**SocialLobster**

**Közösségi Média Projekt**

(Patai Zsolt, Fehér Erik, Szerencsés Attila)

Összefoglaló:

A SocialLobster egy közösségi média weboldal lesz amely az online jelen lévő emberek összeköttetésére szolgál. Egy egyszerű, mégis designos felületen próbáljuk kielégíteni a felhasználók összes igényét. A weboldal minden információt közöl a felhasználóval, ezzel is segítve az egyszerű használatot. Projektünkön 3 db fejlesztő fog dolgozni: Szerencsés Attila, Patai Zsolt, Fehér Erik. 3 Mérföldkőre osztott munka során fogjuk prezentálni a weboldal folyamatos fejlődését.

Rendszerspecifikáció:

Lehetőség lesz egy vendégnek regisztrálnia. Innentől kezdve felhasználóként fog szerepelni a SocialLobster-en. Egy email-es visszaigazolás után már be is tud jelentkezni és használhatja a felületet. A felhasználó beállíthatja saját profilját, megadhatja iskoláját, munkahelyét, lakhelyét, illetve tölthet fel profilképet is. A felhasználónak lehetősége van ismerősöket jelölni, ismerőseinek üzeneteket küldeni. Továbbá képes csoportokat létrehozni, csoportba jelentkezni. Csoport adminisztrátorként lehetőség van tagokat eltávolítani a csoportból is. Továbbá lehetőség van posztokat írni, más posztokat kedvelni, illetve hozzá szólásokat írni egy poszthoz.

Funkcionális követelmények:

Felhasználók kezelése:

* Vendég: Vendégként lehetőség lesz regisztrálni a felületen, így felhasználóvá vállni.
* Felhasználó: A felhasználónak lehetősége lesz, barátokat jelölni, posztokat írni, üzeneteket írni, illetve csoportba szerveződni más emberekkel.

Felhasználói profil feltöltése:

* A felhasználónak lehetősége lesz feltölteni adatait, azokat módosítani. Lehetőség lesz lakhelyet, iskolát, munkahelyet módosítani.
* Lehetőség lesz profilképet feltölteni

Képek feltöltése:

* A felhasználóknak lesz lehetősége képeket feltölteni szinte bármiről, amelyeket egy albumba rendezve fogunk megjeleníteni.

Ismerősnek jelölés:

* A felhasználó ismerősnek tud jelölni más felhasználókat, amit ha elfogadnak, onnantól kezdve ismerősök lesznek. Az ismerősök tudnak egymásnak üzenetet küldeni.

Posztok írása:

* Lehetőség lesz posztok írására is. Felhasználok leírhatják véleményüket, vagy éppen a napjukat, vagy bármit amit szeretnének megosztani. Ezeket a posztokat lehetséges likeolni is.

Kommentek írása:

* A felhasználókat a létrehozott posztokhoz tudnak hozzászólásokat is hozzáfűzni. Ezek is lehet majd likeolni, ezzel kifejezve a tetszésünket iránta.

Üzenetek küldése:

* Lehetőség lesz az ismerősöknek üzenetet küldeni egymásnak. Az üzenetet csak a két fél látja amit elküldtek egymásnak.

Csoportok létrehozása, kezelése:

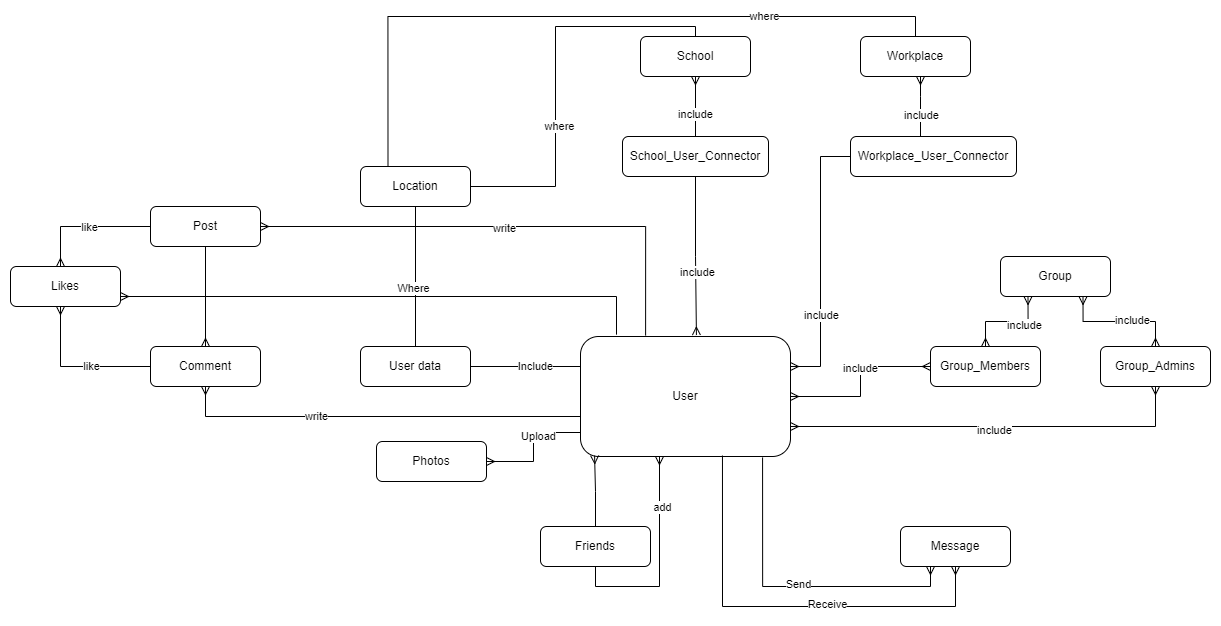
* Lehetőség lesz csoportokat létrehozni, amelybe meghívhatunk több embert is. Ez lehet egy rajongó csoport, vagy akár azonos érdeklődésű emberek csoportja. Minden csoportnak lesz adminisztrátora is, aki képes tagokat törölni a csoportból, ehhez másnak nem lesz joga.

