ДЕРЖАВНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук та інформаційних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Автоматизована система генерації тестових завдань на основі функції Randomize»

Студента 4 курсу, 10 групи, Нагорний Андрій Олегович

спеціальності
122 «Комп'ютерні науки» _____
підпис студента

Науковий керівник
кандидат технічних наук, доцент _____ Демідов Павло Георгійович
підпис керівника

Гарант освітньої програми
кандидат технічних наук, доцент _____ Демідов Павло Георгійович
підпис керівника

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«20» грудня 2022р.

Завдання

на випускну кваліфікаційну роботу (проект) студенту

Нагорному Андрію Олеговичу
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
«Автоматизована система генерації тестових завдань на основі функції Randomize»
Затверджена наказом ректора від *«09» грудня 2022 р. № 3332*
2. Строк здачі студентом закінченої роботи *31 травня 2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка алгоритмів, методів та засобів для автоматизованої генерації тестових завдань на основі функції Randomize.
Об'єкт дослідження: процес оцінювання знань користувачів за допомогою тестових систем
Предмет дослідження: методи та алгоритми генерації тестових завдань на основі функції Randomize
4. Перелік графічного матеріалу :
_____
_____
_____
_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Демідов П.Г.	15.12.2022 р.	15.12.2022 р.
2	Демідов П.Г.	15.12.2022 р.	15.12.2022 р.
3	Демідов П.Г.	15.12.2022 р.	15.12.2022 р.

6.Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ МЕХАНІЗМІВ АВТОМАТИЗОВАНОЇ СИСТЕМИ ОЦІНКИ ЗНАНЬ КОРИСТУВАЧІВ

1.1. Поняття та сутність автоматизованого оцінювання знань користувачів.

1.2. Аналіз відомих методів автоматичної генерації тестових завдань

1.3. Характеристика існуючих методів та програмних засобів оцінювання знань.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ГЕНЕРАЦІЇ ТЕСТОВИХ ЗАВДАНЬ ТА ОЦІНКИ ЗНАНЬ НА ОСНОВІ ФУНКЦІЇ RANDOMIZE

2.1. Постановка задачі

2.1. Метод автоматичної генерації тестових завдань.

2.2. Метод оцінювання результату проходження тесту.

2.3. Алгоритм генерації тестових завдань з використанням функції Randomize.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ГЕНЕРАЦІЇ ТЕСТОВИХ ЗАВДАНЬ ТА ОЦІНКИ ЗНАНЬ НА ОСНОВІ ФУНКЦІЇ RANDOMIZE

3.1. Архітектура системи генерації тестових завдань та оцінки знань користувачів

3.2. Програмна реалізація системи автоматизованого тестування знань користувачів

3.3. Технологія використання розробленої системи тестування.

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*04.10.2022*	*04.10.2022*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.12.2022*	*15.12.2022*
3	*Вступ*	*03.02.2023*	*03.02.2023*
4	*РОЗДІЛ 1. Теоретичне дослідження механізмів автоматизованої системи оцінки знань користувачів*	*28.02.2023*	*28.02.2023*
5	*РОЗДІЛ 2. Проектування системи генерації тестових завдань та оцінки знань на основі функції Randomize*	*06.04.2023*	*06.04.2023*
6	*РОЗДІЛ 3. Розробка програмного забезпечення системи генерації тестових завдань та оцінки знань на основі функції Randomize*	*12.05.2023*	*12.05.2023*
7	*Висновки*	*15.05.2023*	*15.05.2023*
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*30.05.2023*	*30.05.2023*
9	*Попередній захист випускної кваліфікаційної роботи*	*31.05.2023-01.06.2023*	*31.05.2023-01.06.2023*
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*02.06.2023*	*02.06.2023*
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*05.06.2023*	*05.06.2023*
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

7. Календарний план виконання роботи

8. Дата видачі завдання «15» грудня 2022 р.

Керівник випускної кваліфікаційної роботи (проекту)
Демідов П.Г.
(прізвище, ініціали, підпис)

Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

Завдання прийняв студент-дипломник Нагорний А.О
(прізвище, ініціали, підпис)

9. Відгук керівника випускної кваліфікаційної роботи (проекту)

Керівник випускної кваліфікаційної роботи (проекту)
_____ 30.05.2023 р.
(підпис, дата

10. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента Нагорний А.О
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри Пурський О.І.
(підпис, прізвище, ініціали)

«____»_____2023 р.

Анотація

Дана дипломна робота присвячена розробці автоматизованої системи генерації тестових завдань, яка використовує функцію Randomize. В ході роботи було створено зручний та ефективний інструмент для вчителів та освітніх установ, який дозволяє швидко та автоматично створювати варіанти тестових завдань з випадковою генерацією питань. У роботі проведений аналіз існуючих методів генерації тестових завдань, виявлені їх переваги та недоліки. На основі цього аналізу була обрана функція Randomize як основний інструмент для генерації випадкових елементів тестових завдань. Для реалізації системи були використані сучасні технології програмування та інтегровані середовища розробки. Система надає можливість визначати критерії для генерації питань, такі як складність, тип питання, тематична область тощо. Застосування автоматизованої системи генерації тестових завдань значно полегшує роботу вчителів, економлячи час на складання тестів та забезпечуючи велику кількість варіантів для студентів. Система також може бути використана для проведення тестувань у навчальних закладах та в інших сферах, де потрібна випадкова генерація завдань.

Abstract

This thesis is devoted to the development of an automated test task generation system that uses the Randomize function. During the work, a convenient and effective tool for teachers and educational institutions was created, which allows you to quickly and automatically create variants of test tasks with random generation of questions. The paper analyzes the existing methods of generating test tasks, reveals their advantages and disadvantages. Based on this analysis, the Randomize function was chosen as the main tool for generating

random elements of the test items. Modern programming technologies and integrated development environments were used to implement the system. The system provides an opportunity to define criteria for question generation, such as difficulty, question type, topic area, etc. The use of an automated system for generating test tasks greatly facilitates the work of teachers, saving time for taking tests and providing a large number of options for students. The system can also be used for testing in educational institutions and other areas where random task generation is required.

ЗМІСТ

ВСТУП

Актуальність теми полягає у забезпеченні ефективності та економії часу у процесі створення тестових завдань та розширенні різноманітності тестів. Крім того, зростає потреба у персоналізованому навчанні, де кожен студент може отримати індивідуально підібрані завдання залежно від свого рівня знань та навчальних потреб. Автоматизована система генерації тестових завдань з функцією Randomize може створювати унікальні тести для кожного студента, забезпечуючи індивідуальний підхід та розвиток.

Метою дипломної роботи є розробка алгоритмів, методів та засобів для автоматизованої генерації тестових завдань на основі функції Randomize. Завдання включає аналіз можливостей існуючих систем генерації тестових даних, дослідження структури та процесу їх генерації, вибір підходів та методів, розробку архітектури та алгоритмів програми, вибір мови програмування та інших необхідних технологій.

Об’єкт дослідження дипломної роботи процес оцінювання знань користувачів за допомогою тестових систем.

Предметом дослідження дипломної роботи є методи та алгоритми генерації тестових завдань на основі функції Randomize .

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел із 12 найменувань, додатків і містить 46 сторінок основного тексту і 14 рисунків

РОЗДІЛ 1. ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ МЕХАНІЗМІВ АВТОМАТИЗОВАНОЇ СИСТЕМИ ОЦІНКИ ЗНАНЬ КОРИСТУВАЧІВ.

1.1. Поняття та сутність автоматизованого оцінювання знань користувачів.

На сьогоднішній день оцінка якості навчання є однією з найактуальніших проблем у галузі освіти. З одного боку, потрібні великі часові затрати, з іншого - необхідність аналізу отриманих даних та скорочення етапу реагування на результати. Важливо заздалегідь обговорити межі, при перевищенні яких слід приймати управлінські заходи зі зміни якості освіти. Дуже важливо розрахувати витрати на проведення оцінки якості освіти та порівняти їх з кінцевим результатом, який хоче отримати освітня установа.

Педагогічна діагностика розвивається вже з 80-90-х років. Одним з напрямів синтезу даної галузі є поєднання з математичними методами та інформаційними технологіями.

У провідних країнах світу комп'ютерне тестування користується заслуженим довір'ям і має широку область застосування у таких сферах, як міжнародні порівняльні дослідження, моніторинг якості освіти в масштабах країни, ліцензування та державна акредитація навчальних закладів, атестація учнів та студентів, вчителів та викладачів, перевірка професійної придатності фахівців тощо.

Останнім часом в галузі освіти спостерігається зростання інтересу до автоматизації різних видів навчальної та адміністративної діяльності. В процесі навчання це стосується передусім інформатизації контролю результатів навчання. Найпопулярнішим видом такого контролю є тестування, що ґрунтується на діалозі між інформаційною системою та користувачем. Зростання продуктивності технічних засобів, зниження цін на

обчислювальну техніку, поява якісних та потужних систем програмування збільшило потребу в системах, що дозволяють об'єктивно, швидко та надійно оцінювати знання учнів, пропонуючи цікаві форми взаємодії з ними.

Здійснення контролю та досягнення результатів навчання з використанням сучасних засобів інформаційних технологій у процесі навчання порівняно з іншими методами контролю має ряд очевидних переваг, серед яких: високий ступінь стандартизації, об'єктивність оцінки результатів, зручна кількісна форма вираження результатів, підвищена стійкість до фальсифікацій, висока швидкість обробки результатів, єдність вимог до всіх учнів, виключення суб'єктивізму при оцінці результатів.

При цьому зручність наявності кількісних показників виражається в можливості порівняння знань та вмінь одних учнів з іншими або відстеженні динаміки засвоєння знань одним учнем у процесі навчання.

Застосування комп'ютерів в навчальному процесі як засобу контролю якості знань отримує все більше поширення. Практика показує, що за наявності науково-методичної підготовки автоматизація контролю дозволяє помітно підвищити, передусім, індивідуальність самого контролю, варіювати його в залежності від здібностей та освітніх цінностей учнів. Застосування комп'ютера для контролю якості знань учнів припиняє бути фронтальним, набуває ознак індивідуального підходу, що враховує не тільки знання, але й індивідуальний темп навчання. Крім того, автоматизований контроль підвищує об'єктивність самого контролю, дозволяє оцінювати якість знань не тільки "взагалі", але забезпечує кількісну оцінку якості засвоєння певної частини навчального курсу. Ще одним важливим аспектом застосування автоматизованого контролю, як показала практика, стає стимулювання та мотивування учнів до самоосвітньої діяльності. За умови дотримання

належних дидактичних правил, методи автоматизованого контролю надають інформацію про якість знань не тільки викладачу, але й самим учням.

