ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ДЕННОЇ ОСВІТИ

ФОРМА НАВЧАННЯ ДЕННА
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Допускається до захисту

Завідувач кафедри _____О.В. Ольховська
(підпис)

«_____»_____2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО ДИПЛОМНОЇ РОБОТИ

на тему

АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ ТРЕНЕЖЕРА «ПЛОЩИНИ ТА ПРЯМІ В ПРОСТОРІ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «АЛГЕБРА ТА ГЕОМЕТРІЯ. ЧАСТИНА 2»

зі спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерні науки»
ступеня магістра

Виконавець роботи Писаренко Ігор Андрійович
_____ «___»_____2021 р.
(підпис)
Науковий керівник доц., канд. фіз.-мат. наук Ольховський Дмитро Миколайович
_____ «___»_____2021 р.
(підпис)

ПОЛТАВА 2021 р.

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

ЗАТВЕРДЖУЮ

Завідувач кафедри _____О.В. Ольховська
(підпис)
«03» вересня 2021 р.

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ ДИПЛОМНОЇ РОБОТИ

Здобувач вищої освіти зі спеціальності 122 «Комп'ютерні науки»
Освітня програма «Комп'ютерні науки»
Прізвище, ім'я, по батькові: Писаренко Ігор Андрійович
1. **Тема «Алгоритмізація та програмування тренажера «Площини та прямі у просторі» дистанційного навчального курсу «Алгебра та геометрія. Частина 2» затверджена наказом ректора № ____від «___» _____ 2021 р.**
Термін подання студентом дипломної роботи «___» _____ 2021 р.
2.Вихідні дані до дипломної роботи курс лекцій з «Алгебри та геометрії. Частина 2», інформаційні джерела.

3. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
ВСТУП
1 ПОСТАНОВКА ЗАДАЧІ
1.1 Постановка задачі розробки навчального тренажера
2 ІНФОРМАЦІЙНИЙ ОГЛЯД
2.1 Огляд робіт, де розглянуто аналогічне до теми роботи завдання
2.2 Позитивні аспекти розглянутих робіт
2.3 Вади розробок розглянутих робіт
2.4 Необхідність та актуальність теми роботи
3 ТЕОРЕТИЧНА ЧАСТИНА
3.1 Алгоритмізація задачі за темою
3.2 Розробка блок-схеми, яка підлягає програмуванню
3.3 Обґрунтування вибору програмних засобів для реалізації завдання роботи
4 ПРАКТИЧНА ЧАСТИНА
4.1 Опис процесу програмної реалізації
4.2 Опис програми
4.3 Перевірка валідності. Дослідження можливостей програмної реалізації
4.4 Необхідна користувачу програми інструкція
ВИСНОВКИ

4. Перелік графічного матеріалу (з точним визначенням кількості блок-схем, іншого графічного матеріалу) 4-6 аркушів блок-схем, ілюстрації за необхідністю

5. Консультанти розділів дипломної роботи

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1. Вступ	Ольховський Д. М.		
2. Постановка задачі	Ольховський Д. М.		
3. Інформаційний огляд	Ольховський Д. М..		
4. Теоретична частина	Ольховський Д. М.		
5. Практична реалізація	Ольховський Д. М.		

6. Календарний графік виконання дипломної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ		
2. Вивчення методичних рекомендацій та стандартів та звіт керівнику		
3. Постановка задачі		
4. Інформаційний огляд джерел бібліотек та інтернету		
5. Теоретична частина		
6. Практична частина		
7. Закінчення оформлення		
8. Доповідь студента на кафедрі		
9. Доробка (за необхідністю), рецензування		

Дата видачі завдання «03» вересня 2021 р.

Здобувач вищої освіти _____
(підпис)

Науковий керівник _____ доц., канд. фіз.-мат. наук Ольховський Д. М.
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Результати захисту дипломної роботи

Дипломна робота оцінена на _____
(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 2021 р.

Секретар ЕК_____ _____
(підпис) (ініціали та прізвище)

Затверджую
Зав. кафедрою _____
к.ф.-м.н. О. В. Ольховська

«____» _____ 2021 р.

Погоджено
Науковий керівник _____
доцент, к.ф.-м.н. Д. М. Ольховский

«____» _____ 2021 р.

План

Дипломної роботи здобувача вищої освіти ступеня магістра
спеціальності 122 Комп'ютерні науки
освітня програма 122 Комп'ютерні науки
Прізвище, ім'я, по батькові Писаренко Ігор Андрійович
На тему «Алгоритмізація та програмування тренажера «Площини та прямі у просторі» дистанційного навчального курсу «Алгебра та геометрія. Частина 2».

ВСТУП
1 ПОСТАНОВКА ЗАДАЧІ
1.1. Постановка задачі розробки навчального тренажера
2. ІНФОРМАЦІЙНИЙ ОГЛЯД
2.1. Огляд робіт, де розглянуто аналогічне до теми роботи завдання
2.2. Позитивні аспекти розглянутих робіт
2.3. Вади розробок розглянутих робіт
2.4. Необхідність та актуальність теми роботи
3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Алгоритмізація задачі за темою
3.2. Розробка блок-схеми, яка підлягає програмуванню
3.3. Обґрунтування вибору програмних засобів для реалізації завдання роботи
4. ПРАКТИЧНА ЧАСТИНА
4.1. Опис процесу програмної реалізації
4.2. Опис програми
4.3. Перевірка валідності. Дослідження можливостей програмної реалізації
4.4. Необхідна користувачу програми інструкція
ВИСНОВКИ

Здобувач вищої освіти _____ П. І. Андрійович
«____» _____ 2021 р.

РЕФЕРАТ

Записка: 53 сторінка в тому числі основної частини 26, блок-схем – 2 сторінка, рисунків – 11 та літературних джерел – 9 назв.

Предмет розробки – програмна реалізація навчального тренажера на мові програмування JavaScript.

Мета роботи – розробити начальний тренажер з теми «Алгоритмізація та програмування тренажера «Площини та прямі у просторі» дистанційного навчального курсу «Алгебра та геометрія. Частина 2».

Методи, які були використанні для розв'язування задачі:

- Теоретичний та практичний матеріал з теми «Площини та прямі у просторі».

- При реалізації використано середовище розробки Microsoft Visual Studio Code та мова програмування JavaScript.

Розроблено алгоритм навчального тренажера з теми «Площини та прямі у просторі».

Здійснена програмна реалізація навчального тренажера з теми «Площини та прямі у просторі».

Результат розробки впроваджено в дистанційний навчальний курс «Алгебра та геометрія. Частина 2».

Ключові слова: ТРЕНАЖЕР, ПЛОЩИНИБ, ПРЯМІ, АЛГЕБРА, ГЕОМЕТРІЯ, ЧАСТИНА 2

ЗМІСТ

ВСТУП

Сучасний світ є залежним від інформаційних технологій, які широко, інтенсивно та ефективно використовуються у всіх сферах людської діяльності. На сьогоднішній момент стала все більш помітна орієнтованість в сторону інформатизації системи освіти.

Як показує практика, позитивні навчання найбільш виразно проявляються при організації в навчальному процесі тренуючих компонентів, автоматизації контролю та самоконтролю.

Тренажер – це комплекс технічних засобів навчання, реалізацією яких є комп'ютерні та фізичні моделі, спеціальні методики забезпечують контроль якості діяльності студента та призначені для формування і вдосконалення у нього навичок та умінь до прийняття якісних та швидких рішень.

Актуальність теми магістерської роботи полягає в тому, щоб створити навчальний тренажер, який має бути використовуватися в навчанні студентів елементам векторної алгебри.

Мета магістерської роботи – розробити начальний тренажер з теми «Алгоритмізація та програмування тренажера «Площини та прямі у просторі» дистанційного навчального курсу «Алгебра та геометрія. Частина 2».

Об'єкт розробки – матеріальне забезпечення Полтавського університету економіки та торгівлі (лекції/практичні) та інтернет ресурси для вивчення математичних дисциплін

Предмет розробки – програмна реалізація навчального тренажера на мові програмування JavaScript.

Для виконання поставленої мети, необхідно виконати наступні завдання:

1) Розробити постановку завдання.
2) Розробити алгоритм навчального тренажера.
3) На основі алгоритму реалізувати можливість створення задач за темою роботи.

4) Застосувати набуті знання та навички в математичній дисципліни та в програмуванні.

5) Впровадити розроблений навчальний тренажер в дистанційний навчальний курс «Алгебра та геометрія. Частина 2».

Методи розробки:

- Теоретичний та практичний матеріал з теми «Алгоритмізація та програмування тренажера «Площини та прямі у просторі» дистанційного навчального курсу «Алгебра та геометрія. Частина 2».

- При реалізації використано середовище розробки Microsoft Visual Studio Code та мова програмування JavaScript.

Практичне значення роботи полягає в тому, що розроблений навчальний тренажер можна впроваджувати в дистанційне та стаціонарне навчання студентів для дистанційного навчального курсу «Алгебра та геометрія. Частина 2» в темі «Площини та прямі у просторі».

Структура пояснювальної записки до магістерської роботи: вступ, постановка задачі, інформаційний огляд, теоретична частина, практична частина, висновки, список інформаційних джерел, додатки.

Обсяг пояснювальної записки: 53 сторінка в тому числі основної частини 26, блок-схем – 2 сторінка, рисунків – 11 та літературних джерел – 9 назв.

1. ПОСТАНОВКА ЗАДАЧІ

1.1 Постановка задачі розробки навчального тренажера

Основним завданням роботи програмна реалізація навчального тренажеру з теми «Алгоритмізація та програмування тренажера «Площини та прямі у просторі» дистанційного навчального курсу «Алгебра та геометрія. Частина 2».

