МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:
Завідувач кафедри
комп'ютерних технологій та
економічної кібернетики
д. е. н., проф. П. М. Грицюк
_____
«____» _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітньо-кваліфікаційного рівня «бакалавр»
на тему:

«Web-додаток інформаційної системи як засіб оптимізації освітніх процесів школи»

Виконала: здобувачка вищої освіти за ОПП «Інформаційні системи та технології» спеціальності 126 «Інформаційні системи та технології», групи ІСТ-41
Курченко Анна Андріївна
Керівник:
к.е.н., доцент Волошин В.С.
Рецензент:
к.т.н., доцент Гладка О.М.

Рівне – 2024

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 56с., 50 рис., додатки на 57 стор., 11 літературних джерел.

Актуальність бакалавроської роботи полягає сучасному світі інформаційні технології відіграють ключову роль у всіх сферах нашого життя, включаючи освіту. Школи, як основний інститут формування та розвитку особистості, також потребують впровадження сучасних інформаційних технологій для оптимізації навчальних процесів, підвищення ефективності управління та підтримки комунікації між всіма учасниками освітнього процесу. Проектування та розробка інформаційної системи для школи стає актуальною задачею у зв'язку з різноманітністю потреб сучасного освітнього середовища.

Об'єктом дослідження цієї роботи є інформаційна система для школи, яка призначена для оптимізації управління навчальним процесом, забезпечення доступу до необхідної інформації для учнів, вчителів та адміністрації, а також покращення комунікації між усіма учасниками освітнього процесу.

Метою роботи є дослідження процесу проектування та розробки інформаційної системи для школи та розробка концепції та структури інформаційної системи, що відповідає потребам сучасного освітнього процесу. Провести практичну реалізацію розробленої інформаційної системи та оцінити її ефективність та корисність для школи.

Запропонована інформаційна система для школи демонструє високу ефективність та зручність у використанні, забезпечуючи покращення комунікації між учасниками освітнього процесу та оптимізацію управління навчальними процесами. Система створена з урахуванням потреб учнів, вчителів та адміністрації, що сприяє підвищенню якості освіти.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНА СИСТЕМА, ШКОЛА, НАВЧАЛЬНИЙ ПРОЦЕС, УПРАВЛІННЯ, БАЗИ ДАНИХ, ОПТИМІЗАЦІЯ, КОМУНІКАЦІЯ, SCHOOL MANAGEMENT SYSTEM, SQL, AZURE.

ЗМІСТ

ВСТУП

У даній дипломній роботі розглянуто впровадження інформаційних систем в галузі освіти, їх вплив на навчальний процес, управління та комунікацію.

Актуальність теми роботи. У сучасному світі інформаційні технології відіграють ключову роль у всіх сферах нашого життя, включаючи освіту. Школи, як основний інститут формування та розвитку особистості, також потребують впровадження сучасних інформаційних технологій для оптимізації навчальних процесів, підвищення ефективності управління та підтримки комунікації між всіма учасниками освітнього процесу.

Проектування та розробка інформаційної системи для школи стає актуальною задачею у зв'язку з різноманітністю потреб сучасного освітнього середовища. Ця система має вирішувати такі важливі завдання, як автоматизація управління навчальним процесом, моніторинг успішності учнів, забезпечення доступу до навчального матеріалу та сприяння ефективної комунікації між учнями, вчителями та адміністрацією школи.

У дипломній роботі буде розглянуто процес проектування та розробки інформаційної системи для школи. Будуть вивчені вимоги до такої системи, аналіз існуючих підходів та технологій, а також розроблено концепцію та структуру системи, яка відповідає потребам сучасного освітнього процесу.
Ця робота має на меті не лише теоретичний аналіз, але й практичне застосування знань у процесі розробки конкретного проекту інформаційної системи для школи, що сприятиме покращенню якості навчального процесу та ефективності управління навчальним закладом.

Метою роботи є дослідження процесу проектування та розробки інформаційної системи для школи та розробка концепції та структури інформаційної системи, що відповідає потребам сучасного освітнього процесу. Провести практичну реалізацію розробленої інформаційної системи. Оцінити ефективність та корисність інформаційної системи для школи.

Об'єктом дослідження цієї роботи є інформаційна система для школи, яка призначена для оптимізації управління навчальним процесом, забезпечення доступу до необхідної інформації для учнів, вчителів та адміністрації, а також покращення комунікації між усіма учасниками освітнього процесу. Така система має спрощувати проведення адміністративних процедур, автоматизувати ведення обліку та аналізу даних.

Предметом дипломної роботи є процес проектування та розробки інформаційної системи для школи. В рамках роботи будуть досліджені різні аспекти цього процесу, включаючи аналіз потреб користувачів, визначення вимог до системи, проектування архітектури та інтерфейсу, розробку програмного забезпечення, тестування та впровадження системи, а також оцінку її ефективності. Весь цей процес спрямований на створення інформаційної системи, яка оптимізує навчальні процеси, полегшує управління школою та підвищує якість освіти.

Завданням бакалаврської роботи є:

1. **Проведення аналізу:**

   Аналіз сучасних тенденцій у використанні інформаційних технологій у сфері освіти. Визначення ключових питань та проблем, що виникають у процесі розробки та впровадження інформаційних систем у шкільному середовищі.

2. **Визначення вимог**

   Провести опитування або інтерв'ювання представників шкільної громадськості (учнів, вчителів, адміністрації) для визначення їх потреб та вимог до інформаційної системи. Скласти список функціональних та нефункціональних вимог до системи на основі отриманих результатів.

3. **Проектування системи**:

   Розробка структурної схеми інформаційної системи, що відображає всі її компоненти та їх взаємозв'язки. Визначення основних модулів та їх

функціональних можливостей. Вибір оптимальних технологій для реалізації кожного компонента системи.

4. **Розробка програмного забезпечення:**

   Створення дизайну та інтерфейсу користувача на основі визначених вимог, використовуючи обрані технології та методи. Написання програмного коду для реалізації функціональних можливостей системи.

РОЗДІЛ 1. ІНФОРМАЦІЙНЕ ЗАБЕЗБЕЧЕННЯ ОСВІТНІХ ПРОЦЕСІВ ШКОЛИ

1.1 Сучасний стан впровадження інформаційних систем в діяльності начальних закладів загальної середньої освіти

Сучасний розвиток технологій неодмінно впливає на всі сфери нашого життя, включаючи системи освіти. Інформаційні технології стають невід'ємною складовою навчального процесу, а інформаційні системи для шкіл відіграють ключову роль у забезпеченні ефективного управління та навчання.

На сьогоднішній день інформаційні системи у школах використовуються для автоматизації ряду процесів, що дозволяє значно підвищити ефективність управління та покращити якість освіти [7]. Розглянемо основні аспекти впровадження та використання інформаційних систем в навчальних закладах загальної середньої освіти.

1. **Використання технологій у шкільному середовищі:**

   Сучасні школи активно впроваджують інформаційні технології для полегшення навчального процесу та управління шкільною діяльністю. Інформаційні системи забезпечують доступ до даних, автоматизацію процесів та підтримку різних аспектів шкільної діяльності. Зокрема, електронні журнали стають стандартом для ведення обліку відвідуваності, оцінок і домашніх завдань. Це спрощує процес оцінювання та комунікації з батьками.

2. **Розмаїття функцій і можливостей:**

   Сучасні інформаційні системи для шкіл надають широкий спектр функцій, таких як ведення електронного журналу, планування уроків, ведення обліку учнів та вчителів, організація комунікації між учасниками навчального процесу. Інтегровані платформи дозволяють ефективно управляти розкладом, моніторити успішність учнів та забезпечувати взаємодію між учасниками навчального процесу.

3. **Орієнтованість на інтерактивне навчання**:

   В сучасному освітньому середовищі все більше акцентується на інтерактивне навчання та використання цифрових інструментів для залучення учнів до процесу навчання. Інформаційні системи для шкіл відповідають цим вимогам, надаючи інтерактивні навчальні ресурси та інструменти. Використання відео, аудіо, інтерактивних завдань і тестів дозволяє створити більш залучаюче та ефективне навчальне середовище.

4. **Інтеграція з іншими системами:**

   Сучасні інформаційні системи для шкіл здатні інтегруватися з іншими системами, такими як електронні бібліотеки, системи електронного навчання, електронні документообіги та інші, що забезпечує зручну роботу та обмін даними між різними складовими шкільної інфраструктури. Інтеграція з платформами управління навчанням (LMS) дозволяє школам об'єднувати навчальні матеріали, оцінювання та комунікацію в єдину систему.

5. **Зростаюча популярність хмарних технологій:**

   Все більше шкіл вибирають хмарні рішення для зберігання та обробки даних, що дозволяє забезпечити доступ до інформації з будь-якого пристрою та місця з підключенням до Інтернету. Хмарні технології забезпечують доступність навчальних ресурсів і спільну роботу над проектами, документами та іншими навчальними завданнями. Зокрема, такі платформи як Google Classroom та Microsoft Teams стали популярними інструментами для організації дистанційного навчання та спільної роботи учнів і вчителів.

6. **Зростання вимог до безпеки даних:**

   З огляду на збільшення кількості цифрових даних у шкільному середовищі, належна захищеність та конфіденційність цих даних стає все важливішою. Сучасні інформаційні системи для шкіл повинні враховувати цей аспект та надавати ефективні засоби захисту даних.

Захист персональних даних учнів і співробітників та запобігання кібератакам стає критично важливим завданням. Використання двофакторної автентифікації, шифрування даних та регулярних резервних копій є необхідними заходами для забезпечення безпеки.

7. **Автоматизація процесів:**

   Школи все частіше використовують інформаційні системи для автоматизації рутинних процесів, таких як ведення журналів, формування звітів, планування уроків та інші. Це дозволяє звільнити час педагогів та адміністраторів для більш ефективного використання їхніх навчальних та управлінських зусиль. Автоматизоване оцінювання знань учнів та автоматизація адміністративних задач підвищує загальну ефективність шкільної діяльності.

8. **Персоналізоване навчання:**

   Деякі інформаційні системи надають можливість персоналізації навчального процесу для кожного учня, враховуючи їхні індивідуальні потреби, здібності та інтереси. Адаптивне навчання дозволяє системам автоматично підлаштовуватися під рівень знань і темп навчання учня, а також створювати і відстежувати персоналізовані навчальні плани. Це сприяє покращенню результатів навчання та підвищує мотивацію учнів.

9. **Підтримка різних типів шкіл:**

   Інформаційні системи повинні бути адаптовані до різних типів шкіл, таких як загальноосвітні, професійно-технічні, гімназії та інші, враховуючи їхні особливості та потреби. Гнучкість і модульність систем дозволяє налаштовувати їх під конкретні вимоги та потреби кожного навчального закладу.

Здійснюючи огляд сучасного стану галузі інформаційних систем для шкіл, можна визначити значний прогрес та розвиток у цій області. Інформаційні технології стають необхідним інструментом для покращення навчання, адміністрування та спілкування в освітньому середовищі.

Сучасні інформаційні системи для шкіл надають різноманітні можливості, від автоматизації адміністративних процесів до навчання за допомогою дистанційних засобів. Вони сприяють покращенню доступу до навчальних ресурсів, забезпечують ефективне ведення обліку успішності учнів та забезпечують аналіз даних для прийняття обґрунтованих рішень. Подальший розвиток цих систем буде сприяти підвищенню якості освіти та ефективності управління шкільними ресурсами.

1.2 Аналіз інформаційних систем в галузі освіти

Впровадження інформаційних систем в галузі освіти стає дедалі важливішим для забезпечення ефективності навчального процесу, управління ресурсами та комунікації між учасниками освітнього процесу. Аналізуючи сучасні інформаційні системи, які використовуються в галузі освіти, можна виділити кілька ключових аспектів та тенденцій, що визначають їх розвиток та впровадження [11].

1. Типи інформаційних систем для освіти

Інформаційні системи, які використовуються в освіті [10], можна класифікувати за їх функціональним призначенням :

1.1 Системи управління навчальним процесом (LMS): Ці системи дозволяють планувати, реалізовувати та оцінювати процес навчання. Вони забезпечують доступ до навчальних матеріалів, тестів, завдань та забезпечують зворотній зв'язок між учнями та викладачами. Популярними прикладами LMS є Moodle, Google Classroom, Blackboard.

1.2 Системи управління навчальним закладом (SMS): Вони охоплюють адміністративні аспекти управління школою, включаючи облік відвідуваності, розклад занять, облік успішності учнів, управління персоналом та фінансовими ресурсами. Приклади таких систем включають EduPage, Schoology, PowerSchool.

1.3 Електронні бібліотеки та ресурси: Ці системи забезпечують доступ до широкого спектру навчальних матеріалів, наукових статей, книг та інших ресурсів, які підтримують навчальний процес. Прикладами є JSTOR, Google Scholar, eLibrary.

1.4 Системи дистанційного навчання: Платформи для проведення онлайн-занять, вебінарів, відеоконференцій, які стали особливо популярними в умовах пандемії COVID-19. Відомі приклади - Zoom, Microsoft Teams, Cisco WebEx.

2. Функціональні можливості інформаційних систем

Сучасні інформаційні системи для освіти пропонують широкий спектр функціональних можливостей, які сприяють оптимізації освітніх процесів [10]:

- **Адміністрування:** Ведення електронних журналів, управління розкладом, облік відвідуваності та успішності учнів.
- **Навчання:** Доступ до навчальних матеріалів, інтерактивні завдання, тести, форуми для обговорень.
- **Комунікація:** Інструменти для взаємодії між учнями, викладачами та батьками, включаючи чати, повідомлення, оголошення.
- **Аналіз та звітність:** Генерація звітів про успішність учнів, аналітичні інструменти для оцінки ефективності навчального процесу.

3. Тенденції в розвитку інформаційних систем для освіти

Сучасні тенденції у впровадженні інформаційних систем для освіти визначаються кількома ключовими напрямками [11]:

- **Персоналізація навчання:**

  Інформаційні системи все частіше використовують алгоритми адаптивного навчання, які дозволяють підлаштовувати навчальні матеріали та завдання під індивідуальні потреби та здібності учнів.

- **Інтеграція з іншими платформами:**

  Сучасні системи прагнуть до інтеграції з різними освітніми ресурсами та платформами, що забезпечує єдине середовище для управління навчальним процесом.

- **Хмарні технології:**

  Використання хмарних рішень дозволяє зберігати дані в Інтернеті, забезпечуючи доступ до них з будь-якого пристрою та місця, а також сприяє спільній роботі над проектами.

- **Мобільні технології:**

  Зростаюча популярність мобільних пристроїв спонукає до розробки мобільних додатків для освітніх систем, що забезпечує зручний доступ до навчальних матеріалів та комунікації в будь-який час.

- **Безпека даних:**

  Захист персональних даних та конфіденційність стають пріоритетними завданнями для розробників освітніх систем. Впроваджуються нові стандарти безпеки та технології шифрування даних.

4. Вплив інформаційних систем на освітній процес

Впровадження інформаційних систем в освітній процес має значний вплив на його якість та ефективність [11]:

- **Покращення доступності освіти:**

  Інформаційні системи дозволяють учням отримувати доступ до навчальних матеріалів і ресурсів незалежно від їхнього місцезнаходження, що особливо важливо для віддалених районів.

- **Підвищення ефективності навчання:**

  Інтерактивні методи навчання, використання мультимедіа та адаптивних систем підвищують зацікавленість та успішність учнів.

- **Зменшення навантаження на викладачів:**

Автоматизація рутинних процесів, таких як оцінювання, ведення журналів та формування звітів, звільняє час викладачів для більш творчої та індивідуальної роботи з учнями.

- **Підвищення прозорості та підзвітності:**
  Системи забезпечують прозорість оцінювання, доступ до інформації про успішність учнів для батьків, що сприяє кращій взаємодії між школою та сім'єю.

5. Виклики та проблеми впровадження інформаційних систем

Незважаючи на значні переваги, впровадження інформаційних систем в освіті супроводжується певними викликами:

- **Фінансові витрати:**
  Закупівля, впровадження та підтримка інформаційних систем потребує значних фінансових ресурсів, що може бути проблематичним для деяких навчальних закладів.
- **Навчання персоналу:**
  Впровадження нових систем вимагає навчання викладачів та адміністративного персоналу, що може потребувати часу та додаткових ресурсів.
- **Технічні проблеми:**
  Недостатня технічна підтримка, проблеми з інтеграцією систем та технічними збоями можуть створювати труднощі в їх щоденному використанні.
- **Забезпечення безпеки:**
  Питання захисту даних учнів та викладачів, а також запобігання несанкціонованому доступу до систем залишаються актуальними проблемами.

Аналіз інформаційних систем в галузі освіти показує, що їх впровадження є важливим кроком на шляху до модернізації освітнього процесу. Інформаційні системи надають широкий спектр можливостей для покращення управління,

комунікації та навчання в школах. Проте, успішне впровадження цих систем вимагає врахування фінансових, технічних та організаційних аспектів. З урахуванням сучасних тенденцій та викликів, інформаційні системи продовжують розвиватися, сприяючи підвищенню якості освіти та ефективності управління навчальними закладами.

1.3 Порівняльна характеристика мов програмування для роботи з базами даних C# та Java

Мови програмування C# та Java є одними з найбільш популярних і використовуються для розробки програмного забезпечення різного рівня складності, включаючи системи, що працюють з базами даних. У цьому розділі ми детально розглянемо їх особливості, переваги та недоліки у контексті роботи з базами даних.

Основні характеристики мов C# та Java

C# (C-sharp) — це сучасна мова програмування, розроблена компанією Microsoft як частина платформи .NET. Вона орієнтована на спрощення розробки, забезпечуючи високу продуктивність і надійність [8].

Особливості:

- **Інтеграція з .NET Framework:** Дозволяє розробляти різноманітні типи додатків, включаючи веб-додатки, настільні додатки та сервіси.
- **LINQ (Language Integrated Query):** Надає зручний спосіб роботи з базами даних за допомогою запитів, інтегрованих у саму мову.
- **Ефективна підтримка ORM:** Інструменти як-от Entity Framework забезпечують спрощену роботу з базами даних через об'єктно-реляційне відображення (ORM).
- **Підтримка паралельного програмування**: C# дозволяє ефективно працювати з багатопоточністю.

- **Синтаксис**: Інтуїтивно зрозумілий синтаксис, що поєднує найкращі риси C++ та Java.
- **Платформа .NET**: Можливість створення кросплатформених додатків за допомогою .NET Core.

Java — це об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems (нині належить Oracle). Вона є однією з найпопулярніших мов для розробки різноманітних типів програмного забезпечення [9].

Особливості:

- **Платформонезалежність:** Завдяки використанню Java Virtual Machine (JVM) код Java можна виконувати на будь-якій платформі, яка має JVM.
- **JDBC (Java Database Connectivity):** Потужний API для роботи з базами даних, що забезпечує універсальність та гнучкість у роботі з різними СУБД.
- **Ефективна підтримка ORM:** Інструменти як-от Hibernate та JPA (Java Persistence API) надають можливість простого та ефективного управління базами даних через ORM.
- **Широка екосистема бібліотек та фреймворків:** Велика кількість готових рішень для різноманітних задач.
- **Безпека:** Вбудовані механізми для забезпечення безпеки та захисту даних.
- **Мобільність:** Використовується для розробки додатків на Android платформі (таблиця 1).

- **Інтеграція з базами даних C#**

- • **ADO.NET:** Основна технологія для роботи з базами даних у .NET. ADO.NET дозволяє виконувати запити до баз даних, читати та записувати дані, керувати транзакціями.
- • **Entity Framework (EF):** ORM (Object-Relational Mapping) для .NET, що дозволяє працювати з базами даних за допомогою об'єктно-орієнтованого підходу. EF спрощує процес розробки, забезпечуючи автоматичне відображення об'єктів на таблиці бази даних.

- • **LINQ:** Мова запитів, інтегрована в C#, яка дозволяє писати запити до баз даних у стилі, схожому на SQL, але з використанням синтаксису C#.

Таблиця 1

Порівняльна характеристика C# та Java для роботи з базами даних

Параметр	C#	Java
Платформа	.NET Framework/.NET Core	Java SE/EE
Основний API для роботи з БД	ADO.NET, LINQ, Entity Framework	JDBC, JPA, Hibernate
Інтеграція з ORM	Entity Framework, Dapper	Hibernate, JPA
Підтримка паралельного виконання	Потужні можливості багатопоточності, Task Parallel Library (TPL)	Java Concurrency API, Fork/Join Framework
Інструменти для автоматизації тестування	MSTest, NUnit	JUnit, TestNG
Хмарні сервіси	Azure SQL Database	Google Cloud SQL, Amazon RDS
Мобільна розробка	Xamarin	Android SDK
Продуктивність	Висока, особливо в екосистемі Windows	Висока, незалежно від платформи
Безпека	Інтегровані механізми безпеки .NET	Потужні вбудовані механізми безпеки
Гнучкість та універсальність	Добре інтегрована з Microsoft екосистемою	Широка платформа для різних типів додатків
Підтримка спільноти	Активна спільнота, зосереджена на .NET	Дуже велика та активна спільнота

Java:

• **JDBC (Java Database Connectivity):** Основний API для роботи з базами даних у Java. JDBC дозволяє виконувати SQL-запити, керувати з'єднаннями, транзакціями та обробкою результатів.

• **Hibernate:** ORM для Java, що забезпечує автоматичне відображення об'єктів на таблиці бази даних. Hibernate пропонує потужні інструменти для управління даними, кэшування та оптимізації запитів.

• **JPA (Java Persistence API):** Специфікація для ORM у Java, яка підтримується багатьма фреймворками, включаючи Hibernate. JPA стандартизує процес роботи з базами даних, забезпечуючи простий та уніфікований API для взаємодії з різними СУБД.

Порівняння функціональних можливостей

C# (Entity Framework): Забезпечує високу продуктивність і простоту у використанні завдяки автоматичному генеруванню SQL-запитів, підтримці міграцій та зручним інструментам для налаштування моделей даних [8].

Java (Hibernate): Також забезпечує високий рівень продуктивності, але налаштування може вимагати більше зусиль, особливо для новачків. Hibernate пропонує потужні можливості для оптимізації та налаштування кешування, але це додає складності [9].

Підтримка платформ:

C#: Орієнтований на платформу Windows, хоча з появою .NET Core та .NET 5/6 підтримка інших платформ (Linux, macOS) значно покращилася. C# можна використовувати для розробки кросплатформенних додатків завдяки .NET Core [8].

Java: Незалежна від платформи завдяки JVM, що забезпечує високу кросплатформеність та сумісність з різними операційними системами. Java додатки можуть запускатися на будь-якій платформі з встановленою JVM [9].

Продуктивність

C#:

- **Entity Framework:** Може бути менш ефективним у порівнянні з низькорівневими API, такими як ADO.NET, через автоматичне генерування запитів і управління об'єктами. Однак, LINQ забезпечує зручність роботи та скорочення часу розробки, що може компенсувати потенційні втрати продуктивності.
- **ADO.NET:** Забезпечує високу продуктивність завдяки прямому доступу до бази даних і низькорівневим API, але вимагає більше коду та зусиль для реалізації.

Java:

- **Hibernate**: Забезпечує високу продуктивність завдяки оптимізації запитів і підтримці кешування. Однак, налаштування може бути складним і вимагати додаткових знань.
- **JDBC**: Пропонує низькорівневий доступ до баз даних і високу продуктивність, але вимагає більше зусиль для реалізації та управління з'єднаннями і транзакціями.

Хмарні технології

C#

- **Microsoft Azure:** Забезпечує широкий спектр інструментів та сервісів для роботи з базами даних, включаючи Azure SQL Database, Cosmos DB та інші.
- **AWS (Amazon Web Services):** Підтримка таких сервісів, як Amazon RDS, Amazon DynamoDB, що дозволяє розробникам C# ефективно працювати з базами даних у хмарі.

Java

- **Google Cloud Platform:** Надає потужні інструменти для роботи з базами даних, такі як Google Cloud SQL, BigQuery та інші.

- **Amazon Web Services:** Підтримує широкий спектр баз даних, включаючи Amazon RDS, Amazon Aurora та інші, що дозволяє Java розробникам легко інтегрувати свої додатки з хмарними сервісами.

Безпека

C#

- Механізми безпеки .NET: Вбудовані інструменти та бібліотеки для захисту даних, включаючи шифрування, управління автентифікацією та авторизацією, а також забезпечення захищених з'єднань.
- Active Directory Integration: Спрощує управління доступом та забезпечення безпеки в корпоративних середовищах.

Java

- **Java Security:** Потужні інструменти для забезпечення безпеки додатків, включаючи шифрування, підписи, управління автентифікацією та авторизацією.
- **Spring Security:** Популярний фреймворк для управління безпекою додатків, що забезпечує комплексний підхід до захисту даних.

Переваги та недоліки

Переваги C#

- **Інтеграція з Windows:** Повна сумісність та оптимізація під ОС Windows.
- **Потужні засоби для розробки:** Розширений набір інструментів для розробки, відладки та тестування, інтегрованих в Visual Studio.
- **Можливості паралельного програмування:** Потужні засоби для багатопоточності та асинхронного програмування.

Недоліки C#

- **Залежність від платформи:** Хоча .NET Core є кросплатформеним, історично C# був тісно пов'язаний з Windows.
- **Обмежена мобільність:** Для розробки мобільних додатків на інших платформах потрібен додатковий інструментарій, такий як Xamarin.

Переваги Java

- **Платформонезалежність:** Працює на будь-якій платформі з JVM.
- **Широка підтримка спільноти:** Велика кількість ресурсів, бібліотек та фреймворків, доступних для розробників.
- **Гнучкість:** Використовується для розробки веб-додатків, мобільних додатків, серверних рішень тощо.

Недоліки Java

- **Продуктивність:** Може поступатися у швидкодії нативним мовам через абстракції JVM.
- **Складність налаштування:** Налаштування середовища розробки може бути складнішим порівняно з C# та .NET.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Логічна модель даних та база даних системи

Логічна модель даних

Логічна модель даних є ключовим етапом у проектуванні інформаційної системи, оскільки вона є абстрактним представленням даних і взаємозв'язків між ними у базі даних. Вона описує структуру даних, їх атрибути та відносини між ними.

Логічна модель даних складається з основних елементів:

- **Сутності:** Основні об'єкти, про які зберігаються дані. Наприклад, у шкільній інформаційній системі це можуть бути учні, вчителі, класи, предмети тощо.
- **Атрибути:** Властивості або характеристики сутностей. Наприклад, атрибутами сутності "Учень" можуть бути ім'я, прізвище, дата народження, адреса тощо.
- **Зв'язки:** Відношення між сутностями. Вони показують, як сутності взаємодіють одна з одною. Наприклад, зв'язок між сутністю "Учень" і сутністю "Клас" може показувати, до якого класу належить учень.
- **Ключові поля:** Унікальні ідентифікатори для сутностей. Наприклад, у сутності "Учень" це може бути ідентифікаційний номер учня.

Це структуроване представлення даних допомагає розробникам та аналітикам уявити, як дані будуть зберігатися та використовуватися в системі.

За допомогою схеми на рис. 2.1 зображені зв’язки між таблицями.

