МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ВЕБ-ІНТЕРФЕЙСУ ІНТЕРАКТИВНИХ МОДУЛІВ ДЛЯ ВИКОРИСТАННЯ У ОСВІТНЬОМУ ПРОЦЕСІ»**

Виконав: студент 2 курсу, групи 8.1221
спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми комп'ютерні науки
(назва освітньої програми)
Р.І. Руденко
(ініціали та прізвище)

Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н. Решевська К.С
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри інформаційних технологій в туризмі НУЗП, доцент, к.т.н. Морозов Д.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя
2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 комп’ютерні науки

(шифр і назва)

Освітня програма комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри комп’ютерних наук, д.т.н., професор

_____ Чопоров С.В.

(підпис)

« ____ » _____ 2022 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Руденко Роману Івановичу

(прізвище, ім’я та по батькові)

1. Тема роботи Розробка веб-інтерфейсу інтерактивних модулів для використання у освітньому процесі

керівник роботи Решевська Катерина Сергіївна, к. т. н., доцент

(прізвище, ім’я та по батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 05 » 05 2022 року № 500-с

2. Строк подання студентом роботи

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

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

1. Вивчення теоретичних аспектів, огляд технологій для використання інтерактивних завдань
2. Проектування та розробка веб-сервісу
3. Реалізація веб-інтерфейсу інтерактивних модулів

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

Презентація

6. Консультанти розділів роботи

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

7. Дата видачі завдання 05.05.2022

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	02.09.2022	
2.	Дослідження предметної області.	07.09.2022	
3.	Огляд існуючих веб-сервісі інтерактивних	12.09.2022	
	завдань.		
4.	Проектування моделі веб-сервісу	15.09.2022	
	інтерактивних модулів.		
5.	Вибір засобів реалізації веб-сервісу	21.09.2022	
6.	Проектування інтерфейсу програмного	28.09.2022	
	продукту		
7.	Розробка веб-сервісу інтерактивних	02.10.2022	
	модулів		
8.	Тестування та розробка інструкції з	29.10.2022	
	експлуатації		
9.	Оформлення та нормоконтроль	21.11.2022	
	кваліфікаційної роботи.		
10.	Захист кваліфікаційної роботи.	13.12.2022	

Студент _____ Р.І. Руденко
(підпис) (ініціали та прізвище)

Керівник роботи _____ К.С. Решевська
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

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

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка веб-інтерфейсу інтерактивних модулів для використання у освітньому процесі»: 61 с., 16 рис., 3 табл., 22 джерела, 7 додатків.

ВЕБ-СТОРІНКА, ВЕБ-ІНТЕРФЕЙС, ІНТЕРАКТИВНІ ЗАВДАННЯ, НАВЧАЛЬНА ОНЛАЙН-ПЛАТФОРМА, CSS, HTML, JAVASCRIPT, SQL, NODE.JS.

Об’єкт дослідження – інтерактивні завдання в процесі навчання.

Мета роботи – розробка веб-інтерфейсу інтерактивних модулів, які можна використовувати під час освітнього процесу.

Методи дослідження – аналіз, синтез, порівняння, експеримент.

Результати дослідження – розроблена система інтерактивних модулів, представлена у вигляді веб-сайту.

У дипломному проекті розглянуто теоретичні аспекти використання веб-інтерфейсів навчальних модулів, проаналізовано сучасні веб-сайти, призначені для створення інтерактивних завдань. Реалізований програмний продукт, у вигляді веб-сайту, може бути застосований на практичних заняттях під час освітнього процесу.

SUMMARY

Master`s qualifying paper «Development of web-interface of interactive modules for use in the educational process»: 61 pages, 16 figures, 3 tables, 22 references, 7 supplements.

WEB PAGE, WEB INTERFACE, INTERACTIVE TASKS, ONLINE LEARNING PLATFORM, CSS, HTML, JAVASCRIPT, SQL, NODE.JS.

Object of the study – interactive tasks in the learning process.

Aim of the study – development of web-interface of interactive modules for use in the educational process.

Methods of research – analysis, synthesis, modeling, experiment.

Results of research – developed a system of interactive modules, presented in the form of a website.

The diploma project considers the theoretical aspects of using web-interfaces of training modules, analyzes modern websites designed to create interactive tasks. The implemented software product, in the form of a website, can be used in practical classes during the educational process.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – База даних

ІКТ – Інформаційно-комунікаційні технології

ІС – Інформаційна система

СУБД – Система управління базами даних

ACID – Atomicity, consistency, isolation, durability

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

IT – Information Technology

JSON – JavaScript Object Notation

ORM – Object-Relational Mapping

SQL – Structured Query Language

UML – Unified Modelling Language

ВСТУП

Останнім часом з розвитком технологій, поширенням і доступністю Інтернет-зв'язку виявляються унікальні можливості. Інтернет - це не тільки необмежена скарбниця інформації, але і агрегат активної інтелектуальної діяльності прогресивного учня, який дає змогу отримувати знання, покращувати уміння та навички.

Використання сучасних мережевих технологій дає змогу суттєво покращити систему освіти, тому подальша її інформалізація є незворотним і обов’язковим процесом. У цьому випадку освітні веб-ресурси є найкращим інструментом для покращення навчального процесу.

Веб-ресурси вже зараз є одним із пріоритетних інструментів і методів самоосвіти, вони є ефективними організаційними елементами системи освіти, оскільки нові інформаційні технології впливають на всі складові: зміст навчання, методи та організаційні форми.

Особливої популярності зараз набувають веб-сайти та додатки, які дозволяють покращити та урізноманітнити контроль знань учнів. Поєднання традиційних методів перевірки та оцінювання знань з новими технологіями надає вчителям широкі можливості. Наприклад, веб-ресурси, які дозволяють створювати інтерактивні завдання, є дуже актуальними, адже допомагають викладачам створювати завдання, які будуть викликати більший інтерес учнів, ніж традиційні методи контролю. Зазвичай, такі веб-сервіси мають вбудований конструктор, який дозволяє викладачу створювати різні завдання, а учням виконувати завдання та отримувати результат.

Використання веб-ресурсів з можливістю створення завдань стає дедалі популярнішою, особливо під дистанційної форми навчання. Тому розробка веб-сервісу з можливістю створення інтерактивних завдань використання у освітньому процесі є актуальною задачею.

Мета дослідження – розробка веб-інтерфейсу інтерактивних модулів

Об’єкт дослідження – інтерактивні завдання в процесі навчання.

Предмет дослідження – засоби реалізації веб-інтерфейсу інтерактивних завдань.

Наукова новизна дослідження полягає у розробці вільного веб-інтерфейсу інтерактивних модулів.

Практична застосовність виявляється у можливому впровадженню веб-інтерфейсу інтерактивних модулів до роботи після розгортання сервісу на хостингу.

Для виконання поставлених задач використовувалися електронні та друковані ресурси. Розробка веб-інтерфейсу відбувалась за допомогою мови програмування JavaScript.

Структурно дипломний проект складається з трьох розділів. В першому розділі проводиться дослідження предметної області та аналіз схожих за функціоналом веб-сервісів. Другий розділ присвячено проектуванню системи веб-сайту. В третьому проведено розробку та тестування веб-інтерфейсу.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття, призначення та класифікація інтерактивних завдань

Основні методичні інновації пов'язані із застосуванням інтерактивних методів навчання. Один із таких методів – це інтерактивні завдання.

Інтерактивне завдання – це один із варіантів контролю, який використовується з метою засвоєння вивченого матеріалу. Використання сучасних мультимедійних та інтерактивних технологій у навчальному процесі дозволяє підвищити наочність та сприйняття навчального матеріалу, що позитивно відбивається на навчальній мотивації та ефективності навчання.

Сучасні технології збагачують процес навчання, дозволяючи здійснювати взаємодію між учнем та комп’ютером, планшетом і смартфоном. Така взаємодія забезпечує формування та прояв ключових компетенцій, до яких належить насамперед інформаційна.

Застосування інтерактивних завдань під час навчального процесу дозволяє:

– індивідуалізувати навчальний процес, пристосувати його до особистісних особливостей та потреб учнів;

– організувати навчальний матеріал з урахуванням різноманітних способів навчальної діяльності;

– посилити візуальне сприйняття та полегшити засвоєння навчального матеріалу;

– активізувати пізнавальну діяльність учнів (використання елементів анімації, комп'ютерного конструювання дозволяє отримати як знання так і початкові навчальні навички щодо конкретного предмета).

Класифікують три основні групи інтерактивних завдань, які відрізняються методами введення відповіді:

– вказівні – відповідь вводиться шляхом вказівки елемента (групи елементів);

– маніпулятивні – відповідь вводиться шляхом переміщення елемента (групи елементів);

– клавіатурні – відповідь вводиться шляхом формування нового контенту.

Розглянемо ці класифікації більш детально. К вказівним типам відносять завдання:

– на вибір одного варіанта відповіді;

– на вибір кількох варіантів відповіді;

– на вибір (вказівку) елемента на малюнку;

– на вибір елемента зі списку;

Наступним типом є маніпулятивні, до яких відносять завдання:

– на встановлення відповідності;

– на впорядкування (сортування) об'єктів;

– на розподіл за групами (класифікацію) об'єктів;

– на встановлення графічних зв’язків;

– на переміщення об'єктів малюнку (складання композицій).

За клавіатурним типом розрізняють так завдання:

– на введення числа (у т. ч. з контролем точності);

– на введення рядка;

– на введення математичної формули.

Іноді також виділяють змішаний тип, у якому використовують комбінування варіантів завдань.

Спираючись на таку різноманітність варіантів інтерактивних завдань можна зрозуміти, що вони доволі популярні. Адже науковий аналіз, проведений вченим США, Едґардом Дейлом у 60-х роках ХХ ст. засвідчує, що інтерактивне навчання та завдання вможливлює різке збільшення відсотка засвоєння матеріалу.

1.2 Постановка задачі

Дослідивши предметну область та аспекти використання інтерактивних завдань в освітньому процесі можемо сформулювати задачі для нашої роботи:

– проаналізувати схожі існуючи веб-ресурси;

– сформулювати перелік вимоги до сучасного веб-сервісу;

– розробити формальну модель та структурну специфікацію предметної області;

– дослідити існуючи засоби реалізації веб-інтерфейсу та обґрунтувати доцільність вибору інструментів;

– виконати програмну реалізацію веб-інтерфейсу за допомогою обраних засобів;

– протестувати розроблений веб-інтерфейс;

– написання інструкції роботи з веб-інтерфейсом.

1.3 Огляд існуючих веб-інтерфейсів інтерактивних модулів

У сучасному світі вже створено ряд систем, які призначені для створення та виконання інтерактивних завдань. Для виявлення їх переваг та недоліків ми розглянемо та виконаємо порівняльний аналіз найбільш популярних веб-сайтів, а саме «Kahoot!» [1], «LearningApps» [2], «Online Test Pad» [3] та «Quizizz» [4]. Результати дослідження відображено у таблиці 1.1.

Виконавши порівняльний аналіз обраних веб-інтерфейсів інтерактивних модулів, можна зробити висновок, що всі вони мають свої переваги та недоліки. Але основне, що слід зазначити, що ці системи не є досконалими та мають проблеми, які можуть викликати складнощі під час їх використовування.

Таблиця 1.1 – Порівняльна характеристика веб-інтерфейсів навчальних модулів

Назва	Kahoot!	LearningApps	Online Test Pad	Quizizz
Загальний опис	Онлайн сервіс для створення інтерактивних завдань	Сервіс для створення інтерактивних модулів	Онлайн сервіс для створення сучасних завдань	Онлайн сервіс для створення вікторин та флеш-карток
Мова інтерфейсу	Англійська	Підтримує всі мови	Англійська, (поганий переклад на українську)	Англійська
Варіанти завдань	Quiz, тест (правда або брехня), опитування, відкриті відповіді	Велика кількість інтерактивних завдань(21)	Тести, опитування, кросворди, діалогові тренажери	Вікторина та флеш-картки
Інтерфейс	Сучасний та зрозумілий інтерфейс, усі елементи розташовані зручно	Інтерфейс зрозумілий, але застарілий, елементи розташовані зручно, але кегель шрифту замалий	Сучасний та зрозумілий інтерфейс, усі елементи розташовані зручно	Сучасний та зрозумілий інтерфейс, усі елементи розташовані зручно
Адаптивність	Адаптивність під усі пристрої	Адаптивність відсутня	Адаптивність під усі пристрої	Адаптивність під усі пристрої
Ліцензія	Умовно-безкоштовна (більше завдань з платним тарифом)	Безкоштовна	Безкоштовна	Безкоштовна

2 ПРОЕКТУВАННЯ МОДЕЛІ ВЕБ-СЕРВІСУ ІНТЕРАКТИВНИХ МОДУЛІВ

2.1 Вимоги до веб-сервісі інтерактивних модулів

Провівши порівняльний аналіз існуючих веб-інтерфейсів інтерактивних модулів, визначивши їх переваги та недоліки, можна скласти перелік основних вимог.

