

ДЕРЖАВНИЙ УНІВЕРСИТЕТ  
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ  
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ  
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**КВАЛІФІКАЦІЙНА РОБОТА**

на тему: «Розробка веб-сервісу для організації тестування студентів, з використанням бібліотеки React»

на здобуття освітнього ступеня бакалавра  
зі спеціальності 121 Інженерія програмного забезпечення  
освітньо-професійної програми «Інженерія програмного забезпечення»

*Кваліфікаційна робота містить результати власних досліджень.  
Використання ідей, результатів і текстів інших авторів мають посилання  
на відповідне джерело*

\_\_\_\_\_

Вадим ВИСОЦЬКИЙ

(підпис)

Виконав: здобувач вищої освіти групи ПД-42

Вадим ВИСОЦЬКИЙ

Керівник: \_\_\_\_\_

Ірина ЗАМРІЙ

*д.т.н., доц., завідувач кафедри*

Рецензент: \_\_\_\_\_

**Київ 2024**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ  
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**  
**Навчально-науковий інститут інформаційних технологій**

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

**ЗАТВЕРДЖУЮ**

Завідувач кафедри

Інженерії програмного забезпечення

\_\_\_\_\_ Ірина ЗАМРІЙ

« \_\_\_\_ » \_\_\_\_\_ 2024 р.

**ЗАВДАННЯ  
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

\_\_\_\_\_ Висоцькому Вадиму Олеговичу

1. Тема кваліфікаційної роботи: Розробка веб-сервісу для організації тестування студентів, з використанням бібліотеки React»  
керівник кваліфікаційної роботи д.т.н., доц., завідувач кафедри ІІЗ Ірина ЗАМРІЙ

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи розробки веб-сервісів, опис методів генерації тестових завдань, технічна документація з описом бібліотеки React для розробки системи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд та аналіз існуючих методів та технологій для організації тестувань.

2. Проектування веб-сервісу для організації тестувань студентів.

3. Програмна реалізація та опис функціонування веб-сервісу для організації тестувань студентів з використанням бібліотеки React.

4. Тестування застосунку.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Компоненти додатку.
5. Приклад використання.
6. Екранні форми.
7. Апробація результатів дослідження

6. Дата видачі завдання «28» лютого 2024 р.

### КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Огляд існуючих алгоритмів генерації тестових завдань методів та програмних засобів оцінювання знань.	14.03-20.03.2024	
4	Проектування веб-сервісу для організації тестувань студентів з використанням бібліотеки React.	21.03-28.03.2024	
5	Програмна реалізація веб-сервісу	29.03-18.04.2024	
6	Тестування веб-сервісу	19.03-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач(ка) вищої освіти \_\_\_\_\_

(підпис)

Вадим ВИСОЦЬКИЙ

Керівник  
кваліфікаційної роботи \_\_\_\_\_

(підпис)

Ірина ЗАМРІЙ





## РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 42 стор., 12 рис., 19 джерела.

*Мета роботи* – підтримка процесу тестування студентів за допомогою веб-сервісу розробленого з використанням бібліотеки React.

*Об'єкт дослідження* – організація тестування студентів.

*Предмет дослідження* – веб-сервіс для організації тестування студентів.

*Короткий зміст роботи:* Робота містить огляд існуючих систем та методів генерації тестових даних, а також розробку програмного модуля для генерації тестових даних, придатного для використання в автоматизованих тестових програмах. Матеріали дипломної роботи можуть бути використані для наукових досліджень та в навчальному процесі фахівців з системного програмування та тестування. Розроблений програмний модуль є корисним для впровадження в автоматизовані тестові програмні продукти в різних сферах автоматизованого тестування програмного забезпечення. Тема дипломної роботи: "Розробка веб-сервісу для організації тестувань студентів з використанням бібліотеки React".

**КЛЮЧОВІ СЛОВА:** АВТОМАТИЗОВАНЕ ТЕСТУВАННЯ, ВЕБ-СЕРВІС, ГЕНЕРАЦІЮ ТЕСТОВИХ ДАНИХ, РЕАСТ БІБЛІОТЕК

## ЗМІСТ

ВСТУП .....	8
1 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ МЕХАНІЗМІВ АВТОМАТИЗОВАНОЇ СИСТЕМИ ОЦІНКИ ЗНАНЬ КОРИСТУВАЧІВ.....	9
1.1 Поняття та сутність автоматизованого оцінювання знань користувачів. ....	9
1.2 Аналіз відомих методів автоматичної генерації тестових завдань .....	14
1.3 Характеристика існуючих методів та програмних засобів оцінювання знань.....	16
2 ПРОЕКТУВАННЯ СИСТЕМИ ГЕНЕРАЦІЇ ТЕСТОВИХ ЗАВДАНЬ ТА ОЦІНКИ ЗНАНЬ НА ОСНОВІ ФУНКЦІЇ RANDOMIZE .....	20
2.1 Постановка задачі.....	21
2.2 Метод автоматичної генерації тестових завдань. ....	23
2.3 Метод оцінювання результату проходження тесту. ....	27
2.4 Алгоритм генерації тестових завдань з використанням функції Randomize. ....	29
2.5 Аналіз наукових публікацій та літератури з тематики розробки веб-сервісів на React .....	32
2.6 Аналіз сучасних рішень для організації тестувань та тестувань, використовуючи React. ....	33
2.7 Аналіз відповідності функціоналу веб-сервісу технічному завданню та вимогам. ....	35
2.8 Тестування функціоналу та ефективності розробленого веб-сервісу. ....	36
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ГЕНЕРАЦІЇ ТЕСТОВИХ ЗАВДАНЬ ТА ОЦІНКИ ЗНАНЬ НА ОСНОВІ ФУНКЦІЇ RANDOMIZE .....	38
3.1 Обрання СКБД.....	38
3.2 Обрання середовища розробки.....	41
3.3 Обрання мови програмування.....	42
3.4 Архітектура системи генерації тестових завдань та оцінки знань користувачів.....	42
3.5 Програмна реалізація системи автоматизованого тестування знань користувачів.....	43
3.6 Технологія використання розробленої системи тестування.....	51
ВИСНОВКИ.....	55
ПЕРЕЛІК ПОСИЛАНЬ .....	57
ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ .....	59
ДОДАТОК Б. ЛІСТИНГ ОСНОВНИХ МОДУЛІВ .....	67

## ВСТУП

У сучасному світі, освіта стає все важливішою та доступнішою. За останні кілька років, кількість онлайн-курсів та навчальних матеріалів у мережі значно збільшилась. Проте, для ефективного навчання необхідно не лише матеріали, але й засоби контролю знань.

Тому, організація тестувань та тестувань є важливим елементом навчального процесу. Онлайн-тестування дають змогу збирати відповіді студентів в електронному вигляді та отримувати результати навчання в режимі реального часу.

Ця дипломна робота присвячена розробці веб-сервісу для організації тестувань студентів з використанням бібліотеки React. Веб-сервіс забезпечує зручну та швидку організацію тестувань та тестувань, які можуть використовуватись для будь-яких типів навчальних курсів.

У рамках дослідження проведено огляд сучасних рішень для організації тестувань, зокрема засобів, що використовують React. На основі проведеного аналізу, розроблено веб-сервіс, який включає в себе можливість створення тестів з різними типами питань, генерацію звітів та результатів.

*Метою* дипломної роботи є розробка ефективного та зручного веб-сервісу для організації тестувань та тестувань студентів. Основна мета полягає в створенні високоякісного продукту, який можна використовувати в освітньому процесі різних навчальних закладів.

*Об'єкт* дослідження – організація тестування студентів.

*Предмет* дослідження – веб-сервіс для організації тестування студентів.

Дана дипломна робота буде корисною для фахівців у галузі освіти, розробників програмного забезпечення та будь-яких осіб, які цікавляться розробкою веб-сервісів та застосуванням React у практичних проектах. Результати дослідження можуть бути використані для подальшого розвитку та вдосконалення веб-сервісів для навчальних закладів, що дозволить забезпечити більш ефективний та зручний навчальний процес для студентів.



# **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 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 навчанців 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 іспитуваних, 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 статистичними 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.