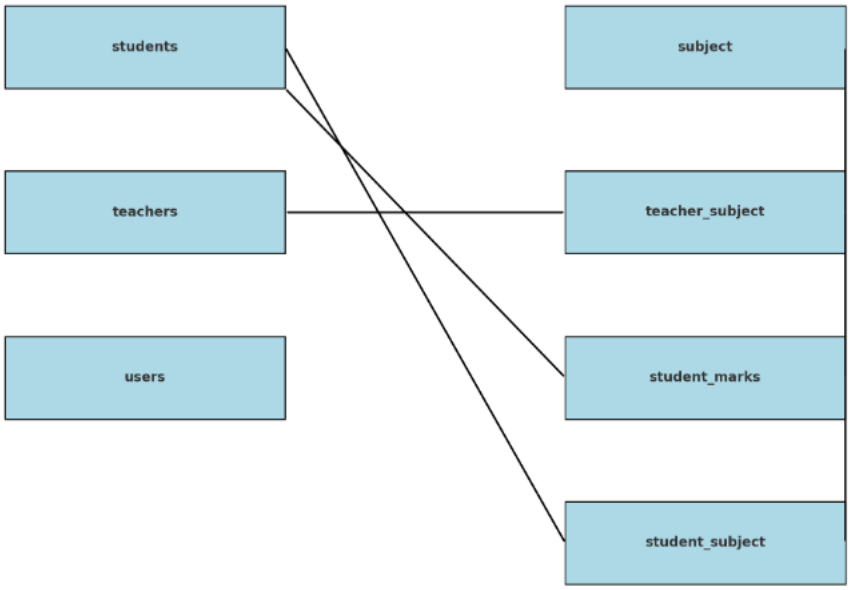


Рис. 2.1. Міні-схема таблиць бази даних інформаційної системи

Схема даних містить такі таблиці:

- Таблиця **students** зберігає інформацію про учнів.
- Таблиця **teachers** зберігає інформацію про вчителів.
- Таблиця **users** містить дані користувачів системи.
- Таблиця **subject** містить інформацію про предмети.
- Таблиця **teacher_subject** зв'язує вчителів з предметами, які вони викладають.
- Таблиця **student_marks** містить оцінки учнів.
- Таблиця **student_subject** зв'язує учнів з предметами, які вони вивчають.

Зв'язки між таблицями вказані стрілками на схемі:

- Учні зв'язані з таблицею предметів через таблицю **student_subject**.
- Учні зв'язані з таблицею оцінок через таблицю **student_marks**.
- Вчителі зв'язані з таблицею предметів через таблицю **teacher_subject**.
- Таблиця **teacher_subject** зв'язана з таблицею оцінок учнів.
- Таблиця предметів зв'язана з таблицями **teacher_subject** та **student_subject**.

Ця схема демонструє основні зв'язки між таблицями, що дозволяють зберігати та обробляти дані в системі управління школою

Ця модель відображає взаємозв'язки між таблицями та забезпечує структуру для зберігання інформації про учнів, класи, вчителів, предмети та оцінки. Кожна таблиця має свій унікальний ідентифікатор (первинний ключ), а зв'язки між ними реалізовані через зовнішні ключі.

База даних системи

База даних (БД) – це організований набір даних, що зберігаються і управляються за допомогою системи управління базами даних (СУБД). Основними завданнями бази даних є зберігання, управління та обробка великих обсягів інформації. Для шкільної інформаційної системи база даних повинна забезпечувати ефективне зберігання та доступ до інформації про учнів, вчителів, класи, предмети та оцінки.

Проектування бази даних включає декілька етапів:

- **Визначення вимог до даних:** Збір та аналіз вимог до інформації, яку повинна містити база даних. Це включає визначення основних сутностей, їх атрибутів та зв'язків між ними.
- **Розробка логічної моделі даних:** Створення схеми, що відображає сутності, їх атрибути та зв'язки між ними.
- **Нормалізація:** Процес оптимізації структури бази даних для усунення надлишкових даних і забезпечення цілісності.
- **Розробка фізичної моделі даних:** Перетворення логічної моделі в конкретну схему бази даних для вибраної СУБД. Це включає створення таблиць, визначення первинних і зовнішніх ключів, індексів тощо.

База даних системи написана на мові SQL(Structured Query Language). Використання SQL має безліч переваг, серед яких:

1. **Універсальність**: SQL є стандартною мовою для роботи з реляційними базами даних. Багато СУБД, включаючи MySQL, PostgreSQL, SQLite, Microsoft SQL Server та Oracle, підтримують SQL, що робить його універсальним інструментом для роботи з даними.

2. **Простота та зручність**: SQL має простий синтаксис, який дозволяє легко створювати, модифікувати та оптимізувати запити до баз даних. Користувачі можуть швидко вивчати основи SQL та ефективно використовувати його для роботи з даними.
3. **Ефективність**: SQL дозволяє виконувати операції з даними швидко та ефективно. Відповідні індекси та оптимізація запитів можуть значно покращити продуктивність баз даних.
4. **Масштабованість**: SQL бази даних можуть бути легко масштабовані вгору або вниз залежно від потреб вашого проекту. Вони можуть обробляти великі обсяги даних та взаємодіяти з великою кількістю користувачів одночасно.
5. **Надійність**: Багато SQL СУБД мають вбудовані механізми забезпечення цілісності даних, резервного копіювання та відновлення, що робить їх досить надійними для зберігання важливої інформації.
6. **Безпека**: SQL дозволяє встановлювати права доступу до даних для різних користувачів та ролей, що забезпечує високий рівень безпеки для вашої інформації.

Таблиці бази даних зображено на рисунку 2.2.

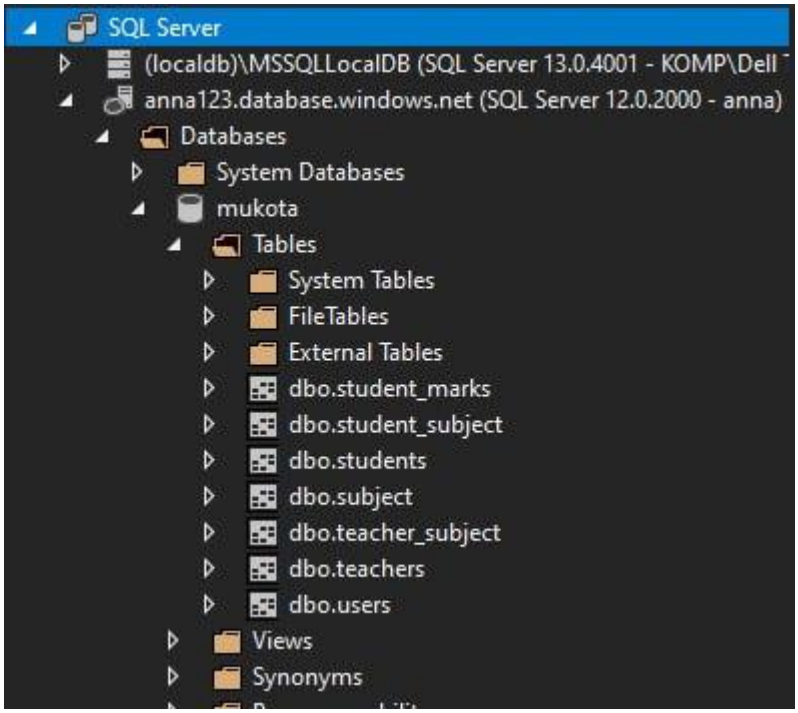


Рис. 2.2 Таблиці бази даних

Таблиця "students", яка зображена на рисунку 2.2, створена за допомогою наступного SQL запиту:

```
CREATE TABLE [dbo].[students] (
    [id]              INT           IDENTITY (1, 1) NOT NULL,
    [student_id]      NVARCHAR (50)  NULL,
    [student_name]    NVARCHAR (100) NULL,
    [student_gender]  NVARCHAR (10)  NULL,
    [student_address] NVARCHAR (255) NULL,
    [student_grade]   NVARCHAR (10)  NULL,
    [student_section] NVARCHAR (50)  NULL,
    [student_image]   NVARCHAR (255) NULL,
    [student_status]  NVARCHAR (50)  NULL,
    [date_insert]     DATETIME       NULL,
    [date_delete]     DATETIME       NULL,
    [student_password] NVARCHAR (15)  NULL,
    PRIMARY KEY CLUSTERED ([id] ASC),
    CONSTRAINT [CK_students_date] CHECK ([date_delete]>=[date_insert])
);
```

Таблиця "teachers", яка зображена на рисунку 2.2, створена за допомогою наступного SQL запиту:

```
CREATE TABLE [dbo].[teachers] (
    [id]              INT           IDENTITY (1, 1) NOT NULL,
    [teacher_name]    NVARCHAR (100) NULL,
    [teacher_gender]  NVARCHAR (10)  NULL,
    [teacher_address] NVARCHAR (255) NULL,
    [teacher_image]   NVARCHAR (255) NULL,
    [teacher_status]  NVARCHAR (50)  NULL,
    [date_insert]     DATETIME       NULL,
    [date_delete]     DATETIME       NULL,
```

```
    [date_update]     DATETIME       NULL,
    [teacher_password] NVARCHAR (15)  NULL,
    [teacher_id]      INT            NULL,
    PRIMARY KEY CLUSTERED ([id] ASC)
);
```

Таблиця "users", яка зображена на рисунку 2.2, створена за допомогою наступного SQL запиту:

```
CREATE TABLE [dbo].[users] (
    [id]       INT           IDENTITY (1, 1) NOT NULL,
    [username] NVARCHAR (50) NULL,
    [password] NVARCHAR (50) NULL,
    PRIMARY KEY CLUSTERED ([id] ASC)
);
```

Таблиця "subject", яка зображена на рисунку 2.2, створена за допомогою наступного SQL запиту:

```
CREATE TABLE [dbo].[subject] (
    [id]   INT           IDENTITY (1, 1) NOT NULL,
    [Name] NVARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([id] ASC)
);
```

Таблиця "teacher_subject", яка зображена на рисунку 2.2, створена за допомогою наступного SQL запиту:

```
CREATE TABLE [dbo].[teacher_subject] (
    [id]         INT IDENTITY (1, 1) NOT NULL,
    [teacher_id] INT NOT NULL,
    [subject_id] INT NOT NULL,
    PRIMARY KEY CLUSTERED ([id] ASC)
);
```

Таблиця "student_marks", яка зображена на рисунку 2.2, створена за допомогою наступного SQL запиту:

```
CREATE TABLE [dbo].[student_marks] (
    [Id]              INT       IDENTITY (1, 1) NOT NULL,
    [teacher_subject_id] INT       NOT NULL,
    [mark]            INT       NOT NULL,
    [data]            DATE      NOT NULL,
    [student_id]      NCHAR (10) NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Таблиця "student_subject", яка зображена на рисунку 2.2, створена за допомогою наступного SQL запиту:

```
CREATE TABLE [dbo].[student_subject] (
    [id]        INT IDENTITY (1, 1) NOT NULL,
    [student_id] INT NOT NULL,
    [subject_id] INT NOT NULL,
    PRIMARY KEY CLUSTERED ([id] ASC)
);
```

В цілому, використання баз даних на мові SQL дозволяє ефективно та надійно керувати даними, що робить їх незамінним інструментом для багатьох програмних проектів.

2.2. Організація введення та виведення інформації

У сучасних інформаційних системах для освіти організація введення та виведення інформації відіграє ключову роль в ефективності їх роботи. Від правильного налаштування цих процесів залежить зручність користування системою, точність даних та швидкість отримання необхідної інформації.

Проведене опитування серед учасників освітнього процесу, включаючи учнів, вчителів та адміністрацію школи, дозволили виявити низку важливих побажань щодо функціональності інформаційної системи. Зокрема, зі сторони учнів було висловлено наступні побажання:

Зі сторони учня:

- **Можливість бачити свої оцінки та викладачів, які їх ставлять:** Учні вважають за необхідне мати постійний доступ до своїх оцінок, щоб мати змогу слідкувати за своїм прогресом та своєчасно реагувати на можливі проблеми у навчанні. Це дозволить учням краще розуміти свої сильні та слабкі сторони, а також планувати свої навчальні зусилля. Крім того, можливість бачити, які саме викладачі поставили ті чи інші оцінки, допомагає учням отримувати зворотній зв'язок від конкретних викладачів. Це сприяє більш ефективній комунікації між учнями та викладачами та покращує взаєморозуміння.

Зі сторони вчителя:

- **Додавання оцінок учням, в яких вони ведуть певний предмет:** Вчителі зазначили важливість можливості оперативно додавати оцінки учням з предметів, які вони викладають. Цей функціонал дозволить викладачам швидко та зручно фіксувати результати поточного та підсумкового оцінювання, що сприятиме підтриманню актуальної та достовірної інформації про успішність учнів. Впровадження такої функції дозволить вчителям ефективніше управляти навчальним процесом та своєчасно інформувати учнів і їхніх батьків про досягнення та прогалини у навчанні.

Таким чином, реалізація зазначених побажань забезпечить створення ефективної та прозорої інформаційної системи для школи, що сприятиме покращенню успішності учнів та підвищенню ефективності навчального процесу.

Для забезпечення зручності та ефективності інтерфейсу користувача було використано технологію WindowsForms. Windows Forms - це технологія розробки графічного інтерфейсу користувача для програм на платформі Microsoft Windows. Вона дозволяє створювати різноманітні елементи інтерфейсу, такі як кнопки, текстові поля, списки, таблиці та інші, і керувати їх поведінкою за допомогою обробників подій. Використання WindowsForms в нашій інформаційній системі забезпечує такі переваги:

1. **Зручність в розробці:** WindowsForms має простий і зрозумілий API, що дозволяє легко створювати інтерфейсні елементи та взаємодіяти з ними.
2. **Широкі можливості керування елементами:** За допомогою WindowsForms можна забезпечити різноманітні можливості керування елементами інтерфейсу, включаючи стилізацію, валідацію введених даних, обробку подій тощо.
3. **Сумісність з платформою Windows:** Оскільки WindowsForms розроблено для платформи Windows, інтерфейс користувача буде виглядати і працювати оптимально на операційних системах сімейства Windows.
4. **Розширення можливостей за допомогою компонентів сторонніх виробників:** У разі потреби можна використовувати сторонні компоненти та бібліотеки для розширення можливостей WindowsForms.

Отже, використання технології WindowsForms дозволить нам створити зручний та інтуїтивно зрозумілий інтерфейс користувача для нашої інформаційної системи, що сприятиме зручній та ефективній роботі з нею.

Платформа Microsoft Azure є однією з провідних хмарних платформ, що надає широкий спектр послуг для зберігання, обробки та аналізу даних. Microsoft Azure була обрана як платформа для розміщення бази даних з кількох причин:

- **Надійність та масштабованість**: Azure надає можливість масштабування ресурсів відповідно до потреб додатку, що забезпечує стабільну роботу навіть при зростанні навантаження.

- **Безпека даних**: Azure пропонує високий рівень безпеки, включаючи шифрування даних, захист від DDoS-атак та інші заходи безпеки.
- **Інтеграція з іншими сервісами:** Azure легко інтегрується з іншими продуктами Microsoft, такими як Office 365, що може бути корисним для шкільної інформаційної системи.
- **Гнучкість у виборі баз даних:** Платформа підтримує різні типи баз даних, включаючи SQL Database, Cosmos DB, PostgreSQL, MySQL та інші, що дозволяє обрати найбільш підходящий варіант. Організація введення та виведення інформації в інформаційній системі школи є важливим аспектом, що впливає на ефективність та зручність використання системи.

Використання сучасних технологій, таких як хмарні рішення Azure, дозволяє забезпечити високу надійність, безпеку та доступність даних.

Реалізовані методи введення та виведення інформації сприяють точності даних та зручності їх обробки, що в цілому покращує якість управління навчальним процесом та ефективність роботи навчального закладу.

