SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR, INFORMATIKA SZAK

IFD - Image Face Detector
Szoftver rendszerek projekt

Témavezető:
Dr. Szántó Zoltán
Végzős hallgatók:

Bereczki-Király Rómeo
Bressel Mark

2024

Tartalomjegyzék

1. Bevezető

A mai világban az arcfelismerés egyre nagyobb szerepet kap a különböző alkalmazásokban, legyen szó biztonsági rendszerekről, személyazonosság-ellenőrzésről vagy egyszerű felhasználói élmény javításról. Az arcfelismerő rendszerek segítenek abban, hogy gyorsan és pontosan azonosíthassuk az embereket különböző környezetekben, így nagyban hozzájárulnak a kényelmes és biztonságos technológiai megoldások kialakításához.

Az IFD (Image Face Detector) rendszer célja, hogy egy olyan megbízható és gyors arcfelismerő megoldást biztosítson, amely képes azonosítani az embereket különböző képeken. Az IFD egy intelligens eszköz, amely a legmodernebb algoritmusokat alkalmazza, hogy minél pontosabb és gyorsabb felismerést végezzen, miközben figyelembe veszi a különböző arcfelismerési kihívásokat, mint a fényviszonyok, a háttér és a személyközi távolságok.

Az arcfelismerés alkalmazása különösen fontos azokban a helyzetekben, ahol a biztonság és az autentikáció kulcsfontosságú, mint például a telefonok vagy számítógépek zárolásánál, valamint különböző beléptető rendszerekben. Az IFD képes nemcsak felismerni az arcokat, hanem fényképet készítve is nyomon tudja követni a képen szereplő arcokat.

Ezen kívül az IFD olyan felhasználói élményt biztosít, amely minden mobil eszközön zökkenőmentesen működik. A rendszer könnyen integrálható más alkalmazásokba, lehetővé téve a különböző funkciók egyszerű bővítését, mint például a személyre szabott ajánlások vagy a felhasználói viselkedés elemzése.

A jövőben az IFD célja, hogy a fejlesztők és a végfelhasználók számára egy olyan megbízható és könnyen használható eszközt biztosítson, amely hatékonyan támogatja a különböző arcfelismerési igényeket.

2. Projekt célja

A második fejezet a projekt célját és annak megvalósítását részletezi, melynek fő fókusza az arcfelismerés gyors és pontos végrehajtása különböző környezetekben. Az IFD (Image Face Detector) rendszer célja, hogy megbízható, skálázható és könnyen integrálható arcfelismerő megoldást kínáljon, amely a legmodernebb algoritmusok és technológiai újítások felhasználásával biztosítja az optimális működést.

A projekt célja nemcsak a megbízható arcfelismerés, hanem a különböző környezeti tényezők – mint például a fényviszonyok, a háttér, valamint az arcok közötti távolságok – figyelembevételével történő felismerés is. Ennek érdekében az IFD képes alkalmazkodni a különböző felhasználói igényekhez, legyen szó biztonsági alkalmazásokról, személyazonosság-ellenőrzésről vagy egyszerűbb felhasználói élmény javításáról.

A projekt célja továbbá, hogy az IFD gyorsan és pontosan végezzen felismerést, miközben minimalizálja a hibalehetőségeket és a feldolgozási időt. A rendszer célja, hogy a felhasználók zökkenőmentes élményben részesüljenek, függetlenül attól, hogy mobil eszközön vagy számítógépen használják a rendszert.

A projekt hosszú távú célja, hogy az IFD folyamatosan fejlődjön, figyelembe véve az arcfelismerési algoritmusok fejlődését és az újabb technológiai lehetőségeket. A cél egy olyan platform létrehozása, amely könnyen adaptálható különböző alkalmazásokhoz, és képes támogatni a változó felhasználói igényeket, például a személyre szabott ajánlásokat vagy a felhasználói viselkedés elemzését.

A projekt megvalósításával kapcsolatos konkrét célok:

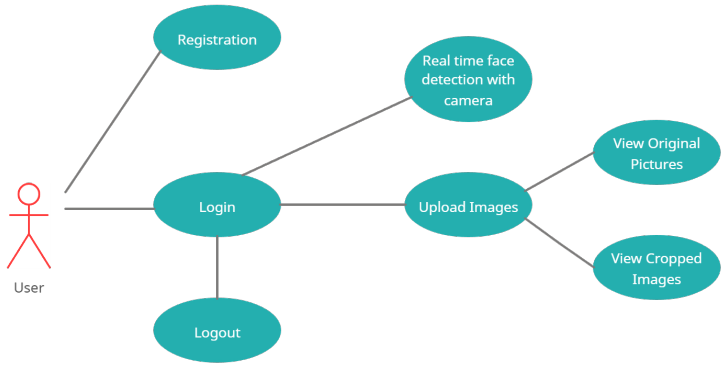
1. **Pontos arcfelismerés**: A rendszer képes felismerni az arcokat különböző fényviszonyok, háttér és távolságok mellett.
2. **Gyors működés**: Az arcfelismerési folyamat gyorsan és hatékonyan történik, minimalizálva a késleltetést.
3. **Mobil eszközökhöz való alkalmazkodás**: A rendszer minden mobil eszközön zökkenőmentesen működik, biztosítva a felhasználói élményt.
4. **Könnyű integrálhatóság**: Az IFD könnyen integrálható más alkalmazásokba, lehetővé téve a további funkciók bővítését.

A projekt célja, hogy a felhasználók számára egy olyan eszközt biztosítson, amely nemcsak a jelenlegi igényeket szolgálja ki, hanem a jövőbeli arcfelismerési technológiai újításokat is támogatja.

3. Követelmény specifikációk

3.1. Felhasználói követelmények. Use case diagram.

User és lehetőségei az alkalmazásban, mint ahogy az 1 -es ábrán láthatjuk. A követelmény-specifikációk az alkalmazás felhasználói igényeit tartalmazzák. Az alábbi ábra bemutatja a felhasználót és az interakciókat.



1. ábra. Use case diagram, amely bemutatja a User-t és az interakciókat.

3.1.1. Use-Case forgatókönyvek

Bejelentkezés:

- A felhasználó megadja az email-címét és jelszavát.
- A rendszer hitelesíti az adatokat a Firebase Authentication segítségével.
- Ha a hitelesítés sikeres, a felhasználó a kezdőképernyőre kerül, ahol elérheti az összes funkciót.

Regisztráció:

- A felhasználó megadja az email-címét és jelszavát.
- A rendszer létrehozza a felhasználói fiókot a Firebase-ben.

- A regisztráció után a felhasználó visszairányításra kerül a bejelentkezési képernyőre.

