**CheatingShazam**

*Sisteme mobile și aplicații*

*2018-2019*

**Realizat de: Robert Sîrb**

**Mirabela Ursu**

**Prof. coord.: Claudiu Groza**

1. **INTRODUCERE**

Aplicația CheatingShazam are ca principală funcționalitate recunoașterea melodiei înregistrate și căutarea unor melodii asemanătoare după algoritmi care compară frecvențele acestora , adică posibile melodii care au copiat sau au fost muze pentru melodia în cauză. Ideea proiectului pornește de la faptul că este foarte greu pentru un calculator să distingă și să recunoască diferențele dintre melodii – creierul uman poate distinge mai multe voci care se aud în același timp, dar pe calculator este foarte greu să reconstruiești sunetul original pentru a afla câte surse au fost defapt, iar acest lucru ni s-a părut o provocare.

Aplicația mobilă care ne-a inspirat este una dintre cele mai cunoscute de pe piața actuală: aplicația Shazam care identifică în câteva secunde o melodie ce cântă în mediul înconjurător dispozitivului pe care rulează. CheatingShazam încearcă să facă același lucru și puțin mai mult pe partea de recunoaștere – să recunoască și melodii asemanătoare.

1. **DESIGN ȘI IMPLEMENTARE**

Aplicația oferă următoarele cazuri de utilizare:

-utilizatorul se poate înregistra folosind un email și o parolă – funcționalitate implementată folosind autentificarea bazată pe email și parolă oferită de *Firebase Authentication SDK*;

-utilizatorul înregistrează o melodie ce cântă în apropierea dispozitivului: înregistrearea sunetului se face folosind clasa *AudioRecord* care oferă metode pentru înregistrarea sunetului ne-compresat; melodia este înregistrată timp de 10 secunde contorizate cu ajutorul unui *CountDownTimer*, iar la finalul acestora este prelucrată;

-bucata de melodie înregistrată este scrisă într-o bază de date în timp real oferită de *Firebase API*, iar scrierea este făcută într-un *thread separat* de thread-ul principal;

-un *mini-server* scris în Python urmărește schimbările din această bază de date, iar când apare o melodie nouă o introduce în procesul de recunoaștere;

-utilizatorului i se afișează o listă cu melodia recunoscută și cu melodiile identificate ca asemănătoare și are posibilitatea de a le introduce într-o listă cu melodii favorite apăsând un buton;

-utilizatorul își poate vizualiza lista de melodii salvate ca favorite – ultimele două funcționalități fiind implementate cu ajutorul conceputului de *RecycleView*, fiecare cu un *Adapter* personalizat;

-pentru ca user-ul să nu fie plictisit cât așteaptă înregistrarea și prelucrarea, imaginea de pe pagina principală are o animație sub formă de puls aplicată cu ajutorul clasei *AnimationUtils*, iar statusul prelucrării se afișează la fiecare pas pe ecran.

**Algoritmul de recunoastere a melodiei** a fost implementat conform descrierii găsite la : coding-geek.com/how-shazam-works.

În linii mari acesta funcționează în felul următor:

1.Pentru fiecare aprox. 0.1 secunde de melodie înregistrată se aplică FastFourierTransform , se crează astfel spectrul aplitudinii raportat la frecvență.

2.Pentru fiecare 0.1 secunde de melodie se alege frecvența cu aplitudinea cea mai mare și se face media tuturor astfel de frecvențe pentru toată melodia

3.Pentru fiecare spectru se țin minte doar frecvențele și momentul în timp al acelor frecvențe ce au aplitudinea mai mare decât o constantă c1 \* media tuturor frecvențelor.

4. Cu lista de puncte (frecvența, moment de timp ) se creează zone țintă formate din 5 asemenea puncte consecutive ( ordonate dupa timp si dupa frecvență). Fiecare zonă țintă are o ancoră , și anume punctul ce are indexul = indexul primului punct în zona -3

5. Pentru fiecare zonă țintă și pentru fiecare punct din acea zonă țintă se creează o adresă de forma :

[frecvența ancorei; frecvența punctului , diferența de timp între ancoră și punct] -> [timpul absolut al ancorei în înregistrare]

6. Baza de date este de forma :

[frecvența ancorei; frecvența punctului , diferența de timp între ancoră și punct] -> [timpul absolut al ancorei în melodie, indexulMelodiei]

7. Pentru fiecare adresa din înregistrare, aceasta se folosește drept cheie pentru a găsi toate tuplurile [timp absolut al ancorei în melodie, indexulMelodiei] ce corespund acelei adrese

8. Fiind folosit dicționare, căutarea se face în O(n) unde n reprezintă numărul de adrese din înregistrare. Astfel e deosebit de eficient, și chiar pentru un număr foarte mare de adrese în care se caută, ( peste 600.000) timpul cel mai mare e consumat tot de transformata Fourier.

