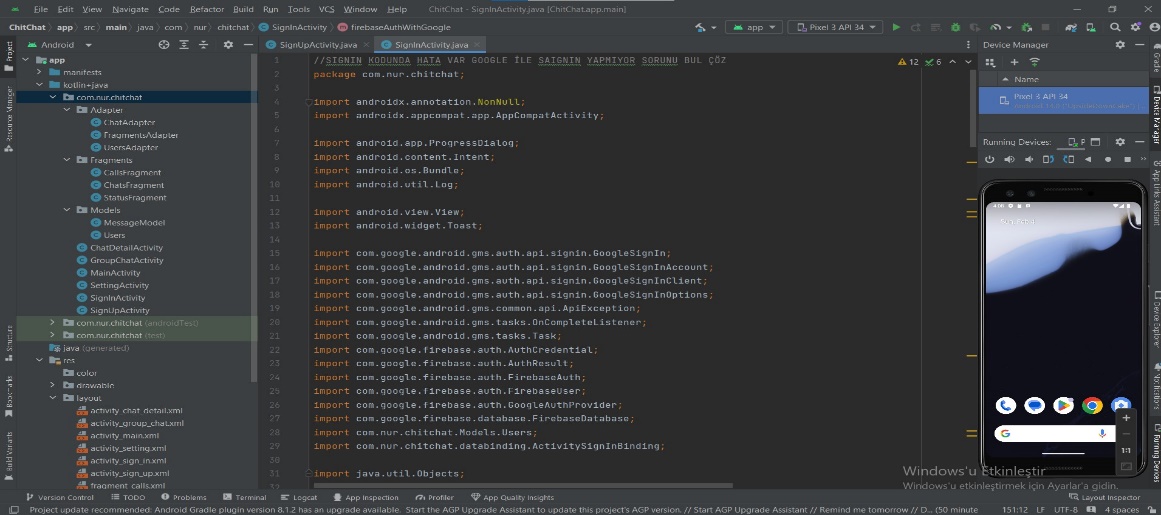
**Şekil 1.** Sign Up Kodları

Sign up başlığı içerisindeki kodlar Android uygulamasında bir kullanıcının kaydolmasını sağlayan bir aktiviteyi (activity) tanımlar. Aşamaları;

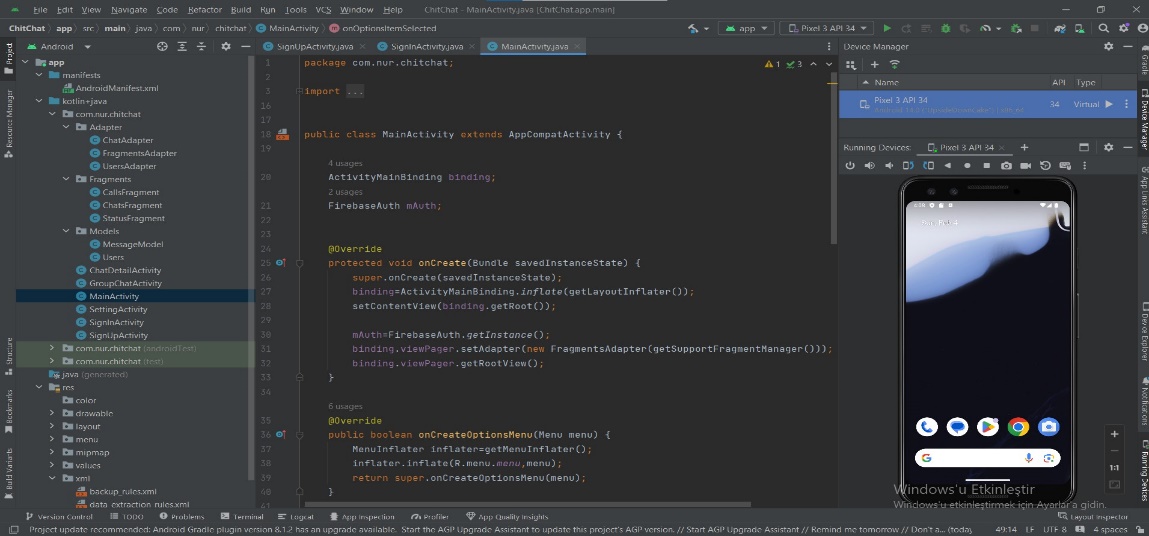
* İlk olarak, gerekli paketler ve sınıflar içe aktarılır (import ifadeleri).
* SignUpActivity sınıfı AppCompatActivity sınıfından türetilir. Bu sınıf, bir kullanıcının kaydolmasını sağlayacak işlevselliği içerir.
* ActivitySignUpBinding sınıfından binding adında bir nesne oluşturulur. Bu nesne, XML dosyasındaki arayüz öğelerine erişimi sağlar.
* FirebaseAuth ve FirebaseDatabase sınıflarından nesneler oluşturulur. Bu nesneler, Firebase Authentication ve Firebase Realtime Database'i kullanarak kullanıcı hesaplarını yönetmek ve veritabanında veri depolamak için kullanılır.
* ProgressDialog sınıfından progressDialog adında bir nesne oluşturulur. Bu nesne, kullanıcı kaydı işlemi sırasında ilerleme durumunu göstermek için kullanılır.
* onCreate() yöntemi, aktivite oluşturulduğunda çağrılır. Bu yöntemde, aktivitenin görünümü ayarlanır, Firebase öğeleri başlatılır ve kullanıcının arayüz etkileşimleri tanımlanır.
* Kullanıcı adı, e-posta ve şifre alanlarının boş olmadığı kontrol edilir. Eğer boş değillerse, kullanıcıyı kaydetme işlemi başlatılır.
* createUserWithEmailAndPassword() yöntemi kullanılarak Firebase Authentication üzerinde yeni bir kullanıcı oluşturulur. Kullanıcı başarıyla oluşturulursa, Users sınıfından bir nesne oluşturulur ve bu kullanıcı bilgileri Firebase Realtime Database'e kaydedilir.
* Kayıt işlemi başarılıysa, kullanıcıya bir bildirim gösterilir. Aksi halde, bir hata bildirimi gösterilir.
* Kullanıcı zaten bir hesaba sahipse, "Already Have an Account?" metni üzerine tıklanınca giriş yapma aktivitesine yönlendirilir.

Bu kod, kullanıcı kaydı işlemi gerçekleştiren bir Android uygulamasının önemli bir bölümünü içerir.

* **Sign In Design**

** Şekil 2.** Sign In Kodları

* Bu kod, bir Android uygulamasında kullanıcıların oturum açmasını sağlayan bir aktiviteyi (activity) tanımlar. Aşamaları:
* Gerekli paketler ve sınıflar içe aktarılır (import ifadeleri).
* SignInActivity sınıfı, AppCompactActivity sınıfından türetilir. Bu sınıf, kullanıcı oturum açma işlevselliğini içerir.
* ActivitySignInBinding sınıfından binding adında bir nesne oluşturulur. Bu nesne, XML dosyasındaki arayüz öğelerine erişimi sağlar.
* Firebase Authentication ve Firebase Realtime Database kullanımını başlatmak için gerekli nesneler oluşturulur.
* Oturum açma işlemi sırasında kullanıcıya ilerleme durumunu göstermek için bir ProgressDialog oluşturulur.
* Google hesabıyla oturum açma için gerekli yapılandırmalar (GoogleSignInOptions) yapılır ve Google ile oturum açma istemcisi oluşturulur (mGoogleSignInClient).
* Kullanıcı e-posta ve şifre bilgilerini girerek giriş yapabilir. Giriş düğmesine (btnSignIn) tıklandığında, kullanıcı Firebase Authentication üzerinden doğrulanır ve eğer başarılıysa ana aktiviteye yönlendirilir.
* Eğer kullanıcı zaten oturum açmışsa, yani ‘mAuthgetCurrentUser ()’değeri null değilse, kullanıcı otomatik olarak ana aktiviteye yönlendirilir.
* "Click to Sign Up" metnine tıklandığında, kullanıcı kaydolma aktivitesine (SignUpActivity) yönlendirilir.
* "Google ile Oturum Aç" düğmesine (btnGoogle) tıklandığında, kullanıcı Google hesabıyla oturum açma işlemine başlatılır (signIn ()yöntemi).
* Oturum açma işlemi sonucunda kullanıcı Firebase Authentication ile doğrulanır. Eğer başarılıysa, kullanıcının Firebase veritabanında kaydedilmesi ve ana aktiviteye yönlendirilmesi gerçekleştirilir.
* Bu kod, kullanıcı oturum açma işlemlerini gerçekleştiren ve gerekli durumları işleyen Android uygulamasının aşamalarını içerir.
* **Connect App to Firebase**

****

**Şekil 3.** Uygulamayı Veri Tabanına bağlama

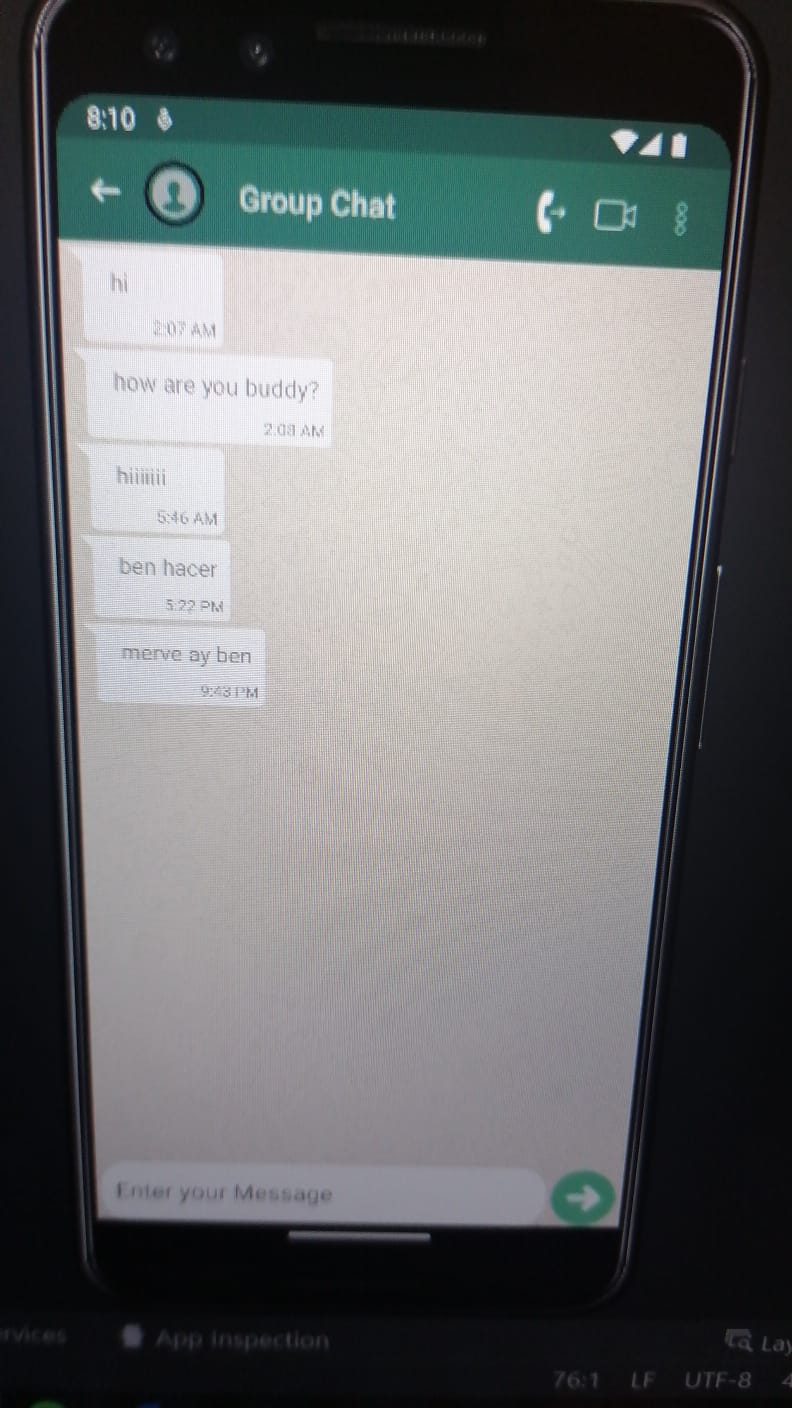
Bu kod, Android uygulamasında ana aktiviteyi (MainActivity) tanımlar. Bu aktivite, uygulamanın ana ekranını ve kullanıcı arayüzünü yönetir.

