Vizsgaprojekt Dokumentáció

Szerk.: Kérészi Martin, Drávucz Márton
2025-2026

Készült a 2025/2026-ös tanév szakmai vizsgájára. HMSZC ReZsi Technikum
https://github.com/DamaszkusziRavaszbuzi/vizsgaprojekt

Tartalomjegyzék

Bevezető

Projektünk egy modern, interaktív szókincsfejlesztő webes alkalmazás oktatási célokra.
Célja a hagyományos papír alapú szókártyák kiváltása egy környezetbarátabb, digitális megoldással, amely intuitív funkcióival hatékonyabbá teszi a nyelvtanulást.

A programot készítették:
Drávucz Márton:
https://github.com/DamaszkusziRavaszbuzi
- -Backend
- -Algoritmizálás
- -Adatbázis struktúra
- -Dokumentáció

Kérészi Martin: https://github.com/kenyesz2
- -Frontend és Design
- -Dokumentáció
- -UX
- -UAT

A csapatmunkát a Discord felületén, privát csevegésben valósítottuk meg. A megírt kódokat, kódrészleteket ott osztottuk meg egymással, a program tervezését, technikai megvalósításait személyesen egyeztettük.
GitHub-ot csak a fejlesztés visszakövethetősége érdekében, illetve a régi verziók archiválására használtuk. Csapatmunkára az iskolában szerzett gyakorlat hiányában, illetve személyes preferenciából nem használtuk.

Projektünk során használtunk AI-t. Mivel az informatikának, legfőképp a szoftverfejlesztésnek ma már szerves része, ezért nem tartottuk érdemesnek sem mellőzni, sem pedig tagadni a használatát. Egy informatikus, programozó számára elengedhetelen, hogy tudása naprakész legyen, illetve lépést tartson a technikai fejlődésével. A projekt során előkerülő problémák kiküszöbölésére használtuk. Ilyen volt például:

-Refaktorolás
-Debugging, hibakeresés

Funkciók

Programunk több, a nyelvtanulást segítő funkcióval is rendelkezik. Többek között: a felhasználó hozzáadhat szavakat és azok fordítását, ezeket virtuális szókártyák formályában megtekintheti, illetve a megszerzett nyelvi tudását ellenőrizheti.
Minden felhasználó rendelkezik egy saját “szótárral”, melyet egy adatbázisabn tárol a program.

A funkciók csoportosíthatóak a CRUD-műveletek szerint:

-Create (Léterehozás):
- -Szavak hozzáadása manuálisan
- -AI által véletlenszerűen választott szavak hozzáadása
- -AI által ajánlott szavak hozzáadása

-Read (Olvasás):
- -Szópárok megjelenítése szókártyák formájában
- -Szavak véletlenszerű gyakorlása
- -A gyakran elhibázott, az algoritmus által gykorlásra szorulónak ítélt szavak gyakorlása

-Update & Delete (Szerkesztés és Törlés):
- -A felhasználói élény szempontjából minimális jelentőséggel rendelkeznek, ezért a szavak módosítása és törlése egy külön meüpontban kaptak helyet.

Funkciók leírása részletesen:

Regisztráció/bejelentkezés:

A felhasználók egy fh.név+jelszó kombinációjával tudnak regisztrálni, illetve bejelentkezni. A jelszó módosítására bejelentkezés után lehetőség van a beállításokban.

Beállítások:

A beállítások menüpontban módosítható a jelszó, illetve a UI téma.

Új szó hozzáadása:

A felhasználó manuálisan hozzáadhat a szótárához egy szót és annak fordítását.

Új random szó:

A program egy véletlenzerűen kiválasztott szó+fordítás párost ajánl fel a felhasználónak, amit ő elfogadhat/elutasíthat.

Új ajánlott szó:

A program a felhasználó jelenlegi szavait figyelembe véve ajánl új szavakat a felhasználónak, amit ő elfogadhat/elutasíthat.

Random:

A program a felhasználó szótárából véletlenszerűen választ ki szavakat, melynek a fordítását a felhasználónak kell kitalálnia, ezzel fejlesztve a szókincsét.

Ebben a módban a felhasználónak lehetősge van:

- -Visszatérni a főmenübe
- -Segítséget kérni:
  - -Feleletválasztós
  - -Magánhangzók megjelenítése
- -Feladni
- -Megfordítani a fordítás irányát
- -Ellenőriznie a megoldást

Gyakorlás:
A program nyomon követi a felhasználó tudását, hogy melyik szavakat ismeri. Az algoritmus a gyakorlásra szoruló szavakat részesíti előnyben, működése megegyezik a “Random” funkcióval.

