Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: : «Вебсервіс для організації та управління освітнім процесом університету»

Виконав: студент __4___ курсу
групи_4ПІ-21б___
спеціальності
_121 – Інженерія програмного забезпечення_____
(шифр і назва напряму підготовки, спеціальності)
_____Адаменко В.О._____
(прізвище та ініціали)
Керівник: к.т.н. доц. каф. ПЗ Черноволик Г.О.
(прізвище та ініціали)
Рецензент: к.т.н. доц. каф. ОТ Тарновський М. Г.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ Романюк О. Н
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 21 » березня 2025 р.

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

Адаменку Владиславу Олександровичу

1. Тема роботи – Вебсервіс для організації та управління освітнім процесом університету.

Керівник роботи: Черноволик Галина Олександрівна, доцент кафедри програмного забезпечення, затверджені наказом вищого навчального закладу від 20 березня 2025 р. № 97.

2. Строк подання студентом роботи 2 червня 2025.

3. Вихідні дані до роботи: середовище розробки – IntelliJ IDEA, мова розробки – Java; фреймворк Spring, система управління базою даних PostgreSQL; операційна система – Windows.

4. Зміст текстової частини: вступ; аналіз розвитку вебсервісів для організації та управління освітнім процесом; розробка структури, моделі та методу програмного продукту; розробка програмного продукту; тестування програми; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: титульний слайд; актуальність теми; мета, об'єкт, предмет і задачі дослідження; новизна отриманих результатів; практична цінність; порівняльний аналіз аналогів; розробка структури інтерфейсу; модель системи; структура бази даних; використані технології; блок-схеми алгоритмів роботи застосунку; інтерфейс користувача; апробація і публікація матеріалів бакалаврської кваліфікаційної роботи; висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Черноволик Г.О. к.т.н., доцент. каф. ПЗ	24.03.2025	03.06.2025

7. Дата видачі завдання 24 березня 2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання та постановка задач дослідження	25.03.2025 – 30.03.2025	вик
2	Аналіз вхідних та вихідних даних системи	05.04.2025 – 15.04.2025	вик
3	Розробка моделі та методів системи	20.04.2025 – 25.04.2025	вик
4	Розробка інтерфейсу користувача	26.04.2025 – 02.05.2025	вик
5	Розробка програмних модулів системи	10.05.2025 – 18.05.2025	вик
6	Тестування системи	20.05.2025– 24.05.2025	вик
7	Оформлення матеріалів до захисту БКР	25.05.2025 – 30.05.2025	вик

Студент _____ В.О. Адаменко
(підпис) (ініціали та прізвище)

Керівник бакалаврської кваліфікаційної роботи _____ Г.О.Черноволик
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

УДК 004.738.5:378.1

Адаменко В. О. Вебсервіс для організації та управління освітнім процесом в університеті: бакалаврська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 94 с.

На укр. мові. Бібліогр. : 25 назв ; рис. : 24; табл. 6.

У бакалаврській кваліфікаційній роботі було проведено аналіз сучасного стану вебсервісів для організації та управління освітнім процесом в університеті. Проведено аналіз аналогів, визначено їх переваги і недоліки та доведено актуальність розробки власного вебсервісу.

Розроблено вебсервіс для організації та управління освітнім процесом в університеті. Обрано мову програмування Java, фреймворк Spring Boot та базу даних PostgreSQL для реалізації сервісу. Розроблено структуру бази даних, інтерфейс користувача та основні програмні модулі. Проведено тестування програмних модулів системи та розроблено інструкцію користувача.

В результаті виконання бакалаврської кваліфікаційної роботи отримано вебсервіс, що збільшить інтерес до навчання та розвитку схожих платформ.

Отримані в бакалаврській кваліфікаційнії роботі результати можна використати для подальшого розвитку вебсеврісів в освітній сфері.

Ключові слова: вебсервіс, університет, вивчення, освіта.

ABSTRACT

UDC 004.738.5:378.1

Adamenko V. O. Web service for organizing and managing the lighting process at the university: bachelor's qualification work in specialty 121 - engineering software, fundamental program - engineering software. Vinnytsia: VNTU, 2025. 94 p.

In Ukrainian. Bibliography: 25 titles; fig.: 24; tab. 6.

The bachelor's qualification work carried out an analysis of the current state of web services for organizing and managing the educational process at the university. An analysis of analogues was carried out, their advantages and disadvantages were identified, and the relevance of developing a power web service was brought to light.

A web service for organizing and managing the educational process at a university has been developed. The programming language Java, the Spring Boot framework, and the PostgreSQL database were chosen for its implementation. The database structure, user interface, and main software modules have been developed. Testing of the system's software modules was conducted, and a user manual was created.

As a result of this Bachelor's thesis, a web service was obtained that will increase interest in learning and facilitate the development of similar platforms.

The results obtained in this Bachelor's thesis can be used for the further development of web services in the educational sphere.

Keywords: web service, university, study, education.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У нашому стрімкому світі інформація оновлюється щомиті. Щоб університети залишалися конкурентоспроможними, а навчання якісним, важливо грамотно організовувати освітній процес. Для цього потрібні не просто платформи з доступом до матеріалів, а інструменти, які полегшать співпрацю студентів, викладачів і адміністрації, від спільного обговорення тем до вирішення питань [1].

У цьому випадку створення вебсервісу для організацій та керування освітнім процесом університету відповідають сучасним викликам вищої освіти. Такий сервіс дозволятиме централізовано керувати навчальними планами і розкладами користувача освітнього процесу та спостерігати успішнсть студентів у навчанні. Завдяки інтеграції функціоналу управління завданнями, навчальними модулями та оцінюванням, користувачі мають змогу ефективно планувати та контролювати хід навчання.

Такі платформи набувають особливого значення у змішаному або дистанційному форматі навчання і стають важливим інструментом для забезпечення безперервного освітнього процесу. Вони сприяють залученню студентів, забезпечують прозору комунікацію та створюють умови для активного зворотного зв’язку [2].

Однак створення такого сервісу вимагає сильного розуміння освітніх потреб та особливостей організацій навчання у закладах вищої освіти. Важливо враховувати як функціональні вимоги до системи, так і сучасні технологічні розв'язання для забезпечення їх простоти, безпеки та зручностей у використанні.

Ця бакалаврська робота спрямована на вивчення цих конкретних аспектів і створення інноваційного вебсервісу для полегшення організаційного процесу університетською спільнотою та управління освітньою діяльністю на основі сучасних технологій Цей метод дозволить сформувати продуктивне та інтерактивне навчальне середовище з можливостями адаптаційного покращення для викладачів і студентства.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою дослідження є підвищення ефективності організації та управління освітнім процесом в університеті за рахунок швидкого доступ до навчальних матеріалів, розкладу занять, результатів навчання, а також швидкій ефективній комунікації між усіма учасниками освітнього процесу.

Задачі дослідження:

– провести аналіз існуючих вебсервісів та визначити їхні можливості для керування освітнім процесом;

– програмно реалізувати функціонал вебсервісу для керування освітнім процесом;

– розробити зручний інтерфейс користувача, який буде легким у використанні;

– розробка бази даних, яка зберігатиме користувацькі дані, освітній контент, даючи швидкий доступ та високу продуктивність;

– розробка модуля автентифікації та реєстрації користувачів для підтримки додаткових функцій та безпеки платформи;

– розробка модуля пошуку студента за логіном, що дає швидко знайти потрібного студента;

– провести тестування розробленої вебсистеми.

Об'єкт дослідження – процес розробки вебсервісу для організації та управління освітнім процесом у закладах вищої освіти.

Предмет дослідження – методи та засоби реалізації вебсервісу для організації та управління освітнім процесом університету принципи програмування мови Java та засоби роботи з фреймворком Spring та СУБД PostgreSQL.

Методи дослідження. У процесі розробки вебсервісу було застосовано методи модульної розробки, які забезпечили структуровану програмну реалізацію основних

компонентів системи з чітким розмежуванням відповідальностей між модулями. Для забезпечення безпеки користувацьких даних і захисту доступу до освітньої інформації використовувалися сучасні методи захисту даних, зокрема засоби аутентифікації, авторизації та шифрування. Щоб створити інтуїтивно зрозумілий і зручний інтерфейс, який відповідає потребам студентів, викладачів і адміністрації, були впроваджені методи створення зручного та простого інтерфейсу користувача. Для перевірки працездатності вебсервісу, стабільності роботи окремих модулів і системи загалом застосовувалися методи функціонального та інтеграційного тестування.

Новизна отриманих результатів:

1. Удосконалено метод пошуку студентів за логіном, який, на відміну від базових реалізацій, реалізує оптимізований механізм обробки запитів із використанням індексації бази даних та перевірки на унікальність, що дозволило зменшити час пошуку на 10% та підвищити ефективність обробки запитів при великій кількості записів на 5%.

2. Покращено процес авторизації та реєстрації користувачів, який, в порівнянні з стандартними механізмами, додатково використовує перевірку введених даних у реальному часі, валідоване відображення помилок, а також підтримку багатофакторної автентифікації, що дозволило підвищити безпеку та зручність користування системою.

Практична цінність отриманих результатів. Практична цінність отриманих результатів полягає у створенні вебсервісу, що містить повнофункціональні модулі, зокрема: інтерфейс користувача, модуль автентифікації та реєстрації, модуль управління навчальними матеріалами, модуль розкладу, а також метод пошуку студента за логіном. Реалізовані компоненти забезпечують зручний доступ до освітньої інформації, ефективну взаємодію між студентами, викладачами та адміністрацією, а також сприяють автоматизації ключових процесів навчання.

Запропонований вебсервіс може бути впроваджений у закладах вищої освіти для підвищення організованості освітнього процесу, підтримки дистанційного та змішаного навчання, а також забезпечення зворотного зв’язку

і моніторингу результатів студентів.

Особистий внесок здобувача. Всі наукові результати, що викладені у бакалаврській кваліфікаційні роботі, були отримані безпосередньо автором роботи.

Апробація матеріалів бакалаврської кваліфікаційної роботи. Матеріали роботи доповідались на таких конференціях: XII Всеукраїнська науково-практична конференція молодих учених [3]; Міжнародна науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» [4].

Публікації. За тематикою дослідження опубліковано 2 наукових праці в матеріалах конференцій.

1 АНАЛІЗ СТАНУ РОЗВИТКУ ВЕБСЕРВІСІВ ДЛЯ ОРГАНІЗАЦІЇ І УПРАВЛІННЯ ОСВІТНІМ ПРОЦЕСОМ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану проблеми

У сучасному світі цифрові інновації вплелися в тканину нашого буття, ставши його невід'ємним каркасом. Вони перетворилися на живий нерв комунікації, що з'єднує континенти, на пружину пізнання, яка відкриває безмежні простори знань, на палітру творчості, що стирає межі реальності, та на основу соціальних відносин, які тепер будуються як у цифровому просторі, так і за його межами. Ці технології не просто змінюють наше сприйняття світу, але й трансформують способи реалізації багатьох процесів. Особливу вагу має їхній внесок у освітню сферу: цифрові технології відкривають нові горизонти для навчання, інтерактивного спілкування та глобального обміну знаннями. У таких умовах спеціалізовані вебсервіси для управління освітнім процесом в університетах перетворюються зі зручного інструменту на стратегічний ресурс, що формує майбутнє вищої освіти.

Завдяки своїй функціональності, такі вебсервіси забезпечують комплексний та зручний спосіб організації навчального процесу. Вони є ідеальними для централізованого управління інформацією, що є важливим для ефективної роботи університету. Викладачі можуть легко публікувати навчальні матеріали, оголошувати про зміни в розкладі, надавати завдання та отримувати зворотний зв'язок від студентів. Студенти, своєю чергою, мають зручний доступ до всіх необхідних ресурсів, можуть слідкувати за своїм прогресом та ефективно комунікувати з викладачами та одногрупниками. Такий інтегрований підхід сприяє підвищенню ефективності навчання та розвитку єдиного інформаційного простору університету [5].

Вебсервіси для організації та управління освітнім процесом університету відіграють центральну роль у створенні гнучкого та інтерактивного навчального

середовища, що відповідає сучасним вимогам доступності та оперативності інформації. Їхні переваги проявляються у багатьох аспектах.

По-перше, такі сервіси стають потужним інформаційним хабом вони забезпечують миттєвий доступ до всіх ключових даних з єдиної точки. Викладачі отримують зручний інструмент для оперативного оновлення курсів, розкладу та навчальних ресурсів, тоді як студенти можуть в один клік отримувати актуальну інформацію про будь-які зміни. Це формує прозоре інформаційне середовище, де кожен учасник навчального процесу завжди знаходиться на одній хвилі з університетським життям.

Другий аспект полягає в тому, що такі вебсервіси активно сприяють колаборативному навчанню. Вони надають інструменти для спільної роботи над проектами, організації групових дискусій, обміну ідеями та навіть проведення онлайн-семінарів. Це розширює можливості для взаємодії та співпраці як між студентами, так і між студентами та викладачами, незалежно від їхнього фізичного розташування.

Крім того, вебсервіси підтримують активне навчання. Інструменти для обговорення матеріалів, форуми, блоги та інтерактивні завдання залучають студентів до активної участі в навчальному процесі. Вони стають не просто пасивними споживачами інформації, а активними учасниками, які аналізують, мислять та висловлюють власні думки.

Ще одним важливим аспектом є те, що вебсервіси надають гнучкість у навчанні. Студенти можуть отримувати доступ до навчальних матеріалів та виконувати завдання у власному темпі та у зручний для них час. Можливість переглядати записи лекцій, брати участь в онлайн-дискусіях та отримувати індивідуалізований зворотний зв'язок робить навчання більш доступним та адаптованим до індивідуальних потреб кожного студента.

І нарешті, вебсервіси сприяють розвитку професійних зв’язків. Вони можуть слугувати платформою для налагодження контактів між студентами, викладачами, випускниками та представниками різних факультетів. Можливості

для створення професійних профілів, участі в тематичних групах та обміну досвідом сприяють професійному розвитку та розширенню мережі контактів.

Зважаючи на те, що стандартні вебплатформи можуть не повністю відповідати специфічним потребам університетської освіти, існує нагальна потреба в розробці спеціалізованих вебсервісів, орієнтованих на оптимізацію освітніх процесів. Основною проблемою універсальних платформ може бути недостатня функціональність для управління навчальним контентом та специфічними академічними потребами. Тому основним завданням розробників програмного забезпечення є створення спеціалізованих інструментів, які не лише відповідають сучасним вимогам академічного середовища, а й передбачають його майбутні потреби. Такі рішення мають поєднувати технологічну потужність з глибоким розумінням освітніх процесів, забезпечуючи зручність, ефективність та адаптивність для всіх учасників навчального процесу.

Важливим аспектом розробки спеціалізованих освітніх платформ є створення інтелектуальних систем управління контентом. Такі модулі мають забезпечувати викладачам зручний та зрозумілий механізм публікації та оновлення навчальних матеріалів будь-якого формату - від текстових документів і презентацій до мультимедійних та інтерактивних елементів. Ключовою вимогою до цих систем є інтуїтивність інтерфейсу, що дозволяє мінімізувати технічні бар'єри та максимально зосередитись на педагогічному процесі.

Не менш важливим напрямком є впровадження розвинених аналітичних інструментів. Такі системи мають надавати викладачам детальну інформацію про навчальну активність студентів: від відстеження базових показників до глибокого аналізу результатів тестувань та рівня залученості до навчального процесу. Отримані дані дозволять не лише оцінювати ефективність навчальних методик, а й оперативно адаптовувати освітній процес до індивідуальних потреб студентів, що є ключовим чинником у підвищенні якості вищої освіти [6].

Отже, розробка спеціалізованих вебсервісів для організації та управління освітнім процесом університету є важливим завданням для модернізації сучасної вищої освіти. Такі інноваційні рішення трансформують традиційну освіту,

відкриваючи нові перспективи для навчання, викладання та міжособистісної взаємодії. Вони роблять освітній процес більш гнучким, персоналізованим та адаптивним, що суттєво підвищує його ефективність і доступність. У світі, де цифрові технології стрімко еволюціонують, університети повинні не лише адаптуватися до цих змін, а й активно формувати цифровий освітній простір лише так вони зможуть підтримувати високі стандарти якості освіти та зберігати свою конкурентоспроможність у глобальному освітньому середовищі.

1.2. Порівняльний аналіз аналогів

В умовах цифровізації освіти програмні платформи для управління освітнім процесом відіграють ключову роль у забезпеченні ефективного навчання. Вони надають університетам сучасні інструменти для організації навчального контенту, комунікації між учасниками освітнього процесу та моніторингу успішності студентів. Існує ряд популярних розв'язань для організації дистанційного навчання та управління навчальними курсами, серед яких:

- Moodle;
- Google Classroom;
- Blackboard Learn;
- Canvas LMS;
- Open edX.

Moodle [7] (див. рисунок 1.1) – це одна з найпопулярніших систем управління навчанням з відкритим кодом. Вона підтримує розширене налаштування курсів, можливість інтеграції з різними інструментами та плагінами. Moodle використовується університетами, школами та компаніями для організації навчальних програм. Система дозволяє автоматизувати оцінювання, відстежувати успішність студентів і надавати гнучкі можливості адміністрування. Moodle також підтримує роботу в офлайн-режимі та синхронізацію даних при відновленні зв’язку, що є важливим для користувачів з обмеженим доступом до інтернету.

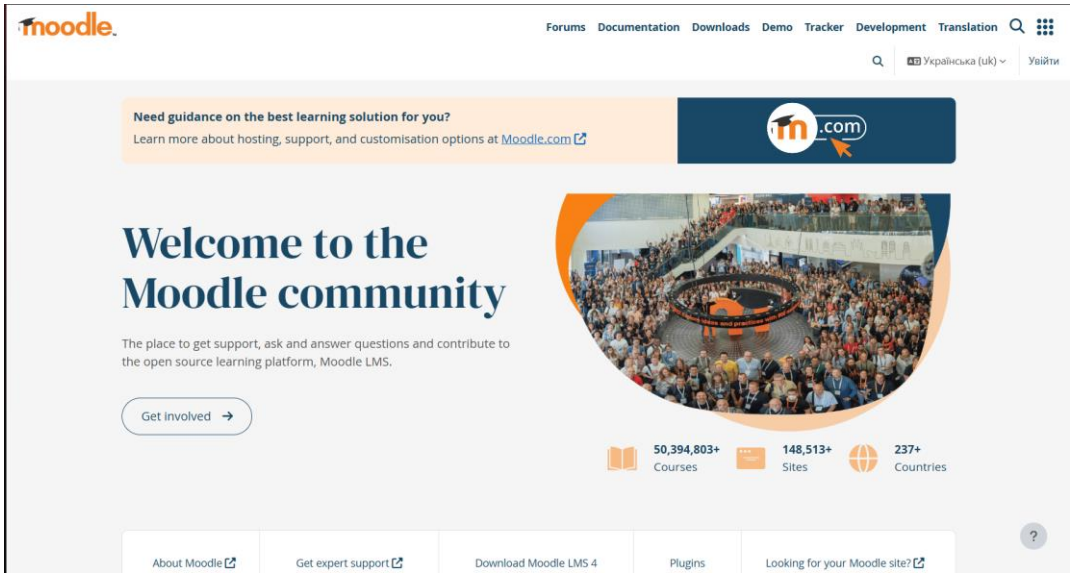


Рисунок 1.1 – Приклад роботи вебсервісу Moodle

Інший прикладом є вебсервіс Google Classroom [8] (див. рисунок 1.2). Це хмарна платформа для онлайн-навчання, інтегрована з екосистемою Google. Вона забезпечує зручний інтерфейс, спрощену систему керування завданнями та інтеграцію з Google Drive. Google Classroom ідеально підходить для шкіл та університетів, які використовують сервіси Google Workspace у своєму освітньому процесі. Цей сервіс дозволяє легко створювати та розповсюджувати навчальні матеріали, а також перевіряти роботи студентів

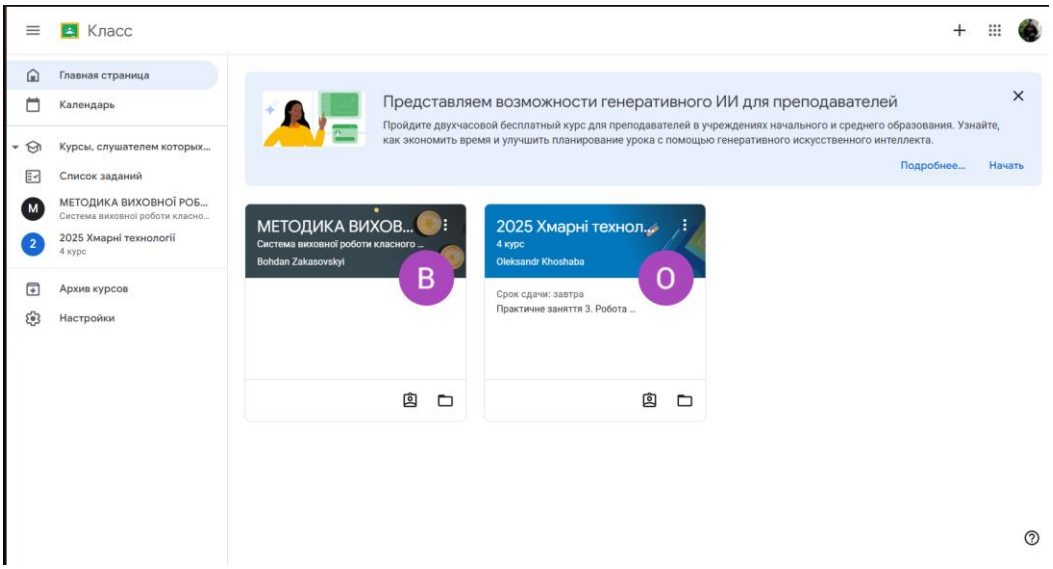


Рисунок 1.2 – Приклад роботи вебсервісу Google Classroom

Blackboard Learn [9] (див. рисунок 1.3) – потужна LMS, що використовується в багатьох університетах для управління освітніми курсами. Вона пропонує розширені можливості для персоналізації навчання, підтримку інтеграції зі сторонніми сервісами та аналітику для оцінки прогресу студентів. Blackboard Learn підходить для великих освітніх установ із складними програмами навчання. Її функціонал дозволяє створювати складні курси з великою кількістю матеріалів та активностей.

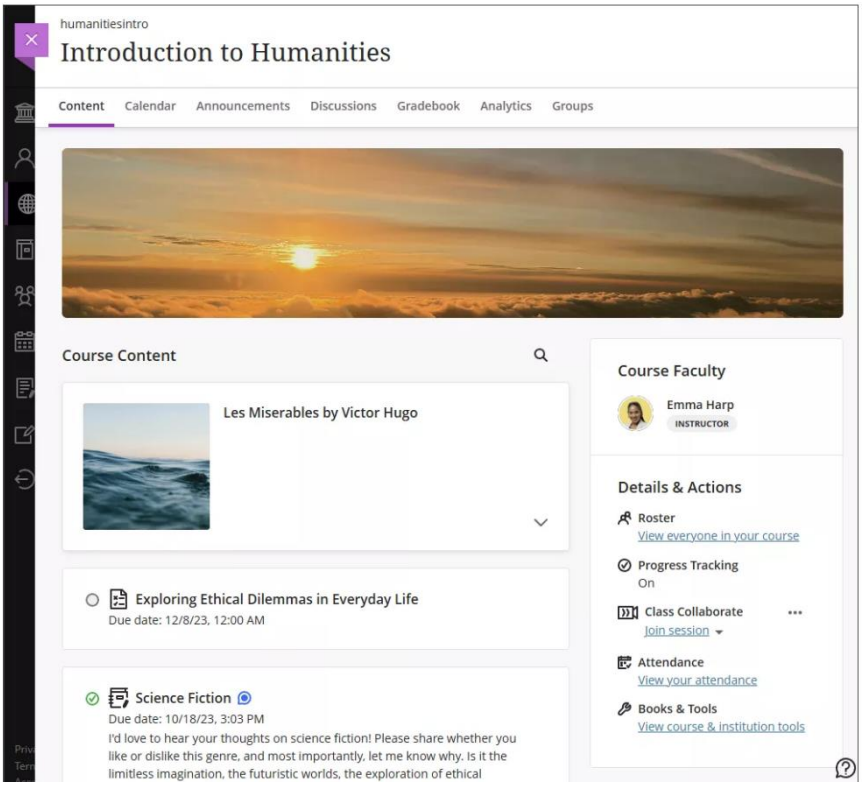


Рисунок 1.3 – Приклад роботи вебсервісу Blackboard Learn

Canvas LMS [10] (див. рисунок 1.4) – це сучасна, масштабована LMS із відкритим кодом, яка підтримує гнучкі інструменти для створення курсів, автоматизацію оцінювання та аналітику. Система відзначається інтуїтивно зрозумілим інтерфейсом та підтримкою спільної роботи, що робить її особливо привабливою для сучасних освітніх потреб. Вона широко використовується у вищих навчальних закладах та корпоративному навчанні завдяки інтеграції з зовнішніми сервісами та мобільним додаткам.

Рисунок 1.4 – Приклад роботи вебсервісу Canvas LMS

Open edX [11] – це відкрита платформа для створення масових відкритих онлайн-курсів MOOC. Вона використовується провідними університетами для розробки онлайн-курсів, забезпечуючи доступ до освітніх матеріалів великій кількості студентів у всьому світі. Open edX (див. рисунок 1.5) відрізняється широкими можливостями кастомізації та підтримкою інтерактивних навчальних матеріалів.

