Vietnam National University HCM
International University
School of Computer Science and Engineering

REPORT

MINI PROJECT

Subject:* *Principle of Database Management

Lecturer: Assoc, Prof. Nguyen Thi Thuy Loan

Members:
Phan Nhật Tân- ITITIU19050
Nguyễn Phước Vĩnh Khang - ITITIU19017
Nguyễn Anh Tuấn - ITITIU19061
Trần Đức Ánh- ITITIU19079
Lê Quang Huy- ITITIU19015

Semester 2, 2021 - 2022

ABSTRACT

This mini-project aims to provide solutions for the database of applications. By doing this mini-project, we can learn about creating a database, ERD model, Relational model, Normalization….This mini-project includes input/output data, update data, Connect database with java platform,......

Table of Content

Part 1
INTRODUCTION

1.1. Overview

Data is essential information for every company . A database is an organized collection of structured information, data, or typically stored electronically in a computer system. SQL is a programming language used by nearly all relational databases to query, manipulate, and define data, and to provide access control (1).My Group chose the topic: Attendance Management System which applies to some software used for school, company, …. In particularly, this database is designed for schools

1.2. General Requirement

1.2.1. Introduction

The absence of lectures in college students has been a big concern since some students pay little attention to their lecturers or professors have problems with their teaching techniques. Most professors prefer to call the roll during class time in order to maintain the attendance rate, and several student attendance systems have been devised to assist them in doing so.The administrators will get a table (an Excel file) containing all the students and courses' information at the beginning of the term, and import it into database with the help of importing module. After that, if any teacher opens the program installed on the computer in the classroom, it will provide a table containing each student's name, ID and a checkbox for marking the absence.

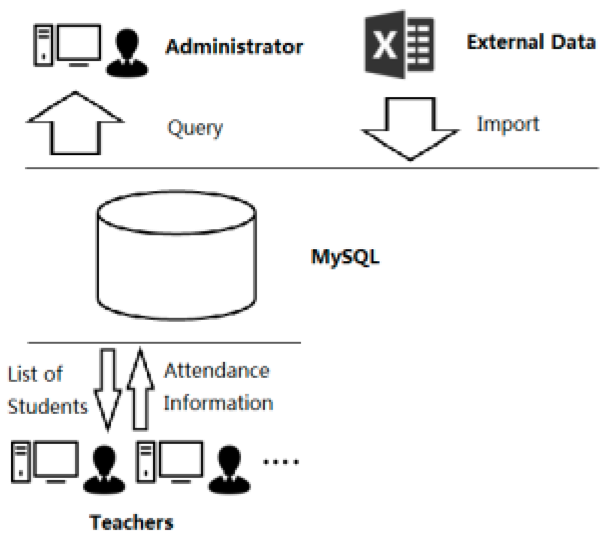


fig 1. Attendance System's process (Source: Google.com)

1.2.3. Project proposals

we list six proposals in following:

1. Discuss about some realistic constraint (eg. No duplicate students in a class, a student can miss 3 lectures…): primary key, foreign key, check constraint, and not null for the tables or attributes.
2. Design the database following an E/R approach, then we also double check for the normalization process to modify the collections of tables that are in higher normal forms.
3. Use MS SQL server to create the normalized tables.
4. Populate the database by using SQL server statements.
5. Using the JDBC library to connect the database with the java platform, then create the winform to demo the efficiencies of the database.

1.3. `Participation

Task	Tân	Huy	Tuấn	Khang	Ánh
Basic Design	x		x		
ER Diagram		x		x	
Relational Model		x		x	x

Modify the Diagrams	x	x	x	x	x
Implement the database	x	x			
Connect database and GUI			x		x
Reports			x	x	

Part 2
DATABASE DIAGRAMS

1. Basic Design

In General, There are three objects for this database: Student, Teacher, Course. From these basic objects, we can develop a database with more than three tables which approach the higher normal form such as Boyce- Codd normal forms. In basic, we will list some tables and their attribute:

Table	Columns	Note
Student	- <u>StudentID</u>- nvarchar(11) - StuName- nvarchar(50) - Gender- nvarchar(10) - DoB- date - Email- nvarchar(50) - Department- nvarchar(50) - classID - nvarchar(11)	Storing the information of the students in the university.
Teacher	- <u>TeacherID</u>- nvarchar(11) - TeacherName- nvarchar(50)	Storing the information of all the lectures.
Class	- <u>classID</u>- nvarchar(11) - className- nvarchar(50) - <u>TeacherID</u>-nvarchar(11) - RoomID- nvarchar(15) - StartWeek- int - EndWeek- int - Class_Begin- int - Class_End- int -DayofWeek- nvarchar(20)	Storing detailed information of all the classes.
Absence	- <u>classID</u>- nvarchar(11) - <u>StudentID</u>- nvarchar(11) - AbsenceDate- date	Storing student record of absence

Table 1: Tables and their attributes in basic

In order to improve performance, some constraints were used to keep spread-out data consistent, any invalid insert or update operation would be rejected.

2. ER diagram:

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research(2). From the basic designs, an ER diagram was created.

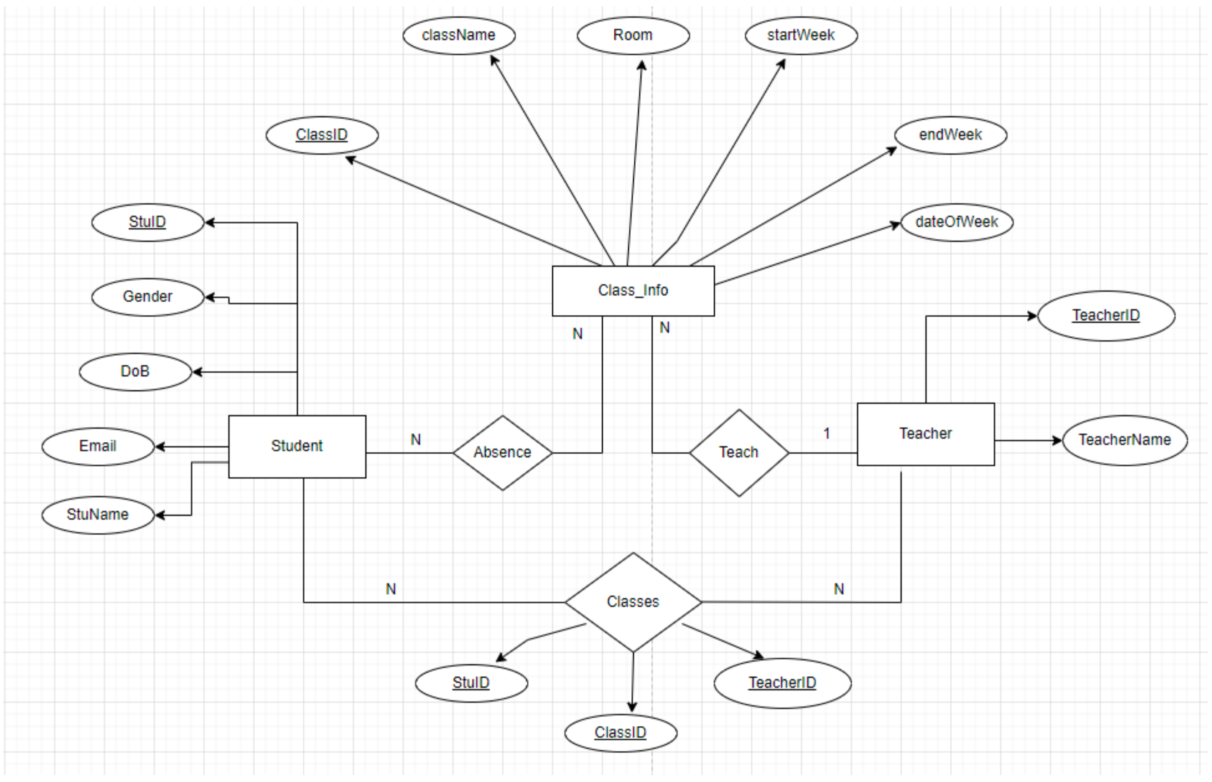


fig 2. ER Diagram for attendance Management system

- The 'Absence' relationship used to keep track the student who absences a class. The class can have many absent students and the student can miss many classes. Therefore, this is a many to many relationship.
- The 'Teach' relationship is the 1 to many. a teacher can teach many classes and one class has only one lecturer.

- The ‘Classes’ relationship can be used to return the list of students in a class. A teacher can have many students and a student has many teachers. Hence, This is a many to many relationship.

3. Relational Model

The ER diagram provides us with basic entity-relationship. But, It isn't particularly thorough because it doesn't demonstrate how to establish a relationship between entity sets when building relations or tables (3). Before we implement our database in the SQL server, we create another model that represents all possible relations within the database. In particular, we must convert the ERD to a relational model.

The primary keys of each individual entity of this database is as followed (table - Key):

+ Student - studentID
+ Teacher - TeacherID
+ Class_Info - classID

According the explanation in the previous part:

+ The ‘Absence’ is a ‘many to many’ relationship. Therefore, an Absence entity which takes primary keys of two tables Student and Class_Info and transfers them to foreign keys is created. In addition, This entity contains the student’s absence date. Because a student can be absent many times, we will set the primary key for absenceDate. Hence, The primary key of this entity is (classID, StudentID, absenceDate).
+ The ‘Classes’ is a ‘many to many’ relationship. Hence, we establish a new ‘Class ’ schema and add as a foreign key all of the primary key attributes of two tables Student and Teacher. In the ‘Class’ schema, we add the classID attribute. The primary key of this schema is a set of (classID, StuID,TeacherID). (Primary key contains classID because a teacher instructs many classes and students can join two or many classes of one lecturer.)

+ The ‘Teach’ is a ‘1 to many’ relationship. Therefore, we just establish a new schema (Teacher and Class_Info) containing the primary keys of the entities that formed it and its own attributes.

