МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра дискретного аналізу та інтелектуальних систем
(повна назва кафедри)

Дипломна робота

ЗАСТОСУНОК ДЛЯ ЗАПИСУ НА ДОДАТКОВІ ДИСЦИПЛІНИ

Виконав: студент групи ПМі-41с
спеціальності
122 «Комп'ютерні науки»
(шифр і назва спеціальності)

Сергієнко А.Р.
(підпис) (прізвище та ініціали)

Керівник доцент Клакович Л.М.
(підпис) (прізвище та ініціали)

Рецензент _____
(підпис) (прізвище та ініціали)

2023

Суть дипломної роботи

Робота складається із вступу, трьох розділів, висновку та джерел. Обсяг дипломної роботи: 28 сторінок сторінки та 10 рисунків. Список використаних джерел складається з 5 найменувань.

Мета даної роботи – створення повноцінного та конкурентноздатного веб-застосунку для запису на додаткові дисципліни у Львівському Національному Університеті імені Івана Франка. Схожі додатки вже були розроблені для інших університетів, тож я проаналізувала їх та відповідно до отриманих даних склала оптимальні вимоги до застосунку, що розробляється.

Автоматизований процес запису на додаткові дисципліни в навчальному закладі є важливим, адже дозволяє зекономити час працівників університетів, який вони витрачають на складання списків за результатами заповнених студентами форм. Крім того, проект можна покращити ще сильніше, інтегрувавши його в інтернет-екосистему університету з використанням протоколу OIDC.

Перший розділ описує теоретичні відомості та досліджує вимоги і варіації розробки застосунку на основі існуючих прикладів. Другий розділ описує проектування та реалізацію проекту. Третій розділ демонструє взаємодію із розробленою програмою.

ЗМІСТ

ВСТУП

У сучасному освітньому середовищі студенти часто стикаються з необхідністю обрати додаткові дисципліни для вивчення в університеті. Вибір таких дисциплін може вплинути на рівень знань, компетенції та професійну кваліфікацію студента. Проте, наразі бракує системи, яка б забезпечувала ефективну та прозору реєстрацію на додаткові дисципліни, враховуючи індивідуальні потреби студентів.

Актуальність проблеми полягає у тому, що необхідно забезпечити студентам можливість свідомого вибору додаткових дисциплін, які відповідають їхнім особистим і професійним цілям. Дозвіл студентам обирати додаткові навчальні дисципліни може підвищити їхню зацікавленість темами, навчальною програмою та певними напрямами досліджень. Це також може розвинути їхню уяву та креативність. Крім того, це дає учням можливість обирати матеріали та ресурси, які вони будуть використовувати. Вони досліджують теми і питання, про які хочуть дізнатися, і розширюють свої інтереси. Це може допомогти студентам обрати предмети, які матимуть значний вплив на їхні кар'єрні можливості.

Крім того, існує потреба у встановленні чітких принципів та процедур реєстрації, які б гарантували рівність умов для усіх студентів і уникнення конфліктів у розкладі.

Метою даної дипломної роботи є розробка та реалізація системи реєстрації на додаткові дисциплін, яка враховуватиме потреби студентів та сприятиме їхньому успішному навчанню. Основними завданнями роботи будуть:

1. Огляд літератури щодо поняття та принципів реєстрації на додаткові дисципліни, аналіз існуючих систем та технологій.

2. Розробка архітектури системи реєстрації, включаючи компоненти, їх функціональність та взаємодію з іншими інформаційними системами.
3. Впровадження та оцінювання ефективності розробленої системи реєстрації на додаткові дисципліни.

Результатом роботи буде створення автоматизованої системи реєстрації, яка сприятиме зручному та прозорому процесу вибору додаткових дисциплін для студентів та допоможе забезпечити їхнє успішне навчання та професійний розвиток.

ПОСТАНОВКА ЗАДАЧІ

Розробити веб-застосунок для запису на додаткові дисципліни в навчальному закладі.

Під час виконання дипломної роботи необхідно виконати такі завдання:

1. Ознайомитись з теоретичною базою та існуючими рішеннями від інших ВНЗ.
2. Спроектувати веб-застосунок: підібрати технології, визначити архітектуру, спроектувати API.
3. Розробити та продемонструвати веб-застосунок згідно опрацьованих вимог та заданої архітектури.

1. ТЕОРЕТИЧНА ЧАСТИНА

1.1 Огляд сучасних методів реєстрації на додаткові дисципліни

У сучасних вищих навчальних закладах в Україні існують різні методи реєстрації на додаткові дисципліни, які надають студентам можливість розширити свої знання та вміння у певній галузі. Давайте розглянемо декілька прикладів таких методів, які використовуються в окремих університетах України:

1. Вінницький національний технічний університет (ВНТУ):

У ВНТУ для реєстрації на додаткові дисципліни використовується система реєстрації на основі вибору за бажанням. Студентам надається можливість обрати додаткові дисципліни з певного списку пропозицій, але кількість доступних місць на кожну дисципліну обмежена. Розподіл місць здійснюється на основі вибору студентів. Цей підхід забезпечує гнучкість та індивідуальний підхід до реєстрації, дозволяючи студентам обирати дисципліни залежно від їхніх інтересів і потреб.

2. Національний університет “Львівська політехніка” (НУЛП):

У НУЛПвикористовується система реєстрації на основі черги. Студенти подають заявки на реєстрацію протягом визначеного періоду часу, після чого система автоматично впорядковує їх у чергу на основі певних критеріїв, таких як час подачі заявки або рейтинг студента. При наявності вільних місць, студенти призначаються на додаткові дисципліни відповідно до чергового порядку. Цей метод спрощує процес реєстрації та забезпечує справедливий розподіл доступних місць.

