**QuickTick**

**Fejlesztői és felhasználói dokumentáció**

**Bevezető**

**A QuickTick** **név eredete**

A **QuickTick** név megalkotása során az volt a célunk, hogy az alkalmazás lényegét egy frappáns, könnyen megjegyezhető névben sűrítsük össze. Mivel a QuickTick egy feladatkezelő alkalmazás, amely lehetővé teszi a felhasználók számára, hogy gyorsan és egyszerűen rendszerezzék, szerkesszék vagy töröljék teendőiket, olyan nevet kerestünk, ami a gyorsaságot, egyszerűséget és a feladatok kipipálását egyaránt kifejezi.

A név két angol szóból áll:

* **„Quick”** jelentése: gyors. Ez a rész a gyorsaságra, hatékonyságra utal, hiszen a célunk az, hogy a felhasználók minimális idő alatt maximális eredményt érjenek el a napi teendőik kezelése során.
* **„Tick”** jelentése: pipa vagy pipálás. Ez a szó a kipipált feladat szimbólumára utal, de finoman emlékeztet az idő múlására, és a határidők fontosságára is.

A **QuickTick** tehát egy olyan név, amely egyszerre tükrözi az alkalmazás funkcionalitását és célját: **gyorsan és hatékonyan elvégezni, majd kipipálni a teendőket**.

**Fejlesztői dokumentáció**

**A fejlesztés során használt programok**

* Visual Studio Code
* Google Chrome
* XAMPP
* GitHub
* Microsoft Word
* Microsoft PowerPoint

Visual Studio Code: A fejlesztéshez használt környezet.

Google Chrome: Weboldal megtekintéséhez használt szoftver.

XAMPP: Adatbázis, illetve weboldal futtatása.

GitHub: A verzió követéshez használt szoftver.

Microsoft Word: A dokumentáció elkészítéséhez használt program.

Microsoft PowerPoint: A prezentáció, illetve bemutatáshoz használt program.

**A program célja**

A **QuickTick** egy feladatkezelő alkalmazás, amely segít a felhasználóknak abban, hogy rendszerezetten és hatékonyan kezeljék a mindennapi teendőiket. A célunk egy olyan eszköz létrehozása volt, amely egyszerűen használható, mégis elegendő rugalmasságot nyújt a különböző típusú feladatok kezeléséhez – legyen szó egy bevásárlólistáról, egy iskolai projekt lépéseiről, vagy akár egy teljes napi munkatervről.

Az alkalmazás lehetővé teszi a feladatok gyors hozzáadását, szerkesztését, rendezését és törlését. A felhasználók testre szabhatják, priorizálhatják a teendőket. Mindezt egy letisztult, átlátható felületen, amelyet úgy terveztünk, hogy segítse a fókuszált munkavégzést.

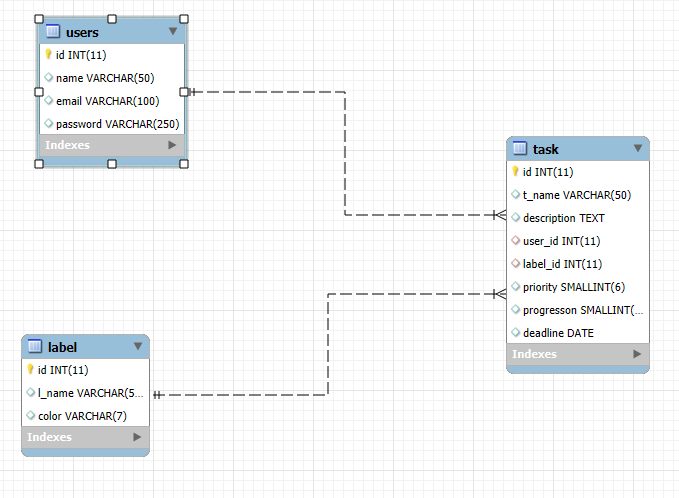
**Rendszerkövetelmény**

|  |  |  |
| --- | --- | --- |
| Rendszerkövetelmény | Minimum | Ajánlott |
| Processzor | Pentium(R) Dual-Core CPUE5700 @ 3.00GHz | Intel Core i3-12100 |
| Memória (RAM) | 4GB | 8GB |
| Merevlemez | HDD vagy SSD | HDD vagy SSD |
| Kijelző | 1080p (1920x1080) | 1080p (1920x1080) |
| Beviteli eszköz | Billentyűzet és egér | Billentyűzet és egér |
| Operációs rendszer | Windows 10 64bit | Windows 10 64bit |

**Adatmodell**

A megfelelő működéshez egy adatbázisra van szükség. A program egy MySQL adatbázissal rendelkezik, a XAMPP és a PHPMyAdmin használatával működik.

