Math Solve

# Bevezetés

## Projekt bemutatása

|  |  |
| --- | --- |
| Mi a projekt célja? | Ez a webalkalmazás lehetőséget biztosít a felhasználóknak, hogy fejlesszék matematikai készségeiket gyakorlófeladatok és versenyek segítségével. Célja, hogy interaktív és motiváló környezetet teremtsen a tanulás és fejlődés érdekében. |
| Miért készült a projekt? | Az oldal oktatási és gyakorlási célokra egyaránt használható, emellett lehetőséget biztosít a versenyzésre és a közösségi élmény megélésére is. Az alkalmazás nyitott minden matematikát kedvelő felhasználó számára, és folyamatos fejlesztés alatt áll, hogy minél több igényt kielégíthessen. |
| Kik a célfelhasználók? | Az alkalmazás bárki számára elérhető, aki szeretné fejleszteni matematikai készségeit, legyen szó diákokról, tanárokról, versenyzőkről vagy egyszerűen csak a matematika iránt érdeklődő felhasználókról. |

## Főbb funkciók összefoglalása

* **Felhasználói fiók kezelése** – Regisztráció, bejelentkezés és profilkezelés
* **Matematikai feladatok megoldása és létrehozása** – Feladatok gyakorlása, saját feladatok feltöltése
* **Fejlődési statisztikák** – Teljesítmény és előrehaladás nyomon követése
* **Versenyeken való részvétel** – Kihívások és matematikai versenyek megoldása
* **Közösségi interakciók – Kapcsolódás más felhasználókhoz, tapasztalatok megosztása, segítségnyújtás és támogatás kérése**

## Pár használt technológia

* **Frontend:**
  + **Vue 3** – A projekt alapja, a felhasználói felület építéséhez használt modern JavaScript framework.
  + **Vuetify** – A felhasználói felület gyors fejlesztését segítő material design komponens könyvtár**.**
  + **Vitest** –A tesztelési keretrendszer, amely biztosítja az alkalmazás egységtesztelését.
  + **Concurrently** –A frontend és backend egyszerre történő futtatásához szükséges eszköz.
  + **Pinia** –Állapotkezelésre szolgáló könyvtár a Vue 3-ban, amely biztosítja a globális állapot kezelését az alkalmazásban.
* **API:**
  + **Axios** –HTTP kérés küldésére és válaszok kezelésére használt könyvtár.
  + **TanStack Query** – Aszinkron adatlekérések és szerver oldali állapotkezelés kezelésére szolgáló könyvtár.
* **Backend:**
  + **Express** – A backend szerver keretrendszer, amely az alkalmazás logikáját és API-jait kezeli.
  + **Sequelize** – ORM (Object-Relational Mapping) könyvtár, amely lehetővé teszi az adatbázis kezelést JavaScript-ben.
  + **Dotenv** – Környezeti változók kezelésére használt könyvtár.
  + **Swagger** – API dokumentáció generálására és tesztelésére használt eszköz.
  + **Multer** – A fájlok feltöltéséhez használt middleware.
* **Database:**
  + **Mysql** – Adatok tárolására való környezet.

### Csapattagok szerepkörei és hozzájárulásai

A projekt megvalósítása során fontos szempont volt számunkra, hogy minden csapattag teljes körű tapasztalatot szerezzen a fejlesztési folyamat minden területén. Ezért úgy döntöttünk, hogy nem szigorúan backend és frontend szerepkörökre osztjuk a munkát, hanem inkább oldalak és komponensek szerint haladunk. Ez a módszer lehetővé tette, hogy mindenki dolgozzon az adatbáziskezelésen (MySQL), a backend fejlesztésén (Express.js, Sequelize ORM), az API integráción (Axios, TanStack Query), valamint a frontend megvalósításán (Vue.js).

Ez a megközelítés segítette a csapattagokat abban, hogy átfogóbb képet kapjanak a teljes rendszer működéséről, és megkönnyítette az egyes funkciók összekapcsolását. A munkamegosztás során mindenki egy-egy oldalt vagy funkcionális egységet kapott, amelyért teljes mértékben felelősséget vállalt. Ezzel biztosítottuk, hogy mindenki aktívan hozzájáruljon a projekt sikeréhez, miközben fejlesztette a különböző technológiákhoz kapcsolódó készségeit.

#### Az alábbiakban összefoglaljuk, hogy ki melyik oldalon vagy komponensen dolgozott:

* Bujdosó Bálint László
  + Fő oldal | Feladat létrehozása | Feladat ellenőrzése | Profil oldal | Áruház oldal
* Mózes Bence
  + Verseny feladatok oldal | Verseny oldal | Rólunk oldal
* Pintea Roland Dániel
  + Közösségi oldal | Bejelentkezés / Regisztráció oldal | Felhasználó beállítások | Admin panel

# Telepítési Útmutató

A projekt telepítése és futtatása a következő lépésekben történik:

1. Előfeltételek
   * Telepítsd a [Node.js-t](https://nodejs.org/), ha még nincs telepítve.
   * Telepítsd a [Git-t](https://git-scm.com/), ha még nincs telepítve.
   * Telepítsd a [XAMP](https://www.apachefriends.org), ha még nincs telepítve.
2. A projekt klónozása
   * Klónozd a projektet a következő parancs futtatásával: git clone <https://github.com/MozesBence/matekos_leetcode.git>
3. Csomagok telepítése
   * Navigálj a projekt mappájába és telepítsd a szükséges csomagokat:
     + cd root (ahova telepítetted a projektet)
     + npm install
     + cd src
     + npm install
4. Projekt futtatása

A projektet a következő parancs futtatásával indíthatod el:

* + XXAMP elindítása
    - MySQL elindítása
    - Apache elindítása
  + Fejlesztői környezet:
    - npm run dev
  + Felhasználó környezet (build készítés):
    - npm run build
    - cd dist
    - npm run preview (frontend elindítása)
    - cd src
    - npm start (backend elindítása)

## API dokumentáció (Swagger)

A projekt **backend** API-jának dokumentációját a **Swagger** segítségével generáltuk, amely biztosítja a felhasználók és fejlesztők számára az API végpontjainak könnyű elérését és használatát.