3. Київський національний університет ім. Тараса Шевченка (КНУ):

В КНУ використовується комбінація ручного підходу та системи реєстрації на основі черги. Студенти мають можливість самостійно звернутися до адміністрації та подати заявки на реєстрацію на додаткові

дисципліни. Після цього, система автоматично впорядковує заявки у чергу відповідно до встановлених критеріїв. Якщо кількість студентів, які бажають зареєструватися на певну дисципліну, перевищує доступні місця, то пріоритет при розподілі місць надається студентам з кращими результатами або іншими критеріями.

Ці приклади ілюструють різні підходи до реєстрації на додаткові дисципліни в різних університетах України. Кожен з цих методів має свої переваги та обмеження, і вищі навчальні заклади вибирають той, який найкраще відповідає їхнім потребам та умовам.

1.2 Аналіз вимог до системи реєстрації на додаткові дисципліни

Перед тим як розпочати розробку додатку для реєстрації на додаткові дисципліни, необхідно провести аналіз вимог до системи. Цей аналіз допоможе з'ясувати, які функціональні та нефункціональні вимоги повинні бути враховані для забезпечення ефективної та надійної реєстраційної системи.

Функціональні вимоги визначають основні функції та можливості, які має мати система реєстрації на додаткові дисципліни. До них можуть належати:

1. Реєстрація студентів на додаткові дисципліни: Система повинна дозволяти студентам подавати заявки на реєстрацію на конкретні додаткові дисципліни та враховувати обмеження щодо кількості доступних місць.

2. Модерування та підтвердження заявок: Адміністратор системи повинен мати можливість переглядати та підтверджувати заявки студентів, забезпечуючи справедливий розподіл місць на додаткові дисципліни.

3. Керування списками додаткових дисциплін: Система повинна дозволяти адміністраторам додавати, видаляти та змінювати список доступних додаткових дисциплін, їх опис та кількість доступних місць.

4. Інформування студентів: Система повинна надсилати сповіщення студентам про статус їх заявок, призначення на додаткові дисципліни та зміни в розкладі.

Нефункціональні вимоги визначають властивості та обмеження системи. До них можуть відноситись:

1. Ефективність: Система повинна працювати швидко та ефективно, забезпечуючи мінімальний час реакції при обробці заявок та запитів користувачів.

2. Надійність: Система повинна бути стабільною та надійною, забезпечуючи захист даних та уникнення втрати інформації.

3. Зручність використання: Інтерфейс системи повинен бути зрозумілим та зручним для студентів та адміністраторів, не вимагати спеціальних навичок або надмірного зусилля для користування.

4. Масштабованість: Система повинна бути готовою до масштабування та обробки збільшеної кількості заявок та користувачів.

Проведений аналіз вимог дозволить чітко визначити функціональні та нефункціональні вимоги до системи реєстрації на додаткові дисципліни. Це стане основою для подальшої реалізації та розробки додатку, який задовольнятиме потреби студентів та адміністраторів університету.

1.3 Принципи та архітектура додатку для реєстрації на додаткові дисципліни

У даному підрозділі буде описано основні принципи та архітектуру додатку, який буде використовуватися для реєстрації студентів на додаткові

дисципліни. Ці принципи та архітектура визначають загальну структуру та функціональні компоненти системи.

Принципи, які лежать в основі додатку для реєстрації на додаткові дисципліни, включають:

1. Розширюваність: Архітектура додатку повинна бути розроблена з урахуванням можливості додавання нового функціоналу та змін без значних змін в загальній структурі системи. Це дозволить легко адаптувати додаток до змінних потреб користувачів.
2. Користувацький інтерфейс: Додаток повинен мати зручний та інтуїтивно зрозумілий користувацький інтерфейс, що дозволяє студентам та адміністраторам легко взаємодіяти з системою. Це включає зрозумілі форми та елементи керування, а також наявність повідомлень та сповіщень для інформування користувачів про статус їх заявок.
3. Безпека: Додаток повинен забезпечувати високий рівень безпеки, включаючи захист персональних даних студентів, автентифікацію та авторизацію користувачів, а також контроль доступу до різних функціональних можливостей системи.

Архітектура додатку для реєстрації на додаткові дисципліни побудована на основі клієнт-серверної архітектури. У такій архітектурі, клієнти (студенти та адміністратори) взаємодіють з сервером, який забезпечує обробку запитів, збереження даних та управління системою.

Центральною складовою архітектури є база даних, де зберігаються інформація про додаткові дисципліни, студентів, заявки та інші важливі дані. Клієнтська частина додатку реалізована у вигляді веб-інтерфейсу, який надає можливість студентам подавати заявки та переглядати статус

своїх заявок, адміністраторам - модерувати заявки та керувати списками додаткових дисциплін.

Для забезпечення безпеки та контролю доступу використовуються механізми аутентифікації та авторизації, наприклад, за допомогою ролей користувачів та доступу до відповідних функцій системи.

Ця архітектура дозволяє ефективно організувати роботу системи реєстрації на додаткові дисципліни, забезпечити швидкий та зручний доступ для користувачів та зберігання необхідної інформації у базі даних.

Таким чином, на основі аналізу застосунків подібної функціональності було визначено вимоги для побудови системи вибору додаткових дисциплін для вивчення студентами, які мають бути враховані при побудові свого застосунку.

2. ПРОГРАМНА РЕАЛІЗАЦІЯ

У даному розділі розглянуто програмну реалізацію додатку для реєстрації на додаткові дисципліни. Додаток базується на платформі ASP.NET Core MVC та використовує MS SQL Server для зберігання даних.

Для розробки додатку обрано ASP.NET Core MVC як основний фреймворк. ASP.NET Core MVC є потужним і гнучким фреймворком, який надає зручні інструменти для створення веб-додатків. Він має вбудовану підтримку моделей, контролерів та представлень, що дозволяє легко структурувати додаток і реалізувати необхідну логіку [1, 2].

Для зберігання даних використано MS SQL Server, що є потужною та надійною реляційною базою даних. MS SQL Server надає можливість створювати таблиці, зберігати дані та виконувати складні запити до бази даних [5].