Kép feltöltése:

- A felhasználó kiválaszt egy képet a galériából.
- A rendszer elküldi a képet a backendnek.
- A backend felismeri az arcokat, kivágja őket, majd a képek URL-jeit eltárolja a Firebase-ben.
- A rendszer visszaadja az URL-eket, amelyeket a felhasználó megtekinthet.

Valós idejű kamera:

- A felhasználó bekapcsolja a kamerát.
- A kamera valós időben detektálja az arcokat az expo-face-detector modul segítségével.
- Ha arcot talál, a rendszer kiemeli vagy rögzíti a képet.

Feltöltött képek megtekintése:

- A felhasználó lekéri az általa feltöltött képeket a Firebase Storage-ből.
- A képek megjelennek egy görgethető listában.

Kivágott arcok megtekintése:

- A felhasználó lekéri a kivágott arcok URL-jeit a backendről.
- A képek megjelennek egy listában.

Kijelentkezés:

- A felhasználó a kijelentkezés gombra kattint.
- A Firebase Authentication API megszakítja a munkamenetet, és a felhasználót kijelentkezteti.

3.1.2. Use-Case Diagram vizuális elemei

Aktor (Felhasználó):

- Egy pálcika figura a diagram bal oldalán, amely az összes funkciót (use-case) vezérli.

Use-Case-ek:

- Ovális alakú elemek, amelyek a funkciókat képviselik (pl. Bejelentkezés, Kép feltöltése).

Kapcsolatok:

- Egyenes vonalak kötik össze az aktort és a use-case-eket.

Egy ilyen diagram jól szemlélteti az alkalmazás funkcionalitását és a felhasználók által végrehajtható folyamatokat.

3.2. Rendszerkövetelmények

3.2.1. Funkcionális

Az IFD alkalmazás funkcionális követelményei azok a képességek, amelyek biztosítják, hogy a rendszer minden szükséges funkcióval rendelkezzen a felhasználók számára.

1. **Arcfelismerés:** Az alkalmazás képes az arcok valós idejű felismerésére képek, illetve élő kamera képek alapján. Az arcokat pontosan kell detektálni a fényviszonyok, háttér és távolságok figyelembevételével.

2. **Kép feltöltése és tárolása:** A felhasználók képeket tölthetnek fel a rendszerbe, amelyeket a backend tárol. A képek URL-jeit a rendszer rögzíti és megjeleníti a felhasználó számára.

3. **Valós idejű kamera használata:** A rendszer képes a felhasználó mobil eszközeinek kameráját valós időben használni az arcok detektálására, és rögzíteni az arcokat.

4. **Bejelentkezés és regisztráció:** A felhasználók bejelentkezhetnek és regisztrálhatnak a Firebase Authentication segítségével, hogy személyre szabott élményt kapjanak.

5. **Képkeresés és arcok megtekintése:** A felhasználók kereshetnek és megtekinthetik az általuk feltöltött képeket, valamint a kivágott arcokat.

6. **Kijelentkezés:** A felhasználó kijelentkezhet a rendszerből, megszakítva a munkamenetet.

3.2.2. Nem-funkcionális

A nem-funkcionális követelmények a rendszer teljesítményére és a felhasználói élmény minőségére vonatkoznak. Az alábbiakban azokat a szempontokat foglaljuk össze, amelyek biztosítják, hogy az alkalmazás stabilan, gyorsan és biztonságosan működjön.

1. **Teljesítmény:** Az alkalmazás gyors válaszidővel rendelkezik, különösen a képfeldolgozás és arcfelismerés során. Az arcfelismerési folyamat 1 másodpercen belül véget ér.

2. **Skálázhatóság:** A rendszer képes skálázódni a nagy felhasználói bázis kiszolgálására, különösen a képek feltöltésének és tárolásának kezelésében.

3. **Biztonság:** Az alkalmazás biztosítja a felhasználói adatokat, beleértve a személyes adatokat, jelszavakat és képeket. Az adatok titkosítása szükséges mind a hálózaton való átvitel, mind a tárolás során.

4. **Használhatóság:** A felhasználói felület egyszerű és intuitív, hogy a felhasználók könnyen navigálhassanak az alkalmazás különböző funkciói között.

5. **Kompatibilitás:** Az alkalmazás zökkenőmentesen működik Android rendszeren, valamint támogat különböző Androidos eszközöket (telefonokat, táblagépeket).

6. **Hibatűrés:** Az alkalmazás képes a hibák kezelésére és a rendszer stabilitásának fenntartására, még akkor is, ha a felhasználó problémákkal találkozik, például a képfeltöltés során.

7. **Hardver:** Az alkalmazás futtatásához az alapvető rendszerkövetelmény az okostelefon, legalább 2 GB RAM-mal és 1 GHz-es processzorral. A felhasználói élmény optimalizálása érdekében ajánlott egy gyorsabb processzor és legalább 4 GB RAM. Android 4.1 (Jelly Bean) vagy újabb.

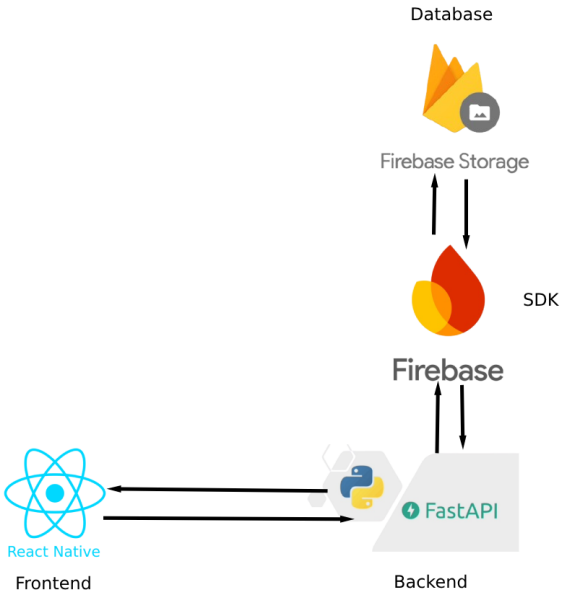
8. **Hálózati kapcsolat:** Stabil internetkapcsolattal kell rendelkeznie az alkalmazás gyors és gördülékeny működése érdekében.

4. Tervezés

4.1. Architektúra. Komponens diagram

Az alkalmazásunk architektúrája a modern fejlesztési technológiákra épít, amelyek lehetővé teszik a rugalmas és hatékony fejlesztést. A gyors válaszidő, stabilitás és hatékonyság kiemelt szempontok voltak a tervezés során. Az alábbiakban részletesen bemutatom a kliens- és szerveroldali fejlesztést.

A 2-es ábrán látható a komponens diagramunk.