Основні завдання магістерської роботи:

- Описати постановку задачі.
- Розробити алгоритм навчального тренажеру.
- Скласти блок-схему алгоритму.
- Описати процес реалізації основних етапів створення навчального тренажеру.
- Перевірити роботу навчального тренажеру з наведенням прикладів.

Для можливості використовувати навчальний тренажер в ході вивчення дистанційного курсу «Алгебра та геометрія. Частина 2» іноземними студентами слід розробити можливість зміни мови.

Розробити навчальний тренажер з більшої кількістю практичних завдань, аніж теоретичних [3-6].

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Огляд робіт, де розглянуте аналогічне до теми завдання

В ході виконання інформаційного огляду були взяті веб-додатки з інтернету для розгляду було взято веб-додаток **naurok.com.ua** та **zno.osvita.ua**

2.2. Позитивні аспекти в розглянутих роботах

При інформаційному огляді робіт, було виявлено наступні позитивні аспекти в розглянутих навчальних тренажерах:

- Можливість повторного використання роботи з тренажером та його складовими частинами.
- Зручні та інтуїтивно зрозумілі інтерфейси.
- Тренажери дають користувачу перевірити, покращити, а також отримати знання з тем, які є актуальним у вивчені розглянути навчальних дисциплін.
- Допомагають викладачеві швидше та в більшій кількості перевірити знання студентів з тем.

2.3. Негативні аспекти в розглянутих роботах

- Велика кількість тестів що не відносяться до теми
- Інтерфейс навчальних тренажерів реалізований лише на одній мові.
- Не завжди зрозуміло, що вписувати, в якому вигляді має бути відповідь.

2.4. Необхідність та актуальність теми роботи

Програмна реалізація навчального тренажера з теми «Алгоритмізація та програмування тренажера «Площини та прямі у просторі» для дистанційного навчального курсу «Алгебра та геометрія. Частина 2» допоможе студентам при

вивчені даної теми перевірити знання в теоретичних аспектах теми та перевірити теорію на практичних прикладах.

В свою чергу викладачеві навчальний тренажер допоможе зробити аналіз вивчення групою або окремим студентом дану тему.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Опис понять з теми роботи

Основні означення. Стереометрія — це розділ геометрії, у якому вивчаються фігури у просторі.

Основними фігурами у просторі є точка, пряма і площина. Введення нового геометричного образу – площини, потребує розширення системи аксіом, тому ми вводимо групу аксіом С, яка виражає основні властивості площин у просторі.

3.2. Алгоритмізація задачі за темою роботи

При запуску навчального тренажеру студенту може обрати мову інтерфейсу. З’являється початковий екран з усіма створеними викладачем тести. З початкового екрану перейти на сторінку обраного тесту

Основні означення з теми

Крок 1. Оберіть одну з відповідей відповідь:

- A.

- B.

Після відповіді відбувається перехід на головну сторінку з усіма тестами, а пройдений помічається як закінчений, відповідь студенту зберігається у базі даних для перевірки викладачем.

Крок 2. Оберіть одну або декілька відповідей.

- A

- B

- C

Після відповіді відбувається перехід на головну сторінку з усіма тестами, а пройдений помічається як закінчений, відповідь студенту зберігається у базі даних для перевірки викладачем.

Крок 3. Впишіть відповідь

Після відповіді відбувається перехід на головну сторінку з усіма тестами, а пройдений помічається як закінчений, відповідь студенту зберігається у базі даних для перевірки викладачем

3.3. Блок-схема задачі за темою роботи

На рисунках 3.1 зображено розроблена блок-схеми навчального тренажеру з теми «Алгоритмізація та програмування тренажера «Площини та прямі у просторі» дистанційного навчального курсу «Алгебра та геометрія. Частина 2».

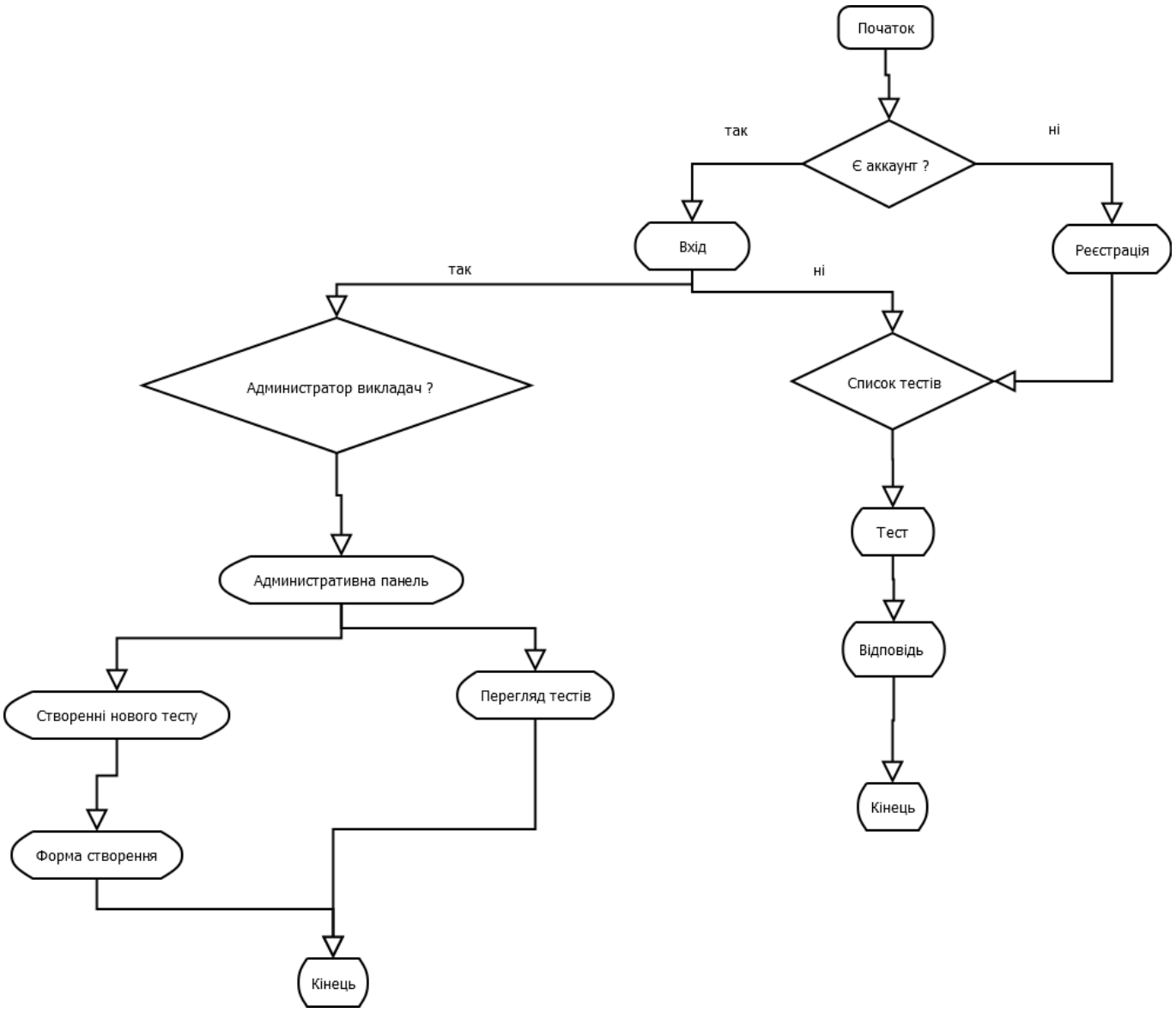

Рисунок 3.1 – Блок-схема навчального тренажера

3.4. Обґрунтування вибору програмних засобів для реалізації завдання роботи

При виборі програмного засобу для реалізації поставленої задачі виникає проблема вибору. Існує велика кількість різних середовищ програмування для ОС Windows.

Для створення даної програми було обрано мову програмування JavaScript, оскільки ця мова має багато переваг, нижче наведені деякі із них:

1. Повністю об’єктно-орієнтована мова, де навіть вбудовані типи представлені класами.
2. Спадкоємець C/C++, що зберігає їх кращі риси.
3. Автоматичне «прибирання сміття».
4. Потужна бібліотека каркасів підтримує зручність побудови різних типів додатків на JavaScript.

Також необхідно обрати середовище для розробки програмного засобу. Серед таких середовищ, як Eclipse, MS Visual Studio MS Visucal Studio Code, IntelliJ IDEA, NetBeans, в межах даної дипломної роботи обрано MS Visual Studio Code, що володіє наступними перевагами [19]:

1) Підтримка популярних мов програмування.
2) Доступність для користувача, так як дане середовище можна вільно знайти та завантажити з мережі Інтернет.
3) Є повністю безкоштовним.
4) Зручний та інтуїтивно зрозумілий інтерфейс.

Для реалізації графічного інтерфейсу необхідно обрати одну з бібліотек: Vuejs, Electronjs Даний інтерфейс дозволяє створювати десктопні програми на мові програмування JavaScript.

Отже, розробка програмної реалізації тренажера здійснюється на мові програмування JavaScript у середовищі візуальної розробки Microsoft Visual Studio Code з використанням бібліотек Vuejs, Electronjs.

