Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

УДК 004.942

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: "Розробка макету програмного забезпечення організації роботи студентів на кафедрі за умов дистанційного навчання. Клієнтське програмне забезпечення"

Спеціальність – 121 "Інженерія програмного забезпечення"

ПОЯСНЮВАЛЬНА ЗАПИСКА

БР.ІПЗ - ___.___.___._____ ІПЗ

Студент
ІПЗ-44_____ /Олег МАРЧЕНКО/

Науковий керівник
ас. _____/Єлизавета ЖАБСЬКА/

Консультант
з питань нормоконтролю

фахівець_____/Тамара ЧАПОВСЬКА /

Допускається до захисту
Завідувач кафедри
к. ф.-м. н., доц _____/Олексій БИЧКОВ/

Київ - 2021

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Освітньо-кваліфікаційний рівень - бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖЕНО

Зав. кафедри програмних систем і технологій

_____ (Олексій БИЧКОВ)

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

Марченку Олегу Сергійовичу

(прізвище, ім'я, по батькові)

1. **Тема роботи:** «Розробка макету програмного забезпечення організації роботи студентів на кафедрі за умов дистанційного навчання. Клієнтське програмне забезпечення.»

Керівник проекту: керівник роботи асистент Жабська Єлизавета Олегівна затверджені на засіданні кафедри програмних систем і технологій, протокол №6 від «11» листопада 2020р.

2. **Строк подання студентом роботи** _____2021р.

3. **Вихідні дані до роботи:** монографії, підручники, навчальні посібники, статті та тези конференцій вітчизняних і зарубіжних авторів, Інтернет–ресурси з питань дистанційного навчання та систем керування навчанням.

4. **Зміст пояснювальної записки:**

1. Огляд концепції дистанційного навчання та характеристика систем керування навчанням
2. Розробка архітектури системи, опис ролей користувачів та моделей даних
3. Розробка програмної реалізації системи

5. **Перелік графічного матеріалу (із зазначенням обов'язкових креслень)**:

1. Приклад вигляду системи WebCT (Рис. 1.1, ст.17)
2. Приклад вигляду системи Moodle (Рис. 1.2, ст.18)
3. Діаграма компонентів системи (Рис. 2.1, ст.25)
4. Приклад вигляду компонента на JSX (Рис. 2.2, ст.26)
5. Структура проекту розробленого веб-додатку (Рис. 3.1, ст.28)

6. **Консультанти розділів проекту:**

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1 і 2 розділи аналітична частина	Єлизавета ЖАБСЬКА	Єлизавета ЖАБСЬКА	
3 і 4 розділи практична частина	Єлизавета ЖАБСЬКА	Єлизавета ЖАБСЬКА	

7. **Дата видачі завдання** ________ 2021 р.

Керівник _____ (Єлизавета ЖАБСЬКА)

Завдання прийняв до виконання _____ (Олег МАРЧЕНКО)

Календарний план

№ з/п	Назва етапів виконання етапів бакалаврської роботи	Термін виконання етапів роботи	Відмітка про виконання
1	Концепція системи дистанційного навчання	29.11.2020-05.12.2020	Виконано
2	Підбір та вивчення літератури	06.12.2020-04.01.2021	Виконано
3	Аналіз існуючих рішень з організації дистанційного навчання	09.01.2021-16.01.2021	Виконано
4	Огляд фреймворків для програмної реалізації клієнтської частини	28.01.2021-14.02.2021	Виконано
5	Розробка архітектури клієнтського веб-застосунку	15.02.2021-20.03.2021	Виконано
6	Розроблення серверного веб-застосунку у вигляді веб-API	21.03.2021-30.04.2021	Виконано

7	Тестування розробленого програмного забезпечення	02.05.2021-15.05.2021	Виконано
8	Оформлення і друк пояснювальної записки	16.05.2021-26.05.2021	Виконано
9	Оформлення презентації	27.05.2021-03.06.2021	Виконано
10	Отримання рецензії		
11	Затвердження пояснювальної записки роботи завідувачем кафедри		
12	Захист дипломної роботи		

Студент-бакалавр Олег МАРЧЕНКО

Керівник роботи Єлизавета ЖАБСЬКА

АНОТАЦІЯ

Випускна кваліфікаційна бакалаврська робота: 59 с., 33 рис., 2 табл., 9 джерел.

Тема: Розробка макету програмного забезпечення організації роботи студентів на кафедрі за умов дистанційного навчання. Клієнтське програмне забезпечення

Об’єкт дослідження: процес організації дистанційного навчання студентів за допомогою програмного забезпечення.

Мета робота: організація роботи студентів за умов дистанційного навчання.

Предмет дослідження: організація дистанційного навчання студентів за допомогою програмного забезпечення з клієнт-серверною архітектурою.

Результати дослідження: досліджено існуюче програмне забезпечення для організації дистанційного навчання. Створено клієнт-серверну систему організації роботи студентів за умов дистанційного навчання, з можливістю формування графіку навчання та створення й перевірки завдань для виконання.

Висновок

В результаті досліджень було розроблено клієнтський застосунок для організації роботи студентів, з можливістю формування навчального графіку, створення й перевірки завдань для виконання студентами.

СИСТЕМА УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ, ВЕБ-ДОДАТОК, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, СИСТЕМА ДИСТАНЦІЙНОГО НАВЧАННЯ

АННОТАЦИЯ

Выпускная квалификационная бакалаврская работа: 59 с., 33 рис., 2 табл., 9 источников.

Тема: Разработка макета программного обеспечения организации работы студентов на кафедре в условиях дистанционного обучения. Клиентское программное обеспечение

Объект исследования: процесс организации дистанционного обучения студентов с помощью программного обеспечения.

Цель работы: организация работы студентов в условиях дистанционного обучения.

Предмет исследования: организация дистанционного обучения студентов с помощью программного обеспечения с клиент-серверной архитектурой.

Результаты исследования: исследовано существующее программное обеспечение для организации дистанционного обучения. Создана система организации работы студентов в условиях дистанционного обучения, с возможностью формирования графика обучения, создания и проверки заданий для выполнения.

Вывод

В результате исследований было разработано клиентское приложение для организации работы студентов, с возможностью формирования графика обучения, создания и проверки заданий для выполнения студентами.

СИСТЕМА УПРАВЛЕНИЯ ОБУЧЕНИЕМ, ВЕБ-ПРИЛОЖЕНИЕ, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, СИСТЕМА ДИСТАНЦИОННОГО ОБУЧЕНИЕ

ABSTRACT

Graduation qualifying bachelor's thesis: 59 p., 33 figs., 2 table, 9 sources.

Topic: Development of a software model for organizing the work of students at the academic department under conditions of distance learning. Client software

Object of research: the process of organizing distance learning of students using software.

Purpose: organization the work of students under conditions of distance learning.

Subject of study: organization of distance learning of students using software with client-server architecture.

Research results: the existing software for the organization of distance learning is investigated. A system of the organization of work of students under the conditions of distance learning has been created, with the abilities to form the training schedule and create/check course tasks.

Conclusion

As a result of the research, a server-side web client application was developed to organize the work of students, with the abilities to form the training schedule and create/check course tasks.

LEARNING PROCESS MANAGEMENT SYSTEM, WEB-APPLICATION, CLIENT-SERVER ARCHITECTURE, DISTANCE LEARNING SYSTEM

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

LMS – система управління навчанням (англ. Learning Management System)

JXS – синтаксис мови програмування JavaScript (англ. JavaScript Programming Language Syntax)

HTTP – протокол передачі гіпертексту (англ. HyperText Transfer Protocol)

JSON – запис об’єктів JSON (англ. Javascript Object Notation)

HTML – мова розмітки веб-сторінок (англ. Web Page Markup Language)

ВСТУП

Сучасний світ знаходиться в зеніті створення нових і кращих ідей вирішення проблем, які раніше здавались для усього світу лише недосяжною точкою в розвитку та укріпленню людства, як розвинутого суспільства. Наразі, людство має доволі розвинуті технології та легше переступає вік цифрових та інформаційних технологій, що дає нам можливість відмовлятись від старих і не таких продуктивних рішень проблем, які на той час були вкрай необхідними для нас. Проте, часи змінюються йдучи рука об руку з технологічним розвитком нашої цивілізації. На це нас підштовхують непередбачувані обставини, як невидима рука, і дає можливість адаптуватися до них, що робить нас більш розвинутими ніж до цього. Адаптування до нових обставин у світі тісно зв'язане з суспільством, що дає усім нам дізнатись щось нове та вдосконалити свої пізнання. Цей феномен суспільства можна описати, як кругообіг знань в нашому світі. У зв'язку із появою різних родом пандемій людство зрозуміло необхідність систем для дистанційного зв'язку та взаємодії. Традиційне навчання міцно закріпилось у свідомості людей, як самий ефективний спосіб навчання. Його головна суть це відвідування оффлайн лекцій та практик, та отримувати знання та навички напряму від відвідувань навчального закладу. Цей спосіб став неможливим за сучасних обставин. Виникла необхідність отримувати знання дистанційним способом, який би не був гіршим за традиційне навчання, а й кращим за нього. Ще до цього почали з'являтися системи дистанційного навчання, але їм не давали такого сущого значення, як традиційному. Тому на даний час створення і підтримка такої системи є сучасним і кращим рішенням для навчання.

Актуальність теми: наявність системи дистанційного навчання вирішує низьку проблем та полегшує проходження навчання на кафедрі. Вона має всю необхідну інформацію в одному місці для всіх учасників навчального процесу та дозволяє керувати проходженням навчання

Метою роботи є розробка макету клієнтської частини до системи, яка надає можливості проведення навчального процесу на кафедрі «ПСТ» за дистанційних умов.

Завдання:

1. Дослідити розробку системи для навчального процесу в дистанційному форматі;
2. Проаналізувати та продумати необхідну функціональність системи, яка саме має бути впроваджена в ній;
3. Створити макет клієнтської частини до системи дистанційного навчального процесу на кафедрі
4. Провести тестування, для вже створеного макету, з допомогою спеціально створених тест-кейсів до цієї системи та описати результати отримані під час його проходження.

Об'єкт дослідження – процес організації дистанційного навчального процесу осіб на кафедрі за допомогою системи дистанційного навчання.

Предметом дослідження є організація дистанційного навчального процесу за допомогою системи для проведення дистанційного навчання.

Методи дослідження : мова програмування JavaScript, редактор коду Visual Studio Code, відкрита JavaScript-бібліотека для створення інтерфейсів користувача React.js, фреймворк Material UI, який базується на принципах Material Design.

Новизна одержаних результатів: було розроблено макет клієнтської частини до системи, яка надає можливості проведення навчального процесу на кафедрі «ПСТ» за дистанційних умов та простого запуску на сервері.

Практичне значення одержаних результатів: практична цінність розробленого макету полягає в можливості проведення навчального процесу в дистанційній формі навчання з допомогою розробленої системи, яка в свою чергу має відкритий код, що дає можливість вдосконалення системи в майбутньому, та не використовує допоміжні системи, які можливо використовувати тільки при оплаті їх послуг, що робить систему безкоштовною в технічному плані.

РОЗДІЛ 1
КОНЦЕПЦІЯ СИСТЕМИ ДИСТАНЦІЙНОГОНАВЧАННЯ

1.1 Поняття дистанційного навчання

Таке поняття, як дистанційне навчання, почали вживати та впроваджувати останні два десятиліття. До цього основним і головним методом навчання було традиційним, і всі вважали, що воно буде головним методом навчання. Деякі форми освітньої реформи сприяють прийняттю прогресивних навчальних практик, більш цілісного підходу, який зосереджується на потребах та самоконтролі окремих учнів. В очах реформаторів від традиційних методів, орієнтованих на вчителя, зосереджених на вивченні і запам'ятовуванні, слід відмовитись на користь орієнтованих на учнів та підходів до навчання, спрямованих на завдання [1].

