Course Management

This file is the technical Documentation for Course Management Module as a part of `Academic System Management Software` by Team 10A.

Function Name: `setAvailableCoursesTable`

Location

File: `StudentCourseReg.h`

Input Parameters

Parameter	Description
`String ^ semester`	Semester for which courses have to be fetched

Output Parameters

Doesn't return anything.

Description

1. Gets all the compulsory courses offered for that semester and populates the `Data Grid View` that displays the courses with Add/Drop button.
2. Adds all the slots of compulsory courses to busy slots for that student in `chosenElectiveSlots` set.

SQL Queries

The following are the SQL queries used in this function:

Query 1: To fetch compulsory courses details from Faculty and Courses Table.

```
SELECT Courses.[course_ID],Courses.[course_name],Faculty.[name]
as prof_name,Courses.[LTPC],Courses.slot
FROM Courses
INNER JOIN Faculty ON
Faculty.faculty_ID = Courses.[prof_ID]
WHERE Courses.[sem_offered] = @Semester
 AND Courses.[is_compulsory] = 1
;
```

Function Name: `DataGridView1_CellContentClick`

Location

File: `StudentCourseReg.h`

Input Parameters

Parameter	Description
`System::Object^ sender`	The object that raises the event.
`System::Windows::Forms::DataGridViewCellEventArgs^ e`	Information about the event data (DataGridViewCellEventArgs).

Output Parameters

Doesn't return anything.

Description

1. Monitors the click event on the "Add/Drop" button in a DataGridView.
2. If the button is clicked:
    - Modifies the Database by adding or dropping a course using `InsertIntoDB` or `DeleteFromDB`.
    - Toggles the button text between "Add" and "Drop".

Function Name: `DeleteFromDB`

Location

File: `StudentCourseReg.h`

Input Parameters

Parameter	Description
`int RollNo`	Roll number of the student.
`String^ cid`	Course ID to be deleted from the student's records.

Output Parameters

Doesn't return anything.

Description

Deletes a record from the "Courses Taken" table in the database for a specific student and course.

SQL Queries

The following SQL query is used in this function:

Query 1: Delete record from "Courses Taken" table

```
DELETE FROM [Courses Taken]
WHERE roll_no = @RollNo
AND Course_ID = @CID
```

This query will delete the course taken by student when he/she presses Drop button.

Function Name: `CommonButtonClickHandler`

Location

File: `StudentCourseReg.h`

Input Parameters

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

Output Parameters

Doesn't return anything.

Description

Handles the click event for common buttons in the UI. Performs actions based on the button clicked, such as adding or dropping courses from the database. Updates UI elements accordingly.

Function Name: `handleThirdyearElectives`

Location

File: `StudentCourseReg.h`

Input Parameters

None.

Output Parameters

Doesn't return anything.

Description

1. Populates the UI ComboBoxes with elective courses for the third year.
2. Separates lab and theory electives into different lists based on the database query results.

SQL Queries

Query 1: Get third-year elective lab and theory courses info from Courses and Faculty Tables

```
SELECT Courses.[course_ID], Courses.[course_name], Faculty.[name]
as prof_name, Courses.[LTPC], Courses.slot, Courses.[is_lab]
FROM Courses
INNER JOIN Faculty
ON Faculty.faculty_ID = Courses.[prof_ID]
WHERE Courses.[sem_offered]
LIKE '%6%' AND Courses.[is_compulsory] = 0
```

This SQL query retrieves elective lab and theory courses offered in the sixth semester.

Function Name: `getColValue`

Location

File: `StudentCourseReg.h`

Input Parameters

Parameter	Description
`String^ colReqd`	The column whose value is required.
`String^ colOrig`	The column to match the specified value against.
`String^ colVal`	The specified value to match against.

Output Parameters

Parameter	Description
`String^`	The value of the specified column based on the condition.

Description

Retrieves the value of a specified column (`colReqd`) from the "Courses" table where the value of another column (`colOrig`) matches the provided value (`colVal`).

SQL Queries

Query 1: Get column value from "Courses" table

```
SELECT Courses.[colReqd] FROM Courses
WHERE Courses.[colOrig] = 'colVal'
```

This SQL query retrieves the specified column value based on the condition.

Function Name: `checkCompatibility`

Location

File: `StudentCourseReg.h`

Input Parameters

Parameter	Description
`String^ course`	The course to be checked for compatibility.
`String^ Slot`	The slot to check compatibility with the course.

Output Parameters

Parameter	Description
`bool`	Returns `true` if the course is compatible, `false` otherwise.

Description

1. Checks if a given course is compatible to be chosen in a specified slot.
2. It considers conflicts based on the type of lab and other chosen elective slots, which are maintained in a map called `chosenElectiveSlots`.
3. Uses map and common time table slot logic to output the boolean value.