Ці дидактичні можливості засобів комп'ютерного автоматичного контролю якості знань актуалізували численні дослідження даної проблематики. Автоматичний контроль якості знань за допомогою комп'ютера має багато дидактичних, методичних та технологічних можливостей, які продовжують розвиватися завдяки швидкому вдосконаленню засобів обчислювальної техніки та програмного забезпечення. Незважаючи на численні дослідження даної проблематики, наукова дослідженість проблеми, як і раніше, не є вичерпною.

Розвиток обчислювальної техніки не тільки розширює можливості автоматизації в навчанні, але також розширює область застосування від природничих до гуманітарних і соціальних дисциплін. Використання автоматизованого контролю якості знань показало підвищення освітньої ефективності контролю в більш ніж два рази. Час відіграє важливу роль у ефективності контролю, і тому результат контролю якості знань повинен бути доступний негайно після виконання роботи.

Комп'ютерне тестування може проводитись у різних формах, що відрізняються за технологією об'єднання завдань у тест.

Перша форма тестування є найпростішою. Готовий тест, який може бути стандартизованим або призначеним для поточного контролю, вводиться в спеціальну оболонку, функції якої можуть різнитися за ступенем повноти. Зазвичай при підсумковому тестуванні оболонка дозволяє пред'являти завдання на екрані, оцінювати результати їх виконання, формувати матрицю результатів тестування, обробляти її та шкалювати перші бали учнів, переводячи їх на одну зі стандартних шкал для видачі кожному з них тестового балу та протоколу їх оцінок за завданнями тесту.

Друга форма комп'ютерного тестування полягає в автоматизованій генерації тестів з використанням інструментальних засобів. Форми створюються з банку каліброваних тестових завдань зі стійкими статичними характеристиками перед або під час тестування. Калібрування досягається завдяки попередній роботі з формування банку, параметри якого отримують на репрезентативній вибірці учнів, зазвичай, протягом 3-4 років з допомогою бланкових тестів. Суттєва валідність та паралельність варіантів забезпечуються строго регламентованим відбором завдань кожного варіанта відповідно до специфікації тесту.

Комп'ютерне адаптивне тестування - **третя форма**, що базується на адаптивних тестах. Адаптивність полягає в тому, що учням не потрібно робити завдання тесту, які вони виконають занадто легко або занадто складно. Тому запропоновано оптимізувати складність завдань, адаптуючи її до рівня підготовки кожного учасника тестування, та скоротити довжину тесту, виключивши деякі завдання.

Вибір форми тестування залежить від мети тестування, змісту тесту та технічних можливостей, також рівень підготовленості вчителя у галузі теорії та методики тестового контролю знань важливий. Найефективніші тести містять широкий зміст та охоплюють глибші рівні знань.

Система комп'ютерного контролю має відповідати таким вимогам:

- тестові питання та варіанти відповідей мають бути чіткими та зрозумілими;
- комп'ютерний тест має бути простим у використанні;
- оцінка ступеня правильності відповіді на кожне питання має бути включена до тестової системи;

- тест повинен містити достатню кількість питань для охоплення всього необхідного матеріалу;
- питання мають подаватися у випадковому порядку, щоб унеможливити механічне запам'ятовування їх послідовності;
- варіанти відповідей мають слідувати у випадковому порядку;
- необхідно обмежувати час відповідей та проводити облік часу.

Обрання СКБД

Система керування базами даних, скор. СКБД (англ. Database Management System, скор. DBMS) - сукупність програмних та лінгвістичних засобів загального або спеціального призначення, що забезпечують управління створенням та використанням баз даних.

СКБД - комплекс програм, що дозволяють створити базу даних (БД) та маніпулювати даними (вставляти, оновлювати, видаляти та вибирати).

Система забезпечує безпеку, надійність зберігання та цілісність даних, а також надає засоби для адміністрування БД.

Типи СКБД:

1. Реляційні.
2. Ключ-значення.
3. Документні.
4. Графові.
5. Колонкові.

Реляційні СКБД.

Класичні, реляційні СКБД найчастіше використовують для побудови рішень OTP (Online Transaction Processing). У таких рішеннях СКБД працює з невеликими за розмірами транзакціями, але такими, що йдуть великим потоком, і при цьому від системи потрібен мінімальний час відгуку, а також

можливість, за певних умов, скасувати будь-які зміни виконуваних у рамках транзакції. Якщо ви будуєте систему, в рамках якої потрібно зберігати значну кількість сутностей (таблиць), з різними типами зв'язків між ними (один-до-одного, один-до-багатьох, багато-багатьох), то це швидше за все про реляційні СКБД. Найбільш відомі СКБД такого типу – Oracle, Microsoft SQL, PostgreSQL, MySQL.

СКБД типу ключ-значення

Один із найпростіших типів СКБД. У спрощеному вигляді, це певна таблиця з унікальним ключем і власне пов'язаним з ним значенням, в якому може бути будь-що. Найчастіше такі СКБД застосовують для кешування, т.к. вони дуже швидко працюють, а це і не складно, коли є унікальний ключ і запит повертає тільки одне значення. У деяких представників даних СКБД є можливість працювати повністю в пам'яті, а також є можливість задавати термін життя запису, після закінчення якого записи будуть автоматично видалятися. Найбільш відомі СКБД такого типу - Redis та Memcached.

Документні СКБД

Документні або документно-орієнтовані СКБД - це один з найпопулярніших різновидів NoSQL СКБД, де основною одиницею логічної моделі даних є документ - структурований текст, з певним синтаксисом.

Іноді зустрічаються думки, що модель даних у документних БД схожа на модель даних в об'єктно-орієнтованих базах даних. У цьому є частка правди, єдина реальна різниця між ними полягає в тому, що бази даних документів зберігають лише стан, але не поведінку.

Так само сама назва "документо-орієнтована" часом вводить в оману, і мені зустрічалися колеги, які вважали, що це база для систем документообігу. Ні це не так.

Цікаво, що документні СКБД розвиваються досить активно, і зараз деякі з них, зокрема, підтримують перевірку схеми.

Відомими представниками таких СКБД є CouchDB, MongoDB, Amazon DocumentDB, OpenTriviaDB

Графові СКБД

Графові СКБД - специфічний тип, призначені для роботи з графами, з їх вузлами, властивостями та довільними відносинами між вузлами.

Дуже простий приклад, це організація зв'язків у різного типу соціальних мереж, де потрібно зберігати зв'язки між користувачами (вузлами) за різними критеріями (родинні зв'язки, колеги, спільні інтереси).

Відомі представники цього типу скбд – Neo4j, Amazon Neptune, InfiniteGraph, InfoGrid

Колонкові СКБД

Колонкові СКБД дуже схожі на реляційні. Вони також складаються з рядків, які мають атрибути, а рядки групуються в таблицях. Відмінності в логічних моделях несуттєві, а на рівні фізичного зберігання даних відмінності значні.

У реляційних СКБД дані зберігаються "построчно", це означає, що для зчитування значення певної колонки, доведеться прочитати практично весь рядок, як мінімум від першої до потрібної колонки. У колонковій СКБД дані зберігаються "поколоночно", тобто. колонка – це як окрема таблиця. Відповідно, читання відбуватиметься з конкретного стовпця відразу. Насправді це реально працює дуже швидко (перевірено мною на кількох реалізованих сховищах даних).

Основні переваги колонкових СКБД – ефективне виконання складних аналітичних запитів на великих обсягах, і легка, практично миттєва, зміна

структури таблиць з даними, плюс суттєва компресія та стиснення, що дозволяє значно економити місце.

Яскраві представники колонкових СКБД – Sybase IQ (нині SAP IQ), Vertica, ClickHouse, Google BigTable, InfoBright, Cassandra.

При виборі типу СКБД слід, перш за все, виходити з типу розв'язуваних завдань, типів даних, перспектив зростання і масштабування.

OpenTriviaDB – це безкоштовний онлайн-ресурс, який надає загальнодоступний набір запитань та відповідей для використання у створенні тестів, вікторин, ігор та інших освітніх програм. Він містить величезну базу даних питань з різних тем і галузей знань, які можна використовувати у різних освітніх контекстах. OpenTriviaDB надає API для доступу до його бази даних, що дозволяє розробникам легко інтегрувати його до своїх програм.

OpenTriviaDB - це хмарна база даних (Cloud-based Database) типу NoSQL, що використовує формат JSON для зберігання та надання інформації у форматі питання-відповідь для використання у додатках, іграх та інших проектах. Таким чином, OpenTriviaDB не відноситься до типу реляційних СУБД, а скоріше до типу документоорієнтованих баз даних.

Функція randomize OpenTriviaDB використовується для генерації випадкових чисел, які можуть бути використані для створення випадкових питань і варіантів відповідей. Щоразу, коли функція викликається, вона генерує нове випадкове число на основі поточного часу та інших факторів, таких як розташування користувача та налаштування комп'ютера. Це дозволяє створювати випадкові тести та питання, що робить процес тестування більш об'єктивним та надійним. Для данного проекту обрана реляційна СКБД Open TriviaDB.

Обрання середовища розробки.

Visual Studio Code – це повнофункціональний текстовий редактор для редагування локальних файлів чи бази коду. Він включає різні функції для редагування бази коду, яка допомагає розробникам відстежувати зміни. Різні функції, що підтримуються в VScode:

1. Підсвічування синтаксису.
2. Авто відступом.
3. Розпізнавання типів файлів.
4. Бічна панель із файлами вказаного каталогу.
5. Макрос.
6. Плагін та пакети.

Vs code використовується як інтегрований редактор розробки (IDE), як Sublime і NetBeans. Поточна версія редактора Vs code сумісна з різними операційними системами, такими як Windows, Linux та MacOS.

Обрання мови програмування.

JavaScript — це динамічна мова комп'ютерного програмування. Він легкий і найчастіше використовується як частина веб-сторінок, реалізації яких дозволяють сценарію на стороні клієнта взаємодіяти з користувачем і створювати динамічні сторінки. Це інтерпретована мова програмування з об'єктно-орієнтованими можливостями.

Головні переваги мови програмування JavaScript:

- JavaScript — це легка інтерпретована мова програмування;
- призначений для створення мережевих додатків;
- доповнює та інтегрується з Java;
- доповнює та інтегрується з HTML;
- відкрита та кросплатформна.

