KELİME TAHMİN OYUNU

1st Yalçın Dağbaşı
Kocaeli üniversitesi
Kocaeli/Türkiye
210202040

2nd Emir Çınar
Kocaeli üniversitesi
Kocaeli/Turkiye
210202014

I. ÖZET

- Proje sayesinde Mobil programlama aracılığıyla oyun geliştirme ve uygulama oluşturma becerisinin geliştirilmesi amaçlanmaktadır.

II. PROJE TANIMI

- Bu projede, mobil programlama kullanılarak kelime tabanlı bir oyun geliştirilmesi hedeflenmektedir. Oyun, iki kişilik olacak ve dinamik özellikler içerecektir. Oyunun alt yapısının sunucu istemci mimarisi kullanılarak tasarlanması ve platform üzerinden birden fazla oyunun aynı anda oynanabilmesi beklenmektedir.

III. GAME AKTİVİTY SINIFI

- Sınıfta, oyun için gereken bilgiler için bir dizi üye değişken ve Firebase ile bağlantılar tanımlanır. Bu, EditText bileşenlerinden Firebase Auth ve Firestore'a kadar değişir. Ayrıca, belirli eylemler için düğmeler (butonlar) tanımlanır ve kullanıcının uygulama durumu (UserState) güncellenir.
- onCreate yöntemi, etkinlik başlatıldığında çalışır ve gerekli bileşenlerin başlatılması, Firebase bağlantısının kurulması ve kullanıcı durumu güncellenmesi için kullanılır.
- Oyunun temel bilgileri kullanıcıdan alınıp (örneğin, harf sayısı, zaman limiti), ardından oyun veritabanına kaydedilir. Bir oyun kimliği (gameId) rastgele oluşturulur ve kullanıcıdan alınan parametrelere göre bir kelime seçilir.
- Oyun oluşturulduktan sonra, oyuncuların oyun durumu veritabanında güncellenir (örneğin, oyunda olup olmadıkları).
- Kullanıcıya, oyun oluşturma düğmesine basıldığında yeni bir oyun oluşturulacağı ve anasayfaya dönme, çıkış yapma ve oturumu kapatma için düğmeler sağlar.
- Oyun oluşturma düğmesine tıklandığında, oyunu başlatmak için createGame yöntemi çağrılır.
- createGame yöntemi, oyunu Firestore'da kaydetmek için kullanılır. Oyunun başarılı bir şekilde oluşturulup oluşturulmadığını kontrol eder ve ona göre kullanıcıya geri bildirim verir.
- Oyunun başarılı şekilde oluşturulmasından sonra, yeni bir etkinlik başlatılır (CreateGameWaitingActivity), kullanıcıyı yeni oluşturulan oyuna yönlendirmek için.
- Kullanıcının oturumunu kapatmak için Firebase Auth'dan çıkış yapar. Bu, aynı zamanda kullanıcının durumu çevrimdışına döner.
- Uygulamadan tamamen çıkış yapmak için de bir düğme vardır (finishAffinity), bu da uygulamanın tüm etkinliklerini kapatır.
- generateGameId, rastgele bir oyun kimliği oluşturur.
- generateRandomWord, belirli bir harf sayısına sahip kelimeleri içeren dosyadan rastgele bir kelime seçer.

IV. CREATEGAMEWAITINGACTIVITY SINIFI

A. onCreate Metodu

- Aktivite başlatıldığında çağrılır ve başlangıç ayarlarını yapar.
- Kullanıcı arabirimini (R.layout.creategamewaiting) ayarlar.
- gameId değişkenini, önceki etkinlikten gelen oyun kimliği ile doldurur.
- Firebase Firestore'a bağlantı kurar.
- lblgameID adlı TextView öğesini bulur ve içine oyun kimliğini koyar.
- checkWaitingState() fonksiyonunu çağırarak oyun durumu değişikliklerini izlemeye başlar.

B. checkWaitingState()

- Oyun durumunu izlemek ve başka oyuncuların katılıp katılmadığını kontrol etmek için oluşturulmuştur
- Firestore'daki "games" koleksiyonunda gameId ile ilgili belgeyi izlemek için bir snapshot listener ekler.
- Bir hata olup olmadığını kontrol eder ve hata mesajını görüntüler.
- Belge mevcutsa ve durumu "JOINED" ise, oyun başlamış demektir. Bu durumda, oyunu başlatmak için startGame() fonksiyonunu çağırır ve oyun durumunu "INPROGRESS" olarak günceller.

