

**Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення**

**КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»**

**РОЗРОБКА СИСТЕМИ ВЕДЕННЯ ОЦІНЮВАННЯ СТУДЕНТІВ
ВИКОРИСТОВУЮЧИ C# ТА РЕЛЯЦІЙНУ МОДЕЛЬ ДАНИХ**

**DEVELOPMENT OF A STUDENT ASSESSMENT MANAGEMENT
SYSTEM USING C# AND A RELATIONAL DATA MODEL**

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-42
Холдаш Кирило Іванович

Керівник:
Гульчук Юрій Миколайович

Кваліфікаційну роботу
допущено до захисту
«__» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

«__» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Холдашу Кирилу Івановичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка системи ведення оцінювання студентів використовуючи C# та реляційну модель даних»

Керівник роботи: Гульчук Ю. М.

затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 541/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра «29» травня 2026 р.

3. Вихідні дані до роботи: предметна область оцінювання студентів у закладах освіти, технології розробки програмного забезпечення (C#, .NET), засоби роботи з базами даних (MySQL, Entity Framework Core).

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): аналіз сучасного стану проблем автоматизації оцінювання студентів, постановка завдання, формування вимог до розроблюваної інформаційної системи, обґрунтування вибору засобів і технологій розробки, практична реалізація об'єкта проєктування, розробка бази даних на основі реляційної моделі, обґрунтування принципів захисту інформаційної системи, аналіз можливостей подальшого розвитку та масштабування системи, тестування та налагодження програмного продукту.

5. Перелік графічного матеріалу: 13 рисунків, 3 таблиці, 14 лістингів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	Гульчук Ю. М.		
Специфікація вимог до розробленої системи	Гульчук Ю. М.		
Розробка об'єкта проектування	Гульчук Ю. М.		
Нормоконтроль	Суринович О. М.		
Гарант ОП	Ліщина Н. М.		
Показник запозичень тексту	_____ %		
Академічна доброчесність	Гульчук Ю. М.		

7. Дата видачі завдання «3» лютого 2026 р.

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 14.02.2026 р.	
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 03.03.2026 р.	
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 14.03.2026 р.	
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 14.04.2026 р.	
5	Практична реалізація об'єкта проектування та розробка бази даних	до 20.04.2026 р.	
6	Тестування та налагодження об'єкта проектування	до 04.05.2026 р.	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедрі	до 29.05.2026 р.	

Здобувач вищої освіти _____

Кирило ХОЛДАШ

Керівник кваліфікаційної роботи _____

Юрій ГУЛЬЧУК

АНОТАЦІЯ

Холдаш К. І. Розробка системи ведення оцінювання студентів використовуючи C# та реляційну модель даних. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел.

У першому розділі здійснено аналіз сучасного стану проблеми автоматизації оцінювання студентів, розглянуто існуючі програмні рішення та обґрунтовано доцільність створення власної інформаційної системи. В другому розділі визначено вимоги до програмного забезпечення, виконано проєктування системи з використанням UML-діаграм, а також обґрунтовано вибір засобів, методів і технологій реалізації. У третьому розділі представлено практичну реалізацію інформаційної системи, розробку бази даних, механізми захисту інформації та результати тестування і налагодження програмного продукту. У висновках узагальнено результати виконаної роботи, підтверджено досягнення поставленої мети та виконання всіх визначених завдань.

Ключові слова: інформаційна система, оцінювання студентів, електронний журнал, база даних, реляційна модель даних, C#, .NET, Windows Forms, Entity Framework, SQL, автоматизація навчального процесу.

ABSTRACT

Kholdash K. I. Development of a student assessment management system using C# and a relational data model. Manuscript. Qualification work for bachelor's degree in Software Engineering, specialty "Software Engineering". Lutsk National Technical University. Lutsk, 2026.

The bachelor's qualification work consists of an introduction, three chapters, conclusions, a list of references.

The first section analyzes the current state of the problem of automating student assessment, considers existing software solutions, and justifies the feasibility of creating your own information system. The second section defines software requirements, designs the system using UML diagrams, and justifies the choice of tools, methods, and implementation technologies. The third section presents the practical implementation of the information system, database development, information protection mechanisms, and the results of testing and debugging the software product. The conclusions summarize the results of the work performed, confirming the achievement of the set goal and the fulfillment of all assigned tasks.

Keywords: information system, student assessment, electronic journal, database, relational data model, C#, .NET, Windows Forms, Entity Framework, SQL, automation of the educational process.

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ	9
1.1 Аналіз сучасного стану проблеми	9
1.2 Постановка завдання на кваліфікаційну роботу бакалавра	13
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ...	15
2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення	15
2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання ...	21
РОЗДІЛ 3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ВЕДЕННЯ ОЦІНЮВАННЯ СТУДЕНТІВ	25
3.1 Практична реалізація об'єкта проєктування.....	25
3.2 Розробка бази даних	32
3.3 Захист інформаційно-комп'ютерної системи	37
3.4 Тестування та налагодження інформаційно-комп'ютерної системи	41
ВИСНОВКИ	45
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	47

ВСТУП

У сучасних умовах розвитку інформаційного суспільства цифровізація освітнього процесу є одним із ключових напрямів підвищення якості освіти. Заклади вищої освіти активно впроваджують інформаційні технології для автоматизації навчальної діяльності, управління освітніми процесами та підвищення ефективності взаємодії між учасниками навчального процесу [1].

Одним із важливих елементів освітнього процесу є система оцінювання студентів, яка дозволяє контролювати рівень засвоєння знань, формувати підсумкові результати навчання та здійснювати аналіз успішності. Проте на практиці в багатьох закладах освіти процес ведення оцінювання все ще частково або повністю здійснюється за допомогою паперових журналів або неінтегрованих електронних засобів, що знижує ефективність обробки даних, ускладнює доступ до інформації та створює ризики помилок.

Актуальність теми кваліфікаційної роботи полягає у необхідності створення сучасної інформаційної системи, яка забезпечує автоматизацію процесу ведення оцінювання студентів, централізоване зберігання даних та їх оперативну обробку. В умовах зростання обсягів інформації та підвищення вимог до якості освіти використання таких систем дозволяє значно підвищити точність обліку результатів навчання, забезпечити прозорість оцінювання та надати інструменти для аналітики успішності студентів.

Крім того, актуальність дослідження зумовлена необхідністю реалізації механізмів розмежування доступу до даних відповідно до ролей користувачів (адміністратор, викладач, студент), що є важливим аспектом інформаційної безпеки. Важливим також є забезпечення можливості формування звітів, статистики та експорту даних, що значно спрощує управління навчальним процесом.

Сучасні програмні рішення, такі як навчальні платформи та системи дистанційного навчання, не завжди повністю відповідають потребам конкретного закладу освіти або мають надлишкову функціональність. Тому

виникає потреба у створенні спеціалізованої системи, адаптованої до структури навчального процесу та вимог користувачів.

Об'єктом дослідження є процес ведення оцінювання студентів у закладах вищої освіти.

Предметом дослідження є методи та засоби розробки інформаційних систем для автоматизації обліку результатів навчання з використанням мови програмування C# та реляційної моделі даних.

Метою кваліфікаційної роботи є розробка програмної системи ведення оцінювання студентів, яка забезпечує автоматизацію обліку навчальних результатів, зручний доступ до даних та формування аналітичної інформації.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати сучасний стан автоматизації процесу оцінювання студентів;
- дослідити існуючі програмні рішення та визначити їх переваги і недоліки;
- сформулювати вимоги до розроблюваної інформаційної системи;
- обґрунтувати вибір технологій та засобів розробки;
- спроектувати структуру системи та базу даних;
- реалізувати програмну систему засобами мови C#;
- забезпечити механізми захисту даних і розмежування доступу;
- виконати тестування та налагодження програмного забезпечення.

Методи дослідження включають аналіз предметної області, методи об'єктно-орієнтованого програмування, проектування реляційних баз даних, використання ORM-технологій, а також методи тестування програмного забезпечення.

Практичне значення одержаних результатів полягає у створенні програмного продукту, який може бути використаний у закладах вищої освіти для автоматизації ведення електронного журналу оцінювання, підвищення ефективності роботи викладачів та забезпечення прозорості контролю успішності студентів.

РОЗДІЛ 1

АНАЛІЗ МЕТОДІВ РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасних умовах розвитку інформаційних технологій автоматизація освітнього процесу є одним із ключових напрямів підвищення якості навчання у закладах вищої освіти [1]. Однією з важливих складових цього процесу є організація системи оцінювання студентів, яка забезпечує контроль рівня знань, формування підсумкових результатів та аналіз успішності [2].

Аналіз наукових джерел показує, що питання автоматизації освітнього процесу та управління навчальними даними розглядається у працях вітчизняних і зарубіжних дослідників у галузі інженерії програмного забезпечення та інформаційних систем. Зокрема, у роботах, присвячених розробці інформаційних систем, значна увага приділяється використанню реляційних моделей даних, принципам побудови клієнт-серверних застосунків, а також застосуванню сучасних ORM-технологій для роботи з базами даних [3].

У нормативних та методичних джерелах (зокрема ISO/IEC 12207, ISO/IEC 25010) визначено основні вимоги до якості програмного забезпечення, серед яких особливе значення мають функціональність, надійність, зручність використання та безпека. Ці характеристики є критично важливими для систем, що працюють з персональними даними студентів та результатами навчання.

Патентний та інформаційний пошук показує, що існує значна кількість програмних рішень, призначених для автоматизації освітнього процесу. До них належать як універсальні системи управління навчанням (Learning Management Systems), так і спеціалізовані електронні журнали.

