GYMNASIUM JANA KEPLERA

Parléřova 2/118, 169 00 Praha 6

Denní matematikcá úloha

Maturitní práce

Autor: Ekaterina Danilina

Třída: R8.A

Školní rok: 2022/2023

Předmět: Informatika

Vedoucí práce: Šimon Schierreich

Praha, 2023

GYMNASIUM JANA KEPLERA
Kabinet informatiky

ZADÁNÍ MATURITNÍ PRÁCE

Student:	**Ekaterina Danilina**
Třída:	**R8. A**
Školní rok:	**2022/2023**
Vedoucí práce:	**Šimon Schierreich**
Název práce:	**Denní matematická úloha**

Pokyny pro vypracování:

První částí projektu by bylo vytvořit databázi úloh s číselným řešením. Druhá část je vytvořit web, na kterém by se každý den ukázala jedna úloha všem uživatelům. Ti pak mohou odevzdat svůj výsledek do systému, který jim to vyhodnotí. Po odevzdání správné odpovědi se uživateli zobrazí vlastní statistika (kolik úloh zvládl na první pokus v kuse, jak často dělají chyby…) a statistika dané úlohy (kolik uživatelů ji vyřešilo a na jaký pokus).

Doporučená literatura:

[1] KOTTWITZ, Stefan. ***LaTeX Beginner's Guide: Create high-quality, professional-looking documents and books for business and science using LaTeX.*** **Packt Publishing, 2011. ISBN: 978-1-847-19986-7**

[2] ESPOSITO, Dino. ***Modern Web Development.*** **Microsoft Press, 2016. ISBN: 978-1-509-30001-3**

URL repozitáře:

`https://github.com/Ekatka/DailyAMC`

student *vedoucí práce*

V Praze dne 29. 9. 2022

Prohlášení

Prohlašuji, že jsem svou práci vypracovala samostatně a použila jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů. Nemám žádné námitky proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Praze dne 24. března 2023 Ekaterina Danilina

Poděkování

Děkuji ing. Šimonu Schierreichovi za vedení práce.

Abstrakt

V této práci popisuji implementaci maturitního projektu z informatiky, jehož hlavním úkolem bylo vytvořit webovou stránku, která každý den zobrazí uživatelům novou matematickou úlohu. Projekt byl realizován s využitím programovacího jazyka Python a konkrétně pak pomocí frameworku FastAPI, na generování templatů byla použita knihovna Jinja2. Pro ukládání a správu dat byla použita databáze MySQL.

Tato práce dále obsahuje podrobný popis všech částí implementace, které zahrnují návrh databázového schématu, implementaci funkcí pro získávání a ukládání matematických úloh a také návrh a tvorbu uživatelského rozhraní webové stránky. Jsou zde také popsány použité technologie, knihovny a nástroje, které byly nezbytné pro úspěšné dokončení projektu.

Klíčová slova

matematika, web

Abstract

In this thesis, I describe the implementation of a computer science graduation project, the main task of which was to create a website that displays a new mathematical problem to users every day. The project was implemented using the Python programming language, specifically with the FastAPI framework, and the Jinja2 library was used for generating templates. MySQL was used for storing and managing data.

This thesis also includes a detailed description of all parts of the implementation, which include the design of the database schema, the implementation of functions for obtaining and storing mathematical problems, and the design and creation of the website's user interface. The technologies, libraries, and tools used, which were essential for the successful completion of the project, are also described.

Keywords

math, web

Obsah

1 Teoretická část **3**
1.1 Cíl práce 3

2 Implementace **5**
2.1 Tvorba databáze 5
2.2 Tvorba stránky 6
2.3 Spouštění webu a resetování hesla 7

3 Technická dokumentace **9**
3.1 Instalace 9
3.2 Používání 9

Závěr **11**

Seznam obrázků **15**

1. Teoretická část

1.1 Cíl práce

Cílem projektu je vytvořit webovou stránku, na které by se každý den ukázala nová úloha ze soutěže AMC12. AMC je Americká matematická soutěž pro středoškoláky, ve které je 25 úloh, které jsou seřazeny podle obtížnosti. AMC12 je určena pro nejstarší žáky, a je tedy nejtěžší. Soutěž AMC jsem vybrala kvůlistupňování obtížnosti, díky kterému mohu v průběhu týdne dávat těžší a těžší úlohy, a také kvůli tomu, že ke každé otázce je pět různých odpovědí, takže uživatel může vybírat z možností. Dalším důvodem je komunita na stránce `https://artofproblemsolving.com/`, díky které je ke každé úloze minimálně jedno řešení.

