VISUAL PROGRAMMING PROJECT DOCUMENTATION STAGES 3

Project Name : Student Information System

In this project, we implemented significant SQL integrations and functionality improvements. By connecting tables, interfaces, and data management, we built a functional system for both teachers and students.

1. Tables and Their Structures

We created **6 tables** in the SQLite database:

1. **students**: A table to store student information.
   - **Columns**: id (PRIMARY KEY), name, department
   - **Usage**: Used for adding, deleting, and validating students.
2. **grades**: A table to store student grade information.
   - **Columns**: id, student_id, course_id, grade
   - **Usage**: Used for adding student grades, viewing grades, and calculating GPA.
3. **courses**: A table to store course information.
   - **Columns**: id, course_name, credits
   - **Usage**: Used to retrieve course names and credits when adding grades.
4. **messages**: A table to store messages sent to students.
   - **Columns**: id, student_id, message, sender
   - **Usage**: Used for teachers to send messages to students and for students to view their own messages.
5. **attendance**: A table to store attendance records.
   - **Columns**: id, student_id, date, status
   - **Usage**: Used for adding and reporting student attendance.
6. **teacher**: A table to store teacher login credentials.
   - **Columns**: id, username, password
   - **Usage**: Used for teacher authentication on the login screen.

busra | Table name: attendance | WITHOUT ROWID | STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER								NULL
2	student_id	INTEGER								NULL
3	date	TEXT								NULL

busra | Table name: courses | WITHOUT ROWID | STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER								NULL
2	course_name	TEXT								NULL
3	instructor	TEXT								NULL
4	credits	INTEGER								NULL

busra Table name: teacher WITHOUT ROWID STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER								NULL
2	name	TEXT								NULL
3	surname	TEXT								NULL
4	password	TEXT								NULL

busra Table name: students WITHOUT ROWID STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER								NULL
2	name	TEXT								NULL
3	surname	TEXT								NULL
4	class	TEXT								NULL
5	password	TEXT								NULL

busra Table name: grades WITHOUT ROWID STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER								NULL
2	student_id	INTEGER								NULL
3	course	TEXT								NULL
4	grade	INTEGER								NULL

busra Table name: messages WITHOUT ROWID STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER								NULL
2	student_id	INTEGER								NULL
3	message	TEXT								NULL
4	sender	TEXT								Teacher
5	timestamp	DATETIME								CURRENT_TIMESTAMP

2. User Data and Handling Invalid Data

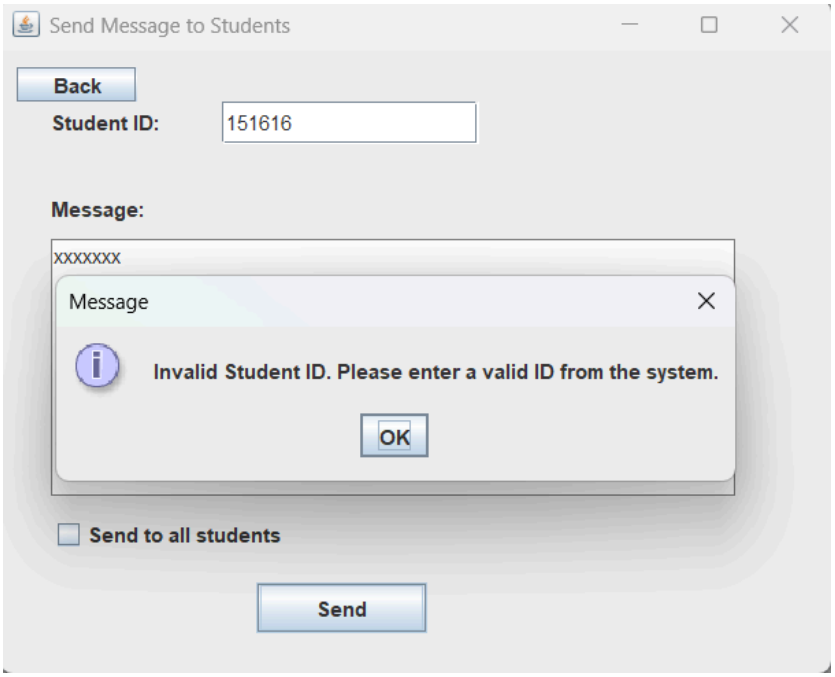
We initialized the SQLite database with 10 default students and 1 teacher, each assigned login credentials to access the system. On the login screen, users can log in using these predefined credentials. Each page dynamically displays information specific to the logged-in student by retrieving data from the database.

