ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра алгебри, геометрії та диференціальних рівнянь

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «магістр»

«Розробка автоматизованої системи контролю освітнього процесу»

(тема кваліфікаційної роботи українською мовою)

«Development of the automate system of educational process»

(тема кваліфікаційної роботи англійською мовою)

Виконала: здобувачка денної форми навчання

спеціальності 123 – Комп'ютерна інженерія

(код, назва спеціальності)

Освітня програма _____

(назва)

Тіхонова Тетяна Андріївна

(прізвище, ім'я, по-батькові здобувача)

Керівник д. ф.-м. н., доц. Варбанець С.П.

(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент доц. Савастру О.В.

(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри

_____

№ ___ від ____.____. 20___ р.

Завідувач(ка) кафедри

_____ Євтухов В.М.

(підпис) (прізвище,ім'я)

Захищено на засіданні ЕК № _____
протокол № __від ____.____.20___ р.

Оцінка_____/_____/_____

(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

_____ _____

(підпис) (прізвище, ім'я)

Одеса 2023

АНОТАЦІЯ

Даний проект спрямований на розробку та впровадження автоматизованої системи для моніторингу та контролю освітнього процесу.

Мета даного дослідження полягає у розробці та впровадженні стратегій для зменшення ймовірності вгадування відповідей на тестові запитання. Зокрема, планується вивчення та впровадження нової структури тестових завдань, що ускладнюють випадкове визначення правильних відповідей. Очікується, що ці заходи сприятимуть покращенню точності тестування та значно зменшать можливість випадкових відповідей, що сприятиме більш об'єктивному вимірюванню рівня знань тестируваних осіб.

Система включатиме функції для відстеження прогресу студентів, управління академічними ресурсами та формування подальшої стратегії навчання. Автоматизуючи ключові аспекти освітнього процесу, ця ініціатива прагне оптимізувати адміністративні завдання, надавати своєчасну інформацію щодо академічної успішності та сприяти загальному покращенню якості освіти.

Проект відповідає зростаючому попиту на інноваційні рішення у сфері освіти та підкреслює важливість використання технологій для оптимізації управління освітнім процесом.

ANNOTATION

This project aims to develop and implement an automated system for monitoring and controlling the educational process.

The research goal is to devise and implement strategies to reduce the likelihood of guessing answers to test questions. Specifically, there are plans to explore and implement a new structure for test tasks that complicates the random determination of correct answers. It is anticipated that these measures will enhance testing accuracy and significantly decrease the possibility of random responses, thereby fostering a more objective measurement of the knowledge level of the test subjects.

The system will encompass features for tracking student progress, managing academic resources, and shaping a comprehensive teaching strategy. By automating key aspects of the educational process, this initiative seeks to optimize administrative tasks, provide timely information on academic performance, and contribute to an overall improvement in the quality of education.

The project aligns with the increasing demand for innovative solutions in the field of education and underscores the importance of utilizing technology to optimize educational management processes.

ЗМІСТ

АНОТАЦІЯ 2

ANNOTATION 3

ЗМІСТ 4

ВСТУП 6

1 МЕТА ТА ФУНКЦІЇ КОНТРОЛЮ ЗА ОСВІТНІМ ПРОЦЕСОМ 7

1.1 Мета контролю 7

1.2 Тестовий метод оцінювання 7

2 АНАЛІЗ ВІДОМИХ ПРОГРАМНИХ ПРОДУКТІВ 9

2.1 Kahoot! 9

2.2 Proprofs 9

2.3 Surveymonkey 10

2.4 Google-форми 10

2.5 Moodle 10

3 МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ 12

3.1 Створення гетерогенного тесту 12

3.1.1 Розрахунок вірогідностей вгадування різних видів тестів: 14

3.1.2 Розрахунок вірогідності вгадування 75% запитань 15

3.1.2 Аналіз результатів тестування 15

3.1.3 Створення подальшої стратегії навчального плану 16

4 ОЧІКУВАНІ РЕЗУЛЬТАТИ ВІД АВТОМАТИЗОВАНОЇ СИСТЕМИ КОНТРОЛЮ ЗА ОСВІТНІМ ПРОЦЕСОМ 17

4.1 PhpStorm 17

4.2 PostgreSQL 17
4.3 Nicepage 18
5 ПОСТАНОВКА ЗАДАЧ ІНФОРМАЦІЙНОЇ СИСТЕМИ 19
6 ПРОЕКТУВАННЯ ІНФОРМАЦИОНОЇ СИСТЕМИ 23
7 ІНФОРМАЦІЙНЕ МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 25
8 СТВОРЕННЯ БАЗИ ДАНИХ 33
9 ЗАПИТИ ДО БАЗИ ДАНИХ ДЛЯ ВИРІШЕННЯ ЗАДАЧ 36
10 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ 38
11 БЕЗПЕКА ІНФОРМАЦІЙНОЇ СИСТЕМИ 45
12 ІНСТРУКЦІЯ КОРИСТУВАЧА 48
ВИСНОВКИ 57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 58
ДОДАТОК А 59

ВСТУП

Важливим фактором систематизації навчального процесу є контроль засвоєння матеріалу курса. Неможливо зробити цей процес керованим та ефективним без налагодження послідовної перевірки знань та навичок. Саме тому є дуже відповідальним етап створення системи контролю знань, призначений для оцінки рівня оволодіння необхідними вміннями та знаннями, успішності навчання, виявлення труднощів та прогалин знань учнів та корекції стратегії викладання матеріалу.

При проектуванні системи контролю за освітнім процессом необхідно обрати методи та форми контролю знань, які відповідають потребам викладача та сприяють ефективному, оперативному та об’єктивному оцінюванню учнів. Для досягнення цієї мети з’являється потреба в створенні автоматизованого методу оцінювання учнів.

Результатом цієї кваліфікаційної роботи повинно бути програмне забезпечення, яке виконує задачі контролю за освітнім процесом та прогнозування подальшого плану надання матеріалу.

1 МЕТА ТА ФУНКЦІЇ КОНТРОЛЮ ЗА ОСВІТНІМ ПРОЦЕСОМ

1.1 Мета контролю

Головна мета контролю освітнього процесу — забезпечення його ефективності приведенням до системи знань, умінь, навичок учнів, самостійного застосування здобутих знань на практиці, стимулювання навчальної діяльності учнів, формування у них прагнення до самоосвіти. [1]

1.2 Тестовий метод оцінювання

У тестовій перевірці знайшла широке застосування тестова методика з альтернативним вибором відповідей. Правильно сформовані тестування мають структуру, еталон та ціну завдання. [2]

Переваги тестування:

- можливість кількісного вимірювання рівня знань і труднощі завдань;
- об’єктивність і порівнянність оцінки та цілковите охоплення знань;
- можливість автоматизації перевірки.
- Переваги тестування:
- можливість угадування;
- відносна складність створення якісного тесту;
- помилки вимірів.

Мінімальні вимоги до складу тестового завдання базуються на трьох важливих частинах. Інструкція. Текст завдання (запитання). Правильні відповіді (еталон).

За конструкцією завдань тести поділяють на:

1) закриті (з вибірковою відповіддю), коли учень вибирає з переліку поданих варіантів відповідь , яку він вважає правильною:
   - Альтернативний тест важається найпростіший у розв’язанні. У ньому висунуте завдання передбачає 4-5 варіантів відповідей, серед яких лише одна – правильна. При цьому чим більше

варіантів відповідей на завдання, тим менша можливість вгадування.

- Вибірковий або варіативний тест зазвичай не викликає у опитуваних особливих труднощів. Передбачає 8-10 варіантів відповідей на тестове завдання, з яких 4-5 відповідей правильні. За вибірковими тестами перевіряють передусім цілісність і повноту знань класифікації того чи іншого явища.
- Послідовний або порядковий тест дає можливість перевірити знання послідовності тієї чи іншої події. У варіантах відповіді на таке тестове завдання відсутні неправильні відповіді, адже запропоновані у невпорядкованому вигляді поняття, слова, ситуації, дати необхідно розташувати у правильній послідовності.
- Схематичний тест подає схему побудови якогось об'єкта без наявних підписів із проставленими цифрами 1, 2, 3 тощо, а під літерами а, б, в тощо, можливі бути визначені ключові слова. Опитуваний повинен спів віднести вказані ознаки з їх означеннями. [3]

2) відкриті (з конструйованою відповіддю), що вимагають від учня самостійної формулювання відповіді:
   - на доповнення (тести-підстановки);
   - із стислою відповіддю;
   - конструктивні тести;
   - типові задачі.

Тести за однорідністю тестових завдань бувають:

- гомогенні (однорідні), що включають тестові завдання однієї конструкції;
- гетерогенні (неоднорідні), що мають тестові завдання різної конструкції. [4]

2 АНАЛІЗ ВІДОМИХ ПРОГРАМНИХ ПРОДУКТІВ

У ході пошуку та аналізу відомих програмних продуктів були виділені наступні чотири додатки, що мають функціонал наближений до того, що повинен вирішувати проблеми висунуті для рішення у даному дипломному проекті.

2.1 Kahoot!

Kahoot! це безкоштовний онлайн-інструмент для створення тестів, який використовує барвисті візуальні елементи та елементи гейміфікації, щоб максимізувати залучення та забезпечити вищий відсоток проходження учнями. Тести креативно називаються «kahoots», доступ до яких можна отримати через веб-браузер або мобільний додаток. За допомогою kahoot maker ви також можете організувати самостійне завдання, яке ваша команда виконує асинхронно.

Особливості:

3) Барвисті та гейміфіковані вікторини
4) Організація ігор для 3-10 гравців
5) Самостійний виклик [5]

2.2 Proprofs

Це програма, яка поєднує в собі все, що потрібно вчителю для повноцінної праці. Її основні переваги:

1) Простий та зрозумілий інтерфейс.
2) Кожен з користувачів може створити онлайн-тест з відео- та аудіо-матеріалами.
3) В системі є багато вже готових тем для оформлення тестів. При необхідності можна створити щось своє.
4) Тест може бути яким завгодно. Наприклад, з декількома відповідями на питання, серед яких потрібно вибрати одну правильну. Чи потрібно вставити слово, яке пропустили.
5) Програма також працює на смартфонах та планшетах.

6) До кожного питання можна залишати коментарі. [6]

2.3 Surveymonkey

Доволі цікавий сайт для створення тестів. Його головними особливостями є:

1) Можливість створити тест з текстом або зображеннями.
2) Тести можна створювати за допомогою готових шаблонів. Також їх зручно настроювати.
3) Система видає результаті одразу у декількох форматах, наприклад, PDF чи XLS.
4) Підрахунок результатів, їх порівняння із зазначеними параметрами проходять в автоматичному режимі. [7]

2.4 Google-форми

Ця програма призначена для проведення онлайн-досліджень, створення тестів (квізів), голосувань, вікторин та ін. На відміну від інших подібних сервісів, вона має декілька переваг:

1) Простота у використанні та зрозумілий інтерфейс.
2) Доступність на протязі всієї доби.
3) Можливість оформити сторінку згідно зі своїми вимогами. [8]

2.5 Moodle

Moodle — це безкоштовна онлайн-система керування навчанням, яка дозволяє викладачам створювати власний приватний веб-сайт, наповнений динамічними курсами, які продовжують навчання в будь-який час і в будь-якому місці.

1) Сучасний, простий у використанні інтерфейс
2) Персоналізована сторінка курсу
3) Інструменти та дії для співпраці
4) Зручне керування файлами
5) Прості та інтуїтивно зрозумілі текстові редактори
6) Відстеження прогресу [9]

Порівняльна характеристика існуючих аналогів програмного забезпечення за функціональними ознаками описаними вище наведена у Таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристики існуючих аналогів програмного забезпечення для перевірки тестувань та аналізу даних

Функції	Kahoot!	Proprofs	Survey monkey	Google-форми	Moodle	Розроблена ситема
Створення тесту	+	+	+	+	+	+
Гетерогенні тести	-	+	-	+	+	+
Автоматична перевірка тесту	+	+	+	+	+	+
Додання зображень	+	+	+	+	+	-
Створення бланків	-	+	+	+	+	+
Створення груп опитуваних	+	+	+	+	+	+
Обмеження за часом	+	+	-	+	+	+
Завдання балів кожному питанню	-	-	-	+	+	+
Аналіз результатів	-	-	-	-	+	+
Запобігання вгадування відповідей	-	-	-	-	-	+

3 МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Об'єктом дослідження є контроль за освітнім процесом.

Предметом дослідження є тестувальний метод контролю за освітнім процесом.

Метою роботи є створення автоматизованої системи автоматизованого контролю за освітнім процессом для забезпечення оперативного та об'єктивного оцінювання досягнень учнів.

Для досягнення даної мети необхідно виконати наступні задачі:

- створення гетерогенного тесту, вірогідність вгадування в якому не перевищує одного відсотка;
- забезпечення можливості надання медіа-, аудіо- та відеофайлів до тесту чи окремих питань;
- забезпечення можливості надання балів за кожне питання;
- забезпечення можливості обмеження часу на виконання тесту;
- забезпечення можливості створення груп опитуваних;
- автоматичне оцінювання результату тесту
- аналіз результатів тестування;
- створення подальшої стратегії навчального плану на основі результатів тестування.

3.1 Створення гетерогенного тесту

Проектуємий тест має складатися з двох частин.

Перша частина складається з питань, які є закритими за своєю конструкцією, а саме:

- альтернативний тест;
- вибірковий або варіативний тест;
- послідовний або порядковий тест;
- схематичний тест.

При створенні питань тестів викладач повинен мати можливість створити різні за конструкцією завдання. Один тест може містити декілька видів запитань. Такий тест буде вважатися гетерогенним.

Друга частина складається з питань, які є відкриті за своєю конструкцією:

- із стислою відповіддю;
- типові задачі.

Друга частина теж може містити декілька видів запитань. Питання в ній можуть мати тільки одну правильну відповідь. Вона повинна бути чітка та не двояка. Для питань із стислою відповіддю існує суттєвий недолік: якщо учень надасть правильну відповідь, але зробить в ній орфографічну помилку, вона не буде зарахована. Для вирішення цієї проблеми можна використовувати автоматичне виправлення тексту, але це може дати учневі підказку чи ускладнити формулювання відповіді для викладача.

Тест з двох частин необхідний для зменшення вірогідності вгадування відповідей. Закриті тести можуть бути легко автоматизовані, на відміну від відкритих через описані вишче причини. Тому учень буде проходити другу частину тільки, якщо відповів вірно на недостатню кількість запитань чи якщо викладач зробить її необхідною. Вчитель має можливість відмовитися від відкритих питань чи зробити їх обов’язковими.

В сумі кількість балів за всі запитання першої частини повинна дорівнюватися максимальній оцінці за тест. Якщо учень вирішує другу частину його оцінка перераховується, ураховуючи додаткові бали.

Перш за все вчитель має вказати теми, за якими проводиться опитування. Для кожної з них необхідно скласти мінімум п’ять закритих запитань з якнайменше п’яттю варіантами відповідей для альтернативного тесту, 2 та 3 варіантами для схематичного тесту, трьома варіантами для ранжування та 4 варіантами для вибіркового тесту. Якщо учень відповів правильно на 75% запитань з кожної теми тесту, він не потребує складати другу частину тесту.

3.1.1 Розрахунок вірогідностей вгадування різних видів тестів:

1) Альтернативный тест

Якщо кількість варіантів відповіді дорівнює 5, то вірогідність вгадати правильну дорівнює 0.2.

$$P = \frac{1}{5} = 0.2 \quad (3.1.1.1)$$

2) Послідовний тест

Відповідь враховується, якщо послідовність повністю правильна. Якщо кількість варіантів відповіді дорівнює 3, то вірогідність вгадати правильний порядок дорівнює приблизно 0.1667.

$$P = \frac{1}{3!} = \frac{1}{6} = 0.1667 \quad (3.1.1.2)$$

3) Вибірковий тест

Відповідь враховується, якщо обрані всі правильні відповіді і жодної неправильної. Серед варіантів повинно бути від двох правильних відповідей та від однієї неправильної. Якщо кількість варіантів відповіді дорівнює 4, то вірогідність вгадати правильну дорівнює 0.1.

$$P = \frac{1}{C_4^2 + C_4^3} = \frac{1}{6+4} = 0.1 \quad (3.1.1.3)$$

$$C_4^2 = \frac{4!}{(4-2)!2!} = \frac{24}{4} = 6 \quad (3.1.1.4)$$

$$C_4^3 = \frac{4!}{(4-3)!3!} = \frac{24}{6} = 4 \quad (3.1.1.5)$$

4) Схематичний тест

Відповідь враховується, якщо всі відповідності правильні. Повинен бути один варіант, який немає відповідності. Якщо кількість варіантів відповіді дорівнює 2 з однієї сторони та 3 з іншої, то вірогідність вгадати правильну дорівнює 0.1.

$$P = P1 * P2 = \frac{1}{3} * \frac{1}{3-1} = \frac{1}{6} = 0.1667 \quad (3.1.1.5)$$

$P1$ – вірогідність правильно вгадати перше співвідношення

$P2$ – вірогідність правильно вгадати друге співвідношення

3.1.2 Розрахунок вірогідності вгадування 75% запитань

Якщо тест складається з 5 альтернативних питань, вірогідність вгадати які найбільша, вірогідність вгадати щонайменше 75% запитань складає 0.672%. Цей відсоток розраховується для кожної теми тесту.

Для успішного складання першої частини тесту учень має відповісти правильно на 4 чи 5 запитань (80-100% правильних відповідей).

$$P = C_5^4 0.2^4 0.8^1 + C_5^5 0.2^5 = 5 * 0.00128 + 0.00032 = 0.0067 \quad (3.1.2.1)$$

Чим більше тест буде містити більш запитань, варіантів відповідей чи типів питань, тим меншою буде вірогідність вгадування.

У другій частині вірогідність вгадати правильну відповідь наближається до 0, бо варіанти відповідей не обмежуються. Друга частина складається з питань, які відносяться до теми, з якої учень не набрав балів.

3.1.2 Аналіз результатів тестування

Аналіз результатів тестування потрібен для кращого розуміння проблем в викладанні та засвоєнні матеріалу, формулюванні запитань та причин виникнення складнощів у учнів.