C. startGame(String, Int)

- Oyun başlamışsa, GameActivity'ye geçiş yaparak oyunu başlatır.
- Intent kullanarak GameActivity'yi başlatır.
- Oyun kimliği, zaman sınırı ve kelime bilgilerini aktarır.
- checkUserState() fonksiyonunu çağırarak kullanıcı durumunu kontrol eder.
- finish() ile mevcut etkinliği sonlandırır.

Fig. 1. GİRİŞ EKRANI

D. updateGameState()

- Oyunun durumunu ”INPROGRESS” olarak güncellemek için oluşturulmuştur
- Firestore’daki ”games” koleksiyonunda gameId belgesini güncelleyerek ”gameState” alanını ”INPROGRESS” olarak ayarlar.
- Güncelleme başarısız olursa bir hata mesajı gösterir.

E. onBackPressed()

- Kullanıcı geri düğmesine bastığında yapılacak işlemleri belirlemek için oluşturulmuştur.
- Kullanıcının durumunu ”ONLINE” olarak günceller.
- Varsayılan geri düğmesi davranışını gerçekleştirmek için super.onBackPressed() çağrısını yapar.

F. checkUserState(String)

- Belirtilen kullanıcı kimliğine (userId) sahip kullanıcının durumunu kontrol etmek için oluştulmuştur
- Firestore’daki ”users” koleksiyonunda ilgili belgeyi alır.
- Başarılıysa, kullanıcının durumunu kontrol eder ve log mesajı yazar.
- Başarısız olursa hata mesajı gösterir.

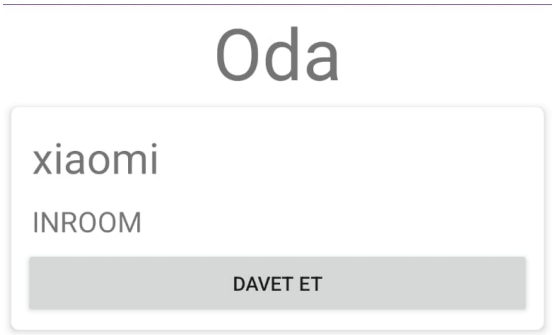

Fig. 2. DAVET EKRANI

V. GameActivity SINIFI

A. onCreate

- Bu fonksiyon, bir Android aktivitesi başlatıldığında çalışır. Kullanıcı arayüzünü ayarlar, Firestore’dan oyun verilerini alır ve oyun durumunu dinler.
- Layout’u ayarlar (setContentView).
- gameId ve kelime değişkenlerini Intent’ten alır.
- Kullanıcı durumunu ”oyunda” olarak günceller.

- Firebase Firestore'dan oyun modelini alır.
- Zamanlayıcıyı başlatır (startCountDown).
- Firestore oyun durumunu dinler. Oyun bitiş durumunu algılar ve endGame fonksiyonunu çağırır.
- Kullanıcı tahminini başlatmak için butonlar ve giriş alanlarını ayarlar.
- Kelimeyi göstermek için kutuları oluşturur (kelimeyiGoster).

B. kelimeyiGoster()

- Bu fonksiyon, oyunun başlangıcında kelimenin temsil edileceği kutuları oluşturur.
- Ekran genişliğini hesaplar.
- Satır (LinearLayout) ve her harfi temsil eden kutuları (TextView) oluşturur.
- Her kutuya boş bir metin ekler ve oyun alanına yerleştirir.

C. olusturSatir()

- Bu fonksiyon, yatay bir satır oluşturur.
- Yatay yönelimde bir LinearLayout oluşturur.
- Uygun yerleşim özelliklerini belirler.
- Satırı döndürür

D. olusturKutu()

- Bu fonksiyon, bir kelime kutusu (TextView) oluşturur.
- Bir TextView oluşturur ve metin, yazı tipi, renk, arka plan, yastık değerleri gibi özelliklerini ayarlar.
- Yerleşim parametrelerini ayarlar (kenarlık, ağırlık, yastık değerleri).
- Kutunun merkezi hizalandığını kontrol eder ve kutuyu döndürür.

E. tahminEt()

Bu fonksiyon, kullanıcının tahmin girdisini alır ve oyunu yönetir.

