МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОГО ФОРМУВАННЯ ТЕСТІВ УСПІШНОСТІ»**

Виконав: студент 4 курсу, групи 6.1220
спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми комп'ютерні науки
(назва освітньої програми)
І.В. Коломійцев
(ініціали та прізвище)
Керівник старший викладач кафедри комп'ютерних наук Циммерман Г.А..
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри програмної інженерії ЗНУ, к.ф.-м.н., доцент Кудін О.В
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 комп'ютерні науки
(шифр і назва)

Освітня програма комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Чопоров С.В.
(підпис)

" ____ " _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Коломійцеву Іллі Вікторовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка системи автоматизованого формування тестів успішності навчання

керівник роботи Циммерман Г.А., старший викладач кафедри комп'ютерних наук
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2024 року № 2180-с

2. Строк подання студентом роботи 05.06.2024

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Вступ, Огляд наукових праць, Постановка завдань дослідження, Аналіз можливих методів вирішення проблеми, Створення проєкту, Демонстрація.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 25.12.2023

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	15.01.2024	
2.	Збір вихідних даних.	07.02.2024	
3.	Обробка методичних та теоретичних	08.03.2024	
	джерел.		
4.	Розробка першого та другого розділу.	10.04.2024	
5.	Розробка третього розділу.	03.05.204	
6.	Оформлення та нормоконтроль	15.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	18.06.2024	

Студент _____ І.В. Коломійцев
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.Г. Циммерман
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка системи автоматизованого формування тестів успішності навчання»: 67 с., 22 рис., 0 табл., 6 джерел, 0 додатків.

Адаптивне тестування, Аналіз результатів, Автоматизація, Веб-додаток, Генерація тестів, Інтерфейс користувача, Індивідуалізація навчання, Машинне навчання, Питання з рівнями складності, Персоналізоване оцінювання, Прогнозування рівня знань, Розробка системи, Рівні складності.

Об’єкт дослідження – технології автоматизації процесу оцінювання знань у навчальному процесі. Це включає в себе методи та алгоритми для створення тестових завдань, адаптації тестів до рівня знань студентів та аналізу результатів тестування.

Мета роботи: Проектування та розробка системи для автоматизованого формування тестів, яка допомагає: генерувати тестові завдання відповідно до індивідуального рівня знань студентів, адаптувати складність питань на основі результатів попередніх тестувань, спрощувати процес управління тестами для викладачів, забезпечуючи ефективний та персоналізований підхід до оцінювання знань.

Метод дослідження – Аналіз та синтез: Файл generate_test.py містить функцію generate_test, яка вибирає питання на основі правил і рівня знань. Моделювання: Файл generate_test.py включає функцію predict_student_level, яка використовує модель машинного навчання для прогнозування рівня складності. Експериментальний метод: Файл analyze_test_results.py забезпечує аналіз результатів тестування та коригування рівня складності питань. Комп'ютерне моделювання: Всі компоненти коду, включаючи app.py, generate_test.py, та analyze_test_results.py, інтегровані для забезпечення комплексного підходу до автоматизації формування тестів.

SUMMARY

Master’s qualifying paper « Development of the system for test

automated generation»: 67 pages, 22 figures, 0 tables, 6 references, 0 supplements.

Adaptive Testing, Results Analysis, Automation, Web Application, Test Generation, User Interface, Personalized Learning, Machine Learning, Questions with Difficulty Levels, Personalized Assessment, Knowledge Level Prediction, System Development, Difficulty Levels.

Object of the study – automation technologies for the knowledge assessment process in the educational process. This includes methods and algorithms for creating test tasks, adapting tests to the students' knowledge levels, and analyzing test results.

Aim of the study: Designing and developing a system for automated test generation, which helps to: generate test tasks according to the individual level of students' knowledge, adapt the complexity of questions based on previous test results, simplify the test management process for educators, ensuring an effective and personalized approach to knowledge assessment.

Methods of research – Analysis and synthesis: The generate_test.py file contains the generate_test function, which selects questions based on rules and knowledge levels.

Modeling: The generate_test.py file includes the predict_student_level function, which uses a machine learning model to predict the difficulty level.

Experimental method: The analyze_test_results.py file provides the analysis of test results and the adjustment of question difficulty levels.

Computer modeling: All components of the code, including app.py, generate_test.py, and analyze_test_results.py, are integrated to provide a comprehensive approach to automated test generation.

ЗМІСТ

ВСТУП

У світі сучасних технологій та цифрового навчання, ефективні інструменти тестування стають все більш важливими для освітнього процесу. Одним з таких інструментів є адаптивне тестування, яке підлаштовується під рівень знань студента і допомагає виявити прогалини у навчанні. Застосування таких систем дозволяє не лише оцінити поточний рівень знань учня, але й стимулювати його до постійного самовдосконалення.

Системи автоматизованого тестування спрямовані на оптимізацію процесу навчання та оцінювання в освітніх установах. Вони забезпечують ефективність для вчителів та зручність для студентів, дозволяючи обом сторонам зосередитися на покращенні знань та навичок. Автоматизація тестування допомагає вчителям швидко створювати тести, оцінювати результати та надавати зворотний зв’язок студентам. Це не тільки скорочує адміністративні витрати, але й знижує навантаження на викладачів, дозволяючи їм більше часу приділяти навчанню.

Для студентів автоматизовані тести пропонують адаптивний і персоналізований підхід. Система автоматично коригує рівень складності тестів на основі успішності студента, що допомагає підтримувати мотивацію і підвищувати ефективність навчання.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

GPU: Graphics Processing Unit

API: Application Programming Interface

SQL: Structured Query Language

HTML: HyperText Markup Language

CSS: Cascading Style Sheets

URL: Uniform Resource Locator

HTTP: HyperText Transfer Protocol

Py: Python

1 ОГЛЯД НАУКОВИХ ПРАЦЬ

1.1.1 Огляд вітчизняних публікацій

Проблеми перевірки результатів навчання є актуальними через потребу в ефективних та об'єктивних методах оцінювання знань студентів. Зростання кількості навчальних матеріалів і різноманітність освітніх підходів вимагають впровадження нових технологій для забезпечення якісного оцінювання. Особливо цікавою є проблема автоматизації такої перевірки, оскільки вона дозволяє знизити людський фактор і збільшити точність оцінювання. Про це свідчать наступні публікації:

1) Білоус В.В., Сидоренко В.К. "Методологія розробки систем автоматизованого тестування". (2020)

– у цій роботі автори розглядають методи розробки систем, що автоматизують процес формування тестів. Основний акцент зроблено на створенні алгоритмів для адаптивного тестування, яке враховує індивідуальні особливості студентів.

– використовуються моделі, що базуються на об’єктивному аналізі знань, отриманих під час навчання, і на їх основі будуються тестові завдання.

2) Кравченко О.М., Литвиненко А.С. "Система автоматизованого оцінювання знань на основі комп’ютерних тестів". (2019)

– у цій статті розглядається побудова системи автоматизованого тестування, яка використовує методи штучного інтелекту для створення тестових завдань різного рівня складності.

– автори пропонують алгоритм автоматичного генерування тестових завдань з врахуванням попередніх результатів навчання студентів.

3) Мартинюк І.В., Павлюк В.М. "Застосування нейронних мереж у системах автоматизованого тестування". (2021)

– у даній роботі досліджується використання нейронних мереж для формування тестів з адаптивними характеристиками.

– авторами запропонована методика інтеграції нейромережевих моделей у систему тестування, що дозволяє підвищити якість оцінювання знань.

1.1.2 Огляд зарубіжних публікацій

Не менш цікаві думки стосовно зазначених проблем висунули наступні зарубіжні автори:

1) Kim, S. & Lee, Y. "Automated Test Generation for Educational Assessment". Educational Technology Research and Development (2020)

– стаття розглядає методи автоматизованого формування тестових завдань на основі моделей машинного навчання, які аналізують великі обсяги освітніх даних.

– авторами запропоновано систему, що генерує адаптивні тести з урахуванням рівня знань студентів та їхньої успішності у навчанні.

2) Johnson, L., Beck, R. "Artificial Intelligence in Computer-Based Testing". Journal of Educational Computing Research (2019)

– у роботі розглядається використання технологій штучного інтелекту для створення динамічних тестових завдань, які автоматично підлаштовуються під потреби користувача.

– автори обговорюють перспективи та виклики використання AI у тестуванні, зокрема питання надійності та безпеки даних.

3) Gierl, M.J., Lai, H. "Using Automatic Item Generation to Create Multiple-Choice Test Items". International Journal of Testing (2018)

– дослідження зосереджено на автоматичному створенні тестових завдань за допомогою технології Automatic Item Generation (AIG), що дозволяє швидко генерувати питання на основі заданих параметрів.

– запропонована система дає змогу зменшити трудомісткість процесу створення тестів і підвищити їхню якість.

1.1.3 Аналіз стану рішення проблеми

Проблема автоматизованого тестування в освіті ще не вирішена повністю, і тому важливо аналізувати її особливості, переваги та виклики. Автоматизація процесу оцінювання знань є ключовою темою досліджень і розробок у сучасних освітніх технологіях.

Переваги:

– адаптивність: Системи автоматизованого тестування дозволяють створювати адаптивні тести, що підлаштовуються під рівень знань кожного студента.

– ефективність: Автоматизація процесу створення тестів значно зменшує витрати часу і ресурсів на розробку тестових завдань.

– масштабованість: Сучасні системи можуть обробляти велику кількість студентів одночасно, що є ключовим для великих навчальних закладів.

Виклики:

– надійність: Забезпечення високої якості тестових завдань та їх відповідності навчальним цілям є постійним викликом.

– безпека: Зберігання та обробка персональних даних студентів потребує забезпечення високого рівня захисту.

– технологічні обмеження: Інтеграція нових технологій, таких як AI і машинне навчання, може вимагати значних ресурсів та спеціалізованих знань.

1.1.4 Вибір та обґрунтування напрямів дослідження

Таким чином, можна сформулювати напрямки досліджень на перспективу:

– Інтеграція штучного інтелекту у системи тестування: Вивчення можливостей використання AI для автоматизації створення більш складних та адаптивних тестових завдань.

– Розробка моделей адаптивного тестування: Дослідження методів адаптації тестів до індивідуальних потреб студентів для підвищення точності оцінки їхніх знань.

– Безпека та конфіденційність даних: Вивчення підходів до забезпечення безпеки та захисту даних студентів у системах автоматизованого тестування.

– Міждисциплінарний підхід: Поєднання знань з галузей педагогіки, психології та інформаційних технологій для розробки комплексних рішень у сфері автоматизованого тестування.

1.1.5 Висновок

Автоматизація процесу формування тестів є важливим кроком у розвитку сучасної освіти. Розробка таких систем сприяє ефективному оцінюванню знань студентів, адаптації навчальних процесів до їх індивідуальних потреб та забезпечує економію часу і ресурсів. Враховуючи значний науковий інтерес та технологічні досягнення у цій галузі, подальші дослідження мають зосередитися на інтеграції новітніх технологій, таких як штучний інтелект, та вирішенні питань безпеки даних.

2 ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

2.1.1 Формулювання проблеми

В умовах стрімкого розвитку інформаційних технологій та зростаючих вимог до якості освіти, питання автоматизації навчального процесу стає все більш актуальним. Однією з ключових складових ефективного навчання є регулярне оцінювання знань студентів, що дозволяє своєчасно виявляти прогалини в знаннях і коригувати навчальний процес. У традиційних методах оцінювання, таких як письмові іспити або контрольні роботи, існують значні обмеження, пов’язані з великими витратами часу на підготовку і перевірку тестів, суб'єктивністю оцінювання, а також складністю адаптації тестів до індивідуальних потреб студентів.

2.1.2 Основні проблеми

1) Трудомісткість та часозатратність створення тестів

Створення якісних тестових завдань є складним і трудомістким процесом. Викладачі витрачають значну кількість часу на підготовку питань, які повинні відповідати програмі курсу, бути збалансованими за рівнем складності і різноманітними за форматом. В умовах великих навчальних груп або онлайн-курсів проблема множиться, оскільки потрібно підготувати значну кількість варіантів тестів для запобігання списуванню.

