Examination Management

This file contains the technical documentation for Examination Management module as a part of `Academic System Managament Software` by Group 10A.

Function Name: `buttonExamTT_Click`

Location

File: `AdminSetDates.h`

Input Parameters

Parameter	Description
`System::Object^ sender`	The object that raises the event
`System::EventArgs^ e`	Event data

Output Parameters

Returns `System::Void`

Description

1. Allots room for examination to each student for each course the student has taken.
2. The exam schedule is automatically fixed by slots, given a starting date. Hence, the remaining task is to provide rooms (for seating arrangement), so, this function is sufficient.
3. Allocation algorithm is based on 'first fit' idea, i.e. allocate the first available room to a student for a particular course.

> Note: We have assumed that there can be only two courses at max scheduled in the same slot. Hence the room allocation strategy first divides the room capacity into halves and uses only one of them for seating. One half is left empty to represent actual empty spaces in exams. The half which is used is further divided into halves and then allocation is done in first fit manner from this pool for a course to students.

SQL Queries

The following are the SQL queries used in this function:

Query 1: To fetch the available room capacity information from the Room Table.

```
SELECT room_ID,capacity FROM [Room] ORDER BY capacity;
```

Query 2: To fetch the information about all students and courses opted by them from the 'Courses Taken' Table.

```
SELECT * FROM [Courses Taken];
```

> Note: There is an **implicit** third query for the update of the `Courses Taken` table, which is automatically handled by the `SqlCommandBuilder^` type.

Function Name: `getIsGradesViewing`

Location

File: `StudentHome.h`

Input Parameters

None

Output Parameters

Parameter	Description
`bool`	Returns `true` if Grades viewing has started, `false` otherwise.

Description

Retrieves the boolean value of the `view_grades` field from the `Admin` table to determine if Grades viewing has started.

SQL Queries

Query 1: Get view_grades value from Admin table

```
-- Retrieving the view_grades value from the Admin table
SELECT Admin.[view_grades] FROM Admin;
```

This SQL query retrieves the boolean value of the view_grades field from the Admin table.

Function Name: `CalcSPI(DataGridView^ dataGridView)`

Location

File: `StudentHome.h`

Input Parameters

Parameter	Description
`DataGridView^ dataGridView`	DataGridView representing the courses taken along with grade that has been fetched from the Database.

Output Parameters

Parameter	Description
`double`	Returns `-1` if some grades are not released, `Semester Point Index` otherwise.

Description

1. The Semester Point Index (SPI) is calculated using the following formula:

[SPI = \frac{\text{Total Weighted Points}}{\text{Total Credits}}]

Where:

- **Total Weighted Points**: Sum of (Subject Credits * Grade Points) for all subjects taken in the semester.
- **Total Credits**: Sum of Subject Credits for all subjects taken in the semester.

Grade Mapping:

The grades are mapped to numerical points as follows:

- **AA or AS**: 10
- **AB**: 9
- **BB**: 8
- **BC**: 7
- **CC**: 6
- **CD**: 5
- **DD**: 4
- **F**: 0

> **NOTE :** If some grade has not been uploaded by the prof (i.e. it is --), this function returns -1 which results in SPI being displayed as NA.

Function Name: `getGrades`

Location

File: `ProfGradesScreen.h`

Input Parameters

Parameter	Description
`String ^ grade`	Grade in number like 10, 9.

Output Parameters

Return Grade in letters like `AA`, `AB`.

Description

The getGrades function translates letter grades into numerical equivalents. It returns corresponding numerical values for grades ranging from `AS` (10) to `F` or `--` (0), with -1 for invalid inputs.

Function Name: `setTakenCourses`

Location

File: `ProfGradesScreen.h`

Input Parameters

Parameter	Description
`String ^ prof_id`	Prof ID whose courses need to be fetched.

Output Parameters

Doesn't return anything.

Description

This function retrieves the courses associated with a given professor ID from a database. It executes a SQL query to select all courses where the professor ID matches the provided ID. Then, it populates a combo box with the IDs of the courses retrieved from the database. If an exception occurs during the database operation, it displays an error message box with the exception message.

SQL Queries

The following are the SQL queries used in this function:

Query 1: To fetch the courses taken by the Prof with ID as prof_id.

```
SELECT * FROM [Courses] WHERE prof_id = prof_id
```

Function Name: `setChart`

Location

File: `ProfGradesScreen.h`

Input Parameters

Parameter	Description
`String^ course_id`	The identifier of the course for which grade statistics are to be retrieved and displayed.