Nem funkcionális követelmények:

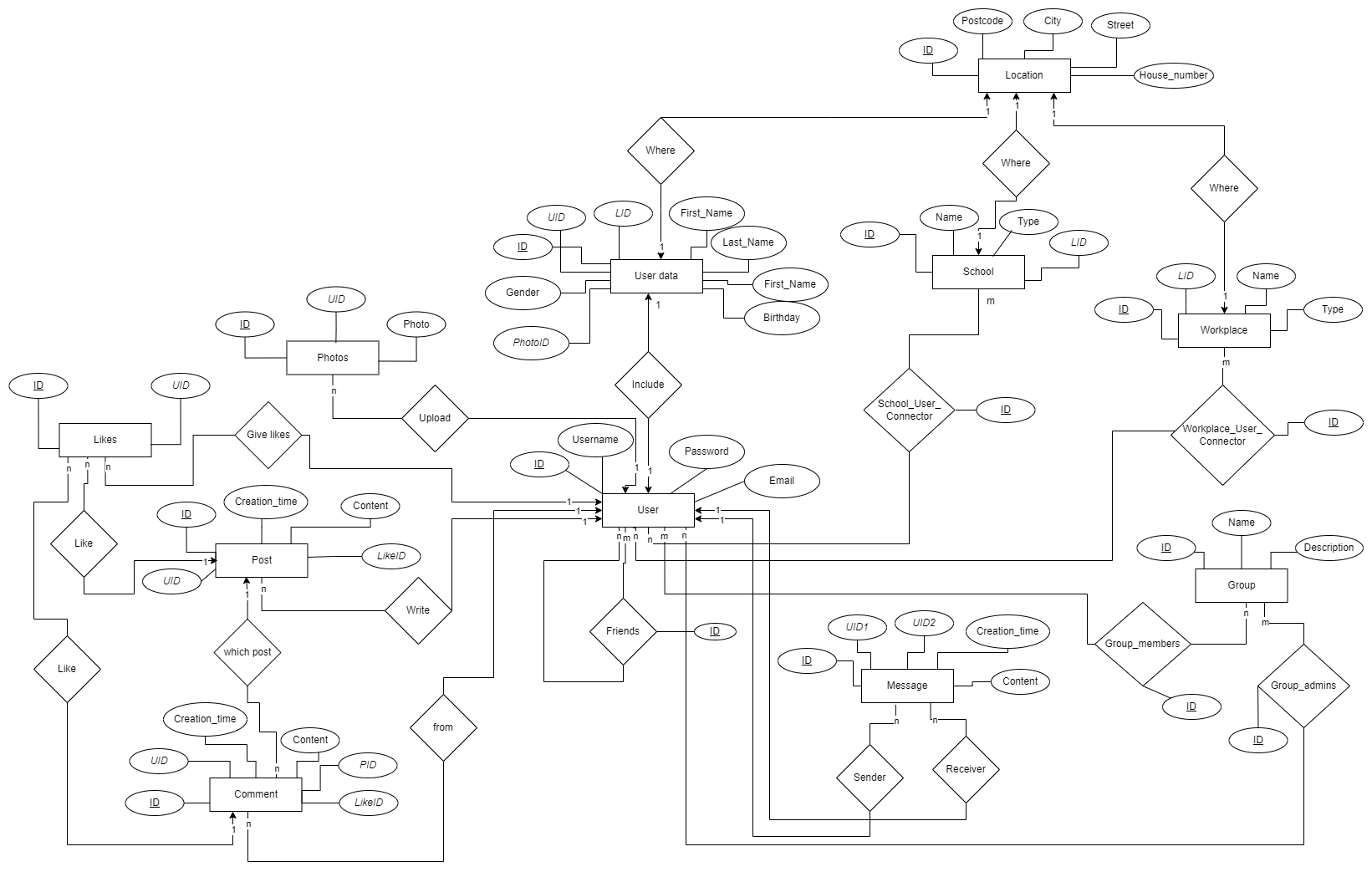
* A kliens oldal platform- és böngészőfüggetlen legyen.
* Reszponzív megjelenés
* Szenzitív adatok biztonságos tárolása
* A legfrissebb technológiák használata a rendszer során.

