МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних інформаційних систем, д-р. техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

АВТОМАТИЗОВАНА СИСТЕМА РЕЙТИНГУВАННЯ СТУДЕНТІВ ЧНУ ІМЕНІ ПЕТРА МОГИЛИ

Спеціальність 122 «Комп'ютерні наук»

122 – БКР – 402.21910216

Виконав студент 4-го курсу, групи 402
_____ М.С.Мельничук
«20» червня 2023 р.

Керівник: канд. фіз.-мат. наук, доцент
_____І.В.Кулаковська
«20» червня 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р. техн. наук, проф.
_____ Ю. П. Кондратенко
«___» _____ 20___ р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук Мельничуку Максиму Сергійовичу.

1. Тема кваліфікаційної роботи «Автоматизована система рейтингування студентів ЧНУ імені Петра Могили».

Керівник роботи Кулаковська Інесса Василівна, канд. фіз.-мат. наук.

Затв. наказом Ректора ЧНУ ім. Петра Могили від « 17 » березня 20 23 р. № 59

2. Строк представлення кваліфікаційної роботи студентом « 19 » червня 20 23 р.

Очікуваний результат: автоматизована система оцінювання навчальних досягнень студентів університету і визначення їх рейтингу.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– аналіз предметної області оцінювання та рейтингування студентів університету;

– огляд застосовуваних на даний момент засобів визначення рейтингу студентів;

– огляд обраних технологій і програмних засобів для розробки вебсистеми;

– опис розробки інформаційної системи та її роботи.

5. Перелік графічного матеріалу: презентація, 52 рисунки, 7 таблиць.

6. Завдання до спеціальної частини: «Охорона праці при роботі з комп'ютерною технікою».

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Боженко А.Л. викладач кафедри екології	

Керівник роботи канд. фіз.-мат. наук Кулаковська І.В.
(наук. ступінь, вчене звання, прізвище та ініціали)

(підпис)

Завдання прийнято до виконання Мельничук М.С.
(прізвище та ініціали)

(підпис)

Дата видачі завдання «_23_» _____листопада____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

виконання бакалаврської кваліфікаційної роботи

Тема: Автоматизована система рейтингування студентів ЧНУ імені Петра Могили

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	28.10.2022	28.10.2022	Виконано
2	Отримання завдання на виконання БКР	25.11.2022	25.11.2022	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	09.12.2022	09.12.2022	Виконано
4	Отримання завдання на переддипломну практику	01.05.2023	01.05.2023	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	01.05.2023	14.05.2022	Виконано
6	Розробка звіту з переддипломної практики	15.05.2023	17.05.2023	Виконано
7	Виконання БКР: розробка бази даних, створення вебсторінок, програмування вебсистеми	15.05.2023	19.06.2023	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	29.05.2023	30.05.2023	Виконано
9	Доробка та остаточне оформлення БКР	02.06.2023	19.06.2023	Виконано
10	Подання БКР рецензенту	15.06.2023	17.06.2023	Виконано
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	19.06.2023	22.06.2023	
12	Захист БКР перед екзаменаційною комісією (ЕК)	26.06.2023	29.06.2023	

Розробив студент Мельничук М.С. _____

(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи канд. фіз.-мат. наук Кулаковська І.В. _____

(*посада, прізвище, ім'я, по батькові)* *(підпис)*

« 9 » 12 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 402 ЧНУ ім. Петра Могили

Мельничука Максима Сергійовича

Тема: «Автоматизована система рейтингування студентів ЧНУ імені Петра Могили»

Актуальність роботи: автоматизація процесу обліку успішності студентів робить його простішим і зручнішим, особливо в умовах дистанційного навчання.

Об'єкт роботи – процес створення вебзастосунку з використанням фреймворку ASP.NET Core.

Предмет роботи – засоби для розробки автоматизованої системи рейтингування студентів університету.

Мета роботи: автоматизація процесу обліку навчальних досягнень студентів ЧНУ імені Петра Могили.

Пояснювальна записка до бакалаврської кваліфікаційної роботи складається зі вступу, чотирьох розділів, висновків та додатків.

У першому розділі аналізується порядок оцінювання і визначення рейтингових балів студентів університету.

У другому розділі описуються використані при розробці технології, фреймворки та програмні засоби.

У третьому розділі представлено розробку вебзастосунку.

У четвертому розділі наведено опис роботи вебзастосунку.

Бакалаврська кваліфікаційна робота містить 86 сторінок, 52 рисунки, 7 таблиць, 30 використаних джерел та 3 додатки.

Ключові слова: вебзастосунок, рейтинг студентів, ASP.NET Core, Entity Framework

ABSTRACT

for bachelor's qualification work of a student of 402 group at Petro Mohyla Black Sea National University

Melnychuk Maksym Serhiiovych

Topic: «Automated student rating system for Petro Mohyla BSNU»

Relevance of work: automation of the assessment of students' performance and calculation of student rating simplifies this process and makes it more convenient, especially in distance learning conditions.

The object of work is the process of creation an web-application using ASP.NET Core framework.

The subject of work is tools for developing an automated web-system for calculation rating of university students.

The purpose of work is automation of the assessment of students' performance and calculation of student rating.

Explanatory note to the bachelor's qualification work contains introduction, four sections, conclusions and appendices.

In the first section, the area of assessment of students' academic performance and calculation of their rating is analyzed.

In the second, the used technologies, frameworks and programs are described.

In the third section, the web-application development is described.

The last section describes how the web-application works.

Bachelor's qualification work contains 86 pages, 52 pictures, 7 tables, 30 references and 3 appendices.

Keywords: web-application, students' rating, ASP.NET Core, Entity Framework.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних;

БКР – бакалаврська кваліфікаційна робота;

СКБД – система керування базами даних;

ЧНУ – Чорноморський національний університет;

API – Application Programing Interface;

ASP.NET – Active Server Pages для .NET;

CSS – Cascading Style Sheets;

EF – Entity Framework;

HTML – HyperText Markup Language;

HTTP – HypertText Transfer Protocol;

IDE – integrated development envilonment;

LINQ – language integrated query;

MVC – Model-View-Controller;

ORM – object-relational mapping;

SPA – Single Page Application;

SQL – Structured query language;

TPC – table per class;

TPH – table per hierarchy;

TPT – table per type.

ВСТУП

Визначальною рисою сьогодення стало впровадження інформаційних технологій практично в усіх сферах життя людини. Доступність різних сервісів онлайн, можливість будь-коли і будь-звідки мати доступ до даних і операцій, що раніше були тільки «на папері», робить життя комфортнішим, а роботу – більш ефективною, крім того, з огляду на події останніх років, є вимушеною необхідністю в деяких випадках. Підходячи ближче до теми роботи, слід зазначити, що освіта, зокрема вища, може відбуватись в дистанційному форматі, що ускладнює процес оцінювання успішності студентів у навчанні. Для студентів же було б зручно мати можливість переглядати результати своєї роботи протягом семестру, такі як бали, рейтинг і обсяг стипендії, актуальні на даний момент.

Метою даної роботи є автоматизація процесу обліку навчальних досягнень студентів університету, а саме – розробка автоматизованої вебсистеми рейтингування студентів Чорноморського національного університету імені Петра Могили.

Об’єктом роботи є процес створення вебзастосунку з використанням фреймворку ASP.NET Core.

Предметом роботи є засоби для розробки автоматизованої системи рейтингування студентів університету.

Для досягнення мети необхідно спершу проаналізувати предметну область – оцінку успішності студентів університету та виведення рейтингового списку на її основі, після чого, обравши технології і фреймворки для розв’язання задачі розробити вебсистему, що дозволятиме відстежувати успішність студентів, формувати рейтингові списки на її основі, надаватиме студентам можливість переглядати свої бали і рейтинг, а викладачам – виставляти бали зі своїх навчальних дисциплін. Також розроблена вебсистема повинна надавати можливість аналізу успішності студентів університету.

1 АНАЛІЗ ПРАВИЛ ОЦІНЮВАННЯ І ФОРМУВАННЯ РЕЙТИНГОВИХ СПИСКІВ СТУДЕНТІВ. ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ.

1.1 Огляд теми бакалаврської кваліфікаційної роботи

Тема бакалаврської кваліфікаційної роботи «Автоматизована система рейтингування студентів ЧНУ імені Петра Могили» передбачає розробку вебсистеми, що дозволяла б користувачам-студентам стежити за своєю успішністю в навчанні переглядаючи бали з різних навчальних дисциплін та рейтинговий список, користувачам-викладачам виставляти та переглядати бали зі своїх дисциплін, список боржників, тощо, відповідним посадовим особам університету отримувати дані про успішність в навчанні і рейтинг студентів для призначення стипендій або визначення списків студентів на відрахування. Така система має передбачати збереження даних, що стосуються користувачів системи та успішності студентів університету, у базі даних і розмежування прав доступу до цих даних відповідно до ролі користувача, належності його до певної групи і встановлених дозволів.

1.2 Аналіз предметної сфери

1.2.1 Користувачі системи

Система рейтингування студентів передбачає наявність трьох груп користувачів:

– студенти;

– викладачі;

– працівники університету, яким відповідно до посадових обов’язків необхідний доступ до даних про успішність студентів.

Також як окрему категорію можна виділити адміністраторів вебсистеми.

Студенти повинні мати доступ тільки до перегляду своїх балів і рейтингового списку своєї групи, викладачі повинні мати можливість виставляти бали зі своїх дисциплін і переглядати їх. Також викладач може бути куратором групи, тоді він матиме доступ до усіх балів студентів цієї групи. Інші можливості в системі, які можуть мати певні її користувачі, наступні:

– виставляти додаткові бали студентам (додаткові бали впливають на рейтинговий бал, що буде описано далі);

– переглядати всі бали та рейтингові списки студентів певного факультету;

– додавати і редагувати навчальні дисципліни;

– змінювати розмір стипендії та відсоток студентів, які її отримують;

– змінювати форму отримання освіти з контрактної на бюджетну.

Адміністратори вебсистеми повинні мати наступні можливості:

– додання, редагування і видалення даних користувачів, груп, спеціальностей, факультетів, кафедр;

– редагування дозволів користувачів, які визначають можливості користувача в системі.

Кожен користувач системи повинен мати логін і пароль для входу, ім’я, роль та дозволи для визначення прав користувача в системі, а також контактну інформацію (телефон, електронна пошта, тощо).

1.2.2 Студенти, студентські групи та спеціальності

Окрім властивостей, наявних в кожного користувача, студент також характеризується номером залікової книжки (унікальне значення, що формується на основі року початку навчання, номера групи на початку навчання і номера студента в групі на початку навчання), номером групи, формою навчання (контракт

або за державним замовленням – бюджет), наявністю і типом стипендії: академічна (за успіхи у навчанні), академічна підвищена, соціальна [1-3].

Студентські групи характеризуються номером групи, номером курсу, спеціальністю, факультетом, формою здобуття вищої освіти. Спеціальність групи може бути одною із переліку, затвердженого Кабінетом Міністрів України [4]. Форми здобуття вищої освіти можуть бути наступні: очна (денна), вечірня і заочна.

Номер групи формується на основі номера курсу, номера спеціальності та порядкового номера групи [5]. Таким чином кожного наступного навчального року група матиме інший номер, тому в якості унікального ідентифікатора групи краще використовувати значення, що включає рік формування групи, наприклад група, сформована в 2019 році з номером «102» матиме ідентифікатор «219102». Група також має закріпленого за нею куратора – викладача, відповідального за організаційну і виховну роботу, пов’язану з групою.

1.2.3 Викладачі, кафедри та факультети

Окрім спільних для всіх користувачів характеристик, викладачі також мають посаду (наприклад «старший викладач кафедри ІІС»), вчений ступінь (наприклад «кандидат технічних наук») та кафедру. Кафедри в свою чергу належать до певного факультету. Кожна кафедра має завідувача кафедри, а кожен факультет – декана факультету. В університеті можуть бути не лише факультети, а й інститути, проте для розроблюваної системи принципової відмінності немає.

1.2.4 Навчальні курси

Успішність студента оцінюється у вигляді оцінки із кожного навчального курсу, пройденого протягом семестру. При цьому курс може мати одну з трьох форм підсумкового контролю: залік, іспит або атестація[6]. Кількість навчальних

годин, передбачених курсом і вплив його оцінки на рейтинговий бал студента визначається кількістю кредитів. Відповідно до Європейської кредитної трансферно-накопичувальної системи (ECTS), запровадженої у вищих навчальних закладах України, кредит ECTS становить 30 навчальних годин[1].

Окремим різновидом навчального курсу можна також вважати практику, яку проходять студенти в кінці кожного курсу, зазвичай протягом двох тижнів. Практика оцінюється балами так само, як і навчальні дисципліни, оцінку виставляє керівник практики від університету, проте ця оцінка не впливає на рейтинговий бал, за тим винятком, що борг із практики є такою ж академічною заборгованістю, що й борг з навчальної дисципліни.

1.2.5 Бали студентів

Успішність студента з кожної навчальної дисципліни оцінюється в кінці семестру за 100-бальною шкалою, виняток складають спеціальності 222 «Медицина» та 226 «Фармація, промислова фармація», успішність студентів яких, натомість, оцінюється за 200-бальною шкалою[6,7]. При цьому окрім згаданих шкал існують ще дві шкали: шкала ECTS та національна 4-бальна шкала. Відповідність між ними та 100(200)-бальною шкалою наведена в табл. 1.1, 1.2[6,7]:

Таблиця 1.1 – Відповідність між балами в ECTS, національній та 100-бальній шкалі

<table>
<tr><td rowspan="2">Оцінка у 100-бальній шкалі</td><td rowspan="2">Оцінка ECTS</td><td colspan="2">Оцінка за національною шкалою</td></tr>
<tr><td>іспит</td><td>залік</td></tr>
<tr><td>90-100</td><td>A</td><td>відмінно</td><td rowspan="5">зараховано</td></tr>
<tr><td>82-89</td><td>B</td><td rowspan="2">добре</td></tr>
<tr><td>75-81</td><td>C</td></tr>
<tr><td>67-74</td><td>D</td><td rowspan="2">задовільно</td></tr>
<tr><td>60-66</td><td>E</td></tr>
<tr><td>35-59</td><td>FX</td><td rowspan="2">незадовільно</td><td rowspan="2">не зараховано</td></tr>
<tr><td>1-34</td><td>F</td></tr>
</table>

Таблиця 1.2 – Відповідність між балами в ECTS, національній та 200-бальній шкалі

Оцінка у 200-бальній шкалі	Оцінка ECTS	Оцінка за національною шкалою	
		іспит	залік
180-200	A	відмінно	зараховано
160-179	B	добре	
150-159	C		
130-149	D	задовільно	
120-129	E		
70-119	FX	незадовільно	не зараховано
1-69	F		

При оцінюванні роботи студента з певної дисципліни протягом семестру обов'язковим є переведення оцінки зі 100-бальної (або 200-бальної) системи в шкалу ECTS та національну шкалу[6].

У разі якщо навчальний курс передбачає наявність підсумкового контролю, то отримана студентом оцінка складається з двох компонентів: оцінка за роботу протягом семестру і оцінка за підсумковий контроль (табл.1.3) [7].

Таблиця 1.3 – Компоненти підсумкової оцінки

Вид підсумкового контролю		Максимальна кількість балів	
		за роботу у семестрі	за підсумковий контроль
за 100-бальною шкалою	Іспит	60	40
	Залік	70	30
	Атестація	100	-
Усі види підсумкового контролю за 200-бальною шкалою		120	80

Для оцінювання успішності студентів викладачеві видається відомість, що містить наступні графи, заповнені співробітниками деканату:

- назва факультету (інституту);
- номер групи;
- навчальний рік;

– номер семестру;
– курс, шифр академічної групи;
– назва навчальної дисципліни;
– дата проведення екзамену;
– форма семестрового контролю;
– загальна кількість кредитів і годин з дисципліни за семестр;
– вчене звання, посада, прізвище, ініціали викладача, який виставляє підсумкову оцінку;
– вчене звання, посада, прізвище, ініціали викладача, який здійснював поточний контроль протягом семестру, проводив практичні (семінарські, лабораторні) заняття;
– номер студента у списку, його прізвище та ініціали, номер залікової книжки;
– порожні графи для оцінки і дати, які заповнюються викладачем-екзаменатором [6].

Атестаційна відомість містить аналогічні дані, за винятком викладача-екзаменатора, бо в разі атестації підсумковий контроль не проводиться.

Окрім балів, що відображають успішність у вивченні певних навчальних дисциплін, студенти також можуть отримувати додаткові бали за громадську діяльність, наукову діяльність або спортивні досягнення. Сума таких балів не може складати більше 10 для студентів, що одержують оцінки за 100-бальною системою або 20 для студентів, що одержують оцінки за 200-бальною (останні за аналогічну активність одержують вдвічі більше балів, ніж перші). Для нарахування балів необхідний документ, що підтверджує підстави їх нарахування. Перелік критеріїв, за якими призначаються додаткові бали, наведено у табл.1.4 [9].

Таблиця 1.4 – Перелік критеріїв, за якими студентам призначаються додаткові бали

Тип	Назва		Бал
громадська діяльність	Участь у громадських заходах		0,3
	Організація громадського заходу		0,5
	Староста групи, асоційований член Студентської колегії		0,3
	Члени студентської колегії, профорги		0,5
	Голова профспілкового комітету		1
	Голова студентської колегії		1
	Керівник громадського заходу на рівні міста, району, області		до 1
	Учасник громадського заходу на рівні міста, району, області		до 0,3
	Членство в громадських організаціях		0,1
	Активна творча діяльність на рівні університету, міста, району, області		до 0,5
	Участь у роботі Культурно-мистецького центру		0,25
наукова робота	Доповідь на науковій конференції університету		1
	Доповідь на Всеукраїнській, міжнародній конференції (очна)		2
	Доповідь на Всеукраїнській, міжнародній конференції (заочна)		1
	Патенти		до 7
	Міжнародні олімпіади	участь	5
		І, ІІ, ІІІ місця	10
	Загальноукраїнські олімпіади	участь	3,5
		І, ІІ, ІІІ місця	7
	Конкурси наукових робіт	участь	5
		І, ІІ, ІІІ місця	10
	Наукові змагання різного виду	участь	4
		І, ІІ, ІІІ місця	8

Продовження таблиці 1.4

наукова робота	Участь у наукових темах кафедри (з урахуванням кількості учасників серед студентів)		теми, зареєстовані в УКРЕНТЕІ	до 3
			бюджетні та госпрозрахункові теми	до 5
	Стаття в збірці наукових робіт			2
	Стаття в фахових виданнях			4
	Статті в закордонних виданнях			5
	Статті в закордонних виданнях, що внесені до науково-метричних баз			7
	Статті в Scopus			10
спортивна діяльність	спортивні змагання	Університетські	участь	0,1
			І, ІІ, ІІІ місце	0,25
		Міські	участь	0,5
			І, ІІ, ІІІ місце	1,5
		Обласні	участь	1,5
			І, ІІ, ІІІ місце	3
		Республіканські	участь	3
			І, ІІ, ІІІ місце	6
		Міжнародні	участь	5
			І, ІІ, ІІІ місце	10

1.2.6 Призначення стипендії

Стипендії можуть призначатись студентам, що навчаються за державним замовленням за денною формою здобуття вищої освіти за результатами навчання. Стипендія може бути двох типів: академічна і соціальна. Перша призначається студентам відповідно до поточного рейтингу успішності студентів, який формується за результатами навчання та діяльності студентів протягом останнього семестру (якщо це перший семестр першого курсу - на підставі конкурсного бала,

отриманого при вступі). Друга призначається студентам, що належать до певних пільгових категорій, у тому випадку, якщо вони не мають заборгованостей з жодної навчальної дисципліни (виплата стипендії поновлюється з наступного місяця після ліквідації заборгованості, якщо така була) [8].

Перед початком підведення підсумків кожного семестрового контролю визначається однаковий для всіх факультетів (інститутів), курсів та спеціальностей ліміт стипендіатів, яким буде призначатися академічна стипендія за результатами семестрового контролю. Цей ліміт встановлюється у відсотках до фактичної кількості студентів денної форми навчання, які навчаються за державним замовленням на певному факультеті (інституті), курсі за певною спеціальністю. Академічна стипендія призначається першим n студентам із рейтингового списку, де n – кількість студентів денної форми навчання, які навчаються за державним замовленням за певною спеціальністю помножена на встановлений ліміт стипендіатів [8].

Академічні стипендії, що призначаються за рейтинговим списком також називають «звичайні» або «ординарні»[8,9]. Також можуть призначатись академічні стипендії у підвищеному розмірі: вони призначаються студентам, які досягли особливих успіхів у навчанні та студентам, які навчаються за спеціальностями (спеціалізаціями), визначеними переліком спеціальностей (спеціалізацій), для яких встановлюється підвищений розмір академічних стипендій [7]. Під особливими успіхами у навчанні зазвичай маються на увазі відмінні оцінки з усіх курсів, пройдених протягом останнього семестру, включно з предметами, формою підсумкового контролю яких є атестація (на рейтинговий бал оцінка з такого курсу не впливає).

1.2.7 Визначення рейтингу студентів

Рейтинговий список студентів формується зі студентів, що навчаються за державним замовленням, денної форми навчання певного курсу і спеціальності, тобто якщо на якомусь курсі є дві групи, що навчаються за однією спеціальністю, то для них формується один спільний рейтинговий список. До рейтингового списку не включаються студенти, які:

– протягом навчального семестру до початку поточного семестрового контролю з будь-якого предмета не набрали бал, необхідний для допуску до заліку або іспиту (відповідно 30 або 20), або не набрали 60 балів у випадку атестації;

– мають хоч одну академічну заборгованість;

– під час семестрового контролю повторно складали іспит або залік;

– не склали іспит або залік з будь-якої навчальної дисципліни або не отримали атестацію з будь-якого із предметів [7].

Для одержання рейтингового списку студентів для кожного студента розраховується рейтинговий бал за результатами успішності та показниками участі у науковій, науково-технічній діяльності, громадському житті та спортивній діяльності за наступною формулою (1.1) [10]:

$$C_{\text{загальний}} = 0{,}9 * C_{\text{зв.бал}} + \sum_{i=1}^{n} C_i\,; \quad \sum_{i=1}^{n} C_i \le C_{\text{макс.дод.бал}}\,, \tag{1.1}$$

де $C_{\text{загальний}}$ – загальний рейтинговий бал;

$C_{\text{зв.бал}}$ – середній зважений бал, що обраховується за результатами навчання студента;

C_i – і-й додатковий бал;

$C_{\text{макс.дод.бал}}$ – максимальна сума додаткових балів, що становить 10 для студентів, що отримують оцінки у 100-бальній системі та 20 для студентів, що отримують оцінки у 200-бальній системі;

n – кількість додаткових балів, нарахованих студенту протягом семестра.

Середній зважений бал, що відображає результати навчання студента, розраховується за наступною формулою (1.2) [10]:

$$C_{\text{зв.бал}} = \sum_{j=1}^{N} \lambda_j C_j \,, \tag{1.2}$$

де C_j – кількість балів j-ї дисципліни;

N – кількість дисциплін,

λ_j – ваговий коефіцієнт, що визначає вплив оцінки з даної дисципліни на загальний рейтинговий бал і розраховується за наступною формулою (1.3) [9]:

$$\lambda_j = \frac{K_j}{\sum_{j=1}^{N} K_j}\,, \tag{1.3}$$

де K_j – кількість кредитів j-ї дисципліни.

Важливо зазначити, що для розрахунку рейтингового балу беруться тільки оцінки з тих дисциплін, за якими наприкінці семестру було передбачено підсумковий контроль, тобто іспит або залік [10].

1.3 Огляд наявних аналогів систем рейтингування студентів в ЧНУ імені Петра Могили

Для оцінки успішності студентів в Чорноморському національному університеті імені Петра Могили використовується система дистанційного навчання MOODLE3[11], що дозволяє викладачам оцінювати роботу студентів протягом семестру перевіряючи і виставляючи оцінки до завантажених робіт. Недоліком з точки зору теми, яка розглядається в даній роботі, є те, що система не дозволяє підрахувати загальний рейтинг студентів – можна лише переглянути бали

всіх студентів з певної дисципліни або бали певного студента з усіх дисциплін (рис.1.1).