Традиційне навчання було основним видом навчання в усьому світі. Тяжко традиційне навчання позначити, як зовсім відсталу і не ефективну форму навчання. Проте, в сучасних обставинах цей вид навчання не дозволяє студентам отримувати ті знання та навички, які необхідні для отримання обраної студентами спеціальності. Та незважаючи на мінуси звичного всім, традиційне навчання має і гарні плюси. Насамперед, при традиційному навчанні студенти мають можливість отримати гарні соціальні навички, так як вони завжди мають спілкуватися та контактувати, як один з одним, так з викладачами. Вміння гарно адаптуватися до соціуму дає студентам перші та базові навички взаємодії з новими людьми, яке вважається одним із базових і необхідних вмінь людини для кращого життя. Також, при традиційному навчанню студенти мають можливість спілкуватися з викладачами віч навіч. Викладачі, це люди які мають не тільки теоретичні знання та практичне застосування тих знань, якими вони володіють, а й досвід в своїй науковій та професійній справі. Студенти отримують знання виносячи з досвіду викладача, даючи їм неоціненний досвід та ті знання, які неможливо отримати за допомогою підручника чи статей по цій темі.

Основні переваги та недоліки традиційного навчання зображені на таблиця 1 та таблиця 2.

Таблиця 1.1

Переваги дистанційного навчання

№	Переваги	Доповнення
1	Очне навчання	Очне навчання дозволяє студентам та вчителям набагато краще пізнати одне одного. Вчителі можуть визначити фізичні черги та нюанси, які зазвичай можуть залишатися непоміченими в Інтернеті. Найголовніше, що взаємодія з учителем віч-на-віч дозволяє студентам побудувати довіру до вихователя.
2	Навички складати розпорядок	Для багатьох студентів на початку навчання дуже важко організовувати свій денний розклад. Під час навчання студенти правильно навчаються складати свій розклад, що полегшує майбутнє життя людини в суспільстві.
3	Практичні приклади	Викладачі мають можливість надати студентам приклади з реального життя, з яких вони можуть зробити висновки на майбутнє. Це є ключовим елементом для практичних навичок студентів.
4	Навчає гарним соціальним навичкам	Ідея потрапити до групи з людьми різного походження та типу особистості дає безліч соціальних можливостей, які студент не обов'язково буде відчувати в онлайн-форматі.

Таблиця 1.2

Недоліки дистанційного навчання

№	Недоліки	Доповнення
1	Негнучкий графік навчання	Викладачам та студентам може бути важко дотримуватись жорсткого графіка навчального закладу, якщо вони мають кілька робочих місць чи інші зобов’язання.
2	Поїздки на навчання	Щоденні поїздки до навчального закладу може

		зайняти багато часу як для викладачів, так і для студентів. Поїздка до навчального закладу та назад може з’їсти багато часу та грошей, витрачених на бензин, обслуговування автомобілів або громадський транспорт.
3	Пасивне слухання	Під час лекцій викладач дає студентам багато інформації, яку йому необхідно донести за навчальним планом, хоча вона не зовсім є цікавою та необхідною для студентів. Тому студенти зловживають цим, і не завжди відвідують лекції.

Дистанційне навчання активно почало розвиватися останні два десятиліття і було спеціально створено для підтримки та проведення навчального процесу в навчальному закладі, при відсутності можливості фізичної присутності, яке є однією з головних вимог проведення традиційного навчання. Популярністю переходу з традиційного навчання до дистанційного стали сучасні обставини появи різного роду пандемій. Для проведення дистанційного навчання необхідно аби учасник навчання мав доступ до інтернету та комп’ютера, в деяких випадках необхідно мати такі прилади, як веб-камера, навушники та мікрофон. Як показує правило, студенти та викладачі взаємодіють між собою через онлайн-чати, форуми та електронну пошту. Для прямого зв'язку також використовуються онлайн-конференції, де студенти та викладачі мають можливість спілкуватися в прямому ефірі.

Перевагами дистанційного навчання є:

- Гнучкість - як правило, дистанційне навчання базується на асинхронній роботі студентів та викладачів, що дає можливість виконувати роботу в будь якому місці та в зручний час.
- Легкий доступ до матеріалів - завдяки системам дистанційного навчання в усіх учасників навчального процесу є доступ до всіх необхідних їм інформаційних даних одразу, що забезпечує більш зручне отримання інформації.

– Взаємодія - сучасні платформи дають можливості дистанційного зв'язку викладачів та студентів у вигляді онлайн-чатів, онлайн-конференцій та електронної пошти, що робить зв'язок учасників навчального процесу простішим ніж при традиційному навчанні.

1.2 Поняття системи дистанційного навчання

За станом на сьогодні, завдяки популярності впровадження і використання систем дистанційного навчання, створюються нові терміни та методи їх використання. Основними є створення віртуального класу, що створює модель очного навчання за допомогою Інтернет-технологій. Це дозволяє студентам та викладачам взаємодіяти між собою та використовує вже всім звичного та зручного виду очного навчання. Цей метод має в собі таку собі мережеву систему.

Наразі системи дистанційного навчання виконують всі необхідні функції для полегшення проходження навчального процесу для всіх діючих осіб в навчальному закладі.

Системи дистанційного навчання, як зазвичай все інше програмне забезпечення, поділяється на дві категорії: платні та з відкритим кодом. Звісно платні системи мають гарний і готовий функціонал, що й вабить користувачів, проте більшість навчальних закладів обирають саме системи з відкритим кодом, оскільки вони є безкоштовними. Проте, системи з відкритим кодом необхідно вдосконалювати власноруч, що змушує навчальні заклади шукати спеціалістів для налаштування, аби система працювала під їх потреби.

Однією з перших таких систем стала WebCT (Рис. 1.1). Була розроблена в Університеті Британської Колумбії в 1995 році та заснована на дослідженнях щодо покращення академічних показників за допомогою інформаційних технологій та веб-ресурсів [4].

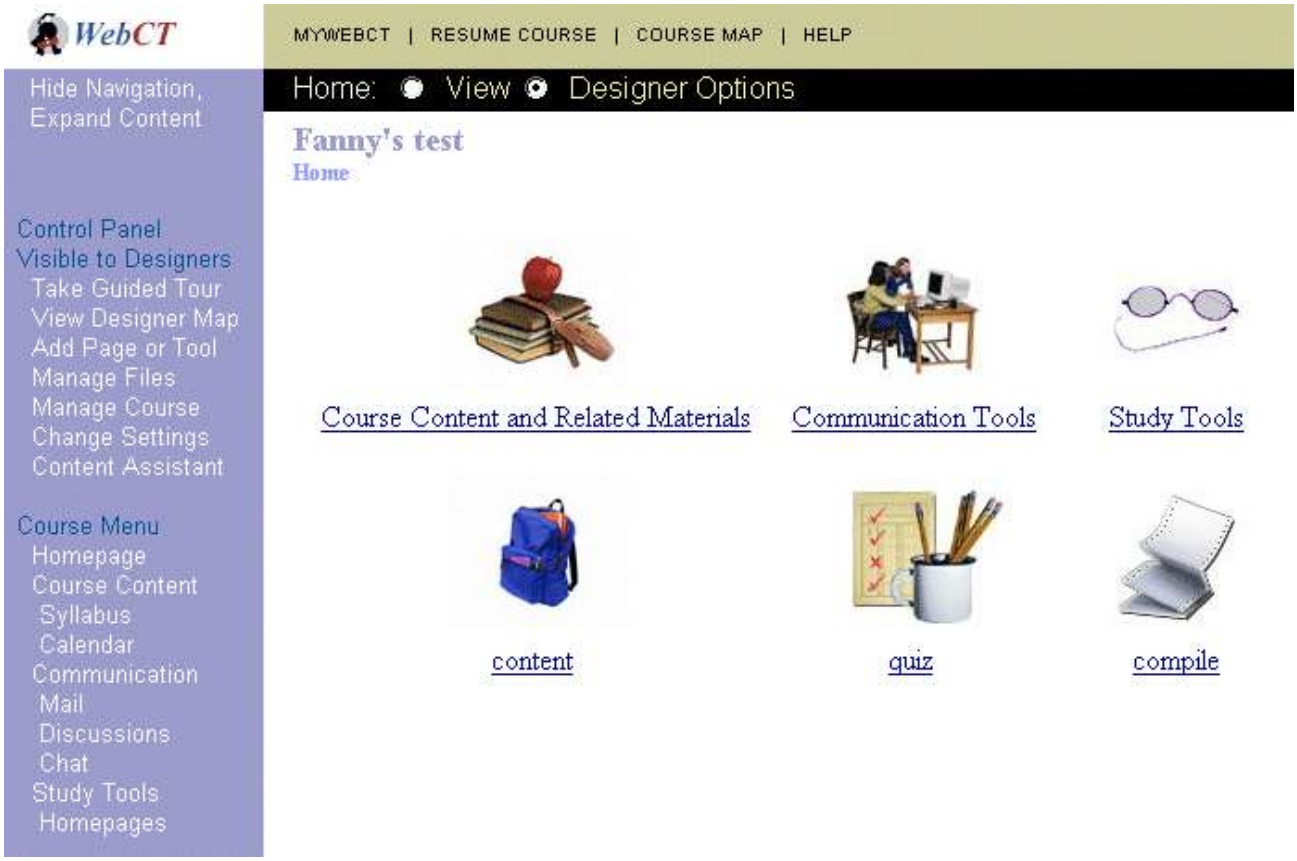

Рис. 1.1 Приклад вигляду системи WebCT

На противагу платним LMS-системам почало з’являтись програмне забезпечення з відкритим вхідним кодом, що робить такі системи більш доступними для використання, оскільки вони були безкоштовними. Самою популярною такою системою стала Moodle (Рис. 1.2), тому що має всю необхідну функціональність для забезпечення навчального процесу за умов дистанційного навчання. Сама суть цієї системи дозволяє ділити ролі в ній на студентів та викладачів, які можуть взаємодіяти між собою. Викладачі можуть створювати курси, розміщати лекційний та навчальний матеріал та створювати завдання і надавати їм кінцевої дати здачі цих завдань студентами. Студенти в свою чергу мають можливість переглянути свої курси, мати доступ лекційних та навчальних матеріалів та відправляти виконанні завдання викладачеві. Насьогодні є тисячі сайтів працюючих на системі Moodle, і на даний момент є лідируючою серед інших систем [5].