2.3. Функціональні можливості інформаційної системи

Під час розробки інформаційної системи для школи було створено програму School Management System, яка забезпечує комплексну автоматизацію та управління навчальним процесом. Основні функціональні можливості системи включають введення, збереження, редагування та виведення даних, що дозволяє ефективно організовувати роботу школи та забезпечувати зручний доступ до необхідної інформації для всіх учасників освітнього процесу.

1. Login – вхід в систему школи. Логін в будь-яку програму або сервіс використовується для ідентифікації користувача та надання йому доступу до певного функціоналу чи ресурсів. Вигляд форми при запусканні програми на рисунку 2.3.

Рис. 2.3 Форма при завантажені програми

Після завантаження програми, відкривається вікно входу у систему. Дана форма містить поле username та password на рисунку 2.4.

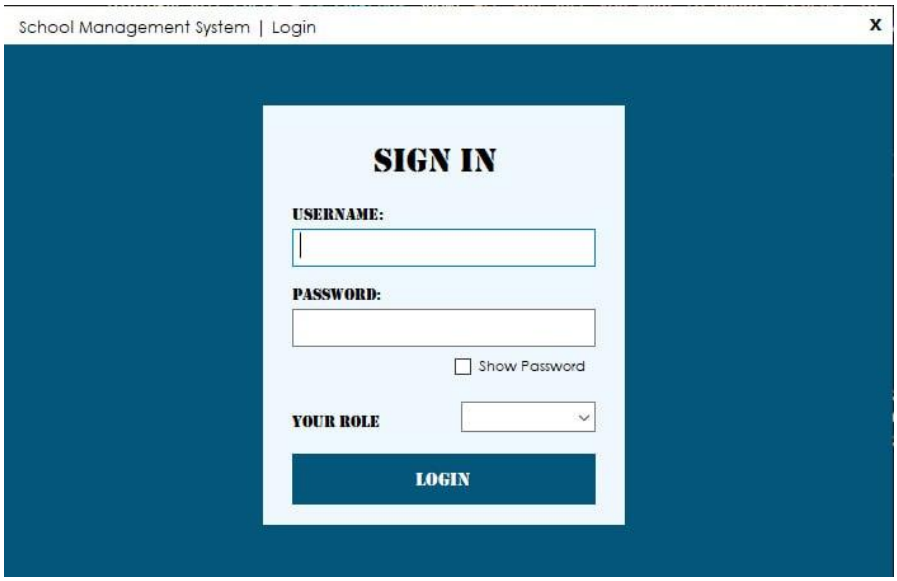


Рис. 2.4 Форма входу в систему

Користувач вводить свої дані та повинен обрати свою роль при вході - вчитель обирає – teacher, учень – student, адміністратор – administrator і відповідно вводять свій логін та пароль на рисунку 2.5.

Рис. 2.5 Вхід до системи

Вхід до системи як адміністратор (administrator)

Отже, заповнюємо дані поля, в даному випадку як адміністратор. Також реалізована функція показувати пароль на рисунку 2.7.

Рис. 2.7 Заповнення даних та можливість показати пароль

Після введенння всіх потрібних даних, натискаємо кнопку “Login” після чого користувач входить до системи на рисунку 2.8.

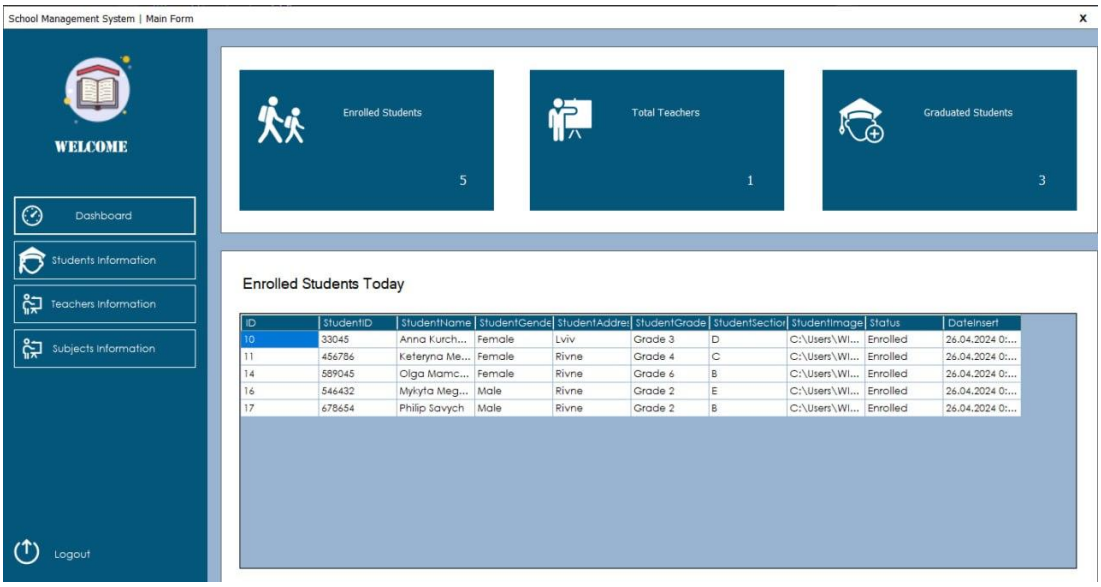


Рис. 2.8 Вигляд основного меню адміністратора

На головному меню для зручного користування зроблені такі дві панелі. Перша панель показує три різних інформаційних дошок -кількість студентів, які зараз навчаються. Виведена загальна кількість вчителів у системі. Та показано загальна кількість студентів які випустилися. Друга панель виводить список всіх студетів, які числяться у системі на сьогоднішній день.

2. Панель роботи з студентом. В данній формі можна додавати, редагувати та видаляти інформацію про студента на рисунку 2.9. Передбачена можливість очищення полів. Також у програмі реалізовано пошук. Та відразу ж оновлюється список студентів в разі їх додавання, редагування чи видалення.

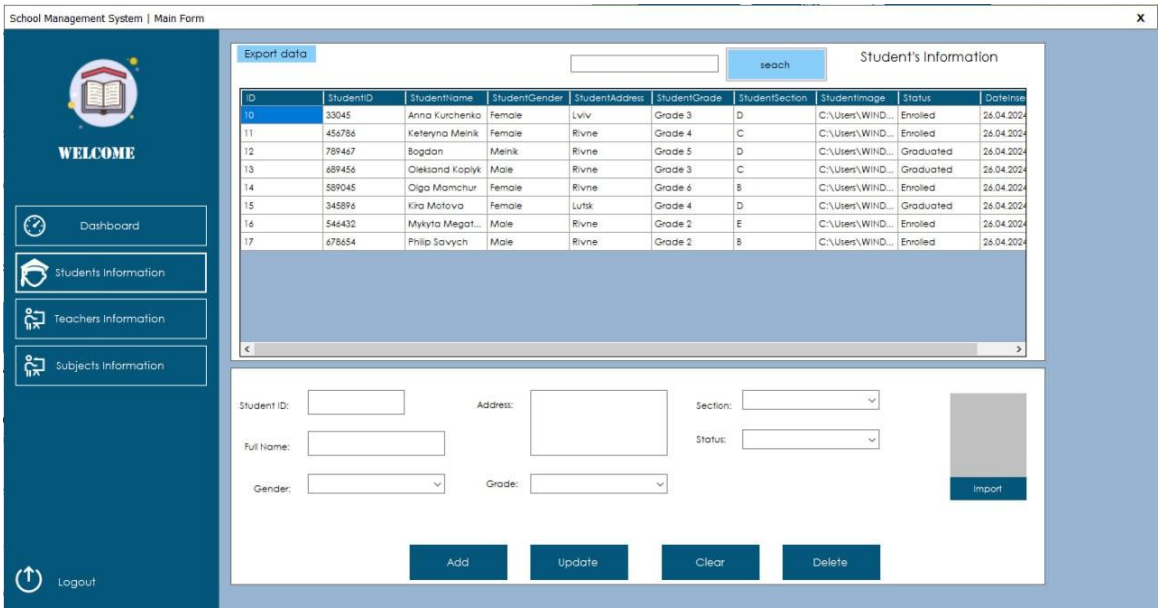


Рис. 2.9 Панель роботи зі студентами

- **Додавання нового студента**

Додаємо нового студента у список та відразу у базу даних. Заповнюємо всі поля та вибираємо фото студента на рисунку 2.10.

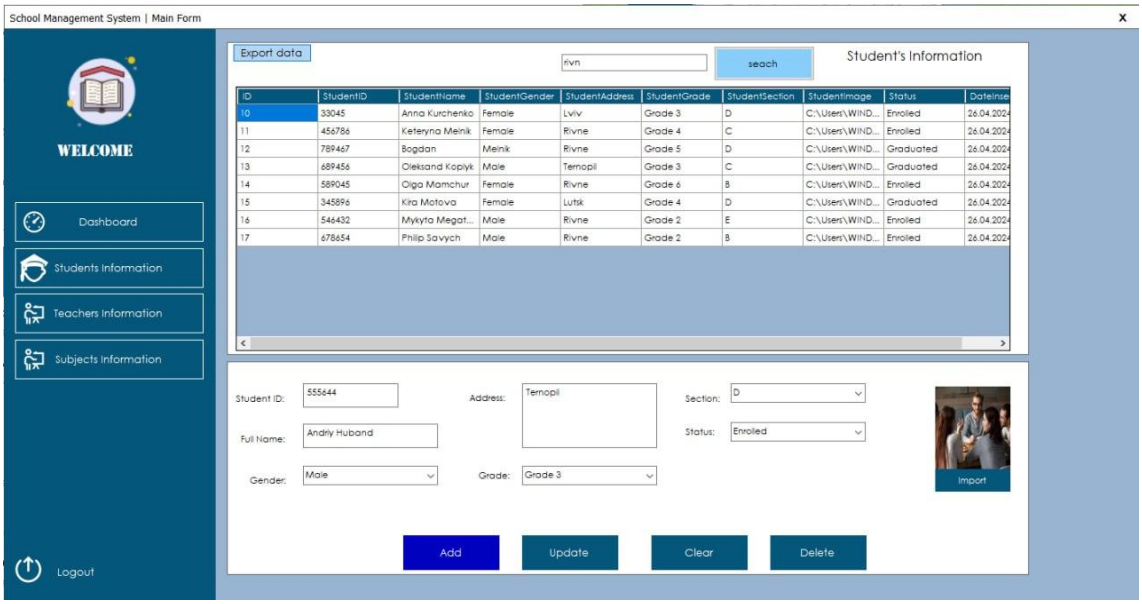


Рис. 2.10 Заповнення полів для додавання студента

Нажимаємо кнопку “ADD” – з’являється повідомлення про успішне додавання студента на рисунку 2.11.

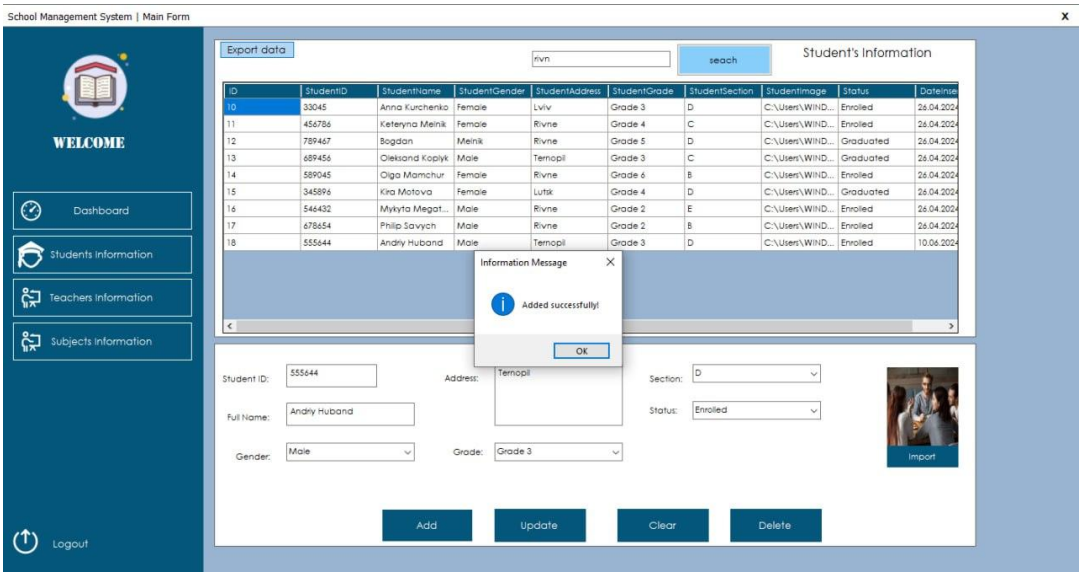


Рис. 2.11 Повідомлення про успішне додавання студента

Після натиснення кнопки “OK”, студент додається до списку основної таблиці, яка відразу оновлюється на рисунку 2.12.

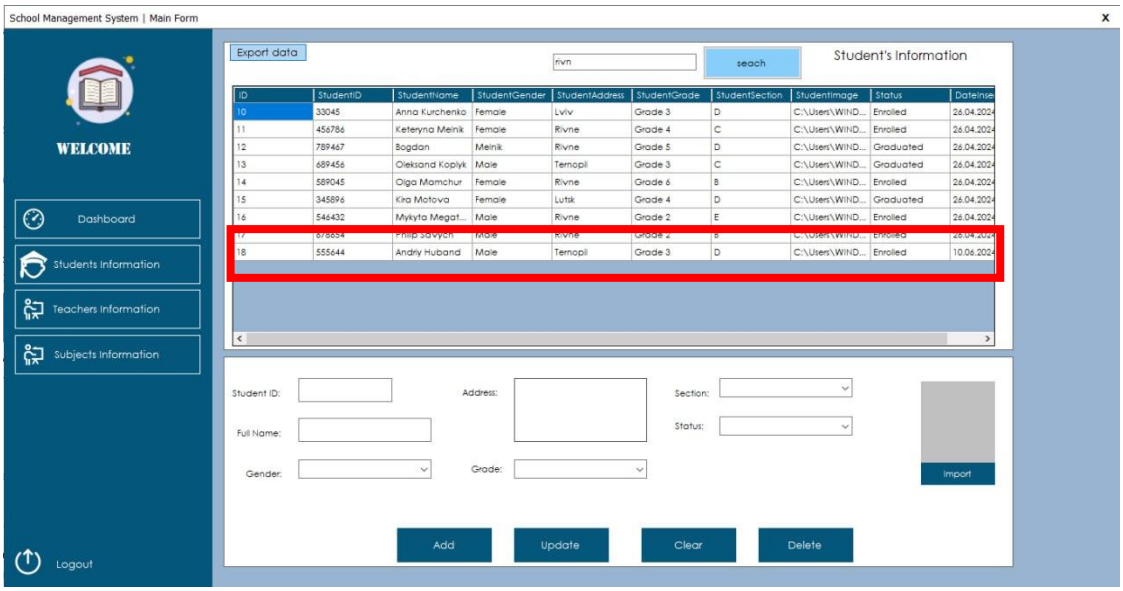


Рис. 2.12 Вигляд таблиці після додавання студента

- **Редагування студента**

Для редагування певного студента – клікаємо на будь-яке поле студента, якого хочемо редагувати на рисунку 2.13.

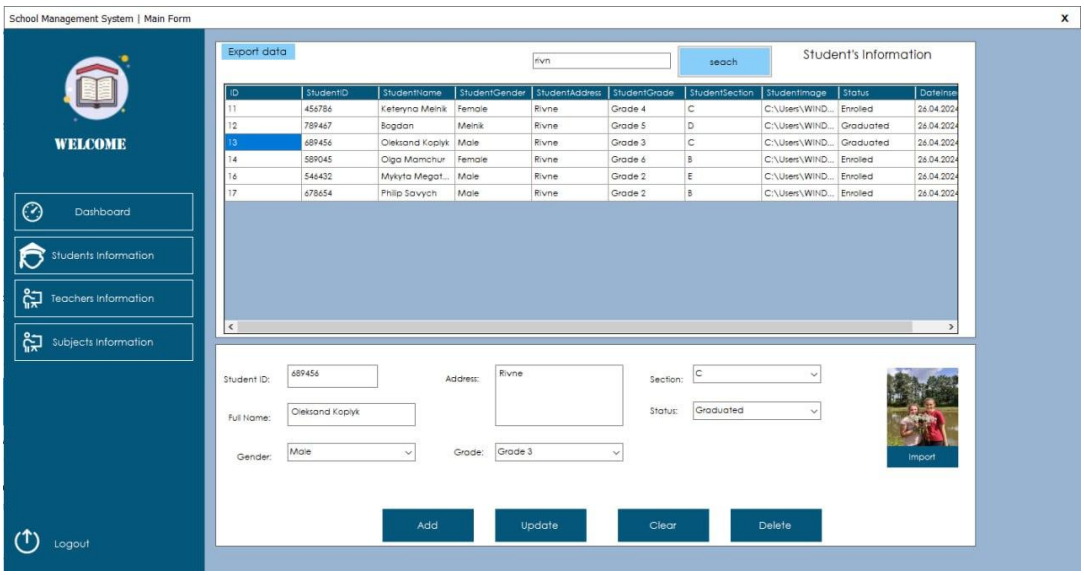
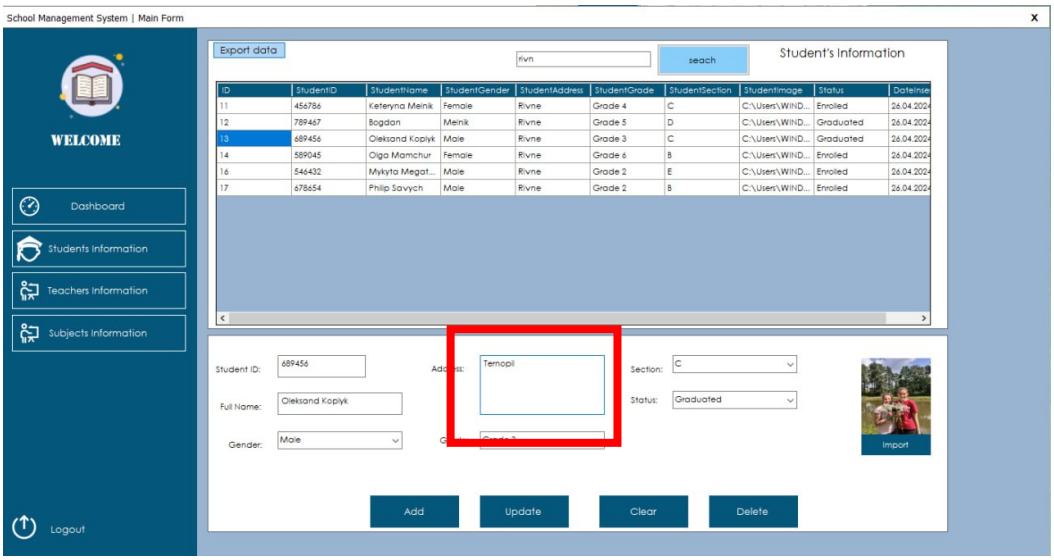


Рис. 2.13 Обираємо студента для редагування

Всі поля відриваються для редагування – змінюємо потрібний текст на рисунку 2.14.



Після чого натискаємо кнопку “Update” і з’являється повідомлення про підтвердження редагування даних на рисунку 2.15.

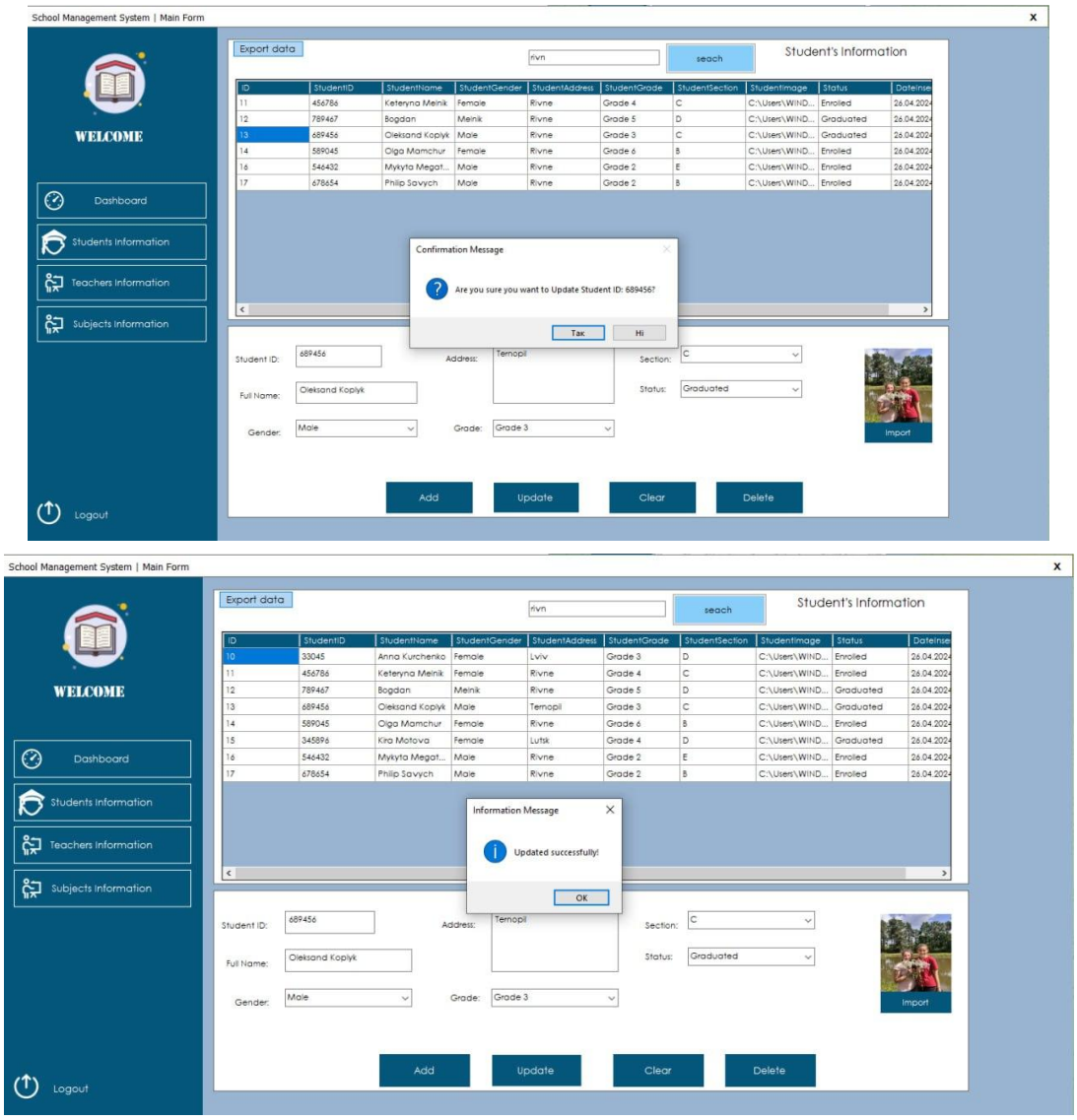


Рис. 2.15 Повідомлення про підтвердження редагування даних

Натискаємо “OK” та бачимо відразу результат в таблиці на рисунку 2.16.

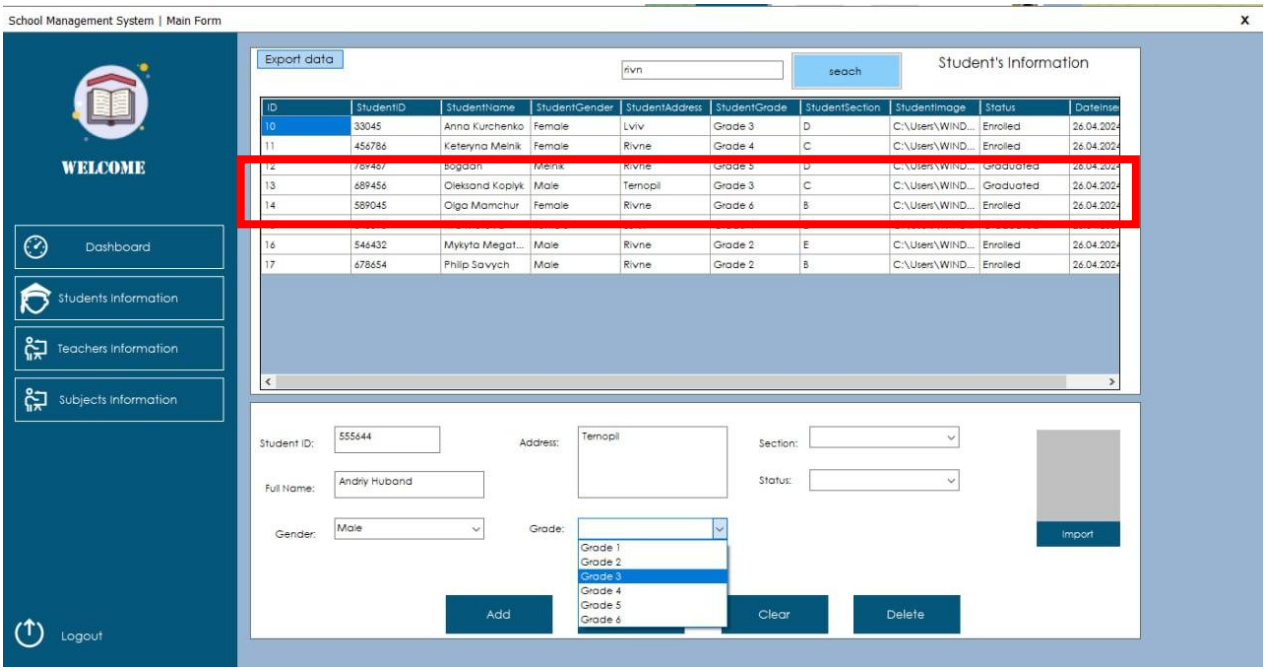


Рис. 2.16 Таблиця після редагування студента

- **Очищення всіх полів**

Для зручності користування полями, додано очищення всіх полів. Для тестування продемонструю заповнення поля – будь-якою непотрібною інформацією на рисунку 2.17.

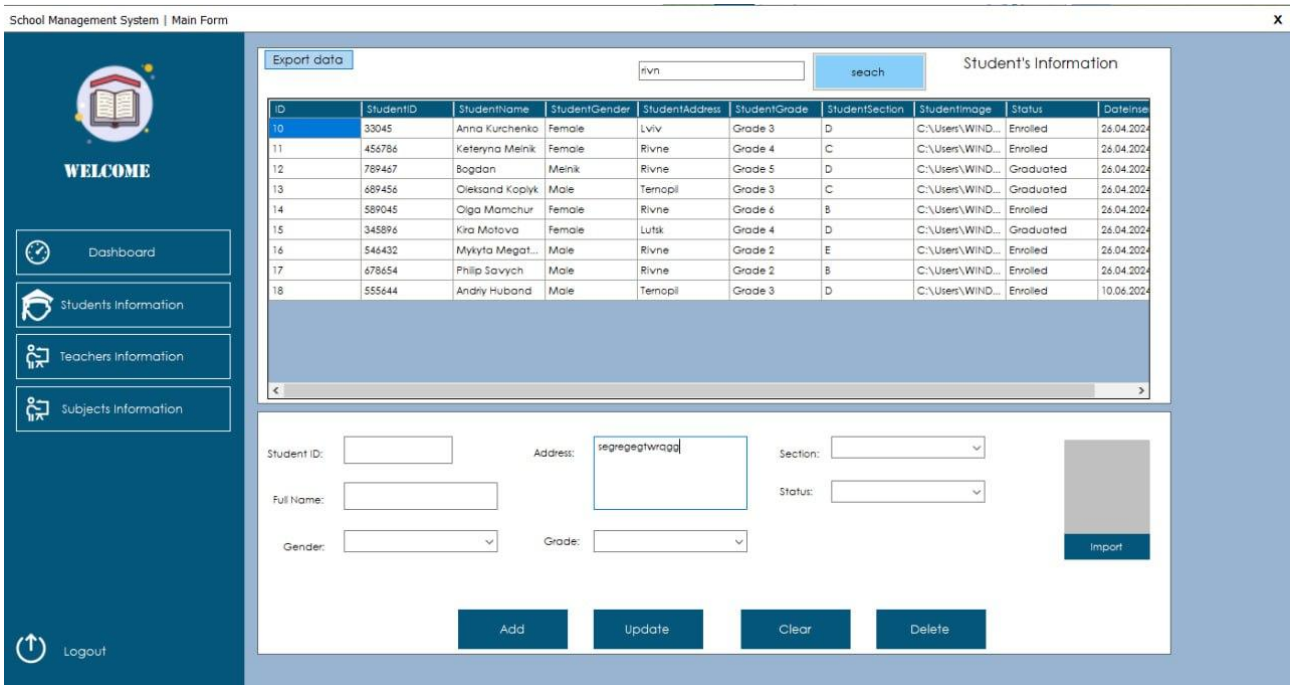


Рис. 2.17 Заповнення поля будь-якою інформацією

Натискаємо кнопку “Clear” та відразу бачимо всі поля чистими на рисунку 2.18.

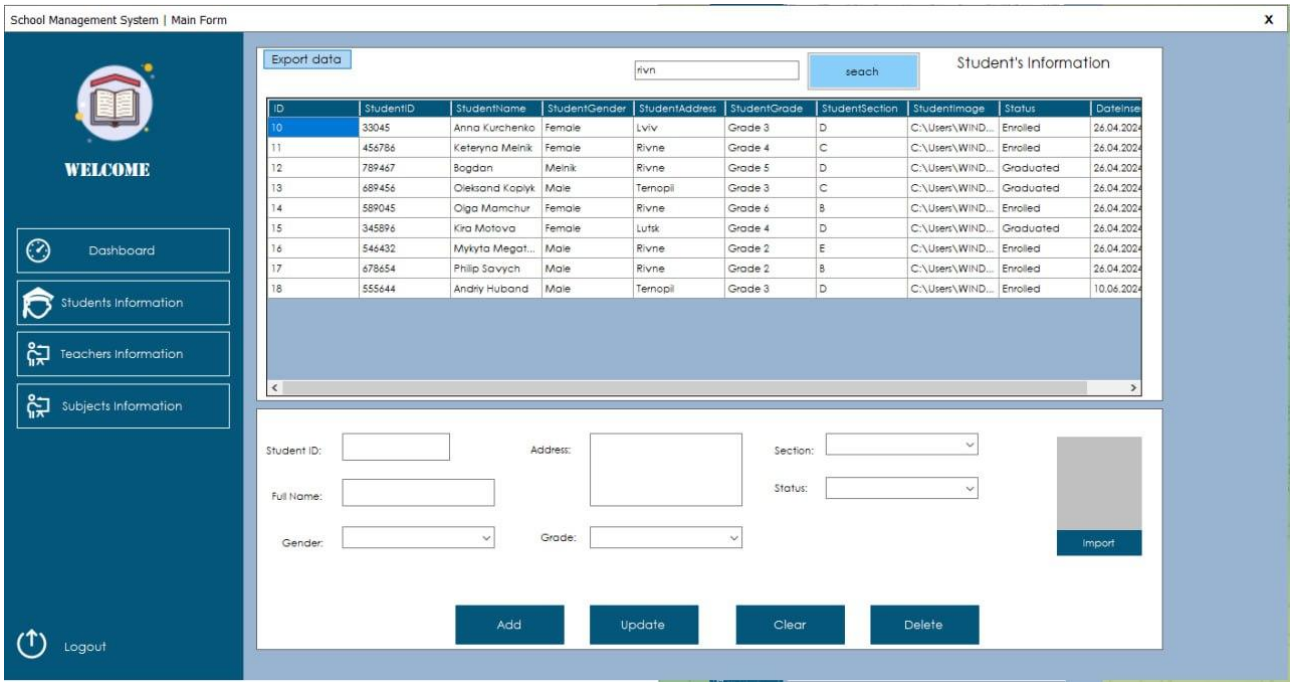


Рис 2.18 Вигляд форми після її очищення

- **Видалення студента**

Для видалення студента зі списку та бази – обираємо студента за допомою клацання на його ID на рисунку 2.19.

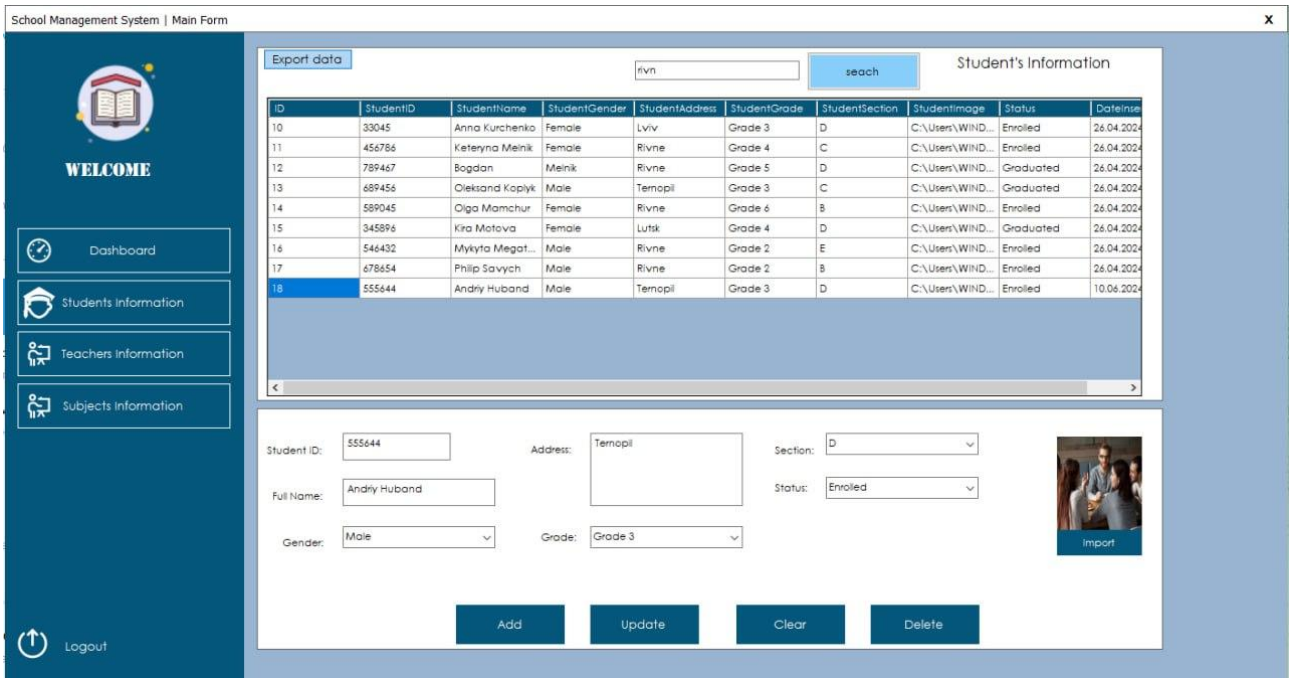


Рис 2.19 Вибираємо студента для його видалення

Натискаємо кнопку “Delete” і з’являється повідомлення про точне видалення студента зі списку на рисунку 2.20.

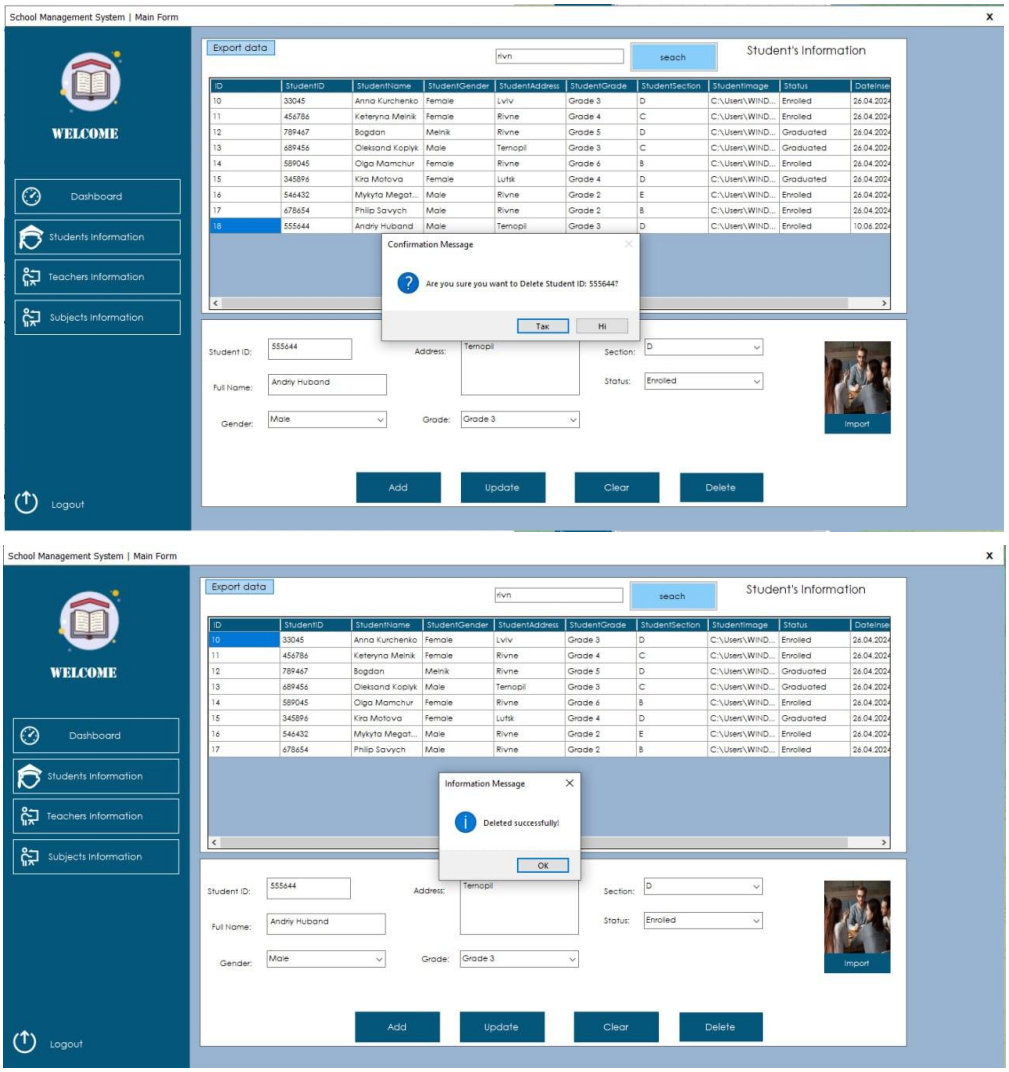


Рис. 2.20 Уточнення про видалення вчителя

Та в результаті, бачимо що даного студента в списку немає на рисунку 2.21.

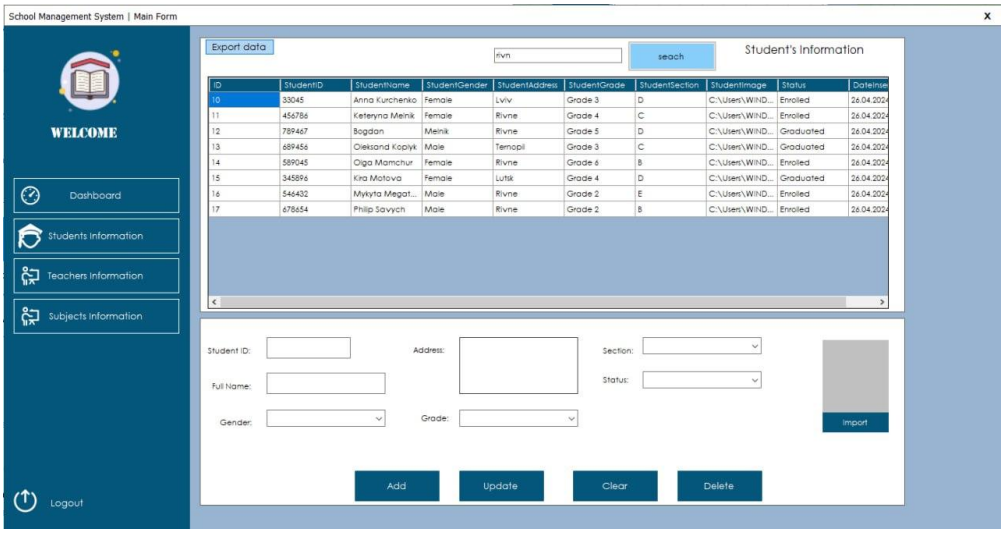


Рис. 2.21 Список, після успішного видалення студента

- **Пошук у списку студентів**

У програмі реалізовано пошук по всій таблиці. Вводимо текст для пошуку інформації та натискаємо кнопку “Search” - відразу бачимо результат в таблиці на рисунку 2.22.

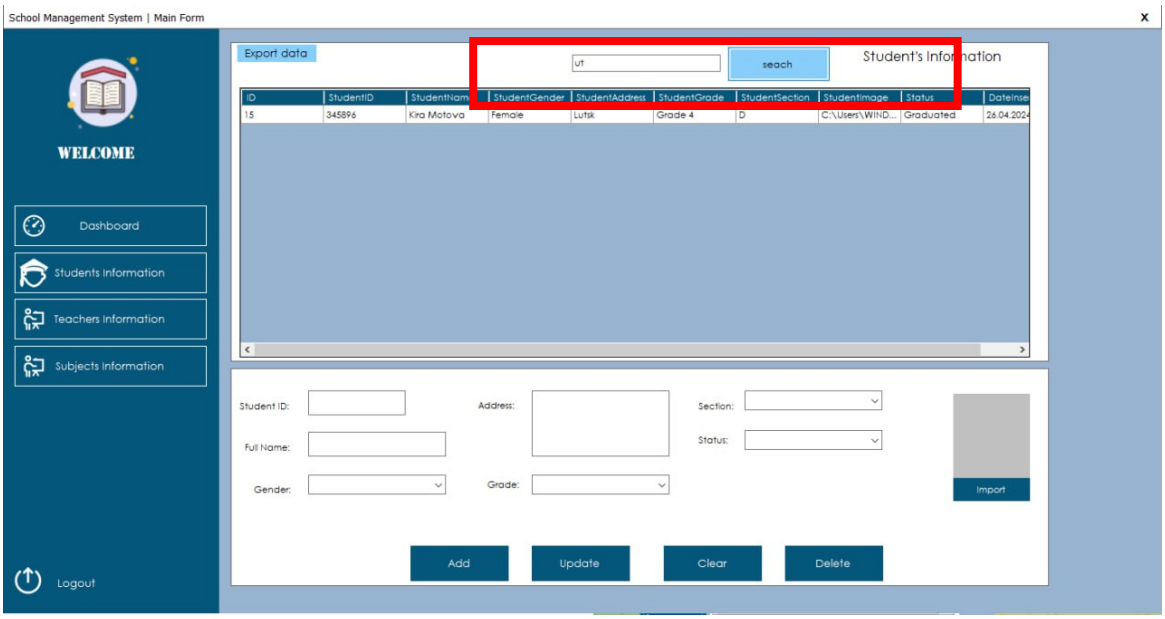


