

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

**«Система з підтримки дистанційної форми
навчання для загальноосвітньої школи (розробка
предметів, завдань та оцінювання)»**

Виконала студентка групи ІПЗ-19-1
КИРИЛЮК Дарія Миколаївна

Керівник роботи:
БАКАЛЮК Тетяна Анатоліївна

Рецензент:
ЧИЖМОТРЯ Олена Геннадіївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення

Андрій МОРОЗОВ

«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КИРИЛЮК Дарія Миколаївна**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: **«Система з підтримки дистанційної форми навчання для загальноосвітньої школи (розробка предметів, завдань та оцінювання)»**,
затверджена **Наказом закладу вищої освіти від «14» лютого 2023 р., №70с**

Вихідні дані для роботи: Інформаційна система з підтримки дистанційної форми навчання для загальноосвітньої школи, а також процес розробки, та використання цієї системи.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т. А.	15 березня 2023	25 березня 2023
2	Вакалюк Т. А.	5 квітня 2023	23 квітня 2023
3	Вакалюк Т. А.	2 травня 2023	26 травня 2023

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Аналізування вимог до програмного продукту	14 лютого 2023 - 28 лютого 2023	Виконано
2	Пошук, огляд та аналіз аналогічних веб-сайтів	1 березня 2023 - 15 березня 2023	Виконано
3	Вивчення матеріалу за даною тематикою	16 березня 2023 - 25 березня 2023	Виконано
4	Планування структури програми	26 березня 2023 - 30 березня 2023	Виконано
5	Проектування та реалізація архітектури інформаційної системи	31 березня 2023 - 3 квітня 2023	Виконано
6	Написання програмного коду	3 квітня 2023 - 24 травня 2023	Виконано
7	Тестування роботи програми	25 травня 2023 - 27 травня 2023	Виконано
8	Аналіз одержаних результатів	27 травня 2023 - 29 травня 2023	Виконано
9	Оформлення пояснювальної записки	30 травня 2023 - 15 червня 2023	Виконано
10	Попередній захист	5 травня 2023	Виконано

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра на тему «Інформаційна система з підтримки дистанційної форми навчання для загальноосвітньої школи (розробка груп, предметів, завдань та оцінювання)» складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаної літератури та додатків.

Текстова частина викладена на 59 сторінках друкованого тексту.

Пояснювальна записка має 21 сторінок додатків. Список використаних джерел містить 25 найменувань і займає 2 сторінки. В роботі наведено 35 рисунків. Загальний обсяг роботи – 83 сторінок.

У першому розділі було обґрунтовано створення інформаційної системи з підтримки дистанційної форми навчання, а також проаналізовано усі вимоги.

У другому розділі було спроектовано інформаційну систему. Будову було показано на прикладі діаграми використання та діаграми класів.

У третьому розділі описано інтерфейс системи та протестовано її. Також описано порядок налаштування та встановлення параметрів системи.

Висновок містить в собі результати виконаної роботи при створенні інформаційної системи з підтримки дистанційної форми навчання для загальноосвітньої школи.

У додатку представлені технічне завдання та код розробленого програмного продукту.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, ШКОЛА, УРОК, ЗАВДАННЯ, ОЦІНКА.

					ІПЗ.КР.Б – 121 – 23 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	Інформаційна система з підтримки дистанційної форми навчання для загальноосвітньої школи (розробка предметів, завдань та оцінювання) Пояснювальна записка		Лім.	Арк.	Аркушів
Розроб.		Кирилюк Д. М.							
Перевір.		Вакалюк Т. А.						1	83
Керівник		Вакалюк Т. А.					Житомирська політехніка, група ІПЗ-19-1		
Н. контр.									
Зав. каф.		Морозов А. В.							

ABSTRACT

The explanatory note to the bachelor's thesis on the topic "Information system to support distance learning for secondary schools (development of subjects, tasks, journals and assessment)" consists of a list of abbreviations, an introduction, three chapters, conclusions, a list of references and appendices.

The text part is set out on 59 pages of printed text.

The explanatory note has 21 pages of appendices. The list of references includes 25 titles and occupies 2 pages. The note contains 35 images. The total volume of the work is 83 pages.

The first chapter justified the creation of an information system to support distance learning and analyzed all the requirements.

In the second chapter, the information system was designed. The structure was shown using the example of a use case diagram and a class diagram.

The third chapter describes the system interface and tests it. The procedure for configuring and setting system parameters is also described.

The conclusion contains the results of the work performed in creating an information system to support distance learning for a secondary school.

The appendix presents the terms of reference and the listing of the developed software product.

Keywords: INFORMATION SYSTEM, SCHOOL, LESSON, TASK, ASSESSMENT.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		2

ЗМІСТ

ВСТУП	4
РОЗДІЛ 1: АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ПЛАНУВАННЯ ТА ОБЛІКУ РОБОЧОГО ЧАСУ НА ПІДПРИЄМСТВІ	6
1.1. Постановка задачі	6
1.2. Аналіз аналогів програмного продукту	6
1.3. Вибір архітектури веб-орієнтованої системи	11
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	13
Висновки до розділу 1	16
РОЗДІЛ 2: ПРОЕКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ З ПІДТРИМКИ ДИСТАНЦІЙНОЇ ФОРМИ НАВЧАННЯ	18
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	18
2.2. Розробка бази даних системи	21
2.3. Проектування та реалізація алгоритмів роботи системи	27
2.4. Реалізація додатку веб-орієнтованої системи з підтримки дистанційної форми навчання	30
Висновки до розділу 2	40
РОЗДІЛ 3: ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ З ПІДТРИМКИ ДИСТАНЦІЙНОЇ ФОРМИ НАВЧАННЯ	42
3.1. Порядок встановлення та налаштування параметрів системи	42
3.2. Структура інтерфейсу веб-орієнтованої системи з підтримки дистанційної форми навчання	44
3.3. Тестування роботи програмного комплексу	53
Висновки до розділу 3	55
ЗАГАЛЬНІ ВИСНОВКИ	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	57
ДОДАТКИ	59

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		3

ВСТУП

Актуальність теми:

Ще в 2019 році з початком пандемії COVID-19 з'явилася велика кількість обмежень пов'язаних з безпекою та здоров'ям усіх людей, а особливо людей що працюють в сферах медицини та освіти. Це призвело до активного впровадження дистанційної форми навчання, як тимчасової міри. Але в подальшому використанні ця ідея виявилася дуже потенційною та зручною і це показало, що така форма навчання може бути не просто тимчасовим вирішенням проблеми, але й постійною частиною навчального процесу в майбутньому.

Також з початком повномасштабного вторгнення багато учнів не могли відвідувати навчальні заклади, бо були обмежені місцезнаходженням, графіком доступності електроенергії та іншими причинами. Але завдяки активній інтеграції дистанційної форми навчання в освітній процес тепер більша кількість учнів має можливість здобувати освіту в зручному для себе місці і в зручний для себе час.

Мета:

Розробка системи з підтримки дистанційної форми навчання для загальноосвітньої школи.

Завдання:

- аналіз вимог та потреб до дистанційного навчання учнів загальноосвітніх шкіл;
- визначення основних вимог до системи з підтримки дистанційної форми навчання;
- проектування архітектури системи, визначення основних компонентів та їх функцій;
- проектування та розробка бази даних за визначеною предметною областю;
- розробка інформаційної системи;

Об'єкт дослідження:

Інформаційна система з підтримки дистанційної форми навчання для загальноосвітньої школи.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

Предмет дослідження:

Процес розробки, використання та впровадження інформаційної системи з підтримки дистанційної форми навчання для загальноосвітньої школи

Методи дослідження:

- аналіз літературних джерел (аналіз наукових публікацій, статей, звітів та інших джерел, що стосуються інформаційних систем та дистанційного навчання);
- аналіз існуючих інформаційних систем (аналіз систем, які вже використовуються в загальноосвітніх школах, що дозволить оцінити їхні недоліки та переваги, та визначити принципи, які можна також використати при розробці нової системи);
- об'єктно-орієнтоване програмування (дозволяє структурувати код системи за допомогою об'єктів, які мають свої властивості і методи);
- веб-програмування (використання веб-технологій, таких як HTML, CSS і JavaScript);
- розробка на основі фреймворку (був використаний певний фреймворк, який надає готові компоненти та інструменти для розробки веб-додатків).

Структура та обсяг роботи:

Кваліфікаційна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 25 найменувань (2 стор.), 17 додатків (21 стор.), 11 таблиць та 35 ілюстрацій по тексту. Повний обсяг роботи становить 83 сторінок, з них 59 сторінок основного тексту.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

РОЗДІЛ 1: АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ПЛАНУВАННЯ ТА ОБЛІКУ РОБОЧОГО ЧАСУ НА ПІДПРИЄМСТВІ

1.1. Постановка задачі

У цьому розділі наведено завдання, які необхідно розв'язати під час розроблення та впровадження форми програмного комплексу. Основними цілями цього проекту є:

1. Реалізація функціоналу предметів (Створення структури курсів, додавання та редагування навчальних модулів, управління доступом до матеріалів користувачем з роллю “адміністратор”);
2. Реалізація функціоналу завдань (Реалізація механізму створення, організації та категоризації навчальних матеріалів вчителем);
3. Звітність та аналітика (Реалізація можливостей для формування журналів та аналізу успішності учнів з подальшою можливістю їх перегляду);
4. Зручний інтерфейс (Розробка інтуїтивно зрозумілого та зручного для використання інтерфейсу системи, що дозволить усім користувачам легко орієнтуватися, виконувати необхідні дії та знаходити необхідну інформацію);
5. Безпека даних (Заходи для захисту конфіденційності та цілісності даних, що забезпечують безпечне зберігання і передачу інформації).

1.2. Аналіз аналогів програмного продукту

Аналіз аналогів програмного продукту допомагає зрозуміти існуючі рішення та їхні можливості, що дає змогу уникнути повторення помилок інших розробників. Для порівняння було обрано такі сервіси, які надають аналогічні послуги:

- Всеосвіта [1];
- E-school [2];
- Human [3].

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

E-school

Сервіс E-school є сучасним порталом, що надає можливість користуватися електронними журналами та щоденниками.

Портал призначений для надання та надання інформації про учнівську успішність, а саме про:

- відмітки і пропуски уроків;
- нотатки до уроків;
- домашні завдання з зазначеними до виконання термінами;
- розклад занять.

Сервіс має мобільний додаток, для більш зручного доступу до усіх потрібних ресурсів. Це дуже допомагає учням в процесі навчання, так як в ньому присутній електронний щоденник. Також додаток надсилає push-повідомлення у визначених випадках та надає постійний доступ батьків до навчального процесу своєї дитини, звітів з відвідуваності навчальних закладів та контролю рівня знань.

Враховуючи, що сервіс має гриф «Рекомендовано МОН», система доступна для усіх закладів освіти незалежно від того приватні вони, чи державні. Головним приводом до впровадження даної системи в навчальний процес є прийняття позитивного рішення Педагогічною радою.

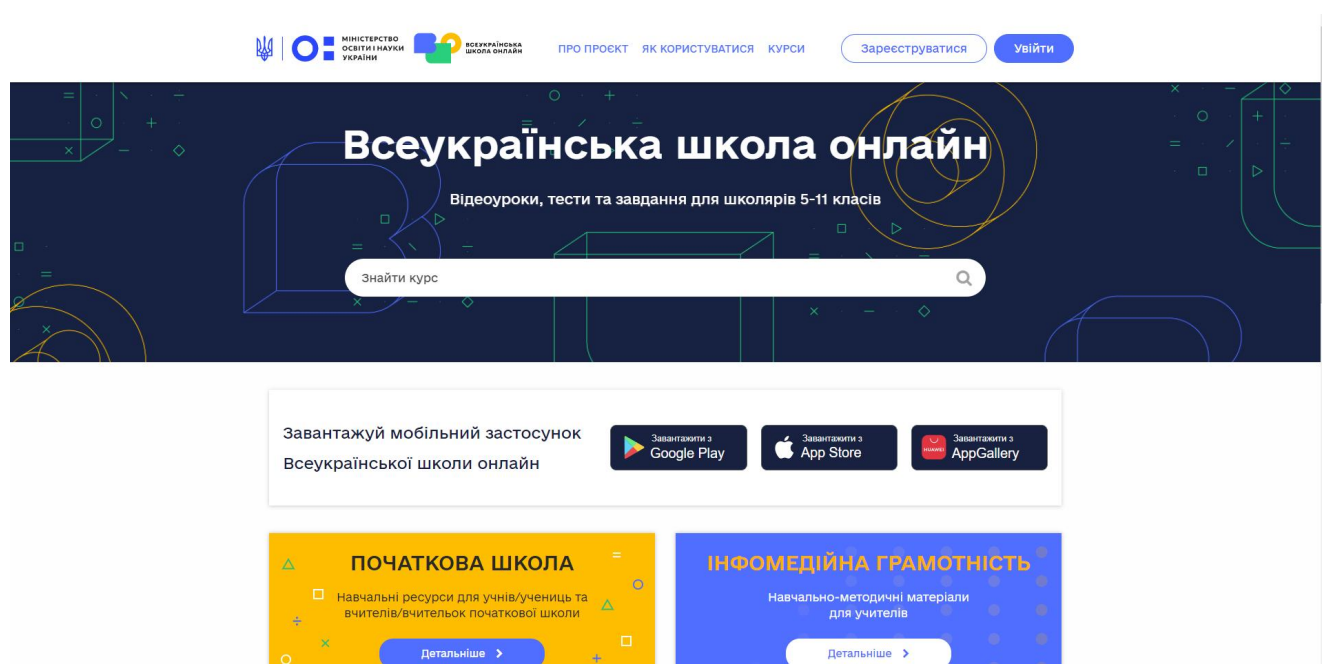


Рисунок 1.1 – Зовнішній вигляд сайту E-school

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

Переваги програми E-school:

- дуже багатий набір функцій і можливостей;
- звичний користувацький інтерфейс;
- більшість функцій безкоштовна;
- кроссплатформений.

Недоліки програми E-school:

- складна для початківців.

Всеосвіта

Сервіс «Всеосвіта» має найбільш розширений та найбільш актуальний перелік навчального матеріалу. Використовуючи даний сервіс Ви маєте можливість переглянути неповторні та рідкісні документи, якими діляться вчителі усіх профілів, що можуть Вам допомогти доволі швидко та ефективно ознайомитись з інформацією, що Ви шукаєте.

Якщо Ви вчитель та бажаєте викласти навчальний матеріал на наш сервіс, ви отримаєте посвідчення про публікацію.

На сайті «Всеосвіта» ви можете знайти різноманітні види публікацій, а саме:

- конспекти
- статті
- відеоуроки
- презентації
- тести
- Методички
- книжки.

Для вчителів даний сервіс надає можливості для планування уроків, створення тестів, організації онлайн-класів та взаємодії з учнями. Вчителі також можуть знаходити матеріали для підготовки до уроків, які було перераховано вище, обмінюватися досвідом з колегами та вести електронний журнал.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

Сайт “Всеосвіта” також має розділи, призначені для батьків, для того щоб вони могли отримувати інформацію про успішність своєї дитини, спілкуватися з вчителями та отримувати поради щодо освіти й виховання.

Завдяки своїм функціям, сервіс “Всеосвіта” покращує доступ до якісної освіти, дуже спрощує навчальний процес і успішно підтримує ефективну взаємодію між вчителями, учнями та іншими учасниками навчального процесу.

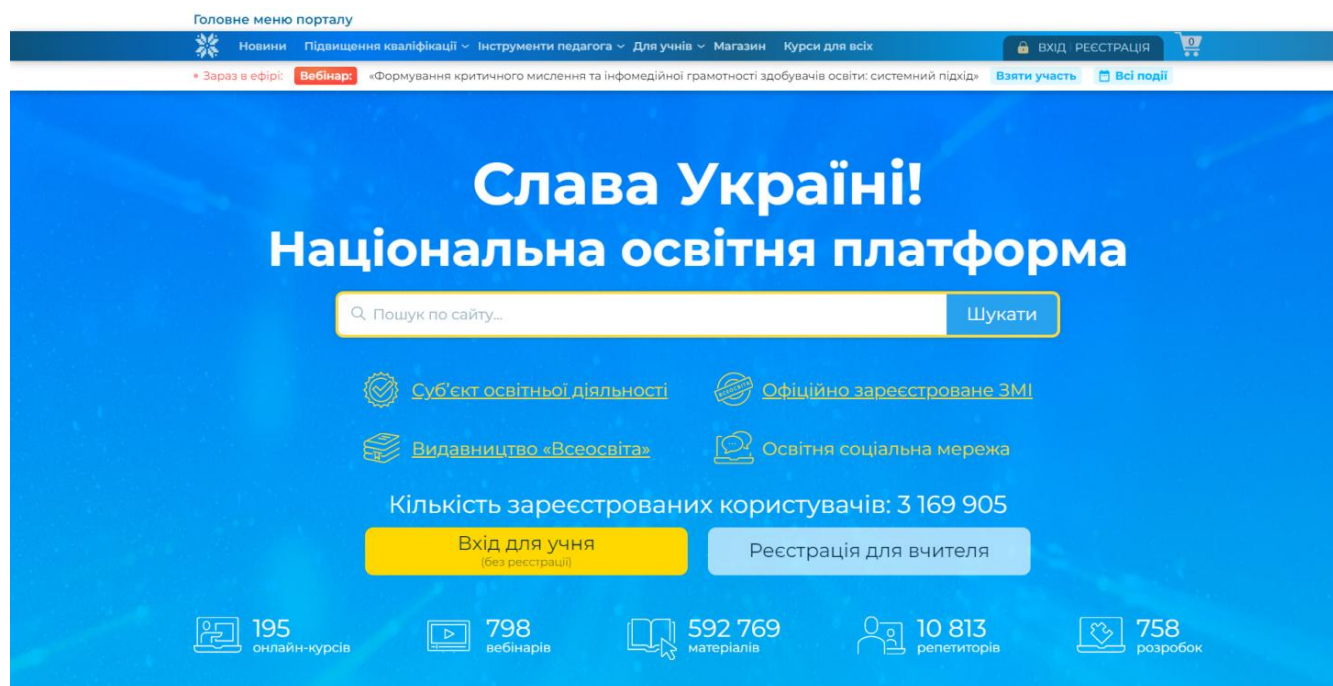


Рисунок 1.2 – Зовнішній вигляд сайту Всеосвіта

Переваги програми Всеосвіта:

- Гриф МОН України «Схвалено для використання в освітньому процесі»;
- Багато зручних функцій для користувача з PRO-акаунтом
- Є безкоштовний доступ;
- Швидкий та інтуїтивно зрозумілий користувацький інтерфейс;
- Можливість доступу до журналу одночасно з декількох пристроїв;
- Можливість експорту даних журналу;
- Можливість користуватися сервісом будучи репетитором, а не вчителем
- Користувач володіє вмістом, який створює

Недоліки програми Всеосвіта:

- Для доступу до більшої кількості функцій потрібно буде заплатити.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

Human

Human — українська компанія, що спеціалізується на онлайн освіті та розробляє продукти для цифровізації освітнього процесу в українських школах.

