



Eötvös Loránd Tudományegyetem

Informatikai Kar

Informatikatudományi Intézet

Média- és Oktatásinformatika Tanszék

Interaktív webes kvízalkalmazás- QuizMaster

Szerző:

Farkas Bálint

Programtervező informatikus BSc.

Belső témavezető:

Dr. Illés Zoltán Gábor

habilitált egyetemi docens

Külső témavezető:

Nagy András

Principal ServiceNow Consultant

Budapest, 2024

EÖTVÖS LORÁND TUDOMÁNYEGYETEM
INFORMATIKAI KAR

SZAKDOLGOZAT TÉMABEJELENTŐ

Hallgató adatai:

Név: Farkas Bálint
Neptun kód: F8R6L8

Képzési adatok:

Szak: programtervező informatikus, alapképzés (BA/BSc/BProf)
Tagozat : Nappali
Külső témavezetővel rendelkezem

Külső témavezető neve: Nagy András

munkahelyének neve: GuideVision Magyarország Kft.
munkahelyének címe: 1119 Budapest, Fehérvári út 79.
beosztás és iskolai végzettsége: Principal ServiceNow Consultant + Mérnök informatikus BSc
e-mail címe: andras.nagy@guidevision.hu

Belső konzulens neve: Illés Zoltán Dr.

munkahelyének neve, tanszéke: ELTE-IK, Média- és Oktatásinformatika Tanszék
munkahelyének címe: 1117, Budapest, Pázmány Péter sétány 1/C.
beosztás és iskolai végzettsége: habilitált egyetemi docens

A szakdolgozat címe: Interaktív webes kvízalkalmazás

A szakdolgozat témája:

(A témavezetővel konzultálva adja meg 1/2 - 1 oldal terjedelemben szakdolgozat témájának leírását)

A szakdolgozatom témája egy full-stack webes kvízalkalmazás elkészítése, mely két különböző jogosultsági szintet különböztet meg: egy adminisztrátori és egy alap felhasználói szintet. Az adminisztrátor kezeli az alkalmazás minden aspektusát, beleértve új kvíztemák létrehozását és a meglévő tartalmak szerkesztését vagy törlését. Az alap felhasználók a regisztráció és bejelentkezés után tölthetik ki a kvizeket, melyek különböző témakörökben érhetők el, és minden felhasználó számára csak egyszer kitölthetők.

Az alkalmazás egy integrált időzítővel méri, mennyi idő alatt töltik ki a felhasználók a kvizeket, és nyilvántartja a helyes válaszok számát. Ezen adatok alapján egy ranglistát állít össze, melyen a legjobb eredmények tekinthetők meg, azaz ki mennyi idő alatt és hány helyes válasszal töltötte ki a kvízt.

A felhasználók lehetőséget kapnak arra is, hogy új kvíztemákat javasoljanak, melyeket az adminisztrátorok elfogadhatnak vagy elutasíthatnak a többi felhasználó által adott visszajelzések alapján. Ezáltal a felhasználók közvetlenül is részt vehetnek az alkalmazás tartalmának alakításában. Az adminisztrátori felület lehetővé teszi a kvizek átfogó kezelését, beleértve az új kérdések hozzáadását és a meglévő adatok módosítását.

Budapest, 2024. 05. 06.

Tartalomjegyzék

1.	Bevezetés	1
2.	Felhasználói dokumentáció.....	2
2.1.	A QuizMaster-ről röviden	2
2.2.	Az alkalmazás futtatása	2
2.3.	Az alkalmazás használata	4
2.3.1.	Szerepkörök és jogosultságaik	4
2.3.2.	Navigációs sáv	5
2.3.3.	Kezdőlap	6
2.3.4.	Bejelentkező oldal	7
2.3.5.	Regisztrációs oldal	9
2.3.6.	Összes Kvíz oldal.....	12
2.3.7.	Kvízkérelmek oldal.....	13
2.3.8.	Kvízkérelem leadása oldal	15
2.3.9.	Eredmények oldal.....	16
2.3.10.	Kvíz kitöltése oldal.....	17
2.3.11.	Előző válaszok megtekintése oldal.....	17
2.3.12.	Új Kvíz létrehozása oldal.....	18
2.3.13.	Kvíz szerkesztése oldal	20
2.3.14.	Az oldal nem található – 404.....	21
3.	Fejlesztői dokumentáció	22
3.1.	Az alkalmazás sematikus felépítése	22
3.2.	Elemzés.....	23
3.2.1.	Feladat leírása	23
3.2.2.	Funkcionális leírás	24
3.2.3.	Nem funkcionális követelmények	32

3.3.	Tervezés.....	33
3.3.1.	Nézet	33
3.3.2.	Felhasznált technológiák	41
3.4.	Megvalósítás.....	45
3.4.1.	Alkalmazás felépítése	45
3.4.2.	Modell és kapcsolatok.....	46
3.4.3.	Adatbázis	79
3.5.	Tesztelés	80
3.5.1.	Szerveroldali tesztelés	80
3.5.2.	Kliensoldali tesztelés	84
4.	Összefoglalás	91
5.	Továbbfejlesztési lehetőségek	92
6.	Felhasznált források	93

1. Bevezetés

A szakdolgozatomban egy interaktív, webes kvízapplikáció fejlesztésének folyamatát szeretném bemutatni, mert fontosnak tartom a játékos tanulás élményét, amely egyszerre szórakoztató és tudásbővítő. Az applikáció célja, hogy hidat képezzen a tanulás és a játék között, ösztönözve a felhasználókat arra, hogy élvezettel mélyítsék ismereteiket.

Az alkalmazás két jogosultsági szintet kezel: adminisztrátori és alap felhasználói szintet. Az adminisztrátorok számára lehetőség nyílik az alkalmazás tartalmainak kezelésére, például új kvizek létrehozására és meglévők szerkesztésére vagy törlésére, míg az alap felhasználók, regisztráció és bejelentkezés után, különböző témakörökben érhető kvizek kitöltésével mérhetik fel tudásukat.

Kvíz kitöltésekor az applikáció egyik fontos eleme a beépített időzítő, amely rögzíti a kitöltési időt és pontosságot, így ranglista készül a legjobb eredményekről, lehetővé téve a felhasználók számára teljesítményük összehasonlítását. Emellett a rendszer lehetőséget biztosít új kvíztémák javaslatára, amelyeket az adminisztrátorok elbírálnak, így a felhasználók közvetlenül hozzájárulhatnak az alkalmazás tartalmi fejlődéséhez.

A felhasználói dokumentáció célja, hogy áttekinthető módon bemutassa az alkalmazás funkcionalitását és annak használatát, lehetővé téve az alapvető és adminisztrátori szintű felhasználók számára a különböző funkciók hatékony kezelését. Az útmutató tartalmazza a kvizek kitöltésének és a kérelmek feladásának lépéseit, a ranglista működését, valamint a felhasználói élményhez kapcsolódó navigációs és interakciós lehetőségeket.

A fejlesztői dokumentáció részletesen leírja az alkalmazás architektúráját, technológiai hátterét és a kódstruktúrát. Fő célja, hogy segítséget nyújtson a fejlesztőknek a kód értelmezésében és az alkalmazás belső működésének megértésében, beleértve a kvízkérdések, kvízkérelmek és ranglista kezelését, az időzítő beállításait.

Célom, hogy egy átfogó, jól strukturált dokumentációval segítsem mind az applikáció használatát, mind az esetleges jövőbeni karbantartást és fejlesztéseket.

2. Felhasználói dokumentáció

2.1. A QuizMaster-ről röviden

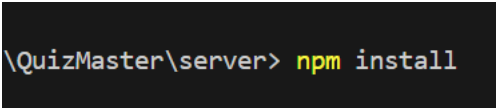
A QuizMaster egy interaktív, online kvízplatform, amely lehetőséget kínál a felhasználóknak, hogy különböző témakörökben teszteljék tudásukat. Az alkalmazás két különálló felhasználói szerepkört támogat: a felhasználókat, akik regisztráció után tölthetik ki a kvízeket, és az adminisztrátorokat, akik a kérdések és kvízek kezeléséért felelősek. A QuizMaster minden kvízkitöltés során időmérővel rögzíti az időeredményeket és nyilvántartja a helyes válaszok számát. Ezen adatok alapján ranglistát hoz létre, amelyen a legjobb eredmények tekinthetők meg. Az alkalmazás lehetőséget biztosít a felhasználóknak, hogy új kvíztémákat javasoljanak, melyeket az adminok visszajelzések alapján fogadhatnak el, így a QuizMaster rugalmasan bővülő, közösségi alapú tartalommal szolgál.

2.2. Az alkalmazás futtatása

Az alkalmazás futtatása fejlesztői környezetben (pl. Visual Studio Code) történik. Az alábbi lépések bemutatják a szerver és a kliens indítását és leállítását. A futtatáshoz szükség van a **Node.js** egy viszonylag friss verziójára (<= 18.20.0), amely letölthető a következő oldalon: <https://nodejs.org/en/download>.


- **Alkalmazás indítása:**

- Nyissunk egy terminált, navigáljunk a szerver mappájába, majd adjuk ki az npm install parancsot a szükséges csomagok telepítéséhez, ezt követően futtassuk az npm run dev parancsot a szerver indításához.



```
\QuizMaster\server> npm install
```

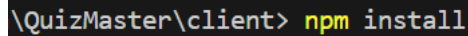
1. ábra: Szükséges csomagok telepítése a szerver mappájában



```
\QuizMaster\server> npm run dev
```

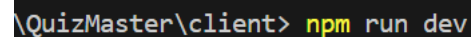
2. ábra: Szerver elindítása

- Nyissunk egy másik terminált, navigáljunk a kliens (client) mappájába, és ugyanúgy futtassuk az npm install parancsot a csomagok telepítésére, majd az npm run dev parancsot a kliens indításához.



```
\QuizMaster\client> npm install
```

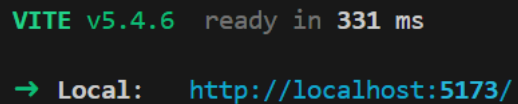
3. ábra: Szükséges csomagok telepítése a kliens mappájában



```
\QuizMaster\client> npm run dev
```

4. ábra: Kliens elindítása

- A kliens elindítása után a Vite automatikusan jelzi, hogy az alkalmazás elérhető a böngészőből a következő címen: <http://localhost:5173/>.

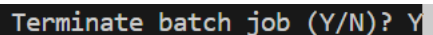


```
VITE v5.4.6 ready in 331 ms  
→ Local: http://localhost:5173/
```

5. ábra: Alkalmazás kliens címe a böngészőben

- **Alkalmazás leállítása:**

- A szerver és a kliens terminálablakában használjuk a Ctrl+C billentyűkombinációt az alkalmazás leállításához. Ha a fejlesztői környezet visszakérdez, erősítsük meg a leállítás szándékát az Y billentyű lenyomásával.



```
Terminate batch job (Y/N)? Y
```

6. ábra: Terminálás megerősítése leállításkor (szerver és kliens)

2.3. Az alkalmazás használata

2.3.1. Szerepkörök és jogosultságai

Az alkalmazásban létezik alapfelhasználó és adminisztrátor felhasználó szerepkör. Az alkalmazásban az oldalak és jogosultságok eszerint a két szerepkör szerint érhetőek el.

- **Be nem jelentkezett felhasználók jogosultságai:**

- Kezdő oldal
- Regisztrációs oldal
- Bejelentkező oldal
- Összes Kvíz oldal
- Eredmények oldal

- **Bejelentkezett alapfelhasználó jogosultságai:**

- Kvíz kitöltése oldal
- Előző válaszok megtekintése oldal
- Kvízkérelem leadása oldal
- Kijelentkezés

- **Bejelentkezett adminisztrátor jogosultságai:**

- Új Kvíz létrehozása oldal
- Kvíz módosítása oldal
- Kvíz törlése

2.3.2. Navigációs sáv

A navigációs sáv az oldal tetején található, és minden oldalon elérhető, biztosítva a könnyű és gyors navigációt az alkalmazás különböző részei között. A navigációs sáv tartalma dinamikusan változik annak függvényében, hogy a felhasználó be van-e jelentkezve, és ha igen, milyen szerepkörrel rendelkezik, így mindig az aktuálisan elérhető funkciók jelennek meg. A sáv bal szélén található logóra vagy QuizMaster felíratra kattintva a kezdőlapra navigálhatunk.

Kisebb képernyők esetén a menüpontok egy hamburger menü (3 sáv) alatt megtalálhatóak, amely a navigációs sáv jobb oldalán helyezkedik el:



7. ábra: Kisebb képernyő esetén hamburger menü a navigációs sávon

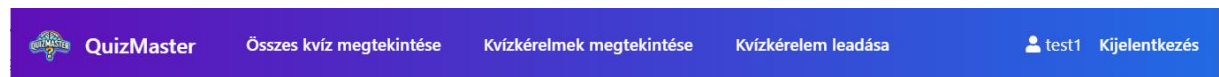
- **Be nem jelentkezett felhasználó navigációs sávja:**



8. ábra: Be nem jelentkezett felhasználó navigációs sávja

Egy be nem jelentkezett felhasználó az Összes kvíz oldalára navigálhat, ott megtekintheti az összes jelenlegi kvízt és az eredményeiket, de kitölteni nem tudja azokat. Valamint nyilvánvalóan a Regisztrációs és Bejelentkező oldal is elérhető számára a navigációs sáv segítségével.

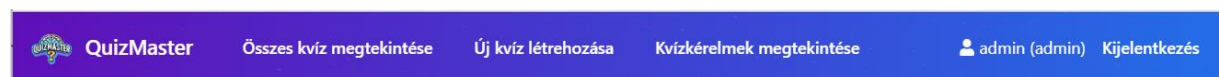
- **Bejelentkezett alapfelhasználó navigációs sávja:**



9. ábra: Bejelentkezett alapfelhasználó navigációs sávja

Egy bejelentkezett alapfelhasználó az Összes kvíz oldalára navigálhat, ott megtekintheti az összes jelenlegi kvízt, az eredményeiket, az előző válaszait és kitöltheti azokat. Továbbá, a Kvízkérelmek oldalára navigálhat a felhasználó, ahol az összes kvízkérérelmet megtekintheti és szavazhat más felhasználók ötletére. A Kvízkérelem leadása oldalra navigálva pedig új kvízkérérelmet adhat le a felhasználó. Természetesen kijelentkezésre is bármikor lehetősége van egy bejelentkezett felhasználónak.

- **Bejelentkezett adminisztrátor navigációs sávja:**



10. ábra: Bejelentkezett adminisztrátor navigációs sávja

Egy bejelentkezett adminisztrátor az Összes kvíz oldalára navigálhat, ott megtekintheti az összes jelenlegi kvízt, az eredményeiket és módosíthatja vagy törölheti azokat. Az Új kvíz létrehozása oldalra navigálva teljesen új kvízt hozhat létre az adminisztrátor. A Kvízkérelmek oldalán az összes kvízkérérelmet megtekintheti és innen témát választhat az új kvíz létrehozásához. Természetesen kijelentkezésre is bármikor lehetősége van egy bejelentkezett felhasználónak.

2.3.3. Kezdőlap

A kezdőlap a QuizMaster alkalmazás bemutatkozó oldala, ahol a felhasználók ízelítőt kapnak az alkalmazás lehetőségeiről és céljáról. Itt egy rövid ismertető mellett találhatóak navigációs gombok is, amelyekkel a felhasználók regisztrálhatnak vagy közvetlenül a kvízek megtekintésére léphetnek. Egy bejelentkezett felhasználó esetén a 'Csatlakozom' gomb már nem látható.



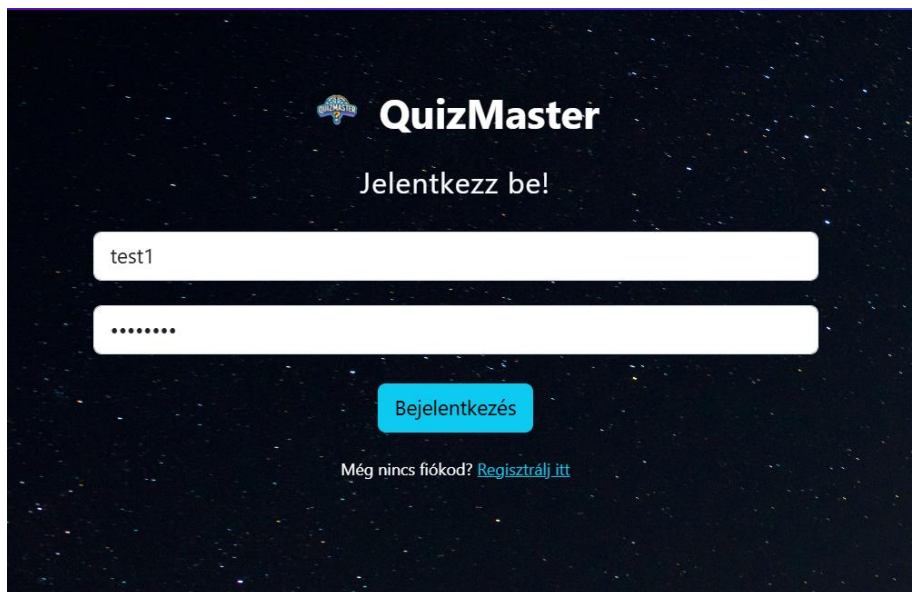
11. ábra: Kezdőlap

2.3.4. Bejelentkező oldal

A bejelentkezési oldal biztosítja a regisztrált felhasználók számára az alkalmazásba való belépést, ahol megadhatják felhasználónevüket és jelszavukat. Ha a bejelentkezés sikertelen, egy hibaüzenet jelenik meg, amely tájékoztatja a felhasználót a hibáról. Az oldalról elérhető a regisztrációs oldal is, amely a be nem jelentkezett felhasználóknak lehetőséget biztosít fiók létrehozására.

Az adatbázisban létrehozott teszt felhasználók:

- Felhasználónév: User1 – Jelszó: User123!
- Felhasználónév: User2 – Jelszó: User234!
- Felhasználónév: User3 – Jelszó: User345!
- Felhasználónév: User4 – Jelszó: User456!
- Felhasználónév: admin – Jelszó: Admin123! (adminisztrátor)



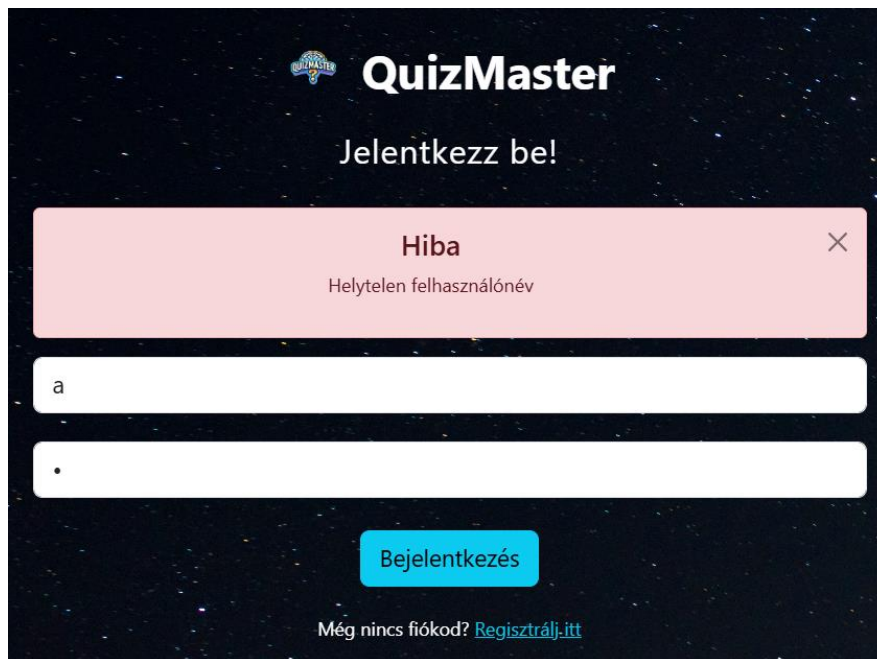
The image shows the QuizMaster login page. At the top, there is a logo with a wizard hat and the text "QuizMaster". Below the logo, the text "Jelentkezz be!" is displayed. There are two input fields: the first contains the text "test1", and the second contains a series of dots representing a password. Below the input fields is a blue button labeled "Bejelentkezés". At the bottom, there is a link that says "Még nincs fiókod? [Regisztrálj itt](#)".