- Kullanıcı tahminini guessInput'tan alır.
- Yeni bir satır oluşturur ve her harfi temsil eden kutuları ekler.
- Tahmin edilen harfi doğru, kısmen doğru, yanlış renklerle işaretler.
- Doğru tahmin edilirse, oyunu bitirir ve Firestore'daki oyunu günceller.
- Yanlış tahmin edilirse, tahmini günceller ve yeni satırı oyun alanına ekler.
- Tahmin edilen kelimeyi ilgili oyuncunun Firestore'daki tahminler dizisine ekler.

F. goToMain()

- Bu fonksiyon, kullanıcıyı ana aktiviteye (ana sayfa) yönlendirir.
- Intent ile MainActivity'ye yönlendirir.
- GameActivity'yi bitirir.

G. endGame(winnerId: String)

Bu fonksiyon, oyunu bitirir ve kazananın kim olduğunu gösterir.

- Kullanıcı tahmin girdi alanını ve tahmin butonunu devre dışı bırakır.
- Kazananı ekranda gösterir ve oyunu durdurur.
- Firestore'daki oyun durumunu "bitmiş" olarak günceller ve kazananı belirler.
- Zamanlayıcıyı durdurur.

H. stopCountDown()

- Bu fonksiyon, geri sayım zamanlayıcısını durdurur.
- startCountDown(timeLimit: Int)
- Bu fonksiyon, geri sayım zamanlayıcısını başlatır
- Bir CountDownTimer oluşturur ve verilen sürede (saniye) geri sayımı başlatır.
- Geri sayım sırasında kalan zamanı gösterir (onTick).
- Süre dolduğunda, oyunu "beraberlik" olarak bitirir (onFinish).

VI. GAMEDATA SINIFI

A. saveGameModel(model: GameModel)

- Bu fonksiyon, bir GameModel nesnesini hem iç bellek hem de Firebase Firestore'a kaydetmek için kullanılır.
- MutableLiveData türündeki gameModel değişkenine yeni bir GameModel nesnesi gönderir (postValue).
- Firebase Firestore'daki "games" koleksiyonuna bu modeli kaydeder. Modelin belgesine, gameId ile erişilir.
- Firebase'deki kaydetme işlemi asenkron olduğundan, hata durumlarıyla başa çıkmak için ek kod gerekebilir.

B. fetchGameModel()

Bu fonksiyon, mevcut oyunu Firebase Firestore'dan dinlemek ve yeni verileri güncellemek için kullanılır.

- GameModel'in mevcut değerine bakar ve Firestore'da bu oyunla ilişkili belgeyi izlemeye başlar.
- addSnapshotListener kullanarak Firestore belgesinde değişiklikleri dinler.
- Herhangi bir değişiklik olduğunda, Firestore belgesinden bir GameModel nesnesi oluşturur ve gameModel'e gönderir (postValue).
- Bu, oyunun durumunun gerçek zamanlı olarak güncellenmesini sağlar.

C. gameModel

Bu, dış sınıfların LiveData¡GameModel¿'e erişmesini sağlayan bir özellik (property) temsil eder.

- MutableLiveData olan gameModel'in dışarıdan okunmasını sağlar, ancak değiştirilmesini engeller. Bu, reaktif programlama modelinde yaygın bir yaklaşımdır.
- Başka bir sınıf, gameModel.observe(...) yöntemi ile bu verileri gözlemleyebilir ve yeni bir değer geldiğinde tepki verebilir.

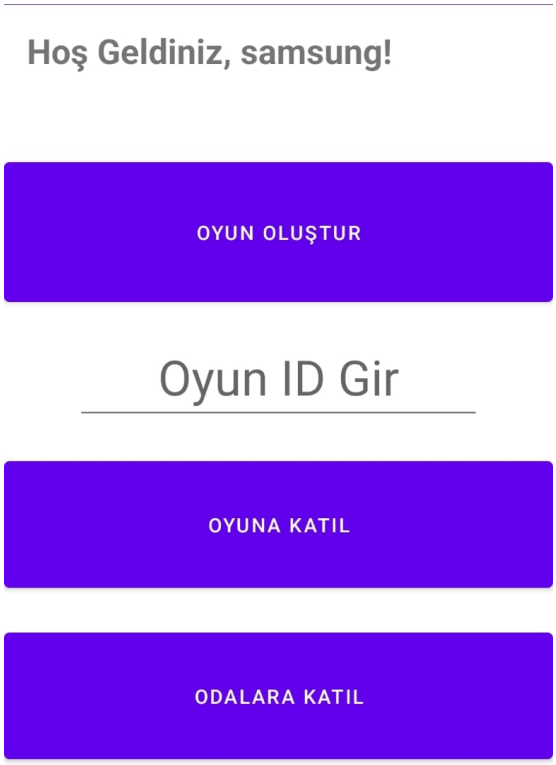

Fig. 3. OYUN GİRİŞ EKRANI

Fig. 4.

