** **

**ZÁRÓDOLGOZAT**

Készítették:

*Pap Dániel*

Miskolc

2022.

**Miskolci SZC Kandó Kálmán Informatikai Technikum**

Az 5-0613-12-03 számú Szoftverfejlesztő és -tesztelő technikus szakképesítés záródolgozat

**IdeaShare**

Készítették:

*Pap Dániel*

Miskolc

2022.

Tartalom

[1. - Témaválasztás 5](#_Toc101719704)

[2. - Felhasznált technológiák és programok 5](#_Toc101719705)

[2.1 - Node.js - Dani 5](#_Toc101719706)

[2.2 - React 6](#_Toc101719707)

[2.3 - XAMPP - Dani 6](#_Toc101719708)

[2.4 - phpMyAdmin - Dani 6](#_Toc101719709)

[2.9 - Visual Studio Code 2019 - Dani 6](#_Toc101719710)

[3. - Programozási nyelvek 7](#_Toc101719711)

[3.1 - JavaScript -Dani 7](#_Toc101719712)

[3.2 - SQL/MySQL - Dani 7](#_Toc101719713)

[3.3 - Bootstrap/React-Bootstrap 7](#_Toc101719714)

[4. - Adatbázis 7](#_Toc101719715)

[4.1- Az adatbázis bemutatása 7](#_Toc101719716)

[4.2 - Relációs Sémák 9](#_Toc101719717)

[5.0 - A vizsgaremek 11](#_Toc101719718)

[5.1 - A backend bemutatása 12](#_Toc101719719)

[5.1.1 - Általános leírása 12](#_Toc101719720)

[5.1.2 - Adatbázis használata 12](#_Toc101719721)

[5.1.3. - Regisztráció 12](#_Toc101719722)

[5.1.4 - Bejelentkezés 13](#_Toc101719723)

[5.1.5 - Kijelentkezés 13](#_Toc101719724)

[5.1.6 - Hitelesítés 13](#_Toc101719725)

[5.1.7 - Article módosítás 14](#_Toc101719726)

[5.1.8 - Komment létrehozás 14](#_Toc101719727)

[6. - A vizsgaremek - A frontend bemutatása 14](#_Toc101719728)

[6.1 - A webes frontend bemutatása 15](#_Toc101719729)

[6.1.1 - A webes frontend bemutatása - Kezdőlap 15](#_Toc101719730)

[6.1.2 - A webes frontend bemutatása - Regisztrációs felület 16](#_Toc101719731)

[6.1.3 - A webes frontend bemutatása - Bejelentkezési felület 17](#_Toc101719732)

[6.1.4 - A webes frontend bemutatása - Articles felület 18](#_Toc101719733)

[6.1.5 - A webes frontend bemutatása - Create Article 18](#_Toc101719734)

[6.1.6 - A webes frontend bemutatása - Edit Article 19](#_Toc101719735)

[6.1.7 - A webes frontend bemutatása - Profile Page 20](#_Toc101719736)

[6.1.8 - A webes frontend bemutatása - Menüsor és kijelentkezés 21](#_Toc101719737)

[7.0 - Továbbfejlesztési lehetőségek 22](#_Toc101719738)

[8. - Források, hivatkozások 23](#_Toc101719739)

# 1. - Témaválasztás

Egy webes és egy mobilos alkalmazást készítettünk. A projekt témáját közösen találtuk ki. Egy, már meglévő, széles körben elterjedt üzleti logikára épül az ötletünk. A weboldalon lehetőséget kínálunk ”ötletelőknek” megosztani az ötleteiket, kezdetleges projektjeiket és egyéb programozást és különféle informatikai tudást igénylő ötletüket, projektjüket. Továbbá egyéb kéréssel, kérdéssel is fordulhatnak az ötletelők az oldal használói felé. Programozók, informatikai tudással és informatikával foglalkozó személyek jelentkezhetnek ezen ötletek, projektek megvalósítására az ötlet kitalálójánál az általunk készített oldal segítségével. Az az, egy olyan platformot hoztunk létre, ahol könnyen, gyorsan és egy helyen találkozhat a megbízó és a kivitelező. Ezt az oldalt inkább az ötlet adók szükségleteihez próbáltuk igazítani, hogy minél több olyan ember életét megkönnyítsük, akik akármilyen okból kifolyólag nem tudnak programozni, de szükségük van a mai fejlett világban egy minőségi szoftverre vagy informatikai segítségre.

A fent említett témájú és üzleti logikájú oldalt azért választottuk, mert könnyen átalakítható más, nem csak programozással kapcsolatos témájúvá, ezáltal nagyobb és változatosabb felhasználó közönségnek tudunk gyors és biztos segítséget nyújtani.

# 2. - Felhasznált technológiák és programok

## 2.1 - Node.js - Dani

A Node.js egy szerver oldali (back-end) JavaScript-futási környezet, ami a V8-as, Google által fejlesztett motoron fut. A Node.js képes JavaScript kód futtatására, web böngésző nélkül.

Nyílt forráskódú (open source), ami azt jelenti, hogy bárki, akármire, ingyen használhatja. Továbbá több, különböző számítógépes platformon is működik, mivel többplatformos (cross-platfromú).

Aszinkron eseményvezérelt JavaScript-futási környezetként a Node.js-t skálázható hálózati alkalmazások létrehozására tervezték.

A programunk back-end elkészítésére a Node.js-t választottuk.

## 2.2 - React

A React egy JavaScript könyvtár. React.js vagy ReactJS-nek is hívják. Felhasználói felületek készítésére használják ezt a JavaScript könyvtárat. A Facebook által van jelenleg fenntartva és néhány kisebb cég által.

A React dekleratív és komponens alapú. Könnyen készíthető vele igényes, egyedi és profi felhasználói felület. Az alkalmazás minden egyes állapotához létrehozható egy nézet. React-ot használva biztosíthatjuk alkalmazásunk gyors működését, mivel a React csak a megfelelő komponenseket fogja frissíteni és újrarenderelni az oldalon. Komplex felhasználói felület létrehozásánál a React komponenseket biztosít a készítőknek. A komponensek logikáját vanilla JavaScriptben lehet megírni.

## 2.2 – React Native

A React Native egy nyílt forráskódú UI szoftver keretrendszer, amelyet a Meta Platforms, Inc. hozott létre. Android, Android TV, iOS, macOS, tvOS, Web, Windows és UWP alkalmazások fejlesztésére szolgál azáltal, hogy lehetővé teszi a fejlesztők számára a React keretrendszer használatát a natív platform képességeivel együtt. Virtuális valóság-alkalmazások fejlesztésére is használják az Oculusnál.

## 2.3 - XAMPP - Dani

A XAMPP többplatformos, az Apache, a MySQL, a PHP és Perl rövidítése. Ez az ingyenes program lehetővé teszi webhelyek offline létrehozását a számítógépen lévő helyi webszerveren.

A fejlesztés során a XAMPP-ot a phpMyAdmin felület használatára, az adatbázis hostolására és a tesztelésére használtuk.

## 2.4 - phpMyAdmin - Dani

A phpMyAdmin egy ingyenes, nyílt forráskódú program, ami segítségével egy grafikus felületen adminisztrálható a MySQL adatbázis. Mind SQL paranccsal, mint grafikus felületen mindenféle adatbázishoz kapcsolódó művelet elvégezhető ezen a felületen.

## 2.9 - Visual Studio Code 2019 - Dani

A Visual Studio Code, más néven VS code, egy nyílt forráskódú kódszerkesztő, amelyet a Microsoft fejleszt.

Nem volt nehéz kiválasztani a számunkra ezt a fejlesztői környezetet, hiszen az első félévtől kezdve használjuk a VS code-ot, így elég szerteágazóan megismertük a működését. Rengeteg hasznos tulajdonsággal rendelkezik, többek között a beépített IntelliSense-el és Git-tel, amelyek nagy segítséget nyújtottak a fejlesztés során, mivel ezt a két funkciót rengetegszer használtuk.

# 3. - Programozási nyelvek

## 3.1 - JavaScript -Dani

A JavaScript egy script vagy programozási nyelv, amely lehetőséget ad complex műveletek létrehozását weboldalakon. Például dinamikusan változó tartalom létrehozására, multimédiák kezelésére és képek animálására alkalmas. Ezeken kívül még természetesen sok másra is alkalmas. A Node.js backend-ünk javascript-el íródott.

## 3.2 - SQL/MySQL - Dani

SQL (Structured Query Language) az általános nyelv, amelyet adatbázisokhoz való hozzáféréshez és a bennük lévő adatok manipulálására használnak.

Az XAMPP phpMyAdmin felületén SQL parancsokkal hoztuk létre a táblákat, amiket használ a projektünk. Sok tesztet végeztünk két vagy több tábla összekapcsolása során és ezek közé tartozott, hogy a phpMyAdmin felületen próbáltuk ki először az API hívásban/kérésben szereplő SQL kódot, hogy tényleg megfelelően működik-e.

## 3.3 - Bootstrap/React-Bootstrap

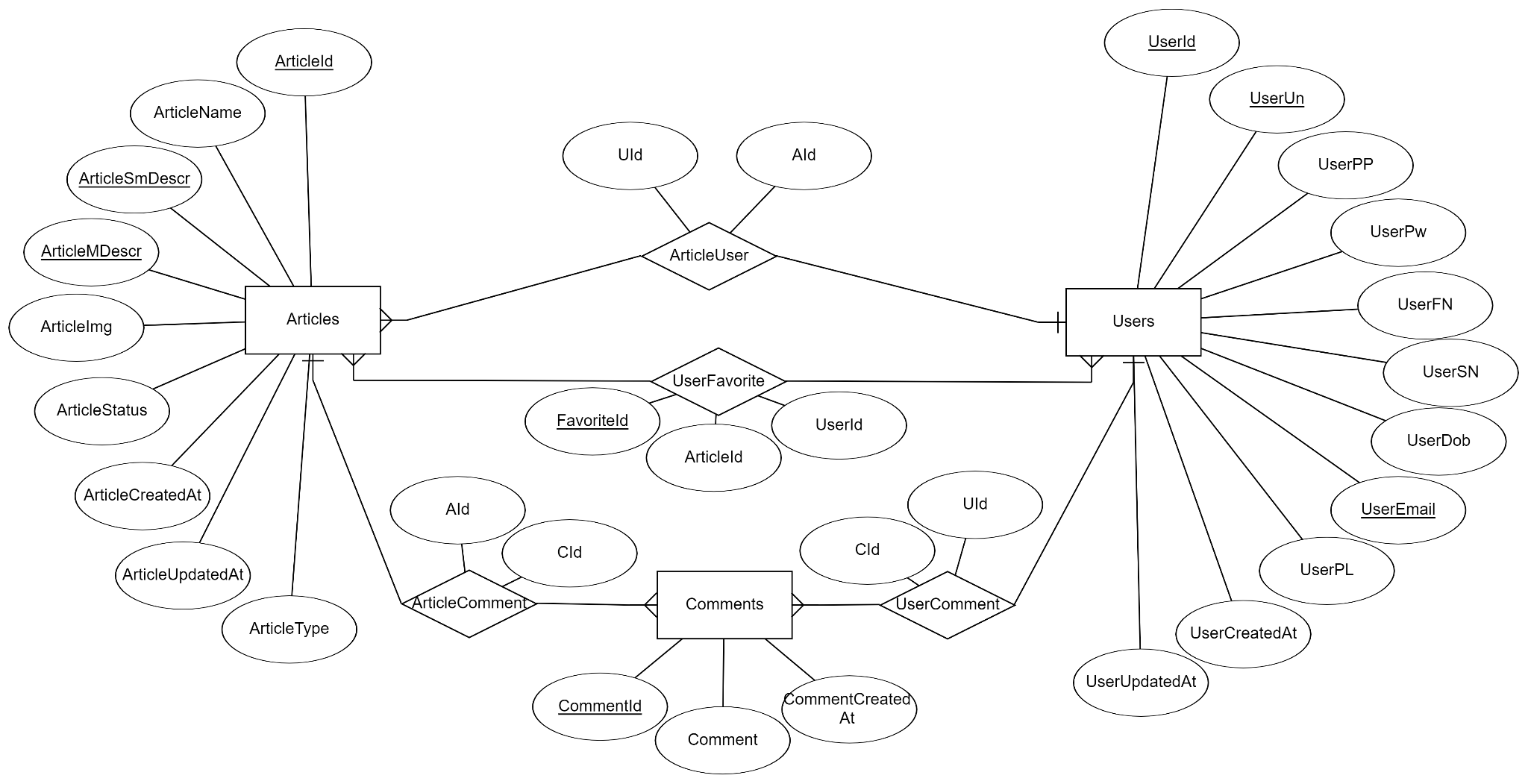
A Bootstrap a legnépszerűbb CSS keretrendszer mobile-first és reszponzív weboldalak készítésére. Jelenleg a Bootstrap 5 a legújabb verzió, mi is ezt a verziót használtuk a fejlesztés során.