Додаток поділений за архітектурою Model-View-Controller (MVC). Ця архітектурна модель дозволяє нам окремо розробляти логіку додатку (контролери), відображення даних (представлення) та зберігання даних (моделі). Цей підхід дозволяє підтримувати чистоту коду, легше розширювати та тестувати додаток.

У додатку реалізовано наступну функціональність:

- Реєстрація користувачів: Користувачі можуть створювати облікові записи, вводячи свої особисті дані та обираючи ідентифікатори для доступу до системи.

- Автентифікація та авторизація: Забезпечується безпека додатку шляхом перевірки введених користувачем даних та контролю рівнів доступу до різних функцій додатку.

- Перегляд доступних додаткових дисциплін: Користувачі можуть переглядати список доступних додаткових дисциплін, які вони можуть вибрати для реєстрації.

- Реєстрація на додаткові дисципліни: Користувачі можуть обрати додаткові дисципліни, на які вони бажають зареєструватися.

Під час розробки додатку, приділено увагу оптимізації та безпеці. Використано ефективні алгоритми та практики розробки, щоб забезпечити швидку роботу додатку та захистити його від можливих атак.

Загалом, програмна реалізація додатку на базі ASP.NET Core MVC та MS SQL Server надає зручний та надійний спосіб реєстрації на додаткові дисципліни [3-5]. Докладено зусиль для забезпечення якості, безпеки та ефективності додатку, щоб користувачі могли з легкістю використовувати його та зареєструватися на бажані дисципліни.

2.1 Реєстрація та авторизація

Клас `AccountController` відповідає за реалізацію функціональності, пов'язаної з реєстрацією та авторизацією користувачів. Контролер має наступні методи:

- `Login()` (GET): Відображає сторінку для входу в систему. Використовується HTTP-метод GET та відповідний шаблон представлення для відображення форми входу.

- `Logout()` (GET): Виконує вихід користувача з системи. Виконується виходу з аутентифікації, очищення зовнішнього кукі та перенаправлення на головну сторінку.

- `Login()` (POST): Обробляє відправлену користувачем форму входу в систему. Перевіряє правильність введених даних та виконує аутентифікацію користувача. При успішній аутентифікації перенаправляє користувача на головну сторінку.

- `Register()` (GET): Відображає сторінку реєстрації нового користувача. Використовується HTTP-метод GET та відповідний шаблон представлення для відображення форми реєстрації.

- `Register()` (POST): Обробляє відправлену користувачем форму реєстрації нового користувача. Перевіряє правильність введених даних та створює нового користувача у системі. При успішній реєстрації перенаправляє користувача на сторінку входу в систему.

Крім того, в коді представлення є два шаблони - `Login.cshtml` та `Register.cshtml`. Шаблон `Login.cshtml` містить форму для входу в систему, де користувач вводить свою електронну пошту та пароль. Шаблон `Register.cshtml` містить форму для реєстрації нового користувача, де користувач вводить електронну пошту, ім'я користувача, пароль та підтвердження паролю.

Цей контролер та його методи забезпечують основну функціональність для реєстрації та авторизації користувачів у додатку, з використанням ASP.NET Core MVC, інфраструктури Identity та бази даних MS SQL Server.

2.2 Головна сторінка

Головна сторінка додатку відображає доступні курси для реєстрації. Для цього використано модель `Course` із списком курсів, які передаються з контролера до представлення.

У коді представлення використано цикл `foreach` для перебору кожного курсу зі списку. Для кожного курсу відображається його назву, опис, максимальну ємність та відповідну дію.

Якщо користувач має роль "Student", а кількість зареєстрованих студентів на курс менше максимальної ємності, ми відображаємо форму для

реєстрації на курс. Форма містить приховані поля для ідентифікатора курсу та студента, а також поле для введення оцінки.

У разі, якщо кількість зареєстрованих студентів досягла максимальної ємності, ми відображаємо кнопку для перегляду результатів курсу.

Якщо користувач вже зареєстрований на курс, ми відображаємо вимкнену кнопку реєстрації.

У разі відсутності доступних курсів, виводимо повідомлення про їх відсутність.

Контролер `HomeController` містить декілька дій:

- `Index()`: Витягує список курсів з бази даних і передає його до представлення `Index.cshtml`.

- `Privacy()`: Відображає сторінку з політикою конфіденційності.

- `Error()`: Відображає сторінку з помилкою, якщо сталася помилка в додатку.

Цей код дозволяє користувачам переглядати доступні курси та реєструватися на них в залежності від їх ролі та наявності місць на курсах.

2.3 Функціонал студента

Клас `StudentController` відповідає за обробку запитів, пов'язаних з функціоналом студента. Даний контролер містить наступні методи:

- `Enroll(int courseId, string studentId, int score)`: Цей метод дозволяє студенту зареєструватися на додаткову дисципліну. Він отримує ідентифікатори курсу та студента, а також оцінку студента. Метод спочатку перевіряє наявність курсу та студента в базі даних. Якщо курс або студент не знайдені, відбувається перенаправлення на головну сторінку. Далі метод перевіряє, чи студент вже зареєстрований на цей курс, і чи є вільні місця на курсі. Якщо студент не зареєстрований на цей курс і є вільні місця,

створюється новий запис про реєстрацію (`Enrollment`) з відповідними даними, і кількість вже зареєстрованих студентів на курсі збільшується. Після цього дані зберігаються в базі даних. На завершення метод перенаправляє користувача на головну сторінку.

- `MyDisciplines()`: Цей метод відображає список додаткових дисциплін, на які зареєстрований поточний студент. Він отримує ідентифікатор поточного користувача, використовуючи ідентифікатор з контексту клеймів (Claims). Потім метод отримує дані з бази даних про реєстрації студента і додаткові дисципліни, до яких він зареєстрований, та передає ці дані у представлення.

- `Results()`: Цей метод відображає список студентів, які успішно пройшли хоча б одну з додаткових дисциплін. Метод отримує дані про студентів з бази даних, використовуючи відповідний запит, і передає ці дані у представлення.