1.2. Аналіз відомих методів автоматичної генерації тестових завдань

Комп'ютерне тестування є однією з найпоширеніших форм перевірки знань, але з розвитком нових технологій і підвищенням рівня інформатизації суспільства та освіти, ефективний контроль знань стає ще важливішим. Для генерації тестових завдань використовують такі підходи, як параметризовані тести, семантичні мережі та понятійно-тезова модель (ПТМ) з її модифікаціями.

Метод параметризованих задач є ефективним інструментом для створення відкритих тестових завдань. Застосування принципу фасетності дозволяє генерувати кілька варіантів завдання в одному питанні. Кожен учень отримує лише один варіант з фасета. Шаблон питання доповнюється параметром, значення якого генерується в межах заданих границь. Шаблон є заготовкою тексту, який можна змінювати відповідно до заданого алгоритму.

{Зимовий, літній} МУССОН ДУЄ

1. із суші на морі

2. з моря на сушу

У фасетному завданні представлені два варіанти напряму Муссона - зимовий та літній. Цей метод генерації завдань називається параметризованим, у якому шаблон питання створюється автором, а параметр, значення якого генерується у заданих межах, доповнюється на момент видачі. Використовуючи цей метод, можна створювати безліч варіантів завдань на основі одного шаблону, змінюючи параметри на вході. Метод параметризованих питань передбачає створення автором шаблону питання. На момент виконання, шаблон доповнюється параметром, значення якого знаходиться в заданих межах. Шаблон зазвичай містить заготовку тексту, в якій деякі елементи можна змінювати відповідно до встановленого алгоритму. Основна ідея методу полягає в тому, що шляхом зміни параметрів

на вході, ми можемо отримати безліч нових варіантів питань. Один з недоліків полягає в складнощах створення шаблонів питань. Проте, перевага методу полягає в тому, що при малій кількості шаблонів можна згенерувати велику кількість різних питань. Наприклад, якщо параметр x може мати 10 різних значень, а параметр y - 15, то ми можемо отримати 150 різних варіантів питань на виході.

Генерація питань на основі алгоритмів є частковим випадком параметризованих завдань. Для створення питання учневі пропонується певний програмний код, який реалізує конкретний алгоритм. Учень повинен визначити значення деякого параметра алгоритму, тим самим демонструючи своє розуміння мови програмування. Суть методу полягає в наступному: на основі умови виходу з циклу можна створити генератор питань.

Метод семантичних мереж використовуються для автоматичної генерації тестових питань на основі бази знань, яку заповнює експерт з предметної області. Структура бази знань представлена тріадою: "поняття" - "ставлення" - "поняття". При формуванні тестів опускається одне з понять в тріаді. Хоча цей метод є ефективним, його недоліки полягають у великих трудових витратах на створення бази знань, необхідності залучення експертів та інженерів знань. Семантичні мережі не є найкращим варіантом для формування якісних тестових завдань в освіті.

Метод генерації тестових завдань на основі понятійно-тезової моделі (ПТМ) є способом формування тестових завдань, який використовує модель ПТМ для формалізації змісту навчального матеріалу. Семантичні елементи ПТМ виділяються з тексту навчального фрагмента, а тестові завдання формуються шляхом вибору контрольної понятійно-тезової пари та дистракторів на основі інших понять чи тез. Цей підхід має простоту формування бази знань та високу якість тестових завдань порівняно з іншими

методами автоматичної побудови тестів. Детальніше про способи формування тестів на базі ПТМ можна дізнатися з попередніх робіт.

1.3. Характеристика існуючих методів та програмних засобів оцінювання знань.

Шкала оцінювання є письмові інструкції або роз'яснення про дії або відповіді індивідів і визначає важливі компоненти оцінюваної роботи. Шкала оцінювання використовують для оцінювання великого набору дій, есе, контрольних робіт, проектів, рефератів, усних доповідей, презентацій, ситуаційних завдань. Оцінювання завдань із вільно конструйованою відповіддю. Завдання з вільно конструйованою відповіддю дозволяють перевірити якісне володіння змістом курсів, що перевіряються, і складними інтелектуальними вміннями: логічно і послідовно викладати свої думки, наводити рішення задачі з обґрунтуванням окремих етапів, застосовувати теоретичні знання для обґрунтування та пояснення запропонованих явищ і процесів, використовувати знання , прогнозувати наслідки, формулювати гіпотези, робити висновки та обґрунтовувати свою точку зору, наводити аргументи на підтримку певної точки зору або в спростуванні її та ін. схем (шкал). Основним недоліком при оцінюванні вільно конструйованих відповідей є вплив суб'єктивності оцінок експертів.

Для зменшення розбіжностей в оцінках експертів розробляється шкала оцінювання (Оцінна схема, рубрика), що включає оптимальну систему критеріїв оцінювання відповідей. Шкала оцінювання завдань із вільно конструйованою відповіддю включає: критерії оцінювання та варіанти (варіант, зразок) правильних відповідей (елементів відповіді), можливі рішення. Схема оцінювання залежить від дисципліни, вона спрямовано оцінку результату, певного набору дій, мети оцінювання і забезпечує

зворотний зв'язок учням щодо того, як поліпшити їх дії. Відповідно до заданих критеріїв оцінювання формулюється завдання. Це означає, що завдання, що оцінюється після прочитання, повинен зрозуміти, яке завдання йому належить виконати, і з якою повнотою він повинен дати відповідь для отримання максимальної оцінки. У завданнях з вільно конструюваною відповіддю, що перевіряють навчальні досягнення (предметні знання та вміння), а не комунікативні вміння, від оцінюваних не повинно вимагатися написання довгого тексту. У завданні повинні бути надані рекомендації про передбачувану довжину відповіді (можлива кількість речень, певна частина сторінки тощо).

Розробка шкали оцінювання виконання завдання з вільно конструюваною відповіддю передбачає такі дії:

1. Визначення підходу до оцінки виконання завдання (інтегральний, або аналітичний).
2. Виділення основних критеріїв оцінювання виконання завдання, за якими можна диференціювати роботи учнів.
3. Підбір прикладів робіт учнів, демонструють весь спектр виконання (від відмінної до незадовільної).
4. Визначення шкали вимірювання для оцінки різного рівня освоєння даного критерію оцінювання та довжини шкали.
5. Експериментальна перевірка розробленого варіанта критеріїв.
6. Доопрацювання системи критеріїв.
7. Додатковий підбір прикладів робіт учнів, які демонструють весь спектр виконання завдання.

Шкала оцінювання може вважатися об'єктивною і надійною, якщо кілька експертів, які перевіряють одну й ту саму роботу, роблять однаковий висновок про підготовку учня (виставляють однакові бали) або, якщо один і

той самий перевіряючий однаково оцінює ту саму роботу учня, перевіряючи її через деякий час.

Статистичні показники, що характеризують якість розробленої шкали оцінювання:

1. Розподіл відповідей оцінюваних. Якщо систему балів підібрано правильно, то відповіді розподіляються по всіх балах. Немає жодного балу, який було отримано незначною групою оцінюваних (менше 5%).
2. Надійність перевірки та повторної перевірки. При повторній перевірці роботи збіг має бути не менш як у 85% випадків.

Можна запропонувати такі види шкал оцінювання:

- аналітична шкала більш достовірна, валідна, дозволяє точніше діагностувати та прогнозувати навчальний процес, а також сприяє взаєморозумінню між викладачем та учням;
- інтегральна (цілісна) шкала розглядає роботу загалом, а не по аспектам.

Аналітичні шкали використовують для оцінки якості засвоєння матеріалу за окремими розділами навчальних дисциплін або всієї дисципліни загалом. Також аналітичні шкали застосовують для оцінювання окремих, індивідуальних частин результату або роботи, для подальшого підсумовування безлічі оцінок, щоб отримати загальний результат. Часто аналітичні шкали використовуються для оцінювання завдань чи робіт, у яких можуть бути одна або кілька прийнятних відповідей.

Можливі сфери застосування аналітичних шкал:

1. Партнерство групи (робота у колективі) – спілкування, готовність відповідати питання, внесок у дії групи.
2. Участь – готовність взяти відповідальність, співробітництво з групою, час, витрачений виконання своєї частини.

3. Домашня робота - своєчасність, охайність, дотримання інструкцій, ретельність.
4. Проекти – творчий потенціал, стиль, пошук рішення, аргументування, пояснення.
5. Поведінка – вміння слухати, грубість, взаємодію Космосу з іншими студентами, шанобливість.
6. Завдання з вільно конструюваною відповіддю – стиль, ясність, граматика.
7. Тайм-менеджмент - оцінювання можливості керувати часом.

Інтегральна шкала вимагає, щоб оцінювали весь процес чи результат загалом, не оцінюючи складники окремо. Такі шкали зазвичай використовуються, коли можна припуститися помилок у деякій частині процесу, а оцінці підлягає кінцевий результат, наприклад, курсова робота, проект, кейс, окремо доповідь з випускної кваліфікаційної роботи. Інтегральні шкали слід рекомендувати з метою оцінки компетенцій під час роботи державної екзаменаційної комісії.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ГЕНЕРАЦІЇ ТЕСТОВИХ ЗАВДАНЬ ТА ОЦІНКИ ЗНАНЬ НА ОСНОВІ ФУНКЦІЇ RANDOMIZE

2.1. Постановка задачі

У сучасному світі, де технології активно використовуються в освітніх процесах, дедалі більше уваги приділяється розробці автоматизованих систем для формування і оцінювання знань студентів. Одним із важливих етапів таких систем є генерація тестових завдань. В цьому процесі важливо забезпечити не тільки випадковість тестових питань, але й їх якість та адекватність рівню знань студентів. Функція randomize, що використовується в багатьох мовах програмування, дає змогу генерувати випадкові числа, які можуть бути використані для формування тестових завдань. Однак, важливо розробити систему, яка використовує цю функцію вірно і ефективно, забезпечуючи якість тестових завдань та їх адаптацію до різних рівнів знань студентів.

Для досягнення мети дипломної роботи потрібно вирішити наступні завдання:

- розглянути теоретичні аспекти генерації тестових завдань на основі функції randomize;
- дослідити існуючі підходи до автоматизованої генерації тестових завдань та їх переваги та недоліки;
- розробити алгоритм та програмну реалізацію системи генерації тестових завдань на основі функції randomize;
- провести експериментальне дослідження розробленого алгоритму та порівняти його ефективність з іншими методами генерації тестових завдань;

- оцінити якість згенерованих тестів та їх придатність для оцінювання знань студентів;
- запропонувати можливі напрями подальшого розвитку системи та покращення її функціональних характеристик.