MySQL Workbench kép:



Az adatbázis 3 külön táblából áll:

1. users tábla: A felhasználó adatait tartalmazza
   * id: Elsődleges kulcsa a táblának, a felhasználó egyedi azonosítója.  
     Adattípus: INT (Auto Increment) automatikusan növekvő
   * name: A felhasználó nevét tartalmazza.  
     Adattípus: maximumim 50 karakterből álló szöveg VARCHAR (50)
   * email: A felhasználó e-mail címet tartalmazza.  
     Adattípus: maximum 100 karakterből álló szöveg VARCHAR (100)
   * password: A felhasználó jelszavát tartalmazza, amely belépéshez szükséges.  
     Adattípus: maximum 250 karakterből álló szöveg VARCHAR (250)
2. task tábla: A feladatok adatait tartalmazza
   * id:Eésődleges kulcsa a táblának, a feladat egyedi azonosítója.  
     Adattípus: INT (Auto Increment) automatikusan növekvő
   * t\_name: A feladat nevét tartalmazza.  
     Adattípus: maximumim 50 karakterből álló szöveg VARCHAR (50)
   * description: A feladat leírása. Adattípus: TEXT
   * user\_id: Idegen kulcsa a táblának, a felhasználó egyedi azonosítója. users tábla id kulcsára hivatkozik. Adattípus: INT (11)
   * label\_id: Idegen kulcsa a táblának, a felhasználó egyedi azonosítója. label tábla id kulcsára hivatkozik. Adattípus: INT (11)
   * priority: A feladat fontossága. Adattípus: SMALLINT (6)
   * progression: A feladat státusza, felvéve, elkészült, lejárt állapotot tartalmazza.  
     Adattípus: SMALLINT (6)
   * deadline: A feladat határideje. Adattípus: DATE
3. label tábla:
   * id: Eésődleges kulcsa a táblának, a címke egyedi azonosítója.  
     Adattípus: INT (Auto Increment) automatikusan növekvő
   * l\_name: A címke nevét tartalmazza.  
     Adattípus: maximum 50 karakterből álló szöveg VARCHAR (50)
   * color: A címke színét tartalmazza hexadecimális kódként  
     Adattípus: maximum 7 karakterből álló szöveg VARCHAR (7)

**Az elkészítéshez használt számítógépek specifikációja**

Az iskolában lévő két külön számítógép, illetve egy személyes laptop amely előre összeállított konfigurációval rendelkezik (Asus TUF Gaming A15, Modellje: FA506II) amely a vizsgán a projekt bemutatásához is szükséges, ezen kívül két személyes asztali számítógép egyedi konfigurációval, volt használva a fejlesztéshez.

Első iskolai számítógép specifikációja:

|  |  |
| --- | --- |
| Processzor | Pentium(R) Dual-Core CPUE5700 @ 3.00GHz |
| Memória (RAM) | 8GB |
| Merevlemez | HDD |
| Kijelző | 1080p (1920x1080) |
| Beviteli eszköz | Billentyűzet és egér |
| Operációs rendszer | Windows 10 64bit |

Második iskolai számítógép specifikációja:

|  |  |
| --- | --- |
| Processzor | Intel(R) Core(TM) i5-14500 |
| Memória (RAM) | 32GB |
| Merevlemez | SSD |
| Kijelző | 1080p (1920x1080) |
| Beviteli eszköz | Billentyűzet és egér |
| Operációs rendszer | Windows 10 64bit |

Laptop specifikációja:

|  |  |
| --- | --- |
| Processzor | AMD Ryzen™ 5 4600H |
| Videókártya | NVIDIA GeForce GTX 1650 TI |
| Memória (RAM) | 8GB |
| Merevlemez | SSD |
| Kijelző | 1080p (1920x1080) |
| Beviteli eszköz | Billentyűzet és egér |
| Operációs rendszer | Windows 11 64bit |

Első otthoni számítógép specifikációja:

|  |  |
| --- | --- |
| Processzor | AMD Ryzen™ 7 7800X3D |
| Videókártya | NVIDIA GeForce RTX 2060 |
| Memória (RAM) | 32GB |
| Merevlemez | SSD |
| Első kijelző | 24’ 1080p (1920x1080) |
| Második kijelző | 27’ 1440p (2560x1440) |
| Beviteli eszköz | Billentyűzet és egér |
| Operációs rendszer | Windows 11 64bit |

Második otthoni számítógép specifikációja:

|  |  |
| --- | --- |
| Processzor | Intel(R) Core(TM) i7-8700 |
| Videókártya | NVIDIA GeForce RTX 3060 |
| Memória (RAM) | 32GB |
| Merevlemez | SSD |
| Első kijelző | 24’ 1080p (1920x1080) |
| Második kijelző | 49’ 1440p(5120x1440) |
| Beviteli eszköz | Billentyűzet és egér |
| Operációs rendszer | Windows 10 64bit |

