MACAU UNIVERSITY OF SCIENCE AND TECHNOLOGY

Faculty of Information and Technology

Project Report

CS108 Advanced Database Systems, Semester 2009

Instructor: Siolong Lo

Student Name: Yuyang Wang

1809853Z-I011-0045

January 1, 2021

Contents

Abstract

CS108 is a practical course which provides me with a general concept about database system, also teaches me varieties of useful SQL methods in data operations including insertion, deletion, updating and query.

In this project, I design a **Online Student Grading System (OSGS)** with **ASP.NET 4.7.2**, **Microsoft Visual Studio Community 2019 16.7.5** for front-end and **C sharp** for back-end, which is based on **SQL Server Management Studio 15.0.18358.0 (SSMS)**. In this online system, teachers can register course or student, as well create assignments for students who take the course. And students can also submit their answers online and the teacher can grade and give them some comments simultaneously.

The whole project is designed by Kennard Wang (Yuyang Wang), totally costs 4 days. The source code will be uploaded to GitHub later, welcome to visit my GitHub page via the following link: `https://github.com/KennardWang?tab=repositories`

1 Project Demo

1.1 Login Page (Login.aspx)

In Figure 1 and 2, the login page must be the initial page which has two different login types, which will handle the login event of both teachers and students. Users can login with correct **ID** and **Password**. The teachers' information is assigned by **SSMS** and students' information is assigned by a teacher.

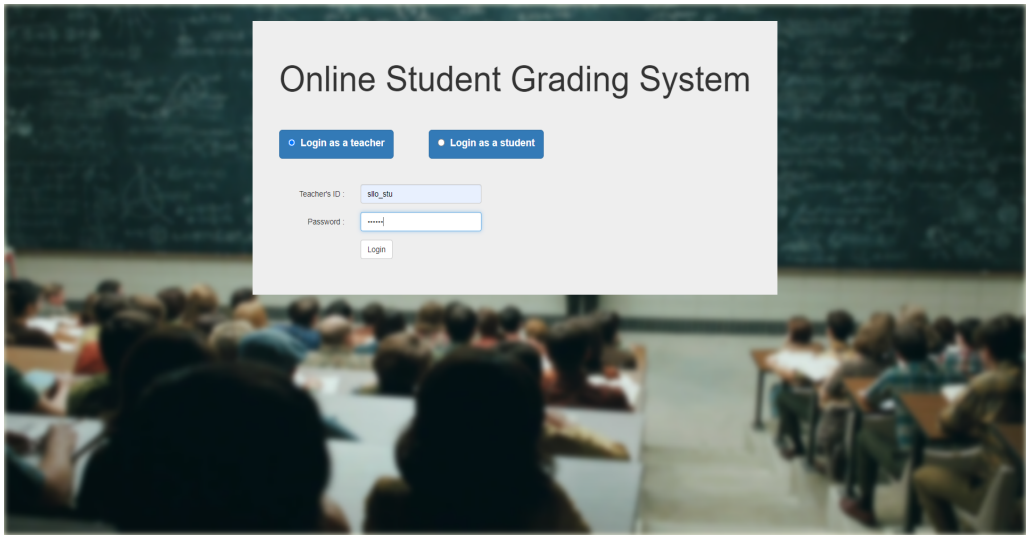

Figure 1: Teacher Login Page

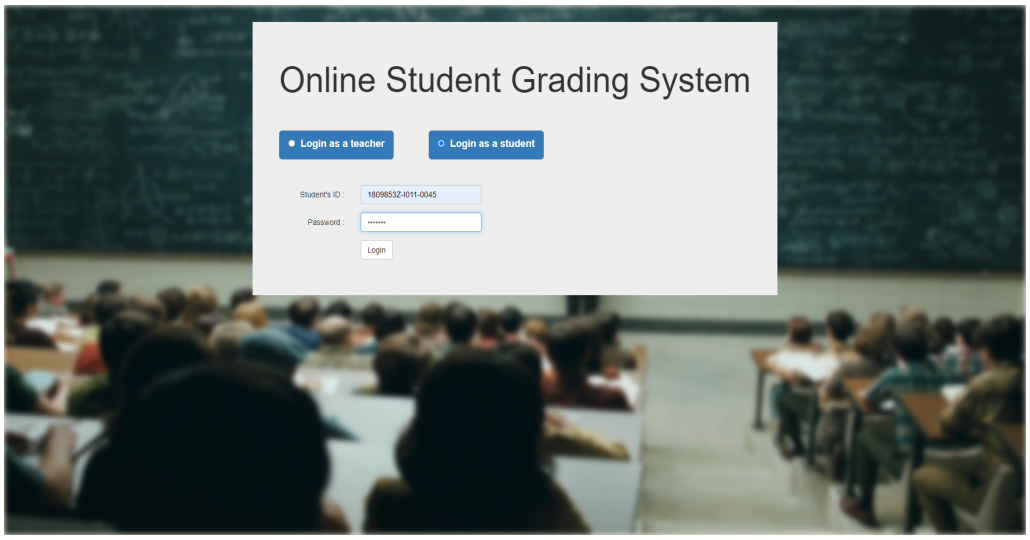

Figure 2: Student Login Page

1.2 Student Account Management Page (StuOverview.aspx, for teacher)

As we all known, a teacher probably have more than one courses. For example, in Figure 3, Professor Lo has two courses CS101 and CS108. In this page, he can see all the students who take CS108 by choosing the **DropDownList** whose value is 'CS108'. By the way, teacher can also **Edit** the information of students (including Name, Password, Email Address) or **Delete** it if the student cancel the course.