Психолого-педагогічні вимоги:

– забезпечення систематичності в створенні модулів інтерактивних завдань

– адаптація інтерактивних завдань до віку потенційних користувачів;

– найбільш правильна візуалізація матеріалу, для завдання.

Функціональні вимоги:

– забезпечення сучасного та інтуїтивно зрозумілого інтерфейсу;

– організація інтерактивної взаємодії між користувачем та веб-сервісом;

– забезпечення реакції на незаплановану поведінку з боку користувачів.

Технічні вимоги:

– оптимізація системи та мінімізація витрат часу для завантаження сторінок та виконання операцій на платформі;

– забезпечення стабільної роботи сервісу;

– запобігання несанкціонованих дій.

Естетичні вимоги:

– вибір красивої палітри кольорів;

– використання якісних та правильно підібраних зображень та графічних елементів.

Вимоги до технічної документації та інструкції з експлуатації:

– покрокова, зрозуміла та чітка інструкція щодо користування веб-інтерфейсом інтерактивних модулів

– детальний опису усіх функцій та можливостей сервісу.

Під час проектування та розробки веб-інтерфейсу інтерактивних модулів необхідно враховувати всі вищезазначені вимоги для подальшого його використання у освітньому процесі.

2.2 Специфікація та концептуальне проектування предметної області

При побудові сучасного програмного продукту однією з невід’ємних частин є етап моделювання організаційної структури додатку. Для вирішення цієї задачі необхідно створити діаграми, використовуючи UML. UML – це мова, яка надає загальний словниковий запас об'єктно-орієнтованих термінів та методи побудови діаграм, які достатньо багаті, щоб змоделювати будь-який проект розробки системи від аналізу до реалізації [5]. Мова UML надає достатню кількість типів діаграм. Для побудування організаційної структури взаємодії користувача з системою використаємо діаграму прецедентів.

Діаграми прецедентів (англ. use case diagram) відображає елементи моделі варіантів використання. На діаграмі графічно відображають відношення між акторами(учасник) та прецедентами(система, підсистема або клас) в системі.

В процесі створення Use Case діаграми нашої системи була складена схема взаємодії системи з трьома типами користувачів:

– гість;

– вчитель;

– учень.

Під гостем нашої веб-сервісу будемо розуміти незареєстрованного та неавторизованого користувача. Гість може:

а) зайти на головну сторінку та подивитись інформацію про сервіс;

б) зареєструватися;

На рисунку 2.1 представлена отримана діаграма.

Рисунок 2.1 – Use Case діаграма гостя (неавторизованого користувача)

Вчитель під час взаємодії з системою може:

а) авторизуватися;

б) керувати курсом:

1) створити курс;

2) видалити курс.

в) керувати дидактичним матеріалом:

1) додати дидактичний матеріал

2) видалити дидактичний матеріал

г) керувати інтерактивним завданням:

1) створити інтерактивне завдання;

2) видалити інтерактивне завдання.

Проілюструємо на рисунку 2.2 отриману діаграму.

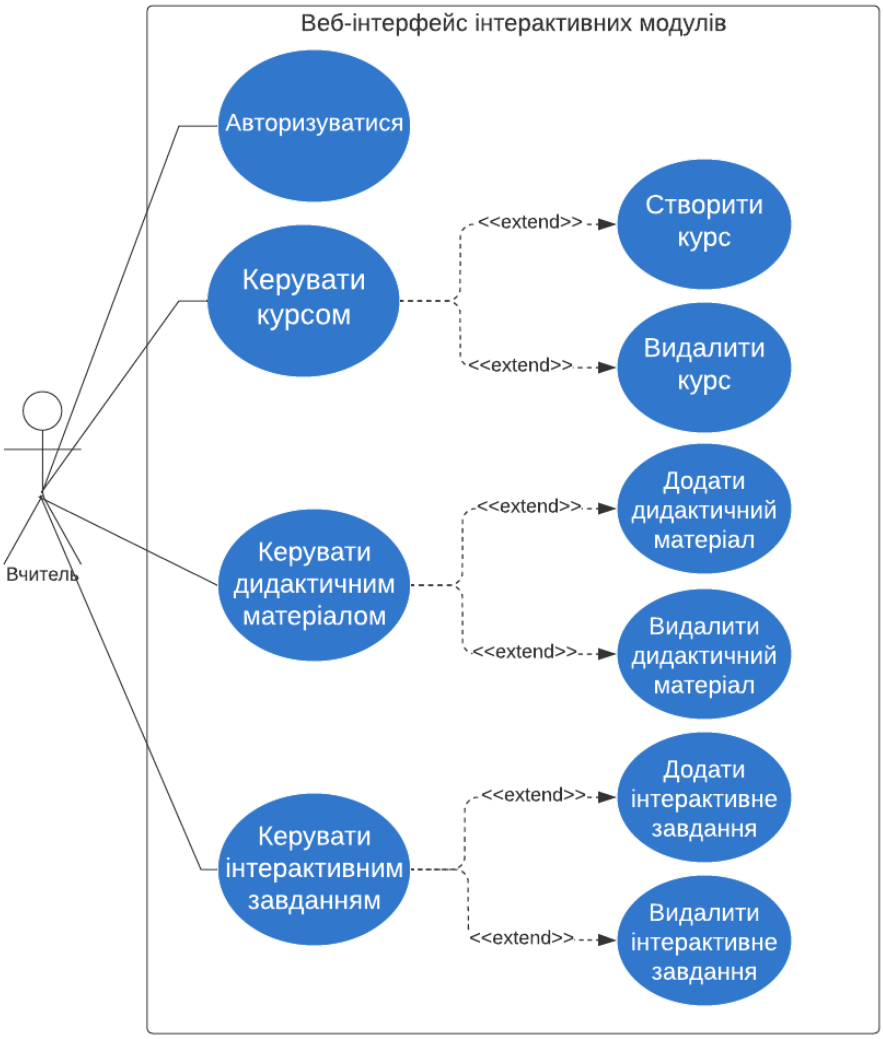


Рисунок 2.2 – Use Case діаграма вчителя

Під час користування системою у учня є можливість:

а) авторизуватися;

б) обирати курс

в) проходити навчання за курсом, а саме:

1) вивчати дидактичний матеріал курсу;

2) виконувати інтерактивні завдання курсу

На схемі, зображеній на рисунку 2.3, проілюструємо отриману діаграму

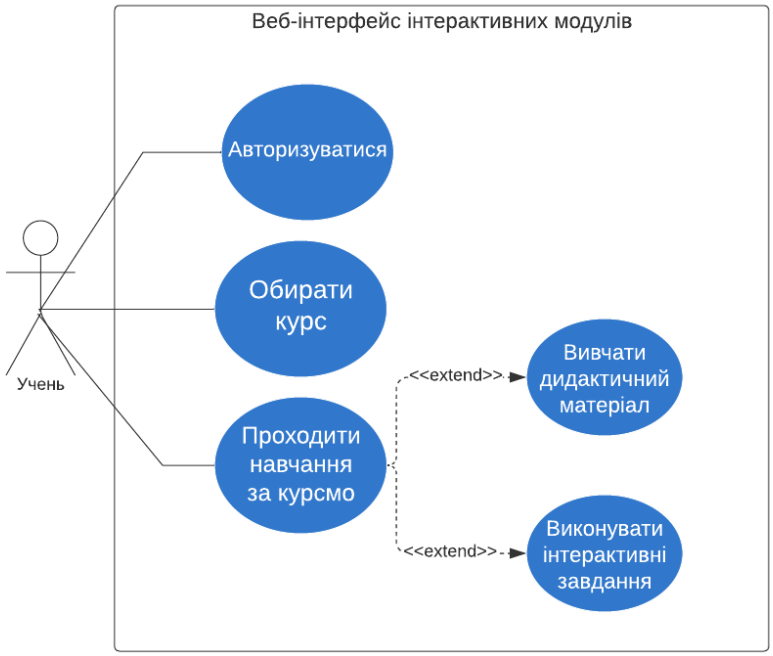


Рисунок 2.3 – Use Case діаграма учня

Врахувавши всі можливі взаємодії всіх користувачів із системою, можемо отримати повну Use Case діаграму, яка представлена в додатку А.

Далі необхідно розробити діаграму послідовності (англ. sequence diagram), на якій продемонструємо життєвий цикл певних об’єктів і взаємодію акторів (дійових осіб) ІС в рамках певного прецеденту.

В першу чергу, зобразимо на рисунку 2.4 діаграму послідовності, що відображатиме процес реєстрації нового користувача.

Ця послідовність починається з того, що гість вводить свої персональні дані в «Форму реєстрації», далі дані передаються на сервер, тобто взаємодія системи з об’єктом «Контролер реєстрації» та збереження їх в базі даних. Наприкінці, гість отримує повідомлення про реєстрацію.

На діаграмі, зображеній на рисунку 2.5, відображена взаємодія вчителя з системою, а саме послідовність дій, які проходить вчитель від авторизації до додавання дидактичного матеріалу та створення інтерактивного завдання.

