МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

ВІДДІЛЕННЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП’ЮТЕРНІ НАУКИ»

ЦИКЛОВА КОМІСІЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи освітнього ступеня «фаховий молодший бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
на тему:

«Система управління студентськими даними коледжу»

Виконав здобувач освіти групи П-41
Дзьома Максим Артемович

Керівник роботи:
Устименко Ярослав Іванович

Рецензент:
Устименко Людмила Миколаївна ч

Житомир – 2025

Зміст

РЕФЕРАТ

Дипломна робота на тему «Система управління студентськими даними коледжу» присвячена розробці програмного забезпечення для автоматизації обліку студентів, викладачів, груп, дисциплін та оцінок у навчальних закладах. Обсяг роботи становить 65 сторінок, включає 34 рисунки, 3 таблиці, 1 додаток і 20 літературних джерел.

Метою роботи є створення настільного додатка, який забезпечує зручне управління даними, швидкий пошук і фільтрацію, а також перевірку унікальності записів. Проведено аналіз аналогічних систем (Moodle, Blackboard, OpenSIS, Fedena), обґрунтовано вибір технологій: .NET Framework, Windows Forms, SQLite, C#, Visual Studio Community.

Розроблено базу даних із п’ятьма таблицями, пов’язаними зовнішніми ключами, та модульну структуру класів (моделі, репозиторії, форми). Реалізовано алгоритми сортування (QuickSort), фільтрації та CRUD-операцій із часовою складністю O(n) або O(n log n). Інтерфейс є україномовним, інтуїтивно зрозумілим, із підтримкою таблиць, списків і пошуку в реальному часі.

Система відповідає вимогам швидкодії, простоти розгортання та низьких апаратних вимог (Windows 7+, 2 ГБ ОЗП). Інструкція користувача забезпечує легке освоєння. Результати підтверджують практичну цінність системи для коледжів, із можливістю подальшого розширення функціоналу.

Ключові слова: інформаційна система, база даних, .NET, Windows Forms, SQLite, управління даними, коледж.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД — система управління базами даних.

SQLite — легка реляційна база даних, яка використовується в системі для локального зберігання даних.

.NET — платформа розробки від Microsoft для створення додатків.

C# — об’єктно-орієнтована мова програмування, використана для розробки системи.

Windows Forms — технологія для створення графічного інтерфейсу настільних додатків.

CRUD — операції з даними: Create (створення), Read (читання), Update (оновлення), Delete (видалення).

LINQ — Language Integrated Query, технологія для роботи з даними в C#.

GUI — Graphical User Interface, графічний інтерфейс користувача.

ПІБ — прізвище, ім’я, по батькові.

SQL — Structured Query Language, мова структурованих запитів для роботи з базами даних.

POCO — Plain Old CLR Object, прості об’єкти для представлення даних у .NET.

ER-діаграма — діаграма «сутність-зв’язок» для моделювання бази даних.

IDE — Integrated Development Environment, інтегроване середовище розробки (наприклад, Visual Studio).

DBNull — спеціальне значення в .NET для позначення відсутності даних у базі.

QuickSort — алгоритм швидкого сортування, використаний для впорядкування даних.

API — Application Programming Interface, інтерфейс програмування додатків.

UI — User Interface, інтерфейс користувача.

ВСТУП

Сучасні освітні заклади, зокрема коледжі, стикаються з необхідністю ефективного управління великими обсягами інформації, пов’язаної з навчальним процесом. Дані про студентів, викладачів, групи, дисципліни та оцінки потребують систематизації, швидкого доступу та надійного зберігання. Ручне ведення таких даних є трудомістким і схильним до помилок, що знижує ефективність роботи адміністрації та викладацького складу. У зв’язку з цим актуальним є створення автоматизованих систем управління, які дозволяють оптимізувати процеси збору, обробки та аналізу інформації.

Тема дипломної роботи — «Система управління студентськими даними коледжу» — спрямована на розробку програмного забезпечення, яке забезпечує комплексне управління даними навчального закладу. Метою роботи є створення зручного та функціонального інструменту для автоматизації адміністративних і академічних процесів, що включають облік студентів, викладачів, груп, дисциплін і оцінок. Така система має спростити взаємодію між учасниками навчального процесу, підвищити точність даних і зменшити час, витрачений на рутинні операції.

Актуальність роботи зумовлена зростанням вимог до цифровізації освітніх процесів, необхідністю забезпечення прозорості та доступності інформації, а також прагненням підвищити якість управління в коледжах. Впровадження подібних систем сприяє не лише оптимізації роботи адміністрації, але й покращенню організації навчального процесу, що є важливим для забезпечення високого рівня освіти.

Об’єктом дослідження є процеси управління студентськими даними в коледжі, а предметом — програмне забезпечення для їх автоматизації. У рамках дипломної роботи передбачається аналіз вимог до системи, проектування її архітектури, розробка функціоналу з використанням сучасних технологій (.NET , SQLite), тестування та оцінка ефективності.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

1.1. Постановка основної задачі розробки

Розробка системи управління студентськими даними коледжу є відповіддю на потребу сучасних навчальних закладів у автоматизації процесів управління інформацією. Основна задача розробки полягає у створенні програмного забезпечення, яке забезпечить централізоване зберігання, обробку та швидкий доступ до даних про студентів, викладачів, навчальні групи, дисципліни та оцінки. Система має надавати зручний інтерфейс для виконання адміністративних і академічних функцій, таких як додавання, редагування, видалення та пошук інформації, а також забезпечувати цілісність і безпеку даних.

Для чіткого визначення задачі розробки необхідно сформулювати ключові вимоги до системи:

1. **Функціональні вимоги**:
   - Можливість ведення обліку студентів (ПІБ, дата народження, телефон, група).
   - Управління даними викладачів (ПІБ, унікальний ідентифікатор).
   - Організація навчальних груп (назва групи, унікальний ідентифікатор).
   - Ведення переліку дисциплін із прив’язкою до викладачів.
   - Реєстрація оцінок студентів за дисциплінами з урахуванням дати виставлення.
   - Фільтрація та пошук даних за різними критеріями (наприклад, ПІБ студента, назва дисципліни).
   - Перевірка унікальності даних (наприклад, ПІБ студента в межах однієї групи).
2. **Нефункціональні вимоги**:
   - Зручний і інтуїтивно зрозумілий інтерфейс користувача, адаптований для адміністраторів і викладачів.
   - Висока швидкодія при обробці запитів до бази даних.

– Надійність і захист від втрати даних.
– Простота розгортання та використання на стандартних робочих станціях коледжу.
– Можливість масштабування системи для додавання нових функцій у майбутньому.

3. **Технічні обмеження**:

– Система має бути реалізована як настільний додаток для операційної системи Windows, що є стандартним середовищем для більшості навчальних закладів.
– Використання легкої бази даних (наприклад, SQLite) для спрощення розгортання без необхідності встановлення серверного програмного забезпечення.
– Розробка з використанням сучасних технологій, які забезпечують стабільність і підтримку.

Основна задача розробки передбачає створення системи, яка інтегрує всі перелічені функції в єдину платформу. Програмне забезпечення має забезпечити автоматизацію рутинних процесів, таких як введення оцінок, формування списків студентів чи груп, а також зменшити ймовірність помилок, пов’язаних із ручним веденням даних. Важливим аспектом є забезпечення гнучкості системи, щоб вона могла адаптуватися до потреб конкретного коледжу, наприклад, шляхом додавання нових полів даних або звітів.

Для реалізації задачі необхідно провести аналіз доступних технологій розробки, включаючи платформи програмування, системи управління базами даних і інструменти створення інтерфейсу користувача. На основі цього аналізу буде обрано оптимальний технологічний стек, який забезпечить ефективну реалізацію системи з урахуванням її функціональних і нефункціональних вимог.

1.2. Огляд та порівняння аналогічних систем.

Для розробки ефективної системи управління студентськими даними коледжу важливо проаналізувати існуючі аналогічні рішення, які використовуються в освітніх закладах. Такі системи можуть бути як комерційними, так і з відкритим кодом, і надають різний набір функціональних можливостей. У цьому підрозділі розглянуто кілька популярних систем, їхні переваги, недоліки та порівняння з вимогами до розроблюваної системи.

Moodle — це платформа з відкритим кодом для дистанційного навчання, яка часто використовується в коледжах і університетах для управління навчальними матеріалами, оцінками та даними студентів.

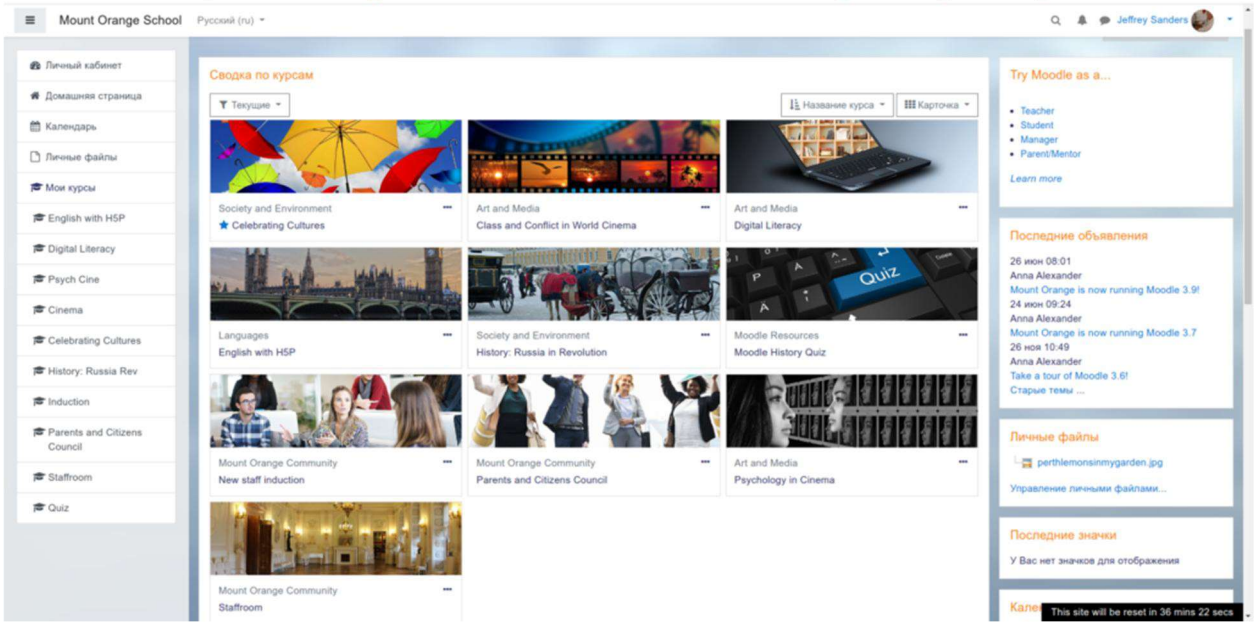


Рисунок 1.2.1 – Moodle

Основні можливості:

- Управління курсами та дисциплінами.
- Ведення оцінок і журналу відвідувань.
- Інтеграція з даними студентів і викладачів.
- Інструменти для створення звітів і аналітики.

Переваги:

- Гнучкість і можливість налаштування.

– Велика спільнота розробників і широкий вибір плагінів.
– Підтримка багатомовного інтерфейсу, зокрема української мови.

Недоліки:

– Вимагає серверного розгортання, що може бути складним для невеликих коледжів.
– Надмірна складність для простих адміністративних задач, таких як облік студентів чи груп.
– Основний акцент на дистанційному навчанні, а не на управлінні даними коледжу.

Moodle частково відповідає вимогам розроблюваної системи, але є надто громіздкою для локального використання в невеликих закладах і не оптимальна для базового управління даними.

Blackboard — комерційна система управління навчанням, популярна в англомовних країнах, яка підтримує облік студентів, оцінки та навчальні ресурси.

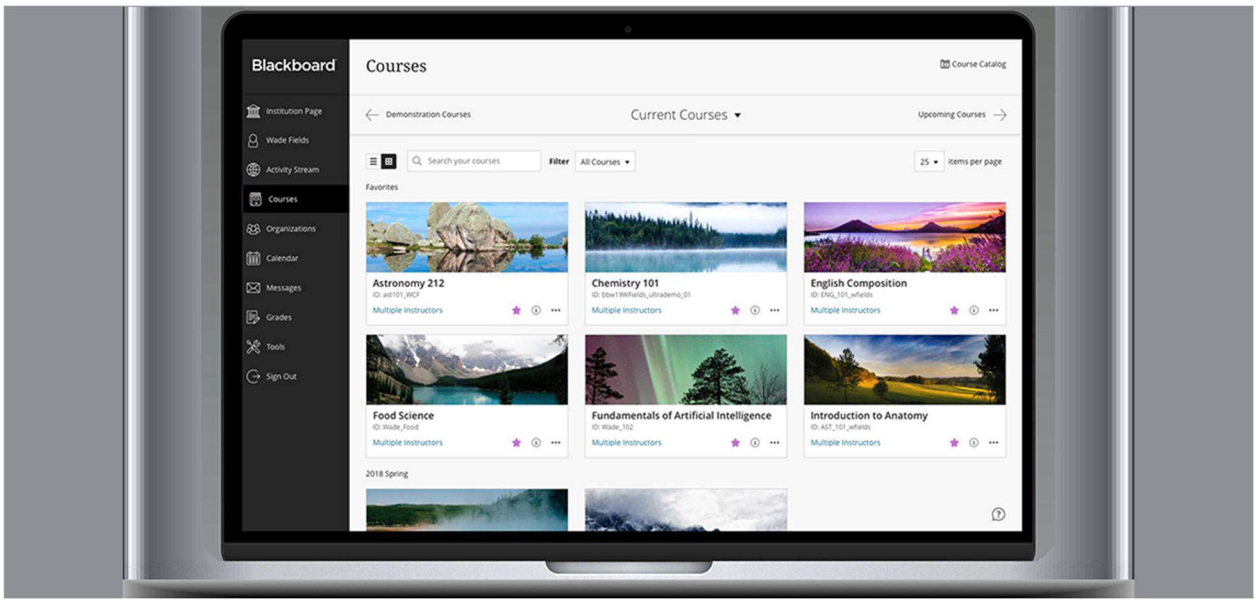


Рисунок 1.2.2 – Blackboard

Основні можливості:

– Управління профілями студентів і викладачів.
– Ведення академічних журналів і оцінок.
– Інтеграція з розкладами та аналітичними звітами.

Переваги:

– Високий рівень інтеграції з іншими системами (наприклад, фінансовими модулями).

– Хмарна інфраструктура, що зменшує потребу в локальному обслуговуванні.

Недоліки:

– Висока вартість ліцензії, що може бути непосильною для невеликих коледжів.

– Орієнтація на англомовний ринок, що ускладнює локалізацію для України.

Blackboard пропонує широкий функціонал, але через високу ціну та складність локалізації не є оптимальним вибором для розроблюваної системи.