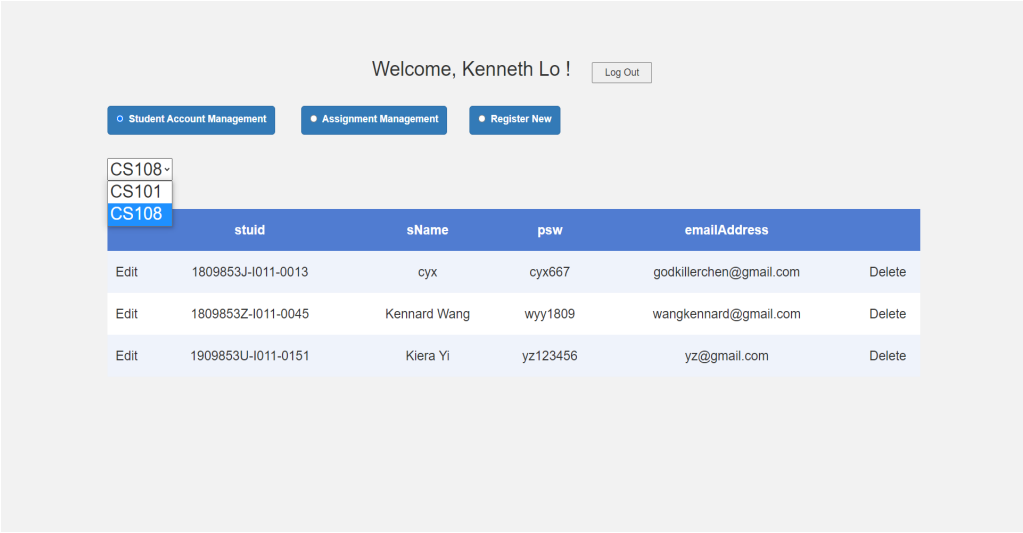

Figure 3: Student Account Management Page

1.3 Assignment Management Page (AssignmentManagement.aspx, for teacher)

Switching the **RadioButtonList** can enter into this page (Figure 4). We can see all the assignments that the teacher gives to each course, and also about assignment info (name, weight, total question number, deadline, submit state), student info, grade info (grade and comment). Clicking the **Select** button to get more information.

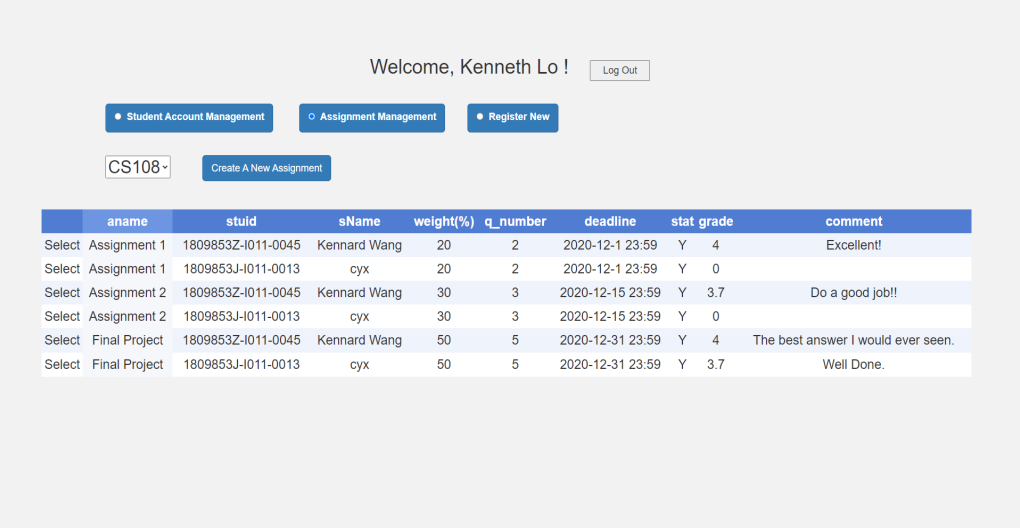

Figure 4: Assignment Management Page

1.4 Assignment Marking Page (AssignmentMarking.aspx, for teacher)

Figure 5 is about assignment and student information, all the questions and student answers (if have) will be displayed. The teacher can give comment and a grade, the data will be updated in the previous page.