Program főbb részleteinek áttekintése

A következő részben a program által használt általunk írt API-ok bemutatása következik röviden milyen bemenő paramétereket várnak milyen lehetséges hibaüzenetek jöhetnek vissza, illetve ha minden rendben milyen adatok érkeznek vissza

register.php

1. Az Api ellenőrzi a metódust, illetve átveszi a bemenő paramétereket

2. Ellenőrzi jött-e üres adat, ha igen hibát dob

3. titkosítja a jelszót

4. Ellenőrzi létezik-e már az email cím

5. beszúrja az új felhasználót

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Body | Válasz |
| POST | ./Assets/php/register.php | JSON | JSON |

Kéréstörzs (Request Body) JSON:

{

username:”Ex Pista”

email:”[ex@gmail.com](mailto:ex@gmail.com)”

password:”123”

}

Lehetséges Hibaüzenetek:

{

//Üres mező esetén

'error' => 'Minden mező kitöltése szükséges']);

//Már használatban lévő email esetén

'error' => 'Email már használatban van'

}

Sikeres Regisztráció esetén

{

'message' => 'Felhasználó sikeresen regisztrált'

}

login.php

Az Api ellenőrzi a metódust, illetve átveszi a változókat

 Ellenőrzi jött-e üres adat, ha igen hibát dob

Ellenőrzi az e-mailt és a jelszót

Beállítja a Session változókat

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Body | Válasz |
| POST | ./Assets/php/login.php | JSON | JSON |

Kéréstörzs (Request Body) JSON:

{

email:”[ex@gmail.com](mailto:ex@gmail.com)”

password:”123”

}

Lehetséges Hibaüzenetek:

{

//Üres mező esetén

'success' => false, 'message' => 'Email és jelszó szükséges'

//Nem helyes felhasználónév vagy jelszó

'success' => false, 'message' => 'Helytelen email cím vagy jelszó'

}

Sikeres Regisztráció esetén

{

'success' => true,

        'message' => 'Login successful',

        'logged\_in' => true,

        'user\_id' => $\_SESSION['user\_id'],

        'email' => $\_SESSION['email']

}

task.php

1.Az Api ellenörzi be van e lépve felhasználó és átveszi a dátumot, illetve a címke id-át egy $\_GET en keresztül

2. Ellenőrzi a dátumot, ha nincs hibát dob

3. Futtatja az alap lekérdezést, illetve ha a címke nem nulla hozzáfűzi azt a paramétert is illetve sorba rendezi fontosság alapján

4. Ha érkezik vissza feladat azt visszaküldi, de ha nem akkor külön üzenetet küld vissza

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Body | Válasz |
| GET | ./Assets/php/task.php | Üres | JSON |

GET paraméterek:

{

date:2025-04-16

label\_id:1 //Nem minden esetben kell használni

}

Lehetséges Hibaüzenetek:

{

//Nincs bejelentkezve felhasználó

'success' => false, 'message' => 'felhasználó azonosító szükséges'

//Nincs dátum

'success' => false, 'message' => 'Dátum szükséges'

//Sikertelen lekérdezés

'success' => false, 'message' => 'Query failed: ' . $e->getMessage()

}

Sikeres lekérdezés esetén

{

// nincs az adott napon feladata a felhasználónak

'success' => true, 'tasks' => [], 'message' => 'Nincs feladatod erre a napra'

//Van feladat

'success' => true, 'tasks' => $tasks

}

add\_task.php

1. Az Api ellenőrzi be van e lépve felhasználó és átveszi az adatokat

2. Ellenőrzi a feladat neve, illetve a dátum meg van e

3. Ellenőrzi a CheckForExistingTask functional létezik e már ilyen című feladat aznap

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Body | Válasz |
| POST | ./Assets/php/add\_task.php | JSON | JSON |

Kéréstörzs (Request Body) JSON:

{

t\_name:”Teszt”

description:”Teszt leírása”

label\_id: 1 // nem minden esetben szükséges

priority:1

progresson:0

deadline:2025-04-16

}

Lehetséges Hibaüzenetek:

{

//Nincs bejelentkezve felhasználó

'success' => false, 'message' => 'felhasználó azonosító szükséges'

//Nincs dátum vagy feladat név

'success' => false, 'message' => 'Feladat név és dátum szükséges'

//Már vettünk fel ilyen néven feladatot arra a napra

'success' => false, 'message' => 'Ilyen nevű feladat már létezik ezen a napon'

//Sikertelen lekérdezés

'success' => false, 'message' => 'Query failed: ' . $e->getMessage()

}

Sikeres beszúrás esetén

{

'success' => true, 'message' => 'Feladat sikeresen hozzáadva '

}

logout.php

1.Ellenőrzi a kérést típust

2.Nullázza a Sessiont

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Body | Válasz |
| POST | ./Assets/php/logout.php | Üres | JSON |

Lehetséges Hibaüzenetek:

{

// Nem megfelelő metódus

'error' => 'Method not allowed']

}

Sikeres kijelentkezés esetén

{

''success' => true

}

labels.php

Lekérdezi az összes címkét

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Body | Válasz |
| GET | ./Assets/php/labels.php | Üres | JSON |

Lehetséges Hibaüzenetek:

{

// Sikertelen lekérdezés

''success' => false, 'message' => 'Query failed: ' . $e->getMessage()]

