Principles of Database Management

Midterm Test – 20\_School Management System

Group member:

* Trần Trọng Thức ITITIU18125
* Nguyễn Lê Thanh Tâm ITITIU18110
* Nguyễn Đức Đăng Khôi ITITIU18066

# Question part:

Topic 20 – School Management System Design a database to maintain information about school staff and students satisfying following properties:

- Staff will be having their id, name and classes they are teaching

- Student will be having name, roll no, section, class

- Another table containing section, subject and teacher information

- Next will be containing fee information for students

- One contains salary information for teachers

- Rooms are assigned to classes keeping in mind that there is no time clash of same room or lab, student cannot be entered in more than one section, no student should be there who has not paid fee up to a particular date.

Answer part:

1. Designing an ERD conceptual schema:

The school management system will manage the information of the student and staff in the school. It will contain the data of student, their id, name, the subject, and class that they are enrolled and the staff teaching them.

Requirements:

1. A student has student ID, student name, gender, level and join in one class.
2. A staff has staff ID, staff name, and a salary.
3. A student has one or many tuition fee (ex: one semester - one fee).
4. A staff has one and only one salary.
5. Subjects which have the same name, can have different subject IDs.
6. A student can learn many subjects which have different subject name.
7. A staff can give subject assignments.
8. Students, staffs, subjects and rooms are allocated together to form a class allocation.

Note:

* + **These are entity classes.**
  + These are cardinalities.
  + *These are relationships.*
  + These are attributes.
  + These are primary attributes.

**E-R Diagram includes:**

- A **Student** *joins in* one **Class**, a **Class** *has* many **Student**.

- A **Student** *has* one or many **Student** **Fee**(s), a **Student** Fee *is* *paid* *by* one and only one Student.

- A **Staff** *receives* one **Staff** **Salary**, a **Staff** **Salary** *is* *owned* *by* one and only one **Staff**.

- A **Staff** *gives* one or many **Classroom** **Assignment**(s), a **Classroom** **Assignment** *is* *given* *by* one and only one **Staff**.

- A **Subject** *can* *has* one or many **Classroom** **Assignment**(s), a **Classroom** **Assignment** *is* *belonging* *to* one **Subject**.

- A **Student** *registers* one or many *registrations* *as* **Check** **Student** **Enrollment**(s), a **Check** **Student** **Enrollment** *is* *created* *by* one **Student**.

- A **Subject** *are* *located* *in* one or many registrations as **Check** **Student** **Enrollment**(s), a **Check** **Student** **Enrollment** *has* only one **Subject**.

- In the class allocation list, a **Subject** *is* *taught* *by* one **Staff** , a **Student** *is* *taught* *by* one or many **Staff**(s) (by registering many **Subject**(s)), and a **Staff** *can* *teach* one or many **Subject**(s)/**Student**(s).

**E-R Diagram Schemas:**

- **Student** (student\_ID, student\_name, student\_gender, level).

- **Staff** (staff\_ID, staff\_name)

- **Class** (class\_ID, class\_section).

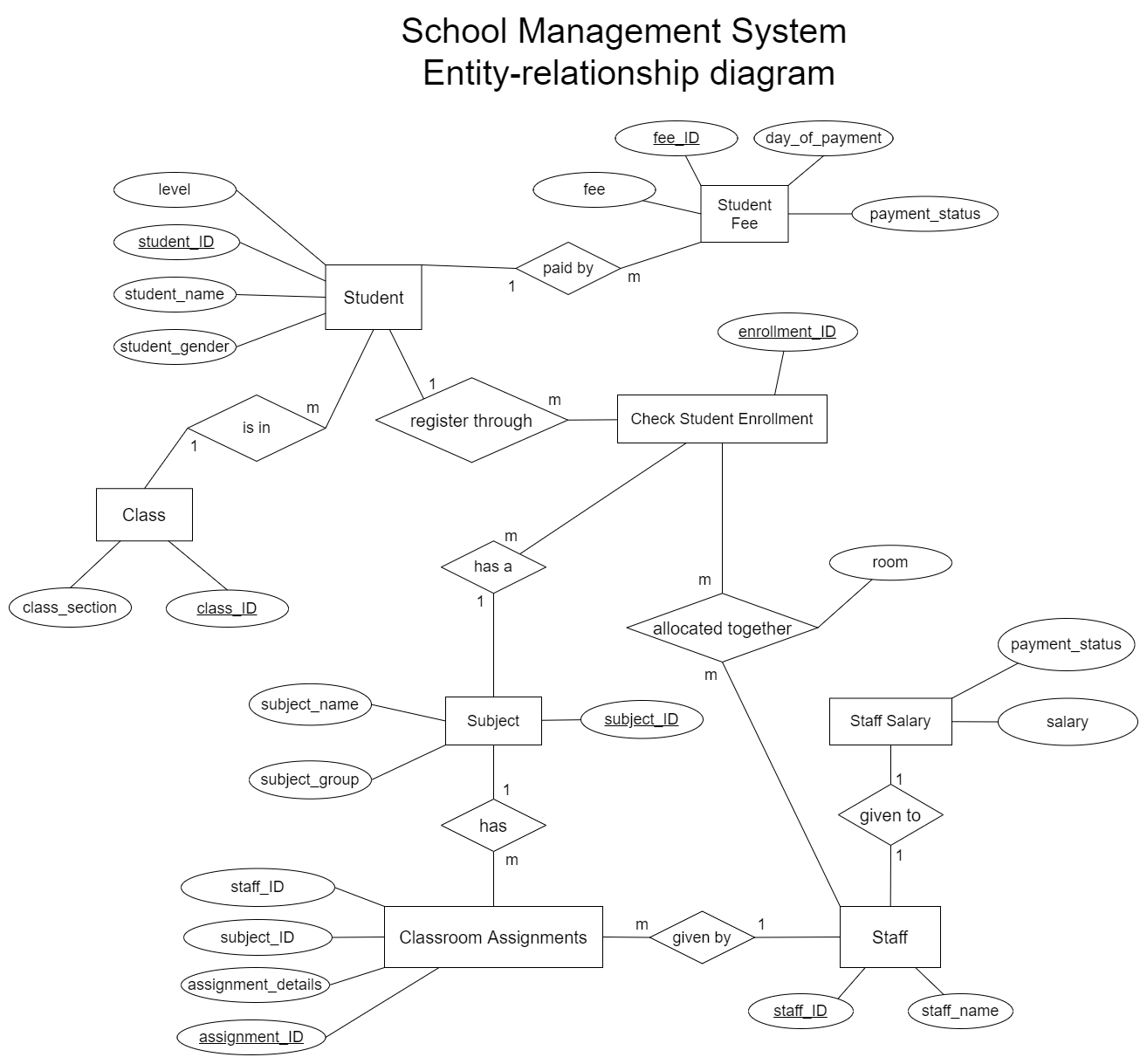
- **Subject** (subject\_ID, subject\_name, subject\_group).

- **Student** **Fee** (fee\_ID, fee, payment\_status, day\_of\_payment).