Součástí projektu je i možnost registrování a následné přihlášení uživatele. Přihlášenému uživateli by se ukládaly odpovědi a po zodpovězení se vypsaly statistiky, například streak (počet správně zodpovězených úloh v řadě), počet celkem zodpovězených úloh a statistiky podle dní.

Projekt může pomoct v přípravě na matematické olympiády, případně s přípravou na maturitu z matematiky.

2. Implementace

2.1 Tvorba databáze

Implementaci jsem rozdělila na dvě části. První spočívala ve vytvoření databáze úloh, druhá v realizaci samotné stránky. Úlohy jsem brala ze stránky `https://artofproblemsolving.com/`, kde jsou sepsaná zadání a řešení všech AMC úloh a všechny matematické výrazy jsou zapsány v o LaTeXu.

Pro parsing zadání jsem použila Python knihovnu BeatifulSoup. Na ukládání zadání úloh, jejich řešení a uživatelů jsem si zvolila MySQL databázi. Hlavně kvůli podpoře Pythonu a dobré dokumentaci. Při parsingu zadání jsem měla několik problémů.

1. Starší úlohy měly na stránce jiné fonty, takže když jsem hledala řešení pomocí regulárního výrazu, občas jsem dostala jenom část řešení.
2. Hodně geometrických úloh obsahovalo obrázky dělané pomocí jazyku Asymptote, ke kterému se mi nepodařilo najít dokumentaci na generaci ve webu. Druhá možnost by byla ukládat všechny obrázky.

První problém jsem vyřešila rozdělením na případy, druhý se mi vyřešit nepodařilo a rozhodla jsem se neukládat úlohy, které obsahují Asymptote.

Po parsování zadání AMC12 z posledních 15 let jsem dostala přibližně 300 úloh, kterým jsem přiřadila obtížnost od 1 do 7. Ke každé úloze jsem si uložila odkaz na stránku s řešením. V databázi mám celkem 5 tabulek. Nejdůležitější je tabulka *Problems*. Zde je ukázka z ní.

id	statement	difficulty	original	answer
100	Five positive consecutive integers starting with ...	3	https://artofproblemsolving.c...	a+4
101	Three congruent isosceles triangles are constr...	3	https://artofproblemsolving.c...	\dfrac{\sqrt3}3
102	David drives from his home to the airport to ca...	3	https://artofproblemsolving.c...	210
103	Two circles intersect at points A and B. T...	4	https://artofproblemsolving.c...	2+\sqrt3
104	A fancy bed and breakfast inn has 5 rooms, ...	4	https://artofproblemsolving.c...	2220
105	Let $a<b<c$ be three integers such that $a,b,...	4	https://artofproblemsolving.c...	2
106	A five-digit palindrome is a positive integer with...	4	https://artofproblemsolving.c...	18

Obrázek 2.1: Ukázka z tabulky Problems

Statement je samotné zadání, difficulty je přiřazená složitost, original je odkaz na řešení a answer je správná odpověď.

Dále mám tabulku *Assignments*, kde podle obtížnosti přiřazuji úlohám datum, kdy budou zobrazeny. V pondělí, by se měly zobrazovat úlohy, kde difficulty je 1, úterý 2 a tak dále. V tabulce emphSolutions jsou uloženy možné odpovědi (ke každé úloze 5) a to, jestli je daná odpověď správná, či ne. Tyto tři tabulky musí být vyplněné před spuštěním programu.

Zbylé dvě tabulku jsou *Users*, kde ukládám email a hash hesla uživatele, a tabulka *Answers*, ve které ukládám id uživatele, id úlohy, zda odpověděl správně a jaká byla jeho odpověď.

Později jsem zjistila, že pro možnost resetování hesla potřebuji přidat do tabulky *Users* sloupec reset

token, jehož hodnota je Null, ale ve chvíli, kdy si uživatel chce změnit heslo, uloží se do ní token, který se po úspěšné změně hesla zase přepíše na Null.

Výstupem je soubor `collecting_problems.py`, ve kterém vkládám data do tabulek *Solutions* a *Problems*.

Tabulky jsem tvořila v souboru `problem_database.ipynb`, protože mi stačilo je vytvořit jednou. Zde je ukázka vytváření tabulky *Assignments*.

```
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="ekatka",
    password="",
    db="dailyAMC"
)
mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE Assignments(id INT NOT NULL AUTO_INCREMENT,"
                 "problem_date DATE, "
                 "problem_id INT, "
                 "PRIMARY KEY (id), "
                 "FOREIGN KEY (problem_id) REFERENCES Problems(id));")
```

Obdobně jsem vytvořila zbylé tabulky.

