

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Відокремлений структурний підрозділ
«Криворізький фаховий коледж Національного авіаційного університету»
Циклова комісія професійно-орієнтованих дисциплін та програмного
забезпечення

КУРСОВА РОБОТА

з навчальної дисципліни «Об'єктно-орієнтоване програмування»
на тему: Облік відвідування занять

Здобувача освіти 4 курсу 3-007 групи
спеціальності 121 «Інженерія
програмного забезпечення»

Юзбекова В.С

(прізвище та ініціали)

Керівник Андрусевич Н.В

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала

Кількість балів: _____ Оцінка:

ECTS

Члени _____

(підпис)

(прізвище та ініціали)

(підпис)

(прізвище та ініціали)

(підпис)

(прізвище та ініціали)

м. Кривий Ріг – 2023 рік

ЗАВДАННЯ
на виконання курсової роботи
здобувача освіти Юзбекова Віктора Сергійовича
варіант 18

Тема курсової роботи:

ОБЛІК ВІДВІДУВАННЯ ЗАНЯТЬ

Технічні вимоги:

1. Використання мови програмування C#.
2. Використання Entity Framework для взаємодії з базою даних.
3. Десктопний додаток для доступу до функцій програми.
4. Розробка застосунку на мові програмування C# з Entity Framework.
5. Розробити застосунки відповідно до варіанту. Номер варіанта визначається за номером в навчальному журналі.
6. Створити повністю робочий застосунок на WPF з трьома меню та можливістю додавати, редагувати та видаляти дані з бази даних.
7. Дані за кожним студентом: прізвище, ім'я, номер залікової книжки. Навчальні дисципліни: назва, семестр, загальна кількість годин. Викладачі: прізвище, ім'я. Розділення занять на види: номер дисципліни, номер викладача, вид заняття, кількість виділених годин. Відмітки: номер розділення, номер студента, вид пропуску, дата, номер пари.
8. Розробити: меню програми і засобів діалогу, форми введення і редагування даних, запити, форматування даних у JSON формат, зчитування даних з JSON формату.

Термін виконання роботи: з 18.09.2023 р. по 20.12.2023 р. р.

Календарний план (етапи роботи над КР):

Стадії та етапи роботи	Зміст робіт	Термін виконання
Технічне завдання	Постановка задачі, визначення вимог, структури даних, метода рішення та т.п.	18.09.2023-02.10.2023
Технічний проект	Розробка алгоритму, визначення форми представлення даних, структури програми. Оформлення першого розділу пояснювальної записки.	03.10.2023-20.10.2023
Робочий проект	Розробка програми та її випробування. Оформлення пояснювальної записки.	21.10.2023-15.12.2023
Захист курсової роботи		18.12.2023-20.12.2023

Зміст

ВСТУП	5
1 СПЕЦИФІКАЦІЯ ПРОГРАМИ	6
2 ПРОЕКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ.....	7
3 ІНСТРУКЦІЯ КОРИСТУВАЧА.....	17
ВИСНОВКИ.....	29
ПЕРЕЛІК ПОСИЛАНЬ.....	30

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 31 с., 35 рис., 4 джерела.

КЛАС, МОВА ПРОГРАМУВАННЯ C#, ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, ПРОГРАМА, ПЛАТФОРМА .NET, РЕГІСТР, СКЛАД.

У наявності дані про кожного студента, включаючи їх прізвище, ім'я та номер залікової книжки. Також є інформація про навчальні предмети: назву, семестр та загальну кількість годин. Дані про викладачів містять їх прізвища та імена. Заняття розподілені за видами, що включають номер предмету, номер викладача, вид заняття та кількість годин, виділених для нього. Також є інформація про оцінки: номер розділу, номер студента, вид пропуску, дата та номер пари.

Необхідно розробити меню програми та інструменти для взаємодії, створити форми для введення та редагування даних, написати запити для взаємодії з даними, провести форматування даних у JSON-форматі, а також забезпечити зчитування даних з JSON-формату.

ВСТУП

Мова програмування C# - це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона широко використовується для створення різноманітних програм, від десктопних до веб-додатків. C# має схожий синтаксис з іншими мовами програмування з родини C, що робить його зручним для вивчення та використання.

Entity Framework (EF) - це набір технологій від Microsoft для роботи з базами даних в рамках платформи .NET. EF надає можливість програмістам працювати з даними за допомогою об'єктно-орієнтованих концепцій, уникнувши необхідності писати багато SQL-коду. Він дозволяє створювати моделі даних, які потім можна використовувати безпосередньо у програмі, і автоматично генерує SQL-запити для взаємодії з базою даних. Крім того, Entity Framework взаємодіє з LINQ (Language Integrated Query), що дозволяє розробникам писати структуровані запити до даних у зручний, мовою програмування C# спосіб.

1 СПЕЦИФІКАЦІЯ ПРОГРАМИ

Ця програма розробляється з метою полегшення роботи з великим обсягом даних, пов'язаних з навчальними процесами, для забезпечення зручності введення, редагування, та оптимізації взаємодії з цими даними.

Програма має наступні функціональні можливості:

1. Меню користувача повинно забезпечувати:
 - Інтуїтивно зрозумілий інтерфейс зі структурованим меню для зручного вибору операцій та навігацій по програмі.
2. Можливість введення та редагування даних:
 - Форми, що дозволяють додавати, редагувати та видаляти інформацію про студентів, навчальні предмети, викладачів, розклад занять та відмітки.
3. Можливість використання запитів:
 - Можливість створювати різноманітні запити для отримання даних за різними критеріями.
4. Можливість формування звітності:
 - Збереження та зчитування інформації у форматі JSON для простоти обміну даними та їх подальшої обробки.
5. Можливість вибору даних за умовами:
 - Фільтрація та зчитування інформації за різними критеріями.

Мета програми полягає у реалізації ефективного збору обліку відвідування студентами занять та ведення даних про навчальний процес, що дозволить ефективно контролювати та аналізувати динаміку відвідування не тільки викладачам, а ще і студентам.

2 ПРОЕКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ

Маючи опис даних треба спроектувати структуру бази даних, визначаючи основні сутності та зв'язки між ними. Ось приклад глобальної моделі цієї бази даних (рис. 1).

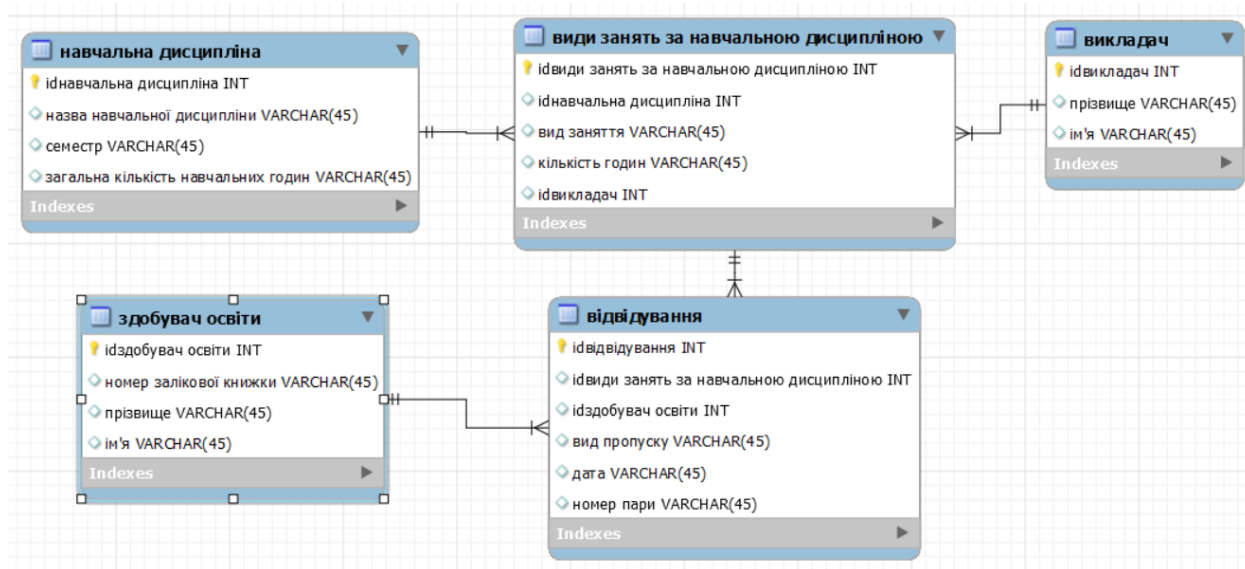


Рисунок 1 – Глобальна модель бази даних

На етапі проектування вирішено, що доцільним або достатнім буде використання у програмі п'яти основних вікон для кожної з таблиць. Реалізовані вони за допомогою елементу TabControl. У цьому контролі були створені п'ять TabItem, присвоївши кожному заголовок, відповідний таблиці у черзі: "Student", "Subject", "Teacher", "TypeSubject", та "Attendance". Для кожної з вкладок потрібно додати колонку, що відображатиме відповідний параметр з таблиці.

