

**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 ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ СТУДЕНТІВ
ВИКОРИСТОВУЮЧИ .NET TA MYSQL**

**DEVELOPMENT OF A STUDENT ACCOUNTING INFORMATION
SYSTEM USING .NET AND MYSQL**

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

Виконав: здобувач вищої освіти
групи ІПЗ-42
Котус Владислав Миколайович

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

Котусу Владиславу Миколайовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка інформаційної системи обліку студентів використовуючи .NET та MySQL»

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: мова програмування C#; платформа .NET; технологія інтерфейсу Windows Forms; система керування базами даних MySQL; інструментальні засоби Visual Studio.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): аналіз сучасного стану проблеми автоматизації обліку студентів, постановка завдання на розробку інформаційної системи, формування вимог до програмного забезпечення обліку студентів, обґрунтування вибору технологій та засобів розробки (платформи .NET та СУБД MySQL), проєктування структури інформаційної системи та бази даних, практична реалізація програмного застосунку для обліку студентів, дисциплін і користувачів, обґрунтування принципів захисту інформації та розмежування прав доступу, тестування та налагодження розробленого програмного забезпечення.

5. Перелік графічного матеріалу: 14 рисунків, 3 таблиці, 5 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 _____

Владислав КОТУС

Керівник кваліфікаційної роботи _____

Юрій ГУЛЬЧУК

АНОТАЦІЯ

Котус В. М. Розробка інформаційної системи обліку студентів використовуючи .NET та MySQL. 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 обліку студентів, розглянуто існуючі аналоги програмних рішень, визначено їх переваги та недоліки, а також обґрунтовано доцільність створення нової системи. В другому розділі визначено вимоги до розроблюваного програмного забезпечення, виконано проектування системи із застосуванням UML-моделей, а також обґрунтовано вибір засобів, методів і технологій реалізації програмного продукту. У третьому розділі представлено практичну реалізацію інформаційної системи, розробку бази даних, опис механізмів захисту даних, а також результати тестування та налагодження програмного забезпечення. У висновках узагальнено результати виконаної роботи, підтверджено досягнення поставленої мети та виконання всіх визначених завдань.

Ключові слова: інформаційна система, облік студентів, .NET, MySQL, Windows Forms, база даних, автоматизація, програмне забезпечення.

ABSTRACT

Kotus V. M. Development of a student accounting information system using .NET and MySQL. 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.

In the first chapter, an analysis of the current state of the problem of student records automation was carried out, existing analogs of software solutions were reviewed, their advantages and disadvantages were identified, and the feasibility of developing a new system was substantiated. In the second chapter, the requirements for the software being developed were defined, the system was designed using UML models, and the choice of tools, methods, and technologies for implementing the software product was justified. In the third chapter, the practical implementation of the information system, the database development, the description of data protection mechanisms, as well as the results of software testing and debugging were presented. The conclusions summarize the results of the completed work and confirm that the stated goal has been achieved and all defined tasks have been fulfilled.

Keywords: information system, student records, .NET, MySQL, Windows Forms, database, automation, software.

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ ОБЛІКУ ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ.....	10
1.1 Аналіз сучасного стану проблеми	10
1.2 Постановка завдання на кваліфікаційну роботу бакалавра	16
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ...	17
2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення	17
2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання ...	30
РОЗДІЛ 3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ СТУДЕНТІВ.....	35
3.1 Практична реалізація об'єкта проєктування.....	35
3.2 Розробка бази даних	42
3.3 Захист інформаційно-комп'ютерної системи	47
3.4 Тестування та налагодження інформаційно-комп'ютерної системи	51
ВИСНОВКИ	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	58

ВСТУП

У сучасних умовах цифрової трансформації освіти особливого значення набуває автоматизація процесів збирання, зберігання, обробки та аналізу інформації про здобувачів освіти. Однією з найважливіших складових управління освітнім процесом у закладах вищої освіти є облік студентів, який охоплює ведення персональних даних, відомостей про академічні групи, факультети, спеціальності, результати навчання, перелік дисциплін, показники успішності та іншу супровідну інформацію. Якість організації такого обліку безпосередньо впливає на оперативність управлінських рішень, ефективність роботи адміністративного персоналу та достовірність інформаційного забезпечення освітнього процесу. Областю застосування розроблюваної програми є інформаційна підтримка діяльності закладів вищої освіти, зокрема процесів, пов'язаних із веденням обліку контингенту студентів. Об'єктом, у якому використовується програмний продукт, є освітнє середовище факультету, кафедри, деканату або іншого структурного підрозділу закладу освіти, де виникає потреба в централізованому збереженні та впорядкуванні відомостей про студентів. Розроблена система може бути використана для перегляду, додавання, редагування, фільтрації та аналізу даних про студентів, а також для організації доступу різних категорій користувачів до відповідної інформації.

Сучасний стан розв'язання проблеми обліку студентів характеризується широким використанням як комплексних університетських інформаційних систем, так і локальних програмних засобів, електронних таблиць, окремих баз даних та файлових рішень. На практиці значна частина завдань, пов'язаних з веденням студентського обліку, уже автоматизована: здійснюється зберігання базових персональних даних, формування списків академічних груп, реєстрація навчальних дисциплін, облік успішності та підготовка окремих видів звітності. Водночас у цій предметній галузі залишаються і суттєві прогалини. Зокрема, у багатьох випадках дані є розподіленими між різними джерелами, відсутня єдина логічна структура їх зберігання, недостатньо реалізоване розмежування доступу

між категоріями користувачів, а також обмежені можливості пошуку, фільтрації, сортування й локального адміністрування інформації. Крім того, частина існуючих рішень є надто складною для невеликих освітніх підрозділів або потребує значних ресурсів для впровадження й супроводу.

Світові тенденції вирішення таких задач пов'язані з переходом до цифрових платформ управління освітнім процесом, активним використанням баз даних, модульних архітектур, багаторівневого розмежування прав доступу, засобів аналітичної обробки інформації та інтеграції різних інформаційних сервісів в єдине програмне середовище. У сучасних програмних продуктах дедалі більшого значення набувають зручність інтерфейсу, надійність зберігання даних, адаптивність системи до змін у структурі предметної області та можливість її подальшого розширення. Для локальних і спеціалізованих завдань, пов'язаних з обліком студентів, актуальними залишаються компактні інформаційні системи, орієнтовані на практичність, простоту використання та швидке впровадження.

Актуальність теми кваліфікаційної роботи зумовлена необхідністю створення зручного та функціонального програмного засобу для централізованого ведення обліку студентів із можливістю збереження структурованих даних, швидкого пошуку інформації, підтримки різних ролей користувачів та використання надійного сховища даних. Використання паперових носіїв, окремих електронних таблиць або неузгоджених файлів не забезпечує належного рівня ефективності, особливо за умови зростання кількості студентів та ускладнення процесів адміністрування освітньої діяльності. Саме тому розробка інформаційної системи обліку студентів із використанням сучасних засобів програмування є практично важливим і доцільним завданням.

Метою кваліфікаційної роботи є розробка інформаційної системи обліку студентів з використанням платформи .NET та системи керування базами даних MySQL, яка забезпечує централізоване зберігання, перегляд, редагування, пошук і аналіз даних про студентів, їх навчальні дисципліни та користувачів системи.

Об'єктом дослідження є процес обліку студентів у закладі вищої освіти.

Предметом дослідження є методи, засоби та програмні технології розробки інформаційної системи обліку студентів на основі .NET та MySQL.

Для досягнення поставленої мети в роботі необхідно розв'язати такі завдання:

- проаналізувати сучасний стан проблеми автоматизації обліку студентів та існуючі підходи до її розв'язання;
- розглянути аналоги інформаційних систем обліку студентів, визначити їх переваги та недоліки;
- сформулювати вимоги до розроблюваної інформаційної системи;
- обґрунтувати вибір технологій, засобів і методів реалізації програмного продукту;
- спроектувати структуру інформаційної системи та базу даних для зберігання відомостей про студентів;
- реалізувати програмний застосунок для ведення обліку студентів, дисциплін і користувачів системи;
- передбачити механізми розмежування доступу та основні засоби захисту даних;
- виконати тестування та налагодження розробленого програмного забезпечення.

У межах роботи розроблено настільний застосунок для операційної системи Windows, який підтримує авторизацію користувачів, рольову модель доступу, ведення переліку студентів, зберігання персональних і навчальних відомостей, роботу з дисциплінами, а також фільтрацію, сортування та адміністрування даних.

Практичне значення одержаних результатів полягає у створенні працездатної інформаційної системи, яка може бути використана для автоматизації процесів обліку студентів у межах факультету, кафедри або іншого структурного підрозділу закладу вищої освіти. Розроблене програмне забезпечення дозволяє підвищити оперативність роботи з даними, зменшити кількість рутинних операцій, покращити структурованість інформації.

РОЗДІЛ 1

АНАЛІЗ МЕТОДІВ РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ ОБЛІКУ ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасних закладах вищої освіти обсяг інформації, пов'язаної з обліком студентів, постійно зростає. Необхідність зберігання, оновлення та аналізу відомостей про контингент здобувачів освіти, їх навчальні досягнення, академічні групи, спеціальності, дисципліни та інші пов'язані дані зумовлює потребу у використанні спеціалізованих інформаційних систем. Традиційні підходи до ведення такого обліку, що базуються на паперовій документації, електронних таблицях або розрізних локальних файлах, вже не забезпечують належного рівня оперативності, цілісності та захищеності даних.

Облік студентів є однією з ключових складових управління освітнім процесом. Від якості організації цього процесу залежить швидкість доступу до актуальної інформації про здобувачів освіти, можливість контролю змін у складі академічних груп, відстеження результатів навчання, обліку успішності, а також формування звітних та довідкових матеріалів для адміністрації закладу освіти. У випадку відсутності єдиної централізованої системи працівники деканату, кафедри або куратори змушені виконувати велику кількість рутинних операцій вручну, що підвищує ризик помилок, дублювання інформації та втрати важливих відомостей.

На практиці в багатьох випадках дані про студентів зберігаються у вигляді набору окремих документів або таблиць, які не завжди синхронізовані між собою. Наприклад, персональні дані можуть зберігатися в одному файлі, інформація про навчальні дисципліни – в іншому, а дані про успішність чи відвідуваність – у ще одному джерелі. Такий підхід ускладнює пошук потрібної інформації, створює проблеми з актуальністю даних та унеможливорює швидке отримання узагальненої картини щодо конкретного студента або групи студентів.

Ще однією актуальною проблемою є розмежування доступу до інформації. У межах освітньої установи різні категорії працівників повинні мати різний рівень доступу до даних. Системний адміністратор повинен мати можливість повного керування системою, декан – перегляду та аналізу даних лише в межах свого факультету, завідувач кафедри – доступу до студентів відповідних спеціальностей, а куратор – роботи лише з призначеними академічними групами. За відсутності програмної реалізації ролей доступу виникає ризик несанкціонованого перегляду, зміни або видалення інформації.

Окрему увагу слід приділити проблемі зручності роботи з даними. У сучасних умовах користувачі очікують від програмного забезпечення не лише можливості зберігати інформацію, але й інструментів для її швидкого пошуку, фільтрації, сортування та редагування. Якщо система не підтримує таких функцій, робота з великим масивом студентських записів стає малоефективною. Особливо це важливо для адміністрування контингенту студентів, коли потрібно оперативно знайти здобувачів за факультетом, спеціальністю, групою, курсом, формою навчання чи іншими параметрами.

Актуальною є також проблема збереження структурованих даних про навчання студента. Для повноцінного обліку недостатньо фіксувати лише базові персональні відомості. Необхідно також зберігати навчальні характеристики: номер студентського квитка, рік вступу, рік завершення навчання, форму навчання, середній бал, перелік дисциплін, статус їх вивчення, кількість кредитів, інформацію про викладача, а також відомості про пропущені та загальну кількість занять. Ведення цих даних у неструктурованому або частково структурованому вигляді значно ускладнює подальший аналіз і контроль освітнього процесу.

Сучасний стан розвитку програмних технологій дозволяє ефективно вирішувати зазначені проблеми шляхом створення спеціалізованих інформаційних систем. Для цього доцільно використовувати архітектурні підходи, що забезпечують відокремлення інтерфейсу користувача, бізнес-логіки та рівня доступу до даних. Такий підхід підвищує зручність супроводу

програмного продукту, спрощує його тестування та створює передумови для подальшого масштабування. У розроблюваній системі це реалізується через поділ на модулі інтерфейсу, сервісів, моделей та репозиторіїв, що відповідає сучасним принципам побудови програмного забезпечення.

