Dokumentáció

Cross Platform Learning Environment

Dev team:

Bartalis Krisztián

Császár Ákos

Fodor Annamária

Horváth Áron

Sapientia Erdélyi Magyar Tudományegyetem

1. Bevezetés

Manapság egyre nagyobb teret hódított meg a digitális tér, egyre többen és egyre többet hagyatkozunk digitális felületekre.

Ezért mi kötelességünknek érezzük hogy, olyan minőségű szoftvert gyártsunk, amely kielégíti a felhasználók igényeit és megfelelő szolgáltatásokat tudnak juttatni.

Erre szolgál a “Cross Platform Learning Environment” projektünk ami egy sajátos, tudásmegosztó oldal, amely a tanulást egy teljesen újfajta köntösbe helyezi. A nyers és tömör tananyagok helyett, a felhasználók képesek lesznek többféle stílust és megközelítést tanulmányozni egy adott témában.

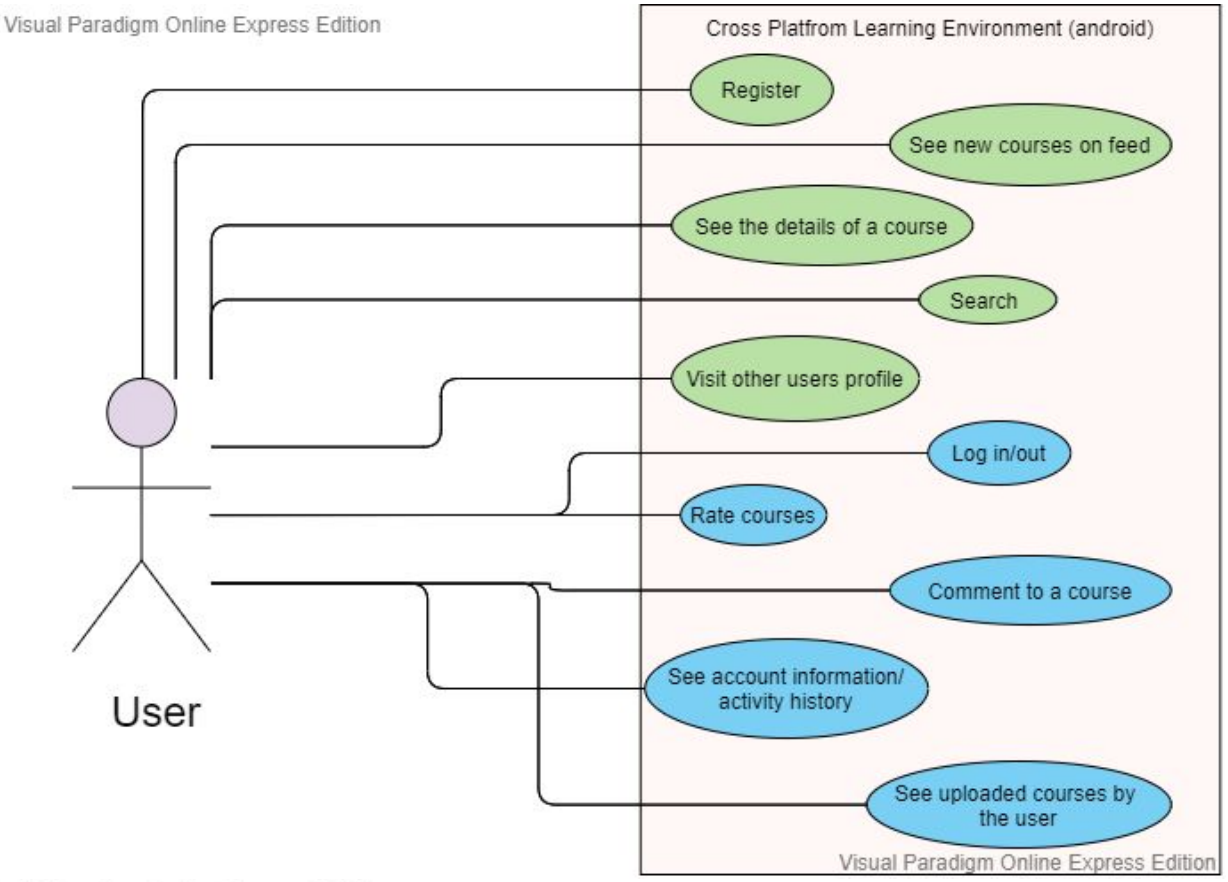
2. Projekt célja

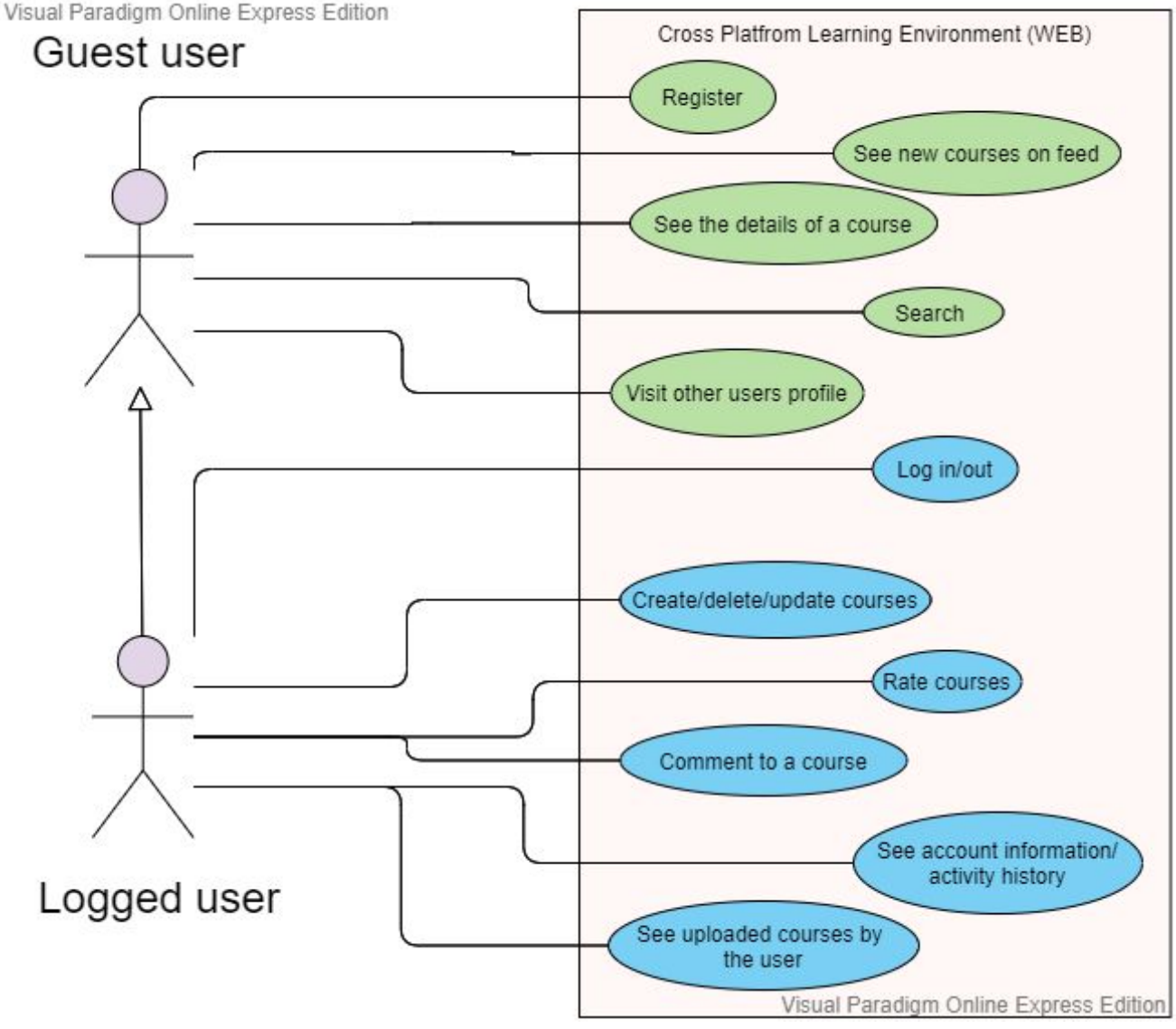
A projekt elsődleges célja hogy a tanulás folyamatát élvezetesebbé és könnyebbé alakítsuk és hogy az időt amelyet a digitális térben töltünk hasznosabb tevékenységre fordítsuk.