2.2 Tvorba stránky

Pro tvorbu stránky jsem se rozhodla využít FastAPI spolu s Jinja2, protože mi připadalo, že ze všechframeworků mají nejlepší dokumentaci. Začala jsem generováním stránky s úlohou. Tu si beru z databáze a pomocí knihovny MathJax zobrazuji LaTeXové výrazy. Když uživatel odpoví, tak mu v závislosti na správnosti jeho odpovědi zobrazím další stránku, kde mu řeknu, zda odpověděl správně, a ukáži odkaz na stránku s řešením. Kód stránky se nachází ve složce templates v souboru `header.html`.

Když je uživatel přihlášen, tak uložím jeho odpověď do databáze, takže mu mohu vypsat jeho statistiky, počet správně zodpovězených úloh v řadě, počet správně zodpovězených úloh a statistiky podle dní.

Když uživatel vytvoří nový účet, uložím si jeho email a hash hesla do databáze. Generování hashe a ověřování zadaného hesla provádím pomocí knihovny passlib. Po registraci nebo přihlášení uživateli vygeneruji token, ten rovnou ukládám do cookie, která se po půlhodině smaže. Pokud uživatel při přihlášení zaškrtne políčko remember me, tak se cookie smaže po 7 dnech. Pro generaci tokenu využívám knihovny z fastapi.security. Projekt jsem dělala podle instrukcí na stránce `https://fastapi.tiangolo.com/tutorial/security/`.

Vždy, když někdo otevře stránku, tak kontroluji jestli je přihlášený, a když je, tak kontroluji, jestli daný den už odpověděl. Pokud ano, tak ho přesměruji na stránku s jeho odpovědí.

Web jsem dělala hlavně pro počítač, ale po menším předělání mám i mobilní verzi, která mi jako jediná část projektu zabrala méně času, než jsem předpokládala. Počítačová verze by měla vypadat dobře ve všech prohlížečích. Mobilní verze mi fungovala pouze v Chromu a Firefoxu, například ve Vivaldi se nějaká pole přesouvají přes sebe, stránka pak jde používat, ale nevypadá tak dobře.

```
{% block content %}

        <p class="tex2jax >{{ problem }}</p>
        <div class="button-container">
                {% for solution in solutions %}
                {% if solution == correct %}
                        <label class="button"
                         style="background-color: #20C34E">
                {% elif solution == answer %}
                        <label class="button"
                         style="background-color: red">
                {% else %}
                        <label class="button"
                        style="height: 3em; padding: 1em">
                {% endif %}
                <input type="radio"
                 name="answer" value="{{ solution }}">
                <p style=" font-size: 2em">
                ${{ solution }}$</p>

        </div>
{% endfor %}
```

Zde je upravená ukázka toho, jak používám Jinja2 v templatu wrong.html, který se zobrazí, když uživatel odpoví špatně. Syntaxe je podobná jako v Pythonu, nejdříve vložím samotnou úlohu, pak dostanu seznam řešení solutions, podle něj pak iteruji. Když se iterované řešení schoduje s odpovědí uživatele, tak jeho pozadí bude červené, když se shoduje se správnou odpovědí bude pozadí zelené. Odpovědi se zobrazí jako checkbox, ale nyní už nejdou zaškrtnout.

Všechny statistiky beru z tabulky *Answers*, do které ukládám odpovědi pouze přihlášených uživatelů, protože, kdybych hodnotila i nepřihlášené, tak by někteří uživaté mohli odpovídat vícekrát, což by zkreslilo statistiku úloh.

2.3 Spouštění webu a resetování hesla

Docker kontejner jsem vytvořila z daného vzoru `https://github.com/qlawmarq/fastapi-mysql-docker`. Nyní mám dva docker soubory: docker-compose.yml a docker-compose-full.yml. Ve druhém je přidaný kontejner na nginx. V souboru requirements.txt je seznam všech potřebných knihoven, které by pak měl při pouštění instalovat docker.

Při pouštění jsem často dostávala chyby, že konterjner fastapi byl připraven, zatímco připojení k databázi ještě neproběhlo, to jsem vyřešila funkcí `wait-for-mysql.sh`, která čeká na to, až proběhně

připojení k databázi.

Bohužel se mi nepodařilo nastavit SSL certifikát, takže spojení není zabezpečeno.