Для зберігання даних про студентів доцільно використовувати реляційну базу даних, оскільки вона забезпечує впорядковане подання інформації, підтримку зв'язків між сутностями, цілісність даних і можливість виконання запитів для вибірки, фільтрації та оновлення записів. Саме тому застосування MySQL як системи керування базами даних є обґрунтованим рішенням для реалізації постійного сховища інформації.

На сьогодні значна кількість програмних продуктів для обліку студентів або є надто складними для локального впровадження, або орієнтовані на великі комплексні інформаційні системи закладу освіти, що потребують значних ресурсів для налаштування та супроводу. Натомість для окремих підрозділів або невеликих навчальних структур актуальною залишається потреба у компактному, зрозумілому та функціональному програмному продукті, який забезпечує базові та розширені операції обліку без надмірної складності використання.

У межах даної кваліфікаційної роботи розглядається розробка інформаційної системи обліку студентів, яка реалізована у вигляді настільного застосунку для операційної системи Windows. Система забезпечує авторизацію користувачів, ведення обліку студентів, роботу з переліком дисциплін, перегляд та редагування профілю студента, фільтрацію й сортування записів, а також адміністрування користувачів. Інтерфейс системи передбачає наявність форми входу, головного вікна з таблицею студентів, вікна редагування профілю студента з вкладками основних даних, навчання та дисциплін, а також окремого модуля керування користувачами. Виходячи з реалізованого функціоналу, система орієнтована на практичне використання в межах освітнього підрозділу для централізованого ведення студентського обліку.

У науковій та навчально-методичній літературі питання автоматизації обліку, зберігання та опрацювання інформації розглядаються в межах досліджень, присвячених інформаційним системам, базам даних, програмній інженерії та автоматизації управлінських процесів. У таких джерелах основна увага приділяється принципам побудови інформаційних систем, методам структурованого зберігання даних, організації взаємодії між користувачем і програмою, а також підходам до забезпечення цілісності, актуальності та безпеки інформації. Аналіз літературних джерел свідчить, що ефективна інформаційна система обліку повинна забезпечувати централізоване збереження даних, підтримувати логічну структуру сутностей предметної області, надавати засоби швидкого пошуку й фільтрації записів та підтримувати контрольований доступ користувачів до різних категорій інформації [1].

У спеціалізованій літературі з програмної інженерії також наголошується на важливості використання багатoshарової архітектури при розробці прикладних інформаційних систем. Поділ системи на рівень представлення, рівень бізнес-логіки та рівень доступу до даних дозволяє спростити розробку, підвищити повторне використання коду та полегшити подальшу модернізацію програмного продукту. Для системи обліку студентів такий підхід є особливо доцільним, оскільки в майбутньому може виникнути потреба у розширенні складу користувачів, додаванні нових модулів або переході від локального до мережевого режиму роботи [2].

Під час аналізу сучасного стану проблеми доцільно розглянути і вже існуючі аналоги інформаційних систем, що використовуються для обліку студентів або частково реалізують подібний функціонал. До таких аналогів можна віднести великі університетські інформаційні системи, електронні деканати, системи управління навчальним процесом, а також окремі програмні продукти локального рівня, побудовані на базі електронних таблиць, локальних баз даних або вебзастосунків. Вони дозволяють автоматизувати певні аспекти роботи з даними студентів, однак мають різний рівень повноти функціональних можливостей [3].

Перевагою існуючих університетських систем є комплексний характер, оскільки вони часто охоплюють не лише облік студентів, а й розклад занять, навчальні плани, електронні журнали, звітність, кабінети викладачів і здобувачів освіти. Такі системи зазвичай забезпечують централізоване зберігання даних, роботу в мережевому середовищі та багатокористувацький доступ. Разом з тим їх недоліками є висока складність впровадження, значні вимоги до технічної інфраструктури, тривалий період налаштування та надмірність функціоналу для окремих локальних завдань, пов'язаних саме з обліком студентів у межах конкретного факультету, спеціальності або групи.

Іншою групою аналогів є системи, реалізовані у вигляді електронних таблиць або невеликих локальних баз даних. Їхньою перевагою є простота створення та використання, а також відсутність необхідності у складному технічному супроводі. Проте такі рішення мають суттєві недоліки: слабкий рівень захисту даних, відсутність повноцінного розмежування доступу, складність підтримання цілісності інформації, незручність роботи з великими масивами записів, обмежені можливості пошуку та фільтрації, а також низьку масштабованість. Крім того, в таких системах часто відсутня чітка модель предметної області, через що одна й та сама інформація може дублюватися в різних файлах або таблицях.

Також існують веборієнтовані рішення для обліку студентів та освітнього адміністрування. Їхньою перевагою є доступність через браузер, можливість централізованого оновлення та зручність у разі територіально розподіленої роботи користувачів. Однак для локального застосування настільні системи часто виявляються практичнішими, оскільки вони простіші у розгортанні, не потребують складного серверного середовища, можуть працювати автономно та забезпечують вищу передбачуваність конфігурації робочого місця користувача. Саме тому для кваліфікаційної роботи доцільним є вибір настільного застосунку як форми реалізації проєкту.

Аналіз аналогів дозволяє виокремити основні вимоги до розроблюваної системи. Вона повинна забезпечувати централізоване зберігання відомостей про

студентів, їх академічні характеристики та дисципліни; підтримувати зручний інтерфейс для перегляду, додавання, редагування й видалення записів; реалізовувати рольову модель доступу для різних категорій користувачів; забезпечувати фільтрацію, сортування та пошук інформації; підтримувати постійне зберігання в базі даних.

Окремим елементом аналізу сучасного стану проблеми є патентний пошук. Для тематики інформаційних систем обліку студентів патентний пошук дозволяє встановити, чи існують зареєстровані технічні рішення, що повністю або частково дублюють функціональність розроблюваного програмного продукту. У межах такого пошуку зазвичай аналізуються винаходи, корисні моделі, програмно-технічні комплекси та інформаційні технології, пов'язані з автоматизацією навчального процесу, обліком контингенту студентів, електронними журналами, системами моніторингу успішності та адміністрування освітніх даних.

За результатами узагальненого аналізу можна зробити висновок, що сама ідея автоматизації обліку студентів не є новою, оскільки належить до широкого класу інформаційних систем управління даними. Водночас конкретна реалізація системи, її структура, набір функцій, модель розмежування доступу, а також архітектурні рішення можуть становити самостійний програмний продукт, розроблений під конкретні практичні потреби. У такому випадку метою кваліфікаційної роботи є не створення принципово нового технічного рішення, а розробка працездатної, логічно спроектованої та практично корисної інформаційної системи.

Таким чином, аналіз сучасного стану проблеми, огляд літературних джерел, розгляд існуючих аналогів та узагальнені результати патентного пошуку підтверджують актуальність теми кваліфікаційної роботи. Розробка інформаційної системи обліку студентів із використанням .NET та MySQL є доцільною, оскільки дозволяє створити програмний продукт, що поєднує зручність настільного застосунку, структуроване зберігання даних, гнучке розмежування доступу та можливість подальшого функціонального розвитку.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи бакалавра є розробка інформаційної системи обліку студентів з використанням платформи .NET та системи керування базами даних MySQL, яка забезпечує централізоване зберігання, перегляд, редагування, пошук і аналіз даних про студентів, їх навчальні дисципліни та користувачів системи.

Для досягнення поставленої мети в кваліфікаційній роботі необхідно розв'язати такі завдання:

- проаналізувати сучасний стан проблеми автоматизації обліку студентів та існуючі підходи до її розв'язання;
- розглянути аналоги інформаційних систем обліку студентів, визначити їх переваги та недоліки;
- сформулювати вимоги до розроблюваної інформаційної системи;
- обґрунтувати вибір технологій, засобів і методів реалізації програмного продукту;
- спроектувати структуру інформаційної системи та базу даних для зберігання відомостей про студентів;
- реалізувати програмний застосунок для ведення обліку студентів, дисциплін і користувачів системи;
- передбачити механізми розмежування доступу та основні засоби захисту даних;
- виконати тестування та налагодження розробленого програмного забезпечення.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

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

Розробка інформаційної системи обліку студентів потребує попереднього аналізу вимог до програмного забезпечення, визначення цілей і задач системи, вибору підходів до моделювання, а також формування узгодженої специфікації функціональних і нефункціональних характеристик. Саме етап аналізу та проєктування є визначальним для забезпечення якості майбутнього програмного продукту, оскільки на ньому закладається структура системи, логіка її функціонування, склад основних модулів, ролі користувачів та правила взаємодії між ними.

Основною ціллю розроблюваної інформаційної системи є автоматизація процесів обліку студентів у закладі вищої освіти, а саме централізоване зберігання, обробка, редагування, пошук та аналіз інформації про студентів, їхні навчальні характеристики, дисципліни та користувачів системи.

Для досягнення поставленої цілі система повинна забезпечувати виконання таких задач: ведення єдиної бази даних студентів; збереження персональних, навчальних і контактних даних студентів; олік дисциплін, що вивчаються студентами; фіксацію результатів навчання та характеристик відвідування; швидкий пошук, фільтрацію та сортування записів; рольове розмежування доступу до інформації; адміністрування користувачів системи; забезпечення збереження, цілісності та актуальності даних.

Для моделювання програмної системи доцільно використати об'єктно-орієнтований підхід та уніфіковану мову моделювання UML, оскільки розроблювана інформаційна система має чітко виражену структуру сутностей, між якими існують логічні зв'язки. До таких сутностей належать студент, дисципліна, користувач, група, спеціальність, факультет, роль користувача та інші елементи предметної області [4, 5].

Виявлення вимог до інформаційної системи здійснюється на основі аналізу предметної області, розгляду функцій, які повинні бути реалізовані в програмі, а також дослідження потреб потенційних користувачів системи. У межах даної роботи використано такі методи аналізу вимог: аналіз предметної області обліку студентів; аналіз існуючих аналогів; функціональна декомпозиція майбутньої системи; визначення ролей користувачів і пов'язаних із ними прав доступу; опис типових сценаріїв використання системи.

У ролі основних користувачів системи визначено: системного адміністратора; декана; завідувача кафедри; куратора академічної групи.

Кожна з цих ролей має власні інформаційні потреби. Системний адміністратор потребує повного доступу до системи, включаючи управління користувачами. Декан працює з даними студентів певного факультету. Завідувач кафедри повинен мати доступ до студентів визначених спеціальностей. Куратор працює лише з окремими академічними групами. Такий підхід дозволяє реалізувати принцип мінімально необхідних прав доступу.

Вимоги до програмного забезпечення доцільно поділити на функціональні та нефункціональні.

Розроблювана інформаційна система обліку студентів повинна забезпечувати повний набір функцій, необхідних для ефективного ведення, зберігання та обробки даних у межах закладу вищої освіти. Однією з базових функцій системи є авторизація користувачів, яка передбачає введення логіна і пароля, перевірку правильності облікових даних, визначення ролі користувача та надання доступу лише до тих функцій, що дозволені відповідно до встановлених прав. У межах демонстраційної реалізації використано спрощений механізм перевірки пароля, однак архітектура системи передбачає можливість інтеграції сучасних засобів криптографічного захисту.

Система повинна підтримувати повноцінну роботу з обліком студентів, зокрема додавання нових записів, редагування вже існуючих даних, видалення записів, а також перегляд повного профілю студента. При цьому має бути забезпечене збереження основних персональних відомостей, контактної

інформації, фотографії студента та додаткових приміток, що можуть бути корисними у процесі адміністрування.

Важливою складовою функціональності є робота з навчальними даними студента. Система повинна надавати можливість фіксувати спеціальність, академічну групу, номер студентського квитка, рік вступу та рік завершення навчання, курс, форму навчання, середній бал, а також статус активності студента.

Окремий блок функцій пов'язаний з роботою з дисциплінами. Для кожного студента система повинна дозволяти додавати дисципліни до профілю, редагувати або видаляти їх, а також зберігати інформацію про статус вивчення дисципліни, отриманий бал, кількість кредитів, число пропущених занять, загальну кількість занять і відомості про викладача.

Не менш важливою функціональною вимогою є забезпечення зручного пошуку, фільтрації та сортування даних. Користувач повинен мати можливість здійснювати загальний пошук за ключовими словами, виконувати відбір записів за групою, факультетом, спеціальністю, курсом, формою навчання, активністю студента та характеристиками дисциплін, а також упорядковувати результати за визначеними критеріями.

