FATAL LOVE TÁRSKERESŐ ALKALMAZÁS - DOKUMENTÁCIÓ

1. BEVEZETÉS

A Fatal Love egy modern társkereső alkalmazás, amelyet háromfős csapatunkkal fejlesztettünk. Az alkalmazás célja egy olyan platform létrehozása volt, ahol a felhasználók könnyen és biztonságosan találhatnak partnert.

Célközönség:

- Fiatal felnőttek

- A modern technológiákat kedvelők

- Biztonságos online ismerkedést keresők

A Fatal Love alkalmazás elsősorban a fiatal felnőttek igényeit szolgálja ki, akik a modern technológiákat kedvelik és biztonságos online ismerkedést keresnek. Az alkalmazás felhasználói közül sokan a hétköznapi életben nehezen találnak partnert, és a Fatal Love segítségével szeretnének új kapcsolatokat kialakítani.

Főbb funkciók:

- Felhasználói fiókok és profilkezelés

- Swipe alapú párosítási rendszer

- Valós idejű üzenetküldés

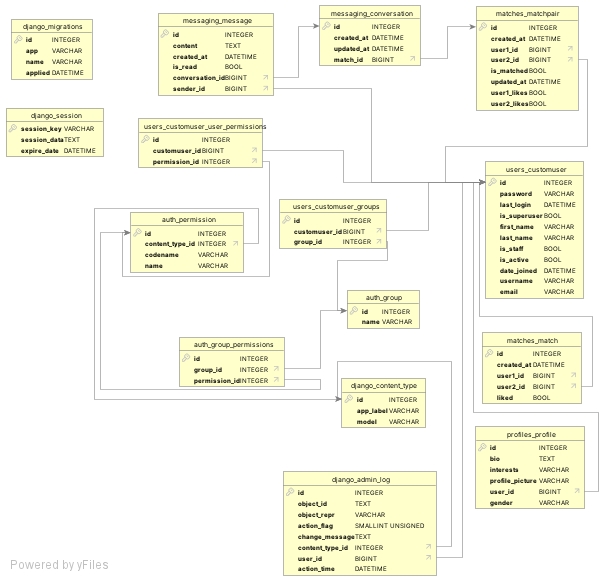
- Reszponzív dizájn

A Fatal Love alkalmazás számos modern funkciót kínál a felhasználóinak. A felhasználói fiókok és profilkezelés révén a felhasználók személyre szabott profilt hozhatnak létre, ahol megoszthatják érdeklődési körüket és személyes adataikat. A swipe alapú párosítási rendszer segítségével könnyen és gyorsan találhatnak partnert, míg a valós idejű üzenetküldés lehetővé teszi a folyamatos kommunikációt. A reszponzív dizájn révén az alkalmazás minden eszközön optimálisan működik, legyen szó mobiltelefonról, táblagépről vagy asztali számítógépről.

2. TECHNIKAI LEÍRÁS

2.1 Backend architektúra

 A backend rendszer a Django REST Framework-re épül, amely robusztus és skálázható alapot biztosít.

 Adatbázis struktúra:

- User tábla:

\* id (Primary Key)

\* username (Unique)

\* email (Unique)

\* password (Hashed)

\* created\_at

\* last\_login

- Profile tábla:

\* id (Primary Key)

\* user\_id (Foreign Key -> User)

\* profile\_picture

\* bio

\* interests

\* location

\* age

- Match tábla:

\* id (Primary Key)

\* user1\_id (Foreign Key -> User)

\* user2\_id (Foreign Key -> User)

\* created\_at

\* status

- Message tábla:

\* id (Primary Key)

\* match\_id (Foreign Key -> Match)

\* sender\_id (Foreign Key -> User)

\* content

\* created\_at

\* status

A backend architektúra a Django REST Framework-re épül, amely robusztus és skálázható alapot biztosít az alkalmazás számára. Az adatbázis struktúra négy fő táblából áll: a User tábla a felhasználói fiókokat tárolja, a Profile tábla a felhasználói profilokat, a Match tábla a párosításokat, és a Message tábla az üzeneteket. Minden tábla egyedi azonosítóval rendelkezik, és a kapcsolatok a külső kulcsok révén valósulnak meg.

