СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)

на тему „Веб-платформа управління навчальним процесом”

Виконала: студентка групи ІПЗ-20д _____ А. О. Гуленко
(підпис) (ініціали і прізвище)

Керівник _____ В. Г. Іванов
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ О.І. Захожай
(підпис) (ініціали і прізвище)

Рецензент ____Лифар В.О.____

Київ – 2024

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування

Освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП
_____ Захожай О.І.
“___” _____ 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Гуленко Анастасія Олександрівна
(прізвище, ім’я, по батькові)

1. Тема роботи: Веб-платформа управління навчальним процесом

Керівник роботи Іванов Віталій Геннадійович, к.т.н. , доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджений наказом університету від “06”травня 2024року №171/15.15-С

2. Строк подання студентом роботи 08.06.2024

3. Вихідні дані до роботи: Об’єктом даної роботи є процес розробки веб- додатку управління навчальним процесом.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): Вступ. Поняття системи управління навчальним процесом з висвітленням наступних питань: визначення системи управління навчальним процесом, завдання системи управління навчальним процесом. Оцінка та порівняння використаних інструментів та технологій для веб-сервісу: HTML&CSS, вибір мови програмування, вибір фремфорків, React, Node.js, вибір серверу та бази даних, Figma. Розробка системи управління Навчальним процесом, в якій висвітлена інформація про загальний опис постанови задачі проекту, архітектуру програмного проекту, розробку моделі та опису бази даних, організацію файлів проєкту, характеристику ключових процесів обробки, продемонстрований логотип та інтерфейс додатку. Висновок. Перелік використаних джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників)Додаток А.

6. Консультанти розділів роботи

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

7. Дата видачі завдання 30.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Отримання завдання для виконання роботи	30.03.24	виконано
2	Розробка і узгодження з керівником плану та етапів виконання роботи	10.04.24	виконано
3	Узагальнення даних літературних джерел, аналіз теоретичної частини	17.04.24	виконано
4	Аналіз можливих підходів до виконання завдання. Вибір і узгодження з керівником найкращого варіанту.	28.04.24	виконано
5	Укладання та тестування програмного продукту	01.05.24	виконано
6	Укладання, оформлення та погодження пояснювальної записки з керівником	06.05.24	виконано
7	Здача готової пояснювальної записки на кафедру	08.06.24	виконано
8	Укладання доповіді і презентації	10.06.24	виконано

Студент _____ А. О. Гуленко
підпис (ініціали і прізвище)

Керівник роботи _____ В. Г. Іванов
підпис (ініціали і прізвище)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студентки гр. ІПЗ-20д Гуленко А. О.

Науковий керівник

к.т.н., доцент _____ Іванов В. Г.

Оцінка наукового керівника: _____

Рецензент __Лифар В.О. , д.т.н., професор кафедри ІТП СНУ ім. В.Даля___
ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

Голова ЕК
Професор кафедри ІТП,
д.т.н. _____ Меняйленко О.С.
підпис

ЗМІСТ

ВСТУП

Актуальність досліджень.

У зв'язку з постійним розвитком інформаційних технологій та зростаючою потребою в ефективному управлінні навчальним процесом, виникає необхідність в створенні та вдосконаленні веб-додатків для управління освітніми програмами. Актуальність цього дослідження полягає в пошуку оптимальних рішень для впровадження таких додатків, які сприятимуть покращенню якості освіти та забезпечать зручний та ефективний процес навчання.

Об’єкт дослідження:

Веб-додаток для управління навчальним процесом.

Предмет дослідження:

Функціональні можливості та ефективність використання веб-додатка для управління навчальним процесом.

Мета дослідження:

Метою даного дослідження є розробка та впровадження веб-додатка, який надає зручний та ефективний інструмент для управління навчальним процесом, сприяючи підвищенню якості освіти та оптимізації роботи навчальних закладів.

Завдання дипломної роботи:

- проведення аналізу потреб користувачів та вимог;
- дослідження переваг різних мов програмування та сучасних інструментів, які можуть бути використані для створення веб-додатку;
- розробка архітектури веб-додатку та його функціональних можливостей;
- створення дизайну, який робить сайт привабливим і зрозумілим для користувачів;
- розробка зручного інтерфейсу, що дозволяє комфортно користуватися сайтом.

Цільова аудиторія: викладачі, яким необхідно розміщувати навчальні матеріали і завдання до них, мати доступ до свої файлів будь-де, студенти денної та заочної форм навчання.

1. ПОНЯТТЯ СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ

1.1. Визначення системи управління навчальним процесом

Система управління навчанням (LMS) – це програмне забезпечення, розроблене спеціально для створення, розповсюдження та керування доставкою освітнього контенту. LMS може бути розміщена як окремий продукт на сервері компанії, або це може бути хмарна платформа, яка розміщується фірмою-розробником програмного забезпечення.[1]

Найпростіша система управління навчальним процесом (LMS) забезпечує основні функції, такі як завантаження навчального матеріалу, проведення уроків для учасників, розсилання сповіщень та обмін даними з авторизованими користувачами. Зазвичай LMS працює у веб-браузері, що забезпечує зручний та безпечний доступ до курсів для учнів і викладачів з будь-якого пристрою.

LMS відіграє важливу роль у підтримці викладачів та студентів у процесі навчання. Для викладачів LMS надає зручний інструмент для планування та виконання навчальних програм. Вони можуть легко створювати розклади занять, завдання для студентів, тестові завдання та інші матеріали, необхідні для проведення курсу. Крім того, LMS дозволяє викладачам відстежувати активність студентів, аналізувати їхній прогрес та готувати звіти про навчальний процес.

З точки зору студентів, LMS відкриває широкі можливості для організації навчання. Вони можуть легко доступатися до навчального матеріалу, завдань та ресурсів, які надаються в рамках курсу. Завдяки системі сповіщень та календарю, студенти можуть вчасно отримувати інформацію про зближені терміни здачі завдань та інші важливі події. Також, LMS створює можливості для спілкування та співпраці між студентами, що сприяє активному обміну знаннями та досвідом.[2]

1.2. Завдання систем управління навчальним процесом

- Організація та планування навчального процесу: забезпечення автоматизацію створення та управління розкладом занять, що включає планування навчальних модулів, курсів та уроків. Це допомагає зменшити адміністративне навантаження та уникнути конфліктів у розкладі.

- Управління навчальними матеріалами: надавання можливість зберігання, організації та надання доступу до навчальних матеріалів у різних форматах (тексти, відео, аудіо, презентації тощо). Це спрощує процес підготовки та використання навчальних ресурсів.

- Моніторинг та оцінювання успішності студентів: підтримуваня ведення електронних журналів успішності, автоматизацію процесу оцінювання та збереження результатів. Вона також має надавати можливість генерувати звіти про успішність студентів для аналізу їх прогресу.

- Підтримка комунікації та співпраці: сприяння ефективній комунікації між студентами та викладачами, а також між самими студентами. Це включає організацію форумів, чатів, відеоконференцій та відправлення сповіщень про важливі події чи зміни

- Адміністрування користувачів та доступу: надаваня можливість управління ролями та правами доступу для різних категорій користувачів, таких як студенти, викладачі та адміністратори. Важливо також забезпечити безпеку та конфіденційність даних користувачів.

- Підтримка навчального процесу: LMS повинна включати інструменти для створення та проведення тестів, опитувань та завдань, а також для організації та управління дистанційними курсами. Це допомагає зробити навчальний процес більш інтерактивним та доступним.

- Ці завдання спрямовані на оптимізацію освітнього процесу, підвищення його ефективності, зручності для користувачів та забезпечення високої якості освіти.

2. ПОРІВНЯННЯ СУЧАСНИХ ПЛАТФОРМ ДЛЯ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ

2.1 Огляд сучасних платформ

Moodle

Moodle - це система управління навчанням (LMS), розроблена для забезпечення викладачів, адміністраторів і учнів єдиною надійною, безпечною та інтегрованою системою для створення персоналізованого навчального середовища.

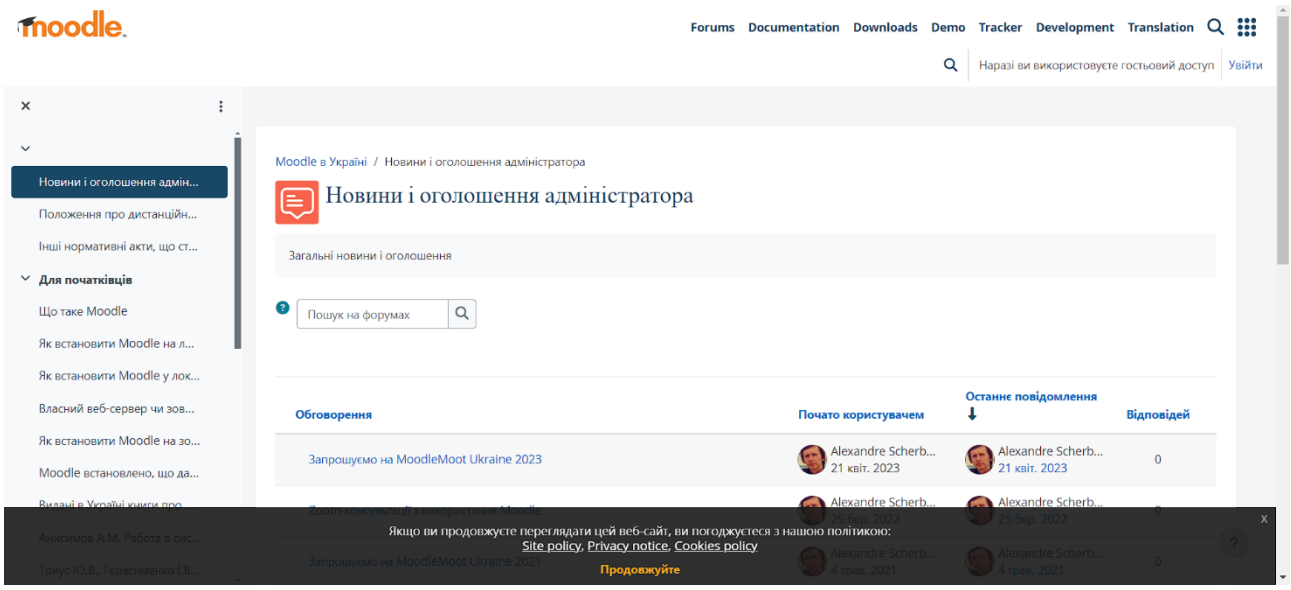


Рисунок 2.1.1. - Moodle

Moodle надається безкоштовно як програмне забезпечення з відкритим вихідним кодом згідно з GNU General Public License . Будь-хто може адаптувати, розширити або модифікувати цю LMS як для комерційних, так і для некомерційних проектів без будь-яких ліцензійних платежів і отримати вигоду від економічності, гнучкості та інших переваг використання Moodle.[3]

Основні функції:

адміністування:

адміністратор керує сайтом, налаштовує його вигляд, може розширити функціонал за допомогою модулів, локалізувати систему для будь-якої країни та мови та вносити будь-які зміни завдяки відкритому коду;

- управління користувачами:

система управління користувачами в Moodle дозволяє реєструвати користувачів за допомогою різних методів, таких як самореєстрація, ручна реєстрація адміністратором або використання LDAP(Легкий протокол доступу до каталогу). Крім того, Moodle надає можливість автоматичного відновлення паролів користувачів через електронну пошту та забезпечує ефективний захист від несанкціонованого доступу. Інформація про користувачів зберігається у їхніх профайлах, де вони можуть додавати та оновлювати інформацію за власним розсудом;

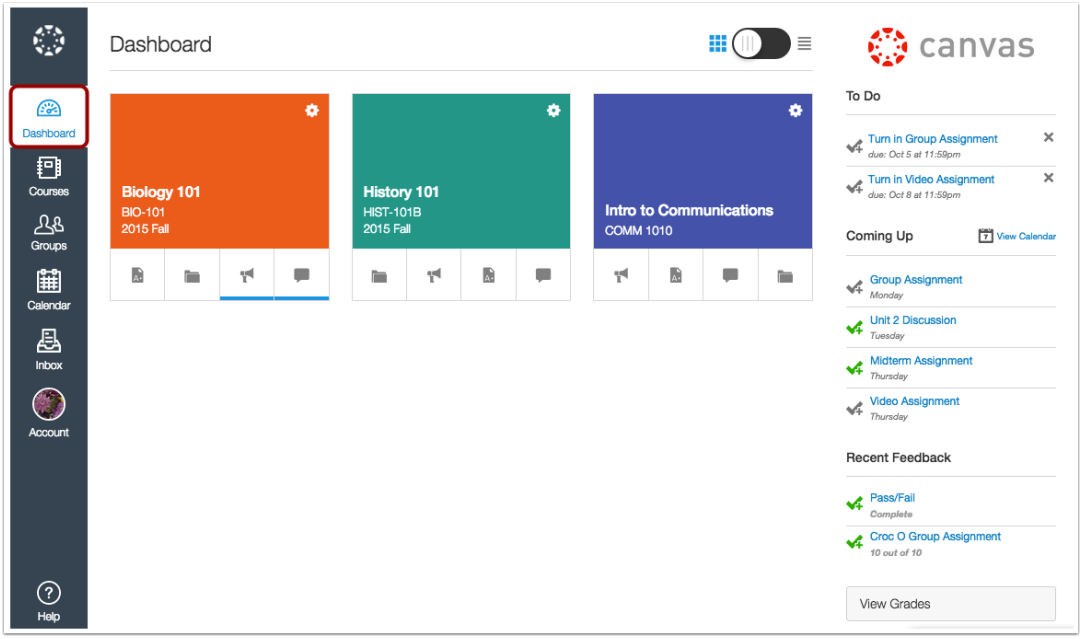
- керування курсами:

за замовчуванням, у Moodle викладач має повний контроль над властивостями курсу, хоча ці можливості можуть бути обмежені адміністратором. Для організації курсів доступні різні формати, такі як SCORM, топіки і т.д., і для кожного курсу можна встановити індивідуальні налаштування. Moodle пропонує широкий набір інтерактивних елементів, таких як форуми, тести, глосарії, ресурси, чати тощо. Всі останні зміни в курсі зберігаються, і для кожного курсу відстежується повна інформація про успішність слухачів. Система Moodle інтегрована з поштовими системами, що дозволяє передавати інформацію між викладачами та слухачами електронною поштою. Дистанційні курси можуть бути упаковані в ZIP-архіви за допомогою функції Backup, і елементи курсів можуть бути імпортовані з інших курсів.

Canvas

Canvas — це веб-система керування навчанням, або LMS. Він використовується навчальними закладами, викладачами та студентами для доступу та керування навчальними матеріалами онлайн-курсу та спілкування щодо розвитку навичок і досягнень у навчанні.

Canvas містить різноманітні настроювані інструменти для створення та керування курсами, аналітику й статистику курсів і користувачів, а також інструменти



внутрішньої комунікації. [4]

Рисунок 2.1.2 - Canvas

Ця LMS містить різноманітні вбудовані інструменти для створення та керування курсами, які можна налаштувати для створення унікальних і доступних процесів викладання та навчання.

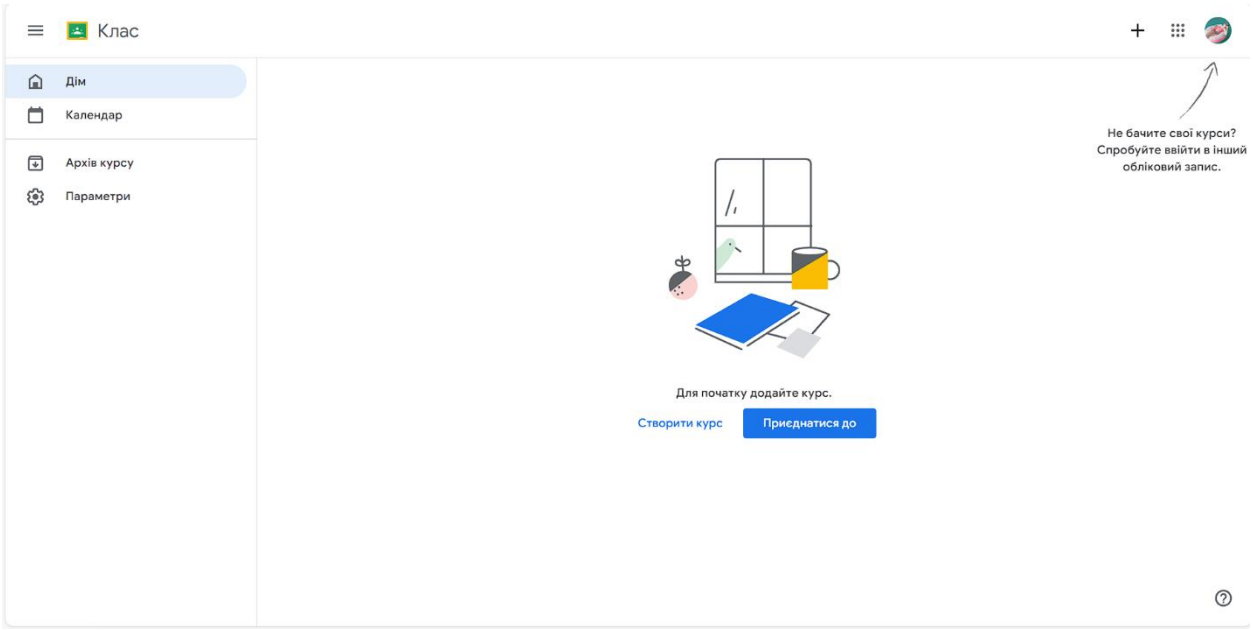
Розробники і викладачі можуть створювати та ділитися вмістом курсу за допомогою завдань , обговорень , модулів , тестів і сторінок . Вони також можуть вибрати сприяння спільному навчанню за допомогою співпраці , конференцій і груп . Залежно від налаштувань курсу, студенти можуть отримати доступ до цих областей у Canvas, щоб знайти навчальні матеріали та взаємодіяти з іншими користувачами курсу.