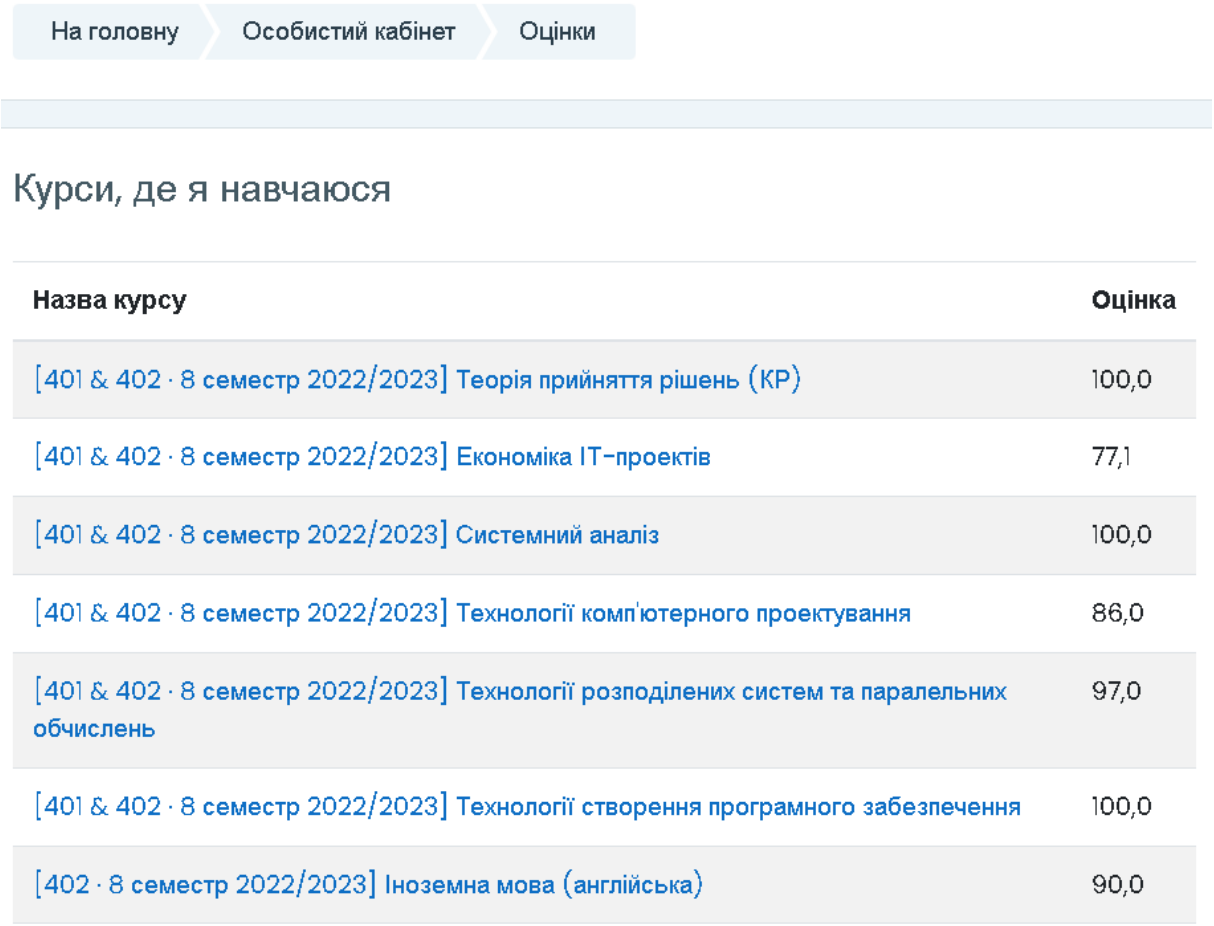


Рисунок 1.1 – Приклад відображення оцінок студента в системі дистанційної освіти MOODLE3 ЧНУ імені Петра Могили

Для підрахунку рейтингу студентів і формування рейтингових списків в ЧНУ імені Петра Могили зазвичай використовуються таблиці Microsoft Excel або Google Sheets. Недоліком такого підходу є те, що співробітникам деканату факультету доводиться вручну передруковувати бали студентів із відомостей, а студенти не завжди мають змогу побачити, які бали і яке місце в рейтингу вони мають на даний момент.

1.4 Основні вимоги до розроблюваної вебсистеми

На основі наведеного вище опису предметної області системи рейтингування студентів виділимо вимоги до розроблюваної системи:

1.Система повинна зберігати у базі даних наступні сутності (табл.1.5).

Таблиця 1.5 – Сутності, які система має зберігати у базі даних

Назва сутності	Поля
Користувач	– ідентифікатор; – логін; – пароль; – повне ім’я; – роль; – контактні дані; – дозволи;
Студент	– ідентифікатор користувача; – номер залікової книжки; – номер групи; – форма навчання (контракт чи бюджет); – наявність та тип стипендії;
Викладач	– ідентифікатор користувача; – посада; – вчений ступінь; – кафедра;
Працівник університету	– ідентифікатор користувача; – посада
Кафедра	– назва кафедри; – скорочена назва кафедри; – завідувач кафедри;

Продовження таблиці 1.5

Студентська група	– ідентифікатор групи; – ідентифікатор спеціальності; – куратор групи; – факультет;
Факультет	– назва факультету; – скорочена назва факультету; – декан факультету;
Навчальний курс	– ідентифікатор; – назва; – тип підсумкового контролю (залік/іспит/атестація); – семестр, в якому викладався курс; – кількість кредитів; – номер групи; – екзаменатор (викладач, що виставляє бали підсумкового контролю); – викладач протягом семестру; – початок і кінець (у випадку, якщо це практика);
Дозволи користувачів	– ідентифікатор; – назва;
Види стипендії	– ідентифікатор; – назва; – обсяг стипендії;
Спеціальність	– код спеціальності; – назва спеціальності; – бальна шкала для оцінки успішності студентів;

Продовження таблиці 1.5

Бал	– дата; – оцінка за 100/200-бальною шкалою; – студент; – курс; – чи була перездача;
Додатковий бал	– тип (за громадську діяльність, наукову роботу або спортивні досягнення); – дата нарахування; – опис; – бал; – документ, що підтверджує підстави нарахування балу; – студент;

2. Вебсистема має забезпечувати авторизацію користувачів, і обмежувати доступ до даних. Мати можливість переглядати оцінки з певної дисципліни повинні тільки викладач, який виставив оцінку, студент, якому її ви ставлено, і працівник університету, чиї посадові обов’язки передбачають доступ до даних про успішність студентів.

3. Система має на основі оцінок за останній семестр формувати рейтингові списки студентів, обраховуючи рейтинговий бал за формулами (1.1–1.3), визначати які студенти претендують на стипендію.

Висновки до розділу 1

В першому розділі БКР було розглянуто предметну область системи, що розробляється, а саме – порядок оцінювання і визначення рейтингу студентів ЧНУ імені Петра Могили. Було розглянуто нормативні документи, що визначають загальні питання щодо освітнього процесу, порядок оцінювання студентів, розрахунку рейтингового бала студента, призначення стипендії, тощо. На основі аналізу предметної сфери було розроблено основні вимоги до розроблюваної вебсистеми рейтингування студентів.

2 ЗАСОБИ РОЗРОБКИ ВЕБСИСТЕМИ

2.1 Архітектура програмного застосунку

Для розробки вебсистеми обрано архітектуру «база даних – сервер – клієнт» (рис.2.1).

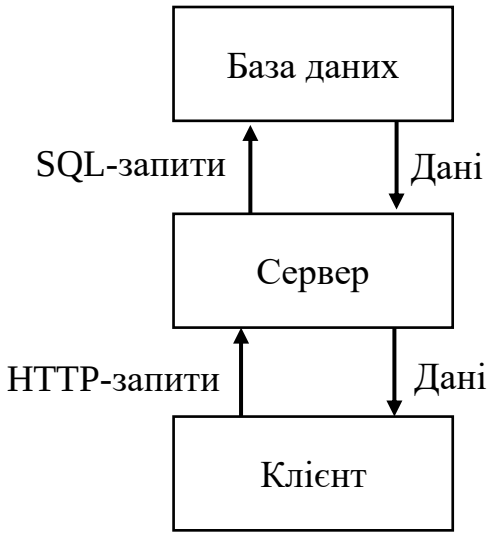


Рисунок 2.1 – Схематичне зображення архітектури «база даних-сервер-клієнт»

Вона полягає в тому, що під роботи вебзастосунку користувач взаємодіє з системою за допомогою вебінтерфейсу, надсилаючи HTTP-запити до сервера. На сервері відбувається аутентифікація та авторизація користувача, тобто визначається чи може він увійти до системи, яку роль і права доступу в ній має, може застосовуватись певна бізнес-логіка. Після цього сервер надсилає відповідний до запитаних користувачем даних запит до бази даних (SQL-запит, якщо використовується реляційна база даних), отримані дані надсилаються клієнту. Доступ до бази даних виключно через сервер запобігає можливості пошкодження даних, база даних забезпечує цілісність даних шляхом використання зовнішніх ключів і зв’язків між її таблицями. Ці три рівні архітектури вебсистеми ще називають рівень представлення, рівень логіки та рівень даних [12].

2.2 Архітектурний шаблон застосунку

Архітектурний шаблон програмного забезпечення – це ефективний спосіб вирішення поширених архітектурних проблем при розробці програмного забезпечення. Архітектурні шаблони подібні до шаблонів проектування, проте на відміну від них мають ширший масштаб: основними елементами структури архітектурних шаблонів є не класи та об’єкти, а умовні частини програмної системи.

В даній роботі при розробці вебсистеми буде використано фреймворк, який реалізує архітектурний шаблон MVC.

Модель-представлення-контролер (Model-View-Controller, MVC) – архітектурний шаблон, використовуваний при проектуванні та розробці програмного забезпечення. Цей шаблон передбачає умовний поділ програмного застосунку на три взаємопов’язані частини: модель даних, представлення даних (також називають вигляд або інтерфейс користувача) та контролер.

Він застосовується для відокремлення даних від інтерфейсу користувача так, щоб зміни інтерфейсу користувача мали мінімальний вплив на роботу з даними, а зміни в моделі даних могли здійснюватись без змін інтерфейсу користувача. Метою використання даного шаблону є гнучкий дизайн програмного застосунку, який полегшуватиме подальші зміни чи розширення програм, а також надаватиме можливість повторного використання окремих компонентів програми. Окрім того, використання цього шаблону сприяє впорядкованості структури системи, спрощує процес розробки застосунку, робить розроблену систему більш зрозумілою

Концепція шаблону MVC передбачає поділ застосунка на 3 частини:

– Модель: описує використовувані в додатку дані та логіку, пов’язану з ними, наприклад валідацію даних. Як правило, об’єкти моделей відповідають сутностям, що зберігаються у базі даних;

– Представлення (View): відповідають за графічний користувацький інтерфейс, представлення не має містити логіку обробки запиту користувача, проте може містити логіку, пов’язану з відображенням даних;

– Контролер: компонент MVC, що забезпечує зв’язок зі сховищем даних [13].

Відношення між компонентами шаблону MVC можна описати схемою, наведеною на рис.2.2 .

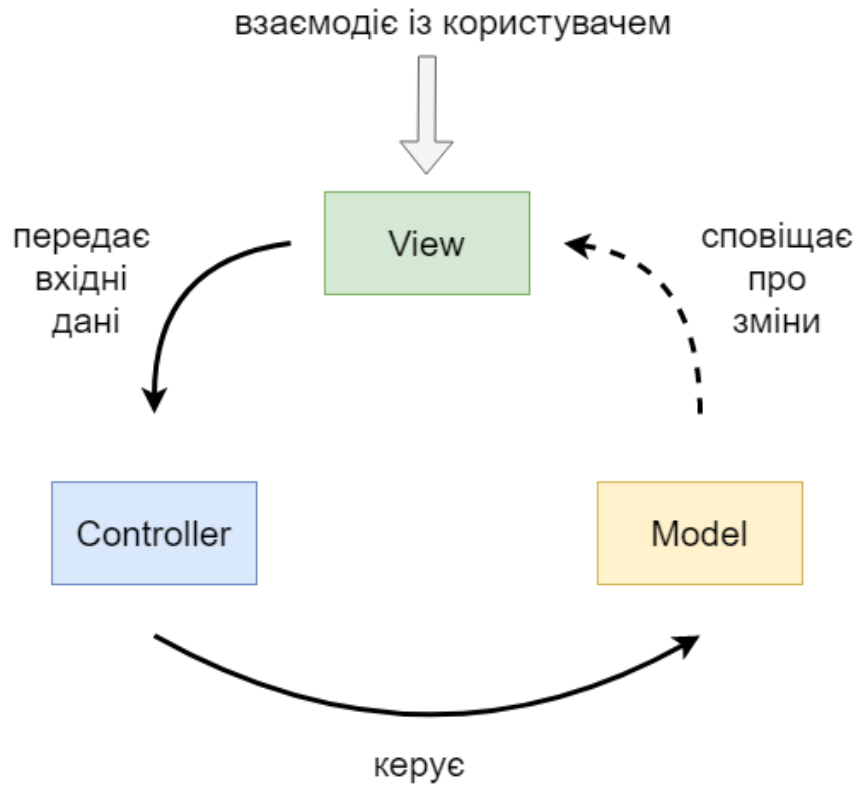


Рисунок 2.2 – Схема взаємодії між компонентами шаблону MVC

2.3 Платформа

ASP.NET – це безкоштовна вебплатформа, створена фахівцями Microsoft для проектування інтерактивних вебзастосунків, які працюють на платформі .NET. Платформа представляє технологію створення вебзастосунків із використанням

мов програмування C# та F#. Нинішній етап розвитку фреймворку – кросплатформна технологія ASP.NET Core, її поточна версія ASP.NET Core 7 випущена в листопаді 2022 року [13-15]. В основі роботи ASP.NET лежить принцип клієнт-серверної архітектури та протокол HTTP:

1) Спочатку клієнт надсилає запит;
2) ASP.NET приймає запит і починає його обробку;
3) Проходить ланцюжок обробників, які можуть виконувати різні завдання, наприклад перевірку авторизації або обробку даних форми;
4) ASP.NET виконує код на сервері, який може генерувати вебсторінки з використанням HTML, CSS, JavaScript для надсилання клієнту;
5) Згенерована сторінка відправляється назад клієнту [15,16].

Узагальнена схема роботи застосунку ASP.NET Core наведена на рис.2.3 .

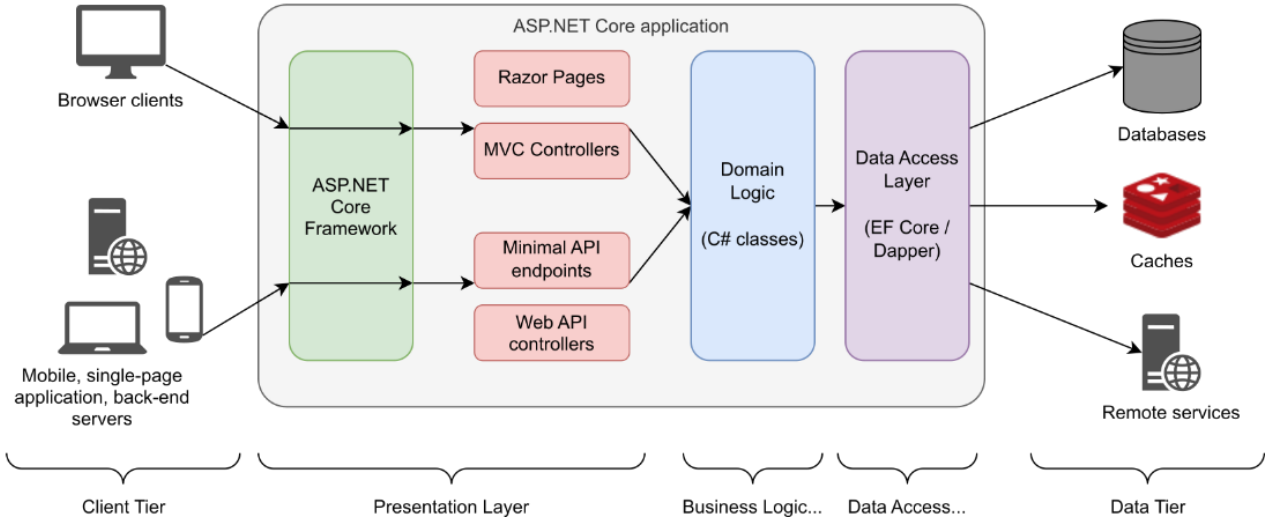


Рисунок 2.3 – Схема роботи застосунку ASP.NET Core

Загальну архітектуру поточної версії ASP.NET Core можна представити як наведено на рис.2.4.

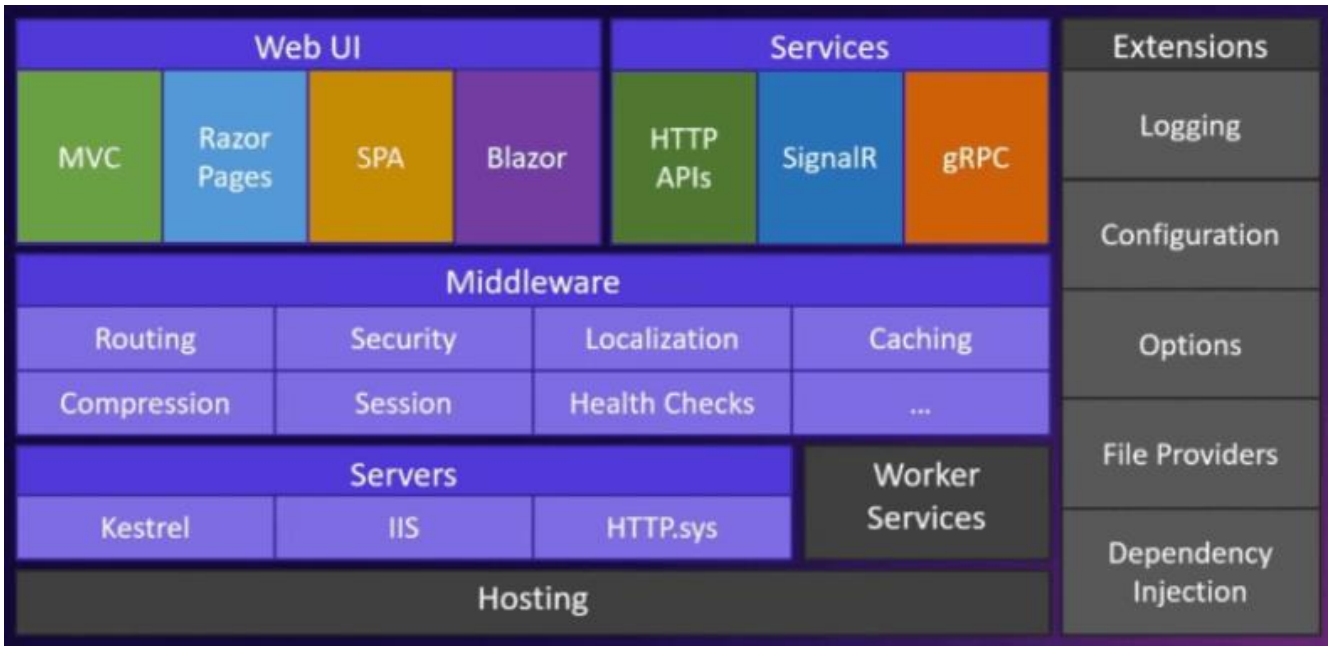


Рисунок 2.4 – Узагальнена архітектура платформи ASP.NET Core

На найвищому рівні розташовані різні моделі взаємодії застосунку з користувачем. Це технології побудови користувацького інтерфейсу і обробки введених користувачем даних, такі як фреймворк MVC, Razor Pages, SPA, Balzor. Окрім того, це сервіси в вигляді вбудованих HTTP API, бібліотеки SignalR або сервісів GRPC. Ці технології взаємодіють з чистим ASP.NET Core, представленим перш за все різними вбудованими компонентами middleware – компонентами, що застосовуються для обробки запиту. Крім того, технології верхнього рівня взаємодіють зі різноманітними розширеннями, що не є безпосередньою частиною ASP.NET Core. На найнижчому рівні застосунок ASP.NET Core працює в рамках певного вебсервера [13,14,16].

Суттєвою перевагою платформи .NET є можливість легко підключати усі необхідні для розробки пакети і бібліотеки за допомогою менеджера пакетів NuGet [14].

2.4 Інтегроване середовище розробки

В якості інтегрованого середовища розробки (IDE) в даній роботі використовуватиметься середовище розробки Microsoft Visual Studio 2022.

Visual Studio 2022 дозволяє підвищити ефективність розробки вебдодатків ASP.NET Core, адже підтримує всі необхідні для цього функції, включаючи засоби розробки та налагодження програм на .NET, засоби редагування коду HTML, JavaScript і CSS, засоби веброзробки із використанням сторінок Razor Pages, інструменти для зручної роботи з базами даних та SQL-кодом, вбудований диспетчер пакетів NuGet [17].

2.5 Клієнтська частина застосунку

Взаємодія автоматизованої системи з користувачем відбуватиметься за допомогою вебінтерфейсу. Вебінтерфейс – це сукупність засобів, за допомогою яких користувач взаємодіє з вебзастосунком через браузер. Найпоширенішим способом створення вебінтерфейсу є використання HTML-сторінок із застосуванням CSS для стилізації сторінки та JavaScript для динамічної зміни контенту сторінки. Зручним засобом оформлення вебсторінок є бібліотека Bootstrap – набір інструментів з відкритим кодом, призначений для створення вебсайтів, який містить шаблони CSS та HTML для форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript, він спрощує розробку вебсайтів і вебзастосунків [18].

Зручним інструментом створення користувацького інтерфейсу при розробці вебзастосунків на платформі ASP.NET Core є технологія Razor Pages. Razor Pages представляє альтернативу фреймворку MVC, будучи дещо простішим в плані організації вебсторінок [19].

Застосунок Razor Pages організований у вигляді сторінок Razor, кожна з яких представляє користувацький інтерфейс і пов’язану з ним логіку. Кожна сторінка

Razor представляє собою файл .cshtml, який містить html-код із включеними в нього конструкціями Razor. Синтаксис Razor заснований на мові програмування C# і дозволяє розміщувати блоки коду C# у HTML-коді. При роботі застосунку програма обробляє ці конструкції і генерує на їх основі HTML-код. Кожна сторінка Razor має свій клас моделі сторінки, в якому визначається логіка обробки запиту користувача [19-21]. Ще однією перевагою Razor Pages є те, що ця технологія передбачає ізоляцію CSS – це означає, що кожна сторінка може мати власну таблицю стилів css – файл .css, назва якого відповідає певним правилам і дозволяє співставити його зі сторінкою Razor. Винесення CSS-стилів, необхідних тільки для однієї сторінки, в окремий файл, пов’язаний зі сторінкою, полегшує процес розробки вебсистеми [22,23].

2.6 Взаємодія з базою даних

В якості системи керування базами даних, яка буде застосовуватись в даній роботі, обрано СКБД Microsoft SQL Server. SQL Server є реляційною системою керування базами даних. Це програмний продукт, основною функцією якого є зберігання та отримання даних за запитом інших програм, які можуть працювати на тому самому комп’ютері або на іншому комп’ютері в мережі. Мова, що використовується цією СКБД для запитів до бази даних Transact-SQL є реалізацією стандарту структурованої мови запитів SQL із розширеннями [24].

Взаємодію програмного застосунку із СКБД забезпечуватиме технологія Entity Framework Core. Entity Framework представляє ORM-технологію (технологію відображення даних на реальні об’єкти), розроблену компанією Microsoft, для доступу до даних. Ця технологія дозволяє абстрагуватися від самої бази даних, її таблиць та SQL-запитів, і працювати із даними як із об’єктами класів незалежно від того, яке сховище для даних використовується і яка СКБД задіяна. Якщо на фізичному рівні необхідно оперувати таблицями, індексами, первинними і зовнішніми ключами, то технологія Entity Framework піднімає процес розробки на

концептуальний рівень, де можна працювати з сутностями бази даних як з об’єктами. Технологія Entity Framework Core працює на платформі .NET і може використовуватись на різних платформах стеку .NET, зокрема і ASP.NET Core. Поточка версія EF Core – 7.0 була випущена в листопаді 2022 року разом із .NET 7. Entity Framework підтримує велику кількість різних СКБД, зокрема MS SQL Server, і дозволяє за потреби перейти на іншу СКБД практично без змін у коді, що отримує і відправляє дані в базу даних.

