Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота магістра

на тему: Створення інтерактивної освітньої вебплатформи як засобу оптимізації навчального процесу

Виконала: студентка групи К23-2м
Спеціальність 122 «Комп'ютерні науки»

Лісова І.О.
(прізвище та ініціали)

Керівник к.ф.-м.н., доцент, Рудянова Т.М.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент Університет митної справи та фінансів
(місце роботи)

доцент кафедри транспортних технологій
та міжнародної логістики
(посада)

к.т.н., доц. Разгонов С.А.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2025

АНОТАЦІЯ

Лісова І.О. Створення інтерактивної освітньої вебплатформи як засобу оптимізації навчального процесу.

Дипломна робота на здобуття освітнього ступеня магістр за спеціальністю 122 «Комп'ютерні науки». – Університет митної справи та фінансів, Дніпро, 2025.

Метою магістерської роботи є створення та оцінка ефективності впровадження освітньої вебплатформи в якості засобу оптимізації процесу здобування освіти на основі функціональних вимог, які є необхідними для учасників навчального процесу.

Об'єктом дослідження є оптимізація навчального процесу, завдяки використанню освітньої вебплатформи.

Предметом дослідження є оцінка спрощення процесу навчання шляхом впровадження інтерактивної освітньої вебплатформи.

Магістерська робота присвячена розробці інтерактивної освітньої вебплатформи в середовищі Visual Studio 2022, як засобу оптимізації навчального процесу.

Актуальність роботи обумовлена важливістю використання в процесі навчання сучасних інтерактивних автоматизованих інструментів, особливо в період збільшення дистанційного формату навчання. Досліджено функціональні особливості необхідні для підвищення якості комунікації учасників освітнього процесу. Розглянуто програмні рішення, які вже існують для проведення навчання. Проаналізовано, які комп'ютерні технології доречно застосувати для виконання поставленого завдання. Визначено методику інтерактивного навчання для підвищення якості взаємодії користувачів (викладачів та студентів). На основі описаного методу розроблено програмне забезпечення (ПЗ), – інтерактивна освітня веб платформа. Також подано шляхи для можливого майбутнього масштабування функціональних можливостей.

Ключові слова: інтерактивність, вебплатформа, оптимізація, ASP.NET Core, функціонал, адаптивність, користувач.

ANNOTATION

Lisova I.O. Creation of an interactive educational web platform as a means of optimizing the learning process.

Master's thesis for obtaining the degree of Master in the specialty 122 "Computer Science". – University of Customs and Finance, Dnipro, 2025.

The goal of the master's thesis is to create and evaluate the effectiveness of implementing an educational web platform as a means of optimizing the educational process based on the functional requirements necessary for participants in the learning process.

The object of the research is the optimization of the educational process using an educational web platform.

The subject of the research is the assessment of simplifying the learning process through the implementation of an interactive educational web platform.

The master's thesis is dedicated to the development of an interactive educational web platform in the Visual Studio 2022 environment as a means of optimizing the educational process.

The relevance of the work is determined by the importance of using modern interactive automated tools in the educational process, especially during the increasing prevalence of distance learning formats. The study examines the functional features necessary to enhance the quality of communication among participants in the educational process. Existing software solutions for conducting education are reviewed. An analysis is conducted to determine which computer technologies are appropriate for achieving the set task. A methodology for interactive learning is defined to improve the quality of interaction between users (teachers and students). Based on this methodology, software—a web-based interactive educational platform—has been developed. Additionally, pathways for potential future scaling of functional capabilities are proposed.

Keywords: interactivity, web platform, optimization, ASP.NET Core, functionality, adaptability, user.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ШІ – штучний інтелект

ПЗ – програмне забезпечення

ООП – об’єктно орієнтоване програмування

MVC (Model-View-Controller) – архітектурний шаблон заснований на розбитті на три компоненти (моделі, представлення, контролера)

EF Core (Entity Framework Core) – об’єктно-реляційний фреймворк для роботи з базами даних

ВСТУП

В умовах збільшення викликів, які не дають змоги проводити очне навчання та стрімкого розвитку інформаційних технологій освіта перебуває в процесі кардинальних трансформацій. Для проведення навчального процесу виникає необхідність впровадження інтерактивних засобів, які забезпечують доступ до навчальних матеріалів та ефективну комунікацію. Інтерактивні освітні вебплатформи допомагають ефективно надавати освіту в дистанційному чи гібридному форматі. Науковці та провідні навчальні заклади акцентують увагу на необхідності розвитку таких платформ. Наприклад, у своїй статті «Education in a post-COVID world: Nine ideas for public action» організація UNESCO визначила дев'ять кроків, які треба виконати для покращення освіти в період після COVID, одним із пунктів є забезпечення освітніх технологій з відкритим доступом для вчителів та учнів [1]. Існує значна кількість цих платформ, але є низка проблем: недостатня інтерактивність, складність використання, велика вартість і т.п.

Обрана тема є *актуальною*, оскільки дослідження та розробка платформи, яка враховує функціональні потреби потенційних користувачів, сприятиме вирішенню проблеми із недостатньою взаємодією між викладачами та студентами. Загалом, дослідження дозволить оптимізувати освітній процес і підвищити якість навчання.

Об'єктом дослідження є процес оптимізації взаємодії між викладачами та студентами. Основними характеристиками об'єкта виступають: використання цифрових інструментів, урахування функціональних потреб учасників навчального процесу, створення спеціалізованої вебплатформи.

Предметом дослідження є оцінка спрощення процесу комунікації в освітньому процесі з використанням функціональних особливостей та механізмів, які притаманні освітній вебплатформі. Тобто охоплюється аналіз ключових компонентів освітньої вебплатформи в дистанційній і гібридній формах навчання.

Метою дослідження є створення та оцінка ефективності впровадження освітньої вебплатформи в якості засобу оптимізації процесу здобування освіти на основі функціональних вимог, які є необхідними для учасників навчання.

Для досягнення поставленої мети були сформовані такі *завдання*:

1. Розглянути сутність вебплатформ та оцінити переваги, недоліки та перспективи розвитку освітніх вебплатформ.

2. Визначити необхідні вимоги до функціоналу освітніх веб платформ.

3. Вибір середовища та технологій для розробки.

4. Розробити освітню вебплатформу на основі визначених вимог.

Для виконання поставлених завдань застосовувалися такі *методи*:

1. Аналіз і синтез було застосовано для вивчення наукових праць та сучасних підходів до створення інтерактивних вебплатформ. Аналіз дозволив визначити ключові функціональні потреби освітніх платформ, а синтез – об’єднати їх у єдиний розроблений продукт.

2. Системний метод – для розробки архітектури вебплатформи з урахуванням взаємозв’язків між основними компонентами.

3. Порівняння передбачало проведення порівняльного аналізу вебплатформ, які існують для проведення освітнього процесу, виділення їх переваг та недоліків.

4. Індуктивний метод було використано для узагальнення окремих досліджень і практик у галузі дистанційного та гібридного навчання.

5. Дедуктивний метод було використано щодо розробки власного рішення для оптимізації освітнього процесу.

Наукова новизна полягає в наступному: в проєкті реалізований алгоритм автоматичного додавання студентів до чатів академічних груп, що враховує як вже зареєстрованих, так і майбутніх здобувачів освіти. Досить складно в сучасних умовах продемонструвати наукову новизну, тому було показано власне бачення та акцентовано на механізмі, який забезпечує безперервність та актуальність складу учасників, що сприяє оптимізації процесу комунікації.

Практичне значення результатів дослідження полягає в розробці альтернативної освітньої вебплатформи та сприянню для розширення наукових знань у сфері розробки вебплатформ для освітнього середовища. Запропоновані алгоритми та методи їх реалізації можуть бути використані як основа для розробки подібних систем в університетах, коледжах та освітніх центрах.

Особистий внесок автора полягає у розробці концепції та реалізації функціоналу освітньої платформи, що забезпечує ефективну комунікацію між викладачами та студентами. Перевагами авторської розробки є:

– реалізація автоматизованого механізму додавання студентів до чатів академічних груп;

– застосування архітектурного шаблону MVC, що спростить майбутню підтримку та масштабування;

– використання сучасних технологій, таких як Razor-синтаксис і фреймворк Bootstrap;

– впровадження модульного підходу до побудови функціоналу, що забезпечує гнучкість і можливість подальшого розширення функціональних можливостей платформи.

Апробація результатів магістерської роботи. Одні з результатів магістерського дослідження були оприлюднені у вигляді тез на конференції та включені до збірника матеріалів «Економіко-правові та управлінсько-технологічні виміри сьогодення: молодіжний погляд». Тема публікації: «Оцінка продуктивності роботи з базою даних з використанням Entity Framework Core при розробці застосунків» [2].

Структура магістерської роботи складається із трьох розділів:

Розділ 1. Теоретичні засади оптимізації навчального процесу засобами вебплатформ. У цьому розділі розглянуто сутність вебплатформ, переваги та недоліки використання освітніх веблатформ та перспективи їх розвитку.

Розділ 2. Аналіз технологій для створення освітньої вебплатформи. Цей розділ передбачає вибір середовища розробки, вибір технологій для розробки front-end та back-end.

Розділ 3. Розробка інтекрактивної освітньої вебплатформи. У розділі сформовано архітектуру вебплатформи, архітектуру бази даних, описано роботу з GitHub і реалізацію функціоналу та перспективи масштабування функціоналу.

Робота складається зі вступу, трьох розділів, висновків; містить 74 сторінки тексту, 33 рисунки, 1 додаток. Список використаних джерел включає 6 сторінок.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ ЗАСАДИ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Концептуальні засади та сутнісні характеристики вебплатформ

Наприкінці 20 століття запустився незворотній процес діджиталізації, тобто для отримання прибутку чимало установ почали використовувати цифрові технології при побудові власних бізнес-моделей. Згодом, з появою Інтернету та смартфонів діджиталізація стала масовим явищем: штучний інтелект (ШІ), Big Data, інтернет речей, машинне навчання, хмарні технології [3]. Через такий активний розвиток було створено декілька різновидів вебрішень, серед яких вебсайти, вебзастосунки, вебплатформи.

Основними перевагами використання вебрішень є:

– доступ із будь-якого місця розташування, де є Інтернет-з’єднання;

– можливість працювати з різних операційних систем та девайсів;

– масштабовані та гнучкі параметри хостингу.

Кожне рішення має певні особливості, технічну структуру та мету. Вебсайт складається з однієї чи декількох вебсторінок (єдиний документ представлений браузером). Вебзастосунок – це вебсайт, який надає ще певні функціональні можливості. Тобто крім перегляду інформації, з’являється певна взаємодія користувача (заповнення певної інформації, завантаження файлів, редагування інформації і т.п.). Вебплатформа має більш розширений функціонал, який зазвичай передбачає взаємодію користувачів між собою [4].

Таким чином всі ці рішення показують еволюцію вебрішень. Початок відбувся в 1990 році, коли Тім Бернерс-Лі запровадив три основи, на яких базується мережа: HTML, HTTP, URI/URL. Поява перших браузерів (наприклад Netscape) дала початок Web 1.0, який слугував тільки для перегляду інформації (1994-2001). Поява Web 2.0 знаменувала появу соціальних мереж та контенту, який міг створювати користувач в Інтернеті (а не тільки веброзробник, як при Web 1.0).

Вебпрограми почали впроваджувати все більше функцій, таких як здійснення платежів, трансляції онлайн, відеодзвінки й т.п. З’являлася все більша кількість фреймворків для frontend. На відміну від Web 2.0, Web 3.0 є децентралізованим, тобто побудований навколо блокчейнів (децентралізованих однорангових вузлів). Також Web 3.0 удосконалений штучним інтелектом і машинним навчанням, що дозволяє розподілити дані з врахуванням особистих потреб користувача [5]. Таким чином, вебтехнології постійно розвиваються, тому веброзробники продовжуватимуть формувати майбутнє вебрішень шляхом розвитку нових можливостей та вирішенням викликів, які виникають.

Як було зазначено одна із основ мережі Інтернет є HTTP, який є протоколом передачі даних на основі якого все працює. Завдяки цьому протоколу є можливість взаємодіяти з браузером. Взагалі HTTP розшифровується, як HyperText Transfer Protocol – протокол передачі гіпертексту (текстові документи, які мають посилання на інші текстові документи), але вже з можливістю передавати будь-які формати даних. Цей протокол працює таким чином: клієнт (зазвичай веббраузер виступає в цій ролі) відправляє запит на сервер, де програма обробляє його, генерує відповідь і надсилає її назад. Для того, щоб зробити протокол HTTP більш безпечним було створено його розширення HTTPs. Це розширення додає можливість шифрування даних які передаються, на відміну від HTTP який передає дані у відкритому вигляді. Розшифровується, як HyperText Transfer Protocol Secure – безпечний протокол передачі гіпертексту. Ця безпека є завдяки об’єднанню HTTP та криптографічного протоколу TLS. HTTPs передбачає, що клієнт та сервер будуть використовувати тимчасовий ключ для шифрування та розшифрування повідомлення (рис. 1.1). Кожного сеансу буде генеруватися новий ключ. Може бути використано метод синхронного шифрування, де буде застосовано один ключ для шифрування та розшифрування. Асиметричне шифрування передбачає використання двох ключів: відкритого та закритого. Відкритим ключем володіє сервер, але він вільно може передаватися клієнтам для шифрування повідомлень. Закритим ключем володіє сервер і він зберігається в секреті, щоб потім бути використаним для розшифрування. Хоч застосування асиметричного шифрування є безпечнішим

варіантом, все одно часто використовують симетричне шифрування через швидкість (асиметричне потребує більше обчислювальних ресурсів). В роботі HTTP використовує порт 80, а HTTPs – 443, але все рівно основною відмінністю є саме безпека [6].

Рисунок 1.1 – Схематичне встановлення безпечного з'єднання через HTTPs

Джерело: розроблено автором на основі [6]

Ще однією важливою складовою є URL та URI адреси. Ці адреси є способом пошуку в Інтернеті. URI – самостійний ресурс або ресурс з протоколом, а URL – підтип URI, що містить ресурс і протокол. Тобто всі URL є URI, але не всі URI є URL [7]. Якщо подається ім'я ресурсу разом із протоколом, то краще все таки називати URL.

Архітектура вебплатформ має чотири ключові компоненти:

1. Компоненти програми інтерфейсу – вебсторінки, на яких відображається основні розділи. Це можуть бути меню, інформаційні панелі і т.п. Зазвичай компоненти інтерфейсу орієнтовані на взаємодію з користувачем.

2. Структурні компоненти відповідають за процес розробки: рівень презентації, бізнес-рівень, рівень збереження даних. Рівень презентації – компоненти інтерфейсу, які є доступними для користувачів і підтримують зв'язок

із системою. На цьому рівні можуть застосовувати HTML, CSS, JavaScript, фреймворки Angular, React. Для бізнес-рівня, тобто прикладного рівня, характерним є приймання запитів від користувачів, обробка і визначення рівня доступу користувачів до даних. Цей прикладний рівень (сервер вебплатформи) може бути створений із використанням PHP, Python, Java, Ruby, .NET, Node.js. Рівень збереження даних є базою даних, де зберігаються необхідні для коректної роботи дані.

3. Наскрізний код – компонент для вирішення питання безпеки, зв'язку, управління.

4. Сторонні інтеграції, якими можна розширити функціональність (ці фрагменти коду називають API). Прикладами таких інтеграцій можуть слугувати платіжні шлюзи, GPS-карти і т.п [8].

Через різні причини можуть виникати помилки. Щоб розуміти причину некоректного запиту HTTP, або чи було виконано запит успішно є спеціальні коди стану відповіді HTTP. Існують певні категорії, на які розбивають ці коди: інформаційні (100-199); успішні (200-299); перенаправлення (300-399); клієнтські помилки (400-499); серверні помилки (500-599).

Кожен із цих кодів має свій опис та означає конкретний результат (успішний або некоректний). Але є можливість створити спеціальну нестандартну відповідь (код) для ПЗ, якщо є така потреба. Є коди більш поширені у використанні та ті, які рідше зустрічаються (рис. 1.2).

Рисунок 1.2 – Найчастіше використовувані статус коди

Вебплатформи підлягають певній класифікації:

1. Платформи електронної комерції – механізм, який працює в онлайн магазині. Тобто, це платформа, яка виконує всі функції магазину, але в онлайн (від перегляду продукції до обробки платежів та здійснення замовлень) [9]. Прикладами таких платформ є Amazon та Shopify.

2. Платформи соціальних мереж – онлайн платформи, які дають можливості для створення, поширення, обміну інформацією, фото [10]. Популярними прикладами серед користувачів є Instagram, Facebook, LinkedIn і т.п.

3. Освітні платформи – цифрове середовище, яке має на меті надати можливість здійснювати управління та надання освіти в онлайн форматі. Ці платформи дозволяють отримувати знання шляхом взаємодії, поширенню ресурсів та відслідковуванню прогресу вивчення [11]. Coursera, Moodle є одними з прикладів таких онлайн платформ.

4. Бізнес-платформи – це цифрові платформи, які є показником певного бренду. Тобто, це своєрідна демонстрація роботи певного бізнесу в онлайн форматі та можливість працювати з колегами та потенційними клієнтами без обмежень місцезнаходження [12]. Наприклад, Slack та Microsoft Teams можна віднести до цієї категорії.

Вебплатформам необхідно постійно адаптуватися до змінних навантажнь, тобто бути масштабованими та гнучкими. Для цього можуть використовуватися хмарні технології (Amazon Web Services, Microsoft Azure, Coogle Cloud), мікросервіси, контейнеризація. Хмарні технології допомагають мати доступ до сховищ, файлів, ПЗ та серверів з використанням власних пристроїв і наявності підключення до Інтернету. Робота постачальників хмарних обчислень полягає в збереженні та обробці даних в певному місці, яке є відділеним від кінцевих користувачів. Основна перевага використання цих технологій – більш економічно вигідний та гнучкий спосіб масштабування [13]. Використання мікросервісів дозволяє використати стиль, який представляє програму у вигляді певної кількості служб [14]. Таким чином, виникає можливість масштабувати кожну службу окремо, що призводить до полегшення всього процесу. Контейнеризація – використання

ізольованих контейнерів для створення та тестування програм. Цей спосіб допомагає швидше запускатися та масштабуватися завдяки ізоляції, яка надає можливість запускати певну кількість контейнерів в одній системі без небезпеки їх впливу один на одного [15].

Одним із ключових аспектів роботи вебрішень є хостинг. Для того, щоб користувачі могли користуватися вебплатформою через Інтернет потрібен якісний хостинг, який забезпечить наявність серверів та ресурсів (рис. 1.3). Існує декілька типів хостингу: вебхостинг, хмарний хостинг, серверний хостинг, VPS-хостинг. Різні види мають свої переваги та недоліки в місці зберігання, продуктивності, вартості [16].

Рисунок 1.3 – Послуги вебхостингу

Джерело: [17]

Для інтерактивної вебплатформи, як і для будь-якого програмного продукту, який взаємодіє із звичайними користувачами необхідна якісна візуалізація. Для цієї мети існує вебдизайн – процес створення вебплатформи, – в той час як розробка сайту – це вже створення функціональності. Якісний дизайн сайту допомагає залучити більшу кількість користувачів та покращити їх враження від використання платформи [18].

Важливим аспектом популярності вебплатформ є їх економічна значущість. Вебплатформи дозволяють надавати послуги та продукти по всьому світу. Це дозволяє підприємствам розповсюджувати свою діяльність по світу без необхідності відкривати фізичні точки в тій чи іншій країні. Але також це розширює

доступ користувачам до цих товарів і послуг. Наприклад, було майже 300000 сторонніх продавців, які приймали участь у платформі Amazon's Marketplace та експортувати товари до інших країн у 2017 році [19]. Вебплатформи є не тільки інформаційними та комунікаційними потоками в Інтернеті, а й ключовими інструментами цифрової торгівлі. Все більше громадян Європейського Союзу використовують онлайн платформи для повсякденного життя. Значне прискорення цього процесу відбулося через пандемію COVID-19, коли через карантин, а отже і соціальне дистанціювання люди почали більше часу проводити в Інтернеті (рис 1.4). У 2020 році вартість ста найкращих платформ світу зросла на 40% (у період з січня по жовтень 2020 року) до 10,5 трлн євро [20].

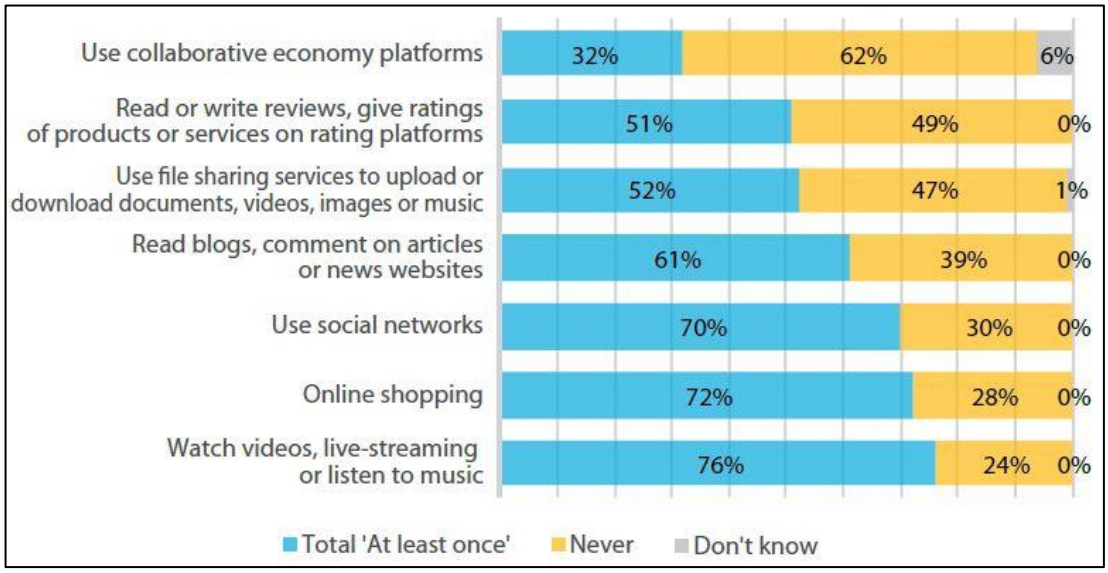

Рисунок 1.4 – Використання онлайн платформ громадянами ЄС у 2020 році

Джерело: [20]

В багато вебплатформ впроваджують новітні технології: ШІ, блокчейн, машинне навчання, великі дані. Це значно впливає на функціонування. Наприклад, ШІ допомагає у створенні персоналізованих рекомендацій для користувачів (в залежності від сфери, на яку орієнтована вебплатформа). У справі захисту даних та зменшення ризику шахрайства допомагає блокчейн. Використання машинного навчання надає можливість покращувати алгоритми на основі зібраних даних. Для більш ефективної взаємодії та аналізу поведінки користувачів можуть використовуватися великі дані. Використання цих інноваційних рішень в платформі

не є обов’язковим, але може допомогти персоналізувати та значно автоматизувати процеси.

Найважливішим аспектом вебплатформ є користувацький досвід. Цей досвід може бути позитивним і негативним, але для успіху платформи – має бути позитивним. Користувацький досвід це не лише те, як виглядає цифровий продукт. Це про відчуття людини, про легкість у використанні та ступеню відповідності очікуванням користувача в процесі виконання завдань. Для покращення досвіду користувачів необхідно провести дослідження цільової аудиторії, оптимізувати швидкість завантаження сторінок [21]. Вебплатформа буде максимально ефективною при забезпеченні позитивного користувацького досвіду та його покращенню на основі відгуків користувачів.

1.2 Аналіз публікацій щодо ефективності навчання з використанням інтерактивних інструментів

Перед тим, як почати розглядати сам процес оптимізації навчального процесу цифровими рішеннями, необхідно розглянути, що собою являє навчальний процес. Всі дії, які є частиною повсякденної роботи закладів освіти називаються навчальним процесом:

– процеси навчання та методика навчання;

– комунікація викладачів із студентами;

– адміністрація факультету;

– фінансово-адміністративна система.

Всі ці елементи навчання впливають на його якість та ефективність. Завдяки вебплатформам і іншим цифровим рішенням є можливим обмін даними, практична та динамічна робота, яка буде спрямована на досягнення спільних цілей при проведенні заходів щодо здобуття освіти [22].

Є низка рішень націлених на оптимізацію навчального процесу:

– автоматизовані системи управління;

– персоналізовані навчальні платформи;

– онлайн ресурси;

– інструменти для комунікації.

Автоматизовані системи управління – централізоване сховище для зберігання та упорядкування великої кількості інформації, яка стосується учнів. Цими записами можуть бути відвідуваність, оцінки, особисті дані. Завдяки таким системам, викладачі та адміністратори мають можливість отримати доступ до інформації про успішність кожного студента, що дає змогу мати персоналізовану підтримку [23]. Тобто автоматизовані системи управління допомагають організувати освітній процес (електронні журнали, платформи для керування курсами). Прикладом такої системи є Moodle – система, яка надає можливості викладачам створювати власний вебсайт з динамічними курсами. Moodle підходить для студентів, викладачів, адміністраторів через здатність до кастомізації [24].

Персоналізовані навчальні платформи є програмними рішеннями, які пропонують користувачам отримати досвід вивчення матеріалу шляхом персоналізованого підходу для досягнення індивідуальних цілей. Для того, щоб виконувати ці функції застосовуються аналіз даних, ШІ і т.п. [25]. Прикладами таких платформ є Coursera, Khan Academy. Coursera – глобальна онлайн платформа, яка забезпечує курсами та іншими формами навчання від провідних університетів та компаній, таких як Standfort, University of Colorado Boulder, Google, IBM, Microsoft, Meta та інші [26]. Khan Academy є платформою, яка містить вправи, відеоуроки та персоналізовані дошки, які показують студентам їх прогрес. Інформація, яка розміщена охоплює знання від дитячого садочка до коледжу з різних наук: математика, читання, історія, мистецтво, інформатика, економіка, природничі науки тощо [27]. Загалом персоналізовані навчальні платформи адаптують навчальний процес під індивідуальні потреби кожного студента та полегшують роботу викладачів.

Онлайн ресурси – навчальні матеріали, тести, відео, які доступні через Інтернет. Наприклад, відкриті курси, бібліотеки, YouTube.

Інструменти для комунікації: месенджери, чати, інтерактивні платформи для спілкування (Microsoft Teams, Zoom, Google Meet).

Деякі програмні освітні рішення можна віднести до декількох категорій. Наприклад, Google Classroom поєднує ознаки автоматизованих систем управління та інструментів для комунікації. Google Classroom безкоштовна вебплатформа, яка була розроблена Google, де викладачі можуть викладати матеріал і завдання та оцінювати учнів. Сервіс дозволяє взаємодіяти з Google Docs, Sheets, Slides [28]. Ця платформа поєднує функції організації навчального процесу та взаємодію між викладачами та студентами.

Важливо розуміти, наскільки ефективним є впровадження засобів інтерактивного онлайн навчання. Наприклад, онлайн курс від Neurobiology вирішили проблему дефіциту взаємодії в онлайн курсах саме впровадженням інтерактивних інструментів. Крім того, в 2020 році вони провели дослідження задоволеності та рівню засвоєння знань студентами шляхом порівняння звичайного вивчення, – та з використанням інтерактивних цифрових засобів. Майже 2/3 учнів підтвердили, що інтерактивні інструменти в онлайн курсі допомогли їм досягти цілей у навчанні. Найбільш корисним серед нововведень вони виділили опитування, які допомагали засвоїти основні моменти в курсі. Великої різниці між двома начальними методами в контексті кінцевого результату іспиту виявлено не було. Всі студенти які проходили курс онлайн здали іспит, а за традиційною системою навчання було три студенти, які іспит не пройшли. Дослідження проводилося серед 74 студентів курсу Neurobiology. З цих студентів 38 проходили онлайн курс, їх середній вік був 27 ± 2,5 роки, тобто від 23 до 35 років (47% були жінками). Традиційним навчанням курс проходили 36 студентів, середнім віком 26 ± 2,3 роки, тобто від 24 до 31 року (33% були жінками). Метою дослідження було оцінити ефективність функцій онлайн курсу з інтерактивними інструментами з точки зору студентів. Викладачами в онлайн курсі були ті самі професори та доценти, які викладають традиційним навчанням цей курс. Час навчання для двох форматів був однаковим. Суб’єктами дослідження були аспіранти, які обрали цей курс.

Для перевірки спроможності інтелектуальних інструментів підвищувати ефективність навчання студентів, які мають нижчі показники академічних знань

було розділено студентів у групах на вищий академічний рівень знань та нижчий (для цього було використано оцінки за навчання). Завдання були видані для студентів однакові (це були тести із короткими варіантами відповідей). Протягом трьох тижнів курсу завдання видавалися один раз на тиждень. Розподіл призвів до участі в онлайн курсі 18 учнів із нижчою успішністю та 20 із високою, а в очному класі – 17 студентів із нижчою та 19 із високою успішністю. Кожному студенту було підраховано середню оцінку. Середня оцінка учнів, які навчалися онлайн – 85, а тих які очно – 83.

Для аналізу проведеного дослідження було використано запитання за шкалою Лайкерта для оцінки задоволеності студентів курсами онлайн. Результати були відображені у пропорціях та проаналізовані через GraphPad Prism 7. Статистика відображалася у відсотках і середніх значеннях ± стандартна помилка (SE). Порівняння балів та ефективності режимів очного та онлайн навчання проводилося за допомогою незалежного t-критерію, при значущому значенню $p \leq 0,05$. Було виявлено значну кореляцію 0,52 та 0,67 між середнім балом за завдання та підсумковим іспитом (рис. 1.5).

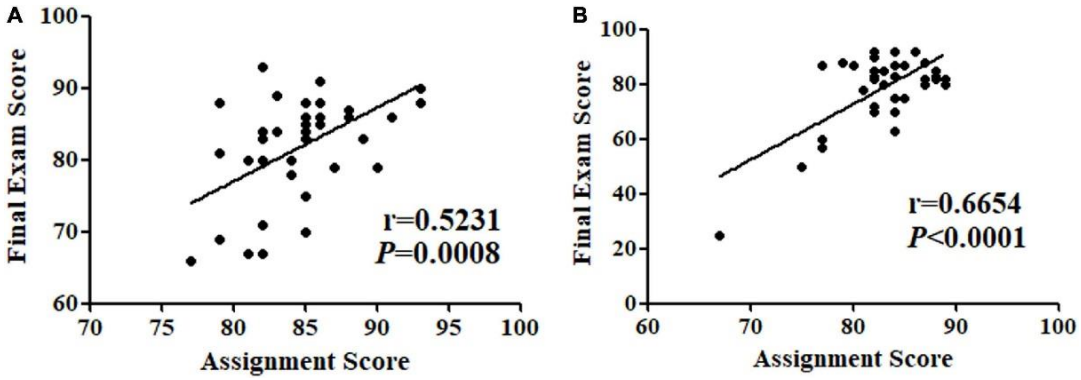

Рисунок 1.5 – Співвідношення результатів підсумкового іспиту з балом за завдання для студентів онлайн (А) і очних (В)

Джерело: [29]

Для кращого засвоєння матеріалу онлайн без нагляду викладачів було скорочено довжину викладання розділу до 30 хвилин. Протягом цього періоду були залучені різні інтерактивні функції: вікторини, голосування, кнопка «не розумію» і багато подібного.

Звичайно для деяких спеціальностей вимагається більше практичних занять, де онлайн навчання не буде таким корисним, але стане корисним допоміжним інструментом. Ще одним не менш важливим аспектом є взаємодія в процесі навчання, адже недостатність цього аспекту може призвести до низької мотивації, відчуття ізоляції та поганого враження від навчання [29].