Створення вкладки "Student" передбачає реалізацію елементу ListView з ім'ям StudentListView, який відображатиме дані, пов'язані з властивістю "Student". Цей ListView використовує GridView для представлення даних у вигляді таблиці з різними колонками. У цьому GridView чотири GridViewColumn, які відповідають чотирьом полям даних студента: "Id", "Прізвище", "Ім'я" та "Номер залікової книжки". Кожна колонка має встановлені параметри ширини та типу відображення даних, що забезпечують

коректне відображення відповідних значень полів Id, Surname, Name та RBNumber для кожного елемента списку "Student" (рис. 2) .

```
<TabItem Name="StudentPage" Header="Student">
  <ListView Name="StudentListView" ItemsSource="{Binding
Student}">
    <ListView.View>
      <GridView>
        <GridViewColumn Header="Id" Width="Auto"
DisplayMemberBinding="{Binding Id}"/>
        <GridViewColumn Header="Прізвище" Width="Auto"
DisplayMemberBinding="{Binding Surname}"/>
        <GridViewColumn Header="Ім'я" Width="Auto"
DisplayMemberBinding="{Binding Name}"/>
        <GridViewColumn Header="Номер залікової книжки"
Width="Auto" DisplayMemberBinding="{Binding RBNumber}"/>
      </GridView>
    </ListView.View>
  </ListView>
</TabItem>
```

Рисунок 2 – Код сторінки «Студент»

Далі потрібно створити відображення таблиці "Subject" потрібно реалізувати вкладку через GridView де є чотири поля даних дисципліни: "Id", "Назва", "Семестр" та "Загальна кількість годин" (рис. 3) .

```
<TabItem Name="SubjectPage" Header="Subject">
  <ListView Name="SubjectListView" ItemsSource="{Binding
Subject}">
    <ListView.View>
      <GridView>
        <GridViewColumn Header="Id" Width="Auto"
DisplayMemberBinding="{Binding Id}"/>
        <GridViewColumn Header="Назва" Width="Auto"
DisplayMemberBinding="{Binding Name}"/>
        <GridViewColumn Header="Семестр" Width="Auto"
DisplayMemberBinding="{Binding Semester}"/>
        <GridViewColumn Header="Загальна кількість
годин" Width="Auto" DisplayMemberBinding="{Binding
TotalHours}"/>
      </GridView>
    </ListView.View>
  </ListView>
</TabItem>
```

Рисунок 3 – Код сторінки «Дисципліна»

Далі потрібно створити відображення таблиці "Teacher" потрібно реалізувати вкладку через GridView де є три поля даних викладача: "Id", "Прізвище" та "Ім'я" (рис. 4) .

```
<TabItem Name="TeacherPage" Header="Teacher">
    <ListView Name="TeacherListView" ItemsSource="{Binding
Teacher}">
        <ListView.View>
            <GridView>
                <GridViewColumn Header="Id" Width="Auto"
DisplayMemberBinding="{Binding Id}"/>
                <GridViewColumn Header="Прізвище" Width="Auto"
DisplayMemberBinding="{Binding Surname}"/>
                <GridViewColumn Header="Ім'я" Width="Auto"
DisplayMemberBinding="{Binding Name}"/>
            </GridView>
        </ListView.View>
    </ListView>
</TabItem>
```

Рисунок 4 – Код сторінки «Викладач»

Далі потрібно створити відображення таблиці "TypeSubject" потрібно реалізувати вкладку через GridView де є п'ять полів даних типів занять: "Id", "ID предмета", "ID вчителя", "Вид заняття" та "Кількість годин" (рис. 5) .

```
<TabItem Name="TypeSubjectPage" Header="TypeSubject">
    <ListView Name="TypeSubjectListView" ItemsSource="{Binding
TypeSubject}">
        <ListView.View>
            <GridView>
                <GridViewColumn Header="Id" Width="Auto"
DisplayMemberBinding="{Binding Id}"/>
                <GridViewColumn Header="ID предмета"
Width="Auto" DisplayMemberBinding="{Binding SubjectId}"/>
                <GridViewColumn Header="ID вчителя"
Width="Auto" DisplayMemberBinding="{Binding TeacherId}"/>
                <GridViewColumn Header="Вид заняття"
Width="Auto" DisplayMemberBinding="{Binding Type}"/>
                <GridViewColumn Header="Кількість годин"
Width="Auto" DisplayMemberBinding="{Binding Hours}"/>
            </GridView>
        </ListView.View>
    </ListView>
</TabItem>
```

Рисунок 5 – Код сторінки «Тип заняття»

Остання сторінка – «Відмітки». Для створення цієї сторінки потрібно реалізувати такі поля: "Id", "ID виду", "ID студента", "Вид пропуску", "Дата" та "Номер пари" (рис. 6) .

```
<TabItem Name="AttendancePage" Header="Attendance">
    <ListView Name="AttendanceListView" ItemsSource="{Binding Attendance}">
        <ListView.View>
            <GridView>
                <GridViewColumn Header="Id" Width="Auto"
DisplayMemberBinding="{Binding Id}"/>
                <GridViewColumn Header="ID виду" Width="Auto"
DisplayMemberBinding="{Binding TypeSubjectId}"/>
                <GridViewColumn Header="ID студента"
Width="Auto" DisplayMemberBinding="{Binding StudentId}"/>
                <GridViewColumn Header="Вид пропуску"
Width="Auto" DisplayMemberBinding="{Binding Skip}"/>
                <GridViewColumn Header="Дата" Width="Auto"
DisplayMemberBinding="{Binding Date}"/>
                <GridViewColumn Header="Номер пари"
Width="Auto" DisplayMemberBinding="{Binding NumberLesson}"/>
            </GridView>
        </ListView.View>
    </ListView>
</TabItem>
```

Рисунок 6 – Код сторінки «Відмітки»

Інформацію до таблиць можна додати за допомогою кнопки, яка розташована у нижній частині екрану і має відповідний код (рис. 7) .

```
private void btnAdd_Click(object sender, RoutedEventArgs e)
{
    if (StudentPage.IsSelected)
    {
        AddWindow windowAdd = new AddWindow("StudentAdd");
        windowAdd.ShowDialog();
    }
    else if (SubjectPage.IsSelected)
    {
        AddWindow windowAdd = new AddWindow("SubjectAdd");
        windowAdd.ShowDialog();
    }
    else if (TeacherPage.IsSelected)
    {
        AddWindow windowAdd = new AddWindow("TeacherAdd");
        windowAdd.ShowDialog();
    }
    else if (TypeSubjectPage.IsSelected)
    {

```

```

        AddWindow windowAdd = new AddWindow("TypeSubjectAdd");
        windowAdd.ShowDialog();
    }
    else if (AttendancePage.IsSelected)
    {
        AddWindow windowAdd = new AddWindow("AttendanceAdd");
        windowAdd.ShowDialog();
    }
    Window_Loaded();
}

```

Рисунок 7 – Код кнопки додавання даних

Програма має клас NintendoContext (рис. 8), що успадковує DbContext, який є основний класом для зв'язку з базою даних. У ньому визначені <DbSet> для кожного з класів-сутностей, вказавши, які саме елементи бази даних треба ініціалізувати.

```

public class NintendoContext : DbContext
{
    public NintendoContext() : base("Nintendo")
    {
        Database.SetInitializer(new
        DropCreateDatabaseIfModelChanges<NintendoContext>());
    }

    public DbSet<Attendance> Attendance { get; set; }
    public DbSet<Student> Student { get; set; }
    public DbSet<Teacher> Teacher { get; set; }
    public DbSet<TypeSubject> TypeSubject { get; set; }
    public DbSet<Subject> Subject { get; set; }
}

```

Рисунок 8 – Код класу NintendoContext

Кожен з наступних класів відображає якусь певну сутність системи: інформацію про студентів (рис. 9), дисципліни (рис. 10), викладачів (рис. 11), типи занять (рис. 12) та відвідування (рис. 13).

```

public class Student
{
    public int Id { get; set; }
    public string Surname { get; set; }
    public string Name { get; set; }
    public int RBNumber { get; set; } // номер залікової книжки

    public virtual ICollection<Attendance> Attendance { get;
set; }
}

```

Рисунок 9 – Код класу Student

```

public class Subject
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Semester { get; set; }
    public decimal TotalHours { get; set; }

    public virtual ICollection<TypeSubject> TypeSubject { get;
set; }
}

```

Рисунок 10 – Код класу Subject

```

public class Teacher
{
    public int Id { get; set; }
    public string Surname { get; set; }
    public string Name { get; set; }

    public virtual ICollection<TypeSubject> TypeSubject { get;
set; }
}

```

Рисунок 11 – Код класу Teacher

```

public class TypeSubject
{
    public int Id { get; set; }
    public int SubjectId { get; set; }
    public virtual Subject Subject { get; set; }
    public int TeacherId { get; set; }
    public virtual Teacher Teacher { get; set; }
    public EnumType Type { get; set; }
    public decimal Hours { get; set; }

    public virtual ICollection<Attendance> Attendance { get;
set; }
}

```

Рисунок 12 – Код класу TypeSubject

```

public class Attendance
{
    public int Id { get; set; }
    public int TypeSubjectId { get; set; }
    public virtual TypeSubject TypeSubject { get; set; }
    public int StudentId { get; set; }
    public virtual Student Student { get; set; }
    public EnumSkip Skip { get; set; }
    public DateTime Date { get; set; }
    public int NumberLesson { get; set; }
}

```

Рисунок 13 – Код класу Attendance

Програма має можливість зберігати дані про відвідування у JSON форматі, а саме – «Назва дисципліни», «ФІО викладача», «ФІО студента» та «Відмітку» (рис. 14).

```

private void JSONButton_Click(object sender, RoutedEventArgs e)
{
    using (NintendoContext db = new NintendoContext())
    {
        var query = from attendance in db.Attendance
                     join typeSubject in db.TypeSubject on
attendance.TypeSubjectId equals typeSubject.Id
                     join student in db.Student on
attendance.StudentId equals student.Id
                     join subject in db.Subject on
typeSubject.SubjectId equals subject.Id
                     select new AttendanceItem
                     {
                         Id = attendance.Id,
                         StudentName = student.Surname + " " +
student.Name,