Представлення `My Selected Disciplines` відображає таблицю з назвою, описом, оцінкою та статусом (пройдено/непройдено) кожної з додаткових дисциплін, на які зареєстрований поточний студент.

Представлення `Student Results` відображає таблицю з іменем та електронною поштою студентів, які пройшли хоча б одну з додаткових дисциплін.

2.4 Функціонал адміністратора

У контролері AdminController створено наступні методи:

- Index: Отримує всі курси з бази даних та відображає їх у вигляді списку на сторінці.

- Create: Відображає форму для створення нового курсу.

- [HttpPost] Create: Додає новий курс до бази даних після успішного заповнення форми.

- Edit: Відображає форму для редагування існуючого курсу з заданим ідентифікатором.

- [HttpPost] Edit: Зберігає зміни, внесені до курсу після редагування, в базі даних.

- Delete: Відображає підтвердження видалення курсу з заданим ідентифікатором.

- [HttpPost] Delete: Видаляє курс з бази даних після підтвердження видалення.

- Ratings: Відображає список студентів, які зареєструвалися на заданий курс, впорядкованих за середнім балом.

- SetMaxCapacity: Відображає форму для встановлення максимальної ємності заданого курсу.

- [HttpPost] SetMaxCapacity: Змінює максимальну ємність заданого курсу після успішного заповнення форми.

- Крім того, в коді також надані відповідні представлення (views) для відображення сторінок:

- Create.cshtml: Відображає форму для створення нового курсу.

- Delete.cshtml: Відображає підтвердження видалення курсу.

- Edit.cshtml: Відображає форму для редагування курсу.

- Index.cshtml: Відображає список курсів разом з можливостями редагування, видалення, перегляду рейтингів та зміни максимальної ємності.

- Ratings.cshtml: Відображає список студентів, зареєстрованих на заданий курс, впорядкованих за середнім балом.

- SetMaxCapacity.cshtml: Відображає форму для зміни максимальної ємності курсу.

Цей функціонал дозволяє адміністратору створювати, редагувати та видаляти курси, а також переглядати рейтинги студентів та змінювати максимальну ємність курсу.

Таким чином, використання платформи ASP.NET Core MVC, технології MS SQL Server дозволило створити застосунок для вибору студентами додаткових дисциплін, який відповідає визначеним раніше у 1 розділі вимогам.

РОЗДІЛ 3. РОБОТА ЗАСТОСУНКУ

3.1 Клієнтська частина сторінок реєстрації та авторизації

Для реалізації цієї функціональності, ми використовуємо React, який є однією з популярних бібліотек JavaScript для створення користувацьких інтерфейсів. Ми також використовуємо бібліотеку react-router-dom для навігації між сторінками додатку.

У нашому коді ми маємо дві компоненти - `Login` (Вхід) і `Register` (Реєстрація). Обидва компоненти мають форми, де користувачі можуть ввести свої дані.

Компонент `Login` містить поля для введення логіну та пароля. При зміні значень цих полів, ми використовуємо хук `useState`, щоб зберігати значення введених даних. При поданні форми, ми перевіряємо, чи обидва поля заповнені. Якщо так, ми виконуємо перенаправлення користувача на сторінку "/main" за допомогою хука `useNavigate`. Якщо форма не заповнена, ми можемо вивести повідомлення або здійснити необхідні дії.

Компонент `Register` працює аналогічно до компонента `Login`, але містить поля для введення логіну та пароля для реєстрації нового користувача. При поданні форми з коректно заповненими полями, також виконується перенаправлення на сторінку "/main".

Обидва компоненти також містять заголовки, кнопки та стилізацію за допомогою CSS-класів.

Ці компоненти можна використовувати як частину більшої системи аутентифікації і реєстрації користувачів. Вони дозволяють користувачам ввести свої дані та виконати відповідні дії на основі введених даних.

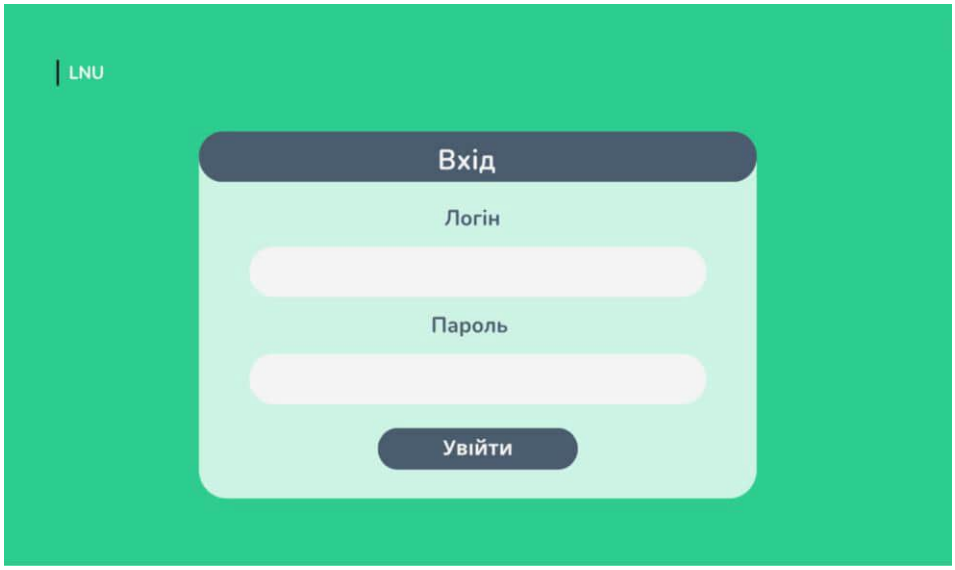

Рисунок 3.1.1 – Сторінка логіну

Реалізація клієнтської частини сторінок реєстрації та авторизації є важливим етапом розробки будь-якого веб-додатку, який вимагає ідентифікації користувачів. Цей функціонал допомагає забезпечити безпеку та персоналізацію веб-додатку, створюючи унікальні облікові записи для кожного користувача та забезпечуючи захист доступу до конфіденційних даних.

