Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Н[авчально-науковий інститут комп'ютерних наук та інформаційних технологій](https://web.kpi.kharkov.ua/if/uk/uaabout)

Кафедра «Комп’ютерна інженерія та програмування»

**Пояснювальна записка до проекту**

з дисципліни "Програмна технологія DOT NET"

за темою

«Студентський кабінет»

Виконав:

студент групи КН-1022в

Гасюк М.О.

Перевірив:

старший викладач

Івашко І.В.

Харків-2024

**РЕФЕРАТ**

**Проект:**" Студентський кабінет "

**Технології:**

* .NET Framework
* WinForms
* JSON
* Git, GitHub
* CI/CD інтеграція (з використанням GitHub Actions)
* Twilio API
* iText

**Опис роботи:**

Метою індивідуального проекту було створення інформаційної системи "Студентський кабінет", яка забезпечує автоматизацію та оптимізацію процесів управління студентськими даними.

**Реалізовано функціонал:**

1. **Обробка студентських даних** з використанням форм, що дозволяють додавати, редагувати та видаляти записи.
2. **Зберігання даних у форматі JSON**, забезпечуючи їх легкий обмін і модифікацію.
3. **Інтеграція з Twilio API** для відправлення SMS-повідомлень.
4. **Генерація звітів** у форматі PDF за допомогою бібліотеки iText, що дає змогу отримувати детальну статистику.
5. **Робота з GitHub** для керування версіями проекту.
6. **CI/CD інтеграція** з автоматизацією збірки та тестування через GitHub Actions.

У результаті була створена програма "Студентський кабінет", яка спрощує роботу з інформацією для студентів і викладачів. Інтерфейс системи забезпечує:

* **Пошук**, **фільтрацію** та **сортування** даних.
* **Запис на курси**, **перегляд оцінок** за семестри.
* **Звіти й статистика** для адміністративного управління.

**ЗМІСТ**

[Вступ 4](#_Toc154212627)

[1 Аналіз та концептуальне моделювання предметної областІ 5](#_Toc154212628)

[1.1Аналіз предметної області 5](#_Toc154212629)

[1.2 Аналіз існуючих аналогів 5](#_Toc154212630)

[2 Постановка задачі 7](#_Toc154212632)

[3 Структура проекту 9](#_Toc154212633)

[4 Опис форм 10](#_Toc154212637)

[5 Сервіси для роботи с даними 38](#_Toc154212638)

[6 Класи для роботи з студентами 34](#_Toc154212639)

[7 Використання TWILIOAPI 42](#_Toc154212640)

[8 Опис структури DataProvider.json 46](#_Toc154212641)

[9 Післярелізна підтримка 49](#_Toc154212642)

[Висновки 54](#_Toc154212643)

ВСТУП

У сучасному світі, де технології невпинно вдосконалюються, автоматизація процесів і оптимізація взаємодії стають невід’ємною частиною нашого повсякденного життя. Цифрові інструменти знаходять застосування у різних сферах, спрощуючи складні процеси і дозволяючи досягати високих результатів із мінімальними витратами часу та ресурсів.

Одним із ключових напрямків розвитку інформаційних систем є створення платформ для голосування та опитувань. У сучасному суспільстві прийняття рішень часто базується на демократичному підході, що вимагає швидкого, зручного та надійного збору думок. Традиційні методи опитувань часто є недостатньо ефективними через свою громіздкість і часові затрати.

Метою проекту «Система голосування та опитувань» є забезпечення ефективного механізму для збору, аналізу та візуалізації результатів опитувань. Створена система покликана спростити взаємодію між організаторами голосувань і їх учасниками, забезпечуючи швидкість і прозорість процесу.

Реалізація проекту базується на використанні сучасних вебтехнологій, таких як PHP, MySQL, HTML, CSS і JavaScript, що дозволяє створити надійну, функціональну та інтуїтивно зрозумілу платформу. Інтеграція з GitHub і використання CI/CD-процесів сприяє автоматизації розробки, а бібліотека iText забезпечує зручну генерацію звітів.

Система надає користувачам інструменти для створення опитувань, участі в них, а також аналізу даних. Організатори отримують можливість гнучкого управління опитуваннями, а учасники — простий і зручний спосіб висловити свою думку. Проект спрямований на підвищення ефективності, демократичності та інтерактивності процесів збору даних і прийняття рішень.

1 АНАЛІЗ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

**1.1 Аналіз предметної області**

Предметна область програми «Студентський кабінет» охоплює управління студентськими даними, забезпечуючи доступ до інформації про успішність, розклад занять, записи на курси та інші функції.

Основними об’єктами управління є:

* **Студенти**, їх персональні дані, оцінки та розклад;
* **Навчальні дисципліни** та їх характеристики;
* **Викладачі**, які проводять заняття;
* **Адміністрація**, яка забезпечує управління студентськими даними.

Процеси, що автоматизуються:

* Внесення, редагування та видалення студентських даних;
* Формування та перегляд звітів за семестрами;
* Відправлення повідомлень студентам через Twilio API.

Система забезпечує автоматизацію цих процесів, дозволяючи ефективно обробляти дані, оптимізуючи роботу адміністрації та забезпечуючи студентів зручним доступом до навчальної інформації.

**1.2 Аналіз існуючих аналогів**

У сучасному світі активно впроваджуються інноваційні системи для управління освітніми процесами. Серед таких — системи дистанційного навчання (LMS) та університетські портали.

**LMS (Learning Management System)**

Системи LMS забезпечують управління освітнім процесом, включаючи дистанційне навчання, облік оцінок та розклад занять.

Переваги:

* Централізація навчальної інформації;
* Зручний доступ до матеріалів курсу;
* Можливість дистанційного навчання.

Недоліки:

* Висока вартість впровадження для навчальних закладів;
* Обмежена гнучкість для налаштування під потреби студентів.

**Університетські портали**

Ці портали забезпечують доступ до інформації про успішність студентів, розкладу занять та інших адміністративних функцій.

Переваги:

* Інтеграція з іншими університетськими системами;
* Зручний доступ для студентів і викладачів.

Недоліки:

* Застарілі інтерфейси у багатьох існуючих системах;
* Відсутність персоналізації для студентів.

**Відмінності системи «Студентський кабінет»**

* Інтуїтивно зрозумілий інтерфейс, розроблений на основі WinForms;
* Використання сучасних технологій, таких як Twilio API, для швидкої комунікації;
* Можливість формування звітів у форматі PDF;
* Простота впровадження та використання, завдяки формату JSON для зберігання даних.

Система «Студентський кабінет» забезпечує швидкий, зручний і персоналізований доступ до інформації для студентів і адміністрації, вирішуючи недоліки існуючих аналогів та відповідаючи сучасним потребам користувачів.

**2 ПОСТАНОВКА** ЗАДАЧІ

Задачею **індивідуального завдання** є проектування та реалізація інформаційної системи (ІС) для «Студентського кабінету» з використанням JSON-файлів для зберігання даних.

Для реалізації інформаційної системи необхідно розробити логічну модель даних, забезпечити уникнення дублювання інформації, а також зберігати та оновлювати дані у JSON-файлах із дотриманням їхньої цілісності та структурованості.

Для кожного студента зберігаються такі дані:

* П.І.Б.;
* група;

Дані про навчальні дисципліни включають:

* назву предмета;
* номер семестру;
* оцінки за ІДЗ, Тести;
* середній бал.

Ця інформаційна система призначена для автоматизації збереження та управління даними про студентів, оцінки, дисципліни, викладачів і семестри у форматі JSON.

**Функціонал інформаційної системи:**

* додавання нового студента до JSON-файлу;
* видалення студента з JSON-файлу;
* внесення оцінок за дисципліни, лабораторні роботи, модулі та підрахунок середнього балу;
* автоматичне формування відомостей із результатами успішності студента за вибраний семестр;
* пошук інформації у JSON-файлах за такими критеріями:
  + пошук студентів за ім’ям, групою;
  + пошук предметів за назвою або номером семестру;
* фільтрування та сортування результатів пошуку:
  + сортування студентів за алфавітом;

**Запити, які підтримує система:**

* отримання статистики успішності студентів за всі семестри;

**Формування звітів:**

Система має генерувати:

* **PDF-звіти** для кожного студента з інформацією про його успішність за вибраний семестр;

**Автоматизація процесів:**

* формування електронних звітів про успішність студентів;

Завданням розробки цієї інформаційної системи є оптимізація навчального процесу, автоматизація роботи зі студентськими даними та полегшення взаємодії між студентами, викладачами та адміністрацією.

**3 СТРУКТУРА ПРОЕКТУ**

У цьому розділі наведена структура проекту, яка організована для забезпечення ефективного збереження, обробки та управління даними студентів у JSON-файлах, а також реалізації необхідного функціоналу.

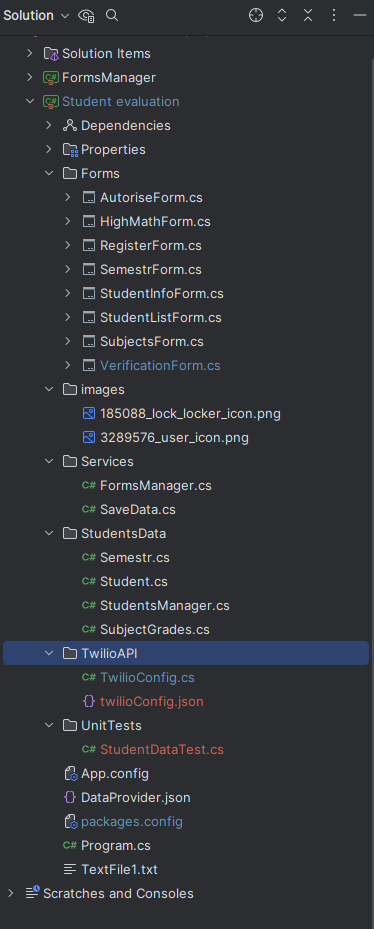


Рисунок 1.1 – Структура проекту

**4 ОПИС ФОРМ**

4.1 Форма авторизації (*AutoriseForm*)

Ця форма відповідає за авторизацію користувачів у системі. Користувач вводить ім'я та пароль, після чого формується подальший доступ до системи залежно від ролі користувача.

1. Ініціалізація форм:

Форма ініціалізує два основні об'єкти для авторизації:

* VerificationForm для вчителя
* StudentInfoForm для студента

2. Кнопка входу (login):

string username **=** loginField**.**Text**;**

string password **=** passField**.**Text**;**

**if** **(**username **==** \_allowedName **&&** password **==** \_allowedPassword**)**

**{**

\_studentForm**.**Show**();**

**this.**Visible **=** **false;**

**}**

**else** **if** **(**username **==** "teacher" **&&** password **==** \_allowedPassword**)**

**{**

\_verifyForm**.**Show**();**

**this.**Visible **=** **false;**

**}**

**else**

**{**

MessageBox**.**Show**(**"Неправильне ім'я користувача або пароль. Спробуйте знову."**);**

**}**

Цей код відповідає за перевірку введених даних і відкриває відповідну форму для студента або вчителя, якщо дані правильні. Якщо дані введені неправильно, відображається повідомлення про помилку.

