ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ФАКУЛЬТЕТ ЕКОНОМІКИ І МЕНЕДЖМЕНТУ

ФОРМА НАВЧАННЯ ДЕННА

КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту

Завідувач кафедри _____О.О. Ємець

(підпис)

«_____»_____2020 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО БАКАЛАВРСЬКОЇ РОБОТИ

на тему

ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АЛГОРИТМІЗАЦІЯ ТРЕНАЖЕРА ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «МАТЕМАТИЧНИЙ АНАЛІЗ» З ТЕМИ «ДИФЕРЕНЦІЮВАННЯ ФУНКЦІЇ З ДВОХ ЗМІННИХ»

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Койло Максим Юрійович _____ «___»____2020р.

(підпис)

Науковий керівникк.ф.-м.н., доц.,Чілікіна Т. В. _____ «___»____2020р.

(підпис)

ПОЛТАВА 2020

Вищий навчальний заклад Укоопспілки

«Полтавський університет економіки і торгівлі»

ЗАТВЕРДЖУЮ

Зав. кафедри ММСІ

_____ О.О. Ємець

(підпис)

«__» _____2019 р.

Завдання та календарний графік

виконання бакалаврської роботи

Студента спеціальності 122 «Комп'ютерні науки»

Койло Максима Юрійовича

1. Тема роботи **«Програмна реалізація та алгоритмізація тренажера дистанційного навчального курсу «Математичний аналіз» з теми «Диференціювання функції з двох змінних»**

затверджена наказом ректора №151-Н від «02» вересня 2019 р.

Термін подання студентом дипломної роботи «20» травня 2020 р.

2.Вихідні дані до дипломної роботи

Публікації по темі роботи, методичні рекомендації, стандарти.

3. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ВСТУП

1 Постановка задачі

1.1 Змістовна постановка

2 Інформаційний огляд

2.1 Необхідність та актуальність теми роботи

2.2 Огляд робіт, де розглянуте аналогічне до теми роботи завдання

2.3 Програми – тренажери в дистанційній освіті

3 Теоретична частина

3.1 Алгоритмізація задачі за темою роботи

3.2 Розробка блок-схеми, яка підлягає програмуванню

4 Практична частина

4.1 Опис процесу програмної реалізації

4.2 Технології реалізації тренажеру

4.3 Опис програми

4.4 Необхідна користувачу програми інструкція

ВИСНОВКИ

4. Перелік графічного матеріалу (з точним визначенням кількості блок-схем, іншого графічного матеріалу) - Блок-схема алгоритму

5. Консультанти розділів дипломної роботи

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
Вступ	Чілікіна Т. В.	30.09.19	
1. Постанова задачі	Чілікіна Т. В.	30.09.19	
2.Інформаційний огляд	Чілікіна Т. В.	30.09.19	
3. Теоретична частина	Чілікіна Т. В.	30.09.19	
4. Практична реалізація	Чілікіна Т. В.	30.09.19	

6. Календарний графік виконання дипломної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ	10.05.20	
2. Вивчення методичних рекомендацій та стандартів та звіт керівнику	01.03.20	
3. Постановка задачі	01.03.20	
4. Інформаційний огляд джерел бібліотек та	15.03.20	

Зміст роботи	Термін виконання	Фактичне виконання
інтернету		
5. Теоретична частина	15.04.20	
6. Практична частина	01.05.20	
7. Закінчення оформлення	15.05.20	
8. Доповідь студента на кафедрі	15.05.20	
9. Доробка (за необхідністю), рецензування	15.06.20	

Дата видачі завдання «30» вересня 2019 р.
Студент (ка) _____
(підпис)
Науковий керівник _____ _к.ф.-м.н. Чілікіна Т. В.____
(підпис) (науковий ступінь, вчене звання, П.І.Б.)

Результати захисту дипломної роботи
Дипломна робота оцінена на _____
(балів, оцінка за національною шкалою, оцінка за ECTS)
Протокол засідання ЕК № ____ від «____» _____ 202_ р.
Секретар ЕК_____ _____
(підпис) (ініціали та прізвище)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
ДК	Дистанційний курс
CMS	Course Management System – система управління курсами
SPA	Single page application – одно сторінкові додатки
SL	Supervised learning – навчання з учителем
JSON	Java script object notation - формат збереження даних

РЕФЕРАТ

Записка: 76 сторінок, 13 рисунків, 1 додатка (на 25 сторінках), 11 літературних джерел.

Предмет розробки – програмний продукт, який реалізує елементи тренажеру з теми «Диференціювання функції двох змінних» на мові програмування JavaScript з використанням бібліотеки React.

Мета роботи – програмні засоби мови JavaScript з використанням бібліотеки React та методи диференціювання функції двох змінних.

Методами розробки є використання редактора Visual Studio Code на мові програмування JavaScript з використанням бібліотеки React та Mobx.

Ключові слова: ТРЕНАЖЕР, WEB-ДОДАТОК, JS, REACT, MOBX

ВСТУП

У сфері освіти зростає роль та використання інформаційних технологій. Головним трендом у сфері освіти стають відкриті та закриті онлайн-курси і освіта основана на медіа технологіях. Курси що проводяться в онлайн режимі та тести стали сьогодні дуже популярним засобом для навчання, така форма навчання дає змогу інтерактивного спілкування між студентами та викладачами, а також онлайн прийому іспитів, наприклад це використовується для отримання цифрових сертифікатів в різних компаніях таких як Microsoft. Це одна із найсучасніших форм дистанційного навчання, що активно розвивається у освіті всього світу.

Теоретичні знання отримані студентом під час роботи з дистанційними матеріалами потребують перевірки за допомогою спеціальних тренажерів і тестових засобів. Вони дозволяють не тільки перевірити знання і навички, а також можуть навчати в інтерактивній формі. Наприклад тренажер в якому студент може самостійно вводити дані та перевіряти результат, і якщо помилився отримувати підказки та виправлення. Така інтерактивна форма допомога краще засвоїти знання, на відміну від простого читання інформації по темі.

Основна мета роботи – створити тренажер з теми «Диференціювання функції з двох змінних» дистанційного навчального курсу «Математичний аналіз», що дасть можливість студентам самостійно та дистанційно готуватись до тематичних та поточних контрольних робіт з даної теми.

Задачі дослідження:

— розглянути особливості математичних тренажерів;
— обґрунтувати вибір програмних засобів для розробки тренажеру;
— створити тренажер за даною темою.

Об'єктом дослідження бакалаврської роботи є створення тренажеру з теми використанням технологій та інструментальних засобів створення комп'ютерних та web- додатків.

Предметом розробки є програмний продукт - тренажер з теми «Диференціювання функції двох змінних»

Методи розробки – програмні засоби мови JavaScript з використанням

бібліотеки React та методи диференціювання функції двох змінних.

Бакалаврська робота складається з трьох розділів, а саме: постановка задачі, інформаційний огляд, теоретична частина, практична частина.

Результатом виконання бакалаврської роботи є розробка алгоритму та програмна реалізація тренажера з теми «Диференціювання функції двох змінних» на мові програмування JavaScript.

Обсяг пояснювальної записки: _76_ стор., в т.ч. основна частина _50_ стор., додатки - _26_ стор., джерел - _11_ назв.

1 ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовна постановка

В бакалаврській роботі потрібно розробити тренажер з теми «Диференціювання функції двох змінних», за допомогою якого студенти можуть перевіряти та закріпляти свої знання з даної теми. Тренажер містить як практичні так і теоретичні питання. Для проходження тестування тренажер повинен мати тести трьох типів. При проходженні тесту завдання повинні показуватися студенту по черзі, якщо відповідь правильна відбувається перехід до наступного питання, а якщо ні то випадає довідка користуючись якою можна отримати правильну відповідь. Після відповіді на останнє запитання повинно відобразитися вікно результату, в якому буде показано, чи було пройдено тест чи ні, також кількість правильних відповідей з пройденого тесту. Якщо студент не пройшов тест він може повторити свою спробу знову.

Завдання в тестах мають бути трьох типів складності:

1. теоретичні питання з вибором варіанту відповіді;
2. практичне завдання з вибором варіанту відповіді;
3. практичне завдання з ручним внесенням відповіді.

Проектування тренажеру складається з наступних кроків.

Крок 1. Формування вимог до тренажеру. Даний крок включає в себе розробку та узгодження з викладачем логічної структури тренажеру, а також дизайну за потребою. Розробка алгоритму його роботи.

Крок 2. Вибір середовища та мови програмування. Цей крок є важливим, так як він вирішує багато питань пов’язаних як з можливістю роботи тренажера на різних системах.

Крок 3. Програмна реалізація на обраній мові програмування з урахуванням всіх вимог до тренажеру.

Крок 4. Тестування та вдосконалення розробленого програмного продукту. На цьому кроці потрібно повністю перевірити програму, її складові частини, роботу в середовищах, інтеграцію в систему дистанційного навчання. За потребою можливо вдосконалити тренажер, додавши нові можливості чи компоненти.

Дотримання наведених кроків і вимог до тренажеру, забезпечує процес створення алгоритму і програмної реалізації всіх елементів тренажеру.

2 ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Необхідність та актуальність теми роботи

Розширення меж використовування технології масового тестування в різних сферах людської діяльності є стійкою тенденцією в усьому світі впродовж 20-го століття. Для педагогів України це питання стало актуальним після того, як в країні з’явилися альтернативні заклади різних форм власності, і, як наслідок цього, з’явився державний механізм сертифікації, атестації та акредитації вузів у його нинішній формі, що містить елементи технологій тестування учнів.

Найважливішим компонентом педагогічної системи і складовою частиною навчального процесу є контроль за навчальною діяльністю учнів, призначений для визначення успішності навчання кожного учня, аналізу отриманих результатів та корекції подальшого процесу навчання. Інструментом підвищення якості освіти разом із реформуванням змісту освіти виступає вдосконалення системи оцінювання, модернізація діагностики навчання учнів.

Останнім часом замість традиційного поняття "контроль", окрім вже згадуваного поняття "діагностика", все частіше стали використовувати поняття "моніторинг". При проектуванні навчального процесу перед педагогом постає завдання вибору методів і форм контролю навчальних досягнень учнів, критеріїв якості засвоєння вивченого матеріалу, розробки процедур здійснення контролю, обґрунтування способів індивідуальної корекції навчальної діяльності учнів.

Необхідність забезпечення контролю й оцінювання не тільки результату, а й процесу навчання сприяє пошуку оперативних та об'єктивних методів контролю знань. Система оцінки і контролю повинна відповідати вимогам управління пізнавальною діяльністю учнів і виступати в ролі відповідного інструментарію для її здійснення [10-11].

Суб'єктивність оцінки знань пов'язана певною мірою з недостатньою розробкою методів контролю системи знань. Об'єктивний підхід полягає в тому, що для виявлення наявності знань завжди використовується адекватний

інструмент. Сучасна методика пропонує тест як інструмент вимірювання рівня знань, за допомогою якого можна не тільки виявити якість навчання, але і оптимально управляти навчальним процесом.

Ефективність функціонування системи професійної освіти значною мірою залежить від впровадження в діяльність вузу інновацій. Вони змушують всіх учасників навчально-виховного процесу визначити й проаналізувати рівень своїх знань, умінь, навичок, спрямувати свою діяльність на шлях перспективних перетворень. При цьому основна роль належить викладачеві. Його готовність до впровадження інновацій, вміння організувати цей процес та управляти ним є невід'ємними складовими успішної інноваційної роботи.

Однією з інноваційних форм інформатизованої системи освіти є тестування. Ця форма, надзвичайно популярна за рубежем, в останні десятиліття активно впроваджується і в Україні. Національний університет "Києво-Могилянська Академія" був першим вищим навчальним закладом нашої держави, який під час відбору студентів почав застосовувати систему тестування. Цю систему було визначено однією з кращих в Європі. Дану практику дедалі частіше впроваджують інші вищі навчальні заклади. Тому в підготовці молодшого спеціаліста використання тестування як методу контролю та оцінки знань є актуальним.

Тест (test) – слово англійського походження, що означає іспит, спробу, випробування. Тест – це стандартизоване завдання, за результатами якого роблять висновок про знання, уміння, навички (здібності, професійну придатність, обдарованість тощо) того, кого випробовують. У сучасній теорії та практиці тестового контролю нараховується понад 20 різновидів тестів: залежно від мети, характеру та функцій контролю, характеру, форми відповіді.

За результатами опитування 32% викладачів вищих навчальних закладів постійно використовують тести для контролю, 68% - користуються епізодично, або не користуватися ними зовсім. Близько 80% викладачів, які використовують тестовий контроль, працюють за авторськими тестами, при цьому вони орієнтуються на стандартні тестові методи, що як правило передбачають закриті

тести з двома – трьома варіантами відповідей. Запитання, як правило, мають випадковий характер, відображаючи емпіричне узагальнення і формування вміння діяти за зразком. Тобто, недоліки традиційного розумово-емпіричного навчання розповсюджується і на зміст тестування.

Цілком очевидною є необхідність не тільки більш широкого та активного використання тестових методів у викладанні спеціальних, соціально-економічних та фундаментальних дисциплін, а й розробки методичних матеріалів.

Тест як система завдань специфічної форми і відповідного змісту є науково обґрунтованим інструментом оцінювання знань, умінь і навичок студентів, допомагає здійснювати індивідуальний контроль результатів навчання кожного з них, мобільно керувати навчально-виховним процесом. Порівняно з традиційними формами контролю знань (контрольна робота, іспит, залік, диференційованих залік.) тестування нерідко виявляється більш ефективно. Адже недостатньо оцінити рівень знань, треба спрогнозувати, як студент зможе ці знання використати. Тобто, зростає роль психологічного тестування.

Виникає необхідність поєднати його з тестуванням рівня знань. Тоді результати будуть більш об’єктивними, враховуватимуть не лише рівень та обсяг знань, а й особливості характеру студента, його нахили та можливості освіти, вміння аналізувати, узагальнювати, встановлювати причинно-наслідкові зв’язки. Як свідчить досвід, найчастіше оцінювання і контроль досягнень студента здійснюється за кінцевим результатом. На жаль, майже не піддається оцінці його діяльність, рівень розвитку в динаміці, вплив зовнішніх факторів на момент оцінювання.