```

```

        SubjectName = subject.Name,
        Skip11 = attendance.Skip.ToString()
    };

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

    var viewModel = new AttendanceViewModel();

    viewModel.Attendance.Clear();

    foreach (var item in query)
    {
        viewModel.Attendance.Add(item);
    }

    string filename = System.IO.Path.Combine(folderPath,
    $"day-{fileCount}.json");
    bool fileExists = File.Exists(filename);
    while (fileExists)
    {
        fileCount++;
        filename = System.IO.Path.Combine(folderPath,
    $"day-{fileCount}.json");
        fileExists = File.Exists(filename);
    }

    string json =
    JsonConvert.SerializeObject(viewModel.Attendance,
    Formatting.Indented);
    File.WriteAllText(filename, json);
    MessageBox.Show("Файл збережено");
    fileCount++;
}
}

```

Рисунок 14 – Код конвертації даних у JSON формат

Також, у програмі мається можливість витягувати дані з JSON формату для зчитування їх у ListView (рис. 15).

```

private void SaveButton_Click(object sender, RoutedEventArgs e)
{
    AttendanceListView.ItemsSource = null;

    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Filter = "JSON files (*.json)|*.json|All
files (*.*)|*.*";
    openFileDialog.InitialDirectory = @"C:\";

    bool? result = openFileDialog.ShowDialog();

    if (result == true)
    {
        string selectedFilePath = openFileDialog.FileName;

        if
(System.IO.Path.GetExtension(selectedFilePath).Equals(".json",
System.StringComparison.OrdinalIgnoreCase))
        {
            string jsonContent =
File.ReadAllText(selectedFilePath);
            Attendance =
JsonConvert.DeserializeObject<List<AttendanceItem>>(jsonContent
);
            AttendanceListView.ItemsSource = Attendance;
        }
        else
        {
            MessageBox.Show("Выбранный файл не является файлом
формата JSON.");
        }
    }
}

```

Рисунок 15 – Код зчитування даних з JSON формату

Ще у програмі є можливість вибору даних про відвідування студентами занять окремих викладачів (рис. 16).

```

private void button_Click(object sender, RoutedEventArgs e)
{
    using (NintendoContext db = new NintendoContext())
    {
        var teacherName =
TeacherComboBox.SelectedItem.ToString();
        var subjectName =
SubjectComboBox.SelectedItem.ToString();

        var teacher = db.Teacher.FirstOrDefault(t => t.Name
== teacherName);
    }
}

```



```
        var subject = db.Subject.FirstOrDefault(s => s.Name
== subjectName);

        var typeSubject = db.TypeSubject.FirstOrDefault(ts
=> ts.TeacherId == teacher.Id && ts.SubjectId == subject.Id);

        if (typeSubject != null)
        {
var processWindow = new ProcessWindow(null, subject, teacher,
null, null);

            processWindow.Show();
        }
        else
        {
            MessageBox.Show("Не суй сюди руки!");
            return;
        }

        this.Close();
    }
}
```

Рисунок 16 – Код вибірки даних про студентів з окремих дисциплін

3 ІНСТРУКЦІЯ КОРИСТУВАЧА

При першому запуску програми нас зустрічає головне меню з кнопками «Звичайна» та «Зчитування даних», де треба обрати який варіант треба запустити (рис. 16).

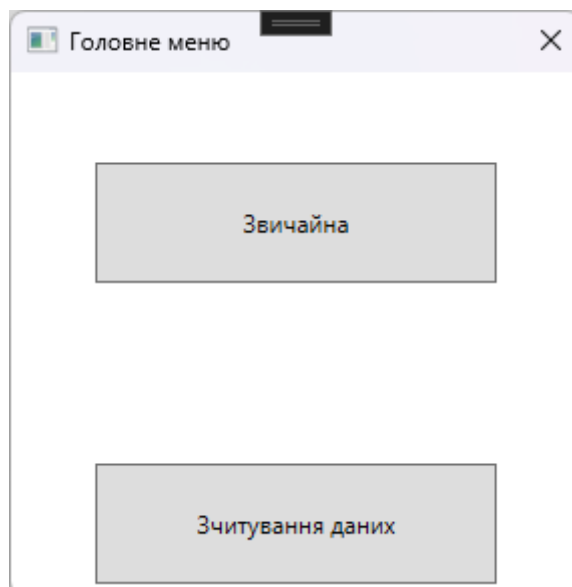


Рисунок 16 – Головне меню програми

Натискаємо кнопку «Звичайна», після чого відкривається вікно під назвою «Звичайна», де розташовані вкладки таблиць та п'ять кнопок (додати, видалити, редагувати, вивести та закрити) (рис. 17).

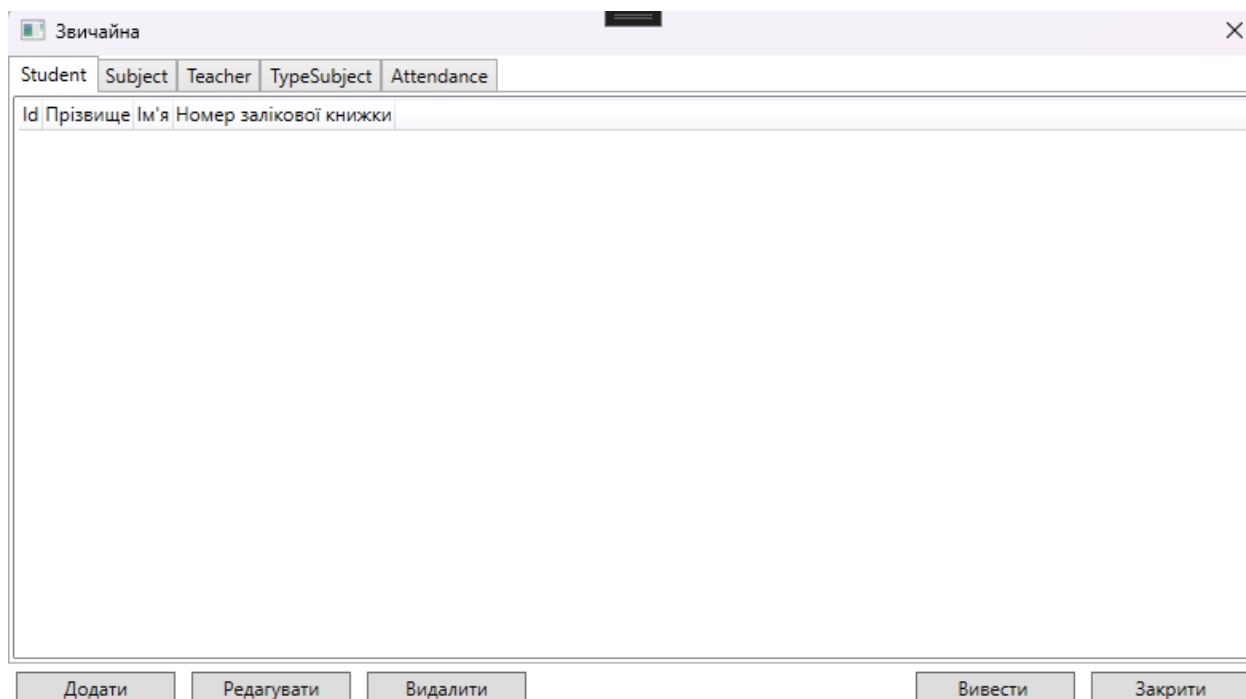
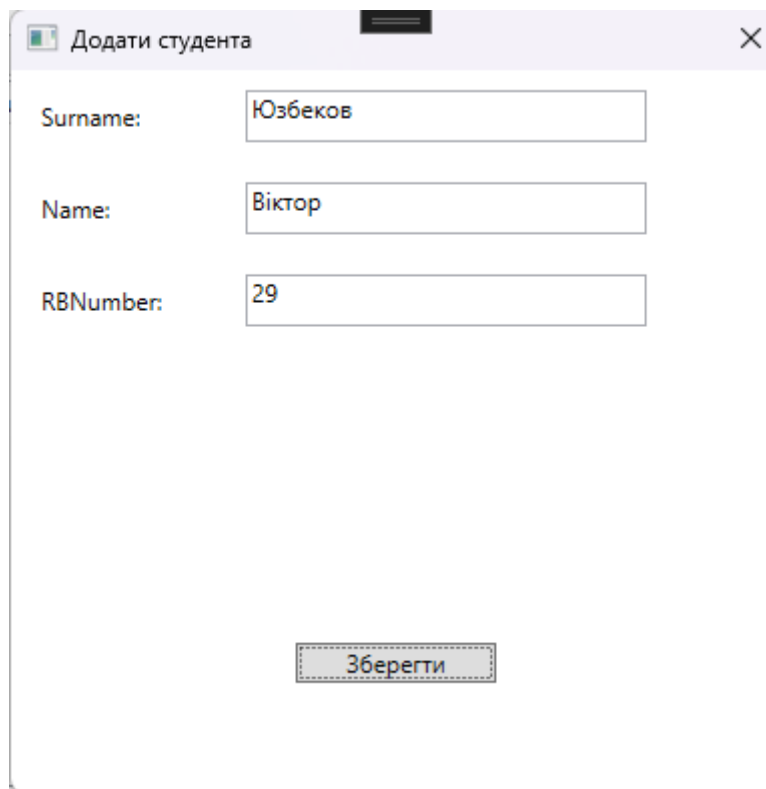


Рисунок 17 – Вікно для відображення таблиць

Натискаємо кнопку додати, після чого відкривається інше вікно під назвою «Додати студента». В ньому нас зустрічає три іменованих текстових поля. Вводимо в них значення та натискаємо кнопку «Зберегти» (рис.18).



Додати студента

Surname: Юзбеков

Name: Віктор

RBNumber: 29

Зберегти

Рисунок 18 – Вікно додавання студентів

Заповнюємо таблицю потрібною інформацією та перевіряємо її правильність. Якщо були додані неправильні дані, або вони мають помилки (рис. 19).