3. **Кнопка закриття форми:**

**private** void closeButton\_Click**(object** sender**,** EventArgs e**)**

**{**

**this.**Close**();**

**}**

Ця кнопка дозволяє закрити форму авторизації.



Рисунок 1.2 – Форма авторизації

Це основний функціонал форми авторизації.

**4.2 Форма верифікації (*VerificationForm*)**

Ця форма використовується для верифікації користувачів через код, що надсилається на мобільний телефон. Після введення правильного коду, користувач отримує доступ до списку студентів.

**1. Ініціалізація форми:**

* Генерація випадкового коду для верифікації.
* Ініціалізація конфігурації для Twilio та надсилання SMS з токеном на зазначений номер телефону.

Public VerificationForm**()**

**{**

InitializeComponent**();**

Random random **=** **new** Random**();**

Token1 **=** random**.**Next**(**1000**,** 10000**).**ToString**();**

var twilioConfigString **=** File**.**ReadAllText**(**"twilioConfig.json"**);**

var twilioConfig **=** JsonSerializer**.**Deserialize**<**TwilioConfig**>(**twilioConfigString**);**

TwilioClient**.**Init**(**twilioConfig**.**AccountSID**,** twilioConfig**.**AuthToken**);**

MessageResource**.**Create**(**

body**:** "Your token is " **+** Token1 **+** ""**,**

**from:** **new** Twilio**.**Types**.**PhoneNumber**(**"+14158559808"**),**

to**:** **new** Twilio**.**Types**.**PhoneNumber**(**"+380986294196"**));**

Console**.**WriteLine**(**"Sms was sent"**);**

\_registerForm **=** **new** RegisterForm**();**

\_studentListForm **=** **new** StudentListForm**();**

**}**

Цей код генерує випадковий токен і надсилає його на телефон за допомогою Twilio.

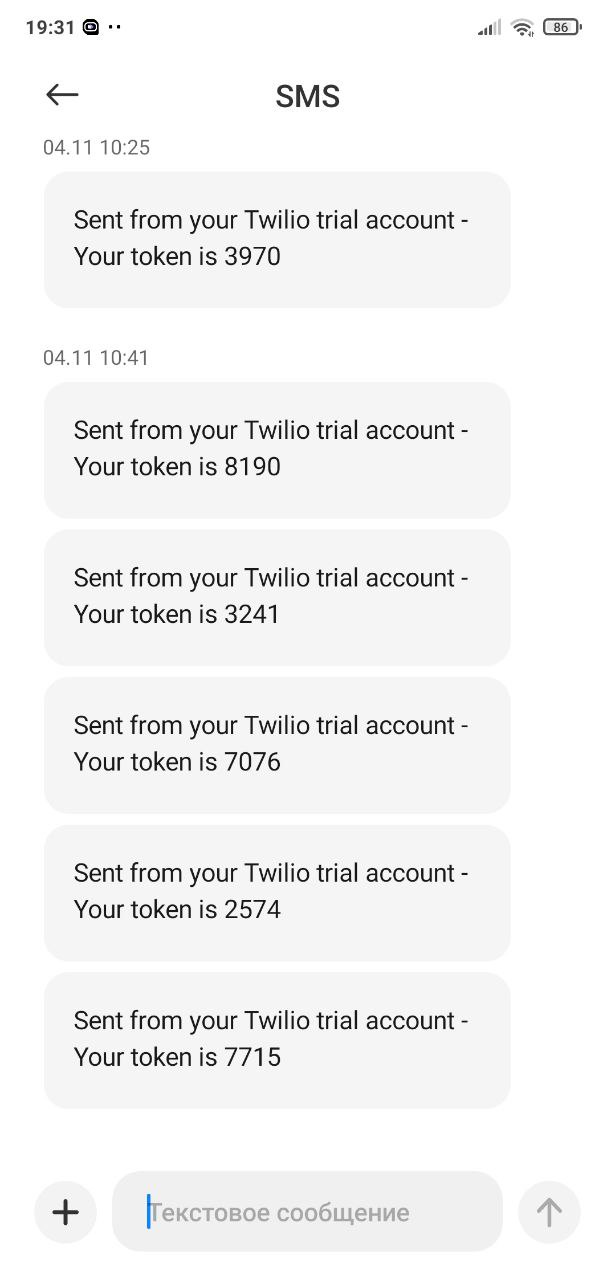


Рисунок 1.3 – Результати роботи Twilio

2. **Кнопка верифікації:**

**private** void button2\_Click**(object** sender**,** EventArgs e**)**

**{**

string token **=** SmsField**.**Text**;**

**if** **(**token **==** Token1**)**

**{**

MessageBox**.**Show**(**"Вхід виконано успішно!"**);**

**this.**Close**();**

\_studentListForm**.**Show**();**

**}**

**else**

**{**

MessageBox**.**Show**(**"Ви не пройшли верифікацію"**);**

**}**

**}**

Ця кнопка перевіряє введений токен. Якщо він збігається з надісланим, відкривається форма для перегляду списку студентів.

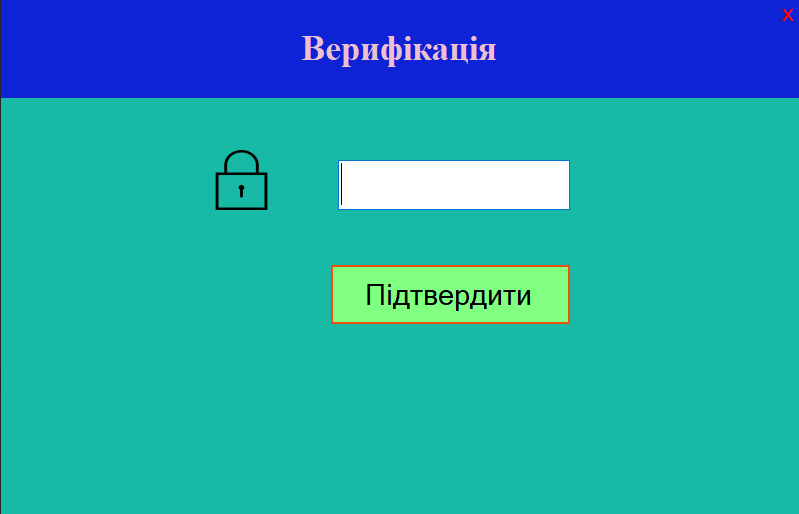


Рисунок 1.4 – Форма верифікації

**4.3 Форма інформації про студента (*StudentInfoForm*)**

Ця форма дозволяє користувачам (студентам) переглядати їхні дані та генерувати звіт у форматі PDF.

**1. Ініціалізація та завантаження студентів:**

Форма ініціалізує список студентів, які зберігаються в системі:

**private** void LoadStudents**()**

**{**

\_students **=** SaveData**.**LoadStudents**();**

UpdateComboBoxItems**(**string**.**Empty**);**

**}**

**2. Текстове поле для вибору студента:**

Користувач вводить прізвище студента у текстове поле, після чого відображається список студентів, що починаються з цього тексту:

**private** void comboBoxStudents\_TextChanged**(object** sender**,** EventArgs e**)**

**{**

string input **=** comboBoxStudents**.**Text**;**

UpdateComboBoxItems**(**input**);**

**}**

**3. Обробка вибору студента:**

Коли користувач вибирає студента зі списку, система зберігає вибір:

**private** void comboBoxStudents\_SelectedIndexChanged**(object** sender**,** EventArgs e**)**

**{**

var selectedText **=** comboBoxStudents**.**SelectedItem**?.**ToString**();**

\_student **=** \_students**.**FirstOrDefault**(**s **=>**

$"{s.Surname} {s.Name} {s.LastName}" **==** selectedText**);**

**}**

**4. Кнопка генерування PDF-звіту:**

Ця кнопка генерує PDF-звіт для обраного студента, що містить інформацію про його семестри та оцінки:

**private** void button4\_Click**(object** sender**,** EventArgs e**)**

**{**

var selectedText **=** comboBoxStudents**.**Text**;**

\_student **=** \_students**.**FirstOrDefault**(**s **=>**

$"{s.Surname} {s.Name} {s.LastName}" **==** selectedText**);**

**if** **(**\_student **!=** **null)**

**{**

GeneratePdfReport**();**

**}**

**else**

**{**

MessageBox**.**Show**(**"Студента не знайдено. Будь ласка, виберіть коректного студента."**);**

**}**

**}**

**5. Метод генерування PDF:**

Метод генерує PDF-документ з даними студента, що містить особисту інформацію та деталі його семестрів і оцінок:

**private** void GeneratePdfReport**()**

**{**

string currentDate **=** DateTime**.**Now**.**ToString**(**"yyyy\\MM\\dd\_HH.mm.ss"**);**

string filePath **=** $"StudentReport\_{\_student.Surname}\_{\_student.Name}\_{currentDate}.pdf"**;**

**if** **(**string**.**IsNullOrEmpty**(**filePath**)** **||** Path**.**GetInvalidPathChars**().**Any**(**filePath**.**Contains**))**

**{**

MessageBox**.**Show**(**"Неправильний шлях до файлу."**);**

**return;**

**}**

**using** **(**PdfWriter writer **=** **new** PdfWriter**(**filePath**))**

**using** **(**PdfDocument pdf **=** **new** PdfDocument**(**writer**))**

**{**

Document **=** **new** Document**(**pdf**);**

PdfFont font **=** PdfFontFactory**.**CreateFont**(**"C:\\Windows\\Fonts\\arial.ttf"**,** "Identity-H"**);**

document**.**Add**(new** Paragraph**(**$"Студент: {\_student.Surname} {\_student.Name} {\_student.LastName}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**$"Група: {\_student.Group}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**"Семестри:"**).**SetFont**(**font**));**

**foreach** **(**var semester **in** \_student**.**Semesters**)**

**{**

document**.**Add**(new** Paragraph**(**$"Семестр {semester.SemesterNumber}:"**).**SetFont**(**font**));**

**if** **(**semester**.**Grades**.**Count **>** 0**)**

**{**

**foreach** **(**var grade **in** semester**.**Grades**)**

**{**

var subject **=** grade**.**Key**;**

var details **=** grade**.**Value**;**

document**.**Add**(new** Paragraph**(**$" Предмет: {subject}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**$" Перше ІДЗ: {details.FirstHW}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**$" Перший тест: {details.FirstTest}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**$" Друге ІДЗ: {details.SecondHW}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**$" Средній бал: {details.AverageScore:F2}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**$" Оцінка: {details.LetterGrade}"**).**SetFont**(**font**));**

**}**

**}**

**else**

**{**

document**.**Add**(new** Paragraph**(**"Немає оцінок за ці семестр"**).**SetFont**(**font**));**

**}**

**}**

document**.**Close**();**

**}**

**try**

**{**

Process**.**Start**(new** ProcessStartInfo

**{**

FileName **=** filePath**,**

UseShellExecute **=** **true**

**});**

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**$"Не вдалося відкрити PDF документ: {ex.Message}"**);**

**}**

MessageBox**.**Show**(**"PDF-звіт успішно згенеровано за шляхом: " **+** Path**.**GetFullPath**(**filePath**));**