OpenSIS — це система з відкритим кодом для управління інформацією про студентів, яка підтримує облік студентів, викладачів, оцінок і розкладів.

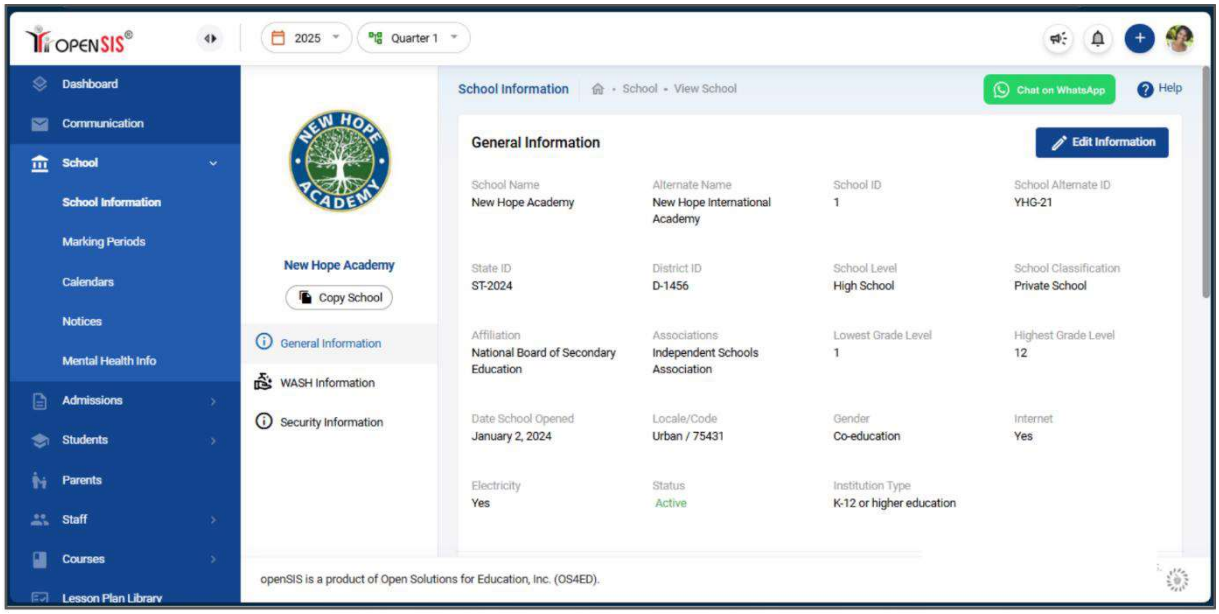


Рисунок 1.2.3 – OpenSIS

Основні можливості:

– Управління даними студентів, груп і викладачів.

– Ведення оцінок і відвідувань.

– Генерація звітів (наприклад, академічної успішності).

Переваги:

– Безкоштовність і можливість модифікації коду.

– Простота розгортання порівняно з Moodle.

Недоліки:

– Обмежений сучасний інтерфейс користувача, що може бути незручним.

– Відсутність повноцінної підтримки української локалізації.

– Менш активна спільнота порівняно з Moodle, що ускладнює підтримку.

OpenSIS ближче до потреб розроблюваної системи, але потребує доопрацювання інтерфейсу та локалізації для українських коледжів.

Fedena — це комерційна система управління навчальними закладами з відкритим кодом у базовій версії, яка підтримує управління студентами, оцінками та розкладами.

Рисунок 1.2.4 – Fedena

Основні можливості:

– Облік студентів, викладачів і груп.

– Управління оцінками та дисциплінами.

– Модуль звітності та аналітики.

Переваги:

- Гнучка модульна структура.
- Підтримка локалізації для різних країн.

Недоліки:

- Платні модулі для розширеного функціоналу.
- Вимагає серверного розгортання, що ускладнює використання в невеликих закладах.
- Обмежена підтримка настільних додатків.

Fedena частково відповідає вимогам, але її серверна орієнтація та платні модулі роблять її менш придатною порівняно з локальним настільним додатком.

Таблиця 1.2.1.
Порівняльна таблиця аналогічних систем

Система	Тип	Локалізація	Настільний додаток	Вартість	Простота розгортання	Зручність інтерфейсу
Moodle	Відкритий код	Часткова	Ні	Безкоштовно	Складно	Середня
Blackboard	Комерційна	Обмежена	Ні	Висока	Просто (хмара)	Висока
OpenSIS	Відкритий код	Обмежена	Так	Безкоштовно	Середньо	Низька
Fedena	Комерційна/ Відкритий код	Часткова	Ні	Безкоштовно/ Платно	Складно	Середня

Аналіз аналогічних систем показує, що більшість із них орієнтовані на веб-платформи або дистанційне навчання, що не завжди відповідає потребам невеликих коледжів, де перевага надається локальним настільним додаткам. Жодна з розглянутих систем не забезпечує повної локалізації для України, простоти розгортання та одночасно зручного інтерфейсу для базового управління даними. Розроблювана система має заповнити цю прогалину, пропонуючи настільний додаток із підтримкою української мови, легкою базою даних (SQLite) і мінімальними вимогами до апаратного забезпечення. Основними відмінностями стануть простота використання, локальна установка без серверів і адаптація до типових адміністративних процесів українських коледжів.**Вибір та обґрунтування технологій розробки**

Для створення системи управління студентськими даними коледжу необхідно обрати технологічний стек, який забезпечить реалізацію всіх функціональних і нефункціональних вимог, буде простим у розгортанні та відповідатиме потребам невеликих навчальних закладів. На основі аналізу вимог і порівняння аналогічних систем було обрано набір технологій, які включають платформу програмування, систему управління базами даних і інструменти для створення інтерфейсу користувача. У цьому підрозділі обґрунтовується вибір технологій та їх відповідність поставленим задачам.

.NET— це платформа розробки від Microsoft, яка підтримує створення надійних настільних додатків із зручним інтерфейсом і широкими можливостями інтеграції. .NET є перевіреною платформою з багаторічною історією використання в корпоративних і освітніх додатках. Оскільки система призначена для настільних комп’ютерів у коледжах, де переважно використовується Windows, .NET забезпечує оптимальну сумісність. C# є основною мовою для .NET, яка поєднує простоту синтаксису, об’єктно-орієнтований підхід і потужні бібліотеки для роботи з даними та інтерфейсом.

Безкоштовне середовище Visual Studio Community надає зручні засоби для кодування, налагодження та дизайну інтерфейсу.

Розглядалися Java (JavaFX) і Python (з бібліотеками Tkinter або PyQt). Java була відхилена через більшу складність розгортання настільних додатків, а Python — через нижчу продуктивність графічного інтерфейсу порівняно з .NET.

Windows Forms — це фреймворк у складі .NET для створення настільних додатків із графічним інтерфейсом. Windows Forms дозволяє швидко створювати інтерфейси за допомогою візуального дизайнера у Visual Studio, що скорочує час розробки. Для системи управління даними достатньо базових елементів керування (таблиці, текстові поля, кнопки, списки), які Windows Forms надає в повному обсязі. Підтримка україномовного інтерфейсу легко реалізується через налаштування текстових міток і повідомлень. Windows

Forms забезпечує швидке відображення інтерфейсу навіть на застарілих комп’ютерах, що актуально для коледжів із обмеженим бюджетом.

WPF (Windows Presentation Foundation) і UWP (Universal Windows Platform) пропонують сучасніший вигляд, але потребують більше ресурсів і мають крутішу криву навчання. Windows Forms обрано через простоту та достатність для задачі.

SQLite — це легка вбудована реляційна база даних, яка не вимагає окремого серверного програмного забезпечення та зберігає дані у файлі. SQLite не потребує встановлення серверів, що ідеально для локального настільного додатка в коледжі. База даних зберігається в одному файлі, що полегшує резервне копіювання та перенесення. Для невеликих обсягів даних (кілька тисяч записів про студентів, викладачів, оцінки) SQLite забезпечує швидкий доступ і обробку. Бібліотека System.Data.SQLite дозволяє легко інтегрувати SQLite із C# і .NET.

Розглядалися Microsoft SQL Server і MySQL. SQL Server відхилено через необхідність серверного розгортання та ліцензійні витрати, а MySQL — через складність налаштування для локального використання. SQLite є оптимальним вибором для настільного додатка з локальною базою.

Visual Studio Community — це безкоштовне інтегроване середовище розробки (IDE) від Microsoft для створення додатків на .NET. Інтеграція з .NET: Visual Studio забезпечує повну підтримку C#, Windows Forms і бібліотек для роботи з SQLite. Дозволяє швидко створювати інтерфейси шляхом перетягування елементів. Вбудовані засоби для тестування та пошуку помилок прискорюють розробку. Community-версія є доступною для студентів і невеликих проєктів, що відповідає умовам дипломної роботи.

Visual Studio Code із розширеннями для C# менш зручний для Windows Forms, а інші IDE (наприклад, JetBrains Rider) є платними.

Таблиця 1.3.1.
Порівняння обраних технологій із вимогами

Вимога	.NET	Windows Forms	SQLite	Visual Studio Community
Настільний додаток	+	+	+	+
Простота розгортання	+	+	+	+
Зручний інтерфейс	+	+	Не застосовується	+
Локалізація (українська)	+	+	+	+
Продуктивність	+	+	+	+
Безкоштовність	+	+	+	+
Масштабованість	+	+	Обмежена	+

Обраний технологічний стек (.NET, Windows Forms, SQLite, Visual Studio Community) є оптимальним для реалізації системи управління студентськими даними коледжу. .NET і C# забезпечують надійність і продуктивність, Windows Forms — швидке створення зручного інтерфейсу, SQLite — просте й ефективне зберігання даних, а Visual Studio — зручне середовище для розробки. Ці технології відповідають вимогам локального настільного додатка, не потребують значних ресурсів і дозволяють створити систему, адаптовану до потреб українських коледжів, із можливістю подальшого розширення функціоналу.

Висновки до розділу 1

У першому розділі дипломної роботи проведено аналіз основних аспектів розробки системи управління студентськими даними коледжу, що дозволило сформувати чітке уявлення про задачу та підібрати оптимальні технології для її реалізації.

Постановка задачі розробки показала, що система має забезпечувати автоматизацію ключових адміністративних і академічних процесів, таких як облік студентів, викладачів, груп, дисциплін та оцінок. Визначено функціональні вимоги (управління даними, пошук, фільтрація, перевірка унікальності) та нефункціональні вимоги (зручність інтерфейсу, швидкодія, простота розгортання), які є основою для подальшої розробки.

Огляд аналогічних систем (Moodle, Blackboard, OpenSIS, Fedena) виявив, що більшість із них орієнтовані на веб-платформи або дистанційне навчання,

що не повною мірою відповідає потребам невеликих коледжів, де перевага надається локальним настільним додаткам. Жодна з розглянутих систем не забезпечує оптимального поєднання простоти розгортання, україномовного інтерфейсу та адаптації до локальних адміністративних процесів, що підкреслює актуальність розробки нової системи.

Вибір технологічного стека (.NET, Windows Forms, SQLite, Visual Studio Community) обґрунтовано їхньою відповідністю поставленим вимогам. .NET забезпечує надійність і продуктивність, Windows Forms дозволяє швидко створити зручний інтерфейс, SQLite гарантує просте зберігання даних без необхідності серверного розгортання, а Visual Studio Community пропонує потужне середовище для розробки. Ці технології є оптимальними для створення локального настільного додатка, який буде доступним і зручним для використання в українських коледжах.

Проведений аналіз закладає міцну основу для проектування та реалізації системи, яка буде ефективно вирішувати задачі управління студентськими даними, враховуючи специфіку освітнього середовища та обмеження ресурсів навчальних закладів.

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА

2.1. Вибір алгоритмів та їх ефективність

На етапі проектування системи управління студентськими даними коледжу важливим є вибір оптимальних алгоритмів для реалізації основних функцій системи, таких як обробка даних, пошук, фільтрація, сортування та перевірка унікальності. Алгоритми мають бути ефективними з точки зору швидкості виконання та використання ресурсів, враховуючи, що система розробляється як настільний додаток із локальною базою даних SQLite. У цьому підрозділі розглянуто основні алгоритми, які використовуються в системі, їхній вибір та оцінка ефективності.

1. **Завантаження даних із бази даних**

Система завантажує списки студентів, викладачів, груп, дисциплін і оцінок із бази даних для відображення у формах (StudentForm, TeacherForm, GroupForm, DisciplineForm, GradeForm).

Алгоритм:

- Використовується прямий SQL-запит (SELECT * FROM TableName) для отримання всіх записів із таблиці.
- Дані зчитуються за допомогою SQLiteDataReader і перетворюються на об’єкти відповідних класів (Student, Teacher, Group, Discipline, Grade) у пам’яті.

Часова складність O(n), де n — кількість записів у таблиці, оскільки кожен запис зчитується один раз. Просторова складність O(n), оскільки всі записи зберігаються в пам’яті як список об’єктів.

Для невеликих обсягів даних (до кількох тисяч записів, що типово для коледжу) цей підхід є ефективним, оскільки SQLite оптимізує запити до локальної бази даних, а завантаження всіх даних у пам’ять спрощує подальшу обробку. Альтернативи, такі як пагінація, ускладнили б інтерфейс і не дали значного виграшу в продуктивності для малих наборів даних.

2. **Сортування даних**

Для зручності користувача списки (наприклад, студентів за ПІБ, груп за назвою, дисциплін за назвою) відображаються в алфавітному порядку в ComboBox і DataGridView.

Алгоритм:

- Використовується LINQ-метод OrderBy для сортування списків об’єктів у пам’яті.
- LINQ під капотом використовує алгоритм швидкого сортування (QuickSort) для впорядкування даних.

Часова складність - O(n log n) у середньому для QuickSort, де n — кількість елементів у списку. Просторова складність - O(n), оскільки створюється новий відсортований список.

QuickSort є оптимальним вибором для сортування невеликих і середніх наборів даних у пам’яті. Для списків із кількома сотнями чи тисячами елементів (студенти, дисципліни) продуктивність залишається високою. Альтернативи, такі як сортування на рівні SQL-запиту (ORDER BY), були відхилені, щоб зменшити навантаження на базу даних і спростити код.

3. **Фільтрація даних**

Користувач може фільтрувати дані в реальному часі за введеним текстом (наприклад, ПІБ студента в StudentForm, назва дисципліни чи ПІБ студента в GradeForm).

Алгоритм:

- Використовується LINQ-метод Where для фільтрації списку об’єктів у пам’яті за умовою часткового збігу рядка (регістронезалежно).
- Фільтрація виконується при кожній зміні тексту в полі пошуку (TextChanged).

Часова складність - O(n), де n — кількість елементів у списку, оскільки кожен елемент перевіряється на відповідність умові. Просторова складність - O(m), де m — кількість елементів, що відповідають умові фільтрації (у гіршому випадку m = n).