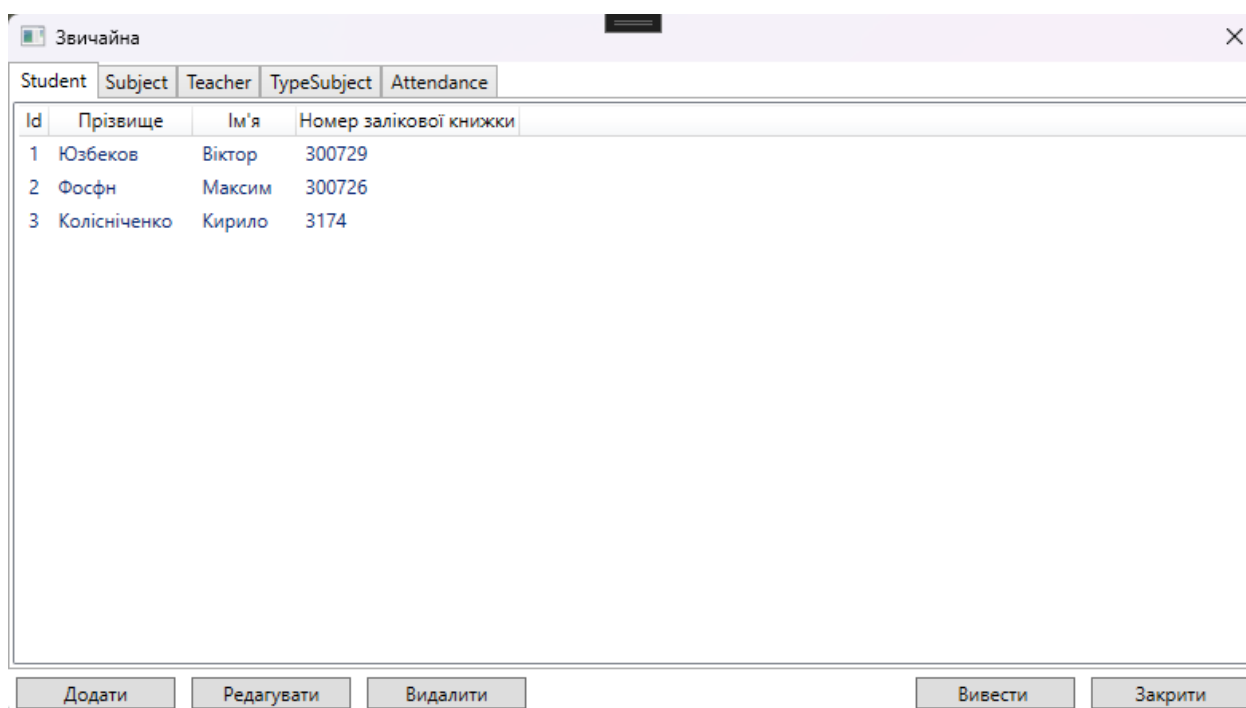


Рисунок 19 – Помилка у написанні прізвища студента

Для виправлення таких ситуацій, була створена кнопка «Редагувати», при натисканні якої з'являється вікно «Редагувати студента» з можливістю редагування даних які були внесені раніше (рис. 20).

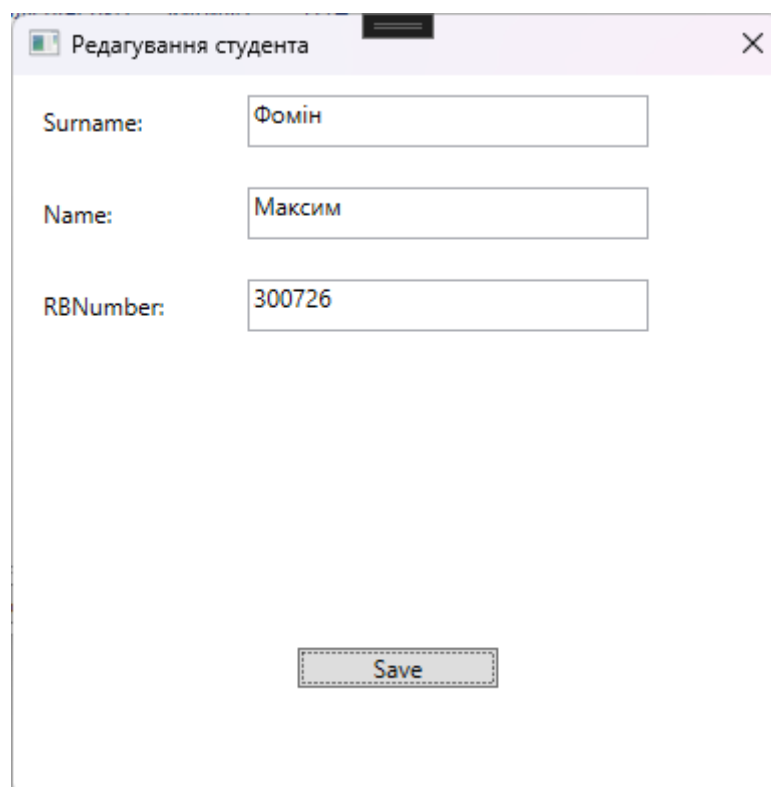


Рисунок 20 – Вікно редагування даних про студентів

Коли введені усі необхідні дані про студентів, можна переходити до наступної вкладки – «Дисципліни», та заповнити її даними (рис. 21).

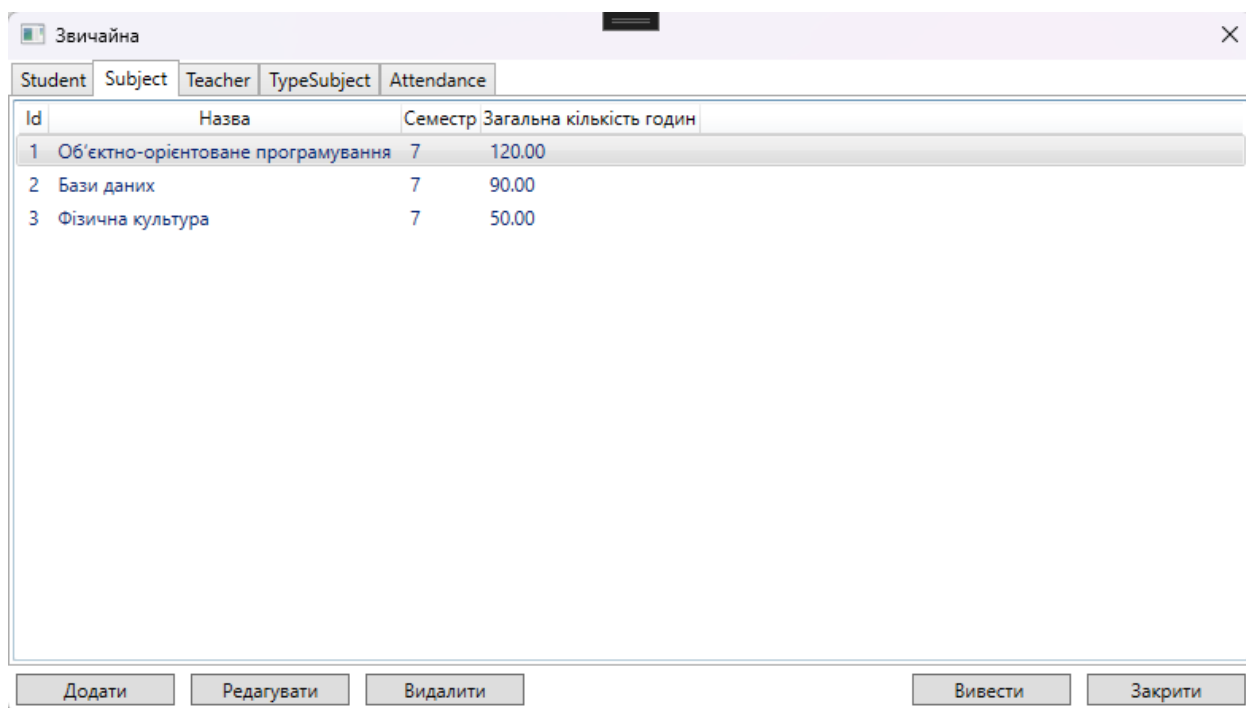


Рисунок 21 – Заповнена таблиця дисциплін

Якщо дані по будь-яким причинам стали не потрібні або втратили актуальність, можна скористатись кнопкою «Видалити» щоб прибрати ці дані з таблиці (рис. 22).

