SŠPU STŘEDNÍ ŠKOLA PRŮMYSLOVÁ A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

Revas - Aplikace pro jednoduché udělání testů

Autor: Patrik Tomaštík
Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na programování
Třída: IT4
Školní rok: 2024/25

Obsah

Poděkování

Děkuji panu učiteli Mgr. Markovi Lucnemu a Ing. Petru Grussmanovi za rady při vytváření tohoto projektu.

Prohlášení

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým a prezentačním účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 1. 1. 2025

. . . . . .

Podpis autora

ÚVOD

Představení projektu

Revas je aplikace, která umožňuje vytváření testů a kvízů, ale i zábavných herních testů, které kombinují otázky a odpovědi s herními prvky. Tento nástroj je navržen tak, aby byl uživatelsky přívětivý a snadno použitelný, a to jak pro učitele, studenty, tak pro širokou veřejnost, která chce spojit zábavu a učení.

Motivace

Revas vznikl z potřeby zjednodušit proces tvorby testů a zároveň přidat do učení prvek zábavy. Motivací byla frustrace z tradičních a nudných způsobů hodnocení a učení. Chtěl jsem vytvořit nástroj, který by učitelům a lektorům umožnil snadno vytvářet personalizované testy, které zároveň zaujmou studenty. Spojení herních prvků, jako je Pexeso, s otázkami a odpovědmi, přináší nový, interaktivní způsob učení, který je motivující a zároveň efektivní. Takový přístup pomáhá zvýšit angažovanost, což je klíčové pro lepší zapamatování a pozitivní vztah k učení.

Cíle projektu

Hlavním cílem této práce je vytvořit jednoduchou apliakci pro dělání testů, které přináši i prvky zábavy.

Struktura práce

Práce je rozdělana na první část, kde je ukázána logika pro dělání testů a v druhé části je ukázána vzhled testů.

1 TEORETICKÁ ČÁST

Testování je důležitým nástrojem nejen v oblasti vzdělávání, ale i v psychologii, medicíně a dalších oblastech, kde je třeba kvantifikovat a analyzovat různé schopnosti a stavy. Testy mohou mít formu otázek s výběrem odpovědí, esejí, úkolů nebo praktických zkoušek, přičemž volba formy závisí na cíli testování a povaze zkoumané oblasti.

1.1 Historie a vznik testování

Testování, jak ho známe dnes, má dlouhou historii, která sahá až do starověkých civilizací. Již ve starověkém Egyptě, Číně a Řecku se používaly formy testů pro výběr jednotlivců na různé pozice, například pro vojenské nebo vládní služby. V průběhu staletí se metody hodnocení vyvíjely, ale teprve na přelomu 19. a 20. století došlo k modernizaci testování díky rozvoji psychologie.

1.1.1 Počátky moderního testování

Za počátky moderního testování je považován vývoj psychometrie, tedy vědy o měření duševních schopností a charakteristik. Na konci 19. století vytvořil francouzský psycholog Alfred Binet první testy inteligence. Binetův test, známý jako Binet-Simon test, byl zaměřen na identifikaci dětí, které měly potíže se školním učením, a byl základem pro vznik dnešních IQ testů.

1.1.2 Rozvoj testování ve 20. století

V průběhu 20. století se testování stalo klíčovým nástrojem nejen ve vzdělávání, ale také v pracovních, vojenských a psychologických výzkumech. V roce 1916 byl ve Spojených státech vyvinut Stanford-Binet test, který rozšířil původní Binetův test a poskytl lepší nástroje pro měření inteligence. Tento test byl použit pro širokou škálu účelů, od školních zkoušek po výběr do armády.

Ve 20. letech 20. století začaly vznikat také první standardizované testy pro hodnocení výkonu studentů v širším měřítku. Testy byly zaměřeny na akademické dovednosti a vědomosti, například matematiku, čtení nebo gramatiku. V roce 1926 byl vyvinut test Scholastic Aptitude Test (SAT), který se stal jedním z nejznámějších a nejpoužívanějších nástrojů pro hodnocení výkonu studentů v USA.

