VIETNAMESE – GERMAN UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER SCIENCE DEPARTMENT

PROJECT REPORT

Classroom Management System

Module 13: Object Oriented Programming with Java

1. Nguyen Xuan Minh – 16209
2. Nguyen Vu Duy Khoi – 15777

Lecturer: Ngoc H. Tran, Ph.D.

Binh Duong, WS2021

Abbreviation

DBMS	Database Management System
JDBC	Java Database Connectivity
GUI	Graphical User Interface
IDE	Integrated Development Environment
DB	Database

List of Figures

List of Tables

Table of Contents

I. INTRODUCTION

We aim to provide an application that assists teachers within the CSE department to add/view/remove/search for students, attendances and register for exams.

Teachers can use these following functions via our application:

- Input a student’s information
- Input a student’s attendance
- Input a student’s exam registration
- Search for a student’s information
- Search for a student’s attendance
- Search for a student’s exam registration
- Display student information
- Display attendance information
- Edit the information of a student
- Delete a student
- Delete an attendance

OOP techniques are applied for designing classes. Encapsulation is used in class Student to hide its attributes. They can be accessed outside of the class using Getter & Setter methods.

II. CLASS ANALYSIS

1. Objects

- In this section, students analyse and list objects (name, behavior, state)
- Group objects having common states and behaviors

No.	Object Name	State	Behaviour

1	Student	Student First Name ("Minh"),
		Student Last Name ("Nguyen")
		Student ID ("16209")
		Major ("CSE")
			isRegisteredForExam()
			isAttendingClass()

2	Attendance	Student ID ("16209") Module ("Java OOP"), Date ("1/1/2022")	
3	Exam	Student ID ("16209") Module ("Java OOP"), Exam Date ("1/1/2022")	isAvailable()

Table1. List of Objects (Student, Exam)

1. Student
   1.1. Student, States(Minh Nguyen, 16209, CSE, 23/12/2021), Behaviours (isRegisteredForExam(), isAttendedClass(), …)
2. Exam
   2.1. Exam, States(Khoi, Nguyen, Java OOP, 1/1/2022), Behaviours (isAvailable())

2. Classes

a. Student

```
public class Student {
private int studentId;
private String firstName;
private String lastName;
private String major;
private String date;
Student (int studentId, String firstName, String lastName, String major, String
date){
this.studentId = studentId;
this.firstName = firstName;
this.lastName = lastName;
this.major = major;
this.date = date;
}
public int getStudentId() {
return studentId;
}
public void setStudentId(int studentId) {
this.studentId = studentId;
}
public String getFirstName() {return firstName;}
public void setFirstName(String firstName) {
this.firstName = firstName;
}
public String getLastName() {return lastName;}
public void setLastName(String lastName) {
this.lastName = lastName;
}
public String getMajor() {return major;}
public void setMajor(String major) {
this.major = major;
}
public String getDate() {return date;}
public void setDate(String date) {this.date = date;}
}
```

b.Exam

```
public class Exam {
private string StudentFirstName;
private String StudentLastName;
private int StudentID;
private String ExamName;
Student (int StudentID, String StudentFirstName, String StudentFirstName, String ExamName){
this.StudentFirstName = StudentFirstName;
this.StudentLastName = StudentLastName;
this.StudentID = StudentID;
this.ExamName = ExamName;
}
public int getStudentId() {
return studentId;
```

```
}
public void setStudentId(int studentId) {
this.studentId = studentId;
}
public String getStudentFirstName() {return Student_First_Name;}
public void setStudentFirstName(String Student_First_Name) {
this.Student_First_Name = Student_First_Name;
}
public String getStudentLastName() {return Student_Last_Name;}
public void setStudentLastName(String Student_Last_Name) {
this.Student_Last_Name = Student_Last_Name;
}
public String getExamName() {return ExamName;}
public void setExamName(String Exam_Name) {
this.Exam_Name = Exam_Name;
}
package com.company.other;

public class Exam {
    private String firstName;
    private String lastName;
    private int studentId;
    private String exam;
    Exam (int studentId, String firstName, String lastName, String exam){
        this.firstName = firstName;
        this.lastName = lastName;
        this.studentId = studentId;
        this.exam = exam;
    }
    public int getStudentId() {
        return studentId;
    }
    public void setStudentId(int studentId) {
        this.studentId = studentId;
    }
    public String getStudentFirstName() {return firstName;}
    public void setStudentFirstName(String Student_First_Name) {
        this.firstName = Student_First_Name;
    }
    public String getStudentLastName() {return lastName;}
    public void setStudentLastName(String Student_Last_Name) {
        this.lastName = Student_Last_Name;
    }
    public String getExamName() {return exam;}
    public void setExamName(String Exam_Name) {
        this.exam = Exam_Name;
    }
}
```

III. CLASS DESIGN

1. Classes

No	Class	Instance Variables	Methods	Description

1	Student	private int studentId;
		
		private String firstName;
		
		private String lastName;
		
		private String major;
		
		private String date;
			public int getStudentId()
			- This method is used to get the student id and return the id
			
			public void setStudentId(int studentId)
			
			public int getStudentId()
			public void setStudentId(int studentId)
			- This method is used to set the student id with input parameter
			public String getFirstName()
			public void setFirstName(String firstName)
			public String getLastName()
			public void setLastName(String lastName)
			public String getMajor()
			public void setMajor(String major)
			public String getDate()
			public void setDate(String date)
				This class is used to manage Student Information and Attendance as well as behaviours.

2	Exam	private string Student_First_Name;
		
		private string Student_Last_Name;
		
		private int Student_Id;
		
		private string Exam_Name;
			public string checkAttendance(int studentId, String examName)
			- This function is used to check whether student meets exam prerequisites or not using student attendance count
				This is used to manage Exam Registration Information and behaviours

DUTY ROSTER

ID	Task	In Charge	Start	End	State	Note
1	Design Class Student	Nguyen Xuan Minh	24-Dec-2021	30-Dec-2021	Done	
2	Design Class Exam	Nguyen Vu Duy Khoi	20-Dec-2021	02-Jan2022	Done	
3	Code Function checkAttendace()	Nguyen Xuan Minh	30-Dec-2021	11-Jan-2022	Done	
4	Code Function checkStudent()	Nguyen Xuan Minh	28-Dec-2021	14-Jan-2022	Done	
5	Code Function showStudent()	Nguyen Xuan Minh	30-Dec-2021	14-Jan-2022	Done	
6	Connect Database via JDBC	Nguyen Xuan Minh	28-Dec-2021	26-Jan-2022	Done	
7	Develop GUI	Nguyen Xuan Minh &	30-Dec-2021	26-Jan-2022	Done	

		Nguyen Vu Duy Khoi				
8	Design Database	Nguyen Xuan Minh	1-Jan-2022	26-Jan-2022	Done	Fields for Exam Registration is under development
9	Create a React Front End	Nguyen Vu Duy Khoi	24-Dec-2021	26-Dec-2021	Done	
10	Create a Node Back End	Nguyen Vu Duy Khoi	26-Dec-2021	28-Dec-2021	Done	
11	Setting up Mongo DB	Nguyen Vu Duy Khoi	31-Dec-2021		In Progress	
12	Connecting Front and Back Ends	Nguyen Vu Duy Khoi	31-Dec-2021		In Progress	
13	Hosting the site	Nguyen Vu Duy Khoi	15-Jan-2022		In Progress	
14	Design Application Classes	Nguyen Xuan Minh	1-Jan-2022	26-Jan-2022	Done	