Практика показала наступні переваги тестування над іншими формами контролю знань:

— упродовж досить обмеженого часу може бути перевірена якість знань, навичок у зазначеної кількості студентів;

— можливий контроль знань, умінь, навичок на необхідному, заздалегідь запланованому рівні;

— реальним є самоконтроль;

— знання оцінюють більш-менш об’єктивно;

— увага студента фіксується не на формуванні відповіді, а не осмисленні її суті;

— створюють умови для постійного зворотного зв’язку між студентом і викладачем.

— Проте тестовий контроль знань має й істотні недоліки:

— ймовірність випадкового вибору правильної відповіді;

— можливість при застосуванні тестів закритого типу оцінки тільки кінцевий результат (правильно - неправильно), у той час як сам процес, що привів до нього, не розкривається;

— психологічний недолік – стандартизація мислення без врахування рівня розвитку особистості;

— велика затрата часу на складання необхідного "банку" тестів, їх варіантів, трудомісткість процесу;

— тести не сприяють розвитку мови [5].

У пошуках шляхів підвищення ефективності системи перевірки рівня засвоєння студентами було введено систематичне тестування. Тести при вивченні діловодства впроваджувались поступово, що дало змогу психологічно підготувати студентів. Спочатку пропонувались прості тести з вибірковими відповідями і лише через деякий час вводились більш складні конструкції. Доведено, що надзвичайно продуктивним є використання навчальних тестів, підготовлених на основі методу ключових ситуацій.

Інформація в питанні тесту може подаватись у будь-якій формі – тексту, графічного зображення, звукового повідомлення, відео сюжету, формули тощо. Причому, ми ввели в практику відкритість та доступність бази тестів, тобто зміст тестів відомий студентам. Але при цьому тести постійно поновлюються з урахуванням змін у навчальних планах, програмах, реаліях сьогодення. Тестування з діловодства гарантує об'єктивність оцінки знань, умінь, навичок студентів, сприяє усуненню проявів суб'єктивізму, а відтак і формуванню позитивного ставлення до даної дисципліни і викладача.

Дуже важливим при проведені тестового контролю є дотримання організаційного моменту (пояснення мети, порядку виконання та оформлення тесту, визначення часу та його виконання, забезпечення кожного студента бланком відповідей стандартного зразка, що великою мірою заощаджує час і студента, і викладача).

Тестові завдання значно скорочують час очікування студентами оцінки, що є суттєвим фактором – к психологічним, так і виховним.

Після тестування, а воно може охоплювати матеріал окремої теми, розділу або курсу в цілому, обов'язково аналізують його результати. Аналіз необхідний для того, щоб студент зміг перевірити, наскільки адекватно він оцінює свої знання, повірити у власні сили і скорегувати свою підготовку. Викладач не лише фіксує факт помилок і називає правильні відповіді, а й докладно пояснює студентам причини помилкових дій.

Студентам, які допустили помилки, пропонується письмово або усно відповісти на запитання: Чому вибрана відповідь неправильна? Наведіть докази на користь правильного варіанта. Робота над помилковими відповідями, усвідомлення помилки, її причини, знайдення помилкової ланки в ланцюгу розумових дій значно зменшують ймовірність запам'ятовування помилкових знань, "витіснення" правильних відповідей. Таким чином викорінюється практика "вгадування" студентом правильного рішення поставленої в тесті задачі.

Наступним етапом впровадження та використання тестування в процесі вивчення різних дисциплін стала їх електронна форма. Було забезпечено їх функціонування на локальному комп'ютері та в локальній мережі навчальної аудиторії. Така форма, як правило, викликає додатковий інтерес у студентів, а крім того, дозволяє їм самостійно без участі викладача перевірити та оцінити рівень власних знань з конкретної теми чи комплексу тем курсу. У комп'ютерній програмі тестування використовуються анімаційні, звукові, ігрові елементи, а також система реєстрації ходу перевірки та її результатів, зокрема, таких показників, як час розв'язання кожної задачі, кількість помилкових і правильних відповідей, підсумкова оцінка [2-3].

Важливою умовою тестування, як універсального педагогічного інструмента, є частота його проведення, яка залежить від дисципліни, її ролі і місця в навчальному плані, особливостей засвоєння знань. Слід зробити тестування звичною і зручною формою регулярного контролю знань студентів. Необхідно пам'ятати, що тестування – це не самоціль, а ефективна форма повторення – узагальнення і впорядкування вивченого. Контрольно-оцінювальна функція навчання – це лише елемент добре організованого і технологічно продуманого навчально-виховного процесу. Якщо студенти матимуть міцні знання, то їх оцінювання не становитиме особливих труднощів, в якій би формі воно не проводилось.

2.2 Огляд робіт, де розглянуте аналогічне до теми роботи завдання

Однією з важливих складових частин дистанційного навчання є його реалізація за допомогою використання інформаційних технологій, а саме системи управління навчанням, створеної для розробки, управління та поширення навчальних матеріалів онлайн із забезпеченням спільного доступу багатьох користувачів як правило через Internet.

На даний момент створена велика кількість різноманітних систем дистанційного навчання як з відкритим кодом, що є як правило безкоштовні повністю чи частково, так і платних, широковживаних та вузькоспеціалізованих. Ознайомимось зосновними системами дистанційного навчання, які використовуються учбовими закладами по всьому світі.

ATutor — веб-орієнтована система керування навчанням (Learning Management System, LMS). Програмний продукт є простим у встановленні, налаштуванні та підтримці для системних адміністраторів; викладачі (інструктори) можуть досить легко створювати та переносити навчальні матеріали та запускати свої онлайн-курси. А оскільки система є модульна, тобто складається з окремих функціональних одиниць — модулів, то вона відкрита для модернізації і розширення функціональних можливостей.

Програма розробляється та підтримується з 2001 року Ґреґом Ґеєм (Greg Gay), Джоелом Кроненберґом (Joel Kronenberg), Гайді Гейзелтон (Heidi Hazelton) із Дослідницького центру адаптивних технологій Університету Торонто (Adaptive Technology Resource Centre, University of Toronto). Система ATutor поширюється на основі GNU General Public License (GPL), яка, зокрема, дозволяє вільно використовувати, змінювати та доповнювати програму.

На рис. 2.1 зображено головну сторінку яку бачить користувач при вході в систему. Як бачимо тут є меню, перелік категорій контенту, так навігація по основним блокам таким як форум, вікі, презентації, календар тощо.

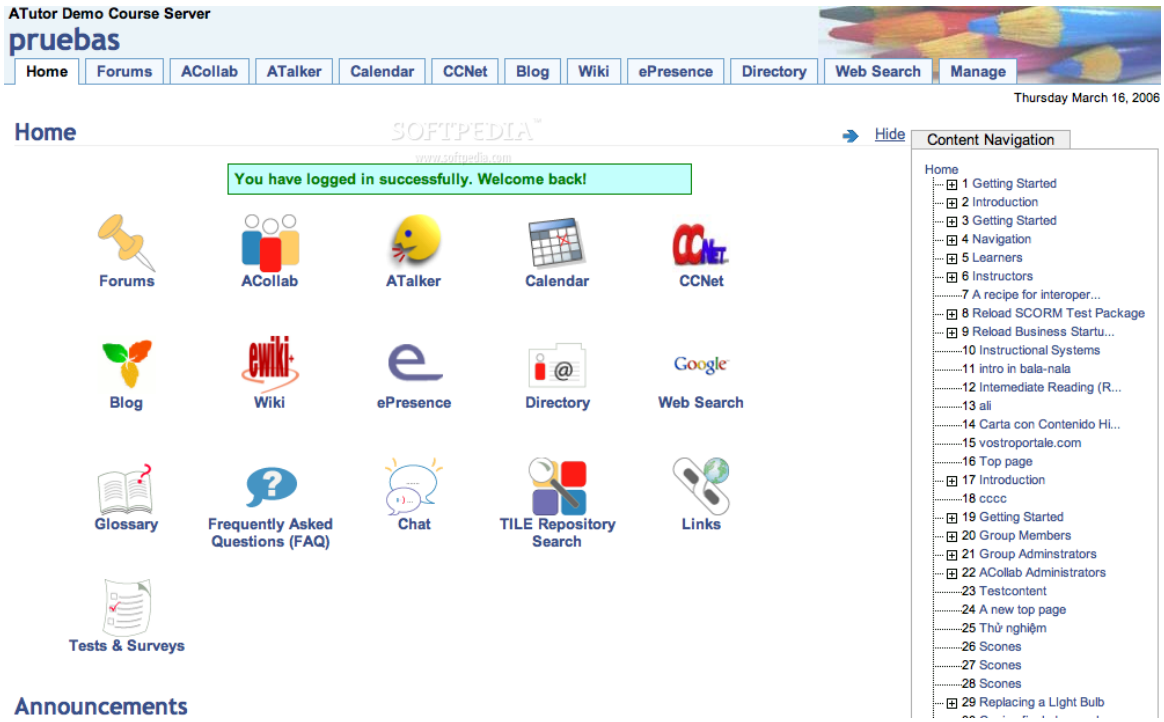


Рис 2.1 – Сторінка користувача ATutor

Claroline - це спільна платформа eLearning та eWorking (система управління навчанням), випущена під ліцензією GPL з відкритим кодом. Вона дає можливістьбагатьом організаціям у всьому світі, починаючи від різноманітних шкіл до університетів, а також від компаній до асоціацій, створюючи та керуючи курсами та місцями співпраці за допомогою мережі Інтернет.

На початку системою Claroline займались працівники навчальних закладів, вона відразу розроблялась з оглядом на педагогічний досвід та потребисаме викладачів. Інтерфейс управління системою інтуїтивно зрозумілий, а менеджмент та конфігурування системи не потребує якихось спеціальних технічних знань та навиків.

Платформа Claroline організована навколо концепції простору, пов'язаного з курсом або педагогічною діяльністю. Кожне частина простору дає перелік інструментів, що дозволяють педагогам:

– створювати опис курсу;
– публікація документів у різному форматі (текст, PDF, HTML, відео та інше);

- керування публічними або приватнимифорумами;
- створення навчальних курсів;
- створення групи користувачів;
- створення вправ;
- менеджментпорядку виконання завдань та термінами;
- повідомлення (також e-mail);
- публікація домашньої роботи, для виконання онлайн;
- перегляд статистики відвідувань та виконання вправ;
- використання вікі дляпублікування спільних документів.

На рис. 2.2 зображено домашню сторінку користувача Claronline. Основними елементами є ліве меню з доступом до всіх частин навчального простору, а також випадаюче меню з додатковими сторінками.

Рис. 2.2 – Домашня сторінка Claronline

Розглянувши ці системи дистанційного навчання ми бачимо, що кожна з них має як свої переваги так і деякі недоліки, тому неможливо чітко вибрати найкращу систему. Кожна система проектувалась з урахуванням покращення навчального процесу і зручності в користуванні як для педагогів так і студентів. Всі системи мають засоби, що дозволяють вести статистику

успішності чи виконання робіт, можливість переписування чи вести дискусії в форматі форуму.

Отже, як бачимо немає чіткої відповіді на питання – яка система найкраща, тому при обранні системи потрібно в першу чергу орієнтуватись на потреби і можливості навчального закладу.

2.3 Програми – тренажери в дистанційній освіті

Великі можливості щодо формування творчої особистості фахівця містять інтерактивні методи навчання. Такі методи реалізують програми тренажери, які дозволяють користувачеві закріпити теоретичні матеріали на практиці, отримати нові знання в інтерактивній формі при взаємодії з програмою.

Тренажер реалізує наступні аспекти:

- оцінка обсягу наданого матеріалу;
- врахування як рівня підготовки так і мотиву навчання;
- встановлення алгоритму виконання певної роботи.

Прийнято, що тренажер складається з декількох дій чи певних кроків. На кожному етапі студент перевіряється на певні знання, при правильних відповідях він отримує можливість рухатись далі і також завершення роботи з програмою. На кожному етапі виконання роботи студент має можливість задати питання викладачеві або звертатись до матеріалів лекції, які також можуть бути інтегровані в тренажер. Об'єм завдань тренажера має бути достатній для забезпечення роботою групу студентів.

3 ТЕОРЕТИЧНА ЧАСТИНА

3.1 Алгоритмізація задачі за темою роботи

Програмна реалізація тренажера, що тестує знання з теми «Диференціювання функції двох змінних» передбачає наявність трьох варіантів тестів, які містять задачі теоретичного і практичного плану з вибором, а також тестових завдань і заповненням відповіді з клавіатури.

Розглянемо кожен типовий тест .

Крок 1. На екрані текст запитання.

Площина, у якій лежать всі дотичні прямі до поверхні в її звичайній точці: називається:

а) дотичною площиною

б) нормальною пдощиною.

Користувач має обрати одну правильну відповідь а), якшо відповідь вибрана вірноі відбувається перехід до наступного кроку, якщо ні випадає повідомлення "Спробуйте ще!".

Крок 2. З’являється наступне питання.

Швидкістю зміни функції є:

а) Границя відношення приросту функції до приросту аргументу за умови, що приріст аргументу прямує до безкінечності;

б) Границя відношення приросту функції до приросту аргументу за умови, що приріст аргументу не прямує до безкінечності;

в) Границя відношення приросту функції до приросту аргументу за умови, що приріст аргументу прямує до нуля;

г) Границя відношення приросту функції до приросту аргументу за умови, що приріст аргументу не прямує до нуля.

Користувач має обрати одну правильну відповідь, після відповіді відбувається перехід до наступного кроку, якшо відповідь вибрана вірноі відбувається перехід до наступного кроку, якщо ні випадає повідомлення "Спробуйте ще!".

Крок 3. З’являється наступне питання.

Виберіть равильну відповідь :

Знайти $Z'_{x,}$ якщо $Z= 3x_2 + 4xy - 7\cos y$

а) $Z'_x = 6x + 4y;$

б) $Z'_x = 6x + 4x;$

в) $Z'_x = 4y + 7\sin y;$

г) $Z'_x = 4x + 7\sin y.$

Користувач має обрати одну правильну відповідь а), після вірної відповіді відбувається перехід до наступного кроку, якщо ні випадає повідомлення "Спробуйте ще!".

Крок 4. З'являється наступне питання.