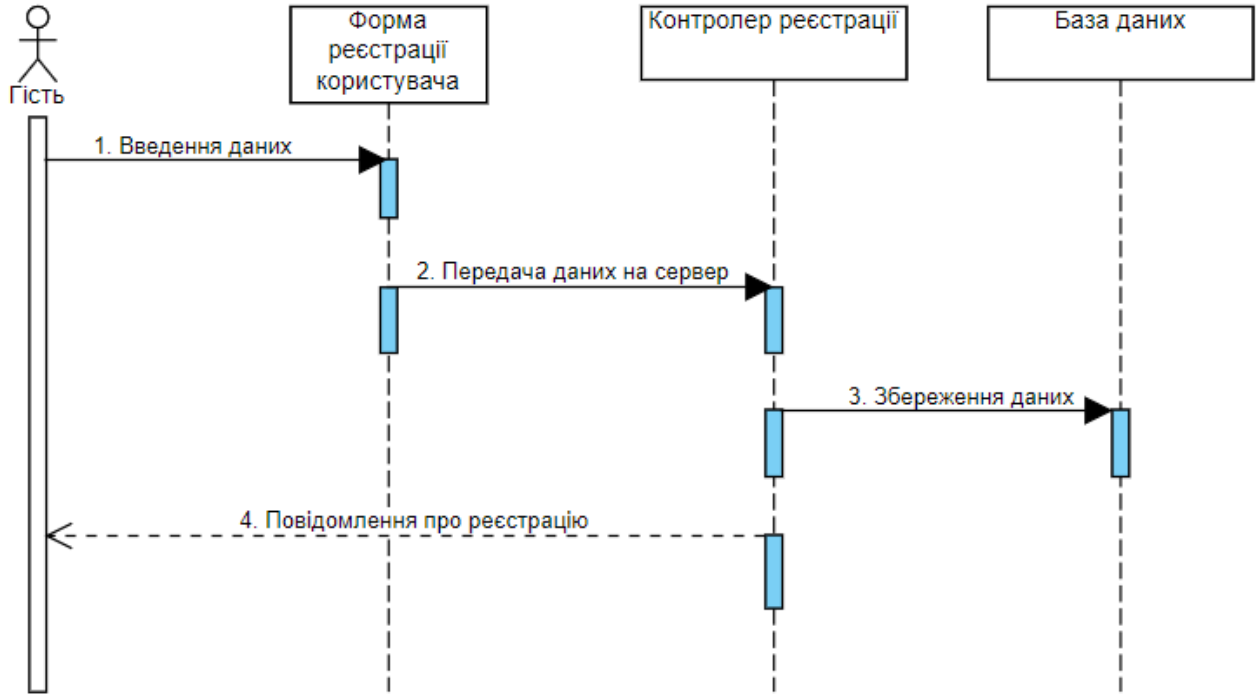


Рисунок 2.4 – Діаграма послідовності реєстрації нового користувача

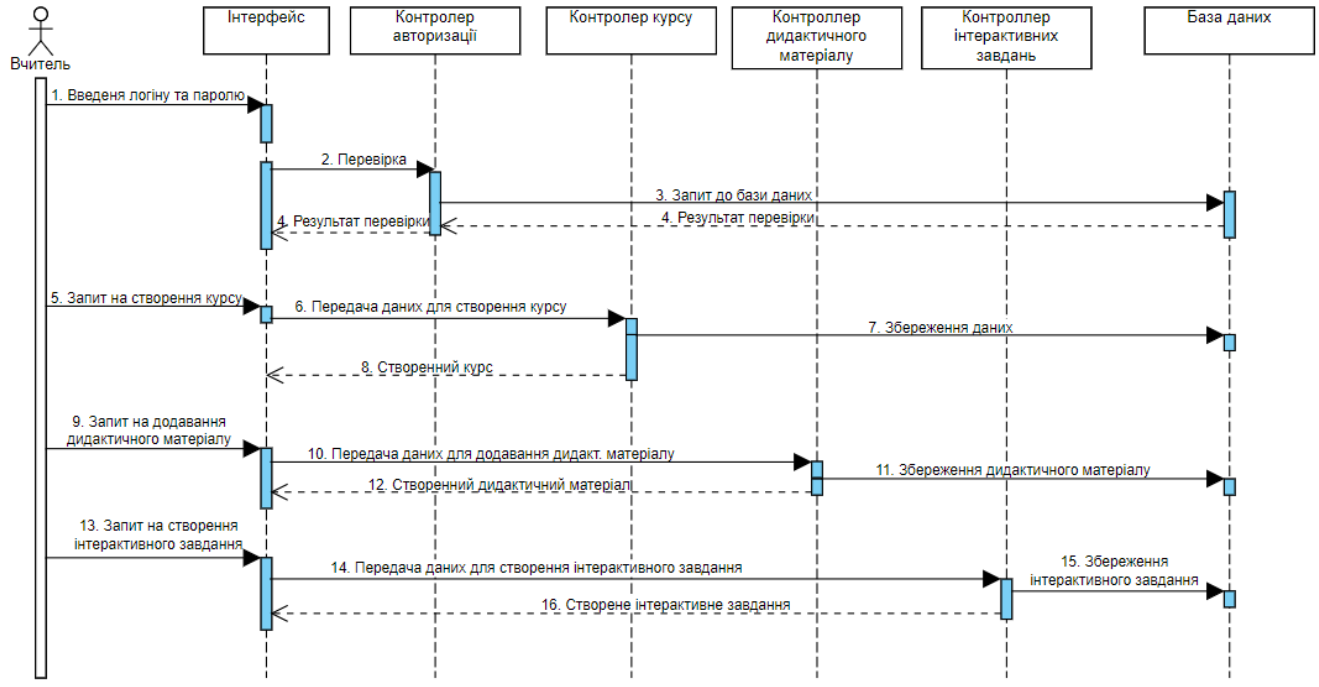


Рисунок 2.5 – Діаграма послідовності взаємодії вчителя з системою

Дана послідовність починається з того, що актор «Вчитель» вводить свої дані для авторизації та отримання доступу до веб-сервісу. Введені дані передаються на сервер, та звіряються контролером авторизації з даними, які збережені в базі, після чого користувач отримує результат перевірки. Наступним процесом в діаграмі – це створення курсу, вчитель відправляє дані, які необхідні для створення курсу та отримує створений курс. І останні процеси додавання дидактичного матеріалу та створення інтерактивного завдання схожі. Вчитель заповнює дані та відправляє запит на сервер, після збереження даних у базі, отримує створений дидактичний матеріал або інтерактивне завдання.

На наступній діаграмі, яка зображена на рисунку 2.6, відображено взаємодію учня з системою.

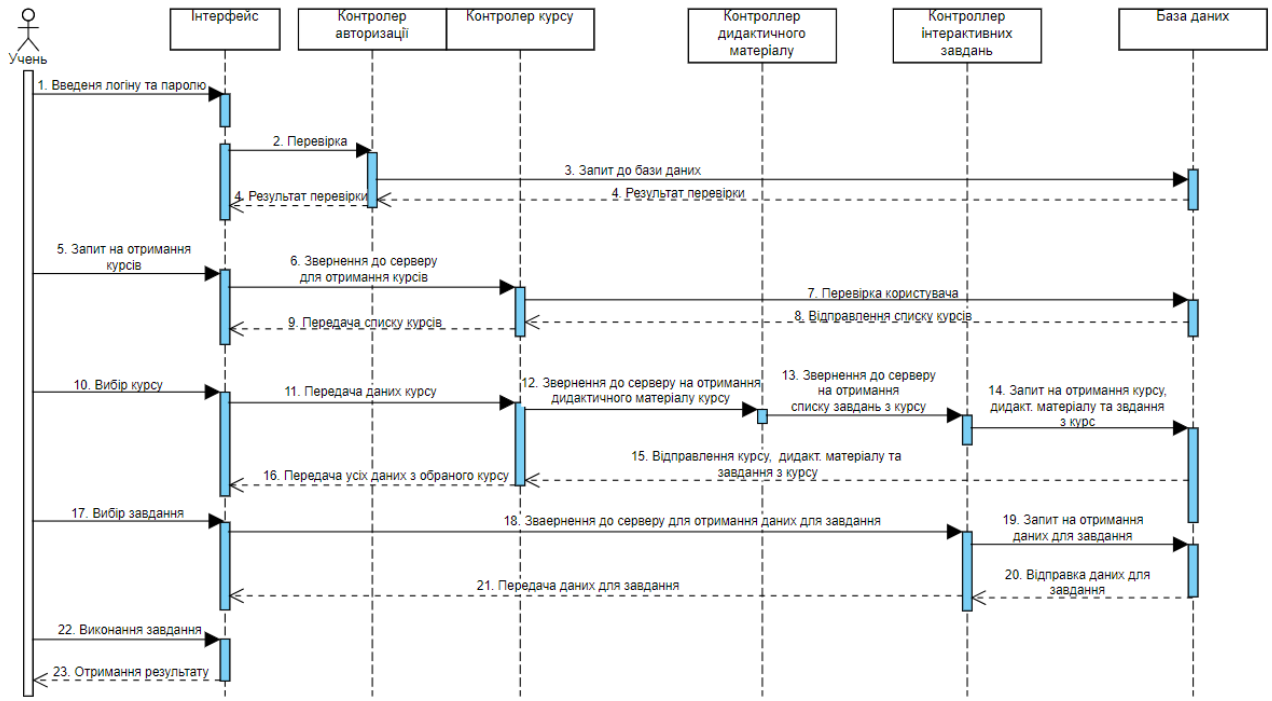


Рисунок 2.6 – Діаграма послідовності взаємодії учня з системою

Представлена послідовність починається з того, що актор «Учень» вводить логін та пароль для авторизації та отримання доступу до веб-сервісу. Введені дані передаються на сервер, та звіряються контролером авторизації з даними, які

збережені в базі, після чого користувач отримує результат перевірки. Наступний процес, що розглядається на діаграмі є безпосереднє навчання. Учень відправляє на сервер запит на перегляд курсів, отримавши відповідь з бази даних може обрати курс. Після вибору курсу відбувається запит на пошук курсу, дидактичного матеріалу та інтерактивних завдань. Отримавши відповідь учень може переглядати дидактичний матеріал, або відправити запит на отримання даних з бази для проходження завдання. Після відповіді серверу, учень виконує інтерактивне завдання та отримує результат.

Визначившись з основними характеристиками і функціоналом веб-сервісу, можна перейти до проектування бази даних.

База даних – це організована структура, призначена для зберігання, зміни й обробки взаємопов’язаної інформації, переважно великих обсягів [6]. Сутність в базі даних – це будь-який об’єкт в базі даних, який можна виділити виходячи з суті предметної області для якої розробляється ця база даних [7].

Для зберігання інформації в базі даних необхідно створити таблицю для кожної сутності. В таблиці 2.1 представлені основні сутності моделі даних.

Таблиця 2.1 – Таблиці бази даних та їх характеристика

Назва таблиці	Опис
users	Містить інформацію про користувачів. Таблиця має зберігати такі дані, як ідентифікатор, ім’я, прізвище, адресу електронної пошти, пароль, тип користувача(вчитель або учень) та назва рисунку(відображається як аватар користувача) .
courses	Містить інформацію про курс. У таблиці зберігаються такі дані: ідентифікатор, назва курсу, рисунок курсу та ідентифікатор вчителя (зовнішній ключ).

Продовження таблиці 2.1

sections	Містить інформацію про секції курсу. Таблиця містить такі поля, як ідентифікатор, назва секції та ідентифікатор курсу (зовнішній ключ).
materials	Містить інформацію про завантажені дидактичні матеріали. Таблиця зберігає такі дані, як ідентифікатор, назва матеріалу, назва документу (зберігається на сервері) та ідентифікатор секції (зовнішній ключ).
tasks	Містить інформацію про інтерактивне завдання. У таблиці зберігаються такі поля, як ідентифікатор, назва завдання, опис завдання, назва картинки (зберігається на сервері), ідентифікатор секції (зовнішній ключ).
answers	Містить інформацію про можливі відповіді до інтерактивного завдання. Таблиця містить такі поля, як ідентифікатор, текст відповіді, чи правильна відповідь (правда або брехня), ідентифікатор завдання (зовнішній ключ).
input_blocks	Містить інформацію про відображення поля вставки. Таблиця зберігає такі дані, як ідентифікатор, позиція по осі x, позиція по осі y, висота поля, ширина поля, ідентифікатор завдання (зовнішній ключ).

Після складання детального опису кожної сутності, для всіх таблиць створимо відповідні атрибути та задаймо їх типи даних.

Детальніше всі атрибути розглянуті в таблиці 2.2.

Таблиця 2.2 – Характеристика атрибутів бази даних

Таблиця	Атрибути	Тип даних	Опис
users	id	SERIAL	Ідентифікатор користувача
	first_name	VARCHAR (255)	Ім’я користувача
	last_name	VARCHAR (255)	Прізвище користувача
	email	VARCHAR (255)	Адреса електронної пошти користувача
	password	VARCHAR (255)	Пароль користувача
	role	ENUM (‘teacher’, ‘student’)	Роль користувача
	photoPath	STRING	Назва картинки користувача
courses	id	SERIAL	Ідентифікатор курсу
	name	VARCHAR (255)	Назва навчального курсу
	imagePath	VARCHAR (255)	Назва картинки курсу
	teacher_id	INTEGER	Ідентифікатор користувача, роль якого вчитель(зовнішній ключ)
sections	id	SERIAL	Ідентифікатор секції
	name	VARCHAR (255)	Назва секції
	course_id	INTEGER	Ідентифікатор курсу (зовнішній ключ)
materials	id	SERIAL	Ідентифікатор дидактичного матеріалу
	name	VARCHAR (255)	Назва дидактичного матеріалу
	file_path	VARCHAR (255)	Назва документу
	section_id	INTEGER	Ідентифікатор секції (зовнішній ключ)
tasks	id	SERIAL	Ідентифікатор завдання
	name	VARCHAR (255)	Назва завдання
	description	VARCHAR (255)	Опис завдання
	image	STRING	Назва картинки завдання
	section_id	INTEGER	Ідентифікатор секції (зовнішній ключ)

Продовження таблиці 2.2

answers	id	SERIAL	Ідентифікатор відповіді
	text	TEXT	Текст відповіді
	is_right	BOOLEAN	Чи правильна відповідь.
	task_id	INTEGER	Ідентифікатор завдання (зовнішній ключ)
input_blocks	id	SERIAL	Ідентифікатор блоку відповіді
	pos_x	FLOAT	Позиція по осі x на екрані
	pos_y	FLOAT	Позиція по осі y на екрані
	height	INTEGER	Висота блоку
	width	INTEGER	Ширина блоку
	task_id	INTEGER	Ідентифікатор завдання (зовнішній ключ)

У всіх таблицях також присутні атрибути created_at(дата створення поля) та updated_at(дата оновлення поля) з типом даних TIMESTAMP.

Наступний крок – це встановлення зв’язку між даними, для цього необхідно визначити відношення між таблицями БД. Зобразимо створену ER-діаграму на рисунку 2.7.

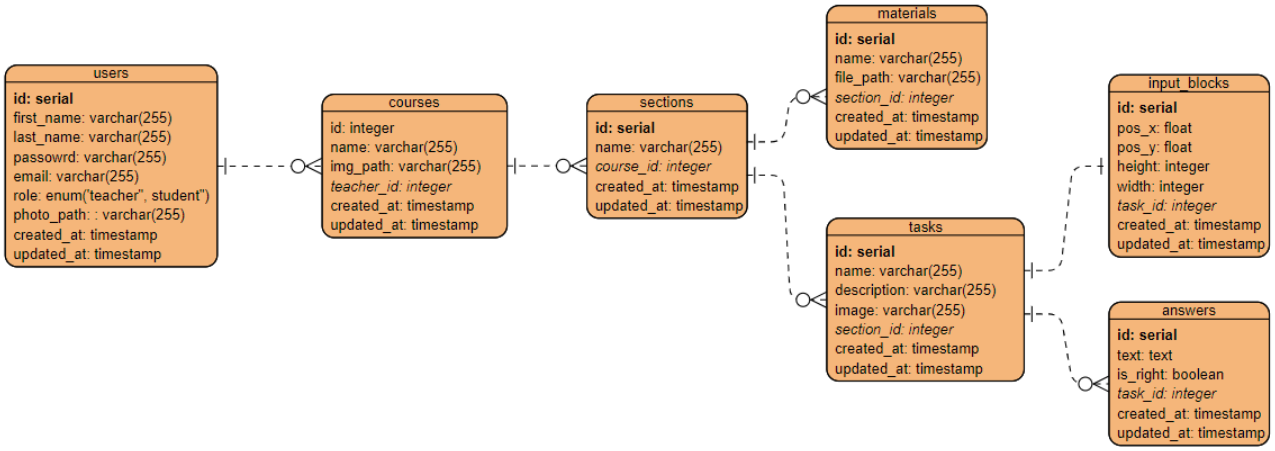


Рисунок 2.7 – Відношення між таблицями бази даних

Розглянемо відношення між таблицями БД більш детально. Кожен авторизований користувач(вчитель або учень) може мати декілька курсів. Кожен

курс може мати декілька секцій. В свою чергу секція може мати декілька дидактичних матеріалів(materials) та інтерактивних завдань(tasks). Кожне завдання може мати лише один блок відповіді(input_blocks) та декілька варіантів відповіді(answers).