9. Se țin minte doar melodiile al căror index apare de mai mult decât o constantă c2 \* numărul total de adrese din înregistrare

10. Pentru fiecare melodie cu index ce se găsește în cele găsite și pentru fiecare adresa se calculează delta = timpul absolut al ancorei in inregistrare - timpul absolut al ancorei in melodie

11. Melodia recunoscuta va fi cea pentru care apare valoarea delta cel mai des.

**Algoritmul de similaritate funcționează în felul următor** :

1. Pentru fiecare aprox 0.1 secunde de melodie se aplică transforamata fourier și se obtine astfel spectrul

2. Pentru fiecare spectru se tine minte doar frecventa cea mai dominanta din interval.

3. Cu lista de frecvente domintante se creza o imagine alb negru , ce are pixel colorat negru la coordonatele(t,f) unde e t e momentul de timp al frecventei in melodie , si f e valoarea frecventei.

4. Pe imaginea creata se aplica metoda de perceptualHash pentru a returna o lista de hash-uri ce caracterizeaza melodia.

5. Pentru fiecare melodie din baza de data , se compara lista hashuri acelei melodii cu lista de hashuri tuturor celorlalte melodii din baza de data.

6. Compatia se face folosind dinstanta Hamming , pentru a da un factor de similaritate intre cele doua hashuri .

7. Valoarea de similaritate va fi suma acelei permutari de hashuri care genereaza valoarea cea mai mare.

8. Melodiile se ordoneaza descrescator dupa valoarea factorului de similairate si se tin minte cele mai mari 3 valori.

**3. STADIUL ACTUAL AL TEHNOLOGIEI**

Cele mai bune aplicații de recunoaștere muzicală la momentul actual pe piață sunt Shazam și Soundhound.

Shazam este una dintre cele mai populare aplicații din lume, utilizată de sute de milioane de oameni în fiecare lună pentru a identifica instantaneu muzica care se joacă și pentru a vedea ce descoperă ceilalți. Funcționalități pe care această aplicație le are în plus față de proiectul nostru:

• Afișare de versuri muzicale sau vizionare de videoclipuri

• Previzualizare melodii și adăugăre în lista de redare Spotify

• Shazam Offline: Identificare muzică, chiar și fără conectare la internet

• Recomandare piese pentru a găsi muzică nouă

• Recunoașterea vizuală Shazam: Atingeți pictograma aparatului foto pentru postere, reviste, cărți și multe altele

SoundHound este o aplicație similară (totuși mai puțin populară decât aplicația prezentată anterior) care vine echipată cu "OK Hound ...", un mod incredibil de puternic de a interacționa cu SoundHound prin voce. "OK Hound ..." ajută la navigarea în aplicație, la descoperirea de melodii noi și la redarea de melodii hands-free. Funcționalități oferite pe care CheatingShazam nu le oferă:

• Recunoaștere vocală: recunoașterea unei melodii cântate de către utilizator;

• Urmăriea descoperirilor prin salvarea automată a unei istorii personale

• Sugerare de melodii cântate în apropiere și în întreaga lume cu Harta muzicii

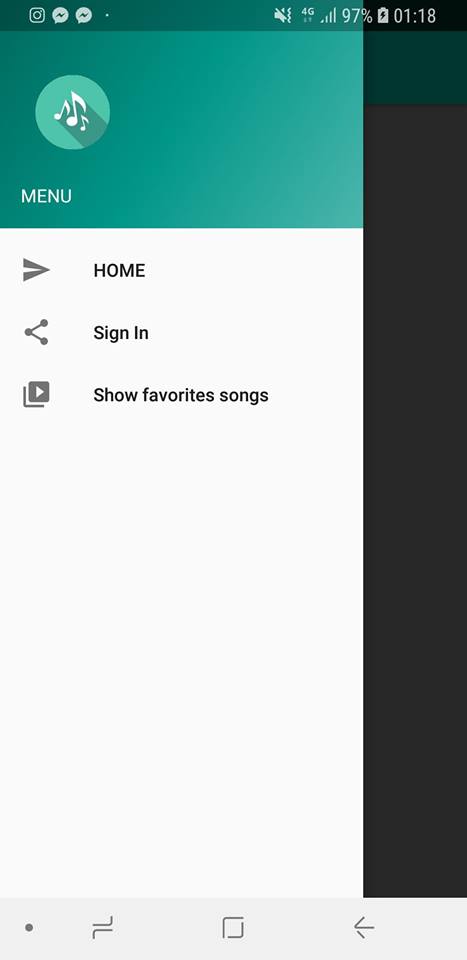
• Cântecele descoperite pot fi redate gratuit prin playerul YouTube încorporat