Ще одне дослідження на тематику ефективності інтерактивних інструментів в онлайн навчанні було проведено у 2022 році. Було проведено дослід із залученням 120 учнів різних рівнів, для виявлення більш активних та менш. Для проведення статистичного аналізу було використано R-Studio. В результаті було визначено, що використання онлайн інструментів в навчанні позитивно корелює з кількістю активних учнів. У віртуальному класі були проведені різноманітні тестування із використанням інтерактивних інструментів та без їх використання, після чого був здійснений порівняльний аналіз. Врахованими були різні зовнішні фактори під час проведення цього тестування. Дані збиралися в однакових умовах. Наприклад, у першому випадку студенти відповідали на запитання викладачів у онлайн форматі, але без використання інтерактивних засобів. Наступного тижня (в той самий час) студентам були задані питання в тій самій онлайн платформі, але вже з використанням інтерактивних засобів. З інтерактивних засобів було використано «Mentimeter». В наступній фазі дослідження було двічі проведено онлайн тест. Перший тест було написано після 10 лекцій без використання інтерактивних засобів, та з їх використанням написано другий після ще 10 лекцій. Отримані результати було використано для порівняльного аналізу (з допомогою RStudio).

Перші 2 тижні онлайн заняття проходили без використання інтерактивних інструментів. На 3-му тижні були введені інтерактивні інструменти, які давали можливість відповідати на запитання викладачів (така практика діяла з 3-го до 9-го тижня). З 9-го й до 12-го тижня використання інтерактивних інструментів було знову припинено. Кількість наданих відповідей постійно записувалася (студентів, які відповіли). На 12-му тижні знову було введено інтерактивні інструменти. Спочатку було виявлено, що кількість зареєстрованих відповідей порівняно із кількістю студентів йшла на спад. Після 3-го тижня цей показник почав зростати.

Також почала збільшуватися кількість студентів, які відвідували. Так продовжувалося до моменту іспиту в середині семестру (8-ий тиждень). Після екзамену (з 9-го по 12-ий тиждень) було припинено використання інтерактивності. Спостерігалося певне зниження кількості відповідей студентів. З 12-го тижня було знижено кількість відвідувань (бо семестр завершувався), але кількість відповідей зросла, адже інтерактивні інструменти знову було впроваджено (рис. 1.6).

Week	Attended	Responded
Week 1	72	27
Week 2	54	11
Week 3	47	26
Week 4	71	46
Week 5	87	61
Week 6	92	67
Week 7	102	71
Week 8	97	68
Week 9	98	46
Week 10	92	37
Week 11	74	29
Week 12	68	43
Week 13	54	32
Week 14	52	37

Рисунок 1.6 – Кількість відвідувань та відповідей студентів

Джерело: [30]

Коефіцієнт кореляції Спірмана (R) (між кількістю студентів які відвідували та їх відповідями) був обчислений та дорівнював 0,77. Тобто вони позитивно корелюють, отже інтерактивне навчання збільшує присутність на заняттях. Значення p = 0,0013, що менше за 0,05, – свідчить про значущість статистичного аналізу в 95%-му довірчому інтервалі. Кількість присутніх на заняттях складає позитивну кореляцію із кількістю наданих студентами відповідей, що продемонстровано на рис. 1.7.

На 2-му етапі експерименту проводилося 2 тести в різні часові періоди. Середні бали були 6,3 з 10 в першому тесті та 7,09 – в другому. Тобто продуктивність значно зросла.

Результати цього дослідження охоплюють лише певний предмет, день тижня (по середам вносилися дані для розрахунку) та час. За інші дні та час також дані

записувалися, але не вносилися до кінцевих розрахунків. Вони можуть використати ще ці дані для подальших досліджень. Використання інтерактивних цифрових інструментів може підвищити кожен етап засвоєння знань. Але дані не враховують наскільки успішно студенти написали тестування (рис. 1.7).

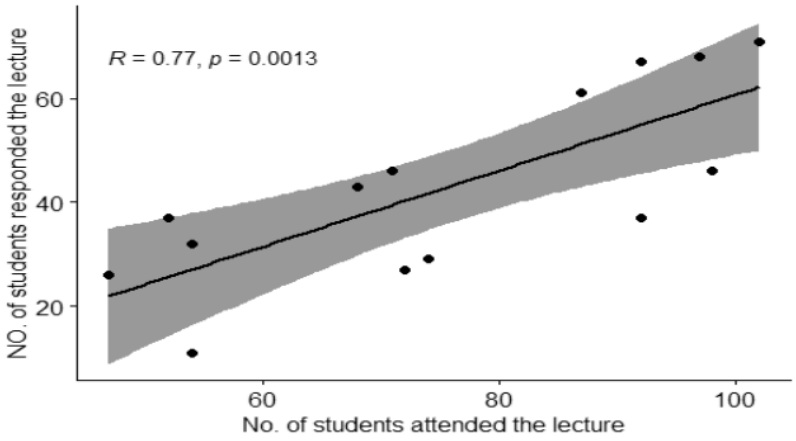

Рисунок 1.7 – Діаграма розсіювання для кількості студентів та їх відповідей

Джерело: [30]

На основі розглянутих досліджень можна виділити основні переваги та недоліки впровадження вебінструментів оптимізації навчання.

Основні переваги освітніх вебінструментів:

1. Активна залученість студентів завдяки більшій зацікавленості, мотивації та кращому засвоєнні навчального матеріалу.

2. Персоналізація навчання. Вебплатформи можуть адаптувати матеріал до індивідуальних потреб студента, що дає можливість рухатися в певному темпі.

3. Миттєвий зворотній зв’язок, який можливо отримати шляхом проходження тестувань та перегляду результатів (для аналізу помилок та покращенню рівня знань, що буде помітно при проходженні наступних опитувань).

4. Підвищена зацікавленість, яка сприяє утриманню уваги студентів.

5. Гнучкість та доступність полягає у легкому доступі до інтерактивних платформ із будь-якої точки світу в зручний час.

6. Спільне навчання та співпраця – чати, групові проєкти та інші інтерактивності, які дозволяють співпрацювати та розвивати навички комунікації та сприяють якісній роботі в команді.

Основні недоліки освітніх вебінструментів:

1. Технічні проблеми, такі як відсутність Інтернет-з’єднання, проблеми з ПЗ та обладнанням.

2. Висока вартість. Для багатьох навчальних закладів буде занадто дорого купувати ліцензії на ПЗ та постійно оновлювати технології.

3. Недостатня технологічна грамотність студентів та викладачів.

4. Зниження соціальної взаємодії, яке може негативно вплинути на соціальні навички студентів.

5. Залежність від технологій, у разі виникнення якої можливе перенавантаження інформацією та виникнення стресу.

6. Необхідність мати самодисципліну, яка виникає через меншу структуризацію та нагляд ніж при традиційній формі навчання.

Якщо порівнювати переваги та недоліки можна помітити, що ймовірність виникнення деяких із недоліків є надзвичайно низькою. З огляду на ефективність впровадження інтерактивних рішень та можливість, як окремого їх використання, так і в якості допоміжного засобу для традиційного навчання доведено доцільність їх створення та використання.

Для проєкту інтерактивної освітньої вебплатформи модель предметної області складається з таких елементів:

1. Користувачі (студенти, викладачі, адміністратор). Студент: має профіль, редагує профіль, бере участь у чатах, переглядає склад власної групи, переглядає викладацький склад, переглядає оголошення. Викладач: має профіль, редагує профіль, переглядає склад учнів по групах, переглядає викладацький склад, переглядає оголошення. Адміністратор: створює чати, переглядає склад студентів по чатах, переглядає користувачів, створює оголошення, переглядає оголошення.

2. Групи: відображають об’єднання студентів за навчальними групами, використовуються для автоматичного додавання учасників до групових чатів.

3. Чати груп: створюються адміністратором для академічних груп.

4. Оголошення: створюються адміністратором, переглядаються всіма ролями користувачів.

1.3 Перспективи еволюції та вдосконалення освітніх вебплатформ

Існує чимало платформ, які допомагають проводити освітній процес. Більшість платформ стали більш використовувані через пандемію COVID-19. Деякі з платформ не були створені конкретно під потреби навчання, але набули широкої популярності серед освітян та здобувачів освіти.

В 2022 році на Міжнародній конференції з обчислювальної науки та обчислювального інтелекту (International Conference on Computational Science and Computational Intelligence (CSCI)) було представлено статтю «Освітні перспективи навчальних онлайн платформ» («An Educational Perspective on Online Learning Platforms»). В статті було розглянуто використання так перспективи таких засобів, як Zoom, Udemy, YouTube, Skype, BlueJeans та Blackboard.

Zoom було створено ще в 2011 році, але найбільш популярною платформа стала в 2020 році, коли викладачі шукали рішення для проведення занять учням. Якщо порівняти кількість користувачів, то вона зросла від 10 мільйонів користувачів у 2019 році до 30 мільйонів у 2020 році (станом на квітень 2020 року). Ця платформа виявилася зручною завдяки можливості проводити заняття із відео та аудіо в онлайн форматі. Також викладачі мали змогу показувати власний екран та записувати заняття (для учнів, які не мали змоги бути присутніми або хочуть переглянути заняття). Але вчені помітили, що через різкий перехід, ці зустрічі здавалися довшими, ніж тривали насправді, та відчувалося певне роздратування. Дослідники виявили, що причиною цьому є те що хоч відеодзвінок відбувається в режимі реального часу, насправді присутня певна затримка до якої мозок не звик. Мозок намагається працювати важче для виправлення цієї затримки, щоб зробити все синхронно, настільки це можливо. Це відбувається, бо мозок звик до швидкої синхронізації при спілкуванні людей один з одним і тому при затримці у відеоконференції (навіть при тому що вона майже не помітна) прикладає багато зусиль для отримання синхронізації, як при звичайному офлайн спілкуванні людей. Ще однією причиною виникнення втоми та роздратування є стрес, який може виникнути під час відеоконференції через ввімкнену камеру.

Udemy – платформа, де зареєстрований користувач може обрати цікаву тематику, знайти курс, купити його та вивчати. Ця програма надає можливість викладачам створювати власні курси та викладати їх на платформу. Основною перевагою є велика кількість курсів з дуже різних сфер. Але недоліком є те, що не має безкоштовних курсів, як в Khan Academy. Але ще одною перевагою Udemy є можливість повертатися до перегляду пройдених уроків і необмежена кількість переглядів. Також є можливість поставити запитання в секції питань та відповідей (укладачі курсів зазвичай відповідають на поставленні запитання). Є можливість також ставити питання одне одному (між тими хто навчається на курсі). Звіти показували, що в квітні та травні 2022 року підвищилася кількість реєстрацій в Udemy на 200%.

YouTube – платформа для поширення відео, створена Google. Це не є освітньою платформою, але багато людей використовують її саме з цією метою. Складність використання цієї програми в кількості відео, серед яких іноді досить складно знайти необхідне. На противагу іншим платформам, YouTube складається з алгоритмів, які пропонують відео тієї тематики, яку найчастіше шукає користувач. На відміну від Google Meet і Zoom, YouTube не надає взаємодії учням в реальному часі (якщо це не прямий ефір). Ще до пандемії COVID-19 чимало студентів та учнів використовували для перегляду додаткової інформації та кращого розуміння відео з тематики, яку вивчали. Під час пандемії та після почалося більш масове використання YouTube серед вчителів. Деякі вмикали певні додаткові відео під час уроків в Google Meet та Zoom, а іноді навіть і самостійно записували відео. Дослідження Joseph Lichter демонструє ефективність використання YouTube, як платформи для навчання ще до пандемії. В досліджені порівнювали успішність складання екзаменів учнями, які використовували YouTube для вивчення та підготовки та тих, які не користувалися. Ще одне дослідження було проведено Y. Chtouki, яке показувало ефективність використання YouTube, як допоміжного інструменту для навчання. Результатом цього дослідження стало те, що 65% студентів склали тест серед тих хто використовував YouTube та 50% серед учнів, які використовували тільки традиційне навчання. Але все одно, YouTube не є

найкращим варіантом, як саме освітньої платформи, адже містить дуже багато різнобічного контенту. Ще одним негативним моментом є реклама, яка з'являється при перегляді відео неодноразово (якщо немає підписки). В статті наголошують, що для розгляду YouTube в якості освітньої онлайн платформи має бути введено ряд змін. По-перше, для пошуку саме навчальних відео має бути створений окремий алгоритм (не заснований на кількості переглядів та вподобань). Друге – створити зв'язок між YouTube та Google Meet, щоб працювати на дві платформи одночасно. Третім аспектом є зменшення кількості реклами та можливість її переглянути на початку відео (замість появи в середині відео). YouTube для навчання краще у якості додаткового інструменту, адже якщо використовувати в якості основної освітньої платформи то недоліків більше, ніж переваг.

Skype – ПЗ, яке дозволяє здійснювати дзвінки, збиратися у відеоконференції та обмінюватися повідомленнями в чаті. Але чимало студентів скаржилися на переривання звуку, відлуння, погане відображення (або взагалі його зникнення) без покупки версії premium.

BlueJeans та Blackboard є двома платформами, які також досягли великої популярності в час пандемії. BlueJeans надає сумісну хмарну платформу для відеоконференцій. В час пандемії сервісом користувалися люди з більше ніж 180 країн. Blackboard – освітнє вебсередовище, розроблене компанією Blackboard Inc. Було проведено опитування серед 494 студентів щодо розуміння думки студентів про Blackboard. Визначено було, що основні фактори впливу використання Blackboard є очікувана продуктивність, соціальний вплив та сприятливі умови. Blackboard є легкою у використані і доступі для студентів та викладачів завдяки зручному інтерфейсу. Проте є й певні недоліки, такі як недостатня навігаційна система, складна навігація в мобільному дизайні, та недостатність тренувань для використання цієї платформи.

Для того щоб впевнитися в продовженні розвитку освітніх онлайн платформ в період після пандемії, на думку авторів статті необхідно більше інвестицій. Також мають з'являтися нові освітні інструменти, покращуватися ті, які існують та збільшуватися доступ до Інтернету та онлайн бібліотек [31].

Є ще одне освітнє вебрішення з великою перспективою майбутнього розвитку – Дія.Освіта. Цей портал створений для отримання актуальних знань та навичок. Дія.Освіта є наступним етапом проєкту Дія.Цифрова освіта (стартував у 2020-му році та був спрямований на цифрову грамотність та навички). В 2023 році було проведено дослідження, яке показало що 59,6% українців володіють цифровою грамотністю хоча б на базовому рівні. Дія.Освіта має мету зробити прорив не тільки в цифрових знаннях, а загалом в різних актуальних у світі навичках та вміннях. Це рішення спрямовано на отримання доступу до освітніх практик для кожного, щоб створити звичку постійно вивчати щось нове. Передбачається, що людина може використовувати сервіс, як для покращення навичок в своїй професійній сфері, так і як можливість вивчити нове та стати професіоналом в зовсім новій для людини сфері. У симуляторах Дія.Освіта людина може подивитися сюжети, з якими стикаються люди з різних професій при виконанні своєї роботи. Було змінено підходи в роботі ІТ студій, для вивчення інформатики школярами. Предмет тепер охоплює цифрову грамотність, медіатворчість, програмування, аналіз даних, які вивчаються завдяки новим підходам – елементам гейміфікації, дослідно-пізнавальним та проєктним підходами, навчання через запитання та використання інформаційних ресурсів. Ще доступний цифрограм – тест на цифрову грамотність, який створений за стандартами Європи та адаптований українськими експертами.

Дія.Освіта регулярно проводить дослідження щодо цифрової грамотності суспільства. Таке дослідження було проведено вперше в 2019 році, потім у 2021 та в 2023 роках. Останнє дослідження (2023 рік) демонструє:

– зростання 5,4% кількості осіб, які мають доступ до Інтернету (в порівнянні з 2019 роком);

– підвищилися цифрові навички на 12,6% у порівняння із 2019 роком (тобто кількість громадян із низькими цифровими навичками знизилася з 53% до 40,4%);

– 58,3% зацікавлені у розвитку цифрових навичок (віком від 18 до 70 років);

– 31% дорослих людей використовує ШІ;

– життєво-необхідним став доступ до Інтернету для 92%;

– в цілях навчання та розвитку Інтернет використовують 72%;

– 96% підлітків використовують Інтернет [32].

Майбутнє навчання в онлайн просторі пов’язано з розвитком новітніх технологій: ШІ, віртуальна реальність (VR), доповнена реальність (AR). Ці технології мають потенціал зробити навчання більш персоналізованим, цікавим та інклюзивним.

При такому розвитку, можливо викладачі будуть більше в ролі модераторів (наставників), які за допомогою аналізу даних покращуватимуть стратегії навчання та підвищують власний професіоналізм. Увага при розробці буде приділена доступності та інклюзивності з ціллю подолати цифрову нерівність. Тобто будуть розроблятися вебплатформи для закриття потреб усіх студентів, в тому числі й людей з обмеженими можливостями.

Майбутнє освітніх вебплатформ націлене ще й на привчання людей, що постійне навчання протягом життя є нормою. Завдяки сертифікованим курсам для підвищення кваліфікації, корпоративним освітнім програмам, працівники підтримують свої знання та навички в актуальності. Але таке навчання все одно ще міститиме ряд проблем: втома від екранів пристроїв, захист даних, потреба в взаємодії з людьми.

Якщо прийняти ці зміни в освіті та підготуватися до майбутніх викликів, які можуть виникнути, то можна створити ефективнішу систему освіти для розкриття потенціалу кожної людини [33].

Для ефективного проведення освітнього процесу із використанням вебплатформ має бути забезпечено функціонал на користь учасників здобування освіти. Тому важливо, щоб платформи надавали, як доступ до навчальних матеріалів та завдань, так і наявність комунікаційної складової. Для багатьох стало б корисним відслідковувати власний прогрес. Звичайно, простота у використані та зручна навігація відіграють важливу роль. Важливо мати різні програми, які будуть інформувати людей про такі платформи та можливості, щоб люди могли використати їх при вивчені чогось нового та покращенні навичок у власній сфері діяльності. Для забезпечення успішного впровадження освітніх платформ необхідно врахувати зручність користування для всіх категорій користувачів.

1.4 Висновки до першого розділу

Вебплатформи є вебінструментом, який на відміну від вебдодатка містить розширений функціонал, який дозволяє користувачам взаємодіяти один із одним. Загалом еволюція вебрішень починалася з появи Web 1.0, переросла в Web 2.0 і починає еволюціонувати, з появою ШІ, в Web 3.0. Основами будь якого вебрішення є протокол обміну даними HTTP, адреси пошуку в Інтернеті URI/URL, гіпертекстова розмітка HTML. Для зручності управління різними негативними ситуаціями існують системні HTTP помилки.

Освітні вебплатформи мають мету – шляхом розміщення інформації, проведення цікавих тестувань та залучення інших інтерактивних інструментів навчати людей та, що саме головне, зацікавити та заохотити до постійного самовдосконалення шляхом вивчення нового та закріплення знань, які вже існують.

Було розглянуто дослідження, де серед студентів які навчалися за традиційною системою без використання інтерактивних інструментів та із їх використанням, було порівняно успішність навчання. Також було розглянуто дослідження, де просто порівнювали традиційну та онлайн освіту, її ефективність. На основі розглянутих досліджень, які існують, можна зробити висновок, що освітні вебплатформи значно підвищують ефективність та результативність навчання студентів.

Різке зростання популярності використання освітніх вебплатформ, пов’язано з пандемією COVID-19. Через необхідність проводити процес навчання в онлайн форматі, щоб не виходити із власного помешкання треба було шукати нові програмні рішення. Такі застосунки, як Zoom, Udemy, Google Classroom, Microsoft Teams та інші стали невід’ємною частиною освітнього процесу. Також ще впроваджують та розширюють Дія.Освіта – вебпортал для вільного доступу громадянам України до освітніх матеріалів із різних професійних сфер.

Загалом інтерактивні вебплатформи є корисним інструментом для цікавого навчання, але важливо мати ще й комунікаційну складову, щоб учасникам навчального процесу було зручніше та цікавіше навчатися.

РОЗДІЛ 2.
ТЕХНОЛОГІЧНИЙ АНАЛІЗ ІНСТРУМЕНТІВ ТА ЗАСОБІВ РОЗРОБКИ ОСВІТНЬОЇ ВЕБПЛАТФОРМИ

2.1 Характеристика середовища розробки програмного забезпечення

Для розробки будь-якого ПЗ, незалежно від його цілей та призначення використовується певне середовище розробки. Сукупність програмних засобів, які програмісти використовують для розробки ПЗ називають середовищем розробки.

Більшість середовищ розробки включають в себе певні компоненти:

– редактор тексту;

– компілятор і/або інтерпретатор;

– засоби автоматизації збирання;

– налагоджувач.

Якщо всі ці компоненти зібрати в одну програму, то це вже буде інтегроване середовище розробки. Високотехнологічні рішення для розробників пропонують такі компанії-розробники, як Microsoft (Microsoft Visual Studio 2022, Microsoft Visual Studio Code), Embarcadero (Embarcadero RAD Studio 12.2, Embarcadero C++Builder 12.2, Embarcadero Delphi 12.2), JetBrains (IntelliJ IDEA, PhpStorm, GoLand, Rider, WebStorm, DataGrip, ReSharper), Atlassian (Jira Software, Jira Confluence) та інші [34].

Вибір правильного середовища розробки пливає на ефективність роботи розробника, а отже й на якість кінцевого програмного продукту. Для розробки вебдодатків на платформі ASP.NET Core найпопулярнішими є Visual Studio та Visual Studio Code. Було обрано Visual Studio 2022 для створення інтерактивної освітньої вебплатформи. Середовище розробки Microsoft Visual Studio 2022 містить великий набір інструментів з якими працюють програмісти для ефективного створення великих програмних проєктів. Було обрано саме це середовище через його повну інтеграцію з технологією .NET та підтримку основних функцій розробки: автозаповнення коду, рефакторинг, відлагодження та тестування.

Особливості Visual Studio 2022:

1. Наявність пропозицій від IntelliSense, якими можна доповнювати код. Також можна вдосконалювати код, якщо звертати увагу на знак лапочки, який пропонує певні дії для покращення коду (перейменування функції, додавання параметра і т.п.).

2. Вбудоване налагодження проєкту, яке дозволяє поетапно переглядати код та значення, що зберігаються в змінних. Це дає можливість спостерігати за змінами значення змінної, вивчати шлях виконання коду.

3. Можливість писати тести для власного програмного вебрішення, щоб полегшити проведення тестування та підвищити якісь ПЗ.

4. Інтегрований контроль версій, який за допомогою вбудованих функцій Git дозволяє клонувати, створювати, відкривати власні репозиторії. У вікні інструментів Git можна вносити зміни в код та зберігати їх, управляти розгалуженнями і вирішувати конфлікти злиття розгалужень, якщо такі виникають. Якщо є обліковий запис GitHub, то можна керувати репозиторіями безпосередньо із Visual Studio.

5. Наявність спільної роботи в режимі реального часу Live Share прискорює роботу в команді над певним програмним продуктом.

6. Не складне розгортання програми в хмарі завдяки надаванню залежностей до бази даних (Azure SQL) чи облікових записів (Azure Studio) із Visual Studio.

7. Можливість для створення кросплатформних програм для Windows, Mac, Linux, iOS, Android.

8. Інструменти профілювання доступні для створення швидких та адаптивних .NET та C++ додатків. Створювати багатофункціональні додатки можна за допомогою різних технологій: WinForms, WPF, WinUI, MAUI, Xamarin.

9. Робота із базою даних в SQL із Visual Studio (без виходу із середовища розробки).

Також наявні різні розширення, які можна за потреби підключити [35]. Розширеннями є додаткові компоненти, які розширять функціональність та адаптують інтегроване середовище під потреби розробника.

2.2 Технологічні засади розробки серверної частини вебплатформи

2.2.1 Особливості використання фреймворку ASP.NET Core для створення вебдодатків

ASP.NET Core – популярний фреймворк для розробки вебдодатків на платформі .NET. Ключовим у цьому фреймворку є його продуктивність, яка в порівнянні з деякими іншими популярними вебфреймворками є вищою (рис. 2.1). ASP.NET Core був випущений в 2016 році, як оновлена версія ASP.NET (який був доступний тільки для Windows) [36].

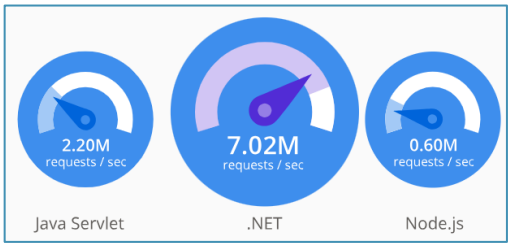

Рисунок 2.1 – Продуктивність .NET в порівнянні з Java Servlet і Node.js

Джерело: [36]

В порівнянні із ASP.NET, в ASP.NET Core було видалено залежність від System.Web.dll. Microsoft створив кросплатформний сервер Kestrel, на якому можна запускати ASP.NET Core проєкти. При створенні вебплатформи ASP.NET Core виступає серверною частиною, яка генерує завантаження сторінок. Веббраузер робить GET-запити з використанням різних URL, які позначають кожну сторінку. Є можливість керувати даними з серверу за допомогою POST, PUT, DELETE запитів.

Браузер виступає презентаційним шаром, в той час як більшість дій відбувається на сервері. Деякі дії, такі як перевірка валідації, або відображення елементів інтерфейсу можуть використовувати JavaScript. ASP.NET Core пропонує декілька технологій для побудови вебдодатків: ASP.NET Core Razor Pages, ASP.NET Core MVC, Blazor [37].

Основними перевагами ASP.NET Core є підтримка модульності, можливість побудови вебінтерфейсів та вебсервісів, підтримка вбудованих залежностей,

можливість подальшого хостингу, інтеграція сучасних фреймворків для створення інтерфейсу, швидкість та відкритий код, паралельне створення вебверсій, кросплатформність.

На противагу .NET, .NET Core є набором NuGet пакетів, які надають окремі невеликі частини функціональності. Це робить додатки легшими, компактнішими та більш гнучкими. Завдяки тому, що .NET Core та ASP.NET Core мають відкритий код, – розробники можуть приймати участь у перевірці коду, виправлені коду. Для побудови програм для Windows, Linux, Mac OS у .NET Core використовується єдина кодова база [38].

Фреймворк ASP.NET Core, завдяки хорошій продуктивності підходить для складних, масштабних вебрішень, таких інтернет-магазини, сервіси для обміну даними, інтерактивні вебплатфоми. Цей фреймворк є зручним для інтеграції з різними базами даних, що є ключовим при створенні вебдодатків.

2.2.2 Характеристика мови програмування C# у контексті розробки вебрішень

C# є об’єктно-орієнтованою мовою програмування (ООП), яка працює на платформі .NET. Мови ООП дають змогу створювати додатки змодельовані за об’єктами реальними. Використання цих мов програмування дає можливість створювати більш інтерактивні додатки. Завдяки ООП, програмісти створюють код, який є більш ефективним і багаторазовим, що заощаджує час і вартість розробки. Об’єкти в ООП – дані, які представлені атрибутами/властивостями та методами. ООП дозволяє об’єктам взаємодіяти між собою завдяки чотирьом принципам:

1. Інкапсуляція є принципом, який має за мету приховати деталі реалізації від зовнішнього оточення. Це означає, що вся важлива та чуттєва інформація міститься в об’єкті, інші дані є доступними зовні. Робота внутрішня, яку виконує кожен об’єкт та його стан залишаються приватними всередині класу, тому інші об’єкти не мають доступу до даних і не можуть вносити зміни. Але взамін вони можуть мати доступ та взаємодіяти з методами, які є публічними. Використання ідентифікаторів

доступу для такого розділення є корисним для забезпечення безпеки програми, контролю за станом об’єкта, зменшує ризик помилок.

2. Наслідування – принцип, який дозволяє створювати нові класи на основі класів, які вже існують. Наслідування передбачає переписування певних методів батьківського класу, або розширення функціоналу класу, який наслідується. Цей підхід є досить корисним, особливо у програмних проєктах, які містять багато рядків коду, адже допомагає уникнути дублювання коду. Тобто розробники можуть в дочірньому класі використовувати логіку роботи батьківського, але при цьому бути різними.

3. Поліморфізм є ще одним принципом, який доповнює наслідування шляхом виконання об’єктами з різних класів дій з однаковими іменами, але з різним кодом. Поліморфізм дозволяє створити загальні методи, які можуть бути використані для різних типів об’єктів.

4. Абстракція – це принцип, який дозволяє зосередитися на основних елементах системи й ігнорувати побічні. Завдяки принципу абстракції кожен об’єкт розкриває лише певний механізм використання [39].

Мова програмування C# походить із сімейства мов програмування C та має багато спільного із такими мовами програмування, як C++ та Java. Переважно використовується для розробки комп’ютерних застосунків, ігор (Unity), мобільних додатків, вебсервісів та іншого. Для створення програм з ASP.NET Core використовується C#. Останньою версією є C# 13, яка була випущена в листопаді 2024 року разом із .NET 9 [40].

C# є мовою програмування високого рівня, яка компілюється в середовищі CLR (Common Language Runtime), яке автоматично керує пам’яттю. Тобто, CLR слугує для збирання сміття, обробки винятків і т.п., тому розробнику не треба писати власний код для виконання цих задач.

При запуску програми на C# компілятор створює допоміжну мову (IL), замість машинної мови, бо CLR розуміє саме IL. Після цього CLR компілює код (метод за методом) у скомпільований машинний код у пам’яті та виконує його. Для виконання цього має бути інстальовано CLR. Усі нові версії Windows мають

вбудовану версію CLR, а на старіших версіях можна зробити її доступною через Windows Update.

В .NET, замість бібліотеки середовища виконання яка призначення для однієї мови програмування, є бібліотека класів .NET Framework (FCL), яка містить десятки тисяч об’єктів, які можна багаторазово використовувати [41].

Перевагами мови програмування C#:

1. Можливість заощадити час завдяки суворій типізації мови, бо скорочується кількість можливих помилок.

2. Відносна легкість у вивчені.

3. Масштабованість та простота в обслуговувані.

4. Велика спільнота (рис. 2.2).

5. Мова є об’єктно-орієнтованою [44].

PYPL Popularity of Programming Language Index

Worldwide, Dec 2024 :

Rank	Change	Language	Share	1-year trend
1		Python	29.71 %	+1.5 %
2		Java	15.43 %	-0.3 %
3		JavaScript	7.99 %	-0.9 %
4		C/C++	7.06 %	+0.3 %
5		C#	6.42 %	-0.2 %

TIOBE Index

Dec 2024	Dec 2023	Change	Programming Language	Ratings	Change
1	1		Python	23.84%	+9.98%
2	3	^	C++	10.82%	+0.81%
3	4	^	Java	9.72%	+1.73%
4	2	v	C	9.10%	-2.34%
5	5		C#	4.87%	-2.43%

Рисунок 2.2 – Мова програмування C# у рейтингах PYPL Index та TIOBE Index

Джерело: [42], [43]

C# є багатофункціональною мовою програмування, яка завдяки інтеграції з .NET Core має потужні бібліотеки, які дозволяють створювати додатки різних рівнів складності (від консольних утиліт до масштабних вебплатформ). Крім цього, гарантовано є регулярні оновлення завдяки активній підтримці Microsoft. Освітня вебплатформа реалізовується в ASP.NET Core, а отже й мовою програмування C#.

2.2.3 Аналіз архітектурної моделі Model-View-Controller (MVC)

Архітектура вебдодатків є одним із основних аспектів при виборі технологій для розробки ПЗ. ASP.NET Core має декілька підходів для реалізації:

1. ASP.NET Core Razor Pages використовується для генерації простих HTML вебсторінок для найпростіших вебсайтів.

2. ASP.NET Core MVC містить архітектурний шаблон MVC, який найчастіше використовується для складних вебсайтів.

3. Blazor дозволяє будувати елементи інтерфейсу з використанням C# та .NET замість таких фреймворків як Angular, React, Vue [37].