1.2 Druhy testů

V současnosti existuje mnoho různých typů testů, které se liší jak formou, tak účelem. Některé z nejběžnějších typů testů zahrnují:

- **Testy s výběrem odpovědí**: Nejrozšířenější formou testování. Studenti vybírají jednu nebo více odpovědí z předem daných možností.
- **Esejové testy**: Testy, které vyžadují, aby studenti napsali odpověď v textové formě. Tyto testy hodnotí schopnost kritického myšlení a organizování informací.
- **Praktické testy**: Používají se k hodnocení praktických dovedností, například při výuce technických nebo uměleckých oborů.
- **Standardizované testy**: Tyto testy mají pevně stanovený formát a jsou používány k měření výkonu studentů ve srovnání s širokou populací. Příkladem jsou SAT, ACT nebo GRE.
- **Formativní testy**: Cílem těchto testů není pouze hodnocení, ale i podpora učení. Jsou používány v průběhu vzdělávacího procesu, aby učitelé zjistili, jaký pokrok studenti dělají a kde je potřeba zlepšení.

1.3 Význam testování ve vzdělávání

Testování má v současném vzdělávacím systému mnoho funkcí. Mezi hlavní patří:

- **Hodnocení pokroku studentů**: Testy umožňují měřit, jak dobře studenti zvládli učivo a jaký pokrok udělali během studia.
- **Identifikace potřeb studentů**: Testy mohou pomoci identifikovat studenty, kteří potřebují další podporu nebo kteří se nacházejí nad průměrem.
- **Motivace k učení**: Testování může sloužit jako motivace pro studenty, kteří se chtějí zlepšit nebo dokázat své schopnosti.
- **Standardizace hodnocení**: Standardizované testy poskytují objektivní způsob hodnocení, který umožňuje porovnávat výkon studentů na širší úrovni.
- **Zpětná vazba pro učitele**: Testy poskytují učitelům důležité informace o tom, jak efektivně vyučují a které oblasti je třeba zlepšit.

1.4 Současné trendy v testování

V současnosti se testování stále vyvíjí a přizpůsobuje novým technologiím a pedagogickým metodám. Mezi současné trendy patří:

- **Online testování**: S nástupem digitálních technologií se testování přesunulo do online prostředí, což umožňuje flexibilitu a dostupnost testů pro širokou veřejnost.

- **Gamifikace testů**: V některých případech je testování spojeno s herními prvky, což má za cíl zvýšit motivaci a angažovanost studentů.
- **Testování na bázi analýzy dat**: Využití pokročilé analytiky a algoritmů k hodnocení výkonu studentů a přizpůsobení testů na míru.
- **Automatizované hodnocení**: Pokroky v umělé inteligenci umožnily automatizaci hodnocení testů, což zrychluje proces a zajišťuje větší objektivitu.

Z tabulky je patrné, že každá aplikace má své silné a slabé stránky. Zatímco Anki vyniká v pokročilých funkcích a statistikách, postrádá moderní uživatelské rozhraní a gamifikační prvky. Quizlet nabízí dobrou rovnováhu mezi funkcionalitou a použitelností, ale mnoho pokročilých funkcí je dostupných pouze v placené verzi. Memrise má propracovaný systém gamifikace, ale je omezen především na jazykové učení a vyžaduje stálé připojení k internetu.

2 POUŽITÉ TECHNOLOGIE

2.1 P5.js

P5.js je knihovna pro JavaScript, která usnadňuje tvorbu interaktivní grafiky a multimediálních aplikací. Používá se například pro vytváření vizualizací nebo interaktivních komponent, které mohou být součástí chatových aplikací nebo testových modulů, kde jsou uživatelské interakce součástí designu.

Hlavní vlastnosti:

- Snadná tvorba interaktivních grafických a multimediálních aplikací
- Podpora animací a dynamických vizualizací
- Flexibilita pro integraci s webovými aplikacemi

2.2 Socket.IO (Live Chat)

Pro implementaci live chatu byla zvolena knihovna Socket.IO, která umožňuje real-time komunikaci mezi serverem a klientem pomocí WebSocketů. Tato technologie je ideální pro chatové aplikace, které vyžadují okamžitou interakci mezi uživateli.

Výhody:

- Real-time komunikace mezi klientem a serverem
- Možnost odesílání a přijímání zpráv bez zbytečného čekání
- Podpora pro různé typy zpráv (textové, obrázky, soubory)
- Integrace s dalšími systémy pro správu uživatelských účtů a chatových místností

2.3 Moodle XML Generator

Pro přeměňování textových formátů do formátu Moodle XML byla použita specializovaná technologie, která umožňuje automatizovaně generovat testy, otázky a odpovědi pro platformu Moodle. Tento proces je nezbytný pro snadné publikování a správu online testů.