> **NOTE :** This implementation considers that A course cant be taken if any of its slot clashes with a Lab and also assumes clash with compulsory courses mandatorily irrespective of whether the student has registered for that or not.

Function Name: `comboBoxY_SelectedIndexChanged`

Location

File: `StudentCourseReg.h`

Input Parameters

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

Output Parameters

Doesn't return anything.

Description

1. Handles the selected index change event for `comboBoxY`.

2. Displays information about the selected course, including the course name, professor ID, and slot in the UI appropriately.

SQL Queries

Query 1: Get course name based on selected course ID

```
SELECT Courses.[course_name] FROM Courses WHERE Courses.[course_ID] =
'selectedItem'
```

This SQL query retrieves the course name based on the selected course ID.

Query 2: Get professor ID and slot based on selected course ID

```
SELECT Courses.[prof_ID], Courses.[slot] FROM Courses WHERE Courses.[course_ID] =
'selectedItem'
```

This SQL query retrieves the professor ID and slot based on the selected course ID.

Function Name: `StudentCourseReg_Load`

Location

File: `StudentCourseReg.h`

Input Parameters

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

Output Parameters

Doesn't return anything.

Description

1. Handles the load event for the `StudentCourseReg` form.
2. Clears the database for the current student, sets up available courses table, and handles elective courses based on the semester.
3. For Second and first yearites no electives are shown and for third yearites electives are shown. The fourth year electives are handled by `FourthYearCourseReg.h`.

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: `StudentHome_Load`

Location

File: `StudentHome.h`

Input Parameters

- `System::Object^ sender`: The object that raised the event.
- `System::EventArgs^ e`: The event data.

Output Parameters

None

Description

1. Displays the user's DP using a method from the `Constants` class.
2. Initializes UI elements, sets labels, and disables buttons.
3. Checks if grades can be viewed using a method named `getisGradesViewing`, displayes the SPI also.
4. Adjusts UI elements based on grades viewing status.

5. Retrieves and displays course details in DataGridView1.
6. Retrieves and displays fee payment status in DataGridView2.
7. Checks and updates buttons for course registration and fee payment.

Function Name: `FetchAndDisplayImage`

Location

File: `Constants.cpp`

Input Parameters

- `String^ rno`: Roll number of the student.
- `PictureBox^ pictureBox`: PictureBox control to display the student's image.

Output Parameters

None

Description

1. Fetches the student's profile picture from the database based on the roll number and displays it in the specified PictureBox, the images are scraped from `IITG CSE` website.
2. Converts the image data to a `MemoryStream` and creates a `Bitmap` from it.
3. A similar function is `fetchAndDisplayProfImage` which displays the image of professor.

SQL Queries

The following is the SQL query used in this function:

Query 1: To fetch Student's profile picture from "Student Database" Table.

```
SELECT DP FROM [Student Database] WHERE roll_number = @rno
```

Fetches the `VARBINARY` encoded image from DB, which is then post processed in Visual C++.

Function Name: `StudentTimetable_Load`

Location

File: `StudentTimetable.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. Renders the dropdown for selecting day and the timetable for Monday by default on load of the screen.

SQL Queries

The following are the SQL queries used in this function:

Query 1: To fetch the room id to name and location mapping information from the Room Table.

```
SELECT room_ID,name,location FROM [Room];
```

Query 2: To fetch the information about the current users' (student) courses from the 'Courses Taken' Table.

```
SELECT course_ID,course_name,slot,room_ID
FROM [Courses]
WHERE course_ID IN
(
    SELECT course_ID
    FROM [Courses Taken]
    WHERE [Courses Taken].roll_no = roll_no
);
```

Function Name: `comboBox1_SelectedIndexChanged`

Location

File: `StudentTimetable.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. Updates the screen according to selected day from the dropdown, displaying the schedule for the selected day.

SQL Queries

The following are the SQL queries used in this function:

Query 1: To fetch the room id to name and location mapping information from the Room Table.

```
SELECT room_ID,name,location FROM [Room];
```

Query 2: To fetch the information about the current users' (student) courses from the 'Courses Taken' Table.

```
SELECT course_ID,course_name,slot,room_ID
FROM [Courses]
WHERE course_ID IN
(
    SELECT course_ID
    FROM [Courses Taken]
    WHERE [Courses Taken].roll_no = roll_no
);
```

Function Name: `chronoSort`

Location

File: `StudentTimetable.h`

Input Parameters

Parameter	Description
`tuple<string, string, string, string>& a`	A reference to a tuple representing a row of the timetable
`tuple<string, string, string, string>& b`	A reference to a tuple representing a row of the timetable

Output Parameters

Parameter	Description
`bool`	`true` if `a` should be placed before `b` in a chronological order, otherwise `false`

Description

1. Custom sorts the rows of the time table in chronologically ascending order, as the default sorting is lexicographic.

> Note: The function is declared `static` to avoid conflicts between managed (garbage collected) versions and native versions of functions, data types and data structures provided by C++.