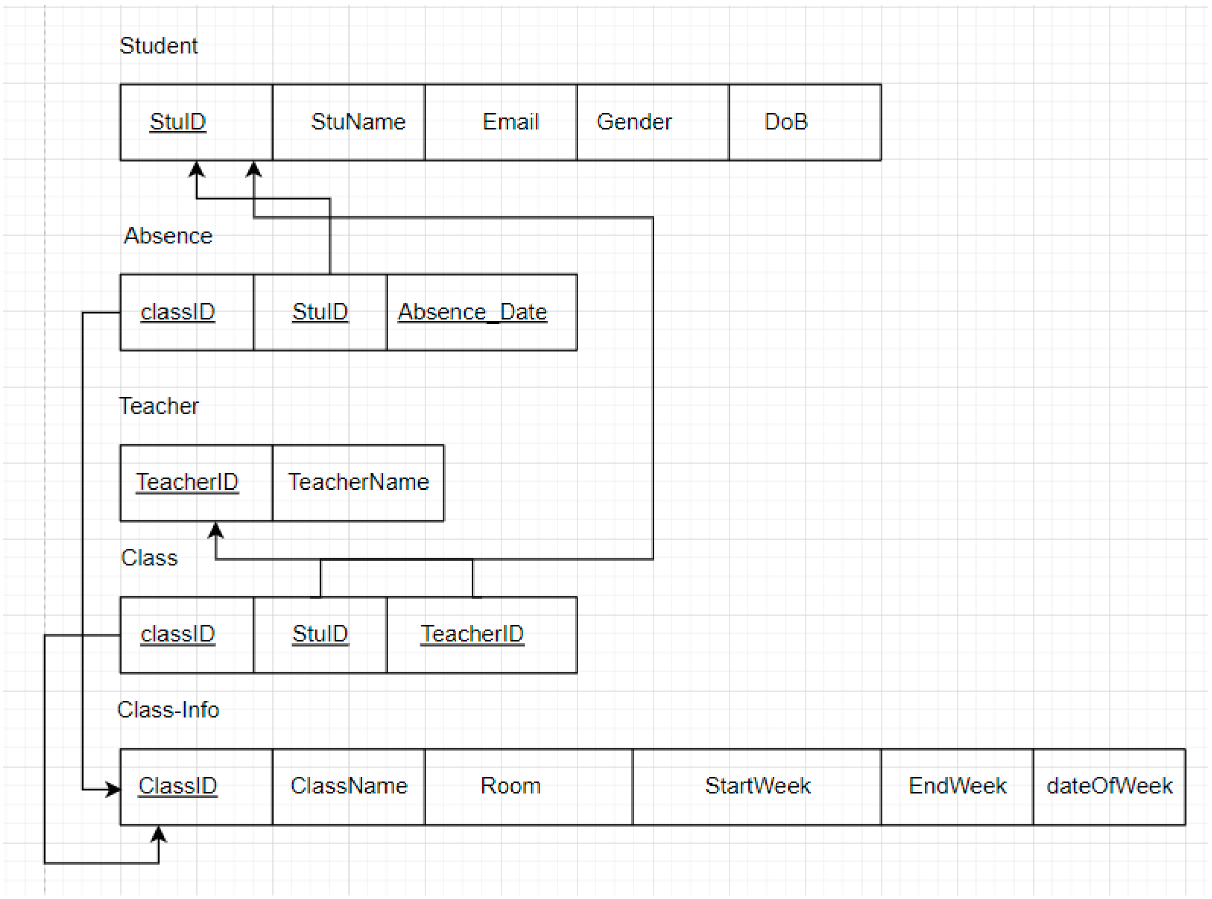


fig 3: Relational Model for attendance management System

<u>Note:</u> The attributes with an underline are the primary key for that table.

From the relational model for this database, our group will design tables and their attributes in detail.

Table	Columns	Note
Student	- <u>StuID</u>- nvarchar(11) - StuName- nvarchar(50) - Email- nvarchar(50) - Gender- nvarchar(10) - DoB- date	Storing the information of the students in the university.

Teacher	- TeacherID- nvarchar(11) - TeacherName- nvarchar(50)	Storing the information of all the lectures.
Class	- classID- nvarchar(11) - StuID- nvarchar(11) - TeacherID-nvarchar(11)	Storing basic information of lecturers.
Class-Info	- classID- nvarchar(11) - className- nvarchar(50) - Room- nvarchar(15) - StartWeek- int - EndWeek- int - DateofWeek - nvarchar(20)	Storing detailed information of a class
Absence	- classID- nvarchar(11) - StudentID- nvarchar(11) - AbsenceDate- date	Storing student record of absence

Table 2: Tables and their Attributes in detail.

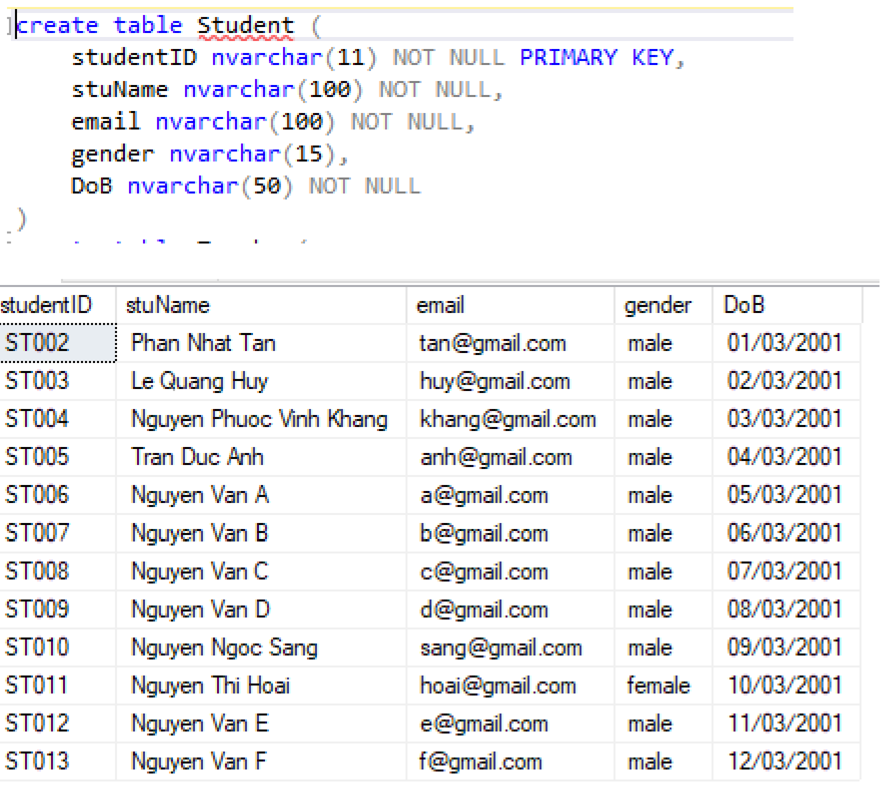
Part 3

Implementation

1. SQL Implementation

The database for attendance management system has 5 tables:

Student:

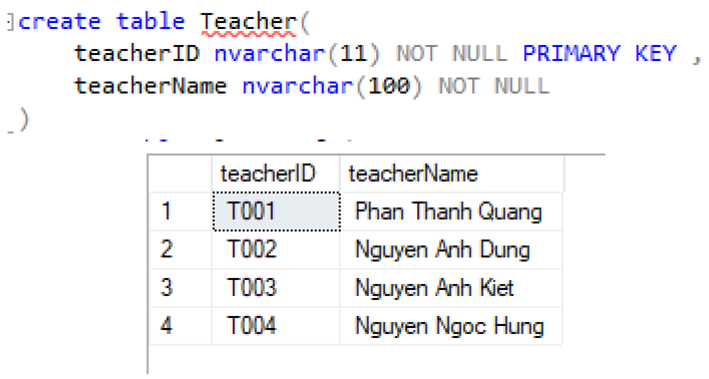


```
create table Student (
    studentID nvarchar(11) NOT NULL PRIMARY KEY,
    stuName nvarchar(100) NOT NULL,
    email nvarchar(100) NOT NULL,
    gender nvarchar(15),
    DoB nvarchar(50) NOT NULL
)
```

	studentID	stuName	email	gender	DoB
1	ST002	Phan Nhat Tan	tan@gmail.com	male	01/03/2001
2	ST003	Le Quang Huy	huy@gmail.com	male	02/03/2001
3	ST004	Nguyen Phuoc Vinh Khang	khang@gmail.com	male	03/03/2001
4	ST005	Tran Duc Anh	anh@gmail.com	male	04/03/2001
5	ST006	Nguyen Van A	a@gmail.com	male	05/03/2001
6	ST007	Nguyen Van B	b@gmail.com	male	06/03/2001
7	ST008	Nguyen Van C	c@gmail.com	male	07/03/2001
8	ST009	Nguyen Van D	d@gmail.com	male	08/03/2001
9	ST010	Nguyen Ngoc Sang	sang@gmail.com	male	09/03/2001
10	ST011	Nguyen Thi Hoai	hoai@gmail.com	female	10/03/2001
11	ST012	Nguyen Van E	e@gmail.com	male	11/03/2001
12	ST013	Nguyen Van F	f@gmail.com	male	12/03/2001

fig 4: SQL code of Student Table.

Teacher:

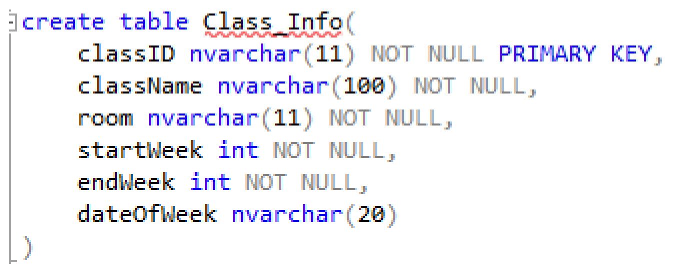


```
create table Teacher(
    teacherID nvarchar(11) NOT NULL PRIMARY KEY ,
    teacherName nvarchar(100) NOT NULL
)
```

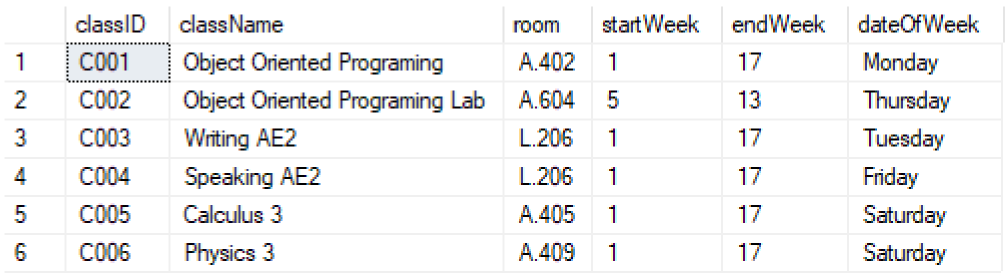
	teacherID	teacherName
1	T001	Phan Thanh Quang
2	T002	Nguyen Anh Dung
3	T003	Nguyen Anh Kiet
4	T004	Nguyen Ngoc Hung

fig 5: SQL code of Teacher Table.