2.2. Метод автоматичної генерації тестових завдань.

Метод автоматичної генерації тестових завдань (Automatic Test Generation Method) - це процес автоматичної генерації тестових даних для тестування програмного забезпечення. Метод використовує різні алгоритми та стратегії для генерації вхідних даних, що можуть бути використані для перевірки роботи програмного забезпечення. Метод автоматичної генерації тестових завдань зазвичай використовується для автоматичної перевірки коректності роботи програмного забезпечення та для покращення якості тестів. Застосування методу дозволяє зменшити кількість помилок, виявлених під час тестування, збільшити покриття коду тестами та скоротити час, необхідний для ручної генерації тестів.

Одним з підходів до автоматичної генерації тестових завдань є генерація випадкових тестів на основі певних вимог та обмежень. Для цього використовуються різноманітні методи, такі як генетичні алгоритми, стохастичні методи та інші. Важливо пам'ятати, що такі методи можуть давати лише загальні результати, які потрібно перевіряти вручну. Для досягнення успіху в автоматичній генерації тестових завдань, необхідно використовувати різноманітні методи та стратегії, які відповідають вимогам конкретної задачі тестування. Також важливо проводити ефективний аналіз результатів тестування та вдосконалювати методи генерації тестів для досягнення максимальної ефективності.

Функція Randomize: що вона робить і як використовувати її в програмуванні.

Функція Randomize є однією з найбільш корисних функцій, які забезпечують генерацію випадкових чисел в програмуванні. Вона дозволяє заповнити різні змінні випадковими значеннями, що може бути корисним у багатьох випадках, наприклад, при створенні ігор, генерації тестових завдань, аналізу даних тощо.

Основна мета функції Randomize полягає в тому, щоб унеможливити передбачуваність результатів виконання програми при використанні функцій, які генерують випадкові числа. Це досягається за допомогою так званого "насіння" генератора випадкових чисел. Насінням є число, яке передається в функцію Randomize. Кожен раз, коли викликається функція генерації випадкових чисел, вона використовує насіння для створення наступного значення.

Основний синтаксис функції Randomize наступний:

Randomize [(seed)]

seed - це необов'язковий параметр, який визначає насіння генератора випадкових чисел. Якщо параметр seed не вказується, то насіння генератора випадкових чисел визначається системним часом.

Функція randomize є важливою складовою в системі генерації тестових завдань та оцінки знань. Вона використовується для створення випадкових чисел, які використовуються при генерації тестових завдань та оцінці відповідей.

Зазвичай, функція randomize використовується разом з функцією random, яка генерує випадкові числа в заданому діапазоні. За допомогою

функції randomize визначається вихідний параматер, що використовується в функції random для генерації випадкових чисел. Наприклад, якщо потрібно створити тест з п'яти питань, то за допомогою функції randomize можна визначити випадковий порядок питань. Також, використання функції randomize дозволяє забезпечити більшу варіативність в тестових завданнях, оскільки вони будуть складатися з різних питань та варіантів відповідей. При проектуванні системи генерації тестових завдань і оцінки знань, використання функції randomize дозволяє створювати більш різноманітні тестові завдання та забезпечити об'єктивну оцінку знань.

Крім того, використання випадкових чисел при генерації тестів робить процес навчання більш цікавим та захоплюючим для учнів. Однак, слід мати на увазі, що використання випадкових чисел при генерації тестів може також привести до некоректної оцінки знань. Тому необхідно забезпечувати високу якість тестів та проводити їх оцінку з урахуванням всіх можливих факторів.

Стохастичні методи.

Стохастичні методи в автоматичній генерації тестових завдань використовуються для створення випадкових тестів, які можуть покривати якомога більше можливих сценаріїв взаємодії користувача з програмним продуктом.

Одним з популярних стохастичних методів є метод Монте-Карло, який використовує випадковість для генерації тестів. Він може бути застосований для генерації тестових даних, які можуть включати різні значення параметрів програмного продукту.

Ще одним стохастичним методом є генетичний алгоритм, який моделює природний відбір для створення оптимальних тестових наборів. Генетичний алгоритм використовує еволюційний підхід, щоб знаходити оптимальні рішення в задачах генерації тестів.

Генетичний Алгоритм.

Вперше комп'ютерним моделюванням еволюційного відбору зайнявся Нільс Барічеллі в 1954 році. Визнання генетичного алгоритму як методу розв'язання оптимізаційних задач відбулося в 1960-70 роках в результаті робіт Інго Рехенберга та Джона Голланда. Генетичний алгоритм (ГА) належить до стохастичних методів і ґрунтується на принципі природного еволюційного відбору. Ідея генетичних алгоритмів запозичена з живої природи і полягає у моделюванні еволюційного процесу, кінцевою метою якого є отримання оптимального рішення складної комбінаторної задачі. У описі методу використовуються спрощені біологічні терміни. Основною ідеєю ГА є боротьба за існування між рішеннями задачі. Кожне рішення записується у вигляді деякого вектора значень ,який називається хромосомою або особою. Сукупність рішень називають популяцією. Кожна особа в популяції оцінюється значенням цільової функції, розрахованої на основі значень з хромосоми. Більш перспективні рішення проходять на наступний етап і впливають на "потомство", тобто на вновину генеровані рішення. Необхідно представити задачу у формі генотипу, тобто вектора значень (генів), щоб її розв'язок можна було записати як генотип.

Наприклад, для пошуку максимуму функції десяти аргументів y(x1, x2,..., x10) генотип буде складатися з десяти чисел (значень x1, x2,..., x10), а значення функції y від цих чисел характеризує придатність генотипу, тому оптимізована функція також називається функцією придатності або фітнес-функцією. Стандартний генетичний алгоритм починає свою роботу з формування початкової популяції як кінцевого набору допустимих рішень задачі (хромосом). Ці рішення можуть бути вибрані як випадковим чином, так і з допомогою приблизних алгоритмів.

На кожному кроці еволюції формується нова популяція з допомогою ймовірнісного оператора селекції. Для кожної нової особини популяції вибираються два рішення, або "батьки" (у загальному випадку батьків може бути більше двох). Ймовірність вибору рішення як батьківського безпосередньо пропорційна його якості. Під час схрещування двох особин вони за допомогою певного правила обмінюються елементами хромосом (такий обмін називають кросовером). Кросовер (використовується також назва скрещування) створює з батьківських хромосом одну або кілька нових хромосом. У найпростішому випадку кросовер в генетичному алгоритмі реалізується шляхом розрізання вектора хромосом батьків у випадковій позиції та обміну частинами векторів (мал. 2.1).

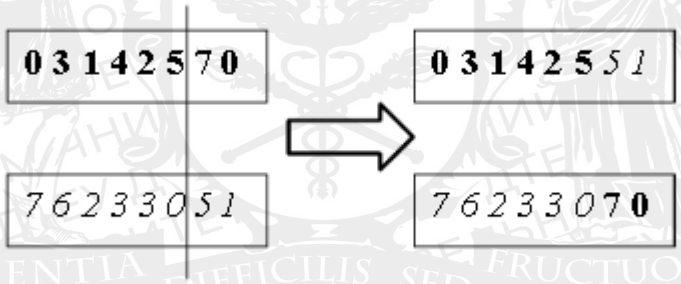

Рис. 2.1. Приклад кросінговерінга.

Оператор схрещування за цими рішеннями будує нове рішення, яке потім піддається невеликим випадковим модифікаціям (мутаціям). Мутація - перетворення хромосоми, випадково змінюючи одну або декілька її позицій (генів). Найбільш поширеним видом мутацій є випадкова зміна тільки одного з генів хромосоми. Мутаціям піддаються не всі хромосоми, а тільки вибрані з певною ймовірністю. В результаті отримується нова популяція, а стара повністю або частково знищується. Популяція наступного покоління в більшості реалізацій генетичних алгоритмів містить стільки ж особин,

скільки й початкова, але завдяки відбору пристосованість в ній в середньому вище. Тепер описані процеси відбору, схрещування та мутації повторюються вже для цієї популяції і т.д. У кожному наступному поколінні спостерігається виникнення нових хороших і поганих рішень розглянутої задачі. Завдяки тому, що кращі рішення відбираються з більшою ймовірністю, їх кількість збільшується. Іноді в ГА зберігають декілька кращих хромосом поточного покоління (елітні хромосоми). Схема роботи ГА може бути представлена наступним чином:

- генерація початкової популяції;
- обчислення цільової функції по кожній особині (хромосомі);
- вибір особин та їх схрещування (кроссінговер);
- випадкові зміни особин (мутація).

2.3. Метод оцінювання результату проходження тесту.

Метод оцінювання результату проходження тесту - це важлива складова процесу тестування, яка дозволяє отримати об'єктивну оцінку знань учня. Існує кілька методів оцінювання результату проходження тесту, які використовуються в різних сферах, включаючи освіту, рекрутинг, медицину та інші. Один з найпоширеніших методів - це метод правильних відповідей, який полягає в підрахунку кількості правильних відповідей учня. Кожне запитання в тесті має визначену кількість балів, яку можна отримати за правильну відповідь. Загальна кількість балів обчислюється як сума балів за правильні відповіді на кожне запитання. Часто використовується шкала оцінювання, яка визначає, яку кількість балів необхідно отримати для отримання певної оцінки.

Інший метод - це метод часткового кредитування, який використовується для складних тестів зі складними запитаннями. За цим методом бали надаються за правильність окремих частин запитання, а не за правильну відповідь в цілому. Це дозволяє учневі отримувати кредит за часткові знання.

Крім того, існує метод відкритих відповідей, використовуваний для тестів, в яких немає конкретних відповідей або відповіді можуть бути різними. За цим методом, учень отримує бали за викладення своїх думок та ідей, а також за вміння аргументувати свої відповіді. Усі ці методи оцінювання результату проходження тесту можуть бути застосовані в різних випадках і залежать від типу тесту та мети його використання. Один з підходів до оцінювання результату проходження тесту полягає в застосуванні балів. Кожне запитання оцінюється певною кількістю балів, які надаються за правильну відповідь. Після проходження тесту бали підсумовуються, і отримується загальний бал за тест. Цей метод підходить для тестів з фіксованою кількістю запитань, де кожне запитання має однакову важливість. Інший підхід - це визначення проценту правильних відповідей. Після проходження тесту кількість правильних відповідей ділиться на загальну кількість запитань, і отримується процент правильних відповідей. Цей метод підходить для тестів з різною кількістю запитань, де деякі запитання можуть бути важливішими за інші.