Виберіть правильну відповідь

Знайти Z'_y , якщо $Z= 3x_2 + 4xy - 7\cos y$

а) $Z'_y = 6x + 4y;$

б) $Z'_y = 4x + 7\sin y;$

в) $Z'_y = 4y + 7\sin y;$

г) $Z'_y = 6x + 4x.$

Користувач має обрати одну правильну відповідь б), після вірної відповіді відбувається перехід до наступного кроку, якщо ні випадає повідомлення "Спробуйте ще!".

Крок 5. З'являється наступне питання.

Потрібно в порожні комірки вписати вірну відповідь

Знайти частичну похідну Z'_x для функції $Z = 2x^5 y^8$

$Z'_x = (\_\_\_)x^{(\_\_)}y^{(\_\_)}$

Користувач вносить в комірки свої відповіді, якщо відповідь вносяться не вірно відбувається підсвічення червоним кольором, після правильної відповіді відбувається перехід до наступного кроку.

Крок 6. З'являється наступне питання.

Потрібно в порожні комірки впишіть вірну відповідь.

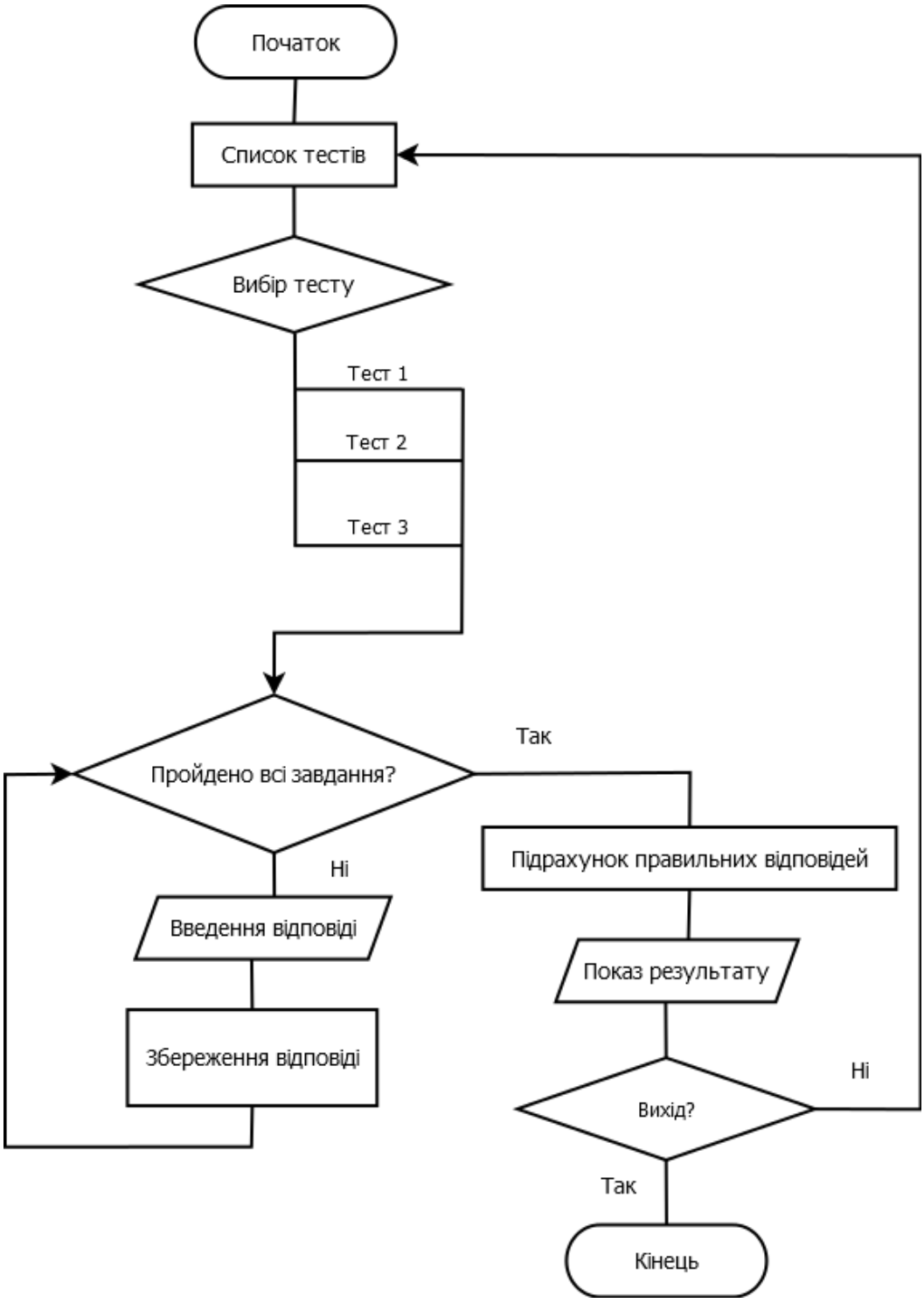
Знайти частичну похідну Z'_y для функції $Z = 4x^2 + 2xy + 3y^2$

Z'_y= (___)$x^{(\_)}$ +(___)x $^{(\_)}y^{(\_)}$ + (___)$y^{(\_)}$

Користувач вносить в комірки свої відповіді, якщо відповідь вносяться не вірно відбувається підсвічення червоним кольором, після отриманні правильної відповіді, відбувається завершення роботи тренажера і , користувач отримує вікно з результатом роботи.

3.2 Розробка блок-схеми, яка підлягає програмуванню

У цьому пункті бакалаврської роботи представлена блок-схема роботи програми, розроблена під час реалізації тренажера. На рис. 3.1 представлена блок-схема алгоритму роботи тренажера.



4 ПРАКТИЧНА ЧАСТИНА

4.1 Опис процесу програмної реалізації

Тренажер було вирішено створювати за допомогою бібліотеки React тому було обрано середовище VS Code. Це найпопулярніший редактор який підтримує встановлення різноманітних плагінів для розробки. Тренажер буде створений за допомогою бібліотеки Ant Designe, яка дозволяє створити зручний, привабливий та сучасний дизайн без особливих зусиль. Нарисунку 4.1. зображено створення нового проекту React в редакторі VS Code.

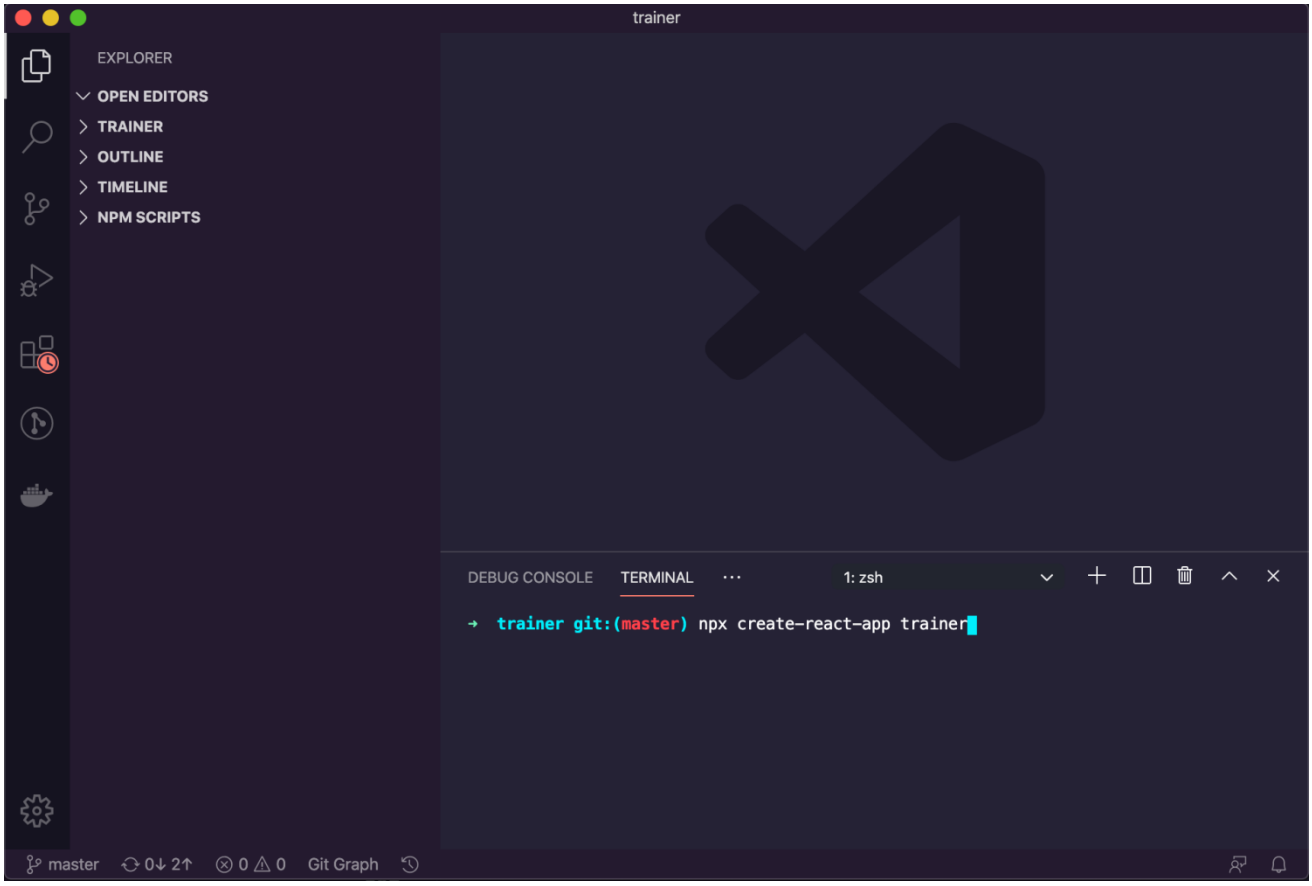


Рисунок. 4.1. Створення React проекту через командну строку

Під час написання програми були реалізовані деякі компоненти, кожні з яких відповідають за певний крок алгоритму. Для відображення карточки доступного тесту та переходу до тесту після стартового вікна був створений такий компонент:

```
<Card
    title={test.title}
    extra={
<div>
```

```
        {test.completed ? (
<>
Пройдено<Icon type="check" style={{ color: "green" }} />
</>
        ) : (
<>
Непройдено<Icon type="close" style={{ color: "red" }} />
</>
        )}
</div>
    }
    actions={
test.completed
        ? []
        : [
<Button type="primary" ghost>
<Link to={`/tests/${test.id}`}>Пройти</Link>
</Button>
          ]
    }
>
    {test.description}
<br />
Кількістьспроб: {test.numberAttempts}
</Card>
```

Для перевірки правильності відповідей користувача на тестбуло написано таку функцію:

```
checkResult() {
   let correctAnswersCount = 0;

this.questions.forEach(question => {
    if (question.answersRadio) {
      if (question.userAnswer === question.correctAnswer) {
        correctAnswersCount += 1;
      }
    } else {
      const isAllInputsCorrect = Object.keys(question.userAnswer).every(
key =>question.userAnswer[key] === question.correctAnswer[key]
      );

      if (isAllInputsCorrect) {
```

```
          correctAnswersCount += 1;
        }
      }
    });

this.result.finished = true;
this.result.questionsCount = this.questions.length;
this.result.correctAnswersCount = correctAnswersCount;
  if (
    Math.round(
      (this.result.correctAnswersCount / this.result.questionsCount) * 100
    ) >= 67
  ) {
this.result.passed = true;
this.completed = true;
  } else {
this.result.passed = false;
  }
this.numberAttempts++;
this.questions.forEach(question => question.clearUserAnswer());
this.resetCurrentQuestion();
 }
```

За домогою цих функцій було створено весь алгоритм роботитренажеру. Повний код програми наведено у додатку А.

4.2 Технології реалізації тренажеру

4.2.1 Visual Studio Code

Visual Studio Code — засіб для створення, редагування та зневадження сучасних веб-застосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X [4].

Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціонується як легковагове рішення, що дозволяє обійтися без повного інтегрованого середовища розробки. Серед підтримуваних мов і технологій: JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F#, DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown, JSON, HTML, CSS, LESS і SASS.

4.2.2 JavaScript

JavaScript (JS) — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки [2].

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Мова JavaScript використовується для:

— написання сценаріїв веб-сторінок для надання їм інтерактивності;
— створення односторінкових веб-застосунків (React, AngularJS, Vue.js);
— програмування на стороні сервера (Node.js);
— стаціонарних застосунків (Electron, NW.js);
— мобільних застосунків (React Native, Cordova);
— сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);
— всередині PDF-документів тощо.

4.2.3 React

React — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників [3].

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

4.2.4 Mobx

MobX - це перевірена в бою бібліотека, яка робить управління станом простим і масштабується за рахунок прозорого застосування функціонально-реактивного програмування (TFRP). Філософія MobX дуже проста: Все, що може бути виведено зі стану додатки, має бути отримано. Автоматично.

React і MobX разом являють собою потужну комбінацію. React відображає

стан додатки, надаючи механізми для його перетворення в дерево візуалізуються компонентів. MobX надає механізм для зберігання та оновлювати додатки, яке потім використовує React.

І React, і MobX надають оптимальні і унікальні рішення загальних проблем при розробці додатків. React надає механізми для оптимальної візуалізації призначеного для користувача інтерфейсу за допомогою віртуального DOM, що зменшує кількість дорогих мутацій DOM. MobX надає механізми для оптимальної синхронізації стану програми з вашими компонентами React, використовуючи реактивний граф станів віртуальної залежності, який оновлюється тільки в разі крайньої необхідності і ніколи не застаріє.

4.2.4 Node.js

Node.js — платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Засновником платформи є Раян Дал (Ryan Dahl). Якщо раніше Javascript застосовувався для обробки даних в браузері на сторонні користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їх виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників [6].

Node.js має наступні властивості:

- асинхронна однопоточна модель виконання запитів;
- не блокуючий ввід/вивід;
- система модулів на базі CommonJS;
- JavaScript Google V8 для виконання коду;

Для керування модулями використовується пакетний менеджер npm (node package manager).

В рамках цього проекту, node.js платформа буде виконувати різні модулі, які потрібні для компіляції Angular та TypeScript.