Система також повинна підтримувати засоби управління користувачами. Зокрема, має бути реалізовано перегляд списку користувачів, створення нових облікових записів, редагування їх параметрів, блокування та активацію користувачів, а також призначення ролей доступу.

З огляду на особливості реалізації, система повинна забезпечувати роботу з різними джерелами даних. У робочому режимі інформація має зберігатися у базі даних MySQL.

Окремою функціональною вимогою є контроль коректності введених даних. Система повинна перевіряти заповнення обов'язкових полів, контролювати правильність числових і текстових значень, забезпечувати логічну узгодженість введеної інформації та запобігати некоректному або неповному введенню даних. Реалізація таких перевірок є необхідною умовою для

підтримання цілісності інформації та зменшення кількості помилок під час роботи користувачів.

Окрім функціональних характеристик, важливе значення для якості програмного продукту мають нефункціональні вимоги, які визначають загальні властивості системи та умови її ефективного використання. Насамперед система повинна бути надійною, тобто забезпечувати коректне збереження даних, запобігати втраті інформації та стабільно функціонувати під час виконання стандартних сценаріїв роботи.

Важливою характеристикою є продуктивність системи. Програмний продукт повинен забезпечувати швидке завантаження списків студентів, оперативне виконання пошуку та фільтрації, а також прийнятний час відкриття форм редагування та інших елементів інтерфейсу.

Однією з ключових нефункціональних вимог є безпека. Система повинна підтримувати базову автентифікацію користувачів, обмеження доступу відповідно до визначених ролей та запобігання несанкціонованому перегляду інформації.

Не менш важливою є зручність використання програмного продукту. Графічний інтерфейс повинен бути зрозумілим для користувача, елементи керування мають бути логічно згруповані, а доступ до основних функцій – швидким і інтуїтивно зрозумілим. Дані повинні відображатися у зручній формі, зокрема у вигляді таблиць, форм редагування та діалогових вікон.

Крім того, система повинна бути придатною до подальшого розвитку та супроводу. Це означає, що архітектура програмного забезпечення має допускати додавання нових модулів, модифікацію структури системи та чіткий розподіл компонентів за архітектурними рівнями. Такий підхід забезпечує кращу підтримуваність програмного коду, спрощує внесення змін і створює умови для масштабування функціональних можливостей у майбутньому. Для систематизації результатів аналізу вимог подамо їх у вигляді таблиці 2.1.

Таблиця 2.1 – Функціональні та нефункціональні вимоги до інформаційної системи обліку студентів

Категорія вимог	Назва вимоги	Зміст вимоги
Функціональна	Авторизація користувачів	Система повинна забезпечувати вхід користувача за логіном і паролем з перевіркою облікових даних
Функціональна	Рольове розмежування доступу	Система повинна обмежувати доступ до окремих функцій відповідно до ролі користувача
Функціональна	Ведення обліку студентів	Система повинна забезпечувати додавання, редагування, перегляд і видалення записів про студентів
Функціональна	Збереження персональних даних	Система повинна зберігати ПІБ, дату народження, студентський квиток, контакти, фото та інші основні відомості
Функціональна	Збереження навчальних даних	Система повинна зберігати факультет, спеціальність, групу, курс, форму навчання, роки вступу та завершення навчання
Функціональна	Робота з дисциплінами	Система повинна дозволяти додавати, редагувати й видаляти дисципліни студента
Функціональна	Облік результатів навчання	Система повинна зберігати бал, кредити, статус дисципліни, кількість пропущених і загальних занять
Функціональна	Пошук інформації	Система повинна забезпечувати текстовий пошук студентів за ключовими полями
Функціональна	Фільтрація та сортування	Система повинна підтримувати відбір записів за факультетом, спеціальністю, групою, курсом, формою навчання та іншими ознаками
Функціональна	Управління користувачами	Система повинна забезпечувати створення, редагування, активацію та блокування користувачів
Функціональна	Підтримка режимів роботи	Система повинна працювати в робочому режимі з MySQL
Нефункціональна	Надійність	Система повинна забезпечувати стабільну роботу та коректне збереження даних
Нефункціональна	Продуктивність	Система повинна швидко виконувати завантаження записів, пошук і фільтрацію
Нефункціональна	Зручність використання	Інтерфейс системи повинен бути зрозумілим, логічним і зручним для користувача
Нефункціональна	Безпека	Система повинна запобігати несанкціонованому доступу до даних
Нефункціональна	Цілісність даних	Система повинна забезпечувати узгодженість і несуперечливість інформації в базі даних
Нефункціональна	Масштабованість	Архітектура системи повинна дозволяти додавання нових модулів і функцій
Нефункціональна	Супровідність	Структура програмного проекту повинна бути зрозумілою та придатною до подальшого розвитку
Нефункціональна	Портативність у межах платформи	Система повинна коректно працювати на комп'ютерах під керуванням Windows із підтримкою .NET

Для уточнення вимог доцільно подати основні сценарії використання системи:

- авторизація користувача. Користувач запускає програму, вводить логін і пароль, після чого система перевіряє правильність введених даних. У разі успішної автентифікації відкривається головне вікно системи з функціональністю, доступною для відповідної ролі;
- додавання нового студента. Користувач із відповідними правами вибирає команду додавання студента, відкриває форму профілю, заповнює основні дані, навчальні характеристики, за потреби додає фотографію та дисципліни, після чого зберігає запис;
- редагування профілю студента. Користувач знаходить потрібного студента у списку, відкриває форму редагування, змінює персональні або навчальні дані, коригує інформацію про дисципліни та зберігає зміни;
- пошук і фільтрація. Користувач вводить пошуковий запит або встановлює параметри фільтрації за групою, курсом, спеціальністю, формою навчання. Система відображає відповідний результат;
- управління користувачами. Системний адміністратор відкриває модуль користувачів, переглядає список облікових записів, додає нового користувача або змінює роль, стан активності чи інші параметри існуючого запису.

Для формалізованого опису вимог доцільно використати таблицю «подія – відгук» (табл. 2.2).

Таблиця 2.2 – Таблиця «подія – відгук»

Подія	Відгук системи
Введення правильного логіна і пароля	Відкриття головного вікна системи
Введення неправильних даних авторизації	Повідомлення про помилку входу
Натискання кнопки «Додати»	Відкриття форми створення нового запису
Натискання кнопки «Редагувати»	Відкриття форми редагування вибраного студента
Натискання кнопки «Видалити»	Видалення або запит на підтвердження видалення
Вибір фільтрів	Оновлення таблиці відповідно до умов відбору
Натискання «Зберегти»	Перевірка даних і запис змін у сховище
Додавання дисципліни	Оновлення списку дисциплін студента
Запуск модуля користувачів	Відображення списку користувачів
Відсутність доступу до функції	Блокування дії або приховування елемента інтерфейсу

На основі встановлених вимог формується модель програмного забезпечення, яка відображає склад основних компонентів системи та зв'язки між ними. До основних сутностей системи належать:

- Student – зберігає основні персональні та навчальні характеристики студента;
- StudentDiscipline – описує навчальну дисципліну, пов'язану зі студентом;
- AppUser – представляє користувача системи;
- Role – визначає категорію доступу користувача;
- EnrollmentForm – містить форму навчання;
- DisciplineStatus – характеризує стан вивчення дисципліни.

З погляду архітектури система поділяється на кілька рівнів:

- рівень представлення – форми Windows Forms;
- рівень бізнес-логіки – сервіси обробки даних, авторизації та доступу;
- рівень моделей – доменні сутності;
- рівень доступу до даних – репозиторії для MySQL.

Для відображення взаємодії користувачів із розроблюваною інформаційною системою доцільно побудувати діаграму прецедентів. Вона дозволяє наочно представити основних акторів системи та перелік дій, які вони можуть виконувати відповідно до своїх ролей. У даній системі основними акторами є адміністратор, декан, завідувач кафедри та куратор. Кожен із них має визначений набір функціональних можливостей, що відповідає реалізованій моделі розмежування доступу.

Адміністратор має найширші повноваження, оскільки виконує функції керування користувачами, редагування даних про студентів та загального адміністрування системи. Декан працює з даними студентів відповідного факультету, завідувач кафедри – з інформацією про студентів визначених спеціальностей, а куратор – із записами закріплених за ним академічних груп. До основних прецедентів системи належать авторизація, перегляд списку студентів,

пошук, фільтрація, додавання, редагування та видалення записів, робота з дисциплінами, а також керування користувачами (рис. 2.1).



Рисунок 2.1 – Діаграма прецедентів інформаційної системи обліку студентів

Для моделювання статичної структури системи використано діаграму класів UML. Вона відображає основні сутності предметної області, їх атрибути, а також зв'язки між ними. До базових класів розробленої системи належать Student, StudentDiscipline та AppUser. Додатково використовуються перелічувані типи для ролей користувачів, форми навчання та статусу дисципліни.

Центральним класом системи є клас Student, який містить персональні, контактні та навчальні відомості про студента. Клас StudentDiscipline описує навчальну дисципліну та її параметри, пов'язані з конкретним студентом. Клас AppUser використовується для представлення облікового запису користувача системи та зберігає дані, необхідні для авторизації та визначення прав доступу. Між класами Student і StudentDiscipline існує зв'язок типу «один до багатьох», оскільки один студент може вивчати багато дисциплін (рис. 2.2).

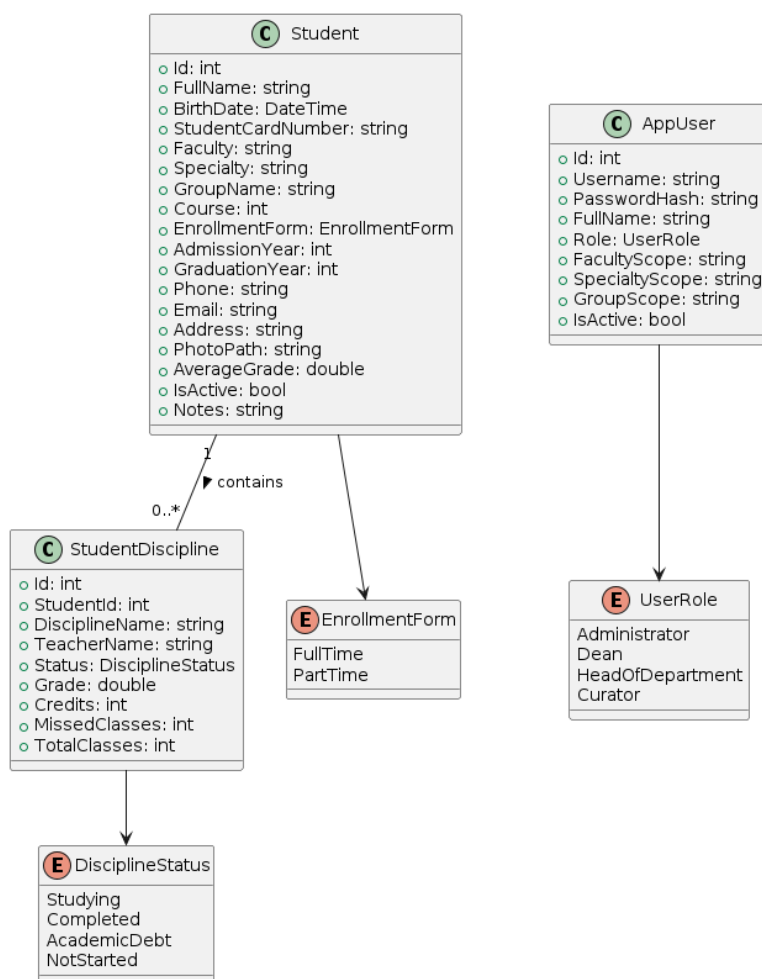


Рисунок 2.2 – Діаграма класів інформаційної системи обліку студентів

Для опису логіки виконання одного з основних бізнес-процесів розроблюваної системи побудовано діаграму діяльності. Як приклад обрано процес додавання нового студента, оскільки він включає введення даних, перевірку їх коректності, формування запису та збереження інформації в обраному сховищі даних.

У межах цього процесу користувач відкриває форму створення нового запису, вводить основні відомості про студента, за потреби додає інформацію про дисципліни, після чого система виконує перевірку коректності введених значень. Якщо перевірка пройдена успішно, дані зберігаються в MySQL залежно від поточного режиму роботи системи. У разі помилки користувач отримує повідомлення і може скоригувати введені дані (рис. 2.3).