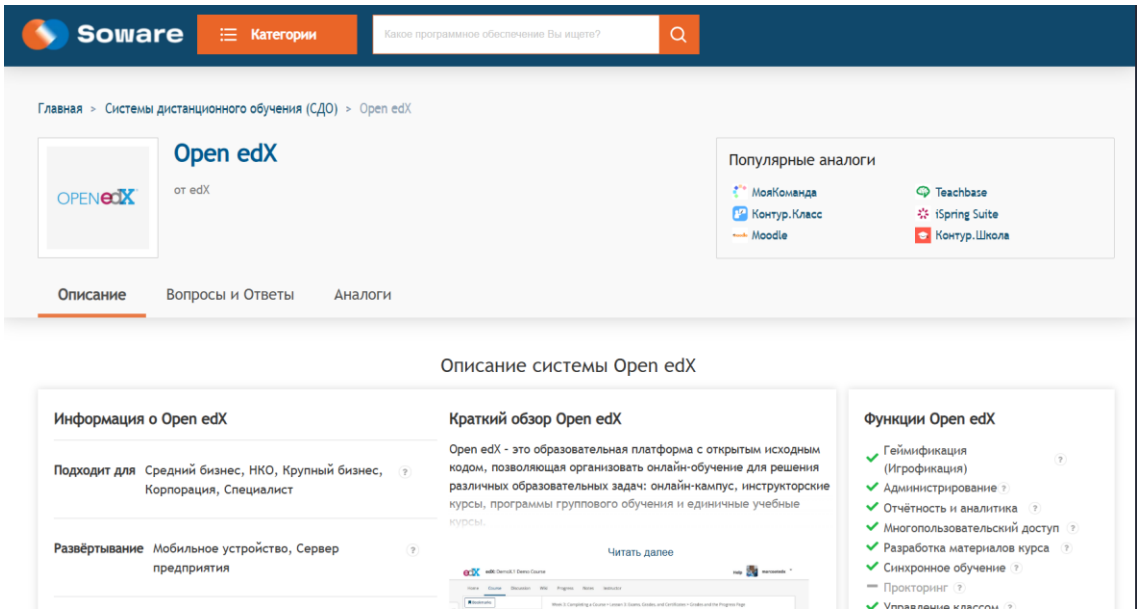


Рисунок 1.5 – Приклад роботи вебсервісу Open edX

Ці інструменти демонструють важливість та потенціал використання інформаційних технологій в освітньому процесі, пропонуючи студентам та викладачам зручні для управління завданнями та комунікацією. Розробка подібних систем вимагає глибокого розуміння потреб користувачів, а також технічної експертизи для створення інтуїтивно зрозумілих, надійних та ефективних вебсервісів, здатних покращити доступність та якість освітнього процесу. В результаті розгляду аналогів було створено таблицю 1.1 для порівняння їхнього функціоналу з власною розробкою.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	Moodle	Google Classroom	Blackboard Learn	Canvas LMS	Open edX	Власна розробка
Простий і інтуїтивно зрозумілий інтерфейс	0	1	0	1	0	1
Отримання інформації про користувача	1	0	1	1	1	1
Авторизація та реєстрація	1	1	0	1	1	1
Перегляд розкладу користувача	0	0	1	1	0	1
Тестування програмного забезпечення	0	0	1	0	0	1
Сумарний коефіцієнт	2	2	3	4	2	5

Відповідно до таблиці порівняльних характеристик, розробка власних програмних модулів для управління освітнім процесом університету є доцільною. Створений продукт зможе усунути недоліки існуючих рішень, таких

як Moodle, Google Classroom, Blackboard Learn, Canvas LMS та Open edX, і надати розширені функціональні можливості для ефективного управління навчальним процесом.

Отже, розробка власного програмного модуля відкриває перед університетами нові перспективи для покращення організації навчання, автоматизації оцінювання та аналітики успішності студентів. Завдяки інтуїтивно зрозумілому інтерфейсу, можливості отримання детальної інформації про студентів та викладачів, а також функціоналу перегляду розкладу, система забезпечить зручність використання для всіх учасників освітнього процесу. Це дозволить підвищити якість освіти, оптимізувати роботу викладачів та забезпечити більш гнучке та індивідуалізоване навчання для студентів.

1.3 Аналіз методів розробки вебсервісу

У процесі створення програмних засобів вебсервісу для організації та управління освітнім процесом університету, потрібно чітко проаналізувати різноманітні підходи до розробки такого вебсервісу. У процесі створення вебсервісу застосовуються різні методи, кожен з яких має власні сильні та слабкі сторони. Врахування цих особливостей дозволяє знаходити найкращі рішення для досягнення поставлених цілей.

Створення власної платформи для вебсервісу є одним із можливих та досить перспективних методів розв’язання завдання. Такий метод дає розробникам широкі можливості, дозволяючи повністю керувати логікою роботи системи та її структурою. Використання власного сервісу забезпечує можливість гнучкого налаштування відповідно до специфічних потреб університету та особливостей його освітніх ініціатив.

Цей підхід пропонує виняткову гнучкість на всіх етапах розробки та підтримки продукту. Він дозволяє повністю кастомізувати функціонал, архітектуру, дизайн та процеси інтеграції з іншими академічними системами. Ключова перевага полягає в можливості створювати індивідуальні рішення для кожного аспекту вебсервісу, що забезпечує ефективне впровадження нових

функцій та швидку адаптацію до змін у вимогах користувачів, включаючи студентів, викладачів та адміністративний персонал університету.

Однак, слід розуміти, що розробка власної платформи може бути затратною за часом і ресурсами. Вона вимагає значних зусиль на кожному етапі, починаючи від глибокого аналізу потреб та проєктування і закінчуючи безпосередньо розробкою, ретельним тестуванням та масштабним впровадженням. Крім того, такий процес може забирати значну кількість людських ресурсів, аналітиків, програмістів, тестувальників, дизайнерів та потребує значних фінансових витрат, що зумовлює необхідність ретельного планування та ефективного керування проєктом. В цілому, створення власної платформи для вебсервісу управління освітнім процесом є варіантом з великим потенціалом для успіху, особливо якщо університет має унікальні потреби, складну структуру та специфічні вимоги до автоматизації своїх процесів. Цей підхід надає можливість реалізувати продукт, що ідеально відповідає специфіці університету, його освітнім програмам та адміністративним процедурам, забезпечуючи гнучкий та індивідуальний підхід до розробки.

Використання відкритих рішень та готових платформ для створення вебсервісу є ще одним можливим методом, який варто розглянути. Цей підхід показує використання вже існуючих систем управління навчанням з відкритим кодом, таких як Moodle, Open edX або Sakai, які можна налаштувати та адаптувати під потреби університету.

Переваги такого підходу досить очевидні, по-перше, він може значно зменшити час розробки та витрати, оскільки фахівці можуть скористатися готовими функціональними модулями та компонентами, замість того щоб розробляти їх з нуля. Це дозволяє прискорити впровадження сервісу та забезпечити більш швидке повернення інвестицій. Крім того, використання відкритих рішень може надати доступ до широкого спектру вже розроблених та протестованих функцій та можливостей, які підтримуються активною спільнотою фахівців. Це може включати такі ключові компоненти, як системи керування курсами, інструменти для створення навчального контенту, форуми

для обговорень, системи оцінювання, інструменти для відеоконференцій тощо.

Однак, варто розуміти, що цей шлях може обмежити гнучкість та особливі можливості розробленого вебсервісу. Використання готових розвязків може призвести до того, що деякі специфічні потреби університету можуть бути не повністю задоволені стандартним функціоналом цих систем. Крім того, інтеграція з існуючими університетськими системами може виявитися складнішою та потребувати додаткових зусиль з програмування. Також існує певний ризик залежності від спільноти фахівців відкритого коду щодо підтримки, оновлень та виправлення помилок.

Третій можливий метод є інтеграція розробленого засобу з існуючими освітніми платформами або сервісами, такими як Google Classroom, Microsoft Teams for Education або інші комерційні LMS. Це дозволить забезпечити певну інтеграцію з інструментами, які вже можуть використовуватися в університеті, та спростити для користувачів перехід до нової системи. Проте такий підхід має й свої обмеження, він може суттєво звужувати спектр функціональних можливостей та рівень контролю над збором і обробкою даних. Фахівцям доведеться працювати в рамках обмежень, що їх накладають API та функціонал сторонніх платформ, що може ускладнювати реалізацію індивідуальних рішень та гнучкість управління даними.

Ураховуючи переваги та недоліки кожного з методів, для розробки вебсервісу для організації та управління освітнім процесом університету рекомендується ретельно зважити потреби та ресурси університету. Для університетів з унікальними та складними вимогами, а також достатніми ресурсами, розробка власної платформи може забезпечити максимальну гнучкість та відповідність потребам. Для університетів з обмеженими ресурсами або потребою у швидкому розгортанні базового функціоналу, використання відкритих рішень може бути більш економічно ефективним варіантом. Комбінований підхід, коли використовуються окремі готові компоненти або інтегруються існуючі сервіси для певних функцій, а основна логіка розробляється індивідуально, також може бути розглянутий як компромісний

варіант.

В результаті детального дослідження різних підходів до створення програмного забезпечення вебсервісу для організації та управління освітнім процесом університету, вибір оптимального підходу залежить від конкретних потреб, ресурсів та стратегічних цілей університету.

Розробка власної платформи надасть повний контроль над продуктом. Це означає, що буде можливість налаштувати функціонал та архітектуру системи відповідно до конкретних потреб освітнього процесу університету, забезпечуючи максимальну гнучкість, індивідуальність та інтеграцію з існуючою інфраструктурою.

Розробка дозволить забезпечити його глибоку адаптацію до специфічних потреб університету. Різні факультети, освітні програми та адміністративні підрозділи можуть мати унікальні вимоги до функціоналу вебсервісу, і розробка власної платформи дозволить врахувати ці потреби та створити інструмент, який оптимально підтримує всі аспекти освітньої діяльності.

Хоча розробка власного вебсервісу може вимагати значних зусиль на початкових етапах, втім, таке рішення відкриває більше перспектив для подальшого вдосконалення та гнучкої адаптації, масштабування та вдосконалення продукту відповідно до майбутніх потреб університету та технологічних інновацій. Отже, на основі цих розглядів, розробка власної платформи є стратегічно вигідним вибором для реалізації комплексного та ефективного вебсервісу для організації та управління освітнім процесом університету, за умови наявності відповідних ресурсів та експертизи.

1.4 Постановка задач дослідження

На основі аналізу функціональних можливостей та обмежень існуючих платформ для управління навчальним процесом було сформульовано основні напрямки розробки нового програмного забезпечення:

– провести аналіз існуючих вебсервісів та визначити їхні можливості для керування освітнім процесом;

– програмно реалізувати функціонал вебсервісу для керування освітнім процесом;

– розробити зручний інтерфейс користувача, який буде легким у використанні;

– розробка бази даних, яка зберігатиме користувацькі дані, освітній контент, даючи швидкий доступ та високу продуктивність;

– розробка модуля автентифікації та реєстрації користувачів для підтримки додаткових функцій та безпеки платформи;

– розробка модуля пошуку студента за логіном, що дає швидко знайти потрібного студента;

– провести тестування розробленої вебсистеми.

1.5 Висновки

У першому розділі було проведено аналіз сучасного стану вебсервісів для організації та управління освітнім процесом університету. Результати підкреслили важливість розвитку таких систем для підтримки освітніх процесів, що може значно полегшити адміністрування навчання, покращити комунікацію та підвищити ефективність освітнього процесу загалом. Порівняльний аналіз платформ, таких як Moodle, Canvas та Blackboard Learn, підтвердив наявність як потужних, так і менш функціональних рішень на ринку, що вказує на потребу в оптимізованих та спеціалізованих інструментах [12].

Детальне дослідження існуючих вебсервісів для управління навчальним процесом виявило низку проблем, що стало основою для формування технічного завдання на розробку вебсервісу організації та управління освітнім процесом університету. Цей висновок підкріплюється потребою в індивідуальних рішеннях, специфічних вимогах університетського середовища та швидким розвитком цифрових технологій. Отже, дослідження підтвердило актуальність та необхідність розробки програмних засобів вебсервісу для організації та управління освітнім процесом університету, з фокусом на створенні функціоналу, який максимально відповідає потребам закладу.

2 РОЗРОБКА СТРУКТУРИ, МОДЕЛІ ТА МЕТОДУ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Розробка структури інтерфейсу програмного засобу

Розробка користувальського інтерфейсу для вебсервісу організації та управління освітнім процесом університету є важливою частиною у створенні ефективної системи. Цей процес демонструє проєктування структури взаємодії користувача з платформою, визначаючи спосіб представлення інформації та інструментів. Інтуїтивно зрозуміле розташування елементів інтерфейсу на вебсторінках є важливим для забезпечення зручності та продуктивності роботи користувачів.

Основна мета цього етапу полягає у формуванні логічної та послідовної архітектури навігації. Це включає систематизацію інформаційних блоків та розробку ієрархічної структури сторінок, що дозволяє користувачам швидко орієнтуватися та знаходити необхідні ресурси чи функції. Важливо забезпечити передбачуваність розміщення елементів, щоб користувач міг інтуїтивно розуміти їх призначення та спосіб використання.

Ключовим аспектом проєктування інтерфейсу є орієнтація на кінцевого користувача. Для вебсервісу університету це охоплює студентів, викладачів та адміністративний персонал. Розуміння їхніх потреб, робочих сценаріїв та технічних навичок є визначальним фактором при розробці структури інтерфейсу, що максимально відповідає їхнім вимогам та сприяє ефективній роботі з системою.

Принципи проєктування користувальського інтерфейсу є основою для створення зручного та ефективного вебсервісу. До них належать простота, послідовність, передбачуваність, зворотний зв'язок та толерантність до помилок. Забезпечення легкості навчання та використання системи, уніфікація елементів керування та візуального стилю, а також надання чітких інструкцій та повідомлень про дії користувача є важливими складовими успішного інтерфейсу.

Одним з фундаментальних принципів є мінімалізм. Перевантаження інтерфейсу зайвими елементами може ускладнити навігацію та знизити ефективність роботи. Узгодженість у використанні візуальних рішень та поведінки елементів інтерфейсу створює відчуття стабільності та полегшує освоєння нових функцій.

Врахування особливостей різних груп користувачів є обов'язковим. Студенти можуть потребувати швидкого доступу до навчальних матеріалів та розкладу, викладачі до інструментів керування курсами та оцінювання, а адміністративний персонал до функцій адміністрування користувачів та генерації звітів. Інтерфейс повинен бути адаптованим до потреб кожної з цих ролей.

Процес розробки інтерфейсу є послідовним і включає етапи аналізу користувацьких вимог, створення ескізів та каркасів інтерфейсу розробку інтерактивних прототипів, проведення тестування з реальними користувачами та внесення коректив на основі отриманих результатів. Такий підхід дозволяє створити інтерфейс, що максимально відповідає потребам користувачів та забезпечує високий рівень зручності використання.

Початкова сторінка вебсервісу дає швидкий доступ до ключових розділів та інформації. Залежно від ролі користувача, вона може містити персоналізовану панель навігації з посиланнями на основні функціональні блоки, такі як перелік курсів, розклад, івентів, особистий профіль та налаштування. Елементи швидкого доступу до часто використовуваних функцій, а також інформаційні блоки з актуальними оголошеннями та подіями університету, можуть бути розміщені на головній сторінці для підвищення оперативності доступу до важливої інформації.

Персоналізація контенту головної сторінки, враховуючи роль та інтереси користувача, сприяє підвищенню залученості та ефективності використання вебсервісу. Чітка структура та логічне розташування елементів інтерфейсу забезпечують зручну та інтуїтивно зрозумілу взаємодію користувачів з платформою, сприяючи ефективному виконанню їхніх навчальних та

адміністративних завдань.

На рисунку 2.1 детально зображено інтерфейс головної сторінки вебсервісу з усіма згаданими елементами, ілюструючи взаємодію користувача з ним.

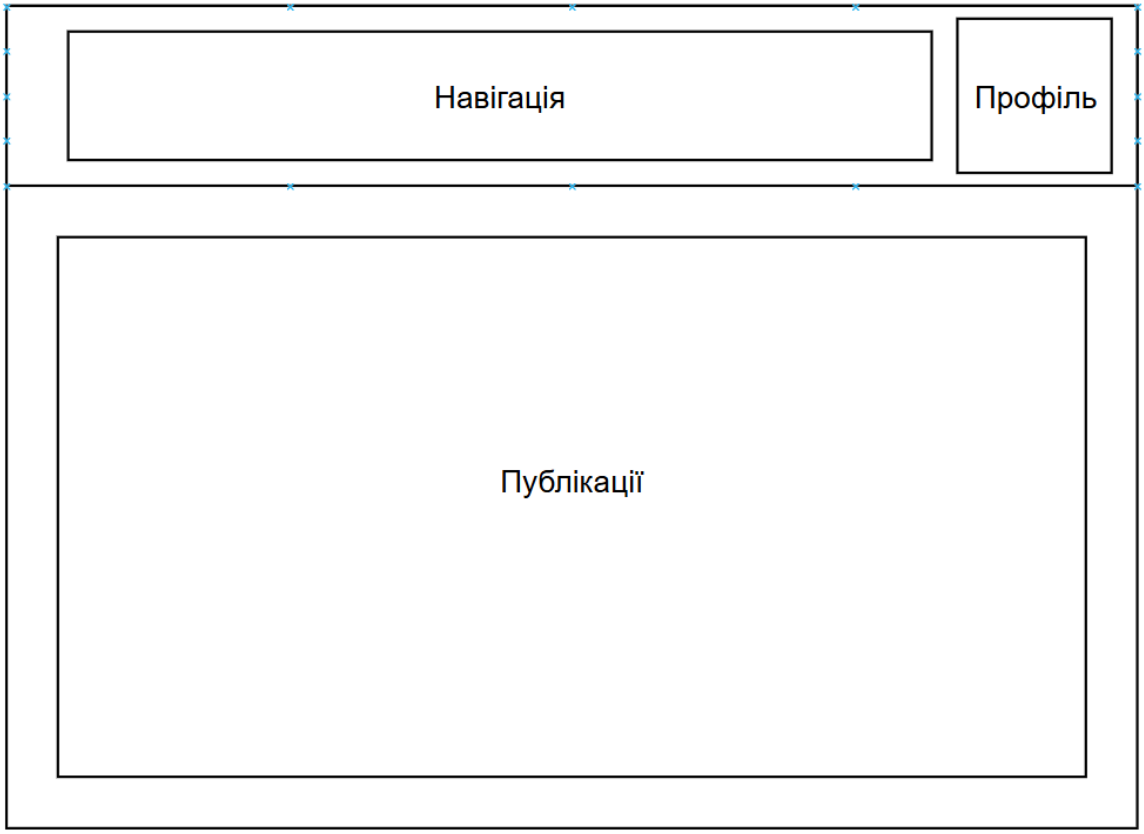


Рисунок 2.1 – Структура інтерфейсу головної сторінки

На сторінці профілю користувача відвідувачі знаходять зручну структуру інтерфейсу, яка забезпечує легкий доступ до всіх важливих аспектів соціальної мережі. Верхня частина сторінки включає панель навігації, яка дозволяє користувачам швидко переходити до основних розділів сайту, таких як курси, предмети, перегляд груп та створення класів. Також на верхня панель містить профіль, який дозволяє перегляд власного профілю та доступ до налаштувань акаунта. Нижче на сторінці розташована панель публікацій, яка надає користувачам можливість перегляду новин та івентів університету.

Структура інтерфейсу сторінки студентів зображена на рисунку 2.2.

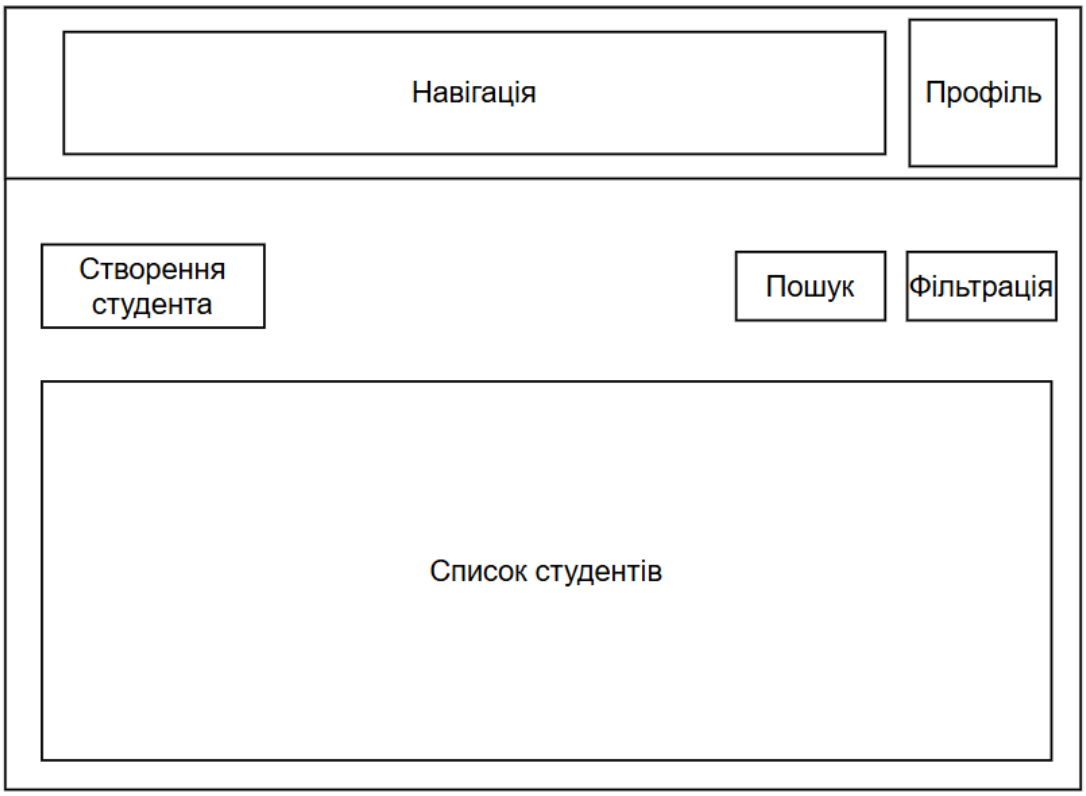


Рисунок 2.2 – Структура інтерфейсу сторінки перегляду студентів

Під панеллю інструментів розташовано головну робочу область інтерфейсу, яка включає функціональні кнопки для створення нового студента, пошуку за ключовими словами та фільтрації результатів. Ці елементи інтерфейсу дозволяють користувачу швидко додавати нову інформацію до системи, знаходити потрібні записи та керувати відображенням даних у списку студентів.

Центральне місце займає блок "Список студентів", який є основним інформаційним елементом сторінки. Тут відображається актуальна інформація про студентів: їхні імена, контактні дані, статус активності тощо. Цей список може бути реалізований у вигляді таблиці або карткового представлення з можливістю взаємодії редагуванням, переглядом або видаленням записів.

Таким чином, дана сторінка є ключовою для адміністративної роботи з базою студентів, дозволяючи ефективно управляти інформацією, швидко орієнтуватися в даних та здійснювати точкові або масові операції над записами. Подібна структурна організація робить робочий процес більш зручним, наочним

та продуктивним як для викладачів, так і для адміністраторів навчального закладу.

Не менш важливо проаналізувати будову сторінки створення студента, яка є ключовою частиною функціоналу системи управління навчальним процесом. Вона надає адміністраторам або викладачам зручний інтерфейс для внесення нових студентів до бази даних, забезпечуючи повне охоплення необхідної інформації.

Загальну структуру інтерфейсу сторінка створення студента відображено на рисунку 2.3.

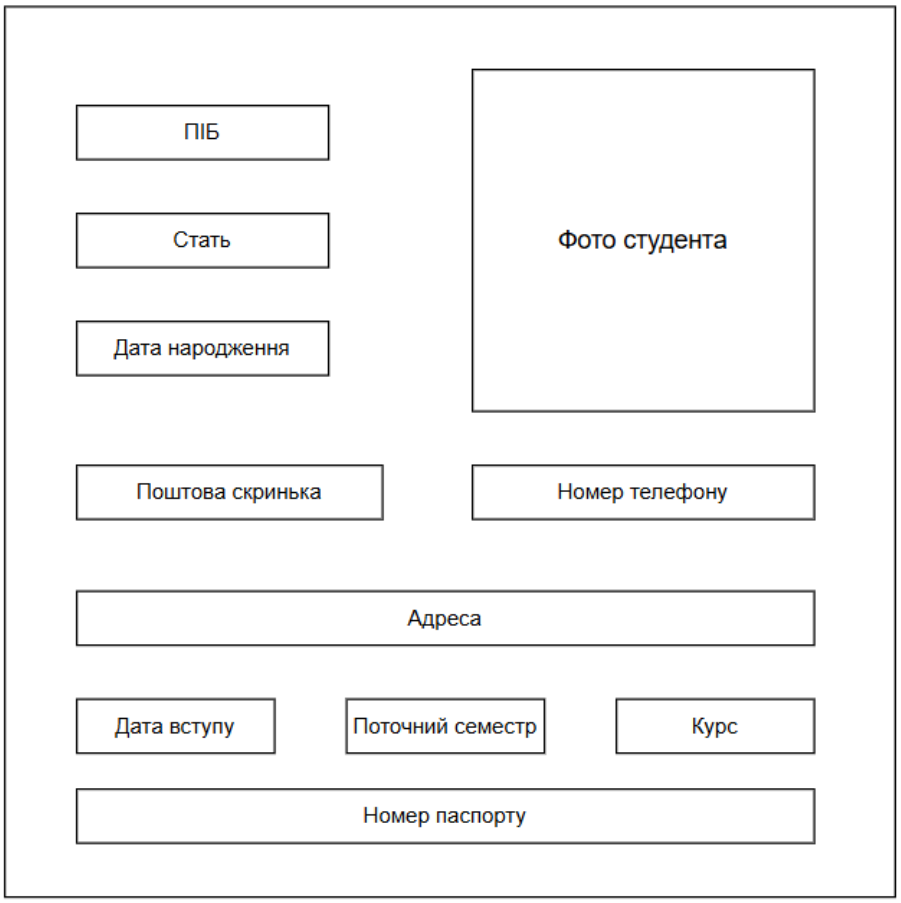


Рисунок 2.3 – Структура інтерфейсу сторінки створення студента

Центральним елементом цієї сторінки є форма введення даних, яка складається з декількох логічно згрупованих полів. Вони охоплюють особисту інформацію студента, включаючи повне ім’я, стать, дату народження, адресу

проживання, номер телефону, електронну пошту та паспортні дані. Додатково вказуються дані щодо навчального процесу: дата вступу, поточний семестр і обраний курс навчання. Такий підхід забезпечує створення повного та структурованого запису кожного студента.