1.5 Assignment Creating Page (AddNewAssignment.aspx, for teacher)

To create a new assignment, the teacher can enter information about this assignment, also choose to add or delete a question, which implements the variability of the question number. You can see the demo in Figure 6.

1.6 Register Page (RegisterNew.aspx, for teacher)

In Figure 7 and 8, the page is used for course register or a new student register, the teacher can select the register type (Student Register for a new student, Course Register for a student who have not taken the course) and add information into the database.

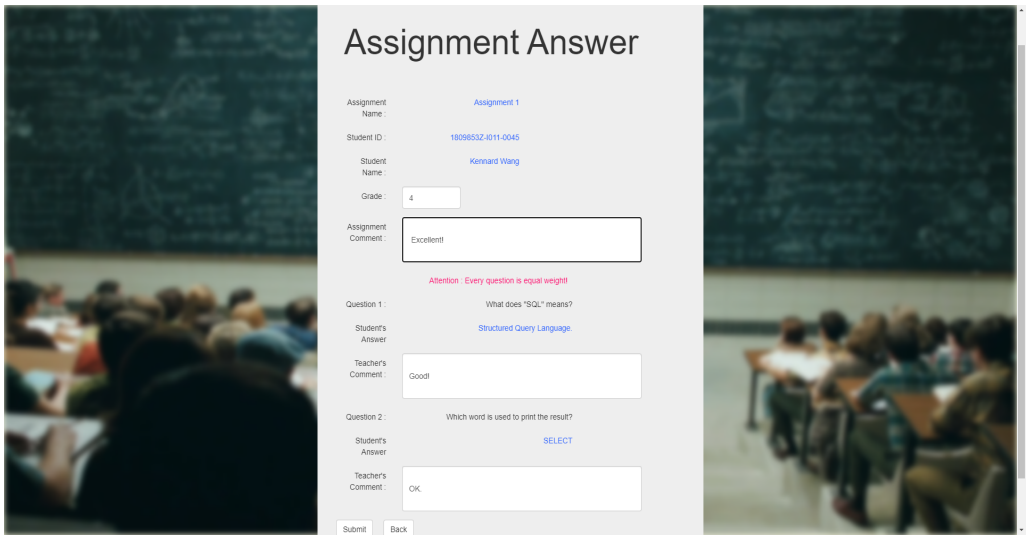

Figure 5: Assignment Marking Page

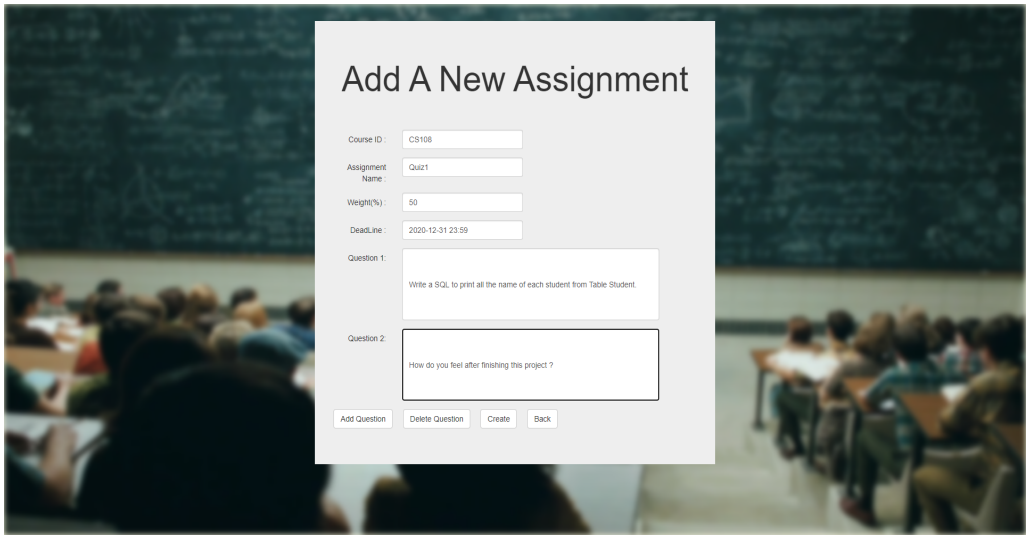

Figure 6: Assignment Creating Page

1.7 Student Assignment Page (AssignmentOverview.aspx, for student)

After logging in as a student, you will enter into this page (Figure 9). For each course you have taken, you can see each of the assignment info (weight, total question number, deadline, grade, submission state [Y for yes/N for no], teacher's comment) and the **GPA** will be computed and displayed automatically.