2) Необхідність адаптивного тестування

Сучасна освіта все більше орієнтується на індивідуальний підхід до навчання. Однак, стандартні тестові завдання часто не враховують різницю у рівнях знань та здібностях студентів. Це призводить до ситуації, коли однакові тестові питання можуть бути занадто легкими для одних і надто складними для інших. Відсутність механізму адаптивного тестування обмежує можливість коректного оцінювання і не дозволяє створити максимально ефективне навчальне середовище.

3) Підвищення об’єктивності та якості оцінювання

Традиційні методи оцінювання часто супроводжуються суб'єктивністю у виставленні оцінок. Від цього страждає об'єктивність оцінювання, що може негативно вплинути на мотивацію студентів. Також, існує потреба у формуванні питань, які б оцінювали не лише запам'ятовування фактів, але й здатність студентів до аналітичного мислення і застосування знань у різних ситуаціях.

4) Інтеграція новітніх технологій в освітній процес

Інноваційні технології, такі як штучний інтелект і машинне навчання, відкривають нові можливості для створення інтелектуальних систем автоматизованого тестування. Проте, їх інтеграція в освітній процес вимагає значних ресурсів і знань. Більшість навчальних закладів стикаються з труднощами при впровадженні таких систем через технічні та організаційні обмеження.

2.1.3 Мета та завдання дослідження

Метою дослідження є розробка системи автоматизованого формування тестів, яка вирішує вищезазначені проблеми. Система повинна забезпечувати:

– автоматизацію процесу створення тестів: Зменшення часу та зусиль, необхідних для підготовки тестових завдань.

– адаптивне тестування: Можливість створення індивідуальних тестів, що враховують рівень знань кожного студента.

– підвищення об'єктивності оцінювання: Використання алгоритмів, які мінімізують суб'єктивність при оцінюванні і забезпечують високу якість тестових питань.

– Інтеграцію сучасних технологій: Впровадження новітніх технологій, таких як штучний інтелект і машинне навчання, для покращення ефективності та функціональності системи.

2.1.4 Очікувані результати

Розробка системи автоматизованого формування тестів має призвести до підвищення ефективності і якості навчального процесу за рахунок:

– скорочення часу на підготовку і проведення тестувань.

– підвищення точності і об'єктивності оцінювання знань студентів.

– забезпечення індивідуального підходу до навчання та оцінювання.

– сприяння інтеграції інноваційних технологій у сферу освіти.

3 АНАЛІЗ МОЖЛИВИХ МЕТОДІВ ВИРІШЕННЯ ПРОБЛЕМИ

3.1.1 Огляд методів автоматизації формування тестів

В рамках створення системи автоматизованого формування тестів успішності навчання існує кілька підходів, кожен з яких має свої переваги і недоліки. Основними методами є генерація тестів на основі правил, автоматичне створення питань за допомогою алгоритмів машинного навчання, а також адаптивне тестування.

1) Генерація тестів на основі правил

Опис:

– цей підхід базується на використанні заздалегідь визначених правил і шаблонів для створення тестових завдань. В правилах вказується, які типи питань повинні бути включені в тест, рівні складності, тематика та інші параметри.
– правила можуть бути налаштовані для кожного курса або навчальної програми, що дозволяє адаптувати тести до конкретних навчальних цілей.

Переваги:

– простота реалізації та легкість налаштування.
– висока передбачуваність і контроль над кінцевим результатом.
– можливість швидкого налаштування тестів під різні навчальні курси та теми.

Недоліки:

– обмежена гнучкість, оскільки тести формуються відповідно до фіксованих правил.
– складність у врахуванні індивідуальних особливостей студентів.

2) Автоматичне створення питань за допомогою алгоритмів машинного навчання

Опис:

– використання алгоритмів машинного навчання (ML) для створення тестових завдань. Ці алгоритми аналізують великий обсяг навчальних даних і генерують питання на основі патернів, виявлених у цих даних.
– моделі, як-от генеративні нейронні мережі, можуть бути використані для створення різноманітних типів питань, включаючи багатоваріантні та відкриті питання.

Переваги:

– висока гнучкість та можливість створення різноманітних питань.
– алгоритми можуть адаптуватися до нових даних і навчатися створювати більш релевантні питання з часом.
– підтримка створення питань, які виходять за межі простого запам’ятовування фактів, і фокусуються на аналітичному мисленні та застосуванні знань.

Недоліки:

– висока складність реалізації та потреба у значних обчислювальних ресурсах.
– необхідність у великих обсягах даних для тренування моделей.
– потенційні труднощі з контролем якості згенерованих питань.

3) Адаптивне тестування

Опис:

– адаптивне тестування (Adaptive Testing) - це метод, який підлаштовується під рівень знань студента в реальному часі. Після

кожної відповіді система визначає, які питання задати далі, на основі відповідей студента.
– цей підхід може використовувати алгоритми, що коригують складність і тематику питань відповідно до прогресу студента.

Переваги:
– індивідуалізація процесу тестування, що дозволяє точно оцінити рівень знань кожного студента.
– зменшення кількості питань, необхідних для точного визначення рівня знань.
– підвищення мотивації студентів через оптимальний рівень викликів.

Недоліки:
– висока складність розробки та реалізації алгоритмів адаптації.
– потреба у постійному моніторингу та коригуванні алгоритмів для підтримки їхньої ефективності.
– необхідність інтеграції з системами управління навчанням для збору даних про прогрес студентів.

3.1.2 Технологічні рішення для реалізації

Після аналізу методів автоматизації формування тестів необхідно вибрати технологічну базу, яка дозволить реалізувати обрані підходи. Основні технологічні рішення можуть включати використання сучасних фреймворків для розробки вебдодатків, платформ для машинного навчання, а також інтеграційних технологій для забезпечення зручної роботи системи.

1) Вебфреймворки

Опис:

– вебфреймворки, такі як Django, Flask (Python), Spring Boot (Java), або ASP.NET (C), дозволяють створювати потужні вебзастосунки, що підтримують інтерактивну роботу з користувачами.
– ці фреймворки надають можливості для швидкої розробки, підтримки безпеки, і масштабованості.

Переваги:
– легкість інтеграції з базами даних та іншими сервісами.
– підтримка широкого спектру бібліотек і модулів для розширення функціональності.
– велике співтовариство та наявність великої кількості навчальних матеріалів.

Недоліки:
– може вимагати значного часу на освоєння для новачків.
– відносно висока складність налаштування середовища розробки.

2) Платформи машинного навчання

Опис:
– платформи для машинного навчання, такі як TensorFlow, PyTorch, або scikit-learn, дозволяють розробляти, тренувати та впроваджувати моделі машинного навчання.
– вони підтримують широкий спектр алгоритмів і моделей, які можуть бути використані для створення і вдосконалення тестових завдань.

Переваги:
– підтримка широкого спектра інструментів і бібліотек для розробки моделей.
– можливість обробки великих обсягів даних і проведення складних аналітичних операцій.

– висока продуктивність і можливість використання графічних процесорів (GPU) для прискорення обчислень.

Недоліки:

– висока складність налаштування і використання для новачків.
– потреба у великих обсягах даних для тренування ефективних моделей.

3) Інтеграційні технології

Опис:

– інтеграційні технології, такі як API, Web Services або message brokers (наприклад, RabbitMQ, Kafka), дозволяють забезпечити взаємодію між різними компонентами системи та зовнішніми сервісами.
– вони можуть використовуватися для інтеграції системи з навчальними платформами, базами даних, та іншими додатками.

Переваги:

– забезпечення безшовної інтеграції з іншими системами та сервісами.
– підвищення гнучкості та можливостей масштабування системи.
– підтримка стандартизованих протоколів для обміну даними.

Недоліки:

– висока складність налаштування та управління інтеграціями.
– потреба у постійному моніторингу та обслуговуванні для забезпечення надійності та безпеки.

3.1.3 Обґрунтування обраного рішення

Вибір оптимального підходу до реалізації системи автоматизованого формування тестів базується на аналізі конкретних вимог до системи, ресурсів, доступних для розробки, а також очікуваних результатів і масштабів застосування. Після розгляду переваг і недоліків кожного методу та

технологічного рішення можна зробити обґрунтований вибір для розробки ефективної системи.

1) Обґрунтування вибору методу

Для реалізації автоматизованого формування тестів пропонується використовувати комбінований підхід, що включає генерацію тестів на основі правил для базових потреб і адаптивне тестування для індивідуалізації оцінювання знань студентів. Це дозволить забезпечити баланс між простотою реалізації та гнучкістю системи, враховуючи різні рівні підготовки студентів.

2) Обґрунтування вибору технологій

Використання веб-фреймворку Flask

– flask був обраний як основний веб-фреймворк для розробки інтерфейсу користувача та серверної логіки. Це легкий та гнучкий фреймворк, який забезпечує необхідний функціонал для створення веб-застосунків.

Переваги:

– легкість освоєння та використання: Flask відомий своєю простотою та зрозумілою документацією, що робить його ідеальним для швидкої розробки прототипів та невеликих проектів.
– гнучкість та розширюваність: Flask надає велику свободу в архітектурі додатків і підтримує інтеграцію з великою кількістю розширень.
– активне співтовариство: Flask має велике співтовариство розробників, що забезпечує доступ до великої кількості ресурсів та підтримки.
– код: Основний веб-застосунок реалізований у файлі `app.py`, який відповідає за маршрутизацію та обробку запитів (наприклад, маршрути `/register`, `/login`, `/test` та інші).

Використання SQLite для управління базою даних:

– SQLite обрано як базу даних для зберігання тестових завдань, результатів тестів та користувацьких даних. Це реляційна база даних, яка підходить для невеликих і середніх проектів завдяки своїй простоті у використанні та налаштуванні.

Переваги:

– вбудованість та легкість використання: SQLite не потребує налаштування окремого серверного середовища, що спрощує розробку та розгортання.
– продуктивність: Для невеликих до середніх навантажень SQLite забезпечує достатню продуктивність.
– мінімальні вимоги до обслуговування: SQLite є самодостатньою і не вимагає складного адміністрування.
– код: База даних створена і використовується у файлах `generate_test.py` та `analyze_test_results.py` для роботи з тестами і результатами (див. `generate_test` для створення тестів та `analyze_results` для аналізу результатів).

Генерація тестів на основі правил:

– для формування тестів використовується підхід генерації на основі правил, де питання вибираються з бази даних відповідно до рівня складності студента, визначеного за допомогою моделі машинного навчання.

Переваги:

– простота реалізації: Використання заздалегідь визначених правил для відбору питань забезпечує передбачувану поведінку системи.
– контроль якості: Викладачі можуть легко контролювати якість і релевантність питань, що входять до тестів.
– адаптивність: Модель адаптивно визначає рівень складності питань на основі історії відповідей студента.

– код: Генерація тестів реалізована у файлі `generate_test.py`, де функція `generate_test` вибирає питання з бази даних відповідно до передбаченого рівня складності студента.

Аналіз результатів та адаптація рівня складності:

– система аналізує результати тестів і автоматично коригує рівень складності питань для кожного студента, що дозволяє адаптивно підлаштовуватися під індивідуальні потреби.

Переваги:

– підвищення точності оцінювання: Адаптивне коригування рівня складності питань дозволяє точніше оцінювати знання студентів.
– мотивація студентів: Система надає студентам відповідні за складністю питання, що сприяє їхньому прогресу і підтримує мотивацію.
– код: Аналіз результатів тестів і коригування рівня складності реалізовано у файлі `analyze_test_results.py`. Функція `analyze_results` оцінює відсоток правильних відповідей і відповідно оновлює рівень складності питань у базі даних.

Використання машинного навчання:

– для прогнозування рівня знань студентів використовується модель машинного навчання. Це дозволяє системі автоматично визначати рівень складності питань, які потрібно включити до тесту.

Переваги:

– індивідуалізація тестів: Модель дозволяє адаптувати тестові завдання до рівня знань кожного студента.
– покращена оцінка знань: Використання ML дозволяє враховувати більш складні патерни в даних студентів, що покращує точність оцінювання.
– код: Модель машинного навчання використовується для прогнозування рівня студентів у функції `predict_student_level` у файлі `generate_test.py`.