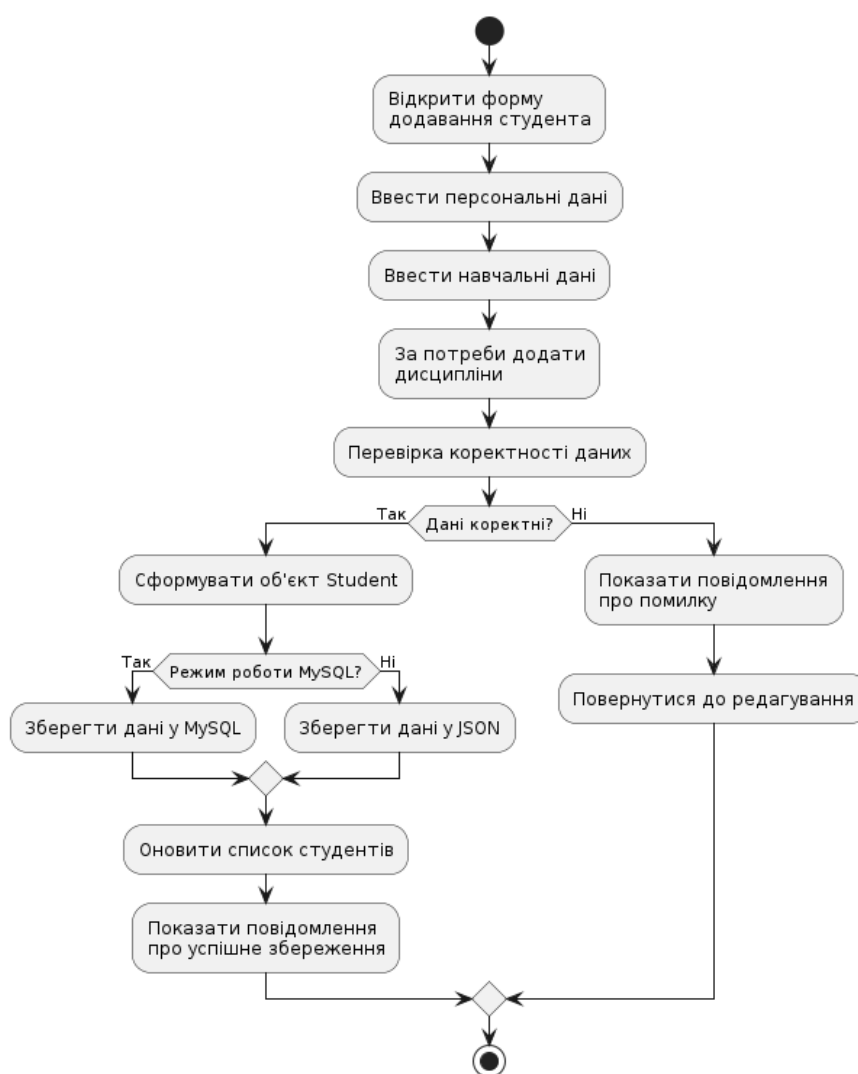


Рисунок 2.3 – Діаграма діяльності процесу додавання студента

Діаграма послідовності авторизації відображає взаємодію між користувачем, формою входу, сервісом автентифікації, репозиторієм користувачів і джерелом даних. Вона показує логіку перевірки облікових даних, визначення ролі користувача та відкриття головного вікна системи відповідно до встановлених прав доступу (рис. 2.4).

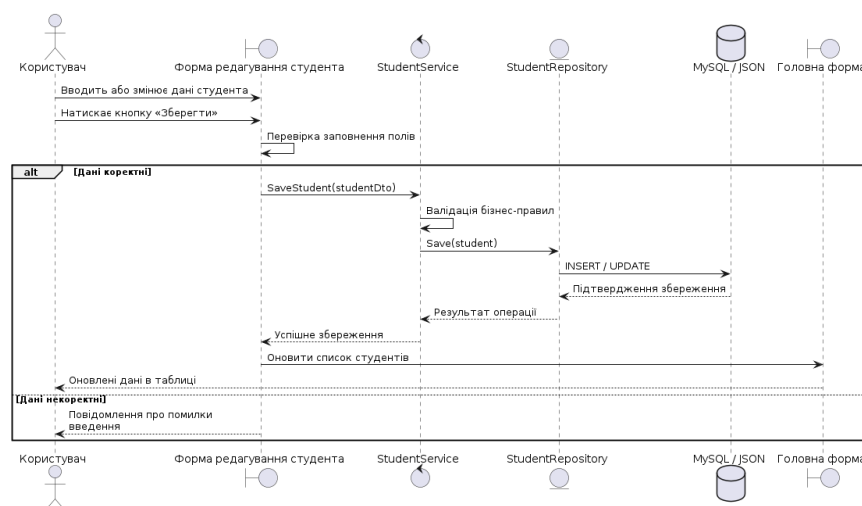


Рисунок 2.4 – Діаграма послідовності процесу авторизації користувача

Ця діаграма відображає послідовність дій під час створення або редагування профілю студента. Вона показує взаємодію між користувачем, формою редагування, сервісом роботи зі студентами, репозиторієм та джерелом даних (рис. 2.5).

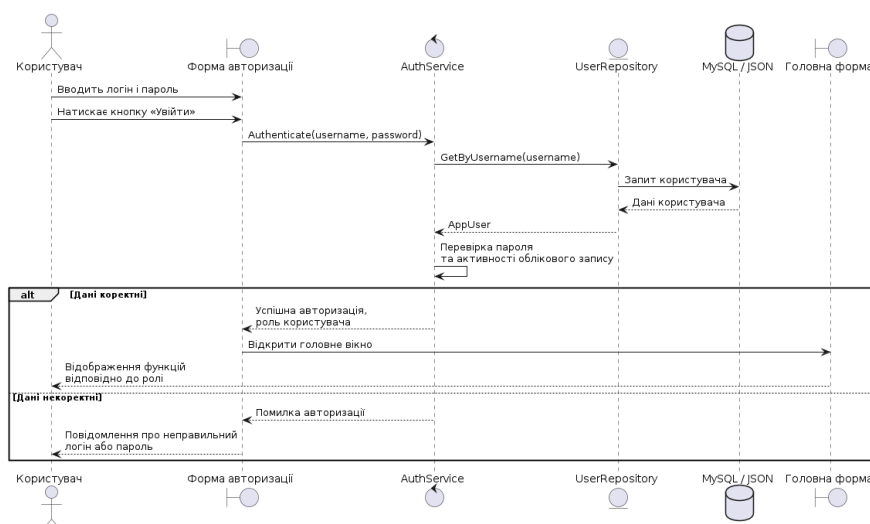


Рисунок 2.5 – Діаграма послідовності процесу збереження даних студента

У межах системи можна виділити основні інформаційні потоки.

Потік введення даних: введення користувачем інформації про студента; заповнення навчальних відомостей; введення даних про дисципліни; створення облікових записів користувачів.

Потік обробки даних: перевірка правильності введення; передавання даних до сервісного рівня; застосування бізнес-правил; передавання даних до репозиторію.

Потік збереження: запис інформації до MySQL; оновлення існуючих записів; видалення або деактивація даних.

Потік отримання інформації: завантаження списків студентів; виведення профілю студента; відображення дисциплін; формування результатів пошуку та фільтрації.

Потік керування доступом: перевірка прав користувача; обмеження доступних функцій; відбір лише дозволених даних.

Технологія інформаційного обслуговування в системі полягає у забезпеченні користувача актуальною, структурованою та доступною інформацією відповідно до його ролі. Інформаційні потреби користувачів можна описати так:

- адміністратор потребує повного керування записами студентів і користувачів;
- декан потребує узагальненого доступу до студентів свого факультету;
- завідувач кафедри потребує доступу до студентів відповідних спеціальностей;
- куратор потребує швидкого доступу до даних своїх груп і можливості коригувати пов'язані записи.

Система повинна надавати кожному користувачеві лише ту інформацію, яка необхідна для виконання його посадових функцій.

Під час проєктування інтерфейсу користувача основна увага приділялася простоті, зрозумілості та логічному групуванню функціональних елементів. Інтерфейс розроблено у вигляді настільного застосунку Windows Forms, що

забезпечує традиційний для користувача формат роботи з вікнами, формами та таблицями.

Система містить такі основні елементи інтерфейсу:

- форму авторизації, на якій користувач вводить логін і пароль;
- головне вікно системи, де розміщено таблицю студентів, кнопки керування записами, блоки фільтрації та сортування;
- форму профілю студента, поділену на вкладки: основні дані, навчання, дисципліни;
- форму редагування дисципліни, призначену для введення параметрів окремої дисципліни;
- форму керування користувачами, що дозволяє адмініструвати облікові записи.

UX-рішення орієнтовані на мінімізацію дій користувача під час виконання типових операцій. Для цього:

- часто використовувані команди винесені на головну форму;
- фільтри згруповані за логікою предметної області;
- редагування студента організоване вкладками;
- табличне подання даних забезпечує швидке сприйняття списку записів;
- дії збереження та скасування мають очевидне розташування.

Отже, аналіз вимог і проєктування програмного забезпечення дали змогу сформулювати узгоджену модель інформаційної системи обліку студентів, визначити її функціональні та нефункціональні характеристики, описати сценарії використання, інформаційні потоки та логіку взаємодії користувачів із системою. Отримані результати є достатньою основою для подальшого вибору технологій реалізації та практичного розроблення програмного продукту.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Для розробки інформаційної системи обліку студентів необхідно було обрати такі засоби та методи, які б забезпечили зручність створення програмного продукту, надійність його функціонування, простоту супроводу та можливість подальшого розширення. Оскільки розроблювана система являє собою настільний застосунок для операційної системи Windows із графічним інтерфейсом користувача, підтримкою авторизації, рольового розмежування доступу, збереженням даних у базі даних MySQL, вибір інструментів повинен був відповідати як функціональним вимогам, так і архітектурним особливостям майбутнього програмного продукту.

На сьогодні для створення подібних інформаційних систем існує значна кількість засобів розробки. Для реалізації клієнтської частини можуть використовуватися як настільні, так і веборієнтовані технології. Серед настільних підходів для платформи Windows найпоширенішими є Windows Forms, WPF, UWP, а також кросплатформні фреймворки на зразок .NET MAUI. Для розробки вебсистем можуть застосовуватися ASP.NET Core, Django, Laravel, Node.js та інші платформи. Вибір між настільним і веборієтованим підходом залежить від характеру поставленого завдання, передбачуваного способу використання системи, вимог до середовища розгортання та рівня складності адміністрування.

Для даної кваліфікаційної роботи доцільним є використання саме настільного застосунку, оскільки він орієнтований на локальну роботу працівників освітнього підрозділу, не потребує обов'язкового розгортання вебсервера, є простішим для впровадження в межах окремого робочого місця та забезпечує передбачувану конфігурацію середовища виконання. Крім того, настільний застосунок є цілком виправданим форматом реалізації для кваліфікаційної роботи, оскільки дозволяє зосередитися на логіці предметної

області, проектуванні архітектури, організації зберігання даних і реалізації основного функціоналу системи.

Як платформу розробки обрано .NET, а як мову програмування – C# [6, 7]. Цей вибір обумовлений тим, що платформа .NET є одним із найпоширеніших і найпотужніших середовищ для створення прикладного програмного забезпечення під Windows. Вона забезпечує розвинуту екосистему бібліотек, підтримку об'єктно-орієнтованого програмування, зручні засоби роботи з графічним інтерфейсом, файловою системою, мережевими ресурсами та базами даних. Мова C# є сучасною, строго типізованою, зручною для побудови структурованих програмних систем і добре пристосованою до реалізації багат шарової архітектури [7].

Серед можливих технологій побудови графічного інтерфейсу в межах .NET можна виділити Windows Forms і Windows Presentation Foundation [8]. Технологія WPF має ширші можливості щодо гнучкого оформлення інтерфейсу, використання шаблонів, стилів і складної візуалізації. Проте для даної системи обрано Windows Forms, оскільки вона є простішою у використанні, добре підходить для створення класичних настільних форм, таблиць, діалогових вікон і засобів введення даних, а також дозволяє швидко реалізувати функціональний інтерфейс без зайвого ускладнення проєкту. Для задачі обліку студентів, де основний акцент робиться на роботі з формами, таблицями, фільтрами та вікнами редагування, можливостей Windows Forms є цілком достатньо.