Основні функції сервісу включають:

- Онлайн навчання (забезпечує платформу для проведення відео-уроків та онлайн-занять. Вчителі можуть викладати матеріали, демонструвати презентації, проводити тестування та стежити за прогресом учнів);
- Електронні підручники (надає доступ до електронних підручників, що спрощує доступ до необхідного навчального матеріалу. Учні можуть вивчати матеріали та здійснювати пошук матеріалів по необхідній темі);
- Завдання та тести (вчителі можуть створювати та надсилати завдання учням через сервіс. Також доступні тести для перевірки знань учнів та збору даних про їхній прогрес);
- Спілкування (надає можливість вчителям, учням та батькам спілкуватися між собою. Вони можуть обговорювати навчальні питання та співпрацювати для досягнення зазначених навчальних цілей);
- Система відстеження прогресу (дозволяє вчителям та батькам відстежувати прогрес учнів. Вони можуть переглядати оцінки, відвідуваність та інші показники успішності).

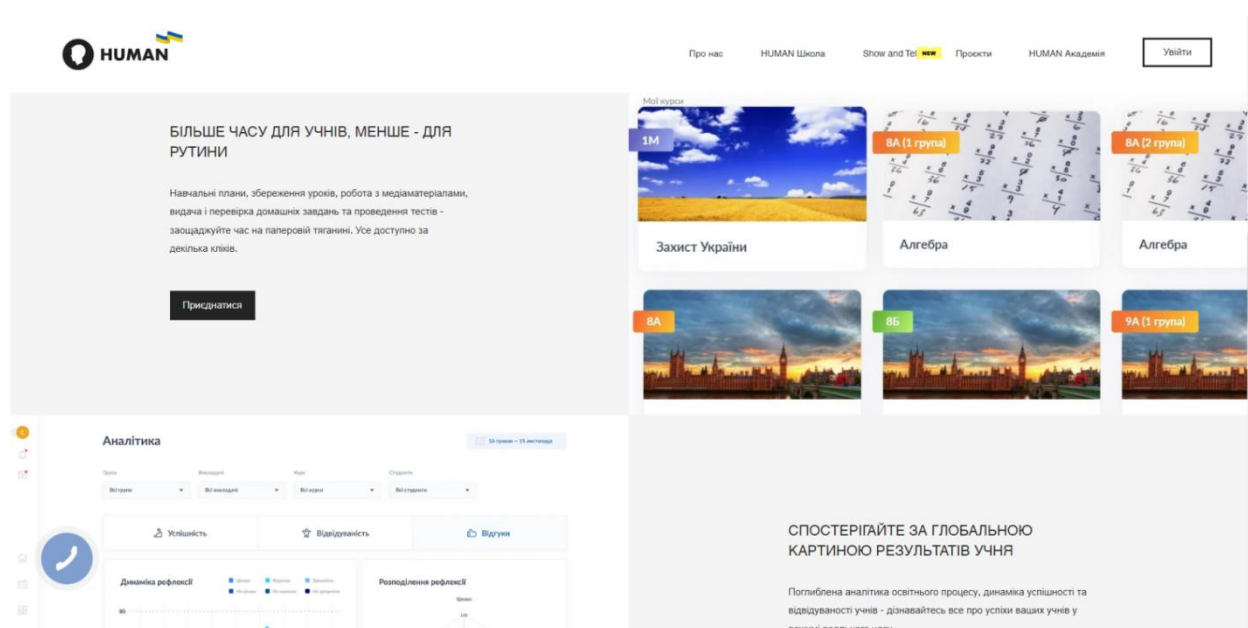


Рисунок 1.3 – Зовнішній вигляд сайту Human

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Переваги програми Human :

- підходить для вчителів та учнів з різним рівнем цифрової грамотності;
- зручний інтерфейс;
- можливість експорту в Excel;
- користувач володіє вмістом, який створює;
- кросплатформенний сервіс;

Недоліки програми Human:

- повністю платний;
- багато скарг користувачів, на помилки в роботі.

Для наочності порівняння отримані результати подані в таблиці 1.1.

Таблиця 1.1

	Простота використання	Наявність всіх необхідних функцій	Експорт	Підтримка
Всеосвіта	+ —	+	+	+
E-school	+	+ —	+	+
Human	+ —	+	+	+

1.3. Вибір архітектури веб-орієнтованої системи

Для розробки проекту було обрано структуру MVC [4].

MVC (Model-View-Controller) - це архітектурний шаблон розробки програмного забезпечення, який ділить додаток на три частини: контролер (Controller), модель (Model) та представлення (View) .

– Моделі відповідають за обробку даних та бізнес-логіку додатку. Вони забезпечують доступ до бази даних, перевірку та виконання обчислень;

– Представлення відповідають за візуалізацію даних та відображення цих даних користувачеві. В нашому випадку ці дані будуть відображатися за допомогою веб-сторінок;

– Контролери отримують вхідні дані від користувача, обробляють їх, після чого взаємодіють з моделями та представленнями. Вони контролюють логіку програми і потік даних, відповідаючи на усі вхідні запити.

Нижче наведено схему, яка візуально відображає принцип роботи архітектурного шаблону MVC.

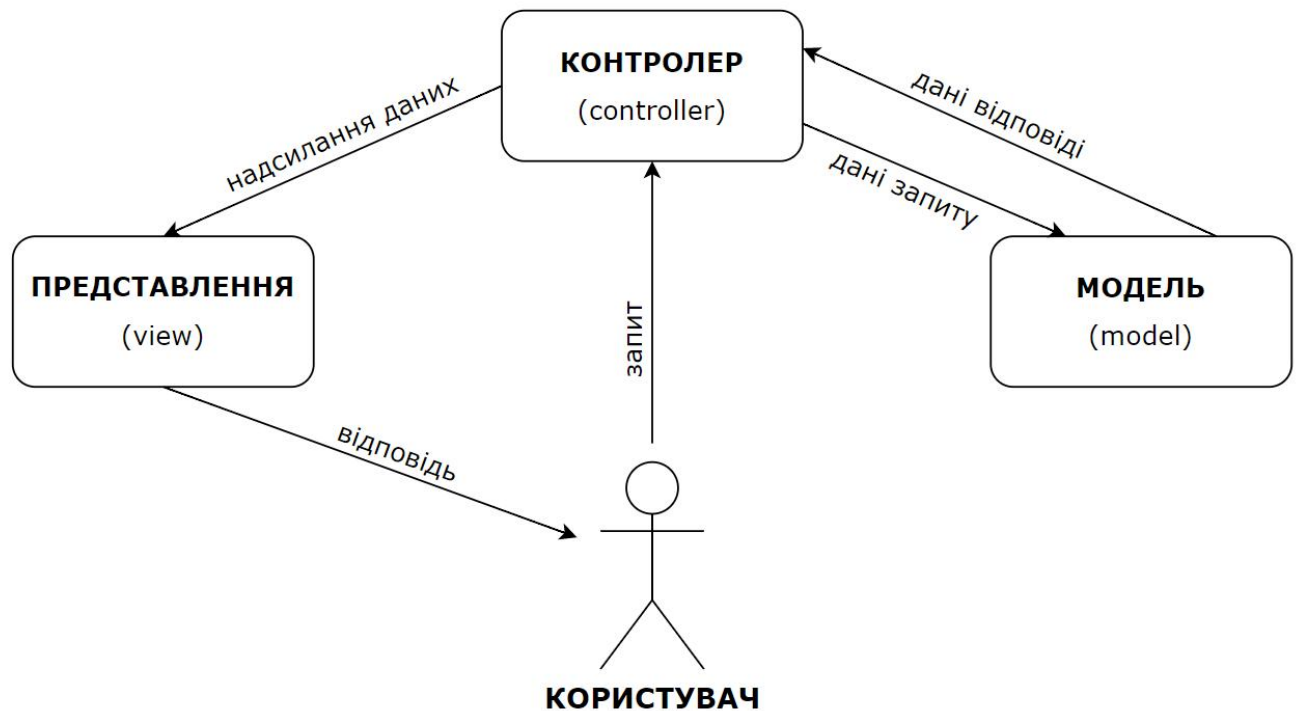


Рисунок 1.4 – Принцип роботи структури MVC

Обрання структури MVC (Model-View-Controller) при розробці додатку має декілька важливих переваг:

– Поділ відповідальностей (додаток розділяється на три основні частини, а саме на контролер, модель та представлення, що дозволяють чітко визначити ролі кожного компонента та полегшує розробку та розгортання проекту, а також подальшу підтримку коду);

– Поділ на модулі (кожна частина в MVC виконує конкретну роль і може бути легко замінена або змінена без будь-якого впливу на інші частини, що дозволяє розробникам працювати над окремими частинами додатку паралельно та забезпечує доволі велику гнучкість проекту);

- Повторне використання коду (так як логіка додатку зосереджена в контролерах і моделях, їх можна використовувати повторно в інших частинах проекту. Це набагато підвищує ефективність розробки);
- Тестування (такий архітектурний шаблон дає можливість проводити тести окремо для контролерів, моделей та представлень, що дуже допомагає знаходити та виправляти помилки);
- Можливість одночасної розробки (у структурі MVC поділ на модулі та чіткий розподіл відповідальностей між учасниками команди дозволяють кільком розробникам працювати над різними частинами проекту одночасно. Так як тема даного дипломного проекту є комплексною та виконується командою з двох людей даний архітектурний шаблон є найбільш зручним).

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Додаток має бути розроблено по принципу на клієнт-сервер та мають використовуватися технології, що не потребують додаткового ліцензування. Для розробки серверної частини додатку було обрано фреймворк Laravel [6], а для розробки клієнтської - React.JS [17]. Для коректної взаємодії клієнтської та серверної частин використовуються Vite.JS [19]

Laravel є найбільш популярним PHP-фреймворком, який базується на архітектурі MVC [5]. Він надає широкий асортимент функцій, зручний синтаксис, що допомагає швидкій та ефективній розробці веб-додатків. Laravel дає можливість використовувати готові рішення для автентифікації, маршрутизації, роботи з базою даних та ін.

React.JS є досить потужною бібліотекою JavaScript для створення користувацьких інтерфейсів. Він дозволяє розробникам будувати дуже складні веб-додатки з високою продуктивністю та можливістю повторно використовувати власно створені компоненти. Це робить розробку дуже ефективною. React.JS надає можливість, яка дозволяє оновлювати лише потрібні частини HTML-коду

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

на сторінці. React.JS забезпечує зручність та швидкість роботи з користувацьким інтерфейсом.

Використання Laravel та React.JS в одному додатку дозволяє отримати зручну та сумісну з багатьма відомими браузерами екосистему для розробки веб-додатку. Laravel надає API для взаємодії з базою даних та бізнес-логікою, а React.JS забезпечує потрібні та зручні інструменти для побудови інтерфейсу користувача. Ці два рішення разом дозволяють розробникам дуже ефективно співпрацювати та розділити функціонал додатку на частини (компоненти).

MySQL [15] було обрано в якості системи управління базами даних (СУБД) так як MySQL відомий своєю стабільністю та надійністю. Це робить його гарним вибором для проектів, які потребують безперебійної роботи та збереження даних. Для нашого додатку потрібно, щоб СУБД мала можливість працювати з великою кількістю даних та обробляти багато запитів, що надсилатимуться одночасно - це є важливим для систем з великою кількістю користувачів. Також MySQL має зрозумілий та легкий у використанні синтаксис SQL, що дуже спрощує розробку та обслуговування бази даних.

В якості системи для контролю версій було обрано Git [25]. Git дає можливість кожному з розробників мати повну копію репозиторію. Це дозволяє зручно працювати над проектом навіть без з'єднання з центральним сервером. Також він має великий набір функцій, які допомагають ефективно керувати версіями коду. Він забезпечує можливість створювати гілки (branches), об'єднувати зміни (merging), відкочуватися до попередніх версій (reverting) та ін.

Функціональні вимоги:

1. Реєстрація користувачів (Система повинна мати можливість реєстрації учнів, вчителів та адміністраторів. Кожен користувач повинен мати унікальний обліковий запис для доступу до системи);

2. Управління користувачами (Система має надавати можливість управління користувачами, включаючи створення, редагування та видалення облікових записів, а також призначення ролей та прав доступу);

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Навчальний контент (Система повинна дозволяти зберігати та керувати навчальним контентом, таким як завдання та тести);

4. Завдання та оцінювання (Система має підтримувати створення, надсилання та зберігання завдань для учнів, а також можливість їх оцінювання);

5. Статистика (Система повинна забезпечувати збір та аналіз даних про активність учнів та їх процес навчання);

6. Оцінювання (Система має надавати можливість оцінювання робіт учнів вчителями та доступу учнів до власних результатів).

Нефункціональні вимоги:

1. Інтерфейс:

- Система повинна мати зручний та інтуїтивно зрозумілий інтерфейс для учнів, вчителів та адміністраторів;
- Вона повинна бути легко використовуватися без особливих навичок.

2. Продуктивність:

- Система повинна підтримувати мінімум 500 користувачів, що одночасно працюють, пов'язаних з спільною базою даних;
- Швидке отримання результатів при здійсненні функцій програми.

3. Надійність:

- Система повинна працювати стабільно та надійно, забезпечуючи доступність для користувачів протягом установленого часу навчання;
- Вона має бути захищеною від можливих витоків даних або несанкційованого доступу.

4. Підтримка на різних пристроях:

- Система повинна бути сумісною з різними пристроями, такими як комп'ютери, ноутбуки, планшети та смартфони.

Системні вимоги:

1. Вимоги до середовища виконання:

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

Система повинна задовольняти зазначеним вимогам на комп'ютері, ноутбуці, планшеті або смартфоні в наступній мінімальній комплектації:

- підтримка браузера Google Chrome версії 45+ або
- підтримка браузера Opera версії 30+ або
- підтримка браузера Mozilla Firefox версії 38+ або
- підтримка браузера Microsoft Edge версії 12+ або
- підтримка браузера Safari версії 9+

2. Захист і безпека:

- Використання механізмів автентифікації, щоб забезпечити тільки авторизованим користувачам доступ до системи та її функцій;
- Шифрування даних, передаваних по мережі, для захисту від несанкційованого доступу та злочинних дій.

3. Вимоги до СУБД та доступу до даних:

У ядрі системи має бути представлена СУБД реляційного доступу.

Бізнес правила:

Система повинна відповідати всім стандартам побудови сучасних веб-застосунків, що використовують архітектуру SPA (single-page application) [7] додатку.

Антивимоги:

- не використовує інформацію, заборонену нормами моралі та законодавством;
- не використовує інформацію, що є недійсною або може зашкодити психіці користувачів (це дуже важливо, так як основними користувачами системи будуть неповнолітні особи).

Висновки до розділу 1

В розділі “Постановка задачі” було сформульовано основні завдання, що необхідно вирішити під час розробки інформаційної системи з підтримки дистанційної форми навчання. Поставлені задачі враховують потреби

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

користувачів та вимоги до функціональності системи. Цей розділ визначає основні напрямки роботи для подальшої розробки системи.

У розділі “Аналіз аналогів програмного продукту” було проведено огляд і дослідження існуючих аналогів інформаційних систем з підтримки дистанційної форми навчання. Аналіз дав змогу виявити переваги та недоліки існуючих рішень, їх функціональні можливості та особливості. Цей розділ надав важливу інформацію для подальшого проектування та розробки нашої інформаційної системи.

У розділі “Вибір архітектури веб-орієнтованої системи” було зроблено висновок, що архітектурою, що найбільш підходить для нашого проекту є архітектура MVC (Model-View-Controller). Обрана архітектура дозволить ефективно організувати робочий процес, розділяючи його на частини (компоненти), що буде допомагати кращій підтримці, зрозумілості коду та допоможе при розробці в команді.

У розділі “Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення” було зроблено висновок, що для даного проекту найбільше підходить PHP-фреймворк Laravel для серверної частини і React для клієнтської частини. Laravel забезпечує широкий функціонал для розробки веб-додатків та безпеку використовуваних даних. React, надає потужні можливості для розробки зручного користувацького інтерфейсу та є дуже ефективним інструментом для створення динамічних компонентів сторінки, що є важливим так як дана система є “single-page” додатком. Обрані інструментальні засоби дозволять розробити ефективну та сучасну інформаційну систему з підтримки дистанційної форми навчання.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2: ПРОЕКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ З ПІДТРИМКИ ДИСТАНЦІЙНОЇ ФОРМИ НАВЧАННЯ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Після детального аналізу вимог для зображення бізнес-логіки, було побудовано діаграму використання (Use Case Diagram) [10]. Кольорами позначено частини роботи різних учасників команди.

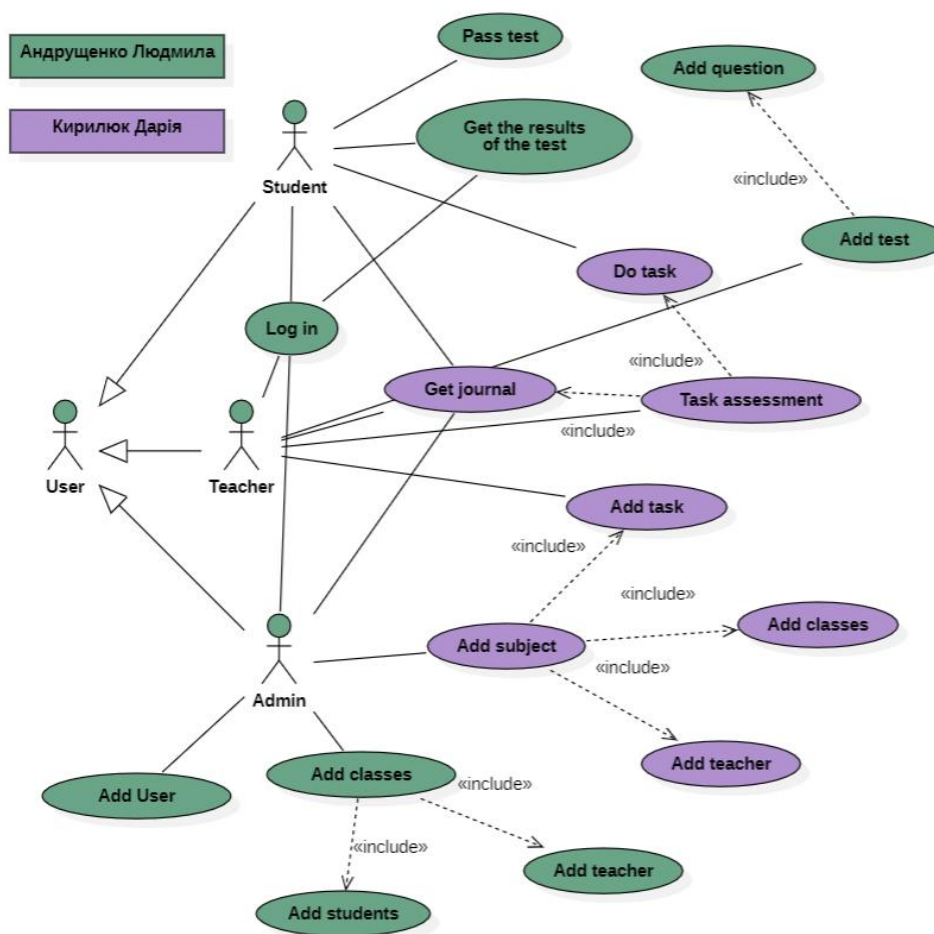


Рисунок 2.1 – Діаграма використання

Діаграми варіантів використання є графічним інструментом моделювання, який допомагає визначити, описати та візуалізувати різноманітні варіанти взаємодії між акторами (користувачами) та системою. Вони зображують, як користувачі взаємодіють з системою для досягнення поставлених цілей.