12. ábra: Bejelentkező űrlap

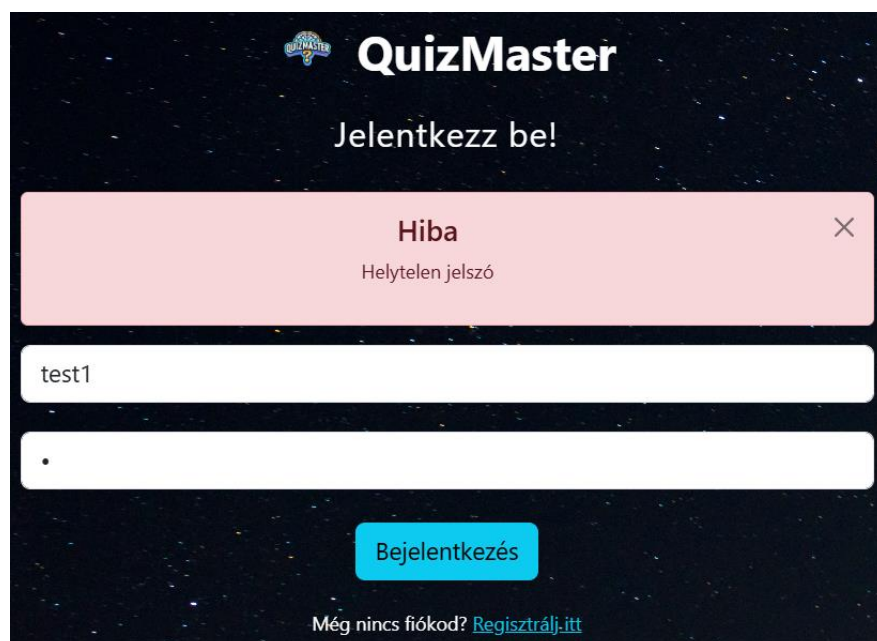


The image shows the QuizMaster login page with an error message. At the top, there is a logo with a wizard hat and the text "QuizMaster". Below the logo, the text "Jelentkezz be!" is displayed. A red error message box is shown, containing the text "Hiba" and "A felhasználónév és a jelszó megadása kötelező." Below the error message, there are two input fields: the first contains the text "a", and the second is labeled "Jelszó". Below the input fields is a blue button labeled "Bejelentkezés". At the bottom, there is a link that says "Még nincs fiókod? [Regisztrálj itt](#)".

13. ábra: Bejelentkezés – kötelezően kitöltendő mezők hiba



14. ábra: Bejelentkezés – nem létező felhasználónév hiba

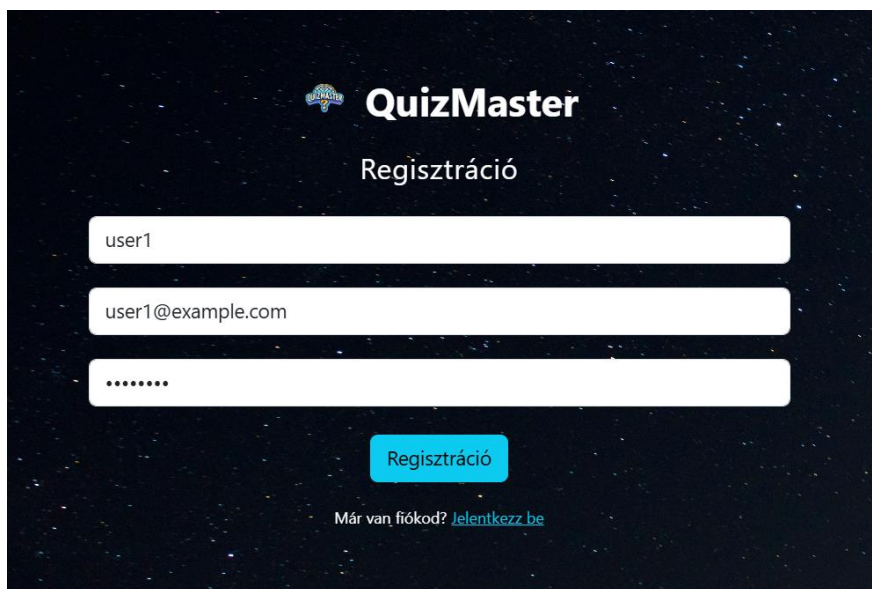


15. ábra: Bejelentkezés – helytelen jelszó hiba

2.3.5. Regisztrációs oldal

A regisztrációs oldal lehetőséget biztosít új felhasználók számára, hogy fiókot hozzanak létre az alkalmazásban. Itt megadhatják a szükséges adatokat, mint a felhasználónév, email cím, és

jelszó. Sikeres regisztráció után a felhasználók automatikusan beléptetjük az alkalmazásba, és a kezdőlapra navigáljuk. Ha a megadott adatok hibásak vagy hiányosak, az oldal figyelmezteti a felhasználót, és segítséget nyújt a regisztráció sikeres befejezéséhez.



The image shows the registration page of the QuizMaster application. At the top, there is a logo and the text "QuizMaster Regisztráció". Below this, there are three input fields: the first contains "user1", the second contains "user1@example.com", and the third is a password field with seven dots. A blue button labeled "Regisztráció" is positioned below the fields. At the bottom, there is a link that says "Már van fiókod? [Jelentkezz be](#)".

16. ábra: Regisztrációs űrlap



The image shows the registration page of the QuizMaster application with an error message. At the top, there is a logo and the text "QuizMaster Regisztráció". Below this, there is a red error message box that says "Hiba" and "Minden mező kitöltése kötelező." (All fields are required). Below the error message, there are three input fields: the first contains "user1", the second is labeled "Email cím", and the third is labeled "Jelszó". A blue button labeled "Regisztráció" is positioned below the fields. At the bottom, there is a link that says "Már van fiókod? [Jelentkezz be](#)".

17. ábra: Regisztráció - kötelezően kitöltendő mezők hiba

The registration form is titled "Regisztráció" (Registration). It features a red error message box with the title "Hiba" (Error) and the text "Ez a felhasználónév már foglalt." (This username is already taken). Below the error message is a single input field containing the text "test1".

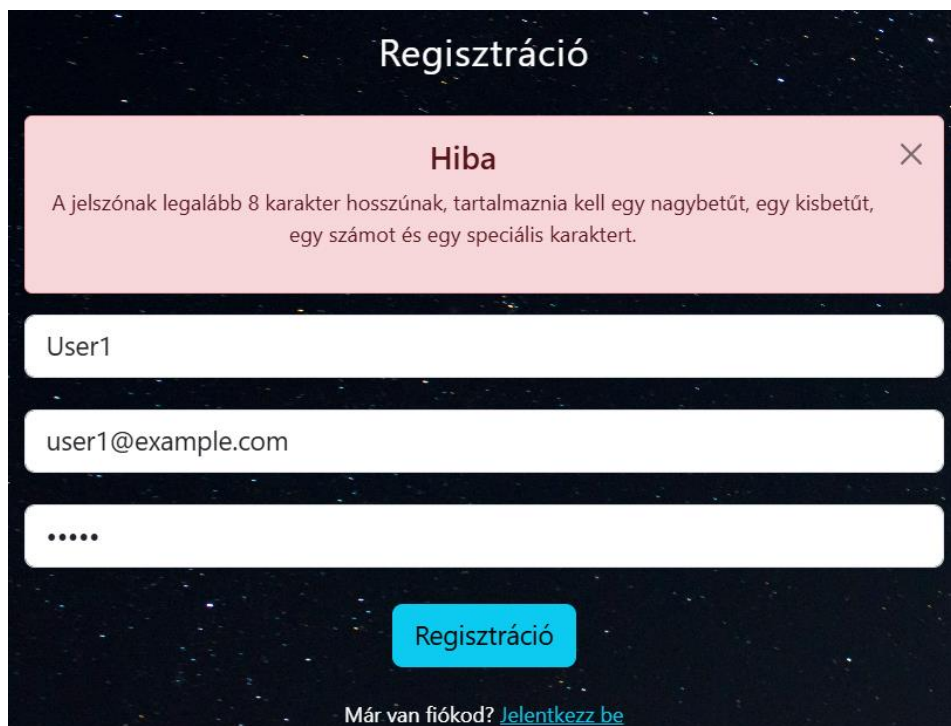
18. ábra: Regisztráció – foglalt felhasználónév hiba

The registration form is titled "Regisztráció" (Registration). It features a red error message box with the title "Hiba" (Error) and the text "Ez az email cím már regisztrálva van." (This email address is already registered). Below the error message are two input fields: the first contains "User1" and the second contains "test1@test1.hu".

19. ábra: Regisztráció - regisztrált email cím hiba

The registration form is titled "Regisztráció" (Registration). It features a red error message box with the title "Hiba" (Error) and the text "Az email cím formátuma érvénytelen." (The email address format is invalid). Below the error message are two input fields: the first contains "User1" and the second contains "user1@user1".

20. ábra: Regisztráció – érvénytelen email cím formátum hiba

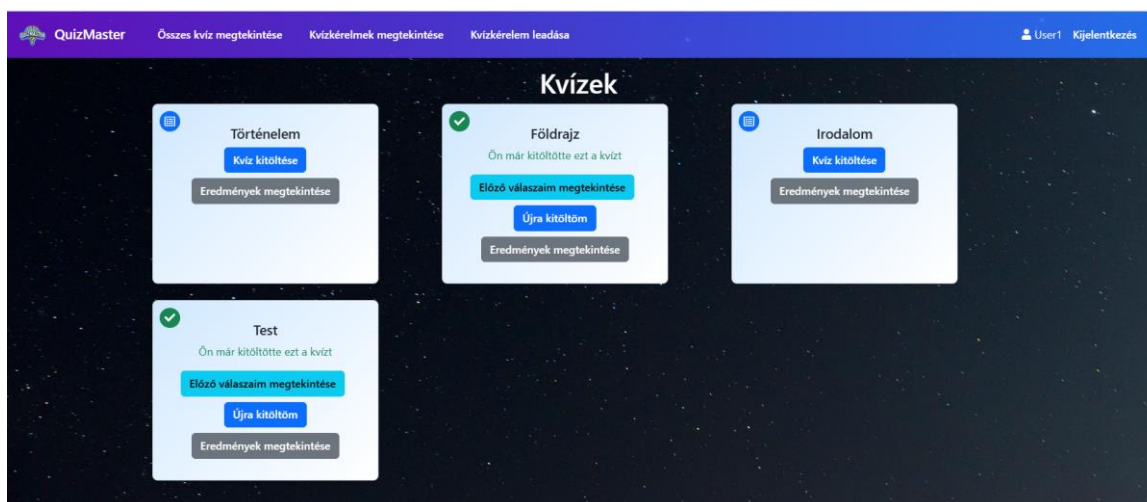


The image shows a registration form titled "Regisztráció" on a dark background. At the top, a pink error box with a close icon contains the text: "Hiba" and "A jelszónak legalább 8 karakter hosszúnak, tartalmaznia kell egy nagybetűt, egy kisbetűt, egy számot és egy speciális karaktert." Below the error box are three input fields: the first contains "User1", the second contains "user1@example.com", and the third contains six dots representing a password. A blue "Regisztráció" button is centered below the fields. At the bottom, there is a link: "Már van fiókod? [Jelentkezz be](#)".

21. ábra: Regisztráció – gyenge jelszó hiba

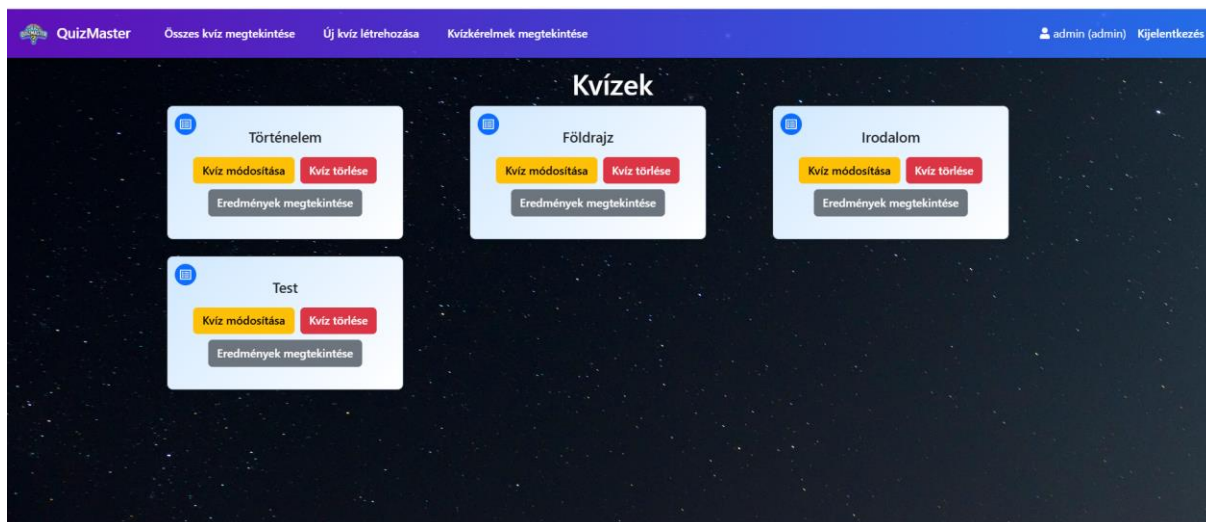
2.3.6. Összes Kvíz oldal

Az Összes Kvíz oldal lehetőséget biztosít a bejelentkezett felhasználóknak, hogy átlássák az alkalmazásban elérhető összes kvízt. A felhasználók könnyedén elérhetik a kvízek kitöltési oldalát, és megtekinthetik előző válaszaikat az általuk már kitöltött kvízeknél. A kvíz eredmények oldalán összehasonlíthatják saját teljesítményüket másokéval, illetve, ha kívánják, újból kitölthetik a kvízeket.



22. ábra: Összes Kvíz oldal (alapfelhasználó)

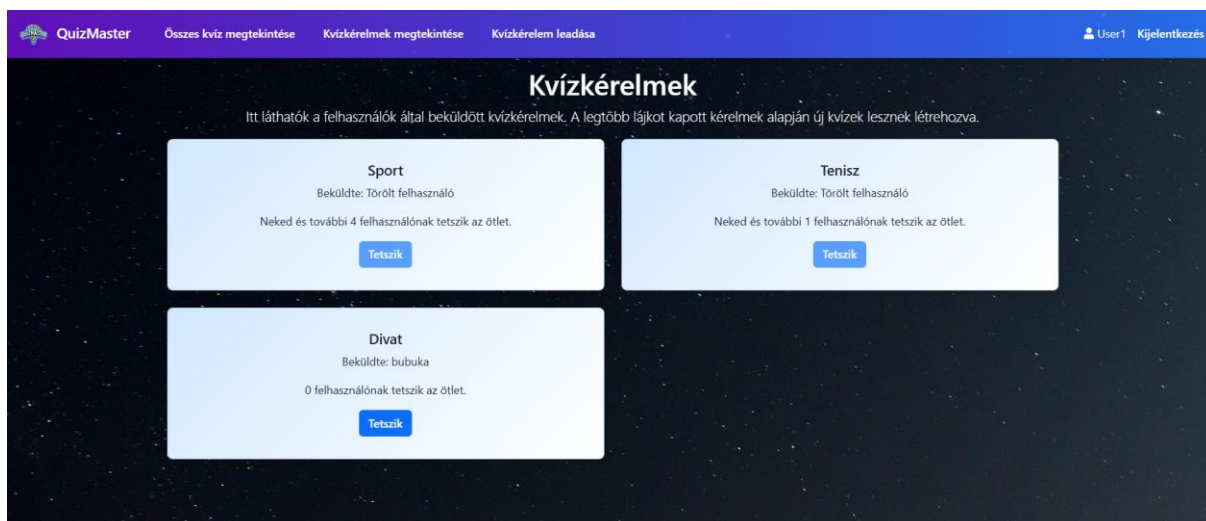
Az Összes Kvíz oldal az adminisztrátor számára átfogó képet nyújt a rendszerben elérhető összes kvízzól. Itt az adminisztrátor megtekintheti a kvizek részleteit, szükség esetén módosíthatja őket, vagy törölheti a már nem kívánt kvizeket. Az admin így könnyen kezelheti és alakíthatja az alkalmazásban elérhető tartalmat, hogy a felhasználók számára mindig naprakész és releváns kvizek álljanak rendelkezésre.



23. ábra: Összes Kvíz oldal (adminisztrátor)

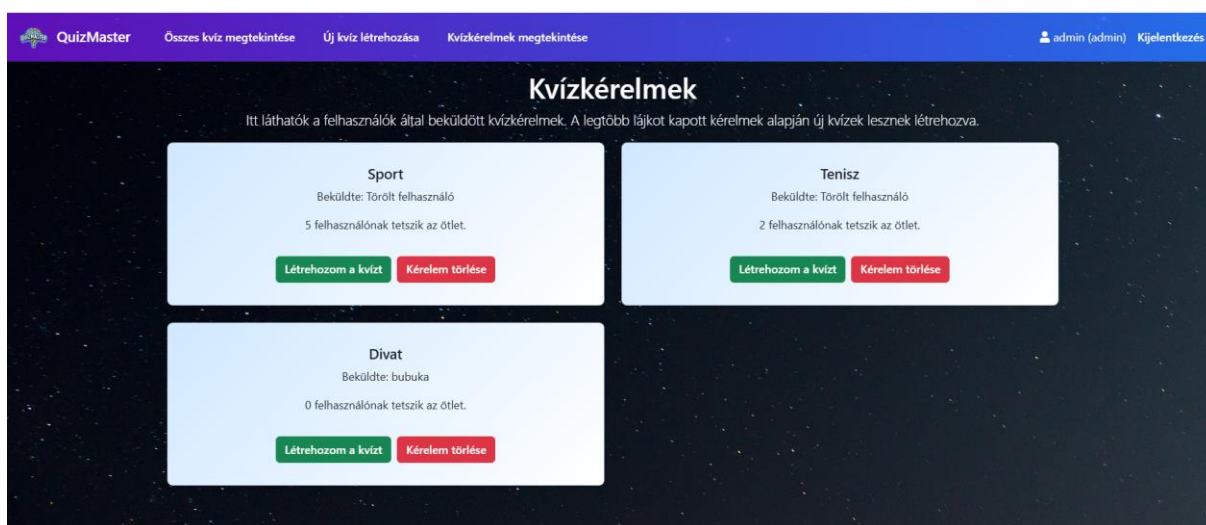
2.3.7. Kvízkérelmek oldal

Az Kvízkérelmek oldal lehetőséget ad az alapfelhasználóknak, hogy megtekinthessék a többi felhasználó által beküldött kvízötleteket, és szavazzanak azokra az ötletekre, amelyek különösen tetszenek nekik. A felhasználók szavazatai alapján az adminisztrátorok eldönthetik, hogy mely ötletekből lesznek új kvizek. Ezáltal az alapfelhasználók közvetlenül is befolyásolhatják az alkalmazás tartalmát, és hozzájárulhatnak új témák megjelenéséhez az alkalmazásban.



24. ábra: Kvízkérelmek oldal (alapfelhasználó)

Az Kvízkérelmek oldal az adminisztrátorok számára lehetőséget nyújt arra, hogy áttekinthessék a felhasználók által beküldött kvízötleteket, és azok népszerűsége alapján döntsenek az ötletek megvalósításáról. Az oldal listázza a felhasználók által javasolt témákat és az ezekre leadott szavazatokat, így az adminisztrátorok egyszerűen láthatják, melyik kvízötlet a legkedveltebb. Az adminisztrátoroknak lehetőségük van azonnal kvízt létrehozni egy adott ötletből, amely egyszerűsíti a kvíztémák bővítését, és aktívan bevonja a közösséget az új tartalmak kialakításába. Emellett az adminisztrátorok kérelmek törlésére is jogosultak, amennyiben egy ötlet nem felel meg a platform irányelveinek vagy céljainak.



25. ábra: Kvízkérelmek oldal (adminisztrátor)

2.3.8. Kvízkérelem leadása oldal

A Kvízkérelem leadása oldal lehetőséget biztosít a regisztrált felhasználók számára, hogy új kvíztémákat javasoljanak az alkalmazásba. Az oldal egy egyszerű űrlapot tartalmaz, ahol a felhasználók megadhatják az ötletük címét, és a leadás után a kérelem az adminisztrátorok számára is láthatóvá válik. Amennyiben a kérelem sikeresen feldolgozásra került, a felhasználó a kérések listájára kerül vissza. Ha valamilyen hiba lép fel a folyamat során, figyelmeztető üzenet jelenik meg, amely segít a felhasználónak a probléma megértésében.

