Картина, която съдържа текст, графична колекция

Описанието е генерирано автоматичноТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра „КНТ“

**КУРСОВ ПРОЕКТ**

**по дисциплината „Софтуерни технологии”**

**на тема: „Система за управление на училище”**

|  |  |
| --- | --- |
| Изготвил: Ивайло Илиев |  |
| Специалност: КСТ |  |
| Група: 3б |  |
| Факултетен номер: 22621518  Курс: 3ти | Проверил: ........................  ас. д-р Даниела Петрова |

2025

Съдържание

1. Задание
2. Кратко описание на програмата
3. Обща структура на програмата
4. Тестови резултати
5. Изводи и бъдещо развитие
6. Source код

зазаддд

**I. Задание**

**Разработване на софтуерна система за управление на училище, която позволява администриране на ученици, учители и предмети.**

**Програмата трябва да включва възможности за добавяне, редактиране и изтриване на данни, както и извършване на справки и генериране на отчети.**

**II.** **Кратко описание на програмата**

1. **Предназначение:**

Програмата **School** е предназначена за управление на училищни данни, включително информация за ученици, учители и предмети. Системата позволява:

* Добавяне и редактиране на информация за ученици и учители.
* Свързване на учениците с предмети и учители.
* Филтриране на ученици по оценки.
* Генериране на справки и статистики.

1. **Данни:**

* **Student**: Име, фамилия, клас, списък с оценки.
* **Teacher**: Име, фамилия, предмети, които преподава.
* **Subject**: Име, учител, списък с ученици.
* **Grade**: Стойност на оценката, ученик, предмет.

1. **Вход и изход на данни:**

* Въвеждане на данни чрез графичен интерфейс.
* Извеждане на справки чрез таблици и текстови полета.
* Запазване на данни в база данни (MS SQL Server).

**III. Обща структура на програмата**

**1. Utilities/Messages.cs**

**- Помощен клас за съобщения (съдържа само константни стрингове)**

**2. Form1 (Главна форма)**

**Описание:** Основен прозорец на приложението, който позволява навигация между различните секции.

**Основни методи:**

* private void Form1\_Load(object sender, EventArgs e) – зареждане на данни
* private void btnStudents\_Click(object sender, EventArgs e) – справка за учениците
* private void btnTeachers\_Click(object sender, EventArgs e) – справка за учителите
* private void button1\_Click(object sender, EventArgs e)– отваря формата за управление на учениците.
* private void button2\_Click(object sender, EventArgs e)– отваря формата за управление на учителите.
* private void btnExams\_Click(object sender, EventArgs e)– отваря формата за управление на изпитите.
* private void btnClose\_Click(object sender, EventArgs e) – затваря програмата

**3. Form2 (Форма за управление на учениците)**

**Описание:** Позволява добавяне, редактиране и изтриване на ученици.

**Основни методи:**

* void Reset() – изчистване на полетата
* private void btnClose\_Click(object sender, EventArgs e)– затваряне на form2 и отваряне на form1
* bool CheckEGN() – проверява дали е валидно ЕГН-то
* bool ValidateBirthDate() -проверка дали е дата и е валидна
* private void btnSearch\_Click(object sender, EventArgs e) – търсене на ученик в базата
* private void btnSave\_Click(object sender, EventArgs e) – редактиране на ученик
* private void btnNew\_Click(object sender, EventArgs e)– добавяне на нов ученик
* private void btnDelete\_Click(object sender, EventArgs e)– изтрива ученик от системата.
* private void btnReset\_Click(object sender, EventArgs e) – Reset

**4. Form3 (Форма за управление на учителите)**

**Описание:** Позволява управление на информацията за учителите.

**Основни методи:**

* private void btnSearch\_Click(object sender, EventArgs e)– зарежда учител.
* private void btnNew\_Click(object sender, EventArgs e)– добавя нов учител.
* private void btnSave\_Click(object sender, EventArgs e)– редактира избран учител.
* private void btnDelete\_Click(object sender, EventArgs e)– премахва учител от системата.
* void Reset() – изчистване на полетата
* private void btnReset\_Click(object sender, EventArgs e) – Reset
* private void btnClose\_Click(object sender, EventArgs e)– затваряне на form3 и отваряне на form1

**5. Form4 (Форма за справки на изпити)**

**Описание:** Извеждане на справки свързани с изпитите, учителите и учениците

**Основни методи:**

* void Reset() – презарежда datagridview
* private void LoadExamData()– зарежда изпитите от базата данни.
* private void Form4\_Load(object sender, EventArgs e) - LoadExamData();
* private void dataGridView1\_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)– извеждане на изпит в етикети
* private void btnClose\_Click(object sender, EventArgs e)– затваряне на form4 и отваряне на form1
* private void LoadStudentsByGrade(decimal grade) – връща справка
* private void btnGradeFilter\_Click(object sender, EventArgs e) – изпълнява LoadStudentsByGrade()
* private void LoadTopStudentsBySubject(string subject) – връща справка
* private void btnSubjectFilter\_Click(object sender, EventArgs e) – изпълнява LoadTopStudentsBySubject()

**6. Form5 (Форма за управление на изпити)**

**Описание:** Управление на информацията за изпитите.

**Основни методи:**

* void Reset() – изчистване на полетата
* bool ValidateDate() - проверка за валидна дата
* bool IsDecimal(string textboox) - проверка за типа
* bool IsInt(string textboox) - проверка за типа
* bool IdStudent() – проверка за валидно Id
* bool IdTeacher()– проверка за валидно Id
* private void btnSearch\_Click(object sender, EventArgs e) – зарежда изпит
* private void btnSave\_Click(object sender, EventArgs e)- редактира избран изпит
* private void btnCreate\_Click(object sender, EventArgs e) – добавя нов изпит
* private void btnDelete\_Click(object sender, EventArgs e) – изтрива избран изпит
* private void btnReset\_Click(object sender, EventArgs e) – Reset
* private void btnClose\_Click(object sender, EventArgs e)– затваряне на form5

**Описание на базата данни**

Базата данни **School** съдържа три основни таблици:

1. **Students**

* ID – уникален идентификатор (първичен ключ)
* Name, Surname, Family – пълно име на ученика
* Email – имейл за контакт
* EGN – единен граждански номер
* BirthDate – дата на раждане
* BornTown, City, District – информация за местожителство

1. **Teachers**

* ID – уникален идентификатор (първичен ключ)
* FirstName, LastName – имена на преподавателя
* Subject1, Subject2 – преподавани предмети

1. **Exams**

* ID – уникален идентификатор (първичен ключ)
* StudentID – външен ключ към Students
* TeacherID – външен ключ към Teachers
* Subject – предмет на изпита
* ExamDate – дата на изпита
* Grade – получена оценка
* MaxGrade – максимална възможна оценка (по подразбиране 6.00)
* Comments – допълнителни бележки

A screenshot of a computer

AI-generated content may be incorrect.

CREATE DATABASE School

USE School

CREATE TABLE Students (

ID INT PRIMARY KEY IDENTITY(1,1),

Name NVARCHAR(50),

Surname NVARCHAR(50),

Family NVARCHAR(50),

Email NVARCHAR(100),

EGN NVARCHAR(10),

BirthDate DATE,

BornTown NVARCHAR(50),

City NVARCHAR(50),

District NVARCHAR(50)

);

CREATE TABLE Teachers (

ID INT PRIMARY KEY IDENTITY(1,1),

FirstName NVARCHAR(50),

LastName NVARCHAR(50),

Subject1 NVARCHAR(50),

Subject2 NVARCHAR(50)

);

INSERT INTO Teachers (FirstName, LastName, Subject1, Subject2)

VALUES

('Georgi', 'Ivanov', 'Mathematics', 'Physics'),

('Elena', 'Petrova', 'Biology', 'Chemistry'),

('Stoyan', 'Georgiev', 'History', 'Philosophy'),

('Maria', 'Dimitrova', 'Literature', 'Bulgarian Language'),

('Dimitar', 'Petrov', 'Geography', 'History'),

('Kristina', 'Todorova', 'English', 'German'),

('Petar', 'Stoyanov', 'IT', 'Mathematics'),

('Ivan', 'Kolev', 'PE', 'Health Education'),

('Tanya', 'Ivanova', 'Music', 'Art'),

('Borislav', 'Dimitrov', 'Physics', 'Chemistry'),

('Stefka', 'Mihailova', 'French', 'Spanish'),

('Hristo', 'Kostov', 'Philosophy', 'Sociology'),

('Valentina', 'Gospodinova', 'Bulgarian Language', 'Literature'),

('Plamen', 'Yordanov', 'Mathematics', 'IT'),

('Silvia', 'Ivanova', 'History', 'Geography'),

('Nikolay', 'Stoev', 'Physics', 'Astronomy'),

('Zornitsa', 'Daneva', 'Music', 'Art'),

('Yordan', 'Vasilev', 'PE', 'Health Education'),

('Daniela', 'Dimitrova', 'Spanish', 'English'),

('Rosen', 'Todorov', 'Chemistry', 'Biology');

--drop table Students

INSERT INTO Students (Name, Surname, Family, Email, EGN, BirthDate, City, District, BornTown)

VALUES

('Ivan', 'Petrov', 'Ivanov', 'ivan.petrov@example.com', '1234567890', '2005-06-15', 'Sofia', 'Mladost', 'Sofia'),

('Maria', 'Georgieva', 'Petrova', 'maria.georgieva@example.com', '0987654321', '2006-08-22', 'Plovdiv', 'Trakia', 'Plovdiv'),

('Georgi', 'Ivanov', 'Dimitrov', 'georgi.ivanov@example.com', '3456789012', '2005-11-10', 'Varna', 'Levski', 'Varna'),

('Elena', 'Todorova', 'Stoyanova', 'elena.todorova@example.com', '5678901234', '2004-05-30', 'Burgas', 'Zornitsa', 'Burgas'),

('Stoyan', 'Dimitrov', 'Petkov', 'stoyan.dimitrov@example.com', '6789012345', '2006-12-01', 'Ruse', 'Center', 'Ruse'),

('Nikola', 'Petkov', 'Iliev', 'nikola.petkov@example.com', '9988776655', '2005-09-20', 'Dobrich', 'Druzhba', 'Dobrich'),

('Viktoria', 'Stoyanova', 'Koleva', 'viktoria.stoyanova@example.com', '8877665544', '2006-03-12', 'Shumen', 'Center', 'Shumen'),

('Alexander', 'Mihailov', 'Georgiev', 'alex.mihailov@example.com', '7766554433', '2007-01-05', 'Pazardzhik', 'Iztok', 'Pazardzhik'),

('Kristina', 'Kirilova', 'Ivanova', 'kristina.kirilova@example.com', '6655443322', '2004-11-23', 'Veliko Tarnovo', 'Kartala', 'Veliko Tarnovo'),

('Martin', 'Dimitrov', 'Petrov', 'martin.dimitrov@example.com', '5544332211', '2005-07-18', 'Stara Zagora', 'Zheleznik', 'Stara Zagora'),

('Stefan', 'Nikolov', 'Georgiev', 'stefan.nikolov@example.com', '5566778899', '2007-07-19', 'Pleven', 'Storgozia', 'Pleven');

INSERT INTO Students (Name, Surname, Family, Email, EGN, BirthDate, City, District, BornTown)

VALUES

('Petya', 'Ivanova', 'Georgieva', 'petya.ivanova@example.com', '1122334455', '2005-02-14', 'Sofia', 'Lozenets', 'Sofia'),

('Vasil', 'Kolev', 'Petrov', 'vasil.kolev@example.com', '2233445566', '2006-03-25', 'Plovdiv', 'Karshiaka', 'Plovdiv'),

('Ani', 'Dimitrova', 'Ivanova', 'ani.dimitrova@example.com', '3344556677', '2005-07-19', 'Varna', 'Asparuhovo', 'Varna'),

('Todor', 'Petrov', 'Georgiev', 'todor.petrov@example.com', '4455667788', '2004-09-10', 'Burgas', 'Meden Rudnik', 'Burgas'),

