VIZSGAREMEK

STUDENT SPOTLIGHT

- School Management System

Készítette: Forrás Márk

Budapest, 2024.04.03

Tartalom

1. School Management System fejlesztése

A rendszer fejlesztésének célja egy olyan felület, amely alternatívaként szolgálhatna az eKréta és hasonló robosztus rendszerek mellett, illetve megkönnyítené az adminisztratív feladatok elvégzését.

A Student Spotlight több felhasználói szintű: tanuló, tanár, iskolai admin, rendszer admin. A célja az adminisztratív feladatok egyszerűbb elvégzése a specifikus iskola, azon tanulói, tanárai és osztályainak kezelése.

Student Spotlight nem csupán egy újabb alkalmazás a piacon, hanem egy eszköz az oktatási intézmények számára, amely segítheti az adminisztrációt és támogatja mind a tanárokat, mind a diákokat. Célunk, hogy hozzájáruljunk az oktatás hatékonyságához és átlátható eredményekkel szolgáljunk, biztosítva a közösségi együttműködést és az egyszerűbb folyamatokat az oktatási környezetben.

A projekt GitHub elérhetősége:

https://github.com/gl4s/StudentSpotlight

Installáció:

1. A projekt tetszőleges mappába való leklónozása után:

git clone https://github.com/gl4s/StudentSpotlight

2. A szükséges lokális adatbázist XAMPP-al kell megvalósítani, az Apache és MySQL szerver indítása után a phpMyAdmin felültén keresztül kell importálni a következő sql fileokat a StudentSpotlight mappából ilyen sorrendben:

- studentspotlight.sql
- studentspotlight_dump_file.sql

Sikeres importálás után visszatérhetünk a StudentSpotlight főmappába.

A backend és frontend telepítése következik.
Terminálból navigáljunk StudentSpotlight főmappába, majd adjuk ki az „npm install” parancsot. Ezt követően a „cd client” paranccsal a frontendes mappába navigálva szintén „npm install” paranccsal telepíthetjük a szükséges kliensoldali csomagjaink.

A projekt indításához a client mappából „cd ..” paranccsal kilépve navigáljunk a StudentSpotlight mappába.
Itt az „npm start” paranccsal indíthatjuk el a projektet.

2. Alkalmazott szoftverek és technológiák

2.1 Szoftverek:

- XAMPP:
  phpMyAdmin 5.2.1
  MySQL 10.3.38,
  Apache 2.4.58,
  MariaDB 10.4.32
- Google Chrome Verzió: 123.0.6312.86
- Microsoft Edge Verzió: 123.0.2420.65
- Microsoft Word 2021 LTSC
- Adobe Photoshop 2023
- Github Desktop v3.3.8
- Visual Studio Code 1.87.2

2.2 Programnyelvek, keretrendszerek és modulok:

Frontend

- **Nyelv:** JavaScript
- **Keretrendszer:** React.js
- **UI Komponenskönyvtár:** Bootstrap
- **Routing:** React Router DOM

Backend

- **Nyelv:** JavaScript (Node.js)
- **Web Framework:** Express.js
- **Adatbázis Kezelő:** MySQL
- **Authentikáció és Autorizáció:** bcrypt, jsonwebtoken
- **HTTP Kliens:** axios
- **CSV Fájlkezelés:** csv-parser
- **Környezeti változók kezelése:** dotenv
- **Middleware:** body-parser, cors, multer
- **Egyéb:** assert (beépített Node.js modul)

Tesztelés: Cypress.JS segítségével

2.3 Megvalósítás

Megvalósítás otthonról történt, saját munkaeszközökön, iskola után, illetve azokon a tanórákon, amelyeken a munka megengedett volt.

A felület megvalósítása a RESTful API elveire épült, biztosítva a frontend és backend közti kommunikációt. A verziókövetés a Github segítségével történt meg az előírtak szerint. Három fajta böngészőre került sor a tesztelések folyamán, Microsoft Edge, Google Chrome és Firefox legújabb verziói. A fejlesztést a Visual Studio Code-ban hajtottuk végre. Funkciói és használata a 3. fejezetben leírtak alapján.

Az elkészült felület bár nem teljes értékű, adminisztratív feladatok ellátására képes, kezelése nem igényel szükséges informatikai előismereteket.

Kihívást jelentettek az új technológiák, mint pl. a React és moduljai ismeretének alapos elsajátítása, azon hibáinak kezelése, az egy főre jutó munka mennyisége, mind a backend és frontend tervezése és a projekt kivitelezése ezen körülmények között.

3. A felület funkciói, cél felépítése és használata

- Négy felhasználói fiókhoz kötött jelszavas belépés Authentikációval és Authorizációval: Rendszergazda, Iskolai Adminisztrátor, Tanár, Tanuló.
- Iskolák regisztrációja.
- Regisztrált Iskolához kötött Tanári és Tanulói regisztráció.
- Regisztrált Iskolához kötött felhasználói adat törlés, és módosítása:
  A regisztrált Iskola, Tanárok, és tanulók részére.
- Regisztrált Iskolához kötött osztályok létrehozása, osztályok módosítása, osztályok törlése, Tanulók hozzárendelése és törlése ezen osztályokból.
- Regisztrált iskolákhoz kötött tanárok hozzárendelése a globálisan elérhető tantárgyakhoz, ezen tanár-tantárgy hozzárendelések törlése.
- Rendszergazda felveheti a globálisan elérhető tantárgyakat, amelyek mindegyik Regisztrált Iskola számára elérhetőek, és törölheti is azokat.

3.1 Jelszavas Belépés folyamata

A felhasználó megnyitja az oldalt. A főoldal fogadja.
3 kártya közül 2 szolgál belépésre, a saját felhasználói fiókjának megfelelőt kell választania:

- Student Page kártyán a „Login” gombra kattintson, ha Tanulói fiókba kíván belépni
- Teacher Page kártyán a „Login” gombra kattintson, ha Tanári, Iskolai Adminisztrátori vagy Rendszergazdai fiókba kíván belépni

Ezen oldalakra lépéskor az azonosításhoz kellő információk megadása után, mint pl.: melyik iskolához van rendelve, a speciális felhasználó neve, kell megadnia jelszavát.

Sikeres jelszavas belépést követően az összes felhasználói fiók átirányításra kerül a számára jogosult fiókjának és rangjának megfelelő főoldalra.

Mivel főleg az Iskola Adminisztrátori fiókról és Rendszergazdai fiókról érhetőek el az elkészült funkcionalitások, így ezekről lesz szó az ezt követő néhány fejezetben.

Példa:

Iskola Adminisztrátori belépés után egy kezelőfelület fogad minket, ahonnan a 3. pontban megjelölt funkcionalitások érhetőek el.

3.2 Iskolák Regisztrációja

A felhasználó megnyitja az oldalt. A főoldal fogadja.

3 kártya közül 2 szolgál belépésre, a „School registration” című kártyán kell rákattintania a „Registration” gombra. Ez ekkor átviszi a /schoolregistration című aloldalra, ahol a regisztrációt megteheti.

Az adatok sikeres beírását követően rákattint a „Register” gombra, majd az oldal visszajelez a regisztráció sikerességéről és átirányítja a főoldalra.

3.3 Regisztrált Iskolához kötött Tanári és Tanulói regisztráció

Belépést követően az iskola Adminisztrátor a kezelőfelületre lesz irányítva (/schooladmin). A kezelőfelületen a „New Student or Teacher” gombra kattintva érheti el a szükséges aloldalt. Ekkor átirányításra kerül a „/newuser” aloldalra, ahol megteheti az új felhasználó felvételét.

3.4 Felhasználói adat törlés, és módosítása: A Regisztrált Iskola, Tanárok, és Tanulók részére.

A Tanári és Tanulói profilok adatmódosítása a kezelőfelületből(/schooladmin) elérhető. Itt a „Member List” gombot kell keressék, amely átirányít a „/members” aloldalra, ahol a Regisztrált Iskolához rendelt összes tanulót- és tanárt láthatják egy táblázatban.
Itt megjelenik a tanuló felhasználóneve, típusa, Kereszt- és vezeték neve, Születési dátuma, azonosítója. Iskola Adminisztrátornak opciója van szűrni Tanárok és Tanulók között.

Minden egyes felhasználó neve mellett egy „Actions” nevű oszlopban megjelennek az adatmódosításhoz szükséges gombok.

Az „Edit” feliratú gombra kattintva egy felugró ablakban betöltenek és módosíthatók a felhasználói adatokat.

A „Delete” feliratú gombra kattintva a felhasználó törlésre kerül az adatbázisból.

Az Iskola Adminisztrátori fiók, amely reprezentálja az iskoláját a kezelőfelületből módosítható. A „School Meta” feliratú gombra kattintva egy aloldalon egy újabb menürendszert érünk el(/schoolmeta), ahol az „Edit School Info” feliratú gombra egy felugró ablakban betöltenek az Iskola Adminisztrátor adatai és amelyek módosíthatóak számára.