1.8 Student Answer Page (StudentAnswer.aspx, for student)

In this page (Figure 10), students can see the details about their assignment answers or submit their answers. But if the student has submitted the answer before, he or

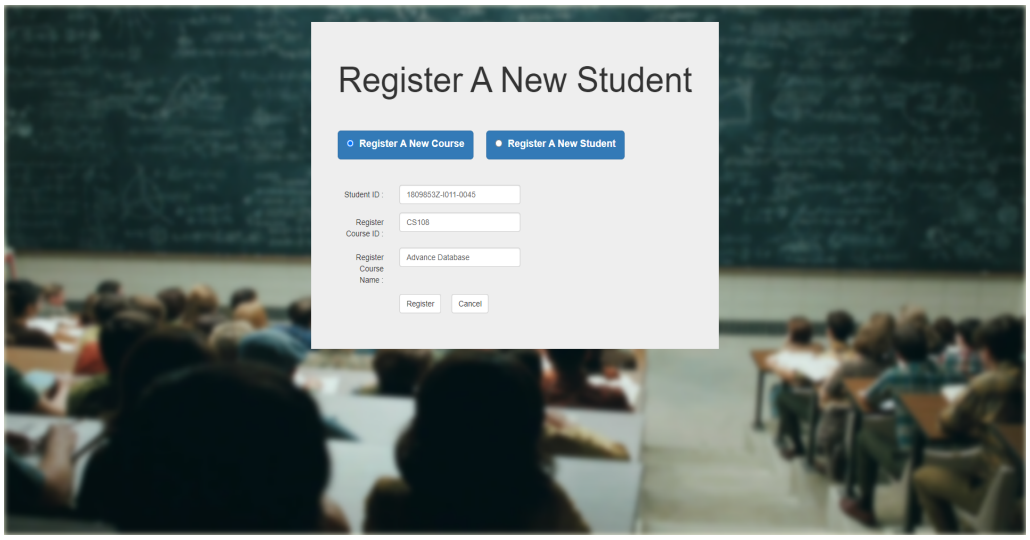

Figure 7: Register Course Page

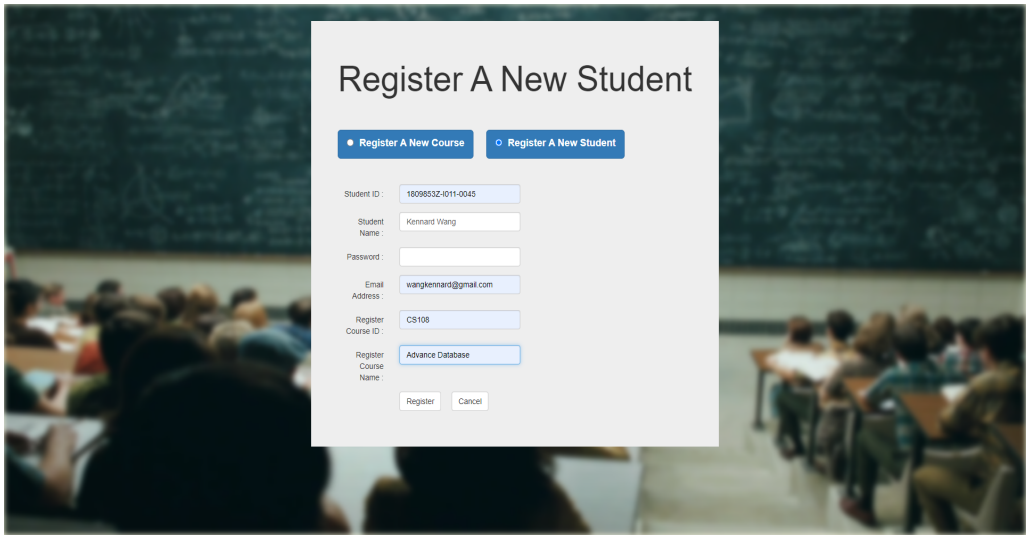

Figure 8: Register Student Page

she will not be allowed to update their answers again. The teacher's comment can also be seen in this page (if has).

2 Physical Design

2.1 Class Diagram

Figure 11 is drawn by **NClass 2.04**, we have 8 **C sharp** classes, corresponding to each front page shown before. For each class, I implement some event functions which can handle all the events (such as **Clicking, DropdownList Changed, Text Changed, RadioButton Changed etc.**) taking place in the current page.