De ahhoz, hogy ez megvalósuljon, be kell csalogatni a felhasználókat. Ennek megvalósításának az érdekében, szeretnénk hogy a felhasználók ne kelljen fizessenek a termékért, és egy közösségi oldal élményét biztosítani, azáltal, hogy a cikkeket értékelni és kommentelni lehessen.

3.Software követelmények

3.2 A felhasználói követelmények


Visual Paradigm Online Express Edition
Cross Platfrom Learning Environment (android)
Register
See new courses on feed
See the details of a course
Search
Visit other users profile
Log in/out
Rate courses
Comment to a course
See account information/ activity history
See uploaded courses by the user
User
Visual Paradigm Online Express Edition


Visual Paradigm Online Express Edition
Guest user
Cross Platfrom Learning Environment (WEB)
Register
See new courses on feed
See the details of a course
Search
Visit other users profile
Log in/out
Create/delete/update courses
Rate courses
Comment to a course
See account information/ activity history
See uploaded courses by the user
Logged user
Visual Paradigm Online Express Edition

Use Case leírás

1. Kurzus kommentelés

Use Case	Comment to a Course
Pre-condition	• A user be kell legyen regisztrálva
Post-condition	• A komment bekerül az adatbázisba és megjelenik a kommentszekcióban
Basic path	1. A user regisztrál 2. Rákattint egy cikkre 3. Belemegy a kommentszekcióba 4. Az "Add comment" szövegdobozba új kommentet ír be 5. Elküldi a kommentet a küldés gombbal
Alternative path	• Az 1. es pontban a user-nek már van fiókja és csak be kell jelentkezzen
Exceptional path	• Ha a szövegdoboz üres akkor hibát dob ki

2. Kurzus böngészése

Use Case	See uploaded Courses
Pre-condition	• Nincs szükség különösebb feltételre
Post-condition	• A listázott kurzosok megjelennek a főképernyőn
Basic path	1. A user elindítja a platformot 2. Ha mobilról jelentkezik be szükséges fiókot készíteni. 3. A főmenü automatikusan megnyílik ahol böngészni tud a feltöltött kurzusok között
Alternative path	• Az 2. pontban abban az esetben ha Webes felületen lép be akkor nem szükséges fiók készítése
Exceptional path	• Ha az adatbázis offline a kurzusok nem érhetőek el

3. Kurzus részleteinek böngészése

Use Case	See the details of one Course
Pre-condition	• A platform elindítása (lehet Web vagy Android)
Post-condition	• Az adatbázis lekéri a kurzus adatait és a lekért kurzosok megjelenik a főképernyőn
Basic path	1. A user elindítja a platformot 2. Ha mobilról jelentkezik be szükséges fiókot készíteni. 3. A főmenü automatikusan megnyílik ahol böngészni tud a feltöltött kurzusok között 4. Egy kurzus megnyitása után rögtön megjelennek a lekért adatok
Alternative path	• A 3. pontban a főmenüből átlehet lépni a keresési opcióra és ott rákeresve meglehet nyitni egy keresett kurzust
Exceptional path	• Ha az adatbázis karbantartás alatt van • A keresett kurzus már nem szerepel az adatbázisba • A keresett kurzus nem található meg

4. Kurzus frissítése

Use Case	Update a Course
Pre-condition	• Webes felületen való bejelentkezés
Post-condition	• A kurzus tartalma frissül az adatbázisba és a képernyőn
Basic path	1. A user elindítja a platformot 2. A webes felületen fiókot hoz létre 3. A főmenü automatikusan megnyílik ahol böngészni tud a feltöltött kurzusok között 4. Egy saját kurzusra belépve az edit opciót megnyomva képes a felhasználó a kurzusát
Alternative path	• Az 2. pontban ha van fiók elégséges belépni, nem szükséges fiókot készíteni
Exceptional path	• Ha az adatbázis offline a kurzusok nem érhetőek el • Ha az update túllépi a megadott karakter keretet akkor hibát dob ki

3.3 Rendszer követelmények

a.) Funkcionális követelmények

Az alábbi fejezetben az alkalmazás főbb funkcionalitás közül ki szeretném emelni a legfontosabbakat:

- **Registration:** ez a gomb a regisztrációt valósítja meg.A gomb megnyomására az eseményfigyelő elemzi a formot. Hogyha a regisztrációs form helyes, vagyis ha a felhasználó minden adatot helyesen írt be akkor az adatbázisba létrejön egy új elem ami a most regisztrált felhasználót jelképezi. Egy egyéni ID-t is kigenerál ami alapján a későbbiekben tudjuk azonosítani.
- **Send:** új komment hozzáfűzése az adatbázishoz. Ha helyes a komment(vagyis ha a felhasználó a karakter limiten belül marad és nem üres a komment) akkor az aktuális cikkhez hozzá fűződik ez a hozzászólás.
- **Rate this article now:** Erre kattintva a felhasználótól bekért értékelés hozzáadódik az adatbázishoz amiből az adott cikk értékelése újraszámolódik. Ez az értékelés, a cikk részleteinél megtekinthető.

b.) Nem funkcionális követelmények

1. **Termék követelmények**

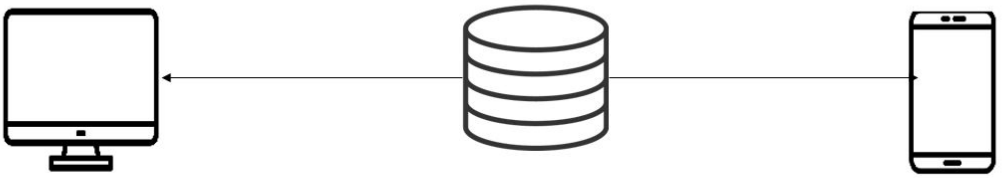
    - *Felhasználhatóság*: A termék Android OS-n (legalább 6.0.1 es rendszer, más néven Marshmallow) és bármely böngészőben futtatható
    - *Hatékonyság*: a termék megfelelően kiszolgál minden kérdést 500 ms alatt megfelelő internet sebesség mellett
    - *Megbízhatóság*: fejlesztés során a legtöbb hibalehetőség tesztelve volt, de nem zárjuk ki a potenciális összeomlás lehetőségét
    -

2. **Folyamat követelmények**
    - *Coding standard:*
        - a camelCase standard követése
        - egy sor egy definició
        - tördelés tabulátorral nem szóközzel
    - *Határidős követelmények*:
        - a verziókövető rendszer felállítása: 2020.10.16
        - feladatok kiosztása: 2020.10.17
        - a projekt struktúra kialakítása: 2020.10.23
        - adatbázis kapcsolat létesítése: 2020.10.30
        - a végső funkcionalitások implementálása: 2020.12.06
        -dokumentáció finalizálása: 2020.12.08
        -finalizálás és projekt beküldés: 2020.12.10

3. **Külső követelmények**
    - *Összeférhetőségi:* Az alkalmazás semmilyen más befolyással nem bír más alkalmazásokra tekintve, teljesen elkülöníthető a már meglévő alkalmazásoktól
    - *Etikai:* A termék semmilyen adatot nem szolgáltat ki külső forrásoknak, a lekért adatok a regisztrációnál, csakis a bejelentkeztetési rendszerhez szükséges és semmilyen megkülönböztetést nem alkalmazunk a felhasználók kezelésében.