API végpontok:

- /api/auth/

\* POST /register/ - Regisztráció

\* POST /login/ - Bejelentkezés

\* POST /logout/ - Kijelentkezés

\* POST /refresh/ - Token frissítés

- /api/profiles/

\* GET / - Profil lista

\* GET /<id>/ - Profil részletek

\* PUT /<id>/ - Profil frissítés

\* POST /<id>/picture/ - Profilkép feltöltés

- /api/matches/

\* GET / - Match lista

\* POST /<id>/like/ - Like küldése

\* POST /<id>/dislike/ - Dislike küldése

- /api/messages/

\* GET /<match\_id>/ - Üzenetek listája

\* POST /<match\_id>/ - Új üzenet

\* PUT /<id>/read/ - Üzenet olvasottként jelölése

Az API végpontok a felhasználói interakciók kezelésére szolgálnak. A /api/auth/ végpontok a felhasználói azonosítást és hitelesítést biztosítják, míg a /api/profiles/ végpontok a profilkezelést teszik lehetővé. A /api/matches/ végpontok a párosítási folyamatot kezelik, és a /api/messages/ végpontok az üzenetküldést és -fogadást teszik lehetővé.

Biztonsági megoldások:

- JWT autentikáció:

\* Access token: 5 perc érvényesség

\* Refresh token: 24 óra érvényesség

\* Token frissítés automatikus

- Adatbázis biztonság:

\* SQL injection védelem

\* Adatok titkosítása

\* Biztonságos jelszó tárolás

- API biztonság:

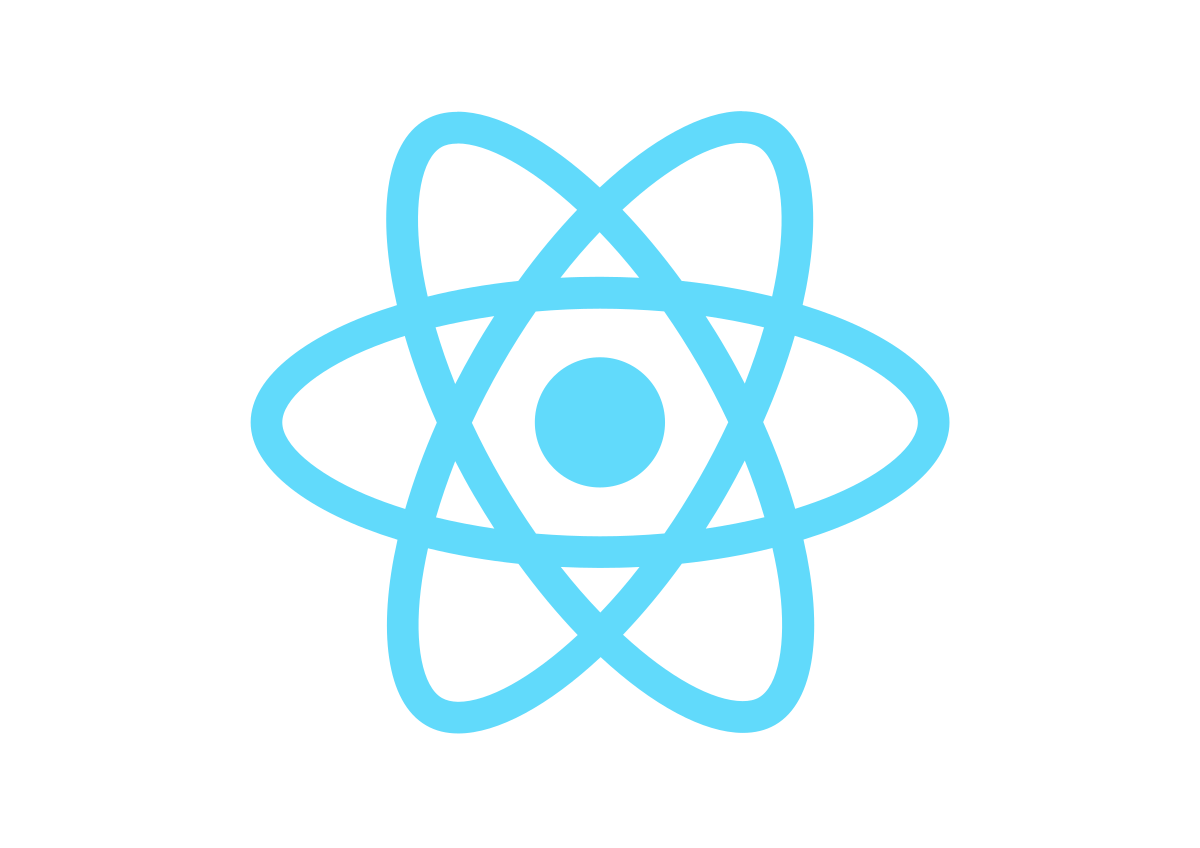
\* CORS védelem

\* Rate limiting

\* Input validáció

A biztonsági megoldások a felhasználói adatok védelmét és az alkalmazás biztonságát biztosítják. A JWT autentikáció révén a felhasználók biztonságosan bejelentkezhetnek, és a tokenek automatikusan frissülnek. Az adatbázis biztonság a SQL injection elleni védelmet, az adatok titkosítását és a biztonságos jelszó tárolást biztosítja. Az API biztonság a CORS védelem, a rate limiting és az input validáció révén biztosítja az API végpontok védelmét.