Діаграма варіантів використання допомагає визначити основні функціональні можливості та те, як актори взаємодіють з цими можливостями. Також вона дозволяє зрозуміти, як система повинна працювати з точки зору користувачів та які взаємодії відбуваються між ними. В нашому випадку система передбачає три види користувачів, а саме учня, вчителя та адміністратора.

Для учня має бути доступний функціонал перегляду списку предметів, які йому призначив адміністратор. Також учень повинен мати можливість для перегляду завдань призначених вчителем, виконання їх в особистому кабінеті та перегляду своєї оцінки за дане завдання в журналі.

Для вчителя має бути доступний функціонал перегляду предметів, які йому призначив адміністратор, створення та перегляду завдань. Також вчитель повинен мати можливість оцінювати виконані учнями завдання та переглядати виставлені оцінки в журналі.

Для адміністратора має бути доступний той самий функціонал що й для вчителя. Окрім цього він повинен мати можливість створення предметів та призначення їх класам та вчителям.

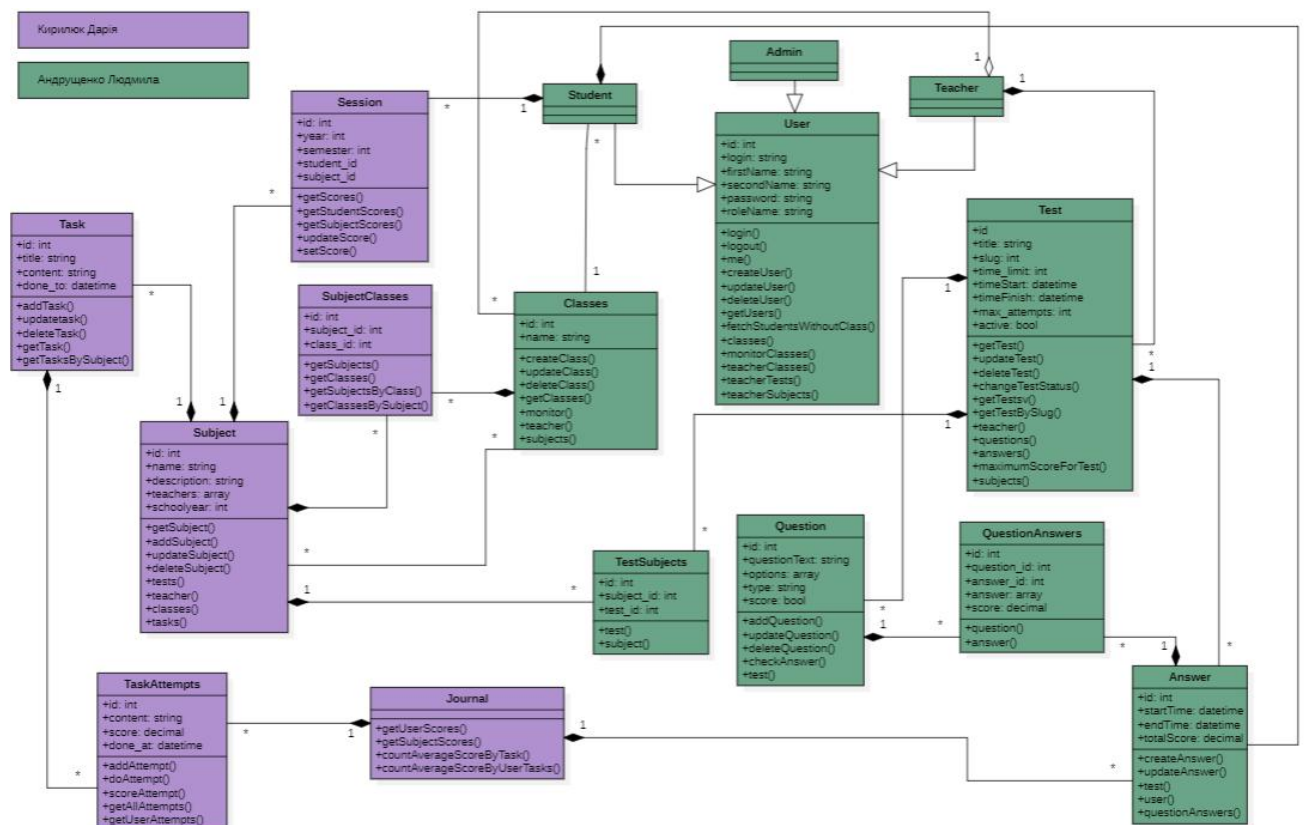


Рисунок 2.2 – Діаграма класів

Об'єктно-орієнтовану модель системи прийнято розглядати у вигляді діаграми класів [12]. Діаграма класів є важливим інструментом в об'єктно-орієнтованому програмуванні. Вона потрібна для відображення структури класів та взаємозв'язків між ними. Така діаграма надає графічне відображення класів, їх атрибутів та методів, а також зв'язків між цими класами.

Діаграма класів дозволяє показати архітектуру системи і виявити потенційні проблеми в проєкті ще на етапі розробки. Вона допомагає програмістам краще організовувати код (присутня можливість повторного використання компонентів), а також забезпечує зрозумілість системи.

Нижче наведено опис класів, а також їх атрибутів та методів:

Клас Subject передбачає роботу з основною інформацією про певний навчальний предмет та містить в собі вчителя, який викладає цей предмет та групи учнів, які його будуть вивчати. Учень може отримати всі предмети, які призначені адміністратором класу до якого він належить. Вчитель може отримати всі предмети, які він буде викладати. Адміністратор має доступ до усіх предметів.

Клас Task призначений для роботи з завданнями, які пов'язані з предметом і надає можливість додавати, редагувати і видаляти завдання. Також за допомогою методів даного класу можна отримати всі завдання, що пов'язані з заданим уроком. Отримані завдання потрібні для відображення їх на сторінці предмету а також в журналі.

Клас TaskAttempt передбачає роботу з оцінками за завдання, що вже додано систему користувачем з роллю “вчитель”. Методи даного класу дозволяють виконувати завдання учнем та вчителю оцінювати виконані учнем завдання. Методи даного класу використовуються для відображення даних про кожну окрему спробу виконання завдання учнем на сторінці завдання, а також для відображення оцінок за завдання в журналі.

Клас SubjectClasses передбачає роботу з предметами та класами, яким ці предмети призначено. Учням та вчителям цей клас дає можливість отримувати інформацію про предмети призначені адміністратором. Адміністратор в свою чергу може призначати ці предмети класам учнів та вчителям.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Клас *Journal* передбачає роботу з оцінками учнів за певні завдання та тести. Цей клас призначений для отримання, обрахунку та відображення даних про оцінки учнів. Адміністратор має можливість отримати інформацію про оцінки усіх учнів. Вчитель може отримати інформацію про оцінки в класах, де він викладає, а також про класи в яких його призначено класним керівником. Учень в свою чергу може отримати свої оцінки за завдання з предметів, які йому було попередньо призначено адміністратором.

Клас *Session* передбачає роботу з семестровими оцінками учнів за предмети. Він дає можливість доступу адміністратору до усіх семестрових оцінок. Вчитель має можливість виставлення цих семестрових оцінок, а учень в свою чергу може побачити ці семестрові оцінки на сторінці профілю.

2.2. Розробка бази даних системи

Для зберігання необхідних даних додатку була обрана система для управління базами даних MySQL. Для управління таблицями було використано міграції, які дає можливість використовувати PHP-фреймворк Laravel.

Laravel міграції - це потужний інструмент, що допомагає управляти структурою бази даних у Laravel-проектах. Вони дозволяють визначати та зберігати зміни в структурі бази даних у вигляді коду, що робить процес міграцій більш контрольованим та зручним.

Основна перевага Laravel міграцій - це їх зручність. За допомогою міграцій можна створювати, змінювати або видаляти таблиці та поля бази даних використовуючи безпосередньо код всередині проекту. Дана функція є дуже корисною при роботі в команді, так як кожен учасник може відтворити необхідну базу даних на локальному чи віддаленому сервері просто однією командою в терміналі.

Крім того, Laravel міграції забезпечують можливість повернення до попередньої версії бази даних (за допомогою відкату міграцій), що дозволяє відстежувати та контролювати зміни в структурі бази під час розробки проекту.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

Вони також підтримують розширення бази даних, індекси, створення зв'язків між таблицями та інші функції.

Реалізована база даних складається з 19 таблиць, які містять у собі всі дані для роботи програми [16].

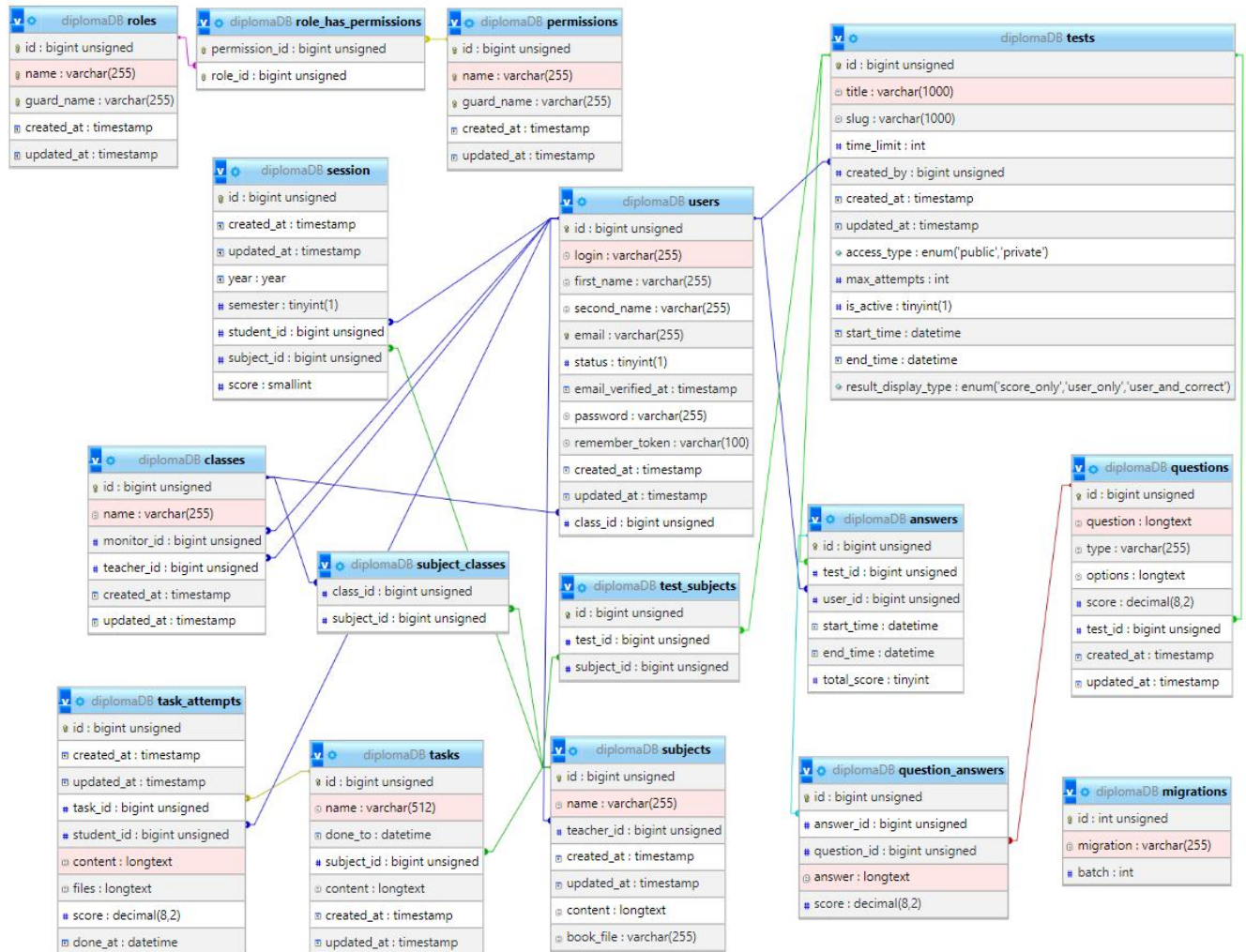


Рисунок 2.3 – Структура бази даних

Найменування та призначення таблиць, що використовувалися для розробки даної частини проекту представлено у таблиці 2.1.

Таблиця 2.1

Назва таблиці	Призначення таблиці
users	Таблиця містить дані про користувачів системи
subjects	Таблиця містить дані про всі предмети

subject_classes	Таблиця містить дані про те якому класу призначені які предмети
tasks	Таблиця містить дані про завдання для відповідних предметів
task_attempts	Таблиця містить дані про спроби виконання учнем завдань
session	Таблиця містить дані про семестрові оцінки учнів
migrations	Таблиця зберігає усі виконані міграції
answers	Таблиця містить дані про результати проходження тестів учнями

Нижче, на таблицях 2.2 – 2.10 наведено опис основних таблиць бази даних.

Таблиця 2.2 – Таблиця users (користувачі)

Назва	Тип даних	PK	FK	Опис поля
id	bigint unsigned	+	–	Ідентифікатор користувача
login	varchar(255)	+	–	Логін користувача
first_name	varchar(255)	–	–	Ім'я користувача
second_name	varchar(255)	–	–	Прізвище користувача
email	varchar(255)	–	–	Електронна пошта користувача
status	tinyint(1)	–	–	Статус користувача
email_verified_at	timestamp	–	–	Час верифікації електронної пошти
password	varchar(255)	–	–	Хеш паролю користувача
remember_token	varchar(100)	–	–	Токен, якщо користувач обрав “запам’ятати мене” при авторизації
created_at	timestamp	–	–	Час створення користувача

Продовження Таблиці 2.2

updated_at	timestamp	–	–	Час останнього оновлення інформації про користувача
class_id	bigint unsigned	–	+	Ідентифікатор класу

Таблиця 2.3 – Таблица subjects (предмети)

Назва	Тип даних	PK	FK	Опис поля
id	bigint unsigned	+	–	Ідентифікатор предмету
name	varchar(255)	–	–	Назва предмету
second_name	varchar(255)	–	–	Прізвище користувача
teacher_id	bigint unsigned	–	+	Ідентифікатор вчителя, що викладає урок
book_file	varchar(255)	–	–	Розташування файлу з книгою
content	longtext	–	–	Інформація про предмет
created_at	timestamp	–	–	Час створення предмету
updated_at	timestamp	–	–	Час останнього оновлення інформації про предмет

Таблиця 2.4 – Таблица classes (класи)

Назва	Тип даних	PK	FK	Опис поля
id	bigint unsigned	+	–	Ідентифікатор класу
name	varchar(255)	–	–	Назва класу
monitor_id	bigint unsigned	–	+	Ідентифікатор старости класу
teacher_id	bigint unsigned	–	+	Ідентифікатор класного керівника
created_at	timestamp	–	–	Час створення класу
updated_at	timestamp	–	–	Час останнього оновлення інформації про клас

Таблиця 2.5 – Таблиця subject_classes (предмет-клас)

Назва	Тип даних	РК	FK	Опис поля
id_subject	bigint unsigned	–	+	Ідентифікатор предмету
id_class	bigint unsigned	–	+	Ідентифікатор класу

Таблиця 2.6 – Таблиця tasks (завдання)

Назва	Тип даних	РК	FK	Опис поля
id	bigint unsigned	+	–	Ідентифікатор завдання
name	varchar(255)	–	–	Назва завдання
done_to	timestamp	–	–	Час до якого треба здати завдання
subject_id	bigint unsigned	–	+	Ідентифікатор предмета
content	longtext	–	–	Текст завдання
created_at	timestamp	–	–	Час створення завдання
updated_at	timestamp	–	–	Час останнього оновлення інформації про завдання

Таблиця 2.7 – Таблиця task_attempts (спроби завдання)

Назва	Тип даних	РК	FK	Опис поля
id	bigint unsigned	+	–	Ідентифікатор спроби
task_id	timestamp	–	+	Ідентифікатор завдання
student_id	bigint unsigned	–	+	Ідентифікатор учня
done_at	timestamp	–	–	Час здачі завдання
score	decimal	–	–	Оцінка за виконане завдання
content	longtext	–	–	Текст відповіді учня
created_at	timestamp	–	–	Час створення спроби
updated_at	timestamp	–	–	Час останнього оновлення інформації про спробу

Таблиця 2.8 – Таблиця session (семестрові оцінки)

Назва	Тип даних	РК	FK	Опис поля
id	bigint unsigned	+	–	Ідентифікатор оцінки
year	smalint	–	–	Рік виставлення
semester	tinyint	–	–	Семестр
student_id	bigint unsigned	–	+	Ідентифікатор учня
subject_id	bigint unsigned	–	+	Ідентифікатор предмету
score	decimal	–	–	Оцінка за семестр
created_at	timestamp	–	–	Час створення оцінки
updated_at	timestamp	–	–	Час останнього оновлення інформації про оцінку

Таблиця 2.9 – Таблиця magrations (міграції)

Назва	Тип даних	РК	FK	Опис поля
id	int unsigned	+	–	Ідентифікатор предмету
migration	varchar(255)	–	–	Назва міграції
batch	int	–	–	Черга групи міграцій

Таблиця 2.10 – Таблиця answers (результати проходження тестів)

Назва	Тип даних	РК	FK	Опис поля
id	bigint unsigned	+	–	Ідентифікатор спроби проходження тесту
user_id	bigint unsigned	–	+	Ідентифікатор учня
test_id	bigint unsigned	–	+	Ідентифікатор тесту
start_time	datetime	–	–	Час початку проходження тесту
end_time	datetime	–	–	Час завершення проходження тесту
total_score	tiny integer	–	–	Оцінка за тест

2.3. Проектування та реалізація алгоритмів роботи системи

Реалізацію алгоритмів роботи системи прийнято зображати у вигляді діаграми діяльності [8].