В аналізі результатів відображаються наступні показники:

- коефіцієнт правильних та неправильних відповідей на запитання серед групи опитуваних та їх упорядкування;
- коефіцієнт обрання для кожного варіанту відповіді;
- коефіцієнт засвоєння тем матеріалу та їх упорядкування;
- коефіцієнт правильних та неправильних відповідей у учня серед усіх питань теста;
- коефіцієнт правильних та неправильних відповідей у учня серед усіх питань окремої теми;
- коефіцієнт засвоєння теми у учня;
- складання рейтингів студентів за пройденим матеріалом, окремим тестом та окремими темами.

3.1.3 Створення подальшої стратегії навчального плану

Система повинна мати наступну інформацію:

- Теми курсу та кількість годин, яка на них відведена
- Розклад занять та проведення тестів
- Кількість годин на консультації

Маючи цю інформацію, можна розрахувати яку найбільшу кількість годин можна виділити на додаткові консультації на кожну тему. Після проведення тесту можна розрахувати скільки часу необхідно на консультації, враховуючи коефіцієнт серед тих, хто присутній на консультації, після цього треба перерахувати максимальну кількість годин на консультації для лишившихся тем. Якщо всі учні засвоїли матеріал більш, ніж на 85%, консультація не проводиться.

4 ОЧІКУВАНІ РЕЗУЛЬТАТИ ВІД АВТОМАТИЗОВАНОЇ СИСТЕМИ КОНТРОЛЮ ЗА ОСВІТНІМ ПРОЦЕСОМ

Результатом цієї кваліфікаційної роботи повинно бути програмне забезпечення, яке виконує задачі контролю за освітнім процесом та прогнозування подальшого плану надання матеріалу.

Для розробки ІС обрані наступні програми:

1) мова програмування PHP, мови розмітки HTML, мови стилю CSS;
2) інтегроване середовище розробки PhpStorm;
3) СУБД PostgreSQL;
4) програмне забезпечення для створення веб-сайтів Nicepage.

4.1 PhpStorm

PhpStorm — це інтегроване середовище розробки (IDE) для розробників PHP. Настільна програма IDE допомагає писати, редагувати, аналізувати, рефакторювати, тестувати та налагоджувати код PHP у Windows, macOS та Linux.

PhpStorm має наступні переваги:

1) Розумний редактор коду PHP.
2) Аналіз якості коду.
3) Легка навігація по коду & Пошук.
4) Налагодження, тестування та профілювання.
5) Редактор HTML і CSS.
6) Редактор JavaScript.
7) Нові технології.
8) VCS.
9) Бази даних & SQL. [10]

4.2 PostgreSQL

Для зберігання інформації про курси, їх матеріали та теми для викладання, розклад занять, тестів та консультацій буде використовуватися СУБД PostgreSQL. На сьогоднішній день існує багато СУБД таких, як MySQL,

Oracle, MariaDB та інші. PostgreSQL вигідно відрізняється від багатьох інших СУБД. Вона володіє практично всіма можливостями, які є в інших базах даних (комерційних або Open Source), а також деякими додатковими:

1) Транзакції
2) Вкладені запити
3) Уявлення
4) Посилальна цілісність - зовнішні ключі
5) Типи, визначені користувачем
6) Спадковість. [11]

4.3 Nicepage

Програмне забезпечення для створення веб-сайтів з можливістю перетягування та розміщення, що дозволяє втілити будь-яку ідею. В Nicepage великий вибір з широкого спектру готових шаблонів. Nicepage може створювати майже будь-який сучасний веб-дизайн. Nicepage була створена після ретельного аналізу тисяч і тисяч веб-дизайнів на Pinterest та Dribbble. Вона зосереджена переважно на найновіших тенденціях у веб-дизайні. [12]

5 ПОСТАНОВКА ЗАДАЧ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Задачі, які вирішуються інформаційною системою:

1) Внесення тем матеріалу курсу, коригування часу на консультації;
2) Створення розкладу занять, консультацій та тестувань;
3) Створення списків учнів, їх додавання та видалення;
4) Фіксація результатів тестування;
5) Створення списків груп та співробітників;
6) Створення, видалення та редагування тестів та варіантів відповідей вчителем та його складання учнем.

Категорії користувачів, які працюють з ІС:

1) Директор - має право на додавання, редагування і видалення інформації про себе, перегляд, внесення та видалення тем матеріалу курсу, створення розкладу занять, консультацій та тестувань, списків учнів, груп та співробітників, їх додавання та видалення.
2) Вчитель - має право на додавання, редагування і видалення інформації про себе, перегляд тем матеріалу курсу та корегування часу на консультації, створення, видалення та редагування тестів та варіантів відповідей, фіксація результатів тестування.
3) Учень – має право на додавання, редагування і видалення інформації про себе, перегляд розкладу занять, консультацій та тестувань, складання тестів та перегляд результатів.

Задачі користувачів, які вирішуються за допомогою ІС, приведені в таблиці 1.1.

Таблиця 1.1 – Задачі користувачів

Номер	Задачі	Вхідні дані	Вихідні дані
	Директор		
Д1	Додавання тем	Назва теми, курс	Нова тема
Д2	Редагування тем	Змінні дані	Оновлена інформація про тему

Продовження таблиці 1.1

Номер	Задачі	Вхідні дані	Вихідні дані
Д3	Видалення тем	Тема, яка видаляється	Відсутні
Д4	Додавання занять	Назва курсу, тривалість, група, викладач, час, статус	Нове заняття
Д5	Редагування занять	Змінні дані	Оновлена інформація про заняття
Д6	Видалення занять	Заняття, яке видаляється	Відсутні
Д7	Додавання консультації	Назва курсу, тривалість, група, викладач, час, статус	Нова консультація
Д8	Редагування консультацій	Змінні дані	Оновлена інформація про консультацію
Д9	Видалення консультації	Консультація, яка видаляється	Відсутні
Д10	Додавання співробітника	Прізвище, ім'я, по батькові, посада, зарплата, дата народження, логін, пароль	Новий тест
Д11	Редагування співробітника	Змінні дані	Оновлена інформація про співробітника
Д12	Видалення співробітника	Співробітник, який видаляється	Відсутні

Продовження таблиці 1.1

Номер	Номер	Номер	Номер
Д13	Додавання групи	Назва, рік навчання, вчитель	Новий група
Д14	Редагування групи	Змінні дані	Оновлена інформація про групу
Д15	Видалення групи	Група, який видаляється	Відсутні
Д16	Додавання учня	Прізвище, ім'я, по батькові, група, дата народження, логін, пароль	Новий учень
Д17	Редагування учня	Змінні дані	Оновлена інформація про учня
Д18	Видалення учня	Учень, який видаляється	Відсутні
	Вчитель		
В1	Додавання питань тестів	Текст питання, тест, тема	Новий тест
В2	Редагування питань тестів	Змінні дані	Оновлена інформація про питання
В3	Видалення питань тестів	Питання, яка видаляється	Відсутні
В4	Додавання відповідей	Текст відповіді, тест, тема, питання	Нова відповідь
В5	Редагування відповідей	Змінні дані	Оновлена інформація про відповідь
В6	Видалення відповідей	Відповідь, яка видаляється	Відсутні

Продовження таблиці 1.1

Номер	Номер	Номер	Номер
В7	Додавання тесту	Тест, учень, оцінка	Новий тест
В8	Редагування тесту	Змінні дані	Оновлена інформація про тест
В9	Видалення тесту	Тест, яка видаляється	Відсутні
В10	Порахувати відсоток відповідених питань	Перша частина контрольної роботи	Оцінка
В11	Порахувати бал за контрольну роботу	Контрольна робота	Відсоток
	Учень		
У1	Складання тестів	Тест, питання, відповідь, учень	Складенний тест

6 ПРОЕКТУВАННЯ ІНФОРМАЦИОНОЇ СИСТЕМИ

Для реалізації даного проекту використовується двухрівнева архітектура клієнт-сервер (рис. 6.1). В основі клієнт-серверної архітектури лежать два компоненти: клієнт і сервер.

Клієнт – комп'ютер на стороні користувача, який відправляє запит до сервера для надання інформації або виконання певних дій.

Сервер – більш потужний комп'ютер або обладнання, призначене для вирішення певних завдань з виконання програмних кодів, виконання сервісних функцій за запитом клієнтів, надання користувачам доступу до певних ресурсів, зберігання інформації і баз даних.

Модель такої системи полягає в тому, що клієнт відправляє запит на сервер, де він обробляється, і готовий результат відправляється клієнтові. Сервер може обслуговувати кілька клієнтів одночасно. Якщо одночасно приходить більше одного запиту, то вони встановлюються в чергу і виконуються сервером послідовно. Іноді запити можуть мати пріоритети. Запити з більш високими пріоритетами повинні виконуватися раніше.

Дворівнева архітектура складається з двох вузлів:

- сервер, який відповідає за отримання запитів і відправку відповідей клієнту, використовуючи при цьому лише власні ресурси;
- клієнт, який представляє користувацький інтерфейс.

Принцип роботи полягає в тому, що сервер отримує запит, обробляє його і відповідає безпосередньо, без використання сторонніх ресурсів. [13]

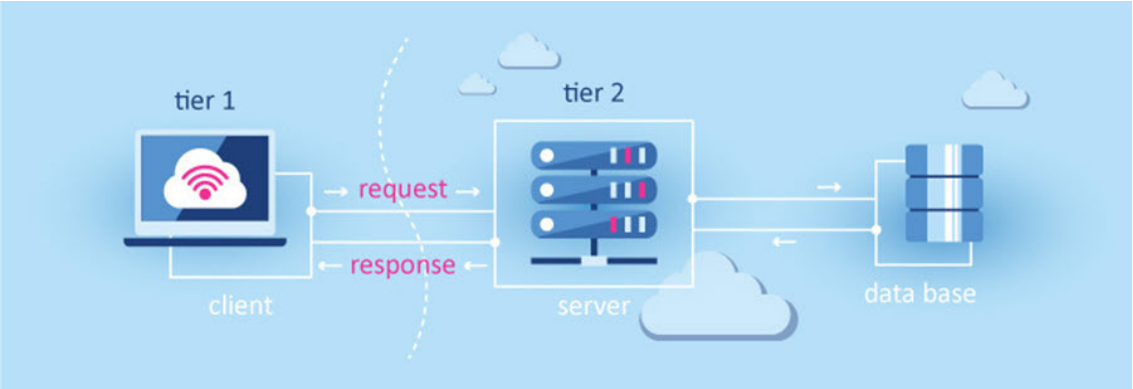

Рисунок 6.1 – Двухрівнева архітектура клієнт-сервер

Як шаблон використовується MVC (Model-View-Controller - Модель-Представлення - Контролер), див. рис. 6.2. Це архітектурна схема, яка складається з трьох компонентів Model, View та Controller, що ефективно відокремлює Business Logic від користувальницького інтерфейсу програми.

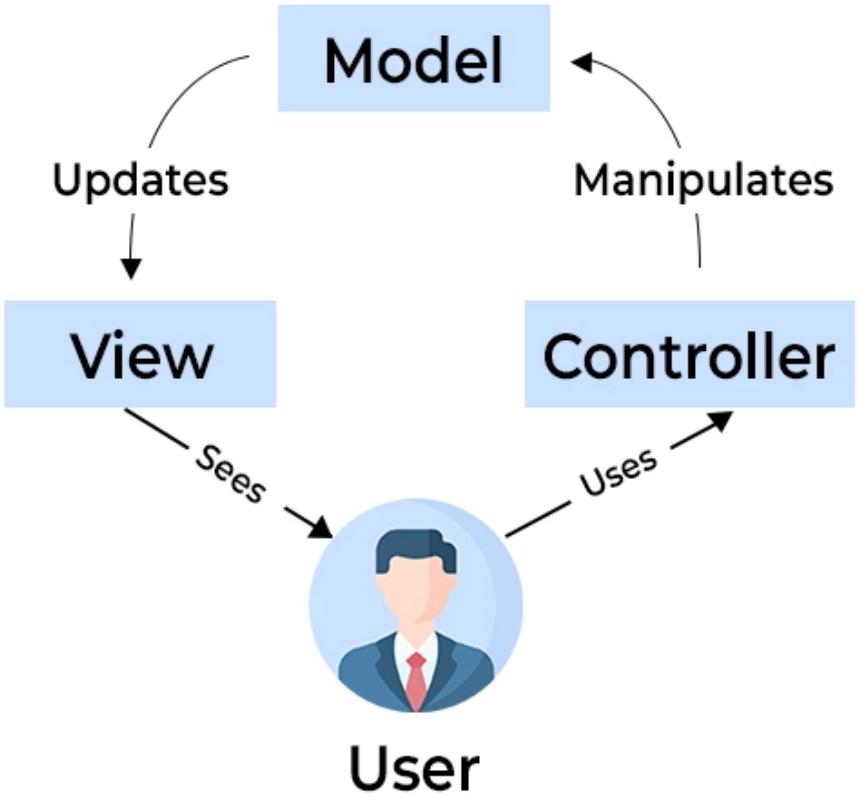

Рисунок 6.2 – Шаблон MVC

Концепція MVC розділяє дані, представлення та обробку дій користувача на три компоненти:

4) **Модель** містить дані про програму. Тут вказується вся інформація, яка має бути важливою для відображення або відображення, її вимоги щодо доступу та інших перевірок.
5) **Перегляд** відображає дані в компоненті Модель. Будь-яка відповідь від користувача також розпізнається та надсилається до компонента Controller.
6) **Контролер** відповідає за надання даних, присутніх у Моделі, компоненту «Вид» та інтерпретацію відповідей користувачів, які розпізнаються компонентом «Перегляд». [14]

7 ІНФОРМАЦІЙНЕ МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

Опис сутностей для вирішення завдань ІС, а також їх атрибутів і обмежень цілісності представлені в таблиці 7.1.

Таблиця 7.1 – Опис сутностей ІС

Властивість	Опис	Обмеження
Сутність «Answer»		
answer_id	Ідентифікатор відповіді	Первинний ключ, не порожнє
question	Питання	Зовнішній ключ с таблицею Question, не порожнє
letter	Буква	До 1 символів, не порожнє
content	Текст відповіді	До 255 символів, не порожнє
status	Правильність відповіді	Значення так чи ні
index	Номер підпитання	Цілісне
score	Бал	Не порожнє, цілісне
Сутність «Chosen_answer»		
chosen_answer_id	Ідентифікатор відповіді учня	Первинний ключ, не порожнє
answer	Варіант відповіді	Зовнішній ключ с таблицею Answer, не порожнє
test_result_part	Контрольна робота	Зовнішній ключ с таблицею Test_result_part, не порожнє
index	Номер підпитання	Цілісне
Сутність «Consultation»		
consultation_id	Ідентифікатор консультації	Первинний ключ, не порожнє

Продовження таблиці 7.1

Властивість	Опис	Обмеження
teacher_group_course	Вчитель, група та курс	Зовнішній ключ с таблицею Teacher_group_course, не порожнє
date_time	Дата та час консультації	Дата та час, не порожне
Сутність «Course»		
course_id	Ідентифікатор курсу	Первинний ключ, не порожнє
course_name	Ім’я курсу	До 40 символів, не порожнє
description	Опис курсу	До 255 символів, не порожнє
Сутність «Director»		
director_id	Ідентифікатор адміністратора	Первинний ключ, не порожнє
first_name	Ім’я адміністратора	До 40 символів, не порожнє
last_name	Фамілія адміністратора	До 40 символів, не порожнє
user	Номер користувача	Зовнішній ключ с таблицею User, не порожнє
Сутність «Exam»		
examt_id	Ідентифікатор екзамену	Первинний ключ, не порожнє
test	Тест	Зовнішній ключ с таблицею Test, не порожнє
date_time	Дата та час екзамену	Дата та час, не порожне
teacher_group_course	Вчитель, група та курс	Зовнішній ключ с таблицею Teacher_group_course, не порожнє
Сутність «Group»		
group_id	Ідентифікатор групи	Первинний ключ, не порожнє

Продовження таблиці 7.1

Властивість	Опис	Обмеження
group_name	Назва групи	Не порожнє, до 20 символів
curator	Вчитель	Зовнішній ключ с таблицею Teacher, не порожнє
Сутність «Lesson»		
lesson_id	Ідентифікатор уроку	Первинний ключ, не порожнє
teacher_group_course	Вчитель, група та курс	Зовнішній ключ с таблицею Teacher_group_course, не порожнє
date_time	Дата та час уроку	Дата та час, не порожне
Сутність «Lesson_consultation_duration»		
lesson_consultation_ duration_id	Ідентифікатор	Первинний ключ, не порожнє
duration	Тривалість уроків та консультацій	Інтервал, не порожнє
Сутність «Matrix_question_li»		
matrix_question_li_id	Ідентифікатор підпитання	Первинний ключ, не порожнє
index	Номер підпитання	Не порожне, цілісне
content	Текст підпитання	Не порожнє, до 1024 символів
question	Питання	Зовнішній ключ с таблицею Question, не порожнє
Сутність «Module»		
module_id	Ідентифікатор теми	Первинний ключ, не порожнє
course	Курс	Зовнішній ключ с таблицею Course, не порожнє

Продовження таблиці 7.1

Властивість	Опис	Обмеження
name	Назва теми	Не порожнє, до 80 символів
Сутність «Open_answer»		
open_answer_id	Ідентифікатор відкритої відповіді учня	Первинний ключ, не порожнє
question	Питання	Зовнішній ключ с таблицею Question, не порожнє
test_result_part	Контрольна робота	Зовнішній ключ с таблицею Test_result_part, не порожнє
open_answer	Текст відповіді	Не порожнє, до 255 символів
Сутність «Question»		
question_id	Ідентифікатор питання	Первинний ключ, не порожнє
content	Текст питання	Не порожнє, до 1024 символів
question_type	Тип питання	До 6 символів
closed_question_type	Тип закритого питання	До 12 символів
open_answer	Відповідь на відкрите питання	До 255 символів
score	Бал	Цілісне
module	Тема	Зовнішній ключ с таблицею Module, не порожнє
Сутність «Question_test_part»		
question_test_part_id	Ідентифікатор	Первинний ключ, не порожнє
question	Питання	Зовнішній ключ с таблицею Question, не порожнє

Продовження таблиці 7.1

Властивість	Опис	Обмеження
test_part	Частина тесту	Зовнішній ключ с таблицею Test_part, не порожнє
Сутність «Student»		
director_id	Ідентифікатор студента	Первинний ключ, не порожнє
first_name	Ім’я студента	До 40 символів, не порожнє
last_name	Фамілія студента	До 40 символів, не порожнє
user	Номер користувача	Зовнішній ключ с таблицею User, не порожнє
group	Група	Зовнішній ключ с таблицею Group, не порожнє
Сутність «Teacher»		
teacher_id	Ідентифікатор вчителя	Первинний ключ, не порожнє
first_name	Ім’я вчителя	До 40 символів, не порожнє
last_name	Фамілія вчителя	До 40 символів, не порожнє
user	Номер користувача	Зовнішній ключ с таблицею User, не порожнє
group	Група	Зовнішній ключ с таблицею Group, не порожнє
Сутність «Teacher_group_course»		
teacher_group_ course_id	Ідентифікатор	Первинний ключ, не порожнє
teacher	Вчитель	Зовнішній ключ с таблицею Teacher, не порожнє

Продовження таблиці 7.1

Властивість	Опис	Обмеження
group	Группа	Зовнішній ключ с таблицею Group, не порожнє
course	Курс	Зовнішній ключ с таблицею Course, не порожнє
Сутність «Test»		
test_id	Ідентифікатор тесту	Первинний ключ, не порожнє
name	Назва тесту	До 40 символів, не порожнє
description	Опис тесту	До 255 символів, не порожнє
time_limit	Обмеження за часом	Цілісне
parts_type	Кількість частин	До 20 символів, не порожнє
course	Курс	Зовнішній ключ с таблицею Course, не порожнє
Сутність «Test_part»		
test_part_id	Ідентифікатор частини тесту	Первинний ключ, не порожнє
index	Номер частини	До 6 символів, не порожнє
test	Тест	Зовнішній ключ с таблицею Test, не порожнє
Сутність «Test_result»		
test_result_id	Ідентифікатор контрольної роботи	Первинний ключ, не порожнє
test	Тест	Зовнішній ключ с таблицею Test, не порожнє
student	Студент	Зовнішній ключ с таблицею Student, не порожнє

Продовження таблиці 7.1

Властивість	Опис	Обмеження
mark	Бал	Цілісне
exam	Екзамен	Зовнішній ключ с таблицею Exam, не порожнє
Сутність «Test_result_part»		
test_result_part_id	Ідентифікатор частини контрольної роботи	Первинний ключ, не порожнє
test_part	Частина тесту	Зовнішній ключ с таблицею Test_part, не порожнє
test_result	Контрольна робота	Зовнішній ключ с таблицею Test_result, не порожнє
mark	Бал	Цілісне
Сутність «User»		
user_id	Ідентифікатор користувача	Первинний ключ, не порожнє
login	Логін	До 20 символів, не порожнє, унікальне
password	Пароль	До 12 символів, не порожнє

У всіх зовнішніх ключів присутня умова заборони на його видалення, якщо існують дочірні записи (on delete restrict), окрім question_id, у якого всі дочірні елементи видаляються з ним (on update cascade), і умова зміни відповідного атрибута у записів при зміні батьківського значення (on update cascade).

