ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему
«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ДИСТАНЦІЙНОГО НАВЧАННЯ: ТЕЛЕГРАМ-БОТ, ОБЛІКОВІ ЗАПИСИ, РОЗГОРТАННЯ»

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

Виконавець роботи Тацій Денис Сергійович
_____ «____»_____ 2023 р.
(підпис)

Науковий керівник к.пед.н., доцент, Кошова Оксана Петрівна
_____ «____»_____ 2023 р.
(підпис)

Рецензент

ПОЛТАВА 2023 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри _____Олена ОЛЬХОВСЬКА
(підпис)
«___» _____ 202__ р.

ЗАВДАННЯ І КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Розробка програмного забезпечення системи дистанційного навчання: телеграм-бот, облікові записи, розгортання**»**
зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня бакалавр
Прізвище, ім'я, по батькові Тацій Денис Сергійович
Затверджена наказом ректора № 144-Н від «01» вересня 2022 р.
Термін подання студентом роботи «___»_____ 202__ р.
Вихідні дані до кваліфікаційної роботи: статті та документації з теми розробки систем дистанційного навчання.

Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ВСТУП

1. РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
2. РОЗДІЛ 2. ОГЛЯД СИСТЕМ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ
   - 2.1. Платформа дистанційного навчання ProProfs LMS.
   - 2.2. Платформа дистанційного навчання Moodle.
   - 2.3. Платформа дистанційного навчання LearnWorlds.
   - 2.4. Платформа дистанційного навчання Canvas.
3. РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
   - 3.1.Опис проектних рішень, інструментів та підходів до розробки.
   - 3.2. Опис обраного підходу до створення програмного забезпечення.
4. РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
   - 4.1. Опис та побудова діаграм роботи частин системи та розгортання.
   - 4.2. Інструкція з використання розроблених частин системи.

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК А

Перелік графічного матеріалу: 2 аркуша блок-схем, 42 ілюстрації.

Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка задачі	Кошова О.П.		
Інформаційний огляд	Кошова О.П.		
Теоретична частина	Кошова О.П.		
Практична частина	Кошова О.П.		

Календарний графік виконання кваліфікаційної роботи

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

Дата видачі завдання «__» _____ 202___ р.

Здобувач вищої освіти _____ Тацій Денис Сергійович
(підпис)

Науковий керівник _____ к.пед.н., доц. Кошова О.П.
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Результати захисту кваліфікаційної роботи

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

Протокол засідання ЕК № ____ від «____» _____ 2023 р.

Секретар ЕК_____ _____
(підпис) (ініціали та прізвище)

Затверджую
Зав. кафедрою _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«____» _____ 202__ р.

Погоджено
Науковий керівник _____
к.пед.н., Оксана КОШОВА
«____» _____ 202__ р.

План

кваліфікаційної роботи на тему
«Розробка програмного забезпечення системи дистанційного навчання: телеграм-бот, облікові записи, розгортання»
зі спеціальності 122 Комп'ютерні науки
освітня програма 122 «Комп'ютерні науки»
ступеня бакалавр
Прізвище, ім'я, по батькові Тацій Денис Сергійович

Здобувач вищої освіти _____ Денис ТАЦІЙ
«____» _____ 202__ р.

РЕФЕРАТ

Записка: 82 сторінки, 42 рисунки, 1 додаток, 28 літературних джерел.

Мета роботи – розробка програмного забезпечення системи дистанційного навчання: облікові записи, телеграм-бот, розгортання.

Об’єкт розробки – дистанційне навчання студентів.

Методи дослідження – використання електронних джерел: статті та документації, використання технологій та інструментів розробки програмного забезпечення на клієнт-серверній архітектурі: платформа Node.js, мова програмування TypeScript, серверний фреймворк NestJS, клієнстька бібліотека React, система управління базами даних PostgreSQL, інструмент для ізоляції програмного забезпечення у віртуалізованих контейнерах Docker, розподілена система контролю версій Git. Систему та її частини було розроблено на клієнт-серверній архітектурі.

Сформульовано вимоги до програмного забезпечення, що було у розробці. Проведено дослідження та аналіз аналогічних систем дистанційного навчання. Складено список технологій та інструментів розробки, і проведено їх опис. Побудовано діаграму прецедентів, що описує функціональність програмного забезпечення. Розроблено програмне забезпечення для частин: облікові записи та телеграм-бот. Проведено розгортання системи дистанційного навчання на віддаленому сервері. Розроблено інструкцію для користувача розробленого програмного забезпечення. Проведено опис процесу розгортання системи дистанційного навчання.

ЗМІСТ

ВСТУП

Актуальність: досвід минулих років показав, що питання розвитку дистанційного навчання є досить актуальним, адже такий вид навчання допомагає у вирішенні проблеми, коли люди не можуть перебувати всі разом в одній аудиторії. Дистанційна форма навчання надає змогу приймати участь у освітньому процесі з будь-якої точки планети як студентам, так і викладачам, що є вагомим плюсом, бо надає змогу повноцінно навчатися або викладати людям, які живуть у важкодоступній місцевості, знаходяться за кордоном тощо. Також до переваг такої форми навчання можна віднести можливість підлаштувати навчання під свій індивідуальний графік, що дозволить навчатись у власному темпі та у зручний час, бо студент, навіть будучи відсутнім на занятті, має доступ до всіх матеріалів, які він може перечитати в будь-який момент, також може переглянути відео-запис лекції та зв’язатися з викладачем online, якщо в нього виникнуть питання. Отже, за допомогою дистанційного навчання, студент може отримувати знання та працювати на роботі або навіть навчатися в декількох навчальних закладах.

Система дистанційного навчання не тільки забезпечує вище перелічені можливості, а й може надавати деякі додаткові: онлайн-тестування, архівація пройдених курсів та оцінок студентів, чат з викладачем, онлайн-журнал оцінок, email-розсилка тощо. Наприклад, онлайн-тестування може зекономити час викладачеві на перевірку, адже система сама порахує кількість правильних відповідей та виставить оцінку, або її може виставити викладач за результатами тестування, в свою чергу онлайн-журнал може допомогти студенту дізнатися оцінки за здані роботи та оцінити свою успішність. Якщо студентам або викладачу потрібно переглянути інформацію за пройдені курси, то тут стане у пригоді архівація курсів, що дозволить переглянути матеріали, завдання та оцінки. Якщо ж у студента чи викладача виникає питання по якійсь темі або завданню, то онлайн-чат може допомогти у їх вирішенні. Також email-розсилка

надає можливість студентам дізнаватися про нові завдання, оцінки, повідомлення тощо.

Зважаючи на вказану вище актуальність, було вирішено розробити програмне забезпечення системи дистанційного навчання, а саме наступні його частини: створення та керування обліковими записами користувачів та телеграм-бот, що надає інформацію про розклад, також забезпечити розгортання системи для її повноцінного функціонування та використання.

Метою роботи є розробка програмного забезпечення системи дистанційного навчання, а саме його частин: телеграм-бот, облікові записи, розгортання.

Об'єктом розробки є дистанційне навчання студентів.

Предметом розробки є програмне забезпечення системи дистанційного навчання, а саме його частини: облікові записи, телеграм-бот, розгортання.

Обсяг та структура проекту. Кваліфікаційна робота складається з вступу, 3 розділів, висновків, та списку використаної літератури. Він викладений на 82 сторінках та ілюстрований 10 рисунками.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

Було вирішено розробити програмне забезпечення системи дистанційного навчання на клієнт-серверній архітектурі, тобто має бути єдиний веб-сервер, який буде контролювати поведінку клієнтської частини, що повинна бути представлена у вигляді веб-сайту, та оперувати даними з бази даних. Також було вирішено розробити телеграм-бота, для перегляду розкладу занять, як окремий сервіс.

В рамках розробки системи дистанційного навчання було поставлено задачу з реалізації таких частин проекту: облікові записи, телеграм-бот та розгортання.

Облікові записи. Реалізація частини з обліковими записами користувачів системи повинна включати розробку як клієнтської, так і серверної логіки. Алгоритм авторизації має бути побудований на основі токену, що зберігає дані користувача для подальшої авторизації користувача. Клієнт повинен отримувати цей токен від серверу після успішної авторизації користувача, а в подальшому використовувати для авторизації запитів до серверу. Якщо буде здійснено повторний вхід в обліковий запис, то старий токен повинен бути анульованим, тим самим, користувачу із старим токеном потрібно буде ще раз увійти до облікового запису. Отже, повинен бути завжди тільки один валідний токен і тільки один авторизований користувач облікового запису. Також, після успішної авторизації, користувач повинен мати можливість вийти з облікового запису.

Також потрібно створити модальне вікно з формою для авторизації та налаштувати комунікацію клієнта і серверу для успішної її роботи. Форма для авторизації повинна проводити валідацію введених даних на стороні клієнта та виводити текст помилок, якщо валідація не була пройдена успішно або відповідне повідомлення повернув сервер.

Окрім авторизації, завдання передбачає реалізацію, на стороні клієнта, панелі адміністратора з присутніми в системі користувачами, що не є

адміністраторами, для керування їхніми обліковими записами. В системі дистанційного навчання такими користувачами можуть бути тільки студенти або викладачі. Тому для цих двох ролей потрібно створити окремі сторінки, що полегшить відокремлення одних від інших. Також панель адміністратора повинна надавати можливість керувати даними про групи та спеціальності системи, також за допомогою виділення окремих сторінок під цю задачу.

Сторінки зі студентами та викладачами повинні мати наступну функціональність:

- створення нового користувача;
- перегляд наявних користувачів у вигляді таблиці даних;
- редагування інформації про користувача;
- можливість згенерувати новий пароль користувача;
- видалення користувача.

Створення та редагування користувача повинно здійснюватися через модальне вікно, яке повинно відкриватися при натисканні на відповідну кнопку. Модальне вікно повинно мати форму для введення даних користувача, що в свою чергу повинна мати клієнтську валідацію та вивід помилок у разі їх наявності. Видалення користувача повинно відбуватися при натисканні на кнопку з іконкою сміттєвого кошика.

Для відправки паролю для входу новому користувачеві має бути реалізований, на серверній частині, сервіс email-розсилки.

Для користувачів з роллю студента потрібно додати можливість редагувати список груп.

Керування даними про групи та спеціальності має відбуватися так само через окремі сторінки і мати функціональність подібну до сторінок зі студентами та викладачами, тобто: створення, перегляд, редагування та видалення.

Телеграм-бот. Реалізація функціональності перегляду розкладу через телеграм-бот включає розробку як серверної логіки системи дистанційного навчання, так і самого сервісу телеграм-боту.

Розробка серверної логіки для забезпечення можливості перегляду розкладу через телеграм-бот має полягати у реалізації частини модуля для роботи з курсами у вигляді: обробників контролера з курсів, що приймають запити на створення, отримання, редагування та видалення даних про сутності розкладу та сутність розкладу, яка описує наступні поля:

– ідентифікатор;
– дата;
– час початку;
– час закінчення.

Також повинен бути сервіс, що буде обробляти та передавати дані про розклад між контролером та базою даних.

Далі потрібно буде додати функціональність авторизації користувача системи в телеграм-боті, шляхом реалізації наступних частин: сервіс та модуль кешування даних, для зберігання тимчасового ключа, який буде використовуватись для авторизації в телеграм-боті; перехоплювачі користувацьких запитів отримання розкладу для перевірки на те, чи справді ці запити відправляються з телеграм-боту авторизованим в ньому користувачем та відповідний обробник для авторизації в самому телеграм-боті.

Розробка сервісу телеграм-боту повинна включати реалізацію його відокремленої від системи роботи та можливість комунікації із сервером для передачі даних авторизації і розкладу користувача, а також можливість виводу цього розкладу у чаті.

Насамкінець, для того, щоб почати роботу з телеграм-ботом, користувач повинен буде перейти до налаштувань свого облікового запису в системі, де має бути кнопка для підключення до телеграм-боту, після натискання якої повинен здійснитися перехід до сторінки з телеграм-ботом для подальшої авторизації користувача в ньому. Після успішної авторизації телеграм-бот повинен надавати доступ до клавіатури головного меню, де має знаходитися три кнопки: «Розклад на сьогодні», «Розклад на завтра» та «Розклад на тиждень». Після натискання кнопки «Розклад на стогодні» телеграм-бот

повинен відправити розклад на сьогоднішній день за всіма курсами користувача, відповідно до назви і у випадку натискання на кнопку «Розклад на завтра», також у обох випадках повинна бути клавіатура під повідомленням про розклад, що дозволить перемикатися між днями. В свою чергу після натискання кнопки «Розклад на тиждень», повинен бути відправлений розклад на поточний тиждень, і також з можливістю перемикатися до наступних або попередніх тижнів за допомогою стрілок внизу повідомлення. Якщо розкладу немає або його неможливо отримати, то телеграм-бот повинен повідомлення про відсутність розкладу.

Розгортання. В результаті розробки програмного забезпечення системи дистанційного навчання, повинно бути проведено його розміщення на хост-сервісі, щоб зробити можливим доступ до системи дистанційного навчання в будь-який час та в будь-якому місці, де є можливість виходу в мережу Інтернет.

Спочатку потрібно вибрати хост-сервіс, далі орендувати віддалений сервер і вже на ньому розмістити систему. Для того, щоб проект запрацював на віддаленому сервері, потрібно також встановити та налаштувати відповідне середовище, де будуть працювати всі складові розробленої системи. Після проведення налаштування та розміщення серверної частини на віддаленому сервері, потрібно запустити сам сервер, зв’язати його з клієнтською частиною, після чого розмістити на віддаленому сервері і клієнтську частину.

Таким чином, система повинна запрацювати та стати доступною для інших користувачів мережі Інтернет, в результаті чого нею можна буде повноцінно користуватися.

РОЗДІЛ 2. ОГЛЯД СИСТЕМ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ

2.1. Система дистанційного навчання ProProfs LMS

ProProfs LMS – система дистанційного навчання, що надає можливість легко організувати та керувати онлайн-курсами та матеріалами до них, що є досить необхідним в умовах сьогодення (Рисунок 2.1).

Рисунок 2.1 – Система дистанційного навчання ProProfs LMS

ProProfs LMS підтримує велику кількість мов, що дає можливість викладати в багатьох куточках світу, навчаючи величезну кількість людей без особливих проблем.

Дана система дистанційного навчання є досить популярною і сягає більш ніж 15 мільйонів унікальних користувачів з понад 150 країн світу, тому що ця система орієнтована на забезпечення гарного досвіду користування.

Особливості системи:

- спрощений процес проведення онлайн-занять за допомогою системи віртуально класу;
- можливість інтеграції системи з іншими інструментами, наприклад: відеодзвінки, соціальні медіа платформи та системами керування відносинами користувачів;
- легке створення власних курсів як з вже існуючими матеріалами, так і з нуля;
- наявність бібліотеки вже створених та підготовлених курсів та шаблонів;
- доступність у будь-який час і будь-де;

– можливість проведення обговорень серед студентів через онлайн-форум на основі запитань і відповідей;
– тестування та оцінювання;
– здійснення трекінгу та звітності, що дозволяє відстежувати активність студентів та переглядати їх прогрес;

Плюси системи:

– проста у використанні;
– можливість повного налаштування;
– підтримка української мови;
– бібліотека готових курсів та шаблонів.

Мінуси системи:

– не інтегрується з іншими подібними системами;
– немає безкоштовного плану.

Ціна ліцензії починається від 2 $ за учня/місяць (рахунок виставляється щорічно), також система має безкоштовну пробну версію для всіх планів протягом 15 днів [1].

2.2. Система дистанційного навчання Moodle

Moodle (Modular Object-Oriented Dynamic Learning Environment) – система керування навчанням, основною задачею якої є забезпечення освітнього середовища для організації дистанційного навчального процесу (Рисунок 2.2).

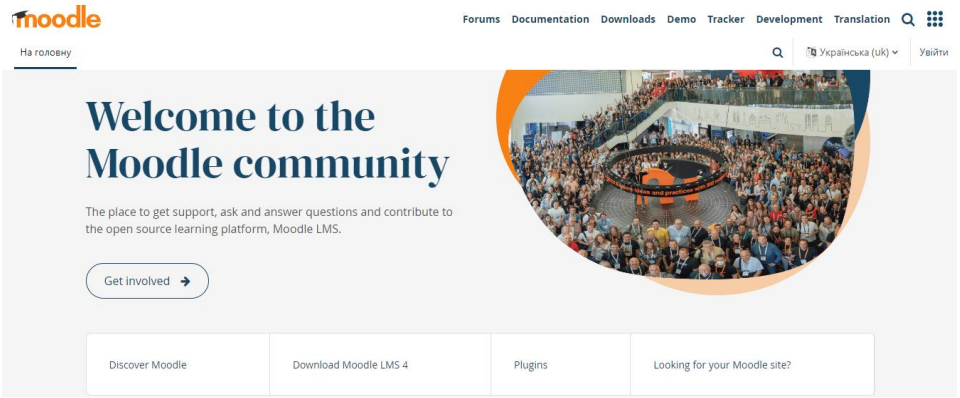

Рисунок 2.2 – Система дистанційного навчання Moodle

Ця система є досить популярну та налічує понад 18 млн. унікальних користувачів в світі, бо є безкоштовною та має відкритий програмний код.

Платформа надає простір для спільної роботи вчителів і студентів. У Moodle доступні різні можливості для відстеження успішності учнів, а також є підтримка реєстрації з безпечною аутентифікацією. Moodle можна налаштувати відповідно до власних уподобань в дизайні і вибрати відповідний макет. Платформу можна інтегрувати з великою кількістю програмного забезпечення, включаючи інструменти для спілкування, спільної роботи, управління документами та інші додатки для підвищення продуктивності. У Moodle відкритий вихідний код, ви можете використовувати систему безкоштовно.

Плюси системи:

- систему може встановити кожен, вона встановлюється безкоштовно;
- необмежена кількість користувачів;
- доступність у будь-який час і будь-де;
- величезна кількість налаштувань і плагінів (більше 1500);
- у Мережі велика кількість інструкцій для установки і налаштування Moodle системи;
- підтримка більш ніж 100 мов, включаючи українську;
- надано спеціальний редактор для створення різних видів матеріалу;
- підтримка різних форматів файлів матеріалів: mp3, aac, flac, m4a, oga, ogg, wav; doc, pdf, xls, csv; jpeg, png, gif; flv, f4v, f4p, mp4, m4v, m4a, 3gp, mov;
- вбудований редактор для створення онлайн тестів, крім того можна використовувати сторонні сервіси для розробки тестів;
- можливість складати звіти по активності користувачів системи;
- система підтримується мобільними браузерами Chrome і Safari, тому навчання може проводиться з мобільних пристроїв користувачів;
- Moodle надає можливість організації продажу курсів;
- є хмарна версія системи для тесту, вона доступна на офіційному сайті Moodle.

Мінуси системи:

– складне адміністрування та налаштування, для початківців користувачів може зайняти багато часу;
– для розробки власних курсів необхідно встановлювати і збирати з нуля;
– дизайн можна вибрати тільки з готових варіантів рішень;
– в системі досить складний інтерфейс для роботи звичайних користувачів і викладачів, які завантажують свої матеріали для навчання.

Освітня платформа Moodle приваблива для користувачів з багатьох обґрунтованих причин, але, як і у всіх інших CMS, у неї є свої недоліки. Система безкоштовна, і це плюс, але, якщо у вас не вийде її налаштувати, то все одно доведеться платити стороннім фахівцям за допомогу. Також знадобиться час для вивчення роботи платформи і її панелі управління. Більшість компаній вважають за краще вже готові рішення, але за такі пропозиції необхідно платити, в Moodle набір основного функціоналу повністю безкоштовний [2].

2.3. Система дистанційного навчання LearnWorlds

LearnWorlds – це хмарна платформа, яка є комплексним рішенням, що пропонує можливість побудувати вашу власну онлайн-академію, створюючи унікальний та інтерактивний досвід навчання (Рисунок 2.3).

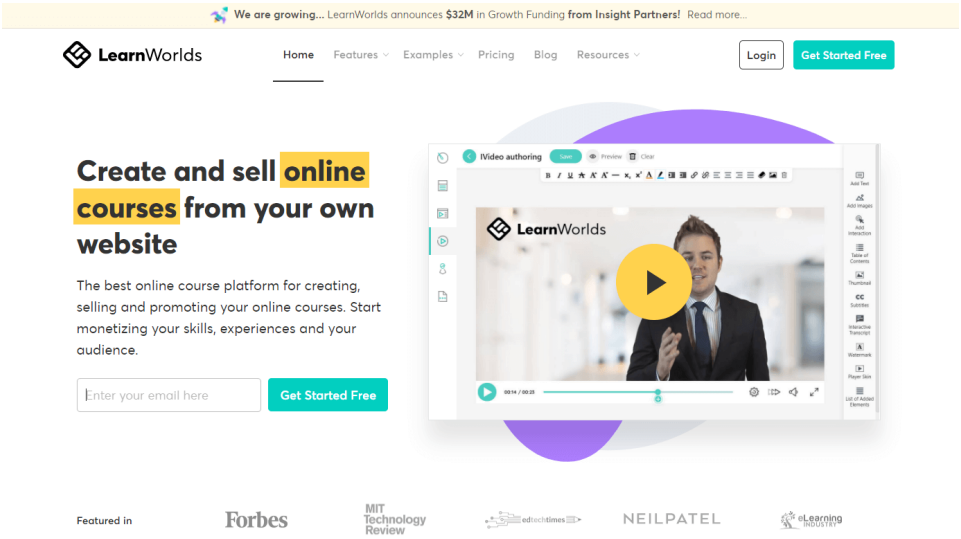

Рисунок 2.3 – Система дистанційного навчання LearnWorlds

Ця платформа об’єднує все, що потрібно для дистанційного навчання, в одному місці, роблячи процес створення онлайн-курсів простим і водночас доступним. Крім того, вона розроблена, щоб надати правильні маркетингові інструменти, які допоможуть просувати та продавати онлайн-курси. Вона також має потужний конструктор веб-сайтів і шаблони сторінок.

LearnWorlds ідеально підходить для підприємців, індивідуальних інструкторів, викладачів, малого та середнього бізнесу, професійних тренерів, компаній, які хочуть навчати своїх співробітників або навчати клієнтів.

Плюси системи:

- легке налаштування;
- просте створення курсів;
- доступність у будь-який час і будь-де;
- не потребує жодних технічних навичок для використання;
- пропонує сучасний конструктор веб-сайтів;
- пропонує повністю настроюваний конструктор мобільних додатків для iOS та Android без кодування;
- має вбудовані функції електронної комерції, маркетингу та партнерської діяльності;
- включає різноманітні засоби навчання та механізми оцінювання;
- пропонує надійну систему передумов;
- має функції онлайн-спільноти та обговорення;
- підтримує файли SCORM і HTML5;
- пропонує оперативну підтримку клієнтів 24/7 із адаптацією для нових клієнтів;

Мінуси системи:

- жодних заздалегідь створених воронок продажів;
- пропонує обмежені вбудовані можливості електронного маркетингу.