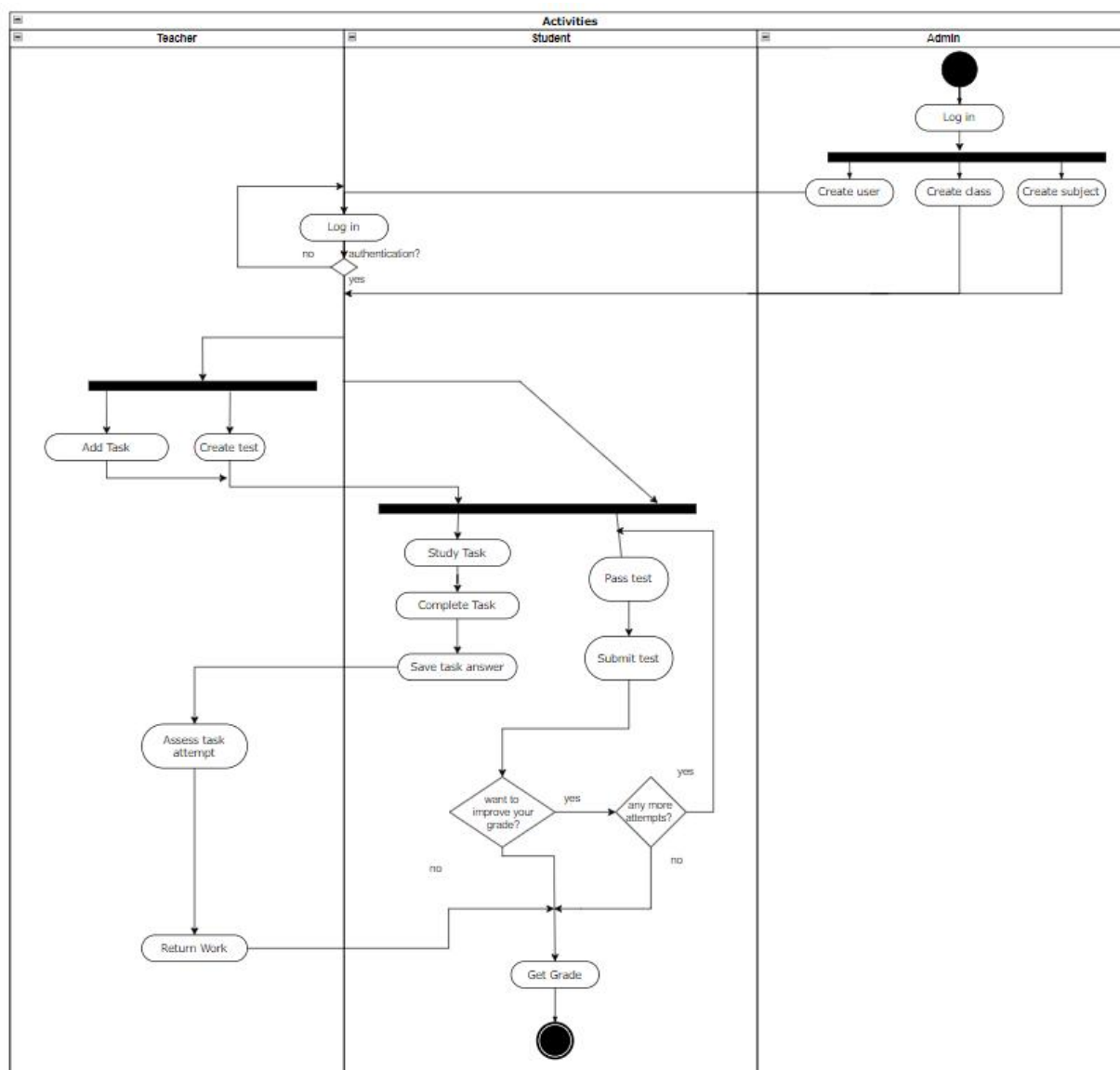


Рисунок 2.4 - Загальна діаграма діяльності

Діаграма діяльності [11] є графічним інструментом для зображення процесів та послідовності дій у системі.

Діаграма діяльності складається з активностей, які представляють окремі дії або кроки в процесі, та з ребер, що з'єднують активності та визначають їх послідовність. Вона дозволяє зображати послідовність дій, умовні розгалуження та одночасну роботу декількох активностей та залежності між ними.

Діаграма діяльності є важливим інструментом в проектуванні, вона допомагає розуміти потреби учасників команди, моделювати процеси, а також знаходити можливості для змін в роботі програми та додавання нових функцій.

Так як в системі передбачено акторів з трьома ролями (адміністратор, вчитель та учень) на Рисунку 2.4 можна побачити стандартний сценарій того, як відбуваються активності під час роботи з програмою.

Розглянемо більш детально алгоритм створення завдання та виконання його учнем. Діаграма процесу створення завдання та його виконання зображена на Рисунку 2.5.

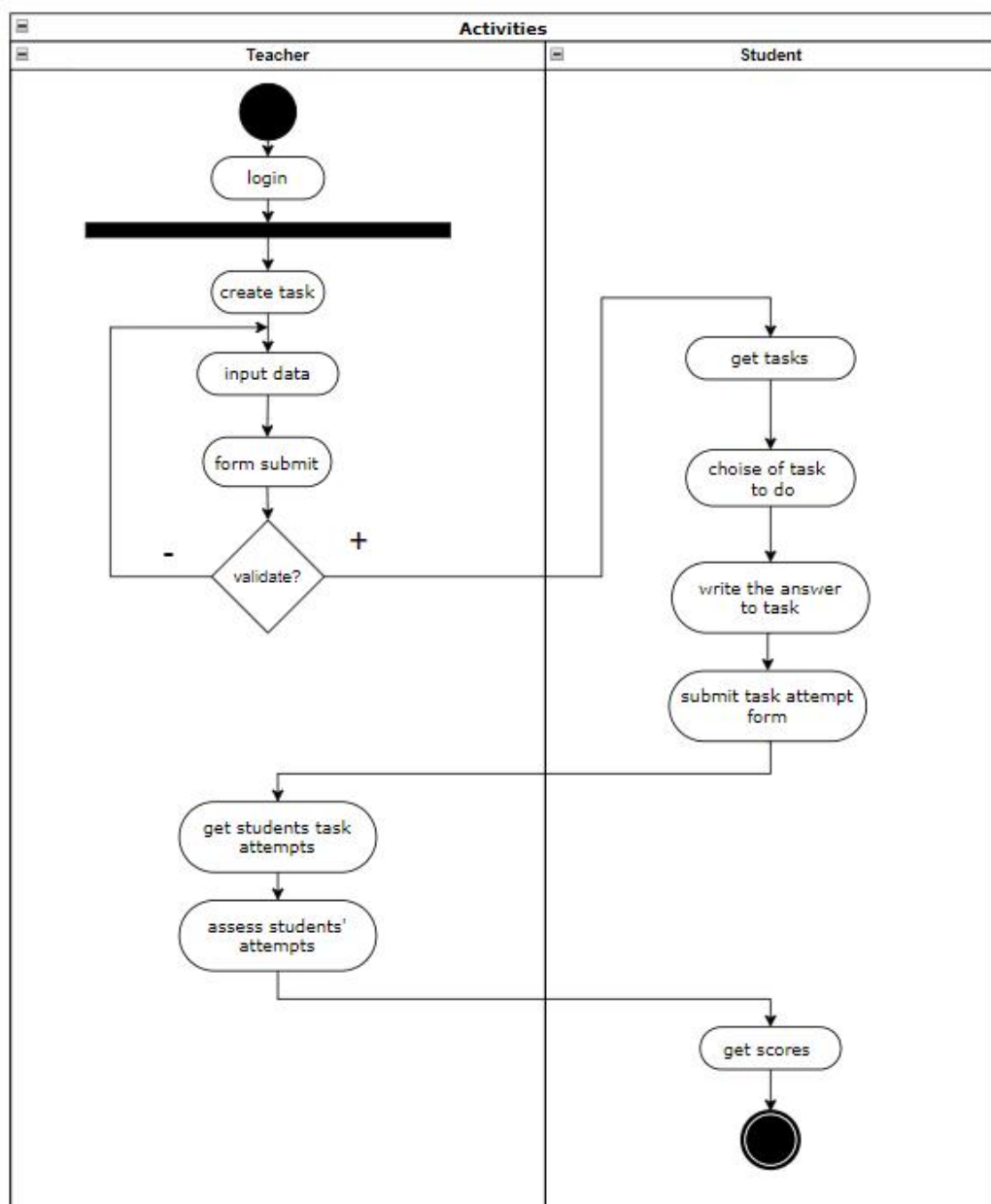


Рисунок 2.5 - Діаграма діяльності (процес створення завдання та його виконання)

Опис діаграми діяльності (створення предмету та завдання):

1. Авторизація користувача
2. Відображення форми додавання завдання
3. Ввід необхідних даних
4. Відправка форми
5. Валідація
 - Якщо ТАК, то продовжуємо виконання
 - Якщо НІ, то повертаємося на форму
6. Учень переглядає свої завдання
7. Обирає, завдання для виконання
8. Виконує завдання, та відправляє свою відповідь вчителю
9. Вчитель переглядає усі виконані учнями завдання
10. Обирає спробу, для того щоб оцінити
11. Оцінює завдання
12. Учень отримує оцінку, та переглядає її в журналі

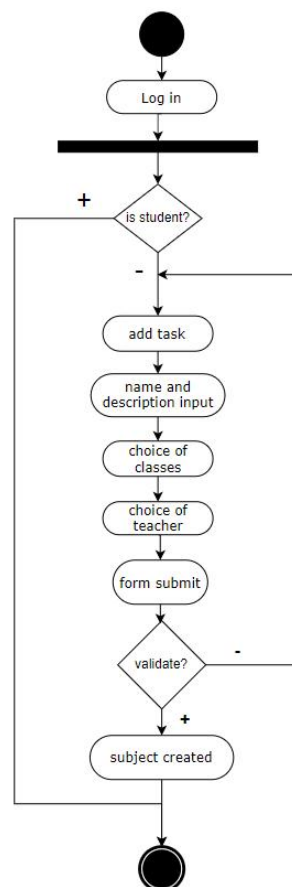


Рисунок 2.6 - Діаграма діяльності (процес створення предмету)

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

Розглянемо більш детально ще один алгоритм, а саме створення предмету адміністратором. Діаграма процесу створення предмету зображена на Рисунку 2.6.

Опис діаграми діяльності (створення предмету та завдання):

1. Авторизація користувача
2. Перевірка чи користувач є студентом
 - Якщо НІ, то продовжуємо
 - Якщо ТАК, завершуємо виконання
3. Відображення форми додавання предмету
4. Ввід назви предмету та його опису
5. Вибір класів, яким призначено даний предмет
6. Відправка форми
7. Валідація
 - Якщо ТАК, то зберігаємо
 - Якщо НІ, то повертаємося на форму
8. Збереження даних

2.4. Реалізація додатку веб-орієнтованої системи з підтримки дистанційної форми навчання

Для роботи з базою даних використовується PHP Artisan [14]. PHP Artisan - це інтерфейс командного рядка, що постачається разом з фреймворком Laravel. За допомогою PHP Artisan можна створювати та мігрувати бази даних, генерувати шаблони коду, розгортати сервер, генерувати ключі для шифрування, налаштовувати середовище та ін.

Розглянемо частину проекту для взаємодії з предметами.

В першу чергу за допомогою міграцій створюється таблиця “subjects”. В даному класі, що наслідується від Illuminate\Database\Migrations\Migration має бути описано два методи up() та down() для запуску та відкату міграцій відповідно.

Фрагмент коду:

```
public function up(): void
{
    Schema::create('subjects', function (Blueprint $table) {
```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

```

        $table->id();
        $table->string('name');
        $table->unsignedBigInteger('teacher_id')->nullable();
        $table->foreign('teacher_id')->references('id')->on('users')->onDelete()-
>cascadeOnUpdate();
        $table->timestamps();
    });
}
public function down(): void
{
    Schema::dropIfExists('subjects');
}

```

Так як один предмет може бути призначений декільком класам та одному класу може бути призначено декілька предметів потрібна ще одна таблиця “subject_classes”.

Фрагмент коду:

```

public function up(): void
{
    Schema::create('subject_classes', function (Blueprint $table) {
        $table->unsignedBigInteger('class_id')->nullable();
        $table->foreign('class_id')->references('id')->on('classes')->cascadeOnUpdate()-
>cascadeOnDelete();
        $table->unsignedBigInteger('subject_id')->nullable();
        $table->foreign('subject_id')->references('id')->on('subjects')->cascadeOnUpdate()-
>cascadeOnDelete();
    });
}
public function down(): void
{
    Schema::dropIfExists('subject_classes');
}

```

Після цього для виконання міграції в консолі виконується команда `php artisan migrate`.

Для взаємодії з даними таблиці “subjects” створено модель Subject. Це клас, що наслідується від `Illuminate\Database\Eloquent\Model`. В методах моделі описано зв’язки поточної моделі з іншими. Модель Subject взаємодіє з моделями User (дані про вчителя, що викладає предмет), Classes (класи яким призначено предмет), Tasks (завдання створені в предметі) і Tests (тести, що відповідають предмету).

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

Фрагмент коду:

```
protected $guarded = [];  
public function teacher()  
{  
    return $this->belongsTo(User::class, 'teacher_id');  
}  
public function classes()  
{  
    return $this->belongsToMany(Classes::class, 'subject_classes', 'subject_id', 'class_id');  
}  
public function tasks(){  
    return $this->hasMany(Task::class, 'subject_id');  
}  
public function tests()  
{  
    return $this->belongsToMany(Test::class, 'test_subjects');  
}
```

Для взаємодії з даними таблиці “subject_classes” створено модель SubjectClass. Модель SubjectClass взаємодіє з моделями Classes (клас, якому призначено предмет), Subject (предмет, який призначено класу).

Фрагмент коду:

```
protected $guarded = [];  
public function classes()  
{  
    return $this->belongsTo(Classes::class, 'class_id');  
}  
public function subject()  
{  
    return $this->belongsTo(Subject::class, 'subject_id');  
}
```

SubjectResource є класом, що наслідується від класу Illuminate\Http\Resources\Json\JsonResource. Цей клас використовується для форматування даних моделі Subject перед їх відправкою у відповідь HTTP-запиту. SubjectResource надає зручну можливість для контролю структури даних, що повертаються. Він дозволяє вибирати, включати або виключати необхідні поля моделі Subject. SubjectResource відокремлює логіку форматування даних від контролерів, що спрощує розробку системи. Для роботи з даними потрібно перевизначити метод батьківського класу toArray().

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

Фрагмент коду:

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'teacher' => $this->teacher,
        'classes' => $this->classes,
        'tasks' => $this->tasks,
        'created_at' => Carbon::parse($this->created_at)->isoFormat('DD MMMM, YYYY HH:mm'),
        'updated_at' => Carbon::parse($this->updated_at)->isoFormat('DD MMMM, YYYY HH:mm'),
    ];
}
```

StoreSubjectRequest є класом, що наслідується від класу Illuminate\Foundation\Http\FormRequest. Цей клас використовується для валідації та обробки даних, що надсилаються при створенні нового предмету в системі.

Клас StoreSubjectRequest визначає правила валідації для полів запиту та забезпечує перевірку на коректність та типи даних, а також забезпечує зручний доступ до обробки помилок валідації. Для роботи з даними потрібно перевизначити методи батьківського класу authorize() (перевірка, чи авторизований користувач) та rules() (правила валідації даних).

Фрагмент коду:

```
public function authorize(): bool
{
    return true;
}

public function rules(): array
{
    return [
        'name' => 'required|string|max:255',
        'teacher_id' => [
            'nullable',
            'exists:users,id',
            function ($attribute, $value, $fail) {
                $user = User::find($value);
                if (!$user->hasRole('teacher')) {
                    $fail("Помилка. " . $user->first_name . " . $user->second_name . " не є вчителем.");
                }
            }
        ],
    ];
}
```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        'classes_ids' => 'nullable|array',
        'classes_ids.*' => [
            'exists:classes,id',
        ],
    ];
}

```

Клас UpdateSubjectRequest є ще одним класом, що наслідується від класу Illuminate\Foundation\Http\FormRequest. Він використовується для валідації та обробки даних, що надсилаються при оновленні існуючого предмету в системі.

Фрагмент коду:

```

public function authorize(): bool
{
    return true;
}

public function rules(): array
{
    return [
        'name' => 'required|string|max:255',
        'teacher_id' => [
            'exists:users,id'
        ],
        'classes_ids' => 'nullable|array',
        'classes_ids.*' => [
            'exists:classes,id'
        ],
    ];
}

```

Контролер SubjectsController (клас, що наслідується від App\Http\Controllers) відповідає за обробку HTTP-запитів, пов'язаних з предметами. Він містить набір методів, які виконують різноманітні дії пов'язані з предметами, такі як створення нового предмета, отримання списку предметів, оновлення інформації про предмет та його видалення.

Метод index() призначений для отримання даних про предмети. Залежно від ролі поточного користувача, він має отримувати доступ до різного переліку предметів. Адміністратор - до всіх, вчитель - до тих, що він викладає, а учень до тих, які призначив адміністратор класу до, якого він належить.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Фрагмент коду:

```
public function index()
{
    $user = auth()->user();
    if ($user->hasRole('admin')) {
        $subjects = Subject::query()->orderBy('updated_at', 'desc')->get();
    } else if ($user->hasRole('teacher')) {
        $subjects = $user->teacherSubjects()->orderBy('name', 'desc')->get();
    } else if ($user->hasRole('student')) {
        $ids = SubjectClass::where('class_id', $user['class_id'])->pluck('subject_id')->toArray();
        $subjects = Subject::whereIn('id', $ids)->orderBy('updated_at', 'desc')->get();
    } else {
        $subjects = [];
    }
    return SubjectResource::collection($subjects);
}
```

Метод `store()` призначений для створення нових предметів. Він тримує дані що пройшли валідацію за допомогою `StoreSubjectRequest`. Після чого додає новий запис в таблицю “subjects” та один, або декілька нових записів в таблицю “subject_classes”.

Фрагмент коду:

```
public function store(StoreSubjectRequest $request)
{
    $data = $request->validated();
    $cls = $data['classes_ids'];
    $subject = Subject::create($data);
    $class = new SubjectClass;
    foreach ($cls as $c)
    {
        $class::insert(['subject_id' => $subject['id'], 'class_id' => $c]);
    }
    return response(new SubjectResource($subject), 201);
}
```

Метод `update()` призначений для оновлення існуючих в системі предметів. Він тримує дані що пройшли валідацію за допомогою `UpdateSubjectRequest`, а також об’єкт класу `Subject`, що буде оновлюватися. Після чого редагує існуючий запис в таблиці “subjects” та зберігає нові записи про класи, яким призначено даний предмет в таблицю “subject_classes”.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

Фрагмент коду:

```
public function update(UpdateSubjectRequest $request, Subject $subject)
{
    $data = $request->validated();
    $cls = $data['classes_ids'];
    $subject->update($data);
    $clss = new SubjectClass;
    $clss::where('subject_id', $subject['id']->delete());
    foreach ($cls as $c)
    {
        $clss::insert(['subject_id' => $subject['id'], 'class_id' => $c]);
    }
    return response(new SubjectResource($subject), 201);
}
```

Метод `destroy()` призначений для видалення існуючих в системі предметів. Він отримує об'єкт класу `Subject`, що буде видалено.

Фрагмент коду:

```
public function destroy(Subject $subject)
{
    $subject->delete();
    return response()->json(['message' => "Subject deleted successfully"], 200);
}
```

Для виконання запитів пов'язаних з предметами, користувач має бути авторизованим. Для перевірки цього використовується проміжне програмне забезпечення (middleware) `App\Http\Middleware\Authenticate`. Для подальшого використання його зазначено у файлі `Kernel.php` у змінній `$middlewareAliases`, яка зберігає усі middleware, що використовуються в даній системі.