Canvas також дозволяє навчальним закладам і викладачам додавати державні та інституційні результати навчання до рубрик , щоб вимірювати та відстежувати розвиток навичок і досягнень у навчанні учнів. Крім того, розробники курсів можуть

використовувати інструмент імпорту курсів для масового завантаження попередніх пакетів курсів LMS та/або матеріалів курсу.

Google Classroom

Google Classroom – це набір онлайн-інструментів, який дозволяє вчителям ставити завдання, надсилати роботи учням, виставляти оцінки та повертати оцінки. Він був створений як спосіб відмовитися від паперу на уроках і зробити можливим цифрове навчання. Спочатку планувалося використовувати його з ноутбуками в школах, такими як Chromebook, щоб дозволити вчителю та учням ефективніше



обмінюватися інформацією та завданнями.[7]

Рисунок 2.1.3 - Google Classroom

Опції для викладачів:

викладачі можуть використовувати функціонал для організації відеозв'язку, створення й керування курсами та завданнями у режимі онлайн, додавання різних матеріалів до завдань, надання зворотного зв'язку учням у реальному часі, публікації оголошень та питань для учнів, а також можливість надсилання інформації батькам та законним представникам учнів.

Опції для учнів:

учні мають можливість відстежувати та виконувати завдання, перевіряти свої роботи на унікальність та отримувати коментарі та оцінки від викладача. Вони можуть обмінюватися інформацією та спілкуватися у стрічці курсу або через електронну пошту.

Опції для адміністратора:

забезпечення захисту конфіденційності даних та можливість налаштування доступу для користувачів. Адміністратори мають можливість налаштовувати параметри своїх курсів та списків учасників, а також додавати чи видаляти учасників. Підтримка надається цілодобово для вирішення будь-яких питань чи проблем.

3. ОЦІНКА ТА ПОРІВНЯННЯ ВИКОРИСТАНИХ ІНСТРУМЕНТІВ ТА ТЕХНОЛОГІЙ ДЛЯ ВЕБ-СЕРВІСУ

3.1. HTML & CSS

HTML розшифровується як HyperText Markup Language (мова розмітки гіпертексту). Це стандартна мова розмітки для створення веб-сторінок. Він дозволяє створювати та структурувати розділи, абзаци та посилання за допомогою елементів HTML, таких як теги та атрибути. [5]

HTML має масу варіантів використання, а саме:

- Веб-розробка. Розробники використовують HTML-код, щоб визначити, як браузер відображає елементи веб-сторінки, такі як текст, гіперпосилання та медіафайли.
- Інтернет-навігація. Користувачі можуть легко переміщатися та вставляти посилання між пов'язаними сторінками та веб-сайтами, оскільки HTML активно використовується для вбудовування гіперпосилань.
- Веб-документація. HTML дає можливість організовувати і форматувати документи, аналогічно Microsoft Word.

Середньостатистичний веб-сайт містить кілька різних HTML-сторінок. Наприклад, домашня сторінка, сторінка про компанію та сторінка контактів матимуть окремі HTML-файли.

HTML-документи – це файли, які закінчуються розширенням .html або .htm. Веб-браузер зчитує HTML-файл і відтворює його вміст, щоб користувачі Інтернету могли його переглянути.

Всі HTML-сторінки мають ряд HTML-елементів, що складаються з набору тегів і атрибутів. Елементи HTML є будівельними блоками веб-сторінки. Тег повідомляє веб-браузеру, де починається та закінчується елемент, тоді як атрибут описує характеристики елемента.

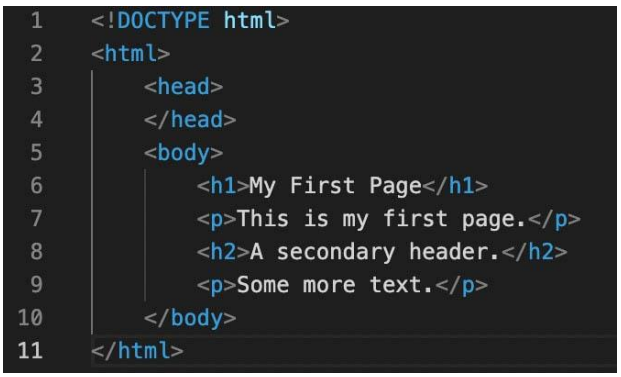


Рисунок 3.1.1 – HTML

Плюси HTML:

- простота вивчення та використання
- універсальна сумісність;
- поділ змісту та уявлення;
- SEO-дружність;
- швидке завантаження;
- розширюваність.

Мінуси HTML:

- обмежена інтерактивність;
- обмеження дизайну;
- залежність від браузера;
- відсутність можливості роботи в автономному режимі;

- питання безпеки;
- обмежені можливості роботи з мультимедіа.

CSS - це мова стилю сторінок, яка використовується для опису представлення документа, написаного на HTML або XML. CSS описує, як елементи повинні відображатися на екрані, на папері, в мові або на інших носіях.

CSS (каскадні таблиці стилів) використовується для стилізації та макета веб-сторінок — наприклад, для зміни шрифту, кольору, розміру та інтервалів вашого вмісту, розділення його на кілька стовпців або додавання анімації та інших декоративних функцій. [6]

```
/* A reference to a type */
span.ts span.type-ref {
    color: rgb(175, 0, 219) !important;
}

/* Signature details */
div.signature > table {
    border-collapse: collapse;
    border: thin darkgray solid;
    width: 60%;
}
```

Рис. 3.1.2 – Стилі CSS

Переваги CSS:

- ефективне використання часу за рахунок можливості створення шаблонів для майбутнього використання;
- забезпечує швидке завантаження веб-сторінок завдяки розділенню дизайну та контенту;
- простота використання та розуміння, що полегшує роботу розробників;
- великий набір стилів, що дозволяє легко налаштувати вигляд веб-сторінок;
- сумісність з різними типами пристроїв, що дозволяє забезпечити правильне відображення на різних екранах.

Недоліки CSS:

- не всі браузери повністю підтримують всі функції CSS, що може призвести до різниці в відображенні сторінок на різних платформах;
- вибір версії CSS може бути проблемою, оскільки різні версії мають відмінності, що ускладнює роботу розробників і користувачів;
- відсутність захисту коду: будь-який користувач може вносити зміни до CSS-коду, а також може стати об'єктом несанкціонованого доступу до сайту.

3.2. Вибір мови програмування

JavaScript — це кросплатформна, об'єктно-орієнтована скриптова мова, яка використовується для інтерактивності веб-сторінок (наприклад, зі складними анімаціями, клікабельними кнопками, спливаючими меню тощо). Існують також більш просунуті версії JavaScript на стороні сервера, такі як Node.js, які дозволяють додати більше функцій до веб-сайту, ніж завантаження файлів (наприклад, спільна робота в реальному часі між кількома комп'ютерами). Усередині хост-середовища (наприклад, веб-браузера) JavaScript може бути підключений до об'єктів свого середовища, щоб забезпечити програмний контроль над ними. [8]

JavaScript містить стандартну бібліотеку об'єктів, таких як , і , а також основний набір елементів мови, таких як оператори, керуючі структури та оператори.

Переваги JavaScript:

- підтримка більшістю браузерів, включаючи взаємодію з CSS і серверною частиною;
- мінімізація трафіку та навантаження на сервер при обробці веб-сторінок на користувацьких ПК;
- широкий вибір плагінів для взаємодії з JavaScript, що полегшує роботу для розробників;
- простота та компактність у використанні;
- зручний інтерфейс, зокрема наведення курсору, заповнення форм, активація кнопок.

Недоліки JavaScript:

- обмеженість в завантаженні та обробці файлів, що потребує додаткової уваги до безпеки;
- можливість виникнення недоліків у перетворенні даних під час роботи;
- відсутність віддаленого доступу до розробницьких продуктів;
- відкритий код може стати об'єктом атак зловмисників, що може призвести до серйозних проблем з безпекою.

3.3. Вибір фремворків

Фреймворки javascript – це потужні інструменти, здатні перетворити звичайні веб-сторінки на інтерактивні та приголомшливі користувацькі інтерфейси. Але чому вони такі важливі? Уявіть собі можливість розробляти складні веб-додатки з легкістю та ефективністю, скорочуючи час розроблення та покращуючи якість коду. У цій статті ми розглянемо, що таке JavaScript фреймворк, які існують популярні фреймворки і як вибрати відповідний фреймворк для ваших завдань. [9]

Мета фреймворку – надати розробникам зручний набір інструментів для створення ефективних і масштабованих веб-додатків. Фреймворки js забезпечують структуру й організацію коду, а також надають готові рішення для завдань, які часто зустрічаються, таких як управління станом, маршрутизація, обробка подій і взаємодія із сервером.

Функції фреймворків:

- керування станом: Автоматизоване управління станом додатка для простоти відстеження та оновлення даних без ручного втручання;
- маршрутизація: Управління маршрутизацією додатка для створення різних сторінок, обробки переходів та оновлення URL відповідно до стану додатка;
- компонентний підхід: Розбиття інтерфейсу на окремі компоненти для спрощення розробки, повторного використання коду та покращення структури додатка;

- взаємодія із сервером: Надання засобів для обміну даними з сервером, включаючи Ajax-запити та вбудовані API, для розробки динамічних додатків без перезавантаження сторінки.

3.4 React

React — це декларативна, ефективна і гнучка JavaScript-бібліотека, призначена для створення інтерфейсів користувача. Вона дозволяє компонувати складні інтерфейси з невеликих окремих частин коду — “компонентів”. [10] Розробляється Meta і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі вебзастосунки, які можуть динамічно оновлювати дані без необхідності перезавантаження сторінки. Основна мета React — забезпечити швидкість, простоту та масштабованість. React зосереджується виключно на користувацькому інтерфейсі застосунків. Він відповідає за видову частину в шаблоні модель-вид-контролер (MVC) і може використовуватися разом з іншими JavaScript бібліотеками або великими MVC-фреймворками, такими як AngularJS. Також React може працювати з надбудовами на його основі, які забезпечують функціонування частин вебзастосунків, що не стосуються користувацького інтерфейсу. [11]

Особливості:

1.Одностороння передача даних. Властивості передаються в рендерер компоненту подібно до атрибутів HTML тегів. Компонент не може змінювати отримані властивості безпосередньо, але може модифікувати їх за допомогою callback-функцій. Цей механізм називається «властивості донизу, події нагору».

2. Віртуальний DOM. React використовує віртуальний DOM замість того, щоб покладатися виключно на DOM браузера. Це дозволяє бібліотеці визначати, які частини DOM змінилися, порівнюючи поточну версію з віртуальним DOM, і таким чином ефективно оновлювати DOM браузера. Програміст працює зі сторінкою, наче

вона повністю оновлюється, але бібліотека самостійно вирішує, які компоненти необхідно оновити.

3. JSX. Компоненти React зазвичай пишуться з використанням JSX. Код на JSX компілюється у виклики методів бібліотеки React. Розробники можуть також писати на чистому JavaScript. JSX схожий на XHP, мову, розроблену у Facebook для розширення PHP.

4. Не лише рендеринг HTML в браузері. React використовується не тільки для рендерингу HTML у браузері. Наприклад, Facebook рендерить динамічні графіки в теги <canvas>. Netflix та PayPal використовують ізоморфне завантаження для рендерингу ідентичного HTML як на сервері, так і на клієнті.

5. Методи життєвого циклу. Методи життєвого циклу в ReactJS дозволяють розробникам обробляти дані на різних етапах життєвого циклу компоненту. [11]

3.5 Node.js

Node.js — це кросплатформне середовище для застосунків, написаних JavaScript із відкритим кодом. Це популярний інструмент майже для будь-якого проекту. [12]

Рисунок 3.5.1 – Логотип Node.js

Node.js запускає двигун V8 JavaScript, ядро Google Chrome, поза браузером. Це дозволяє Node.js бути дуже продуктивним.

Платформа працює в одному процесі без створення нового потоку для кожного запиту. Node.js надає набір асинхронних примітивів вводу/виводу у своїй стандартній бібліотеці, які запобігають блокуванню коду JavaScript, і загалом бібліотеки в Node.js написані з використанням неблокуючих парадигм, що робить поведінку блокування скоріше винятком, ніж нормою. [12]

Node.js здатний виконувати операції введення-виведення, такі як читання з мережі, доступ до баз даних чи файлової системи, без блокування потоків та зайняття циклів процесора на очікування. Замість цього, Node.js відновлює операції, коли отримує відповідь.

Це дозволяє платформі обробляти тисячі одночасних з'єднань з одним сервером, мінімізуючи проблеми паралельного управління потоками, що можуть призводити до помилок.

Node.js має унікальну перевагу, оскільки дозволяє мільйонам інтерфейсних розробників, які пишуть JavaScript для браузера, використовувати той же самий мовний інструментарій для розробки серверних додатків, не потребуючи вивчення зовсім іншої мови програмування.

Переваги:

- Node.js використовує подієву модель та однопотоковий асинхронний ввод-вивід, що дозволяє ефективно обробляти багато одночасних з'єднань без блокування потоків.
- Висока швидкість виконання завдяки використанню швидкого двигуна V8 JavaScript.
- Простота розробки завдяки використанню однієї мови програмування (JavaScript) для фронтенду і бекенду.
- Широка підтримка і активна спільнота розробників, що призводить до швидкого розвитку екосистеми і великого вибору модулів.

– Ідеальне використання для реалізації веб-додатків в реальному часі, таких як чати та стрімінгові сервіси.

Недоліки:

– Обмеження однопотокової моделі, яка може стати проблемою для інтенсивних обчислень або задач, що вимагають великого обсягу CPU.

– Високі вимоги до пам'яті, що може вплинути на продуктивність додатків при обробці великих обсягів даних або великих файлів.

– Нестабільність деяких модулів через швидкий розвиток екосистеми та велику кількість сторонніх пакетів.

– Більшість додатків, які залежать від певної бібліотеки, втрачають працездатність при її видаленні з npm.

3.6 Firebase

Firebase — це комплексна платформа розробки мобільних і веб-додатків, розроблена Google. Він надає розробникам набір хмарних інструментів і служб, призначених для створення, підтримки та вдосконалення програм. [13]

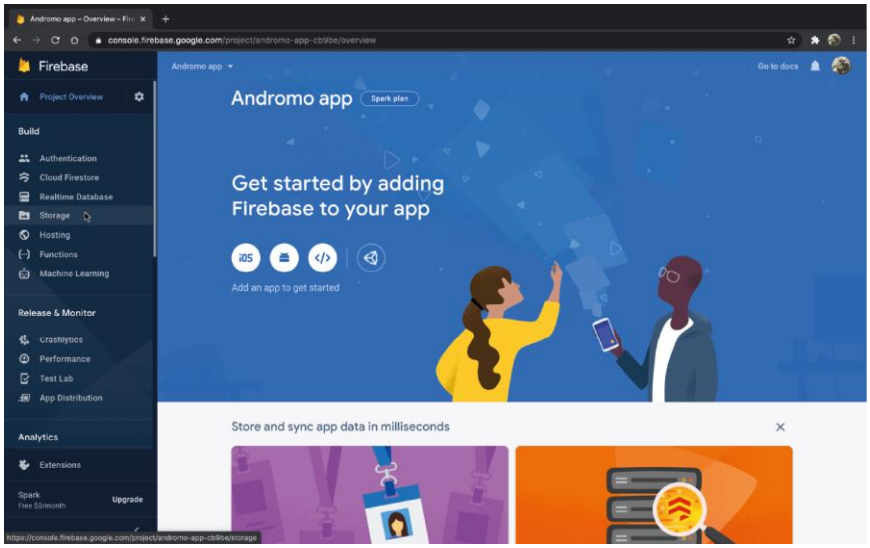


Рисунок 3.6.1 – Firebase

Firebase є платформою Backend-as-a-Service (BaaS), що надає розробникам готовий хмарний сервер для їхніх програм, абстрагуючи їх від складних серверних

операцій. Розробники можуть використовувати API Firebase для автентифікації користувачів, зберігання даних в хмарних базах даних, таких як Realtime Database або Cloud Firestore, та інших завдань, які зазвичай вимагають серверного програмування. Firebase SDK надає інтерфейс для взаємодії програм з цими хмарними службами на різних платформах, включаючи Android, iOS і веб.

Firebase автоматично масштабується зі зростанням бази користувачів і забезпечує надійність захисту даних, зберігаючи їх у своїх базах даних. Вона також включає в себе функції аналітики та інші інструменти, що полегшують розробку й ефективне функціонування додатків. Firebase діє як посередник між додатком та хмарними службами, забезпечуючи необхідні функції для його оптимальної роботи.

Функції:

1. Автентифікація користувачів.

Firebase надає API для аутентифікації користувачів з використанням різних методів, таких як email/password, соціальні мережі (Google, Facebook, Twitter), телефонні номери і т.д. Аутентифікацію Firebase (SDK) можна використовувати, щоб вручну інтегрувати один або кілька методів входу в програму.

2. Бази даних.

пропонує дві основні бази даних: Realtime Database, що працює на основі реального часу та орієнтований на JSON, і Cloud Firestore, який є розширеним, орієнтованим на документи додатком бази даних.

3. Хмарне зберігання.

Забезпечує хмарне зберігання для зберігання і керування медіа-файлами, такими як зображення або відео.

4. Аналітика.

Збирає дані про користувачів та їх поведінку в додатках для надання цінних висновків та аналізу.

5. Хмарні повідомлення (Cloud Messaging).

Надає зручний спосіб надсилання повідомлень на мобільні пристрої та веб-додатки.

6. Спрощений деплой.

Firebase Hosting дозволяє розгортати і хостити ваші веб-сайти і односторінкові додатки безпосередньо на хмарних серверах Firebase.