26. ábra: Kvízkérelem leadása oldal

27. ábra: Kvízkérelem leadása – létező kérelem hiba

28. ábra: Kvízkérelem leadása – létező kvíz hiba

2.3.9. Eredmények oldal

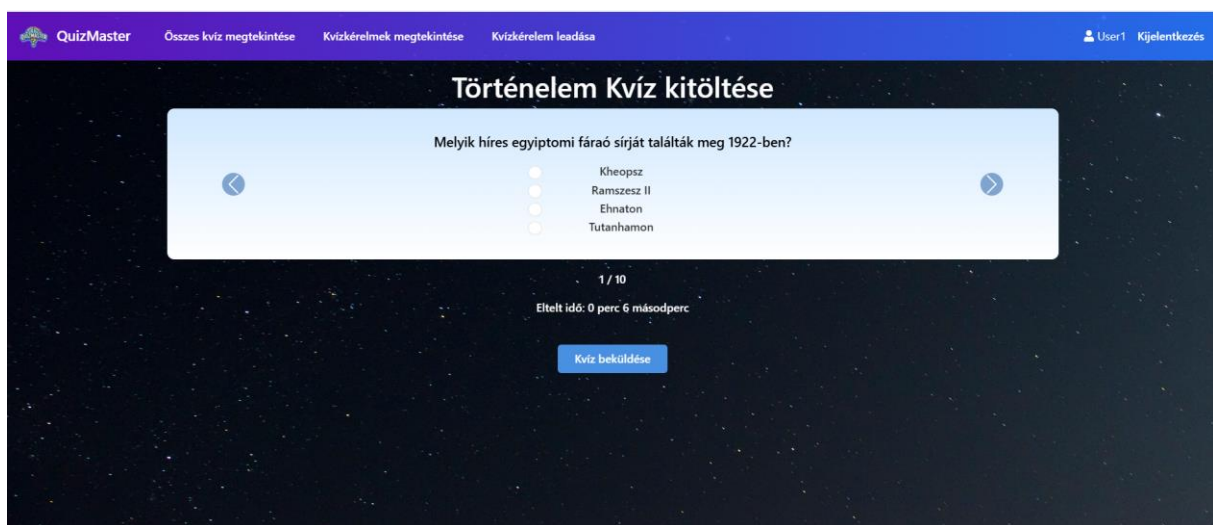
Az Eredmények oldal az adott kvíz legjobb kitöltéseit jeleníti meg egy ranglistában. Itt a felhasználók összehasonlíthatják teljesítményüket más játékosokkal, és láthatják, ki ért el a legtöbb pontot a leggyorsabb idő alatt. A lista tartalmazza a felhasználó nevét, a kvíz során szerzett pontszámot, valamint a kitöltéshez szükséges időt, így könnyen áttekinthetővé válik, ki áll az élen. Az oldal automatikusan rendezi az eredményeket pontszám és kitöltési idő alapján.

Rank	Felhasználó	Pontszám	Kitöltési idő
1	User2	5/5	0 perc 15 másodperc
2	bubuka	5/5	0 perc 24 másodperc
3	User1	4/5	0 perc 17 másodperc
4	test1	2/5	0 perc 15 másodperc

29. ábra: Eredmények oldal

2.3.10. Kvíz kitöltése oldal

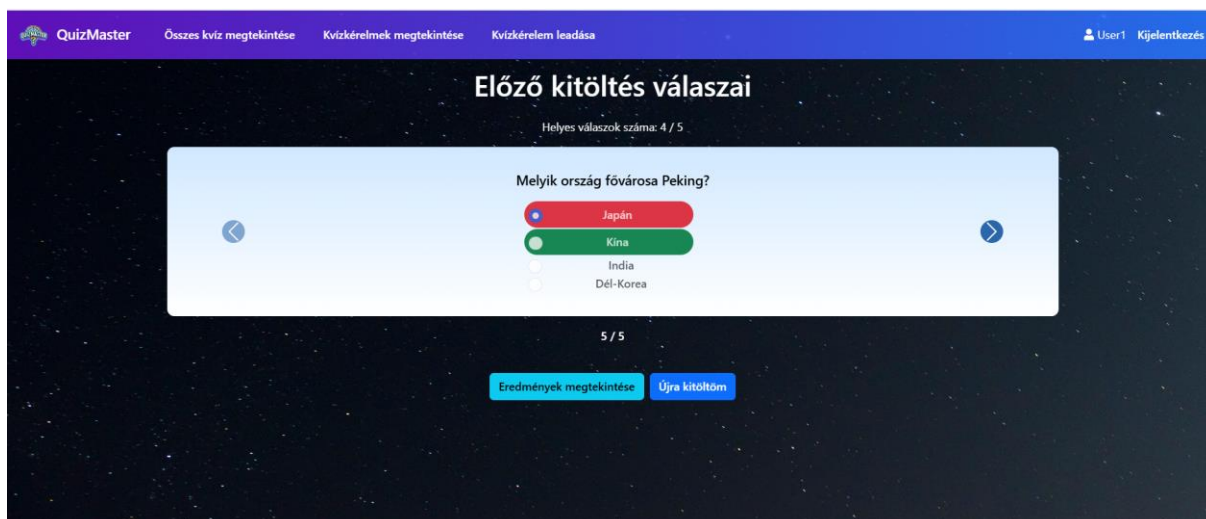
A Kvíz kitöltése oldal lehetőséget biztosít a felhasználóknak, hogy megválaszolják a kvíz kérdéseit, és próbára tegyék tudásukat egy időzített formában. A kérdéseket véletlenszerű sorrendben, egyenként kapják meg, és minden kérdéshez válaszlehetőségek tartoznak, melyek szintén keverten jelennek meg. A felhasználóknak lehetőségük van egy kérdés megválaszolása után a következőre lépni, vagy visszalépni az előző kérdésre. Miután kitöltötték a kvízt, azonnal látják a helyes és helytelen válaszaikat, az eltelt időt és a helyes válaszok számát. Az oldalon elérhető egy Újra kitöltöm lehetőség is, melynek segítségével a felhasználók újra próbálkozhatnak, ha szeretnék javítani az eredményükön.



30. ábra: Kvíz kitöltése oldal

2.3.11. Előző válaszok megtekintése oldal

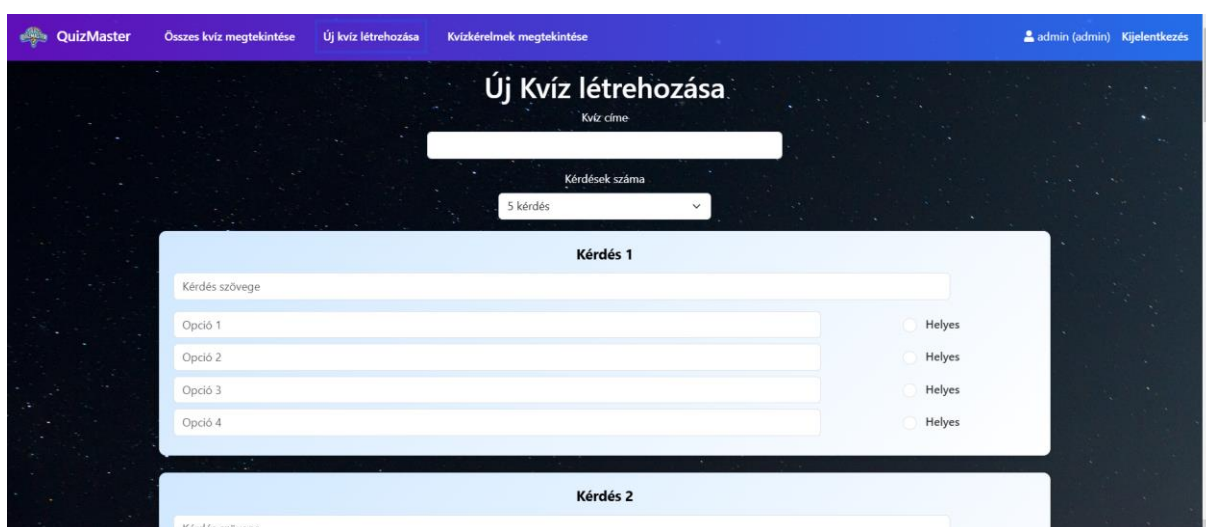
Az Előző válaszok megtekintése oldal lehetőséget biztosít a felhasználók számára, hogy visszanézzék előző kvízkitöltésüket, áttekinthessék válaszaikat a kérdésekre és lássák helyes válaszaik számát. Az oldalon az adott kvíz minden kérdése és az azokhoz tartozó válaszlehetőségek jelennek meg, kiemelve a helyesen és helytelenül megválaszolt kérdéseket, így a felhasználók könnyen láthatják, hogy mely kérdésekre válaszoltak helyesen és melyekre nem. Az oldal segít a felhasználóknak a tanulásban és fejlődésben, mivel visszajelzést nyújt a tudásukról és lehetőséget ad az újra kitöltésre is, hogy javíthassanak eredményeiken.



31. ábra: Előző válaszok megtekintése oldal

2.3.12. Új Kvíz létrehozása oldal

Az Új Kvíz létrehozása oldal kizárólag az adminisztrátorok számára érhető el, és lehetőséget biztosít egy új kvíz összeállítására az alkalmazásban. Az oldalon az adminisztrátor megadhatja a kvíz címét és beállíthatja a kérdések számát (5, 10 vagy 20), majd ennek alapján feltöltheti a szükséges kérdéseket és válaszlehetőségeket. Minden kérdéshez legalább négy válaszopciót kell hozzárendelni, amelyből az egyik helyes válaszként megjelölhető. Az új kérdések könnyedén hozzáadhatók, míg a felesleges kérdések törölhetők. Az adminisztrátor végül elmentheti a kvízt, amely ezután elérhetővé válik a felhasználók számára.



32. ábra: Új Kvíz létrehozása oldal

Új Kvíz létrehozása

Hiba

A kvíz címe kötelező.

X

Kvíz címe

33. ábra: Új Kvíz létrehozása – kötelező cím hiba

Hiba

Minden kérdéshez meg kell adni a kérdés szövegét.

X

Kvíz címe

Sport

Kérdések száma

5 kérdés

Kérdés 1

Kérdés szövege

Opció 1

Opció 2

Opció 3

Opció 4

☐ Helyes
☐ Helyes
☐ Helyes
☐ Helyes

34. ábra: Új Kvíz létrehozása – hiányzó kérdések hiba

Hiba

Minden kérdéshez meg kell adni minden opció szövegét.

X

Kvíz címe

Sport

Kérdések száma

5 kérdés

Kérdés 1

Kérdés 1

Opció 1

Opció 2

Opció 3

Opció 4

☐ Helyes
☐ Helyes
☐ Helyes
☐ Helyes

35. ábra: Új Kvíz létrehozása – hiányzó opciók hiba

Hiba ✕

Minden kérdéshez ki kell választani egy helyes opciót.

Kvíz címe

Sport

Kérdések száma

5 kérdés

Kérdés 1

Kérdés 1

Opció 1

Opció 2

Opció 3

Opció 4

☐ Helyes

☐ Helyes

☐ Helyes

☐ Helyes

36. ábra: Új Kvíz létrehozása – hiányzó helyes opció hiba

Új Kvíz létrehozása

Hiba ✕

Már létezik ilyen nevű kvíz.

Kvíz címe

Irodalom

37. ábra: Új Kvíz létrehozása – létező kvíz hiba

2.3.13. Kvíz szerkesztése oldal

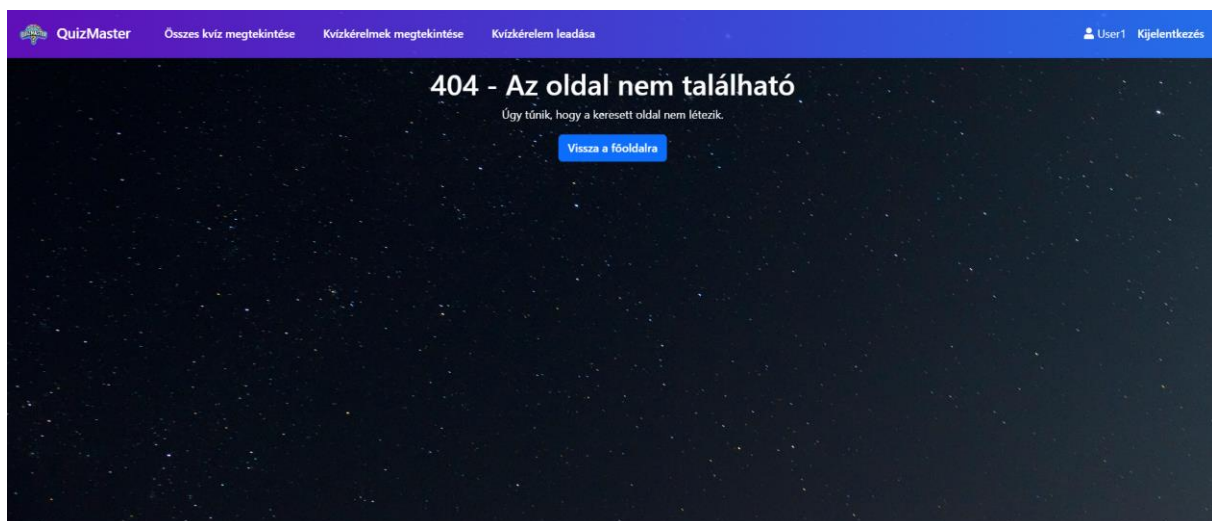
A Kvíz szerkesztése oldal lehetőséget nyújt az adminisztrátorok számára, hogy meglévő kvizeket módosítsanak. Ezen az oldalon az adminisztrátor láthatja a kvíz címét (melyet nem tud szerkeszteni) és a kérdések számát, amely szintén rögzített. Az adminisztrátor módosíthatja a kérdések szövegét és válaszlehetőségeit, új kérdéseket adhat hozzá, vagy törölhet kérdéseket, amíg a kvíz legalább kétszer annyi kérdést tartalmaz, mint a 'Kérdések száma'. Az oldal biztosítja a megfelelő kérdésstruktúra fenntartását, és figyelmeztető üzenetet jelenít meg, ha egy szerkesztési művelet nem megengedett.

38. ábra: Kvíz szerkesztése oldal

A hibaüzenetek megegyeznek a Kvíz létrehozásakor látható hibaüzenetekkel (hiányzó kérdések hiba, hiányzó opciók hiba, hiányzó helyes opció hiba).

2.3.14. Az oldal nem található – 404

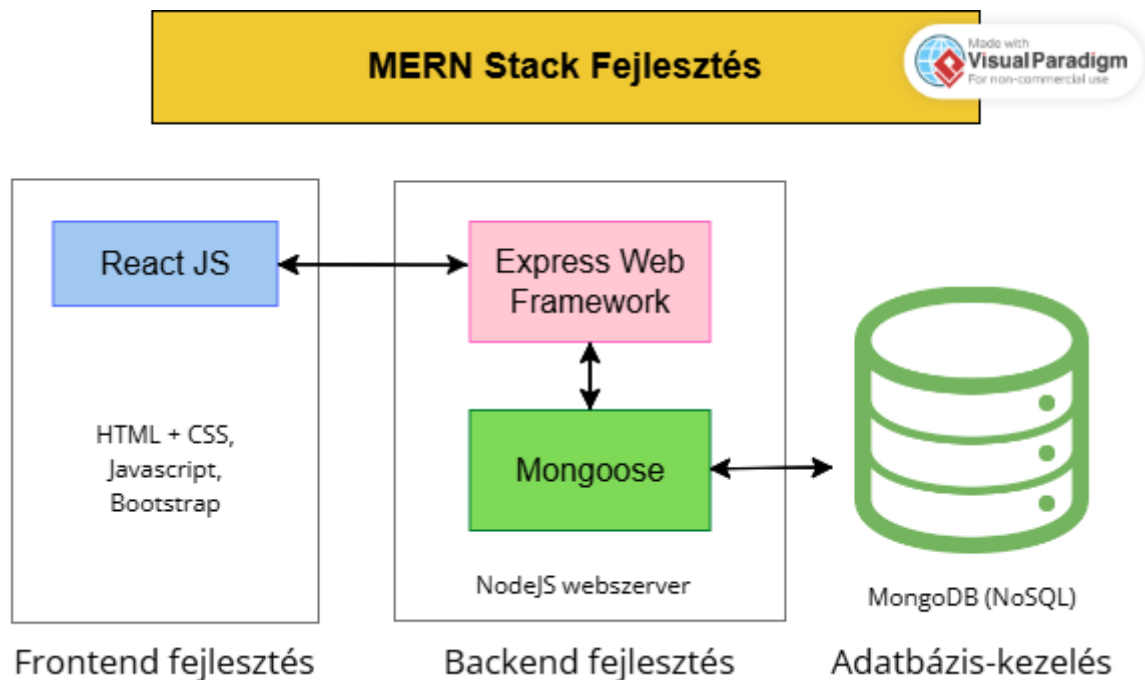
A 404 - Az oldal nem található oldal egy egyszerű hibaüzenetet jelenít meg, amikor a felhasználó olyan oldalra próbál navigálni, amely nem létezik az alkalmazásban. A felhasználó itt egy rövid üzenetet kap arról, hogy a keresett oldal nem található, és egy gomb segítségével visszatérhet a főoldalra.



39. ábra: 404 - Az oldal nem található

3. Fejlesztői dokumentáció

3.1. Az alkalmazás sematikus felépítése



40. ábra: Az alkalmazás sematikus felépítése

A fenti ábrán a **MERN stack fejlesztés** főbb elemei láthatók, amely a modern webalkalmazások fejlesztésének egyik népszerű technológiai stackje. A **frontend fejlesztést** a React JS biztosítja, amely dinamikus és interaktív felhasználói felületek létrehozására szolgál, támogatva a **HTML**, **CSS**, **JavaScript** és **Bootstrap** technológiákat. A **backend fejlesztés** során az **Express Web Framework** és a **Node.js webszerver** gondoskodik a kiszolgálói oldalon futó alkalmazáslogikáról. Az adatbázis-kezelést a **MongoDB** látja el, amely egy NoSQL alapú, rugalmas adatkezelési megoldás. Az összeköttetést az alkalmazás és az adatbázis között a **Mongoose** biztosítja, amely egy erőteljes objektum-dokumentum modellező könyvtár, lehetővé téve az adatok strukturált kezelését.

3.2. Elemzés

3.2.1. Feladat leírása

Az alkalmazás célja egy online kvízrendszer létrehozása, amely lehetőséget biztosít felhasználóknak különböző tematikájú kvizek kitöltésére, saját tudásuk tesztelésére, valamint barátaikkal és a közösséggel való versenyzésre. Az alkalmazás biztosítja az adminisztrátorok számára a kvizek egyszerű kezelését, új kérdések hozzáadását, módosítását és felhasználói kérések alapján új kvizek létrehozását. A rendszer emellett felhasználói pontszámokat is rögzít és ranglistát vezet, így ösztönözve a felhasználókat a fejlődésre.

Az alkalmazás a következő feladatokat látja el:

- **Felhasználói felület:**
 - Intuitív és átlátható felület biztosítása a különböző felhasználói szerepkörök (felhasználók, adminisztrátorok) számára.
- **Felhasználókezelés:**
 - Lehetővé teszi új felhasználók számára a regisztrációt.
 - Biztosítja a bejelentkezési és kijelentkezési funkciókat a már meglévő felhasználóknak.
- **Kvízkezelés:**
 - A felhasználók számára lehetőséget biztosít különböző kvizek böngészésére és kitöltésére.
 - Lehetővé teszi a felhasználóknak, hogy újra kitölthessék a már korábban kitöltött kvizeket.
 - Adminisztrátori jogosultsággal rendelkező felhasználók számára biztosítja a kvizek létrehozását, szerkesztését, valamint az aktuális kvízkérélmek kezelését.
- **Pontszám- és ranglistakezelés:**
 - Minden egyes kitöltés után menti a felhasználó eredményeit.