Також можна застосовувати метод нормалізованих балів. У цьому методі кожне запитання оцінюється від 0 до 1, де 1 - це повна правильна відповідь, а 0 - це повна неправильна відповідь. Загальний бал обчислюється шляхом підсумовування оцінок за кожне запитання та ділення на максимально можливу кількість балів. Цей метод підходить для тестів з різними кількостями запитань та різною важливістю запитань. Крім того, можна застосовувати методи порівняння результатів тестування між різними

учасниками або групами учасників. Наприклад, можна порівняти середній бал за тест між різними групами студентів або середній бал одного студента.

Метод часткових балів.

Один з методів оцінювання результату проходження тесту - метод часткових балів (Partial Credit Method). Цей метод використовується для оцінювання багатовідповідних тестів, де кожен питання має кілька відповідей, кожна з яких може мати свою вагу. Метод часткових балів дозволяє врахувати як правильні, так і частково правильні відповіді на питання. Кожне питання має певну кількість балів, які можуть бути зараховані при відповіді на нього. Якщо користувач відповів правильно на всі підпитання, він отримує максимальну кількість балів за питання. Якщо відповідь на підпитання правильна частково, то користувач отримує меншу кількість балів, але більшу, ніж у випадку, коли він відповів неправильно. Таким чином, користувач отримує кредитні бали за кожну правильну відповідь, причому бали залежать від того, наскільки повно він відповів на питання.

При застосуванні методу часткових балів кожне питання оцінюється окремо, і результати питань потім обчислюються для отримання загального балу за тест. Крім того, метод часткових балів може бути використаний для оцінювання якісних відповідей на питання, наприклад, для тесту з відкритими питаннями. Щоб використовувати метод часткових балів, необхідно визначити вагу кожної відповіді на кожне питання. Це можна зробити шляхом обговорення з експертами у відповідній області знань або за допомогою емпіричних досліджень.

Крім того, слід визначити правильні відповіді для кожного тестового завдання. Це може бути зроблено вручну, якщо тестові завдання не занадто

багаті, або ж за допомогою алгоритмів автоматичного визначення правильних відповідей.

Метод максимального балу.

Ще одним методом для оцінювання результатів проходження тестів є метод "максимального балу". У цьому методі кожне завдання має певний бал, і після проходження тесту сума балів за всі завдання визначає кінцевий результат. Наприклад, припустимо, що тест складається з 10 завдань, кожне завдання має бали від 1 до 10, і для успішного проходження тесту необхідно набрати не менше 50 балів. У цьому випадку при проходженні тесту людині буде присвоєний кінцевий бал, який буде сумою балів за всі завдання. Якщо він набрав 50 і більше балів, тест вважається пройденим, в іншому випадку - не пройденим. Метод максимального балу зазвичай використовується в тестах, де кожне завдання має рівну важливість. Він також може бути використаний разом з іншими методами, наприклад, з методом "правильних відповідей", щоб покращити точність оцінки результатів тестування. Однак, необхідно враховувати, що цей метод не завжди є найкращим варіантом для оцінювання результатів тестування, оскільки він не враховує різну складність завдань та можливу похибку при оцінці.

2.4 Алгоритм генерації тестових завдань з використанням функції Randomize.

Алгоритм генерації тестових завдань з використанням функції Randomize складається з наступних етапів:

- визначення кількості завдань в тесті;

- визначення типів завдань, які будуть використовуватися в тесті (наприклад, вибір одного правильного відповіді, вибір кількох правильних відповідей, встановлення відповідності, заповнення пропусків тощо);
- визначення складності завдань відповідно до рівня знань, на який тест призначений;
- генерація запитань з використанням функції Randomize для створення випадкових чисел;
- генерація варіантів відповідей для кожного запитання з використанням функції Randomize;
- перевірка правильності відповідей та корекція завдань за необхідності.

Рис. 2.2. Схема алгоритму генерації тестових завдань.

Рис. 2.3. Схема алгоритму налаштування системи тестування.

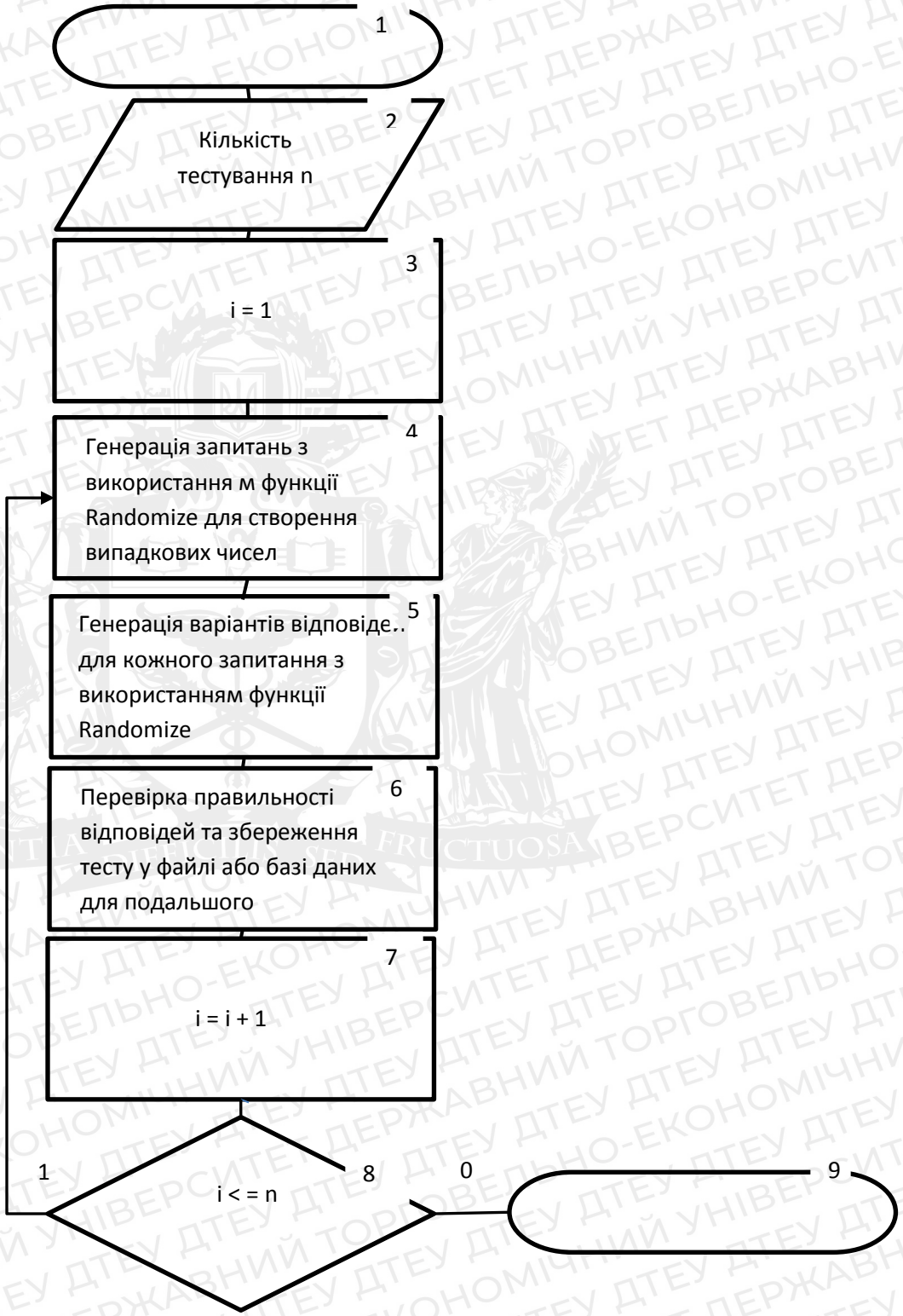

Рис 2.4. Схема алгоритму проведення тестування.

Використання функції Randomize дозволяє створювати випадкові тестові завдання, що робить процес тестування більш об'єктивним та надійним. Проте, для отримання якісних результатів необхідно також враховувати різноманітні фактори, такі як специфіка тестової теми, вік та рівень навчання тестованих тощо.

1. Визначення кількості завдань у тесті.

Першим кроком є визначення кількості завдань, які будуть включені до тесту. Це може залежати від мети тестування, складності завдань та часу, доступного для проходження тесту.

2. Визначення типів завдань, які будуть використовуватись у тесті.

Наступним кроком є визначення типів завдань, які будуть використовуватись у тесті. Наприклад, це може бути вибір однієї правильної відповіді, вибір кількох правильних відповідей, відповідність, заповнення перепусток і т.д. Визначення типів завдань залежить від того, які знання чи навички мають бути перевірені у тесті.

3. Визначення складності завдань відповідно до рівня знань, на який тест призначений.

На цьому етапі визначається складність завдань, які будуть включені до тесту, відповідно до рівня знань, на який тест призначений. Наприклад, якщо тест призначений для студентів початкових класів, то завдання мають бути більш простими, ніж якщо тест призначений для студентів старших класів чи дорослих.

4. Генерація питань із використанням функції Randomize для створення випадкових чисел.

За допомогою функції Randomize генеруються випадкові числа, які будуть використовуватися для створення питань. Це дозволяє створювати

випадкові питання та тести, що робить процес тестування більш об'єктивним та надійним.

5. Генерація варіантів відповідей кожного питання з допомогою функції Randomize.

За допомогою функції Randomize генеруються варіанти відповідей для кожного питання. Залежно від типу питання, для кожної відповіді може бути визначена вага чи правильність відповіді.

6. Перевірка правильності відповідей та коригування завдань за потреби.

Після генерації тесту необхідно перевірити правильність відповідей та коректність формулювань питань. Якщо виявляються помилки чи неточності, необхідно внести коригування завдання.