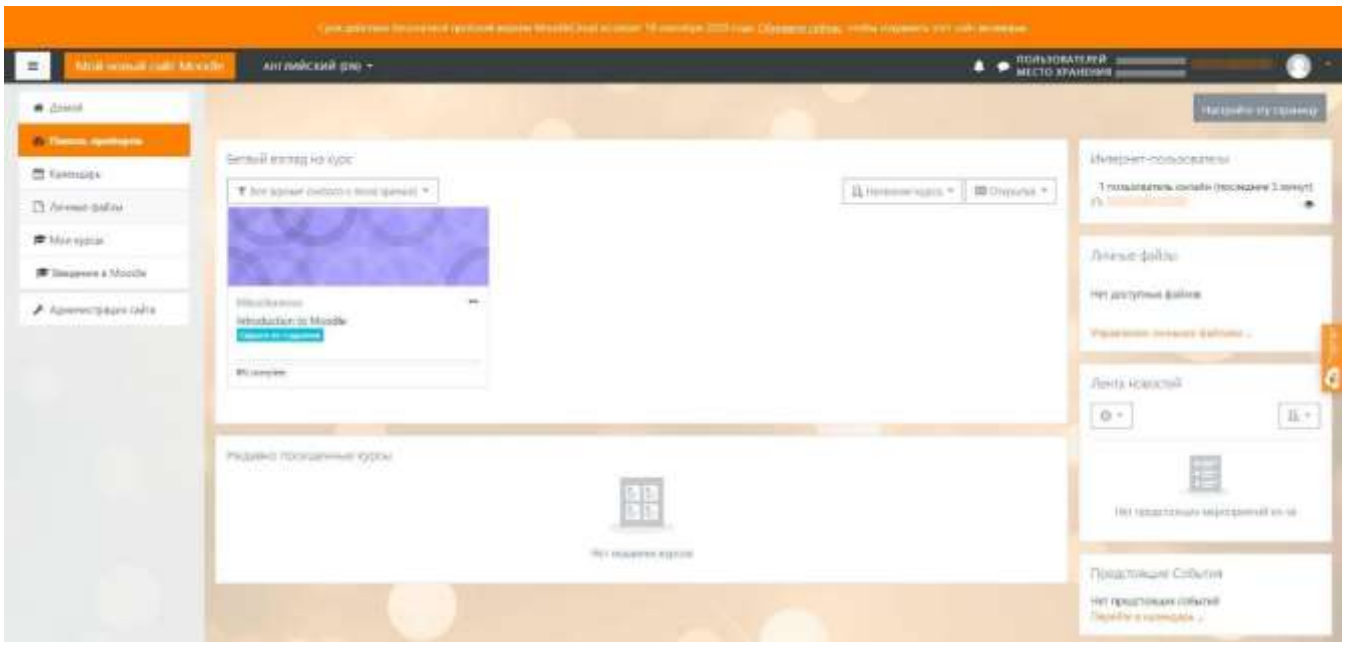

Рис. 1.2 Приклад вигляду системи Moodle

1.3 Напрямки розвитку систем дистанційного навчання

Системи дистанційного навчання почали впроваджувати вже десятки років тому, проте актуальності вони почали набувати лише недавно. Є велика необхідність їх вдосконалення завдяки новим технологіям, і як найшвидшого впровадження. Необхідність їхнього вдосконалення буде зростати доти, поки не з’явиться необхідність у новій формі навчання.

З появою систем дистанційного навчання зростає можливість їх керуванням на різних приладах та застосунках. Тобто система має бути кросплатформеною, аби нею можливо було користуватись, не прив’язуючись до одного приладу. Також в системах LMS виникає необхідність в покращенні зв’язку між учасниками навчального процесу. До цього можна віднести онлайн-чати та онлайн-конференції. Оскільки, якість зв’язку залежить не тільки від гарного підключення до Інтернет-ресурсів, а і якісно розроблена система.

До основних напрямків розвитку систем дистанційного навчання можна віднести:

1. Адаптивне навчання

Адаптивне навчання впроваджене в систему дистанційного навчання дозволило б краще аналізувати результати студентів, на основі їх успіхів, та в свою чергу більш детально підлаштовувало навчальний матеріал для кращого засвоєння і розвитку студентів. Завдяки таким технологіям студенти краще ознайомлюються з навчальним матеріалом, оскільки, система постійно вивчає поведінку та результати кожного студента та надає вказівки, що допомагають їм досягти навчальних цілей курсу.

2. Аналітика

Звітування на основі отриманих результатів, а в випадку навчання це є результати оцінок, на сьогодні є одною із найважливіших функціональностей систем LMS. Чим більш розвинуті аналітичні засоби в системі, тим більш затребуваною та кращою вона буде для користування. Наприклад такими засобами аналітики можуть бути: контроль присутності, аналіз успішності та якість виконання завдань.

3. Засоби комунікації

Можливість комунікації в нас час є одною із головних вимог в розробці будь якого програмного забезпечення. Система LMS надає можливість дистанційного зв’язку з іншими користувачами, тому необхідно аби було впроваджено різні засоби комунікації. Це може бути як і розроблене всередині програмного забезпечення онлайн-чати, так і інтегровані до системи різні засоби комунікації, такі як: Telegram, WhatsApp, Viber та інші. Також необхідним компонентом до системи являється прив’язка електронної пошти до користувача. Це полегшує комунікацію всередині системи, оскільки вона найлегшим, і в той час одною і найважливішим, видом зв’язку в наш час. Для онлайн конференцій можна як і створити свою незалежну систему від інших платформ для онлайн зв’язку, так і інтегрувати вже готову

систему. Як показує практика що інтегрування таких систем як: Google Meet, Zoom, GoToMeeting та інші є гарним варіантом для системи LMS.

4. Хмарні технології

Хмарні технології вже давно користуються популярністю в суспільстві, оскільки вони довели свою корисність в плані легшого доступу до своїх даних та легкість в користуванні. Деякі системи LMS вже мають впроваджені хмарні технології та довели їх корисність та популярність. Раніше це було розкішним додатковим технологічним модулем, проте на сьогодні це є обов'язковим пунктом в розробці систем дистанційного навчання. Наразі складно розробити хмарну технологію з нуля, тому кращим рішенням цієї проблеми буде інтегрування вже існуючих рішень. Це є гарним рішенням, тому що це заощаджує час розробки системи та раніше створені рішення мають всю необхідну функціональність.

5. Кросплатформеність

З технологічним розвитком з'являється нова портативна техніка (зокрема, ноутбуки, смартфони, планшети) і разом з цим виникає необхідність наявності в сучасному програмному забезпечені кросплатформеності. Для прикладу, насьогодні всі користуються смартфонами і є зручнішим мати можливість переглядати навчальний розклад кафедри за допомогою смартфону. Майже всі системи дистанційного навчання мають клієнт-серверну структуру, що надає можливість створювати клієнтську частину незалежно від серверної. Також клієнтський веб-додаток можна розробляти не створюючи мобільний додаток, тому що при розробці з адаптивною верстkою веб-додаток може коректно працювати і на периферійних пристроях (смартфон, планшет, т.щ.).

1.4 Огляд типової функціональності системи дистанційного навчання

При створені системи дистанційного навчання потрібно розуміти, що вона повинна бути надійною у використані, працювати без перебоїв, мати гарну та якісну комунікацію серед користувачів системи та мати доступ до інформації, яку надає сама система. На сьогодні більшість функціональностей до систем LMS не є системними забаганками, а вимогами до неї. Це відбувається завдяки швидкому переході більшості навчальних закладів до дистанційного навчання.

Одними із головних функціональностей системи LMS є:

1. Оцінювання результатів.

Для правильної роботи системи LMS повинна бути розроблена функціональність оцінювання результатів. Це є одною із головних категорій функціональностей, тому що на сьогодні необхідно аби система дозволяла викладати різного виду роботи для оцінювання знань студентів (наприклад, тести, практичних завдань тощо) та мати можливість переглядати результати виконаних робіт. Це спрощує сам підхід до дистанційного навчання, оскільки викладачу не потрібно збирати всіх в онлайн-конференції, та заощаджує час і викладачів та студентів. Також це забезпечує кращого розуміння студента на його успішність.

2. Відслідковування прогресу.

Як показує практика студенти не завжди розуміють в якому стані наразі знаходиться його успішність. Аби студентам було легше це розуміти необхідно впроваджувати таку функціональність, як відслідковування прогресу. Вона вирішує дві великі проблеми в плані навчання. Перша це студент завжди знатиме на якому рівні знаходиться його успішність, що заощаджує час студентів та викладачів, аби перші в свою чергу не турбували других. Друга це полегшення керування звітністю по успішності студентів. Викладачу ця функціональність заощаджує багато часу.

3. Управління дисциплінами.

Категорія функціональності управління дисциплінами є одною із фундаментальних в навчанні, які повинні бути впровадженні в системи LMS. Керування розкладом, можливість зміни або доповнення дисциплін, викладення навчального матеріалу та корисної інформації є самими головними функціями, з якими система повинна працювати.

4. Безпека та конфіденційність.

Насправді, кожне програмне забезпечення повинно бути надійно захищеним від зловмисників та тримати конфіденційність своїх користувачів. Така функціональність, як безпека та конфіденційність обов'язково повна бути впроваджена в систему LMS. На сьогодні такі системи мають можливість автентифікації та авторизації, при якому відправляються запити з даними користувача до бази даних системи. Дані кожного користувача повинні бути конфіденційними, тому система має захищати їх дані.

5. Комунікація викладачів та студентів.

В наш час вже не здивуєш технологіями з різними засобами зв'язку. Сучасні системи LMS повинні надавати засоби комунікації між викладачами та студентами. Ці засоби повинні бути синхронними та асинхронними. Асинхронними засоби відрізняються від синхронних тим, що перші в свою чергу мають односторонній зв'язок, а другі мають з'єднання в режимі реального часу. До асинхронних засобів відносяться електрона пошта, сповіщення тощо. До синхронних засобів відносяться онлайн-конференції, голосові чати, тощо.

6. Доступність користувачам

З розвитком технологій портативної техніки (ноутбуки, смартфони, планшети) з'являється потребність в можливості користування програмного забезпечення на різних пристроях. Це обумовлено тим, що на сьогодні люди не завжди мають час та знаходяться за своїм робочим комп'ютером, тому вони користуються портативною технікою. В дистанційному навчанні також виникає така потребність, наприклад,

переглянути розклад та завдання на наступний день. Розробка такої функціональності не є досить важкою. Такі системи LMS розроблюються за технологією клієнт-сервер, тому серверна частина працює незалежно від усієї системи, а клієнтська частина розробляється з допомогою адаптивної верстки, що дозволяє системі коректно відображатись на різних пристроях.

1.5 Висновки до розділу

В розділі було ознайомлено з концепцією проведення навчального процесу за дистанційних умов, систем дистанційного навчання та їх базову функціональність. Було обрано технологію роботи веб-додатку та ознайомлено з тими вимогами, які система повинна вирішувати.

РОЗДІЛ 2
АРХІТЕКТУРА ВЕБ-ДОДАТКУ ДЛЯ ПРОВЕДЕННЯ НАВЧАЛЬНОГО ПРОЦЕСУ НА КАФЕДРІ ЗА УМОВ ДИСТАНЦІЙНОГО НАВЧАННЯ

2.1 Цілі системи

Система для організації роботи студентів за умов дистанційного навчання повинна:

1. Створювати та керувати користувачами в системі надаючи їм відповідні ролі в системі.
2. Мати можливість авторизації та автентифікації користувачів до системи.
3. Мати можливість створювати та керувати навчальним графіком на кафедрі, створювати навчальні групи та створювати для них завдання для перевірки успішності.
4. Мати можливість змінювати паролі до акаунтів користувачів системи, з цілю уникнути ситуацій з неможливим доступом до акаунту користувача.
5. Мати можливість надати викладачам слідкувати за успішністю студентів, мати доступ до перегляду навчального графіку та надання їм прав для зміни кінцевої дати здачі завдань студентам
6. Мати можливість студентам мати доступ до перегляду навчального графіку, отримання нових завдань для виконання та відправка завдань для перевірки успішності.

2.2 Архітектура системи

Клієнт програма матиме в собі логіку взаємодії самого веб-додатку та серверу. Завдяки можливостям фреймворку React бізнес-логіка веб-додатку виконуватиметься саме в ньому. Це спрощує саму розробку системи та робить її гнучкість завдяки тому, що запити та маніпулювання даними виконується вже в веб-додатку.