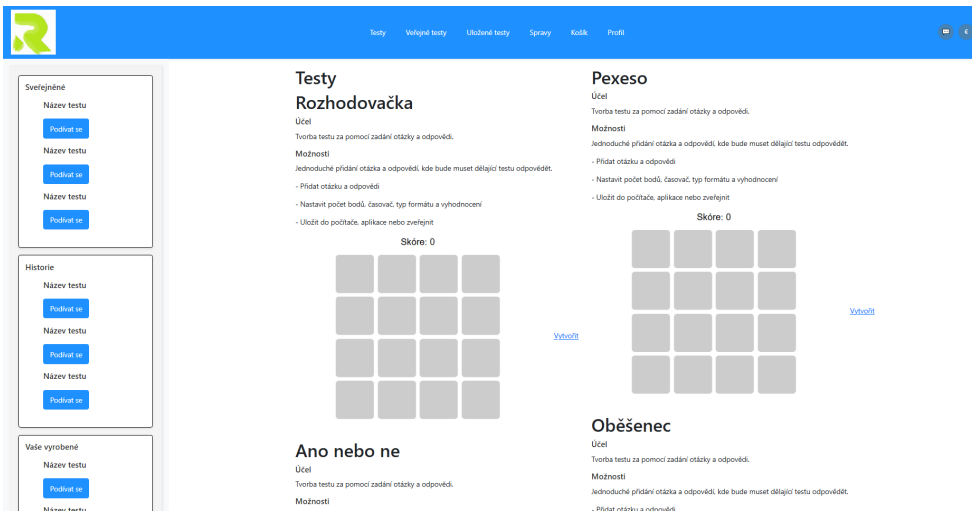
Klíčové výhody:

- Automatizované generování testových otázek a odpovědí
- Podpora pro různé typy otázek (výběr z více možností, textové odpovědi, přiřazování)
- Efektivní správa exportovaných testů pro platformu Moodle

3 ARCHITEKTURA APLIKACE

3.1 Celková architektura

Aplikaci jsem postavil na Django frontend, i když není primárně frontendový framework. Django umožňuje: templates, static files, forms.



Hlavní architektonické principy:

- Lepší testovatelnost jednotlivých komponent
- Snadnější údržba a rozšiřitelnost kódu
- Jasná separace zodpovědností
- Škálovatelnost aplikace

3.2 Datový model

Jako backand jsem využil Django.

```
// Registrace
class RegistrationForm(forms.Form):
    username = forms.CharField(max_length=100, required=True,
    label="Uživatelské jméno")
    first_name = forms.CharField(max_length=100, required=True,
    label="Jméno")
    last_name = forms.CharField(max_length=100, required=True,
    label="Příjmení")
    email = forms.EmailField(required=True, label="Email")
    password = forms.CharField(widget=forms.PasswordInput,
```

```
    min_length=8, required=True, label="Heslo")
    confirm_password = forms.CharField(widget=forms.PasswordInput,
    required=True, label="Potvrďte heslo")
    age = forms.IntegerField(min_value=18, required=True,
    label="Věk")
    gender_choices = [('M', 'Muži'), ('F', 'eny')]
    gender = forms.ChoiceField(choices=gender_choices, required=True,
    label="Pohlaví")
    profile_picture = forms.ImageField(required=False,
    label="Profilový obrázek")

    def clean_email(self):
        email = self.cleaned_data.get('email')
        if User.objects.filter(email=email).exists():
            raise ValidationError("Tento email je již registrován.")
        return email

    def clean_password(self):
        password = self.cleaned_data.get('password')
        if not re.match(r"^(?=.*[A-Z])(?=.*\d).{8,}$", password):
            raise ValidationError("Heslo musí mít alespoň 8 znaků,
            jedno velké písmeno a jednu číslici.")
        return password

    def clean(self):
        cleaned_data = super().clean()
        password = cleaned_data.get("password")
        confirm_password = cleaned_data.get("confirm_password")

        if password != confirm_password:
            raise ValidationError("Hesla se neshodují.")

        return cleaned_data

```

4 IMPLEMENTACE

4.1 Frontend implementace

4.1.1 Uživatelské rozhraní

Vzhled uživatelského rozhraní je velmi důležitý, protože dává první dojmy z aplikace:

Podoba herních testů, Stránky aplikace

4.2 Backend implementace

Backend aplikace jsem postavil na Djangu v aplikaci Pycharm.