Kártyák:
A felhasználó szótárából véletlenszerűen kiválasztott szavak virtuális kártyák formályában kerülnek megjelenítésre. Ezeket a felhasználónak lehetősége van mozgati, illetve felfordítani, ezzel mimikálva a valódi, fizikai szókártyákat. A felhasználó törölhet, illetve hozzáadhat a megjelnített kártyákhoz.

Szavak kezelése:
A felhasználó örölheti, illetve módiíthatja a szótárában szereplő szavakat.

Statisztika:
A felhasználó megtekintheti a program által alkotott statisztikát a felhasználó szókincséről.

Technikai megvalósítások: Frontend

A frontend HTML, CSS, illetve JavaScript-ben készült. A program kizárólag webes felülettel rendelkezik. Elsősorban asztali felhasználásra készült, de rendelkezik mobilon is használható, egy natív alkalmazással megegyező élményt nyújtó, reszponzív felülettel, ezzel eleget téve a projektvizsgára vonatkozó követelményeknek.

Több okból döntöttünk a kizárólag webes felület mellett:

- Platformfüggetlen
- Széles körben elterjedt
- Könnyen létrehozható dinamikus felület
- A programnak nincs szüksége a hardveres erőforrások közvetlen elérésére
- Egy app-hoz képest kevésbé intrúzív
- A fh-nak nincs szüksége az alkalmazást frissítenie

A frontend fejlesztés, design során fontos szempont volt a UX (felhasználói élmény). Több design döntés is segíti a program intuitív használatát:

- Kontrasztos színek
- Nagyméretű gombok
- Ikonok helyett egyértelmű szövegek

A UI tervezése közben nagy hangsújt fektettünk a fizikai szókártyák egyszerűségének megőrzésére. Fölösleges “gimmick” funkciók helyett egy egyszerű, letisztult felhasználói felület.

A frontend JavaScript restAPI-on keresztül kommunikál a backend-del, használva a HTTP metódusokat (GET, POST).

Szintén nagy hangsújt fektettünk a tiszta kód egyik alapelvére, a modulárisságra. A HTML, CSS, JS kódok, külön-külön fájlokban kaptak helyet, oldalak, illetve funkciók szerint szeparálva, ezzel nagyban elősegítve a program jövőbeli fejlesztését.

Itt látható a projekt könyvtárstruktúrája:

```
user@T450:~/VIZSGA/main$ tree
.
├── app.py
├── database.db
├── database.db.hmac
├── dictionary.json
├── static
│   ├── js
│   │   ├── autonew.js
│   │   ├── cards.js
│   │   ├── edit.js
│   │   ├── menuToggle.js
│   │   ├── new.js
│   │   ├── randomPractice.js
│   │   ├── settings.js
│   │   ├── statistics.js
│   │   ├── styleLoader.js
│   │   └── themeChange.js
│   ├── reset_password.gif
│   ├── styles
│   │   ├── components
│   │   │   ├── autonew.css
│   │   │   ├── cards.css
│   │   │   ├── edit.css
│   │   │   ├── landing.css
│   │   │   ├── mainMenu.css
│   │   │   ├── new.css
│   │   │   ├── practice.css
│   │   │   ├── settings.css
│   │   │   └── statistics.css
│   │   ├── global.css
│   │   ├── responsive.css
│   │   └── themes
│   │       ├── themeDark.css
│   │       ├── themeDarkGreen.css
│   │       └── themeDarkPurple.css
│   └── teapot.gif
└── templates
    ├── autonew.html
    ├── base.html
    ├── cards2.html
    ├── coffee.html
    ├── edit.html
    ├── index.html
    ├── landing.html
    ├── new.html
    ├── practice.html
    ├── settings.html
    ├── statistics.html
    └── xd.html
```

Technikai megvalósítások: Backend

A backend Pythonban íródott, a Flask[1] keretrendszert használva. A backendhez egy SQL adatbázis van csatolva.