Комп'ютерне тестування як форму педагогічного контролю доцільно застосовувати для організації контролю рівнів засвоєння, таких як визнання, відтворення та розуміння, та видів знань, таких як знання понять, термінів та фактів. За Б.С. Блумом, це часто рівні знання, розуміння, синтез, оцінка, застосування та аналіз.

**Блок знань** - це рівень запам'ятовування та відтворення матеріалу будь-якого рівня складності (факти, поняття, правила, риси, ознаки тощо). Метою рівня є те, що учень відтворює терміни, знає конкретні правила, поняття, факти тощо.

**Блок розуміння** - рівень засвоєння матеріалу та здатність до його перетворення та інтерпретації. Тут учень розуміє факти, правила, інтерпретує схеми, графіки. На основі наявних даних передбачаємо характер майбутніх наслідків.

**Блок синтезу** - вміння об'єднати елементи в нове ціле. У даному блоку учень виконує творчу роботу, пропонує план проведення якогось експерименту, використовує знання з декількох областей. Це творча переробка інформації для створення нового цілого.

**Блок оцінювання** - це вміння оцінити якість та розуміння матеріалу на основі критеріїв, розроблених учнем або викладачем. Учень може виділяти критерії та дотримуватися їх, бачить різноманітність критеріїв, оцінює відповідність висновків наявним даним, робить різницю.

**Блок застосування** - це вміння застосовувати правила, теорії, методи в конкретних ситуаціях та в нових умовах. Досягнення цілей - учень

використовує раніше отримані знання не тільки в стандартних умовах, але й в нових ситуаціях та правильно їх застосовує.

**Блок аналізу** - це вміння виділяти окремі елементи структури навчального матеріалу, визначати взаємозв'язок між елементами та логіку цього взаємозв'язку. Учень виділяє частини цілого та взаємозв'язки між ними, бачить упущення в логіці міркувань, робить різницю між фактами та наслідками, оцінює значимість даних.

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;
- метод тестування не може бути єдиним способом перевірки знань та умінь учнів, але може бути використаний поруч із традиційними формами перевірки результатів навчання.

## 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.

*1. {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. Тестируваний 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.

### **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 оволодіння 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 максимального бала. У завданнях з вільно конструйованою 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, аналітичний чи поелементний);
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 підбрано 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 чи не по аспектам. Враховує одночасно безліч факторів, а не оцінює кожен окремо. 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, граматики.

7. Тайм-менеджмент - оцінювання можливості керувати часом.

Дескриптори для детермінуючої (основної) ідеї (поелементне оцінювання):

Рівень 5 - ідея, що детермінує, відображає глибоке розуміння.

Рівень 4 – основна ідея змістовна, ясна.

Рівень 3 – основна ідея очевидна, але надто проста чи неоригінальна, шаблонна.

Рівень 2 - основна ідея поверхнева, запозичена або взагалі відсутня.

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.

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

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 параметр, 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.

## 2.1 Постановка задачі

У даному розділі буде розглянута постановка задачі дипломної роботи "Розробка веб-сервісу для організації тестувань студентів з використанням бібліотеки `React`".

Основною метою дипломної роботи є розробка ефективного та зручного веб-сервісу для організації тестувань та тестувань студентів з використанням бібліотеки `React`. Для досягнення даної мети необхідно вирішити такі задачі:

Провести огляд сучасних рішень для організації тестувань та тестувань, зокрема засобів, що використовують `React`. Для цього будуть проаналізовані наступні питання:

- Які існують рішення для організації тестувань та тестувань?
- Які інструменти використовуються для розробки веб-сервісів на `React`?
- Які є переваги та недоліки використання `React` для розробки веб-сервісів для організації тестувань та тестувань?

- Розробити веб-сервіс для організації тестувань та тестувань студентів з використанням бібліотеки React. Для цього необхідно вирішити такі задачі:
- Створити інтерфейс для створення тестів з різними типами питань.
- Реалізувати можливість генерації звітів та результатів навчання.
- Забезпечити зручну та швидку організацію тестувань та тестувань, які можуть використовуватись для будь-яких типів навчальних курсів.
- Протестувати розроблений веб-сервіс та провести аналіз результатів. Для цього необхідно вирішити такі задачі:
- Розробити тести для перевірки роботи веб-сервісу.
- Виконати тестування функціоналу та проаналізувати його результати.
- Виконати тестування на ефективність та проаналізувати результати.

Окрім того, для досягнення поставленої мети дипломної роботи необхідно розв'язати наступні завдання:

- Вивчити теоретичні основи розробки веб-сервісів та засобів їх розробки на React.
- Розробити технічне завдання на розробку веб-сервісу для організації тестувань та тестувань студентів.
- Розробити архітектуру веб-сервісу.
- Вибрати необхідні інструменти для розробки веб-сервісу.
- Реалізувати функціонал веб-сервісу та перевірити його на відповідність вимогам, використовуючи тести.
- Оформити документацію до проекту.

Для вирішення поставлених завдань дипломної роботи будуть використані такі методи дослідження:

- ✓ Аналіз наукових публікацій та літератури з тематики розробки веб-сервісів на React.
- ✓ Аналіз сучасних рішень для організації тестувань та тестувань, використовуючи React.
- ✓ Аналіз відповідності функціоналу веб-сервісу технічному завданню та вимогам.
- ✓ Тестування функціоналу та ефективності розробленого веб-сервісу.

Отже, цей розділ дипломної роботи містить постановку задачі та завдань, які необхідно вирішити для досягнення мети проекту. Для вирішення поставлених завдань будуть використані відповідні методи дослідження та інструменти розробки веб-сервісів на React.

## **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.

### **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.

Інші стохастичні методи включають в себе методи марківських процесів, згорткові нейронні мережі та інші. Використання стохастичних методів може допомогти забезпечити більшу покриття програмного продукту тестами, знизити час на створення тестів та збільшити ефективність процесу тестування.

### **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 (алелей), 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 вноvinу 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  $y(x_1, x_2, \dots, x_{10})$  генотип буде складатися з десяти чисел (значень  $x_1, x_2, \dots, x_{10}$ ), 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 (рисунок 2.1).

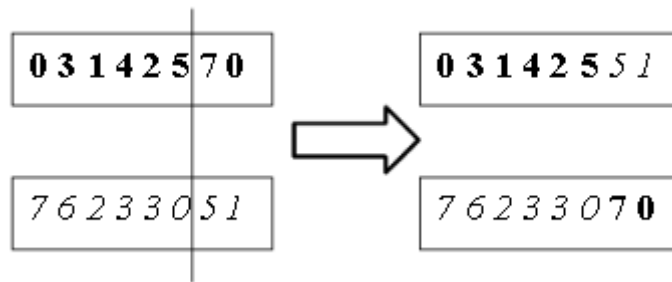


Рис. 2.1 Приклад кросінговерінга

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 (кросінговер).
- Випадкові зміни особин (мутація).
- Якщо виконано умову завершення, то кінець роботи, інакше перейти до п. 2.

### 2.3 Метод оцінювання результату проходження тесту

Метод оцінювання результату проходження тесту - це важлива складова процесу тестування, яка дозволяє отримати об'єктивну оцінку знань тестованої особи. 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 тестованої особи. 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 тестируваній особі 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, тестована особа 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 бали. 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 із методів оцінювання результату проходження тесту - метод часткових балів (Partial Credit Method). Цей метод використовується для оцінювання багатовідповідних тестів, де кожне 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.

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.

## 2.5 Аналіз наукових публікацій та літератури з тематики розробки веб-сервісів на React

Для проведення аналізу наукових публікацій та літератури з тематики розробки веб-сервісів на React були використані наступні джерела:

1. Kirupa Chinnathambi Learning React: A Hands-On Guide to Building Web Applications Using React and Redux. 2nd ed. Boston, 2018. 304 p.
2. Michele Bertoli React Design Patterns and Best Practices: Build easy to scale modular applications using the most powerful components and design patterns. Birmingham, 2017. 318 p.
3. Vipul Amler, Prathamesh Sonpatki ReactJS by Example - Building Modern Web Applications with React. Birmingham, 2018. 360 p.
4. Mark Tielens Thomas React in Action. New York, 2017. 318 p.