Рис. 2.22 Результат пошуку в таблиці

- **Експорт таблиці у файл.** У програмі додана можливість всю інформацію в таблиці експортувати у файл Word або Excel – на вибір адміністратора. Для створення файла натискаємо кнопку “Export”. Далі з’являється ще дві кнопки, де потрібно вибрати в який файл зробити експорт на рисунку 2.23.

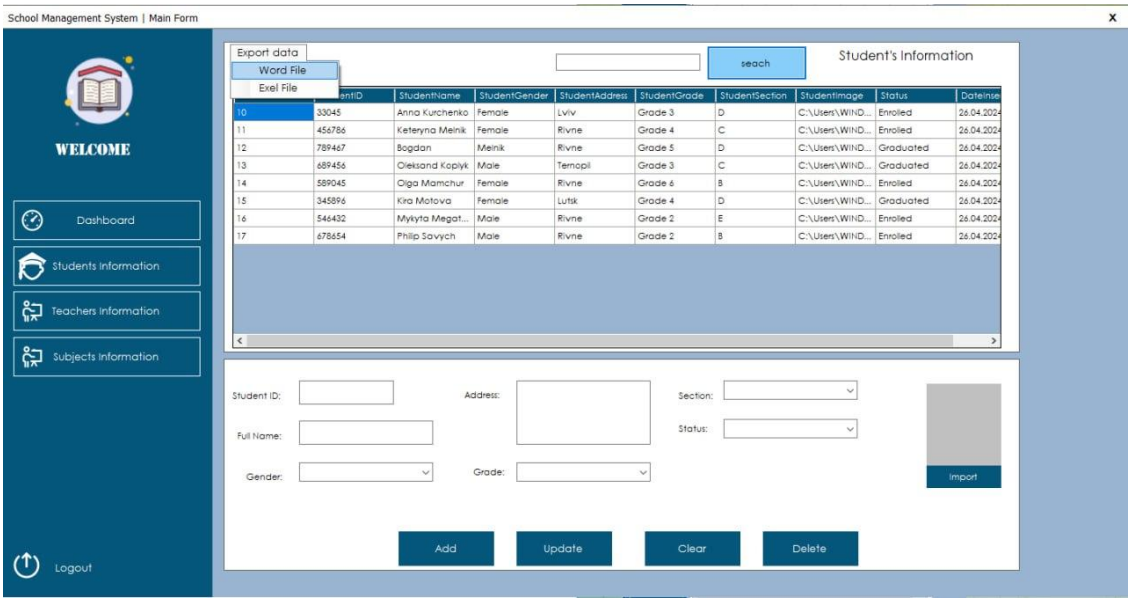


Рис. 2.23 Експорт даних

Вибираємо Word file, з’являється повідомлення про те що створився файл на рисунку 2.24.

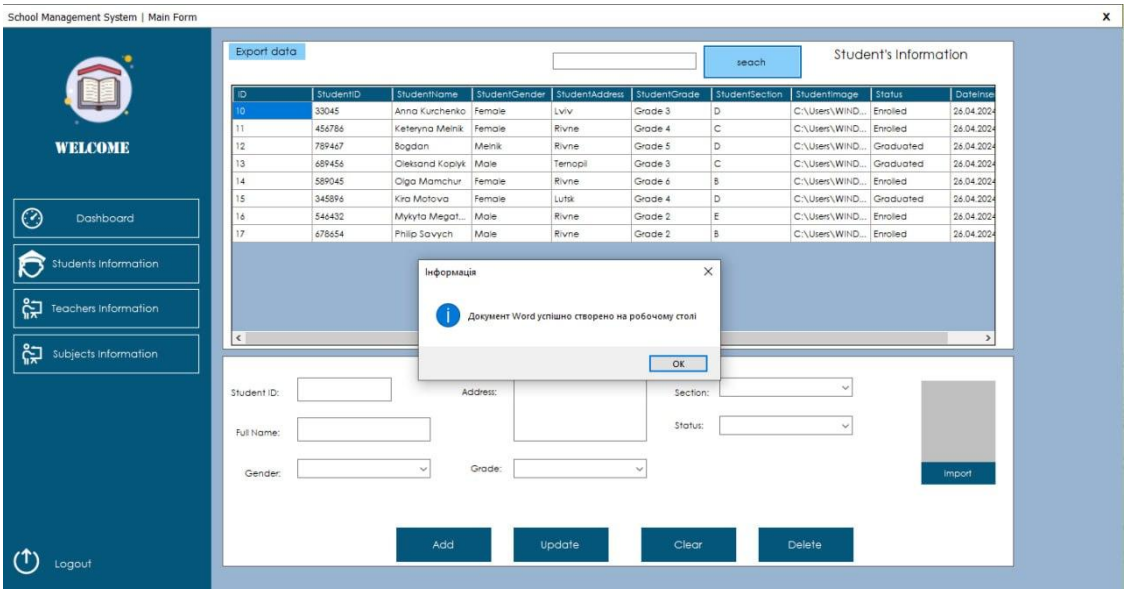
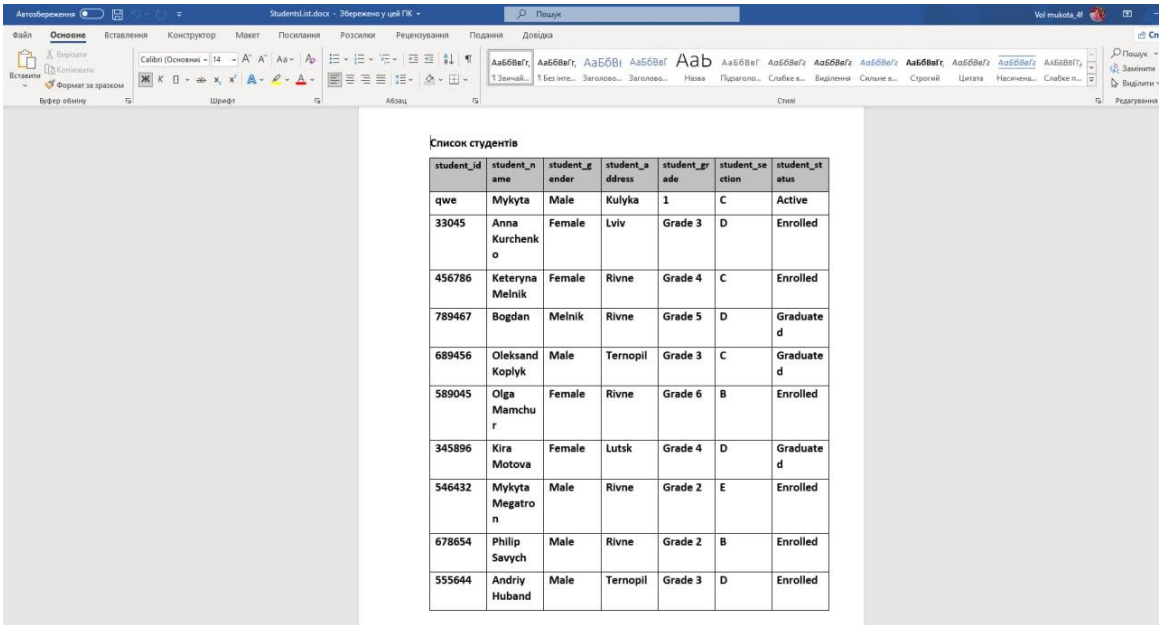


Рис. 2.24 Повідомлення про створення word file

Відриваємо файл, який створився на робочому столі на рисунку 2.25.



Список студентів

student_id	student_name	student_gender	student_address	student_grade	student_section	student_status
qwe	Mykyta	Male	Kulyka	1	C	Active
33045	Anna Kurchenko	Female	Lviv	Grade 3	D	Enrolled
456786	Keteryna Melnik	Female	Rivne	Grade 4	C	Enrolled
789467	Bogdan	Melnik	Rivne	Grade 5	D	Graduated
689456	Oleksand Koplyk	Male	Ternopil	Grade 3	C	Graduated
589045	Olga Mamchur	Female	Rivne	Grade 6	B	Enrolled
345896	Kira Motova	Female	Lutsk	Grade 4	D	Graduated
546432	Mykyta Megatron	Male	Rivne	Grade 2	E	Enrolled
678654	Philip Savych	Male	Rivne	Grade 2	B	Enrolled
555644	Andriy Huband	Male	Ternopil	Grade 3	D	Enrolled

Рис. 2.25 Результат експорту

Вибираємо Excel file, з’являється повідомлення про те що створився файл на рисунку 2.26.

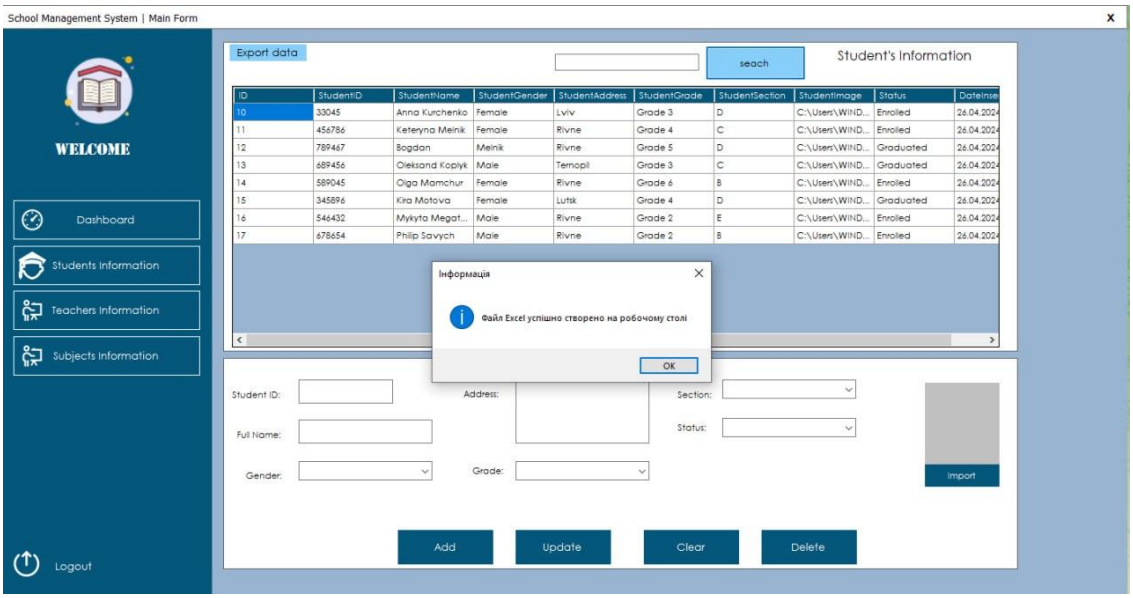
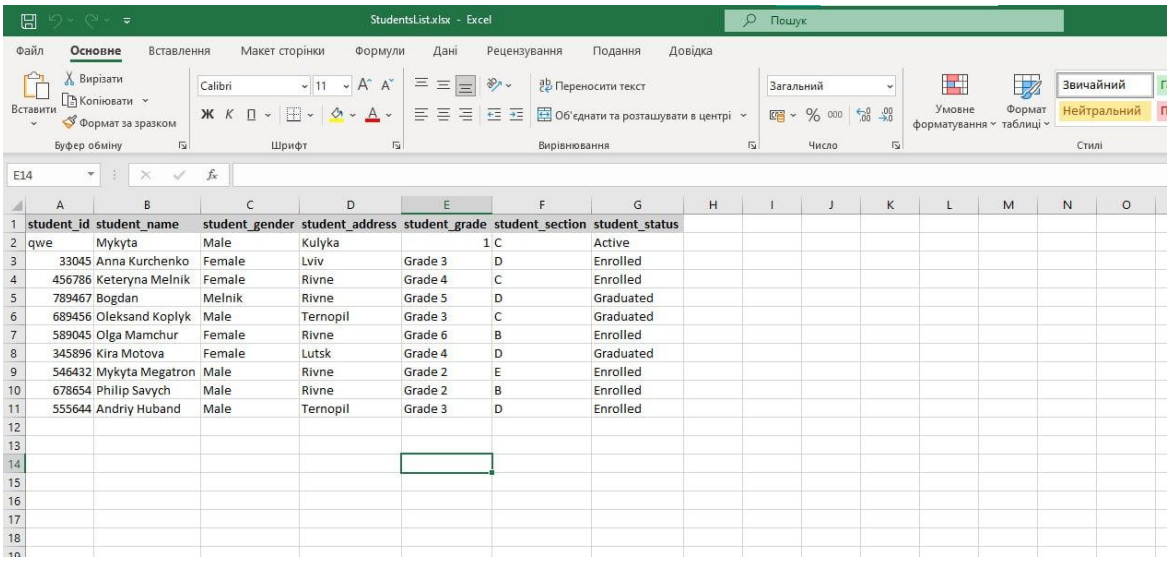


Рис. 2.26 Повідомлення про створення word file

Відриваємо файл, який створився на робочому столі на рисунку 2.27.



student_id	student_name	student_gender	student_address	student_grade	student_section	student_status
qwe	Mykyta	Male	Kulyka	1	C	Active
33045	Anna Kurchenko	Female	Lviv	Grade 3	D	Enrolled
456786	Keteryna Melnik	Female	Rivne	Grade 4	C	Enrolled
789467	Bogdan	Melnik	Rivne	Grade 5	D	Graduated
689456	Oleksand Koplyk	Male	Ternopil	Grade 3	C	Graduated
589045	Olga Mamchur	Female	Rivne	Grade 6	B	Enrolled
345896	Kira Motova	Female	Lutsk	Grade 4	D	Graduated
546432	Mykyta Megatron	Male	Rivne	Grade 2	E	Enrolled
678654	Philip Savych	Male	Rivne	Grade 2	B	Enrolled
555644	Andriy Huband	Male	Ternopil	Grade 3	D	Enrolled

Рис. 2.27 Результат експорту

3. Панель роботи з вчителями. В даній формі можна додавати, редагувати та видаляти вчителів на рисунку 2.28. Відразу ж оновлюється список вчителів в разі їх додавання чи видалення. Також передбачена можливість очищення полів.

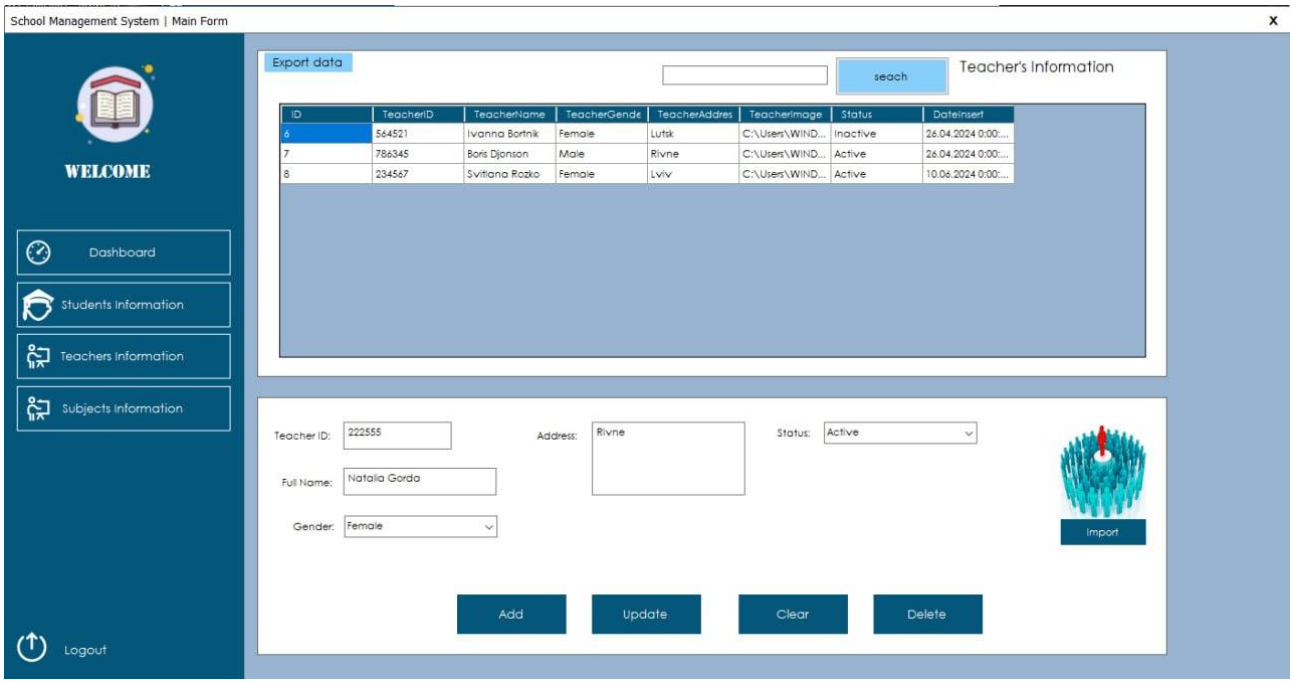


Рис 2.28 Форма роботи з вчителями

- Додавання вчителя

  Додаємо нового вчителя у список та відразу у базу даних. Заповнюємо всі поля та вибираємо фото вчителя на рисунку 2.29.

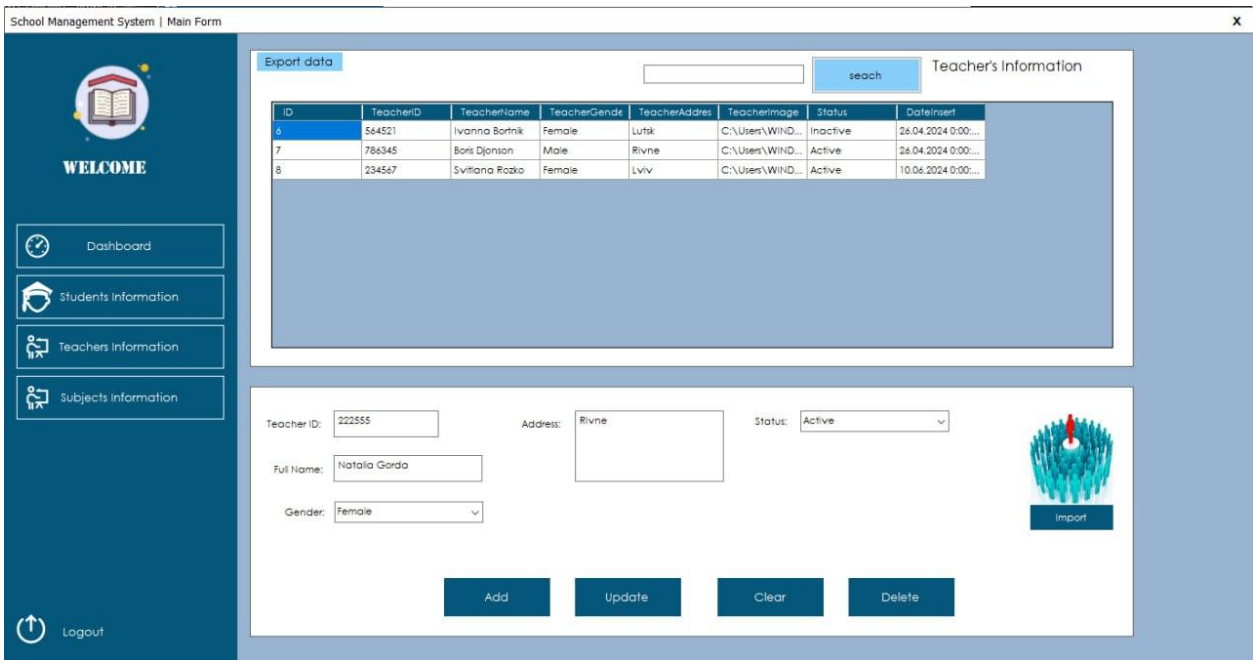


Рис 2.29 Заповнення всіх полів для довання вчителя у список

Нажимаємо кнопку “ADD” – з’являється повідомлення про успішне додавання студента на рисунку 2.30.

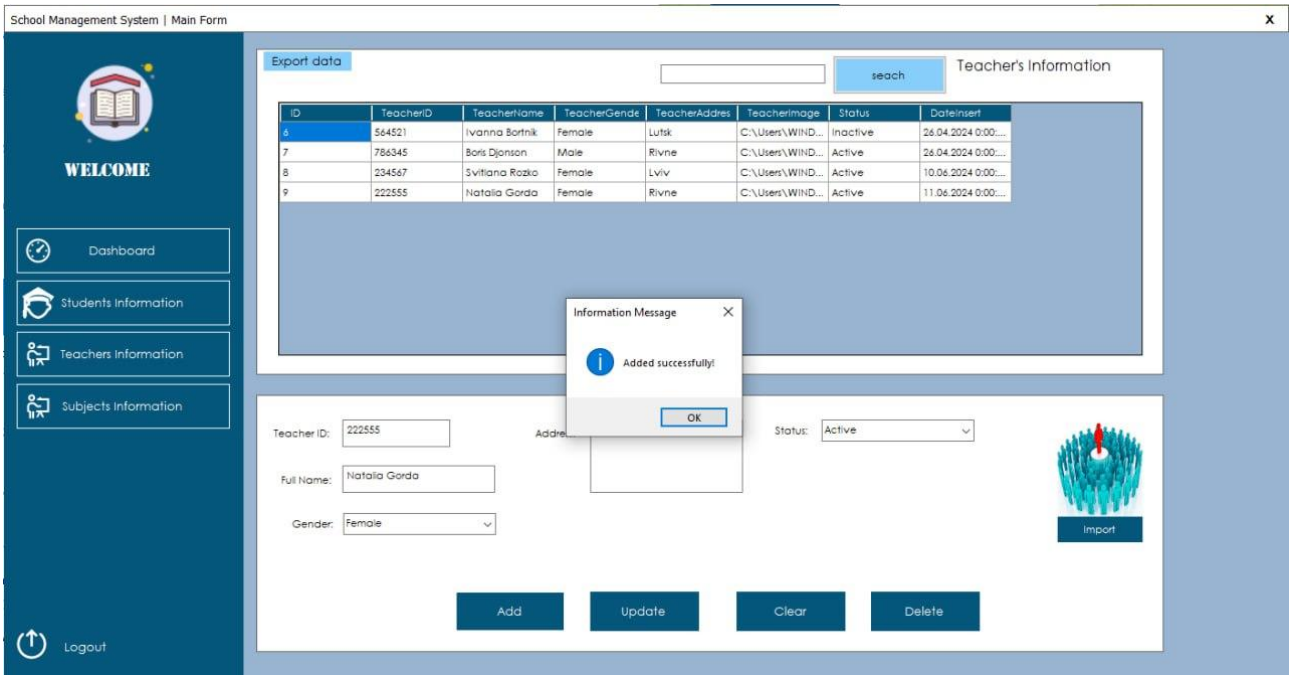


Рис. 2.30 Повідомлення про успішне додавання вчителя

Після натиснення кнопки “OK”, студент додається до списку основної таблиці, яка відразу оновлюється на рисунку 2.31.

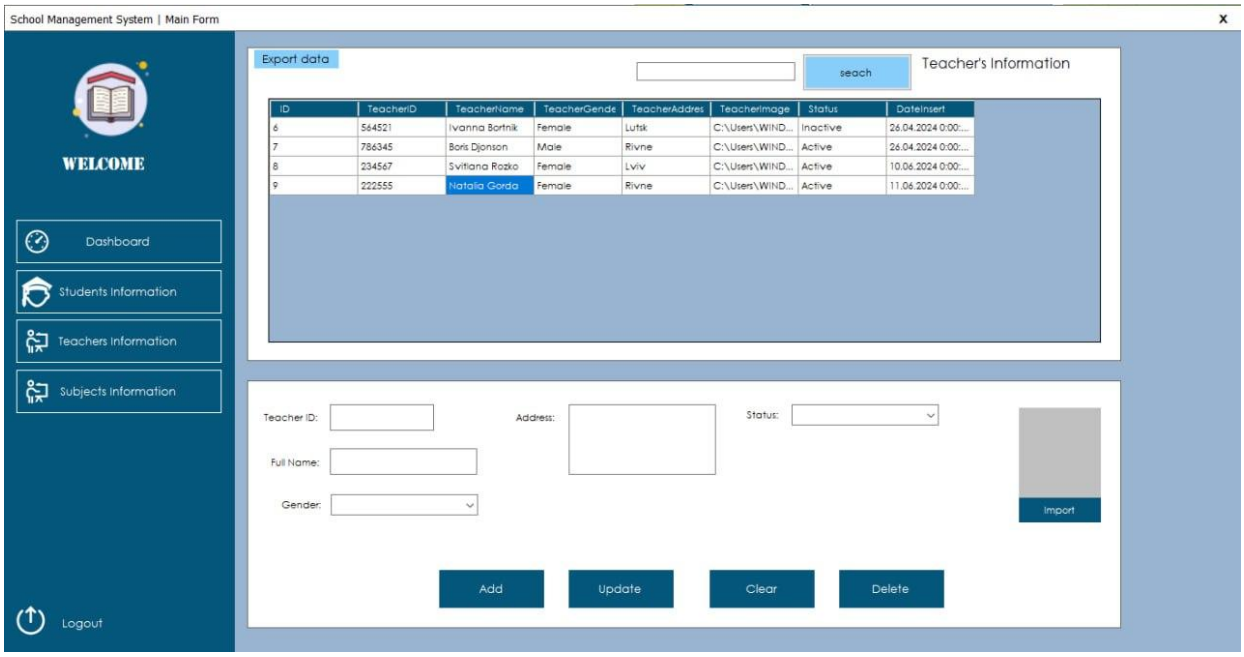


Рис. 2.31 Вигляд таблиці після додавання вчителя

- Редагування вчителя

  Для редагування певного вчителя – клікаємо на будь-яке його поле, якого хочемо редагувати на рисунку 2.32.

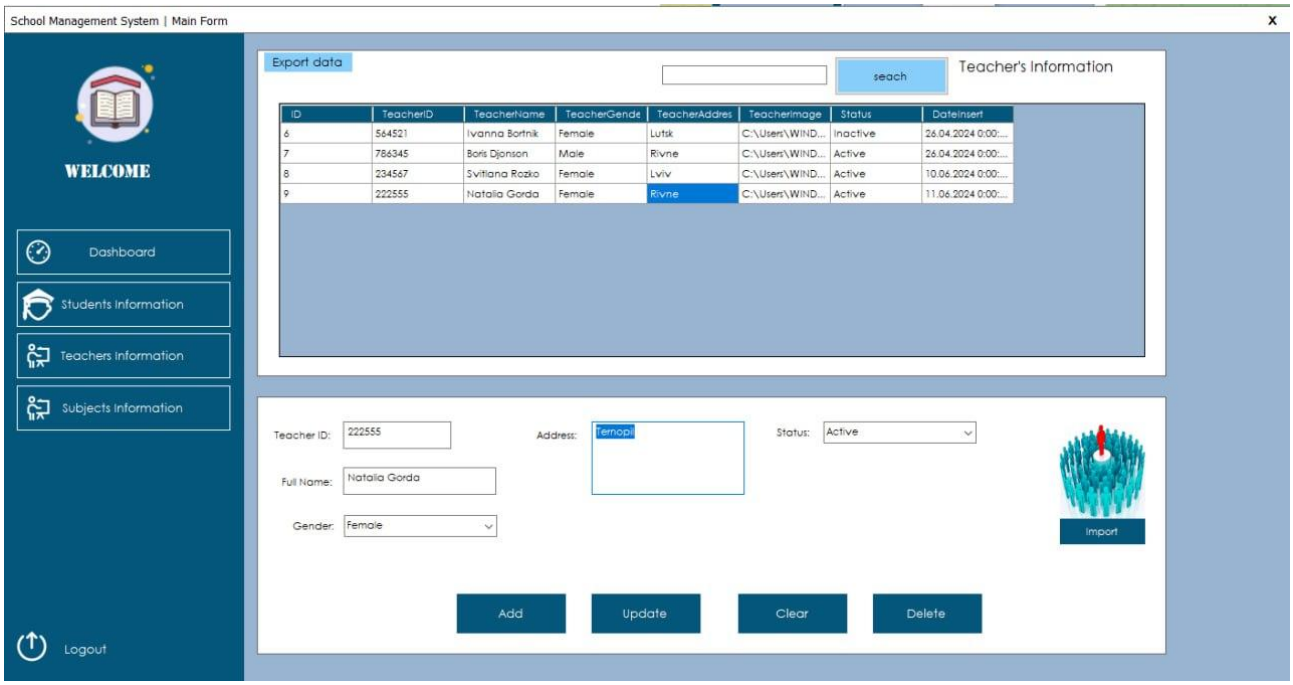


Рис. 2.32 Обираємо студента для редагування

Всі поля відриваються для редагування – змінюємо потрібний текст на рисунку 2.33.

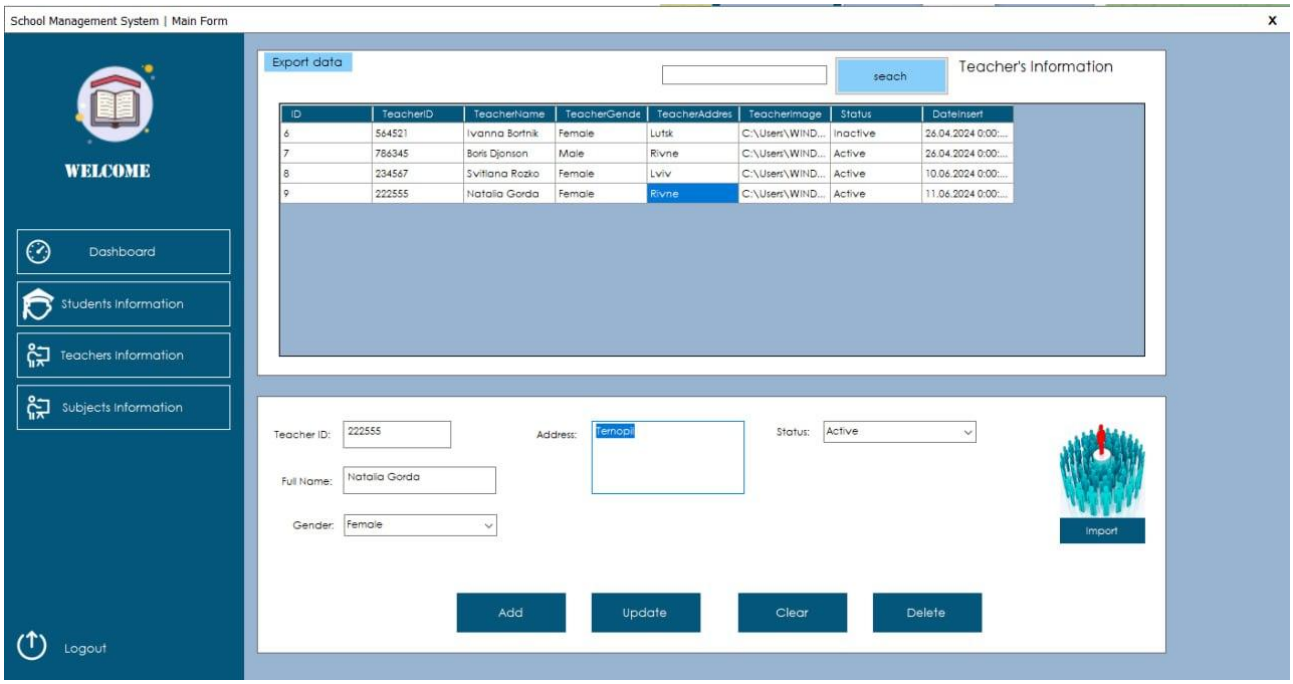


Рис. 2.33 Змінюємо потрібну інформацію

Після чого натискаємо кнопку “Update” і з’являється повідомлення про підтвердження редагування даних на рисунку 2.34.

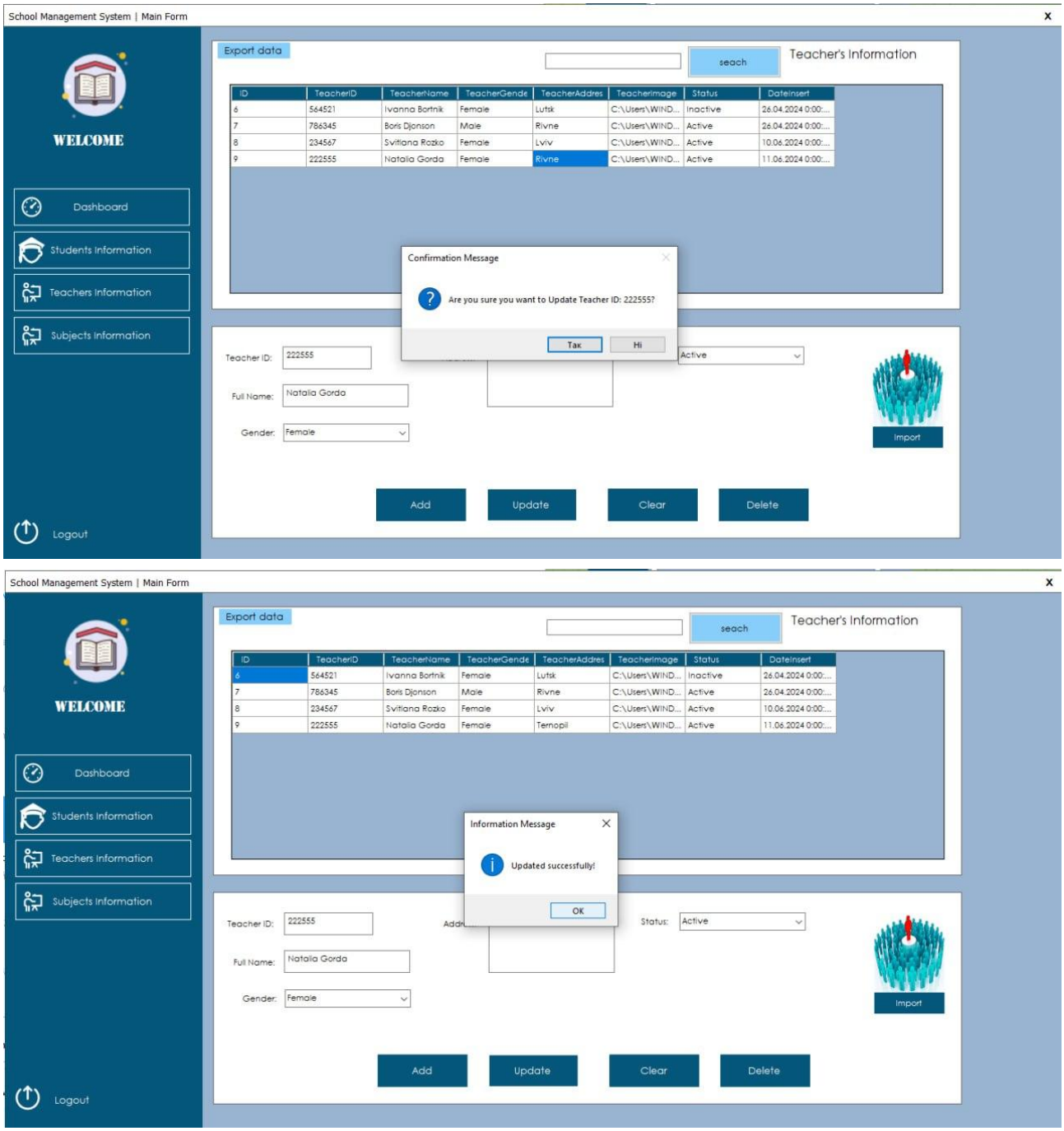


Рис. 2.34 Повідомлення про підтвердження редагування даних

Натискаємо “ОК” та бачимо відразу результат в таблиці на рисунку 2.35.

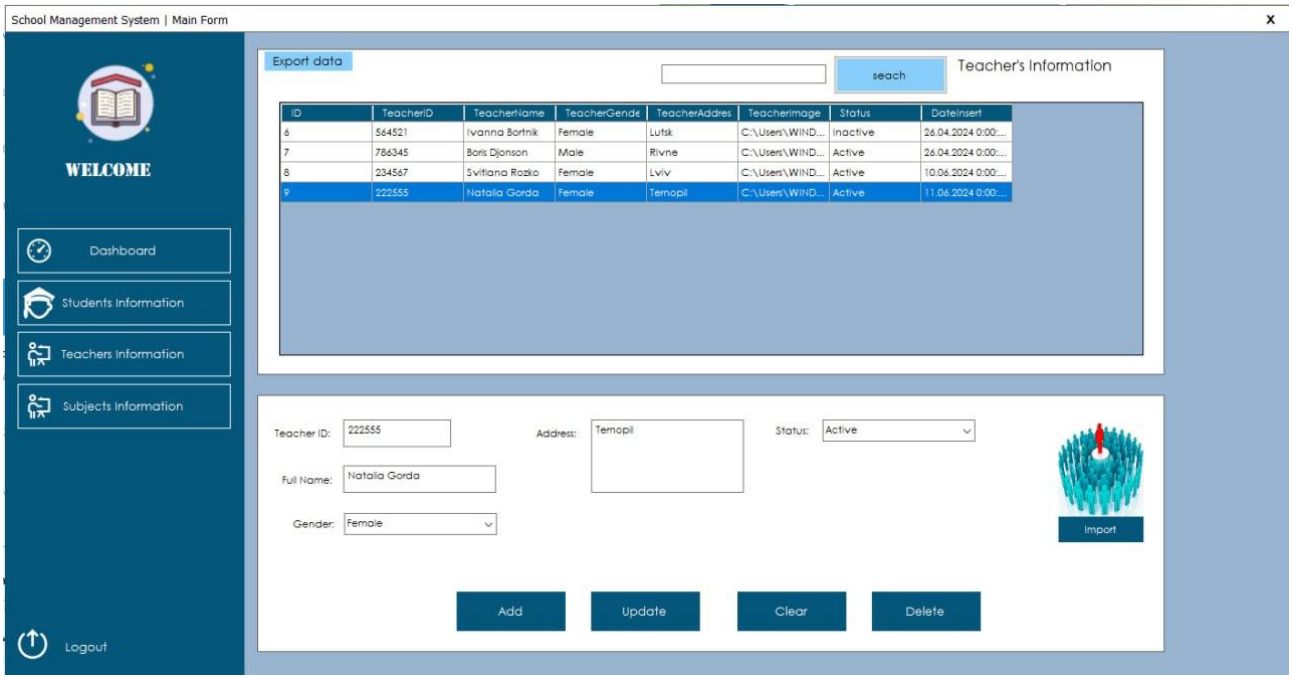


Рис. 2.35 Таблиця після редагування студента

- Очищення полів

  Для зручності користування полями, додано очищення всіх полів. Для тестування продемонструю заповнення поля – будь-якою непотрібною інформацією на рисунку 2.36.

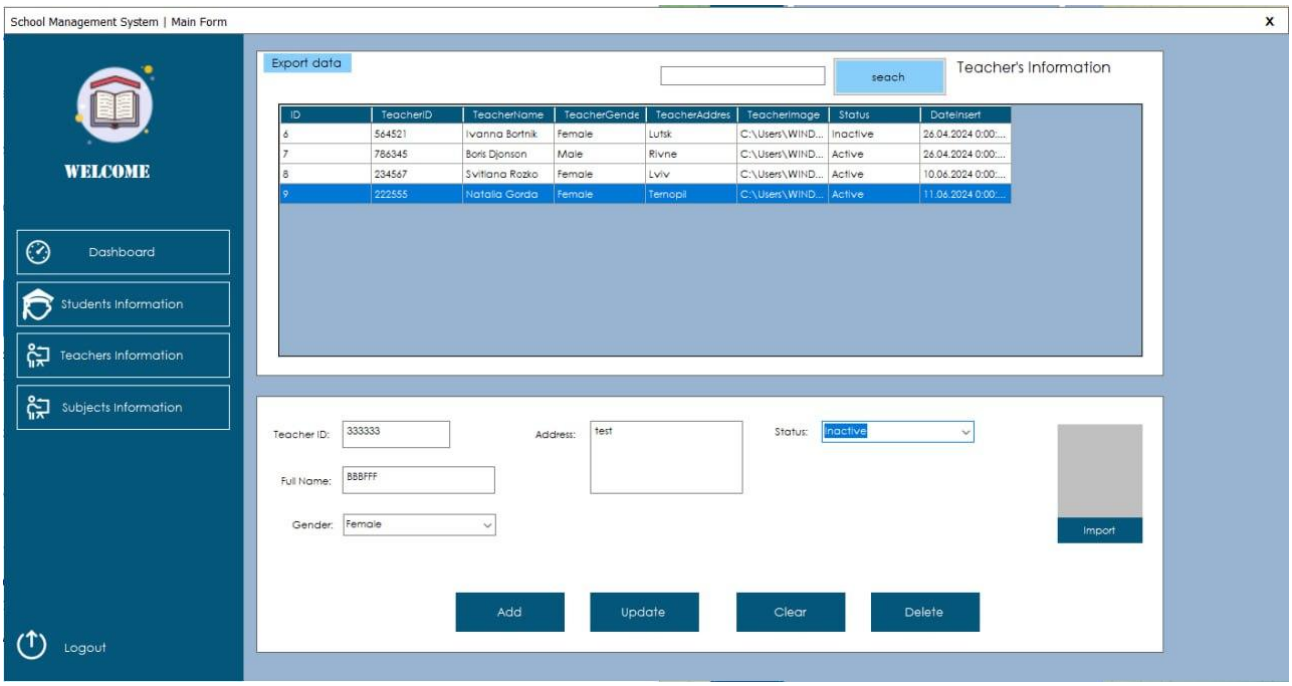


Рис. 2.36 Заповнення полів будь-якою інформацією

Натискаємо кнопку “Clear” та відразу бачимо всі поля чистими на рисунку 2.37.

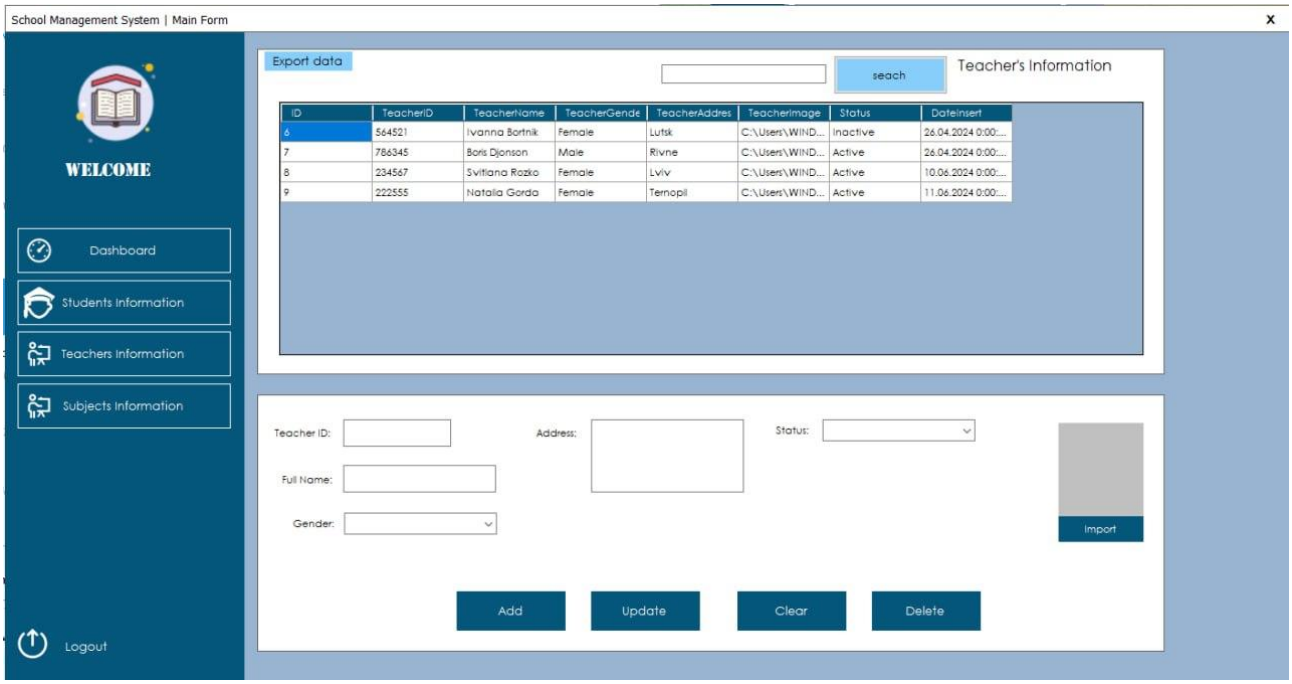


Рис 2.37 Вигляд форми після її очищення

- Видалення вчителя

  Для видалення вчителя зі списку та бази – обираємо вчителя за допомогою клацання на його ID на рисунку та натискаємо кнопку “Delete” і з’являється повідомлення про точне видалення зі списку на рисунку 2.38.

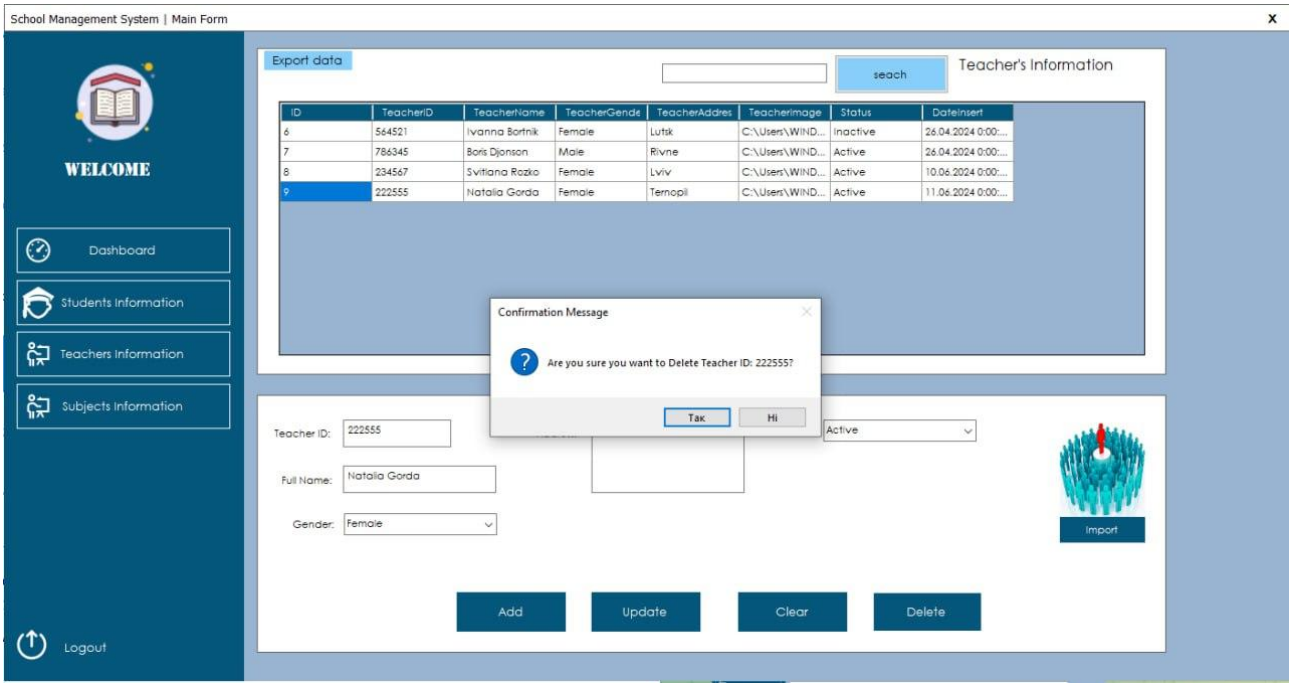


Рис 2.38 Вибираємо вчителя для його видалення

З’являється повідомлення про успішне видалення вчителя 2.39.

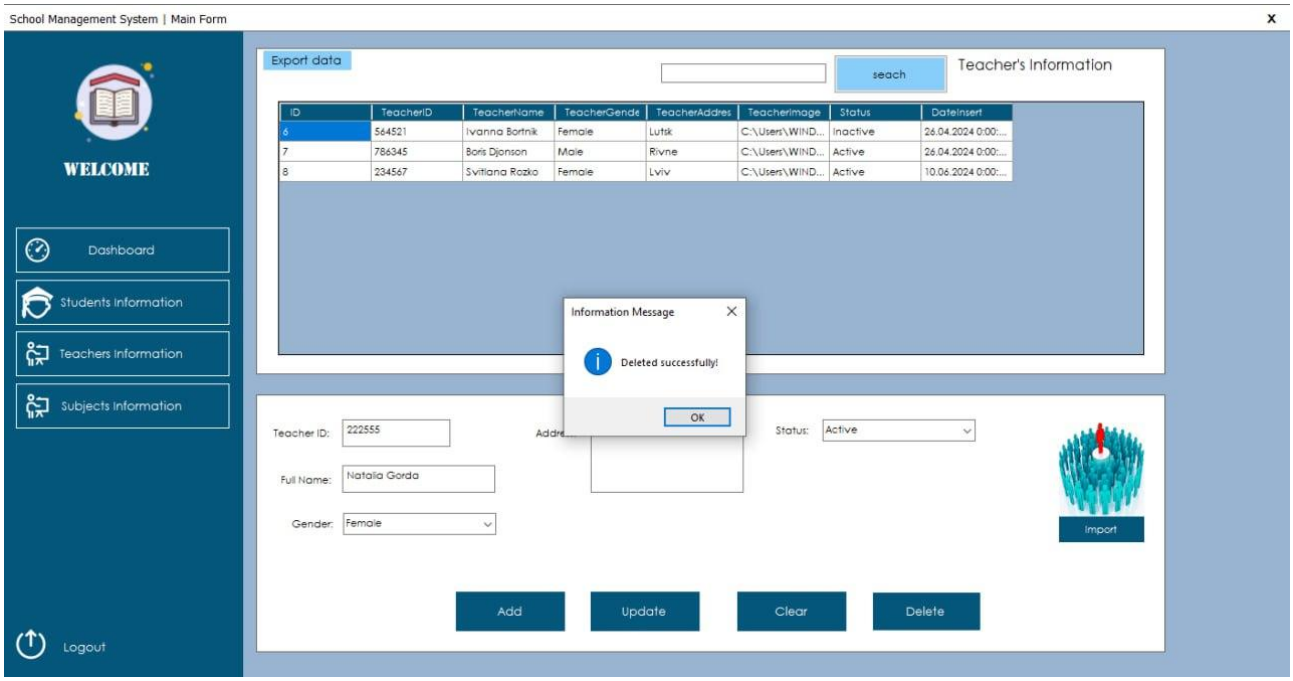


Рис. 2.39 Повідомлення про успішне видалення

Та в результаті, бачимо що даного вчителя в списку немає на рисунку 2.40.

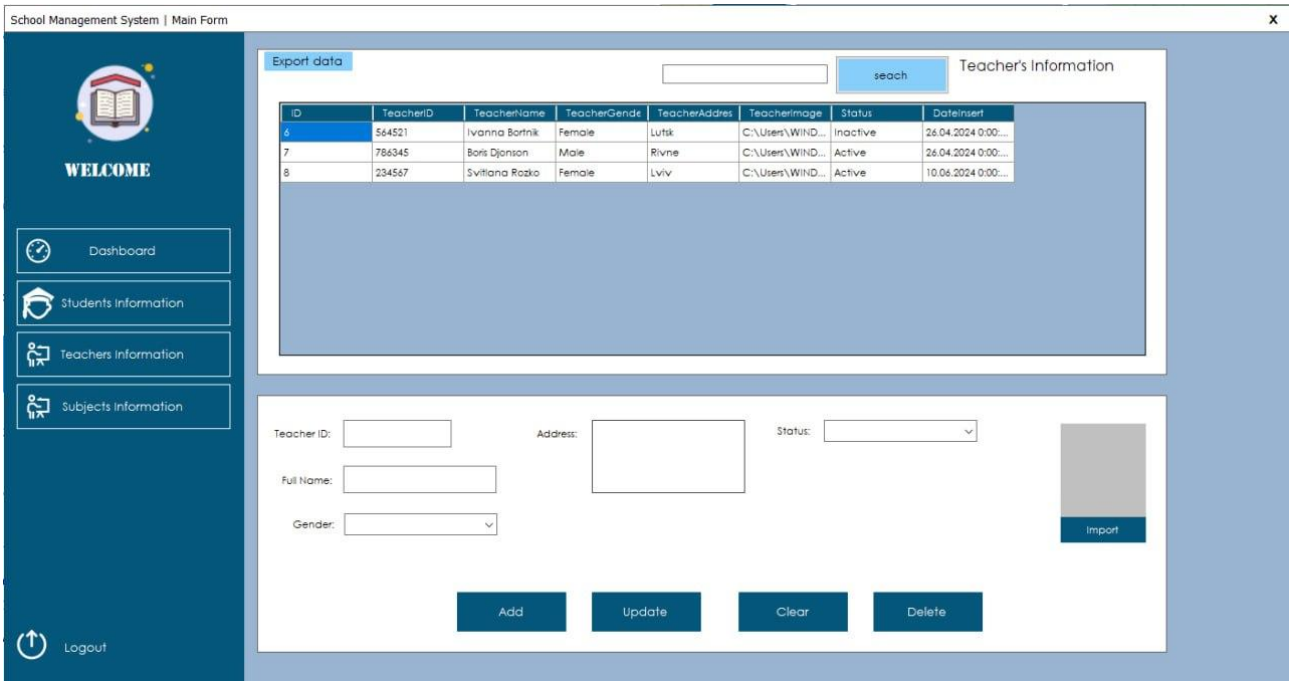


Рис. 2.40 Список, після успішного видалення студента

- Пошук вчителя

