Project name: # ***Student Management System***

Prodan Artem, L5, 55933

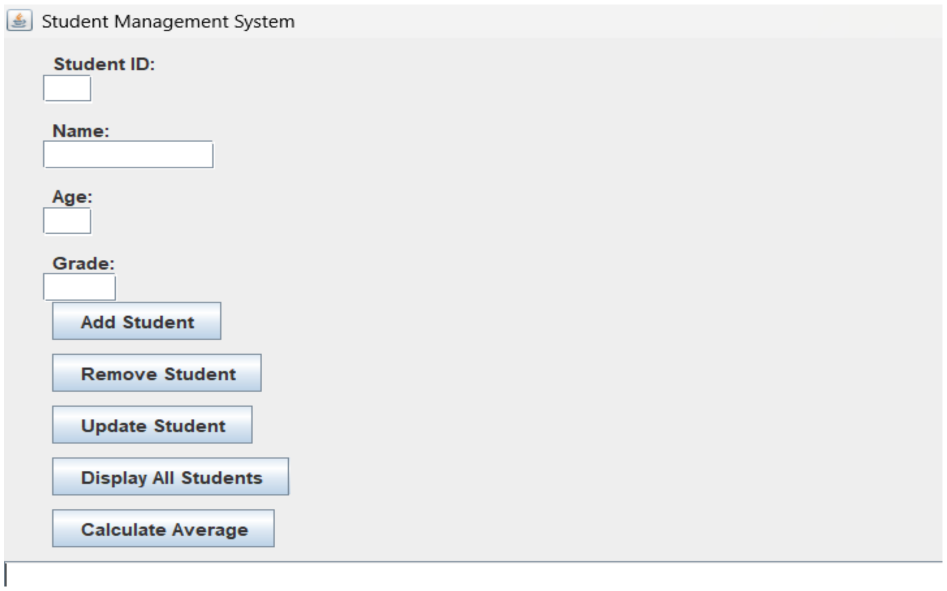
Brief description

Student Management System (SMS) is an application that works thanks to SQL Server and its database, so it can save and remember any incoming data. This system is designed to store and perform some simple operations with student data that the user would like to add.

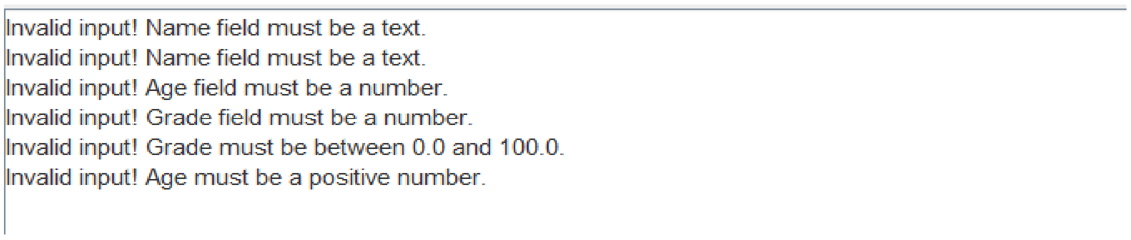
Functionality

- *“SMS” window contains:*

  - 4 panels for input data from the user;
  - 5 buttons for interaction with database through the application;
  - Output panel, where user can get feedback from the system after each action;



- *Input panels:*
  - Student ID (During operations user can enter both string and integer values in this field)
  - Name (During operations user can provide only string values in this field. If numbers are entered, output panel will show error message)
  - Age (During operations user can provide only integer values in this field. If text is entered, the output panel will show an error message. If negative number is entered, the output panel will also show an error message)
  - Grade (During operations user can provide double values (like 20.5 or 20) in this field. If text is entered, the output panel will show an error message. If entered number is out of range of 0 and 100.0, the output panel will also show an error message)