Завершивши етап проектування веб-інтерфейсу інтерактивних модулів можна переходити до вибору за засобів його реалізації.

2.3 Обґрунтування вибору засобів реалізації веб-інтерфейсу

Вибір засобів реалізації проекту є не тільки одним з найважливіших етапів, але й вимагає від розробника особливої уважності до деталей та знання переваг та недоліків обраних програмних продуктів.

Розробка будь-якого веб-інтерфейсу починається зі створення прототипів. Прототипування – це процес створення прототипу майбутнього сайту. Головне завдання прототипування – правильно організувати навігацію, продумавши розташування всіх елементів, взаємодія користувача з контентом і т.п. [8]. Для створення макету проекту був обраний сучасний онлайн-сервіс Figma [9].

Figma – це хмарний сервіс для веб-дизайну, за допомогою якого можна розробляти інтерактивні прототипи сайтів та додатків, елементи інтерфейсу та векторні зображення. Усі документи програми зберігаються в хмарі, завдяки цьому немає необхідності завантажувати файли для редагування.

Наступним кроком, після прототипування, є розробка дизайну веб-сторінок. Для виконання цієї задачі також можна використати сервіс Figma, який має переваги над графічним редактором Adobe Photoshop. Figma доступна на будь-якій платформі, вона надає можливість вести спільну роботу над проектом одразу декільком фахівцям в реальному часі, також в сервісі присутні революційні системи «розумних» компонентів для створення адаптивного дизайну сайту. Також слід зазначити, що у Figma відсутні такі недоліки, як некоректне відображення макету,

поки не встановлено потрібні шрифти та не потрібно після кожного редагування макету пересилати його розробнику. Figma має велику кількість плагінів, які прискорюють роботи та допомагають ефективніше створювати макети.

Для створення найпростішої веб-сторінки, необхідно описати структуру документу. Цю задачу вирішує мова гіпертекстової розмітки HTLM, яка повідомляє браузеру як відображати веб-сторінку, яку відвідує користувач. Мова розмітки складається з ряду елементів, які використовують щоб вкладати або обертати різні частину контенту, такі як текст, малюнки, списки, таблиці, заголовки та ін. Під час створення веб-сервісу використано сучасну версію HTML5.

HTML завдає лише структуру сторінки, а для її оформлення необхідно використовувати каскадні таблиці стилів CSS. Стилі дають змогу розмежувати вміст веб-сторінки від її оформлення. Під час розробки проекту використано CSS препроцесор, а саме SASS та його діалект SCSS. Препроцесор – це програма, яка дозволяє розширити можливості чистого CSS, добавляючи такі опції, як змінні, вкладені правила, доповнення(mixin), аргументи, наслідування. Усі ці опції препроцесору дозволяють прискорити розробку та уникнути повторень.

Для повноцінного функціювання сайту необхідно використати мову програмування JavaScript, яка допомагає зробити сторінки сайту інтерактивними та обробляє дії користувачів сайту.

JavaScript - це легковажна, інтерпретована або JIT-компільована, об'єктно-орієнтована мова з функціями першого класу. Найбільш широке застосування знаходить як мову сценаріїв веб-сторінок [10]. Основні можливості та переваги JS:

– управління мультимедійними можливостями;

– створення автоматично обновлюваного контенту;

– підтримка всіма популярними браузерами;

– велика кількість бібліотек та фреймворків.

При розробці веб-інтерфейсу використана одна з найпопулярніших JavaScript бібліотек, а саме React. React – це відкрита бібліотека для створення інтерфейсу

користувача, яка вирішує проблему з частковим оновлення вмісту веб-сторінки [11]. До переваги бібліотеки можна віднести:

– віртуальна об’єктна модель документа;

– повторне використання компонентів;

– універсальність використання (можна використовувати і на сервері і у мобільних додатках);

– браузерні інструменти розробника.

При побудові веб-інтерфейсу необхідно використовувати СУБД, оскільки сервіс має зберігати дані про користувачів, курси, секції, дидактичний матеріал та завдання.

Оскільки онлайн-платформа має зберігати дані про вчителів, учнів та їх успішність, – виникає потреба у використанні СУБД. При детальному розгляді різноманітних варіантів, було вирішено використовувати систему PostgreSQL.

PostgreSQL – це потужна об’єктно-реляційна база даних з відкритим вихідним кодом, з надійними функціями і продуктивністю [12]. В ролі основної мови для управління БД PostgreSQL використовує мову структурованих запитів SQL. Серед переваг використання цієї СУБД можна виділити:

– підтримка БД необмеженого розміру;

– легка та зручна масштабованість;

– підтримка JSON запитів;

– підтримка розширених типів даних;

– відповідає набору властивостей ACID, що гарантують надійну роботу транзакцій БД.

Для повноцінної взаємодії веб-інтерфейсу з БД необхідно створити та розгорнути сервер, у зв’язку з цим виникає потреба використання Node.js – асинхронне подієве JavaScript-оточення, яке спроектовано для побудови мережних додатків [13]. Node.js містить в собі величезну кількість вбудованих модулів, таких як: робота з файловою системою, робота з протоколами HTTP та HTTPS, робота з веб-адресою та багато інших. Також підтримується npm(Node Package Manager),

який дозволяє користуватися сторонніми модулями для пришвидшення та покращення розробки проектів. Не менш важливим є те, що Node.js має високу масштабованість, допомагає серверу у неблокуючій відповіді і має високу продуктивність у порівнянні з іншими серверними мовами програмування.

При розробці серверу використано Express.js – мінімалістичний та гнучкий веб-фреймворк для програм Node.js, що надає широкий набір функцій для мобільних та веб-додатків Веб-фреймворк має в своєму розпорядженні безліч службових методів HTTP та проміжних обробників, створити надійний API можна швидко та легко [14].

Також для полегшення підключення та пришвидшення написання запитів до БД використано ORM, а саме Sequelize. Ця ORM дозволяє взаємодіяти з БД мовою JavaScript, без використання SQL. Однак існують ситуації, коли запит можна виконати лише за допомогою мови структурних запитів.

3 ВЕБ-ІНТЕРФЕЙСУ ІНТЕРАКТИВНИХ МОДУЛІВ

3.1 Розробка прототипів та дизайн-макетів сторінок веб-інтерфейсу

Спочатку необхідно визначити, які сторінки буде мати наш веб-інтерфейс інтерактивних модулів. Зображену структуру відображено на рисунку 3.1.

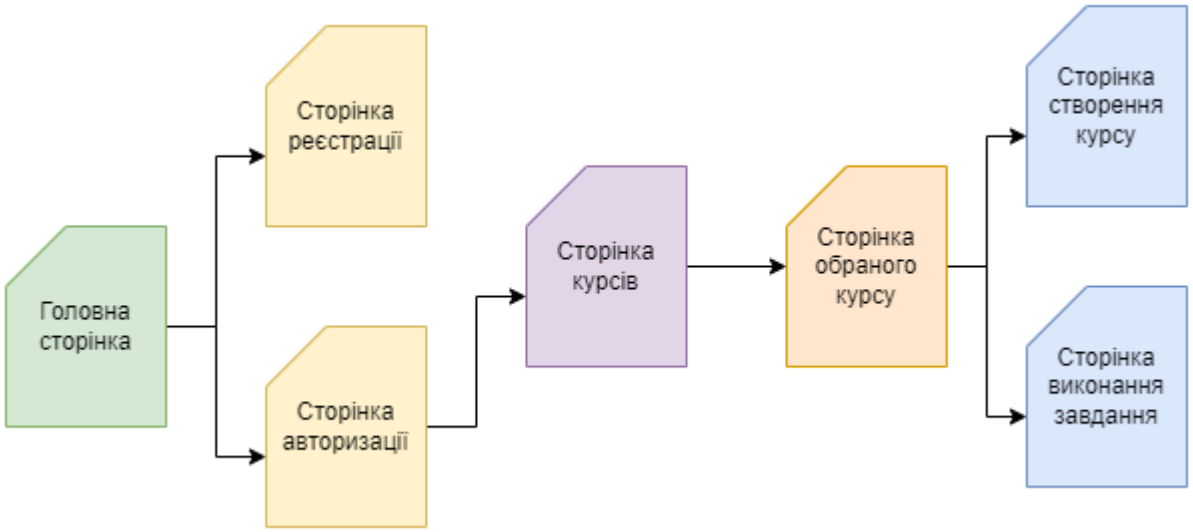


Рисунок 3.1 – Структура веб-інтерфейсу

Використовуючи створену схему, розробимо прототипи сторінок веб-інтерфейсу інтерактивних модулів.

Спочатку створимо прототип головної сторінки, вона буде вміщати в собі логотип, інформацію о проекті і кнопки входу та реєстрації. Прототип головної сторінки відображений на рисунку 3.2.

Рисунок 3.2 – Прототип головної сторінки

Наступним кроком розробимо прототип сторінки реєстрації, на якій буде відображена форма для введення даних користувача. На рисунку 3.3 зображено прототип сторінки «Реєстрація».

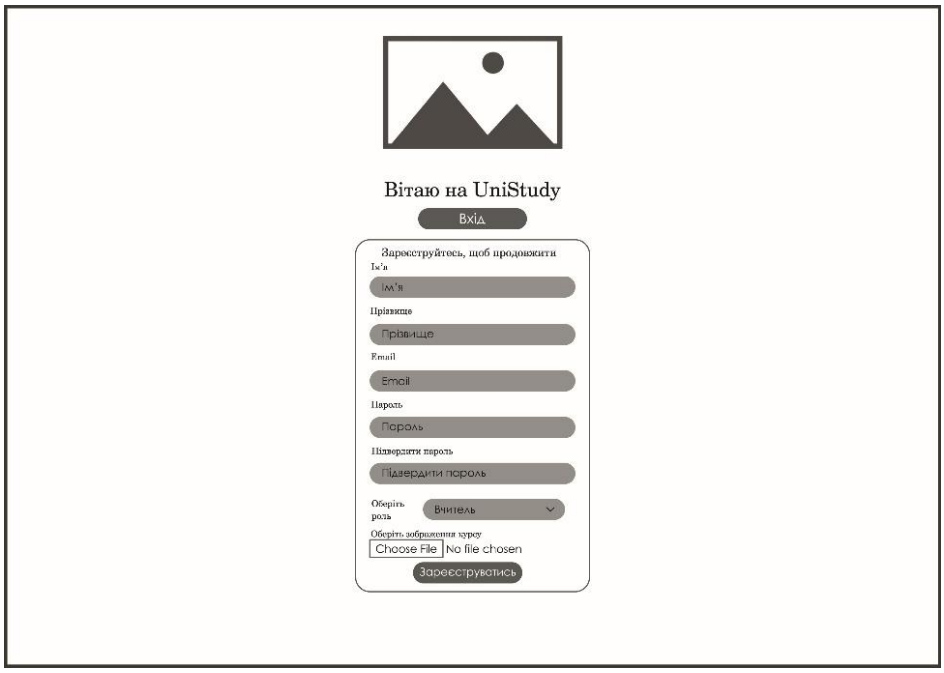


Рисунок 3.3 – Прототип сторінки «Реєстрація»

Виконавши реєстрацію, користувач має побачити форму авторизації, в якій необхідно ввести дані для входу у систему. Прототип сторінки «Авторизація» відображений на рисунку 3.4.