- Pontszám alapján ranglistát vezet az egyes kvízeknél, hogy a felhasználók összehasonlíthassák teljesítményüket.
- A rendszer biztosítja, hogy minden felhasználó csak a legjobb eredményét láthassa a ranglistán, de megőrizze a legutóbbi próbálkozás adatait.
- **Kvízkérelmek kezelése:**
 - Lehetővé teszi a felhasználóknak, hogy új kvíztémákat kérjenek, melyeket a közösség is támogathat.
 - Az adminisztrátorok kezelhetik és feldolgozhatják a kvízkérelmeket, új kvízeket hozhatnak létre a legtöbb támogatást kapott témákból.

3.2.2. Funkcionális leírás

- **Felhasználói esetek**

Eset		Eset leírása	
1	Főoldal megjelenítése	GIVEN	A felhasználó elindította az alkalmazást a lokális környezetében és beírta a böngészőjébe a "http://localhost" címet.
		WHEN	A felhasználó megnyitja ezt az oldalt.
		THEN	Az alkalmazás főoldala megjelenik.

2	Felhasználói regisztráció	GIVEN	A felhasználó még nem regisztrált az oldalon, és a regisztrációs felületre navigált.
		WHEN	Kitölti a regisztrációs adatokat, beleértve a felhasználónevet, email címet, jelszót, majd rákattint a "Regisztráció" gombra.
		THEN	A rendszer sikeresen regisztrálja a felhasználót, és átirányítja őt a bejelentkezési oldalra.
3	Bejelentkezés az alkalmazásba	GIVEN	A felhasználó már regisztrált, és a bejelentkezési oldalon tartózkodik.
		WHEN	Beírja a felhasználónevét és jelszavát, majd a "Bejelentkezés" gombra kattint.
		THEN	A rendszer bejelentkezteti a felhasználót, és átirányítja őt a főoldalra.

4	Navigáció az alkalmazás különböző funkciói között	GIVEN	A felhasználó az alkalmazás bármely oldalán tartózkodik.
		WHEN	A navigációs menüből egy adott menüpontra kattint.
		THEN	A kiválasztott oldal betöltődik, és a felhasználó megtekintheti az adott oldal tartalmát.
5	Új kvíz kitöltése	GIVEN	A felhasználó bejelentkezett és a kvizek oldalon tartózkodik.
		WHEN	A felhasználó kiválaszt egy kvízt, majd a "Kvíz kitöltése" gombra kattint.
		THEN	A kiválasztott kvíz első kérdése megjelenik, és a felhasználó elkezdheti a kvíz kitöltését.
6	Kvíz előző kitöltésének válaszainak megtekintése	GIVEN	A felhasználó az összes kvíz oldalán van az alkalmazásban.
		WHEN	Az adott kvíz kártyáján az „Előző válaszaim megtekintése” gombra kattint.
		THEN	Megjelennek a felhasználó korábbi válaszai és elért pontszáma az adott kvízhez.

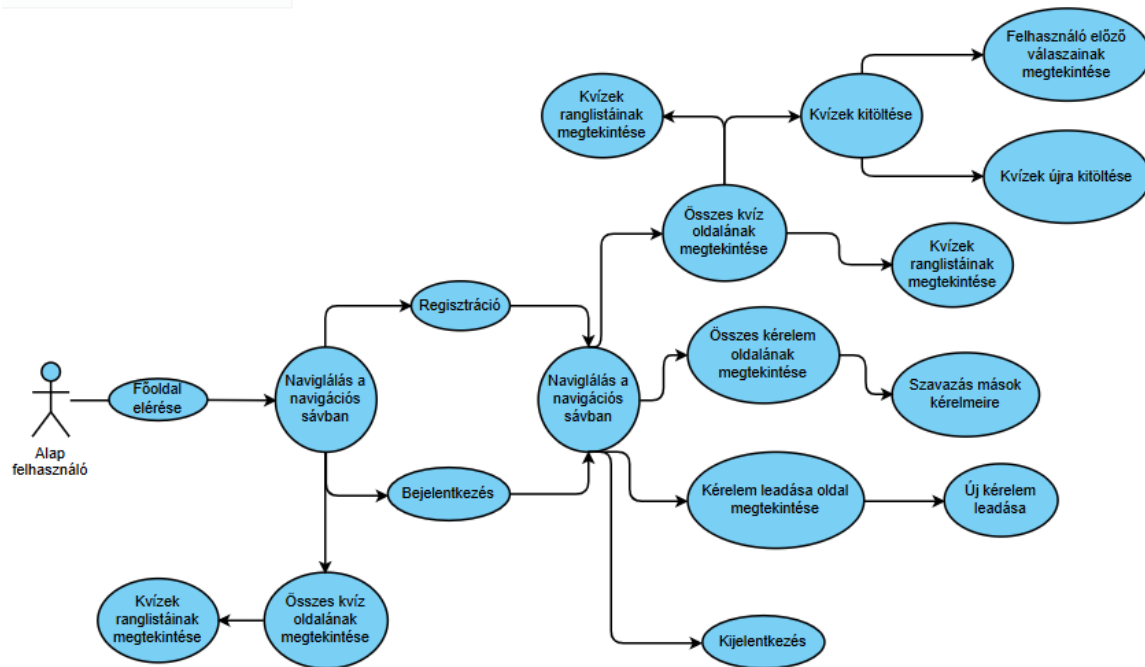
7	Kvíz újra kitöltése	GIVEN	A felhasználó már kitöltött egy kvízt.
		WHEN	A kitöltés után egyből vagy az összes kvíz oldalon az adott kvíz kártyáján az „Újra kitöltöm” gombra kattint.
		THEN	A rendszer lehetőséget ad az újra kitöltésre, ezért megjelenik a kvíz első kérdése.
8	Kvíz ranglista megtekintése	GIVEN	A felhasználó az összes kvíz oldalán van az alkalmazásban.
		WHEN	Az adott kvíz kártyáján az „Eredmények megtekintése” gombra kattint.
		THEN	Megjelenik az adott kvíz ranglistája, ahol láthatja a saját és más felhasználók legjobb eredményeit.
9	Kvízkérelem beküldése	GIVEN	A felhasználó bejelentkezett és a "Kvízkérelem leadása" oldalon tartózkodik.
		WHEN	A felhasználó beírja a kvíz témáját, majd a "Kérelem leadása" gombra kattint.
		THEN	A rendszer rögzíti a kérést, amely így megjelenik a kvízkérelmek oldalán.

10	Adminisztrátor: Új kvíz létrehozása	GIVEN	Az adminisztrátor bejelentkezett az alkalmazásba és az „Új kvíz létrehozása” oldalon van
		WHEN	Kitölti a kvíz címét, kérdéseit és válaszlehetőségeit, majd a „Kvíz létrehozása” gombra kattint.
		THEN	Az új kvíz elmentésre kerül, és megjelenik a kvízlistában.
11	Adminisztrátor: Új kvíz létrehozása kérés alapján	GIVEN	Az adminisztrátor bejelentkezett és az összes kérelem oldalán megtekinti a beküldött kvízkérelmeket.
		WHEN	Egy adott kérésnél a "Létrehozom a kvízt" gombra kattint.
		THEN	A rendszer átirányítja az adminisztrátort a kvíz szerkesztő oldalára a kérdések hozzáadásához.

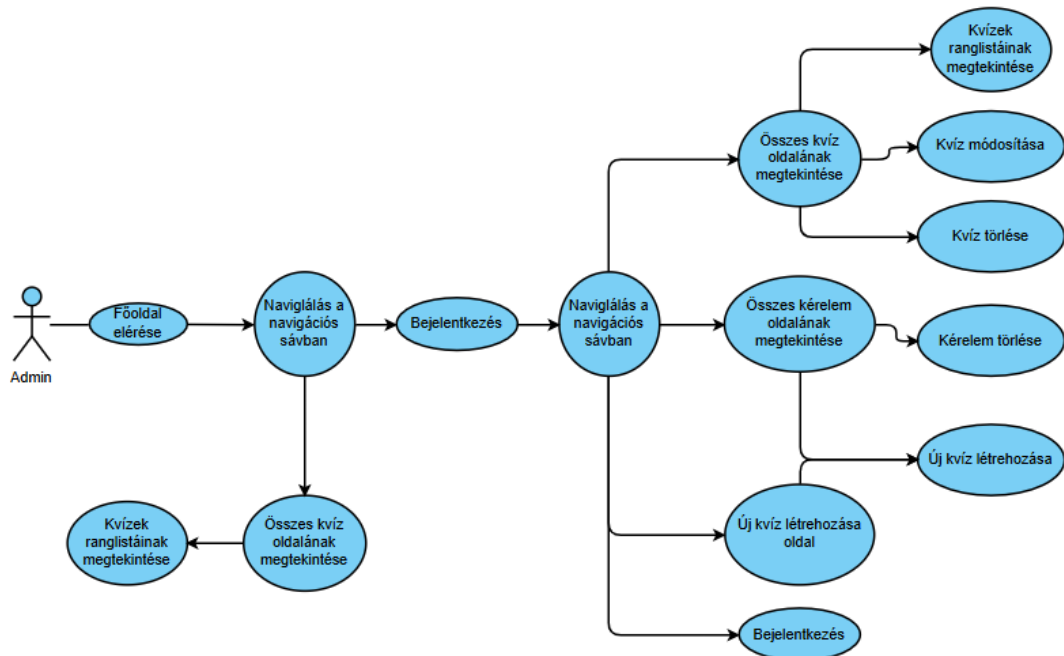
12	Adminisztrátor: Kvíz módosítása	GIVEN	Az adminisztrátor bejelentkezett, és az összes kvíz oldalán van.
		WHEN	A módosítani kívánt kvíz kártyáján a „Kvíz módosítása” gombra kattint, elvégzi a szükséges módosításokat, amiket elment.
		THEN	A kvíz frissített adatai mentésre kerülnek, és az alkalmazásban az új információkkal jelenik meg.
13	Adminisztrátor: Kvíz törlése	GIVEN	Az adminisztrátor bejelentkezett és az összes kvíz oldalán van.
		WHEN	A kívánt kvíz kártyáján a „Kvíz törlése” gombra kattint, és a megjelenő megerősítő ablakban megerősíti szándékát.
		THEN	Az adott kvíz törlődik az adatbázisból és a kvízlistából eltűnik.

14	Adminisztrátor: Kérelem törlése	GIVEN	Az adminisztrátor bejelentkezett és az összes kérelem oldalán van.
		WHEN	A törölni kívánt kérelem kártyáján a „Kérelem törlése” gombra kattint.
		THEN	Az adott kérelem eltűnik a kérések listájából, és törlődik az adatbázisból.

- Felhasználói esetdiagramok



41. ábra: Use case diagram – alap felhasználó



42. ábra: Use case diagram – adminisztrátor

3.2.3. Nem funkcionális követelmények

Biztonság

Az alkalmazásnak olyan biztonsági megoldásokat kell tartalmaznia, amelyek megakadályozzák az illetéktelen hozzáférést és védik a felhasználói adatokat, különösen a kvíz eredményeket és személyes információkat. A bejelentkezéshez JWT token alapú hitelesítést használ, hogy a felhasználók csak a saját adataikhoz férhessenek hozzá.

Teljesítmény

Az alkalmazásnak zökkenőmentes működést kell biztosítania, még nagy számú, egyidejűleg aktív felhasználó mellett is, például amikor több felhasználó egyszerre vesz részt kvízeken vagy tekinti meg az eredményeket. Az adatok gyors elérését optimalizált adatbázis-lekérdezések és hatékony háttér szolgáltatások biztosítják.

Skálázhatóság

A rendszernek támogatnia kell a könnyű skálázást, hogy több felhasználó és kvíz esetén is stabil maradjon a teljesítménye. Ez lehetővé teszi, hogy a háttér szerverek számát rugalmasan növeljük, ha a felhasználók száma jelentősen nő, vagy csökkentsük a terhelést, ha kevesebb aktivitás történik.

Karbantartás

Az alkalmazás kódját és infrastruktúráját úgy kell felépíteni, hogy könnyen frissíthető és karbantartható legyen, például új kvízek hozzáadásakor vagy a rendszeres hibajavítások végrehajtásakor. A tiszta és moduláris kódstruktúra megkönnyíti a bővítést és a hibajavítást.

Hordozhatóság

Az alkalmazás különböző platformokon is működőképes kell, hogy legyen, így például különféle szerverkonfigurációkon, akár Windows, akár Linux rendszereken, minimális változtatásokkal is zavartalanul futtatható.

Megbízhatóság

Az alkalmazás adatkezelése stabil, és minden felhasználói válasz pontosan mentésre kerül, megelőzve az adatvesztést vagy hibás eredmények megjelenését, így a felhasználók mindig megbízható információt kapnak a kvízeikről és eredményeikről.

Használhatóság

Az alkalmazás kialakítása intuitív és átlátható, lehetővé téve, hogy a felhasználók gyorsan eligazodjanak a kvizek között, eredményeik megtekintésében vagy az adminisztrációs feladatokban, ezáltal minden szinten felhasználóbarát.

Kompatibilitás

Az alkalmazásnak együtt kell működnie a gyakran használt böngészőkkel és eszközökkel, hogy a felhasználók zavartalanul használhassák a platformot, függetlenül attól, hogy milyen eszközről vagy operációs rendszerről csatlakoznak.

3.3. Tervezés

3.3.1. Nézet

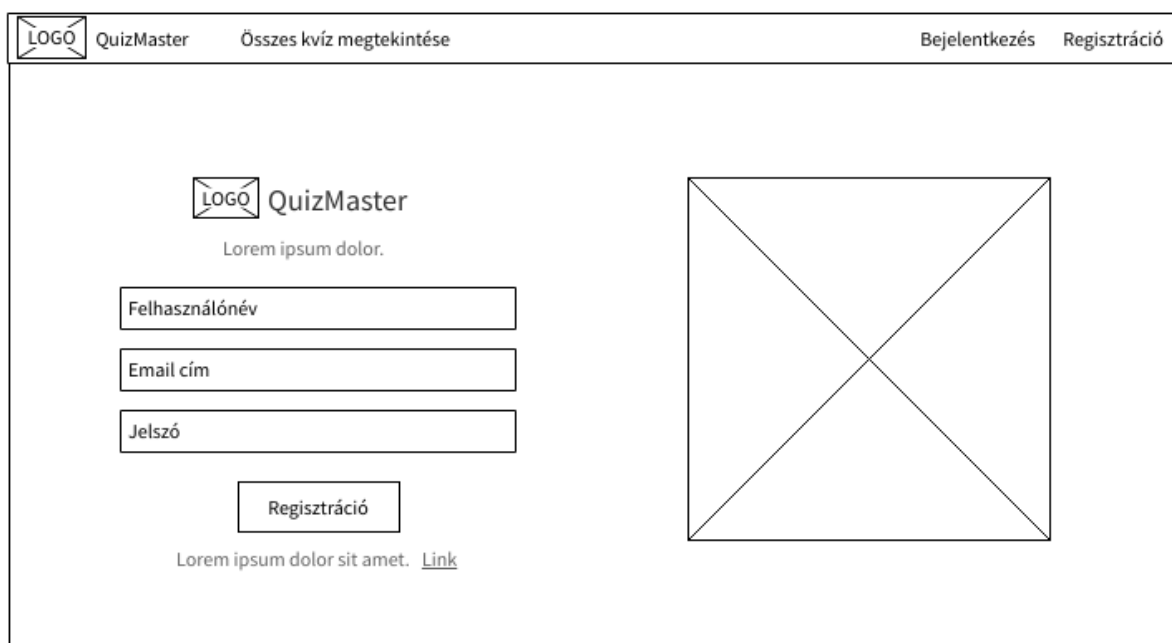
- **Drótvázterv**

A drótvázterv egy vizuális eszköz, amely segít az alkalmazás különböző felületeinek struktúráját és elrendezését megtervezni. Célja, hogy egyszerű, vázlatos módon bemutassa az egyes oldalak elrendezését, a fő funkciók helyét, valamint az elemek közötti kapcsolatokat. A drótváztervek lehetővé teszik a fejlesztők számára, hogy az alkalmazás felépítését és logikáját még a tényleges programozás előtt átlássák, így később elkerülhetők az elrendezéssel kapcsolatos problémák és félreértések.


Az applikációmhoz én azért készítettem drótváztervet, mert így előre meg tudtam tervezni a különböző oldalak felépítését, például a főoldalt, a kvizek listáját, a kvíz kitöltési felületet, és a ranglista oldalt. Ez segített abban, hogy az alkalmazás kezelőfelülete felhasználóbarát legyen, és az elemek logikus elrendezésben kerüljenek megjelenítésre, így megkönnyítve a fejlesztést.




43. ábra: Drótvázterv – Kezdőlap



44. ábra: Drótvázterv - Regisztráció


QuizMaster
Összes kvíz megtekintése

Bejelentkezés
Regisztráció


QuizMaster

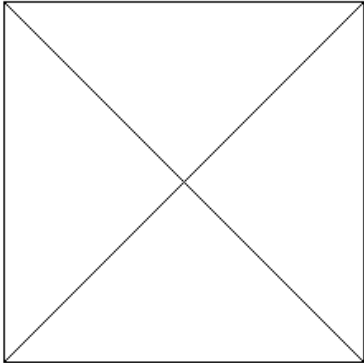
Lorem ipsum dolor.

Felhasználónév

Jelszó

Bejelentkezés

Lorem ipsum dolor sit amet. [Link](#)



45. ábra: Drótvázterv - Bejelentkezés


QuizMaster
Összes kvíz megtekintése

Bejelentkezés
Regisztráció

Kvízek


Kvíz címe
Lorem ipsum dolor sit amet.

Eredmények megtekintése


Kvíz címe
Lorem ipsum dolor sit amet.

Eredmények megtekintése


Kvíz címe
Lorem ipsum dolor sit amet.

Eredmények megtekintése


Kvíz címe
Lorem ipsum dolor sit amet.

Eredmények megtekintése

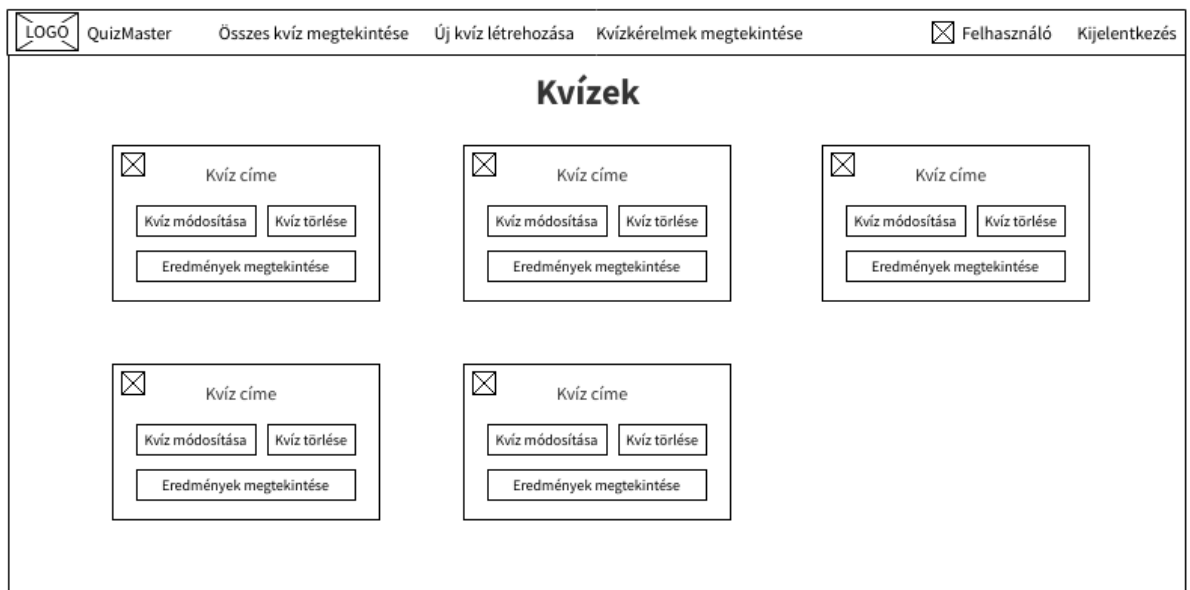

Kvíz címe
Lorem ipsum dolor sit amet.