Таким чином, сторінка створення студента виконує не лише адміністративну функцію, а й є важливим інструментом для забезпечення цілісності та актуальності бази даних, підвищуючи точність обліку та спрощуючи процес внесення нових осіб до системи.

Застосування програмного забезпечення Visio дозволило візуалізувати структуру інтерфейсу вебсервісу для організації та управління освітнім процесом університету, що сприяє оптимізації взаємодії між усіма учасниками освітнього процесу. Створена структура забезпечує ефективне використання функціоналу платформи, швидку навігацію між ключовими розділами та легкий доступ до основних інструментів, персоналізованої інформації та університетських сервісів. Це, своєю чергою, підвищує зручність роботи викладачів, студентів та адміністративного персоналу, сприяючи більш ефективній організації навчального процесу та залученню до активної взаємодії в цифровому середовищі університету.

2.2 Розробка методу реєстрації та авторизації користувача

У даному вебсервісі реалізовано метод реєстрації та входу користувачів, що дає базові заходи безпеки: використання хешування паролів та сеансної автентифікації.

Для захисту облікових записів застосовується хешування паролів перед їх збереженням у базі даних. Один із поширених підходів є алгоритм BCrypt, який вирізняється стійкістю до атак завдяки кільком технічним особливостям [13].

Зокрема, BCrypt додає до пароля випадкову послідовність символів, що унеможливлює повторення хешів навіть для однакових паролів різних користувачів. Це знижує ефективність атак із використанням попередньо обчислених таблиць хешів.

Ще однією важливою властивістю цього алгоритму є можливість регулювання складності обчислень, що дозволяє підлаштувати захист під актуальний рівень технічного прогресу. Зростання кількості ітерацій ускладнює перебір паролів навіть при високій обчислювальній потужності. Це дає змогу підтримувати авторизований стан протягом певного часу без необхідності повторного введення пароля при кожному запиті. Принцип роботи авторизації на основі сеансу у вигляді блок-схеми входу користувача у систему зображено на рисунку 2.4.

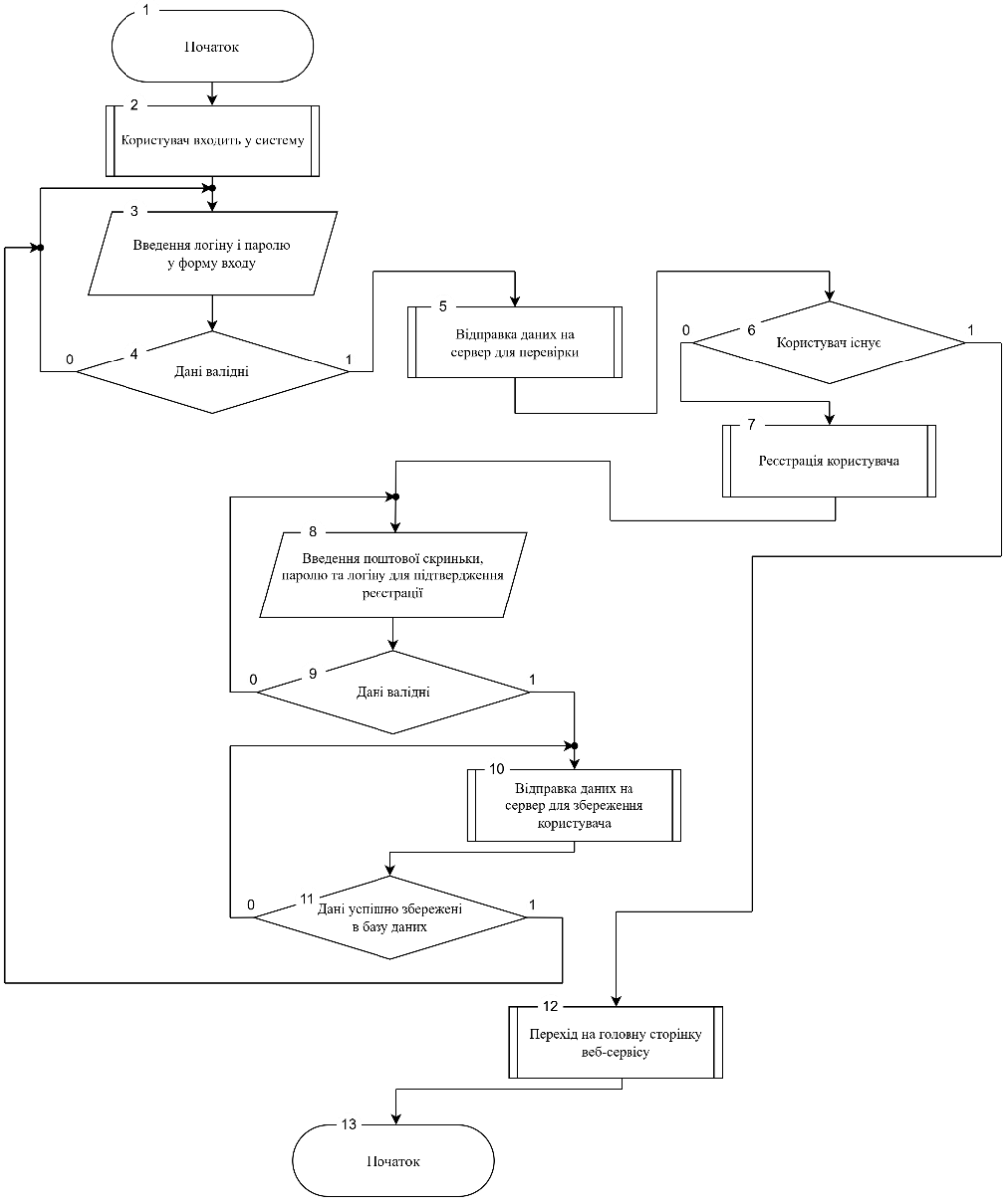


Рисунок 2.4 – Блок-схема входу користувача у систему

Процес входу або реєстрації користувача у вебсистемі розпочинається з того, що користувач заходить на відповідну сторінку. Далі йому пропонується ввести свою електронну пошту та пароль у форму входу, після чого система перевіряє валідність цих даних. Якщо введені дані некоректні, користувачеві буде запропоновано ввести їх повторно. У випадку валідних даних, вони відправляються на сервер для перевірки. Сервер, отримавши дані, перевіряє, чи існує вже користувач з такими обліковими даними. Якщо такого користувача не знайдено, запускається процедура реєстрації, де користувачеві необхідно ввести ім'я, електронну пошту, пароль та підтвердження паролю. Введені реєстраційні дані також проходять перевірку на валідність, і у разі некоректності користувачеві буде запропоновано їх виправити. Після успішної валідації реєстраційні дані відправляються на сервер для збереження в базі даних. Якщо дані успішно збережено, або якщо на кроці перевірки входу було знайдено відповідного користувача, йому відображається особистий кабінет з доступним функціоналом вебсистеми.

Метод авторизації відповідає за безпечну та безперебійну ідентифікацію користувачів у системі, а також за контроль їхніх прав доступу. Це є головним для захисту особистої інформації та забезпечення зручності роботи.

2.3 Розробка моделі роботи вебсервіс для організації та управління освітнім процесом університету

Модель функціонування вебсервісу для організації та управління освітнім процесом університету дає можливість створення та керування різноманітним навчальним контентом, включаючи оголошення, навчальні матеріали, завдання та посилання на зовнішні ресурси, з інструментами для редагування та оцінювання. Сервіс також може включати механізми персоналізації рекомендацій навчальних курсів, додаткових матеріалів та навчальних груп на основі навчальних планів, історії активності користувачів та їхніх навчальних цілей. Це дозволяє надавати користувачам найбільш релевантну інформацію, оптимізувати процес навчання та підвищити їхню залученість до освітнього

середовища університету.

Нижче зображено на рисунку 2.5 діаграму варіантів вебсервісу.

Рисунок 2.5 – Діаграма варіантів для вебсервісу

Діаграма варіантів відображає ключові функціональні процеси, що забезпечують ефективну взаємодію між усіма учасниками навчального середовища. До цих процесів належать реєстрація нових користувачів,

автентифікація для отримання доступу до системи, керування навчальним контентом: створення, редагування, завантаження матеріалів, перегляд навчальних курсів та ресурсів, виконання та оцінювання завдань, а також комунікація між студентами, викладачами та адміністрацією. Вона детально ілюструє послідовність дій користувачів на різних етапах їхньої роботи з вебсервісом.

Функціонування вебсервісу розпочинається з етапу ідентифікації та авторизації користувачів. Система здійснює перевірку облікових даних і надає доступ до відповідних функцій залежно від ролі користувача лише після успішної авторизації. Залежно від своїх прав, користувачі можуть керувати навчальними матеріалами, переглядати розклад занять, брати участь у форумах обговорень, здавати та перевіряти завдання, а також отримувати доступ до персоналізованої інформації.

Для оптимізації навчального процесу та надання користувачам найбільш релевантної інформації, вебсервіс може використовувати алгоритми персоналізації. Ці алгоритми аналізують навчальні плани студентів, їхню історію активності в системі: перегляди курсів, виконані завдання, оцінки, а також інтереси, зазначені в профілі. На основі цього аналізу система формує персоналізовані рекомендації щодо навчальних курсів, додаткових матеріалів та груп для спільної роботи. Це забезпечує високу релевантність контенту та сприяє підвищенню ефективності навчання.

Діаграма варіантів детально відображає основні процеси, що відбуваються в системі. Користувачі проходять етапи реєстрації та авторизації, після чого отримують можливість переглядати навчальні матеріали, виконувати завдання, спілкуватися з іншими учасниками навчального процесу та керувати власним навчальним планом. Діаграма також ілюструє, як система обробляє запити користувачів, перевіряє їхні права доступу до різних ресурсів та функцій, а також надає персоналізовані рекомендації. Наприклад, коли викладач завантажує новий навчальний матеріал, система може автоматично рекомендувати його студентам, які вивчають відповідний курс або мають схожі навчальні інтереси.

Завдяки впровадженню алгоритмів персоналізації та методів аналізу навчальної діяльності, вебсервіс покращує якість взаємодії між усіма учасниками освітнього процесу. Це сприяє підвищенню активності студентів, покращенню комунікації між викладачами та студентами, а також оптимізації адміністративних процесів. В кінцевому результаті, така система забезпечує більш персоналізований та ефективний досвід для всіх користувачів, що є ключовим фактором у створенні успішного вебсервісу для організації та управління освітнім процесом університету.

2.4 Розробка структури бази даних

Для ефективної роботи будь-якого вебзастосунку надзвичайно важливо мати грамотно спроєктовану структуру бази даних. Вона повинна забезпечувати оптимальне зберігання, обробку та доступ до інформації, а також відповідати всім вимогам системи, уникати дублювання даних і забезпечувати зручність у їхньому адмініструванні.

Одним із ключових принципів, що застосовується при проєктуванні структури бази даних, є нормалізація. Цей процес полягає в логічному розподілі даних по окремих таблицях з метою уникнення надмірностей і зменшення зв'язків між ними. Завдяки нормалізації вдається зменшити дублювання інформації, спростити її оновлення та підтримати цілісність даних. У базі даних вебзастосунку таблиці зберігають дані про користувачів, ролі та повідомлення окремо, що підвищує продуктивність і знижує ймовірність виникнення помилок при зміні даних – таких як помилки вставки, оновлення або видалення.

Ще одним важливим методом покращення продуктивності бази даних є використання індексів. Індексація значно прискорює виконання пошукових запитів і доступ до необхідної інформації. Усі таблиці мають унікальні ідентифікатори, які використовуються як первинні ключі. Це не лише прискорює виконання запитів, а й підтримує унікальність записів, що сприяє збереженню цілісності даних.

Крім того, особливе значення має організація зв’язків між таблицями. Вони

дозволяють відобразити логічні взаємозв’язки між сутностями та забезпечити ефективне виконання складних запитів. Наприклад, зв’язок між таблицями користувачів і повідомлень дає змогу швидко визначити, які повідомлення належать конкретному користувачу. Такі зв’язки значно полегшують процес обробки даних і вибірки, що в результаті підвищує швидкодію системи та покращує загальний користувацький досвід.

Таблиця 2.1 ілюструє особливості зв’язків між елементами структури бази даних, що використовується у вебсервісі для організації та управління освітнім процесом університету [14].

Таблиця 2.1 – Характеристика зв’язків бази даних

Ім’я суті 1	Ім’я суті 2	Тип зв’язку	Клас належності
Student	Group	N:1	Необов., Обов.
Student	Course	N:M	Обов., Обов.
Course	Student	M:N	Обов., Обов.
Course	Teacher	N:1	Необов., Обов.
Course	Lesson	1:N	Обов., Необов.
Group	Student	1:N	Обов., Необов.
Lesson	Course	N:1	Обов., Обов.
Lesson	Room	N:1	Необов., Обов.
Room	Lesson	1:N	Обов., Необов.

Модель організації студентів ґрунтується на зв'язку типу "багато до одного" між сутностями Student та Group. У цій моделі кожен окремий студент може належати лише до однієї конкретної групи, у той час як кожна група може об'єднувати у собі цілу когорту студентів. Важливо зазначити, що цей зв'язок має різний ступінь обов'язковості для кожної з сутностей: для студента приналежність до групи є необов'язковою, тоді як для групи наявність хоча б одного студента є обов'язковою вимогою. Така організація даних дозволяє гнучко керувати навчальними колективами, забезпечуючи при цьому

можливість тимчасового існування студентів поза груповою структурою.

Ще одним фундаментальним зв'язком у системі є відношення "багато до багатьох" між Student та Course, яке реалізовано через спеціальну проміжну таблицю student_courses. Ця структура дає можливість кожному студенту бути записаним на довільну кількість курсів, і одночасно кожен курс може включати необмежене число студентів. Особливістю цього зв'язку є його двостороння обов'язковість - система вимагає, щоб кожен студент був приписаний хоча б до одного курсу, і кожен курс містив мінімум одного студента. Така організація даних становить основу академічного процесу, відображаючи реальну практику розподілу навчального навантаження.

Важливу роль у системі відіграє зв'язок "багато до одного" між Course та Teacher. У цій моделі кожен окремий курс закріплений за конкретним викладачем, при цьому один викладач може відповідати за цілий набір різних курсів. Ступінь обов'язковості цього зв'язку асиметричний: курс може існувати у системі тимчасово без призначеного викладача, що дозволяє гнучко планувати навчальне навантаження, але кожен викладач обов'язково повинен вести хоча б один курс, що гарантує його залученість до освітнього процесу.

Для планування навчального процесу у часі використовується зв'язок "один до багатьох" між Course та Lesson. Кожен курс складається з серії занять, причому кожне конкретне заняття жорстко прив'язане до свого курсу. Система вимагає, щоб кожен курс мав хоча б одне заплановане заняття, що забезпечує мінімальну наповненість навчальної програми, водночас окремі заняття можуть існувати у системі тимчасово без явної прив'язки до курсу, що забезпечує гнучкість у плануванні.

Організація просторового планування реалізована через зв'язок "багато до одного" між Lesson та Room. Кожне заняття може проводитися у певній аудиторії, при цьому одна аудиторія може використовуватися для багатьох різних занять. Цей зв'язок дає зрозуміти, що заняття може тимчасово не мати призначеної аудиторії, але кожна аудиторія обов'язково повинна бути задіяна хоча б в одному занятті, що забезпечує ефективне використання навчальних

приміщень.

Усі описані зв'язки реалізовані з урахуванням бізнес-логіки навчального закладу та забезпечують цілісність даних за допомогою обмежених каскадних операцій. Система індексації ключових полів значно підвищує продуктивність запитів, що особливо важливо для великої кількості одночасних звернень до бази даних. Така структура дозволяє ефективно керувати всіма аспектами навчального процесу від розподілу студентів по групах і курсах до планування занять у конкретних аудиторіях.

Розроблена структуру бази даних зображена на рисунку 2.6.

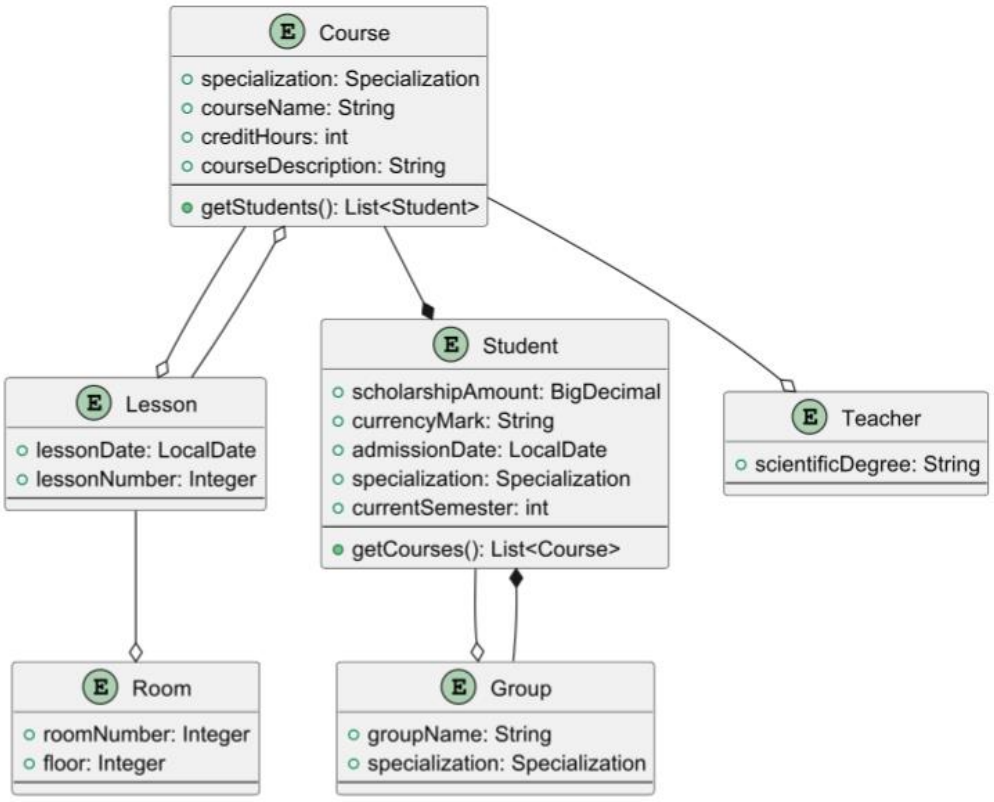


Рисунок 2.6 – Структура бази даних

На рисунку представлена структура бази даних, яка включає декілька таблиць, що взаємодіють між собою. Кожна таблиця має свої поля, які містять конкретні дані.

Таблиця «Course» призначена для зберігання інформації про навчальні курси, їхні характеристики та зв’язки зі студентами. Вона складається з таких

полів:

– «specialization» – спеціалізація курсу;

– «courseName» – назва курсу;

– «creditHours» – кількість кредитних годин;

– «courseDescription» – детальний опис курсу.

– «getStudents()» – повертає список студентів, які записані на курс.

Таблиця «Lesson» використовується для зберігання даних про проведені заняття, включаючи їхні часові параметри. Вона містить такі поля:

- «lessonDate» – дата проведення заняття;

- «lessonNumber» – порядковий номер заняття.

Таблиця «Room» містить інформацію про аудиторії, де проводяться заняття. Поля таблиці:

- «roomNumber» – номер аудиторії;

- «floor» – поверх, де розташована аудиторія.

Таблиця «Student» це центральна таблиця для зберігання даних про студентів, їхню академічну інформацію та фінансову підтримку. Включає такі поля:

- «scholarshipAmount» – розмір стипендії;

- «currencyMark» – валюта стипендії;

- «admissionDate» – дата вступу;

- «specialization» – спеціалізація студента;

- «currentSemester» – поточний семестр навчання.

- «getCourses()» – повертає список курсів, які відвідує студент.

Таблиця «Group» призначена для управління групами студентів та їхніми спеціалізаціями. Поля:

- «groupName» – назва групи;

- «specialization» – спеціалізація групи.

Таблиця «Teacher» зберігає інформацію про викладачів, зокрема їхні наукові ступені. Поле:

- «scientificDegree» – науковий ступінь.

Запропонована організація бази даних створює гнучку основу для майбутнього розвитку соціальної мережі та підтримки різних навчальних проєктів. Застосування реляційної моделі спрощує встановлення зв'язків між даними в таблицях, що є ключовим для забезпечення їхньої коректності. Взаємозв'язок таблиць здійснюється за допомогою унікальних ідентифікаторів, що гарантує цілісність і несуперечність інформації. Крім того, структура бази даних показає наявність індексів, які оптимізують швидкість обробки запитів.

У результаті, розробка бази даних з акцентом на нормалізацію, індексацію, налагодження зв'язків між таблицями та застосування каскадного видалення дає змогу досягти ефективного керування даними, їхньої цілісності та оперативного доступу. Подібні стратегії проектування є дуже важливими для забезпечення стабільної роботи вебзастосунку та його подальшого розвитку.

2.5 Висновки

У розділі було виконано систематичний аналіз основних аспектів створення програмного забезпечення. В першу чергу, була розроблена структура інтерфейсу програмного засобу, яка відповідає сучасним стандартам та забезпечує зручність взаємодії користувача з системою.

Було розглянуто методи забезпечення безпеки, що сприяють захисту конфіденційної інформації користувачів під час реєстрації та авторизації.

Описано модель функціонування системи, яка орієнтована на задоволення потреб користувачів, надаючи їм можливість створювати, редагувати та керування курсів, а також організацію навчання,. Одним із важливих аспектів цієї моделі є розробка структури бази даних, яка гарантує ефективне зберігання та обробку даних, забезпечуючи зручний доступ до необхідної інформації та оптимізуючи процеси вибірки і обробки даних у рамках вебсервісу.

3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного продукту

У процесі створення вебсервісу для організації та управління освітнім процесом в університеті особливо важливим є правильний вибір інструментів розробки. Від обраних технологій залежить стабільність, швидкодія та здатність системи адаптуватися до зростаючих вимог. Застосування мови Java у поєднанні з фреймворком Spring Boot дає змогу реалізувати надійну та масштабовану серверну частину, здатну обробляти велику кількість запитів і працювати з великими обсягами навчальної інформації. Такий технологічний стек добре підходить для побудови захищених і продуктивних вебрішень у сфері освіти. Використання мови Java [15] під час створення серверної частини вебсервісів має низку суттєвих переваг:

1. Перетворення Java-коду в байт-код для JVM дозволяє досягти високої ефективності та продуктивності, що є важливим фактором для серверних питань з великою кількістю одночасних запитів.

2. Канон «один раз написано – працює всюди» дозволяє запускати програмний код на будь-якому пристрої або операційній системі, що підтримує Java Virtual Machine. Саме він забезпечує маштабованість і зручність у розгортанні системи в різних середовищах.

3. Значна кількість готових бібліотек і фреймворків, зокрема Spring Framework ,Hibernate та JUnit, сприяє пришвидшенню розробки, тестування та спрощенню розробки нових функцій, що покращує загальну продуктивність команди.

4. Інтеграція з сучасними базами даних дає можливість працювати з популярними СУБД: PostgreSQL, MySQL, Oracle, що дає обирати сховище для освітніх даних, розкладів, заяв, журналів оцінювання залежно від потреб проєкту.

C# [16] – це об’єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона підтримує сучасні механізми, зокрема асинхронне виконання, автоматичне керування пам’яттю та роботу в межах платформи .NET. За допомогою фреймворку ASP.NET Core можна створювати стабільні, безпечні та масштабовані вебзастосунки. Завдяки середовищу Visual Studio забезпечується вдалу розробку, тестування та підтримку проєктів.

JavaScript [17] використовується не лише на стороні клієнта, але й у серверній частині завдяки платформі Node.js. Її неблокувальна архітектура введення-виведення дає змогу ефективно обробляти велику кількість одночасних запитів. Node.js підходить для створення вебсервісів і застосовується в мікросервісних системах, де важливі швидкодія і масштабованість.

Результати порівняння мов програмування для розробки серверної частини приведено в таблиці 3.1.

Таблиця 3.1 – Порівняльня характеристика мов програмування

Характеристика	Java	C#	JavaScript
Незалежність платформи	1	0	1
Підтримка багатопоточності	1	1	0
Продуктивність	1	1	0
Простота навчання	0	0	1
Активність спільноти та підтримка	1	1	1
Сумарний коефіцієнт	4	3	3

Аналізуючи оновлену таблицю порівняння, можна зробити висновок, що Java отримала найвищий загальний бал, демонструючи найбільшу відповідність вимогам до розробки серверної частини вебзастосунків. Вона вирізняється високою продуктивністю, багатопотоковою обробкою та активною підтримкою спільноти, а також забезпечує платформну незалежність, що дозволяє запускати

додатки на різних операційних системах без модифікацій.

C#, хоча і має потужні можливості у поєднанні з .NET, поступається Java у сфері кросплатформеності, що може бути обмеженням при розгортанні додатків на різних середовищах. Проте він є популярним у корпоративному сегменті завдяки хорошій інтеграції з екосистемою Microsoft і підтримці сучасних технологій.

JavaScript також має хороші показники у платформній незалежності та простоті масштабування, але вона поступається Java за показниками продуктивності та багатопоточності, що може стати важливим у випадках з високим навантаженням на серверну частину.

Отже, Java є найкращим вибором для реалізації серверної частини нашого вебсервісу завдяки поєднанню ефективності, стабільності, розширюваності та технологічної зрілості. Це робить її особливо придатною для проєктів, де важливі надійність, масштабованість і здатність обробляти значні обсяги запитів у реальному часі.

Наступним кроком є визначення фреймворку для розробки вебсервісіу мовою Java. Вибір ґрунтується на швидкому старті, простому управлінні залежностями та підтримці хмарного розгортання. Найбільш популярним залишається Spring Boot. Він облегшує налаштування та запуск додатків, забезпечує комунікацію з іншими модулями Spring [18] і підходить як для монолітних, так і для мікросервісних архітектур. Він характеризується високою надійністю, безпекою та масштабованістю.

