Vas Megyei Szakképzési Centrum

Nádasdy Tamás Technikum és Kollégium

PROJEKTFELADAT

Space Teszt

Redőcs Marcell

Konzulens:
Domnánovich Bálint

2024

Nyilatkozat

Alulírott, Redőcs Marcell kijelentem, hogy a Space Teszt című projektfeladat kidolgozása a saját munkám, abban csak a megjelölt forrásokat, és a megjelölt mértékben használtam fel, az idézés szabályainak megfelelően, a hivatkozások pontos megjelölésével.

Eredményeim saját munkán, számításokon, kutatáson, valós méréseken alapulnak, és a legjobb tudásom szerint hitelesek.

Sárvár, 2024.04.19 _____

Hallgató

Kivonat

Space Teszt

Jelen záródolgozat egy olyan webes alkalmazást mutat be, mely hatékony segítséget nyújt tanároknak a tanulók teljesítményének nyomon követésében. A Space Teszt egy modern platform, mely kiemelkedő lehetőséget biztosít a tanároknak a diákok tudásának értékelésére.

Az alkalmazás egy intuitív felhasználói felülettel rendelkezik, mely lehetővé teszi a tanárok számára, hogy könnyedén felmérjék a diákok tudását és nyomon kövessék fejlődésüket. A platform segítséget nyújt a feladatok értékelésében és a visszajelzés adásában, így elősegítve a tanárok hatékonyabb munkavégzését. Tanároknak segíthet a munka-élet egyensúly fenntartásában.

Emellett a diákok számára is számos előnnyel jár az alkalmazás használata. A feladatlapok egyszerűen értelmezhetőek és kitölthetőek. A gyors visszajelzés a diákok számára, mivel az eredményeik azonnal megtekinthetőek és értékelhetőek.

Abstract

Space Teszt

This final project presents a web application that provides efficient assistance to teachers in tracking students' performance. The Space Test is a modern platform that offers outstanding opportunities for teachers to assess students' knowledge.

The application features an intuitive user interface, allowing teachers to easily assess students' knowledge and track their progress. The platform assists in task evaluation and providing feedback, thereby promoting more effective work for teachers and helping maintain work-life balance.

Additionally, the application brings several benefits to students. The worksheets are easy to understand and complete. Quick feedback is provided to students, as their results can be viewed and evaluated immediately.

Tartalomjegyzék

1. Bevezetés

A 2024-es záróvizsgám beadandójához témának egy olyan weblapot választottam, amiben a tanárok teszteket irattathatnak a diákokkal. Célja az, hogy a diákokat osztályozzák.

Fő célom az volt, hogy egy hatékony eszközt adjak a tanárok számára, akik a diákok dolgozatait kívánják javítani gyorsan és egyszerűen.

Tanárként létrehozhatunk teszteket, amelyeket osztályokra és tantárgyra szűkíteni tudunk. A tesztek teljesen személyre szabhatóak, 3 feladat típus közül lehet választani Igaz / Hamis, Négylehetőséges, Több helyes válasz, és bármennyi kérdés akármennyi válasszal általunk meghatározott pontszámmal hozhatók létre. Ezek a kérdések és válasz lehetőségek kerülnek a diákokhoz. Majd később lehetőségünk a tanároknak lesz arra, hogy értékeljük a leadott dolgozatokat.

A Weblap fejlesztéséhez a React-ban szerzett programozási ismereteimet és a mySQL adatbázis-kezelési készségeimet használtam fel.

- ➔ Az alkalmazáshoz Visual Studio 2024 programozási környezetet használtam JavaScript nyelven, React Weblap-ként.
- ➔ Xampp-ot a lokális környezet kialakításához.
- ➔ Az adatbázis tervezéséhez és elkészítéséhez pedig Xampp azon belül Phpmyadmin segített.
- ➔ Az adatok tárolása MySQL szerverben történik.
- ➔ Api tesztlésekhez Postman-t használtam

A programban használt képeket, illetve ikonokat az irodalomjegyzékben jelölt oldalról és az Adobe Photoshop 2021 program segítségével készítettem.
A React egy JavaScript könyvtár, amely segít felhasználói felületeket készíteni. A legfontosabb előnyei közé tartozik a komponens alapú fejlesztés, a Virtual DOM használata a jobb teljesítményért, az egyszerűség és a könnyű kiterjeszthetőség. A React gyors fejlesztést tesz lehetővé, könnyen tanulható, és élvezheti az aktív közösség támogatását.
A Postman egy sokoldalú eszköz az API-k fejlesztéséhez és teszteléséhez. Főbb feladatokat végeztem benne:

API tesztelése: Postman segítségével különféle HTTP kéréseket küldhetsz (mint GET, POST, PUT, DELETE, stb.) egy adott API végpontraKérések szerkesztése és elemezheted a választ.

Egyszerűen megadhatod a kérések paramétereit és tartalmát.

Automatizált tesztelés: Tesztszkriptek létrehozása a válaszok automatikus ellenőrzésére.

Adatgyűjtés és dokumentáció: Gyűjthetsz API-kéréseket egy "Collection"-ben és dokumentálhatod azokat.

2. Hasonló Weblapok

Ami a legjobban hasonlít az a Redmenta egy online tesztkészítő platform, ahol könnyedén létrehozhatsz és megoszthatsz kvízeket, kérdőíveket vagy felméréseket másokkal. A felhasználók egyszerűen testre szabhatják kérdéseiket és válaszlehetőségeiket, majd megoszthatják azokat a közösséggel. Ideális eszköz tanuláshoz, oktatáshoz és szórakozáshoz egyaránt.

A Google űrlapok az az alkalmazás, ami talán hasonló. Teszteket hozhatunk benne létre és segít a javításban. Azonban az űrlapok nem az iskoláknak készült elsősorban. Egyszerűbb tesztekhez és választások lebonyolítására tökéletes, viszont nem lehet benne diákokat és tanárokat létrehozni vagy kezelni.

Typeform: A Typeform egy másik népszerű online felméréskészítő eszköz, amely intuitív felhasználói felülettel rendelkezik és számos testre szabási lehetőséget kínál.

SurveyMonkey: Ez egy online felméréskészítő platform, amely lehetőséget biztosít arra, hogy létrehozz interaktív kérdőíveket, teszteket és felméréseket, és megoszthatod azokat másokkal.

Kahoot!: Ez egy interaktív játékplatform, ahol létrehozhatsz kvízeket és játékokat, és megoszthatod azokat másokkal.

3. Felhasználói dokumentáció

A program fő célja, hogy segítsen a tanároknak számon kérni a diákokat egyszerű módon.

Lehetőségeink tanárként:

- ➔ Osztályok és diákok listázása
- ➔ Saját profil
- ➔ Dolgozatok létrehozása, kijavítása, visszakövetése, megtekintése

Lehetőségeink diákként:

- ➔ Dolgozatok írása és megtekintése
- ➔ Saját profil

Lehetőségeink admin-ként:

- ➔ Tanárok hozzáadása, szerkesztése, pozíciójának kezelése
- ➔ Diákok hozzáadása, szerkesztése
- ➔ Osztályok létrehozása, szerkesztése

3.1. Hardware és Software igény

A weblap futtatásához (az adatbázis lokálisan fut) hálózati kapcsolat szükséges.