Серед найбільш поширених аналогів можна виділити (табл. 1.1):

– системи дистанційного навчання (LMS), такі як Moodle та Google Classroom. Вони забезпечують широкий спектр функцій, включаючи розміщення навчальних матеріалів, тестування та оцінювання студентів. Переваги: широкий

функціонал; підтримка дистанційного навчання; інтеграція з іншими сервісами. Недоліки: складність налаштування; надлишковість функцій для локальних задач; обмежена адаптація під конкретну структуру закладу освіти;

– електронні журнали та локальні інформаційні системи, що використовуються у навчальних закладах. Переваги: простота використання; орієнтація на конкретні потреби. Недоліки: обмежена функціональність; відсутність гнучкої системи аналітики; недостатній рівень безпеки;

– електронні таблиці (Microsoft Excel, Google Sheets). Переваги: простота реалізації; доступність. Недоліки: відсутність централізованої бази даних; складність контролю доступу; висока ймовірність помилок; відсутність автоматизованої аналітики.

Таблиця 1.1 – Порівняння існуючих аналогів систем оцінювання студентів

Назва аналога	Тип системи	Основні можливості	Переваги	Недоліки
Moodle	Система дистанційного навчання (LMS)	Розміщення навчальних матеріалів, тестування, електронний журнал, оцінювання, звітність	Широкий функціонал, підтримка дистанційного навчання, гнучкість налаштувань, велика поширеність	Складність адміністрування, надлишковість функцій для локального обліку оцінок, потреба в додатковому налаштуванні під структуру закладу
Google Classroom	Хмарна навчальна платформа	Створення курсів, видача завдань, виставлення оцінок, комунікація зі студентами	Простота використання, доступність через вебінтерфейс, інтеграція з сервісами Google	Обмежені можливості аналітики, недостатня адаптація під складну структуру факультетів, кафедр і груп, залежність від зовнішнього сервісу
Microsoft Excel / Google Sheets	Табличний інструмент обліку	Ведення списків студентів, внесення оцінок, ручне формування підсумків	Простота, доступність, швидке початкове впровадження	Відсутність централізованої БД, слабкий контроль доступу, висока ймовірність помилок, складність масштабування, відсутність автоматизованої рольової моделі

Продовження таблиці 1.1

Назва аналога	Тип системи	Основні можливості	Переваги	Недоліки
Університетські електронні журнали локальної розробки	Спеціалізована локальна система	Облік оцінок, ведення журналів, перегляд успішності	Орієнтація на конкретні потреби закладу, простіший інтерфейс, адаптація до локальних процесів	Часто обмежена функціональність, застарілий інтерфейс, слабкі засоби аналітики, складність супроводу та розширення
АСУ деканату / корпоративні освітні системи	Комплексна система управління навчальним процесом	Облік студентів, навчальних планів, результатів навчання, звітність	Комплексне охоплення процесів, централізоване зберігання даних, підтримка адміністративних задач	Висока складність впровадження, значна вартість, орієнтація більше на адміністративний облік, ніж на зручне щоденне ведення журналу викладачем
Розроблювана система ведення оцінювання студентів	Настільна інформаційна система	Ведення довідників, облік студентів і викладачів, управління заняттями, журнал оцінок, статистика, звіти, рольовий доступ, експорт CSV	Адаптованість до структури ЗВО, зручний інтерфейс, централізована реляційна БД, підтримка ролей, наявність статистики та звітів, орієнтація саме на задачу оцінювання	Потребує подальшого розвитку функціоналу, локальне розгортання, залежність від конкретного програмного середовища

Порівняльний аналіз існуючих аналогів показує, що універсальні навчальні платформи мають широкий набір можливостей, проте часто є надлишковими для задачі локального ведення оцінювання студентів. Табличні інструменти є простими у використанні, але не забезпечують цілісності даних, належного розмежування прав доступу та автоматизованої аналітики. Локальні електронні журнали та корпоративні системи частково вирішують ці проблеми, однак часто мають обмеження щодо зручності використання, масштабованості та розширення функціоналу.

Отже, доцільність розробки нової системи полягає у створенні спеціалізованого програмного продукту, який поєднуватиме переваги локальних систем і сучасних інформаційних технологій: зручний інтерфейс, реляційну модель даних, рольовий доступ, автоматизоване формування статистики та звітів, а також адаптацію до організаційної структури закладу вищої освіти.

Крім того, важливою проблемою є відсутність чіткої системи розмежування доступу до даних. У багатьох системах недостатньо реалізовані ролі користувачів, що може призводити до порушення конфіденційності інформації.

Також варто відзначити, що значна частина існуючих систем не забезпечує достатнього рівня аналітики, зокрема автоматичного формування статистики успішності, визначення середнього балу, розподілу оцінок за шкалами ECTS та національною шкалою.

З урахуванням зазначених недоліків виникає необхідність у створенні спеціалізованої інформаційної системи, яка поєднуватиме такі характеристики:

- гнучкість налаштування під структуру навчального закладу;
- підтримка реляційної моделі даних;
- зручний графічний інтерфейс користувача;
- реалізація ролей доступу (адміністратор, викладач, студент);
- автоматизація ведення електронного журналу;
- формування статистики та звітів;
- можливість експорту даних.

Отже, розробка системи ведення оцінювання студентів із використанням мови програмування C# та реляційної моделі даних є актуальною задачею, що дозволяє усунути недоліки існуючих рішень та підвищити ефективність управління освітнім процесом.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

На основі проведеного аналізу сучасного стану проблеми автоматизації процесу оцінювання студентів встановлено, що існуючі програмні рішення не повною мірою задовольняють вимоги закладів вищої освіти. Зокрема, вони або є надмірно складними та універсальними, або мають обмежений функціонал і не забезпечують достатнього рівня зручності, гнучкості та безпеки.

У зв'язку з цим виникає необхідність у розробці спеціалізованої інформаційної системи, яка забезпечує автоматизацію процесу ведення оцінювання студентів, централізоване зберігання даних та їх ефективну обробку.

Метою кваліфікаційної роботи є розробка програмної системи ведення оцінювання студентів із використанням мови програмування C# та реляційної моделі даних, яка забезпечує автоматизацію обліку результатів навчання, зручний доступ до інформації та формування аналітичних звітів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати сучасний стан автоматизації процесу оцінювання студентів;
- дослідити існуючі програмні рішення та визначити їх переваги і недоліки;
- сформулювати вимоги до розроблюваної інформаційної системи;
- обґрунтувати вибір технологій та засобів розробки;
- спроектувати структуру програмної системи та базу даних;
- реалізувати програмну систему засобами мови програмування C#;
- забезпечити механізми захисту даних та розмежування доступу користувачів;
- виконати тестування та налагодження розробленої системи.

Система повинна підтримувати рольову модель доступу, яка передбачає розмежування прав користувачів залежно від їх ролі, зокрема адміністратора, викладача та студента.

Для реалізації поставленого завдання передбачається використання мови програмування C#, платформи .NET, технології Windows Forms для створення графічного інтерфейсу, ORM-засобу Entity Framework Core для роботи з базою даних та системи управління базами даних MySQL.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

Проєктування програмного забезпечення є ключовим етапом розробки інформаційної системи, який визначає її архітектуру, функціональні можливості та подальшу ефективність використання. Для системи ведення оцінювання студентів цей етап має особливе значення, оскільки система працює з великими обсягами структурованих даних та передбачає взаємодію різних категорій користувачів.

Для розробки системи обрано об'єктно-орієнтований підхід, який дозволяє представити предметну область у вигляді взаємопов'язаних об'єктів [4]. Такий підхід забезпечує: модульність програмного забезпечення; повторне використання компонентів; зручність супроводу та масштабування системи.

Для формалізації моделей використано UML, що є загальноприйнятим стандартом у галузі розробки програмного забезпечення [5].

За допомогою діаграми варіантів використання відображено взаємодію основних користувачів із системою. Діаграма класів дозволяє описати ключові сутності програмного забезпечення та зв'язки між ними. ER-діаграма відображає структуру реляційної бази даних. Діаграми діяльності та послідовності використовуються для моделювання логіки процесу виставлення оцінок та взаємодії компонентів системи.

У межах даної роботи розробимо такі UML-діаграми:

- діаграма варіантів використання (Use Case) – для опису взаємодії користувачів із системою;
- діаграма класів – для представлення структури даних і основних сутностей програмного забезпечення;
- діаграма діяльності – для опису логіки виконання основних процесів у системі;

– діаграма послідовності – для відображення порядку взаємодії між користувачем, інтерфейсом, бізнес-логікою та базою даних під час виконання окремих операцій.

Вибір UML обґрунтовується її універсальністю, наочністю та можливістю формалізованого опису як статичних, так і динамічних аспектів системи.

Виявлення вимог до програмного забезпечення здійснювалось із використанням таких методів: аналіз предметної області (освітній процес, система оцінювання); порівняльний аналіз існуючих аналогів; визначення ролей користувачів; функціональна декомпозиція системи.

У результаті було виділено три основні категорії користувачів:

- адміністратор – відповідає за управління системою та довідниками;
- викладач – веде журнал оцінювання;
- студент – переглядає результати навчання.

Основною ціллю системи є створення ефективного програмного засобу для автоматизації процесу ведення оцінювання студентів.

Для досягнення цієї цілі необхідно реалізувати такі задачі:

- організувати централізоване зберігання навчальних даних;
- забезпечити ведення електронного журналу;
- реалізувати механізми доступу до даних відповідно до ролей;
- автоматизувати формування підсумкових оцінок;
- забезпечити аналіз успішності студентів;
- реалізувати функції пошуку та фільтрації;
- забезпечити можливість формування звітів та експорту даних.

Розроблювана система повинна забезпечувати виконання таких функцій:

Робота з користувачами: реєстрація та авторизація; визначення ролі користувача; управління обліковими записами.

Робота з довідниками: створення та редагування факультетів, кафедр, спеціальностей; управління академічними групами; додавання дисциплін.

Облік навчального процесу: створення занять; прив'язка викладачів до дисциплін; ведення електронного журналу.

Оцінювання: внесення оцінок; редагування та видалення оцінок; автоматичний розрахунок підсумкових результатів.

Аналітика та звітність: перегляд успішності студентів; формування статистики; генерація звітів; експорт у формат CSV.

Пошук і фільтрація: пошук студентів; фільтрація за групами, дисциплінами, викладачами.