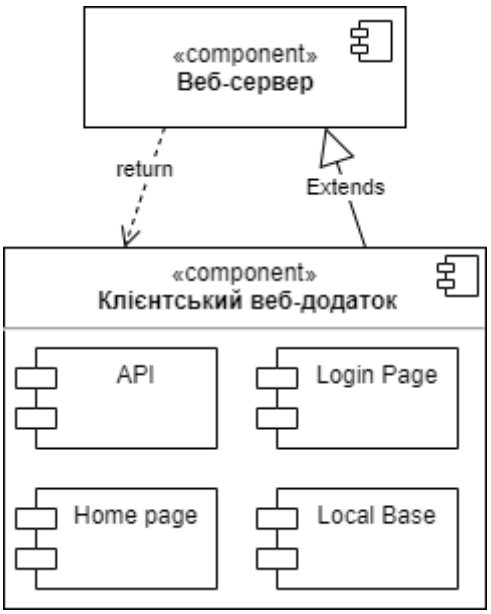

Рис. 2.1 Діаграма компонентів системи

2.3 Ролі в системі

В системі є 3 ролі:

1. Студент

   Має можливість переглядати навчальний графік, отримувати завдання від викладачів, які необхідно виконати, відправляти завдання на перевірку викладачеві, переглядати кінцеві дати для відправки завдань на перевірку викладачеві.

2. Викладач

   Має можливості переглядати навчальний графік, створювати завдання для перевірки успішності навчання студентів, створювати та керувати кінцевими датами для відправки завдань для перевірки успішності студентів

3. Адміністратор

   Має можливість створювати та керувати користувачами системи, створювати та керувати навчальним графіком на кафедрі.

2.4 Огляд технологічних рішень, що були використані для розробки клієнтського веб-додатку

Для розробки клієнтського веб-додатку системи дистанційного навчання було обрано клієнт-серверну технологію. Мову програмування для розробки було обрано JavaScript, тому що для створення веб-додатків. Також використовувався синтаксис JSX (Рис. 2.2), який має всі можливості JavaScript з використанням тегів, як в HTML. Це набагато спрощує розробку веб-додатку, тому що одразу до компоненту є можливість описати необхідну для нього функціональність. Для полегшення розробки системи було обрано використання таких фреймворків, як: Node.js, React.js. Також використовувалась бібліотека Material UI, яка використовувалась для якісного та зрозумілого дизайну веб-додатку.

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Привіт, {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Петро" />,
  document.getElementById('hello-example'),
);
```

Рис. 2.2 Приклад вигляду компонента на JSX

Фреймворк Node.js являється серверною платформою для виконання JavaScript коду через вже створений компанією Google рушій V8. Node.js дозволяє запускати локальний сервер для веб-додатку написаний на JavaScript. Сереверне середовище створене на Node.js є асинхронним, що робить менше навантаження на веб-додаток. Node.js є найкращим варіантом для розробки веб-додатку.

Фреймворк React.js - це інтерфейсна бібліотека JavaScript із відкритим кодом для побудови користувальницьких інтерфейсів або компонентів інтерфейсу. Він підтримується Facebook та спільнотою окремих розробників та компаній. React можна використовувати як основу при розробці односторінкових або мобільних додатків. Однак React займається лише управлінням станом та наданням цього стану в DOM, тому для створення додатків React зазвичай потрібно використовувати додаткові бібліотеки для маршрутизації, а також певну функціональність на стороні клієнта.

Ще однією помітною особливістю є використання віртуальної об'єктної моделі документа або віртуальної DOM. React створює кеш-структуру даних в пам’яті, обчислює отримані відмінності, а потім ефективно оновлює відображуваний DOM браузера. Цей процес називається примиренням. Це дозволяє програмісту писати код так, ніби вся сторінка відображається при кожній зміні, тоді як бібліотеки React відображають лише підкомпоненти, які насправді змінюються. Цей вибірковий візуалізація забезпечує значне підвищення продуктивності. Це економить зусилля з перерахунку стилю CSS, макета сторінки та рендеринга для всієї сторінки.

2.5 Висновки до розділу

В розділі оглянуто ті технології та архітектурні рішення, які були використані впродовж розробки програмного забезпечення. Також описані ті мови програмування та фреймворки, які лягли в основу створення веб-додатку.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ КЛІЄНТСЬКОГО ВЕБ-ДОДАТКУ ДЛЯ ЗАБЕЗПЕЧЕННЯ НАВЧАЛЬНОГО ПРОЦЕСУ НА КАФЕДРІ ЗА УМОВ ДИСТАНЦІЙНОГО НАВЧАННЯ

3.1 Огляд архітектури клієнтського застосунку

Веб-додаток розроблений на React в якому описується відображення інтерфейсу для користувачів. React-додаток – містить файли які, відповідають за відображення компонентів, описують всі підключенні бібліотеки до проекту, додаткові стилі до компонентів та самі компоненти. На рисунку 3.1 представлено структуру розробленого веб-додатку.

Рис. 3.1 Структура проекту розробленого веб-додатку

Структура проекту веб додатку складається з таких модулів:

- Модуль node-modules має в собі всі засоби та бібліотеки встановлені для правильної роботи і покращення веб-додатку.
- Модуль public має в собі сторінку index.html, яка відповідає за відображення проекту в веб-браузері та файл manifest.json, який містить в собі початкові та основні дані проекту.
- Модуль src містить в середині себе всю логіку проекту. Файл index.js відповідає за візуалізацію проекту на сторінці додатку та викликається в файлі index.html. Файл App.js описує додаток і відповідає зп маршрутизацію в середині проекту. Файл axios.js створений для забезпечення авторизації та автентифікації користувачів до системи.
- Модуль components містить в собі всі модулі для відображення на головній сторінці додатку.

Одні із головних бібліотек, які були підключені до проекту та допомагають в коректній роботі є:

- Бібліотека material-ui допомагає брати готові стилі для проекту з фреймворку Material UI. Його принцип роботи складається з того, що при імпорті його до компоненту він надає вказаним тегам вже вбудованого стилю. Це досить гарно забезпечує час розробки і є простим у використанні.
- Бібліотека router-dom допомагає впровадити коректну та просту маршрутизацію по компонентах веб-додатку. Маршрутизація являється одною із головних функціональностей сучасних веб-додатків, тому ця бібліотека є одною із необхідних.
- Бібліотека axios впроваджена для можливості авторизації та автентифікації до системи та обробки HTTP-запитів до неї. Оскільки, клієнтська частина тісно працює в серверною, то за допомоги цієї

бібліотеки клієнтська частина спокійно може звертатись до серверної за даними і навпаки.

В модулі node-modules знаходяться всі підключені бібліотеки для коректної роботи проекту. Не всі вони підключені напряму, проте мають великий функціонал для майбутнього вдосконалення проекту. Більшість з них встановлюються при створенні самого проекту на React, оскільки для його створення використовуються бібліотеки Node.js, яка встановлюється до цього.

В модулі public містяться файл index.html (Рис. 2.4), який відповідає за відображення сторінки в веб-браузері та файл manifest.json (Рис. 2.5), який описує початкові дані проекту.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/Fit.png" />
    <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>LMS-кафедри</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Рис 3.2 Файл index.jsx

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```

Рис. 3.3 Файл manifest.jsx

Модуль src має в собі всю логіку проекту та модуль components. Файл index.js (Рис 2.6) відповідає за візуалізацію всього проекту на сторінці в веб-браузері. Файл App.js (Рис 2.7) відповідає за те, які компоненти мають відображатись та маршрутизацію с системі.

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

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Рис. 3.4 Файл index.js

```
import React from 'react';
import { BrowserRouter, Route } from 'react-router-dom'
import { LoginPage } from './components/LoginPage'
import { HomePage } from './components/HomePage'
import Logout from './components/Logout';

function App() {

  return (
    <div className="App">
      <BrowserRouter>
        <Route path="/" exact component={HomePage}/>
        <Route path="/login" exact component={LoginPage}/>
        <Route path="/logout" component={Logout}/>
      </BrowserRouter>
    </div>
  );
}

export default App
```

Рис. 3.5 Файл App.js

Файл axios.js відповідає за правильну авторизацію та автентифікацію користувача (Рис. 2.7) та його вихід з системи. Для цього він посилається до серверної частини https://lms-api-dev.herokuapp.com/api та при посиланні запиту до

неї повертає токен користувача системи. Також виводить в командній стрічці помилки, які виникають під час роботи системи (Рис. 2.8).

```
import axios from 'axios';

const baseURL = 'https://lms-api-dev.herokuapp.com/api';

const axiosInstance = axios.create({
    baseURL: baseURL,
    timeout: 5000,
    headers: {
        Authorization: localStorage.getItem('access_token')
            ? 'JWT ' + localStorage.getItem('access_token')
            : null,
        'Content-Type': 'application/json',
        accept: 'application/json',
    },
});
```

Рис. 3.6 Отримання токену при авторизації користувача до системи

```
async function (error) {
    const originalRequest = error.config;

    if (typeof error.response === 'undefined') {
        alert(
            'A server/network error occurred. ' +
                'Looks like CORS might be the problem. ' +
                'Sorry about this - we will get it fixed shortly.'
        );
        return Promise.reject(error);
    }

    if (
        error.response.status === 401 &&
        originalRequest.url === baseURL + 'auth/token/refresh'
    ) {
        window.location.href = '/login';
        return Promise.reject(error);
    }
```

Рис. 3.7 Перевірка на помилки під час роботи системи

Модуль components має в собі всі компоненти системи які викликаються на сторінку в веб-браузері. Головними діючими компонентами є сторінка входу до системи, домашня сторінка користувача та сторінка виходу користувача з системи. При початку роботи системи головною сторінкою є домашня сторінка користувача на якій відображуються компоненти головної сторінки та перевіряє чи був користувач авторизований до системи (Рис. 2.9), і якщо користувач вже був раніше авторизований і сесія користування системи не закінчилась, то буде одразу відображена ця сторінка. Проте, якщо користувач не авторизувався до системи його буде перенаправлено на сторінку входу до системи. Вона має в собі також компоненти для відображення на сторінці веб-браузера (Рис. 2.10) та відправку даних для запиту на серверну частину для входу (Рис. 2.11). Сторінка виходу з системи відкривається при завершені сесії користувачем власноруч. Вона виконує очищення всіх даних отриманих для входу користувача до системи, що змушує її перенаправити користувача до сторінки входу, тому що він не має тепер доступу до системи (Рис. 2.12).