Вона пропонує 30-денну безкоштовну пробну версію, а потім плани підписки, які починаються від 29 $ на місяць (плюс 5 $ за курс) [3].

2.4. Система дистанційного навчання Canvas

Canvas – це навчальна онлайн-платформа з академічною спрямованістю. Однак онлайн-викладачі часто використовують його для створення незалежних курсів, тобто вони не пов’язані з жодним типом школи чи бізнесу. Однією з його найкращих особливостей є те, що він надзвичайно гнучкий у плані підтримки різних форматів навчання. Canvas надає можливості створення веб-сайтів для розміщення вашого курсу. Забезпечує відносну свободу налаштування та адаптування сайту за потреби. Крім того, Canvas надає все необхідне, включаючи тести, журнали оцінок, інструменти звітності та інтеграції (Рисунок 2.4).

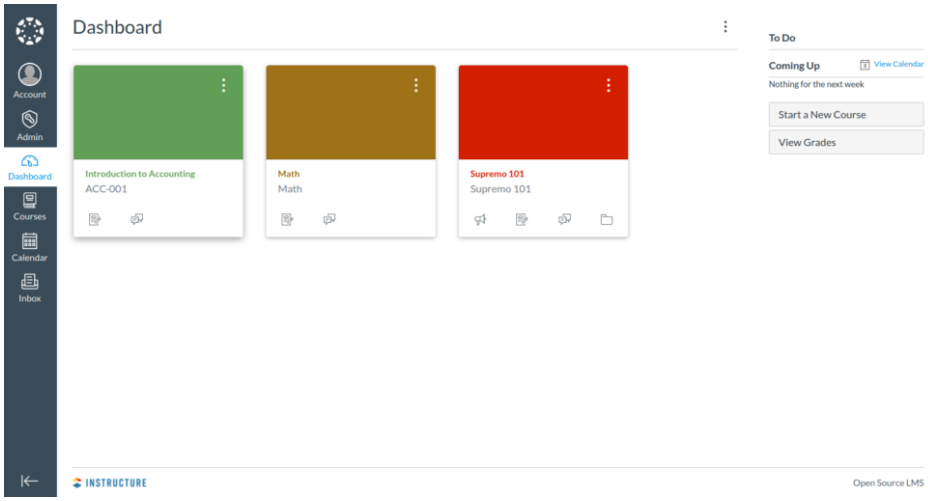

Рисунок 2.4 – Система дистанційного навчання Canvas

Присутня велика кількість можливостей інтеграцій, зокрема Microsoft Teams, Google Workspace і Adobe Creative Cloud. Це особливо важливо, якщо команда розробників курсу розподілена, але є потреба співпрацювати та бачити прогрес один одного. Говорячи про прогрес, функція Canvas SpeedGrader спрощує оцінювання завдань, які система не може автоматично оцінити.

Canvas має спільноту, що пов’язує студентів, де викладачі можуть спілкуватися зі студентами за допомогою аудіо та відео, а також надсилати приватні чи публічні повідомлення. Ця платформа також пропонує багато інструментів для студентів: дошки для обговорень, семінари з відеоконференцій, тощо, які дозволяють їм спілкуватися один з одним.

Canvas – це надзвичайно безпечна система керування навчанням, яка відповідає як організаційним, так і навчальним цілям. Вона чудово підходить для користувачів, які хочуть мати кілька інтерфейсів, кожен з яких оформлений для певної аудиторії. Коли справа доходить до простих речей, Canvas досить інтуїтивно зрозумілий. Але щоб повністю розкрити його потенціал потрібно витратити час на навчання.

Плюси системи:

- все необхідне відразу з коробки;
- підтримка української мови;
- великий вибір інтеграцій;
- доступність у будь-який час і будь-де;
- інструменти організації навчання та співпраці;
- функція SpeedGrader для скорочення часу, який витрачається на оцінювання завдань.

Мінуси системи:

- для повного розкриття потенціалу потрібне інтенсивне навчання.

Canvas – як платна, так і безкоштовна онлайн-платформа для навчання [4].

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Опис проектних рішень та інструментів розробки

Беручи до уваги постановку завдання та специфіку системи, що розробляється, а також в результаті аналізу існуючих інструментів та проектних рішень, було вирішено використати наступні засоби та інструменти розробки:

- платформа: Node.js;
- мова програмування: TypeScript;
- клієнтська частина: React, Vite, Redux, Redux Toolkit, Saga, Axios, Zod, Ant Design;
- серверна частина: NestJs, Passport, TypeORM, Nodemailer, Swagger;
- СУБД: PostgreSQL;
- інструмент адміністрування та розробки для PostgreSQL: pgAdmin;
- розподілене сховище даних, що зберігає інформацію в пам’яті: Redis;
- бібліотека для створення телеграм-ботів за допомогою JavaScript або TypeScript: Telegraf;
- розподілена система контролю версій: Git;
- середовище розробки: Visual Studio Code;
- інструмент розробки дизайну клієнтської частини: Figma;
- програмне забезпечення для автоматизації розгортання і керування додатками: Docker;
- сервіс хостингу у хмарі: DigitalOcean.

Node.js – це середовище, із відкритим програмним кодом, для запуску додатків написаних на мові програмування JavaScript. Node.js кросплатформний, що показує його можливість працювати на різних операційних системах, таких як: Windows, Linux, Unix, macOS тощо.

Node.js дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка та сценаріїв на стороні сервера.

Функціональність запуску сценаріїв на стороні сервера створює динамічний вміст веб-сторінки перед тим, як сторінка надсилається у веб-браузер користувача. Отже, Node.js представляє парадигму «JavaScript всюди», об'єднуючи розробку веб-додатків навколо однієї мови програмування, а не різних мов для серверних і клієнтських сценаріїв [5].

Головною перевагою Node.js є її неблокуюча модель введення-виведення, що дозволяє системі, побудованій на Node.js, обробляти велику кількість користувачів, вибудовуючи чергу за пріоритетністю, в результаті чого набагато легше впоратися з навантаженням, бо немає потреби виділяти окремий потік для кожного з'єднання. Це досягається за рахунок того, що ця система керується подіями та працює асинхронно (Рисунок 3.1).

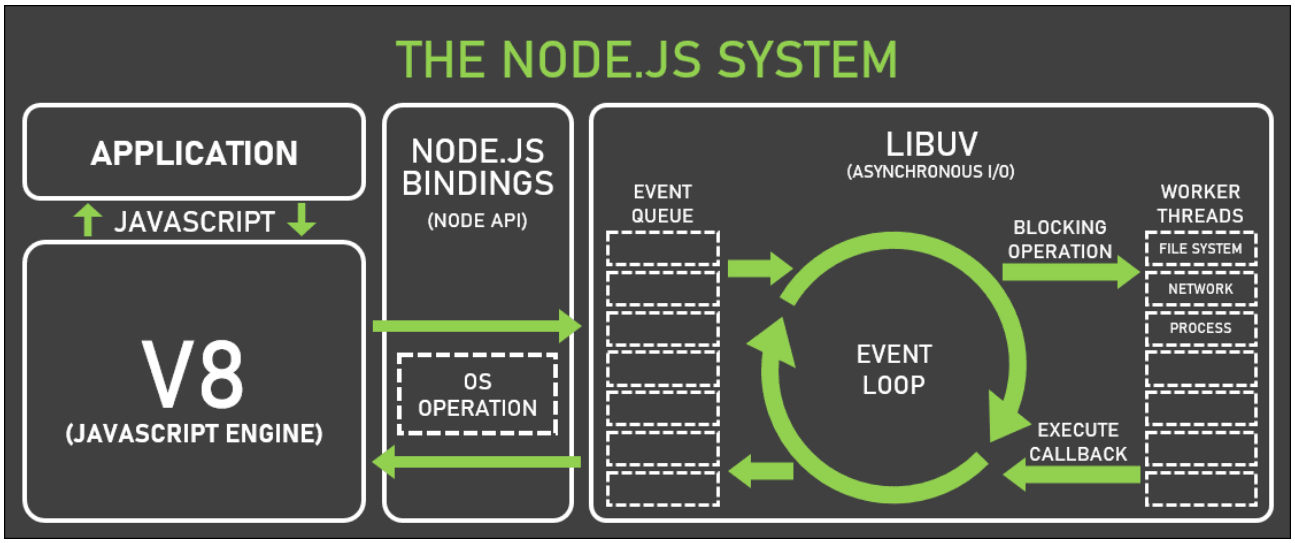

Рисунок 3.1 – Архітектура Node.js

Також і інші переваги Node.js:

– Швидкість. Робоча система, що може впоратися з великою кількістю користувачів одночасно може бути розроблена досить швидко та в подальшому масштабована до більш серйозного продукту.
– Одна мова програмування на клієнті та сервері. Оскільки Node.js використовує мову JavaScript, то розробник клієнтської частини, що працює також на JavaScript, може швидко навчитися розробляти і серверну частину, тим самим зможе бути відповідальним за розробку всього проекту та відповідно бути ціннішим працівником.

– Досить велика та активна спільнота. Існує досить велика кількість модулів та пакетів, що досить добре комбінуються в екосистемі Node.js. Ці модулі розміщуються, розповсюджуються та встановлюються за допомогою менеджеру пакетів npm, що входить до складу Node.js.

Дивлячись на всі перелічені вище переваги платформи, було прийнято рішення використати її у розробці системи.

TypeScript – це строго типізована, об’єктно-орієнтована, компільована мова. Він була розроблена Андерсом Хейлсбергом (розробником C#) у Microsoft. TypeScript – це і мова, і набір інструментів. TypeScript – це типізована надбудова JavaScript, що компілюється у JavaScript. Іншими словами, TypeScript – це JavaScript плюс деякі додаткові функції [6].

Головні переваги TypeScript над JavaScript:

– Сувора типізація. Через динамічну типізацію JavaScript, робота з ним може призвести до появи великої кількості помилок, що забере досить багато часу на їх вирішення, тому TypeScript використовує статичну типізацію, що дозволяє побачити помилку ще на етапі написання коду. В TypeScript є числовий тип, рядковий, логічний тощо. Також TypeScript дозволяє створювати і свої власні типи даних.
– Покращене ООП. Можливість використання об’єктно-орієнтованого програмування присутня як в TypeScript, так і в JavaScript, але в TypeScript більш покращене використання ООП, бо він має дещо більше можливостей: створення інтерфейсів, додавання модифікаторів доступу, визначення полів в конструкторі, абстрактні класи, узагальнення, перетворення типів тощо.

З огляду на перелічені вище переваги, було вирішено обрати мовою програмування системи саме TypeScript, що дозволить більш гнучко та швидко писати код.

React – це JavaScript бібліотека, розроблена командою Facebook, для створення веб-сайтів або користувацьких інтерфейсів, що побудовані на основі композиції блоків окремих компонентів, які пов’язані між собою.

Основні переваги React:

- Просте створення динамічних додатків. React полегшує створення динамічних веб-додатків, оскільки вимагає менше коду та пропонує більше функціональних можливостей, на відміну від JavaScript.
- Покращена продуктивність. React використовує Virtual DOM, завдяки чому швидше створюються веб-додатки. Віртуальний DOM порівнює попередні стани компонентів і оновлює лише ті елементи в реальному DOM, які були змінені, замість того, щоб оновлювати всі компоненти знову, як це роблять звичайні веб-програми.
- Багаторазові компоненти. Компоненти є будівельними блоками будь-якої програми React, і одна програма зазвичай складається з кількох компонентів. Ці компоненти мають свою логіку та елементи керування, і їх можна повторно використовувати в усій програмі, що, у свою чергу, значно скорочує час розробки програми.
- Односпрямований потік даних. React слідує за односпрямованим потоком даних. Це означає, що під час розробки програми React розробники вкладають дочірні компоненти в батьківські компоненти. Оскільки дані передаються в одному напрямку, стає легше виправляти помилки та знати, де виникає проблема в програмі в даний момент.
- Невелика крива навчання. React легко освоїти, оскільки він здебільшого поєднує базові концепції HTML і JavaScript з деякими корисними доповненнями. Однак, як і у випадку з іншими інструментами, вам доведеться витратити деякий час, щоб правильно зрозуміти бібліотеку React.
- Його можна використовувати для розробки як веб-, так і мобільних додатків за допомогою фреймворку під назвою React Native, що є похідним від самого React, який надзвичайно популярний і використовується для створення мобільних додатків.
- Спеціальні інструменти для легкого налагодження. Facebook випустив розширення Chrome, яке можна використовувати для налагодження програм

React. Це робить процес налагодження веб-додатків React швидшим і простішим [7].

З огляду на перелічені переваги, було вирішено обрати саме цю бібліотеку для побудови клієнтської частини.

Vite – це інструмент для побудови сучасних веб-додатків, який має на меті забезпечити швидшу та ефективнішу їх розробку [8].

Vite допомагає розроблювати веб-додатки на React набагато швидше так компфортніше.

Переваги Vite:

- простота налаштування в порівнянні з іншими схожими інструментами;
- наявність заздалегідь створеної конфігурації для забезпечення роботи більшості інструментів «одразу з коробки».

Оскільки даний інструмент забезпечує досить комфортну розробку веб-додатків, було вирішено використати його для розробки системи.

Redux – це бібліотека JavaScript з відкритим кодом, покликана реалізувати централізований доступ до стану програми або просто сховище для зберігання стану змінних у програмі. Redux створює процес і процедури для взаємодії зі сховищем, щоб компоненти не оновлювали або не читали сховище випадковим чином [9].

З огляду на зручність та зрозумілість керування станом додатку за допомогою Redux, його було обрано для розробки клієнтської частини додатку.

Redux Toolkit – це набір інструментів, який допомагає спростити розробку Redux. Він містить утиліти для створення сховищ Redux і керування ними. Команда Redux рекомендує використовувати Redux Toolkit щоразу, коли вам потрібно використовувати Redux [10].

З огляду на полегшення та прискорення написання Redux-коду за допомогою цього набору інструментів, його було вирішено використати під час розробки клієнтської частини додатку.

Redux-Saga – це бібліотека проміжного програмного забезпечення, яка використовується для асинхронної взаємодії сховища Redux із зовнішніми

ресурсами. Вона включає в себе: відправку HTTP-запитів до зовнішніх служб, доступ до сховища браузера та виконання операцій введення/виведення. Ці операції також відомі як побічні ефекти. Redux-Saga допомагає організувати ці побічні ефекти таким чином, щоб легше керувати ними [11].

З метою забезпечення роботи побічних ефектів в легший спосіб, було вирішено використати цю бібліотеку.

Axios – це бібліотека Javascript, яка використовується для виконання HTTP-запитів з процесу Node.js або XMLHttpRequests із браузера [12].

Він полегшує роботу із запитами, що, без сумніву, є великим плюсом, тому його було обрано для налаштування комунікації клієнтської частин зі серверною.

Zod – це бібліотека для оголошення TypeScript-схем і валідації. Термін «схема» використовується для широкого позначення будь-якого типу даних, від простого рядка до складного вкладеного об’єкта.

Zod розроблено так, щоб він був максимально зручним для розробників. Його мета полягає в тому, щоб усунути повторювані оголошення типів. З Zod ви оголошуєте валідатор один раз, і Zod автоматично визначить статичний тип TypeScript. Простіші типи легко поєднувати в складні структури даних [13].

Zod ідеально підходить для валідації введених даних користувачем у будь-які форми клієнтської частини.

З огляду на можливості цієї бібліотеки, її було обрано для валідації користувацьких форм на клієнтській частині системи.

Ant Design – це бібліотека React UI, яка містить прості у використанні компоненти, корисні для створення інтерактивних інтерфейсів користувача. Він дуже простий у використанні та інтеграції. Це один із розумних варіантів розробки веб-додатків за допомогою React. Він надає високоякісні компоненти, якими можна легко користуватися [14].

Особливості:

- має набір якісних компонентів React «з коробки»;
- має підтримку інтернаціоналізації для десятків мов, включаючи українську;

- надає можливість розробити інтерфейс користувача корпоративного класу;
- компоненти написані на TypeScript.

З огляду на можливості цієї бібліотеки, її було обрано для створення та оформлення інтерфейсу клієнтської частини.

NestJS – це фреймворк, який допомагає створювати серверні програми на платформі Node.js. Фреймворк Nest створено за допомогою TypeScript, який дозволяє розробникам створювати високомасштабовані та тестовані програми.

Створений на основі Express.JS, NestJS пропонує численні функції та готові API. Розробники можуть використовувати функції NestJS для створення програм із меншою кількістю коду, що, без сумніву, пришвидшує розробку проекту.

Основні концепції NestJS:

- Модулі. Модулі допомагають упорядкувати код і розділити функції на логічні одиниці для багаторазового використання. Згруповані файли TypeScript прикрашені декоратором «@Module», який надає метадані. NestJS використовує ці метадані для організації структури програми.
- Провайдери. Провайдери абстрагують будь-яку форму складності та логіки. Провайдери можуть бути створені та ін’єктовані в контролери або інші провайдери.
- Контролери. Контролери можуть обробляти вхідні запити та повертати відповіді клієнтській стороні програми (наприклад, виклик API).

Деякі з найкращих функцій NestJS:

- використовує TypeScript;
- потужний інтерфейс командного рядка (CLI);
- містить вбудований контейнер DI (впровадження залежностей або ін’єкція залежностей), який робить програми модульними та ефективнішими;
- модулі відкладеного завантаження, які завантажують речі, коли це необхідно;

- пропонує підтримку для десятків спеціальних модулів Nest, які допомагають легко інтегруватись із загальними технологіями та концепціями, такими як TypeORM, логування, валідація, кешування та багато іншого;
- прості програми для модульного тестування;
- надає детальну документацію [8].

З огляду на вище перелічені функцій та можливостей NestJS, було вирішено, що цей фреймворк повністю задовольняє потреби при розробці програмного забезпечення системи дистанційного навчання.

Passport – найпопулярніша бібліотека автентифікації Node.js, добре відома спільноті та успішно використовується в багатьох робочих програмах.

Цю бібліотеку легко інтегрувати з NestJS. На високому рівні Passport виконує ряд кроків, щоб:

- авторизувати користувача, підтвердивши його «облікові дані» (наприклад, ім’я користувача/пароль, веб-токен JSON (JWT));
- керувати авторизованим станом (шляхом видачі портативного токена, наприклад JWT, або створення експрес-сеансу);
- додати інформацію про авторизованого користувача до об’єкта Request для подальшого використання в обробниках маршрутів.

Passport має багату екосистему стратегій, які реалізують різні механізми автентифікації. Незважаючи на просту концепцію, набір стратегій Passport, які ви можете вибрати, великий і представляє велику різноманітність [16].

З огляду на вище описані особливості цієї бібліотеки, її було обрано для реалізації авторизації в системі.

TypeORM – це ORM, що може працювати на платформах Node.js, Browser, Cordova, PhoneGap, Ionic, React Native, NativeScript, Expo та Electron, а також використовувати з TypeScript і JavaScript (ES5, ES6, ES7, ES8). Його мета полягає в тому, щоб завжди підтримувати найновіші функції JavaScript і надавати додаткові функції, які допомагають розробляти будь-які програми, які

використовують бази даних – від невеликих програм з кількома таблицями до великих корпоративних програм з кількома базами даних.

TypeORM підтримує як шаблони Active Record, так і Data Mapper, на відміну від усіх інших ORM JavaScript, які зараз існують, що означає, що ви можете писати високоякісні, слабко пов’язані, масштабовані та підтримувані програми найбільш продуктивним способом.

На TypeORM сильно впливають інші ORM, такі як Hibernate, Doctrine та Entity Framework. [17].

З огляду на все вище описане, було вирішено використати цю ORM при розробці системи.

Nodemailer – це модуль для програм створених на платформі Node.js, який дозволяє легко надсилати електронні листи. Проект розпочався ще в 2010 році, коли не було відповідної можливості надсилати повідомлення електронної пошти, сьогодні це рішення, до якого більшість користувачів Node.js звертаються за замовчуванням.

Особливості:

- велика увага приділяється безпеці;
- підтримка Unicode для використання будь-яких символів, включаючи emoji;
- підтримка HTML;
- можливість прикріплювати файли до повідомлень;
- безпечна доставка електронної пошти за допомогою TLS/STARTTLS;
- різні методи транспортування на додаток до вбудованої підтримки SMTP;
- нормальна автентифікація OAuth2;
- проксі для з'єднань SMTP [18].

З огляду на все вище описане, було вирішено використати саме цей модуль для відправки повідомлень користувачам.

Swagger – це набір правил, специфікацій та інструментів з відкритим кодом для розробки й опису RESTful API. Фреймворк Swagger дозволяє розробникам створювати інтерактивну документацію API (Рисунок 3.2).

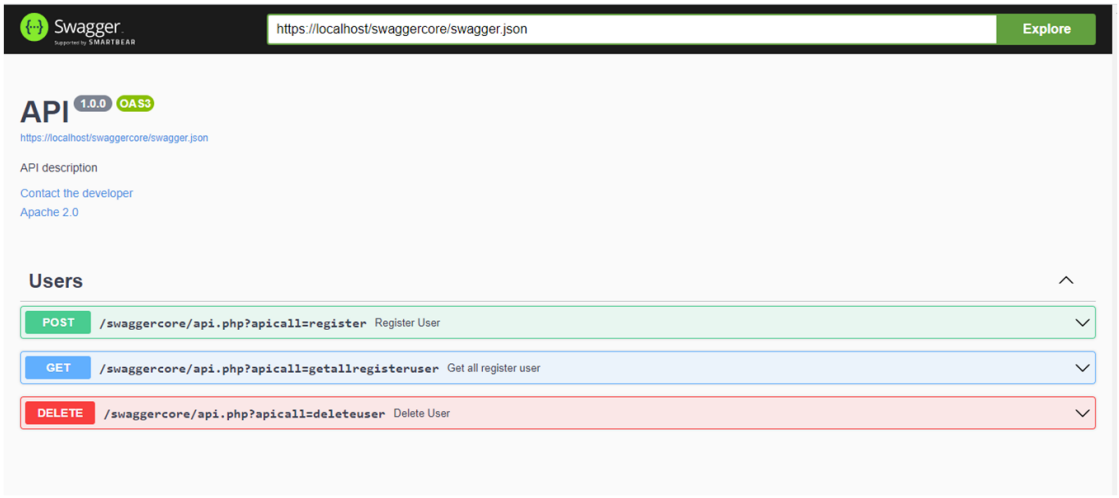

Рисунок 3.2 – Загальний вигляд документації Swagger

Специфікації API зазвичай містять таку інформацію, як підтримувані операції, параметри та результати, вимоги до авторизації, доступні enpoints та необхідні ліцензії. Swagger може генерувати цю інформацію автоматично з вихідного коду, попросивши API повернути файл документації з його анотацій.