Output Parameters

The function does not return a value.

Description

1. It begins by constructing a SQL query to retrieve grade statistics for the specified course from a database. The query selects the grades and their respective counts from the "Courses Taken" table, grouped by grade.
2. Inside a try-catch block, it establishes a database connection using the connection string obtained from `Constants::getdbConnString()`.
3. It then creates a SQL command object (SqlCommand) with the constructed query and opens the database connection.
4. The function prepares the SQL command by adding a parameter (@Course_ID) with the provided course identifier to prevent SQL injection attacks.
5. It executes the SQL command to retrieve the grade statistics and iterates over the result set using a SqlDataReader.
6. While iterating, it updates the chart control (`grades_stats`) with data points representing the number of students who received each grade. It also accumulates the total number of students and calculates the total grade for further statistical analysis.
7. Additionally, it handles the case where grades are not assigned ("--") separately and keeps track of the count of such cases.
8. After processing all records, it adds a data point to the chart for the count of students with grades not assigned.
9. It updates UI elements (label7 and label5) to display the total number of students and the average grade respectively. Finally, it closes the database connection.

Exception Handling

Within the try block, exceptions are caught using a catch block, which displays an error message box containing the exception message if any error occurs during the database operations.

SQL Queries

The following are the SQL queries used in this function:

Query 1: To fetch the stats to be shown on the chart.

```
SELECT [grades],COUNT(*) AS count FROM [Courses Taken] WHERE course_ID =@Course_ID
GROUP BY [grades]
```

Function Name: `fileExists`

Location

File: `ProfGradesScreen.h`

Input Parameters

Parameter	Description
`string filename`	Representing the name of the file to be checked.

Output Parameters

The function returns a bool value.

Description

This function checks if a file exists by attempting to open it. It returns true if the file exists and can be opened successfully, and false otherwise.

Function Name: `downloadCSV`

Location

File: `ProfGradesScreen.h`

Input Parameters

Parameter	Description
`String^ course_id`	Is the course ID whose CSV file needs to be downloaded.

Output Parameters

Notifies the user with a message box upon successful download of the CSV file.

Description

1. This function generates a CSV file containing data from a database table.
2. It constructs an SQL query to retrieve data based on the provided course_id.
3. The retrieved data is formatted into a CSV file with `Roll No` and `Grade` headers.
4. It establishes a database connection and executes the query.
5. If the CSV file doesn't exist, it creates one; otherwise, it truncates the existing file.
6. Data from the database is written to the CSV file.
7. The function handles exceptions such as database connection errors or file writing errors.
8. It closes the database connection and file stream after completion.
9. Upon successful completion, it displays a message indicating the download status.

SQL Queries

The following are the SQL queries used in this function:

Query 1: To fetch the Grades of Students of a course.

```
SELECT * FROM [Courses Taken] WHERE course_ID =Course_ID
```

Function Name: `ProfGradesScreen_Load`

Location

File: `ProfGradesScreen.h`

Input Parameters

Parameter	Description
`System::Object^ sender`	The object that raises the event
`System::EventArgs^ e`	Event data

Output Parameters

The function does not return a value.

Description

1. Executes when the form associated with this function loads.
2. Invokes the `setTakenCourses` function, passing the `Faculty_ID` as an argument, likely to populate the form with relevant data.

Function Name: `button1_Click`

Location

File: `ProfGradesScreen.h`

Input Parameters

Parameter	Description
`System::Object^ sender`	The object that raises the event
`System::EventArgs^ e`	Event data

Output Parameters

The function does not return a value.

Description

1. This function is triggered when `button1` is clicked.
2. It first checks if an item is selected in `comboBox1`.
3. If an item is selected, it calls the setChart function with the selected item as an argument.
4. If no item is selected, it displays a message box instructing the user to select a course from the list.

Function Name: `splitString`

Location

File: `ProfGradesScreen.h`

Input Parameters

Parameter	Description
`const std::string& s`	This is the string need to be splited in substrings.
`char delimiter`	This is the delimiter which separates the substrings in the given string.

Output Parameters

Returns the splited data.

Description

1. Splits the input string s into substrings based on the specified delimiter.
2. It creates a std::istringstream from the input string s.
3. Reads each substring delimited by delimiter using std::getline.
4. Pushes each substring into a std::vectorstd::string.
5. Returns the vector containing the split substrings.

Function Name: `readCSV`

Location

File: `ProfGradesScreen.h`

Input Parameters

Parameter	Description
`const std::string& filename`	The name of the CSV file needs to be read.