Minimális rendszerkövetelmények:

- ➔ Legalább 1 GHz-es processzor
- ➔ 1 GB szabad memória
- ➔ Windows 7 vagy újabb operációs rendszer

Ajánlott rendszerkövetelmények:

- ➔ Legalább 2 GHz-es processzor
- ➔ 2 GB szabad memória
- ➔ Windows 10 vagy újabb operációs rendszer

Itt vannak a legnépszerűbb böngészők, amelyek támogatják a localhost:3000 címen való helyi kiszolgálóhoz való kapcsolódást:

- ➔ Google Chrome
- ➔ Mozilla Firefox
- ➔ Microsoft Edge
- ➔ Apple Safari
- ➔ Opera
- ➔ Brave
- ➔ Vivaldi

3.2. Belépés és regisztráció

Az alkalmazásba előre létrehozott profil vagy regisztráció nélkül nem léphető be! Fiók létrehozására viszont csak az admin jogosult.

Az adminisztrátornak lehetősége van diák vagy tanár profilok létrehozására az adatbázisban. A regisztrációhoz szükség van a diák vagy tanár adataira, mint például teljes név, ami a felhasználóneve lesz, jelszó és e-mail cím. Az adminisztrátor hozzáadja ezeket az adatokat a rendszerhez.

A belépéshez szükséges a megfelelő felhasználónév és jelszó páros megadása, majd a "Bejelentkezés" gombra kattintva vagy az "Enter" lenyomásával megjelenik a főoldal. A felhasználóknak itt lehetősége van neki hozzáférni a szolgáltatásokhoz és tartalmakhoz.

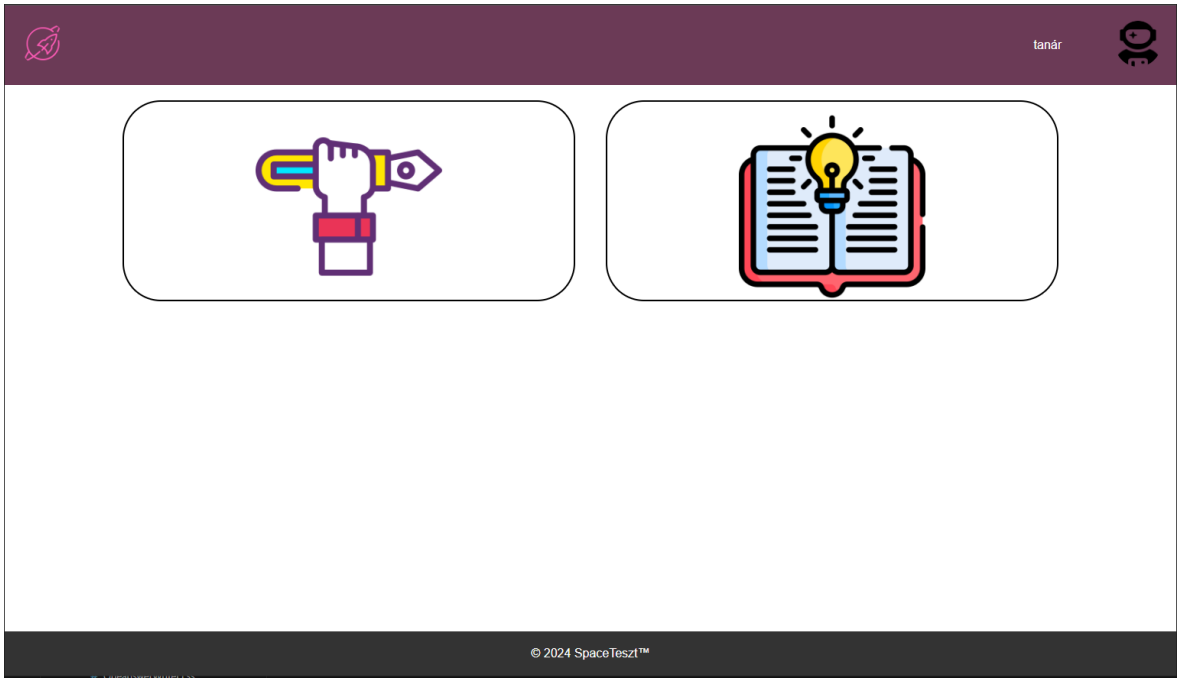
1. ábra a végleges LoginPage-rol

3.3. Főoldal és tartalma

Az oldalnak két főoldala van: az egyik a tanár, a másik a diák felhasználója számára.

A diák főoldalán egy navigációs sáv található, két fő ablak található közepén a weboldalon. A bal oldali ablakban a diák a tanár által összerakott tesztek között találja magát, amelyeket kitöltendők. A jobb oldali ablakban pedig a diák összes írt, javított és pontozott dolgozatát láthatja.

A tanár oldalán hasonlóan egy navigációs sáv és közepén két fő ablak található. A bal oldali ablakban a tanár a Dolgozatok létrehozásához jut, ahol össze tudja állítani a dolgozatokat, rákattintás után. A jobb oldali ablakban pedig az eddig létrehozott dolgozatokhoz fér hozzá a tanár rákattintás után.



2. ábra a főoldalról

3.4. Funkciók ismertetése

3.4.1. Dolgozatok létrehozása

Csak a tanár fiókok tudnak létrehozni teszteket. Itt hozhat létre dolgozatot.
A bal oldali ablakra kattintva elkészítheti a dolgozatát, ahol meg kell adnia melyik osztálynak milyen tantárgyból egy főcímet a dolgozatnak és hogy mennyi idő áll rendelkezése diákoknak a dolgozatot megíratni.