id	name	surname	class	password
1	Ayşe	Yılmaz	9A	123456
2	Ali	Can	10B	abcdef
3	Fatma	Demir	11C	qwerty
4	Mehmet	Kaya	12A	zxcvbn
5	Zeynep	Şahin	9B	asdfgh
6	Mustafa	Çelik	10A	jklşii
7	Elif	Öztürk	11B	mnbvyu
8	Emre	Yıldız	12B	trewqa
9	Selin	Arslan	9C	öçğüiş
10	Oğuz	Doğan	10C	hjklşi

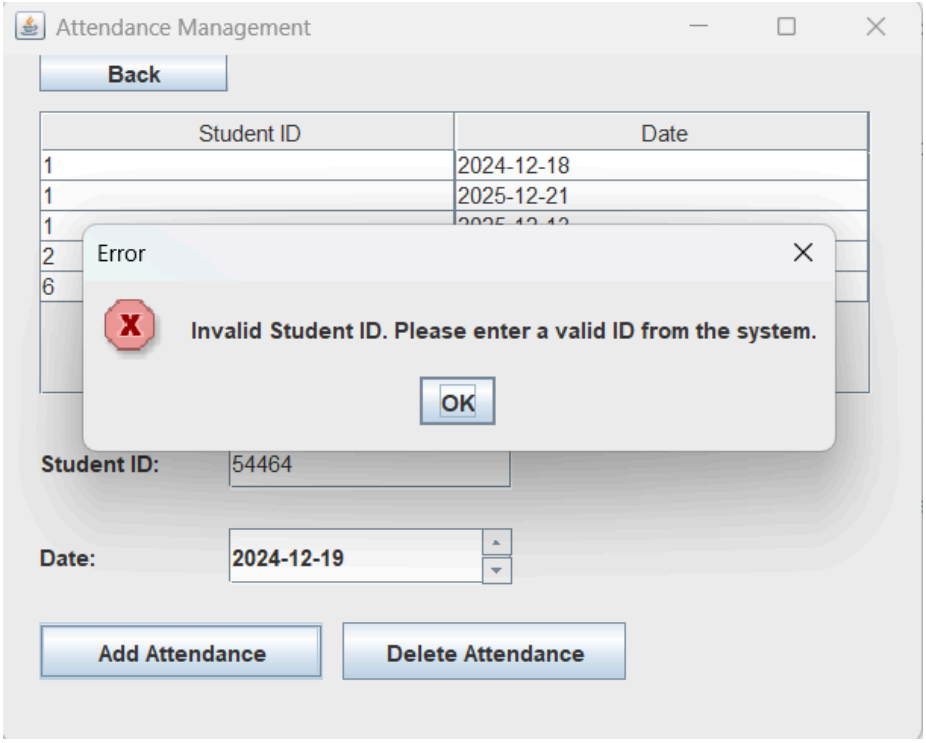
id	name	surname	password
1	Teacher	1	1234

If any invalid student information or mismatched credentials are entered, the system successfully throws an error or displays a validation message. Invalid inputs and database inconsistencies are effectively handled, ensuring the system remains robust and user-friendly.

For example, when a teacher wants to send a message to a student, if an invalid student ID (one that doesn't exist in the database) is entered, the system will display an error message and work successfully.