3) Висновок

Обране рішення поєднує простоту та гнучкість у використанні з можливостями сучасних технологій, що дозволяє створити ефективну і надійну систему автоматизованого формування тестів. Використання Flask для розробки веб-інтерфейсу, SQLite для управління даними, генерації тестів на основі правил та машинного навчання для адаптації тестів під рівень знань студентів забезпечує високу якість оцінювання та індивідуалізацію процесу навчання.

Цей підхід дозволяє ефективно вирішувати основні проблеми, пов'язані з оцінюванням знань, знижуючи навантаження на викладачів та підвищуючи мотивацію і результати студентів.

4 ІДЕЇ АВТОРСЬКОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1.1 Архітектура системи

Архітектура системи автоматизованого формування тестів успішності навчання описує її основні компоненти, їх взаємодію та реалізацію основних функцій. Це дозволяє забезпечити узгоджену роботу всіх частин системи, її гнучкість та масштабованість. Основна архітектура системи включає веб-інтерфейс, базу даних, алгоритми генерації тестів та аналізу результатів, а також інтеграцію з моделями машинного навчання.

Основні компоненти архітектури

1) Веб-інтерфейс

Веб-інтерфейс надає користувачам (студентам та викладачам) зручний спосіб взаємодії із системою. Він забезпечує можливість реєстрації, авторизації, проходження тестів, перегляду результатів та управління тестами.

Компоненти веб-інтерфейсу:

– клієнтська частина: Інтерфейс користувача, реалізований за допомогою HTML, CSS, та JavaScript. Це забезпечує взаємодію користувача з системою через веб-браузер.
– серверна частина: Серверна логіка, реалізована за допомогою фреймворку Flask (файл `app.py`), що обробляє запити клієнтів, взаємодіє з базою даних та виконує бізнес-логіку.

2) База даних

База даних використовується для зберігання інформації про користувачів, питання тестів, результати тестування та інші важливі дані. У цій системі використовується SQLite як проста та ефективна база даних.

Компоненти бази даних:

– таблиці користувачів: Зберігають дані про користувачів, такі як імена, електронні адреси, паролі, ролі (студенти чи викладачі) та інші.
– таблиці тестових питань: Зберігають питання, варіанти відповідей, правильні відповіді, підказки та рівень складності питань.
– таблиці результатів тестування: Зберігають результати відповідей студентів, включаючи правильні та неправильні відповіді, дати тестування та іншу інформацію.

3) Алгоритми генерації тестів

Алгоритми генерації тестів відповідають за формування тестових завдань відповідно до рівня знань студентів та інших критеріїв. Вони забезпечують адаптивність тестування та варіативність питань.

Компоненти алгоритмів:

– функція генерації тестів: Реалізована у файлі `generate_test.py`, функція `generate_test` вибирає питання з бази даних відповідно до рівня складності, визначеного для студента.
– Модель прогнозування рівня знань: Використовується для визначення рівня складності питань, які включаються до тесту.

4) Аналіз результатів тестування

Аналіз результатів тестування дозволяє системі адаптувати рівень складності питань та надавати зворотний зв'язок студентам.

Компоненти аналізу:

– функція аналізу результатів: Реалізована у файлі `analyze_test_results.py`, функція `analyze_results` аналізує результати тестів та відповідно коригує рівень складності питань у базі даних.
– аналіз продуктивності: Визначає загальні тенденції та поведінкові патерни студентів на основі їхніх відповідей.

5) Інтеграція з моделями машинного навчання

Моделі машинного навчання використовуються для прогнозування рівня знань студентів та адаптації тестів відповідно до їхніх потреб.

Компоненти інтеграції:

– тренування моделі: Модель машинного навчання тренується на історичних даних студентів для прогнозування їхнього рівня знань.
– прогнозування рівня знань: Використовується для визначення рівня складності питань, які включаються до тесту (функція `predict_student_level` у файлі `generate_test.py`).

6) Загальна архітектура системи:

Архітектура системи передбачає взаємодію всіх компонентів для забезпечення ефективної роботи. Взаємодія між основними компонентами описана нижче:

– користувачі (студенти та викладачі) взаємодіють з системою через веб-інтерфейс. Веб-інтерфейс обробляє запити користувачів та передає їх до серверної частини.
– серверна частина обробляє запити, виконує необхідну бізнес-логіку та взаємодіє з базою даних для зберігання та отримання інформації.
– алгоритми генерації тестів вибирають питання з бази даних відповідно до рівня знань студента та формують індивідуальні тести.
– моделі машинного навчання використовуються для прогнозування рівня знань студентів та адаптації тестів.
– аналіз результатів тестування коригує рівень складності питань, що дозволяє адаптувати тести під індивідуальні потреби студентів.

Діаграма архітектури системи:

Для кращого розуміння архітектури системи, представлена діаграма (див.рис 4.1), що показує взаємодію між основними компонентами:

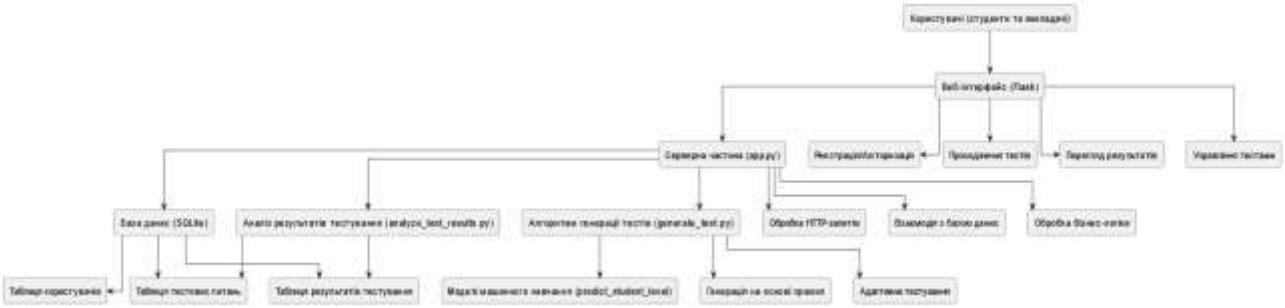


Рисунок 4.1 – Архітектура системи

4.1.2 Висновок

Архітектура системи автоматизованого формування тестів забезпечує ефективну взаємодію між основними компонентами, гнучкість та масштабованість системи. Використання сучасних технологій, таких як Flask,

SQLite та машинне навчання, дозволяє створити надійну систему, яка відповідає потребам сучасної освіти. Всі компоненти системи інтегровані таким чином, щоб забезпечити оптимальну продуктивність та високу якість оцінювання знань студентів.

5 АНАЛІЗ І УЗАГАЛЬНЕННЯ ФАКТИЧНОГО МАТЕРІАЛУ

5.1.1 Аналіз і узагальнення фактичного матеріалу

Під час розробки системи автоматизованого формування тестів було створено базу даних, яка зберігає інформацію про студентів, їхні результати тестування та рівень знань. Аналіз результатів тестів дозволив системі автоматично коригувати рівень складності питань, які будуть задаватися в майбутніх тестах. Це дозволяє забезпечити плавне зростання складності для кожного студента відповідно до його успішності.

1) Структура бази даних

У базі даних, створеній за допомогою SQLite, зберігається декілька основних таблиць:

Таблиця студентів (`Users`):

– зберігає інформацію про студентів, включаючи унікальні ідентифікатори, імена, електронні адреси, рівні складності та інші атрибути.

Таблиця питань (`Questions`):

– містить питання для тестів, їхні рівні складності, правильні відповіді та підказки для студентів.

Таблиця результатів тестів (`TestResults`):

– зберігає дані про відповіді студентів на питання, включаючи інформацію про те, чи були відповіді правильними, та дату проходження тесту.

Таблиця логів (`Logs`):

– зберігає історію дій користувачів для відстеження їхньої активності у системі.

2) Алгоритми оновлення рангу студентів

Після проходження кожного тесту система аналізує результати студентів і автоматично оновлює їхній рівень знань відповідно до їхньої продуктивності:

Успішне проходження тесту:

– студенти, які успішно склали тест (отримали високий відсоток правильних відповідей), отримують підвищення рівня складності питань. Це означає, що наступні тести для них будуть містити більш складні питання, що сприяє їхньому постійному зростанню і розвитку знань.
– реалізація в коді: У файлі `analyze_test_results.py` функція `analyze_results` визначає успішність проходження тесту і підвищує рівень складності питань для студента, якщо відсоток правильних відповідей перевищує певний поріг.

Середній результат:

– студенти, які отримали середній результат (баланс між правильними і неправильними відповідями), зберігають свій поточний рівень складності. Це дозволяє системі не ускладнювати завдання, якщо студент ще не готовий до підвищення рівня.
– реалізація в коді: Функція `analyze_results` також враховує цей сценарій і залишає рівень складності незмінним, якщо результат тесту знаходиться в середньому діапазоні.

Неуспішне проходження тесту:

– студенти, які не склали тест (отримали низький відсоток правильних відповідей), отримують пониження рівня складності. Крім того, їм

надаються підказки та настанови, що допомагають зрозуміти їхні помилки і краще підготуватися до наступних тестів.

– реалізація в коді: Функція `analyze_results` зменшує рівень складності для студента, якщо відсоток правильних відповідей є занадто низьким, і додає відповідні підказки з таблиці питань.

3) Плавне зростання навантаження

Цей підхід до управління рівнем складності питань забезпечує плавне зростання навантаження для студентів. Студенти, які демонструють високі результати, отримують більш складні завдання, що стимулює їх досягати нових висот у навчанні. Ті, хто стикається з труднощами, отримують необхідну підтримку і легші завдання, що дозволяє їм поступово покращувати свої знання без відчуття перевантаження.

Переваги такого підходу:

– індивідуалізація: Система адаптується до індивідуальних потреб кожного студента, забезпечуючи персоналізоване навчання.
– мотивація: Постійне підвищення рівня складності мотивує студентів досягати кращих результатів.
– підтримка: Студенти, які мають труднощі, отримують необхідну допомогу для подолання їхніх слабких місць.

5.1.2 Висновок

Аналіз фактичного матеріалу та впровадження адаптивного підходу до тестування дозволяють створити ефективну систему для оцінювання знань, яка підтримує плавне зростання складності для найшвидшого прогресу студентів.

6 СТВОРЕННЯ ПРОЄКТУ

Для фактичної реалізації запланованих функцій системи було створено пакет скриптів.

6.1.1 Огляд файлів проекту

1) app.py - Основний файл додатку, що містить маршрути та логіку для роботи з користувачами, тестами та іншими компонентами.
2) analyze_student_p.py - Скрипт для аналізу результатів студентів.
3) analyze_test_results.py - Скрипт для аналізу результатів тестів.
4) generate_test.py - Скрипт для генерації тестів.
5) db_setup.sql - SQL-скрипт для налаштування бази даних.
6) templates - Директорія з HTML-шаблонами.
7) static - Директорія зі статичними файлами (CSS, зображення і т.д.).
8) tests.db - Файл бази даних SQLite.

6.1.2 Реалізація проекту

1) app.py – Основний файл додатку

Цей файл містить основні маршрути та логіку для веб-додатку, включаючи реєстрацію користувачів, логін, управління тестами і т.д.

```
from flask import Flask, render_template, request, redirect, url_for, session
import sqlite3
```

```
app = Flask(__name__)
app.secret_key = 'your_secret_key'

# Функція для підключення до бази даних
def get_db_connection():
    conn = sqlite3.connect('database.db')
    conn.row_factory = sqlite3.Row
    return conn

# Головна сторінка
@app.route('/')
def index():
    return render_template('index.html')

# Сторінка реєстрації
@app.route('/register', methods=('GET', 'POST'))
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        conn = get_db_connection()
```

```
            conn.execute('INSERT INTO users (username, password) VALUES (?, ?)', (username, password))
            conn.commit()
            conn.close()
            return redirect(url_for('login'))
        return render_template('register.html')

    # Сторінка входу
    @app.route('/login', methods=('GET', 'POST'))
    def login():
        if request.method == 'POST':
            username = request.form['username']
            password = request.form['password']
            conn = get_db_connection()
            user = conn.execute('SELECT * FROM users WHERE username = ? AND password = ?', (username, password)).fetchone()
            conn.close()
            if user:
                session['user_id'] = user['id']
                return redirect(url_for('index'))
            else:
                return redirect(url_for('login'))
        return render_template('login.html')
```

```
# Сторінка проходження тесту
@app.route('/take_test', methods=('GET', 'POST'))
def take_test():
    if request.method == 'POST':
        # Обробка результатів тесту
        pass
    return render_template('test.html')

# Сторінка адміністратора
@app.route('/admin')
def admin():
    if 'user_id' in session:
        # Логіка для адміністратора
        pass
    return redirect(url_for('login'))

# Вихід з системи
@app.route('/logout')
def logout():
    session.pop('user_id', None)
    return redirect(url_for('index'))
```

```
if __name__ == '__main__':
    app.run()
```

Файл app.py містить конфігурацію та основні маршрути Flask. get_db_connection()-функція для підключення до бази даних SQLite.
index(), register(), login(), take_test(), admin(), logout()-маршрути для обробки запитів та відображення відповідних сторінок.

