华南理工大学

South China University of Technology

Database System Concepts

Experiment 2 Report

(Program Manual & Design Document)

Student Name：Jevin Laudo

Student Number：202069990040

Class：CST2020

Teacher：陈俊龙、汪秀敏、余志文

Contents

SCUT Management Information System

Management information systems (MIS) are the backbones of most organizational operations today. It is a tool to gather and analyse data, as well as to report information gathered as an aid in decision-making.

Moreover, these systems are also utilized throughout universities across the globe, generally as an essential component in the management of academic activities and internal affairs. This project aims to simulate and develop a functional management information system for the college of Computer Science at the South China University of Technology, where academic-related information is stored and managed through the usage of SQL database, a relational database management system.

This paper covers the program design, program manual, database design, general UI design and components utilized, as well as SQL commands utilized to achieve the desired result.

Program Design

The program is developed with Microsoft Visual Studio 2022, with the .NET graphical class library Windows Forms, which is responsible for managing the Graphical User Interface (GUI) of the project. Functionalities of the program is written in the C# programming language, whereas data is stored, retrieved, and managed through Microsoft SQL Server 2019 version 15.0.2000.5.

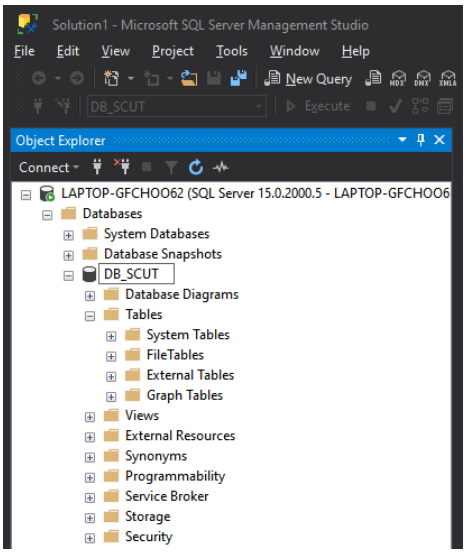
SQL Server Database Setup

The initial pre-development stage of the program design is to construct the SQL Server database that the program will interact with. This stage is done with the aid of SQL Server Management Studio (SSMS), where connections to SQL Server database through localhost will be configured.

As observed in figure 1, a new database has been created named “DB_SCUT”, where information for the MIS will be stored within.

Figure 1

Newly Generated SQL Server Database

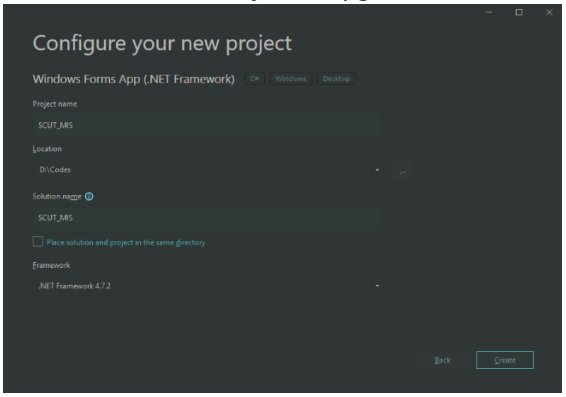


Project Solution Setup

Within Microsoft Visual Studio IDE, the project is configured as a Windows Forms app, running on the .NET framework with C# as its programming language.

Figure 2

Visual Studio 2022 Project Configuration



Within the project solution, the file `app.config` contains the configuration setup made for the project, including the .NET version support, as well as the connection strings for the program to connect to the SQL Server database.

Figure 3

Visual Studio 2022 app.config Project Configuration

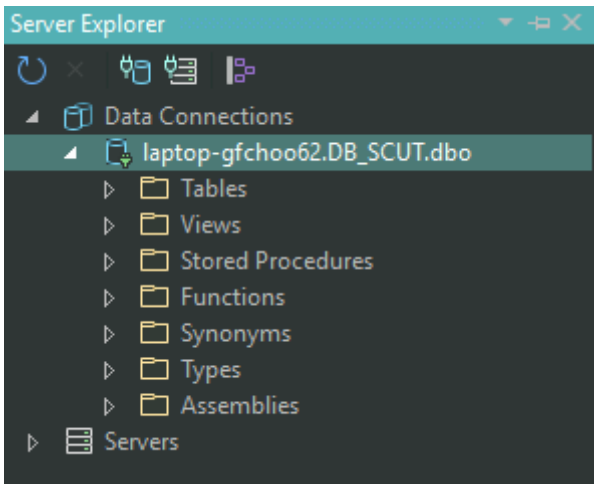
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <connectionStrings>
        <add name="database"
             connectionString="Data Source=LAPTOP-GFCHOO62;
                Initial Catalog=DB_SCUT;
                Integrated Security=True"
             providerName="System.Data.SqlClient"/>
    </connectionStrings>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
    </startup>
</configuration>
```

SQL Connection Configuration

As presented in figure 3, the SQL database connection string for the project is preconfigured and is contained within the `<connectionStrings>` tags. The string will be utilized within the project whenever the SQL database is accessed. This approach is chosen for the project implantation to promote better maintainability, as well as ease of usage.

Figure 4

Server Explorer View for SQL Database Connection



Initial SQL Database Table Setup

Core system information table construction. Following the requirements, four tables are used within the system. Among which are the students' information table, teachers' information table, courses information table, as well as the course choosing table. `students`, `teachers`, `courses`, and `choose` are the names of the tables respectively.

Figure 5

Core System Information Table Creation Commands

```
CREATE TABLE students(
    sid VARCHAR(10) NOT NULL PRIMARY KEY,    -- Student ID
    sname NVARCHAR(20) NOT NULL,             -- Student Name
    sex VARCHAR(6) CHECK(sex = 'male' or sex = 'female') NOT NULL, -- Student Sex
    entrance_age INT CHECK(entrance_age >= 10 and entrance_age <= 50) NOT NULL, -- Student Entrance Age
    entrance_year INT NOT NULL,                  -- Student Entrance Year
    class VARCHAR(20) NOT NULL                   -- Student Class
);

CREATE TABLE teachers(
    tid VARCHAR(5) NOT NULL PRIMARY KEY,     -- Teacher ID
    tname NVARCHAR(20) NOT NULL,             -- Teacher Name
    course NVARCHAR(20) NOT NULL             -- Teacher Course
);

CREATE TABLE courses (
    cid VARCHAR(7) NOT NULL PRIMARY KEY,     -- Course ID
    cname NVARCHAR(20) NOT NULL,             -- Course Name
    tid VARCHAR(5) FOREIGN KEY REFERENCES teachers(tid) ON DELETE SET NULL,    -- Teacher ID
    credit DECIMAL(4,2) NOT NULL,            -- Course Credit
    grade VARCHAR(20) NOT NULL,              -- Course Grade
    cancel_year INT                          -- Course Cancel Year
);

CREATE TABLE choose (
    sid VARCHAR(10) NOT NULL FOREIGN KEY REFERENCES students(sid) ON DELETE CASCADE,  -- Student ID
    cid VARCHAR(7) FOREIGN KEY REFERENCES courses(cid) ON DELETE SET NULL,            -- Course ID
    tid VARCHAR(5) FOREIGN KEY REFERENCES teachers(tid) ON DELETE SET NULL,           -- Teacher ID
    score FLOAT,                 -- Score
    chosen_year INT NOT NULL     -- Chosen Year
);
```

Authorities' information table construction. Three tables are created for each respective authority level: students, teachers, and administrators. Each of the separate tables contains usernames, as well as its associated password, which is a prerequisite before access to the program is granted to the user.

Figure 6

Authorities Table Creation Commands

```
CREATE TABLE account_students (
        username VARCHAR(15) NOT NULL,
        password VARCHAR(15) NOT NULL
);

CREATE TABLE account_teachers (
        username VARCHAR(15) NOT NULL,
        password VARCHAR(15) NOT NULL
);

CREATE TABLE account_admins (
        username VARCHAR(15) NOT NULL,
        password VARCHAR(15) NOT NULL
);
```

Sample Data Insertion

Several sample data are inserted into the database for program testing and observation, which is done within the IDE with the SQL commands as the following, shown in figure 7.

Figure 7

Table Insertion Queries for Students, Teachers, Courses, and Course Choosing

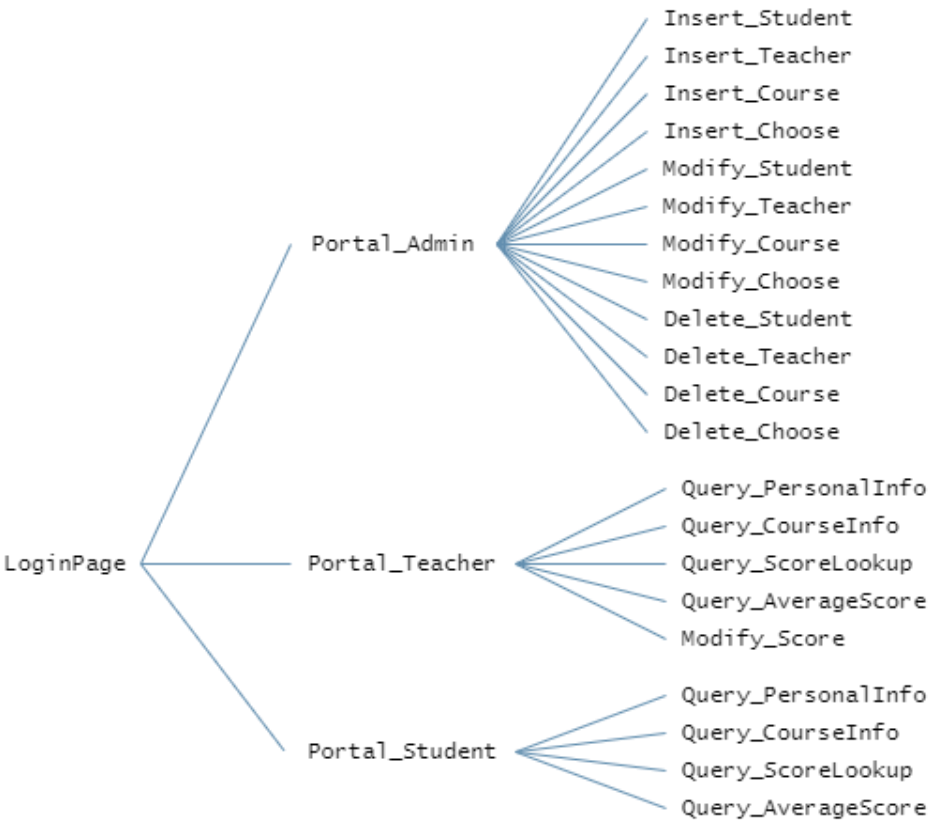
```
INSERT INTO students
         (sid, sname, sex, entrance_age, entrance_year, class)
         VALUES ('1000000001', 'Tom Scott', 'male', 19, 2020, 'CST2020');
INSERT INTO students
         (sid, sname, sex, entrance_age, entrance_year, class)
         VALUES ('1000000002', 'Sheila Clives', 'female', 21, 2020, 'CST2020');
INSERT INTO students
         (sid, sname, sex, entrance_age, entrance_year, class)
         VALUES ('1000000003', 'Mary Jane', 'female', 20, 2021, 'CST2021');
INSERT INTO students
         (sid, sname, sex, entrance_age, entrance_year, class)
         VALUES ('1000000004', 'Thomas Zhang', 'male', 18, 2021, 'CST2021');
INSERT INTO students
         (sid, sname, sex, entrance_age, entrance_year, class)
         VALUES ('1000000005', 'Jeffrey Ho', 'male', 19, 2021, 'CST2022');
INSERT INTO students
         (sid, sname, sex, entrance_age, entrance_year, class)
         VALUES ('1000000006', 'Clay Moore', 'male', 23, 2021, 'CST2022');
INSERT INTO students
         (sid, sname, sex, entrance_age, entrance_year, class)
         VALUES ('1000000007', 'Danny Flemmings', 'male', 18, 2021, 'CST2022');

INSERT INTO teachers
         (tid, tname, course)
         VALUES ('T0001', 'Jefferson Logatto','Linear Algebra');
INSERT INTO teachers
         (tid, tname, course)
         VALUES ('T0002', 'Matt Rockald', 'Intro to CS');
INSERT INTO teachers
         (tid, tname, course)
         VALUES ('T0003', 'Miles Hughes', 'C++ Programming');
INSERT INTO teachers
         (tid, tname, course)
         VALUES ('T0004', 'Scarlet Rosalia', 'Java Programming');
INSERT INTO teachers
         (tid, tname, course)
         VALUES ('T0005', 'Ben Owen', 'Data Structures');
INSERT INTO teachers
         (tid, tname, course)
         VALUES ('T0006', 'Richard Higgings', 'Algorithms');