**}**

**6. Кнопка виходу на форму авторизації:**

Кнопка, що дозволяє повернутися на форму авторизації:

**private** void button1\_Click**(object** sender**,** EventArgs e**)**

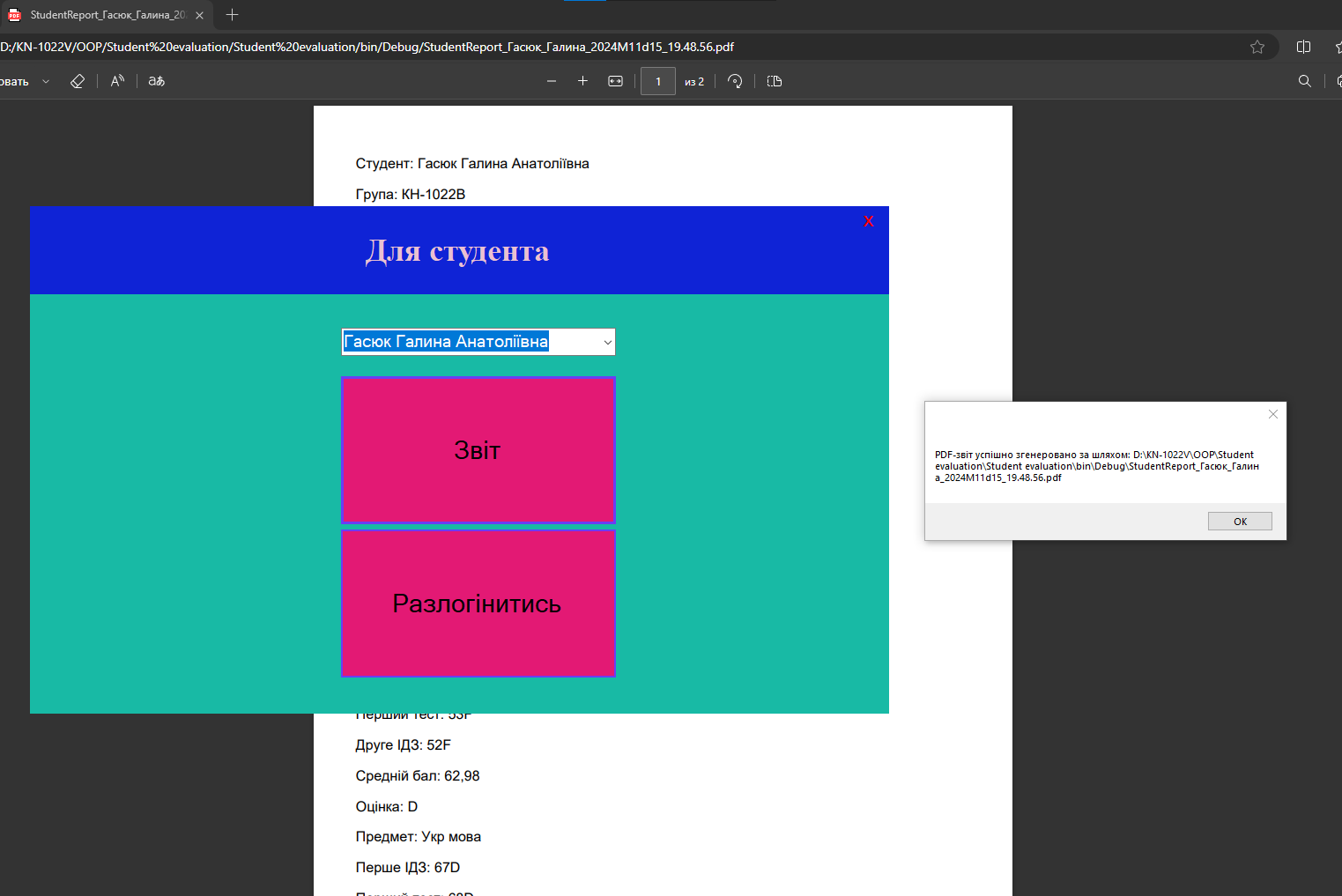
**{**

\_autoriseForm**.**Visible **=** **true;**

**this.**Close**();**

**}**

Рисунок 1.5 – Пошук студента

Рисунок 1.6 – Генерація звіту студентом

Це основні функціональні елементи форми **StudentInfoForm**, яка дозволяє студенту переглядати та генерувати звіт.

**4.4 Форма взаємодії зі студентами (*StudentListForm*)**

Ця форма призначена для відображення списку студентів, організованих по групах, а також надає функціональність редагування, видалення та пошуку студентів.

**Основні елементи:**

1. **Контекстне меню для студентів:** Форма включає контекстне меню для роботи з елементами списку студентів (викликається правою кнопкою миші):
   * **Видалити:** видаляє вибраного студента зі списку.
   * **Редагувати:** відкриває форму для редагування даних студента.
   * **Звіт:** відкриває форму семестру для вибраного студента.

Код для створення контекстного меню:

\_contextMenu **=** **new** ContextMenuStrip**();**

var deleteMenuItem **=** **new** ToolStripMenuItem**(**"Видалити"**,** **null,** DeleteStudent**);**

var editMenuItem **=** **new** ToolStripMenuItem**(**"Редагувати"**,** **null,** EditStudent**);**

var semestrMenuItem **=** **new** ToolStripMenuItem**(**"Звіт"**,** **null,** OpenSemesterForm**);**

\_contextMenu**.**Items**.**Add**(**deleteMenuItem**);**

\_contextMenu**.**Items**.**Add**(**editMenuItem**);**

\_contextMenu**.**Items**.**Add**(**semestrMenuItem**);**

**Кнопки для оновлення і реєстрації студентів:**

* **Оновити:** перевантажує список студентів з файлу та відображає його в ListBox.
* **Реєстрація:** відкриває форму для додавання або редагування студента.

Код для кнопок:

**private** void UpdateButton\_Click**(object** sender**,** EventArgs e**)**

**{**

LoadStudentsIntoListBoxes**();**

**}**

**private** void RegisterButton\_Click**(object** sender**,** EventArgs e**)**

**{**

var registerForm **=** **new** RegisterForm**();**

registerForm**.**Show**();**

**}**

**Функціональність Drag-and-Drop для зміни груп студентів:** Студентів можна перетягувати між ListBox для зміни їх групи. При цьому група оновлюється в файлі даних.

Код для обробки подій Drag-and-Drop:

**private** void listBox\_DragDrop**(object** sender**,** DragEventArgs e**)**

**{**

ListBox targetListBox **=** **(**ListBox**)**sender**;**

string studentDisplayName **=** **(**string**)**e**.**Data**.**GetData**(typeof(**string**));**

**foreach** **(**ListBox **in** **new** ListBox**[]** **{** listBox1**,** listBox2**,** listBox3 **})**

**{**

**if** **(**listBox**.**Items**.**Contains**(**studentDisplayName**))**

**{**

listBox**.**Items**.**Remove**(**studentDisplayName**);**

**break;**

**}**

**}**

targetListBox**.**Items**.**Add**(**studentDisplayName**);**

UpdateStudentGroup**(**studentDisplayName**,** targetListBox**);**

**}**

**Пошук студента за прізвищем:** Користувач може шукати студента за прізвищем, і якщо студент знайдений, його запис буде виділений в списку.

Код для пошуку: **private** void button2\_Click**(object** sender**,** EventArgs e**)**

**{**

string surnameToSearch **=** textBox1**.**Text**.**Trim**();**

**if** **(**string**.**IsNullOrWhiteSpace**(**surnameToSearch**))**

**{**

MessageBox**.**Show**(**"Введіть прізвище для пошуку."**);**

**return;**

**}**

ListBox targetListBox **=** **null;**

**foreach** **(**ListBox **in** **new** ListBox**[]** **{** listBox1**,** listBox2**,** listBox3 **})**

**{**

**foreach** **(**var item **in** listBox**.**Items**)**

**{**

var studentDisplayName **=** item**.**ToString**();**

var studentData **=** studentDisplayName**.**Split**(**' '**);**

**if** **(**studentData**.**Length **>** 0 **&&** studentData**[**0**].**Equals**(**surnameToSearch**,** StringComparison**.**OrdinalIgnoreCase**))**