Серед новіших варіантів варто відзначити Helidon [19], розроблений компанією Oracle. Він доступний у двох варіантах: один орієнтований на простий та реактивний підхід, а інший на повноцінну підтримку стандартів для мікросервісів. Helidon добре підходить для створення невеликих, швидких і ефективних сервісів, особливо у хмарному середовищі.

Quarkus також заслуговує на увагу. Завдяки покращені для GraalVM та HotSpot, він забезпечує низьке споживання пам'яті та швидкий час відгуку. Ці показники є вирішальними для розробки сучасних мікросервісних архітектур та

сервісних рішень.

Для порівняння фреймворків сформовано порівняльну таблицю 3.2.

Таблиця 3.2 – Порівняльна характеристика фреймворків

Характеристика	Spring Boot	Helidon	Quarkus
Споживання пам’яті	1	1	1
Гнучке налаштування	1	0	1
Швидкість запуску	1	1	0
Підтримка мікросервісів	1	1	1
Масштабованість	1	0	0
Сумарний коефіцієнт	5	3	3

Як показало порівняння, Spring Boot перемагає за багатьма показниками. Він споживає менше пам'яті, а його автоматизація конфігурації та коду за допомогою анотацій серйозно полегшує розробку. Окрім того, Spring Boot ідеально підходить для мікросервісів, які зараз просто незамінні для сучасних вебзастосунків. Тому, без сумніву, Spring Boot є найкращим та найоптимальнішим рішенням для ваших проєктів.

Одним із найкращих рішень є PostgreSQL – стабільна та функціональна СУБД, яка добре справляється зі складними транзакціями і аналітичними запитами, що необхідно для забезпечення якісного управління навчальним процесом.

Як альтернативу варто розглянути Oracle Database, яка широко використовується у великих навчальних закладах. Oracle пропонує високу продуктивність, і розвинені механізми безпеки, що робить її ефективним інструментом для роботи з великими обсягами даних і складними бізнес-процесами університету.

Нижче таблиця 3.3 представляє порівняльний аналіз PostgreSQL та Oracle Database, враховуючи їх застосування у Java-розробці вебзастосунків.

Таблиця 3.3 – Порівняльна характеристика СУБД

Характеристика	PostgreSQL	Oracle Database
Сумісність із Java	1	1
Підтримка складних SQL-запитів	1	1
Гнучкість ліцензії та можливості для розширення	1	0
Робота з JSON-форматом і NoSQL-подібними запитами	1	1
Сумарний коефіцієнт	4	3

У рамках розробки вебсервісу для організації освітнього процесу університету, PostgreSQL демонструє суттєві переваги, що відповідають специфіці завдань. Завдяки підтримці складних запитів і можливості ефективного масштабування, ця СУБД ідеально підходить для систем, які потребують гнучкого зростання обсягів даних.

Вибір цих технологій та інструментів є ключовим для створення надійного, безпечного та легко масштабованого програмного забезпечення, здатного підтримувати активну взаємодію користувачів та обмін освітніми ресурсами. Поєднання Java, Spring Boot та PostgreSQL забезпечує міцну платформу для розробки вебсервісів, що ефективно виконує освітні та комунікаційні завдання.

3.2 Розробка модуля реєстрації та автентифікації користувача

Модуль реєстрації та автентифікації користувача є основною частиною будь-якого вебсервісу, і вебсервіс для організації та управління освітнім процесом університету не є винятком. Цей модуль виконує важливу роботу, забезпечуючи створення нових облікових записів для різних категорій користувачів та контролюючи їхній доступ до функціональності та даних системи. Прийняття HTTP-запитів методом POST до /registration для реєстрації нових користувачів відбувається через SignupController, детальний алгоритм роботи методу addUser для цієї операції представлений на рисунках 3.1 та 3.2.

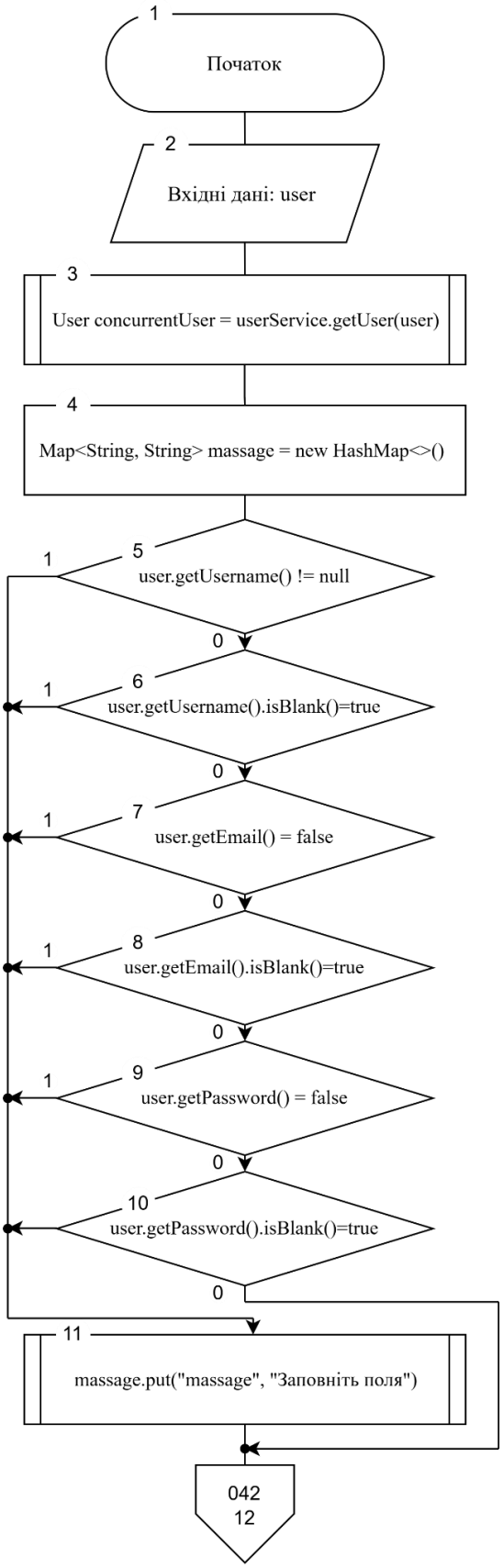


Рисунок 3.1 – Алгоритм методу реєстрація користувача addUser (частина 1)

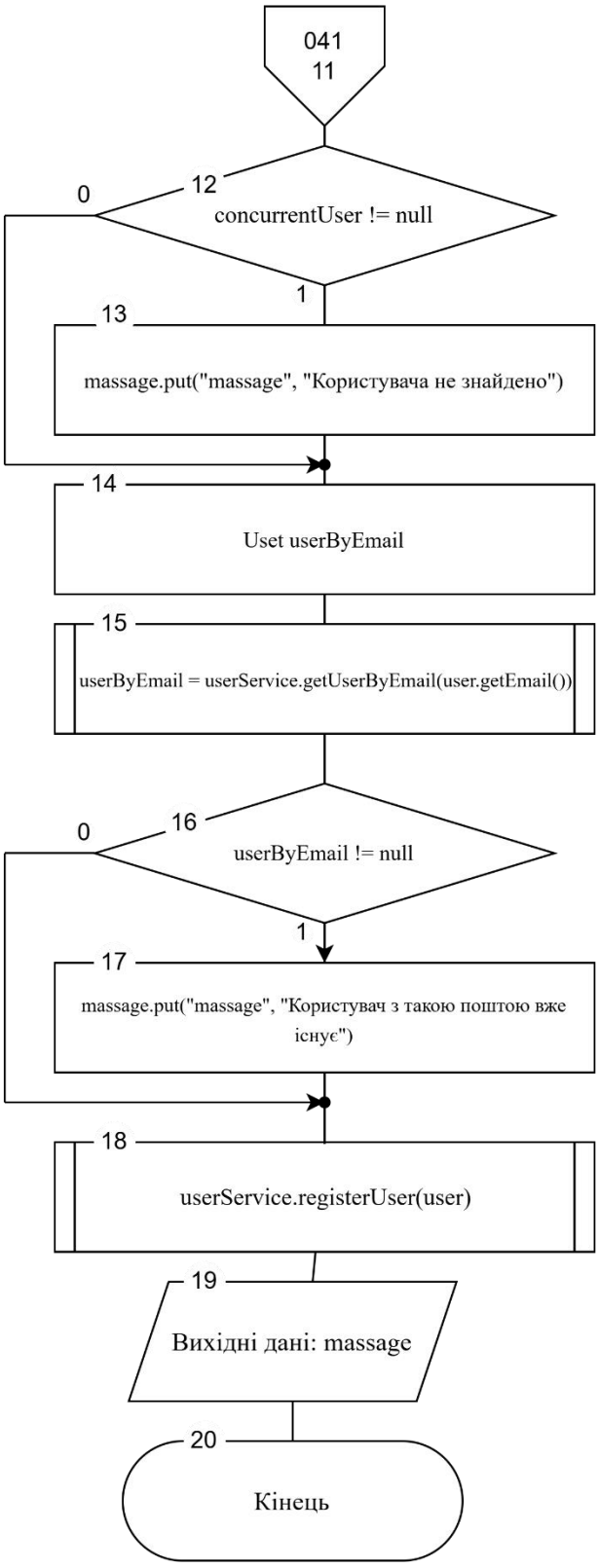


Рисунок 3.2 – Алгоритм методу реєстрація користувача addUser (частина 2)

Контролер SignupController обробляє запити, пов’язані з процесом

створення нового облікового запису. Його функціональність включає:

Метод registration(), який реагує на GET-запити за адресою /registration і відповідає за відображення сторінки реєстрації.

Метод addUser(User user) приймає POST-запити на той самий шлях. Перед тим як створити нового користувача, він перевіряє, чи не зареєстровано вже обліковий запис із такими ж даними. Якщо користувач вже присутній у базі, повертається повідомлення про наявність облікового запису.

Після цього метод перевіряє, чи не використовується вказана електронна адреса іншим користувачем. У разі, якщо email вже закріплений за іншим обліковим записом, система повідомляє про те, що ця адреса вже зайнята.

Якщо трапляються помилки під час цих дій у Spring-застосунках використовується механізм глобального оброблення винятків за допомогою анотації @ControllerAdvice. Це дозволяє обробляти виняткові ситуації в одному місці, незалежно від того, де саме вони виникли у вебзастосунку.

Клас GlobalExceptionHandler слугує основною точкою перехоплення винятків і містить методи для реагування на різні типи помилок. Один з таких методів є handleRedirectWithModelException, який приймає виняток типу RedirectWithModelException, об’єкт Model для передачі атрибутів у представлення та HttpServletRequest.

Клас RedirectWithModelException є винятком, який створений для ситуацій, коли потрібно здійснити перенаправлення на іншу сторінку з одночасною передачею параметрів у модель. Він містить поля для збереження імені та значення переданого атрибута, а також відповідні гетери і конструктор.

Коли виникає RedirectWithModelException, метод у GlobalExceptionHandler додає потрібний атрибут до моделі й перенаправляє користувача, наприклад, на сторінку /login.

Таким чином, система реєстрації та входу користувача забезпечує належний контроль доступу, обробку винятків та зворотній зв’язок для користувачів, що підвищує безпеку та зручність використання вебсервісів для навчання.

3.3 Розробка модуля пошуку студента за логіном

Пошук користувачів за логіном є однією з ключових функцій нашого вебсервісу для організації освітнього процесу в університеті. Вона дозволяє швидко знаходити потрібного користувача, що особливо зручно для викладачів і студентів під час взаємодії в системі. Щоб усе працювало коректно та швидко, при реалізації ми врахували і зручність для користувача, і технічні нюанси взаємодії з базою даних.

Алгоритм роботи методу пошуку студента за логіном зображено на рисунку 3.8

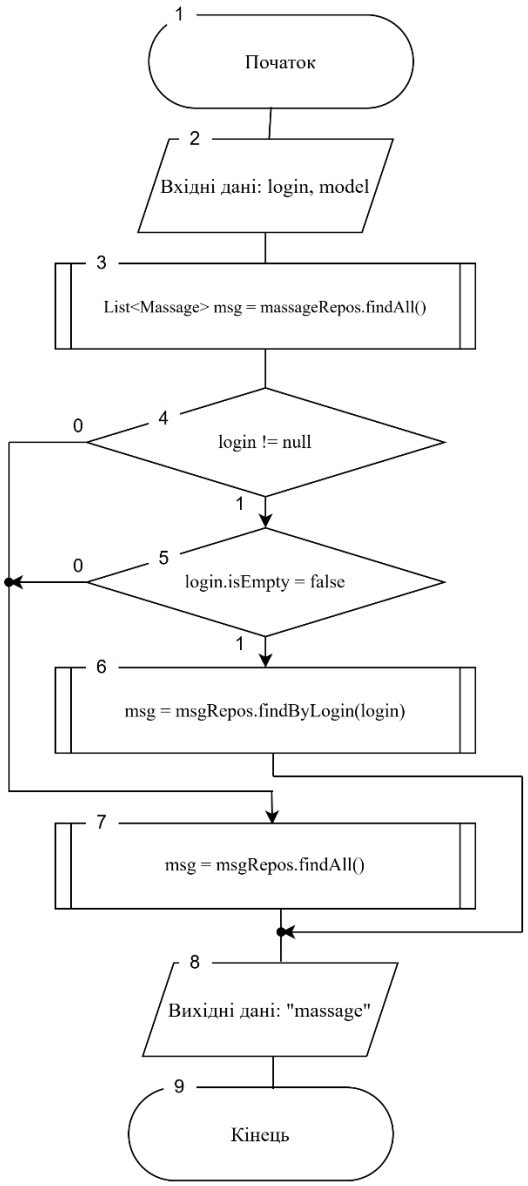


Рисунок 3.3 – Алгоритм роботи методу пошуку студента за логіном

Контролер «MessageController» відповідає за обробку запитів, пов’язаних з переглядом повідомлень на платформі. Його метод обробки запитів на головну сторінку приймає параметр «login», який дозволяє здійснювати фільтрацію повідомлень за іменем користувача. Це дає змогу відображати лише ті повідомлення, які були залишені конкретним користувачем, що підвищує релевантність контенту для кожного відвідувача.

Спочатку метод виконує отримання всіх повідомлень із бази даних за допомогою репозиторію «massageRepos». Але спочатку здійснюється перевірка: якщо значення «login» не є порожнім і не дорівнює null, викликається метод findByLogin, який витягує лише ті повідомлення, що відповідають вказаному імені користувача. У випадку, якщо параметр login відсутній або порожній, система повторно використовує метод findAll для отримання всіх повідомлень без фільтрації.

MessageRepos виступає інтерфейсом доступу до бази даних і визначає методи findAll() та findByLogin(String login), що забезпечують ефективну взаємодію з таблицею повідомлень. Такий підхід дозволяє зменшити обсяг оброблюваних даних і покращити продуктивність запитів при наявності великої кількості записів.

На клієнтській стороні для відображення повідомлень використовується шаблонізатор Thymeleaf, який динамічно формує вміст сторінки. Цикл th:each, перебирає список повідомлень «messages» та відображає кожне з них у зручному для користувача форматі. Це допомагає створенню зрозумілого й наочного інтерфейсу, що дозволяє легко орієнтуватися у поданій інформації.

Завдяки розробці пошуку за логіном, користувачі можуть швидко знаходити інформацію про конкретних студентів. Це значно спрощує навігацію в системі, дозволяє зосередитися на потрібних даних і сприяє ефективнішій взаємодії між учасниками освітнього процесу. Такий функціонал демонструє, як сучасні технології можуть покращити організацію та доступність інформації у навчальному середовищі.

3.4 Розробка інтерфейсу користувача

Щоб спілкуватись з сервером, нам потрібно розробити користувацький зрозумілий інтерфейс.Оскільки нам потрібно спілкуватись з сервером, нам потрібно розробити користувацький зрозумілий інтерфейс [20]. В нашому випадку будемо використовувати JavaScript та CSS для написання інтерфейсу програми.

JavaScript – це мова програмування, яка дозволяє вам реалізувати складні функції на вебсторінках, щоразу, коли вебсторінка робить щось більше, ніж просто сидить і відображає статичну інформацію, яку ви можете подивитися, відображаючи своєчасні оновлення вмісту.

CSS [21] це спеціальна мова, яка використовується для стилізації веб-додатків, розроблених мовою веброзмітки, яка може працювати незалежно від мови програмування, що використовується на стороні сервера. Разом вони дозволяють створювати адаптивні інтерфейси, які підлаштовуються під різні розміри екранів та пристроїв.

Вперше переходячи за посиланням вебсервісу, будь-який користувач є гостем, адже не аутентифікувався та авторизувався. Його зустрічає сторінка аутентифікації, зображення якої проілюстрований на рисунку 3.4.

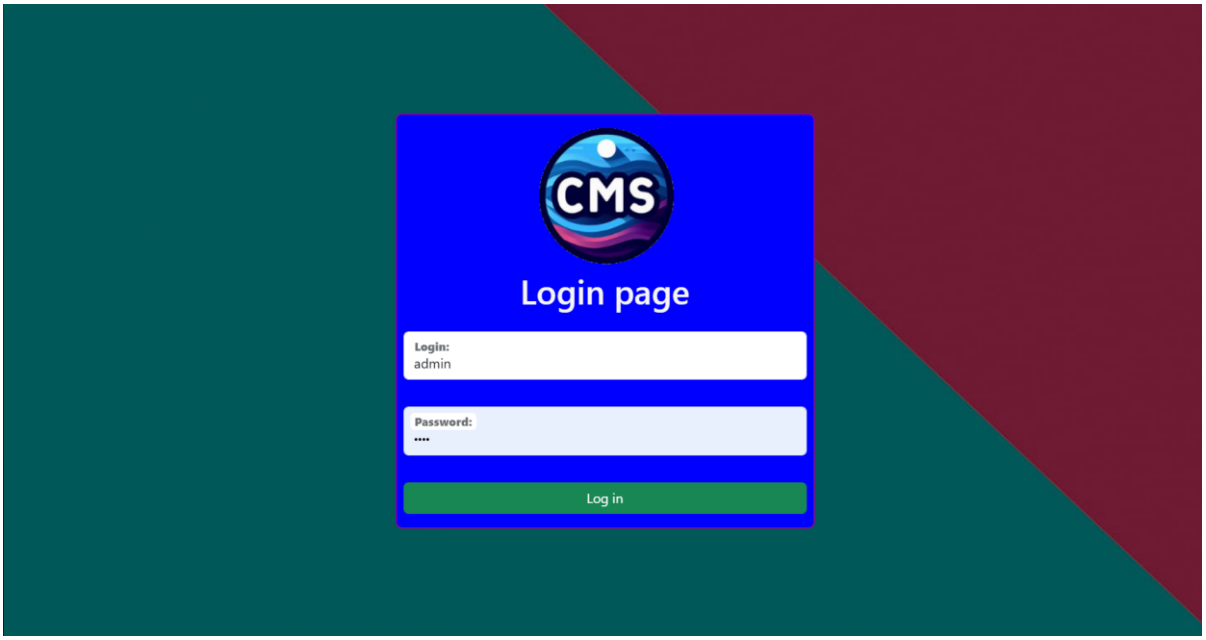


Рисунок 3.4 – Зображення сторінки аутентифікації

Щоб отримати доступ до можливостей, які дає вебсервіс користувач повинен увійти в нього, ввівши коректні облікові дані, тим самим підтвердивши свою належність до всієї систем та роль у ній, шляхом натиснення відповідної кнопки, після чого він буде переспрямований на головну сторінку, частина якої зображена на рисунку 3.5. Залежно від того, хто користувач – адміністратор, викладач чи студент, йому будуть відкриті спільні або відмінні функції.

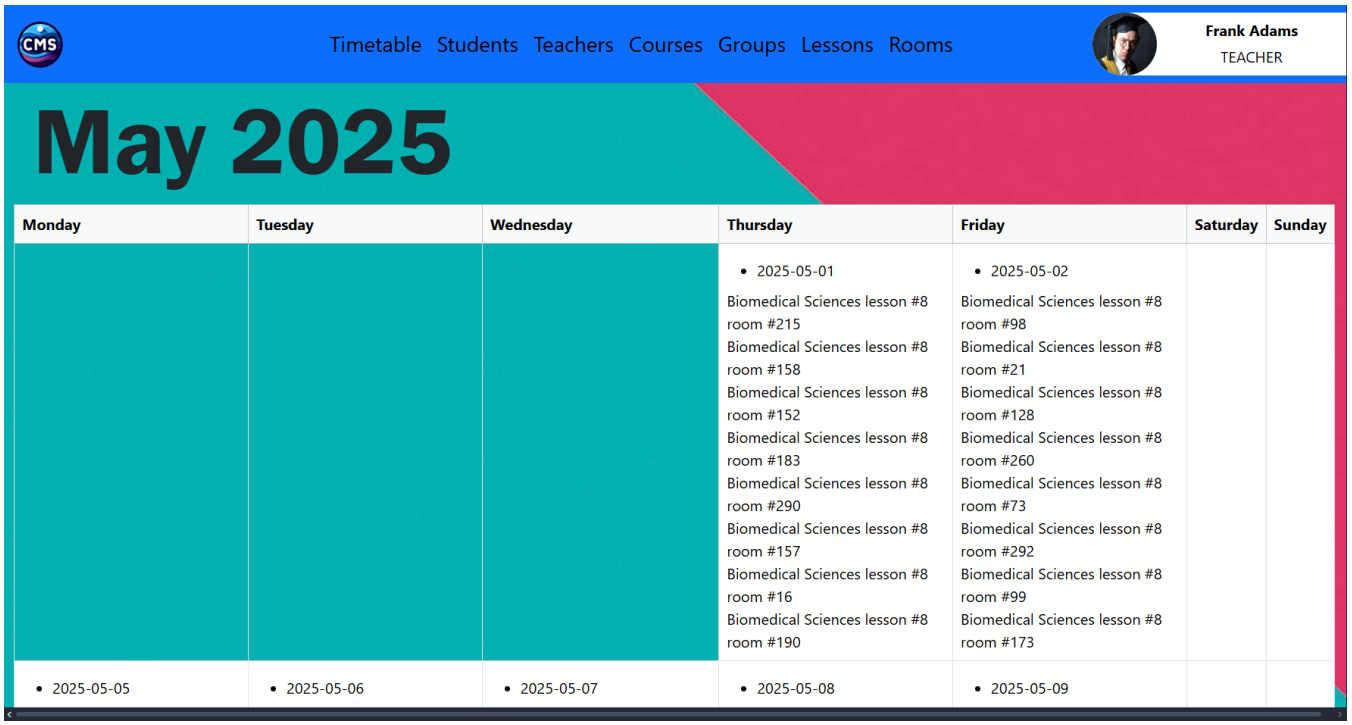
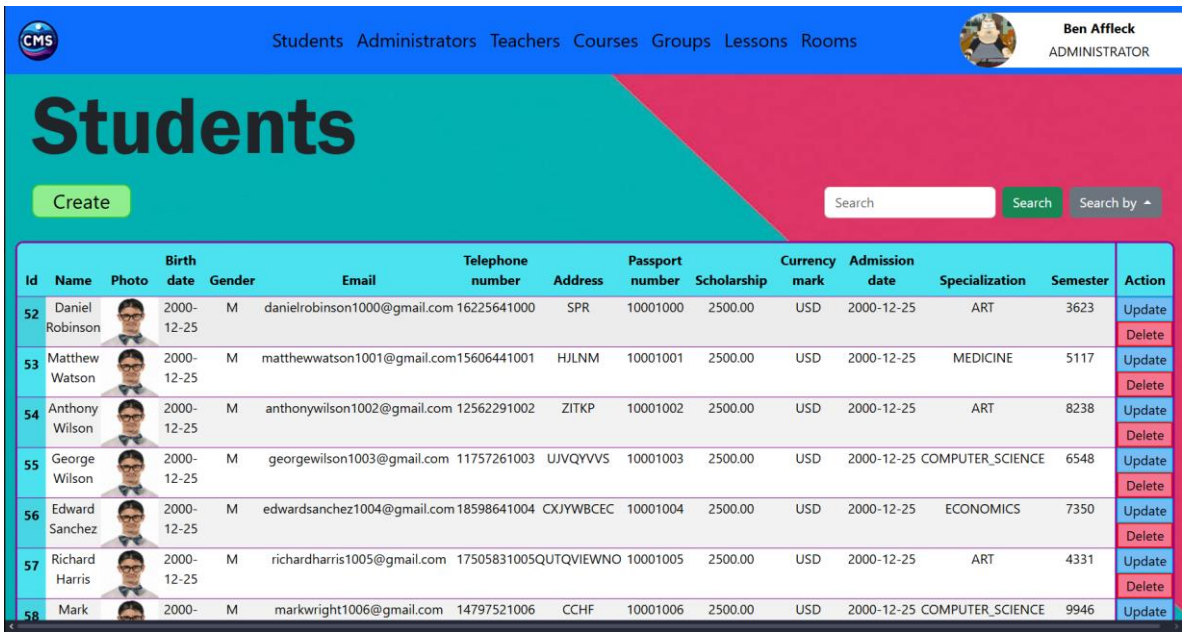


Рисунок 3.5 – Зображення головної сторінки

Ця вебсторінка є головною сторінкою системи управління освітнім процесом. Вона містить основне меню з посиланнями на інші розділи сайту, такі як «Студенти», «Адміністрація», «Викладачі», «Курси», «Предмети» та «Кімнати». На головній сторінці розміщений розклад занять та кнопка профілю.

Сторінка студентів (див. рисунок 3.6) – це сторінка, що дає користувачеві усю необхідну та актуальну інформацію про студентів, які містяться в базі даних.. У правій частині сторінки розташовано меню для пошуку, фільтрації та сортуванню студентів по деяких кретеріях. Також є кнопка "Створити", яка дозволяє створити студента з всіма даними, якщо є на це права.