Лінійний перебір є прийнятним для невеликих списків (до кількох тисяч записів), оскільки забезпечує простоту реалізації та швидке виконання на сучасних комп'ютерах. Для великих обсягів даних можна було б застосувати індексацію або алгоритми пошуку (наприклад, префіксне дерево), але для коледжу з обмеженим обсягом даних це надлишково.

4. **Перевірка унікальності**

Перед додаванням або оновленням студента система перевіряє, чи немає іншого студента з таким самим ПІБ у тій самій групі (StudentForm).

Алгоритм:

- Використовується LINQ-метод Any для перевірки списку студентів за умовою.
- Перевірка виконується в пам'яті після завантаження всіх студентів.

Часова складність - O(n), де n — кількість студентів, оскільки в гіршому випадку перевіряються всі записи. Просторова складність - O(1), оскільки додаткова пам'ять не виділяється (крім завантаженого списку).

Для невеликої кількості студентів (до кількох тисяч) лінійний пошук є достатньо швидким. Альтернативою могла б бути унікальність на рівні бази даних (унікальний індекс), але це ускладнило б логіку, оскільки унікальність залежить від комбінації полів (FullName і GroupId). Хеш-таблиці чи словники не використовуються, щоб уникнути складності коду для простої задачі.

Оновлення даних у DataGridView

Після фільтрації, сортування чи завантаження даних DataGridView оновлюється для відображення актуального списку (наприклад, студентів із назвою групи замість GroupId).

Алгоритм:

- Список об'єктів прив'язується до DataSource DataGridView.
- Для відображення пов'язаних даних (наприклад, назви групи) вручну заповнюються відповідні клітинки:

Часова складність - O(n * m), де n — кількість рядків у DataGridView, m — кількість груп (для пошуку FirstOrDefault). У типовому випадку m набагато

менше n, тому складність близька до O(n). Просторова складність - O(1), оскільки змінюються лише клітинки таблиці.

Лінійний перебір рядків є простим і достатньо швидким для відображення даних. Для прискорення можна було б використати словник для груп (Dictionary<int, Group>), але для малого обсягу даних виграш був би незначним.

Таблиця 2.1.1.
Порівняння алгоритмів із вимогами

Задача	Алгоритм	Часова складність	Просторова складність	Відповідність вимогам
Завантаження даних	SQL + зчитування	O(n)	O(n)	Висока
Сортування	QuickSort (LINQ)	O(n log n)	O(n)	Висока
Фільтрація	Лінійний пошук (LINQ)	O(n)	O(m)	Висока
Перевірка унікальності	Лінійний пошук (LINQ)	O(n)	O(1)	Висока
Оновлення DataGridView	Лінійний перебір	O(n)	O(1)	Висока

Обрані алгоритми (зчитування даних через SQL, сортування QuickSort, лінійна фільтрація та перевірка унікальності через LINQ, перебір для оновлення DataGridView) є оптимальними для системи з огляду на її масштаби та вимоги. Вони забезпечують швидке виконання для невеликих наборів даних (до кількох тисяч записів), простоту реалізації та підтримку в рамках .NET . Для великих обсягів даних можна було б застосувати складніші структури (хеш-таблиці, індекси), але для типового коледжу обрані алгоритми є ефективними та достатніми.

2.2. База даних

Для забезпечення ефективного зберігання та управління даними в системі управління студентськими даними коледжу розроблено базу даних на основі SQLite — легкої реляційної бази даних, яка ідеально підходить для локальних настільних додатків. У цьому підрозділі описано структуру таблиць бази даних, їхні поля, типи даних, а також зв'язки між таблицями, які забезпечують цілісність і логічну організацію даних.

Опис структури таблиць

База даних складається з п'яти основних таблиць: Groups, Teachers, Disciplines, Students і Grades. Кожна таблиця відповідає окремій сутності системи, а їхні поля відображають ключові атрибути, необхідні для виконання функціональних вимог. Нижче наведено детальний опис кожної таблиці.

Таблиця Groups (Групи) - зберігає інформацію про навчальні групи коледжу.

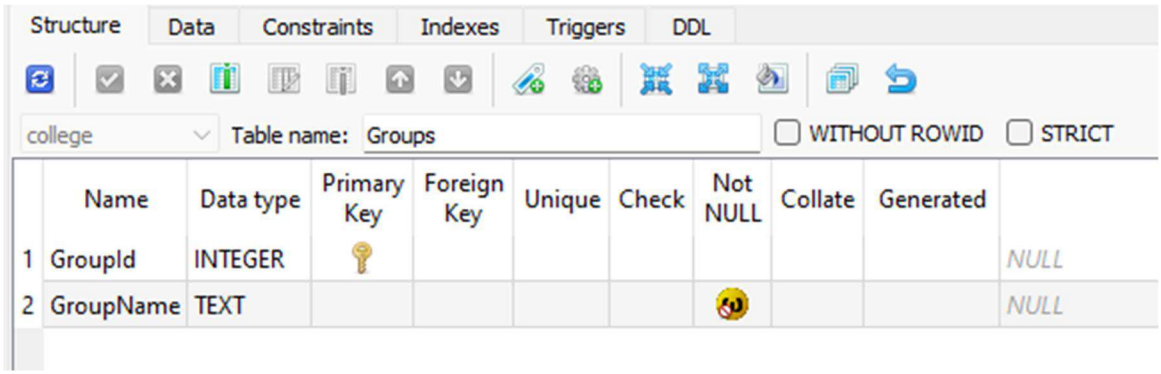


Рисунок 2.2.1 – Таблиця Groups

GroupId (INTEGER, PRIMARY KEY, AUTOINCREMENT) — унікальний ідентифікатор групи.

GroupName (TEXT, NOT NULL) — назва групи (наприклад, "ІПЗ-21").

Поле GroupId є первинним ключем, який автоматично збільшується для кожного нового запису. GroupName унікальне в межах таблиці, щоб уникнути дублювання груп.

Таблиця Teachers (Викладачі) - зберігає дані про викладачів коледжу.

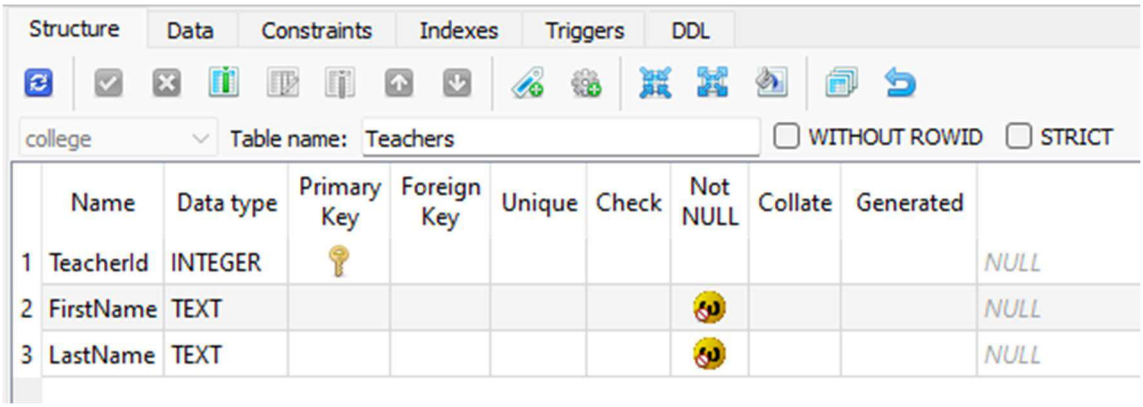


Рисунок 2.2.2 – Таблиця Teachers

TeacherId (INTEGER, PRIMARY KEY, AUTOINCREMENT) — унікальний ідентифікатор викладача.

FirstName (TEXT, NOT NULL) — ім’я викладача.

LastName (TEXT, NOT NULL) — прізвище викладача.

Поля FirstName і LastName використовуються для ідентифікації викладача. Унікальність забезпечується через TeacherId.

Таблиця Disciplines (Дисципліни) - зберігає інформацію про навчальні дисципліни та їхніх викладачів.

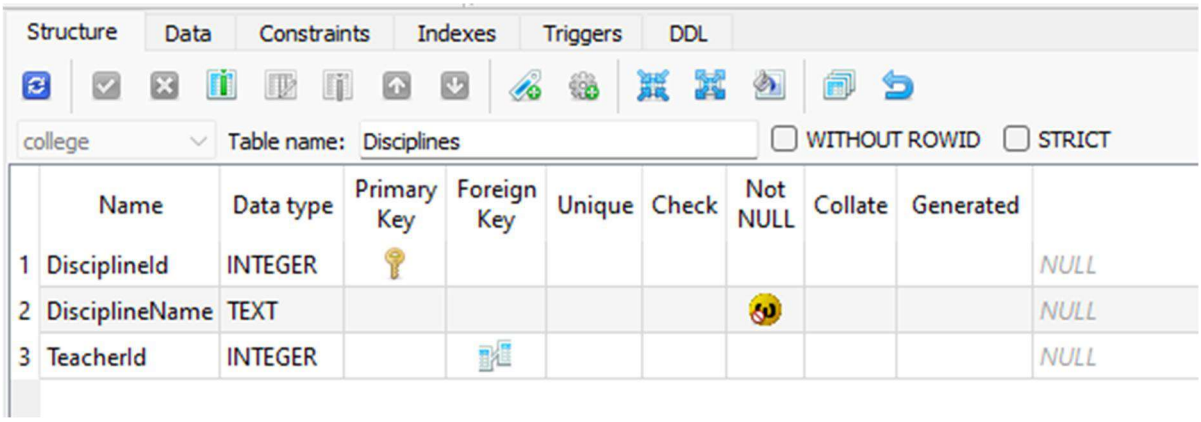


Рисунок 2.2.3 – Таблиця Disciplines

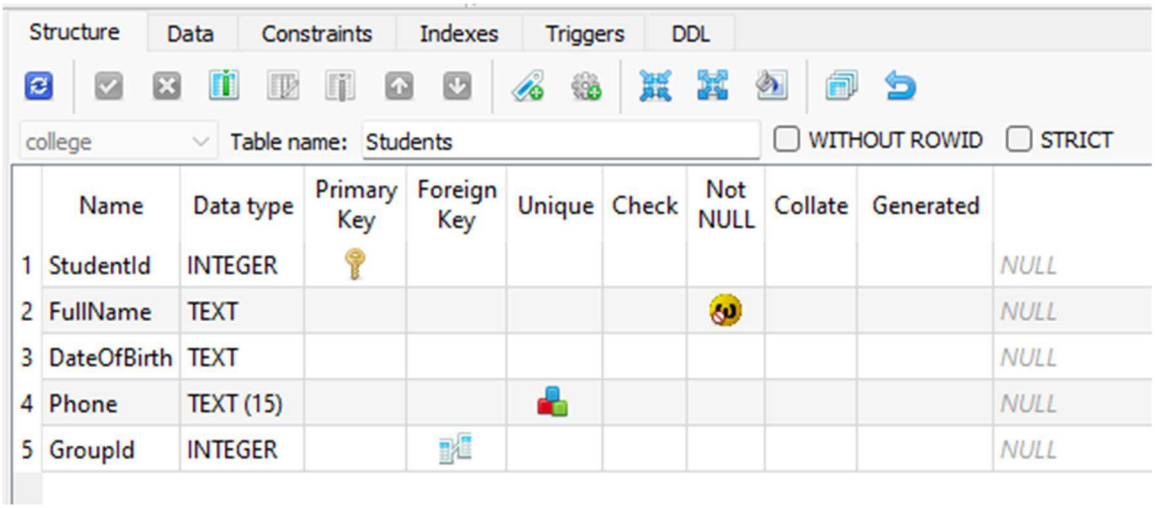
DisciplineId (INTEGER, PRIMARY KEY, AUTOINCREMENT) — унікальний ідентифікатор дисципліни.

DisciplineName (TEXT, NOT NULL) — назва дисципліни (наприклад, "Програмування").

TeacherId (INTEGER, NOT NULL, FOREIGN KEY) — ідентифікатор викладача, який веде дисципліну (посилання на Teachers.TeacherId).

DisciplineName унікальне в межах таблиці. Поле TeacherId є зовнішнім ключем, що забезпечує зв’язок із таблицею Teachers.

Таблиця Students (Студенти) - зберігає дані про студентів коледжу.



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	StudentId	INTEGER								NULL
2	FullName	TEXT								NULL
3	DateOfBirth	TEXT								NULL
4	Phone	TEXT (15)								NULL
5	GroupId	INTEGER								NULL

Рисунок 2.2.4 – Таблиця Students

StudentId (INTEGER, PRIMARY KEY, AUTOINCREMENT) — унікальний ідентифікатор студента.

FullName (TEXT, NOT NULL) — повне ім’я студента (ПІБ).

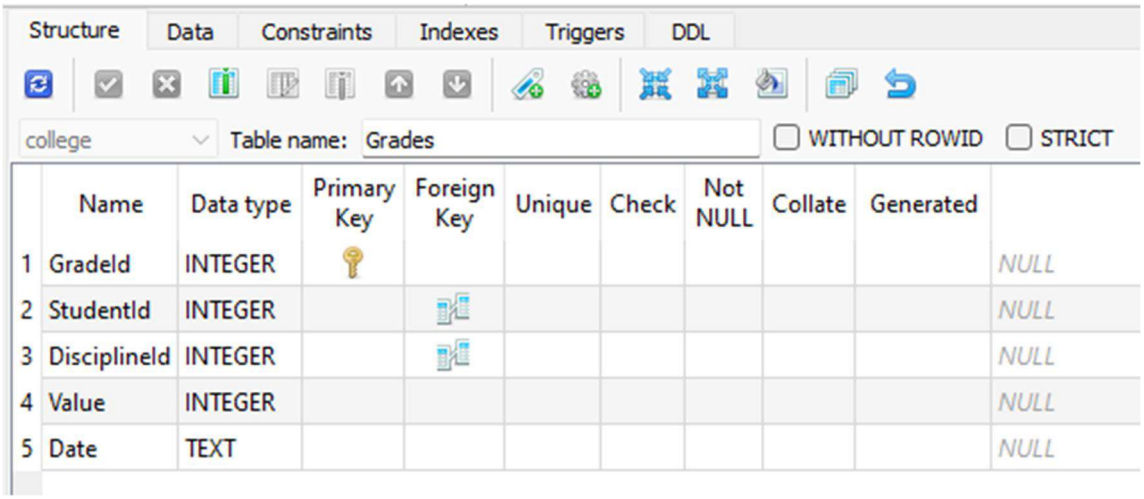
DateOfBirth (TEXT, NULL) — дата народження студента (зберігається як рядок у форматі "YYYY-MM-DD", може бути NULL).

Phone (TEXT, NULL) — номер телефону студента (може бути NULL).

GroupId (INTEGER, NOT NULL, FOREIGN KEY) — ідентифікатор групи, до якої належить студент (посилання на Groups.GroupId).