  Вводимо текст для пошуку інформації про вчителів та натискаємо кнопку “Search” - відразу бачимо результат в таблиці на рисунку 2.41

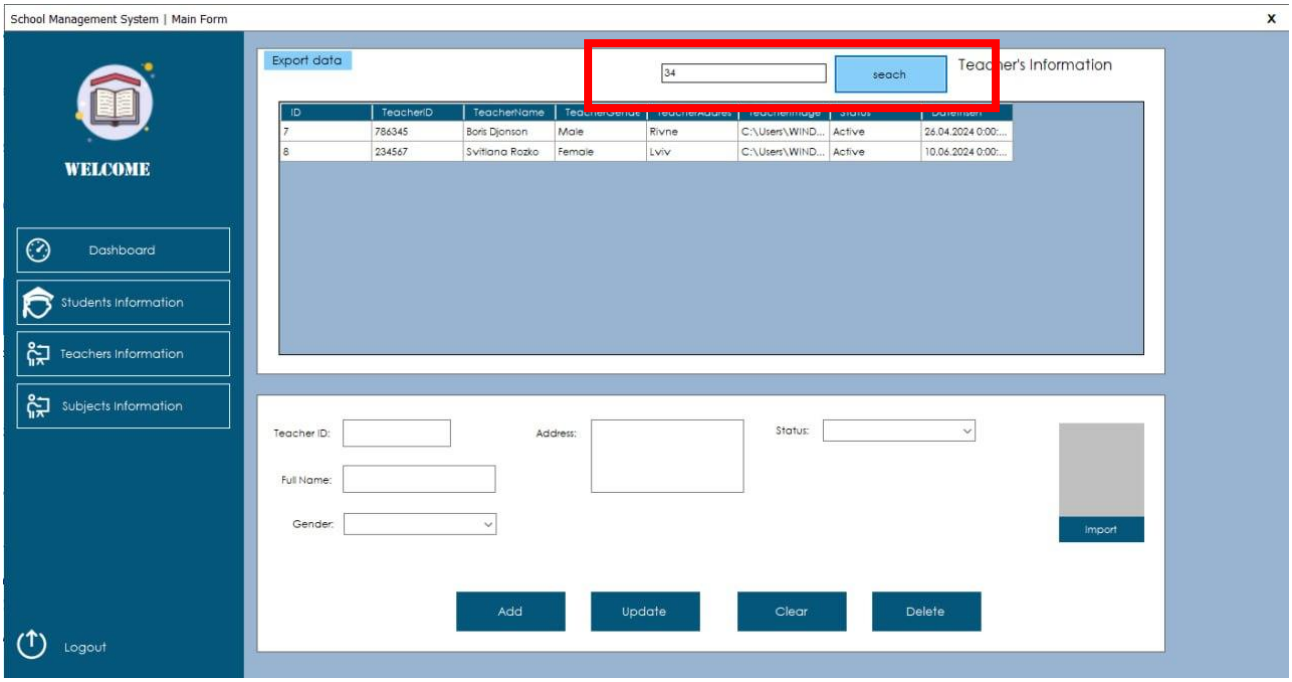


Рис. 2.41 Результат пошуку в таблиці

Також так як і для студента, так і для вчителів є експорт – у Word file та Excel file на рисунку 2.42

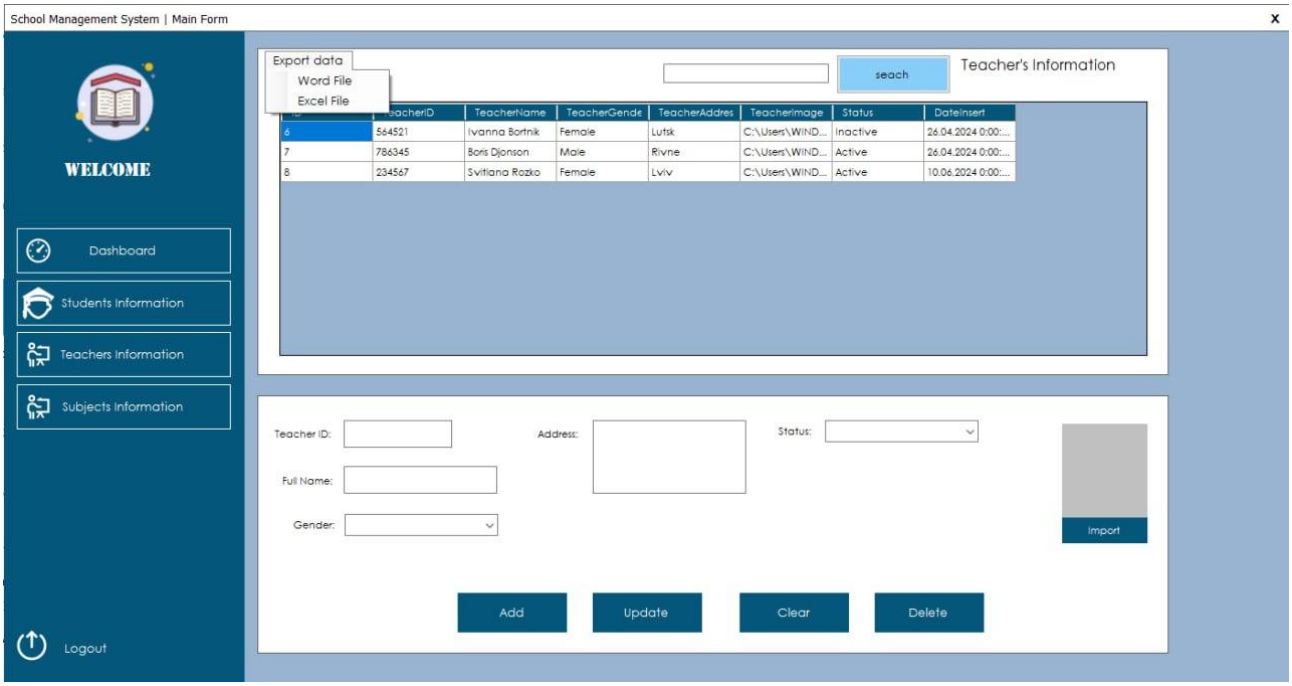


Рис. 2.42 Експорт даних

4. Панель додавання предметів на рисунку 2.43. Тут реалізовано додавання предмета та відразу прив’язка до вчителя, що його викладає і додавання відразу учнів.

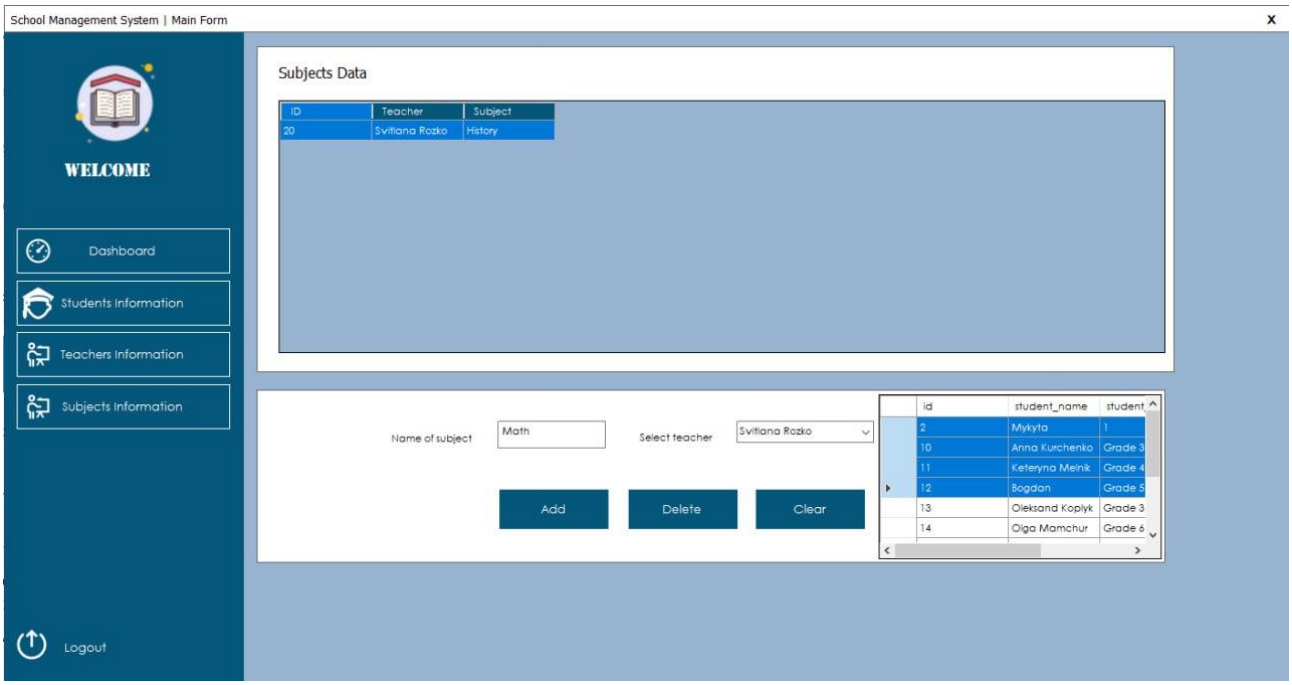


Рис. 2.43 Додавання предмету

Результат, після натиснення кнопки “Add” – оновлена таблиця з новим предметом та його вчителем на рисунку 2.44.

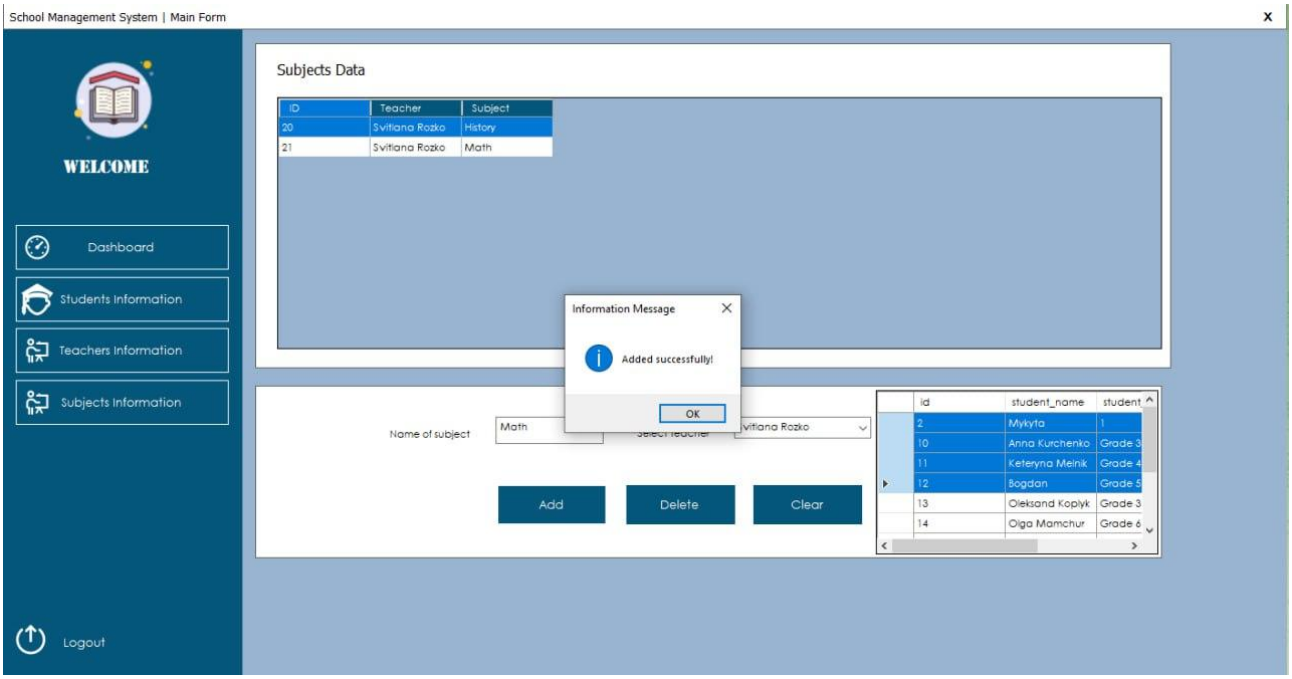


Рис. 2.44 Таблиця, після додавання предмету

5. **Вихід із системи.** створено кнопку Logout на рисунку 2.45. При її натисканні програма перепитує про ваше рішення і повертається на початковий Login на рисунку 2.46.

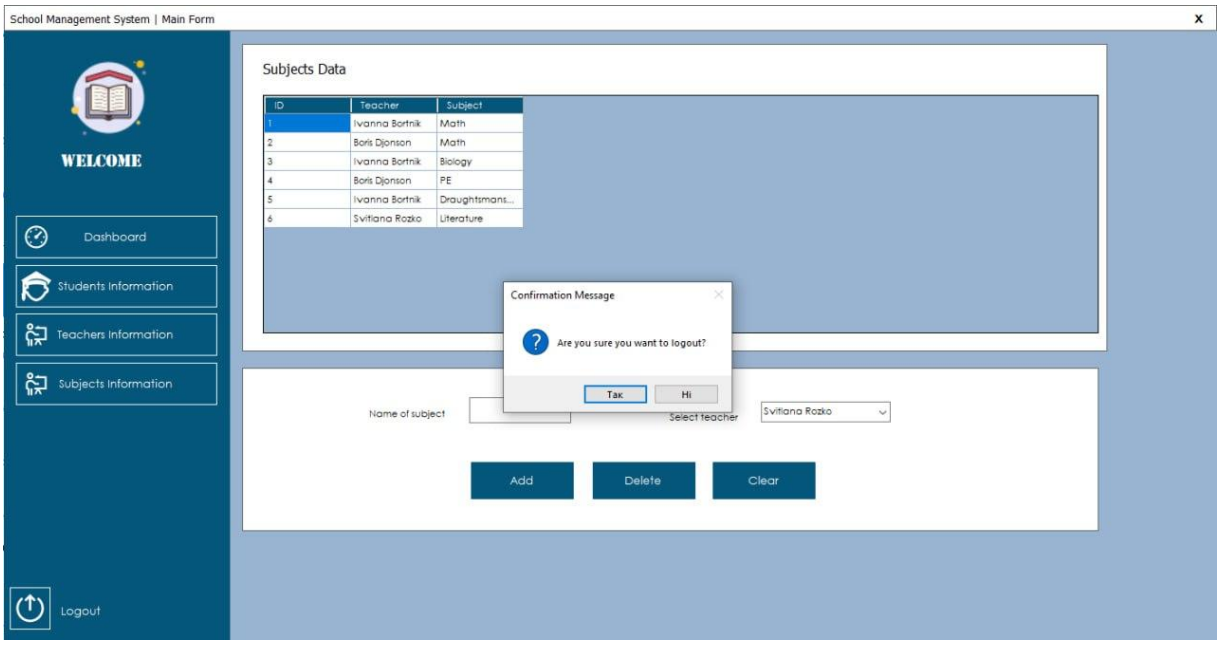


Рис. 2.45 Вихід з системи

Рис. 2.46 Login

Вхід в систему як вчитель (teacher)

Вчитель зі свого акаунту може бачити предмети які він веде та студентів які на цьому предметі на рисунку 2.47.

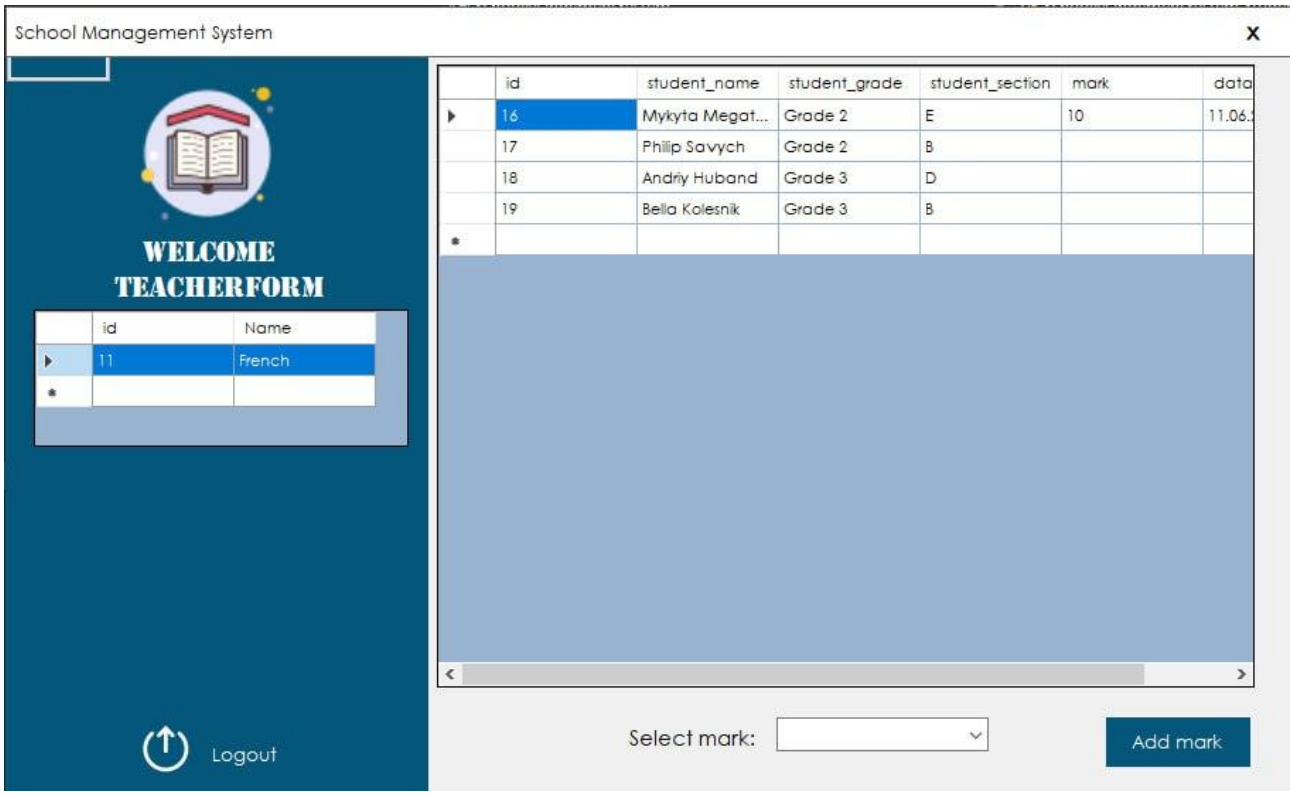


Рис. 2.47 Перегляд свого предмету

Для цих студентів вчитель може виставити оцінку. Вчитель обирає студента та вибирає оціну на рисунку 2.48

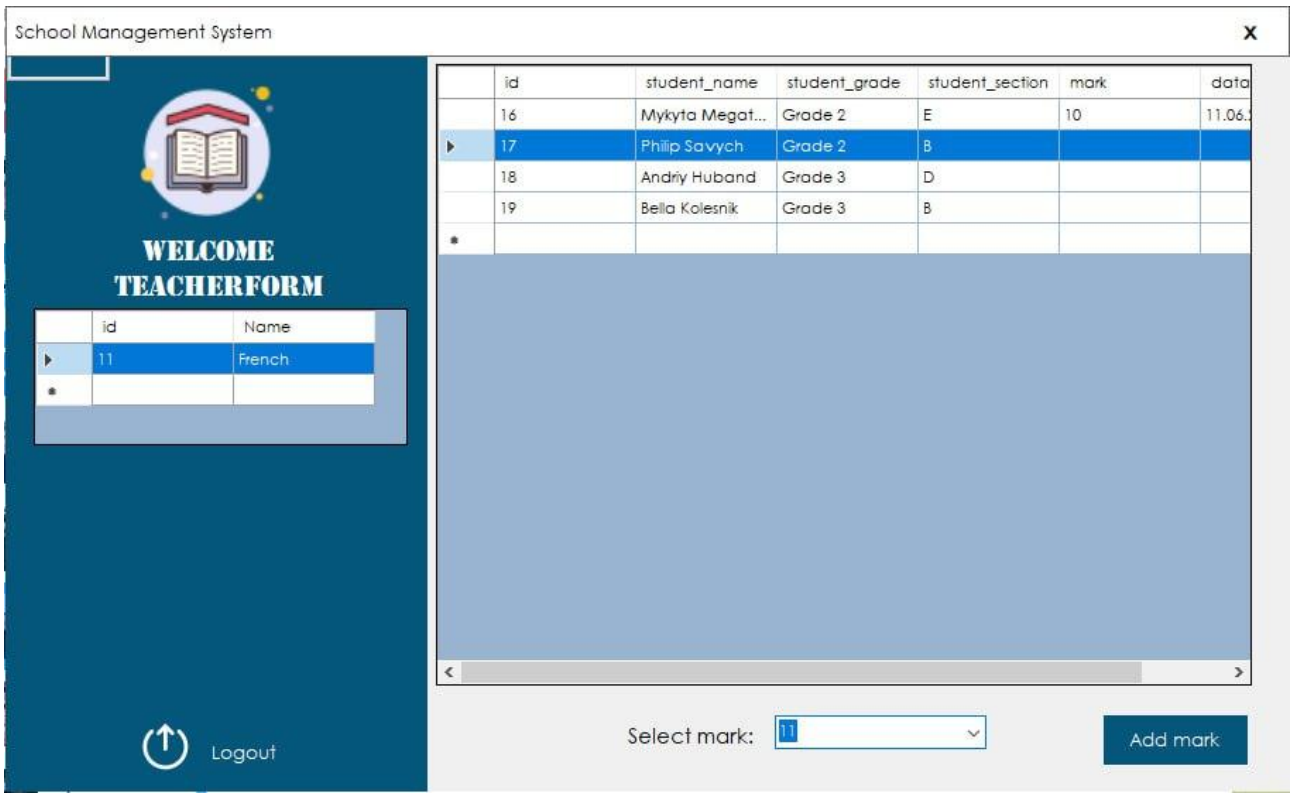


Рис. 2.48 Вибір студента для виставленян оцінки

Натискаємо кнопку “Add mark” та з'являється повідомлення про успішне додавання оцінки, зображено на рисунку 2.49.

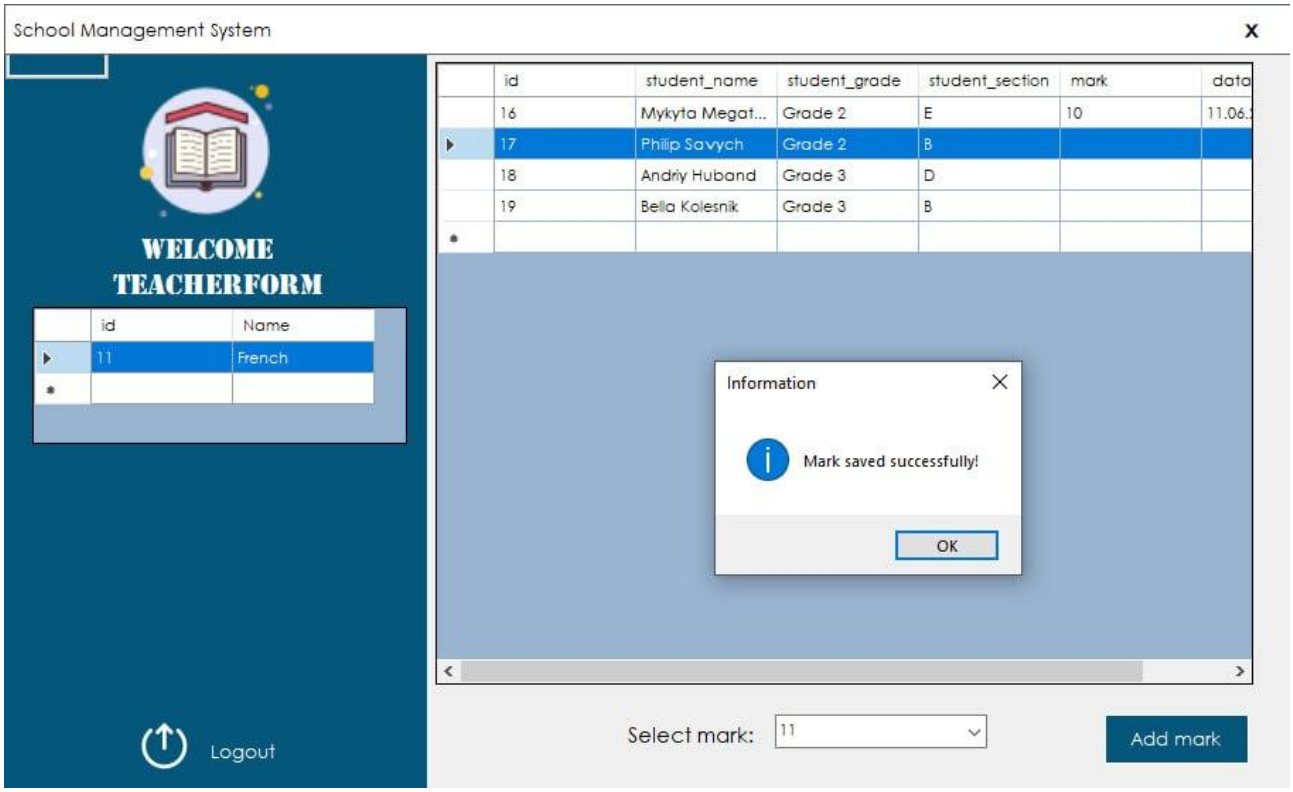


Рис. 2.49 Повідомлення про успішне додавання оцінки

Вигляд таблиці після додавання оцінки на рисунку 2.50.

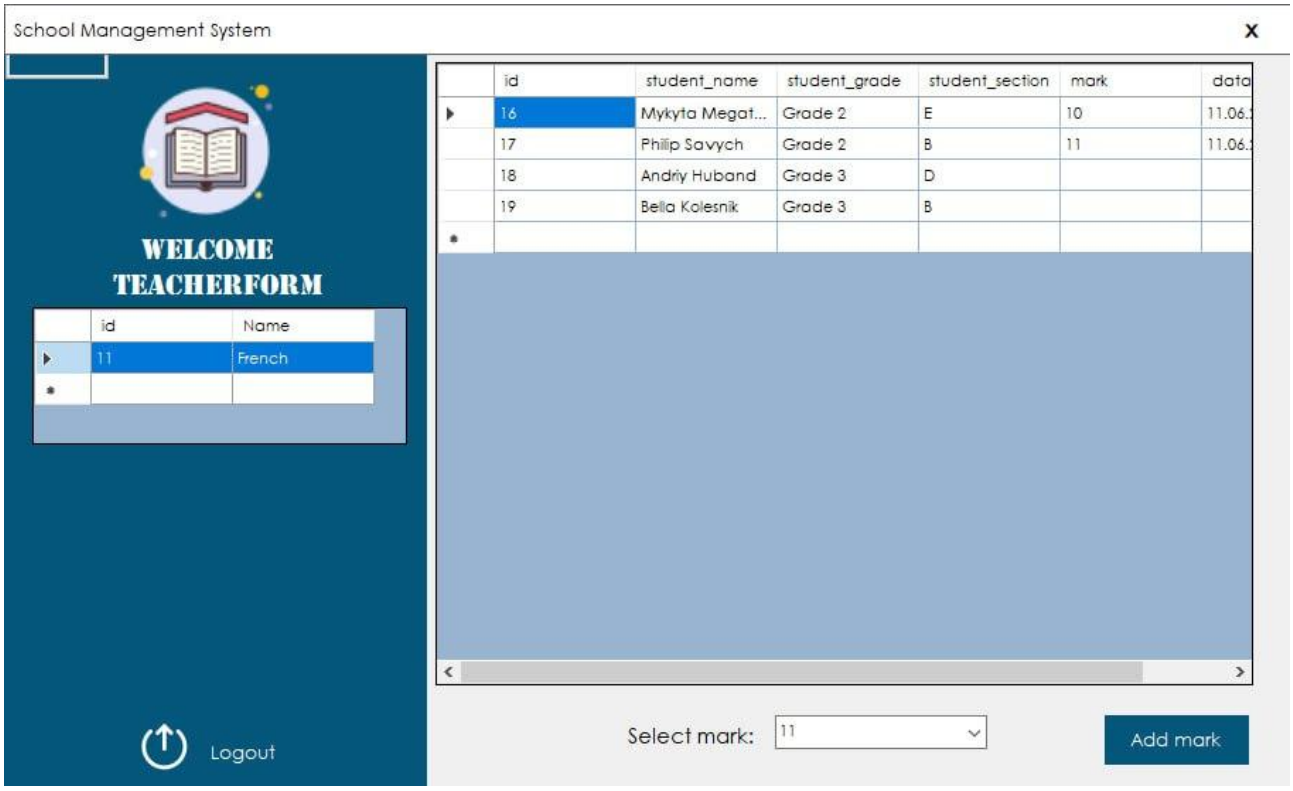


Рис. 2.50 Таблиця з оцінками

Таким чином, програма School Management System створює ефективну та зручну інфраструктуру для управління навчальним процесом, забезпечуючи інтегровані рішення для введення, збереження, редагування та виведення інформації. Це сприяє підвищенню ефективності управління школою.

ВИСНОВОК

У цій дипломній роботі була проведена детальна розробка інформаційної системи для школи з використанням сучасних методів проектування та розробки програмного забезпечення. Починаючи з аналізу потреб користувачів та актуальності завдання, було розглянуто всі аспекти створення такої системи, включаючи логічну модель даних, організацію введення-виведення інформації та функціональні можливості.

В ході проектування було розроблено логічну модель даних, яка включає таблиці для зберігання інформації про учнів, вчителів, класи, предмети та оцінки. Також була розроблена база даних системи, де враховано не тільки структуру даних, а й їхню оптимізацію для ефективного використання.

База даних системи була реалізована з використанням мови SQL, що дозволить ефективно зберігати та оптимізовувати доступ до даних. Використання SQL дозволить легко і швидко взаємодіяти з базою даних, забезпечуючи стабільну та надійну роботу системи.

Організація введення-виведення інформації була вивчена з точки зору зручного та ефективного інтерфейсу користувача, можливості введення нових даних та отримання потрібної інформації з системи.

Крім того, в процесі розробки було створено зручний та інтуїтивно зрозумілий інтерфейс для користувачів на основі платформи WinForm, що сприятиме зручному та ефективному використанню системи

Функціональні можливості інформаційної системи були ретельно розглянуті та визначені з урахуванням потреб користувачів. Система забезпечує широкий функціонал, що дозволить ефективно ведення навчального процесу та адміністрування школи.

Загалом, дипломна робота не лише дозволила глибоко вивчити процес проектування та розробки інформаційної системи для школи, але й надала практичні.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Створення запитів на SQL
URL: https://itvdn.com/ua/blog/article/m-sql
2. Використання UML-diagram:
URL: https://dou.ua/forums/topic/40575/
3. WindowsForms and Database
URL: https://www.c-sharpcorner.com/UploadFile/5d065a/tutorial-1sql-server-database-connection-in-window-form/
4. Windows Forms C#:
URL https://itproger.com/ua/course/csharp-app
5. Основи побудови SQL запитів:
URL: https://foxminded.ua/sql-zapyty/
6. Авторизація користувача через базу даних:
URL:https://www.youtube.com/watch?v=eLQAgHexThM&list=PL0lO_mIqDDFWOMqSKFaLypANf1W7-o87q&index=4&ab_channel=%D0%93%D0%BE%D1%88%D0%B0%D0%94%D1%83%D0%B4%D0%B0%D1%80%D1%8C
7. Сучасні інформаційні технології у школі
URL: https://osvita.ua/school/method/34855/
8. C# language documentation
URL: https://learn.microsoft.com/en-us/dotnet/csharp/
9. Java Platform, Standard Edition Documentation
URL: https://docs.oracle.com/en/java/javase/11/
10. Інформаційні технології в системі сучасної освіти
URL: https://api.dspace.khadi.kharkov.ua/server/api/core/bitstreams/6df1ed46-3c06-490e-8313-fa734a588c91/content
11. Розвиток інформаційних систем управління освітою
URL: https://iea.gov.ua/wp-content/uploads/2020/07/Rozvitok-IS-2020-monografiyaFINAL_sajt.pd

ДОДАТКИ

Код форми AddStudentForm