A React-Bootstrap egy komponens alapú könyvtár, amely a natív Bootstrap komponenseket React komponensekké alakítja.

# 4. - Adatbázis

## 4.1- Az adatbázis bemutatása

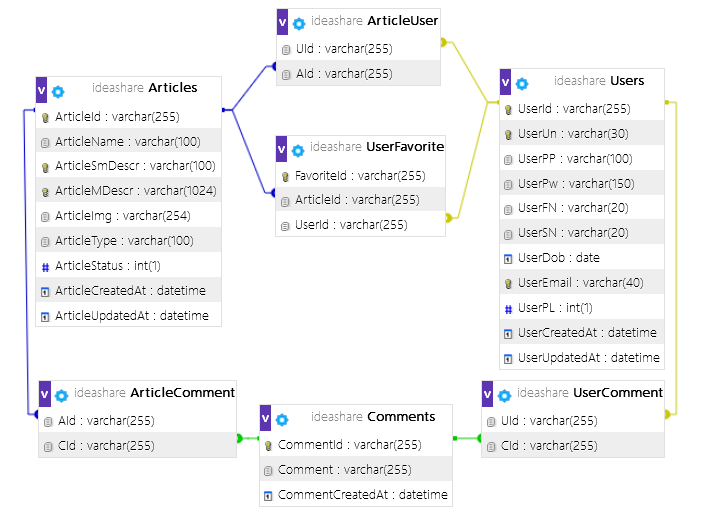
Az adatbázis megtervezéséhez és az egyed-kapcsolati modell (ER-modell) létrehozásához az erdplus (link: https://erdplus.com/) nevű weblapot használtuk.



1. ábra – Entity Relationship Diagram

Az adatbázis hét táblából áll, ebből 4 kapcsoló tábla. Egy Articles nevű táblából, egy Users nevű táblából, egy Comments nevű táblából, egy ArticleUser nevű kapcsoló táblából, egy UserFavorite kapcsoló táblából, egy A 1. ábrán képen látható a táblák közötti kapcsolatok. Egy felhasználónak több posztja lehet, de egy poszt csak egy felhasználóhoz tartozhat. Ehhez hasonló, de N:M, több a többhöz kapcsolatban áll a Users és a Articles tábla.

Az adatbázis relációs adatmodelljét az XAMPP nevű program ‘TERVEZŐ’ felületén készítettük.



2. ábra - phpMyAdmin kapcsolati nézete

## 4.2 - Relációs Sémák

Az adatbázis átláthatósága és az SQL parancsok könnyebb létrehozása érdekében elkészítettük a táblák relációs sémáit is.

Articles nevű tábla relációs sémája:

Articles [ArticleId, ArticleName, ArticleSmDescr, ArticleMDescr, ArticleImg, ArticleType, ArticleStatus, ArticleCreatedAt]

ArticleId = poszt id-ja, VarChar(255) típusú

ArticleName = poszt neve, VarChar(20) típusú

ArticleSmDescr = poszt rövid leírása, VarChar(100) típusú

ArticleMDescr = poszt fő leírása, VarChar(1024) típusú

ArticleImg = poszthoz tartozó kép linkje, TEXT típusú

ArticleType = poszt típusa, VarChar(100) típusú

ArticleStatus = poszt státusza, INT(1) típusú

ArticleCreatedAt = poszt létrehozásának dátuma, DATETIME típusú

ArticleUpdatedAt = poszt módosításának dátuma, DATETIME típusú

Users nevű tábla relációs sémája:

Users [UserId, UserUn, UserPP, UserPw, UserFN, UserSN, UserDob, UserEmail, UserPL, UserCreatedAt, UserUpdatedAt]

UserId = felhasználó id-ja, VarChar(255) típusú

UserUn = felhasználó felhasználó neve, VarChar(30) típusú

UserPP = felhasználó profil képe, VarChar(100) típusú

UserPw = felhasználó jelszava, VARCHAR(150) típusú

UserFN = felhasználó első, keresztneve, VarChar(20) típusú

UserSN = felhasználó második, vezetékneve, VarChar(20) típusú

UserDob = felhasználó születési dátuma, DATE típusú

UserEmail = felhasználó email címe, VarChar(40) típusú

UserCreatedAt = felhasználó létrehozásának dátuma, DATETIME típusú

UserUpdatedAt = felhasználó módosításának dátuma, DATETIME típusú

Comments nevű tábla relációs sémája:

Comments [CommentId, Comment, CommentCreatedAt]

CommentId = hozzászólás id-ja, VarChar(255) típusú

Comment = hozzászólás, VarChar(255) típusú

CommentCreatedAt = hozzászólás létrehozásának dátuma, DATETIME típusú

ArticleComment nevű kapcsoló tábla relációs sémája

ArticleComment [AId, CId]

AId = kommenthez kapcsolódó article id-ja, VarChar(255) típusú

CId = article-höz kapcsolódó komment id-ja, VarChar(255) típusú

UserComment nevű kapcsoló tábla relációs sémája

UserComment [UId, CId]

UId = a kommentet létrehozó felhasználó id-ja, VarChar(255) típusú

CId = a felhasználó által létrehozott komment id-ja, VarChar(255) típusú

ArticleUser nevű kapcsoló tábla relációs sémája:

ArticleUser [UId, PId]

UId = felhasználó id-ja, VarChar(255) típusú

AId = poszt id-ja, VarChar(255) típusú

A relációs sémák és a modellek segítségével könnyebben és gyorsabban tudtuk létrehozni a tábla generáló SQL parancsokat, valamint sokat segítettek abba, hogy az egész rendszer átláthatóbb legyen és a fejlesztés során hozott változtatások implementálása gyorsabb legyen.

# 5.0 - A vizsgaremek

A projektmunkát 3 fő részre osztottuk. Pócs Márk feladata volt a mobil frontend megvalósítása. Várdai Tamás feladata volt a webes admin felület létrehozása. Pap Dániel a webes alkalmazás frontend részét hozta létre. A backend részt együtt, kisebb-nagyobb részekre felosztva, egymásnak sokat segítve alkottuk meg.

## 5.1 - A backend bemutatása

### 5.1.1 - Általános leírása

A backend-et Node.js technológiával, JavaScript-el írtuk meg. A backend 4 fő részre osztható használati szempontból. 1. rész, azok a api hívások, amiket mindannyian használunk, azaz a mobilos alkalmazás, a webes alkalmazás és az webes admin felület is. A maradék 3 rész, pedig olyan api hívások, amelyeket csak a mobil felület vagy csak a webes felület használ. Olyan hívásra is van példa, amit a mobil és a webes felület használ együtt, az admin felület nem.

### 5.1.2 - Adatbázis használata

A fejlesztés során Pócs Márk által igényelt Nethely nevű oldalon használtunk közös adatbazásit, ezzel megkönnyítve a fejlesztés során történt változtatások integrációját a közös adatbázisba, mivel mind a hárman rendelkezünk ehhez a felülethez hozzáféréssel.

A végleges verzióban az XAMPP segítségével, a lokális hálózaton futó adatbázis adatai szerepelnek a programban.

A mobil alkalmazás viszont a Nethely adatbázist használja, amit a backend pontos másolatán keresztül ér el, amely a Heroku-n fut.

### 5.1.3. - Regisztráció

A regisztráció során a felhasználónak generálódik egy egyedi id, ami generálásához a nanoid-t használjuk. A felhasználónak opcionálissá tettük a profilkép feltöltését a regisztrációkor. Amennyiben nem tölt fel képet a felhasználó, egy alapértelmezett képpel fog létrejönni a felhasználó sikeres regisztráció esetén. Természetesen a profilképét a felhasználó a későbbiekben bármikor megváltoztathatja.

A felhasználó jelszavát a bcryptjs segítségével titkosítjuk. Az adatbázisban csak ez a titkosított jelszó található meg. Amennyiben a felhasználó már az adatbázisban szereplő username-el vagy email címmel szeretne regisztrálni, akkor egy hibaüzenet fogja erről tájékoztatni. A felhasználónévnek (username) és az email címnek egyedinek kell lennie.

Sikeres regisztráció esetén szintén értesítjük a felhasználót a sikeres regisztrációról. Fontos megjegyezni, hogy minden felhasználó alapértelmezetten 1-es szintű hozzáféréssel rendelkezik.

### 5.1.4 - Bejelentkezés

Amikor a felhasználó megpróbál bejelentkezni az oldalra a felhasználónevét és a jelszavát kérjük el tőle. A megadott felhasználónév alapján egy keresést végzünk az adatbázisban, hogy szerepel-e ilyen névvel felhasználó. Amennyiben nincs ilyen, akkor kiíratjuk a felhasználó számára a bejelentkezési felületen, hogy nem létezik ilyen felhasználó. Ha az adatbázisban szerepel a megadott felhasználónév, akkor egyéb adatok mellett a hash-ként szereplő jelszót is lekérdezzük az adatbázistól.

A lekérdezett jelszót, a bcryptjs egyik függvényének segítségével, összehasonlítjuk a felhasználó által megadott jelszóval. Amennyiben nem egyezik meg a két jelszó, akkor ezt is tudatjuk a felhasználóval. Ha egyezik a két jelszó, akkor létrehozunk egy session-t. A session létrehoz egy cookie-t is. Ezáltal, ha a felhasználó frissíti a weboldalt, miközben be van jelentkezve, továbbra is bejelentkezve marad.

A felhasználó id-jával, a jwt-t (JSON web token) felhasználva létrehozunk egy token-t. Ezt a tokent fogjuk használni a felhasználó hitelesítésére.

A mobil alkalmazásban szintén a token-t használjuk a hitelesítésre.

### 5.1.5 - Kijelentkezés

Kijelentkezés során a böngészőben tárolt cookie, a session és a token is törlésre kerül. Ezután lefut egy ellenőrzés, ami ezeknek a hiányát észleli és a megfelelő változók frissítésével kijelentkezteti a felhasználót.

A mobil alkalmazás is a token törlésével jelentkezteti ki a felhasználót.

### 5.1.6 - Hitelesítés

A webes alkalmazás használ védett oldalakat. A kezdő oldal kivételével, minden oldal védve van, csak bejelentkezés után lehet a védett oldalakra navigálni. Minden egyes védett oldalra való navigáláskor a frontend elküldi a bejelentkezéskor létrehozott token-t és a backend ezt ellenőrzi és csak sikeres ellenőrzés után engedi tovább a védett oldalra a felhasználót.

Sikertelen ellenőrzés esetén a felhasználót visszairányítjuk a kezdő oldalra, ahol a bejelentkezési felület és a regisztrációs felület érhető csak el. Ez az ellenőrzés bejelentkezés előtt is lezajlik, azaz nem érhető el egyik védett oldal sem.

### 5.1.7 - Article módosítás

Az article módosításához való eljutáshoz először szükség van két lépésre. Az egyik lépés az, amikor a felhasználó a webes alkalmazásban az Edit Article nevű oldalra navigál, ahol még nem tudja megváltoztatni egyik általa létrehozott posztját sem, hanem itt tudja kiválasztani közülük, hogy melyiket szeretné módosítani. Ezen az oldalon csak a bejelentkezett felhasználó saját posztjai jelennek meg. A második lépés, a poszt kiválasztása, amin a felhasználó változtatni szeretne.

Ez az egy módosítás alkalmával kerül sor egyik oldalról, egy másik, új oldalra való navigáláskor adatok átadása. Az első oldalon, ahol kiválasztja a felhasználó a posztot, amin változtatásokat szeretne elvégezni, megkapjuk a kiválasztott poszt id-ját, majd ezt elküldjük az új oldalra, ahol már csak ez az egy article jelenik meg. Az előző oldalról kapott id alapján történik a lekérdezés, ezért csak egy article jelenik meg.

### 5.1.8 - Komment létrehozás

Komment létrehozását a mobilos felületen végezhet el a felhasználó, az article-höz tartozó komment szekcióban. A küldés gombra nyomva egy POST kérés történik, amelyben a felhasználó és a article azonosítója, valamint maga a komment szövege és a létrehozás ideje szerepel. A szöveg és a létrehozás ideje bekerül a kommenteket tartalmazó táblába, egy NanoId által generált azonosítóval megjelölve. Ezután kapcsolótáblák segítségével összekapcsolja a kommenteket az article-el és a felhasználóval, az azonosítókat felhasználva.

### 5.1.9 – Favorite, Kedvencek

Ez a funkció a mobilos alkalmazásban érthető el. Amikor a felhasználó megnyomja az erre kialakított gombot, egy POST kérés megy a BackEnd felé, ami a UserId-t és az AticleId-t , valamint a kettő kombinációjából létrejött FavoriteId-t teszi be az erre használt kapcsolótáblába. A kedvencekből való eltávolítás pedig egyszerűen kitörli a rekordot FavoriteId alapján.