2. ábra. Komponens diagram

Az alkalmazás a következő főbb részekből épül fel:

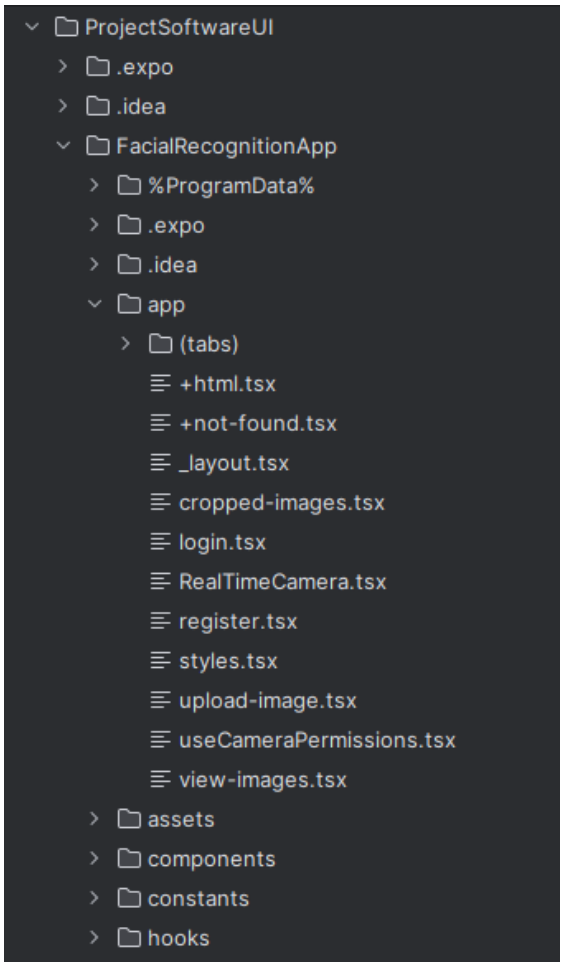
- **Szerver oldal:** Python FastAPI
- **Kliens oldal:** React-Native mobilalkalmazás
- **Adatbázis:** Firebase
- **Adatok begyűjtése:** Firebase Storage

4.1.1. Modulok leírása

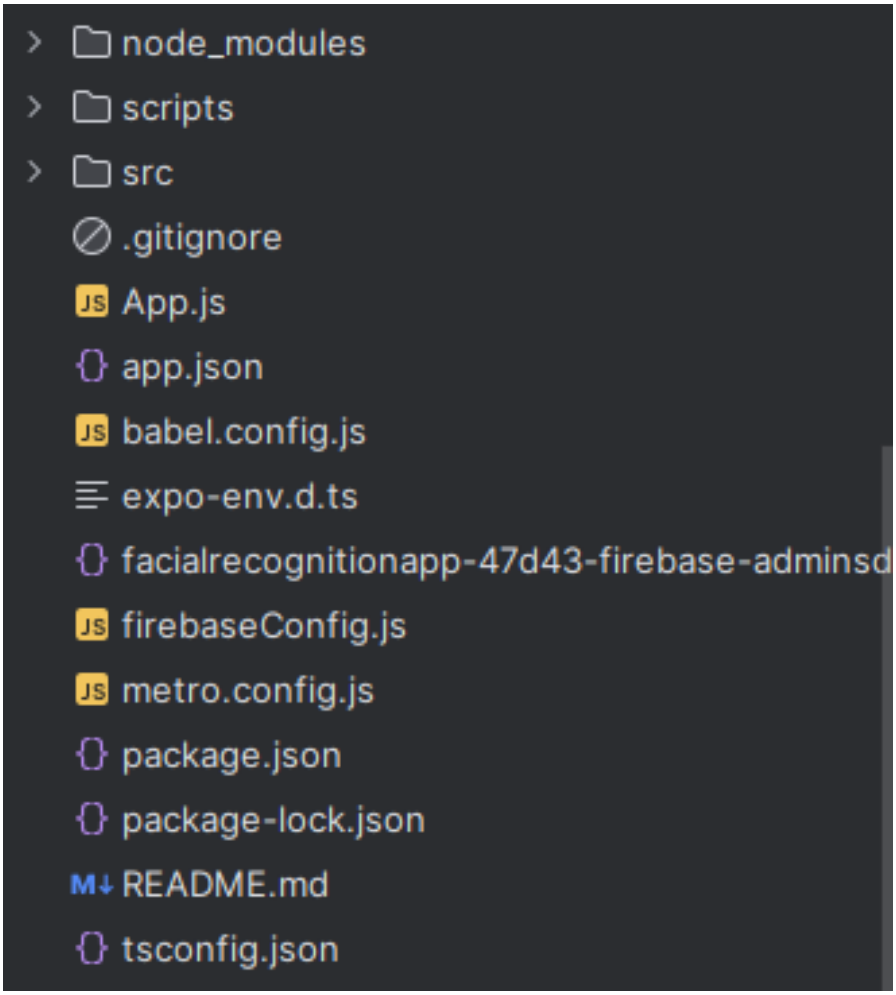
A rendszer moduljai négy fő kategóriába sorolhatók, amelyek külön-külön és együttműködve biztosítják az alkalmazás teljes funkcionalitását:

Frontend modul:

- Felhasználói interakciók kezelése.
- Képkezelési és navigációs funkciók biztosítása.
- Firebase hitelesítés és adatok megjelenítése.



3. ábra. Frontend modul ábrázolása - 1



4. ábra. Frontend modul ábrázolása - 2

UI terv

Az UI terv az IFD alkalmazásunknál egyszerű, intuitív és modern megjelenését célozza meg. Az alkalmazás három fő navigációs csoportot követ:

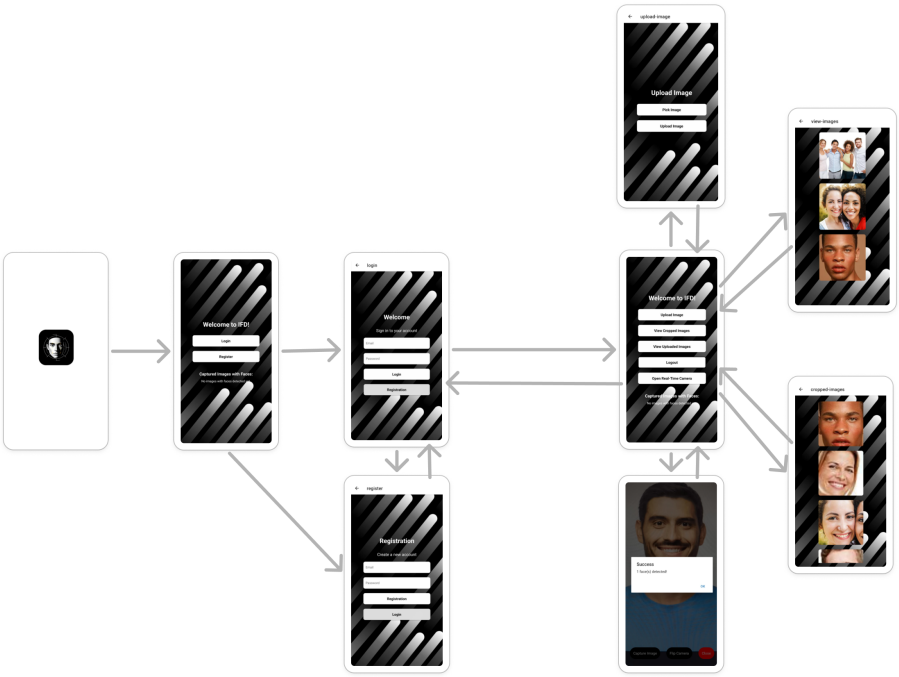
- Hitelesítés
- Képkezelés
- Arcfelismerési eredmények megtekintése
- **Főoldal (HomeScreen)**