INSERT INTO courses
         (cid, cname, tid, credit, grade)
         VALUES ('C000001', 'Linear Algebra', 'T0001', 3.0, '1');
INSERT INTO courses
         (cid, cname, tid, credit, grade)
         VALUES ('C000002', 'Intro to CS', 'T0002', 2.0, '1');
INSERT INTO courses
         (cid, cname, tid, credit, grade)
         VALUES ('C000003', 'C++ Programming', 'T0003', 4.0, '1');
INSERT INTO courses
         (cid, cname, tid, credit, grade)
         VALUES ('C000004', 'Java Programming', 'T0004', 4.0, '2');
INSERT INTO courses
         (cid, cname, tid, credit, grade)
         VALUES ('C000005', 'Data Structures', 'T0005', 3.0, '3');
INSERT INTO courses
         (cid, cname, tid, credit, grade)
         VALUES ('C000006', 'Algorithms', 'T0006', 3.5, '3');
```

```
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000001', 'C000001','T0001', 98, 2020);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000001', 'C000002','T0002', 91, 2020);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000001', 'C000003','T0003', 93, 2020);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000001', 'C000004','T0004', 87.9, 2021);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000001', 'C000005','T0005', 92, 2022);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000001', 'C000006','T0006', 86.67, 2022);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000002', 'C000001','T0001', 92.1, 2020);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000002', 'C000002','T0002', 87.6, 2020);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000002', 'C000003','T0003', 77.9, 2020);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000002', 'C000004','T0004', 93, 2021);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000002', 'C000005','T0005', 88, 2022);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000002', 'C000006','T0006', 85.3, 2022);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000003', 'C000001','T0001', 77.9, 2021);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000003', 'C000002','T0002', 82.1, 2021);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000003', 'C000003','T0003', 73, 2021);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000003', 'C000004','T0004', 84, 2022);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000004', 'C000001','T0001', 83.6, 2021);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000004', 'C000002','T0002', 91.2, 2021);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000004', 'C000003','T0003', 70.1, 2021);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000004', 'C000004','T0004', 66.6, 2022);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000005', 'C000001','T0001', 91, 2021);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000005', 'C000002','T0002', 93.5, 2021);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000005', 'C000003','T0003', 81.2, 2021);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000006', 'C000001','T0001', 77.1, 2021);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000006', 'C000002','T0002', 90.1, 2021);
INSERT INTO choose
          (sid, cid, tid, score, chosen_year)
          VALUES ('1000000006', 'C000003','T0003', 91.2, 2021);
```