3.5 Osztályok Létrehozása, Módosítása, Törlése, Tanulók hozzárendelése és törlése ezen osztályokból.

Az osztályok létrehozását a kezelőfelületen található „School Meta” feliratú gombra kattintva érheti el. Ezután átkerül a „/schoolmeta” aloldalra, és itt az „Add New Class” feliratú gombra kattintva egy felugró ablakon keresztül beírhatja az új osztály nevét és kiválaszthatja annak Osztályfőnökét. Minden tanár csak 1 osztálynak lehet osztályfőnöke.

A szükséges adatok megadása után rákattintva az „Add Class” gombra a rendszer választ küld az osztály hozzáadásának sikerességéről. Ezt követően visszaléphetünk a back gomb megnyomásával a /schoolmeta aloldalra, majd itt az „Active Class” feliratú gombbal megtekinthetjük az aktív osztályokat.

Ez egy újabb felugró ablakban fog megjelenni. Itt egy sorban láthatjuk az aktív osztályokat, és azokra rákattintva fognak táblázatos formában megjelenni az aktív diákok. A dinamikus tábla alján található három gombbal, illetve a tábla utolsó „Select” oszlopaként megjelölt checkbox-al lehet kezelni. A gombok felett helyezkedik el egy legördülő menü, amelyből kiválaszthatjuk a diákokat, akik még nem szerepelnek valamilyen osztályban, majd ezt követően az alatta levő „Add Selected Student” feliratú gombra kattintva hozzáadhatjuk az osztályhoz. Ekkor dinamikusan megjelenik a tábla utolsó helyén.

A „Select” oszlopban kijelölhetünk egy- vagy akár több aktív tanulót is és ezeket a „Remove Selected Student” gombra kattintva eltávolíthatjuk az osztályból. Ekkor visszakerülnek az aktív tanulók közé.

Amikor van egy kiválasztott osztályunk, akkor a „Delete Class” feliratú gomb megnyomásával törölhetjük is azt. Ekkor egy felugró ablak megkérdezi, hogy biztosan törölni akarjuk-e. Az „Igen” -re nyomva kitörlődik az adatbázisból, majd visszajelzést kapunk ennek sikerességéről.

3.6 Tanárok Hozzárendelése és Törlése a globálisan elérhető tantárgyakhoz

A tanár-tantárgy hozzárendeléseket szintén a kezelőfelületről érhetjük el, mint Iskola Adminisztrátor. A „Subject Teacher” feliratú gombra kattintva egy új oldalra kerülünk (/subjectassignment). Itt szintén egy dinamikus, táblázatos formában jelennek meg a hozzárendelések.

Két legördülő menüt láthatunk táblázatunk alatt, az elsőben megjelennek a globálisan elérhető tárgyak, amit a második listából kiválasztott Tanárral fűzhetünk össze az „Assign Subject 11ot he Selected Teacher” feliratú gombbal. Ezt követően dinamikusan megjelenik a listában az új hozzárendelés. Ezt, és az összes hozzárendelést kitörölhetjük az utolsó „Actions” oszlopban található „Delete” feliratú gombra kattintva. Ekkor a rendszer egy megerősítő üzenetet küld a törléssel kapcsolatban, majd ezt megerősítve eltűnik a táblából és adatbázisból is a hozzárendelésünk.

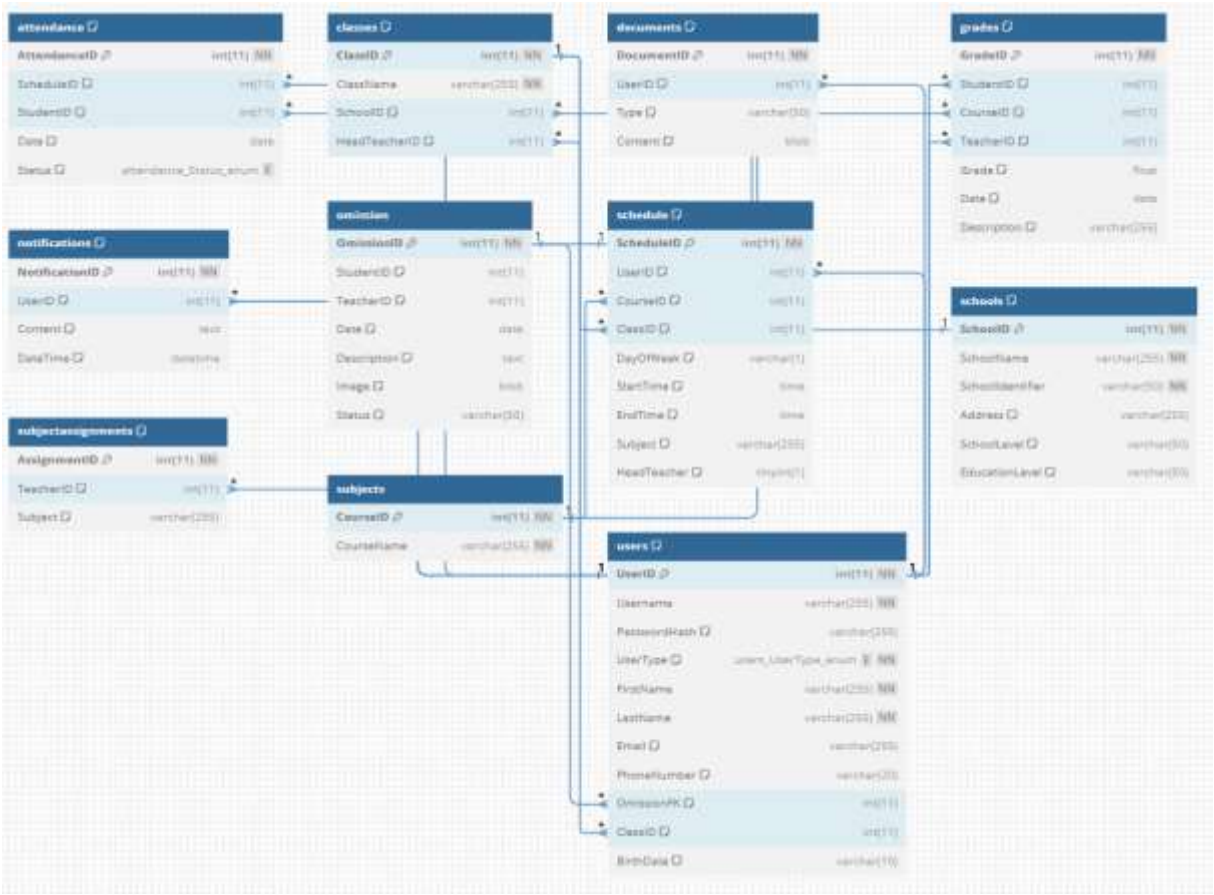
3.7 Rendszergazda Globális Tantárgyfelvétel

Rendszergazdai főoldalra és aloldalaira való belépés privát, hiszen ahhoz csak a tulajdonosnak van engedélye. Innen elérhető azon funkció, amellyel minden Regisztrált Iskola számára látható tantárgyak hozzáadása, és törlése lehetséges. Jelszavas belépést követően a főoldalra, a Rendszergazdai kezelőfelületre irányít minket tovább a rendszer (/systemadmin). A kezelőfelületen a „Global Subjects” feliratú gombra átirányítják a megfelelő aloldalra (/globalsubjects). Itt található egy dinamikus táblázat, amely kiírja az Aktív globális tantárgyakat és azok azonosítóit. A jobb oldalon a megszokott „Actions” oszlopban található a „Delete” gomb, amellyel törölhetünk specifikus tantárgyakat. A táblázat alatt található egy szöveges mező, ide írható a felvenni kívánt tantárgy. Ezután az alatta lévő „Add Subject” feliratú gombra kattintva adható hozzá.

4. Adatbázis

Megvalósítása lokális **MySQL** szerverrel történt, a **XAMPP** keretrendszerrel **MariaDB** és **Apache** szerverrel. A 3. Normálformára törekszik a lehető legkevesebb redundanciával.

4.1 Adatbázis-modell



Bár a teljes adatbázis elkészült, a projekt mérete és a 7. fejezetben említett hiányosságok miatt pár tábla backendes megvalósítása nem tudott megtörténni, a project teljes átláthatóság érdekében leírom és vizualizálom az említett táblákat is.
A használatban lévő táblák a következők:

Users, Subjects, Subjectassignments, Schools, Classes

4.1.1 Kapcsolatok

Attendance tábla:

Kapcsolatok: Kapcsolódik a **Schedule** és **Users** táblákhoz. Az AttendanceID egyedi azonosítója minden jelenléti rekordnak.

Célja: Rögzíti a diákok jelenlétét vagy hiányzását az egyes tanórákon.

Classes tábla:

Kapcsolatok: Kapcsolódik a **Schools** és **Users** táblákhoz. Az ClassID egyedi azonosítója minden osztálynak.

Célja: Tárolja az osztályok adatait, mint például az osztály nevét és osztályfőnökét.