Кожен вид є корисним, залежно від складності та виду проєкту. Архітектура MVC є найпопулярнішою завдяки ефективності розділення логіки додатку. Ця архітектура базується на трьох компонентах: моделі, контролері, представленні. Кожен компонент виконує свою роль у роботі додатку (рис. 2.3).

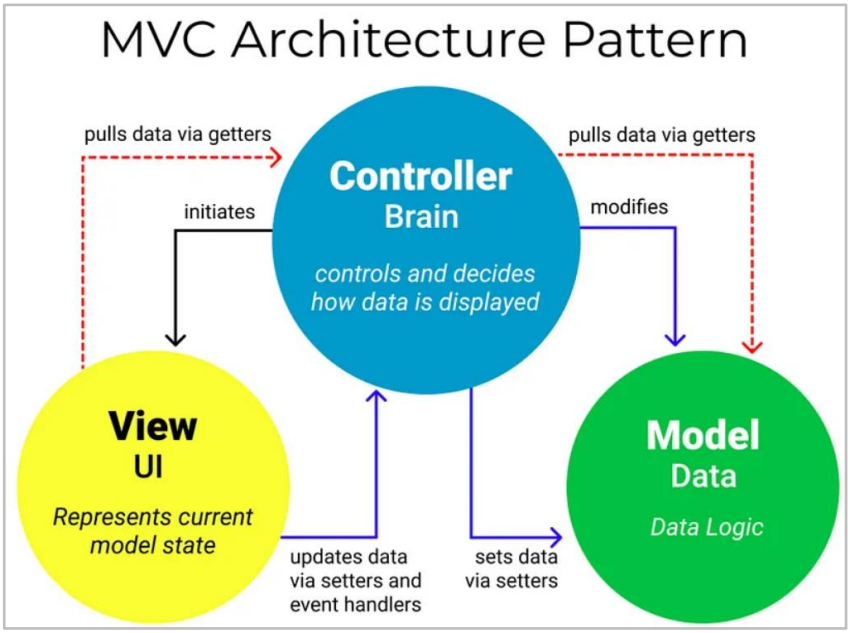

Рисунок 2.3 – Загальний архітектурний шаблон MVC

Джерело: [45]

Модель – компонент, який відповідає за дані з якими працює користувач. Ці дані можуть бути тими, які передаються між представленням та контролером, або інші дані які пов’язані з логікою роботи програми.

Представлення – компонент, який використовується для інтерфейсу користувача (текстові поля, спадні списки та інші елементи з якими взаємодіє користувач).

Контролери взаємодіють між моделями та представленнями для обробки логіки роботи програми та вхідних запитів, управління даними з використанням даних із моделі та взаємодії з представленнями для генерації фінального результату.

Переваги використання ASP.NET MVC:

1. Зручно розробляти складні (кількість функцій), але легкі (розмір) додатки.

2. Наявність розширюваної структури, яку можна змінювати, налаштовувати.

3. Можливість керувати складністю масштабних проєктів і працювати над окремими компонентами.

4. Структура MVC покращує можливості розробки тестів та тестування ПЗ.

5. Підтримує функції ASP.NET, такі як автентифікація, авторизація, маршрутизація та інші [46].

Ідею самої архітектури MVC сформував у кінці 1970-тих років Трюгве Реєнскауг, який тоді працював у Xerox PARC. Загалом MVC є набором архітектурних ідей і принципів, які спрямовані на побудову складних систем із інтерфейсом користувача. В MVC модель є самим незалежним компонентом, якому не треба контролер чи представлення. Представленню вже необхідно мати доступ до моделі, для відображення даних з моделі у форматі зручному для користувачу. Але представлення не має змінювати модель. Контролер обробляє дії користувача, тому може вносити зміни до даних моделі.

Важливим аспектом при створені програми з використанням архітектурного шаблону MVC є правильний поділ на компоненти:

Крок №1. Виокремити бізнес-логіку програмного продукту від інтерфейсу користувача. Спочатку треба розділити на два модулі: перший – ядро системи, в якому буде реалізовано основний функціонал; другий – відображення даних користувачу та логіка взаємодії його з додатком (інтерфейс користувача). Перший модуль, – це і є модель. Незалежна розробка та тестування моделі є основною метою такого розподілу.

Крок №2. Використання спеціального шаблону «Спостерігач» для ще більшої незалежності моделі та синхронізації інтерфейсів користувача. Якщо оновлювати інтерфейс із моделі, то модель буде залежною від інтерфейсу (треба буде пам’ятати різні способи оновлення інтерфейсу). Тому доречним буде реалізувати шаблон «Спостерігач», за допомогою якого модель повідомляє про зміни інтерфейс (тобто інформувати представлення та контролер), але сама модель не має інформації про представлення чи контролер. Цей процес і є синхронізацією користувацьких інтерфейсів.

Крок №3. Поділ інтерфейсу на представлення та контролер. Основне завдання інтерфейсу користувача забезпечити взаємодію із системою. Тобто інтерфейс має дві функції: розміщувати інформацію про систему (представлення), вводити користувацьку інформацію в систему (контролер) [47].

В роботі «Архітектура MVC з точки зору атрибутів якості експлуатації» («MVC Architecture from Maintenance Quality Attributes Perspective») авторка зосереджується на дослідження архітектури MVC, зокрема її підтримку атрибутів експлуатації. При дослідженні було використано атрибути експлуатації стандарту ISO – MMERFT (модифікованість (modifiability), модульність (modularity), розширюваність (extensibility), багаторазовість використання (reusability), гнучкість (flexibility), тестування (testability)). Це дослідження було проведено із фреймворком Yii.

Завдяки роздільності архітектури MVC модифікованості та багаторазовості використання можна застосовувати зміни в кожній частині без ризику вплинути на іншу. Модифікованість пов’язана із багаторазовим використанням, так як розробникам часто необхідно модифікувати старі технології шляхом багаторазового використання нових для виконання вимог користувача (архітектура MVC дозволяє повторно використовувати компоненти). Також багаторазове використання прослідковується в можливості MVC підтримувати принцип Don’t Repeat Yourself (DRY). Можливість змінювати готові компоненти без необхідності заглиблюватися в їх деталі дозволяє мінімізувати потреба в написані додаткового коду, а тобто й ризик виникнення помилок зменшується. В дослідженні на прикладі

«ActiveRecord» у фреймворку Yii, завдяки чому спрощується виконання GRUD операцій.

Модульність є важливим фактором зручності експлуатації, адже модулі повинні мати високу когезію (всі елементи у модулі працюють для спільної мети) та низький зв’язок (модулі мають мінімальну залежність один від одного). Багато фреймворків для побудови цих модулів використовують архітектурний шаблон MVC, що допомагає досягти модульності.

Розширення полегшує адаптацію програми до нових вимог користувачів, що в свою чергу покращує зручність експлуатації. Ці розширення можуть бути побудовані, як модуль чи група модулів для покращення архітектури MVC.

Гнучкість MVC проявляється в можливості використання одного компонента для різних сценаріїв. Наприклад, модель користувача може бути використана, як для процесу реєстрації, так і для процесу входу в систему. Аналогічно модель може бути використана й для декількох різних представлень.

Розділення логіки, завдяки використанню MVC, призводить до полегшеного тестування. Завдяки підтримці модульності, код є більш підготовленим до виконання модульного тестування (unit testing).

Загалом MVC добре підтримує атрибути якості експлуатації, що доводить доречність використання цього архітектурного шаблону при створені додатків [48]. Таким чином, архітектура MVC добре підходить для створення структурованих, масштабованих вебдодатків. Популярність використання цієї архітектури пов’язана з її гнучкістю та здатністю адаптуватися до сучасних вимог розробки.

2.2.4 Інтеграція бази даних у середовищі ASP.NET Core

Для взаємодії з базами даних у ASP.NET Core є декілька способів, вибір варіанту залежить від потреб проєкту та його складності. Найпоширенішим способом є використання Entity Framework Core (EF Core), який дає змогу працювати з базою даних через моделі та LINQ-запити. Для проєктів, де пріоритетом є висока продуктивність часто використовують Drapper, який є легким

і забезпечує швидке виконання SQL-запитів із мінімальними витратами ресурсів. Також можна застосовувати ADO.NET, якщо вимагається детальний контроль над взаємодією з базою даних (наприклад, для створення складних SQL-запитів або роботи з великим обсягом даних).

Dapper є бібліотекою з відкритим кодом для об'єктно-реляційного відображення (Object-relational mapping (ORM)) для додатків .NET і .NET Core. Ця бібліотека дає змогу зіставляти результати запитів з об'єктами, виконувати збережені процедури та ще багато іншого. Drapper доступний серед NuGet пакетів. Для додатків, яким треба висока продуктивність та мінімальна затримка буде актуальною ця ORM. Крім цього, Drapper ще підтримує запити з параметрами для захисту [49].

ADO.NET з'єднувач, який використовує інтерфейс ADO.NET для з'єднання джерел даних, які підтримують стандарт протоколу ADO.NET. Мають бути встановлені необхідні компоненти для коректної роботи ADO.NET драйверів (зазвичай це вимагає реєстрації збірок .NET у глобальному кеші збірок) [50].

Entity Framework Core є легкою, масштабованою, кросплатформною версією технології доступу до даних Entity Framework. Ця ORM надає програмістам автоматичний підхід для доступу та зберігання інформації у базі даних. EF Core призначений для використання з .NET Core, але його можна ще використовувати із стандартним .NET останніми версіями. Підтримується два підходи при розробці з EF Core (рис. 2.4).

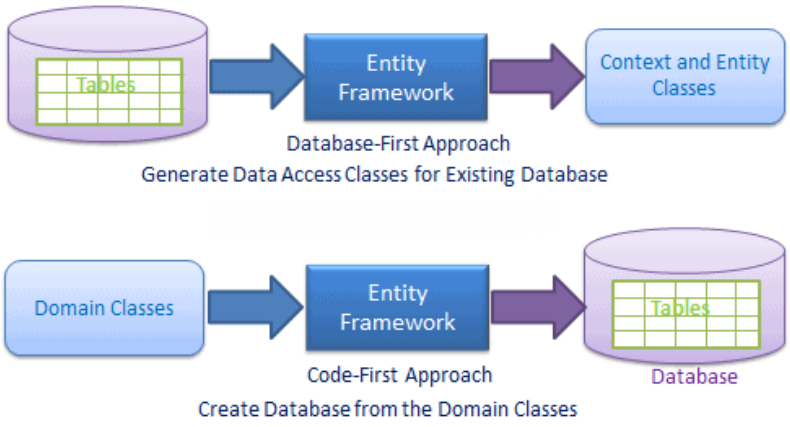

Рисунок 2.4 – Підходи до розробки з EF Core

Джерело: [51]

В підході, де спершу код (code-first) створюється база і таблиці з використанням міграцій, на основі створених розробником класів. В підході, де першою є база даних (database-first) створюються контекстні класи на основі бази даних, яка вже існує завдяки використанню спеціальних команд [51].

Для роботи з EF Core спочатку треба налаштувати підключення до бази даних. В файлі appsettings.json треба вказати рядок підключення (ConnectionString) до бази даних. Далі необхідно додати контекст до бази даних з використанням методу AddDbContext в Program.cs. Потім варто створити класи-моделі, які будуть таблицями в базі даних (зазвичай ці класи розміщують в окремій папці Models). У цих класах мають бути визначені властивості, які відповідатимуть стовпцям таблиці. Для налаштування ключів, або обмежень можуть бути використані спеціальні атрибути, або метод OnModelCreating вже у контексті бази. Власне контекст бази даних – це клас, який успадковується від DbContext. Цей клас містить набір DbSet<T>, які представлятимуть таблиці в базі даних та даватимуть можливість виконувати запити. Ще контекст відповідає за управління підключенням до бази даних та відстеженням змін об’єктів. EF Core використовує систему міграцій для створення та оновлення структури бази даних. Міграції створюються через консольну команду dotnet ef migrations add NameOfMigration, а зміни застосовуються командою dotnet ef database update. Для додавання записів використовують команду Add або AddRange, для видалення – Remove, для оновлення – Update. Для читання даних використовуються різні LINQ-запити. Після внесення всіх необхідних змін викликається метод SaveChanges, який зберігає ці зміни в базу даних.

Однією з основних переваг використання EF Core є можливість виконувати запити до бази даних з використанням LINQ (Language Integrated Query). LINQ є мовою запитів до структурованих даних, що дозволяє зручно виконувати запити до даних без необхідності написання SQL-запитів. Ще однією перевагою є підтримка асинхронних операцій для роботи з базою даних, що дозволяє покращити продуктивність при великих обсягах даних чи тривалих операціях. Для асинхронних операцій використовуються спеціальні методи, такі як ToListAsync(),

FirstOrDefaultAsync(), SaveChangesAsync(), CopyToAsync() та інші. EF Core підтримує роботу з різними базами даних: SQL Server, PostgreSQL, MySQL, SQLite, Azure Cosmos DB. Можна налаштувати складні зв’язки між таблицями (наприклад, один до багатьох, багато до багатьох) та змінювати типи полів або значень за замовчуванням через методи конфігурації у OnModelCreating.

EF Core є потужним інструментом, який спрощує багато операцій з даними, підвищує продуктивність роботи з базами даних і забезпечує високу гнучкість.

2.3 Технологічні засади розробки клієнтської частини вебплатформи

2.3.1 Створення структури та стилізації інтерфейсу з HTML та CSS

Мова розмітки гіпертексту (HyperText Markup Language (HTML)) є набором символів або кодів, який вставляється в файл для подальшого відображення в Інтернеті. Розмітка сповіщає веббраузеру, як відображати текст і зображення на вебсторінці. Кожен елемент розмітки між «<» та «>» називається тегом. Парні теги показують, де початок цього елементу і кінець відповідно [52].

Після появи комп’ютера виникла ідея пов’язати їх один з одним. Чотири американські коледжі працювали над цим завданням. Минуло майже сім років, перш ніж ідея почала втілюватися в життя.

29 жовтня 1969 рік став днем, коли перша комунікаційна сесія між двома вузлами мережі ARPANET була встановлена. Перший знаходився в Каліфорнійському університеті в Лос-Анджелесі (UCLA), другий – в Стенфордському дослідницькому інституті. Відстань між цими двома комп’ютерами була 640 кілометрів. Вчений із Лос-Анджелесу віддалено під’єднався до комп’ютера в Стенфорді. Його колега в Стенфорді спостерігав за введеними на дистанції символами, які з’являлися на екрані та підтверджував їх появу телефоном. Це стало початком комп’ютерних мереж.

Протягом тривалого часу Інтернет використовували лише спеціалісти для обміну технічною документацією та електронними листами.

Тімоті Джон Бернерс-Лі – британський вчений, який винайшов HTML для розмітки та форматування документів в Інтернеті. Тому й перший сайт був створений цим вченим. Дата створення HTML невідома, бо це був досить довгий проєкт, але відомо що перший вебсайт було створено 6 серпня 1991 року. В 1994 Бернс-Лі заснував World Wide Web (W3C). Місія W3C полягає в розробці протоколів та вказівок, які забезпечуватимуть довгострокове зростання мережі та розкриття її повного потенціалу. 22 вересня 1995 року з'явилася версія HTML 2.0. Покращення, які були в цій версії, – це пошук за ключовими словами, поява форм передачі даних з комп'ютера на сервер. З березня 1995 року була розпочата робота над HTML 3.0. Вже перша версія стандарту включала теги для створення таблиць, обтікання текстом, тощо. Під час створення творці зрозуміли, що HTML за задумкою має мати лише структуру документів, без жодних графічних стилів. Але через потребу користувачів треба було щось вигадати для можливості покращення зовнішнього вигляду сторінок, тому було прийнято рішення вигадати новий засіб для створення саме дизайну вебсторінок. Так, 17 грудня 1996 року було створено Cascading Style Sheets.

CSS є таблицями стилів, які можна приєднувати до HTML-документа для візуального оформлення різних частин документів. CSS є незалежним від HTML та має власний синтаксис. Великою перевагою було те, що використання CSS дало змогу впроваджувати зміни у вигляд, але при цьому не змінювати сам HTML-тег. Це прибрало необхідність у створені великої кількості нових тегів.

14 січня 1997 року – поява HTML 3.2. Ця версія була випущена через місяць після затвердження CSS, тому ця версія була вже адаптована під взаємодію з CSS.

У версії 4.0, яка була випущена 18 грудня 1997 року багато минулих тегів було видалено (вони вказувалися як застарілі). Ця версія включала підтримку скриптів, фреймів та загальних процедур, вбудованих об'єктів. Версія 4.0 була розроблена за участі експертів у галузі інтернаціоналізації, тому з'явилася можливість писати текст будь-якою мовою та відправляти його по світу.

24 грудня 1999 року була випущена версія HTML 4.01, в якій було просто підправлено форми, об'єкти та виправлено деякі помилки.

Версія 4.01 була більш стабільною, розробники використовували її майже 10 років. 28 жовтня 2014 року було випущено HTML5. Синтаксис став жорсткішим, на противагу минулим версіям. Покращено було підтримку мультимедійних технологій та додано 28 структурних елементів, які зробили код більш зрозумілим. Чимало застарілих тегів було видалено, а більшу частину уваги приділено підтримці скриптів, таких як JavaScript [53]. HTML та CSS є фундаментальними технологіями, які слугують для створення вебсторінок, при цьому взаємодія між структурою та стилем є досить гармонійною. HTML відповідає за зміст та логічну побудову інтерфейсу, – CSS додає естетики завдяки оформленню шрифтів, кольорів, відступів. Завдяки CSS вебінтерфейси стали більш адаптивними та при цьому забезпечують відображення на різних пристроях, що є зручним для користувачів.

2.3.2 Забезпечення інтерактивності інтерфейсу з JavaScript

Будь-який сайт може працювати коректно з використанням виключно HTML та CSS. Але такий сайт не буде інтерактивним і функціональним, для цього треба використовувати JavaScript.

JavaScript є мовою програмування, яка використовується для створення інтерактивних вебсатів і вебдодатків. Ця мова дозволяє додавати динамічні елементи до вебсторінок для покращення взаємодії із користувачем.

Основні переваги використання JavaScript:

1. Є невід’ємною частиною веброзробки через можливість хорошої інтеграції з верстkою та серверною частиною.

2. Швидкість та продуктивність, завдяки тому що певна частина запитів може оброблятися безпосередньо на пристрої користувача (що є швидшим, ніж кожного разу звертатися до сервера).

3. Велика екосистема та кількість розробників, які користуються цією мовою, що дозволяє швидше знаходити рішення для різних проблем, які можуть виникнути в ході розробки.

4. Простота у використанні та раціональність. JavaScript дає можливість виконувати прості завдання досить швидко, але більш складні завдання також можна виконувати.

5. Простота у вивчені синтаксису, на противагу багатьом іншим мовам програмування.

Основні недоліки використання JavaScript:

1. Неможливість читання та завантаження файлів (можливість є обмеженою та залежить від середовища розробки).

2. Нестрога типізація.

3. Відсутність підтримки віддаленого доступу (можливість є обмеженою та залежить від середовища розробки).

4. Можлива доступність для різних зловмисників.

Для JavaScript головною сферою застосування є frontend-розробка. Ця мова чудово інтегрується разом із HTML і CSS, при цьому розширює можливості для розробки. Найбільш поширеним використанням є написання вебдодатків, які будуть виконуватися в браузері користувача.

Якщо розробник володіє вже ґрунтовними знаннями JavaScript, то окрім вебдодатків, може створювати інші варіанти:

– програми для Android та iOS (з використанням фреймворку React Native);

– серверні програми (з використанням Node.js);

– програми для персональних комп’ютерів (наприклад, із використанням JavaScript розроблені офісні програми Microsoft та програмний пакет Adobe);

– програмувати різну техніку та обладнання (від простих ТВ-приставок до терміналів оплати).

Ще однією доволі простою сферою з точки зору складності створення з використанням JavaScript є створення розширень для браузерів. Переважна кількість браузерних розширень написана саме на JavaScript.

При класичному підході до веброзробки, кожен з елементів відповідає за вирішення певного завдання: HTML – розмітка сторінки, CSS – зовнішній вигляд елементів інтерфейсу, JavaScript – інтерактивність та функціонал.

Основні завдання, які можна вирішити з JavaScript:

1. Організація взаємодії користувача та вебпрограми.

2. Обробка даних HTML. Поля можна створити за допомогою HTML, але реакцію сторінки та ті чи інші дії користувача можна зробити з JavaScript.

3. Додавання різноманітної анімації (рухомі елементи, кнопки та каруселі).

4. Різні математичні обчислення [54].

Завдяки можливостям обробки подій та реалізації анімацій JavaScript добре виконує завдання створення інтерфейсів із високим рівнем інтерактивності. Багато сучасних фреймворків, таких як React, Vue.js, Angular розширюють функціональні можливості цієї мови програмування та дозволяють будувати розробникам більш складні інтерфейси. JavaScript залишається одним із найважливіших інструментів для frontend-розробки, незважаючи на певні обмеження безпеки в браузері.

2.3.3 Реалізація серверної логіки в інтерфейсі з Razor-синтаксисом

Razor-синтаксис – потужний інструмент для створення динамічних інтерфейсів у вебдодатках. Він дає можливість інтегрувати серверну частину безпосередньо в HTML-код для забезпечення ефективної генерації контенту на основі даних із сервера.

Загалом, Razor – це синтаксис, який допомагає розробнику вбудувати серверний код (C#, Visual Basic) у вебсторінки. Це означає, що коли сторінка викликається, то сервер виконує серверний код прямо всередині сторінки до того як повертати сторінку назад у браузер. Можуть виконуватися такі складні завдання, як доступ до бази даних.

При створенні інтерактивної освітньої вебплатформи використовувався Razor-синтаксис, адже Razor був заснований на ASP.NET Core та призначений для створення вебдодатків.

За допомогою Razor можна отримати доступ до Tag Helpers в ASP.NET Core, які можна використовувати вбудовані або створити власні [55]. Tag Helpers представлений спеціальними компонентами, які забезпечують зручний спосіб

роботи з HTML-елементами та додають нові можливості шляхом зміни їх поведінки. Вони використовуються для генерування посилань, форм, скриптів, обробки валідації та інтеграції з моделями даних у проєкті.

Виглядають Tag Helpers як звичайний код HTML:

– asp-for автоматично підключає поля форм до властивостей моделі;

– asp-action і asp-controller слугують для спрощення створення маршрутів для дій контролера.

Для підключення Tag Helpers треба додати рядок підключення: @addTagHelper "*, Microsoft.AspNetCore.Mvc.TagHelpers".

Однією з важливих можливостей Razor є підтримка Layouts, що дає розробникам створювати узгоджений дизайн для всього вебдодатка завдяки використанню спільного шаблону для різних сторінок. Ця можливість значно зменшує дублювання коду та покращує зручність підтримки проєкту. Ще Razor підтримує Partial Views, які є невеликими фрагментами розмітки з можливістю повторного використання у різних частинах додатка. Цей підхід корисний для створення компонентів інтерфейсу, таких як меню, панелі навігації, футери.

Синтаксис Razor підходить для створення багатомовних вебдодатків, завдяки механізмам для локалізації контенту. Вбудована підтримка валідації спрощує процес перевірки правильності введення даних користувачами, без необхідності писати зайвий серверний код.

Використання View Components є підтримуваним у Razor. Ці компоненти є самостійними модулями, які можна використовувати для виконання складних операцій, або генерування спеціального коду HTML. Ця можливість розширює функціонал стандартних часткових представлень та дозволяє створювати більш гнучкі рішення. Окрім цього, Razor оптимізований для швидкої роботи та забезпечує високий рівень безпеки шляхом обмеження шкідливого коду, який може бути введений користувачами.

Інтеграція логіки у вебсторінки можлива в Razor завдяки підтримці конструкцій мови C#. Тобто розробники можуть використовувати умовні оператори, цикли та інші елементи програмування безпосередньо в HTML-коді.

Наприклад завдяки умовним операторам можна виводити ті чи інші елементи інтерфейсу в залежності від виконання тієї чи іншої умови. Для елементів які повторюються, використовувати можна цикли (for, foreach, while). Також тернарний оператор для якихось коротких умов зручно використовувати безпосередньо в HTML. Всі ці оператори, або будь-які інші вбудовані вирази Razor можна використовувати в HTML коді завдяки написанню символу @. За необхідності можна навіть створювати функції, оголошувати змінні та виконувати складні обчислення без необхідності додаткових файлів.

В ASP.NET Core використання Razor дозволяє створювати динамічні вебсторінки простіше завдяки інтеграції елементів серверної логіки (елементів мови програмування C#) до інтерфейсу. Завдяки простому синтаксису та можливості працювати з необхідними даними робота розробників є ефективнішою.

2.3.4 Фреймворк Bootstrap для забезпечення адаптивності інтерфейсу

Фреймворк Bootstrap надає набір готових компонентів, стилів, які можна використовувати для присокрення процесу реалізації сучасних та зручних інтерфейсів для різних пристроїв.

З розвитком технологій виникла потреба відкривати ресурси на різних пристроях. Для цього інтерфейс має бути адаптивним, щоб відображатися в зручному форматі для користувача, незалежно від пристрою який використовується для роботи із вебресурсом. Тобто адаптивність у веброзробці є здатністю вебсторінки чи вебдодатку автоматично підлаштовуватися під різні розміри екранів пристроїв. Завдяки адаптивному дизайну вебрішення працюють коректно та є зручними для користувачів, як на великих екранах персональних комп’ютерів, так і на малих екранах телефонів і планшетів. Це покращує в першу чергу досвід користувача та допомагає уникнути проблем із навігацією. Пошукові системи зазвичай віддають перевагу сайтам, які є оптимізованими під мобільні версії.

Bootstrap допомагає полегшити написання коду сторінки та розробку адаптивності. Основна перевага – сіткова система, яка дозволяє легко створювати

адаптивні макети, що автоматично підлаштовуються під розміри різних екранів. Крім цього пропонується велика кількість готових компонентів інтерфейсу (кнопки, форми, модальні вікна, навігаційні панелі й інші елементи інтерфейсу). Також Bootstrap інтегрується з популярними JavaScript бібліотеками, що розширює його функціональність, а отже дає можливість створювати інтерактивні інструменти без додаткових налаштувань.

Для підключення Bootstrap до програмного проєкту є два способи. Перший спосіб полягає в завантажені файлів на сервер та подальше їх підключення у код сторінки з використанням тегів <link> і <script>. Цей спосіб дає можливість працювати без залежності сторонніх ресурсів, але вимагає більше часу для налаштування та може збільшити час завантаження сторінки (якщо файли не оптимізовані, або сервер має обмежену пропускну здатність). Другий спосіб передбачає використання Content Deliver Network (CDN), де файли Bootstrap завантажуються з віддалених серверів. Це забезпечує більш швидке завантаження. Використання CDN ще має перевагу у вигляді зменшення навантаження на сервер. Недоліком є те, що при проблемах із сервером CDN, вебсайт чи вебдодаток можуть залишитися без необхідних ресурсів. Але саме другий спосіб є більш зручним для швидшої інтеграції Bootstrap, особливо для невеликих проєктів чи прототипів. Вибір між підходами залежить від вимого проєкту та швидкості завантаження.

2.4 Висновки до другого розділу

Для розробки було обрано інтегроване середовище Visual Studio 2022, яке містить зручний редактор коду з IntelliSense, компілятором, налагоджувачем. В цьому середовищі зручно розробляти вебплатформу на основі ASP.NET Core через високий рівень підтримки середовищем платформи .NET.

Для розробки backened-частини використовується фреймворк ASP.NET Core, який є досить зручним для розробки вебдодатків. Цей фреймворк має високий рівень продуктивності та має велику кількість NuGet пакетів, які підтримуються. Для створення додатку з ASP.NET Core використовується мова C#. Так, як C# є

мовою ООП, то це дає можливість застосовувати, за потреби, основні принципи: інкапсуляція, наслідування, поліморфізм, абстракція. Мова є строго типізованою, що полегшує розробку шляхом убезпечення від виникнення більшої кількості помилок. Для створення вебплатформи було обрано архітектурний шаблон MVC, який є досить зручним завдяки розділенню на компоненти: модель, контролер, представлення. Таке розділення дозволяє розробляти кожен компонент окремо, незалежно один від одного та з мінімальними ризиками виникнення помилок при редагуванні якогось із компонентів. Для внесення даних та роботи з ними в ASP.NET Core є різні варіанти, але було обрано EF Core та підхід Code-first.

Для створення frontend-частини було використано HTML (для структури) та CSS (для стилізації). Щоб інтерфейс був інтерактивним було використано JavaScript. Ще для зв’язку з елементами моделі та виконання певної логіки було використано Razor-синтаксис. Для швидшого та простішого впровадження адаптивності та стилізації було використано Bootstrap.

Всі розглянуті інструменти забезпечать ефективну розробку сучасної вебплатформи, а інтегроване середовище розробки забезпечить потужні можливості для створення ПЗ.

РОЗДІЛ 3.
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНТЕРАКТИВНОЇ ОСВІТНЬОЇ ВЕБПЛАТФОРМИ ДЛЯ ЗАКЛАДУ ОСВІТИ

3.1 Архітектура вебплатформи

Концептуальна архітектура вебплатформи представляє собою загальну структуру та компоненти системи, які пов’язані між собою.

Вебплатформа включає декілька основних компонентів: реєстрація та вхід, чати, профілі користувачів, управління ролями та доступом. Необхідним також є інтерфейс для взаємодії користувача із цими даними. Всі ці елементи мають бути доречно спроєктовані та зв’язані між собою для того, щоб досягти цілей системи.

На рисунку 3.1 надано діаграму класів, яка є одним з важливих інструментів візуалізації архітектури системи. Вона показує, як різні об’єкти пов’язані між собою завдяки класам, а також ілюструє їхні атрибути та методи взаємодії.

Діаграма класів є частиною об’єктно-орієнтованого підходу до проєктування, вона допомагає розробникам зрозуміти структуру програми та взаємозв’язки між компонентами на високому рівні.

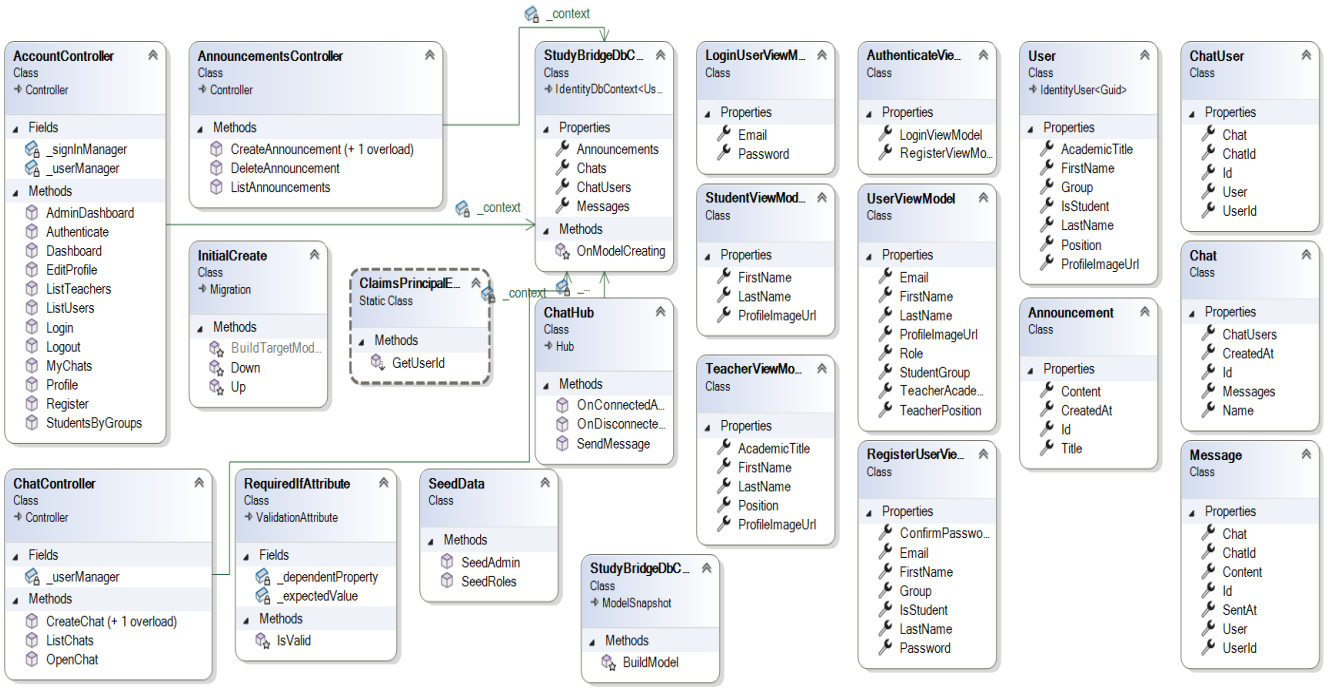

Рисунок 3.1 – Діаграма класів освітньої вебплатформи

Структура файлів і папок (рис. 3.2) є важливим елементом архітектури, оскільки демонструє логічне розподілення компонентів вебплатформи за їх функціоналом. Це розділення покращує читабельність коду та сприяє досягненню модульності та полегшенню майбутнього масштабування та підтримки. Окремі папки для контролерів, моделей, представлень та інфраструктурних компонентів дозволяють чітко визначити функціональну відповідальність кожного модуля та зменшити взаємозалежності між частинами коду.