```
export const HomePage = () => {
    const classes = useStyles();

    if (localStorage.getItem('access_token') == null) {
        return <Redirect to="/login"/>
    } return (
        <div className={classes.root}>
            <Header />
            <Grid item xs={10} container spacing={3}>
                <Grid item xs={3}>
                    <MainList />
                </Grid>
                <Grid item xs={7} className={classes.paper}>
                    <UserList />
                </Grid>
            </Grid>
        </div>
    )
}
```

Рис. 3.8 Компоненти, які відображаються на головній сторінці

```
return (
  <Grid container component="main" className={classes.root}>
    <CssBaseline />
    <Grid item xs={false} sm={4} md={7} className={classes.image} />
    <Grid item xs={12} sm={8} md={5} component={Paper} elevation={6} square>
      <div className={classes.paper}>
        <Avatar className={classes.avatar}></Avatar>
        <Typography component="h1" variant="h5">
          Система електронного навчання на кафедрі "ПТС"
        </Typography>
        <form className={classes.form} Validate>
          <TextField
            variant="outlined" ···
            onChange={handleChange}
          />
          <TextField
            variant="outlined" ···
            onChange={handleChange}
          />
          <FormControlLabel
            control={<Checkbox value="remember" color="primary" />}
            label="Запам'ятати мене"
          />
          <Button
            type="submit" ···
            onClick={handleSubmit}
          >
            Увійти
          </Button>
          <Grid container> ···
        </form>
      </div>
    </Grid>
  </Grid>
);
```

Рис. 3.9 Компоненти сторінки входу до системи

```
const history = useHistory();
const initialFormData = Object.freeze({
  email: '',
  password: '',
});

const [formData, updateFormData] = useState(initialFormData);

const handleChange = (e) => {
  updateFormData({
    ...formData,
    [e.target.name]: e.target.value.trim(),
  });
};

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

  axiosInstance
    .post(`/auth/login/`, {
      email: formData.email,
      password: formData.password,
    })
    .then((res) => {
      localStorage.setItem('access_token', res.data.access);
      localStorage.setItem('refresh_token', res.data.refresh);
      localStorage.setItem('full_name', res.data.user.full_name);
      localStorage.setItem('email', res.data.user.email);
      axiosInstance.defaults.headers['Authorization'] =
        'JWT ' + localStorage.getItem('access_token');
      history.push('/');
      console.log(res.data);
    })
};
```

Рис. 3.10 Відправка даних до запиту для авторизації користувача

```
useEffect(() => {
    const response = axiosInstance.post('/auth/logout/', {
        refresh_token: localStorage.getItem('refresh_token'),
    });
    localStorage.removeItem('access_token');
    localStorage.removeItem('refresh_token');
    localStorage.removeItem('full_name');
    localStorage.removeItem('email');
    axiosInstance.defaults.headers['Authorization'] = null;
    history.push('/login');
});
return(
    <div>Logout</div>
);
```

Рис. 3.11 Очищення всіх даних для виходу користувача з системи

Компоненти Header, MainList та Workspace відображаються на головній сторінці системи. Компонент Header (Рис. 2.13) має в собі посилання до головної сторінки системи (Рис. 2.14), кнопку для нагадувань (Рис. 2.15) та кнопку відкриття модального вікна користувача, де мається можливість виходу з системи (Рис. 2.16).

```
return(
    <div>
        <AppBar position="static">
            <Toolbar>
                <HomeLink />
                <div className={classes.grow} />
                <NotificationIcon />
                <AccIcon />
            </Toolbar>
        </AppBar>
    </div>
)
```

Рис. 3.12 Компонент Header

```
return (
    <div>
        <Button
        component={Link}
        color="inherit"
        to="/">
            <h3>Система дистанційного навчання на кафедрі "ПТС"</h3>
        </Button>
    </div>
)
```

Рис. 3.13 Компонент HomeLink

```
export const NotificationIcon = () => {
   return(
    <div>
        <IconButton color="inherit">
            <NotificationsIcon fontSize="large"/>
        </IconButton>
    </div>
   )
}
```

Рис. 3.14 Компонент NotificationIcon

```
return(
<div>
    {auth && (
        <div>
          <IconButton
            aria-label="account of current user"
            aria-controls="menu-appbar"
            aria-haspopup="true"
            onClick={handleMenu}
            color="inherit"
          >
            <AccountCircleIcon fontSize="large"/>
          </IconButton>
          <Menu
            id="menu-appbar"···
            onClose={handleClose}
          >
            <Grid container spacing={3}>
                <Grid item xs></Grid>
                <Grid item xs={14}>
                    <AccountCircleIcon className={classes.container} color="action" style={{ fontSize: 60 }}/>
                </Grid>
                <Grid item xs></Grid>
            </Grid>
            <Grid container>
                <Grid item xs className={classes.info}>
                    <Typography>{localStorage.getItem('full_name')}</Typography>
                </Grid>
            </Grid>
            <Grid container>
                <Grid item xs className={classes.info}>
                    <Typography>{localStorage.getItem('email')}</Typography>
                </Grid>
            </Grid>

            <Divider variant="middle"/>

            <MenuItem onClick={navigateTo}>
              <ListItemIcon>
                <ExitToAppIcon fontSize="medium" />
              </ListItemIcon>
              Вихід з аккаунту
            </MenuItem>
          </Menu>
        </div>
    )}
</div>
)
```

Рис. 3.15 Компонент AccIcon

Компонент List має в собі навігаційну панель користувача по системі. При натисненні на яку небудь кнопку будуть показуватись ті компоненти, які будуть доступні для використання користувачеві (Рис. 3.16).