Id	Name	Photo	Birth date	Gender	Email	Telephone number	Address	Passport number	Scholarship	Currency mark	Admission date	Specialization	Semester	Action
52	Daniel Robinson		2000-12-25	M	danielrobinson1000@gmail.com	16225641000	SPR	10001000	2500.00	USD	2000-12-25	ART	3623	Update Delete
53	Matthew Watson		2000-12-25	M	matthewwatson1001@gmail.com	15606441001	HJLNM	10001001	2500.00	USD	2000-12-25	MEDICINE	5117	Update Delete
54	Anthony Wilson		2000-12-25	M	anthonywilson1002@gmail.com	12562291002	ZITKP	10001002	2500.00	USD	2000-12-25	ART	8238	Update Delete
55	George Wilson		2000-12-25	M	georgewilson1003@gmail.com	11757261003	UJVQYVVS	10001003	2500.00	USD	2000-12-25	COMPUTER_SCIENCE	6548	Update Delete
56	Edward Sanchez		2000-12-25	M	edwardsanchez1004@gmail.com	18598641004	CXJYWBCEC	10001004	2500.00	USD	2000-12-25	ECONOMICS	7350	Update Delete
57	Richard Harris		2000-12-25	M	richardharris1005@gmail.com	17505831005	QUTQVIEWNO	10001005	2500.00	USD	2000-12-25	ART	4331	Update Delete
58	Mark		2000-	M	markwright1006@gmail.com	14797521006	CCHF	10001006	2500.00	USD	2000-12-25	COMPUTER_SCIENCE	9946	Update

Рисунок 3.6 – Зображення сторінки «Студенти»

Також є кнопка "Створити", яка дозволяє створити студента з всіма даними, якщо є на це права. Натишнувши на неї і скориставшися відповідними полямя можна створити студента з такою інформацію (див. рисунок 3.7) : «ПІБ», «Стать», «Дата народження», «Поштова скринька», «Номер телефону», «Адресса», «Дата початку навчання», «Поточний семестр», «Курс», «Номер телефону» та «Фото користувача».

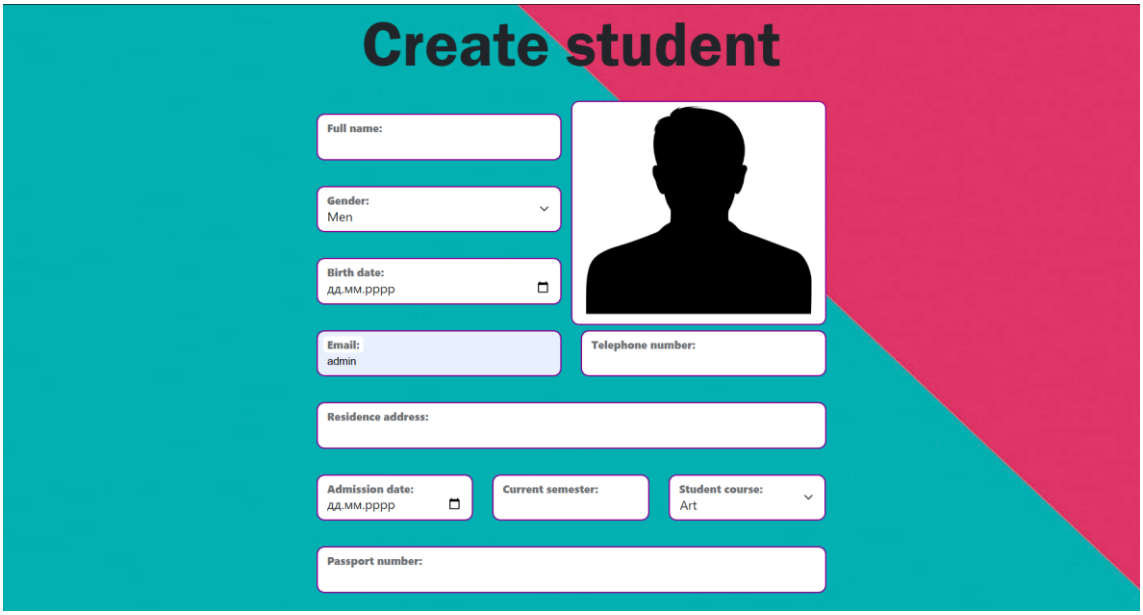


Рисунок 3.7 – Сторінка створення студента

Перейшовши на сторінку «Курси», користувач бачить список курсів та спеціальностей, та може легко керувати ними задопомогою інтерфейсу. Вона також містить інструменти пошуку, фільтрації, сортування за колонками рядків та пагінації сторінок даних, частину чого видно на рисунку 3.8. Ще розроблена можливість видалення студентів з курсу, наприклад, у разі переведення студента на іншу спеціальність або завершення ним певного етапу навчання.

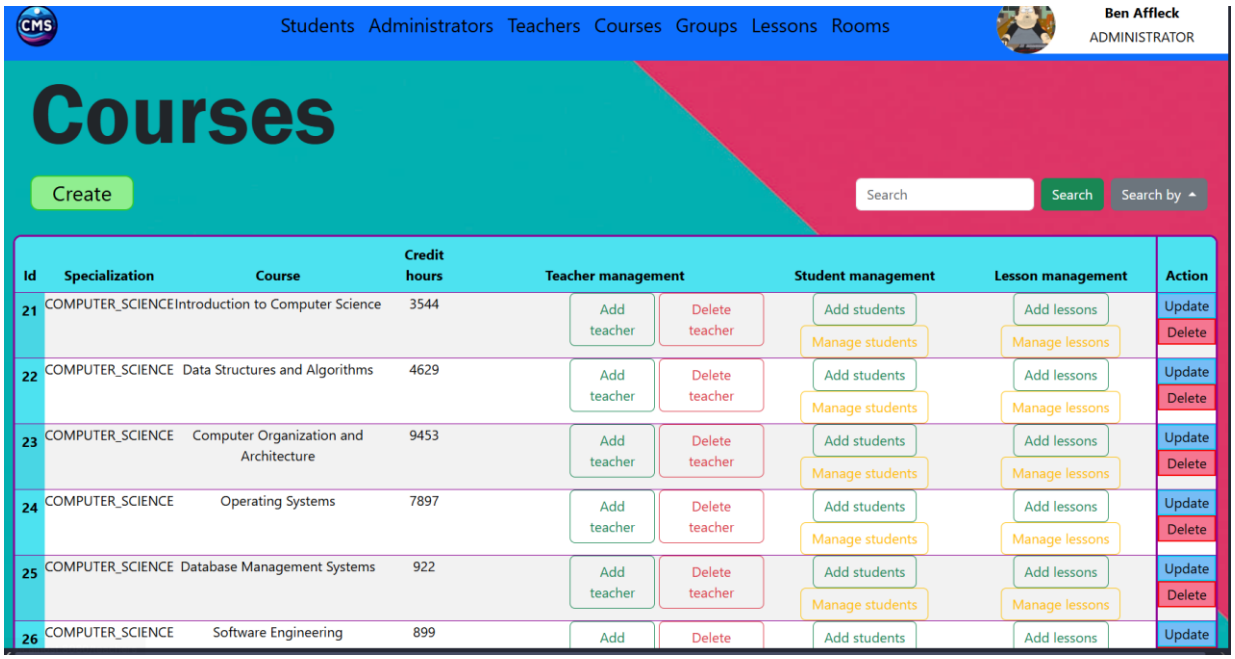


Рисунок 3.8 – Зображення сторінки «Курси»

Отже, у підрозділі роботи, було обґрунтовано розробку важливість розробки інтерфейсу користувача для вебсервісу для організації та управління освітнім процесом університету та детально описано розроблюванні сторінки для системи.

3.5 Висновки

У цьому розділі описано процес створення основних програмних модулів вебсервісу для організації та управління освітнім процесом університету. Вибір таких технологій, як Java, Spring Boot та PostgreSQL, зумовлений їх перевагами у швидкості розробки, забезпеченні безпеки та надійній роботі з даними.

Застосування цих інструментів дало змогу реалізувати стабільну та захищену платформу, здатну витримувати велике навантаження та ефективно працювати з освітнім контентом.

Розроблені модулі включають функціонал для реєстрації та автентифікації користувачів, пошуку студентів за логіном. Ці модулі роблять систему безпечною, простою у використані та більш зрозумілішою, що так подобається студентам. Ще розроблено інтерфейс користувача та описано основні сторінки такі як: головний екран, сторінка студентів та екран курсів. Таким чином, розроблені програмні засоби сприяють створенню інтерактивного та безпечного середовища для освітніх ініціатив.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

На сьогоднішній день дуже багато різних типів тестування програмного забезпечення, додатків, а також сайтів. Дуже важливо вибрати якісний інструмент або план тестування, для того щоб заощадити час і гроші.

Тестування програмного забезпечення [22] – це метод перевірки того, чи підходить програмний продукт потрібним вимогам, і переконатися, що програмний продукт не містить дефектів. Він включає виконання програмних та системних компонентів з використанням ручних або автоматизованих інструментів для оцінки декількох або одного потрібних властивостей. Завданням тестування забезпечення є знаходженння помилок, недоліків або відсутніх вимог порівняно з фактичними вимогами.

Як функціональне і нефункціональне тестування засноване на понятті тестування чорного ящика. Це тестування не дбає про код вашого додатку або вебсервісу. Процес функціонального тестування полягає в тестуванні кожної функції вашої системи. Функціональне тестування для кожної функції пропонує відповідні вхідні дані. Перевірка починається з правильного результату або виведення. Порівняння отриманих результатів є найважливішим кроком, щоб зрозуміти і знайти можливі помилки в тестуванні. Функціональне тестування гарантує вам що ваш сайт або вебсервер буде поводитися очікуваним чином. Воно перевіряє на наявність багів і інших багів, щоб забезпечити максимальну продуктивність.

Функціональне веб-тестування відрізняється від інших форм тестування в цілому. Воно концентрується на тестуванні основних аспектів програми. Дуже важливо перед початком процесу мати список вимог до програмного продукту.

Автоматичне тестування ґрунтується на вже раніше підготовлених тестах, які запускаються автоматично і після порівнює актуальний результат з очікуваним. У багатьох випадках, термін «тестування» вводить в оману, тому що тести автоматизації насправді просто перевіряють, чи працює ваш веб-сайт

належним чином, для ряду заздалегідь певних сценаріїв. Навпаки, ручне тестування проводиться людиною, що знаходиться перед пк і ретельно виконує кроки тесту. Програмне забезпечення для автоматизації тестування також може вводити тестові дані в тестувальну систему, порівнювати очікувані і фактичні результати і створювати докладні звіти про тести.

З вище наведених типів тестування у цьому розділі було проведене функціональне тестування, що полягає у ручній перевірці зазначених вимог та ознак у порівнянні з кінцевим результатом.

4.2 Тестування основного функціоналу

Тестування вебсервісу для організації та управління освітнім процесом в університеті є важливою частиною розробки, який забезпечує виявлення та усунення помилок перед запуском платформи.

На прикладі реєстраційної сторінки можна перевірити, як інтерфейс реагує на некоректне введення даних. Якщо користувач заповнює поля невірно, система має повідомити про помилку (див. рисунок 4.1).

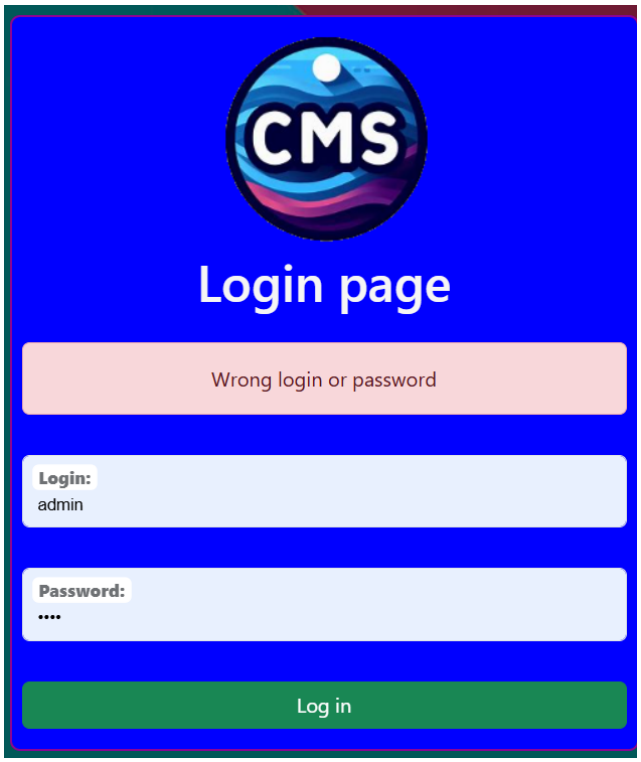


Рисунок 4.1 – Повідомлення про помилку «Не всі поля заповненні»

Другим було протестовано можливість створення нового студента з всіма відповідними даними (див. рисунок 4.2). Під час тестування також була перевірена система валідації, яка не дозволила створити студента при введенні некоректних або неповних даних, забезпечивши тим самим неможливість створення некоректного студента. Це підтвердило, що модуль створення студента працює стабільно та відповідає всім вимогам, поставленим перед ним.

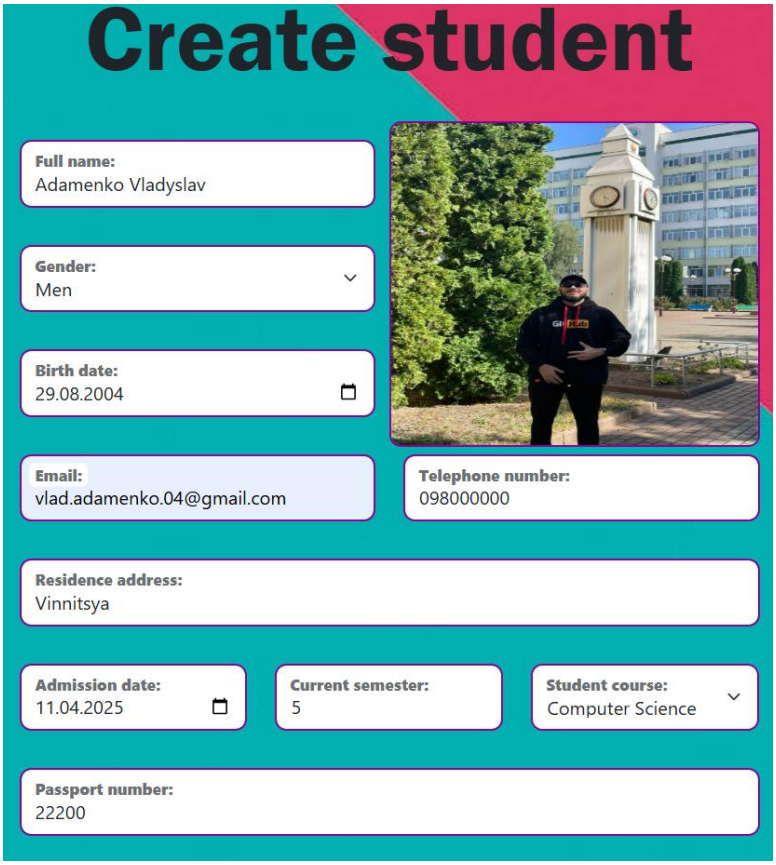


Рисунок 4.2 – Вікно створення студента

Після успішного додавання студента до системи, його запис відображається у загальному списку студентів. У цьому списку міститься основна інформація про кожного студента, включаючи ПІБ, спеціальність, семестр навчання та контактні дані. За потреби користувач може натиснути на певного студента, щоб переглянути детальніші відомості, а також редагувати чи видалити його запис.

На рисунку 4.3 зображена інформація про тільки що створеного студента, в загальному списку. У цьому списку користувач може легко знайти необхідного студента за допомогою пошуку або фільтрації за допомогою імені, номеру телефону, поштової скриньки і спеціальності.

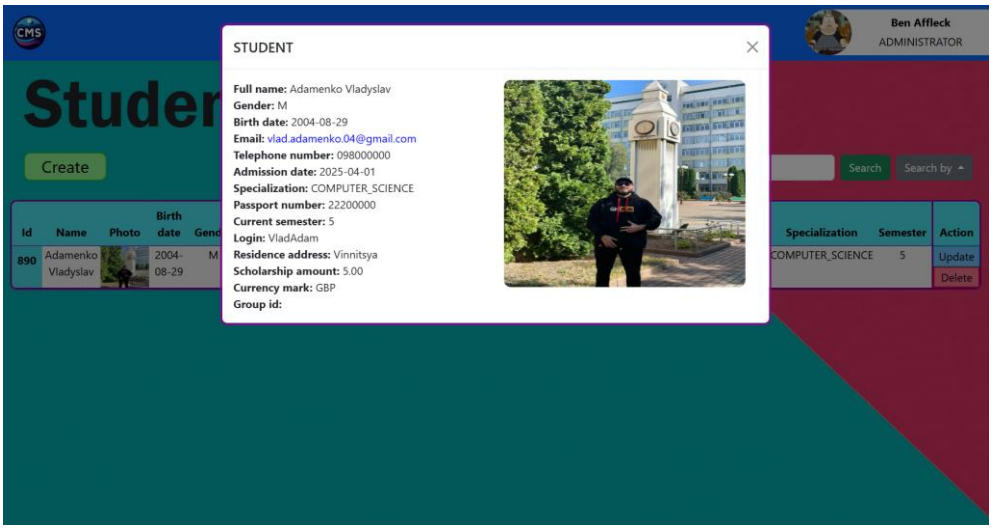


Рисунок 4.3 – Результат створення студента

Далі протестуємо можливість пошуку викладача за його ім’ям. Пошук робиться за допомогою розбиття ПІБ, що й продемонстровано на результаті. Скріншот тестування представлений на рисунку 4.4.

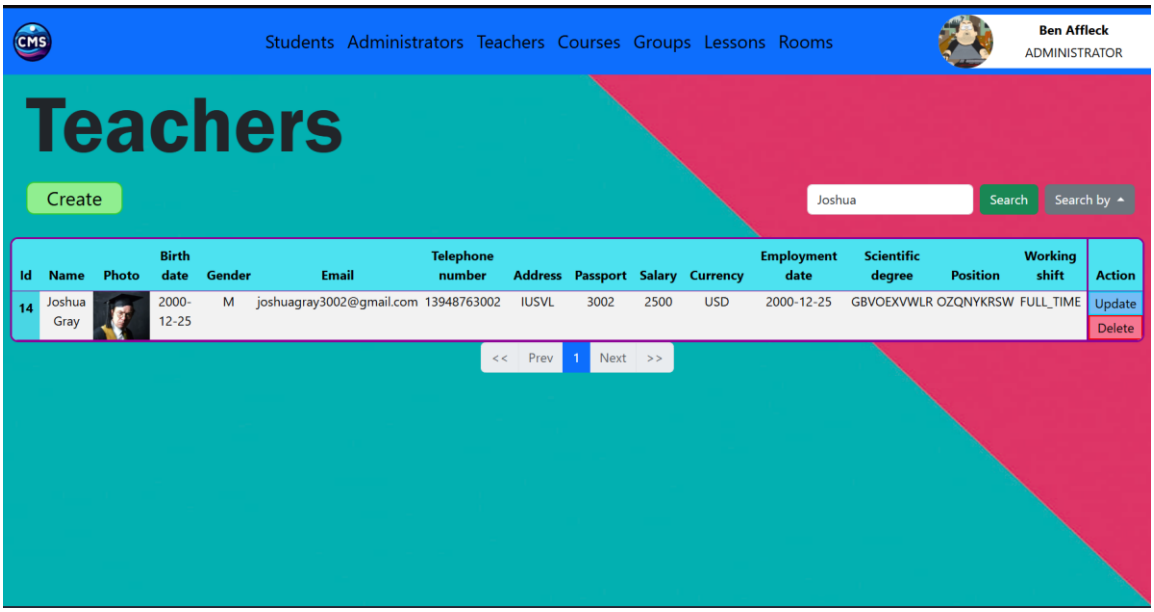


Рисунок 4.4 – Результат пошуку викладача

Сторінка розкладу (див. рисунок 4.5) , ретельно протестована для перевірки точності та надійності відображення дати проведення, класу та назви курсу. Під час тестування звернуто увагу на точність розкладу, який обновлюється динамічно після внесення змін в нього. Розклад занять дає можливість користувачам легко переглядати свої заняття, та активності на вебсервісі. Тестування підтвердило, що розклад коректно відображають усі необхідні дані, зокрема кількість занять, клас, та назву курс. Інтерфейс розкладу забезпечує інтуїтивно зрозумілу навігацію, що дозволяє користувачам без зайвих зусиль зрозуміти де, коли і з чого відбудеться заняття.

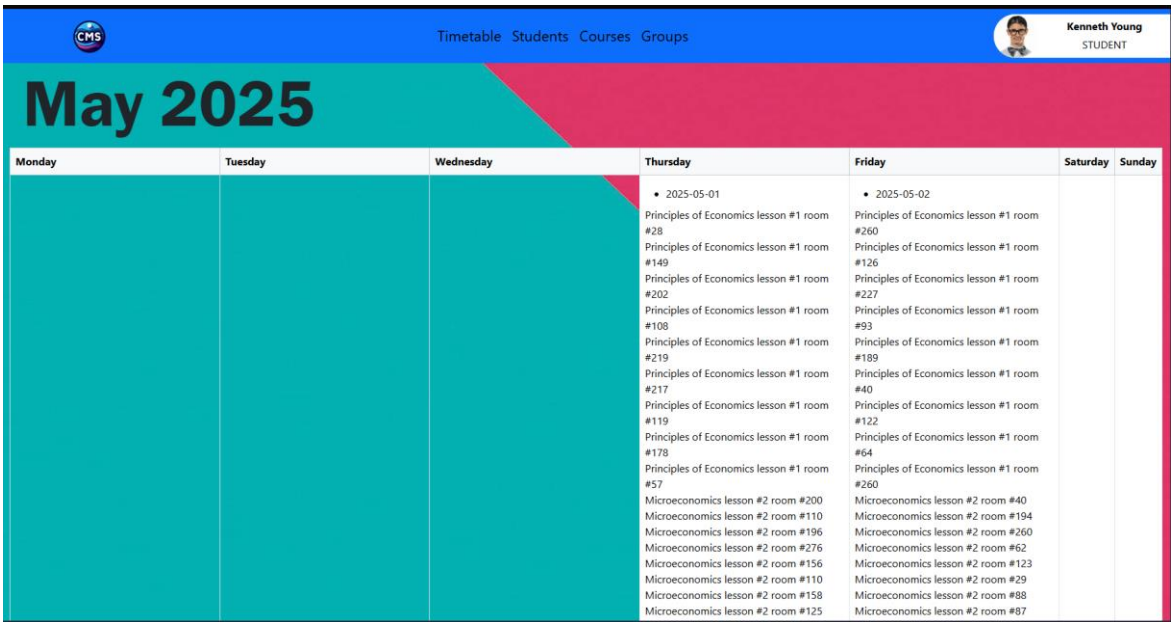


Рисунок 4.5 – Інтерфейс сторінки розкладу

Загальне тестування інтерфейсу вебсервісу для організації та управління освітнім процесом в університеті показало, що вибір кольорової палітри забезпечує максимальну читабельність контенту та зручність навігації. Кольори ефективно використовуються для виділення важливих елементів і створення контрасту.

Оцінювання взаємодії користувачів з окремими елементами інтерфейсу дало змогу виявити певні недоліки та внести необхідні зміни для покращення загального досвіду користування. У підсумку інтерфейс став більш зручним і

дозволяє користувачам зосередитися на ключових можливостях та контенті вебсервісу.

4.3 Розробка інструкції користувача

Інструкція користувача містить інформацію про технічні вимоги для запуску та коректної роботи вебсервісу для організації та управління освітнім процесом університету. Деталі щодо мінімальної та рекомендованої конфігурації користувацького пристрою представлені в таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Вимоги до конфігурування	Рекомендовано	Мінімально
Процесор	Intel Core i7 / AMD Ryzen 7	Intel Core i3 / AMD Ryzen 3
Оперативна пам'ять	16 ГБ RAM	4 ГБ RAM
Вільний простір на диску	20 ГБ	5 ГБ
Браузер	Google Chrome, Mozilla Firefox (остання версія)	Google Chrome (версія не нижче 90.0)
Операційна система	Windows 10 / macOS/ Linux	Windows 7 / macOS Mojave
Доступ до мережі	Швидкісний інтернет (не менше 50 Мбіт/с)	Інтернет-з'єднання (не менше 10 Мбіт/с)

Після встановлення та налаштування вебсервісу користувач відкриває веббраузер і після успішної авторизації переходить на головну сторінку системи управління освітнім процесом університету де доступні наступні основні функції:

– управління студентами, додавання, редагування, перегляд інформації про студентів, а також їх академічної успішності;

– управління викладачами: перегляд списку викладачів, розклад занять та їх контактні дані;

– розклад занять: перегляд розкладу занять для студентів та викладачів у зручному форматі;

– управління дисциплінами, додавання, редагування та видалення навчальних дисциплін;

– створення та управління групами, можливість формування навчальних груп, закріплення викладачів за дисциплінами.

4.4 Висновки

У четвертому розділі було проведено детальне тестування всіх програмних модулів системи, включаючи функціональне тестування користувацьких інтерфейсів, перевірку взаємодії з базою даних. Певну увагу було зосереджено на перевірці правильності введених даних, що сприяє стабільній роботі системи та знижує ризик помилок з боку користувачів. Проведене тестування підтвердило, що повідомлення про помилки та підказки щодо їх виправлення є зрозумілими й ефективними, що позитивно впливає на зручність використання та зменшує кількість некоректних введень.

Було також створено детальну інструкцію користувача, яка включає технічні вимоги для запуску вебсервісу. Цей документ допомагає швидко ознайомитися з основними можливостями платформи та спростити роботу з її інтерфейсом. Завдяки зрозумілому опису функцій, користувачі без труднощів можуть почати роботу із системою, що робить її доступною та зручною у використанні. У результаті вебсервіс показав готовність до впровадження та відповідає ключовим очікуванням користувачів.

ВИСНОВКИ

У бакалаврській кваліфікаційні роботі було розроблено вебсервіс для організації та управління освітнім процесом в університеті. Спочатку було здійснено аналіз сучасного стану вебсервісу для організації та управління освітнім процесом в університеті , що дозволило виділити основні проблеми та недоліки існуючих платформ. Цей аналіз став основою для постановки завдань проєкту. Бакалаврську кваліфікаційну роботу було виконано відповідно до рекомендацій, наведених у документі [23].