('Radost', 'Georgieva', 'Todorova', 'radost.georgieva@example.com', '5566778899', '2006-11-05', 'Ruse', 'Druzhba', 'Ruse'),

('Kalin', 'Iliev', 'Stoyanov', 'kalin.iliev@example.com', '6677889900', '2005-04-30', 'Stara Zagora', 'Zheleznik', 'Stara Zagora'),

('Desislava', 'Stoyanova', 'Petrova', 'desislava.stoyanova@example.com', '7788990011', '2006-08-15', 'Pleven', 'Storgozia', 'Pleven'),

('Hristo', 'Todorov', 'Dimitrov', 'hristo.todorov@example.com', '8899001122', '2004-12-20', 'Dobrich', 'Druzhba', 'Dobrich'),

('Silvia', 'Kirilova', 'Ivanova', 'silvia.kirilova@example.com', '9900112233', '2005-10-12', 'Shumen', 'Center', 'Shumen'),

('Lyubomir', 'Vasilev', 'Georgiev', 'lyubomir.vasilev@example.com', '0011223344', '2006-01-25', 'Pazardzhik', 'Iztok', 'Pazardzhik'),

('Gabriela', 'Nikolova', 'Petrova', 'gabriela.nikolova@example.com', '1122334456', '2005-03-18', 'Veliko Tarnovo', 'Kartala', 'Veliko Tarnovo'),

('Radoslav', 'Petkov', 'Ivanov', 'radoslav.petkov@example.com', '2233445567', '2006-06-22', 'Sofia', 'Mladost', 'Sofia'),

('Yana', 'Ivanova', 'Georgieva', 'yana.ivanova@example.com', '3344556678', '2005-09-14', 'Plovdiv', 'Trakia', 'Plovdiv'),

('Krasimir', 'Georgiev', 'Dimitrov', 'krasimir.georgiev@example.com', '4455667789', '2004-11-30', 'Varna', 'Levski', 'Varna'),

('Maya', 'Dimitrova', 'Stoyanova', 'maya.dimitrova@example.com', '5566778890', '2006-02-10', 'Burgas', 'Zornitsa', 'Burgas'),

('Stanislav', 'Petrov', 'Iliev', 'stanislav.petrov@example.com', '6677889901', '2005-05-05', 'Ruse', 'Center', 'Ruse'),

('Teodora', 'Koleva', 'Todorova', 'teodora.koleva@example.com', '7788990012', '2006-07-19', 'Stara Zagora', 'Zheleznik', 'Stara Zagora'),

('Boris', 'Ivanov', 'Petrov', 'boris.ivanov@example.com', '8899001123', '2004-08-12', 'Pleven', 'Storgozia', 'Pleven'),

('Vanya', 'Todorova', 'Georgieva', 'vanya.todorova@example.com', '9900112234', '2005-12-25', 'Dobrich', 'Druzhba', 'Dobrich'),

('Plamen', 'Stoyanov', 'Dimitrov', 'plamen.stoyanov@example.com', '0011223345', '2006-04-18', 'Shumen', 'Center', 'Shumen'),

('Ralitsa', 'Petrova', 'Ivanova', 'ralitsa.petrova@example.com', '1122334457', '2005-01-30', 'Pazardzhik', 'Iztok', 'Pazardzhik'),

('Ivaylo', 'Georgiev', 'Kirilov', 'ivaylo.georgiev@example.com', '2233445568', '2006-03-15', 'Veliko Tarnovo', 'Kartala', 'Veliko Tarnovo'),

('Nadezhda', 'Ilieva', 'Petrova', 'nadezhda.ilieva@example.com', '3344556679', '2005-06-20', 'Sofia', 'Mladost', 'Sofia'),

('Simeon', 'Dimitrov', 'Todorov', 'simeon.dimitrov@example.com', '4455667790', '2004-10-05', 'Plovdiv', 'Trakia', 'Plovdiv'),

('Eva', 'Koleva', 'Georgieva', 'eva.koleva@example.com', '5566778891', '2006-09-12', 'Varna', 'Levski', 'Varna'),

('Vladimir', 'Petrov', 'Stoyanov', 'vladimir.petrov@example.com', '6677889902', '2005-02-28', 'Burgas', 'Zornitsa', 'Burgas'),

('Zornitsa', 'Ivanova', 'Dimitrova', 'zornitsa.ivanova@example.com', '7788990013', '2006-11-15', 'Ruse', 'Center', 'Ruse'),

('Kaloyan', 'Todorov', 'Petrov', 'kaloyan.todorov@example.com', '8899001124', '2004-07-22', 'Stara Zagora', 'Zheleznik', 'Stara Zagora'),

('Gergana', 'Georgieva', 'Ilieva', 'gergana.georgieva@example.com', '9900112235', '2005-04-10', 'Pleven', 'Storgozia', 'Pleven'),

('Dimitar', 'Petrov', 'Kolev', 'dimitar.petrov@example.com', '0011223346', '2006-08-05', 'Dobrich', 'Druzhba', 'Dobrich'),

('Radka', 'Ilieva', 'Todorova', 'radka.ilieva@example.com', '1122334458', '2005-12-18', 'Shumen', 'Center', 'Shumen'),

('Nikolay', 'Stoyanov', 'Georgiev', 'nikolay.stoyanov@example.com', '2233445569', '2006-05-22', 'Pazardzhik', 'Iztok', 'Pazardzhik'),

('Mihaela', 'Dimitrova', 'Petrova', 'mihaela.dimitrova@example.com', '3344556680', '2005-09-30', 'Veliko Tarnovo', 'Kartala', 'Veliko Tarnovo'),

('Tihomir', 'Kolev', 'Ivanov', 'tihomir.kolev@example.com', '4455667791', '2004-03-15', 'Sofia', 'Mladost', 'Sofia'),

('Yordan', 'Petrov', 'Georgiev', 'yordan.petrov@example.com', '5566778892', '2006-06-10', 'Plovdiv', 'Trakia', 'Plovdiv'),

('Violeta', 'Ivanova', 'Dimitrova', 'violeta.ivanova@example.com', '6677889903', '2005-01-25', 'Varna', 'Levski', 'Varna'),

('Kamen', 'Todorov', 'Petrov', 'kamen.todorov@example.com', '7788990014', '2006-10-12', 'Burgas', 'Zornitsa', 'Burgas'),

('Elena', 'Georgieva', 'Stoyanova', 'elena.georgieva@example.com', '8899001125', '2004-04-18', 'Ruse', 'Center', 'Ruse'),

('Stanimir', 'Iliev', 'Kolev', 'stanimir.iliev@example.com', '9900112236', '2005-07-22', 'Stara Zagora', 'Zheleznik', 'Stara Zagora');

CREATE TABLE Exams (

ID INT PRIMARY KEY IDENTITY(1,1),

StudentID INT FOREIGN KEY REFERENCES Students(ID),

TeacherID INT FOREIGN KEY REFERENCES Teachers(ID),

Subject NVARCHAR(50),

ExamDate DATE,

Grade DECIMAL(4,2),

MaxGrade DECIMAL(4,2) DEFAULT 6.00,

Comments NVARCHAR(255)

);

INSERT INTO Exams (StudentID, TeacherID, Subject, ExamDate, Grade, Comments)

VALUES

(1, 7, 'Mathematics', '2025-02-15', 5.50, 'Excellent problem-solving skills'),

(2, 10, 'Physics', '2025-02-16', 4.00, 'Needs improvement in formulas'),

(3, 5, 'Geography', '2025-02-17', 5.75, 'Great knowledge of maps'),

(4, 12, 'Philosophy', '2025-02-18', 3.50, 'Critical thinking needs work'),

(5, 3, 'History', '2025-02-19', 5.00, 'Well-prepared for the exam'),

(6, 8, 'Physical Education', '2025-02-20', 6.00, 'Outstanding performance'),

(7, 14, 'IT', '2025-02-21', 4.75, 'Good understanding of coding'),

(8, 6, 'English', '2025-02-22', 5.25, 'Strong vocabulary and grammar'),

(9, 2, 'Biology', '2025-02-23', 4.50, 'Average understanding of genetics'),

(10, 9, 'Music', '2025-02-24', 5.80, 'Beautiful singing voice'),

(11, 4, 'Chemistry', '2025-02-25', 4.20, 'Needs to practice more lab experiments'),

(12, 1, 'Art', '2025-02-26', 5.90, 'Creative and expressive work'),

(13, 13, 'French', '2025-02-27', 4.60, 'Good pronunciation but needs grammar improvement'),

(14, 15, 'Economics', '2025-02-28', 5.30, 'Understands financial concepts well'),

(15, 7, 'Mathematics', '2025-02-28', 4.80, 'Shows progress in problem-solving'),

(3, 5, 'Geography', '2025-02-17', 5.75, 'Great knowledge of maps'),

(3, 10, 'Physics', '2025-03-02', 4.20, 'Struggles with some calculations'),

(6, 7, 'Mathematics', '2025-03-05', 5.00, 'Consistent improvement'),

(10, 2, 'Biology', '2025-03-10', 4.80, 'Needs more revision on cell structure'),

(9, 9, 'Music', '2025-03-15', 6.00, 'Outstanding performance and talent');

SELECT

e.ID AS ExamID,

s.ID AS StudentID,

s.Name + ' ' + s.Surname + ' ' + s.Family AS StudentName,

t.ID AS TeacherID,

t.FirstName + ' ' + t.LastName AS TeacherName,

e.Subject,

e.ExamDate,

e.Grade,

e.MaxGrade,

e.Comments

FROM Exams e

JOIN Students s ON e.StudentID = s.ID

JOIN Teachers t ON e.TeacherID = t.ID

--ORDER BY e.ExamDate DESC;

SELECT

s.ID AS StudentID,

s.Name + ' ' + s.Surname + ' ' + s.Family AS StudentName,

e.Grade,

e.Subject,

e.ExamDate

FROM Exams e

JOIN Students s ON e.StudentID = s.ID

ORDER BY e.ExamDate DESC;

-------------------------------------

SELECT

e.ID AS ExamID,

s.ID AS StudentID,

s.Name + ' ' + s.Surname + ' ' + s.Family AS StudentName,

t.ID AS TeacherID,

t.FirstName + ' ' + t.LastName AS TeacherName,

e.Subject,

e.ExamDate,

e.Grade,

e.MaxGrade,

e.Comments

FROM Exams e

JOIN Students s ON e.StudentID = s.ID

JOIN Teachers t ON e.TeacherID = t.ID

WHERE e.Subject = 'Mathematics'

ORDER BY e.Grade DESC;

-------------------------------------

SELECT

e.ID AS ExamID,

s.ID AS StudentID,

s.Name + ' ' + s.Surname + ' ' + s.Family AS StudentName,

t.ID AS TeacherID,

t.FirstName + ' ' + t.LastName AS TeacherName,

e.Subject,

e.ExamDate,

e.Grade,

e.MaxGrade,

e.Comments

FROM Exams e

JOIN Students s ON e.StudentID = s.ID

JOIN Teachers t ON e.TeacherID = t.ID

WHERE e.Subject= 'Biology'

----------------------------------

SELECT

s.ID AS StudentID,

s.Name + ' ' + s.Surname + ' ' + s.Family AS StudentName,

ROUND(CAST(AVG(e.Grade) AS DECIMAL(10, 2)), 2) AS AverageGrade

FROM Exams e

JOIN Students s ON e.StudentID = s.ID

WHERE s.ID = 6

GROUP BY s.ID, s.Name, s.Surname, s.Family;