### 5.1.10 – Keresés

A keresés funkció

### 5.1.11 – Article lista

Az article-ök lekérdezése 20-asával történik, ugyanis ha az összes eredményt egyben lekérdeznénk, több millió rekord esetén súyíos teljesítményproblémák lépnének fel A frontenden a listanézet minden 20-adik elem után elküld egy GET kérést a BackEnd-nek amely lekérdezi a az Article és a hozzá tartozó User adatait, kapcsolótábla és egy dupla INNER JOIN segítségével. Mindig legfeljebb 20 rekordot küld vissza a frontend felé, változó OFFSET értékkel, amit a frontend listaeleme dönt el attól függően, hogy mennyi rekordot töltött már be a listába

# 6. - A vizsgaremek - A frontend bemutatása

A webes alkalmazás létrehozásához a React, illetve a React Native technológiát választottuk.

A mobilos frontend-et Pócs Márk, a webes admin felületet Várdai Tamás és a webes alkalmazás frontend-jét Pap Dániel készítette.

## 6.1 - A webes frontend bemutatása

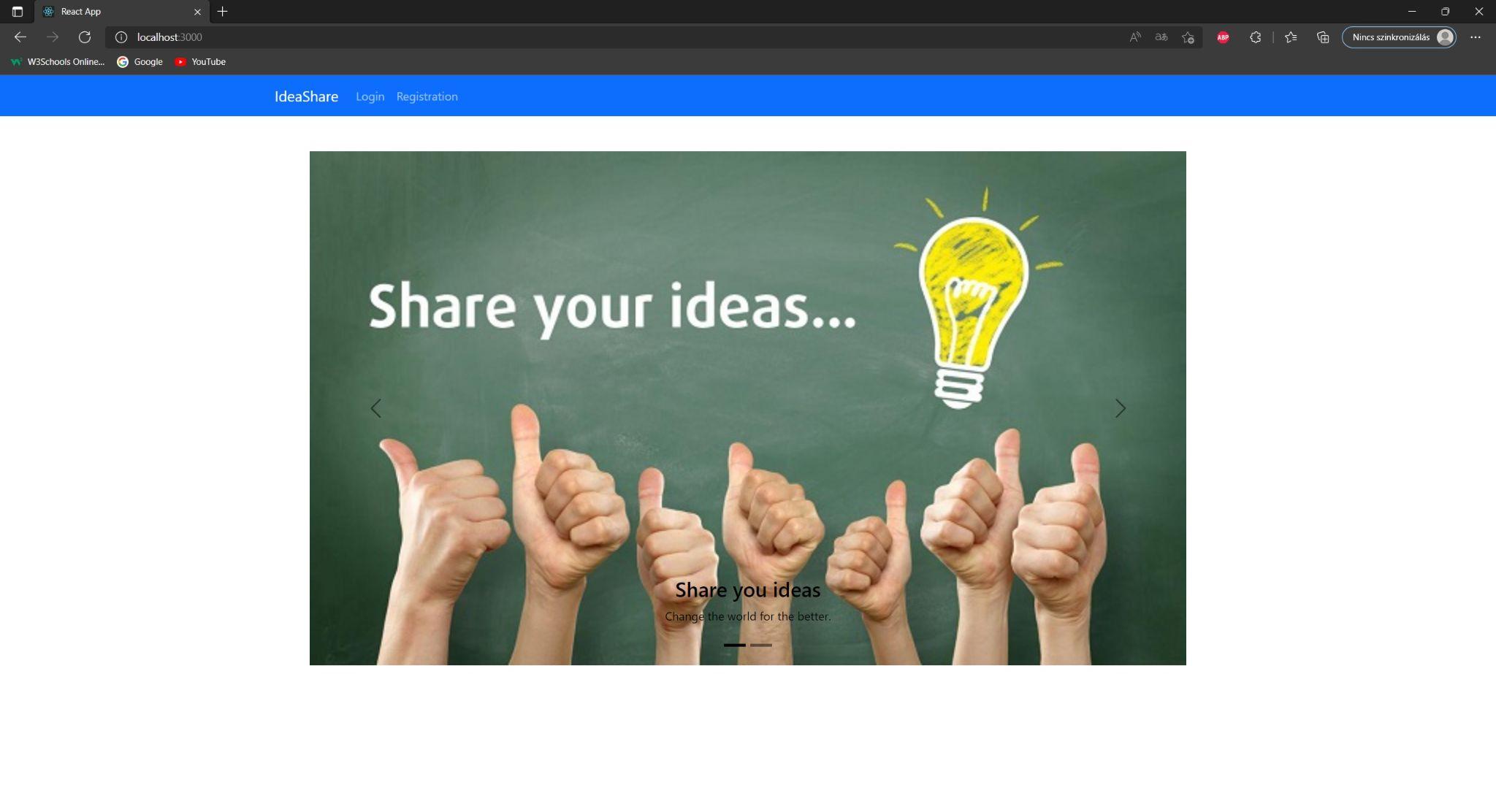
A webes alkalmazás tesztelésére a PapDaniel1997 felhasználónévvel és PapDaniel1997 jelszóval rendelkező felhasználót használtam.

Az alkalmazás futtatásához a backend-nek is futnia kell. A backend-et a megfelelő mappában való npm install, majd ezt követően npm run devStart parancsok kiadásával lehet elindítani. A frontend-et elindítása az npm install majd npm start parancsok kiadásával lehetséges.

A kinézet tervezésekor törekedtünk mindenhol könnyen átláthatóvá tenni a felületeket a felhasználó számára és minden oldalt reszponzívvá kialakítani és mobilon való használatra is alkalmassá tenni.

### 6.1.1 - A webes frontend bemutatása - Kezdőlap

A webes alkalmazás kezdőlapján egy két képből álló slideshow található. A menüsor az oldal tetején található. Bejelentkezés előtt csak két menüpont jelenítődik meg. A login, bejelentkezés és a registration, regisztráció.



3. ábra - Webes alkalmazás - Kezdőlap

Ezekre rákkattintva navigálhatunk a két felület között. Mind a két felület ugyanúgy működik, ami azt jelenti, hogy kattintás után egy teljes képernyős bejelentkezési vagy regisztrációs űrlap jelenik meg.

A bejelentkezési és regisztrációs űrlap és az összes további űrlap a webes alkalmazásban frontend-en lévő valódációval rendelkezik. Amíg a felhasználó a feltételeknek megfelelően nem tölti ki az űrlapot, addig a “Submit” gombra kattintásra semmi nem történik, nincs kommunikációra frontend és backend között. Ezekkel a validációkka ki lehet szűrni a véletlen vagy fölösleges kéréseket a backend felé. Továbbá a hibás vagy az adatbázis számára nem megfelelő adatokat se engedjük így át a backend-nek és a backend az adatbázisnak.

### 6.1.2 - A webes frontend bemutatása - Regisztrációs felület

A regisztrációs felület a felhasználótól a profilkép megadásán kívül minden kötelezővé tettünk. Amennyiben a felhasználó nem tesz eleget a validációs feltételek, akkor nem tud regisztrálni. Ha hibásan töltötte ki a felhasználó az űrlapot és úgy kattint rá a Register gombra, akkor az oldalon a megfelelő beviteli mezők alatt megjelennek a hiba üzenetek. Ezekkel felhívva és tájékoztatva a felhasználót a hibák felismerésére.



4. ábra - Webes alkalmazás - Regisztrációs felület

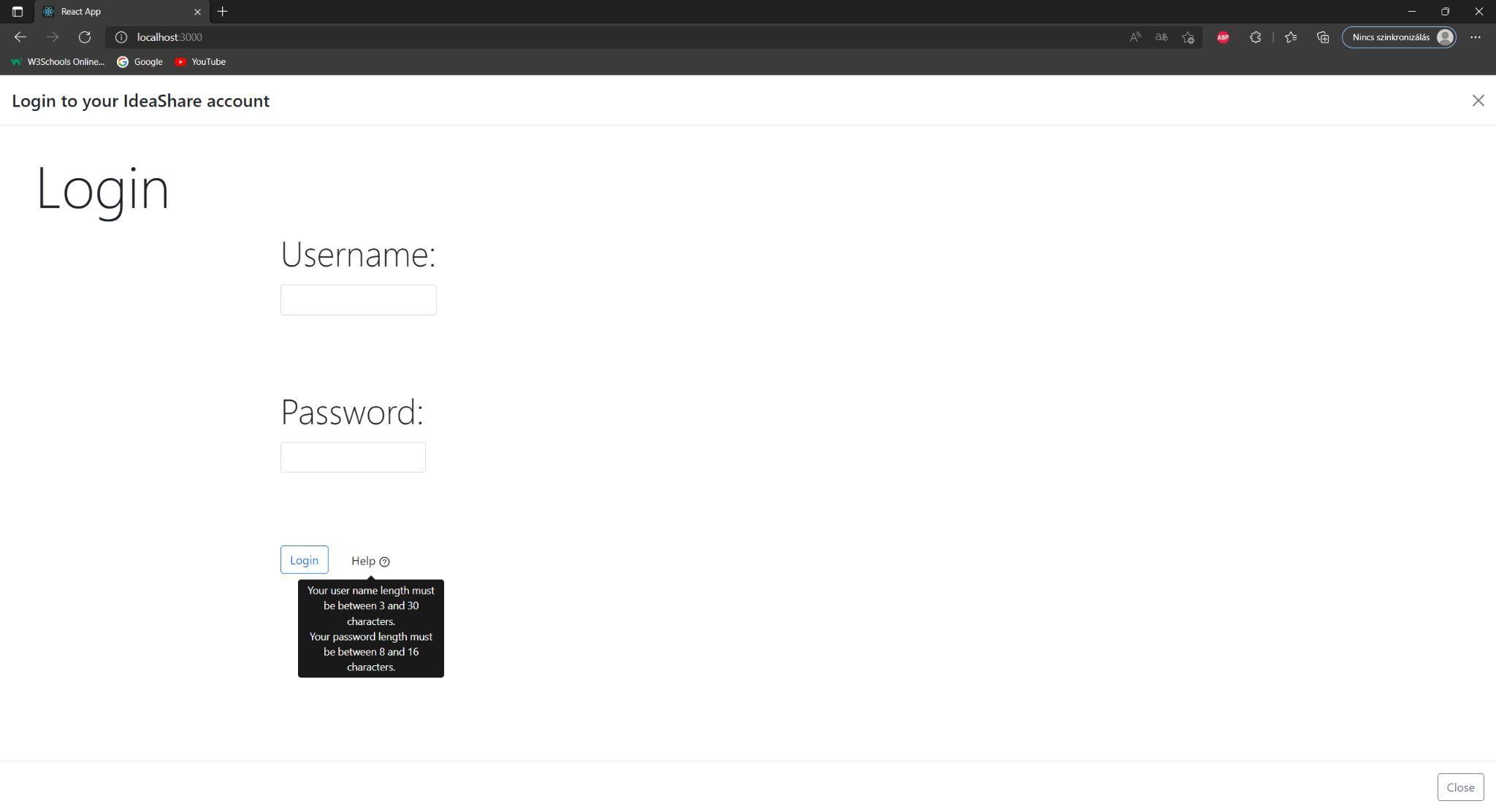
A 4. ábrán egy üres űrlapot próbálunk meg elküldeni, de a Register gombra való kattintás után pirossal megjelennek a hibásan, esetünkben üresen hagyott mezők alatt a hiba üzenetek.

Az email és a születési dátum beviteli mezőn külön ellenőrzések is végrehajtódnak. Az email mezőben szerepelnie kell a ‘@’ jelnek, valamin egy ‘.’-nak. A születési dátum mezőnél be lehet vinni hibás adatot, de a Register gombra való kattintás után további hibaüzenetekkel jelzi a felület a felhasználónak, hogy “hamis” dátumot adott meg.

Sikeres regisztráció esetén egy a képernyő közepén megjelenő kis ablak jelzi a felhasználó számára, hogy jól regisztrált és a kis ablak felületén megjelent gombra kattintva átirányítódik a bejelentkezési felületre.

### 6.1.3 - A webes frontend bemutatása - Bejelentkezési felület

A bejelentkezési felületen a felhasználónak a felhasználónevét és a jelszavát kell megadnia a bejelentkezéshez.