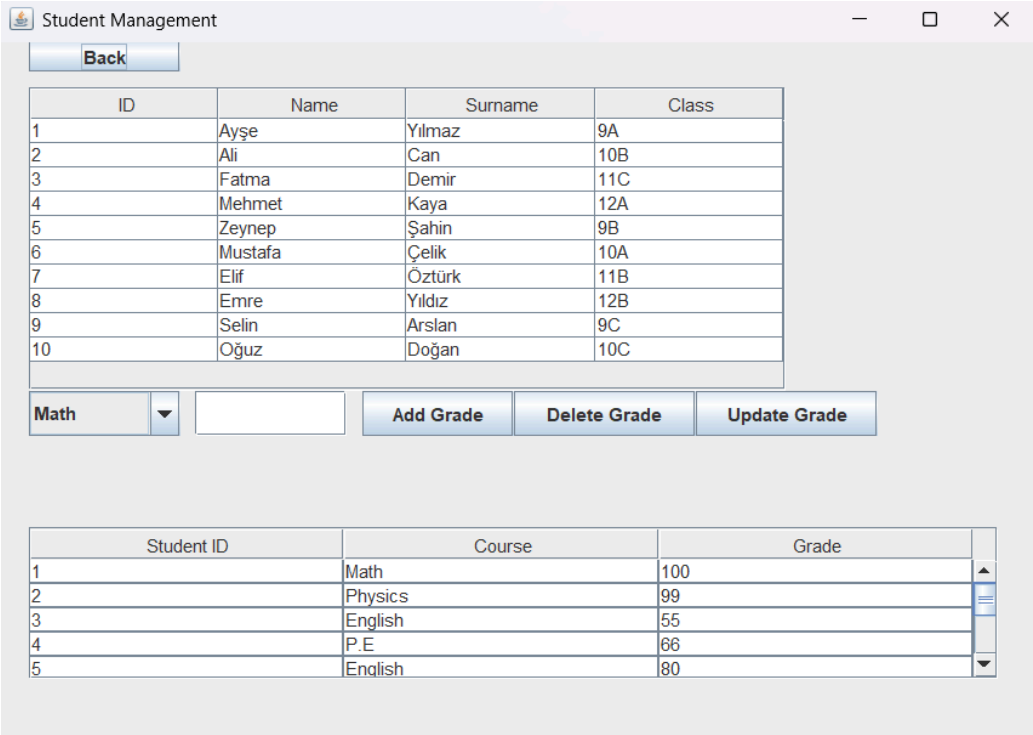


The same way, when the teacher enters attendance into the system, it will give an error if a student ID that does not exist in the database is entered.



3. Collecting Data

In the student management screen, the student table from our database is displayed in front of us. When we select each student and add a course and grade, it is added both in the table below and to the grades table in the database, and then saved.



Structure | Data | Constraints | Indexes | Triggers | DDL

Grid view | Form view

	id	student_id	course	grade
1	1	1	Math	100
2	2	2	Physics	99
3	3	3	English	55
4	4	4	P.E	66
5	6	5	English	80
6	11	6	Chemistry	88
7	12	7	P.E	78
8	13	8	Biology	47
9	14	9	Math	69
10	15	10	Physics	64

This is my **add, delete and update** codes for database:

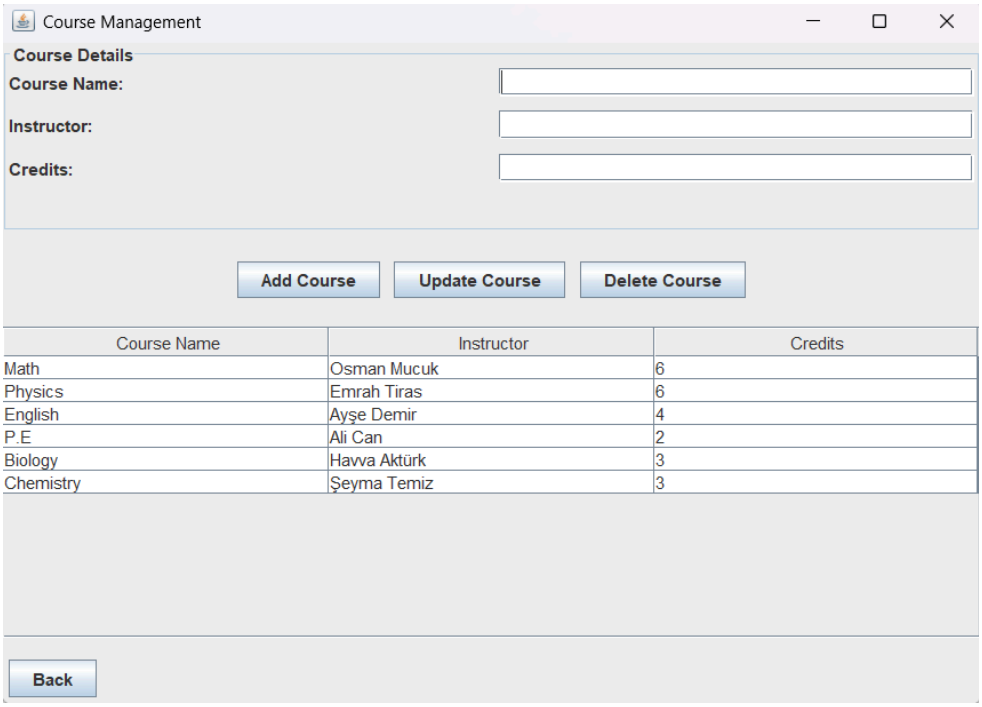
```
    try (Connection conn = DatabaseHelper.connect()) {
        String sql = "INSERT INTO grades (student_id, course, grade) VALUES (?, ?, ?)";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, studentID);
        pstmt.setString(2, course);
        pstmt.setInt(3, grade);

        int rowsInserted = pstmt.executeUpdate();
        if (rowsInserted > 0) {
            JOptionPane.showMessageDialog(null, "Grade added successfully!");
            loadGradeTable(gradeModel); // Notları güncelle
        }
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Error adding grade: " + ex.getMessage(), "Error"
    }
});
```