**Diagramok:**

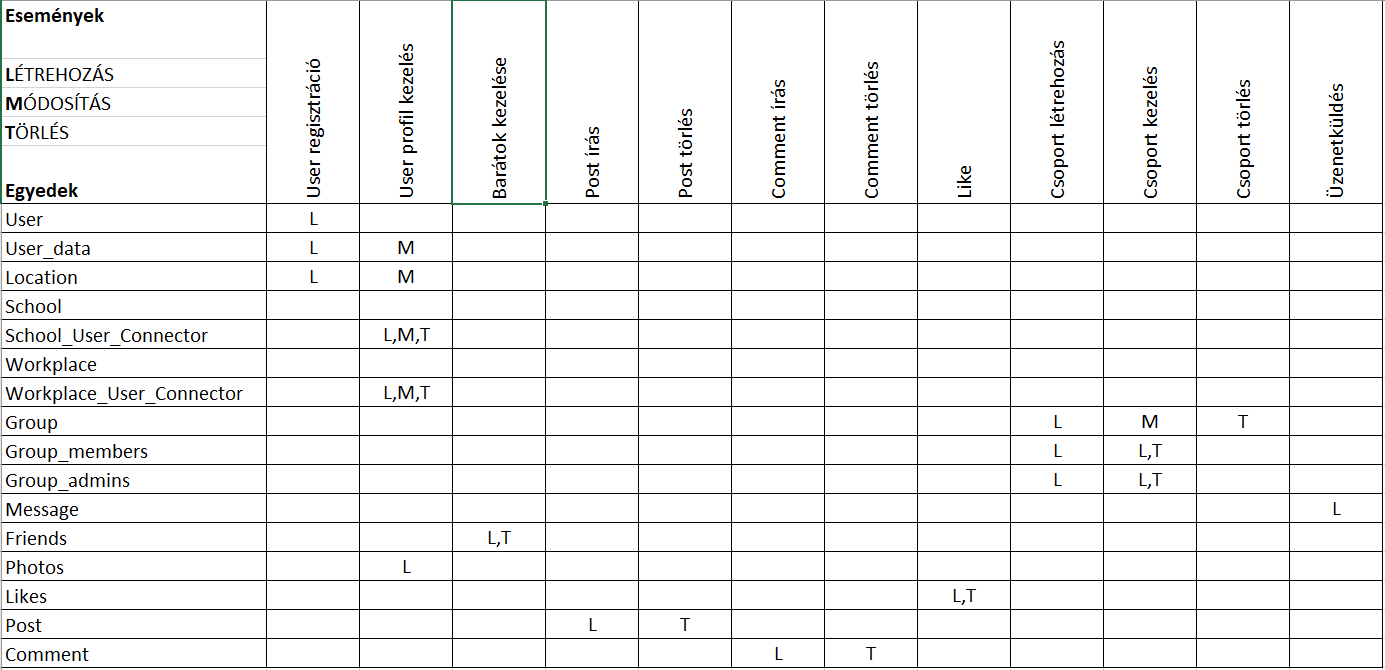
**Egyedmodell:**



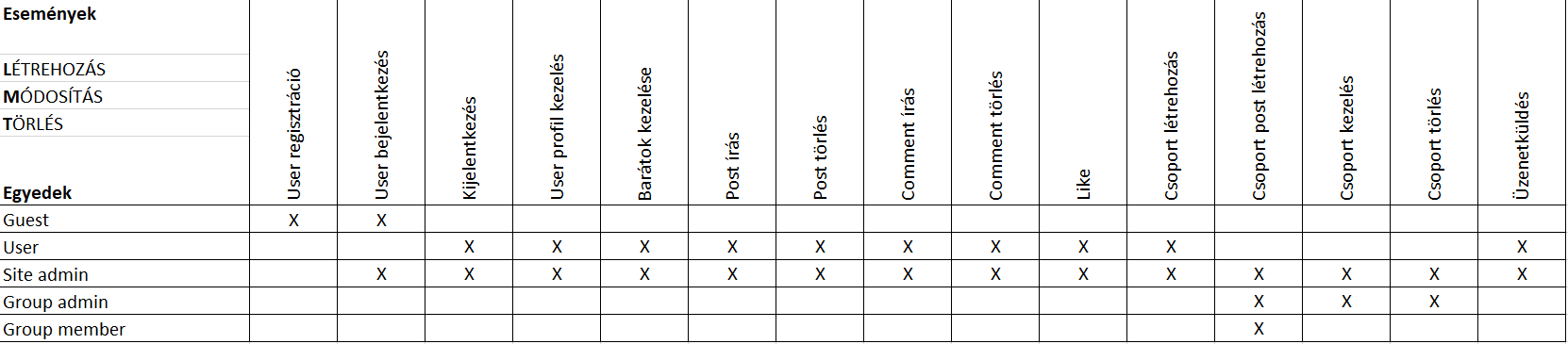
**EK Diagram:**



**Egyed Esemény mátrix:**



**Szerep Funkció mátrix:**



**Funkció meghatározás:**

-User regisztráció és bejelentkezés

-Mások követése

-Csoportok készítése, kezelése, törlése

-Post írása, kezelése, törlése

-Komment írása, kezelése, törlése

-Postok, kommentek likeolása

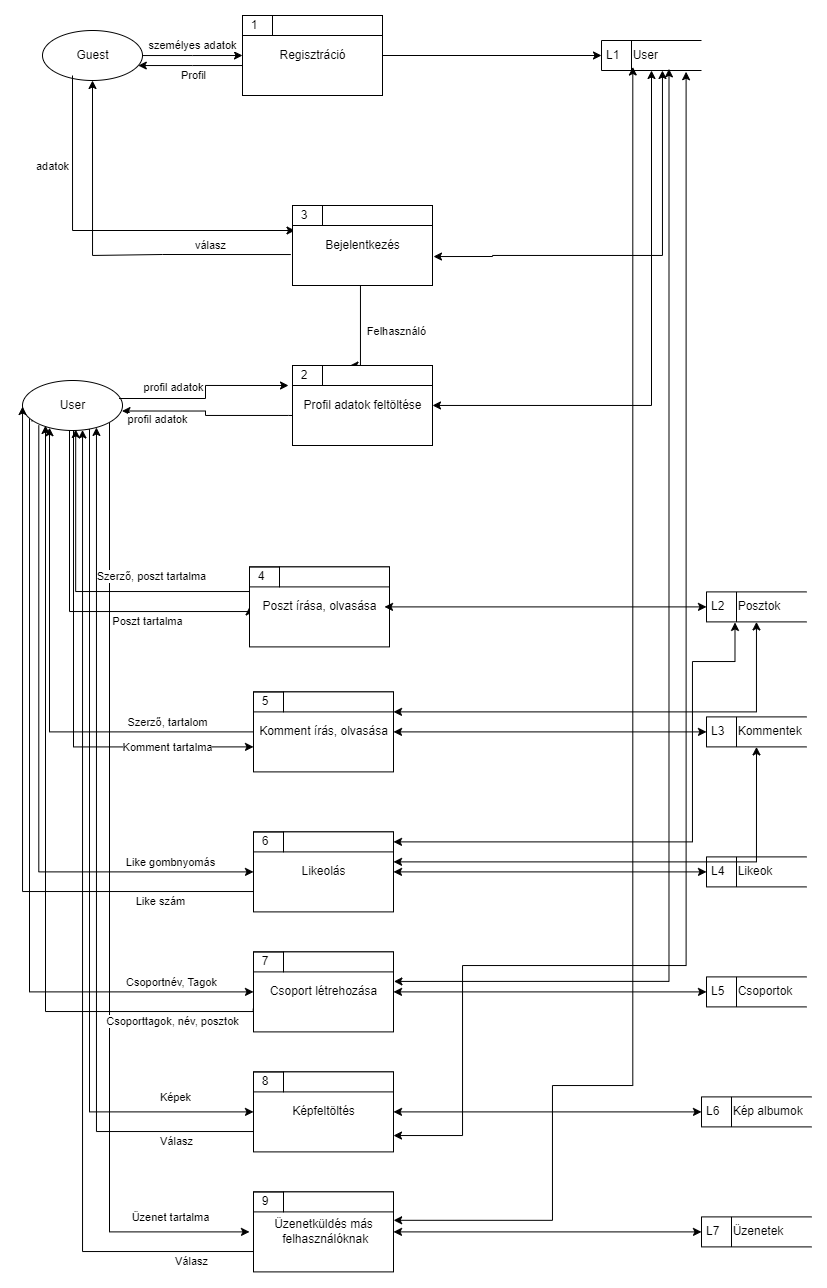
-Üzenetek küldése

-Emberek, csoportok keresése

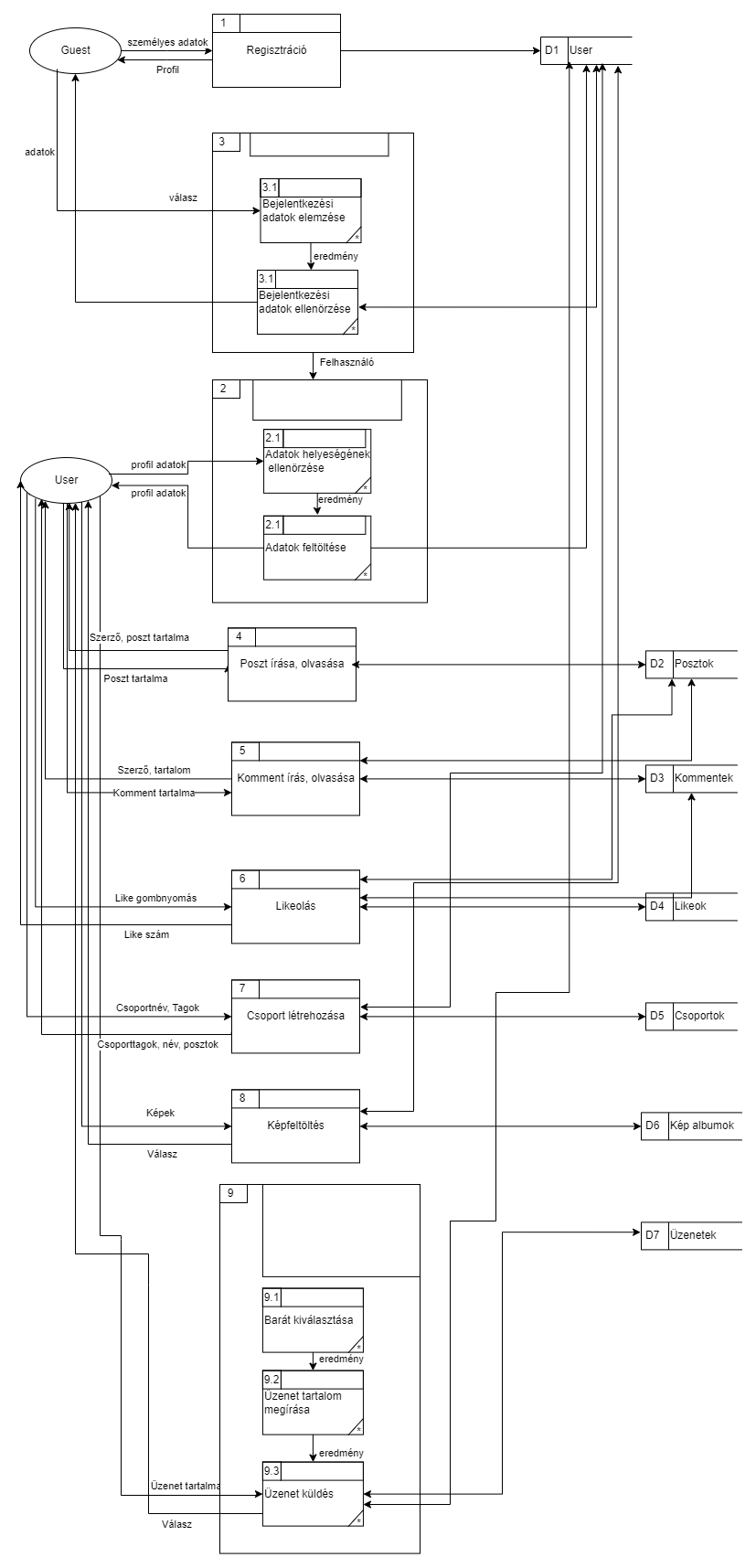
-Feed olvasása

-Profilok kezelése, megtekintése

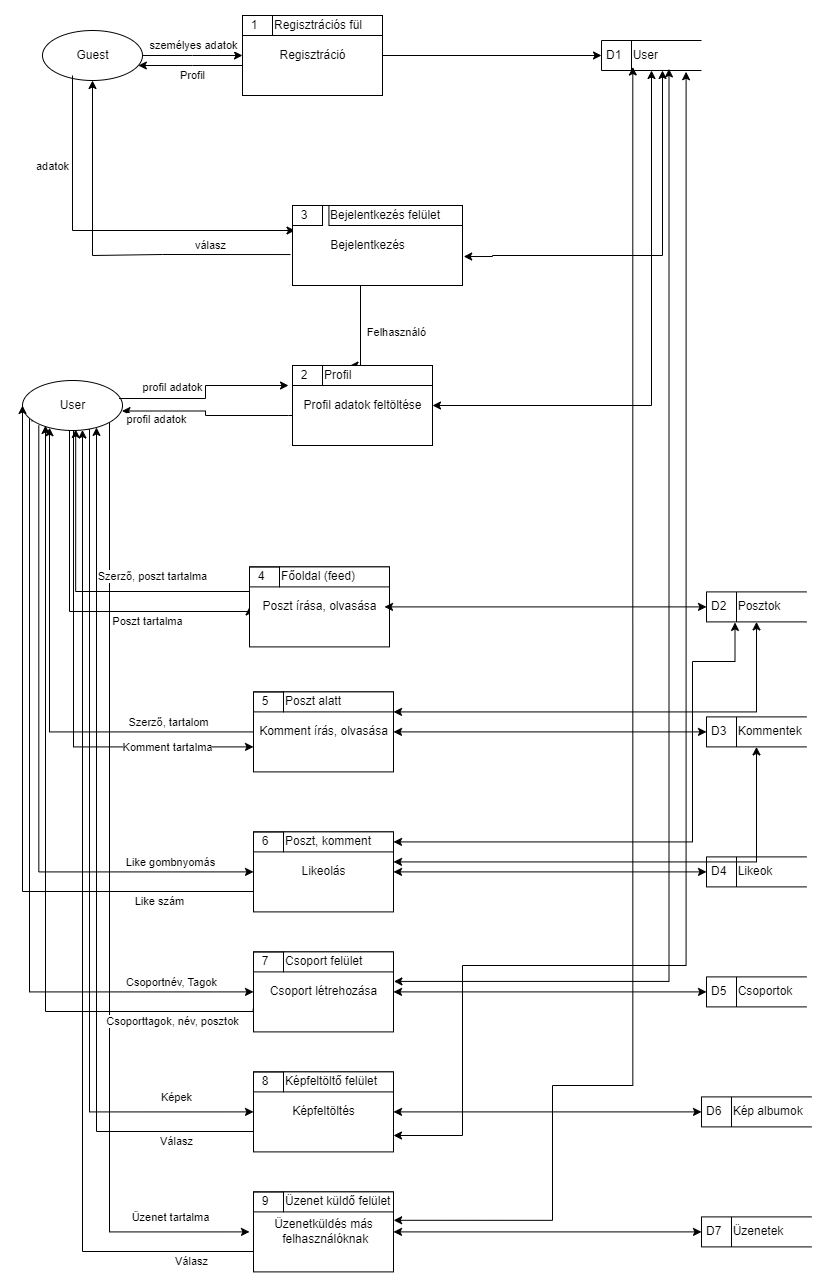