2.2 Frontend architektúra

 A frontend fejlesztés során a React keretrendszert használtuk, amely lehetővé tette a modern, komponens alapú felhasználói felület kialakítását.

Komponens struktúra:

[Komponens diagram]

- Auth komponensek:

\* LoginForm: Bejelentkezési űrlap

\* RegisterForm: Regisztrációs űrlap

\* PasswordReset: Jelszó visszaállítás

- Profile komponensek:

\* ProfileView: Profil megtekintés

\* ProfileEdit: Profil szerkesztés

\* ProfilePictureUpload: Profilkép feltöltés

- Matching komponensek:

\* SwipeCard: Swipe kártya

\* MatchList: Párosítások listája

\* MatchDetail: Párosítás részletei

- Chat komponensek:

\* ChatList: Beszélgetések listája

\* ChatWindow: Beszélgetés ablak

\* MessageInput: Üzenet bevitel

A komponens struktúra a felhasználói felület logikus szervezését biztosítja. Az Auth komponensek a felhasználói azonosítást és hitelesítést kezelik, míg a Profile komponensek a felhasználói profilok kezelését teszik lehetővé. A Matching komponensek a párosítási folyamatot kezelik, és a Chat komponensek az üzenetküldést és -fogadást teszik lehetővé.

State management:

- Redux store struktúra:

\* auth: Bejelentkezési állapot

\* profile: Felhasználói profil

\* matches: Párosítások

\* messages: Üzenetek

\* ui: Felhasználói felület állapot

A state management a Redux keretrendszerre épül, amely lehetővé teszi az alkalmazás állapotának hatékony kezelését. A Redux store öt fő részre oszlik: az auth a bejelentkezési állapotot, a profile a felhasználói profilt, a matches a párosításokat, a messages az üzeneteket, és az ui a felhasználói felület állapotát tárolja.

Routing:

- Főbb útvonalak:

\* / - Kezdőoldal

\* /login - Bejelentkezés

\* /register - Regisztráció

\* /profile - Profil

\* /matches - Párosítások

\* /chat - Üzenetek

A routing a React Router keretrendszerre épül, amely lehetővé teszi az alkalmazás navigációjának hatékony kezelését. A főbb útvonalak a kezdőoldalt, a bejelentkezést, a regisztrációt, a profilt, a párosításokat és az üzeneteket teszik elérhetővé.

Reszponzív dizájn:

- Tailwind CSS breakpoints:

\* sm: 640px

\* md: 768px

\* lg: 1024px

\* xl: 1280px

\* 2xl: 1536px

- Mobil optimalizáció:

\* Touch események

\* Swipe gesztusok

\* Mobilbarát inputok

A reszponzív dizájn a Tailwind CSS keretrendszerre épül, amely lehetővé teszi az alkalmazás reszponzív megjelenését. A Tailwind CSS breakpoints a különböző képernyőméretekhez igazodnak, míg a mobil optimalizáció a touch eseményeket, a swipe gesztusokat és a mobilbarát inputokat biztosítja.

3. TELEPÍTÉSI ÚTMUTATÓ

3.1 Szükséges programok

[Python logó] [Node.js logó] [Git logó]

- Python 3.12 vagy újabb

\* Microsoft Store-ból vagy python.org-ról

\* PATH környezeti változó beállítása

\* Verzió ellenőrzés: python --version

- Node.js LTS verzió

\* nodejs.org-ról letölthető

\* Automatikus PATH beállítás

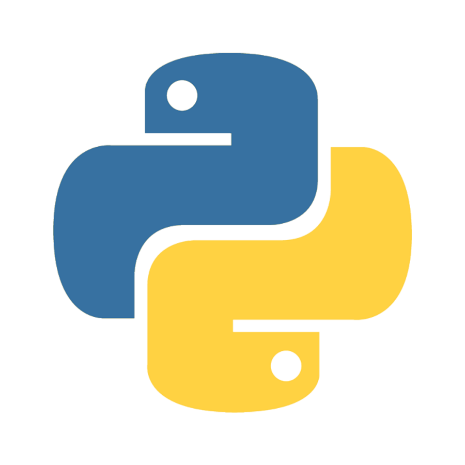
\* Verzió ellenőrzés: node --version

- Git

\* git-scm.com-ról letölthető

\* Verzió ellenőrzés: git --version

A szükséges programok a Fatal Love alkalmazás futtatásához szükségesek. A Python 3.12 vagy újabb verziója a backend fejlesztéshez szükséges, míg a Node.js LTS verziója a frontend fejlesztéshez. A Git a verziókezeléshez szükséges. Minden program telepítése után ellenőrizni kell a verziót, hogy megbizonyosodjunk a sikeres telepítésről.