Class_Info:



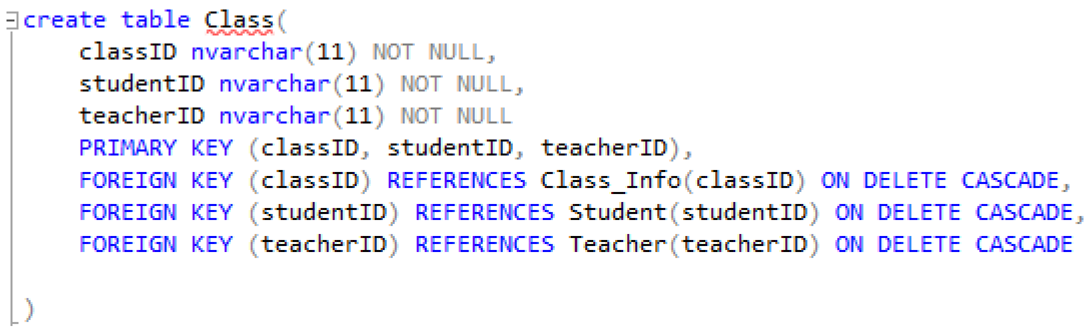
```
create table Class_Info(
    classID nvarchar(11) NOT NULL PRIMARY KEY,
    className nvarchar(100) NOT NULL,
    room nvarchar(11) NOT NULL,
    startWeek int NOT NULL,
    endWeek int NOT NULL,
    dateOfWeek nvarchar(20)
)
```



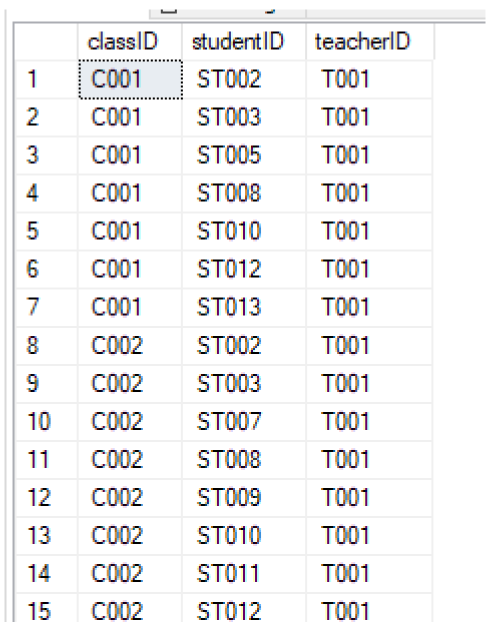
	classID	className	room	startWeek	endWeek	dateOfWeek
1	C001	Object Oriented Programing	A.402	1	17	Monday
2	C002	Object Oriented Programing Lab	A.604	5	13	Thursday
3	C003	Writing AE2	L.206	1	17	Tuesday
4	C004	Speaking AE2	L.206	1	17	Friday
5	C005	Calculus 3	A.405	1	17	Saturday
6	C006	Physics 3	A.409	1	17	Saturday

fig 6: SQL code of Class_Info Table.

Class:



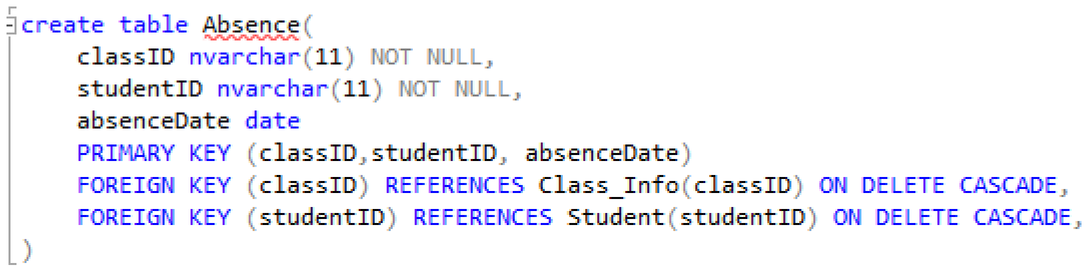
```
create table Class(
    classID nvarchar(11) NOT NULL,
    studentID nvarchar(11) NOT NULL,
    teacherID nvarchar(11) NOT NULL
    PRIMARY KEY (classID, studentID, teacherID),
    FOREIGN KEY (classID) REFERENCES Class_Info(classID) ON DELETE CASCADE,
    FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE,
    FOREIGN KEY (teacherID) REFERENCES Teacher(teacherID) ON DELETE CASCADE

)
```



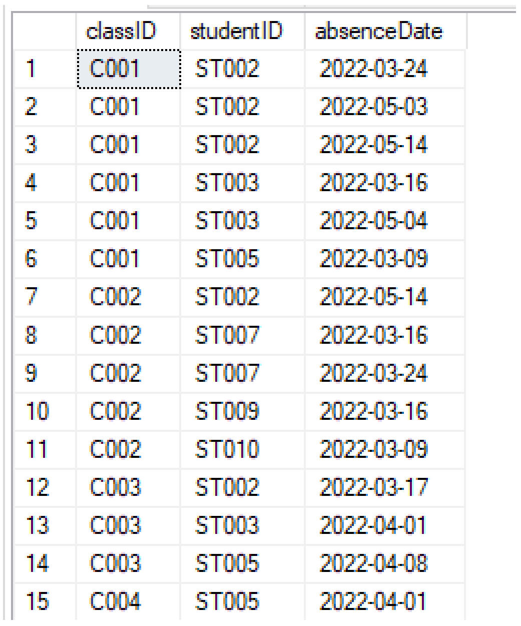
	classID	studentID	teacherID
1	C001	ST002	T001
2	C001	ST003	T001
3	C001	ST005	T001
4	C001	ST008	T001
5	C001	ST010	T001
6	C001	ST012	T001
7	C001	ST013	T001
8	C002	ST002	T001
9	C002	ST003	T001
10	C002	ST007	T001
11	C002	ST008	T001
12	C002	ST009	T001
13	C002	ST010	T001
14	C002	ST011	T001
15	C002	ST012	T001

fig 7: SQL code of Class Table.

Absence:



```
create table Absence(
    classID nvarchar(11) NOT NULL,
    studentID nvarchar(11) NOT NULL,
    absenceDate date
    PRIMARY KEY (classID,studentID, absenceDate)
    FOREIGN KEY (classID) REFERENCES Class_Info(classID) ON DELETE CASCADE,
    FOREIGN KEY (studentID) REFERENCES Student(studentID) ON DELETE CASCADE,
)
```



	classID	studentID	absenceDate
1	C001	ST002	2022-03-24
2	C001	ST002	2022-05-03
3	C001	ST002	2022-05-14
4	C001	ST003	2022-03-16
5	C001	ST003	2022-05-04
6	C001	ST005	2022-03-09
7	C002	ST002	2022-05-14
8	C002	ST007	2022-03-16
9	C002	ST007	2022-03-24
10	C002	ST009	2022-03-16
11	C002	ST010	2022-03-09
12	C003	ST002	2022-03-17
13	C003	ST003	2022-04-01
14	C003	ST005	2022-04-08
15	C004	ST005	2022-04-01

fig 8: SQL code of Absence Table.

Note: ‘ON DELETE CASCADE’ : When we create a foreign key using this option, it deletes the referencing rows in the child table when the referenced row is deleted in the parent table which has a primary key (“SQLshack”, Delete cascade and Update Cascade in SQL server foreign key).