2. *Some OOP techniques*

a. *Overloading method:*

Method Overloading by changing the data type of parameters:

```
class MethodOverloading {
   // this method accepts int
   private static void display(int a){
      System.out.println("Number of students.");
   }
   // this method  accepts String object
   private static void display(String a){
      System.out.println("Name of students.");
   }
   public static void main(String[] args) {
      display(10);
      display("Java_Project");
   }
}
```

This is the result of my example:

```
Number of students.
Name of students.
```

b. *Overriding method*

```
//Creating a parent class.
```

```
class Student1{
  void run() {
System.out.println("Student 1.");
}
}
//Creating a child class
class Student2 extends Student1{
 //defining the same method as in the parent class
  void run(){System.out.println("Student2.");}
  public static void main(String args[]){
  Student2 obj = new Student2();//creating object
  obj.run();//calling method
  }
```

Output:

```
Student2.
```

3. *Inheritance*

```
class Student_Attendance{
 Int attendance = 5;
}
class Attendance extends Student_Attendance{
 int full_attendance = 20;
 public static void main(String args[]){
   Attendance p=new Attendance();
   System.out.println("Attendance of student is:"+p.attendance);
   System.out.println("The attendance of the class is:"+p.full_attendance);
}
}
```

Output:

```
Attendance of student is: 5
 The attendance of the class is: 20
```

IV. Package Design

com package: contains all of the classes, sub-packages, interfaces

1. company package: is the sub-package of com, contains most of the classes of the program and a Main class to execute the application.
    a. add package: is the sub-package of company, contains classes and GUI forms that are specifically use for adding data into the database of the application
    b. edit package: is the sub-package of company, contains classes and GUI forms that are designed for editing data inside the database of the application
    c. view package: is the sub-package of company, contains classes and GUI forms that are built to show data inside the database of the application
2. connection package: is the sub-package of com, contains a single class that is designed for establishing connection between the application and the database

3. images package: is the sub-package of com, contains images that are used for creating icons of buttons, labels inside multiple frames across the application

V. Interface Design

- In this project, we are going to design an interface as GUIs or UIs on the web application.
- It's the visual part of an application or device that determines how a user interacts with it and how information is displayed on the screen.

VI. Access Control

Table 1: Data Access Control Table

No	Data	Class	Modifier	Description
1	student_id, module_id, date	AddAttendance	Public	Name can be public as it is not private information. It does not cause serious consequences if the others know the name of the student.
2	student_id, module_id, attempt	RegisterExam	Private	The attempt of students who enroll in the module must be private as it can cause serious consequences if the others know it.
3	module_id, module_name, exam_date	RegisterExam	Public	Name and id of the module can be public as it is not private information. It does not cause serious consequences if the others know about it. The name and id of the module can be accessed by

				the other classes inside or outside the package of classroom_module.
4	module_id, student_id	EditStudent	Public	Id of student and module can be public as it is not private information. It does not cause serious consequences if the others know about it. The id of module student can be accessed by the other classes inside or outside the package of classroom_stu_modu_id.
5	student_id, first_name, last_name, major	AddStudent	Public	Name, major and id of the student can be public as it is not private information. It does not cause serious consequences if the others know about it.

				The name, id, major of student can be accessed by the other classes inside or outside the package of classroom_student.
6	id, name, password	classroom_teacher	Private	The name and password of the teacher must be private as it can cause serious consequences if the others know it.

VII. Encapsulation vs Inheritance vs Polymorphism

1. Encapsulation

- In encapsulation, the data in a class is hidden from other classes using the data hiding concept which is achieved by making the members or methods of a class private, and the class is exposed to the end-user or the world without providing any details behind implementation using the abstraction concept, so it is also known as a combination of data-hiding and abstraction.
- We use an interface to hide the implementation of the objects in the EditAttendance class. We hide the attendanceSQL variable initialization from the EditAttendance class and then use the variable itself by implementing the Attendance interface.

Attendance Interface

```
 package com.company;


public interface Attendance {
    String attendanceSQL = "SELECT a.student_id AS StudentID,\n" +
            "       m.module_name AS Module,\n" +
            "       a.date AS Date\n" +
            "FROM attendance a\n" +
            "LEFT JOIN module m ON a.module_id = m.module_id\n";
}
```

EditAttendance Class

```
    private String getCheckAttendance(JTextField id) {
        String checkAttendanceSQL = attendanceSQL + "WHERE a.student_id =" + 
id.getText();
        return checkAttendanceSQL;
```

```
            }
```

- We also attempt to hide the variables in the class AppendStudent by declaring them as private and then add public methods, or Getter and Setter, to get variables such as (id, name, major) and set variable major as it can be changed later. We will call these methods to get access to hidden variables of another class in the Execute class.

```
class AppendStudent {
    private int id;
    private String name;
    private String major;

    // getter - get id of student
    public int getId() {
        return id;
    }

    // getter - get name of student
    public String getName() {
        return name;
    }

    // getter - get major of student
    public String getMajor() {
        return major;
    }

    // setter - set major of student
    public void setMajor(String major) {
        this.major = major;
    }

    AppendStudent(){};

    public AppendStudent(int id, String name, String major) {
        this.id = id;
        this.name = name;
        this.major = major;
    }


    public void display() {
        System.out.println("Student ID: " + this.id + ", Name: " +
                this.name + ", Major: " + this.major);
    }
}


class Execute {
    public static void main(String[] args) {
        // create a new instance of AppendStudent
        AppendStudent as = new AppendStudent(16209, "Minh Nguyen", "CSE");
        System.out.println("Student ID: " + as.getId() + ", Name: " +
                as.getName() + ", Major: " + as.getMajor());
        as.setMajor("Java OOP");
        as.display();
     }
```

```
}
```

We will have a result as follows:

```
Student ID: 16209, Name: Minh Nguyen, Major: CSE

Student ID: 16209, Name: Minh Nguyen, Major: Java OOP
```

As described, getter and setter have successfully accessed the private variables. This is our example:

2. Inheritance

- In Java, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:
- subclass ModifyAttendance (child) - the class that inherits from another class
- superclass ModifyStudent (parent) - the class being inherited from
- To inherit from a class, use the extends keyword like class ModifyAttendance extends ModifyStudent
- For Method Overloading, we overload the display() method in the subclass so that we can achieve different results by adding different parameters.
- For Code Reusability.

```
class ModifyStudent {

   int id;

   String name;

   String major;
```

```
    public void edit() {
        EditStudent eds = new EditStudent();
        eds.setVisible(true);
    }
}

class ModifyAttendance extends ModifyStudent {

    public void display() {
        System.out.println("Student ID: " + id + ", Name: " +
                name + ", Major: " + major);
    }

    public void display(String date) {
        System.out.println("Student ID: " + id + ", Name: " +
                name + ", Major: " + major + ", Date: " + date);
    }
}
```

```
class Execute {
    public static void main(String[] args) {
        // create an object of a subclass
        ModifyAttendance mds = new ModifyAttendance();

        // access variables of superclass
        mds.id = 16209;
        mds.name = "Minh Nguyen";
        mds.major = "Java Project";

        // display frame of the superclass
        mds.edit();

        // access method of the superclass using subclass object
        mds.display();

        // access overloaded method of the subclass
        mds.display("2021-05-09");
    }
}
```

We will achieve a result as follows:

```
Student ID: 16209, Name: Minh Nguyen, Major: Java Project
Student ID: 16209, Name: Minh Nguyen, Major: Java Project, Date: 2021-05-09
```

As above, variables and methods from the subclass are both inherited from the superclass