5. ábra - Webes alkalmazás - Bejelentkezési felület

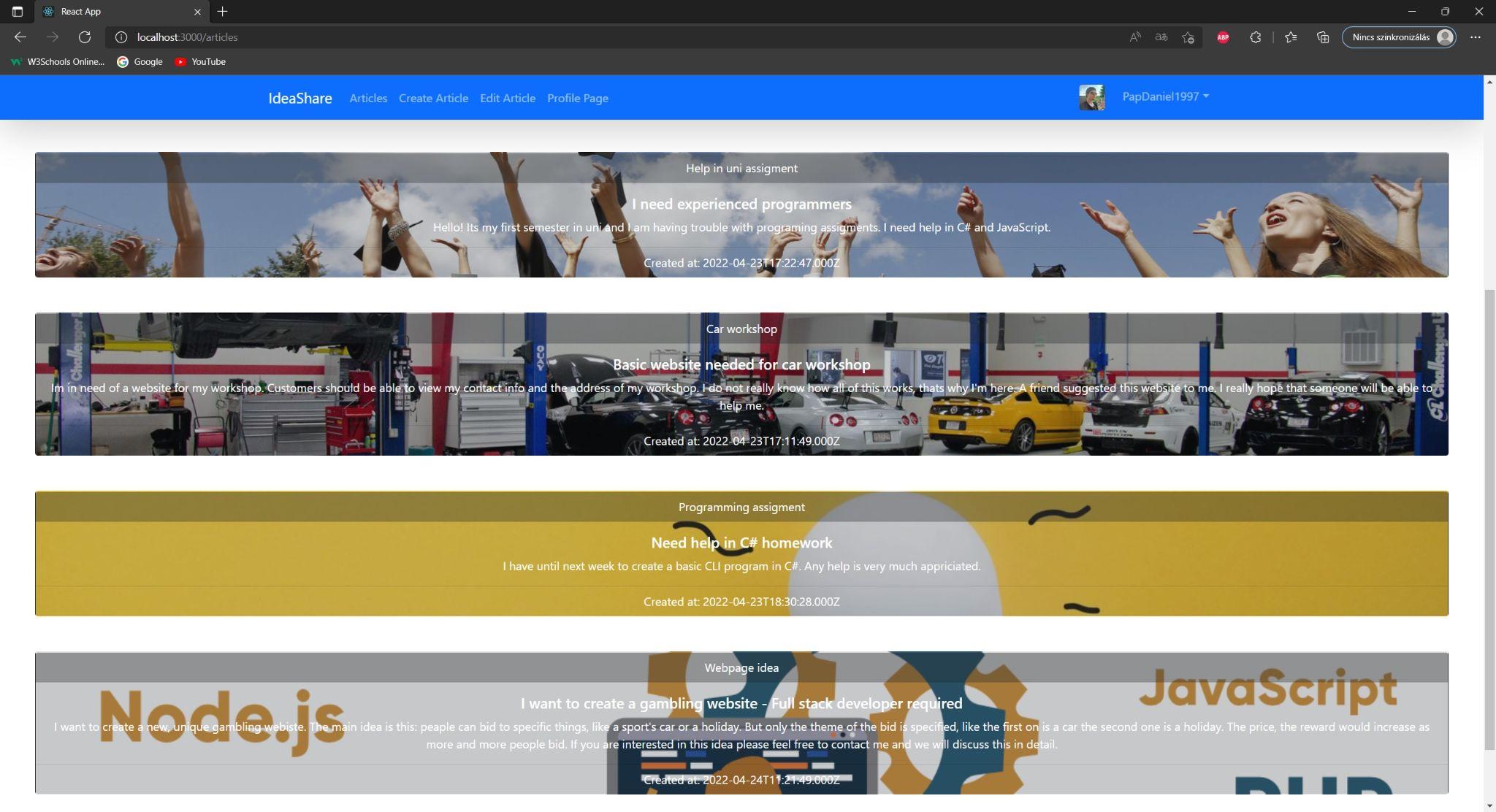
Ezen a felületen is, mint a regisztrációs felületen, ha üresen vagy hibásan próbálja meg elküldeni a felhasználó az adatokat, akkor a megfelelő beviteli mezők alatt megjelennek a hibák, illetve a figyelmeztetések. Figyelmeztetés akkor jelenik meg ha a felhasználó a validációs kritériumoknak eleget téve küldte el az adatokat, de elírta az egyik adatát és nem sikerült bejelentkeztetni.

Az 5. ábrán látható, hogy ez a felület egy segítség ‘gombbal’ is rendelkezik.

Sikeres bejelentkezés esetén az oldal egyből tovább irányítja a felhasználót az ‘Articles’ nevű oldalra.

### 6.1.4 - A webes frontend bemutatása – Articles oldal

Sikeres bejelentkezés után a felhasználó az Articles nevű oldalra irányítódik, ahol az összes aktív státuszú poszt megjelenik.

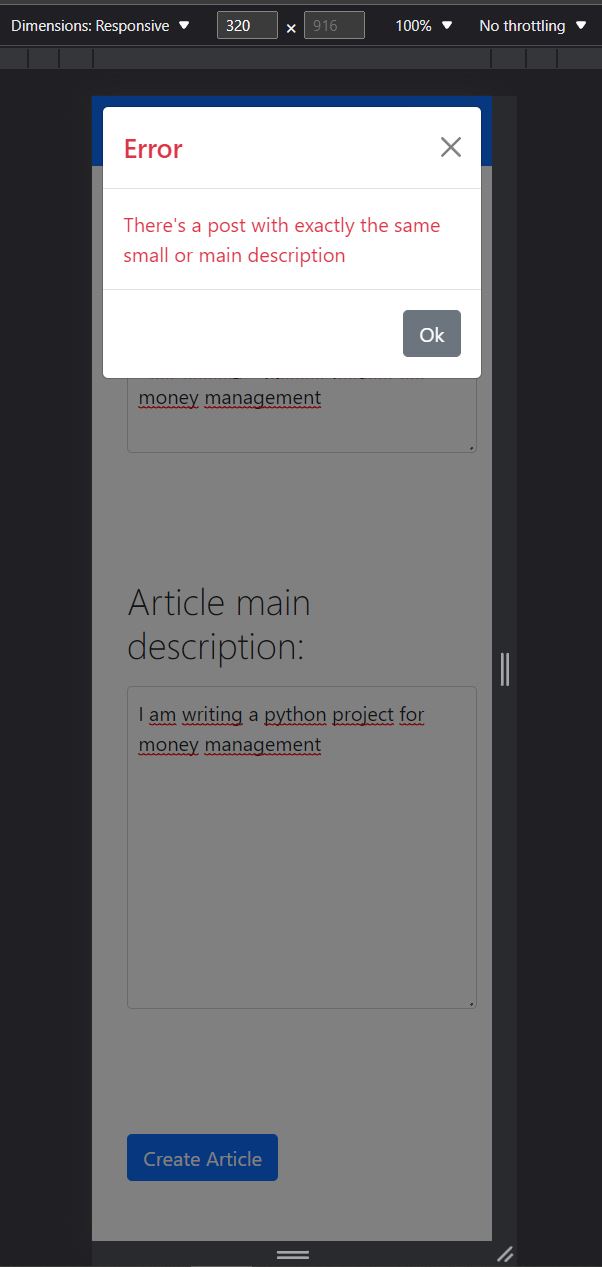


6. ábra - Webes alkalmazás - Article oldala

A felhasználó itt tud keresgélni a posztok között. A posztok betöltése és megjelenítése 20-asával történik. Ha lejjebb görget a felhasználó az oldalon, akkor újabb 20 poszt kerül betöltésre és megjelenítésre.

### 6.1.5 - A webes frontend bemutatása - Create Article

A Create Article oldalon van lehetősége a felhasználónak posztot létrehoznia. Az egész oldal egy nagy űrlap. A sikeresen létrehozott article egyből ‘Aktív’ státusszal kerül be az adatbázisban és azonnal látható lesz az Articles vagy Edit Article oldalon. Ezen az oldalon tökéletesen demonstrálható az egész webes alkalmazásra jellemző reszponzivítás.



7. ábra - Webes alkalmazás - Reszponzivítás demonstrálása

Akár kis mobiltelefonon is tökéletesen használható a webes alkalmazás.

### 6.1.6 - A webes frontend bemutatása - Edit Article

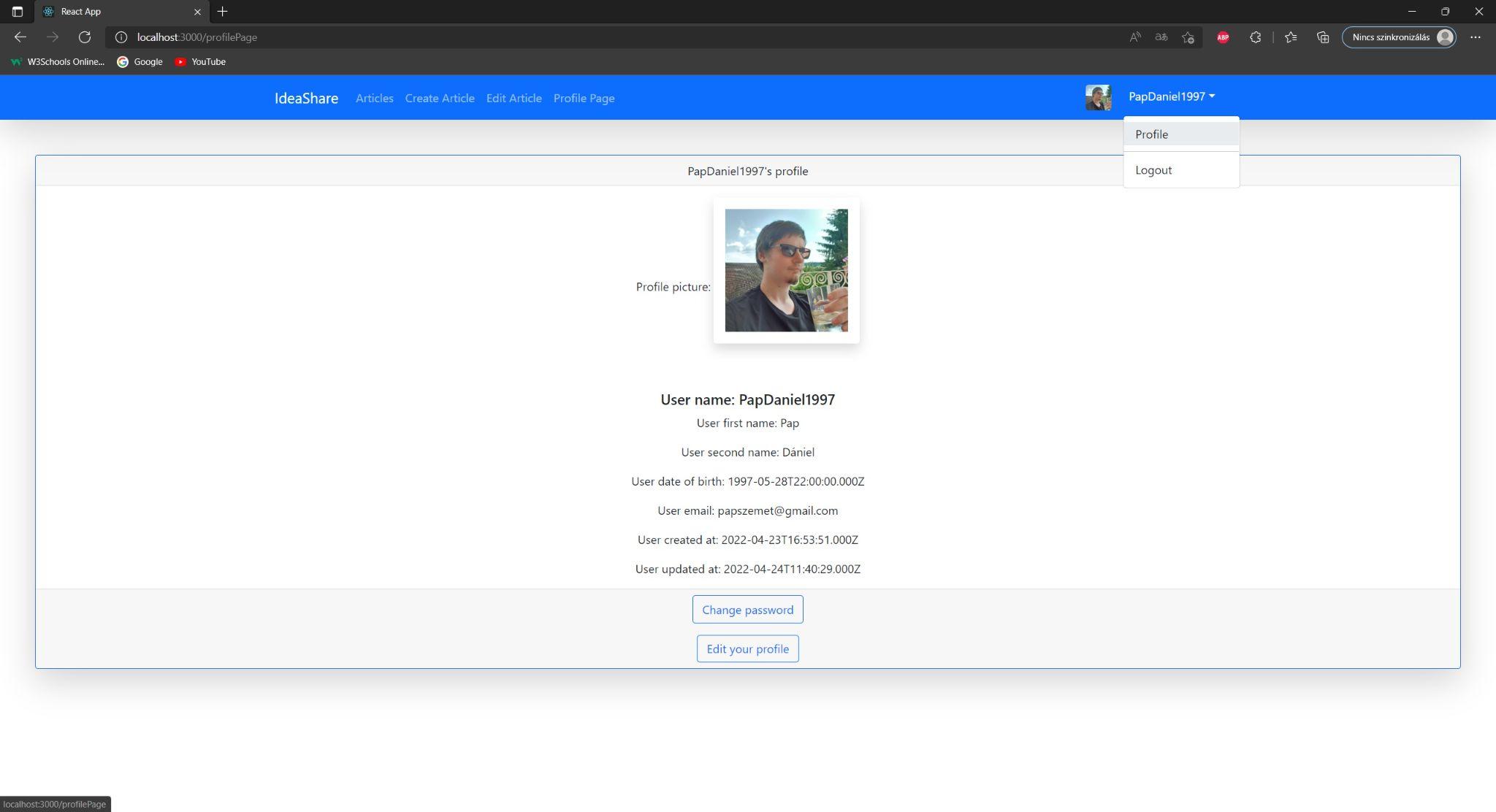
Az Edit Article menüpontra kattintva a felhasználó megnézheti az összes, általa létrehozott posztot, akármilyen is legyen a poszt státusza. A posztoknak négy státusza lehet, Aktív, Törölt, Inaktív és Felfüggesztett, ezeket is tudja a felhasználó állítani a saját posztjain, azzal a kivétellel, hogy nem állíthat be felfüggesztett státuszt, azt csak egy admin az admin felületen állíthathat be.

A poszt adatainak megváltoztatására alkalmas felület csak egy dologban tér el a Create Article nevű felülettől. Ez az eltérés egy plusz legördülő beviteli mező, amely segítségével a poszt státuszát tudja megváltoztatni a felhasználó.

Ha a felhasználó Törölt vagy Inaktív státuszt állít be a posztjának, akkor az a poszt nem fog megjelenni az Articles nevű oldalon, de továbbra is látható lesz a saját posztjai között, az Edit Article nevű oldalon.

### 6.1.7 - A webes frontend bemutatása - Profile Page

A profile page-re kétféle módon lehet eljutni. Az egyik lehetőség a menüsávon a Profile Page-re való kattintással. A másik lehetőség szintén a menüsávon található, de el van “rejtve” egy legördülő menübe. Ezt a legördülő menüt a bejelentkezett felhasználó nevére való kattintással lehet előhozni.