Přidala jsem také funkci resetování hesla. Uživatel zadá svůj mail, tím mu vygeneruji token a uložím ho do tabulky. Pomocí knihovny email mu pošlu mail s odkazem obsahujícím token. Uživatel na daném odkazu vyplní nové heslo, vygeneruji mu nový hash, který uložím do tabulky a smažu token v tabulce.

```
def send_email(user, link):
    try:
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.ehlo()
        server.starttls()
        server.login(os.environ['EMAIL_USER'], os.environ['EMAIL_PASSWORD'])
        msg = MIMEMultipart()
        msg['From'] = os.environ['EMAIL_USER']
        msg['To'] = user
        msg['Subject'] = "Password reset link"
        body = f"Please click on the link to reset your password: {link}"
        msg.attach(MIMEText(body, 'plain'))
        server.sendmail(os.environ['EMAIL_USER'], user, msg.as_string())
        server.quit()
        return True
    except Exception as e:
        print(f"Error: {e}")
        return False
```

Zde je moje funkce na posílání mailu. Pomocí knihovny smtplib vytvořím SMTP spojení. Přihlásím se pomocí proměnných prostředí EMAIL_USER a EMAIL_PASSWORD. Postupovala jsem podle návodu na této stránce `https://realpython.com/python-send-email/`.

3. Technická dokumentace

3.1 Instalace

Je potřeba mít nainstalovaný docker. Po klonovaní databáze, se ve složce DailyAMC musí spustit `docker-compose up`. Tím se stáhnou všechny potřebné knihovny a stránka by měla běžet lokálně. Pro spuštění je potřeba zadat proměnné prostředí EMAIL_USER a EMAIL_PASSWORD, aby šlo odeslat mail. Připadně zakomentovat řádky, kde kontroluju, zda tyto proměnné existujou. Příkaz `docker-compose -f docker-compose-full.yml` up spustí i nginx. Stránka se nachází na odkazu `dailyamc.xyz`.

3.2 Používání

Uživateli se o půlnoci času UTC zobrazí nová matematická úloha, zvolí si jednu z pěti odpovědí, pak klikne na tlačítko Submit. Tím se jeho odpověď vyhodnotí, a když je přihlášen, tak se uloží do databáze.

Nahoře je lišta, Home přeposílá na hlavní stránku, což je stránka s úlohou. About jsou informace o projektu. Login otevře stránku s přihlášením a registrací. Nový uživatel se musí nejdříve zaregistrovat, když už má uživatel účet, stačí se pouze přihlásit. Když uživatel chce zůstat přihlášen delší dobu může zakliknout tlačítko remember me a zůstane přihlášen 7 dní. Nyní když odpoví na denní úlohu, tak se jeho odpověď uloží. Pak se mu ukáže streak a počet zodpovězených úloh podle dní a informace o tom, kolik lidí zodpovědelo na dnešní úlohu a kolik odpovědí bylo správně.

Přihlášenému uživateli se místo Login na horní liště zobrazuje Logout. Když se chce uživatel odhlásit musí nejdřív kliknout na Logout, pak se mu zobrazí nová stránka, kde odhlášení musí ještě jednou potvrdit. Nechala jsem to tak záměrně, aby se nestalo, že se někdo odhlásí omylem.

Pokoušela jsem se nastavit posílání tokenu na resetování hesla, což se mi nepodařilo spustit na serveru přes docker. Nyní když uživatel zapomene heslo, tak mi může poslat email a vyřeším to osobně.

Závěr

Myslím si, že zadání je splněno, ale některé věci bych příště udělala jinak. Místo nginxu bych použila caddy, který by automaticky vyřešil HTTPS. Také bych se pak víc podívala na dokumentaci, protože Jinja2 má několik funkcí, které by mi ulehčily práci a zpřehlednily kód, ale dozvěděla jsem se o nich až po té, co jsem udělala vše, co bylo potřeba. Další nedostatek vidím v přehlednosti kódu a pojmenování funkcí a názvů tabulek.

Na začátku jsem ukládala úlohy na počítač do složek, což jsem později předělala na databázi. Myslím si, že jsem se výrazně zlepšila v práci s databázemi a po té, co jsem musela přepisovat kód v pěti místech po té, co jsem v originálu udělala chybu, se budu snažit víc dodržovat DRY.

Velký problém jsem měla s implementováním stránky, s HTML a CSS nemám moc zkušeností a kód podle toho vypadá, řekla bych ale, že na výsledku to není tolik poznat. Další možný problém vidím v bezpečnosti a nejen v bezpečnosti spojení. Na ukládání přihlašovacího tokenu používám cookies. Příště bych se rozhodla pro bezpečnější metodu.

S výsledkem práce jsem i přes výše vyjmenované nedostatky spokojená. Ke konci jsme měla lepší časový odhad na to, jak dlouho mi bude trvat udělat nějaké části a byla jsem si jistější v tom, co dělám.

Seznam použité literatury

Nixon, R. (2018). Learning Php, MySQL Et JavaScript: With jQuery, Css Et Html 5. O'Reilly.

Perkins, J. (2011). Python text processing with Ntlk 3.0 Cookbook: Lite: Over 80 practical recipes for using Python's Nltk suite of libraries to maximize your natural language processing capabilities. PACKT Publishing.

Seznam obrázků

2.1 Ukázka z tabulky Problems 5