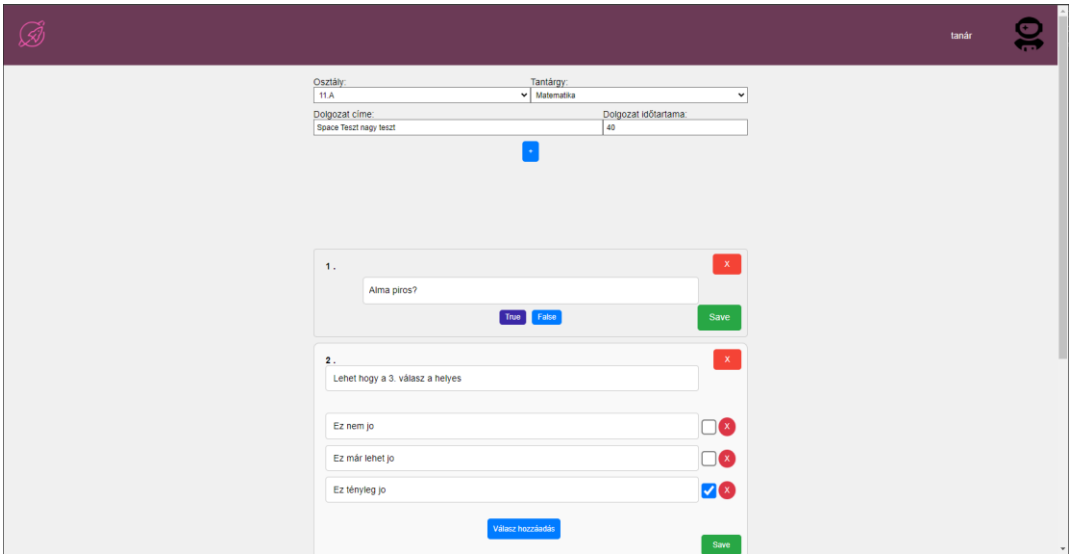
A “+” gomb megnyomása után 3 lehetőség közül választhatunk: ”Igaz/Hamis”, ”Négylehetőséges”, ”Több helyes válasz”.

„Igaz/Hamis”: Hozzá adhatunk Igaz/Hamis típusú kérdést, ahol a diákoknak el kell dönteniük, hogy egy állítás igaz vagy hamis.

„Négylehetőséges”: Hozzá adhatunk többválasztásos kérdéseket, ahol négy lehetőség közül kell kiválasztani a helyes választ. A válaszlehetőség szöveges formában megadhatók.

„Több helyes válasz”: Hozzá adhatunk egy kérdést, ahol több helyes válasz is lehetséges, például hat lehetőség közül három a helyes.

A „Save” gombot megkell nyomni ha feladat késznek gondolja a tanár.
“X” gombbal pedig tudja torolni a kérdéseket. A “Dolgozat véglegesítés” gombbal adhatja ki a feladatsort a diákoknak. Nincsen megkötve, hogy mennyi kérdésnek kell lennie minimum és maximum. Minden válasz lehetőség 1 pontot ér.

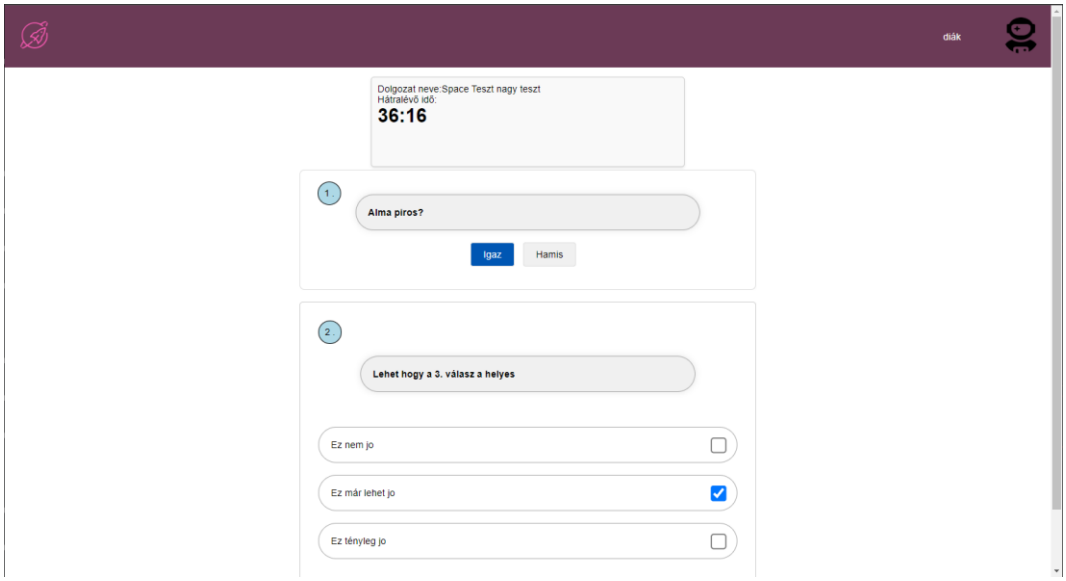


3. ábra a Dolgozat készítés

3.4.2. Dolgozatok megírása

A diák főoldalakon bal oldali ablakra kattintva akkor dobja őt a nem megirt tesztjéhez és az itt felsorolt tesztek közül, ha kattintint az egyik tesztre amin láthatja a dolgozat nevét és létrehozás dátumát. Miután rákattintott bedobja a dolgozat megíróhoz és elindul egy számláló, ami, ha lejár akkor a dolgozatot automatikusan beadja az eddig megadott válaszokkal.

A dolgozat megírása nagyon egyszerű bedobja a feladatok kérdéseket előre megadót válaszokkal és csak a válaszok közül kell választani az allábi válasz lehetőségek közül kell a szerinte helyes válaszokat bejelőlni. Ha diák úgy érzi, hogy a szerinte helyes válaszok megadása után alul a kattintva „Dolgozat Beküldésé” zöld gombra akkor tudja leadni a dolgozatot.



4. ábra a Dolgozat megírása

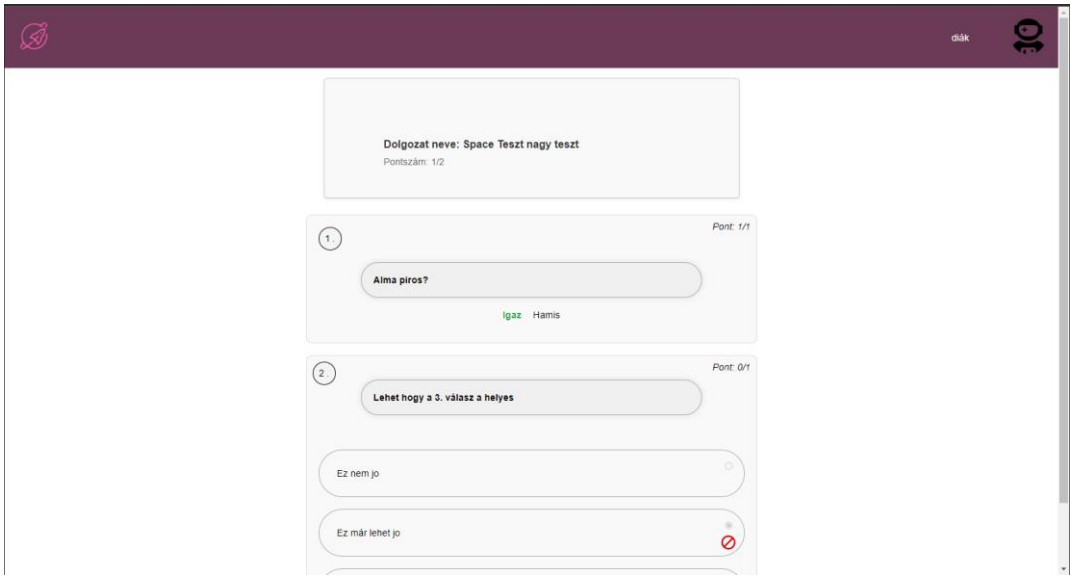
3.4.3. Dolgozatok javítása

Minden dolgozatot automatikusan kijavít a weblap és egyszere lepontozza a tanárnak és a diáknak is egyszere az összes feladatot.