Swagger допомагає користувачам створювати, документувати, тестувати та використовувати веб-сервіси RESTful. Його можна використовувати з підходом розробки API як зверху вниз, так і знизу вгору. У методі «зверху вниз» або «спершу проектування» Swagger можна використовувати для розробки API до написання будь-якого коду. У методі «знизу вгору», або методі «спершу код», Swagger бере код, написаний для API, і створює документацію [19].

Зважаючи на опис вище, було вирішено використати цей набір інструментів для полегшення розробки системи.

PostgreSQL – це потужна об’єктно-реляційна база даних з відкритим вихідним кодом, яка використовує та розширює мову SQL у поєднанні з багатьма функціями, які безпечно зберігають і масштабують найскладніші дані.

PostgreSQL здобула міцну репутацію завдяки своїй перевіреній архітектурі, надійності, цілісності даних, надійному набору функцій, розширюваності та відданості спільноти з відкритим кодом, яка стоїть за програмним забезпеченням, щоб постійно надавати ефективні та інноваційні рішення. PostgreSQL працює на всіх основних операційних системах, сумісний

з ACID з 2001 року та має потужні додаткові модулі, такі як популярний розширювач геопросторових баз даних PostGIS. Не дивно, що PostgreSQL стала реляційною базою даних з відкритим вихідним кодом для багатьох людей і організацій.

PostgreSQL має багато функцій, які допомагають розробникам створювати додатки, адміністраторам – захищати цілісність даних і створювати відмовостійке середовище, а також допомагають вам керувати своїми даними, незалежно від того, наскільки великий чи малий набір даних. Крім того, що PostgreSQL є безкоштовною базою даних з відкритим вихідним кодом, він дуже розширюваний. Наприклад, ви можете визначати власні типи даних, створювати власні функції, навіть писати код з різних мов програмування без перекомпіляції бази даних [20].

Зважаючи на всі перелічені можливості PostgreSQL, її було обрано для розробки основної бази даних системи.

pgAdmin – це найдосконаліший інструмент графічного інтерфейсу користувача з відкритим кодом, розроблений для найдосконаліших інструментів керування реляційними базами даних. Іншими словами, pgAdmin і PostgreSQL разом створюють потужну (і безкоштовну) комбінацію для вашого стеку технологій (Рисунок 3.3).

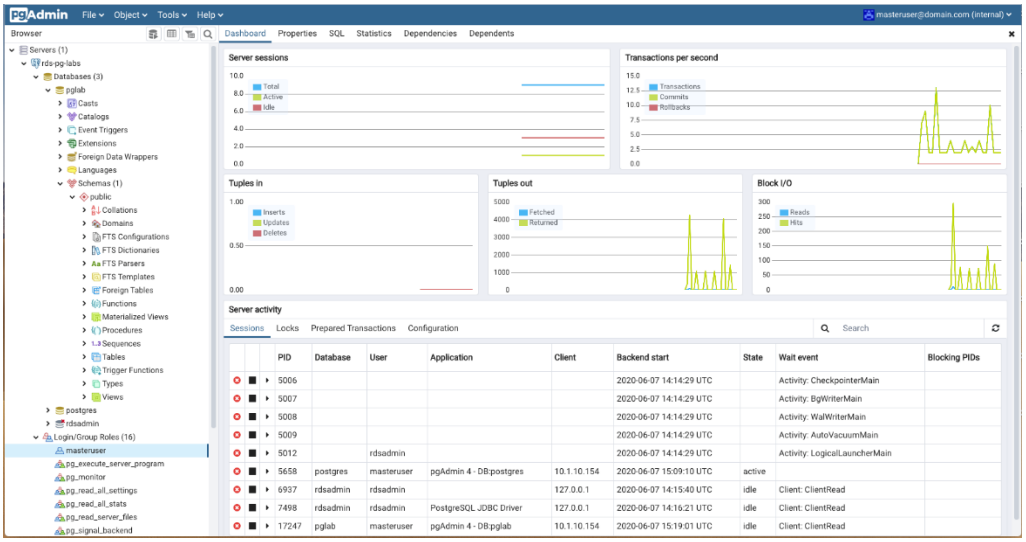

Рисунок 3.3 – Інтерфейс pgAdmin

Основні функції та переваги pgAdmin включають:

- можливість використання в усіх операційних системах, оскільки він сумісний із Windows, Mac і Linux;
- можливість встановити pgAdmin будь-де, де ви використовуєте PostgreSQL;
- сумісність з усіма версіями PostgreSQL і EDB Postgres Advanced Server;
- наявність документації щодо встановлення та використання pgAdmin;
- можливість розгортання в режимі робочого столу або в режимі сервера;
- Наявність інструментів запитів для функцій для швидшого введення даних, налагодження тощо;
- можливість планового технічного обслуговування за допомогою інструментів для резервного копіювання, відновлення, очищення та аналізу;
- можливість переглядати, створювати та редагувати всі стандартні об’єкти в PostgreSQL [21].

Оскільки цей інструмент надає вище перелічені можливості, його було для роботи з PostgreSQL.

Redis – сховище даних, із відкритим вихідним кодом, у пам’яті, яке використовується мільйонами розробників як база даних, кеш-пам’ять, механізм потокового передавання та брокер повідомлень.

Ключові можливості:

- Структури даних в пам'яті. Добре відомий як «сервер структури даних» із підтримкою рядків, хешів, списків, наборів, відсортованих наборів, потоків тощо.
- Програмованість. Створення сценаріїв на стороні сервера за допомогою Lua та збережених процедур на стороні сервера за допомогою функцій Redis.
- Розширюваність. Модуль API для створення спеціальних розширень для Redis на C, C++ і Rust.
- Постійність. Зберігає набір даних у пам’яті для швидкого доступу, але також може зберігати всі записи в постійному сховищі.

– Кластеризація. Горизонтальна масштабованість із сегментуванням на основі хешу, масштабування до мільйонів вузлів із автоматичним повторним розподілом під час розширення кластера.
– Висока доступність. Реплікація з автоматичним відновленням після відмови як для автономних, так і для кластерних розгортань [22].

Оскільки дана технологія задовольняє потребу у кешуванні даних, її було обрано для розробки системи.

Telegraf – це бібліотека, яка спрощує розробку власних телеграм-ботів за допомогою JavaScript або TypeScript.

Особливості:

– повна підтримка Telegram Bot API 6.2;
– чудові типи TypeScript;
– легкий;
– розширюваний [23].

Зважаючи на перераховані особливості, цю бібліотеку було вибрано для розробки телеграм-бота для системи дистанційного навчання.

Git – це безкоштовна розподілена система контролю версій із відкритим вихідним кодом, призначена для швидкої та ефективної обробки будь-яких проектів, від малих до дуже великих.

Git простий у вивченні та має невеликий розмір із блискавичною продуктивністю. Він перевершує такі інструменти SCM, як Subversion, CVS, Perforce і ClearCase, завдяки таким функціям, як дешеве локальне розгалуження, зручні проміжні області та численні робочі процеси [24].

Беручи до уваги все вище сказане, а також те, що Git досить гнучкий та гарантує цілісність даних, його було вибрано в якості розподіленої системи контролю версій для розробки системи.

Visual Studio Code – це редактор вихідного коду, створений Microsoft за допомогою Electron Framework для Windows, Linux і macOS. Функції

включають підтримку налагодження, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти, рефакторинг коду та вбудований Git.

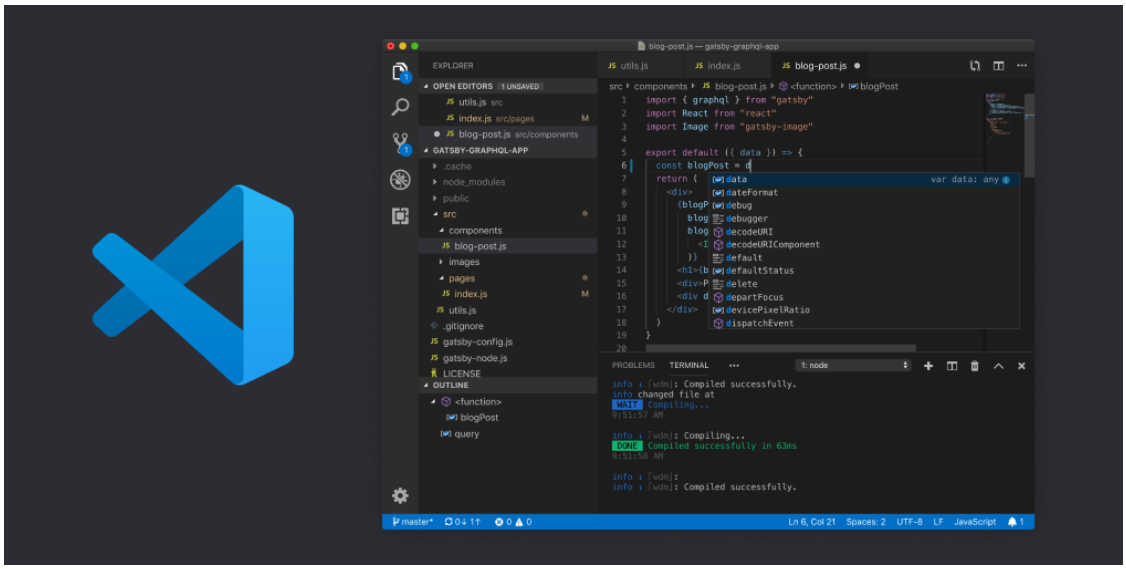

Рисунок 3.4 – Редактор Visual Studio Code

Visual Studio Code містить базову підтримку для більшості поширених мов програмування. Visual Studio Code також постачається з IntelliSense для JavaScript, TypeScript, JSON, CSS і HTML, а також підтримує налагодження Node.js. Підтримку додаткових мов можна отримати за допомогою безкоштовних розширень на VS Code Marketplace [25].

Figma – це веб-додаток для редагування графіки та розробки інтерфейсу користувача. Його можна використовувати для виконання будь-яких видів робіт із графічного дизайну: каркасні веб-сайти, розробка інтерфейсів мобільних додатків, створення прототипів дизайну, створення публікацій в соціальних мережах тощо.

Figma відрізняється від інших інструментів редагування графіки. Головним чином тому, що він працює безпосередньо у браузері. Це означає, що він надає доступ до проектів і дозволяє починати проектування з будь-якого комп’ютера чи платформи без необхідності купувати кілька ліцензій чи інсталювати програмне забезпечення [26].

Дивлячись на опис вище, було вирішено використати саме цей інструмент для створення макету дизайну системи.

Docker – це відкрита платформа для розробки, доставки та запуску програм. Docker дозволяє відокремити програми від інфраструктури, щоб швидко доставляти програмне забезпечення. За допомогою Docker можна керувати своєю інфраструктурою так само, як і своїми програмами. Користуючись методологіями Docker для швидкої доставки, тестування та розгортання коду, можна значно зменшити затримку між написанням коду та початком роботи готового продукту.

Docker надає можливість пакувати та запускати програму в слабко ізольованому середовищі, яке називається контейнером. Ізоляція та безпека дозволяють запускати багато контейнерів одночасно на певному хості. Контейнери легкі та містять усе необхідне для запуску програми, тому не потрібно покладатися на те, що зараз встановлено на хості. Присутня можливість легко ділитися контейнерами під час роботи, і Docker гарантує, що всі контейнери будуть працювати всюди однаково.

Docker надає інструменти та платформу для керування життєвим циклом ваших контейнерів:

- розробка програм та їх допоміжних компонентів за допомогою контейнерів;
- контейнер стає одиницею для розповсюдження та тестування програм;
- контейнер з програмою працює однаково, незалежно від того, чи є виробниче середовище локальним центром обробки даних, хмарним провайдером або їх гібридом [27].

DigitalOcean – американський провайдер хмарного хостингу, який був запущений у Нью-Йорку.

Станом на сьогодні DigitalOcean має центри обробки даних по всьому світу. Він надає розробникам хмарні служби, які допомагають розгортати та масштабувати програми, що працюють одночасно на кількох комп’ютерах.

DigitalOcean є третьою за величиною хостинговою компанією у світі. Багато відгуків DigitalOcean вважають її дешевою, простою та потужною обчислювальною платформою, яку можна використовувати для хостингу, додатків, блогів або веб-сайтів.

Особливості:

- Kubernetes. За допомогою Kubernetes можна легко масштабувати та розгортати контейнерні програми в кластерах. Використовуючи DigitalOcean, ви отримуєте всі основні компоненти Kubernetes безкоштовно.
- Балансувальники навантаження. На DigitalOcean легко ввімкнути балансувальники навантаження. Після ввімкнення ці балансувальники навантаження розподіляють трафік між кількома droplets (віртуальними машинами). Це гарантує, що всі користувачі будуть мати гарний досвід користування.
- Програми в один клік. Завдяки DigitalOcean можна встановлювати деякі програми для droplets в один клік. Ці програми включають: Lamp Stack, WordPress, MySQL, Ruby On Rails, Docker, Node.js тощо.
- Співпраця та надійність. Компанія неймовірно спростила об’єднання команд і співпрацю над створенням програм із підтримкою DigitalOcean. У рамках цього команди працюють разом і отримують рахунки в одному місці, але не мають діляться своїми обліковими даними, що могло б поставити під загрозу безпеку. Однак, для подальшого підвищення надійності, компанія пропонує «плаваючі IP-адреси», за допомогою чого IP-адреси швидко перевизначаються. У будь-якому випадку, коли droplet виходить з ладу; IP-адреса потім повторно призначається резервній droplet, щоб гарантувати, що програма продовжить працювати [28].

З огляду на все вище перелічене, було вирішено обрати саме цю платформу для хостингу системи дистанційного навчання.

3.2. Опис обраного підходу до створення програмного забезпечення

Окрім вище перелічених засобів та інструментів розробки, було вирішено обрати спіральну модель, як підхід до розробки системи дистанційного навчання.

Спіральна модель – це шаблон, що описує послідовність дій, що циклічно повторюються під час розробки програмного забезпечення.

У спіральної моделі є всього 4 основні дії або фази:

1. Аналіз та постановка цілей та вимог. На цьому етапі пророблюються можливі варіанти розвитку проекту та його ризики.

2. Етап розробки. На цьому етапі відбувається сама розробка проекту.

3. Оцінка. Після пройденого етапу розробки відбувається оцінка проведеної роботи, можливих покращень та можливих майбутніх ризиків. Також виділяються загальні перспективи обраного шляху розвитку проекту.

4. Планування наступного циклу розробки. На цьому етапі, на основі отриманих даних, планується наступний цикл розробки, що буде проходити через всі ті самі етапи і так по колу, до самого завершення розробки програмного забезпечення.

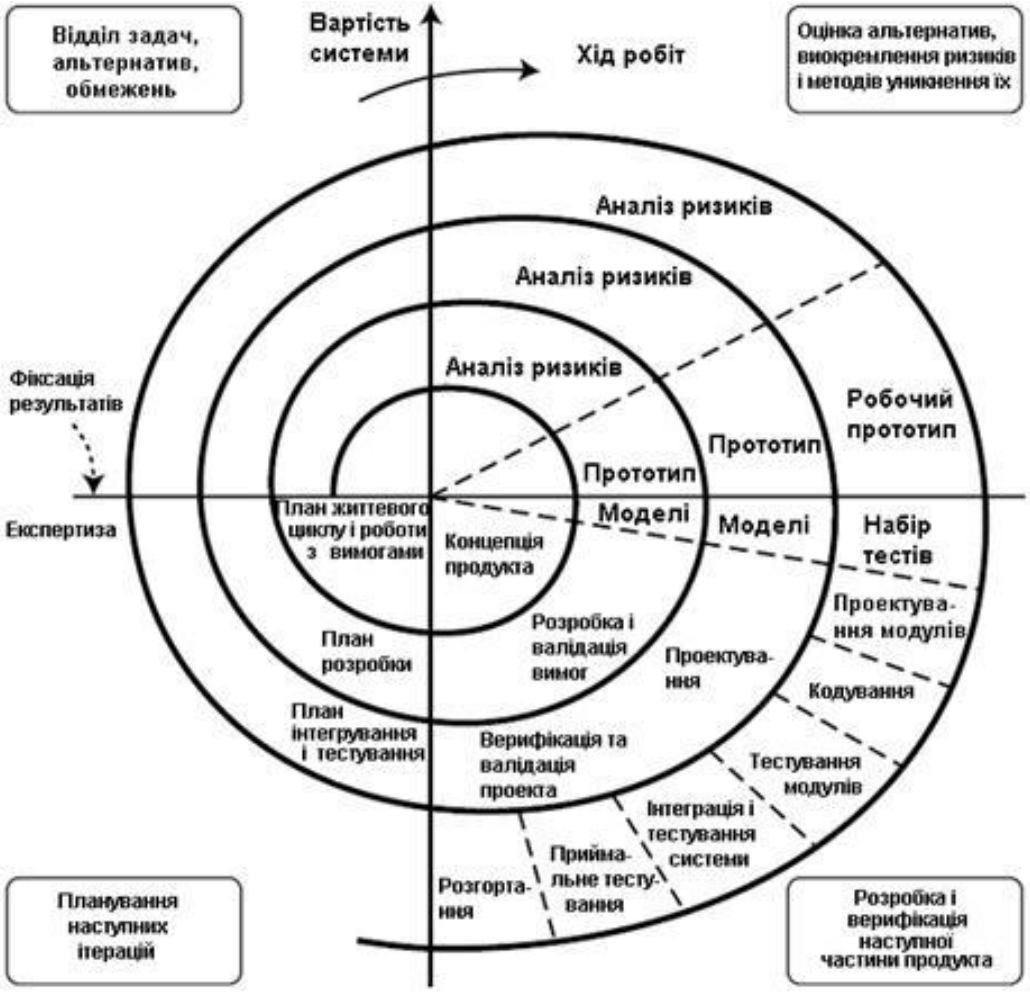

Рисунок 3.5 – Спіральна модель розробки програмного забпезпечння

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис та побудова діаграм роботи частин системи та розгортання

Для опису роботи частин системи, що розробляються, а саме: облікових записів та телеграм-боту, було вирішено побудувати діаграму прецедентів (Рисунок 4.1).

Прецедент – це певна функціональність, яку містить розроблене програмне забезпечення.

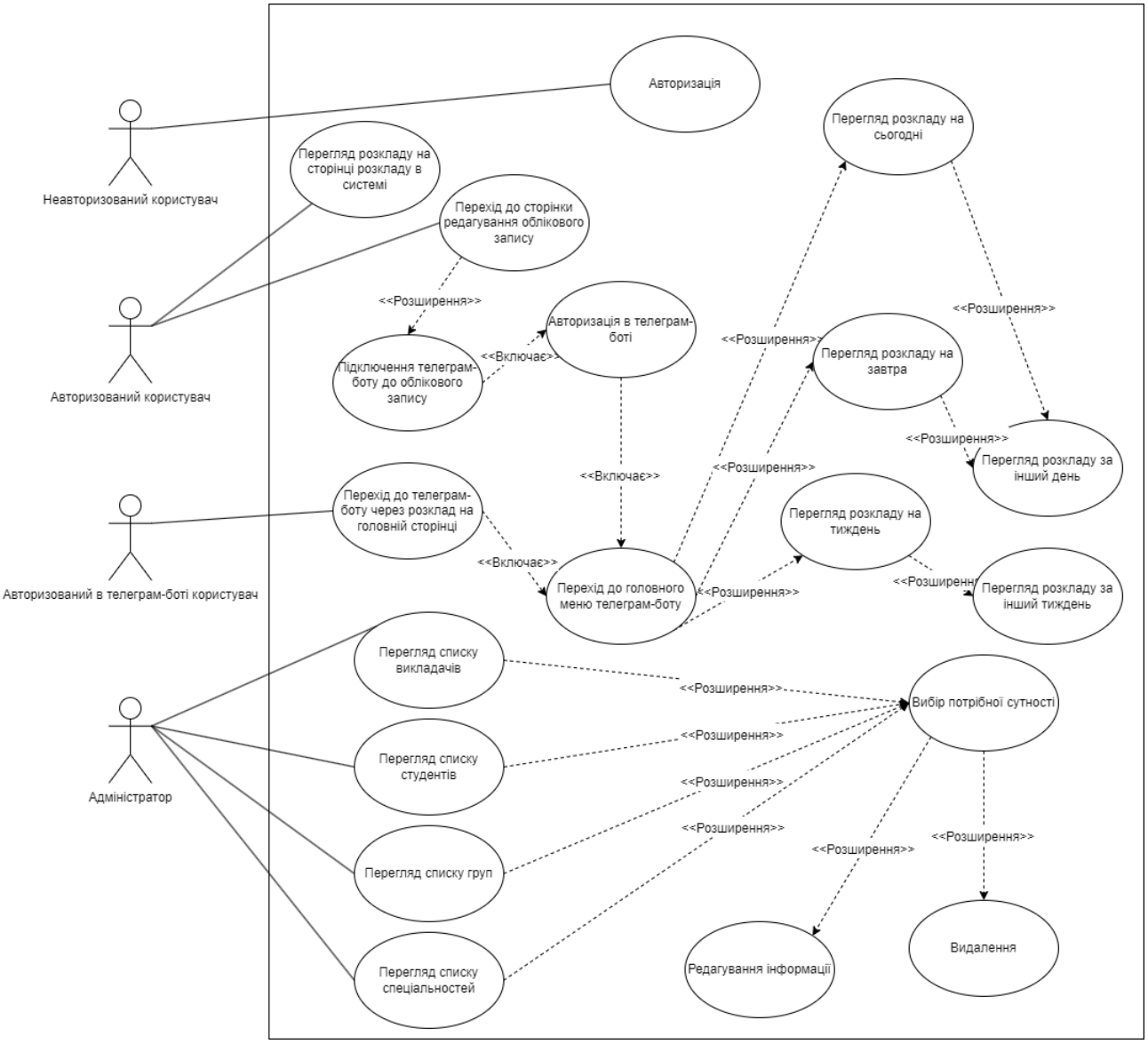

Рисунок 4.1 – Діаграма прецедентів облікових записів та телеграм-бота

ПРЕЦЕДЕНТ: АВТОРИЗАЦІЯ.

Ектор: Неавторизований користувач.

Передумова: Користувач не авторизований в системі.

Післяумова: Користувач авторизований в системі.

Сценарій:

1. Користувач переходить до веб-сайту системи.
2. Натискає на кнопку авторизації.
3. З’являється вікно авторизації.
4. Користувач вводить свій email та пароль.
5. Користувач натискає кнопку «Увійти».
6. Користувач авторизований.

ПРЕЦЕДЕНТ: ПЕРЕГЛЯД РОЗКЛАДУ НА СТОРІНЦІ РОЗКЛАДУ В СИСТЕМІ

Ектор: Авторизований користувач.

Передумова: Користувач авторизований в системі.

Післяумова: Користувач переглянув розклад в системі.

Сценарій:

1. Користувач переходить до сторінки розкладу.
2. Вибирає потрібний місяць.
3. Переглядає розклад.