Function Name: `ProfTimetable_Load`

Location

File: `ProfTimetable.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. Renders the dropdown for selecting day and the timetable for Monday by default on load of the screen.

SQL Queries

The following are the SQL queries used in this function:

Query 1: To fetch the room id to name and location mapping information from the Room Table.

```
SELECT room_ID,name,location FROM [Room];
```

Query 2: To fetch the information about the current users' (faculty) courses from the Courses Table.

```
SELECT slot,course_ID,course_name,room_ID
FROM [Courses]
WHERE prof_ID = faculty_ID;
```

Function Name: `comboBox1_SelectedIndexChanged`

Location

File: `ProfTimetable.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. Updates the screen according to selected day from the dropdown, displaying the schedule for the selected day.

SQL Queries

The following are the SQL queries used in this function:

Query 1: To fetch the room id to name and location mapping information from the Room Table.

```
SELECT room_ID,name,location FROM [Room];
```

Query 2: To fetch the information about the current users' (faculty) courses from the Courses Table.

```
SELECT slot,course_ID,course_name,room_ID
FROM [Courses]
WHERE prof_ID = faculty_ID;
```

Function Name: `chronoSort`

Location

File: `ProfTimetable.h`

Input Parameters

Parameter	Description

Parameter	Description
`tuple<string, string, string, string>& a`	A reference to a tuple representing a row of the timetable
`tuple<string, string, string, string>& b`	A reference to a tuple representing a row of the timetable

Output Parameters

Parameter	Description
`bool`	`true` if `a` should be placed before `b` in a chronological order, otherwise `false`

Description

1. Custom sorts the rows of the time table in chronologically ascending order, as the default sorting is lexicographic.

> Note: The function is declared `static` to avoid conflicts between managed (garbage collected) versions and native versions of functions, data types and data structures provided by C++.

> Note: In both `ProfTimetable.h` and `StudentTimetable.h` there are two additional functions which operate on a `DataGridView`, which are `ClearDataGridView` and `RemoveAutoSorting`, which, as their names suggest, clears the `DataGridView` and removes the automatic (default) sorting (here, lexicographic) on columns of `DataGridView`, respectively.

> Note: Many functions as well as data and its structure in both `ProfTimetable.h` and `StudentTimetable.h` are common, raising the question that they could be abstracted out of these two files and reused everywhere as common functions or common data, however, the issues between the managed runtime and native C++ prevented that.

Function Name: `buttonTT_Click`

Location

File: `AdminSetdates.h`

Input Parameters

Parameter	Description
`sender`	Object, the control that raised the event.
`e`	EventArgs, the event data.

Output Parameters

None

Description

1. This function is triggered when the admin clicks on a button, initiating the generation of a timetable.
2. It's a heuristic-based timetable generator aiming to minimize conflicts between compulsory and elective courses. The algorithm ensures that no two compulsory courses for the same batch or two courses for the same professor are assigned the same slot.
3. The heuristic in play here is that compulsory courses are alloted starting from A to then G slot. However electives are alloted starting from G to then A slot. Hence minimising the conflicts. It also considers different scheduling patterns for morning and afternoon classes for different years.
4. Slots are alloted room each wise i.e. each batch will have a different room for the same slot thus ensuring no clash between them.

SQL Queries

The following are the SQL queries used in this function:

Query 1: Getting Course Data

```
SELECT * FROM Courses;
```

Query 2: Getting Admin Data

```
SELECT * FROM Admin;
```

Query 3: Updating Course Data

```
UPDATE Courses SET slot = @slot, room_ID=@room_ID WHERE course_ID = @CourseCode;
```

Query 4: Updating Admin Data

```
UPDATE Admin SET view_timetable = 'True';
```

> Note: This function doesn't ensure always optimal solution and expects that admin has already mapped professors to their courses. Here the problem reduces to alloting each course a slot as we assume timetable follows the same pattern as existing timetable This function ensures that no 2 compulsory courses for same batch are given same slot This function ensures that no 2 courses for same proffessor are given same slot The 1st and 3rd years have classes in morning and labs in afternoon whereas 2nd and 4th year have classes in afternoon and labs in morning.

Function Name: `MarshalString`

Location

File: `AdminSetdates.h`

Input Parameters

Parameter	Description
`s`	Managed String to be marshaled
`os`	Reference to a C++ string to store the marshaled result

Output Parameters

Output	Description
`os`	Reference to a C++ string containing the marshaled result

Description

1. This function takes a managed string (`s`) and a reference to a C++ string (`os`) as input parameters.
2. It marshals the managed string `s` into a C-style string (`chars`) using `Marshal::StringToHGlobalAnsi`.
3. The C-style string is assigned to the C++ string `os`.
4. Finally, memory allocated by `Marshal::StringToHGlobalAnsi` is released using `Marshal::FreeHGlobal`.

SQL Queries

The function does not involve any SQL queries.