* MainActivity sınıfı, AppCompatActivity sınıfından türetilir ve uygulamanın ana ekranını temsil eder.
* ActivityMainBinding sınıfından binding adında bir nesne oluşturulur. Bu nesne, aktivitenin içindeki arayüz öğelerine erişimi sağlar.
* onCreate() yöntemi, aktivite oluşturulduğunda çağrılır. Bu yöntemde, aktivitenin görünümü ayarlanır ve Firebase Authentication nesnesi başlatılır. Ayrıca, fragment tabanlı bir adapter (FragmentsAdapter) kullanılarak view pager'a bağlanır.
* onCreateOptionsMenu() yöntemi, aktivitenin üst çubuğundaki menüyü oluşturur. XML dosyasında tanımlanan menü öğeleri, MenuInflater kullanılarak yüklenir.
* onOptionsItemSelected() yöntemi, kullanıcının menü öğelerine tıklamasını dinler. Kullanıcı hangi öğeye tıkladıysa, ona göre bir işlem yapılır.
* Eğer kullanıcı "Settings" (Ayarlar) öğesine tıklarsa, SettingActivity'e yönlendirilir.
* Eğer kullanıcı "Group Chat" (Grup Sohbeti) öğesine tıklarsa, GroupChatActivity'e yönlendirilir.
* Eğer kullanıcı "Logout" (Çıkış Yap) öğesine tıklarsa, Firebase Authentication üzerinden çıkış yapılır ve kullanıcı tekrar giriş ekranına yönlendirilir.
* super.onOptionsItemSelected(item) çağrısı, menü öğesinin varsayılan davranışını gerçekleştirir.
* Bu kod, kullanıcı arayüzü etkileşimlerini dinleyen ve belirli işlemleri gerçekleştiren bir Android uygulamasının ana aktivitesini temsil eder.
* **Group Chat Activity**

Bu kod, Android uygulamasında grup sohbetini yöneten bir aktiviteyi (activity) tanımlar.

* GroupChatActivity sınıfı, AppCompatActivity sınıfından türetilir ve grup sohbeti işlevselliğini içerir.
* ActivityGroupChatBinding sınıfından binding adında bir nesne oluşturulur. Bu nesne, aktivitenin içindeki arayüz öğelerine erişimi sağlar.
* onCreate() yöntemi, aktivite oluşturulduğunda çağrılır. Bu yöntemde, aktivitenin görünümü ayarlanır, Firebase'den gelen mesajları görüntülemek için bir RecyclerView ve bir dizi diğer görünüm öğesi yerleştirilir.
* Geri dönüş oku (backArrow) tıklanırsa, kullanıcı ana ekrana yönlendirilir (MainActivity).
* Firebase veritabanı oluşturulur ve mesaj modellerini saklamak için bir dizi oluşturulur.
* Kullanıcının Firebase Authentication ile kimliği alınır ve kullanıcının adı "Group Chat" olarak ayarlanır.
* Bir ChatAdapter oluşturulur ve RecyclerView'e bu adaptör atanır.
* Veritabanından "Group Chat" altındaki veriler dinlenir ve her veri değişikliğinde veri modeli güncellenir.
* Kullanıcı bir mesaj göndermek için "Send" düğmesine tıkladığında, mesaj alınır, bir mesaj modeli oluşturulur ve Firebase veritabanına gönderilir.
* Mesaj gönderildikten sonra, kullanıcıya bir bildirim gösterilir.

Bu kod, bir grup sohbeti uygulamasında kullanıcıların mesaj göndermesini, bu mesajları görmesini ve Firebase veri tabanında saklamasını sağlar.

****

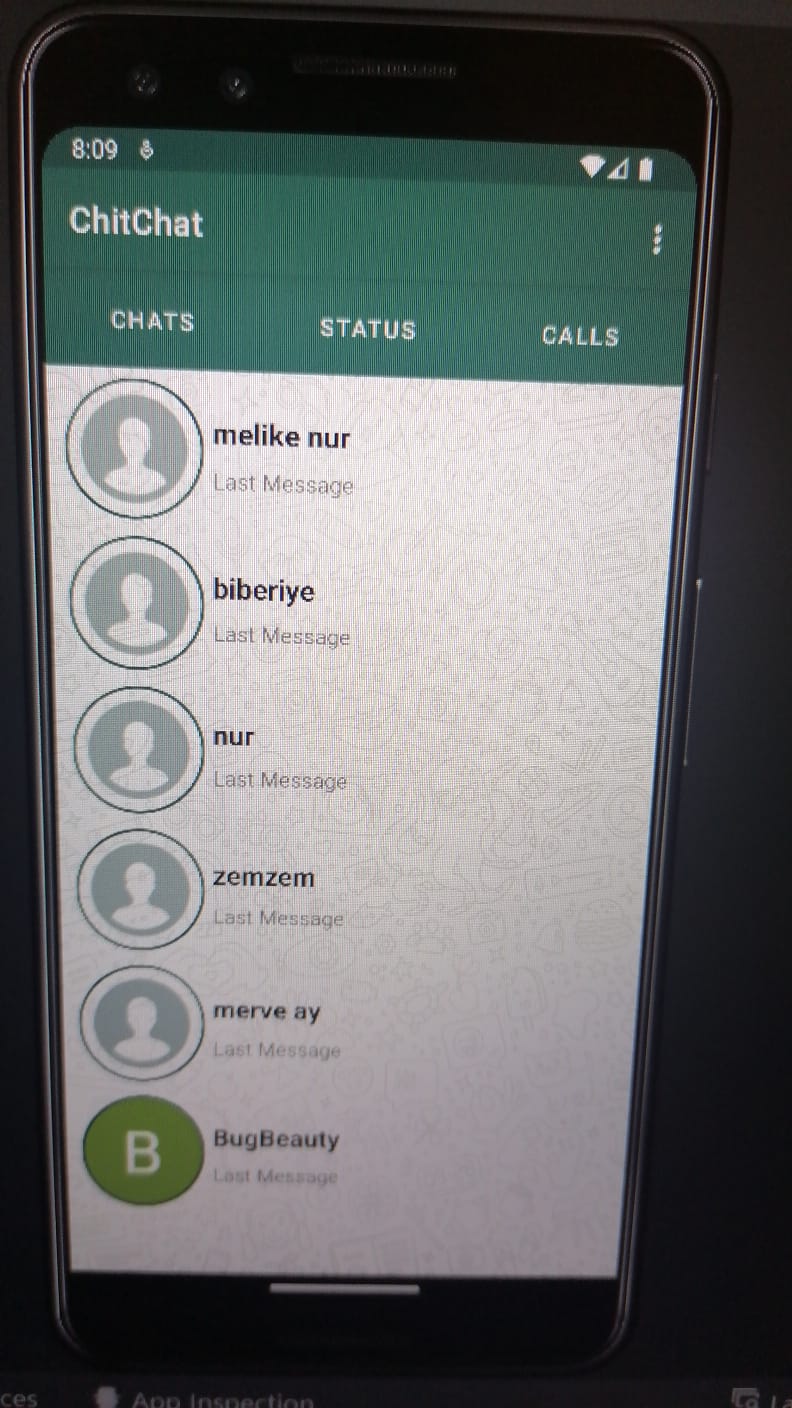
**Şekil 4.** Grup Chat

* **Chat Detail Activity**

Bu kod, bir Android uygulamasında bir kişiyle olan özel sohbetin ayrıntılarını gösteren bir aktiviteyi (activity) tanımlar.

* ChatDetailActivity sınıfı, AppCompatActivity sınıfından türetilir ve özel sohbetin ayrıntılarını içerir.
* ActivityChatDetailBinding sınıfından binding adında bir nesne oluşturulur. Bu nesne, aktivitenin içindeki arayüz öğelerine erişimi sağlar.
* onCreate() yöntemi, aktivite oluşturulduğunda çağrılır. Bu yöntemde, aktivitenin görünümü ayarlanır, Firebase veritabanına ve Firebase Authentication'a erişim için gerekli nesneler başlatılır.
* Intent aracılığıyla alınan alıcı kullanıcının ID'si (recieverId), adı (userName) ve profil resmi (profilePic) alınır ve görselleştirilir.
* Geri dönüş oku (backArrow) tıklanırsa, kullanıcı ana ekrana (MainActivity) yönlendirilir.
* Mesajları saklamak için bir dizi MessageModel oluşturulur ve bu mesajları göstermek için bir ChatAdapter oluşturulur.
* Mesajları göstermek için RecyclerView kullanılır. Veritabanından alınan mesajlar (chats) dinlenir ve alınan her mesaj messageModels dizisine eklenir.
* Kullanıcı mesaj göndermek için "Send" düğmesine tıkladığında, girilen mesaj alınır, bir mesaj modeli oluşturulur ve gönderilen ve alıcı odalarına bu mesajlar eklenir. Bu şekilde, gönderen ve alıcı arasında bir karşılıklı sohbet oluşturulmuş olur.

Bu kod, kullanıcıların özel bir kişiyle sohbet etmelerini, mesaj göndermelerini ve bu mesajları saklamak için Firebase veritabanını kullanan bir Android uygulamasının bir parçasını oluşturur.

****

**Şekil 5.** Chat

* **Setting Activity**

Bu kod, bir Android uygulamasında kullanıcı profili ayarlarının yapıldığı bir etkinlik (activity) tanımlar.

* SettingActivity sınıfı, AppCompatActivity sınıfından türetilir ve kullanıcı profiliyle ilgili ayarları içerir.
* ActivitySettingBinding sınıfından binding adında bir nesne oluşturulur. Bu nesne, etkinliğin içindeki arayüz öğelerine erişimi sağlar.
* onCreate() yöntemi, etkinlik oluşturulduğunda çağrılır. Bu yöntemde, etkinliğin görünümü ayarlanır ve Firebase'e erişim için gerekli nesneler başlatılır.
* Geri dönüş oku (backArrow) tıklanırsa, kullanıcı ana ekrana (MainActivity) yönlendirilir.
* Kullanıcı profili güncelleme butonuna tıklandığında, kullanıcının adını ve durumunu günceller ve Firebase veritabanında saklar.
* Kullanıcının mevcut profil bilgileri Firebase veritabanından alınır ve ilgili alanlara yerleştirilir.
* Profil fotoğrafını değiştirmek için "plus" simgesine tıklandığında, galeriye erişim sağlanır ve yeni profil fotoğrafı seçilir. Seçilen fotoğraf Firebase Storage'a yüklenir ve yükleme başarılı olduğunda Firebase veritabanında güncellenir.
* Diğer metin öğelerine tıklandığında sadece bir Toast mesajı görüntülenir, bunlar genellikle yardım, gizlilik politikası vb. seçenekler içindir.

Bu kod, Android uygulamasında kullanıcı profiliyle ilgili ayarların yapıldığı bir etkinliği temsil eder.

* **Message Model**

Bu kod, bir mesajı temsil eden MessageModel adında bir Java sınıfını tanımlar.

1.Değişkenler (Variables):

* uId: Mesajı gönderen kullanıcının benzersiz kimlik bilgisini (ID) tutar.
* message: Gönderilen mesajın içeriğini tutar.
* messageId: Mesajın benzersiz kimlik bilgisini (ID) tutar.
* timestamp: Mesajın gönderilme zamanını temsil eden bir zaman damgasını (timestamp) tutar.

2.Kurucu Metotlar (Constructors):

* MessageModel(String uId, String message, Long timestamp): Kullanıcı ID'si, mesaj içeriği ve zaman damgasıyla bir mesaj modeli oluşturur.
* MessageModel(String uId, String message): Kullanıcı ID'si ve mesaj içeriğiyle bir mesaj modeli oluşturur.
* MessageModel(): Boş bir kurucu metot.

3.Getter ve Setter Metotlar:

* getuId() / setuId(): uId değişkeninin değerini almak ve ayarlamak için kullanılır.
* getMessage() / setMessage(): message değişkeninin değerini almak ve ayarlamak için kullanılır.
* getMessageId() / setMessageId(): messageId değişkeninin değerini almak ve ayarlamak için kullanılır.
* getTimestamp() / setTimestamp(): timestamp değişkeninin değerini almak ve ayarlamak için kullanılır.