4.2.5 JSON

JSON (англ. JavaScript Object Notation) - текстовий формат обміну даними

заснований на JavaScript. Як і багато інших текстові формати, JSON легко читається людьми. Формат JSON був розроблений Дугласом Крокфорд.

Незважаючи на походження від JavaScript формат вважається незалежним від мови і може використовуватися практично з будь-якою мовою програмування. Для багатьох мов існує готовий код для створення і обробки даних в форматі JSON.

За рахунок своєї лаконічності в порівнянні з XML, формат JSON може бути більш підходящим для серіалізації складних структур. Якщо говорити про веб-додатках, в такому ключі він доречний в задачах обміну даними як між браузером і сервером (AJAX), так і між самими серверами (програмні HTTP-сполучення).

Оскільки формат JSON є підмножиною синтаксису мови JavaScript, то він може бути швидко десеріалізован вбудованою функцією eval (). Крім того, можлива вставка цілком працездатних JavaScript-функцій. У мові PHP, починаючи з версії 5.2.0, підтримка JSON включена в ядро у вигляді функцій json_decode () і json_encode (), які самі перетворять типи даних JSON в відповідні типи PHP і навпаки.

4.2.6 SPA

Односторінковий додаток (англ. Single page application, SPA) - це веб-додаток або веб-сайт, який використовує єдиний HTML-документ як оболонку для всіх веб-сторінок і організуючий взаємодія з користувачем через динамічно завантажуємі HTML, CSS, JavaScript, зазвичай за допомогою AJAX [9].

ОП нагадують рідні (native) додатки, з тією лише різницею, що виконуються в рамках браузера, а не у власному процесі операційної системи[16].

Основними елементами, що використовуються при побудові SPA, є:

— фреймворки для JavaScript, зокрема MVC і MVVM-фреймворки

— роутінг: навігація між сторінками проводиться у фронтенді

— шаблонизатор

— HTML5

— API для бекенда, наприклад, в стилі REST

— Ajax

Переваги:

— додатки на SPA відмінно працюють на пристроях як стаціонарних, так і мобільних. "Великі" комп'ютери, планшети, смартфони, і, в кінці-кінців, прості телефони (деякі) можуть безперешкодно працювати з сайтами побудованих за принципом SPA.

— багатий користувальницький інтерфейс, так званий User Experience. Так як web-сторінка одна, побудувати багатий, насичений користувальницький інтерфейс набагато простіше. Простіше зберігати інформацію про сеанс, управляти станами уявлень (views) і керувати анімацією (в деяких випадках).

— SPA істотно (в рази) скорочує так звані "ходіння по колу", тобто завантаження одного і того ж контенту знову і знову. Так, кешування даних на даному етапі розвитку WWW досягло найвищих результатів, але якщо нічого кешувати, то і час, і ресурси на це не витрачаються.

4.3 Опис програми

На рисунку 4.2 представлений скріншот зовнішнього вигляду програми.

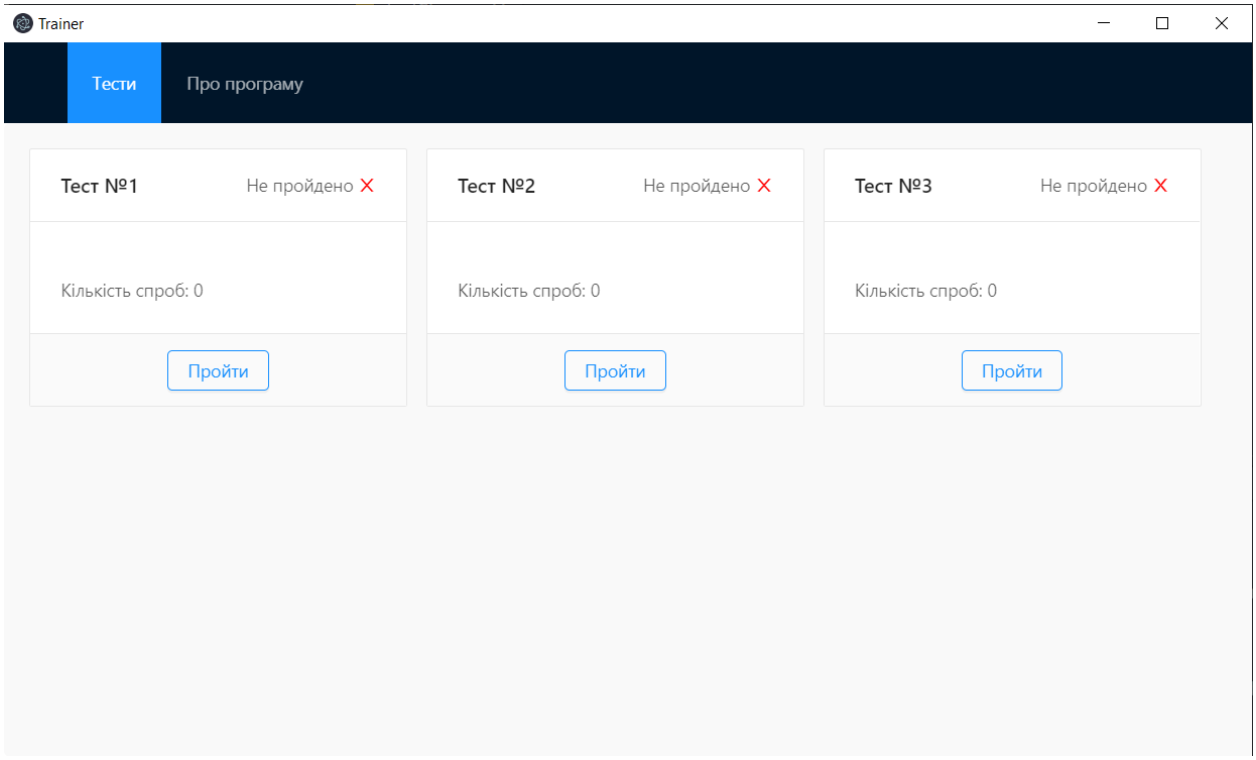


Рисунок 4.2 – Панель доступних тестів сторінка

Функціонал цього вікна було створено за допомогою цього коду:

```
const TestCard = observer(({ test }) => {
  return (
<Card
    title={test.title}
    extra={
<div>
        {test.completed ? (
<>
Пройдено<Icon type="check" style={{ color: "green" }} />
</>
        ) : (
<>
Непройдено<Icon type="close" style={{ color: "red" }} />
</>
```

```
        )}
</div>
    }
    actions={
test.completed
        ? []
        : [
<Button type="primary" ghost>
<Link to={`/tests/${test.id}`}>Пройти</Link>
</Button>
          ]
    }
>
    {test.description}
<br />
Кількістьспроб: {test.numberAttempts}
</Card>
 );
});

const Tests = observer(() => {
 const testStore = useTestStore();

 return (
<Wrapper>
<TestList>
      {testStore.tests.map(test =><TestCard key={test.id} test={test} />)}
</TestList>
</Wrapper>
 );
});
```

На даному скріншоті зображене вікно програми, що містить два посилання на «Тести» та «Про програму».

Для створення

При натисканні на «Тести» відкривається перехід на панель тестів, які доступні студентові.

При натисканні на «Про програму» відкривається інформація про розробника.

На панелі «Тести» присутні карточки всіх тестів, на кожній карточці вказано:

— назва тесту;

— статус тесту (пройдено / не пройдено);

— кількість спроб;

— кнопка для проходження тесту.

Після натискання кнопки пройти запускається проходження вибраного тесту як на рисунку 4.3.

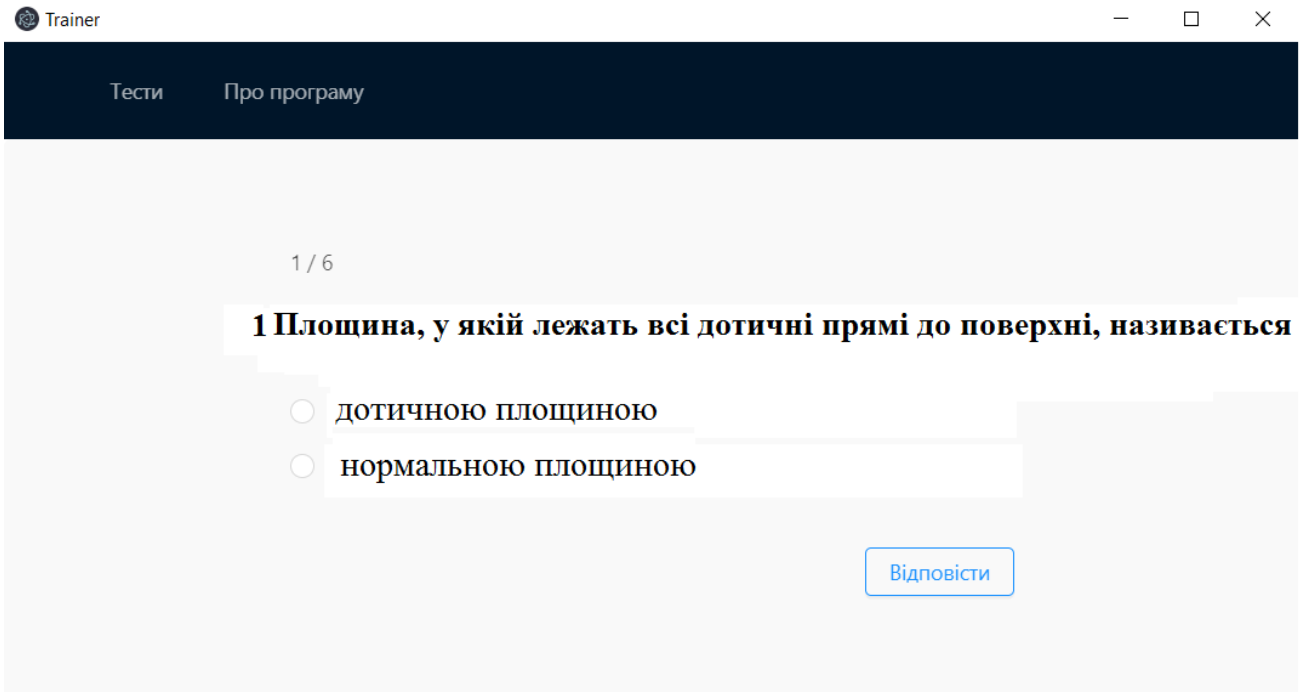


Рисунок 4.3 – Вибір варіанту відповіді

Процес проходження цієї частини тесту реалізовується наступним функціоналом:

```
const PassingTest = observer(() => {
  const { id } = useParams();
  const testStore = useTestStore();

  const test = testStore.getTestById(id);

useEffect(() => {
test.startPassing();
  }, []);

  const onSubmit = event => {
event.preventDefault();

    const testQuestion = test.questions[test.currentQuestion];

    if (testQuestion.answersInput) {
      const userAnswer = {};
      Object.keys(testQuestion.correctAnswer).forEach(key => {
        userAnswer[key] = event.target.elements[key].value.replace(",", ".");
      });

      testQuestion.userAnswer = userAnswer;
    }

test.reply();
  };

  return (
```

```
<Wrapper>
    {test.result.finished ? (
<TestResult title={test.title} result={test.result} />
    ) : (
<Question>
<form onSubmit={onSubmit}>
<div>
          {test.currentQuestion + 1} / {test.questions.length}
</div>

<br />

<Answers question={test.questions[test.currentQuestion]} />

<ButtonWrapper>
<Button htmlType="submit" type="primary" ghost>
Відповісти
</Button>
</ButtonWrapper>
</form>
</Question>
    )}
</Wrapper>
  );
});

const Answers = observer(({ question }) => {
  const onChange = event => {
```

```
question.userAnswer = event.target.value;
  };

  return (
<Wrapper>
<Typography.Title level={3}>
<span dangerouslySetInnerHTML={{ __html: question.text }} />
</Typography.Title>

<Radio.Group value={question.userAnswer} onChange={onChange}>
        {question.answersRadio&&
question.answersRadio.map(answer => (
<Radio key={answer.value} value={answer.value}>
<span
              className="answer"
              dangerouslySetInnerHTML={{ __html: answer.text }}
            />
</Radio>
         ))}

        {question.answersInput&& (
<span className="answer">{question.answersInput.text}</span>
        )}
</Radio.Group>
</Wrapper>
  );
});
```

Після вибору відповіді, при натисненні на кнопку «Відповісти» відбувається перехід до наступного завдання.