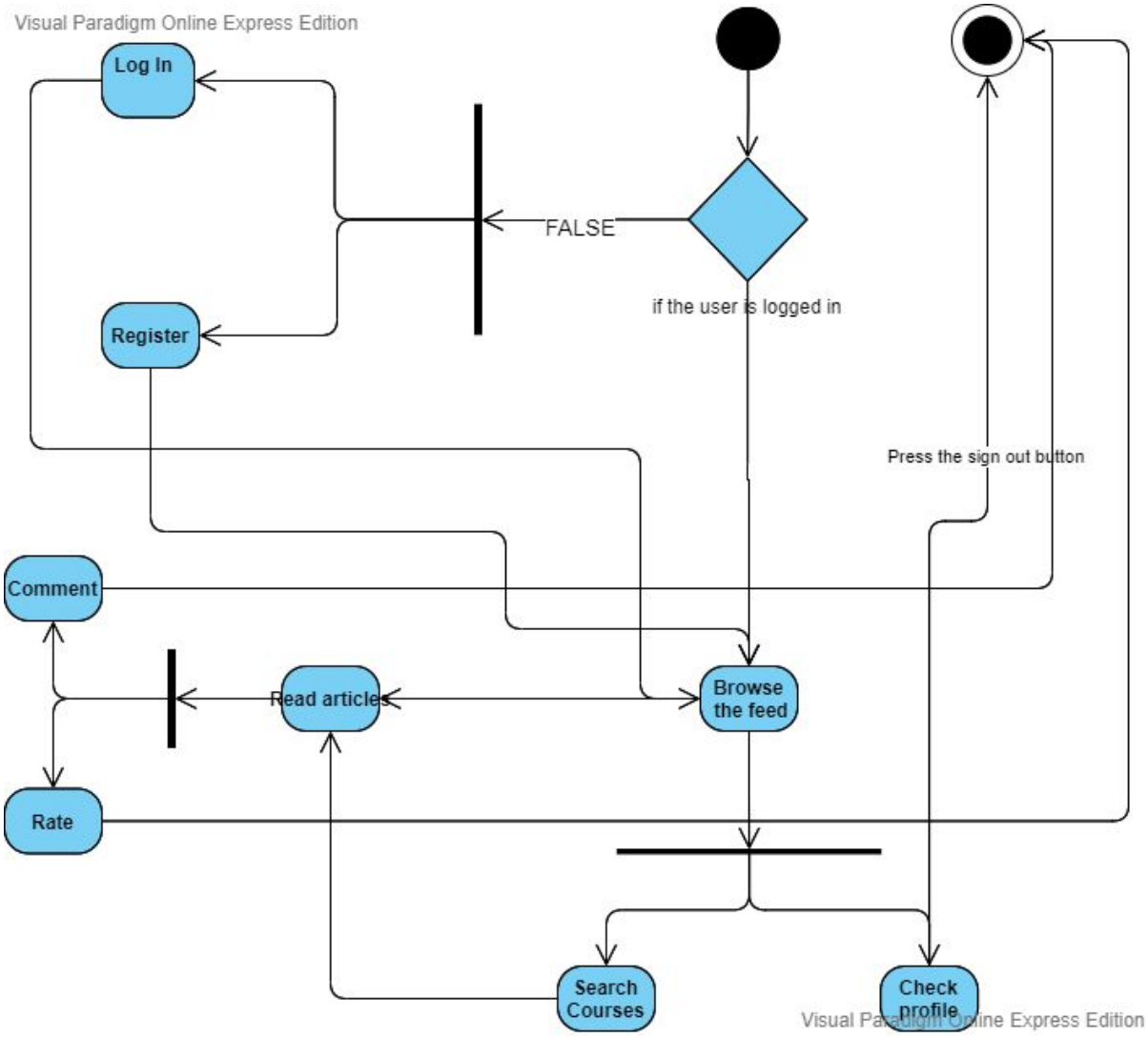
4. Tervezés

4.1. Architektúra felépítése



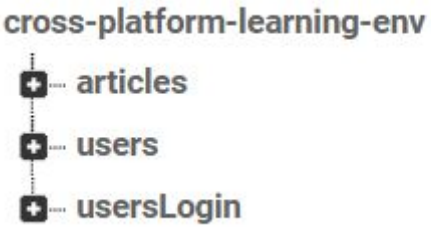
Mivel a projekt két felületből tevődik össze szükségessé vált egy közös adatbázis használata mely mindkét platformot egységesen szolgálja ki.

Activity diagram



4.2. Alkalmazás rövid áttekintése

a.) Az adatbázis szerkezete



Az általunk használt adatbázisséma 3 főbb komponensből tevődik össze:

- az articles referencia gyűjti össze az összes cikket ami az oldalra felkerül. Ez tartalmazza többek között, a cikk azonosítóját, az író azonosítóját, az értékelését, címét ,kommenteket és egy rövid leírást amit a főoldalon jelenítünk meg.
- A users referencia tartalmaz minden olyan információt amit a user végez vagyis a cikkeket amiket létrehozott és amit értékelt
- A usersLogin referencia tartalmaz minden olyan információt amit a regisztrációnál ad meg (azonosító,név, email és hashelt jelszó)

b.) UI terv

Mivel a platform Weben és Mobilon is elérhető lesz ezért szükségünk volt kitalálni közös UI elemeket amelyeket mindkét felületnél alkalmaztunk:

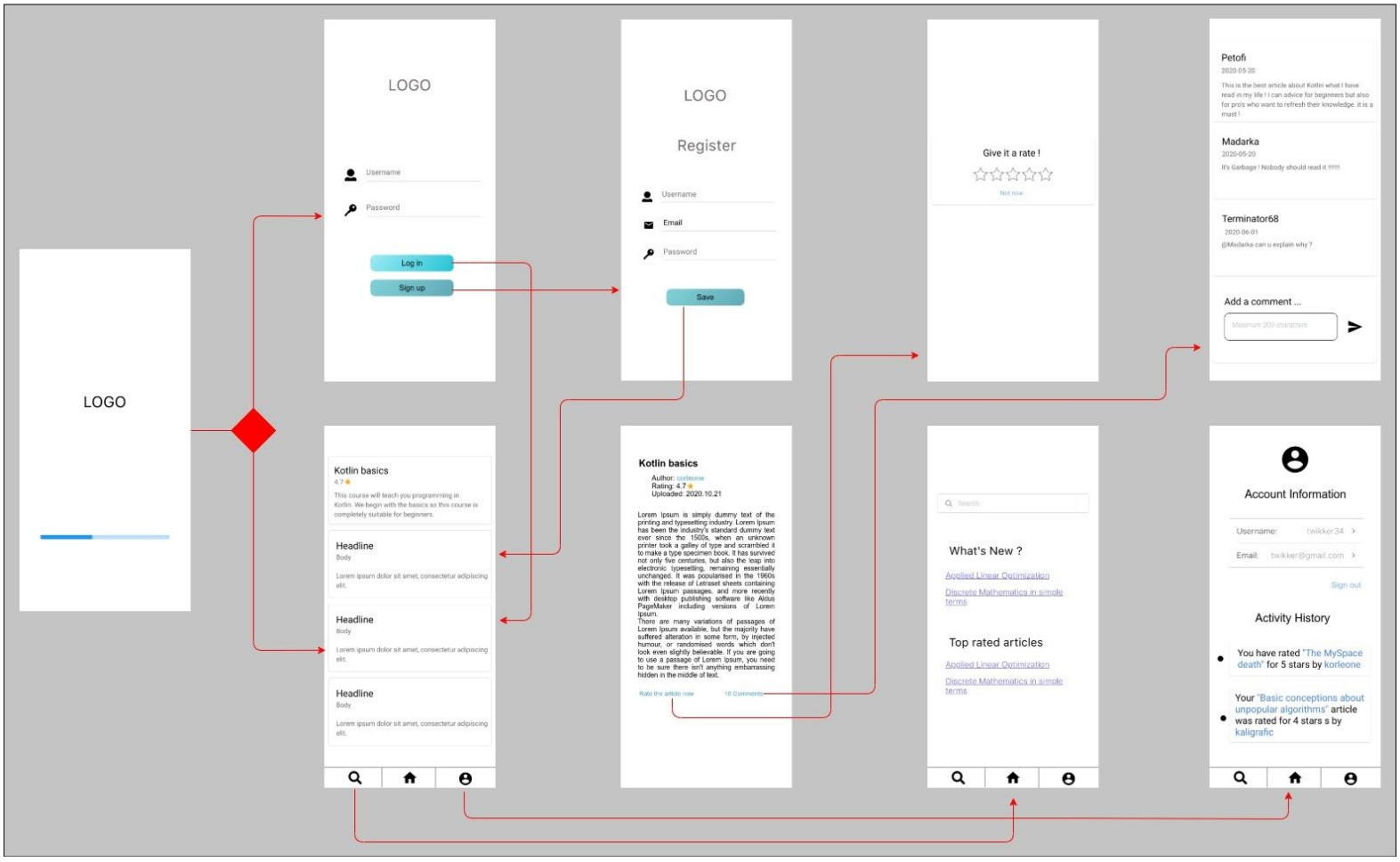
- Közös színpalettát használtunk mely a következő három:

  - Mindkét felület kezdőlapján tekinthetjük meg a meglévő cikkeket. A menüpontok felépítése mind a két esetben megegyezik: egy kereső fül és egy profil megtekintő rész

5. Modulok leírása

5.1 Mobilos felület

Vessünk egy pillantást az alkalmazás szerkezetére



A fenti diagram szemlélteti az alkalmazás elemi felépítését (design nélkül).