Bu sınıf, bir mesajın özelliklerini tutmak için kullanılır. Mesaj gönderenin ID'si, mesaj içeriği, mesajın kendine özgü bir ID'si ve gönderilme zamanı gibi temel bilgileri içerir. Bu bilgiler, mesajları saklamak, almak ve işlemek için kullanılan bir veri yapısı oluşturmak için kullanılır.

* **Users**

Bu kod, kullanıcı bilgilerini temsil eden Users adında bir Java sınıfını tanımlar

1.Değişkenler (Variables):

* profilePic: Kullanıcının profil resminin URL'sini tutar.
* userName: Kullanıcının kullanıcı adını (ismini) tutar.
* mail: Kullanıcının e-posta adresini tutar.
* password: Kullanıcının şifresini tutar.
* userId: Kullanıcının benzersiz kimlik bilgisini (ID) tutar.
* lastMessage: Kullanıcının son iletişimdeki son mesajını tutar.
* status: Kullanıcının durumunu (online, çevrimdışı, meşgul vb.) tutar.

2.Kurucu Metotlar (Constructors):

* Users(): Boş bir kurucu metot.
* Users(String profilePic, String userName, String mail, String password, String userId, String lastMessage, String status): Tüm kullanıcı bilgilerini alan bir kurucu metot.
* Users(String userName, String mail, String password): Sadece kullanıcı adı, e-posta ve şifre bilgilerini alan bir kurucu metot.

3.Getter ve Setter Metotlar:

* getProfilePic() / setProfilePic(): Kullanıcının profil resmi URL'sini almak ve ayarlamak için kullanılır.
* getUserName() / setUserName(): Kullanıcının kullanıcı adını almak ve ayarlamak için kullanılır.
* getMail() / setMail(): Kullanıcının e-posta adresini almak ve ayarlamak için kullanılır.
* getPassword() / setPassword(): Kullanıcının şifresini almak ve ayarlamak için kullanılır.
* getUserId() / setUserId(): Kullanıcının benzersiz kimlik bilgisini almak ve ayarlamak için kullanılır.
* getLastMessage() / setLastMessage(): Kullanıcının son iletişimdeki son mesajını almak ve ayarlamak için kullanılır.
* getStatus() / setStatus(): Kullanıcının durumunu almak ve ayarlamak için kullanılır.

Bu sınıf, bir kullanıcının temel bilgilerini saklamak için kullanılır. Kullanıcının profil resmi, kullanıcı adı, e-posta adresi, şifresi, benzersiz kimlik bilgisi ve diğer bilgileri gibi özellikleri içerir. Bu bilgiler, kullanıcı hesaplarını yönetmek ve kullanıcılar arasında iletişimi kolaylaştırmak için kullanılır.

* **Fragments**

1. **Calls Fragments**

Bu kod, bir Fragment alt sınıfı olan CallsFragment sınıfını tanımlar.

1.Değişkenler (Variables):

* mParam1 ve mParam2: Fragment'a parametre olarak iletilen değerleri tutar. Bu örnek uygulamada, kullanılmadığı için bu parametreler boş bırakılabilir.

2.Kurucu Metot (Constructor):

* CallsFragment(): Boş bir kurucu metottur. Fragment oluşturulurken bu kurucu metot çağrılır.

3.newInstance Metodu:

* newInstance(String param1, String param2): Yeni bir CallsFragment örneği oluşturur ve parametrelerle birlikte döndürür. Bu metod, fragment oluşturulurken parametrelerin iletilmesini sağlar.

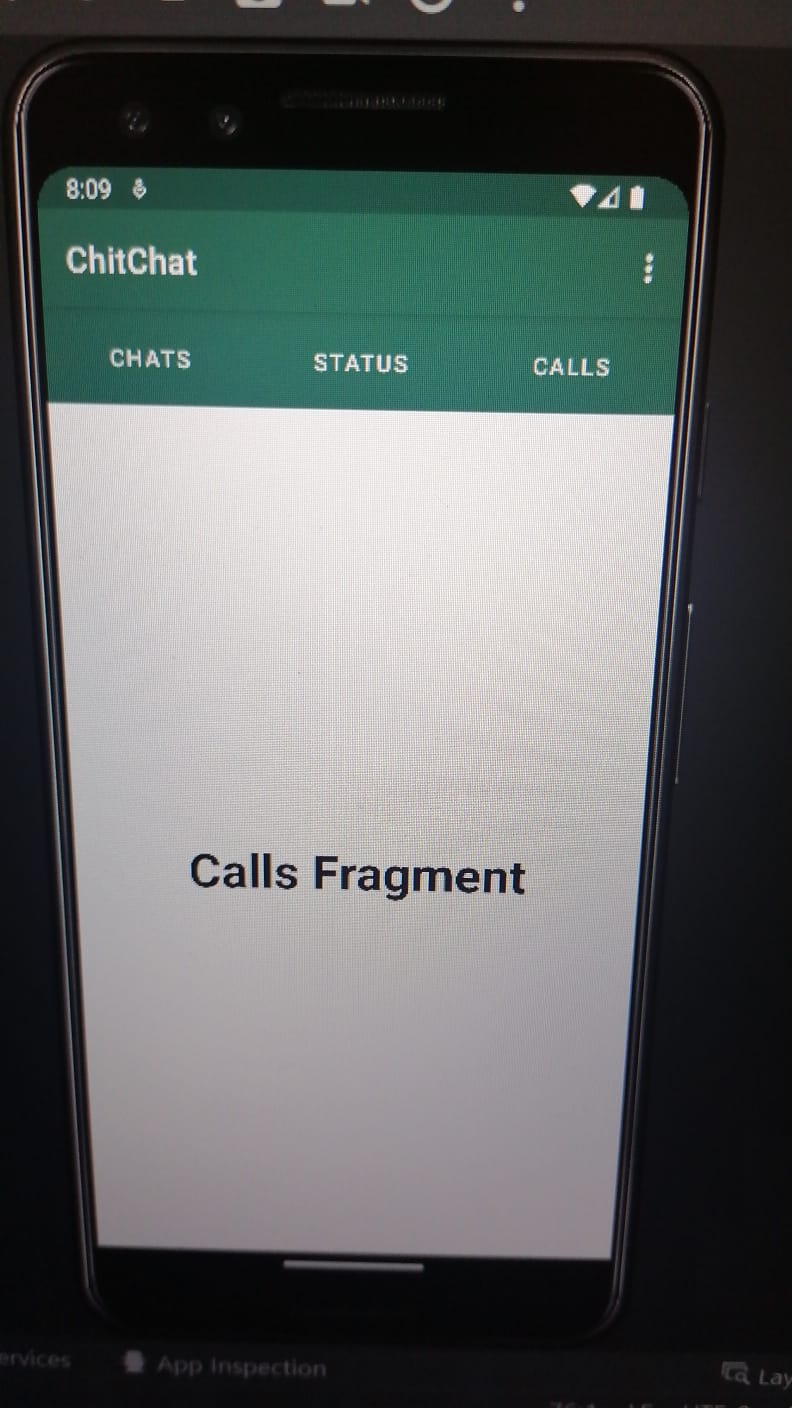
4. onCreate Metodu:

* onCreate(Bundle savedInstanceState): Fragment oluşturulduğunda çağrılan metottur. Eğer fragment'a parametreler iletilmişse, bu metot bu parametreleri alır.

5.onCreateView Metodu:

* onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState): Fragment'ın kullanıcı arayüzünü oluşturmak için çağrılan metottur. Bu metot, fragment'ın XML dosyasını (layout) yükler ve kullanıcı arayüzünü oluşturur.

Bu CallsFragment sınıfı, uygulamanın çağrılar sekmesini temsil eder. Fragment, kullanıcı arayüzünün bir parçası olarak kullanılır ve çağrılarla ilgili içeriği gösterir. Bu sınıf, fragment oluşturmak için gerekli standart yapıları içerir ve fragment'ın yaşam döngüsünü yönetmek için gerekli metotları uygular.

****

**Şekil 6.** Calls Fragments

Formun Üstü

**b)Chats Fragments**

Bu kod bir Fragment alt sınıfı olan ChatsFragment sınıfını tanımlar. Bu sınıf, kullanıcıların sohbet edebileceği diğer kullanıcıları listelemek için kullanılır.

1.Değişkenler (Variables):

* binding: Fragment'ın bağlantı noktasını (binding) temsil eder. Kullanıcı arayüzünün öğelerine erişim sağlar.
* list: Kullanıcıları depolamak için bir liste. Users sınıfından örnekler içerir.
* database: Firebase veritabanı bağlantısını temsil eder. Verilerin alınması için kullanılır.

2.Kurucu Metot (Constructor):

* ChatsFragment(): Boş bir kurucu metottur. Fragment oluşturulurken bu kurucu metot çağrılır.

3.onCreateView Metodu:

* onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState): Fragment'ın kullanıcı arayüzünü oluşturmak için çağrılan metottur. Bu metot, fragment'ın XML dosyasını (layout) yükler ve kullanıcı arayüzünü oluşturur.

4.onDataChange Metodu:

* onDataChange(@NonNull DataSnapshot snapshot): Firebase veritabanında yapılan değişikliklerde tetiklenen olaydır. Kullanıcılar veritabanındaki değişiklikleri dinler ve listeyi günceller.

5.onCancelled Metodu:

* onCancelled(@NonNull DatabaseError error): Veritabanı işleminde bir hata meydana geldiğinde tetiklenen olaydır. Hata durumunda geri dönüş yapar.

Bu ChatsFragment sınıfı, kullanıcıların sohbet edebileceği diğer kullanıcıları listeleyen bir fragmenttır. Firebase veritabanından kullanıcıları çeker ve bir UsersAdapter kullanarak bu kullanıcıları RecyclerView üzerinde listeleyerek gösterir. Kullanıcılar listesi, veritabanındaki değişikliklere dinamik olarak tepki verir ve güncellenir. Bu sayede kullanıcılar, sohbet etmek istedikleri diğer kullanıcıları görüntüleyebilir ve seçebilir.

**c)Status Fragments**

Bu kod bir Fragment alt sınıfı olan StatusFragment sınıfını tanımlar. Bu sınıf, kullanıcıların durumlarını gösterecek bir fragmentı temsil eder. Şimdi bu sınıfın yapısını ve işlevlerini Türkçe olarak açıklayalım:

1.Değişkenler (Variables):

* mParam1 ve mParam2: Fragment'ın oluşturulurken alabileceği parametreler.

2.Kurucu Metot (Constructor):

* StatusFragment(): Boş bir kurucu metottur. Fragment oluşturulurken bu kurucu metot çağrılır.

3.newInstance Metodu:

* newInstance(String param1, String param2): Yeni bir StatusFragment örneği oluşturmak için kullanılan bir fabrika metodu. Parametreleri alır ve bu parametrelerle yeni bir fragment oluşturur.

4.onCreate Metodu:

* onCreate(Bundle savedInstanceState): Fragment oluşturulduğunda çağrılan metottur. Fragment'ın oluşturulması için gerekli başlangıç işlemlerini yapar.

5.onCreateView Metodu:

* onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState): Fragment'ın kullanıcı arayüzünü oluşturmak için çağrılan metottur. Bu metot, fragment'ın XML dosyasını (layout) yükler ve kullanıcı arayüzünü oluşturur.

Bu StatusFragment sınıfı, kullanıcıların durumlarını göstermek için kullanılır. onCreateView metodunda belirtilen layout dosyası, fragment'ın görünümünü oluşturur. newInstance metodu, fragment'ın yeni bir örneğini oluştururken parametrelerin iletilmesini sağlar. Bu sayede fragment'lar arasında veri aktarımı kolaylaşır.

* **Adapter**

1. **Chat Adapter**

Bu kod, bir RecyclerView için özel bir adaptör olan ChatAdapter sınıfını tanımlar. Bu adaptör, sohbet mesajlarını görüntülemek için kullanılır.