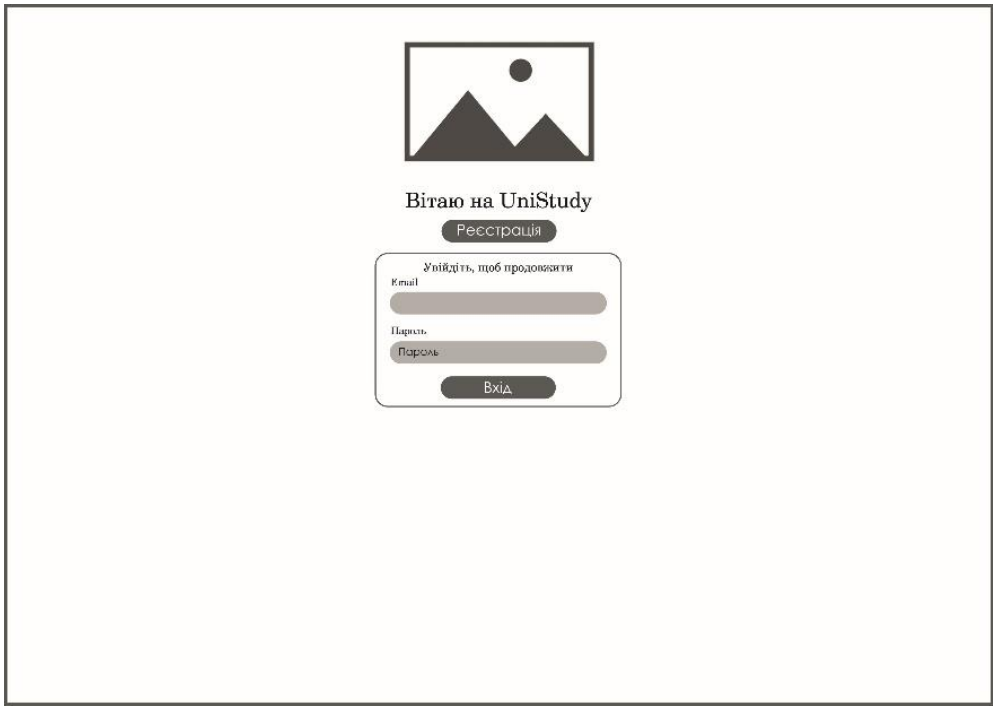


Рисунок 3.4 – Прототип сторінки «Авторизація»

Після процесу авторизації користувач переходить до сторінки курсів. В залежності від ролі користувача, вчитель має побачити тільки його курси, та кнопку для створення нового курсу, а учень має побачити усі існуючи курси. На рисунку 3.5 зображено прототип сторінки «Курси»

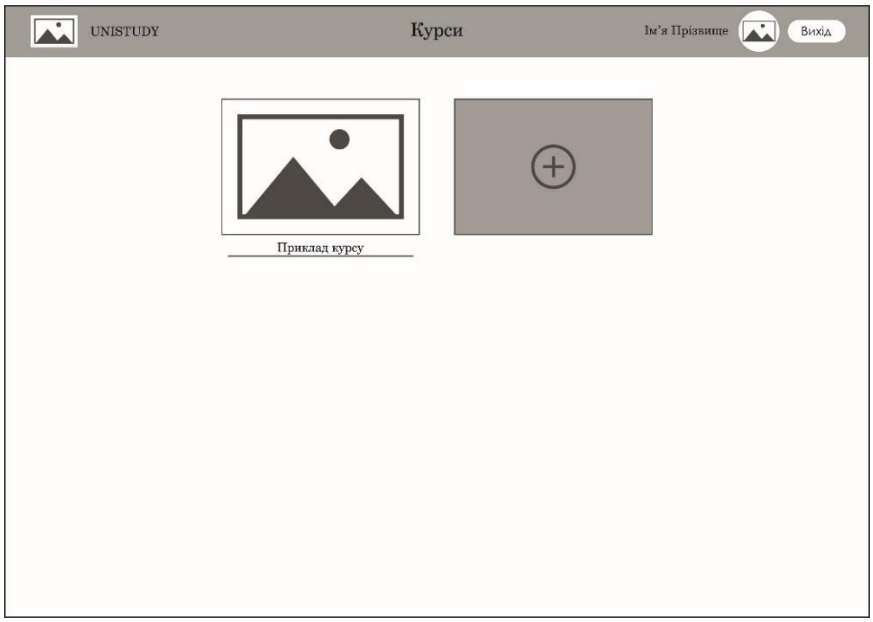


Рисунок 3.5 – Прототип сторінки «Курси»

Обравши курс виконується перехід до сторінки «Обраного курсу», учень має побачити наповнення курсу, а саме секції у яких є дидактичний матеріал та інтерактивне завдання, у вчителя додатково відображаються кнопки створення секції, інтерактивного та додавання завдання дидактичного матеріалу. Прототип сторінки «Обраний курс» відображено на рисунку 3.6.

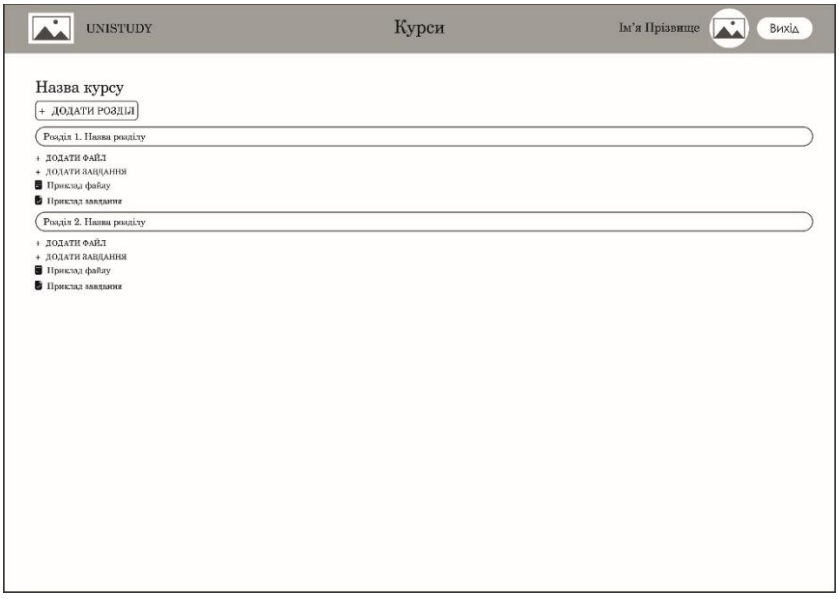


Рисунок 3.6 – Прототип сторінки «Обраний курс»

Обравши завдання в учня виконується перехід до сторінки «Проходження інтерактивного завдання», де відображено опис завдання, варіанти відповіді та інтерактивний блок. Розробляючи прототип даної сторінки, слід враховувати розташування блоків, шрифт тексту та його кегль, оскільки на цій сторінці відображається головна ідея проекту – інтерактивні завдання. На рисунку 3.7 зображено прототип сторінки «Проходження інтерактивного завдання».

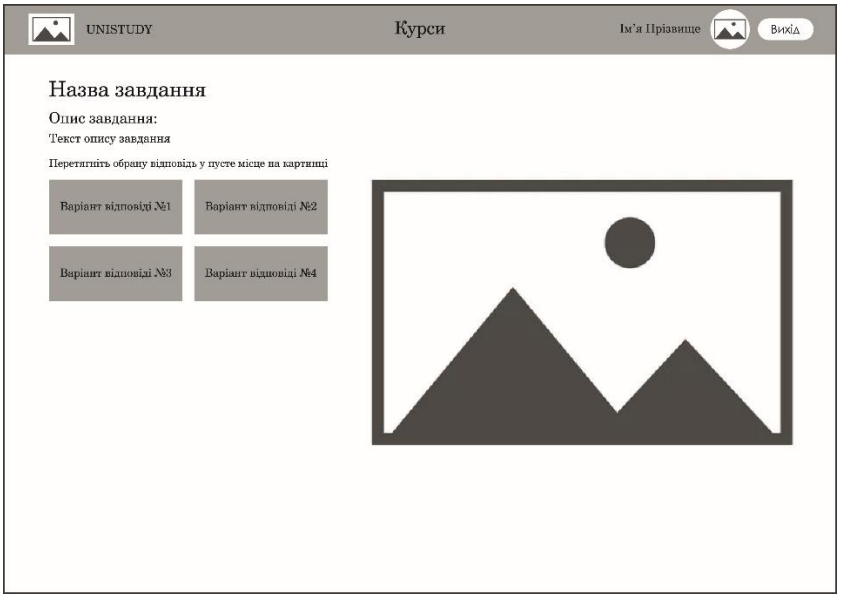


Рисунок 3.7 – Прототип сторінки «Проходження інтерактивного завдання»

Натиснувши на кнопку створення завдання у вчителя виконується перехід на сторінку «Створення інтерактивного завдання», де відображено форму створення інтерактивного завдання та область, де відображається картинка та обирається місце де необхідно вставити відповідь. Прототип сторінки «Створення інтерактивного завдання» відображено на рисунку 3.8.

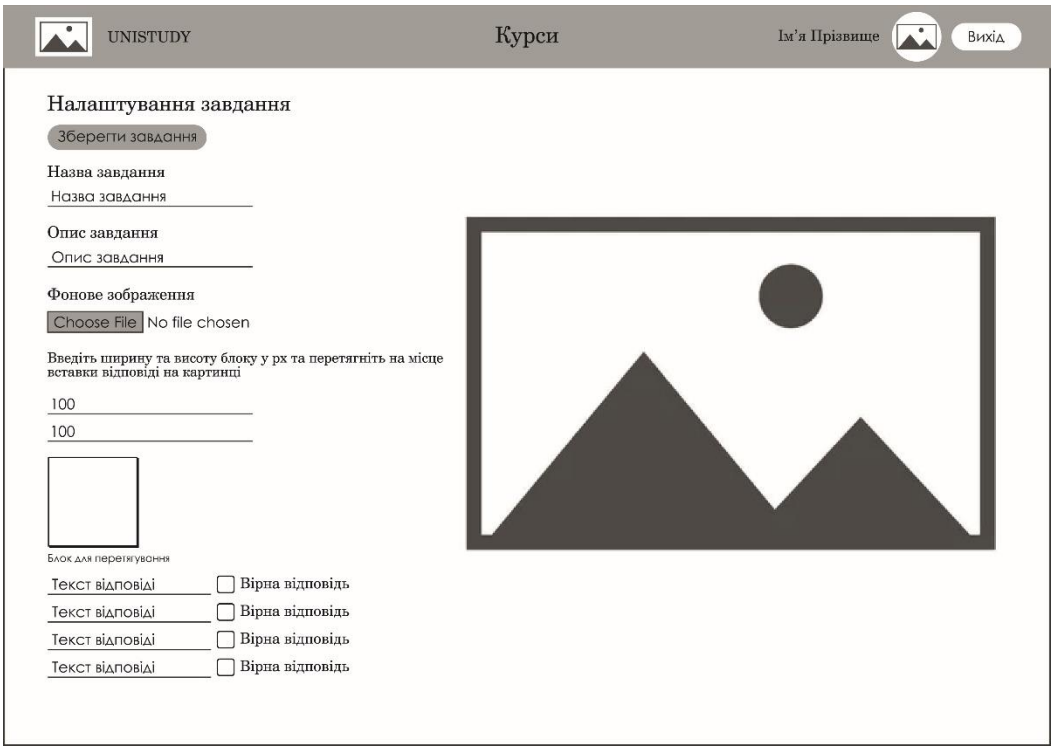


Рисунок 3.8 – Прототип сторінки «Створення інтерактивного завдання»

Використовуючи розроблені прототипи сторінок веб-інтерфейсу інтерактивних модулів, були створені дизайн-макети, наведені в додатку Б.

3.2 Розробка клієнтської частини веб-інтерфейсу інтерактивних модулів

Програмну реалізації клієнтської частини можна розділити на декілька етапів. Перший етап буде присвячений розробці компонентів інтерфейсу користувача з використанням технологій HTML, SCSS, JS та React. В додатку В наведені деякі компоненти, які використані у проекті. Для розширення функціоналу сторінок та надання їм динамічності використовувалось ряд пакетів та бібліотек:

– Formik – бібліотека для швидкої побудови форм у React [15];

– react-draggable – пакет щоб зробити елементи перетягуваними [16];

– react-loading – пакет для анімації завантаження у React [17].

– mui – бібліотека компонентів React, яка надає готові Google рішення для швидкої і простої веб-розробки [18].

Також, на цьому етапі необхідно виділити розробку управління станом веб-сервісу. Для вирішення цього завдання у веб-інтерфейсі використовується Redux – це контейнер передбачуваного стану для JavaScript-додатків [19]. В додатку Г наведений приклад зберігання стану аутентифікації користувача. В основі Redux використовуються 3 основні концепції, а саме:

– існує єдине джерело істини для всього стану програми;

– стан доступний тільки для читання

– всі зміни в стан додатку вносяться за допомогою чистих функцій.

Для повноцінного функціювання Redux, а також для полегшення запитів на сервер були використанні наступні бібліотеки та модулі:

– axios – це HTTP-клієнт, оснований на Promise і використовує XMLHttpRequests [20];

– redux-toolkit – пакет, який спрощує роботу з Redux та вирішує його основні проблеми [21];

– react-redux – пакте, який розроблений для роботи з компонентною моделлю React [22].

3.3 Розробка серверної частини веб-інтерфейсу інтерактивних модулів

Для взаємодії з базою даних PostgreSQL необхідно розробити сервер. Розробку серверної частини також можна поділити на декілька етапів. На першому етапі необхідно провести налаштування серверу, а саме обрати протокол передачі(в нашому випадку – HTTP), вказати порт та шлях на якому працює сервер, для цього використовуються вбудовані модулі Node.js.