На рисунку 4.4. зображено теоретичне питання з вибором варіанту відповіді, є ще два типи завдань, а саме:

1. практичне завдання з вибором варіанту відповіді на рисунку 4.5;
2. практичне завдання з ручним внесенням відповіді на рисунку 4.6

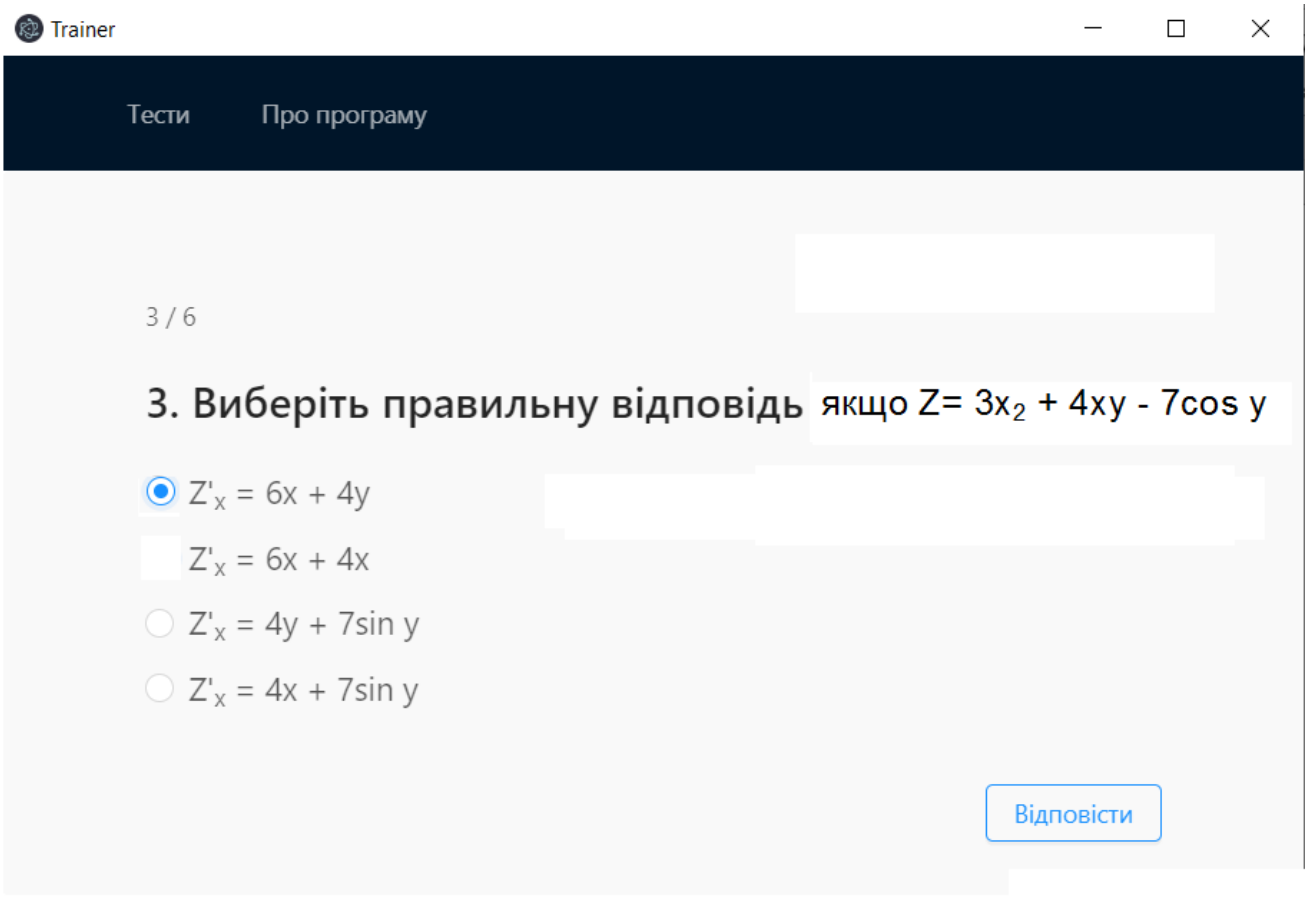


Рисунок 4.5 – Практичне завдання з виборов варіанту відповіді

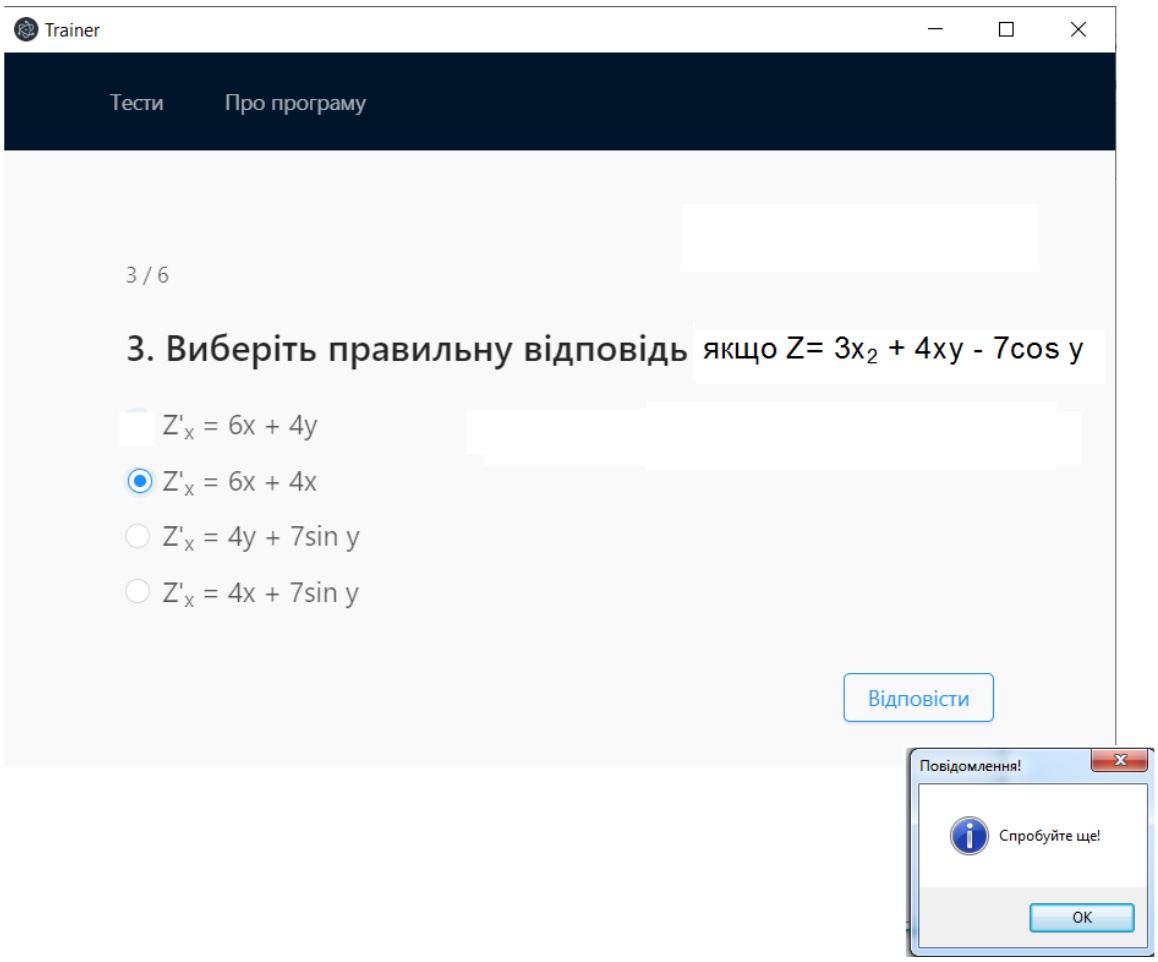


Рисунок 4.5а – Практичне завдання з виборов варіанту відповіді (невірний вибір)

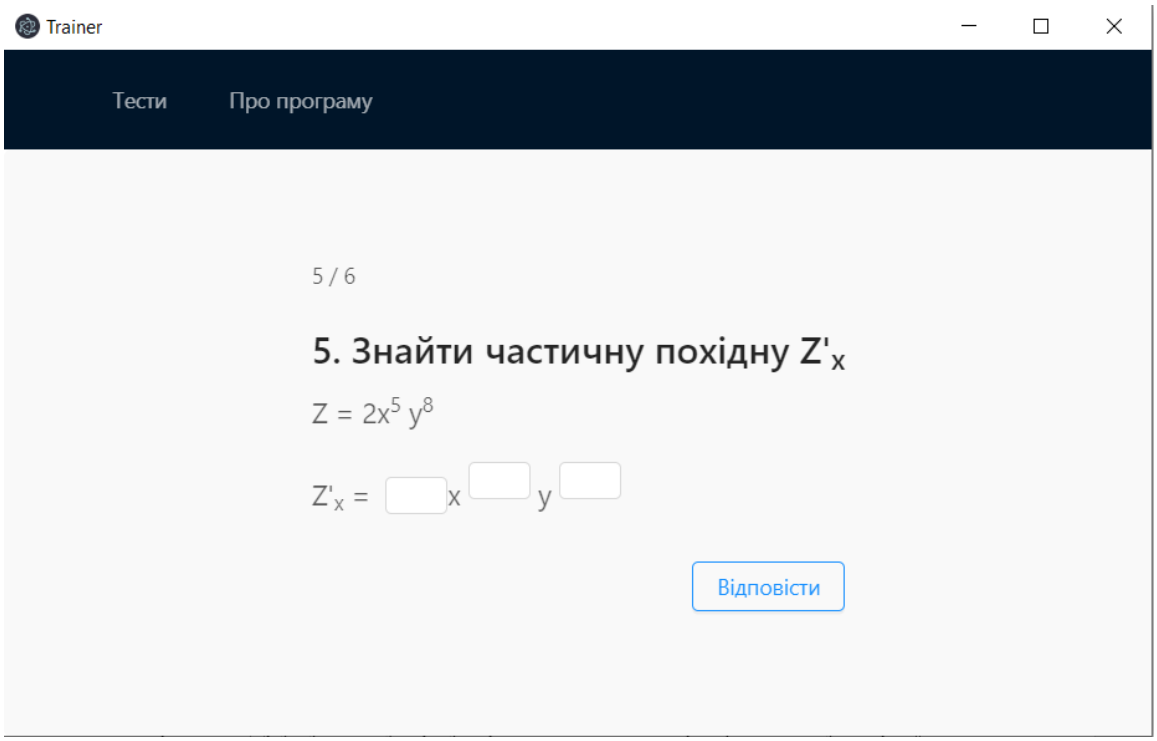


Рисунок 4.6 – Практичне завдання з ручним внесенням результату

Код для практичного запитання з вибором варіанту відповіді та практичне запитання з ручним внесенням відповіді являється спільним, при виводі завдання перевіряється чи є в запитання поле «answersRadio», якщо є то виводяться варіанти відповідей, та чи є поле «answersInput», якщо так то виводиться приклад з вставленими інпутами для вводу відповіді реалізлвано з допомогою функціоналу

```
const Answers = observer(({ question }) => {
  const onChange = event => {
    question.userAnswer = event.target.value;
  };

  return (
    <Wrapper>
      <Typography.Title level={3}>
        <span dangerouslySetInnerHTML={{ __html: question.text }} />
      </Typography.Title>

      <Radio.Group value={question.userAnswer} onChange={onChange}>
        {question.answersRadio &&
          question.answersRadio.map(answer => (
            <Radio key={answer.value} value={answer.value}>
              <span
                className="answer"
                dangerouslySetInnerHTML={{ __html: answer.text }}
              />
            </Radio>
          ))}

        {question.answersInput && (
          <span className="answer">{question.answersInput.text}</span>
        )}
      </Radio.Group>
    </Wrapper>
  );
});
```

Після відповіді на всі завдання, відобразиться результат з кількістю правильних відповідей. При успішному проходженні тесту на рисунку 4.7. та при провальному на рисунку 4.8.