Поле FullName є обов’язковим, тоді як DateOfBirth і Phone опціональні. GroupId зв’язує студента з групою через зовнішній ключ.

Таблиця Grades (Оцінки) - зберігає оцінки студентів за дисципліни з інформацією про дату виставлення.



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	GradeId	INTEGER								NULL
2	StudentId	INTEGER								NULL
3	DisciplineId	INTEGER								NULL
4	Value	INTEGER								NULL
5	Date	TEXT								NULL

Рисунок 2.2.5 – Таблиця Grades

GradeId (INTEGER, PRIMARY KEY, AUTOINCREMENT) — унікальний ідентифікатор оцінки.

StudentId (INTEGER, NOT NULL, FOREIGN KEY) — ідентифікатор студента (посилання на Students.StudentId).

DisciplineId (INTEGER, NOT NULL, FOREIGN KEY) — ідентифікатор дисципліни (посилання на Disciplines.DisciplineId).

Value (INTEGER, NOT NULL) — значення оцінки (ціле число, наприклад, від 0 до 100).

Date (TEXT, NULL) — дата виставлення оцінки (рядок у форматі "YYYY-MM-DD", може бути NULL).

Поля StudentId і DisciplineId є зовнішніми ключами, що забезпечують зв’язок із таблицями Students і Disciplines. Value обов’язкове, тоді як Date може бути відсутнім.

Зв’язки між таблицями

Для забезпечення цілісності даних і логічної структури бази даних таблиці пов’язані через зовнішні ключі. Зв’язки відображають реальні взаємозв’язки між сутностями в системі. Нижче описано типи зв’язків і їхнє призначення:

1. **Groups ↔ Students (Один до багатьох)**

Одна група може містити багато студентів, але кожен студент належить лише до однієї групи. Поле Students.GroupId є зовнішнім ключем, що посилається на Groups.GroupId.

При видаленні групи необхідно або заборонити операцію, якщо в ній є студенти, або каскадно видалити пов’язаних студентів (у системі реалізовано заборону для уникнення втрати даних).

2. **Teachers ↔ Disciplines (Один до багатьох)**

Один викладач може вести кілька дисциплін, але кожна дисципліна прив’язана лише до одного викладача. Поле Disciplines.TeacherId є зовнішнім ключем, що посилається на Teachers.TeacherId. Видалення викладача забороняється, якщо до нього прив’язані дисципліни, щоб уникнути втрати зв’язків.

3. **Students ↔ Grades (Один до багатьох)**

Один студент може мати багато оцінок, але кожна оцінка належить лише одному студентові. Поле Grades.StudentId є зовнішнім ключем, що посилається на Students.StudentId. При видаленні студента пов’язані оцінки також видаляються (каскадне видалення), щоб підтримувати цілісність даних.

4. **Disciplines ↔ Grades (Один до багатьох)**

Одна дисципліна може мати багато оцінок, але кожна оцінка відноситься до однієї дисципліни. Поле Grades.DisciplineId є зовнішнім ключем, що посилається на Disciplines.DisciplineId. Видалення дисципліни забороняється, якщо до неї прив’язані оцінки, щоб зберегти історію оцінювання.

Для наочності структура бази даних представлена у вигляді ER-діаграми.

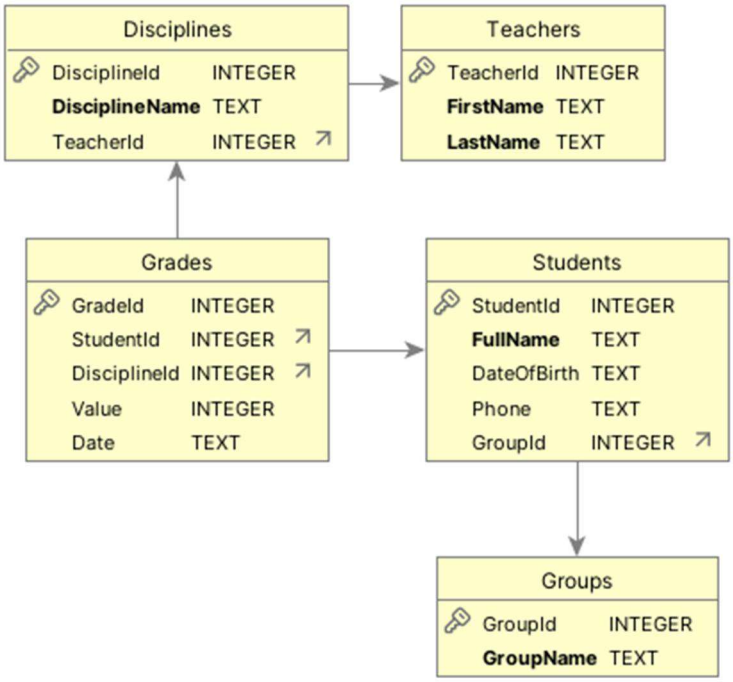


Рисунок 2.2.6 – ER-діаграма

Структура таблиць є мінімальною, але достатньою для виконання всіх функціональних вимог системи (облік студентів, викладачів, груп, дисциплін, оцінок).

Зовнішні ключі забезпечують логічні зв’язки між сутностями та запобігають некоректним даним (наприклад, оцінка не може посилатися на неіснуючого студента).

Опціональні поля (DateOfBirth, Phone, Date) дозволяють адаптувати систему до різних сценаріїв використання.

SQLite ефективно обробляє запити для невеликих баз даних (до кількох тисяч записів), а проста структура зменшує накладні витрати на індексацію та зв’язки.

Розроблена структура бази даних із таблицями Groups, Teachers, Disciplines, Students і Grades відповідає вимогам системи управління студентськими даними. Зв’язки типу "один до багатьох" забезпечують логічну організацію даних, а зовнішні ключі гарантують цілісність. Використання SQLite дозволяє створити компактну та легку у розгортанні базу даних, яка ідеально підходить для локального настільного додатка коледжу.

2.3. Спроєктовані класи програми

Для реалізації системи управління студентськими даними коледжу було спроєктовано набір класів, які відображають основні сутності системи, забезпечують взаємодію з базою даних і реалізують логіку інтерфейсу користувача. Класи поділено на три основні категорії: моделі даних, репозиторії для доступу до бази даних і форми для інтерфейсу користувача. У цьому підрозділі описано структуру класів, їх призначення, атрибути, методи та взаємозв'язки, з акцентом на відповідність функціональним вимогам системи.

1. Моделі даних

Моделі даних відповідають сутностям бази даних (Groups, Teachers, Disciplines, Students, Grades) і використовуються для представлення даних у програмі. Всі моделі розташовані в просторі CollegeManagementSystem.Models. Кожен клас містить властивості, що відображають поля відповідної таблиці, а також атрибути для відображення у формах.

Клас Group - представляє навчальну групу. Клас є простою моделлю для зберігання даних про групу. Використовується у формах для вибору групи (StudentForm, GradeForm) і відображення списків груп (GroupForm).

```
12 references
public class Group
{
    [DisplayName("Ідентифікатор групи")]
    6 references
    public int GroupId { get; set; }
    [DisplayName("Назва групи")]
    13 references
    public string? GroupName { get; set; }
}
```

Рисунок 2.3.1 – Клас Groups

Властивості:

- int GroupId { get; set; } — унікальний ідентифікатор групи (первинний ключ).
- string GroupName { get; set; } — назва групи.

Клас Teacher - представляє викладача. Використовується для управління викладачами (TeacherForm) і вибору викладача для дисципліни (DisciplineForm).

```
15 references
public class Teacher
{
    [DisplayName("Ідентифікатор викладача")]
    9 references
    public int TeacherId { get; set; }

    [DisplayName("Прізвище")]
    13 references
    public string LastName { get; set; }

    [DisplayName("Ім'я, по батькові")]
    13 references
    public string? FirstName { get; set; }

    [DisplayName("ПІБ")]
    1 reference
    public string FullName => $"{LastName} {FirstName}";

}
```

Рисунок 2.3.2 – Клас Teacher

Властивості:

- int TeacherId { get; set; } — унікальний ідентифікатор викладача.
- string FirstName { get; set; } — ім'я викладача.
- string LastName { get; set; } — прізвище викладача. Атрибути:

Клас Discipline - представляє навчальну дисципліну. Використовується для управління дисциплінами (DisciplineForm) і вибору дисципліни для оцінок (GradeForm).

```
16 references
public class Discipline
{
    [DisplayName("Ідентифікатор дисципліни")]
    11 references
    public int DisciplineId { get; set; }

    [DisplayName("Назва дисципліни")]
    17 references
    public string? DisciplineName { get; set; }

    [DisplayName("Ідентифікатор викладача")]
    8 references
    public int TeacherId { get; set; }

}
```

Рисунок 2.3.3 – Клас Discipline

Властивості:

- int DisciplineId { get; set; } — унікальний ідентифікатор дисципліни.
- string DisciplineName { get; set; } — назва дисципліни.
- int TeacherId { get; set; } — ідентифікатор викладача. Атрибути:

Клас Student - представляє студента. Основний клас для управління студентами (StudentForm) і вибору студентів для оцінок (GradeForm).

```
18 references
public class Student
{
    [DisplayName("Код студента")]
    11 references
    public int StudentId { get; set; }
    [DisplayName("ПІБ")]
    [Required(ErrorMessage = "ПІБ обов'язкове")]
    17 references
    public string FullName { get; set; } = string.Empty;
    [DisplayName("Дата народження")]
    7 references
    public DateTime? DateOfBirth { get; set; } = DateTime.Today;
    [DisplayName("Телефон")]
    7 references
    public string? Phone { get; set; } = string.Empty;
    [DisplayName("Ідентифікатор групи")]
    10 references
    public int GroupId { get; set; }
}
```

Рисунок 2.3.4 – Клас Student

Властивості:

- int StudentId { get; set; } — унікальний ідентифікатор студента.
- string FullName { get; set; } — повне ім’я студента.
- DateTime? DateOfBirth { get; set; } — дата народження (опціональна).
- string? Phone { get; set; } — номер телефону (опціональний).
- int GroupId { get; set; } — ідентифікатор групи.

Клас Grade - представляє оцінку студента за дисципліну. Використовується для управління оцінками (GradeForm), пов’язуючи студентів і дисципліни.

```
public class Grade
{
    [DisplayName("Код оцінки")]

    public int GradeId { get; set; }

    [DisplayName("Ідентифікатор студента")]
    10 references
    public int StudentId { get; set; }

    [DisplayName("Ідентифікатор дисципліни")]
    10 references
    public int DisciplineId { get; set; }

    [DisplayName("Оцінка")]
    8 references
    public int Value { get; set; }

    [DisplayName("Дата")]
    7 references
    public DateTime? Date { get; set; }
}
```

Рисунок 2.3.5 – Клас Grade

Властивості:

- int GradeId { get; set; } — унікальний ідентифікатор оцінки.
- int StudentId { get; set; } — ідентифікатор студента.
- int DisciplineId { get; set; } — ідентифікатор дисципліни.
- int Value { get; set; } — значення оцінки (0–100).
- DateTime? Date { get; set; } — дата виставлення оцінки (опціональна).

2. Репозиторії

Репозиторії відповідають за взаємодію з базою даних SQLite, забезпечуючи операції CRUD (Create, Read, Update, Delete) для кожної моделі.

Клас GroupRepository – управління даними груп у базі даних. Використовує SQL-запити через SQLiteConnection для роботи з таблицею Groups. Передає дані у вигляді об’єктів Group.

```
5 references
public class GroupRepository
{
    private readonly string _connectionString;
    2 references
    public GroupRepository(string connectionString)...
    3 references
    public List<Group> GetAll()...
    1 reference
    public void Add(Group group)...
    1 reference
    public void Update(Group group)...
    1 reference
    public void Delete(int groupId)...
}
```

Рисунок 2.3.6 – Клас GroupRepository

Методи:

List<Group> GetAll() — повертає всі групи.

void Add(Group group) — додає нову групу.

void Update(Group group) — оновлює існуючу групу.

void Delete(int groupId) — видаляє групу за ідентифікатором.

Клас TeacherRepository - управління даними викладачів. Працює з таблицею Teachers, забезпечуючи доступ до об’єктів Teacher.

```
5 references
public class TeacherRepository
{
    private readonly string _connectionString;
    2 references
    public TeacherRepository(string connectionString)...
    4 references
    public List<Teacher> GetAll()...
    1 reference
    public void Add(Teacher teacher)...
    1 reference
    public void Update(Teacher teacher)...
    1 reference
    public void Delete(int teacherId)...
}
```

Рисунок 2.3.7 – Клас TeacherRepository

Методи аналогічні GroupRepository (GetAll, Add, Update, Delete).

Клас DisciplineRepository - управління даними дисциплін. Працює з таблицею Disciplines, враховуючи зв'язок із TeacherId.

```
5 references
public class DisciplineRepository
{
    private readonly string _connectionString;
    2 references
    public DisciplineRepository(string connectionString)...
    4 references
    public List<Discipline> GetAll()...
    1 reference
    public void Add(Discipline discipline)...
    1 reference
    public void Update(Discipline discipline)...
    1 reference
    public void Delete(int disciplineId)...
}
```

Рисунок 2.3.8 – Клас DisciplineRepository

Методи аналогічні GroupRepository (GetAll, Add, Update, Delete).

Клас StudentRepository - управління даними студентів. Працює з таблицею Students, обробляючи nullable поля (DateOfBirth, Phone) через DBNull.

```
5 references
public class StudentRepository
{
    private readonly string _connectionString;
    2 references
    public StudentRepository(string connectionString)...
    4 references
    public List<Student> GetAll()...
    1 reference
    public void Add(Student student)...
    1 reference
    public void Update(Student student)...
    1 reference
    public void Delete(int studentId)...
}
```

Рисунок 2.3.9 – Клас StudentRepository

Методи аналогічні GroupRepository (GetAll, Add, Update, Delete).

Клас GradeRepository – управління оцінками. Працює з таблицею Grades, забезпечуючи зв'язки зі Students і Disciplines.

```
3 references
public class GradeRepository
{
    private readonly string _connectionString;
    1 reference
    public GradeRepository(string connectionString)...
    1 reference
    public List<Grade> GetAll()...
    1 reference
    public void Add(Grade grade)...
    1 reference
    public void Update(Grade grade)...
    1 reference
    public void Delete(int gradeId)...
}
```

Рисунок 2.3.10 – Клас GradeRepository

Методи аналогічні GroupRepository (GetAll, Add, Update, Delete).

3. Форми (інтерфейс користувача)

Форми реалізують графічний інтерфейс для взаємодії користувача з системою, використовуючи Windows Forms.