  - **Fő elemek:**
    - Üdvözlő szöveg.
  - **Gombok:**
    - "Kép feltöltése": Navigáció a feltöltési képernyőre.
    - "Valós idejű kamera": Navigáció a kamerához.
    - "Feltöltött képek": Az összes feltöltött kép listázása.
    - "Kivágott arcok": A felismerési eredmények megtekintése.

- **Bejelentkezés (LoginScreen)**
  - **Felhasználói élmény:**
    - Egyszerű form email- és jelszómezőkkel.
  - **Gombok:**
    - Bejelentkezés
    - Regisztráció

- **Kivágott arcok (CroppedImagesScreen)**
  - **Funkció:**
    - Grid nézet a felismerési eredmények (kivágott arcok) megjelenítéséhez.
    - Betöltési indikátor (ActivityIndicator) a lassú hálózati kapcsolat esetére.

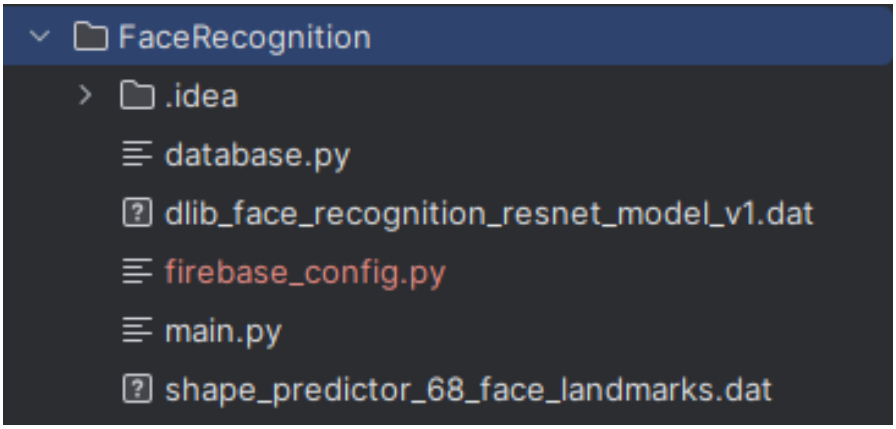
- **Valós idejű kamera (RealTimeCamera)**
  - **Felület:**
    - Teljes képernyős kamera.
  - **Gombok:**
    - "Kép készítése": Pillanatkép rögzítése.
    - "Kamera váltása": Előlapi és hátlapi kamera közötti váltás.



5. ábra. Az UI terv vizuális ábrázolása

Backend modul:

- Python és FastAPI alapú feldolgozó logika.
- Képfeldolgozás (arcfelismerés), metaadatok létrehozása.
- REST API végpontok a frontend és adatbázis kiszolgálására.



6. ábra. Backend modul ábrázolása

Adatkezelési modul:

- Firebase Authentication: Felhasználók hitelesítése.
- Firebase Storage: Eredeti és kivágott képek tárolása.
- Firestore: Metaadatok strukturált tárolása.

Kamera modul:

- Expo Camera API használata a valós idejű arcfelismeréshez.
- Arcok detektálása az expo-face-detector segítségével.

4.2. Technológiai döntések

A technológiák kiválasztásánál az egyszerű fejlesztés és a skálázhatóság volt a cél:

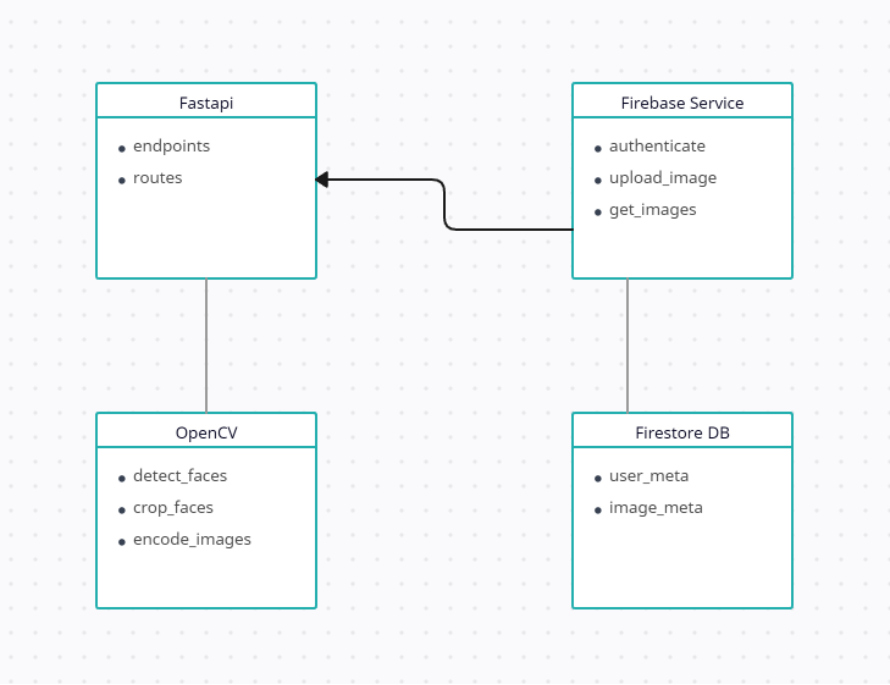
- **React Native + Expo:** Platformfüggetlen mobilfejlesztés és gyors prototípus-készítés.
- **FastAPI:** Könnyen kezelhető és gyors Python-alapú backend keretrendszer.
- **Firebase:** Autentikáció és adatbázis kezelésére egyszerű és megbízható megoldás.