UI Design with Windows Forms

As aforementioned, the UI components of the program are powered by Windows Forms (WinForms) .NET UI class library. With the utilities and tools provided with Visual Studio 2022, 21 UI pages are configured to support the program's complete functionality set, where relationships between each UI page is illustrated in figure 8.

Figure 8
UI Page Connections Diagram



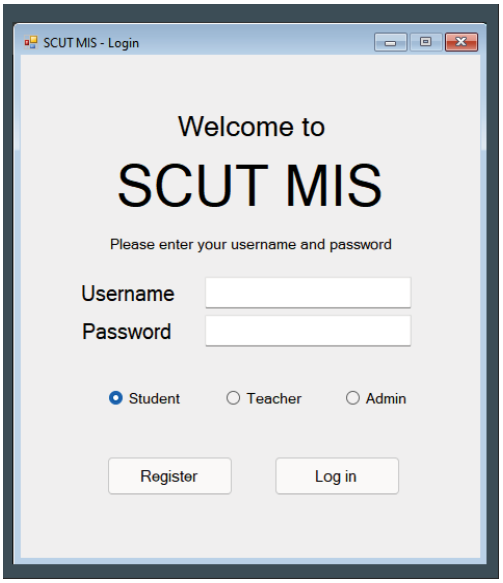
As shown in figure 8, the program is divided based on the authority levels determined during the login process, where administrators are granted the access to insert, modify, and delete entries of students, teachers, courses, or course choosing. Teachers and students share similar feature accessibilities, where they are permitted to query students' personal info, course info, view scores, as well as performing data lookup on average scores achieved.

Moreover, teachers are also given the additional privilege to perform alterations to the scores received by the students.

To access a certain authority level, users need to provide username and password, as well as selecting the account type from the radio button before registering or logging into an account. The username inputted in the textbox will be displayed in the portal display following it.

Figure 9

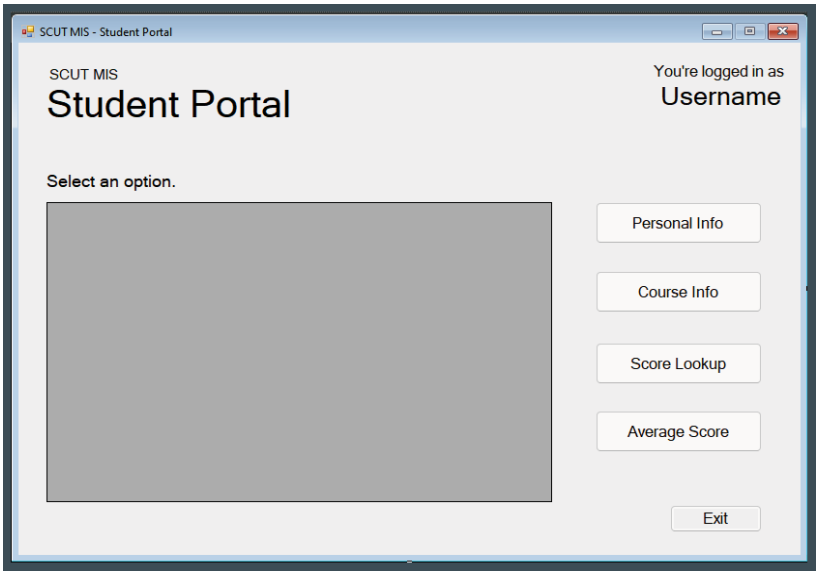
Login Page UI Design



As illustrated by figure 10, after logging into the system as a student, the following student portal dialogue will be displayed onto the user.

Figure 10

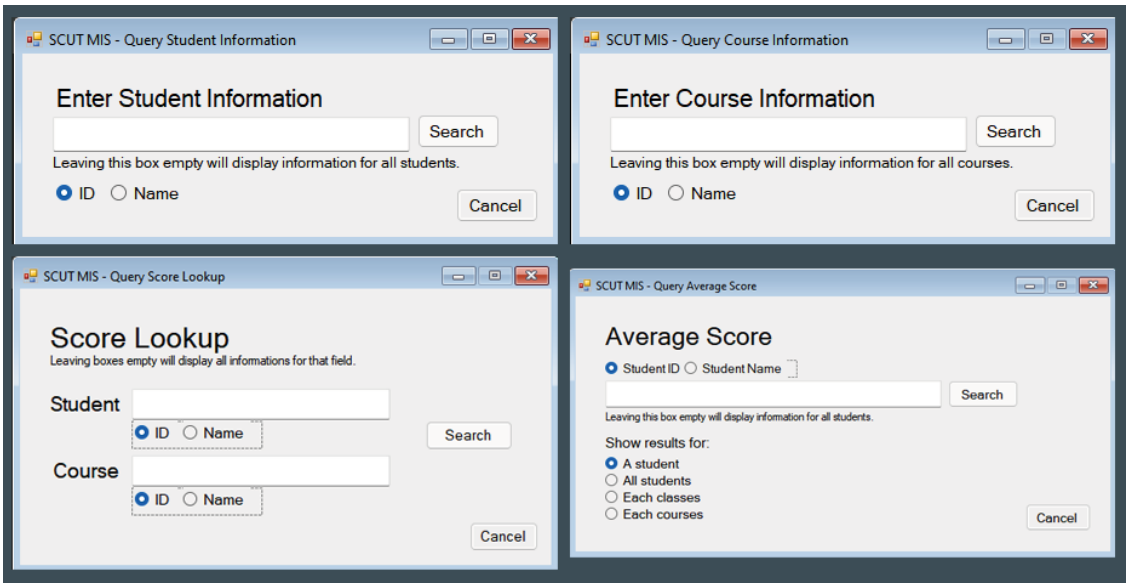
Student Portal UI Design



Each of the buttons from the student portal will generate the following pop-up dialogue respectively.

Figure 11

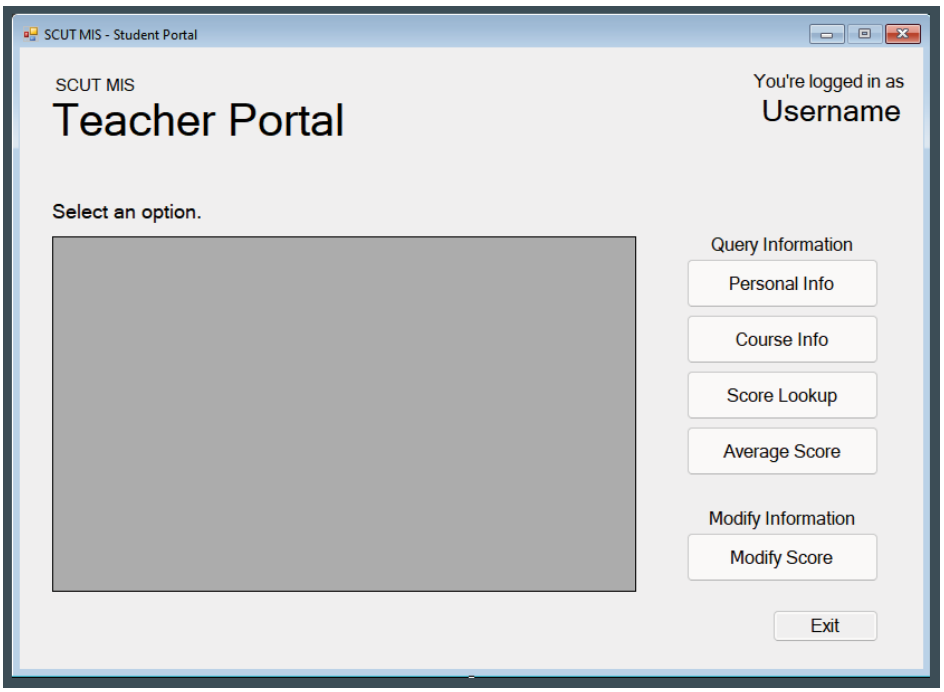
UI Design for Student Portal's Popup Dialogues



Logging in into the program with a teacher's account will display a teacher portal instead, whose UI is illustrated in figure 12.

Figure 12

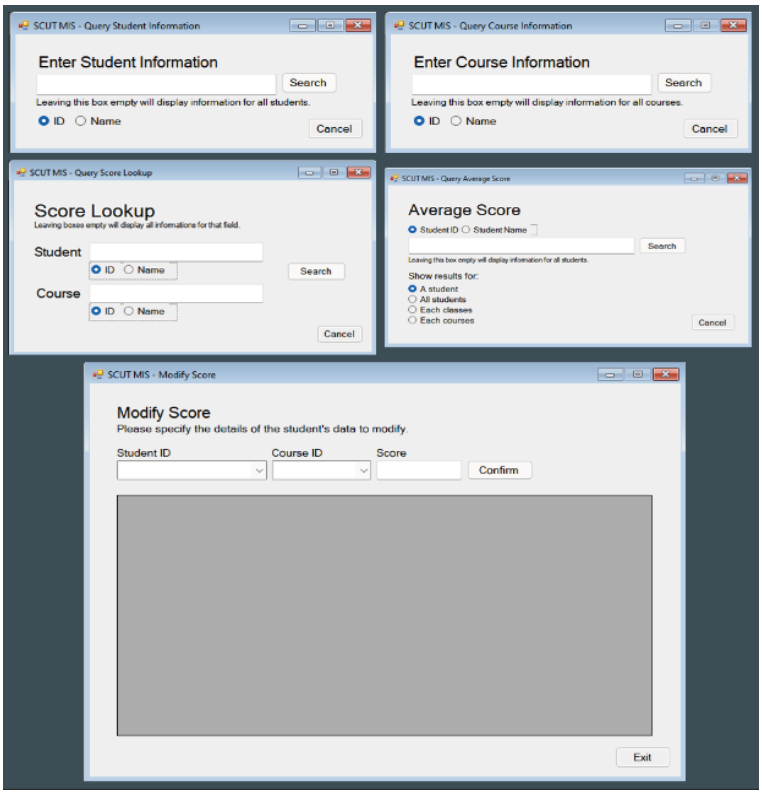
Teacher Portal UI Design



Similarly, the pop-up dialogues available from the teacher's portal is shown in the following.

Figure 13

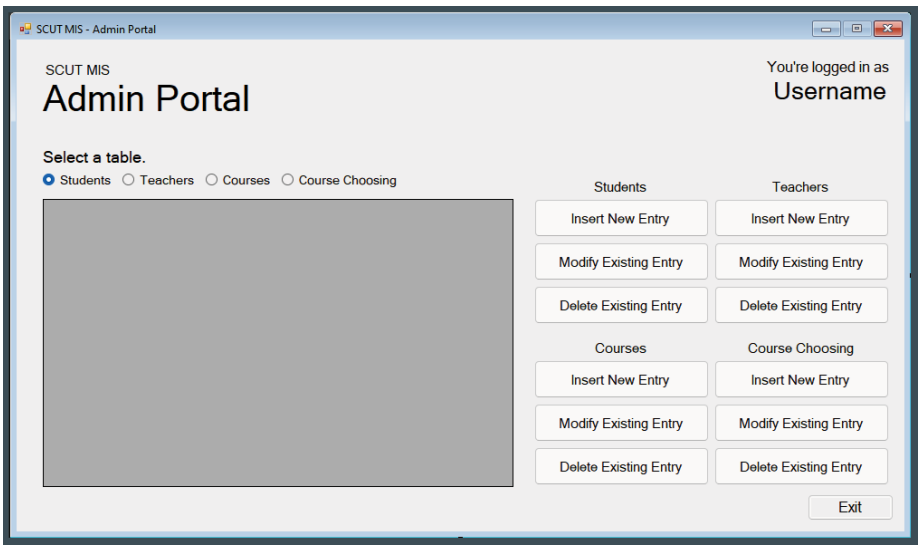
UI Design for Teacher Portal's Popup Dialogues



By selecting the administrator logging in method, the admin portal dialogue box will be displayed. Its UI design is shown in figure 14.

Figure 14

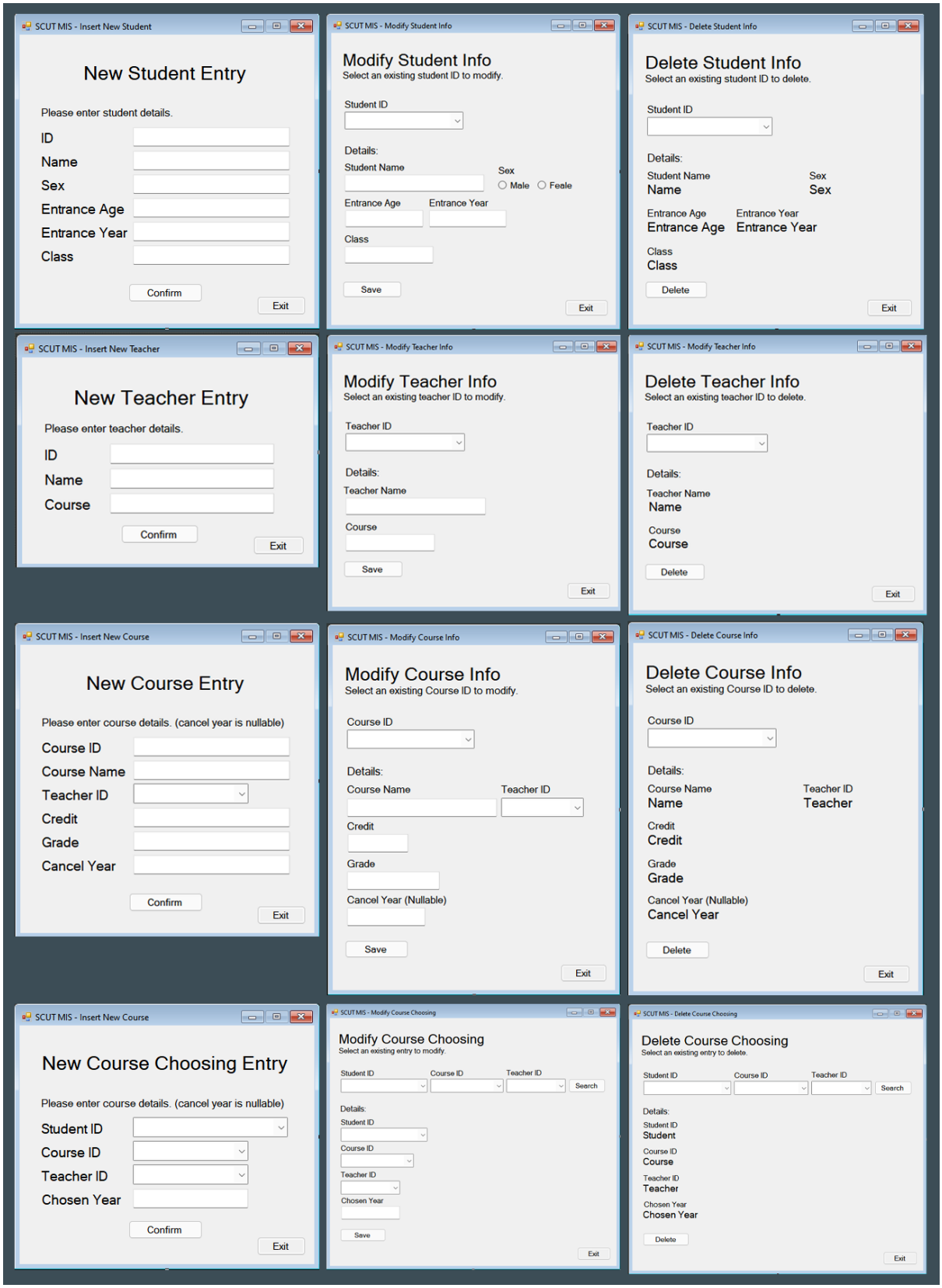
Admin Portal UI Design



Each of the buttons of the admin portal corresponds to individual pop-up dialogue boxes, shown in the following,

Figure 15

UI Design for Admin Portal's Popup Dialogues



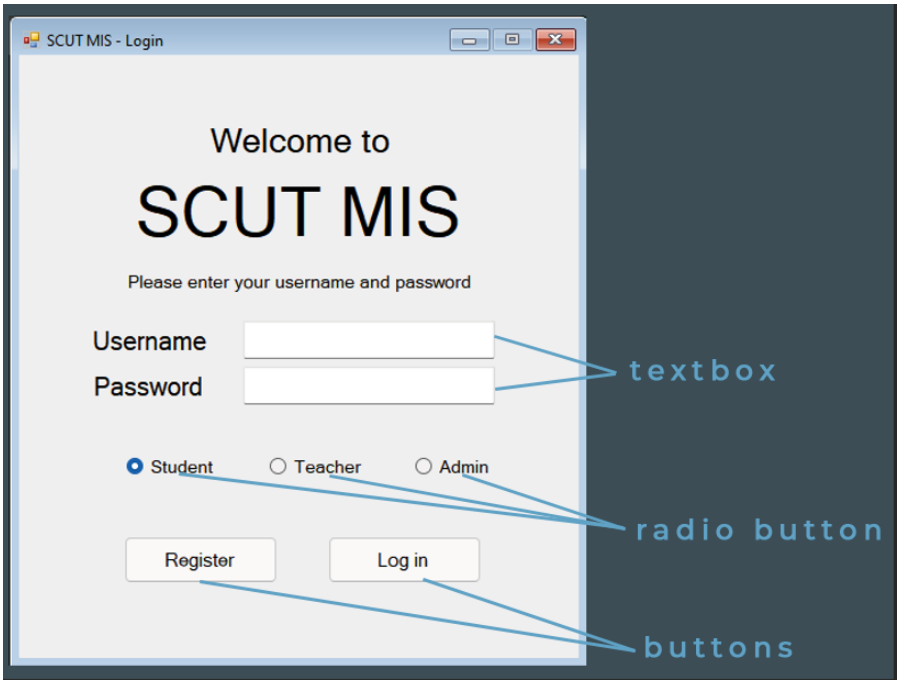
Program Manual

UI Components and Layout

Login Page

The login page is comprised of two textboxes, two buttons, and three radio buttons. The two textboxes, `textbox_username` and `textbox_password`, are responsible for username and password input respectively. Meanwhile, the three radio buttons, `rbtn_student`, `rbtn_teacher`, and `rbtn_admin` are responsible for login method, whereas the two buttons, `btn_register` and `btn_login` are responsible for registering and logging users based on the username and passwords provided in the textboxes.

Figure 16
Login Page UI Components