 3.2 Telepítési lépések

1. Python telepítése:

a) Microsoft Store opció:

- Keress rá: "Python"

- Válaszd a 3.12 vagy újabb verziót

- Kattints a "Telepítés" gombra

b) Manuális telepítés:

 - Látogass el a python.org-ra

- Töltsd le a Python 3.12-t

- Futtasd a telepítőt

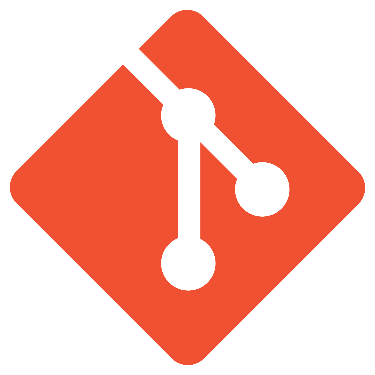
- Jelöld be: "Add Python to PATH"

- Válaszd a "Customize installation" opciót

- Jelöld be az összes opciót

2. Node.js telepítése:

- Látogass el a nodejs.org-ra

 - Töltsd le az LTS verziót

- Futtasd a telepítőt

- Fogadd el az alapértelmezett beállításokat

3. Git telepítése:

- Látogass el a git-scm.com-ra

- Töltsd le a Windows verziót

- Futtasd a telepítőt

- Fogadd el az alapértelmezett beállításokat

4. Projekt letöltése:

git clone https://github.com/[felhasználónév]/Tarskereso.git

cd Tarskereso

5. Backend beállítása:

cd backend

python -m venv venv

venv\Scripts\activate

pip install -r requirements.txt

python manage.py migrate

python manage.py createsuperuser

```

6. Frontend beállítása:

cd frontend

npm install

npm run build