4.3. Osztálydiagramok és Adatbázis Terv

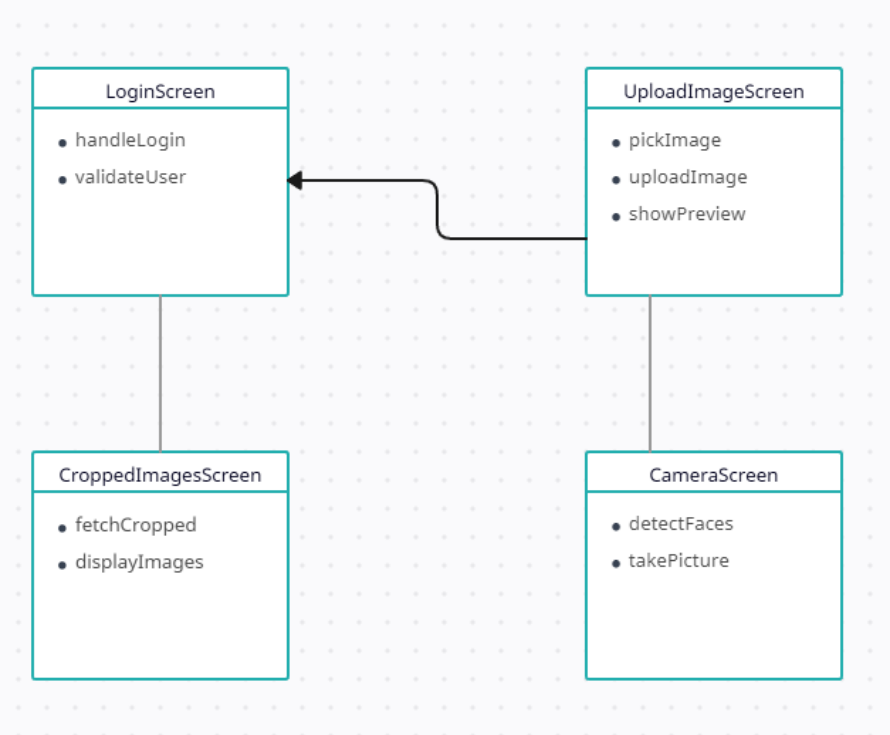
A rendszer több kulcsfontosságú komponensből áll, amelyek egymással szorosan együttműködnek:

- **FastAPI**: Felelős az összes REST API végpont kezeléséért. Ez biztosítja az adatforgalmat és a backend logikát.
- **OpenCV**: Az arcfelismerési logikáért felelős, beleértve az arcok detektálását és kivágását a képeken.
- **Firebase Service**: A képek és azok metaadatainak tárolását végzi a **Firebase Storage** és a **Firestore** segítségével.
- **Frontend Modul**: A képernyőkre osztott, amelyek mindegyike egy-egy funkciót biztosít, mint például a bejelentkezés, kép feltöltése, vagy az arcfelismerés megjelenítése.
- **Kommunikáció a Backenddel**: A képernyők folyamatos kapcsolatban állnak a backenddel és az Expo modulokkal, például a kamerával.

Az alábbi osztálydiagramok bemutatják a rendszer különböző komponenseit és azok közötti kapcsolatokat.



7. ábra. Osztálydiagram - 1



8. ábra. Osztálydiagram - 2

4.4. Verzió követő és projekt menedzsment

Az *IFD* fejlesztését egy jól strukturált projektmenedzsment rendszer és verziókövetési stratégia támogatta. A *Jira board*-ot használtuk a feladatok követésére és elosztására, míg a *GitHub* biztosította a kód verziókezelését és a csapatmunka hatékony lebonyolítását.

4.4.1. Jira Board

A *Jira board*-ot alkalmaztuk, hogy a projekt állapotát és a feladatok előrehaladását vizualizálni tudjuk. Az oszlopok segítségével pontosan nyomon követhettük, mely feladatokon dolgoztunk, melyek álltak készen, és melyek vártak megoldásra.

Kanban oszlopok:

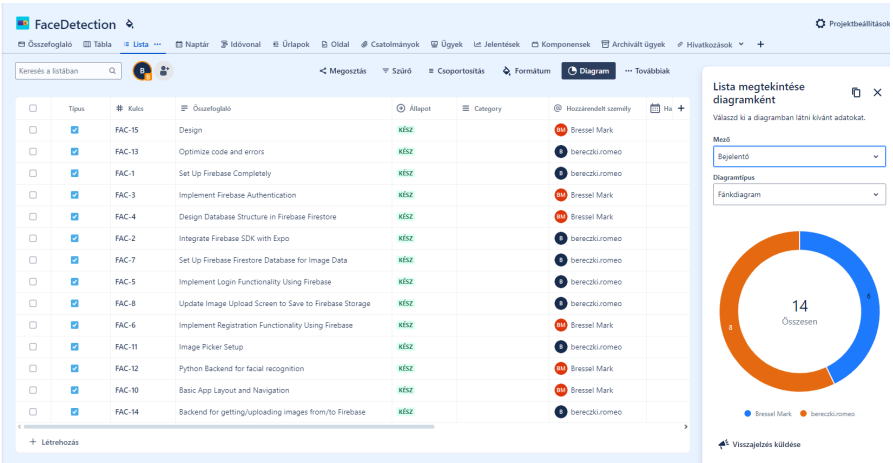
- **Backlog**: Minden új ötletet, funkciót vagy problémát ide vettünk fel.
  - Példák: "Firebase Authentication beállítása.", "CroppedImagesScreen megjelenítése.", "FastAPI REST API végpontok tesztelése."
- **To Do**: A megvalósításra váró, prioritást kapott feladatokat helyeztük ide.

  - Példák: "Expo Camera integrálása.", "Regisztrációs funkció hibajavítása."

- **In Progress**: A jelenleg folyamatban lévő munkák kerültek ebbe az oszlopba. Az egyes taskokhoz hozzárendeltük a felelős fejlesztőt.
  - Példák: "Valós idejű kamera tesztelése.", "Arcfelismerés integrálása az OpenCV segítségével."
- **Code Review**: Az elkészült, de még ellenőrzésre váró feladatokat ide mozgattuk. A csapattagok ellenőrizték egymás munkáját, mielőtt a változtatások bekerültek volna a GitHub branch-ekbe.
- **Done**: A befejezett, tesztelt és már a GitHub fő branch-be integrált feladatokat ide helyeztük.

Jira előnyei:

- **Átláthatóság**: A feladatok állapotát egyértelműen láthattuk a boardon, és pontosan tudtuk, melyik task melyik fejlesztőnél van.
- **Csapatmunka támogatása**: A taskokhoz kommenteket adhattunk, és a Jira automatikusan szinkronizált az issue-k státuszával.
- **Prioritáskezelés**: Könnyedén priorizálhattuk a feladatokat (pl. kritikus hibák, új funkciók).