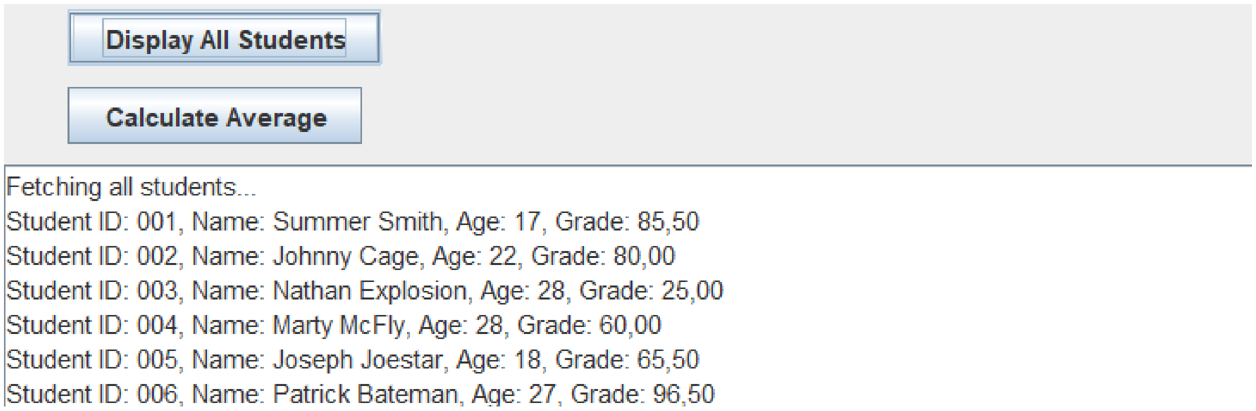


```
Invalid input! Name field must be a text.
Invalid input! Name field must be a text.
Invalid input! Age field must be a number.
Invalid input! Grade field must be a number.
Invalid input! Grade must be between 0.0 and 100.0.
Invalid input! Age must be a positive number.
```

- *Buttons:*

  - *Display all students*

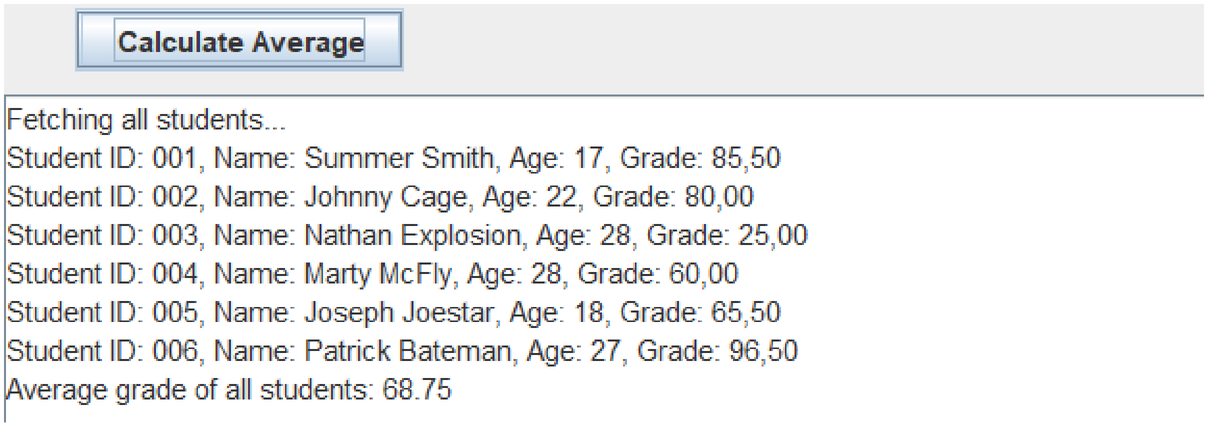
displays all current data from the database



```
Fetching all students...
Student ID: 001, Name: Summer Smith, Age: 17, Grade: 85,50
Student ID: 002, Name: Johnny Cage, Age: 22, Grade: 80,00
Student ID: 003, Name: Nathan Explosion, Age: 28, Grade: 25,00
Student ID: 004, Name: Marty McFly, Age: 28, Grade: 60,00
Student ID: 005, Name: Joseph Joestar, Age: 18, Grade: 65,50
Student ID: 006, Name: Patrick Bateman, Age: 27, Grade: 96,50
```