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис процесу програмної реалізації

При розробці програмного засобу для навчального тренажеру з теми «Алгоритмізація та програмування тренажера «Площини та прямі у просторі» дистанційного навчального курсу «Алгебра та геометрія. Частина 2» були узяті за основу деякі задачі під які створювався додаток нижче наведені деякі з них:

1. Кутовий коефіцієнт прямої $6x + 3y - 7 = 0$ дорівнює: A) 2; B) -2; C) -6; D) 3;
2. Рівняння прямої, перпендикулярної до прямої, $y = 3x + 2$ має вигляд: A) $3x - y + 7 = 0$; B) $2x - y - 4 = 0$; C) $x + 3 + 8 = 0$;
3. Пара прямих ліній $3x + 4y - 5 = 0$ и $12x - 9y + 11 = 0$: A) паралельні; B) перетинаються; C) взаємно перпендикулярні;
4. Нормальний вектор площини $2x - 5y + 7z + 3 = 0$ має координати: A) $(2, -5.3)$; B) $(2.7, 3)$; C) $(2, -5.7)$;
5. Якщо точка $(3,2,1)$ належить рівнині $5x - 4y + Cz - 2 = 0$ то коефіцієнт C дорівнює: A) -1; B) -5; C) 0; D) 5;

В першу чергу після створення проекту, було добавлено, необхідні посилання на створені викладачем тести та зміну мови.

Таким чином було розроблено вікно з вибором мови (рис. 4.1), початкове вікно (рис. 4.2) та питання для відображення (рис. 4.3 та 4.4).

Після розробки вікон, необхідно розробити функціонал навчального тренажера.

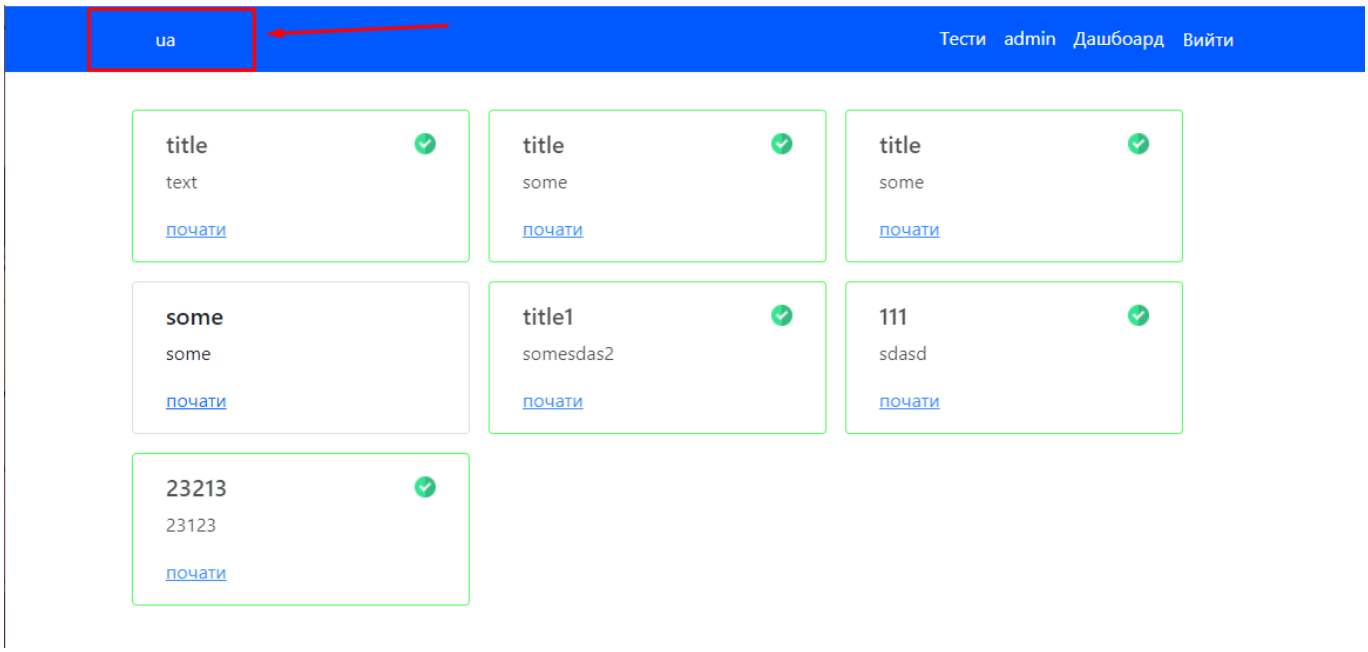

Рисунок 4.1 – Розміщення елементів для вікна з вибором мови

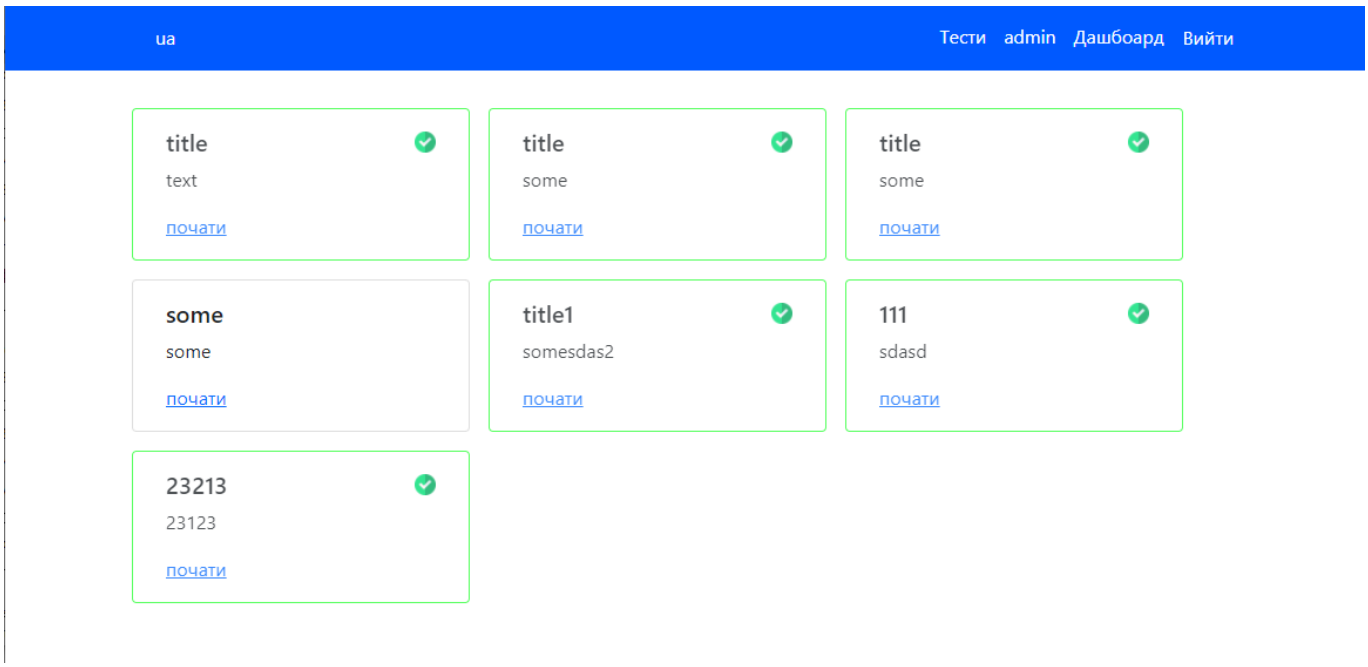

Рисунок 4.2 – Розміщення елементів для початкового вікна

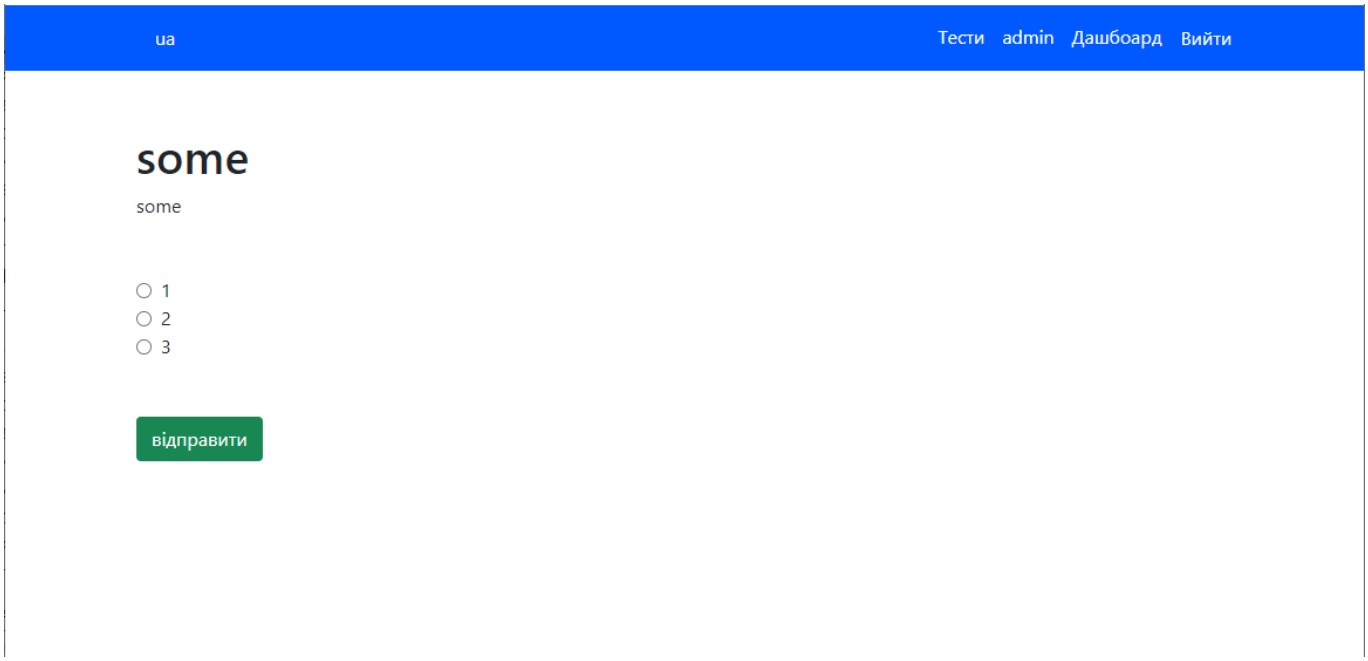

Рисунок 4.3 – Розміщення елементів для відображення питань

Рисунок 4.4 – Розміщення елементів для відображення завдання

На початковому вікні є можливість перейти до одного з двох блоків із завданнями. При натисненні відбувається перехід до першого питання чи завдання.