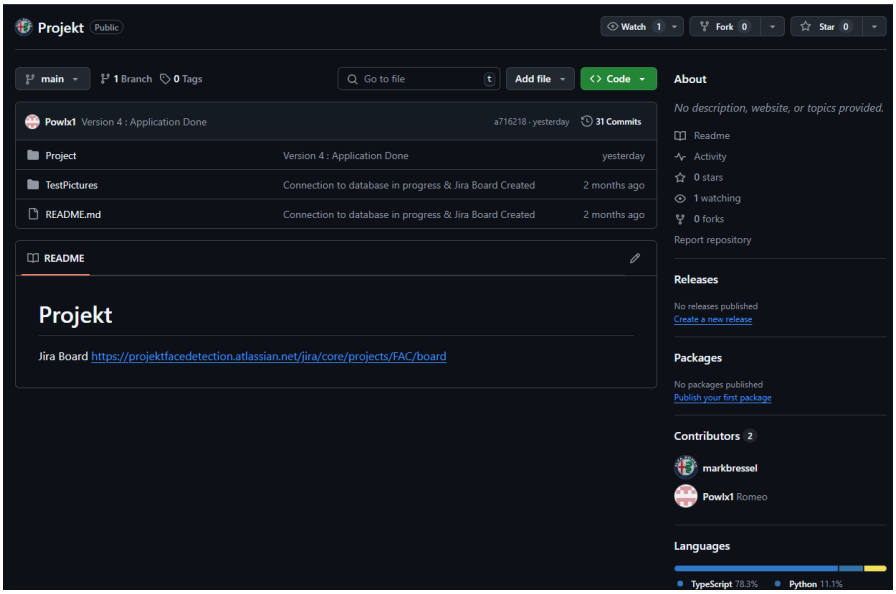
9. ábra. Jira Board példa

4.4.2. GitHub verziókezelés

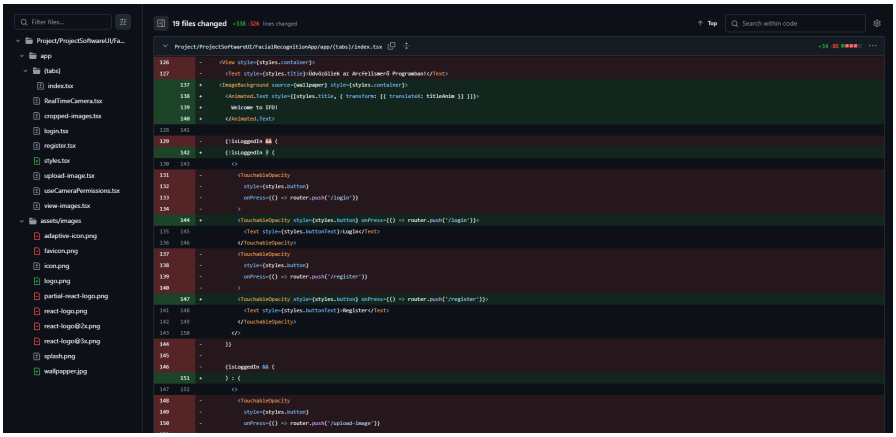
A kód verziókövetéséhez a *GitHub*-ot használtuk. A csapatmunka során fontos volt egy megbízható rendszer, amely támogatja a párhuzamos fejlesztést és az egyszerű összeolvasztást.

GitHub előnyei:

- **Több fejlesztő egyidejű munkájának támogatása**: A branch-struktúra lehetővé tette a párhuzamos fejlesztést.
- **Kódellenőrzés**: A PR-ek és a code review-k biztosították a kód minőségét.
- **Visszakövethetőség**: A commit üzenetek és issue kapcsolatok segítettek a változások visszavezetésében.



10. ábra. GitHub projekt



11. ábra. GitHub módosítások

5. Alkalmazás működése

Az alábbi szakasz részletesen bemutatja az FacialRecognitionApp működését képernyőnként, alátámasztva a rendszer által nyújtott funkciók részletes magyarázatával és illusztrációival. Minden alfejezethez egy képernyőképet mellékelünk, amely vizuálisan bemutatja az alkalmazás működését.

5.1. Üdvözlőképernyő (Splash Screen)

Az üdvözlőképernyő (Splash Screen) az alkalmazás betöltésekor jelenik meg, és egy gyors, vizuális bevezetést nyújt. Ez a képernyő célja, hogy bemutassa az alkalmazás logóját és elnevezését, miközben az alapvető inicializálási folyamatok zajlanak a háttérben.

Funkciók:

- Az alkalmazás logójának megjelenítése.
- A háttérfolyamatok, például a Firebase inicializálása.
- Automatikus navigáció a Bejelentkezési képernyőre, ha az alkalmazás betöltött.

Megvalósítás: Az üdvözlőképernyő az `expo-splash-screen` segítségével valósult meg. Az inicializálás után egy időzített navigációs logika (`setTimeout`) vezeti át a felhasználót a bejelentkezési felületre.

12. ábra. Üdvözlőképernyő

5.2. Bejelentkezés és Regisztráció

A bejelentkezési és regisztrációs képernyők teszik lehetővé a felhasználók számára, hogy hozzáférjenek az alkalmazás funkcióihoz.

5.2.1. Bejelentkezési képernyő (LoginScreen)

Funkciók:

- A felhasználók email-címük és jelszavuk megadásával hitelesíthetik magukat a Firebase Authentication segítségével.
- Sikertelen bejelentkezés esetén hibaüzenet jelenik meg.

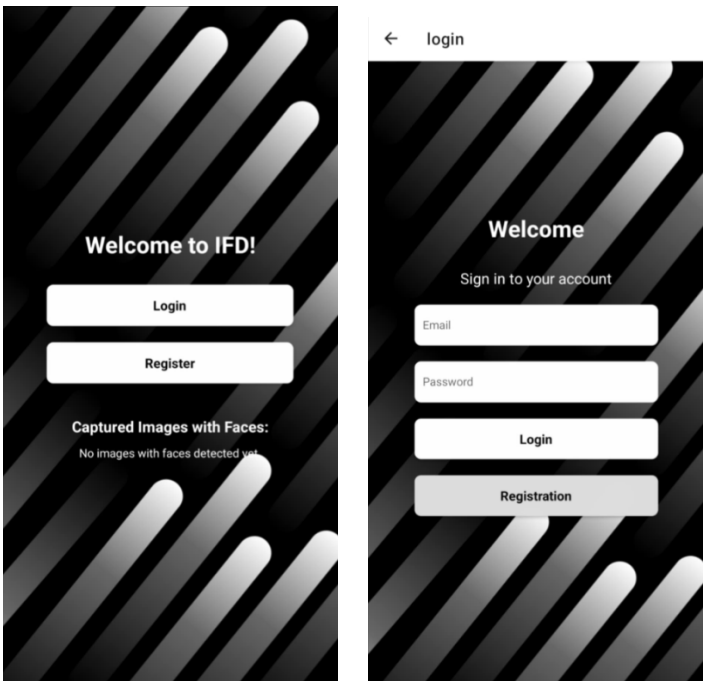
Megvalósítás: Az email-cím és jelszó validációját JavaScript-alapú regex ellenőrzés végzi. A hitelesítési folyamat az `signInWithEmailAndPassword` Firebase függvény segítségével történik.