**Logikai AFD: 1. szintű**



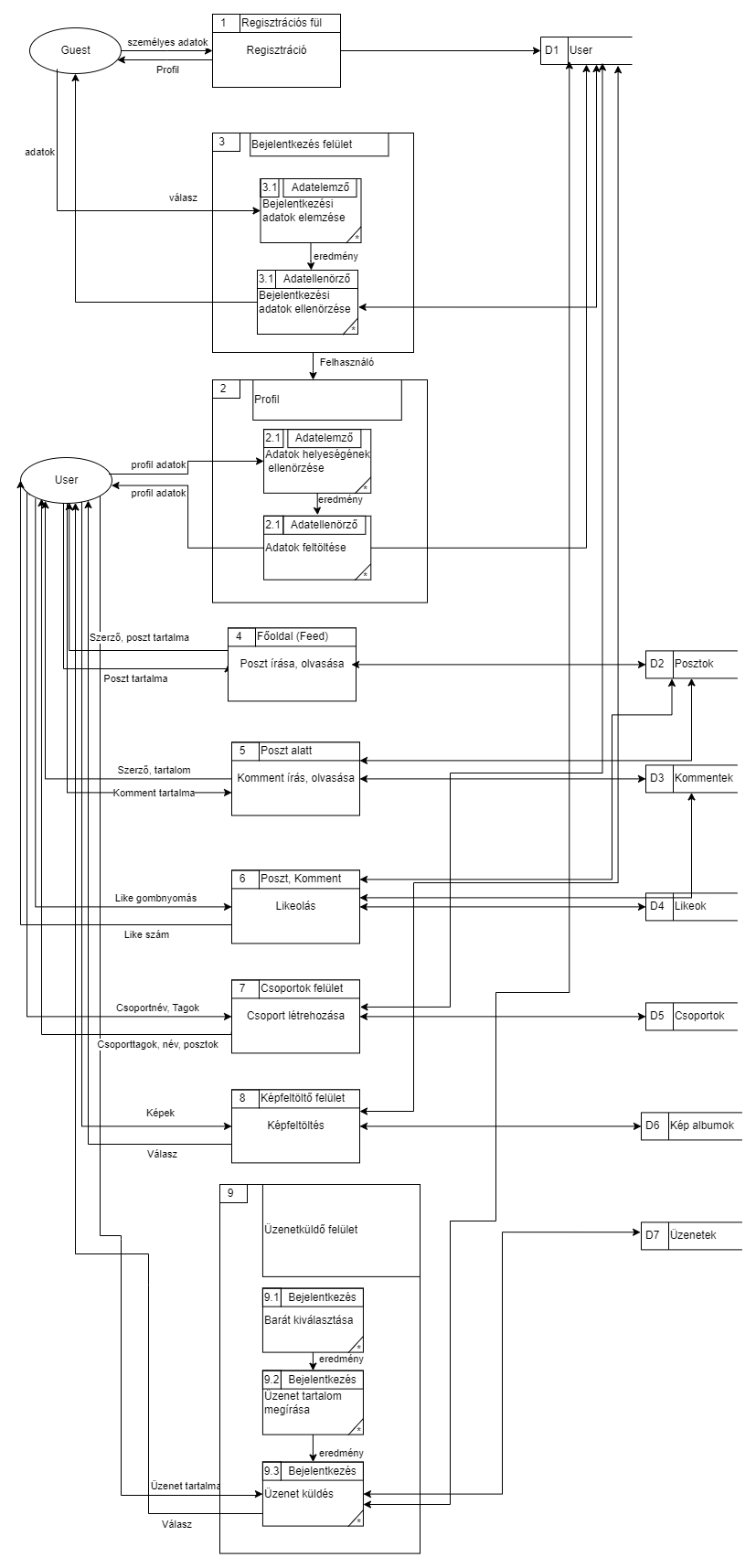
**Logikai AFD: 2. szintű**



**Fizikai AFD: 1. szintű**



**Fizikai AFD: 2. szintű**



**EK Diagram lekepézés:**

(Az aláhúzás jelöli a kulcsot, a dőlt betű pedig az idegen kulcsot.)

User(ID, Username, Password, Email)

User data(ID,*UID*, *LID*, *PhotoID*, First\_name,Last\_name, Gender, Birthday)

School(ID, Name, Type, *LID*)

School\_User\_Connector(ID,*UID*,*SID*)

Workplace(ID,*LID*, Name, Type,)

Workplace\_User\_Connector(ID, *UID*,*WID*)

Location(ID, Postcode, City, Street, Housenumber)

Post(ID, *UID*, Creation\_time, Content, *LikeID*)

Likes(ID,UID)