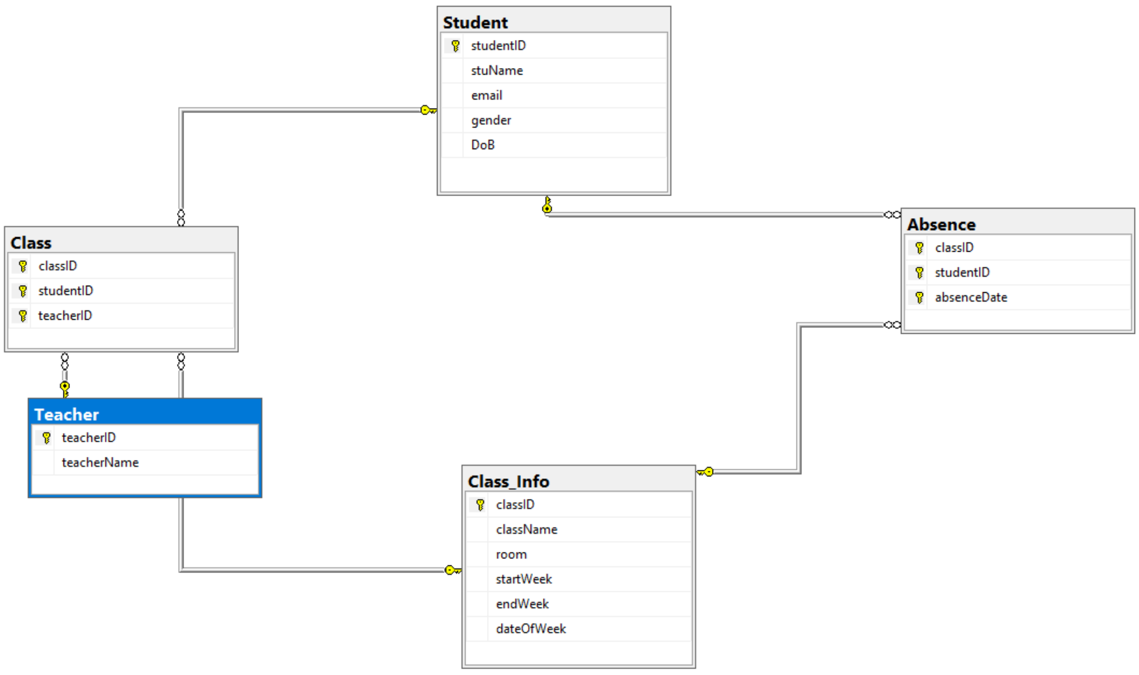
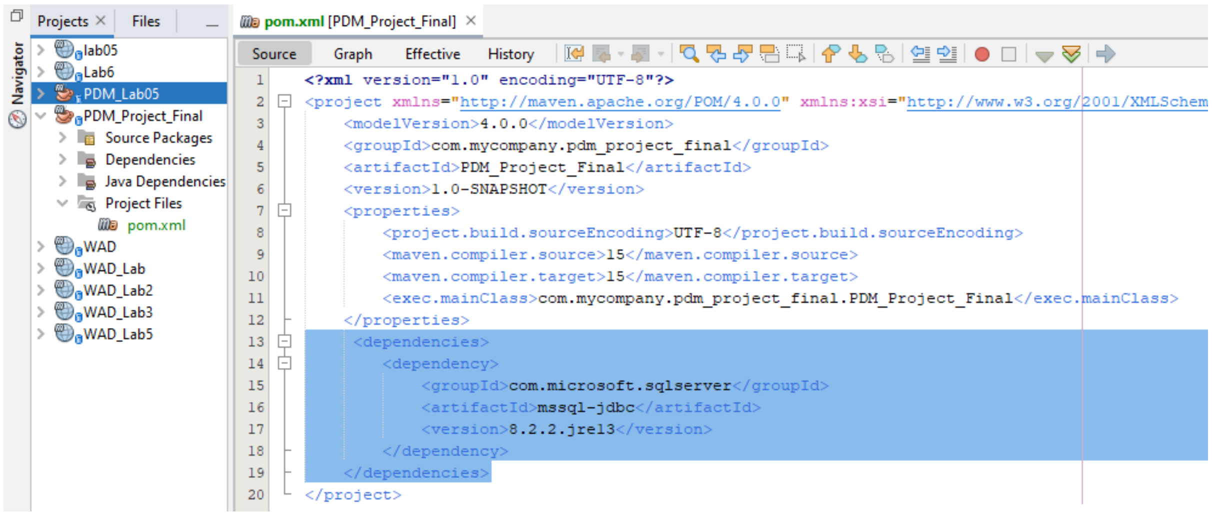


fig 9: Relational Model for attendance management System in SQL- server.

2. Java Implementation

3.2.1. Connect the database to Java

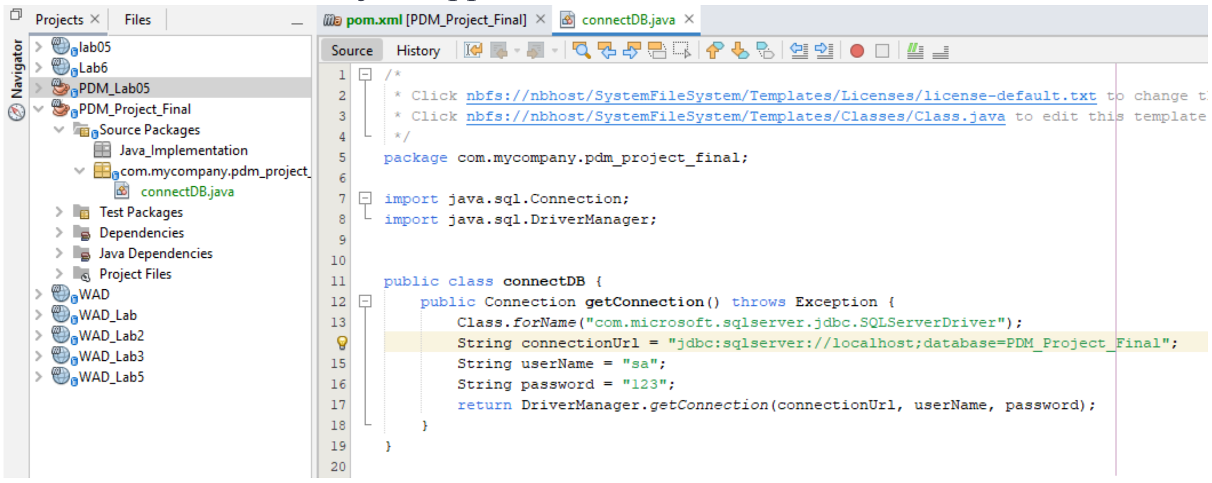
We will use the Java programming language in Netbeans Platform for this project. The Version of Netbeans is 13. Depending on the version of Netbeans and type of application, we will connect by using the libraries folder or the Dependencies. In our Netbeans IDE, we cannot figure out the libraries folder (because it is maven project type). Hence, we will use the 'dependencies' by modifying the 'pom.xml' in the 'Project Files' folder.



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchem
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany.pdm_project_final</groupId>
    <artifactId>PDM_Project_Final</artifactId>
    <version>1.0-SNAPSHOT</version>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>15</maven.compiler.source>
        <maven.compiler.target>15</maven.compiler.target>
        <exec.mainClass>com.mycompany.pdm_project_final.PDM_Project_Final</exec.mainClass>
    </properties>
     <dependencies>
        <dependency>
            <groupId>com.microsoft.sqlserver</groupId>
            <artifactId>mssql-jdbc</artifactId>
            <version>8.2.2.jre13</version>
        </dependency>
    </dependencies>
</project>
```

fig 10: Adding the dependencies to 'pom.xml' file.

After adding this 'dependencies', we will create a class for the connection between this database and our java application.



```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change t
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.pdm_project_final;

import java.sql.Connection;
import java.sql.DriverManager;