Система повинна відповідати наступним критеріям якості:

- надійність – система повинна працювати без збоїв та втрати даних;
- безпека – доступ до даних має бути обмежений ролями;
- зручність використання (Usability) – інтерфейс має бути інтуїтивно зрозумілим;
- продуктивність – швидке оброблення запитів;
- масштабованість – можливість розширення функціоналу;
- цілісність даних – забезпечення коректності зв'язків у БД.

Для аналізу поведінки системи та визначення її реакції на зовнішні впливи використано модель «подія – відгук», результати якої наведено у таблиці 2.1.

Таблиця 2.1 – Модель «подія – відгук»

Подія	Відгук системи
Авторизація користувача	Перевірка облікових даних та визначення ролі
Додавання студента	Збереження запису в базі даних
Внесення оцінки	Оновлення журналу
Запит статистики	Формування аналітичного звіту
Експорт даних	Формування CSV-файлу

Взаємодія користувачів із системою реалізується через такі основні прецеденти: авторизація; управління даними; ведення оцінювання; перегляд статистики; формування звітів.

Структура системи базується на таких основних сутностях: користувач; студент; викладач; дисципліна; група; заняття; оцінка.

Для опису взаємодії користувачів із розроблюваною системою використано діаграму варіантів використання, яка відображає основні функції системи та ролі користувачів. На діаграмі показано, які дії можуть виконувати адміністратор, викладач і студент у межах системи (рис. 2.1).

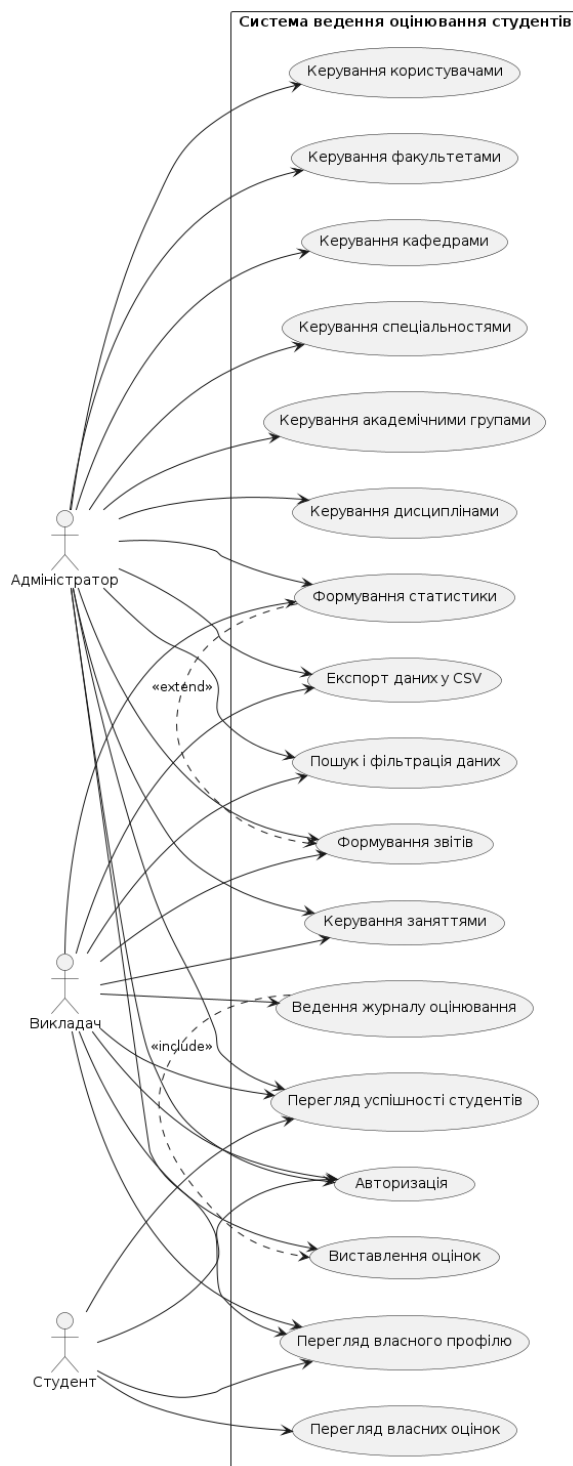


Рисунок 2.1 – Діаграма варіантів використання системи ведення оцінювання студентів

Інтерфейс системи реалізовано у вигляді настільного застосунку, що містить: форму входу; головне меню; робочу область; таблиці для відображення даних; форми редагування.

Основні принципи UX/UI: мінімізація дій користувача; логічна структура меню; швидкий доступ до функцій; візуальна зрозумілість.

Для формалізованого опису структури програмного забезпечення та взаємозв'язків між основними сутностями предметної області побудовано діаграму класів. Вона дозволяє відобразити основні класи системи, їх атрибути та зв'язки між ними (рис. 2.2).

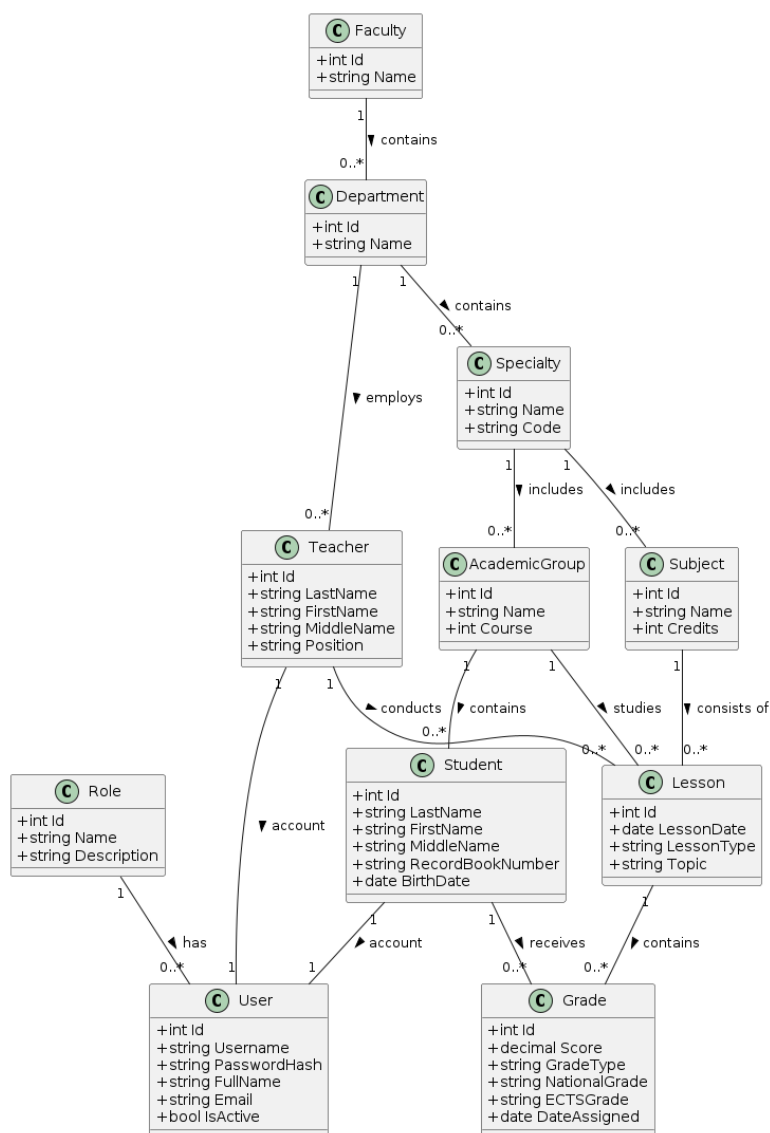


Рисунок 2.2 – Діаграма класів системи ведення оцінювання студентів

Для опису логіки процесу виставлення оцінки студенту використано діаграму діяльності. Вона відображає послідовність дій викладача та реакцію системи на кожному етапі виконання операції (рис. 2.3).

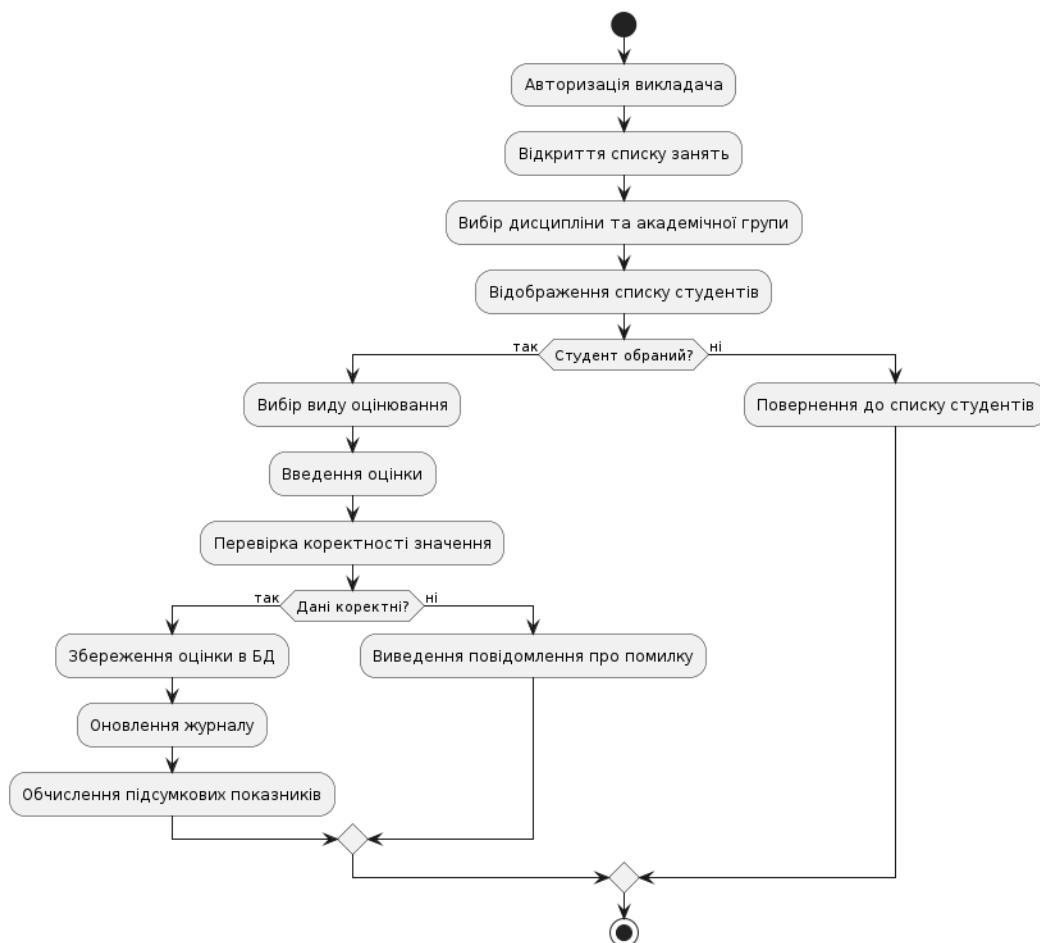


Рисунок 2.3 – Діаграма діяльності процесу виставлення оцінки

У системі реалізовано такі основні інформаційні потоки: введення даних користувачем; обробка інформації; збереження в базі даних; формування результатів; відображення інформації.