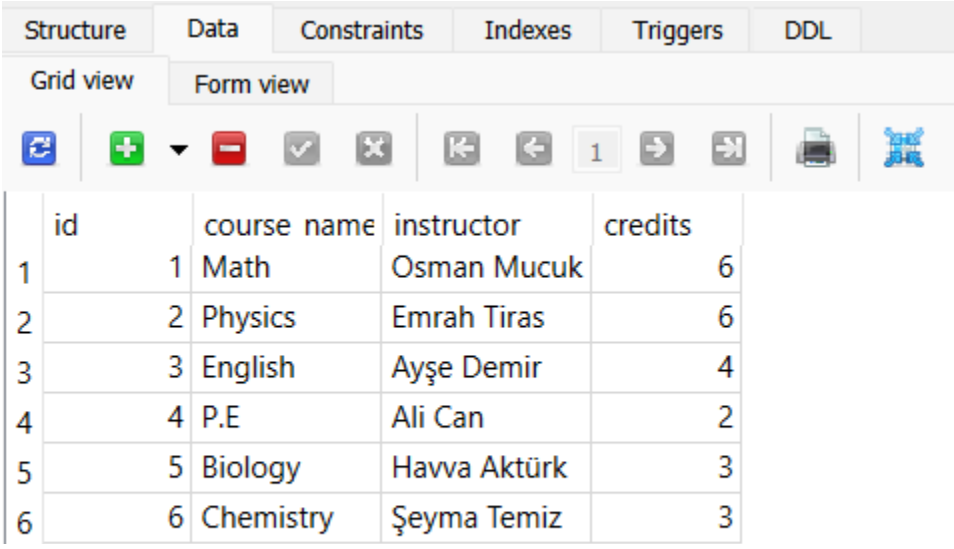
```java
    try (Connection conn = DatabaseHelper.connect()) {
        String sql = "DELETE FROM grades WHERE student_id = ? AND course = ?";
        var pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, studentID);
        pstmt.setString(2, course);

        int rowsDeleted = pstmt.executeUpdate();
        if (rowsDeleted > 0) {
            gradeModel.removeRow(selectedRow);
            JOptionPane.showMessageDialog(null, "Grade deleted successfully.");
        }
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, "Error deleting grade: " + ex.getMessa
    }
}
```

```java
    try (Connection conn = DatabaseHelper.connect()) {
        String sql = "UPDATE grades SET grade = ? WHERE student_id = ? AND course = ?";
        var pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, grade);
        pstmt.setString(2, studentID);
        pstmt.setString(3, course);

        int rowsUpdated = pstmt.executeUpdate();
        if (rowsUpdated > 0) {
            gradeModel.setValueAt(grade, selectedRow, 2);
            JOptionPane.showMessageDialog(null, "Grade updated successfully.");
            gradeField.setText("");
        }

    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, "Error updating grade: " + ex.getMessage(), "Erro
    }
});
```

In the course management screen, after selecting the course name, the instructor's name, and the course credits, when we click the "add" button, it is saved both in the table below and in the course table of our database.