Переваги:

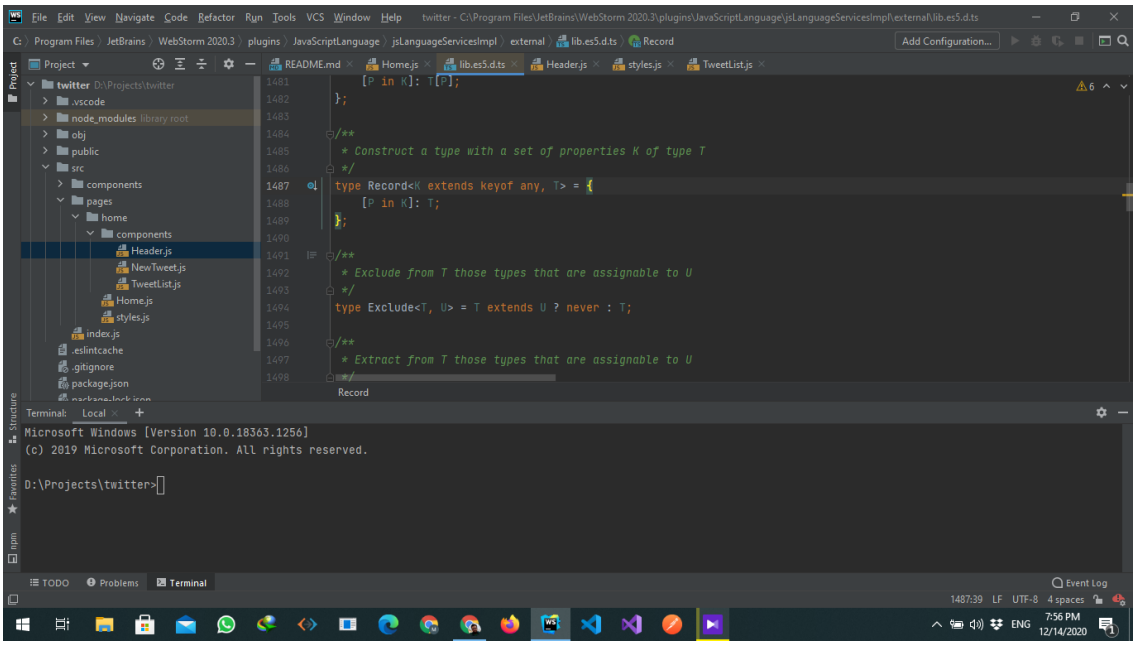
- Різноманітні методи автентифікації: електронна адреса та пароль, аутентифікація Google, Facebook та Github.
- Реальний час для синхронізації даних між різними пристроями і користувачами.
- Готові API, що спрощують розробку інтерфейсів додатків.
- Вбудований захист на рівні вузла даних, забезпечуючи безпеку даних на рівні розробника.
- Сховище файлів із підтримкою Google Cloud Storage для ефективного управління файлами.
- Статичний файлообмінник для швидкої і надійної доставки контенту.
- Гнучка інфраструктура, яка легко адаптується під змінні потреби додатку.

Недоліки Firebase:

- Обмежені можливості щодо складних запитів через модель потоку даних Firebase.
- Традиційні реляційні моделі даних не завжди застосовуються до NoSQL баз даних, що може бути обмеженням для деяких застосувань.

3.7 ID WebStorm

WebStorm - це інтегроване середовище розробки (IDE) від JetBrains. Він включає в



себе все необхідне для розробки JavaScript і TypeScript і дозволяє відразу приступити до кодування. WebStorm також дозволяє легко вирішувати найскладніші завдання. Незалежно від того, чи ви розв'язуєте конфлікти злиття Git, чи перейменовуєте символ у кількох файлах, це займе лише кілька кліків.

Рисунок 3.7.1 – IDE WebStorm

Це потужний інструмент, який надає розробникам усі необхідні можливості для ефективного створення, відлагодження та тестування веб-додатків.

Однією з ключових переваг WebStorm є його інтелектуальні функції підтримки кодування. IDE автоматично пропонує завершення коду, розпізнає синтаксичні помилки та надає поради з рефакторингу коду. Крім того, WebStorm забезпечує розширену навігацію, що дозволяє розробникам легко переходити між файлами та швидко знаходити необхідні елементи коду.

Ще однією важливою функцією є можливість налагодження (debugging) та тестування коду безпосередньо в середовищі розробки. WebStorm дозволяє встановлювати точки зупину (breakpoints), виконувати код крок за кроком, а також використовувати вбудовані інструменти для аналізу виразів та стану програми.

Особливістю є можливість інтеграції з різними засобами та сервісами, такими як системи контролю версій (наприклад, Git), різноманітні інструменти для тестування та збирання коду, а також різноманітні фреймворки та бібліотеки для розробки веб-додатків. Це робить WebStorm універсальним інструментом для роботи над проектами будь-якої складності.

Враховуючи всі ці можливості, WebStorm дозволяє розробникам ефективно працювати над веб-проектами, зосереджуючись на написанні високоякісного коду та швидкому вирішенні завдань.

Ця інтегрована система розробки має наступні переваги:

- можливість проведення рефакторингу коду;
- зручна навігація по коду;
- автоматичне доповнення коду;
- аналіз коду в реальному часі;
- миттєва перевірка змін завдяки інтеграції з системами контролю версій.

3.8 Figma

Figma - векторний онлайн-сервіс розробки інтерфейсів та прототипування з можливістю організації спільної роботи, що розробляється однойменною компанією. Працює у двох форматах: у браузері та як клієнтський застосунок на десктопі користувача. Зберігає онлайн-версії файлів, з якими працював користувач.

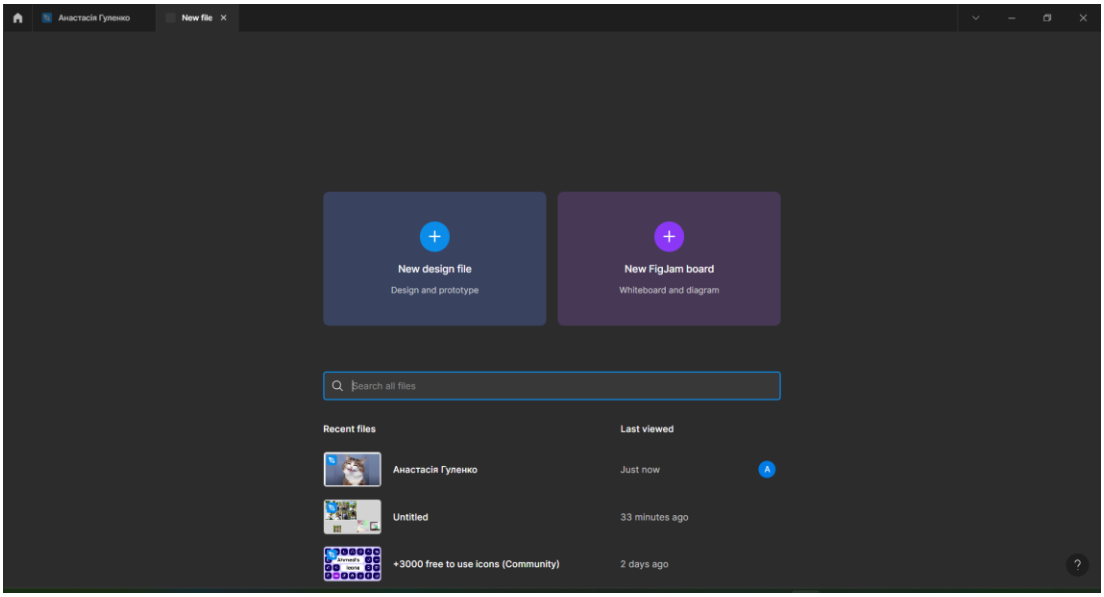


Рисунок 3.8.1 – Зовнішній вигляд Figma

Основні характеристики Figma включають:

- Хмарне рішення: Вся робота в Figma зберігається і синхронізується у хмарному середовищі, що дозволяє робити спільну роботу та швидко зберігати і синхронізувати зміни.
- Дизайн і прототипування: Figma надає інструменти для створення дизайну інтерфейсів, включаючи векторні графічні елементи, інструменти малювання, шари для організації контенту і багато іншого. Також підтримується прототипування, що дозволяє створювати взаємодію між екранами і показувати поведінку додатків.
- Спільна робота: Користувачі можуть працювати одночасно над одним проектом, вносячи зміни, коментуючи і взаємодіючи між собою у реальному часі.
- Інтеграція і експорт: Figma підтримує експорт до різних форматів, таких як PNG, JPG, SVG, а також можливість імпортувати інші файли. Також є можливість інтеграції з іншими інструментами розробки, такими як Zeplin, JIRA, Slack тощо.
- Мультиплатформеність: Figma доступний як для роботи в веб-браузері, так і для настільних платформ, включаючи Windows, macOS і Linux, а також мобільні пристрої на iOS і Android. [14]

Особливістю цього сервісу є те, що під час розробки дизайну, програма автоматично створює CSS-стилі для кожного об'єкта. Це дозволяє ефективніше і

точніше відтворювати стилі в реальному коді. Figma дозволяє експортувати CSS для створених елементів дизайну. Це особливо корисно для висновку розмірів, кольорів, шрифтів та інших стилів елементів, що можуть бути використані для подальшої реалізації в коді.

4. РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ

4.1 Загальний опис постанови задачі

Основним завданням даної роботи є створення веб-платформи для управляння навчанням , яка буде включати в себе всі основні складові навчальнго процесу, які потребуються для комфортної та ефективної організації навчання.

Майбутні можливості продукту:

У початку 2022 року, через повномаштабне вторгнення, більшість освітніх закладів перейшли на змішаний або дистанційний формат навчання, що призвело до високого попиту на сервіси підтримки дистанційного навчання. Навіть у разі повернення до офлайн формату, додатки такого типу залишаються актуальними. Основною проблемою цієї платформи є сильна конкуренція.

Ця платформа буде спрямована на забезпечення зручного та ефективного інструменту для адміністрування, планування та виконання навчальних програм для викладачів, а також сприяння комунікації та співпраці між учасниками навчального процесу - викладачами та учнями.
Дана веб-платформа буде надійним інструментом, призначеним для сприяння навчанню як в онлайн, так і в офлайн режимах. Вона повинна бути реалізована у вигляді веб-додатка і виконувати наступні завдання (функції):

Реєстрація користувачів:

Дуже важливо надати можливість учням, викладачам і адміністраторам реєструватися в системі. Під час реєстрації користувачам надається можливість введення своєї електронної адреси та пароля через окрему сторінку з відповідною

формою. Після введення правильних даних, сервер створює обліковий запис нового користувача у базі даних. Додатково, може бути надана опція реєстрації через корпоративну пошту для спрощення процесу;

Вхід:

Функція авторизації має найвищий пріоритет. Ця функція в веб-додатку є механізмом, який дозволяє користувачеві аутентифікуватися або увійти до свого облікового запису. Основна мета функції Вхід полягає в перевірці ідентифікаційних даних користувача, таких як ім'я користувача (або електронна пошта) і пароль, та наданні доступу до функціоналу, що вимагає авторизації. Якщо дані коректні - відвідувач переходить на сторінку з курсами, якщо ні – з’являється повідомлення про помилку.

Керування дисциплінами:

додати функціонал для створення та редагування курсів, призначення викладачів, адміністраторів та кураторів до певних груп. Функція керування до курсом також має найвищий пріоритет. Користувач повинен мати можливість приєднатись як учень до курсу, створити курс, як адміністратор чи викладач.

Управління розкладом занять:

розробити календарну систему для відображення розкладу занять, іспитів, відпусток, є одним із найголовніших завдань при створенні системи управління навчанням. Це дозволить зручно користуватися розкладом.

Створення оголошення:

ця функція дозволяє викладачам ефективно комунікувати зі своїми студентами, забезпечуючи їх інформацією та уточненнями, що можуть виникнути під час навчання.

Подання та відстеження домашніх завдань:

є критично важливою для ефективного навчання на веб-платформі. Вона надає можливість викладачам створювати завдання для студентів, а студентам - виконувати їх та відстежувати свій прогрес. Ця функція сприяє створенню ефективного

середовища для навчання та забезпечує якісну зворотну зв'язок між викладачами та студентами;

Оцінювання та звітність:

Ця функція допомагає забезпечити адекватну оцінку студентського прогресу та забезпечує перегляд інформації щодо навчального процесу для всіх зацікавлених сторін;

Повідомлення та сповіщення:

Функція повідомлень та сповіщень веб-платформи для управління навчанням забезпечує ефективний зв'язок між викладачами та студентами. Вона дозволяє автоматично надсилати інформацію про важливі події, такі як наближення дедлайнів, оновлення курсів, зміни в розкладі занять та інші актуальні повідомлення. Ця функція сприяє своєчасному та зручному сповіщенню користувачів про всі аспекти навчального процесу, що полегшує їхню участь та організацію навчання;

Однією із задач – є створення саме зручного та простого додатку для навчального процесу. Тому головною ідеєю відобразити на головній сторінці всі головні віджити для навчальноо процесу, такі як розклад, дисципліни, оголошення, підтримка тощо.

4.2 Архітектура програмного проекту

Структура та організація програмного забезпечення, яка визначається його компонентами, їхніми взаємозв'язками, а також принципами, що використовуються для їхньої побудови та взаємодії. Ця структура визначає основні архітектурні вузли, такі як клієнтські та серверні модулі, бази даних та інші сервіси, які використовуються в процесі розробки програмного продукту.

В кожній системі є свої основні концептуальні моделі. Однією із задач є визначення цих моделей.

Таблиця 4.2.1 – Концептуалізації інформаційної системи

Концептуальна модель	Характеристика
Відівдувач	Це користувач, який ще не доєднаний не до одної дисципліни.
Дисйципліна	Це самостійний процес, під час якого відбувається взаємодія і оцінювання учнів.
Учень	Це відвідувач додатку, якого дадає викладач дисципліни, який її створив.
Викладач	Це відвідувач, яким було додано дисципліну та додані завдання і цій дисципліні.
Оголошення	Можуть створювати викладачі та коментувати учні.
Розклад	Графік проведення занять.
Завдання	Це вся необхідна інформація для виконання завдання студентом.

Концептуальні моделі мають такі функції

Таблиця 4.2.2 – Функції концептуальної моделі

Концептуальна модель	Функції
Відвідувач	– Може реєструватися та увійти в систему. – Переглядає доступні курси та їхні матеріали. – Може приєднатися до курсу, який він обрав.
Дисципліна	– Містить інформацію про матеріали курсу, такі як лекції, вправи, тести тощо. – Представляє собою структурований набір текстових матеріалів, посилань тощо. – Може мати певний графік проведення занять,

	який відображається у розкладі.
Учень	– Може переглядати матеріали курсу та проходити завдання. – Бере участь у чаті для спілкування з іншими учасниками та викладачем. – Може відправляти запитання та отримувати допомогу від інших учасників курсу.
Викладач	– Додає матеріали курсу та створює завдання для учнів. – Може взаємодіяти з учнями через оглошення. – Оцінює роботу учнів та надає їм фідбек щодо їхнього прогресу.
Оголошення	– Викладачі можуть створювати оголошення з заголовком та текстом під завданням. – Учні можуть залишати коментарі під оголошеннями.
Розклад	– Містить інформацію про дати та часи проведення занять, а також їхні місця та формат (онлайн або офлайн). – Дозволяє учням та викладачам вчасно планувати свої заняття та взаємодіяти з іншими учасниками курсу.

Завдання	– Містять вимоги до виконання, дедлайни та вказівки щодо подання результатів. – Можуть бути різними за форматом, включаючи письмові роботи, тести, проекти тощо. – Після виконання учні отримують фідбек від викладача та оцінку за завдання.

Основно. Функцією сайта є це робота з дисциплінами. Тому актуальним є створити та проаналізувати діаграму взаємодії об’єктів по відношенню до курсів.