-----------------------------------------------

SELECT

t.ID AS TeacherID,

t.FirstName + ' ' + t.LastName AS TeacherName,

COUNT(e.ID) AS ExamCount

FROM Exams e

JOIN Teachers t ON e.TeacherID = t.ID

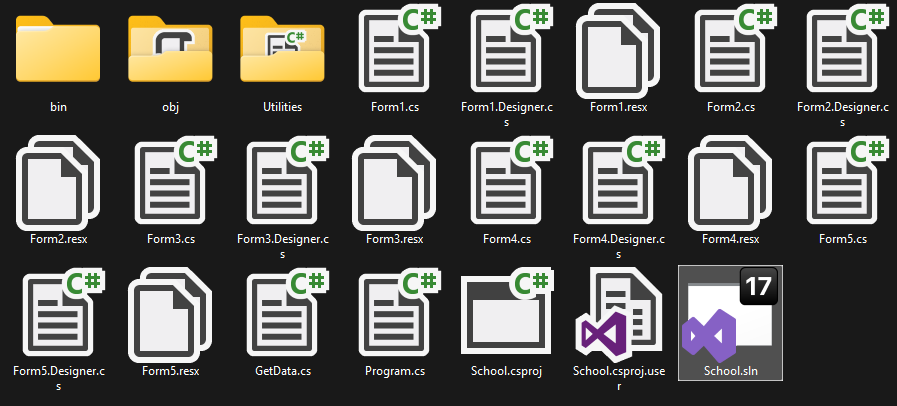
GROUP BY t.ID, t.FirstName, t.LastName

ORDER BY ExamCount DESC;