• Vizualizare de versuri atunci când se redă o melodie de pe YouTube

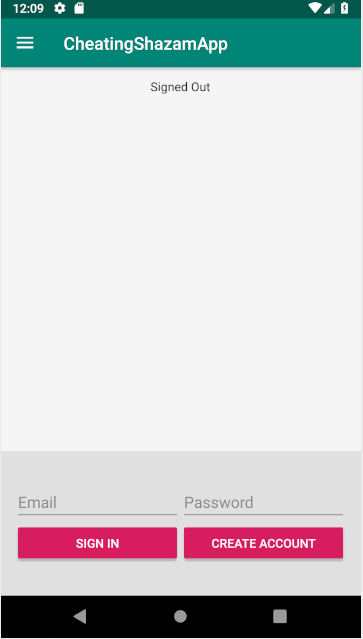
Deasemenea ambele aplicații permit publicarea din aplicație direct în alte aplicații de Social Media a descoperirlor făcute. Deși au toate aceste caracteristici bine dezvoltate și se descurcă în recunoaștere de melodii în timp foarte bun, niciuna din cele două nu caută asemănări între melodii – trăsătura pe care noi am vrut să o aducem în plus.

**4. UTILIZARE**

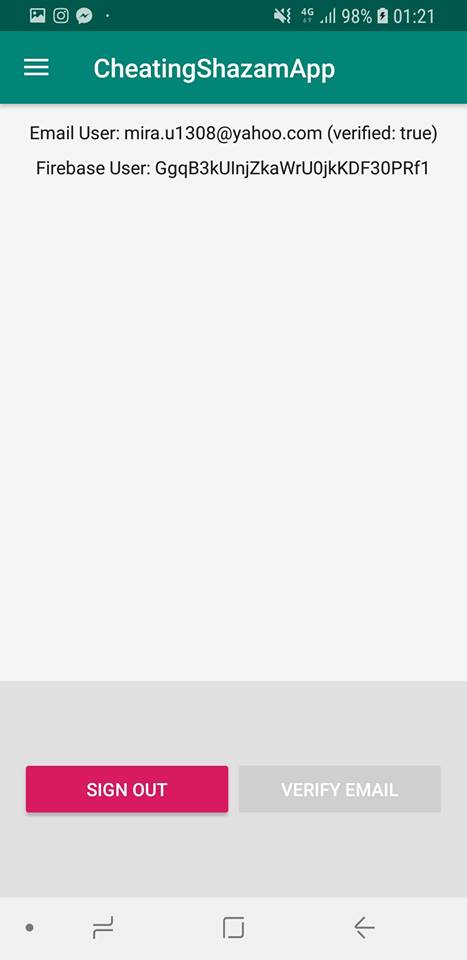
Meniul aplicației poate fi utilizat de pe oricare din înregistrări și permite navigarea între activitățile prin care utilizatorul interacționează cu aplicația.