Eredmények megtekintése

46. ábra: Drótvázterv – Összes kvíz oldala (nem bejelentkezett felhasználó)



47. ábra: Drótvázterv – Összes kvíz oldala (bejelentkezett felhasználó)



48. ábra: Drótvázterv – Összes kvíz oldala (adminisztrátor)


[QuizMaster](#)
[Összes kvíz megtekintése](#)
[Kvízkérelmek megtekintése](#)
[Kvízkérelem leadása](#)

☒ [Felhasználó](#)
[Kijelentkezés](#)

..... Kvíz kitöltése

<

Lorem ipsum dolor sit amet?

☒ Radio 1
 ☐ Radio 2
 ☐ Radio 3
 ☐ Radio 4

>

1 / 10

Lorem ipsum dolor sit amet.

Kvíz beküldése

49. ábra: Drótvázterv – Kvíz kitöltése oldal (alap felhasználó)


[QuizMaster](#)
[Összes kvíz megtekintése](#)
[Kvízkérelmek megtekintése](#)
[Kvízkérelem leadása](#)

☒ [Felhasználó](#)
[Kijelentkezés](#)

Előző kitöltés válaszai

Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet?

☒ Radio 1
 ☐ Radio 2
 ☐ Radio 3
 ☐ Radio 4

>

1 / 10

Eredmények megtekintése

Újra kitöltöm

50. ábra: Drótvázterv – Előző kitöltés válaszai oldal (alap felhasználó)

LOGO

QuizMaster

Összes kvíz megtekintése

Új kvíz létrehozása

Kvízkérelmek megtekintése

☒ Felhasználó

Kijelentkezés

Új Kvíz létrehozása

Label

Kvíz címe

Label

Kérdések száma

Lorem ipsum

Kérdés szövege

Opció 1

Opció 2

Opció 3

Opció 4

☐ Radio 1

☒ Radio 2

☐ Radio 3

☐ Radio 4

Lorem ipsum

LOGO

QuizMaster

Összes kvíz megtekintése

Új kvíz létrehozása

Kvízkérélmek megtekintése

☒ Felhasználó

☐ Kijelentkezés

..... Kvíz szerkesztése

Label

Kvíz címe

Label

Kérdések száma

Lorem ipsum

Kérdés szövege

Opció 1

Opció 2

Opció 3

Opció 4

☐ Radio 1

☒ Radio 2

☐ Radio 3

☐ Radio 4

Lorem ipsum

LOGO	QuizMaster	Összes kvíz megtekintése	Kvízkérelmek megtekintése	Kvízkérelem leadása	<input checked="" type="checkbox"/> Felhasználó	Kijelentkezés
------	------------	--------------------------	---------------------------	---------------------	---	---------------

..... Kvíz eredményei			
Rank	Felhasználó	Pontszám	Kitöltési idő
1	lorem	ipsum	dolor
2	sit	amet	consectetur
3	adipiscing	elit	nunc

53. ábra: Drótvázterv – Kvíz ranglista oldal

LOGO	QuizMaster	Összes kvíz megtekintése	Kvízkérelmek megtekintése	Kvízkérelem leadása	<input checked="" type="checkbox"/> Felhasználó	Kijelentkezés
------	------------	--------------------------	---------------------------	---------------------	---	---------------

Kvízkérelmek

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis.

Kvíz címe

Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Tetszik

Kvíz címe

Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Tetszik

Kvíz címe

Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Tetszik

54. ábra: Drótvázterv – Kvízkérelmek oldal (alapfelhasználó)

LOGO	QuizMaster	Összes kvíz megtekintése	Új kvíz létrehozása	Kvízkérelmek megtekintése	<input checked="" type="checkbox"/> Felhasználó	Kijelentkezés
------	------------	--------------------------	---------------------	---------------------------	---	---------------

Kvízkérelmek

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis.

Kvíz címe

Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Létrehozom a kvízt

Kérelem törlése

Kvíz címe

Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Létrehozom a kvízt

Kérelem törlése

Kvíz címe

Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Létrehozom a kvízt

Kérelem törlése

55. ábra: Drótvázterv – Kvízkérelmek oldal (adminisztrátor)

LOGO	QuizMaster	Összes kvíz megtekintése	Kvízkérelmek megtekintése	Kvízkérelem leadása	<input checked="" type="checkbox"/> Felhasználó	Kijelentkezés
------	------------	--------------------------	---------------------------	---------------------	---	---------------

Kvízkérelem leadása

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Label

Kvíz témája

Kérelem leadása

56. ábra: Drótvázterv – Kvízkérelem leadása oldal (alap felhasználó)

3.3.2. Felhasznált technológiák

A technológiai stack gondos megválasztása kulcsfontosságú az alkalmazás fejlesztéséhez és karbantartásához. A megfelelő eszközök és keretrendszerek alkalmazásával lehetőségem volt hatékonyan megvalósítani az alkalmazás különféle funkcióit, biztosítva a skálázhatóságot, megbízhatóságot és teljesítményt. Az alábbiakban bemutatom a szerveroldali technológiákat, amelyeket a kvíz alkalmazásom fejlesztéséhez használtam.

Szerveroldal – Node.js és Express

A backend oldalt Node.js és Express keretrendszerben készítettem, amely gyors és egyszerű megoldást kínál a RESTful API-k megvalósításához. Az Express minimalista felépítése lehetővé tette a könnyen testreszabható és gyorsan fejleszthető végpontok létrehozását, amelyek az alkalmazás különböző funkcióit szolgálják.

Adatkezelés és hitelesítés

- **bcrypt:** A bcrypt könyvtárat a jelszavak biztonságos tárolásához használtam. Ez a titkosítási algoritmus biztosítja, hogy a jelszavak erős védelmet kapjanak az illetéktelen hozzáférésekkel szemben, hozzájárulva ezzel a felhasználói adatok biztonságához.
- **jsonwebtoken:** A jsonwebtoken könyvtárral kezeltem a felhasználók hitelesítését és jogosultságainak ellenőrzését. A JWT tokenek segítségével a bejelentkezett felhasználók különböző szintű hozzáféréseket kaptak az alkalmazás funkcióihoz, például a kvizek eléréséhez vagy az adminisztrációs felület használatához.
- **mongoose:** Az alkalmazás adatbázis-kezelését a MongoDB-vel és a mongoose könyvtárral valósítottam meg. A mongoose lehetővé teszi a MongoDB-sémák és modellek egyszerű létrehozását és kezelését, ami nagymértékben megkönnyítette a kvíz, felhasználói és eredményadatok tárolását és lekérdezését.

Middleware és API fejlesztés

- **express:** Az Express a szerveroldali alapot képezte, és biztosította az alkalmazás RESTful szolgáltatásait. Az express keretrendszer rugalmas struktúrát nyújtott, amelyen keresztül könnyedén kezelhettem a kvízhez és felhasználói funkciókhoz tartozó HTTP-kéréseket.

- **cors:** A cors csomaggal engedélyeztem, hogy a frontend és a backend különböző domainekről is kommunikálhasson egymással, biztosítva ezzel a biztonságos adatáramlást és a felhasználói élmény zavartalan működését.

Tesztelés és fejlesztési eszközök

- **Mocha és Chai:** A Mocha tesztelési keretrendszert és a Chai asszertációs könyvtárat az egység- és integrációs tesztek futtatására használtam. Ezek az eszközök biztosították, hogy az egyes funkciók megfelelően működjenek, és a végpontok helyesen válaszoljanak a különböző kérésekre.
- **supertest:** A Supertest segítségével HTTP-kéréseket küldtem a szerver végpontjaira, így ellenőriztem az Express API megfelelő működését. Ez a csomag egyszerűvé és hatékonyá tette az API tesztelését különböző scenáriókban.
- **sinon:** A Sinon könyvtárat a függvényhívások és függőségek szimulálására használtam a tesztelés során. Ez segített a szerveroldali logika pontosabb és átfogóbb tesztelésében.

Fejlesztési környezet optimalizálása

- **nodemon:** A nodemon csomag használatával az alkalmazás automatikusan újraindult, amikor a forráskód változott, így a fejlesztési folyamat gyorsabbá és kényelmesebbé vált.
- **@babel/preset-env és @babel/register:** A Babel segítségével modern JavaScript-szintaxist alkalmazhattam a szerveroldalon. Az @babel/preset-env és @babel/register csomagok biztosították, hogy az ES6+ kódok futtathatók legyenek Node.js környezetben, ami javította a kód olvashatóságát és karbantarthatóságát.

Ezekkel a technológiákkal és eszközökkel rugalmas és könnyen skálázható backend rendszert tudtam kialakítani, amely megfelelő teljesítményt nyújt és lehetővé teszi a biztonságos, felhasználóbarát alkalmazás működését.

Kliensoldal – React

Az alábbiakban bemutatom a kliensoldalon felhasznált technológiákat és eszközöket, amelyek lehetővé tették a kvízalkalmazás fejlesztését, az intuitív felhasználói felület kialakítását és a kód hatékony tesztelését.

Felhasználói felület és komponensek

- **React:** A React a kliensoldali fejlesztés alapja, amely lehetővé teszi az alkalmazás komponensekre bontását és a felhasználói interakciók gyors kezelését. A React könyvtár struktúrája átláthatóvá és könnyen karbantarthatóvá teszi az alkalmazás felületét.
- **React Bootstrap:** A React Bootstrap és a **Bootstrap** könyvtárak kombinációja segítségével könnyedén hoztam létre reszponzív, modern felhasználói felületeket anélkül, hogy manuálisan kellett volna stílusokat írnom. Ezáltal az alkalmazás könnyen kezelhető mobil és asztali környezetben egyaránt.
- **React Icons:** Ez a könyvtár különböző ikonokat biztosít, amelyeket a navigációs elemekhez és interaktív gombokhoz használtam. Az ikonok nemcsak vizuálisan javítják az alkalmazást, hanem könnyebben értelmezhetővé teszik a funkciókat.

Routing és jogosultságkezelés

- **React Router DOM:** A React Router DOM segítségével oldottam meg a navigációt az egyes oldalak között, mint például a kvízlistázás, a ranglisták megtekintése és a kvíz kitöltése. Ezzel a könyvtárral kezeltem a különböző útvonalak közötti váltást és az oldal tartalmának dinamikus betöltését.
- **JWT-decode:** A **jsonwebtoken**-nel létrehozott JSON Web Tokenek dekódolásához használtam a jwt-decode könyvtárat, amely a felhasználók jogosultságát és az aktuális állapotukat ellenőrizte. Ez kulcsszerepet játszik a jogosultsági szintek kezelésében, például az adminisztrátori funkciók elérésében.

Tesztelés és fejlesztési eszközök

- **Jest és Testing Library:** A **Jest** az alkalmazás tesztelési keretrendszere, amely lehetővé teszi az egység- és integrációs tesztek gyors végrehajtását. A **Testing Library** különböző kiegészítői, mint például a `@testing-library/react`, segítenek a felhasználói interakciók szimulálásában és az összetett komponensek tesztelésében.
- **Mocha és Chai:** További tesztelési keretrendszerek, amelyekkel részletesebb egység- és integrációs tesztek írtam az alkalmazáshoz. A Mocha és Chai lehetőséget biztosítanak az egyes modulok izolált tesztelésére és az alkalmazás működési logikájának validálására.
- **ESLint:** Az ESLint használatával folyamatosan biztosítottam a kódminőséget és a stílus konzisztenciáját. Az ESLint segít a hibák korai felismerésében és a legjobb gyakorlatok követésében, ami különösen hasznos volt a nagyobb komponensek fejlesztésekor.

Fejlesztési és build eszközök

- **Vite:** A Vite-t használtam a fejlesztési környezet gyors és hatékony kiszolgálásához. A Vite gyorsabb, mint a hagyományos build eszközök, mivel csak a szükséges modulokat tölti be a böngészőbe. Ezáltal az alkalmazás fejlesztése és tesztelése is gördülékenyebb, és gyorsabban reagál a változtatásokra.
- **Babel:** A Babel konfigurációval modern JavaScript kódot írhattam, miközben biztosítottam a visszafelé kompatibilitást. A Babel lehetővé tette az olyan új szintaxisok használatát, amelyek a projekt megvalósítását gyorsabbá és átláthatóbbá tették.

Ezek a technológiák és eszközök együttesen biztosították, hogy az alkalmazás hatékonyan és megbízhatóan működjön, és a felhasználók számára zökkenőmentes élményt nyújtson.

3.4. Megvalósítás

3.4.1. Alkalmazás felépítése

Az alkalmazás egy modern kvízkezelő rendszer, amely a felhasználóbarát funkciókra és a hatékony adatkezelésre fókuszál. A projekt fejlesztéséhez a MERN stack technológiát alkalmazzuk, amely magában foglalja a MongoDB adatbázist, az Express.js backend keretrendszert, a React frontend library-t, valamint a Node.js környezetet. Ezek a technológiák együttesen biztosítják az alkalmazás megbízhatóságát, rugalmasságát és skálázhatóságát.

Backend

Az alkalmazás szerveroldalát a Node.js és az Express.js alkotja, amely egy gyors és minimalista webes keretrendszer. A RESTful API-k lehetővé teszik az adatok hatékony kezelését, például a kvízek létrehozását, módosítását, kitöltését és a felhasználói eredmények tárolását. A szerveroldal **moduláris struktúrája** biztosítja, hogy az alkalmazás egyes komponensei (például hitelesítés, felhasználókezelés, kvízek és kérdések) elkülönítve legyenek implementálva. Ez megkönnyíti a kód olvashatóságát, új funkciók hozzáadását és a hibák izolálását.

A MongoDB NoSQL adatbázis felel az adatok tárolásáért, amely rugalmas sématervezést tesz lehetővé, illeszkedve a modern webes alkalmazások igényeihez. A hitelesítési folyamatot a JSON Web Token (JWT) technológiával valósítottuk meg, amely garantálja a felhasználók biztonságos bejelentkezését és a jogosultságok kezelését. Az adminisztrációs funkciók elkülönítése lehetővé teszi, hogy csak jogosult felhasználók hajthassanak végre kritikus műveleteket, például kvízek törlését vagy módosítását.

Frontend

A kliensoldali alkalmazást a React könyvtár segítségével valósítottuk meg, amely komponens-alapú architektúrájával egyszerűvé és áttekinthetővé teszi a fejlesztést. A Bootstrap és a React Bootstrap integrációja biztosítja az esztétikus és reszponzív felhasználói élményt. A felhasználók bejelentkezés után elérhetik a kvízeket, megtekinthetik a ranglistát, vagy új kvíz ötleteket küldhetnek be.

A felhasználói élményt gazdagítják a dinamikus felületek, például a kvíz-kitöltési folyamat során alkalmazott carousel megoldás. Ez intuitív és látványos módon vezeti végig a felhasználót a kérdéseken.

3.4.2. Modell és kapcsolatok

Szerveroldal

- **Hitelesítés – Autentikáció – Autorizáció**

A szerveroldali hitelesítési mechanizmus a felhasználók hozzáférési jogosultságainak ellenőrzésére és az alkalmazás biztonságának garantálására szolgál. Az alábbi elemekből épül fel:

- **User model**

A felhasználói adatok MongoDB-s kezelése a Mongoose segítségével történik.

Felépítés:

- username: Egyedi felhasználónév, minimum 3 karakter.
- email: Egyedi email cím, amely megfelel az általános email formátumnak.
- password: A felhasználó jelszava hash-elve, amely a bcrypt könyvtár segítségével kerül tárolásra.
- isAdmin: Boolean típusú mező, amely a felhasználó admin jogosultságát jelzi.

Funckiók:

- comparePassword: A felhasználó által megadott jelszó összehasonlítása a hash-elt jelszóval.
- pre('save'): Jelszó automatikus hash-elése mentés előtt.

- **Token alapú hitelesítés**

A felhasználók azonosítását JSON Web Token (JWT) segítségével valósítottuk meg, amely lehetővé teszi a szerver számára a kliens hitelesítését anélkül, hogy állapotot kellene tárolni.

JWT generálása

- Regisztráció és bejelentkezés során a felhasználók egy 24 órás érvényességi idővel rendelkező JWT-t kapnak.

- A token tartalmazza a felhasználó azonosítóját, felhasználónevét és adminisztrátori jogosultságát.
- A tokenek aláírásához használt titkos kulcs a szerver konfigurációs fájljában található.

○ **Jogosultságkezelés (Middleware-ek)**

A szerver oldalon két kulcsfontosságú middleware felelős a hozzáférés ellenőrzéséért:

requireAuth Middleware

- Feladata annak ellenőrzése, hogy a kérés tartalmaz-e érvényes token-t.
- Ha a token hiányzik, a kliens 401-es állapotkóddal és hibaüzenettel kap választ.
- Ha a token lejárt, vagy érvénytelen, a middleware egyértelmű hibaüzenettel utasítja el a hozzáférést.
- Érvényes token esetén az adatokat (felhasználó ID, név, jogosultságok) kibontja, és továbbadja a következő műveletnek.

requireAdmin Middleware

- Ez a middleware ellenőrzi, hogy a felhasználó rendelkezik-e adminisztrátori jogosultságokkal.
- Ha nem, a rendszer megtagadja a hozzáférést, és visszatér egy hibaüzenettel.
- Különösen hasznos az adminisztrátorok számára fenntartott műveletek, például új kvizek létrehozása vagy meglévők törlése esetén.

○ **Felhasználói regisztráció és bejelentkezés**

Regisztráció (POST /users/register)

- Az endpoint a felhasználó által megadott adatokat ellenőrzi, például:
 - Felhasználónév egyedisége.

- Érvényes email cím formátuma.
- Erős jelszó követelményei.
- Sikeres regisztráció esetén a felhasználó adatai bekerülnek az adatbázisba, és egy JWT token-t kap.

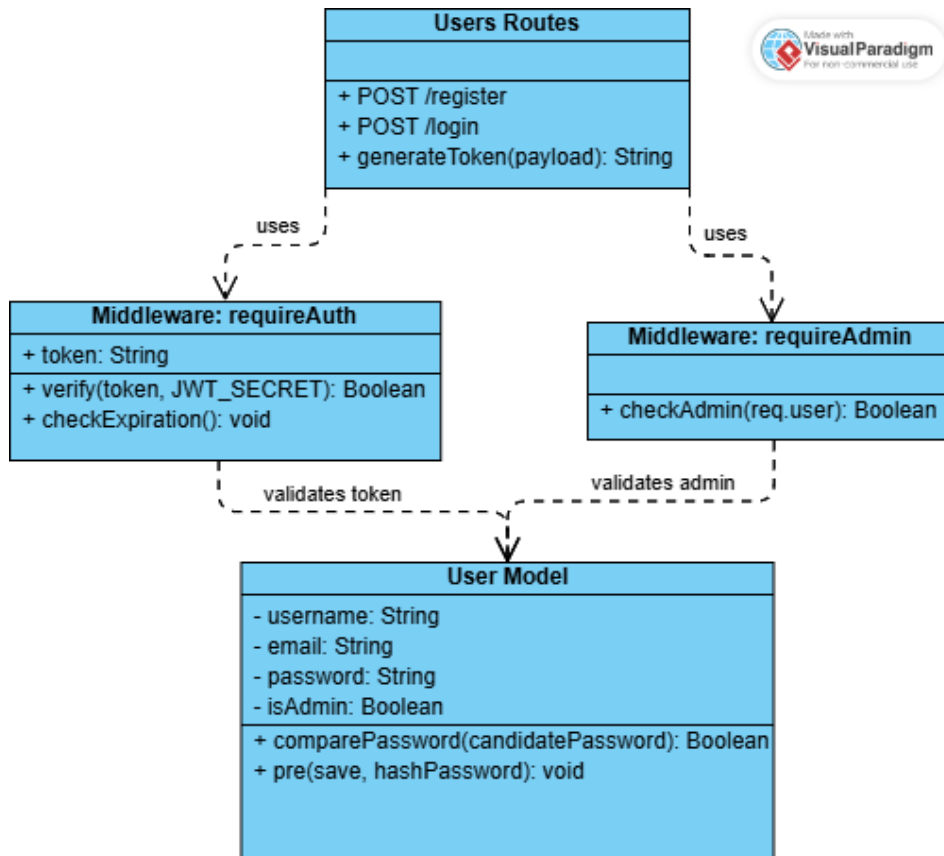
Bejelentkezés (POST /users/login)

- A felhasználó a bejelentkezési adatok (felhasználónév és jelszó) megadásával kap hozzáférést.
- A rendszer ellenőrzi a felhasználó meglétét, majd összeveti a megadott jelszót a tárolt hash-sel.
- Sikeres bejelentkezés után a rendszer generál egy új JWT-t, amely biztosítja a felhasználó hozzáférését az alkalmazáshoz.