Фундаментальний вид зв'язку «один-до-одного» реалізований тільки у таблицях Teacher і User, Student і User та Director і User. Для формалізації цього типу зв'язку первинний ключ user однозв'язної сутності User додається до схеми однозв'язних сутностей Teacher, Student та Director як зовнішній ключ.

Фундаментальний вид зв'язку «багато-до-багатьох» реалізований тільки у таблицях Course, Teacher та Group, а ще у таблицях Question та Test_part. Для формалізації цього типу зв'язку були створені таблиці Course_teacher_group та Question_test_part відповідно, які містять зовнішні ключі course, teacher та group у першій, та question і test_part – у другій.

Зв'язки між іншими таблицями є зв'язками типу «один-до-багатьох» (або до 0).

Одним із засобів графічного представлення предметної області є діаграма «сутність-зв'язок» (ER-діаграмма). Для її побудови біла використана програма Dbdiagram. ER-діаграма приведена на малюнку 7.1.

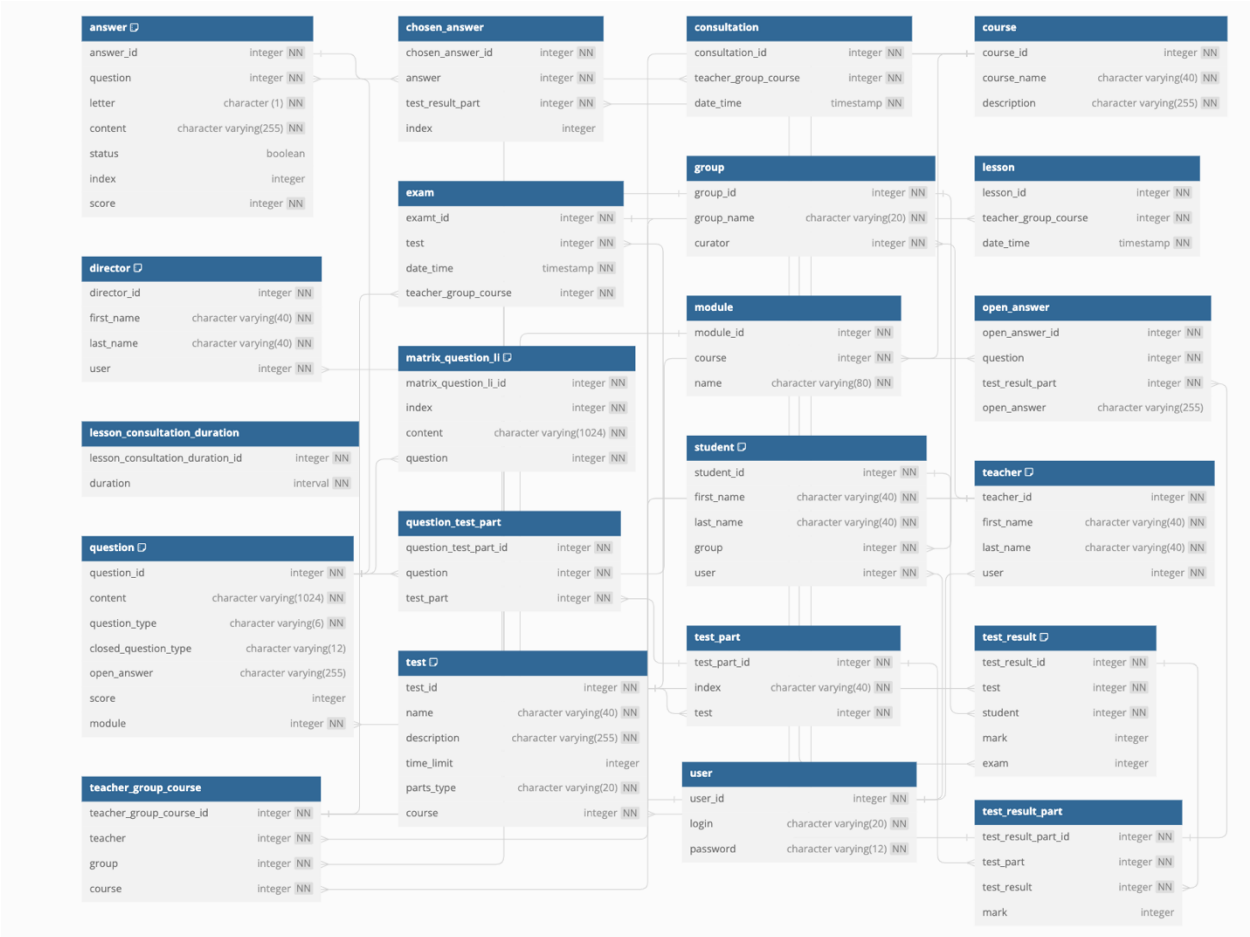

Рисунок 7.1 – ER-діаграма бази даних

8 СТВОРЕННЯ БАЗИ ДАНИХ

Було створено 22 таблиці, 12 збережених процедур та 14 тригерних функцій. Приклад створення таблиці наведений в лістингу 8.1.

Для створення таблиці використовуємо ключове слово create table, далі перераховуємо поля таблиці, біля них вказуємо їх типи, умову not null, обмеження на поля за допомогою ключового слова check, створюємо первинний, унікальні та зовнішні ключі (primary, unique, foreign key).

```
create table question(
     question_id serial not null primary key,
     content varchar(1024) not null,
     question_type varchar(6) not null check(question_type
in('closed', 'open')),
     closed_question_type varchar(12) check(closed_question_type
in('one answer', 'some answers', 'matrix', 'range')),
     score int,
     module int not null,
     foreign key (test_part_module) references test_part_module
(test_part_module_id) on delete restrict on update cascade
);
```

Лістинг 8.1 – Створення таблиці

Приклад створення збереженої процедури наведений в лістингу 8.2.

Збережену процедуру створюємо за допомогою ключового слова CREATE FUNCTION, у дужках передаємо вхідні дані, за допомогою слова RETURNS задаємо тип результату процедури, задаємо змінні за допомогою слова DECLARE, починаємо та закінчуємо за допомогою слів BEGIN та END;

```
CREATE OR REPLACE FUNCTION answer_test()
RETURNS TRIGGER
AS $$
DECLARE
     answer_question integer;
     answer_question_type integer;
    true_answer_count integer;
CREATE OR REPLACE FUNCTION answer_test()
RETURNS TRIGGER
AS $$
DECLARE
     answer_question integer;
     answer_closed_question_type varchar(12);
    true_answer_count integer;
BEGIN
```

```
SELECT question
     INTO answer_question
     FROM answer
     WHERE answer_id=NEW.answer_id;

SELECT closed_question_type
     INTO answer_closed_question_type
     FROM question
     WHERE question_id=NEW.question;

SELECT COUNT(*)
     INTO true_answer_count
     FROM answer
     WHERE question = answer_question and status=true;

     IF (answer_closed_question_type='one answer') THEN
          IF (true_answer_count!=1) THEN
               RAISE EXCEPTION 'Wrong true answers number';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     ELSIF (answer_closed_question_type='some answers') THEN
          IF (true_answer_count<2) THEN
               RAISE EXCEPTION 'Wrong true answers number';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Лістинг 8.2 – Створення збереженої процедури

Приклад створення тригерної функції наведений в лістингу 8.3.

Створення тригерної функції схоже на створення збереженої процедури, тригер можна створити за допомогою ключового слова CREATE TRIGGER , ключові слова AFTER INSERT OR UPDATE вказують, що тригерна функція застосовується після додавання або зміни даних, ON – вказує для якої таблиці працює тригер, FOR EACH ROW – що тригер працює для кожного рядка, EXECUTE PROCEDURE вказує, яку функцію застосовує тригер.

```
CREATE OR REPLACE FUNCTION answer_test()
RETURNS TRIGGER
AS $$
DECLARE
     answer_question integer;
```

```
     answer_question_type integer;
    true_answer_count integer;
CREATE OR REPLACE FUNCTION answer_test()
RETURNS TRIGGER
AS $$
DECLARE
     answer_question integer;
     answer_closed_question_type varchar(12);
    true_answer_count integer;
BEGIN
SELECT question
     INTO answer_question
     FROM answer
     WHERE answer_id=NEW.answer_id;

SELECT closed_question_type
     INTO answer_closed_question_type
     FROM question
     WHERE question_id=NEW.question;

SELECT COUNT(*)
     INTO true_answer_count
     FROM answer
     WHERE question = answer_question and status=true;

     IF (answer_closed_question_type='one answer') THEN
          IF (true_answer_count!=1) THEN
               RAISE EXCEPTION 'Wrong true answers number';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     ELSIF (answer_closed_question_type='some answers') THEN
          IF (true_answer_count<2) THEN
               RAISE EXCEPTION 'Wrong true answers number';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Лістинг 8.3 – Створення тригерної функції

Докладний перелік запитів бази даних вказано у додатку А.

9 ЗАПИТИ ДО БАЗИ ДАНИХ ДЛЯ ВИРІШЕННЯ ЗАДАЧ

1) Для вирішення задач Д1, Д4, Д7, Д10, Д13, Д16, В1, В4, В9, У1 потрібні наступні SQL запити.

```
INSERT INTO module(course, name) VALUES (1, 'Module 1');
INSERT INTO lesson(teacher_group_course, date_time) VALUES
(51, '2023-12-05 08:00:00');
INSERT INTO consultation(teacher_group_course, date_time)
VALUES (51, '2023-12-05 08:00:00');
INSERT INTO consultation(teacher_group_course, date_time)
VALUES (51, '2023-12-05 08:00:00');
INSERT INTO teacher(first_name, last_name, user) VALUES
('Petrenko', 'Ivan', 1);
INSERT INTO group(name, curator) VALUES ('4-A', 11);
INSERT INTO student(first_name, last_name, group, user)
VALUES ('Shevchenko', 'Stepan', 16, 2);
INSERT INTO test(name, description, time_limit, parts_type,
course) VALUES ('Test 1', 'Test 1 Description', 80, 'Full
test', 1);
INSERT INTO question(content, question_type,
closed_question_type, module) VALUES ('Question?', 'closed',
'one answer', 1);
INSERT INTO answer(question, letter, content, status, score)
VALUES (1, 'A', 'Answer', true, 5);
INSERT INTO test_result(test, student, mark, exam) VALUES
(1, 1, 1, null, 1);
INSERT INTO chosen_answer (answer, test_result_part, index)
VALUES (1, 1, 2);
INSERT INTO open_answer (question, test_result_part,
open_answer)VALUES (35, 2, 'answer 1'), (36, 2, 'answer 1'),
(37, 2, 'answer 3');
```

2) Для вирішення задач Д2, Д5, Д8, Д11, Д14, Д17, В2, В5, В8 потрібні наступні SQL запити.

```
update module set course = 2 where module_id = 12;
update lesson set date_time = '2023-12-05 09:30:00' where
lesson_id = 2;
update consultation set date_time = '2023-12-05 09:30:00'
where consultation_id = 4;
update group set name = '5-A' where group_id = 8;
update student set group = 3 where student_id = 7;
update test set time_limit = 60 where test_id = 4;
update question set content = 'New Question' where
question_id = 15;
update answer set content = 'New Answer' where answer_id =
8;
update test_result set mark = 76 where test_result_id = 21;
```

```
update chosen_answer set index = 2 where chosen_answer_id
=2;
update open_answer set open_answer = 'New Answer' where
open_answer_id = 2;
```

3) Для вирішення задач Д3, Д6, Д9, Д12, Д15, Д18, В12, Л15, В3, В6, В9 потрібні наступні SQL запити.

```
delete from module where module_id = 12;
delete from lesson where lesson_id = 2;
delete from consultation where consultation_id = 4;
delete from group where group_id = 8;
delete from student where student_id = 7;
delete from test where test_id = 4;
delete from question where question_id = 15;
delete from answer where answer_id = 8;
delete from test_result where test_result_id = 21;
delete from chosen_answer where chosen_answer_id =2;
delete from open_answer where open_answer_id = 2;
```

4) Для рішення задачі В10 використовується збережена процедура answered_percent(test_result_part_id), яка приймає ідентифікатор першої частини контрольної роботи та виводить відсоток правильно відповідених питань;
5) Для рішення задачі В11 використовується збережена процедура test_result_score(test_result_id), яка приймає ідентифікатор контрольної роботи та виводить оцінку за неї.

Усі вище перечисленні збережені процедури приведені у додатку Е.

10 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ

Для використання коду був написаний сайт за допомогою програм PhpStorm та Nicepage. Нижче наведен листинг коду головного вікна даного сайту та лістинг коду вікон для додавання, зміни та видалення данних, а також виборку.

1) Для вирішення задач Д1, Д4, Д7, Д10, Д13, Д16, В1, В4, В7, В10, В11, У1 потрібні файли addTest.php, addTest.php, createFirstPart.php, createSecondPart.php, addTestSubmit.php, createFirstPartSubmit.php, createSecondPartSubmit.php, testForExam.php, secondTestForExam.php, testForExamSubmit.php, secondTestForExamSubmit.php, administratorAccount.html, teacherAccount.html, studentAccount.html. Нижче наведено приклад файлу addTestSubmit.php на додавання нового тесту до БД.

```
<?php $conn = pg_connect("host=localhost dbname=postgres
user=postgres password=vfvf");

if (!$conn) {
    die("Connection failed: " . pg_last_error($conn));
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    try {
        $name = $_POST["name"];
        $description = $_POST["description"];
        $time_limit = $_POST["time_limit"];
        $course = $_POST['course'];
        $type = $_POST['type'];

        $query = 'INSERT INTO test (name, description,
time_limit, course, parts_type) VALUES ($1, $2, $3, $4, $5)
RETURNING test_id';
        $values = array($name, $description, $time_limit,
$course, $type);
        $result = pg_query_params($conn, $query, $values);

        if (!$result) {
            throw new Exception("Query failed: " .
pg_last_error($conn));
        }
        $row = pg_fetch_assoc($result);
        $test_id = $row['test_id'];
```

```
        pg_close($conn);
        header("Location: createFirstPart.php?test=" .
urlencode($test_id) . "&type=". urlencode($type));
        exit;
    } catch (Exception $e) {
        $error = "Error: " . $e->getMessage();
        header("Location: addTest.php?error=" .
urlencode($error));
    }
}
```

2) Для вирішення задач Д2, Д3, Д5, Д6, Д8, Д9, Д11, Д12, Д14, Д15, Д17, Д18, В2, В3, В5, В6, В8, В9 потрібні файли test.html, administratorAccount.html, teacherAccount.html, studentAccount.html, studentTests.php, studentTestsSubmit.php, editProfile.php, editProfileSubmit.php.