1.Değişkenler (Variables):

* messageModels: Mesajların bulunduğu bir liste.
* context: Adaptörün bağlı olduğu bağlam (context).
* recId: Alıcı kullanıcının kimliği.
* SENDER\_VIEW\_TYPE ve RECIEVER\_VIEW\_TYPE: Görünüm türlerini belirlemek için sabitler.

2.Kurucu Metotlar (Constructors):

* ChatAdapter(ArrayList<MessageModel> messageModels, Context context): Mesaj modelleri ve bağlam bilgisiyle bir adaptör oluşturur.
* ChatAdapter(ArrayList<MessageModel> messageModels, Context context, String recId): Alıcı kimliği bilgisiyle bir adaptör oluşturur.

3.onCreateViewHolder Metodu:

* onCreateViewHolder(ViewGroup parent, int viewType): Yeni bir ViewHolder oluşturmak için çağrılan metottur. Görünüm türüne (alıcı veya gönderici) göre farklı görünüm düzenlerini yükler.

4.getItemViewType Metodu:

* getItemViewType(int position): Belirli bir pozisyondaki öğenin görünüm türünü döndürür. Gönderen veya alıcı mesajına bağlı olarak farklı türler döndürür.

5.onBindViewHolder Metodu:

* onBindViewHolder(RecyclerView.ViewHolder holder, int position): ViewHolder'ın belirli bir pozisyondaki öğe verileriyle doldurulmasını sağlar. Ayrıca, uzun tıklama olayları için bir dinleyici ekler ve silme işlemi için bir iletişim kutusu görüntüler.

6.getItemCount Metodu:

* getItemCount(): Adaptördeki öğe sayısını döndürür.

7.Inner ViewHolder Sınıfları:

* RecieverViewHolder: Alıcı mesaj görünümünü temsil eder.
* SenderViewHolder: Gönderen mesaj görünümünü temsil eder.

Bu ChatAdapter sınıfı, RecyclerView içinde sohbet mesajlarını görüntülemek için kullanılır. Gönderen ve alıcı mesajları arasında farklı görünümler sağlar ve uzun tıklama olayları için silme işlemleri gerçekleştirir.

1. **Fragments Adapter**

Bu kod, bir FragmentPagerAdapter alt sınıfı olan FragmentsAdapter sınıfını tanımlar. Bu adaptör, bir ViewPager içinde birden çok fragment'ı yönetmek için kullanılır.

1.Kurucu Metot (Constructor):

* FragmentsAdapter(FragmentManager fm): Bir FragmentManager nesnesi alarak adaptörü başlatır.

2.getItem Metodu:

* getItem(int position): Belirli bir konum için bir fragment döndürür. Konuma bağlı olarak, ChatsFragment, StatusFragment veya CallsFragment döndürülür.

3.getCount Metodu:

* getCount(): ViewPager'daki toplam sayfa sayısını döndürür. Bu durumda, toplam sayfa sayısı 3'tür.

4.getPageTitle Metodu:

* getPageTitle(int position): Belirli bir konum için başlık metnini döndürür. Konuma bağlı olarak, "CHATS", "STATUS" veya "CALLS" başlıkları döndürülür.

Bu FragmentsAdapter sınıfı, ViewPager içindeki farklı fragmentları yönetmek için kullanılır. Her bir fragment, ViewPager'da farklı bir sekme olarak görüntülenir ve kullanıcının fragmentlar arasında gezinmesine izin verir.

1. **Users Adapter**

Bu kod bir RecyclerView için bir adaptör olan UsersAdapter sınıfını tanımlar. Bu adaptör, bir kullanıcı listesini görüntülemek için tasarlanmıştır.

1.Kurucu Metot (Constructor):

* UsersAdapter(ArrayList<Users> list, Context context): Kullanıcı listesini ve bağlamı alarak adaptörü başlatır.

2.onCreateViewHolder Metodu:

* onCreateViewHolder(ViewGroup parent, int viewType): Yeni bir ViewHolder örneği oluşturur ve onu döndürür. Bu metod, LayoutInflater ile bir görünüm oluşturur ve bu görünümü ViewHolder'a bağlar.

3.onBindViewHolder Metodu:

* onBindViewHolder(ViewHolder holder, int position): ViewHolder'ın içeriğini belirli bir konumda bulunan kullanıcı verileriyle doldurur. Kullanıcı profili resmi, kullanıcı adı ve son mesaj bilgilerini ayarlar. Ayrıca, bir kullanıcıya tıklandığında ChatDetailActivity'ye yönlendirme gerçekleştirir.

4.getItemCount Metodu:

* getItemCount(): Veri kümesindeki toplam öğe sayısını döndürür.

5.ViewHolder Sınıfı:

* ViewHolder, RecyclerView'de her bir öğe için görünüm öğelerini tutar. Bu sınıf, profile\_image adlı bir ImageView ve userNameList ile lastMessage adlı iki TextView içerir.

6.Veritabanı İşlemleri:

* FirebaseDatabase.getInstance().getReference().child("chats"): Firebase veritabanında belirli bir sohbet alanını referans alır.
* orderByChild("timestamp"): Sohbetlerin zaman damgasına göre sıralanmasını sağlar.
* limitToLast(1): Son sohbet mesajını almak için son mesajı sınırlar.
* addListenerForSingleValueEvent(): Bir kez veritabanı değeri değiştiğinde tetiklenen bir olay dinleyicisi ekler.
* onDataChange(): Veri değiştiğinde çağrılan metottur. En son mesajı ayarlar.
* onCancelled(): Veri okuma işlemi iptal edildiğinde çağrılan metottur.

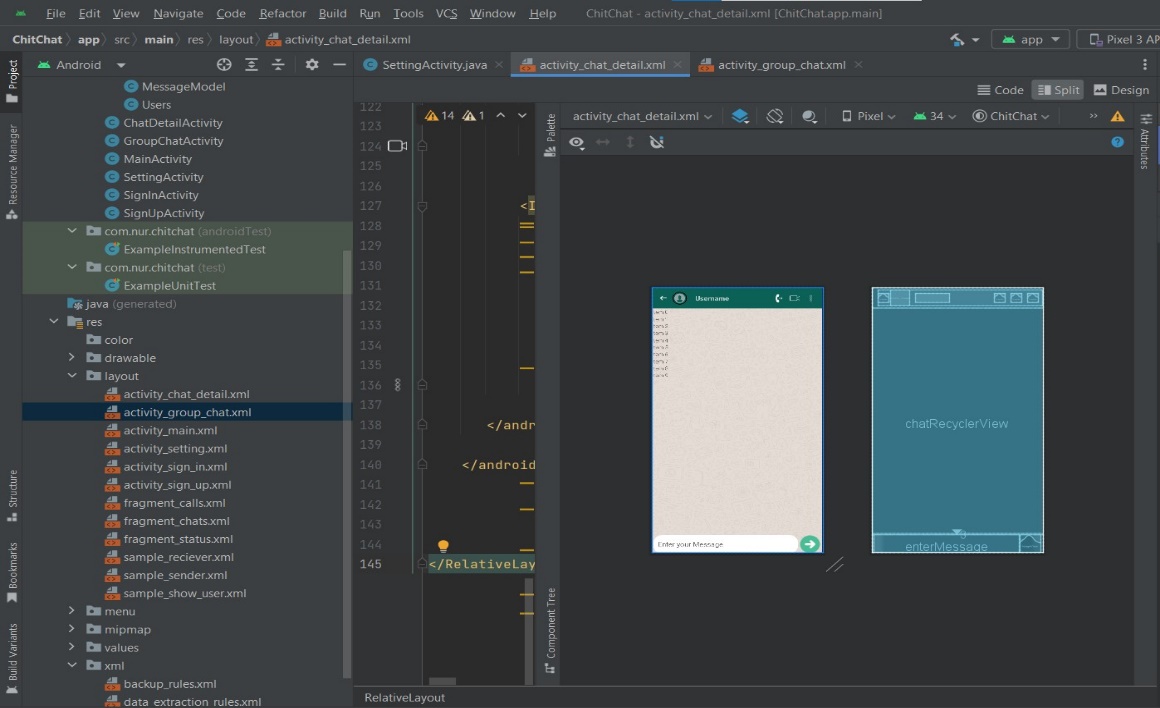
Bu adaptör, bir RecyclerView içindeki kullanıcı listesini göstermek için kullanılır ve her bir kullanıcı öğesi için resim, kullanıcı adı ve son mesajı görüntüler. Kullanıcıya tıklandığında, ilgili kullanıcıyla iletişim kurmak için ChatDetailActivity'ye yönlendirme gerçekleşir.

* **Layout**
* **Chat Detail**

Bu XML dosyası, ChatDetailActivity için bir kullanıcı arayüzü tasarımını tanımlar. Aşağıdaki öğeleri içerir:

* RelativeLayout: Diğer görünüm öğelerini düzenlemek için kullanılan bir düzen yöneticisidir. Öğeler, birbiriyle ilişkili olarak yerleştirilir.
* android:background: Arka plan görüntüsünü tanımlar.
* androidx.recyclerview.widget.RecyclerView: Sohbet mesajlarının listelendiği RecyclerView öğesidir. chatRecyclerView id'siyle tanımlanmıştır.
* LinearLayout: Mesaj yazma alanını ve gönderme düğmesini içeren yatay bir düzendir. linear id'siyle tanımlanmıştır.
* EditText: Kullanıcının mesajlarını girebileceği metin girişi alanıdır. enterMessage id'siyle tanımlanmıştır.
* ImageView: Mesajı göndermek için bir gönderme düğmesidir. send id'siyle tanımlanmıştır.
* androidx.appcompat.widget.Toolbar: Aktivitenin üst kısmındaki araç çubuğunu tanımlar. İçerisinde kullanıcı profili resmi, kullanıcı adı ve çağrı yapma düğmeleri bulunmaktadır.
* de.hdodenhof.circleimageview.CircleImageView: Kullanıcının profili için daire şeklinde bir resim görüntüsüdür. profile\_image id'siyle tanımlanmıştır.
* TextView: Kullanıcının adını gösteren metin alanıdır. userName id'siyle tanımlanmıştır.
* ImageView: Kullanıcıya telefon çağrısı yapma seçeneği sunar. imageView4 id'siyle tanımlanmıştır.
* ImageView: Kullanıcıya video çağrısı yapma seçeneği sunar. imageView5 id'siyle tanımlanmıştır.
* ImageView: Diğer seçenekleri göstermek için bir menü simgesi. imageView6 id'siyle tanımlanmıştır.

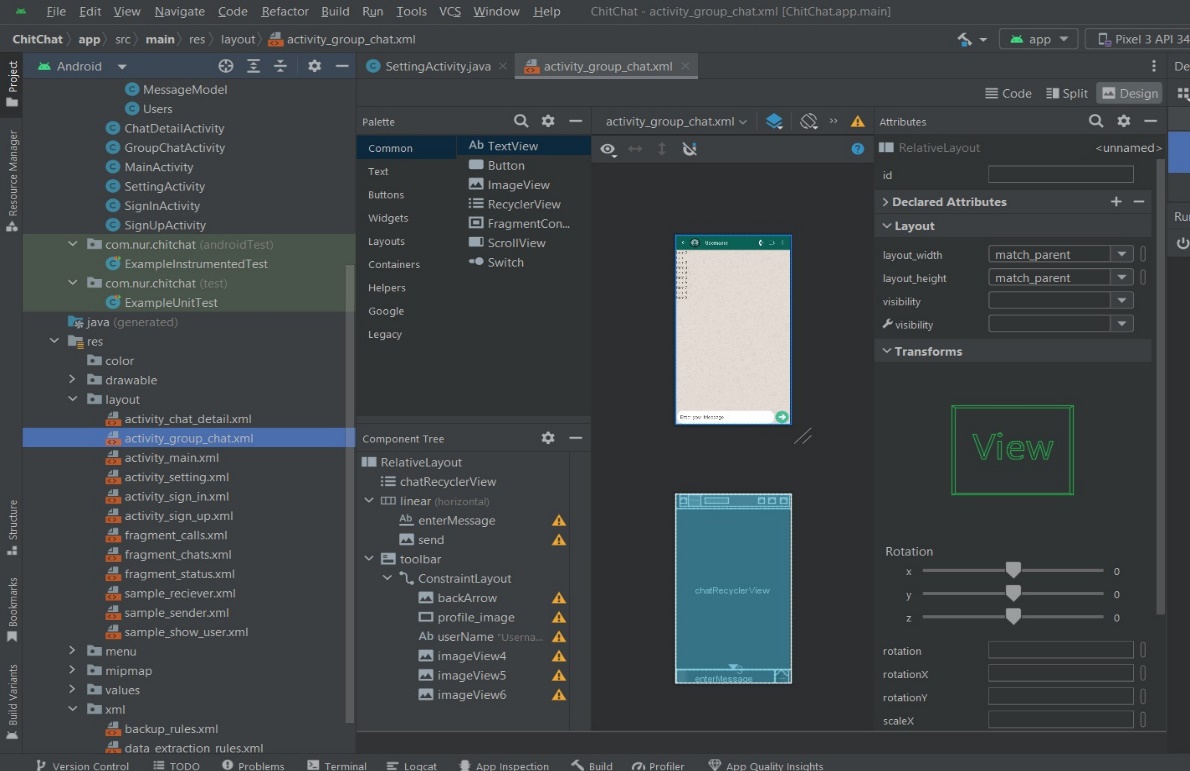
Bu tasarım, kullanıcıların sohbet detaylarını görüntüleyebileceği ve yeni mesajlar gönderebileceği bir arayüz sağlar. RecyclerView, sohbet mesajlarını dinamik olarak gösterirken, Toolbar kullanıcı profili ve çağrı seçeneklerini sunar.