Failsafe mechanisms. As indicated in figure 17 and 18, the login page also provide failsafe mechanisms to handle errors, such as unregistered accounts, incorrect passwords, as well as other invalid credentials that users might provide during their login attempts. During such scenarios, the warning label will inform the user of the error cause, as well as changing colour to red.

Figure 17

Empty username or password error

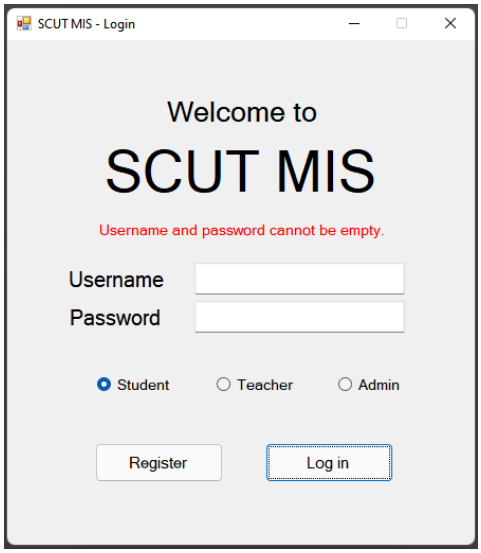
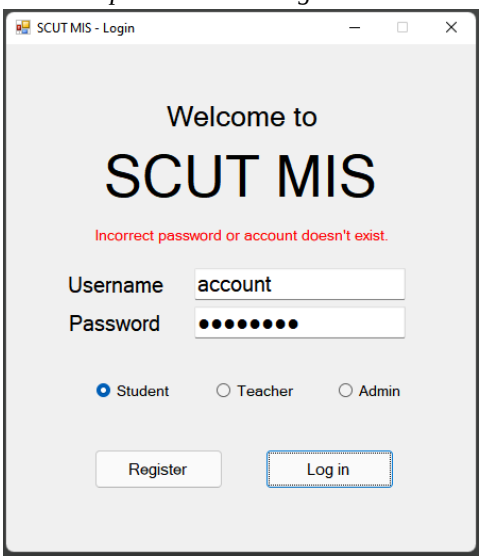


Figure 18

Incorrect password or unregistered account error



Successful actions. When a registration or login attempt is successful, the warning label will turn to green, and notify users of the successful action.

Figure 19

Successful registration message

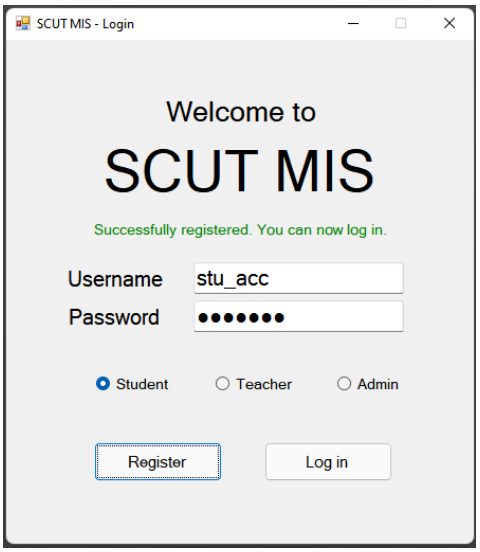
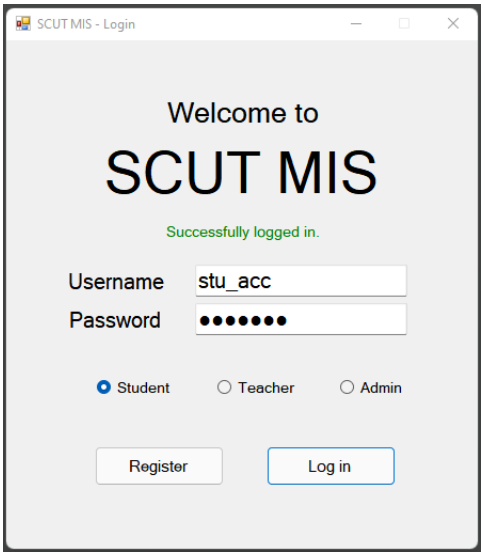


Figure 20

Successful login message

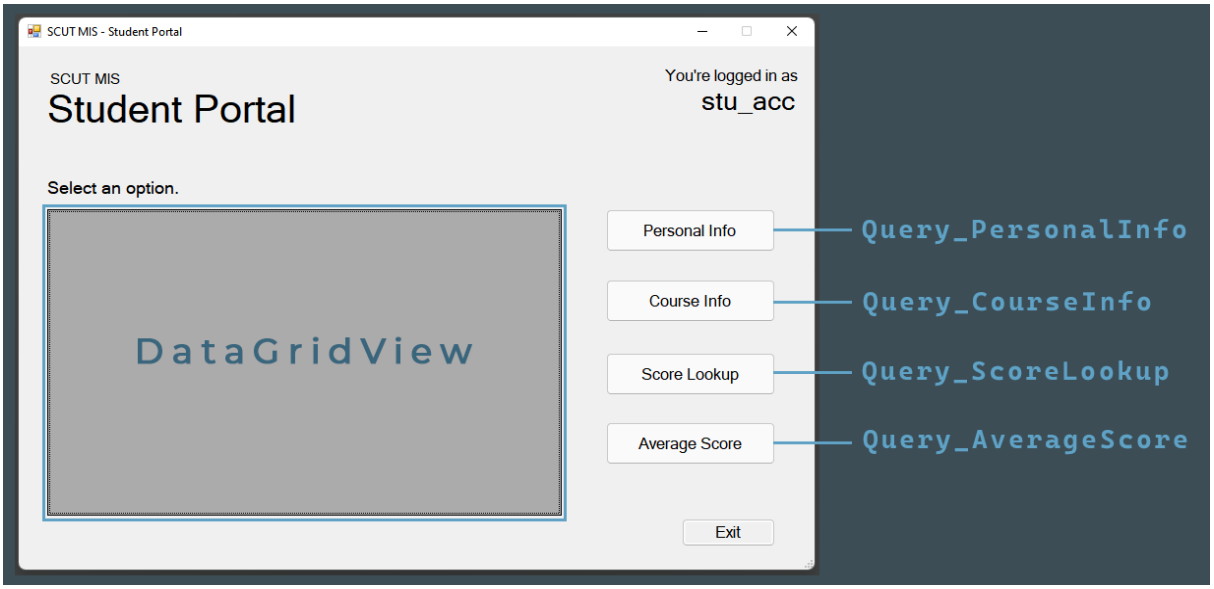


Upon successful login attempts, the program will display portals based on the login credentials, as well as the login method selected from the three radio buttons. The three portals are student portal, teacher portal, as well as admin portals.

Student Portal

The username used during the login process will be displayed by the label on the top right of the dialogue box. Moreover, the portal contains a `DataGridView` component which will display data tables queried by selecting any of the buttons on its right. Among the buttons, `btn_PersonalInfo` will generate a pop-up dialogue of `Query_PersonalInfo` WinForms page, while the remaining buttons, `btn_CourseInfo`, `btn_ScoreLookup`, and `btn_AverageScore` will generate `Query_CourseInfo`, `Query_ScoreLookup`, and `Query_AverageScore` respectively. Finally, when users wish to close the dialogue, they can click on the cross-shaped button on the top right of the window or click on the exit button provided on the bottom right of the dialogue box.

Figure 21
Student Portal UI Components

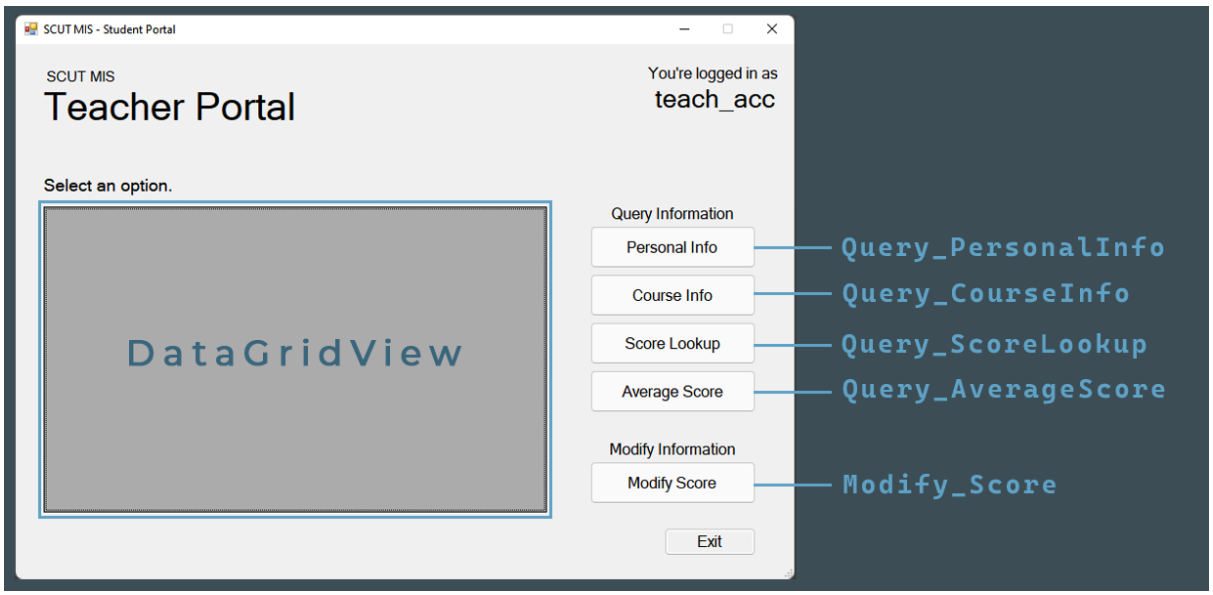


Teacher Portal

The teacher's portal is nearly identical to the student's portal. The difference lies in its fourth button, `btn_ModifyScore`, which will generate the pop-up dialogue `Modify_Score`, where teachers can alter scores assigned to students. Additionally, the buttons are also classified into two groups based on its utility, to query information or to modify information.

Figure 21

Teacher Portal UI Components

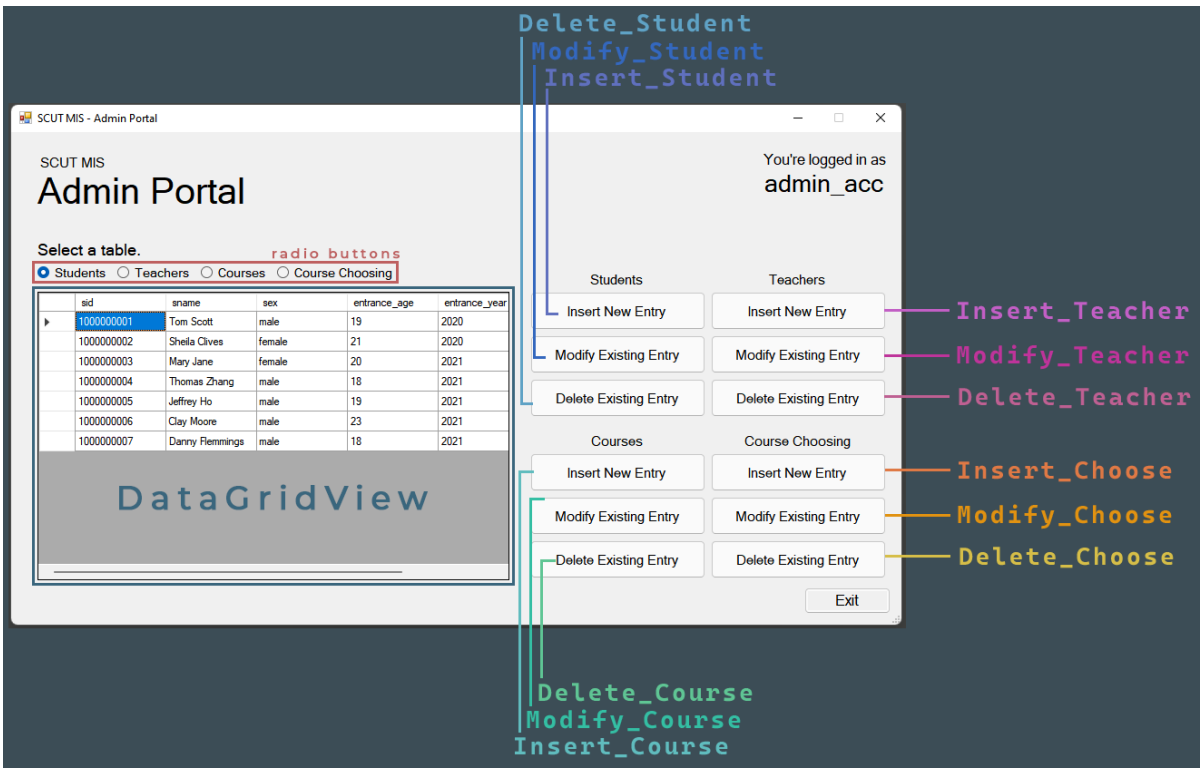


Admin Portal

Similar to other portals, the topmost section of the admin portal reveals general information of the window, as well as providing the username of the administrator account accessed.

Figure 22

Admin Portal UI Components



The admin portal also has one `DataGridView` layout. However, the usage of the layout greatly differs from the portals prior, as it reveals all information of students, teachers, courses, and course choosing as determined by the selected radio buttons on top of it. Figure 23 reveals the inner workings of the feature, where clicking on any one of the four will issue a SQL command to retrieve all information of each table.

Figure 23
Admin Portal Radio Button Source Code