```

7. Program indítása:

cd backend

python manage.py runserver

```

Az alkalmazás elérhető lesz: http://127.0.0.1:8000

A telepítési lépések során először a szükséges programokat kell telepíteni, majd a projektet letölteni és beállítani. A backend beállítása során létre kell hozni egy virtuális környezetet, telepíteni a függőségeket, és létre kell hozni egy superuser-t.

A frontend beállítása során telepíteni kell a függőségeket, és létre kell hozni a build fájlokat. Végül a program indításával az alkalmazás elérhetővé válik.

3.3 Hibaelhárítás

Gyakori problémák és megoldások:

1. "python nem található" hiba:

- Ellenőrizd a PATH környezeti változót

- Telepítsd újra a Python-t

- Indítsd újra a parancssort

2. "npm nem található" hiba:

- Ellenőrizd a Node.js telepítést

- Indítsd újra a parancssort

3. "Port already in use" hiba:

- Keress rá a futó Python folyamatokra

- Állítsd le a folyamatokat

- Indítsd újra a szervert

4. Adatbázis hibák:

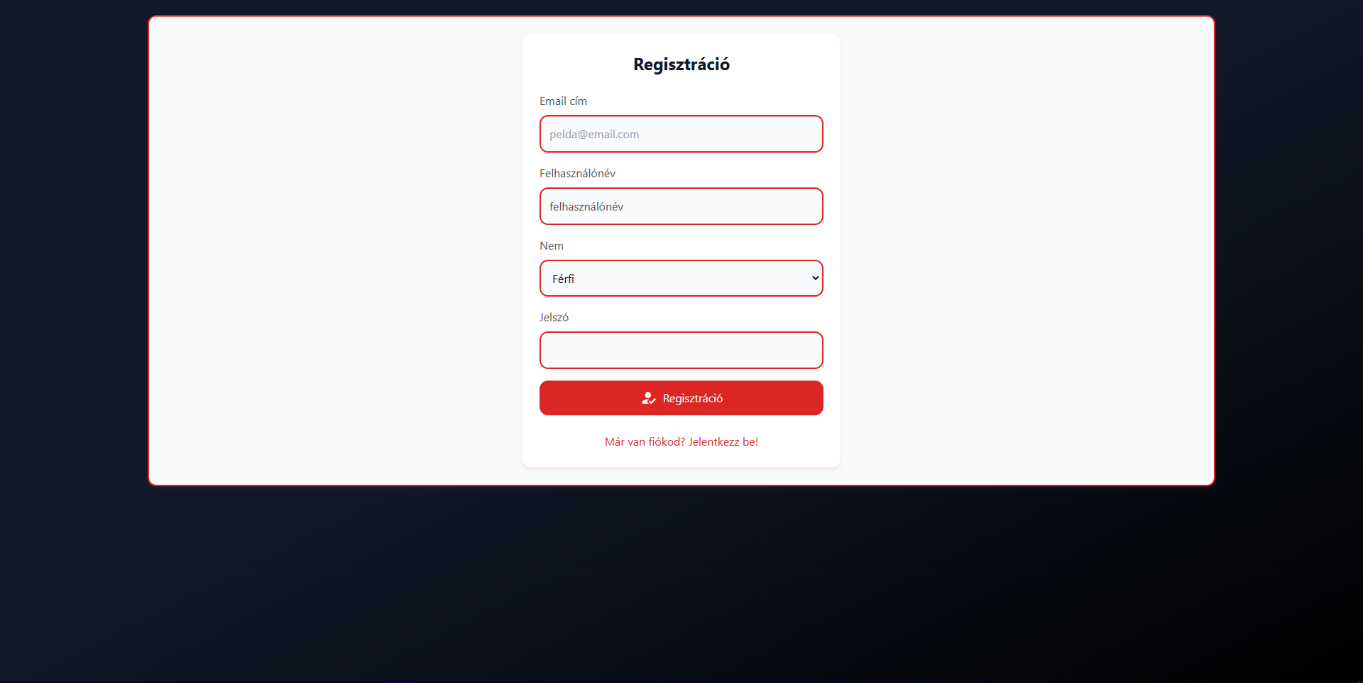
- Töröld a db.sqlite3 fájlt

- Futtasd újra a migrációkat

- Hozd létre újra a superuser-t

A hibaelhárítás során gyakran előforduló problémák és megoldásaik szerepelnek. A "python nem található" hiba esetén ellenőrizni kell a PATH környezeti változót, és szükség esetén újra kell telepíteni a Python-t. Az "npm nem található" hiba esetén ellenőrizni kell a Node.js telepítést. A "Port already in use" hiba esetén meg kell keresni a futó Python folyamatokat, és le kell állítani őket. Az adatbázis hibák esetén törölni kell a db.sqlite3 fájlt, újra kell futtatni a migrációkat, és újra kell létrehozni a superuser-t.

4. HASZNÁLATI ÚTMUTATÓ

4.1 Regisztráció

A regisztrációs folyamat lépései:

1. Nyisd meg a http://127.0.0.1:8000 oldalt

2. Kattints a "Regisztráció" gombra

3. Töltsd ki az űrlapot:

- Felhasználónév (egyedi)

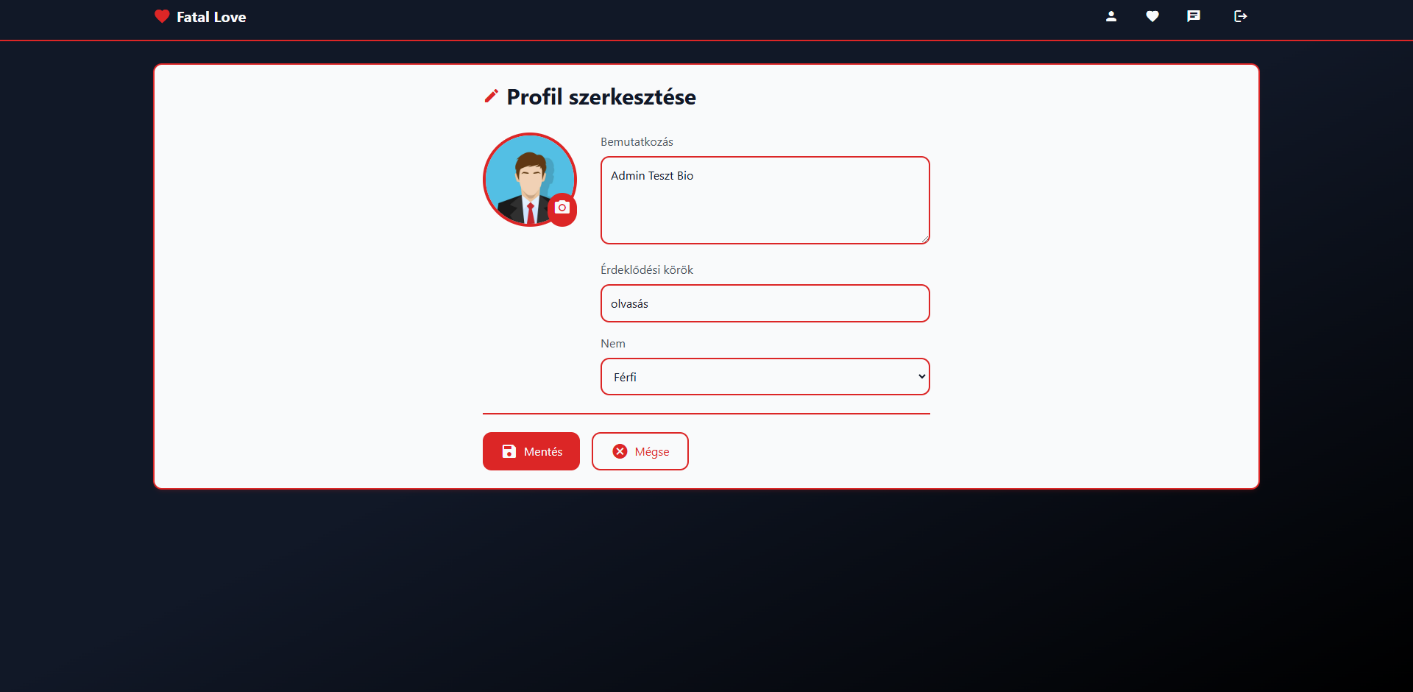
- E-mail cím (érvényes)