Центральною концепцією Entity Framework є сутність (entity). Сутність визначає набір даних, пов’язаних з певним об’єктом і на фізичному рівні відповідає рядку таблиці бази даних. Сутності можуть бути пов’язаними зв’язками один до одного, один до багатьох і багато до багатьох, Entity Framework пропонує різні способи встановлення зв’язків між сутностями.

Відмінною рисою Entity Framework Core як технології ORM є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ можна створювати різні запити на вибірку даних із бази даних, у тому числі пов’язані з різними асоціативними зв’язками. А Entity Framework при виконанні запиту транслює конструкцію LINQ в вираз, зрозумілий для конкретної СКБД (як правило, в SQL-запит) [25-29].

Entity Framework Core генерує таблиці бази даних на основі класів сутностей, проте також надає можливість редагувати таблиці бази даних в конструкторі або за допомогою SQL-запитів (рис.2.5).

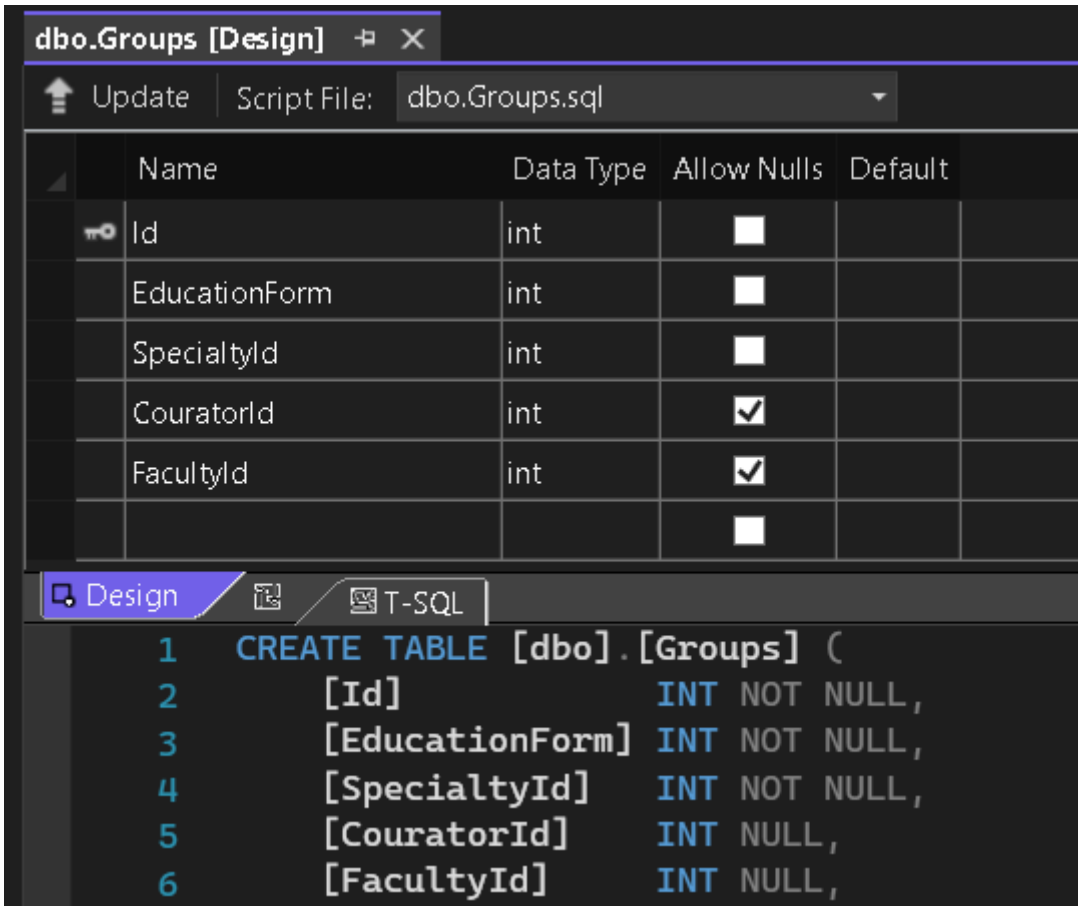


Рисунок 2.5 – Редагування таблиці з БД у Visual Studio 2022 SQL Server Object Explorer

Висновки до розділу 2

В другому розділі БКР було розглянуто використовувані для реалізації програмного застосунку методи, технології та програмні засоби, виділено їх особливості, важливі для даної роботи. Підсумовуючи, можна зазначити, що вебзастосунок матиме архітектуру «база даних – сервер – клієнт», розроблятиметься із використанням фреймворку ASP.NET Core 7, для взаємодії з базою даних використовуватиметься фреймворк Entity Framework, користувацький інтерфейс створюватиметься за допомогою сторінок Razor Pages. В якості інтегрованого середовища розробки використовується Microsoft Visual Studio 2022.

3 РОЗРОБКА ВЕБСИСТЕМИ РЕЙТИНГУВАННЯ СТУДЕНТІВ

3.1 Проектування і розробка бази даних

Формулюючи вимоги до розроблюваної системи було зазначено сутності, які необхідно зберігати в базі даних та їх параметри (табл.1.5). На основі цих вимог спроектуємо перелік таблиць, що мають бути в базі даних(табл.3.1):

Таблиця 3.1 – Перелік таблиць, які повинні бути в базі даних, з їх полями

Назва таблиці	Поля таблиці	Тип даних	Значення за замовчуванням	Особливості
Users	Id	int		primary key
	Name	string		not null
	Login	string		not null
	Role	int		not null
	Password	string	12345678	not null
	Contacts	string		
Students	Id	int		primary key
	StudentNumber	int		not null
	GroupId	int		foreign key, not null
	IsOnBudget	bool	false	not null
	ScholarshipTypeId	int		foreign key
Teachers	Id	int		primary key
	Post	string		
	AcademicDegree	string		
	DepartmentId	int		not null
Employees	Id	int		primary key
	Post	string		not null
Groups	Id	int		primary key
	EducationForm	int	0	not null
	SpecialtyId	int		foreign key, not null
	CouratorId	int		foreign key
	FacultyId	int		foreign key, not null

Продовження таблиці 3.1

Cources	Id	int		primary key
	Name	string		not null
	ShortName	string		
	FinalExamType	int		not null
	FinalExamTime	datetime		
	Semester	int		not null
	Year	int		not null
	Credits	int		not null
	GroupId	int		foreign key, not null
	TeacherId	int		foreign key, not null
	ExaminerId	int		foreign key, not null
Practices	Begin	datetime		
	End	datetime		
Marks	Id	int		primary key
	ExamDate	datetime		not null
	Point	int		not null
	IsRetaken	bool	false	not null
	StudentId	int		foreign key, not null
	CourceId	int		foreign key, not null
AdditionalPoints	Id	int		primary key
	Type	int		not null
	Date	datetime		not null
	Description	string		not null
	Point	float		not null
	StudentId	int		foreign key, not null
	ConfirmationFileName	string		

Продовження таблиці 3.1

Departments	Id	int		primary key
	Name	string		not null
	ShortName	string		
	FacultyId	int		foreign key, not null
	HeadOfDepartmentId	int		foreign key, not null
Faculties	Id	int		primary key
	Name	string		not null
	ShortName	string		
	DeanId	int		
Permissions	Id	int		primary key
	Name	string		not null
ScholarshipTypes	Id	int		primary key
	Name	string		not null
	Size	int		not null
Specialties	Id	int		primary key
	Name	string		not null
	PointScale	int	100	not null

Entity Framework Core надає можливість створення бази даних на основі підходу Code First, суть якого полягає в тому, що замість створення таблиць бази даних з використанням SQL-запитів або у конструкторі необхідно тільки створити класи моделей даних, а базу даних, якщо вона відсутня, буде згенеровано автоматично під час компіляції застосунку. Таким чином цей підхід зводить розробку бази даних до створення класів моделей сутностей бази даних, а також класу контексту бази даних, що має бути похідним від класу DbContext. Існують певні правила найменування властивостей класів моделей сутностей що дозволяють співставляти поля таблиць бази даних з властивостями моделей, наприклад первинним ключем є властивість з назвою «Id» або «<назва класу>Id», проте для визначення правил співставлення таблиць бази даних із класами моделей

можна також використовувати анотації або підхід Fluent API, який, зокрема, дозволяє визначати обмеження або значення за замовчуванням для полів таблиці [25-29]. Лістинг коду оголошення класів моделей, а також клас контексту бази даних наведено у додатку А.

Окремо зупинимось на деяких моментах. По-перше, не всі параметри сутностей, які зберігаються в базі даних, має сенс зберігати як окремі поля, наприклад поточний номер групи: він неодноразово змінюється, а його значення можна розрахувати маючи ідентифікатор групи. Для цього окрім властивостей класи містять методи.

По-друге, типом даних, який повертають деякі властивості, наприклад властивість Role класу User, є перелік (enum). Enum представляє собою набір іменованих цілочисельних констант [30]. У базі даних такій властивості відповідатиме поле з цілочисельним значенням, а об’єкт класу повертатиме його як певний enum [25,28]. За потреби значення можна привести до будь-якого цілочисельного типу, а також до рядкового, наприклад Role.STUDENT.ToString() поверне рядок “STUDENT”. У вигляді переліків створено наступні типи даних:

- тип додаткового балу (AdditionalPointType);
- оцінка за шкалою ECTS (ECTSpoint);
- форма навчання (EducationForm);
- тип підсумкового контролю (ExamType);
- оцінка за національною шкалою (NationalScalePoint);
- бальна шкала (PointScale);
- роль користувача в системі (Role);

Використовувані переліки в проекті розміщено в окремій папці Enums, лістинг коду їх оголошення наведено в додатку Б.

По-третє, деякі сутності в базі даних очевидно вимагають використання наслідування: наприклад і студент, і викладач, і працівник університету є користувачами системи, тобто з точки зору об’єктно-орієнтованого програмування

ці три класи є похідними від базового класу User. Подібна ситуація із навчальним курсом і практикою: практика є окремим різновидом навчального курсу, що повинен мати всі його властивості і дві власні: дату початку і закінчення.

Entity Framework Core передбачає три підходи до організації зв'язків наслідування в базі даних:

– Table Per Hierarchy (TPH) – одна таблиця на ієрархію класів, за такого підходу об'єкти, що походять від одного базового класу зберігаються в одній таблиці, в яку додається додаткова колонка, що вказує, до якого типу належить об'єкт – дискримінатор;

– Table Per Type (TPT) – одна таблиця на кожен клас в ієрархії наслідування, за такого підходу спільні властивості об'єктів різних класів, наприклад імена як студентів так і викладачів, зберігаються в одній таблиці, а інші – в окремих таблицях, пов'язаних зі спільною відношенням один до одного;

– Table Per Class (TPC) – одна таблиця на кожен окремий тип, за якого для кожної моделі створюється окрема таблиця з усіма її властивостями – цей підхід ускладнює зберігання даних, проте працює більш оптимально, адже зменшується кількість таблиць, які необхідно отримувати з бази даних для виконання запиту.

Перший підхід використовується за замовчуванням, другий і третій повинні налаштовуватись за допомогою анотацій та Fluent API [25-27,29].

Для збереження даних про різних користувачів системи доцільно використовувати підхід TPT, адже за використання TPH таблиця користувачів буде дуже громіздкою, а за TPC різні користувачі будуть «розкидані» по різних таблицях, що ускладнюватиме запити до БД для отримання даних про користувачів, наприклад для виконання аутентифікації користувача.

Відношення наслідування між сутностями «навчальний курс» та «практика» реалізоване за підходом TPH, адже практика має тільки два додаткові поля, і виносити їх в окрему таблицю не доцільно.

Структуру зв’язків між таблицями у базі даних демонструє діаграма структури бази даних, наведена на рис.3.1 (діаграму отримано у програмі SQL Server Management Tools).

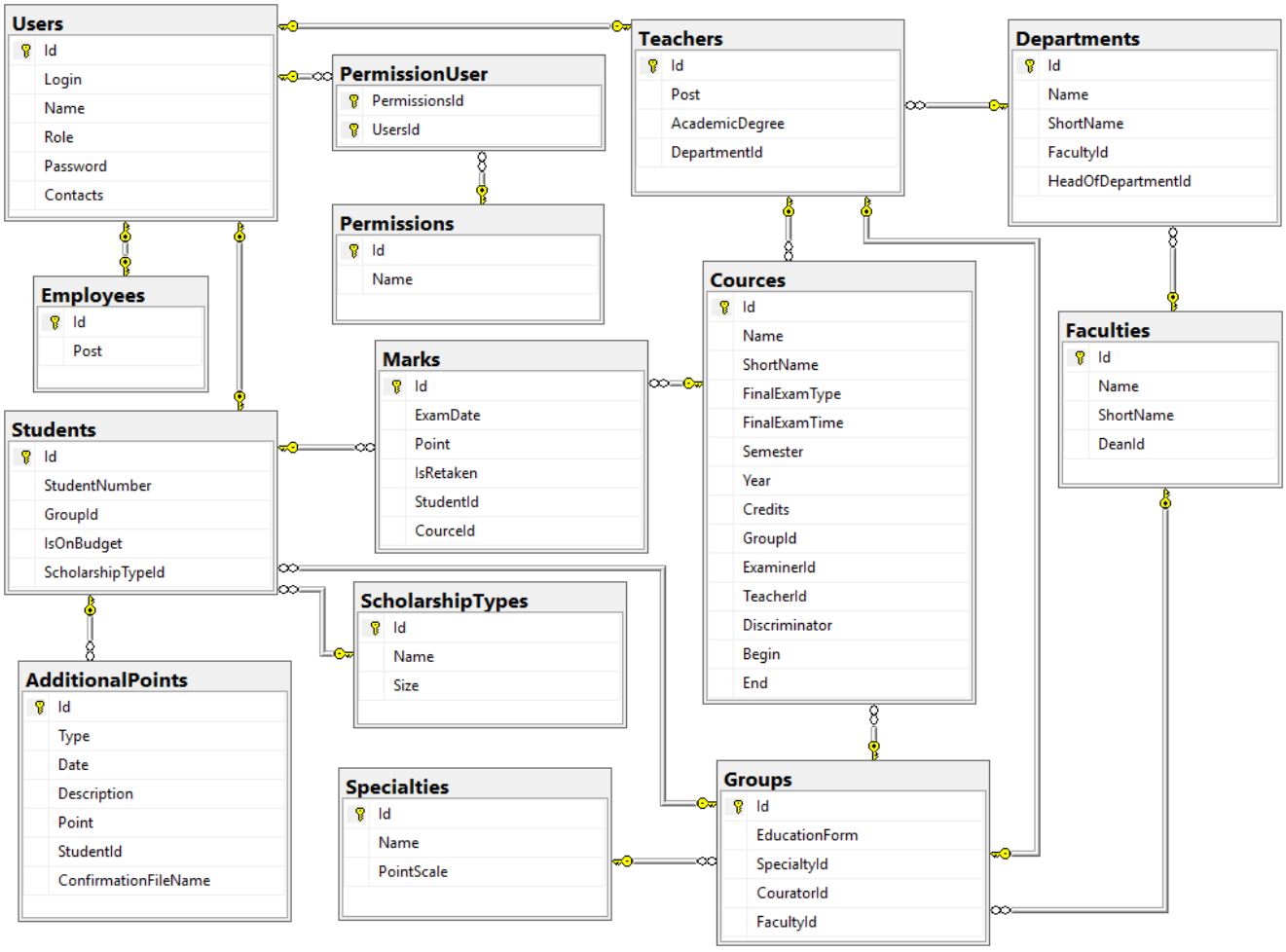


Рисунок 3.1 – Діаграма структури бази даних

3.2 Структура вебсайту

Розроблені сторінки вебсайту та стислу інформацію про їхній вміст наведено у табл.3.2 .

Таблиця 3.2 – Сторінки вебсайту

Посилання	Опис
/Index	головна сторінка, з якої можна перейти до решти сторінок вебсайту
/AdditionalPoints	перегляд додаткових балів студентів, за наявності відповідного дозволу – додання і редагування додаткових балів
/AdminContacts	понтакти адміністраторів вебсайту
/ChangePassword	зміна користувачем свого паролю для входу до системи
/Cources	перегляд студентом або викладачем своїх навчальних курсів, редагування навчальних курсів користувачами з відповідним дозволом
/Cources/{id}/points	виставлення викладачем балів з дисципліни
/Departments	перегляд списку кафедр університету, редагування даних про кафедри користувачами з відповідним дозволом
/Faculties	перегляд списку факультетів університету, редагування даних про факультети користувачами з відповідним дозволом
/Files	перегляд завантажених для ознайомлення користувачами вебсайту документів, перегляд правил і положень, відповідно до яких працює система рейтингування студентів, завантаження і видалення файлів користувачами з відповідним дозволом
/Groups	перегляд списку студентських груп, редагування студентських груп користувачами з відповідним дозволом
/Group/{id}	перегляд списку студентів групи
/Marks	перегляд оцінок студента
/Profile	перегляд користувачем власного профілю
/Profile/{id}	перегляд профілю користувача
/Rating	перегляд рейтингу студентської групи
/Scholarships	редагування стипендій студентів користувачем з відповідним дозволом
/SignIn	вхід на сайт
/Statistics	перегляд статистики успішності студентів
/Users	перегляд даних користувачів сайту, редагування даних користувачів сайту користувачем з відповідним дозволом

На рис.3.2 наведено схему, що відображає структуру вебсайту.

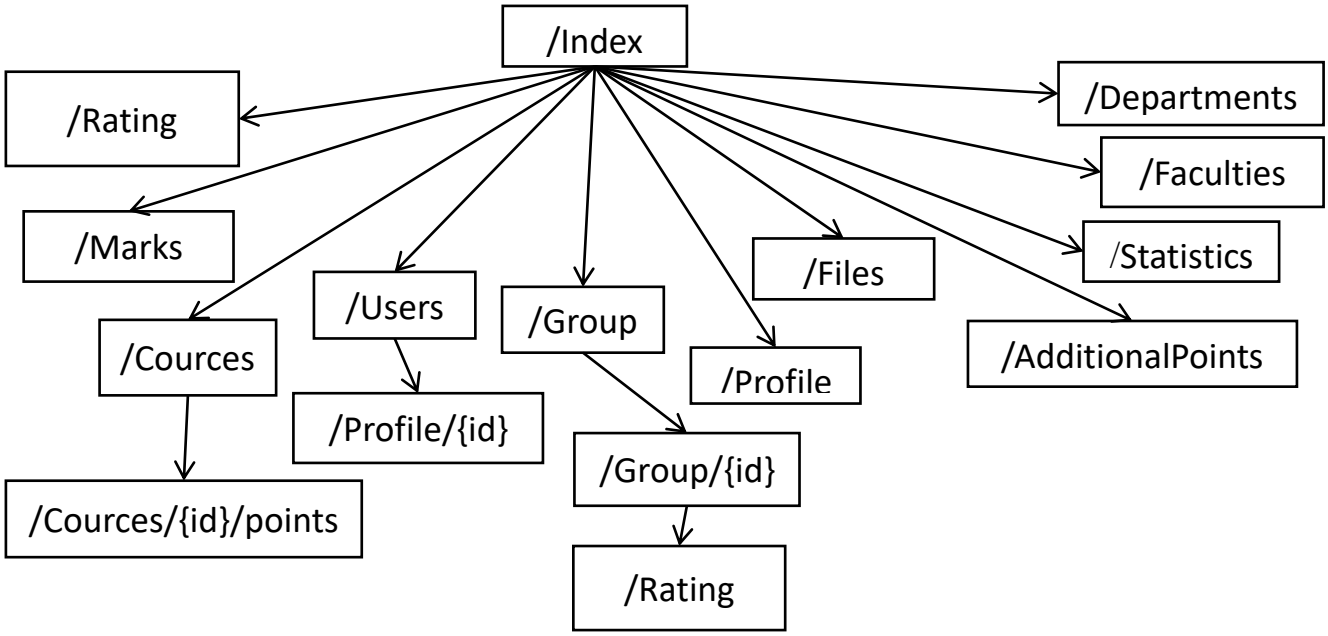


Рисунок 3.2 – Структура вебсайту

Структура вебсайту досить проста, на більшість сторінок можна перейти з головної сторінки.

Дії, які користувач може виконувати у системі і дані, доступні для перегляду користувачем, відрізняються залежно від його ролі і дозволів, що проілюстровано на рис.3.3 у вигляді діаграми прецедентів.

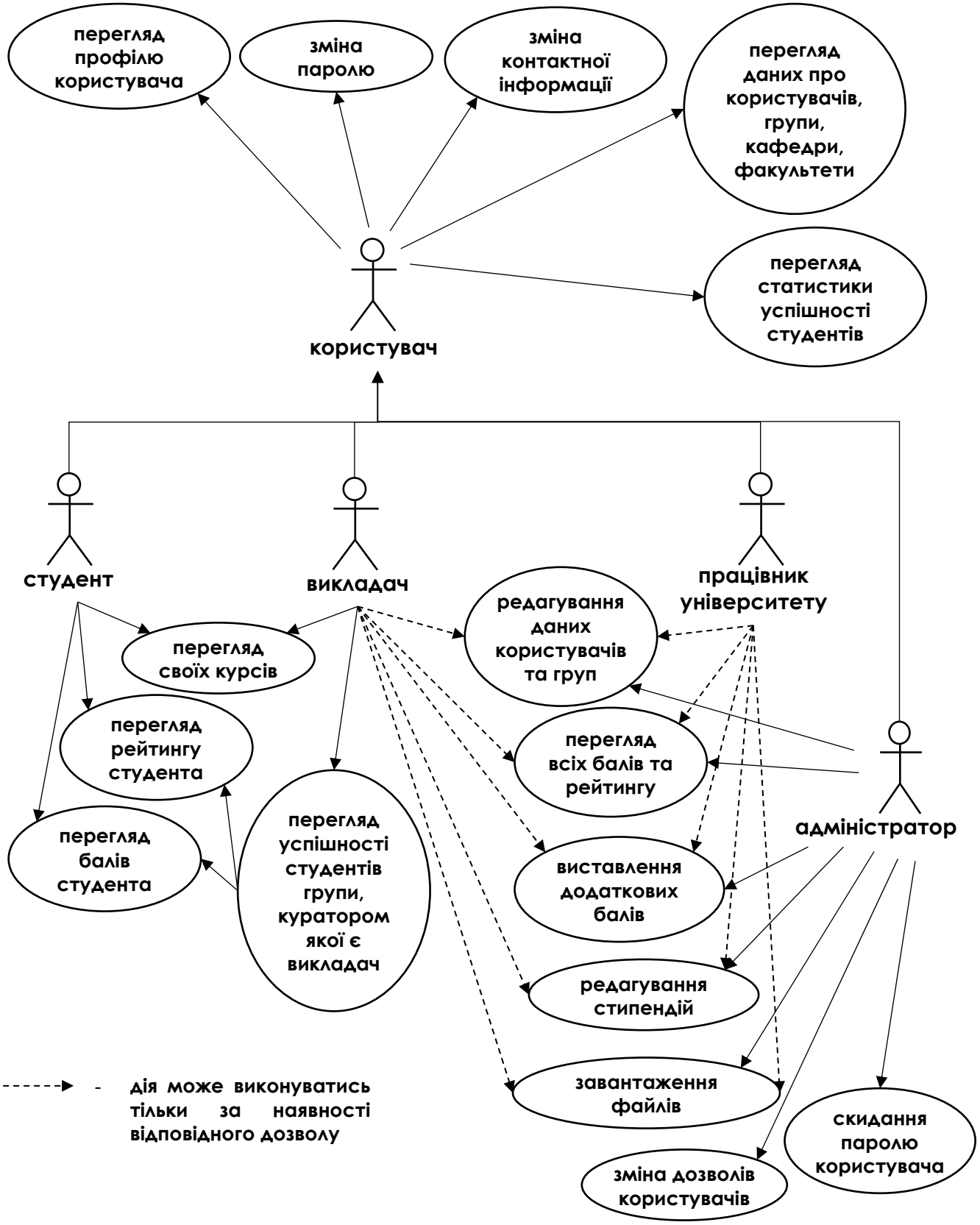


Рисунок 3.3 – Діаграма прецедентів

3.3 Взаємодія користувача з вебзастосунком

3.3.1 Аутентифікація та авторизація користувачів

Оскільки вебсистема, що розробляється, передбачає розподіл користувачів за ролями і надання їм різного функціоналу і рівня доступу до даних відповідно до цих ролей, необхідно передбачити аутентифікацію користувача при вході на сайт і авторизацію, коли користувач намагається отримати доступ до ресурсу, доступ до якого обмежений.