На наступному етапі необхідно встановити ORM Sequelize та створити міграції, які відслідковують зміни в базі даних та моделі, які представляють наші

таблиці в БД, для подальшого їх використання при виконанні запитів до бази даних. Приклад міграції та моделі курсу наведені у додатку Д. Результат міграції – це створені таблиці у БД, які наведені у додатку Ж. Також необхідно налаштувати підключення до бази даних за допомогою JSON файлу:

```
{
  "development": {
    "username": "postgres",
    "password": "postgres",
    "database": "unistudy",
    "host": "127.0.0.1",
    "dialect": "postgres",
    "operatorsAliases": "Op",
    "seederStorage": "sequelize"
  }
}.
```

Оскільки в нашому проекті використовується архітектурний шаблон MVC, діаграма якого наведена на рисунку 3.9, то на наступному етапі нам необхідно створити контролер(модель керування).

Контролер перевіряє вхідні дані на предмет загальної коректності та відповідність вимогам моделі та вносить відповідні зміни до бази даних. У додатку Е представлений приклади контролерів.

Також важливим етапом створення серверу є функції проміжної обробки, утиліти та сервісі. Функції проміжної обробки представляють зручний механізм для фільтрації HTTP-запитів додатку. Під час розробки веб-серверу були створенні такі функції проміжної обробки:

– перевірка токену доступу(access token);

– пошук користувача, курсу та секції;

– обробник помилок.

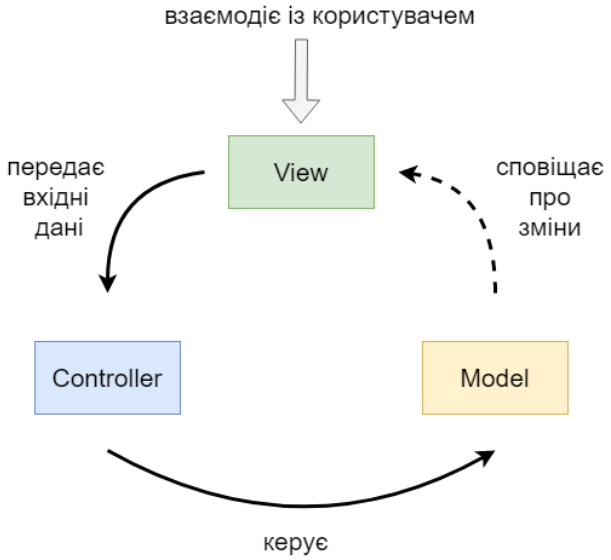


Рисунок 3.9 - Діаграма архітектурного шаблону MVC

Утиліти були розроблені для збереження на сервері картинок та файлів, за допомогою пакету multer, який використовується для завантаження файлів. Також були розроблені сервіси аутентифікації. При кожному вході у систему та оновленні сторінки відправляється запит на сервер, перевіряється токен доступу користувача з існуючим у базі даних, якщо токен не підроблений або не прострочений, то користувач отримує усю необхідну інформацію для роботи з веб-інтерфейсом, інакше йому необхідно виконати запит на авторизацію заново. Важливим моментом залишається безпечне зберігання паролів у базі даних, тому на сервері під час реєстрації та авторизації використовується bcrypt – адаптивна криптографічна функція формування ключа, яка заснована на шифрі Blowfish.

3.4 Розробка інструкції з експлуатації створеного програмного продукту

Для того, щоб гість приєднався до веб-інтерфейсу інтерактивних модулів, йому необхідно пройти реєстрацію та обрати роль вчителя або ученика, її потім змінити не можливо. Пройшовши реєстрацію гість вводить дані до форми авторизації, якщо дані співпадають з тими, що знаходиться у базі даних, то відбувається вхід у систему.

Користувач потрапляє на сторінку з курсами. Вчителю доступна можливість створити новий курс, додати назву та картинку, яка буде використовуватись як обкладинка курсу, але вчителю недоступна можливість переглядати курси інших викладачів. Учню доступні усі курси, які були створені вчителями.

Натиснувши курс, користувач потрапляє на сторінку обраного курсу. Вчитель спочатку має пустий курс та кнопку «Додати секцію», заповнивши форму, додається нова секція. У новій секції знаходяться дві кнопки – «Додати файл» та «Додати завдання». Натиснувши на кнопку «Додати файл» з'являється діалогове вікно, у якому необхідно ввести назву файлу, яка буде відображатись і додати сам файл, файл одразу з'являється на сторінці курсу у відповідній секції.

Натиснувши на кнопку «Створити завдання» вчитель потрапляє на сторінку зі створення інтерактивного завдання, на якій необхідно виконати наступні дії:

– заповнити поле «Назва завдання»;

– заповнити поле «Опис завдання»;

– завантажити рисунок, на якому необхідно буде обрати область вставки відповіді;

– завдати ширину та висоту області вставки відповіді;

– перетягнути область вставки на картинку;

– заповнити поля відповідей та відмітити правильну;

– натиснути на кнопку «Зберегти завдання».

Після збереження завдання виконається перехід на сторінку курсу та завдання вже буде доступно у відповідній секції.

Учень потрапивши на сторінку обраного курсу може переглядати секції, натиснувши на дидактичний матеріал, він почне завантажуватись учню на комп'ютер. Натиснувши на інтерактивне завдання, учень потрапляє на сторінку з завдання, на якій відображається назва завдання, опис завдання, варіанти відповідей та малюнок з позначеною зоною, куди необхідно перетягнути відповідь. Після перетягування відповіді у відмічену зону, учень отримує повідомлення, була відповідь правильна чи ні. Учень має безліч кількість спроб на виконання завдання та може його виконувати безкінечну кількість разів.

ВИСНОВКИ

В даній кваліфікаційній роботі розглядалося створення сучасних веб-інтерфейсів інтерактивних модулів для використання у освітньому процесі, а також виконано огляд існуючих веб-сервісів інтерактивних завдань.

В ході роботи виконані наступні завдання:

– розглянуто теоретичні аспекти використання веб-інтерфейсів інтерактивних модулів;

– проаналізовано схожі веб-сервіси інтерактивних завдань;

– розроблені діаграми прецедентів та послідовності для всіх типів користувачів системи;

– спроектовано базу даних з дотриманням необхідних умов для коректної взаємодії веб-сервісу з таблицями.

– розроблені прототипи та дизайн-шаблони сторінок;

– виконана програмна реалізація клієнтської та серверної частин веб-інтерфейсу інтерактивних модулів;

– сформовано інструкції з використання веб-сервісу.

Загальним результатом виконання кваліфікаційної роботи є розробка веб-інтерфейсу інтерактивних модулів для використання у навчальному процесі, яка може бути в подальшому вдосконалена шляхом додавання нових типів інтерактивних завдань.

ПЕРЕЛІК ПОСИЛАНЬ

1. Kahoot. URL: https://kahoot.com/ (дата звернення: 30.10.22).
2. LearningApps. URL: https://learningapps.org/ (дата звернення: 30.10.22).
3. Online Test Pad. URL: https://onlinetestpad.com/ (дата звернення: 30.10.22).
4. Quizizz: Where motivation meets mastery. URL: https://quizizz.com/ (дата звернення: 30.10.22).
5. Alan Dennis. Systems analysis & design: an object-oriented approach with UML. Wiley Publishing, 2015. 544p.
6. HostIQ. URL: https://hostiq.ua/wiki/ukr/database/ (дата звернення: 30.10.22).
7. Бази даних. Поняття сутності. BestProg. URL: https://www.bestprog.net/uk/2019/01/24/the-concept-of-er-model-the-concept-of-essence-and-communication-attributes-attribute-types-ua/ (дата звернення: 30.10.22).
8. Webpromo. URL: https://web-promo.ua/ua/wordbook/prototipirovanie-sajta/ (дата звернення 31.10.22).
9. Figma: the collaborative interface design tool. URL: https://www.figma.com/ (дата звернення: 31.10.2022).
10. JavaScrpit - MDN Web Docs – Mozilla. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 31.10.2022).
11. React – A JavaScript library for building user interfaces. URL: https://reactjs.org/ (дата звернення 31.10.2022).
12. PostgreSQL: The world`s most advanced pen source database. URL: https://www.postgresql.org/ (дата звернення 31.10.2022).
13. Node.js. URL: https://nodejs.org/uk/ (дата звернення 31.10.2022).
14. Express.js.URL: https://expressjs.com/ (дата звернення 31.10.2022).

15. Formik. Build forms in React. URL: https://formik.org/ (дата звернення 17.05.2019).

16. NPM. React-draggable. URL: https://www.npmjs.com/package/react-draggable (дата звернення 31.10.2022).

17. NPM. React-loading. URL: https://www.npmjs.com/package/react-loading (дата звернення 31.10.2022).

18. MUI. URL: https://mui.com/ (дата звернення 31.10.2022).

19. Redux. URL: https://redux.js.org/ (дата звернення 31.10.2022).

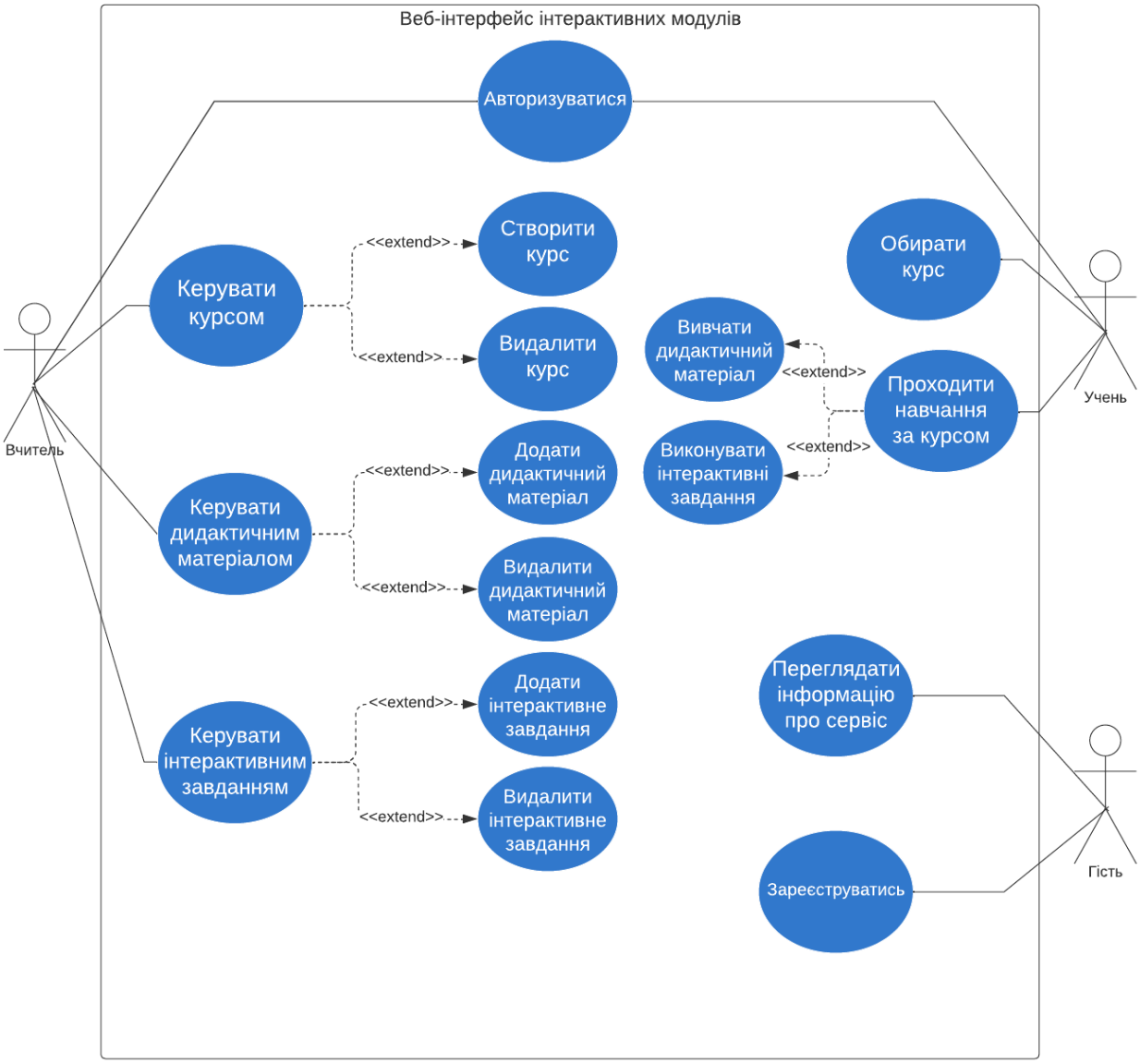
20. Axios. URL: https://axios-http.com/ (дата звернення 31.10.2022).