Documents tábla:

Kapcsolatok: Kapcsolódik a **Users** táblához. A DocumentID egyedi azonosítója minden dokumentumnak.

Célja: Felhasználók által feltöltött dokumentumok tárolása, például tanulmányi anyagok vagy tájékoztatók.

Grades tábla:

Kapcsolatok: Kapcsolódik a **Users** és **Subjects** táblákhoz. A GradeID egyedi azonosítója minden jegynek.

Célja: Jegyek rögzítése diákoknak tanárok által kiosztott kurzusokban.

Notifications tábla:

Kapcsolatok: Kapcsolódik a **Users** táblához. A NotificationID egyedi azonosítója minden értesítésnek.

Célja: Felhasználóknak küldött értesítések tárolása, például rendszerüzenetek vagy egyéb kommunikáció.

Omission tábla:

Kapcsolatok: Kapcsolódik a **Users** táblához. Az OmissionID egyedi azonosítója minden mulasztásnak.

Célja: Diákok által rögzített mulasztások nyilvántartása, például, ha valaki hiányzott egy óráról.

Schedule tábla:

Kapcsolatok: Kapcsolódik a **Users**, **Subjects** és **Classes** táblákhoz. A ScheduleID egyedi azonosítója minden órarend elemnek.

Célja: Az órarendek tárolása, például az órák időpontjai, tárgyak és az azokat tartó tanárok.

Schools tábla:

Kapcsolatok: Nincsenek más táblákkal. Speciális tábla, a SchoolID és SchoolIdentifier szolgál kulcsként és felhasználónévként a **Users** táblában a *„schooladmin”* UserType-al rendelkező felhasználóknak.

Célja: Iskolák adatainak tárolása, például az iskola neve, címe és egyéb információk.

Subjectassignments tábla:

Kapcsolatok: Kapcsolódik a **Users** táblához. Az AssignmentID egyedi azonosítója minden *UserType: „teacher”* és tantárgyhoz való hozzárendelésnek.

Célja: Tanárok által tanított tantárgyak nyilvántartása és hozzárendelése a tanárokhoz.

Subjects tábla:

Kapcsolatok: Nincsenek más táblákkal.

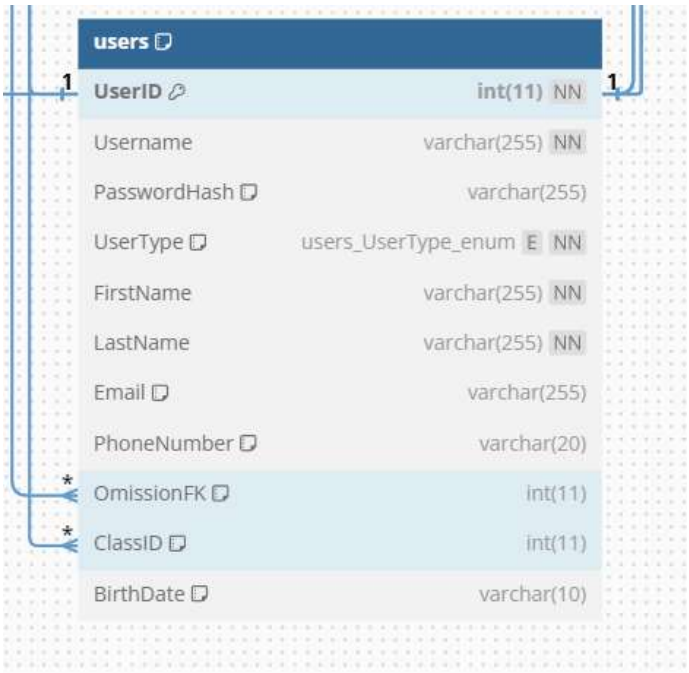
Célja: Tantárgyak tárolása, például a tantárgyak neve és egyéb információk, amelyek minden regisztrált iskola számára elérhetőek, a *„subjectassignment”* funkcionalitásához szükséges tábla.

Users tábla:

Kapcsolatok: Kapcsolódik a(z) **Omission**, **Classes** és **Schedule** táblákhoz. A *UserID* egyedi azonosítója minden felhasználónak a rendszerben.

Célja: Felhasználók adatainak tárolása, mint például a felhasználó típusa (diák, tanár stb.), név, e-mail cím és egyéb információk.

4.2 Users tábla



- **UserID:** int (11) - Egyedi azonosító minden felhasználóhoz a rendszerben.
- **Username:** varchar (255) - A felhasználó felhasználóneve.
- **PasswordHash:** varchar (255) - A felhasználó jelszavának hashelt változata.
- **UserType:** enum('student','teacher','schooladmin','systemadmin') - Meghatározza a felhasználó típusát (diák, tanár, iskolai adminisztrátor vagy rendszergazda).
- **FirstName:** varchar (255) - A felhasználó keresztneve.
- **LastName:** varchar (255) - A felhasználó vezetékneve.
- **Email:** varchar (255) - A felhasználó e-mail címe.
- **PhoneNumber:** varchar (20) - A felhasználó telefonszáma.
- **OmissionFK:** int (11) - Külső kulcs, hivatkozik az omission táblára, ha a felhasználónak van mulasztása.
- **ClassID:** int (11) - Külső kulcs, hivatkozik az osztályok táblára, ha a felhasználó diák vagy tanár.

- **BirthDate:** varchar (10) - A felhasználó születési dátuma.

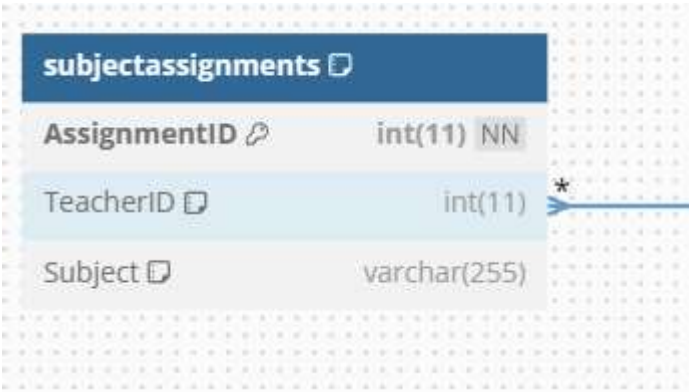
4.3 Schools tábla

- **SchoolID:** int (11) - Egyedi azonosító minden iskolához.
- **SchoolName:** varchar (255) - Az iskola neve.
- **SchoolIdentifier:** varchar (50) - Az iskola azonosítója.
- **Address:** varchar (255) - Az iskola címe.
- **SchoolLevel:** varchar (50) - Az iskola szintje (pl. középiskola).
- **EducationLevel:** varchar (50) - Az oktatás szintje az iskolában. (pl. általános iskola, technikum vagy gimnázium)

4.4 Subjects tábla

- **CourseID:** int (11) - Egyedi azonosító minden tantárgyhoz.
- **CourseName:** varchar (255) - A tantárgy neve.

4.5 Subjectassignment tábla

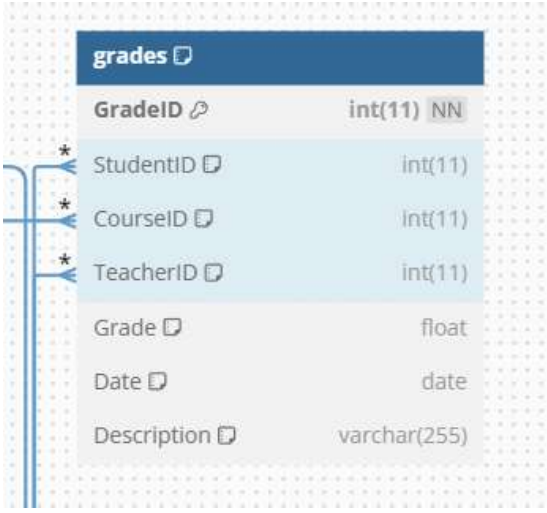


- **AssignmentID:** int (11) - Egyedi azonosító minden tantárgyhoz való hozzárendeléshez.
- **TeacherID**: int (11) - Külső kulcs, hivatkozik a users táblára, azon tanár azonosításához, aki a tantárgyat tanítja.
- **Subject:** varchar (255) - A tantárgy neve.