Використання функції Randomize для генерації тестів дозволяє створювати випадкові питання та варіанти відповідей, що робить процес тестування більш об'єктивним та надійним. Завдяки випадковому вибору питань та відповідей, тести стають різноманітнішими та важче піддаються підготовці заздалегідь. Це може зменшити можливість шахрайства та підвищити достовірність результатів тестування.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ГЕНЕРАЦІЇ ТЕСТОВИХ ЗАВДАНЬ ТА ОЦІНКИ ЗНАНЬ НА ОСНОВІ ФУНКЦІЇ RANDOMIZE

3.1 Архітектура системи генерації тестових завдань та оцінки знань користувачів

Архітектура системи генерації тестових завдань та оцінки знань користувачів включає наступні компоненти:

1. Користувацький інтерфейс - це компонент, що взаємодіє з користувачем системи. Користувач може вводити запити на генерацію тестових завдань, заповнювати тести та переглядати результати оцінювання своїх знань.
2. Система управління базою даних (СУБД) - це компонент, що відповідає за зберігання тестових завдань, відповідей користувачів та інформації про їхній прогрес. СУБД може бути віддаленою або локальною, залежно від потреб користувачів та системних вимог.
3. Генератор тестових завдань - це компонент, що відповідає за створення тестів на основі заданих критеріїв. Генератор може використовувати алгоритми випадкової генерації питань та варіантів відповідей, або мати задану базу питань та відповідей, з якої він буде брати завдання для тесту.
4. Система оцінювання знань - це компонент, що відповідає за перевірку правильності відповідей користувачів на тестові завдання. Система може використовувати різні алгоритми оцінювання, такі як шкала оцінок або процент правильних відповідей.
5. Система звітування - це компонент, що відповідає за формування звіту про результати тестування. Система може включати у себе статистику про правильність відповідей, час проходження тесту, рівень складності завдань та іншу корисну інформацію.

6. Алгоритми машинного навчання - це комп'ютерні алгоритми, що дозволяють робити передбачення та приймати рішення на основі аналізу великих обсягів даних. Вони використовуються в різних галузях, таких як бізнес, наука, медицина, фінанси та інші.

3.2. Програмна реалізація системи автоматизованого тестування знань користувачів.

Реалізація системи генерації тестових завдань та оцінки знань користувачів за допомогою React та Open TriviaDB є сучасним та ефективним рішенням для створення освітніх платформ та додатків. Вона дозволяє швидко та легко створювати інтерактивні тести, які допомагають користувачам перевіряти свої знання та підвищувати свій рівень освіти. Система використовує React для створення інтерфейсу користувача, який дозволяє користувачам легко взаємодіяти з додатком і проходити тести. React є одним з найбільш популярних фреймворків для розробки інтерфейсів веб-додатків і надає потужні інструменти для створення високоякісних інтерфейсів. React також забезпечує високу продуктивність, що дозволяє програмі працювати швидко та чуйно. Для генерації тестових завдань система використовує API Open TriviaDB, що надає доступ до величезної бази питань та відповідей на різні теми. Open TriviaDB забезпечує високу точність та різноманітність питань, що робить тести більш цікавими та корисними для користувачів. Система працює наступним чином:

1. Користувач запускає програму та бачить головний екран, на якому відображаються доступні категорії тестів.
2. Користувач вибирає категорію тестів, яку хоче пройти.
3. Система звертається до API Open TriviaDB, щоб отримати випадковий набір питань із вибраної категорії.

4. Система генерує тестове завдання на основі отриманих питань та відображає його на екрані користувача.
5. Користувач відповідає на запитання та система оцінює його відповіді.
6. Після завершення тесту система відображає користувачеві результати тесту та надає можливість повторити тест або вибрати іншу категорію.

Реалізація системи генерації тестових завдань та оцінки знань користувачів за допомогою React та Open TriviaDB забезпечує високу продуктивність, точність та різноманітність питань, що робить програму цікавою та корисною для користувачів. Ця система є чудовим інструментом для створення освітніх платформ і додатків, які можуть допомогти людям вчитися та підвищувати свої знання у різних галузях. Вона дозволяє створювати тести та опитування за допомогою різних типів питань, у тому числі множинного вибору, правда-брехня та інших.

Завдяки використанню React і Open TriviaDB, розробники можуть створювати інтерфейс користувача, який легко розуміти і використовувати. Крім того, система генерації тестових завдань може використовуватися для створення різних типів завдань, які можуть допомогти людям вчитися та розвиватися у різних галузях знань.

Однією з ключових переваг цієї системи є її простота використання. Розробники можуть легко налаштувати систему для створення тестових завдань та опитувань за допомогою кількох простих кроків. Вони можуть вибрати потрібний тип питань, налаштувати параметри тесту та додати необхідну інформацію для кожного питання.

Крім того, система генерації тестових завдань забезпечує високий рівень автоматизації. Вона може швидко створювати та оцінювати тести, що дозволяє економити час та скорочувати витрати на створення освітніх матеріалів.

На закінчення, система генерації тестових завдань та оцінки знань на основі React та Open TriviaDB є потужним інструментом для створення освітніх програм та платформ. Вона дозволяє створювати тести та опитування різних типів, автоматизувати процес їх створення та оцінки, що спрощує роботу викладачів та покращує якість навчання.

Подивимось докладніше на компоненти додатку:

```
import React from 'react'
import { useGlobalContext } from './context'

import SetupForm from './SetupForm'
import Loading from './Loading'
import Modal from './Modal'
function App() {
  const {
    waiting,
    loading,
    questions,
    index,
    correct,
    nextQuestion,
    checkAnswer,
  } = useGlobalContext()

  if (waiting) {
    return <SetupForm />
  }
  if (loading) {
    return <Loading />
  }
  const { question, incorrect_answers, correct_answer } = questions[index]
  // const answers = [...incorrect_answers, correct_answer]
  let answers = [...incorrect_answers]
  const tempIndex = Math.floor(Math.random() * 4)

  if (tempIndex === 3) {
    answers.push(correct)
  } else {
    answers.push(answers[tempIndex])
    answers[tempIndex] = correct_answer
  }
```

Рис. 3.1. Компонент програми, який визначає основну структуру та поведінку сайту.

```
  return (
    <main>
      <Modal />
      <section className='quiz'>
        <p className='correct-answers'>
          correct_answers : {correct} / {index}
        </p>
        <article className='container'>
          <h2 dangerouslySetInnerHTML={{ __html: question }} />
          <div className='btn-container'>
            {answers.map((answer, index) => {
              return (
                <button
                  key={index}
                  className='answer-btn'
                  onClick={() => checkAnswer(correct_answer === answer)}
                  dangerouslySetInnerHTML={{ __html: answer }}
                />
              )
            })}
          </div>
        </article>
        <button className='next-question' onClick={nextQuestion}>
          next question
        </button>
      </section>
    </main>
  )
}

export default App

```

Рис. 3.2. Компонент програми, який визначає основну структуру та поведінку сайту (2).

Цей код є основним компонентом програми, створеної для генерації тестів. На початку коду імпортуються необхідні компоненти з інших файлів програми та функція useGlobalContext з контексту, що містить стан програми.

Потім всередині функції App() використовуються змінні та функції, визначені в контексті, такі як waiting, loading, questions, index, correct, nextQuestion і checkAnswer.

Перші дві умови if використовуються для відображення форми налаштування та індикатора завантаження, якщо програма знаходиться в очікуванні або завантаженні даних.

Потім з масиву питання вибирається поточне питання і його варіанти відповідей. Варіанти відповідей перемішуються випадковим чином за допомогою функції Math.random() та методу масиву push(). У цій реалізації правильна відповідь завжди знаходиться на останній позиції.

Далі використовується JSX, щоб відобразити поточне запитання та варіанти відповідей у інтерфейсі користувача. Кожен варіант відповіді представлений у вигляді кнопки, і при натисканні на неї викликається функція checkAnswer(), яка перевіряє правильність відповіді та оновлює стан програми.

Нарешті, відображається кнопка "Наступне питання", яка при натисканні викликає функцію nextQuestion(), яка збільшує індекс поточного питання та переходить до наступного питання.

Отже, весь код є логікою програми для генерації тестових завдань з використанням React і Open TriviaDB. Він відповідає за відображення питань та варіантів відповідей у інтерфейсі користувача та за обробку відповідей користувача, оновлення лічильника правильних відповідей і перехід до наступного питання.

Також дуже цікаво розібрати модуль context:

```
20  const API_ENDPOINT = 'https://opentdb.com/api.php?'
21
22  const url = ''
23
24  const AppContext = React.createContext()
25
26  const AppProvider = ({ children }) => {
27    const [waiting, setWaiting] = useState(true)
28    const [loading, setLoading] = useState(false)
29    const [questions, setQuestions] = useState([])
30    const [index, setIndex] = useState(0)
31    const [correct, setCorrect] = useState(0)
32    const [error, setError] = useState(false)
33    const [quiz, setQuiz] = useState({
34      amount: 5,
35      category: 'general',
36      difficulty: 'easy',
37    })
38    const [isModalOpen, setIsModalOpen] = useState(false)
39
40    const fetchQuestions = async (url) => {
41      setLoading(true)
42      setWaiting(false)
43      const response = await axios(url).catch((err) => console.log(err))
44      if (response) {
45        const data = response.data.results
46        if (data.length > 0) {
47          setQuestions(data)
48          setLoading(false)
49          setWaiting(false)
50          setError(false)
```

Рис. 3.3. Компонент програми, що відповідає за створення та управління глобальним станом(контекстом) компонентів.

```
60    const nextQuestion = () => {
61      setIndex((oldIndex) => {
62        const index = oldIndex + 1
63        if (index > questions.length - 1) {
64          openModal()
65          return 0
66        } else {
67          return index
68        }
69      })
70    }
71
72    const checkAnswer = (value) => {
73      if (value) {
74        setCorrect((oldState) => oldState + 1)
75      }
76      nextQuestion()
77    }
78
79    const openModal = () => {
80      setIsModalOpen(true)
81    }
82
83    const closeModal = () => {
84      setWaiting(true)
85      setCorrect(0)
86      setIsModalOpen(false)
87    }
```

Рис. 3.4. Компонент програми, що відповідає за створення та управління глобальним станом(контекстом) компонентів (2).

Цей код є частиною програми для генерації тестів на React і містить компонент AppProvider, який створює контекст AppContext та надає його дочірнім компонентам.

Спочатку коду визначено об'єкт table, який містить відповідності між різними категоріями тестів та відповідними номерами категорій в API Open Trivia DB. Потім визначено константи API_ENDPOINT та url.

Далі слідує визначення функціонального компонента AppProvider, який приймає як властивості дочірні компоненти та містить стан програми, а також функції для зміни станів.

Основні стани програми включають waiting, loading, questions, index, correct, error, quiz і isModalOpen.

Стан waiting визначає, чи чекає програма відповіді від сервера, стан loading - чи завантажує додаток дані, стан questions - містить перелік питань, стан index - поточний індекс питання, стан correct - кількість правильних відповідей, стан error - визначає, чи виникли помилки при завантаженні Дані, стан quiz - містить параметри вибору користувача, такі як кількість питань, категорія і складність, а стан isModalOpen - визначає, чи відкрито модальне вікно.