ПРЕЦЕДЕНТ: АВТОРИЗАЦІЯ В ТЕЛЕГРАМ-БОТІ.

Ектор: Авторизований користувач.

Передумова: Користувач авторизований в системі.

Післяумова: Користувач авторизований в телеграм-боті та перейшов до головного меню телеграм-боту.

Сценарій:

1. Користувач переходить до сторінки редагування власного облікового запису.
2. Натискає на кнопку «Підключити Telegram-розклад».
3. Переходить до сторінки телеграм-бота.

4. Авторизовується в телеграм-боті.
5. Переходить до головного меню телеграм-бота.

Розширення: Перегляд розкладу на завтра, перегляд розкладу на інший день, перегляд розкладу на тиждень, перегляд розкладу на інший тиждень.

ПРЕЦЕДЕНТ: ПЕРЕХІД ДО ТЕЛЕГРАМ-БОТУ ЧЕРЕЗ РОЗКЛАД НА ГОЛОВНІЙ СТОРІНЦІ

Ектор: Авторизований в телеграм-боті користувач.

Передумова: Користувач авторизований в системі і в телеграм-боті.

Післяумова: Користувач перейшов до головного меню телеграм-боту через розклад на головній сторінці.

Сценарій:

1. Користувач переходить до головної сторінки системи.
2. Натискає на кнопку «Telegram-розклад» на карточці з розкладом.
3. Переходить до сторінки телеграм-бота.
4. Переходить до головної сторінки телеграм-бота.

Розширення: Перегляд розкладу на завтра, перегляд розкладу на інший день, перегляд розкладу на тиждень, перегляд розкладу на інший тиждень.

ПРЕЦЕДЕНТ: ПЕРЕГЛЯД СПИСКУ ВИКЛАДАЧІВ СИСТЕМИ.

Ектор: Адміністратор.

Передумова: Адміністратор авторизований в системі.

Післяумова: Адміністратор переглянув список викладачів системи.

Сценарій:

1. Адміністратор натискає на кнопку «Викладачі» на панелі навігації.
2. Адміністратор переходить до сторінки з викладачами системи.
3. Адміністратор переглядає список викладачів системи.

Розширення: Вибір потрібної сутності, редагування, видалення.

ПРЕЦЕДЕНТ: ПЕРЕГЛЯД СПИСКУ СТУДЕНТІВ СИСТЕМИ.

Ектор: Адміністратор.

Передумова: Адміністратор авторизований в системі.

Післяумова: Адміністратор переглянув список студентів системи.

Сценарій:

1. Адміністратор натискає на кнопку «Студенти » на панелі навігації.
2. Адміністратор переходить до сторінки зі студентами системи.
3. Адміністратор переглядає список студентів системи.

Розширення: Вибір потрібної сутності, редагування, видалення.

ПРЕЦЕДЕНТ: ПЕРЕГЛЯД СПИСКУ ГРУП СИСТЕМИ.

Ектор: Адміністратор.

Передумова: Адміністратор авторизований в системі.

Післяумова: Адміністратор переглянув список груп системи.

Сценарій:

1. Адміністратор натискає на кнопку «Групи» на панелі навігації.
2. Адміністратор переходить до сторінки з групами системи.
3. Адміністратор переглядає список груп системи.

Розширення: Вибір потрібного сутності, редагування, видалення.

ПРЕЦЕДЕНТ: ПЕРЕГЛЯД СПИСКУ СПЕЦІАЛЬНОСТЕЙ СИСТЕМИ.

Ектор: Адміністратор.

Передумова: Адміністратор авторизований в системі.

Післяумова: Адміністратор переглянув список спеціальностей системи.

Сценарій:

1. Адміністратор натискає на кнопку «Спеціальності» на панелі навігації.
2. Адміністратор переходить до сторінки зі спеціальностями системи.
3. Адміністратор переглядає список спеціальностей системи.

Розширення: Вибір потрібного сутності, редагування, видалення.

Також було вирішено побудувати діаграму, що описує алгоритм дій, які потрібно виконати для розгортання системи (Рисунок 4.2).

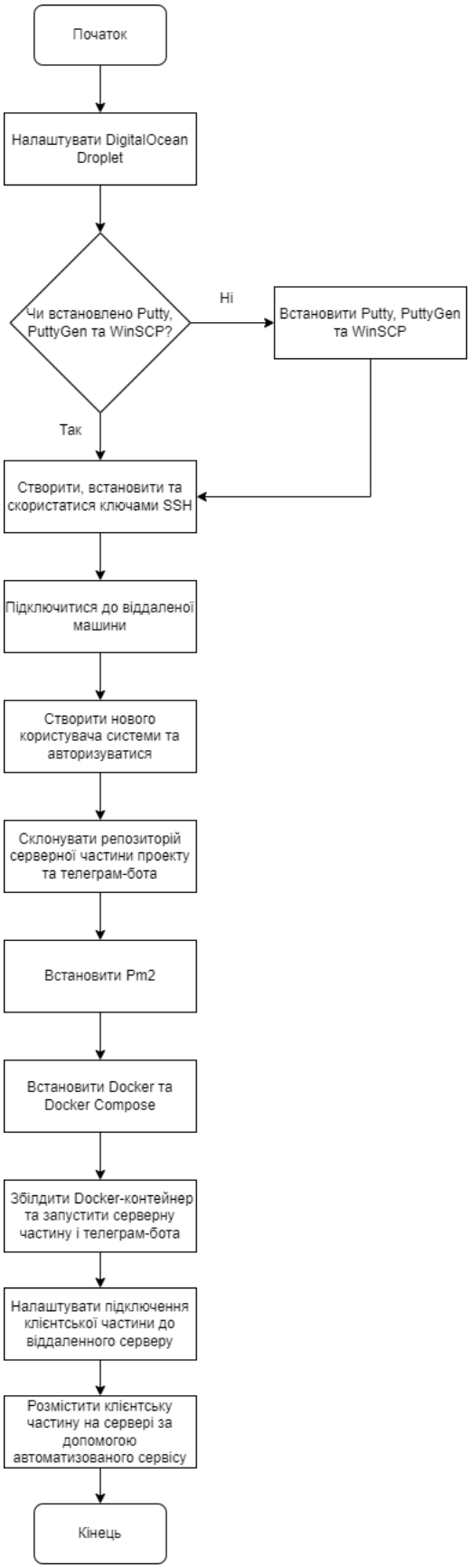

Рисунок 4.2 – Діаграма алгоритму дій при розгортанні системи

4.2. Інструкція з використання розроблених частин системи

В результаті проведеної роботи було реалізовано панель адміністратора, за допомогою якої можна керувати обліковими записами користувачів, групами та спеціальностями, а також було реалізовано повноцінний сервіс телеграм-боту для перегляду розкладу та серверну частину для його комунікації з системою дистанційного навчання.

Облікові записи. Розроблену функціональність системи дистанційного навчання у вигляді облікових записів, можна продемонструвати наступним чином:

1. Користувач, переходить до модального вікна авторизації в системі (Рисунок 4.3).

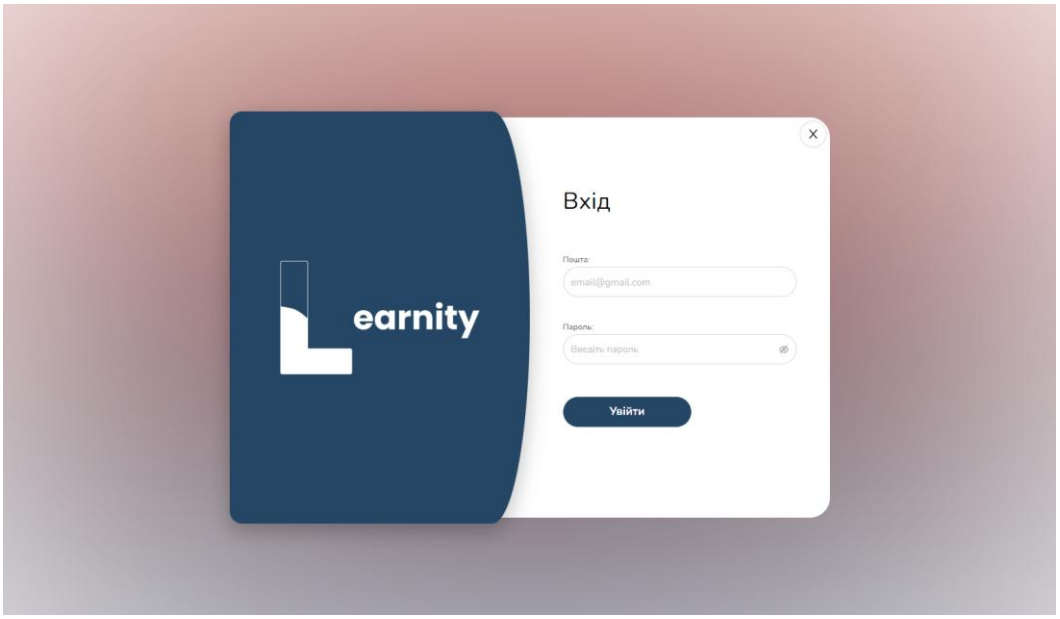

Рисунок 4.3 – Модальне вікно авторизації

2. Якщо користувач вводить некоректні дані, то форма вводу, за допомогою клієнтської валідації, виведе повідомлення про помилки під тими полями для вводу, де вони є (Рисунок 4.4). А якщо помилка виникне при серверній валідації, то у правому верхньому кутку з’явиться повідомлення з текстом помилки, що виникла (Рисунок 4.5).

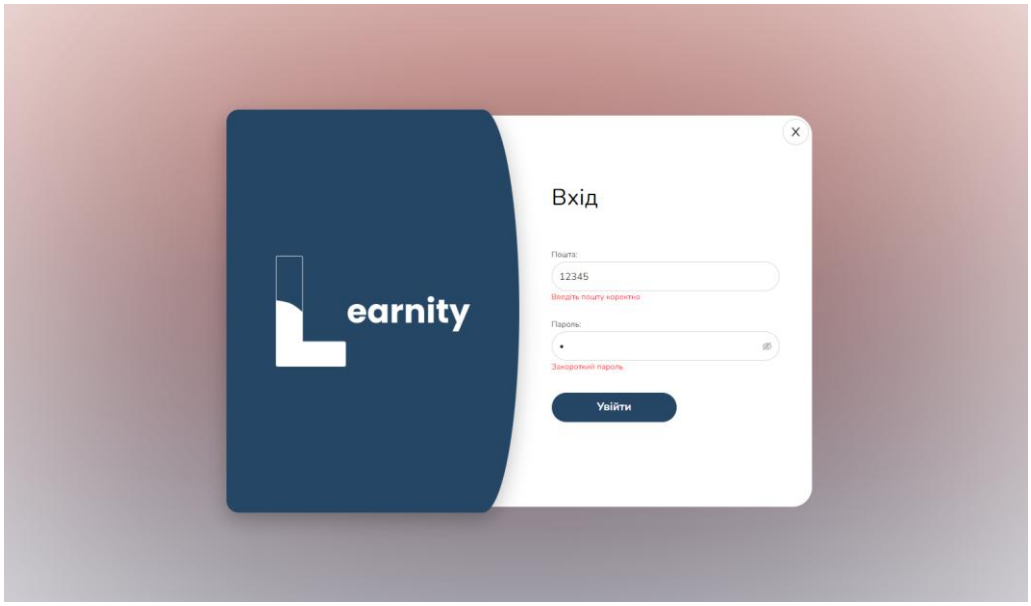

Рисунок 4.4 – Помилки клієнтської валідації

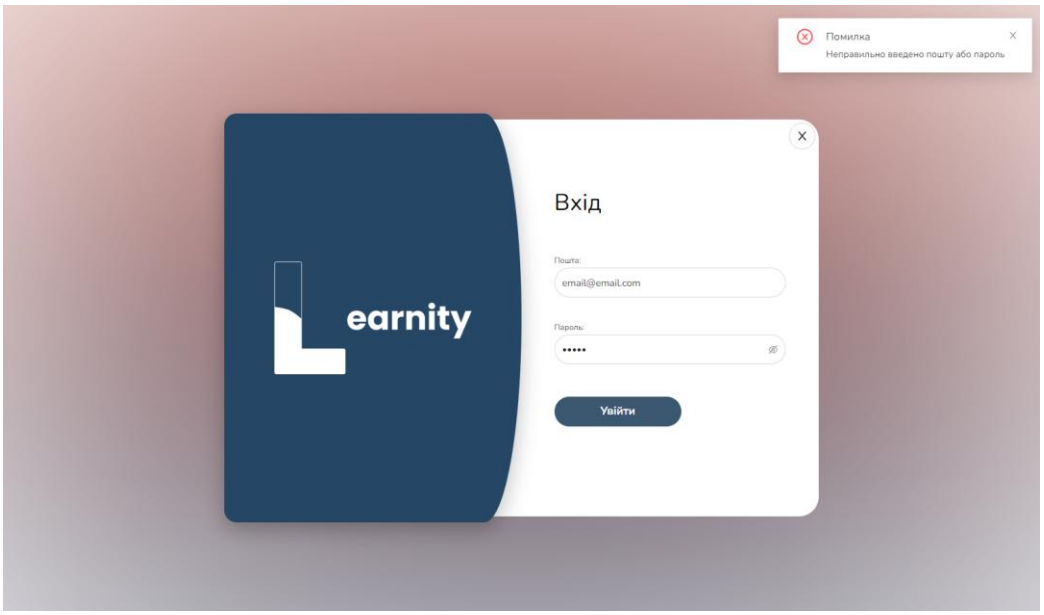

Рисунок 4.5 – Помилка серверної валідації

3. Далі користувач авторизовується в обліковий запис адміністратора і потрапляє до його панелі, де буде відкрита сторінка з таблицею, що містить інформацію про всіх викладачів системи (Рисунок 4.6).

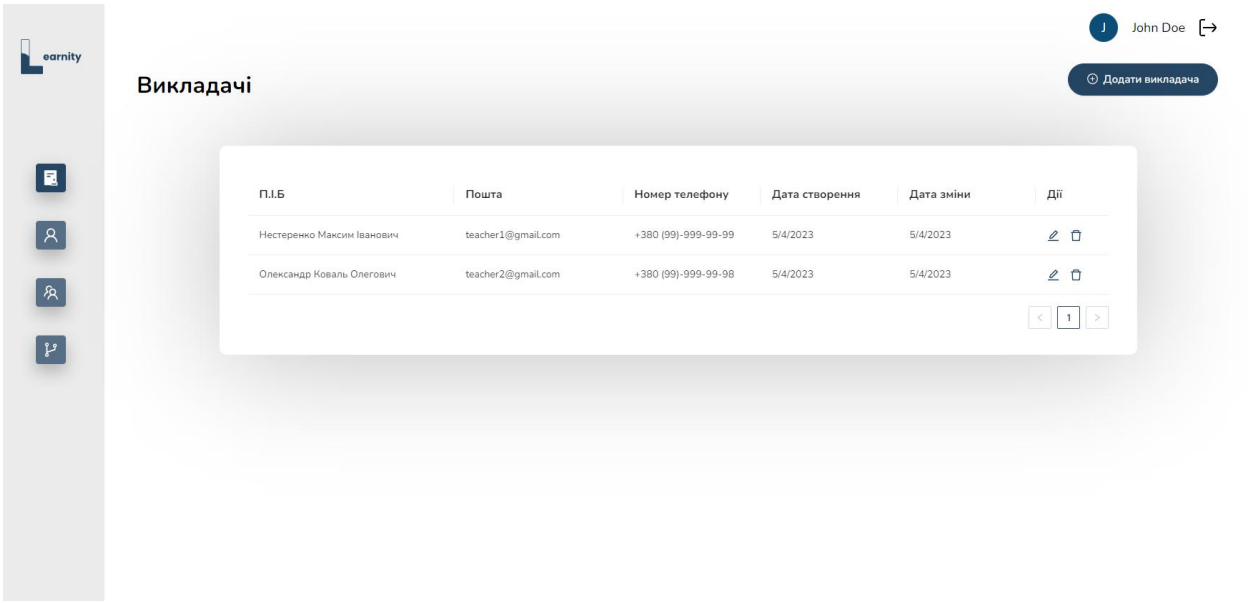

Рисунок 4.6 – Сторінка з всіма викладачами системи

4. На цій сторінці адміністратор має змогу додавати, редагувати та видаляти викладачів. Для того, щоб додати нового викладача адміністратору потрібно натиснути на кнопку «Додати викладача», після чого буде відкрите модальне вікно створення облікового запису для користувача з роллю викладача (Рисунок 4.7). Дане модальне вікно також має валідацію (Рисунок 4.8). Після успішного створення облікового запису викладача буде виведено повідомлення про успіх (Рисунок 4.9) та буде відправлено, на введену пошту, листа з паролем для входу в цей обліковий запис (Рисунок 4.10).