Клас MainForm. Головна форма для навігації до інших форм.

```
4 references
partial class MainForm
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;

    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...

    Windows Form Designer generated code

    private TableLayoutPanel tableLayoutPanel1;
    private Button btnExit;
    private Button btnOpenGroups;
    private Button btnOpenTeachers;
    private Button btnOpenDisciplines;
    private Button btnOpenStudents;
    private PictureBox pictureBox1;
    private Button btnOpenGrades;
}
```

Рисунок 2.3.11 – Клас MainForm

Кнопки для відкриття форм (btnOpenGroups, btnOpenTeachers, btnOpenDisciplines, btnOpenStudents, btnOpenGrades, btnExit). PictureBox для відображення логотипу. TableLayoutPanel для розташування елементів.

Обробники подій для кнопок (наприклад, btnOpenStudents_Click), що відкривають відповідні форми через ShowDialog. btnExit_Click для завершення програми. Опис: Забезпечує централізований доступ до всіх функцій системи.

Клас GroupForm. Управління групами (додавання, редагування, видалення, перегляд). Використовує GroupRepository для доступу до даних.

```
3 references
partial class GroupForm
{
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.IContainer components = null;

    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...

    Windows Form Designer generated code

    private DataGridView dataGridViewGroups;
    private Button btnAdd;
    private Button btnUpdate;
    private Button btnDelete;
    private Button btnClear;
    private TextBox txtGroupName;
    private Label label1;
    private TableLayoutPanel tableLayoutPanel1;
    private TextBox txtSearch;
    private Label label2;
}
```

Рисунок 2.3.12 – Клас GroupForm

DataGridView для списку груп. TextBox для введення назви групи. Кнопки (btnAdd, btnUpdate, btnDelete, btnClear). TextBox для пошуку за назвою. LoadGroups() — завантажує список груп. SettingGrid(DataGridView) — налаштовує DataGridView. Обробники для додавання, оновлення, видалення груп і очищення полів.

Клас TeacherForm. Управління викладачами. Використовує TeacherRepository

```
3 references
partial class TeacherForm
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;

    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...

    Windows Form Designer generated code

    private TableLayoutPanel tableLayoutPanel1;
    private DataGridView dataGridViewTeachers;
    private TextBox txtSearch;
    private Label label2;
    private Button btnAdd;
    private TextBox txtLastName;
    private Label lblLastName;
    private Button btnUpdate;
    private Button btnDelete;
    private Button btnClear;
    private TextBox txtFirstName;
    private Label lblFirstName;
}
```

Рисунок 2.3.13 – Клас TeacherForm

DataGridView для списку викладачів. TextBox для імені та прізвища. Кнопки та поле пошуку. Методи аналогічні GroupForm, адаптовані для Teacher.

Клас DisciplineForm. Управління дисциплінами. Використовує DisciplineRepository і TeacherRepository.

```
3 references
partial class DisciplineForm
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;
    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...
    Windows Form Designer generated code
    private TableLayoutPanel tableLayoutPanel1;
    private Label lblFirstName;
    private DataGridView dataGridViewDisciplines;
    private TextBox txtSearch;
    private Label label2;
    private Button btnAdd;
    private TextBox txtDisciplineName;
    private Label lblLastName;
    private Button btnUpdate;
    private Button btnDelete;
    private Button btnClear;
    private ComboBox comboBoxTeachers;
}
```

Рисунок 2.3.14 – Клас DisciplineForm

DataGridView для списку дисциплін. TextBox для назви дисципліни. ComboBox для вибору викладача. Кнопки та поле пошуку. Методи аналогічні GroupForm, із підтримкою вибору викладача.

Клас StudentForm. Управління студентами. Використовує StudentRepository і GroupRepository.

```
3 references
partial class StudentForm
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;

    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...

    Windows Form Designer generated code

    private TableLayoutPanel tableLayoutPanel1;
    private Label lblGroup;
    private DataGridView dataGridViewStudents;
    private TextBox txtSearch;
    private Label label2;
    private Button btnAdd;
    private TextBox txtFullName;
    private Label lblFullName;
    private Button btnUpdate;
    private Button btnDelete;
    private Button btnClear;
    private ComboBox comboBoxGroups;
    private Label lblDOB;
    private TextBox txtPhone;
    private Label lblPhone;
    private DateTimePicker dateTimePickerDOB;
}
```

Рисунок 2.3.15 – Клас StudentForm

DataGridView для списку студентів. TextBox для ПІБ і телефону. DateTimePicker для дати народження. ComboBox для вибору групи. Кнопки та поле пошуку. Методи LoadStudents(), LoadGroups() — завантаження даних. Перевірка унікальності ПІБ у групі. Обробники для CRUD-операцій.

Клас GradeForm. Управління оцінками. Використовує GradeRepository, StudentRepository, DisciplineRepository.

```
3 references
partial class GradeForm
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;

    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...

    Windows Form Designer generated code

    private TableLayoutPanel tableLayoutPanel1;
    private Label lblDate;
    private Label lblValue;
    private Label lblDiscipline;
    private DataGridView dataGridViewGrades;
    private TextBox txtSearch;
    private Label label2;
    private Button btnAdd;
    private Label lblFullName;
    private Button btnUpdate;
    private Button btnDelete;
    private Button btnClear;
    private ComboBox comboBoxDisciplines;
    private DateTimePicker dateTimePickerDate;
    private ComboBox comboBoxStudents;
    private NumericUpDown numericUpDownValue;
}
```

Рисунок 2.3.16 – Клас GradeForm

DataGridView для списку оцінок. ComboBox для вибору студента і дисципліни. NumericUpDown для оцінки (0–100). DateTimePicker для дати. Кнопки та поле пошуку. Методи: LoadGrades(), LoadStudents(), LoadDisciplines() — завантаження даних. Обробники для додавання, оновлення, видалення оцінок.

Взаємозв’язки класів

- **Моделі → Репозиторії**: Кожен клас моделі (Group, Teacher, тощо) використовується відповідним репозиторієм для передачі даних між базою і програмою.
- **Репозиторії → Форми**: Форми викликають методи репозиторіїв для отримання та маніпуляції даними. Наприклад, StudentForm використовує StudentRepository і GroupRepository.
- **Форми → MainForm**: MainForm є точкою входу і відкриває інші форми через кнопки.
- **Моделі ↔ Форми**: Властивості моделей прив’язуються до елементів керування форм (наприклад, Student.FullName до TextBox у StudentForm).

Спроєктовані класи (Group, Teacher, Discipline, Student, Grade, відповідні репозиторії та форми) забезпечують повну реалізацію системи управління студентськими даними. Моделі відображають структуру бази даних, репозиторії абстрагують доступ до даних, а форми надають зручний інтерфейс. Взаємозв’язки класів логічно відповідають зв’язкам у базі даних, що гарантує цілісність і ефективність системи.

2.4. Програмна реалізація

Програмна реалізація системи управління студентськими даними коледжу виконана з використанням обраного технологічного стека: .NET , Windows Forms, SQLite та Visual Studio Community. У цьому підрозділі описано процес розробки, ключові аспекти реалізації основних компонентів системи, приклади коду для основних функцій, а також пояснено, як програмне забезпечення відповідає функціональним і нефункціональним вимогам. Реалізація охоплює моделі даних, доступ до бази даних через репозиторії, графічний інтерфейс і логіку обробки даних.

Організація проєкту

Проєкт структуровано для забезпечення модульності та зрозумілості коду:

- **Простір імен CollegeManagementSystem.Models.** Містить класи моделей (Group, Teacher, Discipline, Student, Grade), які відображають сутності бази даних.
- **Простір імен CollegeManagementSystem.Data.Repositories**. Містить класи репозиторіїв (GroupRepository, TeacherRepository, тощо) для доступу до бази даних.
- **Простір імен CollegeManagementSystem.Forms**. Містить класи форм (MainForm, GroupForm, TeacherForm, DisciplineForm, StudentForm, GradeForm) для реалізації інтерфейсу користувача.
- **Файл бази даних**. SQLite-база college.db створюється автоматично при першому запуску програми.

Інтерфейс реалізовано через Windows Forms, із формами для кожної сутності. Реалізація вікна StudentForm:

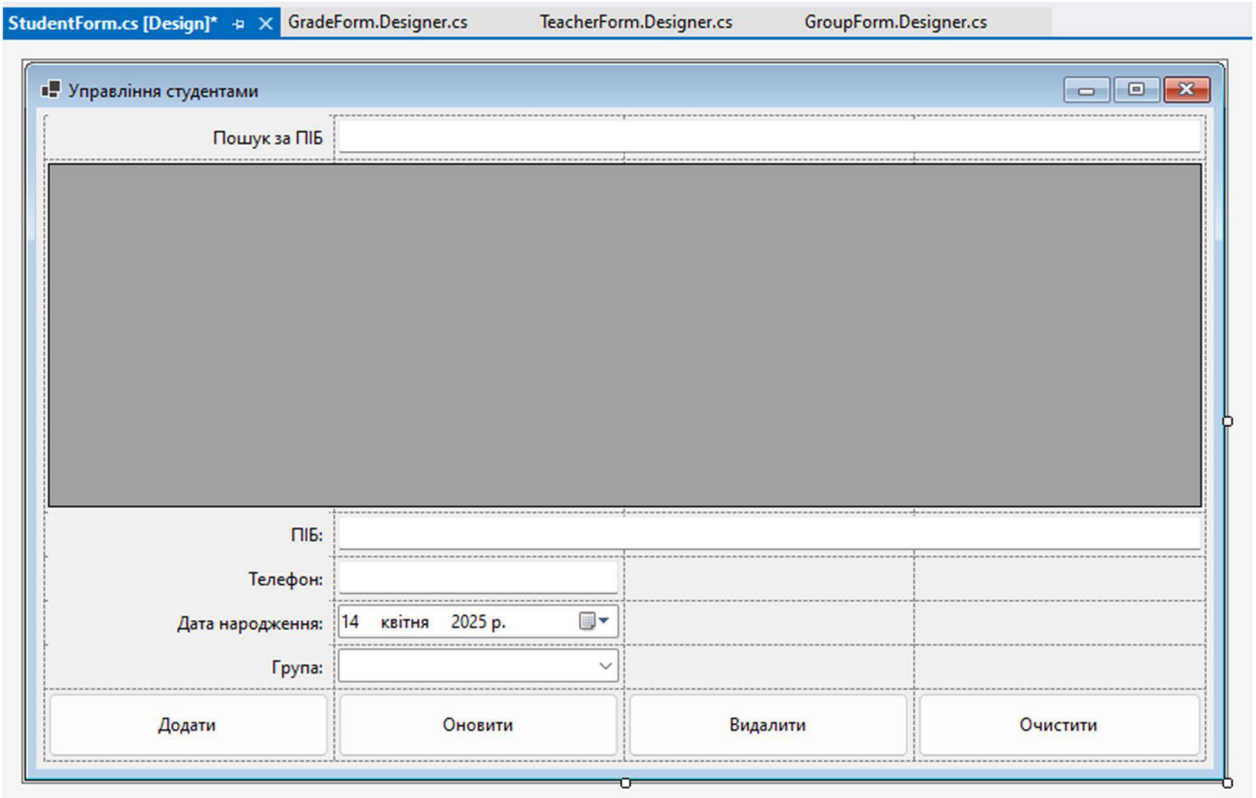


Рисунок 2.4.1 – Вікно класу StudentForm

DataGridView - Використовується для відображення списків із ручним налаштуванням колонок (SettingGrid).

ComboBox - Для вибору груп із відображенням назв (GroupName) і прив'язкою до ідентифікаторів (GroupId).

Виконується перед додаванням чи оновленням студента.

Усі операції в try-catch із виведенням україномовних повідомлень.

Метод ClearInputFields для скидання введених даних.

Аналогічно реалізовано GroupForm, TeacherForm, DisciplineForm, GradeForm.

Реалізація головної форми

MainForm є точкою входу, надаючи доступ до всіх функцій системи:

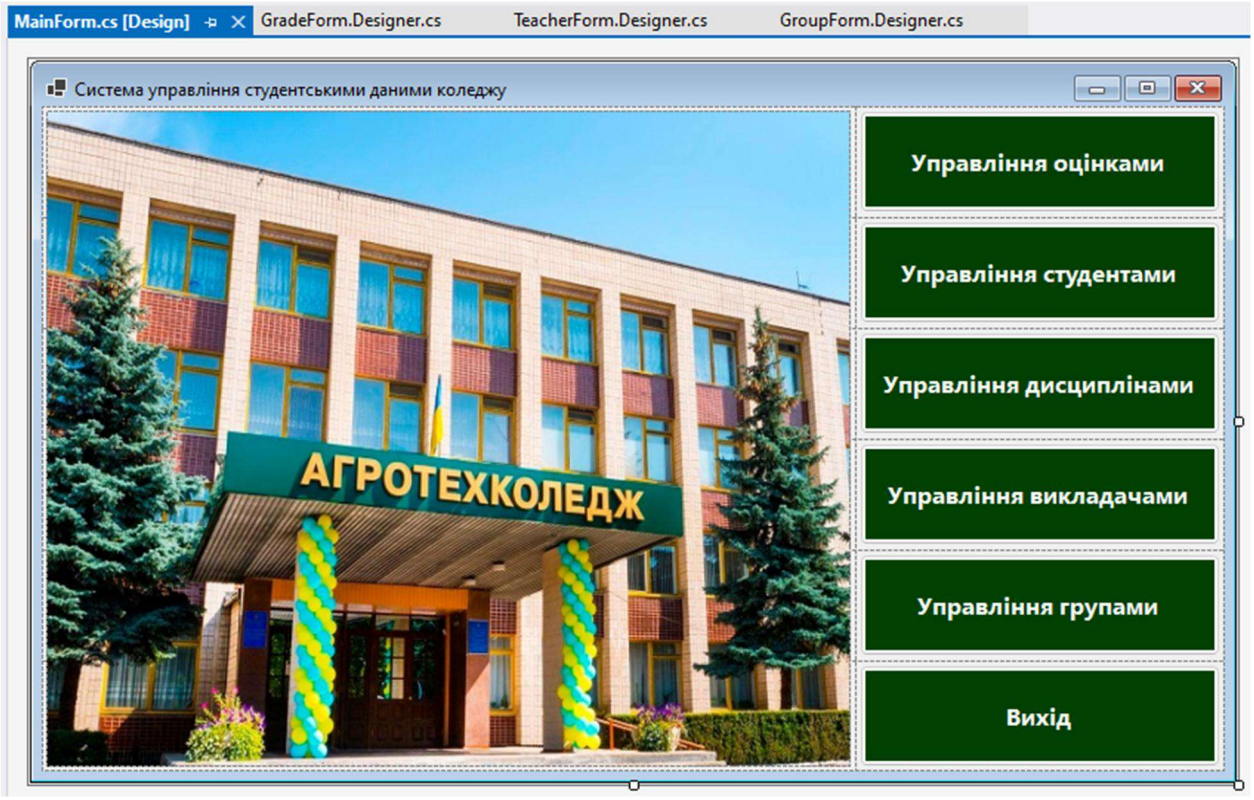


Рисунок 2.4.2 – Конструктор головного вікна

Використання TableLayoutPanel для розташування кнопок і PictureBox для логотипу. Модальний режим (ShowDialog) для відкриття форм. Підтвердження виходу для уникнення випадкового закриття.