3.4.4. Dolgozatok megtekintése

Ha a diák leadta a dolgozatot vagy lejárt az ideje akkor automatikusan az eddigi megírt tesztjeihez fogja dobni, ahol megtudja tekinteni, hogy hány pontot ért el és ha hibás válaszokat adót akkor megtudhatja hol válaszol rosszul és mit kellett volna válaszolnia.

A tanári oldalon ez úgy nézz ki, hogy a diák tesztjeit szeretné megtekinteni akkor a jobb oldali ablakra kattintva akkor átdobja az íratott dolgozataim oldalra. Ott egy szűrővel fog találkozni, ahol az összes osztály közül választhat és azon belül a tantárgyakat és igy tud szúrni, hogy csak azokat a teszteket mutassa, amiket a szűrő beállít. Itt ugyan azt fogja látni, mint a diák felhasználó a dolgozat nevét a maximum pontszámból elért pontszámot és az összes feladatot kijavítva láthatja.

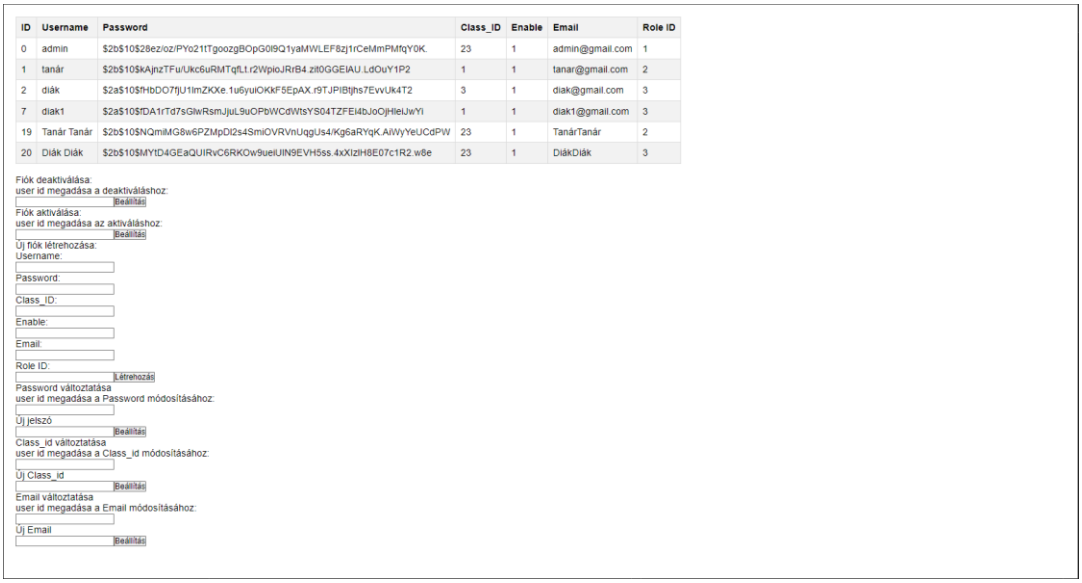


5. ábra a dolgozat javításáról

3.4.5. Admin lehetőségek

Az admin-nak lehetősége van listázni, módosítani, hozzáadni, illetve törölni adatokat. A “UserEditor” menüben láthatja az összes felhasználót. A „Beállítás” gombbal módosíthatja adataikat, kivéve a felhasználóneveket az adatintegritás megőrzése végett. Lehetőség van a deaktiválásra is, ami után az a felhasználó nem tud bejelentkezni. Az új gombbal létrehozhatóak diákok és tanárok.

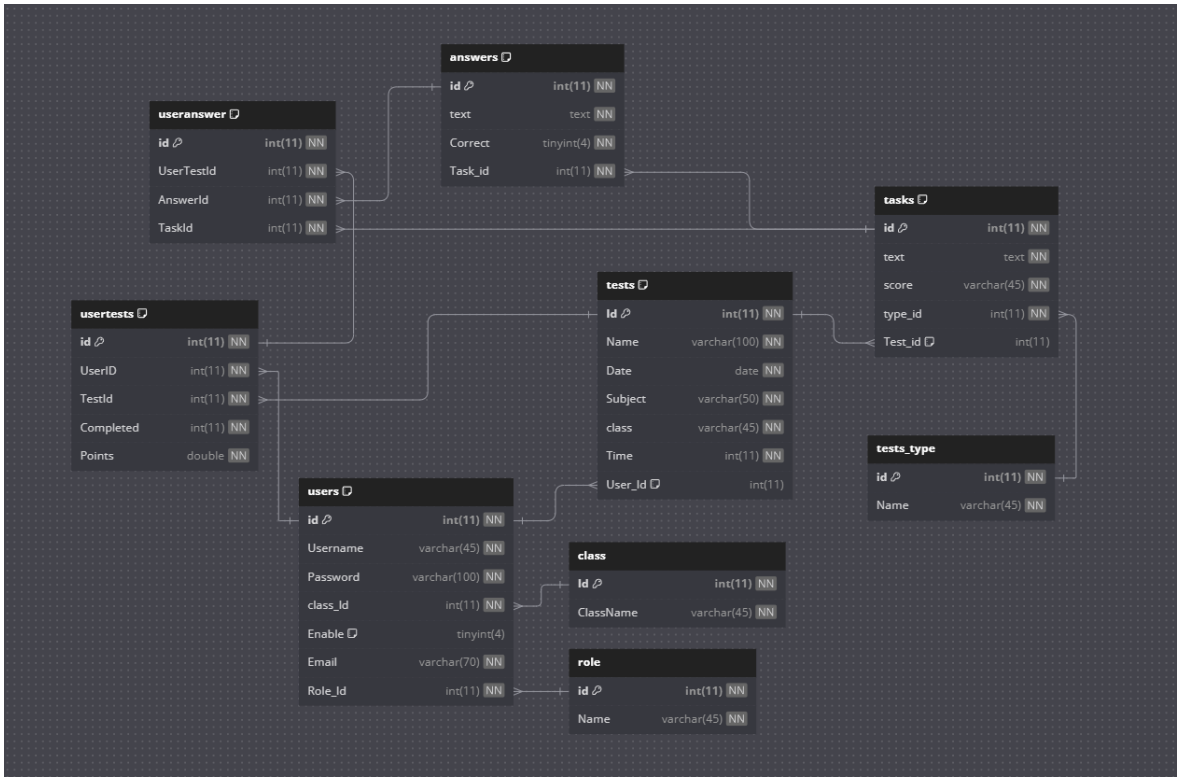
Az osztályokat is tudja kezelni. A következő „ClassEditor”-ban megjelenik az összes osztály. Létrehozható új osztályt és törölhető is.