21. Redux Toolkit. URL: https://redux-toolkit.js.org/ (дата звернення 31.10.2022).

22. React-redux. URL: https://react-redux.js.org/ (дата звернення 31.10.2022)

ДОДАТОК А

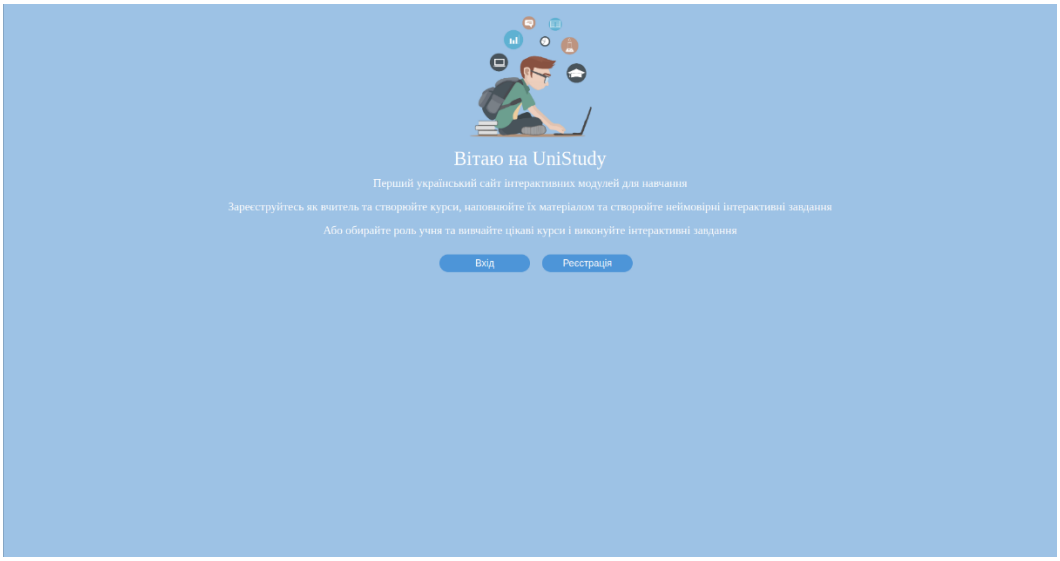
Повна Use Case діаграма



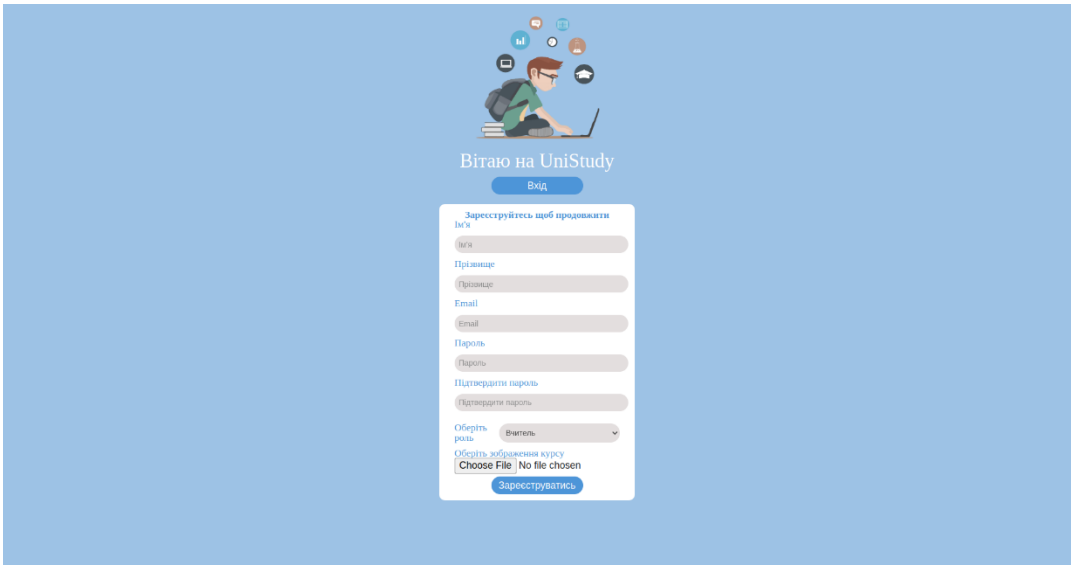
ДОДАТОК Б

Дизайн-макети сторінок веб-інтерфейсу інтерактивних модулів

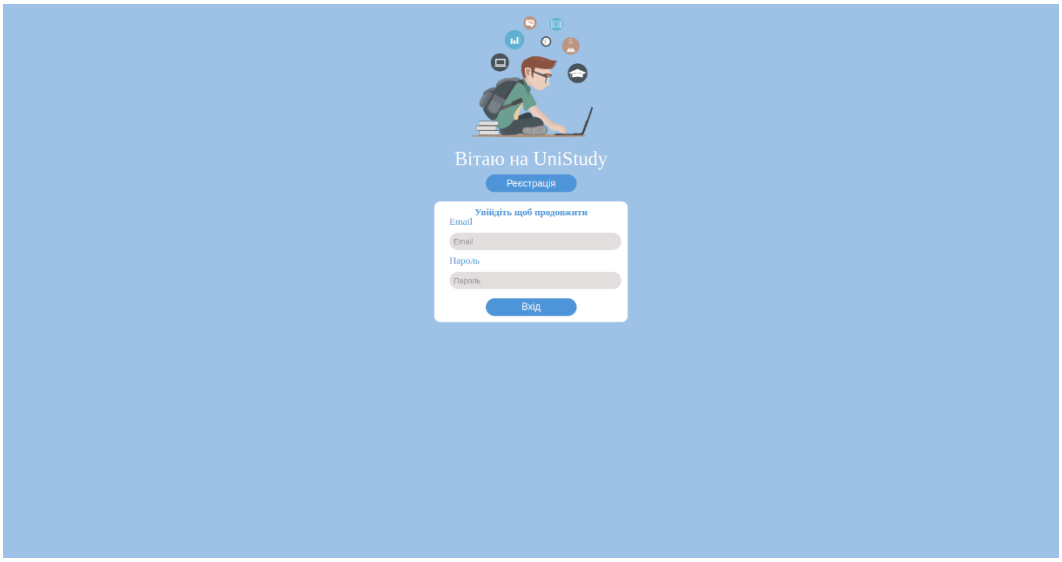
Б.1 Дизайн-макет головної сторінки веб-інтерфейсу інтерактивних модулів



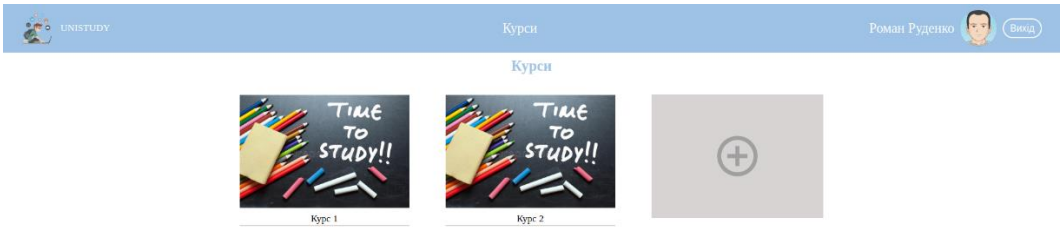
Б.2 Дизайн-макет сторінки реєстрації



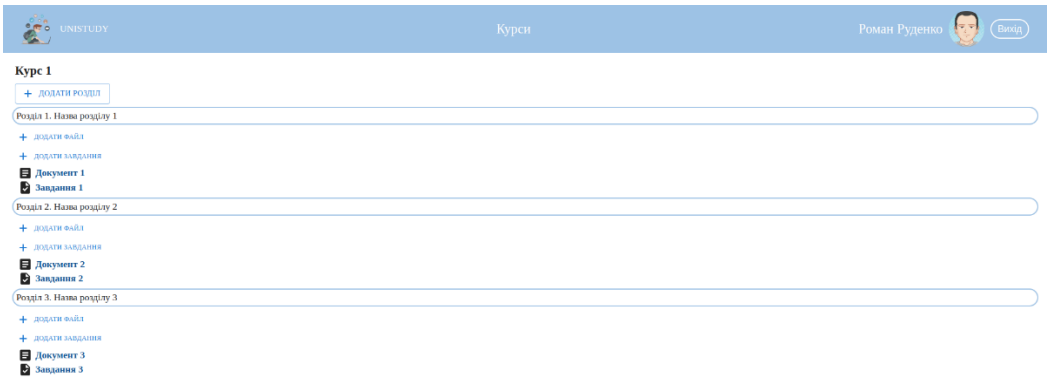
Б.3 Дизайн-макет сторінки авторизації



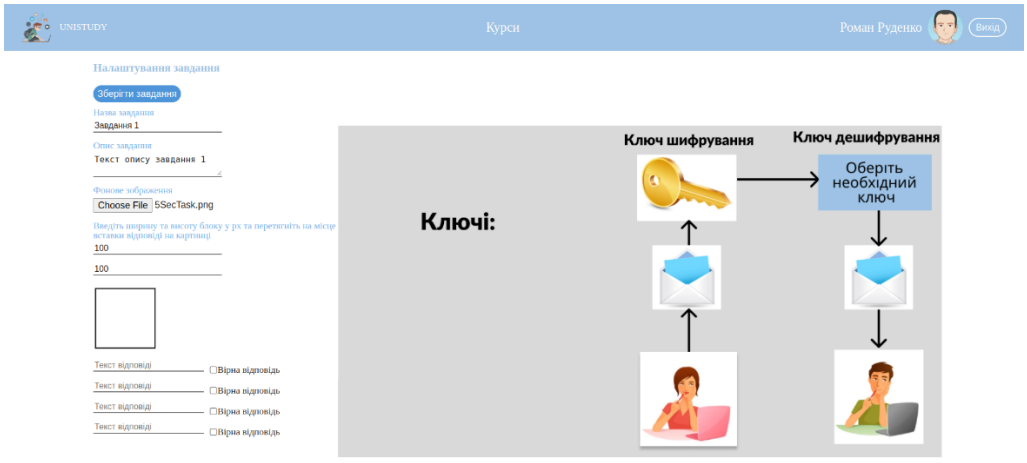
Б.4 Дизайн-макет сторінки «Курси»



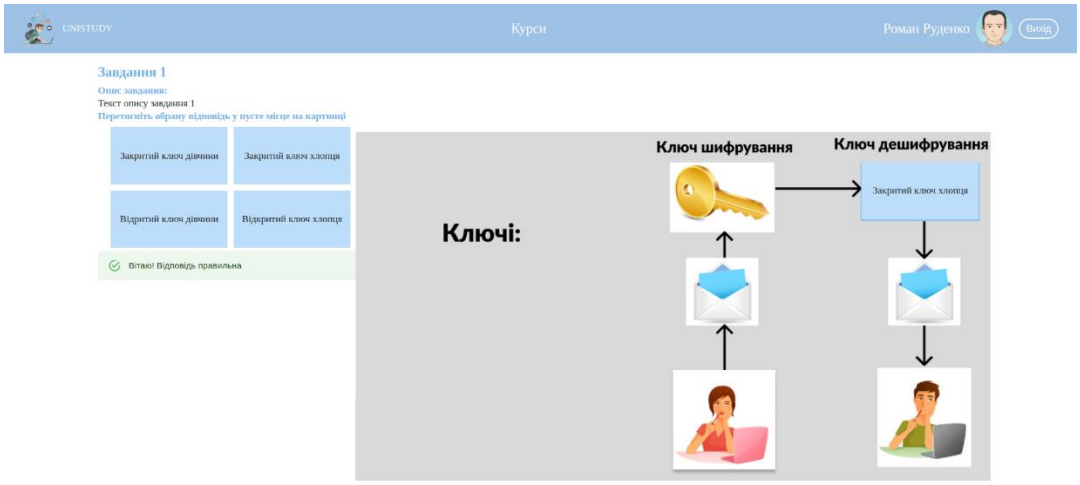
Б.5 Дизайн-макет сторінки обраного курсу



Б.6 Дизайн-макет сторінки для створення завдання



Б.7 Дизайн-макет сторінки виконання завдання


UNISTUDY
Курси
Роман Руденко
Вихід
Завдання 1
Опис завдання:
Текст опису завдання 1
Перетягніть обрану відповідь у пусте місце на картинці
Закритий ключ дівчини
Закритий ключ хлопця
Відкритий ключ дівчини
Відкритий ключ хлопця
Вітаю! Відповідь правильна
Ключі:
Ключ шифрування
Ключ дешифрування
Закритий ключ хлопця