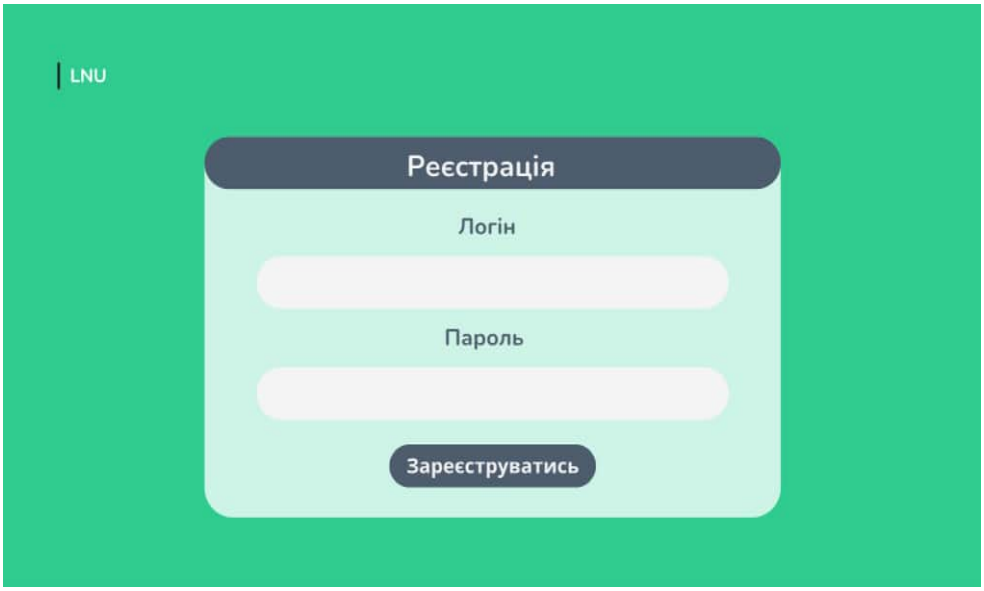

Рисунок 3.1.2 – Сторінка реєстрації

Наша реалізація базується на React і react-router-dom, але може бути адаптована до будь-якого фреймворку або бібліотеки для створення клієнтської частини додатків.

3.2 Клієнтська частина головної сторінки

При розгляді коду ми бачимо, що компонент `Home` має стилізацію за допомогою CSS-класів. Він складається з блоку `.home`, який містить заголовок, блок `.home__block` та кнопки для реєстрації та входу.

У блоку `.home__block` знаходиться `.home__wrapper`, який містить сам заголовок та кнопки. Заголовок представлений у вигляді `<h1>` з текстом "LNU Запис на дисципліни". Кнопки реалізовані за допомогою компонента `Link` з пакету `react-router-dom`, що дозволяє створити посилання на інші сторінки в додатку.

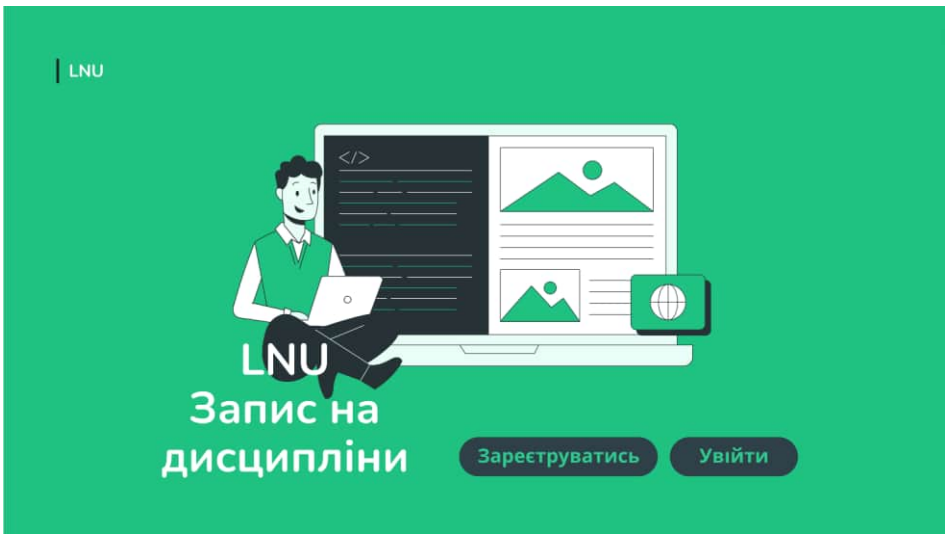

Рисунок 3.2.1 – Головна сторінка

Перша кнопка має посилання на сторінку реєстрації (`/register`). Клас кнопки `.home__btn-reg` задає візуальний стиль для цієї кнопки. Друга кнопка має посилання на сторінку входу (`/login`). Клас кнопки `.home__btn` задає візуальний стиль для цієї кнопки.

Цей компонент дозволяє користувачам зареєструватись або увійти в систему, використовуючи відповідні кнопки.

Важливо зауважити, що для коректної роботи компонента `Link` та навігації між сторінками, потрібно мати налаштовану систему маршрутизації (наприклад, використовуючи `react-router-dom`).

3.3. Клієнтська частина сторінок для студента

Ми розглянемо клієнтську частину сторінки для студента (компонент `StudentAccount`). Цей компонент містить інформацію про дисципліни, які вивчає студент.