```
private void LoadDataGridView()
{
    string sqlQuery = "SELECT * FROM ";

    if (rbtn_Students.Checked)
        sqlQuery += "students";
    else if (rbtn_Teachers.Checked)
        sqlQuery += "teachers";
    else if (rbtn_Courses.Checked)
        sqlQuery += "courses";
    else if (rbtn_Choose.Checked)
        sqlQuery += "choose";
    else return;

    using (SqlConnection sqlConnection = new System.Data.SqlClient.SqlConnection(SqlHelper.CnnVal("database")))
    {
        using (SqlDataAdapter dataAdapter = new SqlDataAdapter(sqlQuery, sqlConnection))
        {
            using (SqlCommandBuilder sqlCommandBuilder = new SqlCommandBuilder(dataAdapter))
            {
                DataTable dataTable = new DataTable();
                dataAdapter.Fill(dataTable);
                dataGridView.DataSource = dataTable;
            }
        }
    }
}

4 references
private void rbtn_OnClick(object sender, EventArgs e) => LoadDataGridView();
```

Moreover, the buttons of the portal are grouped based on their target table. The buttons in students' group will generate the `Insert_Student`, `Modify_Student`, as well as `Delete_Student` dialogue boxes. The teachers' group will generate the `Insert_Teacher`, `Modify_Teacher`, and `Delete_Teacher` dialogue boxes, whereas the courses' group will generate the `Insert_Course`, `Modify_Course`, as well as `Delete_Course` dialogue boxes. The last group, course choosing's, will generate the `Insert_Choose`, `Modify_Choose`, and `Delete_Choose` dialogue boxes.

Query_PersonalInfo

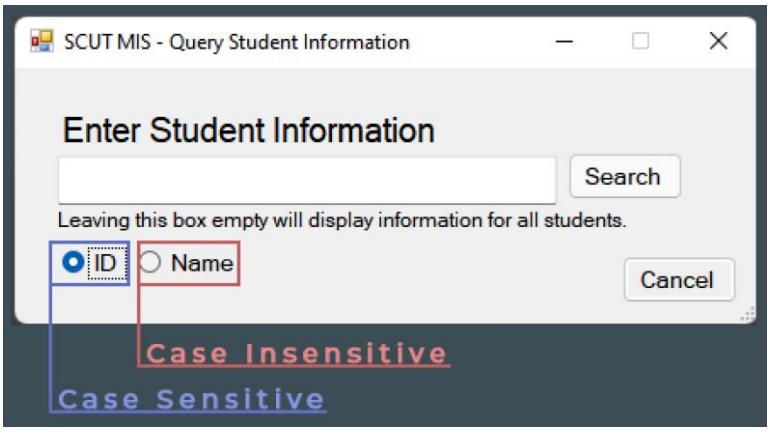
As aforementioned, this UI dialogue box is shown when a student's personal information is about to be queried. Figure 24 shows that there are two ways to provide

students' information identity, through either their student ID, or their names. These searching methods differ in the characters they receive. While the former is case-sensitive, the latter is case-insensitive.

Moreover, the warning label under the text box informs the user that if no information is provided, the system will query information for all students within the database.

To finalize the query action, users need to click on the search button, or the cancel button to close the dialogue box.

Figure 24
Query_PersonalInfo UI Components



Failsafe mechanisms. In the event when the student ID or name does not match any of the records, the warning label will turn red and display the error message.

Figure 25
Query_PersonalInfo ID Not Found Error

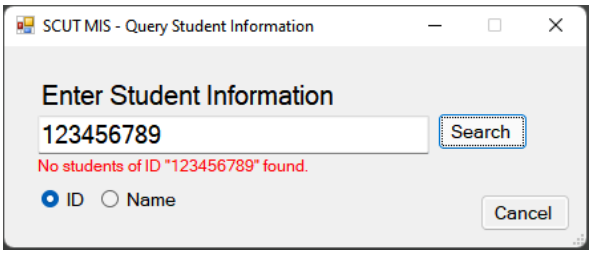
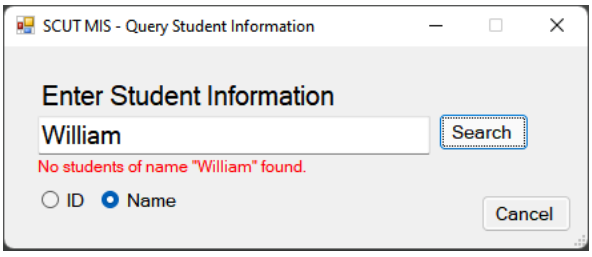


Figure 26
Query_PersonalInfo Name Not Found Error



When no error is found in the information provided, the system will instead retrieve the requested data and close the pop-up box. The retrieved data will be displayed as a table in the `DataGridView` within the portal where the dialogue box was generated from.

Query_CourseInfo

Similar to the previous dialogue box, `Query_CourseInfo` receives two type of input, course id or course name, which are case-sensitive and case-insensitive respectively. Its warning label also informs the user that if no information is provided, all course information will be shown. Additionally, the failsafe mechanism, as well as its search and cancel button also work similarly to the dialogue box prior.

Figure 27

Query_CourseInfo UI Components

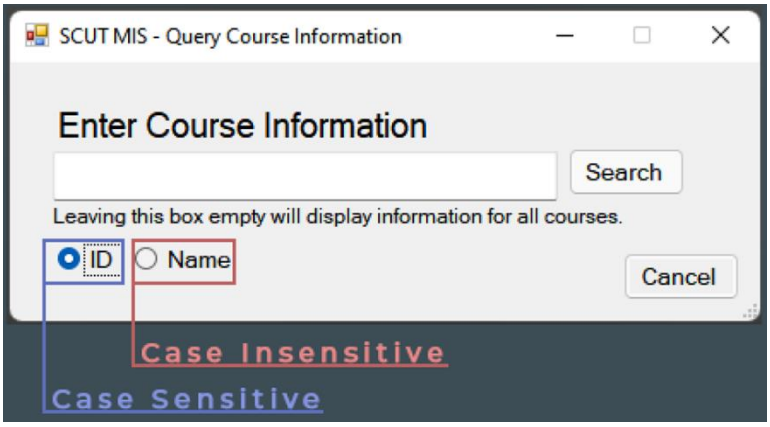


Figure 28

Query_CourseInfo ID Not Found Error

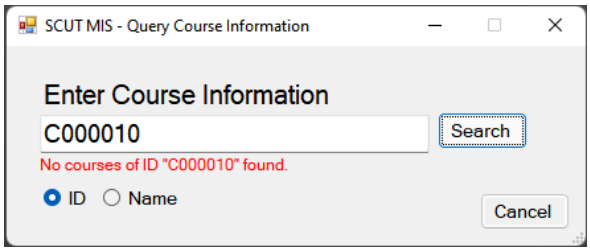
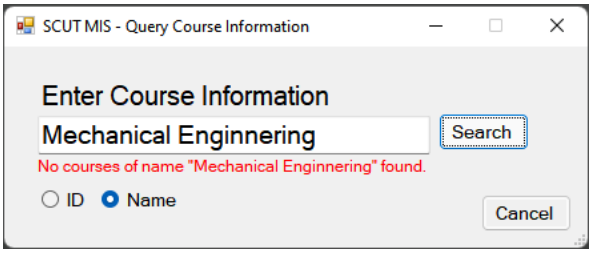


Figure 29

Query_CourseInfo Name Not Found Error



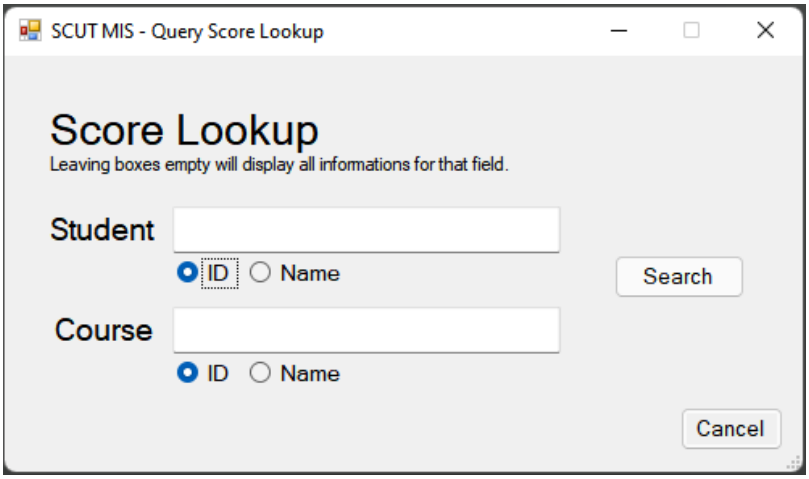
Query_ScoreLookup

To query the scores of individuals or courses, the id or name of the student and course needs to be provided. Similar to the previous dialogue boxes, the IDs are case-sensitive while the names are case-insensitive.