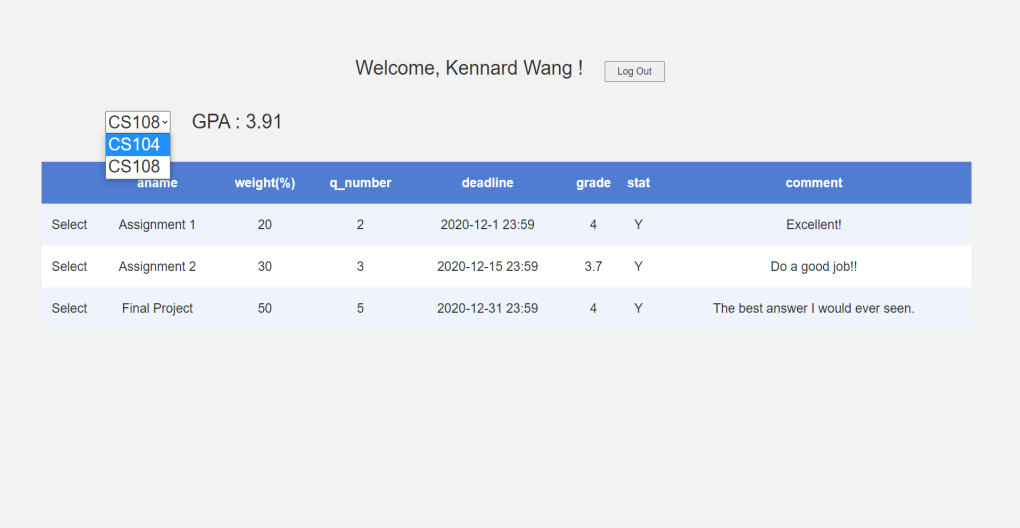

Figure 9: Student Assignment Page

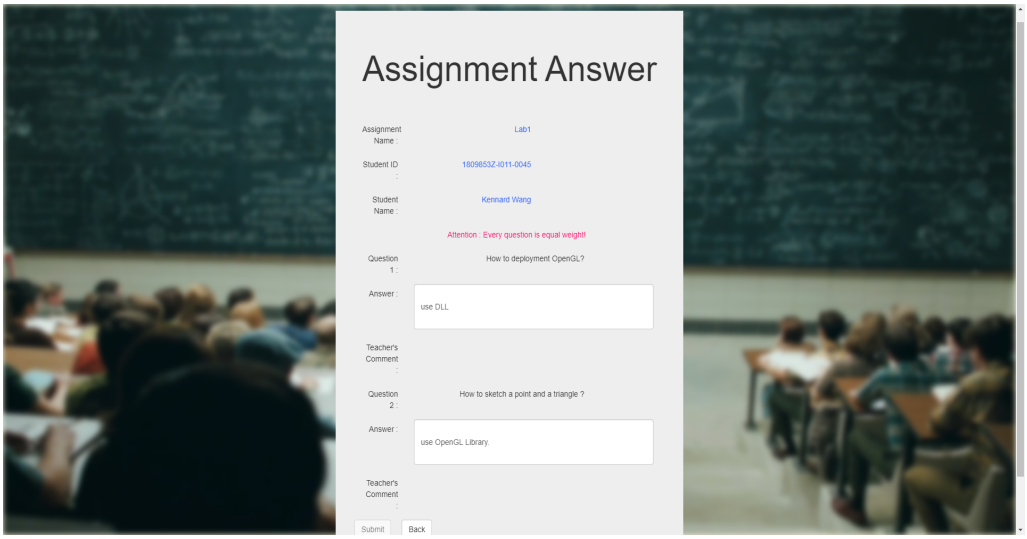

Figure 10: Student Answer Page

2.2 ER Diagram

The ER-diagram of database is shown as Figure 12.

In design phase, I firstly find the **Student Entity** and the **Teacher Entity** has the same attributes like name, password and email address, which reminds me to use **ISA** structure to represent.

And then, **Course Entity** records the course ID and the course name. It has both relationship with **Student** and **Teacher**, that's why **stuid and tid** are foreign keys of **Course**.

For **Assignment Entity**, we set **aid** as the primary key. This entity has lots of attributes but it still has a constraint with **cid** of **Course**. By the way, **Assignment**