Загальний аналіз літератури показав, що React є однією з найпопулярніших бібліотек для розробки веб-додатків. Література зосереджується на показанні технічних особливостей React, таких як компоненти, властивості та стан, а також на використанні React разом з іншими технологіями, такими як Redux, GraphQL та Node.js.

Більшість досліджень зосереджена на розробці веб-додатків, а не на створенні веб-сервісів для організації тестувань та тестувань студентів. Однак, деякі з досліджень включають інформацію про те, як створювати форми та обробляти події, що може бути корисним для розробки веб-сервісу для організації тестувань та тестувань студентів.

Окрім цього, було виявлено, що багато дослідників активно використовують React для розробки веб-додатків з функціоналом, подібним до того, що буде реалізовано в дипломній роботі. Це свідчить про те, що React може бути ефективним інструментом для розробки веб-сервісу для організації тестувань та тестувань студентів.

Отже, аналіз наукових публікацій та літератури з тематики розробки веб-сервісів на React підтвердив високу популярність цієї технології та її



ефективність для створення веб-додатків з різним функціоналом. Деякі дослідження містять інформацію про створення форм та обробку подій, що може бути корисною для розробки веб-сервісу для організації тестувань та тестувань студентів.

Аналіз показав, що React поєднується з іншими технологіями, такими як Redux, GraphQL та Node.js, що робить його універсальним інструментом для розробки веб-додатків. Загалом, література зосереджується на технічних аспектах React, таких як компоненти, властивості та стан, та їх використанні в різних сценаріях.

Зважаючи на високу популярність React та його ефективність для розробки веб-додатків, можна стверджувати, що використання цієї технології для розробки веб-сервісу для організації тестувань та тестувань студентів є обґрунтованим. Однак, детальніші дослідження можуть бути корисними для визначення найкращих практик розробки веб-сервісів на React та їх використання в конкретному сценарії.

## **2.6 Аналіз сучасних рішень для організації тестувань та тестувань, використовуючи React**

React є однією з найпопулярніших бібліотек для розробки веб-додатків, і вона активно використовується для створення веб-сервісів для організації тестувань та тестувань. Нижче наведено деякі з сучасних рішень, які використовують React для розробки веб-сервісів для організації тестувань та тестувань.

*SurveyJS*: SurveyJS - це JavaScript-бібліотека, яка дозволяє створювати опитування та тестування на веб-сайті. Вона підтримує React та React Native, що дозволяє створювати кросплатформові опитування та тестування для веб та мобільних платформ.

*SurveyMonkey*: SurveyMonkey є онлайн-сервісом, який дозволяє створювати опитування та тестування. Вони пропонують бібліотеку React для створення власних тестувань та тестувань на своєму веб-сайті.

*Typeform*: Typeform - це онлайн-сервіс, який дозволяє створювати інтерактивні форми, опитування та тестування. Вони підтримують React та React Native для створення власних форм, тестувань та тестувань.

*Google Forms*: Google Forms - це безкоштовний онлайн-сервіс, який дозволяє створювати опитування та тестування. Вони не пропонують React для створення власних тестувань та тестувань, але можуть бути інтегровані з React-додатками за допомогою Google Forms API.

*Quizlet*: Quizlet - це онлайн-сервіс, який дозволяє створювати навчальні тести та інші навчальні матеріали. Вони пропонують SDK для React Native, що дозволяє створювати мобільні додатки для навчання, включаючи тести та опитування.

Додатково, є також багато інших бібліотек та фреймворків для React, які дозволяють розробляти власні опитування та тестування. Наприклад, Redux Form - це бібліотека, яка дозволяє легко створювати форми в React з використанням Redux. Це може бути корисно для створення тестувань та тестувань з більш складною логікою.

Крім того, є також багато платних та безкоштовних онлайн-сервісів, які дозволяють створювати опитування та тестування з готовими шаблонами та інструментами аналізу результатів, такі як SurveyGizmo, Qualtrics, Formstack та багато інших.

Усі ці рішення мають свої переваги та недоліки, і кращий вибір залежить від конкретних потреб користувача. Наприклад, якщо користувач уже працює з React та хоче створити власний сервіс для тестувань та тестувань, SurveyJS може бути хорошим варіантом. Але якщо користувач шукає безкоштовний сервіс з широким набором інструментів для аналізу результатів, Google Forms може бути більш підходящим варіантом.

## **2.7 Аналіз відповідності функціоналу веб-сервісу технічному завданню та вимогам**

Аналіз відповідності функціоналу веб-сервісу технічному завданню та вимогам - це процес перевірки того, наскільки веб-сервіс відповідає заданому технічному завданню та вимогам, які були встановлені на етапі планування розробки.

При аналізі відповідності функціоналу технічному завданню та вимогам, необхідно оцінювати наступні аспекти:

**Функціональність:** Веб-сервіс повинен відповідати функціональним вимогам, які були встановлені в технічному завданні. Наприклад, якщо у технічному завданні було вказано, що користувач повинен мати можливість створювати, зберігати та переглядати результати тестувань, то веб-сервіс повинен мати таку функціональність.

**Надійність:** Веб-сервіс повинен бути надійним та стійким до можливих помилок та витоків даних. Також, повинна бути забезпечена можливість резервного копіювання даних та відновлення роботи веб-сервісу в разі його відмови.

**Безпека:** Веб-сервіс повинен бути захищеним від можливих атак та витоків конфіденційної інформації. Необхідно забезпечити захист від SQL-ін'єкцій, XSS-атак та інших можливих видів атак.

**Швидкодія та масштабованість:** Веб-сервіс повинен бути швидким та ефективним у використанні, а також повинен бути готовим до масштабування зі зростанням навантаження.

**Взаємодія з користувачами:** Веб-сервіс повинен мати зручний та інтуїтивний інтерфейс, щоб користувачі могли легко взаємодіяти з ним та використовувати його функціонал. Крім того, веб-сервіс повинен забезпечувати користувачам необхідну документацію та інструкції з використання.

**Сумісність:** Веб-сервіс повинен бути сумісним з різними браузерами та операційними системами, які використовуються користувачами.

Загальна оцінка відповідності функціоналу веб-сервісу технічному завданню та вимогам дає змогу зробити висновок про те, наскільки веб-сервіс відповідає вимогам та які зміни можуть бути необхідні для покращення його функціоналу. При необхідності, можна внести зміни в технічне завдання або вимоги, щоб вони були більш адекватні та відповідали реальним потребам користувачів.

## **2.8 Тестування функціоналу та ефективності розробленого веб-сервісу**

Тестування функціоналу та ефективності розробленого веб-сервісу є важливим етапом в розробці будь-якого програмного забезпечення. Цей етап дозволяє перевірити відповідність функціоналу веб-сервісу вимогам, встановленим на етапі планування, а також переконатися, що веб-сервіс працює ефективно та безперебійно.

*Тестування функціоналу* веб-сервісу передбачає проведення різноманітних тестів, які дозволяють перевірити роботу окремих функцій та їх взаємодію між собою. Наприклад, тести на функціональність дозволяють перевірити, чи працюють всі функції веб-сервісу правильно та чи відповідають вони вимогам. Також, проводяться тести на безпеку, які перевіряють стійкість веб-сервісу до атак, та тести на продуктивність, які дозволяють виміряти час відповіді веб-сервісу на запити користувачів та його масштабованість зі зростанням навантаження.

*Тестування ефективності* веб-сервісу передбачає вимірювання часу відповіді веб-сервісу на запити користувачів та перевірку стійкості його роботи при збільшенні навантаження. Цей етап дозволяє виявити можливі проблеми та будувати оптимальні стратегії масштабування веб-сервісу в майбутньому.

В результаті проведення тестування функціоналу та ефективності веб-сервісу можуть бути виявлені можливі проблеми та недоліки, які потрібно буде виправити. Також, результати тестування допоможуть підвищити якість веб-сервісу та забезпечити його безперебійну роботу при збільшенні навантаження.

Для проведення тестування функціоналу та ефективності веб-сервісу можна використовувати різні інструменти та технології, такі як автоматизовані тести, моніторинг веб-сервісу, тестування на реальних користувачах тощо.