ДОДАТОК В

React-компоненти, використані у проекті

В.1 React-компонент форми авторизації

```
import React from "react";
import { Formik, Form, Field } from "formik";
import styles from "./LogInForm.module.scss";
import { useDispatch} from "react-redux";
import { login } from "redux/features/auth/authThunk";
import validationSchema from "../../validation/validationSchema";

const LogInForm = () => {
 const dispatch = useDispatch();

 const initialValues = {
  email: "",
  password: "",
 };

 const handleLogin = ({ email, password }) => {
  dispatch(login({ email, password }));
 };

 return (
  <>
   <Formik
    initialValues={initialValues}
    onSubmit={handleLogin}
    validationSchema={validationSchema.LoginSchema}
   >
    {({ errors, touched }) => (
     <Form className={styles.container}>
      <h2 className={styles.heading}>Увійдіть щоб продовжити</h2>
      <div className={styles.fieldContainer}>
       <p className={styles.fieldName}>Email</p>
       <Field
        name="email"
        className={styles.field}
        type="email"
        placeholder="Email"
```

```
          />
          {errors.email && touched.email ? (
            <div className={styles.validationError}>{errors.email}</div>
          ) : null}
         </div>
         <div className={styles.fieldContainer}>
          <p className={styles.fieldName}>Пароль</p>
          <Field
            name="password"
            className={styles.field}
            type="password"
            placeholder="Пароль"
            autoComplete="on"
          />
          {errors.password && touched.password ? (
            <div className={styles.validationError}>{errors.password}</div>
          ) : null}
         </div>

         <button className={styles.btn} type="submit">
          Вхід
         </button>
        </Form>
      )}
    </Formik>
   </>
  );
};

export default LogInForm;
```

B.2 React-компонент списку курсів

```
import React from "react";
import { Formik, Form, Field } from "formik";
import styles from "./LogInForm.module.scss";
import { useDispatch} from "react-redux";
import { login } from "redux/features/auth/authThunk";
import validationSchema from "../../validation/validationSchema";

const LogInForm = () => {
  const dispatch = useDispatch();

  const initialValues = {
    email: "",
    password: "",
  };
```

```
 const handleLogin = ({ email, password }) => {
  dispatch(login({ email, password }));
 };

 return (
  <>
   <Formik
    initialValues={initialValues}
    onSubmit={handleLogin}
    validationSchema={validationSchema.LoginSchema}
   >
    {({ errors, touched }) => (
     <Form className={styles.container}>
      <h2 className={styles.heading}>Увійдіть щоб продовжити</h2>
      <div className={styles.fieldContainer}>
       <p className={styles.fieldName}>Email</p>
       <Field
        name="email"
        className={styles.field}
        type="email"
        placeholder="Email"
       />
       {errors.email && touched.email ? (
        <div className={styles.validationError}>{errors.email}</div>
       ) : null}
      </div>
      <div className={styles.fieldContainer}>
       <p className={styles.fieldName}>Пароль</p>
       <Field
        name="password"
        className={styles.field}
        type="password"
        placeholder="Пароль"
        autoComplete="on"
       />
       {errors.password && touched.password ? (
        <div className={styles.validationError}>{errors.password}</div>
       ) : null}
      </div>

      <button className={styles.btn} type="submit">
       Вхід
      </button>
     </Form>
    )}
   </Formik>
  </>
 );
};

export default LogInForm;
```

В.3 React-компонент секції у курсі

```
import React, { useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { getSections } from "redux/features/section/sectionThunk";
import styles from "./Section.module.scss";
import AddNewMaterial from "components/AddNewMaterial";
import Material from "components/Material";
import { getMaterials } from "redux/features/material/materialThunk";
import AddNewTask from "components/AddNewTask";
import TaskLink from "components/TaskLink";
import { getTask } from "redux/features/task/taskThunk";

const Section = ({ courseId }) => {
 const dispatch = useDispatch();

 useEffect(() => {
  dispatch(getSections(courseId));
  dispatch(getMaterials(courseId));
  dispatch(getTask(courseId));
 }, [dispatch, courseId]);

 const { sectionData } = useSelector((state) => state.section);
 const { userData } = useSelector((state) => state.auth);

 return (
  <ul>
   {sectionData.map((section, id) => (
    <li key={`${section.id}`}>
     <h3 className={styles.itemHeading}>
      Розділ {id + 1}. {section.name}
     </h3>
     {userData.role === "teacher" ? (
      <div>
       <AddNewMaterial sectionId={section.id} />
       <AddNewTask sectionId={section.id} />
      </div>
     ) : null}
     <Material id={section.id} />
     <TaskLink id={section.id} />
    </li>
   ))}
  </ul>
 );
};

export default Section;
```

ДОДАТОК Г

Приклад зберігання стану аутентифікації користувача

Г.1 authThunk.js – файл у якому виконуються запит на сервер

```
import { createAsyncThunk } from "@reduxjs/toolkit";
import { removeTokens, setToken } from "utils/helperFunctions";
import api from '../../../api/http';
import CONSTANTS from '../../../constants';

export const fetchUserData = createAsyncThunk('auth/fetchUserData',
  async (data, { rejectWithValue }) => {
    try {
      const response = await api.get('auth/getUser');
      return response.data;
    } catch (error) {
      removeTokens();
      rejectWithValue(error);
    }
  });

export const register = createAsyncThunk('auth/register',
  async (payload) => {
    const response = await api.post('auth/register', payload);
    return response.data;
  })

export const login = createAsyncThunk('auth/login',
  async (payload) => {
    const response = await api.post('auth/login', payload);
    setToken(CONSTANTS.ACCESS_TOKEN, response.data.tokenPair.accessToken);
    setToken(CONSTANTS.REFRESH_TOKEN, response.data.tokenPair.refreshToken);
    return response.data;
```

```
  });
export const signOut = createAsyncThunk('auth/signOut', async () => {
  removeTokens();
});
```

Г.2 authSlice.js – файл у якому генеруються творці дій(action creators) і типи дій(action types), які відповідають редукторам і стану

```
import { createSlice } from "@reduxjs/toolkit";
import { fetchUserData, login, signOut, register } from "./authThunk";

const initialState = {
  userData: {},
  token: null,
  loading: false,
};
export const authSlice = createSlice({
  name: 'auth',
  initialState,
  reducers: {},
  extraReducers: {
    [signOut.fulfilled]: (state, action) => {
      state.loading = false;
      state.userData = {};
      state.token = null;
    },
    [login.pending]: (state, action) => {
      state.loading = true;
    },
    [login.fulfilled]: (state, action) => {
      const {tokenPair, user} = action.payload;
      state.loading = false;
      state.userData = user;
      state.token = tokenPair.accessToken;
    },
    [login.rejected]: (state, action) => {
      state.loading = false;
```

```
    },
    [fetchUserData.fulfilled]: (state, action) => {
      const {tokenPair, user} = action.payload;
      state.loading = false;
      state.userData = user;
      state.token = tokenPair.accessToken;
    },
    [fetchUserData.rejected]: (state, action) => {
      state.loading = false;
      state.userData = {};
      state.token = null;
    },
    [register.pending]: (state, action) => {
      state.loading = true;
    },
    [register.fulfilled]: (state, action) => {
      const {data} = action.payload;
      state.loading = false;
      state.userData = data;
    },
    [register.rejected]: (state, action) => {
      state.loading = false;
    },
  }
});

export default authSlice.reducer;
```

ДОДАТОК Д

Приклад міграції та моделі курсу

Д.1 2022100507146-create-course.js – міграція на створення таблиці курсу

```
'use strict';
/** @type {import('sequelize-cli').Migration} */
module.exports = {
  async up(queryInterface, Sequelize) {
    await queryInterface.createTable('courses', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      name: {
        type: Sequelize.STRING,
        allowNull: false
      },
      teacherId: {
        field: 'teacher_id',
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: 'users',
          key: 'id'
        },
        onDelete: 'cascade',
        onUpdate: 'cascade'
      },
      imgPath : {
        field: 'img_path',
        type: Sequelize.STRING,
        allowNull: true
      },
```

```
    createdAt: {
      field: 'created_at',
      allowNull: false,
      type: Sequelize.DATE
    },
    updatedAt: {
      field: 'updated_at',
      allowNull: false,
      type: Sequelize.DATE
    }
   });
  },
  async down(queryInterface, Sequelize) {
   await queryInterface.dropTable('courses');
  }
};
```

Д.2 course.js – модель курсу

```
'use strict';
const { Model } = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class Course extends Model {

    static associate({ User, Section }) {
      Course.belongsTo(User, {foreignKey: 'teacherId'}),
      Course.belongsToMany(User, {
        through: 'students_to_coruses',
        foreignKey: 'courseId',
      }),
      Course.hasMany(Section, {foreignKey: 'courseId'})
    }
  }
  Course.init({
    name: {
      type: DataTypes.STRING,
      allowNull: false,
      validate: {
        notEmpty: true,
        notNull: true
      }
```

```
    },
    imgPath: {
      field: 'img_path',
      type: DataTypes.STRING
    }
  }, {
    underscored: true,
    sequelize,
    tableName: 'courses',
    modelName: 'Course',
  });
  return Course;
};
```

ДОДАТОК Е

Приклад контролерів

Е.1 taskController.js – контролер інтерактивного завдання

```
const { Section, Task } = require('../db/models');
module.exports.createTask = async (req, res, next) => {
  try {
    const { section, body, file: { filename } } = req;
    const parsedAnswers = JSON.parse(body.answers);
    const parsedInputBlock = JSON.parse(body.inputBlock)
    const task = await section.createTask({ image: filename, name: body.name, description:
body.description });
    const answers = parsedAnswers.filter(answer => answer.text !== '');
    await answers.forEach((item) => {
      task.createAnswer(item)
    })
    await task.createInputBlock(parsedInputBlock);
    res.status(200).send('Task created!')
  } catch (error) {
    next(error);
  }
}
module.exports.getTasks = async (req, res, next) => {
  try {
    const { course } = req;
    const sectionsId = await Section.findAll({
      where: { courseId: course.id },
      attributes: ['id']
    })
      .then(sectionsId => sectionsId.map(id => id.id));
    const tasks = await Task.findAll({ where: { sectionId: sectionsId } });
```

```
    res.status(200).send(tasks);
  } catch (error) {
    next(error);
  }
};
module.exports.getTaskById = async (req, res, next) => {
  try {
    const { query: { id } } = req;
    const task = await Task.findByPk(id);
    const answers = await task.getAnswers();
    const inputBlock = await task.getInputBlock();
    res.status(200).send({task, answers, inputBlock})
  } catch (error) {
    next(error);
  }
          }
```

E.2 sectionController – контролер секції

```
module.exports.getSections = async (req, res, next) => {
  try {
    const { course } = req;
    const sections = await course.getSections();
    res.status(200).send(sections);
  } catch (error) {
    next(error);
  }
};
module.exports.createSection = async(req, res, next) => {
  try {
    const {course, body} = req;
    const section = await course.createSection(body);
    res.status(200).send(section);
  } catch (error) {
    next(error);
  }
};
```

ДОДАТОК Ж

Структура таблиць бази даних

Ж.1 Структура таблиці users

Column Name	#	Data type	Identity	Collation	Not Null
id	1	serial4			[v]
first_name	2	varchar(255)		default	[v]
last_name	3	varchar(255)		default	[v]
password	4	varchar(255)		default	[v]
email	5	varchar(255)		default	[v]
role	6	enum_users_role			[v]
photo_path	7	varchar(255)		default	[ ]
access_token	8	text		default	[ ]
created_at	9	timestamptz			[v]
updated_at	10	timestamptz			[v]

Ж.2 Структура таблиці courses

Column Name	#	Data type	Identity	Collation	Not Null
id	1	serial4			[v]
name	2	varchar(255)		default	[v]
teacher_id	3	int4			[v]
img_path	4	varchar(255)		default	[ ]
created_at	5	timestamptz			[v]
updated_at	6	timestamptz			[v]

Ж.3 Структура таблиці sections

Column Name	#	Data type	Identity	Collation	Not Null
id	1	serial4			[v]
name	2	varchar(255)		default	[v]
course_id	3	int4			[v]
created_at	4	timestamptz			[v]
updated_at	5	timestamptz			[v]

Ж.4 Структура таблиці materials

Column Name	#	Data type	Identity	Collation	Not Null
id	1	serial4			[v]
name	2	varchar(255)		default	[v]
file_path	3	varchar(255)		default	[ ]
section_id	4	int4			[v]
created_at	5	timestamptz			[v]
updated_at	6	timestamptz			[v]

Ж.5 Структура таблиці tasks

Column Name	#	Data type	Identity	Collation	Not Null
id	1	serial4			[v]
name	2	varchar(255)		default	[v]
description	3	varchar(255)		default	[v]
image	4	varchar(255)		default	[v]
section_id	5	int4			[v]
created_at	6	timestamptz			[v]
updated_at	7	timestamptz			[v]

Ж.6 Структура таблиці answers

Column Name	#	Data type	Identity	Collation	Not Null
id	1	serial4			[v]
text	2	text		default	[v]
is_right	3	bool			[v]
task_id	4	int4			[v]
created_at	5	timestamptz			[v]
updated_at	6	timestamptz			[v]

Ж.7 Структура таблиці input_blocks

Column Name	#	Data type	Identity	Collation	Not Null
id	1	serial4			[v]
pos_x	2	float8			[v]
pos_y	3	float8			[v]
height	4	int4			[v]
width	5	int4			[v]
task_id	6	int4			[v]
created_at	7	timestamptz			[v]
updated_at	8	timestamptz			[v]