Для детального опису взаємодії між компонентами системи під час внесення оцінки використано діаграму послідовності. Вона демонструє порядок обміну повідомленнями між користувачем, інтерфейсом системи, бізнес-логікою та базою даних (рис. 2.4).

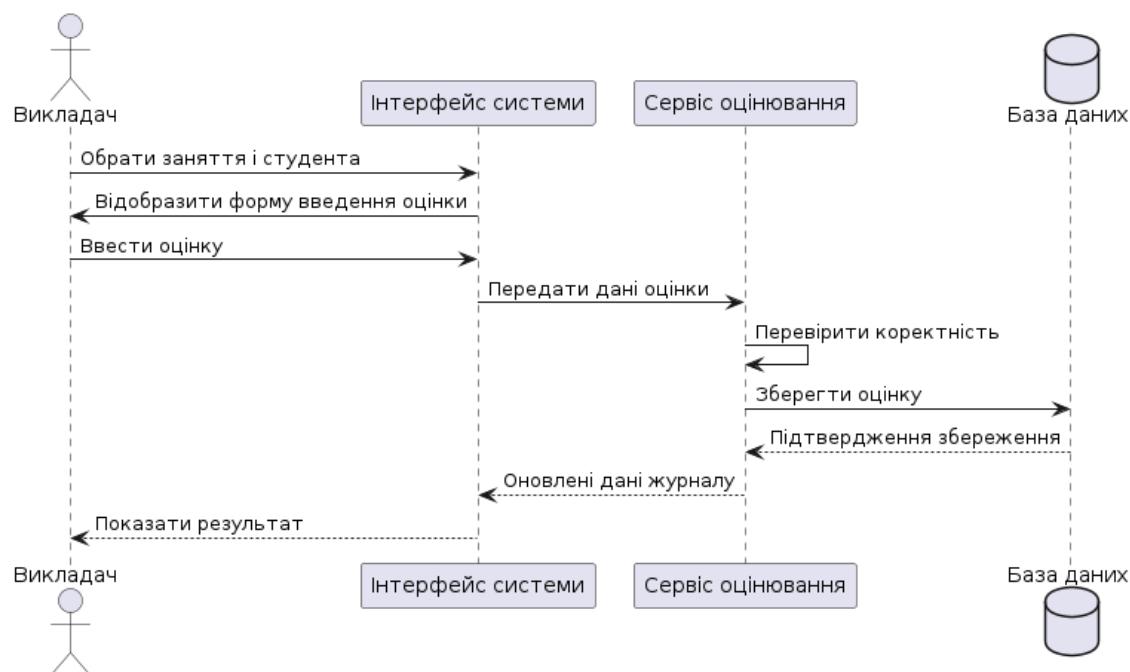


Рисунок 2.4 – Діаграма послідовності внесення оцінки

Запропонований підхід до проєктування дозволяє: створити гнучку та масштабовану систему; забезпечити ефективне управління даними; реалізувати зручний інтерфейс; підвищити надійність та безпеку; забезпечити відповідність сучасним вимогам до інформаційних систем.

Отже, сформована модель програмного забезпечення є обґрунтованою та придатною для реалізації системи ведення оцінювання студентів.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Розробка інформаційної системи ведення оцінювання студентів потребує обґрунтованого вибору програмних засобів, методів реалізації та алгоритмів обробки даних. Такий вибір визначає не лише функціональні можливості системи, але й її продуктивність, надійність, зручність використання та можливість подальшого розвитку.

На сучасному етапі розвитку інформаційних технологій існує значна кількість технологій, які можуть бути використані для створення подібних

систем. Вибір конкретних засобів повинен базуватись на аналізі їхніх можливостей, сумісності між собою та відповідності вимогам предметної області. Зокрема, для розробки інформаційних систем широко застосовуються такі мови програмування, як Java, Python та C#. Кожна з них має свої переваги та обмеження. Java характеризується високою портативністю та надійністю, однак потребує значних ресурсів і є менш зручною для швидкої розробки настільних застосунків. Python дозволяє швидко створювати програмні рішення, проте має обмеження у продуктивності та реалізації складних графічних інтерфейсів. У свою чергу, мова C# є оптимальним вибором для розробки настільних інформаційних систем, оскільки забезпечує високу продуктивність, тісну інтеграцію з платформою .NET та підтримку об'єктно-орієнтованого підходу [6, 7].

Не менш важливим є вибір технології створення інтерфейсу користувача. Веб-технології дозволяють створювати універсальні системи з доступом через браузер, однак потребують розгортання серверної частини та ускладнюють локальне використання системи. Серед настільних технологій виділяють WPF та Windows Forms [8]. WPF забезпечує широкі можливості для створення складних інтерфейсів, проте потребує більше часу на розробку. Windows Forms, навпаки, є простішим і більш швидким у реалізації, що робить його доцільним вибором для створення прикладної інформаційної системи з чітко визначеним функціоналом.

Значну роль у функціонуванні системи відіграє база даних. Для зберігання навчальної інформації доцільно використовувати реляційні системи управління базами даних, такі як MySQL або PostgreSQL. У даній роботі обрано MySQL, оскільки вона поєднує у собі високу продуктивність, простоту використання та підтримку транзакцій, що є важливим для забезпечення цілісності даних. Крім того, MySQL добре інтегрується з платформою .NET, що спрощує процес розробки [9, 10].

Для взаємодії з базою даних використано технологію об'єктно-реляційного відображення Entity Framework Core [9]. Використання ORM

дозволяє працювати з даними на рівні об'єктів, що значно спрощує розробку, зменшує кількість помилок та підвищує читабельність коду. Замість написання складних SQL-запитів розробник має можливість використовувати мову LINQ, яка інтегрована у C#.

У процесі розробки системи застосовано об'єктно-орієнтований підхід, який дозволяє представити предметну область у вигляді сукупності взаємопов'язаних об'єктів. Такий підхід забезпечує модульність системи, спрощує її розширення та супровід. Для організації даних використано реляційну модель, яка дозволяє забезпечити логічну структуру та цілісність інформації.

Важливим аспектом реалізації системи є застосування алгоритмів, що забезпечують виконання основних функцій. Зокрема, алгоритм авторизації користувача передбачає перевірку введених облікових даних, визначення ролі користувача та надання доступу до відповідного функціоналу системи. Це дозволяє реалізувати механізм розмежування доступу та забезпечити безпеку даних.

Алгоритм виставлення оцінки реалізує процес взаємодії викладача з системою. Спочатку викладач обирає дисципліну та академічну групу, після чого система відображає список студентів. Після вибору конкретного студента виконується введення оцінки, яка проходить перевірку на коректність. У разі успішної перевірки дані зберігаються у базі даних, а інформація в електронному журналі оновлюється. Такий підхід забезпечує цілісність даних і виключає можливість введення некоректної інформації.

Алгоритм формування підсумкової оцінки базується на обробці накопичених результатів навчання. Система отримує всі оцінки студента, обчислює середній бал та визначає відповідність шкалам оцінювання. На основі отриманих значень формується підсумкова оцінка, яка відображається користувачу. Використання такого алгоритму дозволяє автоматизувати процес оцінювання та зменшити вплив людського фактора.

Окрім цього, у системі реалізовано алгоритми пошуку та фільтрації даних, які дозволяють швидко знаходити необхідну інформацію за заданими

критеріями. Це є важливим для забезпечення зручності роботи користувачів та підвищення ефективності використання системи.

Для формалізації структури та поведінки системи використовуються UML-діаграми, зокрема діаграма класів, діаграма діяльності та діаграма послідовності. Вони дозволяють наочно представити архітектуру системи, взаємодію її компонентів та логіку виконання основних процесів.

Таким чином, обрані засоби, методи та алгоритми забезпечують ефективну реалізацію інформаційної системи ведення оцінювання студентів. Вони дозволяють досягти високої продуктивності, надійності та зручності використання, а також забезпечують можливість подальшого розвитку програмного продукту відповідно до потреб користувачів.

РОЗДІЛ 3

РОЗРОБКА СИСТЕМИ ВЕДЕННЯ ОЦІНЮВАННЯ СТУДЕНТІВ

3.1 Практична реалізація об'єкта проєктування

У межах кваліфікаційної роботи було розроблено програмну систему ведення оцінювання студентів, реалізовану у вигляді настільного застосунку з використанням мови програмування C# та платформи .NET. Система призначена для автоматизації процесу ведення електронного журналу та обліку результатів навчання студентів.

Програмний продукт розроблено у середовищі Microsoft Visual Studio із застосуванням технології Windows Forms для створення графічного інтерфейсу користувача. Архітектура системи побудована за принципами об'єктно-орієнтованого програмування із розподілом на логічні компоненти: інтерфейс користувача, бізнес-логіку та рівень доступу до даних.