At the same time, the other buttons are also functional and perform delete and update operations in the database.





	id	course_name	instructor	credits
1	1	Math	Osman Mucuk	6
2	2	Physics	Emrah Tiras	6
3	3	English	Ayşe Demir	4
4	4	P.E	Ali Can	2
5	5	Biology	Havva Aktürk	3
6	6	Chemistry	Şeyma Temiz	3

```java
private void addCourseToDatabase(String courseName, String instructor, int credits) {
    String sql = "INSERT INTO courses (course_name, instructor, credits) VALUES (?, ?, ?)";
    try (Connection conn = DatabaseHelper.connect(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, courseName);
        pstmt.setString(2, instructor);
        pstmt.setInt(3, credits);
        pstmt.executeUpdate();
        System.out.println("Course added to database.");
    } catch (SQLException e) {
        System.out.println("Error inserting course: " + e.getMessage());
    }
}

private void deleteCourseFromDatabase(String courseName) {
    String sql = "DELETE FROM courses WHERE course_name = ?";
    try (Connection conn = DatabaseHelper.connect(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, courseName);
        pstmt.executeUpdate();
        System.out.println("Course deleted from database.");
    } catch (SQLException e) {
        System.out.println("Error deleting course: " + e.getMessage());
    }
}
```

```java
private void updateCourseInDatabase(String oldCourseName, String newCourseName, String instructor, int credits) {
    String sql = "UPDATE courses SET course_name = ?, instructor = ?, credits = ? WHERE course_name = ?";
    try (Connection conn = DatabaseHelper.connect(); PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, newCourseName);
        pstmt.setString(2, instructor);
        pstmt.setInt(3, credits);
        pstmt.setString(4, oldCourseName);
        pstmt.executeUpdate();
        System.out.println("Course updated in database.");
    } catch (SQLException e) {
        System.out.println("Error updating course: " + e.getMessage());
    }
}
private void loadCoursesFromDatabase() {
String sql = "SELECT course_name, instructor, credits FROM courses";
try (Connection conn = DatabaseHelper.connect();
     Statement stmt = conn.createStatement();
     ResultSet rs = stmt.executeQuery(sql)) {
    while (rs.next()) {
        tableModel.addRow(new Object[]{
                rs.getString("course_name"),
                rs.getString("instructor"),
                rs.getInt("credits")
        });
```

In the teacher message screen, every time a message is sent, it is stored in the database in the message section along with the ID of the student to whom the message was sent.

```java
String sql = "INSERT INTO messages (student_id, message, sender) VALUES (?, ?, ?)";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setInt(1, Integer.parseInt(studentId));
pstmt.setString(2, message);
pstmt.setString(3, "Teacher");
pstmt.executeUpdate();
```

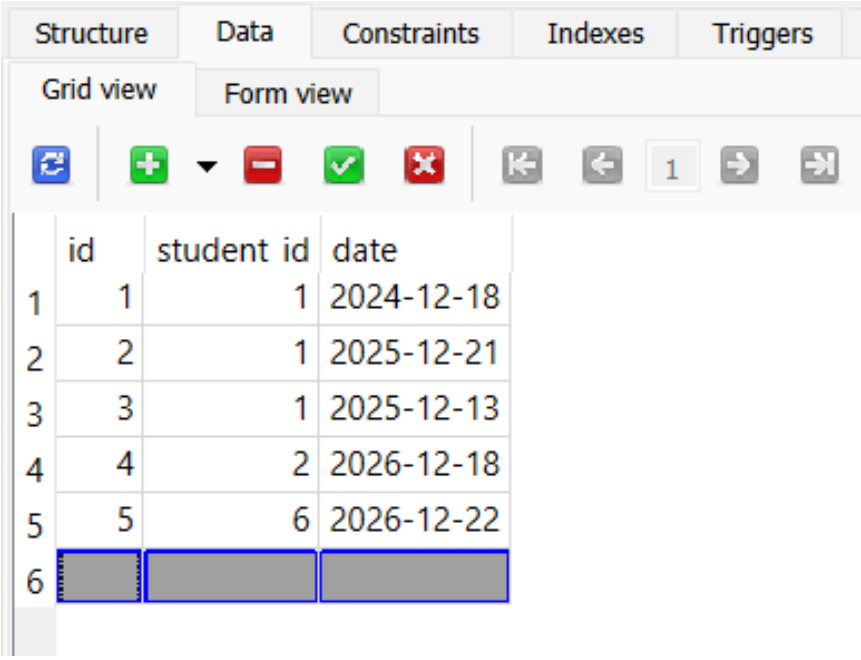
Structure | Data | Constraints | Indexes | Triggers | DDL