```
using System;
using System.Windows.Forms;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Drawing;
//using System.Reflection;
using Word = Microsoft.Office.Interop.Word;
using Excel = Microsoft.Office.Interop.Excel;
using System.Linq;

namespace SchoolMangementSystem
{
   public partial class AddStudentForm : UserControl
   {
      SqlConnection connect = new
SqlConnection("Server=tcp:anna123.database.windows.net,1433;Initial Catalog=mukota;Persist
Security Info=False;User
ID=anna;Password=11082003Nik;Encrypt=True;TrustServerCertificate=False;Connection
Timeout=30;");
      public AddStudentForm()
      {
         InitializeComponent();

         displayStudentData();
      }

      public void displayStudentData()
      {
         AddStudentData adData = new AddStudentData();

         student_studentData.DataSource = adData.studentData();
      }

      private void button2_Click(object sender, EventArgs e)
      {
         if (student_id.Text == ""
             student_name.Text == ""
             student_gender.Text == ""
             student_address.Text == ""
             student_grade.Text == ""
             student_section.Text == ""
             student_status.Text == ""
             student_image.Image == null
             imagePath == null)
         {
            MessageBox.Show("Please fill all blank fields", "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
        }
        else
        {
            if (connect.State != ConnectionState.Open)
            {
                try
                {
                    connect.Open();
                    string checkStudentID = "SELECT COUNT(*) FROM students WHERE student_id
= @studentID";

                    using (SqlCommand checkSID = new SqlCommand(checkStudentID, connect))
                    {
                        checkSID.Parameters.AddWithValue("@studentID", student_id.Text.Trim());
                        int count = (int)checkSID.ExecuteScalar();

                        if (count >= 1)
                        {
                            MessageBox.Show("Student ID: " + student_id.Text.Trim() + " is already exist"
                                , "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);

                        }
                        else
                        {
                            DateTime today = DateTime.Today;
                            string insertData = "INSERT INTO students (student_id, student_name" +
                                ", student_gender, student_address, student_grade, student_section" +
                                ", student_image, student_status, date_insert, student_password) " +
                                "VALUES(@studentID, @studentName, @studentGender, @studentAddress"
+
                                ", @studentGrade, @studentSection, @studentImage, @studentStatus" +
                                ", @dateInsert, @studentPassword)";

                            // TO SAVE TO YOUR DIRECTORY
                            string path = Path.Combine(@"C:\Users\WINDOWS
10\source\repos\SchoolMangementSystem\SchoolMangementSystem\Student_Directory\",
student_id.Text.Trim() + ".jpg");

                            string directoryPath = Path.GetDirectoryName(path);

                            if (!Directory.Exists(directoryPath))
                            {
                                Directory.CreateDirectory(directoryPath);
                            }

                            File.Copy(imagePath, path, true);
using (SqlCommand cmd = new SqlCommand(insertData, connect))
                            {
                                cmd.Parameters.AddWithValue("@studentID", student_id.Text.Trim());
                                cmd.Parameters.AddWithValue("@studentName",
student_name.Text.Trim());
```

```
                        cmd.Parameters.AddWithValue("@studentGender",
student_gender.Text.Trim());
                        cmd.Parameters.AddWithValue("@studentAddress",
student_address.Text.Trim());
                        cmd.Parameters.AddWithValue("@studentGrade",
student_grade.Text.Trim());
                        cmd.Parameters.AddWithValue("@studentSection",
student_section.Text.Trim());
                        cmd.Parameters.AddWithValue("@studentImage", path);
                        cmd.Parameters.AddWithValue("@studentStatus",
student_status.Text.Trim());
                        cmd.Parameters.AddWithValue("@dateInsert", today);
                        cmd.Parameters.AddWithValue("@studentPassword",
textBoxpassword.Text.Trim());

                        cmd.ExecuteNonQuery();

                        displayStudentData();

                        MessageBox.Show("Added successfully!", "Information Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                        clearFields();
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error connecting Database: " + ex, "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);

            }
            finally
            {
                connect.Close();
            }
        }
    }

    public void clearFields()
    {
        student_id.Text = "";
        student_name.Text = "";
        student_gender.SelectedIndex = -1;
        student_address.Text = "";
        student_grade.SelectedIndex = -1;
        student_section.SelectedIndex = -1;
        student_status.SelectedIndex = -1;
        student_image.Image = null;
```

```
        imagePath = "";
        textBoxpassword.Text = "";
    }

    public string imagePath;
    private void button1_Click(object sender, EventArgs e)
    {
        OpenFileDialog open = new OpenFileDialog();
        open.Filter = "Image files (*.jpg; *.png)|*.jpg;*.png";

        if (open.ShowDialog() == DialogResult.OK)
        {
            imagePath = open.FileName;

            student_image.ImageLocation = imagePath;
        }
    }

    private void student_updateBtn_Click(object sender, EventArgs e)
    {
        if (student_id.Text == ""
            student_name.Text == ""
            student_gender.Text == ""
            student_address.Text == ""
            student_grade.Text == ""
            student_section.Text == ""
            student_status.Text == ""
            student_image.Image == null
            imagePath == null)
        {
            MessageBox.Show("Please select item first", "Error Message", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        else
        {
            if (connect.State != ConnectionState.Open)
            {
                try
                {
                    connect.Open();

                    DialogResult check = MessageBox.Show("Are you sure you want to Update Student
ID: "
                        + student_id.Text.Trim() + "?", "Confirmation Message",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);

                    if (check == DialogResult.Yes)
                    {
                        DateTime today = DateTime.Today;

String updateData = "UPDATE students SET student_name = @studentName, " +
```

```
                "student_gender = @studentGender, student_address = @studentAddress, " +
                "student_grade = @studentGrade, student_section = @studentSection, " +
                "student_status = @studentStatus " + // Видалено зайву кому
                "WHERE student_id = @studentID";


                    using (SqlCommand cmd = new SqlCommand(updateData, connect))
                    {
                        cmd.Parameters.AddWithValue("@studentName", student_name.Text.Trim());
                        cmd.Parameters.AddWithValue("@studentGender",
student_gender.Text.Trim());
                        cmd.Parameters.AddWithValue("@studentAddress",
student_address.Text.Trim());
                        cmd.Parameters.AddWithValue("@studentGrade", student_grade.Text.Trim());
                        cmd.Parameters.AddWithValue("@studentSection",
student_section.Text.Trim());
                        cmd.Parameters.AddWithValue("@studentStatus", student_status.Text.Trim());
                        //cmd.Parameters.AddWithValue("@dateUpdate", today);
                        cmd.Parameters.AddWithValue("@studentID", student_id.Text.Trim());
                        cmd.Parameters.AddWithValue("@studentPassword",
textBoxpassword.Text.Trim());
                        cmd.ExecuteNonQuery();

                        displayStudentData();

                        MessageBox.Show("Updated successfully!", "Information Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                        clearFields();

                    }
                }
                else
                {
                    MessageBox.Show("Cancelled.", "Information Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    clearFields();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error connecting Database: " + ex, "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);

            }
            finally
            {
                connect.Close();
            }
        }
    }
```

```
    }

    private void student_studentData_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        if (e.RowIndex != -1)
        {
            DataGridViewRow row = student_studentData.Rows[e.RowIndex];
            student_id.Text = row.Cells[1].Value.ToString();
            student_name.Text = row.Cells[2].Value.ToString();
            student_gender.Text = row.Cells[3].Value.ToString();
            student_address.Text = row.Cells[4].Value.ToString();
            student_grade.Text = row.Cells[5].Value.ToString();
            student_section.Text = row.Cells[6].Value.ToString();

            imagePath = row.Cells[5].Value.ToString();

            string imageData = row.Cells[7].Value.ToString();

            if (imageData != null && imageData.Length > 0)
            {
                student_image.Image = Image.FromFile(imageData);
            }
            else
            {
                student_image.Image = null;
            }

            student_status.Text = row.Cells[8].Value.ToString();

        }
    }

private void student_deleteBtn_Click(object sender, EventArgs e)
    {
        if (student_id.Text == "")
        {
            MessageBox.Show("Please select item first", "Error Message"
                , MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            if (connect.State != ConnectionState.Open)
            {
                DialogResult check = MessageBox.Show("Are you sure you want to Delete Student ID:
"
                    + student_id.Text + "?", "Confirmation Message", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);

                if (check == DialogResult.Yes)
                {
```

```
                try
                {
                    connect.Open();
                    DateTime today = DateTime.Today;

                    string deleteData = "UPDATE students SET date_delete = @dateDelete " +
                        "WHERE student_id = @studentID";

                    using (SqlCommand cmd = new SqlCommand(deleteData, connect))
                    {
                        cmd.Parameters.AddWithValue("@dateDelete", today);
                        cmd.Parameters.AddWithValue("@studentID", student_id.Text.Trim());

                        cmd.ExecuteNonQuery();

                        displayStudentData();

                        MessageBox.Show("Deleted successfully!", "Information Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                        clearFields();
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error connecting Database: " + ex, "Error Message"
                , MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    connect.Close();
                }
            }
            else
            {
                MessageBox.Show("Cancelled.", "Information Message"
                , MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }
    }
}

private void student_clearBtn_Click(object sender, EventArgs e)
{
    //student_id.Text = "";
    //student_name.Text = "";
    //student_gender.SelectedIndex = -1;
    //student_address.Text = "";
    //student_grade.SelectedIndex = -1;
    //student_section.SelectedIndex = -1;
```

```
        //student_status.SelectedIndex = -1;
        //student_image.Image = null;
        //imagePath = "";
        clearFields();
    }


    private void wordFileToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (connect.State != ConnectionState.Open)
        {
            connect.Open();
        }
        string query = "SELECT student_id, student_name, student_gender, student_address,
student_grade, student_section, student_status FROM students";
        SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connect);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable);

        // Створення нового документа Word
        Word.Application wordApp = new Word.Application();
        Word.Document wordDoc = wordApp.Documents.Add();
        Word.Paragraph paragraph = wordDoc.Content.Paragraphs.Add();
        paragraph.Range.Text = "Список студентів";
        paragraph.Range.Font.Bold = 1;
        paragraph.Range.Font.Size = 14;
        paragraph.Range.InsertParagraphAfter();

        // Додавання таблиці
        Word.Table table = wordDoc.Tables.Add(paragraph.Range, dataTable.Rows.Count + 1,
dataTable.Columns.Count);
        table.Borders.Enable = 1;

// Заповнення заголовків таблиці
        for (int i = 0; i < dataTable.Columns.Count; i++)
        {
            table.Cell(1, i + 1).Range.Text = dataTable.Columns[i].ColumnName;
            table.Cell(1, i + 1).Range.Font.Bold = 1;
            table.Cell(1, i + 1).Range.Font.Size = 12;
            table.Cell(1, i + 1).Shading.BackgroundPatternColor = Word.WdColor.wdColorGray25;
        }

        // Заповнення даних таблиці
        for (int i = 0; i < dataTable.Rows.Count; i++)
        {
            for (int j = 0; j < dataTable.Columns.Count; j++)
            {
                table.Cell(i + 2, j + 1).Range.Text = dataTable.Rows[i][j].ToString();
            }
        }
```

```
        // Збереження документа
        string filePath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop),
"StudentsList.docx");
        wordDoc.SaveAs2(filePath);
        wordDoc.Close();
        wordApp.Quit();
        MessageBox.Show("Документ Word успішно створено на робочому столі",
"Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);

    }

    private void exelFileToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (connect.State != ConnectionState.Open)
        {
            connect.Open();
        }

        string query = "SELECT student_id, student_name, student_gender, student_address,
student_grade, student_section, student_status FROM students";
        SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connect);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable);

        // Створення нового документа Excel
        Excel.Application excelApp = new Excel.Application();
        Excel.Workbook workbook = excelApp.Workbooks.Add();
        Excel.Worksheet worksheet = (Excel.Worksheet)workbook.Sheets[1];
        worksheet.Name = "Students";

        // Заповнення заголовків стовпців
        for (int i = 0; i < dataTable.Columns.Count; i++)
        {
            worksheet.Cells[1, i + 1] = dataTable.Columns[i].ColumnName;
            worksheet.Cells[1, i + 1].Font.Bold = true;
            worksheet.Cells[1, i + 1].Interior.Color = Excel.XlRgbColor.rgbLightGray;
        }

        // Заповнення даних
        for (int i = 0; i < dataTable.Rows.Count; i++)
        {
            for (int j = 0; j < dataTable.Columns.Count; j++)
            {
                worksheet.Cells[i + 2, j + 1] = dataTable.Rows[i][j].ToString();
            }
        }

        // Авто-підбір ширини стовпців
        worksheet.Columns.AutoFit();
```

```
        // Збереження файлу
        string filePath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop),
"StudentsList.xlsx");
        workbook.SaveAs(filePath);
        workbook.Close();
        excelApp.Quit();

        MessageBox.Show("Файл Excel успішно створено на робочому столі", "Інформація",
MessageBoxButtons.OK, MessageBoxIcon.Information);

    }

    private void search_Click(object sender, EventArgs e)
    {
        string searchValue = search_text.Text.ToLower();
        CurrencyManager currencyManager =
(CurrencyManager)BindingContext[student_studentData.DataSource];
        currencyManager.SuspendBinding();

        try
        {
            foreach (DataGridViewRow row in student_studentData.Rows)
            {
                row.Visible = row.Cells.Cast<DataGridViewCell>()
                    .Any(c => c.Value != null &&
c.Value.ToString().ToLower().Contains(searchValue));
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        finally
        {
            currencyManager.ResumeBinding();
        }
    }


  }
}
```

Код форми AddSubjectForm

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SchoolMangementSystem
{
   public partial class AddSubjectForm : UserControl
   {
      SqlConnection connect = new
SqlConnection("Server=tcp:anna123.database.windows.net,1433;Initial Catalog=mukota;Persist
Security Info=False;User
ID=anna;Password=11082003Nik;MultipleActiveResultSets=False;Encrypt=True;TrustServerCer
tificate=False;Connection Timeout=30;");
      public AddSubjectForm()
      {
         InitializeComponent();
         AddTeachersData addT = new AddTeachersData();
         comboBox1.Items.AddRange(addT.teacherDataName().ToArray());
         subjectDisplayData();
         LoadStudents();
      }
      public void subjectDisplayData()
      {
         TeacherSubjectData addSB = new TeacherSubjectData();

         subject_gridData.DataSource = addSB.subjectData();

      }
      private void subject_clearBtn_Click(object sender, EventArgs e)
      {
         subject_name.Text = "";
         comboBox1.SelectedIndex = -1;
      }
      public void clearFields()
      {
         subject_name.Text = "";
      }
      private void subject_addBtn_Click(object sender, EventArgs e)
      {
         if (subject_name.Text == "")
         {
            MessageBox.Show("Please fill all blank fields", "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
         }
         else
         {

            if (connect.State != ConnectionState.Open)
            {
               try
               {
```

```
                connect.Open();

                string checkSubjectID = "SELECT Id FROM subject WHERE Name = @NameID";

                using (SqlCommand checkSubjectCmd = new SqlCommand(checkSubjectID,
connect))
                {
                    checkSubjectCmd.Parameters.AddWithValue("@NameID",
subject_name.Text.Trim());
                    var existingSubjectId = checkSubjectCmd.ExecuteScalar();

                    if (existingSubjectId != null) // Предмет вже існує
                    {
                        int subjectId = Convert.ToInt32(existingSubjectId);
                        AddTeacherSubject(subjectId);
                    }
                    else // Предмет ще не існує
                    {
                        string insertSubjectQuery = "INSERT INTO subject (Name) VALUES
(@NameID); SELECT SCOPE_IDENTITY();";
                        using (SqlCommand insertSubjectCmd = new
SqlCommand(insertSubjectQuery, connect))
                        {
                            insertSubjectCmd.Parameters.AddWithValue("@NameID",
subject_name.Text.Trim());
                            int newSubjectId = Convert.ToInt32(insertSubjectCmd.ExecuteScalar());
                            AddTeacherSubject(newSubjectId);
                        }
                    }
                }

                subjectDisplayData();

                MessageBox.Show("Added successfully!", "Information Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                clearFields();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error connecting Database: " + ex.Message, "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                connect.Close();
            }
        }
}
    }
    void AddTeacherSubject(int subjectId)
```

```
    {
        if (studentDataGridView.SelectedRows.Count > 0)
        {
            string insertTeacherSubjectQuery = "INSERT INTO teacher_subject (teacher_id,
subject_id) VALUES (@TeacherID, @SubjectID)";
            using (SqlCommand insertTeacherSubjectCmd = new
SqlCommand(insertTeacherSubjectQuery, connect))
            {
                insertTeacherSubjectCmd.Parameters.AddWithValue("@TeacherID",
returnTeacherIdByName(comboBox1.SelectedItem.ToString())); // assuming you have the
teacher_id value available
                insertTeacherSubjectCmd.Parameters.AddWithValue("@SubjectID", subjectId);
                insertTeacherSubjectCmd.ExecuteNonQuery();
            }

            string insertStudentSubjectQuery = "INSERT INTO student_subject (student_id,
subject_id) VALUES (@StudentID, @SubjectID)";
            using (SqlCommand insertStudentSubjectCmd = new
SqlCommand(insertStudentSubjectQuery, connect))
            {
                foreach (DataGridViewRow row in studentDataGridView.SelectedRows)
                {
                    int studentId = Convert.ToInt32(row.Cells["id"].Value);
                    insertStudentSubjectCmd.Parameters.Clear(); // Clear previous parameters
                    insertStudentSubjectCmd.Parameters.AddWithValue("@StudentID", studentId);
                    insertStudentSubjectCmd.Parameters.AddWithValue("@SubjectID", subjectId);
                    insertStudentSubjectCmd.ExecuteNonQuery();
                }
            }

            MessageBox.Show("Студенти успішно додані до предмету!", "Інформація",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        else
        {
            MessageBox.Show("Будь ласка, виберіть студентів для додавання до предмету.",
"Попередження", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
    private void LoadStudents()
    {
        using (SqlConnection conn = new
SqlConnection("Server=tcp:anna123.database.windows.net,1433;Initial Catalog=mukota;Persist
Security Info=False;User
ID=anna;Password=11082003Nik;MultipleActiveResultSets=False;Encrypt=True;TrustServerCer
tificate=False;Connection Timeout=30;"))
        {
            string query = "SELECT id, student_name, student_grade, student_section FROM
students";

            SqlCommand cmd = new SqlCommand(query, conn);
```

```
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataTable studentsTable = new DataTable();
            adapter.Fill(studentsTable);

            studentDataGridView.DataSource = studentsTable;
        }
    }
    int returnTeacherIdByName(string Name)
    {
        int a = 0;
        AddTeachersData addT = new AddTeachersData();
        foreach (var item in addT.teacherData())
        {
            if(item.TeacherName == Name)
            {
                a = item.TeacherID;
            }
        }
        return a;
    }
    private void subject_gridData_CellContentClick(object sender, DataGridViewCellEventArgs
e)
    {

    }

    private void subject_deleteBtn_Click(object sender, EventArgs e)
    {
        if (subject_gridData.SelectedRows.Count > 0)
        {
            // Отримання ID вибраного рядка
            int selectedSubjectId =
Convert.ToInt32(subject_gridData.SelectedRows[0].Cells["id"].Value);
// Підтвердження видалення
            DialogResult result = MessageBox.Show("Are you sure you want to delete this subject?",
"Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (result == DialogResult.Yes)
            {
                // Видалення з бази даних
                try
                {
                    using (SqlConnection connect = new
SqlConnection("Server=tcp:anna123.database.windows.net,1433;Initial Catalog=mukota;Persist
Security Info=False;User
ID=anna;Password=11082003Nik;MultipleActiveResultSets=False;Encrypt=True;TrustServerCer
tificate=False;Connection Timeout=30;"))
                    {
                        connect.Open();
```

```
                    string deleteQuery = "DELETE FROM teacher_subject WHERE id =
@subjectID";
                    using (SqlCommand cmd = new SqlCommand(deleteQuery, connect))
                    {
                        cmd.Parameters.AddWithValue("@subjectID", selectedSubjectId);
                        cmd.ExecuteNonQuery();
                    }

                    connect.Close();
                }

                // Оновлення відображення даних
                subjectDisplayData();

                MessageBox.Show("Subject deleted successfully!", "Information",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error connecting to the database: " + ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
    else
    {
        MessageBox.Show("Please select a subject to delete.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
  }
 }
}
```

Код форми AddTeachersForm

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using Word = Microsoft.Office.Interop.Word;
using Excel = Microsoft.Office.Interop.Excel;


namespace SchoolMangementSystem
{
```

```
    public partial class AddTeachersForm : UserControl
    {
        SqlConnection connect = new
SqlConnection("Server=tcp:anna123.database.windows.net,1433;Initial Catalog=mukota;Persist
Security Info=False;User
ID=anna;Password=11082003Nik;MultipleActiveResultSets=False;Encrypt=True;TrustServerCer
tificate=False;Connection Timeout=30;");
        public AddTeachersForm()
        {
            InitializeComponent();

            teacherDisplayData();
        }

        public void teacherDisplayData()
        {
            AddTeachersData addTD = new AddTeachersData();

            teacher_gridData.DataSource = addTD.teacherData();
        }

        private void teacher_addBtn_Click(object sender, EventArgs e)
        {
            if (teacher_id.Text == ""
                teacher_name.Text == ""
                teacher_gender.Text == ""
                teacher_address.Text == ""
                teacher_status.Text == ""
                teacher_image.Image == null
                imagePath == null)
            {
                MessageBox.Show("Please fill all blank fields", "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else
            {
                if (connect.State != ConnectionState.Open)
                {
                    try
                    {
                        connect.Open();

                        // WE DON'T LIKE THE DUPLICATE TEACHER ID SO, WE NEED TO CHECK IF
ON THE DATABASE HAS ALREADY TEACHER ID VALUE THAT SAME TO THE USER THAT
WANT TO INSERT
                        string checkTeacherID = "SELECT COUNT(*) FROM teachers WHERE teacher_id
= @teacherID";

                        using (SqlCommand checkTID = new SqlCommand(checkTeacherID, connect))
                        {
                            checkTID.Parameters.AddWithValue("@teacherID", teacher_id.Text.Trim());
```

```
                int count = (int)checkTID.ExecuteScalar();

                if (count >= 1)
                {
                    MessageBox.Show("Teacher ID: " + teacher_id.Text.Trim() + " is already
exist"
                        , "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }
                else
                {
                    DateTime today = DateTime.Today;
                    string insertData = "INSERT INTO teachers " +
                        "(teacher_id, teacher_name, teacher_gender, teacher_address,
teacher_image, teacher_status, date_insert,teacher_password) " +
                        "VALUES(@teacherID, @teacherName, @teacherGender,
@teacherAddress, @teacherImage, @teacherStatus, @dateInsert,@teacherPassword)";

                    // TO SAVE TO YOUR DIRECTORY
                    string path = Path.Combine(@"C:\Users\WINDOWS
10\source\repos\SchoolMangementSystem\SchoolMangementSystem\Teacher_Directory\",
teacher_id.Text.Trim() + ".jpg");

                    string directoryPath = Path.GetDirectoryName(path);

                    if (!Directory.Exists(directoryPath))
                    {
                        Directory.CreateDirectory(directoryPath);
                    }