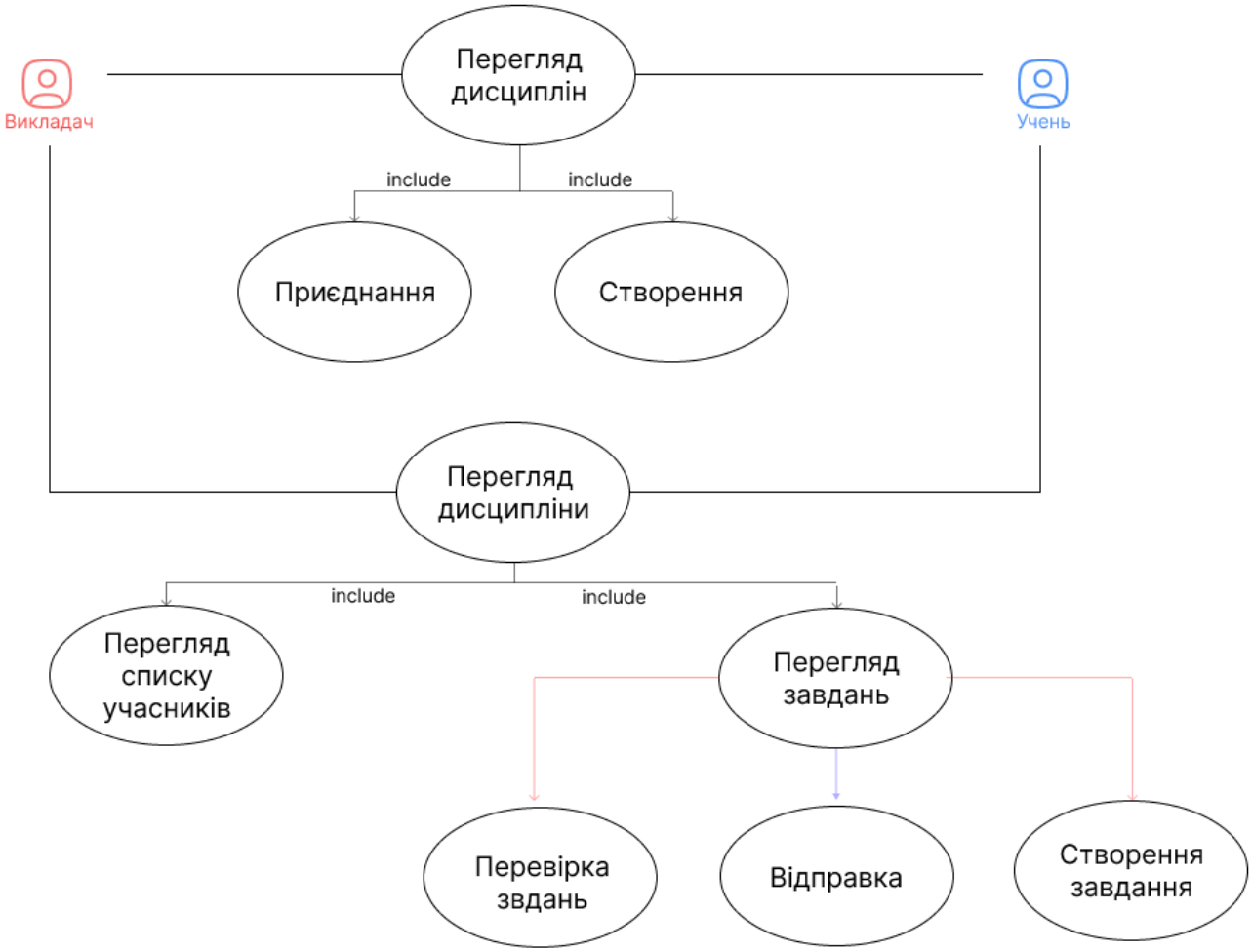


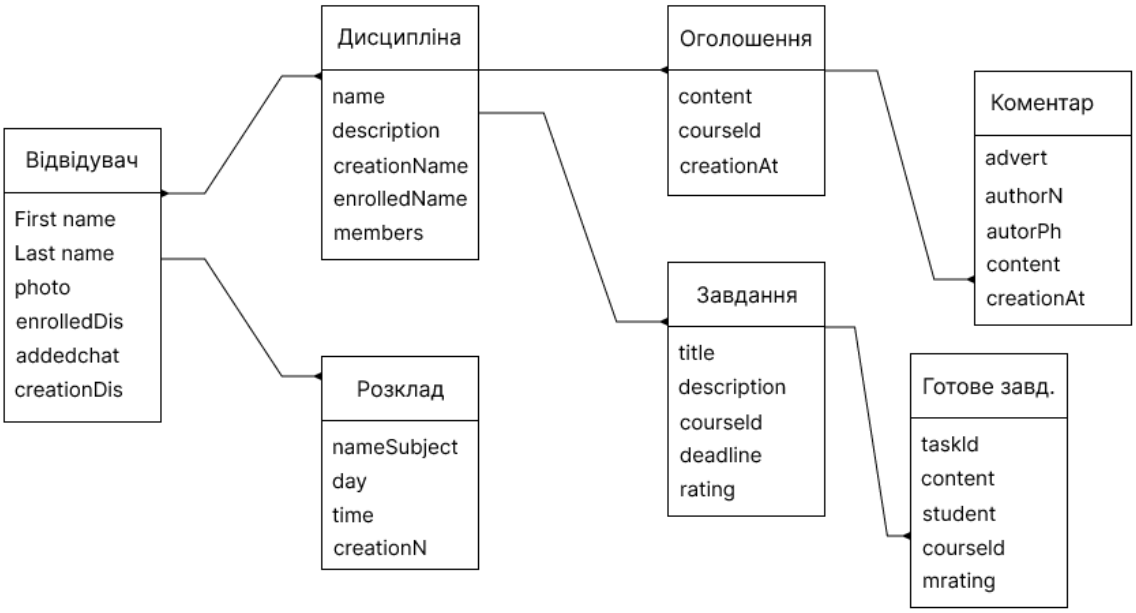
Рисунок 4.2.1 – Діаграма функцій дисципліни

4.3 Розробка моделі та опису бази даних

Розробка моделі та опису бази даних передбачає побудову структури, яка відповідає потребам системи. У даному випадку база даних буде побудована на основі NoSQL Firestore Database. Це означає, що дані будуть зберігатися у форматі NoSQL, що дозволяє швидку і гнучку роботу з ними.

Структура бази даних повністю задовольняє потреби та тематику проєкту. Вона буде складатися з колекцій та документів, що відображатимуть основні сутності системи, такі як користувачі, дисципліни, завдання, коментарі тощо. Кожна сутність матиме свої атрибути та відношення з іншими сутностями, що відображатимуть логіку та взаємозв'язки в системі.

Розробка моделі бази даних включає в себе аналіз потреб системи, визначення сутностей та їх атрибутів, а також взаємозв'язків між ними. На базі цього було створено сітку бази даних, яка буде використовуватися для реалізації бази даних у



середовищі NoSQL Firestore Database.

Рисунок 4.3.1 – Модель БД

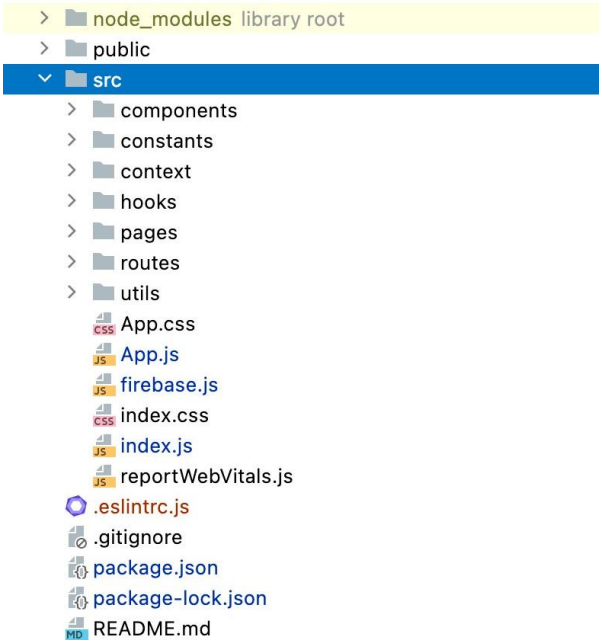
4.4 Організація файлів проєкту

Нище буде описана організація файлів створеного веб додатку:

- public – тека з файлами загального доступу;
- src – тека з усіма засобами проекту:
  - index.js – це основний JavaScript файл, який використовується для ініціалізації і запуску додатка, його приєднують до HTML;
  - context – ця тека містить файли, які стосуються React контекстів. Вони використовуються для передачі даних по всьому дереву компонентів без необхідності передачі їх через проміжні компоненти;
  - components – тека components містить файли React компонентів. Це основна частина проекту, де зберігаються всі компоненти інтерфейсу, які використовуються для побудови веб-сторінок;
  - routes – у цій теці зазвичай зберігаються файли, які визначають маршрути додатку і вказують, які компоненти React відображати при переході за певними URL адресами;
  - App.css – використовується для зберігання CSS стилів, які використовуються в компоненті App.js та інших компонентах додатку;
  - constants – тека, призначена для зберігання незмінних значень, які використовуються багато разів, таких як константи для API URL, кольорів, розмірів і регулярних виразів;
  - hooks – тека, призначена для зберігання React хуків, які є переиспользуемыми функциями, що розширюють можливості компонентів React;
  - firebase.js – створений для забезпечення з'єднання з Firebase та включає додаткові функції, необхідні для коректної роботи з цією платформою;
  - App.js – це основний компонент, який відповідає за відображення та управління іншими компонентами у додатку;

- utils – тека, де зберігаються помічні функції та утиліти, які використовуються в проекті для різних завдань;
- index.css – файл, в якому зберігаються маштабні стилі, які застосовуються до всієї платформи;
- pages – тека, де зберігаються окремі вигляди додатка, які відповідають за відображення конкретних екранів користувачам.

- eslintrc.js – файл конфігурації для ESLint, який використовується для налаштування правил статичного аналізу коду в проекті;
- .gitigore – визначає, які компоненти не потрібно включати до системи контролю версій Git;
- package.json – є центральним файлом проекту в середовищі Node.js. В ньому зберігається інформація про проект, його налаштування, пакети та скрипти для управління проектом;
- package-lock.json – файл, який створюється автоматично при встановленні пакетів з npm. Він містить точні версії всіх пакетів, які встановлені у вашому проекті, а також



їх залежності.

Рисунок 4.4.1 – Організація файлів проєкту

4.5 Характеристика ключових процесів обробки

Головною частиною проєкту є функція приєднання до БЗ. На рисунку 4.5.1

```
const firebaseConfig = {
  apiKey: 'AIzaSyCkcUIPkZWnk5ur4-lcHen2WJ18Lha8mmY',
  authDomain: 'eduspace-15506.firebaseapp.com',
  projectId: 'eduspace-15506',
  storageBucket: 'eduspace-15506.appspot.com',
  messagingSenderId: '895198289171',
  appId: '1:895198289171:web:4e0156630f43f1bb890ee6',
};

const app = firebase.initializeApp(firebaseConfig);
const auth = app.auth();
const db = app.firestore();
```

зображено функцію для встановлення з'єднання з Firebase.

Рисунок 4.5.1 – Приєднання БД та Firebase

Другою основною функцією є – це реєстрація та вхід.

```
const signUpWithEmailAndPassword = async (
    email,
    password,
    firstName,
    lastName,
    onError,
) => {
  try {
    const {user} = await auth.createUserWithEmailAndPassword(email, password);
    const tempUser = {...user, displayName: `${firstName} ${lastName}`};
    await checkUser(tempUser);
  } catch (err) {
    onError(err.message);
  }
};
```

Рисунок 4.5.2 – Реєстрація

Після успішної реєстрації у відвідувача з'являєтеся змога скористатися входом в додаток, залогінившись за допомогою Email та власноруч створеного паролю. Для даної

```
const [login,, loading, signInError] = useSignInWithEmailAndPassword(auth);
const handleSubmit = (e) => {
  e.preventDefault();
  setError( value: null);
  login(email, password);
};
```

функції треба застосувати хук.

Рис. 3.6 – Вхід

Дуже зручною і не менш важливою є фіункція входу (створення акаунту) через Google. В цьому нам допоможе Firebase Authentication. Це сервіс, який надає інструменти для аутентифікації користувачів у додатках з використанням Firebase.

```
const googleProvider = new firebase.auth.GoogleAuthProvider();
// Sign in and check or create account in firestore
const signInWithGoogle = async (onError) => {
  try {
    const response = await auth.signInWithPopup(googleProvider);
    await checkUser(response.user);
  } catch (err) {
    onError(err.message);
  }
};
```

Рисунок 4.5.3 – Авторизація за допомогою Google

Все ж головними функціями є формування, приєднання та дії в дисциплінах. На рисунках 4.5.4 та 4.5.5 зображено функції, які виконує формування та доєднання до дисципліни.

```
const joinClass = async (courseId) => {
  try {
    setIsJoinModalOpen( value: false);
    const classRef = await db.collection( collectionPath: 'courses').doc(courseId).get();
    if (!classRef.exists) {
      return alert(`Class doesn't exist, please provide correct ID`);
    }
    await classRef.ref.collection( collectionPath: 'users').add({
      name: user.name,
      joiningDate: moment().format( format: 'MMM Do YY'),
      courseId,
      studentUid: user.uid,
    });

    const classData = await classRef.data();
    await user.ref.collection('enrolledCourses').add({
      creatorName: classData.creatorName,
      id: courseId,
      name: classData.name,
      description: classData.description,
    });
    alert(`Enrolled in ${classData.name} successfully!`);
  } catch (err) {
    console.error(err);
    alert(err.message);
  }
};
```

Рисунок 4.5.4 – Доєднання до дисципліни

```
const createClass = async (courseName, courseDescription) => {
  setIsCreateModalOpen( value: false);
  try {
    const {id} = await db.collection( collectionPath: 'courses').add({
      creatorName: user.name,
      name: courseName,
      description: courseDescription,
      users: [],
    });
    await user.ref.collection('createdCourses').add({
      id,
      name: courseName,
      description: courseDescription,
    });
  } catch (err) {
    alert(`Cannot create class - ${err.message}`);
  }
};
```

Рисунок 4.5.5 – Формування дисципліни

На головній сторінці курсу є можливість формування викладачем оголошення. На це оголошенні учні мають можливість залишати коментар, так само як і викладачі.

```
const addPost = async (editorState) => {
  await db.collection( collectionPath: 'posts').add({
    content: convertContentToHTML(editorState),
    courseId,
    createdAt: firebase.firestore.FieldValue.serverTimestamp(),
  });
};
```

Ці коментарі можна видаляти авторам.

Рисунок 4.5.6 – Формування оголошення

```
const addComment = async (editorState) => {
  await db.collection( collectionPath: 'comments').add({
    content: convertContentToHTML(editorState),
    postId: post.id,
    createdAt: firebase.firestore.FieldValue.serverTimestamp(),
    authorName: user.name,
    authorPhoto: user.photo,
  });
};
const deletePost = async () => {
  await db.collection( collectionPath: 'posts').doc(post.id).delete();
  const tempComments = await db.collection( collectionPath: 'comments')
      .where( fieldPath: 'postId',  opStr: '==', post.id).get();
  tempComments.docs.forEach((doc : QueryDocumentSnapshot<DocumentData> )=>{
    doc.ref.delete();
  });
};
```

Рисунок 4.5.7 –Коментарі

Завданням викладача є додавання нових завдань для виконання учнями. За

```
const createTask = async () => {
  try {
    await db.collection( collectionPath: 'tasks').add({
      name,
      maxRating: parseInt(maxRating,  radix: 10),
      description: convertContentToHTML(description),
      deadline,
      courseId,
    });
    setName( value: '');
    setDeadline( value: null);
    setMaxRating( value: 30);
    setDescription(EditorState.createEmpty());
    handleClose();
  } catch (err) {
    alert(err.message);
  }
};
```

допомогою функції, зображеної на рисунку 4.5.8, може це реалізувати.

Рисунок 4.5.8 – Додавання нових задач

Завданням учня є виконувати задачі, поставлені викладачем. За допомогою

```
const sendWork = (taskId) => {
  try {
    db.collection( collectionPath: 'completedTasks').add({
      taskId,
      student: user.name,
      studentUid: user.uid,
      date: firebase.firestore.FieldValue.serverTimestamp(),
      content: convertContentToHTML(content),
      courseId,
    });
    setContent(EditorState.createEmpty());
    setIsModalOpen( value: false);
  } catch (e) {
    alert(e.message);
  }
};
```

функції на рисунку 4.5.9 він може це реалізувати.

Рисунок 4.5.9 – Надсилання зробленого завдання

```
const evaluate = () => {
  try {
    db.collection( collectionPath: 'completedTasks').doc(id).update( data: {
      rating: newRating,
    });
    setIsModalOpen( value: false);
  } catch (e) {
    alert(e.message);
  }
};
```

Відповідь від викладача – оцінка.

Рисунок 4.5.10 – Оцінка зданого завдання

4.6 Логотип

Логотип — це ключовий елемент інтерфейсу веб-додатку, який візуально ідентифікує бренд або компанію. Він використовується для створення першого враження інтернет-користувача і може включати у себе графічний елемент, текст або комбінацію обох.

Ключові функції логотипу:

- Ідентифікація;

- Візуальне сприйняття;
- Створення асоціативного зв’язку/

Елементами успішного логотипу повинні бути: правильно підібраний колір, ненав’язливій шрифт, правильно реалізована концепція в залежності від цільової аудиторії та спрямування платформи.
В нашому випадку, логотип не повинен бути нав’язливим. Але при цьому бути помітним та привабливим. Головними кольорами було вибрано синій та блакитний,

вони ідеально підходять для логотипу корпоративного додатку.

Рисунок 4.6.1 - Логотип

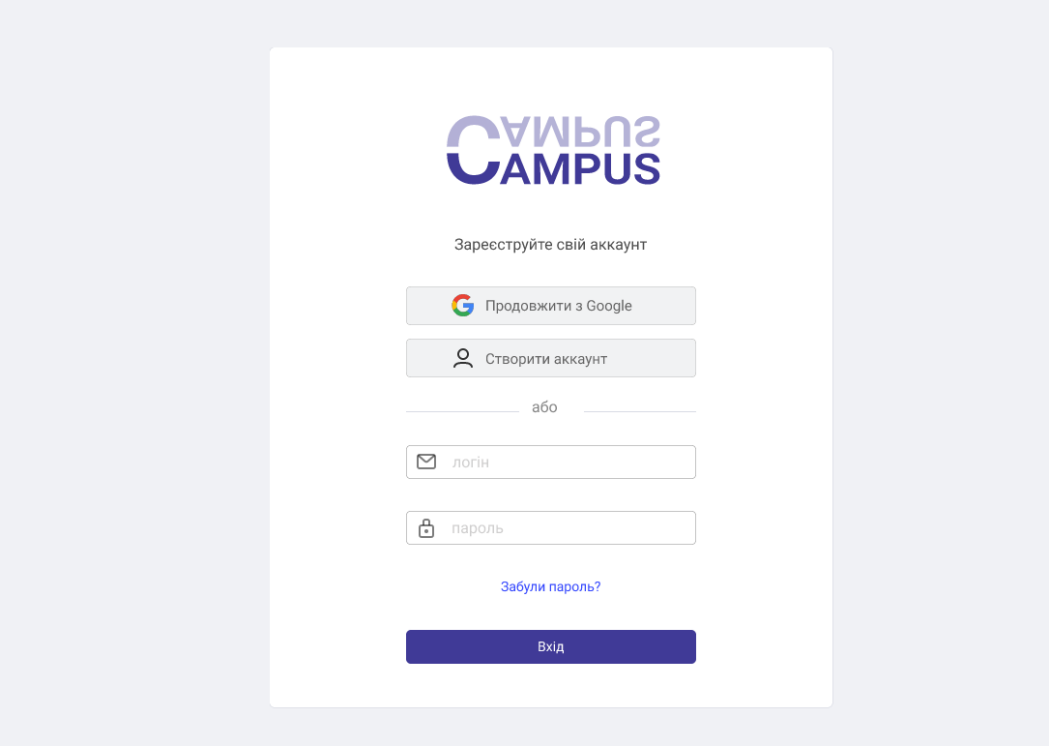
4.7 Інтерфейс веб-платформи

Інтерфейс веб-платформи має дуже важливу роль у візуальному впливі на користувача додатком. Інтерфейс користувача має містити влучні не нав’язливі кольори, шрифти, зображення, кнопки панелі навігації, списки, меню , які дозволяють користувачам взаємодіяти з програмним продуктом.

Головною частиною оформлення є UX (User Experience) дизайн. Важливим є саме створення зручного та простого інтерфейсу, який зможе охопити всі аспекти взаємодії користувача з продуктом, включаючи його враження, реакції та взаємодію. Саме UX дизайн забезпечує ефективність, легкість використання та задоволення від використання додатку.