  - *Calculate average*

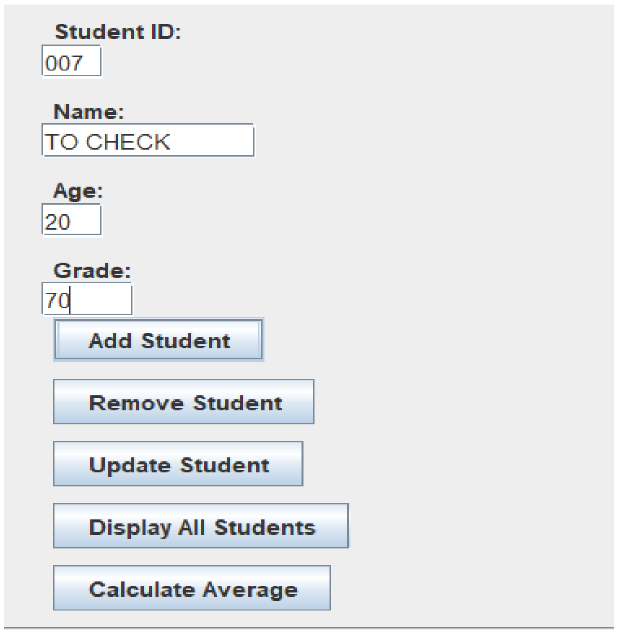
calculates average grade of all students



```
Fetching all students...
Student ID: 001, Name: Summer Smith, Age: 17, Grade: 85,50
Student ID: 002, Name: Johnny Cage, Age: 22, Grade: 80,00
Student ID: 003, Name: Nathan Explosion, Age: 28, Grade: 25,00
Student ID: 004, Name: Marty McFly, Age: 28, Grade: 60,00
Student ID: 005, Name: Joseph Joestar, Age: 18, Grade: 65,50
Student ID: 006, Name: Patrick Bateman, Age: 27, Grade: 96,50
Average grade of all students: 68.75
```

- *Add student*

Adds new student to the database.



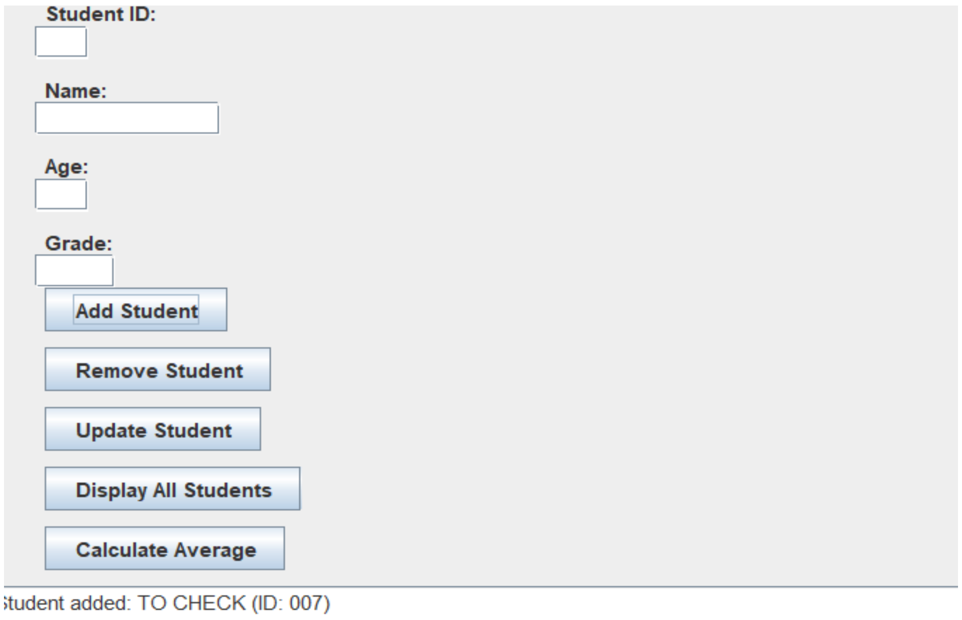
Student ID: 007

Name: TO CHECK

Age: 20

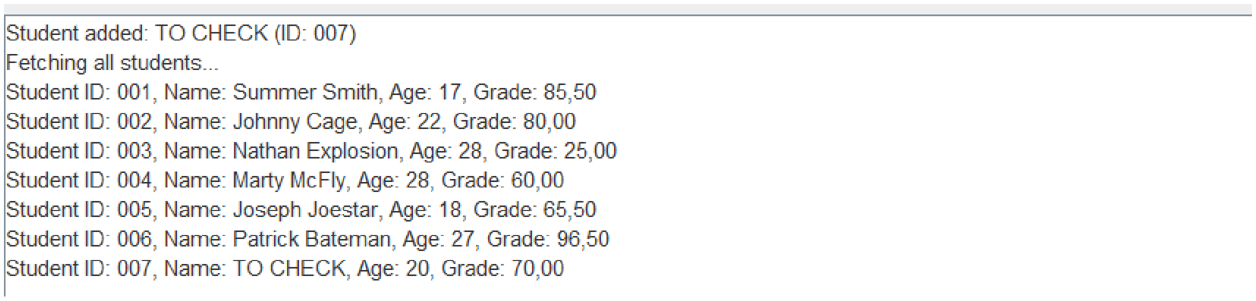
Grade: 70

(Input panels after operation will be cleared)