Для взаємодії з базою даних використано ORM-технологію Entity Framework Core, що дозволило спростити реалізацію операцій додавання, редагування, видалення та пошуку даних. Як систему управління базами даних використано MySQL.

Після запуску програми користувачу відображається форма авторизації (рис. 3.1), на якій необхідно ввести облікові дані. Система перевіряє введену інформацію та визначає роль користувача, після чого відкриває відповідний функціонал. Інтерфейс системи адаптується залежно від ролі користувача, що забезпечує зручність використання та підвищує рівень безпеки.

Система забезпечує можливість ведення обліку студентів, внесення та редагування оцінок, перегляду статистичних даних і пошуку необхідної інформації. Реалізований функціонал дозволяє підвищити ефективність обробки навчальних даних, зменшити кількість помилок під час ведення документації та спростити роботу користувачів із системою.

Вхід у систему

Система оцінювання студентів
Авторизація користувача

Логін
student1

Пароль
••••

☐ Показати пароль

Увійти

Тестовий вхід: admin / admin

Рисунок 3.1 – Форма авторизації користувача

Після успішної авторизації користувач потрапляє до головного вікна програми (рис. 3.2), яке містить меню навігації та робочу область. Основні функціональні модулі системи реалізовані у вигляді окремих форм, що дозволяє логічно розділити функціональність та спростити роботу користувача.

Система оцінювання студентів

Користувач: Головний адміністратор
Роль: SuperAdmin

Система оцінювання студентів

Факультети
Кафедри
Спеціальності
Групи
Студенти
Викладачі
Предмети
Заняття
Журнал оцінок
Статистика
Звіти

Вийти

Головна сторінка

Оберіть потрібний розділ у лівому меню. Набір доступних функцій визначається роллю користувача.

Рисунок 3.2 – Головне вікно інформаційної системи

У системі передбачено можливість управління довідниками навчального процесу, зокрема факультетами, кафедрами, спеціальностями, академічними групами та дисциплінами. Користувач може додавати, редагувати та видаляти записи, що забезпечує актуальність даних. Управління обліковими записами користувачів реалізовано через відповідний інтерфейс (рис. 3.3), який дозволяє створювати, редагувати та видаляти записи.

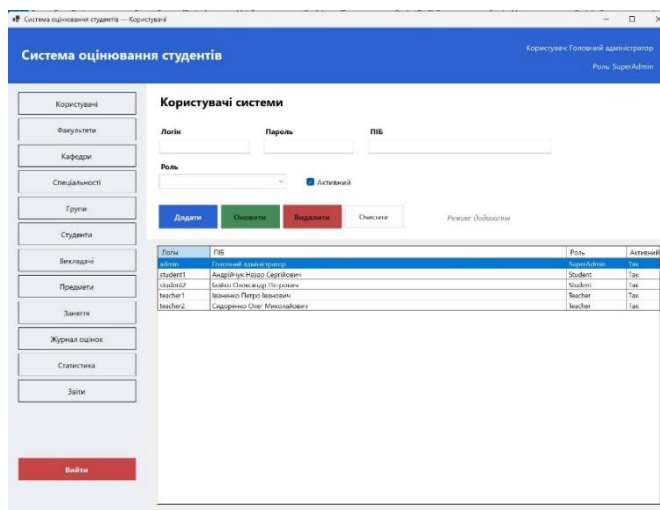


Рисунок 3.3 – Вікно управління користувачами системи

Окрему увагу приділено модулю роботи зі студентами. Система дозволяє зберігати детальну інформацію про студентів, включаючи їх особисті дані та належність до академічних груп (рис. 3.4).

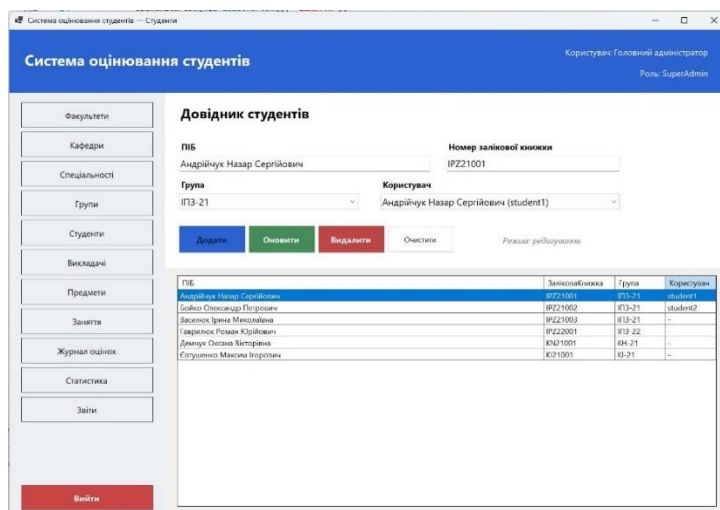


Рисунок 3.4 – Вікно управління даними студентів

Для додавання нового студента використовується ORM-технологія Entity Framework, що дозволяє працювати з базою даних у вигляді об'єктів (ліст. 3.1).

Лістинг 3.1 – Додавання студента до бази даних

```
public void AddStudent(Student student)
{
    _context.Students.Add(student);
    _context.SaveChanges();
}
```

Кінець лістингу 3.1

Передбачено можливість перегляду профілю студента, що дозволяє швидко отримати необхідну інформацію. Перегляд детальної інформації про студента здійснюється у спеціальному вікні профілю (рис. 3.5).

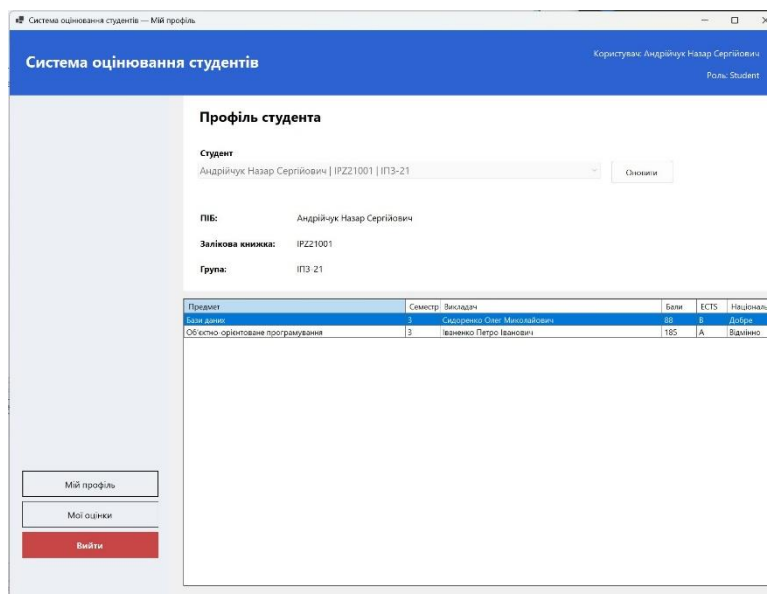


Рисунок 3.5 – Профіль студента у системі

Модуль ведення занять забезпечує створення та редагування інформації про навчальні заняття, включаючи дату проведення, дисципліну, викладача та групу. Це дозволяє організувати структуру навчального процесу та забезпечити зв'язок між усіма його елементами. Модуль управління навчальними заняттями

представлений у вигляді окремого інтерфейсу (рис. 3.6), що дозволяє структурувати навчальний процес.

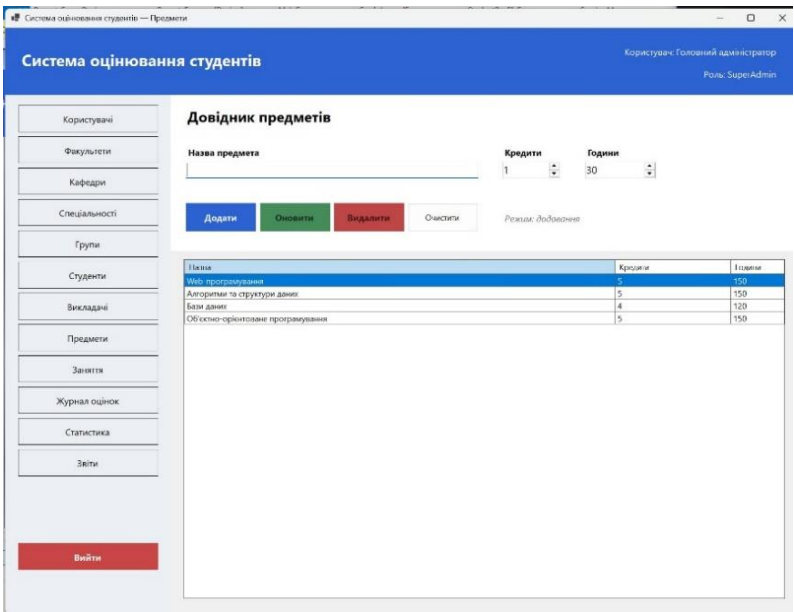


Рисунок 3.6 – Вікно управління навчальними заняттями

Ключовим компонентом системи є електронний журнал оцінювання (рис. 3.7), який дозволяє викладачам виставляти оцінки студентам.