Для зберігання інформації про студентів, дисципліни та користувачів системи необхідно було обрати систему керування базами даних. Серед найпоширеніших рішень можна виділити MySQL, PostgreSQL, Microsoft SQL Server та SQLite. Microsoft SQL Server є потужною платформою для корпоративних застосувань, однак часто використовується в більш складних проєктах і може бути надлишковим для задачі бакалаврської роботи. PostgreSQL вирізняється високою функціональністю та розвинутими засобами обробки складних даних, проте для порівняно компактної інформаційної системи її використання також не є обов'язковим. SQLite є дуже легкою вбудованою базою

даних, однак має обмеження, якщо система орієнтується на розширення й подальшу багатокористувацьку роботу. Саме тому як основне сховище даних обрано MySQL, оскільки ця СКБД є достатньо продуктивною, популярною, підтримує реляційну модель зберігання даних, легко інтегрується з .NET-застосунками та добре підходить для реалізації зв'язків між сутностями предметної області [9, 10].

Для взаємодії застосунку з MySQL використано клієнтську бібліотеку MySqlConnection. Її вибір обумовлений тим, що вона забезпечує пряме підключення до бази даних, виконання SQL-запитів, обробку результатів вибірки, підтримку параметризованих команд і транзакцій. Використання цієї бібліотеки дає змогу реалізувати гнучкий доступ до даних без зайвого перевантаження проєкту складними абстракціями. Оскільки в даній роботі важливо продемонструвати розуміння принципів зберігання та обробки інформації, використання прямої роботи із SQL через репозиторії є цілком виправданим рішенням [9-11].

Важливим елементом вибору засобів є архітектурна організація програмного проєкту. Для даної системи доцільно застосовувати багатоваршівну архітектуру, яка передбачає розділення програми на окремі логічні рівні: рівень інтерфейсу користувача, рівень бізнес-логіки, рівень моделей та рівень доступу до даних. Такий підхід дає змогу відокремити логіку взаємодії з користувачем від логіки обробки інформації та механізмів збереження даних. У результаті підвищується зрозумілість структури програмного коду, полегшується його супровід, спрощується тестування окремих компонентів і створюються умови для подальшого розширення функціональних можливостей системи.

Для реалізації доступу до сховища даних у системі використано репозиторний підхід. Його суть полягає в тому, що робота з даними інкапсулюється в окремих класах-репозиторіях, які реалізують єдиний інтерфейс доступу до інформації. Перевага такого підходу полягає у можливості замінювати конкретне джерело даних без суттєвих змін у бізнес-логіці програми. Саме завдяки цьому в системі реалізовано режим роботи: через MySQL.

Застосування інтерфейсів репозиторіїв дозволяє зберегти єдину логіку використання даних незалежно від конкретного способу їх зберігання.

Для вирішення задач предметної області використано низку методів обробки та організації даних. Одним із основних є метод структурованого збереження інформації у вигляді пов'язаних сутностей. Студент розглядається як основний об'єкт предметної області, з яким пов'язані дисципліни, навчальні характеристики та атрибути доступу до певних категорій користувачів. Такий підхід відповідає принципам реляційного моделювання та забезпечує цілісність інформації.

Для реалізації функцій пошуку і фільтрації застосовано метод послідовного відбору записів за сукупністю критеріїв. Користувач може задавати один або кілька параметрів відбору, після чого система формує результат відповідно до обраних умов. У випадку роботи з MySQL це може реалізовуватися шляхом побудови SQL-запитів із умовами вибірки. Такий метод є зрозумілим, ефективним і добре відповідає характеру оброблюваних даних.

Для авторизації користувачів використовується метод перевірки облікових даних із визначенням ролі користувача. Після введення логіна і пароля система виконує пошук відповідного облікового запису, перевіряє правильність введеної інформації, активність запису та встановлює права доступу відповідно до ролі. На основі цього формується доступний набір функцій у головному вікні системи. Такий підхід дозволяє реалізувати базову модель безпеки та обмежити доступ до конфіденційних або службових даних.

У межах реалізації операцій додавання, редагування та видалення записів застосовується стандартний алгоритм обробки введених даних. На першому етапі користувач заповнює поля форми, після чого система виконує перевірку обов'язкових полів, коректності форматів і логічної узгодженості значень. Якщо перевірка проходить успішно, формується об'єкт відповідної сутності, який передається до сервісного шару, а далі – до репозиторію для збереження в MySQL. У разі виявлення помилки користувач отримує повідомлення і має

можливість скоригувати введену інформацію. Такий алгоритм дозволяє мінімізувати кількість некоректних записів і підтримувати цілісність даних.

Для збереження змін у базі даних використовується метод транзакційної обробки. Це особливо важливо в тих випадках, коли в межах однієї операції необхідно оновити кілька взаємопов'язаних записів, наприклад дані студента та перелік його дисциплін. Використання транзакцій дозволяє забезпечити узгодженість стану бази даних і уникнути ситуацій, коли лише частина змін була записана успішно.

Значну роль у вирішенні поставленого завдання відіграє також методика проєктування інтерфейсу користувача. Для системи обліку студентів обрано підхід, за якого головне вікно містить таблицю з основними записами, панель фільтрів і кнопки виконання основних операцій. Форми редагування побудовані за принципом логічного групування інформації: окремо подано основні персональні дані, окремо – навчальні характеристики, окремо – перелік дисциплін. Така організація спрощує сприйняття інтерфейсу користувачем і зменшує ймовірність помилок під час введення даних.

Обрані засоби та методи дозволяють ефективно вирішити поставлене завдання розробки інформаційної системи обліку студентів. Використання .NET і C# забезпечує надійну технологічну основу для створення настільного застосунку, Windows Forms дозволяє реалізувати зрозумілий графічний інтерфейс, MySQL гарантує структуроване та цілісне зберігання даних. Багатошарова архітектура та репозиторний підхід забезпечують зручність супроводу і подальшого розвитку програмного продукту, а застосовані методи пошуку, фільтрації, перевірки даних і рольового доступу відповідають потребам предметної області.

РОЗДІЛ 3

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ СТУДЕНТІВ

3.1 Практична реалізація об'єкта проєктування

Практична реалізація інформаційної системи обліку студентів передбачала створення настільного програмного застосунку, який забезпечує автоматизацію процесів введення, редагування, пошуку, перегляду та адміністрування студентських даних.

У межах цього етапу було реалізовано основні алгоритми авторизації, відкриття форм, обробки подій, валідації введених даних, фільтрації списків, а також керування записами студентів і дисциплін. Для підтвердження практичної реалізації окремих функцій у роботі доцільно навести характерні фрагменти програмного коду. Першим елементом взаємодії користувача із системою є форма авторизації. Її призначення полягає у введенні логіна та пароля, перевірці облікових даних і наданні доступу до функцій програми відповідно до ролі користувача. Після натискання кнопки входу виконується обробник події, у якому здійснюється перевірка введених значень. Інтерфейс авторизації користувача наведено на рисунку 3.1.

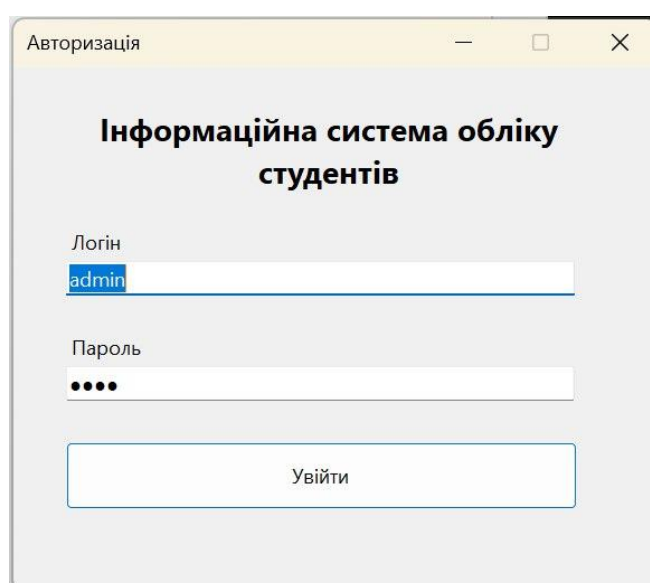


Рисунок 3.1 – Вікно авторизації інформаційної системи

Приклад реалізації такої логіки поданий у лістингу 3.1.

Лістинг 3.1 – Програмна реалізація авторизації користувача

```
private void btnLogin_Click(object sender, EventArgs e)
{
    string login = txtLogin.Text.Trim();
    string password = txtPassword.Text.Trim();
    if (string.IsNullOrEmpty(login) ||
string.IsNullOrEmpty(password))
    {
        MessageBox.Show("Введіть логін і пароль.");
        return;
    }
    var user = _authService.Authenticate(login, password);
    if (user == null)
    {
        MessageBox.Show("Невірний логін або пароль.");
        return;
    }
    if (!user.IsActive)
    {
        MessageBox.Show("Обліковий запис деактивовано.");
        return;
    }
    MainForm mainForm = new MainForm(user);
    mainForm.Show();
    this.Hide();
}
```

кінець лістингу 3.1

Наведений фрагмент демонструє базову реалізацію авторизації користувача: перевірку заповнення полів, виклик сервісу автентифікації, обробку помилкових ситуацій та відкриття головного вікна системи. Такий підхід забезпечує контроль доступу до програми вже на початковому етапі роботи.

Після успішної авторизації користувач переходить до головної форми системи, де відображається список студентів, панелі фільтрації та кнопки керування записами. У головному вікні реалізовано засоби додавання, редагування і видалення записів, а також відображення кількості знайдених та активних студентів. Значна частина логіки побудована на обробці подій

елементів інтерфейсу, що є типовим підходом для Windows Forms-застосунків. Загальний вигляд головного вікна системи зображено на рисунку 3.2.

ID	ПІБ	Група	Факультет	Спеціальність	№ квитка	Курс	Форма	Вступ	Завершен	Серь. бал	Дисциплі	Активні
11	Іваненко Іван Петрович	ІП3-21	Факультет комп'ютерних наук	Інженерія програмного забезпечення	SE-2021-0...	4	Денна	2021	2025	86,5	2	✓
12	Петренко Марія Олександрівна	ІП3-22	Факультет комп'ютерних наук	Інженерія програмного забезпечення	SE-2022-0...	3	Денна	2022	2026	94,2	2	✓
13	Сидоренко Андрій Васильович	КН-23	Факультет комп'ютерних наук	Комп'ютерні науки	CS-2023-0...	2	Заочна	2023	2027	76,8	1	✓
14	Бондаренко Олег Сергійович	ІП3-23	Факультет комп'ютерних наук	Інженерія програмного забезпечення	SE-2023-0...	2	Денна	2023	2027	82,3	1	✓
15	Шевченко Наталія Ігорівна	КН-22	Факультет комп'ютерних наук	Комп'ютерні науки	CS-2022-0...	3	Денна	2022	2026	91,4	1	✓
16	Коваленко Дмитро Олександрович	ІП3-24	Факультет комп'ютерних наук	Інженерія програмного забезпечення	SE-2024-0...	1	Денна	2024	2028	79,6	1	✓
17	Мельничук Ольга Вікторівна	КН-21	Факультет комп'ютерних наук	Комп'ютерні науки	CS-2021-0...	4	Заочна	2021	2025	85,1	1	✓
18	Ткачук Владислав Романович	ІП3-22	Факультет комп'ютерних наук	Інженерія програмного забезпечення	SE-2022-0...	3	Денна	2022	2026	83,7	1	✓
19	Демчук Юрій Іванович	КН-24	Факультет комп'ютерних наук	Комп'ютерні науки	CS-2024-0...	1	Денна	2024	2028	71,2	1	✓
20	Кравчук Анна Петрівна	ІП3-21	Факультет комп'ютерних наук	Інженерія програмного забезпечення	SE-2021-0...	4	Денна	2021	2025	92,6	1	✓
21	Петренко Петро Петрович	ІП3-11	Комп'ютерні науки	Інженерія програмного забезпечення	SE-2021-0...	1	Денна	2026	2030	50	0	✓

Рисунок 3.2 – Головне вікно системи обліку студентів

Відкриття форми редагування даних студента реалізоване в лістингу 3.2.

Лістинг 3.2 – Програмна реалізація відкриття форми редагування даних студента

```
private void btnEditStudent_Click(object sender, EventArgs e)
{
    if (dgvStudents.CurrentRow == null)
    {
        MessageBox.Show("Оберіть студента для редагування.");
        return;
    }
    var student = (Student)dgvStudents.CurrentRow.DataBoundItem;
    using (var form = new StudentForm(student))
    {
        if (form.ShowDialog() == DialogResult.OK)
        {
            LoadStudents();
        }
    }
}
```

кінець лістингу 3.2

Цей код показує, як у системі організовано взаємодію між головною формою та формою профілю студента. Після вибору потрібного запису відкривається окреме діалогове вікно, у якому користувач може змінити необхідні дані, а після завершення редагування список автоматично оновлюється.