2) analyze_student_p.py – Аналіз результатів студентів

Цей скрипт аналізує результати студентів та визначає їх рівень на основі проценту правильних відповідей.

```
import sqlite3


def analyze_student_performance(student_id):
    conn = sqlite3.connect('database.db')
    cur = conn.cursor()


    # Отримуємо результати тесту студента
    cur.execute('SELECT question_id, correct FROM student_answers WHERE student_id = ?', (student_id,))
    answers = cur.fetchall()


    total_questions = len(answers)
    correct_answers = sum(1 for answer in answers if answer[1] == 1)
```

```
    # Визначення рівня студента
    performance_percentage = (correct_answers / total_questions) * 100
    level = determine_level(performance_percentage)

    # Оновлюємо рівень студента в базі даних
    cur.execute('UPDATE students SET level = ? WHERE id = ?', (level, student_id))
    conn.commit()
    conn.close()

def determine_level(percentage):
    if percentage >= 90:
        return 'Advanced'
    elif percentage >= 75:
        return 'Intermediate'
    elif percentage >= 50:
        return 'Beginner'
    else:
        return 'Novice'
```

Опис:

– analyze_student_performance(student_id)-аналізує результати студента та оновлює його рівень в базі даних.

– determine_level(percentage)-визначає рівень студента на основі проценту правильних відповідей.

3) analyze_test_results.py – аналіз результатів тесту

Цей скрипт аналізує результати тестів і визначає рівень складності питань

```
import sqlite3


def analyze_test_results():
    conn = sqlite3.connect('database.db')
    cur = conn.cursor()

    # Отримуємо всі питання
    cur.execute('SELECT id FROM questions')
    questions = cur.fetchall()

    for question in questions:
        question_id = question[0]
        cur.execute('SELECT correct FROM student_answers WHERE question_id = ?', (question_id,))
        answers = cur.fetchall()
```

```
        total_answers = len(answers)

        correct_answers = sum(1 for answer in answers if answer[1] == 1)


        # Визначення рівня складності питання

        difficulty_level = determine_difficulty_level(correct_answers / total_answers)


        # Оновлюємо рівень складності питання в базі даних

        cur.execute('UPDATE questions SET difficulty_level = ? WHERE id = ?', (difficulty_level, question_id))


    conn.commit()

    conn.close()


def determine_difficulty_level(correctness_ratio):

    if correctness_ratio >= 0.9:

        return 'Easy'

    elif correctness_ratio >= 0.7:

        return 'Medium'

    elif correctness_ratio >= 0.5:

        return 'Hard'
```

```
    else:
        return 'Very Hard'
```

Опис:

- analyze_test_results() - аналізує результати тестів і оновлює рівень складності питань в базі даних.
- determine_difficulty_level(correctness_ratio) - визначає рівень складності питання на основі відсотку правильних відповідей.

4) generate_test.py - Генерація тестів

Цей скрипт генерує тести для студентів на основі рівнів складності питань.

```
import sqlite3
import random

def generate_test(student_id):
    conn = sqlite3.connect('database.db')
    cur = conn.cursor()

    # Вибираємо питання з різних рівнів складності
```

```
    cur.execute('SELECT id, question, option1, option2, option3, option4 FROM questions WHERE difficulty_level = "Easy"')

    easy_questions = cur.fetchall()

    cur.execute('SELECT id, question, option1, option2, option3, option4 FROM questions WHERE difficulty_level = "Medium"')

    medium_questions = cur.fetchall()

    cur.execute('SELECT id, question, option1, option2, option3, option4 FROM questions WHERE difficulty_level = "Hard"')

    hard_questions = cur.fetchall()



    # Вибираємо питання для тесту

    selected_questions = random.sample(easy_questions, 3) + random.sample(medium_questions, 3) + random.sample(hard_questions, 3)

    random.shuffle(selected_questions)



    # Зберігаємо вибрані питання для тесту в базу даних

    for question in selected_questions:

        cur.execute('INSERT INTO student_tests (student_id, question_id) VALUES (?, ?)', (student_id, question[0]))



    conn.commit()
```

```
    conn.close()


    return format_test(selected_questions)


def format_test(questions):
    formatted_questions = []
    for question in questions:
        formatted_questions.append({
            'id': question[0],
            'question': question[1],
            'options': [question[2], question[3], question[4], question[5]]
        })
    return formatted_questions
```

Опис:

- generate_test(student_id)-генерує тест для студента, вибираючи питання з різних рівнів складності і зберігає їх у базу даних.
- format_test(questions)-форматує вибрані питання у зручний для рендерингу формат.

5) db_setup.sql - Скрипт для налаштування бази даних

```
        CREATE TABLE Users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE,
    password TEXT NOT NULL,
    role TEXT NOT NULL CHECK(role IN ('student', 'teacher')),
    current_level INTEGER DEFAULT 1 -- Поле для зберігання поточного рівня студента
);

CREATE TABLE Questions (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    question TEXT NOT NULL,
    option1 TEXT NOT NULL,
    option2 TEXT NOT NULL,
    option3 TEXT NOT NULL,
    option4 TEXT,
    correct_answer TEXT NOT NULL,
    hint TEXT,
    difficulty_level INTEGER NOT NULL
);

CREATE TABLE TestResults (
```

```
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    student_id INTEGER NOT NULL,
    question_id INTEGER NOT NULL,
    answer TEXT NOT NULL,
    is_correct BOOLEAN NOT NULL,
    test_date DATETIME NOT NULL,
    FOREIGN KEY (student_id) REFERENCES Users(id),
    FOREIGN KEY (question_id) REFERENCES Questions(id)
);

CREATE TABLE Logs (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INT,
    action TEXT,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Додавання адміністратора
INSERT INTO Users (username, email, password, role) VALUES ('admin', 'admin@example.com', 'admin_password_hash', 'teacher');
```