   Нижче наведено приклад файлу editProfileSubmit.php на змінення особистих даних користувача.

```
<?php
$id = $_GET['id'];
$role = $_GET['role'];

$conn = pg_connect("host=localhost dbname=postgres
user=postgres password=vfvf");

if (!$conn) {
    die("Connection failed: " . pg_last_error($conn));
}

if ($role == 'teacher') {
    $query = 'SELECT first_name, last_name, password from
teacher JOIN "user" on user_id=teacher.user where
teacher_id=$1;';
} else if ($role == 'student') {
    $query = 'SELECT first_name, last_name, password from
student JOIN "user" on user_id=student.user where
student_id=$1;';
} else {
    $query = 'SELECT first_name, last_name, password from
director JOIN "user" on user_id=director.user where
director_id=$1;';
}

$values = array($id);
$result = pg_query_params($conn, $query, $values);
```

```
if ($result === false) {
    die("Query failed: " . pg_last_error($conn));
}

$row = pg_fetch_assoc($result);

if ($row !== false) {
    $first_name = $row['first_name'];
    $last_name = $row['last_name'];
    $password = $row['password'];

} else {
    echo "No data found for ID: $id";
}

pg_close($conn);
?>
<!DOCTYPE html>
<html style="font-size: 16px;" lang="en"><head>
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <meta charset="utf-8">
    <meta name="keywords" content="Get in Touch">
    <meta name="description" content="">
    <link rel="stylesheet" href="nicepage.css"
media="screen">
<link rel="stylesheet" href="editProfile.css"
media="screen">
    <script class="u-script" type="text/javascript"
src="jquery.js" defer=""></script>
    <script class="u-script" type="text/javascript"
src="nicepage.js" defer=""></script>
    <meta name="generator" content="Nicepage 5.20.7,
nicepage.com">
    <meta name="referrer" content="origin">
    <link id="u-theme-google-font" rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Roboto:100,100
i,300,300i,400,400i,500,500i,700,700i,900,900i|Open+Sans:300
,300i,400,400i,500,500i,600,600i,700,700i,800,800i">
    <link id="u-page-google-font" rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Montserrat:100
,100i,200,200i,300,300i,400,400i,500,500i,600,600i,700,700i,
800,800i,900,900i">

    <script type="application/ld+json">{
      "@context": "http://schema.org",
      "@type": "Organization",
      "name": "",
      "logo": "images/PinClipart.com_line-dancing-clip-
art_5521613.png"
}</script>
    <meta name="theme-color" content="#478ac9">
    <meta property="og:title" content="Edit Profile">
```

```
    <meta property="og:type" content="website">
  <meta data-intl-tel-input-cdn-path="intlTelInput/"></head>
  <body data-path-to-root="./" data-include-products="false"
class="u-body u-xl-mode" data-lang="en"><header class="u-
align-center-xs u-border-1 u-border-grey-25 u-clearfix u-
header u-header" id="sec-5640"><div class="u-clearfix u-
sheet u-sheet-1">
        <a href="index.html" class="u-image u-logo u-image-
1" data-image-width="816" data-image-height="605"
title="logo">
          <img src="images/PinClipart.com_line-dancing-clip-
art_5521613.png" alt="logo" class="u-logo-image u-logo-
image-1">
        </a>
        <form action="#" method="get" class="u-border-1 u-
border-grey-15 u-search u-search-right u-search-1">
          <button class="u-search-button" type="submit">
            <span class="u-search-icon u-spacing-10 u-text-
grey-40">
              <svg class="u-svg-link"
preserveAspectRatio="xMidYMin slice" viewBox="0 0 56.966
56.966" style=""><use
xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#svg-
b04b"></use></svg>
              <svg xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1"
id="svg-b04b" x="0px" y="0px" viewBox="0 0 56.966 56.966"
style="enable-background:new 0 0 56.966 56.966;"
xml:space="preserve" class="u-svg-content"><path
d="M55.146,51.887L41.588,37.786c3.486-4.144,5.396-
9.358,5.396-14.786c0-12.682-10.318-23-23-23s-23,10.318-23,23
s10.318,23,23,23c4.761,0,9.298-1.436,13.177-
4.162l13.661,14.208c0.571,0.593,1.339,0.92,2.162,0.92
c0.779,0,1.518-0.297,2.079-
0.837C56.255,54.982,56.293,53.08,55.146,51.887z
M23.984,6c9.374,0,17,7.626,17,17s-7.626,17-17,17  s-17-
7.626-17-
17S14.61,6,23.984,6z"></path><g></g><g></g><g></g><g></g><g>
</g><g></g><g></g><g></g><g></g><g></g><g></g><g></g><g></g>
<g></g><g></g></svg>
            </span>
          </button>
          <input class="u-search-input" type="search"
name="search" value="" placeholder="Search">
        </form>
        <nav class="u-align-left u-font-size-14 u-menu u-
menu-one-level u-nav-spacing-25 u-offcanvas u-menu-1">
          <div class="menu-collapse">
            <a class="u-button-style u-nav-link" href="#"
style="padding: 4px 0px; font-size: calc(1em + 8px);">
              <svg class="u-svg-link"
preserveAspectRatio="xMidYMin slice" viewBox="0 0 302 302"
style=""><use xmlns:xlink="http://www.w3.org/1999/xlink"
```

```
xlink:href="#svg-7b92"></use></svg>
              <svg xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1"
id="svg-7b92" x="0px" y="0px" viewBox="0 0 302 302"
style="enable-background:new 0 0 302 302;"
xml:space="preserve" class="u-svg-content"><g><rect y="36"
width="302" height="30"></rect><rect y="236" width="302"
height="30"></rect><rect y="136" width="302"
height="30"></rect>
</g><g></g><g></g><g></g><g></g><g></g><g></g><g></g><g></g>
<g></g><g></g><g></g><g></g><g></g><g></g><g></g></svg>
            </a>
          </div>
          <div class="u-custom-menu u-nav-container">
            <ul class="u-nav u-unstyled u-nav-1"><li
class="u-nav-item"><a class="u-button-style u-nav-link u-
text-active-palette-1-base u-text-hover-palette-2-base"
href="index.html" style="padding: 10px 20px;">Home</a>
</li></ul>
          </div>
          <div class="u-custom-menu u-nav-container-
collapse">
            <div class="u-align-center u-black u-container-
style u-inner-container-layout u-opacity u-opacity-95 u-
sidenav">
              <div class="u-inner-container-layout u-
sidenav-overflow">
                <div class="u-menu-close"></div>
                <ul class="u-align-center u-nav u-popupmenu-
items u-unstyled u-nav-2"><li class="u-nav-item"><a
class="u-button-style u-nav-link" href="index.html">Home</a>
</li></ul>
              </div>
            </div>
            <div class="u-black u-menu-overlay u-opacity u-
opacity-70"></div>
          </div>
        </nav>
      </div></header>
    <section class="u-clearfix u-section-1"
id="carousel_2fe8">
      <div class="u-clearfix u-sheet u-sheet-1">
        <div class="data-layout-selected u-clearfix u-
layout-wrap u-layout-wrap-1">
          <div class="u-layout">
            <div class="u-layout-row">
              <div class="u-align-left u-container-style u-
layout-cell u-size-26 u-white u-layout-cell-1">
                <div class="u-container-layout u-container-
layout-1">
                  <h2 class="u-align-left u-custom-font u-
font-montserrat u-text u-text-1"><?php echo $first_name . "
" . $last_name ?></h2>
```

```
                    <h3 class="u-align-left u-custom-font u-
font-montserrat u-text u-text-1">Edit Profile</h3>
                    <div class="u-align-left u-expanded-
width-lg u-expanded-width-md u-expanded-width-sm u-expanded-
width-xs u-form u-form-1">
                    <form
action="https://forms.nicepagesrv.com/v2/form/process"
class="u-clearfix u-form-spacing-28 u-form-vertical u-inner-
form" style="padding: 0px;" source="email" name="form">
                      <div class="u-form-group u-label-top
u-form-group-3">
                        <label for="password" class="u-
label">Password</label>
                        <input type="text" value="<?php echo
$password; ?>" placeholder="Enter new password"
id="password" name="password" class="u-border-none u-grey-5
u-input u-input-rectangle u-radius-7 u-input-3">
                      </div>
                      <div class="u-align-right u-form-group
u-form-submit u-label-top">
                        <a href="#" class="u-active-black u-
border-none u-btn u-btn-round u-btn-submit u-button-style u-
hover-black u-palette-3-base u-radius-7 u-text-active-white
u-text-body-alt-color u-text-hover-white u-btn-1">Submit</a>
                        <input type="submit" value="submit"
class="u-form-control-hidden" wfd-invisible="true">
                      </div>
                      <div class="u-form-send-message u-
form-send-success" wfd-invisible="true"> Thank you! Your
message has been sent. </div>
                      <div class="u-form-send-error u-form-
send-message" wfd-invisible="true"> Unable to send your
message. Please fix errors then try again. </div>
                      <input type="hidden" value=""
name="recaptchaResponse" wfd-invisible="true">
                      <input type="hidden"
name="formServices" value="7835512a-9771-78b5-f01f-
ff32271acd2d">
                    </form>
                  </div>
                </div>
              </div>
              <div class="u-container-style u-layout-cell u-
size-34 u-layout-cell-2" data-animation-name="" data-
animation-duration="0" data-animation-delay="0" data-
animation-direction="">
                <div class="u-container-layout u-container-
layout-2">
                  <img class="u-expanded-width u-image u-
image-contain u-image-1" src="images/profile.jpg" data-
image-width="564" data-image-height="564">
                </div>
              </div>
```

```
            </div>
          </div>
        </div>
      </div>
    </section>
    <section class="u-clearfix u-section-2" id="sec-02ee">
      <div class="u-clearfix u-sheet u-sheet-1"></div>
    </section>

    <footer class="u-align-center u-clearfix u-footer u-
grey-80 u-footer" id="sec-0ed4"><div class="u-clearfix u-
sheet u-sheet-1">
        <p class="u-small-text u-text u-text-variant u-text-
1"> Automation of educational process control</p>
      </div></footer>
</body></html>
```

11 БЕЗПЕКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

Безпека на рівні БД забезпечується шляхом створення ролей і наділення їх відповідними прівілегіями. В інформаційній системі для контролю за освітнім процесом реалізовано три ролі: директор, вчитель та учень. В таблиці 11.1 наведені привілеї ролей до таблиць БД. При цьому використані наступні позначення:

C (Create) - створення (додавання) даних;

R (Read) - читання даних;

U (Update) - оновлення даних;

D (Delete) - видалення даних;

E (Execute) - виконання функції.

Таблиця 11.1 – Ролі та привілеї таблиць та функцій БД

Об’єкти БД	Ролі		
	Учень	Вчитель	Директор
Сутності			
Answer	R	CRUD	R
Chosen_answer	CRUD	R	R
Consultation	R	R	CRUD
Course	R	R	CRUD
Director			CRUD
Exam	R	R	CRUD
Group	R	R	CRUD
Lesson	R	R	CRUD
Lesson_consultation _duration	R	R	CRUD
Matrix_question_li	R	CRUD	R
Module	R	R	CRUD
Open_answer	CRUD	R	R
Question	R	CRUD	R

Продовження таблиці 11.1

Об’єкти БД	Ролі		
	Учень	Вчитель	Директор
Question_test_part	R	CRUD	R
Student	CRUD	R	CRUD
Teacher	R	CRUD	CRUD
Teacher_group_course	R	R	CRUD
Test	R	CRUD	R
Test_part	R	CRUD	R
Test_result	R	R	R
Test_result_part	CRUD	R	R
User	CRUD	CRUD	CRUD
Функції			
Answered_percent		E	
First_test_part_score		E	
Get_user_id	E	E	E
Get_user_role	E	E	E
Get_user_role_id	E	E	E
Is_question_answered		E	
Question_score		E	
Second_test_part_score		E	
Test_result_closed_questions_score	E	E	
Test_result_open_questions_score	E	E	
Test_result_question_score	E	E	
Test_result_score	E	E	

Створення ролей і наділення їх привілеями відповідно до наведеної вище таблиці здійснюється за допомогою наступних SQL-запитів:

```
CREATE ROLE director with login password 'directorPassword';
```

```
GRANT SELECT ON answer, chosen_answer, consultation, course,
director, exam, group, lesson, lesson_consultation_duration,
matrix_question_li, module, open_answer, question,
question_test_part, student, teacher, teacher_group_course,
test, test_part, test_result, test_result_part, user to
director;
GRANT GRANT INSERT, UPDATE, DELETE ON consultation, course,
director, exam, group, lesson, lesson_consultation_duration,
module, student, teacher, teacher_group_course, user to
director;
GRANT EXECUTE ON get_user_id, get_user_role, get_user_role_id to
director;

CREATE ROLE teacher with login password 'teacherPassword';
GRANT SELECT ON answer, chosen_answer, consultation, course,
exam, group, lesson, lesson_consultation_duration,
matrix_question_li, module, open_answer, question, student,
teacher, teacher_group_course, test, test_part, test_result,
test_result_part, user to teacher;
GRANT INSERT, UPDATE, DELETE ON answer, matrix_question_li,
question, question_test_part, teacher, test, test_part, user to
teacher;
GRANT EXECUTE ON answered_percent, first_test_part_score,
get_user_id, get_user_role, get_user_role_id,
is_question_answered, question_score, second_test_part_score,
test_result_closed_questions_score,
test_result_open_questions_score, test_result_question_score,
test_result_score to teacher;

CREATE ROLE student with login password 'studentPassword';
GRANT SELECT ON answer, chosen_answer, consultation, course ,
exam, group, lesson, lesson_consultation_duration,
matrix_question_li, module, open_answer, question ,
question_test_part, student, teacher, teacher_group_course,
test, test_part, test_result, test_result_part, user to student;
GRANT INSERT, DELETE, UPDATE ON chosen_answer, open_answer,
student, test_result, test_result_part, user to student;
GRANT EXECUTE ON get_user_id, get_user_role, get_user_role_id,
test_result_closed_questions_score,
test_result_open_questions_score, test_result_question_score,
test_result_score to student;
```

12 ІНСТРУКЦІЯ КОРИСТУВАЧА

При запуску програми користувач бачить домашню сторінку (рис. 12.1). З неї можна зареєструватися в системі (рис. 12.2) чи увійти (рис. 12.3) в неї, скориставшись посиланнями у шапці.