Фрагмент коду:

```
protected $middlewareAliases =
[
    'auth' => \App\Http\Middleware\Authenticate::class,
    // інші middleware
];
```

Файл `api.php` призначений для налаштування маршрутизації в проекті. Він містить маршрути, які використовуються для визначення API-шляхів та їхнього обробки. У файлі `api.php` можна вказати різні маршрути, які прослуховують певні

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

URL-шляхи та відповідають необхідним контролерам або функціям для обробки запитів. Для зазначення маршрутів для роботи з предметами, використано вищезазначений middleware.

Фрагмент коду:

```
Route::middleware('auth:sanctum')->group(function () {  
    Route::apiResource('/subjects', SubjectsController::class);  
  
    // інші маршрути  
});
```

Для реалізації передачі даних з серверної частини на клієнтську використовується Axios [21]. Axios - це бібліотека JavaScript, яка дозволяє здійснювати HTTP-запити з середовища Node.js. Вона надає зручний і простий спосіб взаємодії зі зовнішніми API для отримання даних з сервера.

Основна перевага Axios полягає в його простоті використання. Він підтримує всі основні методи HTTP, такі як GET, POST, PUT, DELETE, і надає можливість налаштування різних параметрів запиту, таких як заголовки, параметри URL, тіло запиту та ін.

Для роботи з Axios, його потрібно налаштувати на клієнтській частині. Для цього створено файл axios-client.js в якому створено об'єкт класу axios та надано можливість його експорту.

Фрагмент коду:

```
const axiosClient = axios.create({  
    baseURL: `${import.meta.env.VITE_API_BASE_URL}/api`  
})  
axiosClient.interceptors.request.use((config) => {  
    const token = localStorage.getItem('ACCESS_TOKEN')  
    config.headers.Authorization = `Bearer ${token}`  
    return config;  
})  
axiosClient.interceptors.response.use((response) => {  
    return response;  
}, (error) => {  
    try {  
        const {response} = error;  
        if (response.status === 401) {  
            localStorage.removeItem('ACCESS_TOKEN');  
        }  
    }  
})
```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

```

    } catch (e) {
      console.error(e);
    }
    throw error;
  })

export default axiosClient;

```

Файл subjectsSlice.js використовується для роботи з axios запитами на сервер за допомогою Redux Toolkit [22]. Він містить початковий стан (initialState), основні редуктори (reducers) та дії (extra reducers), які викликаються для виконання запитів на сервер.

В файлі subjectsSlice.js визначаються декілька дій, таких як getSubjects (запит на отримання списку предметів), createSubject (запит на створення нового предмету), updateSubject (запит на оновлення існуючого предмету) та deleteSubject (запит на видалення предмету). Кожна дія виконує відповідний axios запит на сервер і оновлює стан за допомогою відповідного редуктора.

Так як функції для отримання, створення, оновлення майже ідентичні, нижче приведено приклад коду для виконання запиту на оновлення предмету.

Фрагмент коду:

```

export const updateSubject = createAsyncThunk("subjects/updateSubject",
  async ({id, values}, {rejectWithValue}) => {
    try {
      const res = await axiosClient.put(`/subjects/${id}`, values);
      return res.data;
    } catch (error) {
      if (!error.response)
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  });

```

Для виконання запитів використовується createAsyncThunk, який автоматично генерує три події: pending (запит до сервера), fulfilled (успішне отримання даних з сервера) і rejected (помилка під час виконання запиту).

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

Фрагмент коду:

```
builder.addCase(updateSubject.pending, (state) => {
  state.isLoading = true;
});

builder.addCase(updateSubject.fulfilled, (state, action) => {
  state.status = 'succeeded';
  state.subjects = state.subjects.map(subjectItem => subjectItem.id === action.payload.id ?
action.payload : subjectItem)
  state.visibleData = state.subjects;
});

builder.addCase(updateSubject.rejected, (state, action) => {
  state.status = 'rejected';
  state.errors = action.payload;
});
```

Після отримання необхідних даних з серверу, ці дані використовуються в представленнях (view) для відображення їх користувачу. Для переходу на необхідне представлення використовується React Router [18].

Фрагмент коду:

```
const router = createBrowserRouter([
  {
    path: '/',
    element: <DefaultLayout/>,
    children: [
      {
        path: '/subjects',
        element: <SubjectsList/>
      },
      // інші маршрути
    ]
  }
]);
```

Для отримання необхідних даних зі стану Redux використовується метод `useSelector()`. Після отримання дані можна вивести в `jsx`-файлі. Нижче наведено приклад виводу усіх предметів на відповідній сторінці за допомогою методу `map()`, так як отримані дані є масивом. Для зручності було використано бібліотеку MUI (Material User Interface) [24]. Для зручного відображення форм та їх валідації було використано Formik [20].

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

Фрагмент коду:

```
export default function SubjectsList()
{
  const {subjects} = useSelector((state) => state.subjects)
  // інший код
  return (
    <Grid container spacing={2}>
      {subjects.map((subject) => (
        <Grid key={subject.id}>
          <SubjectCard subjectItem={subject}/>
        </Grid>
      ))}
    </Grid>
  )
}
```

Висновки до розділу 2

В розділі “Визначення варіантів використання та об’єктно-орієнтованої структури системи” було сформовано діаграму використання. За допомогою діаграми використання були ідентифіковані основні актори системи, такі як учні, вчителі, адміністратори, а також визначено різні сценарії використання системи, такі як створення предметів, створення та призначення завдань, оцінювання робіт тощо. Ця діаграма дозволяє краще розуміти, як користувачі будуть взаємодіяти з системою та які можливості вони матимуть. Також була сформована діаграма класів, яка відображає структуру системи з підтримки дистанційної форми навчання та взаємозв’язки між класами. Ця діаграма надає уявлення про основні компоненти системи, їх атрибути та методи, а також про залежності між класами.

В розділі “Розробка бази даних системи” була проведена робота з розробки бази даних для системи з підтримки дистанційної форми навчання. Було визначено необхідні сутності, їх атрибути та зв’язки. Ця база даних забезпечує збереження, організацію та доступ до важливої інформації, необхідної для функціонування системи. Розроблена база даних є гнучкою, масштабованою та піддається розширенню в майбутньому, що сприяє ефективній роботі системи та задоволенню потреб користувачів.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

В розділі “Проектування та реалізація алгоритмів роботи системи” було створено три діаграми діяльності, а саме загальну діаграму діяльності програми. Діаграма діяльності “Виконання завдань та оцінювання”, яка показує послідовність дій, які студент повинен виконати для виконання завдань, які призначені в рамках предмету. Це включає перегляд завдання, його виконання, завантаження результатів та оцінювання викладачем. Ця діаграма допомагає краще управляти процесом виконання завдань та забезпечує чітку взаємодію між студентами та викладачами. Діаграма діяльності “Створення предмету” відображає послідовність кроків, які потрібно виконати для успішного створення нового предмету у системі. Це включає визначення назви предмету та вибір відповідних параметрів.

В розділі “Реалізація додатку веб-орієнтованої системи” було проведено докладний опис процесу розробки та реалізації системи на прикладі роботи з предметами. Під час реалізації було використано різноманітні інструменти та технології, такі як міграції бази даних, API-маршрутизація, використання Redux для керування станом додатку, взаємодія з сервером за допомогою Axios та інші. Були створені різні компоненти, контролери та запити для обробки даних і забезпечення функціональності системи.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

РОЗДІЛ 3: ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ З ПІДТРИМКИ ДИСТАНЦІЙНОЇ ФОРМИ НАВЧАННЯ

3.1. Порядок встановлення та налаштування параметрів системи

Система повинна задовольняти зазначеним вимогам на комп'ютері, планшеті або смартфоні в наступній мінімальній комплектації (підтримка хоча б одного з браузерів):

- Google Chrome версії 45 або вище;
- Opera версії 30 або вище;
- Mozilla Firefox версії 38 або вище;
- Microsoft Edge версії 12 або вище;
- Safari версії 9 або вище.

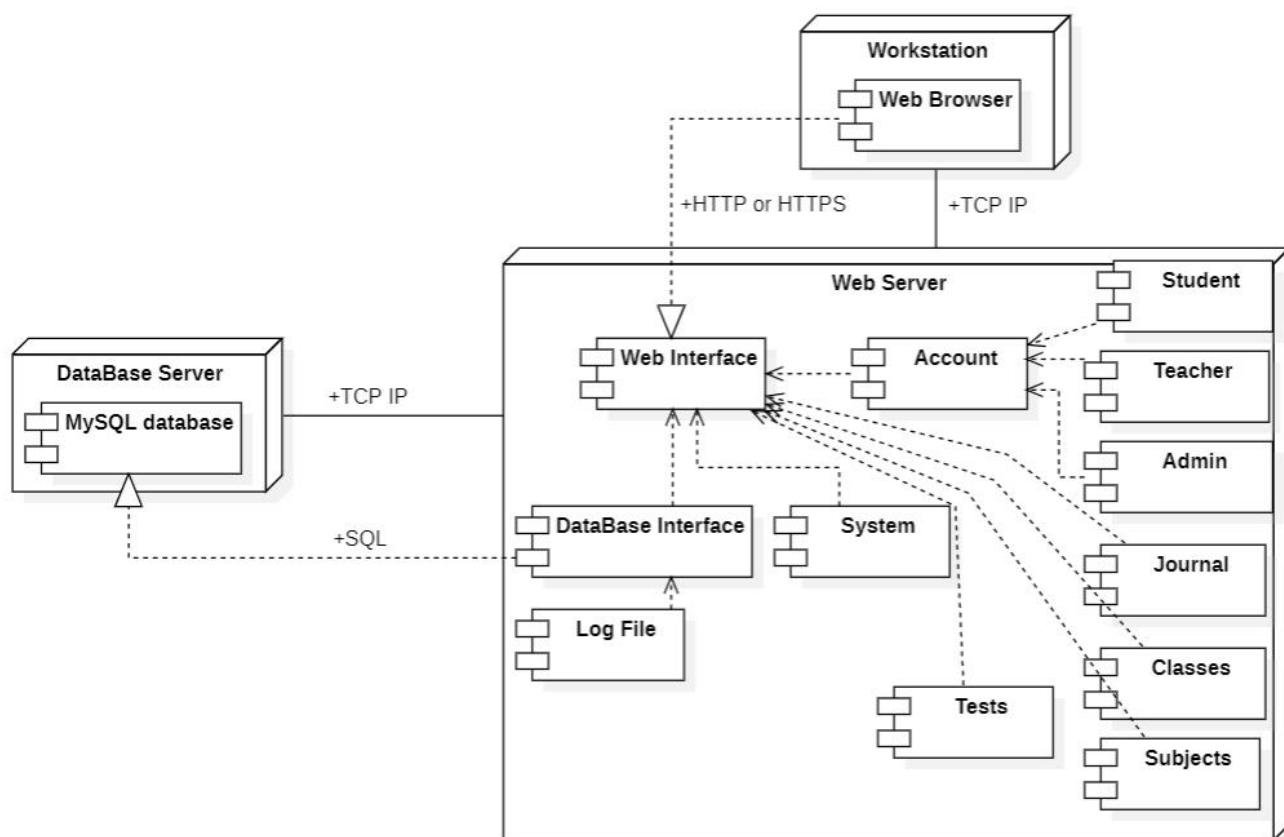


Рисунок 3.1 - Діаграма розгортання

Діаграма розгортання [13] системи відображає фізичну архітектуру системи та взаємодію між її частинами (компонентами). Вона включає в себе сервери, комп'ютери, мережеві та інші пристрої, які використовуються для забезпечення функціонування системи. Ця діаграма показує, які компоненти системи розгорнуті на фізичних пристроях та як вони взаємодіють друг з другом.

У діаграмі розгортання системи з підтримки дистанційної форми навчання зображено такі основні компоненти:

- База даних (це місце збереження даних системи, таких як інформація про користувачів, предмети, завдання, оцінки тощо. База даних може бути розгорнута на окремому сервері);
- Робоча станція (клієнтський пристрій, який використовується користувачами для доступу до системи. Це може бути комп'ютер, ноутбук, планшет або смартфон. На цих пристроях запускаються веб-браузери, через які користувачі взаємодіють з системою);
- Веб-сервер (фізичний сервер, на якому розміщена основна логіка системи. Він відповідає за обробку запитів, управління базою даних та надання необхідних ресурсів для веб-додатків).

При більш детальному розгляді веб-серверу можна зрозуміти, що цей сервер виконує важливі функції, такі як обробка запитів веб-клієнтів, розрахунок та генерація відповідей, доступ до бази даних, збереження стану додатку та керування необхідними ресурсами. Він забезпечує взаємодію між користувачами та системою, обробляючи їх запити і надаючи необхідні дані. Також сервер включає механізми автентифікації та валідації даних при додаванні нових об'єктів у систему таких як предмети, завдання та оцінки.

Axios використовує протокол HTTP для здійснення запитів до сервера. HTTP (Hypertext Transfer Protocol) є протоколом, що використовується для передачі даних між веб-клієнтом (браузером) та веб-сервером. Axios може використовувати як HTTP, так і HTTPS (захищений HTTP) для забезпечення безпеки передачі даних через шифрування.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

3.2. Структура інтерфейсу веб-орієнтованої системи з підтримки дистанційної форми навчання

Структура інтерфейсу системи з підтримки дистанційної форми навчання включає різні сторінки, які забезпечують користувачам доступ до предметів, завдань та журналів. Основні сторінки цієї системи включають:

1. Сторінка предметів (на цій сторінці користувач може переглядати список доступних предметів. Кожен предмет може мати відповідну інформацію, таку як назва, опис, викладач тощо. Користувач може обрати певний предмет для подальшої роботи);

2. Сторінка завдань (ця сторінка відображає список завдань, пов'язаних з обраним предметом. Кожне завдання має заголовок, опис, дату створення та інші деталі. Користувач може переглядати завдання, редагувати їх, додавати нові або відправляти виконані завдання);

3. Сторінка журналу (ця сторінка містить журнал, в якому зберігається інформація про виконані завдання та оцінки учнів. Користувач може переглядати свій особистий журнал).

В залежності від ролі авторизованого користувача однакові сторінки, відображаються по різному. Нижче наведено детальний опис сторінок та можливих дій на них для користувачів з різними ролями.

Гість

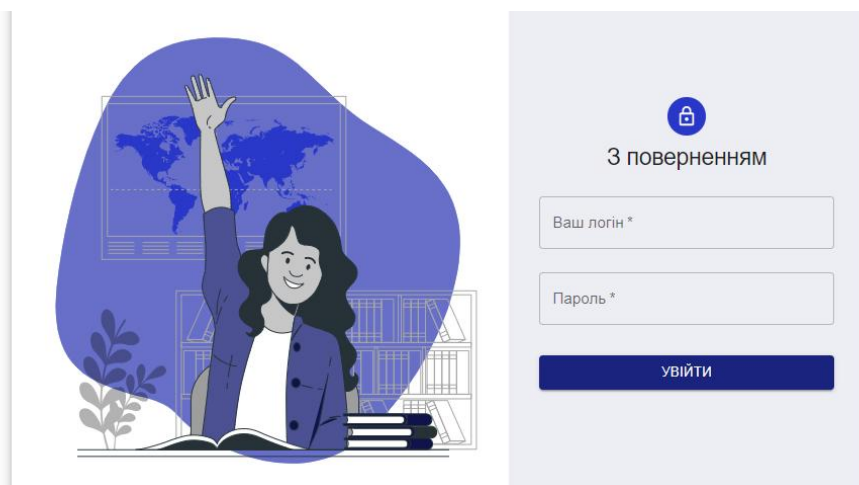


Рисунок 3.2 - Сторінка автентифікації

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

На даній сторінці є можливість автентифікації гостя. Для цього потрібно ввести логін та пароль користувача, попередньо наданий адміністратором. Дана форма перевіряє чи існує такий користувач, і при успішному вході перенаправляє користувача на головну сторінку сервісу. Зліва на головній сторінці користувач може побачити головне меню навігації вкладками системи.

Адміністратор

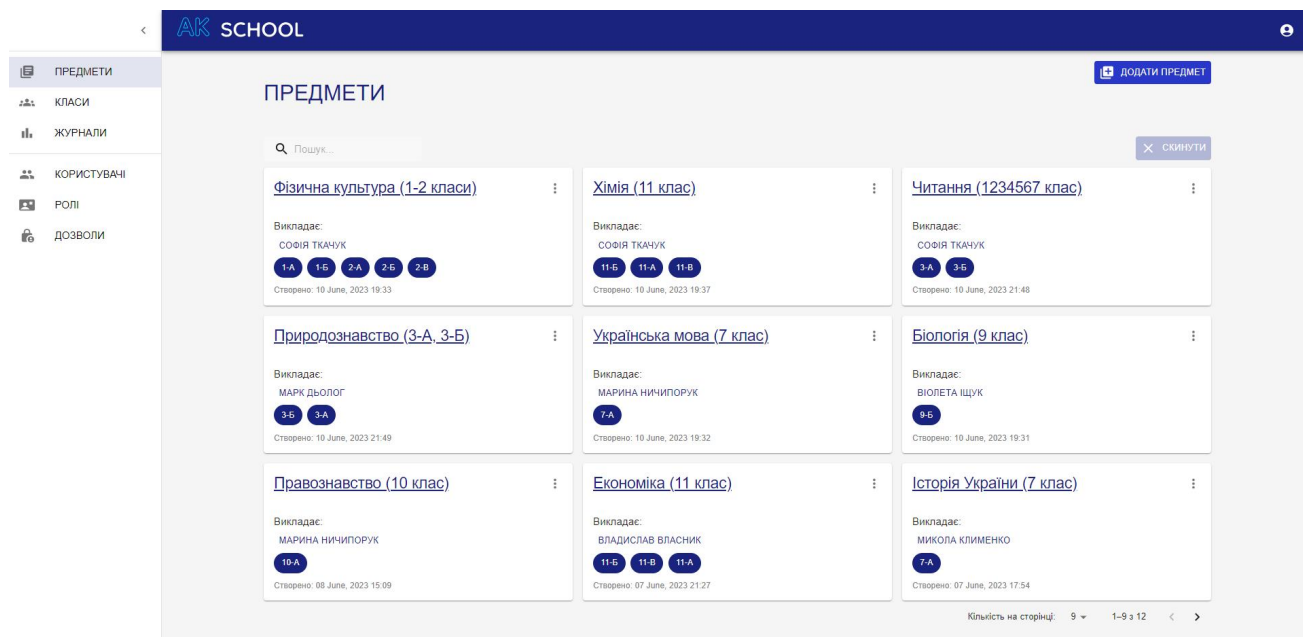


Рисунок 3.3 - Сторінка зі списком предметів

На даній сторінці відображаються усі предмети, які є у системі. Також тут присутня можливість пошуку необхідного предмету, та відображення по сторінкам.