- Jelszó (minimum 8 karakter)

4. Kattints a "Regisztráció" gombra

A regisztrációs folyamat során a felhasználóknak először meg kell nyitniuk az alkalmazás kezdőoldalát, majd a "Regisztráció" gombra kattintva meg kell adniuk a szükséges adatokat.

4.2 Profil kezelés



Profil létrehozása és szerkesztése:

1. Kattints a "Profil" menüpontra

2. Töltsd ki a profil adatait:

- Név

- Életkor

- Helyszín

- Bemutatkozás

3. Tölts fel profilképet:

- Kattints a "Kép feltöltése" gombra

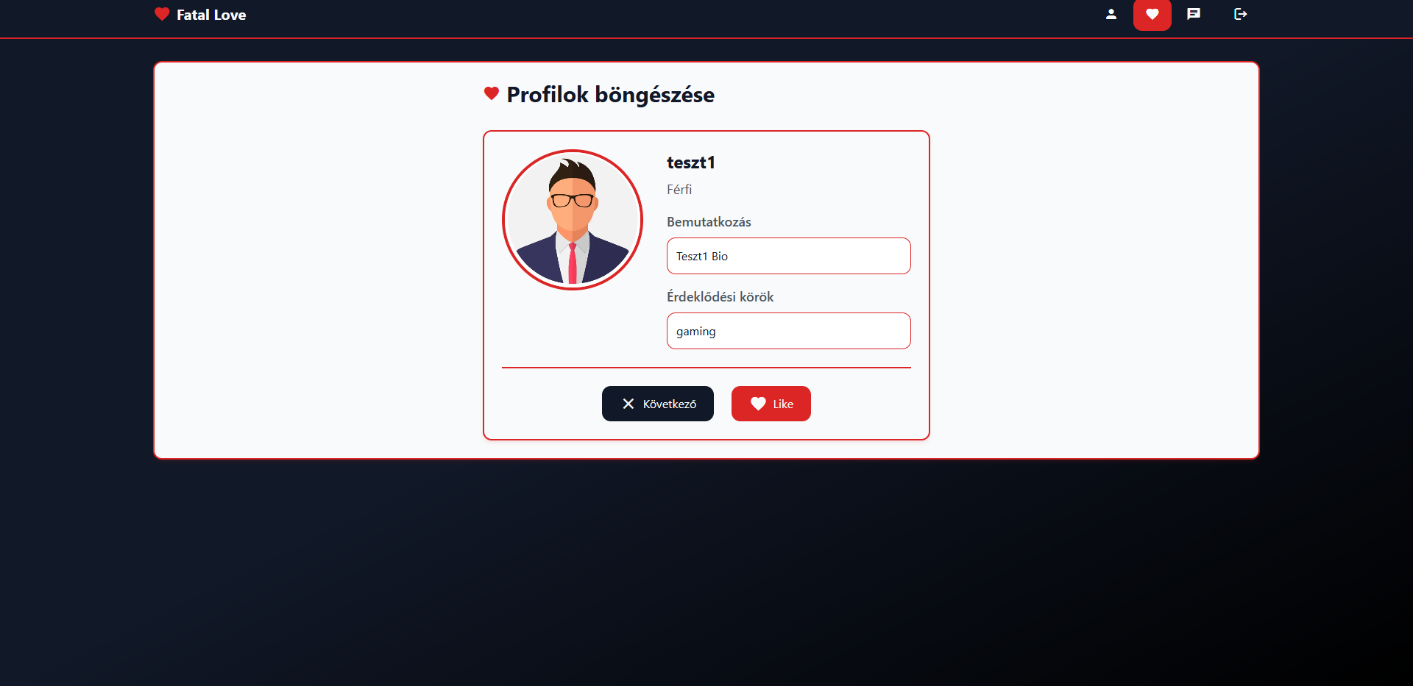
- Válaszd ki a képet

4. Válaszd ki az érdeklődési köröket

5. Kattints a "Mentés" gombra

A profil kezelés során a felhasználóknak a "Profil" menüpontra kattintva meg kell adniuk a szükséges adatokat, és feltölthetik a profilképet. A profil adatainak mentése után a felhasználók személyre szabott profilt hozhatnak létre, ahol megoszthatják érdeklődési körüket és személyes adataikat.

4.3 Párosítási rendszer



Párosítási folyamat:

1. Kattints a "Párosítás" menüpontra

2. Tekintsd meg a megjelenő profilt

3. Válassz a következő lehetőségek közül:

- Jobbra húzás = Like

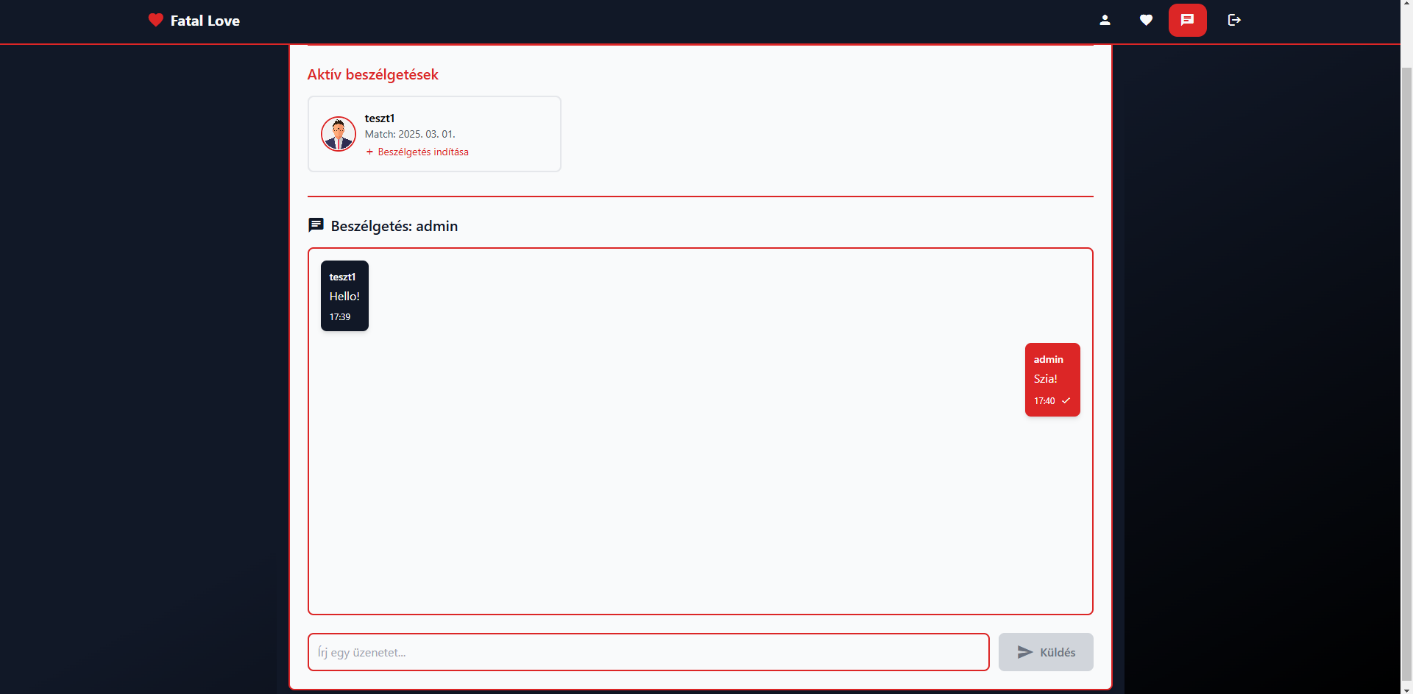
- Balra húzás = Dislike

4. Kölcsönös like esetén:

- Értesítés érkezik

- Megnyílik a chat funkció

A párosítási rendszer során a felhasználóknak a "Párosítás" menüpontra kattintva meg kell tekinteniük a megjelenő profilt, és a swipe gesztusok segítségével dönteniük kell a like vagy dislike mellett. Kölcsönös like esetén a felhasználók értesítést kapnak, és megnyílik a chat funkció.

4.4 Chat funkció

Üzenetküldés és chat kezelés:

1. Kattints a "Chat" menüpontra

2. Válaszd ki a beszélgetőpartnered

3. Írd be az üzenetet

4. Küldés gomb vagy Enter

5. Üzenet státuszok:

- Elküldve

- Kézbesítve

- Olvasva

A chat funkció során a felhasználóknak a "Chat" menüpontra kattintva meg kell választaniuk a beszélgetőpartnerüket, és be kell írniuk az üzenetet. Az üzenet küldése után a felhasználók követhetik az üzenet státuszát, amely lehet elküldve, kézbesítve vagy olvasva.