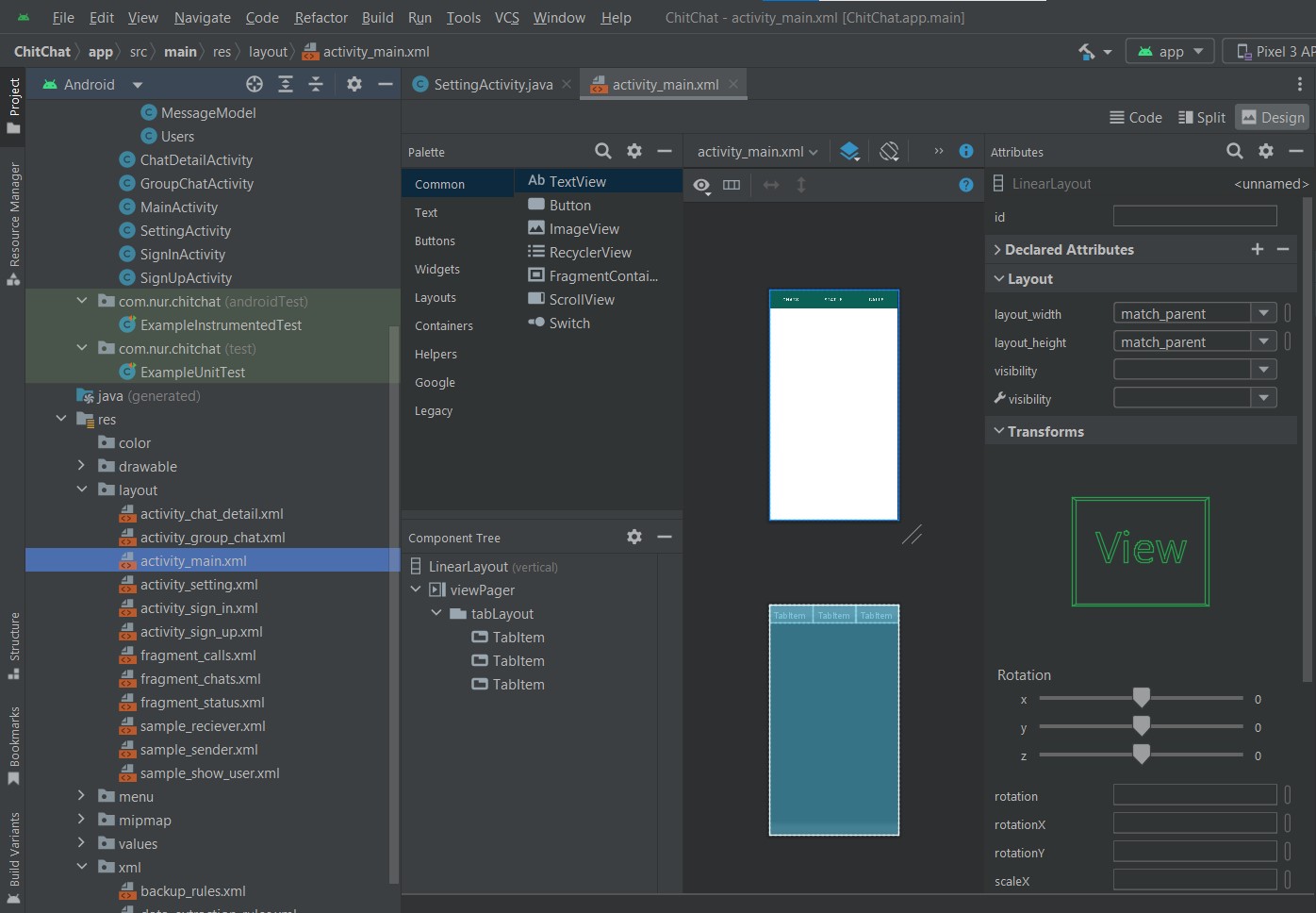


Formun Üstü

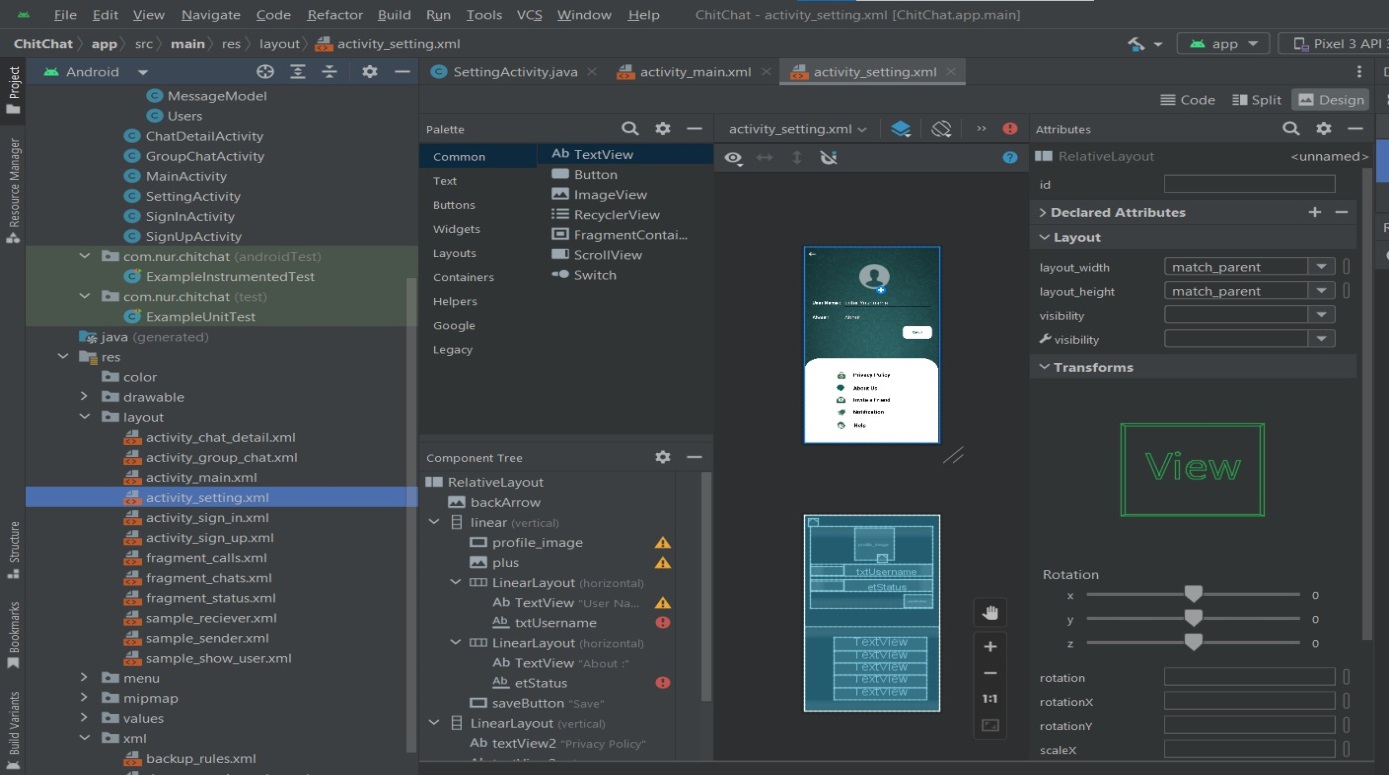
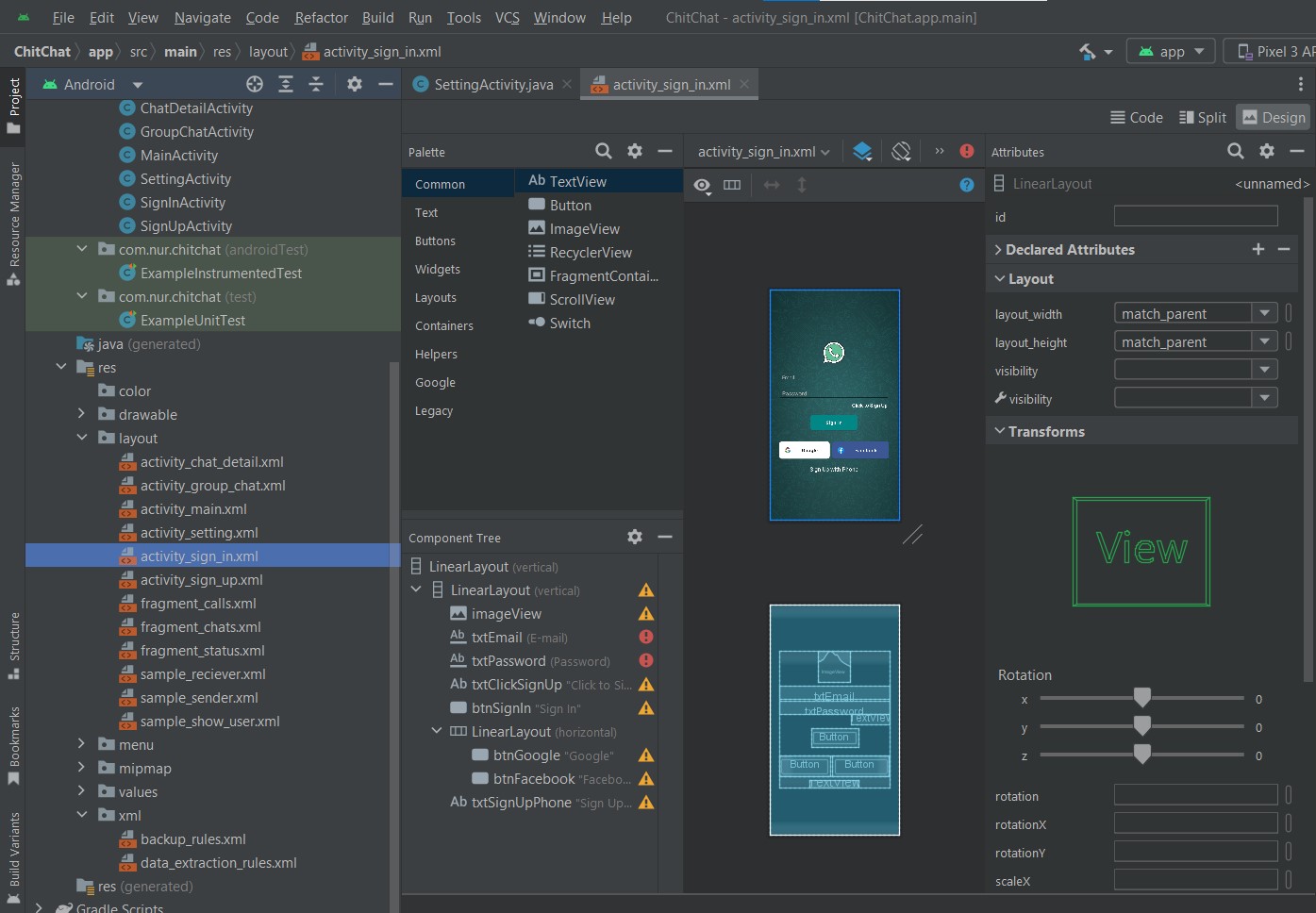
**Şekil 7.**