5.2.2. Regisztrációs képernyő (RegisterScreen)

Funkciók:

- Új felhasználók hozhatják létre fiókjukat.
- A regisztráció során a Firebase elmenti a felhasználó email-címét és titkosított jelszavát.

Megvalósítás: A beviteli mezők validálása biztosítja, hogy az email-cím érvényes legyen. Sikeres regisztráció után a felhasználó átirányításra kerül a bejelentkezési képernyőre.



13. ábra. Bejelentkezés és Regisztráció képernyő

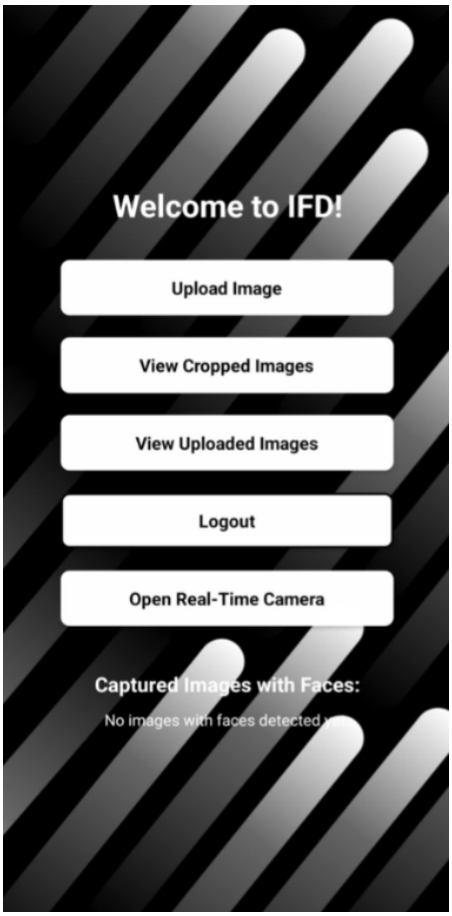
5.3. Home képernyő

A `HomeScreen` az alkalmazás központi navigációs képernyője, ahol a felhasználók kiválaszthatják, hogy melyik funkciót kívánják használni.

Funkciók:

- Navigáció az alkalmazás főbb funkciói között: Kép feltöltése, Valós idejű kamera használata, Feltöltött képek megtekintése, Kivágott arcok megtekintése, Kijelentkezési lehetőség.

Megvalósítás: A navigációs gombokat az `expo-router` kezeli. A képernyőn található gombok animációval jelennek meg.



14. ábra. Home képernyő

5.4. Upload Image képernyő

Az `UploadImageScreen` lehetővé teszi a felhasználók számára, hogy képeket töltsenek fel a galériából, majd azokat a rendszer feldolgozza.

Funkciók:

- Kép kiválasztása az eszköz galériájából.
- A kiválasztott kép előnézetének megjelenítése.
- A kép feltöltése a backendre és a Firebase Storage-be.

Megvalósítás: A kép kiválasztása az `expo-image-picker` modullal történik. A képek metaadatai a Firebase Firestore-ba kerülnek.

15. ábra. Upload Image képernyő

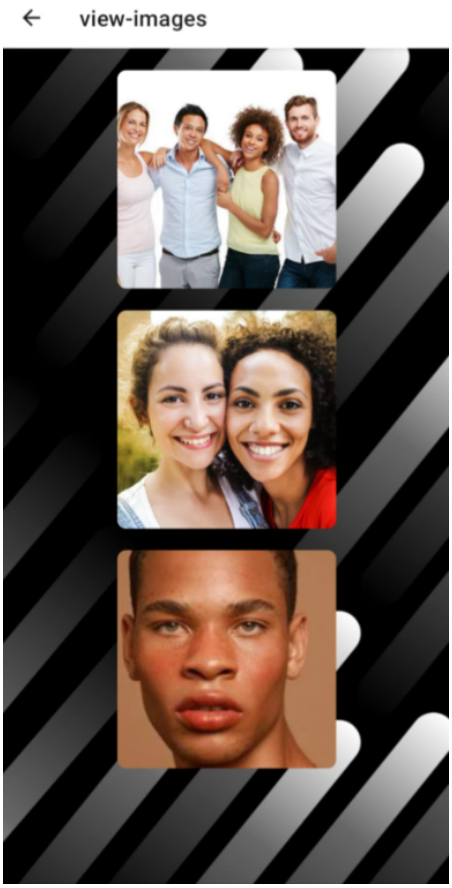
5.5. View Image képernyő

A `ViewImagesScreen` lehetőséget nyújt a felhasználók számára, hogy megtekinthessék a korábban feltöltött képeiket.

Funkciók:

- A feltöltött képek URL-jeinek lekérése a Firebase Firestore-ból.
- Görgethető lista megjelenítése az összes feltöltött képpel.

Megvalósítás: A képek dinamikus listázása a React Native `FlatList` komponensével történik. Az adatok frissítése valós időben történik.



16. ábra. View Image képernyő

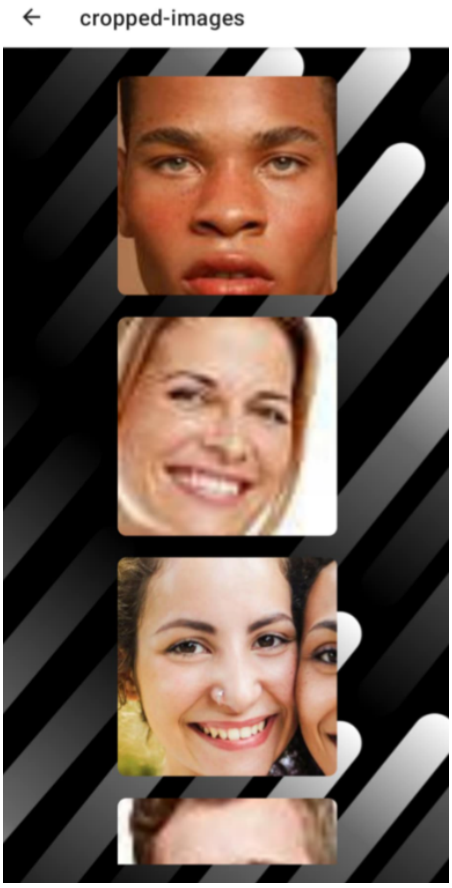
5.6. Cropped Image képernyő

A `CroppedImagesScreen` megjeleníti a feltöltött képekből kivágott arcokat.