○ Hibaüzenetek és visszajelzések

A hitelesítési folyamat során a rendszer részletes visszajelzést ad az esetleges hibákról, például:

- "Hozzáférés megtagadva, hiányzó token. Elvárt formátum: Bearer <token>."
- "A token lejárt, kérjük jelentkezzen be újra."
- "Érvénytelen token."
- "Hozzáférés megtagadva, csak adminisztrátorok számára elérhető."



57. ábra: Autentikáció – Autorizáció UML (szerveroldal)

• Modellek

A modellek az alkalmazás adatkezelésének alapkövei, amelyek az adatok szerkezetét, a közöttük lévő kapcsolatokat és az adatkezelési logikát határozzák meg. A modellek segítségével biztosítjuk, hogy az alkalmazásban használt adatok konzisztens módon legyenek tárolva és kezelve, valamint könnyen bővíthetők és módosíthatók legyenek.

A jelen projektben a modellek kialakítása a MongoDB NoSQL adatbázis sématervezési lehetőségein alapul, amely rugalmas és hatékony megoldást kínál a modern webes alkalmazások igényeihez. A Mongoose könyvtár segítségével definiált modellek nemcsak az adatok tárolását, hanem azok validációját és logikai feldolgozását is lehetővé teszik.

○ User model

A **User** modell a felhasználók adatainak kezeléséért felelős az alkalmazásban. A modell a **Mongoose** segítségével került definiálásra, amely egy MongoDB-hez készült objektum-dokumentum modellező könyvtár. A modell célja, hogy biztosítsa a felhasználói adatok konzisztens és biztonságos tárolását, valamint olyan funkciókat nyújtson, amelyek az autentikációt és az adatok validációját segítik.

Adatszerkezet

A modell által definiált adatmezők és azok szabályai:

- **username (String):**
 - Egyedi, kötelezően megadandó mező.
 - Minimum 3 karakter hosszú.
- **email (String):**
 - Egyedi, kötelező mező.
 - Az általános email cím formátumnak megfelelő reguláris kifejezéssel validált.
- **password (String):**
 - Kötelezően megadandó mező.
 - A jelszó hash-elt formában kerül tárolásra, biztosítva a felhasználói adatok biztonságát.
- **isAdmin (Boolean):**
 - Alapértelmezett értéke: false.
 - Jelzi, hogy a felhasználó rendelkezik-e adminisztrátori jogosultságokkal.

Funkciók

A modell az adatok kezelésére és biztonságos tárolására az alábbi funkciókat tartalmazza:

- **Jelszó hash-elése mentés előtt (pre('save')):**
 - Minden alkalommal, amikor egy felhasználó adatainak mentése történik, a jelszó automatikusan hash-elésre kerül a **bcrypt** könyvtár segítségével.
 - A folyamat biztosítja, hogy a jelszó titkosítva legyen az adatbázisban.
- **Jelszó összehasonlítása (comparePassword):**
 - Egy aszinkron metódus, amely lehetővé teszi a felhasználó által megadott jelszó összehasonlítását az adatbázisban tárolt hash-elt jelszóval.
 - A **bcrypt.compare()** metódust használja a validációhoz.

○ **Option model**

Az **Option** modell a kvízkérdésekhez tartozó válaszlehetőségek tárolására és kezelésére szolgál az alkalmazásban. A modell a **Mongoose** segítségével került kialakításra, és biztosítja, hogy minden válaszlehetőség konzisztens formában kerüljön tárolásra a MongoDB adatbázisban.

Adatszerkezet

A modell által definiált mezők és azok tulajdonságai:

- **optionText (String):**
 - Kötelező mező, amely a válaszlehetőség szövegét tartalmazza.
 - Ez a mező határozza meg, hogy a felhasználó mit lát a válaszlehetőségként.
- **isCorrect (Boolean):**
 - Kötelező mező, amely azt jelzi, hogy a válaszlehetőség helyes-e.
 - A helyes válaszok az érték **true**, a helytelenek az érték **false** beállítást kapják.

Kapcsolatok

Az **Option** modell önállóan definiált, de szorosan kapcsolódik a kérdéseket reprezentáló modellhez (**Question**). Az **Option** dokumentumok általában a kvízkérdések részeként kerülnek tárolásra, és a kérdésekkel együtt kerülnek betöltésre.

○ Question model

A **Question modell** a kvízkérdések tárolására és kezelésére szolgál az alkalmazásban. A modell a Mongoose segítségével került kialakításra, és biztosítja, hogy minden kérdés és az ahhoz tartozó válaszlehetőségek logikusan és konzisztensen tárolódjanak a MongoDB adatbázisban.

Adatszerkezet

A modell által definiált mezők és azok tulajdonságai:

- **optionText (String):**
 - Kötelező mező, amely a válaszlehetőség szövegét tartalmazza.
 - Ez a mező határozza meg, hogy a felhasználó mit lát a válaszlehetőségként.
- **isCorrect (Boolean):**
 - Kötelező mező, amely azt jelzi, hogy a válaszlehetőség helyes-e.
 - A helyes válaszok az érték **true**, a helytelenek az érték **false** beállítást kapják.

Kapcsolatok

A **Question modell** szoros kapcsolatban áll az **Option** modellel. Az egyes kérdésekhez tartozó válaszlehetőségek azonosítói az options mezőben kerülnek tárolásra. Ez a referencia-alapú kapcsolat lehetővé teszi a kérdések és válaszlehetőségek közötti logikai összekapcsolást.

○ Quiz model

A **Quiz modell** az alkalmazásban található kvízek tárolására és kezelésére szolgál. Ez a modell felelős a kvízek alapvető információinak és az ahhoz tartozó kérdések adatainak nyilvántartásáért. A modell a Mongoose segítségével lett kialakítva, amely lehetővé teszi a MongoDB adatbázis hatékony kezelését.

Adatszerkezet

A modell által definiált mezők és azok tulajdonságai:

- **title (String):**
 - Kötelező mező, amely a kvíz címét tartalmazza.
 - A kvíz egyedi megkülönböztetéséhez szolgál, és a felhasználó számára megjelenített nevet határozza meg.
- **questions (Array of ObjectId):**
 - Egy tömb, amely az adott kvízhez tartozó kérdések azonosítóit tartalmazza.
 - Minden elem a MongoDB-ben található **Question** dokumentumra mutat, a ref: 'Question' beállítással.
 - Ez a referencia-alapú kapcsolat biztosítja, hogy a kérdések dinamikusan lekérhetőek legyenek a kvízzel együtt.
- **questionCount (Number):**
 - Kötelező mező, amely a kvízben található kérdések számát jelöli.
 - Az értéke csak az előre meghatározott számok (5, 10, 20) közül lehet, biztosítva a kvíz egységes működését.

Kapcsolatok

A **Quiz modell** szorosan kapcsolódik a **Question** modellekhez. Az egyes kvizekhez tartozó kérdések az questions mezőben tárolt hivatkozások segítségével érhetők el. Ez a kapcsolat lehetővé teszi, hogy a kvíz kérdései dinamikusan betölthetők legyenek a Mongoose populate módszerével.

○ Result model

A **Result modell** az alkalmazásban található felhasználói eredmények tárolására és kezelésére szolgál. Ez a modell rögzíti a felhasználók kvízhez kapcsolódó teljesítményét, mint például az elért pontszámokat, adott válaszokat, és a kvíz kitöltésének idejét. A modell a Mongoose segítségével került definiálásra, amely lehetővé teszi a MongoDB adatbázis hatékony kezelését.

Adatszerkezet

A modell által definiált mezők és azok tulajdonságai:

- **user (ObjectId):**
 - A kvízt kitöltő felhasználóra hivatkozó kötelező mező.
 - A **User** modell egy dokumentumára mutat a ref: 'User' beállítással.
- **quiz (ObjectId):**
 - Az adott eredményhez tartozó kvíz azonosítója.
 - A **Quiz** modell egy dokumentumára mutat a ref: 'Quiz' beállítással.
 - Ez a mező kötelező, hiszen minden eredmény egy adott kvízhez kapcsolódik.
- **score (Number):**
 - A felhasználó által elért pontszámot tárolja.
 - Kötelező mező, amely számszerűsíti a kvíz teljesítményét.

- **answers (Map of ObjectId):**
 - A kvíz során adott válaszokat tartalmazza, ahol a kulcs a kérdés azonosítója, az érték pedig a választott **Option** azonosítója.
- **completionTime (Number):**
 - A kvíz kitöltéséhez szükséges idő másodpercekben mérve.
 - Kötelező mező, amely a teljesítmény egy további aspektusát dokumentálja.
- **selectedQuestions (Array of ObjectId):**
 - A felhasználónak a kvíz során megjelenített kérdéseinek azonosítóit tartalmazza.
 - A **Question** modellekre mutató hivatkozásokat tartalmaz a ref: 'Question' beállítással.
- **orderedOptions (Map of Array of ObjectId):**
 - A kérdésekhez tartozó válaszlehetőségek sorrendjét rögzíti.
 - A kulcs a kérdés azonosítója, az érték pedig az adott kérdéshez tartozó **Option** azonosítóinak rendezett tömbje.

Kapcsolatok

A **Result** modell kulcsszerepet játszik az alkalmazásban, mivel több másik modellel is kapcsolatban áll:

- **User:**
 - Rögzíti, hogy mely felhasználó érte el az adott eredményt.
- **Quiz:**
 - Dokumentálja, hogy az adott eredmény melyik kvízhez tartozik.

- **Question és Option:**

- Az eredmények a kérdésekre adott válaszokat és az opciók sorrendjét is rögzítik.

- **Request model**

A **Request modell** az alkalmazásban található kérések kezelésére szolgál. Ez a modell lehetővé teszi a felhasználók számára, hogy új kvízekre vonatkozó ötleteiket benyújtsák, és a közösség által szavazás útján támogassák azokat. A modell a Mongoose segítségével került kialakításra, amely a MongoDB adatbázis hatékony kezelését biztosítja.

Adatszerkezet

A modell által definiált mezők és azok tulajdonságai:

- **title (String):**

- Kötelező mező, amely a kérelem címét tartalmazza.
- Leírja, hogy milyen kvízötletre vonatkozik a kérelem.

- **user (ObjectId):**

- A kérést benyújtó felhasználó azonosítója.
- A **User** modellre mutató hivatkozás (ref: 'User').
- Kötelező mező, mivel minden kéréshez szükséges egy beküldő felhasználó.

- **upvotes (Array of ObjectId):**

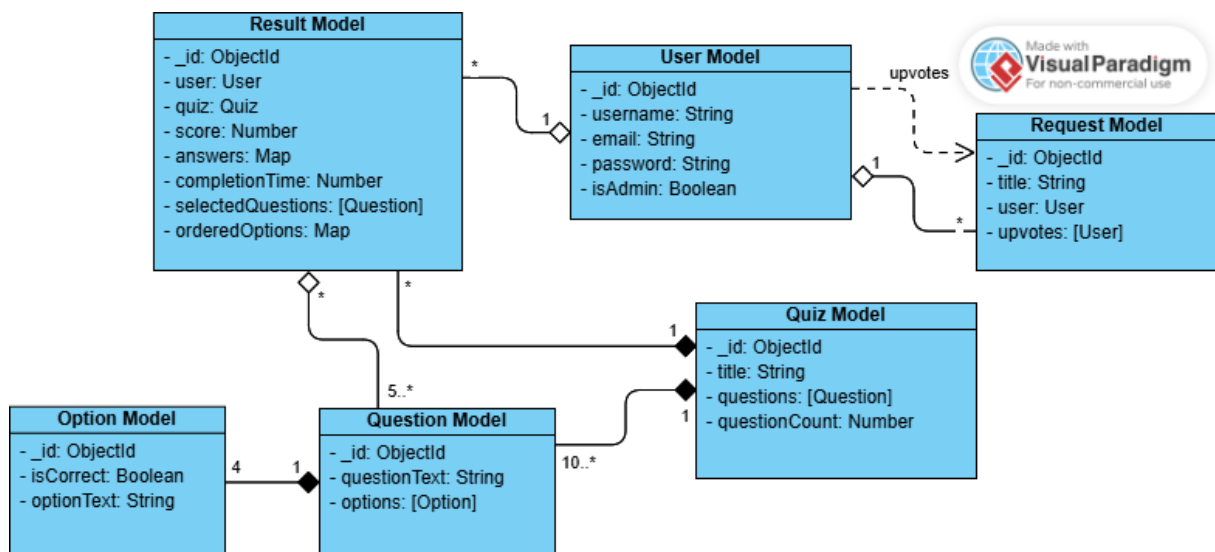
- Az adott kérést támogató felhasználók azonosítóit tartalmazza.
- A tömb elemei a **User** modell dokumentumaira mutató hivatkozások.
- Ez a mező teszi lehetővé a közösségi szavazás funkcióját, amely alapján rangsorolhatók a kérések.

Kapcsolatok

A **Request modell** az alkalmazás **User** modelljéhez kapcsolódik:

- **user mező:**
 - Ez a mező azonosítja azt a felhasználót, aki a kérelmet benyújtotta, és a **User** modellre hivatkozik.
- **upvotes mező:**
 - Ez a mező a kérést támogató felhasználók azonosítóit tárolja, szintén a **User** modell dokumentumaira mutató hivatkozások formájában.

Az **_id** attribútum minden MongoDB dokumentumban automatikusan létrejön és egyedi azonosítónak szolgál. Ez az attribútum alapértelmezetten egy **ObjectId** típusú érték, amely biztosítja a dokumentum egyediségét az adatbázison belül. A Mongoose segítségével definiált modellek esetében az **_id** automatikusan hozzáadódik minden objektumhoz, hacsak explicit módon nem tiltjuk le. Az ObjectId egy 12 bájtos hexadecimális azonosító, amely tartalmaz információt például a létrehozás idejéről. Ez az attribútum elengedhetetlen a különböző modellek közötti kapcsolatok megvalósításához, mivel hivatkozások alapjául szolgál.



58. ábra: Modellek kapcsolatai UML (szerveroldali)

- **Útvonalak és Végpontok**

Az alkalmazás backendje RESTful API-k segítségével valósítja meg a frontend és az adatbázis közötti kommunikációt. Az útvonalak és végpontok felelősek a felhasználók, kvízek, kérdések, válaszok és kérések kezeléséért. A Node.js alapú Express.js keretrendszer használatával az API végpontok könnyen bővíthetők, testreszabhatók, és a különböző logikai rétegek (middleware-ek, modellek) között világos kapcsolatot teremtenek.

Ezek az útvonalak különböző feladatokat látnak el, például:

- Felhasználók regisztrációja és bejelentkezése.
- Kvízek létrehozása, módosítása, törlése és megtekintése.
- Kérdések és válaszlehetőségek kezelése.
- Eredmények tárolása és lekérdezése.
- Új kvízötletekhez kapcsolódó kérelmek létrehozása és szavazása.

Az API a middleware-ek segítségével hitelesíti a felhasználói kéréseket és adminisztrátori jogosultságokat követel meg bizonyos funkciók eléréséhez. Minden végpont pontosan dokumentált, és az alkalmazás következetes válaszokat ad a felhasználói hibák vagy rendszerproblémák esetén.

- **User Routes**

A **User Routes** az alkalmazás felhasználóinak regisztrációját és bejelentkezését kezeli. A végpontok biztosítják az adatok validálását és a JSON Web Token (JWT) alapú hitelesítést, a MongoDB adatbázissal pedig a Mongoose modellen keresztül kommunikálnak.

Végpontok és funkcióik

1. Új felhasználó regisztrálása

- **Útvonal:** /register (POST)

- **Funkció:** Lehetővé teszi új felhasználók regisztrációját, ellenőrzi az email, felhasználónév és jelszó formátumát, valamint biztosítja azok egyediségét az adatbázisban.

2. Felhasználó bejelentkezése

- **Útvonal:** /login (POST)
- **Funkció:** Hitelesíti a felhasználót a megadott felhasználónév és jelszó alapján, majd JWT-t generál, amely a későbbi hozzáférésekhez szükséges.

Biztonság

- **JWT alapú hitelesítés:** A regisztráció és bejelentkezés után 24 órás érvényességű token generálódik, amely tartalmazza a felhasználó azonosítóját, nevét, emailjét és adminisztrátori jogosultságát.
- **Jelszó hash-elése:** A bcrypt könyvtár segítségével a jelszavak hash-elve tárolódnak az adatbázisban, biztosítva a felhasználói adatok biztonságát.

Kapcsolatok

- **User modell:** A végpontok a User modellt használják az adatok tárolására és ellenőrzésére.
- **JWT middleware:** A bejelentkezés során generált token hitelesítést biztosít a későbbi hozzáférésekhez.

○ Option Routes

Az **Option Routes** felelős a kvízkérdésekhez tartozó válaszopciók kezeléséért, ideértve az opciók létrehozását, lekérdezését, frissítését és törlését. A végpontok adminisztrátori jogosultságokhoz kötöttek, kivéve az opciók lekérdezését, amely bárki számára elérhető. Az útvonalak a Mongoose segítségével kommunikálnak a MongoDB-vel, és biztosítják az adatok validálását.

Végpontok és funkciók

1. Új válaszopció hozzáadása egy kérdéshez

- **Útvonal:** /add (POST)
- **Funkció:** Hozzáad egy új opciót egy megadott kérdéshez. Az opciók a questionId, optionText és isCorrect paramétereken keresztül kerülnek megadásra.

2. Válaszopció lekérése ID alapján

- **Útvonal:** /:id (GET)
- **Funkció:** Lekér egy adott opciót azonosító alapján. Ha a felhasználó nem admin, az isCorrect mező rejtett marad.

3. Válaszopció módosítása ID alapján

- **Útvonal:** /:id (PUT)
- **Funkció:** Frissíti egy meglévő opció szövegét és helyességi státuszát adminisztrátori jogosultsággal.

4. Válaszopció törlése ID alapján

- **Útvonal:** /:id (DELETE)
- **Funkció:** Törli az adott opciót és automatikusan eltávolítja azt a kapcsolódó kérdésekből is.

Biztonság

- **Adminisztrátori jogosultságok:** Az opciók létrehozása, frissítése és törlése kizárólag adminisztrátorok számára érhető el.
- **Adatvédelem:** Nem admin felhasználók esetén az isCorrect mező elrejtésre kerül a válaszok lekérésekor.

Kapcsolatok

- **Question modell:** Az opciók a kérdésekhez kapcsolódnak, és törlés esetén a kérdésektől is eltávolításra kerülnek.
- **Middleware-ek:** A végpontok a requireAuth és requireAdmin middleware-eket használják az engedélyek ellenőrzéséhez.

○ Question Routes

A **Question Routes** kezeli a kvízkérdések létrehozását, módosítását, törlését, valamint azok lekérdezését. Az útvonalak adminisztrátori jogosultságokhoz kötöttek, kivéve a kérdések szövegének nyilvános lekérdezését. A végpontok a MongoDB adatbázissal a Mongoose modellen keresztül kommunikálnak, biztosítva a kérdésekhez tartozó válaszopciók kezelését is.

Végpontok és funkcióik

1. Új kérdés hozzáadása

- **Útvonal:** /add (POST)
- **Funkció:** Lehetővé teszi új kérdés létrehozását opcionális válaszopciókkal. A kérdés szövegét és az opciókat a kérés törzsében kell megadni.

2. Kérdés lekérése ID alapján

- **Útvonal:** /:id (GET)
- **Funkció:** Lekér egy adott kérdést az azonosítója alapján, a hozzá tartozó opciókkal együtt. Nem admin felhasználók esetén az opciók isCorrect mezője elrejtésre kerül.

3. Összes kérdés lekérése

- **Útvonal:** / (GET)

- **Funkció:** Lekéri az összes kérdés szövegét az opciók nélkül. Ez a végpont bárki számára elérhető.

4. Kérdés módosítása ID alapján

- **Útvonal:** /:id (PUT)
- **Funkció:** Módosítja az adott kérdés szövegét és opcionálisan a hozzá tartozó opciókat. Az új opciók a kérdés törzsében kerülnek megadásra.

5. Kérdés törlése ID alapján

- **Útvonal:** /:id (DELETE)
- **Funkció:** Törli az adott kérdést és automatikusan eltávolítja a hozzá tartozó opciókat is.