3. Polymorphism

We use polymorphism using method overriding to call the same method in different classes. We aim to display different frames each time the method is called using the instance of a particular class.

```
class AppendStudent {
    public void action() {
        System.out.println("Append a student:");


        // create an instance of AddStudent
        AddStudent = new AddStudent();
        // display the frame
        addStudent.setVisible(true);
    }
}


class AppendAttendance extends AppendStudent{
    public void action() {
        System.out.println("Append an attendance:");


        // create an instance of AddAttendance
        AddAttendance = new AddAttendance();
        // display the frame
        addAttendance.setVisible(true);
    }
}



class Execute {
    public static void main(String[] args) {
        AppendStudent = new AppendStudent();

        AppendAttendance = new AppendAttendance();


        appendStudent.action();

        appendAttendance.action();

    }

}
```

By calling action() methods from 2 different class objects, we achieved two frames, one from AddAttendance and another from AddStudent

VIII. Experiment

1. Environment and Tools

a. Environment:

We have tested the project in the total number of two laptops as follows:

Laptop	CPU	RAM	ROM	OS
Laptop 1 16209	Intel Core i5-8265U CPU @	12GB	256GB	Microsoft Windows 11 Pro

	1.60GHz, 1800 Mhz			
Laptop 2 15777	Intel(R) Core(™) i7-8750H CPU @ 2.20GHz	16GB	464GB	Windows 10 Pro 64-bit (10.0, Build 19043)

b. Tools:

Environment	IDE	GUI	Libraries
Application	Intellij IDEA 2021.2.3	Java Swing	● mysql-connector-java-8.0.27.jar ● rs2xml.jar

Application	Visual Studio Code 1.63.2 (November 2021)	JavaScript	

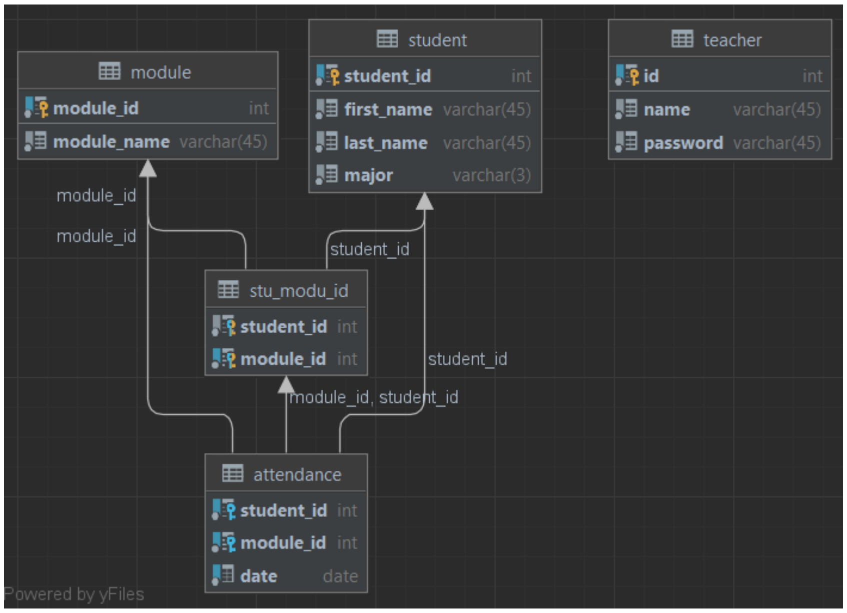
2. Project functions

Name	Attributes	Database Fields	Description
Add Student	Student ID, First Name, Last Name, Major, Module	student, stu_modu_id, module	Add students by passing their studentId, firstName, lastName, major, module as inputs of the Controller, and then they are written into the Database. Attributes of studentID, firstName, lastName and major are updated to the student table, while studentId and module are added to stu_modu_id table.

Edit Student	Student ID, First Name, Last Name, Major, Module	student, stu_modu_id, module	Edit students by using new data from user
Add Attendance	Student ID, Module, Date,	attendance	Add attendances by adding student information such as Student ID, Module, Date to the Controller and then they are updated back to the attendance table
View Attendance	No attributes	attendance	View attendance by list all of the available attendances of the program through the attendance table

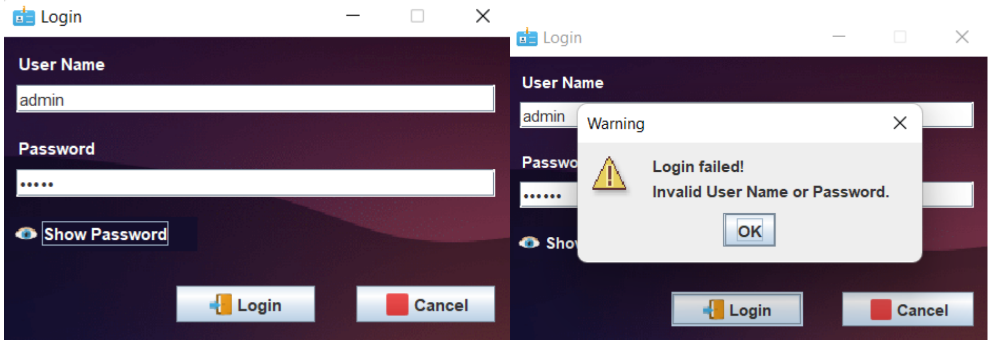
3. Databases

- For the application, we use MySQL as the DBMS. Additionally, DataGrip 2021.3.1 and MySQL Workbench 8.0 CE are the IDEs for the Database.
- Database Relational Diagram



4. GUI

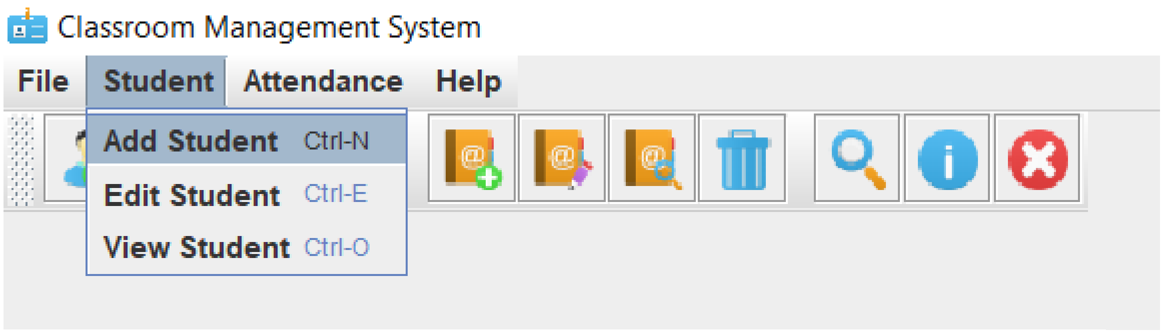
a. Figure 1: Login Dialog

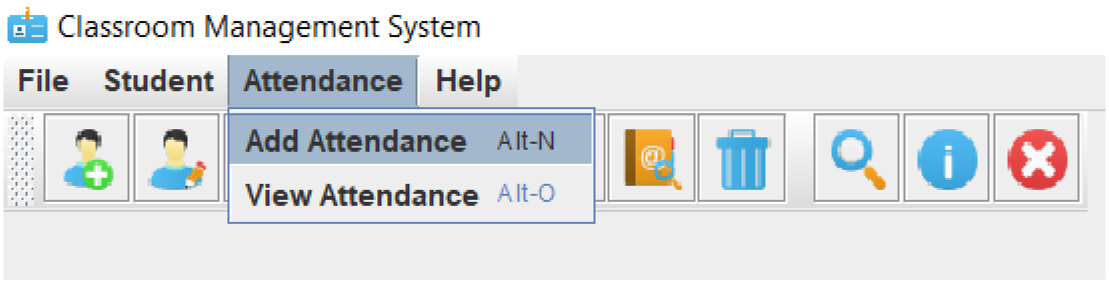


When the user presses the button “Login”, the application will check for the equality between the inputs in the two JTextFields and the data stored inside the teacher table in the database. If the User Name and the Passwords are matched, the user will be moved to the Home page. In case of data mismatch, a Warning Dialog will pop up to warn the user. All happens under the method of login(ActionEvent e). Pressing the “Cancel” button will shut down the application with method cancel(ActionEvent e).