* **Activity Group**

****

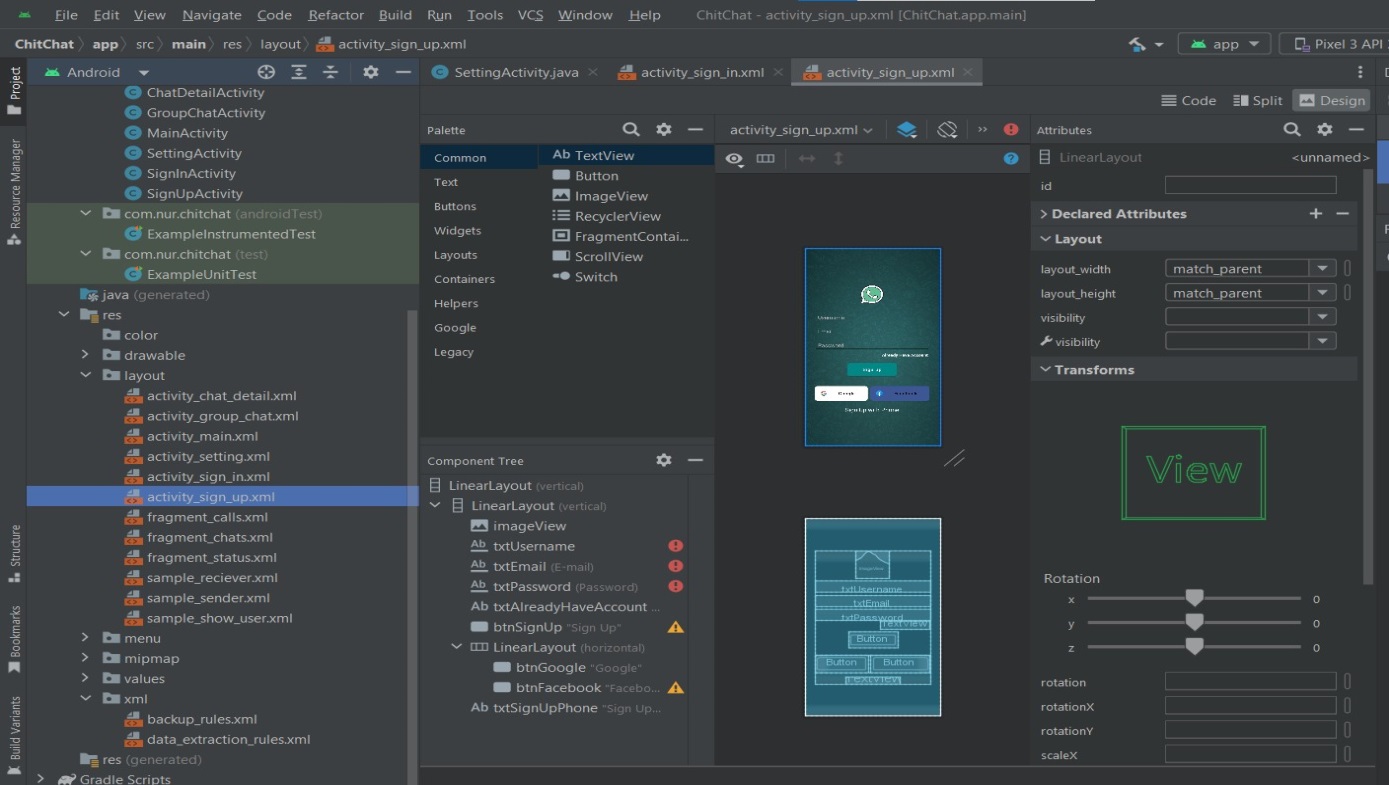
* **Activity Main**

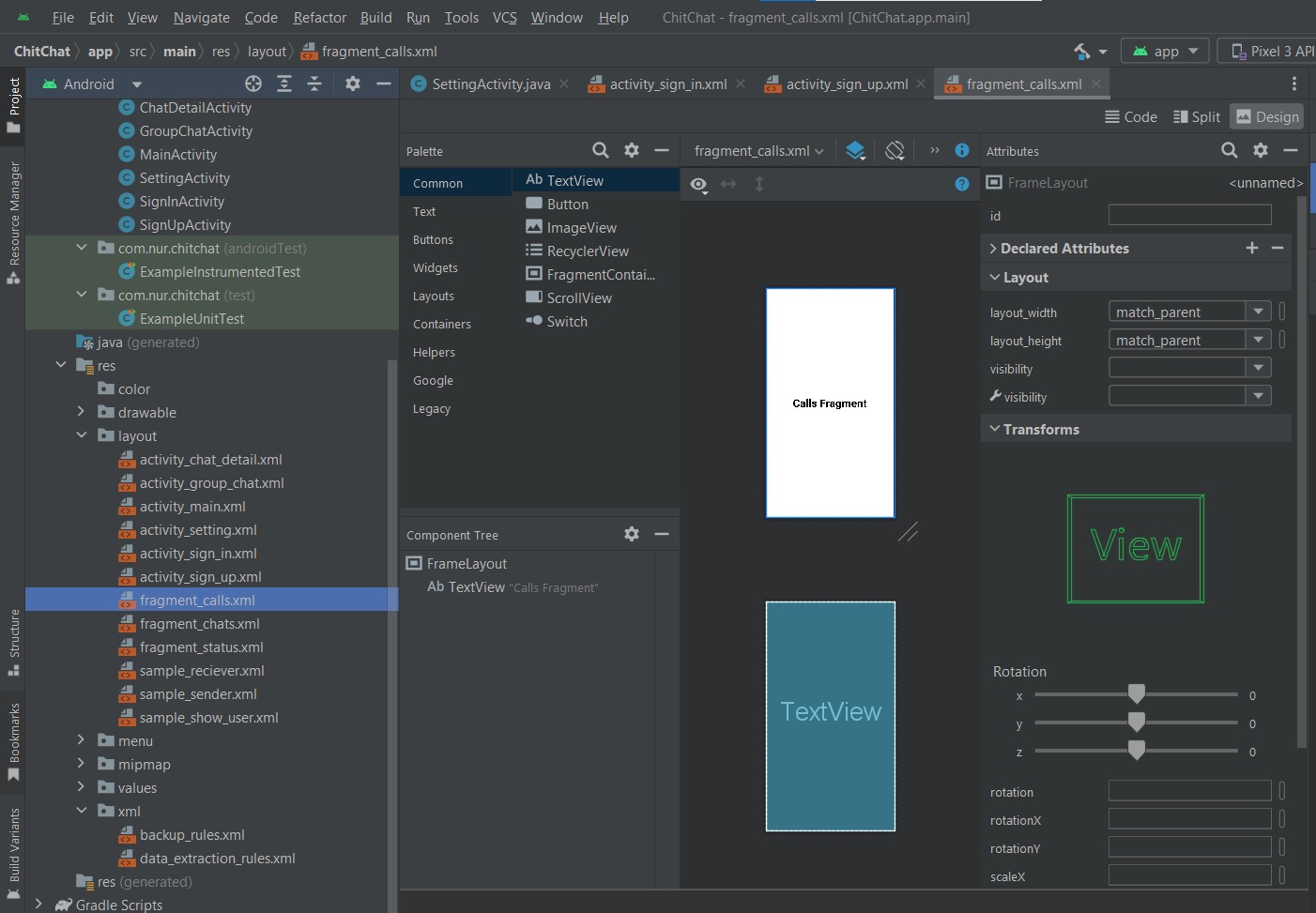
**Şekil 9.**

* **Activity Setting**
* **Activity Sign In**

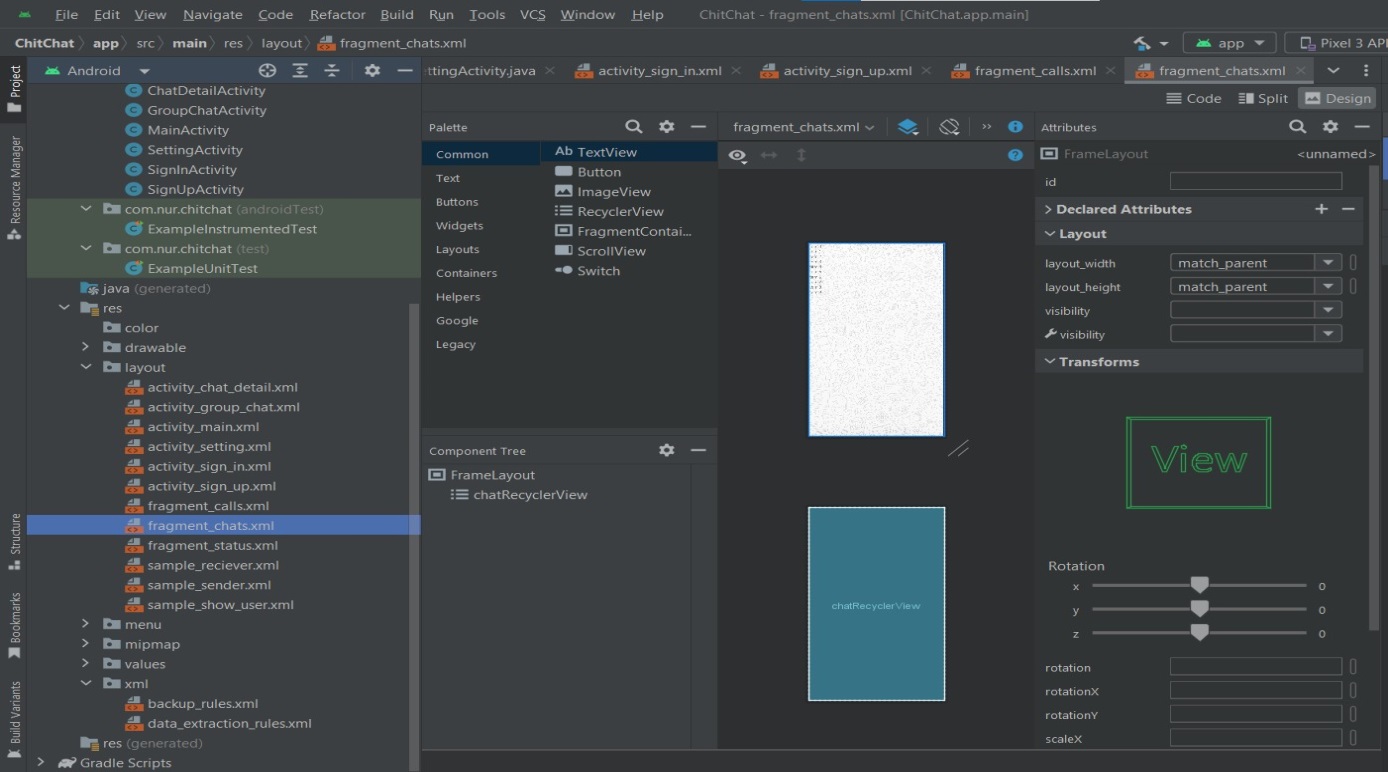
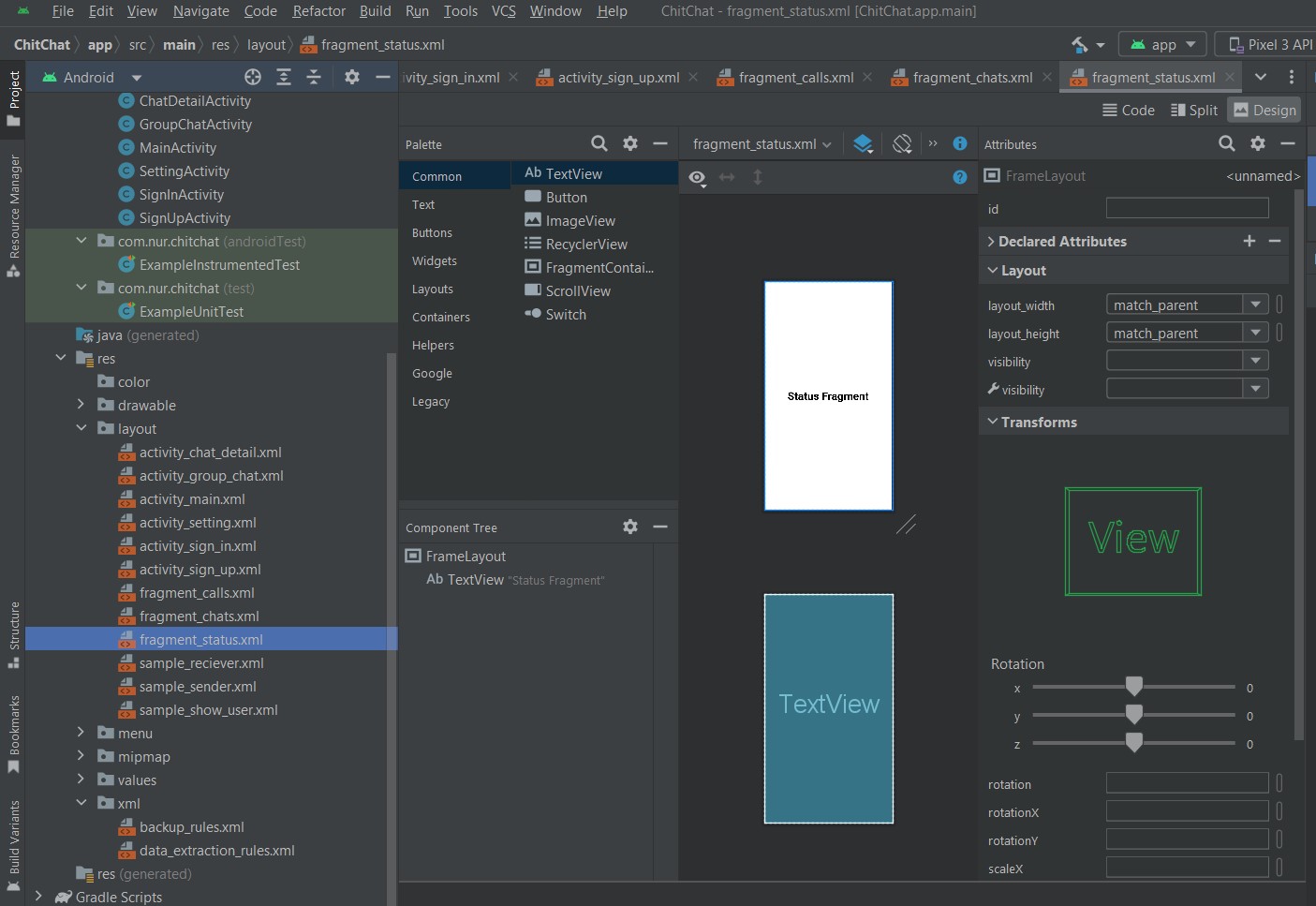
**Şekil 11.**