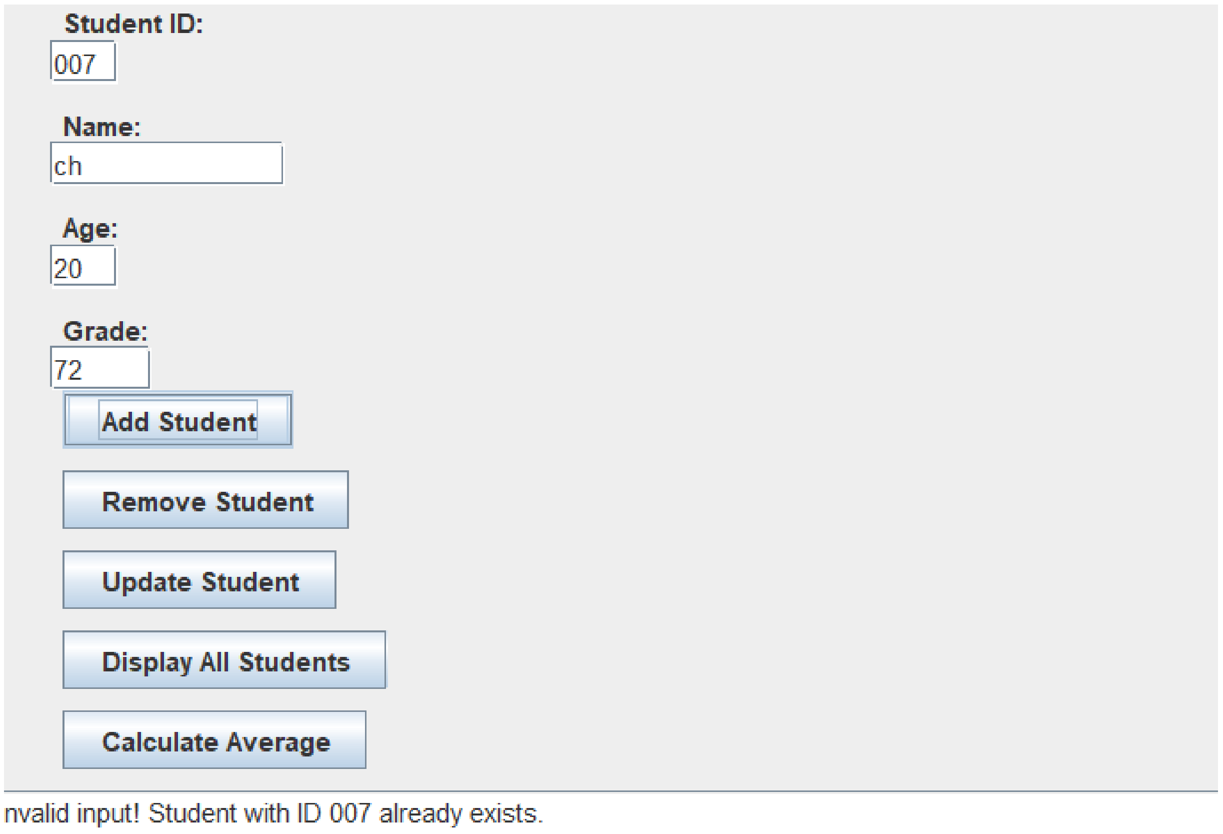
;tudent added: TO CHECK (ID: 007)

After running “Display all students” you can see new student in the output panel right away after the addition.



```
Student added: TO CHECK (ID: 007)
Fetching all students...
Student ID: 001, Name: Summer Smith, Age: 17, Grade: 85,50
Student ID: 002, Name: Johnny Cage, Age: 22, Grade: 80,00
Student ID: 003, Name: Nathan Explosion, Age: 28, Grade: 25,00
Student ID: 004, Name: Marty McFly, Age: 28, Grade: 60,00
Student ID: 005, Name: Joseph Joestar, Age: 18, Grade: 65,50
Student ID: 006, Name: Patrick Bateman, Age: 27, Grade: 96,50
Student ID: 007, Name: TO CHECK, Age: 20, Grade: 70,00
```

When you try to add to the database student with ID that already exist, output panel will send an error message.