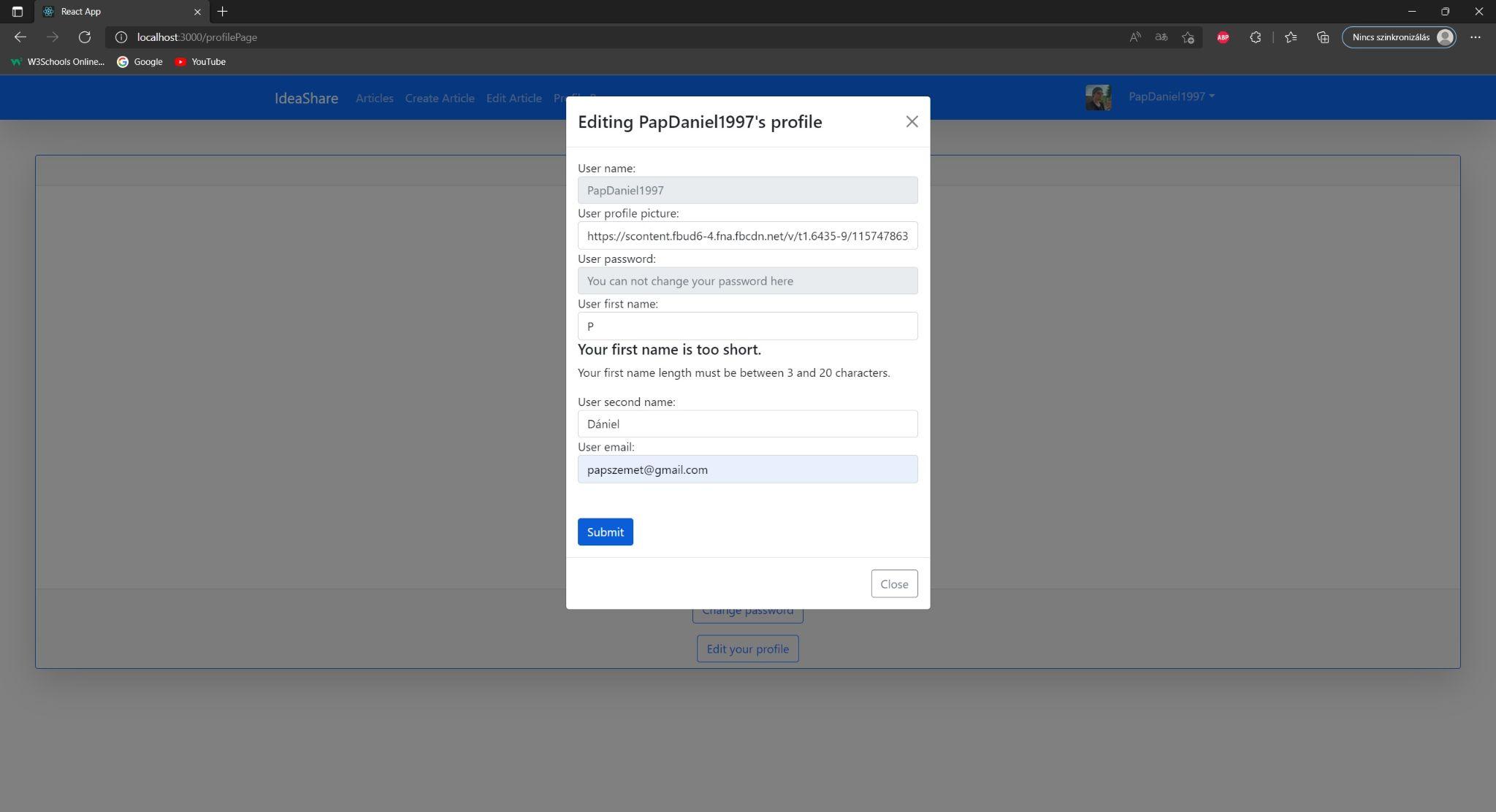
8. ábra - Webes alkalmazás - Porifle Page oldal

A 8. ábrán látható a profil felületről egy képernyőkép. A jobb felső sarkában a képnek látható a legördülő menü.

Ezen a felületen megjelenik a felhasználó összes adata. Külön van lehetőség jelszó változtatásra és külön, a felhasználó többi adatának megváltoztatására.

A felhasználó a felhasználónevét és születési dátumát nem tudja sehol sem megváltoztatni, minden más adatát igen. Egyedül az email címnek kell egyedinek lennie, az az, ha már az adatbázisban szereplő email címet ad meg, akkor nem hajtódik végre az adatok megváltoztatása. Erről egy előugró ablak értesíti a felhasználót, hogy egy már használatban levő email címre próbálta meg megváltoztatni a sajátját.

Ezen a felületen is minden beviteli mező ellenőrizve van, nem tudja a felhasználó ‘hibás’ adatra modosítani a meglévőket. A következő, 9-es ábrán erre látható egy példa.



9. ábra - Webes alkalmazás - Validáció bemutatása

A 9. ábrán a felhasználó megpróbálja a vezeték nevét egybetűsre megváltoztatni, de a Submit gombra való kattintás után a felület kiírja, hogy ‘hibás’ adat szerepel a beviteli mezőben.

### 6.1.8 - A webes frontend bemutatása - Menüsor és kijelentkezés

A menüsor minden oldalon, mindig látható, kivéve a teljes képernyős előugró ablakok esetében, mint például a bejelentkezési vagy regisztrációs felület. Ha lejjebb görget a felhasználó, akkor is a böngésző tartalmat megjelenítő részének a tetején ott marad a menüsor.

A kijelentkezési menüpontot az oldalról csak a legördülő menü sorból lehet elérni. A ‘Logout’-ra való kattintás után egy előugró ablak kérdezi meg a felhasználótól, hogy biztosan ki szeretne-e jelentkezni. Ha az ‘ok’-éra kattint, akkor kijelentkeztetődik a felhasználó, de van lehetősége az X-re vagy a Cancel gombra kattintania, ezekben az esetkeben eltűnik az előugró ablak és bejelentkezve marad a felhasználó.

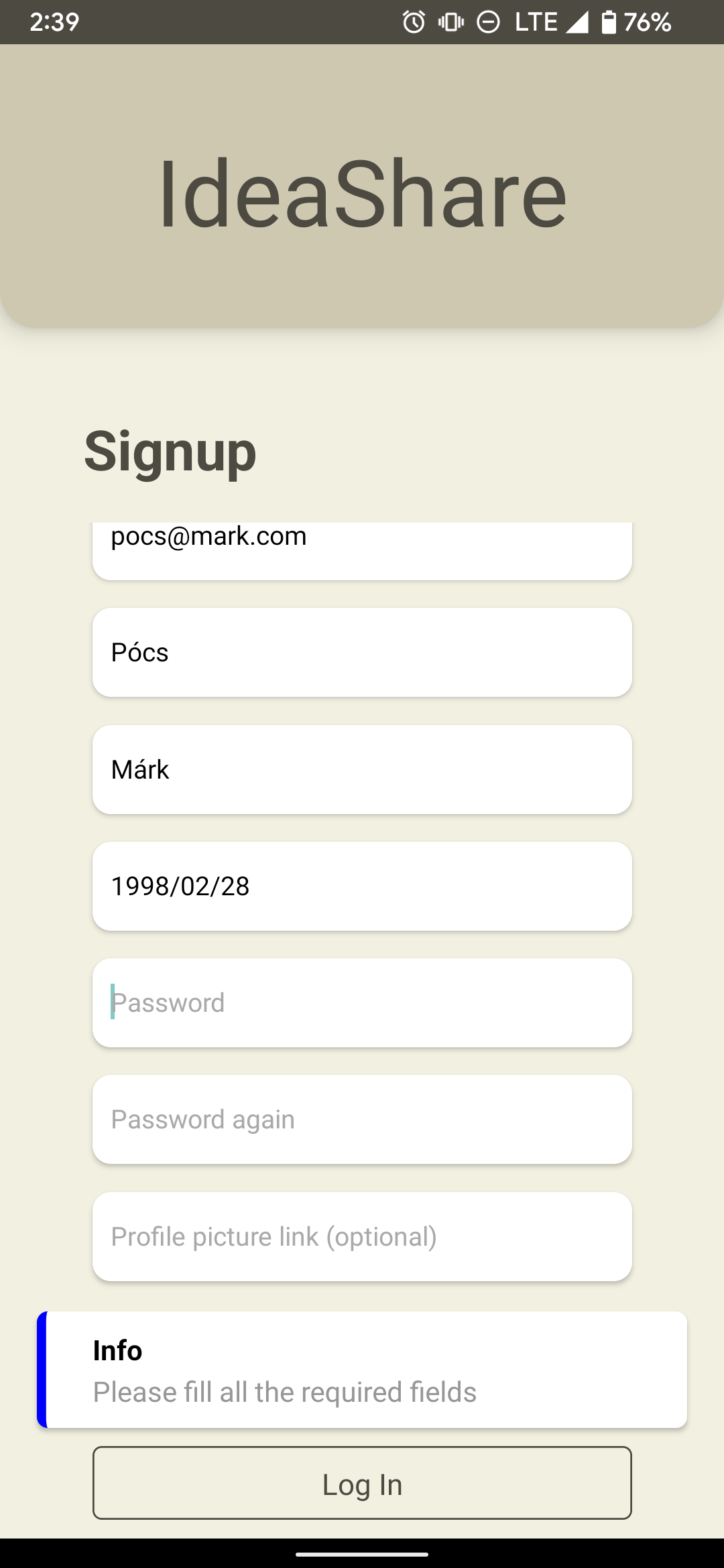
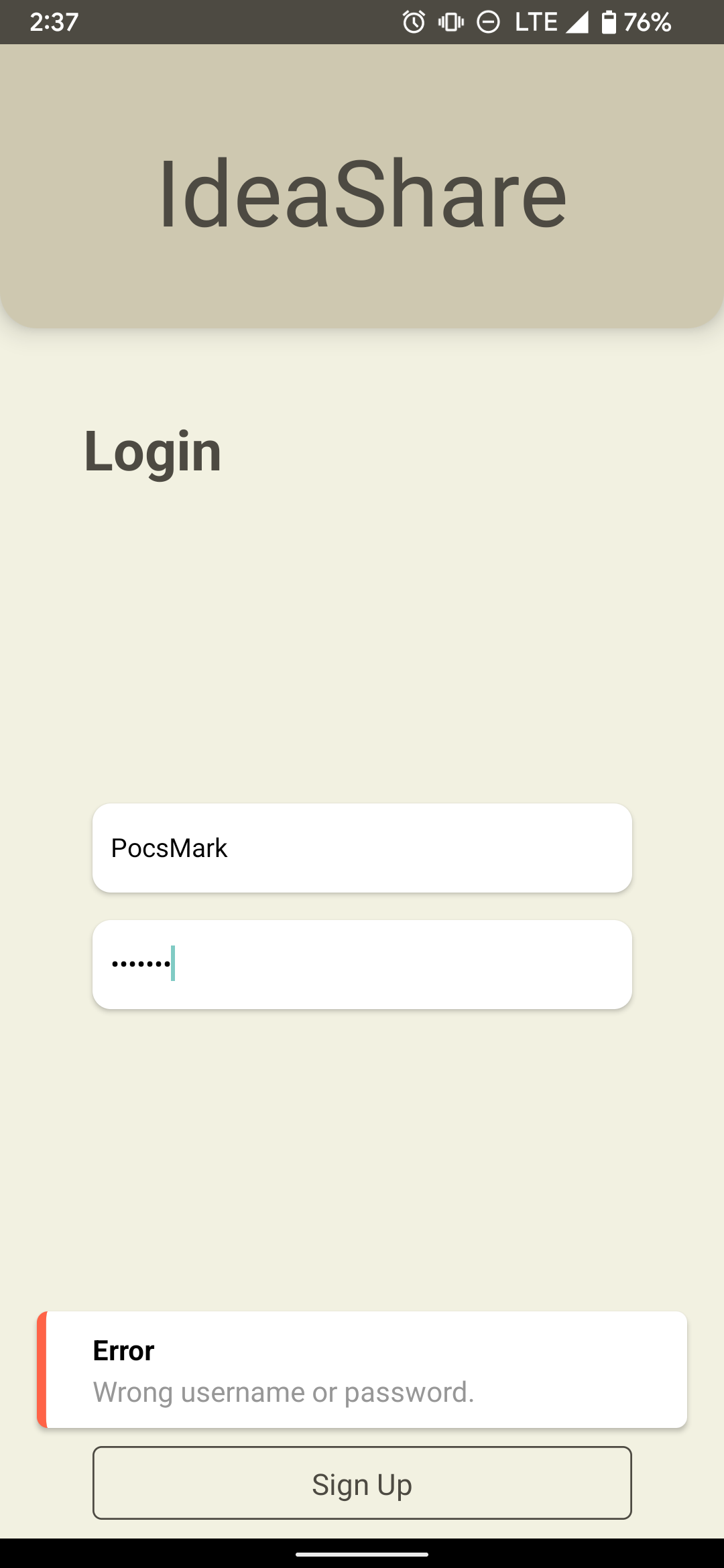
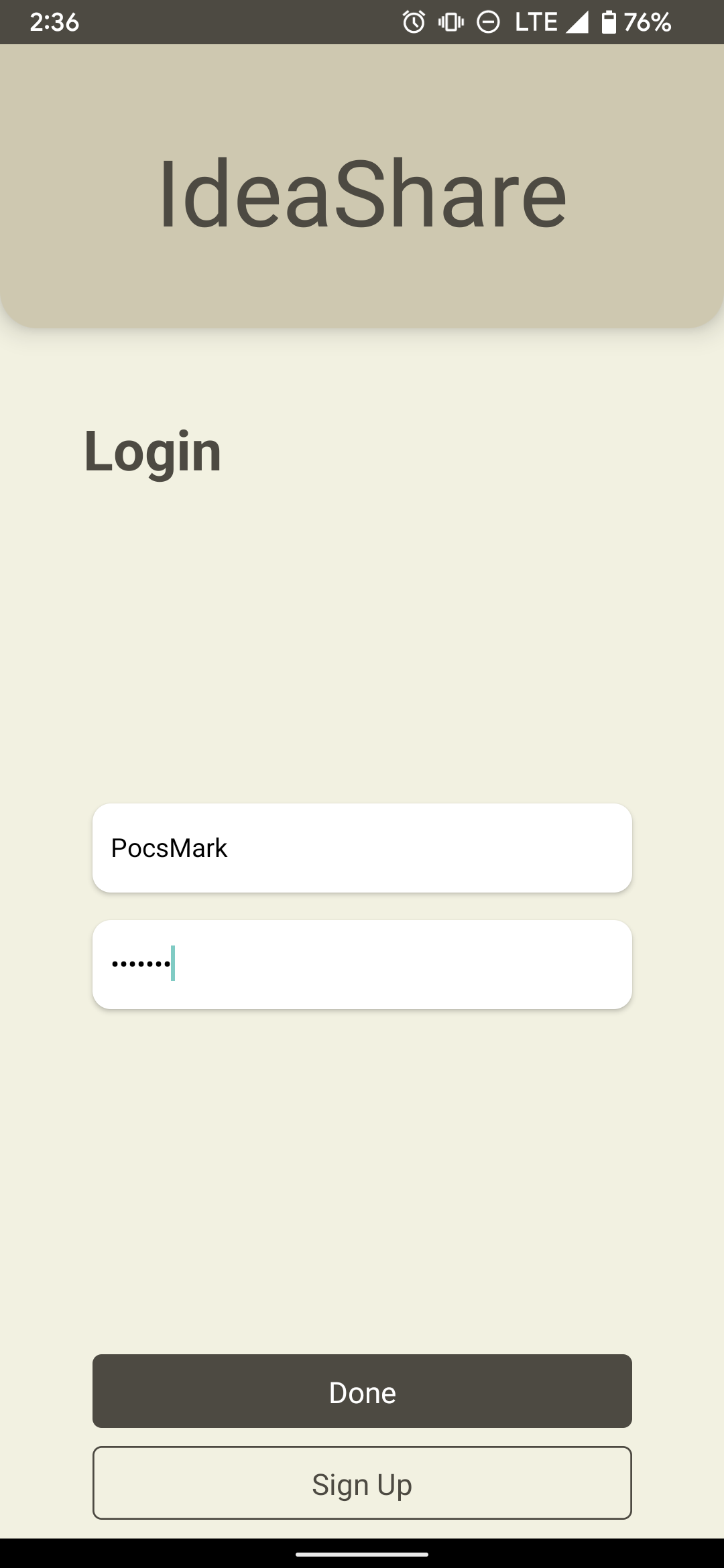
## 6.2 - A mobilos frontend bemutatása