Важливою частиною практичної реалізації стало створення механізму пошуку та фільтрації студентів. Оскільки система орієнтована на роботу з великою кількістю записів, необхідно було забезпечити швидке знаходження потрібної інформації за різними критеріями. Для цього реалізовано загальний пошук, а також фільтрацію за групою, спеціальністю, курсом, формою навчання, роками вступу і завершення навчання, а також за параметрами дисциплін.

Приклад формування набору умов для пошуку зображений в лістингу 3.3.

Лістинг 3.3 – Програмна реалізація формування умов пошуку та фільтрації студентів

```
private void ApplyFilters()
{
    string keyword = txtSearch.Text.Trim().ToLower();
    string speciality = txtSpeciality.Text.Trim().ToLower();
    string educationForm =
cmbEducationForm.SelectedItem?.ToString();
    var filtered = _students.Where(s =>
        (string.IsNullOrEmpty(keyword) ||
s.FullName.ToLower().Contains(keyword)) &&
        (string.IsNullOrEmpty(speciality) ||
s.Speciality.ToLower().Contains(speciality)) &&
        (educationForm == "Yci" || s.EducationForm ==
educationForm)
    ).ToList();
    dgvStudents.DataSource = filtered;
    lblCount.Text = $"Записів: {filtered.Count}";
}

```

кінець лістингу 3.3

Наведений фрагмент ілюструє програмну реалізацію багатокритеріальної фільтрації даних у пам'яті застосунку. Аналогічний підхід може

використовуватися і при роботі з даними, що надходять із зовнішнього джерела збереження.

Для забезпечення надійності функціонування системи було реалізовано валідацію введених даних у формах редагування. Перед збереженням інформації перевіряється наявність обов'язкових полів, правильність числових значень, узгодженість років навчання, допустимі діапазони середнього балу та інших показників.

Для роботи з профілем студента було створено окрему форму, організовану за вкладками: «Основні дані», «Навчання» та «Дисципліни». Це дало змогу логічно розподілити інформацію за тематичними блоками та підвищити зручність користування. На вкладці основних даних реалізовано введення персональної інформації, контактних даних, фото та приміток (рис. 3.3).

Рисунок 3.3 – Форма профілю студента, вкладка «Основні дані»

На вкладці навчання відображаються дані про спеціальність, номер студентського квитка, курс, форму навчання, рік вступу та завершення навчання. Ця вкладка подана на рисунку 3.4.

Профіль студента

Основні дані Навчання Дисципліни

Спеціальність: Інженерія програмного забезпечення

Номер студентського квитка: SE-2021-001

Рік вступу: 2021

Рік завершення навчання: 2025

Курс: 4

Форма навчання: Денна

Зберегти Скасувати

Рисунок 3.4 – Форма профілю студента, вкладка «Навчання»

Вкладка дисциплін дає змогу переглядати список дисциплін студента та керувати ним через окремі діалогові форми; її інтерфейс наведено на рисунку 3.5.

Профіль студента

Основні дані Навчання Дисципліни

Дисципліна	Статус	Бал	Кредит	Пропуски	Всього занять	Викладач
Архітектура прог...	Завершено	90	5	1	30	Ковальчук О.В.
DevOps	В процесі	86	4	2	24	Мельник І.І.

Додати дисципліну Редагувати дисципліну Видалити дисципліну

Зберегти Скасувати

Рисунок 3.5 – Форма профілю студента, вкладка «Дисципліни»

Наприклад, додавання нової дисципліни до профілю студента може бути реалізовано як в лістингу 3.4.

Лістинг 3.4 – Програмна реалізація додавання дисципліни до профілю студента

```
private void btnAddDiscipline_Click(object sender, EventArgs e)
{
    using (var form = new DisciplineForm())
    {
        if (form.ShowDialog() == DialogResult.OK)
        {
            _studentDisciplines.Add(form.Discipline);
            RefreshDisciplineGrid();
        }
    }
}
```

кінець лістингу 3.4

Цей фрагмент демонструє принцип розширення профілю студента пов'язаними об'єктами. Після створення нового запису дисципліни таблиця автоматично оновлюється, а зміни стають доступними для подальшої обробки. Діалогове вікно додавання та редагування дисципліни наведено на рисунку 3.6.

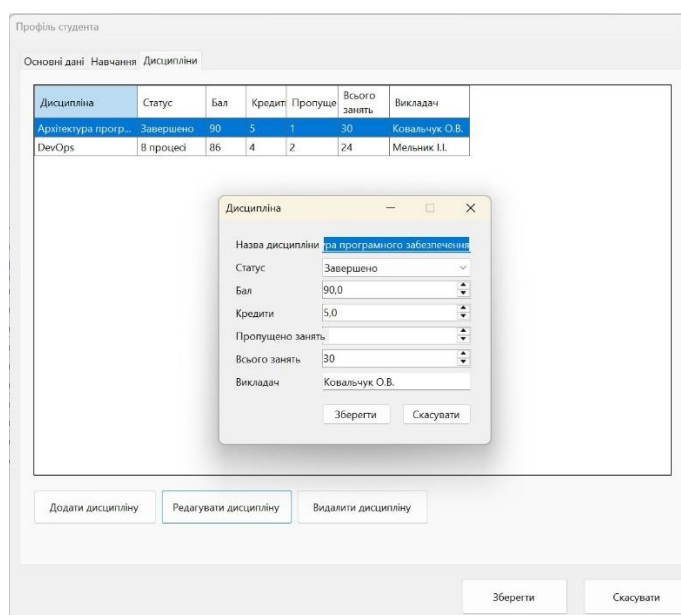


Рисунок 3.6 – Вікно додавання та редагування дисципліни студента

Окремим модулем системи стала форма адміністрування користувачів, яка дозволяє керувати обліковими записами. У цій частині застосунку реалізовано відображення списку користувачів, створення нових записів, зміну ролей, редагування параметрів і зміну статусу активності. Це забезпечує підтримку рольової моделі доступу та дозволяє адаптувати систему до організаційної структури навчального закладу. Інтерфейс модуля адміністрування користувачів наведено на рисунку 3.7.

ID	Логін	Пароль	ПІБ	Роль	Факультет	Активні
1	admin	1234	Системний адміністратор	Admin		☒
2	dean	1234	Декан факультету	Dean	Факультет комп'ют...	☒
3	head.se	1234	Завідувач кафедри ПЗ	HeadOfDepart...		☒
4	head.multi	1234	Завідувач двох спеціал...	HeadOfDepart...		☒
5	curator.ipz	1234	Куратор груп ІПЗ-21 та...	Curator		☒
6	curator.kn	1234	Куратор груп КН-23 та ...	Curator		☒

Додати Редагувати Видалити Увімкнути /

Рисунок 3.7 – Вікно адміністрування користувачів системи

У процесі реалізації об'єкта проєктування було дотримано модульного принципу побудови програмного забезпечення. Інтерфейсні компоненти, бізнес-логіка, засоби перевірки даних і механізми доступу до інформації реалізовано у вигляді окремих логічних частин, що спрощує супровід системи, тестування та подальшу модернізацію.

3.2 Розробка бази даних

У розробленій системі база даних використовується для збереження персональних відомостей про студентів, характеристик їхнього навчання, даних про дисципліни, а також облікових записів користувачів, що працюють із системою.

Для реалізації сховища даних у кваліфікаційній роботі було обрано систему керування базами даних MySQL. Вибір цього засобу зумовлений тим, що MySQL є поширеною реляційною СКБД, яка забезпечує підтримку зв'язків між таблицями, використання первинних і зовнішніх ключів, обмежень цілісності, механізмів індексації та транзакційної обробки даних. Крім того, MySQL добре інтегрується із застосунками, розробленими на платформі .NET, і має зручні інструменти адміністрування, зокрема середовище MySQL Workbench, у якому можна створювати таблиці, переглядати записи та контролювати структуру бази даних.

Під час проєктування бази даних було враховано основні сутності предметної області та зв'язки між ними. У центрі моделі перебуває сутність студента, оскільки саме з нею пов'язані більшість функцій розробленої системи. Для кожного студента необхідно зберігати персональні дані, контактну інформацію, відомості про навчання, статус активності та пов'язані дисципліни. Окрім цього, система підтримує рольовий доступ, а тому окремо повинні зберігатися відомості про користувачів, які входять до системи, а також параметри, що визначають доступ до окремих груп і спеціальностей.

У результаті було сформовано структуру бази даних, яка включає кілька основних таблиць: `students`, `student_disciplines`, `users`, `user_groups` та `user_specialties`. Такий поділ даних дозволяє уникнути надмірного дублювання, забезпечити цілісність інформації та створити гнучку структуру, придатну до подальшого розширення.

Для наочного представлення структури бази даних та взаємозв'язків між її основними сутностями побудовано ER-діаграму, яку наведено на рисунку 3.9. Вона відображає склад основних таблиць бази даних, їх ключові атрибути та зв'язки між записами, що забезпечують цілісність і узгодженість збереження інформації в системі.

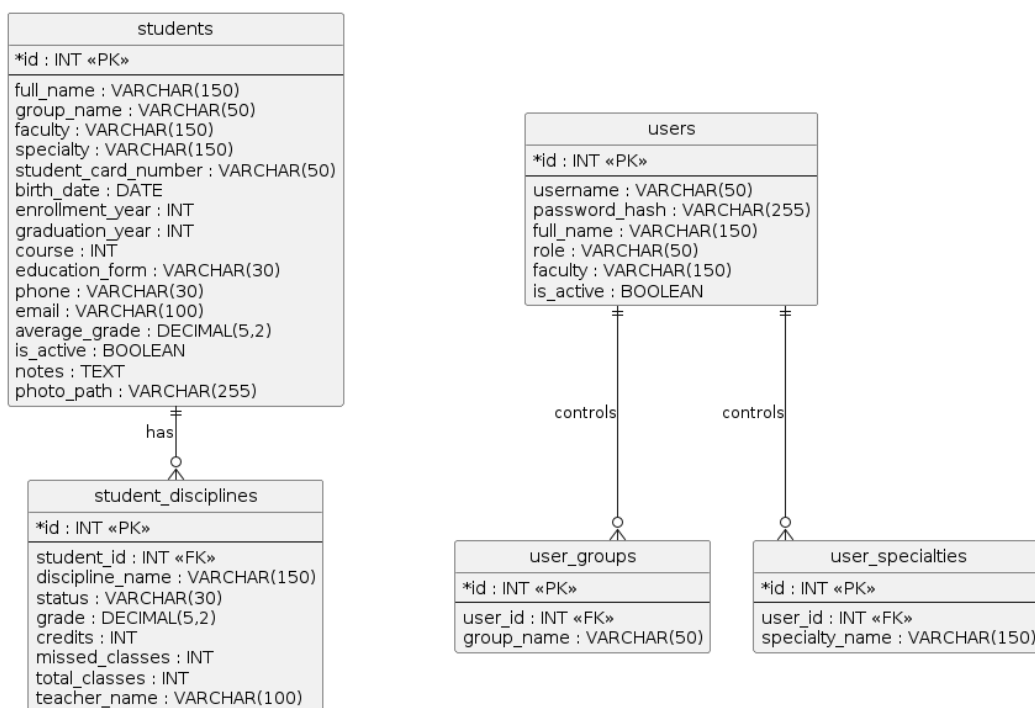


Рисунок 3.9 – ER-діаграма бази даних інформаційної системи обліку студентів

Центральною таблицею бази даних є таблиця `students`, у якій зберігаються основні відомості про студентів. До неї включено поля для збереження повного імені студента, групи, факультету, спеціальності, номера студентського квитка, дати народження, року вступу, року завершення навчання, курсу, форми навчання, контактних даних, середнього балу, статусу активності, приміток і шляху до фотографії. Саме дані цієї таблиці формують основу для відображення записів у головному вікні системи та використовуються для пошуку, фільтрації, редагування і перегляду профілю студента.

SQL-код створення таблиці `students` наведено в лістингу 3.5.