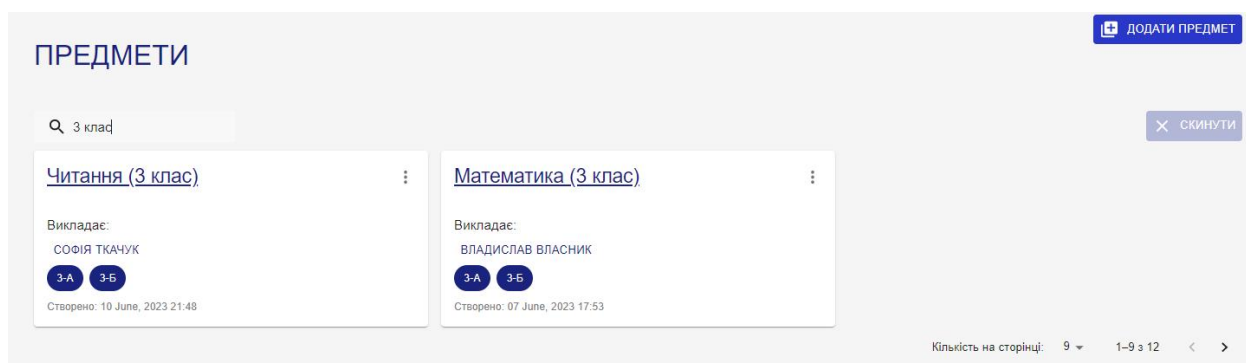


Рисунок 3.4 - Пошук предметів

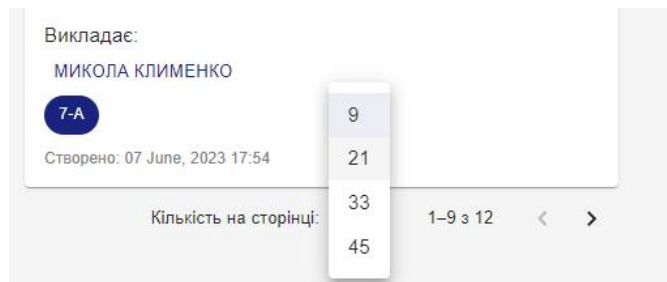


Рисунок 3.5 - Вибір кількості предметів, що відображаються на сторінці

Натиснувши на кнопку “додати предмет” для адміністратора буде відображено форму для додавання нового предмету в систему.

Рисунок 3.6 - Форма додавання нового предмету

На кожній картці предмету є меню, що дозволяє відредагувати, або видалити існуючий предмет.

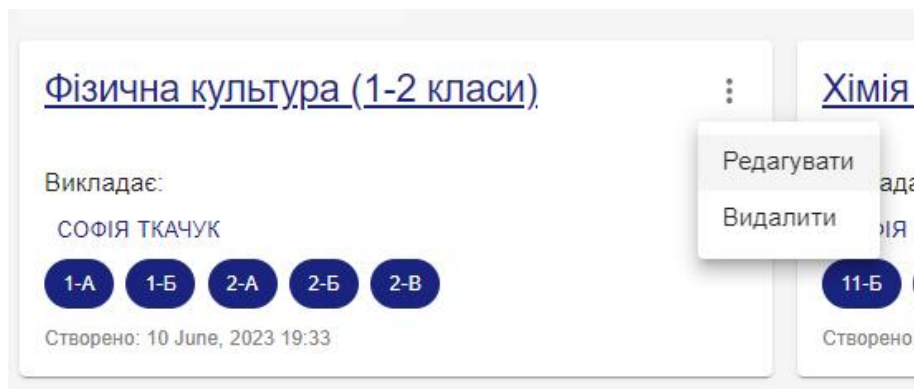


Рисунок 3.7 - Меню предмету

Для редагування предмету потрібно натиснути на кнопку “Редагувати”. Після чого адміністратору відобразиться форма редагування предмету.

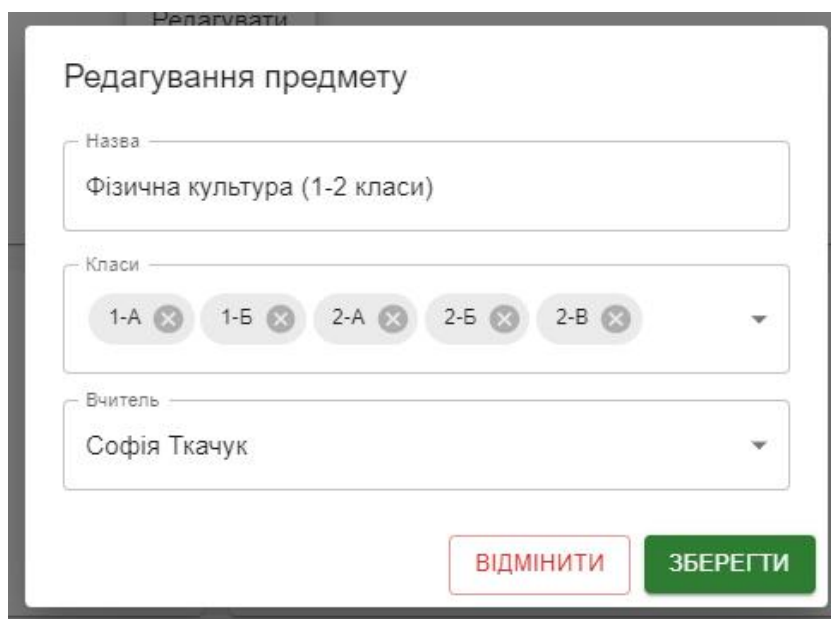


Рисунок 3.8 - Форма редагування предмету

Вчитель

Для переходу на сторінку з завданнями користувачу потрібно натиснути на назву предмету.

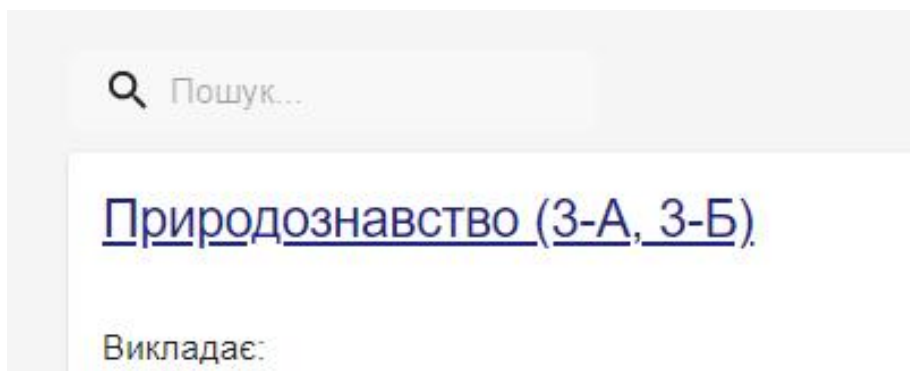


Рисунок 3.9 - Назва предмету на картці

Після натискання відобразиться сторінка зі списком завдань, що вже створено для даного предмету. На ній можна побачити картки завдань з інформацією про них, а саме датою створення завдання, датою останнього оновлення та кількістю учнів, що виконали дане завдання.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

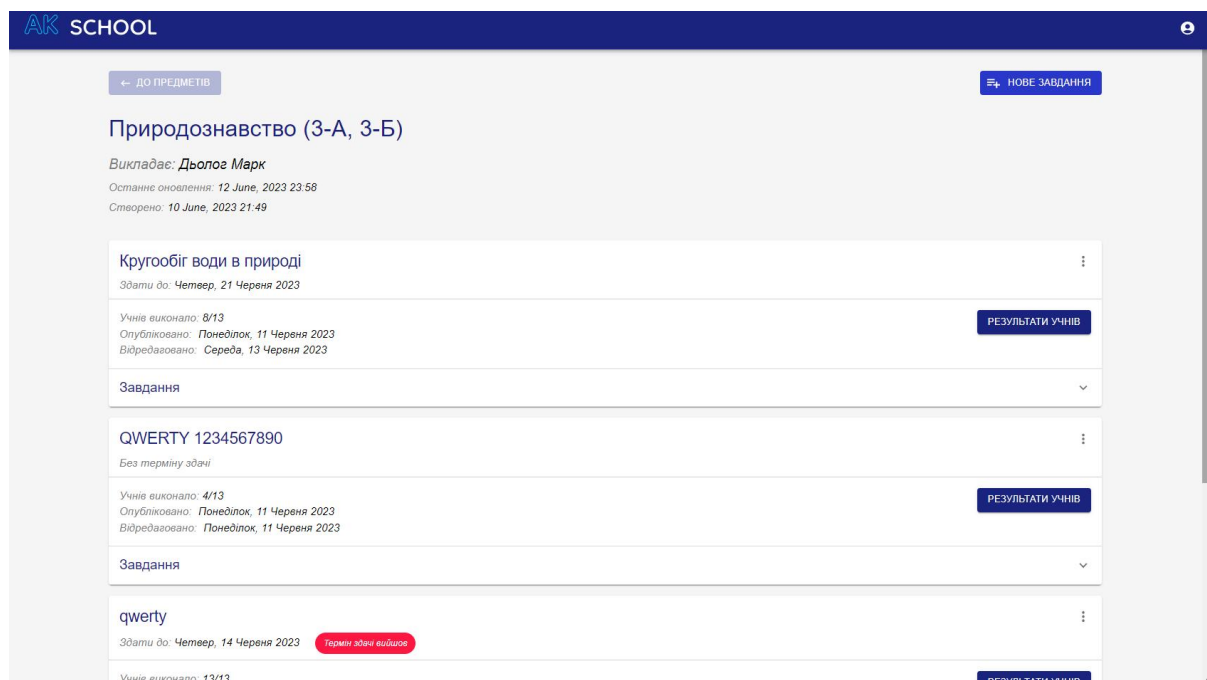


Рисунок 3.10 - Сторінка зі списком завдань

Кожну картку завдання можна розгорнути, щоб побачити контент даного завдання. Для цього потрібно натиснути на кнопку “Завдання” знизу картки.

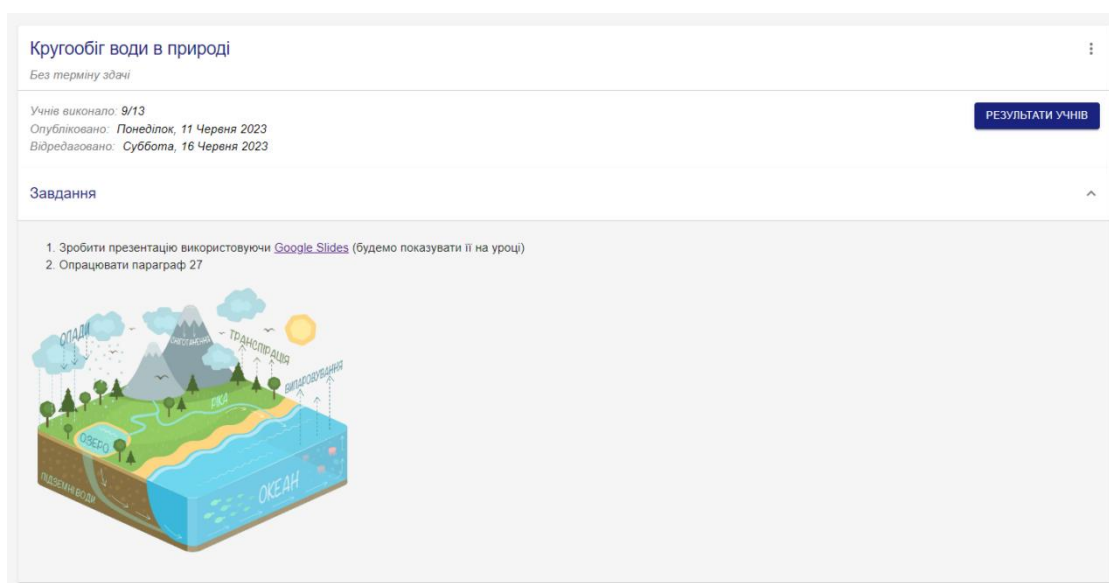


Рисунок 3.11 - Картка завдання

Для додавання нового завдання користувачу потрібно натиснути кнопку “Додати завдання” зверху сторінки. Після чого відобразиться форма створення нового завдання. Для вводу контенту завдання використовується текстовий редактор Quill [23].

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

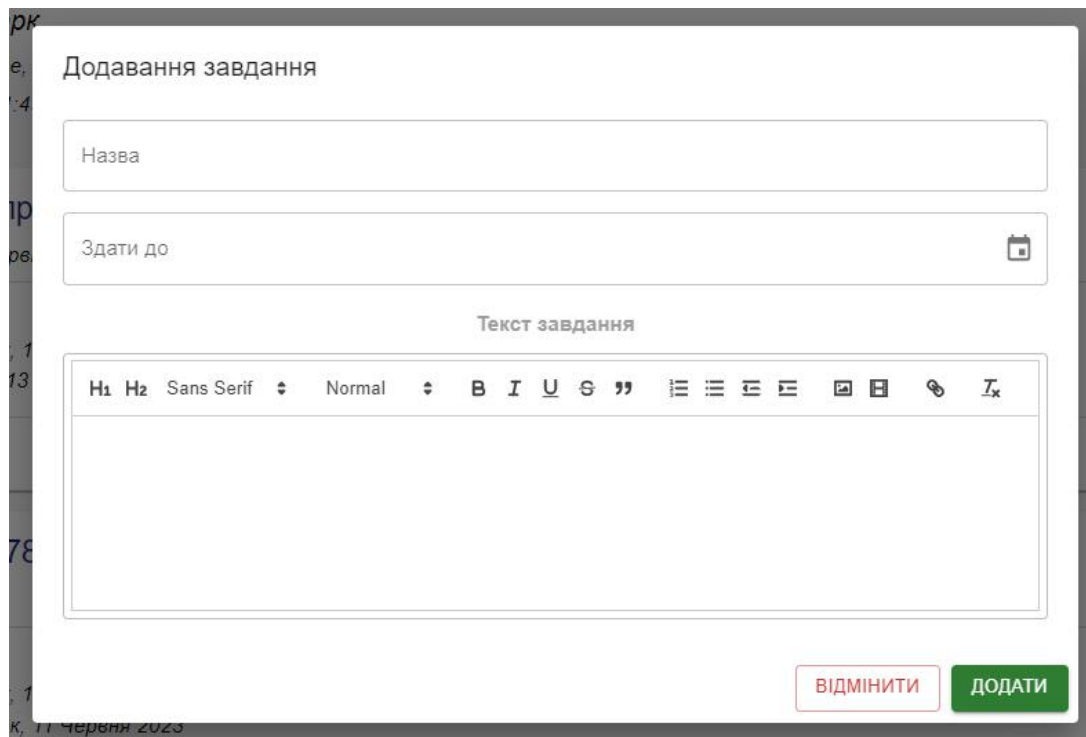


Рисунок 3.12 - Форма створення нового завдання

Для видалення та редагування завдань у карток завдань є таке ж меню, як і у карток предметів. При спробі видалення завдання або предмету користувачеві відображається додаткова форма для підтвердження видалення.

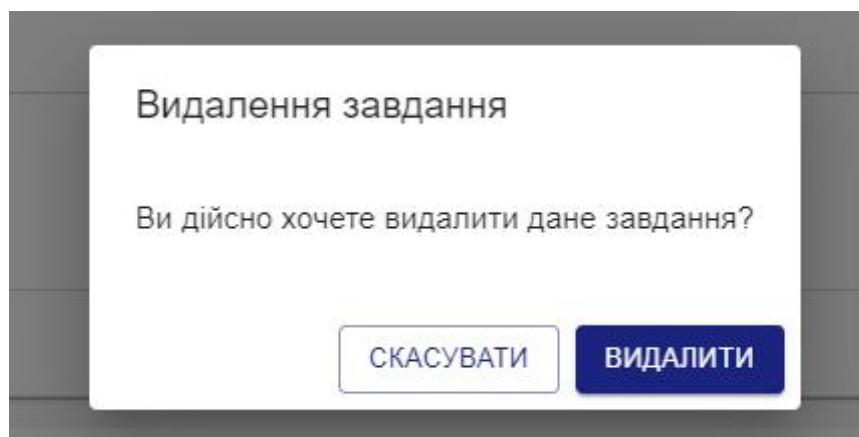


Рисунок 3.13 - Форма підтвердження видалення

Після натискання на кнопку “Результати учнів” користувачеві відобразиться сторінка завдання зі списком усіх учнів, кому це завдання було призначено, та їхніми відповідями. Навпроти імені кожного учня є поле для виставлення оцінки.

AK SCHOOL

Учні

Mattttra ttttt

Оцінка 8

ОЦІНИТИ

Відповіді учня

моя спроба виконати завдання

Філіпов Рима

Оцінка

ОЦІНИТИ

Нора Катерина

Оцінка 5

ОЦІНИТИ

Відповіді учня

Рисунок 3.14 - Сторінка зі спробами учнів виконати завдання

На вкладці “Журнали” можна побачити предмети та класи, яким ці предмети призначені.

AK SCHOOL

Природознавство (3-А, 3-Б)

3-Б

3-А

Читання (3 клас)

3-А

3-Б

Хімія (11 клас)

11-Б

11-А

11-В

Фізична культура (1-2 класи)

1-А

1-Б

2-А

Рисунок 3.15 - Сторінка журналів

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

Для перегляду оцінок виставлених в журнал, користувачеві необхідно натиснути на назву класу.

Природознавство (3-А, 3-Б)						
3-Б						
Завдання	Філіпов Рима	Нора Катерина	Шелест Володислав	Певний Турбог	Зубко Злотован	Середній бал
Кругообіг води в природі	10	5	-	-	-	7.50
QWERTY 1234567890	7	9	-	-	-	8.00
qwerty	10	11	-	-	-	10.50
Агрегатні стани води	4	3	2	1	12	4.40
Тести						
Агрегатні стани	6	11	9	10	9	9.00
Середній бал	7.40	7.80	5.50	5.50	10.50	
3-А						

Рисунок 3.16 - Журнал оцінок класу

Учень

При переході на сторінку предмета користувачеві відображається список усіх завдань по даному предмету.

AK SCHOOL			
← ДО ПРЕДМЕТІВ			
Природознавство (3-А, 3-Б)			
Викладає: Дьолог Марк			
Останнє оновлення: 12 June, 2023 23:58			
Створено: 10 June, 2023 21:49			
Кругообіг води в природі		ЗДАТИ ЗАВДАННЯ	
Без терміну здачі			
Завдання			
QWERTY 1234567890		9.00	
Без терміну здачі		ПЕРЕГЛЯНУТИ	
Завдання			
qwerty		10.00	
Здати до: Четвер, 14 Червня 2023		Термін здачі вищого	
Завдання		ПЕРЕГЛЯНУТИ	
Агрегатні стани води		12.00	
Здати до: Четвер, 28 Червня 2023		ПЕРЕГЛЯНУТИ	
Завдання			

Рисунок 3.17 - Сторінка завдань учня

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

На сторінці можна переглянути свої вже здані та оцінені завдання, а також здати ще не виконані. Для цього потрібно натиснути на кнопку “Здати завдання” на картці завдання. Після чого відобразиться детальна інформація про завдання та форма для здачі.