}

Sikeres Lekérdezés esetén

{

'success' => true, 'labels' => $labels

}

profile\_details.php

Ellenőrzi a metódust, illetve be van e jelentkezve a felhasználó

Végrehajtja a lekérdezést

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Body | Válasz |
| GET | ./Assets/php/profile\_details.php | Üres | JSON |

Lehetséges Hibaüzenetek:

{

//Nincs bejelentkezve felhasználó

'success' => false, 'message' => 'felhasználó azonosító szükséges'

// Sikertelen lekérdezés

''success' => false, 'message' => 'Query failed: ' . $e->getMessage()

}

Sikeres Lekérdezés esetén

{

//Megvan a felhasználó

'success' => true, 'user' => $user

//Nincs ilyen felhasználó

'success' => false, 'message' => 'User not found'

}

get\_task.php

Átveszi a feladat id-át Get paraméterből

ellenőrzi, hogy sikeresen át lett e véve

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Body | Válasz |
| GET | ./Assets/php/get\_task.php | Üres | JSON |

GET paraméterek:

{

task\_id:1

}

Lehetséges Hibaüzenetek:

{

//Nincs feladat azonosító

'success' => false, 'message' => 'Feladat azonosító szükséges'

// Sikertelen lekérdezés

''success' => false, 'message' => 'Query failed: ' . $e->getMessage()

}

Sikeres lekérdezés esetén

{

//megvan a feladat

'success' => true, 'task' => $task

//Nincs meg a feladat

'success' => false, 'message' => 'Task not found'

}

delete\_task.php

1.Ellenőrzi be van e jelentkezve a felhasználó és bekéri a feladat id-át

2. Ellenőrzi megvan e a feladat id-ját

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Body | Válasz |
| GET | ./Assets/php/delete\_task.php | Üres | JSON |

GET paraméterek:

{

task\_id:1

}

Lehetséges Hibaüzenetek:

{

// Nincs bejelentkezve a felhasználó

'success' => false, 'message' => 'Feladat azonosító szükséges'

//Nincs feladat azonosító

'success' => false, 'message' => 'Feladat azonosító szükséges'

// Sikertelen lekérdezés

''success' => false, 'message' => 'Query failed: ' . $e->getMessage()

}

Sikeres lekérdezés esetén

{

//Sikeres törlés

'success' => true, 'message' => 'Feladat sikeresen törölve'

//Nem sikerül a törlés

'success' => false, 'message' =>'Feladatot nem sikerült törölni

}

change\_progress.php

1.Ellenőrzi a metódust és átveszi a két bemenő adatot

2. Ha ’POST’ a kiválasztott feladat előrehaladását 1 (Kész)-re teszi

3. Ha ’GET’ a aznap előtt lejárt feladatok előrehaladását 2 re teszi (Nincs kész) illetve a 30 napnál régebbi feladatokat törli

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Body | Válasz |
| POST/GET | ./Assets/php/change\_progress.php | JSON | JSON |

Kéréstörzs (Request Body) Json:

{

task\_id:1

}

Lehetséges Hibaüzenetek:

{

'success' => false, 'message' => 'Nem megfelelő feladat azonosito',"task\_id"=>$task\_id

// Sikertelen lekérdezés

''success' => false, 'message' => 'Query failed: ' . $e->getMessage()

}

Sikeres lekérdezés esetén

{

//sikeres előrehaladás frissités

'success' => true, 'message' => 'Fealdat sikeresen frissitve'

}

GET paraméterek:

{

//Üres

}

Lehetséges Hibaüzenetek:

{

//Nem támogattot metódus

'success' => false, 'message' => 'Methódus nem megengedett'

// Sikertelen lekérdezés

''success' => false, 'message' => 'Query failed: ' . $e->getMessage()

}

Sikeres lekérdezés esetén

{

'success' => true, 'message' => 'Feladatok sikeresen frissitve régi feladatok törölve'

}

**all\_task.php**

1. Ellenőrzi be van e jelentkezve a felhasználó

2. Átveszi a bejövő adatokat

3. 2 lehetőség van vagy a jövőbeli feladatokat kérjük le vagy a múlt bélieket ezt az type bejövő paraméter dönti el

|  |  |  |  |
| --- | --- | --- | --- |
| Metódus | URL | Body | Válasz |
| GET | ./Assets/php/all\_tasks.php | JSON | JSON |

**Get paraméterek**

{

//alapesetben future

type: ’past’

}

Lehetséges Hibaüzenetek:

{

'success' => false, 'message' => 'Felhasználó nincs bejelentkezve'

'success' => false, 'message' => 'Nem megfelelő típus'

// Sikertelen lekérdezés

''success' => false, 'message' => 'Query failed: ' . $e->getMessage()