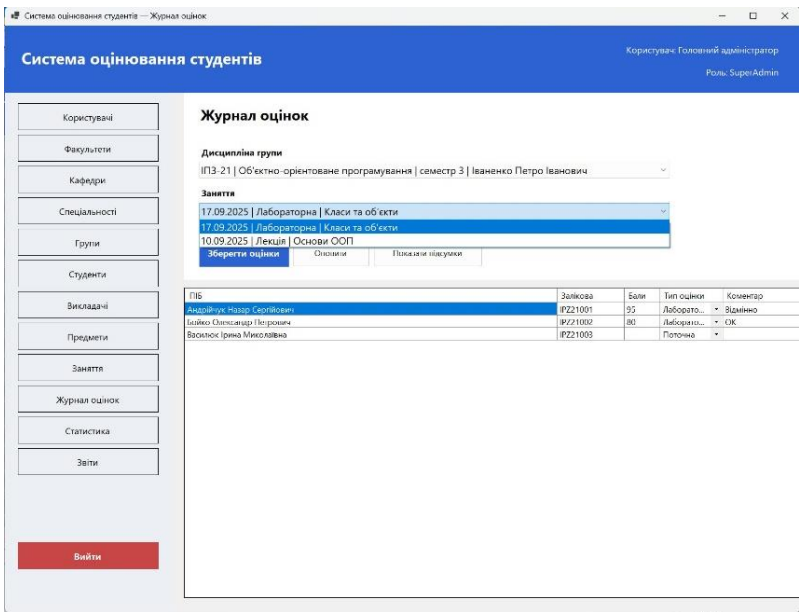


Рисунок 3.7 – Електронний журнал оцінювання студентів

Виставлення оцінки студенту реалізовано шляхом створення нового запису в таблиці оцінок (ліст. 3.2).

Лістинг 3.2 – Реалізація виставлення оцінки

```
public void AddGrade(int studentId, int lessonId, decimal score)
{
    if (score < 0 || score > 100)
        throw new Exception("Некоректне значення оцінки");

    var grade = new Grade
    {
        StudentId = studentId,
        LessonId = lessonId,
        Score = score,
        DateAssigned = DateTime.Now
    };

    _context.Grades.Add(grade);
    _context.SaveChanges();
}
```

Кінець лістингу 3.2

Після збереження оцінок система автоматично оновлює журнал та обчислює відповідні показники. Для аналізу успішності студентів використовується обчислення середнього балу (ліст. 3.3).

Лістинг 3.3 – Обчислення середнього балу студента

```
public double CalculateAverage(int studentId)
{
    return _context.Grades
        .Where(g => g.StudentId == studentId)
        .Average(g => (double)g.Score);
}
```

Кінець лістингу 3.3

У системі реалізовано функціонал перегляду успішності студентів, який дозволяє аналізувати результати навчання як окремого студента, так і групи в цілому. Передбачено формування статистичних звітів, що відображають рівень

успішності та розподіл оцінок. Аналіз результатів навчання реалізовано у вигляді статистичного модуля (рис. 3.8), який відображає рівень успішності студентів.

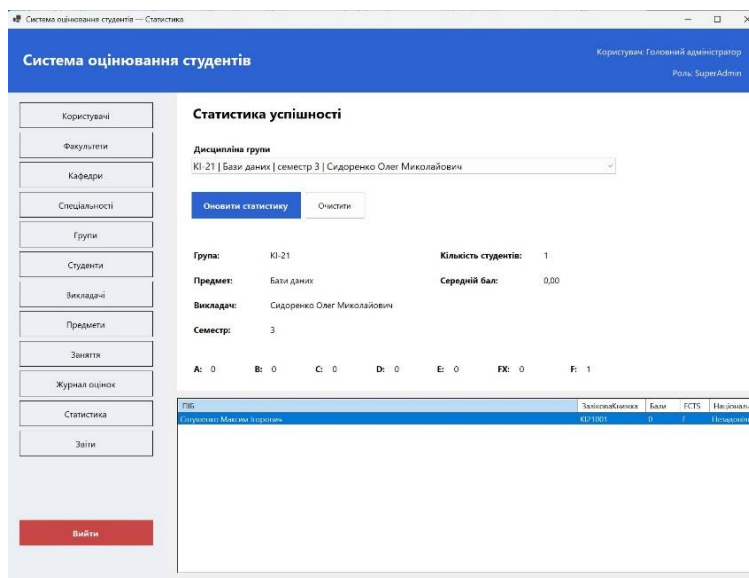


Рисунок 3.8 – Вікно перегляду статистики успішності студентів

Додатково реалізовано можливість експорту даних у формат CSV, що дозволяє використовувати результати у сторонніх програмних засобах, зокрема для подальшого аналізу або формування звітності (ліст. 3.4).

Лістинг 3.4 – Експорт даних у формат CSV

```
public void ExportToCsv(List<Student> students, string filePath)
{
    var lines = new List<string>();
    foreach (var s in students)
    {
        lines.Add($"{s.LastName},{s.FirstName},{s.RecordBookNumber}");
    }
    File.WriteAllLines(filePath, lines);
}
```

Кінець лістингу 3.4

Експорт результатів навчання у формат CSV реалізовано у відповідному модулі системи (рис. 3.9).

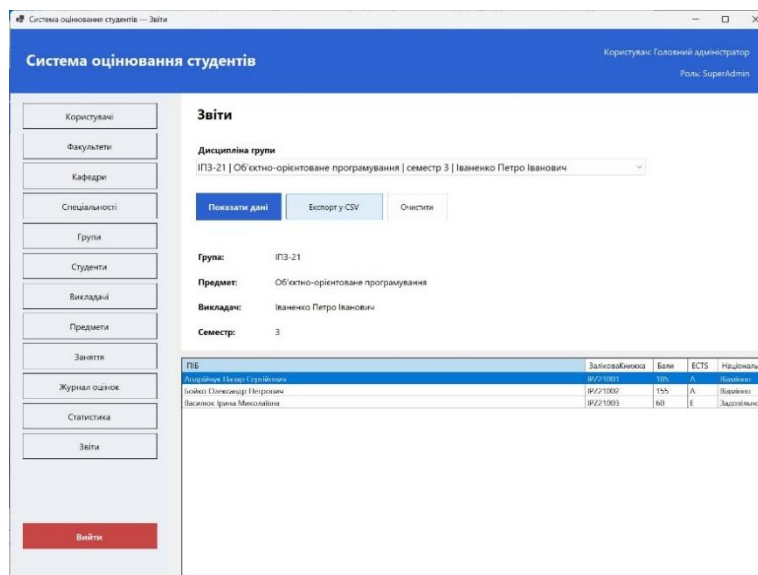


Рисунок 3.9 – Експорт результатів у формат CSV

Важливим елементом системи є реалізація рольової моделі доступу. Адміністратор має повний доступ до всіх функцій системи, викладач може працювати лише з призначеними дисциплінами та групами, а студент має доступ лише до власних результатів навчання.

Таким чином, було створено повноцінну інформаційну систему ведення оцінювання студентів, яка відповідає поставленим вимогам, забезпечує автоматизацію основних процесів навчального обліку та має зручний інтерфейс для користувачів.

3.2 Розробка бази даних

Для організації зберігання даних використано реляційну модель, яка дозволяє ефективно структурувати інформацію та встановлювати зв'язки між її елементами. Для реалізації бази даних обрано систему управління базами даних MySQL, яка забезпечує високу продуктивність, підтримку транзакцій та механізми забезпечення цілісності даних. Взаємодія з базою даних здійснюється за допомогою ORM-технології Entity Framework Core, що дозволяє працювати з даними на рівні об'єктів та спрощує реалізацію операцій додавання, редагування, видалення та пошуку інформації.

У процесі проєктування бази даних було визначено основні сутності предметної області, які відповідають об'єктам реального навчального процесу. До них належать: користувач (User); студент (Student); викладач (Teacher); академічна група (Group); дисципліна (Subject); заняття (Lesson); оцінка (Grade).

Для наочного представлення структури бази даних та зв'язків між таблицями побудовано ER-діаграму (рис. 3.10).

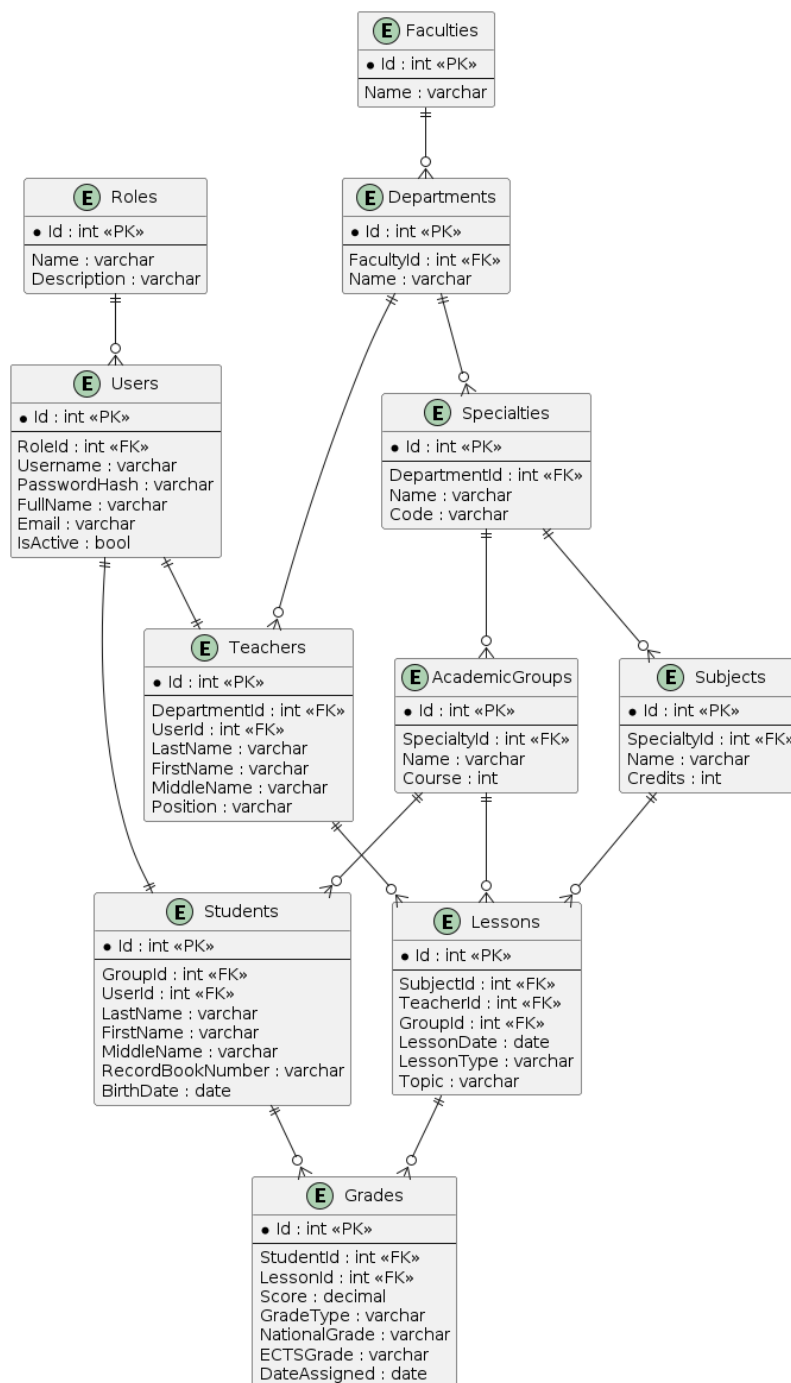


Рисунок 3.10 – ER-діаграма бази даних системи оцінювання студентів

Кожна таблиця відповідає певній сутності предметної області та містить відповідні атрибути.