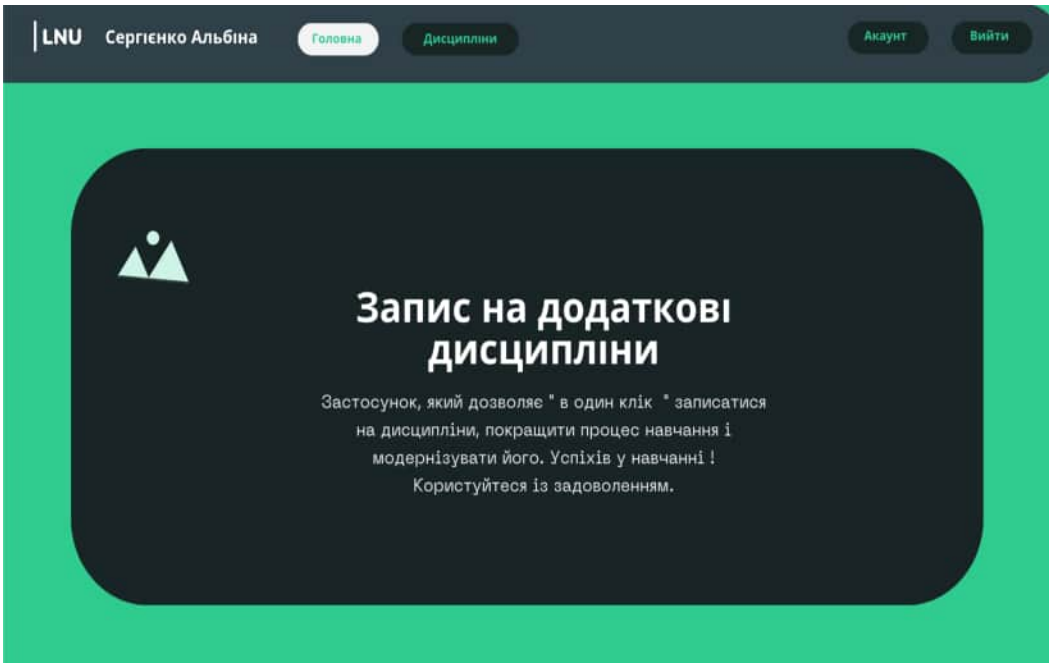

Рисунок 3.3.1 – Головна сторінка студента

Компонент `StudentAccount` має стилізацію за допомогою CSS-класів. Він складається з блоку `.student-account`, який містить заголовок "Мої дисципліни", інформацію про курс та семестр, а також список дисциплін.

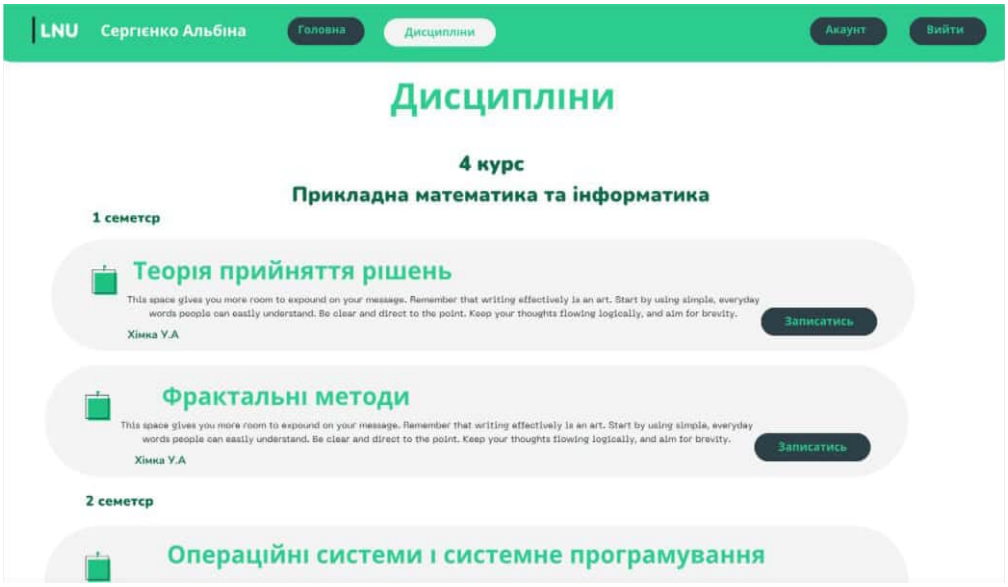

Рисунок 3.3.2 – Сторінка дисциплін для студента

У компоненті `StudentAccount` є дві секції дисциплін. Кожна дисципліна представлена блоком `.discipline-item`. В цьому блоку міститься інформація про дисципліну, включаючи назву, опис, викладача та кнопку.

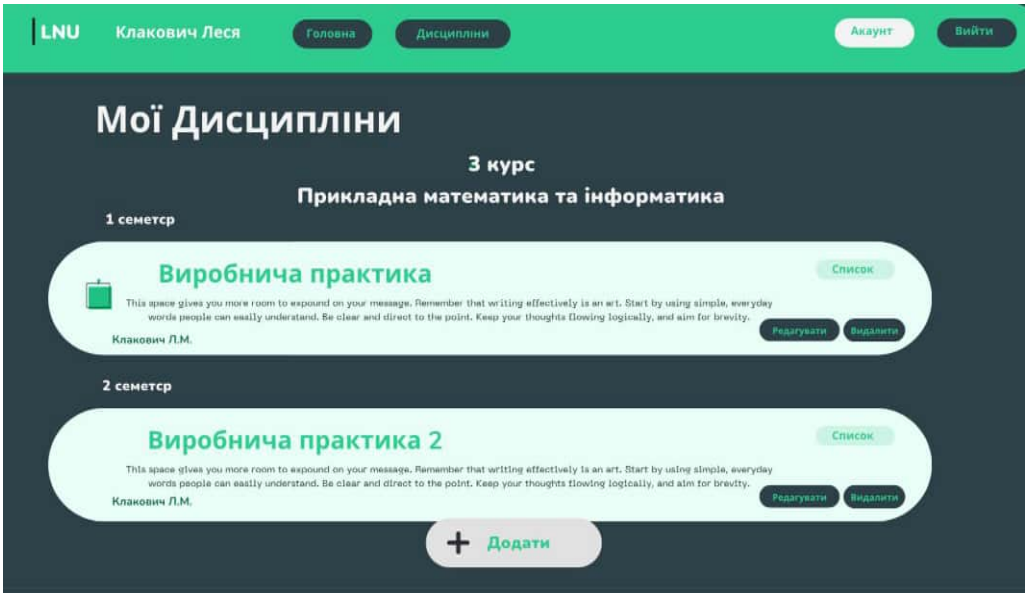

Рисунок 3.3.3 – Сторінка вибраних дисциплін студента

У першому блоку дисципліни ми бачимо іконку `GrSchedules` з пакету `react-icons/gr`, яка позначає розклад занять для даної дисципліни. Назва дисципліни відображається у блоці `.discipline-item__title`. Опис

дисципліни розміщений у блоці `.discipline-item__descr`. Викладач дисципліни зазначений у блоку `.discipline-item__name`.