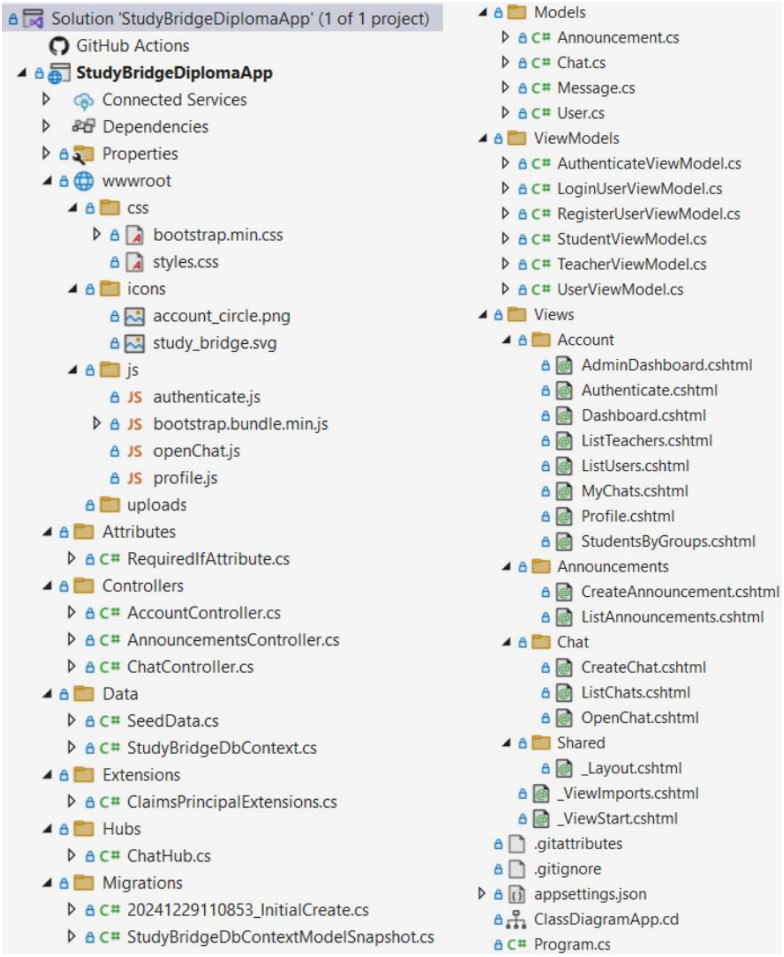

Рисунок 3.2 – Структура файлів і папок освітньої вебплатформи Study Bridge

3.2 Структура бази даних вебплатформи та аналіз її взаємозв’язків

Для зберігання даних про користувачів, їхні чати, ролі та інші елементи освітньої вебплатформи була використана реляційна база даних SQL Server. Ця база даних складається з кількох таблиць, кожна з яких відповідає за певні функціональні аспекти роботи програми (рис. 3.3).

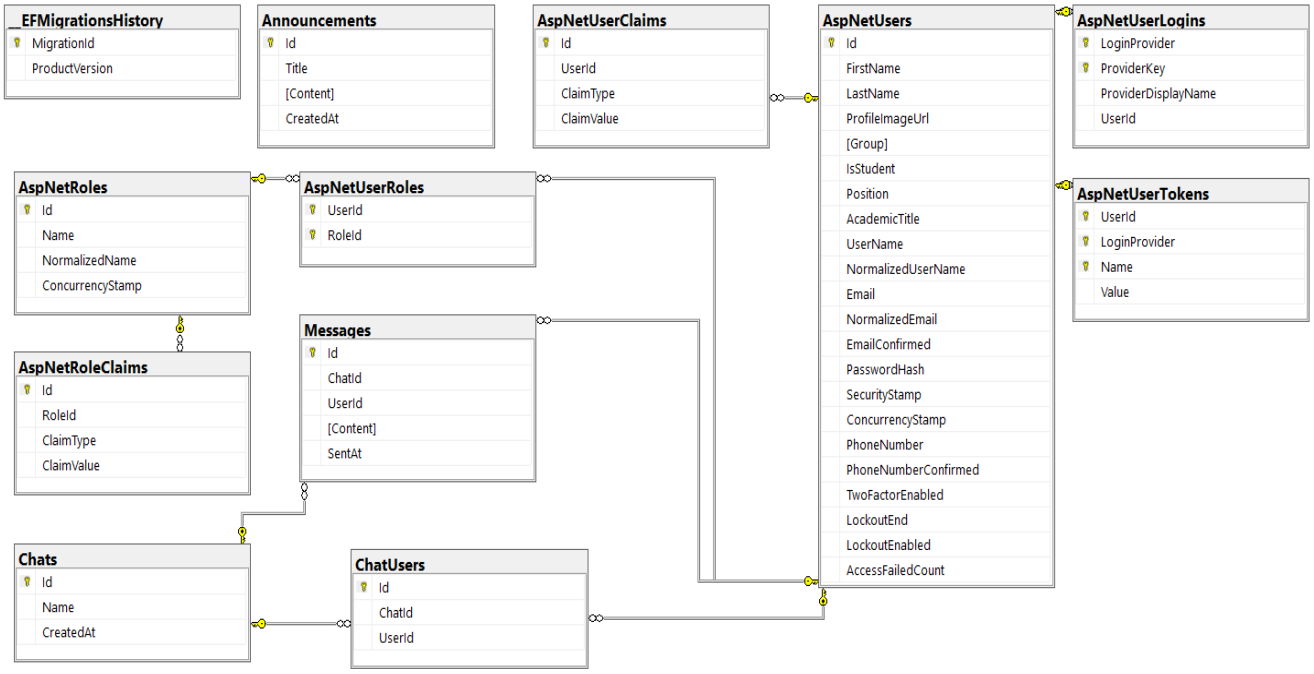

Рисунок 3.3 – Діаграма бази даних вебплатформи Study Bridge

Основними таблицями є AspNetUsers, Chats, ChatUsers, AspNetRoles, Messages та інші, які взаємодіють між собою через зовнішні ключі. Наприклад, таблиця AspNetUsers зберігає інформацію про користувачів (студентів, викладачів, адміністратора), в той час як таблиця ChatUsers містить зв’язки між користувачами (студентами) та чатами до яких вони належать. Це дозволяє організувати правильну ієрархію та забезпечити коректне функціонування взаємодії користувачів з чатом.

Зв’язки між таблицями забезпечують можливість виконання складних запитів, таких як пошук користувачів за групами, фільтрація повідомлень у чатах за різними критеріями та інші операції, які необхідні для роботи вебплатформи.

Для роботи з базою даних було використано EF Core, адже цей фреймворк дозволяє взаємодіяти з базою даних на основі об’єктно-орієнтованого підходу, без необхідності в написанні складних SQL-запитів. Спочатку було застосовано Code-first підхід, тобто створенні моделі для роботи з базами даних та клас контексту, а потім додано міграцію та оновлено базу даних (але це все після створення рядку підключення до бази даних).

Для правильного створення зв’язків між таблицями, які відповідають за функціональні дані чату, було використано метод OnModelCreating. Операції над

базою даних (додавання, оновлення, видалення) виконувалося через LINQ-методи. Для забезпечення доступу до оптимізації запитів використовувалася індексація ключових полів, таких як первинні та зовнішні ключі.

Кожна таблиця відповідає за конкретну сутність:

– AspNetUsers (дані про користувачів);

– AspNetRoles (дані про ролі);

– Announcements (дані про оголошення);

– Chats (дані про чати);

– Messages (дані про повідомлення).

Деякі таблиці слугують для зв’язку між двома іншими таблицями. Наприклад, таблиця ChatUsers використовується для взаємодії таблиць Chats та Users. Ця таблиця містить інформацію про те, до яких чатів під’єднаний користувач. Аналогічно, таблиця AspNetUsersRoles використовується для зберігання інформацію про те, який користувач до якої ролі належить.

Структура бази даних передбачає можливість додавання нових таблиць для розширення функціоналу і без значного впливу на таблиці, які вже існують.

3.3 Алгоритмічні рішення для забезпечення функціональності вебплатформи

В процесі розробки освітньої вебплатформи було використано низку алгоритмічних рішень, які забезпечили її ефективність та зручність для учасників навчального процесу. Алгоритми було розроблено з урахуванням сучасних підходів до побудови вебсистем: модульності, масштабованості, інтеграції з базами даних.

Було створено такі алгоритмічні рішення:

– алгоритм автоматичного додавання студентів до чатів академічних груп;

– алгоритм роботи з базою даних;

– алгоритм авторизації та автентифікації користувачів;

– алгоритм генерації динамічного інтерфейсу з Razor-синтаксисом;

– алгоритм інтеграції оголошень;

– алгоритм обробки помилок.

Алгоритм автоматичного додавання студентів до чатів академічних груп є ключовим для автоматизації процесу формування складу учасників чатів. Перш за все, цей механізм виконує пошук групи студента на основі його реєстраційних даних. По-друге, виконується перевірка, чи є студент учасником відповідного чату. Третім кроком є додавання студентів до чату групи, під час створення чату адміністратором. Та останнім аспектом є оновлення складу чату в реальному часі при реєстрації нового студента групи, для якої вже є чат. Цей підхід сприяє зменшенню навантаження на адміністратора та забезпеченню автоматизованого підходу до формування учасників чату.

Алгоритм роботи з базою даних забезпечує оптимізований підхід створення запитів для швидкого отримання інформації з бази даних. Механізм передбачає використання відносної цілісності між таблицями та реалізації CRUD-операцій. Також гарантується захист від дублювання даних, завдяки перевірці записів на унікальність.

Алгоритм авторизації та автентифікації ґрунтується на ролях користувачів (студент, викладач, адміністратор). Для забезпечення цієї функціональності виконується перевірка достовірності введених облікових даних з використанням хешування паролів. Застосовано принцип розмежування доступу до функціоналу залежно від ролі користувача.

Алгоритм генерації динамічного інтерфейсу з Razor-синтаксисом дозволив генерувати контент сторінок на основі інформації з бази даних та формувати специфічний контент залежно від ролі користувача.

Алгоритм інтеграції оголошень передбачає автоматичне додавання оголошень, які створені адміністратором, до загальнодоступного розділу.

Алгоритм обробки помилок виявляє некоректні запити до серверу та інформує користувача про помилки, з інструкціями по їх усуненню. Цей підхід підвищує надійність платформи.

Застосовані алгоритми забезпечують автоматизацію ключових процесів, вдосконалюють надійність роботи платформи та оптимізують процес комунікації здобувачів освіти.

3.4 Використання системи контролю версій GitHub в розробці проєкту

У процесі розробки інтерактивної освітньої вебплатформи використовувалася система контролю версій Github. Було створено приватний репозиторій, який забезпечував безпечне зберігання коду та управління змінами в проєкті.

Під час розробки вебплатформи використовувалося декілька мов програмування, розмітки та стилізації (рис. 3.4), кожна з яких виконувала важливу роль при побудові функціоналу.

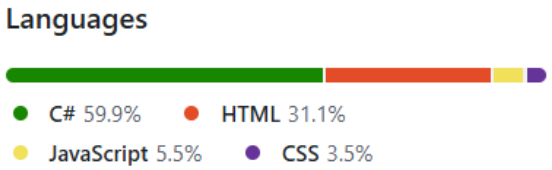

Рисунок 3.4 – Відсотковий розподіл мов програмування та розмітки проєкту

Основну частину серверної логіки реалізовано за допомогою C#, в рамках ASP.NET Core. HTML, CSS та JavaScript використовувалися для створення інтерактивного та адаптивного інтерфейсу.

GitHub є хмарною платформою, яка допомагає зберігати, поширювати код та працювати з іншими програмістами над його розробкою.

Зберігання коду в репозиторії має певні переваги:

– можливість показати свою роботу іншим розробникам;

– відслідковувати та управляти кодом протягом часу роботи над ним;

– можливість отримати відгук про власний код та пропозиції для його покращення;

– здійснювати співпрацю над спільними проєктами без ризику, що зроблені власноруч зміни негативно вплинуть на роботу інших до того, як прийматиметься рішення про впровадження цих змін.

Git, як система контролю версіями, дозволяє розумно відслідковувати зміни, що є досить корисним коли над проєктом працює група людей, яка часто робить аналогічні зміни одночасно.

Для відслідкування змін в проєкті з використанням Git треба:

– створити розгалуження з головної копії файлів;

– робити зміни до файлів незалежно від інших та безпечно, в своєму власному розгалужені («гілці»);

– дозволити Git об’єднати конкретні зміни назад до головної лінії без ризиків вплинути на оновлення інших;

– дозволити Git стежити за змінами, щоб завжди працювати над новішою версією проєкту.

Для роботи в GitHub з командою треба буде регулярно:

1. Отримувати всі останні зміни, які внесли інші (співавтори) до віддаленого сховища за допомогою pull.

2. Переносити власні зміни до віддаленого сховища завдяки push [56].

Застосування системи контролю версій надає можливість швидко повертатися до попередніх версій коду в разі необхідності. Такий підхід сприяє підвищенню якості розробки та дозволяє зменшити ризики, пов’язані з впровадженням нових функцій або виправленням помилок.

Додатково, GitHub інтегрується з багатьма іншими інструментами розробки, що дозволяє автоматизувати частину процесів.

GitHub також надає можливість створювати та вести документацію проєкту прямо в репозиторії, що спрощує комунікацію між розробниками та забезпечує доступ до необхідної інформації про функціонал і структуру коду.

Крім того, в рамках розробки можна застосовувати функції створення issue та відслідковування задач, що дозволить стежити за прогресом виконання різних етапів роботи. Такий підхід сприяє ефективному управлінню проєктом та допомагає вчасно виявляти та усувати проблеми.

Під час роботи над проєктом вебплатформи ключові зміни фіксувалися у вигляді комітів, що дозволило зберегти історію розробки та відслідковувати зміни (рис. 3.5). Коміти містять детальний опис внесених змін, що спрощує роботу над проєктом. Кожен із комітів був присвячений створенню нової можливості чи усуненню наявних на той момент помилок.

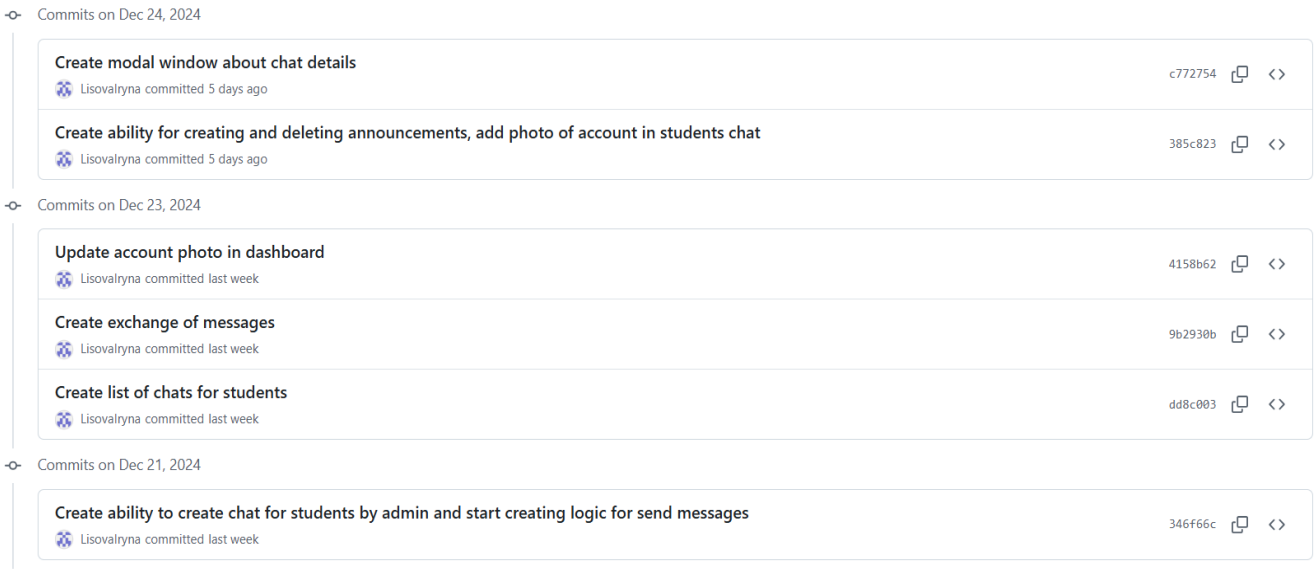

Рисунок 3.5 – Відображення декількох комітів, зроблених при розробці

Відслідковувати зміни можна було в Visual Studio 2022 у вкладці Git Changes. Також там можна було прослідкувати всю історію та виконувати синхронізацію з віддаленим репозиторієм. Загалом, використання Git зробило процес розробки більш структурованим та організованим.

3.5 Реалізація функціоналу інтерактивної освітньої вебплатформи

Основний функціонал інтерактивної освітньої веблатформи Study Bridge:

– реєстрація, вхід користувача;

– особистий профіль користувача та можливість його редагування;

– створення, перегляд та видалення оголошень адміністратором;

– створення чату для студентів однієї групи та перегляд інформації про чат адміністратором;

– чати для обміну текстовими повідомленнями для студентів;

– перегляд студентами та викладачами списку викладачів із інформацією про посади та вчені звання;

– перегляд викладачами списку студентів за групами;

– перегляд списку всіх користувачів адміністратором та їх пошти, групи (для студентів), посади та вченого звання (для викладачів).

Початково було реалізовано можливості для реєстрації та входу користувачів (рис. 3.6). Процес реєстрації дозволяє новим користувачам створити обліковий запис на платформі шляхом вказання необхідних даних: імені, прізвища, електронної пошти, пароля, підтвердження пароля, вибір відмітити помітку для студента, якщо це студент та ввести групу.

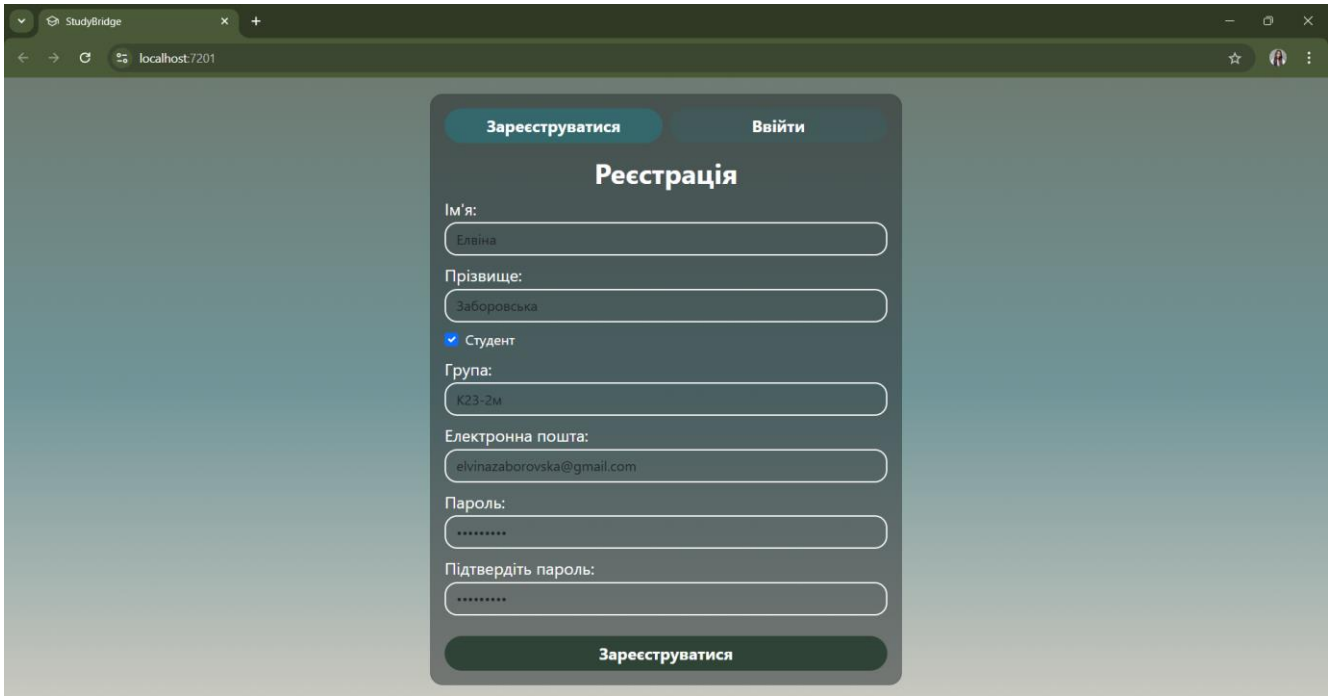

Рисунок 3.6 – Приклад реєстрації користувача (студента)

Для створення функціоналу реєстрації та входу було використано ASP.NET Core Identity, який допомагає керувати користувачами, даними профілів, ролями, підтвердженням електронної пошти тощо. Ідентифікація зазвичай налаштовується за допомогою бази даних SQL Server. Основним пакетом для Identity є Microsoft.AspNetCore.Identity [57].

Після натискання кнопки «Реєстрація» перевіряється правильність полів:

– ім’я та прізвище – починаються з великої літери, містять лише літери (можливе ще використання дефіса та апострофа), кількість символів від 2 до 50 для імені та від 2 до 100 для прізвища;

– шифр групи – від 5 до 8 символів (схема: 1 чи 2 великі літери, дві цифри, дефіс, одна цифра, необов’язкові ще 1 чи дві маленьких літери);

– електронна пошта повинна мати формат електронної пошти;

– пароль – мінімум 8 символів, (мати великі літери, малі літери, цифри, спеціальні символи);

– підтвердження пароля – співпадати має з введеним паролем.

При спробі зареєструватися перевіряється чи не існує такої пошти вже в базі, якщо існує, то користувачу видається помилка і його не реєструє. Всі паролі хешуються стандартними механізмами ASP.NET Core Identity, що забезпечує хороший рівень безпеки.

Після успішної реєстрації користувач може увійти в систему за допомогою електронної пошти та паролю (рис. 3.7). Якщо введена електронна пошта користувача якого не існує в системі, або якщо введений неправильний пароль, – то користувач бачить відповідні повідомлення про помилки.

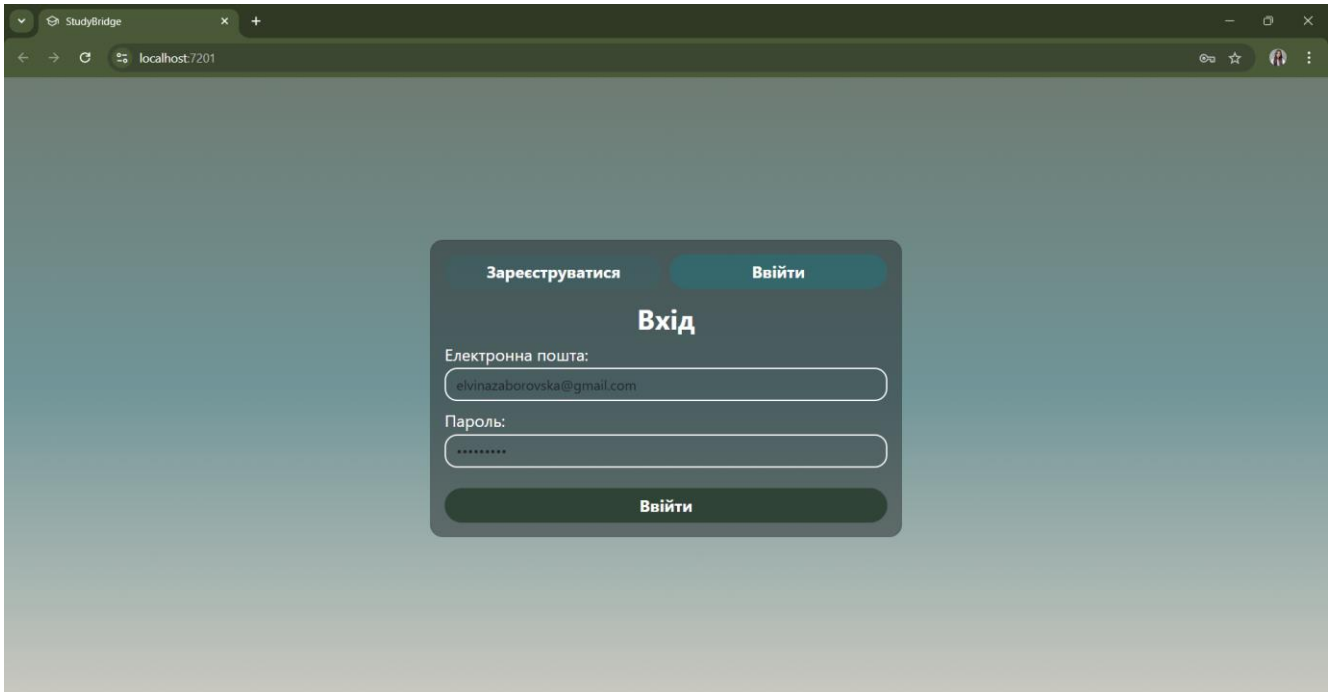

Рисунок 3.7 – Приклад входу користувача

Після успішного входу користувач потрапляє на головну сторінку. На головній сторінці всі користувачі можуть побачити оголошення та перехід до профілю. Інший функціонал на головній сторінці залежить від ролі користувача.

Розроблено три ролі користувача: студент, викладач, адміністратор. Адміністратор створюється при першому запуску системи. Здобувачі освіти та викладачі реєструються через форму реєстрації.

В профілі користувача можна переглянути фото, ім’я, прізвище, пошту, групу (для студентів), посаду та вчене звання (для викладачів). Також є можливість редагувати внесені дані (рис. 3.8).

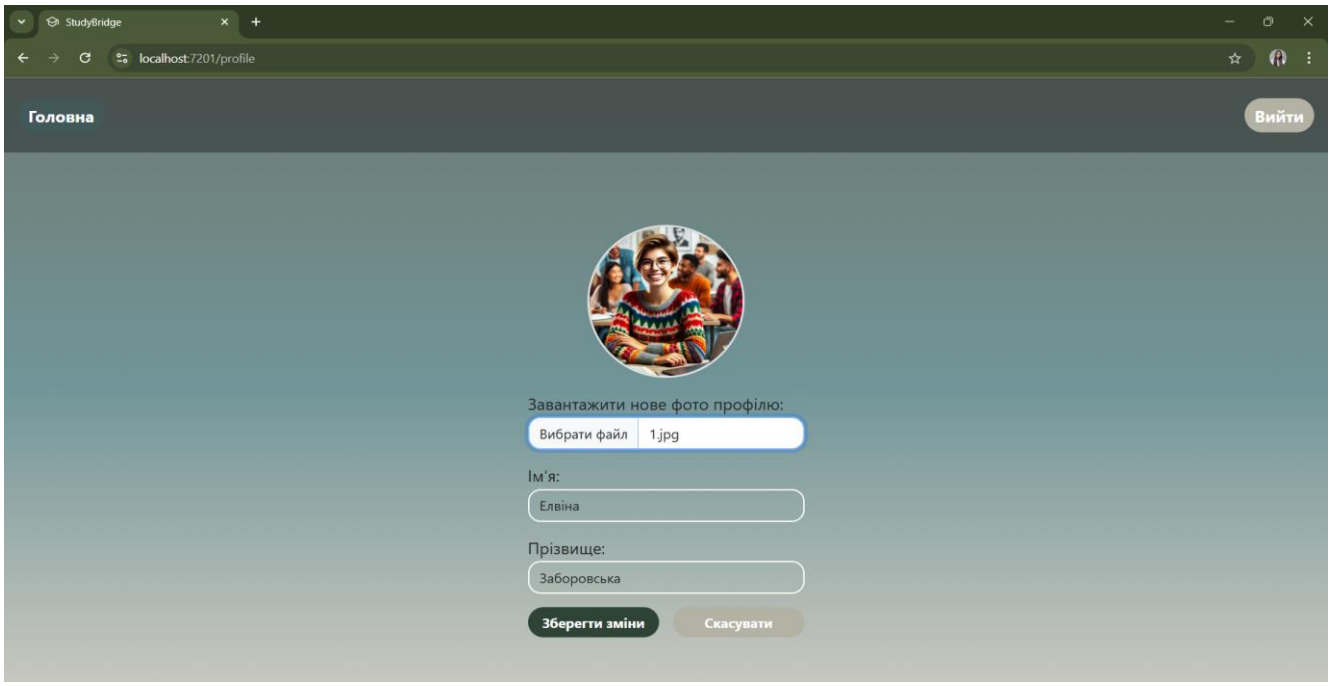

Рисунок 3.8 – Приклад редагування профілю

Для полів ім’я та прізвище зберігаються обмеження на введення тих, чи інших некоректних символів, так як при реєстрації. Фото профілю користувачів буде відображатися у відповідних чатах та списках, де є інформація про цих користувачів і необхідне відображення їхнього фото. Якщо користувач не вносить фото в профілі, то відображається стандартне зображення, яке показує що користувач немає фото профілю. Наявність фото буде перевагою, адже полегшить індинтифікацію цієї людини іншими користувачами.

Для викладача є можливість ще редагувати посаду та вчене звання. Викладачі не вказують ці поля при реєстрації, а мають можливість їх додати в профілі. Якщо викладачі не мають бажання вказувати цю інформацію, то можуть залишити поля не заповненими.

В профілі відображається інформація, яка вже була відредагована (рис. 3.9). Ще в профілі є можливість вийти з нього та перейти на головну сторінку. Також є можливість скасувати дії, якщо користувач передумав редагувати.