If the user toggles the "Show Password" checkbox, the Password will be shown or hidden with the method showPassword(ActionEvent e). All of the methods are in the class Login.

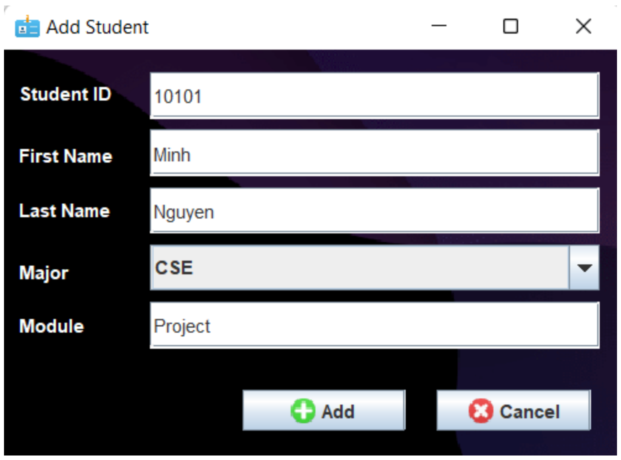
b. Figure 2: Home Dialog



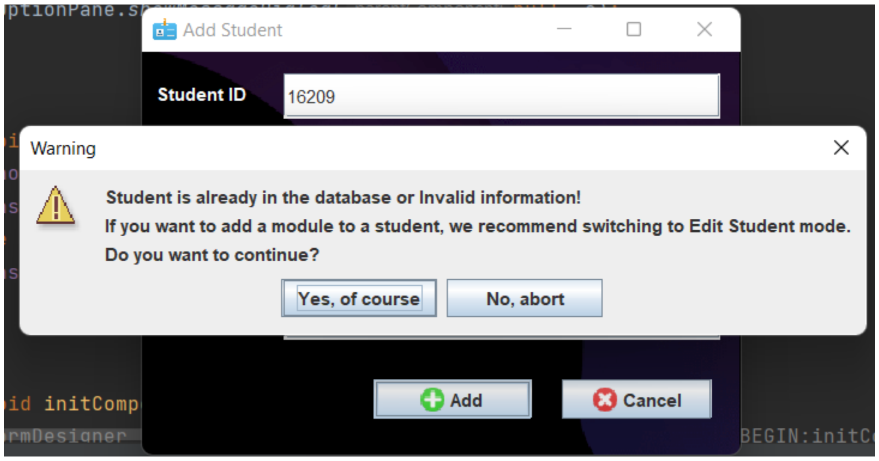


When the user selects the Student menu, an option to "Add Student" will be shown in the menu items. If the user clicks on the Add Student menu item, an Add Student Dialog will pop up via the method addStudent(ActionEvent e). The same thing happens with the Attendance menu. When the "Add Attendance" menu item is clicked, an Add Attendance Dialog will show up through the method addAttendance(ActionEvent e) of the class Home.

c. Figure 3: Add Student Dialog


Add Student

Student ID: 10101
First Name: Minh
Last Name: Nguyen
Major: CSE
Module: Project


Add Student

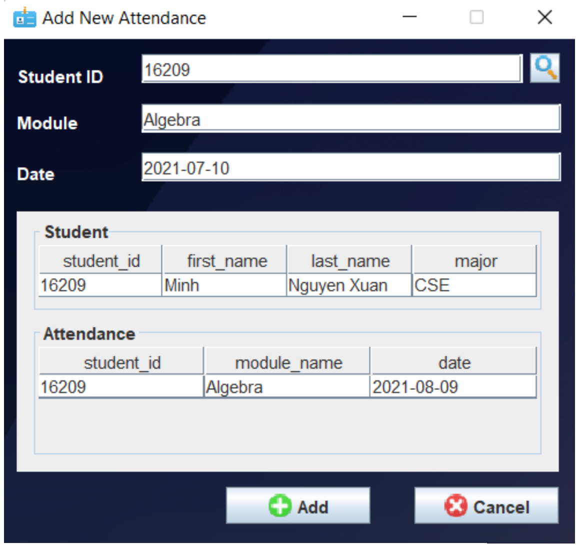
Student ID: 16209

Warning

Student is already in the database or Invalid information!
If you want to add a module to a student, we recommend switching to Edit Student mode.
Do you want to continue?

In the class AddStudent, when the user presses the "Add" button, his/her information, which includes Student ID, First Name, Last Name and Major will be stored to the student table in the database, while Module is stored to the stu_modu_id table with the student_id and the corresponding module_id taken from module table in the database through the method addStudent(ActionEvent e). If there are errors such as missing information or students not existing, the Warning dialog will show up, and then open the Edit Student Dialog when the user presses the "Yes" button.

d. Figure 4: Add Attendance Dialog



Student ID: 16209

Module: Algebra

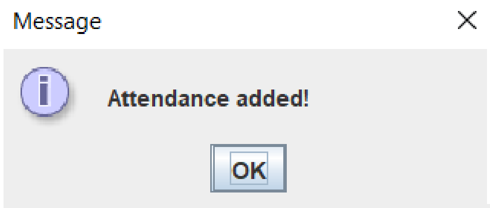
Date: 2021-07-10

Student

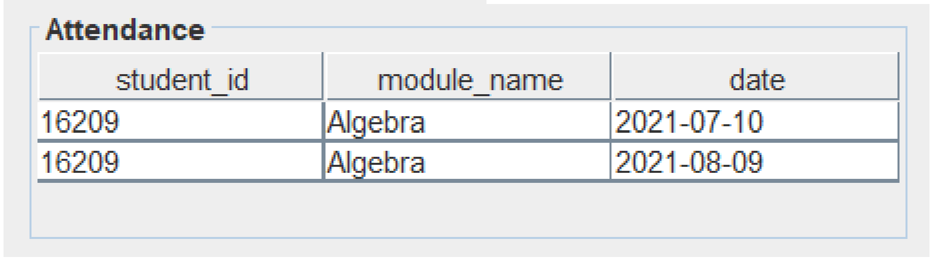
student_id	first_name	last_name	major
16209	Minh	Nguyen Xuan	CSE

Attendance

student_id	module_name	date
16209	Algebra	2021-08-09



Attendance added!



Attendance

student_id	module_name	date
16209	Algebra	2021-07-10
16209	Algebra	2021-08-09

When the user presses the magnifying glass button, the application will look up for data that matches the Student ID in both the student table and attendance table. The records will be shown on the two tables via the method checkStudent(ActionEvent e). If the "Add" button is pressed, his/her data (Student ID, Module, Date) will be updated to the attendance table in the database with the method addNewAttendance(ActionEvent e). The records will show up on the attendance table of the dialog. This happens under the class AddAttendance.