* **Activity Sign Up**

****

* **Fragment Calls**

**Şekil 13.**

* **Fragment Chats**
* **Fragment Status**

**Şekil 15.**

* **Sample Reciever**

Bu XML dosyası, bir alıcı mesajının görünümünü tanımlar. İşlevleri şu şekildedir:

1.ConstraintLayout: Diğer görünüm öğelerini düzenlemek için kullanılan bir düzen yöneticisidir. Görünüm öğeleri, birbiriyle bağlantılı kısıtlar yardımıyla yerleştirilir.

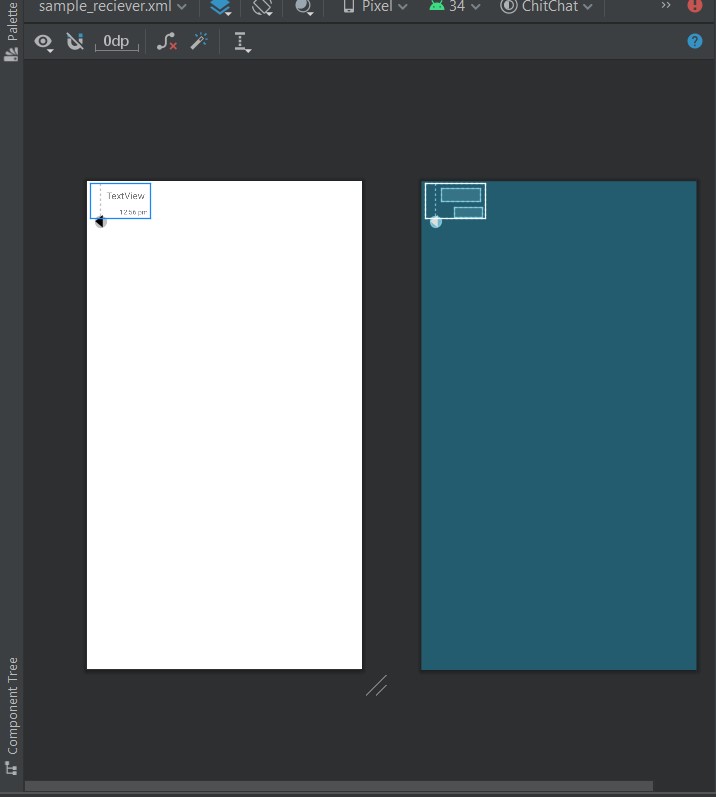
* android:background: Arka plan resmini belirtir.
* android:maxWidth, android:maxHeight, android:minHeight: Görünüm öğesinin maksimum genişlik, maksimum yükseklik ve minimum yükseklik değerlerini belirtir. Bu, metin kutusunun boyutlarını sınırlar.
* android:layout\_marginStart, android:layout\_marginLeft, android:layout\_marginTop, android:layout\_marginRight: Sol, üst ve sağ kenarlara marjlar ekler, düzeni biraz kenara taşır.

2.TextView: Alıcı mesajını içeren metin kutusudur. Mesajı gösterir ve görsel olarak düzenlenir. Metin kutusunun altında bir saat etiketi bulunur.

* android:id: Metin kutusunun benzersiz tanımlayıcısıdır.
* android:maxWidth, android:minWidth: Metin kutusunun maksimum ve minimum genişliğini belirtir.
* app:layout\_constraintBottom\_toTopOf, app:layout\_constraintEnd\_toEndOf, app:layout\_constraintStart\_toStartOf, app:layout\_constraintTop\_toTopOf: Metin kutusunun ConstraintLayout içindeki diğer öğelere bağlı olarak konumunu ve boyutunu belirler.
* app:layout\_constraintVertical\_bias: Dikey konumunu belirler.

3.Guideline: Metin kutusunun sol kenarını belirlemek için kullanılır. Metin kutusunun sol kenarı, belirtilen uzaklıkta bir kılavuz çizgisi ile sabitlenir.

Bu tasarım, alıcı mesajlarını görüntülemek için kullanılır. Mesajlar, metin kutusu içinde yer alır ve zaman damgası metnin altında gösterilir. ConstraintLayout, öğelerin yerleşimini ve boyutlarını düzenlerken, Guideline belirli bir öğenin konumunu belirler.



**Şekil 16.**

* **Sample Sender**

Bu XML dosyası, bir gönderen mesajının görünümünü tanımlar. İşlevleri şu şekildedir:

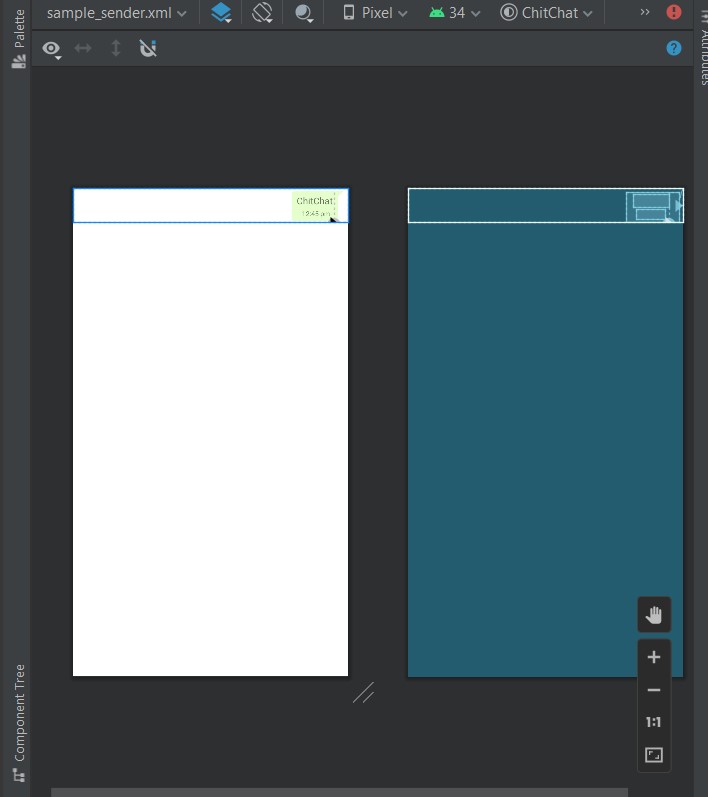
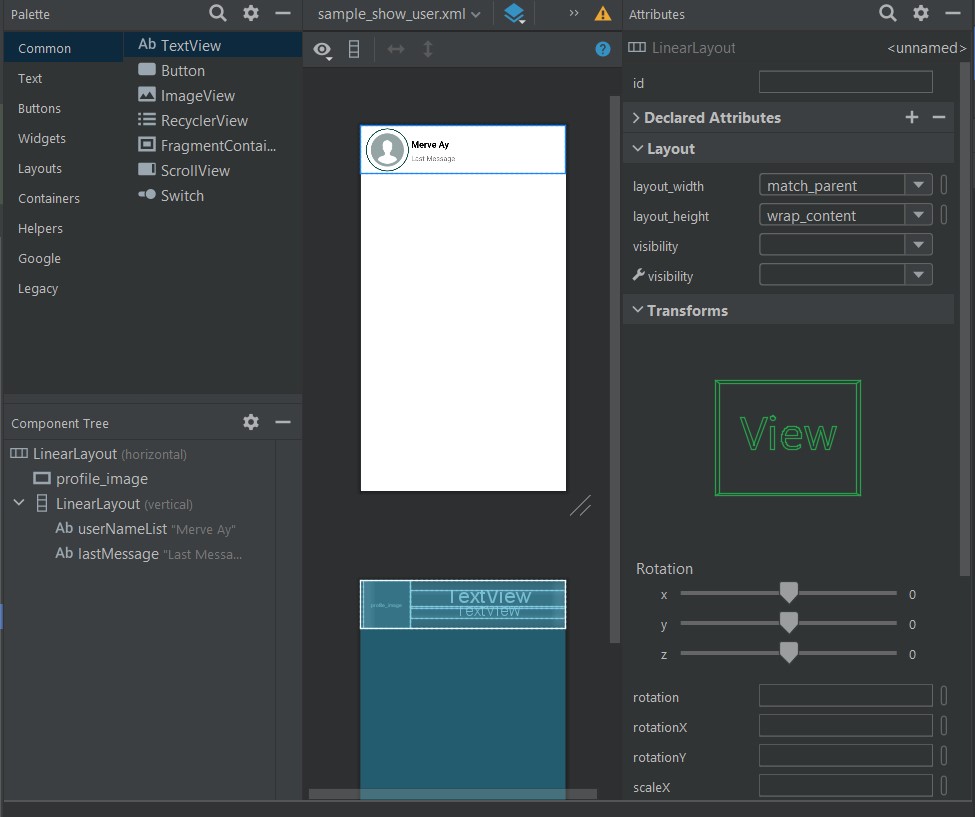
1.RelativeLayout: Diğer görünüm öğelerini düzenlemek için kullanılan bir düzen yöneticisidir. Gönderen mesajın arka planını belirler.

2.ConstraintLayout: Metin kutusu ve zaman damgasını içeren bir içerik düzeni tanımlar. İçeriğin sağ alt köşesine hizalanır.

* android:background: Arka plan resmini belirtir.
* android:maxWidth, android:maxHeight, android:minHeight: Görünüm öğesinin maksimum genişlik, maksimum yükseklik ve minimum yükseklik değerlerini belirtir. Bu, metin kutusunun boyutlarını sınırlar.

3.Guideline: Metin kutusunun sağ kenarını belirlemek için kullanılır. Metin kutusunun sağ kenarı, belirtilen uzaklıkta bir kılavuz çizgisi ile sabitlenir.