У ході роботи було проаналізовано та порівняно існуючі платформи, такі як Moodle, Google Classroom, Blackboard Learn, Canvas LMS та Open edX, що дозволило виявити їх переваги і недоліки та обґрунтувати актуальність створення нової платформи. Цей вебсервіс було створено за допомогою мови програмування Java, фреймворку Spring та СУБД PostgreSQL.

Під час розробки програмного продукту було створено зручний графічний інтерфейс програми. Реалізовано функціонал реєстрації та авторизації користувачів, забезпечено можливість створення нового студента, керування динамічним розкладом. Також було розроблено модуль для пошуку студента за логіном та модуль керування курсами.

Проведено тестування програмних модулів системи, включаючи валідацію полів введення інформації, результати якого підтвердили повну функціональність програми та відповідність технічному завданню. Крім того, було розроблено інструкцію користувача для встановлення та початкової експлуатації програмного продукту, а також підготовлено UML-діаграми для візуалізації моделі системи та алгоритму функціонування.

Таким чином, виконана бакалаврська кваліфікаційна робота підтвердила актуальність та необхідність створення спеціалізованого вебсервісу для організації та управління освітнім процесом в університеті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Актуальність розробки програмних модулів вебсервісу для управління освітнім процесом університету. [Електронний ресурс] – Режим доступу до ресурсу: https://uu.edu.ua/IC_dlya_efect_upravlinnya_ZO.

2. Немченко С. Г., Крижко В. В., Бондар О. С., Радул В. В., Старокожко О. М., Кондратенко Ю. І. Управління закладом освіти: підручник для здобувачів другого рівня вищої освіти педагогічних університетів. Бердянськ: БДПУ, 2022. 298 с.

3. Адаменко В.О. Веб-сервіс для організації та управління освітнім процесом університету. / XII Всеукраїнська науково-практична конференція молодих учених, Київ, 2025. [Електронний ресурс] – Режим доступу до ресурсу: https://zcit.kubg.edu.ua/index.php/journal/issue/view/13/23/3.

4. Адаменко В.О. Порівняльний аналіз Java-фреймворків для розробки вебсервісів. / Матеріали міжнародної науково-практичної інтернет конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)», Вінниця, 2025. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25100/20723.

5. S. K. Halder, S. Roy, S. Saha, and S. Sarkar, "An Augmented Reality-Based Educational Game for Learning," in Proceedings of the 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India – 2016. – 145 с.

6. Integration of web services in university education [Електронний ресурс] – Режим доступу до ресурсу: Integration Services : TechWeb : Boston University.

7. Moodle [Електронний ресурс] – Режим доступу до ресурсу: https://moodle.org.

8. Google Classroom [Електронний ресурс] – Режим доступу до ресурсу: https://sites.google.com/view/classrooms-workspace/home.

9. Blackboard Learn [Електронний ресурс] – Режим доступу до ресурсу: https://bme-o.blackboard.com.

10. Canvas LMS [Електронний ресурс] – Режим доступу до ресурсу: https://www.instructure.com/canvas.

11. Open edX [Електронний ресурс] – Режим доступу до ресурсу: https://openedx.org.

12. Сахай А., Граупнер С. Вебсервіси в підприємстві: концепції, стандарти, рішення та управління. Нью-Йорк: Springer, 2005. 312 с.

13. What is BCrypt? [Електронний ресурс] – Режим доступу до ресурсу: https://nordvpn.com/uk/blog/what-is-bcrypt/.

14. Романюк О. Н. Організація баз даних і знань / О. Н. Романюк, Т. О. Савчук. – Вінниця: УНІВЕРСУМ-Вінниця, 2003. – 105 с.

15. Joshua B. Effective Java Programming Language Guide. Boston: Addison-Wesley Professional, 2018. 412p.

16. C# [Електронний ресурс] – Режим доступу до ресурсу: Документация по .NET | Microsoft Learn.

17. Al Sweigart. The Recursive Book of Recursion: Ace the Coding Interview with Python and JavaScript. No Starch Press, 2022. 328p.

18. Spring Framework [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io.

19. Helidon [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/46099/.

20. User Interface Design – Methods and Qualities of a Good User Interface Design. Ravi Chandra Chaitanya Guntupalli, 2008. – 12с.

21. Daker J. HTML and CSS: Design and Build Websites First Edition. Wrox, 2011. 490p.

22. Myers G. The Art of Software Testing / G. Myers, C. Sandler, T. Badgett. – Canada: JohnWiley & Sons, 2012. – 240с.

23. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад.: О. Н. Романюк, Г. О. Черноволик. – Вінниця : ВНТУ, 2022. – 37с.

ДОДАТКИ

62

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
«21» березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу «: «Вебсервіс для організації та управління освітнім процесом в університеті»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
доц. каф. ПЗ Черноволик Г.О.
«24» 03 2025р.

Виконав:
студент гр. 4ПІ-21Б Адаменко В.О.
«21» 03 2025р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційа робота: «Вебсервіс для організації та управління освітнім процесом в університеті».

Галузь застосування – навчальні інтернет ресурси.

2. Підстава для розробки

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від «20» березня 2025 р. ректора ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки.

Метою дослідження є підвищення ефективності організації та управління освітнім процесом в університеті шляхом розробки вебсервісу, який забезпечує централізований доступ до навчальних матеріалів, розкладу занять, результатів навчання, а також сприяє ефективній комунікації між усіма учасниками освітнього процесу.

Призначення роботи – розробка вебсервісу для організації та управління освітнім процесом в університеті.

4. Вихідні дані для проведення НДР

1. Joshua B. Effective Java Programming Language Guide. Boston: Addison-Wesley Professional, 2018. 412p.

2. Романюк О. Н. Організація баз даних і знань / О. Н. Романюк, Т. О. Савчук. – Вінниця: УНІВЕРСУМ-Вінниця, 2003. – 105 с.

3. User Interface Design – Methods and Qualities of a Good User Interface Design. Ravi Chandra Chaitanya Guntupalli, 2008. – 12с.

5. Технічні вимоги

- середовище розробки – IntelliJ IDEA;
- мова програмування – Java;

- використанні додаткові бібліотеки та фреймворки – Spring;

- вхідні дані: облікові дані користувача;

- вихідні дані: вебсервіс із графічним користувацьким інтерфейсом для перегляду та взаємодії з освітніми ресурсами.

6. Конструктивні вимоги.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до бакалаврської кваліфікаційної роботи;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану питання та постановка задач дослідження	25.03.2025 – 30.03.2025
2	Аналіз вхідних та вихідних даних системи	05.04.2025 – 15.04.2025
3	Розробка моделі та методів системи	20.04.2025 – 25.04.2025
4	Розробка інтерфейсу користувачв	26.04.2025 – 02.05.2025
5	Розробка програмних модулів системи	10.05.2025 – 18.05.2025
6	Тестування системи	20.05.2025 – 24.05.2025
7	Оформлення матеріалів до захисту БКР	25.05.2025 – 30.05.2025

10. Порядок контролю та прийняття.

Всі етапи бакалаврської кваліфікаційної роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту. Дозволяється корегування бакалаврської кваліфікаційної роботи.

66

Додадот Б. Перевірка на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Вебсервіс для організації та управління освітнім процесом в університеті»

Тип роботи: Бакалаврська кваліфікаційна робота

(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ: кафедра програмного забезпечення , ФІТКІ

(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 10,5 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Черноволик Г. О. к.т.н доц. каф. ПЗ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Адаменко В.О.
(підпис) (прізвище, ініціали)

Додаток В. Лістинг програми

Клас AdministratorControllerImpl.java

```
package ua.com.foxmineded.universitycms.controllers.impl;
import java.security.Principal;
import java.util.Arrays;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.validation.ConstraintViolationException;
import org.springframework.data.domain.Sort;
import org.springframework.security.core.userdetails.UserDetailsService;
import lombok.RequiredArgsConstructor;
import lombok.SneakyThrows;
import ua.com.foxmineded.universitycms.controllers.AdministratorController;
import ua.com.foxmineded.universitycms.dto.AdministratorDto;
import ua.com.foxmineded.universitycms.dto.UserDto;
import ua.com.foxmineded.universitycms.exceptions.ServiceDataIntegrityException;
import ua.com.foxmineded.universitycms.exceptions.ServiceException;
import ua.com.foxmineded.universitycms.services.AdministratorService;
@Controller
@RequestMapping("/administrators")
@RequiredArgsConstructor
public class AdministratorControllerImpl implements AdministratorController {
   private final AdministratorService administratorService;
   private final UserDetailsService userDetailsService;
   @GetMapping("/{id}/findById")
   @Override
   public String findById(Principal principal, Model model, @PathVariable Long id)
throws ServiceException {
      AdministratorDto administratorDto = administratorService.findById(id);
      UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
      model.addAttribute("principal", userDto);
      model.addAttribute("administrators", Arrays.asList(administratorDto));
```

```
        model.addAttribute("currentPage", 1);
        model.addAttribute("totalItems", 1);
        model.addAttribute("totalPages", 1);
        model.addAttribute("pageSize", 1);
        model.addAttribute("currentPa3gePath",
"/administrators/%d/findById".formatted(id));
        return "administrators";
    }
    @GetMapping("/{login}/findByLogin")
    @Override
    public String findByLogin(Principal principal, Model model, @PathVariable String
login) throws ServiceException {
        AdministratorDto administratorDto = administratorService.findByLogin(login);
        UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
        model.addAttribute("principal", userDto);
        model.addAttribute("administrators", Arrays.asList(administratorDto));
        model.addAttribute("currentPage", 1);
        model.addAttribute("totalItems", 1);
        model.addAttribute("totalPages", 1);
        model.addAttribute("pageSize", 1);
        model.addAttribute("currentPagePath",
"/administrators/%s/findByLogin".formatted(login));
        return "administrators";
    }

    @GetMapping("/{passportNumber}/findByPassportNumber")
    @Override
    public String findByPassportNumber(Principal principal, Model model,
@PathVariable String passportNumber)
            throws ServiceException {
        AdministratorDto administratorDto =
administratorService.findByPassportNumber(passportNumber);
        UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
        model.addAttribute("principal", userDto);
        model.addAttribute("administrators", Arrays.asList(administratorDto));
        model.addAttribute("currentPage", 1);
        model.addAttribute("totalItems", 1);
        model.addAttribute("totalPages", 1);
        model.addAttribute("pageSize", 1);
        model.addAttribute("currentPagePath",
"/administrators/%s/findByPassportNumber".formatted(passportNumber));
        return "administrators";
    }
```

```
    @GetMapping("/{telephoneNumber}/findByTelephoneNumber")
    @Override
    public String findByTelephoneNumber(Principal principal, Model model,
@PathVariable String telephoneNumber)
            throws ServiceException {
        AdministratorDto administratorDto =
administratorService.findByTelephoneNumber(telephoneNumber);

        UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
model.addAttribute("principal", userDto);

        model.addAttribute("administrators", Arrays.asList(administratorDto));
        model.addAttribute("currentPage", 1);
        model.addAttribute("totalItems", 1);
        model.addAttribute("totalPages", 1);
        model.addAttribute("pageSize", 1);
        model.addAttribute("currentPagePath",
"/administrators/%s/findByTelephoneNumber".formatted(telephoneNumber));
        return "administrators";
    }
    @GetMapping("/{name}/findByName")
    @Override
    public String findByName(Principal principal, @RequestParam(defaultValue = "1")
int page,
            @RequestParam(defaultValue = "10") int size, Model model, @PathVariable
String name) {
        Page<AdministratorDto> pageAdministrators =
administratorService.findAllByName(name,
                PageRequest.of(page - 1, size).withSort(Sort.by("id")));
        UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
        model.addAttribute("principal", userDto);
        model.addAttribute("administrators", pageAdministrators.getContent());
        model.addAttribute("currentPage", pageAdministrators.getNumber() + 1);
        model.addAttribute("totalItems", pageAdministrators.getTotalElements());
        model.addAttribute("totalPages", pageAdministrators.getTotalPages());
        model.addAttribute("pageSize", size);
        model.addAttribute("currentPagePath",
"/administrators/%s/findByName".formatted(name));
        return "administrators";
    }
    @GetMapping
    @Override
```

```
  public String findAll(@RequestParam(defaultValue = "1") int page,
@RequestParam(defaultValue = "10") int size,
      Principal principal, Model model) throws ServiceException {
    Page<AdministratorDto> pageAdministrators = administratorService
        .findAll(PageRequest.of(page - 1, size).withSort(Sort.by("id")));
    UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
    model.addAttribute("principal", userDto);
    model.addAttribute("administrators", pageAdministrators.getContent());
    model.addAttribute("currentPage", pageAdministrators.getNumber() + 1);
    model.addAttribute("totalItems", pageAdministrators.getTotalElements());
    model.addAttribute("totalPages", pageAdministrators.getTotalPages());
    model.addAttribute("pageSize", size);
   model.addAttribute("currentPagePath", "/administrators");
    return "administrators"
  }
  @GetMapping("/new")
  @Override
   public String createAdministrator(Model model, HttpServletRequest request) {
    AdministratorDto administratorDto = new AdministratorDto();
    String referer = request.getHeader("referer").replace("http://localhost:8080", "");
    model.addAttribute("administrator", administratorDto);
    model.addAttribute("referer", referer);
    return "administrator-create";
  }


  @PostMapping("/save")
   @SneakyThrows
   @Override
   public String saveAdministrator(@ModelAttribute("administrator")
dministratorDto administratorDto, Model model,
      HttpServletRequest request) {
    String referer = (request.getParameter("referer").contains("/new")
        || request.getParameter("referer").contains("/update")) ? "/administrators"
            : request.getParameter("referer");
   model.addAttribute("referer", referer);
    String viewName = request.getParameter("viewName");
    try {
      administratorService.save(administratorDto);

    } catch (ConstraintViolationException e) {
model.addAttribute("validationExceptions", e.getConstraintViolations());

      model.addAttribute("administrator", administratorDto);
```

```
        if ("administrator-create".equals(viewName)) {
            return "administrator-create";
        }
        if ("administrator-update".equals(viewName)) {
            return "administrator-update";
        }
    } catch (ServiceDataIntegrityException e) {
        model.addAttribute("integrityExceptions", e.getExceptions());

        model.addAttribute("administrator", administratorDto);
        if ("administrator-create".equals(viewName)) {
            return "administrator-create";
        }       if ("administrator-update".equals(viewName)) {
            return                                          "administrator-update";
        }
    }
    return              "redirect:"              +              referer;
}
@GetMapping("/{id}/update")
@Override
public String updateAdministrator(@PathVariable Long id, Model model,
HttpServletRequest request)          throws ServiceException {      String referer =
request.getHeader("referer").replace("http://localhost:8080", "");
    AdministratorDto administratorDto = administratorService.findById(id);

    model.addAttribute("administrator", administratorDto);
    model.addAttribute("referer", referer);
    return "administrator-update";
}
@PostMapping("/{id}/delete")
@Override
public String deleteAdministrator(@PathVariable Long id, HttpServletRequest
request) throws ServiceException {
    String referer = request.getHeader("referer").replace("http://localhost:8080", "");
    administratorService.deleteById(id);
    return "redirect:" + referer;
}
}
```

Клас StudentControllerImpl.java

```
package ua.com.foxmineded.universitycms.controllers.impl;

import jakarta.servlet.http.HttpServletRequest;
import jakarta.validation.ConstraintViolationException;
```

```
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Sort;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import ua.com.foxmineded.universitycms.controllers.StudentController;
import ua.com.foxmineded.universitycms.dto.StudentDto;
import ua.com.foxmineded.universitycms.dto.UserDto;
import ua.com.foxmineded.universitycms.exceptions.ServiceDataIntegrityException;
import ua.com.foxmineded.universitycms.exceptions.ServiceException;
import ua.com.foxmineded.universitycms.services.StudentService;

import java.security.Principal;
import java.util.Arrays;
import java.util.Objects;

@Controller
@RequestMapping("/students")
@RequiredArgsConstructor
public class StudentControllerImpl implements StudentController {
   private final StudentService studentService;
   private final UserDetailsService userDetailsService;

   @GetMapping("/{id}/findById")
   @Override
   public String findById(@RequestParam(required = false) Long courseId,
@RequestParam(required = false) Long groupId,
                          @RequestParam(required = false) String purpose, Model model,
Principal principal, @PathVariable Long id)
           throws ServiceException {
       StudentDto studentDto = studentService.findById(id);
       UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
       model.addAttribute("principal", userDto);
       model.addAttribute("students", Arrays.asList(studentDto));
       model.addAttribute("currentPage", 1);
       model.addAttribute("totalItems", 1);
       model.addAttribute("totalPages", 1);
       model.addAttribute("pageSize", 1);
       model.addAttribute("courseId", courseId);
       model.addAttribute("groupId", groupId);
       model.addAttribute("purpose", purpose);
```

```
        String currentPagePath = "/students/%d/findById".formatted(id);
        if (Objects.nonNull(purpose)) {
            currentPagePath += "?purpose=%s".formatted(purpose);
        }
        if (Objects.nonNull(courseId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"courseId=%s".formatted(courseId);
        }
        if (Objects.nonNull(groupId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"groupId=%s".formatted(groupId);
        }
        model.addAttribute("currentPagePath", currentPagePath);
        return "students";
    }

    @GetMapping("/{passportNumber}/findByPassportNumber")
    @Override
    public String findByPassportNumber(@RequestParam(required = false) Long
courseId,
                                       @RequestParam(required = false) Long groupId,
@RequestParam(required = false) String purpose, Model model,
                                       Principal principal, @PathVariable String passportNumber)
throws ServiceException {
        StudentDto studentDto =
studentService.findByPassportNumber(passportNumber);
        UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
        model.addAttribute("principal", userDto);
        model.addAttribute("students", Arrays.asList(studentDto));
        model.addAttribute("currentPage", 1);
        model.addAttribute("totalItems", 1);
        model.addAttribute("totalPages", 1);
        model.addAttribute("pageSize", 1);
        model.addAttribute("courseId", courseId);
        model.addAttribute("groupId", groupId);
        model.addAttribute("purpose", purpose);
        String currentPagePath =
"/students/%s/findByPassportNumber".formatted(passportNumber);
        if (Objects.nonNull(purpose)) {
            currentPagePath += "?purpose=%s".formatted(purpose);
        }
        if (Objects.nonNull(courseId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"courseId=%s".formatted(courseId);
```

```
        }
        if (Objects.nonNull(groupId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"groupId=%s".formatted(groupId);
        }
        model.addAttribute("currentPagePath", currentPagePath);
        return "students";
    }

    @GetMapping("/{login}/findByLogin")
    @Override
    public String findByLogin(@RequestParam(required = false) Long courseId,
                              @RequestParam(required = false) Long groupId,
@RequestParam(required = false) String purpose, Model model,
                              Principal principal, @PathVariable String login) throws
ServiceException {
        StudentDto studentDto = studentService.findByLogin(login);
        UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
        model.addAttribute("principal", userDto);
        model.addAttribute("students", Arrays.asList(studentDto));
        model.addAttribute("currentPage", 1);
        model.addAttribute("totalItems", 1);
        model.addAttribute("totalPages", 1);
        model.addAttribute("pageSize", 1);
        model.addAttribute("courseId", courseId);
        model.addAttribute("groupId", groupId);
        model.addAttribute("purpose", purpose);
        String currentPagePath = "/students/%s/findByLogin".formatted(login);
        if (Objects.nonNull(purpose)) {
            currentPagePath += "?purpose=%s".formatted(purpose);
        }
        if (Objects.nonNull(courseId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"courseId=%s".formatted(courseId);
        }
        if (Objects.nonNull(groupId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"groupId=%s".formatted(groupId);
        }
        model.addAttribute("currentPagePath", currentPagePath);
        return "students";
    }

    @GetMapping("/{telephoneNumber}/findByTelephoneNumber")
```

```
    @Override
    public String findByTelephoneNumber(@RequestParam(required = false) Long
courseId,
                                        @RequestParam(required = false) Long groupId,
@RequestParam(required = false) String purpose, Model model,
                                        Principal principal, @PathVariable String
telephoneNumber) throws ServiceException {
        StudentDto studentDto =
studentService.findByTelephoneNumber(telephoneNumber);
        UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
        model.addAttribute("principal", userDto);
        model.addAttribute("students", Arrays.asList(studentDto));
        model.addAttribute("currentPage", 1);
        model.addAttribute("totalItems", 1);
        model.addAttribute("totalPages", 1);
        model.addAttribute("pageSize", 1);
        model.addAttribute("courseId", courseId);
        model.addAttribute("groupId", groupId);
        model.addAttribute("purpose", purpose);
        String currentPagePath =
"/students/%s/findByTelephoneNumber".formatted(telephoneNumber);
        if (Objects.nonNull(purpose)) {
            currentPagePath += "?purpose=%s".formatted(purpose);
        }
        if (Objects.nonNull(courseId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"courseId=%s".formatted(courseId);
        }
        if (Objects.nonNull(groupId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"groupId=%s".formatted(groupId);
        }
        model.addAttribute("currentPagePath", currentPagePath);
        return "students";
    }

    @GetMapping
    @Override
    public String findAll(@RequestParam(defaultValue = "1") int page,
@RequestParam(defaultValue = "10") int size,
                          @RequestParam(required = false) Long courseId,
@RequestParam(required = false) Long groupId,
                          @RequestParam(required = false) String purpose, Principal principal,
Model model) {
```

```
        Page<StudentDto> pageStudents = studentService.findAll(PageRequest.of(page
- 1, size).withSort(Sort.by("id")));
        UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
        model.addAttribute("principal", userDto);
        model.addAttribute("students", pageStudents.getContent());
        model.addAttribute("currentPage", pageStudents.getNumber() + 1);
        model.addAttribute("totalItems", pageStudents.getTotalElements());
        model.addAttribute("totalPages", pageStudents.getTotalPages());
        model.addAttribute("pageSize", size);
        model.addAttribute("courseId", courseId);
        model.addAttribute("groupId", groupId);
        model.addAttribute("purpose", purpose);
        String currentPagePath = "/students";
        if (Objects.nonNull(purpose)) {
            currentPagePath += "?purpose=%s".formatted(purpose);
        }
        if (Objects.nonNull(courseId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"courseId=%s".formatted(courseId);
        }
        if (Objects.nonNull(groupId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"groupId=%s".formatted(groupId);
        }
        model.addAttribute("currentPagePath", currentPagePath);
        return "students";
    }

    @GetMapping("/{name}/findAllByName")
    @Override
    public String findAllByName(@RequestParam(defaultValue = "1") int page,
@RequestParam(defaultValue = "10") int size,
                                @RequestParam(required = false) Long courseId,
@RequestParam(required = false) Long groupId,
                                @RequestParam(required = false) String purpose, Principal
principal, Model model,
                                @PathVariable String name) {
        Page<StudentDto> pageStudents = studentService.findAllByName(name,
                PageRequest.of(page - 1, size).withSort(Sort.by("id")));
        UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
        model.addAttribute("principal", userDto);
        model.addAttribute("students", pageStudents.getContent());
        model.addAttribute("currentPage", pageStudents.getNumber() + 1);
```

```
        model.addAttribute("totalItems", pageStudents.getTotalElements());
        model.addAttribute("totalPages", pageStudents.getTotalPages());
        model.addAttribute("pageSize", size);
        model.addAttribute("courseId", courseId);
        model.addAttribute("groupId", groupId);
        model.addAttribute("purpose", purpose);
        String currentPagePath = "/students/%s/findAllByName".formatted(name);
        if (Objects.nonNull(purpose)) {
            currentPagePath += "?purpose=%s".formatted(purpose);
        }
        if (Objects.nonNull(courseId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"courseId=%s".formatted(courseId);
        }
        if (Objects.nonNull(groupId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"groupId=%s".formatted(groupId);
        }
        model.addAttribute("currentPagePath", currentPagePath);
        return "students";
    }

    @GetMapping("/{groupName}/findAllByGroupName")
    @Override
    public String findAllByGroupName(@RequestParam(defaultValue = "1") int page,
                                     @RequestParam(defaultValue = "10") int size,
@RequestParam(required = false) Long courseId,
                                     @RequestParam(required = false) Long groupId,
@RequestParam(required = false) String purpose,
                                     Principal principal, Model model, @PathVariable String
groupName) {
        Page<StudentDto> pageStudents =
studentService.findAllByGroupName(groupName,
                PageRequest.of(page - 1, size).withSort(Sort.by("id")));
        UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
        model.addAttribute("principal", userDto);
        model.addAttribute("students", pageStudents.getContent());
        model.addAttribute("currentPage", pageStudents.getNumber() + 1);
        model.addAttribute("totalItems", pageStudents.getTotalElements());
        model.addAttribute("totalPages", pageStudents.getTotalPages());
        model.addAttribute("pageSize", size);
        model.addAttribute("courseId", courseId);
        model.addAttribute("groupId", groupId);
        model.addAttribute("purpose", purpose);
```

```
        String currentPagePath =
"/students/%s/findAllByGroupName".formatted(groupName);
        if (Objects.nonNull(purpose)) {
            currentPagePath += "?purpose=%s".formatted(purpose);
        }
        if (Objects.nonNull(courseId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"courseId=%s".formatted(courseId);
        }
        if (Objects.nonNull(groupId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"groupId=%s".formatted(groupId);
        }
        model.addAttribute("currentPagePath", currentPagePath);
        return "students";
    }

    @GetMapping("/{courseName}/findAllByCourseName")
    @Override
    public String findAllByCourseName(@RequestParam(defaultValue = "1") int
page,
                                      @RequestParam(defaultValue = "10") int size,
@RequestParam(required = false) Long courseId,
                                      @RequestParam(required = false) Long groupId,
@RequestParam(required = false) String purpose,
                                      Principal principal, Model model, @PathVariable String
courseName) {
        Page<StudentDto> pageStudents =
studentService.findAllByCourseName(courseName,
                PageRequest.of(page - 1, size).withSort(Sort.by("id")));
        UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
        model.addAttribute("principal", userDto);
        model.addAttribute("students", pageStudents.getContent());
        model.addAttribute("currentPage", pageStudents.getNumber() + 1);
        model.addAttribute("totalItems", pageStudents.getTotalElements());
        model.addAttribute("totalPages", pageStudents.getTotalPages());
        model.addAttribute("pageSize", size);
        model.addAttribute("courseId", courseId);
        model.addAttribute("groupId", groupId);
        model.addAttribute("purpose", purpose);
        String currentPagePath =
"/students/%s/findAllByCourseName".formatted(courseName);
        if (Objects.nonNull(purpose)) {
            currentPagePath += "?purpose=%s".formatted(purpose);
```

```
        }
        if (Objects.nonNull(courseId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"courseId=%s".formatted(courseId);
        }
        if (Objects.nonNull(groupId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"groupId=%s".formatted(groupId);
        }
        model.addAttribute("currentPagePath", currentPagePath);
        return "students";
    }

    @GetMapping("/{teacherName}/findAllByTeacherName")
    @Override
    public String findAllByTeacherName(@RequestParam(defaultValue = "1") int
page,
                                       @RequestParam(defaultValue = "10") int size,
@RequestParam(required = false) Long courseId,
                                       @RequestParam(required = false) Long groupId,
@RequestParam(required = false) String purpose,
                                       Principal principal, Model model, @PathVariable String
teacherName) {
        Page<StudentDto> pageStudents =
studentService.findAllByTeacherName(teacherName,
                PageRequest.of(page - 1, size).withSort(Sort.by("id")));
        UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
        model.addAttribute("principal", userDto);
        model.addAttribute("students", pageStudents.getContent());
        model.addAttribute("currentPage", pageStudents.getNumber() + 1);
        model.addAttribute("totalItems", pageStudents.getTotalElements());
        model.addAttribute("totalPages", pageStudents.getTotalPages());
        model.addAttribute("pageSize", size);
        model.addAttribute("courseId", courseId);
        model.addAttribute("groupId", groupId);
        model.addAttribute("purpose", purpose);
        String currentPagePath =
"/students/%s/findAllByTeacherName".formatted(teacherName);
        if (Objects.nonNull(purpose)) {
            currentPagePath += "?purpose=%s".formatted(purpose);
        }
        if (Objects.nonNull(courseId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"courseId=%s".formatted(courseId);
```

```
        }
        if (Objects.nonNull(groupId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"groupId=%s".formatted(groupId);
        }
        model.addAttribute("currentPagePath", currentPagePath);
        return "students";
    }

    @GetMapping(value = "/{specialization}/findAllBySpecialization")
    @Override
    public String findAllBySpecialization(@RequestParam(defaultValue = "1") int
page,
                                          @RequestParam(defaultValue = "10") int size,
@RequestParam(required = false) Long courseId,
                                          @RequestParam(required = false) Long groupId,
@RequestParam(required = false) String purpose,
                                          Principal principal, Model model, @PathVariable String
specialization) throws ServiceException {
        Page<StudentDto> pageStudents =
studentService.findAllBySpecialization(specialization,
                PageRequest.of(page - 1, size).withSort(Sort.by("id")));
        UserDto userDto = (UserDto)
userDetailsService.loadUserByUsername(principal.getName());
        model.addAttribute("principal", userDto);
        model.addAttribute("students", pageStudents.getContent());
        model.addAttribute("currentPage", pageStudents.getNumber() + 1);
        model.addAttribute("totalItems", pageStudents.getTotalElements());
        model.addAttribute("totalPages", pageStudents.getTotalPages());
        model.addAttribute("pageSize", size);
        model.addAttribute("courseId", courseId);
        model.addAttribute("groupId", groupId);
        model.addAttribute("purpose", purpose);
        String currentPagePath =
"/students/%s/findAllBySpecialization".formatted(specialization);
        if (Objects.nonNull(purpose)) {
            currentPagePath += "?purpose=%s".formatted(purpose);
        }
        if (Objects.nonNull(courseId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"courseId=%s".formatted(courseId);
        }
        if (Objects.nonNull(groupId)) {
            currentPagePath += (currentPagePath.contains("?") ? "&" : "?") +
"groupId=%s".formatted(groupId);
```

```
        }
        model.addAttribute("currentPagePath", currentPagePath);
        return "students";
    }

    @GetMapping("/new")
    @Override
    public String createStudent(Model model, HttpServletRequest request) {
        StudentDto studentDto = new StudentDto();
        String referer = request.getHeader("referer").replace("http://localhost:8080", "");
        model.addAttribute("student", studentDto);
        model.addAttribute("referer", referer);
        return "student-create";
    }

    @PostMapping("/save")
    @Override
    public String saveStudent(@ModelAttribute("student") StudentDto studentDto,
Model model, HttpServletRequest request)
            throws ServiceDataIntegrityException {
        String referer = (request.getParameter("referer").contains("/new")
                || request.getParameter("referer").contains("/update")) ? "/students" :
request.getParameter("referer");
        model.addAttribute("referer", referer);
        String viewName = request.getParameter("viewName");

        try {
            studentService.save(studentDto);
        } catch (ConstraintViolationException e) {
            model.addAttribute("validationExceptions", e.getConstraintViolations());
            model.addAttribute("student", studentDto);
            if ("student-create".equals(viewName)) {
                return "student-create";
            }
            if ("student-update".equals(viewName)) {
                return "student-update";
            }
        } catch (ServiceDataIntegrityException e) {
            model.addAttribute("integrityExceptions", e.getExceptions());
            model.addAttribute("student", studentDto);
            if ("student-create".equals(viewName)) {
                return "student-create";
            }
            if ("student-update".equals(viewName)) {
                return "student-update";
```

```
            }
        }
        return "redirect:" + referer;
    }

    @GetMapping("/{id}/update")
    @Override
    public String updateStudent(@PathVariable Long id, Model model,
HttpServletRequest request)
            throws ServiceException {
        String referer = request.getHeader("referer").replace("http://localhost:8080", "");
        StudentDto studentDto = studentService.findById(id);
        model.addAttribute("student", studentDto);
        model.addAttribute("referer", referer);
        return "student-update";
    }

    @PostMapping("/{id}/delete")
    @Override
    public String deleteStudent(@PathVariable Long id, HttpServletRequest request)
throws ServiceException {
        String referer = request.getHeader("referer").replace("http://localhost:8080", "");
        studentService.deleteById(id);
        return "redirect:" + referer;
    }

}
```