A mobil frontend Android platformra íródott. Itt is törekedtünk az egyszerűségre, átláthatóságra. Az alkalmazás 6 képernyóre van felosztva, ezek két csoportra, stack-re vannak bontva. Bejelentkezett stack és Kijelentkezett stack. Azt, hogy melyik stack aktív, a token megléte dönti el. Kijelentkezett stack esetében a bejelentkezési képernyó érhető el, amely tartalmazza a regisztációs felületet is. A Bejelentkezett stack pedig az összes többi képernyőt tartalmazza, amik a kövezkezők: Home, My Ideas, Favorites, New Idea, és My Profile.

### 6.2.1 - A mobilos frontend bemutatása – Bejelentkezési és Regisztációs képernyő

Legelső alkalommal ez a épernyő fogaja a felhasználót. Felül olvasható az alkalmazás neve, középtájt pedig a bejelentkezési, vagy regiszrációs felület látható attól függően, hogy a körvonalas gomb segítségével melyiket választja a felhasználó. A regisztrációs felület görgethető, hogy kiférjen az összes kitöltendő mező a képernyőre akkor is, ha túl kicsi lenne a készülék, vagy annak DPI-ja.

Ha a felhasználó valamelyik kötelező mezőt nem tölti ki, illetve nem egyeznek a beírt jelszavak a regisztrációnál, vagy hibás a jelszó illetve felhasználónév a bejelentkezésnél, hibaüzenetet dob fel egy úgynevezett Toast message-ben.



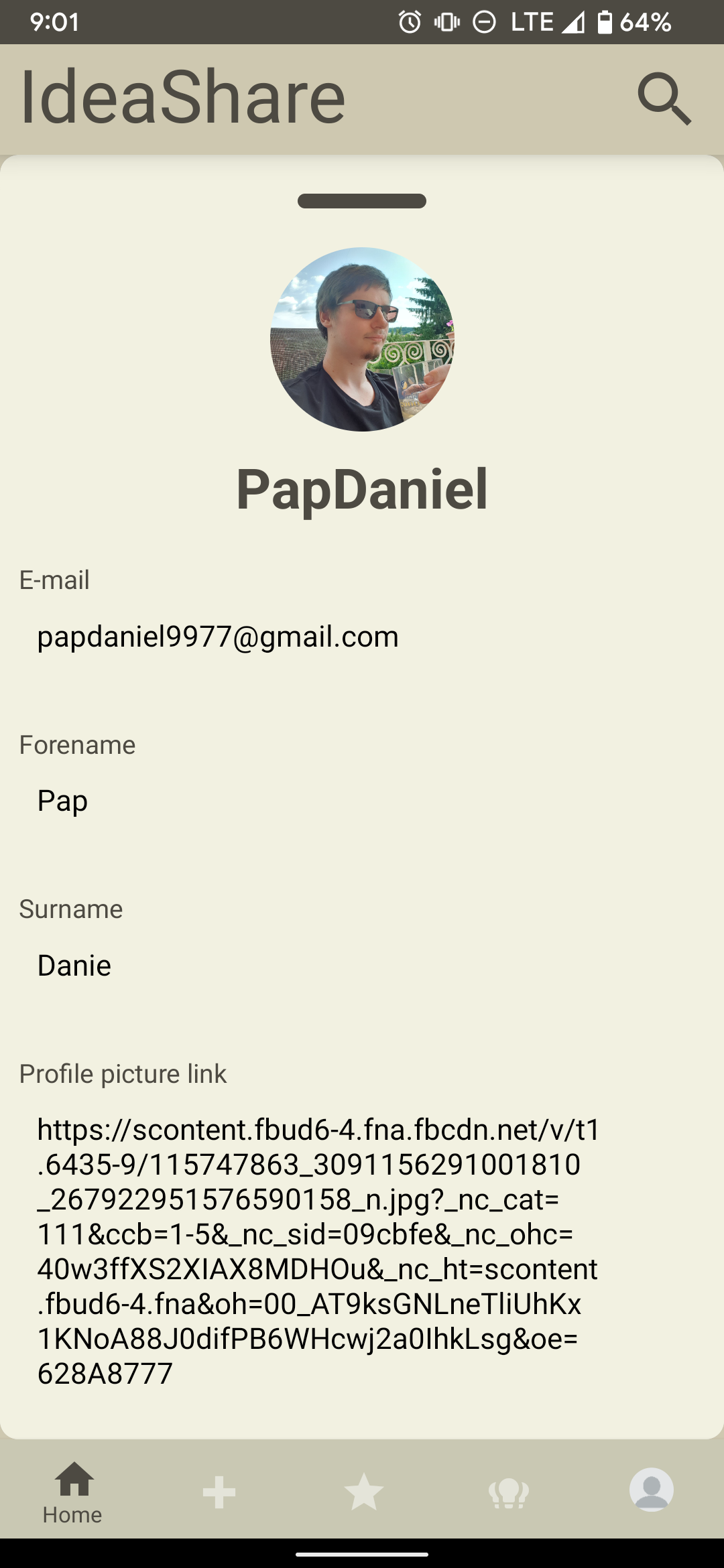
### 6.2.2 - A mobilos frontend bemutatása – Home

A bejelentkezés után ez a képernyő fogad. Itt jelennek meg a megosztott ötletek kártya formában. Minden feltöltött ötlet megjelenik ezen a képernyőn időrendi sorrendben a legutójára frissítettel kezdve, abban az esetben, ha az aktív státuszban van. Üres területre nyomva a kártya teljes képernyőssé növekszik, és a rövid helyett a hosszú leírás jelenik meg. Amennyiben így sem férne ki a hosszú leírás, görgethetővé válik a szövegmező. A kártya tartalma fentről lefelé haladva:

A kártya felső részében a bejegyzést létrehozó felhasználó profilképe és felhasználóneve jelenik meg. Ezt megérintve megjelennek a felhasználó további adatai: Az e-mail címe, a vezetékneve, a keresztneve, valamint profilképének linkje. Ezek az adatok hosszú rányomással kijelölhetőek, másolhatóak. Szintúgy görgethetővé válik amennyiben nem férnek ki az adatok a képernyőre. Ez az előugró ablak vissza gombbal, vagy a tetején lefelé pöccintéssel bezárható.

A középső része két részre osztható: Itt jelenik meg a bejegyzés címe, valamint rövid és hosszú leírása.

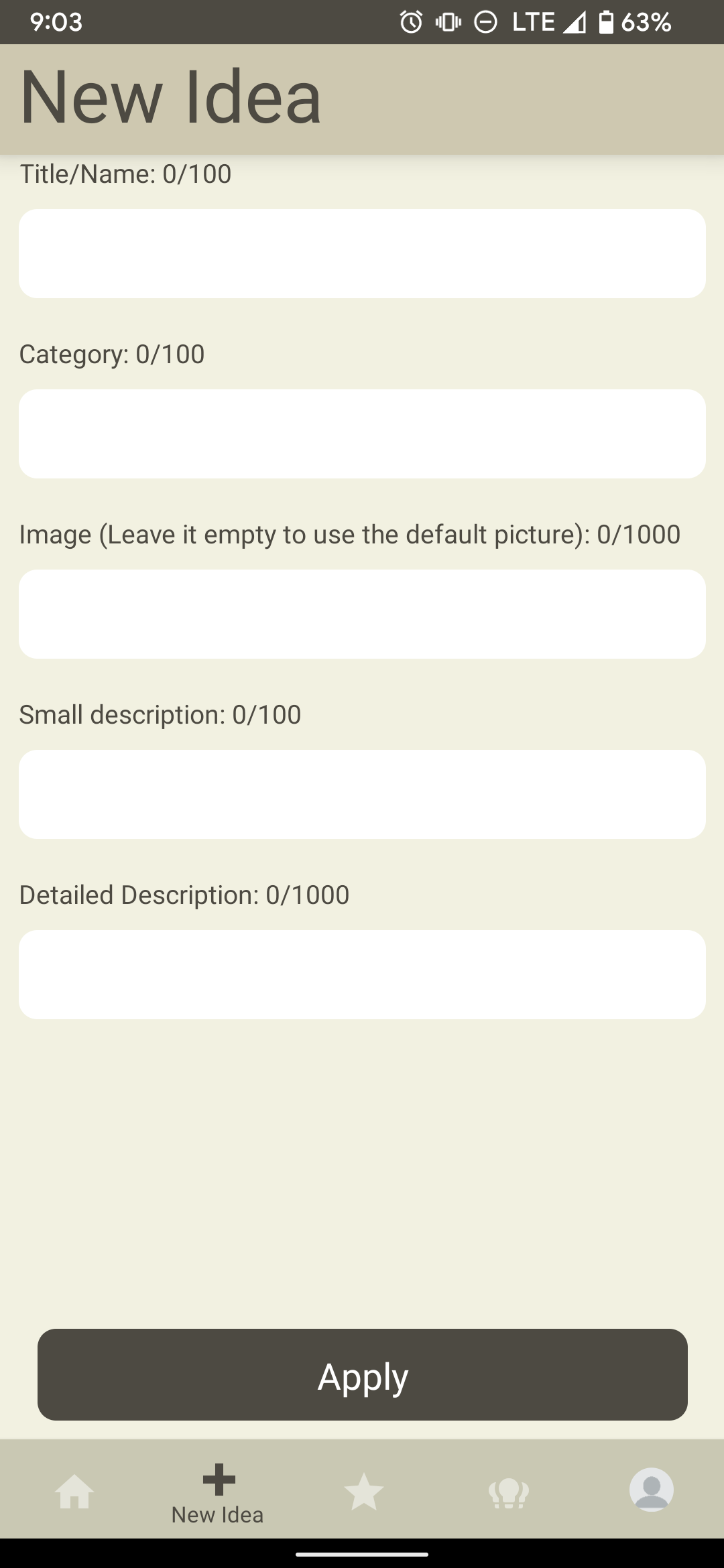
Az alsó részén két gomb látható, A kedvencekhez adás, valamint a komment szekció, ezekről lentebb lesz szó.