                    File.Copy(imagePath, path, true);
using (SqlCommand cmd = new SqlCommand(insertData, connect))
                    {
                        cmd.Parameters.AddWithValue("@teacherID", teacher_id.Text.Trim());
                        cmd.Parameters.AddWithValue("@teacherName",
teacher_name.Text.Trim());
                        cmd.Parameters.AddWithValue("@teacherGender",
teacher_gender.Text.Trim());
                        cmd.Parameters.AddWithValue("@teacherAddress",
teacher_address.Text.Trim());
                        cmd.Parameters.AddWithValue("@teacherImage", path);
                        cmd.Parameters.AddWithValue("@teacherStatus",
teacher_status.Text.Trim());
                        cmd.Parameters.AddWithValue("@dateInsert", today);
                        cmd.Parameters.AddWithValue("@teacherPassword",
textboxpass.Text.Trim());
                        cmd.ExecuteNonQuery();

                        teacherDisplayData();
```

```
                    MessageBox.Show("Added successfully!", "Information Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                    clearFields();
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error connecting Database: " + ex, "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);

        }
        finally
        {
            connect.Close();
        }
    }
}

public void clearFields()
{
    teacher_id.Text = "";
    teacher_name.Text = "";
    teacher_gender.SelectedIndex = -1;
    teacher_address.Text = "";
    teacher_status.SelectedIndex = -1;
    teacher_image.Image = null;
    imagePath = "";
    textboxpass.Text = "";
}

private string imagePath;
private void button1_Click(object sender, EventArgs e)
{
    OpenFileDialog open = new OpenFileDialog();
    open.Filter = "Image files (*.jpg; *.png)|*.jpg;*.png";

    if (open.ShowDialog() == DialogResult.OK)
    {
        imagePath = open.FileName;

        teacher_image.ImageLocation = imagePath;
    }
}

private void teacher_clearBtn_Click(object sender, EventArgs e)
{
    clearFields();
```

```
    }

    private void teacher_updateBtn_Click(object sender, EventArgs e)
    {
        if (teacher_id.Text == ""
            teacher_name.Text == ""
            teacher_gender.Text == ""
            teacher_address.Text == ""
            teacher_status.Text == ""
            teacher_image.Image == null
            imagePath == null)
        {
            MessageBox.Show("Please select item first", "Error Message", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        else
        {
            if (connect.State != ConnectionState.Open)
            {
                try
                {
                    connect.Open();

                    DialogResult check = MessageBox.Show("Are you sure you want to Update Teacher
ID: "
                        + teacher_id.Text.Trim() + "?", "Confirmation Message",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);

                    if (check == DialogResult.Yes)
                    {
                        DateTime today = DateTime.Today;

String updateData = "UPDATE teachers SET " +
                            "teacher_name = @teacherName, teacher_gender = @teacherGender" +
                            ", teacher_address = @teacherAddress" +
                            ", teacher_status = @teacherStatus" +
                            ", date_update = @dateUpdate WHERE teacher_id = @teacherID";


                        using (SqlCommand cmd = new SqlCommand(updateData, connect))
                        {
                            cmd.Parameters.AddWithValue("@teacherName", teacher_name.Text.Trim());
                            cmd.Parameters.AddWithValue("@teacherGender",
teacher_gender.Text.Trim());
                            cmd.Parameters.AddWithValue("@teacherAddress",
teacher_address.Text.Trim());
                            cmd.Parameters.AddWithValue("@teacherStatus", teacher_status.Text.Trim());
                            cmd.Parameters.AddWithValue("@dateUpdate", today);
                            cmd.Parameters.AddWithValue("@teacherID", teacher_id.Text.Trim());
                            cmd.Parameters.AddWithValue("@teacherPassword",
textboxpass.Text.Trim());
```

```
                    cmd.ExecuteNonQuery();

                    teacherDisplayData();

                    MessageBox.Show("Updated successfully!", "Information Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                    clearFields();

                }
            }
            else
            {
                MessageBox.Show("Cancelled.", "Information Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                clearFields();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error connecting Database: " + ex, "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);

        }
        finally
        {
            connect.Close();
        }
    }
}

private void teacher_gridData_CellClick(object sender, DataGridViewCellEventArgs e)
{

    if (e.RowIndex != -1)
    {
        DataGridViewRow row = teacher_gridData.Rows[e.RowIndex];
        teacher_id.Text = row.Cells[1].Value.ToString();
        teacher_name.Text = row.Cells[2].Value.ToString();
        teacher_gender.Text = row.Cells[3].Value.ToString();
        teacher_address.Text = row.Cells[4].Value.ToString();

        imagePath = row.Cells[5].Value.ToString();

        string imageData = row.Cells[5].Value.ToString();

        if (imageData != null && imageData.Length > 0)
        {
            teacher_image.Image = Image.FromFile(imageData);
        }
```

```
            else
            {
                teacher_image.Image = null;
            }

            teacher_status.Text = row.Cells[6].Value.ToString();

        }

    }

    private void teacher_deleteBtn_Click(object sender, EventArgs e)
    {
        if (teacher_id.Text == "")
        {
            MessageBox.Show("Please select item first", "Error Message"
                , MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            if (connect.State != ConnectionState.Open)
            {
                DialogResult check = MessageBox.Show("Are you sure you want to Delete Teacher
ID: "
                    + teacher_id.Text + "?", "Confirmation Message", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);

                if (check == DialogResult.Yes)
                {

                    try
                    {
                        connect.Open();
                        DateTime today = DateTime.Today;
string deleteData = "UPDATE teachers SET date_delete = @dateDelete " +
                            "WHERE teacher_id = @teacherID";

                        using (SqlCommand cmd = new SqlCommand(deleteData, connect))
                        {
                            cmd.Parameters.AddWithValue("@dateDelete", today);
                            cmd.Parameters.AddWithValue("@teacherID", teacher_id.Text.Trim());

                            cmd.ExecuteNonQuery();

                            teacherDisplayData();

                            MessageBox.Show("Deleted successfully!", "Information Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                            clearFields();
                        }
```

```
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error  connecting Database: " + ex, "Error Message"
                , MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    connect.Close();
                }
            }
            else
            {
                MessageBox.Show("Cancelled.", "Information Message"
                , MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }
    }
}

private void exelFileToolStripMenuItem_Click_1(object sender, EventArgs e)
{
    try
    {
        if (connect.State != ConnectionState.Open)
        {
            connect.Open();
        }

        string query = "SELECT teacher_id, teacher_name, teacher_gender, teacher_address,
teacher_status FROM teachers";
        SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connect);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable);

        // Створення нового документа Excel
        Excel.Application excelApp = new Excel.Application();
        Excel.Workbook workbook = excelApp.Workbooks.Add();
        Excel.Worksheet worksheet = workbook.Worksheets[1];

        // Заповнення заголовків стовпців
        for (int i = 0; i < dataTable.Columns.Count; i++)
        {
            worksheet.Cells[1, i + 1] = dataTable.Columns[i].ColumnName;
            worksheet.Cells[1, i + 1].Font.Bold = true;
        }

        // Заповнення даних
        for (int i = 0; i < dataTable.Rows.Count; i++)
        {
```

```
            for (int j = 0; j < dataTable.Columns.Count; j++)
            {
                worksheet.Cells[i + 2, j + 1] = dataTable.Rows[i][j].ToString();
            }
        }

        // Збереження файлу
        string filePath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop),
"TeachersList.xlsx");
        workbook.SaveAs(filePath);
        workbook.Close();
        excelApp.Quit();

        MessageBox.Show("Файл Excel успішно створено на робочому столі", "Інформація",
MessageBoxButtons.OK, MessageBoxIcon.Information);

        connect.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Виникла помилка: " + ex.Message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void wordFileToolStripMenuItem_Click_1(object sender, EventArgs e)
{
    try
    {
        if (connect.State != ConnectionState.Open)
        {
            connect.Open();
        }
string query = "SELECT teacher_id, teacher_name, teacher_gender, teacher_address,
teacher_status FROM teachers";
        SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connect);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable);

        // Створення нового документа Word
        Word.Application wordApp = new Word.Application();
        Word.Document document = wordApp.Documents.Add();

        // Додавання заголовка
        Word.Paragraph para = document.Content.Paragraphs.Add();
        para.Range.Text = "Teachers List";
        para.Range.Font.Size = 24;
        para.Range.Font.Bold = 1;
        para.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;
        para.Range.InsertParagraphAfter();
```

```
            // Додавання таблиці
            Word.Table table = document.Tables.Add(para.Range, dataTable.Rows.Count + 1,
dataTable.Columns.Count);
            table.Borders.Enable = 1;

            // Заповнення заголовків стовпців
            for (int i = 0; i < dataTable.Columns.Count; i++)
            {
                table.Cell(1, i + 1).Range.Text = dataTable.Columns[i].ColumnName;
                table.Cell(1, i + 1).Range.Bold = 1;
                table.Cell(1, i + 1).Range.Font.Size = 12;
                table.Cell(1, i + 1).Shading.BackgroundPatternColor =
Word.WdColor.wdColorGray25;
            }

            // Заповнення даних
            for (int i = 0; i < dataTable.Rows.Count; i++)
            {
                for (int j = 0; j < dataTable.Columns.Count; j++)
                {
                    table.Cell(i + 2, j + 1).Range.Text = dataTable.Rows[i][j].ToString();
                    table.Cell(i + 2, j + 1).Range.Font.Size = 12;
                }
            }

            // Збереження файлу
            string filePath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop),
"TeachersList.docx");
            document.SaveAs2(filePath);
            document.Close();
            wordApp.Quit();

            MessageBox.Show("Файл Word успішно створено на робочому столі", "Інформація",
MessageBoxButtons.OK, MessageBoxIcon.Information);

            connect.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Виникла помилка: " + ex.Message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void search_Click(object sender, EventArgs e)
    {
        string searchValue = search_text.Text.ToLower();
        CurrencyManager currencyManager =
(CurrencyManager)BindingContext[teacher_gridData.DataSource];
```

```
        currencyManager.SuspendBinding();

        try
        {
            foreach (DataGridViewRow row in teacher_gridData.Rows)
            {
                row.Visible = row.Cells.Cast<DataGridViewCell>()
                    .Any(c => c.Value != null &&
c.Value.ToString().ToLower().Contains(searchValue));
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        finally
        {
            currencyManager.ResumeBinding();
        }
    }
  }
}
```

Код форми DashboardForm

```
using System;
using System.ComponentModel;
using System.Data;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace SchoolMangementSystem
{
  public partial class DashboardForm : UserControl
  {
    SqlConnection connect = new
SqlConnection("Server=tcp:anna123.database.windows.net,1433;Initial Catalog=mukota;Persist
Security Info=False;User
ID=anna;Password=11082003Nik;MultipleActiveResultSets=False;Encrypt=True;TrustServerCer
tificate=False;Connection Timeout=30;");
    private BackgroundWorker backgroundWorker1;
    public DashboardForm()
    {
      InitializeComponent();

      displayTotalES();
      displayTotalTT();
      displayTotalGS();
      displayEnrolledStudentToday();

    }
```

```
    public void displayTotalES()
    {
        if(connect.State != ConnectionState.Open)
        {
            try
            {
                connect.Open();
                string selectData = "SELECT COUNT(id) FROM students WHERE student_status =
'Enrolled' AND date_delete IS NULL";

                using (SqlCommand cmd = new SqlCommand(selectData, connect))
                {
                    SqlDataReader reader = cmd.ExecuteReader();
                    int tempES = 0;
                    if (reader.Read())
                    {
                        tempES = Convert.ToInt32(reader[0]);
                        total_ES.Text = tempES.ToString();
                    }
                }

            }
            catch(Exception ex)
            {
                MessageBox.Show("Error to connect Database: " + ex, "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);

            }
            finally
            {
                connect.Close();
            }
        }
    }

    public void displayTotalTT()
    {
        if (connect.State != ConnectionState.Open)
        {
            try
            {
                connect.Open();
                string selectData = "SELECT COUNT(id) FROM teachers WHERE teacher_status =
'Active' AND date_delete IS NULL";

                using (SqlCommand cmd = new SqlCommand(selectData, connect))
                {
                    SqlDataReader reader = cmd.ExecuteReader();
                    int tempTT = 0;
                    if (reader.Read())
                    {
```

```
                tempTT = Convert.ToInt32(reader[0]);

                total_TT.Text = tempTT.ToString();
            }
        }

    }
    catch (Exception ex)
    {
        MessageBox.Show("Error to connect Database: " + ex, "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);

    }
    finally
    {
        connect.Close();
    }
}
}

public void displayTotalGS()
{
    if (connect.State != ConnectionState.Open)
    {
        try
        {
            connect.Open();
            string selectData = "SELECT COUNT(id) FROM students WHERE student_status =
'Graduated' AND date_delete IS NULL";

            using (SqlCommand cmd = new SqlCommand(selectData, connect))
            {
                SqlDataReader reader = cmd.ExecuteReader();
                int tempGS = 0;
                if (reader.Read())
                {
                    tempGS = Convert.ToInt32(reader[0]);

                    total_GS.Text = tempGS.ToString();
                }
            }

        }
        catch (Exception ex)
        {
            MessageBox.Show("Error to connect Database: " + ex, "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}
        finally
        {
            connect.Close();
```

```
            }
        }
    }

    public void displayEnrolledStudentToday()
    {
        AddStudentData asData = new AddStudentData();

        dataGridView1.DataSource = asData.dashboardStudentData();
    }

    private void DashboardForm_Load(object sender, EventArgs e)
    {

    }

    private void total_ES_TextChanged(object sender, EventArgs e)
    {
        displayTotalES();
    }
  }
}
```

Код форми Form1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SchoolMangementSystem
{
  public partial class Form1 : Form
  {
    public Form1()
    {
        InitializeComponent();
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        panel2.Width += 6;

        if(panel2.Width >= 525)
        {
            timer1.Stop();

            LoginForm lForm = new LoginForm();
```

```
            lForm.Show();
            this.Hide();
        }
    }
  }
}
```

Код форми LoginForm

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data;
using System.Data.SqlClient;

namespace SchoolMangementSystem
{
   public partial class LoginForm : Form
   {
      SqlConnection connect = new
SqlConnection(@"Server=tcp:anna123.database.windows.net,1433;Initial
Catalog=mukota;Persist Security Info=False;User
ID=anna;Password=11082003Nik;MultipleActiveResultSets=False;Encrypt=True;TrustServerCer
tificate=False;Connection Timeout=30;");
      public LoginForm()
      {
         InitializeComponent();
      }

      private void label1_Click(object sender, EventArgs e)
      {
         Application.Exit();
      }

      private void loginBtn_Click(object sender, EventArgs e)
      {
         if (username.Text == "" || password.Text == "")
         {
            MessageBox.Show("Please fill all blank fields", "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
         }
         else
         {
            if (comboBox_role.SelectedItem.ToString()== "Administrator")
         {

               try
```

```
            {
                connect.Open();

                String selectData = "SELECT * FROM users WHERE username = @username
AND password = @password";

                using (SqlCommand cmd = new SqlCommand(selectData, connect))
                {
                    cmd.Parameters.AddWithValue("@username", username.Text.Trim());
                    cmd.Parameters.AddWithValue("@password", password.Text.Trim());
                    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
                    DataTable table = new DataTable();
                    adapter.Fill(table);

                    if (table.Rows.Count >= 1)
                    {
                        MessageBox.Show("Login Successfully!", "Information Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                        MainForm mForm = new MainForm();
                        mForm.Show();
                        this.Hide();
                    }
                    else
                    {
                        MessageBox.Show("Incorrect Username/Password", "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);

                    }
                }

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error connecting Database: " + ex, "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);

            }
            finally
            {
                connect.Close();
            }

        }
        else if (comboBox_role.SelectedItem.ToString() == "Teacher")
        {
            try
            {
                connect.Open();
```

```
                String selectData = "SELECT * FROM teachers WHERE teacher_id = @username
AND teacher_password = @password";

                using (SqlCommand cmd = new SqlCommand(selectData, connect))
                {
                    cmd.Parameters.AddWithValue("@username", username.Text.Trim());
                    cmd.Parameters.AddWithValue("@password", password.Text.Trim());

                    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
                    DataTable table = new DataTable();
                    adapter.Fill(table);
if (table.Rows.Count >= 1)
                    {
                        using (SqlDataReader myReader = cmd.ExecuteReader())
                        {
                            while(myReader.Read())
                                {
                                int id = Int32.Parse(username.Text.Trim());
                                string name = myReader["teacher_name"].ToString();
                                MessageBox.Show("Login Successfully!", "Information Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                                TeacherForm tForm = new TeacherForm(id, name);
                                tForm.Show();
                                this.Hide();
                            }

                        }

                    }
                    else
                    {
                        MessageBox.Show("Incorrect Username/Password", "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);

                    }
                }

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error connecting Database: " + ex, "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);

            }
            finally
            {
                connect.Close();
            }
        }
        else if (comboBox_role.SelectedItem.ToString() == "Student")
```

```
        {
            try
            {
                connect.Open();

                String selectData = "SELECT * FROM students WHERE id = @username AND
student_password = @password";

                using (SqlCommand cmd = new SqlCommand(selectData, connect))
                {
                    cmd.Parameters.AddWithValue("@username", username.Text.Trim());
                    cmd.Parameters.AddWithValue("@password", password.Text.Trim());

                    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
                    DataTable table = new DataTable();
                    adapter.Fill(table);

                    if (table.Rows.Count >= 1)
                    {
                        using (SqlDataReader myReader = cmd.ExecuteReader())
                        {
                            while (myReader.Read())
                            {
                                int id = Int32.Parse(username.Text.Trim());
                                string name = myReader["student_name"].ToString();
                                MessageBox.Show("Login Successfully!", "Information Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);

                                StudentForm sForm = new StudentForm(id, name);
                                sForm.Show();
                                this.Hide();
                            }

                        }

                    }
                    else
                    {
                        MessageBox.Show("Incorrect Username/Password", "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);

                    }
                }

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error connecting Database: " + ex, "Error Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);

            }
```

```
            finally
            {
                connect.Close();
            }
        }

    }
}
private void showPass_CheckedChanged(object sender, EventArgs e)
    {
        password.PasswordChar = showPass.Checked ? '\0' : '*';
    }
  }
}
```

Код форми MainForm

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SchoolMangementSystem
{
  public partial class MainForm : Form
  {
    public MainForm()
    {
      InitializeComponent();
    }

    private void button3_Click(object sender, EventArgs e)
    {

    }

    private void button4_Click(object sender, EventArgs e)
    {
      dashboardForm1.Visible = false;
      addStudentForm1.Visible = false;
      addTeachersForm1.Visible = true;
      addTeachersForm1.Update();
    }

    private void label3_Click(object sender, EventArgs e)
    {
      Application.Exit();
    }
```

```
        private void button3_Click_1(object sender, EventArgs e)
        {
            DialogResult check = MessageBox.Show("Are you sure you want to logout?",
"Confirmation Message", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

            if(check == DialogResult.Yes)
            {
                LoginForm lForm = new LoginForm();
                lForm.Show();
                this.Hide();
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            DashboardForm dForm = new DashboardForm();
            dForm.displayEnrolledStudentToday();
            dForm.displayTotalGS();
            dForm.displayTotalTT();
            dForm.displayTotalES();

            dashboardForm1.Visible = true;
            dashboardForm1.Update();
            addStudentForm1.Visible = false;
            addTeachersForm1.Visible = false;
        }

        private void AddStudent_btn_Click(object sender, EventArgs e)
        {
            dashboardForm1.Visible = false;
            addStudentForm1.Visible = true;
            addStudentForm1.Update();
            addTeachersForm1.Visible = false;
        }

        private void btn_addSubject_Click(object sender, EventArgs e)
        {
            dashboardForm1.Visible = false;
            addStudentForm1.Visible = false;
            addSubjectForm1.Visible = true;
            addSubjectForm1.Update();
            addTeachersForm1.Visible = false;
        }


    }
}
```

Код форми TeacherForm

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SchoolMangementSystem
{
   public partial class TeacherForm : Form
   {
      string connect = "Server=tcp:anna123.database.windows.net,1433;Initial
Catalog=mukota;Persist Security Info=False;User
ID=anna;Password=11082003Nik;MultipleActiveResultSets=False;Encrypt=True;TrustServerCer
tificate=False;Connection Timeout=30;";
      private int TeacherId;
      private string Name;
      public TeacherForm(int id,string name)
      {

         Name = name;
         TeacherId = id;
         InitializeComponent();
         LoadTeacherSubjects();
         subjectsGridView.SelectionChanged += subjectsGridView_SelectionChanged;
      }
      private void button3_Click(object sender, EventArgs e)
      {
         DialogResult check = MessageBox.Show("Are you sure you want to logout?",
"Confirmation Message", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

         if (check == DialogResult.Yes)
         {
            LoginForm lForm = new LoginForm();
            lForm.Show();
            this.Hide();
         }

      }

      private void LoadTeacherSubjects()
      {
         using (SqlConnection conn = new SqlConnection(connect))
         {
            string query = "SELECT s.id, s.Name FROM subject s " +
                     "INNER JOIN teacher_subject ts ON s.id = ts.subject_id "+
                      "WHERE ts.teacher_id = @TeacherId";
            int a = TeacherId;
            SqlCommand cmd = new SqlCommand(query, conn);
```

```
            cmd.Parameters.AddWithValue("@TeacherId", TeacherId);

            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataTable subjectsTable = new DataTable();
            adapter.Fill(subjectsTable);

            subjectsGridView.DataSource = subjectsTable;
        }
    }

    private void subjectsGridView_SelectionChanged(object sender, EventArgs e)
    {
        if (subjectsGridView.SelectedRows.Count > 0)
        {
            int subjectId = Convert.ToInt32(subjectsGridView.SelectedRows[0].Cells["id"].Value);
            LoadStudents(subjectId);
        }
    }
    private void studentGridView_SelectionChanged(object sender, EventArgs e)
    {
        if (subjectsGridView.SelectedRows.Count > 0)
        {
            int studentId = Convert.ToInt32(subjectsGridView.SelectedRows[0].Cells["id"].Value);
        }
    }
    private void saveMarkButton_Click(object sender, EventArgs e)
    {
        if (studentsGridView.SelectedRows.Count > 0 && subjectsGridView.SelectedRows.Count >
0)
        {
            int studentId = Convert.ToInt32(studentsGridView.SelectedRows[0].Cells["id"].Value);
            int subjectId = Convert.ToInt32(subjectsGridView.SelectedRows[0].Cells["id"].Value);
            int mark;

            if (int.TryParse(selectmarks.Text, out mark))
            {
                SaveMark(studentId, subjectId, mark);
            }
            else
            {
                MessageBox.Show("Please enter a valid mark.");
            }
        }
    }

    private void SaveMark(int studentId, int subjectId, int mark)
    {
        string connectionString = "Server=tcp:anna123.database.windows.net,1433;Initial
Catalog=mukota;Persist Security Info=False;User
ID=anna;Password=11082003Nik;MultipleActiveResultSets=False;Encrypt=True;TrustServerCer
tificate=False;Connection Timeout=30;";
```

```
        try
        {
            using (SqlConnection conn = new SqlConnection(connectionString))
            {
                conn.Open();
// Перевірка наявності оцінки для студента і предмету
                string checkQuery = @"
            SELECT COUNT(*)
            FROM student_marks
            WHERE student_id = @StudentId AND teacher_subject_id = @SubjectId";

                using (SqlCommand checkCmd = new SqlCommand(checkQuery, conn))
                {
                    checkCmd.Parameters.AddWithValue("@StudentId", studentId);
                    checkCmd.Parameters.AddWithValue("@SubjectId", subjectId);

                    int count = (int)checkCmd.ExecuteScalar();

                    if (count > 0)
                    {
                        // Оновлення існуючої оцінки
                        string updateQuery = @"
                    UPDATE student_marks
                    SET mark = @Mark, data = @Date
                    WHERE student_id = @StudentId AND teacher_subject_id = @SubjectId";

                        using (SqlCommand updateCmd = new SqlCommand(updateQuery, conn))
                        {
                            updateCmd.Parameters.AddWithValue("@Mark", mark);
                            updateCmd.Parameters.AddWithValue("@Date", DateTime.Now);
                            updateCmd.Parameters.AddWithValue("@StudentId", studentId);
                            updateCmd.Parameters.AddWithValue("@SubjectId", subjectId);

                            updateCmd.ExecuteNonQuery();
                        }
                    }
                    else
                    {
                        // Вставка нової оцінки
                        string insertQuery = @"
                    INSERT INTO student_marks (teacher_subject_id, mark, data, student_id)
                    VALUES (@SubjectId, @Mark, @Date, @StudentId)";

                        using (SqlCommand insertCmd = new SqlCommand(insertQuery, conn))
                        {
                            insertCmd.Parameters.AddWithValue("@SubjectId", subjectId);
                            insertCmd.Parameters.AddWithValue("@Mark", mark);
                            insertCmd.Parameters.AddWithValue("@Date", DateTime.Now);
                            insertCmd.Parameters.AddWithValue("@StudentId", studentId);
```

```
                    insertCmd.ExecuteNonQuery();
                }
            }
        }
    }

    MessageBox.Show("Mark saved successfully!", "Information", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    LoadStudents(subjectId); // Оновлення відображення студентів після збереження
оцінки
}
catch (Exception ex)
{
    MessageBox.Show("Error saving mark: " + ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
private void LoadStudents(int subjectId)
{
    using (SqlConnection conn = new SqlConnection(connect))
    {
        string query = @"
        SELECT
            st.id,
            st.student_name,
            st.student_grade,
            st.student_section,
            sm.mark,
            sm.data
        FROM
            students st
        INNER JOIN
            student_subject ss ON st.id = ss.student_id
        LEFT JOIN
            student_marks sm ON st.id = sm.student_id AND ss.subject_id =
sm.teacher_subject_id
        WHERE
            ss.subject_id = @SubjectId";

        SqlCommand cmd = new SqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@SubjectId", subjectId);

        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        DataTable studentsTable = new DataTable();
        adapter.Fill(studentsTable);
studentsGridView.DataSource = studentsTable;
    }
}
private void label3_Click(object sender, EventArgs e)
{
    Application.Exit();
```

```
        }
    }
}
```

Код форми StudentForm

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SchoolMangementSystem
{
    public partial class StudentForm : Form
    {
        string connect = "Server=tcp:anna123.database.windows.net,1433;Initial
Catalog=mukota;Persist Security Info=False;User
ID=anna;Password=11082003Nik;MultipleActiveResultSets=False;Encrypt=True;TrustServerCer
tificate=False;Connection Timeout=30;";
        private int studentId;
        private string Name;
        public StudentForm(int id, string name)
        {
            Name = name;
            studentId = id;
            InitializeComponent();
            LoadStudentMarks(id);
        }

        private void label1_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            DialogResult check = MessageBox.Show("Are you sure you want to logout?",
"Confirmation Message", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

            if (check == DialogResult.Yes)
            {
                LoginForm lForm = new LoginForm();
                lForm.Show();
                this.Hide();
            }
        }
```

```
        private void LoadStudentMarks(int studentId)
        {
            try
            {
                using (SqlConnection connection = new SqlConnection(connect))
                {
                    string query = @"SELECT s.student_name, su.Name AS subject_name, sm.mark,
sm.data
                                     FROM students s
                                     JOIN student_subject ss ON s.id = ss.student_id
                                     JOIN teacher_subject ts ON ss.subject_id = ts.subject_id
                                     JOIN subject su ON ts.subject_id = su.id
                                     JOIN student_marks sm ON ts.id = sm.teacher_subject_id
                                     WHERE s.id = @studentId
                                     AND s.date_delete IS NULL"; // Завантаження даних тільки для конкретного
студента та активних студентів

                    SqlCommand command = new SqlCommand(query, connection);
                    command.Parameters.AddWithValue("@studentId", studentId);
                    SqlDataAdapter adapter = new SqlDataAdapter(command);
                    DataTable dataTable = new DataTable();

                    adapter.Fill(dataTable);

                    studentGridView.DataSource = dataTable;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error loading student marks: " + ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

    }
}
```