Клас adminastrationCreate.js:

```
document.addEventListener('DOMContentLoaded', function() {
    document.getElementById('file').addEventListener('change', function() {
        var fileInput = document.getElementById('file');
        var file = fileInput.files[0];

        if (file) {
            var formData = new FormData();
            formData.append('file', file);
            var csrfToken = document.getElementById("csrf").value;
            formData.append('_csrf', csrfToken);
            var xhr = new XMLHttpRequest();
            xhr.open('POST', '/avatars?role=ADMINISTRATOR', true);

            xhr.onload = function() {
                if (xhr.status >= 200 && xhr.status < 300) {
```

```
            var contentType = xhr.getResponseHeader('Content-Type');
            if (contentType && contentType.includes('text/html')) {
              document.open();
              document.write(xhr.responseText);
              document.close();
            } else {
              var response = JSON.parse(xhr.responseText);
              let imageUrl = "/avatars/" + response.id;
              document.getElementById('photoId').value = response.id;
              document.getElementById('imagePreview').src = imageUrl;
            }
          }
        };
        xhr.send(formData);
      }
    });
});

document.addEventListener('DOMContentLoaded', function() {
    var selectElement = document.getElementById("gender");
    if (window.gender !== null) {
      selectElement.classList.add('is-valid');
    }
});

document.addEventListener('DOMContentLoaded', function() {
    var selectElement = document.getElementById("workingShift");
    if (window.workingShift !== null) {
      selectElement.classList.add('is-valid');
    }
});
```

Клас timetable.js:

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<head>

    <head
      th:replace="~{fragments/general.html :: head(description='The page to visualize
timetable for a student', title='Timetable')}">
    </head>
    <link href="/css/timetable.css" rel="stylesheet" type="text/css">
</head>
```

```
<body>
   <nav th:replace="~{fragments/general.html :: navbar(${principal})}"></nav>
   <div th:replace="~{fragments/general.html ::
principalModel(${principal})}"></div>
   <div class="container-fluid">
     <main>
       <div class="row">
         <div class="col-xs-12">
           <a>
             <h1>[[${month}]] [[${year}]]</h1>
           </a>
         </div>
       </div>
       <div class="row">
         <div class="col-xs-12">
           <table class="table table-bordered">
             <thead class="table-light">
               <tr>
                 <th scope="col">Monday</th>
                 <th scope="col">Tuesday</th>
                 <th scope="col">Wednesday</th>
                 <th scope="col">Thursday</th>
                 <th scope="col">Friday</th>
                 <th scope="col">Saturday</th>
                 <th scope="col">Sunday</th>
               </tr>
             </thead>
             <tbody>
               <tr th:each="externalEntry : ${timetable}">
                 <td th:each="internalEntry : ${externalEntry.value}" ,
                   th:class="${internalEntry.value == null ? 'transparent-cell' : ''}">
                   <ul class="my-2">
                     <li th:each="lesson, i : ${internalEntry.value}" th:if="${i.index
== 0}"
                       th:class="${lesson.lessonDate} == ${todayDate} ? 'color-red' :
''">
                       <span>[[${lesson.lessonDate}]]</span>
                     </li>
                   </ul>
                   <ul class="my-0 mx-0 py-0 px-0" style="list-style-type: none;">
                     <span th:if="${internalEntry.value != null} and
${#lists.size(internalEntry.value)} == 0">
                       <br><br><br><br>
                     </span>
```

```
                        <li th:each="lesson, i : ${internalEntry.value}"
                          th:class="${lesson.lessonDate} == ${todayDate} ? 'color-red' :
"">
                          <span>[[${lesson.courseName}]] lesson
#[[${lesson.lessonNumber}]] room #[[${lesson.roomNumber}]]</span>
                        </li>
                      </ul>
                    </td>
                  </tr>
                </tbody>
              </table>
            </div>
          </div>
          <div class="row">
            <div class="col-sm-6 text-end">
              <a th:href="${currentUrl} + '?addMonths=' + ${addMonths - 1}">
                <button type="button" class="btn btn-danger btn-lg">Previous
month</button>
              </a>
            </div>
            <div class="col-sm-6 text-start">
              <a th:href="${currentUrl} + '?addMonths=' + ${addMonths + 1}">
              <button type="button" class="btn btn-primary btn-lg">Next
month</button>
              </a>
            </div>
          </div>
        </main>
      </div>
      <script src="/js/timetable.js"></script>
      <script src="/webjars/jquery/3.7.1/jquery.min.js"></script>
      <script src="/webjars/bootstrap/5.3.2/js/bootstrap.bundle.min.js"></script>
</body>

</html>
```

Клас groups.html:

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<head>

  <head
```

```
    th:replace="~{fragments/general.html :: head(description='The page to visualize
all available groups',title='Groups')}">
  </head>
  <link href="/css/groups.css" rel="stylesheet" type="text/css">

</head>

<body>
  <nav th:replace="~{fragments/general.html :: navbar(${principal})}"></nav>
  <div th:replace="~{fragments/general.html ::
principalModel(${principal})}"></div>
  <div class="container-fluid">
    <main>
      <div class="row">
        <div class="col-md-4">
          <a href="/groups">
            <h1>Groups</h1>
          </a>
        </div>
      </div>
      <div class="row">
        <div class="col-sm-6">
          <a sec:authorize="hasRole('ADMINISTRATOR')" href="/groups/new">
            <button class="create-button">Create</button>
          </a>
        </div>
        <div class="col-sm-6">
          <nav class="navbar navbar-expand-lg">
            <div class="container-fluid">
              <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <form action="#" id="searchForm" class="d-flex ms-auto"
method="GET">
                  <div class="me-2">
                    <input id="searchInput" class="form-control" type="number"
placeholder="Search"
                      aria-label="Search">
                  </div>
                  <button class="btn btn-success me-2"
type="submit">Search</button>
                  <div class="btn-group dropup">
                    <button type="button" class="btn btn-secondary dropdown-
toggle"
                      data-bs-toggle="dropdown" aria-expanded="false">
                      Search by
                    </button>
```

```
                    <ul class="dropdown-menu dropdown-menu-lg-end text-menu"
                      aria-labelledby="dropdownMenuClickableInside">
                      <li><a id="idLink" class="dropdown-item" data-
type="number">id</a></li>
                      <li><a id="groupNameLink" class="dropdown-item" data-
type="text">group
                          name</a></li>
                      <li><a id="specializationLink" class="dropdown-item"
                          data-type="text">specialization</a></li>
                      <li><a id="studentAmountLink" class="dropdown-item"
                          data-type="number">student amount</a></li>
                    </ul>
                  </div>
                </form>
              </div>
            </div>
          </nav>
        </div>
      </div>
      <div class="row">
        <div class="col-xs-12">
          <table class="table-responsive-md table table-hover table-striped my-
custom-table">
            <thead>
              <tr>
                <th scope="col">Id</th>
                <th scope="col">Group name</th>
                <th scope="col">Specialization</th>
                <th sec:authorize="hasRole('ADMINISTRATOR') or
hasRole('TEACHER')" scope="col">Student management</th>
                <th class="my-custom-border"
sec:authorize="hasRole('ADMINISTRATOR')" scope="col">Action
                </th>
              </tr>
            </thead>
            <tbody>
              <tr th:each="group : ${groups}">
                <th scope="row" th:text="${group.id}"></th>
                <td th:text="${group.groupName}"></td>
                <td th:text="${group.specialization}"></td>
                <td sec:authorize="hasRole('ADMINISTRATOR') or
hasRole('TEACHER')">
                  <a
th:href="@{/students/{specialization}/findAllBySpecialization?purpose=addToGrou
p&groupId={groupId} (specialization=${group.specialization},
```

```
groupId=${group.id})}">
                        <button type="button" class="btn btn-outline-success">Add
students</button>
                    </a>
                    <a
th:href="@{/students/{groupName}/findAllByGroupName?purpose=deleteFromGro
up&groupId={groupId} (groupName=${group.groupName},
groupId=${group.id})}">
                        <button type="button" class="btn btn-outline-warning">Manage
students</button>
                    </a>
                  </td>
                  <td class="my-custom-border"
sec:authorize="hasRole('ADMINISTRATOR')">
                    <a th:href="@{/groups/{id}/update (id=${group.id})}">
                      <button type="button" class="update-button">Update</button>
                    </a>
                    <br>
                    <form th:action="@{/groups/{id}/delete (id=${group.id})}"
method="POST">
                      <button type="submit" class="delete-button">Delete</button>
                    </form>
                  </td>
                </tr>
              </tbody>
            </table>
            <div th:unless="${groups.size() > 0}">
              <div class="row">
                <div class="col-xs-12 text-center mt-4">
                  <span>No groups found!</span>
                </div>
              </div>
            </div>
            <div class="row">
              <div class="col-xs-12">
                <nav
                  th:replace="~{fragments/general :: pagination(${currentPagePath},
${currentPage}, ${totalPages}, ${page}, ${pageSize})}">
                </nav>
              </div>
            </div>
          </div>
        </div>
      </main>
    </div>
```

```
    <script src="/js/groups.js"></script>
    <script src="/webjars/jquery/3.7.1/jquery.min.js"></script>
    <script src="/webjars/bootstrap/5.3.2/js/bootstrap.bundle.min.js"></script>
</body>

</html>
```

Клас AdministratorService.java:

```
package ua.com.foxmineded.universitycms.services.impl;

import java.util.Objects;
import org.modelmapper.ModelMapper;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.validation.annotation.Validated;
import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import ua.com.foxmineded.universitycms.dao.AdministratorRepository;
import ua.com.foxmineded.universitycms.dao.StudentRepository;
import ua.com.foxmineded.universitycms.dao.TeacherRepository;
import ua.com.foxmineded.universitycms.dto.AdministratorDto;
import ua.com.foxmineded.universitycms.entities.impl.Administrator;
import ua.com.foxmineded.universitycms.exceptions.ServiceDataIntegrityException;
import ua.com.foxmineded.universitycms.exceptions.ServiceException;
import ua.com.foxmineded.universitycms.services.AdministratorService;

@Slf4j
@Service
@Validated
@RequiredArgsConstructor
public class AdministratorServiceImpl implements AdministratorService {
    private final ModelMapper modelMapper;
    private final PasswordEncoder passwordEncoder;
    private final AdministratorRepository administratorRepository;
    private final StudentRepository studentRepository;
    private final TeacherRepository teacherRepository;

    @Override
    public Page<AdministratorDto> findAll(Pageable pageable) {
      return administratorRepository.findAll(pageable).map(v -> modelMapper.map(v,
```

```
AdministratorDto.class));
    }

    @Override
    public AdministratorDto findById(Long administratorId) throws ServiceException
{
        return administratorRepository.findById(administratorId)
                .map(administrator -> modelMapper.map(administrator,
AdministratorDto.class)).orElseThrow(() -> {
                    String message = "The administrator with id = %d was not
found".formatted(administratorId);
                    log.error(message);
                    return new ServiceException(message);
                });
    }

    @Override
    public AdministratorDto findByLogin(String login) throws ServiceException {
        return administratorRepository.findByLogin(login)
                .map(administrator -> modelMapper.map(administrator,
AdministratorDto.class)).orElseThrow(() -> {
                    String message = "The administrator with login = %s was not
found".formatted(login);
                    log.error(message);
                    return new ServiceException(message);
                });
    }

    @Override
    public AdministratorDto findByPassportNumber(String passportNumber) throws
ServiceException {
        return administratorRepository.findByPassportNumber(passportNumber)
                .map(administrator -> modelMapper.map(administrator,
AdministratorDto.class)).orElseThrow(() -> {
                    String message = "The administrator with passport number = %s was not
found"
                            .formatted(passportNumber);
                    log.error(message);
                    return new ServiceException(message);
                });
    }

    @Override
    public Page<AdministratorDto> findAllByName(String name, Pageable pageable)
{
```

```
        return administratorRepository.findAllByName(name, pageable)
                .map(v -> modelMapper.map(v, AdministratorDto.class));
    }

    @Override
    public AdministratorDto findByTelephoneNumber(String telephoneNumber)
throws ServiceException {
        return administratorRepository.findByTelephoneNumber(telephoneNumber)
                .map(administrator -> modelMapper.map(administrator,
AdministratorDto.class)).orElseThrow(() -> {
                    String message = "The administrator with telephone number = %s was not
found"
                            .formatted(telephoneNumber);
                    log.error(message);
                    return new ServiceException(message);
                });
    }

    @Override
    public AdministratorDto save(@Valid AdministratorDto administratorDto) throws
ServiceDataIntegrityException {
        ServiceDataIntegrityException serviceDataIntegrityException = new
ServiceDataIntegrityException();
        if (studentRepository.findByLogin(administratorDto.getLogin()).isPresent()
                || teacherRepository.findByLogin(administratorDto.getLogin()).isPresent()) {
            String message = "A user with this login already exists";
            log.error(message);
            serviceDataIntegrityException.getExceptions().add(new
ServiceDataIntegrityException("login", message));
        } else {

administratorRepository.findByLogin(administratorDto.getLogin()).ifPresent(value -
> {
                    if (!Objects.equals(administratorDto.getId(), value.getId())) {
                        String message = "A user with this login already exists";
                        log.error(message);
                        serviceDataIntegrityException.getExceptions()
                                .add(new ServiceDataIntegrityException("login", message));
                    }
                });
        }
        if (studentRepository.findByEmail(administratorDto.getEmail()).isPresent()
                || teacherRepository.findByEmail(administratorDto.getEmail()).isPresent()) {
            String message = "A user with this email already exists";
            log.error(message);
```

```
        serviceDataIntegrityException.getExceptions().add(new
ServiceDataIntegrityException("email", message));
    } else {

administratorRepository.findByEmail(administratorDto.getEmail()).ifPresent(value -
> {
          if (!Objects.equals(administratorDto.getId(), value.getId())) {
            String message = "A user with this email already exists";
            log.error(message);
            serviceDataIntegrityException.getExceptions()
                .add(new ServiceDataIntegrityException("email", message));
          }
        });

    }
    if
(studentRepository.findByTelephoneNumber(administratorDto.getTelephoneNumber
()).isPresent()
          ||
teacherRepository.findByTelephoneNumber(administratorDto.getTelephoneNumber(
)).isPresent()) {
        String message = "A user with this telephone number already exists";
        log.error(message);
        serviceDataIntegrityException.getExceptions()
            .add(new ServiceDataIntegrityException("telephoneNumber", message));
    } else {
administratorRepository.findByTelephoneNumber(administratorDto.getTelephoneNu
mber()).ifPresent(value -> {
          if (!Objects.equals(administratorDto.getId(), value.getId())) {
            String message = "A user with this telephone number already exists";
            log.error(message);
            serviceDataIntegrityException.getExceptions()
                .add(new ServiceDataIntegrityException("telephoneNumber",
message));
          }
        });
    }
    if
(studentRepository.findByPassportNumber(administratorDto.getPassportNumber()).i
sPresent()
          ||
teacherRepository.findByPassportNumber(administratorDto.getPassportNumber()).is
Present()) {
        String message = "A user with this passport number already exists";
        log.error(message);
```

```
        serviceDataIntegrityException.getExceptions()
            .add(new ServiceDataIntegrityException("passportNumber", message));
      } else {
administratorRepository.findByPassportNumber(administratorDto.getPassportNumbe
r()).ifPresent(value -> {
          if (!(Objects.equals(administratorDto.getId(), value.getId()))) {
            String message = "A user with this passport number already exists";
            log.error(message);
            serviceDataIntegrityException.getExceptions()
                .add(new ServiceDataIntegrityException("passportNumber", message));
          }
        });
      }
      if (!serviceDataIntegrityException.getExceptions().isEmpty()) {
        throw serviceDataIntegrityException;
      }
      if (Objects.isNull(administratorDto.getId())) {

administratorDto.setPassword(passwordEncoder.encode(administratorDto.getPasswo
rd()));
      }
      AdministratorDto result = modelMapper.map(
          administratorRepository.save(modelMapper.map(administratorDto,
Administrator.class)),
          AdministratorDto.class);
      String message = "The administrator with id = %d was
saved".formatted(result.getId());
      log.info(message);
      return result;
    }
    @Override
    public void deleteById(Long administratorId) throws ServiceException {
      administratorRepository.findById(administratorId).orElseThrow(() -> {
        String message = "The administrator with id = %d was not
found".formatted(administratorId);
        log.error(message);
        return new ServiceException(message);
      });
      administratorRepository.deleteById(administratorId);
      String message = "The administrator with id = %d was
deleted".formatted(administratorId);
      log.info(message);
    }
}
```

Додаток Г. Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА
ВЕБСЕРВІС ДЛЯ ОРГАНІЗАЦІЇ ТА УПРАВЛІННЯ ОСВІТНІМ ПРОЦЕСОМ В УНІВЕРСИТЕТІ