VII. GameTypeActivity sinifi

A. *onCreate(Bundle?)*

- Bu, Android'de bir aktivite oluşturulduğunda çağrılan ana başlangıç fonksiyonudur. Bu fonksiyon, sınıf içindeki diğer öğeleri ve işlevleri başlatır. İşte bu fonksiyonun başlıca bileşenleri:
- firebaseAuth: Firebase Authentication nesnesini başlatır ve kullanıcı oturum yönetimi için kullanılır.
- userId: Şu anda oturum açmış kullanıcının kimlik bilgilerini alır. Kullanıcı çevrimiçi duruma getirilir.
- btnHomePage: Ana sayfaya gitmek için bir buton. Kullanıcı bu butona tıklayınca MainActivity'ye yönlendirilir ve bu aktivite sonlandırılır (finish()).
- btnSignOut: Kullanıcının oturumunu kapatmak için bir buton. Bu buton tıklandığında, Firebase oturumu kapatılır, SignInActivity'ye yönlendirilir, kullanıcı durumu "OFFLINE" olarak güncellenir ve mevcut aktivite sonlandırılır.
- buttonLogout: Uygulamadan çıkış yapmak için bir buton. Bu buton tıklandığında, kullanıcı durumu "OFFLINE" olarak güncellenir ve uygulama kapatılır (finishAffinity()).
- btngameType1 ve btngameType2: Oyun türü seçmek için butonlar. btngameType1 butonu tıklandığında, oyun türü "1" olarak ayarlanır ve RoomsActivity'ye yönlendirilir. btngameType2 için de benzer bir işlem yapılır, ancak oyun türü "2" olarak ayarlanır. Her iki durumda da mevcut aktivite sonlandırılır.

VIII. JoinGameActivity Sinifi

A. onCreate(Bundle?)

- Bu, aktivite ilk kez başlatıldığında çalışan fonksiyondur. UI öğelerini başlatır ve oyunun durumunu kontrol etmek için gerekli ayarlamaları yapar. İşte ana noktaları:
- gameId'yi alır: Intent ile gelen oyun kimliğini alır.
- firebaseFirestore ve firebaseAuth başlatılır: Firebase hizmetleriyle çalışmak için başlatılır.
- Oyun kimliği TextView'e atanır: Kullanıcıya oyun kimliğini göstermek için lblGameID
- TextView'ini doldurur.
- Firestore'dan oyun belgesini alır: gameId ile Firestore'dan oyunun belgesini alır. Eğer belge yoksa, hata mesajı gösterir ve aktiviteyi sonlandırır.
- Oyun durumu ve player2Id'yi günceller: Belge başarılı bir şekilde alındığında, oyun durumunu "JOINED" olarak günceller ve ikinci oyuncunun kimliğini ayarlar.
- checkWaitingState()'i çağırır: Oyun durumunu dinlemek ve değişikliklere göre hareket etmek için bu fonksiyonu çağırır.

B. checkWaitingState()

- Bu fonksiyon, Firestore'da bir oyunun durumunu dinler ve oyun durumuna göre oyun başlatma veya beklemeye devam etme işlemlerini yapar. İşte fonksiyonun ana hatları:
- gameListener'ı temizler: Yeni bir dinleyici eklemeden önce mevcut dinleyiciyi kaldırır.
- SnapshotListener ekler: gameId ile ilgili belgeyi dinler. Eğer bir hata olursa, hata mesajı gösterir ve aktiviteyi sonlandırır.
- gameState kontrolü: Eğer "JOINED" durumuna gelirse, startGame() fonksiyonunu çağırarak oyunu başlatır. Oyun başlamadan önce kelime ve zaman sınırı alınır.
- Kelime kontrolü: Eğer kelime veya zaman sınırı eksikse, hata mesajı gösterir ve checkWaitingState()'i yeniden çağırır.
- startGame(kelime: String, timelimit: Int)
- Bu fonksiyon, oyun başladığında yeni bir GameActivity açmak için kullanılır. Gerekli verileri Intent ile aktarır ve yeni aktiviteyi başlatır. İşte fonksiyonun ana noktaları:
- Intent oluşturur: GameActivity'ye geçmek için Intent oluşturur.
- Gerekli verileri ekler: Oyun kimliği, kelime ve zaman sınırı gibi bilgileri Intent içerisine koyar.
- GameActivity'yi başlatır: Intent ile yeni aktiviteyi başlatır ve bu aktiviteyi sonlandırır.