4.6 Classes tábla

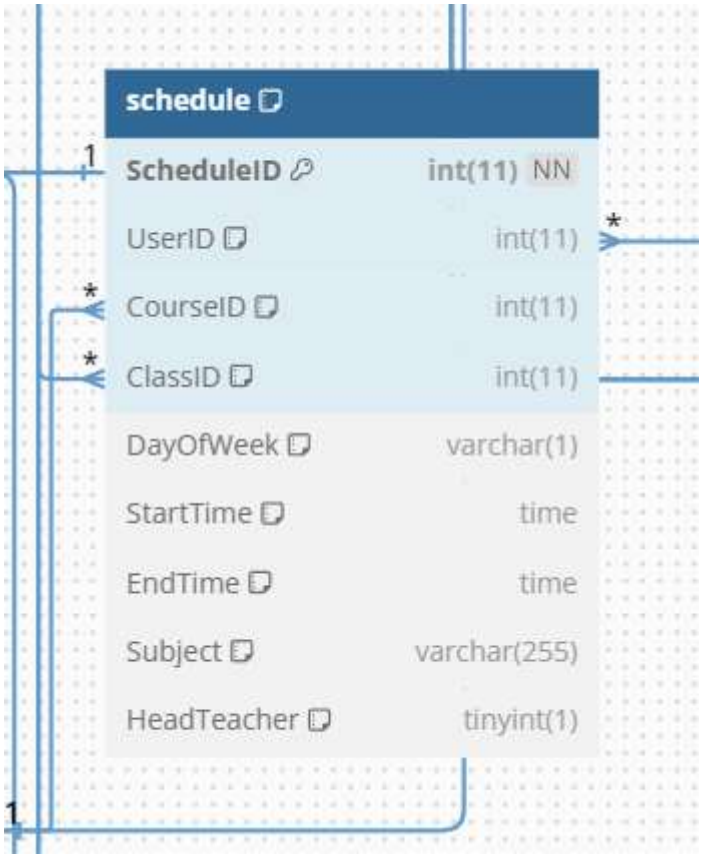
- **ClassID:** int (11) - Egyedi azonosító minden osztályhoz.
- **ClassName:** varchar (255) - Az osztály neve.
- **SchoolID:** int (11) - Külső kulcs, hivatkozik a schools táblára, azon iskola azonosításához, ahol az osztály van.
- **HeadTeacherID:** int (11) - Külső kulcs, hivatkozik a users táblára, azon osztályfőnök azonosításához.

4.7 Grades tábla



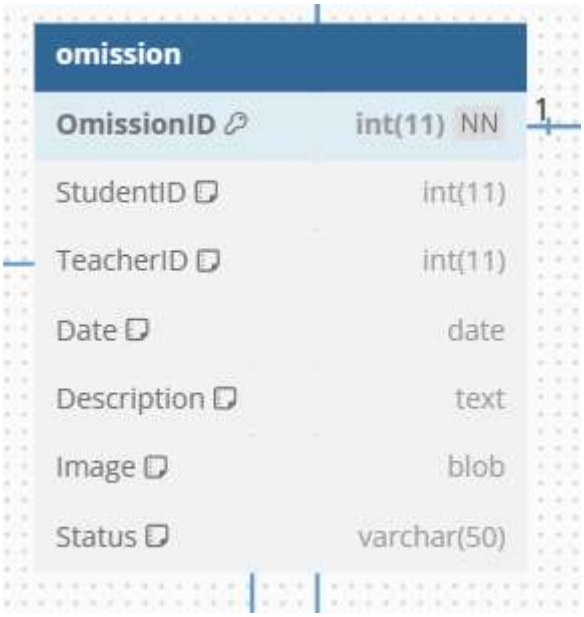
- **GradeID:** int (11) - Egyedi azonosító minden osztályhoz.
- **StudentID:** int (11) - Külső kulcs, hivatkozik a users táblára, azon diák azonosításához.
- **CourseID:** int (11) - Külső kulcs, hivatkozik a subjects táblára, azon kurzus azonosításához, amelyhez a jegy tartozik.
- **TeacherID:** int (11) - Külső kulcs, hivatkozik a users táblára, azon tanár azonosításához, aki a jegyet kiadta.
- **Grade**: float - A jegy értéke.
- **Date:** date - A jegy kiadásának dátuma.
- **Description:** varchar (255) - Jegy leírása.

4.8 Schedule tábla



- **ScheduleID:** int (11) - Egyedi azonosító minden órarend elemhez.
- **UserID:** int (11) - Külső kulcs, hivatkozik a users táblára, azon felhasználó (ez esetben iskolai adminisztrátori fiók) azonosításához, akihez az órarend tartozik.
- **CourseID:** int (11) - Külső kulcs, hivatkozik a subjects táblára, azon kurzus azonosításához, amelyhez az óra tartozik.
- **ClassID:** int (11) - Külső kulcs, hivatkozik a classes táblára, azon osztály azonosításához, amelyiknek az órája van.
- **DayOfWeek:** varchar (1) - Az óra napja a héten.
- **StartTime:** time - Az óra kezdési ideje.
- **EndTime:** time - Az óra befejezési ideje.
- **Subject:** varchar (255) - Az óra tárgya.
- **HeadTeacher:** tinyint (1) - Az óra tanára vagy osztályfőnöke.

4.9 Omission(hiányzások) tábla



- **OmissionID:** int (11) - Egyedi azonosító minden mulasztáshoz.
- **StudentID:** int (11) - Külső kulcs, hivatkozik a users táblára, azon diák azonosításához, aki mulasztott.
- **TeacherID:** int (11) - Külső kulcs, hivatkozik a users táblára, azon tanár azonosításához, aki rögzítette a mulasztást.
- **Date:** date - A mulasztás dátuma.
- **Description:** text - A mulasztás leírása.
- **Image:** blob - A mulasztásról készült kép.
- **Status:** varchar (50) - A mulasztás státusza.

4.10 Documents tábla

- **DocumentID:** int (11) - Egyedi azonosító minden dokumentumhoz.
- **UserID:** int (11) - Külső kulcs, hivatkozik a users táblára, azon felhasználó azonosításához, aki a dokumentumot feltöltötte.
- **Type:** varchar (50) - A dokumentum típusa.
- **Content:** blob - A dokumentum tartalma.

4.11 Attendance tábla

- **AttendanceID:** int (11) - Egyedi azonosító minden jelenléti rekordhoz.
- **ScheduleID:** int (11) - Külső kulcs, hivatkozik a schedule táblára, az órarend azonosításához.
- **StudentID:** int (11) - Külső kulcs, hivatkozik a users táblára, azon diák azonosításához.
- **Date:** date - A jelenlét dátuma.
- **Status:** enum('present','absent') - A jelenlét státusza (jelenlévő vagy hiányzó).

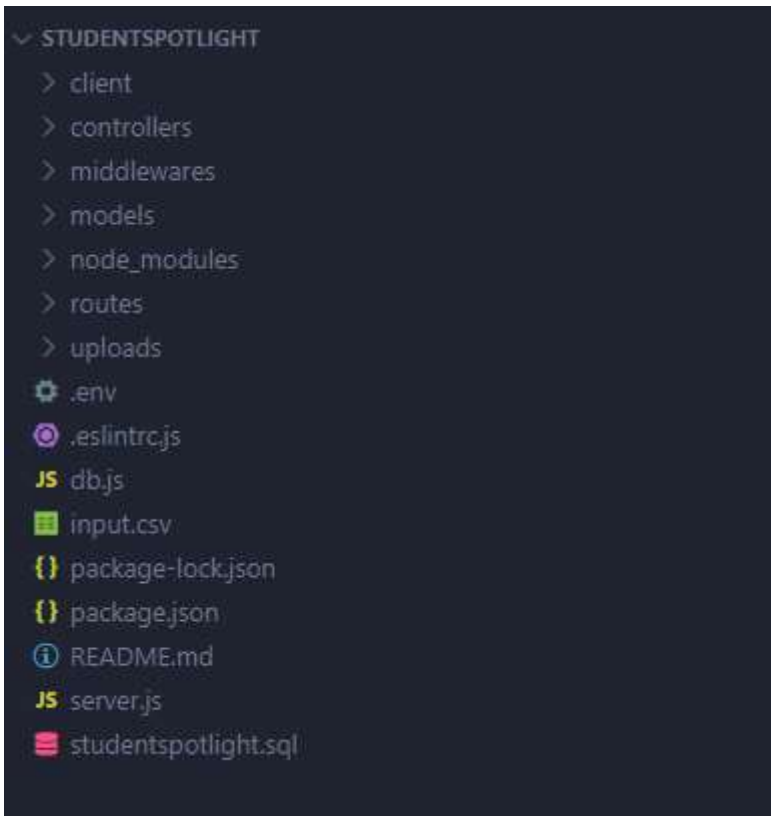
4.12 Notifications tábla

- **NotificationID:** int (11) - Egyedi azonosító minden értesítéshez.
- **UserID:** int (11) - Külső kulcs, hivatkozik a **Users** táblára, azon felhasználó azonosításához, aki az értesítést kapja.
- **Content:** text – Címző és az értesítés tartalma.
- **DateTime:** datetime - Az értesítés időpontja.

5. Forráskód és a Projekt felépítése

A projekt felépítése *RESTful API* szerint történik, a backend és frontend kommunikációja megvalósul.