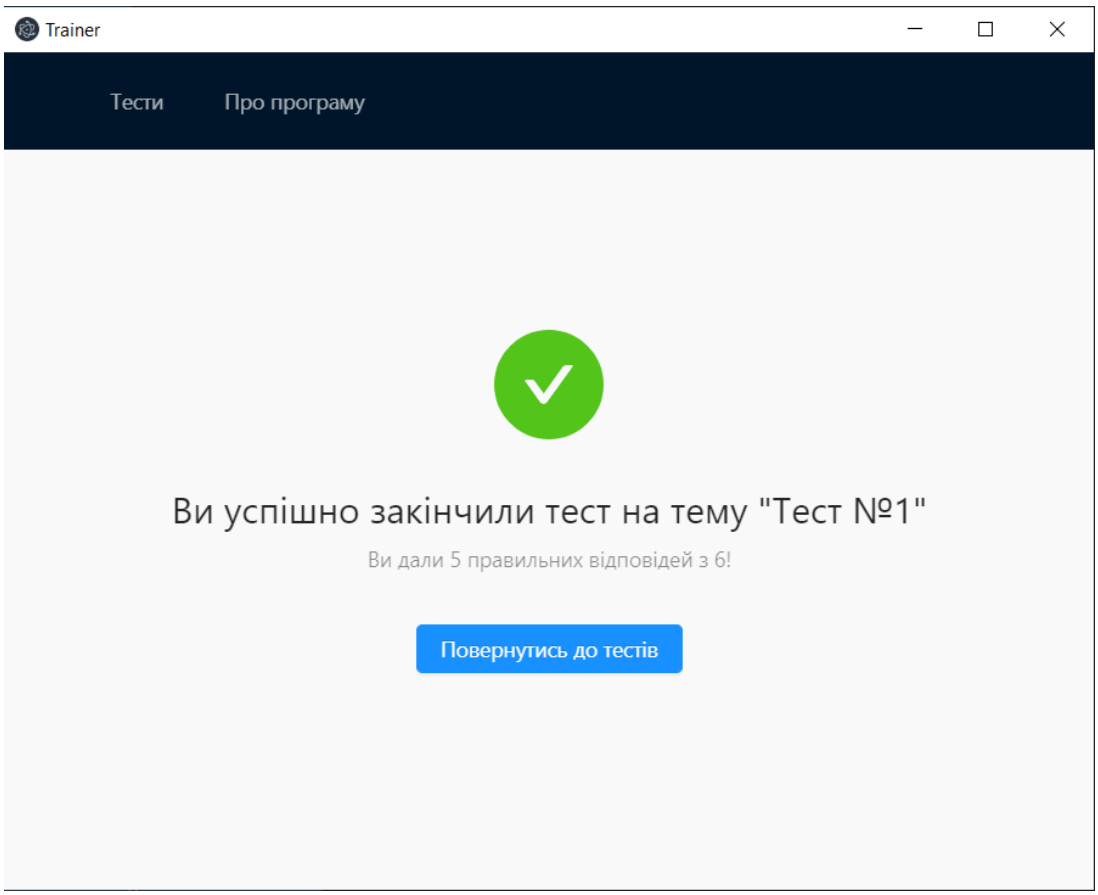


Рисунок 4.7 – Успішне проходження тесту

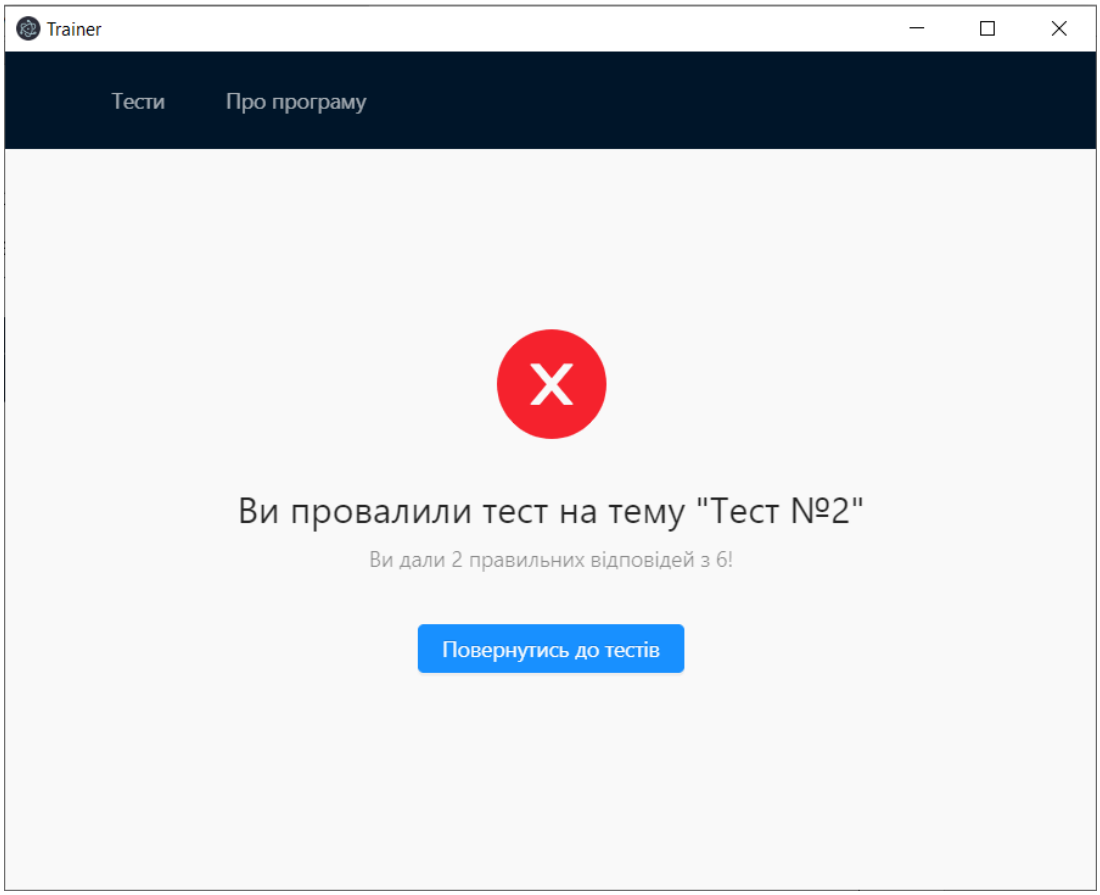


Рисунок 4.8 – Не успішне проходження тесту

Код виводу результату:

```
const TestResult = observer(({ title, result }) => {
  return (
<Result
      status={result.passed ? "success" : "error"}
      title={
result.passed
          ? `Виуспішнозакінчилитестнатему "${title}"`
: `Ви провалили тест на тему "${title}"`
      }
subTitle={`Ви дали ${result.correctAnswersCount} правильних відповідей з ${result.questionsCount}!`}
```

```
extra={[
<Button key="2" type="primary">
<Link to="/tests">Повернутись до тестів</Link>
</Button>
]}
    />
  );
});
```

4.4 Необхідна користувачу програми інструкція

1. Для того щоб почати роботу потрібно зайти на сайт за посиланням https://maxim-koylo-trainer.herokuapp.com/
2. В кладці «Тести» присутні всі доступні для проходження тести, для початку проходження тесту потрібно натиснути на кнопку «Пройти» зображеної на рисунку 4.9
3. Проходження тесту зображено на рисунку 4.10
4. Результати проходження тесту зображені на рисунку 4.11

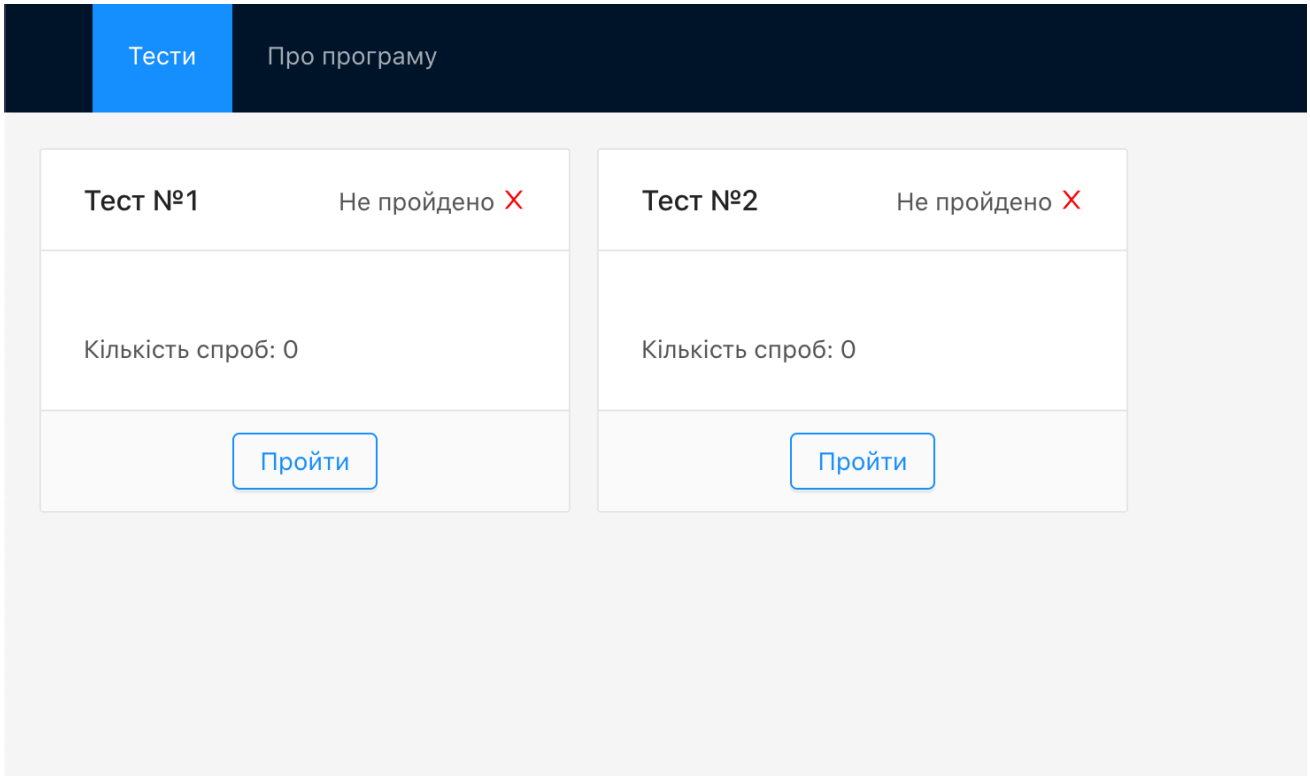


Рисунок 4.9 – Список доступних тестів

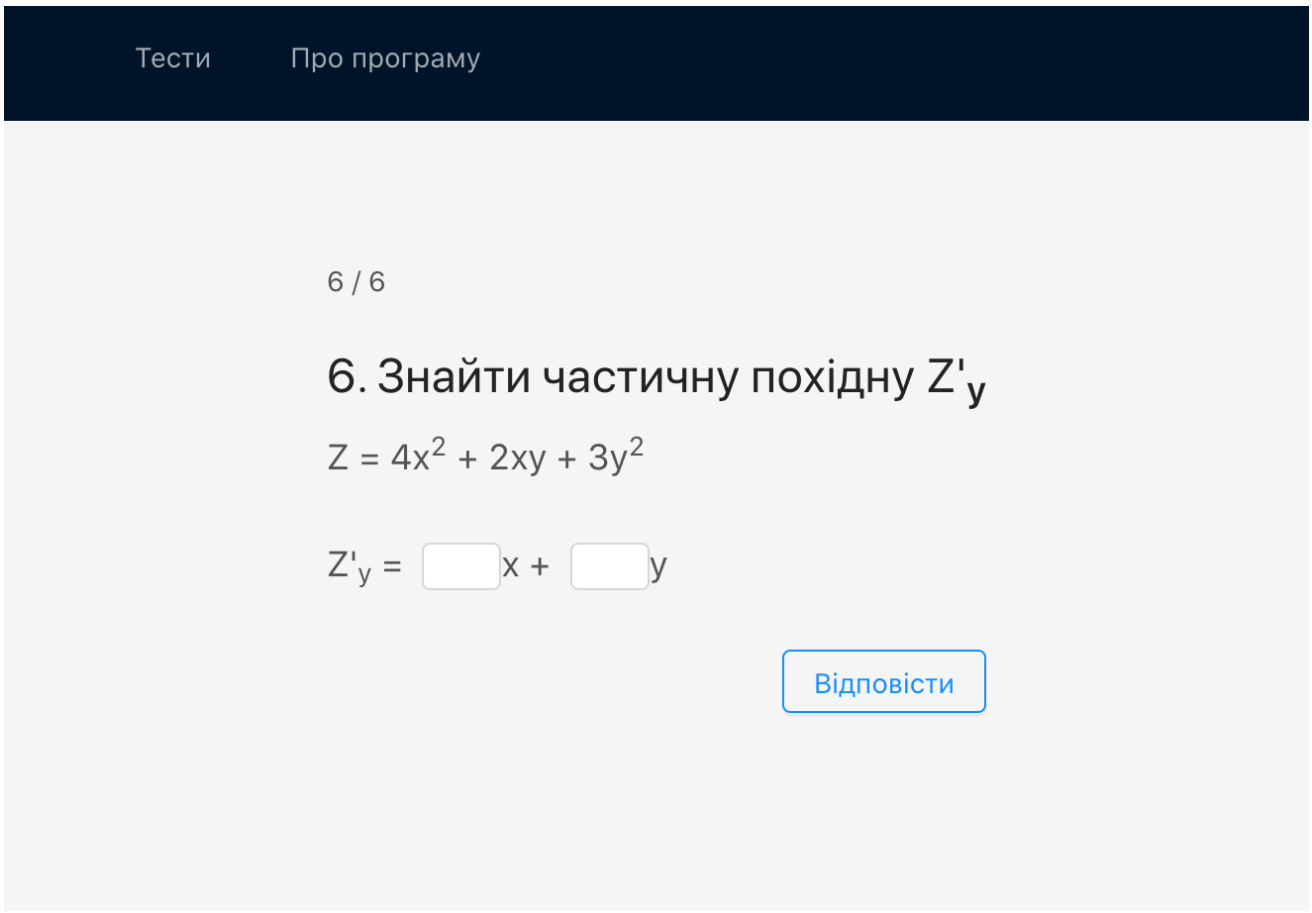


Рисунок 4.10 – Проходження тесту

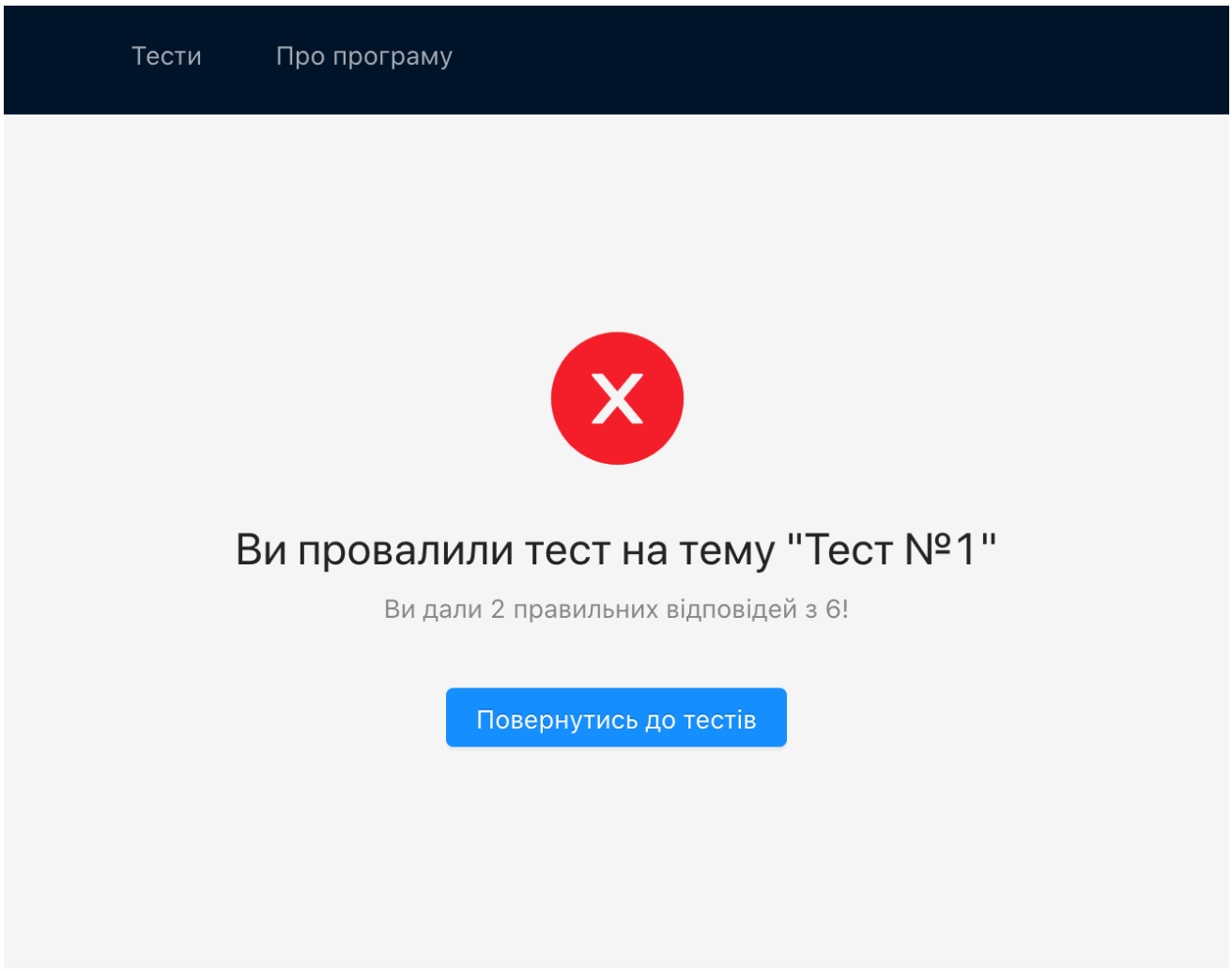


Рисунок 4.11 – Результат проходження тесту

ВИСНОВКИ

Тренажери в дистанційній освіті займають одне з важливих місць, тому їх кількість потрібно збільшувати, щоб забезпечити якомога більше дисциплін ними, а якість покращувати, щоб користувач отримував максимум користі від їх використання.