C. onBackPressed()

- Bu fonksiyon, kullanıcının geri tuşuna bastığında ne olacağını belirler. Bazı ek işlemler yapar:
- gameListener'ı kaldırır: Oyun durumunu dinlemeyi durdurur.
- Kullanıcı durumunu günceller: Geri döndüğünde kullanıcıyı "ONLINE" durumuna getirir.
- super.onBackPressed() çağırır: Varsayılan geri tuşu davranışını yerine getirir.

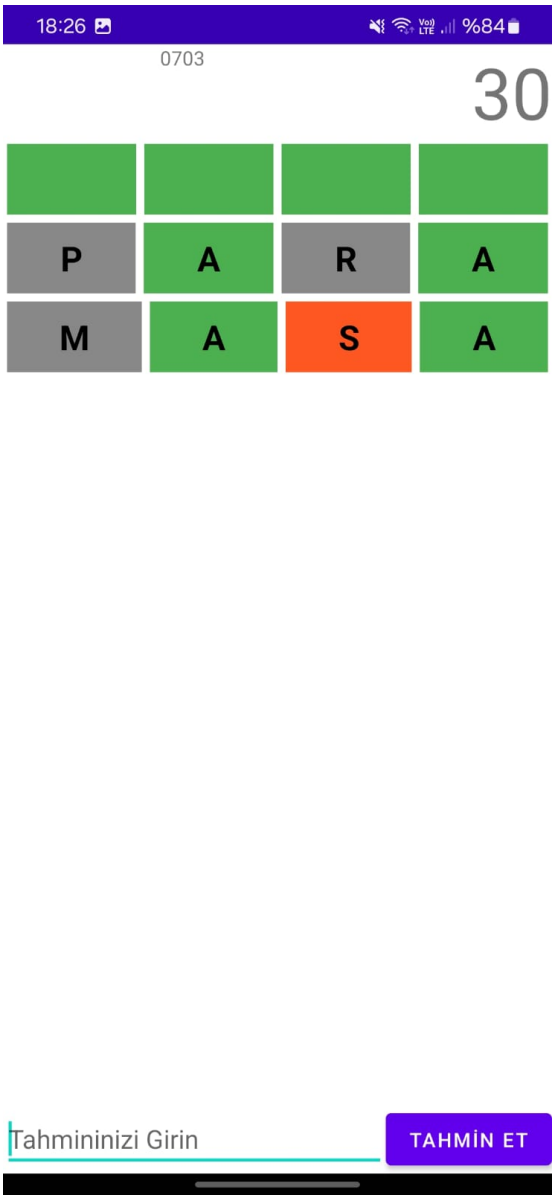

Fig. 5.

IX. MainActivity Sinifi

A. onCreate(Bundle?)

- Bu, aktivite oluşturulduğunda çalışan ana fonksiyondur. Kullanıcıyı karşılar ve UI öğelerini başlatır. İşte ana işlevleri:
- firebaseAuth ve database başlatılır: Firebase hizmetleriyle çalışmak için bu nesneler başlatılır.
- Kullanıcı adını alır: Firebase Firestore'dan kullanıcı adını almak için kullanıcı kimliğini kullanır. Alınan kullanıcı adı, bir karşılama mesajı oluşturmak için TextView öğesine atanır.

- Kullanıcı durumunu ”ONLINE” yapar: Utils.updateUserState fonksiyonunu kullanarak, kullanıcının durumunu ”ONLINE” olarak günceller.
- Firestore’dan kullanıcı bilgilerini alır: Firestore’da users koleksiyonundaki kullanıcı belgesini alarak kullanıcı adını elde eder ve hoş geldiniz mesajını günceller.

B. createGameRoom()

Bu fonksiyon, oyun oluşturma işlemini başlatmak için kullanılır:

- Intent oluşturur: CreateGameActivity’ye geçmek için bir Intent oluşturur.
- Yeni aktiviteyi başlatır: Intent ile CreateGameActivity’yi başlatır.