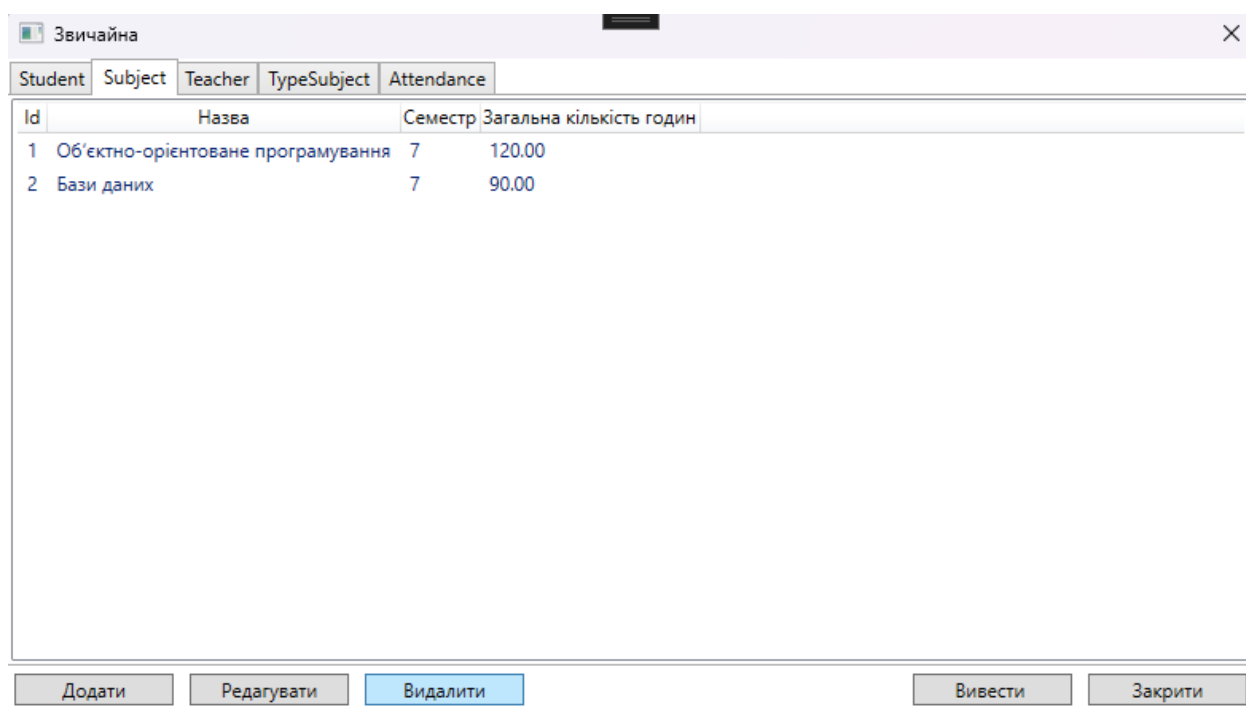


Рисунок 22 – Приклад видалення даних з таблиць

Коли введені усі необхідні дані про дисципліни, можна переходити до наступної вкладки – «Викладачі», та заповнити її даними (рис. 23).

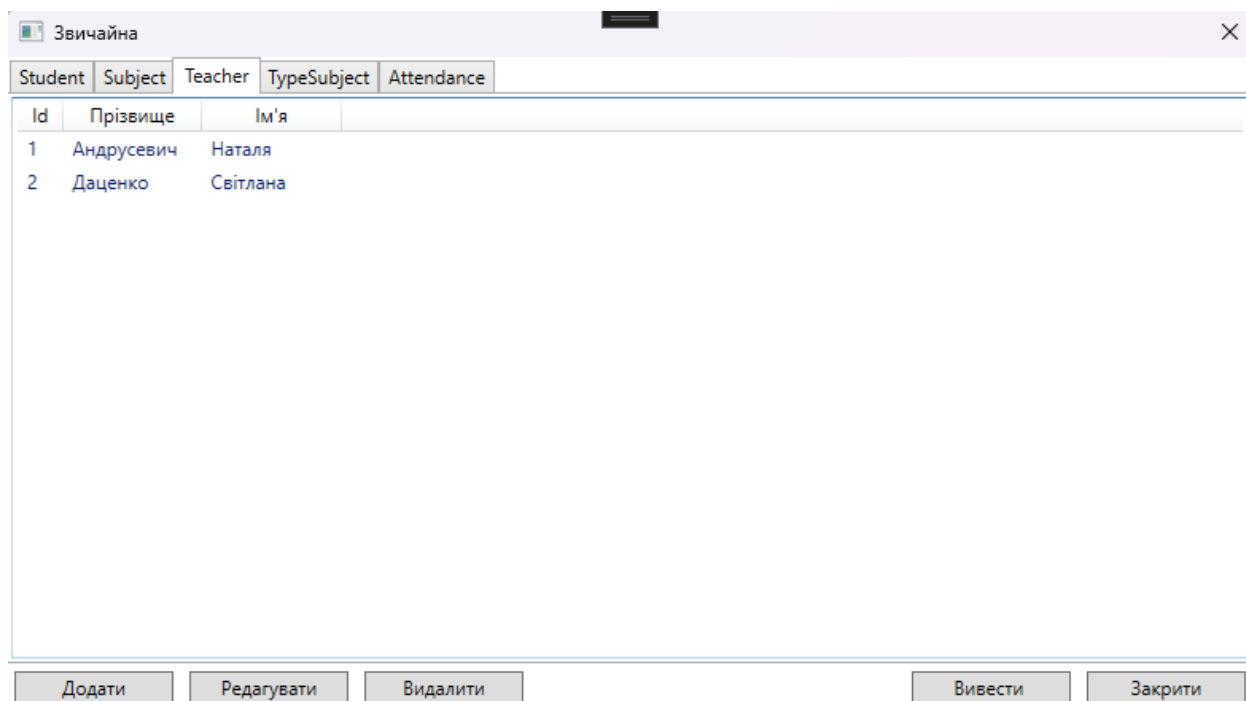
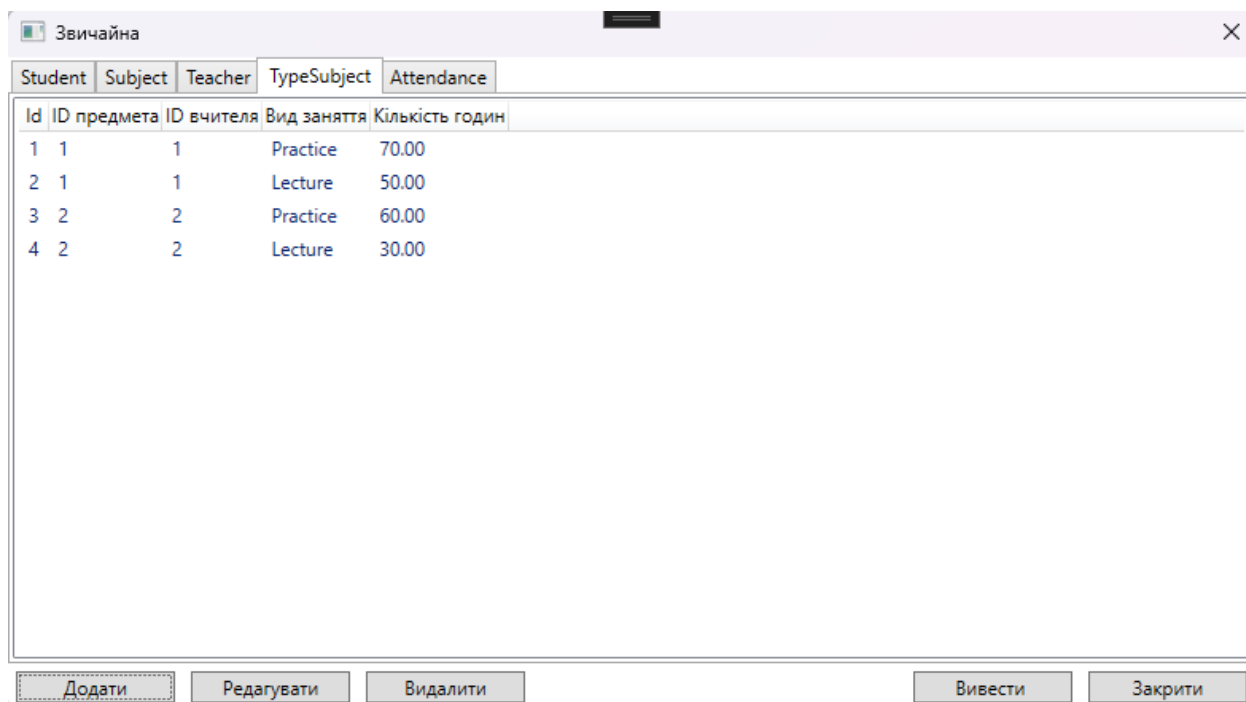


Рисунок 23 – Заповнена таблиця викладачів

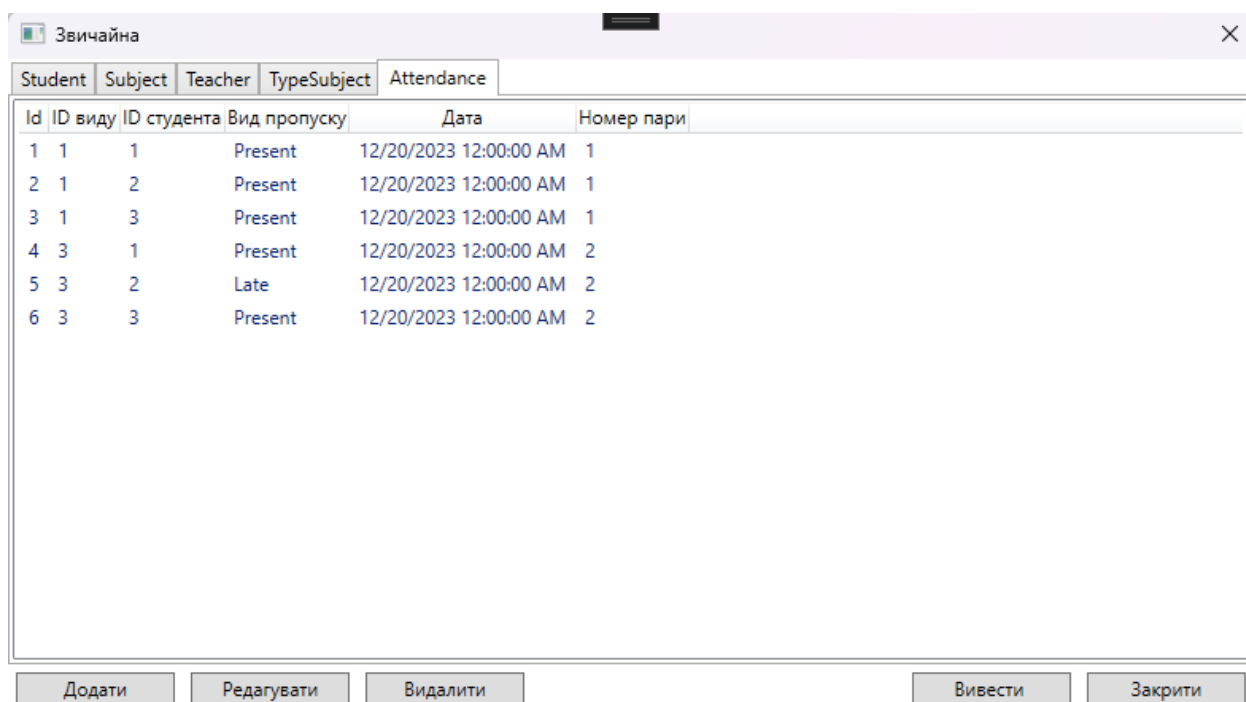
Коли введені усі необхідні дані про викладачів, можна переходити до наступної вкладки – «Типи занять», та заповнити її даними (рис. 24).