```
return(
    <div>
        <Paper className={classes.mainList}>
            <ListItem button>
                <ListItemIcon>
                    <TodayIcon />
                </ListItemIcon>
                <ListItemText primary="Календарь" />
            </ListItem>
            <ListItem button>
                <ListItemIcon>
                    <CollectionsBookmarkIcon />
                </ListItemIcon>
                <ListItemText primary="Предмети" />
            </ListItem>
            <ListItem button>
                <ListItemIcon>
                    <AssessmentIcon />
                </ListItemIcon>
                <ListItemText primary="Оцінки" />
            </ListItem>
            <ListItem button>
                <ListItemIcon>
                    <AccountBoxIcon />
                </ListItemIcon>
                <ListItemText primary="Профіль" />
            </ListItem>
            <ListItem button>
                <ListItemIcon>
                    <EmailIcon />
                </ListItemIcon>
                <ListItemText primary="Повідомлення" />
            </ListItem>
            <Divider variant="middle"/>
        </Paper>
    </div>
)
```

Рис. 3.16 Компонент MainList

3.2 Зовнішній вигляд програмного забезпечення

Сторінка входу має наступний вигляд (Рис. 3.17). На сторінці знаходиться форма для заповнення даних необхідних для авторизації користувача. А саме, поле пошти користувача, поле паролю користувача та функціональна кнопка входу до системи. Кнопка виконує запит до серверної частини для автентифікації на основі тих даних, що ввів користувач в полях пошти та паролю. Сторінка є адаптивною, що дозволяє комфортну роботу через різні розширення екрану.

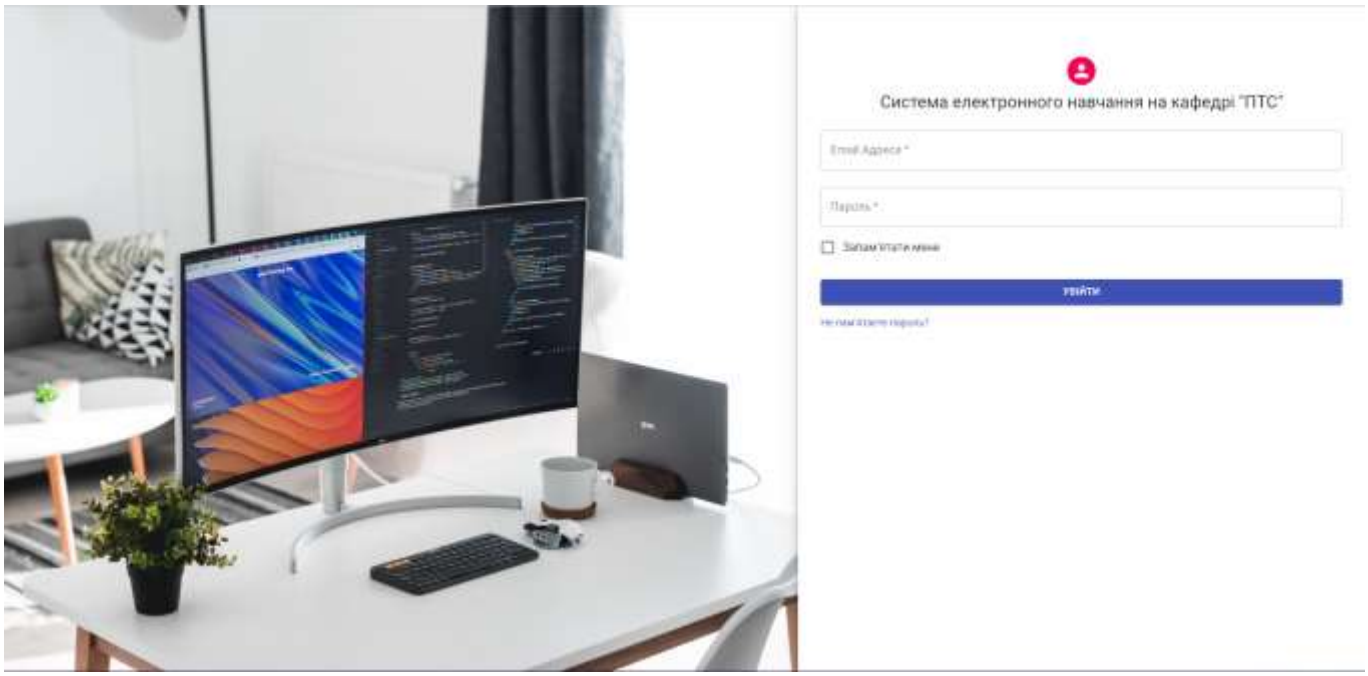

Рис. 3.17 Сторінка входу до системи

Головна сторінка системи має наступний вигляд (Рис. 3.18). В собі вона містить верхню панель системи, панель управління системи головний робочий компонент, який відображає компоненти обрані на панелі управління. В верхній панелі системи мається назва системи, яка при натисненні перенаправляє користувача до головної сторінки, кнопка сповіщення та кнопка користувача, яка при натисненні відкриває модальне вікно з інформацією про користувача та можливістю виходу з системи (Рис. 3.19). Панель управління системи знаходиться з лівого боку робочої зони на якій маються кнопки, які при натисненні показують на головну робочу зону відповідні компоненти (Рис. 3.20).

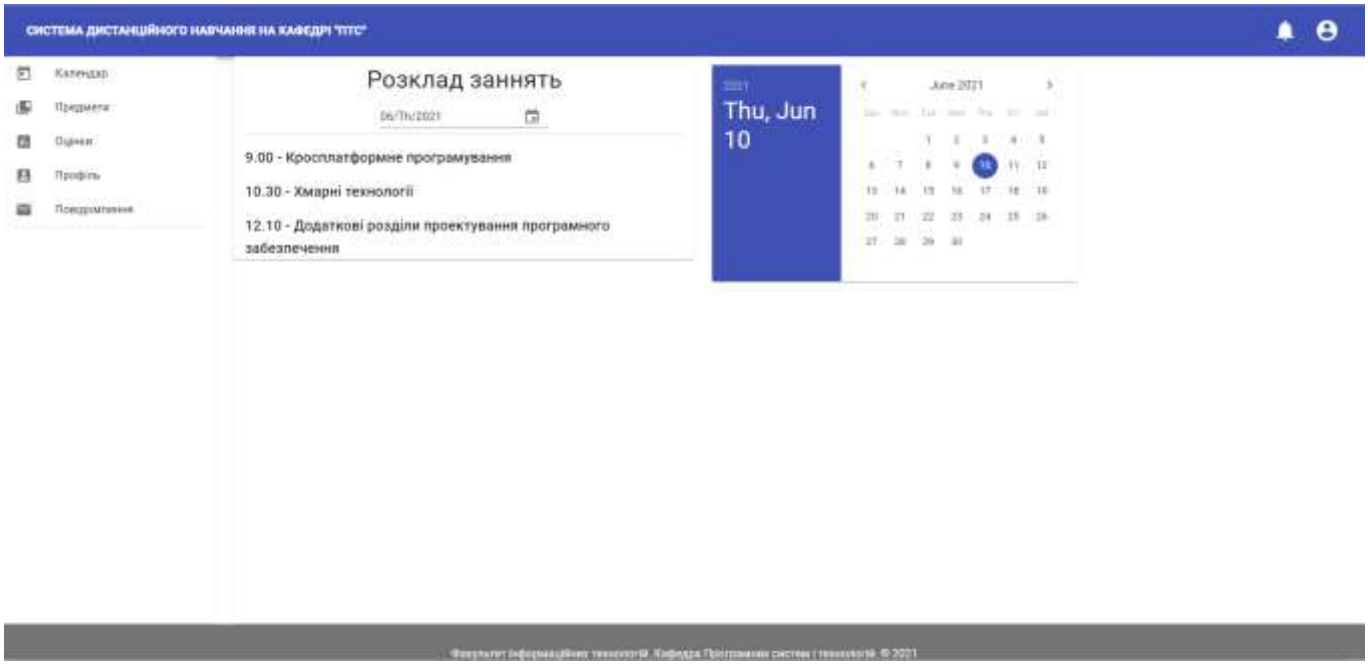

Рис. 3.18 Головна сторінка системи

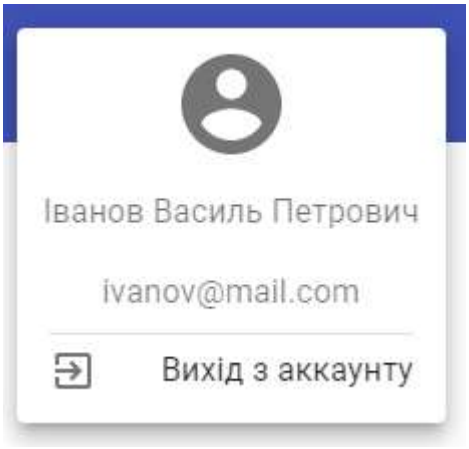

Рис. 3.19 Модальне вікно з інформацією користувача

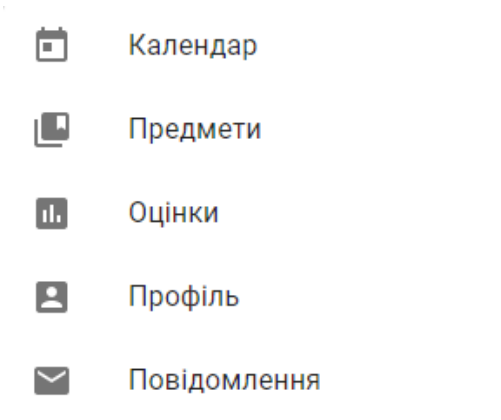

Рис.3.20 Панель управління системи

На головній робочій зоні розташовуються ті компоненти, які прив’язані до панелі управління системи. Для прикладу студент при натисненні на кнопку «Календар» на панелі управління, отримає компоненти розкладу занять та календар (Рис. 3.21). А при натисненні на кнопку «Профіль» на панелі управління, отримає компонент інформації про користувача в функціональними кнопками (Рис. 3.22).

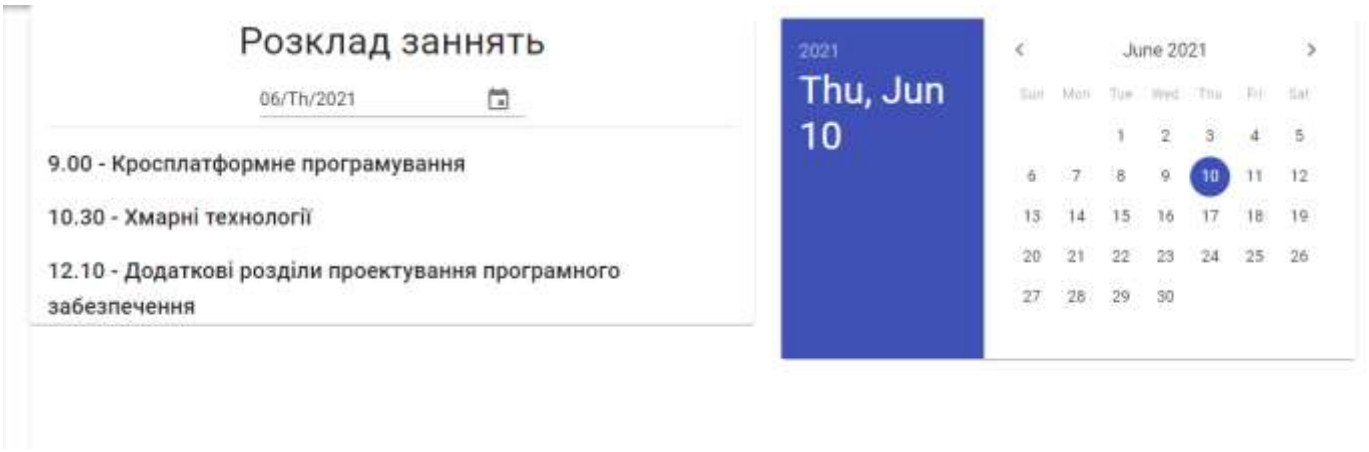

Рис. 3.21 Компоненти календарю в головній робочій зоні

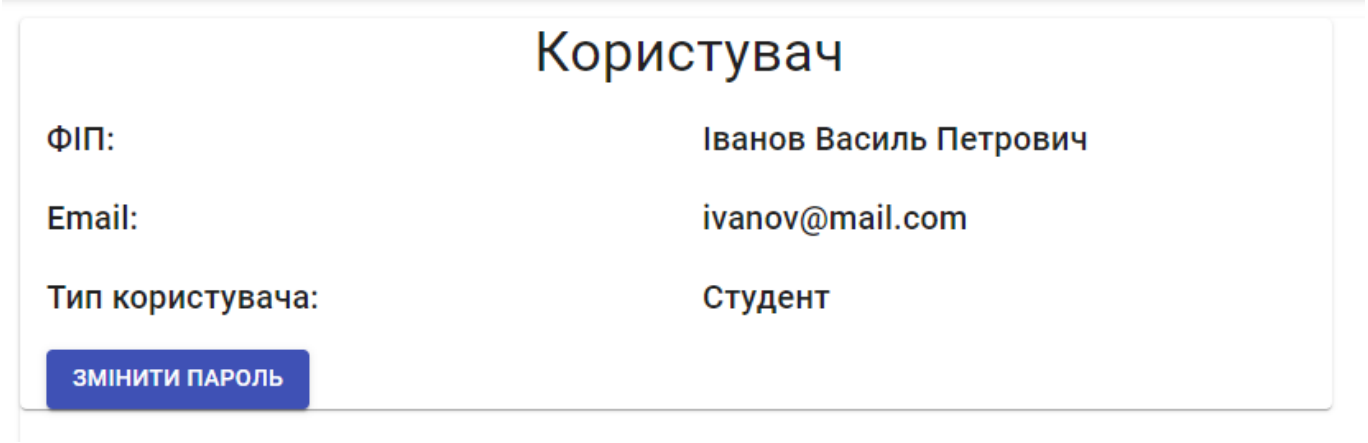

Рис. 3.22 Компонент інформації користувача в головній робочій зоні

3.3 Інструкція користувача

Основні характеристики застосунку:

Мова програмування: JavaScript

Середовище розробки: Visual SC

Мінімальні системні вимоги:

- 64-бітна операційна система Windows
- Node.js (завантажити на офіційному сайті розробників та встановити);
- Material UI (завантажити на офіційному сайті розробників);
- Material Icon (завантажити на офіційному сайті розробників);
- Axios (завантажити та встановити через командну строку);
- Router Dom (завантажити та встановити через командну строку);
- Moment (завантажити та встановити через командну строку);
- 256 МБ вільної пам’яті на комп’ютері.

Вхідні дані:

- HTTP-запити ;
- параметри: дані у JSON-форматі.

Вихідні дані:

- HTTP-відповіді з даними у JSON-форматі.

Запустити веб-додаток безпосередньо через локальний сервер можна на комп’ютері за портом 3000 (http://localhost:3000). Для запуску проекту на локальному сервері необхідно через командну строку перейти в поточний каталог проекту та прописати команду: npm start. Після цього проект відкриється самостійно в браузері по замовченню.

Для безпосереднього запуску системи через локальний сервер необхідно мати вже встановлену бібліотеку Node.js, яка дозволяє запускати проект на локальному сервері через командну строку. Node.js це платформа, яка дозволяє запускати проект на локальному сервері, має асинхронну одно-ниткову модель виконання запитів та має підтримку V8(рушій JavaScript), який компілює скрипти в веб-

додатку. Також необхідно мати встановлені всі використовувані бібліотеки, які забезпечують коректну роботу системи (наприклад, material-ui, axios, react-router-dom та інші).

Для повноцінної роботи необхідно мати дані користувача, для входу до системи. Це напряму залежить від серверної частини, тому необхідно мати при собі вхідні данні. Без вхідних даних користувач не зможе зайти до системи, і в такому випадку не матиме можливості користуватися системою.

Також для роботи проекту необхідно мати веб-браузер, який дозволяє напряму взаємодіяти з системою. Загалом, без різниці який браузер обирати для користування системою, проте для розробників рекомендується обирати веб-браузер Google Chrome. Його можна завантажити та встановити з офіційного сайту розробників. Рекомендується використовувати саме цей веб-браузер, тому що він має багато розширень для розробників на React, який допомагає в зручності розробки та має всі необхідні інструменти та засоби для розробки веб-додатків.