public class connectDB {
    public Connection getConnection() throws Exception {
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        String connectionUrl = "jdbc:sqlserver://localhost;database=PDM_Project_Final";
        String userName = "sa";
        String password = "123";
        return DriverManager.getConnection(connectionUrl, userName, password);
    }
}

```

fig 11:Java Class for the connection database.

After these steps, we finish the task of connecting the database to our java application.

3.2.2. Java for GUI

We use the drag-drop function of java.swing to create the form of this project. The Netbeans will generate GUI's code automatically. We just define some functions for buttons and how the data to be loaded in our GUI.

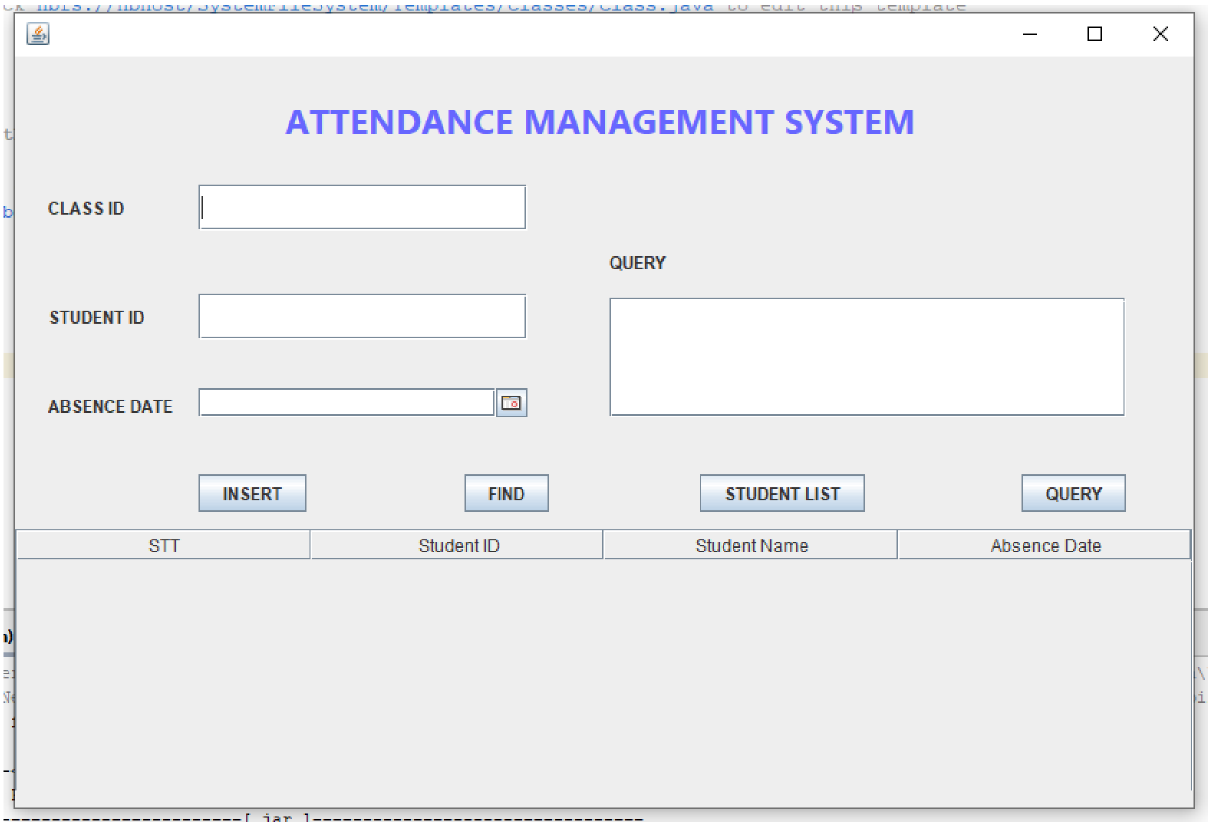
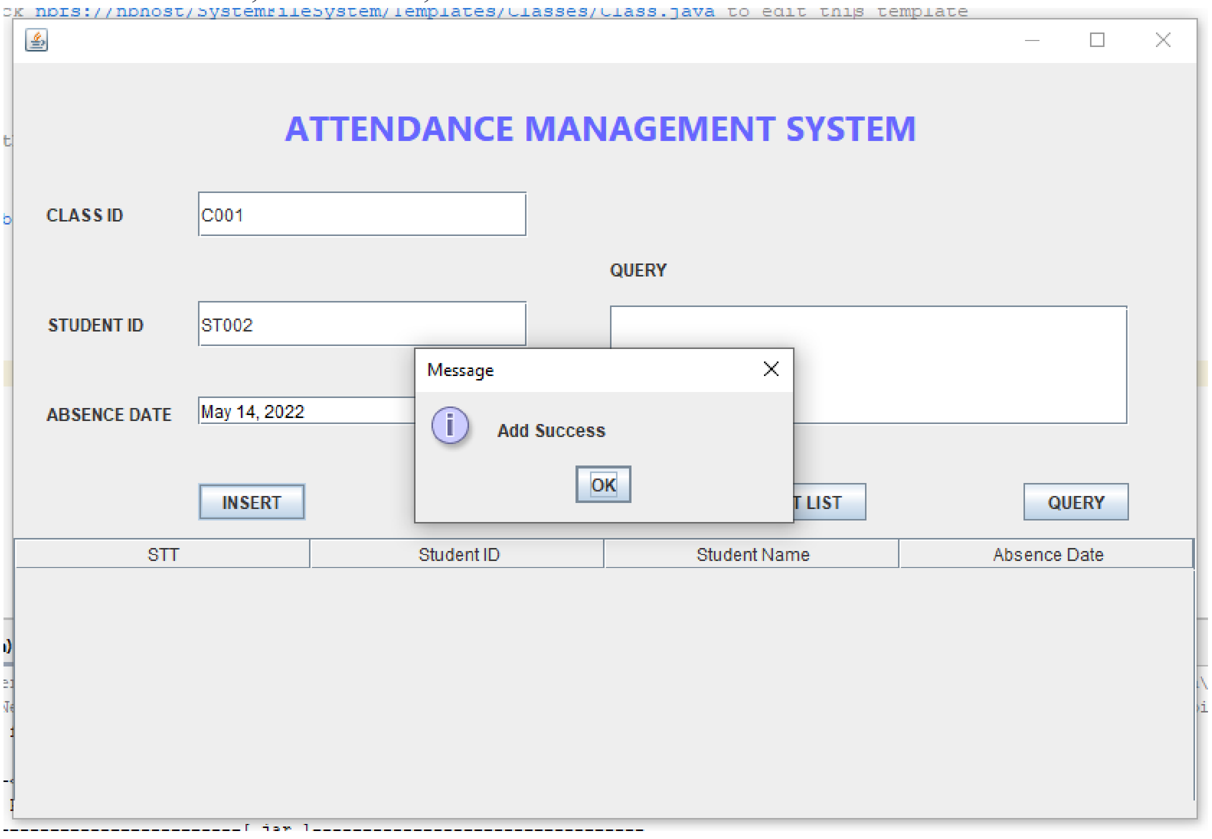


fig 12: User Interface for Querying data.

There are four function which help users query data easily:

+ INSERT: insert the absence student into the Absence Table. Users will enter the classID, StudentID, Absence date to add a record.


ATTENDANCE MANAGEMENT SYSTEM

CLASS ID: C001

STUDENT ID: ST002

ABSENCE DATE: May 14, 2022

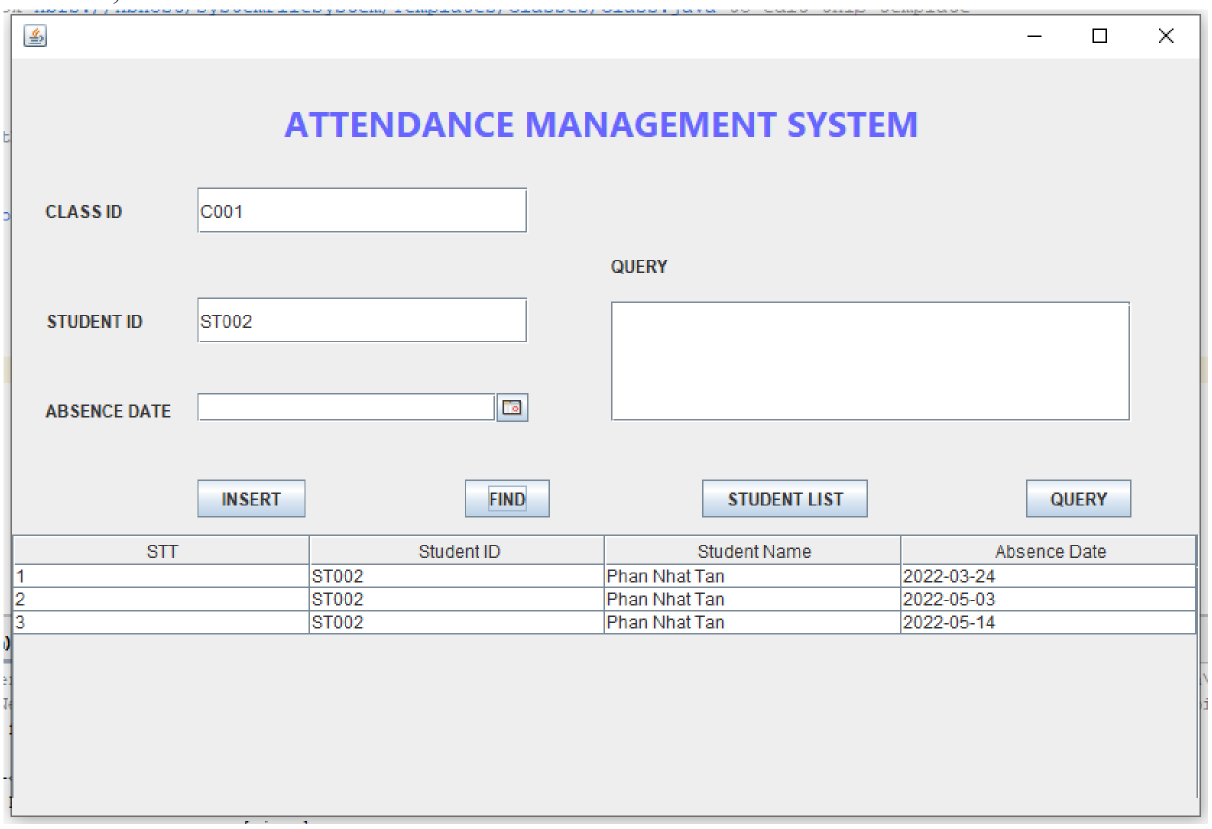
QUERY

Message: Add Success

STT	Student ID	Student Name	Absence Date

fig 13: Insert Function.

+ <u>FIND:</u> to return the absence date of a student. Users will input the classID, studentID.


ATTENDANCE MANAGEMENT SYSTEM

CLASS ID: C001

STUDENT ID: ST002

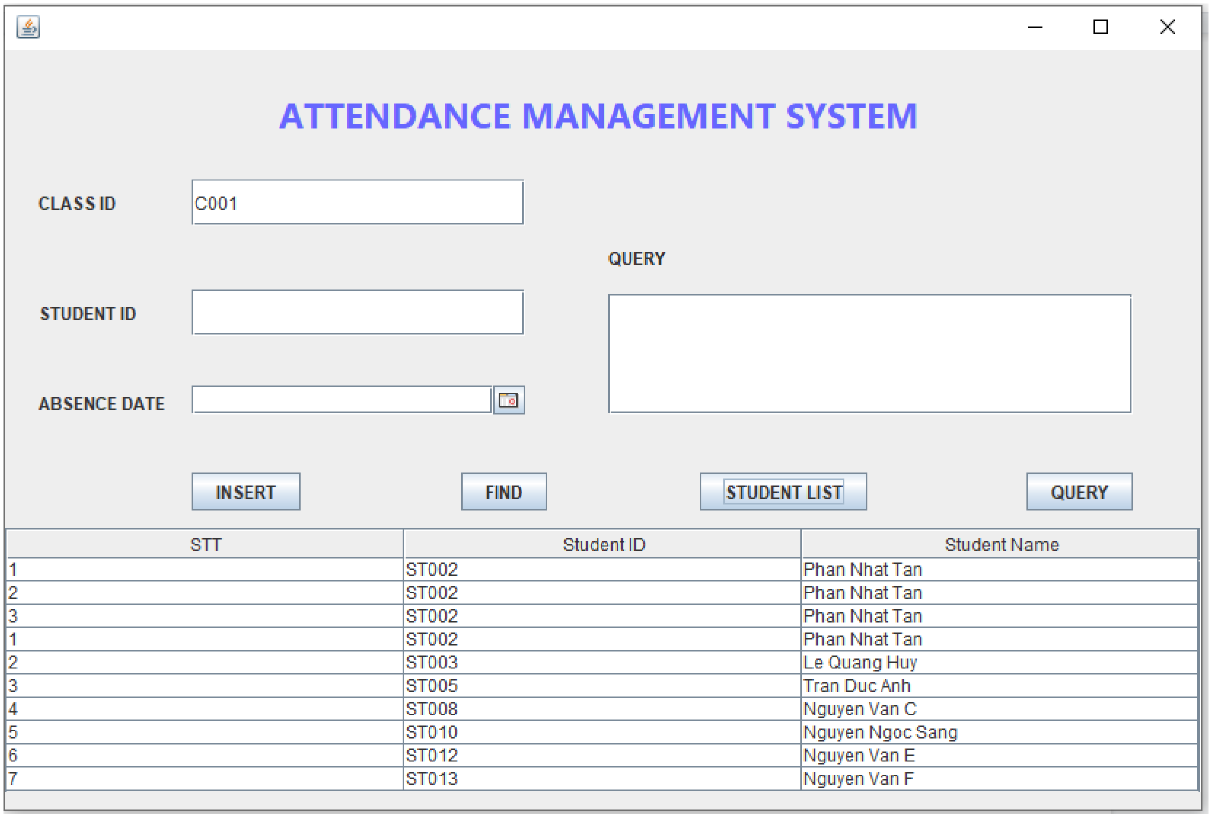
ABSENCE DATE:

QUERY

STT	Student ID	Student Name	Absence Date
1	ST002	Phan Nhat Tan	2022-03-24