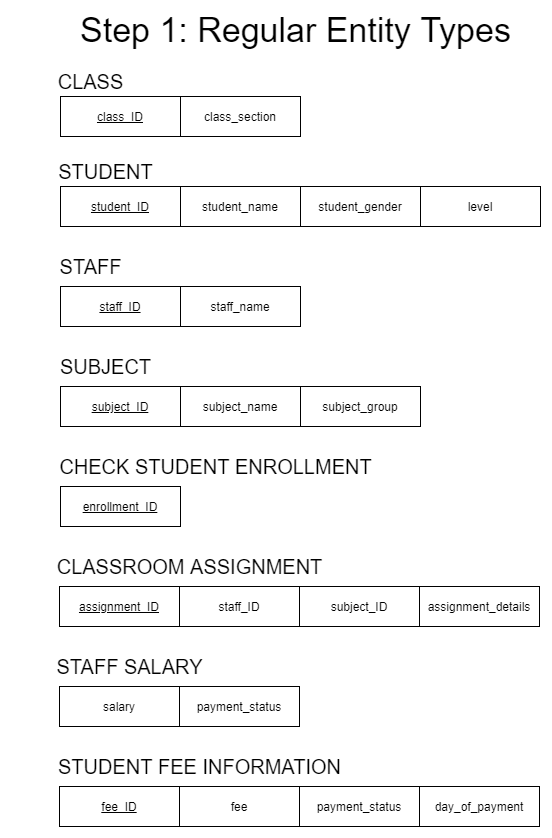
- **Staff** **Salary** (salary, payment\_status).

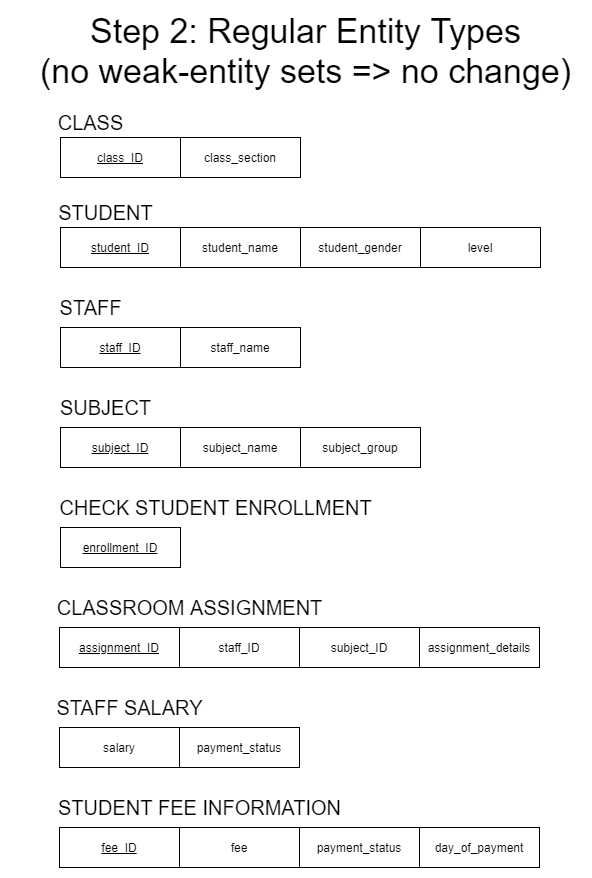
- **Check** **Student** **Enrollment** (enrollment\_ID).

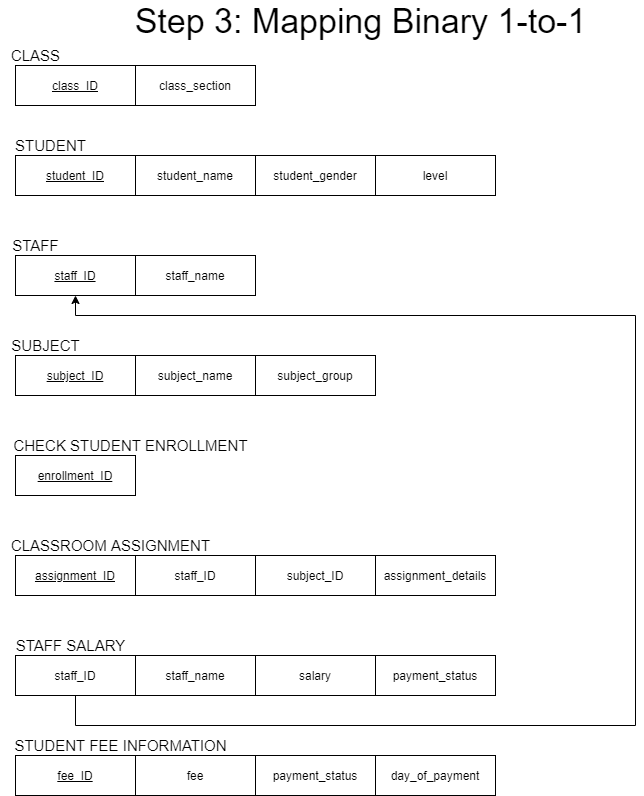
- **Classroom** **Assignment** (assignment\_ID, staff\_ID, subject\_ID, assignment \_details)