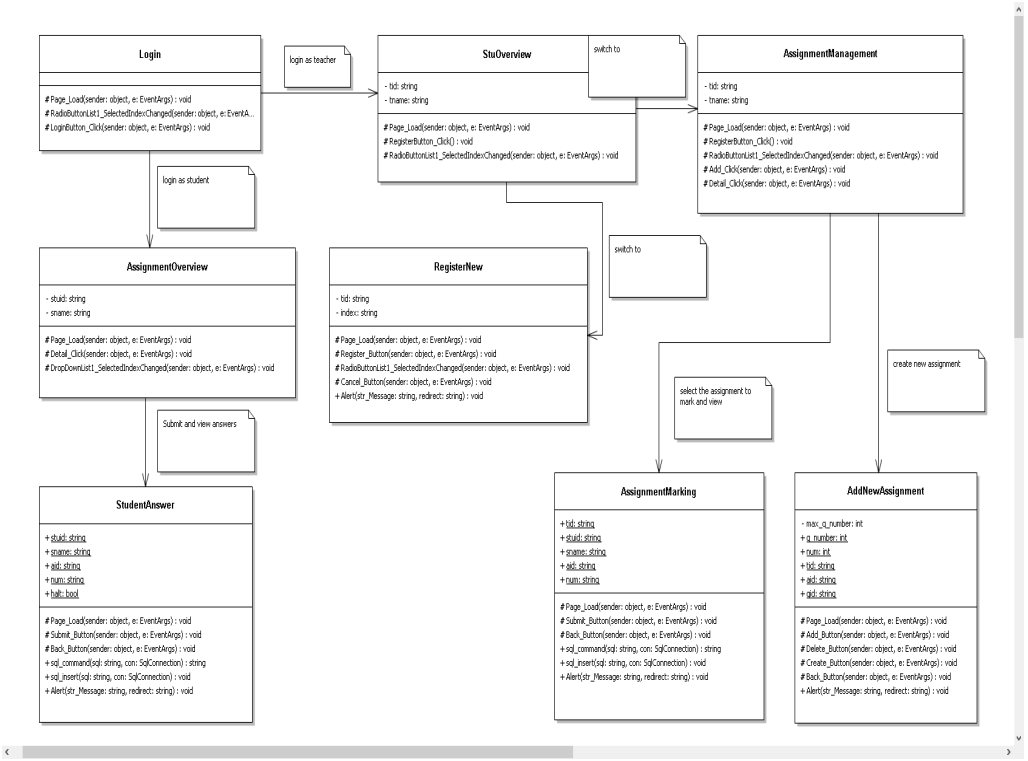

Figure 11: Class Diagram

Entity will store some important messages like assignment name, which course, how many questions does have, weight, and also deadline.

In **Question Entity**, it records question content and the index of question, **aid** is a foreign key because I consider that many questions might be related to one assignment so that we can find the exact question if we know about the **aid** and the **q_index**.

The **Answer Entity** has two foreign keys, one is **qid** and another is **stuid** because for the same question, we need to know which student has answered it. It is the reason why both foreign keys are necessarily required. Additionally, the content of student's answer and teacher's comment will be recorded, too.

The last entity is **Submit Entity**, this entity stores the grade information, submission statement and teacher's comment about the whole assignment. **Submit Entity** has a relationship with **Student** and **Assignment** because it records all assignment information of each student.

2.3 Schema

Figure 13 is the schema of database design shown before.