У рамках бакалаврської роботи було розроблено тренажер з теми «Диференціювання функції двох змінних». Додаток можна застосовувати для перевірки знань студентів з даної теми.

Тренажери допомагають студентам самостійно покращувати й перевіряти свої навички і теоретичні знання з певних дисциплін. Також віртуальні тренажери можуть бути застосовані не тільки під час самостійних робіт, але й при проведені лабораторних занять.

Результати виконаної роботи:

— проведений огляд деяких систем дистанційної освіти;

— розглянуто деякі елементи тренажерів та їх недоліки і переваги;

— розроблено основні вимоги до тренажера;

— розглянуті інтерактивні технології навчання, їх порівняння;

— розроблено алгоритми роботи web-додатку тренажеру;

— розробка компонентів тренажеру;

— виконана демонстрація роботи реалізованих елементів тренажеру.

Тренажер було створено за допомогою наступних технологій:

— мова програмування JavaScript;

— бібліотека React для створення інтерфейсу;

— бібліотека Mobx для управління стану.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Диференціал (математика), Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу URL: https://uk.wikipedia.org/wiki/Диференціал_(математика) – Назва з екрану (дата звернення 20.12.2019).
2. JavaScript, Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу URL: https://uk.wikipedia.org/wiki/JavaScript – Назва з екрану (дата звернення 20.12.2019).
3. React, Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу URL: https://uk.wikipedia.org/wiki/React – Назва з екрану (дата звернення 20.12.2019).
4. Visual Studio Code, Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу URL: https://uk.wikipedia.org/wiki/Visual_Studio_Code – Назва з екрану (дата звернення 10.11.2019).
5. Visual Studio Code, Official documentation [Електронний ресурс] – Режим доступу до ресурсу URL: https://code.visualstudio.com/docs – Назва з екрану (дата звернення 10.11.2019).
6. Node.js — Вікіпедія [Електронний ресурс]. – 2020. – https://uk.wikipedia.org/wiki/nodejs – Дата доступу: серп. 2020. – Назва з екрану.
7. ATutor — Вікіпедія [Електронний ресурс]. – 2019. – https://uk.wikipedia.org/wiki/ ATutor – Дата доступу: жовт. 2020. – Назва з екрану.
8. ATutor User Documentation. [Електронний ресурс]. – Режим доступу: http://help.atutor.ca/general/.
9. SPA — Вікіпедія [Електронний ресурс]. – 2019. – https://uk.wikipedia.org/wiki/SPA – Дата доступу: черв. 2020. – Назва з екрану.
10. Ємець О. О. Про розробку тренажерів для дистанційних курсів кафедрою ММСІ ПУЕТ / О.О. Ємець // Інформатика та системні науки (ІСН-2015): матеріали VI Всеукраїнської науково-практичної конференції за

міжнародною участю, (м. Полтава, 19–21 берез. 2015 р.). – Полтава: ПУЕТ, 2015.

11. Ольховська О. В. Технології підтримки системи дистанційного навчання в Полтавському університеті економіки і торгівлі / О. В. Ольховська.