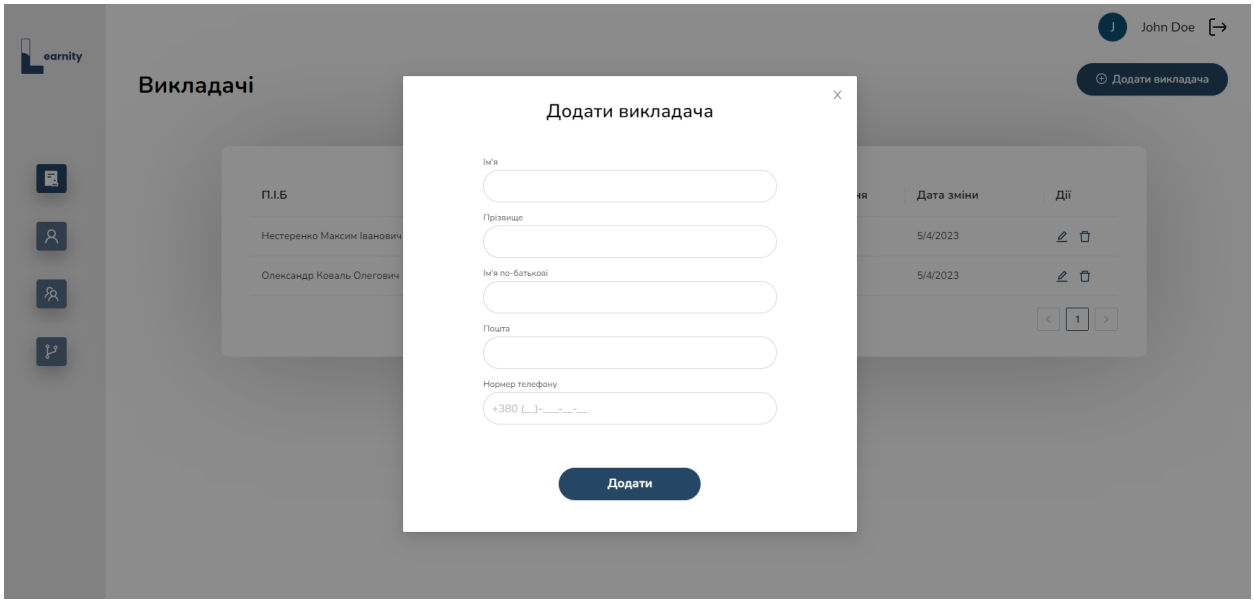

Рисунок 4.7 – Модальне вікно створення облікового запису викладача

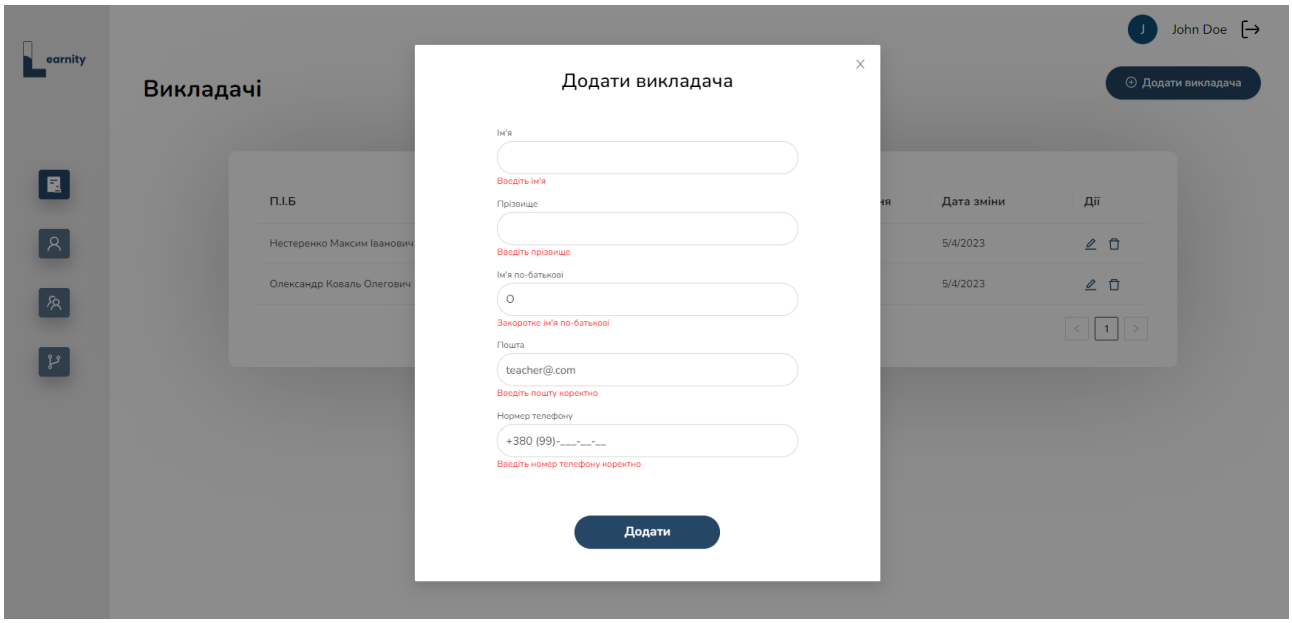

Рисунок 4.8 – Помилки клієнтської валідації

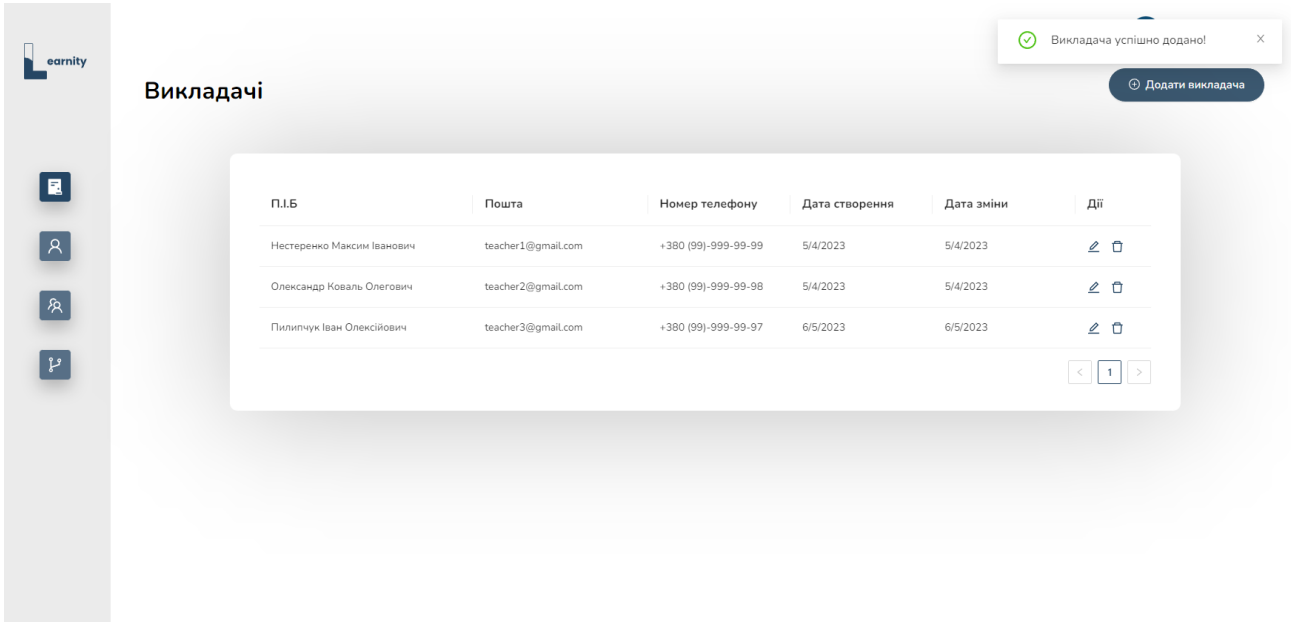

Рисунок 4.9 – Повідомлення про успішне створення облікового запису викладача

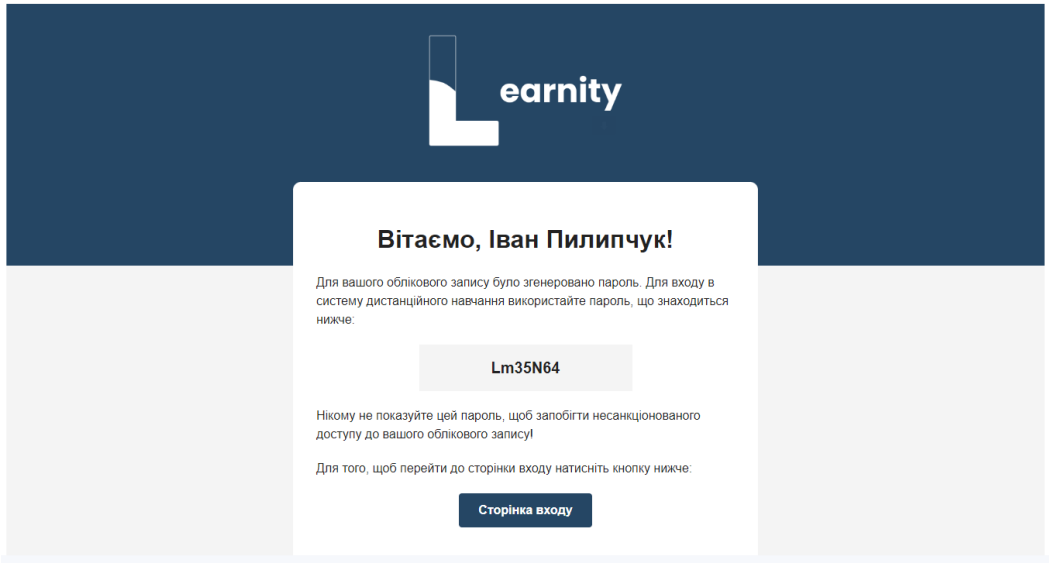

Рисунок 4.10 – Повідомлення з паролем для входу в обліковий запис

5. Для редагування та видалення адміністратору потрібно натиснути відповідні кнопки у вигляді олівця та сміттєвого кошика у таблиці навпроти потрібного викладача. При натисканні редагування з’явиться модальне вікно подібне до модального вікна створення, але воно буде використовувати введені дані редагування, а не створення облікового запису викладача (Рисунок 4.11). Також адміністратор може згенерувати новий пароль для облікового запису викладача натиснувши відповідну кнопку в модальному вікні редагування. В свою чергу при натисканні на видалення з’явиться вікно підтвердження (Рисунок 4.12). Всі дії супроводжуються серверними повідомленнями.

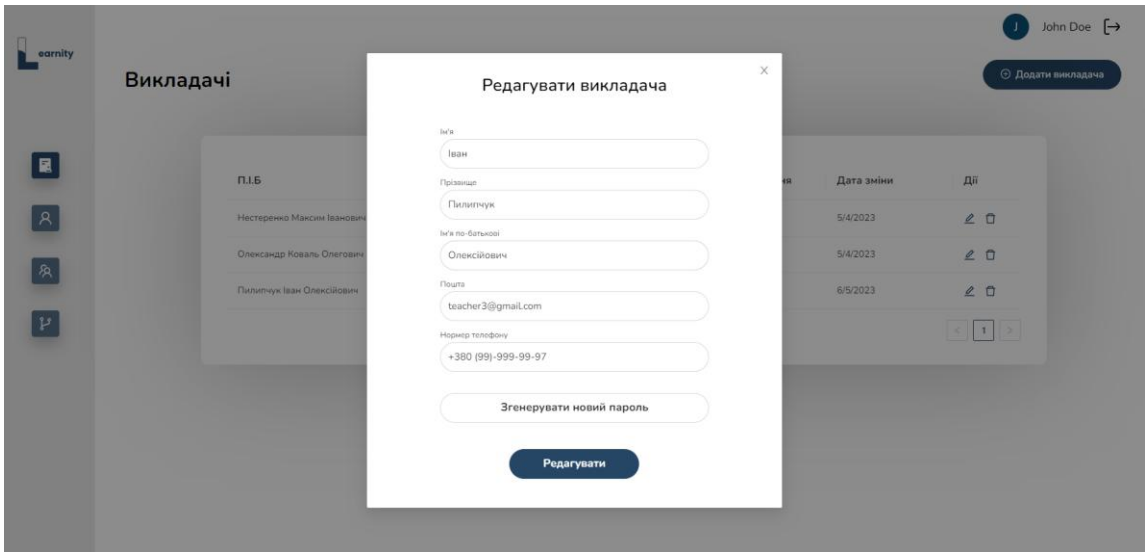

Рисунок 4.11 – Модальне вікно редагування облікового запису викладача

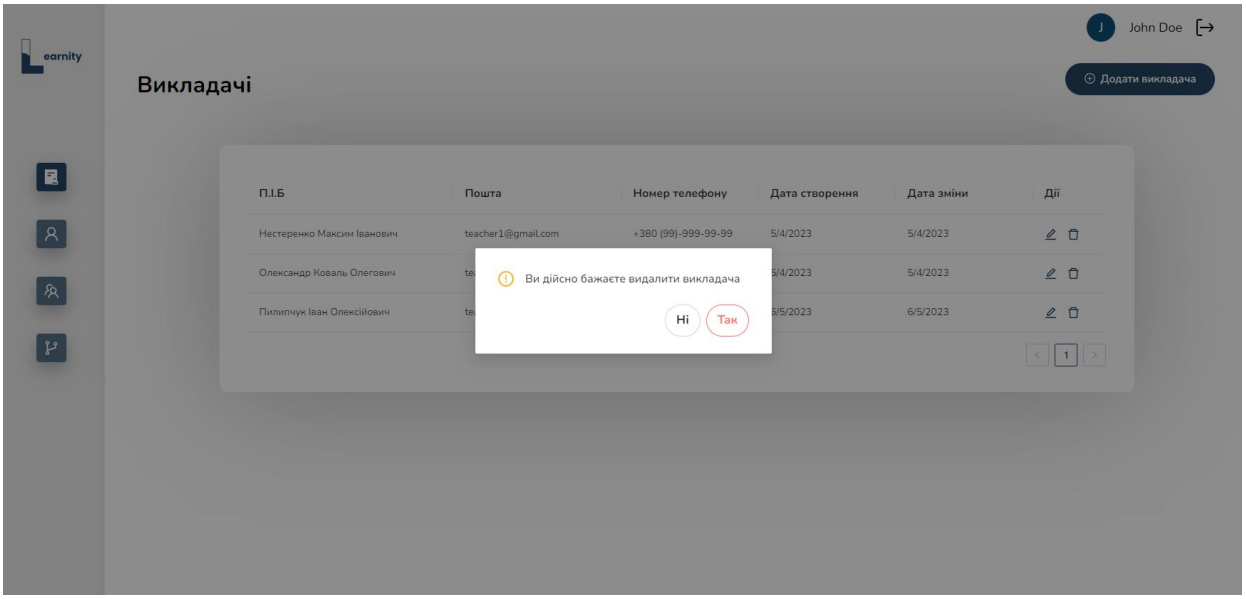

Рисунок 4.12 – Модальне вікно підтвердження видалення викладача

6. Після цього, адміністратор може додати декілька облікових записів студентів, перейшовши на відповідну сторінку (Рисунок 4.13), яка містить функціональність подібну до сторінки з викладачами, та натиснувши на кнопку «Додати студента» і заповнивши модальне вікно створення облікового запису студента (Рисунок 4.14).

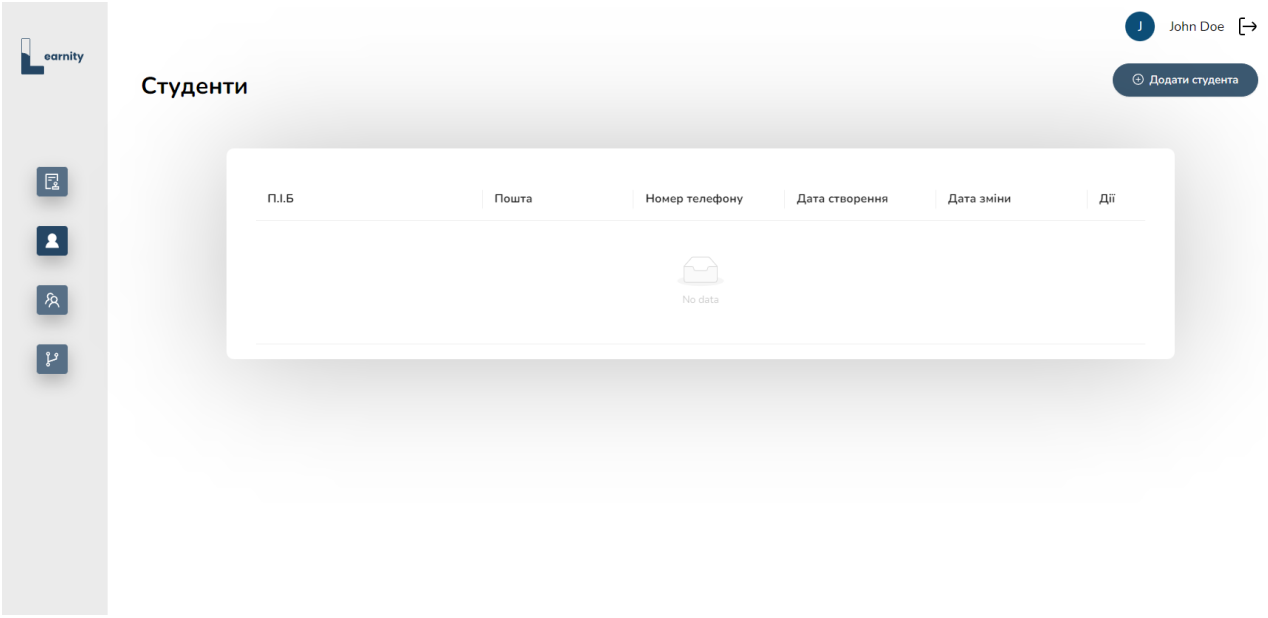

Рисунок 4.13 – Сторінка з обліковими записами студентів

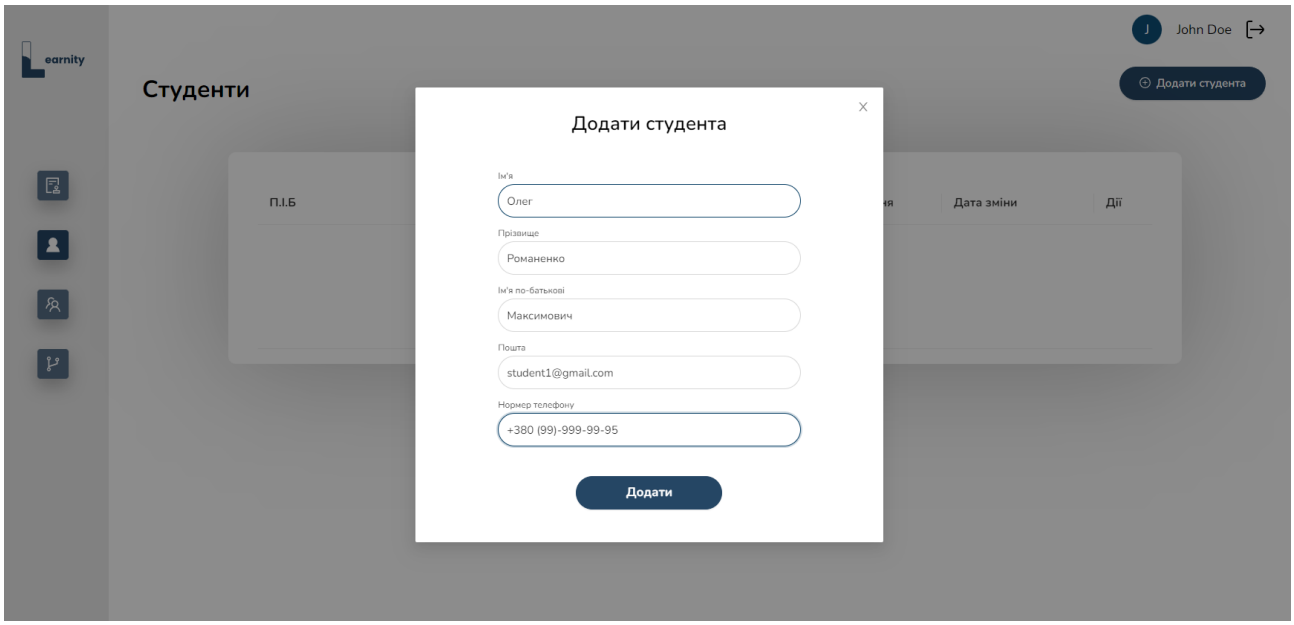

Рисунок 4.14 – Модальне вікно створення облікового запису студента

Після створення, облікові записи студентів з’являться у відповідній таблиці сторінки (Рисунок 4.15). Облікові записи студентів можна також редагувати та видаляти.

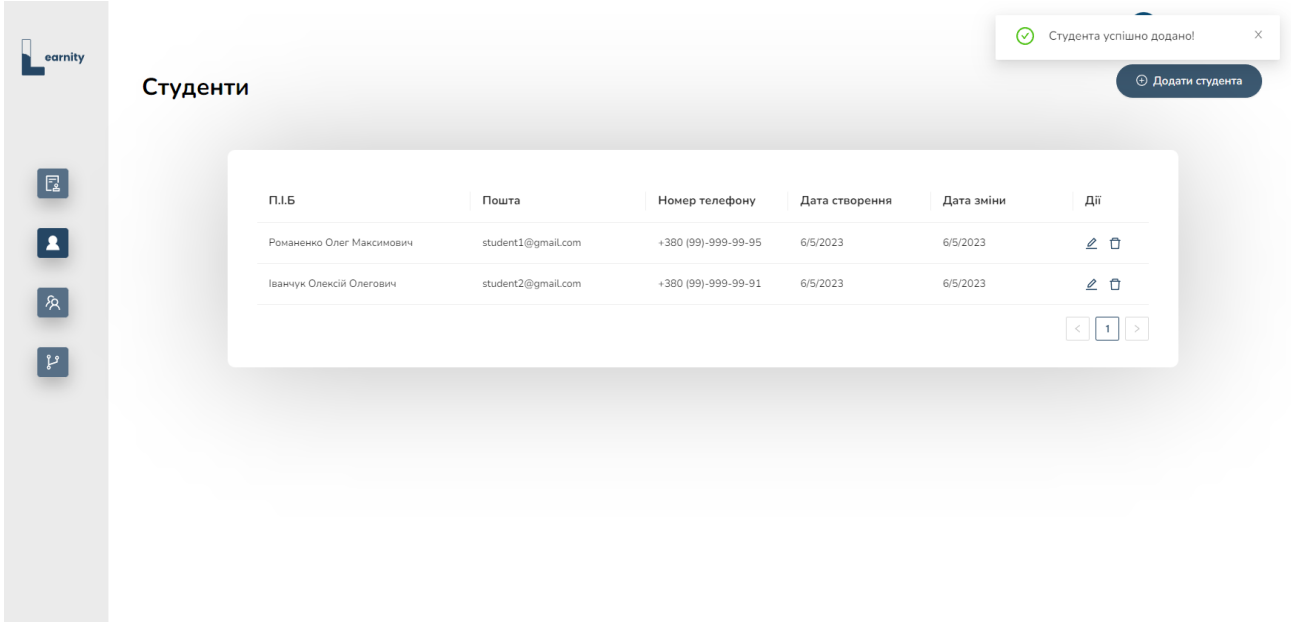

Рисунок 4.15 – Створені облікові записи студентів

7. Далі адміністратор може перейти до сторінки зі спеціальностями (Рисунок 4.16) та створити їх декілька, натиснувши на кнопку «Додати

спеціальність», заповнивши модальне вікно створення спеціальності (Рисунок 4.17) та натиснути «Додати».

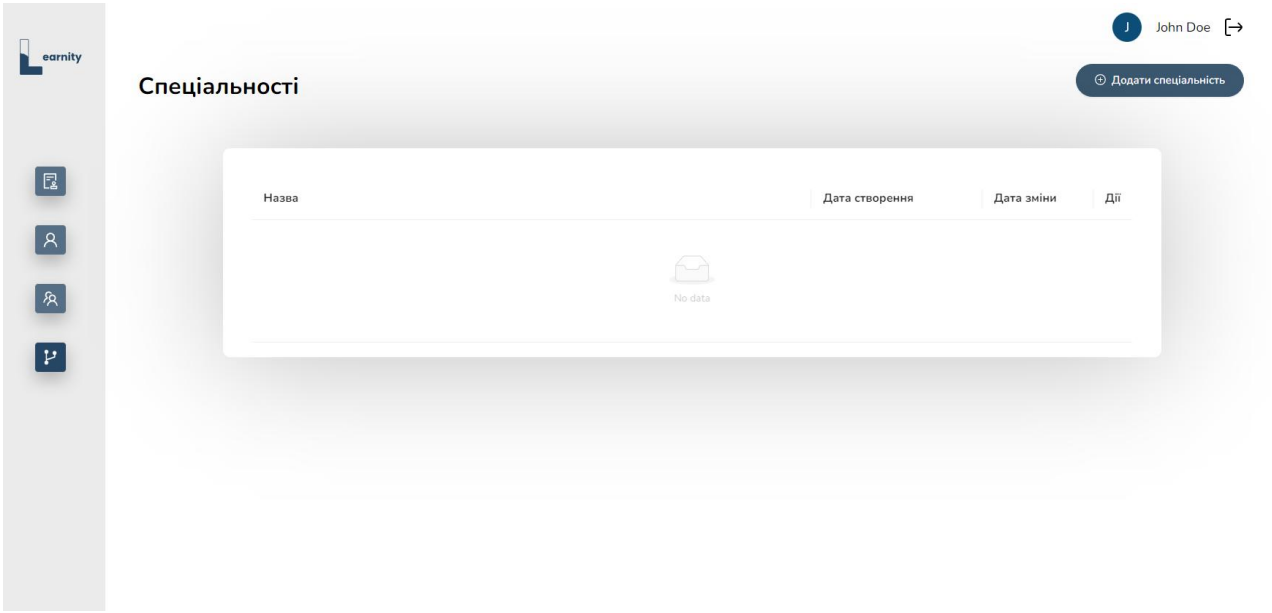

Рисунок 4.16 - Сторінка спеціальностей системи

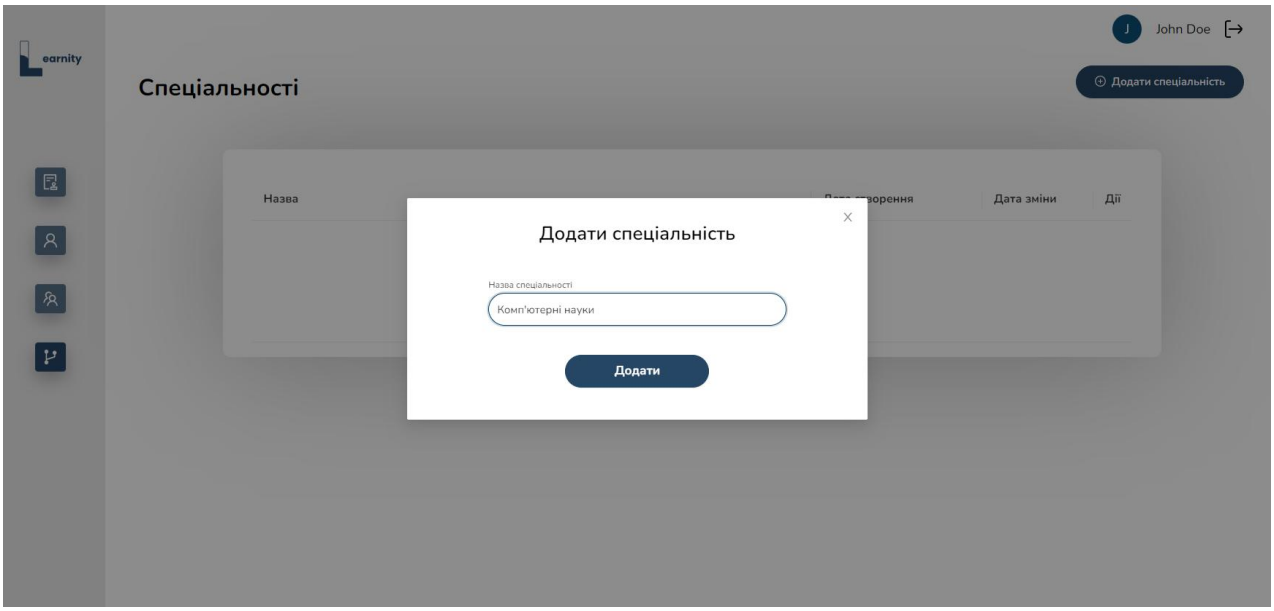

Рисунок 4.17 – Модальне вікно створення спеціальності

В результаті буде створено та відображено декілька спеціальностей у відповідній таблиці (Рисунок 4.18). Створені спеціальності також можна редагувати та видаляти.