7 ДЕМОНСТРАЦІЯ

7.1.1 Вікно логіну та реєстрації

Представлено на рис.7.1-7.2

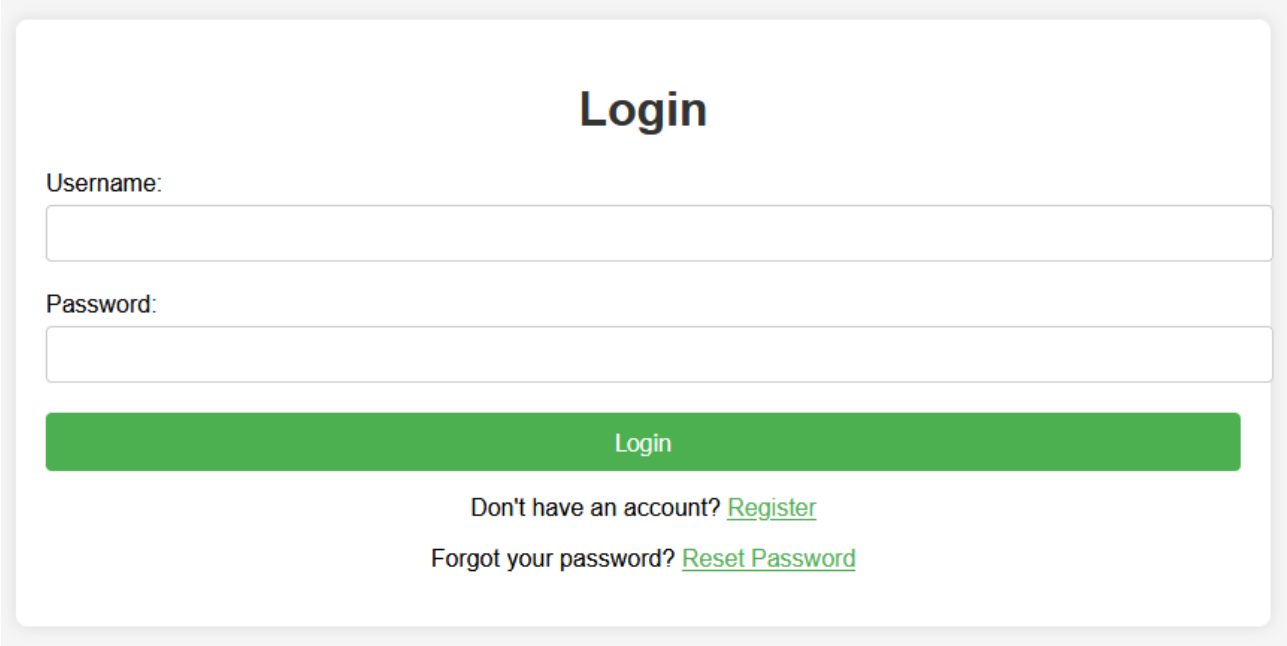


Рисунок 7.1 – Логін

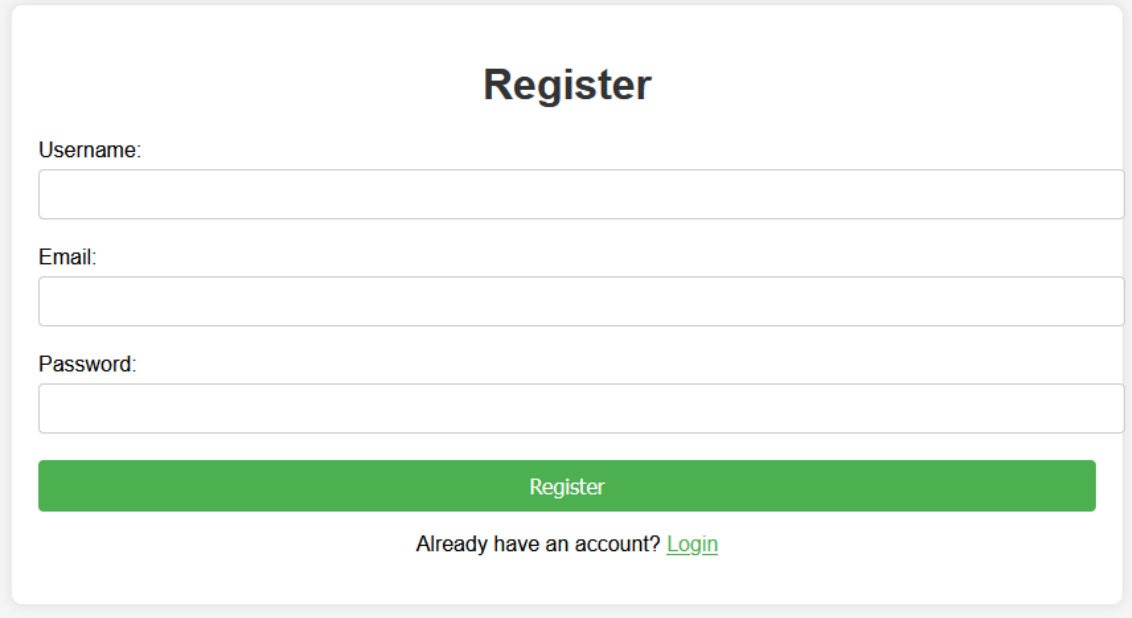


Рисунок 7.2 – Реєстрація

7.1.2 Демонстрація сайту з перспективи вчителя

Основні сторінки представлено на рис.7.3-7.11

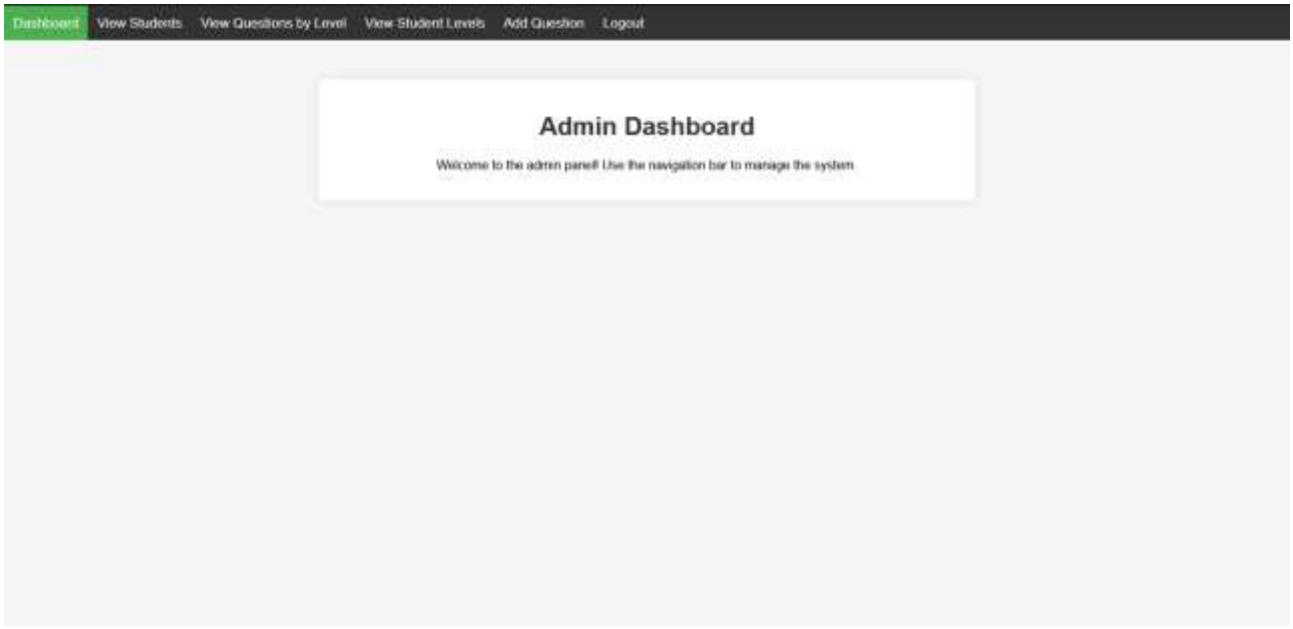


Рисунок 7.3 – Головна сторінка вчителя

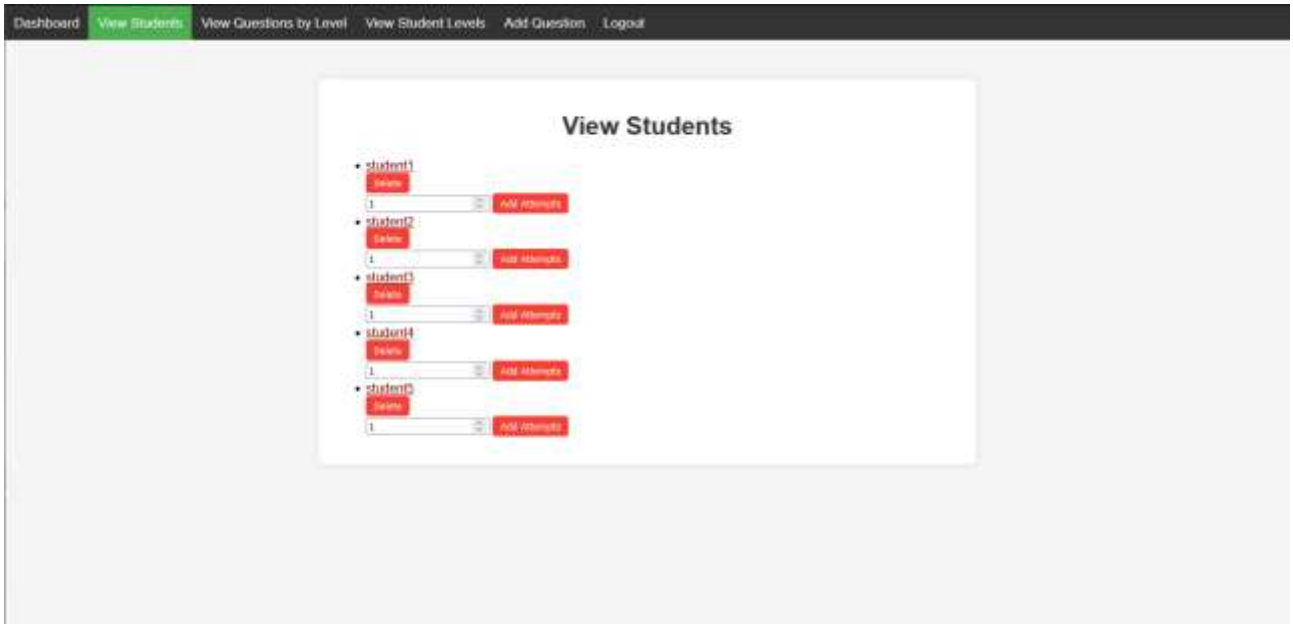


Рисунок 7.4 – Сторінка менеджменту студентів

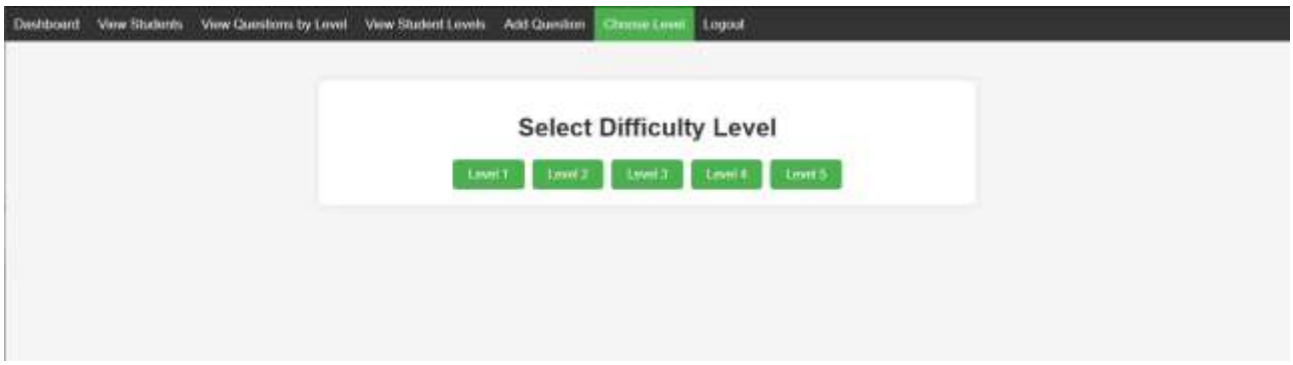


Рисунок 7.5 – Сортування питань по рівню складності

Questions - Level 1

Question	Option 1	Option 2	Option 3	Option 4	Correct Answer	Actions
Який результат додавання 2 + 2?	3	4	5	6	4	Edit Delete
Яка столиця України?	Львів	Київ	Одеса	Харків	Київ	Edit Delete
Якого кольору небо в ясний день?	Червоний	Блакитний	Зелений	Жовтий	Блакитний	Edit Delete
Яка планета найближча до Сонця?	Венера	Земля	Меркурій	Марс	Меркурій	Edit Delete
Який океан найбільший на Землі?	Атлантичний	Індійський	Тихий	Арктичний	Тихий	Edit Delete

Рисунок 7.6 – Питання першого рівня складності

Questions - Level 2

Question	Option 1	Option 2	Option 3	Option 4	Correct Answer	Actions
Який квадратний корінь з 16?	2	4	6	8	4	Edit Delete
Хто автор пєси "Ромео і Джульєтта"?	Чарльз Діккенс	Марк Твен	Вільям Шекспір	Джейн Остін	Вільям Шекспір	Edit Delete
Яка температура кипіння води в градусах Цельсія?	50°C	100°C	150°C	200°C	100°C	Edit Delete
Яка основна мова, якою говорять в Бразилії?	Іспанська	Португальська	Французька	Німецька	Португальська	Edit Delete
Хто намалював "Мону Лізу"?	Вінсент ван Гог	Пабло Пікассо	Леонардо да Вінчі	Мікеланджело	Леонардо да Вінчі	Edit Delete

Рисунок 7.7 – Питання другого рівня складності

Questions - Level 3

Question	Option 1	Option 2	Option 3	Option 4	Correct Answer	Actions
Який хімічний символ золота?	Au	Ag	Pt	Pb	Au	Edit Delete
Яка найдовша річка у світі?	Ніл	Амазонка	Янцзи	Міссісіпі	Ніл	Edit Delete
Хто вважається батьком сучасної фізики?	Альберт Ейнштейн	Ісаак Ньютон	Галілео Галілей	Нікола Тесла	Ісаак Ньютон	Edit Delete
Яка столиця Японії?	Пекін	Сеул	Токіо	Бангкок	Токіо	Edit Delete
Яка швидкість світла у вакуумі?	300,000 км/с	150,000 км/с	1,000,000 км/с	600,000 км/с	300,000 км/с	Edit Delete

Рисунок 7.8 – Питання третього рівня складності

Questions - Level 4

Question	Option 1	Option 2	Option 3	Option 4	Correct Answer	Actions
Яка похідна функції sin(x)?	cos(x)	-cos(x)	sin(x)	-sin(x)	cos(x)	Edit Delete
Яка найбільша планета в нашій Сонячній системі?	Земля	Марс	Юпітер	Сатурн	Юпітер	Edit Delete
Хто є автором книги "1984"?	Джордж Орвелл	Олдос Хакслі	Рей Бредбері	Дж.Р.Р. Толкін	Джордж Орвелл	Edit Delete
Яка столиця Канади?	Торонто	Ванкувер	Оттава	Монреаль	Оттава	Edit Delete
Яка основна мова програмування для розробки Android додатків?	Swift	Python	Java	Ruby	Java	Edit Delete

Рисунок 7.9 – Питання четвертого рівня складності

Questions - Level 5

Question	Option 1	Option 2	Option 3	Option 4	Correct Answer	Actions
Яка формула для обчислення площі кола?	πr^2	$2\pi r$	πd	$2r$	πr^2	Edit Delete
Хто розробив теорію відносності?	Ісаак Ньютон	Галілео Галілей	Альберт Ейнштейн	Нільс Бор	Альберт Ейнштейн	Edit Delete
Яка хімічна формула води?	H_2O	CO_2	O_2	CH_4	H_2O	Edit Delete
Який елемент має найбільший атомний номер?	Уран	Плутоній	Оганессон	Гідроген	Оганессон	Edit Delete
Яка основна функція ЦПУ в компютері?	Зберігання	Обробка	Вхід	Вихід	Обробка	Edit Delete