Лістинг 3.5 – Програмна реалізація збереження даних студента

```

CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    full_name VARCHAR(150) NOT NULL,
    group_name VARCHAR(50),
    faculty VARCHAR(150),
    specialty VARCHAR(150),
    student_card_number VARCHAR(50) UNIQUE,
    birth_date DATE,
    enrollment_year INT,

```

```

graduation_year INT,
course INT,
education_form VARCHAR(30),
phone VARCHAR(30),
email VARCHAR(100),
average_grade DECIMAL(5,2),
is_active BOOLEAN DEFAULT TRUE,
notes TEXT,
photo_path VARCHAR(255)
);

```

кінець лістингу 3.5

У наведеному фрагменті поле `id` використовується як первинний ключ, що однозначно ідентифікує кожного студента. Поле `student_card_number` визначено як унікальне, що дозволяє уникнути дублювання номерів студентських квитків. Для збереження середнього балу використано тип `DECIMAL`, який забезпечує точність числових значень, а для дати народження – тип `DATE`. Така структура є достатньою для збереження основної інформації про студента та відповідає вимогам розробленої інформаційної системи.

Для збереження відомостей про дисципліни студентів було створено окрему таблицю `student_disciplines`. Необхідність її введення пояснюється тим, що одному студентові може відповідати кілька дисциплін, кожна з яких має власні параметри: назву, статус вивчення, бал, кількість кредитів, кількість пропущених занять, загальну кількість занять і викладача. Саме тому між таблицями `students` і `student_disciplines` реалізовано зв'язок типу один до багатьох, за якого один студент може мати багато пов'язаних записів про дисципліни.

У цій таблиці поле `student_id` є зовнішнім ключем, який посилається на запис у таблиці `students`. Використання зовнішнього ключа забезпечує збереження логічного зв'язку між студентом і його дисциплінами. Параметр `ON DELETE CASCADE` означає, що в разі видалення запису про студента всі пов'язані з ним дисципліни також будуть автоматично видалені. Це дозволяє підтримувати цілісність бази даних і уникати появи записів, які не мають зв'язку з основною сутністю.

Окремою складовою структури бази даних є таблиця `users`, яка використовується для збереження облікових записів користувачів системи. У ній зберігаються логін, пароль, повне ім'я користувача, його роль, факультет і статус активності. Саме ці дані застосовуються під час авторизації користувача в системі та подальшого визначення доступних функцій залежно від його повноважень.

Така структура таблиці дозволяє зберігати достатній обсяг інформації для підтримки рольової моделі доступу. Наявність унікального поля `username` запобігає створенню кількох користувачів із однаковим логіном, а поле `is_active` використовується для активації або блокування облікового запису без його фізичного видалення з бази даних.

Для реалізації деталізованого розмежування доступу у системі було також передбачено допоміжні таблиці `user_groups` та `user_specialties`. Вони застосовуються для збереження інформації про те, до яких академічних груп або спеціальностей має доступ певний користувач. Такий підхід є доцільним, оскільки не всі ролі в системі мають глобальний доступ до всієї інформації. Наприклад, куратор може працювати лише з визначеними групами, а завідувач кафедри – зі студентами окремих спеціальностей.

Під час розробки бази даних було враховано принципи нормалізації. Основна ідея полягала в тому, щоб розподілити дані між таблицями відповідно до їх змісту та зв'язків, уникаючи дублювання інформації в одному місці. Саме тому дисципліни було винесено в окрему таблицю, а інформацію про права доступу користувачів – у додаткові таблиці зв'язків. Це не лише спрощує оновлення даних, а й робить структуру бази більш гнучкою та зручною для масштабування.

Окрім створення таблиць, важливим етапом було наповнення бази даних тестовими записами та перевірка виконання основних операцій: додавання студентів, редагування персональних відомостей, перегляду списку студентів, збереження дисциплін, створення користувачів та перевірки їхніх прав доступу. Таке тестування дозволило переконатися, що структура бази даних відповідає

логіці предметної області та забезпечує коректне функціонування всіх основних сценаріїв роботи системи.

Фактичне представлення записів у таблиці `students` у середовищі MySQL Workbench наведено на рисунку 3.8. Цей рисунок ілюструє практичний результат реалізації бази даних та підтверджує можливість збереження й перегляду студентських даних у зовнішньому сховищі.

id	full_name	group_name	faculty	specialty	student_card_number	birth_date	enrollment_year	graduation_year
11	Іваненко Іван Петрович	ІТЗ-21	Факультет комп'ютерних інформаційних тех...	Інженерія програмного забезпечення	SE-2021-001	2004-03-12	2021	2025
12	Петренко Марія Олександрівна	ІТЗ-22	Факультет комп'ютерних інформаційних тех...	Інженерія програмного забезпечення	SE-2022-014	2005-01-20	2022	2026
13	Сидоренко Андрій Васильович	ІН-23	Факультет комп'ютерних інформаційних тех...	Комп'ютерна наука	CS-2023-021	2005-09-08	2023	2027
14	Бондаренко Олег Сергійович	ІТЗ-23	Факультет комп'ютерних інформаційних тех...	Інженерія програмного забезпечення	SE-2023-032	2006-02-11	2023	2027
15	Шевченко Наталя Ігорівна	ІН-22	Факультет комп'ютерних інформаційних тех...	Комп'ютерна наука	CS-2022-017	2004-12-05	2022	2026
16	Коваленко Дмитро Олександрович	ІТЗ-24	Факультет комп'ютерних інформаційних тех...	Інженерія програмного забезпечення	SE-2024-005	2006-08-17	2024	2028
17	Мельничук Олена Вікторівна	ІН-21	Факультет комп'ютерних інформаційних тех...	Комп'ютерна наука	CS-2021-009	2003-11-14	2021	2025
18	Ткачук Владислав Романович	ІТЗ-22	Факультет комп'ютерних інформаційних тех...	Інженерія програмного забезпечення	SE-2022-030	2005-04-09	2022	2026
19	Демчук Юрій Іванович	ІН-24	Факультет комп'ютерних інформаційних тех...	Комп'ютерна наука	CS-2024-011	2006-06-30	2024	2028
20	Кравчук Ана Петрівна	ІТЗ-21	Факультет комп'ютерних інформаційних тех...	Інженерія програмного забезпечення	SE-2021-018	2004-07-25	2021	2025
21	Петренко Петро Петрович	ІТЗ-11	Комп'ютерні інформаційні технології			2026-03-13	2026	2030

Рисунок 3.8 – Відображення таблиці `students` бази даних інформаційної системи у середовищі MySQL Workbench

Для взаємодії програмного застосунку з базою даних у середовищі .NET використовувалися SQL-запити на додавання, вибірку, оновлення та видалення записів. Наприклад, під час збереження нового студента у базу даних використовується команда вставки нового запису, яка передає значення всіх основних полів до таблиці `students`.

3.3 Захист інформаційно-комп'ютерної системи

Під час розробки інформаційної системи обліку студентів важливим завданням було забезпечення належного рівня захисту інформації, оскільки система працює з персональними даними здобувачів освіти, службовою інформацією про навчальний процес та обліковими записами користувачів.

Порушення цілісності, конфіденційності або доступності таких даних може призвести до втрати важливої інформації, несанкціонованого доступу до персональних відомостей, спотворення результатів обліку та порушення нормальної роботи програмного продукту. Саме тому при створенні системи було передбачено комплекс заходів, спрямованих на підвищення її безпеки [12].

Одним із базових механізмів захисту в розробленій системі є автентифікація користувачів. Доступ до основного функціоналу програми надається лише після введення логіна та пароля у вікні авторизації. Такий підхід дозволяє обмежити використання системи сторонніми особами та забезпечити персоніфікований доступ до даних. Вхід до системи виконується лише після перевірки правильності облікових даних, а в разі помилки користувач отримує відповідне повідомлення. Інтерфейс авторизації було наведено раніше на рисунку 3.1.

Після успішної автентифікації в системі реалізується авторизація, тобто надання доступу до функцій відповідно до ролі користувача. У програмному продукті передбачено кілька ролей, зокрема адміністратор, декан, завідувач кафедри та куратор. Для кожної з них визначено власний рівень доступу до інформації. Адміністратор має повний доступ до керування користувачами та студентськими даними, тоді як інші ролі працюють лише з тією частиною інформації, яка відповідає їх службовим повноваженням. Такий підхід дозволяє реалізувати базове розмежування доступу між категоріями користувачів.

Суттєвим елементом захисту є розмежування доступу до даних на рівні предметної області. У системі передбачено можливість обмеження доступу користувачів до окремих факультетів, спеціальностей або академічних груп. Це означає, що користувач працює лише з тими записами, які належать до сфери його відповідальності. Наприклад, куратор має доступ до студентів конкретної групи, а завідувач кафедри – до студентів певних спеціальностей. Такий механізм не лише підвищує рівень інформаційної безпеки, а й відповідає організаційній структурі закладу освіти.

Ще одним важливим засобом захисту є контроль активності облікових записів. У системі передбачено можливість блокування або деактивації користувача без фізичного видалення його запису з бази даних. Завдяки цьому адміністратор може оперативним чином припинити доступ до системи для окремої особи, зберігши при цьому інформацію про існування такого облікового запису. Це особливо важливо у випадках зміни персоналу, завершення терміну повноважень або необхідності тимчасового обмеження доступу.

На рівні програмної реалізації важливу роль у забезпеченні безпеки відіграє валідація введених даних. Усі основні форми системи містять перевірки, що контролюють заповнення обов'язкових полів, допустимість числових значень, правильність логічних залежностей між параметрами та загальну коректність введеної інформації. Такий контроль дозволяє запобігти випадковому внесенню помилкових даних, які можуть призвести до спотворення записів або порушення цілісності бази даних. Наприклад, у системі перевіряється правильність років вступу та завершення навчання, межі середнього балу, наявність ПІБ студента та інших обов'язкових реквізитів.

Окрему увагу під час розробки приділено захисту бази даних від некоректних змін. Для цього в MySQL використовуються первинні ключі, зовнішні ключі, унікальні обмеження та зв'язки між таблицями. Такі механізми забезпечують цілісність структури даних і запобігають появі суперечливих записів. Наприклад, унікальність номера студентського квитка не дозволяє створити кілька студентів з однаковим ідентифікатором, а зовнішні ключі між таблицями студентів і дисциплін підтримують правильність зв'язків між записами. Структуру бази даних і взаємозв'язки між її сутностями було подано в підрозділі 3.2.

Для зменшення ризику несанкціонованого впливу на базу даних у програмі доцільно використовувати параметризовані SQL-запити. Такий підхід знижує ймовірність SQL-ін'єкцій і забезпечує коректну передачу значень від прикладної програми до системи керування базами даних. Навіть у разі введення користувачем спеціальних символів або нетипових значень параметризований

запит інтерпретує їх як дані, а не як частину SQL-команди. Це є важливим елементом програмного захисту при взаємодії з зовнішнім сховищем інформації.

Використання параметризованого SQL-запиту під час додавання студента зображено в лістингу 3.6.

Лістинг 3.6 – Програмна реалізація збереження змін до бази даних MySQL

```

string query = @"INSERT INTO students
(full_name, group_name, faculty, specialty, student_card_number,
birth_date,
enrollment_year, graduation_year, course, education_form, phone,
email,
average_grade, is_active, notes, photo_path)
VALUES
(@full_name, @group_name, @faculty, @specialty,
@student_card_number, @birth_date,
@enrollment_year, @graduation_year, @course, @education_form,
@phone, @email,
@average_grade, @is_active, @notes, @photo_path)";

using var command = new MySqlCommand(query, connection);
command.Parameters.AddWithValue("@full_name", student.FullName);
command.Parameters.AddWithValue("@group_name", student.GroupName);
command.Parameters.AddWithValue("@faculty", student.Faculty);
command.Parameters.AddWithValue("@specialty", student.Specialty);
command.Parameters.AddWithValue("@student_card_number",
student.StudentCardNumber);

```

кінець лістингу 3.6

Наведений фрагмент демонструє, що значення передаються до SQL-команди через параметри, а не шляхом безпосереднього формування рядка запиту. Це підвищує рівень безпеки та робить роботу з базою даних більш надійною.

До заходів захисту інформаційно-комп'ютерної системи також належать організаційні засоби безпеки. Вони полягають у визначенні правил використання системи, призначенні відповідальних осіб, обмеженні доступу до робочих місць, контролі за передачею облікових даних третім особам та дотриманні правил роботи з персональними даними. Навіть за наявності належних програмних механізмів захисту значна частина ризиків пов'язана саме з людським фактором,

тому організаційні заходи є важливим доповненням до технічних засобів безпеки.

У перспективі рівень захисту розробленої системи може бути додатково підвищений шляхом використання хешування паролів, ведення журналу дій користувачів, автоматичного завершення сеансу при тривалій неактивності, розмежування доступу на рівні окремих операцій, а також впровадження механізмів шифрування резервних копій. У межах кваліфікаційної роботи реалізовані базові та достатньо ефективні механізми захисту, які відповідають характеру поставленого завдання та особливостям розробленої інформаційної системи.