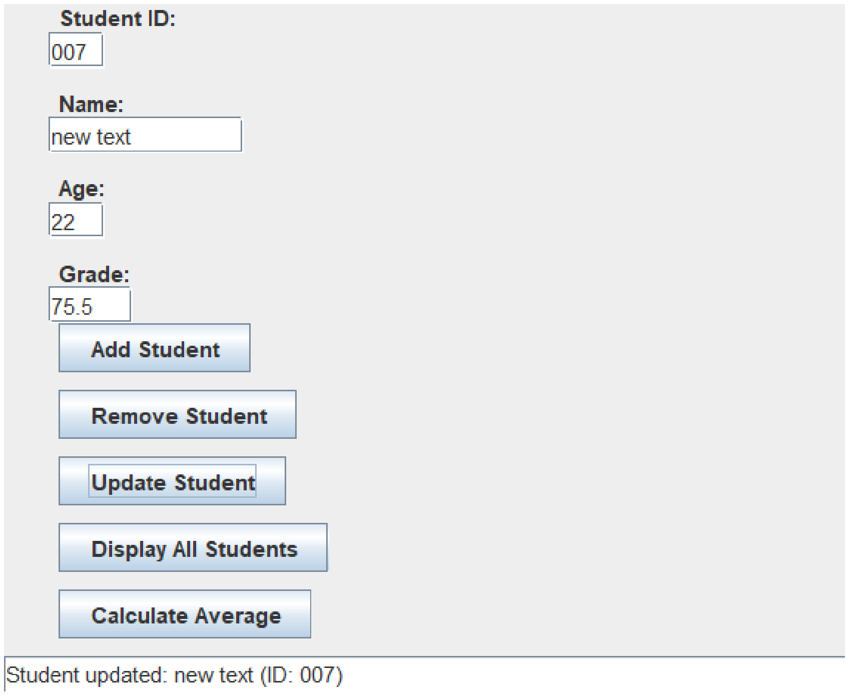
Student ID:	007
Name:	ch
Age:	20
Grade:	72

```
nvalid input! Student with ID 007 already exists.
```

- *Update student*

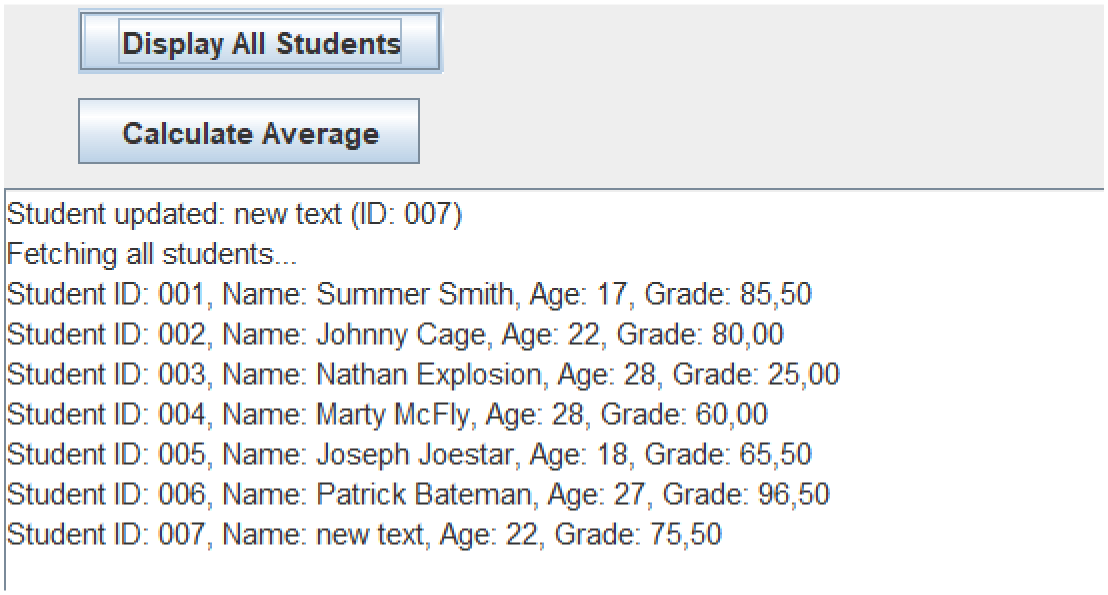
Updates the information about existing students.

(all the input fields must be filled for this operation)