select \* from Exams

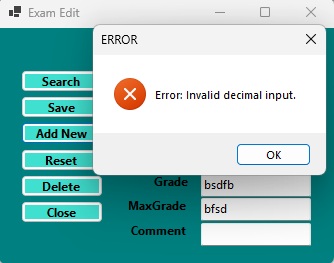
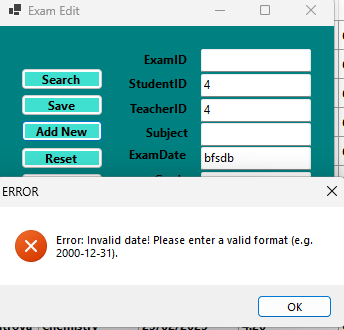
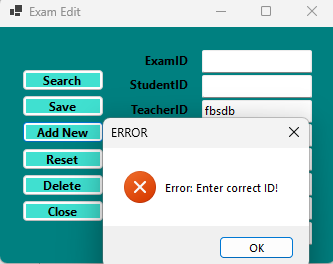
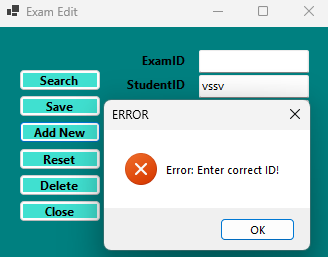
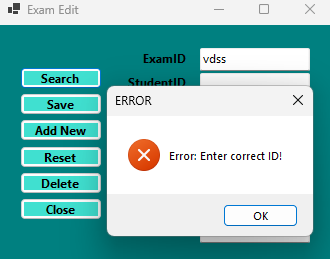
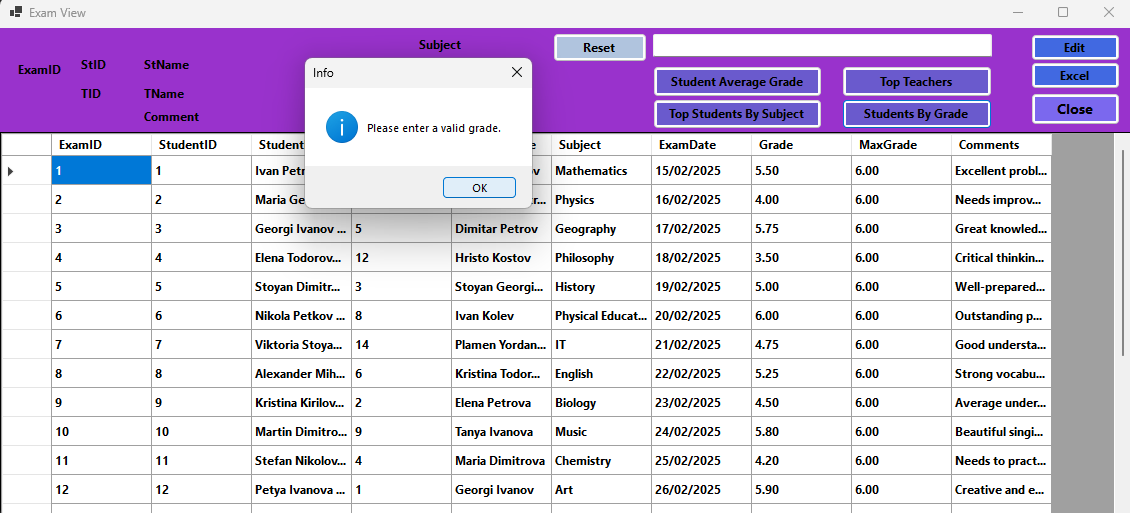
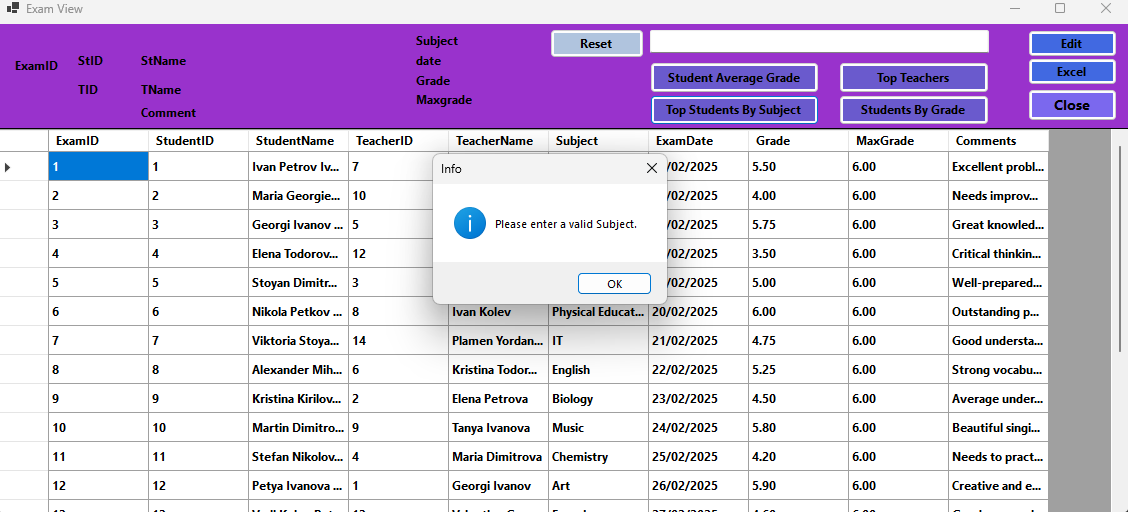
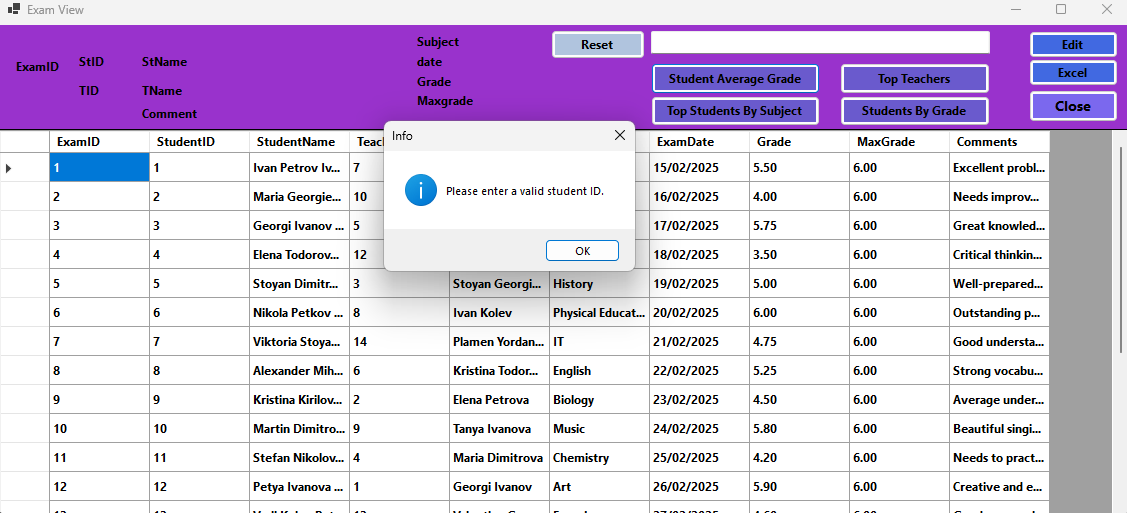
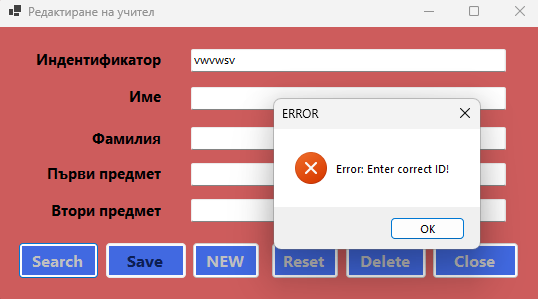
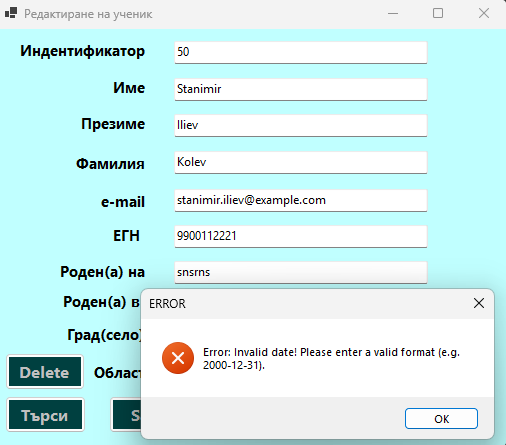
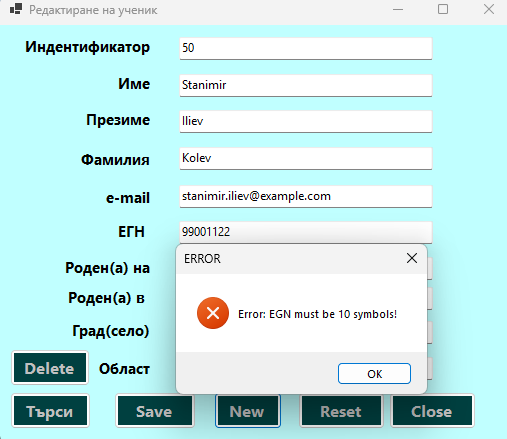
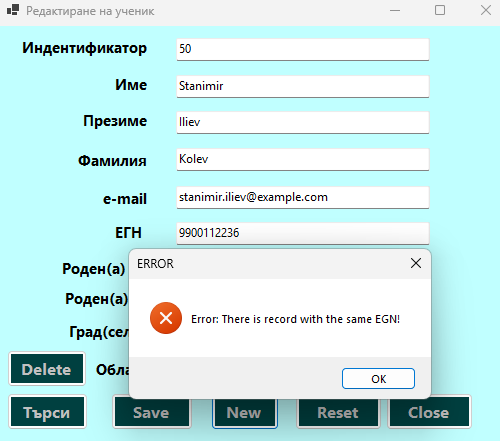
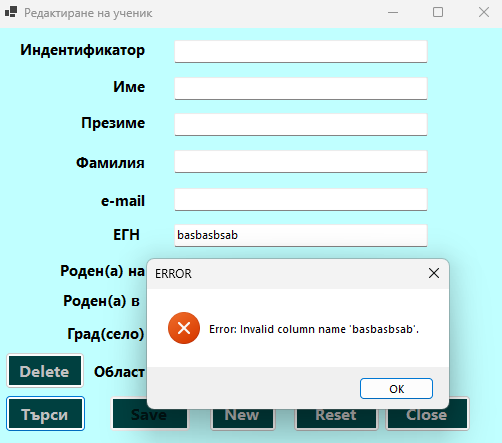
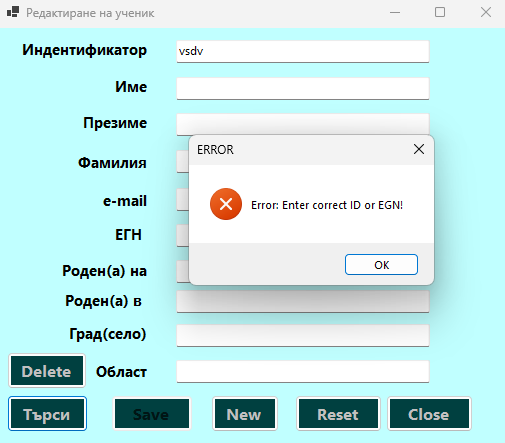
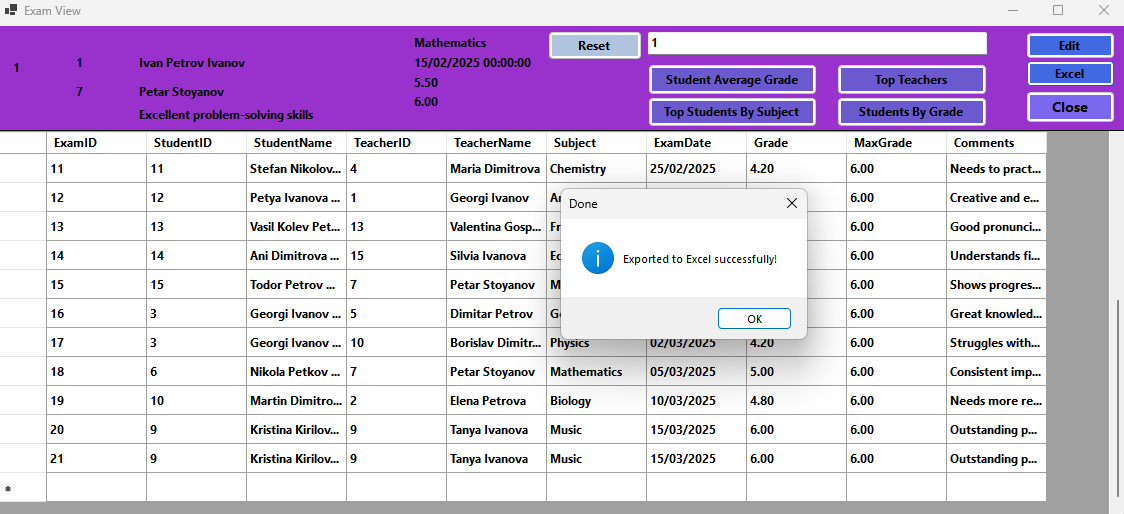
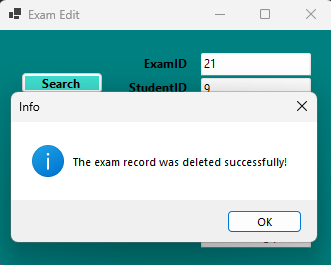
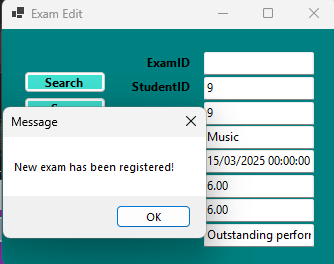
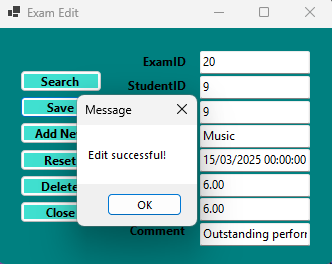