5. FEJLESZTÉSI FOLYAMAT

5.1 Csapatmunka

A fejlesztési folyamat során a következő módon működtünk:

Csapat felépítése:

A projektet három fejlesztőből álló csapatunkkal valósítottuk meg.A fejlesztés koordinálására rendszeresen kommunikáltunk Meet-en és Messenger-en keresztül, megbeszéltük a következő lépéseket és a jelenlegi kihívásokat.

Kód megosztás:

A kód megosztásra a GitHub repository-t használtuk. Minden új változást könnyem tudtunk figyelni.

6. TECHNOLÓGIAI STACK

6.1 Backend

A backend fejlesztés során a Django 5.1.6 keretrendszert használtuk, amely robusztus alapot biztosított az alkalmazásunk számára. A Django REST Framework 3.14.0 segítségével valósítottuk meg a RESTful API-t, míg a JWT autentikáció biztosította a biztonságos felhasználói azonosítást. A valós idejű kommunikációhoz WebSocket technológiát alkalmaztunk.

6.2 Frontend

A frontend fejlesztés során a React 19.0.0 keretrendszert használtuk, amely lehetővé tette a modern, komponens alapú felhasználói felület kialakítását. A Tailwind CSS segítségével valósítottuk meg a reszponzív dizájnt, míg a React Router biztosította az intuitív navigációt. Az API hívásokhoz az Axios könyvtárat használtuk.

7. TESZTELÉSI DOKUMENTÁCIÓ

7.1 Tesztelési terv

A tesztelés során a következő területeket vizsgáltuk:

Regisztráció és bejelentkezés:

A regisztrációs és bejelentkezési folyamat során ellenőriztük és a műveletek sikeresen lefutnak.

Profilkezelés:

A profilkezelés során teszteltük az adatok mentésének megbízhatóságát, a kép feltöltés funkcióját, a profil szerkesztés lehetőségét és az adatok megjelenítésének helyességét.

Párosítási rendszer:

A párosítási rendszer tesztelése során ellenőriztük a swipe funkció működését, a like/dislike rendszer megbízhatóságát, a match létrehozás folyamatát és a chat funkció megnyitásának helyességét.