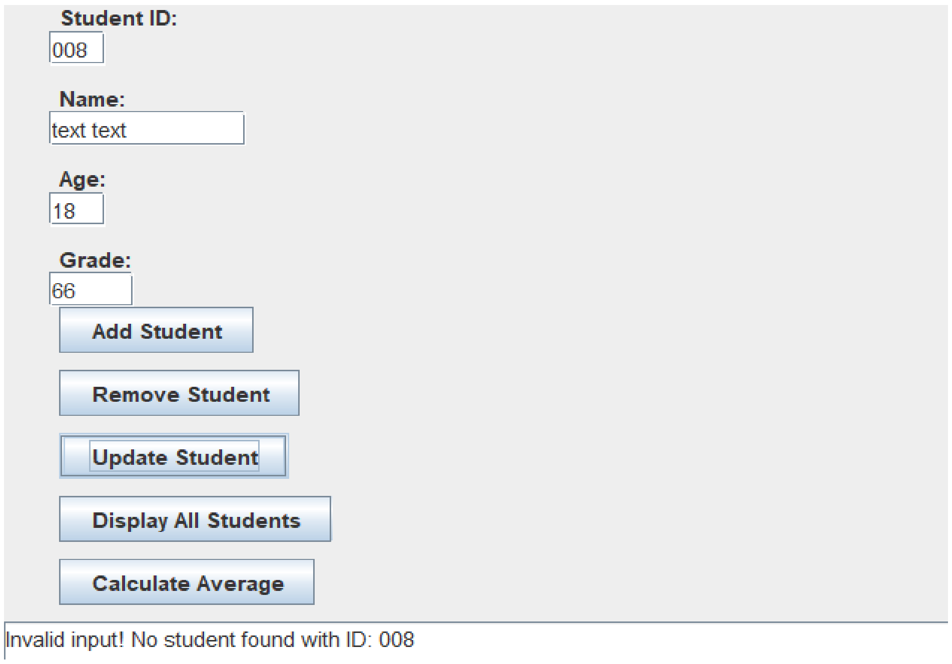
```
Student updated: new text (ID: 007)
```

After running “Display all students” you can see updated student in the output panel right away after the updating.



```
Student updated: new text (ID: 007)
Fetching all students...
Student ID: 001, Name: Summer Smith, Age: 17, Grade: 85,50
Student ID: 002, Name: Johnny Cage, Age: 22, Grade: 80,00
Student ID: 003, Name: Nathan Explosion, Age: 28, Grade: 25,00
Student ID: 004, Name: Marty McFly, Age: 28, Grade: 60,00
Student ID: 005, Name: Joseph Joestar, Age: 18, Grade: 65,50
Student ID: 006, Name: Patrick Bateman, Age: 27, Grade: 96,50
Student ID: 007, Name: new text, Age: 22, Grade: 75,50
```

When you try to update a student with ID that does not exist in the database, the output panel will send an error message.

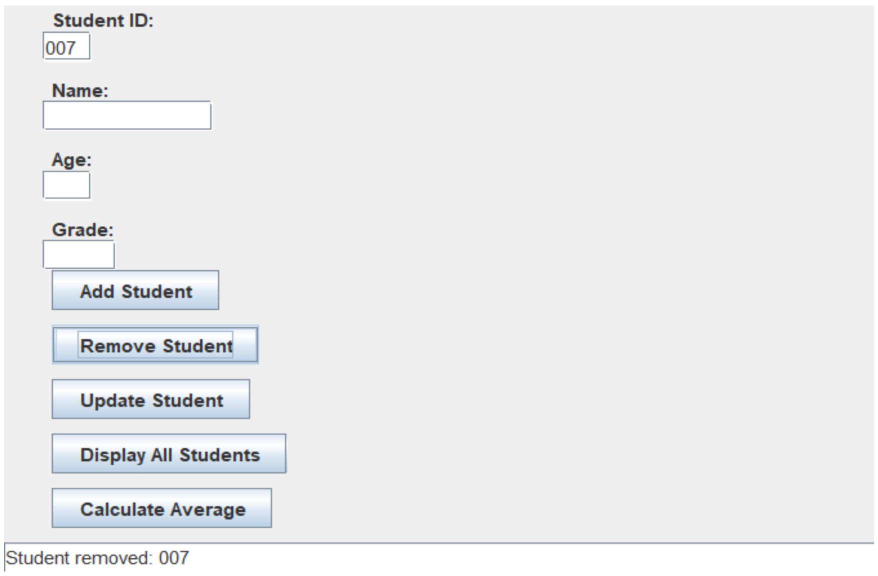


```
Invalid input! No student found with ID: 008
```