У кінцевому результаті, тестування функціоналу та ефективності розробленого веб-сервісу є важливим етапом, який допомагає забезпечити якість та безперебійну роботу веб-сервісу, збільшити його ефективність та задоволення користувачів. Тому, проведення цього етапу розробки веб-сервісу є необхідним кроком для досягнення успіху в його подальшій роботі.

## **3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ГЕНЕРАЦІЇ ТЕСТОВИХ ЗАВДАНЬ ТА ОЦІНКИ ЗНАТЬ НА ОСНОВІ ФУНКЦІЇ RANDOMIZE**

### **3.1 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 OLTP (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 будете 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 "документно-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, OpenTrivialDB

### **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 даного проекту обрана реляційна СКБД Open TriviaDB.

### **3.2 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:

- Підсвічування синтаксису
- Авто відступ
- Розпізнавання типів файлів
- Бічна панель із файлами вказаного каталогу
- Макрос
- 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.

### **3.3 Обрання мови програмування**

#### **JavaScript**

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.

Специфікація ECMA-262 визначила стандартну версію основної мови 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

### **3.4 Архітектура системи генерації тестових завдань та оцінки знань користувачів**

Архітектура системи генерації тестових завдань та оцінки знань користувачів може включати 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.

### **3.5 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 тестувань 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:

```

1 import React from 'react'
2 import { useGlobalContext } from './context'
3
4 import SetupForm from './SetupForm'
5 import Loading from './Loading'
6 import Modal from './Modal'
7 function App() {
8   const {
9     ...waiting,
10    ...loading,
11    ...questions,
12    ...index,
13    ...correct,
14    ...nextQuestion,
15    ...checkAnswer,
16    ...} = useGlobalContext()
17
18   if (waiting) {
19     return <SetupForm />
20   }
21   if (loading) {
22     return <Loading />
23   }
24   const { question, incorrect_answers, correct_answer } = questions[index]
25   // const answers = [...incorrect_answers, correct_answer]
26   let answers = [...incorrect_answers]
27   const tempIndex = Math.floor(Math.random() * 4)
28
29   if (tempIndex === 3) {
30     answers.push(correct)
31   } else {
32     answers.push(answers[tempIndex])
33   }
34 }

```

Рис. 3.1 App.js центральний компонент

```

Quiz-React-App > src > JS App.js > App
35
36   ..return [
37   ...<main>
38   ...<<Modal />
39   ...<<section className='quiz'>
40   ...<<p className='correct-answers'>
41   ...<<connect_answers : {connect} / {index}>
42   ...<</p>
43   ...<<article className='container'>
44   ...<<h2 dangerouslySetInnerHTML={{ __html: question }} />
45   ...<<div className='btn-container'>
46   ...<<{answers.map((answer, index) => {
47   ...<<return (
48   ...<<button
49   ...<<key={index}
50   ...<<className='answer-btn'
51   ...<<onClick={() => checkAnswer(connect_answer === answer)}
52   ...<<dangerouslySetInnerHTML={{ __html: answer }}
53   ...<</>
54   ...<<)}
55   ...<<)}}
56   ...<</div>
57   ...<</article>
58   ...<<button className='next-question' onClick={nextQuestion}>
59   ...<<next question
60   ...<</button>
61   ...<</section>
62   ...<</main>
63   ..]
64 }
65
66 export default App

```

Рис.3.2 App.js центральний компонент(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)

```

Рис. 3.3 Модуль `context`

```

58 }
59
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 }
88
89 const handleChange = (e) => {
90   const name = e.target.name

```

Рис. 3.4 Модуль context(2)

Цей код є частиною програми для генерації тестів на React і містить компонент `AuthProvider`, 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 `AuthProvider`, 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`.

```

1  import React from 'react'
2  import { useGlobalContext } from './context'
3
4  const Modal = () => {
5    const { isModalOpen, closeModal, correct, questions } = useGlobalContext()
6    return (
7      <div
8        className={` ${
9          isModalOpen ? 'modal-container isOpen' : 'modal-container'
10        } `}
11      >
12        <div className='modal-content'>
13          <h2>congrats!</h2>
14          <p>
15            You answered  $\{((correct / questions.length) * 100).toFixed(0)}\%$  of
16            questions correctly
17          </p>
18          <button className='close-btn' onClick={closeModal}>
19            play again
20          </button>
21        </div>
22      </div>
23    )
24  }
25
26  export default Modal
27

```

Рис. 3.5 модуль `Modal`

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 .

```

1  import React from 'react'
2  import { useGlobalContext } from './context'
3
4  const SetupForm = () => {
5    const { quiz, handleChange, handleSubmit, error } = useGlobalContext()
6    return (
7      <main>
8        <section className='quiz quiz-small'>
9          <form className='setup-form'>
10             <h2>Setup Trivia</h2>
11             <div className='form-control'>
12               <label htmlFor='amount'>number of questions</label>
13               <input
14                 type='number'
15                 name='amount'
16                 id='amount'
17                 className='form-input'
18                 value={quiz.amount}
19                 onChange={handleChange}
20                 min={1}
21                 max={50}
22               />
23             </div>
24             <div className='form-control'>
25               <label htmlFor='category'>category</label>
26               <select
27                 name='category'
28                 id='category'
29                 className='form-input'
30                 value={quiz.category}
31               />
32             </div>

```

Рис. 3.6 Модуль SetupForm(початок)