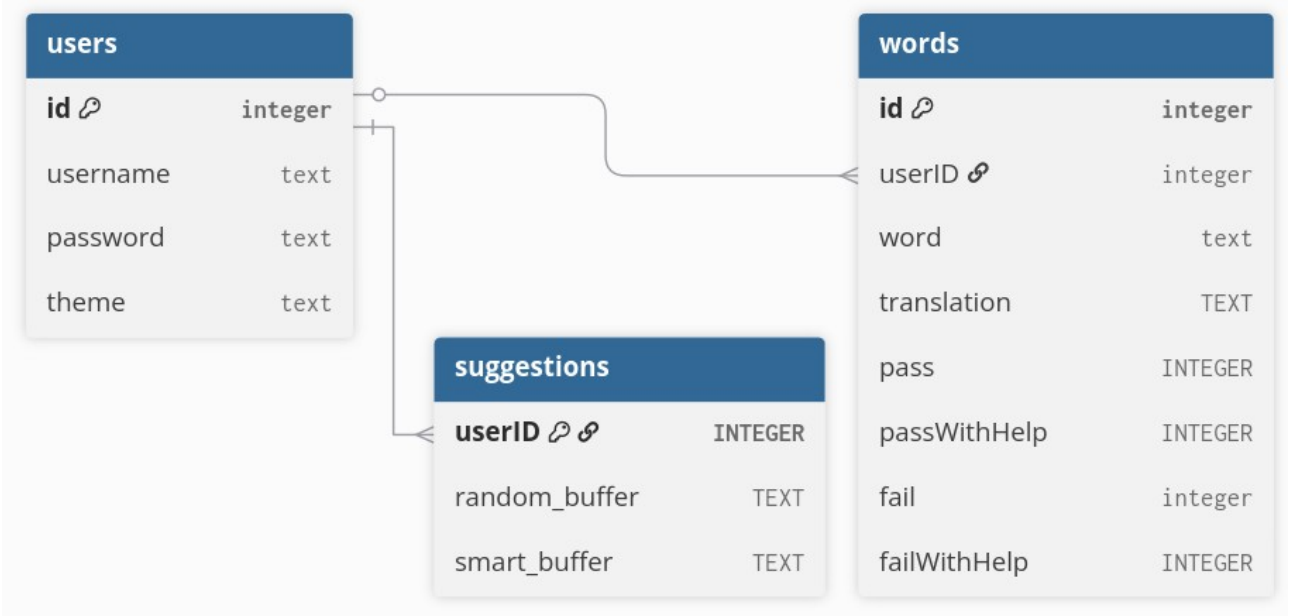
A felépítése javarészt követi a szabványos Flask backend struktúráját - pl.:

```
///
@app.route('/login')
    def login():
        …
///
```

-így ezt nem érdemes részletezni.

Adatbázis:

Az adatbázis struktúráját az alábbi relációs ábra jól szemlélteit:



[2]

Az azdatbázist a backend hasheli, a HMAC kritptográfiai technika segítségével.

A felhasználók és a szavak külön-külön táblákban vannak tárolva.
Mindekettő esetében egy egyedi ID a PK, a userID alapján vannak összekapcsolva.
Az AI-generált szó-fordítás javaslatokat a backend az adatbázisban „pre-cache”-li.

A backend két fájlból áll:
app.py
    A tényleges backend.
run_app.py
    Helper script. Létrehozza a venv-et (ha még nincs) illetve telepíti a szükséges modulokat. Ha minden követelmény teljesül, futtatja az app.py-t.

run_app.py kapcsolói:
    run_app.py: Létrehoz egy .venv-et, ha még nincs, telepíti a szükséges függőségéeket, majd futtatja az app.py-t
    run_app.py --install-only: Létrehoz egy .venv-et, telepíti a függőségeket, majd kilép.
    run_app.py –venv-dir <dir>: Megadható saját venv elérési út.

Lényegesebb algoritmusok:

“Magabiztosság” (confidenceIndex) kiszámítása:
    A program nyomon követi a felhasználó válaszainak helyességét a Random és Gyakorlás módban. A felhasználó válaszait 4 értékelésbe lehet besorolni:
    -**Pass**: A felhasználó jó választ adott meg
    -**PassWithHelp**: A felhasználó jó fálaszt adott meg segítséggel
    -**Fail**: A felhasználó helytelen választ adott meg vagy feladta

-**FailWithHelp** A felhasználó helytelen választ adott meg vagy feladta segítséggel

Ezeket a backend számon artja az adatázisban (lást: fentebbi ábra), és ezek alapján számol ki egy “magabiztossági indexet” (a programban: confidenceIndex, továbbiakban: **cI)**, mely megközelítőleg megmutatja, hogy a felhasználó mennyire ismeri az adott szót, mennyire biztos a tudásában.
A következő képlettel lehet kiszámítani:
cI = pass*2+passWithHelp-fail-failWithHelp*2
Egy helyes válasz 2, a segítséggel megadott helyes válasz 1, a hibás megoldás -1, illetve a hibás megoldás segítséggel -2 “pontot” ér.
Ez a változó a Gyakorlás módban rendelkezik jelentős szereppel.

Gyakorlás:

A negatív cI-vel rendelkező szavakat kiválogatja a backend. Ha 5-nél több ilyen szó van, azokat átadja a frontendnek, ami azokat véletlenszerű sorrendben “gyakoroltatja” a felhasználóval. Minden szó csak 1x kerül kiválasztása a frontenden. Ha elfogytak a szavak, a backend újra kiszámítja a cI-ket, majd a ciklus kezdődik előről.