Фреймворк ASP.NET Core має вбудовану підтримку різних типів аутентифікації. Для виконання аутентифікації в конвеєрі обробки запиту існує спеціальний компонент middleware – AuthenticationMiddleware, для вбудовування якого використовується метод UseAuthentication(). Для виконання аутентифікації цей компонент використовує сервіс аутентифікації IAuthenticationService, що реєструється в застосунку за допомогою методу AddAuthentication(), що приймає рядок, який позначає схему аутентифікації [14,16].

Одним із поширених способів аутентифікації в вебзастосунках є аутентифікація за допомогою cookies. Для застосування цього способу в метод AddAuthentication() передається рядок “Cookies”, після чого викликається метод AddCookie() для налаштування параметрів аутентифікації:

```
builder.Services.AddAuthentication("Cookies")
    .AddCookie(
        options =>
        {
            options.LoginPath = "/SignIn";
             options.LoginPath = "/SignOut";
            options.AccessDeniedPath = "/Errors/403";
        }
          );
```

Сторінки, для доступу до яких користувач має бути аутентифікованим, вказуються в методі AddRazorPages() (додаток В). Якщо при вході на одну з цих

сторінок користувач не аутентифікований, його буде переспрямовано на сторінку /SignIn з формою для введення логіна і пароля. Після їх введення обчислюється хеш-функція паролю, і отримане значення порівнюється зі збереженим у базі даних. Якщо введено правильні логін і пароль, будуть створені аутентифікаційні куки, що зберігатимуть ідентифікатор користувача і його роль. Описані дії виконує код, наведений нижче:

```
using (var db = new StudentRatingDbContext())
{
	HashCalculator hashCalculator =
	PageContext.HttpContext.RequestServices.GetService<HashCalculator>();
	User user = db.Users.Where(u => u.Login == login && u.Password ==
	hashCalculator.CalculateHash(password)).FirstOrDefault();
	if(user == null)
	{
	    return RedirectToPage(new{ showWarning = true });
	}
	else
	{
	    var claims = new List<Claim> { new Claim("ID", user.Id.ToString()), new
	Claim(ClaimTypes.Name, login), new Claim(ClaimTypes.Role,
	user.Role.ToString()) };
	    ClaimsIdentity claimsIdentity = new ClaimsIdentity(claims, "Cookies");
	    await PageContext.HttpContext
	.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new
	ClaimsPrincipal(claimsIdentity));
	}
}
```

При будь-якому подальшому запиті користувача можна буде з контексту запиту одержати його роль і ідентифікатор, за яким вже одержати решту даних з БД. Це дозволить надавати користувачеві лише ті дані і права в системі, які передбачає його роль і дозволи.

3.3.2 Обробка запитів користувача

Користувач взаємодіє із вебзастосунком шляхом надсилання йому запитів певного типу, найчастіше – get або post. Технологія Razor Pages дозволяє обробляти такі запити в методах класу моделі вебсторінки. Кожна сторінка Razor має свій клас

моделі, похідний від класу PageModel, що за замовчуванням має назву <назва сторінки>Model, наприклад IndexModel. В ньому визначаються методи для обробки користувацьких запитів, що повинні мати назву відповідно до типу оброблюваного запиту, тобто OnGet(), OnPost(). Якщо запит має певні параметри, то метод мати параметри з аналогічними назвами [13,14,16]. Може виникнути ситуація, коли одна сторінка має обробляти кілька запитів одного і того ж типу, наприклад сторінка Users має обробляти додавання і редагування користувачів з різними ролями, відповідно є щонайменше три запити типу post, які вона має обробляти. В такому разі до назви методу додається назва обробника сторінки (page handler), наприклад OnPostStudent() , а в параметрах рядка запиту або в параметрах маршруту має бути передано параметр hanlder (для наведеного прикладу його значенням має бути рядок «student»). Нижче наведено метод, що обробляє запит типу post і виконує додання або редагування студента.

```
public IActionResult OnPostStudent(int id, string name, int number, int group, int
isOnBudget, string login = "")
        {
            if (!CurrentUserCanEditUsers()){return Forbid();}
            using (var db = new StudentRatingDbContext())
            {
                Student student = db.Students.Find(id)??new Student() { Login=login};
                student.Name = name;
                student.StudentNumber = number;
                student.GroupId = group;
                student.IsOnBudget = isOnBudget == 1;
                if(id == 0)
                {
                    db.Students.Add(student);
                }
                else
                {
                    db.Students.Update(student);
                }
                db.SaveChanges();
                if(id == 0)
                {
                    id = db.Students.Where(s => s.StudentNumber ==
student.StudentNumber).First().Id;
                }
            }
            return RedirectToPage("/Profile", new { id = id }); ;
        }
```

3.3.3 Обробка помилок

Іноді може скластись ситуація, коли користувач намагається отримати доступ до сторінки, якої не існує, або до якої він не має доступу. Для обробки такої ситуації спеціальний middleware компонент вбудовується шляхом додання наступного коду в метод main:

```
app.UseStatusCodePagesWithReExecute("/Errors/{0}");
```

У параметрах цього методу передається шаблон адреси, на яку переспрямовуватиметься користувач, якщо після обробки його запиту було повернуто статусний код, наприклад для коду 404 – на сторінку /Errors/404 .

Висновки до розділу 3

В цьому було коротко розглянуто структуру таблиць розробленої бази даних та зв’язки між ними, структуру вебсайту, ролі користувачів та дії, які вони можуть виконувати на сайті. Розглянуто аутентифікацію і авторизацію користувачів, обробку вебзастосунком користувацьких запитів.

4 ОПИС РОБОТИ ВЕБСИСТЕМИ

4.1 Вхід користувача на сайт, головна сторінка

Якщо при вході на одну зі сторінок, що вимагають аутентифікації, користувач не є аутентифікованим, його буде переспрямовано на сторінку входу на сайт з формою для введення логіна і пароля (рис.4.1).

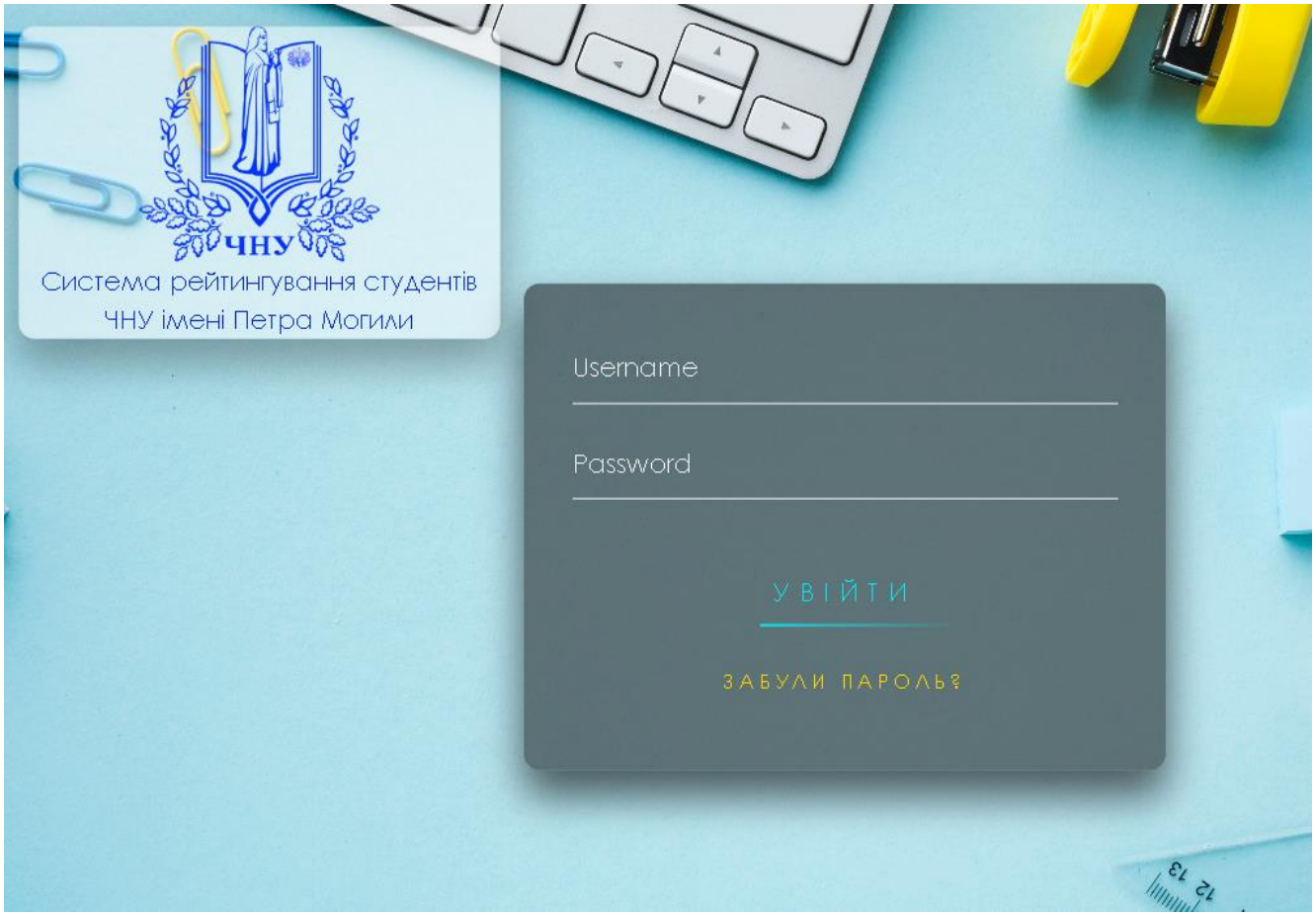


Рисунок 4.1 – Форма для входу на сайт

Окрім кнопки «Увійти» ця форма також містить посилання «Забули пароль?», що переспрямовує на сторінку з контактами адміністраторів вебсистеми (рис.4.2).

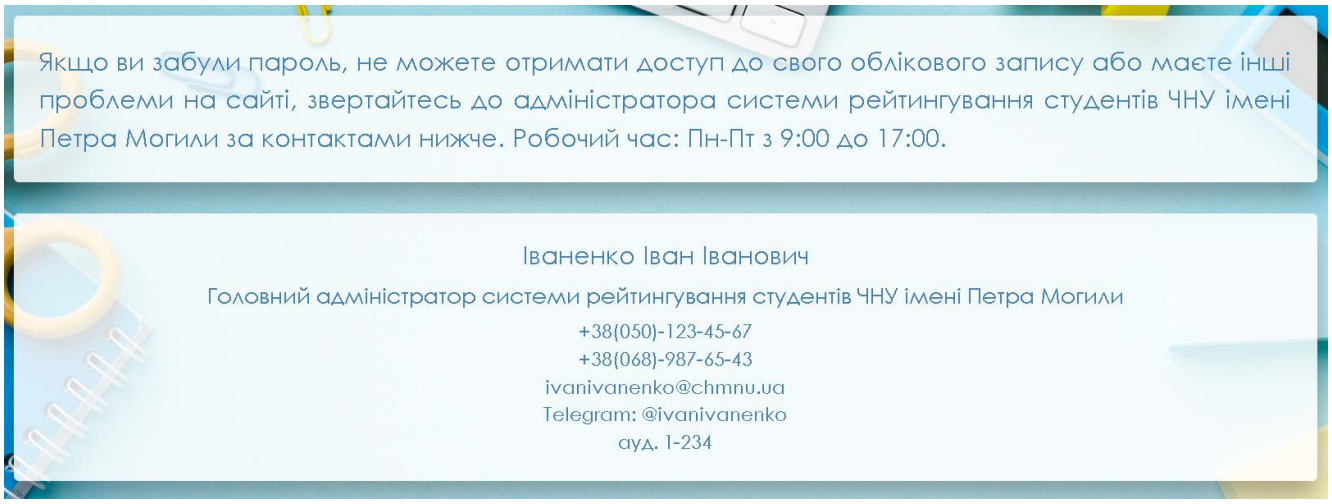


Рисунок 4.2 – Сторінка з контактними даними адміністратора вебсистеми

Якщо введено неправильний пароль, буде відображено повідомлення про це (рис.4.3), якщо ж аутентифікація успішна, користувача буде переспрямовано на головну сторінку (рис.4.4).

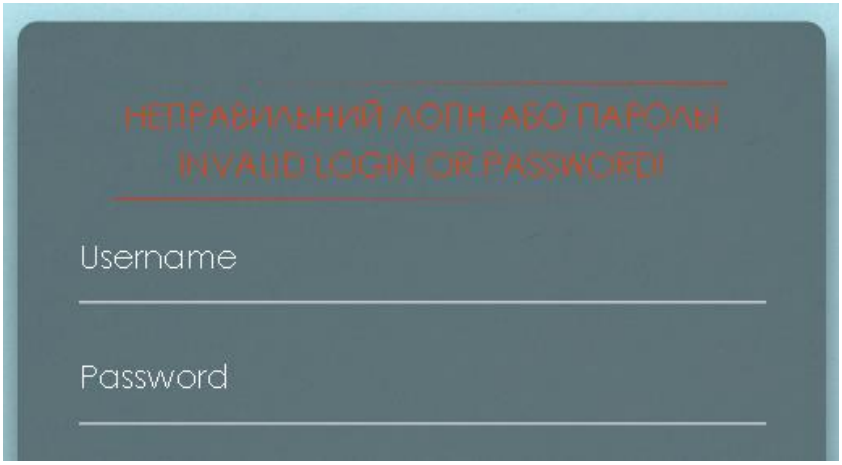


Рисунок 4.3 – Повідомлення про введення неправильного пароля

Рисунок 4.4 – Головна сторінка вебсайту

З головної сторінки можна перейти на інші сторінки вебсайту, а також вона містить посилання для завантаження документів: це може бути розклад, важливі документи, тощо – файли, завантажені користувачем з відповідним дозволом. Перейти на деякі сторінки можна також за допомогою навігаційної панелі, що відрізняється для різних ролей користувачів.

4.2 Профіль користувача

Перейшовши за посиланням «Профіль» на навігаційній панелі, користувач потрапляє на сторінку профілю користувача, де він може переглянути дані про себе, змінити пароль і вказати контактні дані, які бачитимуть інші користувачі (рис.4.5).

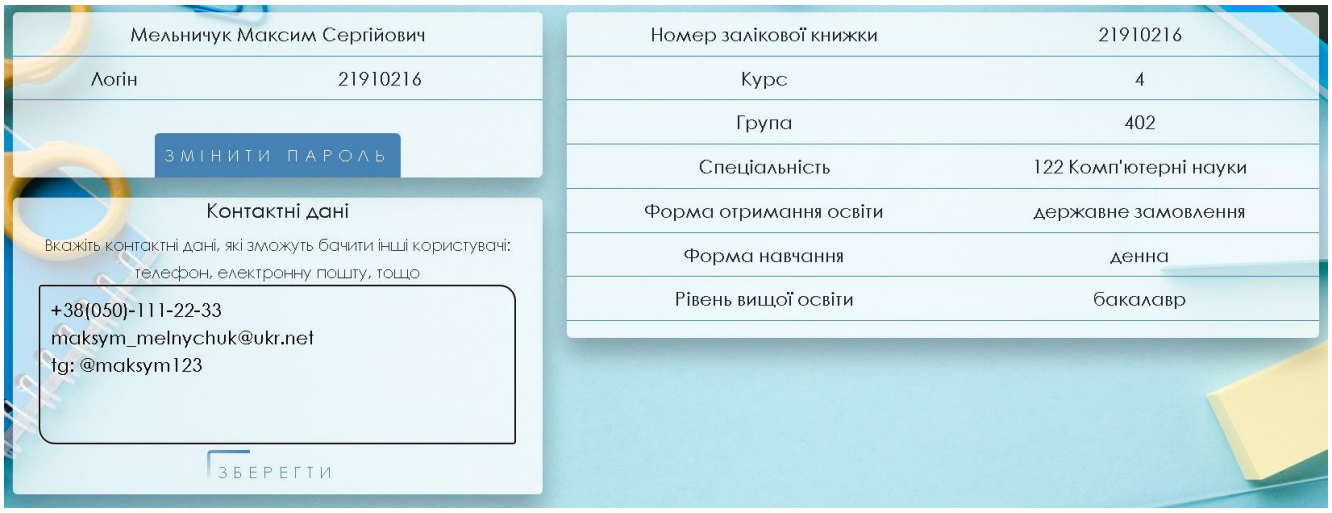


Рисунок 4.5 – Перегляд користувачем-студентом власного профілю

Натиснувши «Змінити пароль» користувач переходить до форми зміни паролю (рис.4.6).

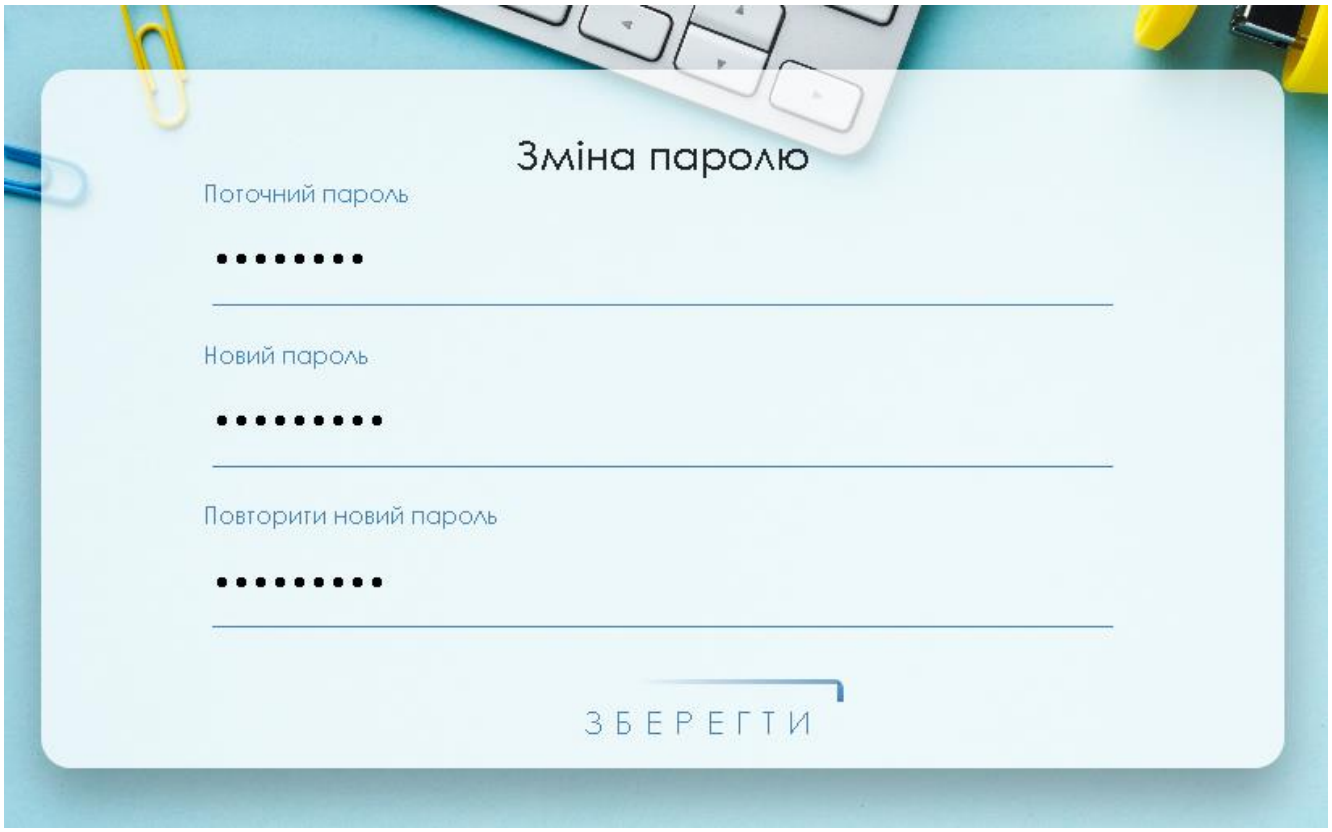


Рисунок 4.6 – Форма для зміни паролю користувачем

Перейти до перегляду профілю іншого користувача (рис.4.7) можна зі сторінки «Користувачі».

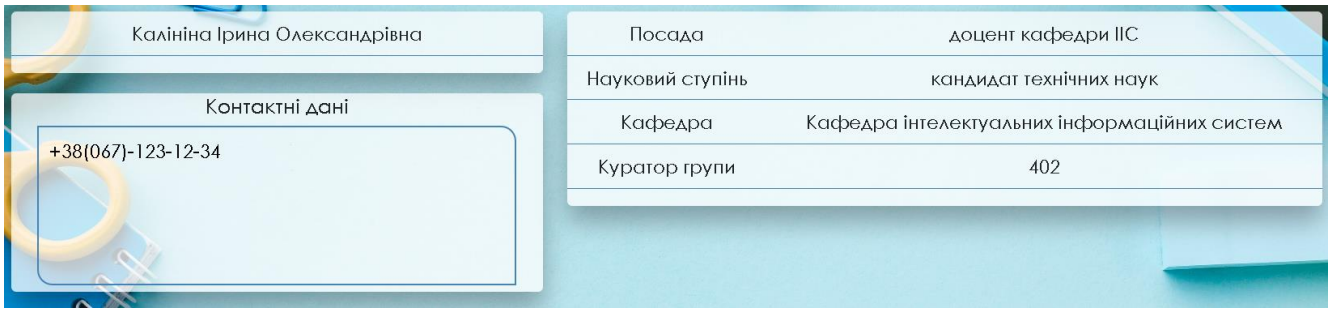


Рисунок 4.7 – Перегляд профілю користувача-викладача

Адміністратор вебсистеми може скинути пароль користувача (рис.4.8), в такому разі буде встановлено значення паролю за замовчуванням, таке ж, яке встановлюється при доданні нового користувача – «12345678», користувач зможе зайти за цим паролем, після чого змінити пароль.