Grid view | Form view

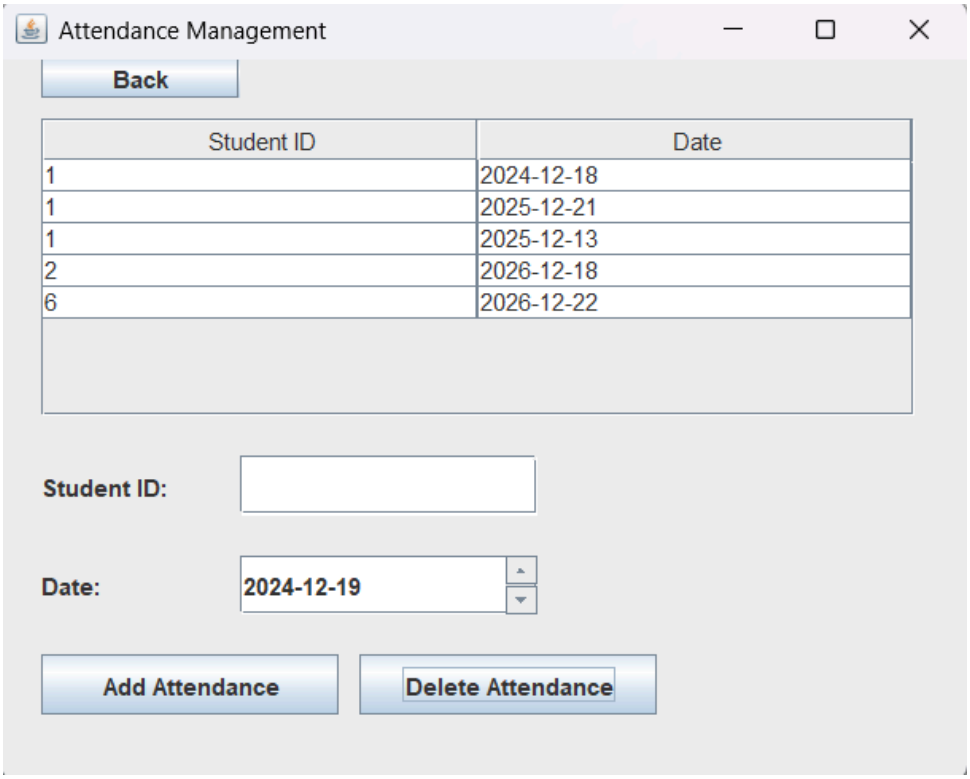
Filter data — Total rows loaded: 13

	id	student_id	message	sender	timestamp
1	1	1	Our class this week has been cancelled.	Teacher	2024-12-18 17:25:44
2	2	7815145	fgfdagadfdgad	Teacher	2024-12-18 17:32:47
3	3	1	hello	Teacher	2024-12-18 17:43:55
4	4	2	hello	Teacher	2024-12-18 17:43:55
5	5	3	hello	Teacher	2024-12-18 17:43:55
6	6	4	hello	Teacher	2024-12-18 17:43:55
7	7	5	hello	Teacher	2024-12-18 17:43:55
8	8	6	hello	Teacher	2024-12-18 17:43:55
9	9	7	hello	Teacher	2024-12-18 17:43:55
10	10	8	hello	Teacher	2024-12-18 17:43:55
11	11	9	hello	Teacher	2024-12-18 17:43:55
12	12	10	hello	Teacher	2024-12-18 17:43:55
13	13	2	The project assignment submission deadline has been extended by 1 week.	Teacher	2024-12-18 18:13:43

In the attendance management section, each attendance record is saved both in the table and in the database.



	id	student_id	date
1	1	1	2024-12-18
2	2	1	2025-12-21
3	3	1	2025-12-13
4	4	2	2026-12-18
5	5	6	2026-12-22
6			



In the reporting section, both student and grades information are retrieved from SQLite, and a comprehensive report table is generated.

Teacher Report

Back

Student ID	Name	Surname	Class	Course	Grade
1	Ayşe	Yılmaz	9A	Math	100
2	Ali	Can	10B	Physics	99
3	Fatma	Demir	11C	English	55
4	Mehmet	Kaya	12A	P.E	66
5	Zeynep	Şahin	9B	English	80
6	Mustafa	Çelik	10A	Chemistry	88
7	Elif	Öztürk	11B	P.E	78
8	Emre	Yıldız	12B	Biology	47
9	Selin	Arslan	9C	Math	69
10	Oğuz	Doğan	10C	Physics	64

In the student section, when each student logs in with their own information, they can view their grades in the course and grades section. This information is retrieved from the database.