In the case where a box is left empty, all information of the field will be revealed. For instance, when student textbox is unfilled, all student's information will be revealed, and the same happens when the contents of the course textbox is blank.

Figure 30

Query_ScoreLookup UI Components



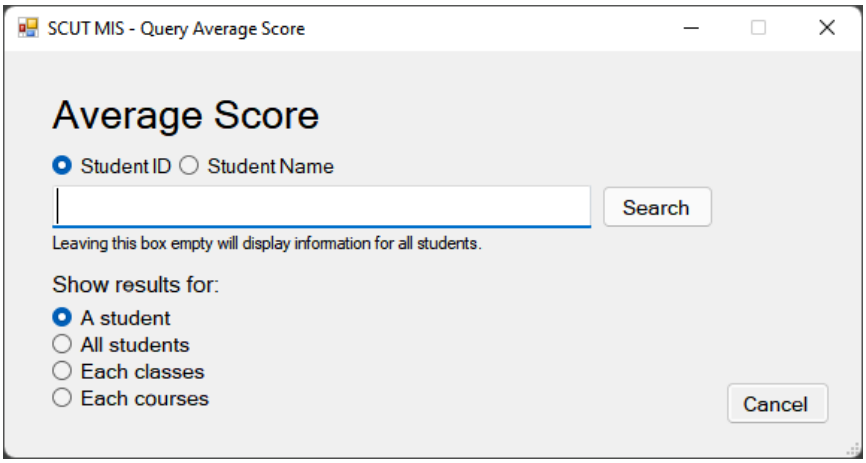
Query_AverageScore

Users can retrieve the average scores of individual students, all students, students who share the same class, or students who share the same courses. These options are determined by selecting any of the four radio buttons on the bottom of the dialogue box.

Individual student. As figure 31 indicates, when the search option is set to "A student", users can select to either input the ID or name student to be displayed in the query result. Leaving the textbox empty however, will display the average scores for all students.

Figure 31

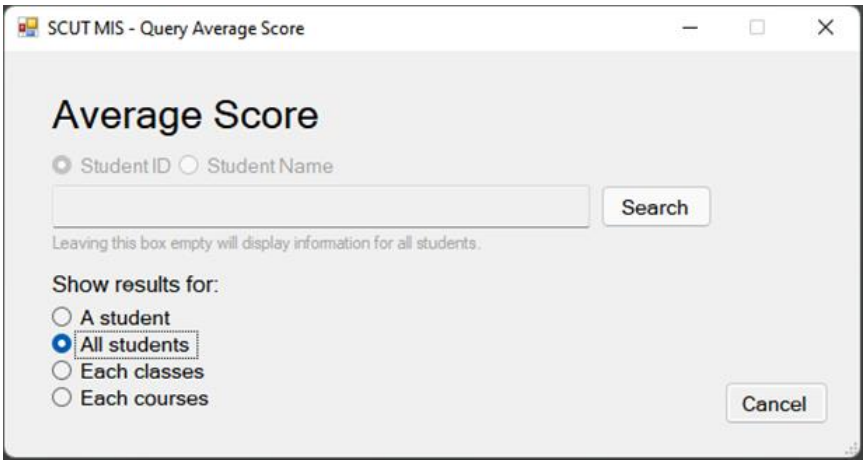
Query_AverageScore UI Components (Individual Student)



All students. Seen on Figure 32, when the “All students” option is selected, the textbox and the radio buttons are disabled and the user is only left with three options, to change the search result settings, to continue with the query by clicking the search button, or to cancel and close the dialogue box instead.

Figure 32

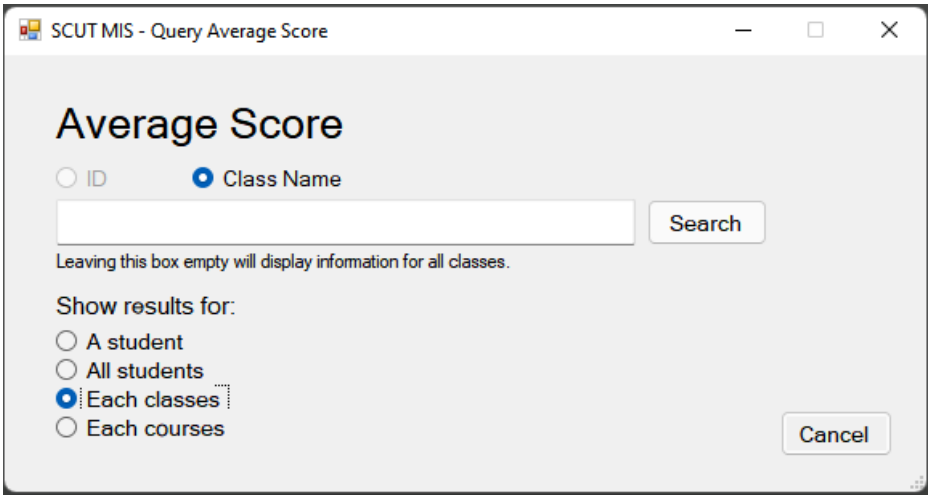
Query_AverageScore UI Components (All Students)



Students in the same class (each classes). Figure 33 shows the display change when the option “Each classes” is selected instead. As no class ID is implemented within the database tables, the “ID” radio button is disabled while the “Class Name“ is selected automatically. Moreover, the warning label under the textbox changed to inform the user of the outcome if it were to be left empty.

Figure 33

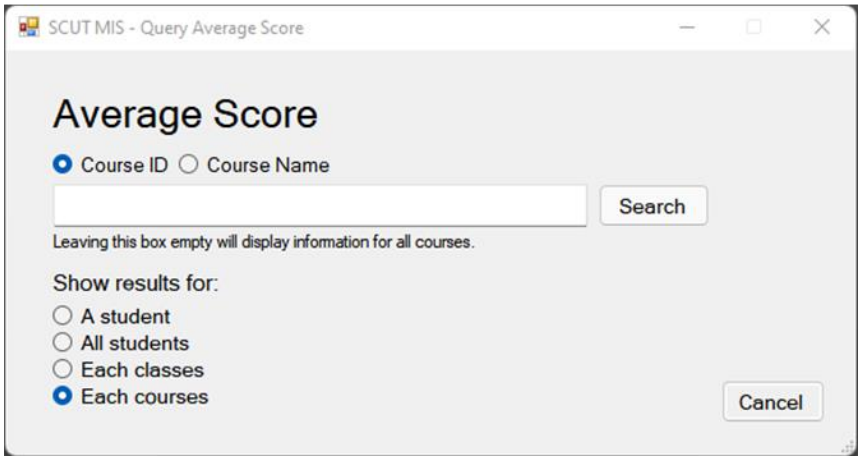
Query_AverageScore UI Components (Each Classes)



Students in the same course. As shown in figure 34, when this option is selected, the radio button's texts changed into "Course ID" and "Course Name" respectively, while the warning label informs the user that emptying the textbox will result in all courses being display instead.

Figure 34

Query_AverageScore UI Components (Each Courses)

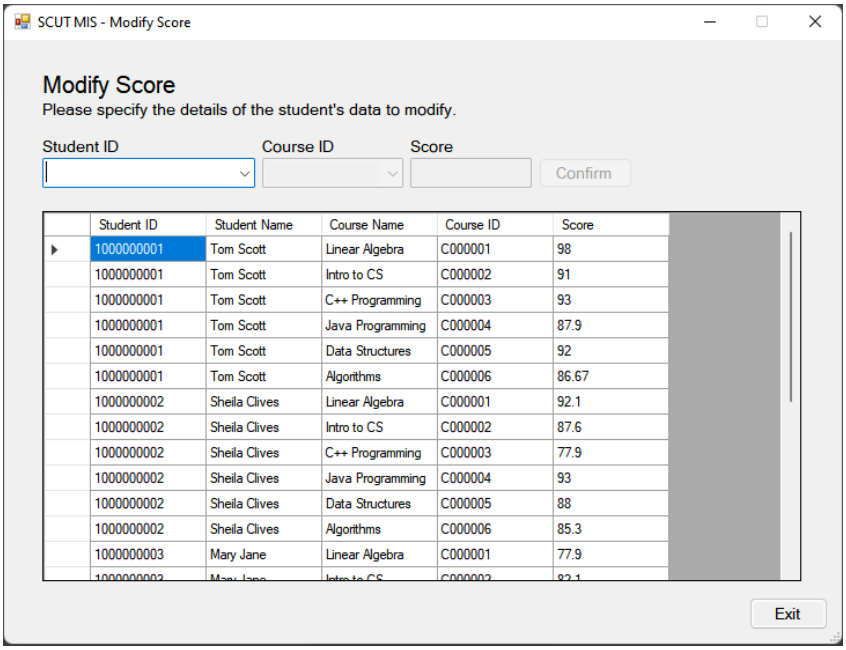


Modify_Score

The UI dialogue is restricted to be only accessible to teachers. It comprises of two dropdown combo boxes, one textbox, and one `DataGridView` to display the students' ID, their course choosing, as well as scores obtained each.

Figure 35

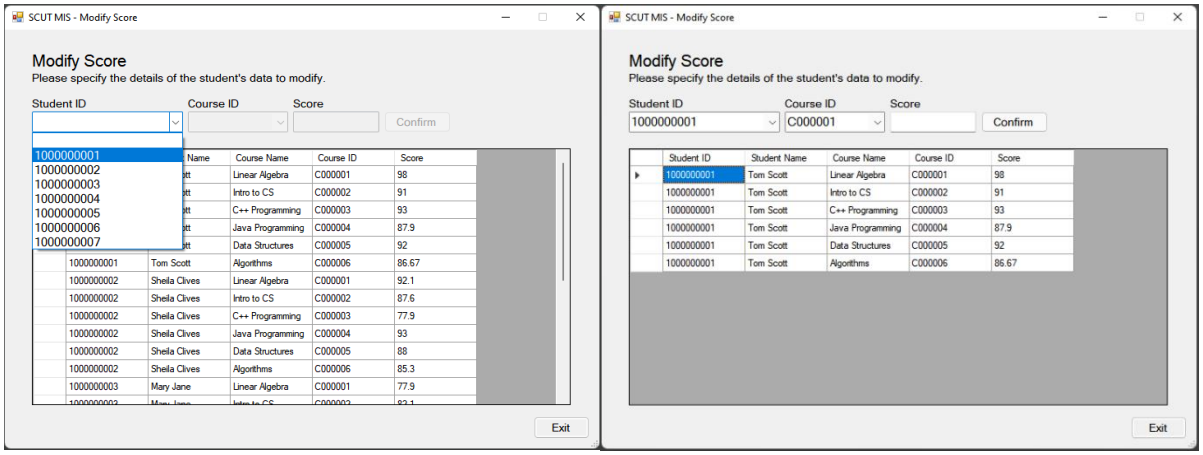
Modify_Score UI Components



As seen on figure 35, initially, only the student ID combo box is enabled on default. The remaining boxes for course ID and score, however, is only enabled after one of the available student ID is selected.

Figure 36

Modify_Score UI Components (Student ID Selection Result)

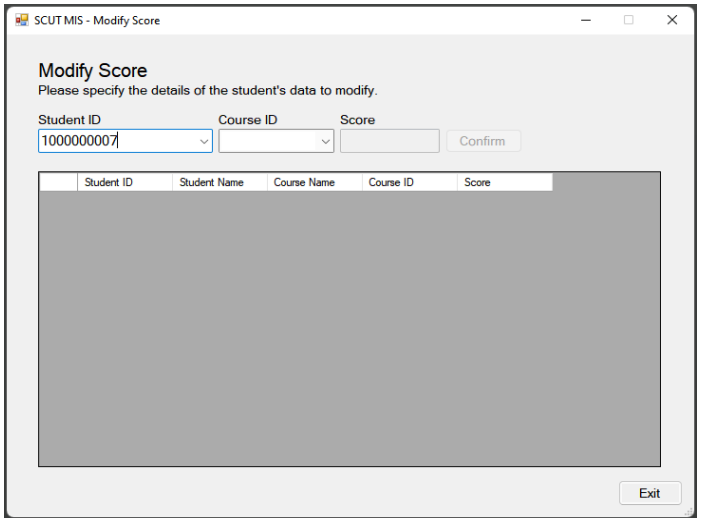


Following the selection of a student ID, as shown in figure 36, the table displayed is filtered to only display the course choosing of the student ID selected, and teachers can modify the scores of the student by selecting a course ID of their course choosing to modify, followed by clicking on the confirmation button to execute the SQL query.

Failsafe Mechanism. In the event that there exists a student without any course choosing, the course id field is emptied and score box is set to disabled while the `DataGridView` display will show no records.

Figure 37

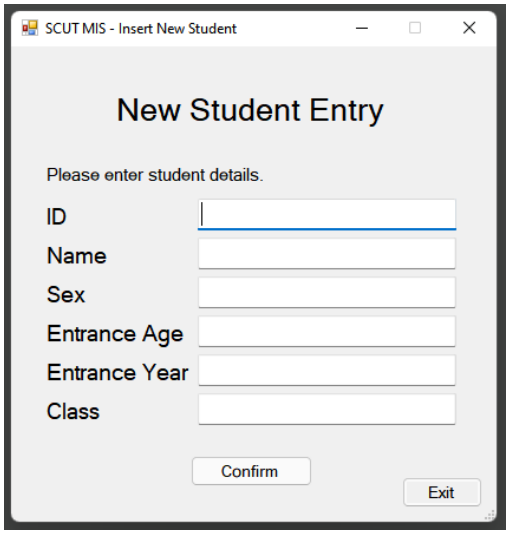
Modify_Score Failsafe Mechanism (Student with No Course Choosing)



Insert_Student

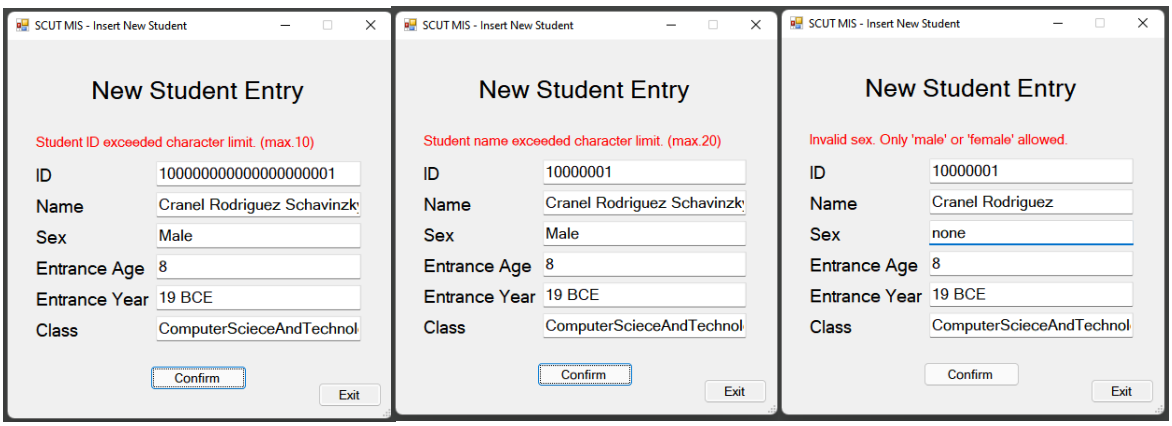