Далі визначені функції fetchQuestions, nextQuestion, checkAnswer, openModal, closeModal, handleChange і handleSubmit, які змінюють відповідні стану програми у відповідь введення користувача та відповіді від сервера.

Нарешті, компонент AppProvider забезпечує доступ до стану програми та функцій через контекст AppContext, який експортується для використання в інших компонентах за допомогою функції useGlobalContext.

Важливою частиною додатку також є модуль Modal.

```jsx
import React from 'react'
import { useGlobalContext } from './context'

const Modal = () => {
  const { isModalOpen, closeModal, correct, questions } = useGlobalContext()
  return (
    <div
      className={`${
        isModalOpen ? 'modal-container isOpen' : 'modal-container'
      }`}
    >
      <div className='modal-content'>
        <h2>congrats!</h2>
        <p>
          You answered {((correct / questions.length) * 100).toFixed(0)}% of
          questions correctly
        </p>
        <button className='close-btn' onClick={closeModal}>
          play again
        </button>
      </div>
    </div>
  )
}

export default Modal
```

Рис. 3.5. компонент Modal, який відповідає за створення, управління та вигляд модального вікна.

Цей код визначає компонент Modal, який є модальним вікном, що відображає результати вікторини і дозволяє користувачеві розпочати нову гру. Компонент використовує стан глобального контексту, який визначений у файлі context.js за допомогою хука useGlobalContext.

Коли користувач закінчує вікторину, стан isModalOpen стає true, що показує модальне вікно. Результати гри, які зберігаються в станах correct та questions, відображаються всередині модального вікна. Користувач може розпочати нову гру, натиснувши кнопку "play again", яка викликає функцію closeModal. Компонент Modal повертає елемент div із класом "modal-container", який відображає модальне вікно, що містить заголовок "congrats!"

та результати гри у параграфі. Компонент також включає кнопку "play again", яка викликає функцію closeModal. Заключний модуль, перед демонстрацією додатку, є SetupForm .

```
import React from 'react'
import { useGlobalContext } from './context'

const SetupForm = () => {
  const { quiz, handleChange, handleSubmit, error } = useGlobalContext()
  return (
    <main>
      <section className='quiz quiz-small'>
        <form className='setup-form'>
          <h2>Setup Trivia</h2>
          {/* amount */}
          <div className='form-control'>
            <label htmlFor='amount'>number of questions</label>
            <input
              type='number'
              name='amount'
              id='amount'
              className='form-input'
              value={quiz.amount}
              onChange={handleChange}
              min={1}
              max={50}
            />
          </div>
          {/* category */}
          <div className='form-control'>
            <label htmlFor='category'>category</label>
            <select
              name='category'
              id='category'
              className='form-input'
              value={quiz.category}
```

Рис. 3.6. Компонент SetupForm, який відповідає за відображення форми, обробку та валідацію введених у неї відповідей.

```
          >
            <option value='general'>General Knowledge</option>
            <option value='computers'>Computers</option>
            <option value='books'>Books</option>
            <option value='maths'>Mathematics</option>
            <option value='science'>Science & Nature</option>
            <option value='music'>Music</option>
            <option value='film'>Film</option>
            <option value='theatre'>Musicals & Theatres</optio
            <option value='sports'>Sports</option>
            <option value='anime'>Anime & Manga</option>
            <option value='video'>Video Games</option>
            <option value='history'>History</option>
            <option value='politics'>Politics</option>
          </select>
        </div>
        {/* difficulty */}
        <div className='form-control'>
          <label htmlFor='difficulty'>Difficulty</label>
          <select
            name='difficulty'
            id='difficulty'
            className='form-input'
            value={quiz.difficulty}
            onChange={handleChange}
          >
            <option value='easy'>Easy</option>
            <option value='medium'>Medium</option>
            <option value='hard'>Hard</option>
          </select>
        </div>
```

Рис. 3.7. Компонент SetupForm, який відповідає за відображення форми, обробку та валідацію введених у неї відповідей (2).

Цей код відображає форму для настроювання вікторини. Він використовує хук useGlobalContext для отримання стану та методів з контексту, наданого компонентом ContextProvider.

У формі є три поля для налаштування:

1. Кількість питань (від 1 до 50).
2. Категорія питань (загальне знання, комп'ютери, книги, математика, наука, музика, фільми, мюзикли та театри, спорт, аніме та манга, відеоігри, історія, політика).
3. Рівень складності питань (легкий, середній чи важкий).

Також у формі є кнопка "Start" для початку вікторини та повідомлення про помилку, якщо не вдається згенерувати питання з вибраними опціями. При зміні значень полів використовується метод handleChange з контексту, а при відправленні форми викликається метод handleSubmit.

3.3 Технологія використання розробленої системи тестування.

Технологія використання розробленої системи тестування полягає в наступному:

1. Користувач заповнює форму з параметрами тестування, використовуючи компонент SetupForm, реалізований на React.
2. При натисканні на кнопку "Start" з форми, запит відправляється на сервер Open TriviaDB для генерації питань, відповідно до введених параметрів.
3. Отримані питання обробляються і зберігаються в контексті за допомогою хука useGlobalContext, який дозволяє ділитися даними між компонентами.
4. Користувач відповідає на питання, використовуючи компонент Question, який збирає відповіді та обробляє їх.
5. Після відповіді на останнє питання, результат тестування відображається за допомогою модального вікна Modal, яке показує відсоток правильних відповідей та кнопку "Play Again".

Отже, технологія використання системи тестування базується на використанні компонентів React для розробки інтерфейсу користувача та Open TriviaDB для генерації питань, які оброблюються та відображаються на сторінці за допомогою збереження даних у контексті та відображення модального вікна з результатами.

Подивимось докладніше, з першого кроку:

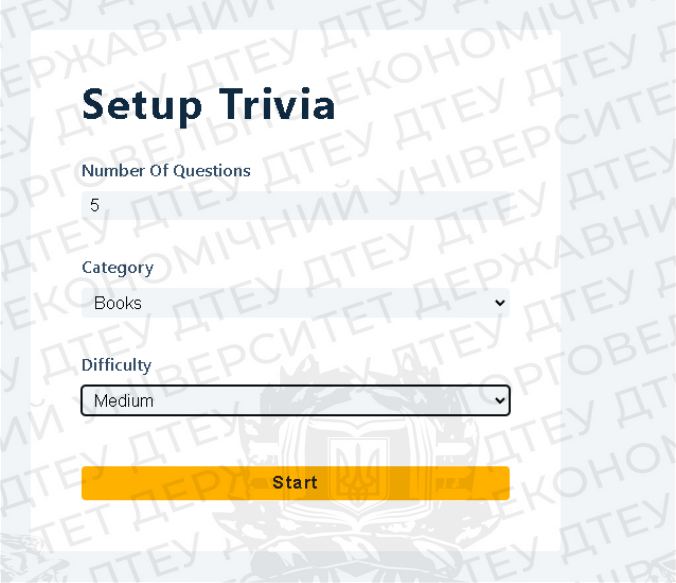

Рис. 3.8. Крок 1. Налаштування системи проведення тестування.

1. Користувач відкриває веб-додаток з тестами, написаний на React.
2. Користувач бачить форму налаштувань тестування, де можна вибрати кількість питань, категорію тестування та його складність.
3. Користувач вводить налаштування тестування та натискає на кнопку "Start".

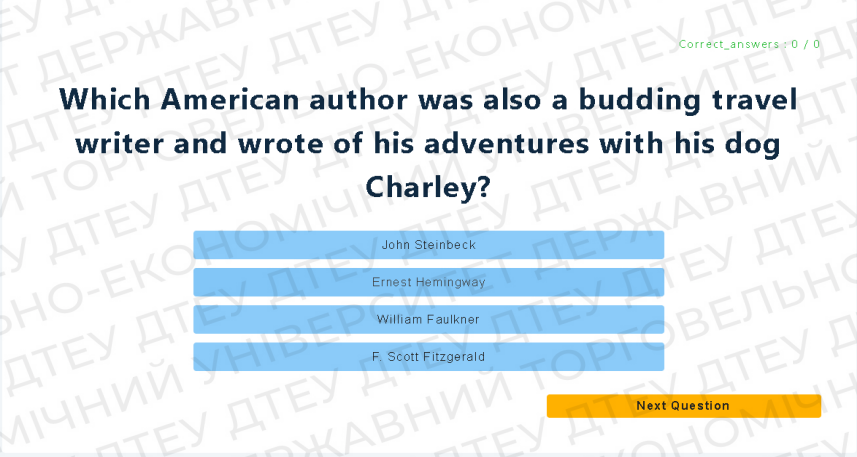

Рис. 3.9. Крок 2. Генерація запитань.

4. Відбувається запит до API Open TriviaDB, де за допомогою вибраних налаштувань генеруються питання для тестування.
5. Система отримує запит з питаннями і зберігає їх у стані додатку.
6. Користувач бачить екран тестування, де він відповідає на запитання, обираючи один з варіантів відповіді.

Після того, як користувач відповів на всі запитання, система відображає результат тестування. Результати тестування відображаються у вигляді відсотку правильних відповідей. Користувач має можливість зіграти в тест знову або повернутися до форми налаштувань тестування:

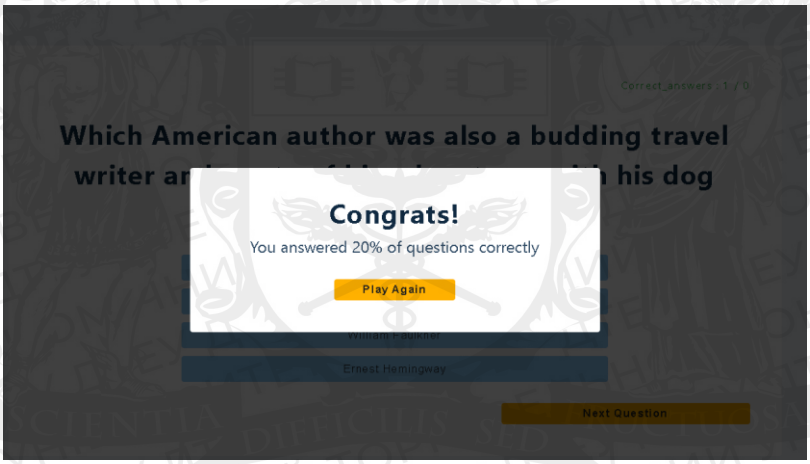

Рис. 3.10. Фінальний крок. Відображення кількості правильних відповідей.