Рисунок 3.18 - Форма здачі завдання

Для перегляду оцінок в журналі потрібно перейти на вкладку “Журнал”. На цій вкладці відображаються оцінки поточного користувача.

AK SCHOOL

9

Природознавство (3-А, 3-Б)

Завдання	Оцінка
Кругообіг: води в природі	
QWERTY 1234567890	9.00
qwerty	10.00
Агрегатні стани води	12.00

Читання (3 клас)

Завдання	Оцінка
qwerty	1.00

Математика (3 клас)

Завдання відсутні

Рисунок 3.19 - Журнал оцінок учня

3.3. Тестування роботи програмного комплексу

При спробі створення предмету без зазначення його назви, та вчителя, що буде його викладати форма відображає помилку, так як поля “назва” та “вчитель” є обов’язковими.

Рисунок 3.20 - Помилка при створенні предмету

При спробі створення завдання без назви форма видає помилку, так як це поле є обов’язковим. При виборі дати здачі, що вже пройшла, форма також видає помилку.

Рисунок 3.21 - Помилка при створенні завдання

При спробі виставити оцінку не з діапазону від 0 до 12 форма видає помилку та не дозволяє таку оцінку виставити.

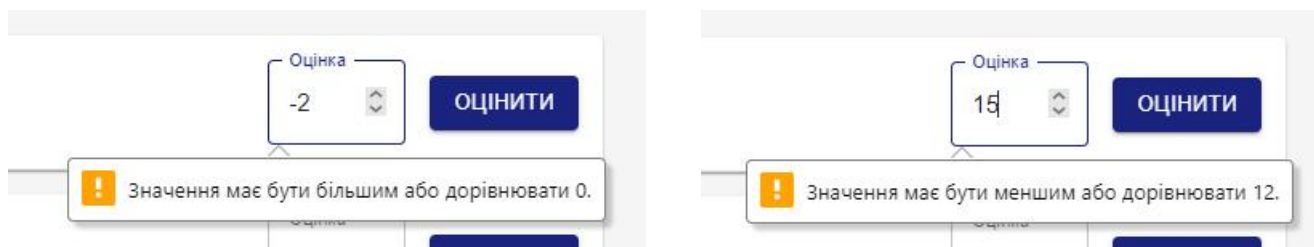


Рисунок 3.22 - Помилка при виставленні оцінки

При перевірці відображення сторінки на пристроях з різними розширеннями екрану помилок не виявлено.

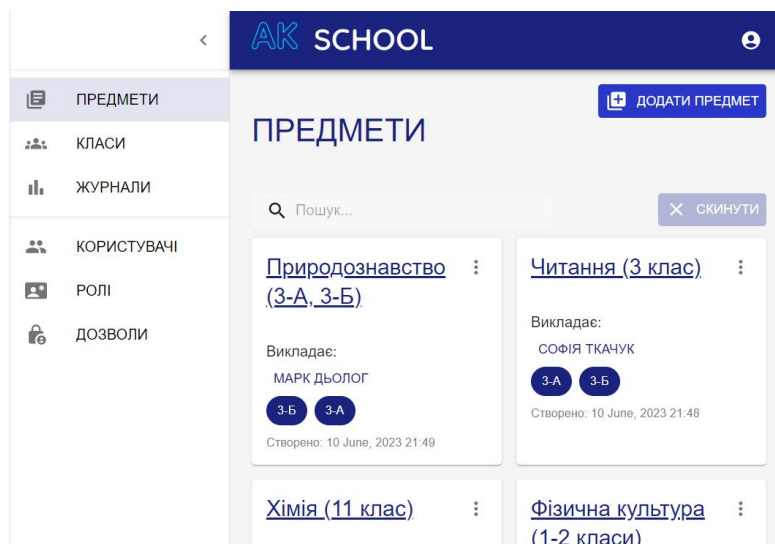


Рисунок 3.23 - Відображення інтерфейсу на планшетах

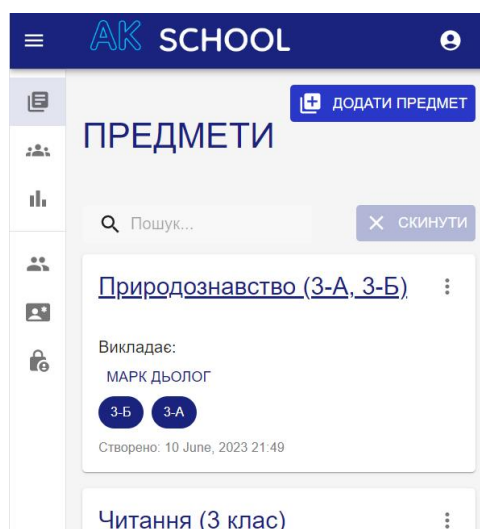


Рисунок 3.24 - Відображення інтерфейсу на мобільному пристрої

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

При спробі доступу до сторінки, до якої у даного користувача доступу немає система перенаправляє на сторінку помилки “Сторінку не знайдено”.

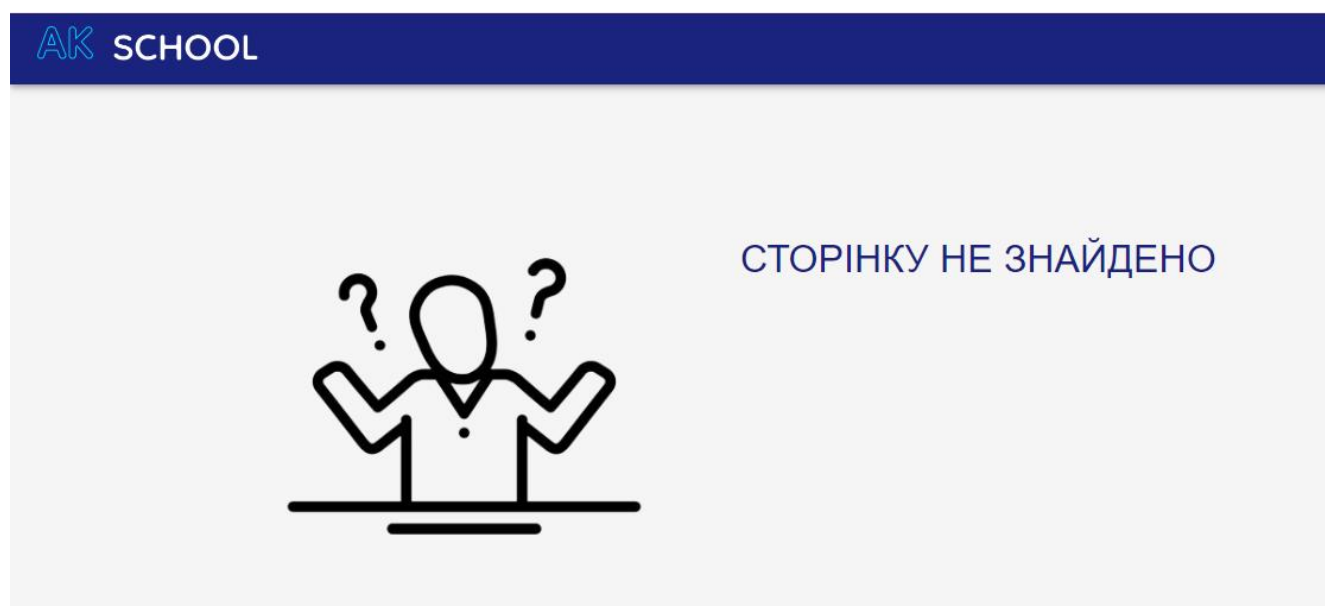


Рисунок 3.25 - Сторінка помилки 404 “Не знайдено”

Висновки до розділу 3

У розділі “Порядок встановлення та налаштування параметрів системи” було представлено діаграму розгортання та діаграму компонентів, які грають важливу роль у встановленні та налаштуванні системи. Вони допомагають зрозуміти структуру системи, її компоненти та взаємозв'язки між ними, що сприяє ефективному процесу встановлення та налаштування системи.

У розділі “Структура інтерфейсу веб-орієнтованої системи з підтримки дистанційної форми навчання” було розглянуто основні складові інтерфейсу системи, такі як головне меню, панель навігації, розділи з предметами, завданнями та журналами. Ця структура надає користувачам зручний спосіб доступу до різних функцій системи і дозволяє ефективно організовувати навчальний процес.

У розділі “Тестування роботи програмного комплексу” було проведено тестування роботи програмного комплексу та було виявлено, що він успішно виконує свої основні функції. Функціональність системи була перевірена та підтверджено, що вона працює так як і очікувалося.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗАГАЛЬНІ ВИСНОВКИ

Під час виконання кваліфікаційної роботи було:

- успішно розроблено програмний комплекс, який надає зручний та ефективний спосіб організації дистанційного навчання. Система включає в себе функціонал, необхідний для проведення навчальних процесів у загальноосвітніх школах;
- використано принципи об'єктно-орієнтованого програмування та вибрано архітектурний підхід MVC. Це забезпечило гнучкість та розширюваність системи, а також полегшує її подальше обслуговування та розвиток;
- використано сучасні технології, такі як Laravel для серверної частини, React.JS для клієнтської, MySQL для бази даних та Axios для здійснення запитів на сервер. Ці технології дозволили створити потужний та ефективний програмний комплекс;
- приділено увагу розробці інтуїтивно зрозумілого та зручного інтерфейсу, який дозволяє користувачам легко орієнтуватися у системі та використовувати її функціонал без особливих зусиль;
- виконано тестування системи, що допомогло виявити та виправити помилки, забезпечити відповідність вимогам та забезпечити надійну та ефективну роботу. Тестування є важливим етапом розробки програмного забезпечення, який сприяє підвищенню якості та стабільності системи.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Всеосвіта - Національна освітня платформа [Електронний ресурс] – Режим доступу до ресурсу: <https://vseosvita.ua>.
2. Всеукраїнська онлайн-школа "Е-school" [Електронний ресурс] – Режим доступу до ресурсу: <https://lms.e-school.net.ua>.
3. HUMAN - Компанія освітніх технологій [Електронний ресурс] – Режим доступу до ресурсу: <https://www.human.ua>.
4. MVC in Computer Science – The MVC Model [Електронний ресурс] – Режим доступу до ресурсу: <https://www.freecodecamp.org/news/what-does-mvc-mean-in-computer-science/>.
5. Introduction to Laravel and MVC Framework [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/introduction-to-laravel-and-mvc-framework/>.
6. Офіційна документація Laravel [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/docs>.
7. SPA (Single-page application) [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Glossary/SPA>.
8. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] – Режим доступу до ресурсу: <https://evergreens.com.ua/ua/articles/uml-diagrams.html>.
9. Основи UML [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-basics.html>.
10. Побудова діаграми використання [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>.
11. Побудова діаграми діяльності [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>.
12. Побудова діаграми класів [Електронний ресурс] – Режим доступу до ресурсу: https://flexberry.github.io/gpg_class-diagram.html.

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Deployment Diagram [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/>.
14. Artisan (command-line interface) [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel-docs.readthedocs.io/en/stable/artisan/>.
15. Офіційна документація MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/>.
16. PhpMyAdmin documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.phpmyadmin.net/en/latest/>.
17. Офіційна документація React.JS [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/learn>.
18. React Router official documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://reactrouter.com/en/main>.
19. Vite.JS (local development server) documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://vitejs.dev/guide/>.
20. Офіційна документація Formik [Електронний ресурс] – Режим доступу до ресурсу: <https://formik.org/docs/overview>.
21. Axios client documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://axios-http.com/docs/intro>.
22. Офіційна документація Redux.JS [Електронний ресурс] – Режим доступу до ресурсу: <https://redux.js.org/introduction/getting-started>.
23. Quill.JS editor's documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://quilljs.com/docs/quickstart/>.
24. Material UI (library of React UI) documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://mui.com/material-ui/getting-started/overview/>.
25. GIT documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://git-scm.com/doc>.

ДОДАТКИ

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

ClassController.php:

```
namespace App\Http\Controllers;

use App\Http\Requests\StoreSubjectRequest;
use App\Http\Requests\StoreTaskRequest;
use App\Http\Requests\UpdateSubjectRequest;
use App\Http\Requests\UpdateTaskRequest;
use App\Http\Resources\SubjectResource;
use App\Models\Subject;
use App\Models\SubjectClass;
use App\Models\Task;
use App\Models\TaskAttempt;
use App\Models\User;
use Carbon\Carbon;
use DateTime;
use DateTimeZone;
use Illuminate\Http\Request;
use PHPUnit\Metadata\Uses;

class SubjectsController extends Controller
{
    public function index(){
        $user = auth()->user();
        if ($user->hasRole('admin')) {
            $subjects = Subject::query()->orderBy('created_at', 'desc')->get();
        } else if ($user->hasRole('teacher')) {
            $subjects = $user->teacherSubjects()->orderBy('created_at', 'desc')->get();
        } else if ($user->hasRole('student')) {
            $ids = SubjectClass::where('class_id', $user['class_id']->pluck('subject_id')->toArray());
            $subjects = Subject::whereIn('id', $ids)->orderBy('created_at', 'desc')->get();
        } else {
            $subjects = [];
        }
        return SubjectResource::collection($subjects);
    }

    public function store(StoreSubjectRequest $request){
        $data = $request->validated();
        $cls = $data['classes_ids'];
        unset($data['classes_ids']);
        $subject = Subject::create($data);
        $clss = new SubjectClass;
        foreach ($cls as $c) {
            $clss::insert(['subject_id' => $subject['id'], 'class_id' => $c]);
        }
        return response(new SubjectResource($subject), 201);
    }
}
```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

```

public function update(UpdateSubjectRequest $request, Subject $subject){
    $data = $request->validated();
    $cls = $data['classes_ids'];
    unset($data['classes_ids']);
    $subject->update($data);
    $clss = new SubjectClass;
    $clss::where('subject_id', $subject['id'])->delete();
    foreach ($cls as $c) {
        $clss::insert(['subject_id' => $subject['id'], 'class_id' => $c]);
    }

    return response(new SubjectResource($subject), 201);
}

public function destroy(Subject $subject)
{
    $subject->delete();
    return response()->json(['message' => "Subject deleted successfully"], 200);
}

public function createTask(StoreTaskRequest $request, string $id){
    $data = $request->validated();
    $task = Task::create($data);
    $class_ids = SubjectClass::where('subject_id', $task['subject_id'])->select('class_id')->get();
    $student_ids = User::whereIn('class_id', $class_ids)->select('id')->get();
    $attempts = [];
    $date = new DateTime("now", new DateTimeZone('Europe/Kiev') );
    foreach ($student_ids as $student )
        $attempts[] = [
            'task_id' => $task['id'],
            'student_id' => $student['id'],
            'updated_at' => $date->format('Y-m-d H:i:s'),
            'created_at' => $date->format('Y-m-d H:i:s'),
        ];
    TaskAttempt::insert($attempts);
    return response($task, 201);
}

public function updateTask(UpdateTaskRequest $request, string $id)
{
    $data = $request->validated();
    $task = Task::findOrFail($id);
    $task->update($data);
    return response($task, 201);
}

public function deleteTask(string $id)
{
    $task = Task::findOrFail($id);

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

```

        $task->delete();

        return response('Task was deleted', 204);
    }
}

```

SubjectResource.php:

```

namespace App\Http\Resources;

use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Http\Resources\Json\JsonResource;

class SubjectResource extends JsonResource
{
    public function toArray(Request $request): array
    {
        return [
            'id' => $this->id,
            'name' => $this->name,
            'teacher' => $this->teacher,
            'classes' => $this->classes,
            'tasks' => $this->tasks,
            'created_at' => Carbon::parse($this->created_at)->isoFormat('DD MMMM, YYYY
HH:mm'),
            'updated_at' => Carbon::parse($this->updated_at)->isoFormat('DD MMMM, YYYY
HH:mm'),
        ];
    }
}

```

Subject.php:

```

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use Spatie\Permission\Traits\HasRoles;

class Subject extends Model
{
    use HasApiTokens, HasFactory, Notifiable, HasRoles;
    protected $guarded = [];

    public function teacher()

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

```

{
    return $this->belongsTo(User::class, 'teacher_id');
}

public function classes()
{
    return $this->belongsToMany(Classes::class, 'subject_classes', 'subject_id', 'class_id');
}

public function tasks(){
    return $this->hasMany(Task::class, 'subject_id');
}

public function tests()
{
    return $this->belongsToMany(Test::class, 'test_subjects');
}
}

```

SubjectClass.php:

```

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use Spatie\Permission\Traits\HasRoles;

class SubjectClass extends Model
{
    use HasApiTokens, HasFactory, Notifiable, HasRoles;
    protected $guarded = [];

    public function classes()
    {
        return $this->belongsTo(Classes::class, 'class_id');
    }

    public function subject()
    {
        return $this->belongsTo(Subject::class, 'subject_id');
    }
}

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

StoreSubjectRequest.php:

```
namespace App\Http\Requests;

use App\Models\User;
use Illuminate\Foundation\Http\FormRequest;

class StoreSubjectRequest extends FormRequest
{
    public function authorize(): bool
    {
        return true;
    }

    public function rules(): array
    {
        return [
            'name' => 'required|string|max:255',
            'teacher_id' => [
                'nullable',
                'exists:users,id',
                function ($attribute, $value, $fail) {
                    $user = User::find($value);
                    if (!$user->hasRole('teacher')) {
                        $fail("Помилка. " . $user->first_name . " " . $user->second_name . " не є вчителем.");
                    }
                }
            ],
            'classes_ids' => 'nullable|array',
            'classes_ids.*' => [
                'exists:classes,id',
            ],
        ];
    }
}
```

UpdateSubjectRequest.php:

```
namespace App\Http\Requests;

use App\Models\User;
use Illuminate\Foundation\Http\FormRequest;

class UpdateSubjectRequest extends FormRequest
{
    public function authorize(): bool
    {
        return true;
    }
}
```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

```

        return true;
    }

    public function rules(): array
    {
        return [
            'name' => 'required|string|max:255',
            'teacher_id' => [
                'nullable',
                'exists:users,id',
                function ($attribute, $value, $fail) {
                    $user = User::find($value);
                    if (!$user->hasRole('teacher')) {
                        $fail("Помилка. " . $user->first_name . " . $user->second_name . " не є вчителем.");
                    }
                }
            ],
            'classes_ids' => 'nullable|array',
            'classes_ids.*' => [
                'exists:classes,id',
            ],
        ];
    }
}

```

2023_05_01_150514_create_subjects_table.php:

```

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('subjects', function (Blueprint $table) {
            $table->id();
            $table->string('name');

            $table->unsignedBigInteger('teacher_id')->nullable();
            $table->foreign('teacher_id')->references('id')->on('users')->onDelete('cascadeOnUpdate');

            $table->timestamps();
        });
    }
}

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public function down(): void
{
    Schema::dropIfExists('subjects');
}
};

```

2023_05_06_192121_create_subject_classes_table.php:

```

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('subject_classes', function (Blueprint $table) {
            $table->unsignedBigInteger('class_id')->nullable();
            $table->foreign('class_id')->references('id')->on('classes')->cascadeOnUpdate()-
>cascadeOnDelete();