2	ST002	Phan Nhat Tan	2022-05-03
3	ST002	Phan Nhat Tan	2022-05-14

fig 14: Search Function.

+ Student List: return the student list of a class.

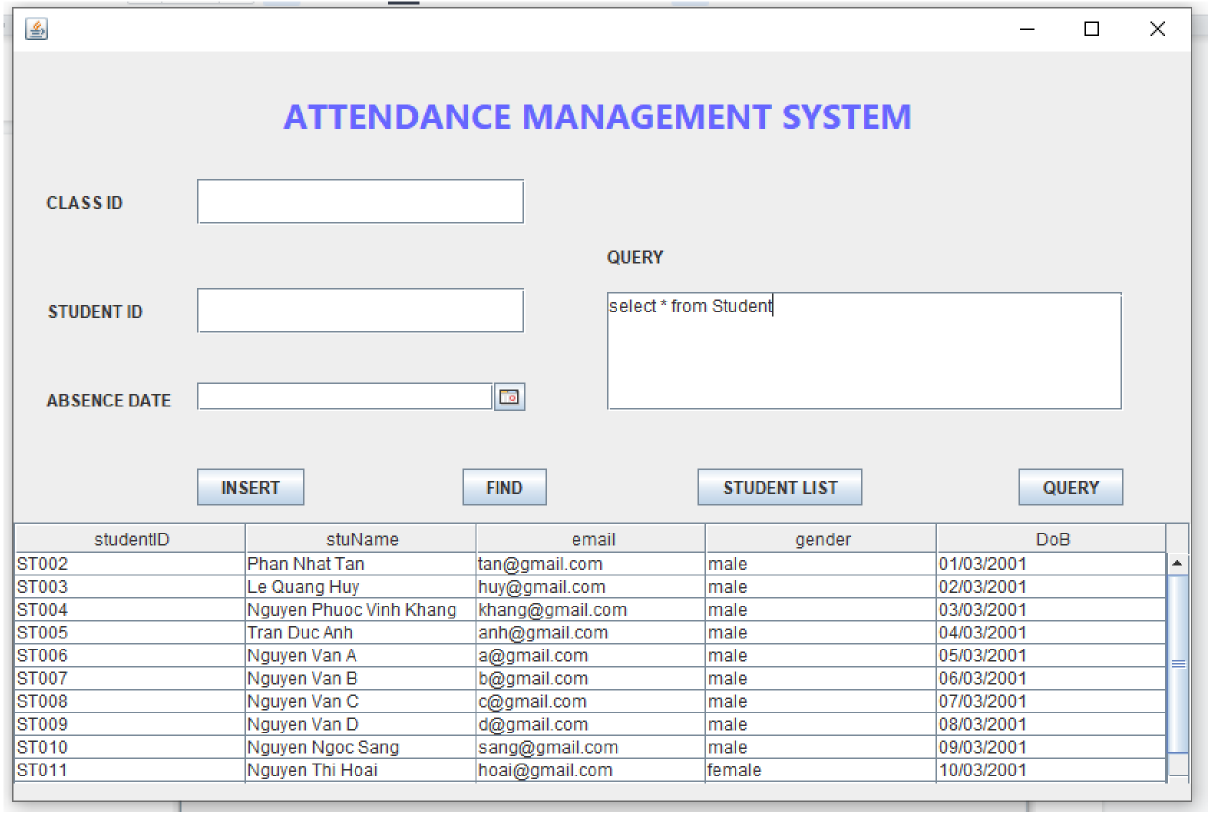


CLASS ID: C001

STT	Student ID	Student Name
1	ST002	Phan Nhat Tan
2	ST002	Phan Nhat Tan
3	ST002	Phan Nhat Tan
1	ST002	Phan Nhat Tan
2	ST003	Le Quang Huy
3	ST005	Tran Duc Anh
4	ST008	Nguyen Van C
5	ST010	Nguyen Ngoc Sang
6	ST012	Nguyen Van E
7	ST013	Nguyen Van F

fig 15: ‘Student List’ Function.

+ Query: In advance , users can input a SQL sentence to get data from the database.



QUERY: select * from Student

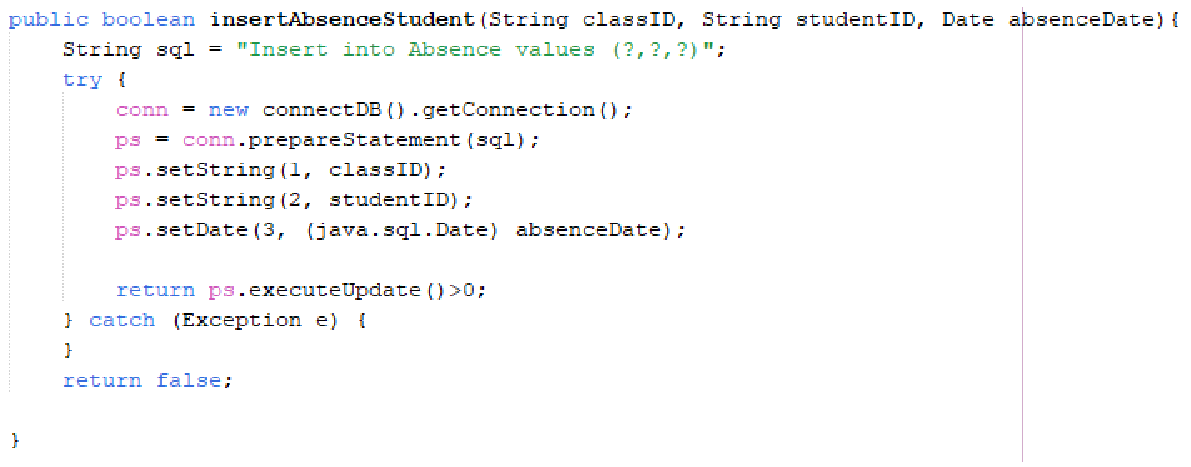
studentID	stuName	email	gender	DoB
ST002	Phan Nhat Tan	tan@gmail.com	male	01/03/2001
ST003	Le Quang Huy	huy@gmail.com	male	02/03/2001
ST004	Nguyen Phuoc Vinh Khang	khang@gmail.com	male	03/03/2001
ST005	Tran Duc Anh	anh@gmail.com	male	04/03/2001
ST006	Nguyen Van A	a@gmail.com	male	05/03/2001
ST007	Nguyen Van B	b@gmail.com	male	06/03/2001
ST008	Nguyen Van C	c@gmail.com	male	07/03/2001
ST009	Nguyen Van D	d@gmail.com	male	08/03/2001
ST010	Nguyen Ngoc Sang	sang@gmail.com	male	09/03/2001
ST011	Nguyen Thi Hoai	hoai@gmail.com	female	10/03/2001

fig 16: Query Function.

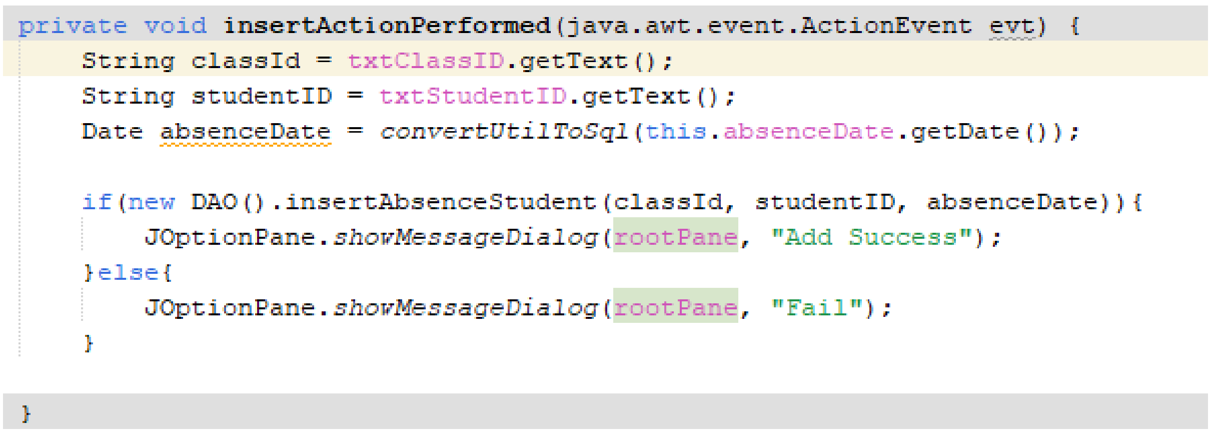
In this application, we have two main classes:

+ DAO.java: containing some functions to get data from the database.
+ View.java: containing code for GUI and functions to define the method for buttons.

Here is some important code to define the button's function and populate data to the table.

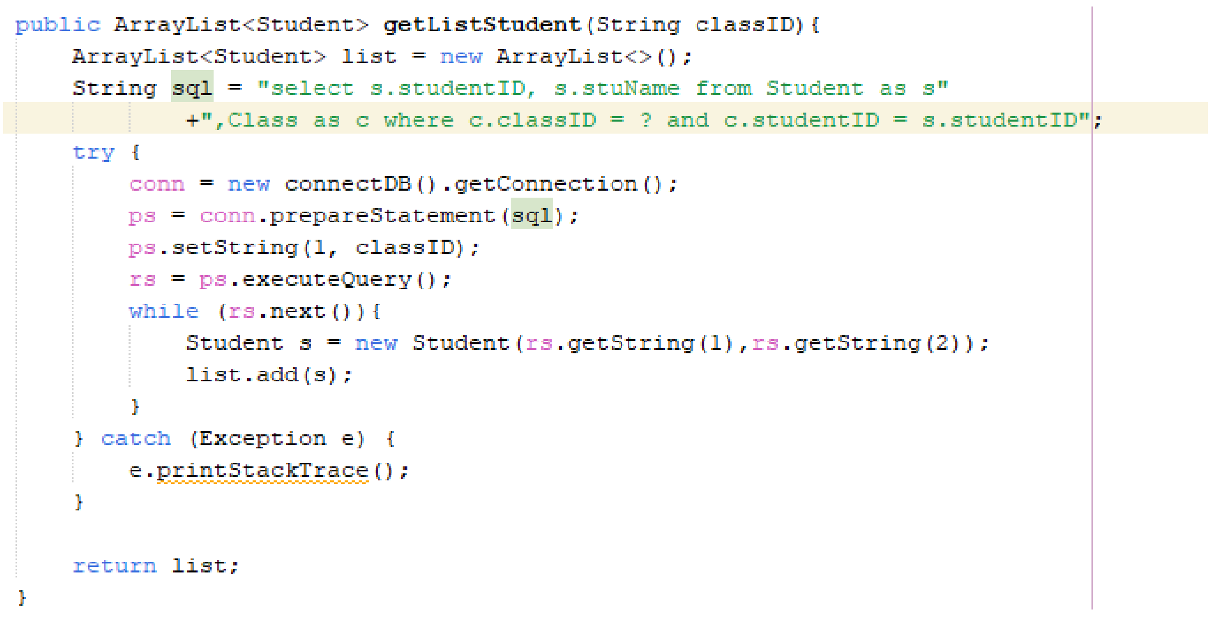