}

Sikeres lekérdezés esetén

{

'success' => true, 'események' => $events

}

Tesztelés

Az alkalmazás működését manuális teszteléssel ellenőriztük, amely során különböző tipikus és szélsőséges felhasználói műveleteket próbáltunk ki annak érdekében, hogy feltárjuk az esetleges hibákat. A tesztelés az alábbi fő szempontok szerint zajlott:

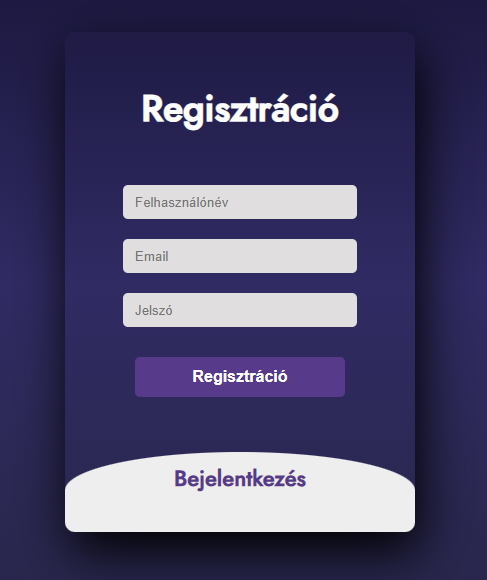
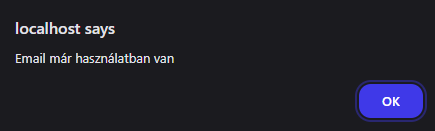
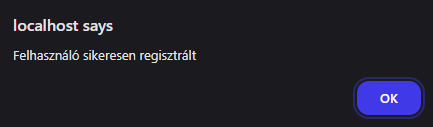
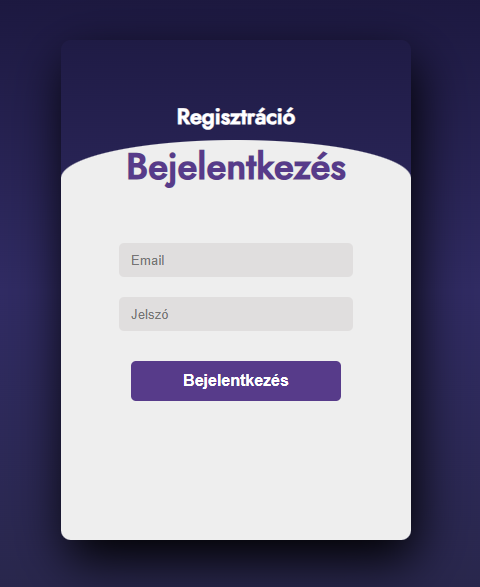
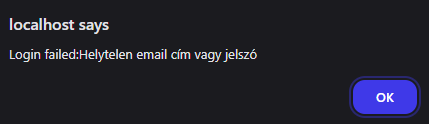
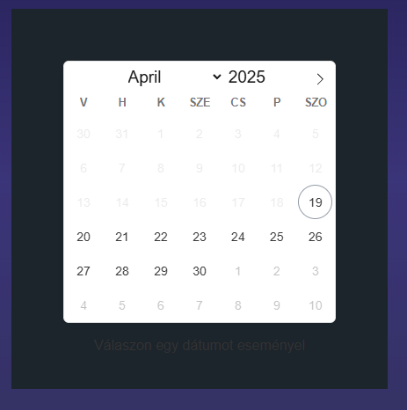
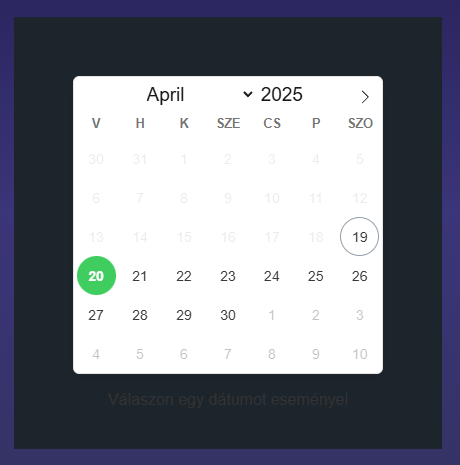
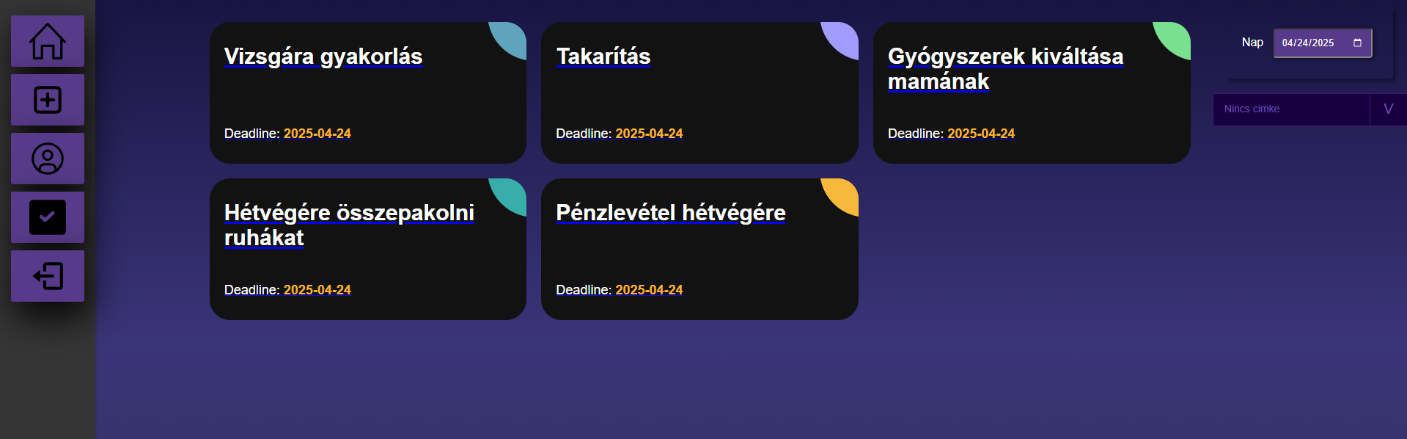
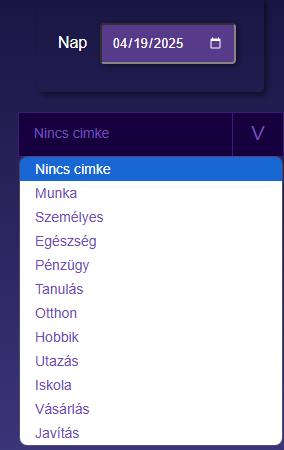