- *Remove student*

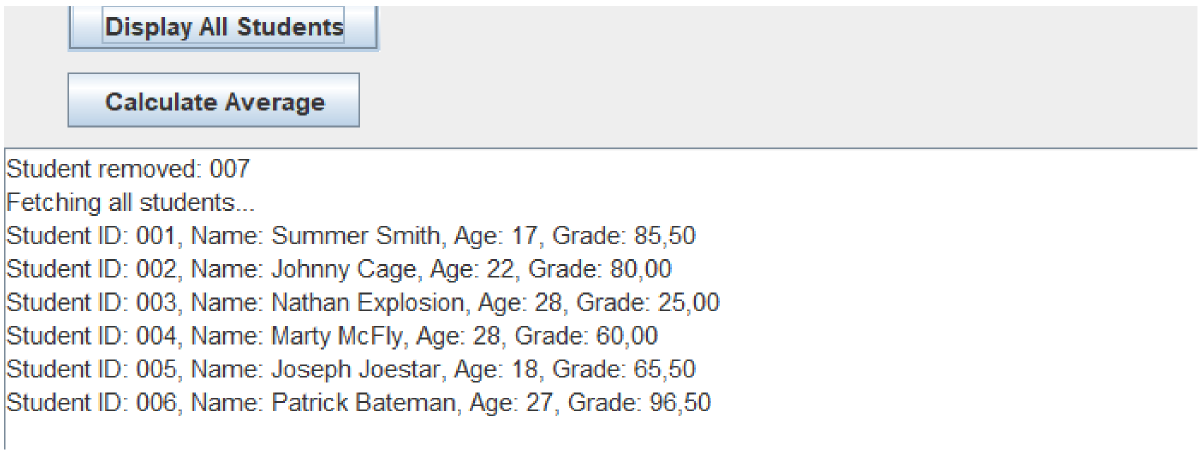
Removes the student from the database, using its ID.

(Other input fields can be left empty for this operation)



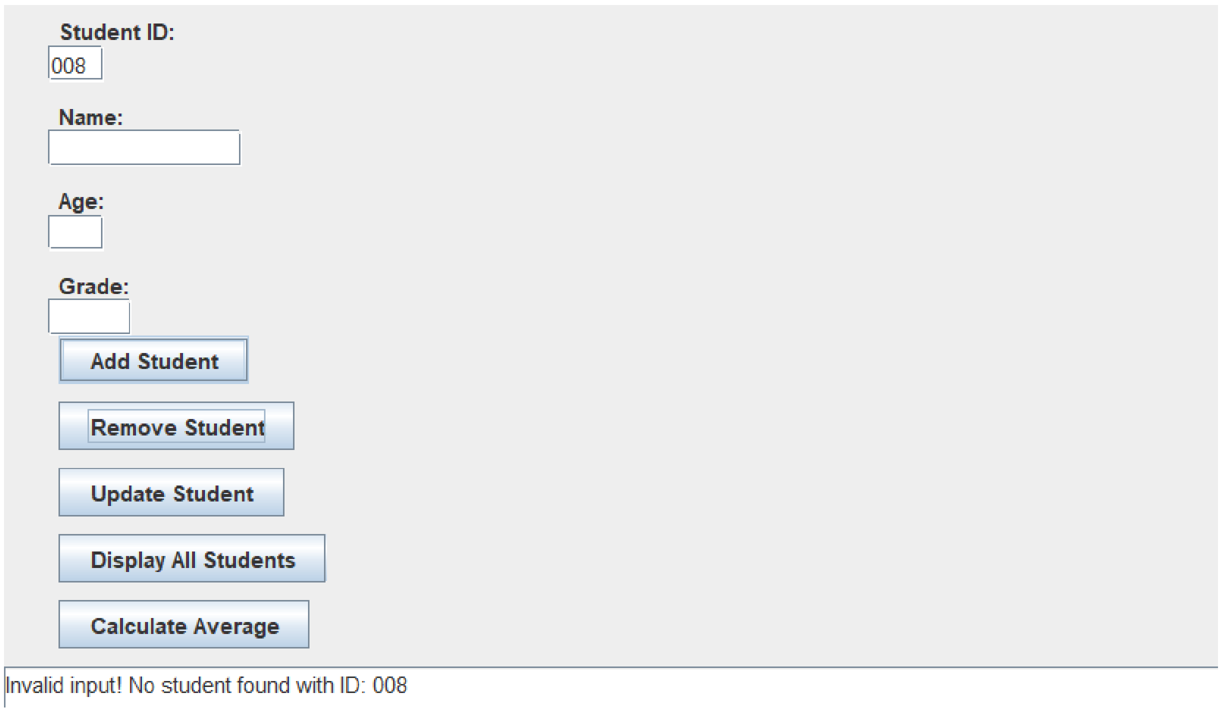
```
Student removed: 007
```

After running “Display all students” you can see in the output panel that student was removed from the database.



```
Student removed: 007
Fetching all students...
Student ID: 001, Name: Summer Smith, Age: 17, Grade: 85,50
Student ID: 002, Name: Johnny Cage, Age: 22, Grade: 80,00
Student ID: 003, Name: Nathan Explosion, Age: 28, Grade: 25,00
Student ID: 004, Name: Marty McFly, Age: 28, Grade: 60,00
Student ID: 005, Name: Joseph Joestar, Age: 18, Grade: 65,50
Student ID: 006, Name: Patrick Bateman, Age: 27, Grade: 96,50
```

When you try to remove a student with ID that does not exist in the database, the output panel will send an error message.