```
public boolean insertAbsenceStudent(String classID, String studentID, Date absenceDate){
    String sql = "Insert into Absence values (?,?,?)";
    try {
        conn = new connectDB().getConnection();
        ps = conn.prepareStatement(sql);
        ps.setString(1, classID);
        ps.setString(2, studentID);
        ps.setDate(3, (java.sql.Date) absenceDate);

        return ps.executeUpdate()>0;
    } catch (Exception e) {
    }
    return false;

}
```

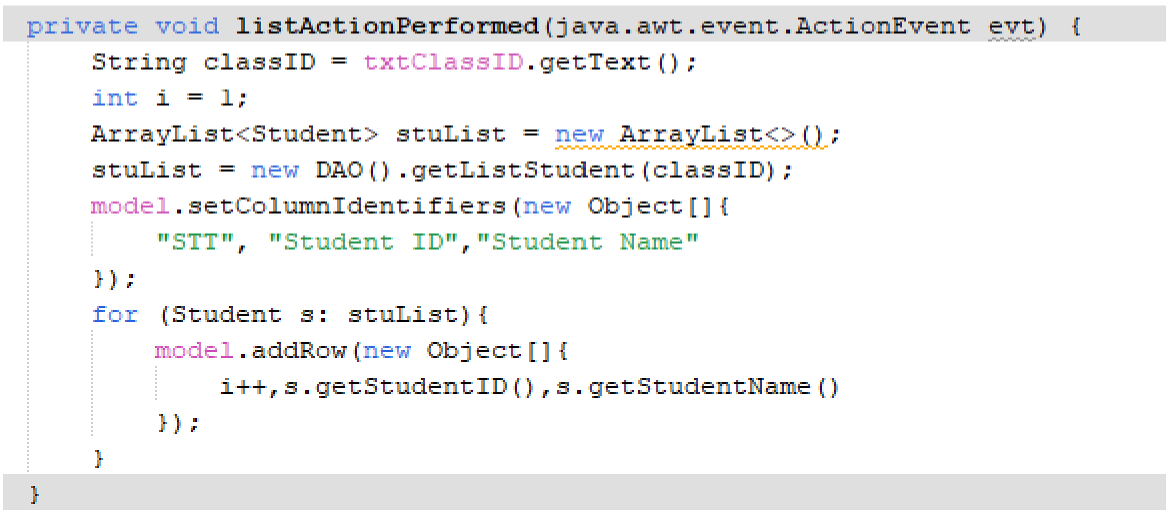


```
private void insertActionPerformed(java.awt.event.ActionEvent evt) {
    String classId = txtClassID.getText();
    String studentID = txtStudentID.getText();
    Date absenceDate = convertUtilToSql(this.absenceDate.getDate());

    if(new DAO().insertAbsenceStudent(classId, studentID, absenceDate)){
        JOptionPane.showMessageDialog(rootPane, "Add Success");
    }else{
        JOptionPane.showMessageDialog(rootPane, "Fail");
    }

}
```

fig 17: Code for Insert function in DAO.java class and View.java class.

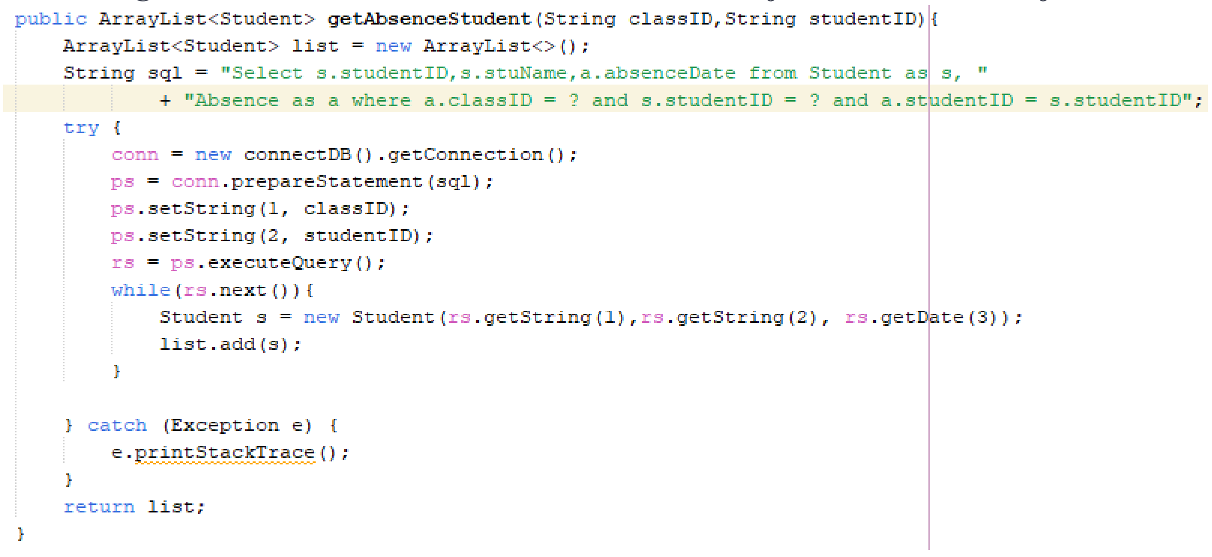


```java
public ArrayList<Student> getListStudent(String classID){
    ArrayList<Student> list = new ArrayList<>();
    String sql = "select s.studentID, s.stuName from Student as s"
            +",Class as c where c.classID = ? and c.studentID = s.studentID";
    try {
        conn = new connectDB().getConnection();
        ps = conn.prepareStatement(sql);
        ps.setString(1, classID);
        rs = ps.executeQuery();
        while (rs.next()){
            Student s = new Student(rs.getString(1),rs.getString(2));
            list.add(s);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    return list;
}
```

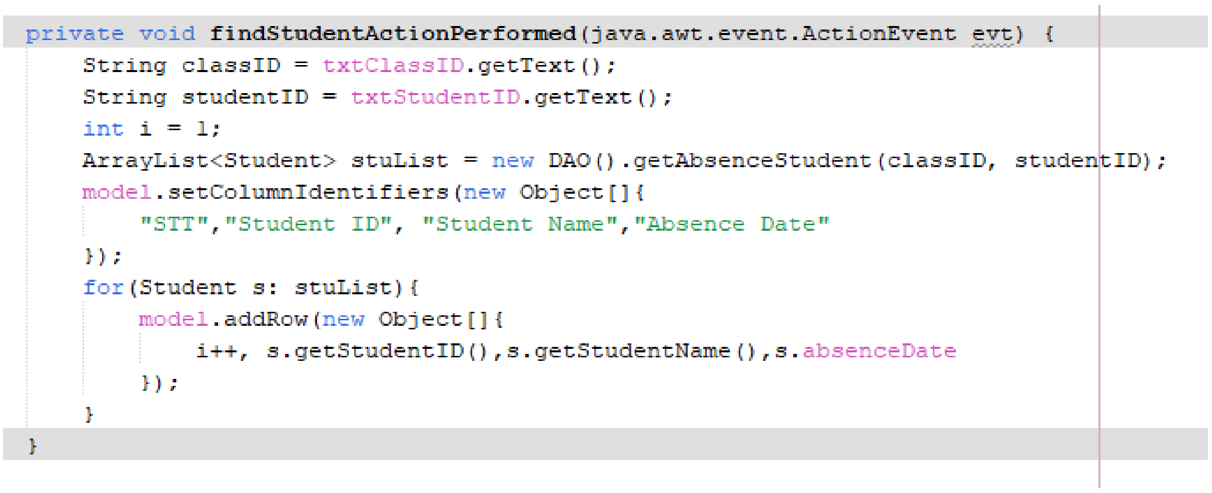


```java
private void listActionPerformed(java.awt.event.ActionEvent evt) {
    String classID = txtClassID.getText();
    int i = 1;
    ArrayList<Student> stuList = new ArrayList<>();
    stuList = new DAO().getListStudent(classID);
    model.setColumnIdentifiers(new Object[]{
        "STT", "Student ID","Student Name"
    });
    for (Student s: stuList){
        model.addRow(new Object[]{
            i++,s.getStudentID(),s.getStudentName()
        });
    }
}
```

fig 18: Code for 'Student List' function in DAO.java class and View.java class.

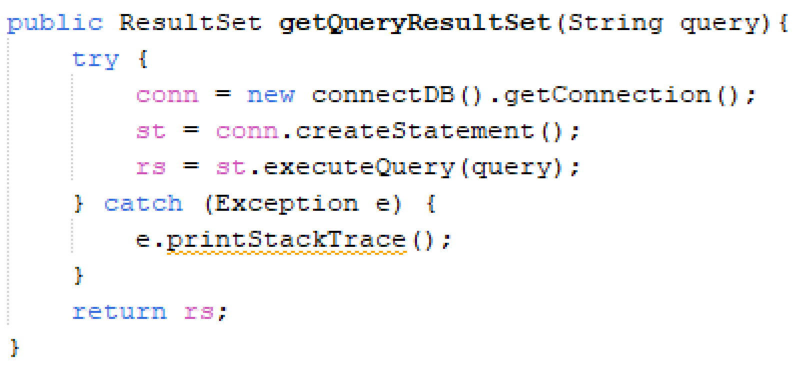