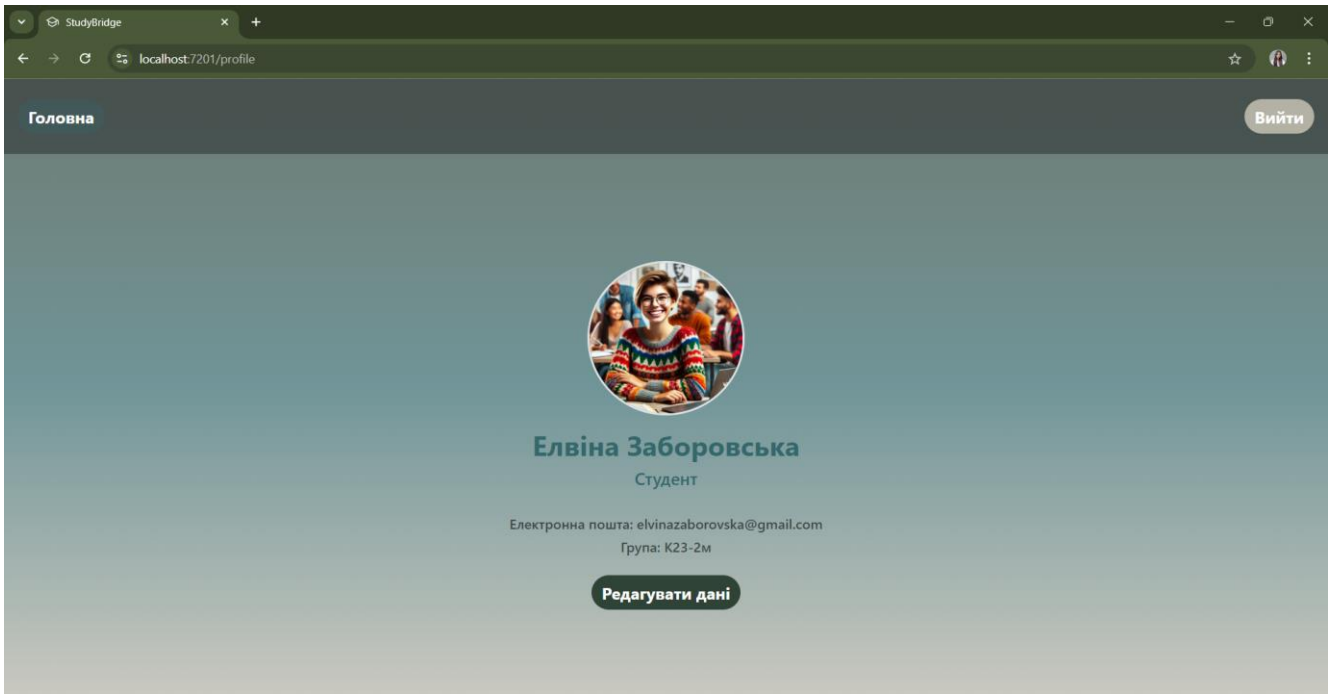

Рисунок 3.9 – Приклад відображення профіля користувача (студента)

Адміністратор на головній сторінці матиме кнопку, яка здійснить його перехід на адмін-панель, де в нього є можливість виконувати певні дії:

– створення чату;

– перегляд чатів;

– створення оголошення;

– перегляд оголошень;

– перегляд користувачів.

Створення оголошення передбачає введення назви оголошення та контенту. Поля є обов’язковими для введення, щоб створити оголошення. Також є можливість натиснути кнопку «Скасувати», щоб відмінити створення. Кнопка переходу на головне сторінку присутня.

При переході до перегляду оголошень (рис. 3.10), адміністратор отримує можливість побачити таблицю з інформацією:

– назва оголошення;

– контент оголошення;

– дата створення оголошення.

Також є можливість видалити те чи інше оголошення (кнопка «Видалити»).

Всі створені адміністратором оголошення будуть відображатися на головній сторінці. Для демонстрації було створено 7 оголошень.

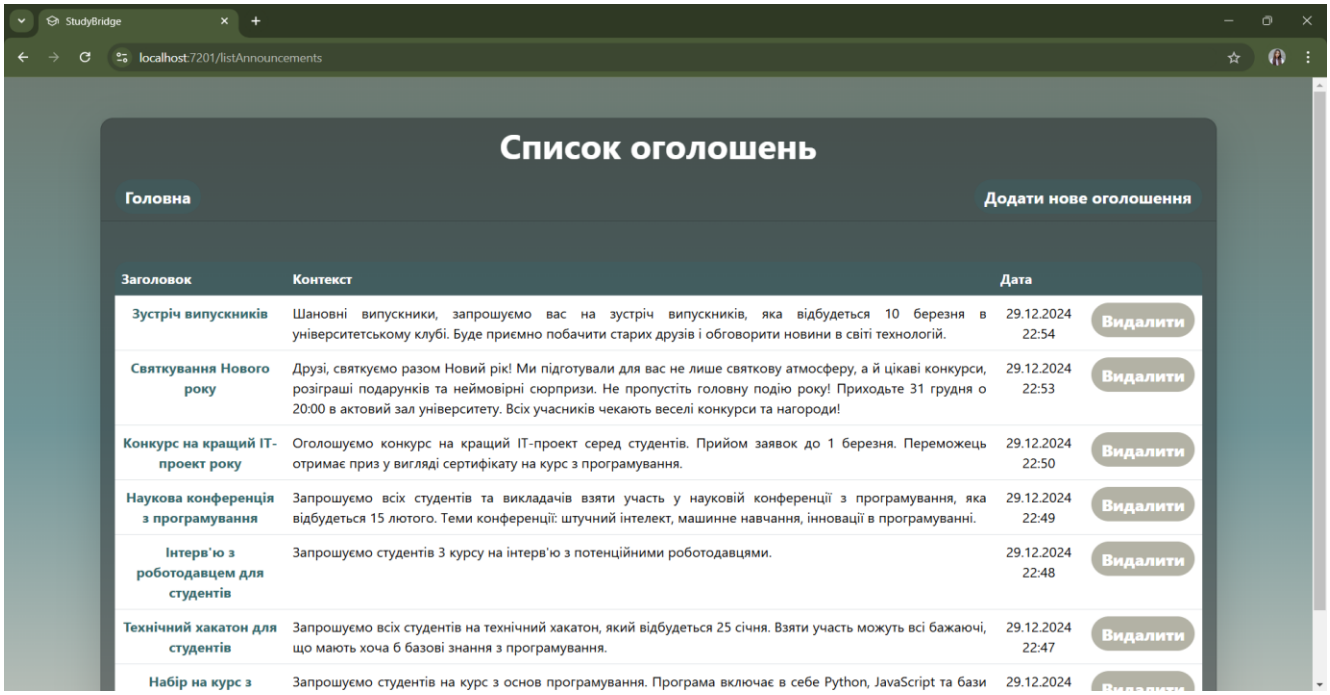

Рисунок 3.10 – Перегляд та можливість видалення оголошень

Також на сторінці перегляду оголошень адміністратор може перейти на головну сторінку або на сторінку додавання нового оголошення.

Адміністратор також може створювати чати для студентів та переглядати інформацію про них. Для створення чату треба ввести шифр групи, вимоги до правильності введення групи аналогічні тим, які необхідні при реєстрації студента. Також є наявна кнопка «Скасувати» та переходу на головну сторінку.

Студенти до цієї групи приєднуються автоматично ті, які вже є в базі даних. Так само нових студентів приєднує автоматично після їх успішної реєстрації.

При перегляді інформації про групи, адміністратор бачить назву групи, дату створення чату та список студентів, які належать до цього чату.

Для демонстрації роботи програми було створено три чати: К23-2м (17 студентів), К23-1м (12 студентів), К19-2 (7 студентів).

Слід відмітити, що імена та прізвища студентів було згенеровано генератором випадкових імен та штучним інтелектом. Електронні пошти та зображення профілів були згенеровані штучним інтелектом.

Рисунок 3.11 – Приклад перегляду створених чатів для студентів груп

Для студентів є можливість обмінюватися повідомленнями в створених чатах.

Для створення функціоналу групових чатів у вебплатформі було використано SignalR – бібліотеку в ASP.Net Core, яка дозволила реалізувати двосторонню комунікацію між клієнтом і сервісом у реальному часі. SignalR забезпечує можливість миттєвого обміну повідомленнями без необхідності оновлювати сторінку.

Етапи впровадження SignalR у проєкт:

1. У проєкті створено спеціальний хаб (ChatHub) – клас, що відповідає за обробку повідомлень і дозволяє відправляти дані від сервера до клієнта і навпаки.

2. Клієнтська логіка реалізована за допомогою JavaScript, що забезпечує підключення до SignalR хабу та обробку вхідних і вихідних повідомлень.

3. Для кожної групи студентів створено окремий чат. Кожен користувач у групі підключається до відповідної групи у SignalR, що дозволяє передавати повідомлення лише її учасникам.

Кожен студент на головній сторінці має посилання на чати. При відкритті, якщо чатів ще для його групи немає, то студент бачить відповідне повідомлення

про відсутність чатів. Якщо чат є, то студент його бачить та бачить кількість учасників. При натисканні на назву чату відкривається відповідний чат (рис. 3.12).

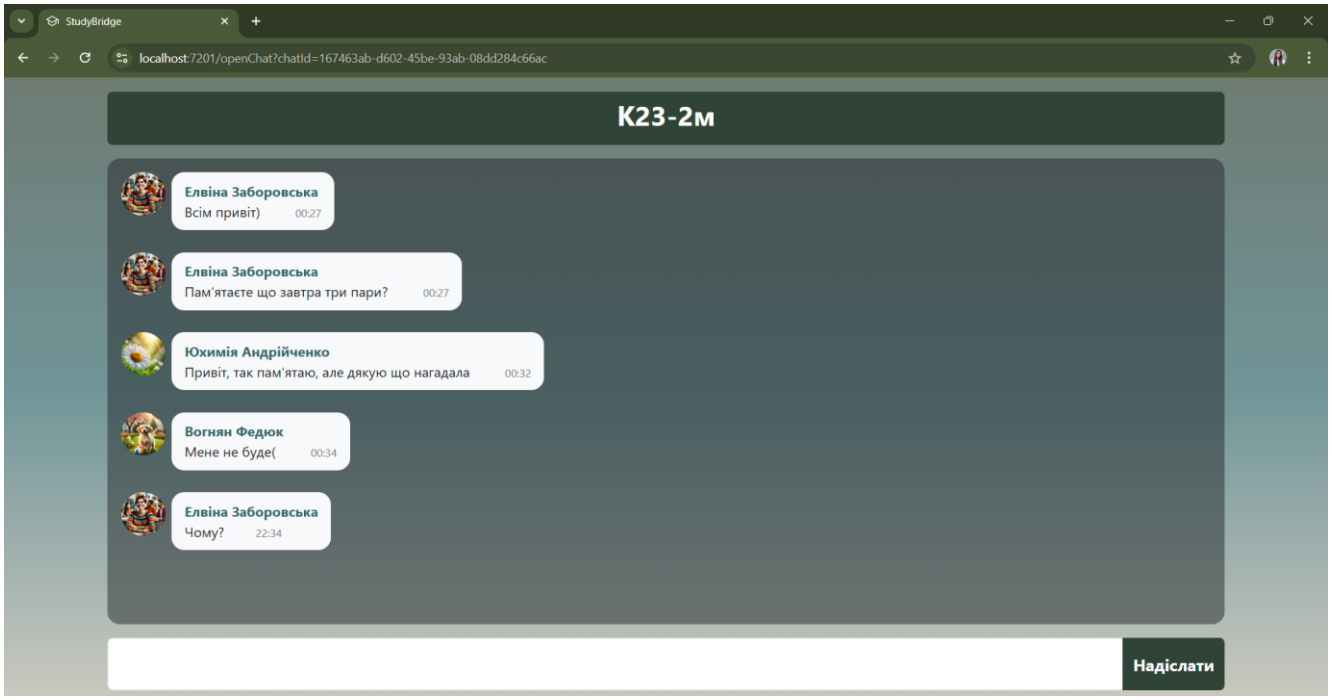

Рисунок 3.12 – Приклад спілкування студентів у чаті групи

Використання SignalR у цьому проєкті дозволило забезпечити миттєвий обмін текстовими повідомленнями між студентами. Це значно покращило інтерактивність та функціональність вебплатформи.

При натисканні на назву чату студенти зможуть побачити перелік учасників чату та його кількість у вигляді модального вікна, яке містить їх імена, прізвища, фото профілів.

Для студентів та викладачів доступна вкладка перегляду списку викладачів (рис. 3.13), де розміщується список викладачів з їх посадами, вченими званнями та фотографіями профілів. Якщо посада та вчене звання не були внесені в профілі, то буде відповідна інформація буде відсутня.

Для демонстрації цього функціоналу було введено дані 5 викладачів, інформація про яких була згенерована штучним інтелектом.

Відображення викладачів здійснюється в порядку їх реєстрації (від найстаріших до найновіших), що дозволяє зручно відстежувати послідовність додавання користувачів.

Функціонал є зручним як для студентів так і для викладачів, оскільки забезпечує доступ до базової інформації про колег та викладацький склад. Завдяки чіткій організації таблиці є можливість швидкого доступу до потрібної інформації.

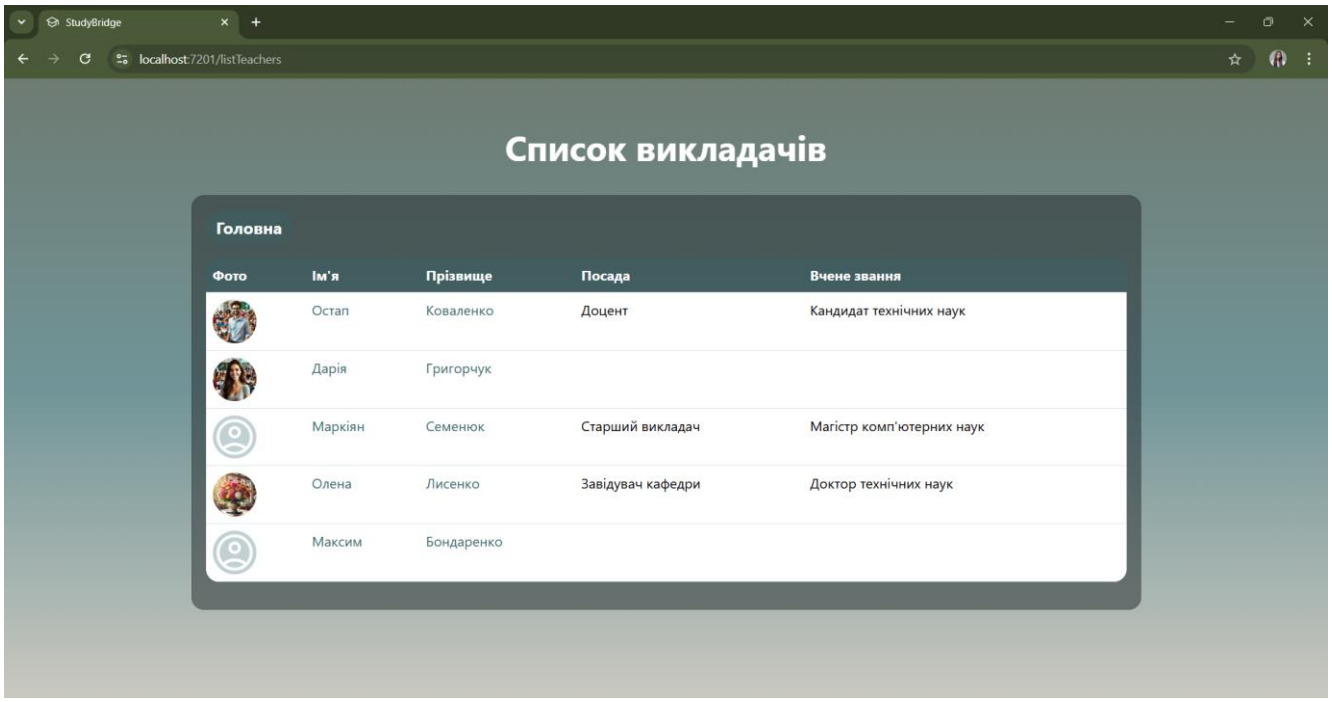

Рисунок 3.13 – Приклад списку викладачів

Для викладачів є можливість переглянути список студентів по групах (рис. 3.14). Для цього треба перейти на вкладку «Список студентів», обрати групу та розгорнути її. Список студентів буде відсортовано в алфавітному порядку.

Ця функція є досить зручною, оскільки дозволяє швидко отримати доступ до актуальної інформації про студентів кожної групи.

Алфавітне сортування спрощує пошук конкретного студента, а структурований вигляд списків дозволяє ефективно працювати з великою кількістю даних.

Така організація сприяє оптимізації часу викладачів, забезпечує легкість використання. Цей функціонал також дозволяє викладачам швидко перевіряти наявність студентів у групах, що важливо для організації робочого процесу (наприклад, для планування групових занять або оцінки явки студентів). Крім того, є можливість перегляду списку студентів у відсортованому вигляді, що мінімізує ймовірність помилок при роботі з великими списками.

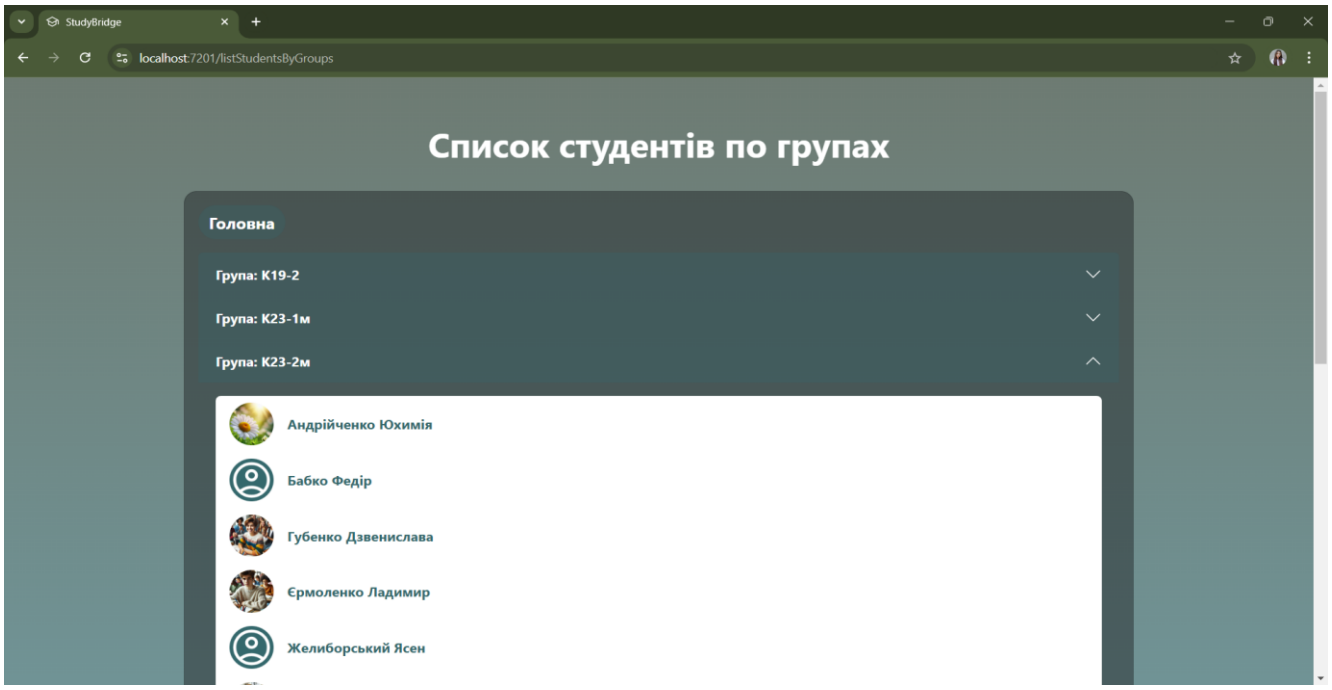

Рисунок 3.14 – Список студентів за групами

Для адміністратора на платформі є можливість перегляду списку всіх користувачів (рис. 3.15). Це дуже зручно, оскільки дозволяє швидко отримувати повну картину щодо користувачів системи: студентів та викладачів.

У списку відображаються основні дані: ім’я, електронна пошта, група (для студентів) або посада та вчене звання (для викладачів). Це значно спрощує управлінські процеси, що дозволяє адміністратору в будь-який момент отримати необхідну інформацію про користувачів. Завдяки такій організації даних адміністратор може швидко виявляти потрібних осіб для розв'язання будь-яких питань, наприклад, щодо доступу до ресурсів або вирішення адміністративних задач. Крім того, це покращує контроль за інформацією в системі та сприяє її актуальності та правильності.

Централізована система управління значно полегшує роботу адміністратора, знижує ризики помилок і покращує загальний досвід користування платформою. Такий функціонал надасть можливість первинного перегляду, а вже при виявленні проблеми у користувача адміністратор відкриє базу даних, щоб вирішити її. Так йому буде простіше працювати, адже в нього вже буде певна інформація за якою можна побувати потрібний запит.

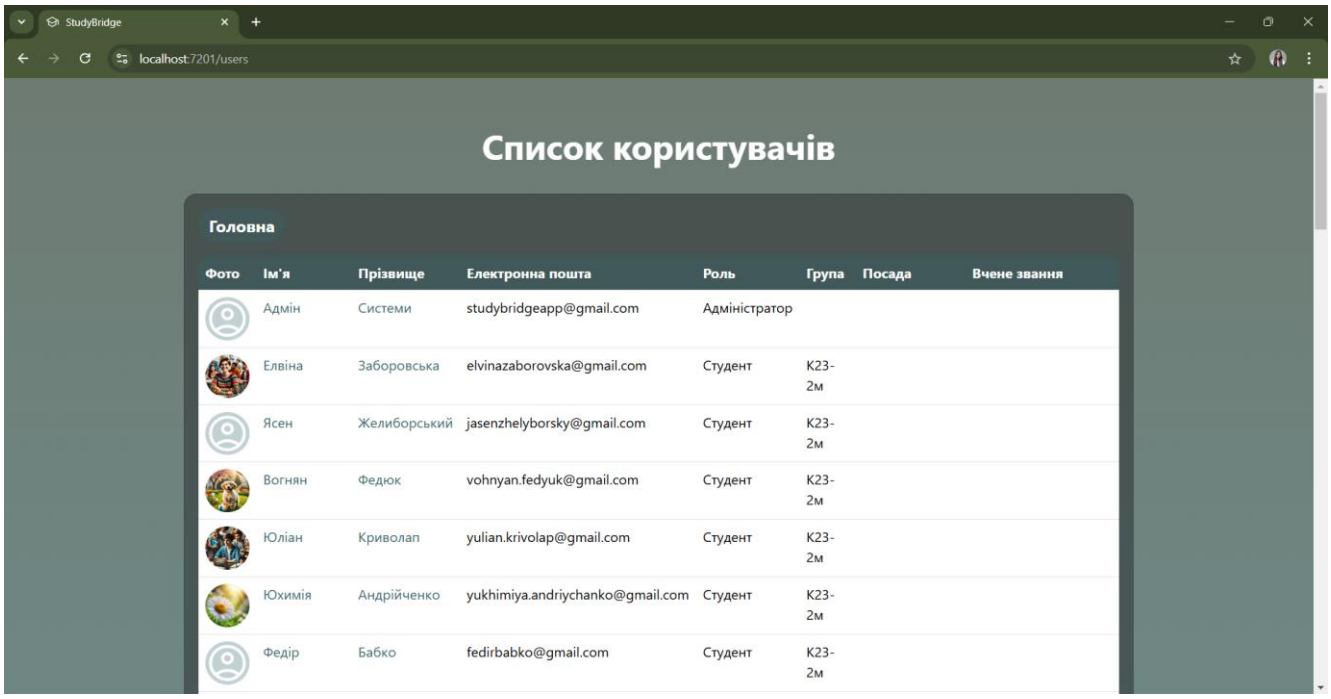

Рисунок 3.15 – Приклад списку користувачів

Вебплатформа Study Bridge є інноваційним рішенням для взаємодії між студентами та викладачами. Платформа має зручний інтерфейс для управління акаунтами, групами, чатами та оголошеннями. Вона сприяє організації освітнього процесу завдяки дозволу студентам ефективно спілкуватися, ділитися інформацією та взаємодіяти в межах своїх груп.

Головна сторінка, яка показана на рис. 1.6 є початковою точкою після успішної реєстрації та відрізняється для користувачів (залежить від ролі). Тобто головна сторінка містить основні оголошення, посилання на профіль (де фото профілю розташоване) та кнопки, які відкриють сторінки із відповідним необхідним функціоналом.

Основна різниця для різних ролей:

– студенти мають посилання на чати та список викладачів;

– викладачі мають посилання на список викладачів та список студентів;

– адміністратор має посилання на адмін-панель.

Інтуїтивно зрозумілий інтерфейс забезпечує простоту використання навіть для нових користувачів, що дозволяє зекономити час на навчання. Функціонал платформи включає реєстрацію та авторизацію, персоналізацію профілів,

створення та перегляд оголошень, а також чати для обміну інформацією. Усі ці функції сприяють підвищенню ефективності освітнього процесу та забезпечують комфортну взаємодію та адміністрування.

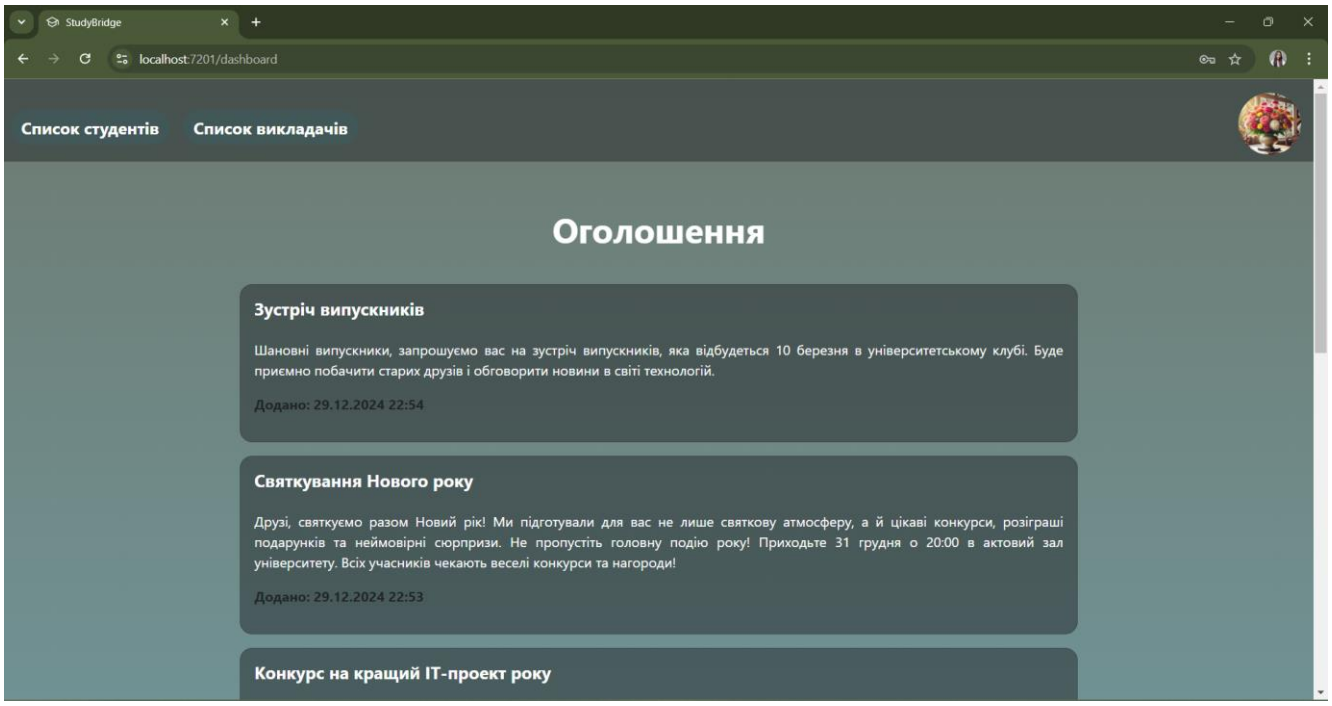

Рисунок 3.16 – Приклад головної сторінки (вхід з роллю викладача)

У майбутньому функціонал вебплатформи Study Bridge може бути розширений шляхом додавання нових можливостей, які дозволять ще ефективніше підтримувати освітній процес.

Однією із пропозицій щодо вдосконалення вебплатформи, які можна реалізувати в системі, є введення механізму підтвердження електронної пошти під час реєстрації, що дозволить гарантувати, що користувач вказав правильну та активну адресу. Це в свою чергу, підвищить безпеку платформи та зменшить ймовірність створення фальшивих акаунтів.

Процес підтвердження пошти зазвичай передбачає надсилання на вказану адресу листа з унікальним посиланням для активації акаунту. Лише після підтвердження пошти користувач отримає доступ до свого профілю. Додаткове підтвердження допоможе знизити ризики несанкціонованого доступу та забезпечити, щоб усі зареєстровані користувачі дійсно мали доступ до вказаних контактних даних.

Також доцільним може бути реалізація функціоналу відновлення пароля: користувачі зможуть запросити скидання пароля через свою електронну пошту у випадку, якщо забудуть його, що допоможе уникнути ситуацій втрати доступу до акаунтів. Завдяки цьому, безпека платформи буде значно підвищена, оскільки несанкціонований доступ буде ускладнений через необхідність мати реальну електронну пошти користувача, а також можливість змінювати пароль лише через підтверджену пошту.

Наступною пропозицією щодо оптимізації функціонування платформи – надання можливості адміністратору створювати чати для спілкування між студентами та викладачами з конкретних предметів. Це дозволить зібрати в одному місці всі комунікації, які стосуються конкретного курсу та дозволило б організувати їх більш ефективно. Корисно було б ввести обмеження на час дії цього чату, наприклад, чат для предмета може бути активним протягом усього семестру. Це дозволить студентам та викладачам взаємодіяти упродовж навчального процесу: задавати питання, обговорювати матеріали курсу та отримувати зворотній зв'язок. Після завершення семестру, чат автоматично стає неактивним для нових повідомлень, всі попередні повідомлення будуть доступні для перегляду. Таким чином, чати зберігатимуть свою цінність, як архіви для подальшого доступу, але нові обговорення вже не матимуть місця в старому чаті.

Доцільним було б також додати можливість розширення функціоналу адміністратора, яке дозволило б здійснювати керування користувачами більш ефективно, на відміну від наявного (рис. 3.17).

Одним з ключових доповнень могла б стати можливість змінювати ролі користувачів, що дало б змогу адміністратору адаптувати доступ до різних функцій системи в залежності від потреб, наприклад, адміністратор може змінювати роль викладача на адміністратора. Це дозволило б безперешкодно оновлювати інформацію про користувачів і зробило б більшу гнучкість в управлінні ролями.

Також було б корисно якби адміністратор також міг змінювати деякі інші параметри профілю, такі як посада та вчене звання для викладачів та групу для студентів.