Az alkalmazás két fő részre különíthető el:

- **Bejelentkeztetés:** Ahogy fent is látszik, könnyen kivehető hogy a az alkalmazás kezdetekor egy splash screen hívódik meg. Ott döntjük el hogy az adott készüléken van-e bejelentkezve felhasználó (lásd 6.2.2 fejezet). Ha nincs akkor kell regisztráljon ha van akkor pedig a főoldalra visszük
- **Főmenü**: Itt van implementálva minden funkcionalitás, amely az alkalmazás fő céljait elégíti ki (cikk olvasás, kommentelés, értékelés, stb.)

5.2 Webes felület

A webes felület tartalmaz minden olyan funkciót amelyet az androidos applikáció tartalmaz, valamint ki is bővíti ezeket a dokumentumok szerkesztésével, azaz a felhasználó létrehozhat és törölhet dokumentumokat. Ezek a dokumentumok (köszönhetően a közös adatbázisnak) megtekinthetőek az androidon applikáción belül is.

5.3 Adatbázis

Az adatbázis meghatározásakor több szempontot figyelembe kellett vegyünk:

Az adatbázis típusát illetően szükséges volt meghatározni hogy noSQL vagy SQL típusú adatbázist használunk. Mivel elég sok adat kezdetben null értéket vesz fel (példának okáért a kommentek és az értékelések) ezért előnyösebbek találtuk a noSQL adatbázis típust, ezzel is sok tárhelyet spórolhatunk rajta.

Másfelől, számításba kellett vegyük, hogy az adatbázis folyamatosan ki tudja szolgálni a felhasználókat ezért felhő alapú szolgáltatást kellett keressünk.

A fenti szempontokat figyelembe véve és azt hogy ingyenes szolgáltatást használjunk, közös megegyezés alapján a Firebase adatbázist választottuk.

6. Kivitelezés

6.1 Verziókövetés

A projekt kivitelezése mögött egy négy fős csapat dolgozott ezért a *verziókövetés* egy elengedhetetlen része lett a fejlesztés ideje alatt.

Mint a legtöbb fejlesztő csapat mi is a GIT verziókövető rendszert használtunk s a kódot pedig a GitHub-on hostoltuk.

Mivel a projektünk nem csak egy forráskódból tevődött össze, hanem kettőből (**c#-javascript** és **Kotlin**) ésszerűnek láttuk ha egy organization-t hozunk létre és azon belül pedig 2 darab repository. Egyet a c#-javascript-nek és egyet a Kotlinnak. Igy könnyen elhatárolható volt a két projekt s párhuzamosan tudtunk haladni.

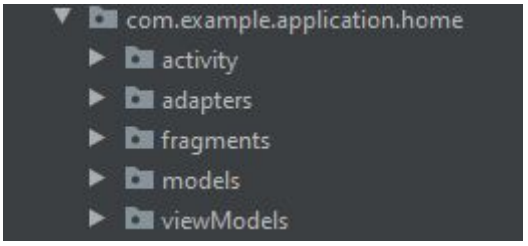
Mivel több funkcionalitáson dolgoztunk többen egyidőben ezért szükségesnek találtuk minden funkcionalitásnak külön branchet létrehozni, melyet a develop-ba egyesítettünk, ott kiküszöbölve az esetleges merge conflictokat.

Project management boardnak a Github integrált tábláját használtuk: A cetlik mozgatása pedig a következő elvet követte: meghatároztuk a to-do oszlopban, hogy melyek azok a funkcionalitások, vagy melyek azok a szükséges lépések hogy az alkalmazás tovább haladjon. Ezeket pedig hozzárendeltük a csapat egyik tagjához. Abban az esetben amikor valamelyik tag elkezdte az egyik lépést, az annak megfelelő cetlit átvitte az *“In Progress”* oszlopba és ha befejeződött akkor a *“Done”* oszlopba került. A *“Backlog”* oszlopba a szükséges háttéranyagok kerültek amiket a projekt során használtunk.

6.2 Android tervezési elvek

Fejlesztés során igyekeztünk a lehető legtöbb tervezési elvet betartani, azok közül eddig a következők valósultak meg:

- A fileokat funkcionalitásuknak megfelelően rendeztük el. Minden csomag egy funkcionalitásért felel. Például a *fragments* csomag a képernyők mögötti logikát tartalmazza) az *adapters* pedig a listákat köti össze az elemekkel.



- Figyelembe vettük az elnevezési konvenciókat ezért a package neveket kisbetűvel,míg az osztály neveket nagybetűvel kezdtük, valamint az osztályok metódusait szintén kis betűvel írtuk.

- Követtük a “több fragmens kevés activity” elvet ezér egyetlen activitybe beleilessztettük az összes fragmentet melyet egy navigációval kötöttünk össze
- “open-close “ elv : ha egy funkcionalitást megvalósítottunk a lehetséges kiegészítéseket külön metódusban, külön fileban folytattuk és nem írtuk felül a régit

6.2.1.Menürendszer

A menüt három fő szálra bontottuk. Mindhárom menüpontból különböző irányba tudunk elmenni:

- **home:** Innen tekinthető meg a feed ahonnan a kurzusok megtekinthetőek. Itt olvashatóak a kurzusok, a kurzusok kommentjei és itt értékelhetünk cikket
- **search:** Ez a funkció szolgálja a *“search and discovery”* funkcionalitást, itt lehet rákeresni bizonyos kurzusokra és lehet böngészni közöttük
- **profile:** a felhasználó itt láthatja az alapvető információit mint a felhasználónevet, emailt, és itt tud kijelentkezni az alkalmazásból.

6.2.2 Főbb implementálási ötletek

a.) Shared preferences

Ahhoz hogy megmondjuk, hogy az adott készüléken már volt-e felhasználó bejelentkezve szükségünk volt az Android által biztosított Shared Preferences csomagot használni

Néhány fontosabb komponens:

- **Létrehozás**:

```
sharedPref = requireActivity().
    getSharedPreferences( name: "credentials", Context.MODE_PRIVATE)
```

A “credentials” az a sharedPreferences azonosítója, ez alapján azonosítjuk az alkalmazás többi részében

- **Értékadás**: a Shared preferences értékeit egy editor segítségével lehet állítani, azon belül is a putstring metódussal. Ezek az értékek kulcs-érték párokat hoznak létre:

```
sharedPref =
    context?.getSharedPreferences( name: "credentials", Context.MODE_PRIVATE)!!
val editor = sharedPref.edit()
editor.clear()
editor.putString("email", user.email)
editor.putString("password", passwordHash)
editor.putString("userId", userID.toString())
editor.putString("username", userName)
editor.apply()
```

- **Érték lekérés:** a shared preferences értékeit a getString metódussal tudjuk lekérni:

```
sharedPref.getString( key: "username", defValue: "").toString())
```

b.) Adatbázissal való kapcsolat kialakítása

- **Írás:** A Firebase szolgáltat egy ún. Real-time Database terméket amelybe könnyedén lehet írni adatokat a következőképpen:

```
myRef.child( pathString: "users").child(user.userID.toString()).child( pathString: "userId").setValue(user.userID.toString())
myRef.child( pathString: "users").child(user.userID.toString()).child( pathString: "username").setValue(user.userName)
```