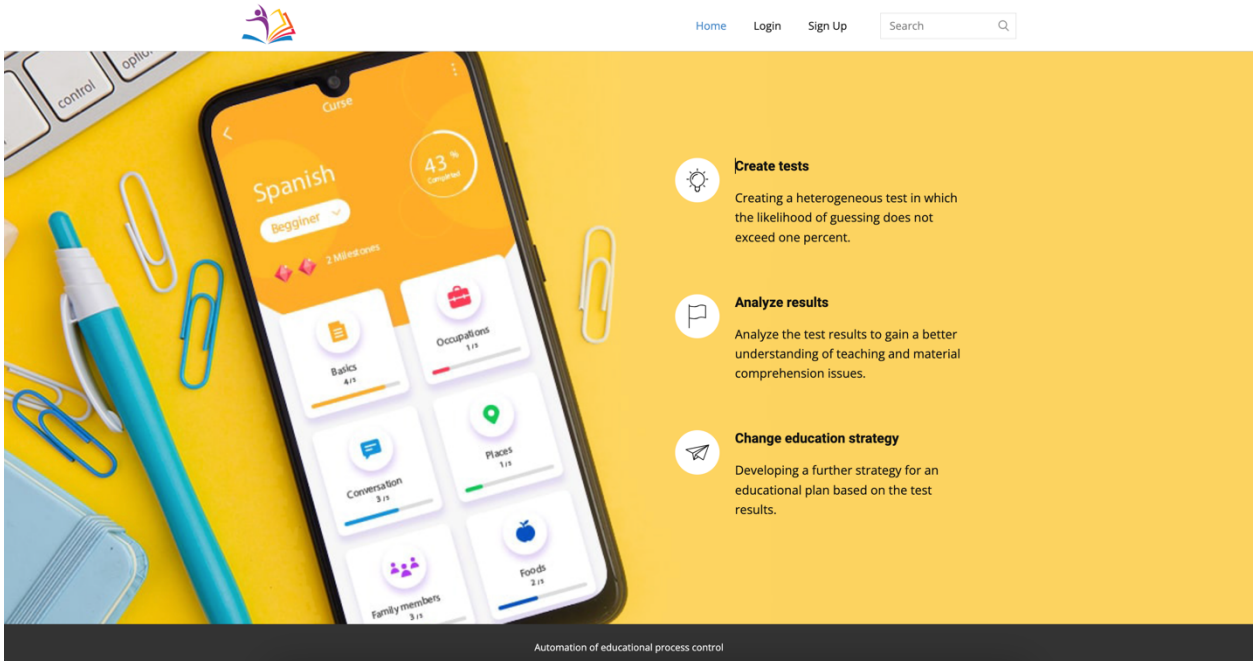

Рисунок 12.1 – Домашня сторінка

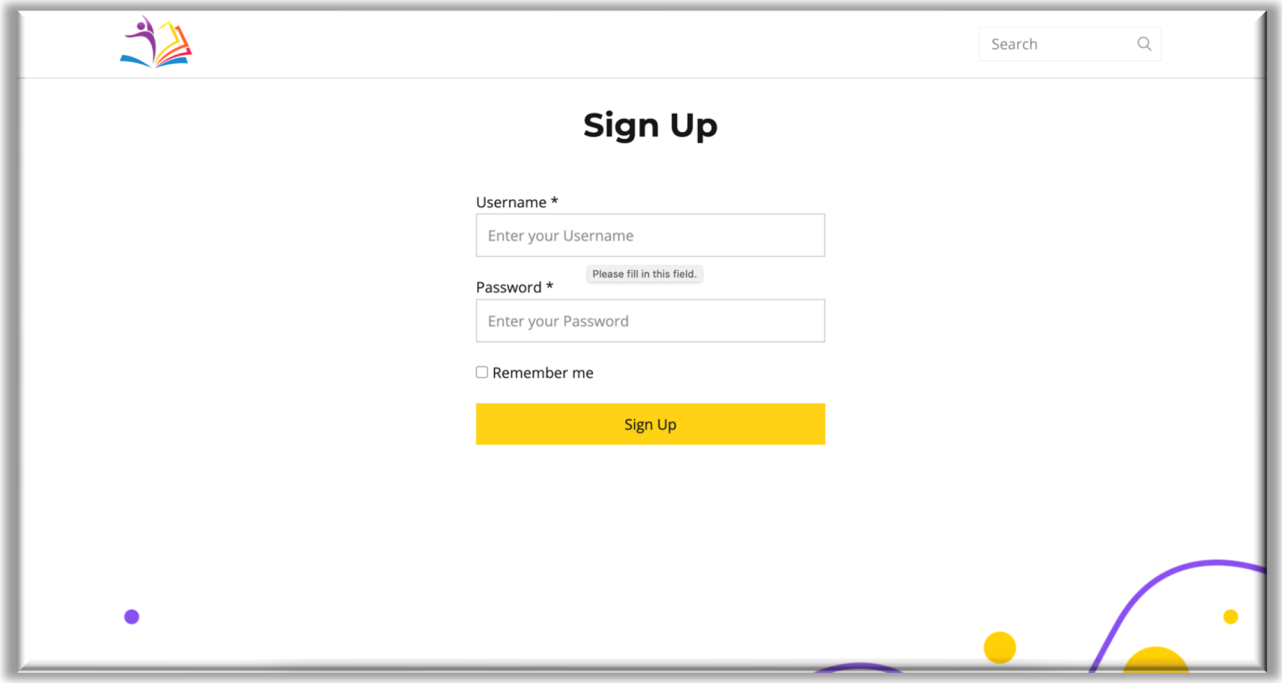

Рисунок 12.2 – Реєстрація

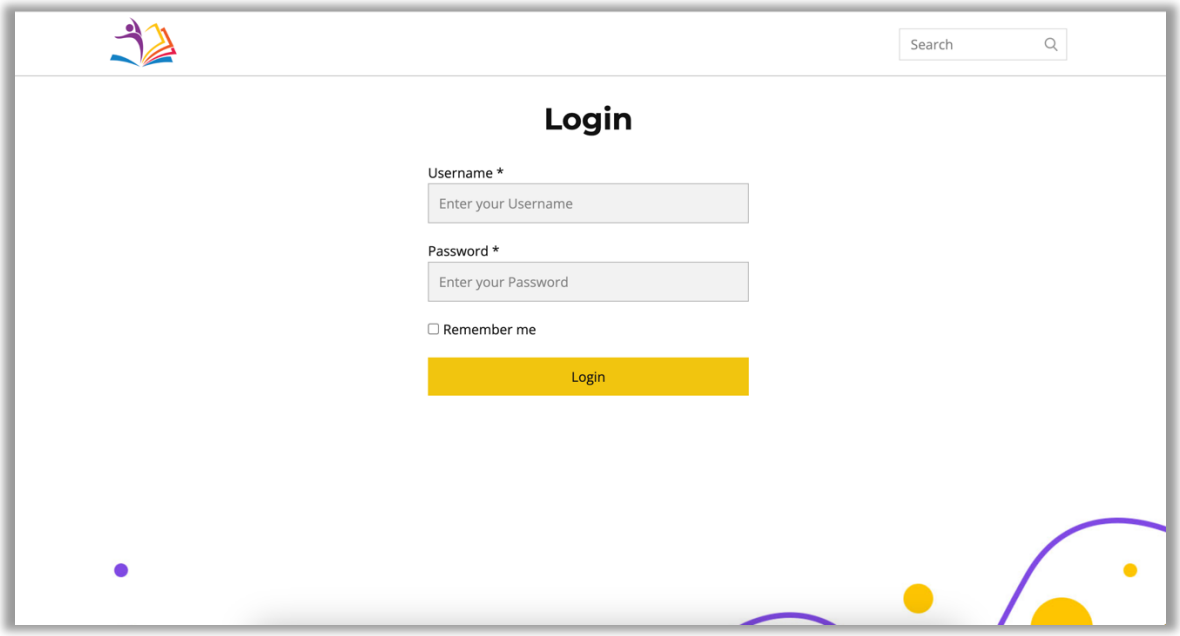

Рисунок 12.3 – Вхід в систему

В системі можна зареєструватися, як вчитель, учень або директор (рис. 12.4).

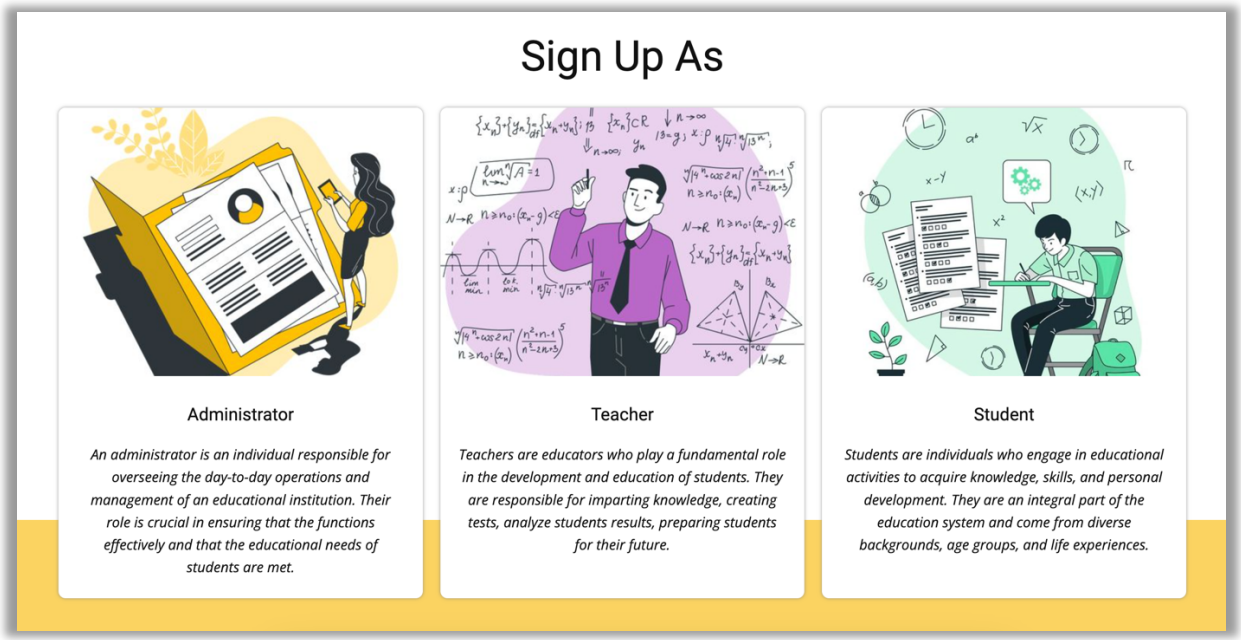

Рисунок 12.4 – Варіанти реєстрації

Далі користувач потрапляє на свій аккаунт. На рисунку 12.5 наведений приклад аккаунту вчителя. Натиснувши на кнопку Get Details у розділі тест вчитель може створити, перевірити та проаналізувати тести (рисю 12.6).

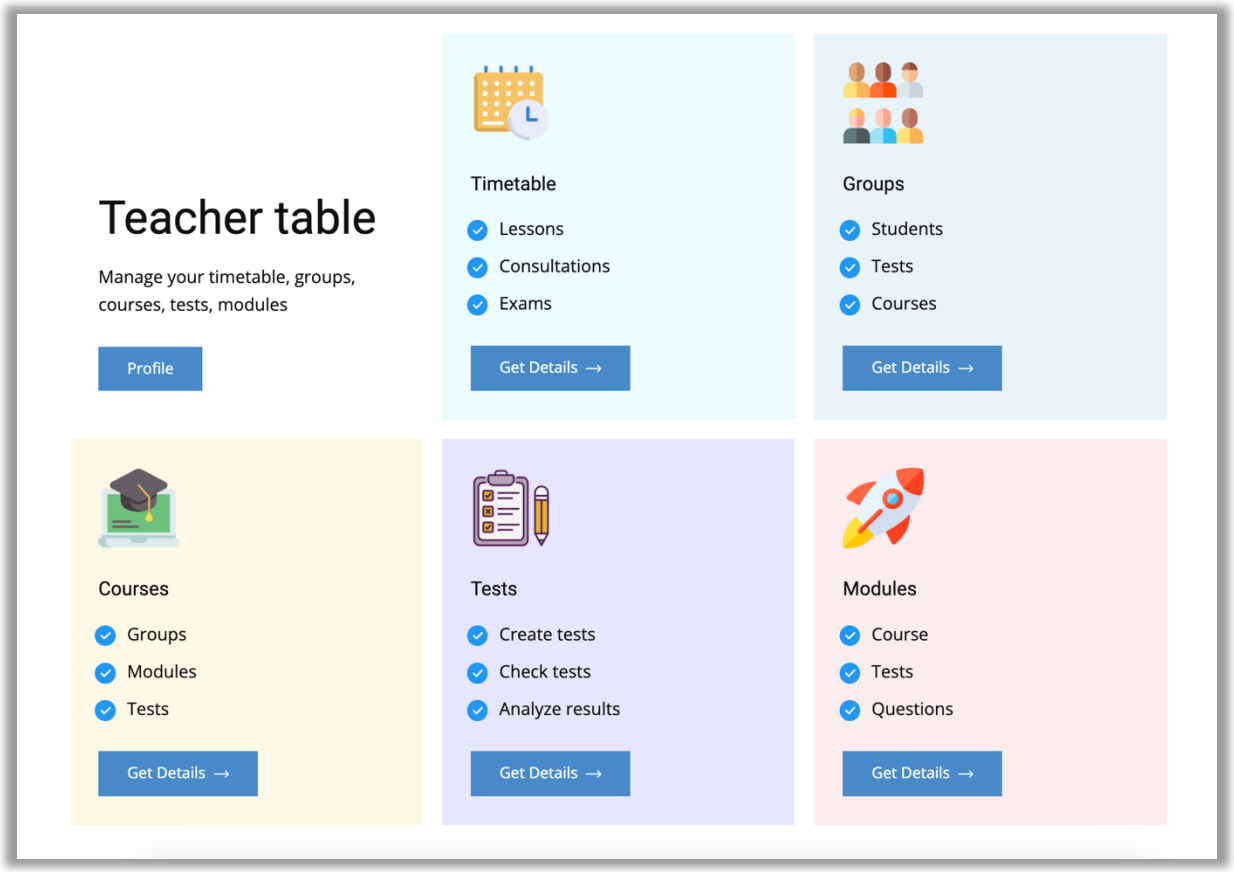

Рисунок 12.5 – Варіанти реєстрації

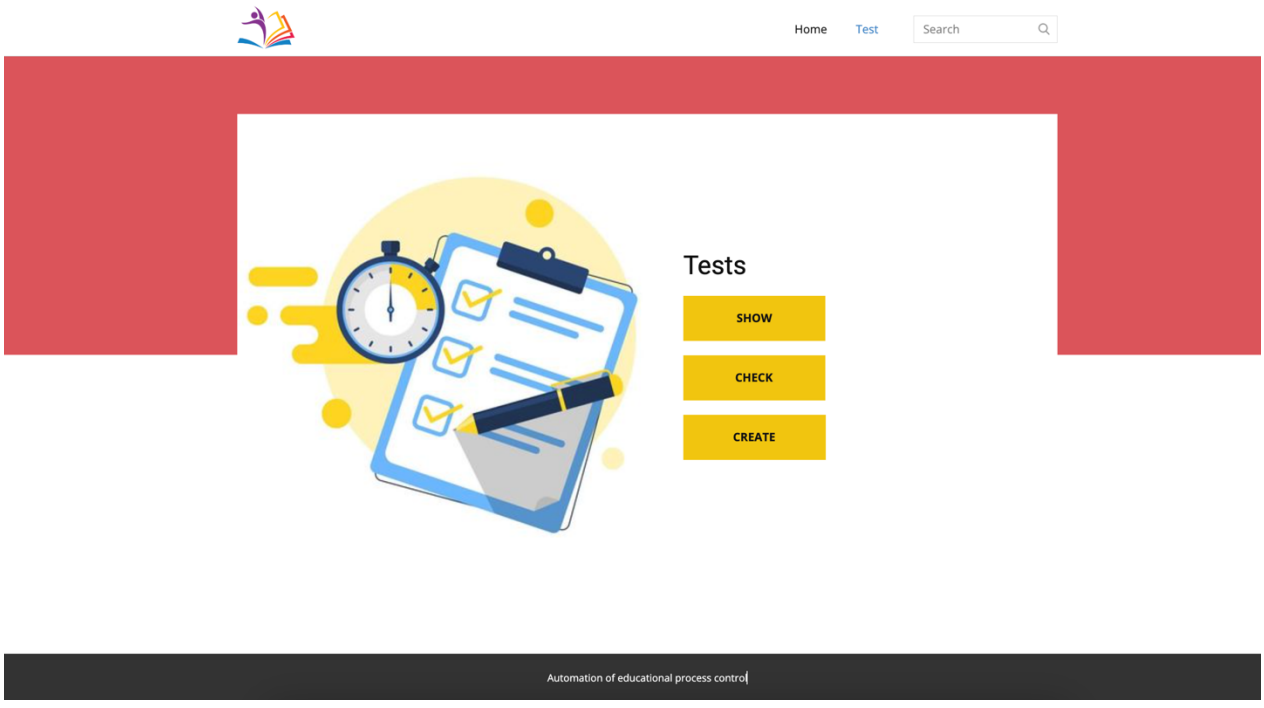

Рисунок 12.6 – Варіанти реєстрації

Для створення тесту треба натиснути на кнопку Create та заповнити всі необхідні поля (рис. 12.7).

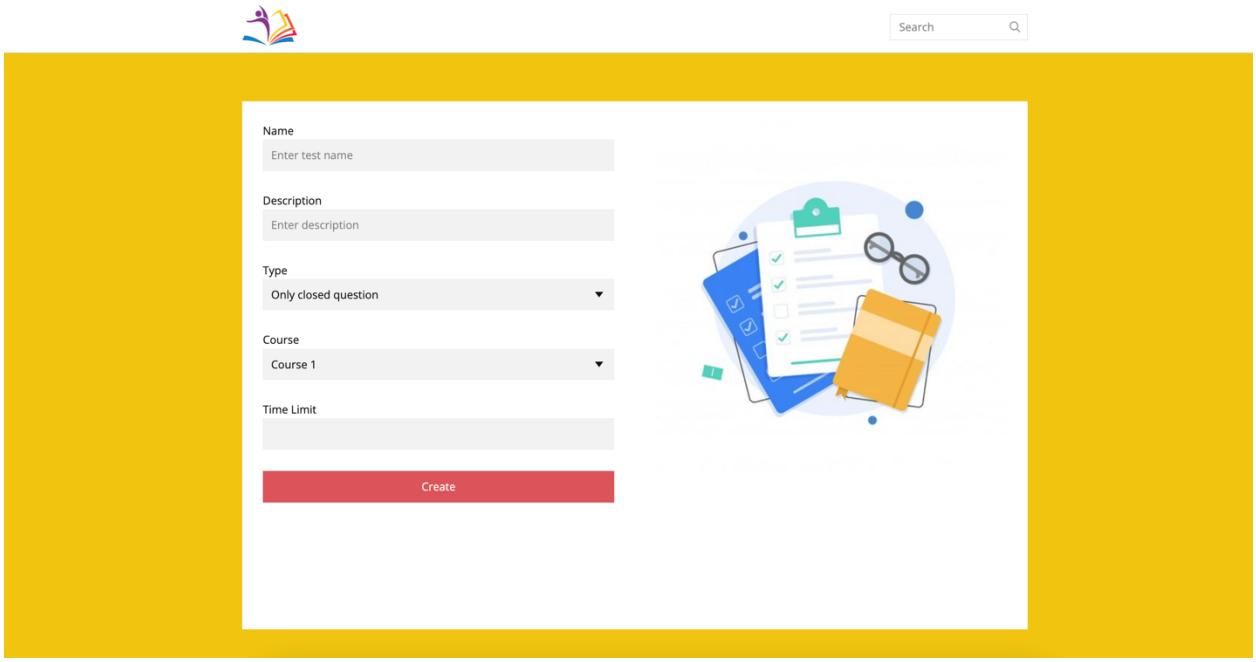

Рисунок 12.7 – Створення тесту

Далі в залежності від того, зі скількох частин складається тест, треба створити одну чи дві частини тесту. Тип може буди «First Part», «Full Test», «Depend of condition». Якщо обрати «First Part» складається тільки одна частина, якщо «Full Test» або «Depend of condition» - дві частини.

У цій формі можна додавати питання, відповіді та теми до тесту. Кількість питань в темі повинна бути не менше 5. Кількість варіантів відповідей теж повинна перевищувати якесь число залежно від типу питання. В першій частині можуть бути тільки закриті питання, друга частина призначена для питань з відкритою відповіддю (рис. 12.8 та 12.9).

Вчитель може побачити та проаналізувати результати учнів, якщо натисне на кнопку «Check». Тут зберігається вся необхідна інформація про тести (рис. 12.10). Якщо натиснути на кнопку «Show» з’являться результати тестування учнів (рис. 12.11). Є можливість проаналізувати питання, які були на тесті (рис. 12.12).