- Databáze SQL (Ukládání testů a uživatelů)
- Moodle XML Generator
- Realtime chat

4.3 Implementace Testů

Implementace přidání otázky

```
function addQuestion() {
    questionCount++;
    const questionHTML = `
        <div class="question-wrapper" id="question${questionCount}">
            <label>Otázka ${questionCount}:</label>
            <input type="text" name="question${questionCount}_text"
            placeholder="Zadejte otázku" required>
            <div class="error-message" id="errorQuestion${questionCount}"
            style="display: none;">Tato otázka musí mít text.</div>

            <label>Body:</label>
            <input type="number" name="question${questionCount}_points"
            min="1" value="1">

            <div class="answers" id="answers${questionCount}">
                <label>
                    <input type="checkbox"
                    name="question${questionCount}_correct_answer"
```

```
                    value="1">
                    <input type="text" placeholder="Odpověď 1" required>
                    <button type="button"
                    onclick="removeAnswer(${questionCount}, 1)">
                    Smazat odpověď</button>
                </label>
            </div>

            <div class="error-message" id="errorAnswer${questionCount}"
            style="display: none;">Musíte označit správnou odpověď.</div>

            <button type="button" onclick="addAnswer(${questionCount})">
            Přidat odpověď</button>
            <button type="button" onclick="deleteQuestion(${questionCount})">
            Smazat otázku</button>
        </div>
    `;
    document.getElementById('questionsContainer').
    insertAdjacentHTML('beforeend', questionHTML);
}

```

4.4 Pro ukládání otázek

Implementace pro uložení otázky.

```
// Uložení otázky
document.getElementById("saveQuestionBtn").addEventListener("click",
function() {
            const questionText =
            document.getElementById("questionInput").value;
            const answers = [
                { text: document.getElementById("answer1").value, correct:
                document.getElementById("correct1").checked },
                { text: document.getElementById("answer2").value, correct:
                document.getElementById("correct2").checked },
                { text: document.getElementById("answer3").value, correct:
                document.getElementById("correct3").checked },
```

```
                { text: document.getElementById("answer4").value, correct:
                document.getElementById("correct4").checked }
    ];

```

4.5 Náhled testů

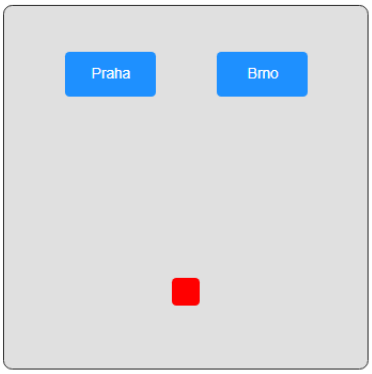
Implementace offline režimu je založena na lokální SQLite databázi.

Klíčové aspekty offline funkcionality:

Implementace pro náhled testů.

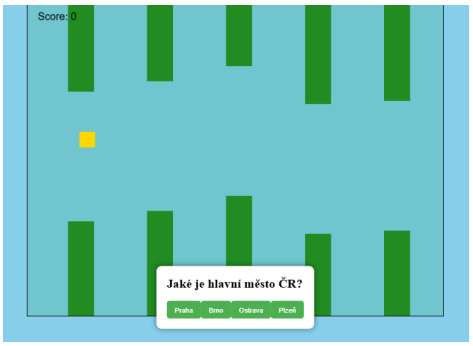
```
// Funkce pro náhled testu
    function previewTest() {
        const testContent = document.getElementById('testContent');
        const testName = document.getElementById('testName').value;
        testContent.innerHTML = `<h2>${testName}</h2>`;

        const questions = document.querySelectorAll('.question-wrapper');
        questions.forEach(question => {
        const questionText =
        question.querySelector('input[name^="question"]
        [name$="_text"]').value;
        const options = [];
        question.querySelectorAll('input[name$="_option1"],
        input[name$="_option2"], input[name$="_option3"],
        input[name$="_option4"]').forEach(option => {
            options.push(option.value);
            });
            testContent.innerHTML += `
                <div class="test-question">
                    <p>${questionText}</p>
                    <ul>
                        <li>${options[0]}</li>
                        <li>${options[1]}</li>
                        <li>${options[2]}</li>
                        <li>${options[3]}</li>
                    </ul>
```