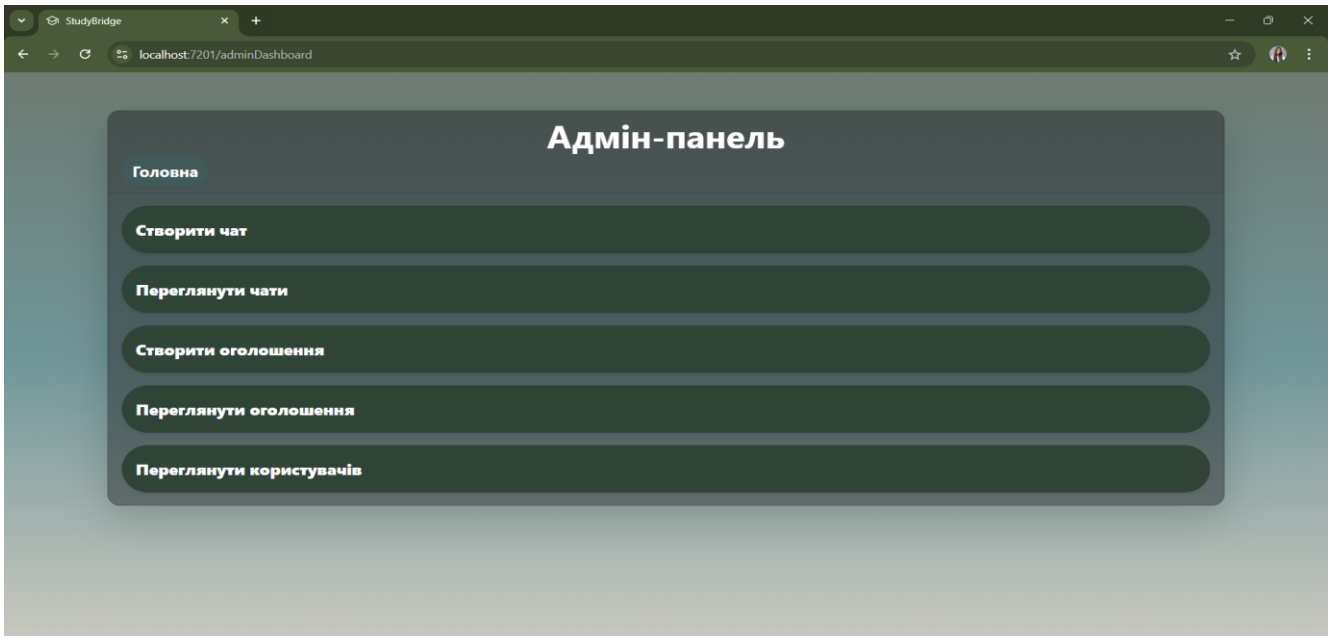

Рисунок 3.17 – Функції, які може виконувати адміністратор безпосередньо в інтерактивній освітній вебплатформі

Іншим важливим доповненням стала б можливість видаляти користувачів з системи. Це корисно, коли потрібно видалити облікові записи студентів або викладачів, які більше не є частиною навчального процесу або у разі порушення правил користування платформою. Видалення користувача могло б включати блокування доступу до системи та очищення даних профілю, якщо це необхідно для безпеки та конфіденційності.

Поліпшенням для освітньої вебплатформи стала б можливість для викладачів створювати завдання та опитування для студентів безпосередньо на платформі, що значно розширило б функціонал і дозволило інтегрувати важливі освітні інструменти без потреби в сторонніх сервісах.

Для підвищення зручності та комунікації в чатах можна було б додати можливість надсилати фото та стікери, що дозволить користувачам додавати більше емоцій у свої повідомлення та створити більш дружню атмосферу в чатах, що особливо важливо для студентів. Можна вбудувати колекцію стікерів, пов’язаних із навчальним процесом, або ж дозволити користувачам додавати свої власні фото для створення більш особистісного підходу у спілкуванні. Ця функціональність підвищила б інтерес студентів до чату і полегшила комунікацію.

Також корисно було б, щоб на платформі кожен студент мав свою індивідуальну статистику, яка включатиме важливу інформацію про виконання завдань та проходження тестів. Це дозволить студентам легко відстежувати та аналізувати свій прогрес і бачити, над чим потрібно працювати. До того ж, корисно мати функціонал для перегляду графіку перескладань на основі їх оцінок чи пропусків предметів. Студент міг би побачити всі майбутні заходи, екзамени або інші важливі події, які потрібно пройти (з датами відповідними). Така система могла би відображати дані у вигляді графіка або списку, що дозволило б студентам більш ефективно планувати своє навчання.

Код освітньої інтерактивної вебплатформи представлено в додатку.

3.6 Регресійний аналіз активності користувачів освітньої платформи

Для забезпечення ефективного функціонування інтерактивної освітньої вебплатформи важливо не лише створити функціонал, а й дослідити поведінку користувачів під час її використання. Одним із важливих аспектів є аналіз активності студентів у групових чатах, що дозволяє оцінити рівень комунікації між учасниками освітнього процесу.

Модель аналізу активності користувачів базується на відстеженні зміни кількості повідомлень у чатах та кількості учасників цих чатів протягом року (початок з вересня). Такий підхід дозволяє виявити динаміку використання платформи та вчасно вносити зміни для вдосконалення платформи та стимулювання взаємодії між користувачами.

Побудована модель відстежує кількість повідомлень надісланих у чаті та кількість учасників, що беруть в них участь протягом року (з вересня до серпня включно). Дані для аналізу є теоретичними, які створені для моделювання поведінки користувачів. Ці дані базується на припущенні, що активність користувачів варіюється залежно від академічного календаря, зокрема зростає під час навчального процесу та знижується під час канікул (рис. 3.18). При можливому запуску проєкту дані могли б бути замінені на реальні, які були б отриманні шляхом

аналізу логів системи, які фіксували б кількість повідомлень у чатах та кількість учасників.

Місяць	Кількість повідомлень в чаті студентів	Кількість студентів у чаті
Вересень	350	31
Жовтень	300	31
Листопад	400	31
Грудень	350	28
Січень	100	28
Лютий	200	32
Березень	300	32
Квітень	450	32
Травень	550	32
Червень	350	29
Липень	50	29
Серпень	30	29

Рисунок 3.18 – Дані для аналізу активності користувачів у чатах

Для виконання дослідження було створено кореляційне поле, що відображає залежність між кількістю повідомлень надісланих у чаті (вісь Y) та кількістю учасників чату (вісь X) протягом року. Дані для аналізу включають показники з вересня по серпень, що дозволило провести спостереження за зміною активності за весь період навчального року. Також було додано лінію тренду для візуалізації загальної тенденції зміни показників протягом часу. Лінія тренду побудована на основі методу найменших квадратів, що дозволяє оцінити лінійну залежність між кількістю учасників та кількістю повідомлень.

На рис. 3.19 продемонстровано результат побудови лінійної моделі залежності кількості повідомлень від кількості учасників у чаті програми протягом року (з вересня до серпня включно). Коефіцієнт детермінації вказує, наскільки добре лінія тренду пояснює варіацію кількості учасників чату та кількості повідомлень. Отримане значення є доволі низьким, що вказує на відсутність значного лінійного зв’язку. Кількість учасників чату не має значного впливу на кількість повідомлень, лише на 30,37% має вплив. Скоріше за все це пов’язано з досить незначною зміною складу групи, адже кількість учасників групи варіюється в залежності від їх зарахування та відрахування (в більшості випадків це незначні зміни). Рівняння лінії тренду містить коефіцієнт нахилу, який свідчить що з кожним додатковим учасником кількість повідомлень зростає на 55.

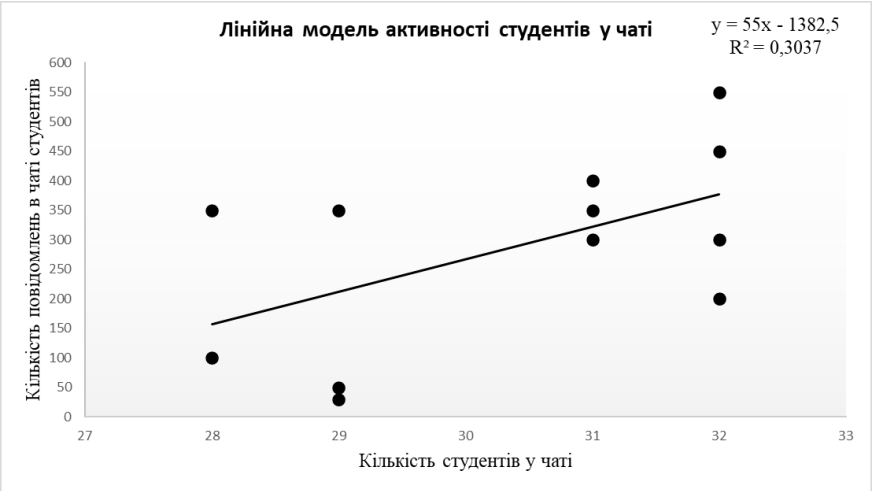

Рисунок 3.19 – Лінійна апроксимація залежності кількості повідомлень від кількості учасників чату

Лінійна модель забезпечує простий і зрозумілий опис даних, однак використання експонентної, поліноміальної та степеневої моделі дозволить врахувати можливу нелінійну залежність та сприятиме більш точному опису тенденцій, що корисно для подальшого вдосконалення програми.

На рис. 3.20 представлена експонентна модель, яка дозволяє врахувати більш складний характер взаємозв’язку між параметрами. Хоча коефіцієнт детермінації є невисоким, що свідчить про наявність факторів які не враховані в моделі, експонентна апроксимація демонструє потенціал для кращого опису тенденцій порівняно з лінійною.

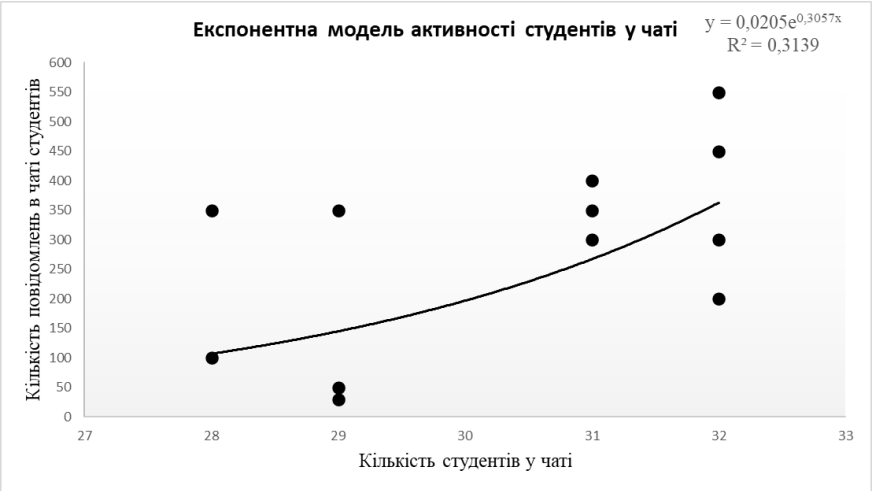

Рисунок 3.20 – Експонентна апроксимація залежності кількості повідомлень від кількості учасників чату

Поліноміальна модель другого ступеня, яка представлена на рис. 3.21, враховує нелінійний характер зв’язку між показниками. Зростання коефіцієнту детермінації порівняно з лінійною та експонентною моделлю демонструє, що поліноміальна апроксимація більш точно відображає тренд.

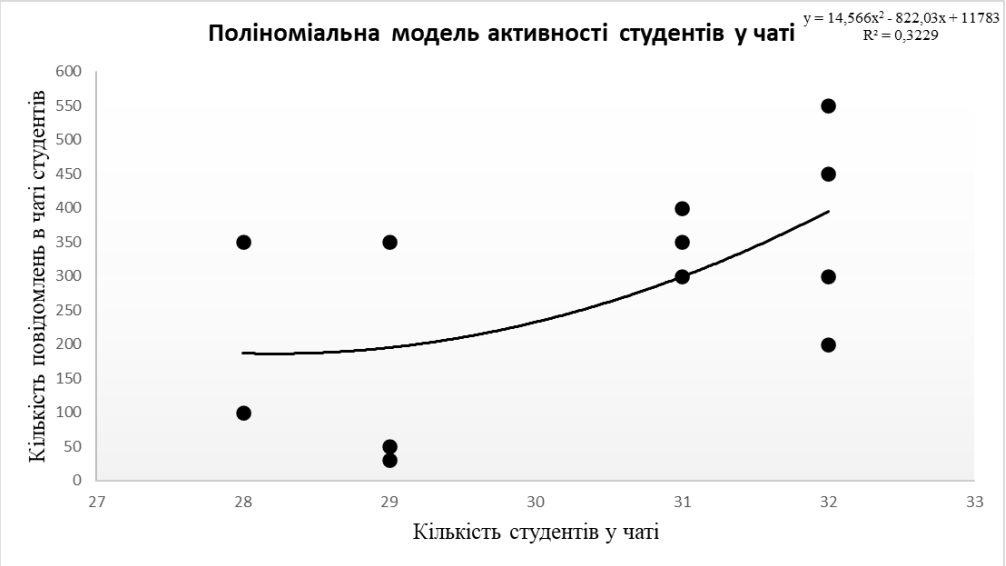

Рисунок 3.21 – Поліноміальна апроксимація залежності кількості повідомлень від кількості учасників чату

Також було побудовано степеневу модель (рис. 3.22), яка забезпечує схожу точність в порівнянні з іншими, але може бути корисною якщо поведінка користувачів має більш різкий або нерівномірний характер залежності.

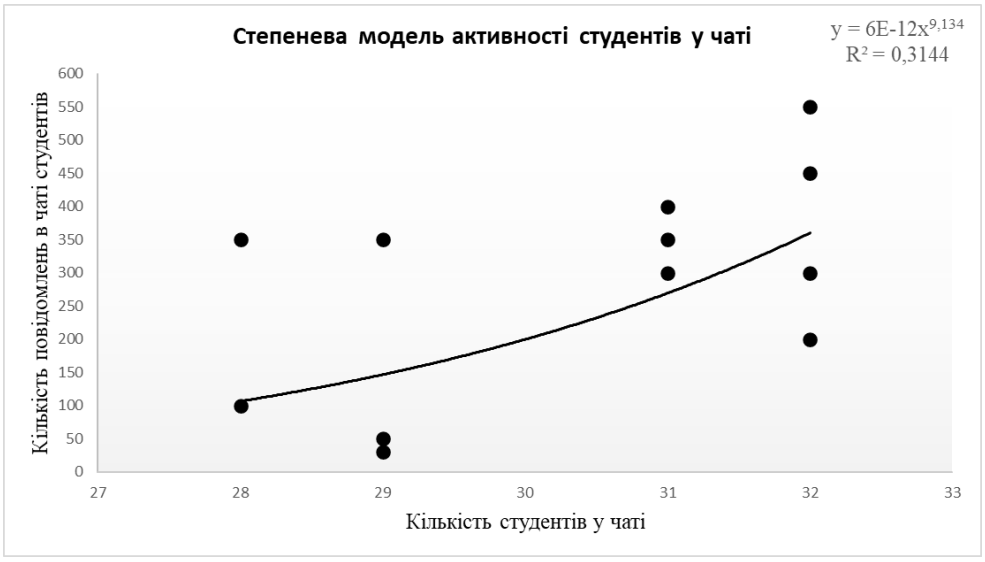

Рисунок 3.22 – Степенева апроксимація залежності кількості повідомлень від кількості учасників чату

В результаті було отримано, що зв’язок між кількістю учасників та кількістю повідомлень існує, але він слабкий, оскільки коефіцієнт детермінації є низьким. Це означає, що збільшення кількості учасників зазвичай веде до збільшення кількості повідомлень, але є інші фактори, які несуть вплив. Прикладами таких факторів можуть бути період обміну повідомленнями (збільшення кількості в період робочих днів, екзаменів, дипломування), наявність світла та Інтернету. Подальший аналіз може включати дослідження впливу інших факторів для кращого прогнозування динаміки обміну повідомленнями.

Основною метою побудови регресійних моделей було створення перевірки доцільності вибору чи відхилення розгляду фактору кількості учасників, як показника впливу на кількість повідомлень. Ця перевірка, в свою чергу, буде корисною для прогнозування майбутнього навантаження на систему.

3.7 Висновки до третього розділу

Аналіз архітектури вебплатформи продемонстрував її гнучкість і масштабованість, а також наскільки ефективно взаємодіють різні компоненти платформи. Окремо було відзначено важливість використання Git для збереження та відслідковування змін в проєкті.

Архітектура бази даних була описана через взаємозв'язки між основними таблицями та їх роллю в функціонуванні платформи. Система на основі Entity Framework Core забезпечує надійну взаємодію між компонентами вебплатформи та базою даних, що забезпечує збереження та обробку даних користувачів, чатових повідомлень та іншої важливої інформації.

Завдяки реалізації основних функціональних можливостей платформи: реєстрації, авторизації, створення чатів, адміністрування даних користувачів та інших була досягнута стабільна робота основних процесів. Користувачі можуть легко взаємодіяти в межах груп, обмінюватися текстовими повідомленнями, а також ефективно працювати з інформацією про викладачів та студентів.

Наявність ряду покращень дозволила б підвищити зручність та безпеку.

Можливі майбутні варіанти вдосконалення проєкту:

– підтвердження пошти;

– можливість зміни ролей адміністратором;

– створення тимчасових чатів з предмету з викладачами (на період семестру);

– відображення власної статистики студенту та інформацію про розклад, перескладання, заходи;

– можливість надсилання в чатах стікерів, фото та файлів;

– можливість створення викладачами завдань та опитувань та їх оцінювання.

Подальший розвиток програми може включати нові функції для розширення можливостей взаємодії між студентами та викладачами.

Для тестування платформи було введено дані про користувачів (студентів, викладачів) та оголошення, які були згенеровані штучним інтелектом. Аналогічно було виконано для генерації зображень для фото акаунтів.

У процесі розробки освітньої вебплатформи реалізовано алгоритм автоматичного додавання студентів до чатів, який спрощує адміністрування.

Для аналізу активності користувачів побудовано математичні моделі (лінійну, експонентну, поліноміальну, степеневу), які дозволяють виявити рівень залежності між кількістю учасників і кількістю повідомлень. Отримані результати можна використовувати для подальшої оптимізації програми.

Загалом, платформа вже має потужний функціонал із потенціалом для її подальшого вдосконалення та оптимізації, що дозволить значно покращити досвід користувачів і розширити її можливості.

ВИСНОВКИ

В ході роботи було проаналізовано наукові праці та інші джерела інформації на тему ефективності застосування інтерактивних вебплатформ в галузі освіти, що в підсумку показало, що ці інструменти позитивно впливають на успішність здобувачів.

У результаті виконання дипломної роботи було розроблено інтерактивну освітню вебплатформу Study Bridge, що забезпечує ефективну взаємодію між студентами, викладачами та адміністраторами системи. Створена платформа має чітко структурований функціонал, який включає такі складові: реєстрацію користувачів, створення чатів для студентів, перегляд списків викладачів і студентів, а також адміністрування користувачів та їх даних. Реалізація функціоналу вебплатформи дозволила створити зручну та зрозумілу систему для всіх учасників освітнього процесу.

Особливу увагу було приділено архітектурі програмного забезпечення та базі даних, що дозволило забезпечити цілісність даних, їх обробку та доступність у будь-який момент. Використання ASP.NET Core та Entity Framework Core для побудови вебплатформи дозволило досягти її високої ефективності та безпеки. Інтеграція з Git та використання системи контролю версій забезпечили зручне збереження та відстеження змін у процесі розробки, що дозволило мінімізувати можливі помилки та спростило роботу. Використання GitHub для управління репозиторієм дозволило організувати ефективну взаємодію та зберігання коду на етапах розробки проєкту.

Важливою частиною роботи була також розробка та реалізація пропозицій щодо вдосконалення програми та підвищення безпеки, а також зручності використання платформи. А саме:

– підтвердження пошти;

– розширення можливостей для адміністраторів і викладачів;

– додавання мультимедійних елементів в чати.

Ці зміни підвищать функціональність і комфорт використання платформи.

Під час розробки освітньої вебплатформи було реалізовано низку алгоритмічних рішень, що забезпечують ефективність та функціональність системи. Зокрема, створено алгоритм автоматичного додавання студентів до чатів, який враховує як уже зареєстрованих користувачів групи, так і тих, які будуть додані до групи в майбутньому. Цей алгоритм забезпечить підтримку актуального складу учасників чату та оптимізує комунікацію в освітньому середовищі.

Додатково було проведено дослідження взаємозв’язку між кількістю учасників у чатах та кількістю повідомлень, які вони надсилають. Для цього побудовано декілька математичних моделей: лінійну, експонентну, поліноміальну, степеневу. На основі побудованих моделей та розрахунку коефіцієнта детермінації було зроблено висновок про наявність слабкого зв’язку між показниками. Доцільною була б побудова аналогічних моделей для інших факторів.

Реалізована платформа вже має потужний функціонал для вирішення основних завдань у сфері освітніх інтерфейсів. Подальший розвиток проєкту може бути спрямований на розширення функціоналу, що зробить платформу більш конкурентоспроможною та ефективною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Education in a post-COVID world: Nine ideas for public action. 2023. URL: https://www.unesco.org/en/articles/education-post-covid-world-nine-ideas-public-action (дата звернення: 18.12.2024).

2. Лісова І.О. Оцінка продуктивності роботи з базою даних з використанням Entity Framework Core під час розробки застосунків. Матеріали міжнародної науково-практичної конференції «Економіко-правові та управлінсько-технологічні виміри сьогодення: молодіжний погляд». Том 3. 2023. С. 262. URL: https://drive.google.com/file/d/1izpGVnAb8lBPpf7AwQGmTZcLeyzZTQ7_/view (дата звернення: 18.12.2024).

3. Що таке діджиталізація та які переваги вона надає бізнесу. 2023. URL: https://gigacloud.ua/blog/navchannja/scho-take-didzhitalizacija-ta-jaki-perevagi-vona-nadae-biznesu (дата звернення: 18.12.2024).

4. Difference between web platform, web page and apps. 2024. URL: https://improvitz.com/difference-between-web-platform-web-page-and-apps/ (дата звернення: 18.12.2024).

5. Anna Shevtsova. Evolution of the Web: from Roots to the Horizons of Web 3.0. 2024. URL: https://outstaffyourteam.com/articles/web-30 (дата звернення: 18.12.2024).

6. Різниця між HTTP та HTTPs. URL: https://hostiq.ua/wiki/ukr/http-https/ (дата звернення: 18.12.2024).

7. Difference between URI та URL. URL: https://danielmiessler.com/blog/difference-between-uri-url/ (дата звернення: 18.12.2024).

8. Bhaval Patel. Web Application Architecture Explained: A Comprehensive Guide. 2024. URL: https://www.spaceotechnologies.com/blog/web-application-architecture/ (дата звернення: 18.12.2024).

9. What are eCommerce platforms? 2024. URL: https://artelogic.net/blog/what-are-ecommerce-platforms/ (дата звернення: 18.12.2024).

10. Social Media Platforms. URL: https://www.wix.com/encyclopedia/definition/social-media-platforms (дата звернення: 18.12.2024).

11. Elizabeth Aguiar Chacón. Educational platform. URL: https://www.iseazy.com/glossary/educational-platform/ (дата звернення: 18.12.2024).

12. Shoriful Islam. What is a Business Website? 2024. URL: https://www.linkedin.com/pulse/what-business-website-shoriful-islam-vxarc (дата звернення: 19.12.2024).

13. What Is Cloud Computing? URL: https://www.salesforce.com/ca/cloud-computing/ (дата звернення: 19.12.2024).

14. What are microservices? URL: https://microservices.io/ (дата звернення: 19.12.2024).

15. У чому суть контейнеризації. 2023. URL: https://foxminded.ua/shcho-take-konteineryzatsiia/ (дата звернення: 19.12.2024).

16. Hosting – Definition. URL: https://blog.it-planet.com/en/glossar/hosting/ (дата звернення: 19.12.2024).

17. What is Web Hosting? How Does it Work. 2023. URL: https://nescom.co.ke/what-is-web-hosting (дата звернення: 19.12.2024).

18. Purpose of Web Design. URL: https://channelcreative.co.uk/web-design-northampton/insight/purpose-of-web-design/ (дата звернення: 19.12.2024).

19. An Introduction to Online Platforms and their Role in the Digital Transformation. Primary impacts of online platforms on economies and societies. 2019. С. 29. URL: https://www.oecd.org/content/dam/oecd/en/publications/reports/2019/05/an-introduction-to-online-platforms-and-their-role-in-the-digital-transformation_970fc377/53e5f593-en.pdf#page3

20. How do online platforms shape our lives and businesses? Brochure. 2019. URL: https://digital-strategy.ec.europa.eu/en/library/how-do-online-platforms-shape-our-lives-and-businesses-brochure (дата звернення: 19.12.2024).

21. The Importance of User Experience (UX) in Website Planning. URL: https://www.websiteplanningguide.com/importance-ux-in-website-planning/ (дата звернення: 19.12.2024).

22. Education Processes: What Are the Challenges to Optimization? 2023. URL: https://www.sydle.com/blog/education-processes-636c19402b1bb867e7b3b454 (дата звернення: 19.12.2024).

23. Benjamin Callahan. Top 11 Benefits & Advantages of a Student Management System. 2024. URL: https://classconnectpro.com/blog/benefits-of-a-student-management-system (дата звернення: 19.12.2024).

24. Moddle features. URL: https://docs.moodle.org/405/en/Features (дата звернення: 20.12.2024).

25. How Personalized Learning Platforms Boost Employee Engagement in 2025. 2024. URL: https://disprz.ai/blog/personalized-learning-platforms-boost-employee-engagement (дата звернення: 20.12.2024).

26. What Is Coursera? 2024. URL: https://www.coursera.org/articles/what-is-coursera (дата звернення: 20.12.2024).

27. A personalized learning resource for all ages. URL: https://www.khanacademy.org/about (дата звернення: 20.12.2024).

28. Ganesh Mukundan. Google Classroom: Everything you need to know. 2024. URL: https://hiverhq.com/blog/google-classroom-basics (дата звернення: 20.12.2024).

29. Ping Wang, Teng Ma, Li-Bo Liu, Chao Shang, Ping An, Yi-Xue Xue. A Comparison of the Effectiveness of Online Instructional Strategies Optimized With Smart Interactive Tools Versus Traditional Teaching for Postgraduate Students. Neuroscience, Learning and Educational Psychology. Vol. 12. 2021. URL: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2021.747719/full

30. Majumdar Subhechha, Bhowmik Tanmay. A Study on Effectiveness of Interactive Tools in the Online Teaching-Learning Process. SSRN. 2022. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4036562&

31. Cameron Schwartz, Jason Wahby, Geko Ezekiel Jimenez, Kemal Koymen, Hunter Doll, Mohammed Mahmoud. An educational perspective on online learning platforms. International Conference on Computational Science and Computational Intelligence (CSCI). 2022. URL: https://american-cse.org/csci2022-ieee/pdfs/CSCI2022-2lPzsUSRQukMlxf8K2x89I/202800c116/202800c116.pdf

32. Дія.Освіта – національна едьютейнмент освітня платформа актуальних знань та навичок. URL: https://osvita.diia.gov.ua/about (дата звернення: 20.12.2024).

33. Любомир Сірський. Яке майбутнє електронного навчання?! 2024. URL: https://kwiga.com/ua/blog/yake-majbutnye-elektronnogo-navchannya (дата звернення: 20.12.2024).

34. Розробка ПЗ. URL: https://smbsoft.com.ua/software-development/ (дата звернення: 21.12.2024).

35. Visual Studio 2022. URL: https://visualstudio.microsoft.com/vs/ (дата звернення: 21.12.2024).

36. What is ASP.NET Core? URL: https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core (дата звернення: 21.12.2024).

37. Mark J. Price. C# 12 and .NET 8 Modern Cross-Platform Development Fundamentals. С. 629-630.

38. ASP.NET Core Advantages and Disadvantages. 2021. URL: https://redwerk.com/blog/asp-net-core-pros-and-cons/ (дата звернення: 21.12.2024).

39. Nataliya Revutska. What is Object-Oriented Programming (oop)? Explaining four major principles. 2023. URL: https://career.softserveinc.com/en-us/stories/what-is-object-oriented-programming-oop-explaining-four-major-principles (дата звернення: 21.12.2024).

40. C# Tutorial. 2024. URL: https://www.geeksforgeeks.org/csharp-programming-language/ (дата звернення: 21.12.2024).

41. What is C#? URL: https://csharp-station.com/ (дата звернення: 21.12.2024).

42. PYPL PopularitY of Programming Language. 2024. URL: https://pypl.github.io/PYPL.html (дата звернення: 21.12.2024).

43. TIOBE Index for December 2024. 2024. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 21.12.2024).

44. What is C# Programming? A Beginner's Guide | Pluralsight. 2024. URL: https://www.pluralsight.com/resources/blog/software-development/everything-you-need-to-know-about-c-?__cf_chl_tk (дата звернення: 22.12.2024).

45. Isha Sharma. ASP.NET Core MVC Fundamentals. 2022. URL: https://medium.com/@isharma6624/asp-net-core-mvc-fundamentals-558464455830 (дата звернення: 22.12.2024).

46. MVC Framework – Introduction. URL: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm (дата звернення: 22.12.2024).

47. Part 7. Introduction to the MVC (Model-View-Controller) Pattern. 2023. URL: https://javarush.com/en/groups/posts/en.2536.part-7-introduction-to-the-mvc-model-view-controller-pattern (дата звернення: 22.12.2024).

48. Safia Nahhas. MVC Architecture from Maintenance Quality Attributes Perspective. International Journal of Computer Science and Security (IJCSS). Vol. 15. 2021. URL: https://cscjournals.org/manuscript/Journals/IJCSS/Volume15/Issue5/IJCSS-1636.pdf

49. Welcome to Learn Dapper. URL: https://www.learndapper.com/ (дата звернення: 22.12.2024).

50. ADO.NET. URL: https://docs.simego.com/data-sync/connectors/databases/ado-net/ (дата звернення: 22.12.2024).

51. Entity Framework Core. URL: https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx (дата звернення: 22.12.2024).

52. Adam Hayes. HyperText Markup Language (HTML): What It Is and How It Works. 2024. URL: https://www.investopedia.com/terms/h/html.asp (дата звернення: 22.12.2024).

53. The History of the Development of HTML. 2016. URL: https://vertex-academy.com/tutorials/en/html-history/ (дата звернення: 22.12.2024).

54. Олег Дутченко. Що таке JavaScript та навіщо його вчити? URL: https://wezom.academy/ua/chto-takoe-javascript-i-zachem-ego-uchit/ (дата звернення: 22.12.2024).

55. ASP.NET Razor – Markup. URL: https://www.w3schools.com/asp/razor_intro.asp (дата звернення: 22.12.2024).

56. About GitHub and Git. URL: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git (дата звернення: 23.12.2024).

57. Introduction to Identity on ASP.NET Core. URL: https://learn.microsoft.com/uk-ua/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio (дата звернення: 25.12.2024).