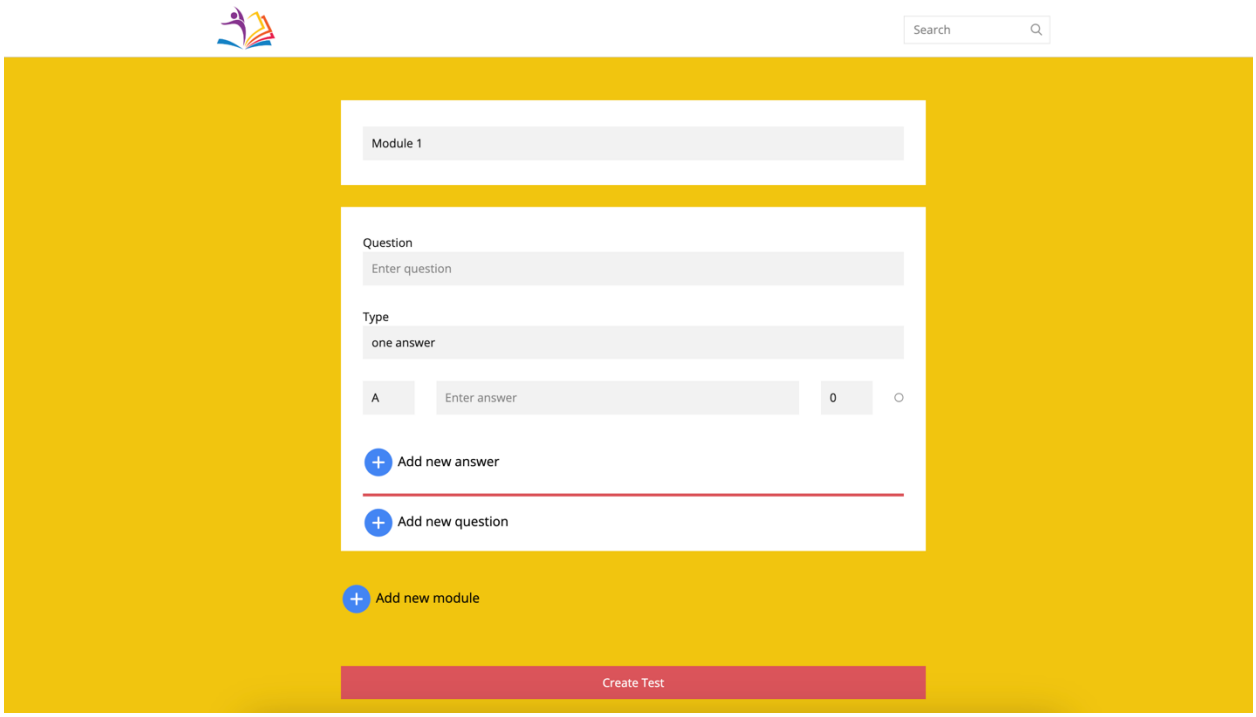

Рисунок 12.8 – Створення першої частини тесту

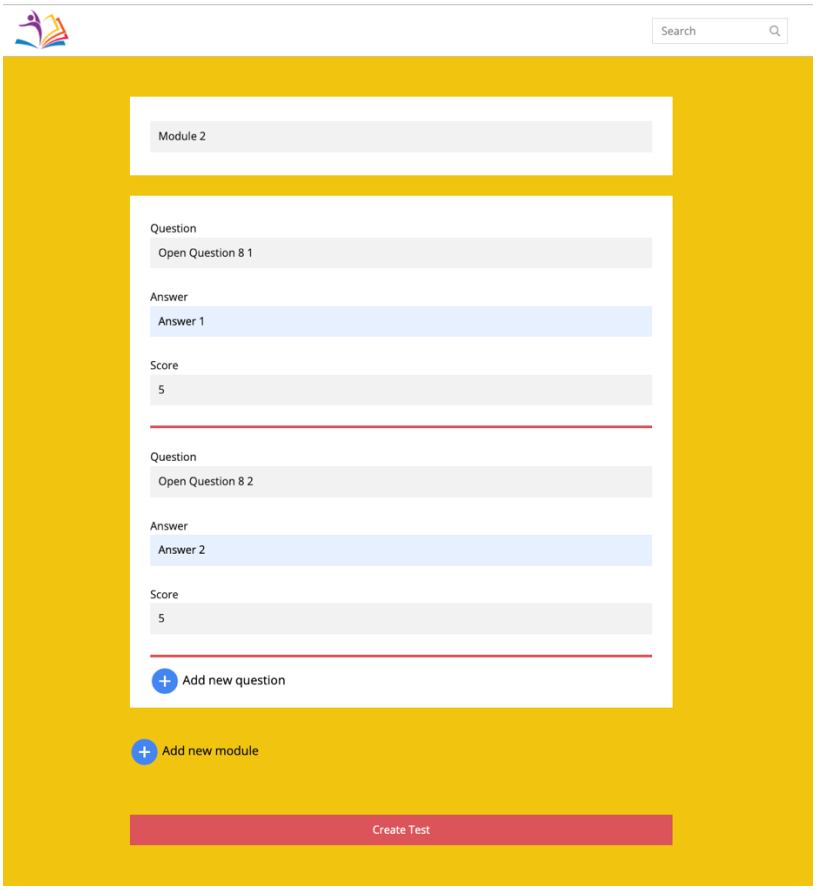

Рисунок 12.9 – Створення другої частини тесту

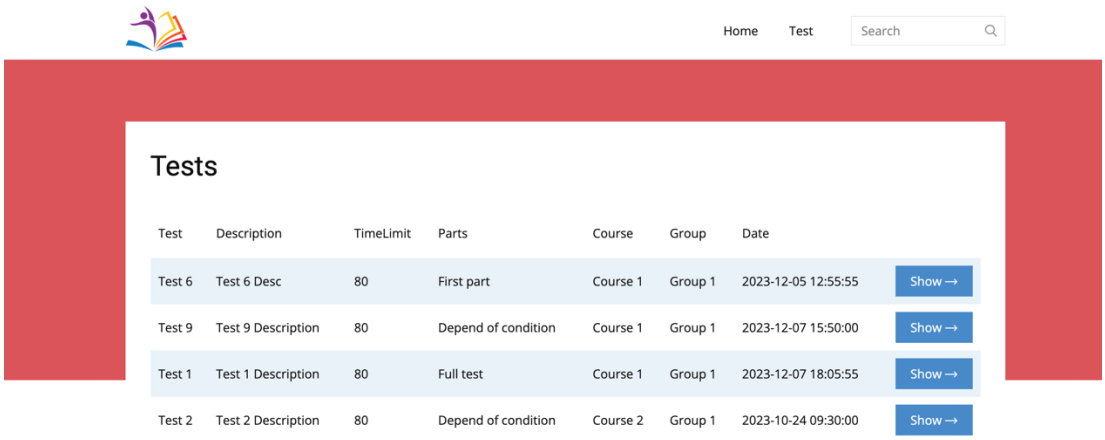

Рисунок 12.10 – Інформація про тести

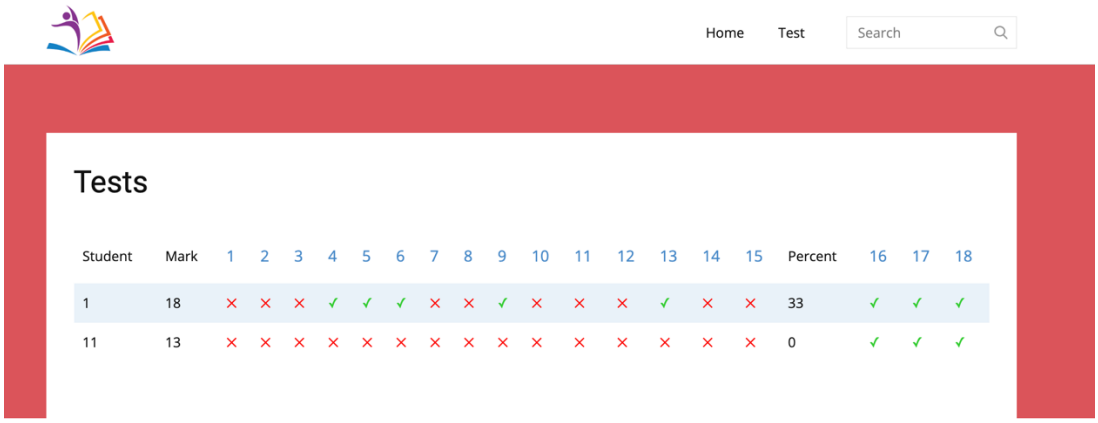

Рисунок 12.11 – Результати тестування

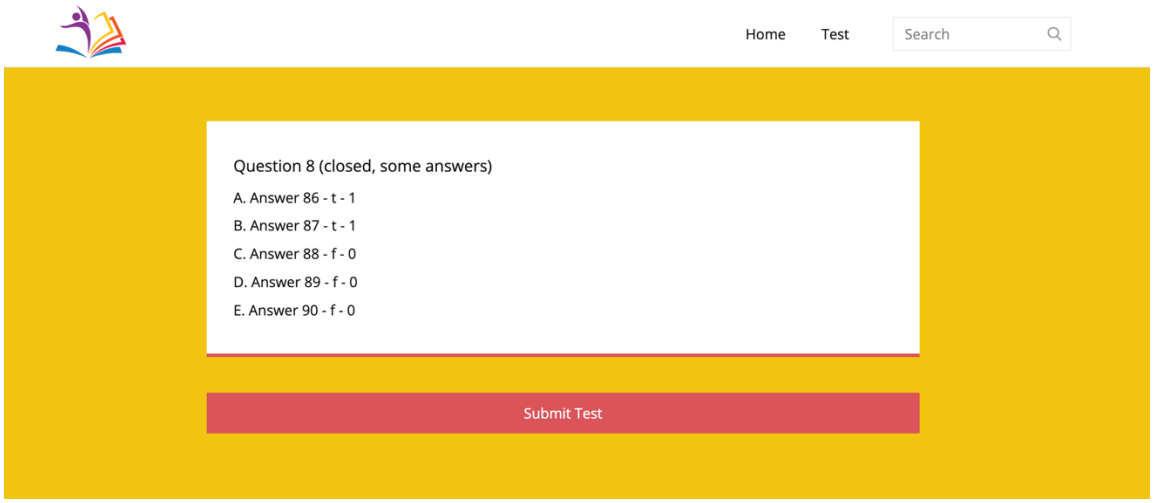

Рисунок 12.11 – Питання з тесту

На рисунку 12.12 зображено аккаунт студента. Натиснувши на кнопку «Get Details» можна отримати інформацію про екзамени та скласти їх, якщо є активні (рисунку 12.13).

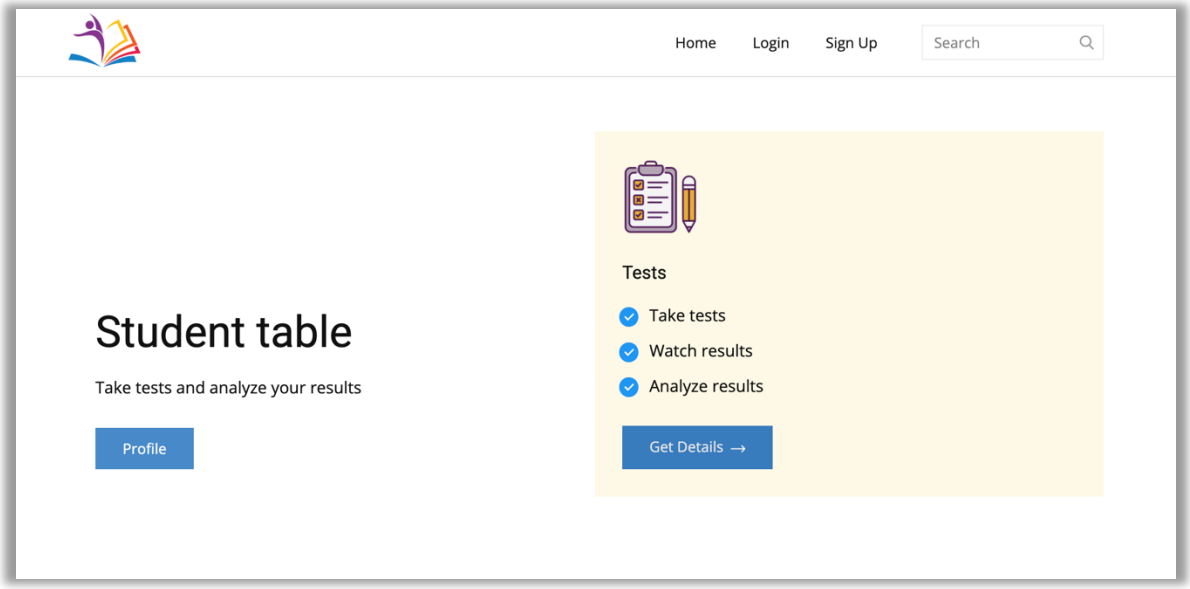

Рисунок 12.12 – Аккаунт учня

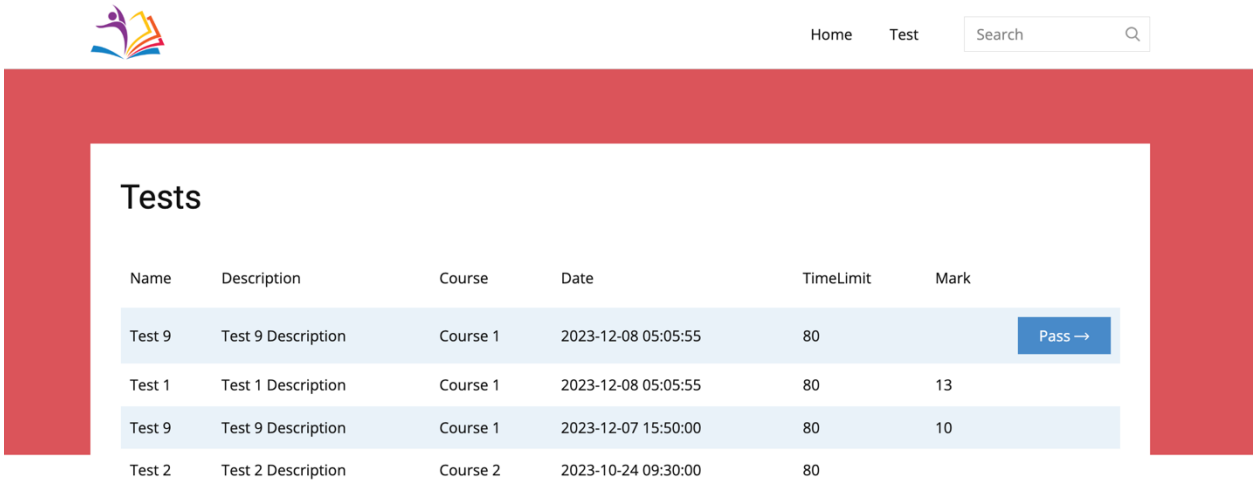

Рисунок 12.13 – Тести

Після цього можна відповісти на запитання у вікні 12.14. Якщо менш ніж 75% запитань першої частини складено правильно, то необхідно пройти другу частину (рис. 12.15). Результати тестування представленні на рисунку 12.16.