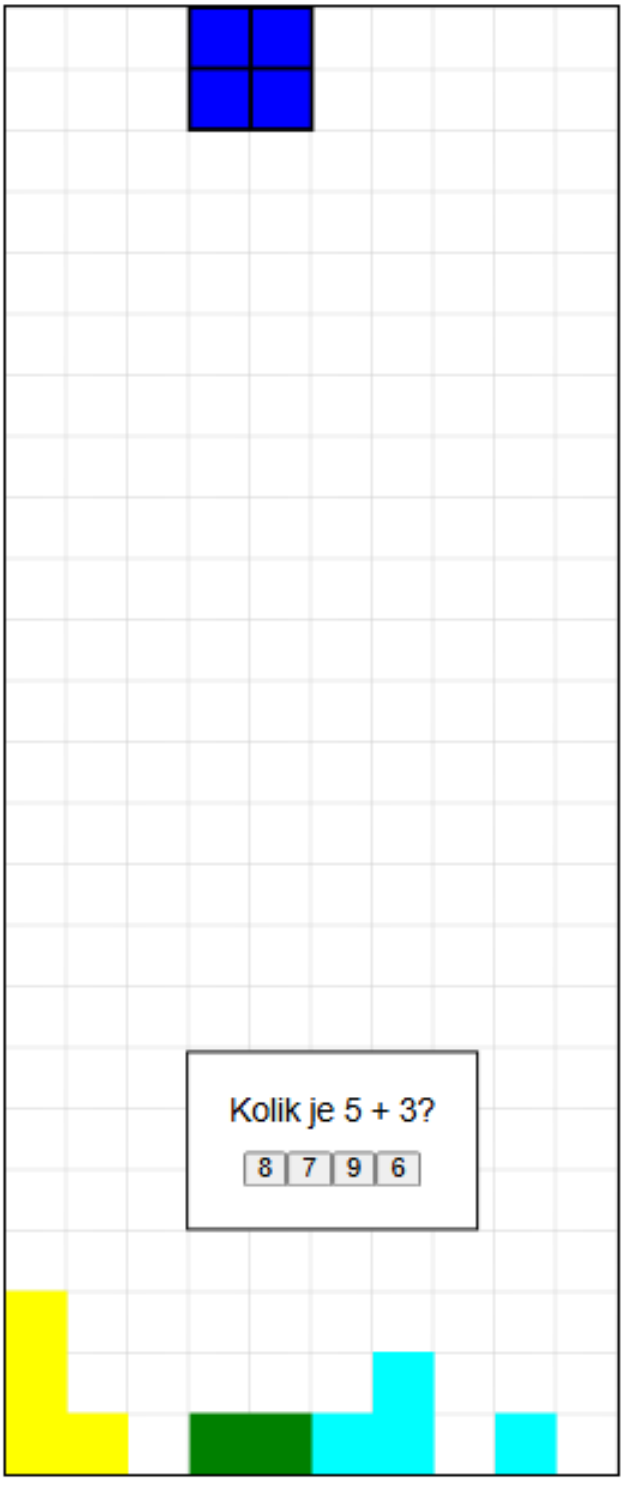
```
                </div>
            `;
        });

        document.getElementById('previewTest').style.display = 'block';
    }

```



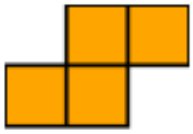
Hlavní město ČR?