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

```
import React from "react";
import ReactDOM from "react-dom";
import { Router } from "react-router-dom";
import { createBrowserHistory } from "history";

import { syncHistoryWithStore } from "mobx-react-router";

import { routingStore } from "rootStore";

import App from "./App";

import "antd/dist/antd.css";
import "./index.css";

const browserHistory = createBrowserHistory();

const history = syncHistoryWithStore(browserHistory, routingStore);

ReactDOM.render(
<Router history={history}>
<App />
</Router>,
document.getElementById("root")
);

import React from "react";
import { Switch, Route, Link } from "react-router-dom";

import { Layout, Menu } from "antd";

import Home from "modules/home";
import Tests from "modules/tests";
import PassingTest from "modules/passingTest";

import { observer } from "mobx-react";
import { useStore } from "shared/hooks/useStore";

import styled from "styled-components/macro";

const MainContent = styled(Layout.Content)`
  height: 100vh;
`;

const Header = styled(Layout.Header)`
```

```
  position: sticky;
  top: 0;
  width: 100%;
  z-index: 1;
`;

const Content = styled(Layout.Content)`
  height: calc(100vh - 64px);
  padding: 20px;
  background-color: #f9f9f9;
  border-radius: 5px;
`;

const About = () => {
  return <div>©{new Date().getFullYear()} СтвореноКойлоМаксимом</div>;
};

const App = observer(() => {
  const { routing } = useStore();

  if (routing.location.pathname === "/") {
routing.push("/tests");
  }

  return (
<div className="app">
<MainContent>
<Header>
<Menu
        theme="dark"
        mode="horizontal"
        selectedKeys={[routing.location.pathname]}
        style={{ lineHeight: "64px" }}
>
        {/* <Menu.Item key="/">
<Link to="/">Главная</Link>
</Menu.Item> */}

<Menu.Item key="/tests">
<Link to="/tests">Тести</Link>
</Menu.Item>

<Menu.Item key="/about">
<Link to="/about">Пропрограму</Link>
</Menu.Item>
</Menu>
</Header>
```

```
<Content>
<Switch>
<Route exact path="/" component={Home} />
<Route exact path="/tests" component={Tests} />
<Route path="/tests/:id" component={PassingTest} />
<Route path="/about" component={About} />
</Switch>
</Content>
</MainContent>

    {/* <Modals /> */}
</div>
  );
});

export default App;

import { RouterStore } from "mobx-react-router";

import { TestStore } from "modules/tests/store";

const routingStore = new RouterStore();

class RootStore {
constructor() {
this.routing = routingStore;
this.testStore = new TestStore(this);
  }
}

export { RootStore, routingStore };

import React from "react";

import TestCard from "./TestCard";

import { observer } from "mobx-react";
import { useTestStore } from "shared/hooks/tests/useTestStore";

import styled from "styled-components/macro";

const Wrapper = styled.div``;

const TestList = styled.div`
  display: flex;
  flex-wrap: wrap;
```

```
.ant-card {
   margin: 0 15px 15px 0;
   width: 300px;
 }
`;

const Tests = observer(() => {
 const testStore = useTestStore();

 return (
<Wrapper>
<TestList>
      {testStore.tests.map(test =><TestCard key={test.id} test={test} />)}
</TestList>
</Wrapper>
 );
});

export default Tests;

import React from "react";
import { Link } from "react-router-dom";

import { Card, Button, Icon } from "antd";

import { observer } from "mobx-react";

const TestCard = observer(({ test }) => {
 return (
<Card
    title={test.title}
    extra={
<div>
        {test.completed ? (
<>
Пройдено<Icon type="check" style={{ color: "green" }} />
</>
        ) : (
<>
Непройдено<Icon type="close" style={{ color: "red" }} />
</>
        )}
</div>
    }
    actions={
test.completed
```

```
        ? []
        : [
<Button type="primary" ghost>
<Link to={`/tests/${test.id}`}>Пройти</Link>
</Button>
          ]
    }
>
    {test.description}
<br />
Кількістьспроб: {test.numberAttempts}
</Card>
  );
});

export default TestCard;

import React from "react";
import { Input } from "antd";

import { observable, computed, decorate, toJS } from "mobx";

import { TestModel } from "./models/test";

const initialTests = [
  {
    id: "1",
    title: "Тест №1",
    description: "",
    numberAttempts: 0,
    questions: [
      {
        id: "1",
        text: "1. Середньоюшвидкістюзмінифункціїє: ",
        answersRadio: [
          {
            value: "1",
            text: "Відношенняприростуфункціїдоприростуаргументу."
},
          {
            value: "2",
            text: "Відношення приросту аргументу до приросту функції."
          }
        ],
        correctAnswer: "1"
      },
      {
```

```
            id: "2",
            text: "2. Швидкістю зміни функції є: ",
            answersRadio: [
              {
                value: "1",
                text:
                  "Границя відношення приросту функції до приросту аргументу<br /> за умови,
що приріст аргументу прямує до безкінечності."
              },
              {
                value: "2",
                text:
                  "Границя відношення приросту функції до приросту аргументу<br /> за умови,
що приріст аргументу не прямує до безкінечності."
              },
              {
                value: "3",
                text:
                  "Границя відношення приросту функції до приросту аргументу<br /> за умови,
що приріст аргументу прямує до нуля"
              },
              {
                value: "4",
                text:
                  "Границя відношення приросту функції до приросту аргументу<br /> за умови,
що приріст аргументу не прямує до нуля"
              }
            ],
            correctAnswer: "3"
          },
          {
            id: "3",
            text: "3. Виберіть правильну відповідь 3x<sup>2</sup> + 4xy - 7cos y",
        answersRadio: [
              {
                value: "1",
                text: "Z'<sub>x</sub> = 6x + 4y"
              },
              {
                value: "2",
                text: "Z'<sub>x</sub> = 6x + 4x"
              },
              {
                value: "3",
                text: "Z'<sub>x</sub> = 4y + 7sin y"
              },
              {
```

```
          value: "4",
          text: "Z'<sub>x</sub> = 4x + 7sin y"
        }
      ],
      correctAnswer: "1"
    },
    {
      id: "4",
      text: "4. Виберітьправильнувідповідь Z'<sub>y</sub> = 4y + 7sin y",
      answersRadio: [
        {
          value: "1",
          text: "Z'<sub>y</sub> = 6x + 4y"
        },
        {
          value: "2",
          text: "Z'<sub>y</sub> = 4x + 7sin y"
        },
        {
          value: "3",
          text: "Z'<sub>y</sub> = 4y + 7sin y"
        },
        {
          value: "4",
          text: "Z'<sub>y</sub> = 6x + 4sx"
        }
      ],
      correctAnswer: "2"
    },
    {
      id: "5",
      text: "5. Знайтичастичнупохідну Z'<sub>x</sub>",
      answersInput: {
        text: (
<>
          Z = 2x<sup>5</sup> y<sup>8</sup>
<br />
<br />
          Z'<sub>x</sub> = <Input name="1-5-1" size="small" />x
<sup style={{ top: "-0.7em" }}>
<Input name="1-5-2" size="small" />
</sup>{" "}
          y
<sup style={{ top: "-0.7em" }}>
<Input name="1-5-3" size="small" />
</sup>
</>
```

```
        )
       },
       correctAnswer: { "1-5-1": "10", "1-5-2": "4", "1-5-3": "8" }
      },
      {
       id: "6",
       text: "6. Знайтичастичнупохідну Z'<sub>y</sub>",
       answersInput: {
        text: (
<>
          Z = 4x<sup>2</sup> + 2xy + 3y<sup>2</sup>
<br />
<br />
          Z'<sub>y</sub> = <Input name="1-6-1" size="small" />x +{" "}
<Input name="1-6-2" size="small" />y
</>
        )
       },
       correctAnswer: { "1-6-1": "2", "1-6-2": "6" }
      }
    ]
   },
   {
    id: "2",
    title: "Тест №2",
    description: "",
    numberAttempts: 0,
    questions: [
     {
      id: "1",
      text: "1. Середньоюшвидкістюзмінифункції є: ",
      answersRadio: [
       {
        value: "1",
        text: "Відношенняприростуфункціїдоприростуаргументу."
},
       {
        value: "2",
        text: "Відношення приросту аргументу до приросту функції."
       }
      ],
      correctAnswer: "1"
     },
     {
      id: "2",
      text: "2. Швидкістю зміни функції є: ",
      answersRadio: [
```

```
            {
              value: "1",
              text:
                "Границя відношення приросту функції до приросту аргументу<br /> за умови, що приріст аргументу прямує до безкінечності."
            },
            {
              value: "2",
              text:
                "Границя відношення приросту функції до приросту аргументу<br /> за умови, що приріст аргументу не прямує до безкінечності."
            },
            {
              value: "3",
              text:
                "Границя відношення приросту функції до приросту аргументу<br /> за умови, що приріст аргументу прямує до нуля"
            },
            {
              value: "4",
              text:
                "Границя відношення приросту функції до приросту аргументу<br /> за умови, що приріст аргументу не прямує до нуля"
            }
          ],
          correctAnswer: "3"
        },
        {
          id: "3",
          text: "3. Виберіть правильну відповідь 3x<sup>2</sup> + 4xy - 7cos y",
      answersRadio: [
            {
              value: "1",
              text: "Z'<sub>x</sub> = 6x + 4y"
            },
            {
              value: "2",
              text: "Z'<sub>x</sub> = 6x + 4x"
            },
            {
              value: "3",
              text: "Z'<sub>x</sub> = 4y + 7sin y"
            },
            {
              value: "4",
              text: "Z'<sub>x</sub> = 4x + 7sin y"
            }
```

```
      ],
      correctAnswer: "1"
    },
    {
      id: "4",
      text: "4. Виберітьправильнувідповідь Z'<sub>y</sub> = 4y + 7sin y",
      answersRadio: [
        {
          value: "1",
          text: "Z'<sub>y</sub> = 6x + 4y"
        },
        {
          value: "2",
          text: "Z'<sub>y</sub> = 4x + 7sin y"
        },
        {
          value: "3",
          text: "Z'<sub>y</sub> = 4y + 7sin y"
        },
        {
          value: "4",
          text: "Z'<sub>y</sub> = 6x + 4sx"
        }
      ],
      correctAnswer: "2"
    },
    {
      id: "5",
      text: "5. Знайтичастичнупохідну Z'<sub>x</sub>",
      answersInput: {
        text: (
<>
            Z = 2x<sup>5</sup> y<sup>8</sup>
<br />
<br />
            Z'<sub>x</sub> = <Input name="1-5-1" size="small" />x
<sup style={{ top: "-0.7em" }}>
<Input name="1-5-2" size="small" />
</sup>{" "}
            y
<sup style={{ top: "-0.7em" }}>
<Input name="1-5-3" size="small" />
</sup>
</>
        )
      },
      correctAnswer: { "1-5-1": "10", "1-5-2": "4", "1-5-3": "8" }
```

```
    },
    {
     id: "6",
     text: "6. Знайтичастичнупохідну Z'<sub>y</sub>",
answersInput: {
      text: (
<>
        Z = 4x<sup>2</sup> + 2xy + 3y<sup>2</sup>
<br />
<br />
        Z'<sub>y</sub> = <Input name="1-6-1" size="small" />x +{" "}
<Input name="1-6-2" size="small" />y
</>
       )
     },
     correctAnswer: { "1-6-1": "2", "1-6-2": "6" }
    }
   ]
  }
];

class TestStore {
  tests = [];

  constructor(rootStore) {
this.rootStore = rootStore;

this.tests = initialTests.map(test => new TestModel(this, test));
  }

  getTestById(id) {
   return this.tests.find(test => test.id === id);
  }

  get toJS() {
   return toJS(this);
  }
}

decorate(TestStore, { tests: observable, toJS: computed });

export { TestStore };

import { observable, computed, action, decorate, toJS } from "mobx";

import { QuestionModel } from "./question";
```

```
const initialResult = {
  finished: false,
  passed: false,
  questionsCount: 0,
  correctAnswersCount: 0
};

class TestModel {
  id;
  title;
  description;
  completed;
  numberAttempts;
  questions;

  currentQuestion = 0;

  result = initialResult;

constructor(
testStore,
    {
id,
title = "",
description = "",
completed = false,
numberAttempts = 0,
questions = []
    }
  ) {
this.testStore = testStore;

this.id = id || Date.now();
this.title = title;
this.description = description;
this.completed = completed;
this.numberAttempts = numberAttempts;
this.questions = questions.map(
question => new QuestionModel(this, question)
    );
  }

startPassing() {
this.result = initialResult;
  }

reply() {
```

```
    if (this.result.finished === false) {
this.currentQuestion += 1;

      if (this.currentQuestion === this.questions.length) {
this.checkResult();
      }
    }
  }

checkResult() {
    let correctAnswersCount = 0;

this.questions.forEach(question => {
      if (question.answersRadio) {
        if (question.userAnswer === question.correctAnswer) {
          correctAnswersCount += 1;
        }
      } else {
        const isAllInputsCorrect = Object.keys(question.userAnswer).every(
key =>question.userAnswer[key] === question.correctAnswer[key]
        );

        if (isAllInputsCorrect) {
          correctAnswersCount += 1;
        }
      }
    });

this.result.finished = true;
this.result.questionsCount = this.questions.length;
this.result.correctAnswersCount = correctAnswersCount;

    if (
      Math.round(
        (this.result.correctAnswersCount / this.result.questionsCount) * 100
      ) >= 67
    ) {
this.result.passed = true;
this.completed = true;
    } else {
this.result.passed = false;
    }

this.numberAttempts++;
this.questions.forEach(question => question.clearUserAnswer());
this.resetCurrentQuestion();
  }
```

```
resetCurrentQuestion() {
this.currentQuestion = 0;
  }

  get toJS() {
    return toJS(this);
  }
}

decorate(TestModel, {
  id: observable,
  title: observable,
  description: observable,
  completed: observable,
  questions: observable,

  currentQuestion: observable,
  result: observable,

  startPassing: action,
  reply: action,
  checkResult: action,
  resetCurrentQuestion: action,
  toJS: computed
});

export { TestModel };

import { observable, computed, action, decorate, toJS } from "mobx";

class QuestionModel {
  id;
  text;
  answersRadio;
  answersInput;
  correctAnswer;
  userAnswer;

constructor(
testModel,
{
id,
text = "",
answersRadio,
answersInput,
correctAnswer = "",
```

```
userAnswer = ""
    }
  ) {
this.testModel = testModel;

this.id = id || Date.now();
this.text = text;
this.answersRadio = answersRadio;
this.answersInput = answersInput;
this.correctAnswer = correctAnswer;
this.userAnswer = userAnswer;
  }

clearUserAnswer() {
this.userAnswer = undefined;
  }

  get toJS() {
    return toJS(this);
  }
}

decorate(QuestionModel, {
  id: observable,
  text: observable,
  correctAnswer: observable,
  userAnswer: observable,
  clearUserAnswer: action,
  toJS: computed
});

export { QuestionModel };

import React, { useEffect } from "react";
import { useParams } from "react-router-dom";

import { Button } from "antd";

import Answers from "./Answers";
import TestResult from "./TestResult";

import { observer } from "mobx-react";
import { useTestStore } from "shared/hooks/tests/useTestStore";

import styled from "styled-components/macro";

const Wrapper = styled.div`
```

```
  margin-top: 50px;
  display: flex;
  align-items: center;
  justify-content: center;
`;

const Question = styled.div`
  min-width: 300px;
`;

const ButtonWrapper = styled.div`
  margin-top: 30px;
  text-align: right;
`;

const PassingTest = observer(() => {
  const { id } = useParams();
  const testStore = useTestStore();

  const test = testStore.getTestById(id);

useEffect(() => {
test.startPassing();
  }, []);

  const onSubmit = event => {
event.preventDefault();

    const testQuestion = test.questions[test.currentQuestion];

    if (testQuestion.answersInput) {
      const userAnswer = {};
      Object.keys(testQuestion.correctAnswer).forEach(key => {
        userAnswer[key] = event.target.elements[key].value.replace(",", ".");
      });

      testQuestion.userAnswer = userAnswer;
    }

test.reply();
  };

  return (
<Wrapper>
      {test.result.finished ? (
<TestResult title={test.title} result={test.result} />
      ) : (
```

```
<Question>
<form onSubmit={onSubmit}>
<div>
            {test.currentQuestion + 1} / {test.questions.length}
</div>

<br />

<Answers question={test.questions[test.currentQuestion]} />

<ButtonWrapper>
<Button htmlType="submit" type="primary" ghost>
Відповісти
</Button>
</ButtonWrapper>
</form>
</Question>
     )}
</Wrapper>
  );
});

export default PassingTest;

import React from "react";

import { Typography, Radio } from "antd";

import { observer } from "mobx-react";

import styled from "styled-components/macro";

const Wrapper = styled.div`
.ant-radio-wrapper {
   margin: 10px 0;
   display: flex;
   align-items: center;
  }

.answer {
   font-size: 18px;

.ant-input {
     margin-left: 5px;
     width: 40px;
   }
  }
```

```
`;

const Answers = observer(({ question }) => {
  const onChange = event => {
question.userAnswer = event.target.value;
  };

  return (
<Wrapper>
<Typography.Title level={3}>
<span dangerouslySetInnerHTML={{ __html: question.text }} />
</Typography.Title>

<Radio.Group value={question.userAnswer} onChange={onChange}>
       {question.answersRadio&&
question.answersRadio.map(answer => (
<Radio key={answer.value} value={answer.value}>
<span
              className="answer"
              dangerouslySetInnerHTML={{ __html: answer.text }}
            />
</Radio>
         ))}

       {question.answersInput&& (
<span className="answer">{question.answersInput.text}</span>
)}
</Radio.Group>
</Wrapper>
  );
});

export default Answers;

import React from "react";
import { Link } from "react-router-dom";

import { Result, Button } from "antd";

import { observer } from "mobx-react";

const TestResult = observer(({ title, result }) => {
  return (
<Result
     status={result.passed ? "success" : "error"}
     title={
result.passed
```

```
            ? `Виуспішнозакінчилитестнатему "${title}"`
        : `Ви провалили тест на тему "${title}"`
            }
            subTitle={`Ви дали ${result.correctAnswersCount} правильних відповідей з
${result.questionsCount}!`}
        extra={[
        // <Button key="1" type="primary">
        //   <Link to="/">Наглавную</Link>
        // </Button>,
        <Button key="2" type="primary">
        <Link to="/tests">Повернутисьдотестів</Link>
        </Button>
            ]}
          />
        );
        });

        export default TestResult;

        /* eslint-disable react/jsx-props-no-spreading */
        import React from "react";
        import { Drawer, Tabs } from "antd";

        import { observer } from "mobx-react";
        import { useStore } from "shared/hooks/useStore";

        import "./style.css";

        const { TabPane } = Tabs;

        const DrawerBottom = observer(() => {
          const { drawerStore } = useStore(store => store.navigation);

          const { isOpen, tabs, activeKey } = drawerStore;

          let activeKeyAttribute = {};

          if (activeKey) {
            activeKeyAttribute = {
              activeKey
            };
          }

          return (
        <Drawer
            placement="right"
            visible={isOpen}
```

```
      className="drawer"
      onClose={() => drawerStore.close()}
>
<Tabs {...activeKeyAttribute}>
       {tabs &&
tabs.map(tab => (
<TabPane key={tab.key} tab={tab.title}>
            {tab.component}
</TabPane>
         ))}
</Tabs>
</Drawer>
  );
});

export default DrawerBottom;

import React from "react";

import { Steps } from "antd";

import { ReactComponent as OkIcon } from "shared/assets/img/form/StepOK.svg";
import { ReactComponent as ErrorIcon } from "shared/assets/img/form/StepError.svg";
import { ReactComponent as ProcessIcon } from "shared/assets/img/form/StepProcess.svg";

import styled from "styled-components/macro";

const StyledSteps = styled(Steps)`
&&& {
.ant-steps-item-finish
> .ant-steps-item-container
> .ant-steps-item-tail::after {
      background-color: #4caf50;
    }
.ant-steps-item-title {
      font-size: 12px;
      color: #535c6f;
font-weight: 600;
    }
  }
`;

const CustomSteps = ({ fields }) => {
  return (
<StyledSteps direction="vertical">
      {fields.map(field => {
        let icon;
```

```
        switch (field.status) {
          case "finish":
            icon = <OkIcon />;
            break;
          case "error":
            icon = <ErrorIcon />;
            break;
          case "process":
            icon = <ProcessIcon />;
            break;
          default:
            icon = null;
        }

        return (
<Steps.Step
            key={field.key}
            title={field.title}
            status={field.status}
            description={field.component}
            icon={icon}
          />
        );
      })}
</StyledSteps>
  );
};

export default CustomSteps;

import { observable, action, decorate } from "mobx";

class DrawerStore {
  isOpen = false;
  activeKey = null;
  tabs = null;
  options = {};

open({ tabs, activeKey = null, options = {} }) {
this.isOpen = true;
this.tabs = tabs;

    if (activeKey) {
this.activeKey = tabs[0] ? tabs[0].key : activeKey;
    } else {
this.activeKey = activeKey;
    }
```

```
this.options = options;
  }

close() {
this.isOpen = false;
  }
}

decorate(DrawerStore, {
  isOpen: observable,
  tabs: observable,
  activeKey: observable,
  open: action,
  close: action
});

export { DrawerStore };

import React, { useEffect } from "react";
import { useParams } from "react-router-dom";

import { Button } from "antd";

import Answers from "./Answers";
import TestResult from "./TestResult";

import { observer } from "mobx-react";
import { useTestStore } from "shared/hooks/tests/useTestStore";

import styled from "styled-components/macro";

const Wrapper = styled.div`
  margin-top: 50px;
  display: flex;
  align-items: center;
  justify-content: center;
`;

const Question = styled.div`
  min-width: 300px;
`;

const ButtonWrapper = styled.div`
  margin-top: 30px;
  text-align: right;
`;
```

```
const PassingTest = observer(() => {
  const { id } = useParams();
  const testStore = useTestStore();

  const test = testStore.getTestById(id);

useEffect(() => {
test.startPassing();
  }, []);

  const onSubmit = event => {
event.preventDefault();

    const testQuestion = test.questions[test.currentQuestion];

    if (testQuestion.answersInput) {
      const userAnswer = {};
      Object.keys(testQuestion.correctAnswer).forEach(key => {
        userAnswer[key] = event.target.elements[key].value.replace(",", ".");
      });

      testQuestion.userAnswer = userAnswer;
    }

test.reply();
  };

  return (
<Wrapper>
      {test.result.finished ? (
<TestResult title={test.title} result={test.result} />
      ) : (
<Question>
<form onSubmit={onSubmit}>
<div>
            {test.currentQuestion + 1} / {test.questions.length}
</div>

<br />

<Answers question={test.questions[test.currentQuestion]} />

<ButtonWrapper>
<Button htmlType="submit" type="primary" ghost>
Відповісти
</Button>
```

```
</ButtonWrapper>
</form>
</Question>
    )}
</Wrapper>
  );
});

export default PassingTest;

import React from "react";

import { Typography, Radio } from "antd";

import { observer } from "mobx-react";

import styled from "styled-components/macro";

const Wrapper = styled.div`
.ant-radio-wrapper {
   margin: 10px 0;
   display: flex;
   align-items: center;
  }

.answer {
   font-size: 18px;

.ant-input {
     margin-left: 5px;
     width: 40px;
   }
  }
`;

const Answers = observer(({ question }) => {
  const onChange = event => {
question.userAnswer = event.target.value;
  };

  return (
<Wrapper>
<Typography.Title level={3}>
<span dangerouslySetInnerHTML={{ __html: question.text }} />
</Typography.Title>

<Radio.Group value={question.userAnswer} onChange={onChange}>
```

```
        {question.answersRadio&&
      question.answersRadio.map(answer => (
      <Radio key={answer.value} value={answer.value}>
      <span
                 className="answer"
                 dangerouslySetInnerHTML={{ __html: answer.text }}
               />
      </Radio>
           ))}

        {question.answersInput&& (
      <span className="answer">{question.answersInput.text}</span>
      )}
      </Radio.Group>
      </Wrapper>
       );
      });

      export default Answers;
      import React from "react";
      import { Link } from "react-router-dom";

      import { Result, Button } from "antd";

      import { observer } from "mobx-react";

      const TestResult = observer(({ title, result }) => {
       return (
      <Result
          status={result.passed ? "success" : "error"}
          title={
      result.passed
             ? `Виуспішнозакінчилитестнатему "${title}"`
      : `Ви провалили тест на тему "${title}"`
          }
          subTitle={`Ви дали ${result.correctAnswersCount} правильних відповідей з
${result.questionsCount}!`}
      extra={[
      // <Button key="1" type="primary">
      //   <Link to="/">Наглавную</Link>
      // </Button>,
      <Button key="2" type="primary">
      <Link to="/tests">Повернутисьдотестів</Link>
      </Button>
          ]}
      />
       );
```

```
});

export default TestResult;
```