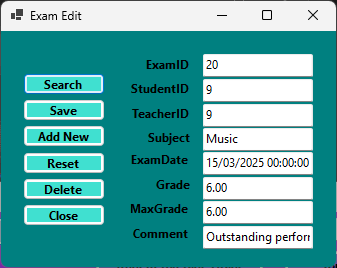
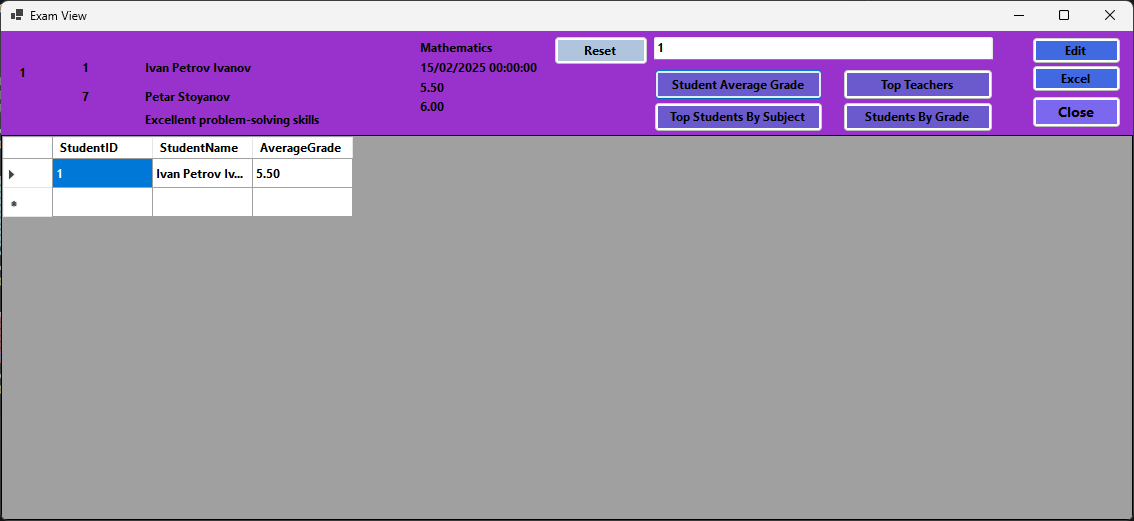
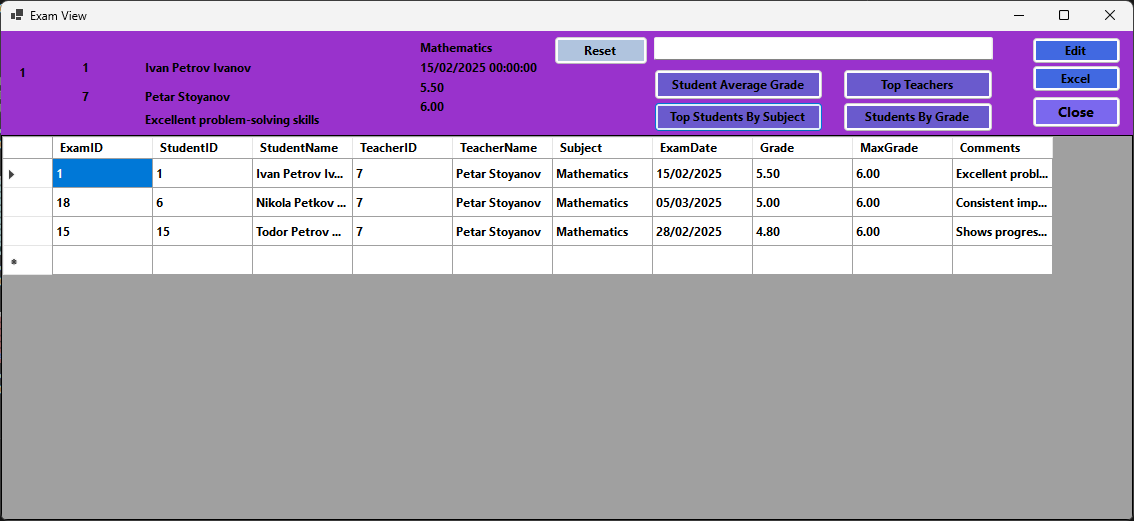
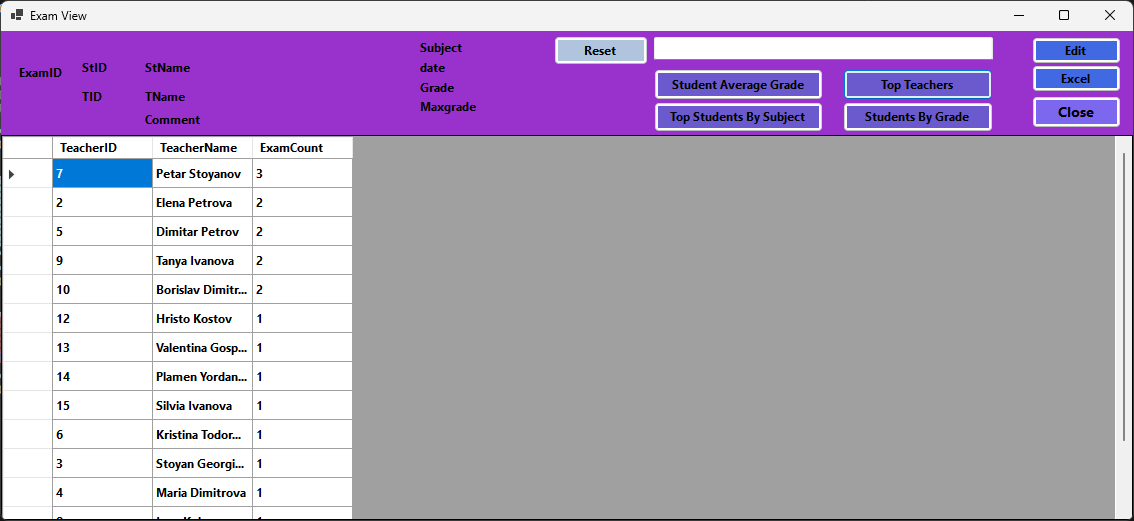
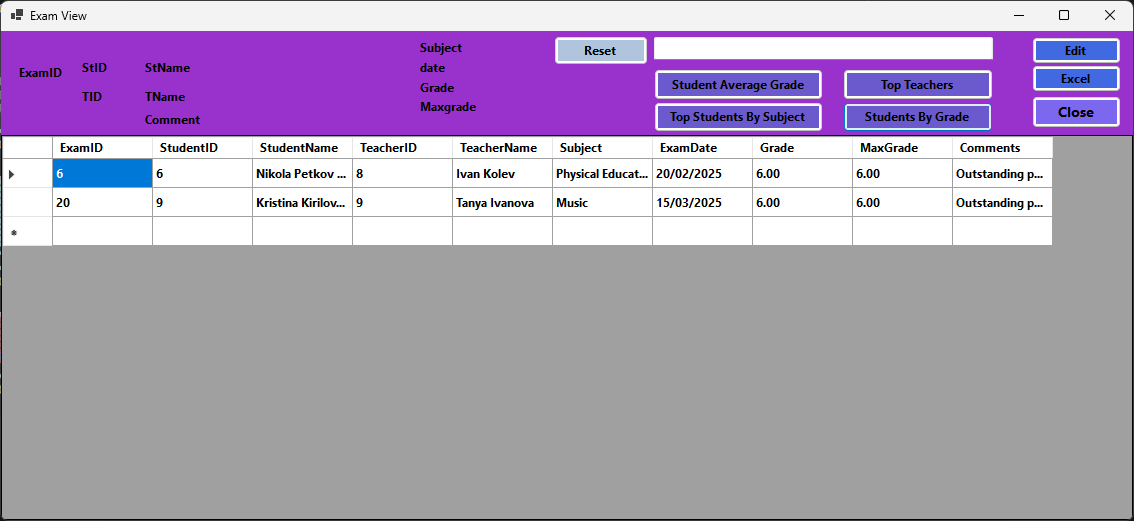
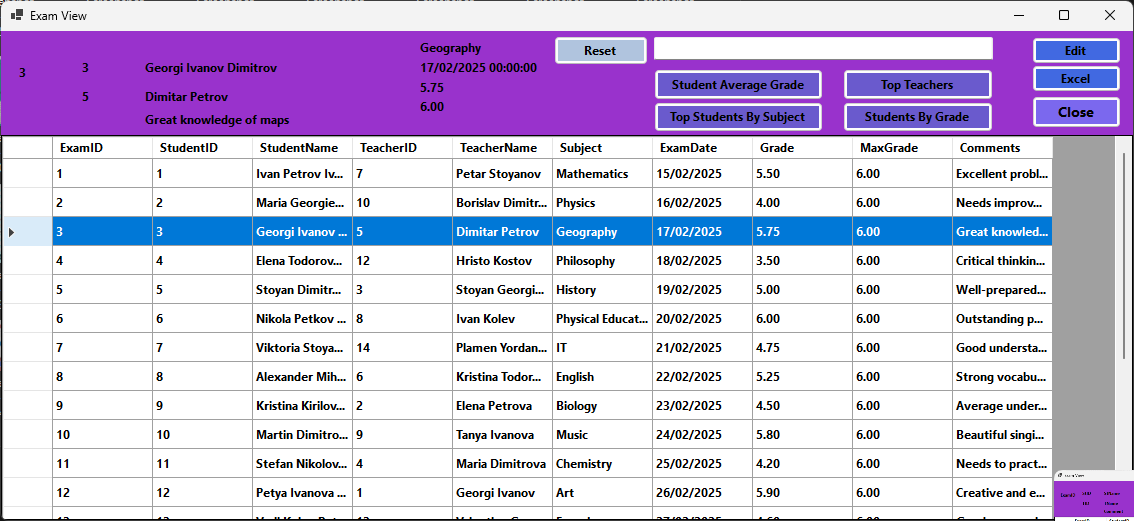
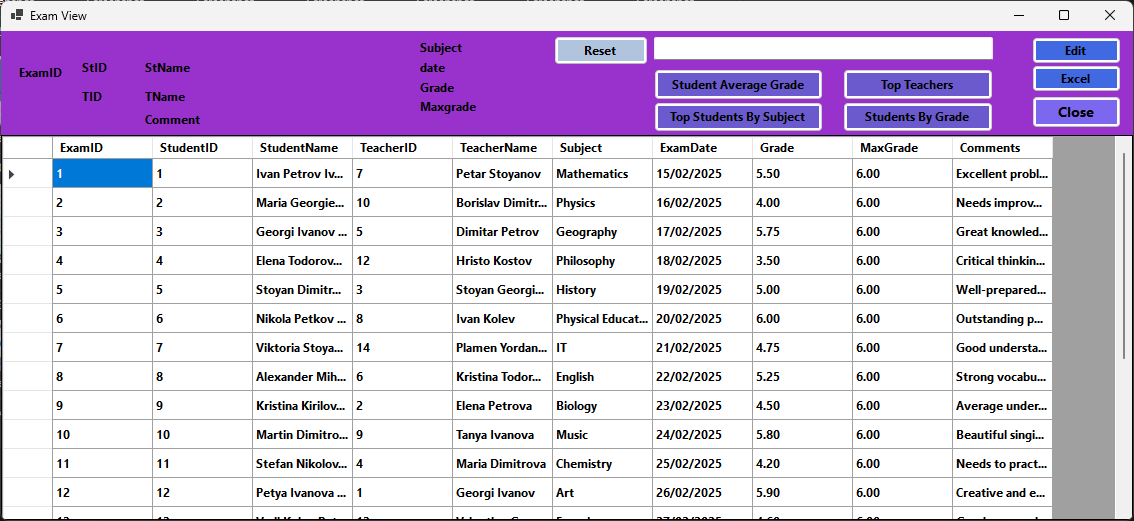
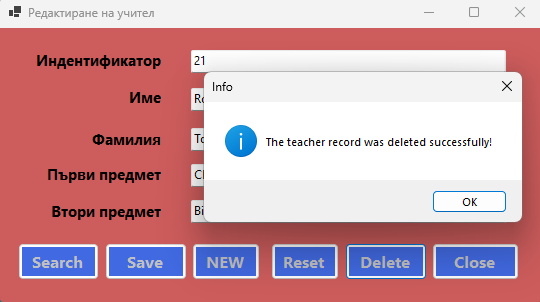
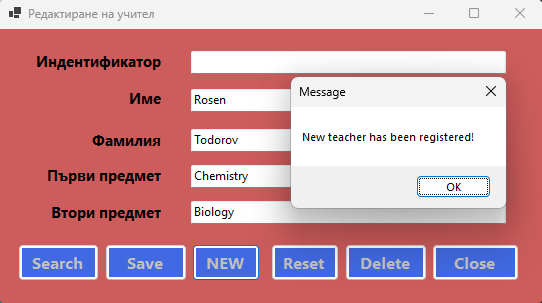
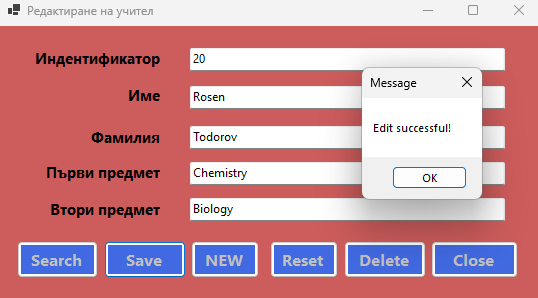
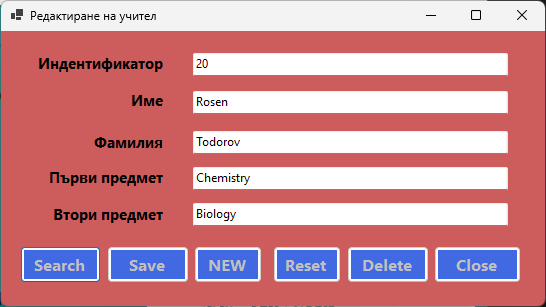
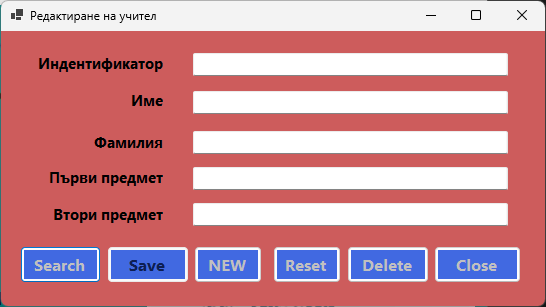
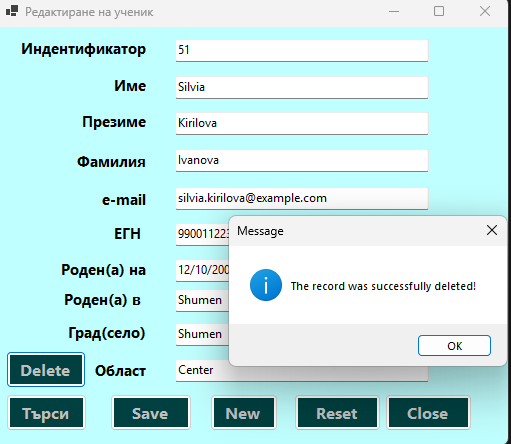
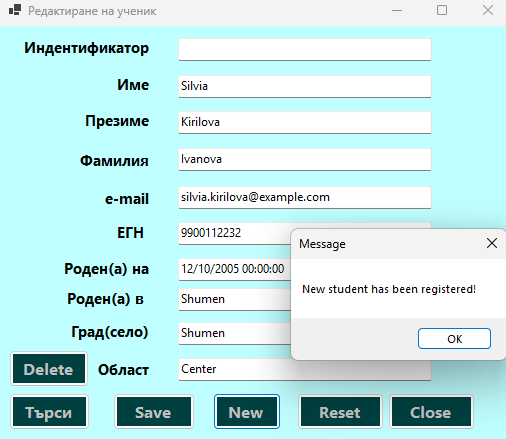
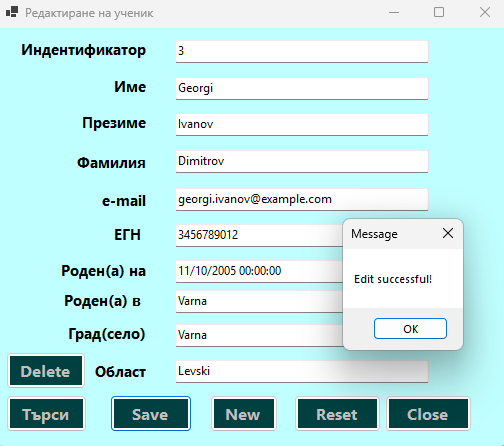
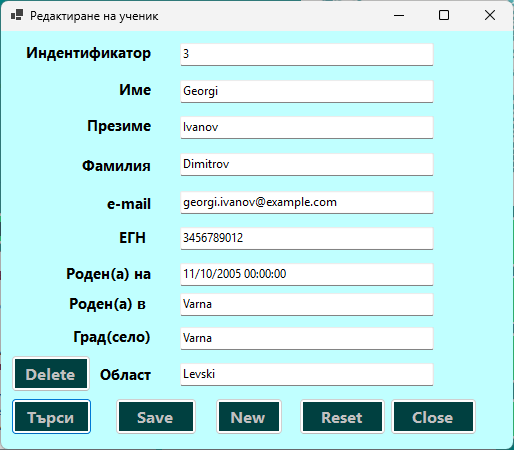
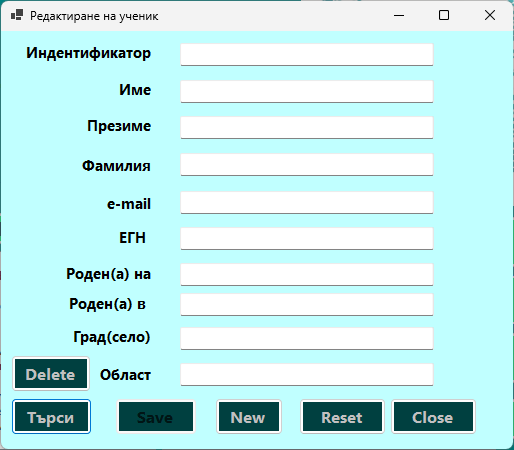
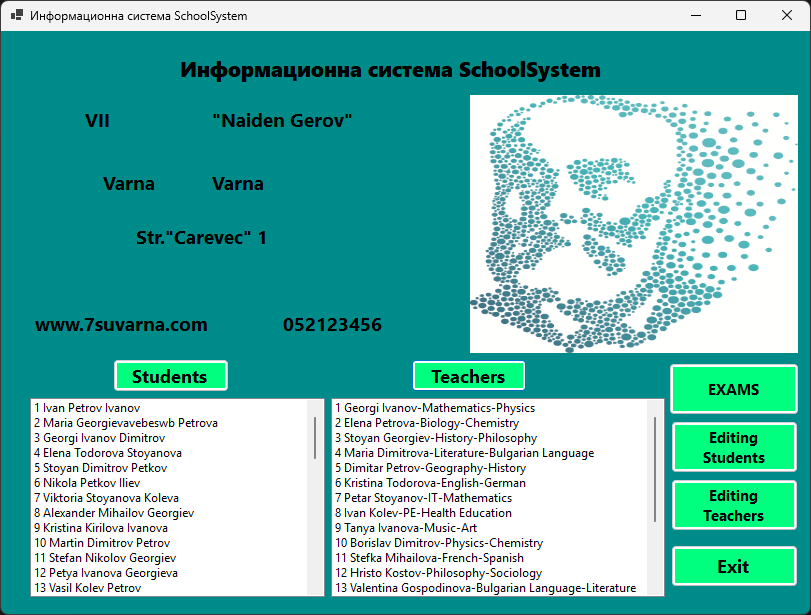
**IV. Тестови резултати**