4.2. Опис програми

Після завантаження з’явиться. На початковому зображені тести створені викладачем и посилання на зміну мови (рис 4.5), також, якщо користувач є викладачем, тоді є посилання на адміністративну панель (рис. 4.6).

Для переходу до завдань з перевірки основних означень з теми, необхідно натиснути на першу кнопку «Почати». Виведеться запитання (рис. 4.7).

На цьому кроці (рис. 4.8) перед студентом з’являється запитання з вибором декількох вірних відповідей. Також є завдання с однією правильною відповіддю (рис 4.9) та с ручним внесенням відповіді (рис 4.10).

Щоб завершити роботу з тренажером необхідну натиснути на крестик у правому верхньому куті програми (рис 4.11), інформацію про пройдені студентом тести буде збережена у базі даних

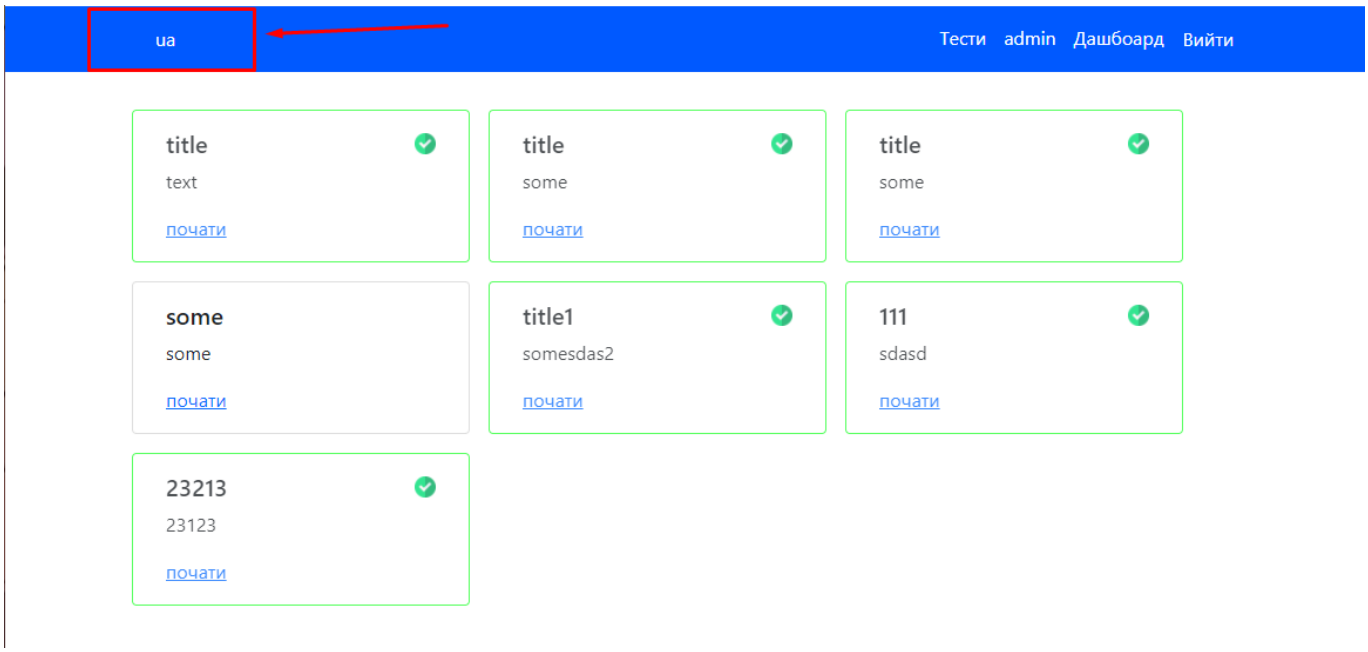

Рисунок 4.5 – Посилання на зміну мови інтерфейсу

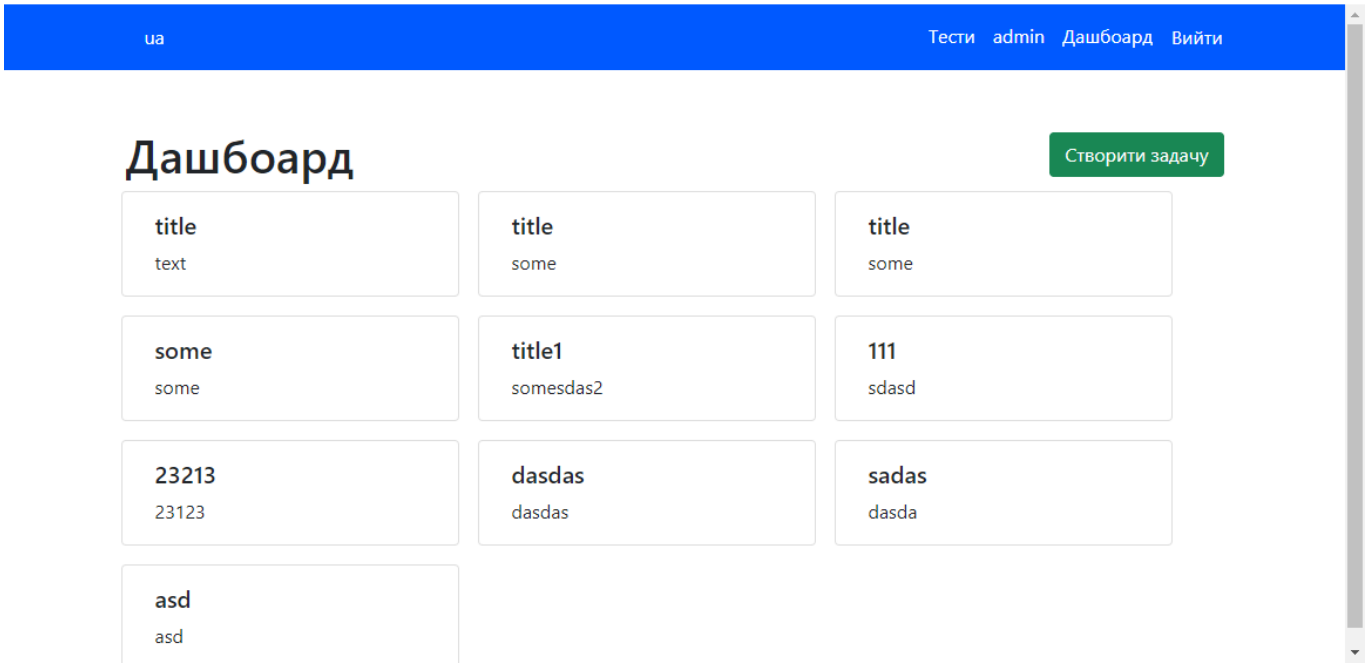

Рисунок 4.6 – Адміністративна панель

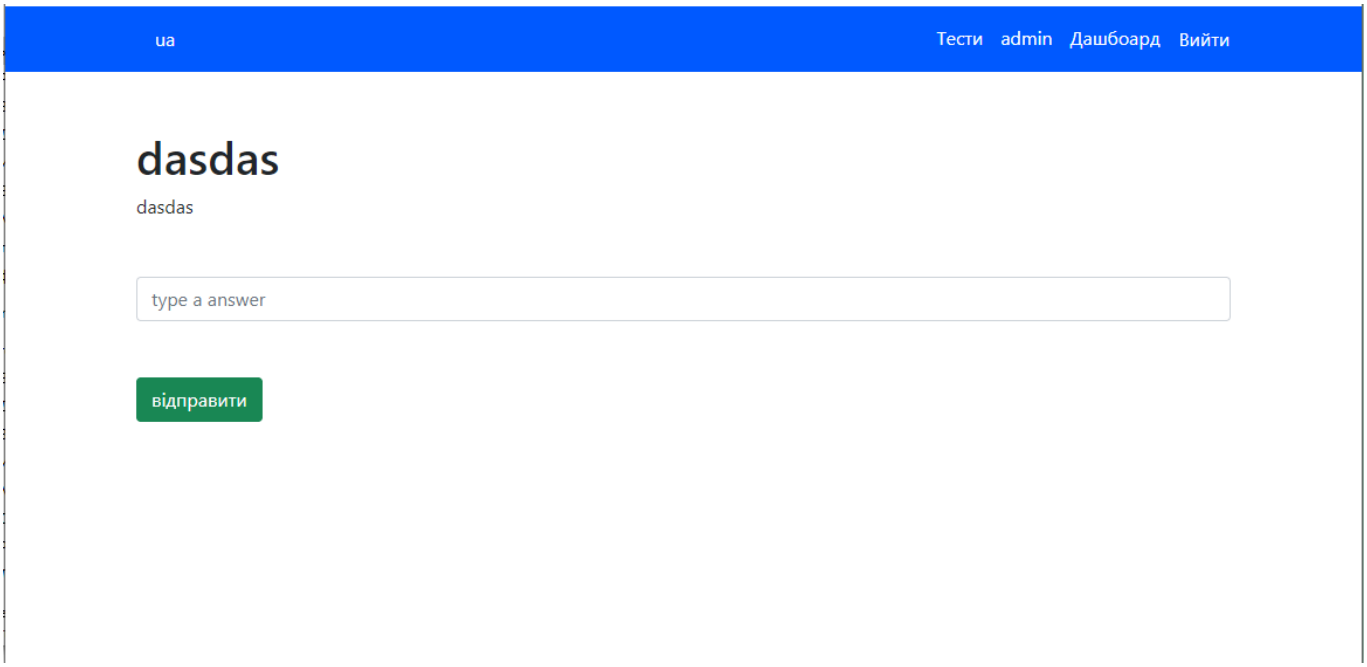

Рисунок 4.7 – Обране студентом запитання с ручним введенням відповіді, створене викладачем в адміністративній панелі