**{**

targetListBox **=** listBox**;**

listBox**.**SelectedItem **=** item**;**

**break;**

**}**

**}**

**if** **(**targetListBox **!=** **null)**

**break;**

**}**

**if** **(**targetListBox **==** **null)**

**{**

MessageBox**.**Show**(**"Студента з таким прізвищем не знайдено."**);**

**}**

**}**



Рисунок 1.7 – Пошук студента



Рисунок 1.8 – Додаткове меню

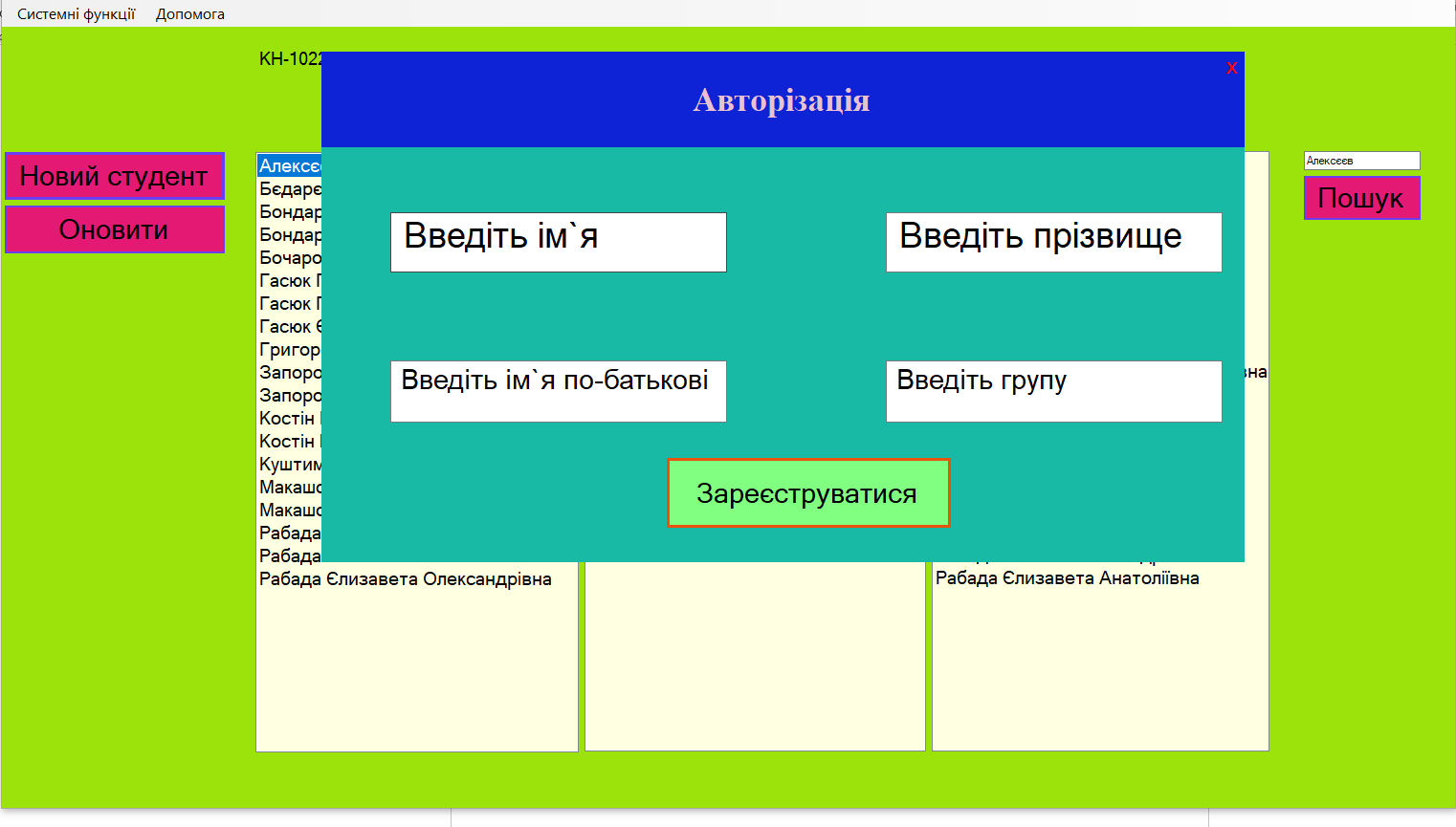


Рисунок 1.9 – Створення нового студенту

Рисунок 1.10 – Використання контекстного меню

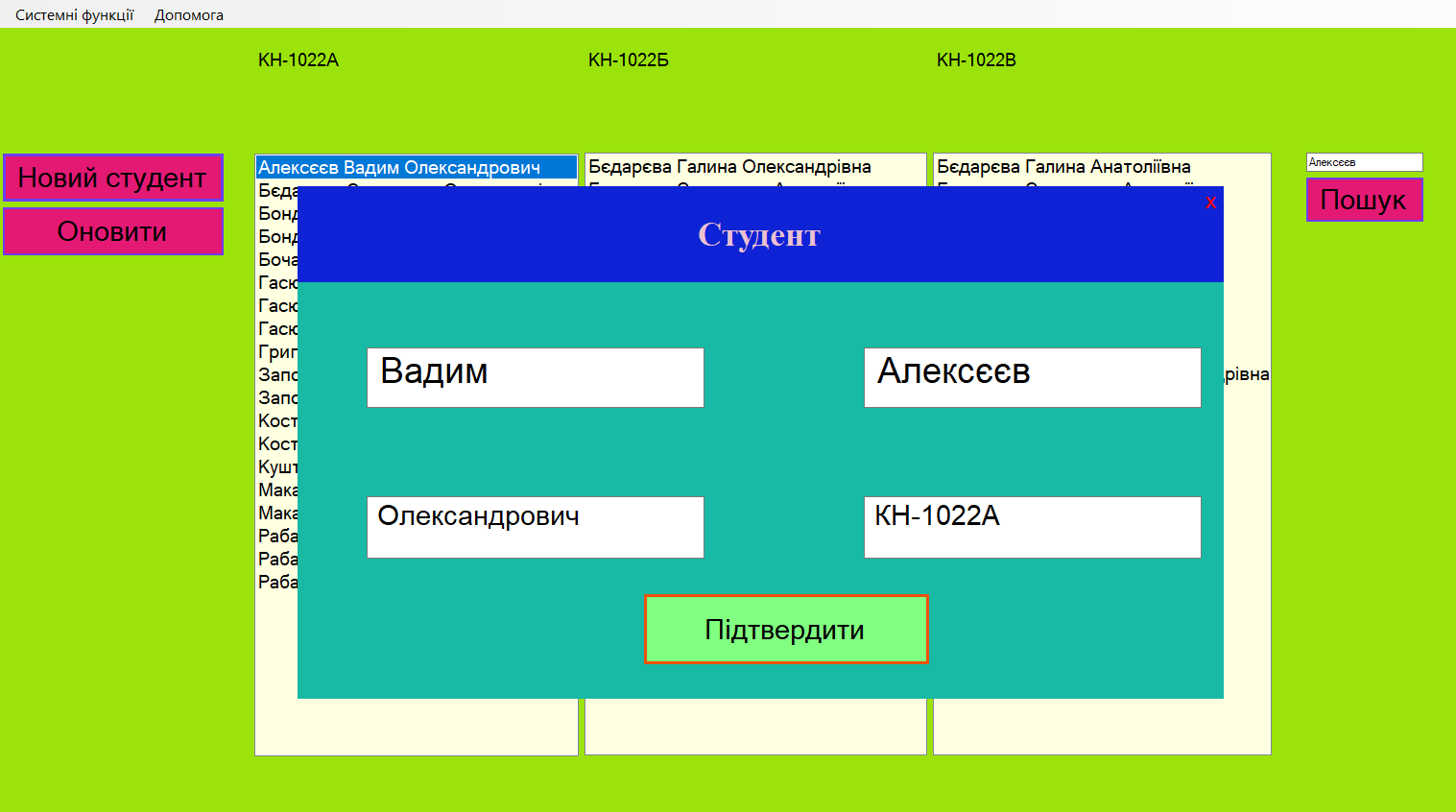


Рисунок 1.11 – Редагування існуючого студента

Рисунок 1.12 – Створення звіту про студента(Дабл клік -> Звіт)



Рисунок 1.13 – Обираємо семестр та натискаєм на Результат

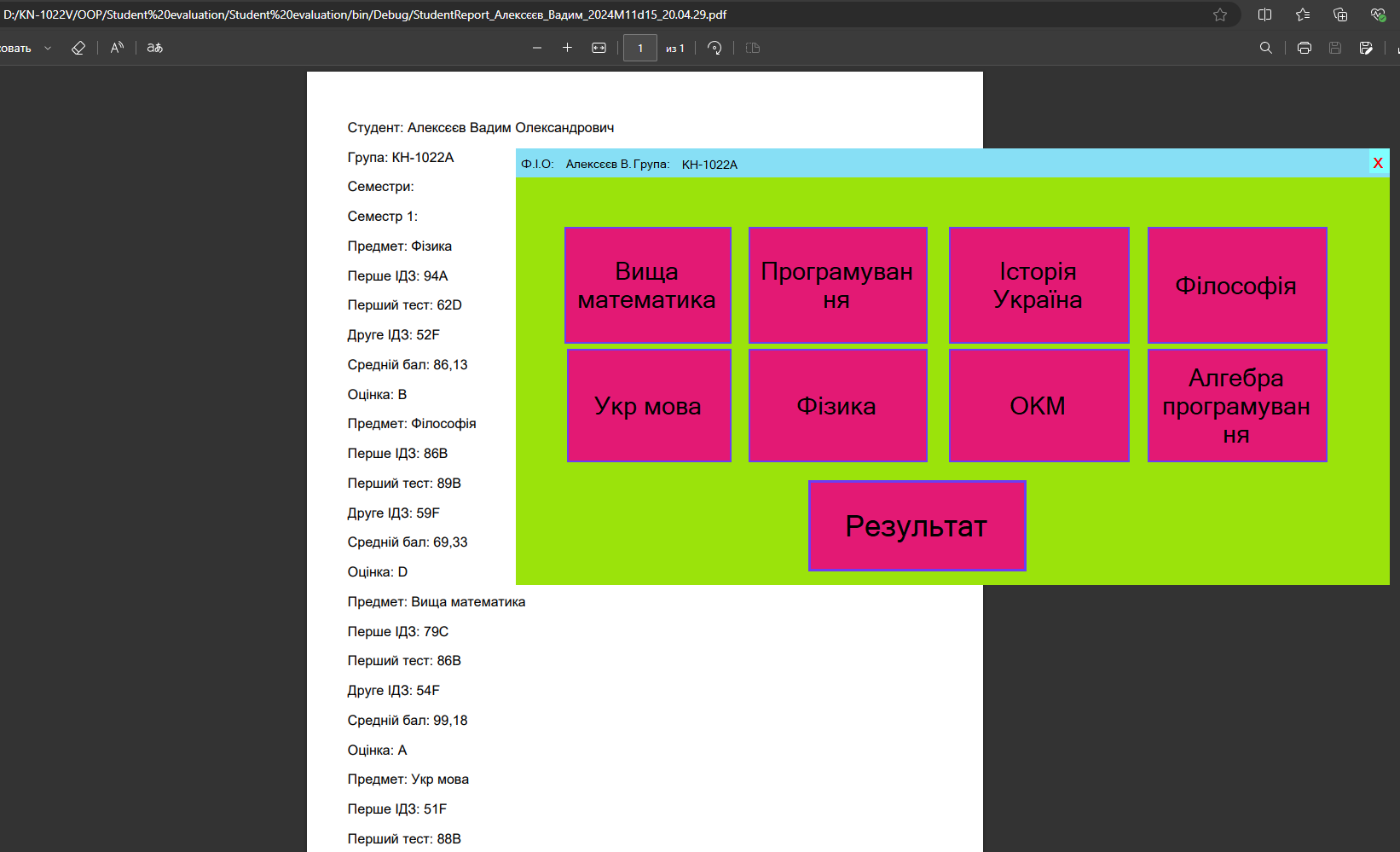


Рисунок 1.14 – Генеруємо звіт про студента

**4.5 Форма реєстрації/редагування студента (*RegisterForm*)**

Ця форма використовується для реєстрації нового студента або редагування існуючого, залежно від того, чи активований режим редагування.

* 1. **Конструктор форми:**

**public** RegisterForm**(**bool editMode **=** **false,** Student studentToEdit **=** **null)**

**{**

InitializeComponent**();**

userNameField**.**Text **=** "Введіть ім`я"**;**

userSurnameField**.**Text **=** "Введіть прізвище"**;**

userLastnameField**.**Text **=** "Введіть ім`я по-батькові"**;**

userGroupField**.**Text **=** "Введіть групу"**;**

isEditMode **=** editMode**;**

**if** **(**isEditMode **&&** studentToEdit **!=** **null)**

**{**

userNameField**.**Text **=** studentToEdit**.**Name**;**

userSurnameField**.**Text **=** studentToEdit**.**Surname**;**

userLastnameField**.**Text **=** studentToEdit**.**LastName**;**

userGroupField**.**Text **=** studentToEdit**.**Group**;**

button1**.**Text **=** "Підтвердити"**;**

label1**.**Text **=** "Студент"**;**

originalDisplayName **=** $"{studentToEdit.Surname} {studentToEdit.Name} {studentToEdit.LastName}"**;**

**}**

**}**

Конструктор ініціалізує поля форми. Якщо форма працює в режимі редагування, поля заповнюються даними студента, якому належать ці значення.