IX. Conclusion

1. Project's Pros and Cons:
    a. Pros:
        i. We will demonstrate this project with 2 applications:
            1. GUI
            2. Web application
        ii. This project will help teachers to manage their class more easily than before by using some special functions for each role.
    b. Cons:
        i. It's too difficult to connect with databases by using web applications.
        ii. Our project can not instantiate its design due to lack of interface and abstract class during doing the project.
2. We think that our project can be scored around 75 - 85 / 100. As our project which is not good at all (i.e. It lacks some OOP functions, more functions in the app, full of conditions and roles in the database, however, to be precise, it can represent 2 kinds of applications such as GUI and web.)
3. In the future, we would like to improve the project better, for instance, we will add more classes like "Exam_Registration" and "Attend_Class" to control easily with a huge number of students in the school.

REFERENCE

1. OOP Concept for Beginners, FreeCodeCamp
   https://www.freecodecamp.org/news/java-object-oriented-programming-system-principles-oops-concepts-for-beginners/

2. Java Tutorials, Oracle
   https://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html

3. Java Tutorials, BroCode
   https://youtu.be/xk4_1vDrzzo

4. SQL Tutorial, tutorialspoint
   https://www.tutorialspoint.com/sql/sql-sub-queries.htm

5. Create a React Front End
   https://www.youtube.com/watch?v=HT82p_re-EY

6. Create a Node Back End
   https://www.freecodecamp.org/news/building-a-simple-node-js-api-in-under-30-minutes-a07ea9e390d2/

7. Setting up MongoDB
   https://www.prisma.io/dataguide/mongodb/setting-up-a-local-mongodb-database

8. Connecting the Front and Back Ends
   https://www.youtube.com/watch?v=ACI8EDbaXzo

9. Learn React JS and .NET Core API by Creating a Full Stack Web App from Scratch
   https://www.youtube.com/watch?v=gpfP60KjmZU

APPENDIX A: DATABASE

- Using MySQL database
- Data Diagram (Tables to use relational database system management)

APPENDIX B: WEB APPLICATION

- Using React to create a web application
- Using MongoDB database
- The Fetch API

(Submitting by the end of this week)

APPENDIX C: GUI

I. Libraries

JFormDesigner https://www.formdev.com/jformdesigner/doc/ides/eclipse/ is also a Eclipse plugin

rs2xml https://hacksmile.com/rs2xml-jar-free-download/

II. Figures

This is the extension of the GUI in VIII: Experiment - Part 4. We will include all of the figures of the GUI rather than just four basic ones.

Figures of the GUI Table

Figure Order	**Figure Name**	**Methods**
1	Login Dialog	● initComponents() ● login(ActionEvent e) ● showPassword(ActionEvent e) ● cancel(ActionEvent e)
2	Home Dialog	Methods to call others dialogs
3	Add Student Dialog	● initComponents() ● addStudent(ActionEvent e) ● cancel(ActionEvent e)
4	Edit Student Dialog	● initComponents() ● updateStudent(ActionEvent e) ● removeStudent(ActionEvent e) ● cancel(ActionEvent e)
5	View Student Dialog	● initComponents()

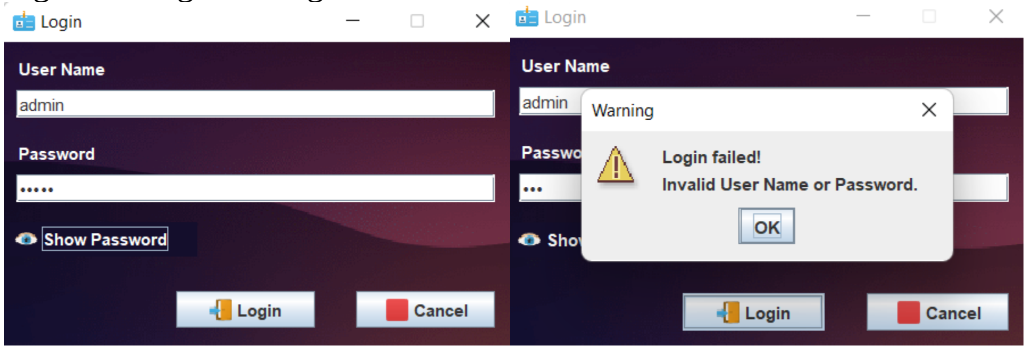
		• addStudent(ActionEvent e) • refresh(ActionEvent e) • cancel(ActionEvent e)
6	Remove Student Dialog	• initComponents() • removeStudent(ActionEvent e) • cancel(ActionEvent e)
7	Add Attendance Dialog	• initComponents() • addAttendance(ActionEvent e) • cancel(ActionEvent e)
8	Edit Attendance Dialog	• initComponents() • updateAttendance(ActionEvent e) • removeAttendance(ActionEvent e) • attendanceTabelDisplayMouseClicked(MouseEvent e) • cancel(ActionEvent e)
9	View Attendance Dialog	• initComponents() • addAttendance(ActionEvent e) • refresh(ActionEvent e) • cancel(ActionEvent e)
10	About Dialog	• addAttendance(ActionEvent e) • refresh(ActionEvent e) • cancel(ActionEvent e)

11	Register Exam Dialog	● initComponents() ● addExamRegistration(ActionEvent e)cancel(ActionEvent e)
12	Search Dialog	● initComponents() ● searchStudent(ActionEvent e) ● searchAttendance(ActionEvent e) ● searchExam(ActionEvent e) ● cancel(ActionEvent e)

Detail of Figures

initComponents() will be the initial methods or the primary methods that contain all of the methods in the class. Moreover, initComponents() will also build the GUI frame of the class