Biztonság

- **Adminisztrátori jogosultságok:** A kérdések létrehozása, módosítása és törlése kizárólag adminisztrátorok számára érhető el.
- **Adatvédelem:** Nem admin felhasználók számára az isCorrect mező elrejtésre kerül az opciók lekérésekor.

Kapcsolatok

- **Option modell:** A kérdésekhez tartozó opciók kezeléséhez szorosan kapcsolódik. Opciók törlésekor automatikusan frissíti az érintett kérdéseket.
- **Middleware-ek:** A végpontok a requireAuth és requireAdmin middleware-eket használják a jogosultságok ellenőrzésére.

○ Quiz Routes

A **Quiz Routes** a kvízek kezelését biztosítja az alkalmazásban, lehetővé téve a kvízek létrehozását, lekérdezését, módosítását, törlését, beküldését, és az eredmények feldolgozását. Az útvonalak az adminisztrátori jogosultságokkal és a bejelentkezett felhasználók által elérhető funkciókat egyaránt tartalmazzák.

Végpontok és funkciók

1. Új kvíz létrehozása

- **Útvonal:** /add (POST)

- **Funkció:**

Lehetővé teszi új kvíz létrehozását adminisztrátorok számára. A kvíz tartalmazhat kérdéseket és válaszopciókat, amelyek validációkon mennek keresztül (pl. duplikációk ellenőrzése).

2. Kvíz lekérése azonosító alapján

- **Útvonal:** /:id (GET)

- **Funkció:**

Lekéri egy kvíz adatait, beleértve a kérdéseket és a válaszopciókat. Csak bejelentkezett felhasználók számára elérhető.

3. Összes kvíz lekérése

- **Útvonal:** / (GET)

- **Funkció:**

Visszaadja az összes kvíz címét, kérdések és válaszopciók nélkül. Bárki számára elérhető.

4. Kvíz módosítása

- **Útvonal:** /:id (PUT)

- **Funkció:**

Lehetővé teszi egy meglévő kvíz módosítását adminisztrátorok számára, beleértve a cím, kérdések és válaszok frissítését.

5. Kvíz törlése

- **Útvonal:** /:id (DELETE)

- **Funkció:**

Törli a kvízt, valamint a hozzá tartozó kérdéseket és opciókat. Az ehhez kapcsolódó eredményeket is törli az adatbázisból. Csak adminisztrátorok számára elérhető.

6. Kvíz beküldése és eredmény feldolgozása

- **Útvonal:** /:id/submit (POST)

- **Funkció:**

Lehetővé teszi a felhasználóknak egy kvíz beküldését, az eredmények feldolgozását, és az adatbázisban történő tárolását. Frissíti a korábbi eredményeket, ha az új eredmény jobb.

7. Kitöltöttség ellenőrzése

- **Útvonal:** /:id/completed (GET)

- **Funkció:**

Ellenőrzi, hogy a bejelentkezett felhasználó már kitöltötte-e a kvízt, és visszaadja az előző válaszokat, ha rendelkezésre állnak.

Biztonság:

- **Adminisztrátori jogosultságok:** Az új kvizek létrehozása, módosítása és törlése kizárólag adminisztrátorok számára engedélyezett.
- **Hitelesítés:** A bejelentkezés JWT alapú hitelesítéssel valósul meg, amely biztosítja az útvonalak védelmét.

Kapcsolatok:

- **Question modell:** A kvizek kérdésekhez kapcsolódnak, és törléskor a kapcsolódó kérdések automatikusan eltávolításra kerülnek.
- **Option modell:** A kérdésekhez tartozó opciók a kvizek törlése során szintén törlődnek.
- **Result modell:** A kvizek eredményeihez kapcsolódó adatokat a rendszer automatikusan eltávolítja, ha a kvíz törlésre kerül.

○ Result Routes

A **Result Routes** az alkalmazásban a felhasználói eredmények kezelését biztosítja. Ezek az útvonalak lehetővé teszik az eredmények létrehozását, valamint azok lekérdezését egy adott felhasználóhoz vagy kvízhez. Az útvonalak a megfelelő hozzáférési jogosultságok ellenőrzésével biztosítják az adatok biztonságos kezelését.

Végpontok és funkciók

1. Új eredmény létrehozása egy adott kvízhez

- **Útvonal:** /quiz/:quizId (POST)
- **Funkció:**
Lehetővé teszi egy bejelentkezett felhasználó számára, hogy egy adott kvízhez új eredményt adjon hozzá. A pontszám (score) és a kitöltési idő (completionTime) megadása kötelező.

2. Eredmények lekérése egy adott felhasználóhoz

- **Útvonal:** /user/:userId (GET)
- **Funkció:**
Lehetővé teszi egy felhasználó számára, hogy megtekintse a saját eredményeit. Az adminisztrátorok más felhasználók eredményeihez is hozzáférhetnek.

3. Eredmények lekérése egy adott kvízhez

- **Útvonal:** /quiz/:quizId (GET)
- **Funkció:**
Visszaadja egy adott kvízhez tartozó összes eredményt. A válasz tartalmazza a felhasználó adatait (felhasználónév) és a kvíz címét.

Biztonság:

- **requireAuth middleware:** A bejelentkezési hitelesítés a requireAuth middleware segítségével biztosított, amely védi az eredményekhez kapcsolódó végpontokat.
- **Hozzáférési szintek:** Az adminisztrátorok hozzáférhetnek minden eredményhez, míg a bejelentkezett felhasználók kizárólag a saját eredményeiket érhetik el.

Kapcsolatok:

- **User modell:** Az eredmények kapcsolódnak a felhasználókhoz, lehetővé téve az egyéni eredmények kezelését és lekérdezését.
- **Quiz modell:** Az eredmények a kvizekhez kapcsolódnak, biztosítva az eredmények és a hozzájuk tartozó kvizek közötti összefüggések kezelését.

○ Request Routes

A **Request Routes** az alkalmazásban a kvízötletek és felhasználói kérések kezelésére szolgál. Az útvonalak lehetővé teszik új kérések létrehozását, a meglévő kérések lekérdezését, valamint a kérések támogatását vagy törlését. Ezek a funkciók biztosítják a felhasználók által beküldött ötletek nyomon követését és rangsorolását.

Végpontok és funkciók

1. Új kérés hozzáadása

- **Útvonal:** /add (POST)

- **Funkció:**

Lehetővé teszi a bejelentkezett felhasználók számára, hogy új kvízötletet nyújtsanak be. Az útvonal ellenőrzi a cím (title) formátumát és egyediségét.

2. Összes kérdés lekérése

- **Útvonal:** / (GET)

- **Funkció:**

Visszaadja az összes beküldött kérdést, beleértve a beküldő felhasználó nevét. Bárki számára elérhető.

3. Upvote hozzáadása egy kérdéshez

- **Útvonal:** /:id/upvote (PUT)

- **Funkció:**

Lehetővé teszi a bejelentkezett felhasználók számára, hogy támogassanak egy kérelmet. A felhasználó nem szavazhat a saját kérdésére, és egy kérelemre csak egyszer szavazhat.

4. Kérelem törlése ID alapján

- **Útvonal:** /:id (DELETE)

- **Funkció:**

Csak adminisztrátorok törölhetnek kéréseket az azonosítójuk alapján.

Biztonság

- **Hitelesítés:**

A requireAuth middleware biztosítja, hogy a felhasználók be legyenek jelentkezve.

- **Jogosultságkezelés:**

A kérések törléséhez adminisztrátori jogosultság szükséges, amelyet a requireAdmin middleware kezel.

Kapcsolatok

- **Request modell:** A kérések tárolására szolgál, és kapcsolódik a **User** modellhez a beküldő felhasználók azonosítása érdekében.
- **Quiz modell:** A kérések ellenőrzik, hogy már létezik-e az adott című kvíz, mielőtt új kérelmet hoznak létre.

• Felépítés

A szerveroldal a Node.js és az Express.js keretrendszerre épül. A projekt főbb komponensei közé tartoznak a middleware-ek, modellek, útvonalak, valamint a konfigurációs fájlok. A projekt moduláris szerkezete könnyen bővíthetővé és karbantarthatóvá teszi az alkalmazást.

Szerkezet:

- **node_modules**
 - A Node.js függőségek mappája, amelyek a package.json alapján telepítésre kerülnek.
- **middleware**
 - **auth.js:** Hitelesítési és jogosultság-ellenőrzési middleware-ek, például a token-alapú hozzáférés kezelése.
- **models**
 - **Option.js:** A kvízkérdésekhez tartozó válaszlehetőségek modellje.
 - **Question.js:** A kvíz kérdések adatainak modellje.
 - **Quiz.js:** A kvizek struktúráját és kapcsolódásait leíró modell.
 - **Request.js:** A felhasználók által benyújtott kvízötletek modellje.
 - **Result.js:** A felhasználók kvízeredményeit tároló modell.
 - **User.js:** A felhasználók adatainak kezelésére szolgáló modell.

- **routes**

- **options.js**: Az opciókhoz tartozó API-végpontok definiálása.
- **questions.js**: A kérdések CRUD műveleteinek útvonalai.
- **quizzes.js**: A kvízekhez tartozó API-végpontok.
- **requests.js**: A kvízötletek kezelésére szolgáló végpontok.
- **results.js**: Az eredményekhez kapcsolódó API-végpontok.
- **users.js**: A felhasználói műveletek, például regisztráció és bejelentkezés végpontjai.

- **test**

- Az alkalmazás tesztjeinek tárolására szolgáló mappa.

- **config.js**

- Az alkalmazás konfigurációs beállításait tartalmazza, például titkos kulcsokat és adatbázis-kapcsolati információkat.

- **index.js**

- Az alkalmazás belépési pontja, amely inicializálja a szerveret és betölti a szükséges modulokat.

- **package-lock.json és package.json**

- Az alkalmazás függőségeinek leírását és azok pontos verzióit tartalmazzák.

- **seeds.js**

- Adatbázis feltöltésére szolgáló szkript, például tesztadatok generálására.

Kliensoldal

• Hitelesítés – Autentikáció – Autorizáció

A QuizMaster alkalmazás hitelesítési folyamatai a felhasználók biztonságos bejelentkezését, regisztrációját, valamint az adminisztrátori jogosultságok kezelését biztosítják. A React-alapú kliensoldalon a hitelesítést egy **AuthContext** segítségével valósítjuk meg, amely globális állapotként tárolja a felhasználói információkat és a hitelesítési állapotot.

○ AuthContext

- A kontextus biztosítja a hitelesítési állapot (pl. bejelentkezett-e a felhasználó, admin-e) kezelését az alkalmazás egészében.
- A felhasználói adatok helyi tárolóban (localStorage) kerülnek mentésre egy JWT token formájában, amelyet a rendszer az oldalak közötti navigáció során ellenőriz.
- A token érvényessége ellenőrzésre kerül a belépéskor, és lejárt token esetén a felhasználó automatikusan kijelentkeztetésre kerül.

Fő funkciók

1. login(token)

- A kapott JWT token mentése a helyi tárolóba.
- A felhasználó állapotának frissítése (bejelentkezett és adminisztrátor-e).

2. logout()

- A token eltávolítása a helyi tárolóból.
- A felhasználó kijelentkeztetése és állapotának visszaállítása alaphelyzetbe.

3. checkTokenValidity()

- A helyi tárolóban lévő token dekódolása és érvényességének ellenőrzése.

- Lejárt vagy hibás token esetén automatikusan kijelentkezteti a felhasználót.

- **Bejelentkezés és regisztráció**

- A felhasználók azonosítása a Login és Register komponensek űrlapjai segítségével történik.
- A regisztráció és a bejelentkezés során a backend validálja az adatokat, és egy érvényes JWT tokenel válaszol sikeres hitelesítés esetén. A token a kliensoldalon a helyi tárolóba kerül, és a rendszer ezt használja az API-kérések hitelesítésére.

- **Védett útvonalak**

- **ProtectedRoute:** Kizárólag bejelentkezett felhasználók férhetnek hozzá. Nem hitelesített felhasználók esetén a rendszer automatikusan a bejelentkezési oldalra irányít.
- **AdminProtectedRoute:** Csak az adminisztrátorok számára elérhető oldalak védelme. Ha a felhasználó nem rendelkezik admin jogosultsággal, a főoldalra kerül átirányításra.

- **Kijelentkezés**

- A **Logout** komponens biztosítja a felhasználók kijelentkezését, amely során a token törlődik a helyi tárolóból, és a felhasználó visszakerül a bejelentkezési oldalra.

- **Kapcsolatok**

- **Backend integráció:** A regisztráció és bejelentkezés során a kliensoldal a szerverrel kommunikál, ahol a hitelesítési adatok validálásra kerülnek.
- **JWT használata:** A hitelesítés biztonságos megvalósítására JSON Web Tokenek szolgálnak, amelyek a felhasználói jogosultságokat is tartalmazzák.

- **Middleware-ek:** A `ProtectedRoute` és `AdminProtectedRoute` middleware-ek segítségével biztosítjuk, hogy csak jogosult felhasználók érjék el az adott oldalakat.

- **Biztonság**

- A JWT tokenek titkosítása és az időbeli érvényességük garantálja, hogy csak a megfelelő felhasználók férhessenek hozzá az alkalmazás erőforrásaihoz.
- A token helyességének ellenőrzése minden oldalbetöltéskor megtörténik, hogy a rendszer automatikusan kijelentkeztesse a jogosulatlan felhasználókat.

- **Kvízekkel kapcsolatos komponensek**

- **Quizzes**

Cél: Az összes kvíz listázása, a kitöltött kvízek megjelölése és kezelése.

Fő funkciók:

- **fetchQuizzes:** Lekéri az összes kvíz adatait a backendről, majd ellenőrzi, hogy a felhasználó kitöltötte-e az adott kvízt.
- **handleDelete:** Egy adott kvíz törlése adminisztrátor által. Sikeres törlés után eltávolítja a kvízt a listából.
- **confirmDelete:** Megnyit egy megerősítő modált a kvíz törléséhez.
- **closeModal:** Bezárja a törlés megerősítő modált.
- **handleRetakeQuiz:** Újraindítja a kiválasztott kvízt, átirányítva a felhasználót a kvíz oldalára.

- **Quiz**

Cél: Egy adott kvíz kitöltése, válaszok beküldése és előző eredmények megtekintése.

Fő funkciók:

- **validateQuiz**: Ellenőrzi a kvíz adatainak helyességét (pl. minimum kérdésszám, válaszlehetőségek száma).
- **shuffle**: Véletlenszerű sorrendbe helyezi a kérdéseket és válaszokat.
- **handleOptionChange**: Kezeli, hogy melyik válaszopciót választotta ki a felhasználó.
- **handleSubmit**: Beküldi a felhasználó válaszait a backendnek. Ellenőrzi, hogy minden kérdés megválaszolt-e, és menti az eredményeket.
- **handleNextSlide** és **handlePrevSlide**: Kezeli a kvízben a kérdések közötti navigációt.
- **handleRetake**: Újraindítja a kvízt a felhasználó számára, törölve az előző válaszokat.

○ NewQuiz

Cél: Új kvíz létrehozása adminisztrátorok számára.

Fő funkciók:

- **handleQuestionCountChange**: Kezeli a kérdések számának megadását, és automatikusan frissíti a szükséges kérdések számát.
- **handleQuestionChange**: Módosítja a kérdés szövegét az adott indexű kérdésben.
- **handleOptionChange**: Frissíti az opció szövegét vagy helyességi státuszát az adott kérdésben.
- **addQuestion**: Új kérdést ad a kvízhez.
- **deleteQuestion**: Töröl egy kérdést, ha az összes kérdés száma meghaladja a minimális követelményt.
- **handleSubmit**: A kvíz adatainak beküldése a backendnek, új kvíz létrehozása céljából.

- **EditQuiz**

Cél: Meglévő kvíz adatainak szerkesztése adminisztrátor által.

Fő funkciók:

- **fetchQuiz:** Lekéri a kiválasztott kvíz adatait a backendről.
- **handleQuestionChange:** Módosítja a kérdés szövegét az adott indexű kérdésben.
- **handleOptionChange:** Frissíti az opció szövegét vagy helyességi státuszát az adott kérdésben.
- **addNewQuestion:** Új kérdést ad a kvízhez.
- **deleteQuestion:** Töröl egy meglévő kérdést, ha az összes kérdés száma meghaladja a minimális követelményt.
- **handleSubmit:** A szerkesztett kvíz adatainak beküldése a backendnek.

- **Kérelmekkel kapcsolatos komponensek**

- **Requests**

Cél: A felhasználók által beküldött kvízkérelmek megjelenítése, szavazás és kérések kezelése.

Fő funkciók:

- **fetchRequests:** Lekéri az összes kérést a backendről és szortírozza azokat a szavazatok száma alapján.
- **handleUpvote:** Lehetővé teszi a felhasználók számára, hogy tetszésüket fejezzék ki egy kérés iránt.
- **handleDeleteRequest:** Adminisztrátorok számára biztosítja a kérések törlésének lehetőségét.

- **handleCreateQuiz:** Egy kérelem alapján új kvíz létrehozásának kezdeményezése adminisztrátor által.

- **NewRequest**

Cél: Új kvízkérelmek beküldése felhasználók által.

Fő funkciók:

- **handleSubmit:** Feldolgozza a felhasználó által beküldött kérést, validálja az adatokat, majd elküldi azokat a backendnek.
- **setError:** Hibakezelési mechanizmust biztosít, például amikor a felhasználó nincs bejelentkezve vagy hiba történik az adatfeldolgozás során.

- **Eredményekért felelős komponens**

- **Leaderboard**

Cél: A felhasználók által elért eredmények megjelenítése egy adott kvízhez, rangsor alapján.

Fő funkciók:

- **fetchLeaderboard:**
 - Lekéri az adott kvíz eredményeit a backendből.
 - Az eredményeket pontszám és kitöltési idő alapján rendezi.
 - Beállítja a kvíz címét az adatok alapján.
- **formatTime:** A kitöltési időt (másodpercekben) emberi olvasható formátumra alakítja: perc és másodperc bontásban.

- **Navigálásért felelős komponens**

- **Navigációs Sáv (Navbar)**

Cél: Az alkalmazás különböző szekciói közötti navigáció biztosítása, figyelembe véve a felhasználó bejelentkezési és jogosultsági állapotát.

Fő funkciók:

- **toggleNavbar:** A navigációs sáv mobilnézetben történő megnyitása és bezárása.
- **closeNavbar:** Biztosítja, hogy az oldalak közötti váltáskor a navigációs sáv automatikusan bezáródjon.
- **Felhasználónév megjelenítése:** A JWT token alapján megjeleníti a bejelentkezett felhasználó nevét, valamint admin esetén az (admin) jelölést.

Dinamikus tartalom: Más-más menüpontok láthatóak be nem jelentkezett felhasználók, bejelentkezett alapfelhasználók és adminisztrátorok számára

- **Felépítés**

A kliensoldal React.js alapú alkalmazás, amely modern eszközökre épül, mint a Vite, a JSX és a Context API. A projekt jól szervezett szerkezete lehetővé teszi az alkalmazás logikus felépítését, könnyű karbantarthatóságát és bővíthetőségét.

Szerkezet:

- **node_modules**

- A React.js és a kapcsolódó könyvtárak függőségeinek tárolása, a package.json alapján telepítve.

- **src**

- Az alkalmazás fő forráskódját tartalmazza, ideértve a komponenseket, az oldalakat, a stílusokat és a konfigurációs fájlokat.

- **mocks**
 - Teszteléshez szükséges mock adatok és függőségek.
- **assets**
 - Statikus erőforrások, például képek, logók és egyéb médiafájlok.
- **components**
 - **AdminProtectedRoute.jsx**: Csak adminisztrátori hozzáférést biztosító útvonalvédelmi komponens.
 - **Navbar.jsx**: Az alkalmazás navigációs sávját biztosító reszponzív komponens.
 - **ProtectedRoute.jsx**: Bejelentkezett felhasználók számára elérhetővé tett oldalakhoz tartozó útvonalvédelem.
- **context**
 - **AuthContext.jsx**: Az alkalmazás globális hitelesítési állapotát kezelő komponens.
- **pages**
 - **auth**
 - **Login.jsx**: Bejelentkezési oldal a felhasználók számára.
 - **Logout.jsx**: Kijelentkezési funkciót megvalósító oldal.
 - **Register.jsx**: Regisztrációs oldal új felhasználók számára.
 - **quiz**
 - **EditQuiz.jsx**: Kvizek szerkesztésére szolgáló adminisztrátori oldal.
 - **NewQuiz.jsx**: Új kvizek létrehozását lehetővé tevő adminisztrátori oldal.
 - **Quiz.jsx**: Kvíz kitöltési oldal, dinamikus kérdésekkel és válaszokkal.
 - **Quizzes.jsx**: Az összes kvíz listáját megjelenítő oldal.