Не менш важливу роль відіграє UI (User Interface) дизайн. Перше, що робе користувач, відвідуючи сайт – це вдивиться на естетичне оформлення елементів. UI дизайн відповідає за те, як інформація та функції представлені і доступні для користувача. Його основна мета — зробити взаємодію з продуктом якомога простішою, зручною та естетичною.

Перше, що бачить відвідувач, зайшовши на платформу – це вхід та реєстрація. На сторінці входу зображені поля для входу та кнопки з посиланнями на реєстрацію



(Рисунок 4.7.1).

Рисунок 4.7.1 – Вхід

Натиснувши кнопку «Створити акаунт», відвідувач переходить до вкладки з реєстраційною формою. Випадку, якщо користувач ввів не коректо пошту або взагалі нічого не ввів, з’являється віко помилки (Рисунок 4.7.3).

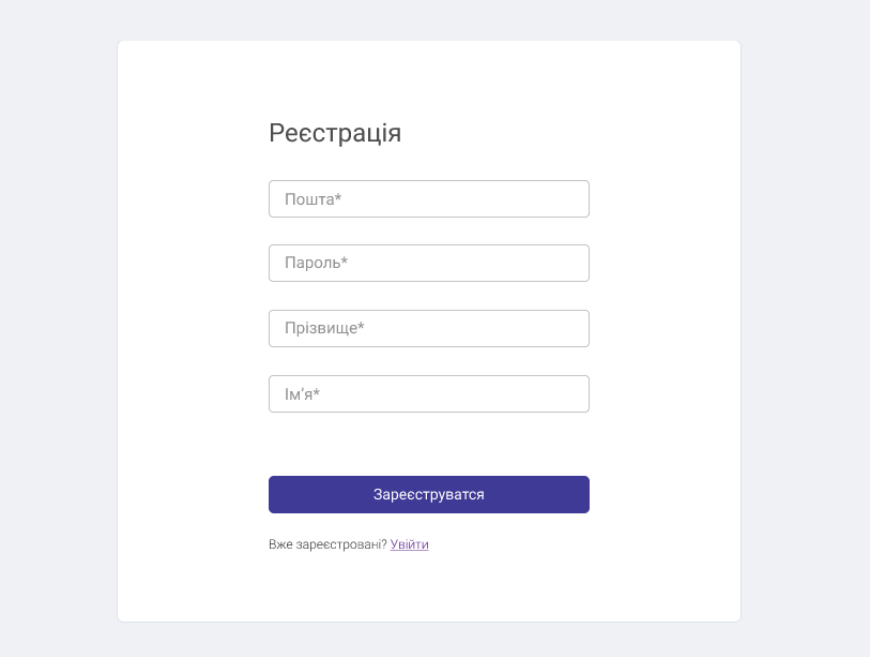


Рисунок 4.7.2 – Реєстраційна форма

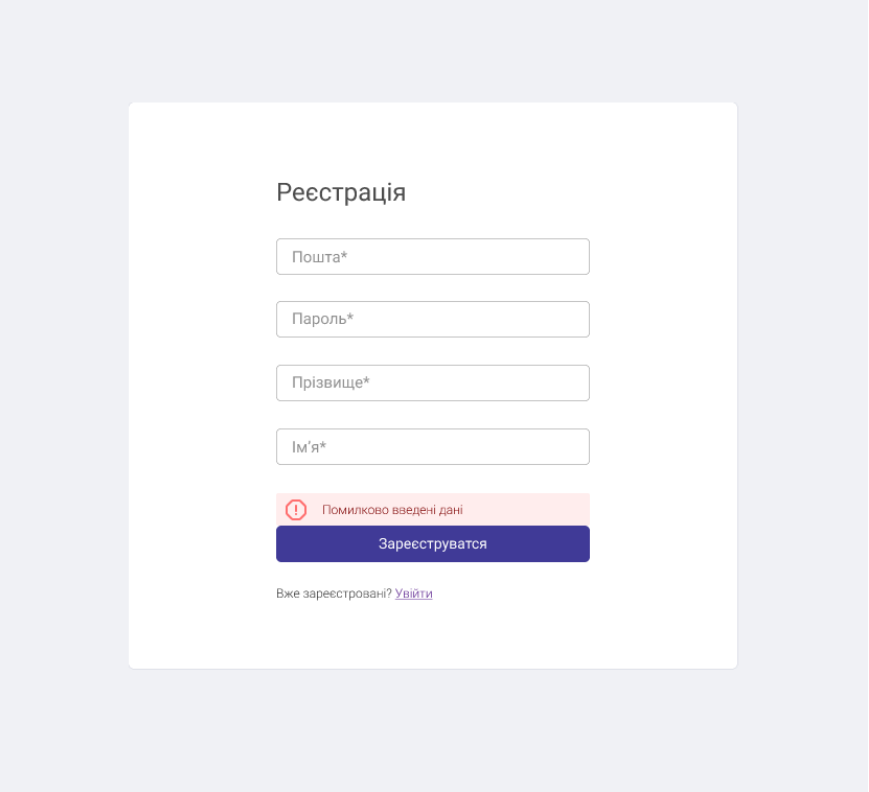


Рисунок 4.7.3 – Реєстраційна форма без введених даних

Також можна побачити помилку при неправильному введені логіну чи паролю.

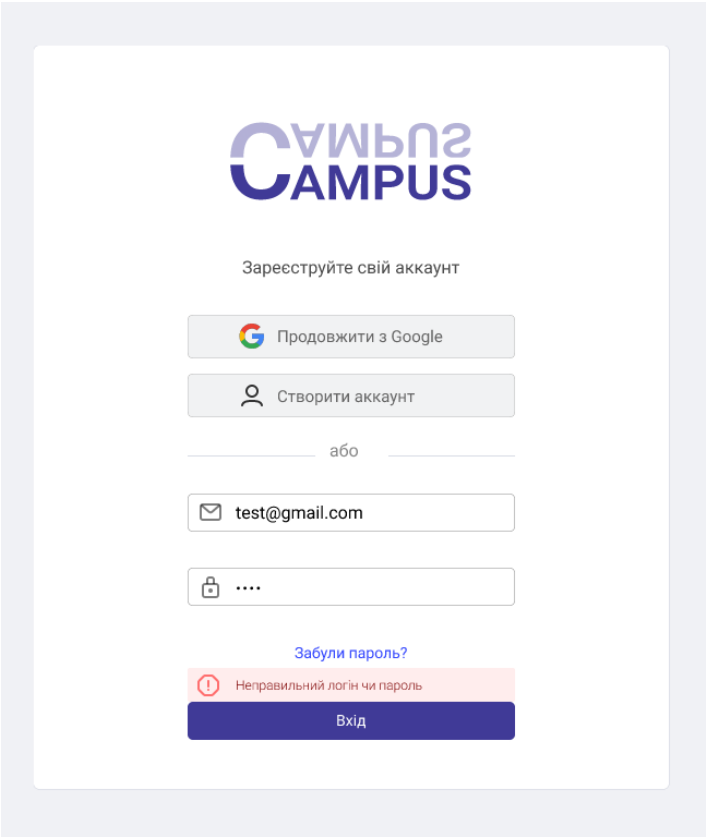


Рисунок 4.7.4 – Не правильно введені дані

Зайшовши, відвідувач потрапляє на головну сторінку, на якій він може побачити розклад та недавно відкриті дисципліни. Особливістю є розміщене зліва меню з кнопками бистрого доступу (Рисунок 4.7.5). Також, користувач може змінити мову.

У викладачів головна сторінка відрізняється тим, о вони можу створювати курс.

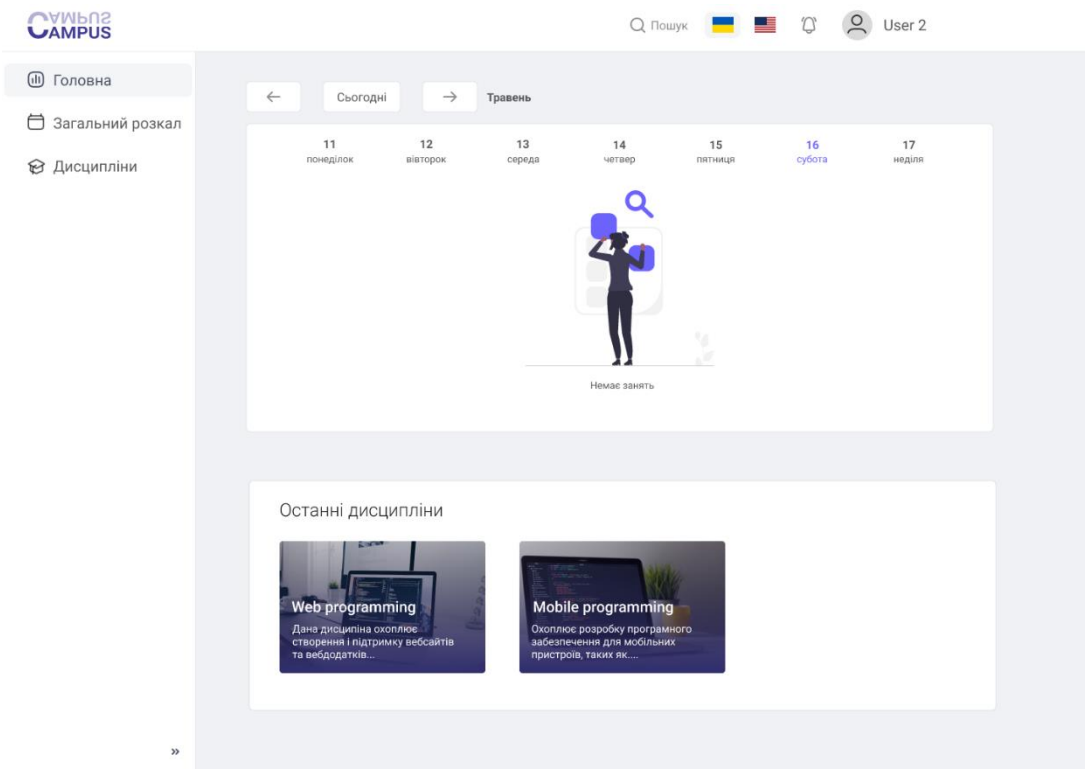


Рисунок 4.7.5 – Головна сторінка для учня

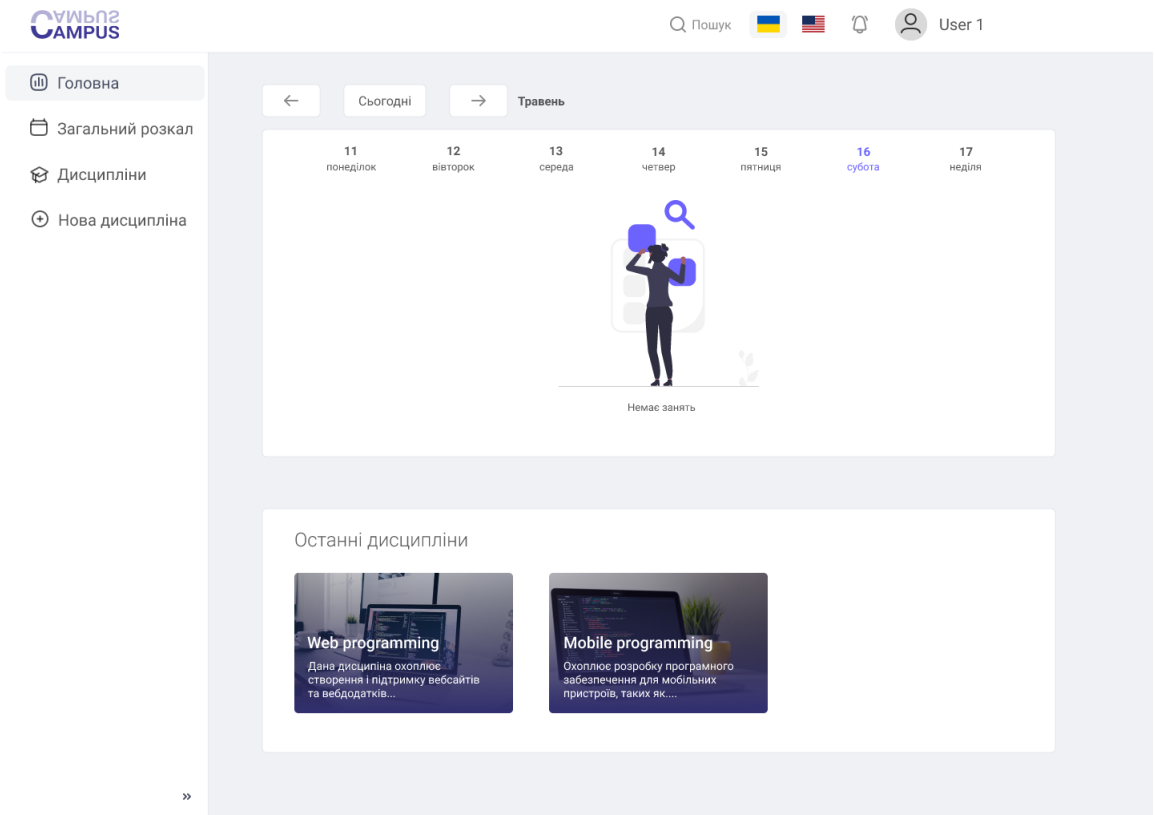
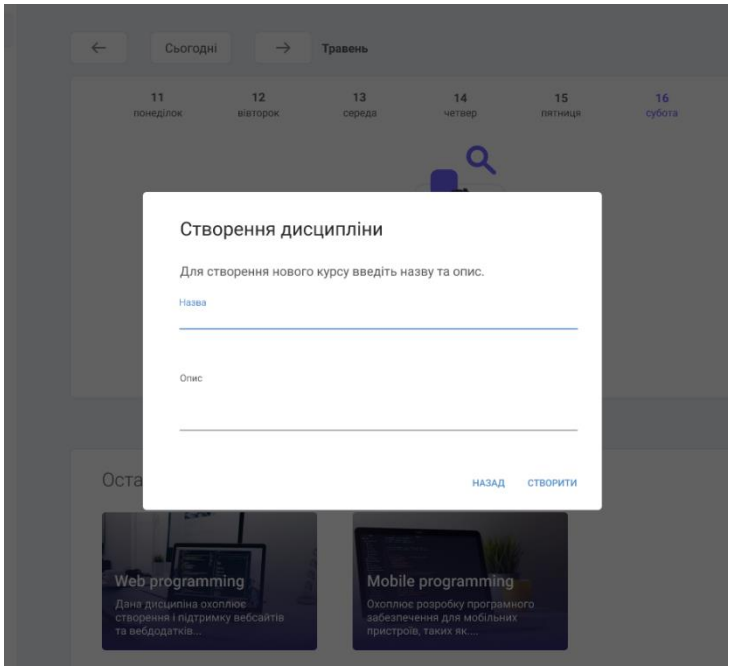


Рисунок 4.7.6 – Головна сторінка для викладача

На рисунку 4.7.7 ви можете побачити вікно створення курсу. Для створення



дисципліни треба ввести її назву та опис.

Рисунок 4.7.7 – Створення дисципліни

Також викладач, після створення курсу, може додавати учнів, знаючи їх Ім’я та Прізвище, яке вже є базі зареєстрованих користувачів.

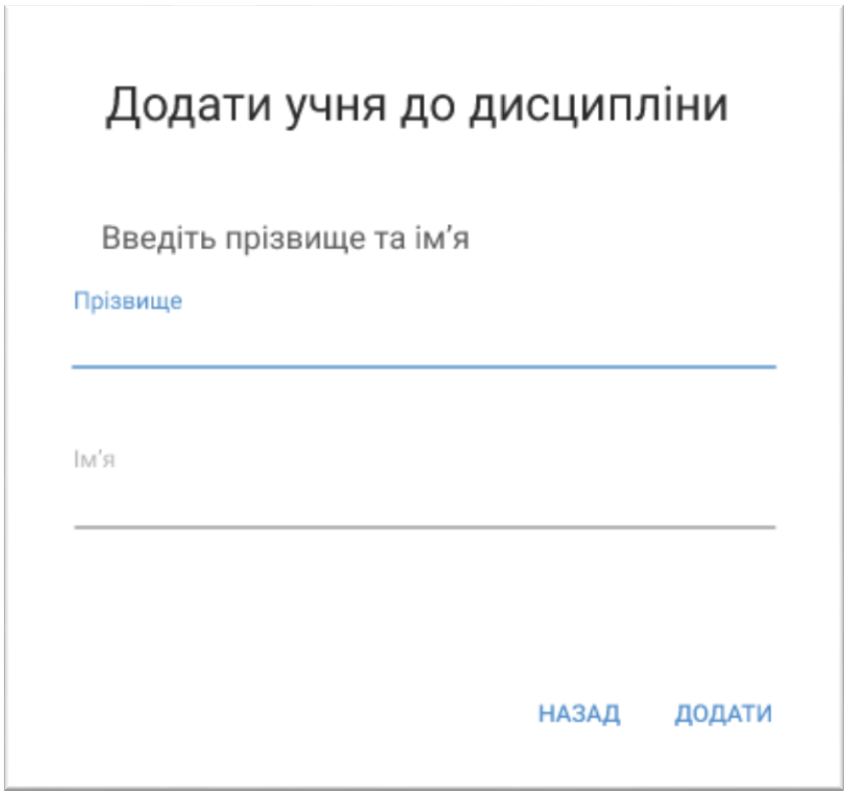
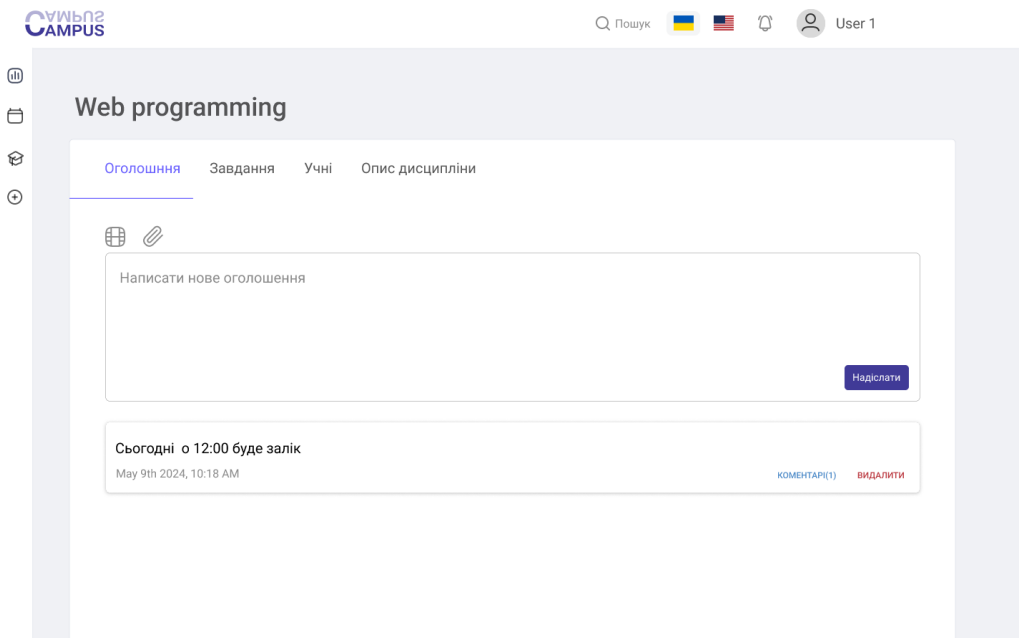


Рисунок 4.7.8 – Додавання учня

Після вже створеної дисципліни. Можна потрапити на саму сторінку курсу, на якій зображено декілька вкладок: оголошення, завдання, учні, опис дисципліни.