6. ábra a “UserEditor” *admin*

4. Fejlesztői dokumentáció

4.1. Az adatbázis táblái és kapcsolatai



7. ábra az adatbázis struktúrájáról

Az alkalmazáshoz tartozó adatok tárolására SpaceTeszt_db nevű adatbázis nyújt lehetőséget. Az adatbázisnak a lokális szerveren kell futnia jelszó nélkül. Jelenleg az adatbázis 9 darab táblából és közöttük 11 kapcsolatból áll. Továbbá alapból tartalmaz adatokat, amelyek szükségesek a működéséhez teszteléséhez.

4.1.1. users tábla

➔ A felhasználók tárolására szolgáló tábla.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
Username	VARCHAR, PK, NN	Felhasználónév
Password	TEXT, NN	Jelszó
Enable	INT, NN, def.:1	Státusz (logikai)
ClassID	INT, FK, NN	Osztályának azonosítója
Email	TEXT	Email cím
RoleID	INT, FK, NN, def.:0	Jogosultságának azonosítója

4.1.2. role tábla

➔ Ez a tábla segít elkülöníteni a jogosultságokat a felhasználók között. Jelenleg ez az admin tanár és a diák.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
Name	VARCHAR, NN	Jogosultság Megnevezése

4.1.3. class tábla

➔ Segédtábla, itt tárolja a program az osztályokat

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
ClassName	VARCHAR, NN	Osztály megnevezése

4.1.4. tests tábla

➔ A kiadott dolgozatok mentésére szolgáló tábla.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
Name	TEXT, NN	A dolgozat címe
Subject	INT, FK, NN	Tantárgy azonosítója
Class	INT, FK, NN	Osztály azonosítója
Date	DATETIME, NN	Dolgozat létrehozás dátuma
Time	INT	Dolgozat időtartama
User_id	VARCHAR, FK, NN	A felhasználó azonosítója

4.1.5. tasks tábla

➔ Minden dolgozatnak vannak kérdései, ezek tárolását teszi lehetővé.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
Test_ID	INT, FK, NN	Teszt azonosítója
Text	TEXT, NN	Maga a kérdés
Score	INT, NN	A kérdés pontértéke
Type_ID	INT, FK, NN	Típusának azonosítója

4.1.6. task_types tábla

➔ Segédtábla a kérdéstípusok tárolására.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
Name	VARCHAR, NN	Kérdés típusának megnevezése

4.1.7. answers tábla

➔ A kérdésekhez tartozó válaszlehetőségeket tárolja.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
Task_ID	INT, FK, NN	Kérdés azonosítója
Text	TEXT, NN	Maga a válasz
Correct	INT, NN, def.:0	Helyes-e (logikai)

4.1.8. usertests tábla

➔ Ez a tábla kapcsolja össze a dolgozatokat a diákokkal, hogy kinek melyikeket kell megírnia, illetve tárolja megírás után az eredményt.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
UserID	VARCHAR, FK, NN	Felhasználó azonosítója
TestID	INT, FK, NN	Teszt azonosítója
Completed	INT, NN, def.: 0	Leadási állapota (logikai)
Points	INT, NN	Elért pontszám

4.1.9. useranswer tábla

➔ Ebben a táblában a diákok válaszai kerülnek mentésre.

Mezőnév	Típus	Megjegyzés
ID	INT, PK, NN, AI	Egyedi azonosító
UserTest_ID	INT, FK, NN	Felhasználó dolgozatának azonosítója
Answer_ID	INT, FK, NN	A válasz azonosítója
Task_ID	INT, FK, NN	Kérdés azonosítója

4.1.10. Összes kapcsolat az adatbázisban

answers tábla kapcsolatai:
- answers.Task_id → tasks.id

tasks tábla kapcsolatai:
- tasks.Test_id → tests.Id
- tasks.type_id → tests_type.id

tests tábla kapcsolatai:
- tests.User_Id → users.id

useranswer tábla kapcsolatai:
- useranswer.TaskId → tasks.id
- useranswer.UserTestId → usertests.id
- useranswer.AnswerId → answers.id

users tábla kapcsolatai:
- users.class_Id → class.Id
- users.Role_Id → role.id

usertests tábla kapcsolatai:
- usertests.TestId → tests.Id
- usertests.UserID → users.id

4.2. Components

Újra felhasználhatóság: Komponenseket lehet újra felhasználni különböző részekben és funkciókban a projektben, ami csökkenti a kód ismétlődését és segít az alkalmazás karbantarthatóságának növelésében.

Modularitás: A komponensek moduláris szerkezetűek, ami lehetővé teszi, hogy az alkalmazást kisebb részekre bontsuk, így könnyebb megérteni és fejleszteni.

Könnyű karbantarthatóság: Az egyes komponensek önállóan működnek, így könnyen lehet őket tesztelni, fejleszteni és javítani anélkül, hogy az egész alkalmazást érintené.

Átláthatóság: A komponensek segítik a weblap strukturáltabbá tételét, mivel minden egyes részegység felelős egy adott funkcióért vagy feladatért.

A szakdolgozatom készítésé alatt sok komponenst használtam szám szerint 18db.

Név szerint a legfontosabbak:

4.2.1. LoginForm

„**LoginForm**”: A felhasználó bejelentkezésére szolgáló űrlap, ahol felhasználónév és jelszó megadása szükséges.

4.2.2. Navbar

„Navbar”: Az oldal tetején elhelyezett navigációs sáv, amelyen keresztül a felhasználók könnyen elnavigálhatnak az oldal különböző részei között.

4.2.3. TasksWriterComponents

„**MultipleAnswersWriter**”: Olyan feladattípus, amelyben a Diákok több válaszlehetőség közül kell kiválasztaniuk a megfelelőt.

„**OneAnswerWriter**”: Egy válaszlehetőséget kínáló feladattípus, ahol a Diákoknak csak egy helyes választ kell kiválasztania.

„**TrueFalseWriter**”: Igaz-hamis típusú feladat, ahol a Diákoknak meg kell állapítania, hogy egy állítás igaz-e vagy hamis.

4.2.4. TasksComponents

„**MultipleAnswers**”: A MultipleAnswers feladat típusának a szerkesztője, ahol a Tanár új feladatokat állíthatnak össze több válaszlehetőséggel.

„**OneAnswer**”: Az OneAnswer feladattípus szerkesztője, ahol a Tanár új feladatokat hozhatnak létre egy válaszlehetőséggel.

„**TrueFalse**”: A TrueFalse feladattípus szerkesztője, ahol a Tanár új igaz-hamis típusú feladatokat hozhatnak létre.

4.2.5. TestBoard

„**TestBoard**”: Egy felület ahol a Tanár a dolgozatot létrehozza és kiválasztja hogy melyik Osztály, melyik Tantárgy, mi legyen a Dolgozat címe, és a Dolgozat időtartamát adja meg és a három feladat tipus kozul választhat hogy mik szerepeljenek a dolgozatban.