Student Report Screen

Back

Course Name	Credits	Grade	Pass/Fail
Math	6	100	Pass
Chemistry	3	98	Pass
Biology	3	74	Pass
Physics	6	54	Pass
English	4	75	Pass

In the transcript section, the grades for all the courses taken and the corresponding course credits are processed to calculate the GPA.

Transcript

Back

Student Information

Student Info

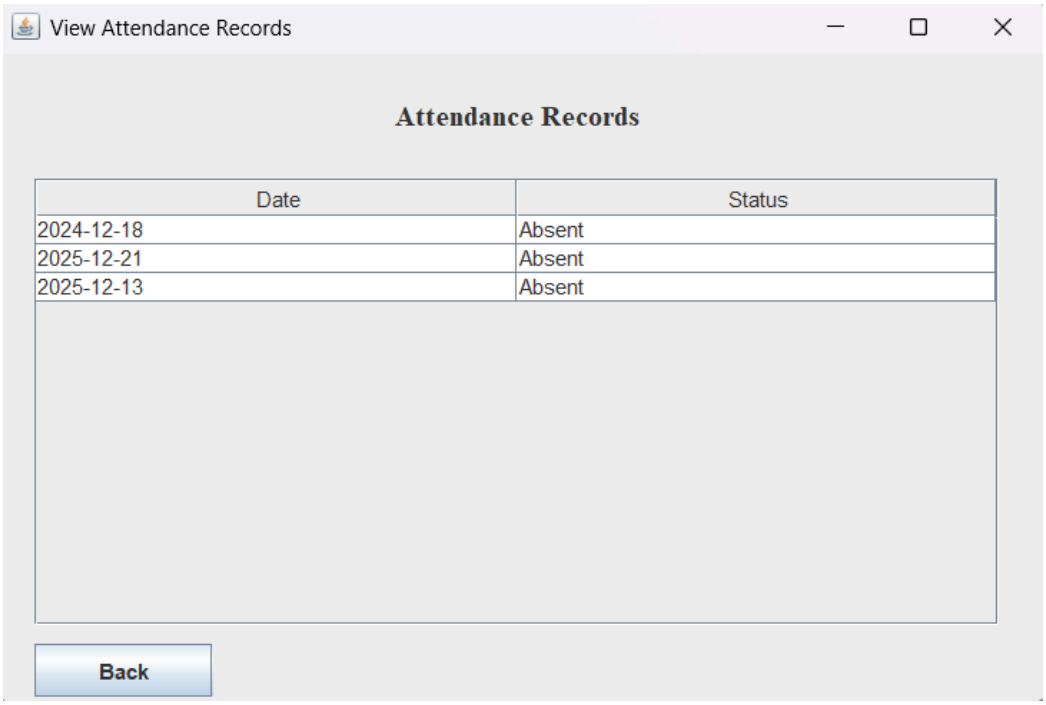
Student ID: 1

Department: Placeholder Department

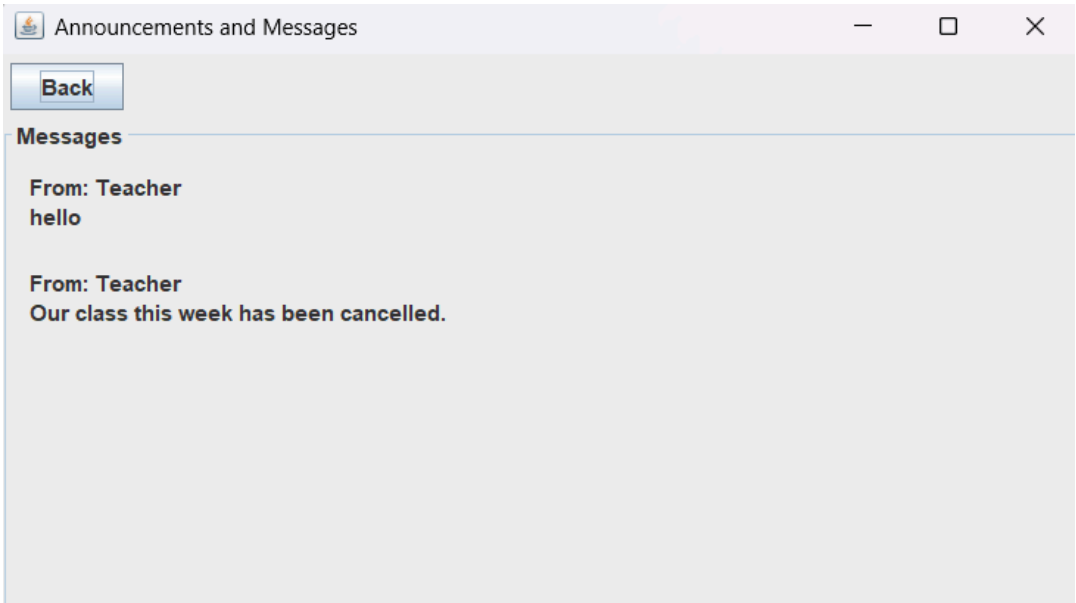
Course Na...	Credits	Grade	Total Points	Pass/Fail
Math	6	100	600	Pass
Chemistry	3	98	294	Pass
Biology	3	74	222	Pass
Physics	6	54	324	Pass
English	4	75	300	Pass

GPA: 79.09

In the attendance section, if a student was absent on a specific date, that information is retrieved from the attendance table in the database and displayed in the table.



In the messages and announcements section, the messages sent by the teacher to the student are displayed. This information is retrieved from the message table in the database.



Now I will show some of my SQL code.

busra

Query History

```sql
CREATE TABLE IF NOT EXISTS students (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    surname TEXT NOT NULL,
    class TEXT,
    password TEXT NOT NULL
);
INSERT INTO students (name, surname, class, password) VALUES ('Ayşe', 'Yılmaz', '9A', '123456');
INSERT INTO students (name, surname, class, password) VALUES ('Ali', 'Can', '10B', 'abcdef');
INSERT INTO students (name, surname, class, password) VALUES ('Fatma', 'Demir', '11C', 'qwerty');
INSERT INTO students (name, surname, class, password) VALUES ('Mehmet', 'Kaya', '12A', 'zxcvbn');
INSERT INTO students (name, surname, class, password) VALUES ('Zeynep', 'Şahin', '9B', 'asdfgh');
INSERT INTO students (name, surname, class, password) VALUES ('Mustafa', 'Çelik', '10A', 'jklşii');