5.1 Gyökérkönytár

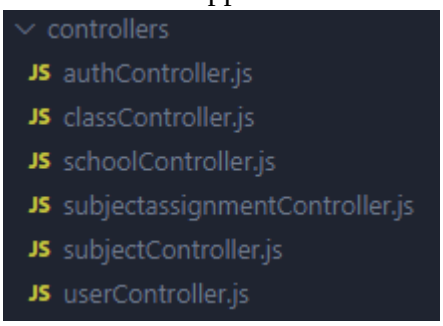


A Gyökérkönyvtárban található meg a React keretrendszerrel létrehozott **Client** mappa, amiben megtalálható a projektünk *Frontend* része.
Emellett a függőségeken kívül megtalálhatóak a *Backend* almappák is, amelyek megfelelnek az MVC architektúrának: **Controllers, Models, Routes, Middlewares**

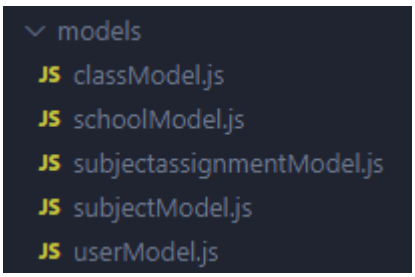
5.2 Backend és Routejai

A backendet a „**server.js**” nevű fájl köti össze. Itt található az összes router meghívva, és innen indítható el az egész weboldal is.
A **Route**s mappában található fájlok az alkalmazás háttérlogikájának előhívására szolgálnak a frontendről. Ezek a specifikus útvonalak kapcsolódnak a megfelelő vezérlőfájlhoz(**controllers**) és annak funkcióihoz. Ezek a funkciók pedig már meglévő **modelleket** használnak fel. Ezek a fájlok objektumorientált alapúak, és a **kontroller** rétegben összpontosítanak az adatok kezelésére és az üzleti logika végrehajtására.
Mivel 5 táblára készültek el a backend funkcionalitások, így ezek a következő fájlokban találhatóak:

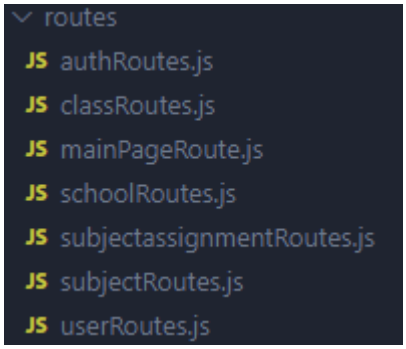
Controllers mappa:



Models mappa:



Routes mappa:



Az Authentikációért az **authController.js** és az **authRoutes.js** felel, egy **middleware** modul segítségével:

```
// authMiddleware.js
require('dotenv').config()
const tokenSecretKey = process.env.JWT_SECRET;
console.log(process.env.JWT_SECRET);
const jwt = require('jsonwebtoken');

const authenticateToken = (req, res, next) => {
    // Extract the token from the Authorization header
    const authHeader = req.headers.authorization;
    const token = authHeader && authHeader.split(' ')[1];

    console.log('Received Token:', token);

    if (!token) {
        return res.sendStatus(401); // Unauthorized
    }

    jwt.verify(token, tokenSecretKey, (err, user) => {
        if (err) {
            console.error('Token Verification Error:', err);
            return res.status(403).json({ error: 'Invalid token' });
        }

        req.user = user;
        console.log('req.user set:', req.user);
        next();
    });
};


module.exports = authenticateToken;
```

5.2.2 Route-ok

authRoutes.js:

```
// routes/authRoutes.js
const express = require('express');
const router = express.Router();
const authController = require('../controllers/authController');
const schoolRoutes = require('./schoolRoutes');
const authenticateToken = require('../middlewares/authMiddleware');
const multer = require('multer');
const upload = multer();

// Register User
router.post('/register', upload.none() , authenticateToken, authController.registerUser);

// Login User (without authentication for frontend fetch)
router.post('/login', authController.loginUser);

// Verify User (with authentication)
router.get('/verify', authenticateToken, authController.verifyUser);

// Protected route for SchoolAdmin
router.get('/schooladmin', authenticateToken, (req, res) => {
  res.json({ message: 'This is a protected route for SchoolAdmin' });
});

router.use('/school', schoolRoutes);

module.exports = router;
```

classRoutes.js:

```
const express = require('express');
const router = express.Router();
const classController = require('../controllers/classController');


// POST endpoint to add a new class
router.post('/addclass', classController.addClass);

// GET endpoint to fetch available teachers
router.get('/availableteachers', classController.getAvailableTeachers);

//GET endpoint to fetch available students
router.get('/availablestudents', classController.getAvailableStudents);

// GET endpoint to fetch classes with associated teachers
router.get('/activeclasses', classController.getClassesWithTeachers);

// GET endpoint to fetch students by school and class
router.get('/students/:classId', classController.getStudentsByClass);

// POST endpoint to add a student to a class
router.post('/:classId/students', classController.addStudentToClass);

// DELETE endpoint to delete selected class
router.delete('/:classId', classController.deleteClass);

// DELETE endpoint to remove selected students from a class
router.delete('/:classId/students', classController.removeSelectedStudents);

module.exports = router;
```

schoolRoutes.js:

```
const express = require('express');
const router = express.Router();
const schoolController = require('../controllers/schoolController');
// const authController = require('../controllers/authController');

// Register School
router.post('/register', schoolController.registerSchool);

// Fetching all schools from db
router.get('/schools', schoolController.getAllSchools);

// Fetching a specific school from db
router.get('/schools/:schoolId', schoolController.getSchoolById);


//Update both at once
router.put('/editSchoolAdmin/:userId', schoolController.editSchoolAdmin);


module.exports = router;
```

subjectRoutes.js:

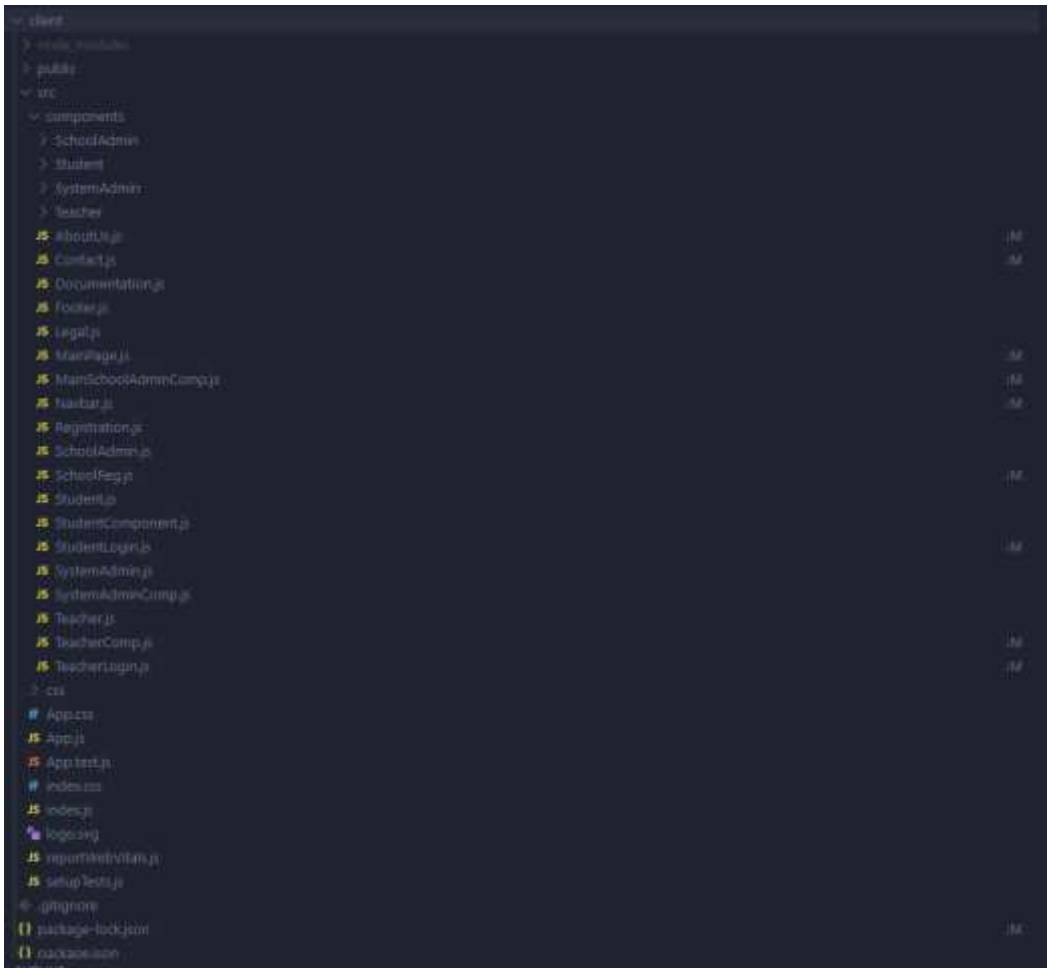
```
const express = require('express');
const router = express.Router();
const subjectController = require('../controllers/subjectController');

// Route to add a new subject
router.post('/add', subjectController.addSubject);

// Route to delete a subject
router.delete('/delete/:courseId', subjectController.deleteSubject);

// Route to get all subjects
router.get('/all', subjectController.getAllSubjects);

module.exports = router;
```

userRoutes.js:

```
// routes/userRoutes.js
const express = require('express');
const router = express.Router();
const userController = require('../controllers/userController');
const authenticateToken = require('../middlewares/authMiddleware');

//GET all members associated with the Logged in school
router.get('/members', authenticateToken, userController.getSchoolMembers);

//GET Loads in user types for a dropbox to be able to filter the table
router.get('/types', authenticateToken, userController.getUserTypes);

//DELETE Delete a Selected User from Users table
router.delete('/delete/:userId', authenticateToken, userController.deleteUser);

//PUT Updates the selected Users data
router.put('/edit/:userId', authenticateToken, userController.editUser);

module.exports = router;
```

5.3 Frontend és Routejai

A *Frontend* JavaScript alapú **React** keretrendszerrel és azon moduljai segítségével készült el, **CSS** és **Bootstrap** alkalmazásával. A tesztelésére **Cypress.JS**-el került sor.
Ez többnyire egymásba ágyazott komponenseket jelent, amelyek formátuma .jsx .
A felugró ablakok, mint például a profiladatok, a **React Modal** nevű moduljaival valósultak meg.