4.2.6. TestBoardWriter

„**TestBoardWriter**”: Itt irják meg a Diákok a Tanár által összeállított dolgozatot

4.2.7. TasksResult

„**TasksResult**”: A teljesített dolgozatokat tekinthetik meg a felhasználók a kijavítva és lepontozva

4.2.8. TestsWindowUnCompletedTest

„**TestsWindowUnCompletedTest**”: Egy ablak vagy oldalrész, ahol a felhasználók megtekinthetik a be nem fejezett teszteket.

4.2.9. Timer

„**Timer**”: Időmérő funkció, amely lehetővé teszi a Tanárnak hogy a dolgozat időtartamát meghatározza hogy mennyi idő áll rendelkezésre a Diákoknak a dolgozat leadására.

4.3. Szerver-Kliens Kapcsolat (API)

4.3.1. API

Az API (Application Programming Interface) egy olyan interfész, amely lehetővé teszi az alkalmazások közötti kommunikációt. A szerver-kliens kapcsolat alapvetően az alábbiakon alapul:

Protokoll: Általában HTTP vagy HTTPS protokollon keresztül történik a kommunikáció.

Kérés-Felelet Modell: A kliens küld egy kérést a szervernek, ami válaszol a kérésre.

Metódusok: „GET”, „POST”, „PUT”, „DELETE” stb. segítségével történik az adatok kezelése.

Adatformátumok: Általában JSON vagy XML formátumban küldik az adatokat.

Hitelesítés és Biztonság: Felhasználói hitelesítésre és adatbiztonságra szolgáló mechanizmusok alkalmazása.

Dokumentáció: Részletes leírás a rendelkezésre álló végpontokról és használati utasításokról.

Tesztelés és Hibakezelés: Az API-k alapos tesztelése és megfelelő hibakezelési mechanizmusok alkalmazása.

4.4. GitHub

A GitHub a szakdolgozat készítése során több okból is hasznos volt:

Verziókezelés: Lehetővé teszi a dokumentumok verzióinak nyomon követését és visszaállítását, így biztonságban tudathattam a munkám.

Dokumentáció: Markdown formátumú dokumentumok kezelésére alkalmas, így könnyedén formázhattam és strukturálhattam a szövegeket.

Projektmenedzsment: Segítséget nyújtót a feladatok nyomon követésében és a projektek hatékonyabb menedzselésében.

Hozzáférés és biztonság: Könnyen kezelhetem hogy ki férhet hozzá a dokumentumokhoz, és a GitHub automatikusan menti és biztonságosan tárolja azokat.

Összességében a GitHub lehetővé tette a dokumentumok nyomon követését és a projektmenedzsmentet a szakdolgozat készítése során.

4.5. Jelszó titkosítás

A szakdolgozatom készítése során a bcrypt-et használtam a felhasználók jelszavaknak biztonságos tárolására és ellenőrzésére. Az alábbi kód részlete a backend része a server.js fájlban:

A bcrypt egy kriptográfiai hash funkció, amelyet a jelszavak biztonságos tárolására és az azokkal való azonosításra használtam. A funkció nagy előnye, hogy lassú, ami megnehezíti a brute force vagy rainbow támadásokat, ahol a támadók több próbálkozást tesznek a helyes jelszó megtalálására.

4.5.1. Hogyan működik:

Salt generálása: A bcrypt véletlenszerűen létrehoz egy ún. "sót", amelyet a jelszóval együtt használ. Ez a só egyedi az adott jelszóhoz, és segít megakadályozni az előre elkészített táblázatok (rainbow tables) használatát a támadók részéről.

Hash generálása: A jelszó és a só kombinációját több körben hasheli. Ezeket a köröket "költségfaktornak" is nevezik. Minél magasabb a költségfaktor, annál hosszabb ideig tart a hashelés, és annál nehezebb a támadóknak feltörni a jelszót.

Hash tárolása: A végső hash-távot tárolják, amely tartalmazza mind a sót, mind a hashelt jelszót. Ez az érték kerül mentésre az adatbázisba.

A szakdolgozatomban használtam a felhasználók beléptetésére. Az app.post('/users', ...) részben a felhasználó által megadott jelszót a bcrypt segítségével hasonlítom össze a tárolt, hashelt jelszóval a adatbázisban. Ha a jelszavak megegyeznek, akkor JWT token-t generálok és visszaküldöm a felhasználónak, különben hibaüzenetet küldök vissza.

4.6. Autentikáció

A szerver oldalon az autentikáció egy folyamat, amely a felhasználók beazonosítását és hitelesítését szolgálja. A folyamat a következő lépésekből áll:

- **Felhasználó azonosítása:** Amikor a felhasználó bejelentkezik, a szerver ellenőrzi az adatbázisban, hogy létezik-e a megadott felhasználónévvel rendelkező felhasználó.

- **Jelszó ellenőrzése**: Ha a felhasználó létezik az adatbázisban, a szerver ellenőrzi a megadott jelszót a tárolt jelszóval szemben. Ez általában egy hash-függvény használatával történik, hogy a jelszavak ne tárolódjanak nyílt szövegként.

- **Token létrehozása**: Ha a jelszó helyes, a szerver létrehoz egy JSON Web Token (JWT) -t. Ebben a tokénben általában azonosító információk, például a felhasználó azonosítója (id) és a jogosultsági szintje (roleid), vannak kódolva.Válasz küldése a kliensnek: A szerver visszaküldi a kliensnek a JWT tokent. Ezt a tokent a kliens későbbi kérések során használja felhasználói azonosításra és hitelesítésre.

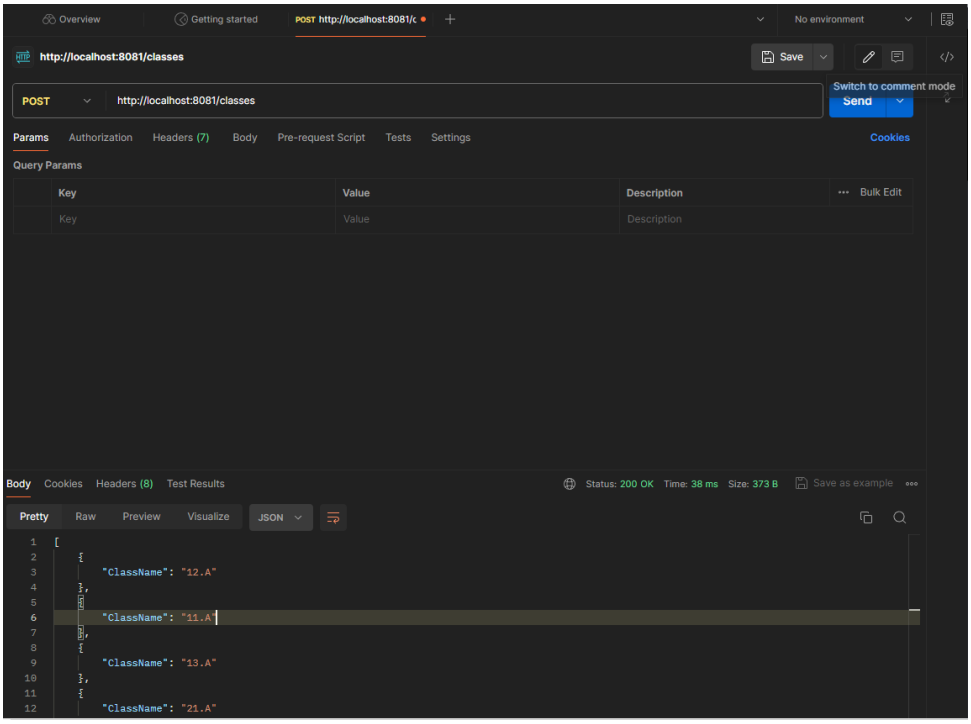
- **Süti vagy más azonosító küldése**: A szerver általában egy sütit küld vissza a kliensnek azonosítás céljából. Ez lehetőséget ad arra, hogy a kliens további kéréseket küldjön a szervernek anélkül, hogy újra be kellene jelentkeznie.