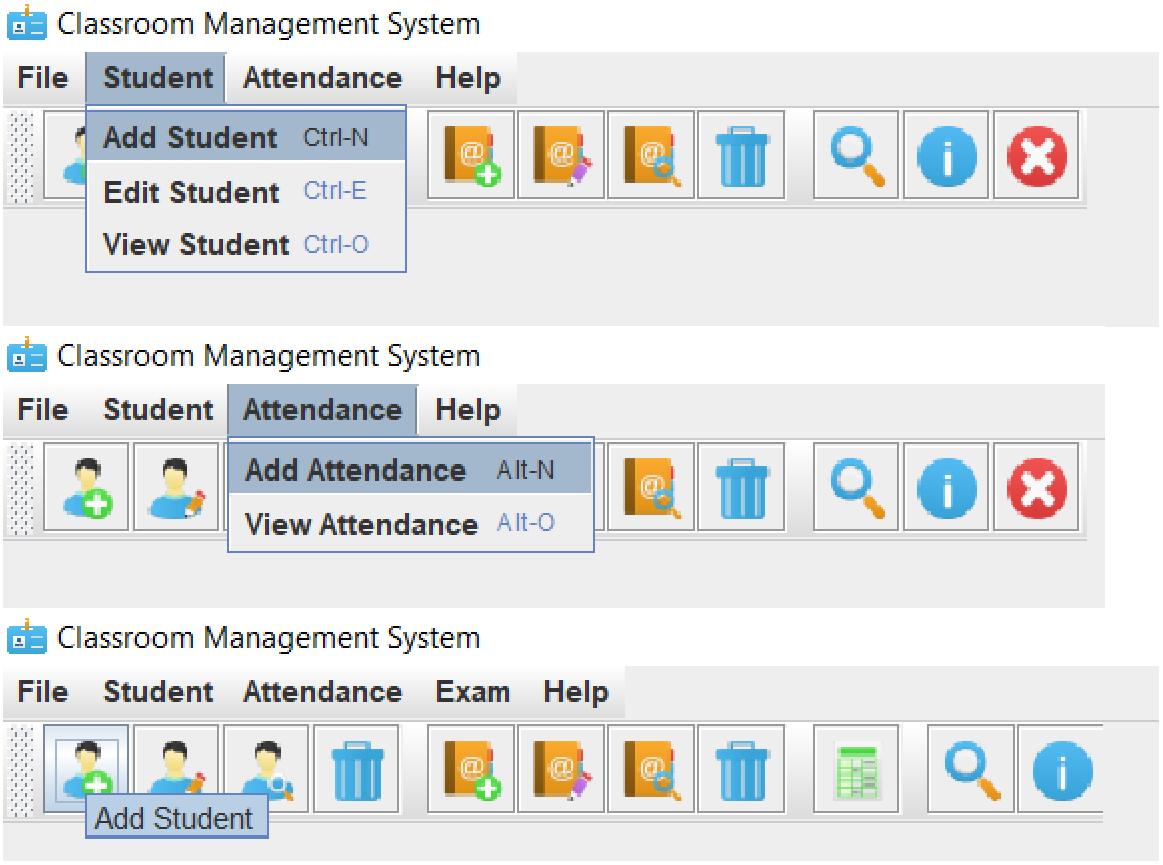
a) Figure 1: Login Dialog



Warning
Login failed!
Invalid User Name or Password.
OK

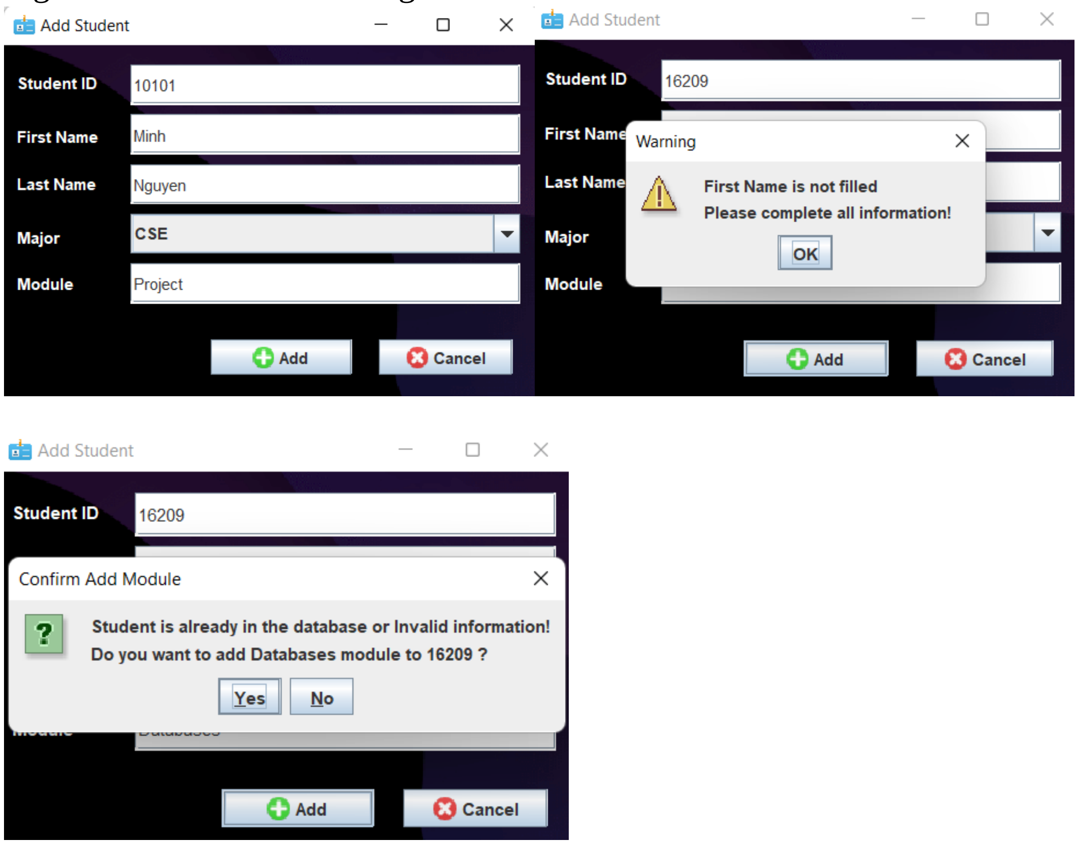
When the user presses the button "Login", the application will check for the equality between the inputs in the two JTextFields and the data stored inside the teacher table in the database. If the User Name and the Passwords are matched, the user will be moved to the Home page. In case of data mismatch, a Warning Dialog will pop up to warn the user. All happens under the method of login(ActionEvent e). Pressing the "Cancel" button will shut down the application with method cancel(ActionEvent e). If the user toggles the "Show Password" checkbox, the Password will be shown or hidden with the method showPassword(ActionEvent e). All of the methods are in the class Login.

b) Figure 2: Home Dialog



When the user selects the Student menu in the class Home, an option to "Add Student" will be shown in the menu items. If the user clicks on the Add Student menu item, an Add Student Dialog will pop up via the method addStudent(ActionEvent e). The same thing happens with the Attendance menu. When the "Add Attendance" menu item is clicked, an Add Attendance Dialog will show up through the method addAttendance(ActionEvent e) of the class Home. If the user is hovering over a Toolbar icon, a toolTipText will be shown to guide the user.

c) Figure 3: Add Student Dialog



Add Student

Student ID: 10101
First Name: Minh
Last Name: Nguyen
Major: CSE
Module: Project

Add Student

Student ID: 16209

Warning

First Name is not filled
Please complete all information!

Add Student

Student ID: 16209

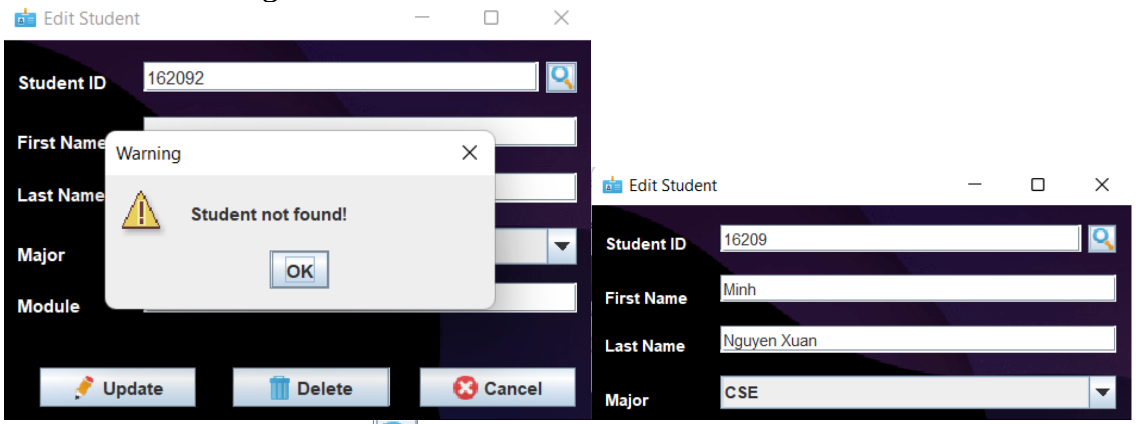
Confirm Add Module

Student is already in the database or Invalid information!
Do you want to add Databases module to 16209 ?