**Нужните файлове за работа с програмата:**



A close-up of a folder

AI-generated content may be incorrect.

Примерна работа на програмата:

**V. Изводи и бъдещо развитие**

* Изводи:

Разработената система за управление на училище демонстрира основните принципи на съвременно приложение с графичен потребителски интерфейс и свързана база данни. Чрез реализирането на различни форми за администриране на ученици, учители и изпити, проектът постига:

* Успешно свързване с MS SQL Server и работа с релационни таблици;
* Интуитивен и функционален потребителски интерфейс с помощта на Windows Forms;
* Коректно валидиране на входни данни и изчерпателно обработване на грешки;
* Възможности за извеждане на справки, включително по оценки, предмети и статистика.

Системата е стабилна в основната си функционалност и може да бъде използвана като основа за реални учебни нужди или като база за бъдещи софтуерни разширения.

* Бъдещо развитие:

За надграждане на проекта и повишаване на неговата практическа стойност, се предлагат следните възможности за развитие:

* Въвеждане на роли и потребители: администратор, учител, ученик, с различни нива на достъп;
* Онлайн достъп чрез ASP.NET Web API или Blazor – възможност за достъп до системата чрез уеб браузър;
* Автоматично изпращане на известия (например имейли при насрочени изпити или получена оценка);
* Импорт и експорт на данни в Excel или PDF формат;
* Мобилна версия чрез .NET MAUI или Xamarin;
* Интеграция с график за учебни часове и учебна програма;
* Машинно обучение за анализ на резултатите на учениците – предвиждане на напредък или отклонения.

**VI.Source код**

**Utilities/Messages.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace School.Utilities

{

public class Messages

{

public const string IdOrEGN = "Enter correct ID or EGN!";

public const string sizeEGN = "EGN must be 10 symbols!";

public const string doesntExist = "There is no such record!";

public const string EGNExist = "There is record with the same EGN!";

public static readonly string InvalidDate = "Invalid date! Please enter a valid format (e.g. 2000-12-31).";

public static readonly string incorrectId = "Enter correct ID!";

public static readonly string notDecimal = "Invalid decimal input.";

public const string stExist = "There is no such student!";

public const string tExist = "There is no such Teacher!";

}

}

Form1

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 School

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

public static object schData { get; internal set; }

private void Form1\_Load(object sender, EventArgs e)

{

pbLogo.Image = Image.FromFile(@"C:\Users\iwail\Desktop\SQL+VS\school5\EmbeddedImage.jpg");

//SqlConnection cn = new SqlConnection(@"Server=IVO03\_HELIOS\SQLEXPRESS;Database=School;Integrated Security = True");

SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True");

string queryString =

"SELECT Name, Patron, City, District, Address, Website, Telephon FROM SchoolData";

SqlCommand cmd = new SqlCommand(queryString, cn);

cn.Open();

using (SqlDataReader sqlDataReader = cmd.ExecuteReader())

{

if (sqlDataReader.Read())

{

lblName.Text = sqlDataReader["Name"].ToString();

lblPatron.Text = sqlDataReader["Patron"].ToString();

lblCity.Text = sqlDataReader["City"].ToString();

lblDistrict.Text = sqlDataReader["District"].ToString();

lblAddress.Text = sqlDataReader["Address"].ToString();

lblWebpage.Text = sqlDataReader["Website"].ToString();

lblTelephone.Text = sqlDataReader["Telephon"].ToString();

}

}

}

private void btnStudents\_Click(object sender, EventArgs e)

{

listBox1.Visible = true;

listBox1.Items.Clear();

//SqlConnection cn = new SqlConnection(@"Server=IVO03\_HELIOS\SQLEXPRESS;Database=School;Integrated Security = True");

SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True");

string queryString = "SELECT \* FROM Students";

SqlCommand cmd = new SqlCommand(queryString, cn);

cn.Open();

using (SqlDataReader sqlDataReader = cmd.ExecuteReader())

{

while (sqlDataReader.Read())

{

listBox1.Items.Add(sqlDataReader["ID"] + " " + sqlDataReader["Name"] + " " + sqlDataReader["Surname"] + " " + sqlDataReader["Family"]);

}

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

Application.Exit();

}

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

{

//this.Hide();

Form2 f = new Form2();

f.Show();

}

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

{

//this.Hide();

Form3 f = new Form3();

f.Show();

}

private void btnTeachers\_Click(object sender, EventArgs e)

{

listBox2.Visible = true;

listBox2.Items.Clear();

//SqlConnection cn = new SqlConnection(@"Server=IVO03\_HELIOS\SQLEXPRESS;Database=School;Integrated Security = True");

SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True");

string queryString = "SELECT \* FROM Teachers";

SqlCommand cmd = new SqlCommand(queryString, cn);

cn.Open();

using (SqlDataReader sqlDataReader = cmd.ExecuteReader())

{

while (sqlDataReader.Read())

{

listBox2.Items.Add(sqlDataReader["ID"] + " " + sqlDataReader["FirstName"] + " " + sqlDataReader["LastName"] + "-" + sqlDataReader["Subject1"] + "-" + sqlDataReader["Subject2"]);

}

}

}

private void btnExams\_Click(object sender, EventArgs e)

{

this.Hide();

Form4 f = new Form4();

f.Show();

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

Form2

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Linq;

using School.Utilities;

namespace School

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

void Reset()

{

txtID.Text = string.Empty;

txtName.Text = string.Empty;

txtSurname.Text = string.Empty;

txtFamily.Text = string.Empty;

txtEmail.Text = string.Empty;

txtEGN.Text = string.Empty;

txtBorn.Text = string.Empty;

txtTownBorn.Text = string.Empty;

txtCity.Text = string.Empty;

txtDistrict.Text = string.Empty;

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

//Form1 form1 = new Form1();

//form1.Show();

}

bool CheckEGN()

{

if (txtEGN.Text.Length != 10) throw new ArgumentException(Messages.sizeEGN);

bool exists = false;

using (SqlConnection cnn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security=True"))

{

cnn.Open();

string query = "SELECT 1 FROM Students WHERE EGN = " + txtEGN.Text;

using (SqlCommand cmmd = new SqlCommand(query, cnn))

{

cmmd.Parameters.AddWithValue("@egn", txtEGN.Text);

var res = cmmd.ExecuteScalar();

exists = res == null;

}

}

return exists;

}

bool ValidateBirthDate()

{

bool isValid = DateTime.TryParse(txtBorn.Text, out DateTime birthDate);

return isValid;

}

private void btnSearch\_Click(object sender, EventArgs e)

{

try

{

if (!int.TryParse(txtID.Text, out int result) && ((string.IsNullOrEmpty(txtEGN.Text) || string.IsNullOrWhiteSpace(txtEGN.Text))))

throw new ArgumentException(Messages.IdOrEGN);

else if (int.TryParse(txtID.Text, out int result2))

{

bool exists = false;

using (SqlConnection cnn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security=True"))

{

cnn.Open();

string query = "SELECT 1 FROM Students WHERE ID = " + int.Parse(txtID.Text);

using (SqlCommand cmmd = new SqlCommand(query, cnn))

{

var res = cmmd.ExecuteScalar();

exists = res == null;

}

}

if (exists)

{

throw new ArgumentException(Messages.doesntExist);

}

else

{

//SqlConnection cn = new SqlConnection(@"Server=IVO03\_HELIOS\SQLEXPRESS;Database=School;Integrated Security = True");

SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True");

string queryString = "SELECT \* FROM Students WHERE ID = " + int.Parse(txtID.Text);

SqlCommand cmd = new SqlCommand(queryString, cn);

cn.Open();

using (SqlDataReader sqlDataReader = cmd.ExecuteReader())

{

while (sqlDataReader.Read())

{

txtName.Text = sqlDataReader["Name"].ToString();

txtSurname.Text = sqlDataReader["Surname"].ToString();

txtFamily.Text = sqlDataReader["Family"].ToString();

txtEmail.Text = sqlDataReader["Email"].ToString();

txtEGN.Text = sqlDataReader["EGN"].ToString();

txtBorn.Text = sqlDataReader["BirthDate"].ToString();

txtTownBorn.Text = sqlDataReader["BornTown"].ToString();

txtCity.Text = sqlDataReader["City"].ToString();

txtDistrict.Text = sqlDataReader["District"].ToString();

}

}

}

}

else if (!(string.IsNullOrEmpty(txtEGN.Text) || string.IsNullOrWhiteSpace(txtEGN.Text)))

{

/\* if(txtEGN.Text.Length != 10) throw new ArgumentException(Messages.sizeEGN);

bool exists = false;

using (SqlConnection cnn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security=True"))

{

cnn.Open();

string query = "SELECT 1 FROM Students WHERE EGN = " + txtEGN.Text;

using (SqlCommand cmmd = new SqlCommand(query, cnn))

{

cmmd.Parameters.AddWithValue("@egn", txtEGN.Text);

var res = cmmd.ExecuteScalar();

exists = res == null;

}

}

\*/

if (CheckEGN())

{

throw new ArgumentException(Messages.doesntExist);

}

else

{

//SqlConnection cn = new SqlConnection(@"Server=IVO03\_HELIOS\SQLEXPRESS;Database=School;Integrated Security = True");

SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True");

string queryString = "SELECT \* FROM Students WHERE EGN = " + txtEGN.Text;

SqlCommand cmd = new SqlCommand(queryString, cn);

cn.Open();

using (SqlDataReader sqlDataReader = cmd.ExecuteReader())

{

while (sqlDataReader.Read())

{

txtID.Text = sqlDataReader["ID"].ToString();

txtName.Text = sqlDataReader["Name"].ToString();

txtSurname.Text = sqlDataReader["Surname"].ToString();

txtFamily.Text = sqlDataReader["Family"].ToString();

txtEmail.Text = sqlDataReader["Email"].ToString();

txtEGN.Text = sqlDataReader["EGN"].ToString();

txtBorn.Text = sqlDataReader["BirthDate"].ToString();

txtTownBorn.Text = sqlDataReader["BornTown"].ToString();

txtCity.Text = sqlDataReader["City"].ToString();

txtDistrict.Text = sqlDataReader["District"].ToString();

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

if (txtName.Text != "" || txtSurname.Text != "" || txtFamily.Text != "" || txtEmail.Text != "") btnSave.Enabled = true;

}

private void Form2\_Load(object sender, EventArgs e)

{

}

private void btnSave\_Click(object sender, EventArgs e)

{

try

{

if (txtEGN.Text.Length != 10) throw new ArgumentException(Messages.sizeEGN);

if (!ValidateBirthDate())

{

throw new ArgumentException(Messages.InvalidDate);

}

//SqlConnection cn = new SqlConnection(@"Server=IVO03\_HELIOS\SQLEXPRESS;Database=School;Integrated Security = True");

SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True");

cn.Open();

string queryString = "UPDATE Students set Name=@Name, Surname=@Surname, Family=@Family, " +

"Email=@Email, EGN=@EGN, BornTown=@BornTown, City=@City, " +

"District=@District, BirthDate=@BirthDate WHERE ID = @ID";

SqlCommand cmd = new SqlCommand(queryString, cn);

cmd.Parameters.AddWithValue("@ID", txtID.Text);

cmd.Parameters.AddWithValue("@Name", txtName.Text);

cmd.Parameters.AddWithValue("@Surname", txtSurname.Text);

cmd.Parameters.AddWithValue("@Family", txtFamily.Text);

cmd.Parameters.AddWithValue("@Email", txtEmail.Text);

cmd.Parameters.AddWithValue("@EGN", txtEGN.Text);

cmd.Parameters.AddWithValue("@BornTown", txtTownBorn.Text);

cmd.Parameters.AddWithValue("@City", txtCity.Text);

cmd.Parameters.AddWithValue("@District", txtDistrict.Text);

cmd.Parameters.AddWithValue("@BirthDate", DateTime.Parse(txtBorn.Text));

cmd.ExecuteNonQuery();

cn.Close();

MessageBox.Show("Edit successful!", "Message", MessageBoxButtons.OK);

Reset();

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void btnNew\_Click(object sender, EventArgs e)

{

try

{

if (!CheckEGN())

{

throw new ArgumentException(Messages.EGNExist);

}

if (!ValidateBirthDate())

{

throw new ArgumentException(Messages.InvalidDate);

}

//SqlConnection cn = new SqlConnection(@"Server=IVO03\_HELIOS\SQLEXPRESS;Database=School;Integrated Security = True");

SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True");

cn.Open();

string queryString = "INSERT INTO Students VALUES (@Name,@Surname,@Family,@Email,@EGN,@BirthDate,@BornTown,@City,@District)";