Ezek az autentikáció fő lépései, amelyeket a szerver.js oldalon valósultja meg, míg a kliens oldalon ezeket a tokent kezeli és használja az azonosításhoz és a jogosultságok kezeléséhez.

4.7. Tesztdokumentáció

A fejlesztés során rendszeresen teszteltem a kódot, és ezt párhuzamosan végeztem a fejlesztési folyamattal. Miután elkészült a program, többször is kipróbáltam annak működését különböző helyzetekben, hogy meggyőződjek a stabilitásáról és funkcionalitásáról.

Az API-khoz történő kérések ellenőrzésére és validálására a terminálban consologok segítségével és postmanban tettem. Ezek a konzolüzenetek segítettek abban, hogy könnyedén futtathassam és ellenőrizzem az összes API hívást, biztosítva azok helyes működését.



8. ábra az Postman api tesztelés

5. Az weblap design-ja

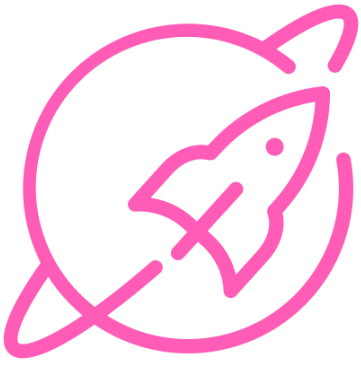
A weblap kitalálásánál még nem tudtam, hogy milyen kinézetett szeretnék csinálni.

Neki álltam a weblap elkészítésének és átgondolva mivel egyszerű feladatok lesznek benne ezért inkább ez az általános iskolás diákoknak fog készülni azért merült fel bennem a kérdés, hogy a kisgyerekek mik szeretnének lenne, ha felnőnek és akkor jött az „űrhajós” téma és ezért választottam ezt a gyerek barát űrutazás űrhajós témát a weblapnak.

A kinézetet folyamatosan készítettem a kódolással párhuzamosan, mivel gyakran ez is befolyásolta azt, hogy mi és hogyan fog működni az egyes oldalakon.

A felhasznált ikonokat képeket flaticon-rol és freepik-röl és Ai segítségével generáltat majd photoshopban átszerkeszt, átszínezve képeket használtam fel. A logó ötleténél először a StarTrek-hez hasonló ikonok között gondolkodtam és végül megtaláltam a számomra ideális logót. Az oldal nevével kapcsolatban össze akartam rakni a 2 fő elemet a urutazás témát és a dolgozat irás és igy találtam ki az oldal nevét „Space Teszt” Ami magyarul annyit jelent hogy „Űr Teszt”.

A designhoz teljesen saját CSS-t csináltam nem használtam olyan előre elkészített komponenseket, mint például a MUI.



9. ábra az weblap logójáról

6. Összefoglalás

6.1. A szakdolgozat célja

A szakdolgozatom célja egy olyan weblap volt, amellyel a tanárok egyszerűen irathatnak dolgozatokat a diákjaikkal. Egy ilyen alkalmazás nagyon jól tud jönni akár az online oktatás alatt is. Segít osztályozni, illetve felmérni az aktuális tudását a diákoknak.

6.2. Megvalósítása

Ahogy a projekt fejlődött, számos kihívással és problémával találkoztam, amelyek megoldása során sokat tanultam. A design tervtől a backend fejlesztéséig folyamatosan haladtam előre, és minden lépésnél újabb részleteket kellett kidolgoznom.

Miután elkészültem a design tervekkel, az adatbázis tervezése és implementálása volt a következő lépés. Az adatok strukturálása és kezelése kulcsfontosságú volt ahhoz, hogy az alkalmazás megfelelően működjön. A kódolás során folyamatosan bővítettem az adatbázist az alkalmazás igényeinek megfelelően.

A bejelentkezési felület kialakítása a biztonság és az autentikáció szempontjából fontos lépés volt. A jelszavak titkosítása és a felhasználók jogosultságainak kezelése az adatbiztonság egyik alappillére.

A főoldal és a keretrendszer létrehozása során az volt a célom, hogy egy átlátható és könnyen navigálható felületet hozzak létre. A React segítségével létrehoztam egy rugalmas és dinamikus struktúrát, amely lehetővé tette a különböző oldalak és komponensek könnyű kezelését és módosítását.

A homepage-k kialakítása közben számos kihívással találkoztam, de sikerült megoldást találnom rájuk az AI és a StackOverflow segítségével. Fontos volt, hogy az oldal összképe megfeleljen a tervezési elképzeléseimnek, és a felhasználók számára is intuitív legyen a használata.

A dolgozat létrehozásának oldala volt az egyik legnehezebb része a projektnek. Itt kellett külön komponenseket létrehoznom minden feladattípushoz, és dinamikusan megjelenítenem őket a tanár választása szerint. Ez a rész különösen sok kommunikációt igényelt a frontend és a backend között.

A backend fejlesztése során az összes API hívást a server.js-ben valósítanom meg. Az adatok kezelése, tárolása és frissítése volt a legnagyobb kihívás, különösen a kérdések, válaszok és helyes válaszok tárolása és azok kapcsolódása a dolgozatokhoz.

A diák oldalon a teszt megírása is fontos volt, és gondoskodnom kellett arról, hogy a diákok könnyen megtalálják és megírják a rájuk váró teszteket.
A dolgozat megírásának oldalán pedig kiemelten fontos volt az instrukciók és feladatok megfelelő megjelenítése és kezelése.

Végül az admin felületen az osztályok és felhasználók kezelése volt a fő feladat, és itt is nagy hangsúlyt kellett fektetnem az adatok kezelésére.

Az egész fejlesztési folyamat során rengeteg tapasztalatot szereztem, és a projekt végére egy komplex és jól működő alkalmazást hoztam létre, amelynek számos funkciója van és kielégíti a felhasználók igényei.