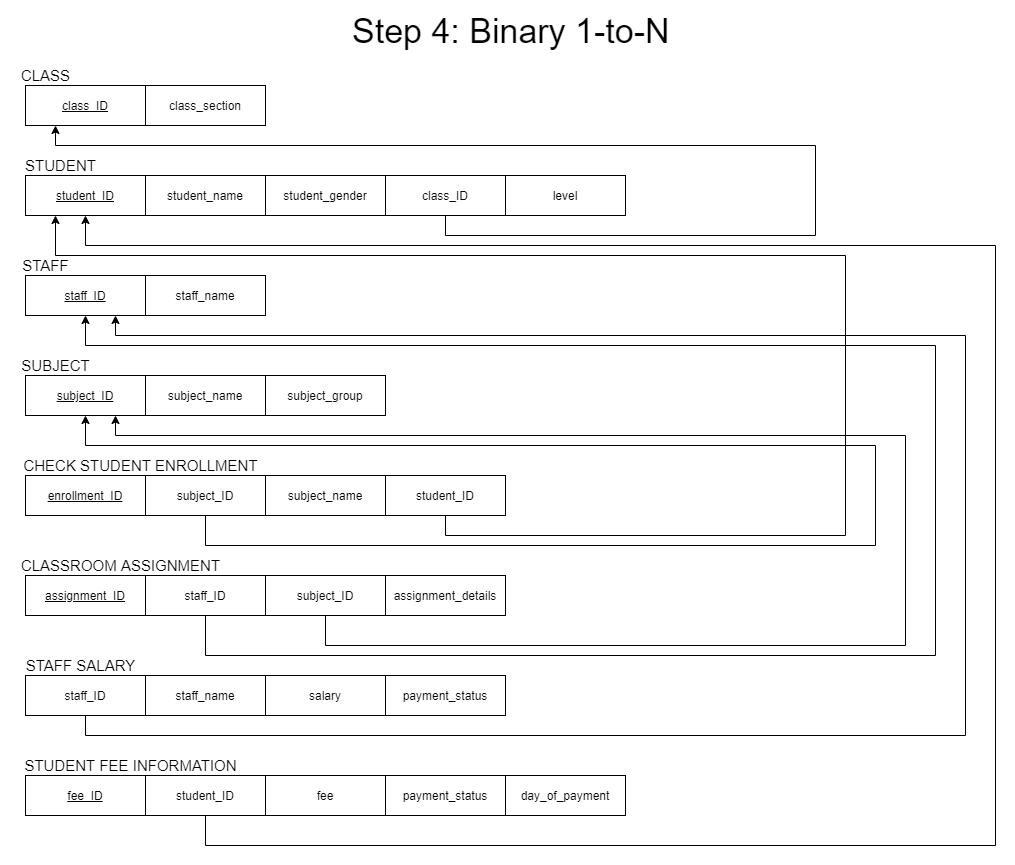


1. Converting from ERD to Relational Model:

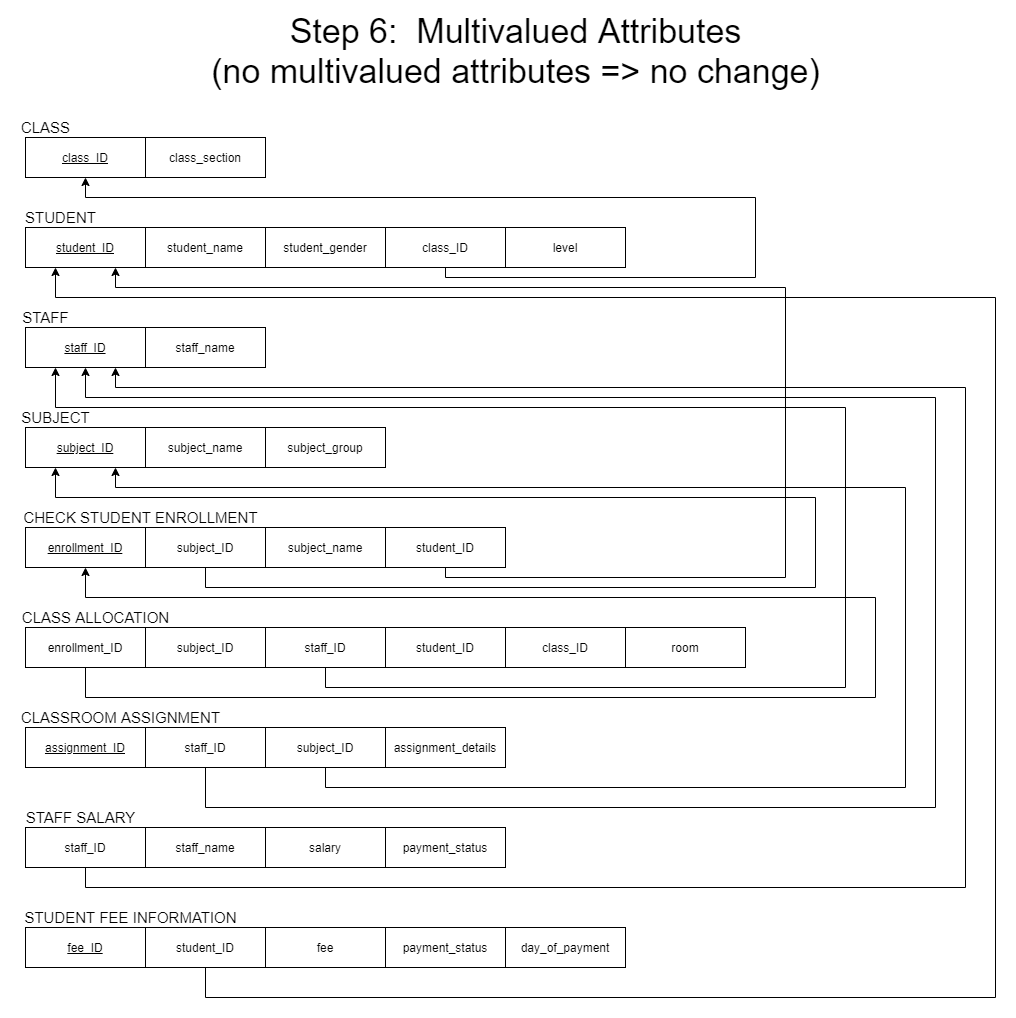












1. Create the database:

USE master

IF EXISTS (SELECT name FROM sys.databases WHERE name = 'SchoolManagementSystem')

DROP DATABASE SchoolManagementSystem;

GO

CREATE DATABASE SchoolManagementSystem;

GO

USE SchoolManagementSystem

GO

DROP TABLE IF EXISTS dbo.AdminCode

CREATE TABLE AdminCode(

access\_code VARCHAR(6) UNIQUE,

CHECK (access\_code LIKE '[0-9]%')

)

GO

DROP TABLE IF EXISTS dbo.Account

CREATE TABLE Account (

name VARCHAR(255),

username VARCHAR(255),

password VARCHAR(255),

role INT,

ID INT,

CHECK (password <> username),

UNIQUE (username, password)

)

GO

DROP TABLE IF EXISTS dbo.Class;

CREATE TABLE Class (

class\_ID VARCHAR(255) PRIMARY KEY,

class\_section VARCHAR(255) NOT NULL,

);

GO

DROP TABLE IF EXISTS dbo.Student;

CREATE TABLE Student (

student\_ID INT PRIMARY KEY,

student\_name VARCHAR(255) NOT NULL,

student\_gender VARCHAR(6) NOT NULL,

class\_ID VARCHAR(255) REFERENCES Class (class\_ID)

ON DELETE CASCADE ON UPDATE SET NULL,

level VARCHAR(255) NOT NULL

);

GO

DROP TABLE IF EXISTS dbo.Staff;

CREATE TABLE Staff (

staff\_ID INT PRIMARY KEY,

staff\_name VARCHAR(255) NOT NULL,

);

GO

DROP TABLE IF EXISTS dbo.Subject;

CREATE TABLE Subject (

subject\_ID VARCHAR(255) PRIMARY KEY,

subject\_name VARCHAR(255) NOT NULL,

subject\_group INT NOT NULL

)

GO

DROP TABLE IF EXISTS dbo.Check\_student\_enrollment;

CREATE TABLE Check\_student\_enrollment(

enrollment\_ID INT PRIMARY KEY ,

student\_ID INT REFERENCES Student (student\_ID)

ON DELETE CASCADE ON UPDATE NO ACTION,

subject\_name VARCHAR(255),

subject\_ID VARCHAR(255) REFERENCES Subject (subject\_ID)

ON DELETE CASCADE ON UPDATE SET NULL,

UNIQUE (student\_ID, subject\_name)

)

GO

DROP TABLE IF EXISTS dbo.Class\_allocation;

CREATE TABLE Class\_allocation (

enrollment\_ID INT REFERENCES Check\_student\_enrollment(enrollment\_ID)

ON DELETE CASCADE ON UPDATE CASCADE,

subject\_ID VARCHAR(255) ,

staff\_ID INT REFERENCES Staff (staff\_ID)

ON DELETE SET NULL ON UPDATE SET NULL,

student\_ID INT,

class\_ID VARCHAR(255),

room VARCHAR(255) NOT NULL,

);