Головна сторінка – це сторінка, де викладач може надсилати оголошення учням,



а вони можуть коментувати.

Рисунок 4.7.9 – Оголошення

Завдання – це вкладка, на якій викладач може додавати нові завдання, оцінювати та видаляти. А учні надсилають вирішені завдання.

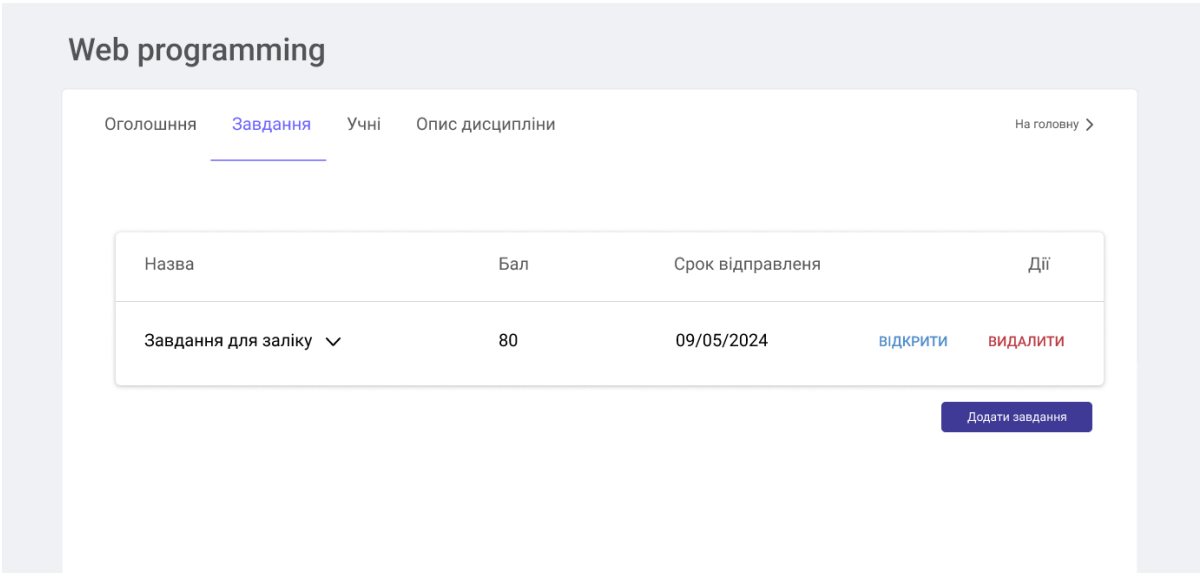


Рисунок 4.7.10 – Вкладка завдання для викладачів

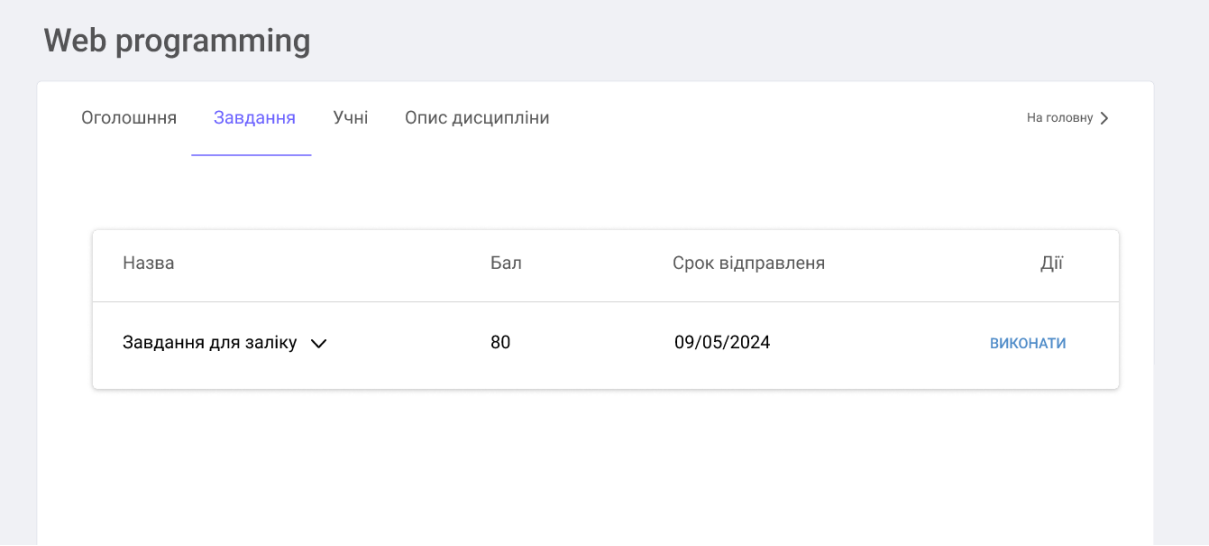


Рисунок 4.7.11 – Вкладка завдання для учнів

На рисунку 4.7.12 зображено вікно створення завдання. Викладач повинен заповнити назву завдання, та додати заме задання в опис. Також треба проставити максимальну кількість балів за завдання та термін.

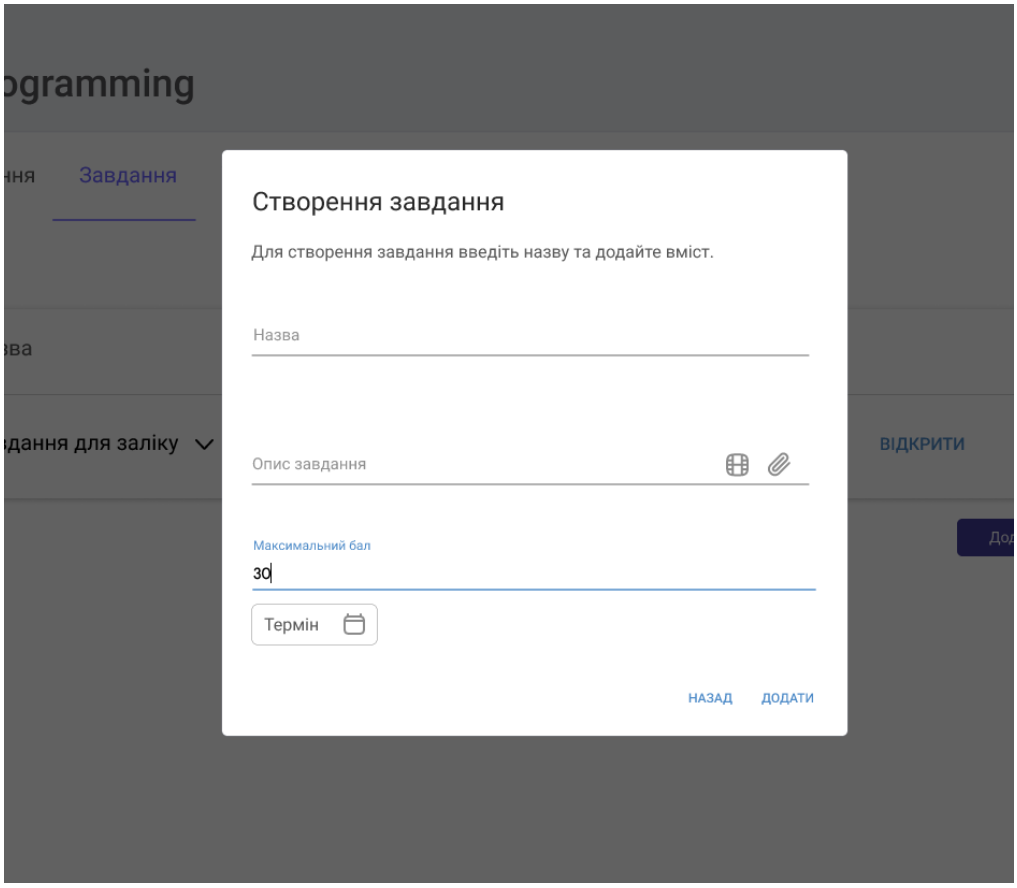


Рисунок 4.7.12 – Створення завдання

На рисунку 4.7.13 ви можете побачити як студент вже здав завдання, але воно ще не оцінене викладачем.

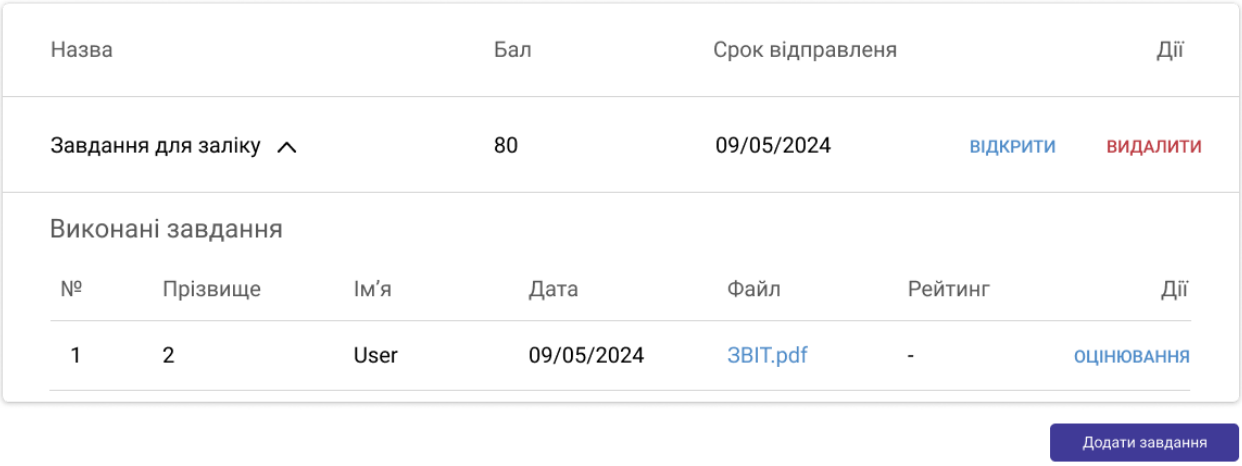
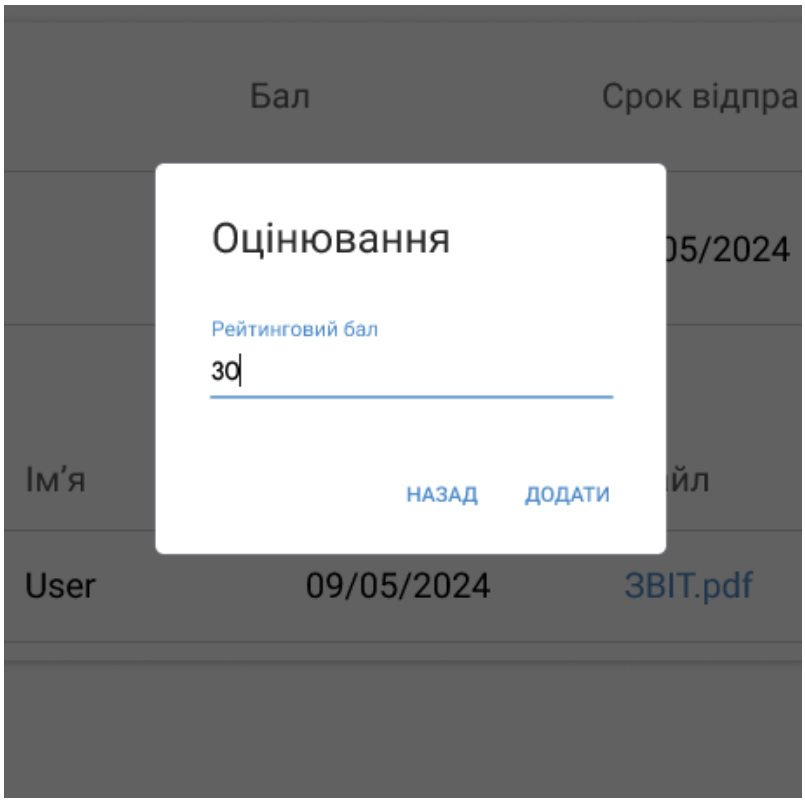


Рисунок 4.7.13 – Здане завдання

Викладач може перевірити здане завдання, відкривши файл, який був закріплений студентом, після перевірки він може оцінити, проставивши рейтинговий



бал.

Рисунок 4.7.14 – Оцінювання

На рисунку 4.7.14 зображено вже оцінене завдання.

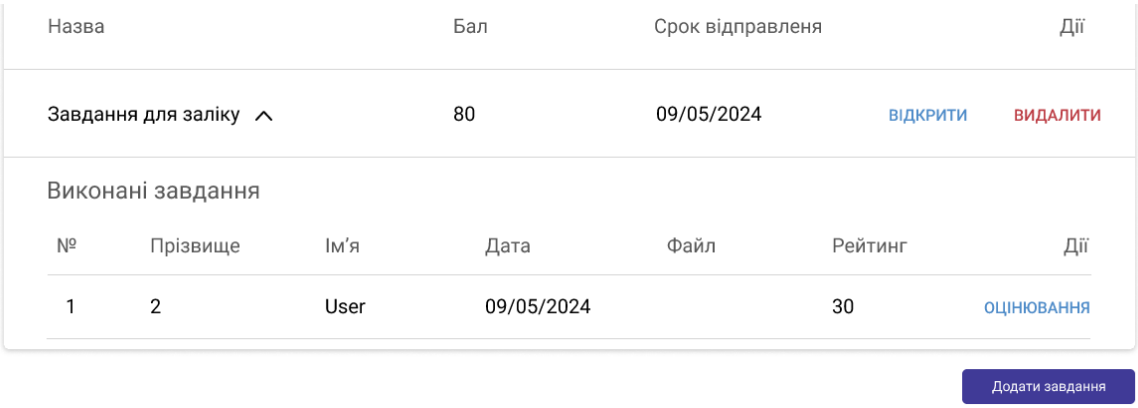


Рисунок 4.7.15 – Оцінене завдання

На вкладці «Учні» можна побачити повний список учнів з їх загальними балами.

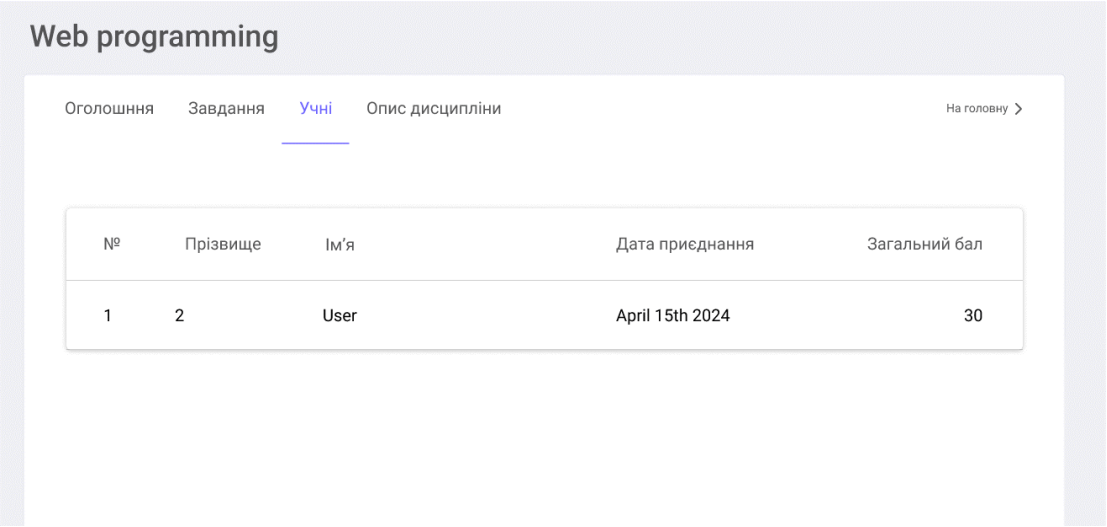


Рисунок 4.7.16– Вкладка «Учні»

На рисунку 4.7.17 зображений загальний розклад. Він створений у простому форматі, де відображаються дні тижня, дисципліни, хто їх створив та час. Нажавши на дисципліну, користувачі можуть перейти на сторінку курсу.