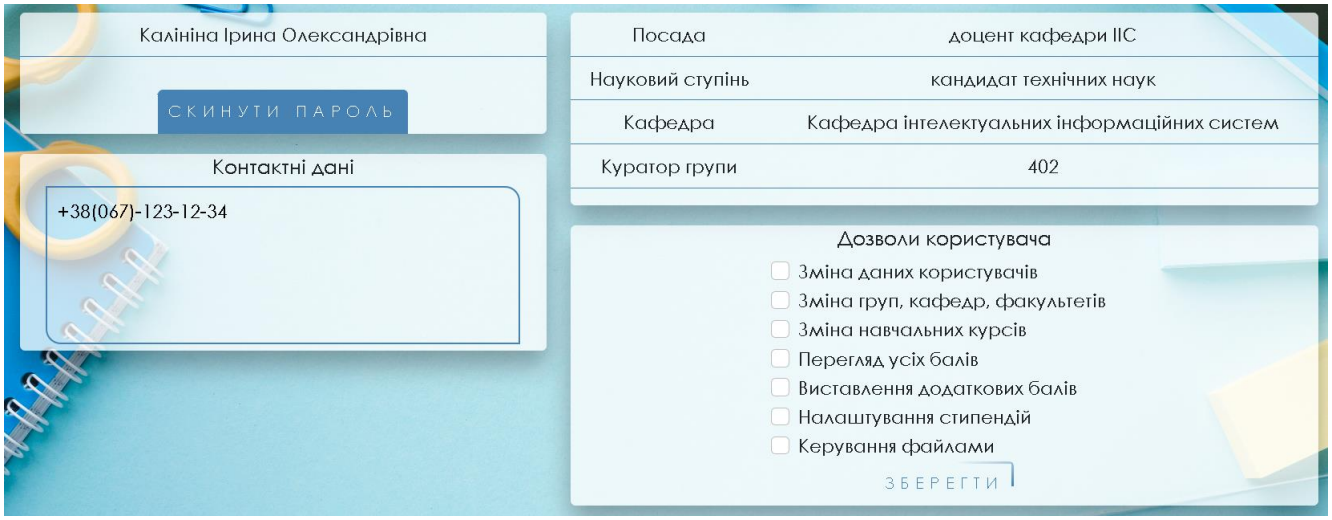


Рисунок 4.8 – Перегляд адміністратором профілю користувача

В профілі викладача або працівника університету адміністратор також має можливість редагувати перелік дозволів користувача, що можна побачити на рис.4.8 . В вебсистемі передбачено наступні дозволи:

– Зміна даних користувачів – додання, редагування та видалення даних користувачів вебсистеми;

– Зміна груп, кафедр, факультетів – додання, редагування та видалення даних про групи, кафедри і факультети;

– Зміна навчальних курсів – додання, редагування і видалення даних про навчальні курси;

– Перегляд усіх балів – можливість перегляду балів всіх студентів і рейтингових списків всіх груп;

– Виставлення додаткових балів – додання, редагування і видалення додаткових балів студентів;

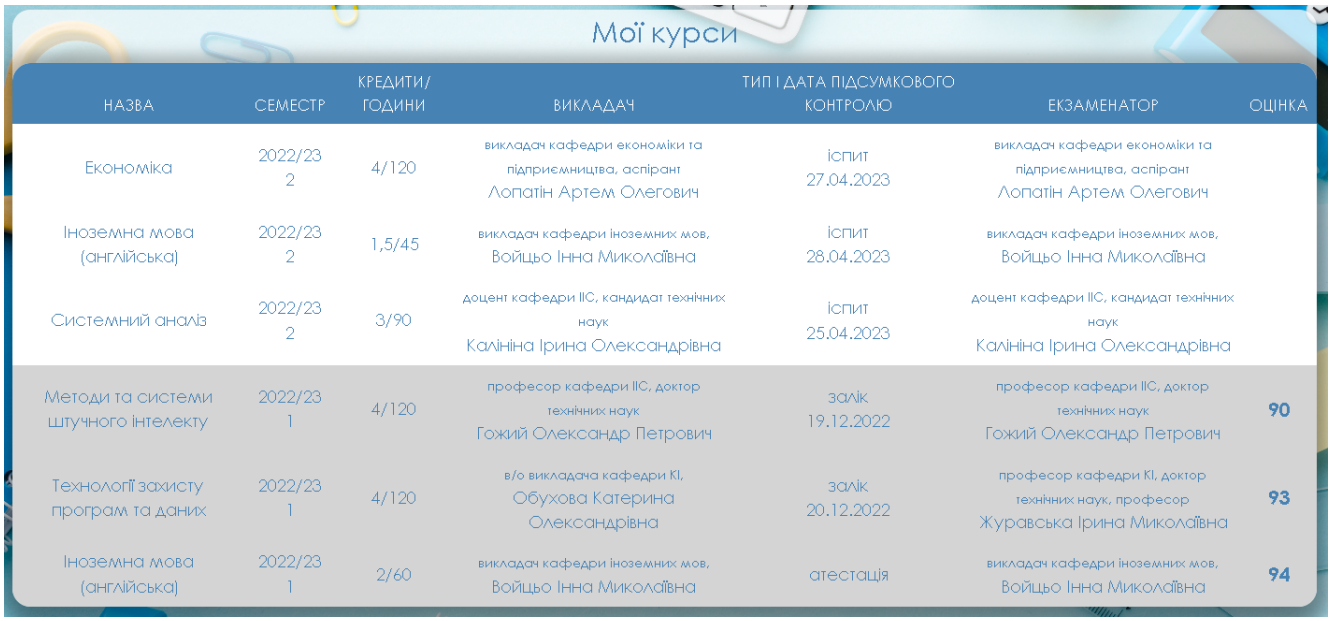
– Налаштування стипендій – зміна розміру різних типів стипендій, зміна відсотка студентів-бюджетників, що отримують стипендію, призначення студентам стипендії;

– Керування файлами: завантаження і видалення файлів для ознайомлення користувачами вебсайту;

4.3 Бали студентів

Доступ до балів та рейтингових списків обмежений: всі бали студента може переглянути сам студент, куратор його групи та користувач із дозволом на перегляд усіх балів. Рейтинговий список певної спеціальності та курсу може переглядати студент цієї спеціальності та курсу, викладач що є куратором однієї з груп на цьому курсі і спеціальності і користувач з дозволом на перегляд всіх балів.

Свої бали у 100-бальній (200-бальній) шкалі студент може переглянути на сторінці «Курси» («/Cources») (рис.4.9), а більш детально – на сторінці «Бали» («/Marks») (рис.4.10).



Мої курси

НАЗВА	СЕМЕСТР	КРЕДИТИ/ ГОДИНИ	ВИКЛАДАЧ	ТИП І ДАТА ПІДСУМКОВОГО КОНТРОЛЮ	ЕКЗАМЕНАТОР	ОЦІНКА
Економіка	2022/23 2	4/120	викладач кафедри економіки та підприємництва, аспірант Лопатін Артем Олегович	іспит 27.04.2023	викладач кафедри економіки та підприємництва, аспірант Лопатін Артем Олегович	
Іноземна мова (англійська)	2022/23 2	1,5/45	викладач кафедри іноземних мов, Войцьо Інна Миколаївна	іспит 28.04.2023	викладач кафедри іноземних мов, Войцьо Інна Миколаївна	
Системний аналіз	2022/23 2	3/90	доцент кафедри ІІС, кандидат технічних наук Калініна Ірина Олександрівна	іспит 25.04.2023	доцент кафедри ІІС, кандидат технічних наук Калініна Ірина Олександрівна	
Методи та системи штучного інтелекту	2022/23 1	4/120	професор кафедри ІІС, доктор технічних наук Гожий Олександр Петрович	залік 19.12.2022	професор кафедри ІІС, доктор технічних наук Гожий Олександр Петрович	90
Технології захисту програм та даних	2022/23 1	4/120	в/о викладача кафедри КІ, Обухова Катерина Олександрівна	залік 20.12.2022	професор кафедри КІ, доктор технічних наук, професор Журавська Ірина Миколаївна	93
Іноземна мова (англійська)	2022/23 1	2/60	викладач кафедри іноземних мов, Войцьо Інна Миколаївна	атестація	викладач кафедри іноземних мов, Войцьо Інна Миколаївна	94

Рисунок 4.9 – Таблиця з даними про навчальні курси студента

Таблиця з оцінками студента на сторінці /Marks стилізовано під сторінки залікової книжки студента (рис.4.10) перегортаючи які можна переглянути бали за різні семестри.

7 семестр 20 22 / 23 навчального року

ІСПИТИ

№	Дисципліна	Кількість Кредитів	Годин	Екзаменатор	Оцінка за шкалою 100-б.	національною	ECTS	Дата складання
1	ТІС	4	120	Дончєнко М. В.	100	відмінно	A	24.12.22
2	Психологія	4	120	Опанасенко Л. Я.	97	відмінно	A	26.12.22
3	ТІПС	3	90	Скакодуб Ф. С.	100	відмінно	A	28.12.22

4 КУРС

ЗАЛІКИ

№	Дисципліна	Кількість Кредитів	Годин	Викладач	Оцінка за шкалою 100-б.	національною	ECTS	Дата складання
1	БЖД	3	90	Макарова Ф. В.	100	зарах.	A	19.12.22
2	МСШІ	4	120	Гожий Ф. П.	90	зарах.	A	19.12.22
3	ТЗТІД	4	120	Журавська І. М.	93	зарах.	A	20.12.22

ЧОРНОМОРСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ПЕТРА МОГИЛИ

№ залікової книжки: 21910216
ПІБ студента: Мельничук Максим Сергійович
Факультет: Факультет Комп'ютерних
Група: 219102 (402)
Спеціальність: 122 Комп'ютерні на
Форма навчання: денна
Освітній рівень: бакалавр

№ | Дисципліна
1 | ТІС 22
2 | Психо 22
3 | ТІПС 22

ПРАКТИКА

Назва	Тривалість від	до	Керівник	Оцінка за шкалою 100-б.	національною	ECTS	Дата
Виробнича практика	19.09 2022	16.10 2022	Сіденко Є.В.	100	відмінно	A	17.10 2022

Рисунок 4.10 – Перегляд оцінок студента на сторінці /Marks

Назви предметів і прізвища викладачів в таблиці наведено в скороченому вигляді, проте навівши курсор на них можна побачити їх повністю (рис.4.11):

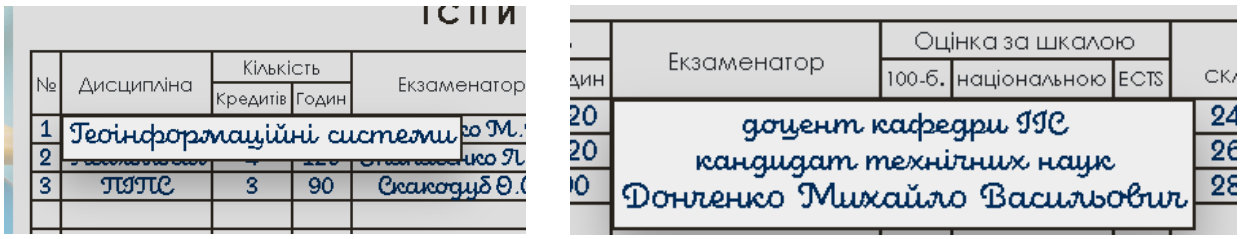


Рисунок 4.11 – Відображення повністю назви предмета та імені викладача в таблиці оцінок студента

4.4 Рейтинговий список студентів

Для перегляду рейтингового списку студент має перейти на сторінку /Rating натиснувши кнопку «Рейтинг» на навігаційній панелі, також це можна зробити натиснувши кнопку «Рейтинг групи» на сторінці списку студентської групи. Ця сторінка містить рейтингові списки студентів певного курсу і спеціальності за всі семестри, за замовчуванням відображається рейтинг за останній семестр (рис.4.12).

У списку спочатку наведено студентів, що навчаються за державним замовленням і не мають боргів або перездач, відсортованих за зменшенням рейтингового балу. Далі наведено студентів, що навчаються за контрактом і не мають боргів, далі – студентів, які мають принаймні одну заборгованість. Студентів, що навчаються за контрактом, також відсортовано за рейтинговим балом, що дозволяє побачити кого з них в першу чергу можна перевести на бюджет в разі звільнення місця, а боржників відсортовано за кількістю боргів.

Рейтинг студентів 4 курсу спеціальності 122 - Комп'ютерні науки

7 семестр

СЕРЕДНІЙ БАЛ	СУМА ДОДАТКОВИХ БАЛІВ				КІЛЬКІСТЬ БОРГІВ	КІЛЬКІСТЬ ПЕРЕЗДАЧ
	ВСЬОГО	ЗА ГРОМАДСЬКУ ДІЯЛЬНІСТЬ	ЗА НАУКОВУ РОБОТУ	ЗА СПОРТИВНІ ДОСЯГЕННЯ		
75,797325	3,3	1,5	0,8	1	23	6

№	ІМ'Я СТУДЕНТА	№ ЗАЛІКОВОЇ КНИЖКИ	ГРУПА	ФОРМА НАВЧАННЯ	СЕРЕДНІЙ ЗВАЖЕНИЙ БАЛ	КІЛЬКІСТЬ БОРГІВ	КІЛЬКІСТЬ ПЕРЕЗДАЧ	ДОДАТКОВІ БАЛИ	РЕЙТИНГОВИЙ БАЛ	СТИПЕНДІЯ
1	Пилипчук Богдан Віталійович	21910119	401	ДЗ	98,28	0	0	1,5	89,95	2000
2	Слободенюк Антоніна Ігорівна	21910122	401	ДЗ	98,72	0	0	0,8	89,65	2000
3	Пещерьова Вікторія Сергіївна	21910118	401	ДЗ	99,52	0	0	0	89,57	2000
4	Бухтіярова Олена Олегівна	21910202	402	ДЗ	98,56	0	0	0	88,7	2000
5	Колодяжний Кирило Олексійович	21910112	401	ДЗ	97,28	0	0	1	88,55	2000

Рисунок 4.12 – Вміст сторінки /Rating

Вгорі відображаються деякі статистичні дані: середній бал (середнє арифметичне середньозважених балів студентів групи), сума додаткових балів за різні типи активності, кількість боргів і перездач. Нижче наводиться список студентів, рядки якого виділено різними кольорами для різних студентів, значення кольорів наведено в кінці таблиці (рис.4.13).

№	ІМ'Я СТУДЕНТА	ЗАЛІКОВОЇ КНИЖКИ	ГРУПА	ФОРМА НАВЧАННЯ	ЗВАЖЕНИЙ БАЛ	КІЛЬКІСТЬ БОРГІВ	КІЛЬКІСТЬ ПЕРЕЗДАЧ	ДОДАТКОВІ БАЛИ	РЕЙТИНГОВИЙ БАЛ	СТИПЕНДІЯ
	Олександрович									

Значення кольорів рядків таблиці

Студент(ка) претендує на підвищену стипендію

Студент(ка) претендує на звичайну стипендію

Студент(ка) не претендує на стипендію і не має боргів

Студент(ка) претендує на соціальну стипендію

Студент(ка) навчається за контрактом і не має боргів

Студент(ка) має заборгованість принаймні з однієї дисципліни

Рисунок 4.13 – Пояснення значення кольорів рядків рейтингового списку

В рейтинговому списку наводиться ім’я студента, номер залікової книжки, група, поточна стипендія, та ті дані, які є суттєвими для визначення місця в рейтингу.

4.5 Навчальні дисципліни і оцінювання студентів

Дії із навчальними курсами виконуються на сторінці /Cources. Якщо користувач має відповідний дозвіл, він може додавати і редагувати навчальні курси. Форму для додання і редагування навчального курсу наведено на рис 4.14.

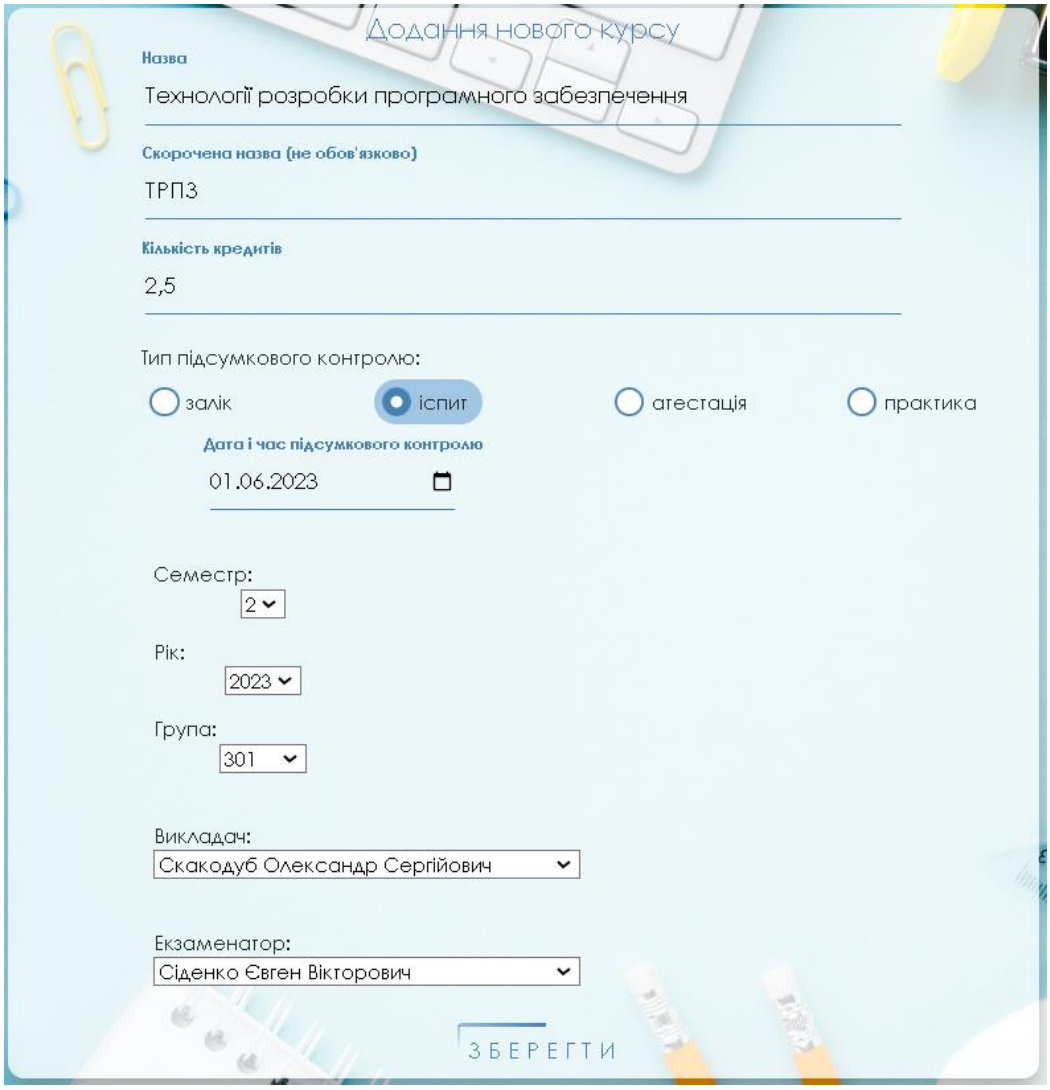


Рисунок 4.14 – Форма для додання і редагування навчального курсу

Студенти і викладачі на сторінці /Cources можуть переглянути свої курси (рис.4.9, 4.15).

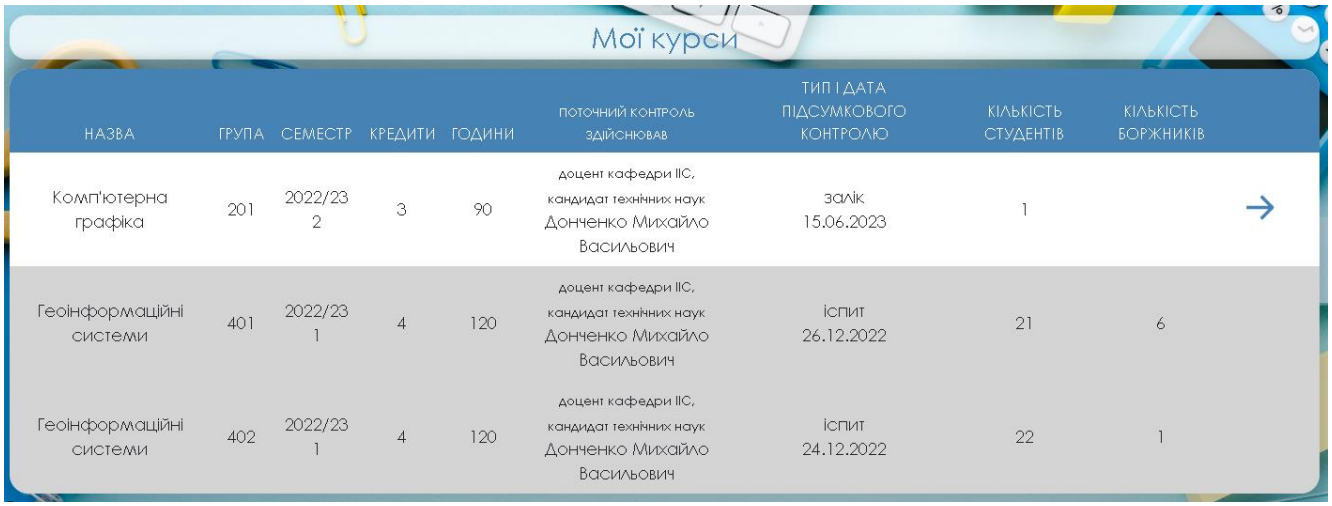


Рисунок 4.15 – Таблиця з даними про навчальні курси викладача

Натиснувши кнопку зі стрілкою, викладач може перейти до оцінювання студентів з певної дисципліни (рис.4.16). Біля кожної оцінки є поле для вибору дати, коли її виставлено, якщо оцінку не виставлено, то воно містить сьогоднішню дату.

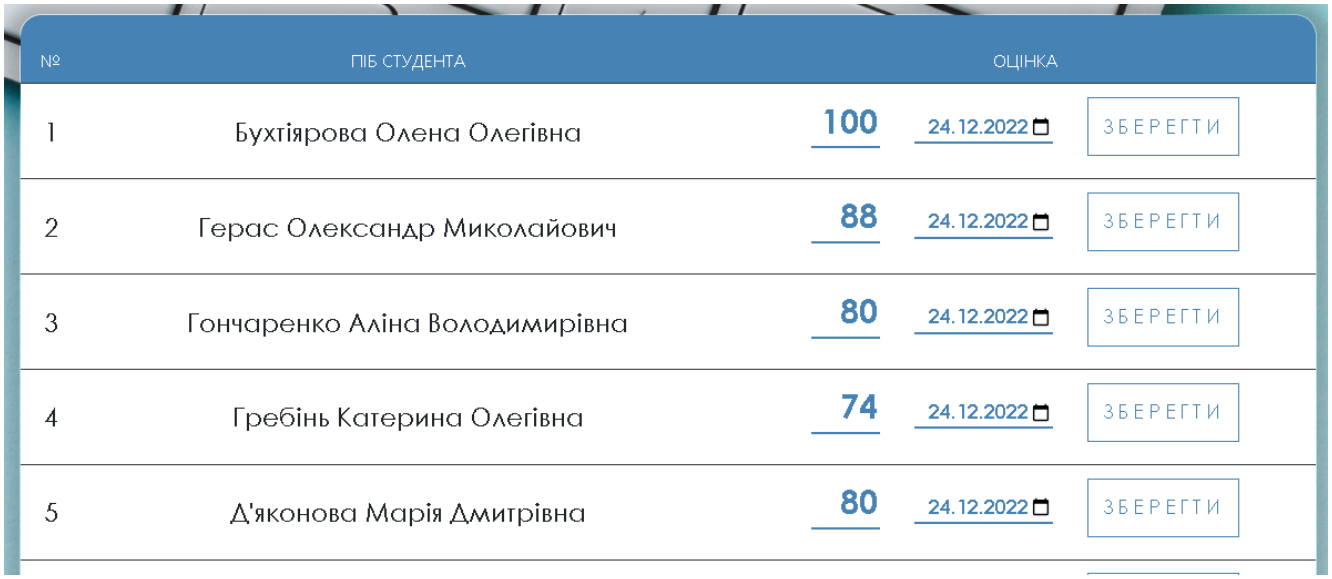


Рисунок 4.16 – Частина сторінки для виставлення балів студентам

4.6 Користувачі вебсистеми

Переглянути список користувачів вебсистеми можна на сторінці /Users. Тут користувачів залежно від ролі згруповано у 3 таблиці, дані в яких можна відсортувати та відфільтрувати за значеннями стовпців (рис.4.17 – 4.19).

Студенти

ПІБ	№ залікової книжки	Курс	Спеціальність	Група	Форма освіти	Форма навчання
				402	К	
Д'яконова Марія Дмитрівна	22020202	4	122	402	К	денна
Рева Владислав Володимирович	21810319	4	122	402	К	денна
Чекамова Дар'я Юріївна	21910226	4	122	402	К	денна

Рисунок 4.17 – Таблиця студентів, відфільтрована за групою і формою освіти

Викладачі

ПІБ	Кафедра	Науковий ступінь	Посада
	інтелектуальних інф		
Болюбаш Надія Миколаївна	Кафедра інтелектуальних інформаційних систем	кандидат педагогічних наук, доцент	доцент кафедри ІІС
Воробйова Алла Іванівна	Кафедра інтелектуальних інформаційних систем	кандидат фізико-математичних наук, доцент	доцент кафедри ІІС
Гожий Олександр Петрович	Кафедра інтелектуальних інформаційних систем	доктор технічних наук	професор кафедри ІІС
Донченко Михайло Васильович	Кафедра інтелектуальних інформаційних систем	кандидат технічних наук	доцент кафедри ІІС

Рисунок 4.18 – Таблиця викладачів, відфільтрована за назвою кафедри

Рисунок 4.19 – Таблиця працівників університету

За замовчуванням студентів відсортовано за іменем і групою, а викладачів – за іменем і назвою кафедри.

Додавати і редагувати користувачів може адміністратор вебсистеми або користувач із відповідним дозволом, в такому разі в крайній справа колонці таблиці наведено кнопки для редагування і видалення, для решти користувачів там буде тільки кнопка для перегляду профілю користувача.

На рис.4.20 – 4.22 наведено форми для додання і редагування користувачів з різними ролями.