Ahol a **myRef** egy referencia az adatbázisra. Mivel az adatbázis többszintű szükséges használni a **child** függvényt ami az alatta levő referenciát téríti vissza

- **Olvasás:** A Firebase adatbázisban levő adatok lekéréshez szükséges a listener használata:

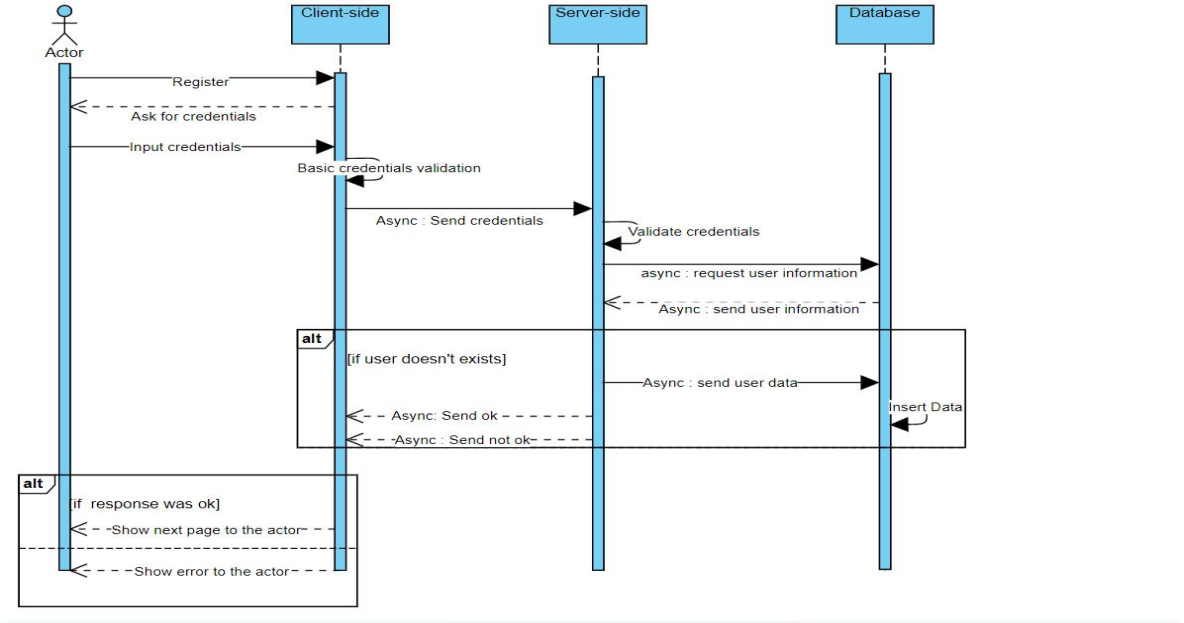
```
usersRef.addValueEventListener(object : ValueEventListener {
    override fun onDataChange(dataSnapshot: DataSnapshot) {
        for (data in dataSnapshot.children) {
            // if the credentials are correct
            if (data.key.toString() == userID.toString()) {
                generateID()
            }
        }
    }

    override fun onCancelled(databaseError: DatabaseError) {
        println("The read failed: " + databaseError.code)
    }
})
```

A fenti példában az egyedi ID generálásának az algoritmusa látható.

6.3 Tervezési elvek - Web

A weboldal technikai felépítése nagyban eltér az androidtól.



Ahogy a fenti, regisztrációt ábrázoló szekvencia diagram is mutatja, a kliens oldal nem direkt módon kommunikál az adatbázissal, hanem van egy külön szerver

oldalunk ahol minden beérkező és kimenő adatot leellenőrzünk és ennek függvényében végezzük el az adatbázis műveleteket, jelenítjük meg a felhasználó számára az oldalakat.

Az alábbi ábrán látható, hogy a kliens oldalhoz React Js-t és Node Js-t, a szerver oldalhoz .Net-et és C#-ot, valamint az adatbázishoz a Firebase fejlesztői platform Realtime Database részét használtuk.

A kliens a szerver jól meghatározott, publikus endpointjait bármikor elérheti regisztráció vagy bejelentkezés nélkül. Ezek az endpointok segítségével lekérheti, megtekintheti a különböző dokumentumokat a felhasználó, viszont nem hozhat létre, nem törölhet és nem módosíthat.

Ezek a műveletek csak ellenőrzött endpointokkal lehetségesek, melyekhez regisztrációkor vagy bejelentkezéskor a szerver által generált és kiküldött egyedi token szükséges.

```
[Authorize]
[EnableCors]
[HttpPost("RateArticle")]
0 references
public ActionResult<string> RateArticle(object obj)
{
```

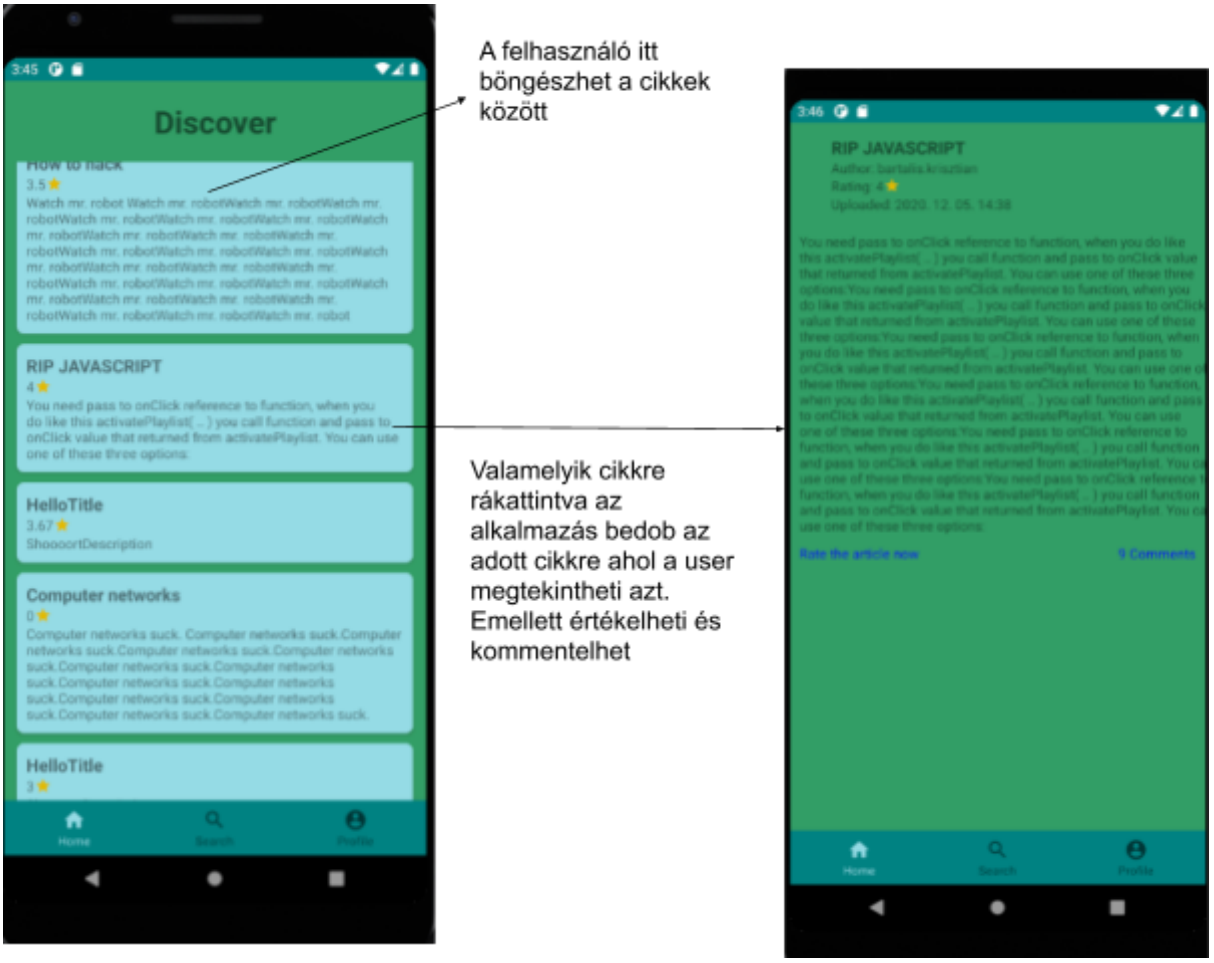
Ezek a tokenek kliens oldalon sütiben (cookie) tárolódnak és 1 napig érvényesek. A lejártuk után a kliens nem éri el az ellenőrzött endpointokat és újra végre kell hajtsa a bejelentkezés/regisztrálás műveletet egy új token igényléséhez.