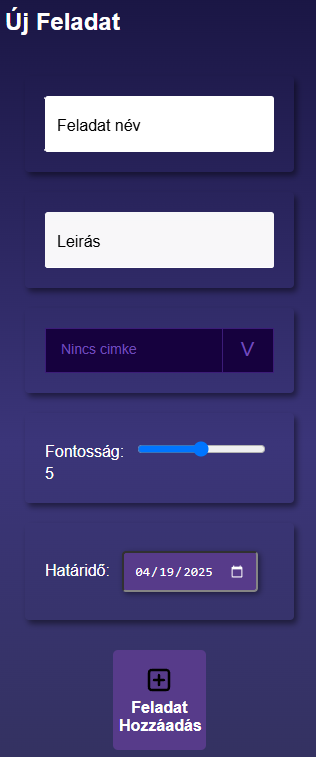
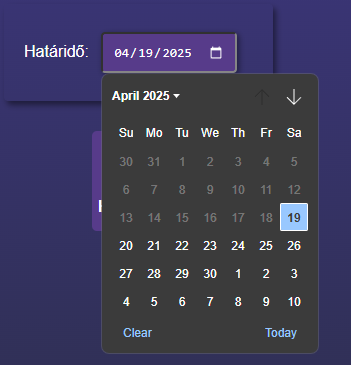
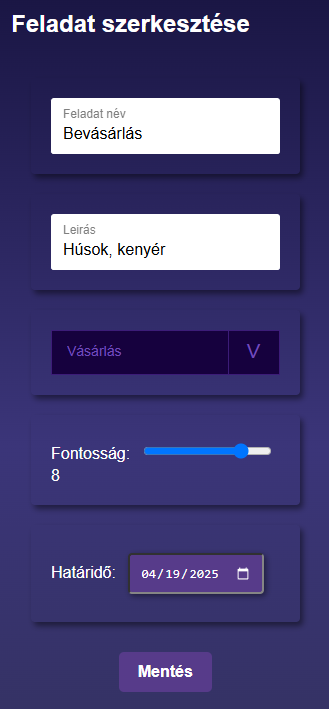
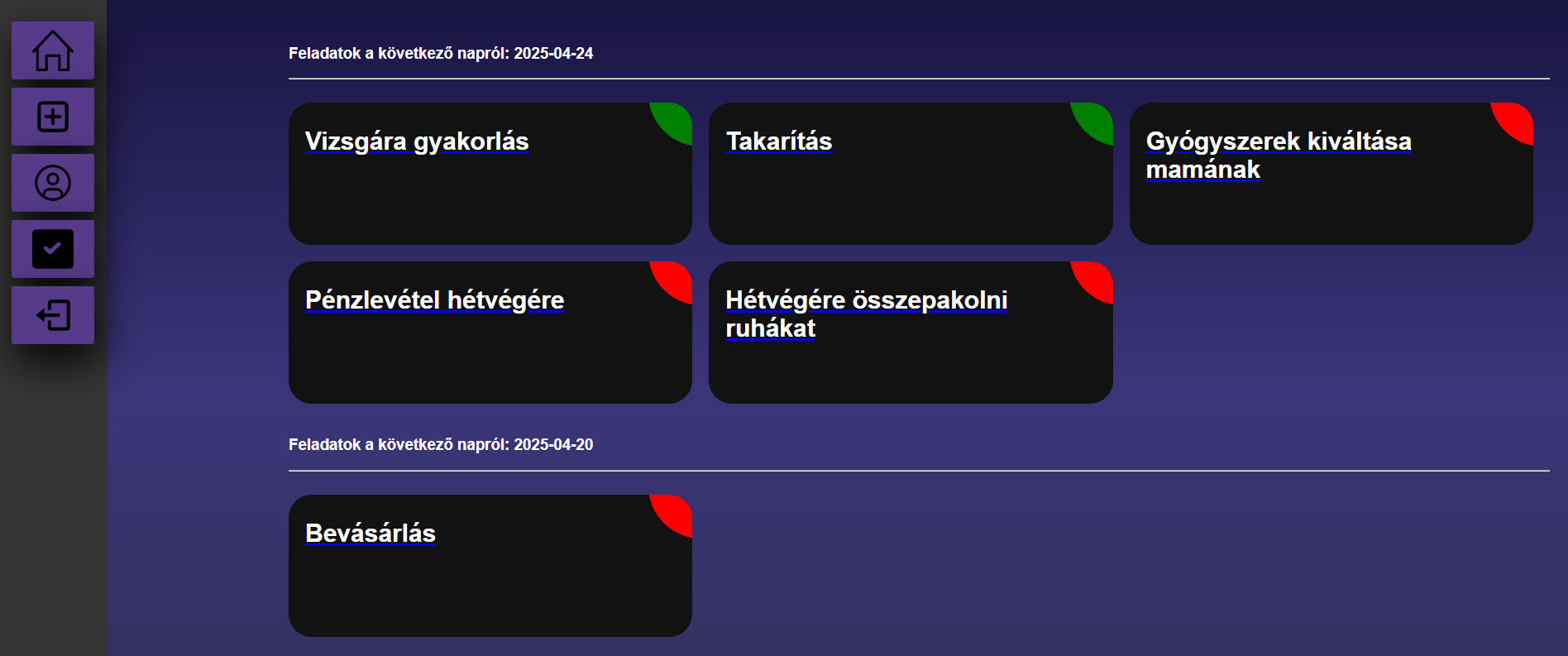
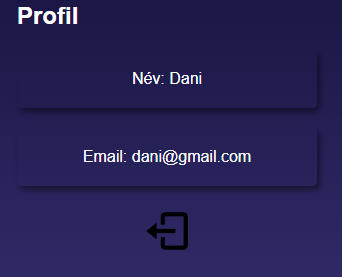
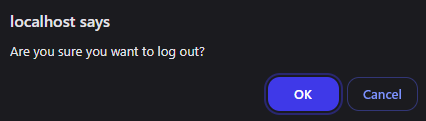
* **Duplikáció kezelése**: ugyanannak a feladatnak a kétszeri felvételének próbája annak ellenőrzésére, hogy a rendszer megfelelően kezeli az ismétlődő bejegyzéseket.
* **Regisztrációs ellenőrzés**: ugyanazzal az e-mail címmel történő többszöri regisztráció tesztelése, a felhasználói azonosítás és hibakezelés helyességének vizsgálatára.
* **Nagy adatmennyiség kezelése**: nagyszámú feladat felvételével ellenőriztük, hogyan viselkedik a frontend felület, illetve milyen hatással van a teljesítményre és a megjelenítésre.
* **Új feladatok rögzítése**: az új feladatok felvételénél teszteltük a fontossági szint beállítását a csúszkán keresztül, különös figyelmet fordítva arra, hogy a nagyobb fontosságú feladatok valóban előrébb (vagy megfelelő helyre) kerülnek-e.
* **Feladatok kezelése**: a meglévő feladatok szerkesztését és törlését is teszteltük, annak érdekében, hogy minden módosítás és törlés megfelelően frissüljön a rendszerben és a felületen.
* **Címkék szerinti szűrés**: kipróbáltuk a címkék alapján történő szűrést is, hogy megbizonyosodjunk róla, hogy a felhasználók hatékonyan tudják szűkíteni és rendszerezni a feladataikat különböző kategóriák szerint.