Funkciók:

- A kivágott arcok URL-jeinek lekérése a Firestore-ból.
- Görgethető nézet az összes kivágott arccal.

Megvalósítás: A képek megjelenítése szintén a `FlatList` segítségével történik. Az arcfelismerési metaadatok (pl. detektálási idő) külön megjeleníthetők.



17. ábra. Cropped Image képernyő

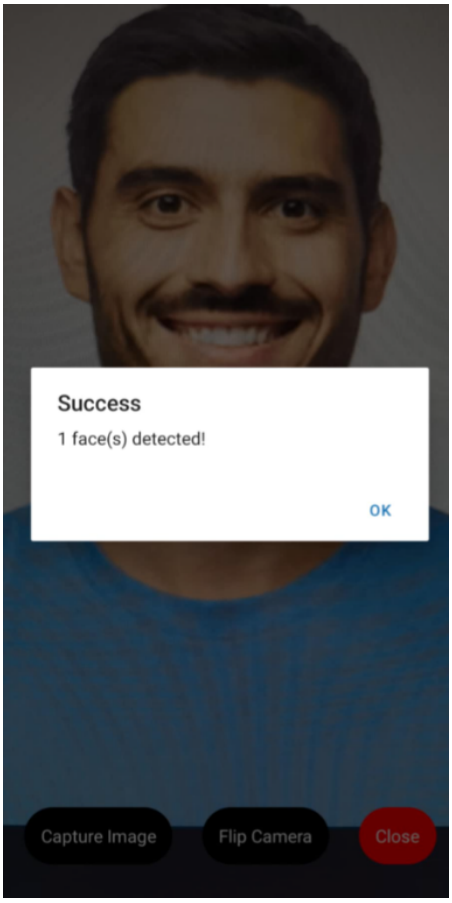
5.7. Real Time Camera képernyő

A `RealTimeCameraScreen` a valós idejű arcfelismerés funkciót biztosítja, amely során a felhasználók a kamerával készített képeiket közvetlenül a rendszerbe küldhetik.

Funkciók:

- Élő kamerakép megjelenítése.
- Arcok detektálása a kamerán keresztül az `expo-face-detector` segítségével.
- Pillanatkép készítése és annak mentése.

Megvalósítás: A kamera működését az `expo-camera` kezeli. Az arcfelismerési koordináták alapján dinamikusan rajzolunk kereteket a képernyőn.



18. ábra. Real Time Camera képernyő

6. Összefoglaló

Az IFD egy React Native és Expo alapú mobilalkalmazás, amely modern technológiákat használva kínál arcfelismerési és képfeldolgozási megoldásokat. A rendszer a Firebase és a FastAPI integrációjával valósítja meg a felhasználók hitelesítését, a képek tárolását, valamint az arcfelismerési funkciók végrehajtását. Az alkalmazás célja, hogy intuitív felhasználói felülettel és hatékony háttérlogikával biztosítson kiváló élményt.

6.1. További fejlesztési lehetőségek

1. **Továbbfejlesztett arcfelismerési funkciók**
    - *Emotion Detection*: Az arcfelismerés kiegészítése érzelemfelismeréssel

(pl. boldogság, szomorúság detektálása).

- *Age and Gender Estimation*: Becslés az arcról a felhasználó életkorára és nemére vonatkozóan.
- *Arcazonosítás (Face Recognition)*: Lehetőség arra, hogy az alkalmazás felismerje és azonosítsa a korábban regisztrált arcokat.

2. **Felhasználói élmény javítása**

   - *Offline mód*: A képek feldolgozásának lehetősége internetkapcsolat nélkül, az eredmények későbbi szinkronizálásával.
   - *Sötét mód*: A felhasználói felület automatikus átváltása sötét módba, a felhasználó preferenciái alapján.
   - *Valós idejű visszajelzés*: Az arcfelismerés során élőben megjeleníteni az észlelt arcok körvonalát és a felismerési eredményeket.

3. **Backend fejlesztések**

   - *Mesterséges intelligencia modell integrálása*: Az OpenCV-t kiegészítve modern AI modellek, például a Mediapipe vagy a Dlib használatával pontosabb és gyorsabb felismerést lehetne elérni.
   - *Skálázhatóság*: Az alkalmazás backendjének áthelyezése felhőalapú konténeres infrastruktúrára (pl. Kubernetes vagy AWS Lambda), hogy nagyobb terhelés mellett is stabilan működjön.
   - *Adatok anonimizálása*: A felhasználók adatainak GDPR-kompatibilis kezelése, beleértve a képek és metaadatok anonimizálását.

4. **Társadalmi funkciók**

   - *Közösségi megosztás*: A feldolgozott képek megosztása közvetlenül közösségi platformokon (Facebook, Instagram).
   - *Felhasználói ranglisták*: Egyes funkciókhoz (pl. hány képet dolgoztak fel) gamifikációs elemek hozzáadása.

5. **Fejlett statisztikák és elemzések**

   - *Statisztikai jelentések*: Az arcfelismerési eredmények részletes elemzése diagramokkal és grafikonokkal.
   - *Adatok időbeli alakulása*: Idővonalas grafikonok, amelyek mutatják az arcfelismerés fejlődését (pl. heti/havi képfeldolgozások száma).

6. **Több platformos támogatás**

- *Webes alkalmazás*: Az alkalmazás kiterjesztése böngészőben futtatható változatra.
- *Asztali alkalmazás*: Külön verzió készítése Windows és Mac rendszerekre.

7. **Integráció harmadik fél szolgáltatásaival**

   - *Google Photos integráció*: Lehetővé tenné a képek közvetlen importálását és exportálását a felhasználók Google Photos tárhelyéről.
   - *Machine Learning API-k használata*: Például az Amazon Rekognition vagy a Microsoft Azure Face API integrálása további képességekért.

7. Ábrák jegyzéke

8. Irodalomjegyzék

1. Python Software Foundation. *Python Language Reference, version 3.x*. Available at: `https://www.python.org/doc/`.

2. Firebase. *Firebase Documentation*. Available at: `https://firebase.google.com/docs`.

3. FastAPI. *FastAPI Documentation*. Available at: `https://fastapi.tiangolo.com/`.

4. Firestore. *Cloud Firestore Documentation*. Available at: `https://firebase.google.com/docs/firestore`.

5. OpenCV. *OpenCV Documentation*. Available at: `https://opencv.org/`.

6. React Native. *React Native Documentation*. Available at: `https://reactnative.dev/docs/getting-started`.

7. Expo. *Expo Documentation*. Available at: `https://docs.expo.dev/`.