7.Használati útmutató

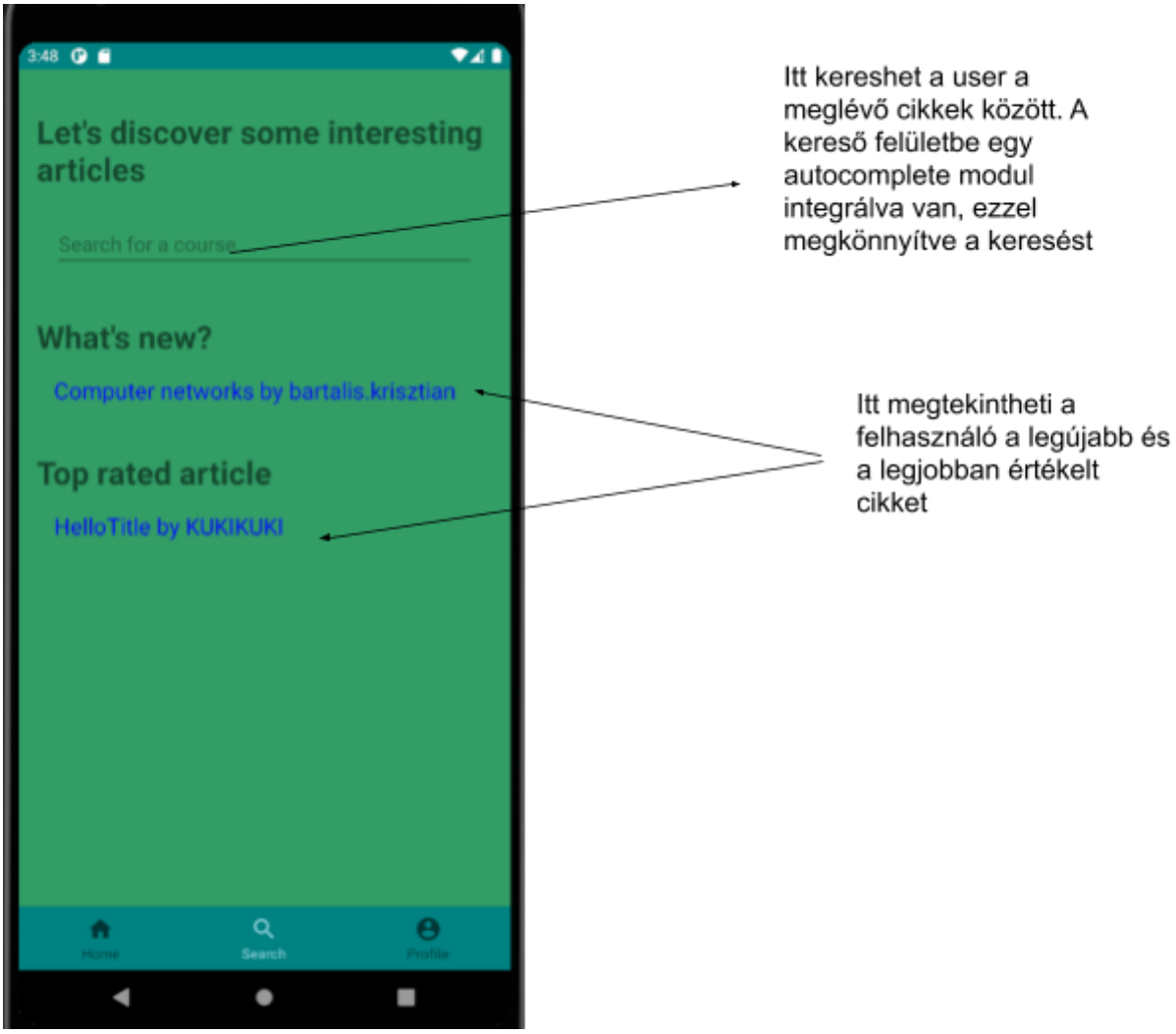
7.1 Android

a.) Bejelentkezés

b.) Főoldal böngészése

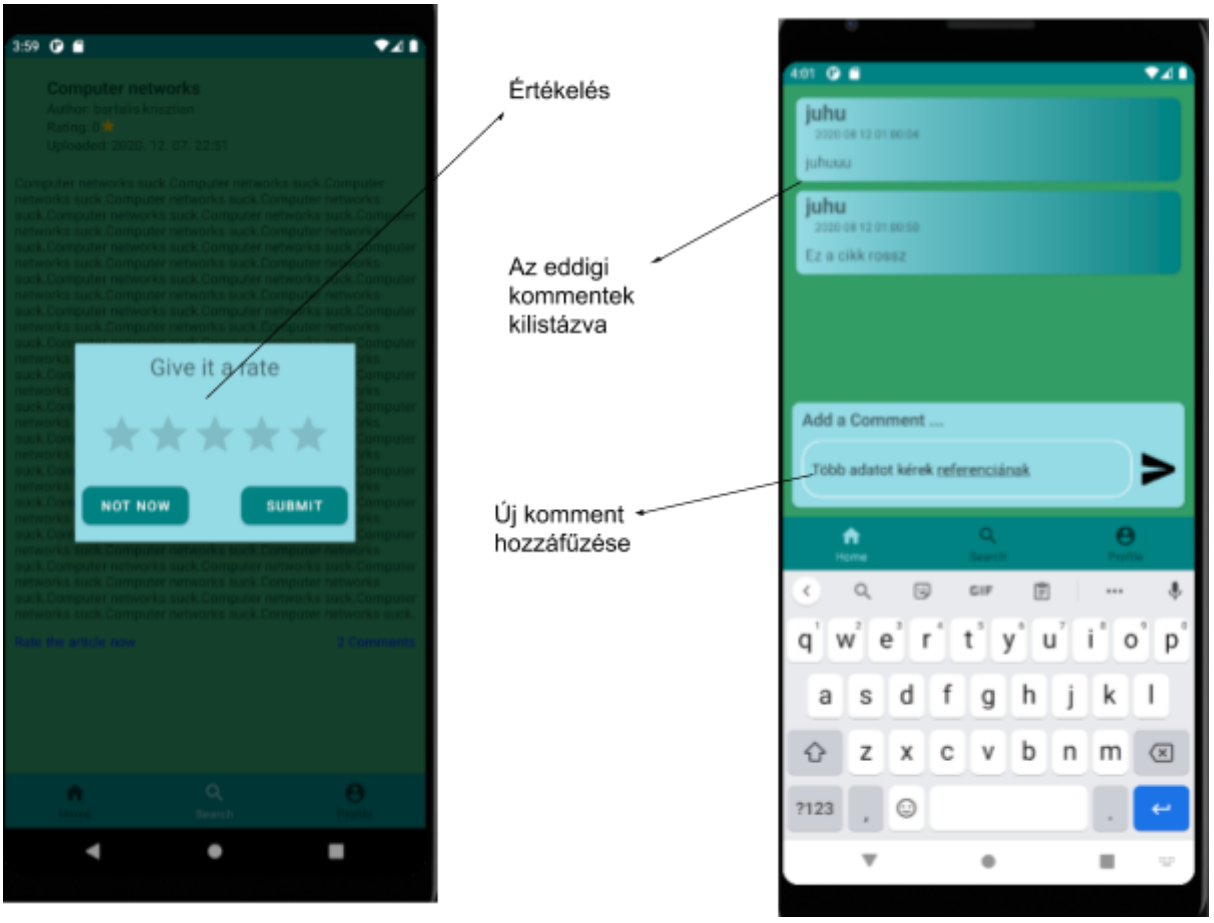


c.) Cikk keresése



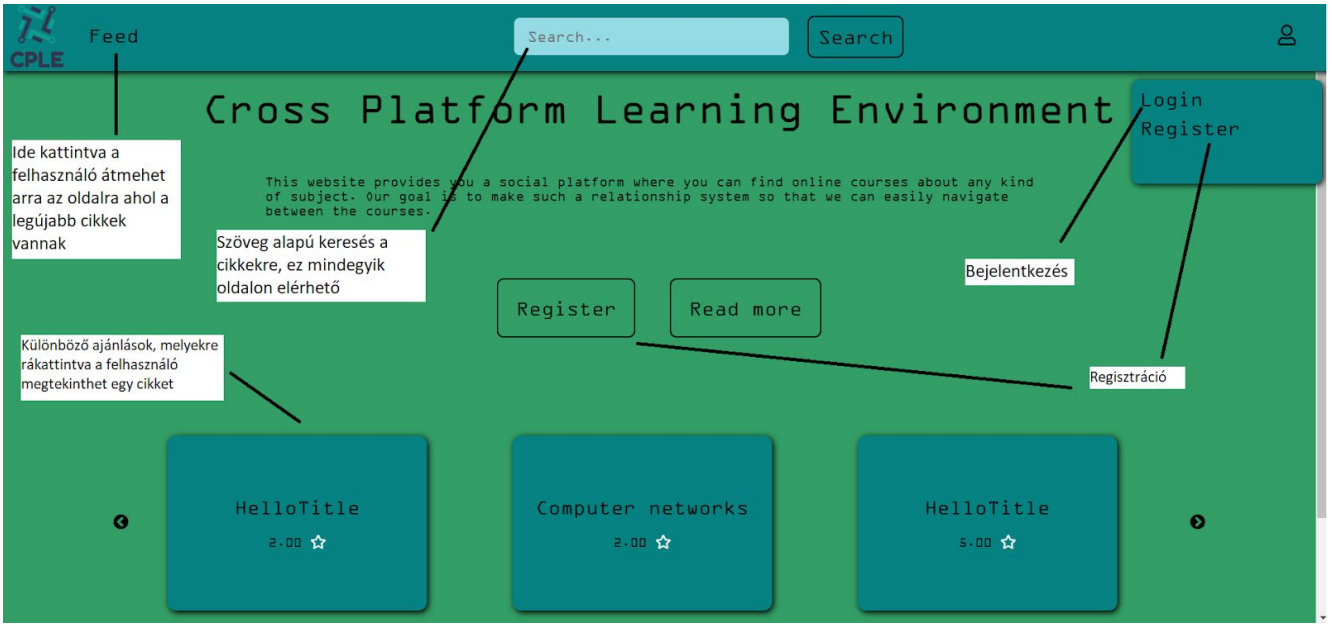
d.) Profil megtekintése

e.) Mellékfunkciók

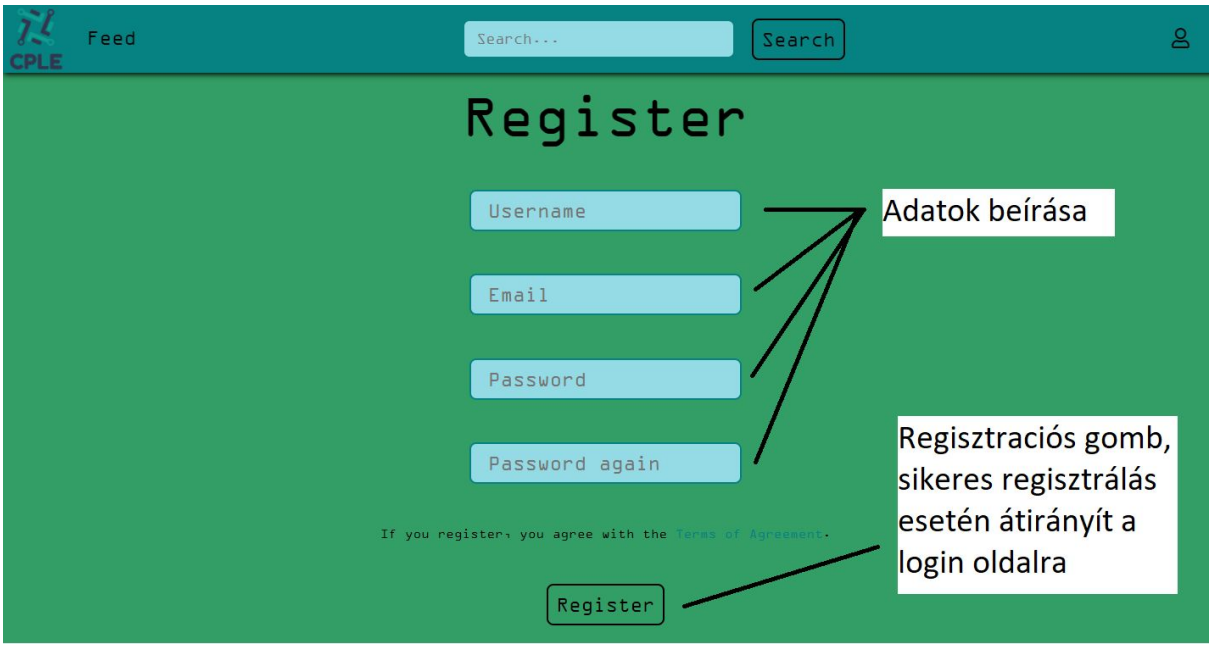


7.2 Web

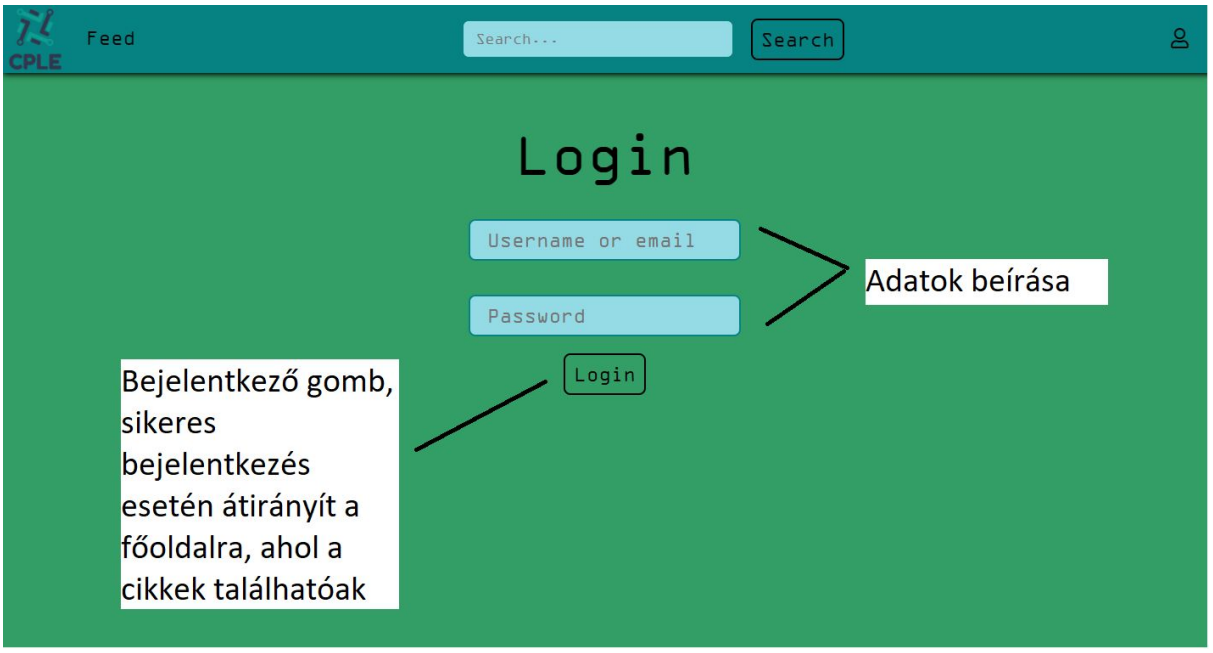
a.) “Üdvözlő” oldal



b.) Regisztráció



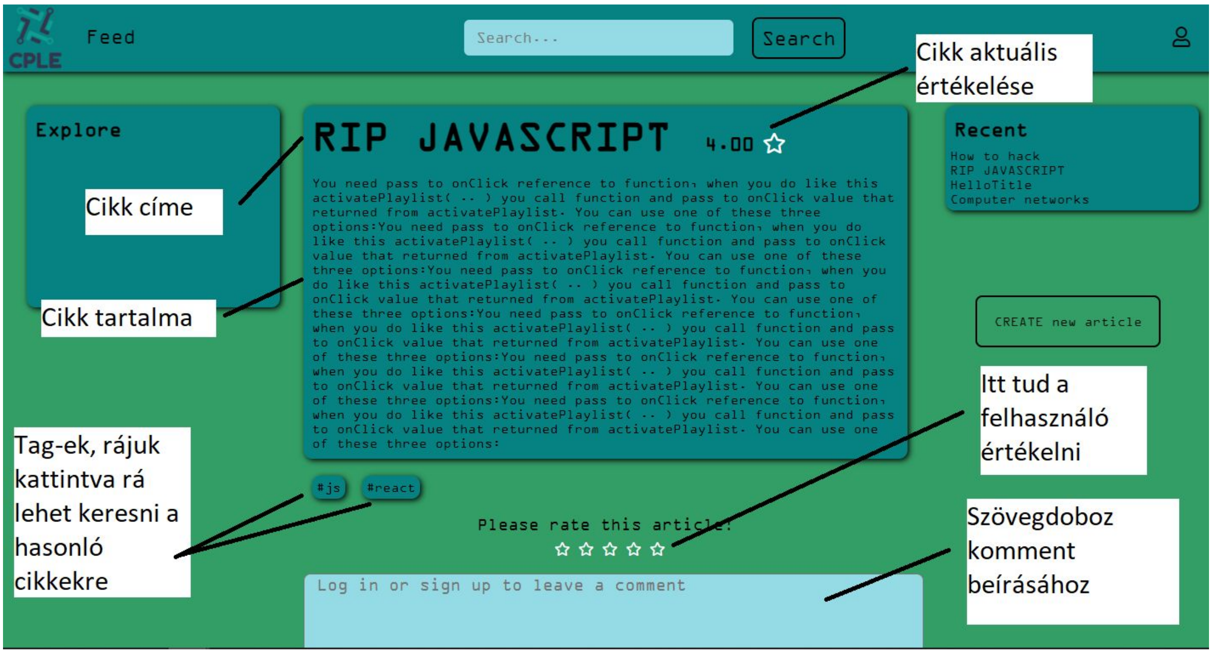
c.) Bejelentkezés

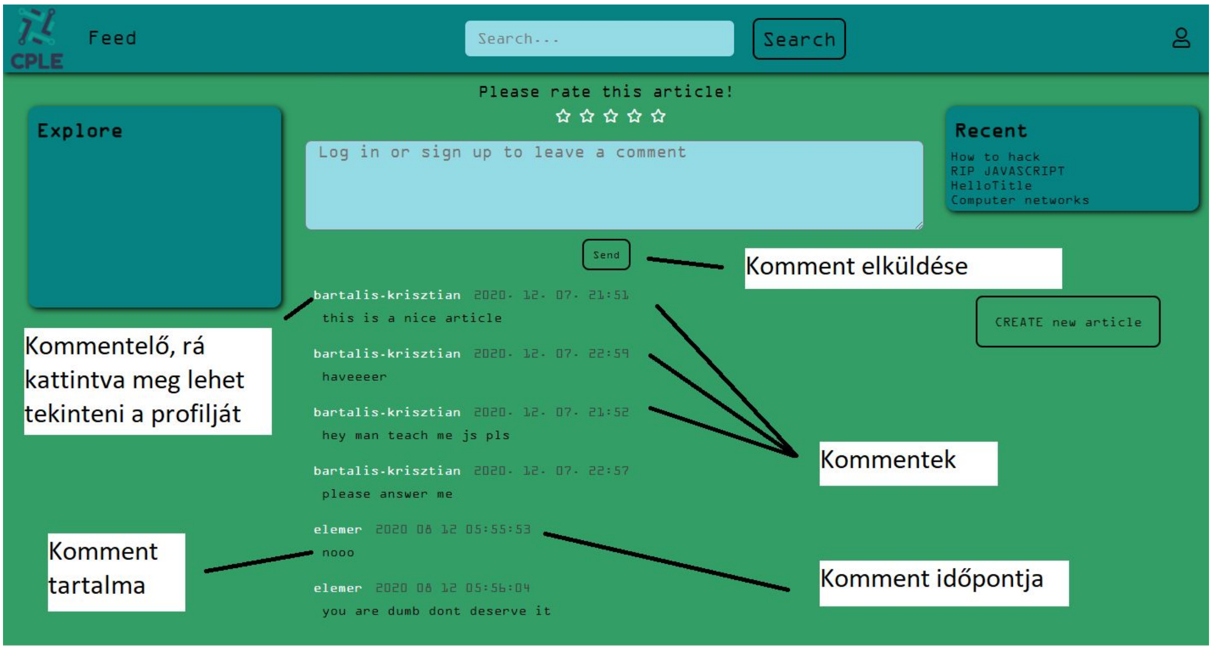


d.) Főoldal

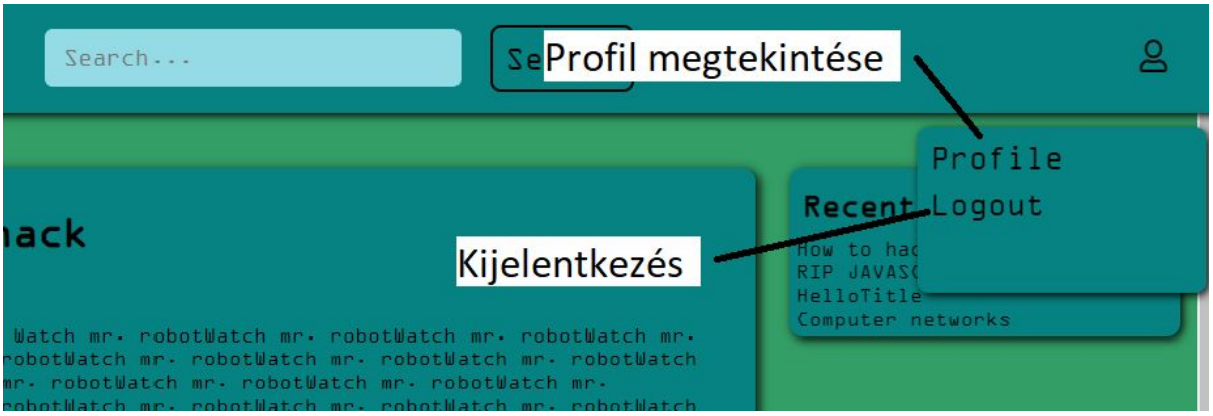
e.) Cikk létrehozása