4.TextView: Gönderen mesajı içeren metin kutusudur. Metin kutusu, mesajı gösterir ve görsel olarak düzenlenir. Metin kutusunun altında bir saat etiketi bulunur.

* android:id: Metin kutusunun benzersiz tanımlayıcısıdır.
* android:maxWidth, android:minWidth: Metin kutusunun maksimum ve minimum genişliğini belirtir.
* app:layout\_constraintBottom\_toTopOf, app:layout\_constraintEnd\_toStartOf, app:layout\_constraintStart\_toStartOf, app:layout\_constraintTop\_toTopOf: Metin kutusunun ConstraintLayout içindeki diğer öğelere bağlı olarak konumunu ve boyutunu belirler.
* app:layout\_constraintVertical\_bias: Dikey konumunu belirler.
* **Sample Show User**

**Şekil 18.**

* **Android Manifest**

Bu XML dosyası, Android uygulamanızın manifest dosyasını temsil eder. Android uygulamalarının yapılandırılması ve bileşenlerinin tanımlanması için kullanılır. Aşağıdaki özellikleri içerir:

1.<manifest>: Dosyanın kök öğesi. Uygulamanın genel özelliklerini ve izinlerini tanımlar.

* xmlns:android: Android XML dosyası için varsayılan ad alanını belirtir.
* xmlns:tools: Android Studio tarafından kullanılan araçlar için ad alanını belirtir.

2. <application>: Uygulamanın ana etiketidir. Uygulamanın temel özelliklerini ve tüm bileşenlerini içerir.

* android:allowBackup: Uygulama verilerinin yedeklenip yedeklenmeyeceğini belirtir.
* android:dataExtractionRules: Veri çıkarma kurallarını belirtir.
* android:fullBackupContent: Tam yedekleme içeriğini belirtir.
* android:icon: Uygulama simgesini belirtir.
* android:label: Uygulama etiketini belirtir.
* android:roundIcon: Yuvarlak simgeyi belirtir.
* android:supportsRtl: Sağdan sola dil desteğini belirtir.
* android:theme: Uygulama için kullanılan tema stili.
* tools:targetApi: Araçlar için hedef API sürümünü belirtir.

3. <activity>: Uygulamanın aktivitelerini tanımlar. Her aktivite, kullanıcı arayüzünün bir parçasını oluşturur.

* android:name: Aktivitenin ismini belirtir.
* android:exported: Aktivitenin diğer uygulamalara açık olup olmadığını belirtir.
* android:label: Aktivitenin etiketini belirtir.
* android:theme: Aktivite için kullanılan tema stili.
* <intent-filter>: Aktiviteye gelen belirli türdeki giriş işlemlerini tanımlar.
* <action>: Eylemi belirtir (Örneğin: "android.intent.action.MAIN").
* <category>: Eylemin kategori bilgisini belirtir (Örneğin: "android.intent.category.LAUNCHER").

Bu manifest dosyası, uygulamanızın hangi aktivitelerin ve bileşenlerin olduğunu, hangi izinlere sahip olduğunu ve uygulama başlatıldığında hangi aktivitenin görüntüleneceğini belirtir.

* **Build Gradle**

Bu Android projesinin Gradle dosyasıdır. Gradle, Android uygulamalarının derlenmesi ve bağımlılıklarının yönetilmesi için kullanılan bir yapılandırma dosyasıdır. Verilen Gradle dosyasındaki bileşenler ve yapılandırmalar şu şekildedir:

1.Plugins (Eklentiler):

* com.android.application: Android uygulaması için gerekli olan temel Gradle eklentisidir.
* com.google.gms.google-services: Google Play Hizmetleri ile entegrasyonu sağlayan Gradle eklentisi.
* org.jetbrains.kotlin.android: Kotlin dilinin Android projesine entegrasyonunu sağlayan Gradle eklentisi.

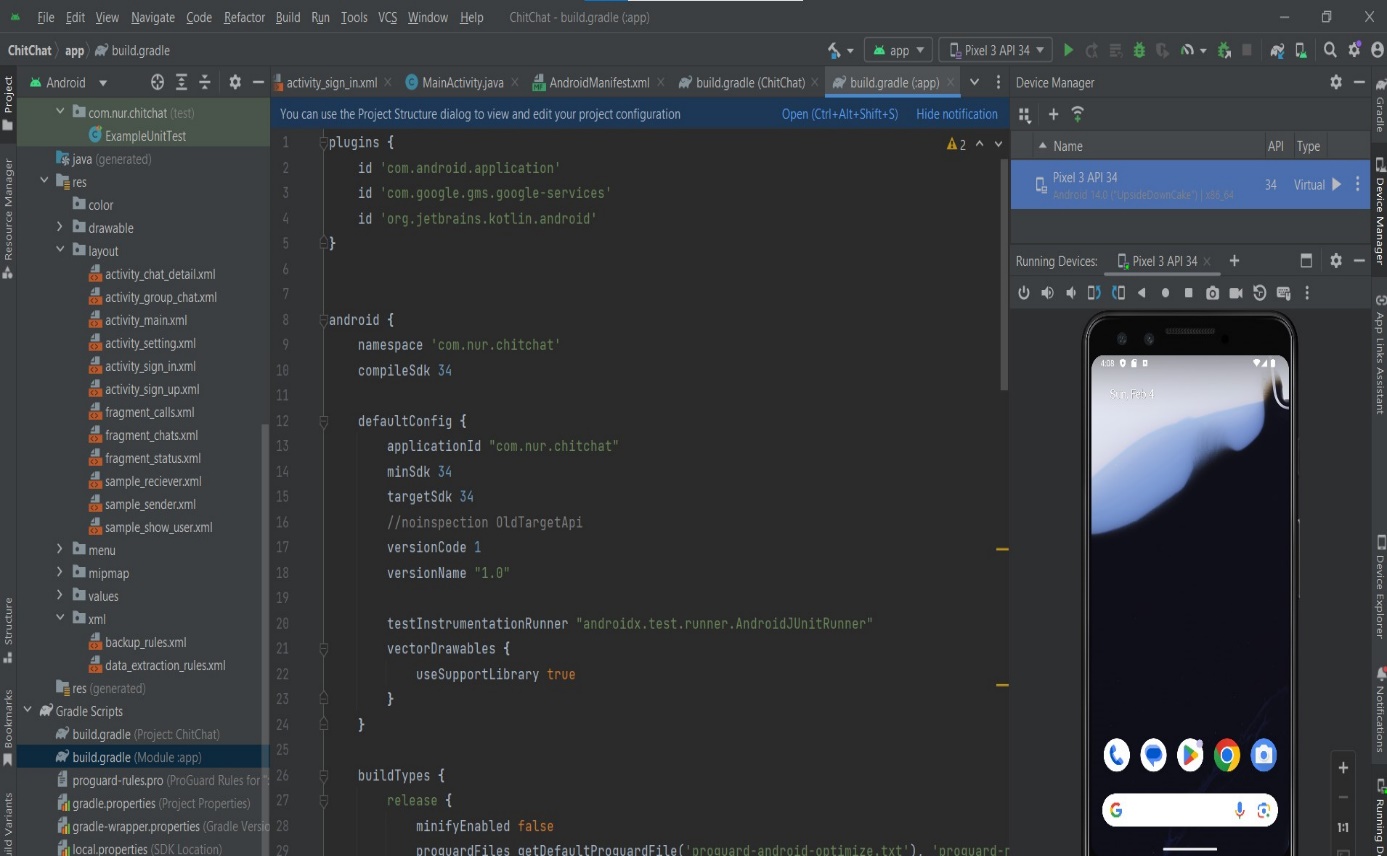
2.Android Bloğu:

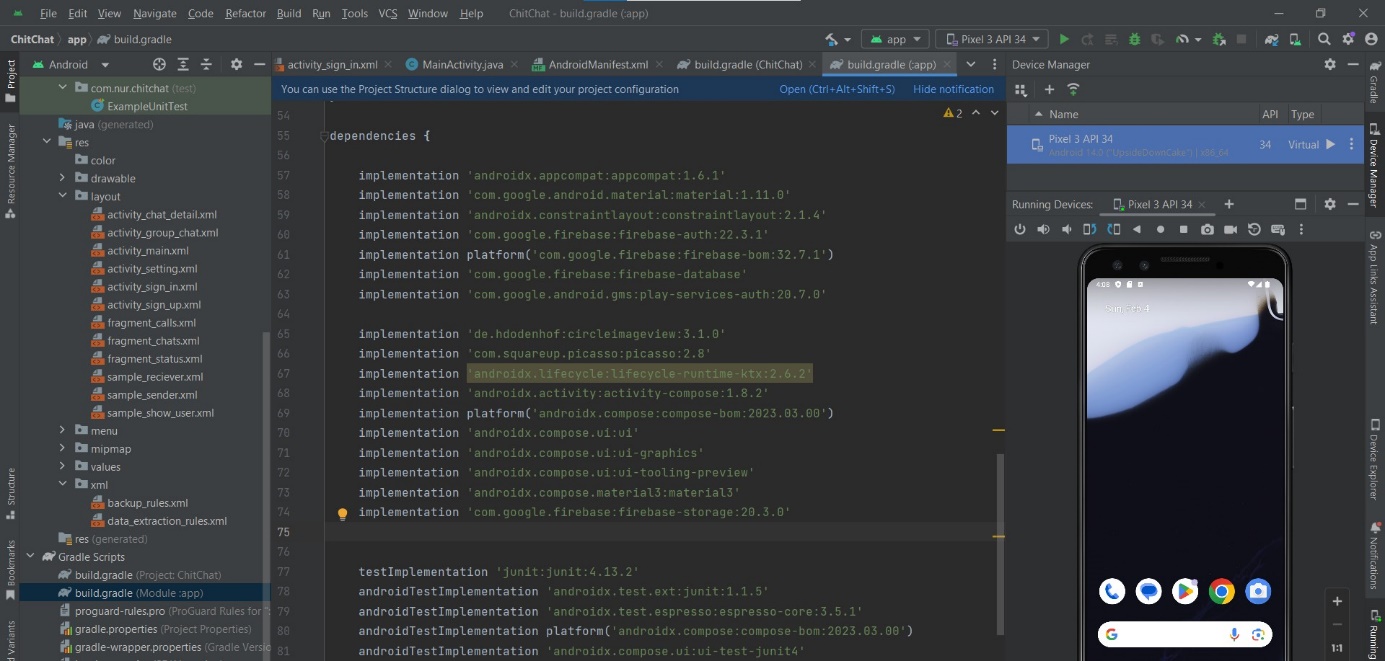
* namespace: Paket adını belirtir.
* compileSdk, minSdk, targetSdk: Uygulamanın derleme, minimum ve hedef Android SDK sürümlerini belirtir.
* defaultConfig: Temel yapılandırma ayarlarını içerir (uygulama kimliği, sürüm numarası, test koşucusu vb.).
* buildTypes: Uygulama sürümleri için yapılandırma ayarları (örneğin, yayınlanacak sürümde sıkıştırma ve ProGuard dosyalarının kullanılması gibi).
* buildFeatures: Kullanılabilir özellikleri etkinleştirir (örneğin, View Binding ve Jetpack Compose).
* compileOptions, kotlinOptions: Derleme ve Kotlin yapılandırma seçenekleri.
* composeOptions: Jetpack Compose yapılandırma seçenekleri.
* packaging: Paketleme yapılandırması (kaynak dosyalarının hariç tutulması gibi).

3.Dependencies (Bağımlılıklar):

* Uygulamanın kullanacağı harici kütüphaneleri ve bağımlılıkları içerir. Bu bağımlılıklar, Android Jetpack bileşenleri, Firebase servisleri, dış kütüphaneler ve test kütüphanelerini içerebilir.

Bu Gradle dosyası, Kotlin dili ile geliştirilen bir Android uygulamasının yapılandırmasını ve bağımlılıklarını belirtir. Uygulamanın özelliklerine ve gereksinimlerine göre değişiklik gösterebilir.



 **Şekil 19.**

**Şekil 20.**

Formun Üstü

**3.Kaynaklar**

**Youtube – BTech Days**

**Udemy – Android ile Mobil Uygulama Geliştirme**