A fenti manuális tesztelési lépések segítettek abban, hogy megerősítsük az alkalmazás stabilitását, használhatóságát, és minimalizáljuk a hibák előfordulását éles környezetben.

**Felhasználói dokumentáció**

A program használatához a XAMPP nevezetű alkalmazás szükséges. Telepítés után az Apache és a MySQL modulok elindítása szükséges.  
A modulok, ha elindultak a böngészőben a localhost megnyitása után a phpMyAdmin fülön egy új adatbázist kell létrehozni “quicktick” néven, majd ebbe importálni a “todolist.sql” fájlt. Ez után a XAMPP mappáján belül a “htdocs” nevezetű mappába kell behelyezni a projekt mappáját. Ezt követően a böngészőbe a “localhost/QuickTick/ToDoList/WebPage/” -el ugorhatunk a weboldalra.

**A program áttekintése:**

1. **Regisztráció**  
   A regisztrációkor a felhasználónak meg kell adnia ez felhasználónevet, egy emailt, és egy jelszót. Az email és a jelszó a bejelentkezéshez lesz szükséges a későbbiekben.  
     
     
   Az email megadásánál kötelező egy @ használata, ha ez nem teljesül akkor erre egy hibaüzenetet is kapunk.  
     
     
     
     
   Ha már egy regisztrált email címmel próbál meg a felhasználó regisztrálni akkor a következő üzenet jelenik meg képernyőn.  
     
     
     
   A sikeres regisztrációról pedig egy megerősítés ugrik fel a képernyő tetején.  
     
   
2. **Bejelentkezés**  
   Abejelentkezéshez meg kell adnia egy már regisztrált email-t és az ahhoz tartozó jelszót. A bejelentkezés után a felhasználó láthatja a saját oldalát, ahol a saját feladatait tekintheti meg.  
     
     
     
     
   Az alábbi hibaüzenet jelenik meg akkor, ha a felhasználó rossz email címet vagy jelszót ad meg.  
     
   
3. **Főoldal**  
   Ha egy új felhasználó lép be, vagy egy olyan nap van kiválasztva, ahol nincs feladat felvéve akkor a következő kinézet fogadja:  
     
     
     
   Ha van másik napra felvéve feladat akkor pedig zölddel jelenik meg az adott nap, ahol van feladat.  
     
     
     
   Ha az adott napon van feladata a felhasználónak akkor pedig az alábbi kinézettel találkozik:   
   Ha az egyik kártyára kattintunk akkor pedig az alábbi kinézet jelenik meg.  
   Itt tudjuk a feladatot szerkeszteni, törölni, illetve készként megjelölni, amely három egyértelműen megkülönböztethető ikonnal van jelölve.  
   Az adott ikonra kattintva szerkeszthetjük törljetjük illetve, késznek jelölhetjük az adott feladatot.  
   A kártya háttér színe pedig a címkétől függ, amely a feladat felvételekor beállítható.  
     
     
   A feladatokat címkék szerint lehet szűrni, amely segíti az átláthatóságot egy olyan napon, ahol nagyobb mennyiségű feladat van felvéve.  
     
   A jobb felső sarokban a dátum alatt lenyitható listával kiválasztható, hogy csak az adott címkékkel ellátott feladatok jelenjenek meg.  
     
   A nincs címke pedig megjeleníti az összes feladatot.  
   
4. **Új feladat hozzáadása**  
   Amikor új feladatot szeretnénk felvenni, akkor a bal oldali sávon a + jel kinézetű gombra kattintva tehetjük meg, amely a következő képernyőre viszi a felhasználót.  
     
     
     
   Itt a felhasználónak meg kell adnia egy nevet, leírást, kiválasztania egy címkét, beállítania egy fontosságot, illetve a határidőt kiválasztani.  
     
   A címkék a szűrésben nyújtanak nagy segítséget, a fontosság pedig a sorba rendezés miatt szükséges. Minél nagyobb fontosságot állítunk be, annál előrébb lesz a feladatok között. Ez egy 0-10-es csúszkán állítható.  
     
   A határidőre való kattintással megjelenik egy naptár, ahol kiválasztható a kívánt dátum.  
     
     
     
     
   A feladat felvételét pedig a határidő alatti Feladat hozzáadás gombbal tudjuk megtenni.  
     
     
     
   Sikeres feladat hozzáadásakor pedig a megnyomott gomb alatt egy sikert jelző üzenet jelenik meg.  
     
     
     
   Ha már egy meglévő feladatot újra megpróbálunk felvenni ugyan arra a napra akkor a következő hibaüzenet jelenik meg.  
     
   
5. **Feladatt állapotának szerkesztése**  
   A szerkesztéskor az új feladat felvételéhez hasonló oldal fogadja a felhasználót, ahol a feladat felvételénél megadott adatokat tölti be az oldal.  
     
   Itt módosítható a feladat neve, leírása, címkéje, fontossága, illetve határideje.  
     
   A módosítások után a mentés gombra kattintva, a változtatások elmentődnek és a felhasználó vissza kerül a főoldalra, ahol láthatja a módosított feladatot.  
     
   
6. Régi feladatok megtekintése  
   A határidőig nem teljesített feladatok itt jelennek meg, dátum szerint elkülönítve.  
   A kész feladatok zölddel vannak jelölve, a pirossal pedig azok, amelyek nem lettek késznek jelölve.  
     
   
7. **Profil**  
   A profil fülre kattintva láthatjuk a bejelentkezett felhasználó nevét, illetve email címét, alatta pedig egy kijelentkezés gombot.  
     
     
     
   A gomb megnyomására egy felugró ablakban a felhasználónak meg kell erősítenie a kijelentkezési szándékát.  
     
     
     
   A kijelentkezés megerősítése után pedig a regisztrációhoz, illetve a bejelentkezéshez dob vissza az oldal.

A QuickTick tovább fejlesztésének lehetőségei

Summary in English