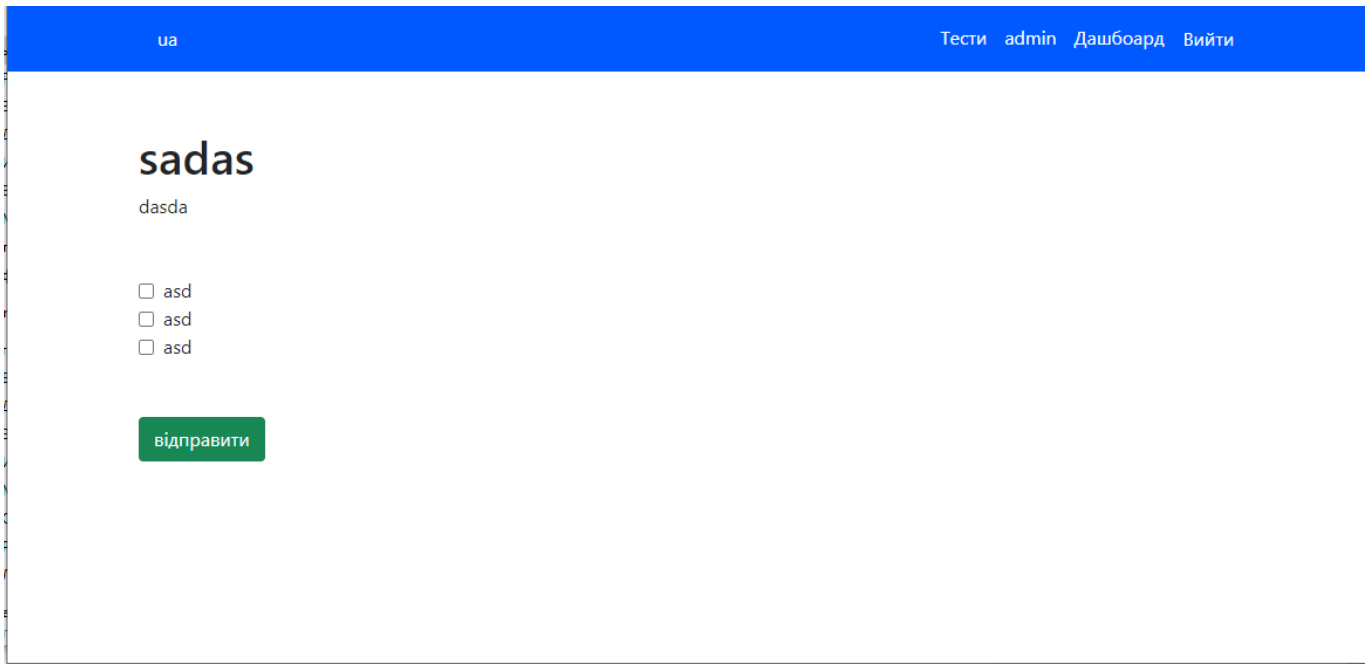

Рисунок 4.8 – Завдання с декількома правильними відповідями

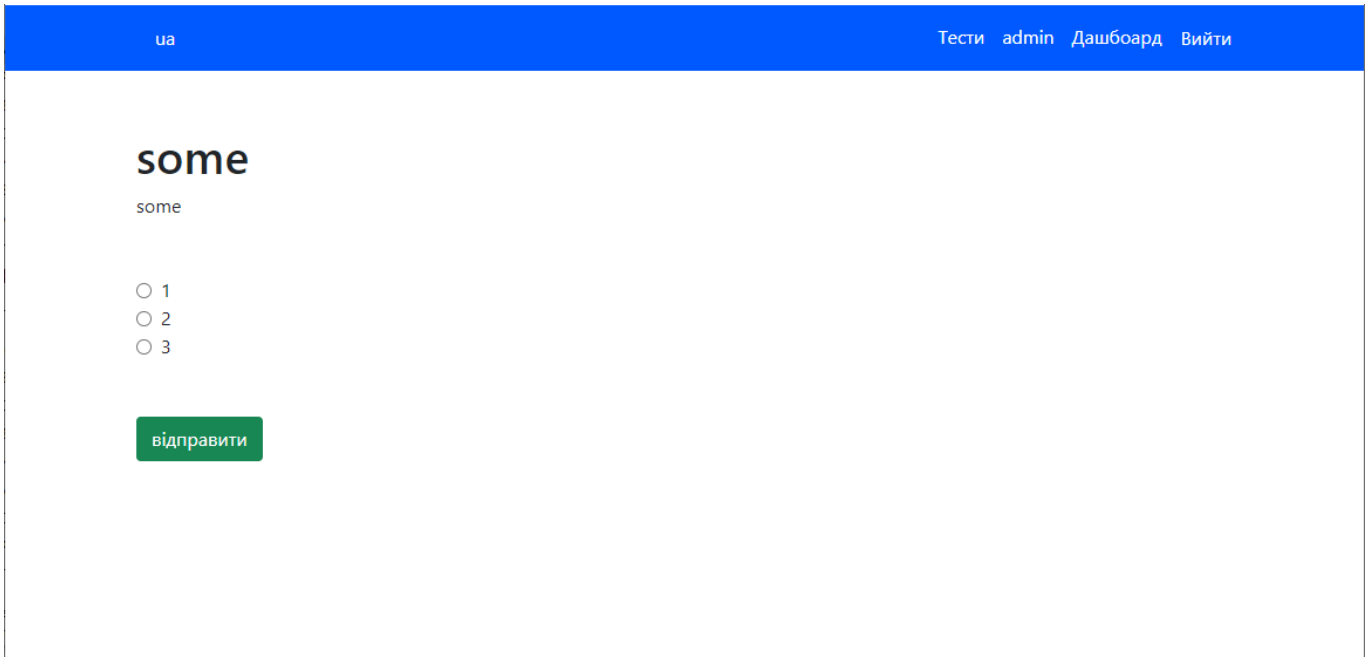

Рисунок 4.9 – Завдання з однією правильною відповіддю

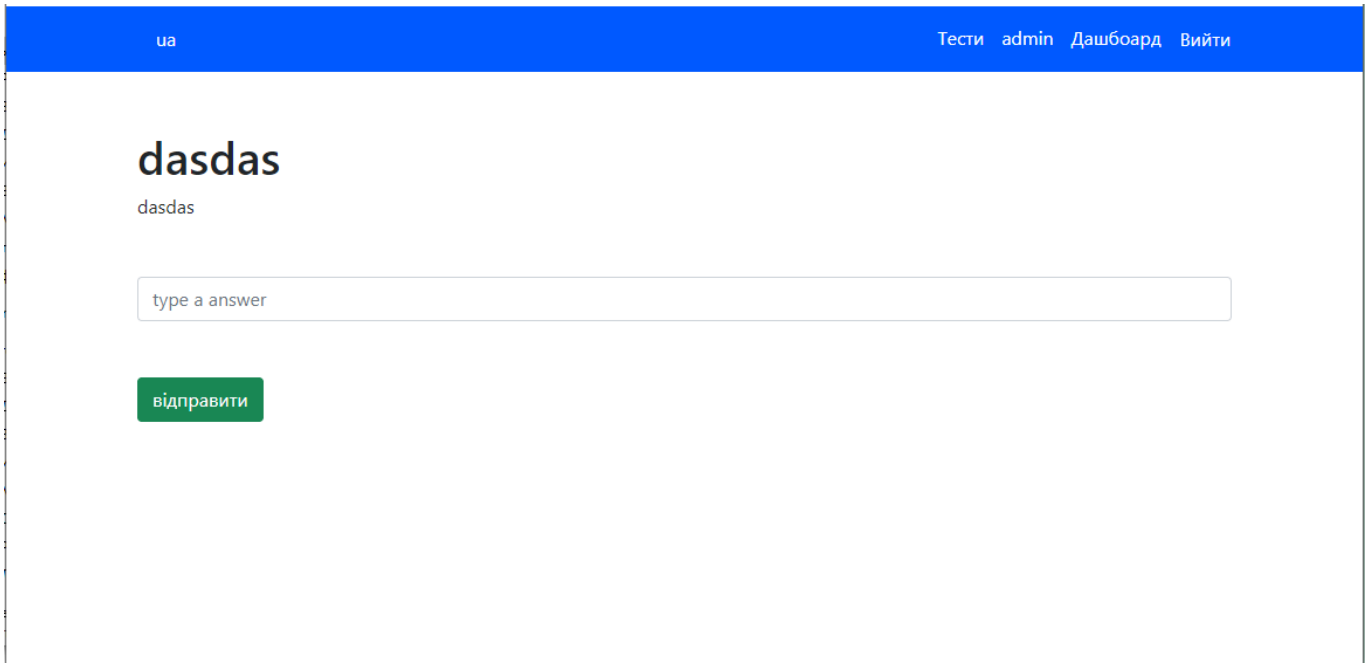

Рисунок 4.10 – Завдання з ручним внесенням відповіді

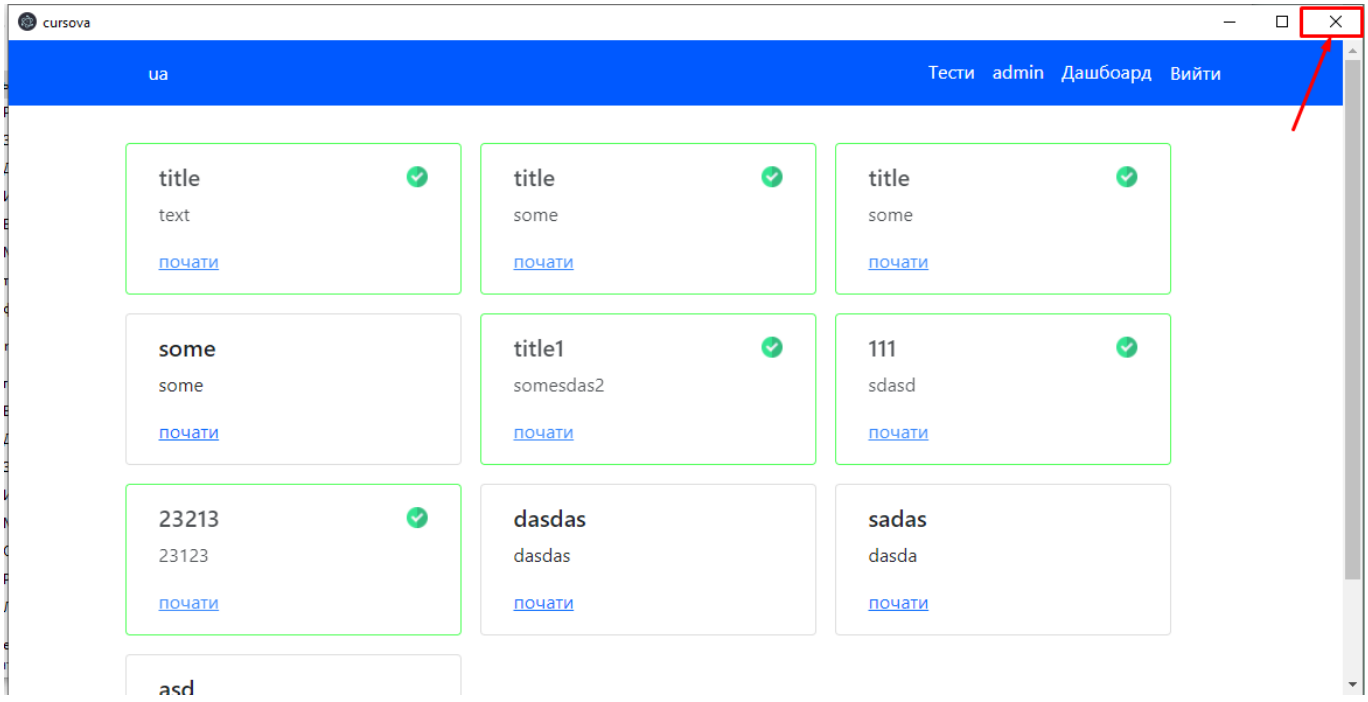

Рисунок 4.11 – Закриття програми

4.3. Перевірка основних кроків навчального тренажера

Після розробки програмного забезпечення наступним завданням є перевірити його працездатність на основних кроках. На кожному кроці були протестовані основні ситуації, які може зробити студент при виконанні.

Після запуску навчального тренажера було протестовані основні кнопки «Почати» та посилання зміни мови інтерфейсу. Також були протестовані форми входу на реєстрації користувача до додатку и форма створення нового тесту викладачем у адміністративній панелі

На наступному кроці було протестоване зберігання створених тестів у базу даних

Далі було протестоване зберігання відповідей студенту на тести створені викладачем

Після перевірки всіх кроків з розділів «Основні означення з теми» та «Практичні завдання» перевірено кнопка «Закрити».

4.4. Необхідна користувачу програми інструкція

Даний навчальний тренажер може використовуватися для студентів спеціальності 122 «Комп’ютерні науки».

Для запуску навчального тренажера, необхідно спочатку завантажити його з дистанційного курсу «Алгебра та геометрія. Частина 2». Після чого перевірити чи встановлено доповнення Nodejs

Коли запуститься навчальний тренажер є можливість обрати мову інтерфейсу, та, якщо користувач є студентом, почати проходити створені викладачем тести або, якщо користувач є викладачем (для цього необхідно зайти в додаток з логіном admin і паролем admin), зайти в адміністративну панель и створювати нові тести

По завершенню натиснути на крестик і завиршити роботу з навчальним тренажером.

ВИСНОВКИ

При виконані дипломної роботи виконано основне завдання, а саме: програмна реалізація навчального тренажеру з теми «Алгоритмізація та програмування тренажера «Площини та прямі у просторі» дистанційного навчального курсу «Алгебра та геометрія. Частина 2».

Виконано наступні основні завдання роботи:

- Описано постановку задачі.
- Розроблено алгоритм навчального тренажеру.
- Складено блок-схему алгоритму.
- Описано процес реалізації основних етапів створення навчального тренажеру.
- Перевірено роботу навчального тренажеру з наведенням прикладів.