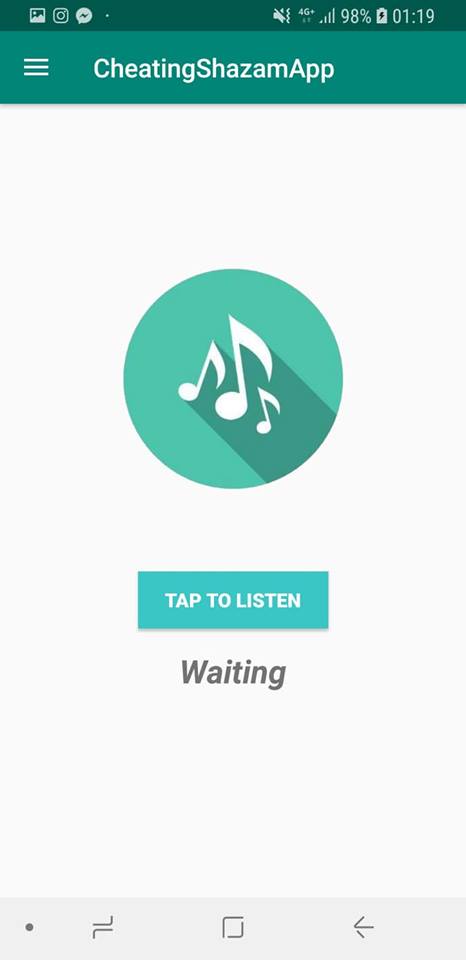
Activitatea de Sign in este cea care se afișează prima atunci când un utilizator încearcă să folosească aplicația și nu este logat. Acesta își poate crea un cont sau se poate loga cu un email și o parolă.



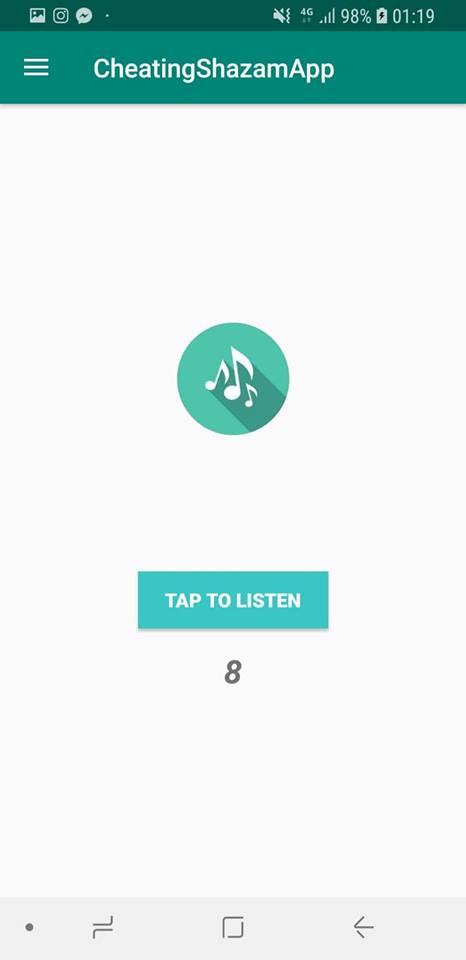
După logare, pagina se adaptează noii stări :



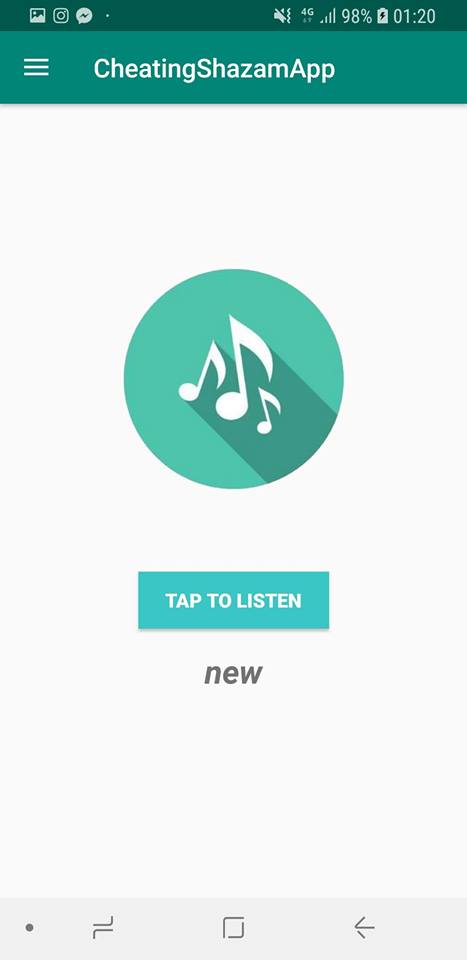
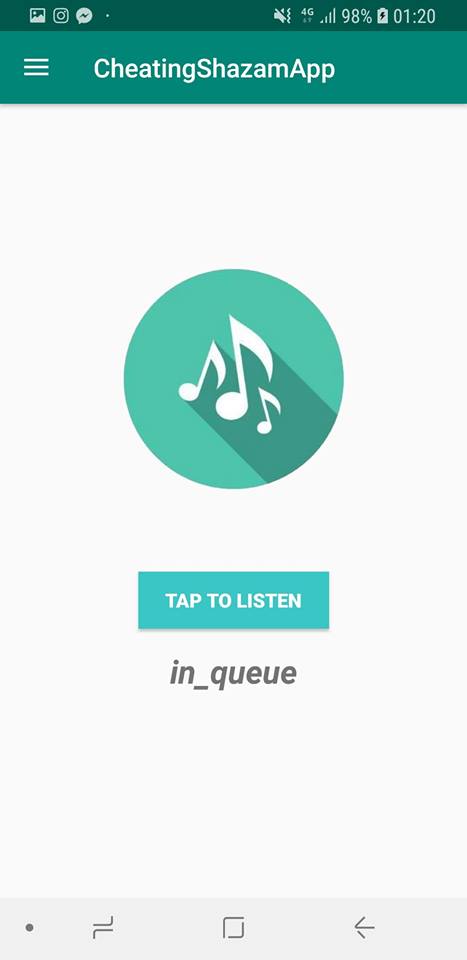
Unui utlizator deja logat i se afișează activitatea principală a aplicației, activitate care prezintă butonul ce permite înregistrarea melodiei, câmpul pe care se afișează starea procesării și animația sub formă de puls activă în timpul înregistrării.