Chat funkció:

A chat funkció tesztelése során vizsgáltuk az üzenetküldés megbízhatóságát, az üzenetek megjelenítésének helyességét, a státusz frissítés folyamatát és a chat előzmények kezelését.

8. JÖVÖBELI FEJLESZTÉSI LEHETŐSÉGEK

8.1 Tervezett funkciók

A jövőben tervezzük a következő funkciók bevezetését:

- Helyzet alapú párosítás:

\* Felhasználók közelében lévő partnerek megjelenítése

\* Távolság alapú szűrés

\* Helyzet megosztás beállítások

- Videó chat funkció:

\* Valós idejű videó beszélgetések

- Bővített profilbeállítások:

\* Részletesebb érdeklődési körök

\* Több nyelv támogatás

\* Profil témák és dizájnok

A tervezett funkciók a felhasználói élmény további fejlesztését célozzák. A helyzet alapú párosítás révén a felhasználók könnyen találhatnak partnert a közelükben, míg a videó chat funkció lehetővé teszi a valós idejű videó beszélgetéseket. A bővített profilbeállítások révén a felhasználók személyre szabottabb profilt hozhatnak létre, és több nyelven is használhatják az alkalmazást.

8.2 Technikai fejlesztések

A technikai oldalon a következő fejlesztéseket tervezzük:

- Teljesítmény optimalizálás:

\* Adatbázis lekérdezések gyorsítása

\* Kép optimalizálás

- Felhasználói visszajelzések alapján módosítások:

\* Hibajavítások

\* UI/UX fejlesztések

\* Új funkciók bevezetése

A technikai fejlesztések az alkalmazás teljesítményének növelését célozzák. A teljesítmény optimalizálás révén az alkalmazás gyorsabbá válik. A felhasználói visszajelzések alapján történő módosítások révén az alkalmazás folyamatosan fejlődik és alkalmazkodik a felhasználói igényekhez.

TARTALOMJEGYZÉK

1. Bevezetés

2. Technikai leírás

3. Telepítési útmutató

4. Használati útmutató

5. Fejlesztési folyamat

6. Technológiai stack

7. Tesztelési dokumentáció

8. Jövőbeli fejlesztések

Készítették:

- Bodnár Nikolett

- Vasas Ágoston

- Dér Sándor