Для можливості використовувати навчальний тренажер в ході вивчення дистанційного курсу «Алгебра та геометрія. Частина 2» іноземними студентами розроблено можливість зміни мови.

Розроблено навчальний тренажер з прикладами по практичним завданням

Результати дипломної роботи були представлені на науково-практичному семінарі «Комп’ютерні науки і прикладна математика». Опубліковано наукову статтю в Полтавському університеті економіки та торгівлі.

СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Интернет тренажеры в сфере образования. [Електронний ресурс]: Режим доступу – http://icmim.sfu-kras.ru/sites/icm.institute.sfu-kras.ru/files/Rukovodstvo_polzovatelya_Internet-trenazhery.pdf

2. Словари и энциклопедии на Академии [Електронний ресурс]: Режим доступу – https://psychology.aca-demic.ru/2691/тренажер

3. Аксіоми стереометрії. [Електронний ресурс]: Режим доступу – https://fizma.net/index.php?idi=geo/prostir

4. Інформатика та системні науки (ІСН – 2017) : матеріал VIII Всеукраїнської науково-практичної конференції за міжнародною участю (м. Полтава, 16-18 березня 2017 р) / за ред. Ємця О. О. – Полтава : ПУЕТ, 2017. – 333 с. Режим доступу: http://dspace.puet.edu.ua/handle/123456789/5798

5. Комп'ютерні науки і прикладна математика (КНіПМ – 2018): матеріали науково-практичного семінару. Випуск 1 / за ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2018. – 64 с. Режим доступу: http://dspace.puet.edu.ua/handle/123456789/6563

6. Комп'ютерні науки і прикладна математика (КНіПМ – 2018): матеріали науково-практичного семінару. Випуск 2 / за ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2018. – 64 с. Режим доступу: http://dspace.puet.edu.ua/handle/123456789/6987

7. Комп'ютерні науки і прикладна математика (КНіПМ – 2019): матеріали науково-практичного семінару. Випуск 3 / за ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2019. – 83 с. Режим доступу: http://dspace.puet.edu.ua/handle/123456789/7048

8. Комп'ютерні науки і прикладна математика (КНіПМ – 2021): матеріали науково-практичного семінару. Випуск 6 / за ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2021. – 106 с. Режим доступу: http://dspace.puet.edu.ua/handle/123456789/11355

9. Ємець О. О. Методичні рекомендації до виконання дипломної роботи для студентів ступеня магістра спеціальності 122 «Комп’ютерні науки» / О. О. Ємець, Т. О. Парфьонова – Полтава: РВВ ПУЕТ, 2018. – 35с.