Amennyiben 5-nél kevesebb negatív cI-vel rendelkező szó van a negatív cI-vel rendelkező szavak kiválasztásra kerülnek, a kimaradt szavak pedig a cI-jük alapján növekvő sorrendben rendeződnek. A lista elejéről (alacsony cI) addig kerülnek szavak kiválasztásra, amíg a kiválasztott szavak száma el nem éri a 10-et. Ekkor a szavak ismét átadásra kerülnek a frontendnek, …

Új random és ajánlott szó:

Programunk rendelkezik AI alapú szóajánlással. Képes random szavakat, illetve „okos” szavakat ajánlani. Az utóbbi esetében figyelembe veszi a felhasználó már meglévő szavait, és azok alapján ajánl újakat.

Jelenleg az Ollama[3] keretrendszer segítségével képes interfészelni egy helyileg futattot nyelvi modelt, mely jelen esetben a Google által fejlesztett Gemma3 modell 4b[4] verziója.

A következő proptokkal kér új szavakat az AI-tól:

„A set of words is given. Give me 4 completely random English word that matches the commonness/level of the given words. Also give the words' translation in Hungarian, separate the word and it's translation by a ":". Begin the next word in a new line. The given words are: [SZAVAK]. Don't think for long. Pick words that have exact translations.”

„Give me 4 completely random English word. Also give the words' translation in Hungarian, separate the word and it's translation by a ":". Begin the next word in a new line. Don't think for long. Pick words that have exact translations.”

A program jeleneg a bacend indulásakor minden felhasználó számára „pre-cache”-li szavakat. Ennek célja a felhasználói élmény jevítása a futásidő alatt. Mivel korlátozott hardver-erőforrásokkal rendelkezünk, ezért volt szükség erre a funkcióra. A későbbiekben lehet elhanyagolható lesz.

A projekt jövője

Mivel ezt a projektet csupán vizsgaprojetként fejlesztettük, ezért a továbbiakban nincs vele tervük.

De elméletben még sok mindent lehetne rajta módosítani, fejleszteni:

- **”Production Release”:** A szoftver kiadása, és az az előtti lépések:
  - GitHub oldal, telepítési, üzemeltetési instrukciókkal, dokumentációval, angol nyelven
  - Licenszelés, pl: GNU General Public License

Programunkat úgy terveztük, hogy azt könnyedén lehessen self-hostolni (pl: végfelhasználók, vagy akár iskolák által), de biztosítanánk szervert azon felhasználók számára, akik nem szeretnék/nem tudnák maguknak üzemeltetni a webappot.
Ennek több lépése, eleme lenne:

- Megfelelő szervergép
- Domain + TSL tanusítvány
- Támadások elleni védelem (DDoS,…)
- Nyelvi model futtatására alkalmas hardver
- stb...

Csak hát ez drága, nekünk meg kevés az ösztöndíj, nincs rá pénz :/

- **Nyílt forráskódúvá tétel:** Más fejlesztők is részt tudnak venni a projekt fejlesztésében. Mivel egy nyítforrású program teljesen átlátható, ezért a felhasználók nagyobb bizalommal haszálnák.

- **Megfelelő biztonsági funkciók, titkosítás, támadások elleni védelem:** e-mail-hez kötött

regisztráció, adatbázis titkosítása, HTTPS, SQL injection ellni védelem… Mivel ezek a funkciók ma már elengedhetelenek, sok széles körben elterjedt módszer létezik, miket nem lenne kihívás implementálni.

-**További nyelvek:** Jelenleg csak Angol-Magyar nyelvet támogat, demonstráció képpen. A moduláris felépítéséből adódóan könnyű lenne más nyelveket hozzáadni, csupán az UI elemek szövegeit kellen lefordítani. Egy fordítás-kezelő szolgltatással, mint a Crowdin, POEditor, a mások által létrehozott fordítások integrálása egyszerű lenne.

-Külső AI API integráció: A program jelenleg csak helyileg futtatott nyelvi modelleket támogat az Ollama keretrendszeren keresztül. Ha biztosítanánk integrációt nyelvi modellek használatához API-okon keresztül (Pl: OpenAI Chat-GPT), az nagyban megkönnyíteni a program self-hostolását, mivel a webappot futtatót felhasználóknak nem lenne szükségük saját hardveren futtatni nyelvi modelleket, illetve hozzáférnének jobb, fejletebb, gyorsabb modellekhez. A felhasználóknak meg kellene adniuk a saját API kulcsukat.

Appendix

[1] https://flask.palletsprojects.com/en/stable
[2] https://dbdiagram.io
[3] https://ollama.com
[4] https://deepmind.google/models/gemma/gemma-3/