```

34             <option value='general'>General Knowledge</option>
35             <option value='computers'>Computers</option>
36             <option value='books'>Books</option>
37             <option value='maths'>Mathematics</option>
38             <option value='science'>Science & Nature</option>
39             <option value='music'>Music</option>
40             <option value='film'>film</option>
41             <option value='theatre'>Musicals & Theatres</option>
42             <option value='sports'>Sports</option>
43             <option value='anime'>Anime & Manga</option>
44             <option value='video'>Video Games</option>
45             <option value='history'>History</option>
46             <option value='politics'>Politics</option>
47           </select>
48         </div>
49         <div className='form-control'>
50           <label htmlFor='difficulty'>Difficulty</label>
51           <select
52             name='difficulty'
53             id='difficulty'
54             className='form-input'
55             value={quiz.difficulty}
56             onChange={handleChange}
57           >
58             <option value='easy'>Easy</option>
59             <option value='medium'>Medium</option>
60             <option value='hard'>Hard</option>
61           </select>
62         </div>

```

Рис. 3.7 Модуль SetupForm(основна частина)

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`.

### **3.6 Технологія використання розробленої системи тестування**

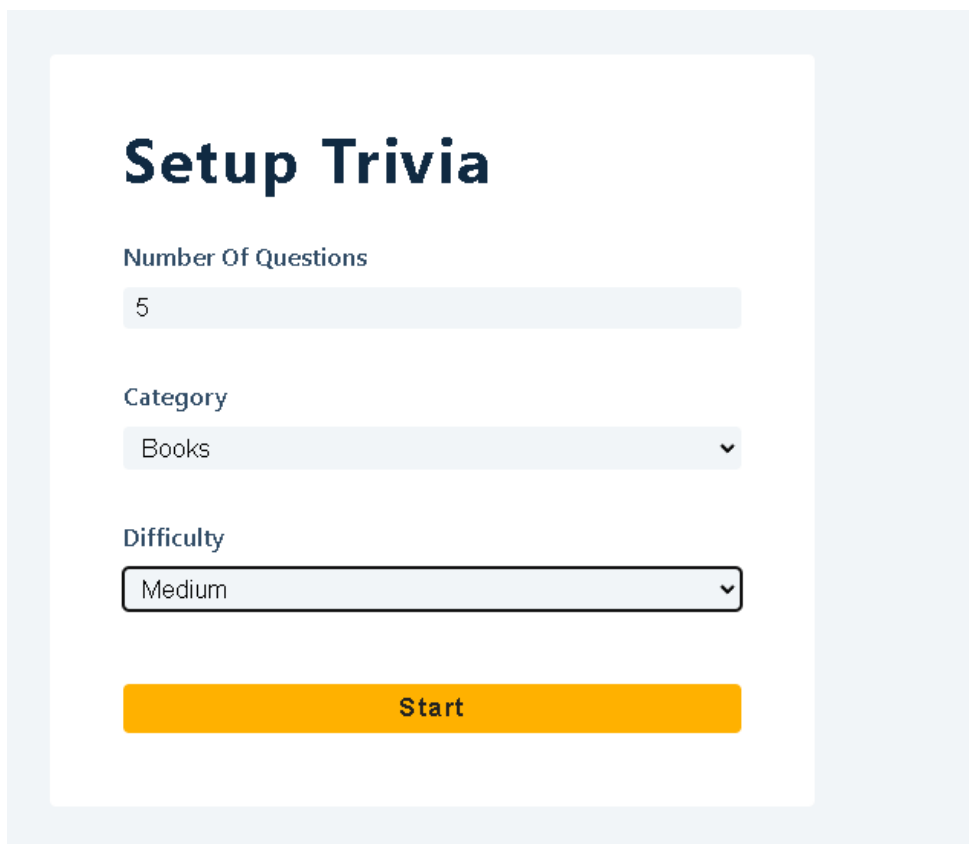
Технологія використання розробленої системи тестування полягає в наступному:

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(рисунок 3.7):



The image shows a web form titled "Setup Trivia". It contains three input fields: "Number Of Questions" with the value "5", "Category" with the value "Books", and "Difficulty" with the value "Medium". A yellow "Start" button is located at the bottom of the form.

Рис. 3.7 Крок перший

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".

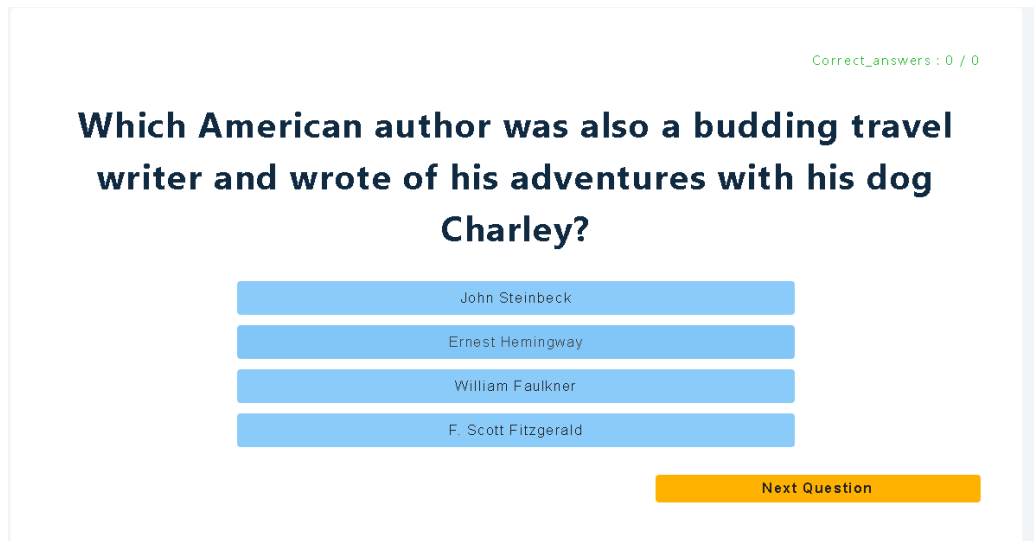


Рис. 3.8 Крок 2

4. Відбувається запит до API Open TriviaDB, де за допомогою вибраних налаштувань генеруються питання для тестування.

На третьому кроку відбувається наступне:

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.

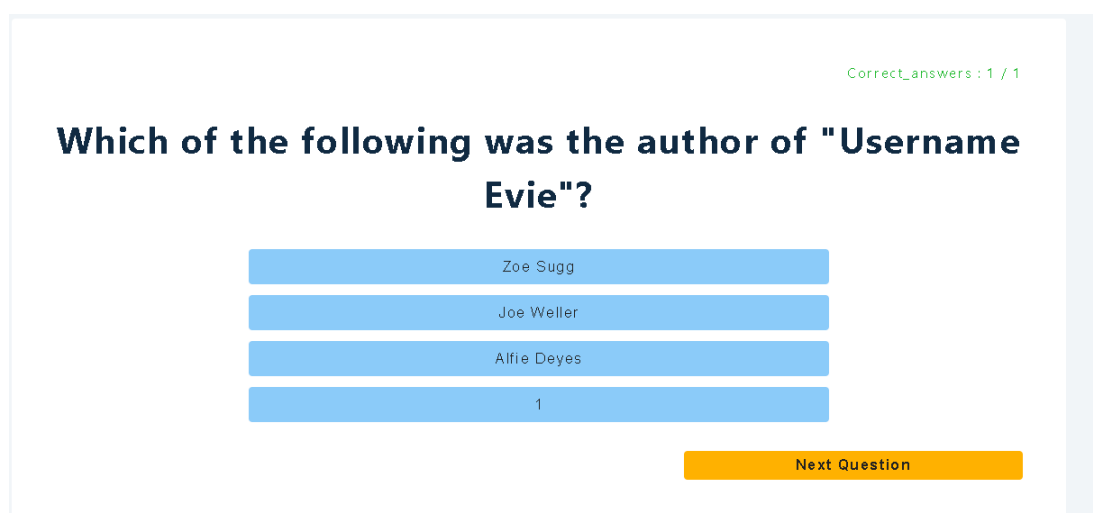


Рис. 3.9 Кроки 3 та 4

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:

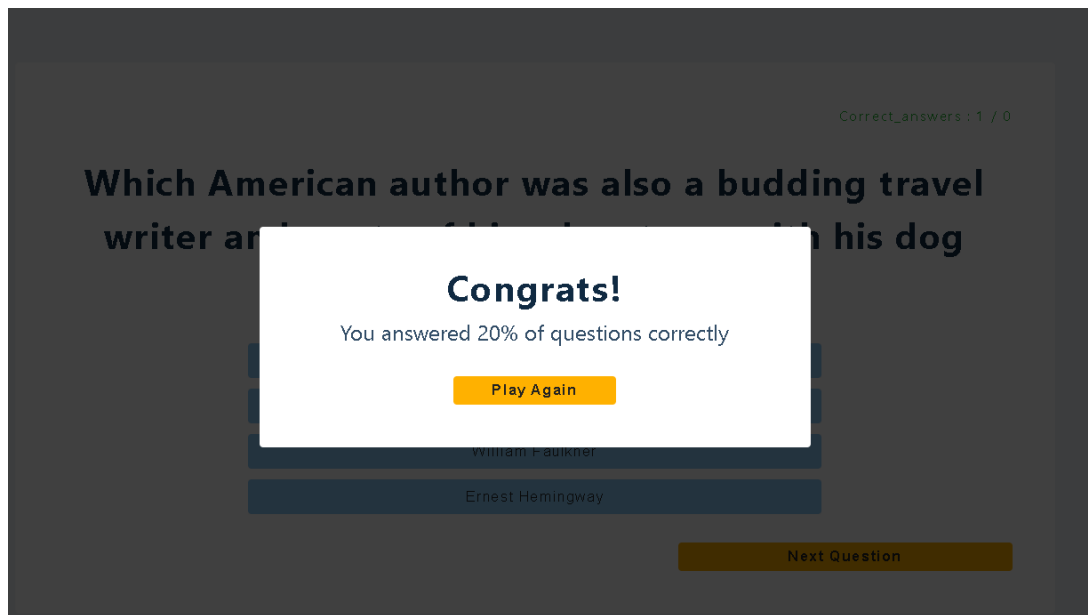


Рис. 3.10 Фінальний крок

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 "Розробка веб-сервісу для організації тестувань студентів з використанням бібліотеки React" була успішно виконана, що підтверджується результатами мануального тестування розробленого веб-сервісу.

Проведено аналіз існуючих програмних засобів оцінювання знань, що допомогло обрати основні інструменти для реалізації веб-сервісу.

Проведено аналіз існуючих аналогів, що допомогло встановити ключові функціональні можливості та визначити необхідні дії для покращення веб-сервісу, такі як налаштування параметрів генерації тестів.

Визначено основні вимоги до веб-сервісу. Головна перевага полягає у реалізації можливості автоматизованої генерації тестових завдань.

Спроектовано та розроблено веб-сервіс для тестувань студентів з метою покращення освітнього процесу.

Для зберігання даних було використано базу даних OpenTriviaDB, що дозволяє забезпечити безпеку, легке зберігання та оновлення даних. Також це дозволило автоматизувати процес створення тестових завдань.

Розроблено можливість генерації випадкових запитань та варіантів відповідей до них завдяки функції Randomize, що робить процес тестування більш об'єктивним та надійним.

Налаштовано можливість вибору кількості, теми та складності запитань у генерованому тесті.

У результаті роботи були досягнуті поставлені цілі, а саме: розроблено веб-сервіс для організації тестувань студентів з використанням бібліотеки React, що забезпечує зручний та швидкий доступ до тестувань.

Отже, розроблений веб-сервіс може бути використаний у навчальних закладах, компаніях для проведення тестувань та досліджень, а також для особистого використання. В цілому, дипломна робота дозволила вивчити

сучасні технології веб-розробки та їх застосування для розробки корисних та інноваційних веб-додатків. Виконана робота може бути використана як основа для подальшого розроблення та удосконалення веб-сервісу, а також відповідає сучасним вимогам до якості та швидкодії веб-додатків. Дипломна робота є важливим кроком у підвищенні рівня знань та навичок веб-розробки студента, а також внеском у розвиток технологій та їх застосування в різних сферах діяльності.

Робота пройшла апробацію:

Висоцький В.О., Замрій І.В. Вдосконалення функціоналу та інтерфейсу веб-сервісу для організації тестувань студентів. Всеукраїнська науково-технічна конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024, С. 200-201.

Висоцький В.О., Замрій І.В. Визначення вимог до веб-сервісу для організації тестувань студентів, з використанням бібліотеки React. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024, С. 465-466.



## ПЕРЕЛІК ПОСИЛАНЬ

1. Kirupa Chinnathambi Learning React: A Hands-On Guide to Building Web Applications Using React and Redux. 2nd ed. Boston, 2018. 304 p.
2. Michele Bertoli React Design Patterns and Best Practices: Build easy to scale modular applications using the most powerful components and design patterns. Birmingham, 2017. 318 p.
3. Vipul Amler, Prathamesh Sonpatki ReactJS by Example - Building Modern Web Applications with React. Birmingham, 2018. 360 p.
4. Mark Tielens Thomas React in Action. New York, 2017. 318 p.
5. React documentation URL: <https://reactjs.org/docs/getting-started.html> (date of access: 19.02.2024).
6. Firebase documentation URL: <https://firebase.google.com/docs> (date of access: 19.02.2020).
7. W3Schools - React Tutorial URL: <https://www.w3schools.com/react/default.asp> (date of access: 18.02.2024).
8. How to deploy Next.js to Firebase URL: <https://dev.to/glennviroux/how-to-deploy-nextjs-to-firebase-18bb> (date of access: 23.02.2024).
9. React Basics: React Forms Examples URL: <https://www.telerik.com/blogs/react-basics-react-forms-examples> (date of access: 01.03.2024).
10. SurveyJS Form Library for React URL: <https://www.npmjs.com/package/survey-react> (date of access: 01.03.2024).
11. Fullstack React GraphQL TypeScript Tutorial URL: <https://www.youtube.com/watch?v=I6ypD7qv3Z8> (date of access: 05.03.2024).