3.4 Тестування та налагодження інформаційно-комп'ютерної системи

Після завершення основного етапу розробки інформаційної системи обліку студентів було виконано тестування та налагодження програмного забезпечення. Метою цього етапу є перевірка коректності роботи реалізованих функцій, виявлення помилок у логіці обробки даних, оцінювання стійкості системи до типових помилок користувача та підтвердження відповідності програмного продукту поставленим вимогам. Оскільки розроблена система працює з персональними даними студентів, навчальною інформацією та механізмами розмежування доступу, особливу увагу було приділено правильності функціонування інтерфейсу, збереженню даних, роботі з базою даних і перевірці прав доступу користувачів.

Тестування розробленого програмного забезпечення здійснювалося комплексно. Насамперед було проведено функціональне тестування, у межах якого перевірялася робота основних модулів системи. Зокрема, було перевірено коректність авторизації користувачів, можливість додавання, редагування та видалення записів про студентів, роботу з навчальними дисциплінами, пошук, фільтрацію, сортування, а також механізми управління користувачами.

Результати такого тестування дали змогу переконатися, що ключові функції програми реалізовано відповідно до поставлених завдань.

Важливим етапом стало тестування інтерфейсу користувача. У процесі перевірки оцінювалася зручність переходу між формами, правильність розташування елементів керування, коректність відображення таблиць, полів введення, кнопок та діалогових вікон. Також перевірялося, чи отримує користувач зрозумілі повідомлення про помилки, підтвердження виконання дій і результати операцій пошуку або збереження. Особливу увагу було звернено на тестування форм введення та редагування даних, оскільки саме на цьому етапі найчастіше виникають помилки користувача або некоректне заповнення полів.

Окремий напрям тестування становила перевірка коректності роботи з базою даних MySQL. У ході цього етапу контролювалося створення записів у таблицях, оновлення даних, видалення записів, збереження зв'язків між студентами та дисциплінами, а також правильність відображення змін у користувацькому інтерфейсі після виконання SQL-операцій. Крім того, перевірялася відповідність типів даних, обробка порожніх значень, коректність запису числових та текстових параметрів і стабільність роботи програми у випадку повторного відкриття або оновлення форм. Результати перевірки показали, що застосування реляційної моделі та параметризованих SQL-запитів забезпечує належну надійність взаємодії прикладної програми з базою даних.

Під час тестування також було перевірено правильність реалізації розмежування прав доступу. Для цього вхід до системи виконувався під різними обліковими записами, яким призначено ролі адміністратора, декана, завідувача кафедри та куратора. Було встановлено, що адміністратор має повний доступ до функціоналу системи, тоді як інші категорії користувачів працюють лише з визначеним колом даних відповідно до їхніх повноважень. Така перевірка дозволила підтвердити коректність роботи механізму авторизації та відповідність реалізованої моделі доступу вимогам інформаційної безпеки.

У процесі налагодження значна увага приділялася перевірці валідації даних. Аналізувалися ситуації введення неповної, некоректної або логічно

суперечливої інформації. Зокрема, перевірялося, чи блокує система збереження запису у випадках, коли не заповнено обов'язкові поля, вказано помилкові числові значення, порушено формат даних або введено нелогічні параметри, наприклад рік завершення навчання, що передує року вступу. У результаті налагодження було вдосконалено механізми перевірки введених значень, що дозволило підвищити достовірність інформації та знизити ризик внесення помилкових даних до системи.

Крім позитивних сценаріїв використання, було розглянуто і негативні сценарії, які моделювали можливі збої та помилки. Зокрема, перевірялася поведінка системи у випадках неправильного введення логіна або пароля, спроби збереження незаповнених форм, введення недопустимих символів, роботи без активного підключення до бази даних. Такий підхід дозволив оцінити стійкість програми до типових помилок експлуатації та своєчасно усунути недоліки, що могли б погіршити користувацький досвід.

У ході налагодження було виявлено та усунуто низку технічних недоліків, які виникали на різних етапах реалізації. До них належали помилки відображення окремих полів у формах, неточності у фільтрації записів, некоректне оновлення таблиць після редагування, проблеми із завантаженням фотографій студентів, а також окремі ситуації, за яких дані зберігалися не в повному обсязі. Усунення таких недоліків виконувалося шляхом перевірки логіки обробки подій, корекції запитів до бази даних, уточнення валідаційних умов і доопрацювання взаємодії між рівнями архітектури системи.

Для систематизації результатів тестування доцільно подати основні перевірки у вигляді таблиці 3.1.

Таблиця 3.1 – Основні результати тестування інформаційної системи

№	Об'єкт тестування	Суть перевірки	Результат
1	Авторизація користувача	Перевірка входу за правильними та неправильними обліковими даними	Авторизація працює коректно, при помилці виводиться повідомлення
2	Додавання студента	Створення нового запису з повним набором даних	Запис успішно додається та відображається у списку

Продовження таблиці 3.1

№	Об'єкт тестування	Суть перевірки	Результат
3	Редагування студента	Зміна персональних і навчальних даних студента	Зміни зберігаються коректно
4	Видалення студента	Видалення запису зі списку студентів	Видалення виконується успішно
5	Робота з дисциплінами	Додавання, редагування та видалення дисциплін студента	Операції виконуються правильно
6	Пошук і фільтрація	Пошук за ключовими словами, групою, факультетом, спеціальністю	Пошук і фільтрація працюють коректно
7	Розмежування доступу	Перевірка доступу для різних ролей користувачів	Обмеження доступу реалізовано правильно
8	Валідація даних	Перевірка реакції системи на порожні та некоректні значення	Некоректні дані не зберігаються
9	Робота з MySQL	Збереження, оновлення та читання записів із БД	Операції виконуються стабільно

За результатами проведеного тестування можна зробити висновок, що розроблена інформаційна система обліку студентів у цілому працює стабільно та забезпечує реалізацію передбачених функціональних можливостей. Налагодження дозволило усунути виявлені помилки, покращити зручність роботи користувача, підвищити надійність збереження даних і забезпечити узгоджену роботу окремих програмних модулів. Отримані результати підтверджують працездатність створеного програмного продукту та його придатність до використання для автоматизації обліку студентів у закладі вищої освіти.

ВИСНОВКИ

У кваліфікаційній роботі бакалавра виконано розробку інформаційної системи обліку студентів із використанням платформи .NET та системи керування базами даних MySQL. У процесі роботи було досліджено предметну область, пов'язану з автоматизацією обліку студентського контингенту у закладі вищої освіти, проаналізовано сучасний стан проблеми, існуючі підходи до ведення студентських даних, а також виявлено основні недоліки традиційних способів обліку, пов'язані з фрагментарністю збереження інформації, складністю її актуалізації, ризиком дублювання даних та недостатнім рівнем розмежування доступу.

У процесі виконання кваліфікаційної роботи було розв'язано всі поставлені завдання.

Проаналізовано сучасний стан проблеми автоматизації обліку студентів та існуючі підходи до її розв'язання. У роботі досліджено особливості ведення студентського обліку в закладах вищої освіти, визначено основні труднощі традиційного підходу, пов'язані з великим обсягом даних, складністю їх актуалізації, дублюванням інформації та недостатньою оперативністю доступу. Проведений аналіз показав доцільність використання сучасних інформаційних систем для централізованого збереження й обробки відомостей про студентів.

Розглянуто аналоги інформаційних систем обліку студентів, визначено їх переваги та недоліки. У першому розділі було проаналізовано існуючі програмні рішення, що застосовуються для автоматизації освітнього процесу та ведення студентських даних. У результаті порівняння встановлено, що наявні системи мають ряд переваг, зокрема широкий функціонал і підтримку облікових операцій, однак часто є надмірно складними, недостатньо гнучкими або орієнтованими на масштабні корпоративні впровадження. Це стало підставою для обґрунтування створення власного програмного продукту.

Сформульовано вимоги до розроблюваної інформаційної системи. У другому розділі було визначено функціональні та нефункціональні вимоги до

програмного забезпечення. Зокрема, встановлено вимоги до авторизації користувачів, ведення обліку студентів, роботи з дисциплінами, пошуку, фільтрації, управління користувачами, коректності введення даних, надійності, зручності використання та безпеки. Це створило основу для подальшого проєктування та реалізації системи.

Обґрунтовано вибір технологій, засобів і методів реалізації програмного продукту. У роботі аргументовано доцільність використання мови програмування C#, платформи .NET, технології Windows Forms, системи керування базами даних MySQL, бібліотеки MySqlConnection. Вибір саме цих засобів зумовлений їхньою доступністю, зручністю практичного застосування, можливістю швидкої реалізації настільного застосунку та забезпеченням надійного збереження даних.

Спроектовано структуру інформаційної системи та базу даних для зберігання відомостей про студентів. У процесі виконання роботи розроблено архітектуру застосунку з поділом на логічні модулі, що відповідають за інтерфейс, бізнес-логіку, моделі даних і доступ до сховища. Також було спроектовано структуру реляційної бази даних, яка включає таблиці для збереження студентів, дисциплін, користувачів та параметрів доступу. Для наочного представлення структури даних побудовано відповідні UML- та ER-діаграми.

Реалізовано програмний застосунок для ведення обліку студентів, дисциплін і користувачів системи. У третьому розділі описано практичну реалізацію інформаційної системи у вигляді настільного застосунку. Програма забезпечує додавання, редагування, перегляд і видалення записів про студентів, збереження навчальних даних, ведення дисциплін, а також управління користувачами системи. Окрім цього, реалізовано пошук, фільтрацію, сортування, імпорт і експорт даних, що забезпечує зручну роботу з інформацією.

Передбачено механізми розмежування доступу та основні засоби захисту даних. У системі реалізовано рольову модель доступу, відповідно до якої кожен користувач отримує лише ті права, які відповідають його функціональним

обов'язкам. Також передбачено авторизацію за логіном і паролем, блокування облікових записів, перевірку введених даних, підтримку цілісності бази даних та використання параметризованих SQL-запитів. Це дозволило підвищити рівень безпеки програмного продукту та зменшити ризик несанкціонованого доступу.

Виконано тестування та налагодження розробленого програмного забезпечення. У роботі проведено перевірку основних функціональних можливостей системи, зокрема авторизації, додавання, редагування та видалення студентів, роботи з дисциплінами, пошуку, фільтрації, доступу користувачів відповідно до ролей, а також взаємодії з базою даних. Під час тестування виявлено та усунуто окремі технічні недоліки, що дало змогу забезпечити стабільну та коректну роботу програмного застосунку.

Таким чином, розроблена інформаційна система обліку студентів є працездатним програмним продуктом, який може бути використаний для автоматизації обліку студентських даних у закладах вищої освіти. Практичне значення роботи полягає у створенні зручного інструмента для централізованого збереження, редагування, пошуку та аналізу інформації про студентів, їх навчальні дисципліни та користувачів системи. Перспективами подальшого розвитку програмного продукту є розширення функціональності, створення веб-версії системи, впровадження журналу дій користувачів, удосконалення аналітичних можливостей та інтеграція з іншими інформаційними системами закладу освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. McGraw-Hill, 2020.
2. OECD. Digital Education Outlook 2023. OECD Publishing, 2023
3. UNESCO. Digital Transformation in Higher Education. 2022-2024. URL: <https://www.unesco.org/> (дата звернення: 24.04.2026).
4. Newman S. Building Microservices. 2nd ed. O'Reilly Media, 2021.
5. Kleppmann M. Designing Data-Intensive Applications. O'Reilly Media, URL: <https://surl.li/uhfxta>.
6. Microsoft. .NET Documentation. URL: <https://learn.microsoft.com/en-us/dotnet/> (дата звернення: 24.04.2026).
7. Microsoft. C# Documentation. URL: <https://learn.microsoft.com/en-us/dotnet/csharp/> (дата звернення: 24.04.2026).
8. Microsoft. Windows Forms Documentation. URL: <https://learn.microsoft.com/en-us/dotnet/desktop/winforms/> (дата звернення: 24.04.2026).
9. Microsoft. ADO.NET Overview. URL: <https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/> (дата звернення: 24.04.2026).
10. Microsoft. Entity Framework Core Documentation. URL: <https://learn.microsoft.com/en-us/ef/core/> (дата звернення: 24.04.2026).
11. MySqlConnection. Official Documentation. URL: <https://mysqlconnector.net/> (дата звернення: 24.04.2026).
12. ISO. ISO/IEC 27001:2022 Information Security Management. URL: <https://www.iso.org/standard/27001> (дата звернення: 24.04.2026).