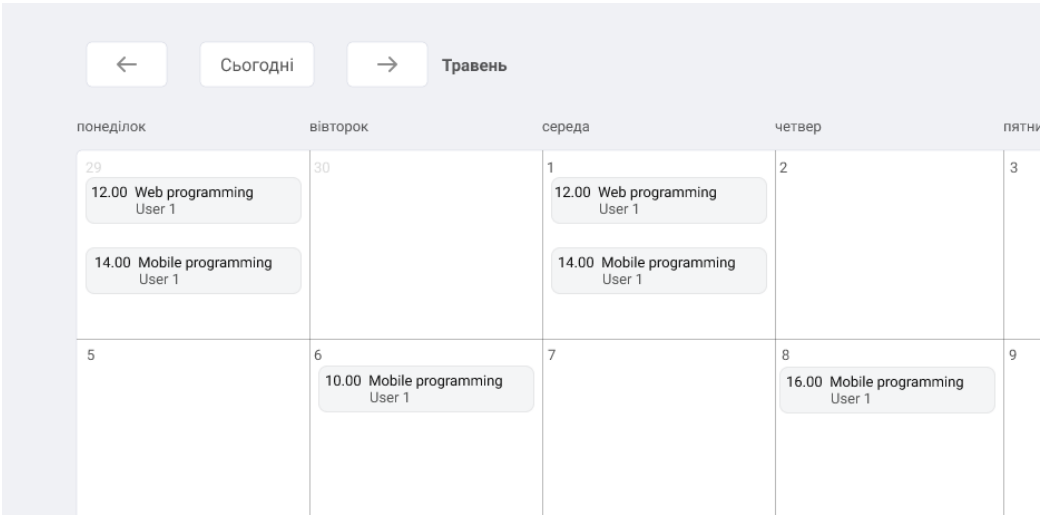


Рисунок 4.7.17 – Розклад

ВИСНОВОК

В процесі розробки було створено систему управління навчальним процесом, яка включає в себе функції керування користувачами, курсами, завданнями, оголошенням і розкладом. В основі архітектури лежить модульний підхід, що дозволяє легко розширювати та модифікувати функціональні можливості системи.

Процес розробки включав аналіз вимог користувачів, вибір технологій, постанову задачі та реалізацію інтерфейсу користувача .Використання сучасних інструментів та технологій, таких як React для фронтенду та Firebase для зберігання та управління даними, дозволило створити ефективну та надійну платформу.

Додаток, який було розроблено, відповідає всім вимогам і стандартам, які були встановлені на етапі формулювання завдання.

Платформа має інтуїтивно зрозумілий інтерфейс, що дозволяє зручно навігуватися та використовувати всі його функціональні можливості. В результаті, користувачі можуть з легкістю виконувати основні завдання, пов'язані з управлінням навчальним процесом.

Отримані результати підтверджують ефективність розробленої системи, яка відповідає сучасним вимогам до платформ для навчання та управління навчальним процесом. Впровадження розробленої платформи сприяє поліпшенню якості навчання та зручності користувачів, забезпечуючи централізований доступ до необхідної інформації та взаємодію між всіма учасниками навчального процесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Learning Management System (LMS) [Електронний ресурс] – Режим доступу: https://www.valamis.com/hub/what-is-an-lms
2. The Evolution and Diffusion of Learning Management Systems: The Case of Canvas LMS [Електронний ресурс] – Режим доступу: https://ohiostate.pressbooks.pub/drivechange/chapter/the-evolution-and-diffusion-of-learning-management-systems-the-case-of-canvas-lms/
3. About Moodle [Електронний ресурс] – Режим доступу: https://docs.moodle.org/403/en/About_Moodle
4. What is Canvas? [Електронний ресурс] – Режим доступу: https://community.canvaslms.com/t5/Canvas-Basics-Guide/What-is-Canvas/ta
5. What Is HTML? Hypertext Markup Language Basics Explained [Електронний ресурс] – Режим доступу: https://www.hostinger.com/tutorials/what-is-html#How_Does_HTML_Work
6. CSS [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS
7. Google Classroom can make teaching and learning easier for students and teachers and here's how [Електронний ресурс] – Режим доступу: https://www.techlearning.com/features/what-is-google-classroom
8. Introduction [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction
9. Огляд популярних JavaScript фреймворків [Електронний ресурс] – Режим доступу: https://foxminded.ua/freimvorky-javascript/
10. React. Початок роботи [Електронний ресурс] - Режим доступу: https://uk.legacy.reactjs.org/docs/getting-started.html
11. React [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/React