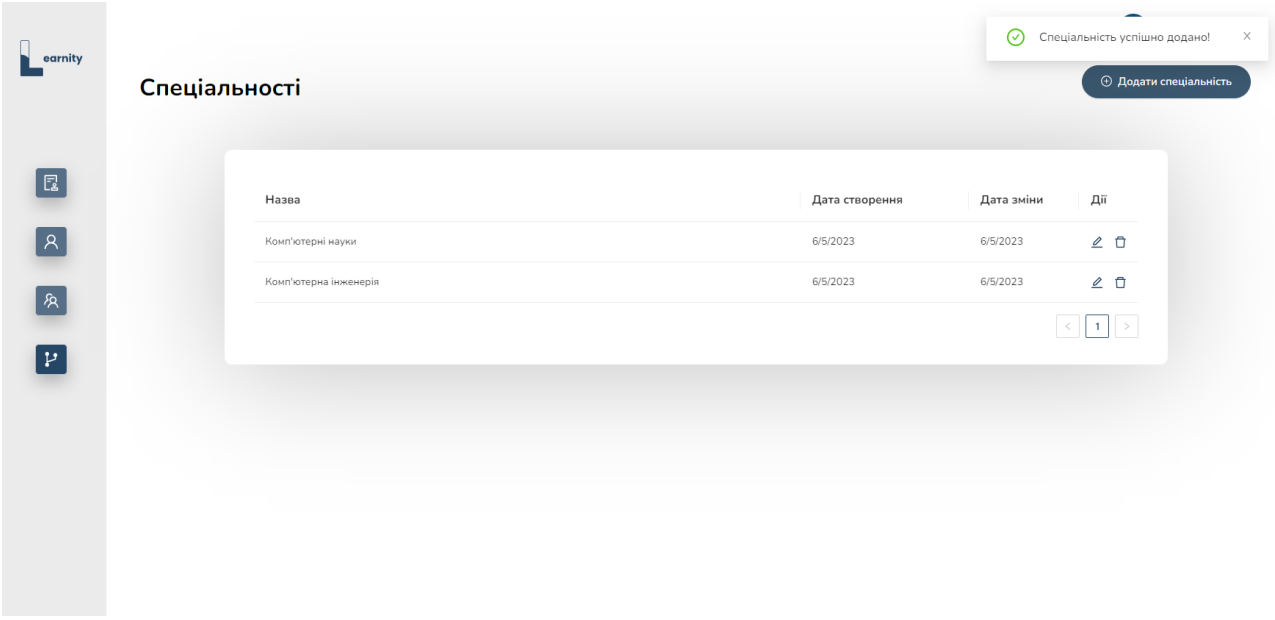

Рисунок 4.18 – Створені спеціальності

8. Тепер адміністратор може додати декілька груп до щойно створених спеціальностей. Для цього йому потрібно перейти до сторінки з групами та, відповідно до попередніх сторінок, натиснути кнопку «Додати спеціальність», заповнити модальне вікно створення (Рисунок 4.19) та натиснути «Додати». Після чого, всі створені групи з’являться у відповідній таблиці з групами (Рисунок 4.20).

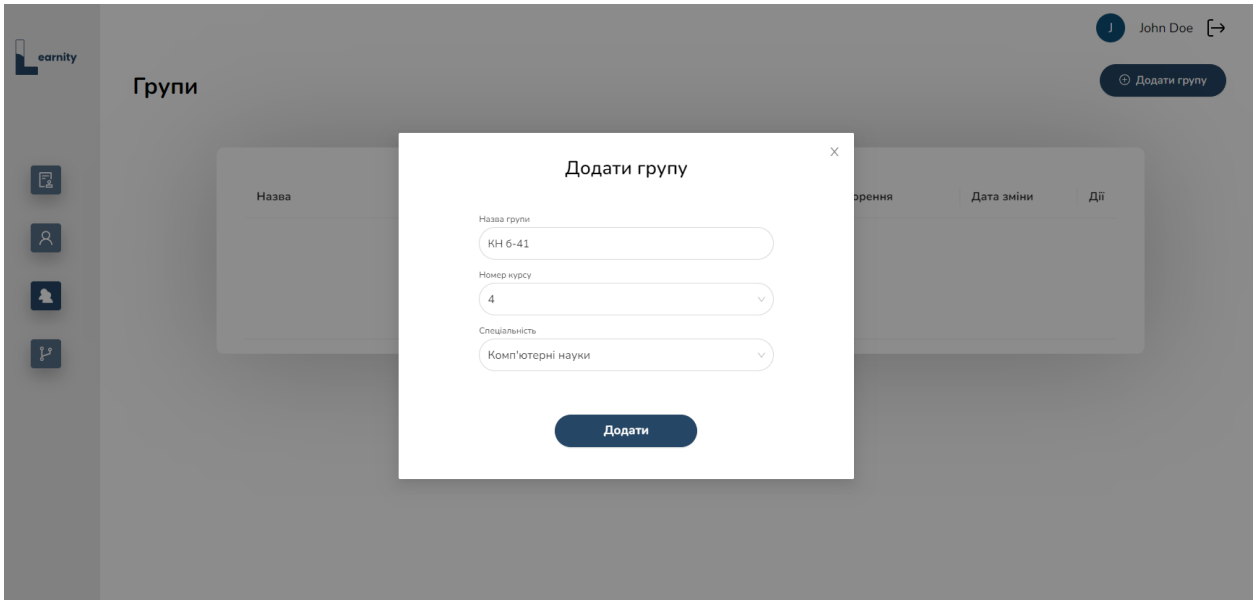

Рисунок 4.19 – Модальне вікно створення групи

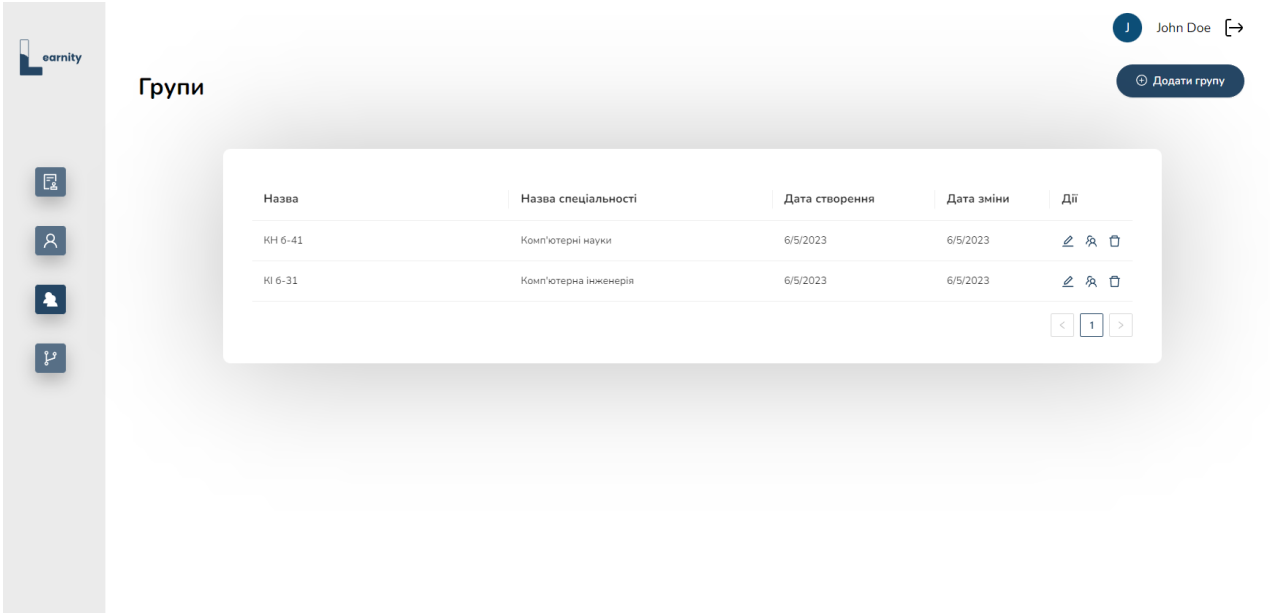

Рисунок 4.20 – Створені групи

9. Після цього, адміністратор може додавати студентів до створених груп натиснувши на кнопку у вигляді пари людей в таблиці навпроти потрібної групи, після чого буде відкрито модальне вікно з таблицею учасників цієї групи (Рисунок 4.21). Далі адміністратору потрібно знайти, за допомогою пошуку, потрібно студента та додати його до групи (Рисунок 4.22). Також студента можна видалити з групи (Рисунок 4.23).

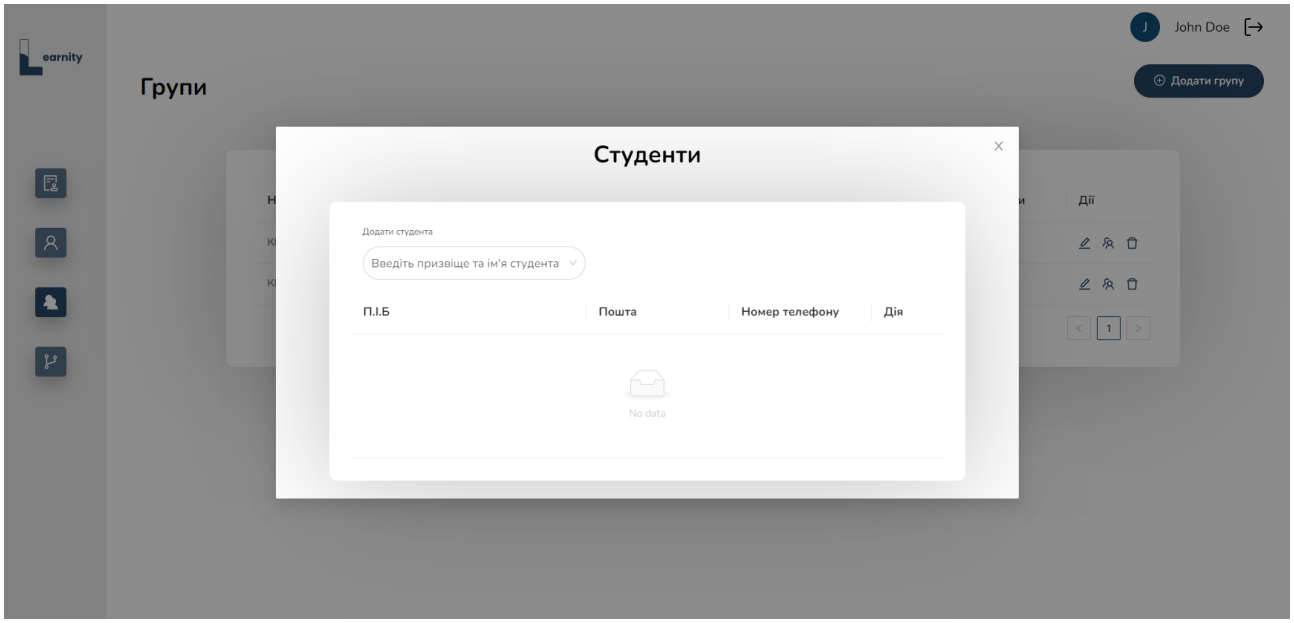

Рисунок 4.21 – Модальне вікно з таблицею учасників групи

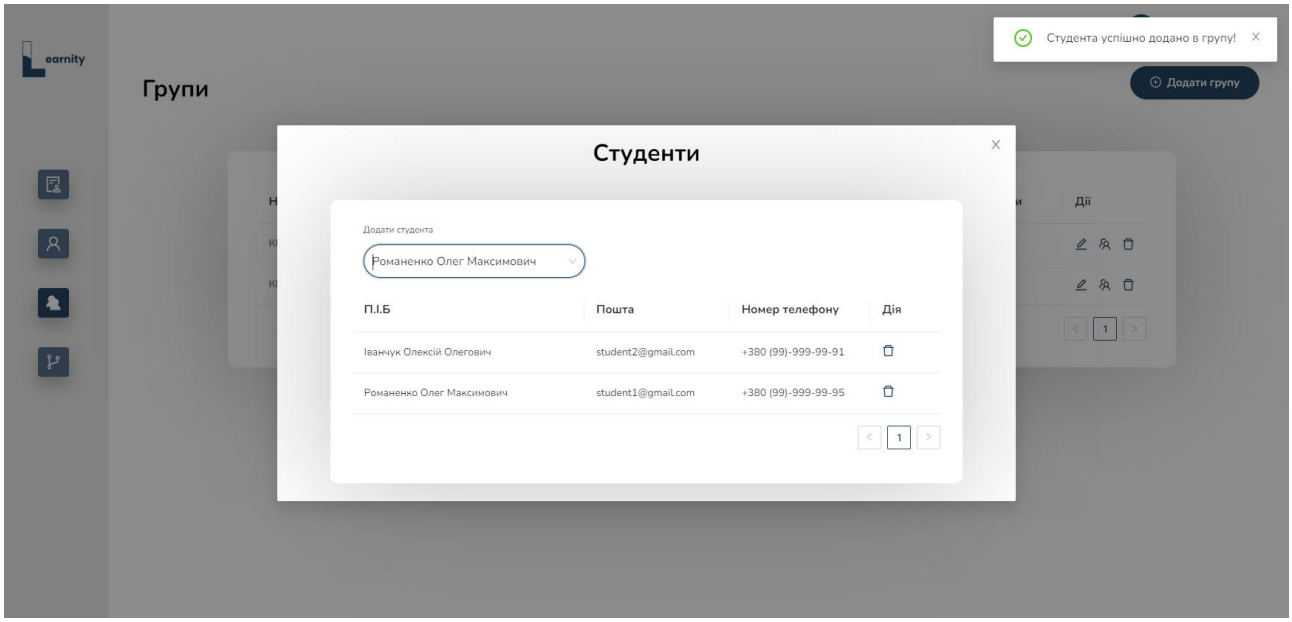

Рисунок 4.22 – Додані до групи студенти

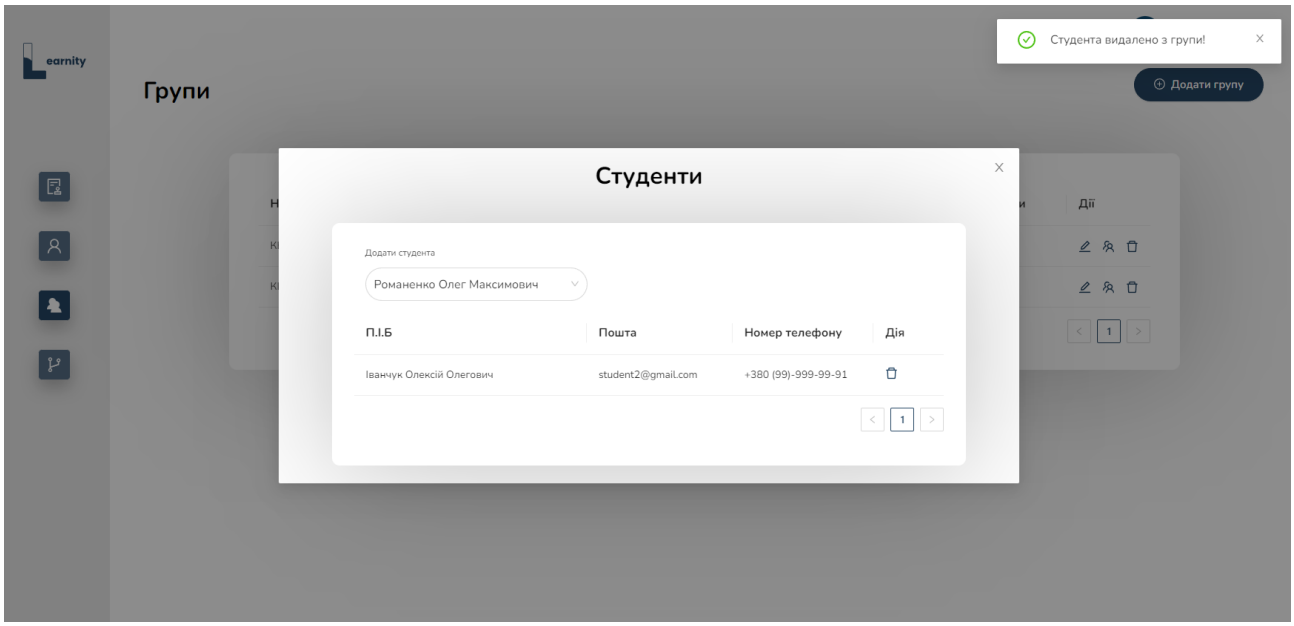

Рисунок 4.23 – Успішне видалення студента

Телеграм-бот. Процес роботи з телеграм-ботом можна описати наступним чином:

1. Користувач заходить в систему дистанційного навчання, переходить до сторінки налаштування власного облікового запису та натискає на кнопку «Підключити Telegram-розклад» (Рисунок 4.24).

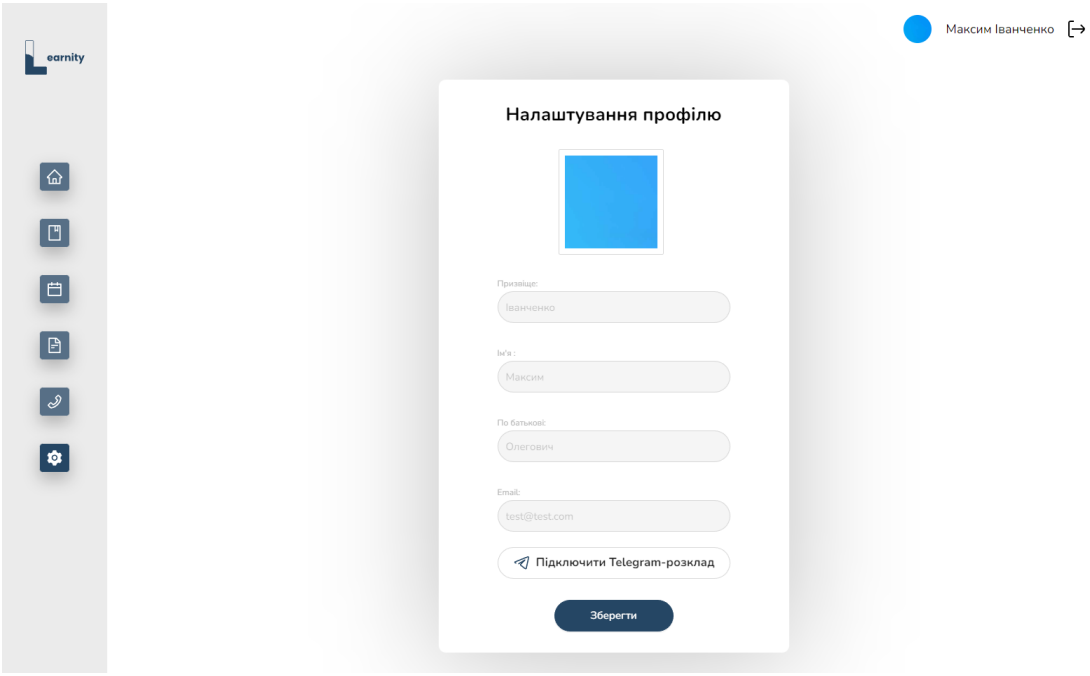

Рисунок 4.24 – Сторінка налаштування облікового запису

2. Після натискання кнопки на сервері створюється телеграм-токен і користувача перенаправляє до сторінки з телеграм-ботом, де в посиланні буде вказано цей токен (Рисунок 4.25).

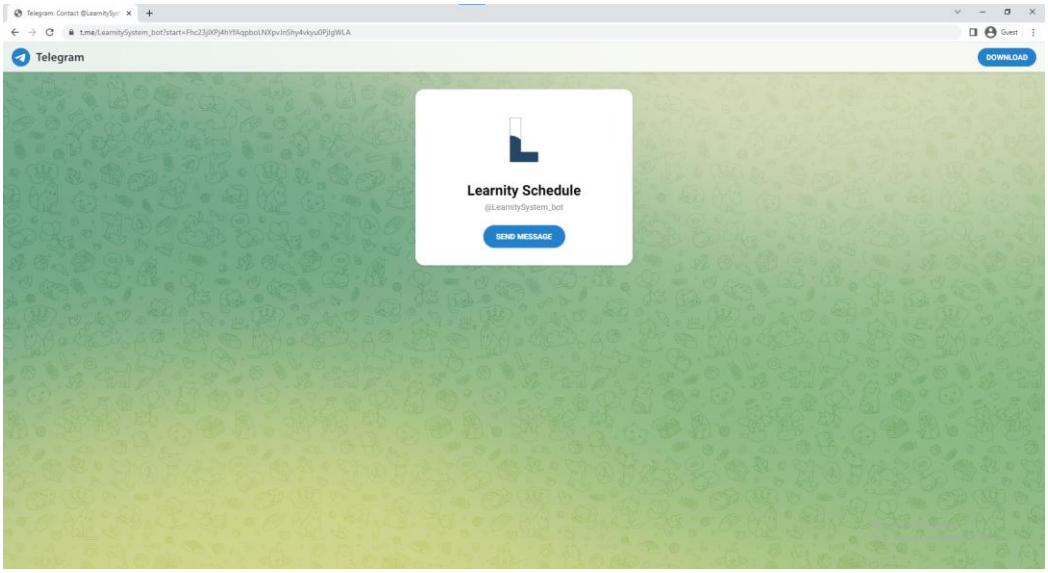

Рисунок 4.25 – Сторінка телеграм-боту з токеном у посиланні

3. Далі користувач переходить до чату з телеграм-ботом, де після початку листування буде здійснено спробу авторизації за допомогою серверу системи

дистанційного навчання і, у випадку успіху, бот відправить повідомлення з привітанням та головним меню (Рисунок 4.26). У випадку помилки під час авторизації буде відправлено відповідне повідомлення з помилкою (Рисунок 4.27). Після успішної авторизації сервер зберігає телеграм-ідентифікатор користувача в базі даних і в подальшому звіряє його з тим, який приходить від телеграм-боту, щоб впевнитися, що бот використовується авторизованим користувачем. Також у користувача з'явиться можливість швидкого переходу до чату з телеграм-ботом з головної сторінки системи за допомогою кнопки «Telegram-розклад» (Рисунок 4.28).