Output Parameters

Returns a 2D vector of CSV data read from the given file.

Description

1. Opens the specified CSV file named filename.
2. Reads the contents of the file line by line.
3. For each line, it splits the line into tokens using the splitString function (presumably defined elsewhere).
4. Adds the tokens as a row to a 2D vector (data), representing the CSV data.
5. Returns the 2D vector containing the CSV data.

Function Name: `updateGrades`

Location

File: `ProfGradesScreen.h`

Input Parameters

Parameter	Description
`std::vector<std::vector<std::string>> csvData`	Contain the data for CSV needs to be uploaded.
`String^ course_ID`	Course ID of the course.

Output Parameters

Returns noting.

Description

1. Constructs an SQL query to update grades in the database table [Courses Taken].
2. Iterates through each row of the csvData.
3. Constructs and executes an SQL command for each row to update the corresponding grade in the database.
4. Uses the `Constants::getdbConnString()` method to obtain the database connection string.
5. Utilizes `Constants::strCnvStr()` to convert C++ string data to .NET String^.
6. Catches and displays any exceptions that occur during database operations using a message box.

SQL Queries

The following are the SQL queries used in this function:

Query 1: To Upload grades of each students.

```
UPDATE [Courses Taken] SET grades=@Grades WHERE roll_no=@Roll_No AND
course_ID='"+course_ID+"'
```

Function Name: `checkCSVContent`

Location

File: `ProfGradesScreen.h`

Input Parameters

Parameter	Description
`std::vector<std::vector<std::string>> csvData`	Data of CSV file which is to be checked.

Output Parameters

Returns a bool weather the content and format of CSV data is correct or not.

Description

1. Checks the content of the provided CSV data to ensure its validity.
2. Verifies that the CSV file is not empty and contains the expected number of columns (two columns: `Roll No` and `Grade`).
3. Ensures that the column names are correct (`Roll No` and `Grade`).
4. Validates the format and data integrity of each row:
5. Checks if the grade value is valid by calling the getGrades function with the grade value.
6. Verifies that the `Roll No` field contains exactly 9 characters and consists only of numerical digits.
7. Displays appropriate error messages using message boxes if any issues are detected.

Function Name: `isGradesSubmission`

Location

File: `ProfGradesScreen.h`

Input Parameters

No input is given.

Output Parameters

Returns bool weather grades are submitted successfully or not.

Description

1. Provides a mechanism to determine whether the system currently allows grade submissions.
2. Helps control access to the grade submission functionality based on the system configuration.
3. Queries the database to check whether the system allows grade submissions.
4. Retrieves the value of the `is_grade_submission` field from the [Admin] table.
5. Returns true if grade submission is enabled (`is_grade_submission` equals true), otherwise returns false.
6. Catches any exceptions that may occur during database access and displays an error message using a message box.

SQL Queries

The following are the SQL queries used in this function:

Query 1: To fetch data about admin.

```
SELECT * FROM [Admin]
```

Function Name: `button3_Click`

Location

File: `ProfGradesScreen.h`

Input Parameters

Parameter	Description
`System::Object^ sender`	The object that raises the event
`System::EventArgs^ e`	Event data

Output Parameters

Returns noting.

Description

1. Handles the click event of button3.
2. Checks if a course is selected from the comboBox1.
3. Calls the isGradesSubmission function to determine if grade submission is allowed.
4. If grade submission is allowed:
5. Opens a file dialog (`OpenFileDialog`) to select a CSV file containing grade data.
6. Reads the selected CSV file using the `readCSV` function.
7. Validates the content of the CSV file using the `checkCSVContent` function.
8. If the CSV file content is valid:
9. Updates the grades in the database using the updateGrades function.
10. Displays a message box indicating successful upload of the grades CSV.
11. If an exception occurs during file reading, displays an error message using a message box.
12. If grade submission is not allowed, displays a message box indicating that course grades submission is not permitted.
13. If no course is selected from the list, displays a message box prompting the user to select a course.

Function Name: `button2_Click`

Location

File: `ProfGradesScreen.h`

Input Parameters

Parameter	Description
`System::Object^ sender`	The object that raises the event
`System::EventArgs^ e`	Event data

Output Parameters

Returns noting.

Description

1. Handles the click event of `button2`.
2. Checks if a course is selected from the `comboBox1`.
3. If a course is selected:
4. Calls the `downloadCSV` function to generate and download a CSV file containing grade data for the selected course.
5. If no course is selected from the list, displays a message box prompting the user to select a course.