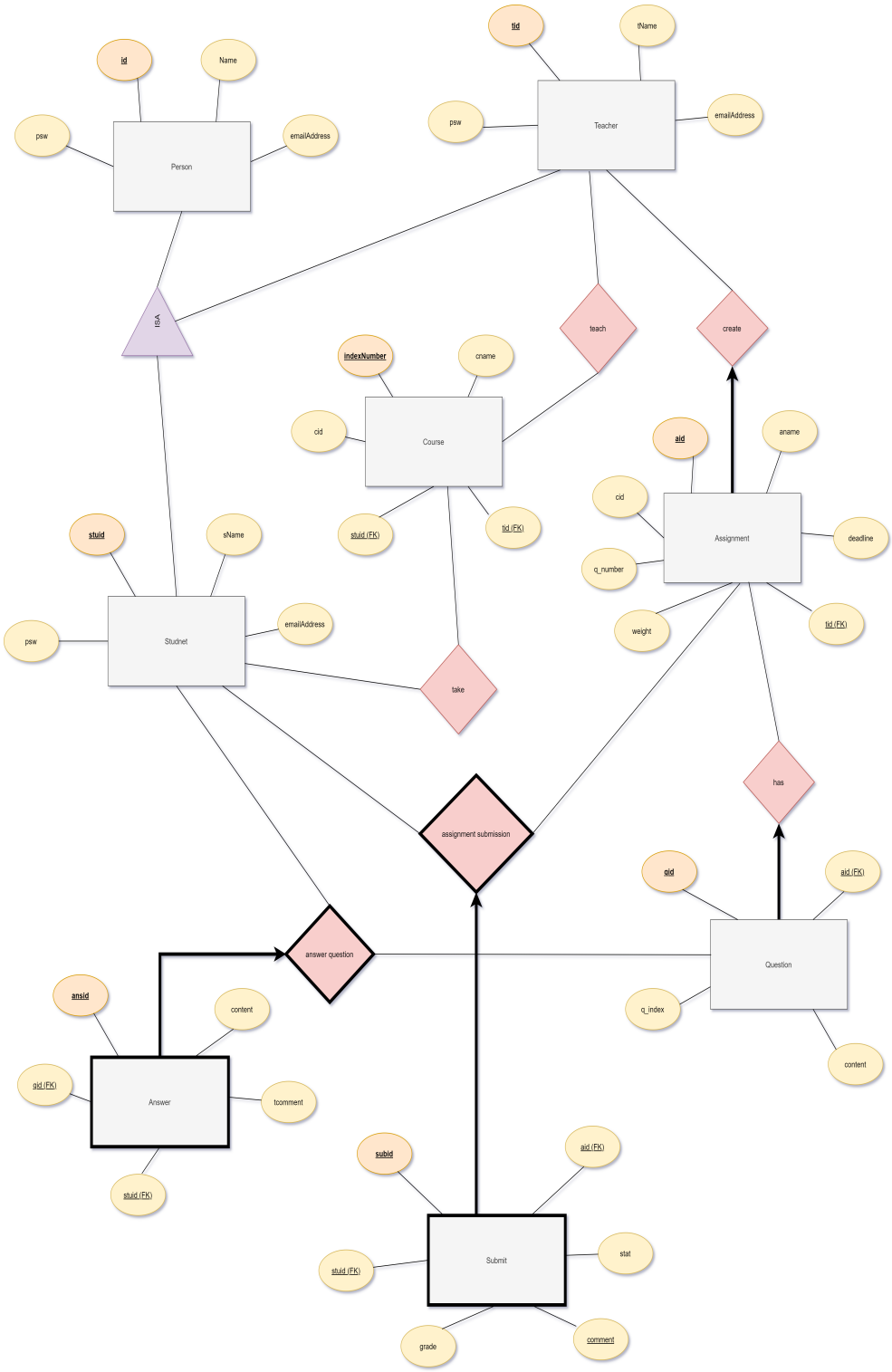

Figure 12: ER Diagram

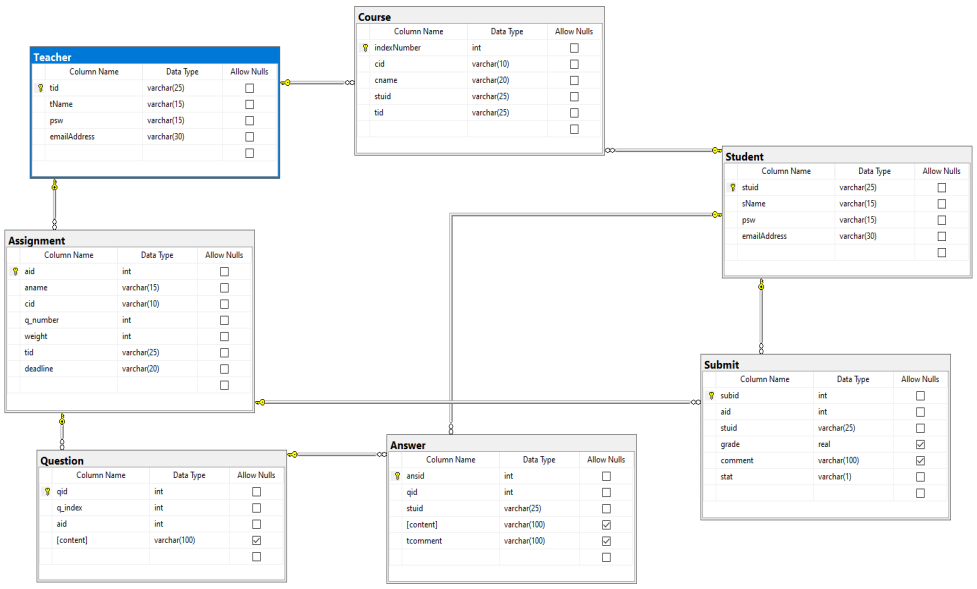

Figure 13: Schema

3 Features of Design

3.1 Basic features noted in DPProject.pdf

3.2 Login/Logout Implementation

To be more realistic, I design a dual-login system for both teacher and student, the user can switch the login type to enter the system. After entering the system, the user can click **logout** button to back to the home page.

3.3 Course Management

I allow the user to take more than one courses and the system will display correct information corresponding to the selected course.

3.4 Variability of Question Number

Question number of each assignment is not fixed, teachers can add or delete questions if they want.

3.5 Student Account Management

In **StuOverview.aspx Page**, teacher can reset student's login password directly by clicking **Edit** button and the data will be updated.

3.6 Friendly GUI

The bootstrap framework provides a user-friendly GUI design which is supported at most devices.

4 Code Fragment

4.1 Database Creation

```
-- create database
CREATE DATABASE OSGS
ON(
        NAME='OSGS',
        FILENAME='C:\EX\OSGS.mdf',
        SIZE=17MB,
        FILEGROWTH=5MB
)

LOG ON
(
        NAME='OSGSLog',
        FILENAME='C:\EX\OSGSLog.ldf',
        SIZE=5MB,
        FILEGROWTH=5MB
)
```