Рисунок 7.10 – Питання п’ятого рівня складності

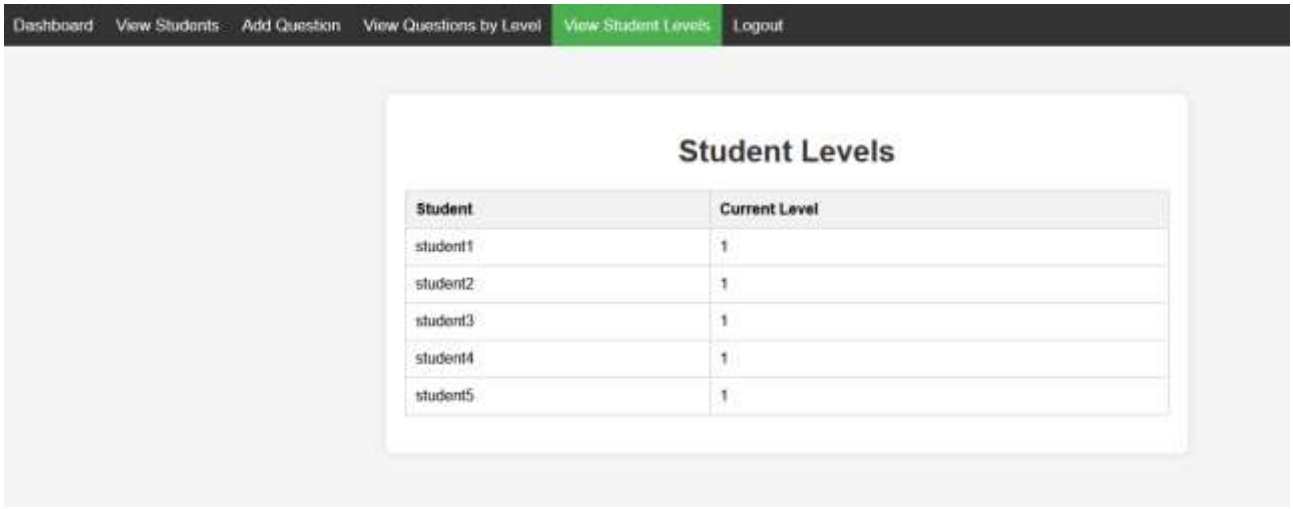


Рисунок 7.11 – Рівень знань учня

7.1.3 Демонстрація від лиця студента

Проходження тесту студентом представлено на рис.7.12-7.14

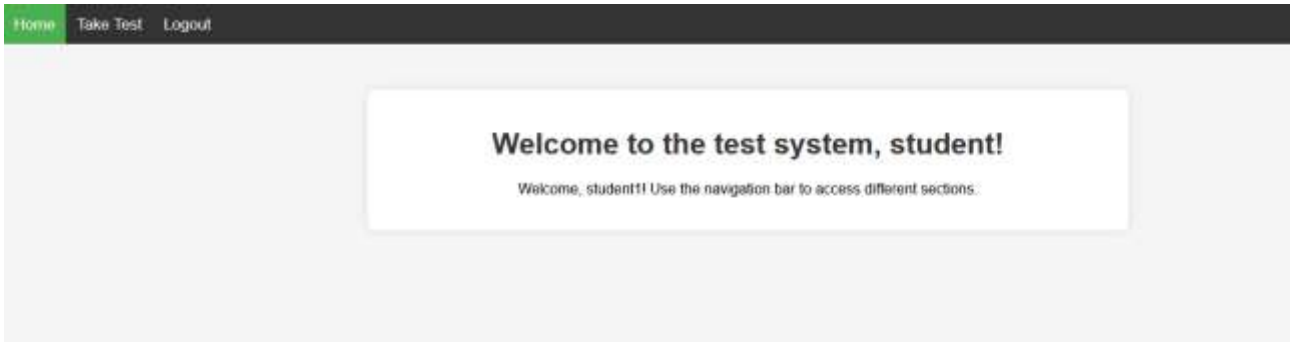
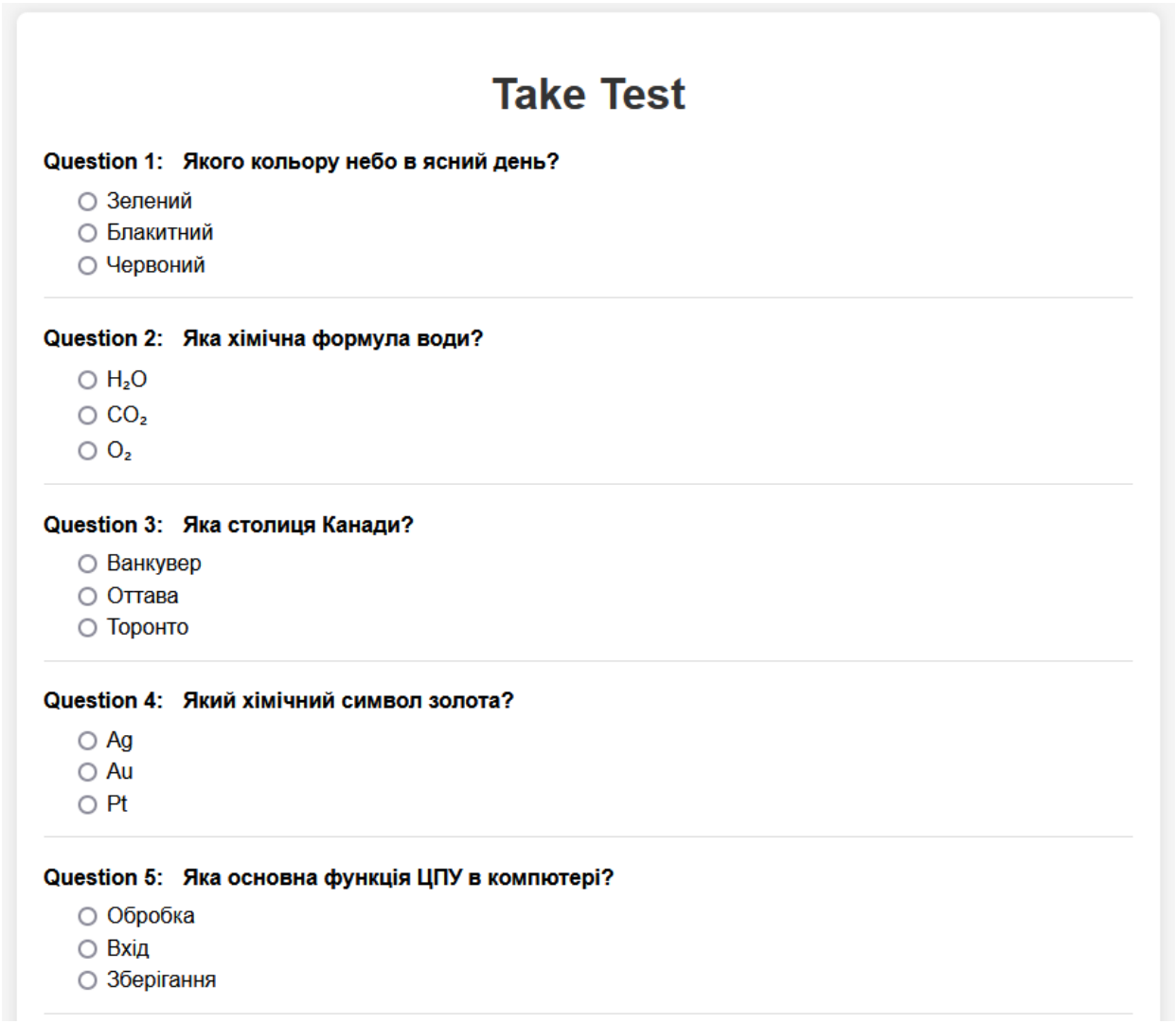
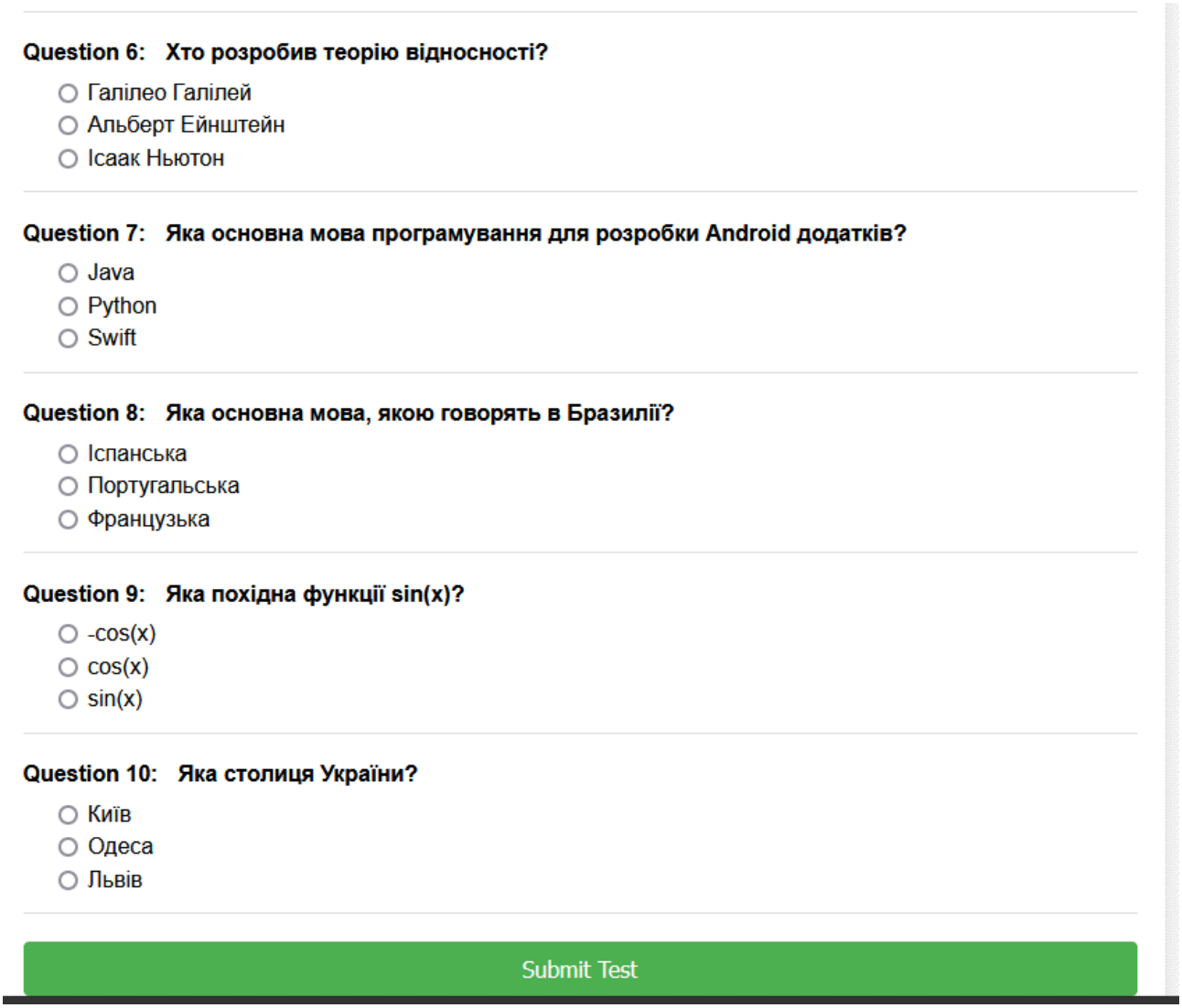