Таблиця Users призначена для зберігання інформації про користувачів системи. Вона містить такі поля: ідентифікатор користувача, логін, пароль та роль. Роль визначає рівень доступу до функціоналу системи.

Таблиця Students містить інформацію про студентів, включаючи прізвище, ім'я, номер залікової книжки та належність до академічної групи.

Таблиця Teachers зберігає дані про викладачів, зокрема їхні персональні дані та інформацію про кафедру.

Таблиця Groups використовується для зберігання інформації про академічні групи, включаючи їх назву та зв'язок зі спеціальністю.

Таблиця Subjects містить перелік навчальних дисциплін, що викладаються у закладі освіти.

Таблиця Lessons призначена для зберігання інформації про навчальні заняття, зокрема дату проведення, дисципліну, викладача та групу.

Таблиця Grades є ключовою для системи, оскільки містить інформацію про оцінки студентів. Вона включає посилання на студента, заняття, значення оцінки та дату її виставлення.

У базі даних реалізовано зв'язки типу «один до багатьох», які забезпечують логічну цілісність даних:

- один студент може мати багато оцінок;
- одна дисципліна може мати багато занять;
- один викладач може проводити багато занять;
- одна група може містити багато студентів;
- одне заняття може мати багато оцінок.

Зв'язки реалізовані за допомогою зовнішніх ключів, що дозволяє забезпечити узгодженість даних та уникнути появи «осиротілих» записів.

Створення бази даних здійснюється за допомогою SQL-запитів. Нижче наведено приклади створення основних таблиць.

Для забезпечення зберігання облікових записів користувачів у базі даних створено таблицю Users, структура якої наведена у лістингу 3.5.

Лістинг 3.5 – Таблиця користувачів

```
CREATE TABLE Users (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Username VARCHAR(50) NOT NULL,
    PasswordHash VARCHAR(255) NOT NULL,
    Role VARCHAR(20) NOT NULL
);
```

Кінець лістингу 3.5

Для зберігання інформації про студентів, їх персональні дані та належність до академічних груп використовується таблиця Students, структура якої представлена у лістингу 3.6.

Лістинг 3.6 – Таблиця студентів (Students)

```
CREATE TABLE Students (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    RecordBookNumber VARCHAR(20),
    GroupId INT,
    FOREIGN KEY (GroupId) REFERENCES Groups(Id)
);
```

Кінець лістингу 3.6

Для організації обліку навчальних занять у системі реалізовано таблицю Lessons, яка містить інформацію про дисципліни, викладачів, групи та дату проведення занять (ліст. 3.7).

Лістинг 3.7 – Таблиця занять (Lessons)

```
CREATE TABLE Lessons (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    SubjectId INT,
    TeacherId INT,
    GroupId INT,
    Date DATE,
```

```

FOREIGN KEY (SubjectId) REFERENCES Subjects(Id),
FOREIGN KEY (TeacherId) REFERENCES Teachers(Id),
FOREIGN KEY (GroupId) REFERENCES Groups(Id)
);

```

Кінець лістингу 3.7

Зберігання результатів навчання студентів реалізовано за допомогою таблиці Grades, яка містить інформацію про оцінки, прив'язані до конкретних студентів і занять (ліст. 3.8).

Лістинг 3.8 – Таблиця оцінок (Grades)

```

CREATE TABLE Grades (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    StudentId INT,
    LessonId INT,
    Score DECIMAL(5,2),
    DateAssigned DATETIME,
    FOREIGN KEY (StudentId) REFERENCES Students(Id),
    FOREIGN KEY (LessonId) REFERENCES Lessons(Id)
);

```

Кінець лістингу 3.8

Для забезпечення коректності роботи бази даних застосовано такі механізми:

- використання первинних ключів для унікальної ідентифікації записів;
- використання зовнішніх ключів для підтримки зв'язків між таблицями;
- обмеження типів даних для запобігання введенню некоректної інформації;
- перевірка значень оцінок на рівні бізнес-логіки;
- використання транзакцій для гарантування цілісності даних під час виконання операцій.

Таким чином, розроблена база даних забезпечує ефективне зберігання та обробку інформації про студентів, викладачів, дисципліни та результати навчання.

3.3 Захист інформаційно-комп'ютерної системи

Захист інформаційно-комп'ютерної системи є одним із ключових аспектів розробки програмного забезпечення, особливо у випадку обробки персональних даних студентів, результатів їх навчання та службової інформації. Забезпечення безпеки даних включає реалізацію комплексу заходів, спрямованих на збереження конфіденційності, цілісності та доступності інформації [11, 12].

У розробленій системі ведення оцінювання студентів враховано основні загрози інформаційній безпеці, серед яких несанкціонований доступ до системи, спроби зміни або видалення даних, введення некоректної інформації, а також можливі програмні збої. Для протидії цим загрозам реалізовано відповідні механізми захисту, які інтегровані як на рівні програмного коду, так і на рівні бази даних.

Одним із базових механізмів захисту є автентифікація користувачів, яка дозволяє ідентифікувати особу, що намагається отримати доступ до системи. У розробленій системі передбачено введення логіна та пароля, після чого здійснюється перевірка відповідності введених даних записам у базі даних.

Процес автентифікації передбачає пошук користувача за заданими обліковими даними та повернення відповідного об'єкта користувача у разі успішної перевірки. Якщо користувача не знайдено, доступ до системи не надається. Такий підхід дозволяє запобігти несанкціонованому входу до системи.

Реалізація механізму автентифікації користувача наведена у лістингу 3.9.

Лістинг 3.9 – Реалізація автентифікації користувача

```
public User Authenticate(string username, string password)
{
    return _context.Users
        .FirstOrDefault(u => u.Username == username
            && u.PasswordHash == password);
}
```

Кінець лістингу 3.9

Після успішного проходження автентифікації система виконує авторизацію користувача, тобто визначає його права доступу до функціоналу системи. Це дозволяє обмежити можливості користувача відповідно до його ролі та запобігти виконанню несанкціонованих дій. У межах демонстраційної реалізації використано спрощений механізм перевірки пароля, однак архітектура системи передбачає можливість інтеграції сучасних засобів криптографічного захисту.

У системі реалізовано рольову модель доступу, яка передбачає поділ користувачів на адміністраторів, викладачів та студентів. Кожна з ролей має власний набір дозволених операцій. Адміністратор має повний доступ до всіх модулів системи, викладач працює з журналом оцінювання та навчальними даними, а студент має доступ лише до перегляду власної успішності.

Після визначення ролі користувача виконується перенаправлення до відповідного інтерфейсу системи. Це дозволяє не лише підвищити безпеку, але й покращити зручність роботи користувача.

Фрагмент програмного коду, що реалізує розмежування доступу за ролями, наведено у лістингу 3.10.

Лістинг 3.10 – Реалізація розмежування доступу за ролями

```

if (currentUser.Role == "Admin")
{
    OpenAdminPanel();
}
else if (currentUser.Role == "Teacher")
{
    OpenTeacherPanel();
}
else
{
    OpenStudentPanel();
}

```

Кінець лістингу 3.10

Однією з важливих складових захисту інформаційної системи є контроль коректності введених даних. Невалідні або некоректні значення можуть

призвести до порушення логіки роботи системи або до появи помилок у базі даних.

У розробленій системі передбачено перевірку введених значень перед їх збереженням. Зокрема, при введенні оцінок перевіряється, щоб значення знаходилось у допустимому діапазоні. У разі виявлення некоректних даних система генерує відповідне повідомлення про помилку.

Реалізація перевірки коректності оцінки наведена у лістингу 3.11.

Лістинг 3.11 – Перевірка коректності введених даних

```
if (score < 0 || score > 100)
{
    throw new Exception("Оцінка повинна бути в межах 0–100");
}
```

Кінець лістингу 3.11

Захист бази даних реалізовано шляхом використання механізмів, що забезпечують цілісність та узгодженість інформації. Зокрема, застосовуються первинні ключі для унікальної ідентифікації записів та зовнішні ключі для підтримки зв'язків між таблицями.

Крім того, використання ORM-технології Entity Framework Core дозволяє уникнути прямого формування SQL-запитів, що значно знижує ризик SQL-ін'єкцій. Всі запити формуються автоматично та параметризуються, що забезпечує додатковий рівень безпеки.

Цілісність даних є критично важливою для коректного функціонування системи. Для її забезпечення використовуються транзакції, які дозволяють виконувати групу операцій як єдине ціле. У разі виникнення помилки всі зміни відміняються, що запобігає частковому запису даних.

У розробленій системі транзакції застосовуються при виконанні критичних операцій, зокрема при додаванні оцінок.

Фрагмент коду, що демонструє використання транзакцій, наведено у лістингу 3.12.

Лістинг 3.12 – Використання транзакцій для забезпечення цілісності даних

```
using (var transaction = _context.Database.BeginTransaction())
{
    try
    {
        _context.Grades.Add(grade);
        _context.SaveChanges();

        transaction.Commit();
    }
    catch
    {
        transaction.Rollback();
        throw;
    }
}
```

Кінець лістингу 3.12

Для підвищення надійності роботи системи реалізовано механізм обробки виняткових ситуацій. Це дозволяє уникнути аварійного завершення роботи програми та забезпечує інформування користувача про помилки.