SqlCommand cmd = new SqlCommand(queryString, cn);

//cmd.Parameters.AddWithValue("@ID", txtID.Text);

cmd.Parameters.AddWithValue("@Name", txtName.Text);

cmd.Parameters.AddWithValue("@Surname", txtSurname.Text);

cmd.Parameters.AddWithValue("@Family", txtFamily.Text);

cmd.Parameters.AddWithValue("@Email", txtEmail.Text);

cmd.Parameters.AddWithValue("@EGN", txtEGN.Text);

cmd.Parameters.AddWithValue("@BirthDate", DateTime.Parse(txtBorn.Text));

cmd.Parameters.AddWithValue("@BornTown", txtTownBorn.Text);

cmd.Parameters.AddWithValue("@City", txtCity.Text);

cmd.Parameters.AddWithValue("@District", txtDistrict.Text);

cmd.ExecuteNonQuery();

cn.Close();

MessageBox.Show("New student has been registered!", "Message", MessageBoxButtons.OK);

Reset();

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void label10\_Click(object sender, EventArgs e)

{

}

private void txtTownBorn\_TextChanged(object sender, EventArgs e)

{

}

private void btnReset\_Click(object sender, EventArgs e)

{

Reset();

}

private void btnDelete\_Click(object sender, EventArgs e)

{

/\*

using (SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True;"))

{

cn.Open();

string query = "DELETE FROM Students WHERE ID = @ID";

SqlCommand cmd = new SqlCommand(query, cn);

cmd.Parameters.AddWithValue("@ID", int.Parse(txtID.Text));

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Записът е изтрит успешно!", "Съобщение", MessageBoxButtons.OK, MessageBoxIcon.Information);

Reset();

}

else

{

MessageBox.Show("Записът не беше намерен!", "Грешка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}\*/

using (SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security=True;"))

{

cn.Open();

// Delete the student

string deleteQuery = "DELETE FROM Students WHERE ID = @ID";

using (SqlCommand cmd = new SqlCommand(deleteQuery, cn))

{

cmd.Parameters.AddWithValue("@ID", int.Parse(txtID.Text));

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("The record was successfully deleted!", "Message", MessageBoxButtons.OK, MessageBoxIcon.Information);

Reset();

// OPTIONAL: Reset identity to the MAX(ID)

string reseedQuery = "DECLARE @maxId INT = ISNULL((SELECT MAX(ID) FROM Students), 0); " +

"DBCC CHECKIDENT ('Students', RESEED, @maxId)";

using (SqlCommand reseedCmd = new SqlCommand(reseedQuery, cn))

{

reseedCmd.ExecuteNonQuery();

}

}

else

{

MessageBox.Show("The record was not found!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

}

}

Form3

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Linq;

using School.Utilities;

namespace School

{

public partial class Form3 : Form

{

public Form3()

{

InitializeComponent();

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void txtSurname\_TextChanged(object sender, EventArgs e)

{

}

private void txtBorn\_TextChanged(object sender, EventArgs e)

{

}

private void label8\_Click(object sender, EventArgs e)

{

}

private void label10\_Click(object sender, EventArgs e)

{

}

private void label7\_Click(object sender, EventArgs e)

{

}

void Reset()

{

txtID.Text = string.Empty;

txtName.Text = string.Empty;

txtFamily.Text = string.Empty;

txtSubject1.Text = string.Empty;

txtSubject2.Text = string.Empty;

}

private void btnSearch\_Click(object sender, EventArgs e)

{

try

{

if (!int.TryParse(txtID.Text, out int result)) throw new ArgumentException(Messages.incorrectId);

else

{

bool exists = false;

using (SqlConnection cnn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security=True"))

{

cnn.Open();

string query = "SELECT 1 FROM Teachers WHERE ID = " + int.Parse(txtID.Text);

using (SqlCommand cmmd = new SqlCommand(query, cnn))

{

var res = cmmd.ExecuteScalar();

exists = res == null;

}

}

if (exists)

{

throw new ArgumentException(Messages.doesntExist);

}

//SqlConnection cn = new SqlConnection(@"Server=IVO03\_HELIOS\SQLEXPRESS;Database=School;Integrated Security = True");

SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True");

string queryString = "SELECT \* FROM Teachers WHERE ID = " + int.Parse(txtID.Text);

SqlCommand cmd = new SqlCommand(queryString, cn);

cn.Open();

using (SqlDataReader sqlDataReader = cmd.ExecuteReader())

{

while (sqlDataReader.Read())

{

txtName.Text = sqlDataReader["FirstName"].ToString();

txtFamily.Text = sqlDataReader["LastName"].ToString();

txtSubject1.Text = sqlDataReader["Subject1"].ToString();

txtSubject2.Text = sqlDataReader["Subject2"].ToString();

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

if (txtName.Text != "" || txtFamily.Text != "") btnSave.Enabled = true;

}

private void btnSave\_Click(object sender, EventArgs e)

{

//SqlConnection cn = new SqlConnection(@"Server=IVO03\_HELIOS\SQLEXPRESS;Database=School;Integrated Security = True");

SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True");

cn.Open();

string queryString = "UPDATE Teachers set FirstName=@Name, LastName=@Family, Subject1=@Subject1, Subject2=@Subject2 WHERE ID = " + int.Parse(txtID.Text);

SqlCommand cmd = new SqlCommand(queryString, cn);

cmd.Parameters.AddWithValue("@Name", txtName.Text);

cmd.Parameters.AddWithValue("@Family", txtFamily.Text);

cmd.Parameters.AddWithValue("@Subject1", txtSubject1.Text);

cmd.Parameters.AddWithValue("@Subject2", txtSubject2.Text);

cmd.ExecuteNonQuery();

cn.Close();

MessageBox.Show("Edit successful!", "Message", MessageBoxButtons.OK);

Reset();

}

private void btnNew\_Click(object sender, EventArgs e)

{

//SqlConnection cn = new SqlConnection(@"Server=IVO03\_HELIOS\SQLEXPRESS;Database=School;Integrated Security = True");

SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True");

cn.Open();

string queryString = "INSERT INTO Teachers VALUES (@FirstName,@LastName,@Subject1,@Subject2)";

SqlCommand cmd = new SqlCommand(queryString, cn);

//cmd.Parameters.AddWithValue("@ID", txtID.Text);

cmd.Parameters.AddWithValue("@FirstName", txtName.Text);

cmd.Parameters.AddWithValue("@LastName", txtFamily.Text);

cmd.Parameters.AddWithValue("@Subject1", txtSubject1.Text);

cmd.Parameters.AddWithValue("@Subject2", txtSubject2.Text);

cmd.ExecuteNonQuery();

cn.Close();

MessageBox.Show("New teacher has been registered!", "Message", MessageBoxButtons.OK);

Reset();

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

// Form1 f1 = new Form1();

// f1.Show();

}

private void btnReset\_Click(object sender, EventArgs e)

{

Reset();

}

private void btnDelete\_Click(object sender, EventArgs e)

{

/\* using (SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True;"))

{

cn.Open();

string query = "DELETE FROM Teachers WHERE ID = " + int.Parse(txtID.Text);

SqlCommand cmd = new SqlCommand(query, cn);

//cmd.Parameters.AddWithValue("@ID", int.Parse(txtID.Text));

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Записът е изтрит успешно!", "Съобщение", MessageBoxButtons.OK, MessageBoxIcon.Information);

Reset();

}

else

{

MessageBox.Show("Записът не беше намерен!", "Грешка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}\*/

using (SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security=True;"))

{

cn.Open();

string deleteQuery = "DELETE FROM Teachers WHERE ID = @ID";

using (SqlCommand cmd = new SqlCommand(deleteQuery, cn))

{

cmd.Parameters.AddWithValue("@ID", int.Parse(txtID.Text));

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("The teacher record was deleted successfully!", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);

Reset();

// Reset identity to the current MAX(ID)

string reseedQuery = @"

DECLARE @maxId INT = ISNULL((SELECT MAX(ID) FROM Teachers), 0);

DBCC CHECKIDENT ('Teachers', RESEED, @maxId);";

using (SqlCommand reseedCmd = new SqlCommand(reseedQuery, cn))

{

reseedCmd.ExecuteNonQuery();

}

}

else

{

MessageBox.Show("The record was not found!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

}

}

Form4

using ClosedXML.Excel;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Xml.Linq;

namespace School

{

public partial class Form4 : Form

{

public Form4()

{

InitializeComponent();

}

string connectionString = @"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True";

private void LoadExamData()

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

try

{

conn.Open();

string query = @"

SELECT

e.ID AS ExamID,

s.ID AS StudentID,

s.Name + ' ' + s.Surname + ' ' + s.Family AS StudentName,

t.ID AS TeacherID,

t.FirstName + ' ' + t.LastName AS TeacherName,

e.Subject,

e.ExamDate,

e.Grade,

e.MaxGrade,

e.Comments

FROM Exams e

JOIN Students s ON e.StudentID = s.ID

JOIN Teachers t ON e.TeacherID = t.ID";

SqlDataAdapter adapter = new SqlDataAdapter(query, conn);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

}

private void Form4\_Load(object sender, EventArgs e)

{

LoadExamData();

}

private void dataGridView1\_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView1.Rows[e.RowIndex];

label1.Text = row.Cells["ExamID"].Value.ToString();

label2.Text = row.Cells["StudentID"].Value.ToString();

label3.Text = row.Cells["StudentName"].Value.ToString();

label4.Text = row.Cells["TeacherID"].Value.ToString();

label5.Text = row.Cells["TeacherName"].Value.ToString();

label6.Text = row.Cells["Subject"].Value.ToString();

label7.Text = row.Cells["ExamDate"].Value.ToString();

label8.Text = row.Cells["Grade"].Value.ToString();

label9.Text = row.Cells["MaxGrade"].Value.ToString();

label10.Text = row.Cells["Comments"].Value?.ToString();

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

Form1 form1 = new Form1();

form1.Show();

}

private void LoadStudentsByGrade(decimal grade)

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

string query = @"

SELECT

e.ID AS ExamID,

s.ID AS StudentID,

s.Name + ' ' + s.Surname + ' ' + s.Family AS StudentName,

t.ID AS TeacherID,

t.FirstName + ' ' + t.LastName AS TeacherName,

e.Subject,

e.ExamDate,

e.Grade,

e.MaxGrade,

e.Comments

FROM Exams e

JOIN Students s ON e.StudentID = s.ID

JOIN Teachers t ON e.TeacherID = t.ID

WHERE e.Grade = @Grade

ORDER BY e.StudentID;";

SqlDataAdapter adapter = new SqlDataAdapter(query, conn);

adapter.SelectCommand.Parameters.AddWithValue("@Grade", grade);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

}

private void btnGradeFilter\_Click(object sender, EventArgs e)

{

decimal grade;

if (decimal.TryParse(txtBoxInputFilter.Text, out grade))

{

LoadStudentsByGrade(grade);

}

else

{

MessageBox.Show("Please enter a valid grade.","Info",MessageBoxButtons.OK,MessageBoxIcon.Information);

}

txtBoxInputFilter.Text = string.Empty;

}

private void LoadTopStudentsBySubject(string subject)

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

string query = @"

SELECT

e.ID AS ExamID,

s.ID AS StudentID,

s.Name + ' ' + s.Surname + ' ' + s.Family AS StudentName,

t.ID AS TeacherID,

t.FirstName + ' ' + t.LastName AS TeacherName,

e.Subject,

e.ExamDate,

e.Grade,

e.MaxGrade,

e.Comments

FROM Exams e

JOIN Students s ON e.StudentID = s.ID

JOIN Teachers t ON e.TeacherID = t.ID

WHERE e.Subject = @Subject

ORDER BY e.Grade DESC;";

SqlDataAdapter adapter = new SqlDataAdapter(query, conn);

adapter.SelectCommand.Parameters.AddWithValue("@Subject", subject);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

}

private void btnSubjectFilter\_Click(object sender, EventArgs e)