Comment(ID, *UID*, *PID* Creation\_time, Content, *LikeID*)

Group(ID, Name, Description)

Group\_members(ID, *GroupId*, *UID*)

Group\_admins(ID, *GroupID*, *UID*)

Message(ID, *UID1*, *UID2*, Content, Creation\_time)

Photos(ID, *UID*, Photo)

Friends(ID, *UID1*, *UID2*)

**Normalizálás:**

Mivel mindegyik megfelel a 3. normálformának, ezért nincs szükség változtatásra.

Az adatbázis elkészítésekor figyelembe vettük a normálformákat, ezért eszerint alakítottuk ki a legelejétől kedve. Emiatt az EK diagramm lekepézés teljesíti a 3NF-et.

1. normálforma: Minden tulajdonság csak elemi adatokat tartalmaz. A felhasználó neve felbontásra került vezetéknévre és keresztnévre, illetve a helyszín is szétbontásra került, irányítószám, város, utca, házszám részekre.

2. normálforma: Első normálfának megfelel. Minden táblához felvettünk egy külön id-t ami alapján minden adata beazonosítható, ezek lettek az elsődleges kulcsok. Ezáltal a reláció minden tulajdonsága csak az elsődleges kulcstól függ.

3. normálforma: 2. normálformának megfelel. A táblákat pedig úgy vettük fel, hogy ne alakuljon ki tranzitív függőség. Egyik tábla sem tartalmaz olyan adatot, amelynek nincs köze az adott táblához.

**Adattáblák leírása:**

CREATE TABLE user(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

username varchar2(50) NOT NULL,

password varchar2(50) NOT NULL,

email varchar2(50) NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE location(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

postcode varchar2(10),

city varchar2(30),

street varchar2(30),

housenumber NUMBER,

PRIMARY KEY (id)

);

CREATE TABLE userdata(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

uid NUMBER(10) NOT NULL,

lid NUMBER(10) NOT NULL,

photoid NUMBER(10),

first\_name varchar2(30),

last\_name varchar2(30),

gender varchar2(1),

birthday DATE,

CONSTRAINT loc

FOREIGN KEY (lid)

REFERENCES location(id)

ON DELETE CASCADE,

CONSTRAINT usr

FOREIGN KEY (uid)