4.2 Table Creation

```
-- create table
USE OSGS
CREATE TABLE Teacher(
        tid varchar(25) NOT NULL PRIMARY KEY,
        tName varchar(15) NOT NULL,
        psw varchar(15) NOT NULL,
        emailAddress varchar(30) NOT NULL
)

CREATE TABLE Student(
        stuid varchar(25) NOT NULL PRIMARY KEY,
        sName varchar(15) NOT NULL,
        psw varchar(15) NOT NULL,
        emailAddress varchar(30) NOT NULL
)
```

```
CREATE TABLE Course(
    indexNumber INT NOT NULL PRIMARY KEY,
        cid varchar(10) NOT NULL,
        cname varchar(20) NOT NULL,
        stuid varchar(25) NOT NULL FOREIGN KEY REFERENCES
        Student(stuid),
        tid varchar(25) NOT NULL FOREIGN KEY REFERENCES
        Teacher(tid)
)

CREATE TABLE Assignment(
    aid INT NOT NULL PRIMARY KEY,
        aname varchar(15) NOT NULL,
        cid varchar(10) NOT NULL,
        q_number INT NOT NULL DEFAULT 0,
        [weight] INT NOT NULL,
        tid varchar(25) NOT NULL FOREIGN KEY REFERENCES
        Teacher(tid),
        deadline varchar(20) NOT NULL,
)

CREATE TABLE Question(
    qid INT NOT NULL PRIMARY KEY,
        q_index INT NOT NULL,
        aid INT NOT NULL FOREIGN KEY REFERENCES Assignment(aid),
        content varchar(100) DEFAULT NULL
)

CREATE TABLE Answer(
    ansid INT NOT NULL PRIMARY KEY,
        qid INT NOT NULL FOREIGN KEY REFERENCES Question(qid),
        stuid varchar(25) NOT NULL FOREIGN KEY REFERENCES
        Student(stuid),
        content varchar(100) DEFAULT NULL,
        tcomment varchar(100) DEFAULT NULL
)

CREATE TABLE Submit(
    subid INT NOT NULL PRIMARY KEY,
        aid INT NOT NULL FOREIGN KEY REFERENCES Assignment(aid),
        stuid varchar(25) NOT NULL FOREIGN KEY REFERENCES
        Student(stuid),
        grade real DEFAULT 0.0,
        comment varchar(100) DEFAULT NULL,
        stat varchar(1) NOT NULL DEFAULT 'N'
```

```
)
```

4.3 Data Update

In **StuOverview.aspx**, the teacher can edit student's information:

```
UPDATE Student
SET sName = @sName, psw=@psw, emailAddress=@emailAddress
WHERE stuid = @stuid
```

4.4 Data Query

I use the query of **Assignment Management.aspx** as an example:

```
SELECT A.aid, A.aname, S.stuid, S.sName, A.weight AS [weight(%)],
A.q_number, A.deadline, SU.stat, SU.grade, SU.comment
FROM Assignment AS A
JOIN Submit AS SU
ON SU.aid = A.aid
JOIN Course AS C
ON C.cid=A.cid AND C.tid = A.tid AND C.cid = @cid AND
C.tid = @tid AND SU.stuid = C.stuid
JOIN Student AS S
ON S.stuid = C.stuid
```

We can see that all the columns totally involve 3 tables (**Assignment**, **Submit** and **Course**), so the first operation is definitely joining all three tables. And then, we need to check:

- The current teacher exactly teaches this course (probably there are 2 teachers teach CS108 together).
- The students exactly take the course of this teacher.
- The result belongs to current selected course (if selected CS101, some of the students will miss).
- The submission is corresponding to those students.

4.5 Data Insertion

SQL Connecting:

```
SqlConnection con = new SqlConnection("Data Source=DESKTOP
-KV5M48K;Initial Catalog=OSGS;Integrated Security=True");

con.Open();
```

Insertion Method:

```
// insert, update and delete
public void sql_insert(string sql, SqlConnection con)
{
    SqlCommand cmd = new SqlCommand(sql, con);
    try
    {
        cmd.ExecuteNonQuery();
    }
    catch
    {
        halt = true;
        Response.Write("<script type='text/javascript'>
        alert('Insert Fail!');</script>");
    }

}
```

Insert Question Table:

```
sql_que = "INSERT INTO Question(qid, q_index, aid, content)
VALUES (" + qid + ", 1, " + aid + ", '" + Q_one.Text + "')";
```

Insert Assignment Table:

```
sql_assign = "INSERT Assignment(aid, aname, cid, q_number,
[weight], tid, deadline)
VALUES(" + aid + ", '" + aname.Text + "', '" + cid.Text +
"', " + num.ToString() +", " + weight.Text +", '" + tid +
"', '" + ddl.Text +"')";
```

Insert Answer Table:

```
sql = "INSERT Answer(ansid, qid, stuid, content, tcomment)
VALUES(" + ansid.ToString() + ", " + qid_1 + ", '" + stuid
+"', '" + ans.Text + "', '')";
```

Insert Submit Table:

```
sql1 = "INSERT Submit(subid, aid, stuid, grade, comment,
stat)
VALUES(" + subid.ToString() + ", " + aid + ", '" +
tmp[0].ToString() + "', 0.0, ' ', 'N')";
```

How to assign a new ID as primary key (like new aid, qid, ansid, subid):

```
// get the largest ansid
sql = "SELECT TOP 1 ansid FROM Answer ORDER BY ansid DESC";
ansid = int.Parse(sql_command(sql, con));

ansid++; // a new ansid
```

In conclusion, all the parameters of previous insertion methods can be divided into 3 types: new ids (**aid, qid, ansid, subid**), passed by **Textbox** (has **.Text**), passed by URL (**stuid** or **tid**). They all can be connected as **String Type** into SQL.

References

[1] SqlDataReader Class
`https://docs.microsoft.com/zh-tw/dotnet/api/system.data.sqlclient.sqldatareader?view=dotnet-plat-ext-5.0`

[2] How To Jump With Alert
`https://www.cnblogs.com/mic86/articles/1779921.html`