The admin-accessible UI dialogue of `Insert_Student` is comprised of 6 textboxes, where each act as a container for individual student information component, namely student ID, name, sex, entrance age, entrance year, and class respectively.

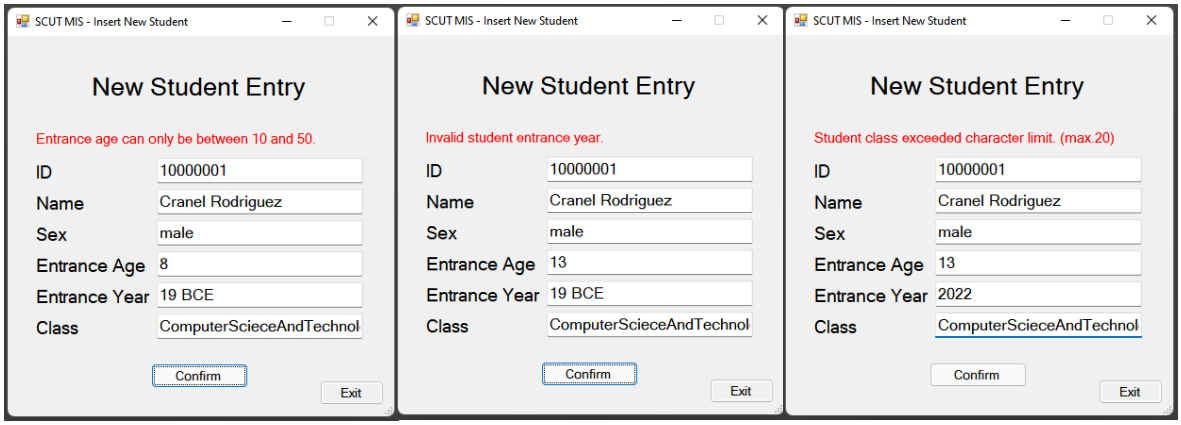
Figure 38
Insert_Student UI Components



Parameters restriction. Several restrictions are imposed to the textboxes, where student ID can be no longer than 10 characters, with the student's name consisting of no more than 20 characters. Moreover, the sex of the student can only be either "male" or "female", while their age during entrance should be between 10 to 50 years of age and the class name of the student should be no longer than 20 characters. In scenarios where any of these criteria are not met, the system will display an error message for admins to revise their inputs.

Figure 39
Insert_Student UI Error Messages



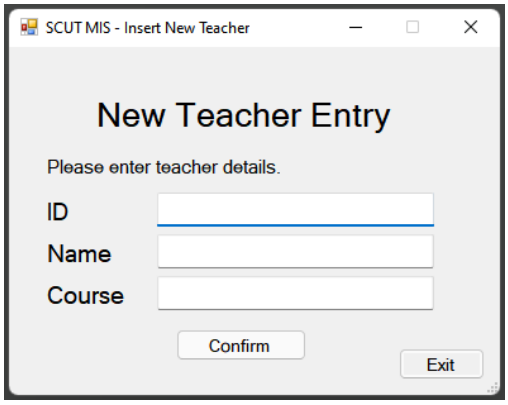


Insert_Teacher

The admin-accessible UI dialogue is comprised of three textboxes, representing teacher's ID, name, and the course they taught.

Figure 40

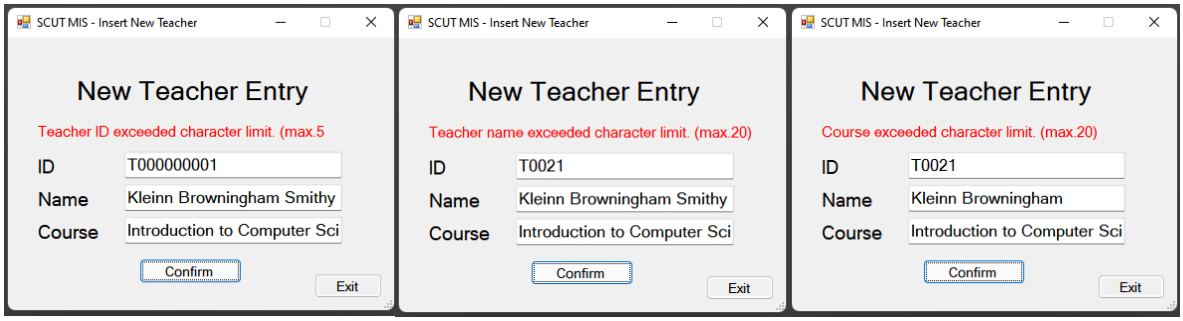
Insert_Teacher UI Components



Moreover, restrictions are set to ensure the fields are inputted appropriately, where their teachers' ID should be a string of characters of no more than 5 with name and course taught consisting of no more than 20 characters.

Figure 41

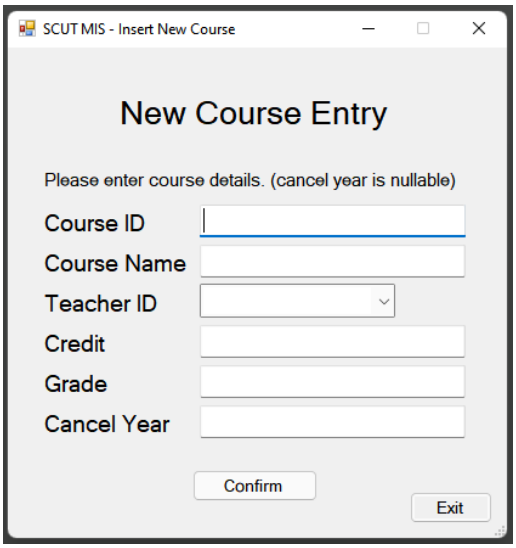
Insert_Teacher UI Error Messages



Insert_Course

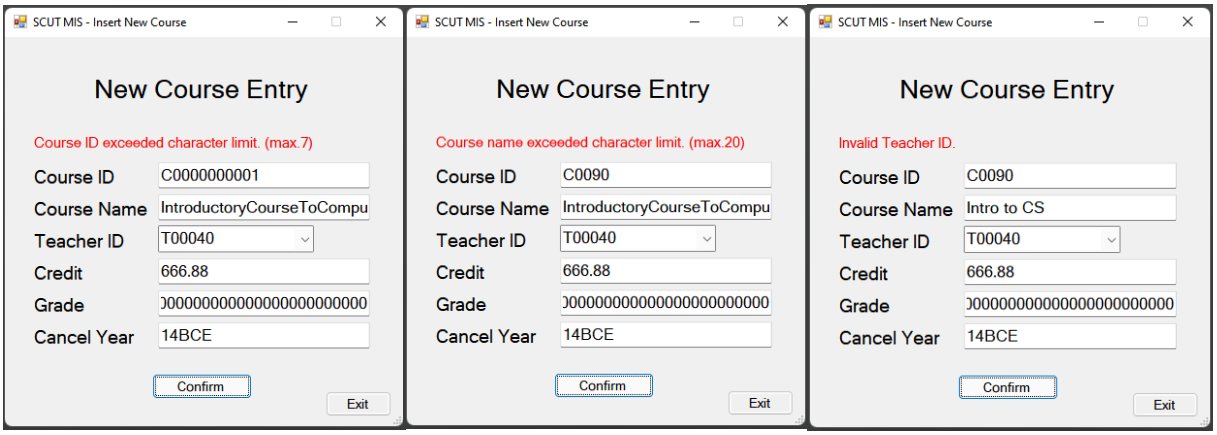
The admin-accessible UI dialogue is comprised of five textboxes and one combo box dropdown, representing course ID, name, teacher ID, course credit, course grade and the cancellation year of the course.

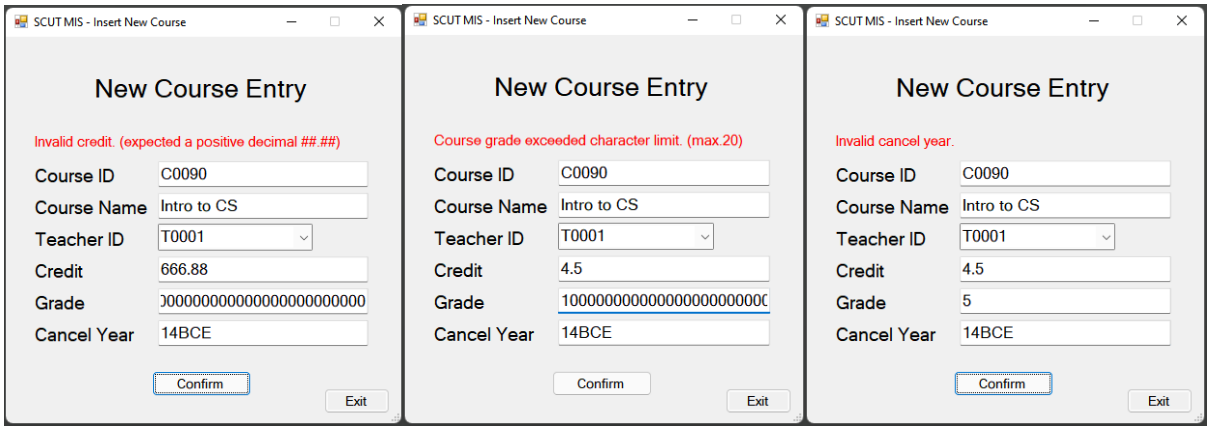
Figure 42
Insert_Course UI Components



Following the listed requirements of the projects, restrictions are imposed within the dialogue. Notably, course ID should be comprised of no longer than 7 characters, while course name and grade has the character limit of 20. The teacher in charge of the course should be listed as well, by providing the ID of an existing teacher. During inputting the data, admins need to comply these requirements. Should any data be invalid, error will be shown.

Figure 43
Insert_Course Error Messages



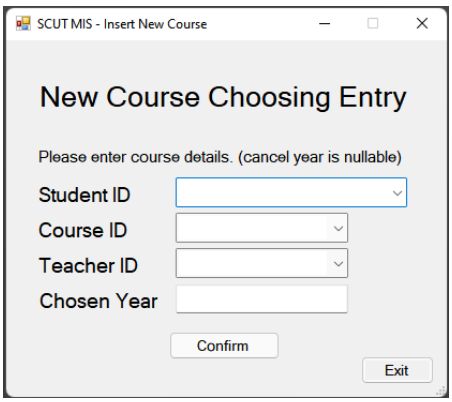


Insert_Choose

The admin-accessible UI dialogue is comprised of three combo box dropdowns, each representing ID for students, courses and teachers, with one textbox for inputting the course's chosen year.

Figure 44

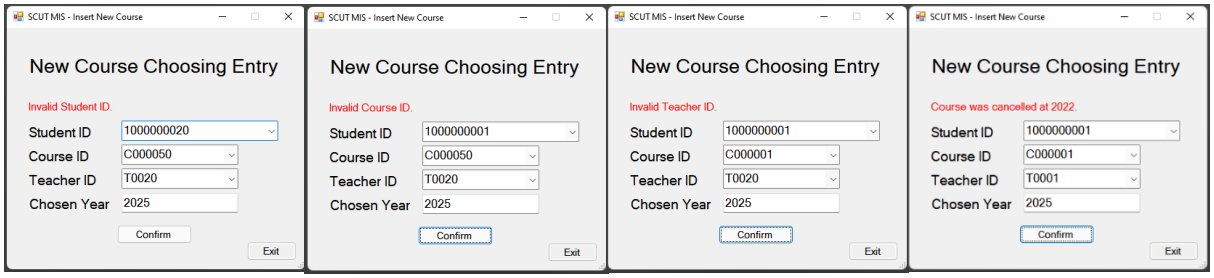
Insert_Choose UI Components



Restrictions are set for the IDs chosen or entered to only be from an existing entity from each respective field. Moreover, the chosen year need to be before the course's cancellation year and is checked prior to the course choosing's registration into the database.

Figure 45

Insert_Choose UI Error Messages



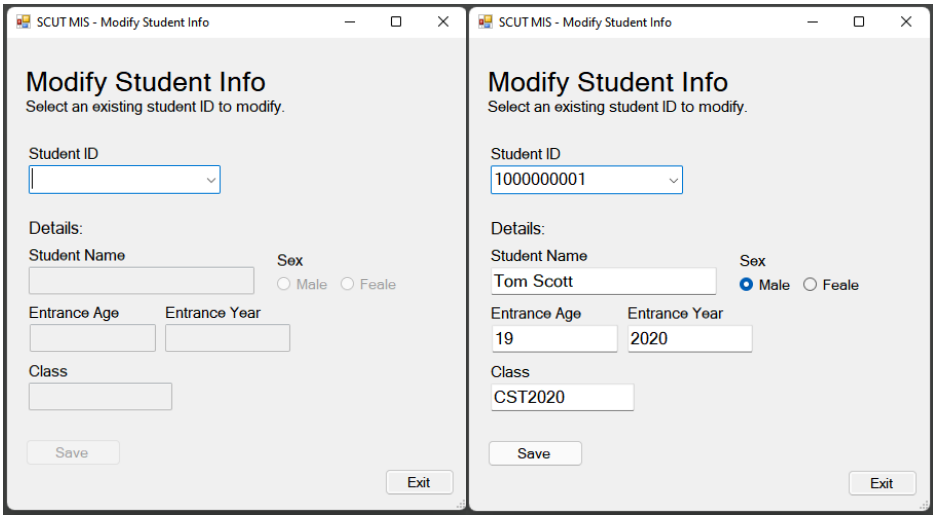
Modify_Student

The admin-accessible UI dialogue is comprised of one combo box dropdown, where admins can choose a student ID to select one student for their data to be modified. The detail fields are comprised of textboxes and a radio button, where the student's name, entrance age, entrance year, as well as class will be contained, and their sex will be chosen.

Upon the selection of a student, the system will retrieve their personal data from the database, display them and enable the textboxes for possible alterations.

The restrictions set to the dialogue is implemented to be similar to `Insert_Student`'s.

Figure 46
Modify_Student UI Components



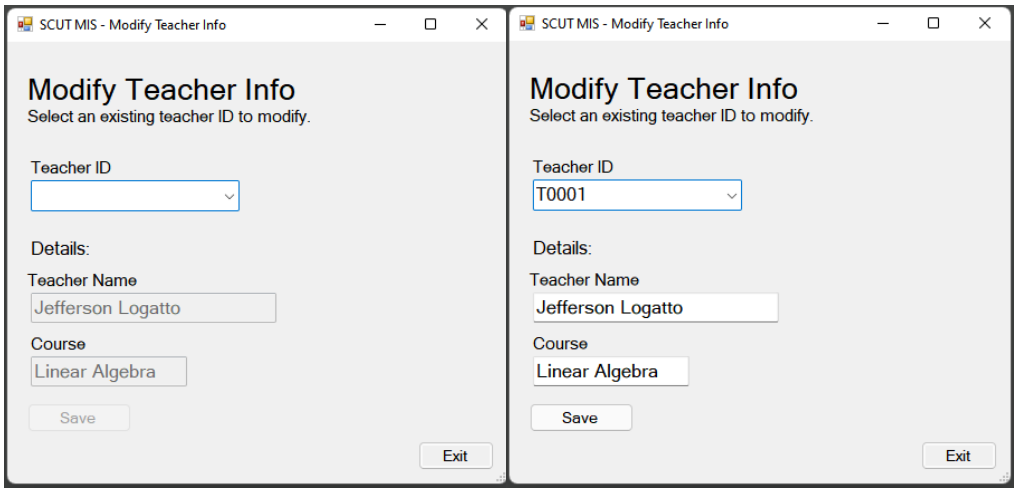
Modify_Teacher

The admin-accessible UI dialogue is comprised of one combo box dropdown, where the ID of the teachers can be selected and retrieved, while two textboxes are contained in the details section, where the teacher's name and the course they taught is listed.