Id	ID предмета	ID вчителя	Вид заняття	Кількість годин
1	1	1	Practice	70.00
2	1	1	Lecture	50.00
3	2	2	Practice	60.00
4	2	2	Lecture	30.00

Рисунок 24 – Заповнена таблиця типи занять

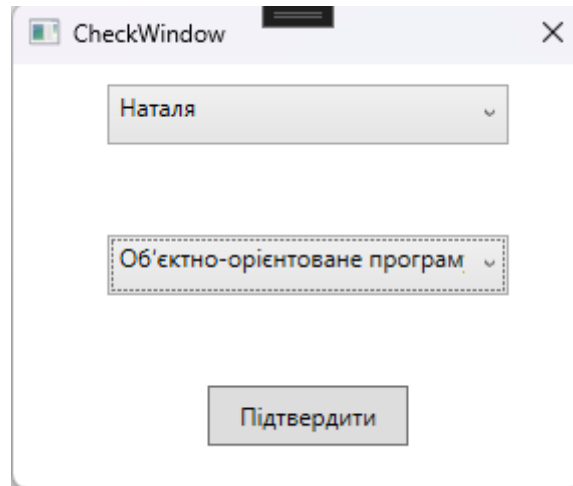
Те Коли введені усі необхідні дані про типи занять, можна переходити до наступної вкладки – «Відмітки», та заповнити її даними (рис. 25).



Id	ID виду	ID студента	Вид пропуску	Дата	Номер пари
1	1	1	Present	12/20/2023 12:00:00 AM	1
2	1	2	Present	12/20/2023 12:00:00 AM	1
3	1	3	Present	12/20/2023 12:00:00 AM	1
4	3	1	Present	12/20/2023 12:00:00 AM	2
5	3	2	Late	12/20/2023 12:00:00 AM	2
6	3	3	Present	12/20/2023 12:00:00 AM	2

Рисунок 25 – Заповнена таблиця відвідування

Тепер можна натиснути на кнопку «Вивести», після чого відкриється вікно, де треба вказати викладача та дисципліну, за якою ми хочемо вивести інформацію (рис. 26).



CheckWindow

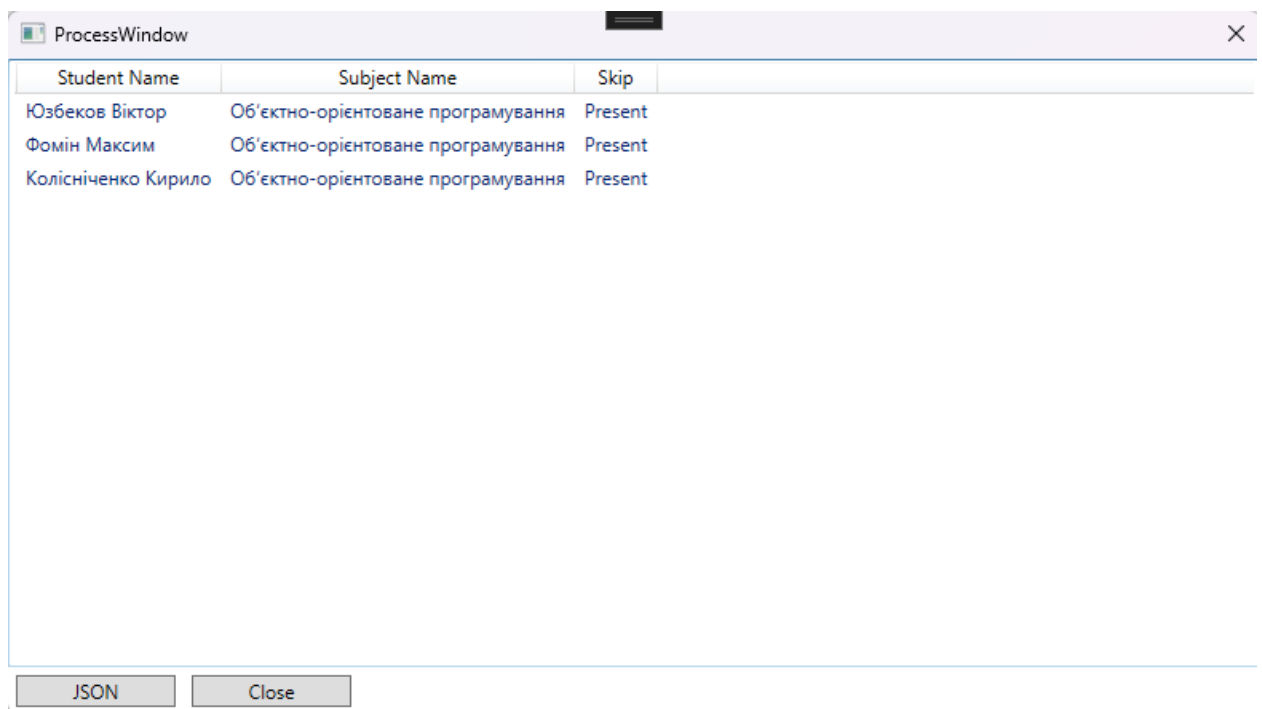
Наталя

Об'єктно-орієнтоване програм

Підтвердити

Рисунок 26 – Вікно вибору викладача та дисципліни

Після чого в нас відкриється вікно, де показані усі студенти, які були присутні на занятті (рис. 27).



ProcessWindow

Student Name	Subject Name	Skip
Юзбеков Віктор	Об'єктно-орієнтоване програмування	Present
Фомін Максим	Об'єктно-орієнтоване програмування	Present
Колісніченко Кирило	Об'єктно-орієнтоване програмування	Present

JSON Close

Рисунок 27 – Відображення даних про відвідування заняття

Якщо по якійсь причині студент мусив покинути заняття, в нас є можливість за допомогою подвійного натискання на студента – відредагувати його статус на занятті (рис. 28).

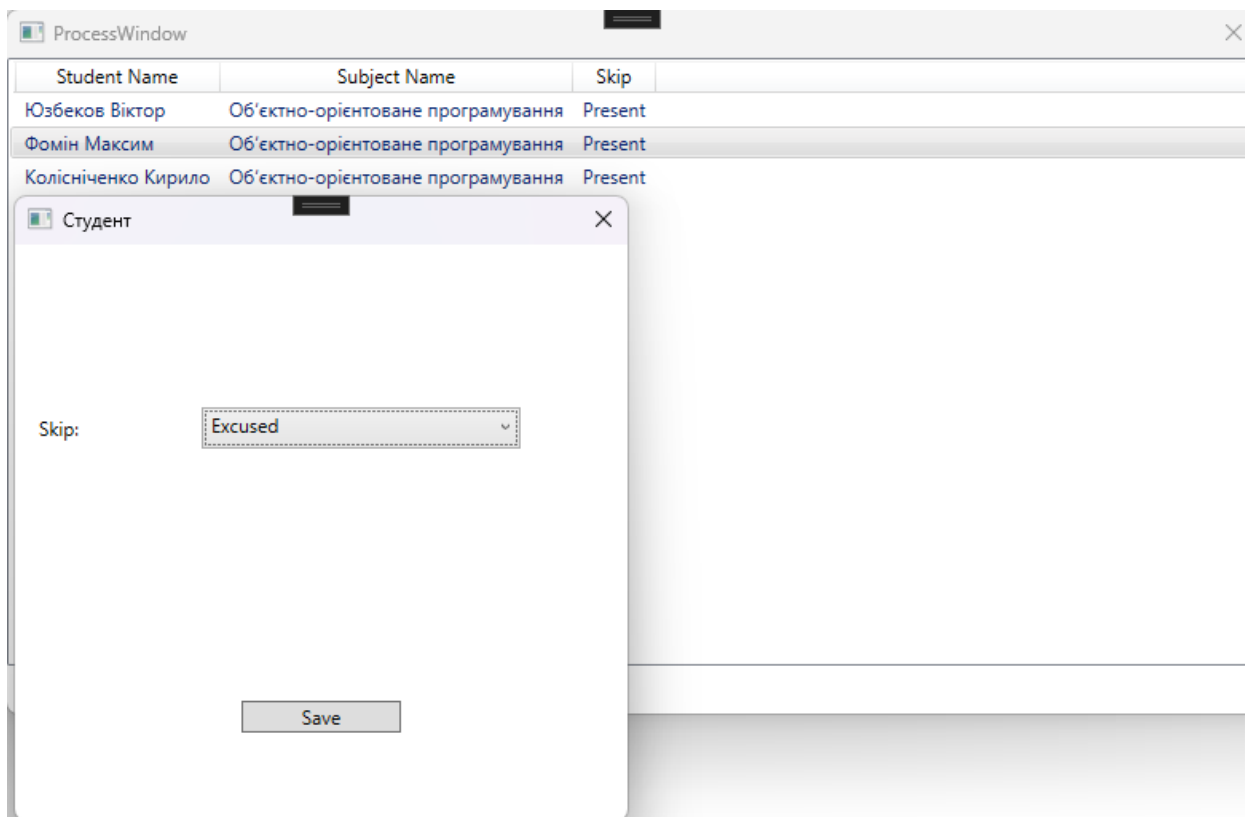


Рисунок 28 – Приклад редагування статусу

Для збереження записів студентів на занятті, можна по натисканню на кнопку «JSON» створити файл, який буде зберігати в собі дані за цю пару (рис. 29 – 30).