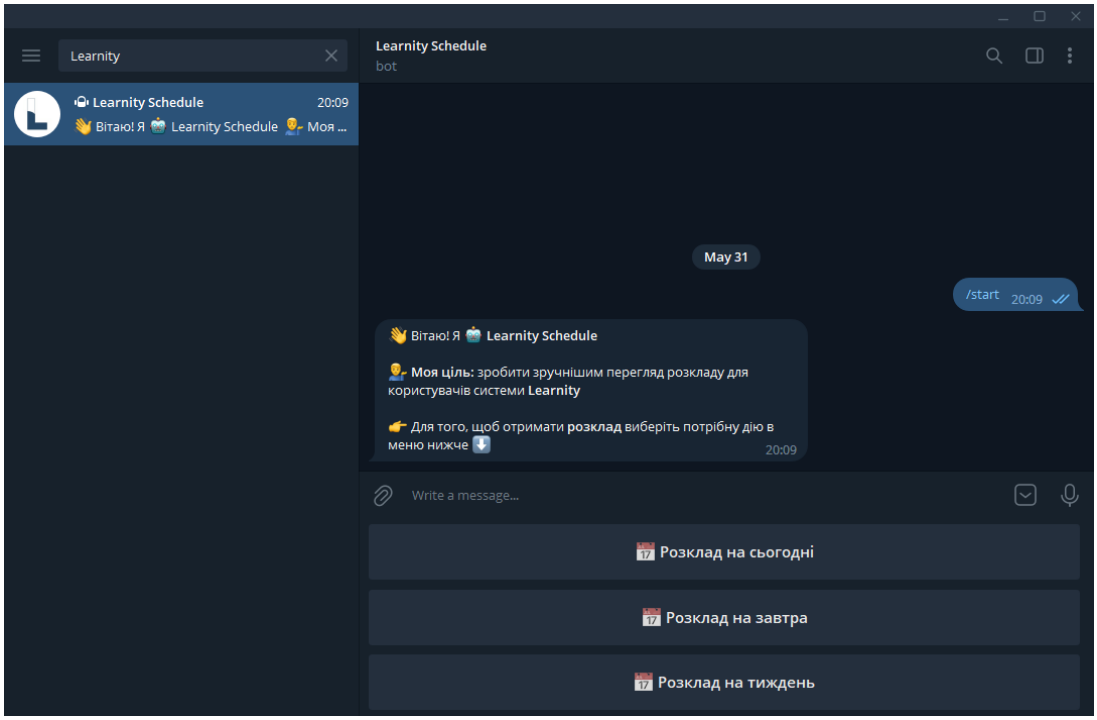

Рисунок 4.26 – Повідомлення з привітанням

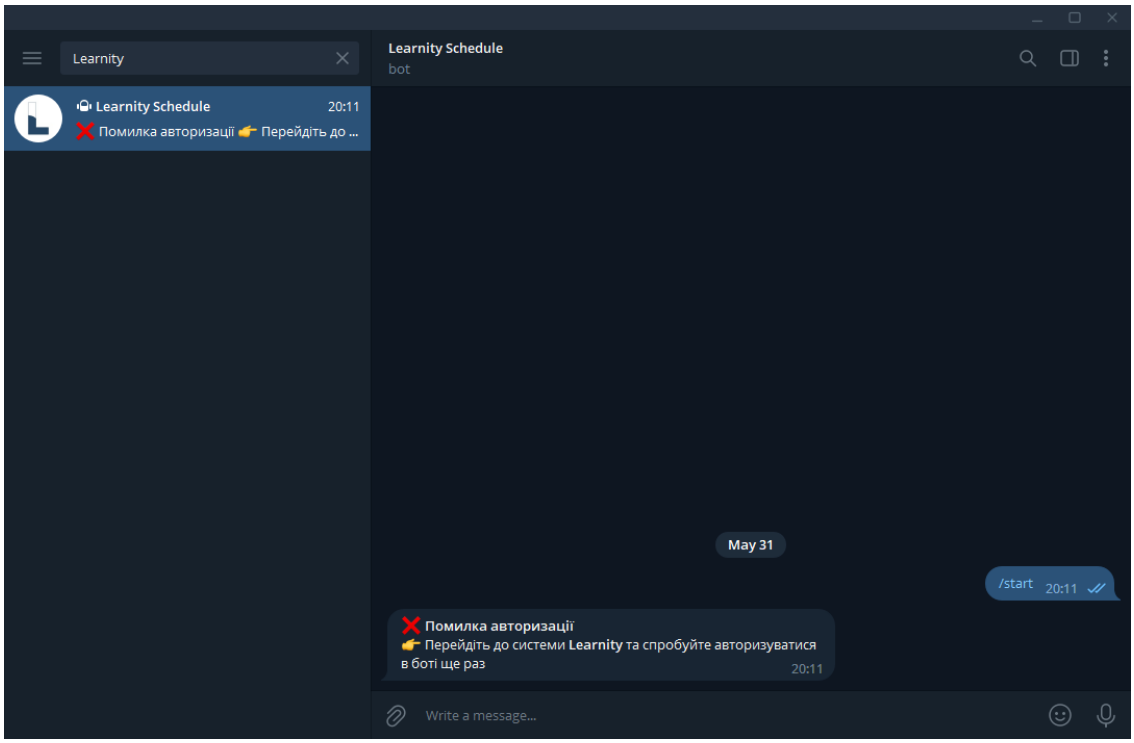

Рисунок 4.27 – Повідомлення про помилку авторизації

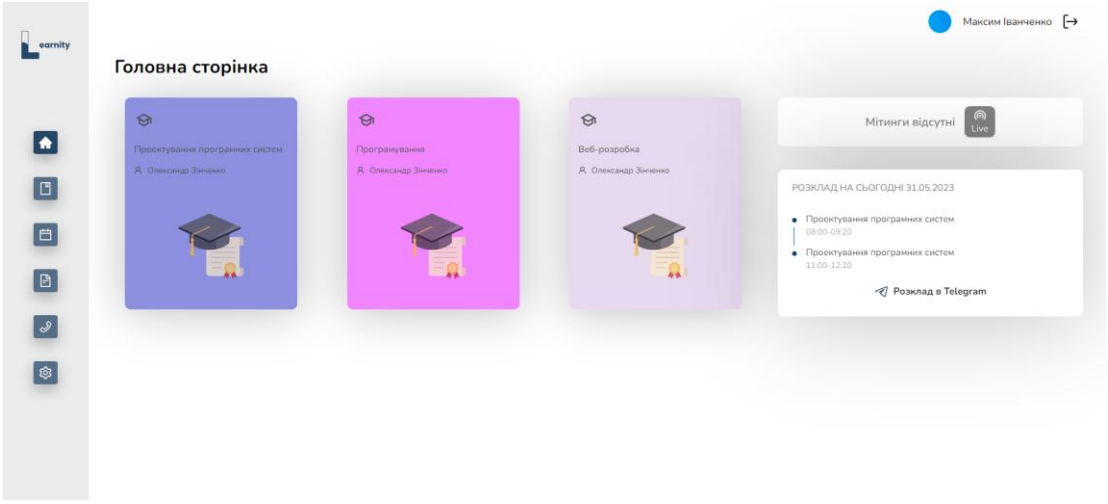

Рисунок 4.28 – Головна сторінка системи дистанційного навчання

4. Тепер користувач має змогу повноцінно використовувати телеграм-бота для перегляду свого розкладу занять системи дистанційного навчання, для цього потрібно натискати відповідні кнопки у головному меню, наприклад: «Розклад на сьогодні», після чого з’явиться відповідне повідомлення з розкладом на сьогодні (Рисунок 4.29). Для інших кнопок будуть відправлятися

відповідні повідомлення подібні до повідомлення з розкладом на сьогодні (Рисунок 4.30).

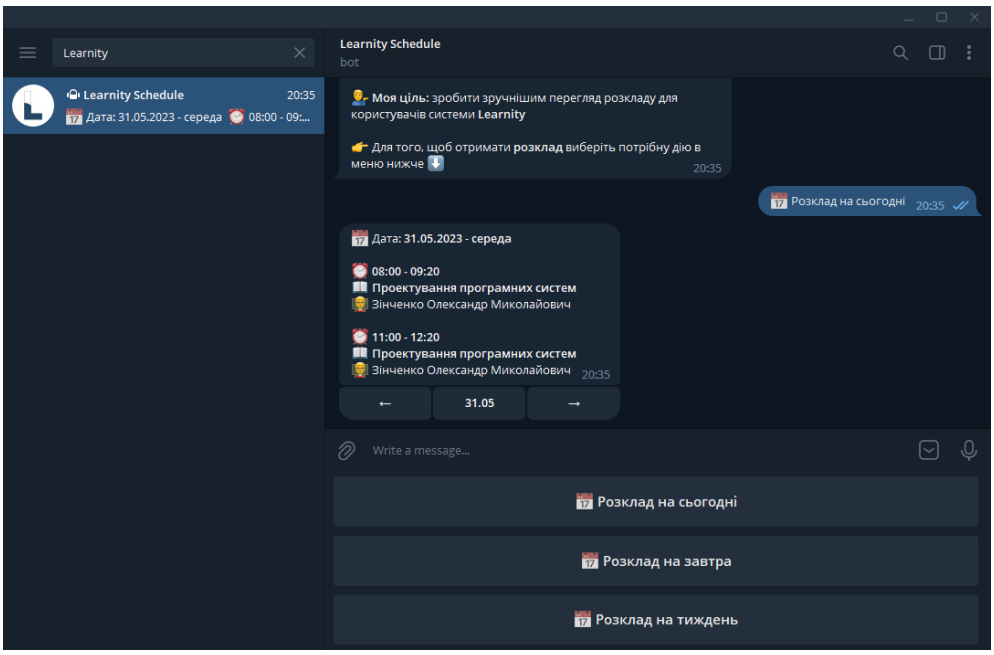

Рисунок 4.29 – Повідомлення з розкладом на сьогодні

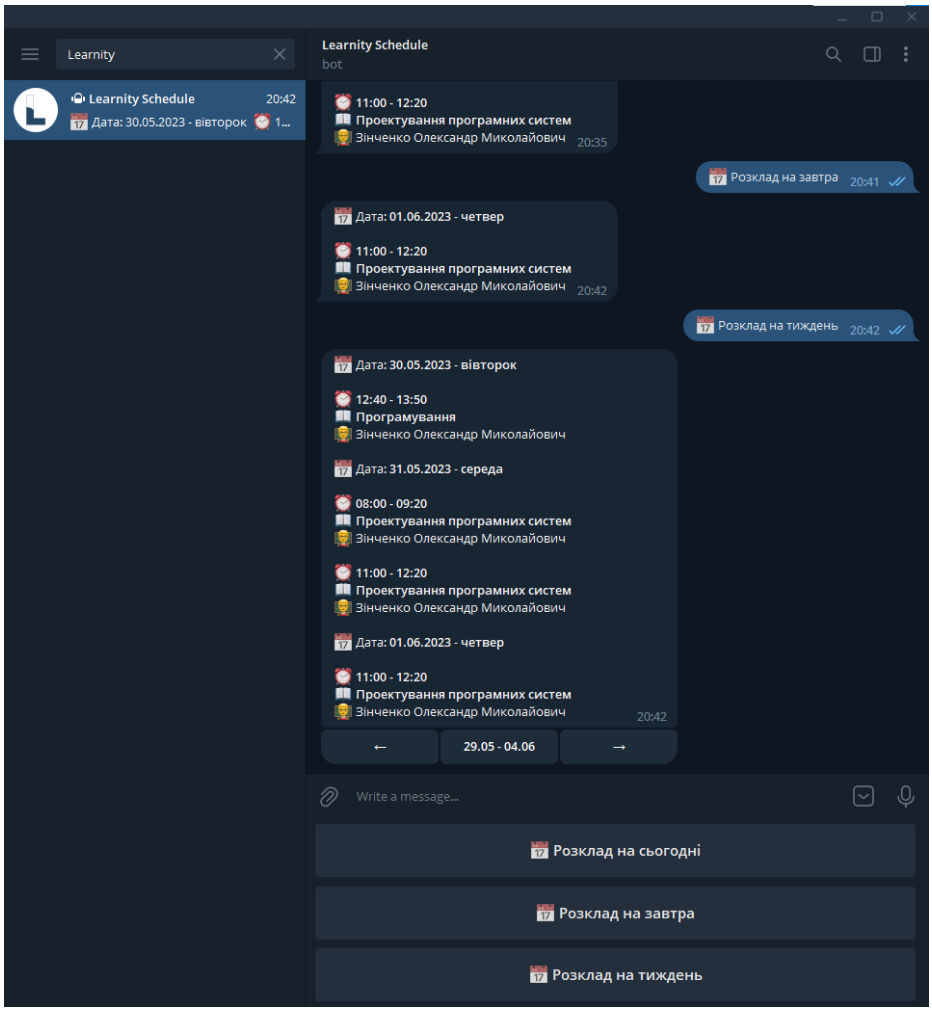

Рисунок 4.30 – Повідомлення про розклад на завтра і на тиждень

5. Також в телеграм-боті присутня можливість гортати розклад, натискаючи на відповідні стрілки під останнім повідомленням. При натисканні такої стрілки в залежності від вибраного виду розкладу (тиждень для «Розклад на тиждень» та день для «Розклад на сьогодні» і «Розклад на завтра»), користувачу буде відправлено розклад на наступний або попередній день чи тиждень, а кнопки зі стрілками буде прибрано з минулого повідомлення (Рисунок 4.31). Якщо розкладу не буде знайдено на потрібну дату, то буде виведено відповідне повідомлення (Рисунок 4.32).

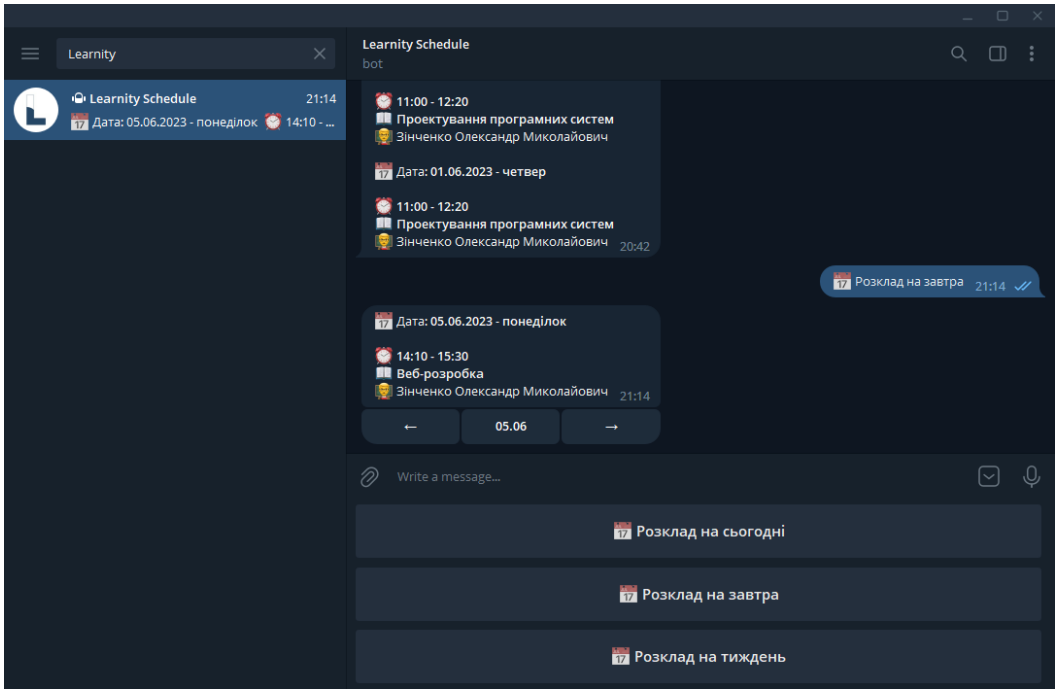

Рисунок 4.31 – Повідомлення про розклад на декілька днів вперед

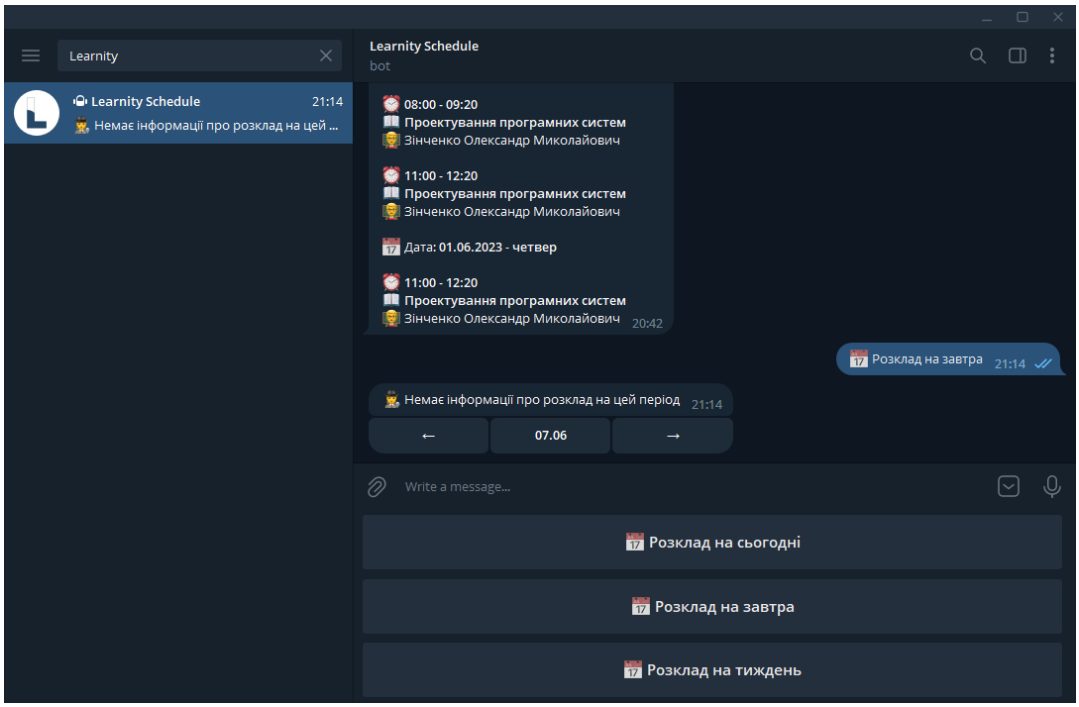

Рисунок 4.32 – Повідомлення про відсутність розкладу

Також, окрім можливості перегляду розкладу через телеграм-бота, було реалізовано функціональність перегляду розкладу в самій системі. Щоб переглянути розклад на сайті системи, потрібно перейти до сторінки з розкладом, натиснувши на відповідну кнопку в навігаційній панелі, після чого вибрати потрібний місяць. В результаті чого буде показано розклад за всіма курсами користувача за вибраний місяць (Рисунок 4.33).

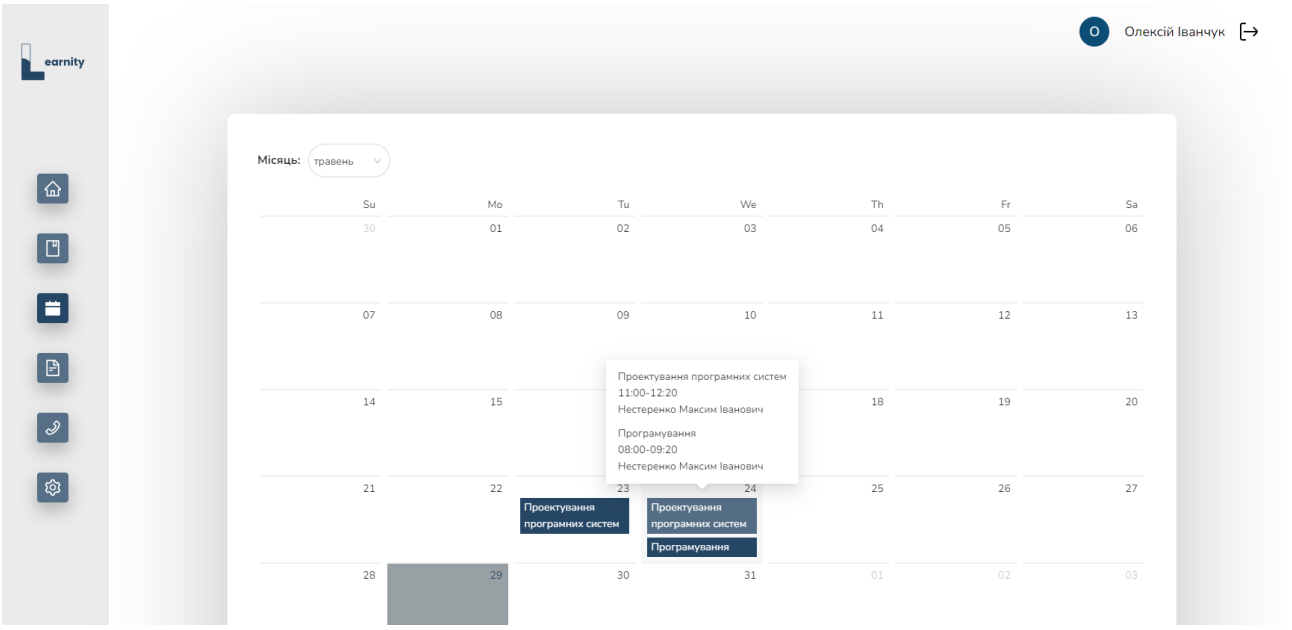

Рисунок 4.33 – Розклад по всім курсам за вибраний місяць

Розгортання. Для хостингу розробленої системи було вирішено обрати сервіс DigitalOcean. DigitalOcean пропонує досить багато варіантів того, як можна розгорнути власне програмне забезпечення на їх серверах. Одним із них є досить простий, а тому і не найдешевший, коли процес розгортання робиться майже автоматично, потрібно тальки вказати репозиторій з проектом, сервер, на якому він буде працювати; вибрати вартість та вказати змінні середовища для програми, а для баз даних та сховищ файлів все ще простіше: потрібно просто вибрати сервер та вартість. Інший варіант пропонує розгортання на віддаленій машині вручну, тоді ціна буде менша, але і складність більше, а при цьому базу даних та сховище можна все одно використовувати за допомогою автоматизованих сервісів. Останній варіант: розгортання всіх складових розробленої системи на віддалено сервері та їх налаштування вручну, що потребує певну кількість часу та енергії, а тому цей варіант є найдешевшим.

Було вирішено обрати третій варіант для серверної частини та перший для клієнтської, тобто сервер з базою даних, інструментом кешування, сховищем файлів та сервісом телеграм-боту було розгорнуто на віддаленій машині вручну, а клієнтська частина була розміщена за допомогою автоматизованого інструменту DigitalOcean. Було вирішено спробувати ці два варіанти, адже вони дають змогу спробувати різні способи розгортання на цьому сервісі, при найменших витратах.

Алгоритм проведеного розгортання системи на сервісі DigitalOcean можна описати наступним чином:

1. Було орендовано сервер під розгортання системи.

2. Налаштовано доступ до цього серверу SSH-доступу, за допомогою програми PuTTY.

3. Згенеровано та використано SSH-ключ для доступу до репозиторіїв серверної частини системи дистанційного навчання та сервісу телеграм-боту на GitHub.

4. Клоновано ці репозиторії до віддаленого серверу.

5. Так як серверну частину розробленої системи містить в собі файл docker-compose, що описує те, які інструменти потрібно буде запустити в ізольованому

контейнері Docker, а саме: базу даних, інструмент кешування та сховище файлів, то на віддалений сервер було встановлено Docker та Docker compose.