GO

DROP TABLE IF EXISTS dbo.Classroom\_assignments;

CREATE TABLE Classroom\_assignments (

assignment\_ID INT PRIMARY KEY,

staff\_ID INT REFERENCES Staff (staff\_ID)

ON DELETE CASCADE ON UPDATE NO ACTION,

subject\_ID VARCHAR(255) REFERENCES Subject (subject\_ID)

ON DELETE CASCADE ON UPDATE NO ACTION,

assignment\_details VARCHAR(255) NOT NULL,

);

GO

DROP TABLE IF EXISTS dbo.Staff\_salary

CREATE TABLE Staff\_salary (

staff\_ID INT REFERENCES Staff (staff\_ID)

ON DELETE CASCADE ON UPDATE NO ACTION,

staff\_name VARCHAR(255) NOT NULL,

salary MONEY NOT NULL,

payment\_status VARCHAR(3) DEFAULT 'NO'

);

GO

DROP TABLE IF EXISTS dbo.Student\_fee\_information

CREATE TABLE Student\_fee\_information (

fee\_ID INT PRIMARY KEY,

student\_ID INT REFERENCES Student (student\_ID)

ON DELETE CASCADE ON UPDATE NO ACTION,

fee MONEY NOT NULL,

payment\_status VARCHAR(3) DEFAULT 'NO',

day\_of\_payment DATE NOT NULL,

);

GO

--Here below are queries to import data from .xls (excel) files into the tables:

/\* Before you can run a distributed query,

you have to enable the ad hoc distributed queries server configuration option, as shown in the following example.

sp\_configure 'show advanced options', 1;

RECONFIGURE;

GO

sp\_configure 'Ad Hoc Distributed Queries', 1;

RECONFIGURE;

GO

\*\*COMMON ERRORS\*\*

1) Microsoft.ACE.OLEDB.12.0" has not been registered

This error occurs because the OLEDB provider is not installed.

Install it from Microsoft Access Database Engine 2010 Redistributable.

Be sure to install the 64-bit version if Windows and SQL Server are both 64-bit.

The full error is:

[...]

The OLE DB provider "Microsoft.ACE.OLEDB.12.0" has not been registered.

Please download from below link and set up it.

https://www.microsoft.com/en-us/download/details.aspx?id=13255

2) Cannot create an instance of OLE DB provider "Microsoft.ACE.OLEDB.12.0" for linked server "(null)"

This indicates that the Microsoft OLEDB has not been configured properly.

The full error is:

[...]

Cannot create an instance of OLE DB provider "Microsoft.ACE.OLEDB.12.0" for linked server "(null)".

Run the following Transact-SQL code to resolve this:

EXEC sp\_MSset\_oledb\_prop N'Microsoft.ACE.OLEDB.12.0', N'AllowInProcess', 1

EXEC sp\_MSset\_oledb\_prop N'Microsoft.ACE.OLEDB.12.0', N'DynamicParameters', 1

3) The 32-bit OLE DB provider "Microsoft.ACE.OLEDB.12.0" cannot be loaded in-process on a 64-bit SQL Server

This occurs when a 32-bit version of the OLD DB provider is installed with a 64-bit SQL Server.

To resolve this issue, uninstall the 32-bit version and install the 64-bit version of the OLE DB provider instead.

The full error is:

[...]

The 32-bit OLE DB provider "Microsoft.ACE.OLEDB.12.0" cannot be loaded in-process on a 64-bit SQL Server.

4) For further error, please visit the link below:

https://docs.microsoft.com/en-us/sql/relational-databases/import-export/import-data-from-excel-to-sql?view=sql-server-ver15#common-errors

\*/

-- Note: Please CHANGE the path name of address of all the import queries below:

DROP TABLE IF EXISTS dbo.ImportIntoTable

SELECT \* INTO ImportIntoTable

FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0',

'Excel 12.0; Database=C:\SQL Lab\ASSIGNMENT\Import Data - Excel Files\Admin\_code.xls; HDR=YES; IMEX=1',

'SELECT \* FROM [Sheet1$]');

GO

INSERT INTO AdminCode

SELECT \* FROM ImportIntoTable

GO

DROP TABLE IF EXISTS dbo.ImportIntoTable

SELECT \* INTO ImportIntoTable

FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0',

'Excel 12.0; Database=C:\SQL Lab\ASSIGNMENT\Import Data - Excel Files\Account.xls; HDR=YES; IMEX=1',

'SELECT \* FROM [Sheet1$]');

GO

INSERT INTO Account

SELECT \* FROM ImportIntoTable

GO

DROP TABLE IF EXISTS dbo.ImportIntoTable

SELECT \* INTO ImportIntoTable

FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0',

'Excel 12.0; Database=C:\SQL Lab\ASSIGNMENT\Import Data - Excel Files\Class.xls; HDR=YES; IMEX=1',

'SELECT \* FROM [Sheet1$]');

GO

INSERT INTO Class

SELECT \* FROM ImportIntoTable

GO

DROP TABLE IF EXISTS dbo.ImportIntoTable

SELECT \* INTO ImportIntoTable

FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0',

'Excel 12.0; Database=C:\SQL Lab\ASSIGNMENT\Import Data - Excel Files\Student.xls; HDR=YES; IMEX=1',

'SELECT \* FROM [Sheet1$]');

GO

INSERT INTO Student

SELECT \* FROM ImportIntoTable

GO

DROP TABLE IF EXISTS dbo.ImportIntoTable

SELECT \* INTO ImportIntoTable

FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0',

'Excel 12.0; Database=C:\SQL Lab\ASSIGNMENT\Import Data - Excel Files\Staff.xls; HDR=YES; IMEX=1',

'SELECT \* FROM [Sheet1$]');

GO

INSERT INTO Staff

SELECT \* FROM ImportIntoTable

GO

DROP TABLE IF EXISTS dbo.ImportIntoTable

SELECT \* INTO ImportIntoTable

FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0',

'Excel 12.0; Database=C:\SQL Lab\ASSIGNMENT\Import Data - Excel Files\Subject.xls; HDR=YES; IMEX=1',

'SELECT \* FROM [Sheet1$]');

GO

INSERT INTO Subject

SELECT \* FROM ImportIntoTable

GO

DROP TABLE IF EXISTS dbo.ImportIntoTable

SELECT \* INTO ImportIntoTable

FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0',

'Excel 12.0; Database=C:\SQL Lab\ASSIGNMENT\Import Data - Excel Files\Check\_student\_enrollment.xls; HDR=YES; IMEX=1',

'SELECT \* FROM [Sheet1$]');

GO

INSERT INTO Check\_student\_enrollment

SELECT \* FROM ImportIntoTable

GO

DROP TABLE IF EXISTS dbo.ImportIntoTable

SELECT \* INTO ImportIntoTable

FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0',

'Excel 12.0; Database=C:\SQL Lab\ASSIGNMENT\Import Data - Excel Files\Class\_allocation.xls; HDR=YES; IMEX=1',

'SELECT \* FROM [Sheet1$]');