### 6.2.2 - A mobilos frontend bemutatása – Favorites

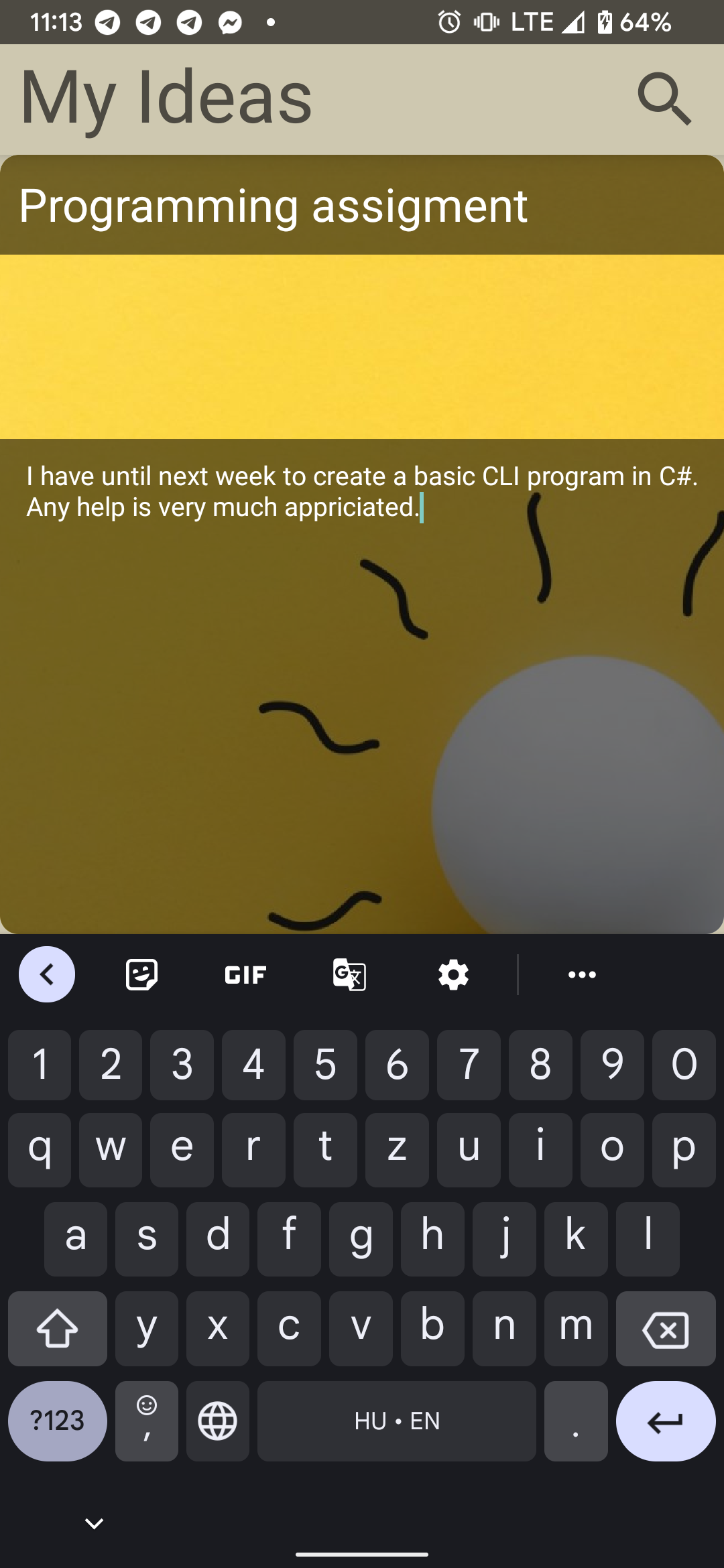
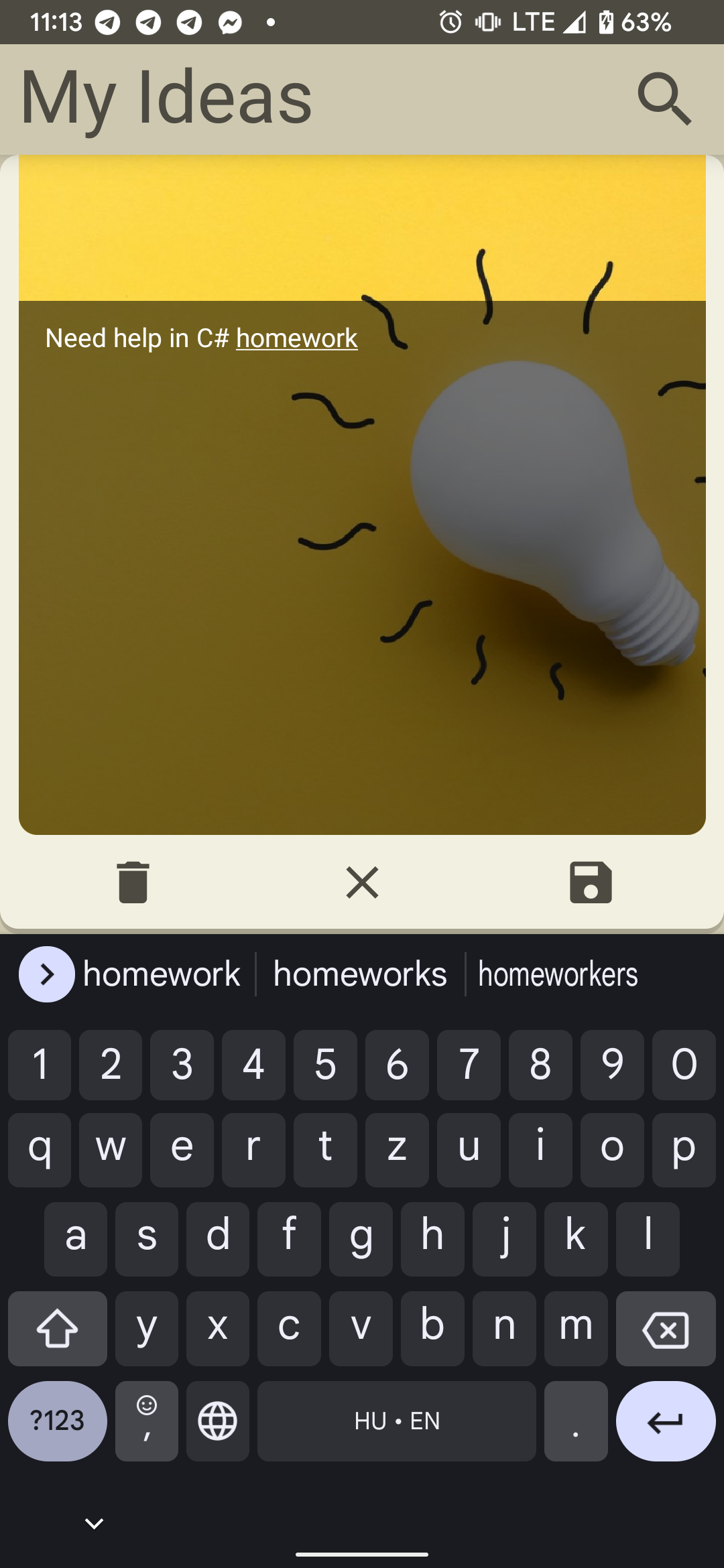
Ide a Home képernyőn kedvencekbe tett bejegyzések, ötletek kerülnek be. Az X gomb megnyomásával kitörölhető a kedvencek közül.

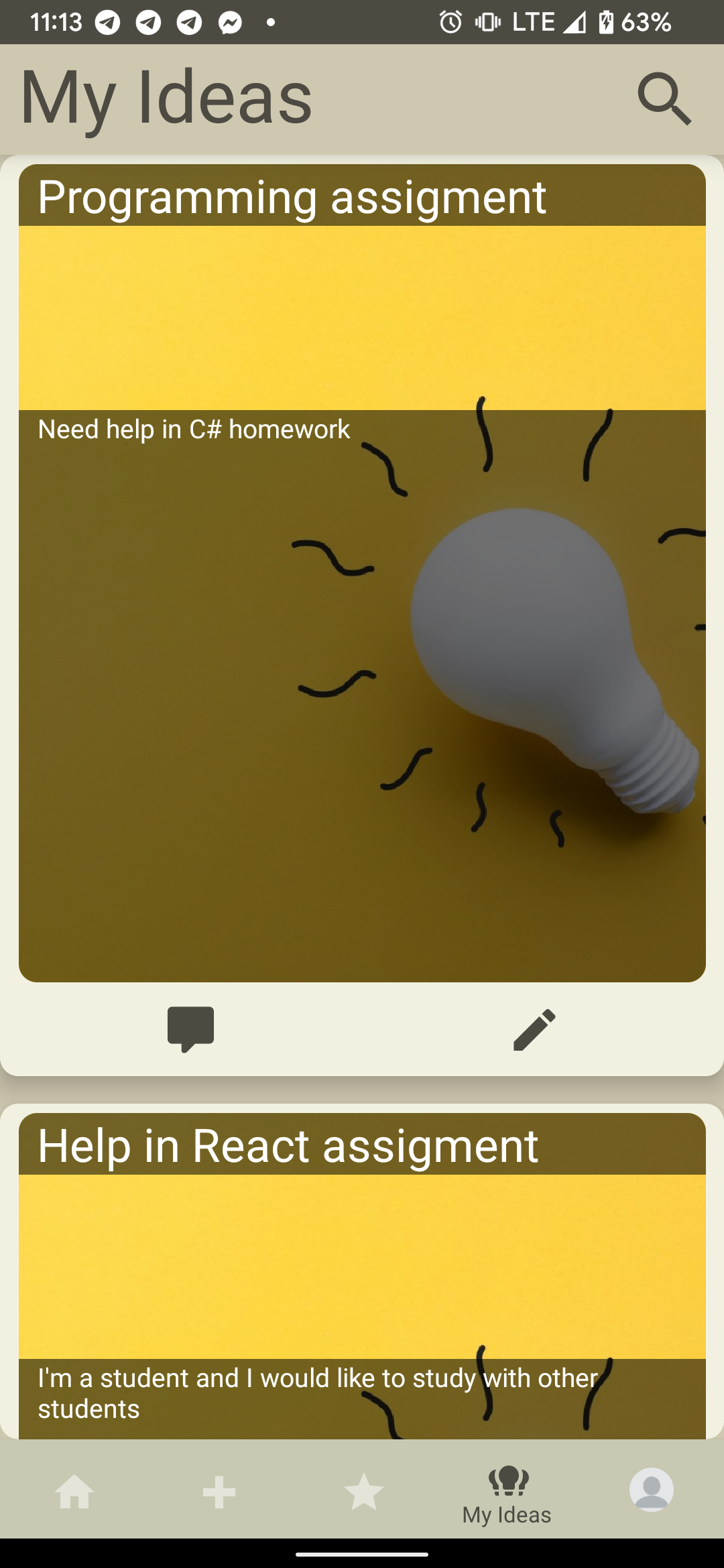
### 6.2.3 - A mobilos frontend bemutatása – New Idea

 Ezen a képen a felhasználó a saját ötletét, bejegyzését írhatja meg. Minden beviteli mező felett egy számláló mutatja azt, hogy mennyi karaktert tud a felhasználó még írni a szövegmezőbe. Ha a kép linkjéhez nem ír semmit, egy alapértelmezett képet fog feltölteni. Ha bármelyik másik mező üres marad, vagy ha a leírás megegyezik egy már meglévő poszttal, akkor a szerver hibát küld vissza ami egy Toast message-ben jelenik meg. Amennyiben sikeresen feltöltődött az új ötlet, Szintúgy Toast message-ben értesíti a felhasználót, majd megjelenik az ötlet a My Ideas képernyőn miután frissítette a listát egy lehúzással.