12. How to Build a Quiz App Using React – with Tips and Starter Code URL: <https://www.freecodecamp.org/news/how-to-build-a-quiz-app-using-react/> (date of access: 06.03.2024).
13. Best React UI component libraries and frameworks for your React app in 2021 URL: <https://blog.logrocket.com/best-react-ui-component-libraries-and-frameworks-for-your-react-app-in-2021/> (date of access: 10.03.2024).
14. React Native Elements URL: <https://reactnativeelements.com/> (date of access: 10.03.2024).
15. React Survey - Survey library for React.js URL: <https://surveyjs.io/Documentation/Library/> (date of access: 10.03.2024).
16. React Bootstrap URL: <https://react-bootstrap.github.io/> (date of access: 11.03.2024).
17. How to Create a Survey Application Using Firebase, React, and Formik URL: <https://medium.com/swlh/how-to-create-a-survey-application-using-firebase-react-and-formik-f3114bf1c942> (date of access: 17.03.2024).
18. React Developer Tools URL: <https://reactjs.org/blog/2019/08/15/new-react-devtools.html> (date of access: 20.03.2024).
19. How to build a simple survey app using React URL: <https://www.sitepoint.com/how-to-build-a-simple-survey-app-using-react/> (date of access: 25.03.2024).

## ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

---



ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ  
ТЕХНОЛОГІЙ  
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



### «Розробка веб-сервісу для організації тестування студентів, з використанням бібліотеки React»

Виконав студент 4 курсу  
групи ПД– 42  
Висоцький Вадим Олегович  
Керівник роботи  
Д.т.н., доц., завідувач кафедри ПІЗ Замрій І.В.

Київ – 2024

---

### МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – підтримка процесу тестування студентів за допомогою веб-сервісу розробленого з використанням бібліотеки React.
- **Об’єкт дослідження** – організація тестування студентів.
- **Предмет дослідження** – веб-сервіс для організації тестування студентів.

## ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз галузі тестування студентів.
2. Провести аналіз існуючих програмних засобів оцінювання знань.
3. Провести аналіз готових рішень в галузі тестування студентів.
4. Розробка та формування вимог до веб-сервісу.
5. Спроекувати веб-сервіс для тестування студентів.
6. Реалізувати веб-сервіс для тестування студентів.
7. Реалізувати основні та додаткові функції до веб-сервісу.
8. Провести тестування сервісу.

3

### АНАЛІЗ АНАЛОГІВ

Функціональні можливості	ClassMarker	LearningApps	Quiz React App
Вибір кількості запитань	+	-	+
Можливість обрати категорію тестування	-	+	+
Наявність готової бази тестів	-	+	+
Підрахунок результатів тестування	+	-	+
Вибір складності тесту	-	-	+
Можливість автоматизованої генерації тестів	-	+	+

6

## ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### Функціональні вимоги:

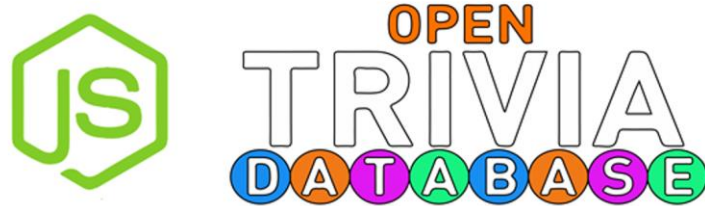
1. Вибір кількості запитань в тесті.
2. Вибір категорії для створення тестів.
3. Вибір складності завдань.
4. Підрахунок результатів тестування.
5. Автоматизована генерація тесту.
6. Можливість пройти тестування.

### Нефункціональні вимоги:

1. Підключення бази даних для автоматизованої генерації тестів.
2. Генерація питань із використанням функції Randomize для створення випадкових чисел.
3. Генерація варіантів відповідей до кожного питання з допомогою функції Randomize.
4. Швидкодія та стабільність роботи сервісу.

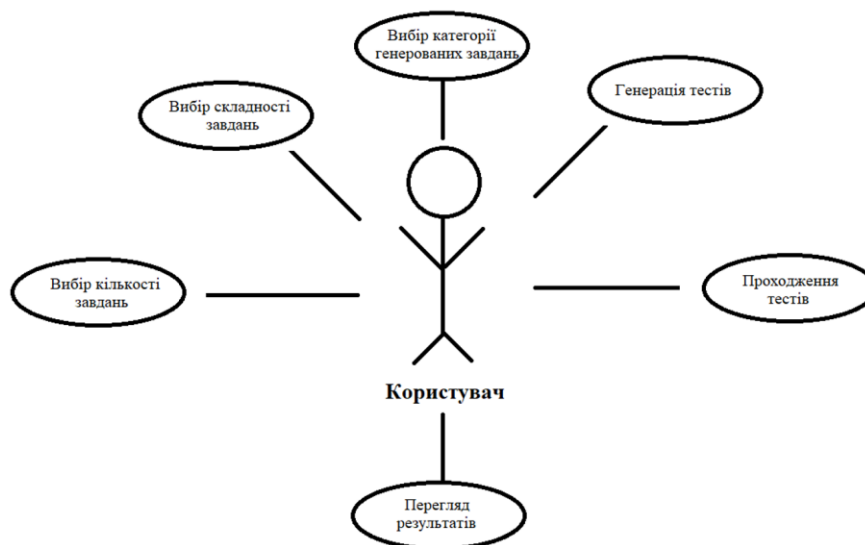
5

## ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



6

## Діаграма варіантів використання



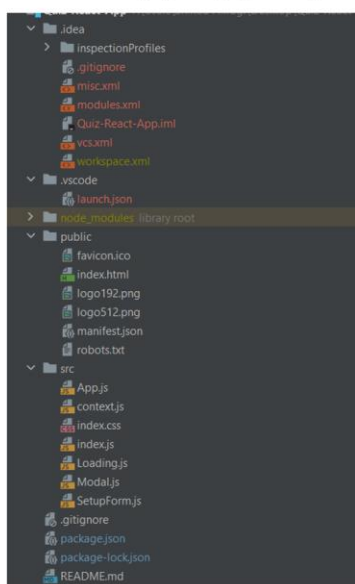
7

## Блок-схема алгоритму роботи



8

## Файлова структура проекту



9

## Екранні форми

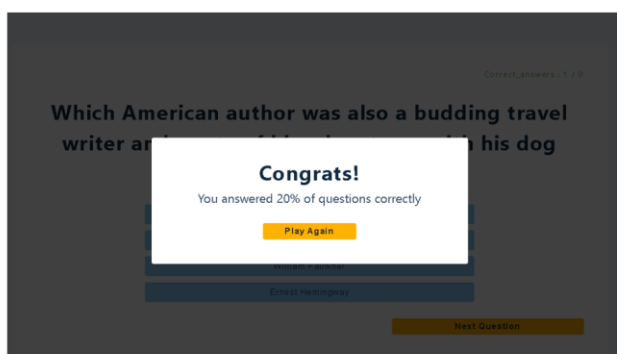
Налаштування тестування

Екран проходження тесту



10

## Екранні форми

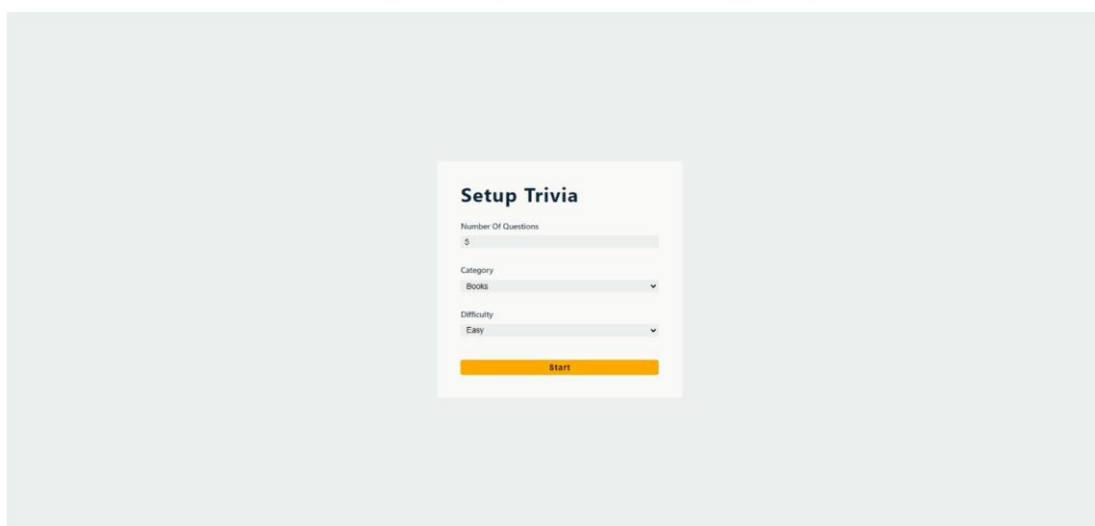


Виведення результатів



11

## Демонстрація роботи веб-сервісу



12



## АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Висоцький В.О., Замрій І.В. Вдосконалення функціоналу та інтерфейсу веб-сервісу для організації тестувань студентів. Всеукраїнська науково-технічна конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті» Збірник тез.15.05.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024, С. 200-201.
2. Висоцький В.О., Замрій І.В. Визначення вимог до веб-сервісу для організації тестувань студентів, з використанням бібліотеки React. Всеукраїнська науково-технічна конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті» Збірник тез.15.05.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024, С. 465-466.



13

## АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Висоцький В.О., Замрій І.В. Вдосконалення функціоналу та інтерфейсу веб-сервісу для організації тестувань студентів. Всеукраїнська науково-технічна конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті» Збірник тез.15.05.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024. Подано до друку.
2. Висоцький В.О., Замрій І.В. Визначення вимог до веб-сервісу для організації тестувань студентів, з використанням бібліотеки React. Всеукраїнська науково-технічна конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті» Збірник тез.15.05.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024. Подано до друку.



13

## Висновки

1. Проведено аналіз предметної галузі.
2. Проведено аналіз існуючих програмних засобів оцінювання знань, що допомогло обрати основні інструменти для реалізації веб-сервісу.
3. Проведено аналіз існуючих аналогів, що допомогло встановити ключові функціональні можливості та визначити необхідні дії для покращення веб-сервісу, такі як налаштування параметрів генерації тестів.
4. Визначено основні вимоги до веб-сервісу. Головна перевага полягає у реалізації можливості автоматизованої генерації тестових завдань.
5. Спроектовано та розроблено веб-сервіс для тестувань студентів з метою покращення освітнього процесу.
6. Використано базу даних OpenTriviaDB для автоматизації генерації тестових завдань.
7. Розроблено можливість генерації випадкових запитань та варіантів відповідей до них завдяки функції Randomize, що робить процес тестування більш об'єктивним та надійним.
8. Налаштовано можливість вибору кількості, теми та складності запитань у генерованому тесті.
9. Проведено мануальне тестування сервісу для перевірки його працездатності.



ДЯКУЮ ЗА УВАГУ!



## ДОДАТОК Б. ЛІСТИНГ ОСНОВНИХ МОДУЛІВ