            $table->unsignedBigInteger('subject_id')->nullable();
            $table->foreign('subject_id')->references('id')->on('subjects')->cascadeOnUpdate()-
>cascadeOnDelete();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('subject_classes');
    }
};

```

api.php:

```

use App\Http\Controllers\AuthController;
use App\Http\Controllers\SubjectsController;
use App\Http\Controllers\TasksController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\UserController;
use App\Http\Controllers\RoleController;
use App\Http\Controllers\ClassesController;
use App\Http\Controllers\TestController;
use App\Http\Controllers\AnswersController;

Route::middleware('auth:sanctum')->group(function () {

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

```

Route::post('/logout', [AuthController::class, 'logout']);
Route::get('/user', function (Request $request) {
    return $request->user();
});
Route::get('/me', [AuthController::class, 'me']);

Route::get('/users', [UserController::class, 'index']);
Route::get('/students-without-class/{classId?}', [UserController::class, 'getStudentsForClass']);

Route::group(['middleware' => ['permission:create users']], function () {
    Route::post('/users/create', [UserController::class, 'store']);
});
Route::group(['middleware' => ['permission:delete users']], function () {
    Route::delete('/users/{user}', [UserController::class, 'destroy']);
});
Route::group(['middleware' => ['permission:update users']], function () {
    Route::put('/users/{user}', [UserController::class, 'update']);
});
Route::group(['middleware' => ['permission:generate new password']], function () {
    Route::post('/users/new-password/{user}', [UserController::class, 'newPassword']);
});

Route::group(['middleware' => ['permission:create tests']], function () {
    Route::post('/tests/create', [TestController::class, 'store']);
});

Route::group(['middleware' => ['permission:update tests' , ]], function () {
    Route::post('/tests/add-questions/{test_id}', [TestController::class, 'addQuestions']);
    Route::put('/tests/update-questions/{question_id}', [TestController::class,
'updateQuestions']);
    Route::delete('/tests/delete-questions/{question_id}', [TestController::class,
'deleteQuestions']);

    Route::put('/tests/{test}', [TestController::class, 'update']);
    Route::post('/tests/{test}/changeTestStatus', [TestController::class, 'changeTestStatus']);
});
Route::group(['middleware' => ['permission:delete tests' , ]], function () {
    Route::delete('/tests/{test}', [TestController::class, 'destroy']);
});

Route::group(['middleware' => [
]], function () {
    Route::get('/tests/get-by-slug/{test:slug}', [TestController::class, 'getBySlug']);
});

Route::apiResource('/tests', TestController::class);
Route::apiResource('/roles', RoleController::class);

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

```

Route::apiResource('/classes', ClassesController::class);

Route::apiResource('/subjects', SubjectsController::class);

Route::apiResource('/answers', AnswersController::class);

Route::post('/subjects/create-task/{subject_id}', [SubjectsController::class, 'createTask']);
Route::put('/subjects/update-task/{task_id}', [SubjectsController::class, 'updateTask']);
Route::delete('/subjects/delete-task/{task_id}', [SubjectsController::class, 'deleteTask']);

Route::apiResource('/tasks', TasksController::class);
Route::put('/tasks/update-attempt/{attempt_id}', [TasksController::class, 'updateTaskAttempt']);
});

Route::post('/login', [AuthController::class, 'login']);

```

backend/vite.config.js

```

import { defineConfig } from 'vite';
import laravel from 'laravel-vite-plugin';

export default defineConfig({
  plugins: [
    laravel({
      input: ['resources/css/app.css', 'resources/js/app.js'],
      refresh: true,
    }),
  ],
});

```

frontend/vite.config.js

```

import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

export default defineConfig({
  plugins: [react()],
})

```

axios-client.js

```

import axios from "axios";

const axiosClient = axios.create({
  baseURL: `${import.meta.env.VITE_API_BASE_URL}/api`
})

axiosClient.interceptors.request.use((config) => {

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

```

const token = localStorage.getItem('ACCESS_TOKEN')
config.headers.Authorization = `Bearer ${token}`
return config;
})

axiosClient.interceptors.response.use((response) => {
  return response;
}, (error) => {
  try {
    const {response} = error;
    if (response.status === 401) {
      localStorage.removeItem('ACCESS_TOKEN');
    }
  } catch (e) {
    console.error(e);
  }
  throw error;
})
export default axiosClient;

```

index.html

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/logo_short.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>AK SCHOOL</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>

```

main.jsx

```

import React, { useMemo, useState } from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import { RouterProvider } from 'react-router-dom';
import router from './router.jsx';
import { Provider } from 'react-redux';
import { store } from './store/store.js';
import { createTheme, ThemeProvider } from '@mui/material/styles';
import { theme } from './utils/theme.js';
import * as locales from '@mui/material/locale';

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

```

const App = () => {
  const [locale, setLocale] = useState('ukUA');
  const themeWithLocale = useMemo(() => createTheme(theme, locales[locale]), [locale, theme]);
  return (
    <React.StrictMode>
      <ThemeProvider theme={themeWithLocale}>
        <Provider store={store}>
          <RouterProvider router={router} />
        </Provider>
      </ThemeProvider>
    </React.StrictMode>
  );
};

ReactDOM.createRoot(document.getElementById('root')).render(<App />);

```

router.jsx

```

import {createBrowserRouter, Navigate} from "react-router-dom";
import DefaultLayout from "../componets/DefaultLayout.jsx";
import GuestLayout from "../componets/GuestLayout.jsx";
import UsersList from "../componets/Users/UsersList.jsx";
import Login from "../componets/Users/Login.jsx";
import ClassesList from "../componets/Classes/ClassesList.jsx";
import TestsList from "../componets/Tests/TestsList.jsx";
import TestsEditor from "../componets/Tests/TestsEditor.jsx";
import SubjectsList from "../componets/Subjects/SubjectsList.jsx";
import TestPage from "../componets/Tests/testPassing/TestPage.jsx";
import TasksList from "../componets/Tasks/TasksList.jsx";
import TaskPage from "../componets/Tasks/Attempts/TaskPage.jsx";
import NotFound from "../componets/errors/NotFound";
import Journal from "../componets/journal/Journal.jsx";
import Profile from "../componets/Users/Profile.jsx";

```

```

const router = createBrowserRouter([
  {
    path: '/',
    element: <DefaultLayout/>,
    children: [
      {
        path: "/",
        element: <Profile/>
      },
      {
        path: '/dashboard',
        element: <Navigate to="/" />
      },
    ],
  },
]);

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

```

{
    path: "/users",
    element: <UsersList/>
},
{
    path: "/roles",
    element: <UsersList/>
},
{
    path: "/permissions",
    element: <UsersList/>
},
{
    path: "/classes",
    element: <ClassesList/>
},
{
    path: "/tests",
    element: <TestsList/>
},
{
    path: '/tests/:id',
    element: <TestsEditor/>,
},
{
    path: '/tests/passing/:slug',
    element: <TestPage/>,
},
{
    path: '/tests/passing/:slug',
    element: <TestPage/>,
},
{
    path: '/subjects',
    element: <SubjectsList/>
},
{
    path: '/subjects/:id',
    element: <TasksList />
},
{
    path: '/task/:id',
    element: <TaskPage/>
},
{
    path: 'error/404',
    element: <NotFound/>
},
},

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

```

        {
            path: '/journal',
            element: <Journal/>
        }
    ]
},
{
    path: '/',
    element: <GuestLayout/>,
    children: [
        {
            path: '/login',
            element: <Login/>
        },
    ],
},
{}
]);

```

export default router;

subjectsSlice.js

```

import {createAsyncThunk, createSlice} from "@reduxjs/toolkit";
import axiosClient from "../././axios-client.js";

const initialState = {
    subjects: [],
    subject: {},
    visibleData: [],
    errors: null,
    isLoading: true,
    status: 'idle',
}

// П Р Е Д М Е Т И
export const getSubjects = createAsyncThunk("subjects/getSubjects",
    async (_, {rejectWithValue}) => {
        try {
            const res = await axiosClient.get('/subjects');
            // console.log(res.data)
            return res.data;
        } catch (error) {
            if (!error.response) {
                return rejectWithValue("Не вдалося з'єднатися з сервером.");
            }
            const {errors} = error.response.data;
            return rejectWithValue(errors);
        }
    }
);

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

    }
  });
export const createSubject = createAsyncThunk("subjects/createSubject",
  async (payload, {rejectWithValue}) => {
    try {
      // console.log(payload)
      const res = await axiosClient.post('/subjects', payload);
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  });
export const updateSubject = createAsyncThunk("subjects/updateSubject",
  async ({id, values}, {rejectWithValue}) => {
    try {
      // console.log(values)
      const res = await axiosClient.put(`/subjects/${id}`, values);
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  });
export const deleteSubject = createAsyncThunk("subjects/deleteSubject",
  async ({id, subjectItem}, {rejectWithValue}) => {
    try {
      const res = await axiosClient.delete(`/subjects/${id}`);
      // console.log(res);
      return subjectItem;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  });

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

```
//          З А В Д А Н Н Я
export const getAttempts = createAsyncThunk('tasks/getTasks',
  async (_, {rejectWithValue}) => {
    try {
      const res = await axiosClient.get('/tasks');
      // console.log(res.data)
      return res.data;
    } catch (error) {
      if (!error.response) {
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  });

export const createTask = createAsyncThunk("tasks/createTask",
  async ({subject_id, values}, {rejectWithValue}) => {
    try {
      const res = await axiosClient.post(`/subjects/create-task/${subject_id}`, values);
      console.log(res.data)
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  });

export const updateTask = createAsyncThunk("tasks/updateTask",
  async ({task_id, values}, {rejectWithValue}) => {
    try {
      const res = await axiosClient.put(`/subjects/update-task/${task_id}`, values);
      console.log(res.data);
      return res.data;
    } catch (error) {
      if (!error.response) {
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  });

export const deleteTask = createAsyncThunk("tasks/deleteTask",
  async ({task_id, task}, {rejectWithValue}) => {
    try {
      await axiosClient.delete(`/subjects/delete-task/${task_id}`, task);
      return task;
    }
  });
```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

    } catch (error) {
      if (!error.response) return rejectWithValue("Не вдалося з'єднатися з сервером.");
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  })

const subjectsSlice = createSlice({
  name: "subjects",
  initialState,
  reducers: {
    clearErrors: (state) => {
      state.errors = null;
    },
    reset: (state) => {
      state.subjects = [];
      state.visibleData = state.subjects;
      state.errors = null;
    },
    searchSubject: (state, {payload}) => {
      state.visibleData = state.subjects.filter(
        (item) =>
          item.name.toLowerCase().includes(payload.toLowerCase())
      );
    },
    getSubjectById: (state, {payload}) => {
      state.subject = state.visibleData.find(subj => subj.id === parseInt(payload))
    }
  },
  extraReducers: (builder) => {
    //          П Р Е Д М Е Т И
    builder.addCase(getSubjects.pending, (state) => {
      state.isLoading = true;
    });
    builder.addCase(getSubjects.fulfilled, (state, action) => {
      state.status = 'succeeded';
      state.subjects = action.payload.data;
      state.visibleData = state.subjects;
      state.isLoading = false;
    });
    builder.addCase(getSubjects.rejected, (state, action) => {
      state.status = 'rejected';
      state.errors = action.payload;
      state.isLoading = false;
    });
    builder.addCase(createSubject.pending, (state) => {
      state.isLoading = true;
    });
    builder.addCase(createSubject.fulfilled, (state, action) => {

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        state.status = 'succeeded';
        state.subjects.unshift(action.payload);
        state.visibleData.unshift(action.payload);
    });
    builder.addCase(createSubject.rejected, (state, action) => {
        state.status = 'rejected';
        state.errors = action.payload;
    });
    builder.addCase(updateSubject.pending, (state) => {
        state.isLoading = true;
    });
    builder.addCase(updateSubject.fulfilled, (state, action) => {
        state.status = 'succeeded';
        state.subjects = state.subjects.map(subjectItem =>
            subjectItem.id === action.payload.id ? action.payload : subjectItem)
        state.visibleData = state.subjects;
    });
    builder.addCase(updateSubject.rejected, (state, action) => {
        state.status = 'rejected';
        state.errors = action.payload;
    });
    builder.addCase(deleteSubject.pending, (state) => {
        state.isLoading = true;
    });
    builder.addCase(deleteSubject.fulfilled, (state, action) => {
        state.status = 'succeeded';
        state.subjects = state.subjects.filter(subjectItem => subjectItem.id !==
action.payload.id);
        state.visibleData = state.subjects;
    });
    builder.addCase(deleteSubject.rejected, (state, action) => {
        state.status = 'rejected';
        state.errors = action.payload;
    });

//      З А В Д А Н Н Я
    builder.addCase(createTask.pending, (state) =>{
        state.isLoading = true;
    });
    builder.addCase(createTask.fulfilled, (state, action) =>{
        state.status = 'succeeded';
        state.subject.tasks.push(action.payload);
    });
    builder.addCase(createTask.rejected, (state, action) => {
        state.status = 'rejected';
        state.errors = action.payload;
    });
    builder.addCase(updateTask.pending, (state) => {})

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        builder.addCase(updateTask.fulfilled, (state, action) => {
            state.status = 'succeeded';
            state.subject.tasks = state.subject.tasks.map(task => task.id === action.payload.id ?
action.payload : task);
        })
        builder.addCase(updateTask.rejected, (state, action) => {
            state.status = 'rejected';
            state.errors = action.payload;
        })
        builder.addCase(deleteTask.pending, (state) => {})
        builder.addCase(deleteTask.fulfilled, (state, action) => {
            state.status = 'succeeded';
            state.subject.tasks = state.subject.tasks.filter(task => task.id !== action.payload.id);
        })
        builder.addCase(deleteTask.rejected, (state, action) => {
            state.status = 'rejected';
            state.errors = action.payload;
        })
    }
})
export const {clearErrors, reset, searchSubject, getSubjectById} = subjectsSlice.actions;
export default subjectsSlice.reducer;

```

SubjectsList.jsx

```

import Grid from "@mui/material/Grid";
import React, {useEffect, useState} from "react";
import {useDispatch, useSelector} from "react-redux";
import SubjectCard from "../SubjectCard.jsx";
import {useNavigate} from "react-router-dom";
import {searchSubject} from "../../store/subject/subjectsSlice.js";
import Typography from "@mui/material/Typography";
import styles from "../../styles/Grid.module.css";
import {Button, CircularProgress} from "@mui/material";
import {theme} from "../../utils/theme.js";
import PersonAddAltOutlinedIcon from "@mui/icons-material/PersonAddAltOutlined.js";
import AddToPhotosIcon from '@mui/icons-material/AddToPhotos';
import FormClass from "../Classes/FormClass.jsx";
import FormSubject from "../FormSubject";
import {Search, SearchIconWrapper, StyledInputBase} from "../../styles/searchStyles.js";
import SearchIcon from "@mui/icons-material/Search.js";
import ClearIcon from "@mui/icons-material/Clear.js";
import TablePagination from "@mui/material/TablePagination";
import Notification from "../../core/Notification.jsx";
import Box from "@mui/material/Box";

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

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const {user} = useSelector((state) => state.currentUser)
const {subjects, isLoading, visibleData} = useSelector((state) => state.subjects)
const [notification, setNotification] = useState(false);
const [open, setOpen] = useState(false);
const [searchValue, setSearchValue] = useState("");
const [page, setPage] = useState(0);
const [rowsPerPage, setRowsPerPage] = useState(9);

const handleChangePage = (event, newPage) => {
    setPage(newPage);
};
const handleChangeRowsPerPage = (event) => {
    setRowsPerPage(parseInt(event.target.value, 10));
    setPage(0);
};
const handleCloseAlert = (event, reason) => {
    if (reason === 'clickaway') {return;}
    setNotification(false);
};
const handleSearch = (event) => {
    setSearchValue(event.target.value);
};
const onClickReset = () => {
    setSearchValue("");
};
const handleClickOpen = () => {
    setOpen(true);
};
const handleClose = (value) => {
    setOpen(false);
    if (value) setNotification('Предмет успішно додано!');
};

const displayedSubjects = visibleData.slice(page * rowsPerPage, page * rowsPerPage +
rowsPerPage);

useEffect(() => {
    dispatch(searchSubject(searchValue));
}, [searchValue])

return (
    <>
        <Grid container justifyContent="space-between"
            style={{flexWrap: 'wrap', paddingBottom: 5}}
            className={styles['no-padding-top']]>
            <Grid item xs={6} lg={8} style={{alignItems: 'end', paddingBottom: 5}}
                className={styles['no-padding-top']]>
                <Typography component="h2" variant="h4" color="primary" py="30px"

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

gutterBottom>

```
                ПРЕДМЕТИ
            </Typography>
        </Grid>
        <Grid item xs={6} lg={3} style={{textAlign: 'right', paddingBottom: 5}}
            className={styles['no-padding-top']}>
            {user.permissions?.includes("create classes") &&
                <Button color="secondary"
                    style={{ backgroundColor: theme.palette.secondary.main,
color: 'white' }}
                    onClick={handleClickOpen}>
                <AddToPhotosIcon style={{marginRight: 10}}/>
                ДОДАТИ ПРЕДМЕТ
            </Button>
            }
        <FormSubject open={open}
            onClose={handleClose}
        />
    </Grid>
    <Grid item xs={7} lg={2} style={{alignItems: 'end', paddingBottom: 5}}
        className={styles['no-padding-top']}>
        <Search>
            <SearchIconWrapper>
                <SearchIcon/>
            </SearchIconWrapper>
            <StyledInputBase
                placeholder="Пошук..."
                inputProps={{'aria-label': 'search'}}
                onChange={handleSearch}
                value={searchValue}
            />
        </Search>
    </Grid>
    <Grid item xs={5} lg={3} style={{textAlign: 'right'}}
        className={styles['no-padding-top']}>
        <Button style={{backgroundColor: '#b3b7d7', color: 'white'}}
            onClick={onClickReset}
        >
            <ClearIcon style={{marginRight: 10}}/>
        </Button>
        СКИНУТИ
    </Grid>
</Grid>
{isLoading &&
    <Box sx={{display: 'flex', justifyContent: 'center'}}>
        <CircularProgress/>
    </Box>
}
{!isLoading &&
```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

    </>
    <Grid container spacing={2}>
      {displayedSubjects.map((subject) => (
        <Grid key={subject.id} item xs={12} sm={6} md={4}>
          <SubjectCard subjectItem={subject}/>
        </Grid>
      ))}
    </Grid>
    <TablePagination
      rowsPerPageOptions={[9, 21, 33, 45]}
      component="div"
      count={subjects.length}
      page={page}
      onPageChange={handleChangePage}
      rowsPerPage={rowsPerPage}
      onRowsPerPageChange={handleChangeRowsPerPage}
      labelRowsPerPage="Кількість на сторінці:"
    />
    {notification && (
      <Notification notification={!notification}
        handleCloseAlert={handleCloseAlert}
        hideDuration={3000}
        text={notification}/>
    )}
  </>
}
</>
);
}

export default SubjectsList;

```

		Кирилюк Д.М.			ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		80