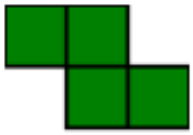
Čas: 27s





Next:





Score: 0

Test: Spojte správné páry

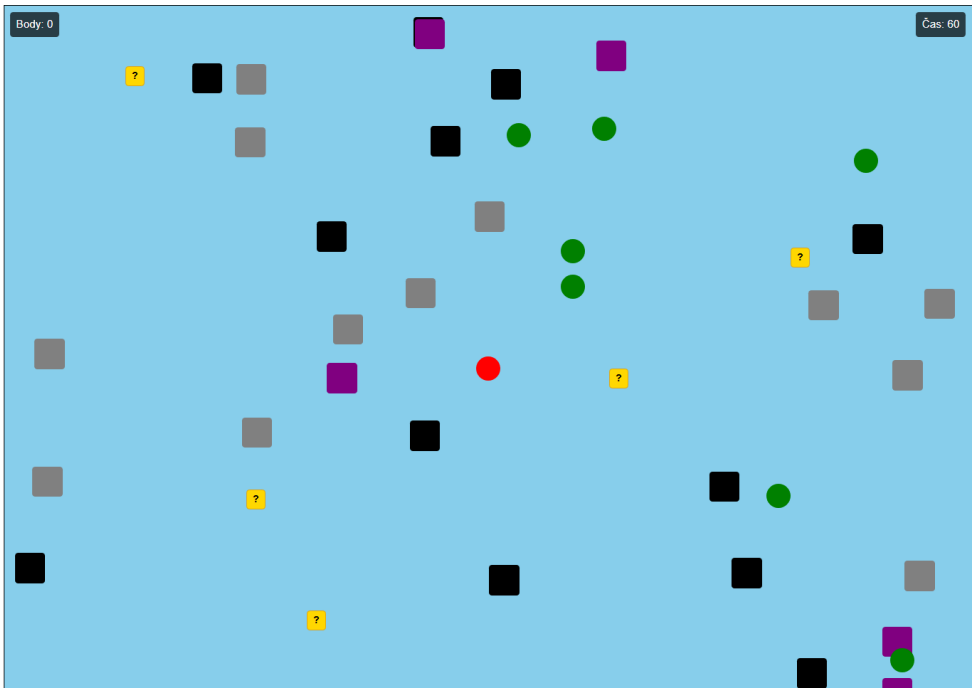
Hlavní město Německa?	Berlín
Hlavní město ČR?	Praha
Hlavní město Slovenska?	Bratislava

Odeslat odpovědi

Jaké je nebe?

Modré | Žluté | Černé | Jak to mám vědě, nahoru nekoukám

Další otázka



5 VÝSLEDKY

5.1 Splněné cíle

V rámci projektu se mi podařilo úspěšně vytvořit aplikaci pro dělání testů a k tomu udělat live chat s možností nákupu vymožeností.

5.2 Nedostatky a budoucí zlepšení

- Přidání více testů
- Přidání větší volnosti při vytváření testů
- Lépe si zorganizovat času
- Přidání dalších testů v podobě kahootu a nebo denních výzev

5.3 Přínosy

Projekt Revas úspěšně dosáhl svého cíle. Je možné vytvořit jednoduché a hravé testy.

Hlavní přínosy projektu:

- Efektivnější dělání testů.
- Zábavu
- Zkušenosti jak zacházet s časem a plánování
- Zkušenosti do budoucna

6 ZÁVĚR

Aplikace Revas představuje moderní nástroj, který výrazně usnadňuje tvorbu a správu testů v různých formách. Díky flexibilnímu přístupu a intuitivnímu uživatelskému rozhraní nabízí uživatelům širokou škálu možností pro tvorbu jak statických, tak dynamických testů, ať už ve formě otázek a odpovědí nebo gamifikovaných testů, které mohou být zábavným a efektivním způsobem učení.

Zdrojový kód projektu je na GitHubu (`https://github.com/Patrik1T/Maturitniprojekt?tab=readme-ov-file#classquiz`).

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] Quiz Flask App by simboli: Open-source webová aplikace postavená na Pythonu a Flasku s MySQL databází, umožňuje vytváření a správu kvízů. Dostupné z: https://github.com/thepasterover/flask-quiz-app

[2] Quiz Flask App Open-source webová aplikace postavená na Pythonu a Flasku s MySQL databází, umožňuje vytváření a správu kvízů. Dostupné z: https://github.com/simboli/quiz-flask-app

[3] ClassQuiz: Open-source aplikace podobná Kahoot!, umožňuje učitelům vytvářet kvízy, které mohou studenti hrát na dálku. Dostupné z: https://github.com/mawoka-myblock/ClassQuiz

[4]Moosh: Nástroj pro správu Moodle z příkazové řádky, který může pomoci při exportu testů do Moodle. Dostupné z: https://github.com/tmuras/moosh