* 1. **Поля вводу:**

**private** void userNameField\_Enter**(object** sender**,** EventArgs e**)**

**{**

**if** **(**userNameField**.**Text **==** "Введіть ім`я"**)**

**{**

userNameField**.**Text **=** ""**;**

userNameField**.**ForeColor **=** Color**.**Black**;**

**}**

**}**

**private** void userNameField\_Leave**(object** sender**,** EventArgs e**)**

**{**

**if** **(**userNameField**.**Text **==** ""**)**

**{**

userNameField**.**Text **=** "Введіть ім`я"**;**

userNameField**.**ForeColor **=** Color**.**Gray**;**

**}**

**}**

**private** void userSurnameField\_Enter**(object** sender**,** EventArgs e**)**

**{**

**if** **(**userSurnameField**.**Text **==** "Введіть прізвище"**)**

**{**

userSurnameField**.**Text **=** ""**;**

userSurnameField**.**ForeColor **=** Color**.**Black**;**

**}**

**}**

**private** void userSurnameField\_Leave**(object** sender**,** EventArgs e**)**

**{**

**if** **(**userSurnameField**.**Text **==** ""**)**

**{**

userSurnameField**.**Text **=** "Введіть прізвище"**;**

userSurnameField**.**ForeColor **=** Color**.**Gray**;**

**}**

**}**

Ці методи відповідають за очищення текстових полів при фокусі і повернення стандартного тексту, якщо поле залишилося порожнім після виходу з фокуса.

* 1. **Кнопка підтвердження:**

**private** void button1\_Click**(object** sender**,** EventArgs e**)**

**{**

Student **=** **new** Student

**{**

Name **=** userNameField**.**Text**,**

Surname **=** userSurnameField**.**Text**,**

LastName **=** userLastnameField**.**Text**,**

Group **=** userGroupField**.**Text

**};**

**if** **(**isEditMode**)**

**{**

SaveData**.**EditStudent**(**originalDisplayName**,** student**);** // Редактируем существующего студента

FormsManager**.**OpenForm**(new** SemestrForm**(**student**));**

**this.**Visible **=** **false;**

**}**

**else**

**{**

SaveData**.**AddStudent**(**student**);** // Добавляем нового студента

FormsManager**.**OpenForm**(new** SemestrForm**(**student**));**

**this.**Visible **=** **false;**

**}**

**}**

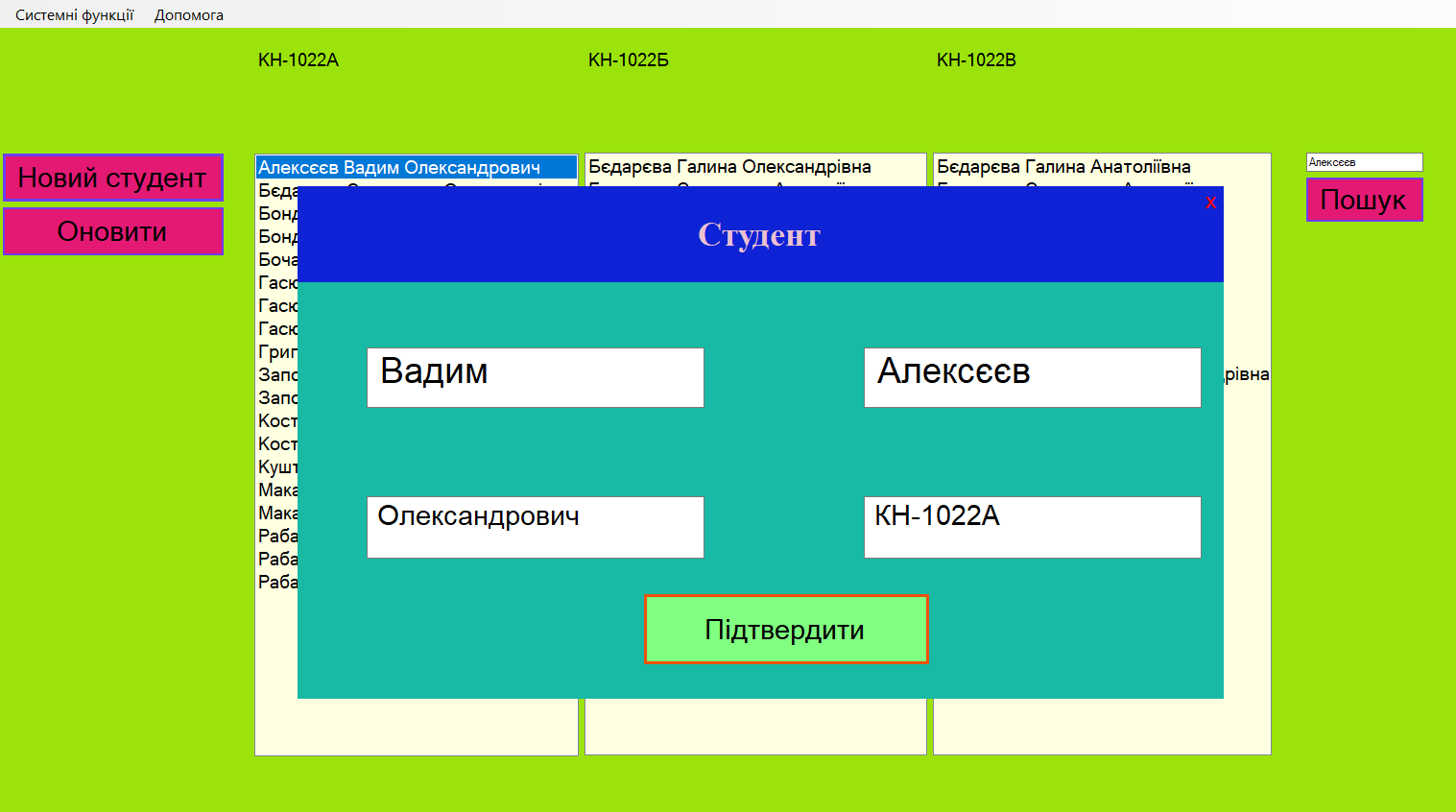
Ця кнопка зберігає дані нового студента або редагує існуючого, після чого відкриває форму для введення інформації про семестр.

Рисунок 1.15 – Редагування існуючого студента



Рисунок 1.16 – Створення нового студенту

**4.6 Форма семестра (*SemestrForm*)**

Ця форма використовується для роботи зі студентами та їхніми семестровими даними. Вона дозволяє додавати нові семестри для студента та відображати деяку основну інформацію про нього.

**1. Ініціалізація:**

У конструкторі форми отримується інформація про студента, що передається в параметрах, і генерується коротке ім'я студента:

\_student **=** student**;**

\_shortName **=** GenerateShortName**(**\_student**.**Surname**,** \_student**.**Name**,** \_student**.**LastName**);**

label2**.**Text **=** \_shortName**;**

label4**.**Text **=** \_student**.**Group**;**

**2. Генерація короткого імені:**

Метод GenerateShortName використовується для формування короткого імені студента у форматі: Прізвище І. Б.

**protected** string GenerateShortName**(**string lastName**,** string firstName**,** string middleName**)**

**{**

string firstLetterOfFirstName **=** **!**string**.**IsNullOrEmpty**(**firstName**)** **?** firstName**.**Substring**(**0**,** 1**)** **+** "." **:** string**.**Empty**;**

string firstLetterOfMiddleName **=** **!**string**.**IsNullOrEmpty**(**middleName**)** **?** middleName**.**Substring**(**0**,** 1**)** **+** "." **:** string**.**Empty**;**

**return** $"{lastName} {firstLetterOfFirstName}{firstLetterOfMiddleName}"**;**

**}**

**3. Кнопка "Семестр 1":**

При натисканні на кнопку додається новий семестр для студента і відкривається форма для введення предметів:

**private** void button4\_Click**(object** sender**,** EventArgs e**)**

**{**

**this.**Visible **=** **false;**

var newSemester **=** **new** Semester

**{**

SemesterNumber **=** "1"**,**

Grades **=** **new** Dictionary**<**string**,** SubjectGrades**>()**

**};**

\_student**.**Semesters**.**Add**(**newSemester**);**

FormsManager**.**OpenForm**(new** SubjectsForm**(**\_student**,** "1"**,** **this));**

**}**

**4. Кнопка "Семестр 2":**

При натисканні на кнопку додається другий семестр для студента:

**private** void button8\_Click**(object** sender**,** EventArgs e**)**

**{**

**this.**Visible **=** **false;**

var newSemester **=** **new** Semester

**{**

SemesterNumber **=** "2"**,**

Grades **=** **new** Dictionary**<**string**,** SubjectGrades**>()**

**};**

\_student**.**Semesters**.**Add**(**newSemester**);**

FormsManager**.**OpenForm**(new** SubjectsForm**(**\_student**,** "2"**,** **this));**

**}**



Рисунок 1.17 – Форма для семестрів зі студентом

Ця форма дозволяє додавати нові семестри студенту та обробляти дані для подальшої роботи.

**4.7 Форма предметів (*SubjectsForm*)**

Ця форма дозволяє студенту вибирати предмети для конкретного семестру. Кожен предмет відкриває відповідну форму для введення або перегляду оцінок.

**1. Ініціалізація форми:**

Форма отримує об'єкти студента, семестру та форму, з якої була викликана ця форма. Встановлюються значення для імені студента та його групи:

\_student **=** student**;**

\_semesterNumber **=** semesterNumber**;**

\_semestrForm **=** semestrForm**;**

\_shortName **=** GenerateShortName**(**\_student**.**Surname**,** \_student**.**Name**,** \_student**.**LastName**);**

label2**.**Text **=** \_shortName**;**

label4**.**Text **=** \_student**.**Group**;**

2. **Метод генерації скороченого імені:**

**protected** string GenerateShortName**(**string lastName**,** string firstName**,** string middleName**)**

**{**

string firstLetterOfFirstName **=** **!**string**.**IsNullOrEmpty**(**firstName**)** **?** firstName**.**Substring**(**0**,** 1**)** **+** "." **:** string**.**Empty**;**

string firstLetterOfMiddleName **=** **!**string**.**IsNullOrEmpty**(**middleName**)** **?** middleName**.**Substring**(**0**,** 1**)** **+** "." **:** string**.**Empty**;**

**return** $"{lastName} {firstLetterOfFirstName}{firstLetterOfMiddleName}"**;**

**}**

Цей метод генерує коротке ім'я студента, яке відображається на формі.

**3. Кнопки вибору предметів:**

Кожна кнопка відповідає за відкриття відповідної форми для предмету. Коли студент натискає на одну з кнопок, відкривається форма для введення або перегляду оцінок з відповідного предмету:

**private** void button10\_Click**(object** sender**,** EventArgs e**)**

**{**

\_Subject **=** "Вища математика"**;**

FormsManager**.**OpenForm**(new** HighMathForm**(**\_student**,** \_semesterNumber**,** \_shortName**,** \_Subject**));**

**}**

І так для кожного предмета, наприклад:

**private** void button1\_Click**(object** sender**,** EventArgs e**)**

**{**

\_Subject **=** "Прогамування"**;**

FormsManager**.**OpenForm**(new** HighMathForm**(**\_student**,** \_semesterNumber**,** \_shortName**,** \_Subject**));**

**}**

4. **Кнопка закриття форми:**

**private** void closeButton\_Click**(object** sender**,** EventArgs e**)**

**{**

\_semestrForm**.**Visible **=** **true;**

**this.**Close**();**

**}**

Ця кнопка дозволяє повернутися до попередньої форми (форми семестру) та закрити поточну форму предметів.