A képen a „**client**” Frontend mappa struktúrája látható, és mappákban találhatóak a további komponensek, amelyek a Jelszavas belépést követően érhetőek el a különböző jogosultságú felhasználóknak.

Az oldal dinamikusságának érdekében használtam még **React Hookokat**, illetve *HTTP* kérések küldésére az **Axios** kliens könyvtárat használtam. Hibakezelésre az **ESLinter** könyvtárát alkalmaztam.

5.3.2 Frontend: Tervezés, Aloldalak, Routeok és Subdomainek

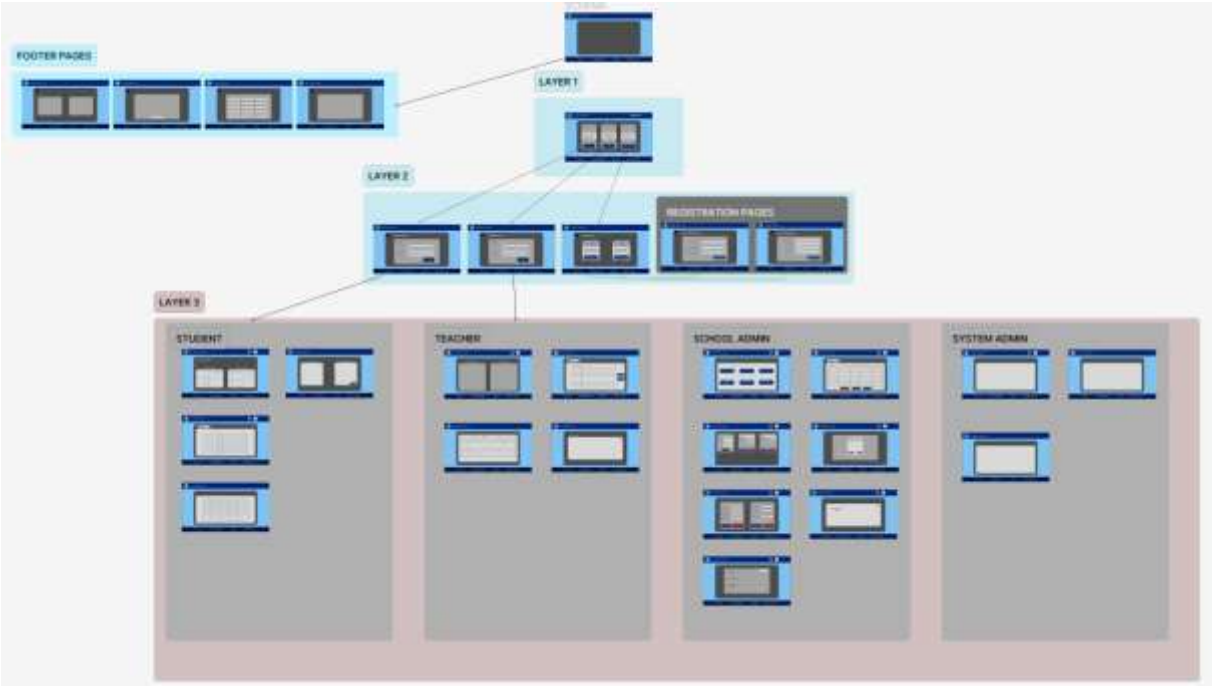
```
const App = () => {
  return (
    <Router>
      <Routes>
        <Route path='/' element={<MainPage />} />
        <Route path="/aboutus" element={<AboutUs/>} />
        <Route path="/documentation" element={<Documentation />} />
        <Route path="/contact" element={<Contact />} />
        <Route path="/legal" element={<Legal />} />
        <Route path="/studentlogin" element={<StudentLogin />} />
        <Route path="/teacherlogin" element={<TeacherLogin />} />
        <Route path="/schoolregistration" element={<SchoolReg />} />
        <Route path="/student" element={<Student />} />
        <Route path="/teacher" element={<Teacher />} />
        <Route path="/schooladmin" element={<SchoolAdmin />} />
        <Route path="/systemadmin" element={<SystemAdmin />} />

        <Route path="/members" element={<Members />} />
        <Route path="/newuser" element={<NewUser />} />
        <Route path="/schoolmeta" element={<SchoolMeta />} />
        <Route path="/schoolschedule" element={<SchoolSchedule />} />
        <Route path="/statistics" element={<Statistics />} />
        <Route path="/subjectassignment" element={<SubjectAssignment />} />

        <Route path="/globalsubjects" element={<GlobalSubjects/>} />

        <Route path="/teacherclasses" element={<TeacherClasses/>}/>
        <Route path="/teacherassignments" element={<TeacherAssignments/>}/>
        <Route path="/teachergrades" element={<TeacherGrades/>}/>
        <Route path="/teacherprofile" element={<TeacherProfile/>}/>

        <Route path="/studentfullscheduele" element={<StudentFullScheduele/>}/>
        <Route path="/studentallgrades" element={<StudentAllGrades/>}/>
        <Route path="/studentprofile" element={<StudentProfile/>}/>


      </Routes>
    </Router>
  );
};

export default App;
```

A képen megtekinthetően az összes Frontendben található Route/Subdomain amelyek elérhetőek a weboldalon.

A tervezést figmában készítettem el, az alábbi linken található:

[Student Spotlight – Figma]

Képernyőkép a tervekről:



Az oldalt 3 részre bontottuk.

Első réteget képezi a főoldal az onnan elérhető statikus Navigációs sávval és lábléccel.

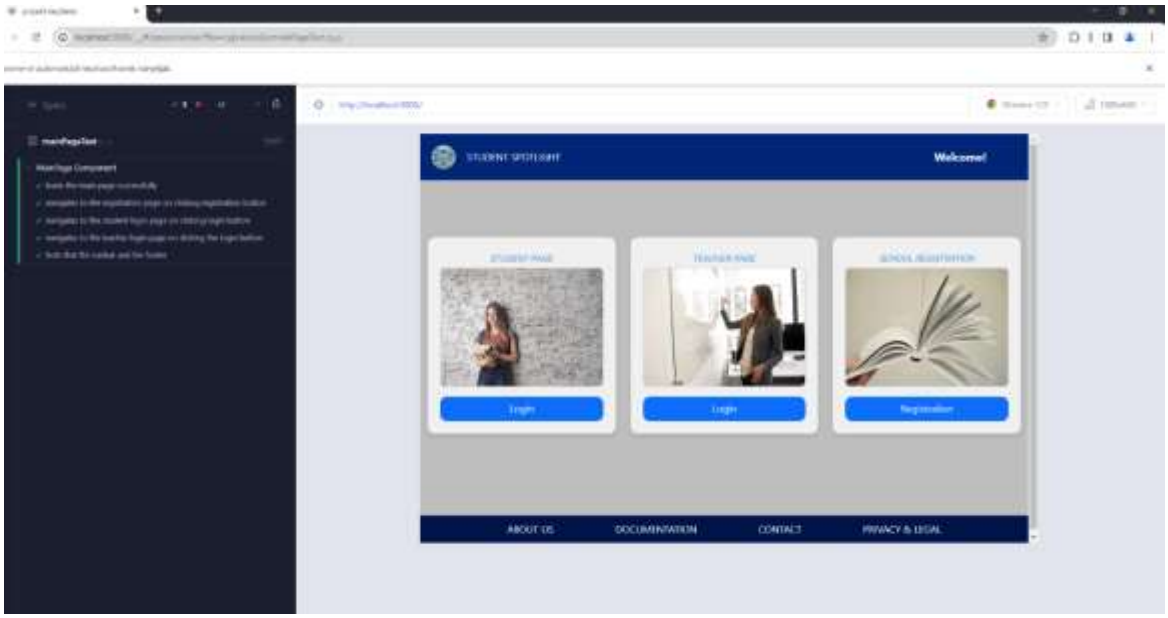
Ezt követi a második réteg, ahol a belépések és regisztrációk valósulnak meg. A kezelhetőség érdekében ezt leegyszerűsítettem, így nem az eredeti tervek alapján készült el a regisztráció.