When the user presses the "Add" button in the class AddStudent class, his/her information, which includes Student ID, First Name, Last Name and Major will be stored to the student table in the DB, while Module is stored to the exam_regis table with the student_id, the corresponding module_id taken from module table and the default attempt as "NULL" to indicate that the student have not register for that exam. The method addStudent(ActionEvent e) will be called to handle such tasks. If the user left a field blank, the Warning dialog will show up until there are no missing fields with isMissing(JTextField[], String[]) method.Even If the student is already in the

DB, a Warning dialog will appear to warn the user and ask for the next actions. When the user presses the “Yes” button, Module will be added to the DB as a new module for the student. “No” or “Cancel” button will discard the dialog with the cancel(ActionEvent e) method.

d) Edit Student Dialog

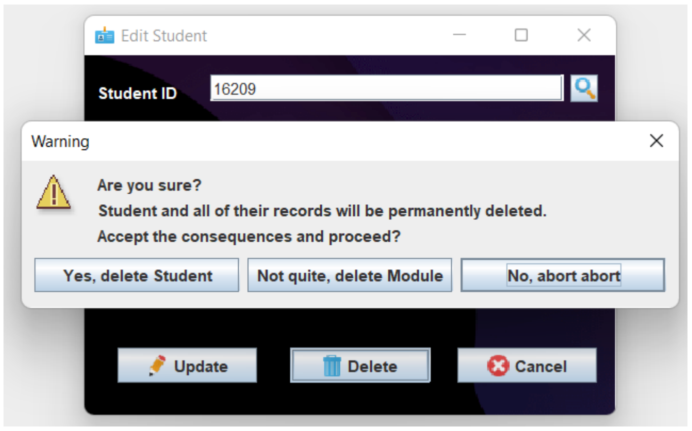


Student ID: 162092

Student not found!

Student ID: 16209
First Name: Minh
Last Name: Nguyen Xuan
Major: CSE

When the user presses the “Check” button, FirstName, LastName and Major information related to the StudentID in the DB will be imported to the input fields with the checkStudent(ActionEvent e) method. If the student does not exist, a Warning dialog will pop up to warn the user. When the “Update” button is pressed, the information of the student will be updated to the DB with (StudentID, FirstName, LastName and Major) to the student table via the method updateStudent(ActionEvent e).

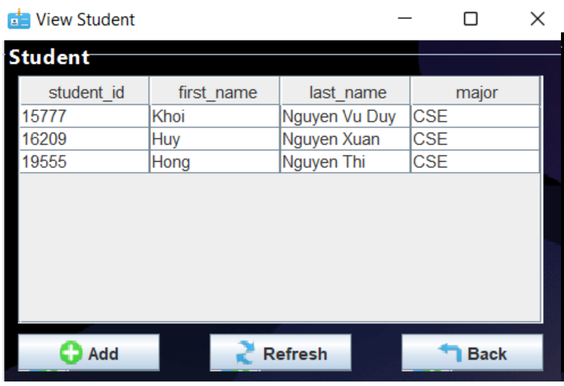


Student ID: 16209

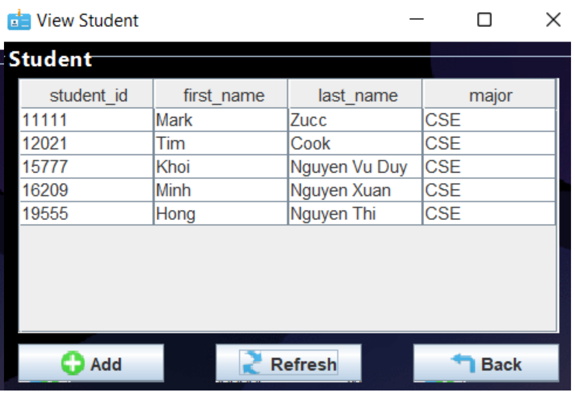
Are you sure?
Student and all of their records will be permanently deleted.
Accept the consequences and proceed?

The “Remove” button will remove the information based on the choice of the user in the Warning dialog with the method removeStudent(ActionEvent e). If the user chooses “Yes, remove Student”, the student’s information will be deleted from the student table in the DB. In case of “Not quite, remove Module”, only the Module of the student will be removed from the exam_regis table. “No, abort abort” or “Cancel” button will close the dialog. All methods are implemented in the class Edit Student

e) View Student Dialog



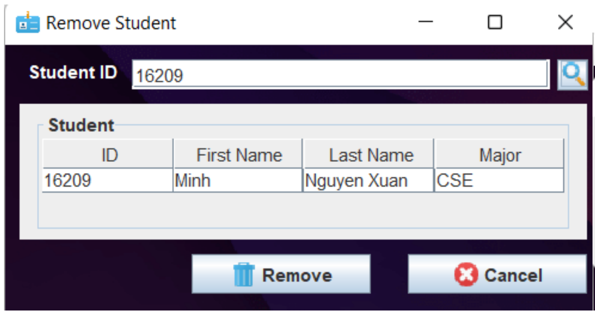
student_id	first_name	last_name	major
15777	Khoi	Nguyen Vu Duy	CSE
16209	Huy	Nguyen Xuan	CSE
19555	Hong	Nguyen Thi	CSE



student_id	first_name	last_name	major
11111	Mark	Zucc	CSE
12021	Tim	Cook	CSE
15777	Khoi	Nguyen Vu Duy	CSE
16209	Minh	Nguyen Xuan	CSE
19555	Hong	Nguyen Thi	CSE

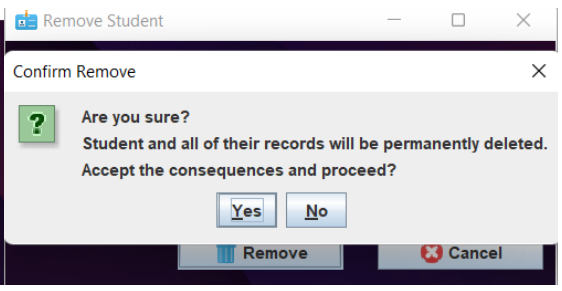
In the class View Student, when the user presses the “Add” button, the application will close the current dialog and open up the Add Student dialog. Every change in the DB will not be directly shown up on the Student table. The user needs to press the “Refresh” button to get the latest data on the Student table. “Back” button will close the dialog.

f) Remove Student



Student ID 16209

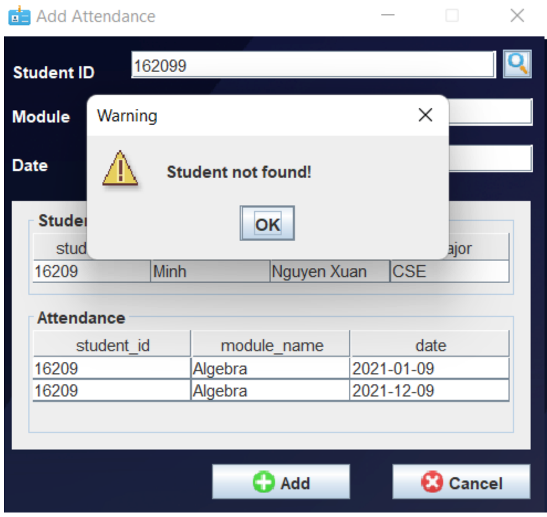
ID	First Name	Last Name	Major
16209	Minh	Nguyen Xuan	CSE



Are you sure?
Student and all of their records will be permanently deleted.
Accept the consequences and proceed?