Upon selecting a teacher's ID, the relevant information of the teacher will be retrieved and displayed into the editable textboxes.

The restrictions imposed within the dialogue is similarly implemented as `Insert_Teacher`'s.

Figure 47
Modify_Teacher UI Components

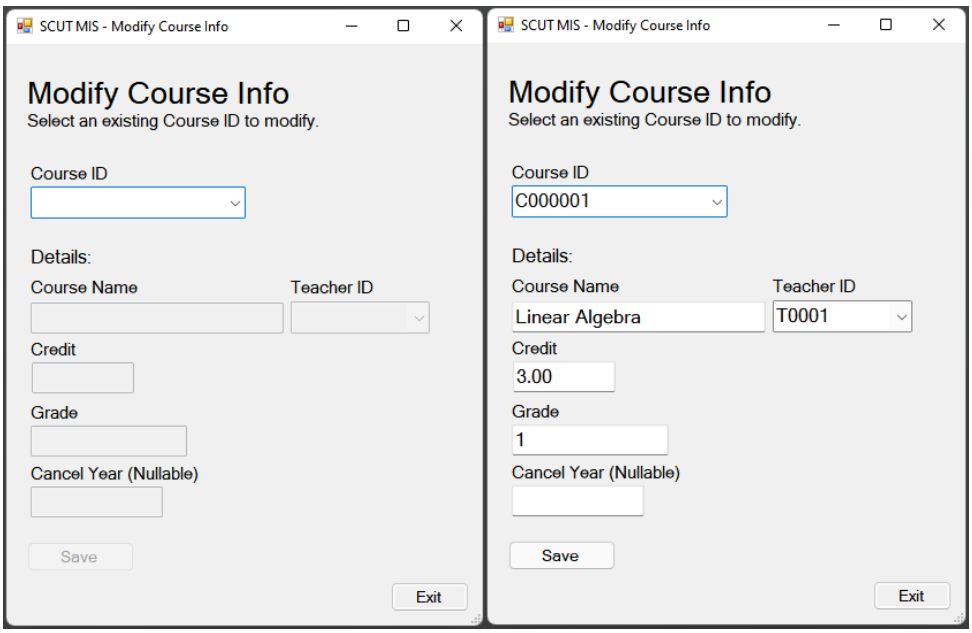


Modify_Course

The admin-accessible UI dialogue is comprised of one combo box under the title, where the ID of the course can be selected, which will retrieve the relevant information of the course and allow the details section to be alterable, where the course name, teacher's ID, course credit, grade, and the course's cancellation year is listed.

Notably, the restrictions imposed within the dialogue is similarly implemented as `Insert_Course`'s.

Figure 48
Modify_Course UI Components



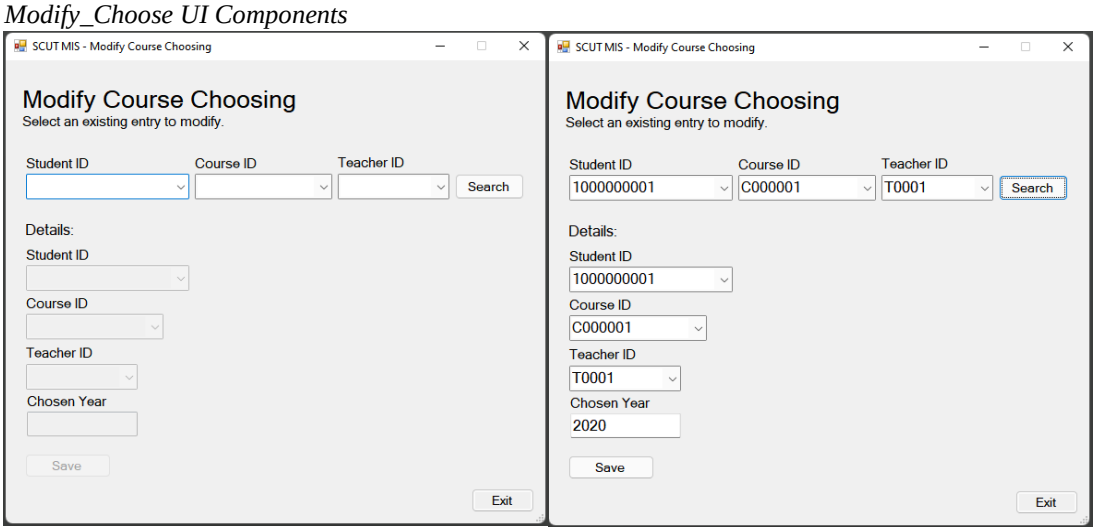
Modify_Choose

The admin-accessible UI dialogue has three combo box dropdowns under its title, where the students', courses', and teachers' ID need to be provided before any modifications take place.

Its details section contains the information of the choosing's student ID, course ID, teacher ID, and its chosen year, which upon clicking the save button, the renewed info will be stored into the database.

Figure 49

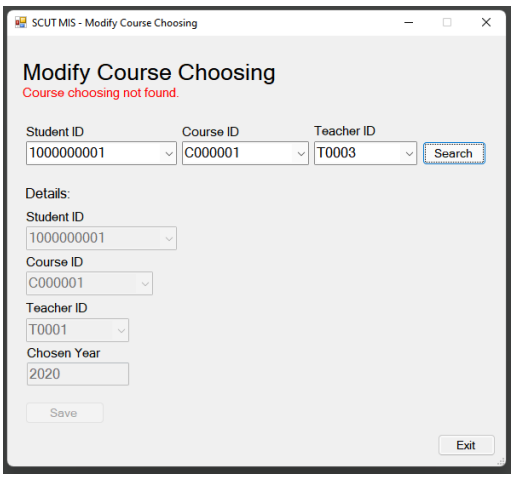
Modify_Choose UI Components



Aside from the similar restrictions to `Insert_Choose` imposed onto the dialogue, in a scenario where the student ID, course ID, and teacher ID combination does not exist, the system will throw an error and inform the user of no such entry found within the database.

Figure 50

Modify_Choose UI Error Message

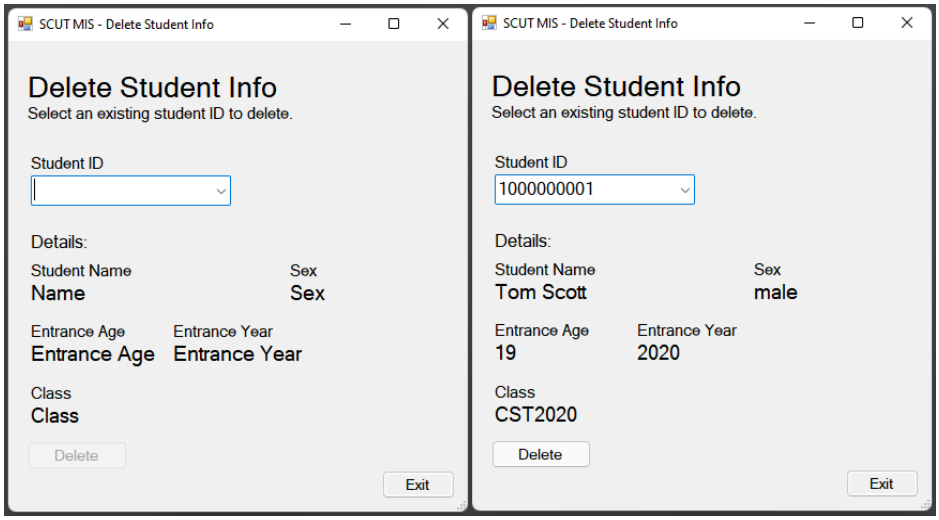


Delete_Student

The admin-accessible dialogue contains one combo box dropdown, where an ID of a student can be selected. By selecting one student, the relevant information of them will be displayed under the details section, and upon clicking the delete button, their records, along with their course choosing, will be deleted.

Figure 51

Delete_Student UI Components

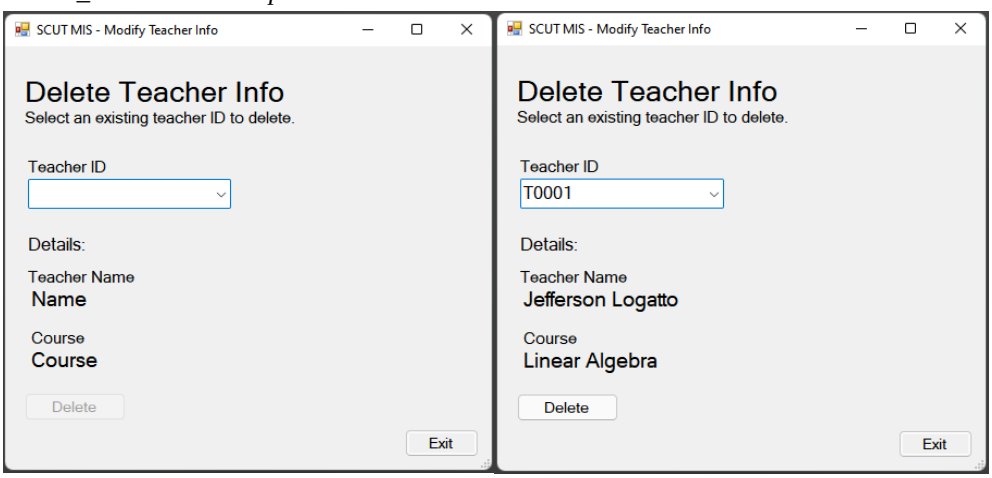


Delete_Teacher

The admin-accessible dialogue contains one combo box dropdown, where the ID of the teacher will need to be provided to correct identify the teacher entry to be deleted. Upon selection, the details of the teachers will be revealed, and the action can be completed by clicking the delete button.

Figure 52

Delete_Teacher UI Components

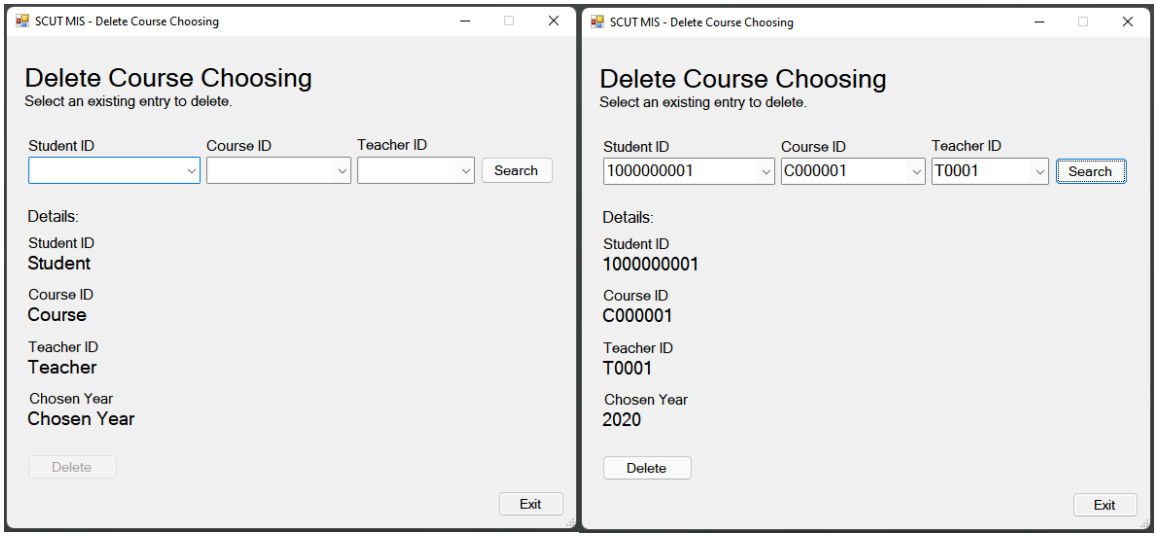


Delete_Choose

The admin-accessible dialogue box contains three combo box dropdowns to identify the course choosing entry to delete. Once valid student ID, course ID, and teacher ID are correctly identified, clicking the search button will verify the input and retrieve the corresponding information onto the display.

Figure 53

Delete_Choose UI Components



Conclusion

All in all, the design of the project has followed the general system design procedures, whilst satisfying the project requirements specified to be implemented, notably through SQL queries, managing students', teachers', courses', and course choosings' information, and to create a management information system with varying access privilege permission levels for students, teachers, and administrators, achieved with the C# programming language, Windows Forms GUI, and Microsoft SQL Server.

Covering all user input or misinput, the project is also designed with all possible scenarios in mind, by introducing failsafe mechanisms and imposing restrictions on data entry, modification, and deletion.

Ultimately, the project has been successfully in designing and implementing all requirements specified, in which is built with real-world scenario in mind.