7. Fejlesztési lehetőségek

7.1. Több Interaktív feladatlap elemek hozzáadása

- **Párosítás:** Két elem összekapcsolása valamilyen összefüggés alapján
- **Sorrendbe rendezés:** Válaszlehetőségek, melyek közül csak egy helyes válasz jelölhető
- **Táblázat**: A táblázat hiányzó adatainak pótlása a megfelelő információkkal
- **Halmazba rendezés:** A megadott elemek rendezése az előre meghatározott halmazokba

7.2. Feladatmentés automatizálása

Ahelyett, hogy külön „save” gombra kellene kattintania a Tanárnak a mentéshez, az alkalmazás automatikusan menthetné az előrehaladást. Például, amikor egy Tanár félben hagyja a dolgozat létrehozást kilép az oldalról, az alkalmazás automatikusan elmenti az eddigi munkáját.

7.3. Profilbeállítások

7.3.1. Jelszó módosítás

Adjon lehetőséget a felhasználóknak, hogy könnyen módosíthassák jelszavukat a profilbeállítások menüpont alatt.

7.3.2. Kétlépcsős azonosítás

Opcionálisan kínálja a kétlépcsős azonosítást a biztonságosabb hozzáférés érdekében.

8. Irodalomjegyzék

Fénykép források:

- https://www.freepik.com/

Tervezéshez és szerkesztéshez:

- https://www.figma.com/

Programozási problémák megoldásai:

- https://stackoverflow.com
- https://chat.openai.com/
- https://react.dev/reference/react
- https://www.w3schools.com/nodejs/default.asp

Ikonok:

- https://www.flaticon.com/

9. Mellékletek

9.1. Login server.js-ben

```
app.post('/users', (req, res) => {
    const sql = 'SELECT * FROM users WHERE Username = ?';
    db.query(sql, [req.body.username], (err, data) => {
        console.log(data)
        if (err) return res.json({Error: "Bejelentkezési hiba a
szerveren"});
        if(data.length > 0) {


            bcrypt.compare(req.body.password.toString(), data[0].Password,
(err, response) => {
                if(err) return res.json({Error: "Jelszó összehasonlítási
hiba"});
                if (response)
                 {
                        const id = data[0].id;
                        const roleid = data[0].Role_Id;

                        const token = jwt.sign({id,roleid}, "jwt-secret-
key", {expiresIn: 84600});

                        res.cookie('token', token, { httpOnly: true });
                        return res.json({ token });
                } else {
                    return res.json({Error: "Hibás Felhasználó vagy
Jelszó"});
                }
            })
        } else {
        return res.json({Error: "Hibás Felhasználó vagy Jelszó"});
        }})


    })
```

9.2. 3 TestsType dolgozat létrehozásakor server.js-ben

```
for (let i =0;i<tasks.length;i++)
    {
        // feladatok

        sql = `INSERT INTO tasks ( text,score,type_id, test_id) VALUES
(?,1,?,?)`;
        db.query(sql, [tasks[i].question,tasks[i].questionType,dolgozatid],
(err, datatask) => {
        if (err) {
            console.error(err);
            return res.json({ Error: 'Hiba a válaszok beszúrásakor' });
        }

        let taskid = datatask.insertId

        //válaszok
        if (tasks[i].questionType ==1)
        {
            //truefalse
            sql = `INSERT INTO answers ( text,correct, task_id) VALUES
('igaz',${tasks[i].correctanswer  ? 1 : 0},${taskid})`;
            db.query(sql, [req.body.tasks], (err, dataanswer) => {
                if (err) {
                    console.error(err);
                    return res.json({ Error: 'Hiba a válaszok beszúrásakor'
});
                }})
            sql = `INSERT INTO answers ( text,correct, task_id) VALUES
('hamis',${tasks[i].correctanswer ? 0 : 1},${taskid})`;
            db.query(sql, [req.body.tasks], (err, dataanswer) => {
                if (err) {
                    console.error(err);
                    return res.json({ Error: 'Hiba a válaszok beszúrásakor'
});
                }})

        }
```

```
  if (tasks[i].questionType ==2)
        {


            //multianswer
            for (let j = 0; j<tasks[i].answer.length;j++)
            {

                sql = `INSERT INTO answers ( text,correct, task_id) VALUES
(?,${tasks[i].correctanswer.includes(j+1) ? 1 : 0},${taskid})`;
                db.query(sql, [tasks[i].answer[j]], (err, dataanswer) => {
                    if (err) {
                        console.error(err);
                        return res.json({ Error: 'Hiba a válaszok
beszúrásakor' });
                    }})
            }

        }

        if(tasks[i].questionType ==3)
        {
            //oneanswer
            for (let j = 0; j<tasks[i].answer.length;j++)
            {
                sql = `INSERT INTO answers ( text,correct, task_id) VALUES
(?,${tasks[i].correctanswer == j+1 ? 1 : 0},${taskid})`;
                db.query(sql, [tasks[i].answer[j]], (err, dataanswer) => {
                    if (err) {
                        console.error(err);
                        return res.json({ Error: 'Hiba a válaszok
beszúrásakor' });
                    }
                })
            }
        }

    })
```

9.3. TrueFalse componens

```
import React, { useState } from 'react';
import './TrueFalse.css';


const TrueFalse = ({questionNumber, questionValue, onResponse, removeTask})
=> {
  const [question, setQuestion] = useState(' ');
  const [answer, setAnswer] = useState(null);

  const handleQuestionChange = (e) => {
    setQuestion(e.target.value);
  };


  const handleTrue = () => {
    if(question !== '') {
      setAnswer(true);
    }
  };


  const handleFalse = () => {
    if(question !== '') {
      setAnswer(false);
    }
  };


  const handleSave = () => {
    onResponse(question, answer, questionNumber, answer);
  };


  const handleRemove = () => {
    removeTask(questionNumber);
  };
```

```
  return (
    <div className="TrueFalse-container">
      <div className="TrueFalse-Task-serial-number">{questionNumber} .
</div>
      <label className="Label-TrueFalse-Question">
        <input className='Input-TrueFalse-Question' type="text"
placeholder={`Ide írja a választ`}
          value={question === " " ? questionValue : question}
          onChange={handleQuestionChange}
          />
      </label>
      <div className="Answer-TrueFalse-Buttons">
        <button onClick={handleTrue} className={`${answer ? 'selected' :
''}`}>True</button>
        <button onClick={handleFalse} className={`${answer === false ?
'selected' : ''}`}>False</button>
      </div>

      <button className="Remove-TrueFalse-Button"
onClick={handleRemove}>X</button>

       <button className='Save-Button' onClick={handleSave}>Save</button>
    </div>
  );
}
export default TrueFalse;
```

9.4. Jelszó titkosítása

```
const plainTextPassword = 'tanár';
bcrypt.hash(plainTextPassword, salt, function(err, hash) {
    if (err) {
        // Handle error
    } else {
        // Store the hash in the database



    }
});
```