GO

INSERT INTO Class\_allocation

SELECT \* FROM ImportIntoTable

GO

DROP TABLE IF EXISTS dbo.ImportIntoTable

SELECT \* INTO ImportIntoTable

FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0',

'Excel 12.0; Database=C:\SQL Lab\ASSIGNMENT\Import Data - Excel Files\Classroom\_assignments.xls; HDR=YES; IMEX=1',

'SELECT \* FROM [Sheet1$]');

GO

INSERT INTO Classroom\_assignments

SELECT \* FROM ImportIntoTable

GO

DROP TABLE IF EXISTS dbo.ImportIntoTable

SELECT \* INTO ImportIntoTable

FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0',

'Excel 12.0; Database=C:\SQL Lab\ASSIGNMENT\Import Data - Excel Files\Staff\_Salary.xls; HDR=YES; IMEX=1',

'SELECT \* FROM [Sheet1$]');

GO

INSERT INTO Staff\_Salary

SELECT \* FROM ImportIntoTable

GO

DROP TABLE IF EXISTS dbo.ImportIntoTable

SELECT \* INTO ImportIntoTable

FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0',

'Excel 12.0; Database=C:\SQL Lab\ASSIGNMENT\Import Data - Excel Files\Student\_fee\_information.xls; HDR=YES; IMEX=1',

'SELECT \* FROM [Sheet1$]');

GO

INSERT INTO Student\_fee\_information

SELECT \* FROM ImportIntoTable

GO

DROP TABLE IF EXISTS dbo.ImportIntoTable

GO

--Some queries that to retrieve all the tables data

SELECT \* FROM Class;

GO

SELECT \* FROM Student;

GO

SELECT \* FROM Staff;

GO

SELECT \* FROM Subject;

GO

SELECT \* FROM Check\_student\_enrollment;

GO

SELECT \* FROM Class\_allocation;

GO

SELECT \* FROM Classroom\_assignments;

GO

SELECT \* FROM Staff\_salary;

GO

SELECT \* FROM Student\_fee\_information;

GO

SELECT \* FROM Account;

GO

SELECT \* FROM AdminCode;

GO

--Select all student name and their ID who enrolled all Subject

SELECT DISTINCT S.student\_name, S.class\_ID

FROM Student S, Class\_allocation C

WHERE S.student\_ID = C.student\_ID

AND C.student\_ID IN

(

SELECT DISTINCT C.student\_ID

FROM Class\_allocation C

GROUP BY C.student\_ID

HAVING COUNT(\*) =

(

SELECT COUNT(Result.sub\_name)

FROM

(

SELECT S.subject\_name AS sub\_name

FROM Subject S

GROUP BY S.subject\_name

) AS Result

)

)

GO

--Find all student name and their ID who is Junior student and paid for school fee before 10/04/2020

SELECT S.student\_name, S.student\_ID, S.level, Sf.day\_of\_payment

FROM Student S, Student\_fee\_information Sf

WHERE S.student\_ID = Sf.student\_ID

AND S.level = 'Junior'

AND Sf.payment\_status = 'YES' and Sf.day\_of\_payment <= '2020-04-10'

GO

--Find all the staff and their class who is teaching 4 classes

SELECT S.staff\_ID, S.staff\_name, Result1.number\_of\_teaching\_classes

FROM Staff S,

(

SELECT Result.staff\_ID, COUNT(Result.subject\_ID) AS number\_of\_teaching\_classes

FROM

(

SELECT DISTINCT C.subject\_ID, C.staff\_ID

FROM Class\_allocation C

) AS Result

GROUP BY Result.staff\_ID

) AS Result1

WHERE S.staff\_ID = Result1.staff\_ID

AND Result1.number\_of\_teaching\_classes = 4

GO

--Find all student who is female:

SELECT \*

FROM Student WHERE student\_gender = 'Female';

GO

--Find staff who teaches Marxism:

SELECT DISTINCT S.staff\_name

FROM Staff S, Class\_allocation C, Subject Su

WHERE S.staff\_Id = C.staff\_ID

AND C.subject\_ID = Su.subject\_ID

AND Su.subject\_name = 'Marxism';

GO

--Find all the subject name and subject group of staff name “Vo Thi Luu Phuong”:

SELECT DISTINCT S.subject\_name, S.subject\_group

FROM Subject S, Class\_allocation C , Staff Sf

WHERE S.subject\_ID = C.subject\_ID

AND Sf.staff\_name = 'Vo Thi Luu Phuong'

AND C.staff\_ID = Sf.staff\_ID

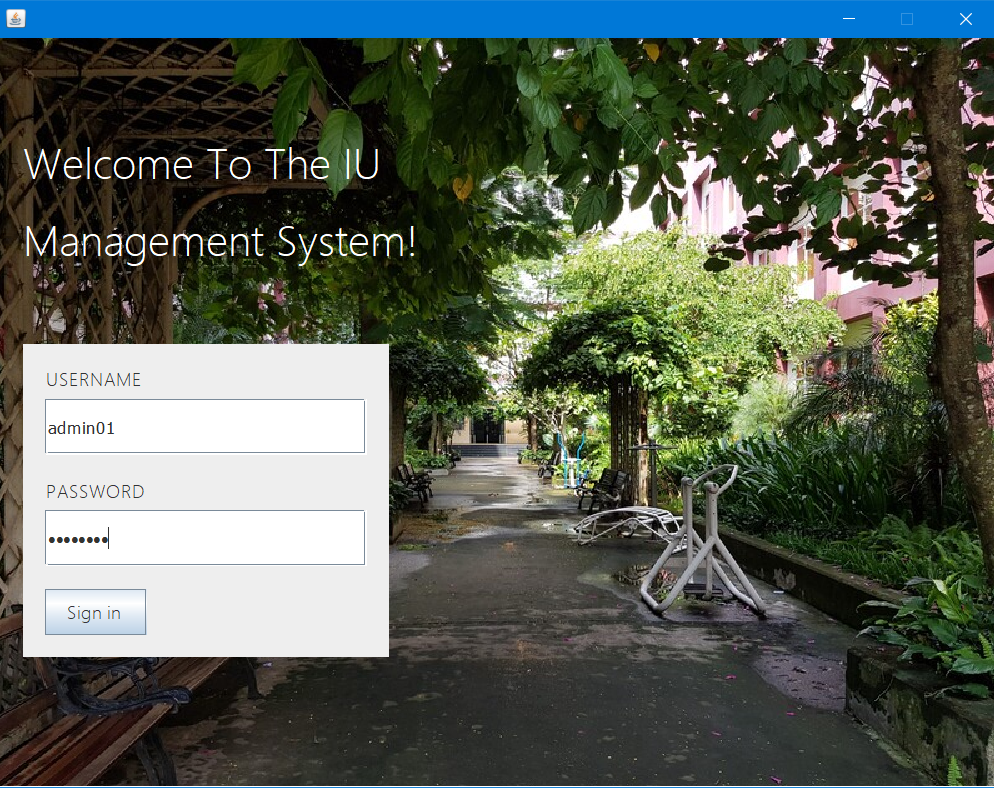
GO

1. Querying the database using JDBC:

By using the attached java GUI file, we introduce the user through the user interface, allowing them to simplify the operation with the database.