Harmadik réteget képezik az összes olyan weboldal, amely a Jelszavas bejelentkezést követően érhető el. A Felhasználói fiókoknak megfelelően 4 további részre bontva.

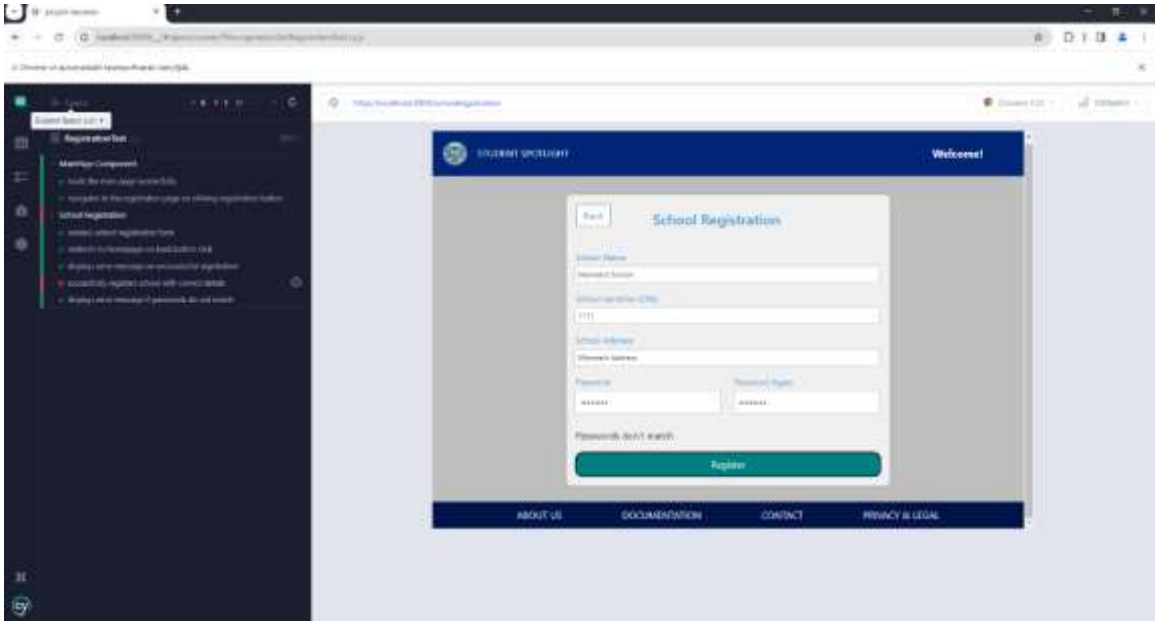
6. Tesztelés

A tesztelés a Cypress.js keretrendszer segítségével történt. Ez biztosította a Frontend és Backend tesztelését.

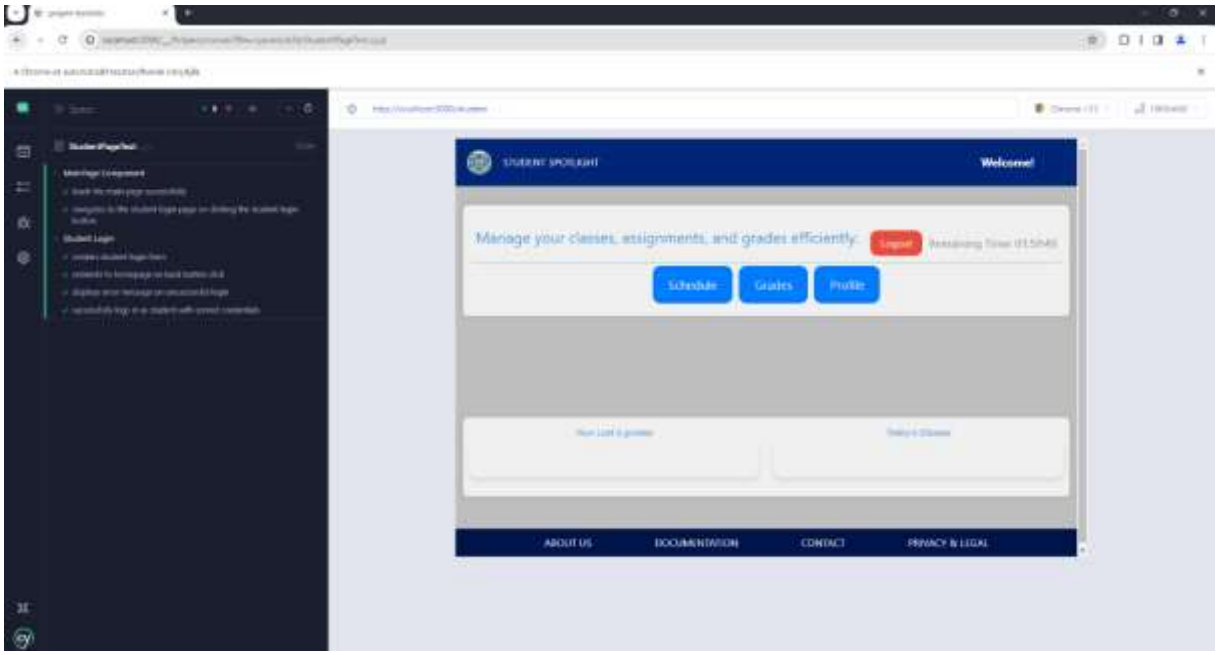
Főoldal:



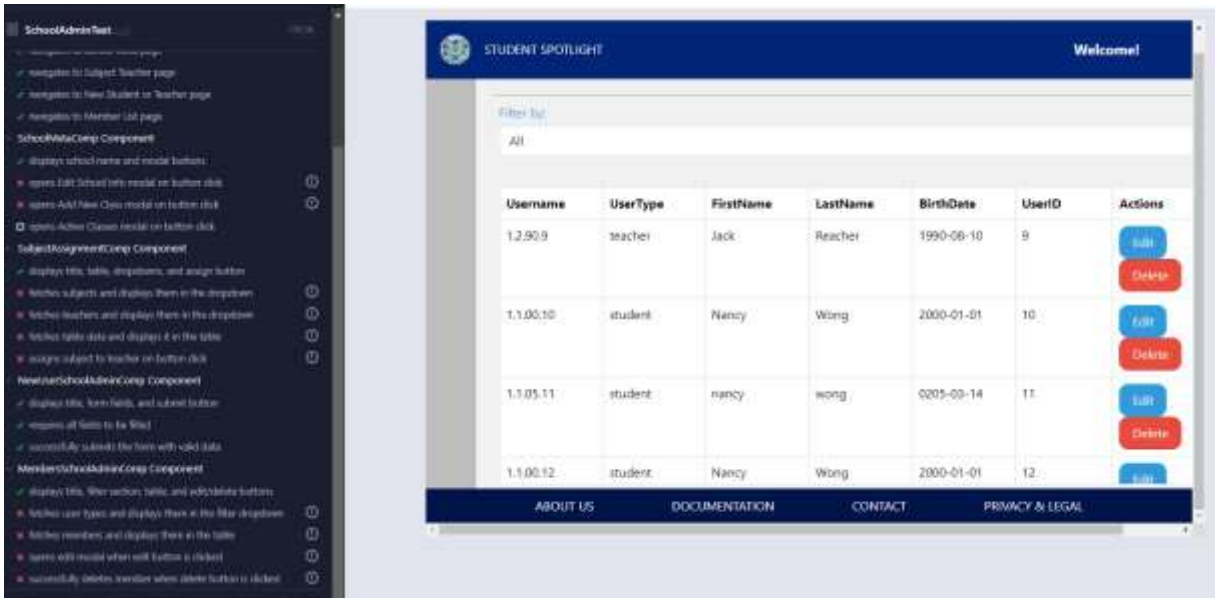
Iskola regisztrációja:



Tanulói aloldal:

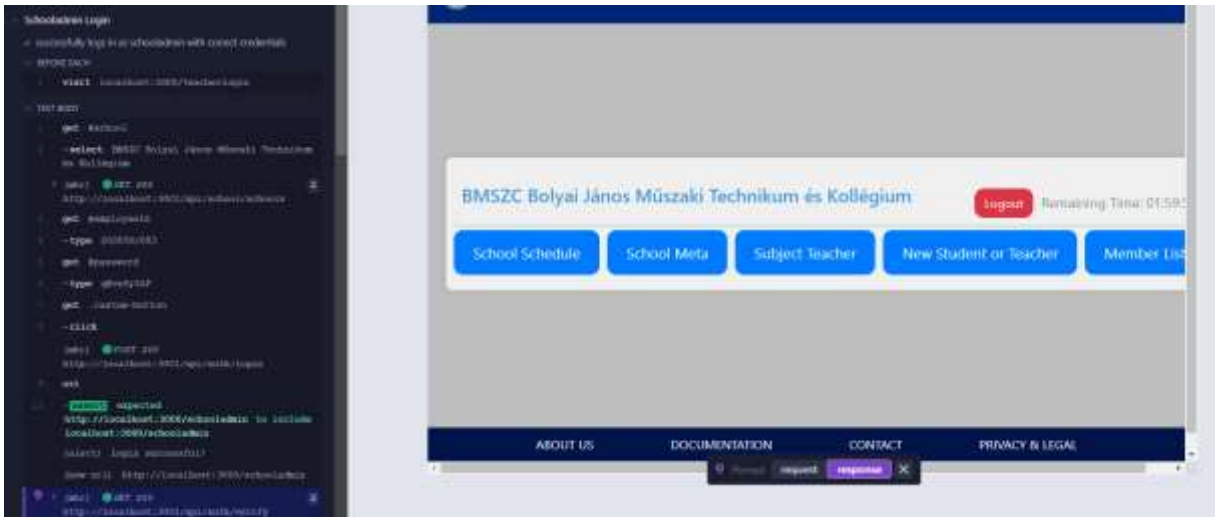


Backend tesztelések:

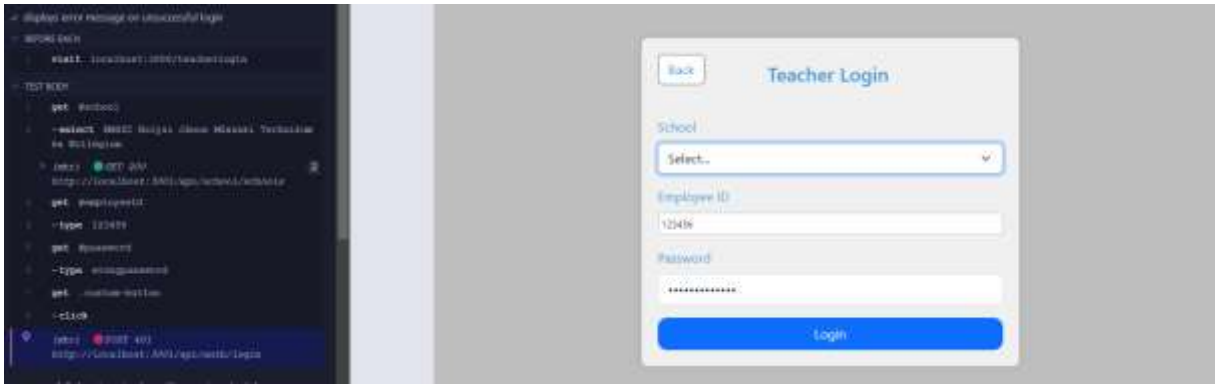


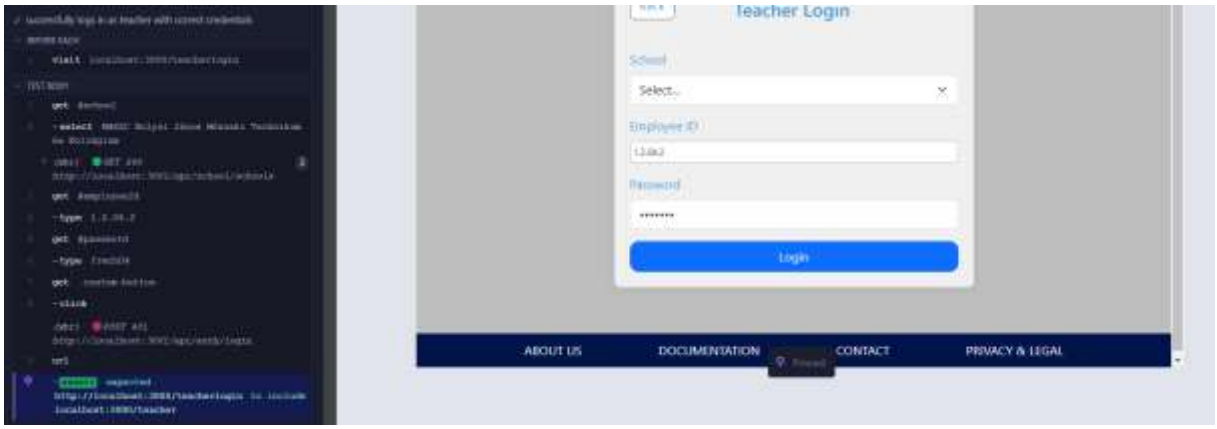
- Megjegyzés: Bár a tesztek itt hibát mutatnak valamiért, de a felsorolt funkcionalitások működőképesek, tesztelve lettek Postmannel is és az oldalon belül se volt probléma használatukkor.

Iskolai Adminisztrátori kezelőfelület:



Teacher Login page:





- Megjegyzés: Innen lehetséges a belépés a Tanári, Iskolai Adminisztrátor és rendszergazda fiókokba.

- A Tanári felhasználói fiókok speciális, Automatán generált Employee ID-val rendelkeznek, amelyet az iskolától kaphatnak meg. Ugyanez érvényesül a Tanulókra is.

7. Csapatmunka, Kontribúció és azok hibái

Az eredeti tervek szerint ketten dolgoztunk a projekten. Frech Szabolcs a frontenden és teszteléseken, amíg én, Forrás Márk a project lead szerepben a backenden és adatbázison. Sajnálatos módon a kommunikáció hiánya és annak minősége nem érte el a projekt folytonos aktív fejlesztéséhez szükséges szintet, így a csapatmunka nem valósult meg és rákényszerültem a projekt egyes részeinek átvételére, beleértve a teljes tervezést és a frontend 80%-át. A projekt tervezett eredeti mérete túl nagy lett volna, hogy időre elkészüljön egyedül általam és csapattársam túl későn csatlakozott a munkához, így konzulens utasítására megfeleztem a projektet.

A csapatmunka nem valósult meg teljes formájában a már fent említett kommunikációs és érdekeltségi problémák miatt, ezért szeretném tisztázni a Kontribúcióink mértékét:

Tervezés (figma) – Forrás Márk
Média fájlok, logok, statikus lábléc aloldalak – Frech Szabolcs
Projekt alapjainak felépítése – Forrás Márk
Adatbázis megtervezése és felépítése – Forrás Márk
Frontend és **Backend** oldalon az *1. rétegbeli publikus* aloldalak elkészítése – Forrás Márk
Frontend és **Backend** oldalon a *2. rétegbeli publikus* aloldalak elkészítése – Forrás Márk
Frontend és **Backend** oldalon a *3. rétegbeli privát* , **Iskola Adminisztrátori** és **Rendszergazdai** aloldalak elkészítése(Az elkészült backend funkcionalitások nagy része ezeken az oldalakon található) – Forrás Márk
Frontend oldalon a *3. rétegbeli privát*, **Tanulói** és **Tanári** aloldalak – Frech Szabolcs
Tesztelések mind **Frontend** és **Backend** oldalon – Frech Szabolcs
Dokumentáció – Forrás Márk
PPT Prezentáció – közösen.

- *Megjegyzés: A rétegeket az 5.3.2-es fejezetnél találja meg.*

8. Összegzés

Összegezve a fejlesztéssel töltött elmúlt hónapokat a választott- és jóváhagyott projektünk érdekesnek bizonyult és feszegette készségeink határát, illetve új lehetőségeket tárt fel tudásunk bővítésének és egyéni fejlődésünk terén. Rámutatott, hogy mennyi mindenben szükséges még fejlődnünk, hogy piacképesek legyünk a szoftverfejlesztés- és tesztelés területén.

8.1 Fejlesztési lehetőségek

Ahogy már az Adatbázis résznél említettem, mivel projektünk megfelezése szükségessé vált, így félkész. A közeljövőben tervezem befejezni a projektet, hogy a teljes, eltervezett funkcionalitása megvalósulhasson:

1. Órarend rendszer, amelyet a Regisztrált Iskolák képesek maguk kezelni, és amelyet az ott Tanuló diákok és Tanárok is elérnek.
2. Az órarend rendszerre épülő jelenléti, és hiányzás rendszer, amelyet a Tanárok képesek kezelni.
3. A diák profilok képesek legyenek hiányzásaik megtekintésére és igazolására.
4. Illetve az erre épülő értesítési rendszer a diák profilok felé.
5. Osztályzati rendszer az órarend alapján.
6. Statisztikai aloldalak létrehozása minden felhasználói szinten jogosultságnak megfelelően.

8.2 Forrásmegjelölés és elérési hivatkozások

Github: https://github.com/gl4s/StudentSpotlight

Figma: Student Spotlight – Figma

Fontawesome: Font Awesome

ColorsUI: Colors UI

Adobe Stock: https://stock.adobe.com/

React Documentation: React Reference Overview – React

Cypress.js Documentation: Cypress Documentation

Node.js Documentation: Index | Node.js v21.7.2 Documentation (nodejs.org)

Bootstrap: Bootstrap · The most popular HTML, CSS, and JS library in the world.

ParseJWT function:

How to decode jwt token in javascript without using a library? - Stack Overflow

Stack Overflow - Where Developers Learn, Share, & Build Careers