C. joinGameRoom()

Bu fonksiyon, bir oyuna katılma işlemini başlatır:

- Oyun kimliği kontrolü yapar: Kullanıcıdan oyun kimliğini alır. Eğer kimlik boşsa, bir hata mesajı gösterir.
- Intent oluşturur: JoinGameActivity’ye geçmek için bir Intent oluşturur ve oyuna katılmak için gerekli verileri ekler.
- Yeni aktiviteyi başlatır: Intent ile JoinGameActivity’yi başlatır.
- btnHomePage: Bu buton, ana sayfaya dönmek için kullanılır. Butona tıklandığında finish() çağrılarak mevcut aktivite sonlandırılır.
- btnSignOut: Bu buton, oturumu kapatmak için kullanılır. Butona tıklandığında Firebase oturumunu kapatır ve SignInActivity’ye geçer. Kullanıcı durumunu ”OFFLINE” olarak günceller ve mevcut aktiviteyi sonlandırır.
- buttonLogout: Bu buton, uygulamadan tamamen çıkmak için kullanılır. Butona tıklandığında finishAffinity() çağrılarak uygulama tamamen sonlandırılır ve kullanıcı durumu ”OFFLINE” olarak güncellenir.
- btncreateGame: Bu buton, yeni bir oyun oluşturmak için kullanılır. Butona tıklandığında createGameRoom() çağrılır.
- btnjoingame: Bu buton, bir oyuna katılmak için kullanılır. Butona tıklandığında joinGameRoom() çağrılır.
- btnGameTypeRoom: Bu buton, farklı oyun türlerini listelemek için kullanılır. Butona tıklandığında GameTypeActivity’ye geçilir.

X. RoomsActivity Sinifi

A. onCreate(Bundle?)

- Aktivite oluşturulduğunda çalışan ana fonksiyon. Aşağıdakileri içerir:

B. onResume()

- Aktivite yeniden başladığında çalışır. Ana işlevi, kullanıcının oda türünü ve durumunu eski haline getirmektir.

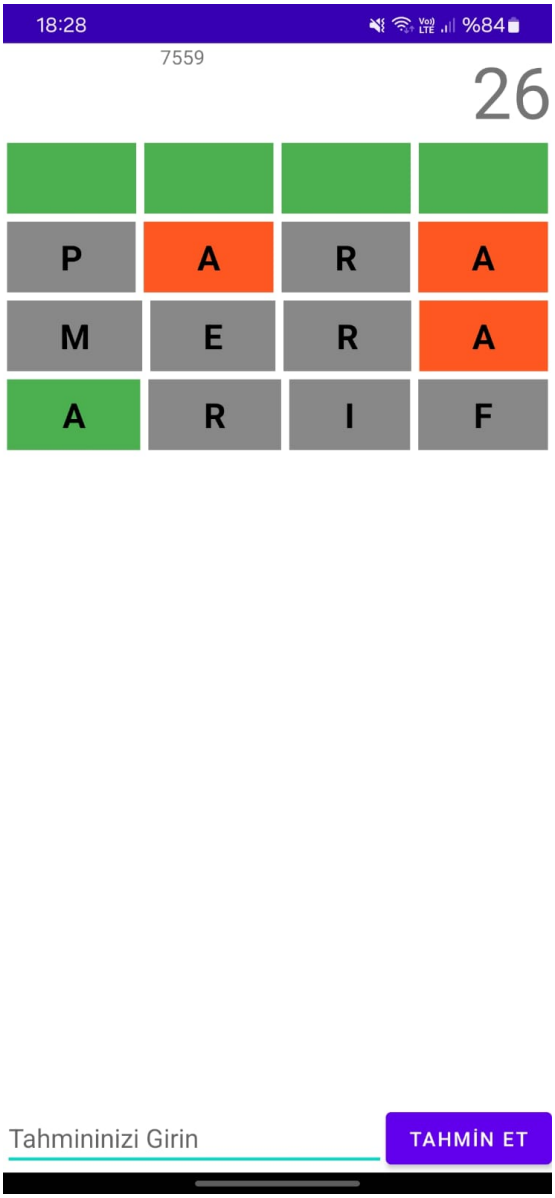

Fig. 6.