У разі виникнення помилки система перехоплює виключення та відображає відповідне повідомлення. Це підвищує зручність використання та дозволяє швидко реагувати на проблемні ситуації.

Приклад обробки помилок наведено у лістингу 3.13.

Лістинг 3.13 – Обробка помилок у системі

```
try
{
    SaveData();
}
catch (Exception ex)
{
    MessageBox.Show("Помилка: " + ex.Message);
}
```

Кінець лістингу 3.13

Таким чином, у розробленій інформаційній системі реалізовано комплексний підхід до забезпечення інформаційної безпеки. Використані

механізми автентифікації, авторизації, валідації даних, контролю цілісності та обробки помилок дозволяють забезпечити надійну роботу системи та захист інформації від несанкціонованого доступу і пошкодження.

3.4 Тестування та налагодження інформаційно-комп'ютерної системи

У межах даної роботи тестування інформаційної системи ведення оцінювання студентів проводилось на різних етапах розробки з метою забезпечення її надійності та стабільності.

Процес тестування розробленої інформаційної системи включав комплексну перевірку її функціональних можливостей, коректності обробки вхідних та вихідних даних, правильності взаємодії з базою даних, а також оцінювання зручності користувацького інтерфейсу. Особливу увагу було приділено перевірці критично важливих функцій системи, до яких належать авторизація користувачів, внесення та редагування оцінок, обчислення підсумкових результатів навчання, пошук і фільтрація інформації, а також експорт даних. Кожна з цих функцій перевірялась у різних сценаріях використання, включаючи як стандартні, так і граничні випадки [13].

У процесі розробки системи було застосовано функціональне тестування, яке дозволило оцінити відповідність реалізованого програмного продукту встановленим функціональним вимогам. Перевірка здійснювалась шляхом виконання тестових сценаріїв, що відображають типові дії користувачів системи. Для кожного сценарію визначались вхідні дані, очікуваний результат та фактичний результат виконання. Це дозволило виявити можливі відхилення у роботі системи та своєчасно їх усунути.

Окрім функціонального тестування, проводилось модульне тестування окремих компонентів системи. Зокрема, перевірялись модулі роботи з базою даних, логіка обчислення оцінок, механізми авторизації та обробки введених даних. Модульне тестування дозволило ізолювати окремі частини системи та

перевірити їх коректність незалежно від інших компонентів. Такий підхід значно спростив процес пошуку помилок та підвищив якість програмного коду.

Важливим етапом перевірки стало інтеграційне тестування, яке було спрямоване на оцінювання взаємодії між різними модулями системи. Зокрема, перевірялась коректність передачі даних між інтерфейсом користувача та базою даних, правильність виконання операцій додавання, редагування та видалення записів, а також узгодженість даних після виконання комплексних операцій. Інтеграційне тестування дозволило виявити проблеми, пов'язані з взаємодією компонентів, які не можуть бути знайдені під час модульного тестування.

Окрему увагу було приділено тестуванню користувацького інтерфейсу. Перевірялась зручність навігації, логічність розташування елементів управління, зрозумілість повідомлень системи та швидкість виконання операцій. Тестування інтерфейсу дозволило оцінити, наскільки система є інтуїтивно зрозумілою для користувача, а також виявити можливі недоліки у взаємодії користувача з програмним продуктом.

Додатково перевірялась стійкість системи до помилкових дій користувача, зокрема введення некоректних даних, залишення обов'язкових полів незаповненими або спроб виконання заборонених операцій. У таких випадках система повинна коректно обробляти помилки, не допускаючи аварійного завершення роботи та надаючи користувачу зрозумілі повідомлення про причину помилки.

Таким чином, застосування комплексного підходу до тестування дозволило всебічно перевірити функціонування інформаційної системи, забезпечити її стабільність, надійність та відповідність встановленим вимогам, а також підвищити якість кінцевого програмного продукту.

Для перевірки роботи системи було розроблено набір тестових сценаріїв, які охоплюють основні функції програмного продукту.

Результати перевірки основних функціональних можливостей системи наведено у таблиці 3.1.

Таблиця 3.1 – Результати тестування системи

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат
1	Авторизація користувача	Логін і пароль	Успішний вхід	Успішно
2	Невірний пароль	Невірний пароль	Повідомлення про помилку	Відображено
3	Додавання студента	Дані студента	Запис у БД	Успішно
4	Виставлення оцінки	Оцінка 85	Збереження оцінки	Успішно
5	Некоректна оцінка	Оцінка 150	Помилка	Виявлено
6	Пошук студента	Прізвище	Відображення результатів	Успішно
7	Експорт даних	Список студентів	CSV файл	Створено

У процесі тестування було виявлено ряд помилок, пов'язаних із введенням некоректних даних, обробкою винятків та взаємодією з базою даних. Для їх усунення застосовувались методи налагодження, які включали аналіз коду, використання засобів відлагодження середовища Visual Studio та покрокове виконання програми.

Особливу увагу було приділено обробці помилок, що дозволило запобігти аварійному завершенню роботи програми. Для цього було реалізовано механізм обробки виключень.

Фрагмент коду, що демонструє обробку помилок, наведено у лістингу 3.14.

Лістинг 3.14 – Обробка помилок при виконанні операцій

```
try
{
    SaveChanges();
}
catch (Exception ex)
{
    MessageBox.Show("Помилка: " + ex.Message);
}
```

Кінець лістингу 3.14

Результати проведеного тестування показали, що система коректно виконує всі основні функції, передбачені вимогами. Усі критичні помилки були виявлені та усунені на етапі налагодження. Система стабільно працює при різних сценаріях використання та забезпечує коректну обробку даних.

Таким чином, проведене тестування підтвердило працездатність та надійність розробленої інформаційної системи. Використання різних видів тестування дозволило виявити та усунути помилки, а застосування методів налагодження забезпечило стабільну роботу програмного продукту.

ВИСНОВКИ

У кваліфікаційній роботі було виконано поставлене завдання щодо аналізу сучасного стану проблеми автоматизації оцінювання студентів. У ході дослідження розглянуто існуючі програмні рішення, визначено їхні переваги та недоліки, що дозволило обґрунтувати доцільність розробки власної інформаційної системи.

У процесі виконання роботи було сформульовано вимоги до програмного забезпечення та здійснено проектування системи. Зокрема, визначено функціональні можливості, побудовано UML-моделі, включаючи діаграми варіантів використання, діяльності та послідовності, що дозволило формалізувати структуру та логіку роботи системи.

Було обґрунтовано вибір засобів, методів і технологій розробки програмного забезпечення. Для реалізації системи обрано мову програмування C#, платформу .NET, технологію Windows Forms та реляційну базу даних MySQL, що забезпечило ефективну реалізацію функціоналу та зручність використання програмного продукту.

У межах роботи здійснено практичну реалізацію інформаційної системи ведення оцінювання студентів. Реалізовано основні функціональні модулі, зокрема авторизацію користувачів, управління даними студентів, ведення журналу оцінювання, формування статистики та експорт результатів, що забезпечує автоматизацію навчального процесу.

Було розроблено структуру бази даних на основі реляційної моделі, визначено основні сутності та зв'язки між ними, а також реалізовано SQL-запити для створення таблиць. Це забезпечило цілісність, узгодженість та ефективне зберігання даних у системі.

У процесі роботи було реалізовано механізми захисту інформаційної системи, включаючи автентифікацію користувачів, розмежування прав доступу, валідацію даних, обробку помилок та забезпечення цілісності даних, що дозволило підвищити рівень безпеки системи.

Також було проведено тестування та налагодження програмного забезпечення, що дозволило перевірити коректність роботи системи, виявити та усунути помилки, а також підтвердити відповідність реалізованого функціоналу поставленим вимогам.

Таким чином, у результаті виконання кваліфікаційної роботи всі поставлені завдання було повністю виконано, а розроблена інформаційна система є працездатною, надійною та може бути використана для автоматизації процесу оцінювання студентів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ministry of Education and Science of Ukraine. Цифрова трансформація освіти в Україні. URL: (дата звернення: 14.04.2026).
2. World Bank. Digital Technologies in Education 2023. URL: <https://www.worldbank.org/en/topic/edutech> (дата звернення: 14.04.2026).
3. Адаменко В. О. Веб-сервіс для організації та управління освітнім процесом університету. 2025. URL: <https://eportfolio.kubg.edu.ua/data/conference/11944/document.pdf> (дата звернення: 14.04.2026).
4. Object Management Group. Unified Modeling Language (UML) Specification. URL: <https://www.omg.org/spec/UML/> (дата звернення: 14.04.2026).
5. ISO. ISO/IEC/IEEE 12207:2017 + Amendments. URL: <https://www.iso.org/standard/63712.html> (дата звернення: 24.04.2026).
6. Microsoft. .NET documentation. URL: <https://learn.microsoft.com/dotnet> (дата звернення: 24.04.2026).
7. Microsoft. C# documentation. URL: <https://learn.microsoft.com/dotnet/csharp/> (дата звернення: 24.04.2026).
8. Microsoft. Windows Forms documentation. URL: <https://learn.microsoft.com/dotnet/desktop/winforms/> (дата звернення: 24.04.2026).
9. Microsoft. Entity Framework Core documentation. URL: <https://learn.microsoft.com/ef/core/> (дата звернення: 24.04.2026).
10. Oracle. MySQL Documentation. URL: <https://dev.mysql.com/doc/> (дата звернення: 24.04.2026).
11. OWASP Foundation. OWASP Top 10: The Ten Most Critical Web Application Security Risks, 2021. URL: <https://owasp.org/www-project-top-ten/> (дата звернення: 20.05.2026).
12. National Institute of Standards and Technology. Secure Software Development Framework (SSDF) Version 1.1, 2022. URL:

<https://csrc.nist.gov/publications/detail/sp/800-218/final> (дата звернення: 20.05.2026).