Програмна реалізація системи управління студентськими даними коледжу успішно виконана з використанням .NET, Windows Forms і SQLite. Моделі даних, репозиторії та форми забезпечують повний функціонал для управління студентами, викладачами, групами, дисциплінами та оцінками. Код є модульним, зручним для підтримки та відповідає всім поставленим вимогам, включаючи україномовний інтерфейс, швидкодію та простоту розгортання. Реалізація готова до тестування та впровадження в реальних умовах коледжу.

Висновки до розділу 2

Другий розділ дипломної роботи присвячено проектуванню та розробці системи управління студентськими даними коледжу. Проведений аналіз і практична реалізація дозволили сформувати цілісне рішення, яке відповідає поставленим вимогам і забезпечує автоматизацію ключових процесів навчального закладу.

Вибір алгоритмів для обробки даних (завантаження, сортування, фільтрація, перевірка унікальності) ґрунтується на їхній ефективності для невеликих обсягів даних, типових для коледжу. Використання LINQ і QuickSort для сортування, лінійного пошуку для фільтрації та прямого доступу до SQLite забезпечує швидкодію та простоту реалізації, що відповідає вимогам продуктивності системи.

Структура бази даних, що включає таблиці Groups, Teachers, Disciplines, Students і Grades, спроєктована з урахуванням логічних зв’язків між сутностями. Зовнішні ключі та обмеження цілісності гарантують надійність даних, а використання SQLite забезпечує компактність і легкість розгортання. Зв’язки типу "один до багатьох" відображають реальні взаємозв’язки в системі, сприяючи зрозумілості та гнучкості бази даних.

Спроєктовані класи програми поділено на моделі (Group, Teacher, Discipline, Student, Grade), репозиторії (GroupRepository, StudentRepository,

тощо) та форми (MainForm, StudentForm, тощо). Така модульна структура забезпечує чіткий розподіл відповідальності, полегшує підтримку коду та дозволяє легко розширювати функціонал у майбутньому.

Програмна реалізація виконана з використанням .NET , Windows Forms і SQLite, що дозволило створити настільний додаток із зручним україномовним інтерфейсом. Реалізовані функції управління даними, пошуку, фільтрації та перевірки унікальності відповідають функціональним вимогам, а обробка винятків і параметризовані запити забезпечують надійність. Система є простою в розгортанні та адаптованою до потреб невеликих коледжів.

У розділі 2 створено повноцінну основу для системи управління студентськими даними, яка поєднує ефективні алгоритми, продуману базу даних, чітку організацію коду та зручний інтерфейс. Отримані результати підтверджують можливість практичного застосування системи в освітніх закладах і готовність до подальшого тестування та впровадження.

РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМОЮ

3.1. Мінімальні вимоги до системи

Для коректної роботи системи управління студентськими даними коледжу необхідно забезпечити відповідність апаратного та програмного забезпечення мінімальним вимогам. Система розроблена як настільний додаток для операційної системи Windows, що робить її доступною для більшості комп'ютерів, які використовуються в освітніх закладах.

Комп'ютер має бути оснащений процесором із частотою не нижче 1 ГГц, оперативною пам'яттю об'ємом від 2 ГБ і вільним місцем на диску щонайменше 100 МБ для зберігання програми та бази даних SQLite. Операційна система Windows 7 або новіша версія (Windows 8, 10, 11) забезпечить сумісність із додатком. Для стабільної роботи рекомендується встановити .NET, який зазвичай уже присутній у сучасних версіях Windows, але може потребувати окремого завантаження для старіших систем.

Наявність монітора з роздільною здатністю від 1024x768 пікселів гарантує комфортне відображення інтерфейсу. Додаткове серверне програмне забезпечення чи підключення до Інтернету не потрібні, оскільки база даних SQLite працює локально. Ці вимоги дозволяють використовувати систему навіть на застарілих комп'ютерах, що є типовим для багатьох коледжів із обмеженим бюджетом.

3.2. Інтерфейс користувача

Інтерфейс системи управління студентськими даними коледжу розроблено з використанням Windows Forms, що забезпечує простоту, інтуїтивність і зручність для користувачів із різним рівнем підготовки. Усі елементи інтерфейсу локалізовані українською мовою, а структура форм побудована таким чином, щоб мінімізувати час на освоєння програми. Система складається з головної форми та п'яти спеціалізованих форм для

управління групами, викладачами, дисциплінами, студентами та оцінками. Нижче наведено детальний опис інтерфейсу програми.

Головна форма (MainForm)

При запуску програми користувач потрапляє на головну форму, яка є центральною точкою навігації.

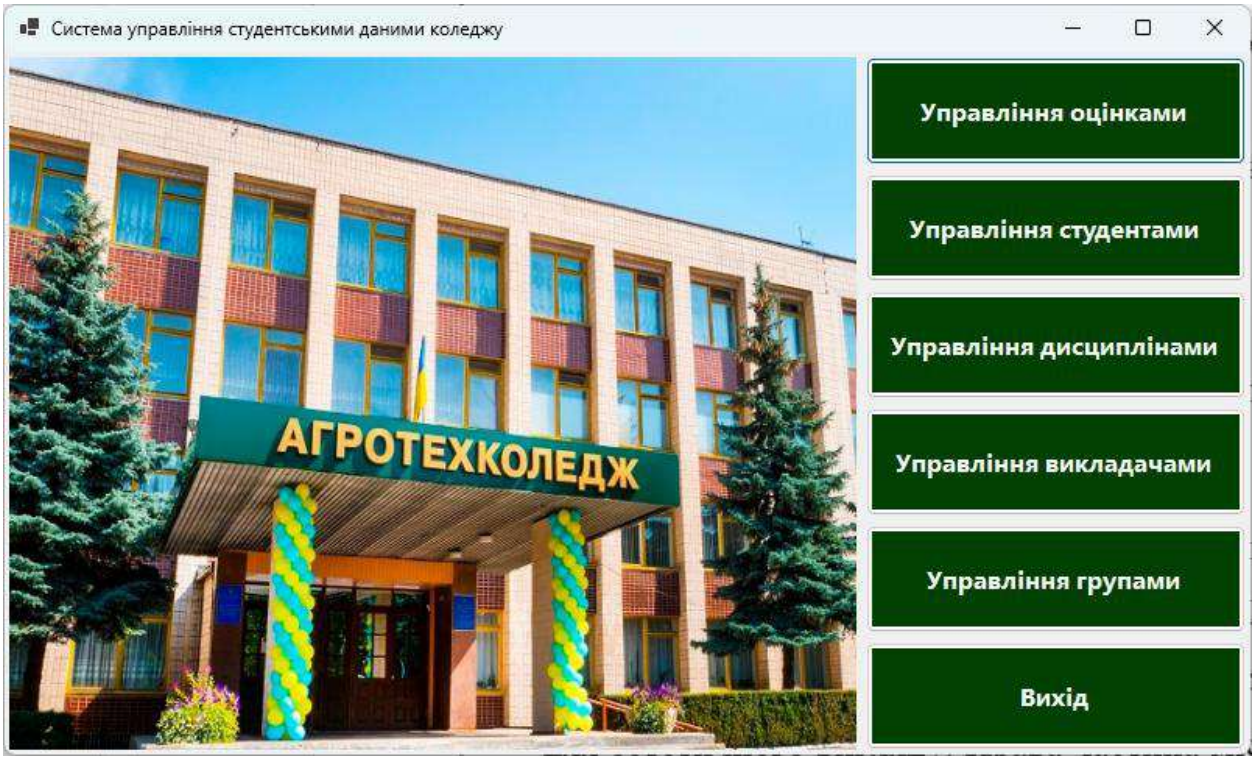


Рисунок 3.2.1 – Головна форма

Форма має розмір 800x450 пікселів і поділена на дві частини за допомогою компонента TableLayoutPanel. Ліву частину займає PictureBox із логотипом або зображенням коледжу, що розтягується на всю висоту форми для естетичного вигляду. Права частина містить шість кнопок, розташованих вертикально:

- "Управління оцінками" — відкриває форму для роботи з оцінками.
- "Управління студентами" — відкриває форму для управління студентами.
- "Управління дисциплінами" — відкриває форму для роботи з дисциплінами.
- "Управління викладачами" — відкриває форму для управління викладачами.

- "Управління групами" — відкриває форму для роботи з групами.
- "Вихід" — закриває програму з підтвердженням дії.

Кнопки мають темно-зелений фон (RGB: 0, 64, 0), білий текст і шрифт Segoe UI розміром 12 пунктів із жирним накресленням, що забезпечує чіткість і гарну читабельність. Форма відкривається в центрі екрана, її розмір фіксований, а кнопка максимізації відключена для збереження компактного вигляду.

Форма управління групами (GroupForm)

Форма управління групами дозволяє додавати, редагувати, видаляти та переглядати навчальні групи.

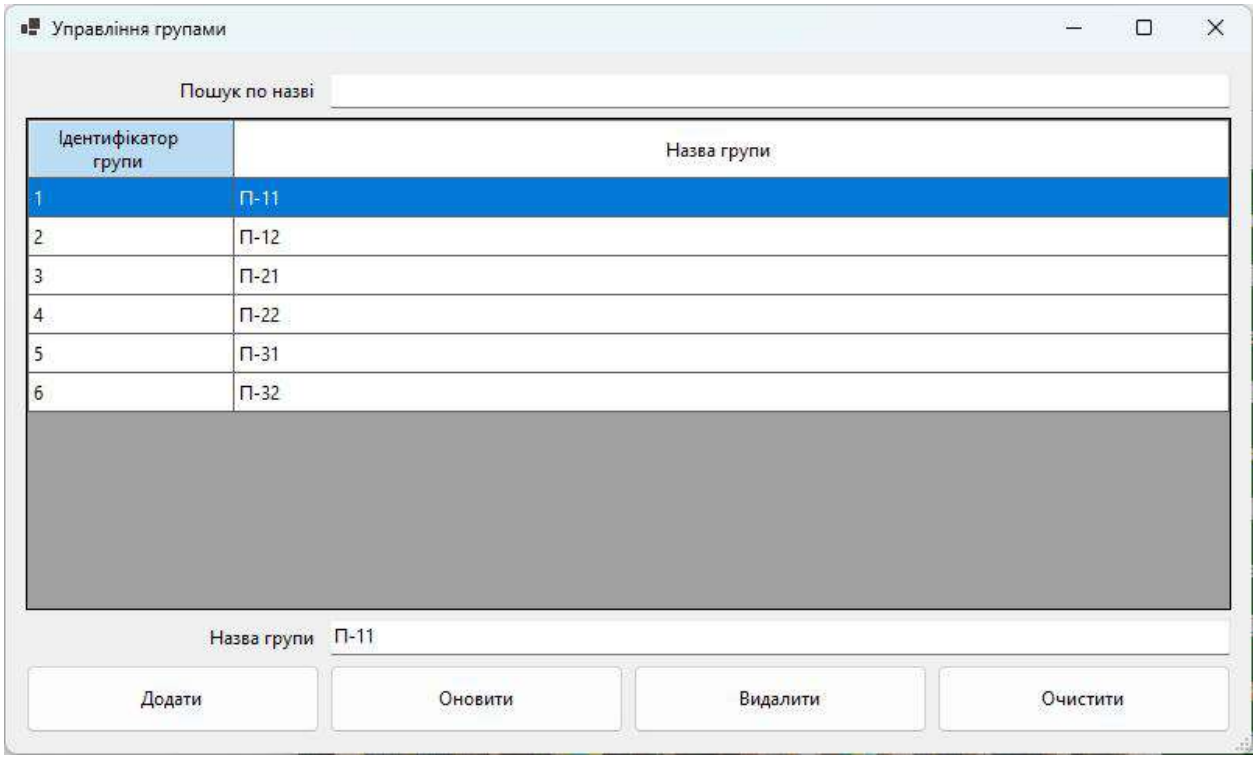


Рисунок 3.2.2 – Форма управління групами

У верхній частині розташовано DataGridView, який відображає список груп із колонками "Код групи" та "Назва групи". Таблиця займає більшу частину ширини форми (розмір форми 424x420 пікселів) і підтримує вибір одного рядка для редагування. Під таблицею розміщено текстове поле для введення назви групи, яке супроводжується міткою "Назва групи:". Нижче розташовані чотири кнопки: "Додати", "Оновити", "Видалити" та "Очистити", які дозволяють виконувати відповідні дії. У самому низу форми є поле для

пошуку з міткою "Пошук за назвою:", яке фільтрує групи в реальному часі за введеним текстом. Усі написи та повідомлення українською мовою, а елементи керування мають чітке розташування для зручного доступу.

Форма управління викладачами (TeacherForm)

Ця форма аналогічна за структурою до GroupForm, але призначена для роботи з викладачами.

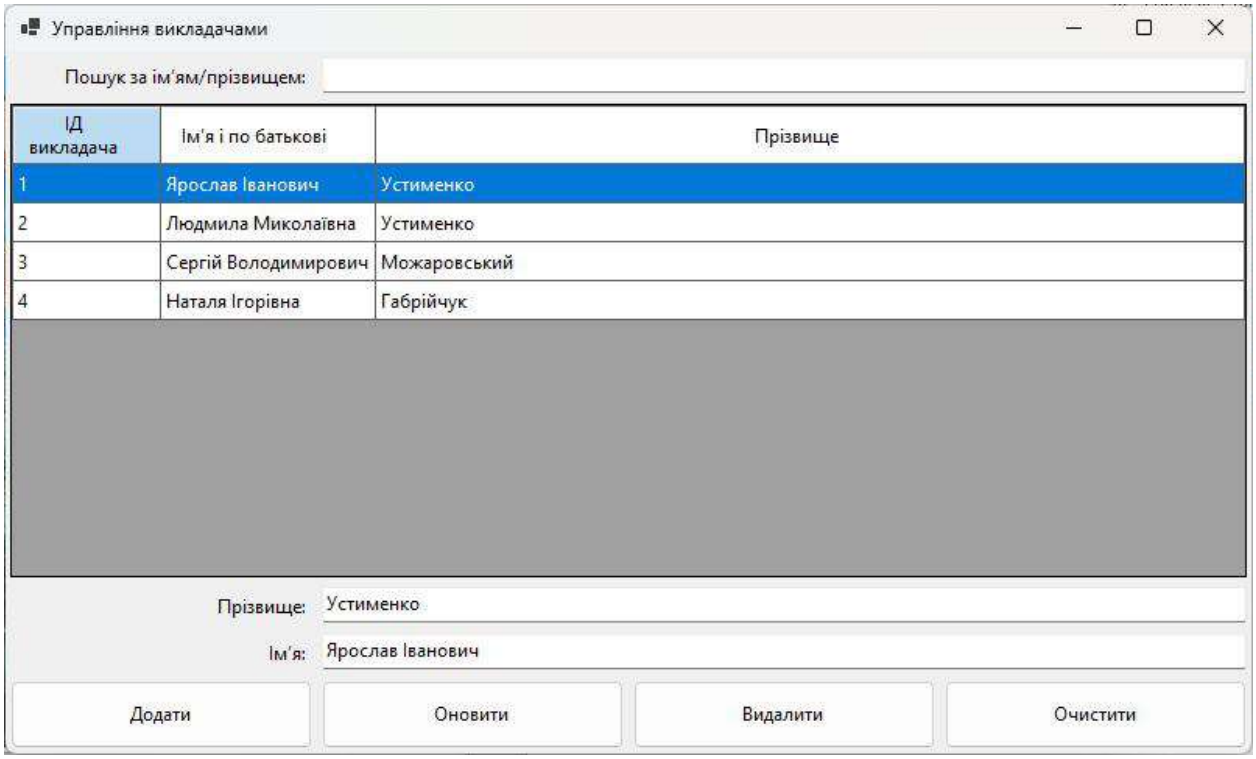


Рисунок 3.2.3 – Форма управління викладачами

DataGridView відображає колонки "Код викладача", "Ім’я" та "Прізвище". Під таблицею розташовані два текстові поля для введення імені та прізвища з відповідними мітками, а також кнопки "Додати", "Оновити", "Видалити" та "Очистити". Поле пошуку дозволяє фільтрувати викладачів за іменем або прізвищем. Інтерфейс зберігає єдиний стиль із іншими формами, включаючи розмір, кольорову схему кнопок і розташування елементів.