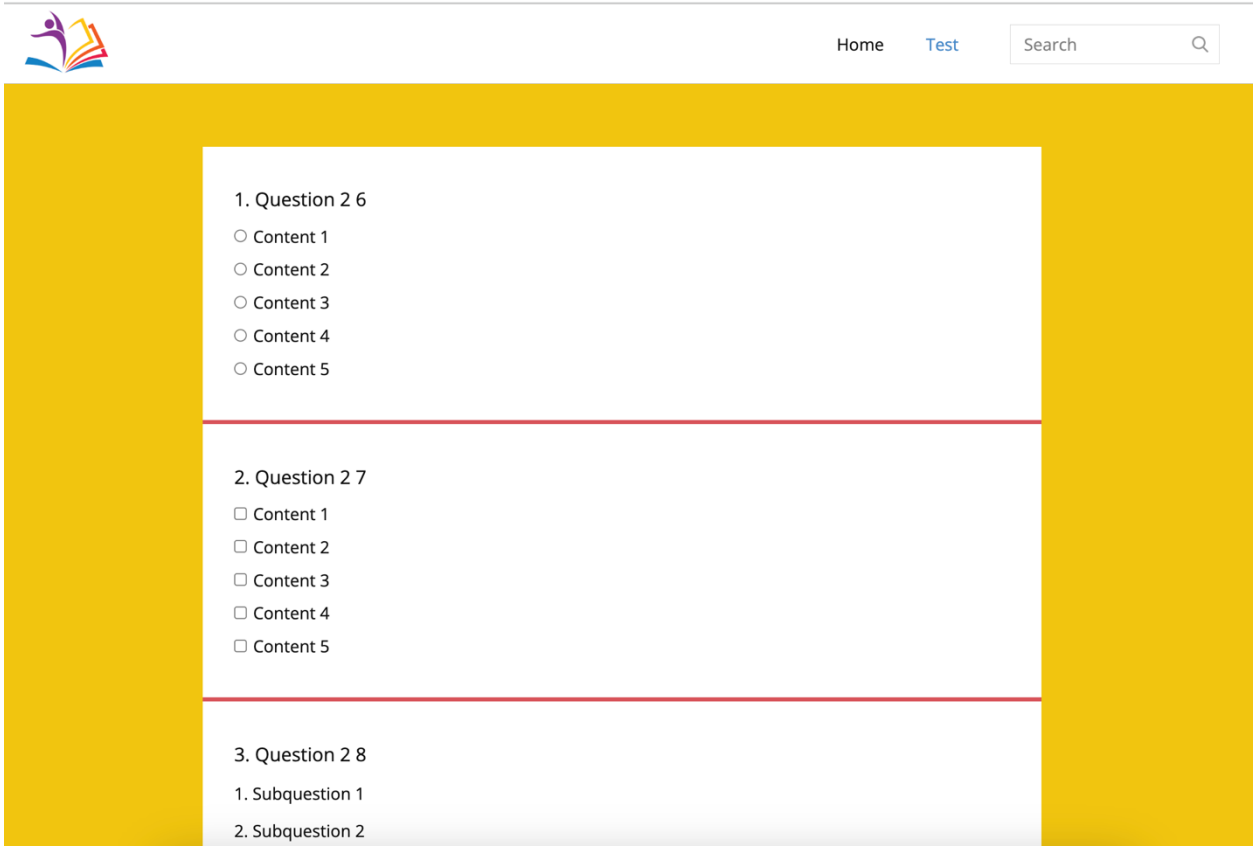

Рисунок 12.13 – Питання першої частини тесту

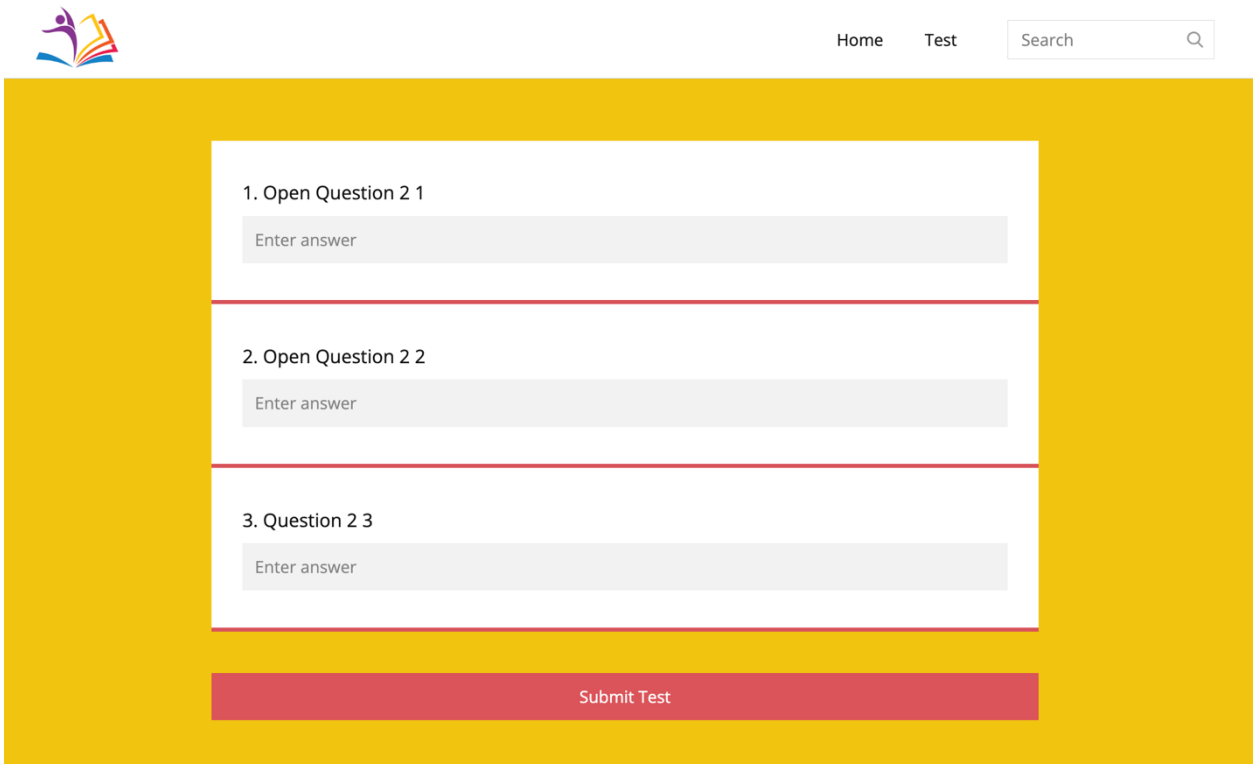

Рисунок 12.14 – Питання другої частини тесту

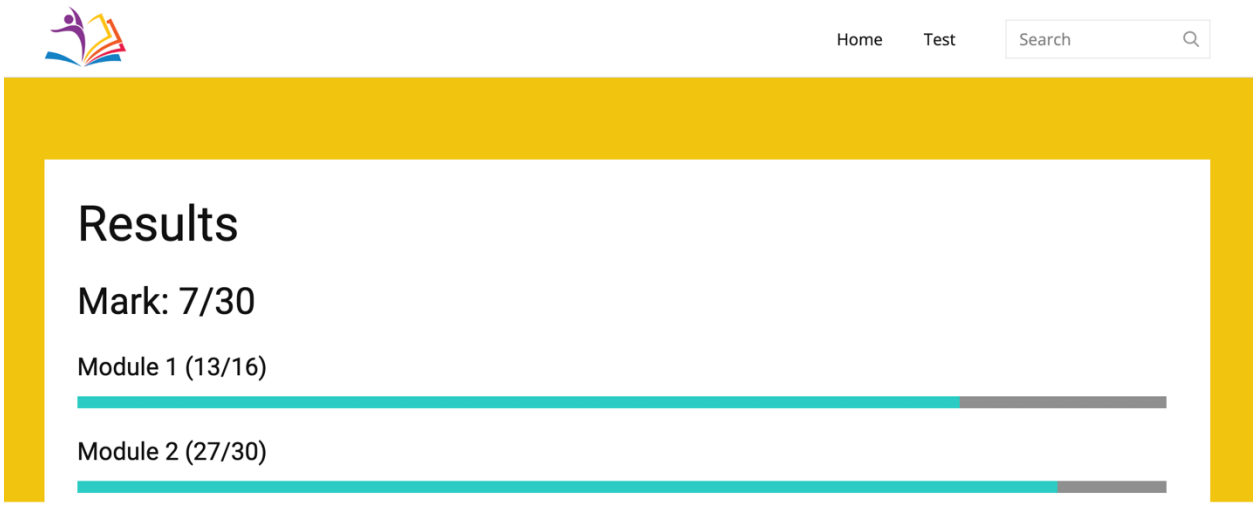

Рисунок 12.15 – Результати учня

ВИСНОВКИ

Даний проект, спрямований на розробку та впровадження автоматизованої системи для моніторингу та контролю освітнього процесу, має важливий потенціал в сфері покращення методів тестування та управління освітнім процесом. Мета дослідження, а саме зменшення ймовірності вгадування відповідей на тестові запитання, вказує на важливість об'єктивності та точності в оцінці знань студентів.

Впровадження нової структури тестових завдань, спрямованої на ускладнення випадкового визначення правильних відповідей, може сприяти підвищенню якості тестування та забезпечити більш об'єктивне вимірювання рівня знань студентів. Функції системи, такі як відстеження прогресу студентів, управління академічними ресурсами та формування стратегії навчання, додатково сприятимуть оптимізації освітнього процесу.

Цей проект відповідає вимогам сучасного освітнього середовища, де інновації та використання технологій важливі для підвищення ефективності та якості навчання. Зокрема, автоматизація ключових аспектів освітнього процесу сприятиме не лише оптимізації адміністративних завдань, але й наданню звітування щодо академічної успішності в реальному часі.

В цілому, проект відображає стратегічний підхід до вдосконалення системи освіти через використання сучасних технологій та інновацій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Основні види контролю успішності учнів. Взято з https://osvita.ua/vnz/reports/pedagog/14440/

[2] Види контролю знань, умінь і навичок учнів. (2010). Маріуполь. Взято з https://studfile.net/preview/3283315/page:26/

[3] Шуліга А.Є. (2018, Липень 27). Тести як засіб перевірки знань школярів. Взято з https://www.historyua.com/2018/07/27/testy-yak-zasib-perevirky-znan-shkolyariv/

[4] Використання тестових технологій для контролю знань та умінь учнів. Взято з https://naurok.com.ua/vikoristannya-testovih-tehnologiy-dlya-kontrolyu-znan-ta-umin-uchniv-155709.html

[5] Kahoot! for schools: how it works | Feature overview. Взято з https://kahoot.com/schools/how-it-works/

[6] What is ProProfs and How Does It Work? Best Tips and Tricks | Tech & Learning. Взято з https://www.techlearning.com/how-to/what-is-proprofs-and-how-does-it-work-best-tips-and-tricks

[7] What is SurveyMonkey? Взято з https://www.genroe.com/blog/what-is-surveymonkey

[8] The Features and Functions of Google Forms, Explained. Взято з https://form-publisher.com/blog/features-and-functions-of-google-forms/

[9] Features – MoodleDocs. Взято з https://docs.moodle.org/403/en/Features

[10] Features – PhpStorm. Взято з https://www.jetbrains.com/phpstorm/features/

[11] SQL Complete Developer's Course (MySQL & PostgreSQL). Взято з https://oracle-patches.com/en/lib/books/databases/sql-the-complete-developer-s-guide-mysql,-postgresql

[12] Nicepage Reviews - 2023. Взято з https://slashdot.org/software/p/Nicepage

[13] КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА. Взято з https://training.qatestlab.com/blog/technical-articles/client-server-architecture/

[14] Що таке MVC? - Робота та переваги - Сфера та кар'єра. Взято з https://uk.education-wiki.com/3886982-what-is-mvc