* + 1. **Кнопка для генерації PDF-звіту:**

**private** void button8\_Click**(object** sender**,** EventArgs e**)**

**{**

string selectedSemester **=** \_semesterNumber**;**

GeneratePdfReport**(**selectedSemester**);**

**}**

Ця кнопка генерує PDF-звіт для обраного семестру, який містить дані про студента та оцінки. Звіт зберігається в файл і відображається користувачу.

**6. Метод для генерації PDF-звіту:**

**private** void GeneratePdfReport**(**string selectedSemester**)**

**{**

string currentDate **=** DateTime**.**Now**.**ToString**(**"yyyy\\MM\\dd\_HH.mm.ss"**);**

string filePath **=** $"StudentReport\_{\_student.Surname}\_{\_student.Name}\_{currentDate}.pdf"**;**

**if** **(**string**.**IsNullOrEmpty**(**filePath**)** **||** Path**.**GetInvalidPathChars**().**Any**(**filePath**.**Contains**))**

**{**

MessageBox**.**Show**(**"Неправильний шлях до файлу"**);**

**return;**

**}**

**using** **(**PdfWriter writer **=** **new** PdfWriter**(**filePath**))**

**{**

**using** **(**PdfDocument pdf **=** **new** PdfDocument**(**writer**))**

**{**

Document **=** **new** Document**(**pdf**);**

PdfFont font **=** PdfFontFactory**.**CreateFont**(**"C:\\Windows\\Fonts\\arial.ttf"**,** "Identity-H"**);**

document**.**Add**(new** Paragraph**(**$"Студент: {\_student.Surname} {\_student.Name} {\_student.LastName}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**$"Група: {\_student.Group}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**$"Семестри:"**).**SetFont**(**font**));**

var semester **=** \_student**.**Semesters**.**FirstOrDefault**(**s **=>** s**.**SemesterNumber **==** selectedSemester**);**

**if** **(**semester **!=** **null)**

**{**

document**.**Add**(new** Paragraph**(**$"Семестр {semester.SemesterNumber}:"**).**SetFont**(**font**));**

**if** **(**semester**.**Grades**.**Count **>** 0**)**

**{**

**foreach** **(**var grade **in** semester**.**Grades**)**

**{**

var subject **=** grade**.**Key**;**

var details **=** grade**.**Value**;**

document**.**Add**(new** Paragraph**(**$" Предмет: {subject}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**$" Перше ІДЗ: {details.FirstHW}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**$" Перший тест: {details.FirstTest}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**$" Друге ІДЗ: {details.SecondHW}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**$" Средній бал: {details.AverageScore:F2}"**).**SetFont**(**font**));**

document**.**Add**(new** Paragraph**(**$" Оцінка: {details.LetterGrade}"**).**SetFont**(**font**));**

**}**

**}**

**else**

**{**

document**.**Add**(new** Paragraph**(**$"Немає оцінок за цей семестр"**).**SetFont**(**font**));**

**}**

**}**

document**.**Close**();**

**}**

**}**

**try**

**{**

Process**.**Start**(new** ProcessStartInfo

**{**

FileName **=** filePath**,**

UseShellExecute **=** **true**

**});**

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**$"Не вдалося відкрити PDF документ: {ex.Message}"**);**

**}**

MessageBox**.**Show**(**"PDF-звіт успішно згенеровано за шляхом: " **+** Path**.**GetFullPath**(**filePath**));**

**}**

Цей метод створює звіт у форматі PDF з оцінками студента, що зберігається в файл та відкривається на комп'ютері користувача.



Рисунок 1.18 – Зовнішній вигляд форми студента

**4.8 Форма з оцінками (*HighMathForm*)**

Ця форма дозволяє вводити оцінки студентів для предметів " та обчислювати середній бал і буквене оцінювання.

1. **Ініціалізація:**

**public** HighMathForm**(**Student student**,** string semesterNumber**,** string shortName**,** string Subject**)**

**{**

InitializeComponent**();**

\_Subject **=** Subject**;**

\_student **=** student**;**

\_semesterNumber **=** semesterNumber**;**

\_shortName **=** shortName**;**

label2**.**Text **=** \_shortName**;**

label4**.**Text **=** \_student**.**Group**;**

label5**.**Text **=** Subject**;**

**}**

Цей код ініціалізує форму, встановлюючи значення для предмету, групи та імені студента.

**2. Кнопка для обчислення оцінок:**

**private** void button1\_Click**(object** sender**,** EventArgs e**)**

**{**

int firstHWScore **=** int**.**Parse**(**textBox1**.**Text**);**

int firstTestScore **=** int**.**Parse**(**textBox6**.**Text**);**

int secondHWScore **=** int**.**Parse**(**textBox14**.**Text**);**

double averageScore **=** **(**firstHWScore **+** firstTestScore **+** secondHWScore**)** **/** 3.0**;**

string letterGrade**;**

**if** **(**averageScore **>=** 90**)** letterGrade **=** "A"**;**

**else** **if** **(**averageScore **>=** 80**)** letterGrade **=** "B"**;**

**else** **if** **(**averageScore **>=** 70**)** letterGrade **=** "C"**;**

**else** **if** **(**averageScore **>=** 60**)** letterGrade **=** "D"**;**

**else** letterGrade **=** "F"**;**

label12**.**Text **=** averageScore**.**ToString**(**"F2"**);**

label13**.**Text **=** letterGrade**;**

var grades **=** **new** SubjectGrades

**{**

FirstHW **=** firstHWScore **+** textBox4**.**Text**,**

FirstTest **=** firstTestScore **+** textBox5**.**Text**,**

SecondHW **=** secondHWScore **+** textBox13**.**Text**,**

AverageScore **=** averageScore**,**

LetterGrade **=** letterGrade

**};**

SaveData**.**UpdateStudentGrades**(**\_student**.**Name**,** \_semesterNumber**,** \_Subject**,** grades**);**

**}**

Цей код обчислює середню оцінку на основі введених оцінок і визначає буквене оцінювання. Після цього оцінки зберігаються в JSON через метод SaveData.UpdateStudentGrades.

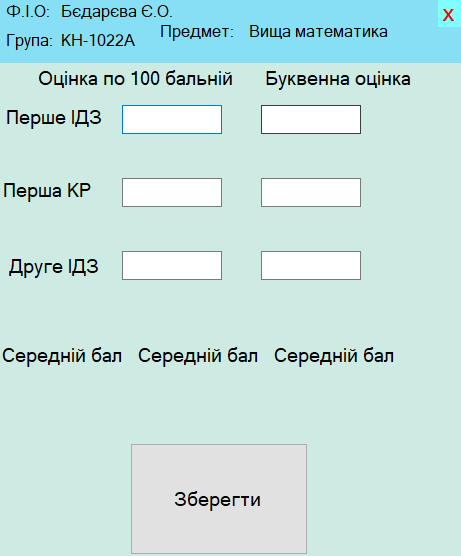


Рисунок 1.19 – Зовнішній вигляд форми для оцінок

**5. Сервіси для роботи з даними(Services)**

**5.1** **Сервіс для роботи з даними студентів** (*SaveData*)

Цей розділ відповідає за збереження, завантаження, додавання, редагування та видалення студентів у файлі JSON. Клас SaveData реалізує всі необхідні методи для роботи з даними студентів.

Методи класу SaveData:

Завантаження студентів з файлу:

**public** static List**<**Student**>** LoadStudents**()**

**{**

**if** **(!**File**.**Exists**(**FilePath**))**

**{**

System**.**Diagnostics**.**Debug**.**WriteLine**(**"File does not exist, creating a new list."**);**

**return** **new** List**<**Student**>();**

**}**

var json **=** File**.**ReadAllText**(**FilePath**);**

**if** **(**string**.**IsNullOrWhiteSpace**(**json**))**

**{**

System**.**Diagnostics**.**Debug**.**WriteLine**(**"File is empty, creating a new list."**);**

**return** **new** List**<**Student**>();**

**}**

System**.**Diagnostics**.**Debug**.**WriteLine**(**$"Loaded data: {json}"**);**

**return** JsonConvert**.**DeserializeObject**<**List**<**Student**>>(**json**)** **??** **new** List**<**Student**>();**

**}**

Цей метод завантажує список студентів з файлу DataProvider.json. Якщо файл не існує або він порожній, повертається порожній список.

**Збереження студентів у файл:**

**public** static void SaveStudents**(**List**<**Student**>** students**)**

**{**

System**.**Diagnostics**.**Debug**.**WriteLine**(**$"Saving students: {JsonConvert.SerializeObject(students, Formatting.Indented)}"**);**

var json **=** JsonConvert**.**SerializeObject**(**students**,** Formatting**.**Indented**);**

File**.**WriteAllText**(**FilePath**,** json**);**

**}**

Метод серіалізує список студентів в JSON та зберігає його у файл.

Додавання нового студента:

**public** static void AddStudent**(**Student student**)**

**{**

var students **=** LoadStudents**();**

students**.**Add**(**student**);**

System**.**Diagnostics**.**Debug**.**WriteLine**(**$"Adding student: {JsonConvert.SerializeObject(student, Formatting.Indented)}"**);**

SaveStudents**(**students**);**

**}**

Цей метод додає нового студента до списку та зберігає оновлений список у файл.

Оновлення оцінок студента:

**public** static void UpdateStudentGrades**(**string studentName**,** string semesterNumber**,** string subjectName**,** SubjectGrades grades**)**

**{**

var students **=** LoadStudents**();**

var student **=** students**.**FirstOrDefault**(**s **=>** s**.**Name **==** studentName**);**

**if** **(**student **!=** **null)**

**{**

var semester **=** student**.**Semesters**.**FirstOrDefault**(**s **=>** s**.**SemesterNumber **==** semesterNumber**);**

**if** **(**semester **==** **null)**

**{**

semester **=** **new** Semester **{** SemesterNumber **=** semesterNumber **};**

student**.**Semesters**.**Add**(**semester**);**

**}**

semester**.**Grades**[**subjectName**]** **=** grades**;**

SaveStudents**(**students**);**

**}**

**}**

Цей метод оновлює оцінки студента для конкретного семестру та предмету. Якщо семестр відсутній, він створюється.

Оновлення групи студента:

**public** static void UpdateStudentGroup**(**string studentDisplayName**,** string newGroup**)**

**{**

var students **=** LoadStudents**();**

var student **=** students**.**FirstOrDefault**(**s **=>** $"{s.Surname} {s.Name} {s.LastName}" **==** studentDisplayName**);**

**if** **(**student **!=** **null)**

**{**

student**.**Group **=** newGroup**;**

SaveStudents**(**students**);**

**}**

**}**

Цей метод дозволяє змінити групу студента, шукаючи його за повним іменем.

Видалення студента:

**public** static void RemoveStudent**(**string studentDisplayName**)**

**{**

var students **=** LoadStudents**();**

var student **=** students**.**FirstOrDefault**(**s **=>** $"{s.Surname} {s.Name} {s.LastName}" **==** studentDisplayName**);**

**if** **(**student **!=** **null)**

**{**

students**.**Remove**(**student**);**

SaveStudents**(**students**);**

**}**

**}**

Видаляє студента зі списку, шукаючи його за повним іменем.