Guideline:

* First, for each new student entering the school, they will be given an account to view their information and register for courses, each teacher will also be given an account to view their information, his courses that assigned by the manager and also create an assignment for the students. Admins also have accounts but with more special functions.
* All kinds of users will log in to the system via their account.
* They are only allowed to enter the wrong account or password 5 times only.



+ If they are an admin, here is the menu:



+ If they are a student, here is the menu:



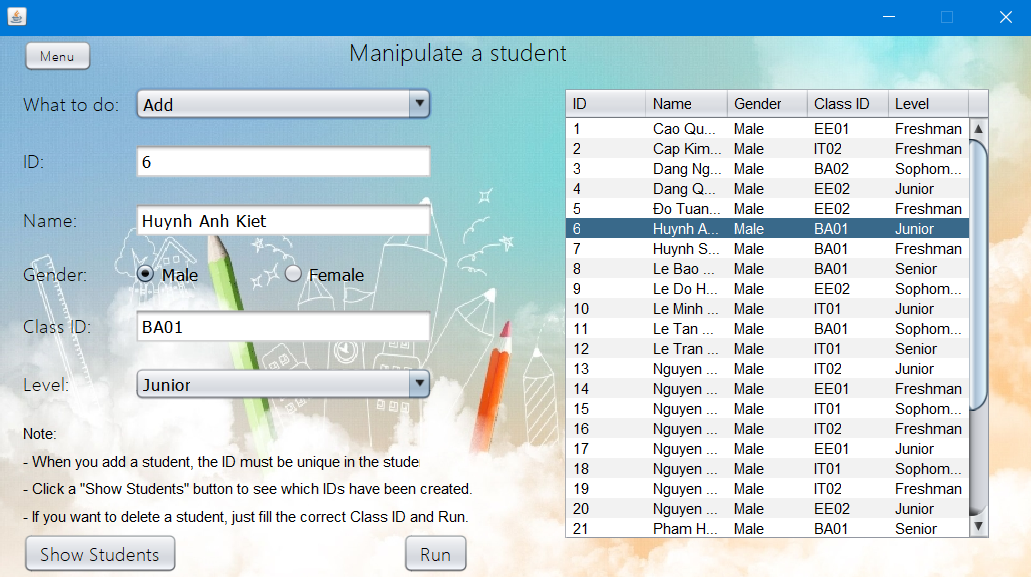
+ If they are a staff, here is the menu:

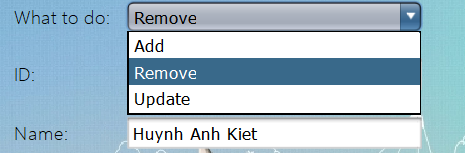


Access rights:

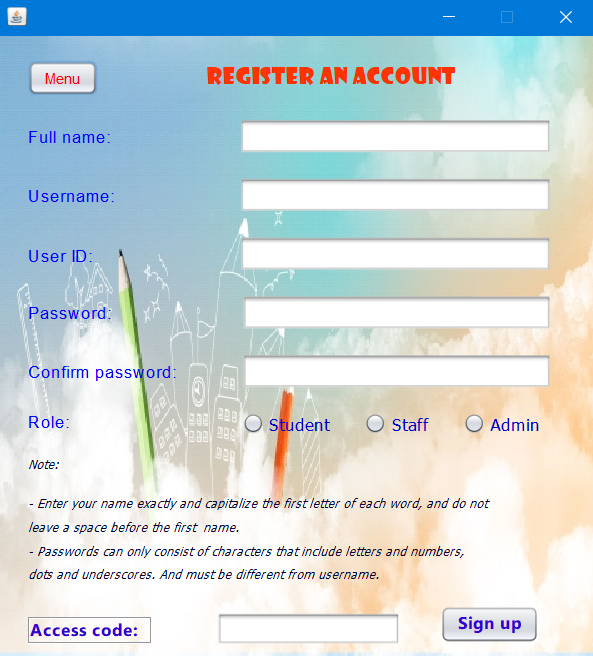
* An admin has full rights to edit/manipulate the database, including special rights like creating an account (It should be noted that this is a school management database, so, for a normal person, it is impossible to arbitrarily create an account to gain access to the database).

\*For example, an admin can do such these things: (add/delete/update)



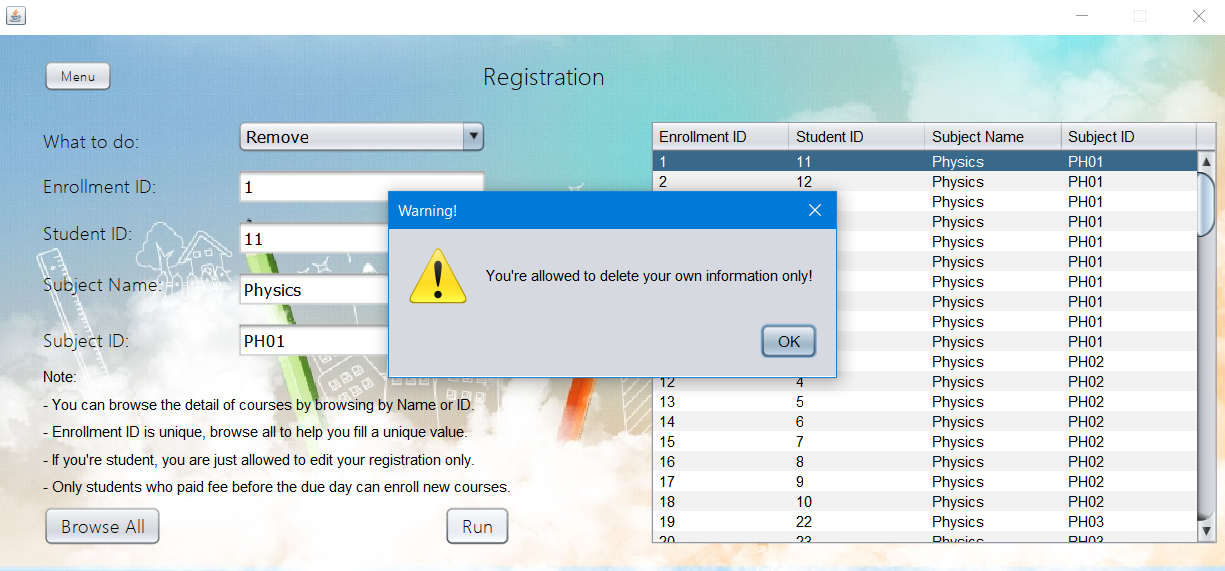


\*A super admin right is to create an account, they can create a student account, staff, and even an admin account, however, the people who have been granted this admin account can only do this similar works if they have access code:

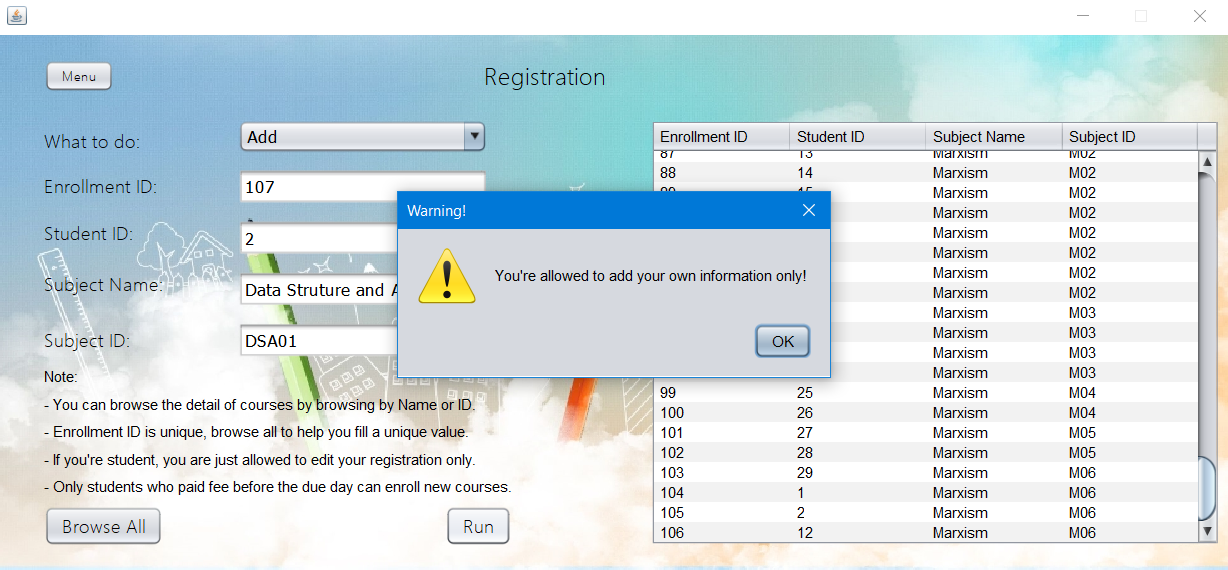


* A student can register for a course, as long as he pays the course fee by a certain deadline and makes sure he/she does not register two courses of the same name (Note: unlike admin and staff, he can only edit his own information only). In addition, he/she can view his/her subject organization, assignments and tuition status.

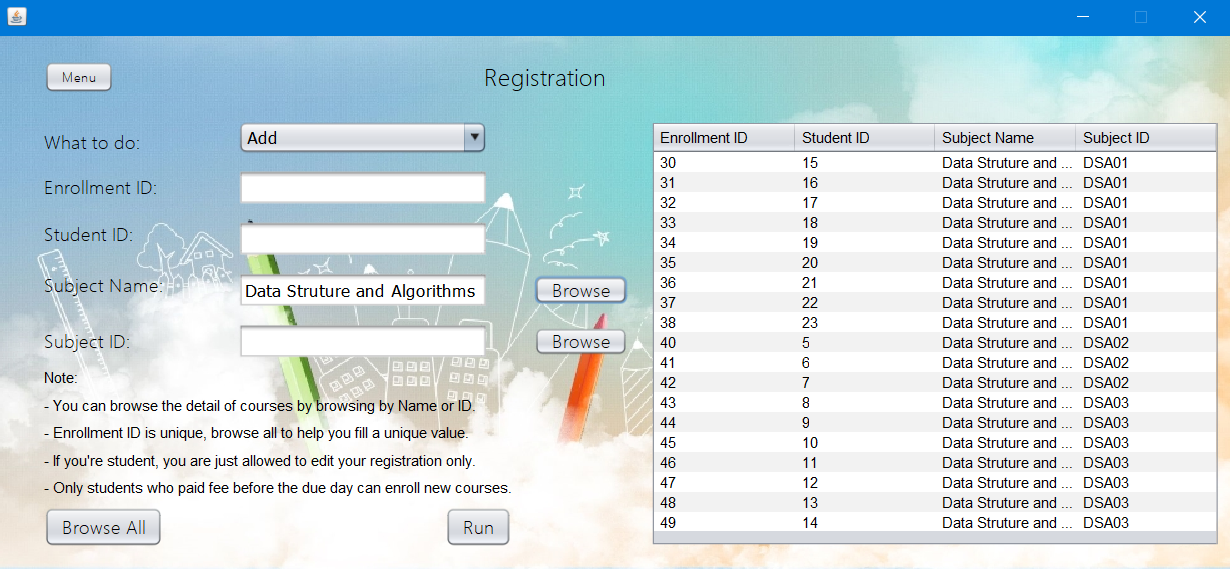
\*For example, this student tries to delete someone’s registration, which is not allowed to:

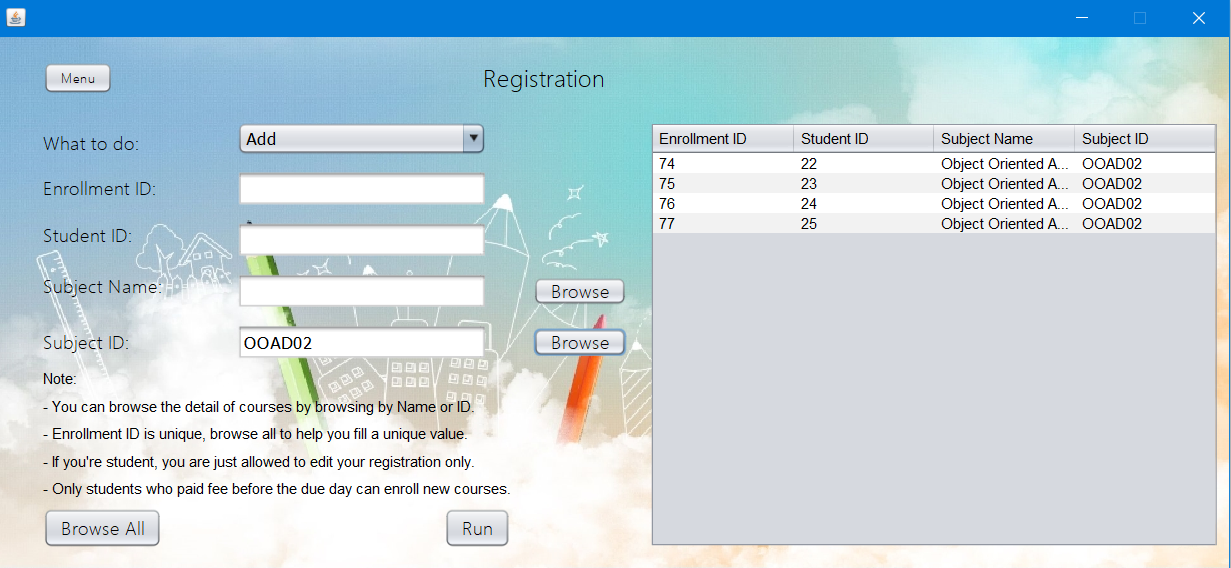


\*The same for adding a registration:

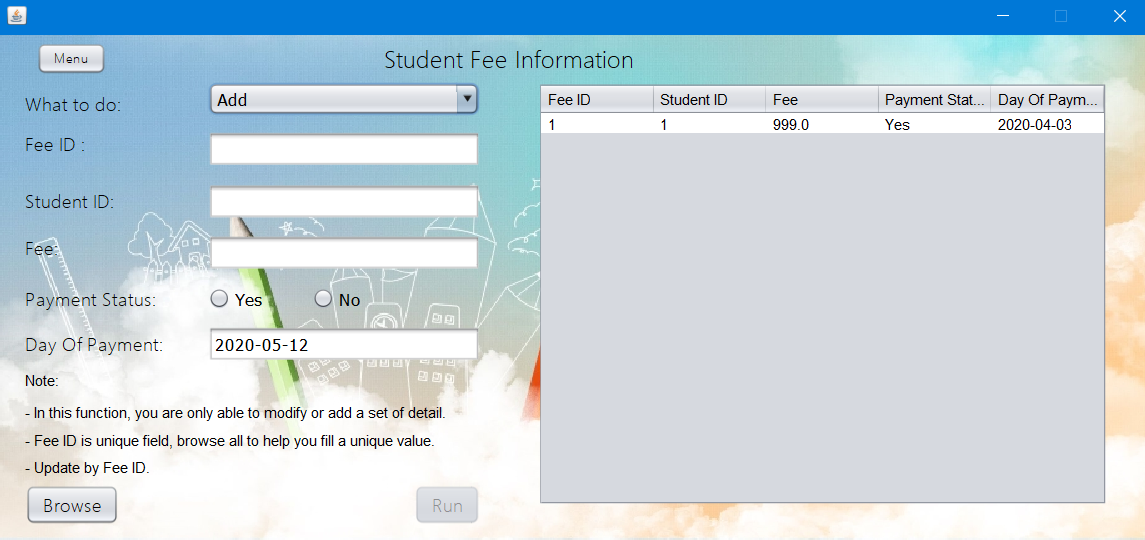


\*In some functions, we have added browse buttons (all, by name, by ID) to help users to be able to operate correctly with the system, to avoid having to retype. Here is an example of “Browse” with “by subject name” or “by subject ID”.



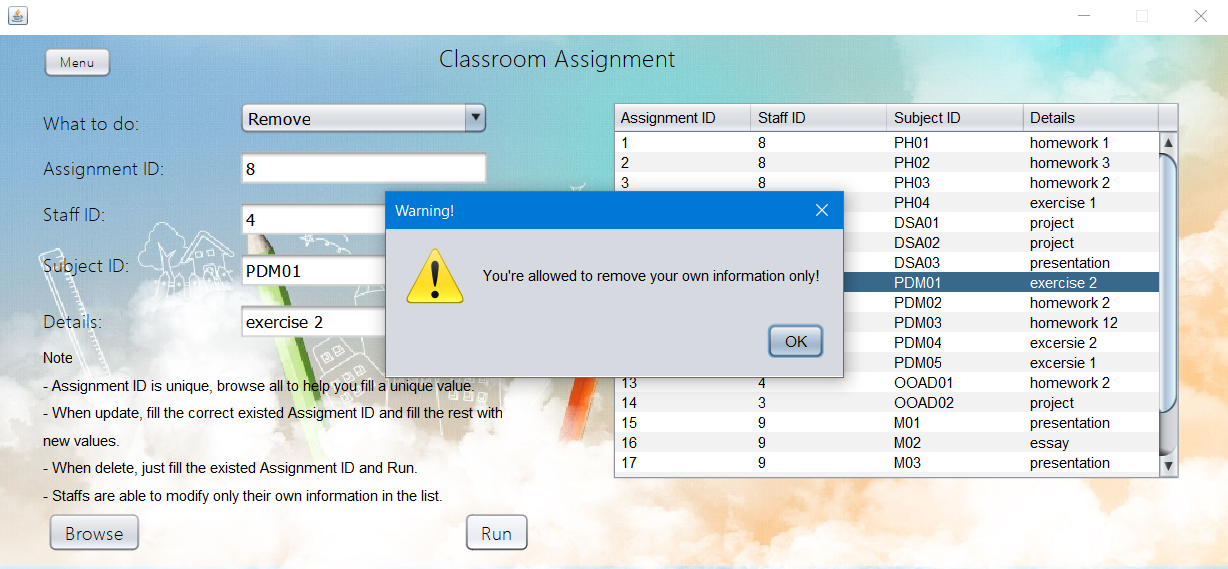


\*Some functions may be restricted to students. For example, students can only view their own tuition, and cannot edit anything that not in their scope: (Cao Quoc Vi has logged in)

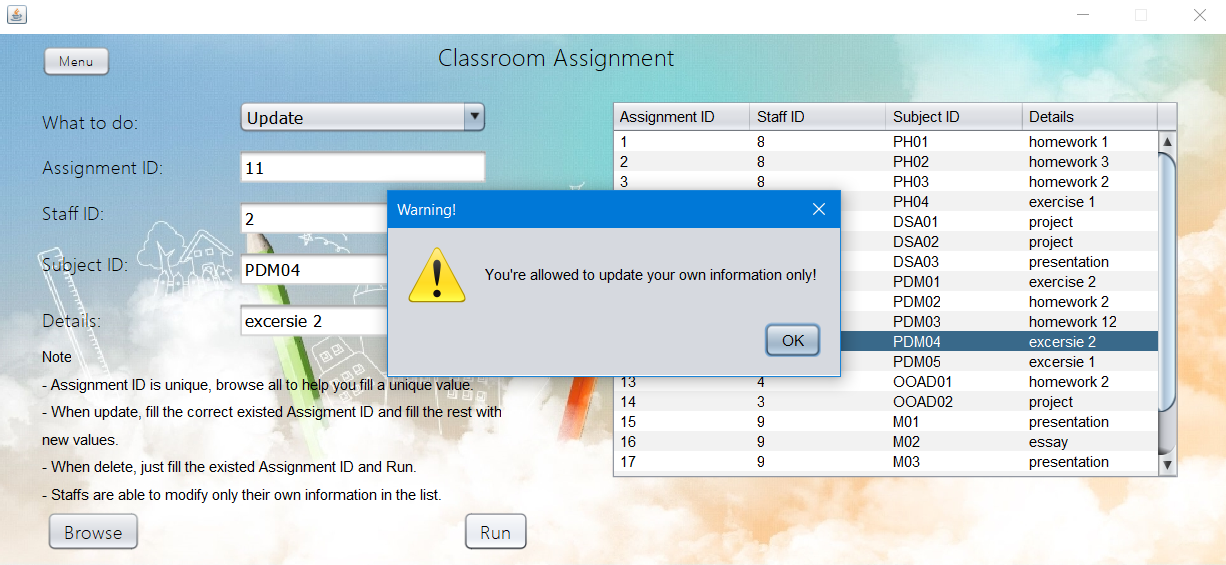


* An employee can register a course for students who need help, view their class information, create and edit their own assignments and of course, they can also see salary information.

\*For example, this staff tries to delete another staff’s assignment, which is not allowed to:



\*The same for adding / modifying an assignment:



\*Some functions may be restricted to staffs. For example, staffs can only view their own salary, and cannot edit anything that not in their scope: (Vo Thi Luu Phuong has logged in):