C. getUsersInRoom(RoomType)

- Belirli bir oda türündeki kullanıcıları listelemek için kullanılan fonksiyon. Şu işlemleri yapar:
- Firestore’dan kullanıcıları alır: Belirtilen oda türündeki tüm kullanıcıları alır.
- Kullanıcıları liste olarak gösterir: Alınan kullanıcıları ScrollView içinde listeler. Her kullanıcı için bir CardView oluşturur ve kullanıcı bilgilerini, davet durumlarını gösterir.
- Davet gönderme ve kabul etme işlemleri için butonları ayarlar: Kullanıcı için davet butonu oluşturur. Butona tıklandığında sendInvite veya acceptInvite fonksiyonlarını çağırır.
- Davet durumunu günceller: Butonun metnini, kullanıcının

davet durumuna göre günceller.

D. sendInvite(fromUserId, toUserId)

- Bir kullanıcıdan diğerine davet göndermek için kullanılan fonksiyon. İşleyişi:
- Davet listelerini günceller: fromUserId için sentInvites, toUserId için receivedInvites güncellenir.
- Oyun durumunu dinler: joinGameListener kullanılarak, kullanıcı bir oyuna girerse JoinGameActivity'ye yönlendirilir.

E. acceptInvite(fromUserId, toUserId)

- Bir kullanıcının davetini kabul etmek için kullanılan fonksiyon. Şu işlemleri yapar:
- Davet listelerini günceller: toUserId için receivedInvites, fromUserId için sentInvites güncellenir.
- Oyun başlatır: Davet kabul edildiğinde CreateGameActivity'ye yönlendirir ve gerekli bilgileri iletir.

F. onBackPressed()

- Geri düğmesine basıldığında çalışır. Kullanıcı durumunu "ONLINE" olarak günceller ve roomListener'ı kaldırır.

G. onDestroy()

- Aktivite yok edildiğinde çalışır. Tüm dinleyicileri kaldırır ve uygulama kapanmadan önce kaynakları temizler.

XI. RoomsActivity Sinifi

A. onCreate(Bundle?)

Aktivite oluşturulduğunda çalışan ana fonksiyon. Aşağıdaki işlemleri gerçekleştirir:

- Başlangıç ayarlarını yapar: setContentView ile görünüm ayarlanır ve firebaseAuth başlatılır. Kullanıcı durumu güncellenir: Utils.updateUserState ile kullanıcının durumu "ONLINE" olarak güncellenir.
- Ana Sayfa ve Oturum Kapatma Butonları Ayarlanır:
- btnHomePage: Ana sayfaya dönüş sağlar.
- btnSignOut: Oturum kapatma işlemini gerçekleştirir ve SignInActivity'ye yönlendirir.
- buttonLogout: Uygulamadan tamamen çıkış yapar (finishAffinity).
- btnroom4: Oda uzunluğu 4 olan odalara yönlendirir. Oyun türüne göre doğru oda türünü belirler.
- btnroom5: Oda uzunluğu 5 olan odalara yönlendirir.
- btnroom6: Oda uzunluğu 6 olan odalara yönlendirir.
- btnroom7: Oda uzunluğu 7 olan odalara yönlendirir. Her bir buton, farklı oda türlerine yönlendirme sağlayacak şekilde RoomActivity'ye Intent ile bilgi gönderir. Gönderilen bilgiler şunlardır:
- roomType: Oda türünü temsil eder (oyun türüne ve kelime uzunluğuna göre).
- gameType: Oyunun türünü temsil eder.
- lettercount: Kelimenin uzunluğunu temsil eder.

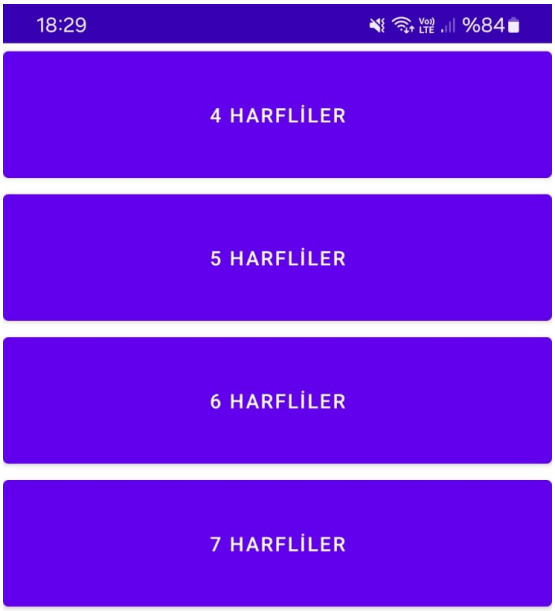

Fig. 7.