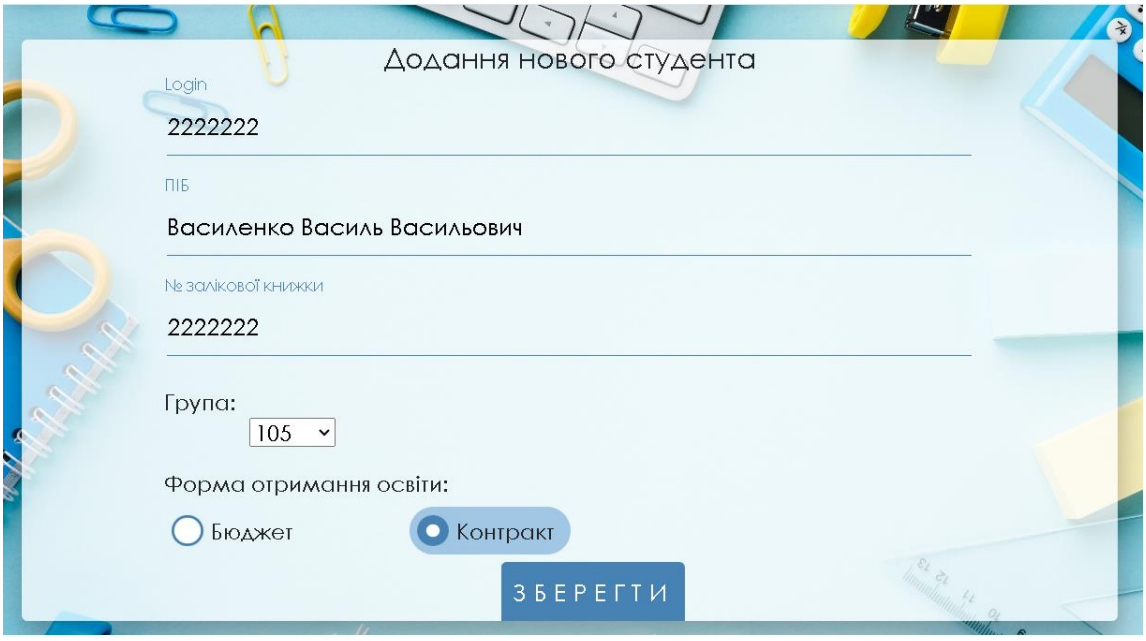


Рисунок 4.20 – Форма для додання нового студента

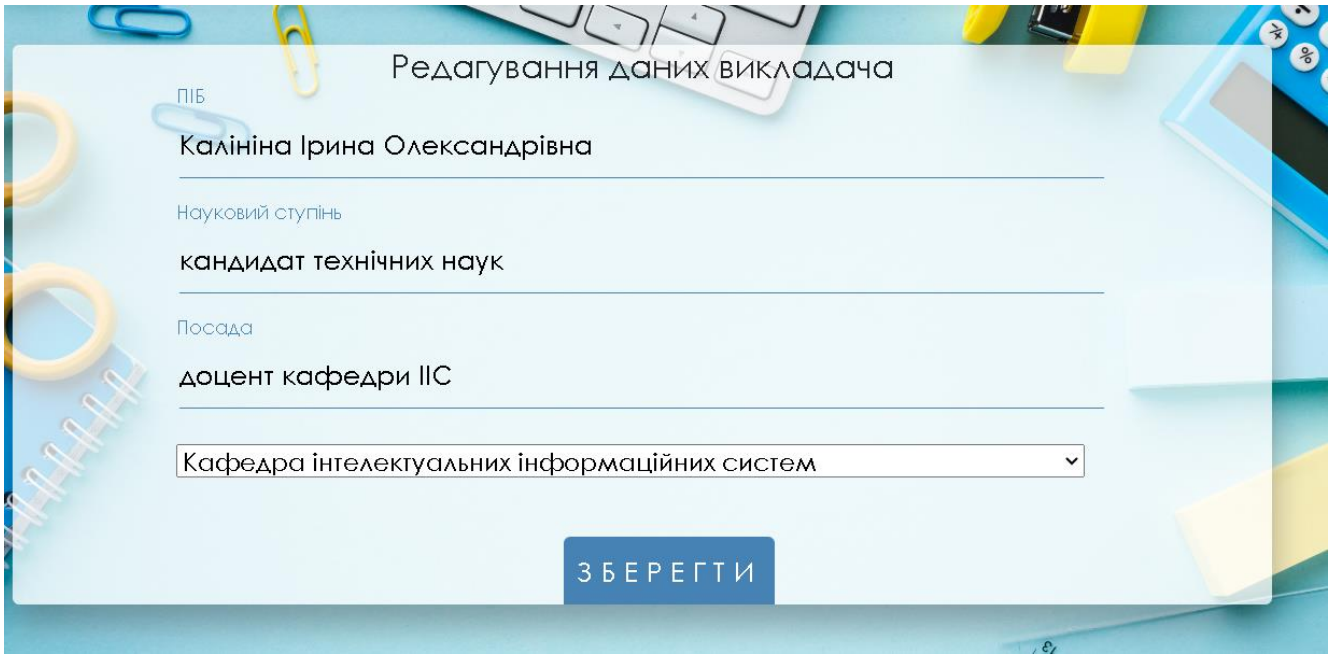


Рисунок 4.21 – Форма для редагування викладача

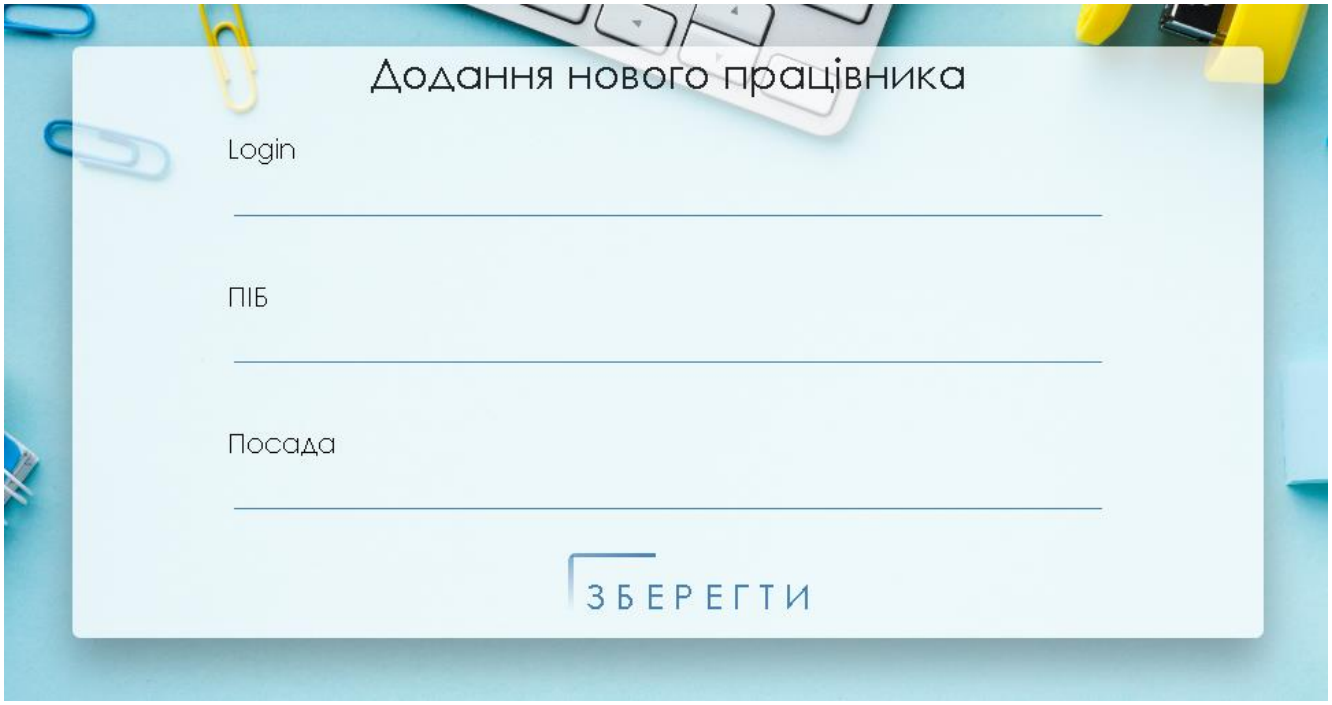


Рисунок 4.22 – Форма для додання працівника університету

4.7 Студентські групи і підрозділи університету

Переглянути список студентських груп можна на сторінці /Groups, користувач з відповідним дозволом також матиме тут кнопки для додання, редагування і видалення групи (рис.4.23). Форму для додання і редагування групи наведено на рис.4.24.

Рисунок 4.23 – Таблиця студентських груп

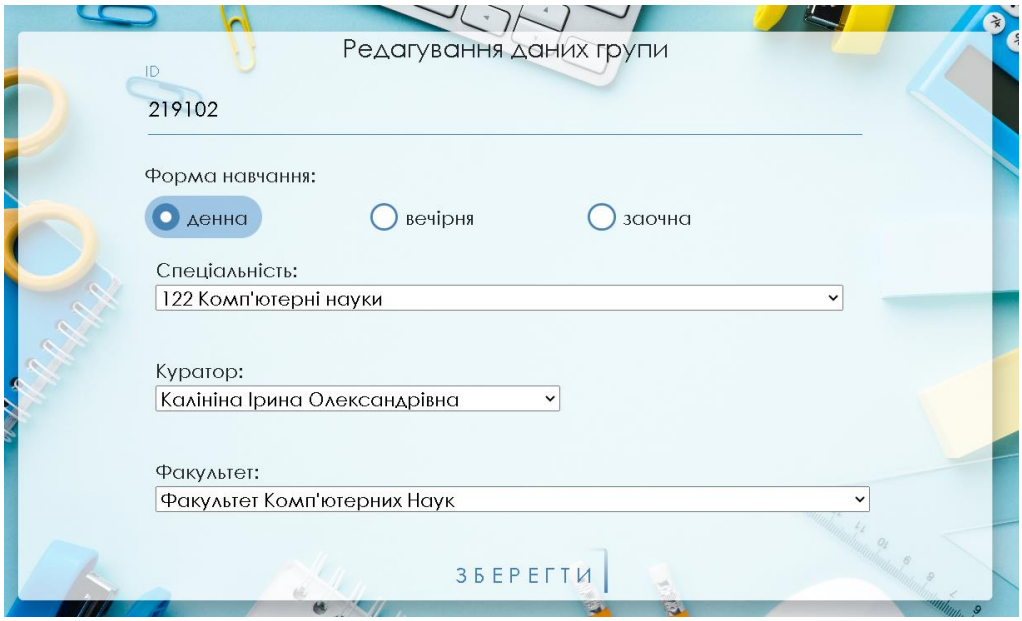
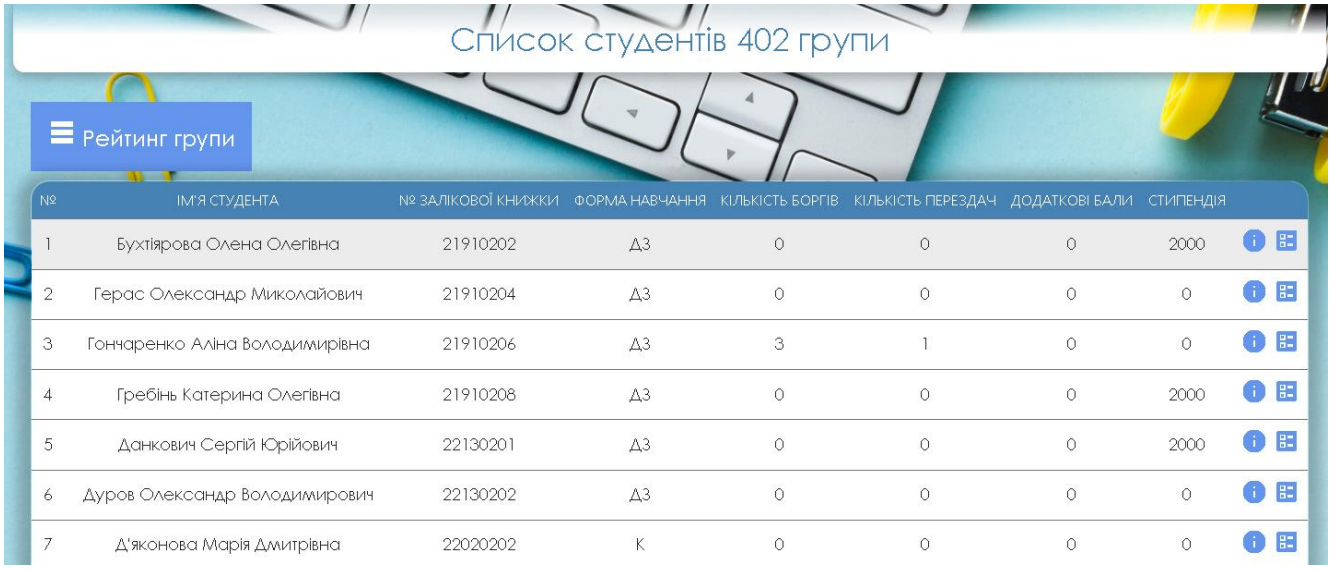


Рисунок 4.24 – Форма редагування студентської групи

Також біля кожної групи в таблиці (рис.4.23) є кнопка для перегляду списку студентів групи, список студентів однієї з груп наведено на рис.4.25 .



№	ІМ'Я СТУДЕНТА	№ ЗАЛІКОВОЇ КНИЖКИ	ФОРМА НАВЧАННЯ	КІЛЬКІСТЬ БОРГІВ	КІЛЬКІСТЬ ПЕРЕЗДАЧ	ДОДАТКОВІ БАЛИ	СТИПЕНДІЯ
1	Бухтіярова Олена Олегівна	21910202	ДЗ	0	0	0	2000
2	Герас Олександр Миколайович	21910204	ДЗ	0	0	0	0
3	Гончаренко Аліна Володимирівна	21910206	ДЗ	3	1	0	0
4	Гребінь Катерина Олегівна	21910208	ДЗ	0	0	0	2000
5	Данкович Сергій Юрійович	22130201	ДЗ	0	0	0	2000
6	Дуров Олександр Володимирович	22130202	ДЗ	0	0	0	0
7	Д'яконова Марія Дмитрівна	22020202	К	0	0	0	0

Рисунок 4.25 – Список студентів групи

Біля кожного студента групи є кнопки для переходу до профілю студента та до його оцінок, а над таблицею – кнопка для переходу до рейтингу студентів цієї групи.

Переглянути список кафедр і факультетів університету можна на сторінках /Departments (рис.4.26) і /Faculties (рис.4.27) відповідно.

Назва	Факультет	Завідувач
Кафедра інтелектуальних інформаційних систем	Факультет Комп'ютерних Наук	Кондратенко Юрій Пантелійович
Кафедра комп'ютерної інженерії	Факультет Комп'ютерних Наук	Крайник Ярослав Михайлович
Кафедра автоматизації та комп'ютерно-інтегрованих технологій	Факультет Комп'ютерних Наук	
Кафедра інженерії програмного забезпечення	Факультет Комп'ютерних Наук	Давиденко Євген Олександрович

Рисунок 4.26 – Таблиця кафедр університету

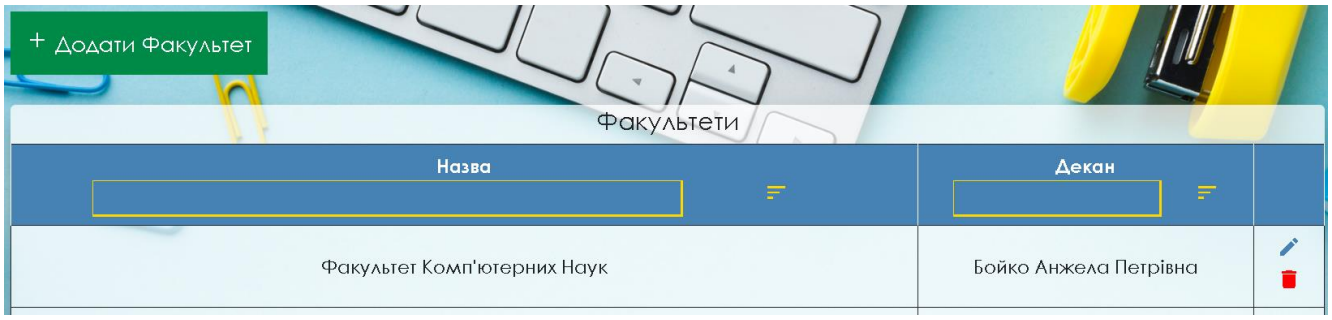


Рисунок 4.27 – Таблиця факультетів університету

Адміністратор вебсистеми або користувач з відповідним дозволом може редагувати дані про факультети і кафедри (рис.4.28, 4.29), зокрема змінювати завідувача кафедри і декана факультету, яких можна вибирати із числа викладачів кафедри або факультету.

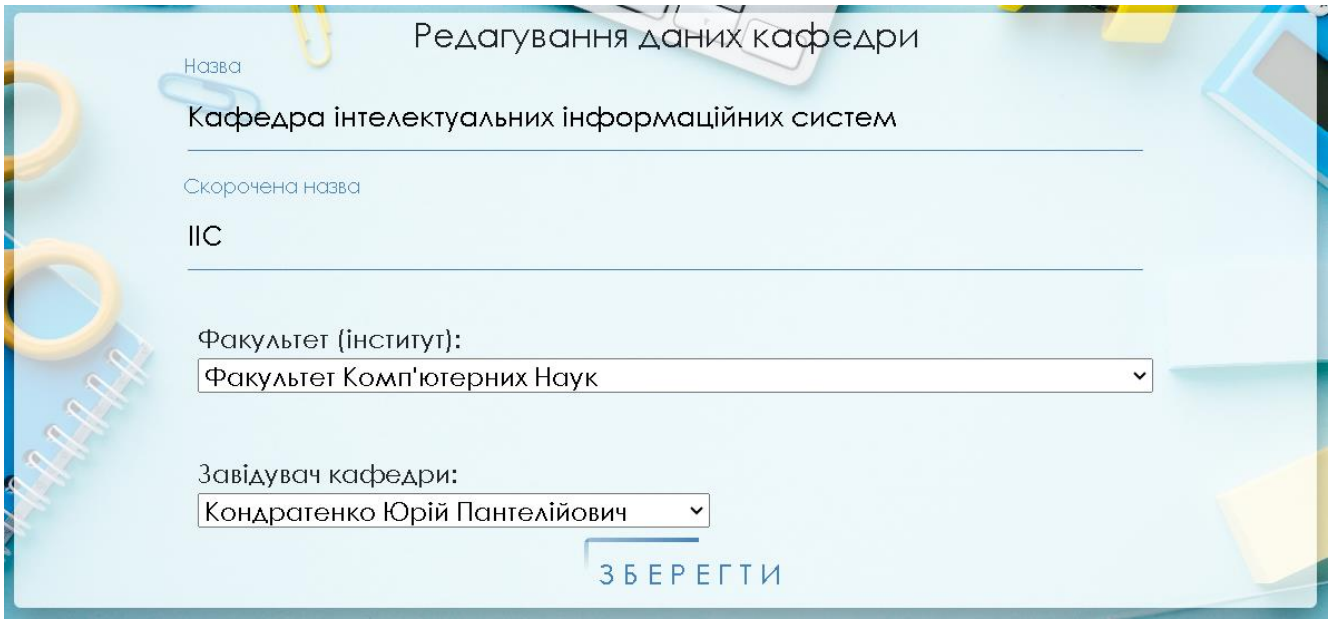


Рисунок 4.28 – Форма для редагування кафедри

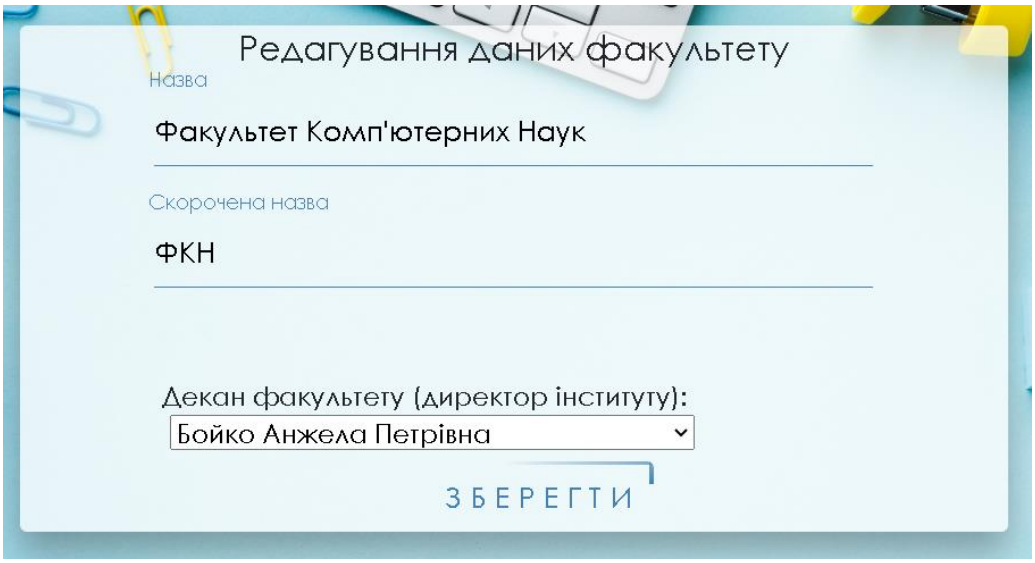


Рисунок 4.29 – Форма для редагування факультету

У всіх формах на сайті, де є випадаючі списки для вибору студента, викладача або кафедри вони групуються відповідно за групою, кафедрою і факультетом (рис.4.30).

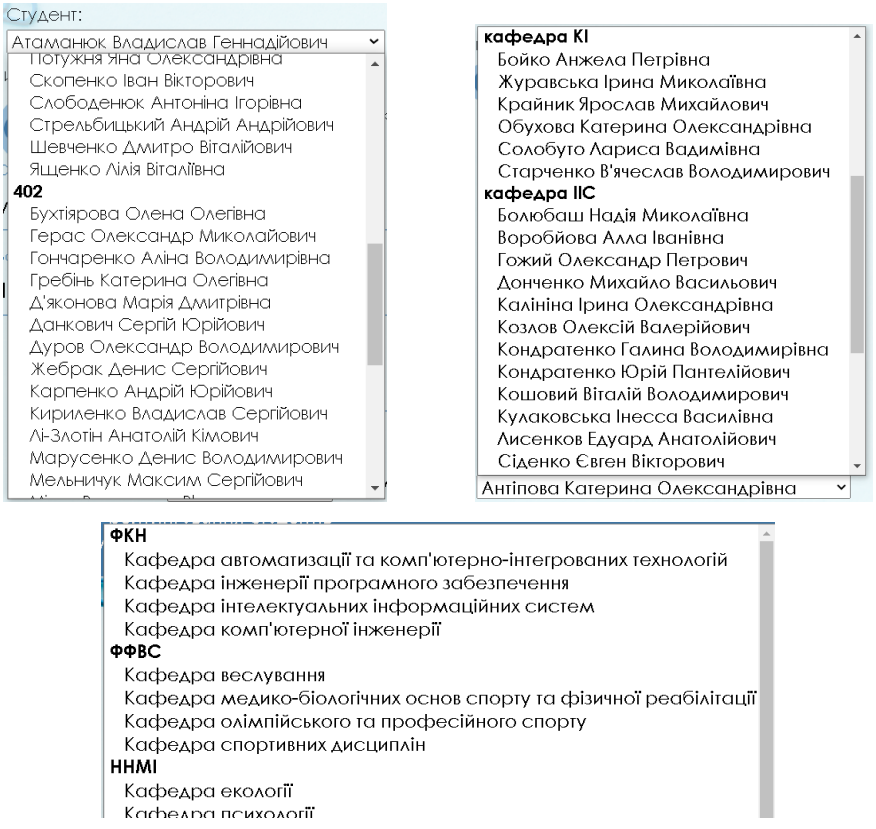


Рисунок 4.30 – Випадаючі списки для вибору студента, викладача, кафедри

4.8 Додаткові бали

Суму додаткових балів студента, які впливають на поточний рейтинг, тобто за попередній семестр, можна переглянути у рейтинговому списку студентів (рис.4.12) і в списку студентської групи (рис.4.25). Більш детальні відомості про додаткові бали студентів можна переглянути на сторінці /AdditionalPoints, на яку можна перейти натиснувши посилання «Додаткові бали» на головній сторінці. Список додаткових балів студентів однієї з груп наведено на рис.4.30. До кожного додаткового балу наводиться його тип, дата нарахування, опис, власне сам бал і посилання для завантаження файлу, що представляє собою документ, який підтверджує підстави нарахування додаткового бала. Якщо такий файл не було завантажено, то замість кнопки буде повідомлення про це (рис.4.31), а сам додатковий бал не буде враховуватись при визначенні суми додаткових балів студента за семестр і розрахунку рейтингового бала студента.