У блоку `.discipline-item__wrapper-btn` знаходиться кнопка з іконкою `BsFillCheckSquareFill` з пакету `react-icons/bs`. Ця кнопка може використовуватись для позначення того, що студент вже обрав цю дисципліну або виконав певні дії.

3.4 Клієнтська частина сторінки адміністратора та списку студентів

Ми розглянемо клієнтську частину сторінки адміністратора (компонент `TeacherAccount`). Цей компонент містить інформацію про дисципліни, які викладає адміністратор.

Компонент `TeacherAccount` також має стилізацію за допомогою CSS-класів. Він складається з блоку `.teacher-account`, який містить заголовок "Мої Дисципліни", інформацію про курс, факультет та семестр, а також список дисциплін.

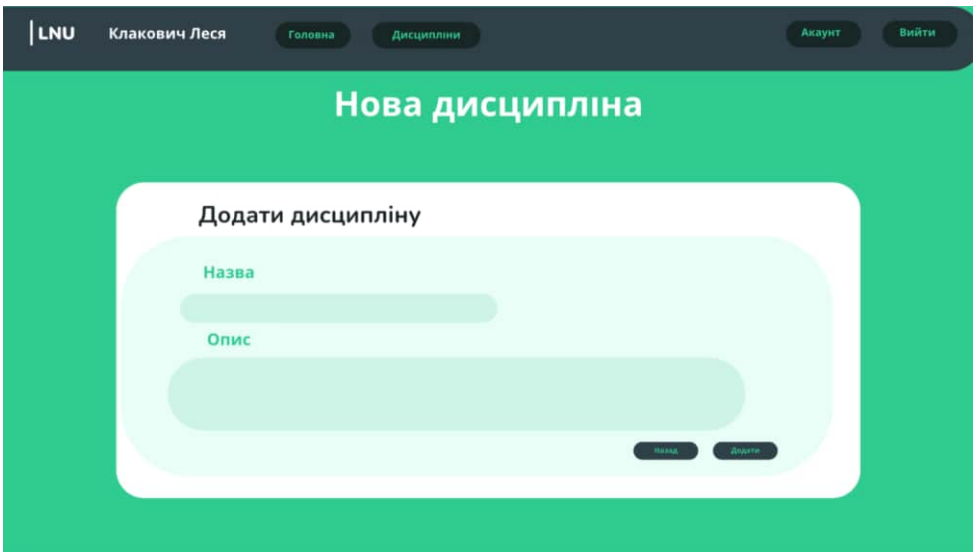

Рисунок 3.4.1 – Форма додавання дисципліни

У компоненті `TeacherAccount` є дві секції дисциплін. Кожна дисципліна представлена блоком `.discipline-item`. В цьому блоку міститься інформація про дисципліну, включаючи назву, опис та викладача.

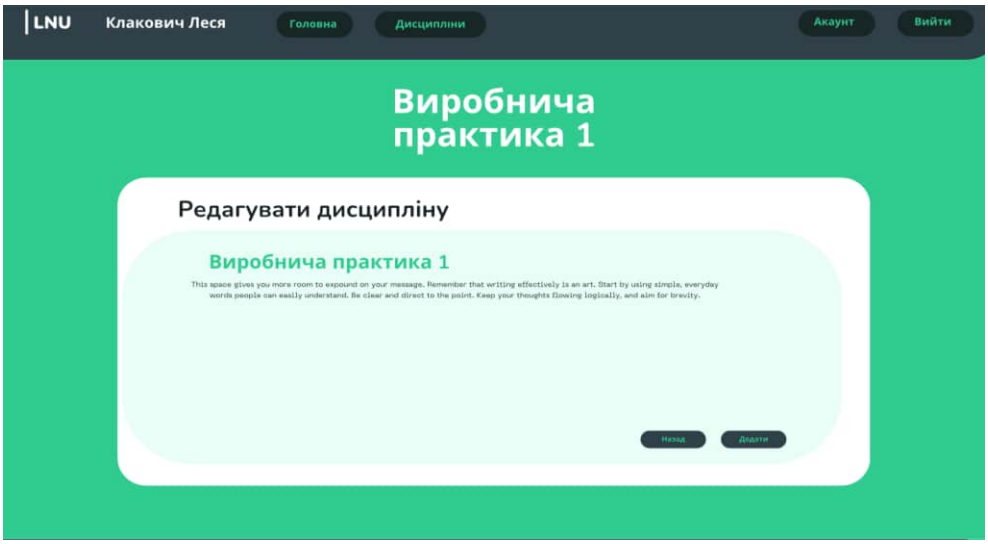

Рисунок 3.4.2 – Форма редагування дисципліни

У кожному блоку дисципліни ми бачимо іконку `GrSchedules` з пакету `react-icons/gr`, яка позначає розклад занять для даної дисципліни. Назва дисципліни відображається у блоці `.discipline-item__title`. Опис дисципліни розміщений у блоці `.discipline-item__descr`. Викладач дисципліни зазначений у блоку `.discipline-item__name`.