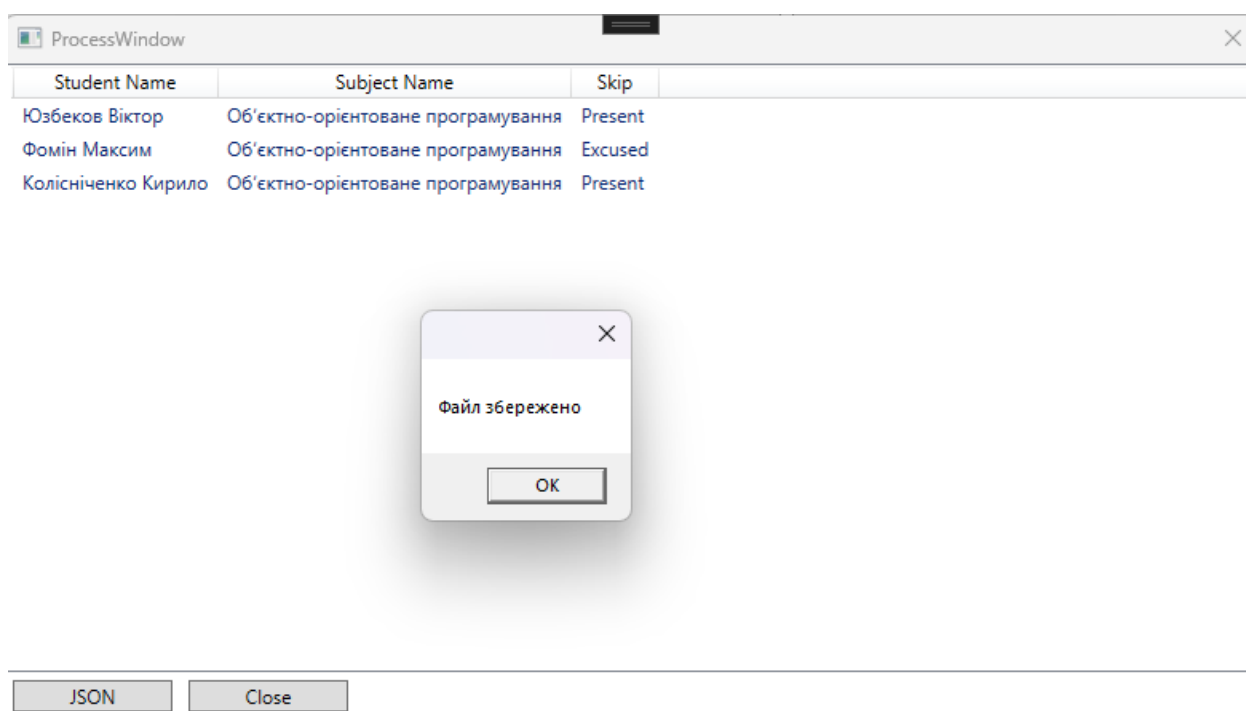


Рисунок 29 – Приклад натискання на кнопку

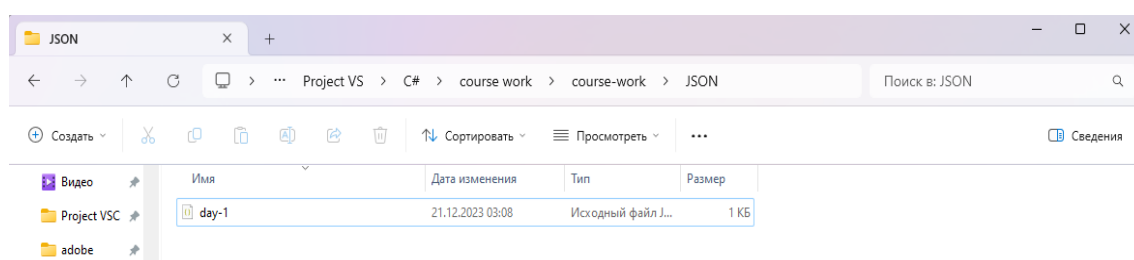


Рисунок 30 – Перевірка наявності файлу

Для перевірки даних цього файлу, натискаємо кнопку «Close» у цьому вікні, та кнопку «Закрити» у наступному. Після чого, ми опиняємося у головному меню, де нам треба натиснути на кнопку «Зчитування даних – відкривається вікно з таблицею та двома кнопками (рис. 31).

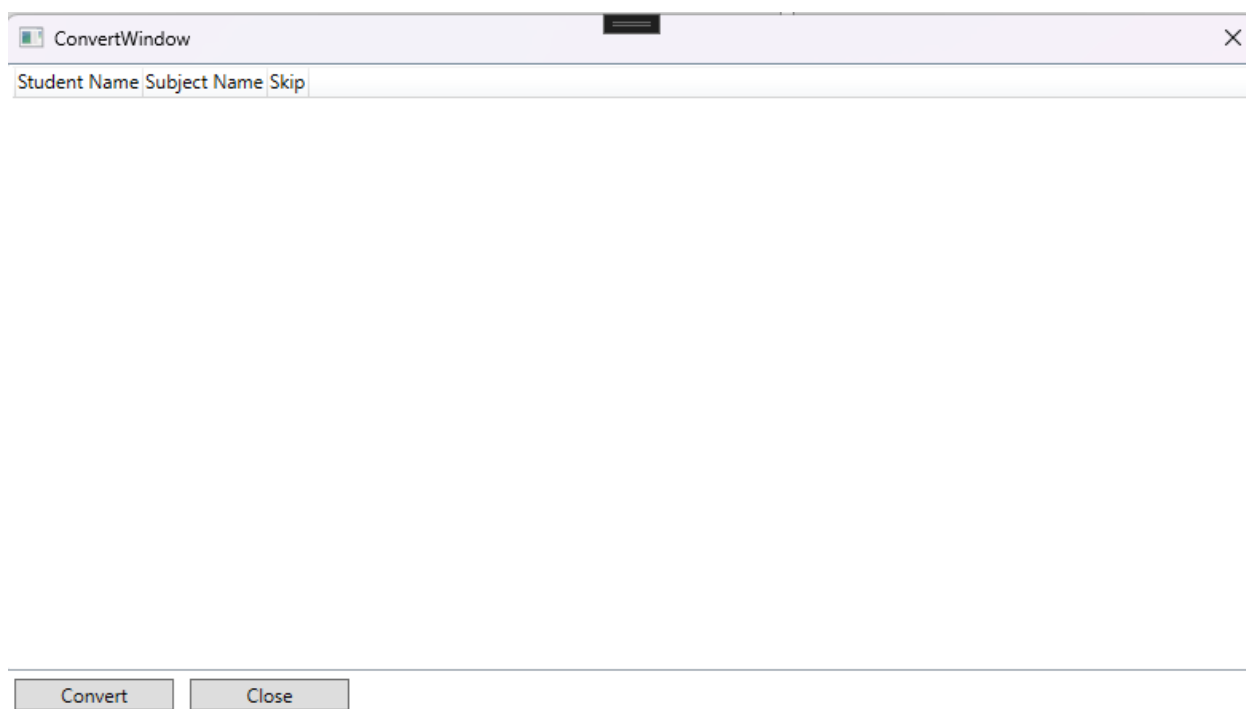


Рисунок 31 – Вікно для зчитування даних

У цьому вікні нам треба натиснути кнопку «Convert», після чого відкриється діалогове вікно для вибору файлу, нам потрібно перейти у папку де зберігся наш файл, і обрати його (рис. 32).

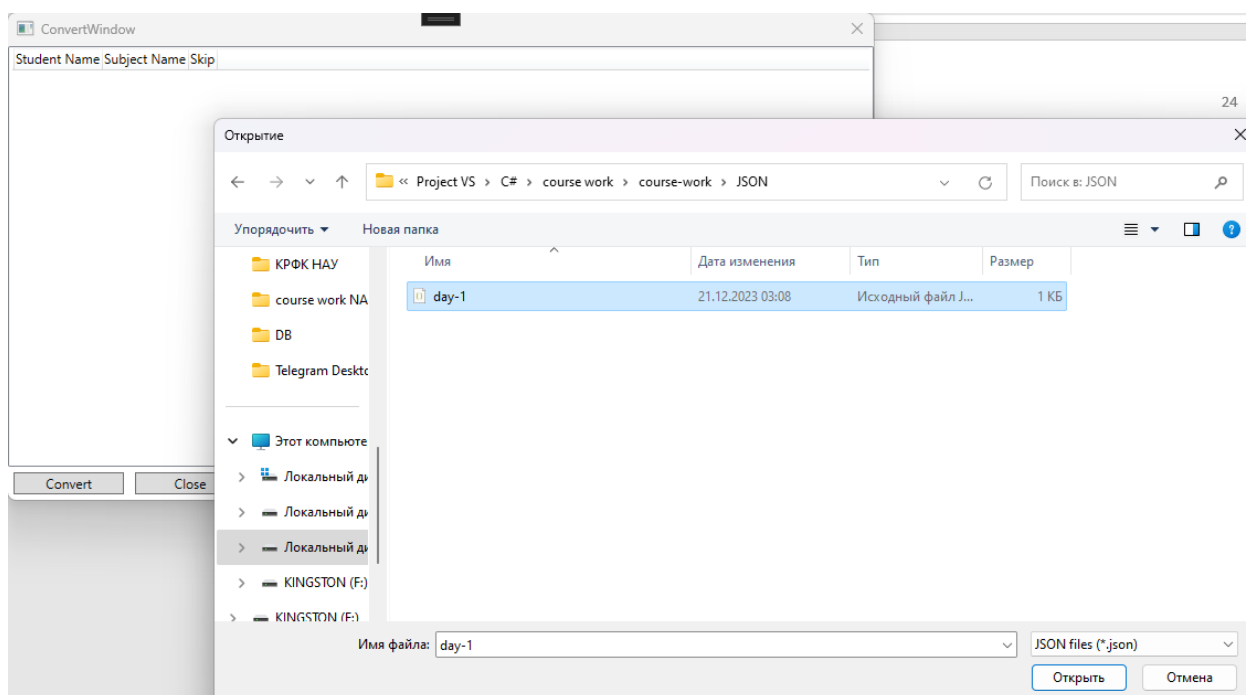


Рисунок 32 – Обирання раніше збереженого файлу

Після того, як ми обрали цей файл, в нас підвантажуються усі дані з файлу (рис. 33).