3.4 Тестування програмного забезпечення

Для тестування системи необхідно перевірити коректну роботу авторизацію та автентифікацію користувача до системи, маршрутизацію користувача по системі та коректну роботу всіх компонентів системи.

Для того аби перевірити коректну роботу авторизації та автентифікації користувача до системи необхідно ввести на сторінці входу один раз коректні дані та некоректні. При коректних ведених даних користувачем система отримує токен користувача (Рис. 3.23), та перенаправляє його до домашньої сторінки(Рис. 3.24). При некоректних ведених даних користувачем система показує помилку про те, що дані не є коректними (Рис. 3.25) та залишає користувача на сторінці входу до системи (Рис. 3.26).

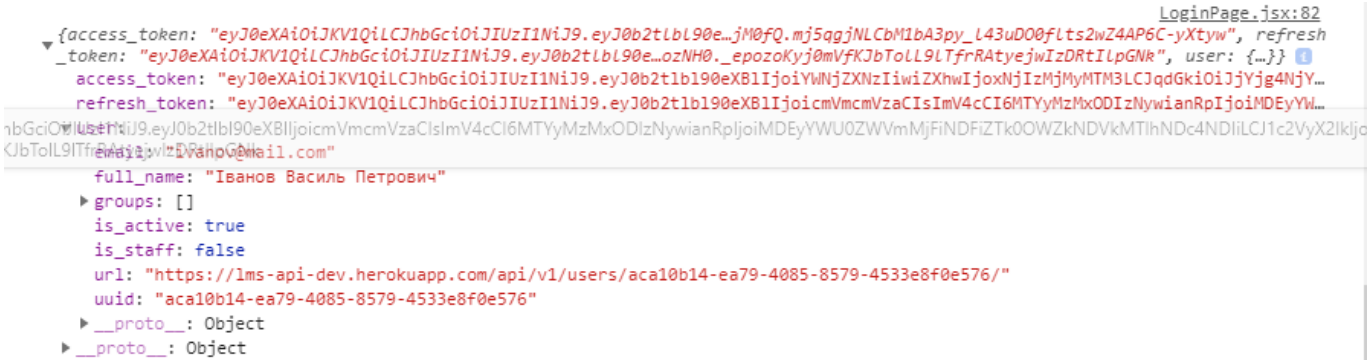

Рис. 3.23 Отриманий токен з даними користувача

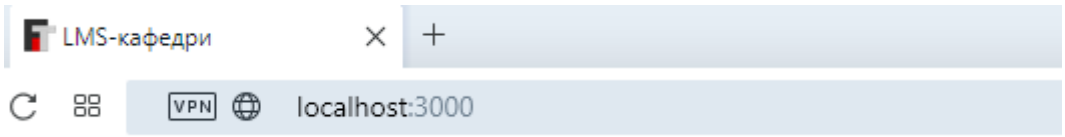

Рис. 3.24 Домашня сторінка системи

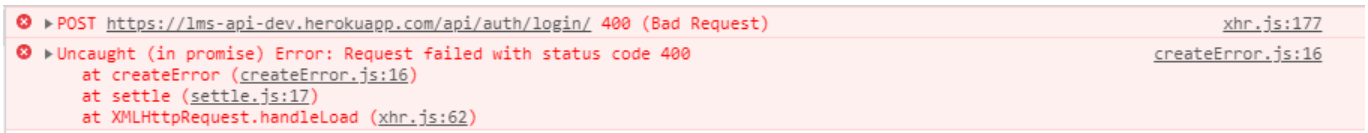

Рис. 3.25 Помилка 400 про невірні дані при вході до системи

Рис. 3.26 Сторінка входу до системи

Для перевірки маршрутизації необхідно перевірити перехід до сторінки входу до системи, якщо користувач не робив авторизацію і система не має отриманого токену з серверного додатку (Рис. 3.27). І навпаки перевірити перехід користувача до головної сторінки, якщо система має токен користувача, що означає успішну авторизацію, та сесія його роботи в системі не завершилась (Рис 3.28).

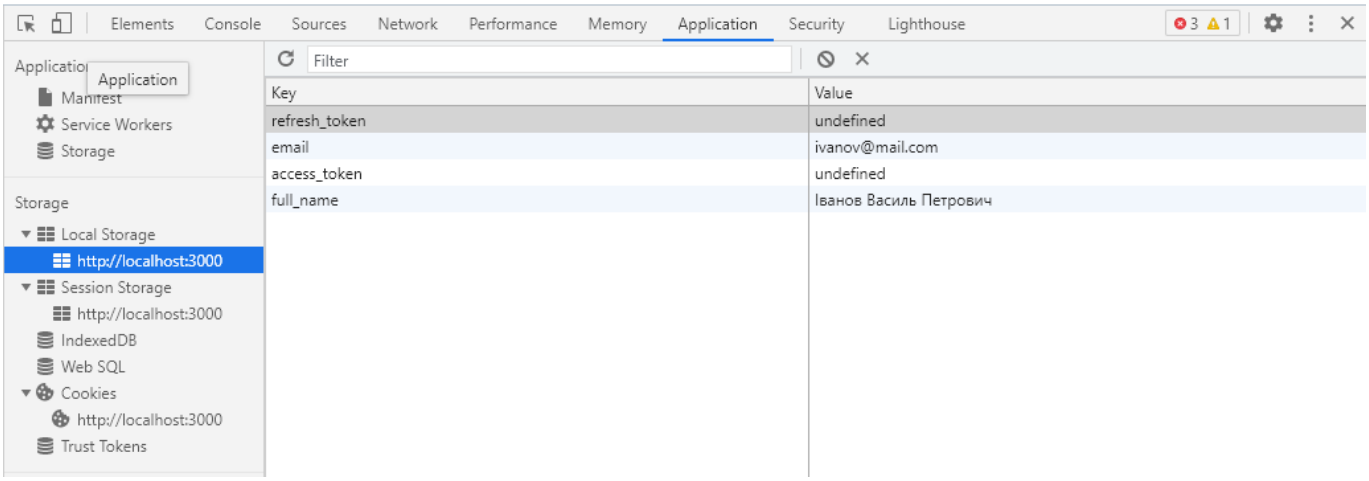

Рис. 3.27 Отриманий токен користувача та його дані при переході на домашню сторінку системи

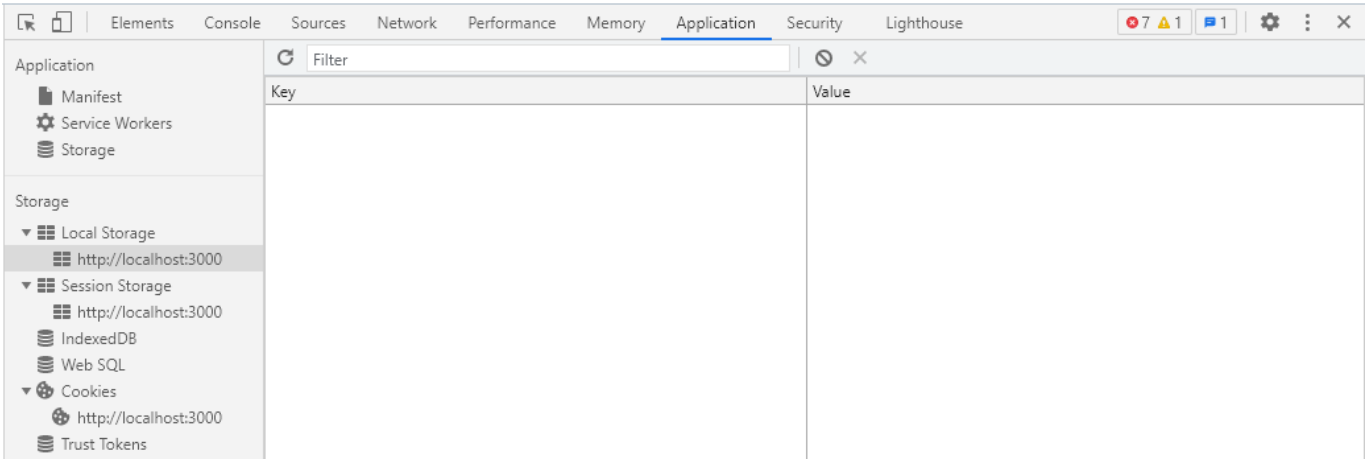

Рис. 3.28 Токен відсутній при переході на домашню сторінку системи і перенаправлення до сторінки входу

Для перевірки коректності роботи всіх компонентів було відображено всі компоненти правильно і вони працюють згідно того як вони мають працювати. До прикладу відображення даних користувача та кнопка виходу з системи на модальному вікні користувача при натисненні на кнопку користувача (Рис. 3.29).

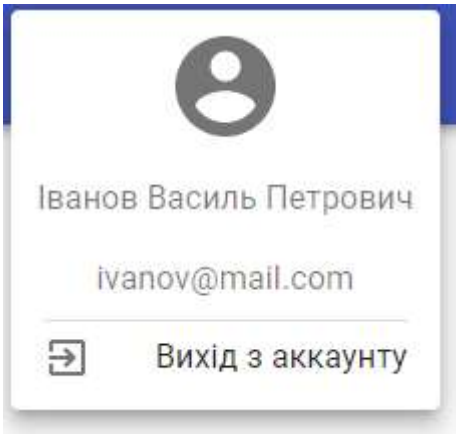

Рис. 3.29 Модальне вікно користувача при натисненні на кнопку користувача

3.5 Висновки до розділу

В третьому розділі було описано реалізацію веб-додатку у вигляді рисунків реалізованих на фреймворку React та представлено зовнішній вигляд основних сторінок та компонентів веб-додатку в веб-браузері Google Chrome

ВИСНОВКИ

В ході виконання дипломної роботи було освоєно та вивчено такі теми як: концепція проведення дистанційного навчання та систем керування навчального процесу, розробка веб-додатків за допомогою фреймворків React та Material UI, застосування HTTP-запитів в веб-додатках. Для реалізації клієнтського веб-додатку для організації роботи студентів боло використано мову програмування JavaScript, фреймворки React та Material UI, бібліотеки Node.js, Axios, Router DOM.

Як результат, було розроблено макет клієнтського веб-додатку, що дозволяє вирішити такі проблеми:

1. Відображення та керування графіком навчального процесу на кафедрі для всіх користувачів системи.
2. Можливість безпечної авторизації та автентифікації користувачів системи з захистом особистих даних.
3. Мобільна та зрозуміла маршрутизація по системі дистанційного навчання, яка не вимагає шукати різну функціональність в системі за допомогою документації.
4. Вся інформація для комфортного проходження навчального процесу знаходиться в одній системі, що надає можливість користувачам мати все необхідне на одній платформі.

Розроблений веб-додаток має сучасний та зрозумілий інтерфейс та надає можливість користувачам в різними правами доступу виконувати та користуватись тою функціональністю, яку їм надано. Є можливості перегляду та керуванням розкладу навчального процесу, створення завдань для студентів та їх перегляд студентами, налаштування кінцевої дати здачі завдань викладачам, відправка завдань на перевірку викладачеві.

В майбутньому розглядається можливість вдосконалення системи, а саме впровадити таку функціональність:

- Інтегрувати додаткові засоби комунікації, такі як онлайн-чати та онлайн конференції для кращого зв’язку між студентами та викладачами.
- Створення документації всередині системи, на базі отриманих результатів проходження навчання студентами.
- Створення автоматизованих засобів для перевірки виконаних завдань студентами, для буде заощаджувати час і студентів і викладачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Traditional education – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: en.wikipedia.org/wiki/Traditional_education

2. Watson WR., Watson SL., An Argument for Clarity: What are Learning Management Systems, What are They Not, and What Should They Become? // TechTrends, 2007. – 34 p.

3. Learning management system – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: en.wikipedia.org/wiki/Learning_management_system

4. WebCT – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: en.wikipedia.org/wiki/WebCT

5. Moodle – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: en.wikipedia.org/wiki/Moodle

6. React (JavaScript library) - Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: en.wikipedia.org/wiki/React_(JavaScript_library)

7. Библиотека React. – [Електронний ресурс]. – Режим доступу: ru.hexlet.io/blog/posts/biblioteka-react-review-article

8. Material design – [Електронний ресурс]. – Режим доступу: uk.wikipedia.org/wiki/Material_design

9. Білодід М.М., Білоусова Н.О., Іванчук В.С., Коржова Є.В., Прищепа В.В., Танасійчук Д.О., Хлопенко О.О. Проведення практики студентів кафедри ПСТ, погляд сучасних студентів. – 2021. – С. 9–22.

ДОДАТОК А

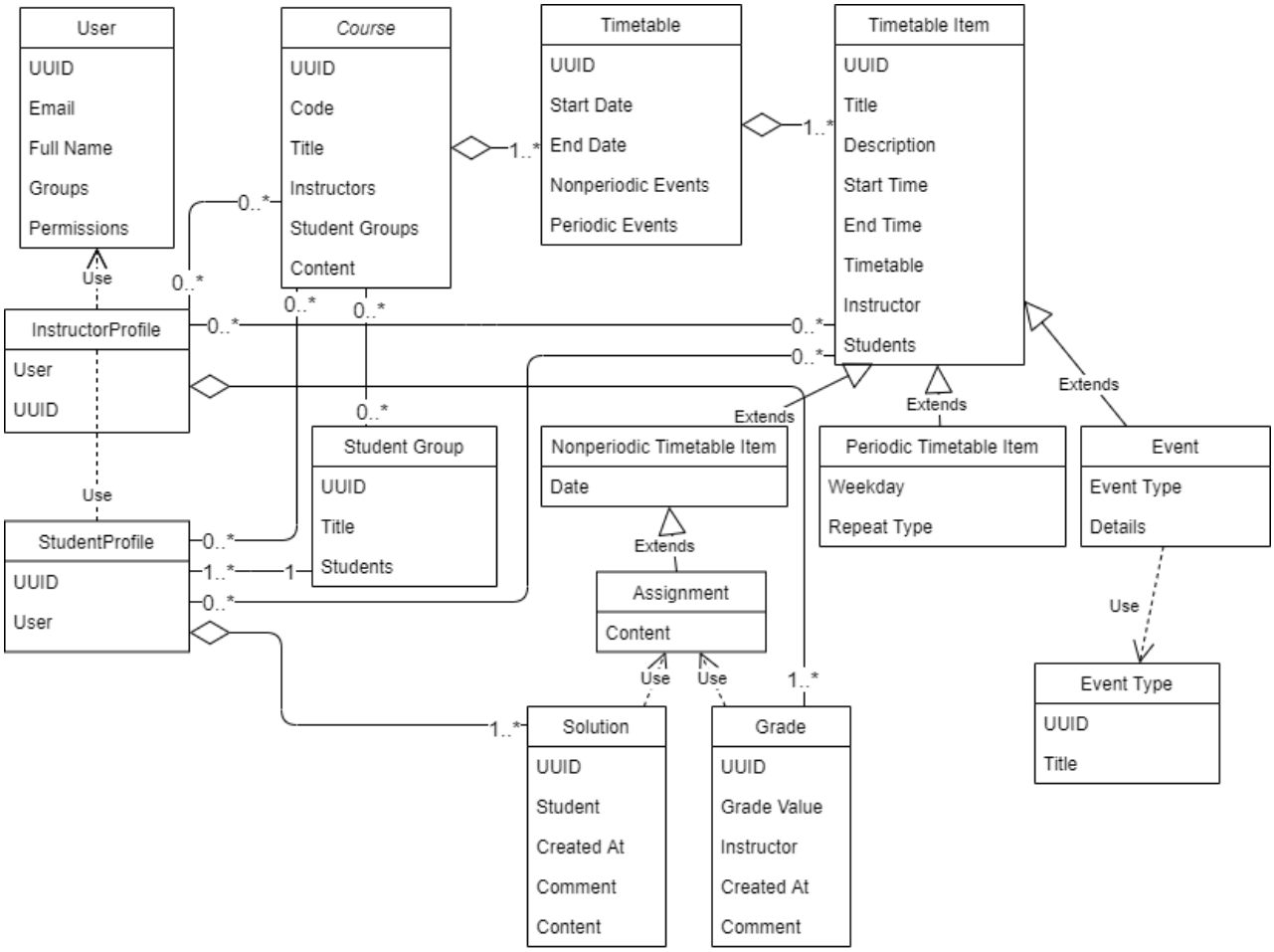

Рисунок Д1 Діаграма сутностей системи

Рисунок Д2 Діаграма авторизації та автентифікації користувача до системи

ДОДАТОК Б

Файл App.js