Рисунок 7.12 – головна сторінка студента



а)перша половина тесту



б)друга половина тесту

Рисунок 7.13 – Тест

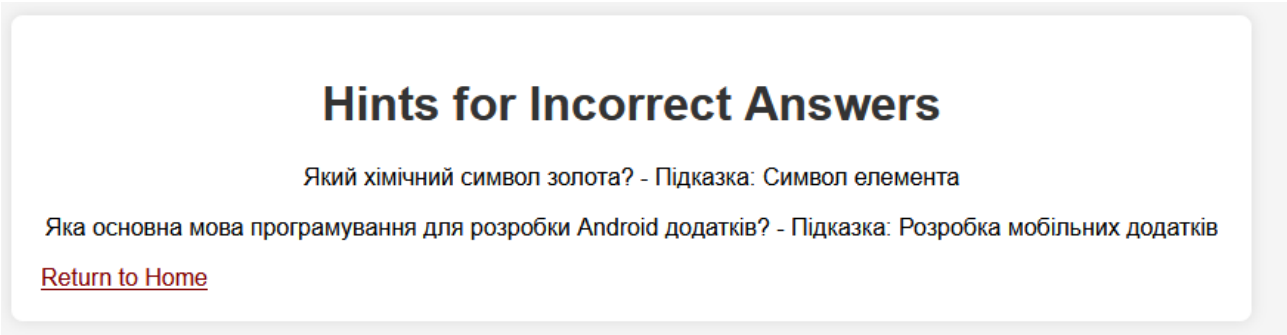


Рисунок 7.14 – Підказки на неправильні відповіді

7.1.4 Демонстрація адаптивності системи

Представлення змін у системі після проходження тесту на рис. 7.15-7.21

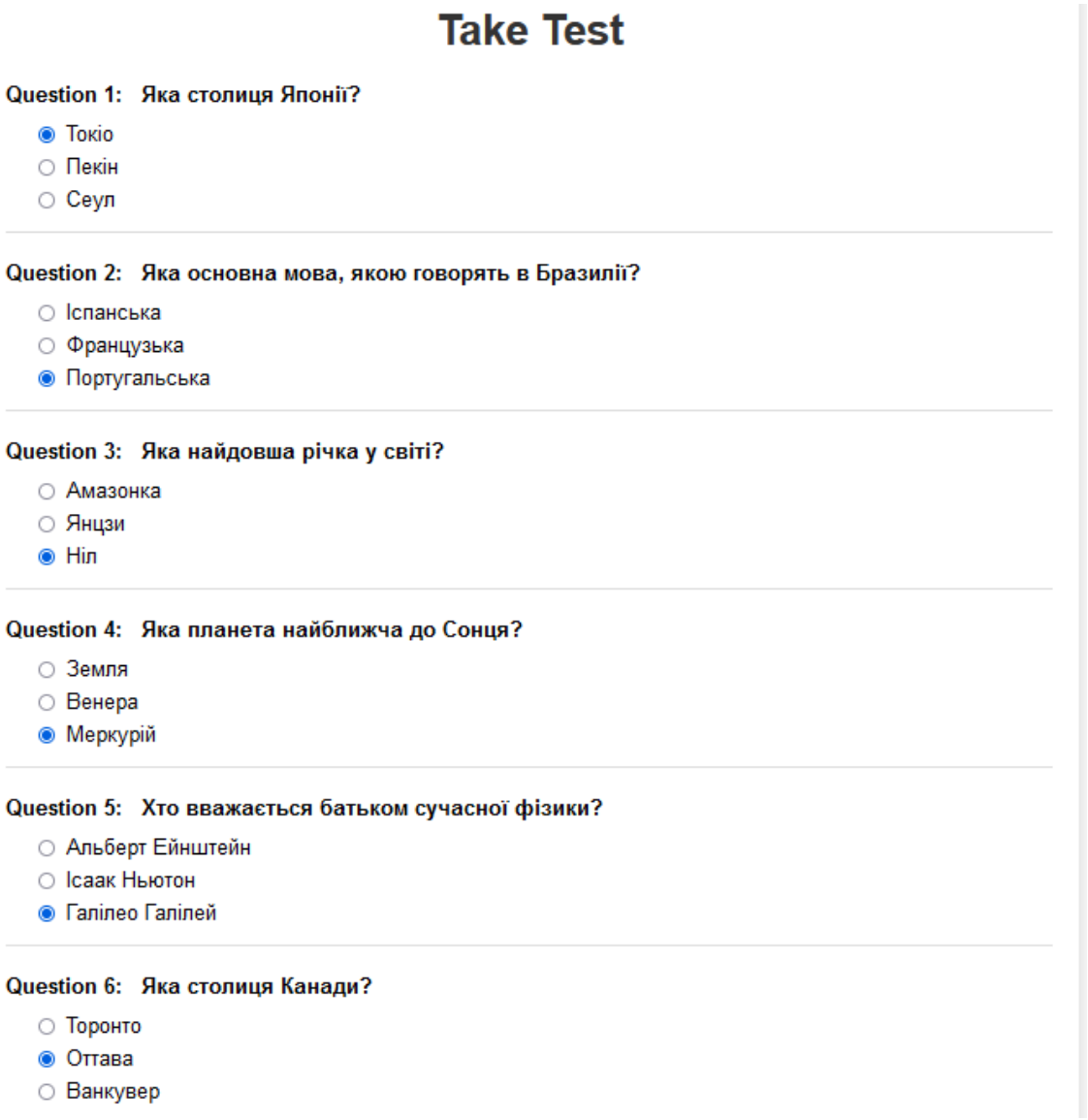


Рисунок 7.15 – Вірні відповіді

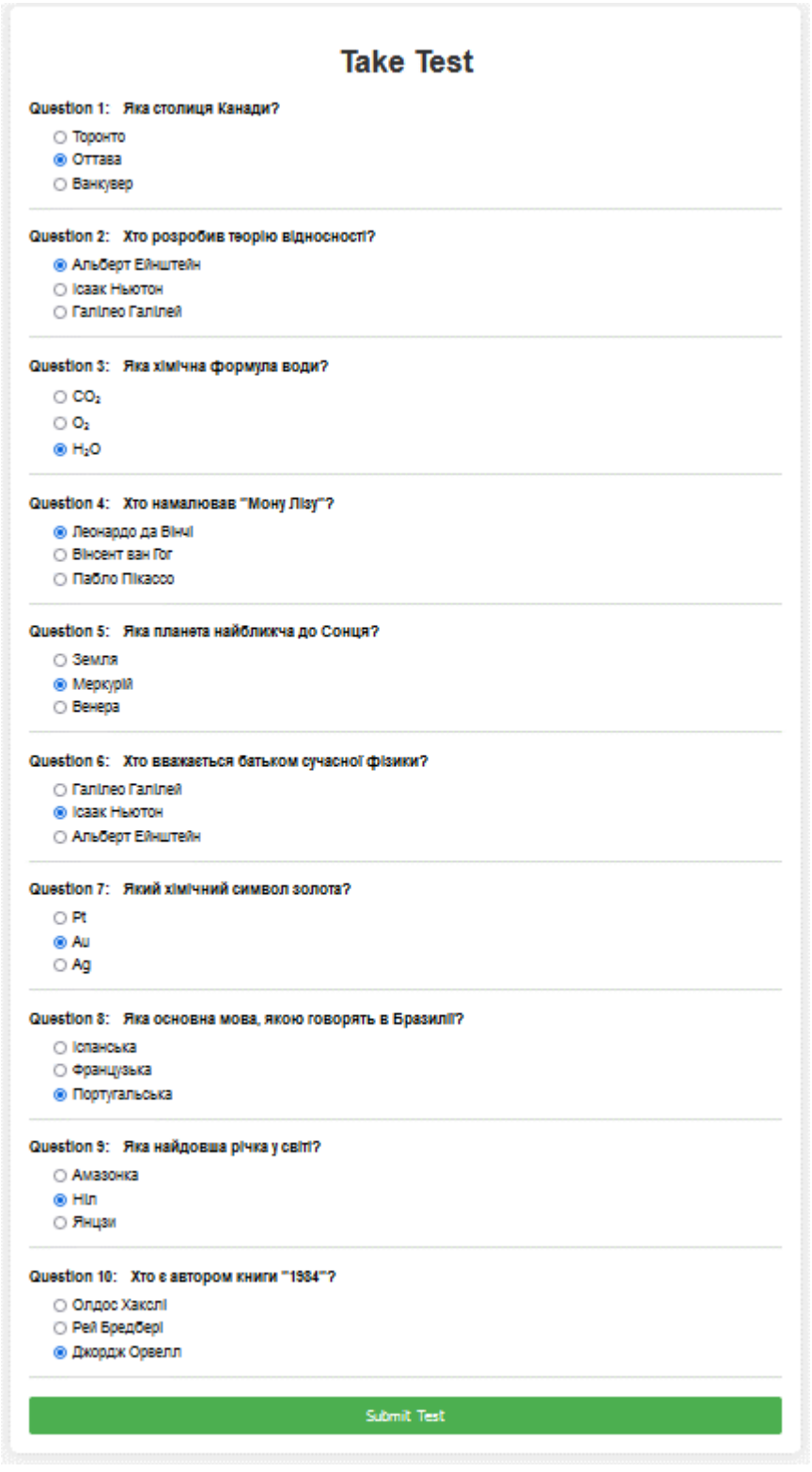


Рисунок 7.16 – Вірні відповіді

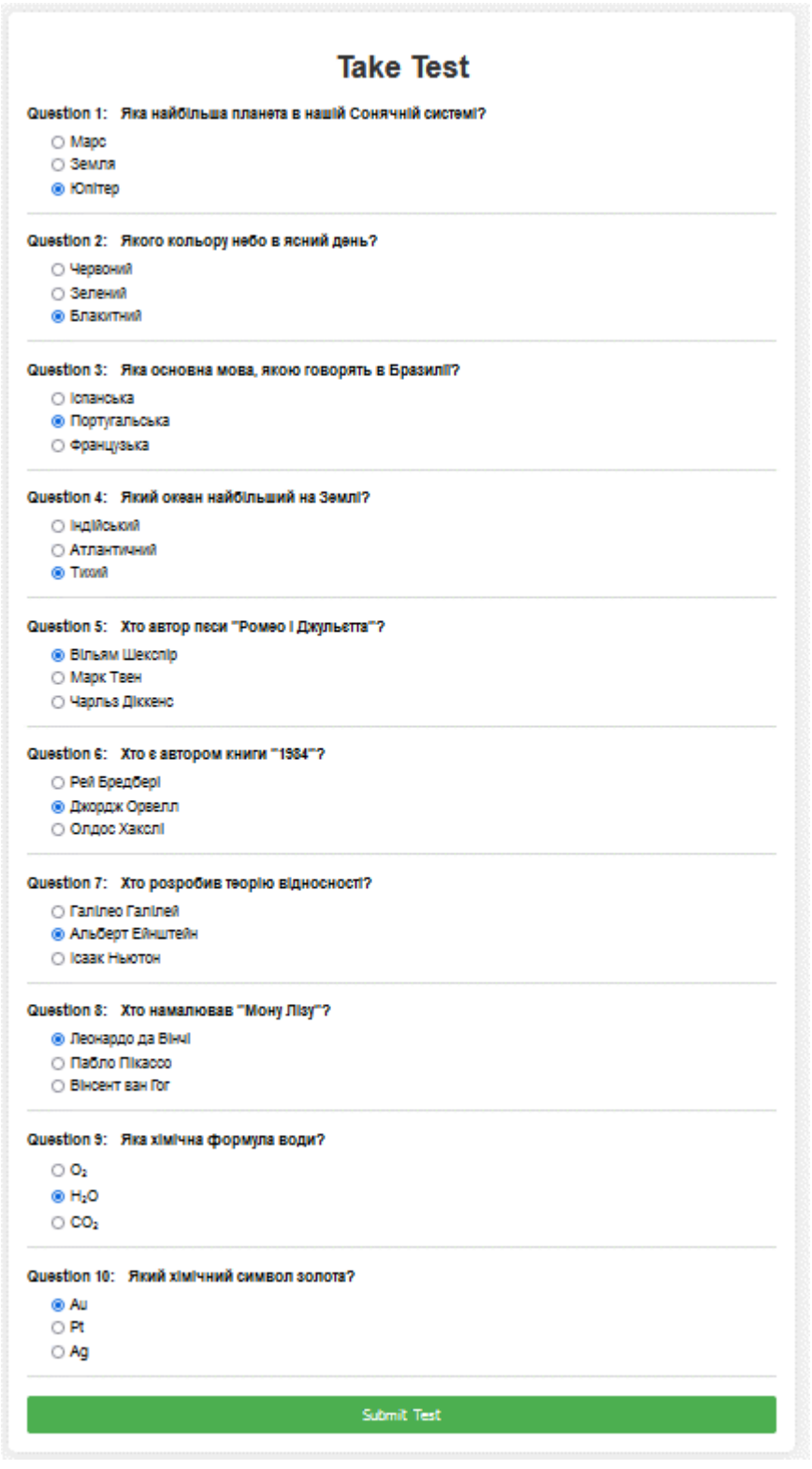


Рисунок 7.17 – Вірні відповіді

Student Levels

Student	Current Level
student1	2
student2	1
student3	1
student4	1
student5	1

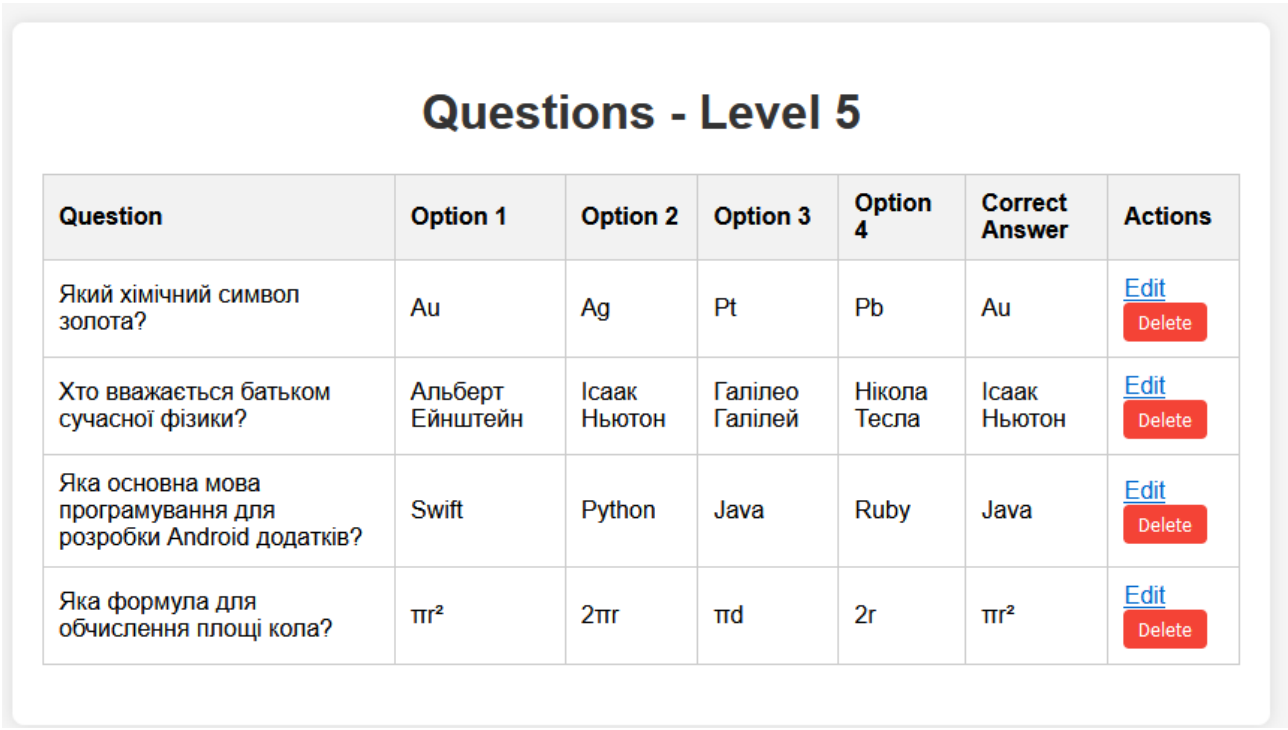
Рисунок 7.18 – Відмінник підвищив рівень

Рівень складності питань змінився(див. рис. 7.19). на них він відповів правильно забагато разів, тому система підлаштувалась під нього, раніше складні питання стали нижче по рівню.

Questions - Level 2

Question	Option 1	Option 2	Option 3	Option 4	Correct Answer	Actions
Який квадратний корінь з 16?	2	4	6	8	4	Edit Delete
Хто є автором книги "1984"?	Джордж Орвелл	Олдос Хакслі	Рей Бредбері	Дж.Р.Р. Толкін	Джордж Орвелл	Edit Delete

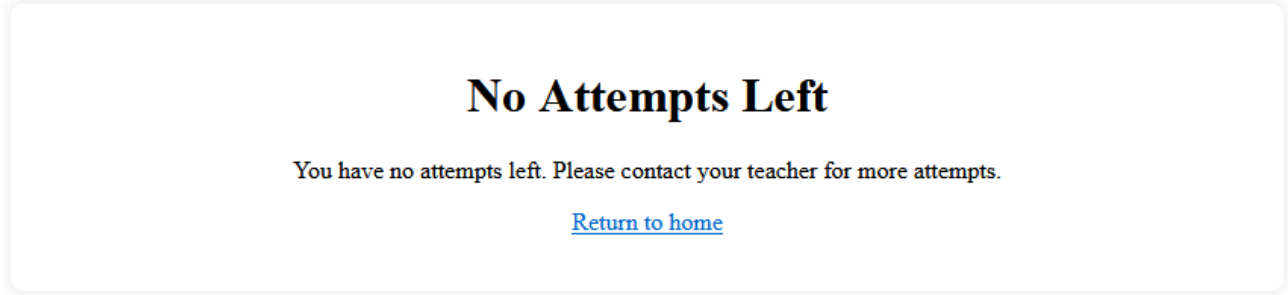
Рисунок 7.19 – Відмінник підвищив рівень



Questions - Level 5

Question	Option 1	Option 2	Option 3	Option 4	Correct Answer	Actions
Який хімічний символ золота?	Au	Ag	Pt	Pb	Au	Edit Delete
Хто вважається батьком сучасної фізики?	Альберт Ейнштейн	Ісаак Ньютон	Галілео Галілей	Нікола Тесла	Ісаак Ньютон	Edit Delete
Яка основна мова програмування для розробки Android додатків?	Swift	Python	Java	Ruby	Java	Edit Delete
Яка формула для обчислення площі кола?	πr²	2πr	πd	2r	πr²	Edit Delete

Рисунок 7.20 – Зміна питань у 5 рівні



No Attempts Left

You have no attempts left. Please contact your teacher for more attempts.

Return to home

Рисунок 7.21 – У учня не залишилось спроб

7.1.5 Алгоритм використання коду для вчителя

1) Вхід в Систему

Перейдіть на сторінку входу:

– відкрийте веб-браузер та перейдіть на сторінку входу за URL ` http://127.0.0.1:5000/login`.

Увійдіть як вчитель:

– введіть своє ім'я користувача та пароль.
– натисніть кнопку "Login".

2) Перехід до Панелі Управління Вчителя

Перейдіть до панелі управління:

– після входу натисніть на посилання "Admin Dashboard" або автоматично буде перенаправлено на цю сторінку.

3) Вибір Рівня Складності для Перегляду Питань

Перегляд питань за рівнями:

– на сторінці панелі управління натисніть одну з кнопок, що відповідає рівню складності (Level 1, Level 2, ... Level 5).
– ви перейдете на сторінку з переліком питань відповідного рівня складності.

4) Додавання або Редагування Питань

Додавання нового питання:

На сторінці перегляду питань натисніть кнопку "Add Question".

Заповніть форму:

– question: Текст питання.
– option 1, Option 2, Option 3, Option 4: Варіанти відповідей.
– correct Answer: Правильна відповідь.
– hint: Підказка для студента (опціонально).
– difficulty Level: Оберіть рівень складності (від 1 до 5).
– натисніть кнопку "Add Question" для збереження нового питання.

Редагування існуючого питання:

– на сторінці перегляду питань знайдіть питання, яке потрібно редагувати.
– натисніть кнопку "Edit" поруч з питанням.
– внесіть необхідні зміни у форму редагування.
– натисніть кнопку "Save Changes" для збереження змін.

5) Видалення Питання

Видалення питання:

– на сторінці перегляду питань знайдіть питання, яке потрібно видалити.
– натисніть кнопку "Delete" поруч з питанням.
– підтвердіть видалення у спливаючому вікні.

6) Перегляд Рівнів Складності Студентів

Перегляд рівнів студентів:

– на панелі управління натисніть на посилання "View Student Levels".
– ви перейдете на сторінку, де буде відображено поточний рівень складності кожного студента.

7) Вихід із Системи

Вихід:

– Натисніть на посилання "Logout" у навігаційній панелі, щоб вийти з системи.

.7.1.6 Алгоритм використання коду для учня

1) Вхід в Систему

Перейдіть на сторінку входу:

– відкрийте веб-браузер та перейдіть на сторінку входу за URL `http://127.0.0.1:5000/login`.

Увійдіть як учень:

– введіть своє ім'я користувача та пароль.
– натисніть кнопку "Login".