ДОДАТОК А. КОД ПРОГРАМИ

```
import axios from "axios";

const api = axios.create({
  baseURL: process.env.VUE_APP_API_URL,
});

export default api;
<template>
  <form class="form" @submit.prevent="sumbitHandler">
    <div class="form-group">
      <label for="login">{{ $t("loginTitle") }}</label>
      <input
        type="text"
        class="form-control"
        id="login"
        placeholder="Enter login"
        v-model.trim="login"
      />
    </div>
    <div class="form-group">
      <label for="password">{{ $t("password") }}</label>
      <input
        type="password"
        class="form-control"
        id="password"
        :placeholder="$t('password')"
        v-model.trim="password"
      />
    </div>
    <button
      :disabled="isLoadig || isFilled"
      class="btn btn-success mt-4"
      type="submit"
    >
      {{ isLogin ? $t("login") : $t("register") }}
```

```
    </button>
    <p v-if="isFail" class="error mt-5">{{ $t("error") }}</p>
  </form>
</template>

<script>
export default {
  name: "AuthForm",
  data() {
    return {
      login: "",
      password: "",
      isLoadig: false,
    };
  },
  props: {
    isLogin: {
      type: Boolean,
      default: true,
    },
  },
  computed: {
    isFail() {
      return this.$store.state.auth.fail;
    },
    isFilled() {
      return !this.login || !this.password;
    },
  },
  methods: {
    async sumbitHandler() {
      if (!this.login || !this.password) {
        return;
      }
      this.isLoadig = true;
      const formData = {
```

```
      login: this.login,
      password: this.password,
    };
    if (this.isLogin) {
      await this.$store.dispatch("login", formData);
    } else {
      await this.$store.dispatch("register", formData);
    }
    this.login = "";
    this.password = "";
    this.isLoadig = false;
    if (!this.isFail) {
      this.$router.push("/");
    }
   },
  },
};
</script>

<style lang="scss" scoped>
button {
  width: 100%;
}
.form-group {
  &:first-child {
    margin-bottom: 1rem;
  }
}
.error {
  background-color: rgba(212, 41, 41, 0.5);
  padding: 0.5rem;
  border: 1px solid rgb(212, 41, 41);
  border-radius: 0.75rem;
}
</style>
<template>
```

```
  <div class="card" :class="{ checked: isChecked && !dashboard }">
    <div class="card-body">
      <h5 class="card-title">
        <span>{{ task.title }}</span
        ><img src="@/assets/check.png" v-if="isChecked && !dashboard" />
      </h5>
      <p class="card-text">
        {{ task.text }}
      </p>
      <router-link v-if="!dashboard" :to="{ path: `/task/${task.id}` }">{{
        $t("start")
      }}</router-link>
    </div>
  </div>
</template>

<script>
export default {
  name: "Card",
  props: {
    dashboard: {
      type: Boolean,
      default: false,
    },
    task: {
      type: Object,
      require,
    },
  },
  computed: {
    isChecked() {
      if (this.user.question.find((el) => el.id === this.task.id)) {
        return true;
      }
      return false;
    },
```

```
    user() {
      return this.$store.state.auth.userInfo;
    },
  },
};
</script>

<style lang="scss" scoped>
.card {
  width: 18rem;
  margin: 0.5rem;
}
.checked {
  pointer-events: none;
  opacity: 0.8;
  border-color: rgb(47, 255, 57);
}
.card-title {
  display: flex;
  justify-content: space-between;
  align-items: center;
  img {
    max-width: 100%;
    widows: 1.125rem;
    height: 1.125rem;
  }
}
</style>
<template>
  <div class="mt-3">
    <div class="your-answer" v-if="currentOption === 'one'">
      <input
        class="form-control mt-3"
        type="text"
        placeholder="Введіть правильну відповідь УКР"
        v-model="ua"
```

```
    />
    <input
      class="form-control mt-3"
      type="text"
      placeholder="Type correct answer EN"
      v-model="en"
    />
   </div>
   <div v-if="currentOption === 'two'">
    <button class="btn btn-primary" type="button" @click="addOption">
      {{ $t("add") }}
    </button>
    <input
      v-for="n in checkboxes"
      :key="`checkboxes-${n}`"
      type="text"
      class="mb-2 mt-2 form-control checkbox"
      :placeholder="$t('typeAnswer')"
    />
   </div>
   <div v-if="currentOption === 'three'">
    <button class="btn btn-primary" type="button" @click="addOption">
      {{ $t("add") }}
    </button>
    <input
      v-for="n in checkboxes"
      :key="`radio-${n}`"
      type="text"
      class="mb-2 mt-2 form-control radio"
      :placeholder="$t('typeAnswer')"
    />
   </div>
  </div>
</template>

<script>
```

```
export default {
  name: "CreateOptions",
  props: {
    currentOption: {
      default: null,
    },
  },
  data() {
    return {
      checkboxes: 0,
      ua: "",
      en: "",
    };
  },
  watch: {
    currentOption() {
      this.checkboxes = 0;
    },
  },
  methods: {
    addOption() {
      this.checkboxes++;
    },
  },
};
</script>

<style lang="scss" scoped></style>
<template>
  <header>
    <div class="container">
      <button class="btn my" @click="setLocale">{{ $i18n.locale }}</button>
      <div class="nav">
        <router-link to="/">{{ $t("tests") }}</router-link>
        <router-link :to="{ path: info ? '/' : '/login' }">{{
          info ? info.login : $t("login")
```

```
        }}</router-link>
        <router-link to="/dashboard" v-if="info && info.role === 'admin'">{{
          $t("dash")
        }}</router-link>
        <button class="btn my" @click="logout" v-if="info">
          {{ $t("logout") }}
        </button>
      </div>
    </div>
  </header>
</template>

<script>
export default {
  name: "Header",
  computed: {
    info() {
      return this.$store.state.auth.userInfo;
    },
  },
  methods: {
    logout() {
      this.$store.dispatch("logout");
      this.$router.push("/login");
    },
    setLocale() {
      if (this.$i18n.locale === "ua") {
        this.$i18n.locale = "en";
      } else {
        this.$i18n.locale = "ua";
      }
    },
  },
};
</script>
```

```
<style lang="scss" scoped>
header {
  background-color: rgb(0, 89, 255);
  padding: 0.938rem;
  .container {
    display: flex;
    justify-content: space-between;
  }
  a {
    display: block;
    text-decoration: none;
    color: #fff;
    &:first-child {
      margin-right: 0.938rem;
    }
    &:nth-child(2) {
      margin-right: 0.938rem;
    }
  }
  .my {
    padding: 0;
    color: #fff;
    margin-left: 0.938rem;
  }
}
</style>
import VueI18n from "vue-i18n";
import Vue from "vue";

Vue.use(VueI18n);

export const i18n = new VueI18n({
  locale: "ua",
  fallbackLocale: "en",
  messages: {
    en: {
```

```
  loginTitle: "Login",
  password: "password",
  login: "login",
  register: "register",
  error: "Something went wrong",
  start: "start",
  typeAnswer: "type text",
  add: "add option",
  tests: "Tests",
  dash: "Dashboard",
  logout: "logout",
  ttitle: "Task title",
  ttext: "Task text",
  chose: "chose type of answer",
  ya: "your answer",
  wo: "with options",
  on: "one correct answer",
  create: "Create task",
  acc: "dont have a account",
  acc2: "already has an account ?",
  submit: "submit",
 },
 ua: {
  loginTitle: "Логін",
  password: "пароль",
  login: "логін",
  register: "реєстрація",
  error: "Щось пішло не так",
  start: "почати",
  typeAnswer: "Уведіть текст",
  add: "добавити варіант",
  tests: "Тести",
  dash: "Дашбоард",
  logout: "Вийти",
  ttitle: "Заголовок задачі",
  ttext: "Текст задачі",
```

```
      chose: "Оберіть тип відповіді",
      ya: "своя відповідь",
      wo: "з декількома відповідями",
      on: "одна правильна відповідь",
      create: "Створити задачу",
      acc: "Не маєте аккаунту ?",
      acc2: "Вже маєте аккаунт ?",
      submit: "відправити",
    },
  },
});
import Vue from "vue";
import VueRouter from "vue-router";
import Home from "../views/Home.vue";
import Login from "../views/Login.vue";
import Register from "../views/Register.vue";
import Dashboard from "../views/Dashboard.vue";
import Article from "../views/Article.vue";
import CreateTask from "../views/CreateTask.vue";
import Task from "../views/Task.vue";
import store from "../store";

Vue.use(VueRouter);

const routes = [
  {
    path: "/",
    name: "Home",
    component: Home,
    meta: { requiresAuth: true },
  },
  {
    path: "/login",
    name: "Login",
    component: Login,
  },
```

```
  {
    path: "/register",
    name: "Register",
    component: Register,
  },
  {
    path: "/dashboard",
    name: "Dashboard",
    component: Dashboard,
    meta: { requiresAuth: true },
  },
  {
    path: "/dashboard/:id",
    component: Article,
    meta: { requiresAuth: true },
  },
  {
    path: "/create",
    component: CreateTask,
    meta: { requiresAuth: true },
  },
  {
    path: "/task/:id",
    component: Task,
    meta: { requiresAuth: true },
  },
];

const router = new VueRouter({
  mode: "history",
  routes,
});

const isAuthenticated = () => {
  return store.state.auth.userInfo;
};
```

```
router.beforeEach((to, from, next) => {
  if (to.matched.some((route) => route.meta?.requiresAuth)) {
    if (isAuthenticated()) {
      next();
    } else {
      next("/login");
    }
  } else {
    next();
  }
});

export default router;
import api from "@/api";

const auth = {
  state: {
    userInfo: null,
    fail: false,
  },
  mutations: {
    SET_USER_INFO(state, data) {
      state.userInfo = data;
    },
    SET_FAIL(state, data) {
      state.fail = data;
    },
  },
  actions: {
    async answer({ state, commit }, data) {
      try {
        await api.patch(`/users/${data.user}`, {
          question: [
            ...state.userInfo.question,
            { id: data.id, answer: data.answer },
```

```
      ],
    });
    const user = await api.get(
      `/users?login=${state.userInfo.login}&password=${state.userInfo.password}`
    );
    commit("SET_USER_INFO", user.data[0]);
  } catch (err) {
    throw new Error(err);
  }
},
async register({ commit }, data) {
  try {
    const isExist = await api.get(
      `/users?login=${data.login}&password=${data.password}`
    );
    if (isExist.data.length) {
      commit("SET_FAIL", true);
      return;
    } else {
      commit("SET_FAIL", false);
      data.question = [];
      const res = await api.post("/users", data);
      commit("SET_USER_INFO", res.data);
    }
  } catch (err) {
    throw new Error(err);
  }
},
async login({ commit }, data) {
  try {
    const res = await api.get(
      `/users?login=${data.login}&password=${data.password}`
    );
    if (res.data.length) {