```
function App() {

  return (
    <div className="App">
      <BrowserRouter>
        <Route path="/" exact component={HomePage}/>
        <Route path="/login" exact component={LoginPage}/>
        <Route path="/logout" component={Logout}/>
      </BrowserRouter>
    </div>
  );
}

export default App
```

Файл LoginPage.jsx

```
const useStyles = makeStyles((theme) => ({
  root: {
    height: '100vh',
  },
  image: {
    backgroundImage: 'url(https://source.unsplash.com/collection/88072
26/)',
    backgroundRepeat: 'repeat',
    backgroundColor:
      theme.palette.type === 'light' ? theme.palette.grey[50] : theme.
palette.grey[900],
    backgroundSize: 'cover',
    backgroundPosition: 'center',
  },
  paper: {
    margin: theme.spacing(8, 4),
    display: 'flex',
    flexDirection: 'column',
    alignItems: 'center',
  },
  avatar: {
    margin: theme.spacing(1),
    backgroundColor: theme.palette.secondary.main,
  },
  form: {
    width: '100%', // Fix IE 11 issue.
    marginTop: theme.spacing(1),
  },
  submit: {
```

```
    margin: theme.spacing(3, 0, 2),
  },
}));

export const LoginPage = () => {
  const classes = useStyles();

  const history = useHistory();
  const initialFormData = Object.freeze({
    email: '',
    password: '',
  });

  const [formData, updateFormData] = useState(initialFormData);

  const handleChange = (e) => {
    updateFormData({
      ...formData,
      [e.target.name]: e.target.value.trim(),
    });
  };

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

    axiosInstance
      .post(`/auth/login/`, {
        email: formData.email,
        password: formData.password,
      })
      .then((res) => {
        localStorage.setItem('access_token', res.data.access);
        localStorage.setItem('refresh_token', res.data.refresh);
        localStorage.setItem('full_name', res.data.user.full_name);
        localStorage.setItem('email', res.data.user.email);
        localStorage.setItem('uuid', res.data.user.uuid);
        localStorage.setItem('is_staff', res.data.user.is_staff);
        axiosInstance.defaults.headers['Authorization'] =
          'JWT ' + localStorage.getItem('access_token');
        history.push('/');
        console.log(res.data);
      })
  };

  return (
    <Grid container component="main" className={classes.root}>
      <CssBaseline />
      <Grid item xs={false} sm={4} md={7} className={classes.image} />
      <Grid item xs={12} sm={8} md={5} component={Paper} elevation={6}
square>
        <div className={classes.paper}>
```

```
<Avatar className={classes.avatar}></Avatar>
<Typography component="h1" variant="h5">
  Система електронного навчання на кафедрі "ПСТ"
</Typography>
<form className={classes.form} Validate>
  <TextField
    variant="outlined"
    margin="normal"
    required
    fullWidth
    id="email"
    label="Email Адреса"
    name="email"
    autoComplete="email"
    autoFocus
    onChange={handleChange}
  />
  <TextField
    variant="outlined"
    margin="normal"
    required
    fullWidth
    name="password"
    label="Пароль"
    type="password"
    id="password"
    autoComplete="current-password"
    onChange={handleChange}
  />
  <FormControlLabel
    control={<Checkbox value="remember" color="primary" />}
    label="Запам'ятати мене"
  />
  <Button
    type="submit"
    fullWidth
    variant="contained"
    color="primary"
    className={classes.submit}
    onClick={handleSubmit}
  >
    Увійти
  </Button>
  <Grid container>
    <Grid item xs>
      <Link href="#" variant="body2">
        Не пам'ятаєте пароль?
      </Link>
    </Grid>
  </Grid>
</form>
```

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

Файл HomePage.jsx

```
const useStyles = makeStyles((theme) => ({
    root: {
      flexGrow: 1,
    },

    paper: {
        padding: theme.spacing(2),
    },
  }));

export const HomePage = () => {
    const classes = useStyles();

    if (localStorage.getItem('access_token') == null) {
        return <Redirect to="/login"/>
    } return (
        <div className={classes.root}>
            <Header />
            <Workspace />
            <Footer />
        </div>
    )
}
```

Файл Logout.jsx

```
export default function Logout() {
    const history = useHistory();

    useEffect(() => {
        const response = axiosInstance.post('/auth/logout/', {
            refresh_token: localStorage.getItem('refresh_token'),
        });
        localStorage.removeItem('access_token');
        localStorage.removeItem('refresh_token');
        localStorage.removeItem('full_name');
        localStorage.removeItem('email');
        axiosInstance.defaults.headers['Authorization'] = null;
        history.push('/login');
    });
    return(
        <div>Logout</div>
```

```
    );
}
```

Файл AccIcon.jsx

```
const useStyles = makeStyles((theme) => ({
        grow: {
            flexGrow: 1,
        },
        info: {
            padding: theme.spacing(1),
            textAlign: 'center',
            color: theme.palette.text.secondary,
        },
    }),
);

export const AccIcon = () => {
    const classes = useStyles();
    const [auth, setAuth] = React.useState(true);
    const [anchorEl, setAnchorEl] = React.useState(null);
    const open = Boolean(anchorEl);

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

    const handleClose = () => {
      setAnchorEl(null);
    };

    const history = useHistory();
    const navigateTo = () => history.push('/logout');

    return(
    <div>
        {auth && (
            <div>
              <IconButton
                aria-label="account of current user"
                aria-controls="menu-appbar"
                aria-haspopup="true"
                onClick={handleMenu}
                color="inherit"
              >
                <AccountCircleIcon fontSize="large"/>
              </IconButton>
              <Menu
                id="menu-appbar"
                anchorEl={anchorEl}
```

```
                anchorOrigin={{
                  vertical: 'top',
                  horizontal: 'right',
                }}
                keepMounted
                transformOrigin={{
                  vertical: 'top',
                  horizontal: 'right',
                }}
                open={open}
                onClose={handleClose}
              >
                <Grid container spacing={3}>
                    <Grid item xs></Grid>
                    <Grid item xs={14}>
                        <AccountCircleIcon className={classes.containe
r} color="action" style={{ fontSize: 60 }}/>
                    </Grid>
                    <Grid item xs></Grid>
                </Grid>
                <Grid container>
                    <Grid item xs className={classes.info}>
                        <Typography>{localStorage.getItem('full_name')
}</Typography>
                    </Grid>
                </Grid>
                <Grid container>
                    <Grid item xs className={classes.info}>
                        <Typography>{localStorage.getItem('email')}</T
ypography>
                    </Grid>
                </Grid>

                <Divider variant="middle"/>

                <MenuItem onClick={navigateTo}>
                  <ListItemIcon>
                    <ExitToAppIcon fontSize="medium" />
                  </ListItemIcon>
                  Вихід з аккаунту
                </MenuItem>
              </Menu>
            </div>
        )}
    </div>
  )
}
```

Файл Header.jsx

```
const useStyles = makeStyles((theme) => ({
        grow: {
            flexGrow: 1,
        },
    }),
);

export const Header = () => {
    const classes = useStyles();

    return(
        <div>
            <AppBar position="static">
                <Toolbar>
                    <HomeLink />
                    <div className={classes.grow} />
                    <NotificationIcon />
                    <AccIcon />
                </Toolbar>
            </AppBar>
        </div>
    )
}
```

Файл HomeLink.jsx

```
export const HomeLink = () => {

    return (
        <div>
            <Button
            component={Link}
            color="inherit"
            to="/">
                <h3>Система дистанційного навчання на кафедрі "ПСТ"</h
3>
            </Button>
        </div>
    )
}
```

Файл Footer.jsx

```
export const Footer = () => {
    return (
        <Box bgcolor="text.secondary" color="white">
            <Container maxWidth="lg">
                <Box textAlign="center" pt={{ sm: 2}} >
```

```
                    Факультет інформаційних технологій. Кафедра Програ
мних систем і технологій. &reg; {new Date().getFullYear()}
                </Box>
            </Container>
        </Box>
    )
}
```

Файл LeftPanelStudent.jsx

```
const useStyles = makeStyles((theme) => ({
    mainList: {
        height: 800,
    },
}),
);

export const LeftPanelStudent = () => {
    const classes = useStyles();

    return(
        <div>
            <Paper className={classes.mainList}>
                <ListItem button>
                    <ListItemIcon>
                        <TodayIcon />
                    </ListItemIcon>
                    <ListItemText primary="Календар" />
                </ListItem>
                <ListItem button>
                    <ListItemIcon>
                        <CollectionsBookmarkIcon />
                    </ListItemIcon>
                    <ListItemText primary="Предмети" />
                </ListItem>
                <ListItem button>
                    <ListItemIcon>
                        <AssessmentIcon />
                    </ListItemIcon>
                    <ListItemText primary="Оцінки" />
                </ListItem>
                <ListItem button>
                    <ListItemIcon>
                        <AccountBoxIcon />
                    </ListItemIcon>
                    <ListItemText primary="Профіль" />
                </ListItem>
                <ListItem button>
                    <ListItemIcon>
                        <EmailIcon />
```

```
                    </ListItemIcon>
                    <ListItemText primary="Повідомлення" />
                </ListItem>
                <Divider variant="middle"/>
            </Paper>
        </div>
    )
}
```