```

Quiz-React-App > src > JS App.js
1  import React from 'react'
2  import { useGlobalContext } from './context'
3
4  import SetupForm from './SetupForm'
5  import Loading from './Loading'
6  import Modal from './Modal'
7  function App() {
8    ...const {
9    ...waiting,
10   ...loading,
11   ...questions,
12   ...index,
13   ...correct,
14   ...nextQuestion,
15   ...checkAnswer,
16   ...} = useGlobalContext()
17
18   ...if (waiting) {
19   ...return <SetupForm />
20   ...}
21   ...if (loading) {
22   ...return <Loading />
23   ...}
24   ...const { question, incorrect_answers, correct_answer } = questions[index]
25   ...// const answers = [...incorrect_answers, correct_answer]
26   ...let answers = [...incorrect_answers]
27   ...const tempIndex = Math.floor(Math.random() * 4)
28
29   ...if (tempIndex === 3) {
30   ...answers.push(correct)
31   ...} else {
32   ...answers.push(answers[tempIndex])
33   ...}
34   ...return (
35     ...<main>
36       ...<Modal />
37       ...<section className='quiz'>
38         ...<p className='correct-answers'>
39           ...correct_answers : {correct} / {index}
40         ...</p>
41         ...<article className='container'>
42           ...<h2 dangerouslySetInnerHTML={{ __html: question }} />
43           ...<div className='btn-container'>
44             ...{answers.map((answer, index) => {
45               ...return (
46                 ...<button
47                   ...key={index}
48                   ...className='answer-btn'
49                   ...onClick={() => checkAnswer(correct_answer === answer)}
50                   ...dangerouslySetInnerHTML={{ __html: answer }}
51                 ...</button>
52               ...)}
53             ...</div>
54           ...</article>
55           ...<button className='next-question' onClick={nextQuestion}>
56             ...next question
57           ...</button>
58         ...</section>
59       ...</main>
60     ...</>
61   ...</>
62   ...</>
63   ...</>

```