      commit("SET_FAIL", false);
      commit("SET_USER_INFO", res.data[0]);
```

```
      } else {
        commit("SET_FAIL", true);
      }
    } catch (err) {
      throw new Error(err);
    }
  },
  logout({ commit }) {
    commit("SET_USER_INFO", null);
  },
 },
};

export default auth;
import Vue from "vue";
import Vuex from "vuex";
import createPersistedState from "vuex-persistedstate";

import api from "@/api";
import auth from "./auth";

Vue.use(Vuex);

export default new Vuex.Store({
 plugins: [
  createPersistedState({
   paths: ["auth"],
  }),
 ],
 state: {
  tasks: null,
  task: null,
 },
 mutations: {
  SET_TASKS(state, data) {
   state.tasks = data;
```

```
  },
  SET_TASK(state, data) {
    state.task = data;
  },
 },
 actions: {
  async getTasks({ commit }) {
   try {
    const res = await api.get("/tasks");
    const { data } = res;
    commit("SET_TASKS", data);
   } catch (err) {
    throw new Error(err);
   }
  },
  async createTask(_, data) {
   try {
    await api.post("/tasks", data);
   } catch (err) {
    throw new Error(err);
   }
  },
  async getTaskById({ commit }, id) {
   try {
    const res = await api.get(`/tasks/${id}`);
    commit("SET_TASK", res.data);
   } catch (err) {
    throw new Error(err);
   }
  },
 },
 modules: { auth },
});
<template>
 <div class="container mt-5">
  <div class="title">
```

```
      <h1>{{ $t("dash") }}</h1>
      <router-link class="btn btn-success" to="/create">{{
        $t("create")
      }}</router-link>
    </div>
    <div class="row">
      <Card
        :dashboard="true"
        v-for="task in tasks"
        :key="task.id"
        :task="task"
      />
    </div>
  </div>
</template>

<script>
import Card from "../components/Card.vue";
export default {
  components: { Card },
  name: "Dashboard",
  computed: {
    tasks() {
      return this.$store.state.tasks;
    },
  },
};
</script>

<style lang="scss" scoped>
.title {
  display: flex;
  justify-content: space-between;
  align-items: center;
  h1 {
    margin-bottom: 0;
```

```
  }
}
</style>
<template>
  <div class="home">
    <div class="container mt-4">
      <div class="row">
        <Card v-for="task in tasks" :task="task" :key="task.id" />
      </div>
    </div>
  </div>
</template>

<script>
import Card from "@/components/Card.vue";

export default {
  name: "Home",
  components: { Card },
  data() {
    return {};
  },
  computed: {
    tasks() {
      return this.$store.state.tasks;
    },
  },
  async mounted() {
    await this.$store.dispatch("getTasks");
  },
};
</script>

<style lang="scss" scoped></style>
import Vue from "vue";
import App from "./App.vue";
```

```
import { i18n } from "./plugins/i18n";
import router from "./router";
import store from "./store";
import FlagIcon from "vue-flag-icon";

import "./styles/style.scss";

Vue.config.productionTip = false;
Vue.use(FlagIcon);

new Vue({
  router,
  store,
  i18n,
  render: (h) => h(App),
}).$mount("#app");
<template>
  <div class="container mt-5">some2</div>
</template>

<script>
export default {
  name: "Article",
};
</script>

<style lang="scss" scoped></style>
<template>
  <form @submit.prevent="create">
    <div class="container mt-5">
      <div class="form-group">
        <label for="title">{{ $t("ttitle") }}</label>
        <input
          type="text"
          v-model="title"
          class="form-control"
```

```
    id="title"
    :placeholder="$t('ttitle')"
   />
  </div>
  <div class="form-group">
   <label for="text">{{ $t("ttext") }}</label>
   <textarea
    name="text"
    class="form-control"
    id="text"
    cols="30"
    rows="10"
    v-model="text"
   ></textarea>
  </div>
  <div class="checboxes">
   <p class="mt-5">{{ $t("chose") }}</p>
   <div class="z">
    <label for="html">{{ $t("ya") }}</label>
    <input
     v-model="currentOption"
     type="radio"
     name="fav_language"
     id="html"
     value="one"
    />
   </div>
   <div class="z">
    <label for="css">{{ $t("wo") }}</label>
    <input
     v-model="currentOption"
     type="radio"
     name="fav_language"
     id="css"
     value="two"
    />
```

```
      </div>
      <div class="z">
       <label for="js">{{ $t("on") }}</label>
       <input
        v-model="currentOption"
        type="radio"
        name="fav_language"
        id="js"
        value="three"
       />
      </div>
      <CreateOptions ref="create" :currentOption="currentOption" />
    </div>
    <button :disabled="isLoad" type="submit" class="s mt-5 btn btn-success">
     {{ $t("create") }}
    </button>
   </div>
  </form>
</template>

<script>
import CreateOptions from "@/components/CreateOptions.vue";

export default {
  name: "CreateTask",
  components: { CreateOptions },
  data() {
   return {
    currentOption: null,
    isLoad: false,
    title: "",
    text: "",
   };
  },
  methods: {
   async create() {
```

```
this.isLoad = true;
const data = {};
if (this.currentOption === "one") {
  data.type = "typing";
  data.rightUnswer = [this.$refs.create.en, this.$refs.create.ua];
} else if (this.currentOption === "two") {
  data.type = "checkboxes";
  data.questions = [];
  for (
    let i = 0;
    i < this.$refs.create.$el.querySelectorAll(".checkbox").length;
    i++
  ) {
    data.questions.push(
      this.$refs.create.$el.querySelectorAll(".checkbox")[i].value
    );
  }
} else if (this.currentOption === "three") {
  data.type = "radios";
  data.questions = [];
  for (
    let i = 0;
    i < this.$refs.create.$el.querySelectorAll(".radio").length;
    i++
  ) {
    data.questions.push(
      this.$refs.create.$el.querySelectorAll(".radio")[i].value
    );
  }
}
data.title = this.title;
data.text = this.text;
this.text = "";
this.title = "";
this.currentOption = null;
await this.$store.dispatch("createTask", data);
```

```
      this.isLoad = false;
    },
  },
};
</script>

<style lang="scss" scoped>
form {
  display: flex;
  flex-direction: column;
}
.z {
  display: flex;
  align-items: center;
  label {
    margin-right: 0.5rem;
  }
}
.s {
  width: 100%;
}
</style>
<template>
  <div class="container mt-5">
    <div class="title">
      <h1>{{ $t("dash") }}</h1>
      <router-link class="btn btn-success" to="/create">{{
        $t("create")
      }}</router-link>
    </div>
    <div class="row">
      <Card
        :dashboard="true"
        v-for="task in tasks"
        :key="task.id"
        :task="task"
```

```
    />
   </div>
  </div>
</template>

<script>
import Card from "../components/Card.vue";
export default {
  components: { Card },
  name: "Dashboard",
  computed: {
    tasks() {
      return this.$store.state.tasks;
    },
  },
};
</script>

<style lang="scss" scoped>
.title {
  display: flex;
  justify-content: space-between;
  align-items: center;
  h1 {
    margin-bottom: 0;
  }
}
</style>
<template>
  <div class="home">
    <div class="container mt-4">
      <div class="row">
        <Card v-for="task in tasks" :task="task" :key="task.id" />
      </div>
    </div>
  </div>
```

```
</template>

<script>
import Card from "@/components/Card.vue";

export default {
  name: "Home",
  components: { Card },
  data() {
    return {};
  },
  computed: {
    tasks() {
      return this.$store.state.tasks;
    },
  },
  async mounted() {
    await this.$store.dispatch("getTasks");
  },
};
</script>

<style lang="scss" scoped></style>
<template>
  <div class="container mt-5">
    <div class="some">
      <p>{{ $t("acc") }}</p>
      <router-link to="/register">{{ $t("register") }}</router-link>
    </div>
    <AuthForm />
  </div>
</template>

<script>
import AuthForm from "@/components/AuthForm.vue";
```

```
export default {
  name: "Login",
  components: { AuthForm },
};
</script>

<style lang="scss" scoped>
.some {
  display: flex;
  justify-content: center;
  align-items: center;
  p,
  a {
    margin-bottom: 0;
    padding: 0;
  }
  a {
    margin-left: 0.5rem;
  }
}
</style>
"use strict";

import { app, protocol, BrowserWindow } from "electron";
import { createProtocol } from "vue-cli-plugin-electron-builder/lib";
import installExtension, { VUEJS_DEVTOOLS } from "electron-devtools-installer";
const isDevelopment = process.env.NODE_ENV !== "production";

// Scheme must be registered before the app is ready
protocol.registerSchemesAsPrivileged([
  { scheme: "app", privileges: { secure: true, standard: true } },
]);

async function createWindow() {
  // Create the browser window.
  const win = new BrowserWindow({
```

```
    width: 800,
    height: 600,
    autoHideMenuBar: true,
    webPreferences: {
      // Use pluginOptions.nodeIntegration, leave this alone
      //  See  nklayman.github.io/vue-cli-plugin-electron-builder/guide/security.html#node-integration  for more info
      nodeIntegration: process.env.ELECTRON_NODE_INTEGRATION,
      contextIsolation: !process.env.ELECTRON_NODE_INTEGRATION,
    },
  });

  if (process.env.WEBPACK_DEV_SERVER_URL) {
    // Load the url of the dev server if in development mode
    await win.loadURL(process.env.WEBPACK_DEV_SERVER_URL);
    if (!process.env.IS_TEST) win.webContents.openDevTools();
  } else {
    createProtocol("app");
    // Load the index.html when not in development
    win.loadURL("app://./index.html");
  }
}

// Quit when all windows are closed.
app.on("window-all-closed", () => {
  // On macOS it is common for applications and their menu bar
  // to stay active until the user quits explicitly with Cmd + Q
  if (process.platform !== "darwin") {
    app.quit();
  }
});

app.on("activate", () => {
  // On macOS it's common to re-create a window in the app when the
  // dock icon is clicked and there are no other windows open.
  if (BrowserWindow.getAllWindows().length === 0) createWindow();
```

```
});

// This method will be called when Electron has finished
// initialization and is ready to create browser windows.
// Some APIs can only be used after this event occurs.
app.on("ready", async () => {
  if (isDevelopment && !process.env.IS_TEST) {
    // Install Vue Devtools
    try {
      await installExtension(VUEJS_DEVTOOLS);
    } catch (e) {
      console.error("Vue Devtools failed to install:", e.toString());
    }
  }
  createWindow();
});

// Exit cleanly on request from parent process in development mode.
if (isDevelopment) {
  if (process.platform === "win32") {
    process.on("message", (data) => {
      if (data === "graceful-exit") {
        app.quit();
      }
    });
  } else {
    process.on("SIGTERM", () => {
      app.quit();
    });
  }
}
```