Форма управління дисциплінами (DisciplineForm)

Форма для управління дисциплінами має схожу компоновку, але включає додатковий елемент керування.

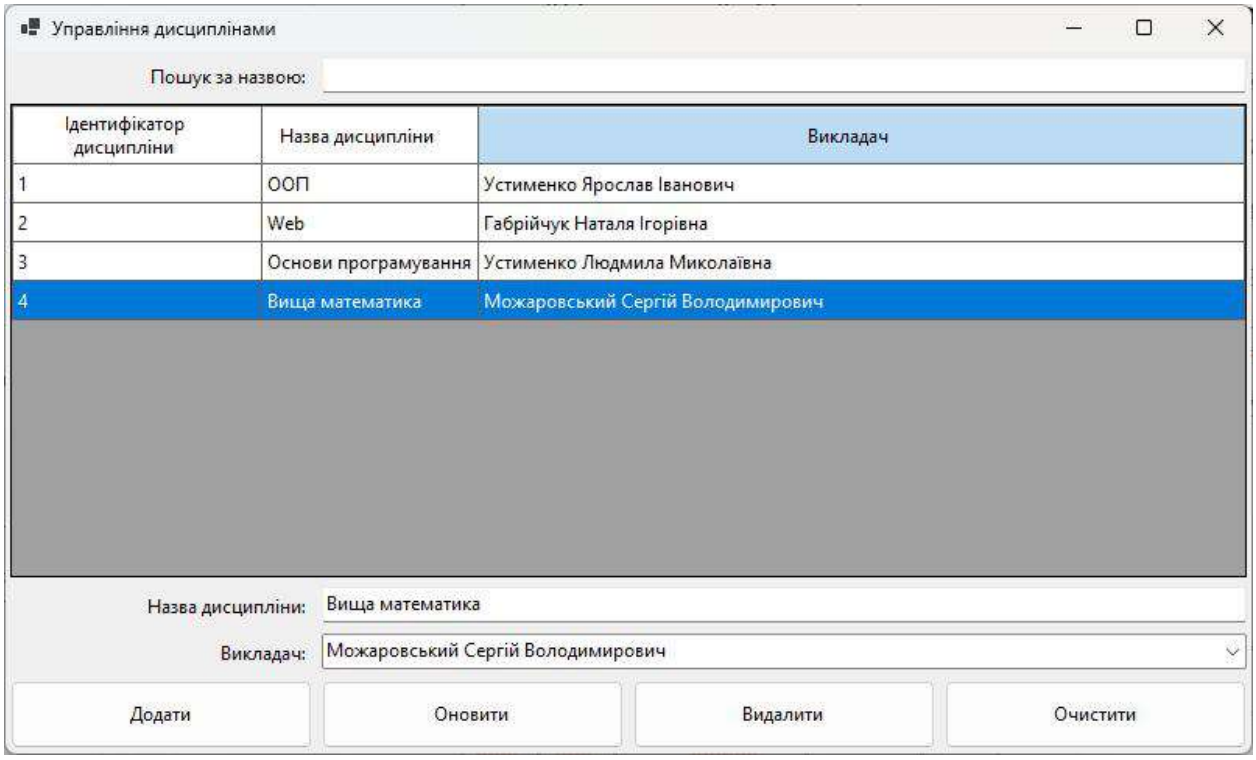


Рисунок 3.2.4 – Форма управління дисциплінами

DataGridView показує колонки "Код дисципліни", "Назва дисципліни" та "ПІБ викладача". Під таблицею розташовані текстове поле для назви дисципліни, ComboBox для вибору викладача (з відображенням повного імені) і стандартний набір кнопок. Поле пошуку фільтрує дисципліни за назвою. ComboBox автоматично заповнюється списком викладачів, а вибір викладача є обов’язковим для додавання чи оновлення дисципліни.

Форма управління студентами (StudentForm)

Ця форма є однією з ключових і дозволяє керувати даними студентів.

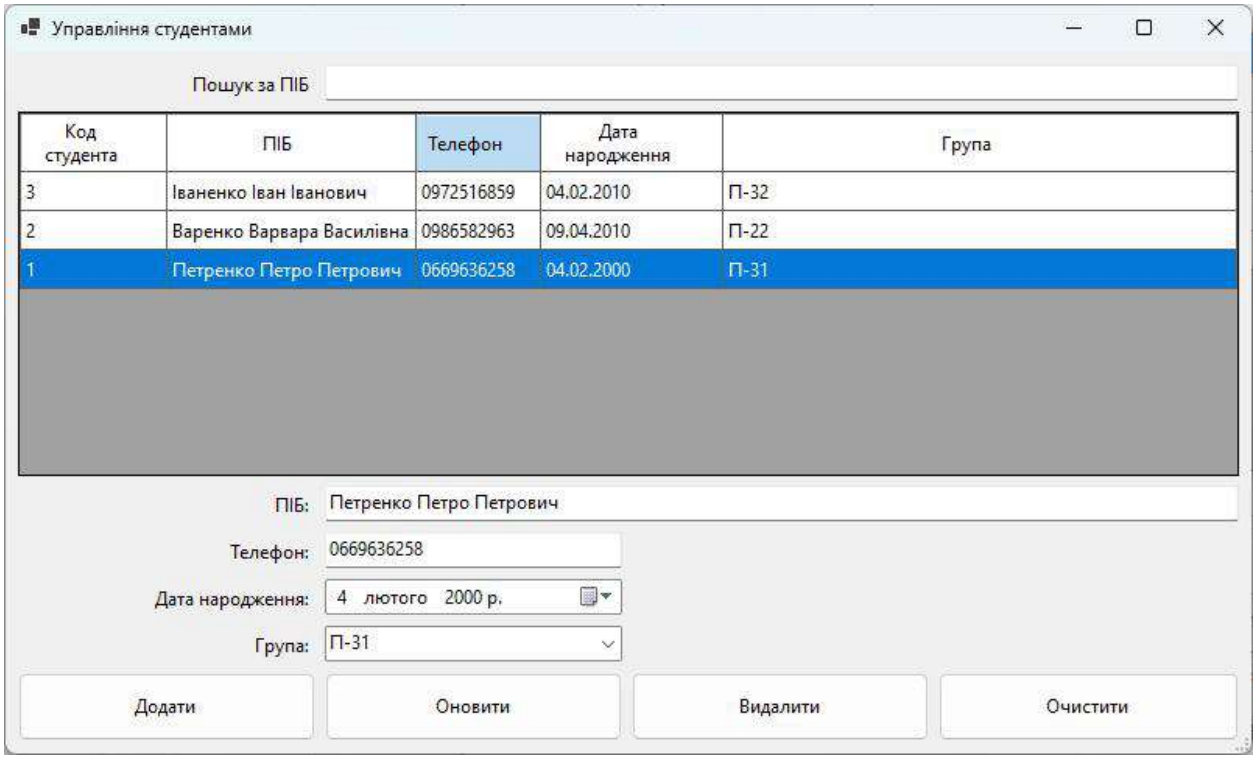


Рисунок 3.2.5 – Форма управління студентами

DataGridView відображає колонки "Код студента", "ПІБ", "Телефон", "Дата народження" та "Група". Під таблицею розташовані елементи введення: текстове поле для ПІБ, DateTimePicker для дати народження, текстове поле для телефону та ComboBox для вибору групи. Поле ПІБ є обов'язковим, тоді як телефон і дата народження опціональні. Кнопки "Додати", "Оновити", "Видалити" та "Очистити" дозволяють виконувати операції з даними. Поле пошуку фільтрує студентів за ПІБ. При додаванні чи оновленні система перевіряє унікальність ПІБ у межах вибраної групи, виводячи відповідне повідомлення у разі дублювання.

Форма управління оцінками (GradeForm)

Форма для роботи з оцінками має найскладніший інтерфейс через необхідність пов’язувати студентів і дисципліни.

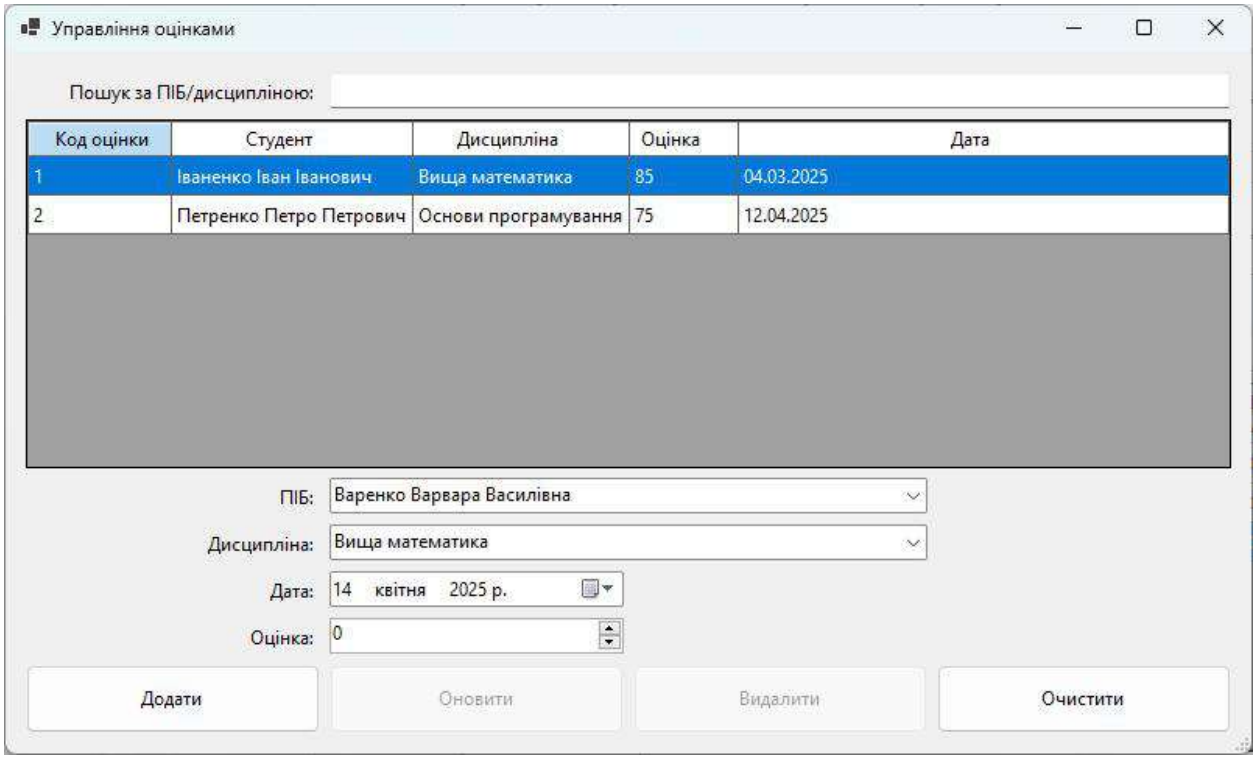


Рисунок 3.2.6 – Форма управління оцінками

DataGridView відображає колонки "Код оцінки", "Студент", "Дисципліна", "Оцінка" та "Дата". Під таблицею розміщено два ComboBox для вибору студента (за ПІБ) і дисципліни (за назвою), NumericUpDown для введення оцінки (з діапазоном 0–100) і DateTimePicker для вибору дати (опціонально). Кнопки "Додати", "Оновити", "Видалити" та "Очистити" відповідають за управління оцінками. Поле пошуку дозволяє фільтрувати записи за ПІБ студента або назвою дисципліни. Якщо список студентів чи дисциплін порожній, відповідні ComboBox і кнопки деактивуються для уникнення помилок.

Усі форми мають однаковий стиль оформлення: кнопки з темно-зеленим фоном і білим текстом, чіткі мітки українською мовою, таблиці з автоматичним підбором ширини колонок. DataGridView у всіх формах налаштований для відображення лише потрібних даних, із можливістю вибору одного рядка та забороною редагування безпосередньо в таблиці. Пошук

працює в реальному часі, що дозволяє швидко знаходити потрібні записи. Повідомлення про помилки чи успішні операції виводяться у спливаючих вікнах українською мовою, із відповідними іконками (попередження, інформація, помилка). Форми мають фіксований розмір (зазвичай 424x420 пікселів, крім MainForm), що забезпечує однаковий вигляд на різних комп'ютерах.

Інтерфейс системи є інтуїтивно зрозумілим і адаптованим до потреб адміністраторів і викладачів коледжу. Логічна структура форм, україномовна локалізація, чітке розташування елементів і зручні засоби пошуку забезпечують ефективну роботу з даними. Єдиний стиль оформлення та продумана навігація сприяють швидкому освоєнню програми, а підтримка всіх необхідних функцій через окремі форми гарантує виконання поставлених завдань.

3.3. Інструкція для роботи з програмою

Система управління студентськими даними коледжу розроблена для спрощення адміністративних і академічних процесів, таких як облік студентів, викладачів, груп, дисциплін і оцінок. Ця інструкція призначена для користувачів із базовими навичками роботи з комп'ютером і описує послідовність дій для виконання основних операцій у програмі. Інтерфейс є україномовним, інтуїтивно зрозумілим і не потребує спеціальної підготовки.

Встановлення та запуск програми

Перед початком роботи переконайтеся, що ваш комп'ютер відповідає мінімальним вимогам: операційна система Windows 7 або новіша, .NET, 2 ГБ оперативної пам'яті та 100 МБ вільного місця на диску. Програма постачається у вигляді виконуваного файлу (CollegeManagementSystem.exe) та файлу бази даних (college.db).