12. Introduction to Node.js [Електронний ресурс] – Режим доступу: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

13. What is firebase [Електронний ресурс] – Режим доступу: https://www.resmo.com/blog/what-is-firebase

14. Figma [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Figma

ДОДАТКИ

ДОДАТОК А. ПРОГРАМНИЙ КОД

```
import React from 'react';

import ReactDOM from 'react-dom';
import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';
import {UserProvider} from
'./context/userContext';

import  '../node_modules/react-draft-wysiwyg/dist/react-draft-wysiwyg.css';

ReactDOM.render(

    <React.StrictMode>

      <UserProvider>

        <App />

      </UserProvider>

     </React.StrictMode>,
     document.getElementById('root')
     ,);

reportWebVitals();

import React from 'react';

import {BrowserRouter, Link, Navigate, Route, Routes} from 'react-router-
dom'; import PublicLayout from './components/templates/PublicLayout';

import {HOME, LOGIN} from './constants/routes'; import {publicRoutes} from
'./routes/routes'; import PublicRoute from './routes/PublicRoute';

import ProtectedLayout from
'./components/templates/ProtectedLayout'; import ProtectedRoute from
'./routes/ProtectedRoute';

import Courses from
'./pages/Courses'; import Course
from './pages/Course';

function App() {
  return (

    <BrowserRouter>

      <Routes>

        <Route path="/" element={<PublicLayout />}>

          <Route index element={<Navigate to={LOGIN} replace />} />

          {publicRoutes.map(({path, element}) => (

            <Route

              key={path}
              path={`/${path}`}
```

```
              element={<PublicRoute>{element}</PublicRoute>}/>))}
          </Route>
          <Route path="/" element={<ProtectedLayout />}>
            <Route
              path={HOME}
              element={<ProtectedRoute><Courses/></ProtectedRoute>}
            />
            <Route path="course"><Route path=":courseId" element={
               <ProtectedRoute><Course /></ProtectedRoute>
            } />
          </Route>
          </Route>
          <Route path="*" element={<Link to="login">Go to log in</Link>} />
        </Routes>
      </BrowserRouter>);}

export default App;

import firebase from
'firebase'; const
firebaseConfig = {
  apiKey: 'AIzaSyCkcUIPkZWnk5ur4-lcHen2WJ18Lha8mmY',
  authDomain: 'eduspace-15506.firebaseapp.com',
  projectId: 'eduspace-15506',
  storageBucket: 'eduspace-15506.appspot.com',
  messagingSenderId: '895198289171',
  appId: '1:895198289171:web:4e0156630f43f1bb890ee6',};

const app = firebase.initializeApp(firebaseConfig);
const auth = app.auth();
const db = app.firestore();

const checkUser = async (user) =>
  { const querySnapshot = await db
      .collection('users')
      .where('uid', '==', user.uid)
      .get();
  if (querySnapshot.docs.length === 0) {
    // create a new user
    await
      db.collection('users').add({
      uid: user.uid,
```

```
      name: user.displayName,
      photo:
      user.photoURL,});}};

const googleProvider = new firebase.auth.GoogleAuthProvider();

const signInWithGoogle = async (onError)
  => { try {

    const response = await auth.signInWithPopup(googleProvider);
    await checkUser(response.user);

  } catch (err) {
    onError(err.message);}};

const signUpWithEmailAndPassword = async (
    email,

    password,
    firstName,

    lastName,
    onError,) =>
    {

  try {const {user} = await auth.createUserWithEmailAndPassword(email, password);
  const tempUser = {...user, displayName: `${firstName} ${lastName}`}; await
  checkUser(tempUser);} catch (err) { onError(err.message);}};

const logout = () => {
  auth.signOut();};

xport {app, auth, db, signInWithGoogle, signUpWithEmailAndPassword, logout};
import React from 'react';

import LoginForm from '../components/organisms/LoginForm';

function Login()
  { return (

    <LoginForm />

  );}

  export default Login; import React from 'react';

import SignupForm from '../components/organisms/SignupForm';


function Login()
  { return (

    <SignupForm />);}

export default Login;

import React, {useEffect, useState} from
'react'; import {

  Container, Divider, Grid,

} from '@mui/material';

import H1 from '../components/atoms/H1';
import H2 from '../components/atoms/H2';
```

```
import CourseCard from '../components/molecules/CourseCard';
import AddCourseCard from
'../components/molecules/AddCourseCard'; import {db} from
'../firebase';

import DialogModal from
'../components/organisms/DialogModal'; import useUser from
'../hooks/useUser';

import JoinCourseModal from
'../components/organisms/JoinCourseModal'; import moment from
'moment';

function Courses() {

  const [isCreateModalOpen, setIsCreateModalOpen] =
  useState(false); const [isJoinModalOpen, setIsJoinModalOpen] =
  useState(false); const {user} = useUser();

  const [enrolledCourses, setEnrolledCourses] = useState([]);

  const [createdCourses, setCreatedCourses] = useState([]); const [loading,
  setLoading] = useState(false);

  const fetchCourses = async () =>
    { try {setLoading(true);

      await user.ref.collection('enrolledCourses')

          .onSnapshot((snapshot) => {

            const courses =
              snapshot?.docs.map((doc)=>{ return
              doc.data();});

            setEnrolledCourses(courses);});

      await user.ref.collection('createdCourses')

          .onSnapshot((snapshot) => {

            const courses =
              snapshot?.docs.map((doc)=>{ return
              doc.data();});

            setCreatedCourses(courses);});

    } catch (error) {
      console.error(error.message);

    } finally {
      setLoading(false);}};

  useEffect(() =>
    {
    fetchCourses(
    );}, [user]);

  const joinClass = async (courseId) =>
    { try {

      setIsJoinModalOpen(false);
```

```
      const classRef = await
      db.collection('courses').doc(courseId).get(); if
      (!classRef.exists) {

        return alert(`Class doesn't exist, please provide correct ID`);}

      await
        classRef.ref.collection('users').add({
        name: user.name,

        joiningDate: moment().format('MMM Do
        YY'), courseId,

        studentUid: user.uid,});

      const classData = await classRef.data();

      await user.ref.collection('enrolledCourses').add({
        creatorName: classData.creatorName,

        id: courseId,

        name: classData.name,

        description: classData.description,});

      alert(`Enrolled in ${classData.name} successfully!`);

    } catch (err) {
      console.error(err);
      alert(err.message);}};>

export default useUser;

import React, {useEffect, useState, createContext} from 'react';
import {useAuthState} from 'react-firebase-hooks/auth';

import {auth, db} from
'../firebase'; import {node} from
'prop-types';

export const UserContext = createContext({
  user: null,

  setUser: ()=>{},});

export const UserProvider = ({children})
  => { const [user, setUser] =
  useState(null); const [authUserAuth] =
  useAuthState(auth);

  useEffect(() => {

    if (authUserAuth) {

      const unsubscribe = db.collection('users')

          .where('uid', '==', authUserAuth.uid)

          .onSnapshot((snapshot) => {

            const docRef = snapshot?.docs[0];
            setUser({...docRef.data(), ref:
            docRef.ref});});
```

```
      return () => {
        unsubscribe();};

    } else {
      setUser(nul
      l)},
      [authUserAu
      th]);

  return (

    <UserContext.Provider
      value={{user,
      setUser}}>

      {children}

    </UserContext.Provider>);};

UserProvider.propTypes = {
  children:
  node.isRequired,};

import {createContext} from 'react';

export const TeacherContext = createContext(false); export const LOGIN = '/login';

export const SIGNUP = '/signup';

export const HOME = '/home';

export const COURSE = '/course/:courseId';

import React from
'react'; import {Box}
from '@mui/material';

import {Outlet} from 'react-router-dom';

import PublicHeader from
'../organisms/PublicHeader'; import Container from
'../atoms/Container';

function
  PublicLayout() {
  return (

    <div>

      <PublicHeader />

      <Container>

        <Box
          sx={{
          mx:
          'auto',
          mt: 4,

          maxWidth: 500,}}>

          <Outlet />

        </Box>
```

```
      </Container>
    </div>
  );}

export default PublicLayout;

import React from
'react'; import
{Outlet,}
  from 'react-router-dom';

import ProtectedHeader from '../organisms/ProtectedHeader';

function ProtectedLayout() {
  return (
    <div>
      <ProtectedHeader />
      <Outlet />
    </div>);}

export default ProtectedLayout;

import * as React from 'react';

import Table from '@mui/material/Table';

import TableBody from
'@mui/material/TableBody'; import TableCell
from '@mui/material/TableCell';

import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';

import TableRow from
'@mui/material/TableRow'; import Paper
from '@mui/material/Paper'; import {db}
from '../../firebase';

import {useEffect, useState} from
'react'; import {useParams} from 'react-
router-dom';

export default function Users() {
  const [users, setUsers] = useState([]);
  const [completedTasks, setCompletedTasks] = useState([]);
  const {courseId} = useParams();
  useEffect(async ()=>{
    const tempUsers = await db.collection(`courses/${courseId}/users`).get();
    setUsers(tempUsers.docs.map((doc)=>doc.data()));
    const tempCompletedTasks = await
db.collection('completedTasks').where('courseId', '==',
courseId).get();
    setCompletedTasks(tempCompletedTasks);
```

```
  }, [courseId]);

  const getCurrentRating = (studentUid)
    => { let rating = 0;

    completedTasks.docs?.filter((doc)=>doc.data().studentUid ===
studentUid).forEach((task)=>{

      rating+=parseInt(task.data()?.rating);});

    return rating;};

/* eslint-disable react/prop-types,max-len */
import React, {useEffect, useState} from
'react'; import {db} from '../../firebase';

import TableRow from '@mui/material/TableRow';
import TableCell from
'@mui/material/TableCell'; import IconButton
from '@mui/material/IconButton';

import KeyboardArrowUpIcon from '@mui/icons-material/KeyboardArrowUp';
import KeyboardArrowDownIcon from '@mui/icons-
material/KeyboardArrowDown'; import moment from 'moment';

import {
  Button,
  Dialog,
  DialogActi
  ons,
  DialogCont
  ent,

  DialogContentTe
  xt,
  DialogTitle,

} from '@mui/material';

const TeacherTask = ({task}) => {

  const [open, setOpen] = React.useState(false);

  const [isDeleteModalOpen, setIsDeleteModalOpen] = useState(false);

  const [isDescriptionModalOpen, setIsDescriptionModalOpen] =
  useState(false); const deleteTask = async () => {

    await
    db.collection('tasks').doc(task.id).delete();
    const tempCompletedTasks = await

db.collection('completedTasks').where('taskId', '==', task.id).get();
    tempCompletedTasks.docs.forEach((doc)=>{

      doc.ref.delete();});};

        </DialogContent>

        <DialogActions>

          <Button onClick={()=>setIsDescriptionModalOpen(false)}>Cancel</Button>

        </DialogActions>
```

```
      </Dialog>
    </React.Fragment>);};

function TeacherTasks() {
  const [isModalOpen, setIsModalOpen] = useState(false);
  const {courseId} = useParams();
  const [tasks, setTasks] = useState([]);
  useEffect(()=>{
    db.collection('tasks').where('courseId', '==', courseId)
        .orderBy('deadline',
          'asc').onSnapshot((snapshot)=>{ const tempTasks
          = [];
          snapshot.docs.forEach((doc)=>tempTasks.push({...doc.data(),  id:
doc.id}));
          setTasks(tempTasks);});
  }, [courseId]);
  return (<>
    {tasks.length === 0 ? <p style={{textAlign: 'center', color:
'gray', marginBottom: '12px'}}>No tasks</p> : <TableContainer
component={Paper}>
      <Table aria-label="collapsible table">
        <TableHead>
          <TableRow>
            <TableCell/>
            <TableCell>Name</TableCell>
            <TableCell align="right">Max rating</TableCell>
            <TableCell align="right">Deadline</TableCell>
            <TableCell align="right">Actions</TableCell>
          </TableRow>
        </TableHead>
        <TableBody>
          {tasks.map((task) => (
            <TeacherTask key={task.id} task={task} />
          ))}</TableBody>
      </Table>
    </TableContainer>}
    <Box sx={{display: 'flex', justifyContent: 'end', p: 2}}>
```

```
            <Button variant="contained"
                        onClick={()=>setIsModalOpen(true)}> Create new
                        task

            </Button>

          </Box>

          <CreateTaskModal isOpen={isModalOpen}
      handleClose={()=>setIsModalOpen(false)}/>

</>);};

      export default TeacherTasks;

      import * as React from 'react';
      import {useContext} from
      'react';

      import {TeacherContext} from
      '../../context/TeacherContext'; import StudentTasks from
      './StudentTasks';

      import TeacherTasks from './TeacherTasks';

      export default function Tasks() {

        const isTeacher =
        useContext(TeacherContext); if (isTeacher)
        return <TeacherTasks />;

        else return <StudentTasks />;}

      /* eslint-disable react/prop-types,max-len */
      import React, {useEffect, useState} from
      'react'; import {useParams} from 'react-
      router-dom'; import {db} from
      '../../firebase';

      import TableContainer from '@mui/material/TableContainer';
      import Paper from '@mui/material/Paper';

      import Table from '@mui/material/Table';

      import TableHead from
      '@mui/material/TableHead'; import TableRow from
      '@mui/material/TableRow'; import TableCell from
      '@mui/material/TableCell'; import TableBody
      from '@mui/material/TableBody'; import {


      const StudentTasks = () => {

        const [tasks, setTasks] = useState([]);

        const [completedTasks, setCompletedTasks] = useState([]);
        const {courseId} = useParams();

        const {user} = useUser();

        useEffect(()=>{

          db.collection(`tasks`).where('courseId', '==',
              courseId).onSnapshot( (snapshot) => {
```

```
          const tempTasks = [];
          snapshot.docs.forEach((doc)=>tempTasks.push({...doc.data(), id:doc.id}));

        },setTasks(tempTasks);

    db.collection('completedTasks').where('studentUid', '==',
user.uid).onSnapshot((snapshot) => {

      const tempCompletedTasks = [];
      snapshot.docs.forEach((doc)=>tempCompletedTasks.push({...doc.data(), id:

doc.id}));

      setCompletedTasks(tempCompletedTasks);});

  }, [courseId]);

  const isSent = (taskId) => !!completedTasks.find((task)=>task.taskId ===
taskId);

  const getRating = (taskId) => {

    return completedTasks.find((task)=>task.taskId === taskId)?.rating;};

  return (<>};

export default StudentTasks;

import React, {useContext, useEffect, useState} from
'react'; import H1 from '../atoms/H1';

import './Stream.css';

import {db} from '../../firebase';

import {useParams} from 'react-router-dom';

import {convertContentToHTML} from
'../../utils/convertContentToHTML'; import firebase from 'firebase';

import Post from './Post';

import TextEditor from '../molecules/TextEditor';

import {TeacherContext} from '../../context/TeacherContext';

const Stream = () => {

  const isTeacher = useContext(TeacherContext);
  const {courseId} = useParams();

  const [posts, setPosts] = useState([]);
  const addPost = async (editorState) => {

    await db.collection('posts').add({

      content:
      convertContentToHTML(editorState),
      courseId,

      createdAt: firebase.firestore.FieldValue.serverTimestamp(),

    });

  };
```

```
  useEffect(() => {
    db.collection('posts').where('courseId', '==',
    courseId)

        .orderBy('createdAt',
          'desc').onSnapshot((snapshot)=>{ const tempPosts
          = [];

          snapshot.docs.forEach(
              (doc)=>tempPosts.push({...doc.data(), id:
              doc.id}),);

          setPosts(tempPosts);});

  }, [courseId]);
  return (

export default Stream;

import React, {useState} from 'react';

import {Alert, Button, TextField, Typography} from '@mui/material';
import {Link} from 'react-router-dom';

import {LOGIN} from '../../constants/routes';

import {signInWithGoogle, signUpWithEmailAndPassword} from '../../firebase';
function SignupForm() {const [email, setEmail] = useState('');

  const [password, setPassword] = useState('');
  const [firstName, setFirstName] =
  useState(''); const [lastName, setLastName] =
  useState(''); const [error, setError] =
  useState(null);

  const handleSubmit = (e) => {
    e.preventDefault();

    signUpWithEmailAndPassword(email, password, firstName, lastName, setError);};

export default
SignupForm; import React
from 'react';

import {AppBar, Toolbar} from '@mui/material';

import Logo from '../atoms/Logo';

import Container from '../atoms/Container';


function
  PublicHeader() {
  return (

    <AppBar position="static">

      <Container>

        <Toolbar disableGutters>

          <Logo /></Toolbar>

   </Container>

    </AppBar>);}
```

```
export default function ProtectedHeader() {

  const [anchorEl, setAnchorEl] = React.useState(null);

  const [mobileMoreAnchorEl, setMobileMoreAnchorEl] = React.useState(null);

  const isMenuOpen = Boolean(anchorEl);

  const isMobileMenuOpen = Boolean(mobileMoreAnchorEl);

  const handleProfileMenuOpen = (event) => {
    setAnchorEl(event.currentTarget);};

  const handleMobileMenuClose = () => {
    setMobileMoreAnchorEl(null);};

  const handleMenuClose = () => {
    setAnchorEl(null);
    handleMobileMenuClose();};

  const handleLogout = () =>
    { setAnchorEl(null);
    handleMobileMenuClose();
    logout();};


  const handleMobileMenuOpen = (event) =>
  {setMobileMoreAnchorEl(event.currentTarget);};

  const {user} = useUser();

  const menuId = 'primary-search-account-menu';
  const renderMenu = (

      <MenuItem onClick={handleMenuClose}>Profile</MenuItem>

      <MenuItem onClick={handleLogout}>Logout</MenuItem>

    </Menu>);

  const mobileMenuId = 'primary-search-account-menu-
  mobile'; const renderMobileMenu = (

      id={mobileMenuId}
      keepMounted
      transformOrigin={{

        vertical: 'top',
        horizontal:
        'right',}}

      <


/* eslint-disable react/prop-types,max-len */

import React, {useContext, useEffect, useState} from
'react'; import {

  Box
  ,
  But
  ton
  ,
```

```
  Dia
  log
  ,
  DialogActions,
  DialogContent,
  DialogContentT
  ext,
  DialogTitle,
  Grid,
  Paper,
} from '@mui/material';
import moment from
'moment';

import Comment from '../molecules/Comment';
import TextEditor from
'../molecules/TextEditor'; import {db} from
'../../firebase';

import {convertContentToHTML} from
'../../utils/convertContentToHTML'; import firebase from 'firebase';

import useUser from '../../hooks/useUser';

import {TeacherContext} from '../../context/TeacherContext';


const Post = ({post}) => {

  const [showComments, setShowComments] =
  useState(false); const [isOpenModal, setIsOpenModal] =
  useState(false); const [comments, setComments] =
  useState([]);

  const {user} = useUser();

  const isTeacher = useContext(TeacherContext);
  const addComment = async (editorState) => {

    await db.collection('comments').add({
      content:
      convertContentToHTML(editorState),
      postId: post.id,

      createdAt: firebase.firestore.FieldValue.serverTimestamp(),
      authorName: user.name,

      authorPhoto: user.photo,});};

  const deletePost = async () => {

    await
    db.collection('posts').doc(post.id).delete();
    const tempComments = await
    db.collection('comments')

        .where('postId', '==', post.id).get();
    tempComments.docs.forEach((doc)=>{

      doc.ref.delete();});};

  useEffect(()=>{
```

```
    db.collection('comments').where('postId', '==', post.id)

        .orderBy('createdAt',
          'desc').onSnapshot((snapshot)=>{ const
          tempComments = [];

          snapshot.docs.forEach(
              (doc)=>tempComments.push({...doc.data(), id:
              doc.id}),);

          setComments(tempComments);});

  }, [post.id]);
  return (

    <>

      <Paper sx={{padding: '16px', mb: '16px'}} key={post.id}>

        <Grid container wrap="nowrap" spacing={2}>

          <Grid justifyContent="left" item xs zeroMinWidth>

            <div dangerouslySetInnerHTML={{__html: post.content}}></div>

            <Box sx={

              {display: 'flex',

                justifyContent: 'space-
                between', alignItems:
                'center',}}>

              <p style={{textAlign: 'left', color: 'gray'}}>

                {moment(post.createdAt?.toDate() || 0)

                    .format('MMMM Do YYYY, h:mm a')}

              </p>

              <div>

                <Button
                  variant="t
                  ext"

                  onClick={()=>setShowComments(!showComments)}

                >

                  {showComments ?

                      'Hide comments' : 'Show comments' +


export default Post;

import React, {useState} from 'react';

import {Alert, Button, TextField, Typography} from '@mui/material';
import {Link} from 'react-router-dom';

import {SIGNUP} from '../../constants/routes';

import {auth, signInWithGoogle} from '../../firebase';
```

```
import {useSignInWithEmailAndPassword} from 'react-firebase-
hooks/auth'; import Loader from '../molecules/Loader';


function LoginForm() {

  const [email, setEmail] = useState('');
  const [error, setError] =
  useState(null);

  const [password, setPassword] = useState('');

  const [login,, loading, signInError] = useSignInWithEmailAndPassword(auth);
  const handleSubmit = (e) => {

    e.preventDefault();
    setError(null);
    login(email,
    password);

  };

  if (loading) {
    return <Loader
    />;

  }

// eslint-disable-next-line react/prop-types

const JoinCourseModal = ({isOpen, handleClose, handleSubmit, title, label})
  => { const [value, setValue] = useState('');

  return (

    <Dialog open={isOpen} onClose={handleClose}>

      <DialogTitle>{title}</DialogTitle>

      <DialogContent>

        <DialogContentText>

                    To join a course, please enter course ID

        </DialogContentText>

        <TextField
          autoFocus
          margin="dens
          e" id="id"
          label={label
          }
          type="text"
          fullWidth

          variant="standard"
          value={value}

          onChange={(e)=>setValue(e.target.value)}

        />

      </DialogContent>

      <DialogActions>
```

```
        <Button onClick={handleClose}>Cancel</Button>
        <Button onClick={()=>handleSubmit(value)}>Submit</Button>
      </DialogActions>
    </Dialog>);};

export default JoinCourseModal;

import React, {useState} from
'react'; import {
  Button,
  Dialog,
  DialogActi
  ons,
  DialogCont
  ent,
  DialogContentTe
  xt,
  DialogTitle,
  TextField,
} from '@mui/material';

// eslint-disable-next-line react/prop-types
const DialogModal = ({isOpen, handleClose, handleSubmit, title, label}) => {
  const [value, setValue] = useState('');
  const [description, setDescription] =
  useState(''); return (
    <Dialog open={isOpen} onClose={handleClose}>
      <DialogTitle>{title}</DialogTitle>
      <DialogContent>
        <DialogContentText>
            To create a new course, please enter name and description.
        </DialogContentText>
        <TextField
          autoFocus
          margin="dens
          e" id="name"
          label={label
          }
          type="text"
          fullWidth
          variant="standard"
          value={value}
          onChange={(e)=>setValue(e.target.value)}/>
        <TextField
          variant="standard"
```

```
          label="Description"
          value={description}

          onChange={(e)=>setDescription(e.target.value)}
          multiline

          fullWid
          th
          rows={4
          }

          sx={{marginTop: '16px'}}/>

      </DialogContent>

      <DialogActions>

        <Button onClick={handleClose}>Cancel</Button>

        <Button onClick={()=>handleSubmit(value, description)}>Submit</Button>

      </DialogActions>

    </Dialog>);};


export default DialogModal;

import React, {useState} from 'react';
import {Editor} from 'react-draft-
wysiwyg'; import {Box, Button} from
'@mui/material';

import SendIcon from '@mui/icons-
material/Send'; import {EditorState} from
'draft-js';

import './TextEditor.css';

const TextEditor = ({onSend}) => {

  const [editorState, setEditorState] = useState(EditorState.createEmpty());
  const handleSend = async () => {

    try {

      await onSend(editorState);

    } catch (e) {

      alert(e.message);

    } finally {
      setEditorState(EditorState.createEmpty())
      ;}};

  return (<>

      <Editor
        editorState={editorState}
        wrapperClassName='demo-
        wrapper'
        editorClassName="demo-
        editor"
```

```
        onEditorStateChange={setEditorStat
        e} placeholder="Create new
        announcement"/>

      <Box sx={{display: 'flex', justifyContent: 'end', p: 2}}>

        <Button variant="contained" endIcon={<SendIcon />}
                    onClick={handleSend}> Send

        </Button>

      </Box></>);};

export default
TextEditor; import React
from 'react';

import {Box, CircularProgress} from '@mui/material';

const Loader = () => {
  return (

    <Box sx={{display: 'flex', width: '100%', justifyContent: 'center'}}>

      <CircularProgress />

    </Box>);};

export default Loader;

/* eslint-disable max-len */

import React, {useState} from
'react'; import {

  Button,
  Dialog,
  DialogActi
  ons,
  DialogCont
  ent,

  DialogContentTe
  xt,
  DialogTitle,
  TextField,

} from '@mui/material';

import {DatePicker, LocalizationProvider} from '@mui/x-date-
pickers'; import {AdapterDateFns} from '@mui/x-date-
pickers/AdapterDateFns'; import {Editor} from 'react-draft-wysiwyg';

import {EditorState} from 'draft-
js'; import
'./CreateTaskModal.css'; import
{db} from '../../firebase';

import {convertContentToHTML} from
'../../utils/convertContentToHTML'; import {useParams} from 'react-
router-dom';

const CreateTaskModal = ({isOpen, handleClose}) => {
  const [name, setName] = useState('');
```

```
  const [maxRating, setMaxRating] = useState(30);

  const [description, setDescription] = useState(EditorState.createEmpty());
  const [deadline, setDeadline] = useState(null);

  const {courseId} =
  useParams(); const createTask
  = async () => {

    try {await db.collection('tasks').add({ name,

        maxRating: parseInt(maxRating, 10),
        description:
        convertContentToHTML(description),
        deadline,

        courseId,});

      setName('');
      setDeadline(null);
      setMaxRating(30);

      setDescription(EditorState.createEmpty())
      ; handleClose();

    } catch (err) {

      alert(err.message);}};

  return (

 <Dialog open={isOpen} onClose={handleClose}>

   <DialogTitle>Create new task</DialogTitle>

      <DialogContent>

        <DialogContentText>

            To create a new task, please enter name, content, deadline and max

rating here.

        </DialogContentText>

        <TextField
          autoFocus
          margin="dens
          e" id="name"
          label="Name"
          type="text"
          fullWidth

          variant="standard"
          value={name}

          onChange={(e)=>setName(e.target.value)}/>

        <Editor
          editorState={description}
          wrapperClassName='demo-
          wrapper'

          editorClassName="create-task-
          editor"
          onEditorStateChange={setDescript
```

```
          ion}             placeholder="Add
          description"/>

        <TextField
          autoFocus
          margin="dens
          e"
          id="score"

          label="Max
          rating"
          type="number"
          fullWidth
          variant="standa
          rd"
          value={maxRatin
          g}

          onChange={(e)=>setMaxRating(e.target.value)}/>

        <LocalizationProvider dateAdapter={AdapterDateFns}>

          <DatePicker
            disablePast
            label="Deadli
            ne"
            value={deadli
            ne}

            onChange={(newValue) => {setDeadline(newValue);}}

            renderInput={(params) => <TextField margin='dense' {...params} />}/>

        </LocalizationProvider>

      </DialogContent>

      <DialogActions>

        <Button onClick={handleClose}>Cancel</Button>

        <Button onClick={createTask}>Submit</Button>

      </DialogActions>

    </Dialog>);};

export default CreateTaskModal;

import React from
'react'; import {

  Card, CardActionArea, CardContent, CardMedia, Typography,

} from '@mui/material';

import {string} from 'prop-types';

import {useNavigate} from 'react-router-dom';


function CourseCard({title, description, id,
  author}) { const navigate = useNavigate();
```

```
  const goToClass = () => {
    navigate(`/course/${id}`);

  };

  return (

    <Card sx={{maxWidth: 345}}>

      <CardActionArea onClick={goToClass}>

        <CardMedia
          component="img"
          height="140"

          image="https://happymonday.ua/wp-
content/uploads/talks/2018/10/courses-in-
min.png"

          alt="course photo"/>

        <CardContent>

          <Typography
            gutterBottom
            variant="h5"
            component="di
            v"
            title={title}
            sx={{

              overflow: 'hidden',
              textOverflow: 'ellipsis',
              display: '-webkit-box',
              WebkitLineClamp: '1',
              WebkitBoxOrient:
              'vertical',

            }}>

            {title}

          </Typography>

          <Typography
            variant="body2"
            color="text.secondar
            y"

            title={descriptio
            n} sx={{

              overflow: 'hidden',
              textOverflow: 'ellipsis',
              display: '-webkit-box',
              WebkitLineClamp: '2',
              WebkitBoxOrient:
              'vertical',

            }}>

            {description}

          </Typography>
```

```
          {author &&
              <Typography sx={{mt: 1.5, fontSize: 12}} color="text.secondary">
                {author}
              </Typography>}
        </CardContent>
      </CardActionArea>
    </Card>);}

CourseCard.propTypes = {
  title:
  string.isRequired,
  description: string.isRequired,};

export default CourseCard;

/* eslint-disable react/prop-types,max-len */
import React, {useEffect, useState} from
'react'; import Typography from
'@mui/material/Typography'; import Table from
'@mui/material/Table';

import TableHead from
'@mui/material/TableHead'; import TableRow from
'@mui/material/TableRow'; import TableCell from
'@mui/material/TableCell'; import TableBody
from '@mui/material/TableBody'; import {db}
from '../../firebase';

import {Button, Dialog, DialogActions, DialogContent, DialogTitle,
TextField} from '@mui/material';

import moment from 'moment';

const CompletedTask = ({id, student, date, rating, content, maxRating,
number, deadline}) => {

  const [isModalOpen, setIsModalOpen] = useState(false);
  const handleRating = (e) => {

    const newValue = e.target.value;

    if (newValue >= 0 && newValue <= maxRating) {
      setNewRating(newValue);}};

  const [newRating, setNewRating] = useState(rating || 0);
  const evaluate = () => {

    try {

      db.collection('completedTasks').doc(id).update
        ({ rating: newRating,});

      setIsModalOpen(false);} catch (e) {

      alert(e.message);}};

  return <>
```

```
    <TableRow sx={{backgroundColor: deadline < date ? 'rgb(253, 237, 237)' :
'inherit'}}>

      <TableCell>

       {number}

     </TableCell>

     <TableCell component="th" scope="row">

       {student}

      </TableCell>

      <TableCell>{moment(date?.toDate() || 0).format('L')}</TableCell>

      <TableCell align="right">

        {rating === undefined ? '-' : rating}

      </TableCell>

      <TableCell align="right"><Button variant="text"
onClick={()=>setIsModalOpen(true)}>{rating === undefined ? 'Evaluate' : 'Change
rating'}</Button></TableCell>

    </TableRow>

    <Dialog
      open={isModalOpe
      n}

      onClose={()=>setIsModalOpen(fals
      e)} aria-labelledby="alert-
      dialog-title"

      aria-describedby="alert-dialog-description">

      <DialogTitle id="alert-dialog-
                title"> Evaluate
                student&amp;apos;s work

      </DialogTitle>

      <DialogContent>

        <div dangerouslySetInnerHTML={{__html: content}}></div>

        <TextField
          autoFocus
          margin="dens
          e"
          id="score"
          label="Ratin
          g"
          type="number
          " fullWidth

          variant="standard"
          value={newRating}
          onChange={handleRating}/
          >

      </DialogContent>
```

```
      <DialogActions>
        <Button onClick={()=>setIsModalOpen(false)}>Cancel</Button>
        <Button onClick={evaluate}>Submit</Button>
      </DialogActions>
    </Dialog>
  </>;};

const CompletedTasks = ({taskId, maxRating, deadline}) => {
  const [completedTasks, setCompletedTasks] = useState([]);

  useEffect(()=>{
    db.collection('completedTasks').where('taskId', '==',

taskId).onSnapshot((snapshot)
      =>{ const tempTasks =
      [];

      snapshot.docs.forEach((doc)=>tempTasks.push({...doc.data(), id: doc.id}));
      cosole.log(tempTasks);}
```