f.) Egy cikk részletes megtekintése





g.) Profil megtekintése



8. Összegzés

Az elmúlt másfél hónapot kiértékelve kijelenthetjük, hogy sikeresen létrehoztunk egy működő platformot amit a felhasználó nyugodtan tud használni. A főbb funkcionalitásokat implementáltuk mint a regisztrációt, bejelentkezést, kurzus hozzáadást, frissítést,kurzus olvasást, kommentelést,értékelést.

Tisztában vagyunk a ténnyel hogy ezek a funkcionalitások még kezdetlegesek. A továbbiakban tervben van egy közösségi funkció integrálása mint a chat, közös érdeklődésű csoportok létrehozása, osztálytermek, kvízek implementálása.

A termék létrehozása során a tervezésre sok hangsúlyt fektettünk és ezért sikerült komoly alapzatra helyezni. Erre az alapzatra a későbbiekben az építkezés sokkal folyékonyabban fog menni, ezáltal egyre több funkcionalitást leszünk képesek integrálni, rövidebb időn belül.

A “Cross Platform Learning Environment” már ebben az állapotban is képes a tanulást élvezetessé és könnyebbé tenni ingyenesen és ezért úgy gondoljuk hogy a kezdeti céljainkat teljesítettük így a platform jövőjéről már nagyobb távlatokba is gondolkodhatunk.