```java
public ArrayList<Student> getAbsenceStudent(String classID,String studentID){
    ArrayList<Student> list = new ArrayList<>();
    String sql = "Select s.studentID,s.stuName,a.absenceDate from Student as s, "
            + "Absence as a where a.classID = ? and s.studentID = ? and a.studentID = s.studentID";
    try {
        conn = new connectDB().getConnection();
        ps = conn.prepareStatement(sql);
        ps.setString(1, classID);
        ps.setString(2, studentID);
        rs = ps.executeQuery();
        while(rs.next()){
            Student s = new Student(rs.getString(1),rs.getString(2), rs.getDate(3));
            list.add(s);
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
    return list;
}
```

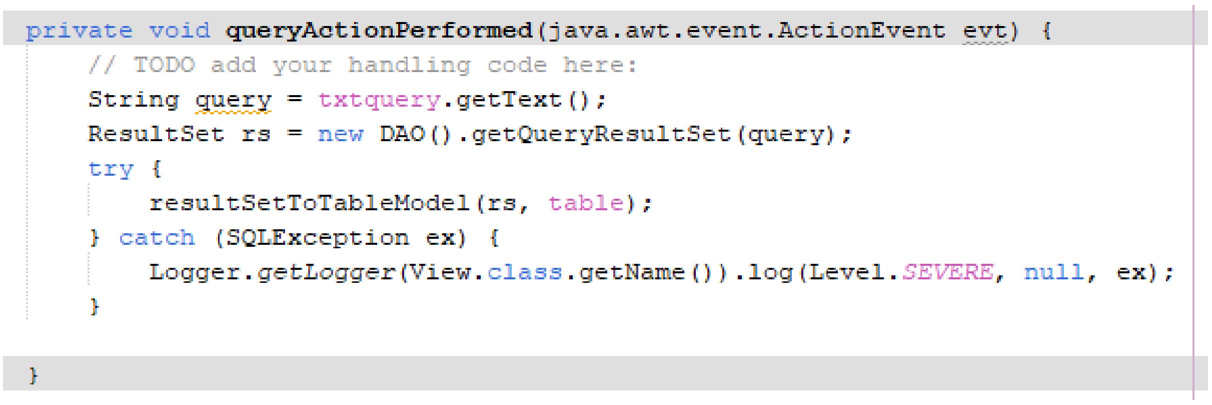


```java
private void findStudentActionPerformed(java.awt.event.ActionEvent evt) {
    String classID = txtClassID.getText();
    String studentID = txtStudentID.getText();
    int i = 1;
    ArrayList<Student> stuList = new DAO().getAbsenceStudent(classID, studentID);
    model.setColumnIdentifiers(new Object[]{
        "STT","Student ID", "Student Name","Absence Date"
    });
    for(Student s: stuList){
        model.addRow(new Object[]{
            i++, s.getStudentID(),s.getStudentName(),s.absenceDate
        });
    }
}
```

fig 19: Code for ‘Find’ function in DAO.java class and View.java class.

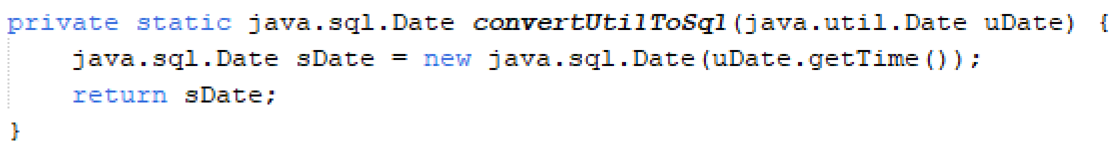


```java
public ResultSet getQueryResultSet(String query){
    try {
        conn = new connectDB().getConnection();
        st = conn.createStatement();
        rs = st.executeQuery(query);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return rs;
}
```

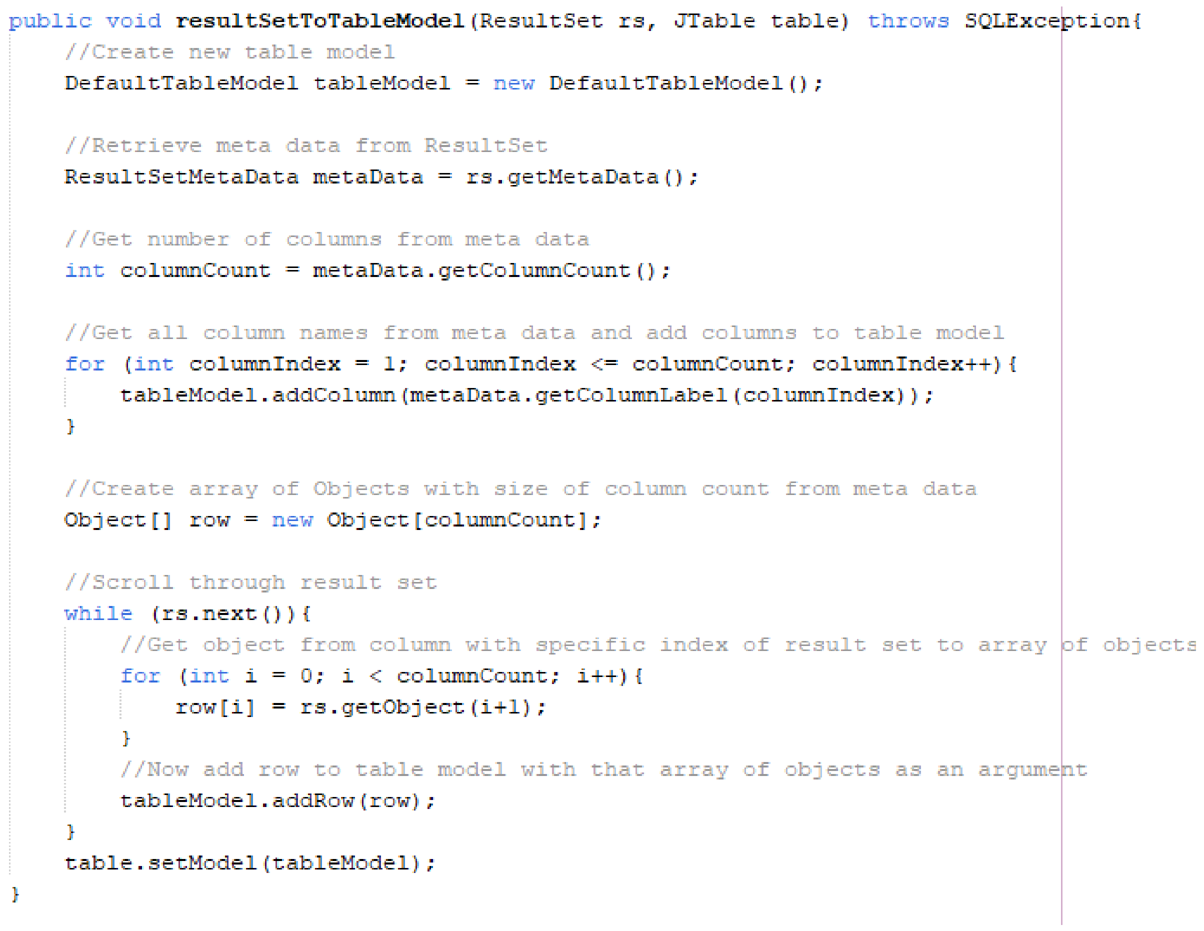


```java
private void queryActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String query = txtquery.getText();
    ResultSet rs = new DAO().getQueryResultSet(query);
    try {
        resultSetToTableModel(rs, table);
    } catch (SQLException ex) {
        Logger.getLogger(View.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```

fig 20: Code for ‘Find’ function in DAO.java class and View.java class.



```java
private static java.sql.Date convertUtilToSql(java.util.Date uDate) {
    java.sql.Date sDate = new java.sql.Date(uDate.getTime());
    return sDate;
}
```



```java
public void resultSetToTableModel(ResultSet rs, JTable table) throws SQLException{
    //Create new table model
    DefaultTableModel tableModel = new DefaultTableModel();

    //Retrieve meta data from ResultSet
    ResultSetMetaData metaData = rs.getMetaData();

    //Get number of columns from meta data
    int columnCount = metaData.getColumnCount();

    //Get all column names from meta data and add columns to table model
    for (int columnIndex = 1; columnIndex <= columnCount; columnIndex++){
        tableModel.addColumn(metaData.getColumnLabel(columnIndex));
    }

    //Create array of Objects with size of column count from meta data
    Object[] row = new Object[columnCount];

    //Scroll through result set
    while (rs.next()){
        //Get object from column with specific index of result set to array of objects
        for (int i = 0; i < columnCount; i++){
            row[i] = rs.getObject(i+1);
        }
        //Now add row to table model with that array of objects as an argument
        tableModel.addRow(row);
    }
    table.setModel(tableModel);
}
```

fig 21: Two functions converting util.date to sql.date and populate data from result set to table .

Part 4
CONCLUSION

In conclusion, our group designed and developed a simple database for attendance management. From this database, we can connect to the java platform and implement some function to populate data from database to GUI. Therefore, we can understand SQL knowledge, converting ERD to Relational Model, normalization of database, or how data to be stored in an application.

Contribution:

Name	Contribution
Tuấn	20%
Tân	20%
Ánh	20%
Khang	20%
Huy	20%

Part 5
REFERENCES

1. What is a database, Oracle, from :
   https://www.oracle.com/database/what-is-database/
2. attendance management, part 2 from:
   https://notesformsc.org/attendance-management-part-2/
3. “What is an ER Diagram”, LucidChart, from:
   https://www.lucidchart.com/pages/er-diagrams#section_5
4. How to convert ER Diagram to relational Database ,Learn Database, from:
   https://www.learndb.com/databases/how-to-convert-er-diagram-to-relational-database
5. Delete cascade and Update Cascade in SQL server foreign key, SQLshack, from:
   https://www.sqlshack.com/delete-cascade-and-update-cascade-in-sql-server-foreign-key/