{

if (!String.IsNullOrEmpty(txtBoxInputFilter.Text) &&!String.IsNullOrWhiteSpace(txtBoxInputFilter.Text))

{

LoadTopStudentsBySubject(txtBoxInputFilter.Text);

txtBoxInputFilter.Text = string.Empty;

}

else

MessageBox.Show("Please enter a valid Subject.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void LoadStudentAverageGrade(int studentID)

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

string query = @"

SELECT

s.ID AS StudentID,

s.Name + ' ' + s.Surname + ' ' + s.Family AS StudentName,

ROUND(CAST(AVG(e.Grade) AS DECIMAL(10, 2)), 2) AS AverageGrade

FROM Exams e

JOIN Students s ON e.StudentID = s.ID

WHERE s.ID = @StudentID

GROUP BY s.ID, s.Name, s.Surname, s.Family;";

SqlDataAdapter adapter = new SqlDataAdapter(query, conn);

adapter.SelectCommand.Parameters.AddWithValue("@StudentID", studentID);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

}

private void btnStAVGGrade\_Click(object sender, EventArgs e)

{

int studentID;

if (int.TryParse(txtBoxInputFilter.Text, out studentID))

{

LoadStudentAverageGrade(studentID);

}

else

{

MessageBox.Show("Please enter a valid student ID.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void LoadTopTeachers()

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

string query = @"

SELECT

t.ID AS TeacherID,

t.FirstName + ' ' + t.LastName AS TeacherName,

COUNT(e.ID) AS ExamCount

FROM Exams e

JOIN Teachers t ON e.TeacherID = t.ID

GROUP BY t.ID, t.FirstName, t.LastName

ORDER BY ExamCount DESC;";

SqlDataAdapter adapter = new SqlDataAdapter(query, conn);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

}

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

{

LoadTopTeachers();

}

private void btnEdit\_Click(object sender, EventArgs e)

{

Form5 form = new Form5();

form.Show();

}

private void ExportToExcel()

{

using (SaveFileDialog saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "Excel Files|\*.xlsx";

saveFileDialog.Title = "Save as Excel file";

saveFileDialog.FileName = "ExamResults.xlsx";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

using (XLWorkbook workbook = new XLWorkbook())

{

var worksheet = workbook.Worksheets.Add("Exams");

// Заглавия на колоните

for (int col = 0; col < dataGridView1.Columns.Count; col++)

{

worksheet.Cell(1, col + 1).Value = dataGridView1.Columns[col].HeaderText;

worksheet.Cell(1, col + 1).Style.Font.Bold = true; // Подчертаване на заглавията

}

// Данни от DataGridView

for (int row = 0; row < dataGridView1.Rows.Count; row++)

{

for (int col = 0; col < dataGridView1.Columns.Count; col++)

{

worksheet.Cell(row + 2, col + 1).Value = dataGridView1.Rows[row].Cells[col].Value?.ToString();

}

}

// Автоматично нагласяне на ширината на колоните

worksheet.Columns().AdjustToContents();

// Запазване на файла

workbook.SaveAs(saveFileDialog.FileName);

}

MessageBox.Show("Exported to Excel successfully!", "Done", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show("Error exporting to Excel: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

private void btnExportToExcel\_Click(object sender, EventArgs e)

{

ExportToExcel();

}

private void buttonReset\_Click(object sender, EventArgs e)

{

dataGridView1.DataSource = null;

LoadExamData();

}

}

}

Form5

using DocumentFormat.OpenXml.Office.Word;

using DocumentFormat.OpenXml.Wordprocessing;

using School.Utilities;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Xml.Linq;

namespace School

{

public partial class Form5 : Form

{

public Form5()

{

InitializeComponent();

}

string connectionString = @"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True";

void Reset()

{

txtExamId.Text = string.Empty;

txtStId.Text = string.Empty;

txtTId.Text = string.Empty;

txtSubject.Text = string.Empty;

txtExamDate.Text = string.Empty;

txtGarde.Text = string.Empty;

txtMaxGrade.Text = string.Empty;

txtComment.Text = string.Empty;

}

bool ValidateDate()

{

bool isValid = DateTime.TryParse(txtExamDate.Text, out DateTime birthDate);

return isValid;

}

bool IsDecimal(string textboox)

{

return Decimal.TryParse(textboox, out decimal result);

}

bool IsInt(string textboox)

{

return int.TryParse(textboox, out int result);

}

bool IdStudent()

{

if (!IsInt(txtStId.Text))

throw new ArgumentException(Messages.incorrectId);

bool exists = false;

using (SqlConnection cnn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security=True"))

{

cnn.Open();

string query = "SELECT 1 FROM Students WHERE ID = " + int.Parse(txtStId.Text);

using (SqlCommand cmmd = new SqlCommand(query, cnn))

{

var res = cmmd.ExecuteScalar();

exists = res == null;

}

}

return exists;

}

bool IdTeacher()

{

if (!IsInt(txtTId.Text))

throw new ArgumentException(Messages.incorrectId);

bool exists = false;

using (SqlConnection cnn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security=True"))

{

cnn.Open();

string query = "SELECT 1 FROM Teachers WHERE ID = " + int.Parse(txtTId.Text);

using (SqlCommand cmmd = new SqlCommand(query, cnn))

{

var res = cmmd.ExecuteScalar();

exists = res == null;

}

}

return exists;

}

private void btnCreate\_Click(object sender, EventArgs e)

{

try

{

if (IdStudent())

{

throw new ArgumentException(Messages.stExist);

}

if (IdTeacher())

{

throw new ArgumentException(Messages.tExist);

}

if (!ValidateDate())

{

throw new ArgumentException(Messages.InvalidDate);

}

if (!IsDecimal(txtGarde.Text) && !IsDecimal(txtMaxGrade.Text))

{

throw new ArgumentException(Messages.notDecimal);

}

using (SqlConnection conn = new SqlConnection(connectionString))

{

string query = @"INSERT INTO Exams (StudentID, TeacherID, Subject, ExamDate, Grade, MaxGrade, Comments) VALUES (@StudentID, @TeacherID, @Subject, @ExamDate, @Grade, @MaxGrade, @Comments)";

using (SqlCommand cmd = new SqlCommand(query, conn))

{

cmd.Parameters.AddWithValue("@StudentID", int.Parse(txtStId.Text));

cmd.Parameters.AddWithValue("@TeacherID", int.Parse(txtTId.Text));

cmd.Parameters.AddWithValue("@Subject", txtSubject.Text);

cmd.Parameters.AddWithValue("@ExamDate", DateTime.Parse(txtExamDate.Text));

cmd.Parameters.AddWithValue("@Grade", Decimal.Parse(txtGarde.Text));

cmd.Parameters.AddWithValue("@MaxGrade", Decimal.Parse(txtMaxGrade.Text));

cmd.Parameters.AddWithValue("@Comments", txtComment.Text ?? (object)DBNull.Value);

conn.Open();

cmd.ExecuteNonQuery();

}

MessageBox.Show("New exam has been registered!", "Message", MessageBoxButtons.OK);

}

Reset();

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void btnDelete\_Click(object sender, EventArgs e)

{

/\*

using (SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True;"))

{

cn.Open();

string query = "DELETE FROM Exams WHERE ID = @ID";

SqlCommand cmd = new SqlCommand(query, cn);

cmd.Parameters.AddWithValue("@ID", int.Parse(txtExamId.Text));

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Записът е изтрит успешно!", "Съобщение", MessageBoxButtons.OK, MessageBoxIcon.Information);

Reset();

}

else

{

MessageBox.Show("Записът не беше намерен!", "Грешка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

\*/

using (SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security=True;"))

{

cn.Open();

// Delete the exam by ID

string deleteQuery = "DELETE FROM Exams WHERE ID = @ID";

using (SqlCommand cmd = new SqlCommand(deleteQuery, cn))

{

cmd.Parameters.AddWithValue("@ID", int.Parse(txtExamId.Text));

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("The exam record was deleted successfully!", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);

Reset();

// Optional: Reseed IDENTITY to the current MAX(ID)

string reseedQuery = @"

DECLARE @maxId INT = ISNULL((SELECT MAX(ID) FROM Exams), 0);

DBCC CHECKIDENT ('Exams', RESEED, @maxId);";

using (SqlCommand reseedCmd = new SqlCommand(reseedQuery, cn))

{

reseedCmd.ExecuteNonQuery();

}

}

else

{

MessageBox.Show("The record was not found!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

private void btnSave\_Click(object sender, EventArgs e)

{

try

{

if (IdStudent())

{

throw new ArgumentException(Messages.stExist);

}

if (IdTeacher())

{

throw new ArgumentException(Messages.tExist);

}

if (!ValidateDate())

{

throw new ArgumentException(Messages.InvalidDate);

}

if (!IsDecimal(txtGarde.Text) && !IsDecimal(txtMaxGrade.Text))

{

throw new ArgumentException(Messages.notDecimal);

}

using (SqlConnection conn = new SqlConnection(connectionString))

{

string query = @"

UPDATE Exams

SET StudentID = @StudentID, TeacherID = @TeacherID, Subject = @Subject, ExamDate = @ExamDate,

Grade = @Grade, MaxGrade = @MaxGrade, Comments = @Comments

WHERE ID = " + int.Parse(txtExamId.Text);

using (SqlCommand cmd = new SqlCommand(query, conn))

{

cmd.Parameters.AddWithValue("@StudentID", int.Parse(txtStId.Text));

cmd.Parameters.AddWithValue("@TeacherID", int.Parse(txtTId.Text));

cmd.Parameters.AddWithValue("@Subject", txtSubject.Text);

cmd.Parameters.AddWithValue("@ExamDate", DateTime.Parse(txtExamDate.Text));

cmd.Parameters.AddWithValue("@Grade", Decimal.Parse(txtGarde.Text));

cmd.Parameters.AddWithValue("@MaxGrade", Decimal.Parse(txtMaxGrade.Text));

cmd.Parameters.AddWithValue("@Comments", txtComment.Text ?? (object)DBNull.Value);

conn.Open();

cmd.ExecuteNonQuery();

}

MessageBox.Show("Edit successful!", "Message", MessageBoxButtons.OK);

}

Reset();

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void btnSearch\_Click(object sender, EventArgs e)

{

try

{

if (!int.TryParse(txtExamId.Text, out int result))

throw new ArgumentException(Messages.incorrectId);

else

{

bool exists = false;

using (SqlConnection cnn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security=True"))

{

cnn.Open();

string query = "SELECT 1 FROM Exams WHERE ID = " + int.Parse(txtExamId.Text);

using (SqlCommand cmmd = new SqlCommand(query, cnn))

{

var res = cmmd.ExecuteScalar();

exists = res == null;

}

}

if (exists)

{

throw new ArgumentException(Messages.doesntExist);

}

//SqlConnection cn = new SqlConnection(@"Server=IVO03\_HELIOS\SQLEXPRESS;Database=School;Integrated Security = True");

SqlConnection cn = new SqlConnection(@"Server=IVO\_LAPTOP\SQLEXPRESS;Database=School;Integrated Security = True");

string queryString = "SELECT \* FROM Exams WHERE ID = " + int.Parse(txtExamId.Text);

SqlCommand cmd = new SqlCommand(queryString, cn);

cn.Open();

using (SqlDataReader sqlDataReader = cmd.ExecuteReader())

{

while (sqlDataReader.Read())

{

//txtName.Text = sqlDataReader["Name"].ToString();

txtExamId.Text = sqlDataReader["ID"].ToString();

txtStId.Text = sqlDataReader["StudentID"].ToString();

txtTId.Text = sqlDataReader["TeacherID"].ToString();

txtSubject.Text = sqlDataReader["Subject"].ToString();

txtExamDate.Text = sqlDataReader["ExamDate"].ToString();

txtGarde.Text = sqlDataReader["Grade"].ToString();

txtMaxGrade.Text = sqlDataReader["MaxGrade"].ToString();

txtComment.Text = sqlDataReader["Comments"].ToString();

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void btnReset\_Click(object sender, EventArgs e)

{

Reset();

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}