Редагування студента:

**public** static void EditStudent**(**string originalDisplayName**,** Student updatedStudent**)**

**{**

var students **=** LoadStudents**();**

var student **=** students**.**FirstOrDefault**(**s **=>** $"{s.Surname} {s.Name} {s.LastName}" **==** originalDisplayName**);**

**if** **(**student **!=** **null)**

**{**

student**.**Name **=** updatedStudent**.**Name**;**

student**.**Surname **=** updatedStudent**.**Surname**;**

student**.**LastName **=** updatedStudent**.**LastName**;**

student**.**Group **=** updatedStudent**.**Group**;**

SaveStudents**(**students**);**

**}**

**}**

Цей метод дозволяє редагувати дані студента, зберігаючи зміни в файл.

Цей клас надає базову функціональність для зберігання та обробки студентських даних у вигляді JSON-файлу. Завдяки методам, як додавання, редагування, оновлення та видалення студентів, він забезпечує взаємодію з даними студентів у системі.

**5.2 Менеджер форм (*FormsManager*)**

Цей клас відповідає за керування відкриттям форм у додатку. Він дозволяє централізовано відкривати будь-яку форму за допомогою одного методу.

Методи класу FormsManager:

Метод для відкриття форми:

**public** static void OpenForm**(**Form form**)**

**{**

form**.**Show**();**

**}**

**6. Класи для роботи з студентами(StudentsData)**

У цьому розділі будуть знаходитися класи, які описують структури даних, пов'язані зі студентами, їх семестрами та оцінками. Перший клас цього розділу – Semester, який відповідає за семестр студента.

**6.1 Клас Semester**

Клас Semester описує один семестр студента, включаючи його номер та оцінки з різних предметів.

**public** class Semester

**{**

**public** string SemesterNumber **{** **get;** **set;** **}**

**public** Dictionary**<**string**,** SubjectGrades**>** Grades **{** **get;** **set;** **}** **=** **new** Dictionary**<**string**,** SubjectGrades**>();**

**}**

Опис полів:

* SemesterNumber: Це властивість зберігає номер семестру, наприклад, "1", "2".
* Grades: Це словник, в якому ключем є назва предмету (наприклад, "Математика", "Фізика"), а значенням є об'єкт типу SubjectGrades, який зберігає оцінки студента з цього предмету.

Цей клас забезпечує структуру для зберігання оцінок студента в різних семестрах. Для кожного семестру можна зберігати різні предмети з відповідними оцінками, що дозволяє динамічно додавати нові предмети та оновлювати оцінки.

**6.2 Клас Student**

Клас Student описує студента та його інформацію, включаючи його ім'я, прізвище, групу та список семестрів.

**public** class Student

**{**

**public** string Name **{** **get;** **set;** **}**

**public** string Surname **{** **get;** **set;** **}**

**public** string LastName **{** **get;** **set;** **}**

**public** string Group **{** **get;** **set;** **}**

**public** List**<**Semester**>** Semesters **{** **get;** **set;** **}** **=** **new** List**<**Semester**>();**

**}**

**Опис полів:**

* **Name**: Властивість для зберігання імені студента.
* **Surname**: Властивість для зберігання прізвища студента.
* **LastName**: Властивість для зберігання по батькові студента.
* **Group**: Властивість для зберігання групи, до якої належить студент.
* **Semesters**: Це список об'єктів класу **Semester**, кожен з яких відповідає за оцінки студента в певному семестрі. Це дозволяє зберігати та маніпулювати даними для кожного семестру окремо.

Цей клас дозволяє зберігати основну інформацію про студента, а також його оцінки по всіх семестрах.

**6.3 Клас StudentsManager**

Клас StudentsManager управляє списком студентів. Він дозволяє додавати студентів до колекції і зберігати їх для подальшого використання.

**public** class StudentsManager

**{**

**public** List**<**Student**>** Students **{** **get;** **private** **set;** **}** **=** **new** List**<**Student**>();**

**public** void AddStudent**(**Student student**)**

**{**

Students**.**Add**(**student**);**

**}**

**}**

Опис:

* **Students**: Це властивість, яка зберігає список студентів. Вона ініціалізується як порожній список. Ви можете додавати до цього списку нових студентів.
* **AddStudent**: Метод для додавання нового студента в список **Students**. Метод приймає об'єкт класу **Student** і додає його в колекцію.

Цей клас є корисним для управління групою студентів в програмі.

**6.4 Клас SubjectGrades**

Клас SubjectGrades представляє оцінки студента з певного предмета, зокрема домашні завдання, тести, середній бал та буквена оцінка.

**public** class SubjectGrades

**{**

**public** string FirstHW **{** **get;** **set;** **}**

**public** string FirstTest **{** **get;** **set;** **}**

**public** string SecondHW **{** **get;** **set;** **}**

**public** double AverageScore **{** **get;** **set;** **}**

**public** string LetterGrade **{** **get;** **set;** **}**

**}**

Опис:

* FirstHW: Оцінка або коментар до першої домашньої роботи. Це строкове значення, яке може містити числову оцінку або буквене значення.
* FirstTest: Оцінка або коментар до першого тесту. Як і FirstHW, це строкове значення.
* SecondHW: Оцінка або коментар до другої домашньої роботи.
* AverageScore: Середній бал студента за предмет (числове значення).
* LetterGrade: Буквена оцінка, що відображає загальний результат студента (наприклад, "A", "B", "C", "D", "F").

Цей клас дозволяє зберігати оцінки за конкретний предмет, що полегшує обчислення середнього балу та визначення відповідної буквеної оцінки для студента.

**7. Використання TwilioAPI**

**7.1 Клас TwilioConfig**

Клас TwilioConfig зберігає конфігураційні дані для підключення до API Twilio, зокрема **AccountSID** та **AuthToken**, які потрібні для аутентифікації при використанні Twilio API.

**internal** class TwilioConfig

**{**

**public** string AccountSID **{** **get;** **set;** **}**

**public** string AuthToken **{** **get;** **set;** **}**

**}**

Опис:

* **AccountSID**: Унікальний ідентифікатор облікового запису Twilio. Він використовується для аутентифікації при відправленні запитів до API.
* **AuthToken**: Токен аутентифікації, який разом з **AccountSID** дозволяє отримати доступ до моєго облікового запису Twilio і використовувати функціонал API.

7.2 **Конфігураційний JSON для Twilio**

Зазначений JSON може бути використаний для зберігання параметрів підключення до API Twilio у зовнішньому файлі конфігурації щоб зручно отримувати доступ до цих параметрів у програмі.

**{**

"AccountSID"**:** "AC9abaeb1f2551de309b675703d53df500"**,**

"AuthToken"**:** "bd2dcb01976c4a6619f205a1630cabcf"

**}**

**8.Опис структури DataProvider.json**

В файлі DataProvider.json зберігаються дані про студентів, які включають їхні особисті відомості, групу та оцінки за різні семестри. Кожен студент має такі атрибути:

1. **Ім'я (Name)** — ім'я студента.
2. **Прізвище (Surname)** — прізвище студента.
3. **По батькові (LastName)** — по батькові студента.
4. **Група (Group)** — номер групи, до якої належить студент.
5. **Семестри (Semesters)** — список об'єктів, що містять інформацію про семестри, у яких студент отримував оцінки.

У кожному семестрі зберігаються такі дані:

* **SemesterNumber** — номер семестра.
* **Grades** — оцінки по кожному предмету, де для кожного предмета є:
  + **FirstHW** — перша домашня робота.
  + **FirstTest** — перший тест.
  + **SecondHW** — друга домашня робота.
  + **AverageScore** — середній бал.
  + **LetterGrade** — оцінка в літерному вигляді.

Ось як виглядають дані для двох студентів у файлі:

1. **Галина Куштим Олександрівна** з групи **КН-1022Б**:
   * У першому семестрі вона оцінювалася з фізики та філософії, де отримала оцінки "F" (не складено) з фізики та "C" (задовільно) з філософії.
   * У другому семестрі її оцінки з вищої математики були "A" (відмінно), а з української мови — "B" (добре).
2. **Максим Бочаров Олександрович** з групи **КН-1022В**:
   * У першому семестрі його оцінки з фізики і філософії були середні — "F" і "C", а з вищої математики і української мови — "C".
   * У другому семестрі він покращив свої результати з фізики та філософії, отримавши "A" (відмінно), а з вищої математики та української мови результат залишився на рівні "D" (не задовільно).

Приклад студента:

{

"Name": "Максим",

"Surname": "Гасюк",

"LastName": "Олександрович",

"Group": "КН-1022Б",

"Semesters": [

{

"SemesterNumber": "1",

"Grades": {

"HigherMathematics": {

"FirstHW": "100A",

"FirstTest": "90A",

"SecondHW": "75C",

"AverageScore": 88.333333333333329,

"LetterGrade": "B"

},

"Алгебра програмування": {

"FirstHW": "60E",

"FirstTest": "80B",

"SecondHW": "70C",

"AverageScore": 70.0,

"LetterGrade": "C"

},

"Вища математика": {

"FirstHW": "55",

"FirstTest": "55",

"SecondHW": "55",

"AverageScore": 5.0,

"LetterGrade": "F"

},

"Фізика": {

"FirstHW": "55",

"FirstTest": "55",

"SecondHW": "51",

"AverageScore": 5.0,

"LetterGrade": "F"

}

}

}

]

}

Ця структура дозволяє чітко зберігати та обробляти дані про студентів, їх оцінки та прогрес у навчанні по різних предметах і семестрах.

**9. Післярелізна підтримка**

**9.1 Післярелізна підтримка з допомогою GitHub Actions**

Основні моменти налаштування:

1. **GitHub Actions** — це інструмент автоматизації, що дозволяє запускати певні дії на основі подій, таких як коміти в репозиторії. У нашому випадку це конфігурація для автоматичного запуску збірки.
2. **Workflow** — у GitHub Actions створено workflow, який реагує на події комітів у репозиторії. Коли новий коміт надходить у основну гілку, workflow автоматично запускається і виконує наступні етапи:
   * **Білд проєкту**: виконується збірка проєкту для перевірки на наявність помилок.
   * **Запуск тестів**: після успішної збірки проводяться автоматизовані тести для перевірки функціональності.
3. **Налаштування файла .github/workflows/ci.yml**: Цей файл містить налаштування для GitHub Actions і вказує, які дії слід виконати під час комітів.

Конфігурація:

name**:** **.**NET

on**:**

push**:**

branches**:** **[** "master" **]**

pull\_request**:**

branches**:** **[** "master" **]**

jobs**:**

build**:**

runs**-**on**:** windows**-**latest

steps**:**

**-** uses**:** actions**/**checkout@v2

**-** name**:** Setup MSBuild

uses**:** microsoft**/**setup**-**msbuild@v1

**-** name**:** Setup NuGet

uses**:** NuGet**/**setup**-**nuget@v2

**-** name**:** Restore NuGet packages

run**:** nuget restore "Student evaluation.sln"

**-** name**:** Build the Solution

run**:** msbuild "Student evaluation.sln"

**-** name**:** Test

run**:** dotnet test "Student evaluation.Test" **--**no**-**build **--**verbosity normal

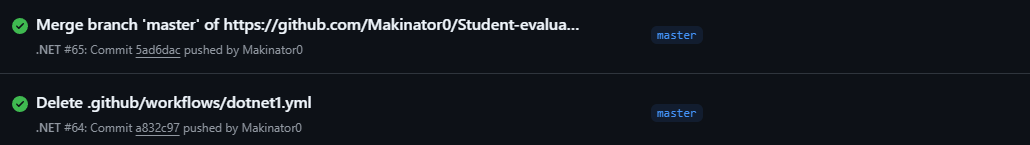


Рисунок 1.20 – Результат пайплайну

**9.2 Післярелізна підтримка з допомогою Юніт тестів**

Було створено клас StudentDataTest, який перевіряє правильність створення випадкових студентів та їх додавання до загального списку.