6. Далі зроблено білд контейнеру Docker та запущено його

8. Було налаштовано змінні середовища серверної частини та сервіс телеграм-бота.

9. Встановлено pm2 та запущено через нього серверну частину та сервіс телеграм-боту для того, щоб нівелювати їх шанс безповоротного вимкнення або падіння внаслідок перенавантаження серверу або якихось інших причин.

10. Потім було розгорнути клієнтську частину за допомогою автоматизованого сервісу розгортання додатків DigitalOcean. Під час налаштувань було вказано змінні середовища для того, щоб налагодити комунікацію клієнтської частини із серверною.

11. Після цього розгортання системи дистанційного навчання було завершено і нею тепер може користуватися будь-хто в мережі Інтернет.

ВИСНОВКИ

В наш час питання організації дистанційного навчання є дуже актуальним, бо воно допомагає людині безперешкодно здобувати знання, вирішуючи проблеми: часу, доступності, місцезнаходження та комунікації. З огляду на цю актуальність і було вирішено провести детальний аналіз цієї області та скласти список технологій та інструментів, потрібних для розробки програмного забезпечення системи дистанційного навчання, в рамках курсового проекту.

В результаті проведеної роботи в рамках кваліфікаційної роботи:

- визначено актуальність теми кваліфікаційної роботи;
- було складено вимоги до функціональності частин розробленої системи дистанційного навчання;
- здійснено огляд систем аналогічного призначення;
- складено список технологій та інструментів, потрібних для розробки системи;
- побудовано діаграму прецедентів для частин, що будуть розроблятися та діаграму алгоритму при розгортанні.
- реалізовано всю функціональність, що було описана у вимогах до розробки частин системи;
- складено та описано інструкцію з використання розроблених частин;
- проведено розгортання системи на віддаленому сервері.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. 10 Best Distance Learning Software Trending Now [Електронний ресурс]. – 2022. – Режим доступу: URL: https://www.proprofstraining.com/blog/best-distance-learning-software.
2. CMS Moodle - система для навчання і тестування онлайн. Огляд функціоналу [Електронний ресурс] – Режим доступу: URL: https://hyperhost.ua/info/uk/cms-moodle-sistema-dlya-navchannya-i-testuvannya-onlayn-oglyad-funktsionalu.
3. 27 Best Online Learning Platforms [Електронний ресурс] – Режим доступу: URL: https://www.learnworlds.com/online-learning-platforms/#:~:text=Some%20of%20the%20best%20online,Thinkific%2C%20Kajabi%2C%20and%20Podia.
4. Best Online Learning Platforms – Starting Off on the Right Foot [Електронний ресурс] – Режим доступу: URL: https://www.ispringsolutions.com/blog/best-online-learning-platforms#canvas.
5. Node.js [Електронний ресурс] – Режим доступу: URL: https://en.wikipedia.org/wiki/Node.js.
6. TypeScript - Overview [Електронний ресурс] – Режим доступу: URL: https://www.tutorialspoint.com/typescript/typescript_overview.
7. The Best Guide to Know What Is React [Електронний ресурс] – Режим доступу: URL: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs.
8. Vite | Next Generation Frontend Tooling [Електронний ресурс] – Режим доступу: URL: https://vitejs.dev/.
9. What Is Redux? [Електронний ресурс] – Режим доступу: URL: https://medium.com/swlh/what-is-redux-b16b42b33820.
10. Learn Redux Toolkit – The Recommended Way to Use Redux [Електронний ресурс] – Режим доступу: URL: https://www.freecodecamp.org/news/learn-redux-toolkit-the-recommended-way-to-use-redux/.

11. Introduction to Redux Saga [Електронний ресурс] – Режим доступу: URL: https://blog.loginradius.com/engineering/introduction-to-redux-saga/.
12. What is Axios.js and why should I care? [Електронний ресурс] – Режим доступу: URL: https://medium.com/@MinimalGhost/what-is-axios-js-and-why-should-i-care-7eb72b111dc0.
13. Zod [Електронний ресурс] – Режим доступу: URL: https://github.com/colinhacks/zod.
14. Ant Design [Електронний ресурс] – Режим доступу: URL: https://www.geeksforgeeks.org/ant-design/.
15. What Is Nest.JS? Why Should You Use It? [Електронний ресурс] – Режим доступу: URL: https://www.turing.com/blog/what-is-nest-js-why-use-it-in-2022/.
16. Authentication [Електронний ресурс] – Режим доступу: URL: https://docs.nestjs.com/security/authentication.
17. TypeORM [Електронний ресурс] – Режим доступу: URL: https://typeorm.io/.
18. Nodemailer [Електронний ресурс] – Режим доступу: URL: https://nodemailer.com/about/.
19. Swagger [Електронний ресурс] – Режим доступу: URL: https://www.techtarget.com/searchapparchitecture/definition/Swagger.
20. PostgreSQL: About [Електронний ресурс] – Режим доступу: URL: https://www.postgresql.org/about/.
21. What is pgAdmin? [Електронний ресурс] – Режим доступу: URL: https://www.adservio.fr/post/what-is-pgadmin.
22. Redis [Електронний ресурс] – Режим доступу: URL: https://redis.io/.
23. telegraf.js [Електронний ресурс] – Режим доступу: URL: https://telegraf.js.org/.
24. Git [Електронний ресурс] – Режим доступу: URL: https://git-scm.com/.
25. Visual Studio Code [Електронний ресурс] – Режим доступу: URL: https://en.wikipedia.org/wiki/Visual_Studio_Code.

26. What is Figma? (And How to Use Figma for Beginners) [Електронний ресурс] – Режим доступу: URL: https://www.theme-junkie.com/what-is-figma/.
27. Docker overview | Docker Documentation [Електронний ресурс] – Режим доступу: URL: https://docs.docker.com/get-started/overview/.
28. DigitalOcean Hosting Review PROS & CONS (2022) – Excellent Choice For Hosting? [Електронний ресурс] – Режим доступу: URL: https://www.kasareviews.com/digitalocean-hosting-review-pros-cons.

ДОДАТОК А. ВИХІДН КОДИ

КОД ГОЛОВНОГО ФАЙЛУ

```
import { Context, Telegraf } from 'telegraf';


import { Timetable, TimetableData } from './api/types';
import {
  checkIsObjectEmpty,
  formatDate,
  formatDateToTime,
  formatDateWithoutYear,
  getDateWithOffset,
  getNextDate,
  getNextWeekEndDate,
  getNextWeekStartDate,
  getPreviousDate,
  getPreviousWeekEndDate,
  getPreviousWeekStartDate,
  getWeekDay,
  getWeekEndDate,
  getWeekStartDate,
} from './helpers';
import { envConfig, localization } from './config';
import { AuthMessages, MainMenuMessages } from './messages';
import { MainApiService, mainAxiosRequestConfig } from './api';
import {
  TimetableCallbackData,
  TimetableCallbackDataType,
  TimetableModeEnum,
  TimetableSliderData,
} from './keyboards/types';
```

```
import { UsersStateManager } from './tools/UsersStatesManager';
import { MainMenuKeyboards } from './keyboards';
import { BotInlineKeyboard } from './types';

const { todayTimetable, tomorrowTimetable, weekTimetable } =
localization.keyboards.mainMenu;
const { noSchedule } = localization.messages.errors;
const { previous, next } = localization.inlineKeyboards.timetableSlider;

const bot = new Telegraf(envConfig.botToken);
const usersStatesManager = new UsersStateManager();

const makeMessageFromTimetables = (timetables: Timetable[]): string => {
  return timetables
    .map(({ date, coursesTimetables }) =>
      [
        `□ Дата: <b>${formatDate(new Date(date))}</b> - <b>${getWeekDay(new
Date(date))}</b>`,
        coursesTimetables
          .map((courseTimetable) =>
            [
              `□ <b>${formatDateToTime(new Date(courseTimetable.startTime))} -
${formatDateToTime(
                new Date(courseTimetable.endTime),
              )}</b>`,
              `□ <b>${courseTimetable.courseName}</b>\n□□
${courseTimetable.teacherName}`,
            ].join('\n'),
          )
          .join('\n\n'),
```

```
      ].join('\n\n'),
    )
    .join('\n\n');
};

const parseTimetableDataList = (dataList: TimetableData[]): Timetable[] => {
  const result: Timetable[] = [];
  for (let i = 0; i < dataList.length; i++) {
    const dataEntry = dataList[i];
    const parsedTimetable = result.find((timetable) => timetable.date ===
dataEntry.date);
    const teacher = dataEntry.course.teacher;
    const teacherName = `${teacher.lastName} ${teacher.firstName}
${teacher.middleName}`;
    if (parsedTimetable) {
      parsedTimetable.coursesTimetables.push({
        teacherName,
        courseName: dataEntry.course.name,
        startTime: dataEntry.startTime,
        endTime: dataEntry.endTime,
      });
      continue;
    }
    result.push({
      date: dataEntry.date,
      coursesTimetables: [
        {
          teacherName,
          courseName: dataEntry.course.name,
          startTime: dataEntry.startTime,
```

```
          endTime: dataEntry.endTime,
        },
      ],
    });
  }
  return result;
};

const getTimetablesData = async (
  startDate: Date,
  endDate: Date,
  telegramId: number,
): Promise<TimetableData[] | undefined> => {
  const result = await MainApiService.selectTimetables(
    { startDate, endDate },
    { ...mainAxiosRequestConfig.headers, 'Telegram-Bot-User-Token': telegramId },
  );
  return result.data;
};

const getCallbackData = (
  mode: TimetableModeEnum,
  type: TimetableCallbackDataType,
  startDate: Date,
  endDate: Date,
): TimetableCallbackData => ({
  mode,
  type,
  dates:
    mode === TimetableModeEnum.DAY
```

```
      ? {
          startDate:
            type === TimetableCallbackDataType.PREVIOUS
              ? getPreviousDate(startDate)
              : getNextDate(startDate),
          endDate:
            type === TimetableCallbackDataType.PREVIOUS
              ? getPreviousDate(endDate)
              : getNextDate(endDate),
        }
      : {
          startDate:
            type === TimetableCallbackDataType.PREVIOUS
              ? getPreviousWeekStartDate(startDate)
              : getNextWeekStartDate(startDate),
          endDate:
            type === TimetableCallbackDataType.PREVIOUS
              ? getPreviousWeekEndDate(endDate)
              : getNextWeekEndDate(endDate),
        },
});

const getTimetableSliderData = (
  mode: TimetableModeEnum,
  startDate: Date,
  endDate: Date,
): TimetableSliderData => {
  const previousCallbackData = getCallbackData(
    mode,
    TimetableCallbackDataType.PREVIOUS,
```

```
    startDate,
    endDate,
  );
  const nextCallbackData = getCallbackData(
    mode,
    TimetableCallbackDataType.NEXT,
    startDate,
    endDate,
  );
  return {
    text:
      mode === TimetableModeEnum.DAY
        ? formatDateWithoutYear(startDate)
        : `${formatDateWithoutYear(startDate)} -
${formatDateWithoutYear(endDate)}`,
    previous: previousCallbackData,
    next: nextCallbackData,
  };
};

const editTimetableMessage = (
  chatId: number,
  messageId: number,
  timetablesMessage: string,
  timetableSliderText: string,
) => {
  bot.telegram.editMessageText(chatId, messageId, undefined, timetablesMessage, {
    parse_mode: 'HTML',
    ...(MainMenuKeyboards.getTimetableSliderInlineKeyboard(
      timetableSliderText,
```

```
    ) as BotInlineKeyboard),
  });
};

const sendTimetableMessage = async (
  ctx: Context,
  userId: number,
  startDate: Date,
  endDate: Date,
  mode: TimetableModeEnum,
) => {
  const timetablesData = await getTimetablesData(startDate, endDate, userId);
  const timetableSliderData = getTimetableSliderData(mode, startDate, endDate);

  usersStatesManager.set(userId, { timetableSliderData });

  if (!timetablesData || !timetablesData.length) {
    const message = await MainMenuMessages.sendNoScheduleMessage(ctx,
timetableSliderData.text);
    usersStatesManager.set(userId, { lastTimetableMessageId: message.message_id });
    return;
  }

  const timetables = parseTimetableDataList(timetablesData);
  const timetablesMessage = makeMessageFromTimetables(timetables);
  const message = await MainMenuMessages.sendScheduleMessage(
    ctx,
    timetablesMessage,
    timetableSliderData.text,
  );
```

```
  usersStatesManager.set(userId, { lastTimetableMessageId: message.message_id });
};

const updateTimetableMessage = async (
  chatId: number,
  userId: number,
  messageId: number,
  startDate: Date,
  endDate: Date,
  mode: TimetableModeEnum,
) => {
  const timetablesData = await getTimetablesData(startDate, endDate, userId);
  const timetableSliderData = getTimetableSliderData(mode, startDate, endDate);

  usersStatesManager.set(userId, { timetableSliderData });

  if (!timetablesData || !timetablesData.length) {
    editTimetableMessage(chatId, messageId, noSchedule, timetableSliderData.text);
    return;
  }

  const timetables = parseTimetableDataList(timetablesData);
  const timetablesMessage = makeMessageFromTimetables(timetables);

  editTimetableMessage(chatId, messageId, timetablesMessage,
timetableSliderData.text);
};

const removeMessageInlineKeyboard = async (chatId: number, messageId: number)
=> {
```

```
  return bot.telegram.editMessageReplyMarkup(chatId, messageId, undefined,
undefined);
};

bot.start(async (ctx) => {
  const telegramId = ctx.from.id;
  if (!ctx.startPayload) {
    const result = await MainApiService.getLoginStatus({
      ...mainAxiosRequestConfig.headers,
      'Telegram-Bot-User-Token': telegramId,
    });
    if (result.error) return AuthMessages.sendErrorMessage(ctx);
    return;
  }
  const result = await MainApiService.login({ telegramId, token: ctx.startPayload });
  if (result.error) return AuthMessages.sendErrorMessage(ctx);
  MainMenuMessages.sendEnterMessage(ctx);
});

bot.hears([todayTimetable, tomorrowTimetable, weekTimetable], async (ctx) => {
  const userFromId = ctx.from.id;
  const userState = usersStatesManager.get(userFromId);

  let timetableMode = TimetableModeEnum.DAY;
  let startDate = new Date();
  let endDate = new Date();

  if (ctx.message.text === tomorrowTimetable) {
    startDate = getDateWithOffset(startDate, 1);
    endDate = getDateWithOffset(endDate, 1);
```

```
  } else if (ctx.message.text === weekTimetable) {
    timetableMode = TimetableModeEnum.WEEK;
    startDate = getWeekStartDate(startDate);
    endDate = getWeekEndDate(endDate);
  }

  await sendTimetableMessage(ctx, userFromId, startDate, endDate, timetableMode);
  if (userState && userState.lastTimetableMessageId) {
    removeMessageInlineKeyboard(ctx.chat.id, userState.lastTimetableMessageId);
  }
});

bot.action([previous.callback_data, next.callback_data], async (ctx) => {
  const callbackQuery = ctx.callbackQuery;
  const callbackQueryData = callbackQuery.data;
  const userFromId = callbackQuery.from.id;
  const userState = usersStatesManager.get(userFromId);
  if (
    !userState ||
    checkIsObjectEmpty(userState) ||
    !userState.timetableSliderData ||
    !callbackQueryData
  )
    return;

  const { mode, dates } = userState.timetableSliderData[
    callbackQueryData as keyof TimetableSliderData
  ] as TimetableCallbackData;
  const lastTimetableMessageId = userState.lastTimetableMessageId;
```

```
  let timetableMode = mode;
  let startDate = new Date(dates.startDate);
  let endDate = new Date(dates.endDate);

  if (lastTimetableMessageId && ctx.chat) {
    updateTimetableMessage(
      ctx.chat.id,
      userFromId,
      lastTimetableMessageId,
      startDate,
      endDate,
      timetableMode,
    );
  }
});

bot.launch({
  webhook: {
    domain: `${envConfig.webhookServerDomain}`,
    hookPath: `/bot/${bot.telegram.token}`,
    port: parseInt(envConfig.port),
  },
});
```

КОД СЕРВІСУ РОЗКЛАДУ КУРСУ

```
import { ConflictException, Injectable, NotFoundException } from
'@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { instanceToPlain } from 'class-transformer';
import {
```

```
  FindConditions,
  FindManyOptions,
  FindOneOptions,
  FindOptionsUtils,
  RemoveOptions,
  Repository,
  SaveOptions,
  SelectQueryBuilder,
} from 'typeorm';

import { ErrorTypeEnum } from 'src/common/enums';
import { CourseEntity, CourseTimetableEntity } from '../entities';
import { CoursesService } from './courses.service';
import { PaginationCourseTimetablesDto } from '../dto';
import { UserEntity, UserRoleEnum } from 'src/modules/users/entities';
import { SelectCourseTimetableForUserDto } from '../dto/course-timetables/select-
course-timetables-for-user.dto';

@Injectable()
export class CourseTimetablesService {
  /**
   * [description]
   * @param courseTopicEntityRepository
   * @param courseService
   */
  constructor(
    @InjectRepository(CourseTimetableEntity)
    private readonly courseTopicEntityRepository:
Repository<CourseTimetableEntity>,
    private readonly courseService: CoursesService,
```

```
  ) {}

  /**
   * [description]
   * @param conditions
   * @param entityLike
   * @param user
   * @param options
   */
  public async createOne(
    conditions: Partial<CourseEntity>,
    entityLike: Partial<CourseTimetableEntity>,
    options: SaveOptions = { transaction: false },
  ): Promise<CourseTimetableEntity> {
    return this.courseTopicEntityRepository.manager.transaction(async () => {
      const course = await this.courseService.selectOne(conditions);
      const entity = this.courseTopicEntityRepository.create({ ...entityLike, course });
      const { id } = await this.courseTopicEntityRepository.save(entity,
options).catch(() => {
        throw new
ConflictException(ErrorTypeEnum.COURSE_TIMETABLE_ALREADY_EXISTS);
      });

      return this.selectOne({ id });
    });
  }

/**
   * [description]
   * @param optionsOrConditions
```

```
   */
  public find(
    optionsOrConditions?: FindManyOptions<CourseTimetableEntity>,
  ): SelectQueryBuilder<CourseTimetableEntity> {
    const metadata = this.courseTopicEntityRepository.metadata;
    const qb = this.courseTopicEntityRepository.createQueryBuilder(
      FindOptionsUtils.extractFindManyOptionsAlias(optionsOrConditions) ||
metadata.name,
    );

    if (
      !FindOptionsUtils.isFindManyOptions(optionsOrConditions) ||
      optionsOrConditions.loadEagerRelations !== false
    ) {
      FindOptionsUtils.joinEagerRelations(qb, qb.alias, metadata);

      /**
       * Place for common relation
       * @example qb.leftJoinAndSelect('CourseTopicEntity.relation_field',
'CourseTopicEntity_relation_field')
       */
    }

    return FindOptionsUtils.applyFindManyOptionsOrConditionsToQueryBuilder(qb,
optionsOrConditions);
  }

/**
   * [description]
   * @param conditions
```

```
   * @param user
   * @param options
   */
  public async selectMany(
    conditions: Partial<CourseEntity>,
    options: FindManyOptions<CourseTimetableEntity> = { loadEagerRelations: false
},
  ): Promise<PaginationCourseTimetablesDto> {
    const course = await this.courseService.selectOne(conditions);
    const qb = this.find(instanceToPlain(options));
    if (options.where) qb.where(options.where);
    qb.andWhere({ course });
    return qb
      .getManyAndCount()
      .then((data) => new PaginationCourseTimetablesDto(data))
      .catch(() => {
        throw new
NotFoundException(ErrorTypeEnum.COURSE_TIMETABLES_NOT_FOUND);
      });
  }

public async selectManyForUser(
    user: Partial<UserEntity>,
    options: SelectCourseTimetableForUserDto,
  ): Promise<CourseTimetableEntity[]> {
    const qb = this.find();
    qb.innerJoinAndSelect('CourseTimetableEntity.course',
'CourseTimetableEntity_course');
    if (user.role === UserRoleEnum.TEACHER)
      qb.andWhere('CourseTimetableEntity_course.teacherId = :teacherId', {
```

```
        teacherId: user.id,
      });
    if (user.role === UserRoleEnum.STUDENT)
      qb.innerJoin(
        'CourseTimetableEntity_course.participants',
        'CourseTimetableEntity_course_participants',
        'CourseTimetableEntity_course_participants.userId = :userId',
        { userId: user.id },
      );
    qb.innerJoinAndSelect(
      'CourseTimetableEntity_course.teacher',
      'CourseTimetableEntity_course_teacher',
    );
    qb.andWhere(
      'CourseTimetableEntity.date >= :startDate AND CourseTimetableEntity.date <=
:endDate',
      options,
    );
    qb.orderBy('date', 'ASC');
    return qb.getMany().catch(() => {
      throw new
NotFoundException(ErrorTypeEnum.COURSE_TIMETABLES_NOT_FOUND);
    });
  }

/**
   * [description]
   * @param conditions
   * @param user
   * @param options
```

```
   */
  public async selectOne(
    conditions: FindConditions<CourseTimetableEntity>,
    options: FindOneOptions<CourseTimetableEntity> = { loadEagerRelations: false
},
  ): Promise<CourseTimetableEntity> {
    const qb = this.find(instanceToPlain(options)).where(conditions);
    qb.innerJoin('CourseTimetableEntity.course', 'CourseTimetableEntity_course');
    return qb.getOneOrFail().catch(() => {
      throw new
NotFoundException(ErrorTypeEnum.COURSE_TIMETABLE_NOT_FOUND);
    });
  }

  /**
   * [description]
   * @param conditions
   * @param entityLike
   * @param user
   * @param options
   */
  public async updateOne(
    conditions: Partial<CourseTimetableEntity>,
    entityLike: Partial<CourseTimetableEntity>,
    options: SaveOptions = { transaction: false },
  ): Promise<CourseTimetableEntity> {
    return this.courseTopicEntityRepository.manager.transaction(async () => {
      const mergeIntoEntity = await this.selectOne(conditions);
      const entity = this.courseTopicEntityRepository.merge(mergeIntoEntity,
entityLike);
```

```
      const { id } = await this.courseTopicEntityRepository.save(entity,
options).catch(() => {
        throw new
ConflictException(ErrorTypeEnum.COURSE_TIMETABLE_ALREADY_EXISTS);
      });
      return this.selectOne({ id });
    });
  }

/**
   * [description]
   * @param conditions
   * @param user
   * @param options
   */
  public async deleteOne(
    conditions: FindConditions<CourseTimetableEntity>,
    options: RemoveOptions = { transaction: false },
  ): Promise<CourseTimetableEntity> {
    return this.courseTopicEntityRepository.manager.transaction(async () => {
      const entity = await this.selectOne(conditions);
      return this.courseTopicEntityRepository.remove(entity, options).catch(() => {
        throw new
NotFoundException(ErrorTypeEnum.COURSE_TIMETABLE_NOT_FOUND);
      });
    });
  }
```