### Mi a Swagger?

A **Swagger** (más néven OpenAPI) egy eszközkészlet, amely lehetővé teszi az API dokumentáció automatikus generálását, tesztelését és megértését. A Swagger UI egy vizuális felületet biztosít az API végpontjainak böngészésére, ahol a felhasználók megtekinthetik az összes elérhető végpontot, azok paramétereit, válaszait, és kipróbálhatják a különböző HTTP kéréseket.

### Hogyan érheted el a Swagger dokumentációt?

A Swagger API dokumentáció az alábbi linken érhető el a projekt **backend** részén:

**A helyi fejlesztési környezetben elérhető a Swagger UI az alábbi URL-en, miután a backend szerver fut:**

[**Swagger API Dokumentáció**](http://localhost:3000/api-docs)

### Használat:

1. **API végpontok böngészése** – Az összes elérhető API végpont megtekintése.
2. **API kérések kipróbálása** – A Swagger UI lehetővé teszi a különböző API kérések tesztelését, például GET, POST, PUT, DELETE kéréseket. A felhasználók egyszerűen beírhatják a kívánt paramétereket és végrehajthatják a kérést közvetlenül az UI-ból.
3. **Példák és válaszok** – Minden végponthoz tartozik egy példakérés és a választ is bemutatja a Swagger, így könnyen megérthető, hogy miként működik az adott API végpont.

## Tesztelés

A projekt tesztelése a backend alkalmazásban a [Jest](https://jestjs.io/) és [Supertest](https://www.npmjs.com/package/supertest) segítségével történik, amelyek lehetővé teszik a unit tesztelést és az API végpontok integrált tesztelését.

### Tesztelési keretrendszer

* **Jest**: A Jest egy JavaScript tesztelési keretrendszer, amely a tesztelési logikát és a tesztek futtatását kezeli. Lehetővé teszi az aszinkron műveletek kezelését, és részletes tesztelési eredményeket biztosít.
* **Supertest**: A Supertest egy HTTP aszinkron tesztelő eszköz, amely lehetővé teszi az API végpontok tesztelését. A Supertest segítségével szimulálhatunk HTTP kéréseket és ellenőrizhetjük a válaszokat.

### Tesztelési parancsok

* **A tesztek futtatása**:
  + cd backend\_test
  + npm install
  + npm run test

### Tesztelési struktúra

A tesztek a backend projektben a *tests* mappában találhatók. Az egyes tesztelési fájlok általában az adott funkciók vagy API végpontok tesztelésére vannak kialakítva.

* **Unit tesztek**: A backend logika egyes részleteit, mint például az adatbázis műveletek vagy egyes segéd funkciók, unit tesztekkel ellenőrizzük.
* **API tesztek**: Az API végpontokat a Supertest segítségével teszteljük. Az API tesztek szimulálják a HTTP kéréseket és ellenőrzik a válaszokat, például a státuszkódokat, a válaszadatokat és a hibakezelést.
* **Adatbázis**: Biztosítani kell hogy az adatbázis MySQL-ben létre van hozva és eléri.

### Tesztelési hibák kezelése

Amennyiben tesztelés közben hibák merülnek fel, ellenőrizni kell:

* **Az adatbázis kapcsolódik-e megfelelően**: Biztosítani kell, hogy a tesztelés előtt az adatbázis csatlakozik és inicializálódik.
* **Tesztadatok létezése**: Tesztelés előtt érdemes biztosítani, hogy az adatbázis tesztadatokkal rendelkezzen.

Ha a tesztek során egy-egy hiba felmerül, azokat a konzolban részletes üzenetek formájában ellenőrizhetjük, ami segít a problémák gyors azonosításában.

## Adatbázis

A projekt során az adatok hatékony és biztonságos tárolása érdekében a MySQL adatbázis-kezelőt választottuk. A MySQL egy megbízható, skálázható és széles körben elterjedt relációs adatbázis-kezelő rendszer, amely lehetővé teszi az adatok strukturált és optimalizált tárolását.

Az adatbázis kialakításakor törekedtünk a megfelelő normalizálásra, hogy minimalizáljuk az adatredundanciát és biztosítsuk a gyors lekérdezéseket. Az adatbázis szerkezetét az alkalmazás igényeihez igazítottuk, beleértve a felhasználói adatok, jogosultságok és egyéb fontos entitások kezelését. Az adatbázis műveleteket a Sequelize ORM segítségével valósítottuk meg, amely megkönnyíti a MySQL adatbázissal való kommunikációt, biztosítja a tranzakciókezelést, valamint elősegíti a biztonságos és hatékony adatkezelést.