ДОДАТОК

Код програми:

AccountController.cs:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using StudyBridgeDiplomaApp.Data;
using StudyBridgeDiplomaApp.Models;
using StudyBridgeDiplomaApp.ViewModels;
using System.Text.RegularExpressions;

namespace StudyBridgeDiplomaApp.Controllers;

public class AccountController(StudyBridgeDbContext dbContext, UserManager<User>
userManager, SignInManager<User> signInManager) : Controller
{
    private readonly StudyBridgeDbContext _context = dbContext;
    private readonly UserManager<User> _userManager = userManager;
    private readonly SignInManager<User> _signInManager = signInManager;

    [HttpGet]
    [Route("/")]
    public IActionResult Authenticate()
    {
        return View();
    }

    [HttpPost]
    [Route("/register")]
    public async Task<IActionResult> Register(AuthenticateViewModel model)
    {
        ViewBag.ActiveForm = "register";

        if (!ModelState.IsValid)
            return View("Authenticate", model);

        var existingUser = await
_userManager.FindByEmailAsync(model.RegisterViewModel!.Email);
        if (existingUser != null)
        {
            ModelState.AddModelError("RegisterViewModel.Email", "Користувач з такою
електронною поштою вже існує");
            return View("Authenticate", model);
        }

        var user = new User
        {
            FirstName = model.RegisterViewModel!.FirstName,
            LastName = model.RegisterViewModel!.LastName,
            Email = model.RegisterViewModel!.Email,
            NormalizedEmail = model.RegisterViewModel.Email.ToUpper(),
            UserName = model.RegisterViewModel.Email,
            NormalizedUserName = model.RegisterViewModel.Email.ToUpper(),
            IsStudent = model.RegisterViewModel!.IsStudent,
            Group = model.RegisterViewModel!.IsStudent ? model.RegisterViewModel.Group :
null
        };
```

```
        var result = await _userManager.CreateAsync(user,
model.RegisterViewModel.Password);
        if (result.Succeeded)
        {
            var role = model.RegisterViewModel.IsStudent ? "Студент" : "Викладач";
            await _userManager.AddToRoleAsync(user, role);

            if (model.RegisterViewModel.IsStudent &&
!string.IsNullOrEmpty(model.RegisterViewModel.Group))
            {
                var groupChats = await _context.Chats
                    .Where(chat => chat.Name == model.RegisterViewModel.Group)
                    .ToListAsync();

                foreach (var chat in groupChats)
                {
                    var chatUser = new ChatUser
                    {
                        ChatId = chat.Id,
                        UserId = user.Id
                    };
                    _context.ChatUsers.Add(chatUser);
                }

                await _context.SaveChangesAsync();
            }

            await _signInManager.SignInAsync(user, isPersistent: true);

            return RedirectToAction("Dashboard");
        }

        return RedirectToAction("Dashboard");
    }

    [HttpPost]
    [Route("/login")]
    public async Task<IActionResult> Login(AuthenticateViewModel model)
    {
        ViewBag.ActiveForm = "login";

        if (!ModelState.IsValid)
            return View("Authenticate", model);

        var user = await _userManager.FindByEmailAsync(model.LoginViewModel!.Email);
        if (user == null)
        {
            ModelState.AddModelError("LoginViewModel.Email", "Користувача з такою
електронною поштою не знайдено");
            return View("Authenticate", model);
        }
        var result = await _signInManager.PasswordSignInAsync(user,
model.LoginViewModel.Password, isPersistent: true, lockoutOnFailure: false);
        if (!result.Succeeded)
        {
            ModelState.AddModelError("LoginViewModel.Password", "Невірний пароль");
            return View("Authenticate", model);
        }

        return RedirectToAction("Dashboard");
    }

    [HttpGet]
    [Authorize]
    [Route("/dashboard")]
```

```
    public async Task<IActionResult> Dashboard()
    {
        var userEmail = User.Identity?.Name;
        if (userEmail == null)
            return RedirectToAction("Authenticate");

        var user = await _context.Users
            .AsNoTracking()
            .FirstOrDefaultAsync(user => user.Email == userEmail);
        if (user == null)
            return RedirectToAction("Authenticate");
        if (string.IsNullOrEmpty(user.ProfileImageUrl))
            user.ProfileImageUrl = "account_circle.png";

        var announcements = await _context.Announcements
            .OrderByDescending(announcement => announcement.CreatedAt)
            .ToListAsync();

        ViewBag.Announcements = announcements;

        return View(user);
    }

    [HttpGet]
    [Authorize(Roles = "Студент")]
    [Route("/myChats")]
    public async Task<IActionResult> MyChats()
    {
        var userId = _userManager.GetUserId(User);
        var chats = await _context.Chats
            .Include(chat => chat.ChatUsers)
            .Where(chat => chat.ChatUsers.Any(chatuser => chatuser.UserId ==
Guid.Parse(userId!)))
            .ToListAsync();

        return View(chats);
    }

    [Authorize(Roles = "Адміністратор")]
    [Route("/adminDashboard")]
    public IActionResult AdminDashboard()
    {
        return View();
    }

    [HttpPost]
    [Authorize]
    [Route("/logout")]
    public async Task<IActionResult> Logout()
    {
        await _signInManager.SignOutAsync();
        return RedirectToAction("Authenticate");
    }

    [HttpGet]
    [Authorize]
    [Route("/profile")]
    public async Task<IActionResult> Profile()
    {
        var userEmail = User.Identity?.Name;
        if (userEmail == null)
            return RedirectToAction("Authenticate");

        var user = await _context.Users
            .AsNoTracking()
```

```
            .FirstOrDefaultAsync(user => user.Email == userEmail);
        if (user == null)
            return RedirectToAction("Authenticate");
        if (string.IsNullOrEmpty(user.ProfileImageUrl))
            user.ProfileImageUrl = "account_circle.png";

        return View(user);
    }

    [HttpPost]
    [Authorize]
    [Route("/profile/edit")]
    public async Task<IActionResult> EditProfile(IFormFile? profileImage, string
firstName, string lastName, string? position, string? academicTitle)
    {
        string pattern = @"^[A-ZA-ЯIЇЄҐ][a-za-яiїєґ'-]*$";

        var userEmail = User.Identity?.Name;
        if (userEmail == null)
            return RedirectToAction("Authenticate");

        var user = await _context.Users.FirstOrDefaultAsync(user => user.Email ==
userEmail);
        if (user == null)
            return RedirectToAction("Authenticate");

        if (profileImage != null && profileImage.Length > 0)
        {
            if (!string.IsNullOrEmpty(user.ProfileImageUrl))
            {
                var oldProfileImagePath = Path.Combine(Directory.GetCurrentDirectory(),
"wwwroot/uploads", user.ProfileImageUrl!);
                if (System.IO.File.Exists(oldProfileImagePath))
                    System.IO.File.Delete(oldProfileImagePath);
            }

            var newFileName = Guid.NewGuid().ToString() +
Path.GetExtension(profileImage.FileName);
            var newProfileImagePath = Path.Combine(Directory.GetCurrentDirectory(),
"wwwroot/uploads", newFileName);
            using (var stream = new FileStream(newProfileImagePath, FileMode.Create))
            {
                await profileImage.CopyToAsync(stream);
            }
            user.ProfileImageUrl = newFileName;
        }

        if (!Regex.IsMatch(firstName, pattern) || !Regex.IsMatch(lastName, pattern))
            return RedirectToAction("Profile");

        if (!string.IsNullOrEmpty(firstName))
            user.FirstName = firstName;

        if (!string.IsNullOrEmpty(lastName))
            user.LastName = lastName;

        if (!string.IsNullOrEmpty(position))
            user.Position = position;

        if (!string.IsNullOrEmpty(academicTitle))
            user.AcademicTitle = academicTitle;

        _context.Users.Update(user);
        await _context.SaveChangesAsync();
```

```
            return RedirectToAction("Profile");
        }

        [HttpGet]
        [Authorize(Roles = "Студент, Викладач")]
        [Route("/listTeachers")]
        public async Task<IActionResult> ListTeachers()
        {
            var allUsers = await _context.Users.ToListAsync();
            var teachers = new List<TeacherViewModel>();

            foreach (var user in allUsers)
            {
                var roles = await _userManager.GetRolesAsync(user);
                if (roles.Contains("Викладач"))
                {
                    teachers.Add(new TeacherViewModel
                    {
                        FirstName = user.FirstName,
                        LastName = user.LastName,
                        ProfileImageUrl = string.IsNullOrEmpty(user.ProfileImageUrl) ?
"/icons/account_circle.png"
                        : $"/uploads/{user.ProfileImageUrl}",
                        Position = user.Position,
                        AcademicTitle = user.AcademicTitle
                    });
                }
            }

            return View(teachers);
        }

        [HttpGet]
        [Authorize(Roles = "Адміністратор")]
        [Route("/users")]
        public async Task<IActionResult> ListUsers()
        {
            var allUsers = await _context.Users.ToListAsync();
            var users = new List<UserViewModel>();

            foreach (var user in allUsers)
            {
                var roles = await _userManager.GetRolesAsync(user);
                var role = roles.FirstOrDefault();

                users.Add(new UserViewModel
                {
                    FirstName = user.FirstName,
                    LastName = user.LastName,
                    Email = user.Email!,
                    ProfileImageUrl= string.IsNullOrEmpty(user.ProfileImageUrl) ?
"/icons/account_circle.png"
                        : $"/uploads/{user.ProfileImageUrl}",
                    Role = role,
                    StudentGroup = user.Group,
                    TeacherPosition = user.Position,
                    TeacherAcademicTitle = user.AcademicTitle
                });
            }

            return View(users);
        }

        [HttpGet]
        [Authorize(Roles = "Викладач")]
```

```
    [Route("/listStudentsByGroups")]
    public async Task<IActionResult> StudentsByGroups()
    {
        var students = await _context.Users
            .Where(user => user.IsStudent && !string.IsNullOrEmpty(user.Group))
            .OrderBy(user => user.Group)
            .ThenBy(user => user.LastName)
            .ThenBy(user => user.FirstName)
            .ToListAsync();

        var groupedStudents = students
            .GroupBy(student => student.Group)
            .ToDictionary(
                group => group.Key!,
                group => group.Select(student => new StudentViewModel
                {
                    FirstName = student.FirstName,
                    LastName = student.LastName,
                    ProfileImageUrl = string.IsNullOrEmpty(student.ProfileImageUrl)
                        ? "/icons/account_circle.png"
                        : $"/uploads/{student.ProfileImageUrl}"
                }).ToList()
            );

        return View(groupedStudents);
    }
}
```

AnnouncementsController.cs:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using StudyBridgeDiplomaApp.Data;
using StudyBridgeDiplomaApp.Models;

namespace StudyBridgeDiplomaApp.Controllers;

[Authorize(Roles = "Адміністратор")]
[Route("/announcements")]
public class AnnouncementsController(StudyBridgeDbContext context) : Controller
{
    private readonly StudyBridgeDbContext _context = context;

    [HttpGet("/createAnnouncement")]
    public IActionResult CreateAnnouncement()
    {
        return View();
    }

    [HttpPost("/createAnnouncement")]
    public async Task<IActionResult> CreateAnnouncement(string title, string content)
    {
        if (string.IsNullOrEmpty(title) || string.IsNullOrEmpty(content))
            return View();

        var announcement = new Announcement
        {
            Title = title,
            Content = content,
            CreatedAt = DateTime.UtcNow
        };
        _context.Announcements.Add(announcement);
        await _context.SaveChangesAsync();
```

```
        return RedirectToAction("ListAnnouncements");
    }

    [HttpGet("/listAnnouncements")]
    public async Task<IActionResult> ListAnnouncements()
    {
        var announcements = await _context.Announcements
            .OrderByDescending(announcement => announcement.CreatedAt)
            .ToListAsync();

        return View(announcements);
    }

    [HttpPost("/deleteAnnouncement/{id}")]
    public async Task<IActionResult> DeleteAnnouncement(Guid id)
    {
        var announcement = await _context.Announcements.FindAsync(id);
        if (announcement == null)
            return NotFound();
        _context.Announcements.Remove(announcement);
        await _context.SaveChangesAsync();

        return RedirectToAction("ListAnnouncements");
    }
}
```

ChatController.cs:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using StudyBridgeDiplomaApp.Data;
using StudyBridgeDiplomaApp.Models;

namespace StudyBridgeDiplomaApp.Controllers;

[Authorize]
[Route("/chats")]
public class ChatController(StudyBridgeDbContext context, UserManager<User> userManager)
: Controller
{
    private readonly StudyBridgeDbContext _context = context;
    private readonly UserManager<User> _userManager = userManager;

    [HttpGet]
    [Authorize(Roles = "Адміністратор")]
    [Route("/createChat")]
    public IActionResult CreateChat()
    {
        return View();
    }

    [HttpPost]
    [Authorize(Roles = "Адміністратор")]
    [Route("/createChat")]
    public async Task<IActionResult> CreateChat(string groupName)
    {
        if (string.IsNullOrEmpty(groupName))
            return View();

        var existingChat = await _context.Chats
            .FirstOrDefaultAsync(c => c.Name == groupName);
        if (existingChat != null)
        {
            TempData["ErrorEditNameMessage"] = "Чат з такою групою вже існує";
```

```
            return View();
        }

        var chat = new Chat { Name = groupName };
        _context.Chats.Add(chat);
        await _context.SaveChangesAsync();

        var students = await _userManager.Users
            .Where(user => user.IsStudent && user.Group == groupName)
            .ToListAsync();
        foreach (var student in students)
        {
            var chatUser = new ChatUser
            {
                ChatId = chat.Id,
                UserId = student.Id
            };
            _context.ChatUsers.Add(chatUser);
        }

        await _context.SaveChangesAsync();

        return RedirectToAction("ListChats");
    }

    [HttpGet]
    [Authorize(Roles = "Адміністратор")]
    [Route("/listChats")]
    public async Task<IActionResult> ListChats()
    {
        var chats = await _context.Chats
            .Include(chat => chat.ChatUsers)
            .ThenInclude(chatUser => chatUser.User)
            .Where(chat => chat.ChatUsers.All(chatUser => chatUser.User != null))
            .ToListAsync();

        return View(chats);
    }

    [HttpGet]
    [Authorize(Roles = "Студент")]
    [Route("/openChat")]
    public async Task<IActionResult> OpenChat(Guid chatId)
    {
        var chat = await _context.Chats
            .Include(chat => chat.Messages)
            .Include(chat => chat.ChatUsers)
            .ThenInclude(chatUser => chatUser.User)
            .FirstOrDefaultAsync(chat => chat.Id == chatId);
        if (chat == null)
            return NotFound();

        return View(chat);
    }
}
```

Announcement.cs:

```
namespace StudyBridgeDiplomaApp.Models;

public class Announcement
{
    public Guid Id { get; set; } = Guid.NewGuid();
    public string Title { get; set; } = string.Empty;
    public string Content { get; set; } = string.Empty;
    public DateTime CreatedAt { get; set; } = DateTime.UtcNow;
```

```
}
```

Chat.cs:

```
namespace StudyBridgeDiplomaApp.Models;

public class Chat
{
    public Guid Id { get; set; }
    public required string Name { get; set; }
    public DateTime CreatedAt { get; set; } = DateTime.Now;

    public List<ChatUser> ChatUsers { get; set; } = [];

    public List<Message> Messages { get; set; } = [];
}

public class ChatUser
{
    public Guid Id { get; set; }

    public Guid ChatId { get; set; }
    public Chat? Chat { get; set; }

    public Guid UserId { get; set; }
    public User? User { get; set; }
}
```

Message.cs:

```
namespace StudyBridgeDiplomaApp.Models;

public class Message
{
    public Guid Id { get; set; }

    public Guid ChatId { get; set; }
    public Chat? Chat { get; set; }

    public Guid UserId { get; set; }
    public User? User { get; set; }

    public required string Content { get; set; }
    public DateTime SentAt { get; set; } = DateTime.Now;
}
```

User.cs:

```
using Microsoft.AspNetCore.Identity;

namespace StudyBridgeDiplomaApp.Models;

public class User : IdentityUser<Guid>
{
    public required string FirstName { get; set; }
    public required string LastName { get; set; }
    public string? ProfileImageUrl { get; set; }

    public required string? Group { get; set; }
    public bool IsStudent { get; set; }

    public string? Position { get; set; }
    public string? AcademicTitle { get; set; }
}
```

AuthenticateViewModel.cs:

```
namespace StudyBridgeDiplomaApp.ViewModels;
```

```
public class AuthenticateViewModel
{
    public RegisterUserViewModel? RegisterViewModel { get; set; }
    public LoginUserViewModel? LoginViewModel { get; set; }
}
```

LoginUserViewModel.cs:

```
using System.ComponentModel.DataAnnotations;

namespace StudyBridgeDiplomaApp.ViewModels;

public class LoginUserViewModel
{
    [Required(ErrorMessage = "Необхідно ввести електронну пошту")]
    [EmailAddress(ErrorMessage = "Невірний формат електронної пошти")]
    [Length(6, 320, ErrorMessage = "Електронна пошта повинна мати мінімум 6 символів,
максимум - 320")]
    public required string Email { get; set; }

    [Required(ErrorMessage = "Необхідно ввести пароль")]
    public required string Password { get; set; }
}
```

RegisterUserViewModel.cs:

```
using System.ComponentModel.DataAnnotations;
using StudyBridgeDiplomaApp.Attributes;

namespace StudyBridgeDiplomaApp.ViewModels;

public class RegisterUserViewModel
{
    [Required(ErrorMessage = "Необхідно ввести ім'я")]
    [Length(2, 50, ErrorMessage = "Ім'я повинно мати мінімум 2 символа, максимум - 50")]
    [RegularExpression(@"^[A-ZA-ЯІЇЄҐ][a-za-яіїєґ'-]*$",
        ErrorMessage = "Ім'я може містити лише літери, дефіс або апостроф і повинно
починатися з великої літери")]
    public required string FirstName { get; set; }

    [Required(ErrorMessage = "Необхідно ввести прізвище")]
    [Length(2, 100, ErrorMessage = "Прізвище повинно мати мінімум 2 символа, максимум -
100")]
    [RegularExpression(@"^[A-ZA-ЯІЇЄҐ][a-za-яіїєґ'-]*$",
        ErrorMessage = "Прізвище може містити лише літери, дефіс або апостроф і повинно
починатися з великої літери")]
    public required string LastName { get; set; }

    public bool IsStudent { get; set; }

    [RequiredIf("IsStudent", true, ErrorMessage = "Необхідно ввести групу")]
    [Length(5,8,ErrorMessage = "Шифр групи повинен мати мінімум 5 символів, максимум -
8")]
    [RegularExpression(@"^[A-ZA-Я]{1,2}\d{2}-\d[a-яіїє]{0,2}$",
        ErrorMessage = "Група введена невірно (невірний формат)")]
    public string? Group { get; set; }

    [Required(ErrorMessage = "Необхідно ввести електронну пошту")]
    [EmailAddress(ErrorMessage = "Невірний формат електронної пошти")]
    [Length(6, 320, ErrorMessage = "Електронна пошта повинна мати мінімум 6 символів,
максимум - 320")]
    public required string Email { get; set; }

    [Required(ErrorMessage = "Необхідно ввести пароль")]
```

```
    [Length(8, 128, ErrorMessage = "Пароль повинен мати мінімум 8 символів, максимум -
128")]
    [RegularExpression("^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)(?=.*[!@#$%^&*])[A-Za-
z\\d!@#$%^&*]{8,128}$",
        ErrorMessage = "Пароль повинен містити щонайменше одну велику літеру, одну малу
літеру, одну цифру та один спеціальний символ")]
    public required string Password { get; set; }

    [Required(ErrorMessage = "Необхідно ввести підтвердження пароля")]
    [Compare("Password", ErrorMessage = "Паролі не співпадають")]
    public required string ConfirmPassword { get; set; }
}
```

StudentViewModel.cs:

```
namespace StudyBridgeDiplomaApp.ViewModels;

public class StudentViewModel
{
    public required string FirstName { get; set; }
    public required string LastName { get; set; }
    public string? ProfileImageUrl { get; set; }
}
```

TeacherViewModel.cs:

```
namespace StudyBridgeDiplomaApp.ViewModels;

public class TeacherViewModel
{
    public required string FirstName { get; set; }
    public required string LastName { get; set; }
    public string? ProfileImageUrl { get; set; }
    public string? Position { get; set; }
    public string? AcademicTitle { get; set; }
}
```

UserViewModel.cs:

```
namespace StudyBridgeDiplomaApp.ViewModels;

public class UserViewModel
{
    public required string FirstName { get; set; }
    public required string LastName { get; set; }
    public required string Email { get; set; }
    public string? ProfileImageUrl { get; set; }
    public string? Role { get; set; }
    public string? StudentGroup { get; set; }
    public string? TeacherPosition { get; set; }
    public string? TeacherAcademicTitle { get; set; }
}
```

ChatHub.cs:

```
using Microsoft.AspNetCore.SignalR;
using StudyBridgeDiplomaApp.Data;
using StudyBridgeDiplomaApp.Models;

namespace StudyBridgeDiplomaApp.Hubs;

public class ChatHub(StudyBridgeDbContext context) : Hub
{
    private readonly StudyBridgeDbContext _context = context;

    public async Task SendMessage(string messageContent, Guid chatId, Guid userId)
    {
        var user = await _context.Users.FindAsync(userId);
```

```
        if (user == null) return;

        string profileImageUrl;
        if (!string.IsNullOrEmpty(user.ProfileImageUrl) &&
            File.Exists(Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/uploads",
user.ProfileImageUrl)))
            profileImageUrl = $"/uploads/{user.ProfileImageUrl}";
        else
            profileImageUrl = "/icons/account_circle.png";

        var message = new Message
        {
            Id = Guid.NewGuid(),
            ChatId = chatId,
            UserId = userId,
            Content = messageContent,
            SentAt = DateTime.UtcNow
        };
        _context.Messages.Add(message);
        await _context.SaveChangesAsync();

        await Clients.Group(chatId.ToString()).SendAsync(
            "ReceiveMessage",
            $"{user.FirstName} {user.LastName}",
            messageContent,
            message.SentAt.ToString("HH:mm"),
            profileImageUrl
        );
    }

    public override async Task OnConnectedAsync()
    {
        var chatId = Context.GetHttpContext()!.Request.Query["chatId"];
        await Groups.AddToGroupAsync(Context.ConnectionId, chatId!);
        await base.OnConnectedAsync();
    }

    public override async Task OnDisconnectedAsync(Exception? exception)
    {
        var chatId = Context.GetHttpContext()!.Request.Query["chatId"];
        await Groups.RemoveFromGroupAsync(Context.ConnectionId, chatId!);
        await base.OnDisconnectedAsync(exception);
    }
}
```

ClaimsPrincipalExtensions.cs:

```
using System.Security.Claims;

namespace StudyBridgeDiplomaApp.Extensions;

public static class ClaimsPrincipalExtensions
{
    public static Guid GetUserId(this ClaimsPrincipal user)
    {
        var userId = user.FindFirst(ClaimTypes.NameIdentifier)?.Value;
        return Guid.TryParse(userId, out var id) ? id : Guid.Empty;
    }
}
```

StudyBridgeDbContext.cs:

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using StudyBridgeDiplomaApp.Models;
```

```
namespace StudyBridgeDiplomaApp.Data;

public class StudyBridgeDbContext(DbContextOptions<StudyBridgeDbContext> options) :
IdentityDbContext<User, IdentityRole<Guid>, Guid>(options)
{
    public DbSet<Chat> Chats { get; set; }
    public DbSet<ChatUser> ChatUsers { get; set; }
    public DbSet<Message> Messages { get; set; }
    public DbSet<Announcement> Announcements { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.Entity<ChatUser>()
            .HasOne(chatUser => chatUser.Chat)
            .WithMany(chat => chat.ChatUsers)
            .HasForeignKey(chatUser => chatUser.ChatId)
            .OnDelete(DeleteBehavior.Cascade);

        modelBuilder.Entity<ChatUser>()
            .HasOne(chatUser => chatUser.User)
            .WithMany()
            .HasForeignKey(chatUser => chatUser.UserId)
            .OnDelete(DeleteBehavior.Cascade);

        modelBuilder.Entity<ChatUser>()
            .Navigation(chatUser => chatUser.User)
            .AutoInclude(false);

        modelBuilder.Entity<Message>()
            .HasOne(message => message.Chat)
            .WithMany(chat => chat.Messages)
            .HasForeignKey(message => message.ChatId)
            .OnDelete(DeleteBehavior.Cascade);

        modelBuilder.Entity<Message>()
            .HasOne(message => message.User)
            .WithMany()
            .HasForeignKey(message => message.UserId)
            .OnDelete(DeleteBehavior.Cascade);
    }
}
```

SeedData.cs:

```
using Microsoft.AspNetCore.Identity;
using StudyBridgeDiplomaApp.Models;

namespace StudyBridgeDiplomaApp.Data;

public class SeedData
{
    public static async Task SeedRoles(IServiceProvider serviceProvider)
    {
        var roleManager =
serviceProvider.GetRequiredService<RoleManager<IdentityRole<Guid>>>();

        string[] roles = ["Адміністратор", "Викладач", "Студент"];
        foreach (var role in roles)
        {
            if (!await roleManager.RoleExistsAsync(role))
            {
                var newRole = new IdentityRole<Guid>
                {
```

```
                    Id = Guid.NewGuid(),
                    Name = role,
                    NormalizedName = role.ToUpper()
                };
                await roleManager.CreateAsync(newRole);
            }
        }
    }

    public static async Task SeedAdmin(IServiceProvider serviceProvider)
    {
        var userManager = serviceProvider.GetRequiredService<UserManager<User>>();
        var roleManager =
serviceProvider.GetRequiredService<RoleManager<IdentityRole<Guid>>>();
        var configuration = serviceProvider.GetRequiredService<IConfiguration>();

        var adminEmail = configuration["AdminSettings:Email"];
        var adminPassword = configuration["AdminSettings:Password"];
        var adminRole = "Адміністратор";

        var admin = await userManager.FindByEmailAsync(adminEmail!);
        if (admin == null)
        {
            admin = new User
            {
                FirstName = "Адмін",
                LastName = "Системи",
                Email = adminEmail,
                NormalizedEmail = adminEmail!.ToUpper(),
                UserName = adminEmail,
                NormalizedUserName = adminEmail.ToUpper(),
                IsStudent = false,
                Group = null
            };

            var result = await userManager.CreateAsync(admin, adminPassword!);
            if (!result.Succeeded)
            {
                throw new Exception($"Помилка створення адміністратора: {string.Join(",
", result.Errors.Select(error => error.Description))}");
            }
        }

        if (!await userManager.IsInRoleAsync(admin, adminRole))
            await userManager.AddToRoleAsync(admin, adminRole);
    }
}
```

RequiredIfAttribute.cs:

```
using System.ComponentModel.DataAnnotations;

namespace StudyBridgeDiplomaApp.Attributes;

[AttributeUsage(AttributeTargets.Property)]
public class RequiredIfAttribute(string dependentProperty, object expectedValue) :
ValidationAttribute
{
    private readonly string _dependentProperty = dependentProperty;
    private readonly object _expectedValue = expectedValue;

    protected override ValidationResult? IsValid(object? value, ValidationContext
validationContext)
    {
        var dependentProperty =
validationContext.ObjectType.GetProperty(_dependentProperty);
```

```
        if (dependentProperty == null)
            return new ValidationResult($"Властивість {_dependentProperty} не знайдено");

        var dependentValue =
dependentProperty.GetValue(validationContext.ObjectInstance);
        if (dependentValue?.Equals(_expectedValue) == true &&
string.IsNullOrEmpty(value?.ToString()))
            return new ValidationResult(ErrorMessage);

        return ValidationResult.Success;
    }
}
```

Program.cs:

```
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using StudyBridgeDiplomaApp.Data;
using StudyBridgeDiplomaApp.Hubs;
using StudyBridgeDiplomaApp.Models;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllersWithViews();
builder.Services.AddDbContext<StudyBridgeDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
builder.Services.AddIdentity<User, IdentityRole<Guid>>()
    .AddEntityFrameworkStores<StudyBridgeDbContext>()
    .AddDefaultTokenProviders();
builder.Services.Configure<IdentityOptions>(options =>
{
    options.Password.RequireDigit = true;
    options.Password.RequiredLength = 8;
    options.Password.RequireNonAlphanumeric = true;
    options.Password.RequireUppercase = true;
    options.Password.RequireLowercase = true;
});
builder.Services.ConfigureApplicationCookie(options =>
{
    options.LoginPath = "/Account/Login";
    options.Cookie.Name = "StudyBridgeCookie";
});
builder.Services.AddAuthorization();
builder.Services.AddSignalR();

var app = builder.Build();
app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();
app.UseStaticFiles();
using (var scope = app.Services.CreateScope())
{
    var services = scope.ServiceProvider;
    await SeedData.SeedRoles(services);
    await SeedData.SeedAdmin(services);
}
app.MapHub<ChatHub>("/chatHub");

app.Run();
```

appsettings.json:

```
{
  "ConnectionStrings": {
```

```
    "DefaultConnection":
"Server=(localdb)\\mssqllocaldb;Database=StudyBridge;Trusted_Connection=True;MultipleActi
veResultSets=true"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AdminSettings": {
    "Email": "studybridgeapp@gmail.com",
    "Password": "Admin@123"
  },
  "AllowedHosts": "*"
}
```

_ViewStart.cshtml:

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

_ViewImports.cshtml:

```
@addTagHelper "*, Microsoft.AspNetCore.Mvc.TagHelpers"
@using StudyBridgeDiplomaApp.ViewModels
@using StudyBridgeDiplomaApp.Extensions
@using System.IO
```

_Layout.cshtml:

```
<!DOCTYPE html>

<html lang="uk">
<head>
    <meta name="viewport" content="width=device-width" />
    <title>StudyBridge</title>
    <link rel="icon" href="~/icons/study_bridge.svg" type="image/svg" />
    <link href="~/css/bootstrap.min.css" rel="stylesheet" />
    <link href="~/css/styles.css" rel="stylesheet" />
    <style>
        body {
            background: linear-gradient(rgba(48, 67, 55, 0.7), rgba(52, 104, 108, 0.7),
rgba(179, 178, 165, 0.7));
        }
    </style>
</head>
<body>
    <div>
        @RenderBody()
    </div>
    <script src="~/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

OpenChat.cshtml:

```
@model StudyBridgeDiplomaApp.Models.Chat

<div class="container min-height-page" id="chatData" data-chat-id="@Model.Id" data-user-
id="@User.GetUserId()">
    <button class="btn btn-success w-100 mt-3 mb-3 p-2" data-bs-toggle="modal" data-bs-
target="#chatParticipantsModal">
        <h2><b>@Model.Name</b></h2>
    </button>
```

```
    <div id="messagesList" class="p-3 mb-3">
        @foreach (var message in Model.Messages.OrderBy(message => message.SentAt))
        {
            <div class="message-container d-flex align-items-start mb-3">
                <img src="@(message.User!.ProfileImageUrl != null
&& System.IO.File.Exists(System.IO.Path.Combine(Directory.GetCurrentDirectory(),
"wwwroot/uploads", message.User.ProfileImageUrl))
? Url.Content($"~/uploads/{message.User.ProfileImageUrl}")
: Url.Content("~/icons/account_circle.png"))"
                     class="rounded-circle profile-image-chat">
                <div class="message-box">
                    <b class="text-primary-color">@message.User!.FirstName
@message.User.LastName</b>
                    <div class="message-content">
                        @message.Content
                        <small class="message-
time">@message.SentAt.ToLocalTime().ToString("HH:mm")</small>
                    </div>
                </div>
            </div>
        }
    </div>

    <div class="input-group">
        <textarea id="messageInput" class="form-control" rows="2" style="resize:
none;"></textarea>
        <button id="sendButton" class="btn btn-success">Надіслати</button>
    </div>
</div>

<div class="modal fade" id="chatParticipantsModal" tabindex="-1" aria-
labelledby="chatParticipantsLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title text-dark" id="chatParticipantsLabel">Учасники
чату</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Закрити"></button>
            </div>
            <div class="modal-body">
                <p class="text-dark">Кількість учасників:
<b>@Model.ChatUsers.Count()</b></p>
                <ul class="list-group">
                    @foreach (var participant in Model.ChatUsers)
                    {
                        <li class="list-group-item d-flex align-items-center text-
primary-color">
                            <img src="@(participant.User!.ProfileImageUrl != null
&& System.IO.File.Exists(System.IO.Path.Combine(Directory.GetCurrentDirectory(),
"wwwroot/uploads", participant.User.ProfileImageUrl))
? Url.Content($"~/uploads/{participant.User.ProfileImageUrl}")
: Url.Content("~/icons/account_circle.png"))"
                                 class="rounded-circle me-2 img-chatparticipants">
                            @participant.User.FirstName @participant.User.LastName
                        </li>
                    }
                </ul>
            </div>
        </div>
    </div>
</div>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
```

```
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></scri
pt>
<script src="https://cdnjs.cloudflare.com/ajax/libs/microsoft-
signalr/7.0.5/signalr.min.js"></script>
<script src="~/js/openChat.js"></script>
```

ListChats.cshtml:

```
@model List<StudyBridgeDiplomaApp.Models.Chat>

<div class="container min-height-page pt-5">
    <h1 class="text-center text-white"><b>Список чатів</b></h1>
    <div class="d-flex justify-content-between align-items-center w-85 mb-3">
        <div class="background-back">
            <a href="/dashboard" class="btn btn-primary">Головна</a>
        </div>
    </div>
    <div class="row mt-4">
        @foreach (var chat in Model)
        {
            <div class="col-md-6 mb-4">
                <div class="card shadow-lg text">
                    <div class="card-header text-white text-center">
                        <h4 class="mb-0"><b>@chat.Name</b></h4>
                    </div>
                    <div class="card-body">
                        <h6>Дата створення: @chat.CreatedAt</h6>
                        <h6>Учасники:</h6>
                        <ul class="list-group">
                            @foreach (var chatUser in chat.ChatUsers)
                            {
                                <li class="list-group-item text-white p-2">
                                    @chatUser.User!.FirstName @chatUser.User.LastName
                                    <span class="text-
white">(@chatUser.User.Email)</span>
                                </li>
                            }
                        </ul>
                    </div>
                </div>
            </div>
        }
    </div>
</div>
```

CreateChat.cshtml:

```
@if (TempData["ErrorEditNameMessage"] != null)
{
    <script>
        alert("Чат з такою групою вже існує");
    </script>
}

<div class="container min-height-page pt-5">
    <div class="card shadow-lg">
        <div class="card-header text-white text-center">
            <h1 class="mb-0"><b>Створити чат</b></h1>
            <div class="d-flex justify-content-start align-items-center">
                <a href="/dashboard" class="btn btn-primary">Головна</a>
            </div>
        </div>
        <div class="card-body">
            <form method="post" asp-action="CreateChat">
                <div class="mb-4">
```

```
                    <label class="form-label">Група:</label>
                    <input type="text" name="groupName" class="form-control rounded-pill
shadow-sm"
                           required
                           minlength="5"
                           maxlength="8"
                           pattern="^[A-ZA-Я]{1,2}\d{2}-\d[a-яіїє]{0,2}$"
                           title="Група введена невірно (невірний формат)" />
                </div>

                <div class="d-flex gap-3">
                    <button type="submit" class="btn btn-success w-
50"><b>Створити</b></button>
                    <a href="admindashboard" class="btn btn-secondary w-
50"><b>Скасувати</b></a>
                </div>
            </form>
        </div>
    </div>
</div>
```

ListAnnouncements.cshml:

```
@model IEnumerable<StudyBridgeDiplomaApp.Models.Announcement>

<div class="container min-height-page pt-5 mb-3">
    <div class="card shadow-lg">
        <div class="card-header text-center text-white">
            <h1><b>Список оголошень</b></h1>
            <div class="d-flex justify-content-between align-items-center mt-3">
                <a href="/dashboard" class="btn btn-primary">Головна</a>
                <a href="/createAnnouncement" class="btn btn-primary">Додати нове
оголошення</a>
            </div>
        </div>

        <div class="mt-3 mb-3 p-3 table-responsive">
            <table class="table mt-3">
                <thead>
                    <tr>
                        <th>Заголовок</th>
                        <th>Контекст</th>
                        <th>Дата</th>
                        <th></th>
                    </tr>
                </thead>
                <tbody>
                    @foreach (var announcement in Model)
                    {
                        <tr>
                            <td class="text-primary-color text-
center"><b>@announcement.Title</b></td>
                            <td class="text-align-justify">@announcement.Content</td>
                            <td class="text-
center">@announcement.CreatedAt.ToLocalTime().ToString("dd.MM.yyyy HH:mm")</td>
                            <td>
                                <form asp-action="DeleteAnnouncement" asp-route-
id="@announcement.Id" method="post" class="d-inline">
                                    <button type="submit" class="btn btn-
secondary"><b>Видалити</b></button>
                                </form>
                            </td>
                        </tr>
                    }
                </tbody>
            </table>
```

```
        </div>
    </div>
</div>
```

CreateAnnouncement.cshtml:

```
<div class="container pt-5 min-height-page">
    <div class="card shadow-lg">
        <div class="card-header text-center text-white">
            <h1><b>Створити оголошення</b></h1>
            <div class="d-flex justify-content-between align-items-center w-85 mb-3">
                <a href="/dashboard" class="btn btn-primary">Головна</a>
            </div>
        </div>
        <div class="card-body">
            <form asp-action="CreateAnnouncement" method="post">
                <div class="mb-4">
                    <label class="form-label">Заголовок оголошення:</label>
                    <input type="text" class="form-control rounded-pill shadow-sm"
name="title" required>
                </div>

                <div class="mb-4">
                    <label class="form-label">Контент оголошення:</label>
                    <textarea class="form-control" name="content" rows="5"
required></textarea>
                </div>

                <div class="d-flex gap-3">
                    <button type="submit" class="btn btn-success w-
50"><b>Створити</b></button>
                    <a href="/admindashboard" class="btn btn-secondary w-
50"><b>Скасувати</b></a>
                </div>
            </form>
        </div>
    </div>
</div>
```

StudentsByGroups.cshtml:

```
@model Dictionary<string, List<StudyBridgeDiplomaApp.ViewModels.StudentViewModel>>

<div class="container d-flex flex-column align-items-center flex-grow-1 min-height-page">
    <h1 class="p-2 text-white rounded-2 mt-5"><b>Список студентів по групах</b></h1>
    <div class="card w-85 mt-3 p-3">
        <div class="d-flex justify-content-between align-items-center w-85 mb-3">
            <a href="/dashboard" class="btn btn-primary">Головна</a>
        </div>
        <div class="accordion w-100" id="studentsAccordion">
            @foreach (var group in Model)
            {
                <div class="accordion-item">
                    <h2 class="accordion-header" id="heading-@group.Key">
                        <button class="accordion-button collapsed" type="button" data-bs-
toggle="collapse"
                        data-bs-target="#collapse-@group.Key" aria-expanded="false" aria-
controls="collapse-@group.Key">
                            Група: @group.Key
                        </button>
                    </h2>
                    <div id="collapse-@group.Key" class="accordion-collapse collapse"
aria-labelledby="heading-@group.Key"
                    data-bs-parent="#studentsAccordion">
                        <div class="accordion-body">
                            <ul class="list-group">
```

```
                                @foreach (var student in group.Value)
                                {
                                    <li class="list-group-item d-flex align-items-center
bg-white">
                                        <img src="@student.ProfileImageUrl" class="me-3
bg-primary img-list">
                                        <div class="text-primary-color">
                                            <strong>@student.LastName
@student.FirstName</strong>
                                        </div>
                                    </li>
                                }
                            </ul>
                        </div>
                    </div>
                }
            </div>
        </div>
    </div>
```

Profile.cshtml:

```
@model StudyBridgeDiplomaApp.Models.User

<div class="container-fluid vh-100 d-flex flex-column">
    <nav class="navbar navbar-expand-lg nav-dashboard">
        <div class="container-fluid">
            <div class="d-flex justify-content-end align-items-center">
                <a href="/dashboard" class="btn btn-primary m-3">Головна</a>
            </div>
            <div class="d-flex justify-content-end align-items-center">
                <form asp-action="Logout" method="post">
                    <button type="submit" class="btn btn-secondary m-3">Вийти</button>
                </form>
            </div>
        </div>
    </nav>

    <div class="container d-flex flex-column align-items-center justify-content-center
flex-grow-1 mt-4">
        <div class="text-center">
            <img id="profileImagePreview"
                 src="@(Model.ProfileImageUrl != null
&& System.IO.File.Exists(System.IO.Path.Combine(Directory.GetCurrentDirectory(),
"wwwroot/uploads", Model.ProfileImageUrl))
? Url.Content($"~/uploads/{Model.ProfileImageUrl}")
: Url.Content("~/icons/account_circle.png"))"
                 class="rounded-circle border border-3 img-profile" />
        </div>
        <div id="userNameContainer" class="text-center mt-3">
            <h2 class="text-primary-color"><b>@Model.FirstName @Model.LastName</b></h2>
            @if (User.IsInRole("Студент"))
            {
                <h5 class="text-primary-color">Студент</h5>
            }
            @if (User.IsInRole("Викладач"))
            {
                <h5 class="text-primary-color">Викладач</h5>
            }
            @if (User.IsInRole("Адміністратор"))
            {
                <h5 class="text-primary-color">Адміністратор</h5>
            }
            <br />
            <h6 class="text-muted">Електронна пошта: @Model.Email</h6>
```

```
            @if (User.IsInRole("Студент"))
            {
                <h6 class="text-muted">Група: @Model.Group</h6>
            }
            @if (User.IsInRole("Викладач"))
            {
                <h6 class="text-muted">Посада: @(string.IsNullOrEmpty(Model.Position) ?
"-" : Model.Position)</h6>
                <h6 class="text-muted">Вчене звання:
@(string.IsNullOrEmpty(Model.AcademicTitle) ? "-" : Model.AcademicTitle)</h6>
            }
            <button class="btn btn-success mt-3" id="editButton">Редагувати дані</button>
        </div>

        <div id="editForm" class="d-none">
            <form asp-action="EditProfile" method="post" enctype="multipart/form-data">
                <div class="mb-3">
                    <label class="form-label mt-3">Завантажити нове фото профілю:</label>
                    <input type="file" class="form-control" accept="image/*"
id="profileImage" name="ProfileImage" />
                </div>
                <div class="mb-3">
                    <label class="form-label">Ім'я:</label>
                    <input type="text" class="form-control" value="@Model.FirstName"
name="firstName"
                           minlength="2"
                           maxlength="50"
                           pattern="^[A-ZA-ЯІЇЄҐ][a-za-яіїєґ'-]*$"
                           title="Ім'я може містити лише літери, дефіс або апостроф, і
повинно починатися з великої літери" />
                </div>
                <div class="mb-3">
                    <label class="form-label">Прізвище:</label>
                    <input type="text" class="form-control" value="@Model.LastName"
name="lastName"
                           minlength="2"
                           maxlength="100"
                           pattern="^[A-ZA-ЯІЇЄҐ][a-za-яіїєґ'-]*$"
                           title="Прізвище може містити лише літери, дефіс або апостроф,
і повинно починатися з великої літери" />
                </div>
                @if (User.IsInRole("Викладач"))
                {
                    <div class="mb-3">
                        <label class="form-label">Посада:</label>
                        <input type="text" class="form-control" value="@Model.Position"
name="position" />
                    </div>
                    <div class="mb-3">
                        <label class="form-label">Вчене звання:</label>
                        <input type="text" class="form-control"
value="@Model.AcademicTitle" name="academicTitle" />
                    </div>
                }
                <div class="d-flex justify-content-between">
                    <button type="submit" class="btn btn-success w-50 me-2 text-
medium">Зберегти зміни</button>
                    <button type="button" class="btn btn-secondary w-50 ms-2 text-medium"
id="cancelEdit">Скасувати</button>
                </div>
            </form>
        </div>
    </div>
</div>
```

```
<script src="~/js/profile.js"></script>
```

MyChats.cshtml:

```
@model IEnumerable<StudyBridgeDiplomaApp.Models.Chat>

<div class="container-fluid pt-5 d-flex flex-column w-85 min-height-page">
    <h1 class="mb-4 text-white text-center p-2 rounded-4"><b>Ваші чати</b></h1>
    <div class="d-flex justify-content-between align-items-center w-85 mb-3">
        <div class="background-back m-1">
            <a href="/dashboard" class="btn btn-primary">Головна</a>
        </div>
    </div>
    @if (!Model.Any())
    {
        <h3 class="text-white mt-3">У вас поки немає чатів</h3>
    }
    else
    {
        <ul class="list-group mt-3">
            @foreach (var chat in Model)
            {
                <li class="list-group-item d-flex justify-content-between align-items-
center m-2">
                    <a href="@Url.Action("OpenChat", "Chat", new { chatId = chat.Id })"
class="text-decoration-none text-primary">
                        <b>@chat.Name</b>
                    </a>
                    <span class="badge bg-main">Учасники: @chat.ChatUsers.Count</span>
                </li>
            }
        </ul>
    }
</div>
```

ListUsers.cshtml:

```
@model IEnumerable<StudyBridgeDiplomaApp.ViewModels.UserViewModel>

<div class="container d-flex flex-column align-items-center flex-grow-1 min-height-page">
    <h1 class="p-2 text-white rounded-2 mt-5"><b>Список користувачів</b></h1>
    <div class="card w-85 mt-3 mb-3 p-3 table-responsive">
        <div class="d-flex justify-content-between align-items-center w-85 mb-3">
            <a href="/dashboard" class="btn btn-primary">Головна</a>
        </div>
        <table class="table">
            <thead>
                <tr>
                    <th>Фото</th>
                    <th>Ім'я</th>
                    <th>Прізвище</th>
                    <th>Електронна пошта</th>
                    <th>Роль</th>
                    <th>Група</th>
                    <th>Посада</th>
                    <th>Вчене звання</th>
                </tr>
            </thead>
            <tbody>
                @foreach (var user in Model)
                {
                    <tr>
                        <td>
                            <img src="@user.ProfileImageUrl" class="img-list" />
                        </td>
                        <td class="text-primary-color">@user.FirstName</td>
```

```
                        <td class="text-primary-color">@user.LastName</td>
                        <td>@user.Email</td>
                        <td>@user.Role</td>
                        <td>@user.StudentGroup</td>
                        <td>@user.TeacherPosition</td>
                        <td>@user.TeacherAcademicTitle</td>
                    </tr>
                }
            </tbody>
        </table>
    </div>
</div>
```

ListTeachers.cshtml:

```
@model IEnumerable<StudyBridgeDiplomaApp.ViewModels.TeacherViewModel>

<div class="container d-flex flex-column align-items-center flex-grow-1 min-height-page">
    <h1 class="p-2 text-white rounded-2 mt-5"><b>Список викладачів</b></h1>
    <div class="card w-85 mt-3 p-3 table-responsive mb-3">
        <div class="d-flex justify-content-between align-items-center w-85 mb-3">
            <a href="/dashboard" class="btn btn-primary">Головна</a>
        </div>
        <table class="table">
            <thead>
                <tr>
                    <th>Фото</th>
                    <th>Ім'я</th>
                    <th>Прізвище</th>
                    <th>Посада</th>
                    <th>Вчене звання</th>
                </tr>
            </thead>
            <tbody>
                @foreach (var teacher in Model)
                {
                    <tr>
                        <td>
                            <img src="@teacher.ProfileImageUrl" class="img-list" />
                        </td>
                        <td class="text-primary-color">@teacher.FirstName</td>
                        <td class="text-primary-color">@teacher.LastName</td>
                        <td>@teacher.Position</td>
                        <td>@teacher.AcademicTitle</td>
                    </tr>
                }
            </tbody>
        </table>
    </div>
</div>
```

Dashboard.cshtml:

```
<div class="container-fluid container-fluid-all d-flex flex-column min-height-page">

    <nav class="navbar navbar-expand-lg nav-dashboard">
        <div class="container-fluid align-content-center">
            <div class="d-flex justify-content-center mt-3">
                @if (User.IsInRole("Студент"))
                {
                    <a href="/myChats" class="btn btn-primary mx-2 m-2">Чати</a>
                }
                @if (User.IsInRole("Викладач"))
                {
                    <a href="/listStudentsByGroups" class="btn btn-primary m-2">Список
студентів</a>
```

```
                }
                @if (User.IsInRole("Студент") || User.IsInRole("Викладач"))
                {
                    <a href="/listTeachers" class="btn btn-primary m-2">Список
викладачів</a>
                }
                @if (User.IsInRole("Адміністратор"))
                {
                    <a href="/adminDashboard" class="btn btn-primary mx-2 m-2">Адмін-
панель</a>
                }
            </div>
            <div class="d-flex justify-content-end align-items-center mt-2">
                <a href="/profile" class="d-flex align-items-center text-decoration-none
text-dark w-auto">
                    <img src="@(Model.ProfileImageUrl != null
&& System.IO.File.Exists(System.IO.Path.Combine(Directory.GetCurrentDirectory(),
"wwwroot/uploads", Model.ProfileImageUrl))
? Url.Content($"~/uploads/{Model.ProfileImageUrl}")
: Url.Content("~/icons/account_circle.png"))"
                         class="rounded-circle img-account"/>
                </a>
            </div>
        </div>
    </nav>

    <div class="container d-flex flex-column align-items-center justify-content-center
flex-grow-1">
        <br />
        <br />
        <h1 class="mb-3 text-white p-2 rounded-4"><b>Оголошення</b></h1>
        @foreach (var announcement in ViewBag.Announcements)
        {
            <div class="card w-75 mt-3">
                <div class="card-body">
                    <h5 class="card-title"><b>@announcement.Title</b></h5>
                    <p class="card-text text-align-justify pt-
3">@announcement.Content</p>
                    <p class="text-dark"><b>Додано:
@announcement.CreatedAt.ToLocalTime().ToString("dd.MM.yyyy HH:mm")</b></p>
                </div>
            </div>
        }
    </div>
</div>
```

Authenticate.cshtml:

```
@model AuthenticateViewModel

<div class="container d-flex justify-content-center align-items-center min-height-page">
    <div class="card p-3 w-authenticate" id="forms-container" data-active-
form="@ViewBag.ActiveForm">

        <div class="d-flex justify-content-between mb-3 gap-2">
            <button id="registerBtn" class="active-button btn btn-primary w-50"
onclick="showRegisterForm()">Зареєструватися</button>
            <button id="loginBtn" class="btn btn-primary w-50"
onclick="showLoginForm()">Ввійти</button>
        </div>

        <form asp-action="Register" id="registerForm" method="post" class="form-login
@(ViewBag.ActiveForm == "register" ? "active" : "")">
            <h2 class="text-center"><b>Реєстрація</b></h2>
            <div class="mb-2 text-start">
```

```
                <label asp-for="RegisterViewModel!.FirstName" class="form-
label">Ім'я:</label>
                <input type="text" asp-for="RegisterViewModel!.FirstName" class="form-
control"
                       required
                       minlength="2"
                       maxlength="50"
                       pattern="^[A-ZA-ЯІЇЄҐ][a-za-яіїєґ'-]*$"
                       title="Ім'я може містити лише літери, дефіс або апостроф, і
повинно починатися з великої літери" />
                <span class="text-danger">@Html.ValidationMessageFor(model =>
model.RegisterViewModel!.FirstName)</span>
            </div>
            <div class="mb-2 text-start">
                <label asp-for="RegisterViewModel!.LastName" class="form-
label">Прізвище:</label>
                <input type="text" asp-for="RegisterViewModel!.LastName" class="form-
control"
                       required
                       minlength="2"
                       maxlength="100"
                       pattern="^[A-ZA-ЯІЇЄҐ][a-za-яіїєґ'-]*$"
                       title="Прізвище може містити лише літери, дефіс або апостроф, і
повинно починатися з великої літери" />
                <span class="text-danger">@Html.ValidationMessageFor(model =>
model.RegisterViewModel!.LastName)</span>
            </div>
            <div class="mb-2 form-check text-start">
                <label asp-for="RegisterViewModel!.IsStudent" class="form-check-
label">Студент</label>
                <input type="checkbox" class="form-check-input" asp-
for="RegisterViewModel!.IsStudent" id="isStudent" onclick="toggleGroupField()" />
            </div>
            <div class="mb-2 text-start" id="groupField">
                <label asp-for="RegisterViewModel!.Group" class="form-
label">Група:</label>
                <input type="text" asp-for="RegisterViewModel!.Group" class="form-
control"
                       minlength="5"
                       maxlength="8"
                       pattern="^[A-ZA-Я]{1,2}\d{2}-\d[a-яіїє]{0,2}$"
                       title="Група введена невірно (невірний формат)" />
                <span class="text-danger">@Html.ValidationMessageFor(model =>
model.RegisterViewModel!.Group)</span>
            </div>
            <div class="mb-2 text-start">
                <label asp-for="RegisterViewModel!.Email" class="form-label">Електронна
пошта:</label>
                <input type="email" asp-for="RegisterViewModel!.Email" class="form-
control"
                       required
                       minlength="6"
                       maxlength="320" />
                <span class="text-danger">@Html.ValidationMessageFor(model =>
model.RegisterViewModel!.Email)</span>
            </div>
            <div class="mb-2 text-start">
                <label asp-for="RegisterViewModel!.Password" class="form-
label">Пароль:</label>
                <input type="password" asp-for="RegisterViewModel!.Password" class="form-
control"
                       required
                       minlength="8"
                       maxlength="128"
                       pattern="^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[!@@#$%^&*]).{8,}$"
```

```
                    title="Пароль повинен містити щонайменше одну велику літеру, одну
малу літеру, одну цифру та один спеціальний символ" />
                <span class="text-danger">@Html.ValidationMessageFor(model =>
model.RegisterViewModel!.Password)</span>
            </div>
            <div class="mb-2 text-start">
                <label asp-for="RegisterViewModel!.ConfirmPassword" class="form-
label">Підтвердіть пароль:</label>
                <input type="password" asp-for="RegisterViewModel!.ConfirmPassword"
class="form-control" required />
                <span class="text-danger">@Html.ValidationMessageFor(model =>
model.RegisterViewModel!.ConfirmPassword)</span>
            </div>
            <button type="submit" class="btn btn-success w-100 mt-
3">Зареєструватися</button>
        </form>

        <form asp-action="Login" id="loginForm" method="post" class="form-register
@(ViewBag.ActiveForm == "login" ? "active" : "")">
            <h2 class="text-center"><b>Вхід</b></h2>
            <div class="mb-2 text-start">
                <label asp-for="LoginViewModel!.Email" class="form-label">Електронна
пошта:</label>
                <input type="email" asp-for="LoginViewModel!.Email" class="form-control"
                       required
                       minlength="6"
                       maxlength="320" />
                <span class="text-danger">@Html.ValidationMessageFor(model =>
model.LoginViewModel!.Email)</span>
            </div>
            <div class="mb-2 text-start">
                <label asp-for="LoginViewModel!.Password" class="form-
label">Пароль:</label>
                <input type="password" asp-for="LoginViewModel!.Password" class="form-
control" required />
                <span class="text-danger">@Html.ValidationMessageFor(model =>
model.LoginViewModel!.Password)</span>
            </div>
            <button type="submit" class="btn btn-success w-100 mt-3">Ввійти</button>
        </form>
    </div>
</div>

      <script src="~/js/authenticate.js"></script>
```

AdminDashboard.cshtml:

```
<div class="container pt-5 min-height-page">
    <div class="card shadow-lg">
        <div class="card-header text-white text-center">
            <h1 class="mb-0"><b>Адмін-панель</b></h1>
            <div class="d-flex justify-content-start align-items-center">
                <a href="/dashboard" class="btn btn-primary">Головна</a>
            </div>
        </div>
        <div class="card-body d-flex flex-column gap-3">
            <a asp-controller="Chat" asp-action="CreateChat" class="btn btn-success
rounded-pill p-3 text-start shadow-sm">
                <b>Створити чат</b>
            </a>
            <a asp-controller="Chat" asp-action="ListChats" class="btn btn-success
rounded-pill p-3 text-start shadow-sm">
                <b>Переглянути чати</b>
            </a>
```

```
            <a asp-controller="Announcements" asp-action="CreateAnnouncement" class="btn
btn-success rounded-pill p-3 text-start shadow-sm">
                <b>Створити оголошення</b>
            </a>
            <a asp-controller="Announcements" asp-action="ListAnnouncements" class="btn
btn-success rounded-pill p-3 text-start shadow-sm">
                <b>Переглянути оголошення</b>
            </a>
            <a asp-controller="Account" asp-action="ListUsers" class="btn btn-success
rounded-pill p-3 text-start shadow-sm">
                <b>Переглянути користувачів</b>
            </a>
        </div>
    </div>
</div>
```

profile.js:

```
const userNameContainer = document.getElementById('userNameContainer');
const editForm = document.getElementById('editForm');
const editButton = document.getElementById('editButton');
const cancelEdit = document.getElementById('cancelEdit');
const profileImagePreview = document.getElementById('profileImagePreview');
const initialImageSrc = profileImagePreview.src;

editButton.addEventListener('click', function () {
    editForm.classList.remove('d-none');
    editButton.classList.add('d-none');
    userNameContainer.classList.add('d-none');
});

cancelEdit.addEventListener('click', function () {
    editForm.classList.add('d-none');
    editButton.classList.remove('d-none');
    userNameContainer.classList.remove('d-none');

    profileImagePreview.src = initialImageSrc;

    document.getElementById('profileImage').value = '';
});

document.getElementById('profileImage').addEventListener('change', function (event) {
    const file = event.target.files[0];
    if (file) {
        const reader = new FileReader();
        reader.onload = function (e) {
            profileImagePreview.src = e.target.result;
        };
        reader.readAsDataURL(file);
    }
});
```

openChat.js:

```
document.addEventListener("DOMContentLoaded", function () {
    const chatDataElement = document.getElementById("chatData");
    const chatId = chatDataElement.dataset.chatId;
    const userId = chatDataElement.dataset.userId;

    const messagesList = document.getElementById("messagesList");
    const sendButton = document.getElementById("sendButton");
    const messageInput = document.getElementById("messageInput");

    function scrollToBottom() {
        if (messagesList) {
            messagesList.scrollTop = messagesList.scrollHeight;
```

```
        }
    }

    const connection = new signalR.HubConnectionBuilder()
        .withUrl(`/chatHub?chatId=${chatId}`)
        .build();

    connection.on("ReceiveMessage", function (user, message, timestamp, profileImageUrl)
{
        const messageContainer = document.createElement("div");
        messageContainer.classList.add("message-container", "d-flex", "align-items-
start", "mb-3");

        const profileImage = document.createElement("img");
        profileImage.src = profileImageUrl || "/icons/account_circle.png";
        profileImage.alt = "Фото профілю";
        profileImage.classList.add("rounded-circle", "profile-image-chat");

        const messageBox = document.createElement("div");
        messageBox.classList.add("message-box");

        const userElement = document.createElement("b");
        userElement.classList.add("text-primary-color");
        userElement.textContent = user;

        const messageContent = document.createElement("div");
        messageContent.classList.add("message-content");
        messageContent.innerHTML = message.replace(/\n/g, "<br>");

        const timeElement = document.createElement("small");
        timeElement.classList.add("message-time");
        timeElement.textContent = timestamp;

        messageContent.appendChild(timeElement);
        messageBox.appendChild(userElement);
        messageBox.appendChild(messageContent);

        messageContainer.appendChild(profileImage);
        messageContainer.appendChild(messageBox);

        if (messagesList) {
            messagesList.appendChild(messageContainer);
        }

        scrollToBottom();
    });

    connection.start().catch(function (err) {
        console.error(err.toString());
    });

    if (sendButton) {
        sendButton.addEventListener("click", function (event) {
            const messageContent = messageInput.value;
            if (!messageContent.trim()) {
                return;
            }
            connection.invoke("SendMessage", messageContent, chatId,
userId).catch(function (err) {
                console.error(err.toString());
            });
            messageInput.value = "";
            event.preventDefault();
            scrollToBottom();
        });
```

```
    }

    if (messageInput) {
        messageInput.addEventListener("keydown", function (event) {
            if (event.key === "Enter" && !event.shiftKey) {
                event.preventDefault();
                sendButton.click();
            }
        });
    }

    scrollToBottom();
});
```

authenticate.js:

```
document.addEventListener("DOMContentLoaded", function () {
    const activeForm = document.getElementById('forms-container').getAttribute('data-
active-form').trim();
    if (activeForm == "register") {
        showRegisterForm();
    } else if (activeForm == "login") {
        showLoginForm();
    } else {
        showRegisterForm();
    }

    toggleGroupField();
});

function showRegisterForm() {
    document.getElementById("loginForm").style.display = "none";
    document.getElementById("registerForm").style.display = "block";

    document.getElementById("loginBtn").classList.remove("active-button");
    document.getElementById("registerBtn").classList.add("active-button");
}

function showLoginForm() {
    document.getElementById("loginForm").style.display = "block";
    document.getElementById("registerForm").style.display = "none";

    document.getElementById("loginBtn").classList.add("active-button");
    document.getElementById("registerBtn").classList.remove("active-button");
}

function toggleGroupField() {
    var checkbox = document.getElementById("isStudent");
    var groupField = document.getElementById("groupField");
    if (checkbox.checked) {
        groupField.style.display = "block";
        checkbox.value = true;
        groupInput.setAttribute("required", "true");
    } else {
        groupField.style.display = "none";
        checkbox.value = false;
        groupInput.setAttribute("required", "false");
    }
}
```

styles.css:

```
.min-height-page {
    min-height: 100vh !important;
}
```

```
.w-authenticate {
    max-width: 75vh;
    width: 100%
}

.nav-dashboard {
    background-color: rgba(34, 39, 42, 0.5);
}

form .form-label {
    margin-bottom: 2px;
    font-size: larger;
}

form .form-control {
    border-color: #FFFFFF;
    border-width: 2px;
    border-radius: 2vh;
    background-color: transparent;
}

.card {
    border-radius: 2vh;
    background: rgba(34, 39, 42, 0.5);
    color: white;
}

.btn {
    border-radius: 5vh;
}

.btn:hover {
    background: #C5D3E3;
}

.btn-primary {
    background-color: rgba(52, 104, 108, 0.3);
    border: none;
    font-weight: bold;
    font-size: larger;
}

.btn-success {
    background-color: #304337 !important;
    border: none !important;
    font-weight: bold !important;
    font-size: larger !important;
}

.btn-secondary {
    background-color: #B3B2A5 !important;
    border: none !important;
    font-weight: bold !important;
    font-size: larger !important;
}

.active-button {
    background-color: rgb(52, 104, 108);
}

.form-login, .form-register {
    display: none;
}
```

```
.form-login.active, .form-register.active {
    display: block;
}

.container-fluid {
    padding: 0;
}

.img-account {
    height: 10vh;
    width: 10vh;
    margin-right: 2vh;
}

.img-list {
    background-color: rgba(52, 104, 108, 0.3);
    width: 7vh;
    height: 7vh;
    border-radius: 5vh;
}

.container-fluid-all {
    margin-bottom: 2%;
}

.d-none {
    display: none;
}

.list-group-item {
    background: transparent;
    background-color: rgba(34, 39, 42, 0.5);
    border: none;
}

.bg-main {
    background: #304337;
}

#messagesList {
    height: 75vh;
    overflow-y: auto;
    background: rgba(34, 39, 42, 0.5);
    border-radius: 2vh;
}

.text-primary {
    color: #304337 !important;
    background-color: white;
    border-radius: 2vh;
    padding: 1vh;
}

.text-primary-color {
    color: rgb(52, 104, 108) !important;
}

.message-box {
    display: flex;
    flex-direction: column;
    align-items: flex-start;
    background-color: #f8f9fa;
    border-radius: 2vh;
    padding: 10px 15px;
    max-width: 70%;
```

```
    width: fit-content;
    word-wrap: break-word;
    margin-bottom: 10px;
}

.message-content {
    position: relative;
    word-break: break-word;
}

.message-time {
    bottom: 0;
    right: 0;
    font-size: 0.8em;
    color: grey;
    margin-left: 5vh;
}

.text-align-justify {
    text-align: justify;
}

.text-dark {
    color: #304337;
}

.profile-image-chat {
    width: 7vh;
    height: 7vh;
    margin-right: 1vh;
}

.w-85 {
    width: 85%;
}

.table {
    border-radius: 15px !important;
    overflow: hidden !important;
}

.table thead tr th {
    background-color: rgba(52, 104, 108, 0.3) !important;
    color: white;
    font-weight: bold;
}

.accordion-item {
    border: none;
    background: transparent;
}

.accordion-button:focus {
    box-shadow: none;
}

.accordion-button.collapsed {
    background-color: rgba(52, 104, 108, 0.3);
    color: white;
    font-weight: bold;
}

.accordion-button:not(.collapsed) {
    background-color: rgba(52, 104, 108, 0.3);
    color: white;
```

```
    font-weight: bold;
}

.bg-primary {
    background-color: rgb(52, 104, 108) !important;
}

.accordion-button::after {
    filter: brightness(0) invert(1);
}

.img-profile {
    width: 25vh;
    height: 25vh;
}

.text-medium {
    font-size: medium !important;
}

.background-back {
    background: rgba(34, 39, 42, 0.5);
    border-radius: 5vh;
}

.img-chatparticipants {
    background: rgba(34, 39, 42, 0.5);
    width: 40px;
    height: 40px;
    object-fit: cover;
}
```