REFERENCES user(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

CREATE TABLE school(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

lid NUMBER(10) NOT NULL,

name varchar2(30),

school\_type varchar2(30),

CONSTRAINT loc

FOREIGN KEY (lid)

REFERENCES location(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

CREATE TABLE school\_user\_connector(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

uid NUMBER(10) NOT NULL,

sid NUMBER(10) NOT NULL,

CONSTRAINT sch

FOREIGN KEY (sid)

REFERENCES school(id)

ON DELETE CASCADE,

CONSTRAINT usr

FOREIGN KEY (uid)

REFERENCES user(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

CREATE TABLE workplace(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

lid NUMBER(10) NOT NULL,

name varchar2(30),

workplace\_type varchar2(30),

CONSTRAINT loc

FOREIGN KEY (lid)

REFERENCES location(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

CREATE TABLE workplace\_user\_connector(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

uid NUMBER(10) NOT NULL,

wid NUMBER(10) NOT NULL,

CONSTRAINT wrk

FOREIGN KEY (wid)

REFERENCES workplace(id)

ON DELETE CASCADE,

CONSTRAINT usr

FOREIGN KEY (uid)

REFERENCES user(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

CREATE TABLE post(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

uid NUMBER(10) NOT NULL,

creation\_time DATETIME NOT NULL,

content varchar2(999) NOT NULL,

CONSTRAINT usr

FOREIGN KEY (uid)

REFERENCES user(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

CREATE TABLE post\_likes(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

uid NUMBER(10) NOT NULL,

pid NUMBER(10) NOT NULL,

CONSTRAINT usr

FOREIGN KEY (uid)

REFERENCES user(id)

ON DELETE CASCADE,

CONSTRAINT pst

FOREIGN KEY (pid)

REFERENCES post(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

CREATE TABLE comment(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

uid NUMBER(10) NOT NULL,

creation\_time DATETIME NOT NULL,

content varchar2(999) NOT NULL,

CONSTRAINT usr

FOREIGN KEY (uid)

REFERENCES user(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

CREATE TABLE comment\_likes(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

uid NUMBER(10) NOT NULL,

cid NUMBER(10) NOT NULL,

CONSTRAINT usr

FOREIGN KEY (uid)

REFERENCES user(id)

ON DELETE CASCADE,

CONSTRAINT cmt

FOREIGN KEY (cid)

REFERENCES comment(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

CREATE TABLE group(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

name varchar2(50) NOT NULL,

description varchar2(999),

PRIMARY KEY (id)

);

CREATE TABLE group\_members(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

uid NUMBER(10) NOT NULL,

gid NUMBER(10) NOT NULL,

CONSTRAINT usr

FOREIGN KEY (uid)

REFERENCES user(id)

ON DELETE CASCADE,

CONSTRAINT grp

FOREIGN KEY (gid)

REFERENCES group(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

CREATE TABLE group\_admins(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

uid NUMBER(10) NOT NULL,

gid NUMBER(10) NOT NULL,

CONSTRAINT usr

FOREIGN KEY (uid)

REFERENCES user(id)

ON DELETE CASCADE,

CONSTRAINT grp

FOREIGN KEY (gid)

REFERENCES group(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

PRIMARY KEY (id)

);

CREATE TABLE message(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

uid1 NUMBER(10) NOT NULL,

uid2 NUMBER(10) NOT NULL,

content varchar2(999) NOT NULL,

creation\_time DATETIME NOT NULL,

CONSTRAINT usr1

FOREIGN KEY (uid1)

REFERENCES user(id)

ON DELETE CASCADE,

CONSTRAINT usr2

FOREIGN KEY (uid2)

REFERENCES user(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

CREATE TABLE photos(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

uid NUMBER(10) NOT NULL,

photo VARCHAR2(50) NOT NULL,

CONSTRAINT usr

FOREIGN KEY (uid)

REFERENCES user(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

CREATE TABLE message(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

uid1 NUMBER(10) NOT NULL,

uid2 NUMBER(10) NOT NULL,

CONSTRAINT usr1

FOREIGN KEY (uid1)

REFERENCES user(id)

ON DELETE CASCADE,

CONSTRAINT usr2

FOREIGN KEY (uid2)

REFERENCES user(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

CREATE TABLE follows(

id NUMBER(10) NOT NULL AUTO\_INCREMENT,

uid1 NUMBER(10) NOT NULL,

uid2 NUMBER(10) NOT NULL,

CONSTRAINT usr1

FOREIGN KEY (uid1)

REFERENCES user(id)

ON DELETE CASCADE,

CONSTRAINT usr2

FOREIGN KEY (uid2)

REFERENCES user(id)

ON DELETE CASCADE,

PRIMARY KEY (id)

);

**Feladatkiosztás:**