INSERT INTO students (name, surname, class, password) VALUES ('Elif', 'Öztürk', '11B', 'mnbvyu');
INSERT INTO students (name, surname, class, password) VALUES ('Emre', 'Yıldız', '12B', 'trewqa');
INSERT INTO students (name, surname, class, password) VALUES ('Selin', 'Arslan', '9C', 'öçğüiş');
INSERT INTO students (name, surname, class, password) VALUES ('Oğuz', 'Doğan', '10C', 'hjklşi');

CREATE TABLE IF NOT EXISTS teacher (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    surname TEXT NOT NULL,
    password TEXT NOT NULL
);
INSERT INTO teacher (name, surname, password) VALUES ('Teacher', '1', '1234');
CREATE TABLE IF NOT EXISTS grades (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    student_id INTEGER NOT NULL,
    course TEXT NOT NULL,
    grade INTEGER NOT NULL,
    FOREIGN KEY (student_id) REFERENCES students(id)
);
SELECT * FROM grades;
SELECT
    s.id AS student_id,
    s.name,
    s.surname,
    s.class,
    g.course,
    g.grade
FROM
    students s
INNER JOIN
    grades g
ON
    s.id = g.student_id;

CREATE TABLE IF NOT EXISTS attendance (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    student_id INTEGER NOT NULL,
    date TEXT NOT NULL,
    FOREIGN KEY (student_id) REFERENCES students (id)
);
SELECT c.course_name, c.credits, g.grade,
       CASE WHEN g.grade >= 50 THEN 'Pass' ELSE 'Fail' END AS pass_fail
FROM grades g
JOIN courses c ON g.course = c.course_name
WHERE g.student_id = ?;

CREATE TABLE IF NOT EXISTS messages (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    student_id INTEGER, -- null olursa tüm öğrencilere gönderilmiş demektir
    message TEXT NOT NULL,
    sender TEXT NOT NULL DEFAULT 'Teacher',
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);
```