Додаток для тестування з використанням бібліотеки React та API Open TriviaDB дозволяє генерувати питання з різних категорій, складностей та кількостей за допомогою форми на сторінці налаштувань. Крім того, додаток підтримує відслідковування кількості правильних та неправильних відповідей користувача під час тестування, а також відображає результати в модальному вікні після закінчення тесту. Також, додаток є корисним інструментом для тестування знань, особливо для тих, хто вивчає певну тему

та хоче перевірити свої знання в цій галузі. Він може бути використаний як для самостійного вивчення, так і для використання в освітніх закладах. Додаток має простий та зрозумілий інтерфейс, що дозволяє користувачам легко використовувати його без додаткової підготовки або навичок.

ВИСНОВКИ

Дипломна робота на тему "Автоматизована система генерації тестових завдань на основі функції Randomize, з використанням React та Trivia DB" дозволила розробити веб-додаток для генерації тестових завдань з різних категорій та складностей. Використання функції Randomize забезпечує випадковість тестів, що дозволяє користувачам отримувати унікальні тестові завдання при кожному запуску.React був обраний як фреймворк для розробки веб-додатку з використанням його компонентної структури та підходу до роботи зі станами. Використання Trivia DB як джерела даних дозволило легко та швидко отримувати запитання та відповіді до тестів.

Додаток має зручний та простий інтерфейс, що дозволяє користувачам встановлювати параметри тестування, отримувати запитання та перевіряти свої відповіді. Розробка веб-додатку показала важливість використання сучасних технологій та даних, що є доступними з Інтернету, для створення корисних та простих у використанні програм.

У результаті дипломної роботи було успішно розроблено та протестовано веб-додаток для автоматизованої генерації тестових завдань. Розроблений додаток може бути використаний у навчальних закладах, компаніях для проведення тестувань, а також для особистого навчання та розвитку знань та вмінь користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The React Handbook: Your Guide to Building Web Applications with React by Flavio Copes
2. Learning React: A Hands-On Guide to Building Web Applications Using React and Redux, Second Edition by Kirupa Chinnathambi
3. Trivia DB API Documentation: https://opentdb.com/api_config.php
4. Pro React 16 by Adam Freeman
5. React: Up & Running: Building Web Applications by Stoyan Stefanov and Chad R. Adams
6. Testing React: A Practical Guide to Testing React Applications, Components, and Libraries by Daniel Irvine
7. Building Progressive Web Applications with React: Create lightning fast web apps with native power using React and Firebase by Scott Domes
8. React Native: Building Mobile Apps with JavaScript by Vladimir Novick
9. Learning React: A Hands-On Guide to Building Web Applications Using React and Redux by Kirupa Chinnathambi
10. The Road to learn React: Your journey to master plain yet pragmatic React.js by Robin Wieruch.
11. Клименко В. І., Мазурець О. В. Аналіз сучасних методів генерації тестових завдань. Актуальні проблеми комп’ютерних технологій : збірник наукових праць за матеріалами X міжнародної науково-технічної конференції. (м. Хмельницький, 31 травня 2016 р.).
12. Rullian Zhao and Qing Li., “Automatic test generation for dynamic data structure”, Technical report, Beijing University of Chemical Technology, 2006.

ДОДАТОК

Лістинг коду основних модулів

```
import React from 'react'
import { useGlobalContext } from './context'

import SetupForm from './SetupForm'
import Loading from './Loading'
import Modal from './Modal'
function App() {
  const {
    waiting,
    loading,
    questions,
    index,
    correct,
    nextQuestion,
    checkAnswer,
  } = useGlobalContext()

  if (waiting) {
    return <SetupForm />
  }
  if (loading) {
    return <Loading />
  }
  const { question, incorrect_answers, correct_answer } = questions[index]
  // const answers = [...incorrect_answers, correct_answer]
  let answers = [...incorrect_answers]
  const tempIndex = Math.floor(Math.random() * 4)

  if (tempIndex === 3) {
    answers.push(correct)
  } else {
    answers.push(answers[tempIndex])
    answers[tempIndex] = correct_answer
  }
```

```
  return (
    <main>
      <Modal />
      <section className='quiz'>
        <p className='correct-answers'>
          correct_answers : {correct} / {index}
        </p>
        <article className='container'>
          <h2 dangerouslySetInnerHTML={{ __html: question }} />
          <div className='btn-container'>
            {answers.map((answer, index) => {
              return (
                <button
                  key={index}
                  className='answer-btn'
                  onClick={() => checkAnswer(correct_answer === answer)}
                  dangerouslySetInnerHTML={{ __html: answer }}
                />
              )
            })}
          </div>
        </article>
        <button className='next-question' onClick={nextQuestion}>
          next question
        </button>
      </section>
    </main>
  )
}

export default App
```

```
import axios from 'axios'
import React, { useState, useContext, useEffect } from 'react'

const table = {
  general: 9,
  books: 10,
  film: 11,
  music: 12,
  theatre: 13,
  video: 15,
  science: 17,
  computers: 18,
  maths: 19,
  sports: 21,
  history: 23,
  politics: 24,
  anime: 31,
}

const API_ENDPOINT = 'https://opentdb.com/api.php?'

const url = ''

const AppContext = React.createContext()

const AppProvider = ({ children }) => {
  const [waiting, setWaiting] = useState(true)
  const [loading, setLoading] = useState(false)
  const [questions, setQuestions] = useState([])
  const [index, setIndex] = useState(0)
  const [correct, setCorrect] = useState(0)
  const [error, setError] = useState(false)
```

```
33  const [quiz, setQuiz] = useState({
34    amount: 5,
35    category: 'general',
36    difficulty: 'easy',
37  })
38  const [isModalOpen, setIsModalOpen] = useState(false)
39
40  const fetchQuestions = async (url) => {
41    setLoading(true)
42    setWaiting(false)
43    const response = await axios(url).catch((err) => console.log(err))
44    if (response) {
45      const data = response.data.results
46      if (data.length > 0) {
47        setQuestions(data)
48        setLoading(false)
49        setWaiting(false)
50        setError(false)
51      } else {
52        setWaiting(true)
53        setError(true)
54      }
55    } else {
56      setWaiting(true)
57    }
58  }
```

```
  const checkAnswer = (value) => {
    if (value) {
      setCorrect((oldState) => oldState + 1)
    }
    nextQuestion()
  }

  const openModal = () => {
    setIsModalOpen(true)
  }

  const closeModal = () => {
    setWaiting(true)
    setCorrect(0)
    setIsModalOpen(false)
  }

  const handleChange = (e) => {
    const name = e.target.name
    const value = e.target.value
    setQuiz({ ...quiz, [name]: value })
  }

  const handleSubmit = (e) => {
    e.preventDefault()
    const { amount, category, difficulty } = quiz

    const url = `${API_ENDPOINT}amount=${amount}&category=${table[category]}&difficulty=${difficulty}&type=multiple`
    fetchQuestions(url)
  }


  return (
    <AppContext.Provider
      value={{
        waiting,
        loading,
        questions,
        index,
        correct,
        error,
        isModalOpen,
        nextQuestion,
        checkAnswer,
        closeModal,
        quiz,
        handleChange,
        handleSubmit,
      }}
    >
      {children}
    </AppContext.Provider>
  )
}
// make sure use
export const useGlobalContext = () => {
  return useContext(AppContext)
}

export { AppContext, AppProvider }
```

```jsx
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'
import App from './App'
import { AppProvider } from './context'

ReactDOM.render(
  <React.StrictMode>
    <AppProvider>
      <App />
    </AppProvider>
  </React.StrictMode>,
  document.getElementById('root')
)

```

```jsx
import React from 'react'

const Loading = () => {
  return (
    <main>
      <div className='loading'></div>
    </main>
  )
}

export default Loading

```

```jsx
import React from 'react'
import { useGlobalContext } from './context'

const Modal = () => {
  const { isModalOpen, closeModal, correct, questions } = useGlobalContext()
  return (
    <div
      className={`${
        isModalOpen ? 'modal-container isOpen' : 'modal-container'
      }`}
    >
      <div className='modal-content'>
        <h2>congrats!</h2>
        <p>
          You answered {((correct / questions.length) * 100).toFixed(0)}% of
          questions correctly
        </p>
        <button className='close-btn' onClick={closeModal}>
          play again
        </button>
      </div>
    </div>
  )
}

export default Modal

```

```jsx
import React from 'react'
import { useGlobalContext } from './context'

const SetupForm = () => {
  const { quiz, handleChange, handleSubmit, error } = useGlobalContext()
  return (
    <main>
      <section className='quiz quiz-small'>
        <form className='setup-form'>
          <h2>Setup Trivia</h2>
          {/* amount */}
          <div className='form-control'>
            <label htmlFor='amount'>number of questions</label>
            <input
              type='number'
              name='amount'
              id='amount'
              className='form-input'
              value={quiz.amount}
              onChange={handleChange}
              min={1}
              max={50}
            />
          </div>
          {/* category */}
          <div className='form-control'>
            <label htmlFor='category'>category</label>
            <select
              name='category'
              id='category'
              className='form-input'
              value={quiz.category}
```

```
          >
            <option value='general'>General Knowledge</option>
            <option value='computers'>Computers</option>
            <option value='books'>Books</option>
            <option value='maths'>Mathematics</option>
            <option value='science'>Science & Nature</option>
            <option value='music'>Music</option>
            <option value='film'>Film</option>
            <option value='theatre'>Musicals & Theatres</option>
            <option value='sports'>Sports</option>
            <option value='anime'>Anime & Manga</option>
            <option value='video'>Video Games</option>
            <option value='history'>History</option>
            <option value='politics'>Politics</option>
          </select>
        </div>
        {/* difficulty */}
        <div className='form-control'>
          <label htmlFor='difficulty'>Difficulty</label>
          <select
            name='difficulty'
            id='difficulty'
            className='form-input'
            value={quiz.difficulty}
            onChange={handleChange}
          >
            <option value='easy'>Easy</option>
            <option value='medium'>Medium</option>
            <option value='hard'>Hard</option>
          </select>
        </div>
```

```
            value={quiz.difficulty}
            onChange={handleChange}
          >
            <option value='easy'>Easy</option>
            <option value='medium'>Medium</option>
            <option value='hard'>Hard</option>
          </select>
        </div>
        {error && (
          <p className='error'>
            can't generate questions, please try different options
          </p>
        )}
        <button type='submit' onClick={handleSubmit} className='submit-btn'>
          Start
        </button>
      </form>
    </section>
  </main>
  )
}

export default SetupForm

```