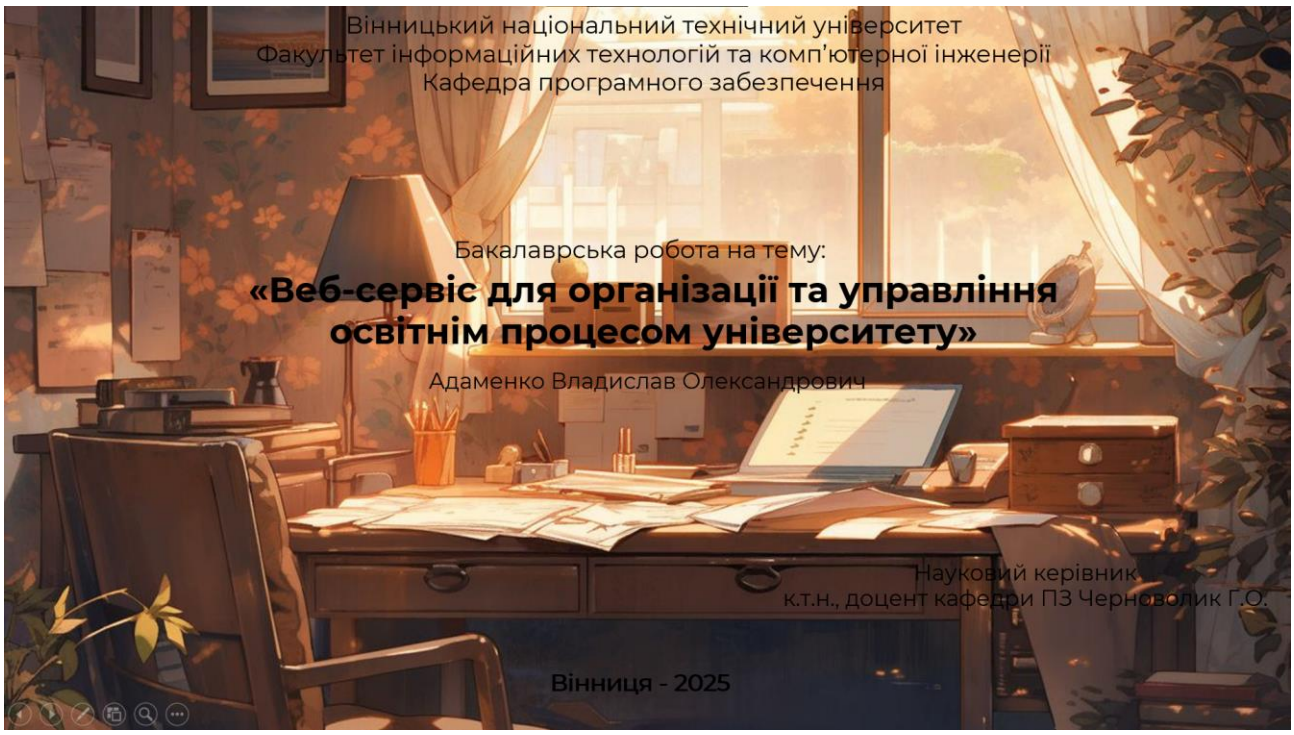


Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Актуальність розробки

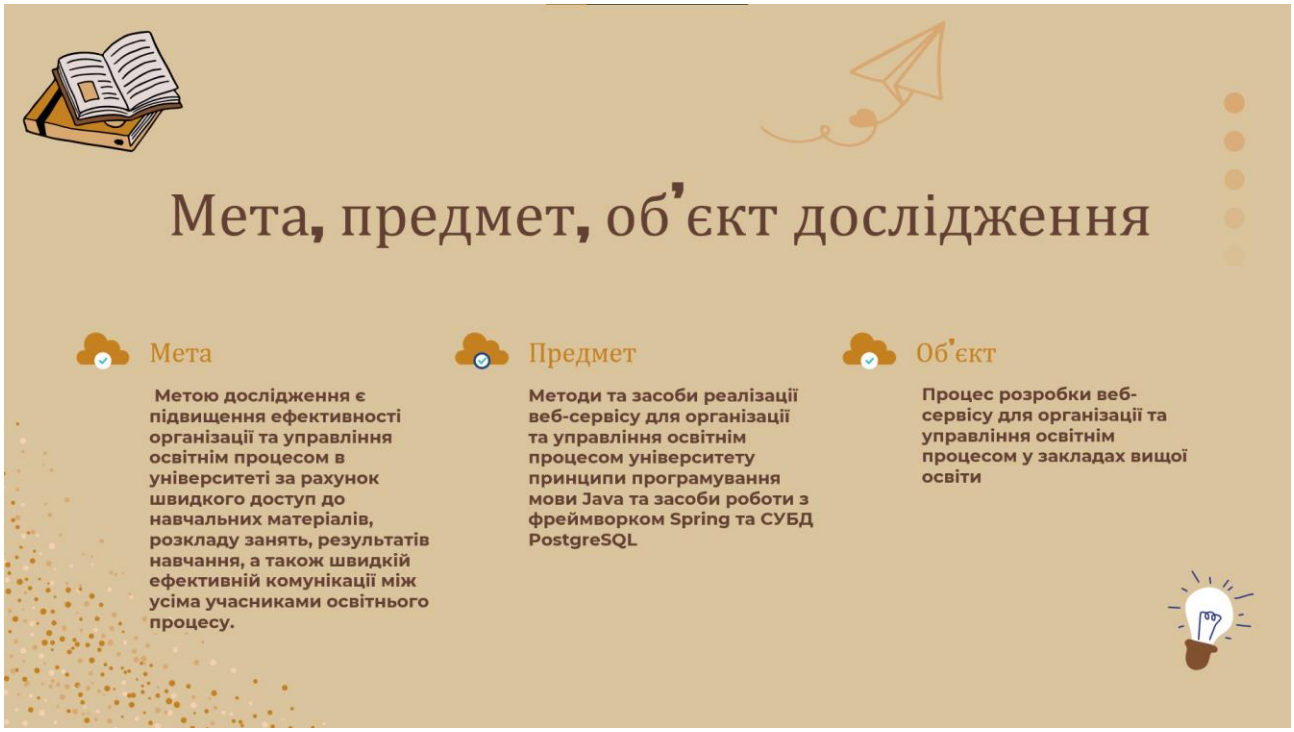


Рисунок Г.3 – Мета, об’єкт та предмет дослідження

Рисунок Г.4 – Задачі дослідження

Рисунок Г.5 – Наукова новизна

Рисунок Г.6 – Практична цінність

Критерії	Moodle	Google Classroom	Blackboard Learn	Canvas LMS	Open edX	Власна розробка
Простий і інтуїтивно зрозумілий інтерфейс	0	1	0	1	0	1
Отримання інформації про користувача	1	0	1	1	1	1
Авторизація та реєстрація	1	1	0	1	1	1
Перегляд розкладу користувача	0	0	1	1	0	1
Тестування програмування забезпечення	0	0	1	0	0	1
Сумарний коефіцієнт	2	2	3	4	2	5

Рисунок Г.7 – Порівняння з аналогами

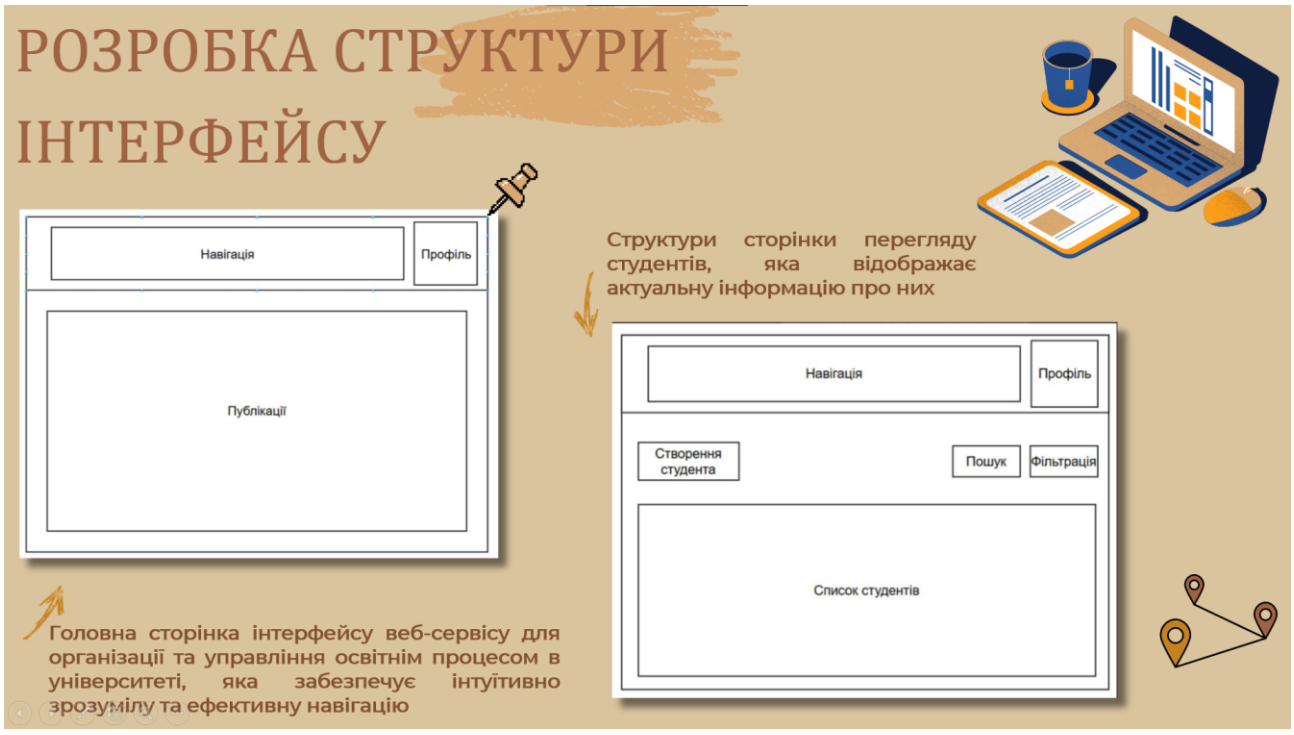


Рисунок Г.8 – Розробка структури інтерфейсу

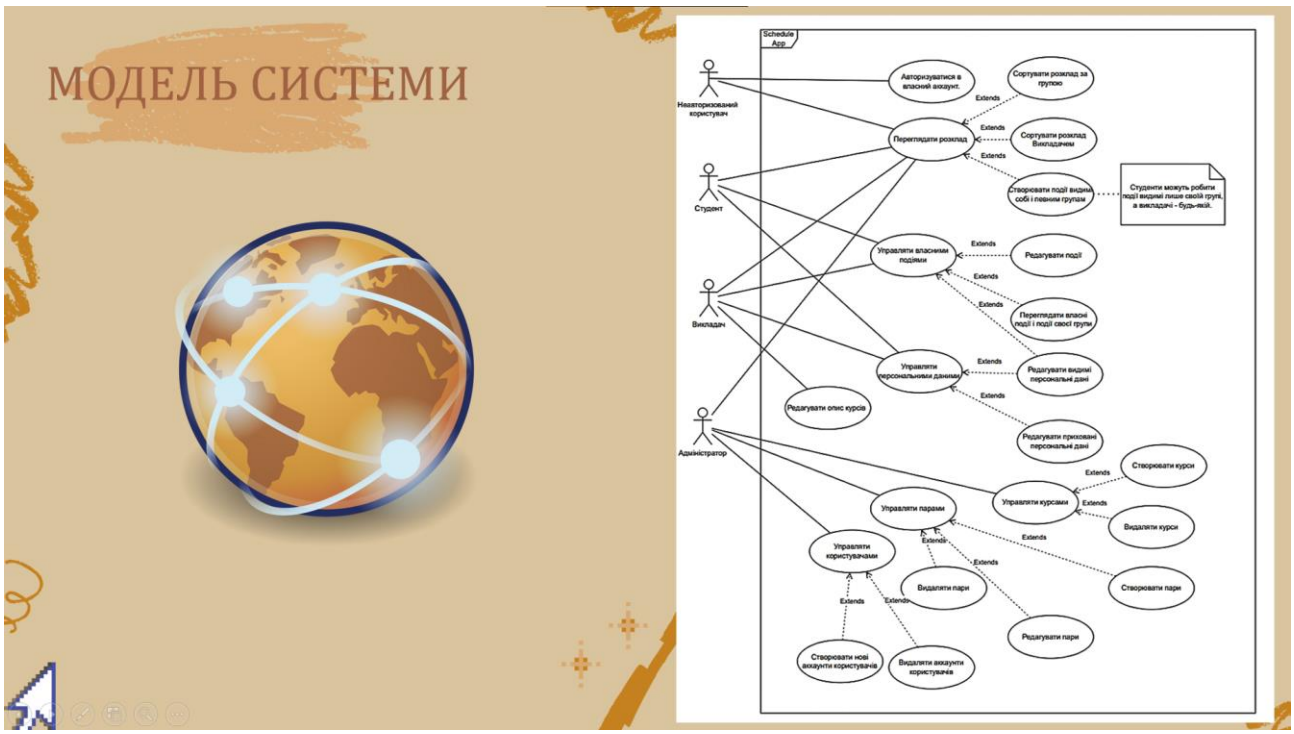


Рисунок Г.9 – Модель роботи вебсервісу для організації та управління освітнім процесом в університеті

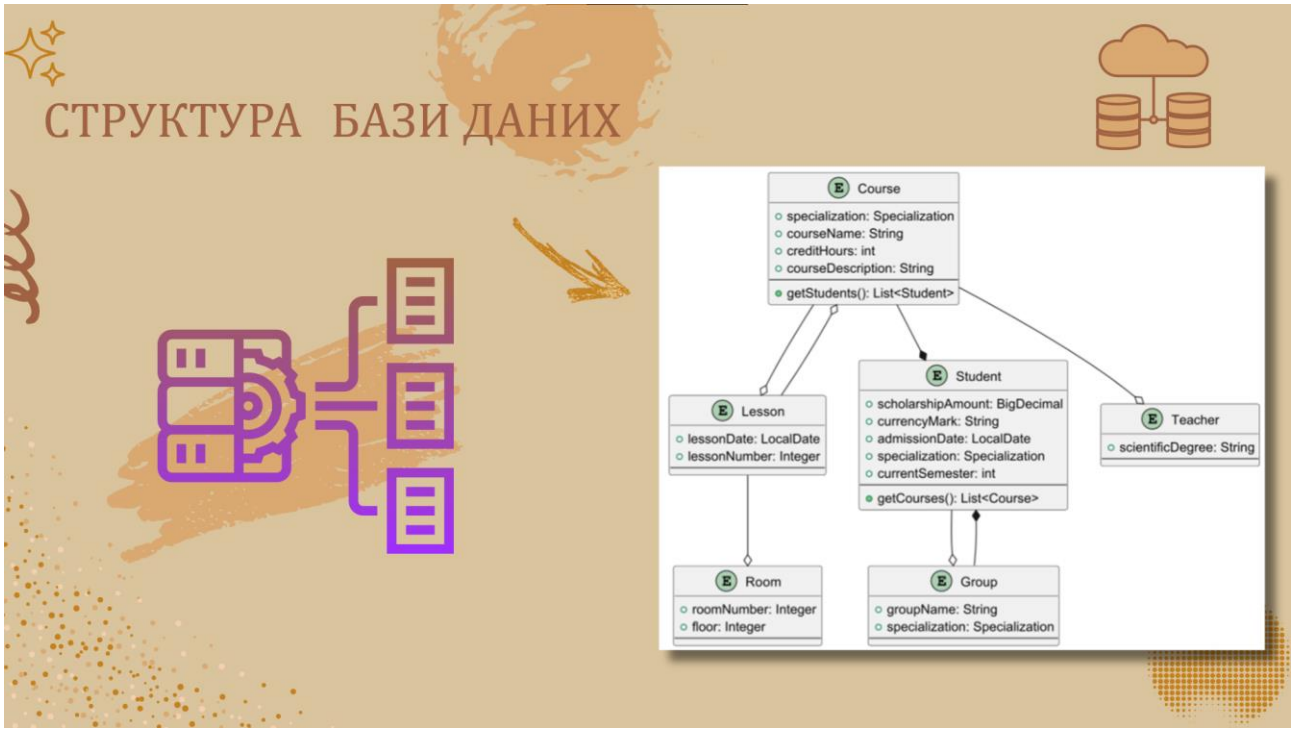


Рисунок Г.10 – Структура бази даних

Рисунок Г.11 – Використані технології

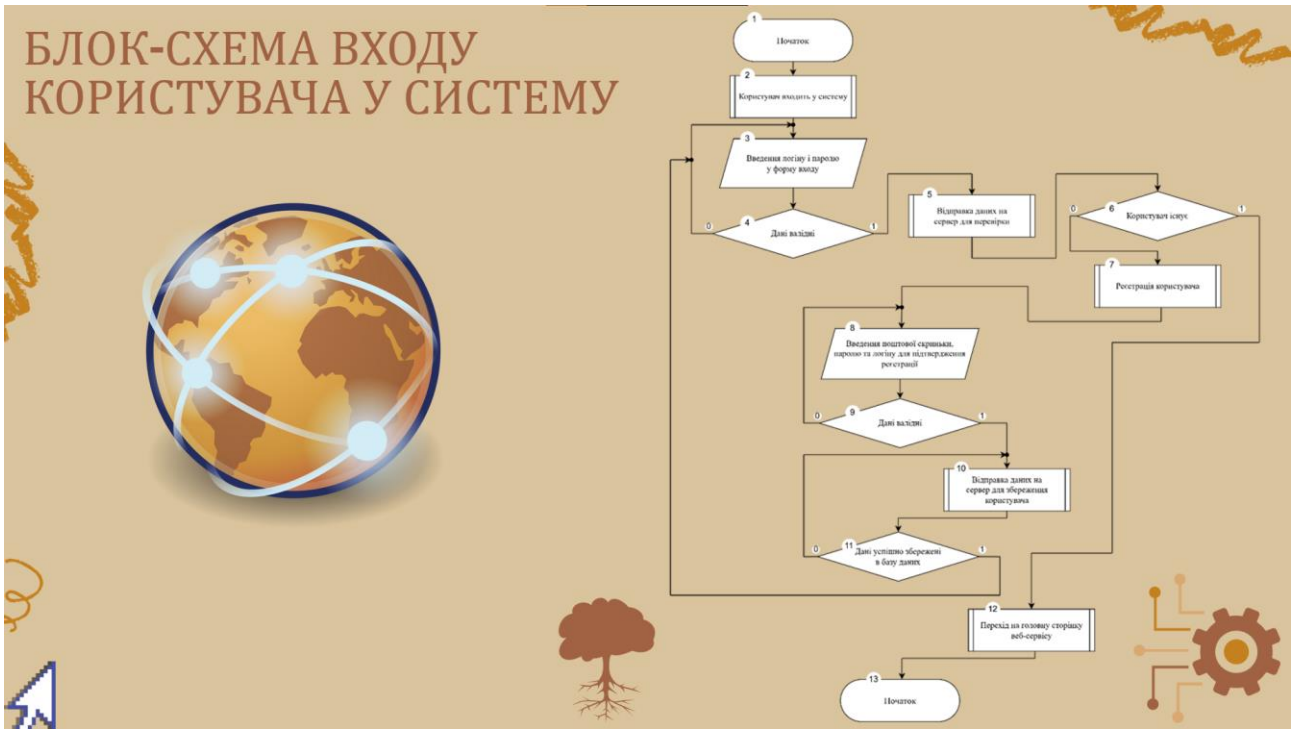


Рисунок Г.12 – Блок-схема входу користувача у систему

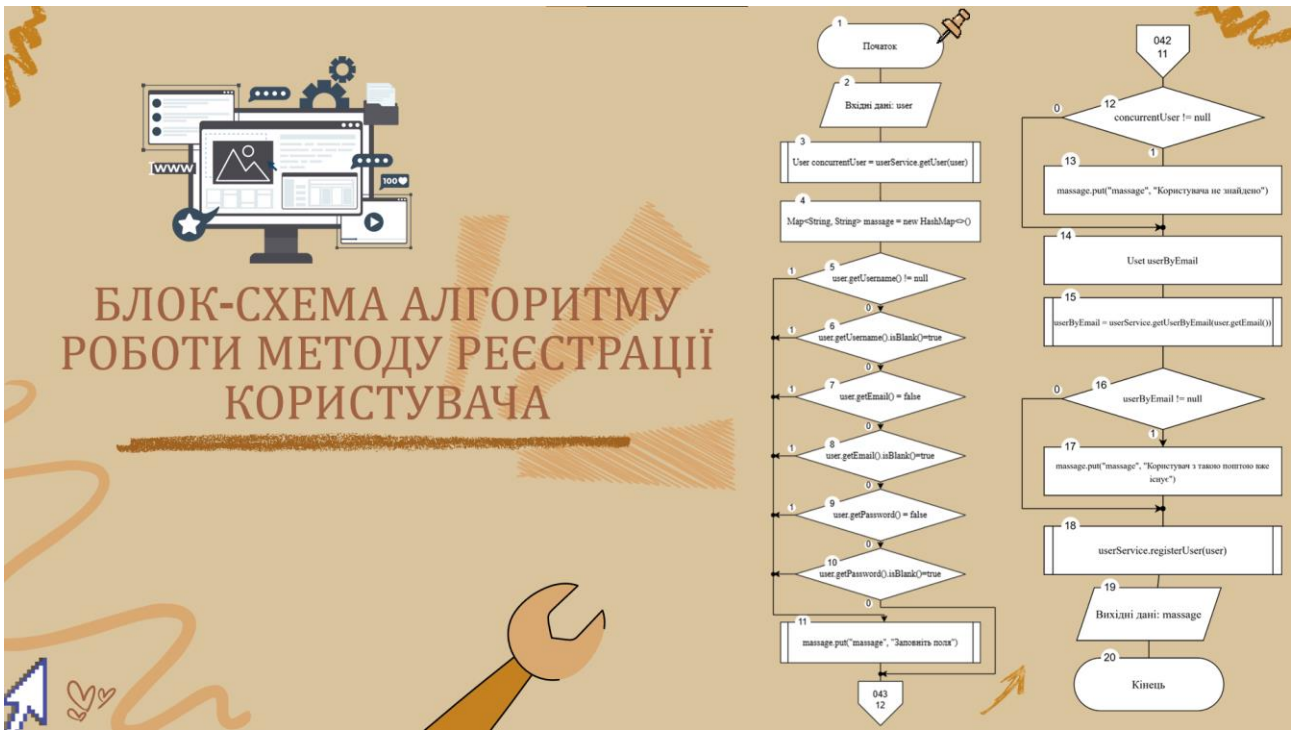


Рисунок Г.13 – Блок-схема алгоритму роботи методу реєстрації користувача

Рисунок Г.14 – Блок-схема алгоритму роботи методу пошуку студента за логіном

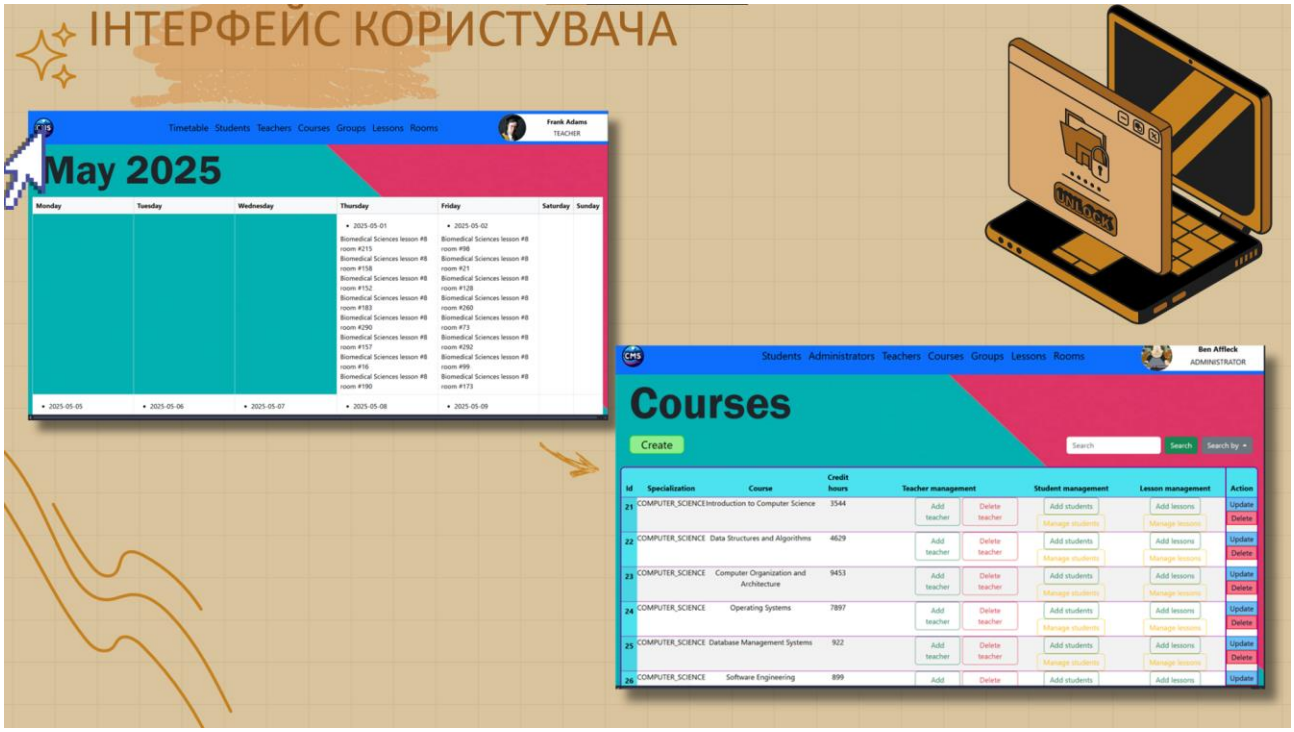


Рисунок Г.15 – Інтерфейс користувача

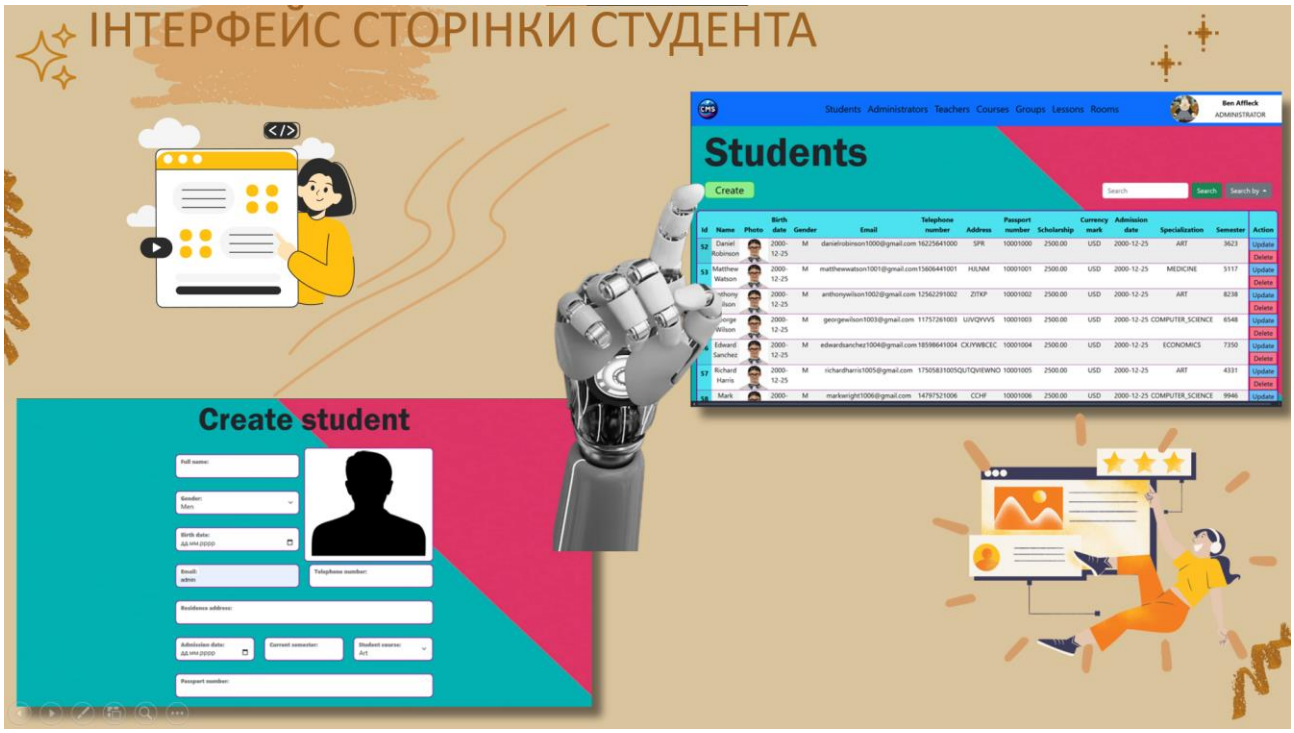


Рисунок Г.16 – Інтерфейс користувача

Рисунок Г.17 – Апробація матеріалів кваліфікаційної роботи

Рисунок Г.18 – Висновки