Student ID: 008

Name:

Age:

Grade:

```
Invalid input! No student found with ID: 008
```

Code organization

Whole java project consists of:

1. Main class;
2. Student class;
3. StudentManager interface;
4. StudentManagerImpl class;
5. ConnectionDB class;
6. StudentManagementGUI class;

- ***Main class***

I used this class for various verifications such as checking connection with database or already created methods. I do not use this class for running the application, StudentManagementGUI class does this.

- ***Student class***

This class serves as a data model representing a student in the system. It contains four attributes (`ID, Name, Age, grade).`

Functionality:

- **Constructor** (Initializes a student object with given values)
- **Getters & Setters** (Provide controlled access to the student) attributes.
- **`toString()` Method** (Formats student details into a readable string for display)

- ***StudentManager interface***

This interface lists methods that StudentManagerImpl class must follow.

- ***StudentManagerImpl class***

This class is responsible for implementing methods from “StudentManager interface” and is responsible for managing student data in a database (using SQL). It interacts with the database to perform CRUD operations on student records (Create, Read, Update, Delete).

- ***ConnectionDB class***

This class is responsible for managing database connection and resource handling for SQL operations. It provides methods for establishing, closing, and testing the connection to a Microsoft SQL Server database.

Before creating this class, I created “StudentManagementDB”database in SQL manually, to be sure that I can work with it.

- ***StudentManagementGUI class***

This class is a graphical user interface (GUI) for managing student records in a student management system. It is built using the “Java Swing” library and provides functionality for adding, removing, updating, displaying, and calculating the average grade of students. The GUI interacts with the “`StudentManagerImpl`” class to perform CRUD operations on the student database. This class runs the application.

How to compile and run:

Requirements:

- Java Development Kit (JDK) (I used jdk-23 for windows)
- SQL Server database (I used Microsoft SQL management studio)
- JDBC for database connectivity (I used mssql-jdbc-12.8.1.jre11.jar)

1 step) Database setup

1. Connect to the SQL server “DESKTOP-35EOBF9\MSSQLSERVER01”

(use Login: “sa”, and password: “Ar27-9c110rRr”)

2. Create new Query with content:

```
IF NOT EXISTS (SELECT * FROM sys.databases WHERE name = 'StudentManagementDB')
BEGIN
   CREATE DATABASE StudentManagementDB;
END
```

```
USE StudentManagementDB;

DROP TABLE IF EXISTS students;

CREATE TABLE students (

   name VARCHAR(100), --VARCHAR is for TEXT

   age INT,

   grade REAL,

   studentID VARCHAR(50) PRIMARY KEY

);

USE StudentManagementDB;

ALTER AUTHORIZATION ON DATABASE::StudentManagementDB TO sa;
```

2 step) Find directory with files on your computer

Attached “SMSproject.zip” file contains folder “SourceCode” with such files inside:

“ConnectionDB.java,

Main.java,

Student.java,

StudentManagementGUI.java,

StudentManager.java,

StudentManagerImpl.java”

Open a **terminal/command prompt** and find the directory containing these files on your computer.

3 step) To compile

Use command:

```
javac -cp .;Your\Directory\To\mssql-jdbc-12.8.1.jre11.jar Main.java
Student.java StudentManager.java StudentManagerImpl.java
ConnectionDB.java StudentManagementGUI.java
```

To compile all files to ".class" format.

4 step) Now you can run the application.

Use command:

```
java -cp .;Your\Directory\To\mssql-jdbc-12.8.1.jre11.jar
StudentManagementGUI
```

5 step (Optional)

If you want to run application with already existing sample data, you may use this SQL query in Microsoft SSMS:

```
USE StudentManagementDB;

INSERT INTO students (name, age, grade, studentID)

VALUES

    ('Summer Smith', 17, 85.5, '001'),

    ('Johnny Cage', 22, 80.0, '002'),
    ('Nathan Explosion', 28, 25.0, '003'),

    ('Marty McFly', 28, 60.0, '004'),

    ('Joseph Joestar', 18, 65.5, '005'),

    ('Patrick Bateman', 27, 96.5, '006');
```