**Ключові моменти класу StudentDataTest:**

Генерація випадкових даних для студента

Клас використовує списки імен, прізвищ та інших параметрів, розділені за статтю, для створення випадкового студента:

**private** static string**[]** maleSurnames **=** **{** "Алексєєв"**,** "Бондаренко"**,** "Бочаров" **};**

**private** static string**[]** femaleSurnames **=** **{** "Бєдарєва"**,** "Гасюк"**,** "Григоренко" **};**

**private** static string**[]** maleNames **=** **{** "Олександр"**,** "Дмитро"**,** "Максим" **};**

**private** static string**[]** femaleNames **=** **{** "Галина"**,** "Єлизавета" **};**

bool isMale **=** random**.**Next**(**2**)** **==** 0**;**

string name **=** isMale **?** maleNames**[**random**.**Next**(**maleNames**.**Length**)]** **:** femaleNames**[**random**.**Next**(**femaleNames**.**Length**)];**

string surname **=** isMale **?** maleSurnames**[**random**.**Next**(**maleSurnames**.**Length**)]** **:** femaleSurnames**[**random**.**Next**(**femaleSurnames**.**Length**)];**

Це дозволяє врахувати статеву приналежність для створення реалістичних даних.

**Модульний тест TestCreateRandomStudent**

Тест перевіряє, чи створений студент додається до бази даних, а також чи він існує після збереження:

**[**TestMethod**]**

**public** void TestCreateRandomStudent**()**

**{**

string **group** **=** groups**[**random**.**Next**(**groups**.**Length**)];**

var student **=** **new** Student

**{**

Name **=** name**,**

Surname **=** surname**,**

LastName **=** patronymic**,**

Group **=** **group,**

Semesters **=** **new** List**<**Semester**>**

**{**

**new** Semester **{** SemesterNumber **=** "1"**,** Grades **=** GenerateRandomGrades**()** **},**

**new** Semester **{** SemesterNumber **=** "2"**,** Grades **=** GenerateRandomGrades**()** **}**

**}**

**};**

SaveData**.**AddStudent**(**student**);**

var students **=** SaveData**.**LoadStudents**();**

bool studentExists **=** students**.**Any**(**s **=>** s**.**Name **==** student**.**Name **&&** s**.**Surname **==** student**.**Surname **&&**

s**.**LastName **==** student**.**LastName **&&** s**.**Group **==** student**.**Group**);**

Assert**.**IsTrue**(**studentExists**,** "Student was not added to the list."**);**

**}**

**Генерація оцінок**

Оцінки створюються для кожного предмета за допомогою наступного методу:

**private** static Dictionary**<**string**,** SubjectGrades**>** GenerateRandomGrades**()**

**{**

var grades **=** **new** Dictionary**<**string**,** SubjectGrades**>();**

**foreach** **(**var subject **in** subjects**)**

**{**

double averageScore **=** Math**.**Round**(**random**.**NextDouble**()** **\*** 50 **+** 50**,** 2**);** // Генерація середнього балу

grades**[**subject**]** **=** **new** SubjectGrades

**{**

FirstHW **=** GenerateRandomGradeWithLetter**(),**

FirstTest **=** GenerateRandomGradeWithLetter**(),**

SecondHW **=** GenerateRandomGradeWithLetter**(),**

AverageScore **=** averageScore**,**

LetterGrade **=** GetLetterGrade**(**averageScore**)**

**};**

**}**

**return** grades**;**

**}**

**Розрахунок літерної оцінки**

Оцінка визначається на основі середнього балу:

**private** static string GetLetterGrade**(**double score**)**

**{**

**if** **(**score **>=** 90**)** **return** "A"**;**

**if** **(**score **>=** 80**)** **return** "B"**;**

**if** **(**score **>=** 70**)** **return** "C"**;**

**if** **(**score **>=** 60**)** **return** "D"**;**

**return** "F"**;**

**}**

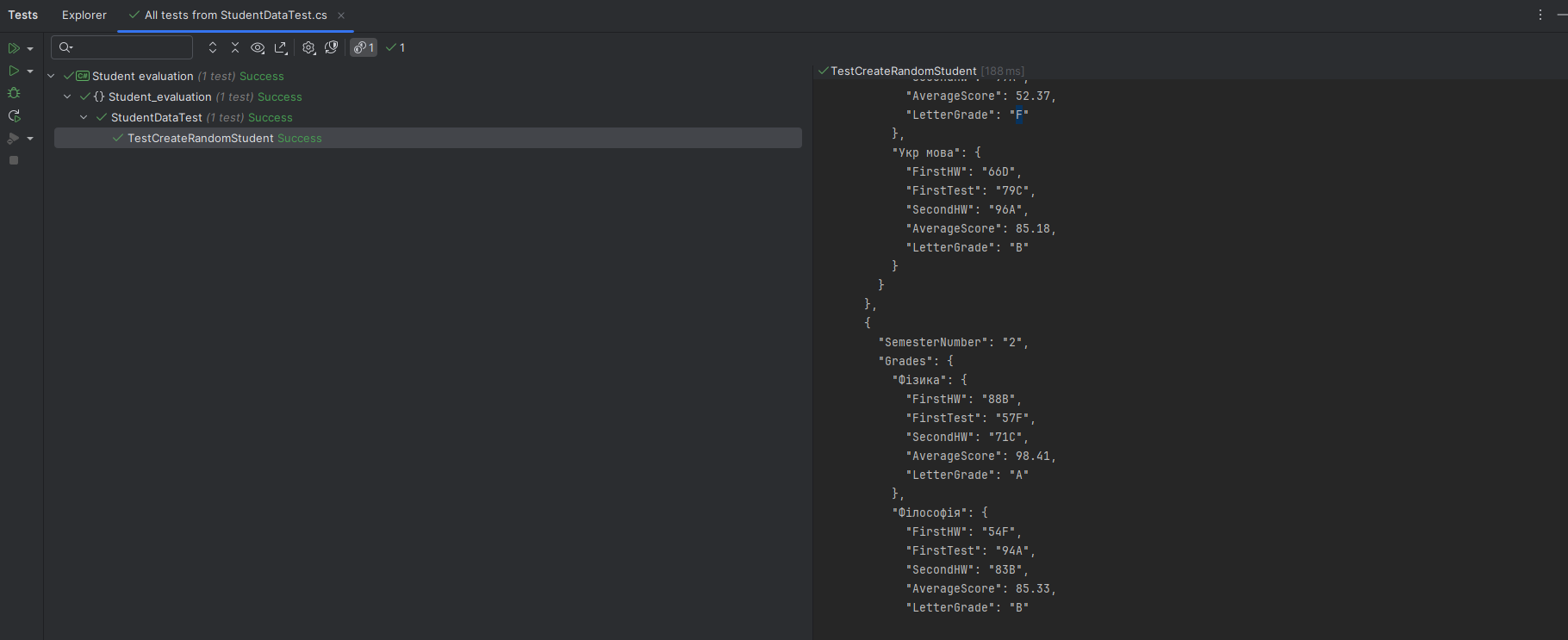


Рисунок 1.21 – Результат тесту

**Висновки**

У процесі виконання курсового проєкту була розроблена інформаційна система для автоматизації роботи з даними студентів, яка надає можливості збереження, обробки та аналізу навчальної успішності. Реалізація проєкту дозволила набути значного практичного досвіду у використанні сучасних технологій та підходів до розробки програмного забезпечення.

Отримані практичні навички

1. Створення структурованої бази даних. Система використовує формат JSON для зберігання інформації про студентів, їхні оцінки за предметами, а також успішність по семестрах. Це дозволяє легко інтегрувати базу з іншими додатками або використовувати її для подальшого аналізу.
2. Розробка WinForms-додатку. Для реалізації графічного інтерфейсу був використаний інструментарій WinForms та мова програмування C#. Це забезпечило створення зручного та інтуїтивно зрозумілого інтерфейсу користувача, який дозволяє ефективно взаємодіяти з системою.
3. Функціонал управління даними. У додатку реалізовано функції для роботи з даними студентів, зокрема:
   * додавання нових записів;
   * редагування існуючої інформації;
   * видалення студентів з бази даних;
   * пошук за різними критеріями, такими як прізвище або група.

Це дозволяє легко оновлювати та актуалізувати дані в системі.

1. Генерація звітів у форматі PDF. У додатку розроблено функціонал для створення звітів з оцінками студентів у вигляді PDF-документів. Генерація звітів здійснюється автоматично, що спрощує роботу користувачів із великими обсягами інформації. Звіти можуть містити інформацію про всі семестри або вибраний період.
2. Реалізація модульного тестування. Для перевірки коректності роботи системи було впроваджено модульні тести за допомогою бібліотеки Microsoft.VisualStudio.TestTools.UnitTesting. Це забезпечило стабільність та надійність роботи програми, а також дозволило оперативно виявляти помилки у функціоналі.
3. Автоматизація CI/CD за допомогою GitHub Actions. У проєкті реалізовано автоматизацію процесів збірки, тестування та розгортання системи за допомогою GitHub Actions. Завдяки цьому забезпечується безперервна інтеграція та швидке виявлення помилок на етапах розробки.
4. Наприклад, при кожному коміті проєкт автоматично збирається, запускаються всі тести, і у разі їх успішного проходження результати інтегруються до основної гілки.

Переваги розробленої системи

Розроблена програма є зручною у використанні, забезпечує швидкий доступ до необхідної інформації та автоматизує рутинні завдання, пов’язані з обліком успішності студентів. Інтуїтивно зрозумілий інтерфейс дозволяє мінімізувати час, необхідний для навчання користувачів.

Сучасні технології та підходи, використані у проєкті, дозволяють:

* гарантувати стабільну роботу програми;
* знижувати ризики помилок при обробці даних;
* забезпечувати можливість масштабування системи.

Перспективи подальшого розвитку

Система має потенціал для розширення та інтеграції з іншими інформаційними системами. Можливі напрямки розвитку включають:

* впровадження веб-інтерфейсу для віддаленого доступу до даних;
* додавання функцій глибокого аналізу даних, таких як прогнозування успішності студентів;
* інтеграція з навчальними платформами для автоматичного отримання результатів виконання завдань;
* створення мобільного додатку для зручності використання на смартфонах.

Загальний підсумок

Розроблена система може бути корисною для навчальних закладів, оскільки значно спрощує та пришвидшує облік успішності студентів. Завдяки її гнучкості та функціональності, вона може стати основою для створення більш складних інформаційних систем.