ПІБ студента	Група	Тип	Дата	Опис	Бал	Документ	
	401						
Пилипчук Богдан Віталійович	401	громадська діяльність	12.10.2022	участь у громадському заході	1,5	ЗАВАНТАЖИТИ ФАЙЛ	
Слободенюк Антоніна Ігорівна	401	наукова робота	22.10.2022	участь в олімпіаді з програмування	0,8	ЗАВАНТАЖИТИ ФАЙЛ	
Колодяжний Кирило Олексійович	401	спортивні досягнення	10.11.2022	участь у змаганнях	1	ЗАВАНТАЖИТИ ФАЙЛ	

Рисунок 4.31 – Список додаткових балів студентів однієї з груп

Мельничук Максим Сергійович	402	громадська діяльність	14.06.2023	участь у громадському заході	1	ФАЙЛ ВІДСУТНІЙ

Рисунок 4.32 – Приклад додаткового бала без підтверджуючого документа

Редагувати додаткові бали студентів і нараховувати нові може користувач із дозволом на редагування додаткових балів. Форму для редагування додаткового бала студента наведено на рис.4.32 .

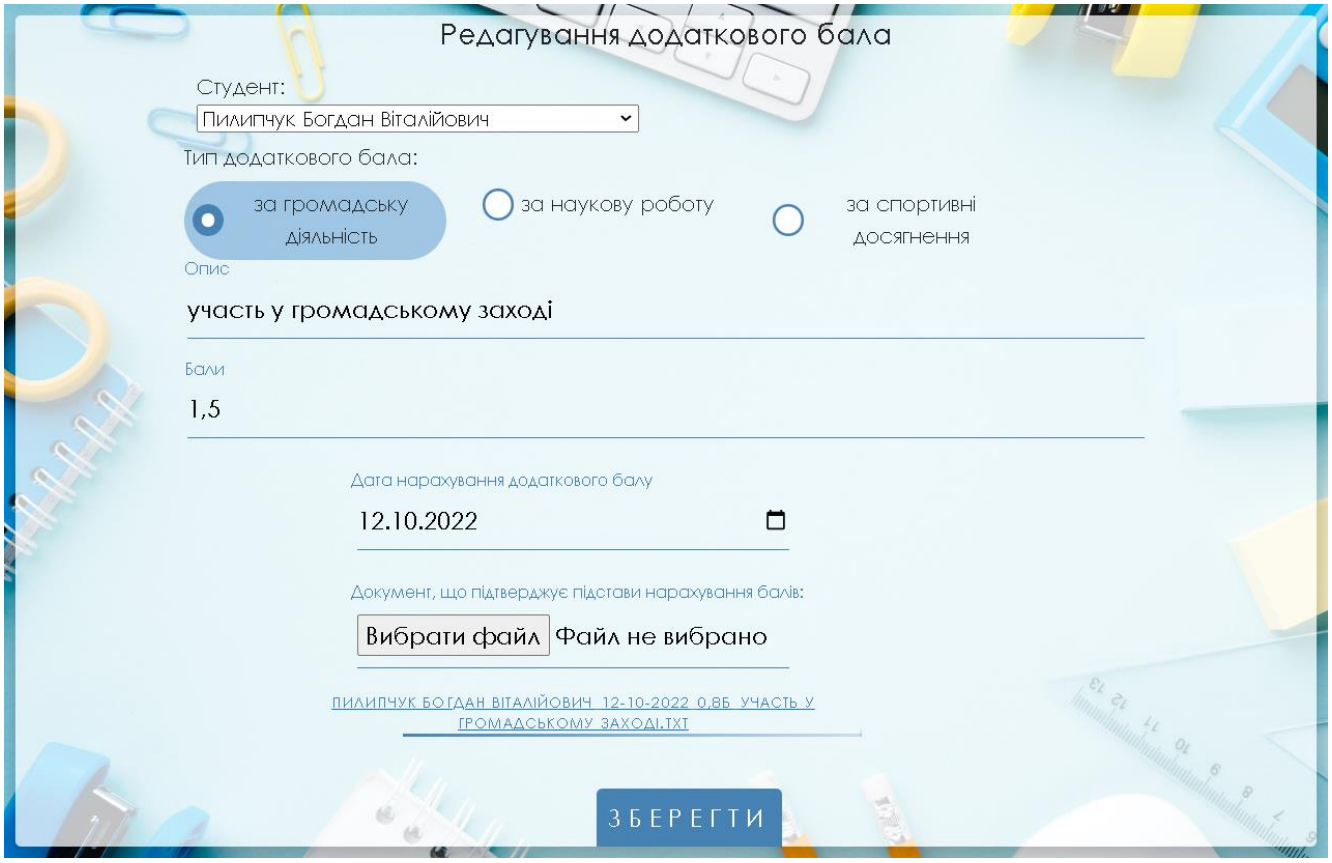


Рисунок 4.33 – Форма для редагування додаткового бала

Якщо раніше вже було завантажено файл, що підтверджує підстави нарахування додаткових балів, то форма міститиме посилання для його завантаження. Якщо завантажити інший файл і натиснути кнопку «Зберегти», то старий файл буде видалено. Збережений файл зберігається з назвою, що містить

повне ім’я студента, дату нарахування балу, кількість балів та опис. Файли зберігаються в каталозі WebRoot, за замовчуванням це папка wwwroot в папці проекту застосунку.

4.9 Статистика успішності студентів

Статистичні дані про успішність студентів певного курсу і спеціальності можна переглянути на сторінці рейтингу студентів, вони наведені над рейтинговим списком (рис.4.34).

Рисунок 4.34 - Статистика успішності студентів певного курсу і спеціальності за семестр

Більш детальну статистику можна переглянути на сторінці /Statistics перейшовши на неї з головної сторінки за посиланням «Статистика».

Рисунок 4.35 – Вміст сторінки /Statistics

На цій сторінці наведено середній бал студентів і суму додаткових балів, а також зображено середні бали студентів різних курсів у вигляді гістограми і співвідношення між додатковими балами за різні типи діяльності у вигляді діаграми. Конкретні значення можна побачити, навівши курсор в область діаграми (рис. 4.36).

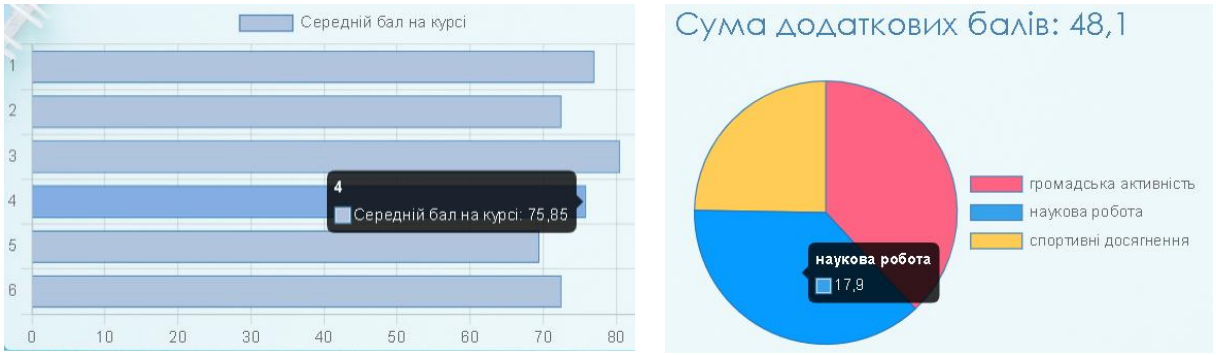


Рисунок 4.36 – Діаграми на сторінці /Statistics

За замовчуванням статистика наводиться для університету загалом, проте можна вибрати конкретну спеціальність і переглянути статистику для неї. Перемикачі зліва і справа дозволяють перейти до статистики за попередні семестри.

4.10 Стипендії студентів

Редагування обсягу стипендії і кількості стипендіатів відбувається на сторінці /Scholarships , на яку можуть перейти з головної сторінки тільки користувачі, що мають відповідний дозвіл.

У верхній частині сторінки (рис.4.37) розташовано форму для редагування обсягу стипендій різних типів: академічної звичайної, академічної підвищеної і соціальної, форму для редагування відсотка стипендіатів і кнопку для автоматичного призначення стипендій за рейтинговими списками.

Рисунок 4.37 – Частина вмісту сторінки /Scholarships

Відсоток стипендіатів визначає частину студентів певного курсу і спеціальності, що навчаються на бюджеті, які мають отримувати стипендію, він встановлюється однаковим для всіх спеціальностей.

Кнопка призначення стипендій автоматично дозволяє призначити стипендії тим студентам, що за поточними рейтинговими списками претендують на них. Та оскільки призначені стипендії можуть відрізнятись від тих, на які студенти претендують за балами, а також для призначення соціальних стипендій, виплата яких не залежить від місця в рейтингу, нижче наведено таблицю, що дозволяє призначати стипендію кожному студенту окремо (рис.4.38)

Рисунок 4.38 – Таблиця для редагування стипендій студентів

4.11 Документи

На головній сторінці користувачі можуть ознайомитись з переліком завантажених документів (рис.4.39). Тут може бути розклад, шаблони заяв, документи для ознайомлення, загалом будь-які файли, які користувач із дозволом на завантаження файлів вважатиме за необхідне завантажити на сайт.

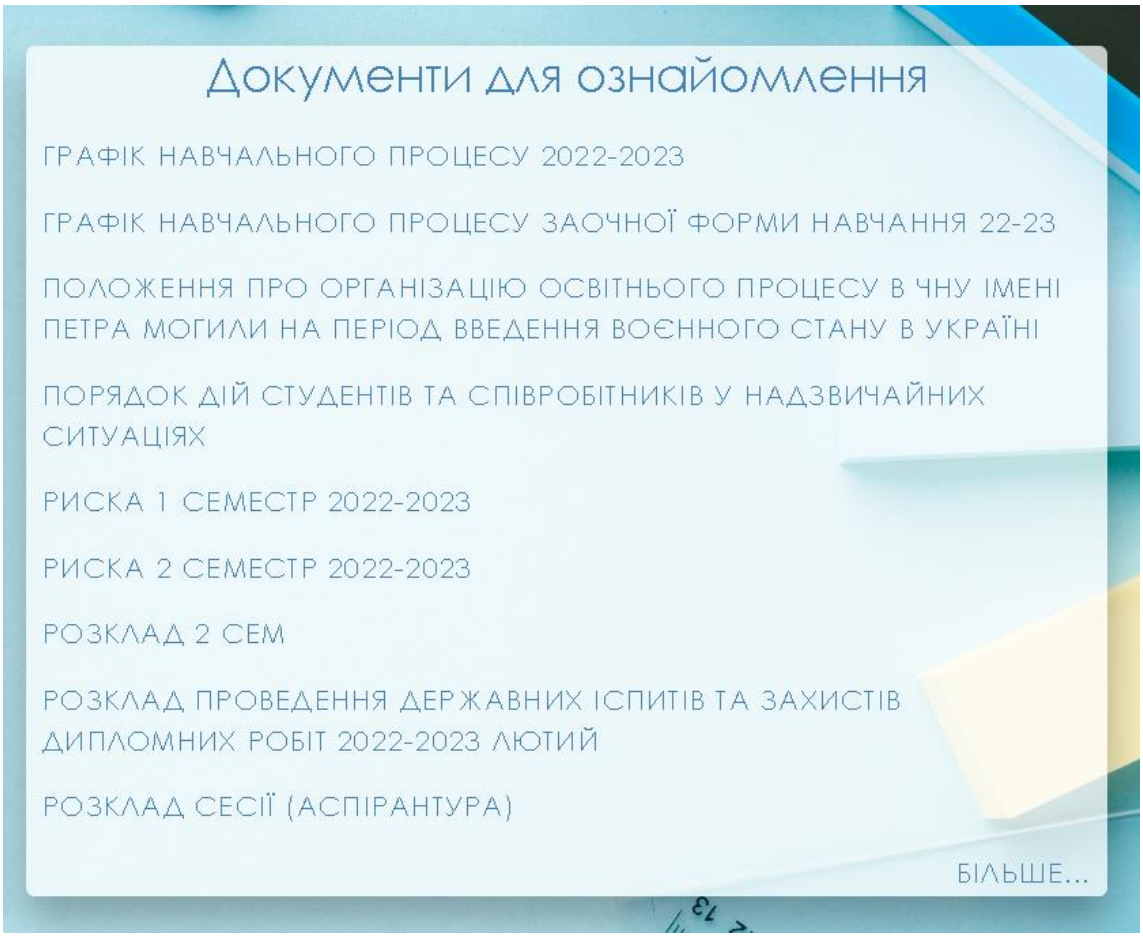


Рисунок 4.39 – Розділ «Документи для ознайомлення» на головній сторінці

На головній сторінці відображаються тільки декілька останніх завантажених файлів, перейшовши за посиланням «Більше…» користувач потрапляє на сторінку /Files, де він може переглянути всі файли (рис.4.40).

Ця сторінка містить також окремий розділ «Правила і положення» – в ній наведено документи, що визначають порядок і правила оцінювання і розрахунку рейтингу студентів, зокрема ті, за якими працює дана вебсистема.

Рисунок 4.40 – Вміст сторінки /Files

Адміністратор вебсистеми, та інші користувачі, хто має дозвіл на завантаження файлів, можуть додати файл за допомогою форми (рис.4.41). Також вони можуть видалити раніше завантажені файли натиснувши кнопку біля назви файлу (рис.4.42).

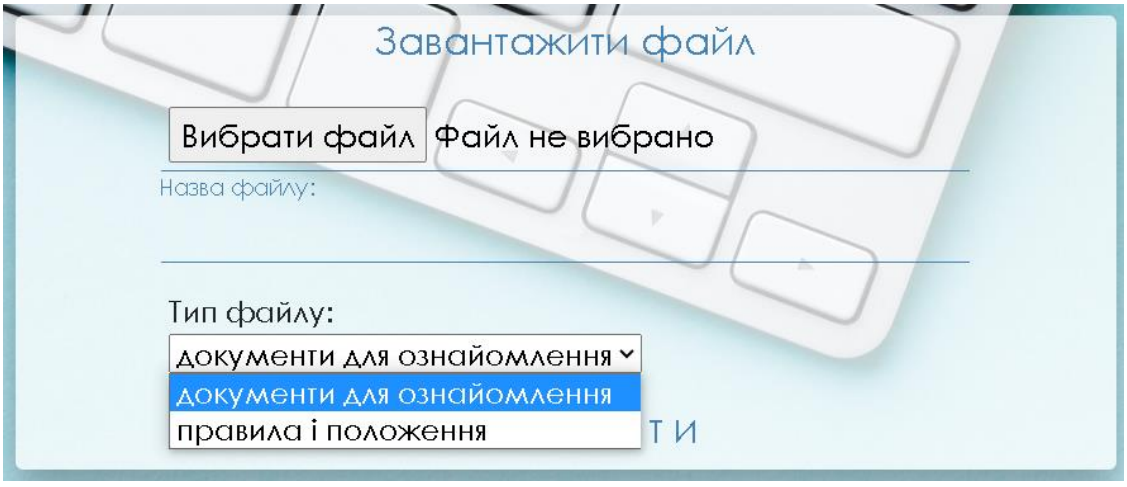


Рисунок 4.41 – Форма для завантаження файла

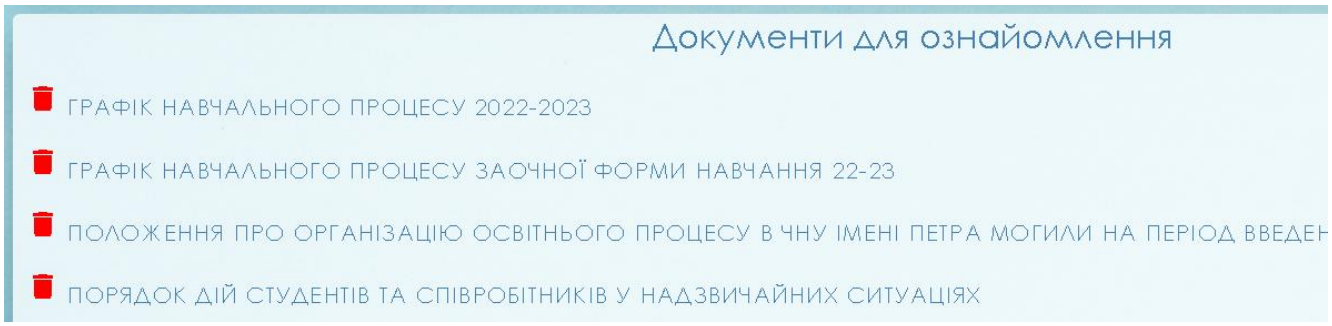


Рисунок 4.42 – Завантажені файли з кнопками для видалення

4.12 Помилки

В разі, якщо сторінки, на яку користувач намагається перейти, не існує, буде відображено сторінку, наведену на рис.4.43. Якщо користувач не має прав доступу до запитаної сторінки, буде відображено сторінку, наведену на рис.4.44.

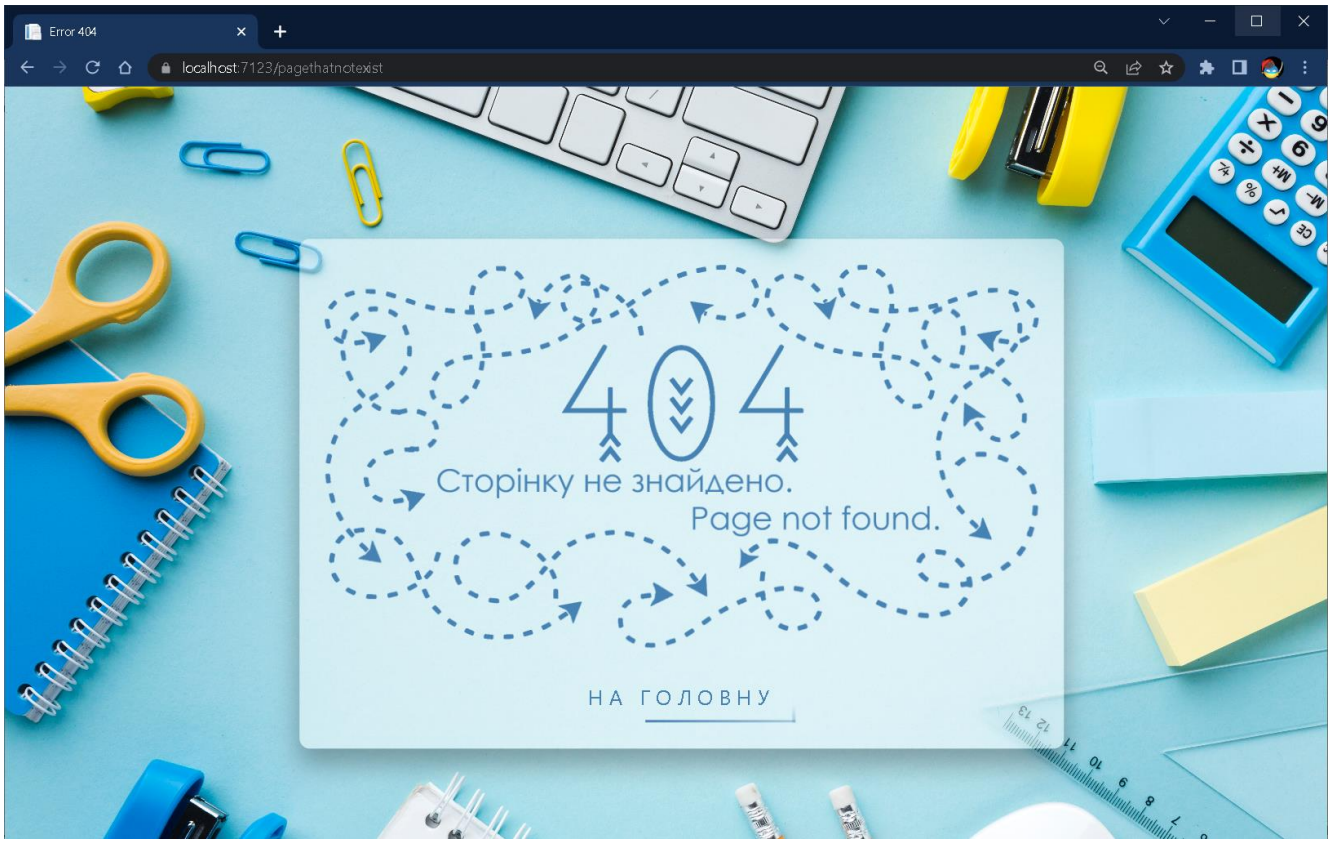


Рисунок 4.43 – Сторінка «Сторінку не знайдено»

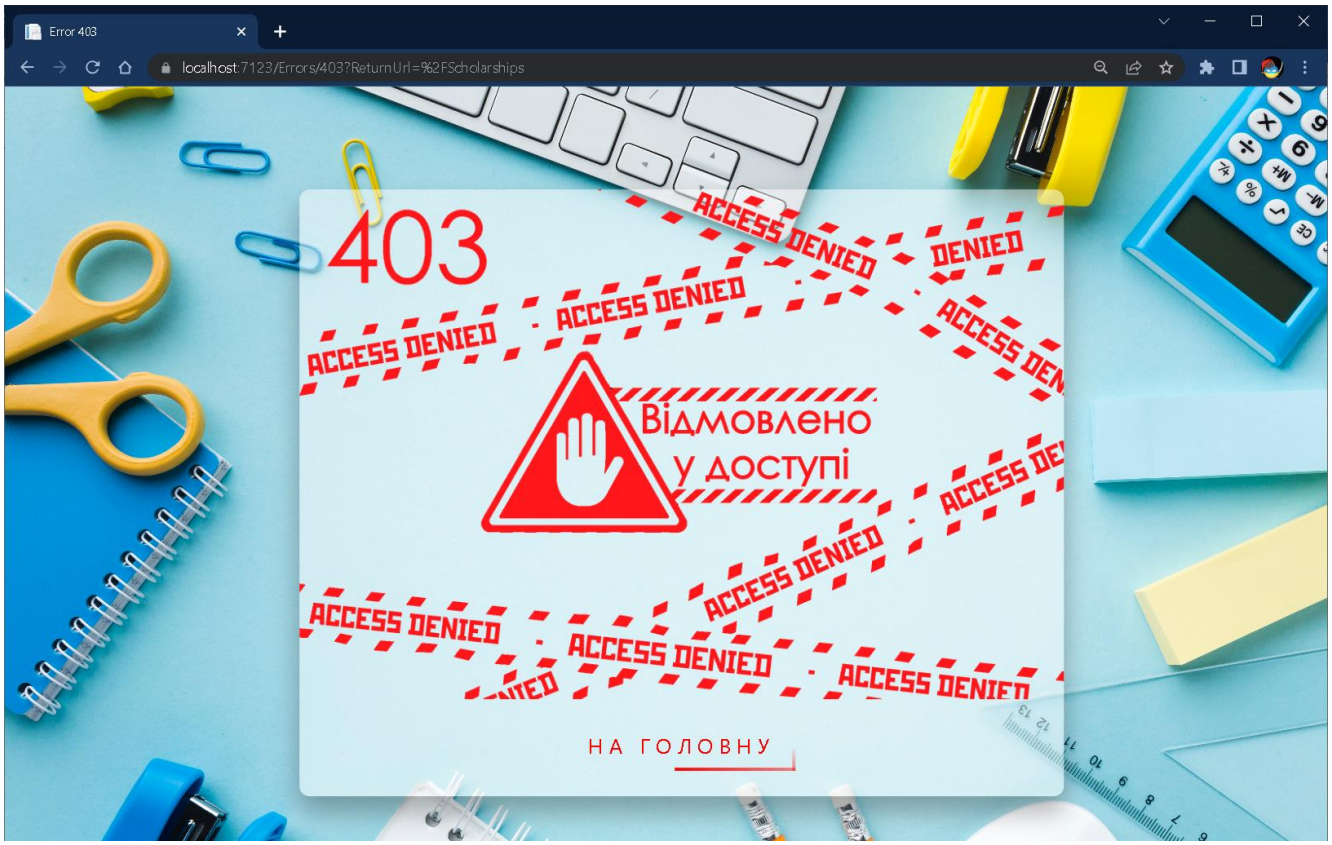


Рисунок 4.44 – Сторінка «Відмовлено у доступі»

Висновки до розділу 4

В даному розділі було детально описано і проілюстровано роботу вебсистеми, розглянуто використання вебсистеми користувачами з різними ролями, наведено відповідні скріншоти. Оцінюючи роботу вебсистеми можна зазначити, що вона відповідає поставленим вимогам.