1. Скопіюйте файли програми в будь-яку папку на комп'ютері (наприклад, C:\CollegeSystem).

2. Двічі клацніть на CollegeManagementSystem.exe, щоб запустити програму.
3. При першому запуску програма автоматично створить базу даних college.db, якщо вона відсутня. Якщо з’являються повідомлення про помилки, перевірте наявність .NET або зверніться до адміністратора.

Робота з головною формою

Після запуску відкриється головна форма програми розміром 800x450 пікселів. Вона містить зображення зліва (логотип коледжу) та шість кнопок справа для доступу до основних функцій:

- Управління оцінками
- Управління студентами
- Управління дисциплінами
- Управління викладачами
- Управління групами
- Вихід

Щоб перейти до потрібного розділу, натисніть відповідну кнопку. Наприклад, виберіть "Управління студентами", щоб відкрити форму для роботи зі студентами. Для завершення роботи натисніть "Вихід" — програма попросить підтвердження, щоб уникнути випадкового закриття.

Управління групами

Форма управління групами дозволяє створювати, редагувати, видаляти та переглядати навчальні групи. Після відкриття форми у верхній частині з’явиться таблиця зі списком груп, що містить колонки "Код групи" та "Назва групи". У полі "Назва групи" введіть назву (наприклад, "П-21") і натисніть "Додати". Якщо поле порожнє або назва вже існує, з’явиться повідомлення про помилку. Після успішного додавання таблиця оновиться.

Виберіть групу в таблиці (клацніть на рядок), і її назва з’явиться в полі введення. Змініть назву та натисніть "Оновити".

Виберіть групу та натисніть "Видалити". Якщо група містить студентів, програма видасть попередження, і видалення буде скасовано.

Натисніть "Очистити", щоб скинути поле введення та зняти вибір із таблиці.

У полі "Пошук за назвою" введіть частину назви групи, і таблиця автоматично відфільтрує відповідні записи. Щоб повернути повний список, очистіть поле пошуку.

Для повернення до головної форми закрийте вікно, натиснувши на хрестик.

Управління викладачами

Форма управління викладачами працює аналогічно до форми груп, але дозволяє керувати даними викладачів.

Таблиця показує колонки "Код викладача", "Ім’я" та "Прізвище". Для додавання викладача введіть ім’я та прізвище у відповідні поля (наприклад, "Іван", "Петренко") і натисніть "Додати". Обидва поля є обов’язковими.

Для редагування викладача виберіть викладача в таблиці, змініть ім’я чи прізвище та натисніть "Оновити".

Для видалення викладача виберіть викладача та натисніть "Видалити". Якщо викладач пов’язаний із дисциплінами, видалення буде заборонено.

Для очищення полів натисніть "Очистити" для скидання полів.

Для пошуку введіть частину імені чи прізвища в поле пошуку, щоб відфільтрувати викладачів.

Управління дисциплінами

Форма управління дисциплінами дозволяє працювати з переліком дисциплін і їхніми викладачами.

Перегляд дисциплін: Таблиця містить колонки "Код дисципліни", "Назва дисципліни" та "ПІБ викладача".

Додавання дисципліни: Введіть назву дисципліни (наприклад, "Програмування"), виберіть викладача зі списку в ComboBox і натисніть "Додати". Якщо назва вже існує, з’явиться повідомлення про помилку.

Редагування дисципліни: Виберіть дисципліну, змініть назву чи викладача та натисніть "Оновити".

Видалення дисципліни: Виберіть дисципліну та натисніть "Видалити". Якщо до дисципліни прив’язані оцінки, видалення буде заборонено.

Очищення полів: Натисніть "Очистити", щоб скинути поля та вибір.

Пошук: Введіть частину назви дисципліни в поле пошуку для фільтрації.

Управління студентами

Форма управління студентами є однією з ключових і дозволяє працювати з даними студентів.

Перегляд студентів: Таблиця показує колонки "Код студента", "ПІБ", "Телефон", "Дата народження" та "Група".

Додавання студента: Введіть ПІБ у відповідне поле (обов’язкове). За бажанням вкажіть номер телефону та виберіть дату народження за допомогою DateTimePicker (за замовчуванням — поточна дата; якщо не змінювати, поле залишиться порожнім).

Виберіть групу зі списку в ComboBox.

Натисніть "Додати". Якщо ПІБ уже існує в цій групі, з’явиться повідомлення про помилку.

Редагування студента: Виберіть студента в таблиці, змініть потрібні дані (ПІБ, телефон, дату народження, групу) та натисніть "Оновити".

Видалення студента: Виберіть студента та натисніть "Видалити". Пов’язані оцінки автоматично видаляються.

Очищення полів: Натисніть "Очистити" для скидання всіх полів і вибору.

Пошук: Введіть частину ПІБ у поле пошуку, щоб відфільтрувати студентів.

Управління оцінками

Форма управління оцінками дозволяє реєструвати оцінки студентів за дисциплінами. Таблиця містить колонки "Код оцінки", "Студент", "Дисципліна", "Оцінка" та "Дата".

Додавання оцінки:

- Виберіть студента зі списку в ComboBox (за ПІБ).

- Виберіть дисципліну зі списку в ComboBox (за назвою).
- Введіть оцінку (0–100) за допомогою NumericUpDown.
- За бажанням виберіть дату за допомогою DateTimePicker (якщо не змінювати, поле залишиться порожнім).
- Натисніть "Додати". Якщо студент чи дисципліна не вибрані, з’явиться повідомлення про помилку.

Редагування оцінки: Виберіть оцінку в таблиці, змініть студента, дисципліну, оцінку чи дату та натисніть "Оновити".

Видалення оцінки: Виберіть оцінку та натисніть "Видалити".

Очищення полів: Натисніть "Очистити", щоб скинути вибір і поля.

Пошук: Введіть частину ПІБ студента чи назви дисципліни в поле пошуку для фільтрації оцінок.

Усі форми мають однаковий стиль: кнопки темно-зеленого кольору, чіткі україномовні мітки, таблиці з автопідбором ширини колонок.

Якщо виникають помилки (наприклад, спроба видалити групу зі студентами), програма виводить зрозумілі повідомлення з описом проблеми.

Для швидкого пошуку вводьте текст у поле пошуку — таблиця оновлюється автоматично.

Регулярно створюйте резервні копії файлу college.db, щоб уникнути втрати даних.

Якщо програма не запускається, перевірте наявність .NET або зверніться до адміністратора.

Щоб вийти з програми, поверніться на головну форму та натисніть "Вихід". Підтвердіть дію у спливаючому вікні, вибравши "Так". Усі зміни автоматично зберігаються в базі даних після виконання операцій (додавання, оновлення, видалення).

Програма є простою у використанні та дозволяє швидко виконувати всі необхідні операції з даними коледжу. Дотримання цієї інструкції забезпечить ефективну роботу з системою для адміністраторів і викладачів без потреби в додатковому навчанні.

Висновки до розділу 3

Третій розділ дипломної роботи присвячено керівництву користування розробленою системою управління студентськими даними коледжу, що забезпечує чітке розуміння її функціонування для кінцевих користувачів.

Описано мінімальні системні вимоги, які є доступними для більшості комп'ютерів у навчальних закладах: операційна система Windows 7 або новіша, .NET , 2 ГБ оперативної пам'яті та 100 МБ вільного місця. Це дозволяє використовувати програму навіть на застарілих машинах без потреби в серверному обладнанні чи підключенні до Інтернету, що відповідає потребам невеликих коледжів.

Інтерфейс користувача, реалізований через Windows Forms, є інтуїтивно зрозумілим і україномовним. Головна форма забезпечує зручну навігацію, а спеціалізовані форми для управління групами, викладачами, дисциплінами, студентами та оцінками мають єдиний стиль із чіткими мітками, таблицями, полями пошуку та кнопками. Така структура мінімізує час на освоєння програми та сприяє ефективній роботі.

Інструкція для роботи з програмою детально описує послідовність дій для виконання основних операцій: від запуску до управління даними та завершення роботи. Вона охоплює всі аспекти використання форм, включаючи додавання, редагування, видалення, пошук і обробку помилок, що робить систему доступною для користувачів із базовими комп'ютерними навичками.

Керівництво підтверджує, що система є зручною, функціональною та готовою до використання в реальних умовах коледжу. Продуманий інтерфейс і чіткі інструкції забезпечують швидке освоєння програми, а її низькі системні вимоги гарантують широку сумісність і легкість розгортання.

ВИСНОВКИ

Дипломна робота на тему «Система управління студентськими даними коледжу» була спрямована на створення ефективного програмного забезпечення для автоматизації адміністративних і академічних процесів у навчальних закладах. У процесі виконання роботи було досягнуто поставленої мети — розроблено настільний додаток, який забезпечує зручне управління даними про студентів, викладачів, групи, дисципліни та оцінки, відповідаючи потребам невеликих коледжів.

На етапі аналізу визначено ключові вимоги до системи, проведено огляд аналогічних рішень (Moodle, Blackboard, OpenSIS, Fedena) і обґрунтовано вибір технологічного стека: .NET , Windows Forms, SQLite і Visual Studio Community. Ці технології дозволили створити надійний, швидкий і простий у розгортанні додаток із україномовним інтерфейсом, адаптованим до локальних умов.

У процесі проектування розроблено оптимальні алгоритми для обробки даних (сортування, фільтрація, перевірка унікальності), спроєктовано базу даних із п’ятьма таблицями (Groups, Teachers, Disciplines, Students, Grades), пов’язаними зовнішніми ключами, та створено модульну структуру класів, що включає моделі, репозиторії та форми. Такий підхід забезпечив логічну організацію коду, легкість підтримки та можливість подальшого розширення.

Програмна реалізація охопила всі функціональні вимоги: облік даних, пошук, фільтрацію, перевірку унікальності та захист від помилок. Інтерфейс користувача, побудований на основі Windows Forms, виявився інтуїтивно зрозумілим завдяки єдиному стилю, чітким міткам і зручним елементам керування. Інструкція з використання підтверджує простоту освоєння програми для адміністраторів і викладачів без спеціальної підготовки.

Система відповідає нефункціональним вимогам, включаючи швидкодію, компактність і низькі системні вимоги (Windows 7+, 2 ГБ ОЗП, 100 МБ на диску). Використання SQLite усуває потребу в серверному

програмному забезпеченні, що робить додаток доступним для коледжів із обмеженим бюджетом.

Розроблена система є готовим рішенням для автоматизації управління студентськими даними, яке може бути впроваджене в реальних умовах. У перспективі її функціонал можна розширити, додавши модулі для створення звітів, інтеграцію з веб-інтерфейсом або підтримку додаткових сутностей, таких як розклади чи відвідуваність. Робота підтвердила актуальність і практичну цінність автоматизації освітніх процесів, а отримані результати можуть бути основою для подальших удосконалень системи.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Буч Г. Об’єктно-орієнтований аналіз і проектування з прикладами додатків / Г. Буч, Р. Максімчук, М. Енгл ; пер. з англ. — 3-тє вид. — К. : ДіаСофт, 2018. — 720 с.
2. Гаврилов О. В. Програмування на C# для початківців / О. В. Гаврилов. — Х. : Фабрика знань, 2020. — 432 с.
3. Гамаюнов Ю. П. Сучасні технології розробки програмного забезпечення / Ю. П. Гамаюнов, О. І. Коваль. — К. : НТУУ «КПІ», 2019. — 380 с.
4. Глушаков С. В. Бази даних : навч. посіб. / С. В. Глушаков, Д. В. Ломотько. — Х. : Фоліо, 2017. — 512 с.
5. Дейт К. Дж. Вступ до систем баз даних / К. Дж. Дейт ; пер. з англ. — 8-ме вид. — К. : Видавнича група BHV, 2019. — 928 с.
6. Еванс Е. Предметно-орієнтоване проектування / Е. Еванс ; пер. з англ. — К. : Фабула, 2021. — 448 с.
7. Коваленко О. О. Основи програмування на C# : навч. посіб. / О. О. Коваленко, В. П. Сидоренко. — К. : Видавництво «Ліра-К», 2020. — 296 с.
8. Конноллі Т. Бази даних : проектування, реалізація та підтримка / Т. Конноллі, К. Бег ; пер. з англ. — 6-те вид. — К. : Видавнича група BHV, 2018. — 1440 с.
9. Кормен Т. Алгоритми: побудова і аналіз / Т. Кормен, Ч. Лейзерсон, Р. Рівест, К. Штайн ; пер. з англ. — 3-тє вид. — К. : Видавництво «КМ-Букс», 2020. — 1320 с.
10. Крупник А. Б. Проектування інформаційних систем : навч. посіб. / А. Б. Крупник. — Л. : Видавництво Львівської політехніки, 2019. — 264 с.
11. Лафоре Р. Структури даних та алгоритми в C# / Р. Лафоре ; пер. з англ. — К. : ДіаСофт, 2017. — 608 с.
12. Макконнелл С. Досконалий код / С. Макконнелл ; пер. з англ. — 2-ге вид. — К. : Видавнича група BHV, 2020. — 896 с.

13. Мартин Р. Чистий код: створення, аналіз і рефакторинг / Р. Мартин ; пер. з англ. — К. : Фабула, 2019. — 464 с.
14. Мельник В. П. Основи роботи з базами даних SQLite / В. П. Мельник, І. О. Золотарьов. — К. : Видавництво «Центр учбової літератури», 2018. — 208 с.
15. Олійник О. М. Інформаційні системи в освіті : навч. посіб. / О. М. Олійник, Т. В. Кузнецова. — К. : Видавництво «Ліра-К», 2021. — 320 с.
16. Прохоренок Н. А. C# 9: професійне програмування / Н. А. Прохоренок, В. А. Дронов. — СПб. : Пітер, 2021. — 704 с.
17. Ріхтер Дж. CLR via C# / Дж. Ріхтер ; пер. з англ. — 4-те вид. — К. : Видавнича група BHV, 2019. — 864 с.
18. Скіна Дж. Алгоритми для початківців / Дж. Скіна ; пер. з англ. — К. : Видавництво «КМ-Букс», 2020. — 384 с.
19. Таненбаум Е. Сучасні операційні системи / Е. Таненбаум, Г. Бос ; пер. з англ. — 4-те вид. — К. : Видавнича група BHV, 2018. — 1120 с.
20. Шилдт Г. C#: повне керівництво / Г. Шилдт ; пер. з англ. — 5-те вид. — К. : Видавництво «ДіаСофт», 2020. — 1056 с.

ДОДАТКИ

ДОДАТОК А.

Програмний код модулів

```
namespace CollegeManagementSystem.Forms
{
   partial class MainForm
   {
      /// <summary>
      ///  Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;
      /// <summary>
      ///  Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }
      #region Windows Form Designer generated code
      /// <summary>
      ///  Required method for Designer support - do not modify
      ///  the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
```