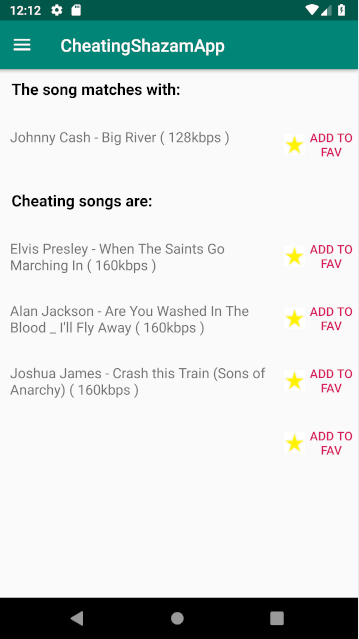
La apăsarea butonului Tap to listen, i se va cere utilizatorului permisiunea de a avea acces la microfon. Dacă o acordă, începe înregistrarea melodiei timp de 10 secunde în care animația este activă, iar pe ecran se afișează o numărătoare inversă.



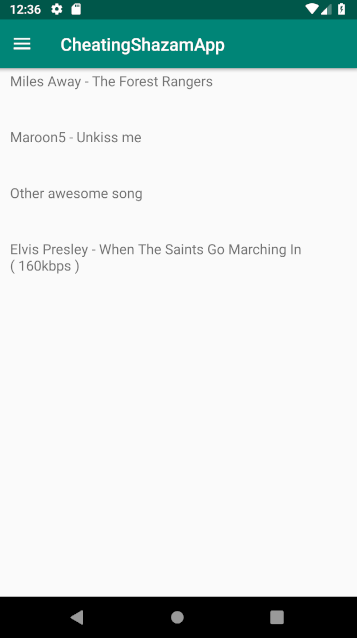
La sfărșitul numărătorii, statusul melodiei este setat pe new și începe procesarea acesteia pe ecran afișându-se etapa în care se află.

La sfârșitul procesări, adică în medie după 2.4 secunde, se afișează pe ecranul dispozitivului melodia identificată, dar și melodiile asemănătoare cu aceasta. Utilizatorul are posibilitatea de a le salva în lista lui de favorite pentru o vizualizare ulterioară.



Lista cu melodii favorite poate fi vizualizată oricând, navigarea la aceasta fiind făcută prin meniu.



**5. CONCLUZII**

Tehnologii pe care nu le-am mai uitilizat înainte de realizarea acestui proiect:

* limbajul Kotlin;
* sistemul de operare Android și componentele acestuia;
* interacțiunea cu baza de date Firebase.

Considerăm că cel mai greu a fost să alegem ce concepte și servicii sunt cele mai potrivite pentru nevoile aplicației noastre. Deasemena, algoritmul de recunoaștere al melodiilor asemănătoare a necesitat mai multă cercetare și testare decât am plănuit inițial.

Pasul următor este să îmbunătățim atât algoritmul, pentru o recunoaștere mai precisă, dar și să oferim utilizatorilor funcționalități în plus, cum ar fi posibilitatea de a asculta melodiilor salvate ca favorite sau de a vizualiza versurile acestora.