2) Перехід до Домашньої Сторінки Учня

Перейдіть до домашньої сторінки:

– після входу натисніть на посилання "Home" або автоматично буде перенаправлено на цю сторінку.

3) Початок Проходження Тесту

Розпочніть тест:

– на домашній сторінці учня натисніть на посилання "Take Test".
– ви перейдете на сторінку з питаннями тесту.

4) Проходження Тесту

Вибір відповідей:

– прочитайте питання та виберіть один з варіантів відповіді.
– питання включають варіанти з різних рівнів складності, по 5 питань з кожного рівня.

Надсилання відповідей:

– після заповнення всіх відповідей натисніть кнопку "Submit Test".

5) Перегляд Результатів

Перегляньте результати:

– після надсилання тесту система автоматично обробить ваші відповіді.
– якщо ви не відповіли на 70% питань вищого рівня складності, ваш рівень знизиться, і наступний тест буде містити питання з нижчого рівня складності.
– якщо ви відповіли на більше 70% питань, ваш рівень підвищиться.

Отримання підказок:

– якщо ви допустили помилки, вам можуть бути надані підказки для покращення розуміння теми.

6) Перегляд Домашньої Сторінки Після Тесту

Перейдіть на домашню сторінку:

– після завершення тесту ви можете повернутися на домашню сторінку, натиснувши на посилання "Home".

7) Вихід із Системи

Вихід:

– натисніть на посилання "Logout" у навігаційній панелі, щоб вийти з системи.

ВИСНОВКИ

Розроблена система тестування забезпечує ефективний спосіб для вчителів і студентів взаємодіяти з навчальним контентом. Вона дозволяє вчителям легко додавати, редагувати та управляти питаннями, а також відстежувати прогрес студентів. Для студентів система пропонує гнучкий і адаптивний підхід до тестування, який автоматично коригує рівень складності тестів на основі їхніх результатів.

Основні можливості системи:

1) Управління питаннями з різними рівнями складності:
   - Вчителі можуть переглядати питання за рівнями складності та додавати нові питання з урахуванням різних рівнів.
   - Можливість редагувати і видаляти питання забезпечує динамічне управління контентом тестів.
2) Адаптивне тестування для студентів:
   - Система автоматично генерує тести, що містять по 5 питань з кожного рівня складності.
   - Рівень складності наступних тестів коригується залежно від результатів студентів, що сприяє підвищенню їхніх навчальних досягнень.
3) Персоналізований підхід до навчання:
   - Студенти отримують тести, які відповідають їхньому поточному рівню знань.
   - Підказки для невірних відповідей допомагають студентам краще зрозуміти матеріал і виправити помилки.
4) Простий і інтуїтивний інтерфейс:
   - Веб-додаток має зрозумілий інтерфейс, що дозволяє вчителям і студентам легко навігувати по системі.
   - Кнопки рівнів складності та інші елементи управління розташовані логічно і зручно для користувачів.

5) Безпечний доступ і управління користувачами:

– Система підтримує безпечний вхід для вчителів та студентів.
– Вчителі можуть переглядати і управляти рівнями студентів, що забезпечує гнучкість у навчальному процесі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Kim, S. & Lee, Y. "Automated Test Generation for Educational Assessment". Educational Technology Research and Development (2020) URL:https://www.researchgate.net/publication/346301548

2. Gierl, M.J., Lai, H. "Using Automatic Item Generation to Create Multiple-Choice Test Items". International Journal of Testing (2018) URL:https://www.researchgate.net/publication/293168299_Using_Automatic_Item_Generation_to_Improve_the_Quality_of_MCQ_Distractors

3. Johnson, L., Beck, R. "Artificial Intelligence in Computer-Based Testing". Journal of Educational Computing Research (2019) URL: https://journals.sagepub.com/doi/full/10.1177/07356331241248686

4. Мартинюк І.В., Павлюк В.М. "Застосування нейронних мереж у системах автоматизованого тестування". (2021) URL:https://conf.ztu.edu.ua/informatsijno-kompyuterni-tehnologiyi-2021-1-3-kvitnya-2021-r/

5. Python SQLite. YouTube URL:https://www.youtube.com/watch?v=TwnCXdCa8qg&list=PLA0M1Bcd0w8x4Inr5oYttMK6J47vxgv6J&index=1

6. Pareto law explained. worksection URL:https://worksection.com/en/blog/pareto-principle-80-20-rule.html