XII. SignInActivity Sinifi

A. onCreate(Bundle?)

- Aktivite başlatıldığında çağrılan ana fonksiyon. Bu fonksiyon aşağıdaki işlemleri gerçekleştirir:
- Görünümü ayarla: ActivitySignInBinding.inflate ile görünüm bağlanır ve setContentView ile yapılandırılır.
- FirebaseAuth başlatılır: firebaseAuth nesnesi oluşturulur. Kayıt Olma Etkinliği: binding.textView üzerine tıklandığında, kullanıcıyı kayıt olma ekranına (SignUpActivity) yönlendirir.
- Giriş Butonu Olayı: binding.button üzerine tıklanıldığında, e-posta ve şifre ile giriş işlemini gerçekleştirir.
- Boş Alan Kontrolü: E-posta ve şifre alanlarının boş olup

olmadığı kontrol edilir. Eğer boş ise, kullanıcıya bir uyarı mesajı gösterilir.

- Firebase Authentication ile Giriş: E-posta ve şifre bilgileri girildikten sonra Firebase Authentication kullanılarak giriş yapılır. Giriş başarılı olursa kullanıcı ana ekrana (MainActivity) yönlendirilir. Giriş başarısız olursa hata mesajları gösterilir.
- Bağlantı hatası kontrolü: Firebase Authentication bir hata döndürürse, hata mesajının türüne göre uygun bir uyarı mesajı gösterilir. Örneğin, "Geçersiz şifre", "Geçersiz e-posta formatı", veya "Bu kullanıcı mevcut değil" gibi.
- Giriş Başarılı: Giriş başarılı olduğunda kullanıcı ana ekrana yönlendirilir.

B. onStart()

- Otomatik Giriş Kontrolü: Aktivite başladığında, eğer zaten giriş yapılmışsa (firebaseAuth.currentUser != null), kullanıcıyı ana ekrana (MainActivity) yönlendirir.

XIII. SignUpActivity Sinifi

A. onCreate(Bundle?)

- Aktivite başlatıldığında çağrılan ana fonksiyon. Bu fonksiyon aşağıdaki işlemleri gerçekleştirir:
- Görünümü ayarla: ActivitySignUpBinding.inflate ile görünüm bağlanır ve setContentView ile yapılandırılır.
- FirebaseAuth başlatılır: firebaseAuth nesnesi oluşturulur.
- Giriş Ekranına Yönlendirme: binding.textView üzerine tıklanıldığında, kullanıcıyı giriş ekranına (SignInActivity) yönlendirir.
- Kayıt Olma Butonu Olayı: binding.button üzerine tıklandığında, kullanıcı girişine göre hesap oluşturma işlemini gerçekleştirir.
- Boş Alan Kontrolü: İsim, e-posta, şifre ve şifre doğrulama alanlarının boş olup olmadığı kontrol edilir. Boş alan varsa, kullanıcıya uyarı mesajı gösterilir.
- Şifre Eşleşme Kontrolü: Şifre ile şifre doğrulama alanlarının aynı olup olmadığı kontrol edilir. Eğer şifreler eşleşmiyorsa, kullanıcıya "Şifreler eşleşmiyor" mesajı gösterilir.
- Firebase Authentication ile Hesap Oluşturma: E-posta ve şifre kullanılarak Firebase Authentication ile hesap oluşturulur. Oluşturma başarılı olursa kullanıcı giriş ekranına (SignInActivity) yönlendirilir ve kullanıcı Firestore veritabanına eklenir. Başarısız olursa hata mesajı gösterilir.

B. addUserToDatabase

Kullanıcıyı Veritabanına Ekleme: Kullanıcı bilgileri Firestore'a eklenir. Kullanıcının kimliği (userId), isim, e-posta, şifre ve kayıt tarihi User sınıfından bir nesne oluşturularak Firestore'da saklanır.

- Başarı Durumları: Kullanıcı Firestore'a başarıyla eklenirse, bir log kaydı oluşturulur. Başarısız olursa bir hata mesajı loglanır.

XIV. KAYNAKÇA

- https://www.youtube.com/watch?v=C7g4VBRSyjo
- https://www.youtube.com/watch?v=9Yl8IeTRJdE
- https://www.youtube.com/watch?v=7oA0Sqe3xUYlist=PLYczkp81TE