ВИСНОВКИ

Метою бакалаврської кваліфікаційної роботи була автоматизація процесу рейтингування студентів Чорноморського національного університету імені Петра Могили. Серед основних вимог до системи було визначено наступні: збереження даних про студентів та їх бали, формування рейтингових списків студентів, надання студентам можливості перегляду своїх балів та рейтингу, надання викладачам можливості виставлення та перегляду балів зі своїх дисциплін, зручний функціонал призначення студентам стипендій і додаткових балів.

В ході виконання роботи було розглянуто нормативні документи та положення щодо правил оцінювання і підрахунку рейтингу студентів ЧНУ імені Петра Могили, на основі чого визначено вимоги до вебсистеми. Було визначено технології та фреймворки, які доцільно застосувати для розробки вебсистеми.

Було розроблено вебсистему із використанням фреймворку для розробки вебзастосунків ASP.NET Core 7, фреймворку для взаємодії з базою даних Entity Framework 7 та технології створення користувацького вебінтерфейсу Razor Pages.

Результатом роботи є автоматизована вебсистема рейтингування студентів, що забезпечує можливість зручного перегляду та виставлення балів студентів, підбивання підсумків у вигляді рейтингових списків та статистики успішності студентів, автоматичного визначення стипендії, на яку претендує студент.

Підсумовуючи, можна зазначити, що розроблений вебзастосунок відповідає поставленим вимогам, і мета бакалаврської кваліфікаційної роботи щодо автоматизації рейтингування студентів Чорноморського національного університету імені Петра Могили досягнута.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про вищу освіту : Закон України, редакція від 31.03.2023. URL: https://zakon.rada.gov.ua/laws/show/1556-18#Text (дата звернення: 04.05.2023).

2. Положення про організацію освітнього процесу в Чорноморському національному університеті імені Петра Могили. Офіційний сайт ЧНУ імені Петра Могили : вебсайт. URL: https://chmnu.edu.ua (дата звернення: 04.05.2023).

3. Положення про порядок призначення стипендії в Чорноморському національному університеті імені Петра Могили. Офіційний сайт ЧНУ імені Петра Могили : вебсайт. URL: https://chmnu.edu.ua (дата звернення: 05.05.2023).

4. Про затвердження переліку галузей знань і спеціальностей, за якими здійснюється підготовка здобувачів вищої освіти : Постанова Кабінету Міністрів України від 29 квітня 2015 року №266. URL: https://zakon.rada.gov.ua/laws/show/266-2015-%D0%BF#Text (дата звернення: 07.05.2023).

5. Довідник першокурсника. Офіційний сайт ЧНУ імені Петра Могили : вебсайт. URL: https://chmnu.edu.ua (дата звернення: 05.05.2023).

6. Положення про порядок і методику проведення заліків і екзаменів у Чорноморському національному університеті імені Петра Могили. Офіційний сайт ЧНУ імені Петра Могили : вебсайт. URL: https://chmnu.edu.ua (дата звернення: 05.05.2023).

7. Положення про систему рейтингової оцінки Чорноморського національного університету імені Петра Могили. Офіційний сайт ЧНУ імені Петра Могили : вебсайт. URL: https://chmnu.edu.ua (дата звернення: 05.05.2023).

8. Порядок призначення стипендії в Чорноморському національному університеті імені Петра Могили. Офіційний сайт ЧНУ імені Петра Могили : вебсайт. URL: https://chmnu.edu.ua (дата звернення: 08.05.2023).

9. Студентські стипендії. Міністерство освіти і науки України : вебсайт. URL: https://mon.gov.ua (дата звернення: 08.05.2023).

10. Рейтинг студента. Офіційний сайт ЧНУ імені Петра Могили : вебсайт. URL: https://chmnu.edu.ua (дата звернення: 08.05.2023).

11. MOODLE3 ЧНУ імені Петра Могили : вебсайт. URL: https://moodle3.chmnu.edu.ua (дата звернення: 12.05.2023).

12. What is three-tier architecture? ibm.com : вебсайт. URL: https://www.ibm.com/topics/three-tier-architecture (дата звернення: 27.05.2023).

13. Overview of ASP.NET Core. Microsoft ASP.NET documentation : вебсайт. URL: https://learn.microsoft.com/aspnet/core/ (дата звернення: 27.05.2023).

14. Руководство по ASP.NET Core 7. Metanit.com : вебсайт. URL: https://metanit.com/sharp/aspnet6/ (дата звернення: 28.05.2023).

15. Що таке ASP.NET? Принцип функціонування та моделі розробки. Highload : вебсайт. URL: https://highload.today/uk/shho-take-asp-net-printsip-funktsionuvannya-ta-modeli-rozrobki/ (дата звернення: 28.05.2023).

16. Лок Е. ASP.Net Core в действии / пер. с англ. Д.А.Беликова. – М.:ДМК Пресс, 2021. – 906 с.:ил. .

17. Огляд IDE Visual Studio 2022. visualstudio.microsoft.com : вебсайт. URL: https://visualstudio.microsoft.com/ (дата звернення: 28.05.2023).

18. Документація Bootstrap. Bootstrap21.org : вебсайт. URL: https://bootstrap21.org/uk/ (дата звернення: 28.05.2023).

19. Руководство по Razor Pages. Metanit.com : вебсайт. URL: https://metanit.com/sharp/razorpages (дата звернення: 28.05.2023).

20. Learn Razor Pages : вебсайт. URL: https://www.learnrazorpages.com (дата звернення: 28.05.2023).

21. Brind M. ASP.NET Core Razor Pages in Action: Manning, 2022. – 456с. .

22. Razor Pages CSS Isolation. DEV Community : вебсайт. URL: https://dev.to/karenpayneoregon/razor-pages-css-isolation-3f5 (дата звернення: 28.05.2023).

23. Изоляция CSS в Razor Pages. Habr : вебсайт. URL: https://habr.com (дата звернення: 28.05.2023)

24. Офіційний сайт Microsoft SQL Server : вебсайт. URL: https://www.microsoft.com/ru-ru/sql-server/sql-server-2019 (дата звернення: 28.05.2023).

25. Entity Framework Core. Microsoft documentation : вебсайт. URL: https://learn.microsoft.com/uk-ua/ef/core/ (дата звернення: 28.05.2023).

26. Смит Дж.П. Entity Framework Core в действии / пер. с англ. Д.А.Беликова. – М.:ДМК Пресс, 2022. – 690с.:ил. .

27. Руководство по Entity Framework Core 7. Metanit.com : вебсайт. URL: https://metanit.com/sharp/efcore/ (дата звернення: 28.05.2023).

28. Entity Framework Core. Entity Framework Tutorial : вебсайт. URL: https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx (дата звернення: 28.05.2023).

29. Entity Framework Core Documentation and Tutorials. Learn Entity Framework Core : вебсайт. URL: https://www.learnentityframeworkcore.com/ (дата звернення: 28.05.2023).

30. The C# type system. Microsoft documentation : вебсайт. URL: https://learn.microsoft.com /dotnet/csharp/fundamentals/types (дата звернення: 28.05.2023).

ДОДАТОК А

Класи для генерації таблиць БД

```
public class AdditionalPoint
    {
        public int Id { get; set; }
        public AdditionalPointType Type { get; set; }
        public DateTime Date { get; set; }
        public string Description { get; set; }
        public float Point { get; set; }
        public int StudentId { get; set; }
        public Student Student { get; set; }
    }
public class Cource
    {
        const int COUNT_OF_HOURS_IN_CREDIT = 30;
        public int Id { get; set; }
        public string Name { get; set; }
        public string? ShortName { get; set; }
        public ExamType FinalExamType { get; set; }
        public DateTime? FinalExamTime { get; set; }
        public int Semester { get; set; }
        public int Year { get; set; }
        public float Credits { get; set; }
        public int? GroupId { get;set; }
        public Group? Group { get; set; }
        public int? ExaminerId { get; set; }
        public Teacher? Examiner { get; set; }
        public int? TeacherId { get; set; }
        public Teacher? Teacher { get; set; }
        public bool IsCurrent() => Semester == AcademicTime.Now.SemesterNumber && Year ==
DateTime.Now.Year;
        public AcademicTime GetSemester() => new AcademicTime(Semester, Semester == 1 ?
Year : Year - 1);
        public int GetSemesterNumber() => (Group.GetCourceNumber(GetSemester()) - 1) * 2 +
Semester;
        public int GetCourceNumber() => Group.GetCourceNumber(GetSemester());
        public int GetHoursCount() => (int)(Credits * COUNT_OF_HOURS_IN_CREDIT);
        public int GetGroupNumber() => Group.GetCourceNumber(GetSemester()) * 100 +
(Group.Id % 100);
        public string GetFinalExamTypeName()
        {
            switch (FinalExamType)
            {
                case ExamType.ATTESTATION: return "атестація";
                case ExamType.CREDIT: return "залік";
                case ExamType.EXAM: return "іспит";
                case ExamType.PRACTICE: return "практика";
                default: return "";
            }
        }
        public string GetAbbreviation()
             {
                    if (ShortName is not null)
                    {
                return ShortName;
                    }
```

```
            string abbreviation = "";
            string prepositions = "і та з";
            foreach (var word in Name.Split(' ','-'))
                   {
                          if (!prepositions.Contains(word))
                          {
                    abbreviation += word[0];
                          }
                   }
            return abbreviation.ToUpper();
             }
        public List<Mark> Marks { get; set; }
    }
public class Department
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string ShortName { get; set; }
        public int FacultyId { get; set; }
        public Faculty Faculty { get; set; }
        public int HeadOfDepartmentId { get; set; }
        public Teacher? GetHeadOfDepartment()
             {
            return Teachers.Find(t => t.Id == HeadOfDepartmentId)??null;
             }

        public List<Teacher> Teachers { get; set; }
    }
[Table("Employees")]
    public class Employee : User
    {
        public string Post { get; set; }
    }
public class Faculty
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string ShortName { get; set; }
        public int? DeanId { get; set; }
        public Teacher? GetDeanOfFaculty()
             {
            List<Teacher> teachers = new List<Teacher>();
            foreach(var dep in Departments)
                   {
               teachers.AddRange(dep.Teachers);
                   }
            return teachers.FirstOrDefault(t => t.Id == DeanId);
             }
        public List<Department> Departments { get; set; } = new();
        public List<Group> Groups { get; set; } = new();
    }
public class Group
    {
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int Id { get; set; }
        public EducationForm EducationForm { get; set; }
```

```
public int SpecialtyId { get; set; }
public Specialty Specialty { get; set; }
public int? CouratorId { get; set; }
public Teacher? Courator { get; set; }

public int? FacultyId { get; set; }
public Faculty? Faculty { get; set; }
public List<Student> Students { get; set; } = new();
public List<Cource> Cources { get; set; } = new();
public int GetCurrentGroupNumber()
{
    return (Id % 100) + (GetCurrentCource() * 100);
}
public int GetCurrentCource()
{
    int currentYear = DateTime.Now.Year;
    int currentMonth = DateTime.Now.Month;
    int currentCource = currentYear - GetAdmissionYear();
    currentCource += ((Id % 1000) / 100)-1;
    if(currentMonth >= 9)
    {
        currentCource++;
    }
    return currentCource;
}

public int GetCourceNumber(AcademicTime semester)
     {
    return semester.BeginYear - GetAdmissionYear() + 1;
}
public int GetCurrentSemesterNumber()
     {
    return (GetCurrentCource() - 1) * 2 + AcademicTime.Now.SemesterNumber;
     }
private int GetAdmissionYear()
{
    return ((Id / 1000) % 100)+2000;
}

public string GetEducationForm()
{
    switch (EducationForm)
    {
        case EducationForm.FULL_TIME_DAY: return "денна";
        case EducationForm.FULL_TIME_EVENING: return "вечірня";
        case EducationForm.EXTERNAL: return "заочна";
        default: return "не вказано";
    }
}

public string GetEducationLevel()
{
    if(GetCurrentCource() <= 4)
    {
        return "бакалавр";
    }
    else
    {
```

```
                return "магістр";
            }
        }
    }
public class Mark
    {
        public int Id { get; set; }
        public DateTime ExamDate { get; set; }
        public int Point { get; set; }
        public bool IsRetaken { get; set; }
        public int StudentId { get; set; }
        public Student Student { get; set; }

        public int CourceId { get; set; }
        public Cource Cource { get; set; }
        public bool AffectsCurrentRating() => Cource.GetSemester() ==
AcademicTime.PreviousSemester;
        public bool IsCredited() => Point >= 60;
        public  ECTSpoint ConvertToECTS()
        {
            if(Point >= 90)
            {
                return ECTSpoint.A;
            }
            if(Point >= 85)
            {
                return ECTSpoint.B;
            }
            if(Point >= 75)
            {
                return ECTSpoint.C;
            }
            if(Point >= 70)
            {
                return ECTSpoint.D;
            }
            if(Point >= 60)
            {
                return ECTSpoint.E;
            }if(Point >= 35)
            {
                return ECTSpoint.FX;
            }
            return ECTSpoint.F;
        }

        public NationalScalePoint ConvertToNationalScale()
        {
            if(Cource.FinalExamType == ExamType.CREDIT)
                    {
                return Point >= 60 ? NationalScalePoint.CREDITED :
NationalScalePoint.NOT_CREDITED;
                    }
            if (Point >= 90)
            {
                return NationalScalePoint.EXELENT;
            }
            if (Point >= 75)
```

```
            {
                return NationalScalePoint.GOOD;
            }
            if (Point >= 60)
            {
                return NationalScalePoint.FAIR;
            }
            return NationalScalePoint.POOR;
        }

        public string ConvertToNationalScaleAsString()
        {
            if (Cource.FinalExamType == ExamType.CREDIT)
            {
                return Point >= 60 ? "зарах." : "не зарах.";
            }
            if (Point >= 90)
            {
                return "відмінно";
            }
            if (Point >= 75)
            {
                return "добре";
            }
            if (Point >= 60)
            {
                return "задовільно";
            }
            return "незадовільно";
        }
    }
}
    public class Permission
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public List<User> Users { get; set; } = new();
    }
public class Practice : Cource
      {
            public DateTime Begin { get; set; }
            public DateTime End { get; set; }
      }
public class ScholarshipType
    {
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int Id { get; set; }
        public string Name { get; set; }
        public int Size { get; set; }
        public List<Student> Students { get; set; } = new();
    }
public class Specialty
    {
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int Id { get; set; }
        public string Name { get; set; }
```

```
        public PointScale PointScale { get; set; }
        public List<Group> Groups { get; set; } = new();
    }
[Table("Students")]
    public class Student : User, IComparable, IComparer<Student>
    {
        public int StudentNumber { get; set; }
        public int GroupId { get; set; }
        public Group Group { get; set; }
        public bool IsOnBudget { get; set; }
        public int ScholarshipTypeId { get; set; }
        public ScholarshipType Scholarship { get; set; }
        public List<Mark> Marks { get; set; }
        public List<AdditionalPoint> AdditionalPoints { get; set; }

        public float CalculateRatingPoint()
        {
            List<Mark> marks = Marks.ToList().Where(x =>
x.AffectsCurrentRating()).ToList();
            float sumOfAdditionalPoints = AdditionalPoints.Where(x =>
AcademicTime.GetByDate(x.Date)==AcademicTime.PreviousSemester ).Sum(x => x.Point);
            if (marks.Any(mark => !mark.IsCredited() || mark.IsRetaken))
            {
                return 0;
            }
            marks.RemoveAll(x => x.Cource.FinalExamType == ExamType.ATTESTATION);
            return CalculateRatingPoint(CalculateWeightedAveragePoint(marks),
sumOfAdditionalPoints);
        }
        public float CalculateRatingPointForSemester(AcademicTime semester)
        {
            List<Mark> marks = Marks.Where(x => x.Cource.GetSemester() == semester
).ToList();
            float sumOfAdditionalPoints = AdditionalPoints.Where(x =>
AcademicTime.GetByDate(x.Date) == semester).Sum(x => x.Point);
            if (marks.Any(mark => !mark.IsCredited() || mark.IsRetaken))
            {
                return 0;
            }
            marks.RemoveAll(x => x.Cource.FinalExamType == ExamType.ATTESTATION);
            return CalculateRatingPoint(CalculateWeightedAveragePoint(marks),
sumOfAdditionalPoints);
        }
        public float CalculateWeightedAveragePoint()
        {
            return CalculateWeightedAveragePoint(Marks.Where(x =>
x.AffectsCurrentRating()).ToList());
        }
        public float CalculateWeightedAveragePoint(List<Mark> marks)
        {
            float sumOfCredits = marks.Sum(x => x.Cource.Credits);
            float ratingPoint = 0;
            foreach (var mark in Marks)
            {
                ratingPoint += mark.Point * mark.Cource.Credits / sumOfCredits;
            }
            return ratingPoint;
        }
```

```
        private float CalculateRatingPoint(float weightedAveragePoint , float
sumOfAdditionalPoints)
        {
            return weightedAveragePoint * 0.9f + sumOfAdditionalPoints;
        }
        public int CountOfAcademicArrears() => Marks.Where(x => x.Point < 60).Count();
        public int CountOfRetakenExams() => Marks.Where(x => x.AffectsCurrentRating() &&
x.IsRetaken).Count();
        public bool IsExcellentStudent() => Marks.Count()>0 && Marks.All(x =>
x.AffectsCurrentRating() && x.Point >= 90);
        public int GetCource() => Group.GetCurrentCource();
        public Specialty GetSpecialty() => Group.Specialty;
        public int CompareTo(object? obj)
        {
            Student st2 = obj as Student;
            if(st2 == null)
                return -1;
            }
            if(CountOfAcademicArrears() != st2.CountOfAcademicArrears()){
                return CountOfAcademicArrears() - st2.CountOfAcademicArrears();
            }
            if (IsOnBudget != st2.IsOnBudget) {
                return IsOnBudget ? -1 : 1;
            }
            if (Scholarship.Name == "Соціальна" && st2.Scholarship.Name != "Соціальна"){
                return 1;
            }
            if (Scholarship.Name != "Соціальна" && st2.Scholarship.Name == "Соціальна"){
                return -1;
            }
            return (int)(st2.CalculateRatingPoint()*100 - CalculateRatingPoint()*100);

        }
        public int Compare(Student? x, Student? y)
        {
            return x.CompareTo(y);
        }
        public int CompareWithoutRatingPoint(Student? x, Student? y)
        {
            if(x.GroupId == y.GroupId) {
                return x.Name.CompareTo(y.Name);
            }
            return x.GroupId.CompareTo(y.GroupId);
        }
        public override bool Equals(object? obj)
        {
            return obj is Student student &&
                   StudentNumber == student.StudentNumber;
        }
    }
[Table("Teachers")]
    public class Teacher : User, IComparable, IComparer<Student>
    {
        public string? Post { get; set; }
        public string? AcademicDegree { get; set; }
        public int DepartmentId { get; set; }
        public Department Department { get; set; }
```

```
        public Group? CurratedGroup { get; set; }
        List<Cource> Cources { get; set; } = new();
        public int Compare(Student? x, Student? y)
        {
            return x.CompareTo(y);
        }
        public int CompareTo(object? obj)
        {
            Teacher t = obj as Teacher;
            if (t == null)
            {
                return -1;
            }
            if(DepartmentId == t.DepartmentId)
            {
                return Name.CompareTo(t.Name);
            }
            return DepartmentId - t.DepartmentId;
        }
    }
public class User
    {
        public int Id { get; set; }
        public string Login { get; set; }
        public string Name { get; set; }
        public Role Role { get; set; }
        public string Password { get; set; }
        public string Contacts { get; set; }

        public List<Permission> Permissions { get; set; } = new();
        public string GetShorterName()
             {
            string familyname = Name.Split(' ')[0];
            string firstName = Name.Split(' ')[1];
            string patronimicName = Name.Split(' ')[2];
            return familyname + " " + firstName[0]+ "." + patronimicName[0]+".";
             }
    }
    public class StudentRatingDbContext : DbContext
    {
        public DbSet<AdditionalPoint> AdditionalPoints { get; set; } = null!;
        public DbSet<Cource> Cources { get; set; } = null!;
        public DbSet<Department> Departments  { get; set; } = null!;
        public DbSet<Employee> Employees { get; set; } = null!;
        public DbSet<Faculty> Faculties  { get; set; } = null!;
        public DbSet<Group> Groups { get; set; } = null!;
        public DbSet<Mark> Marks  { get; set; } = null!;
        public DbSet<Permission> Permissions  { get; set; } = null!;
        public DbSet<Practice> Practices { get; set; }=null!;
        public DbSet<ScholarshipType> ScholarshipTypes { get; set; } = null!;
        public DbSet<Specialty> Specialties  { get; set; } = null!;
        public DbSet<Student> Students { get; set; } = null!;
        public DbSet<Teacher> Teachers { get; set; } = null!;
        public DbSet<User> Users { get; set; } = null!;
        public StudentRatingDbContext() => Database.EnsureCreated();//creates db if it not
exist
        protected override void OnModelCreating(ModelBuilder modelBuilder)
```

```
        {
            modelBuilder.Entity<User>().Property(u => u.Password).HasDefaultValue(new
HashCalculator().CalculateHash("12345678"));
            modelBuilder.Entity<User>().Property(u => u.Contacts).HasDefaultValue("");
            modelBuilder.Entity<Specialty>().Property(s =>
s.PointScale).HasDefaultValue(PointScale.ONE_HUNDRED_POINT_SCALE);
            modelBuilder.Entity<Student>().Property(s =>
s.Role).HasDefaultValue(Role.STUDENT);
            modelBuilder.Entity<Teacher>().Property(t =>
t.Role).HasDefaultValue(Role.TEACHER);
            modelBuilder.Entity<Employee>().Property(a =>
a.Role).HasDefaultValue(Role.UNIVERSITY_EMPLOYEE);

        }
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {

optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=StudentRatingDb;Trust
ed_Connection=True;");
        }
    }
```

ДОДАТОК Б

Використані в проекті переліки (enum)

```
public enum AdditionalPointType
    {
        PUBLIC_ACTIVITY,
        SCIENTIFIC_WORK,
        SPORT_ACHIEVEMENTS
      }
public enum ECTSpoint
    {
        A,
        B,
        C,
        D,
        E,
        FX,
        F,
    }
public enum EducationForm
    {
        FULL_TIME_DAY,
        FULL_TIME_EVENING,
        EXTERNAL
    }
public enum ExamType
    {
        ATTESTATION,
        CREDIT,
        EXAM,
        PRACTICE
    }
public enum NationalScalePoint
    {
        EXELENT,
        GOOD,
        FAIR,
        POOR,
        CREDITED,
        NOT_CREDITED,
        ATESTATION,
    }
public enum PointScale
    {
        ONE_HUNDRED_POINT_SCALE=100,
        TWO_HUNDRED_POINT_SCALE=200
    }
public enum Role
    {
       ADMIN,
       STUDENT,
       TEACHER,
       UNIVERSITY_EMPLOYEE
    }
```

ДОДАТОК В

Лістинг коду оголошення класу Program

```
using StudentRatingSystemWebApp;
using StudentRatingSystemWebApp.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddRazorPages(options =>
{
    options.Conventions.AuthorizePage("/AdditionalPoints");
    options.Conventions.AuthorizePage("/ChangePassword");
    options.Conventions.AuthorizePage("/Cources");
    options.Conventions.AuthorizePage("/Departments");
    options.Conventions.AuthorizePage("/Faculties");
    options.Conventions.AuthorizePage("/Files");
    options.Conventions.AuthorizePage("/Group");
    options.Conventions.AuthorizePage("/Groups");
    options.Conventions.AuthorizePage("/Index");
    options.Conventions.AuthorizePage("/Marks");
    options.Conventions.AuthorizePage("/Profile");
    options.Conventions.AuthorizePage("/Rating");
    options.Conventions.AuthorizePage("/Scholarship");
    options.Conventions.AuthorizePage("/SignOut");
    options.Conventions.AuthorizePage("/Users");
});
builder.Services.AddAuthentication("Cookies")
    .AddCookie(
        options =>
        {
            options.LoginPath = "/SignIn";
            options.AccessDeniedPath = "/Errors/AccessDenied";
        }
    );
builder.Services.AddAuthorization();
builder.Services.AddSingleton<HashCalculator>();
builder.Services.AddSingleton<ScholarshipCalculator>();

var app = builder.Build();
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
    app.UseHsts();
}
app.UseStatusCodePagesWithReExecute("/Errors/{0}");

app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

app.MapRazorPages();

app.Run();
```