When the user presses the “Check” button, with the checkStudent(ActionEvent e) method, all information related to the StudentID will be displayed in the Student table below. If the user clicks the “Remove” button, an Option dialog will appear to confirm the deletion of the student with the “Yes”button, and then the record of that student is wiped out from the DB . “No” or “Cancel” button will discard the dialog. All happens in the RemoveStudent class.

g) Add Attendance



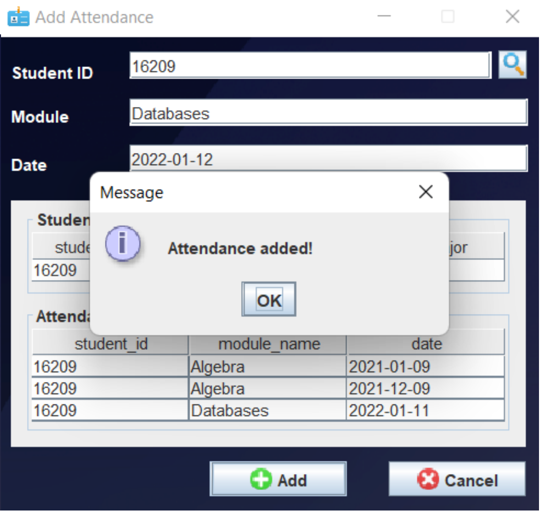
Student ID: 162099
Module
Date

Warning
Student not found!

stud…			…ajor
16209	Minh	Nguyen Xuan	CSE

Attendance

student_id	module_name	date
16209	Algebra	2021-01-09
16209	Algebra	2021-12-09



Student ID: 16209
Module: Databases
Date: 2022-01-12

Message
Attendance added!

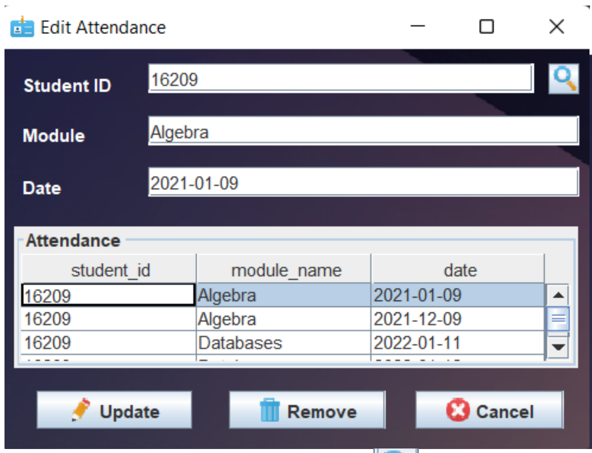
stude…	…jor
16209	

Attend…

student_id	module_name	date
16209	Algebra	2021-01-09
16209	Algebra	2021-12-09
16209	Databases	2022-01-11

When the user presses the “Check” button, the application will look up for data that matches the Student ID in both the student table and attendance table. The records will be shown on the two tables via the method checkStudent(ActionEvent e). If the “Add” button is pressed, his/her data (Student ID, Module, Date) will be updated to the attendance table in the database with the method addAttendance(ActionEvent e). The records will show up on the attendance table of the dialog. This happens under the class AddAttendance.

h) Edit Attendance

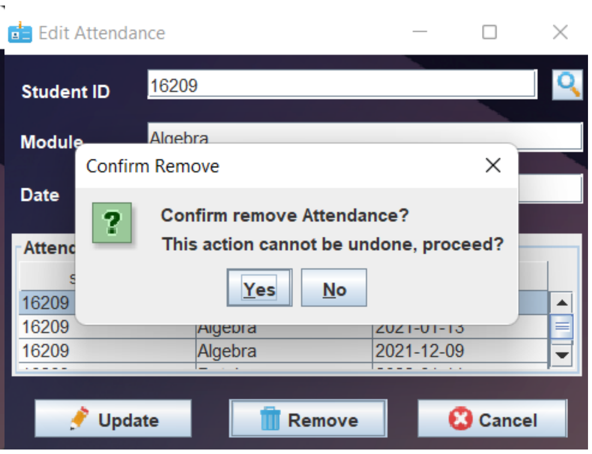


Edit Attendance

Student ID: 16209
Module: Algebra
Date: 2021-01-09

Attendance

student_id	module_name	date
16209	Algebra	2021-01-09
16209	Algebra	2021-12-09
16209	Databases	2022-01-11



Edit Attendance

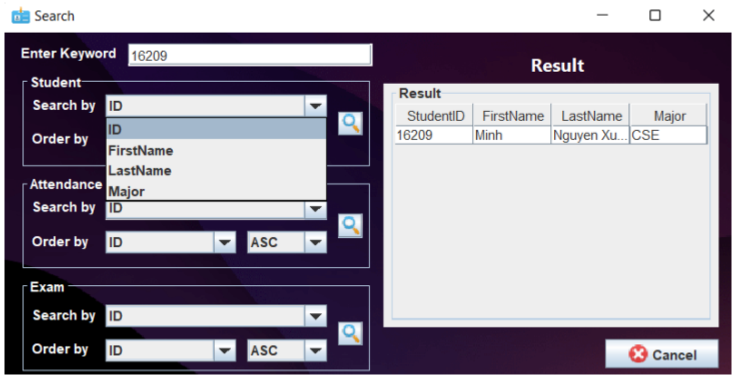
Student ID: 16209
Module: Algebra

Confirm Remove

Confirm remove Attendance?
This action cannot be undone, proceed?

When the user presses the “Check” button in the EditAttendance class, all information related to the StudentID will be displayed in the Attendance below via the checkStudent(ActionEvent e) method. If the user clicks on any row in the Attendance table, its corresponding data will be imported to the input fields via method attendance Table Display MouseClicked(MouseEvent e). “Update” button will update the new attendance to the attendance table via method updateAttendance(ActionEvent e), while the “Remove” button deletes the information of attendance related to the input data in the attendance table via method removeAttendance(ActionEvent e).

i) Search



Search

Enter Keyword: 16209

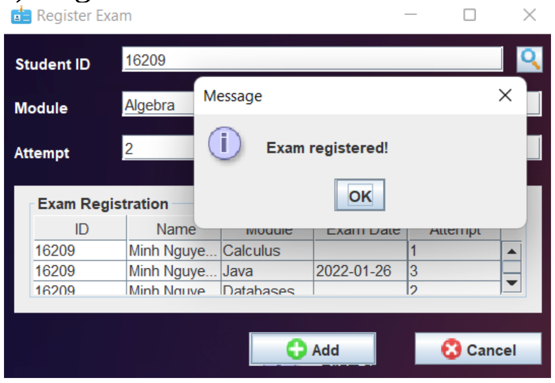
Student — Search by: ID; Order by: ID / FirstName / LastName / Major
Attendance — Search by: ID; Order by: ID, ASC
Exam — Search by: ID; Order by: ID, ASC

Result

StudentID	FirstName	LastName	Major
16209	Minh	Nguyen Xu...	CSE

In the Search class, when the user hits the “Check” button, the application will look for the information in the keyword field using the method searchStudent(ActionEvent e), searchAttendance(ActionEvent e) and searchExam(ActionEvent e) corresponding to the fields which the user wants to search. The result will be displayed on the Result table via the method showResultTable(String sql). The user can choose the filters in the combo boxes which are linked with each field DB’s columns. For instance, Search by combo box in Student fields is linked to the student table in the DB with 4 columns student_id, first_name, last_name and major. “Cancel” button will close the current dialog

f) Register Exam



Student ID 16209

Module Algebra

Attempt 2

Message

Exam registered!

OK

Exam Registration

ID	Name	Module	Exam Date	Attempt
16209	Minh Nguye...	Calculus		1
16209	Minh Nguye...	Java	2022-01-26	3
16209	Minh Nguye...	Databases		2

When the user presses the “Check” button in the RegisterExam class, all information related to the StudentID will be displayed in the Exam Registration below via the check(ActionEvent e) method. If the user clicks the “Add” button, a new exam registration of that student will be added to the DB using the method addExamRegistration(ActionEvent e). “Cancel” button will discard the current dialog.