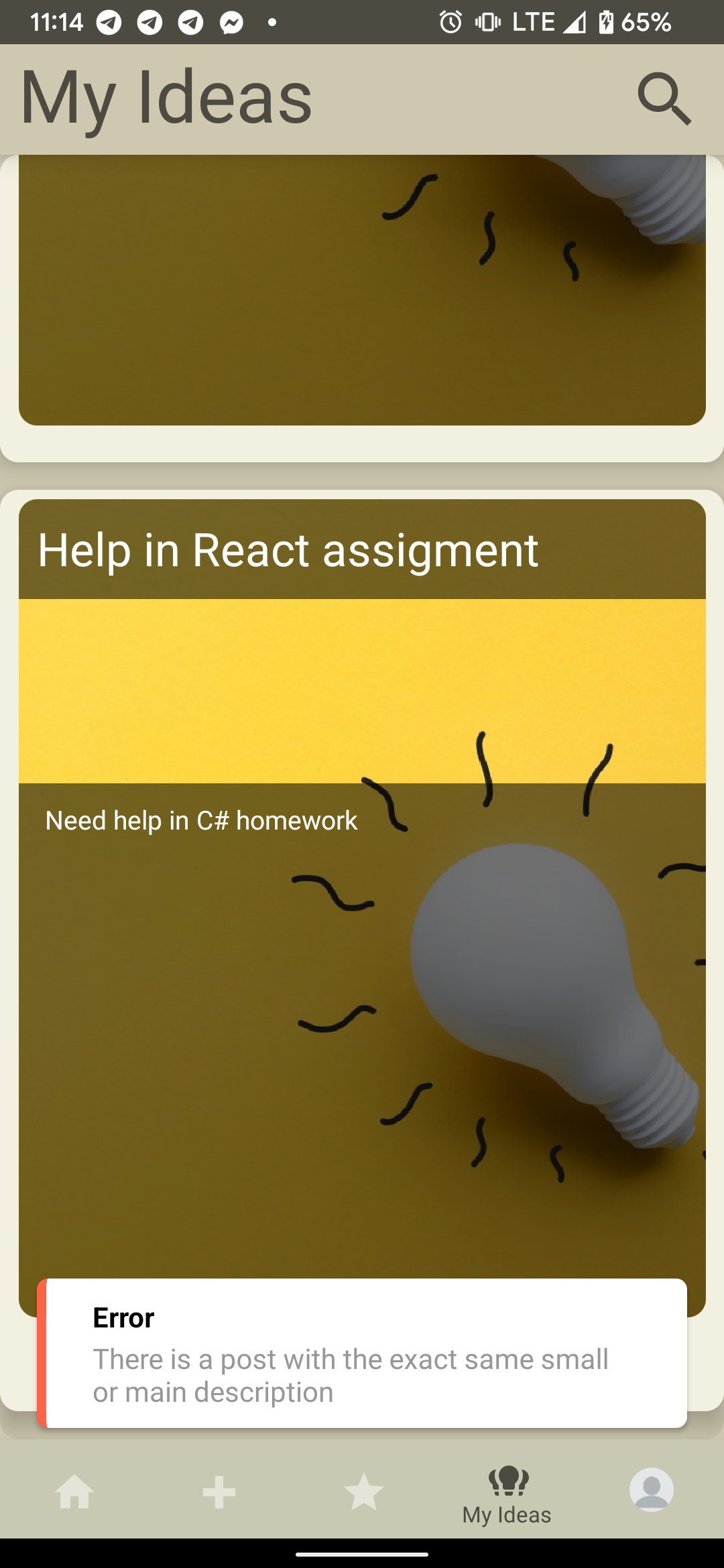
### 6.2.4 - A mobilos frontend bemutatása – My Ideas

Ezen a képernyőn a toll ikont megérintve átírhatóvá válik a cím és a leírás, így a felhasználó szerkesztheti a már megírt bejegyzését. A cím és a leírás közé nyomva itt is teljes képernyőssé növekszik, így szerkeszthető a hosszú leírás.



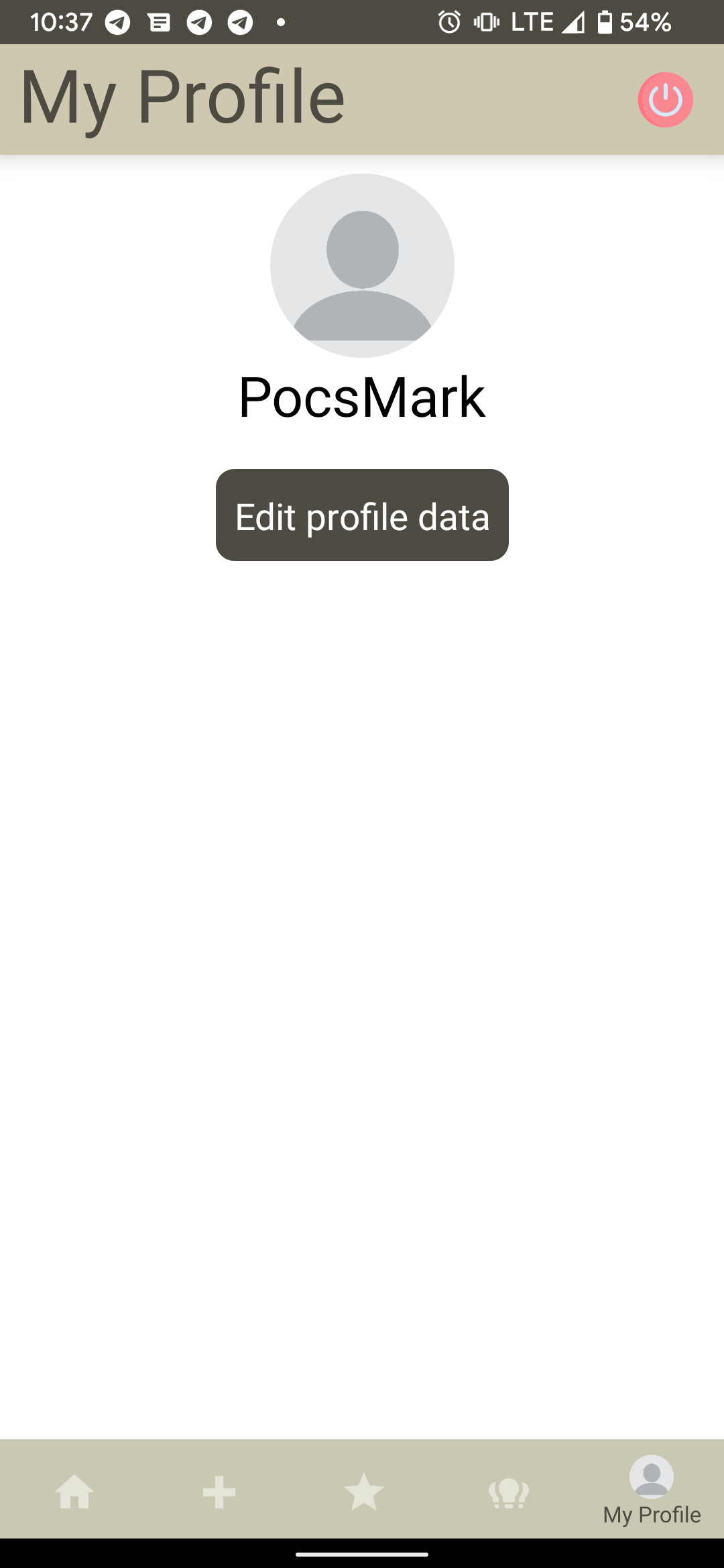
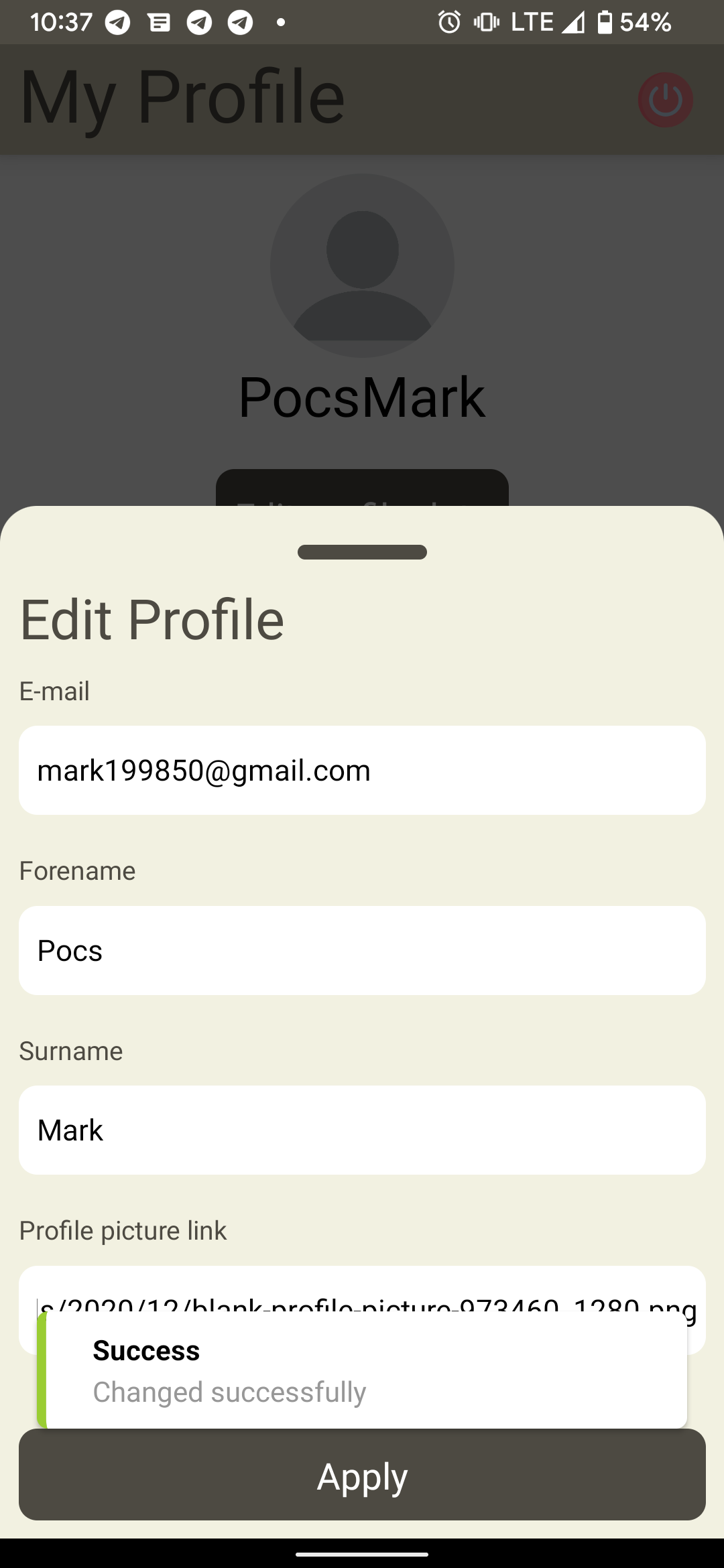


Az X gomb megnyomásával visszavonhatja a változtatásokat. Abban az esetben, ha a rövid vagy hosszú leírás pontosan megegyezik egy már meglévő ötlet leírásával, akkor a szerver hibát ad vissza azzal az üzenettel, hogy már létezik ilyen. Így valamelyest elkerülhetőek a duplikált posztok. Ha



### 6.2.5 - A mobilos frontend bemutatása – My Profile

Ezen a képernyőn módosíthatja az adatait a felhasználó



# 7.0 - Továbbfejlesztési lehetőségek

A továbbfejlesztést az ötletek megvalósítására jelentkező emberek szűrésében és megfelelő értékelésében látjuk. Egy értékelő rendszer kitalálásával próbálhatnánk meg a regisztrált felhasználók szűrését. Ez megnövelné mind az ötlet adók és megvalósítók körében az elégedettséget és a bizalmat, továbbá nagy százalékban növelné a sikeres üzletek számát. További szűréseket a regisztráció közben lehet beiktatni, de úgy gondoljuk, hogy ilyen mértékű szűrés sok potenciális felhasználót ijesztene el az oldalról.

Tovább, még gondolkodtunk az oldal pénzgeneráló lehetőségeiről is.

Az oldal pénzbevételt úgy generálna, hogy az ötletadó és az ötletet vagy segítséget végrehajtó ember(ek) megállapodnak a felületen egy összegben és ennek a megbeszélt összegnek egy bizonyos százalékát az oldal levonja. Minden egyes sikeres ”üzlet” után az oldal bevételt generálna.

# 8. - Források, hivatkozások

1. <https://en.wikipedia.org/wiki/Node.js> - 2.1 Node.js
2. <https://nodejs.org/en/about/> - 2.1 Node.js
3. <https://blog.templatetoaster.com/xampp-phpmyadmin/> 2.4 - phpMyAdmin
4. <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript> - 3.1 JavaScript
5. <https://www.w3schools.com/whatis/whatis_bootstrap.asp> - 3.3 bootstrap