- **request**
 - **NewRequest.jsx**: Új kvízkérelmek beküldésére szolgáló oldal.
 - **Requests.jsx**: A már meglévő kvízkérelmeket megjelenítő oldal.
- **result**
 - **Leaderboard.jsx**: Az egyes kvízek eredménylistáját megjelenítő oldal.
- **Home.jsx**: Az alkalmazás nyitóoldala.
- **NotFound.jsx**: 404-es hibaoldal a nem létező útvonalak kezelésére.
- **styles**
 - Az alkalmazás globális és komponens-specifikus stíluslapjait tartalmazza.
- **test**
 - Az alkalmazás komponenseinek és funkcióinak tesztjei.
- **App.jsx**
 - Az alkalmazás fő komponense, amely az összes oldalt és route-ot tartalmazza.
- **main.jsx**
 - Az alkalmazás indítási fájlja, amely a React DOM rendereléséért felel.
- **router.jsx**
 - Az alkalmazás útvonalainak (routes) konfigurációját kezeli.
- **.gitignore**
 - A Git által figyelmen kívül hagyott fájlokat definiálja.
- **eslint.config.js**
 - Az ESLint konfigurációs fájlja, amely a kódminőség és szabványok biztosítására szolgál.
- **index.html**
 - Az alkalmazás HTML belépési pontja.

- **jest.setup.js**
 - Teszteléshez szükséges konfigurációs fájl.
- **package-lock.json** és **package.json**
 - Az alkalmazás függőségeit és azok verzióit tartalmazó fájlok.
- **vite.config.js**
 - A Vite build eszköz konfigurációját tartalmazó fájl.

3.4.3. Adatbázis

Az alkalmazás **MongoDB** adatbázist használ, amely egy **dokumentumalapú, NoSQL adatbázis-kezelő rendszer**. Ez a megoldás különösen rugalmas és skálázható, mivel az adatok JSON-szerű dokumentumok formájában tárolódnak, amelyeket dinamikusan lehet strukturálni. A MongoDB a felhőben is elérhető, így könnyen integrálható különböző környezetekbe, és biztosítja a magas rendelkezésre állást.

Adatbázis Kapcsolat

A szerver és az adatbázis közötti kapcsolat a **Mongoose** könyvtár segítségével valósul meg, amely egy JavaScript ORM (Object-Relational Mapping) eszköz. Ez leegyszerűsíti az adatbázis-műveletek kezelését, például a **CRUD (Create, Read, Update, Delete)** funkciók végrehajtását. Az adatbázis URI (Uniform Resource Identifier) elérési útvonala titkosítva kerül tárolásra a kódbázisban, és az alkalmazás elindításakor automatikusan kapcsolódik az adatbázishoz.

Felhőalapú tárolás

Az adatbázis **felhőalapú verzióját** használjuk, amely a MongoDB Atlas szolgáltatáson keresztül érhető el. Ez lehetővé teszi az adatok biztonságos és gyors tárolását egy megbízható környezetben. A felhőalapú tárolás további előnye, hogy egyszerűsíti az adatbázis skálázását, például ha az alkalmazás nagyobb terhelést kap.

Konfiguráció

A konfigurációs beállítások, például az adatbázis URL-je (MONGO_URI), egy dedikált konfigurációs fájlban kerülnek tárolásra, amelyet az alkalmazás a környezeti változók alapján tölt be. Ez biztosítja, hogy a helyi fejlesztés és a felhőalapú környezet között egyszerű legyen a váltás.

Gyűjtemények - Táblák

- **users:** A regisztrált felhasználók adatait tárolja, például felhasználónév, email cím, és jogosultsági szint (admin vagy felhasználó).
- **options:** A kvízkérdések lehetséges válaszlehetőségeit tartalmazza, beleértve a helyes választ.
- **questions:** A kvízkérdéseket tárolja, amelyek kapcsolódnak az opciókhoz.
- **quizzes:** Az egyes kvízek metaadatait (pl. cím, kérdések listája) tárolja.
- **results:** A felhasználók által elért eredményeket, pontszámokat és a kitöltési időket rögzíti.
- **requests:** A felhasználók által benyújtott új kvízötleteket és azok lájkjait (upvote) kezeli.

3.5. Tesztelés

3.5.1. Szerveroldali tesztelés

A szerveroldali tesztelést **unit** és **integrációs** tesztekkel végeztem a Mocha tesztkeretrendszer és a Babel segítségével. A unit tesztek a kód egyes részegységeit, például a modelleket és middleware-eket ellenőrzik, míg az integrációs tesztek az útvonalak működését és az API válaszokat vizsgálják. A tesztek a következő könyvtárstruktúrában találhatóak:

- **Unit tesztek:**
 - **middleware:** Az autentikációt kezelő middleware tesztjei, pl. `authMiddleware.test.js`.
 - **models:** Az adatbázis modellek helyes működésének ellenőrzése, pl. `option.test.js`.
- **Integrációs tesztek:**
 - Az API útvonalak működését tesztelik, pl. `optionsRoutes.test.js`.

A teszteket a következő parancsokkal lehet futtatni:

- **Unit tesztek futtatása:** `npm run test-unit`
- **Integrációs tesztek futtatása:** `npm run test-integration`
- **Minden teszt futtatása:** `npm run test-all`

A tesztesetek:

```
Auth Middleware
  requireAuth
    ✓ should call next if token is valid
    ✓ should return 401 if no token is provided
    ✓ should return 401 if token is invalid
  requireAdmin
    ✓ should call next if user is admin
    ✓ should return 401 if user is not admin
    ✓ should return 401 if user is undefined
```

59. ábra: Middleware tesztesetek (szerveroldal)

Option Model Unit Tests

- ✓ should create a new option with valid fields
- ✓ should fail if required fields are missing
- ✓ should fail if optionText is missing
- ✓ should fail if isCorrect is missing

Question Model Unit Tests

- ✓ should create a new question with valid fields
- ✓ should fail if required fields are missing
- ✓ should allow a question with no options
- ✓ should reference saved options correctly

Quiz Model Unit Tests

- ✓ should create a new quiz with valid fields, including questionCount
- ✓ should fail if title is missing
- ✓ should fail if questionCount is missing
- ✓ should fail if questionCount is not 5, 10, or 20
- ✓ should reference saved questions correctly

60. ábra: Modell tesztesetek 1 (szerveroldal)

Request Model Unit Tests

- ✓ should create a new request with valid fields (93ms)
- ✓ should fail if title is missing (73ms)
- ✓ should allow users to upvote a request (156ms)
- ✓ should reference the user who created the request (82ms)

Result Model Unit Tests

- ✓ should create a result with valid fields
- ✓ should fail if required fields are missing
- ✓ should reference the user and quiz correctly
- ✓ should validate selectedQuestions references correctly

User Model Unit Tests

- ✓ should create a new user with valid fields (77ms)
- ✓ should fail if required fields are missing
- ✓ should hash the password before saving (142ms)
- ✓ should compare passwords correctly (215ms)

31 passing (2s)

61. ábra: Modell tesztesetek 2 (szerveroldal)

```

Options Routes
Sikeres kapcsolat a MongoDB-hez.
A szerver a 5000 porton fut.
  POST /options/add
    ✓ should add a new option to a question (85ms)
  GET /options/:id
    ✓ should retrieve an option by ID
  PUT /options/:id
    ✓ should update an option by ID (49ms)
  DELETE /options/:id
    ✓ should delete an option by ID (72ms)

Question Routes Tests
  POST /questions/add
    ✓ should add a new question (71ms)
  GET /questions/:id
    ✓ should retrieve a question by ID
  GET /questions
    ✓ should retrieve all questions (76ms)
  PUT /questions/:id
    ✓ should update a question by ID (64ms)
  DELETE /questions/:id
    ✓ should delete a question by ID (60ms)

```

62. ábra: Integrációs tesztesetek 1 (szerveroldal)

```

Quizzes Routes
  GET /quizzes/:id
    ✓ should retrieve a quiz by ID (92ms)
  PUT /quizzes/:id
    ✓ should update a quiz by ID (449ms)
  DELETE /quizzes/:id
    ✓ should delete a quiz by ID (2100ms)

Requests Routes
  POST /requests/add
    ✓ should create a new request with valid data (91ms)
    ✓ should not create a request with missing title
    ✓ should not create a request if quiz with same title exists (97ms)
    ✓ should not allow creating a duplicate request title (62ms)
  GET /requests
    ✓ should retrieve all requests (59ms)
  PUT /requests/:id/upvote
    ✓ should upvote a request by a different user (91ms)
    ✓ should not upvote own request

```

63. ábra: Integrációs tesztesetek 2 (szerveroldal)

```
Results Routes
  POST /results/quiz/:quizId
    ✓ should add a new result with valid data, without answers (61ms)
  GET /results/user/:userId
    ✓ should retrieve all results for the user (58ms)
    ✓ should not allow other users to access these results (260ms)
  GET /results/quiz/:quizId
    ✓ should retrieve all results for a quiz (60ms)

Users Routes
  POST /register
    ✓ should register a new user with valid fields (153ms)
    ✓ should not register a user with missing fields
    ✓ should not register a user with an existing username (131ms)
    ✓ should not register a user with a duplicate email (158ms)
    ✓ should not register a user with an invalid email format
    ✓ should not register a user with a weak password
  POST /login
    ✓ should log in an existing user with correct credentials (117ms)
    ✓ should not log in a user with incorrect password (109ms)
    ✓ should not log in a non-existing user

32 passing (10s)
```

64. ábra: Integrációs tesztesetek 3 (szerveroldal)

3.5.2. Kliensoldali tesztelés

A kliensoldali tesztelés biztosítja, hogy az alkalmazás React-alapú komponensei és funkciói megfelelően működjenek. Ehhez a **Jest** keretrendszert használtuk, amely lehetőséget nyújt mind unit tesztek, mind integrációs tesztek írására. A tesztek az alkalmazás különböző részeit fedik le, mint például a komponensek viselkedése, a kontextuskezelés, valamint az oldalak funkcionalitása.

A tesztek a projekt **test** mappájában találhatók, és különböző al-mappákba vannak rendezve:

- **components:** A fontosabb komponensek, például a navigációs sáv (Navbar) és a védett útvonalak (ProtectedRoute, AdminProtectedRoute) tesztjei.
- **context:** Az alkalmazás hitelesítési logikáját kezelő AuthContext tesztjei.

- **pages:** Az egyes oldalakhoz tartozó funkcionalitás ellenőrzése, mint például a Login, Quiz vagy a Leaderboard oldalak.

Minden teszt futtatása: npm run test-all

A tesztesetek (54 db) összefoglalása:

	Teszteset	Eredmény
1-6	<p>Navigációs sáv megjelenése és linkek működése</p>	<p>A QuizMaster logó és a "Összes kvíz megtekintése" link helyesen jelenik meg. Az admin-specifikus és felhasználó-specifikus linkek csak a megfelelő jogosultságú felhasználóknál érhetők el. A bejelentkezési és regisztrációs linkek megjelennek, ha a felhasználó nincs bejelentkezve, míg a kijelentkezési link bejelentkezett felhasználóknál érhető el.</p>
7-9	<p>ProtectedRoute viselkedése</p>	<p>"Loading..." üzenet jelenik meg betöltés közben. Bejelentkezés nélkül a login oldalra irányít. Bejelentkezett felhasználók számára a védett tartalom helyesen megjelenik.</p>

10-13	AdminProtectedRoute viselkedése	<p>"Loading..." üzenet jelenik meg admin funkciók betöltése közben. Ha a felhasználó nincs bejelentkezve, a login oldalra irányít. Bejelentkezett, nem admin felhasználók a főoldalra kerülnek átirányításra. Az admin tartalom kizárólag admin jogosultságú felhasználóknál érhető el.</p>
14-18	AuthContext tesztelése	<p>A login metódus érvényes tokennel frissíti az állapotot, az admin jogokat is beleértve. A logout metódus kijelentkezteti a felhasználót, és törli a localStorage tartalmát. Érvénytelen token esetén a rendszer automatikusan kijelentkezteti a felhasználót.</p>
19-22	Regisztrációs folyamat tesztelése	<p>A regisztrációs oldal form elemei (felhasználónév, email stb.) helyesen megjelennek. Hibás regisztráció esetén a megfelelő hibaüzenet látható. Sikeres regisztráció során a login metódus meghívásra kerül a tokennel. Betöltési állapotban a Regisztráció gomb le van tiltva.</p>

23-25	Bejelentkezési folyamat tesztelése	<p>Hibás jelentkezés esetén a megfelelő hibaüzenet jelenik meg.</p> <p>Sikeres jelentkezéskor a login metódus meghívásra kerül a tokennel.</p> <p>Betöltési állapotban a Bejelentkezés gomb le van tiltva.</p>
26	Kijelentkezés tesztelése	<p>A logout metódus helyesen meghívásra kerül, amikor a felhasználó a kijelentkezés oldalt megnyitja.</p>
27-29	Kezdőoldal tesztelése	<p>A QuizMaster logó, a címsor és a bemutatkozó szövegek helyesen megjelennek. Ha a felhasználó nincs bejelentkezve, a "Csatlakozom" és a "Megnézem a kvizeket" gombok is megjelennek. Bejelentkezett felhasználóknál csak a "Megnézem a kvizeket" gomb látható.</p>
30-31	Kvizek megjelenítése és állapotuk tesztelése	<p>A kvizek listája helyesen megjelenik. A már kitöltött kvizeknél "Ön már kitöltötte ezt a kvízt" üzenet látható, míg a kitöltetlen kvizek elérhetőek kitöltésre. Hibás API-hívás esetén megfelelő hibaüzenet jelenik meg.</p>

32-33	Kvíz törlése admin felhasználóként	Adminisztrátorok számára a kvíz törlésekor egy megerősítő modális ablak jelenik meg. A "Törlés" gombra kattintva a kvíz helyesen eltávolításra kerül a listából. Hibás törlési folyamat esetén megfelelő hibaüzenet jelenik meg.
34	Kvíz kérdések megjelenítése betöltéskor	A rendszer sikeresen lekéri a kvíz adatait, beleértve a kérdéseket és a válaszlehetőségeket. Az első kérdés és annak opciói helyesen láthatók.
35-36	Új kvíz létrehozásának tesztelése	Sikeres kvíz létrehozáskor a rendszer helyesen hívja meg az API-t a megadott adatokkal. Hibás kéréseknél megjelenik a megfelelő hibaüzenet a felhasználó számára.
37-39	Kvíz szerkesztésének tesztelése	A komponens helyesen lekéri és megjeleníti a kvíz adatait betöltéskor. Sikeres mentés esetén a megfelelő API-hívás történik a frissített adatokkal. Hibás API-hívás esetén a felhasználó értesítést kap a problémáról.

<p>40-44</p>	<p>Leaderboard tesztelése</p>	<p>Az eredmények betöltésekor a "Eredmények betöltése..." üzenet jelenik meg. Az adatok helyesen jelennek meg, beleértve a felhasználóneveket, pontszámokat és kitöltési időket. Hiba esetén megfelelő hibaüzenet látható, és üres eredményeknél a "Nincsenek eredmények" üzenet jelenik meg. Az eredményeket helyesen rendezi pontszám és kitöltési idő alapján.</p>
<p>45-49</p>	<p>Kérelmek tesztelése</p>	<p>A kérések megfelelően megjelennek, beleértve a kvízkérelmek címét és lájkszámát. A felhasználók tetszési szavazatai (upvote) helyesen frissítik az adatokat. Adminisztrátor felhasználók számára megjelenik a "Létrehozom a kvízt" gomb. Üres lista esetén a "Jelenleg nincsenek kérelmek." üzenet jelenik meg. Adatbetöltés alatt a "Kérelmek betöltése..." üzenet látható.</p>

50-52	Új kérelem tesztelése	Ha a felhasználó nincs bejelentkezve, a "Bejelentkezés szükséges a kérés leadásához." üzenet jelenik meg. Bejelentkezett felhasználók esetén a kérés sikeresen beküldhető, és a megfelelő API-hívás végrehajtásra kerül. API-hiba esetén a "Sikertelen kérés létrehozás." hibaüzenet jelenik meg.
53-54	404 oldal tesztelése	A "404 - Az oldal nem található" üzenet és a leíró szöveg helyesen megjelenik. A "Vissza a főoldalra" link működőképes, és a főoldalra irányít.

4. Összefoglalás

Az alkalmazás egy modern kvízplatform, amely lehetőséget biztosít a felhasználóknak tudásuk tesztelésére és új kvizek létrehozására. A projekt a **MERN stack** technológiáit alkalmazza, ahol a backend a REST API-t és az adatkezelést biztosítja, míg a frontend reszponzív felületet nyújt. Az adminisztrátorok külön jogosultságokat kapnak, például kvizek törlésére és szerkesztésére.

A dokumentáció részletesen bemutatja az alkalmazás szerkezetét, a szerver- és kliensoldali komponenseket, valamint az adatbázis-modellt. A rendszer tesztelése során unit és integrációs tesztek végeztünk elsősorban **Jest**, **Mocha**, és **Chai** segítségével, biztosítva a megfelelő tesztlefedettséget.

A projekt célja egy olyan kvízplatform létrehozása volt, amely tanulási és szórakoztatási célokat szolgál, lehetőséget adva a felhasználóknak tudásuk bővítésére és mások kihívására. Az alkalmazás fejlesztése során kiemelt szempont volt a skálázhatóság, a modern technológiák alkalmazása, valamint a biztonságos és felhasználóbarát működés. A dokumentáció az alkalmazás használatának, fejlesztésének és működésének pontos rögzítésére szolgál, segítve ezzel a további fejlesztéseket és a rendszer karbantartását.

5. Továbbfejlesztési lehetőségek

- **Kvíz szerkesztése alapfelhasználók által:** Egy funkció bevezetése, amely lehetővé teszi a regisztrált alapfelhasználóknak, hogy saját kvizeket hozzanak létre és osszanak meg a közösséggel.
- **Elismerési rendszer bevezetése:** Jelvények, szintek és összesített ranglista bevezetése a felhasználók aktivitásának és teljesítményének ösztönzésére.
- **Közösségi interakciók lehetősége:** Hozzászólások és értékelések engedélyezése a kvizek alatt, hogy a felhasználók megoszthassák véleményüket és tapasztalataikat.
- **Többnyelvű támogatás bevezetése:** Az oldalak és kvizek többnyelvűvé tétele, hogy szélesebb közönség számára is elérhetővé váljanak.

6. Felhasznált források

- [1] NodeJS dokumentáció: <https://nodejs.org/docs/latest/api/> - 2024.11.30.
- [2] NodeJS letöltés: <https://nodejs.org/en/download> - 2024.11.30.
- [3] React dokumentáció: <https://react.dev/> - 2024.11.30.
- [4] Bootstrap dokumentáció: <https://getbootstrap.com/docs/5.3/getting-started/introduction/> - 2024.11.30.
- [5] MongoDB Atlas: <https://www.mongodb.com/products/platform/cloud> - 2024.11.30.
- [6] MERN Stack útmutató: <https://www.mongodb.com/resources/languages/mern-stack-tutorial> - 2024.11.30.
- [7] Drótváztervek – MockFlow WireframePro: <https://mockflow.com/apps/wireframepro/> - 2024.11.30.
- [8] UML diagramok – Visual Paradigm: <https://online.visual-paradigm.com/diagrams/> - 2024.11.30.
- [9] Háttér a kezdőlap: <https://www.vecteezy.com/photo/15131452-man-holding-question-mark-concept-of-question-mark-and-fags-ask-question-online-faq-concept-what-how-and-why-search-information-on-internet> - 2024.11.30.
- [10] Háttér a többi oldalon: <https://www.rawpixel.com/image/91698/free-photo-image-night-sky-starry> - 2024.11.30.
- [11] Kép a regisztrációs és bejelentkező oldalon: https://favpng.com/png_view/thinking-man-incandescent-light-bulb-cartoon-clip-art-png/bBAv0Tkg - 2024.11.30.
- [12] 'shuffle' metódus a Quiz.jsx komponensben: <https://stackoverflow.com/questions/2450954/how-to-randomize-shuffle-a-javascript-array> - 2024.11.30.