ДОДАТОК А

Запити БД

```
//створення таблиці course
create table course(
     course_id serial not null primary key,
     course_name varchar(40) not null,
     description varchar(255) not null
);

CREATE table user(
     user_id serial not null,
     login varchar(20) not null unique,
     password varchar(12) not null
);

//створення таблиці teacher
create table teacher(
     teacher_id serial not null primary key,
     first_name varchar(40) not null,
     last_name varchar(40) not null,
     user int not null unique,
     foreign key (user) references user (user_id) on delete
restrict on update cascade
);

//створення таблиці director
create table director(
     director_id serial not null primary key,
     first_name varchar(40) not null,
     last_name varchar(40) not null,
     user int not null unique,
     foreign key (user) references user (user_id) on delete
restrict on update cascade

);

//створення таблиці group
create table public.group(
     group_id serial not null primary key,
     group_name varchar(20) not null,
     curator int not null,
     foreign key (curator) references teacher (teacher_id) on
delete restrict on update cascade
);

//створення таблиці student
create table student(
     student_id serial not null primary key,
     first_name varchar(40) not null,
     last_name varchar(40) not null,
     groups int not null,
```

```
     user int not null unique,
     foreign key (user) references user (user_id) on delete
restrict on update cascade,
     foreign key (group) references public.group (group_id) on
delete restrict on update cascade
);

//створення таблиці teacher_group_course
create table teacher_group_course(
     teacher_group_course_id serial not null primary key,
     teacher int not null,
     groups int not null,
     course int not null,
     foreign key (teacher) references teacher (teacher_id) on
delete restrict on update cascade,
     foreign key (groups) references public.group (group_id) on
delete restrict on update cascade,
     foreign key (course) references course (course_id) on
delete restrict on update cascade,
     UNIQUE (teacher, group, course)
);

//створення таблиці lesson
create table lesson(
     lesson_id serial not null primary key,
     teacher_group_course int not null,
     date_time timestamp not null,
     foreign key (teacher_group_course) references
teacher_group_course (teacher_group_course_id) on delete
restrict on update cascade
);

//створення таблиці consultation
create table consultation(
     consultation_id serial not null primary key,
     teacher_group_course int not null,
     date_time timestamp not null,
     foreign key (teacher_group_course) references
teacher_group_course (teacher_group_course_id) on delete
restrict on update cascade
);

//створення таблиці module
create table module(
     module_id serial not null primary key,
     course int not null,
     name varchar(80) not null,
     foreign key (course) references course (course_id) on
delete restrict on update cascade
);

//створення таблиці test
create table test(
```

```
     test_id serial not null primary key,
     name varchar(40) not null,
     description varchar(255) not null,
     time_limit int,
     max_mark int not null,
     parts_type varchar(20) not null check(parts_type in('Depend
of condition', 'Full test', 'First part'))
);

//створення таблиці test_part
create table test_part(
     test_part_id serial not null primary key,
     index varchar(40) not null,
     test varchar(255) not null,
     max_mark int not null,
     foreign key (test) references test (test_id) on delete
restrict on update cascade
);
//створення таблиці test_part_module
create table test_part_module(
     test_part_module_id serial not null primary key,
     test_part int not null,
     module int not null,
     foreign key (test_part) references test_part (test_part_id)
on delete restrict on update cascade,
     foreign key (module) references module (module_id) on
delete restrict on update cascade,
     UNIQUE (test_part, module)
);

//створення таблиці test_result
create table test_result(
     test_result_id serial not null primary key,
     test int not null,
     student int not null,
     mark int not null,
     exam int not null,
     foreign key (test) references test (test_id) on delete
restrict on update cascade,
     foreign key (exam) references test (test_id) on delete
restrict on update cascade,
     foreign key (student) references student (student_id) on
delete restrict on update cascade
);

//створення таблиці test_result_part
create table test_result_part(
     test_result_part_id serial not null primary key,
     test_part int not null,
     test_result int not null,
    mark int not null,
     foreign key (test_result) references test_result
(test_result_id) on delete restrict on update cascade,
```

```
     foreign key (test_part) references test_part (test_part_id)
on delete restrict on update cascade,
     UNIQUE (test_part, test_result);
);

//створення таблиці question
create table question(
     question_id serial not null primary key,
     content varchar(1024) not null,
     question_type varchar(6) not null check(question_type
in('closed', 'open')),
     closed_question_type varchar(12) check(closed_question_type
in('one answer', 'some answers', 'matrix', 'range')),
     score int,
     module int not null,
     foreign key (test_part_module) references test_part_module
(test_part_module_id) on delete restrict on update cascade
);

//створення таблиці matrix_question_li
create table matrix_question_li(
     matrix_question_li_id serial not null primary key,
     index int not null,
     content varchar(1024) not null,
     question int not null,
     foreign key (question) references question (question_id) on
delete restrict on update cascade
);

//створення таблиці answer
create table answer(
     answer_id serial not null primary key,
     question int not null,
     letter char(1) not null,
     content varchar(255) not null,
     status boolean,
     index int,
     score int not null,
     foreign key (question) references question (question_id) on
delete restrict on update cascade
);

//створення таблиці chosen_answer
create table chosen_answer(
     chosen_answer_id serial not null primary key,
     answer int,
     test_result_part int not null,
     index int,
     foreign key (answer) references answer (answer_id) on
delete restrict on update cascade,
     foreign key (test_result_part) references test_result_part
(test_result_part_id) on delete restrict on update cascade
);
```

```
//створення таблиці open_answer
CREATE TABLE open_answer(
     open_answer_id serial not null primary key,
     question int not null,
     test_result_part int not null,
     open_answer varchar(255),
     foreign key (question) references question (question_id) on
delete restrict on update cascade,
     foreign key (test_result_part) references test_result_part
(test_result_part_id) on delete restrict on update cascade
);

//створення таблиці lesson_consultation_duration
create table lesson_consultation_duration(
     lesson_consultation_duration_id int not null primary key,
     duration interval not null
);

//створення таблиці exam
create table exam(
     examt_id serial not null primary key,
     test int not null,
     teacher_group_course int not null,
     date_time timestamp not null,
     foreign key (test) references test (test_id) on delete
restrict on update cascade,
     foreign key (teacher_group_course) references
teacher_group_course (teacher_group_course_id) on delete
restrict on update cascade
);

//таблица test_part_module, курс должен біть одинаковым
SELECT test_part_module_id, test_part.test, test.course,
module.module_id, module.course, module.name
FROM test_part_module
JOIN test_part ON test_part.test_part_id =
test_part_module.test_part
JOIN test ON test.test_id = test_part.test
JOIN public.module ON module.module_id =
test_part_module.module;

//список вопросов для теста
Select question.question_id, question.content,
question.question_type, question.closed_question_type,
question.test_part_module, question.open_answer
from question
JOIN test_part_module on test_part_module.test_part_module_id =
question.test_part_module
JOIN test_part on test_part.test_part_id =
test_part_module.test_part
JOIN test on test.test_id = test_part.test
Where test_part.test=1;
```

```
//список ответов для вопроса
SELECT *
from answer
where question = 9;

//список подвопросов
SELECT *
from matrix_question_li
where question = 9;

INSERT INTO chosen_answer (answer, test_result_part, index,
open_answer)
VALUES ();

INSERT INTO open_answer (question, test_result_part,
open_answer)
VALUES (35, 2, 'answer 1'),
        (36, 2, 'answer 1'),
        (37, 2, 'answer 3');

//список правильных ответов на тест
Select question.question_id, answer_id, status, answer.index
from question
JOIN test_part_module on test_part_module.test_part_module_id =
question.test_part_module
JOIN test_part on test_part.test_part_id =
test_part_module.test_part
JOIN test on test.test_id = test_part.test
JOIN answer on answer.question = question.question_id
Where test_part.test=1 and (status=true or answer.index is not
null);

//проверка количества правильных ответов (1)
CREATE OR REPLACE FUNCTION answer_test()
RETURNS TRIGGER
AS $$
DECLARE
     answer_question integer;
     answer_question_type integer;
    true_answer_count integer;
CREATE OR REPLACE FUNCTION answer_test()
RETURNS TRIGGER
AS $$
DECLARE
     answer_question integer;
     answer_closed_question_type varchar(12);
    true_answer_count integer;
BEGIN
SELECT question
     INTO answer_question
     FROM answer
     WHERE answer_id=NEW.answer_id;
```

```
SELECT closed_question_type
     INTO answer_closed_question_type
     FROM question
     WHERE question_id=NEW.question;

SELECT COUNT(*)
     INTO true_answer_count
     FROM answer
     WHERE question = answer_question and status=true;

     IF (answer_closed_question_type='one answer') THEN
          IF (true_answer_count!=1) THEN
               RAISE EXCEPTION 'Wrong true answers number';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     ELSIF (answer_closed_question_type='some answers') THEN
          IF (true_answer_count<2) THEN
               RAISE EXCEPTION 'Wrong true answers number';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER answer_test_trigger
AFTER INSERT OR UPDATE
ON answer
FOR EACH ROW
EXECUTE PROCEDURE answer_test();

//проверка формулировки вопросов правильных ответов (2-3)
CREATE OR REPLACE FUNCTION answer_status_test()
RETURNS TRIGGER
AS $$
DECLARE
     matrix_answers_number integer;
     matrix_questions_number integer;
BEGIN
SELECT *
     INTO matrix_answers_number
     FROM answer
     WHERE question = NEW.question and NEW.index is not null;

SELECT *
     INTO matrix_questions_number
     FROM matrix_question_li
     WHERE question = NEW.question;
```

```
     IF (NEW.closed_question_type='one answer'
          or NEW.closed_question_type='some answers') THEN
          IF (NEW.status is null or NEW.index is not null) THEN
               RAISE EXCEPTION 'Stasus can not be null and index
must be null';
               RETURN NULL;
          ELSIF (NEW.status==false and score!=0) THEN
               RAISE EXCEPTION 'In false answers score must be
0';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
    ELSIF (NEW.closed_question_type='matrix') THEN
          IF (NEW.status is not null) THEN
               RAISE EXCEPTION 'Remove status from matrix
question answer';
               RETURN NULL;
          ELSIF (matrix_answers_number!=matrix_questions_number)
THEN
               RAISE EXCEPTION 'Wrong answer number for matrix
question';
               RETURN NULL;
          ELSIF (NEW.index==false and score!=0) THEN
               RAISE EXCEPTION 'In false answers score must be
0';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
    ELSIF (NEW.closed_question_type='range') THEN
          IF (NEW.status is not null or NEW.index is null) THEN
               RAISE EXCEPTION 'Stasus can not be not null and
index must be not null';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     ELSIF (NEW.closed_question_type='open answer') THEN
          IF (NEW.status is not null or NEW.index is not null)
THEN
               RAISE EXCEPTION 'Open answer has not got status
or index';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER answer_status_test_trigger
AFTER INSERT OR UPDATE
```

```
ON answer
FOR EACH ROW
EXECUTE PROCEDURE answer_status_test();

DECLARE
     matrix_answers_number integer;
     matrix_questions_number integer;
     answer_closed_question_type varchar(12);
     answer_question_type varchar(6);
BEGIN
SELECT COUNT(*)
     INTO matrix_answers_number
     FROM answer
     WHERE question = NEW.question and index is not null;

SELECT COUNT(*)
     INTO matrix_questions_number
     FROM matrix_question_li
     WHERE question = NEW.question;

SELECT closed_question_type, question_type
     INTO answer_closed_question_type, answer_question_type
     FROM question
     WHERE question_id = NEW.question;

     IF (answer_closed_question_type='one answer'
          or answer_closed_question_type='some answers') THEN
          IF (NEW.status is null or NEW.index is not null) THEN
               RAISE EXCEPTION 'Stasus can not be null and index
must be null';
               RETURN NULL;
          ELSIF (NEW.status is false and NEW.score!=0) THEN
               RAISE EXCEPTION 'In false answers score must be
0';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
    ELSIF (answer_closed_question_type='matrix') THEN
          IF (NEW.status is not null) THEN
               RAISE EXCEPTION 'Remove status from matrix
question answer';
               RETURN NULL;
          ELSIF (matrix_answers_number!=matrix_questions_number)
THEN
               RAISE EXCEPTION 'Wrong answer number for matrix
question % =! %', matrix_answers_number,
matrix_questions_number;
               RETURN NULL;
          ELSIF (NEW.index is null and NEW.score!=0) THEN
               RAISE EXCEPTION 'In false answers score must be
0';
               RETURN NULL;
```

```
          ELSE
               RETURN NEW;
          END IF;
    ELSIF (answer_closed_question_type='range') THEN
          IF (NEW.status is not null or NEW.index is null) THEN
               RAISE EXCEPTION 'Stasus can not be not null and
index must be not null';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     ELSIF (answer_question_type='open answer') THEN
          IF (NEW.status is not null or NEW.index is not null)
THEN
               RAISE EXCEPTION 'Open answer has not got status
or index';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
    END IF;
END;

update answer as t set
    status = c.status,
     score = c.score
from (values
    (1, true, 1),
    (3, false, 0)
) as c(answer_id, status, score)
where c.answer_id = t.answer_id;

CREATE OR REPLACE FUNCTION answers_number_test()
RETURNS TRIGGER
AS $$
DECLARE
     answers_number integer;
     matrix_subquestions_number integer;
     answer_closed_question_type varchar(12);
     matrix_answers_number integer;
BEGIN
SELECT COUNT(*)
     INTO answers_number
     FROM answer
     WHERE question = NEW.question;

SELECT COUNT(*)
     INTO matrix_subquestions_number
     FROM matrix_question_li
     WHERE question = NEW.question;

SELECT closed_question_type
     INTO answer_closed_question_type
```

```
     FROM question
     WHERE question_id = NEW.question;

SELECT COUNT(*)
     INTO matrix_answers_number
     FROM answer
     WHERE question = NEW.question and index is not null;

     IF (answer_closed_question_type='one answer' ) THEN
          IF (answers_number<5) THEN
               RAISE EXCEPTION 'Min variants number of question
with one answer is 5';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     ELSIF (answer_closed_question_type='some answers') THEN
          IF (answers_number<4) THEN
               RAISE EXCEPTION 'Min variants number of question
with some answers is 4';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
    ELSIF (answer_closed_question_type='matrix') THEN
          IF (matrix_subquestions_number<2 and
matrix_answers_number<3
               and (matrix_answers_number-
matrix_subquestions_number)>0) THEN
               RAISE EXCEPTION 'Min variants number of matrix
question is 3';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
    ELSIF (answer_closed_question_type='range') THEN
          IF (answers_number<3) THEN
               RAISE EXCEPTION 'Min variants number of range
question is 3';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     ELSE
          RETURN NEW;
     END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER answers_number_test_trigger
AFTER INSERT OR UPDATE
ON answer
FOR EACH ROW
```

```
EXECUTE PROCEDURE answers_number_test();

CREATE OR REPLACE FUNCTION index_test()
RETURNS TRIGGER
AS $$
DECLARE
     answer_closed_question_type varchar(12);
BEGIN

SELECT closed_question_type
     INTO answer_closed_question_type
     FROM question
     WHERE question_id = NEW.question;

    IF (answer_closed_question_type='matrix' or
answer_closed_question_type='range') THEN
          IF (new.index in (SELECT index FROM answer where
question = new.question)) THEN
               RAISE EXCEPTION 'This index already exists';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     ELSE
          RETURN NEW;
     END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER index_test_trigger
AFTER DELETE OR UPDATE
ON answer
FOR EACH ROW
EXECUTE PROCEDURE index_test();

CREATE OR REPLACE FUNCTION question_status_test()
RETURNS TRIGGER
AS $$
DECLARE
     matrix_answers_number integer;
     matrix_questions_number integer;
BEGIN
     IF (NEW.question_type='open') THEN
          IF (NEW.closed_question_type is not null) THEN
               RAISE EXCEPTION 'There are no variants type for
open question';
               RETURN NULL;
          ELSIF (NEW.open_answer is null) THEN
               RAISE EXCEPTION 'There must be open answer for
open question';
               RETURN NULL;
          ELSE
               RETURN NEW;
```

```
          END IF;
     ELSE
          IF (NEW.closed_question_type is null) THEN
               RAISE EXCEPTION 'There must be variants type for
closed question';
               RETURN NULL;
          ELSIF (NEW.open_answer is not null) THEN
               RAISE EXCEPTION 'There are no open answer for
closed question';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER question_status_test_trigger
AFTER INSERT OR UPDATE
ON question
FOR EACH ROW
EXECUTE PROCEDURE question_status_test();

CREATE OR REPLACE FUNCTION question_status_test()
RETURNS TRIGGER
AS $$
DECLARE
     matrix_answers_number integer;
     matrix_questions_number integer;
BEGIN
     IF (NEW.question_type='open') THEN
          IF (NEW.closed_question_type is not null) THEN
               RAISE EXCEPTION 'There are no variants type for
open question';
               RETURN NULL;
          ELSIF (NEW.open_answer is null) THEN
               RAISE EXCEPTION 'There must be open answer for
open question';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     ELSE
          IF (NEW.closed_question_type is null) THEN
               RAISE EXCEPTION 'There must be variants type for
closed question';
               RETURN NULL;
          ELSIF (NEW.open_answer is not null) THEN
               RAISE EXCEPTION 'There are no open answer for
closed question';
               RETURN NULL;
          ELSE
               RETURN NEW;
```

```
          END IF;
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER question_status_test_trigger
AFTER INSERT OR UPDATE
ON question
FOR EACH ROW
EXECUTE PROCEDURE question_status_test();

CREATE OR REPLACE FUNCTION matrix_question_li_index_test()
RETURNS TRIGGER
AS $$
BEGIN
          IF (new.index in (SELECT index FROM matrix_question_li
where question = new.question)) THEN
               RAISE EXCEPTION 'This index already exists';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER matrix_question_li_index_test_trigger
AFTER INSERT OR UPDATE
ON matrix_question_li
FOR EACH ROW
EXECUTE PROCEDURE matrix_question_li_index_test();

CREATE OR REPLACE FUNCTION letter_test()
RETURNS TRIGGER
AS $$
BEGIN
          IF (new.letter in (SELECT index FROM answer where
question = new.question)) THEN
               RAISE EXCEPTION 'This variant already exists';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER letter_test_trigger
AFTER INSERT OR UPDATE
ON answer
FOR EACH ROW
EXECUTE PROCEDURE letter_test();

CREATE OR REPLACE FUNCTION test_module_course_test()
RETURNS TRIGGER
```

```
AS $$
DECLARE
     module_course integer;
     test_course integer;
BEGIN
     SELECT module.course, test.course
     INTO module_course, test_course
     from test_part_module
     JOIN test_part on test_part_module.test_part =
test_part.test_part_id
     JOIN test on test_part.test = test.test_id
     JOIN module on test_part_module.module = module.module_id
     where
test_part_module.test_part_module_id=NEW.test_part_module;

     IF (module_course!=test_course) THEN
          RAISE EXCEPTION 'This module from other course';
          RETURN NULL;
     ELSE
          RETURN NEW;
     END IF;

END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER test_module_course_test_trigger
AFTER INSERT OR UPDATE
ON answer
FOR EACH ROW
EXECUTE PROCEDURE test_module_course_test();

CREATE OR REPLACE FUNCTION test_module_course_test()
RETURNS TRIGGER
AS $$
DECLARE
     module_course integer;
     test_course integer;
BEGIN
     SELECT module.course, test.course
     INTO module_course, test_course
     from question
     JOIN test_part on question.test_part =
test_part.test_part_id
     JOIN test on test_part.test = test.test_id
     JOIN module on question.module = module.module_id
     where question.question_id=NEW.question_id;

     IF (module_course!=test_course) THEN
          RAISE EXCEPTION 'This module from other course';
          RETURN NULL;
     ELSE
          RETURN NEW;
     END IF;
```

```
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER test_module_course_test_trigger
AFTER INSERT OR UPDATE
ON question
FOR EACH ROW
EXECUTE PROCEDURE test_module_course_test();

CREATE OR REPLACE FUNCTION test_part_test()
RETURNS TRIGGER
AS $$
DECLARE
     test_part_index varchar(12);
     test_part int;
BEGIN
SELECT test_part.index
INTO test_part_index
FROM question
JOIN test_part on question.test_part = test_part.test_part_id;

     IF (NEW.question_type='closed' ) THEN
          IF (test_part_index='Second') THEN
               RAISE EXCEPTION 'Enter test part with closed
questions';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     ELSIF (NEW.question_type='open') THEN
          IF (test_part_index='First') THEN
               RAISE EXCEPTION 'Enter test part with open
questions';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     ELSE
          RETURN NEW;
     END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER test_part_test_trigger
AFTER INSERT OR UPDATE
ON question
FOR EACH ROW
EXECUTE PROCEDURE test_part_test();

CREATE OR REPLACE FUNCTION question_number_test()
RETURNS TRIGGER
AS $$
```

```
DECLARE
     questions_number integer;
     test_part_index varchar(6);
BEGIN
SELECT count(*)
INTO questions_number
FROM question_test_part
WHERE question_test_part_id=new.question_test_part_id
GROUP BY test_part;

SELECT test_part_index
INTO test_part_index
FROM test_part
WHERE test_part_id=new.test_part
GROUP BY test_part;

     IF(test_part_index='First') THEN
          IF (questions_number<5) THEN
               RAISE EXCEPTION 'Min number of question for test
by one module is 5';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     ELSE
          RETURN NEW;
     END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER question_number_test_trigger
AFTER INSERT OR UPDATE
ON question_test_part
FOR EACH ROW
EXECUTE PROCEDURE question_number_test();

//оцінка за питання
CREATE OR REPLACE FUNCTION question_score(question_id integer)
RETURNS integer
AS $$
DECLARE
     question_score integer;
BEGIN
     SELECT SUM(score)
     INTO question_score
     FROM answer
     WHERE question=question_id;

    RETURN question_score;
END;
$$ LANGUAGE plpgsql;

//оцінка за першу частину тесту
```

```
CREATE OR REPLACE FUNCTION first_test_part_score(test_id
integer)
RETURNS integer
AS $$
DECLARE
     first_test_part_score integer;
BEGIN
     SELECT SUM(question_score(question_id))
     INTO first_test_part_score
     FROM question
     JOIN question_test_part on
question_test_part.question=question.question_id
     JOIN test_part on
question_test_part.test_part=test_part.test_part_id
     WHERE test_part.test=test_id and test_part.index='First';


    RETURN first_test_part_score;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION second_test_part_score(test_id
integer)
RETURNS integer
AS $$
DECLARE
     second_test_part_score integer;
BEGIN
     SELECT SUM(question.score)
     INTO second_test_part_score
     FROM question
     JOIN question_test_part on
question_test_part.question=question.question_id
     JOIN test_part on
question_test_part.test_part=test_part.test_part_id
     where test_part.test=test_id and test_part.index='Second';

    RETURN second_test_part_score;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION parts_type_test()
RETURNS TRIGGER
AS $$
DECLARE
     parts_type varchar(20);
BEGIN
SELECT parts_type
INTO parts_type
FROM test
WHERE test_id=new.test;

     IF(parts_type='First part') THEN
```

```
          IF (new.index='Second') THEN
               RAISE EXCEPTION 'There are no second part for
this test';
               RETURN NULL;
          ELSE
               RETURN NEW;
          END IF;
     ELSE
          RETURN NEW;
     END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER parts_type_test_trigger
AFTER INSERT OR UPDATE
ON test_part
FOR EACH ROW
EXECUTE PROCEDURE parts_type_test();

CREATE OR REPLACE FUNCTION
test_result_question_score(test_result_part_id integer,
question_id integer)
RETURNS boolean
AS $$
DECLARE
     test_result_question_score integer;
BEGIN
     select sum(score)
     INTO test_result_question_score
     from chosen_answer
     join answer on chosen_answer.answer=answer.answer_id
     where test_result_part=test_result_part_id and
answer.question=question_id and (status is true or (status is
null and answer.index=chosen_answer.index));

    RETURN test_result_question_score;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION
is_question_answered(test_result_part_id integer, question_id
integer)
RETURNS boolean
AS $$
DECLARE
     test_result_question_score integer;
BEGIN
     IF(test_result_question_score(test_result_part_id,
question_id)!=question_score(question_id)) THEN
          RETURN FALSE;
    ELSE
          RETURN TRUE;
     END IF;
```

```
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION
answered_percent(test_result_part_identifier integer)
RETURNS numeric(4,1)
AS $$
DECLARE
     answered_percent numeric(4,2);
     question_count integer;
     answered_question_count integer;

BEGIN
SELECT COUNT(DISTINCT question_id)
INTO answered_question_count
FROM question
JOIN answer on answer.question=question.question_id
JOIN chosen_answer on chosen_answer.answer=answer.answer_id
WHERE test_result_part=test_result_part_identifier and
is_question_answered(test_result_part_identifier, question);

SELECT COUNT(DISTINCT question)
INTO question_count
FROM question_test_part
JOIN test_result_part on
question_test_part.test_part=test_result_part.test_part
WHERE
test_result_part.test_result_part_id=test_result_part_identifier
;

    IF question_count = 0 THEN
        RAISE EXCEPTION 'Division by zero is not allowed';
     ELSE
          RETURN (answered_question_count * 100) /
question_count;
     END IF;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION
test_result_open_question_score(test_result_part_id integer)
RETURNS integer
AS $$
DECLARE
     open_question_score integer;
BEGIN
     select sum(score)
     INTO open_question_score
     from open_answer
     join question on question.question_id =
open_answer.question
     where test_result_part=test_result_part_id and
open_answer.open_answer=question.open_answer;
```

```
     IF open_question_score is NULL THEN
          RETURN 0;
     ELSE
          RETURN open_question_score;
     END IF;
END;
$$ LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION
closed_questions_score(test_result_part_id integer)
RETURNS integer
AS $$
DECLARE
     closed_questions_score integer;
BEGIN
     select sum(score)
     INTO closed_questions_score
     from chosen_answer
     join answer on chosen_answer.answer=answer.answer_id
     where test_result_part=1 and (status is true or (status is
null and answer.index=chosen_answer.index));

     IF closed_questions_score is NULL THEN
          RETURN 0;
     ELSE
          RETURN closed_questions_score;
     END IF;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION
test_result_score(test_result_identifier integer)
RETURNS numeric(4,2)
AS $$

DECLARE
     test_result_parts_type varchar(20);
     test_result_first_part integer;
     test_result_second_part integer;
     test_first_part integer;
     test_second_part integer;
     result_mark numeric;
     full_mark numeric;
     final_mark numeric;
BEGIN
     SELECT parts_type
     INTO test_result_parts_type
     from test_result
     join test on test_result.test = test.test_id
     where test_result_id=test_result_identifier;
```

```
     RAISE NOTICE 'test_result_parts_type: %',
test_result_parts_type;

     SELECT test_result_part_id, test_part
     INTO test_result_first_part, test_first_part
     from test_result_part
     join test_part on test_result_part.test_part =
test_part.test_part_id
     where test_result=test_result_identifier and
test_part.index='First';

     RAISE NOTICE 'test_result_first_part: %',
test_result_first_part;

     SELECT test_result_part_id, test_part
     INTO test_result_second_part, test_second_part
     from test_result_part
     join test_part on test_result_part.test_part =
test_part.test_part_id
     where test_result=test_result_identifier and
test_part.index='Second';

     RAISE NOTICE 'test_result_second_part: %',
test_result_second_part;

     RAISE NOTICE 'closed_questions_score : %',
test_result_closed_questions_score(test_result_first_part);
     RAISE NOTICE 'open_questions_score : %',
test_result_open_questions_score(test_result_second_part);
     RAISE NOTICE 'first_test_part_score: %',
first_test_part_score(test_first_part);
     RAISE NOTICE 'second_test_part_score : %',
second_test_part_score(test_second_part);

     result_mark :=
ROUND(((test_result_closed_questions_score(test_result_first_par
t) + test_result_open_questions_score(test_result_second_part))
* first_test_part_score(test_first_part))::numeric, 2);
     RAISE NOTICE 'result_mark : %', result_mark;
     full_mark
:=  ROUND((first_test_part_score(test_first_part)+second_test_pa
rt_score(test_second_part))::numeric, 2);
     RAISE NOTICE 'full_mark : %', full_mark;
     final_mark :=  ROUND(result_mark/full_mark, 0);
     RAISE NOTICE 'final_mark : %', final_mark;

     IF test_result_parts_type='First part' THEN
          RETURN
test_result_closed_questions_score(test_result_first_part);
     ELSIF test_result_parts_type='Depend of condition' THEN
          IF answered_percent(test_result_first_part)<75 THEN
               RETURN final_mark;
          ELSE
```

```
              RETURN
test_result_closed_questions_score(test_result_first_part);
          END IF;
     ELSE
          RETURN final_mark;
     END IF;
END;

$$ LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION get_user_id(login_param varchar(20),
password_param varchar(12))
RETURNS integer AS $$
DECLARE
    userId integer;
BEGIN
    SELECT user_id INTO userId
    FROM "user"
    WHERE login = login_param AND password = password_param;

    IF userId IS NULL THEN
        RAISE EXCEPTION 'Login or password are wrong!';
    END IF;

    RETURN userId;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get_user_role(userId integer)
RETURNS varchar(8) AS $$
DECLARE
    role varchar(8);
    directorId integer;
    teacherId integer;
     studentId integer;
BEGIN
    SELECT director_id INTO directorId
    FROM director
    WHERE user = userId;

    SELECT teacher_id INTO teacherId
    FROM teacher
    WHERE user = userId;

    SELECT student_id INTO studentId
    FROM student
    WHERE user = userId;

    IF directorId IS NOT NULL THEN
          RETURN 'director';
     ELSIF teacherId IS NOT NULL THEN
          RETURN 'teacher';
```

```
     ELSIF studentId IS NOT NULL THEN
          RETURN 'student';
     ELSE
        RAISE EXCEPTION 'Continue to sign up';
          RETURN NULL;
    END IF;
END;
$$ LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION get_user_role_id(userId integer)
RETURNS integer AS $$
DECLARE
    directorId integer;
    teacherId integer;
     studentId integer;
BEGIN
    SELECT director_id INTO directorId
    FROM director
    WHERE user = userId;

    SELECT teacher_id INTO teacherId
    FROM teacher
    WHERE user = userId;

    SELECT student_id INTO studentId
    FROM student
    WHERE user = userId;

    IF directorId IS NOT NULL THEN
          RETURN directorId;
     ELSIF teacherId IS NOT NULL THEN
          RETURN teacherId;
     ELSIF studentId IS NOT NULL THEN
          RETURN studentId;
     ELSE
        RAISE EXCEPTION 'Continue to sign up';
          RETURN NULL;
    END IF;
END;
$$ LANGUAGE plpgsql;
```