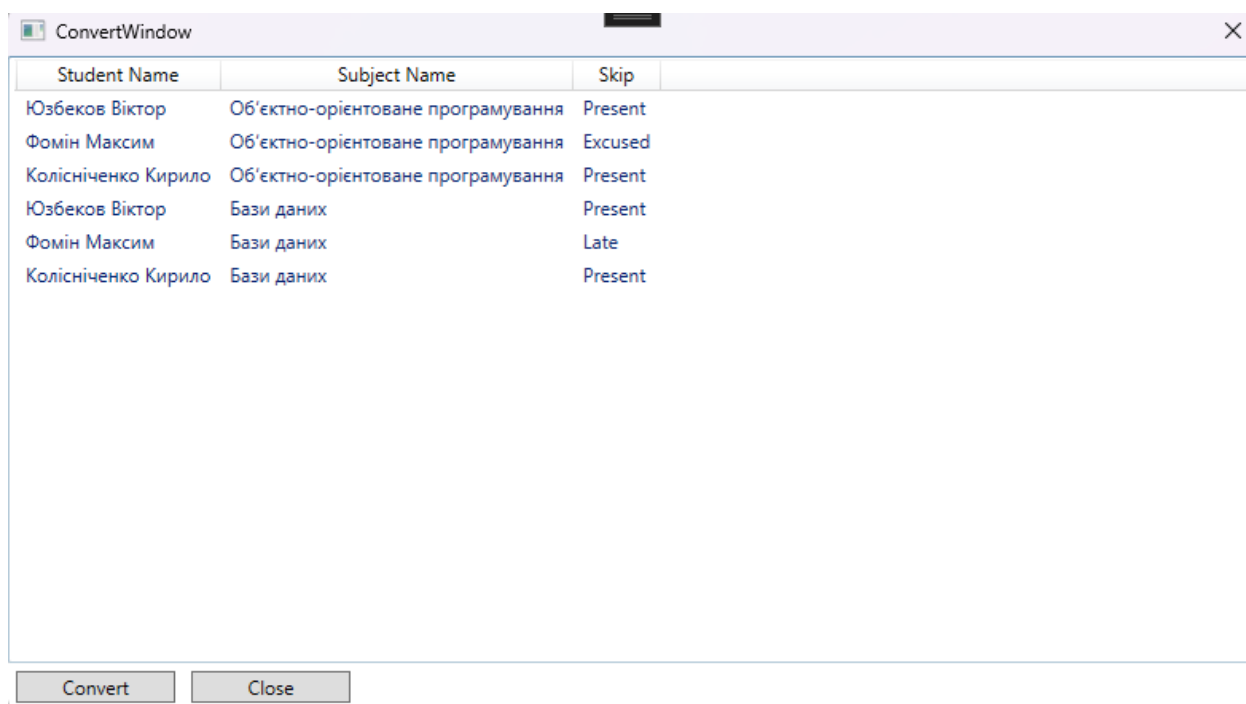


Рисунок 33 – Зчитування даних з файлу

У головному меню програми ми можемо натиснути кнопку для вибору даних про студентів за окремою дисципліною (рис34 – 35).

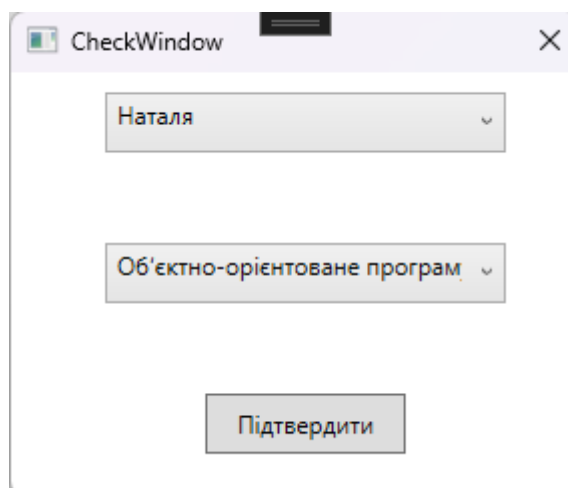
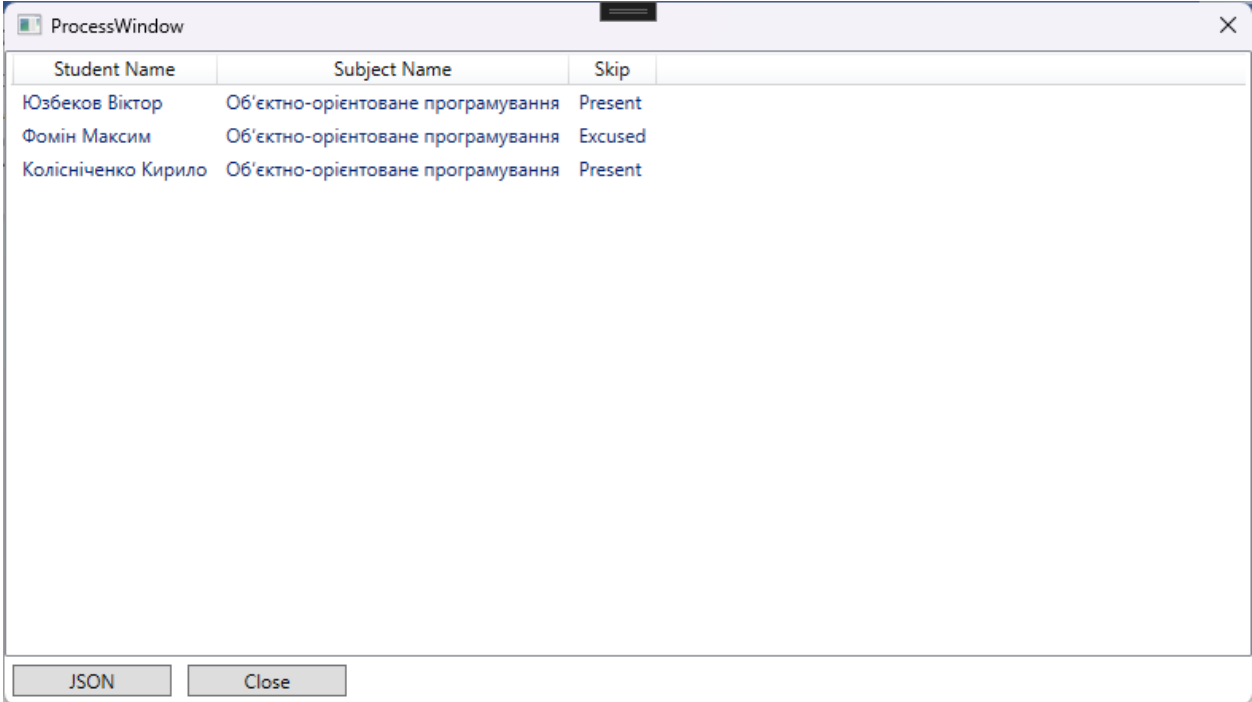


Рисунок 34 – Вибір даних для пошуку



The screenshot shows a window titled 'ProcessWindow' with a close button (X) in the top right corner. Inside the window is a table with three columns: 'Student Name', 'Subject Name', and 'Skip'. The table contains three rows of data. Below the table, there are two buttons: 'JSON' and 'Close'.

Student Name	Subject Name	Skip
Юзбеков Віктор	Об'єктно-орієнтоване програмування	Present
Фомін Максим	Об'єктно-орієнтоване програмування	Excused
Колісниченко Кирило	Об'єктно-орієнтоване програмування	Present

Рисунок 35 – Відображення даних по окремій дисципліні

ВИСНОВКИ

У ході розробки проекту на мові програмування C# з використанням Entity Framework було досягнуто декілька заданих цілей та здійснено успішну реалізацію функціоналу програми.

Основні операції програми, створення, читання, оновлення, видалення, здійснюються ефективно та інтуїтивно. Реалізовано можливість зберігання даних, що підвищує комунікабельність даних між користувачами.

Entity Framework виявився дуже сильним інструментом для взаємодії з базами даних, спрощуючи роботу з ними, а також з LINQ.

В майбутньому проект буде розширюватись. Буде розширюватись функціонал, міграції, оптимізація.

Отже, розробка обліку відвідування занять стала дуже цікавим досвідом, що дав мені нові навички розробки програм та більше мотивації розробляти більше таких проектів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Learn Entity Framework and ADO.NET. URL: <https://github.com/TempnameIRN199/ADO.NET.git>
2. Learn Entity Framework Core. URL: <https://www.udemy.com/course/entity-framework-core-the-complete-guide-net-5/>
3. Learn SQL. URL: <https://www.udemy.com/course/sql-oracle-certification/>