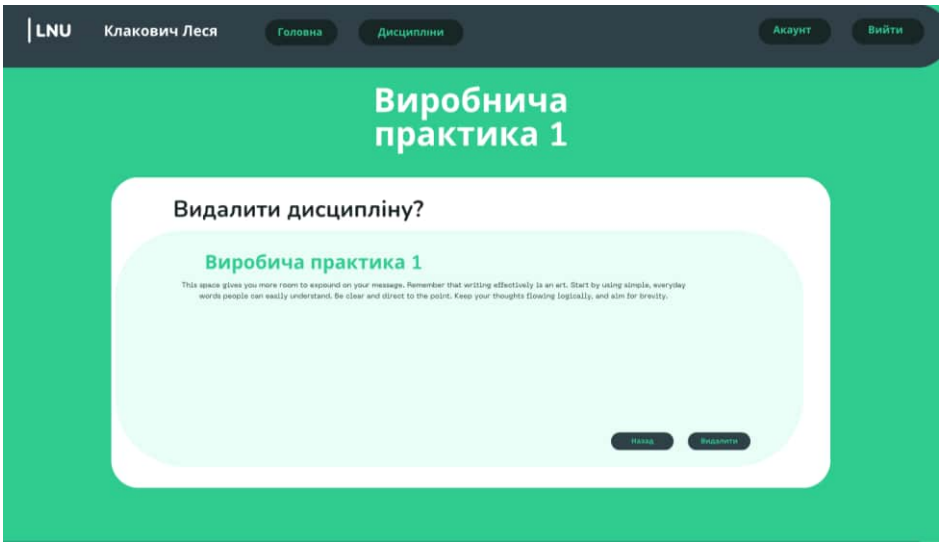

Рисунок 3.4.3 – Форма видалення дисципліни

Кожен блок дисципліни має також блок `.teacher-account__wrapper-btn`, який містить три кнопки. Кнопка з класом `.teacher-account__list` відображає посилання на сторінку "Список". Кнопка з класом `.discipline-item__btn.r` відображає посилання на сторінку "Редагувати". Кнопка з класом `.discipline-item__btn` відображає посилання на сторінку "Видалити".

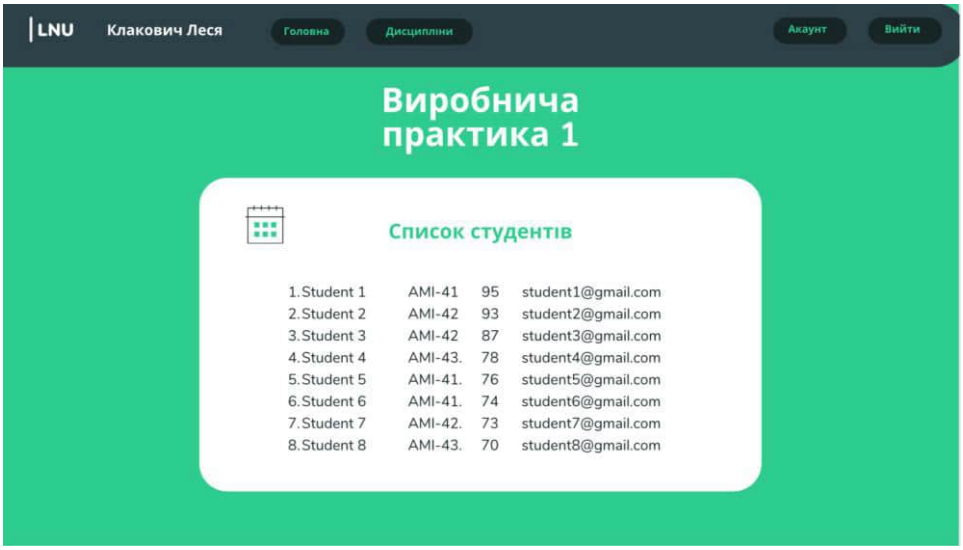

Рисунок 3.4.4 – Список студентів на певній дисципліні

У кінці сторінки адміністратора є блок `.absolute`, який містить кнопку з класом `.absolute__btn`. Ця кнопка містить іконку `CgMathPlus` з пакету `react-icons/cg` та надпис "Додати". Вона може використовуватись для переходу на сторінку додавання нової дисципліни.

Таким чином, у системі передбачено усі функціональні елементи, які визначені у підрозділі 1.2.

ВИСНОВКИ

У даній дипломній роботі було розроблено додаток для реєстрації на додаткові дисципліни, який надає функціональні можливості для управління курсами та рейтингами студентів. Під час розробки було проведено аналіз різних методів та підходів до реалізації додатку, враховуючи сучасні технології та інструменти.

В ході роботи був створений дизайн додатку, включаючи структуру бази даних, архітектуру контролерів та представлень. Було розроблено контролер AdminController з відповідними методами для створення, редагування та видалення курсів, а також для відображення рейтингів студентів та зміни максимальної ємності курсу. Були реалізовані відповідні представлення для відображення сторінок із відповідними формами та даними.

Результатом роботи є функціональний додаток, який дозволяє адміністратору ефективно керувати курсами та рейтингами студентів. Додаток має потенціал для подальшого розширення та вдосконалення, наприклад, можливість реєстрації студентів на курси, розширення функціоналу для викладачів тощо.

У майбутньому планується розширення функціоналу додатку, додавання нових можливостей та покращення користувацького інтерфейсу. Також можна розглянути можливість розробки мобільного додатку або підтримки інших платформ для більш широкого охоплення користувачів.

У цілому, дана дипломна робота успішно реалізувала поставлені цілі та завдання, дозволяючи ефективно керувати реєстрацією на додаткові дисципліни та оцінками студентів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Microsoft Documentation: ASP.NET Core MVC. [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/mvc/?view=aspnetcore-5.0
2. Microsoft Documentation: Entity Framework Core. [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/ef/core/
3. Microsoft Documentation: C# Programming Guide. [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/
4. Microsoft Documentation: ASP.NET Core Razor Pages. [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-5.0
5. Microsoft Documentation: MS SQL Server. [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/sql/?view=sql-server-ver15