```

32   ...answers.push(answers[tempIndex])
33   ...answers[tempIndex] = correct_answer
34   ...}
35
36   ...return (
37     ...<main>
38       ...<Modal />
39       ...<section className='quiz'>
40         ...<p className='correct-answers'>
41           ...correct_answers : {correct} / {index}
42         ...</p>
43         ...<article className='container'>
44           ...<h2 dangerouslySetInnerHTML={{ __html: question }} />
45           ...<div className='btn-container'>
46             ...{answers.map((answer, index) => {
47               ...return (
48                 ...<button
49                   ...key={index}
50                   ...className='answer-btn'
51                   ...onClick={() => checkAnswer(correct_answer === answer)}
52                   ...dangerouslySetInnerHTML={{ __html: answer }}
53                 ...</button>
54               ...)}
55             ...</div>
56           ...</article>
57           ...<button className='next-question' onClick={nextQuestion}>
58             ...next question
59           ...</button>
60         ...</section>
61       ...</main>
62     ...</>
63   ...</>

```

```

Quiz-React-App > src > JS context.js > table
1  import axios from 'axios'
2  import React, { useState, useContext, useEffect } from 'react'
3
4  const table = {
5    ··general: 9,
6    ··books: 10,
7    ··film: 11,
8    ··music: 12,
9    ··theatre: 13,
10   ··video: 15,
11   ··science: 17,
12   ··computers: 18,
13   ··maths: 19,
14   ··sports: 21,
15   ··history: 23,
16   ··politics: 24,
17   ··anime: 31,
18 }
19
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 }
34
35 Git Graph  GIST [Kapood]  Reviewing Ln 18, Col 2 Spaces: 2 UTF-8 LF () JavaScript Go Live Prettier

```

```

Quiz-React-App > src > JS context.js > table
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)
51       } else {
52         ··setWaiting(true)
53         ··setError(true)
54       }
55     } else {
56       ··setWaiting(true)
57     }
58   }
59
60   ··const nextQuestion = () => {
61
62   }
63 }
64
65 Git Graph  GIST [Kapood]  Reviewing Ln 18, Col 2 Spaces: 2 UTF-8 LF () JavaScript Go Live Prettier

```

```

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  ✓  ...  }
88
89  ✓  ...const handleChange = (e) => {
90  ✓  ...  ...const name = e.target.name
91  ✓  ...  ...const value = e.target.value
92  ✓  ...  ...setQuiz({ ...quiz, [name]: value })
93  ✓  ...  }
94
95  ✓  ...const handleSubmit = (e) => {
96  ✓  ...  ...e.preventDefault()
97  ✓  ...  ...const { amount, category, difficulty } = quiz
98
99  ✓  ...  ...const url = `${API_ENDPOINT}amount=${amount}&category=${table[category]}&difficulty=${difficulty}&type=multiple`
100 ✓  ...  ...fetchQuestions(url)
101 ✓  ...  }
102
103 ✓  ...return (

```

```

Quiz-React-App > src > JS context.js > table
102
103  ...return (
104  ...  <AppContext.Provider
105  ...    ...value={{
106  ...      ...waiting,
107  ...      ...loading,
108  ...      ...questions,
109  ...      ...index,
110  ...      ...correct,
111  ...      ...error,
112  ...      ...isModalOpen,
113  ...      ...nextQuestion,
114  ...      ...checkAnswer,
115  ...      ...closeModal,
116  ...      ...quiz,
117  ...      ...handleChange,
118  ...      ...handleSubmit,
119  ...    ...}}
120  ...  >
121  ...  {children}
122  ...  </AppContext.Provider>
123  ...  )
124  ...  }
125  // make sure use
126  export const useGlobalContext = () => {
127  ...  ...return useContext(AppContext)
128  ...  }
129
130  export { AppContext, AppProvider }
131

```

```

Quiz-React-App > src > JS index.js
1  import React from 'react'
2  import ReactDOM from 'react-dom'
3  import './index.css'
4  import App from './App'
5  import { AppProvider } from './context'
6
7  ReactDOM.render(
8    <<React.StrictMode>
9      <<AppProvider>
10         <<App />
11       </AppProvider>
12     </React.StrictMode>,
13     document.getElementById('root')
14   )
15

```

```

Quiz-React-App > src > JS Loading.js > ...
Aditya Raj, 16 months ago | 1 author (Aditya Raj)
1  import React from 'react'
2
3  const Loading = () => {
4    return (
5      <<main>
6        <<div className='loading'></div>
7      </main>
8    )
9  }
10
11  export default Loading
12

```

```

1  import React from 'react'
2  import { useGlobalContext } from './context'
3
4  const Modal = () => {
5    const { isModalOpen, closeModal, correct, questions } = useGlobalContext()
6    return (
7      <<div
8        className={` ${
9          isModalOpen ? 'modal-container isOpen' : 'modal-container'
10        } `>
11        <<div className='modal-content'>
12          <<h2>congrats!</h2>
13          <<p>
14            You answered  $\{((correct / questions.length) * 100).toFixed(0)}\%$  of
15            questions correctly
16          </p>
17          <<button className='close-btn' onClick={closeModal}>
18            play again
19          </button>
20        </div>
21      </div>
22    )
23  }
24  export default Modal
25
26
27

```

```

1  import React from 'react'
2  import { useGlobalContext } from './context'
3
4  const SetupForm = () => {
5    const { quiz, handleChange, handleSubmit, error } = useGlobalContext()
6    return (
7      <main>
8        <section className="quiz quiz-small">
9          <form className="setup-form">
10             <h2>Setup Trivia</h2>
11             {/* amount */}
12             <div className="form-control">
13               <label htmlFor="amount">number of questions</label>
14               <input
15                 type="number"
16                 name="amount"
17                 id="amount"
18                 className="form-input"
19                 value={quiz.amount}
20                 onChange={handleChange}
21                 min={1}
22                 max={50}>
23             </div>
24             {/* category */}
25             <div className="form-control">
26               <label htmlFor="category">category</label>
27               <select
28                 name="category"
29                 id="category"
30                 className="form-input"
31                 value={quiz.category}
32                 onChange={handleChange}

```

```

33             </select>
34             </div>
35             <option value="general">General Knowledge</option>
36             <option value="computers">Computers</option>
37             <option value="books">Books</option>
38             <option value="maths">Mathematics</option>
39             <option value="science">Science & Nature</option>
40             <option value="music">Music</option>
41             <option value="film">Film</option>
42             <option value="theatre">Musicals & Theatres</option>
43             <option value="sports">Sports</option>
44             <option value="anime">Anime & Manga</option>
45             <option value="video">Video Games</option>
46             <option value="history">History</option>
47             <option value="politics">Politics</option>
48             </select>
49             </div>
50             {/* difficulty */}
51             <div className="form-control">
52               <label htmlFor="difficulty">Difficulty</label>
53               <select
54                 name="difficulty"
55                 id="difficulty"
56                 className="form-input"
57                 value={quiz.difficulty}
58                 onChange={handleChange}
59               >
60                 <option value="easy">Easy</option>
61                 <option value="medium">Medium</option>
62                 <option value="hard">Hard</option>
63               </select>
64             </div>

```

```
57 .....value={quiz.difficulty}
58 .....onChange={handleChange}
59 .....>
60 .....<option value='easy'>Easy</option>
61 .....<option value='medium'>Medium</option>
62 .....<option value='hard'>Hard</option>
63 .....</select>
64 .....</div>
65 .....{error && (
66 .....<p className='error'>
67 .....  can't generate questions, please try different options
68 .....</p>
69 .....)}
70 .....<button type='submit' onClick={handleSubmit} className='submit-btn'>
71 .....  Start
72 .....</button>
73 .....</form>
74 .....</section>
75 .....</main>
76 ..)
77 }
78
79 export default SetupForm
80
```