Admissions and Registration

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

Function Name: `ValidateLogin`

Location

File: `loginHandle.h`

Input Parameters

Parameter	Description
`String^ ID`	User ID for login
`String^ password`	User password for login
`String^ role`	User role (e.g., “Admin,” “Professor,” “Student”)

Output Parameters

Returns a boolean value indicating whether the login is valid or not.

Description

1. Validates user login credentials based on the provided ID, password, and role.
2. Connects to the database and performs role-specific queries to check the credentials.

SQL Queries

The following SQL queries are used in this function:

Query 1: For Admin Role

```
SELECT COUNT(*) FROM dbo.Admin WHERE admin_ID = @adminID AND
        password = @password
```

Query 2: Professor Login Validation

```
SELECT COUNT(*) FROM dbo.Faculty WHERE faculty_ID = @facultyID
        AND password = @password
```

Query 3: Professor Login Validation

```
SELECT COUNT(*) FROM dbo.[Student Database] WHERE email_id =
        @studentID AND password = @password
```

Function Name: GetUserName

Location

File: `loginHandle.h`

Input Parameters

Parameter	Description
String^ ID	User ID for which the name is to be retrieved
String^ role	User role (e.g., “Professor,” “Student”)

Output Parameters

Returns the user name based on the specified role and ID.

Description

1. Retrieves the user name from the database based on the provided ID and role.
2. Connects to the database and performs role-specific queries to fetch the user name.
3. Role-specific queries include fetching the name from the Faculty table for professors and the [Student Database] table for students.

SQL Queries

Query 1: For Professor Role

```
SELECT name FROM dbo.Faculty WHERE faculty_ID = @facultyID
```

Query 2: For Student Role

```
SELECT name FROM dbo.[Student Database] WHERE email_id =
        @studentID
```

Function Name: GetRoll

Location

File: `loginHandle.h`

Input Parameters

Parameter	Description
String^ ID	User ID for which the roll number is to be retrieved (typically a student ID)

Output Parameters

Returns the user roll number.

Description

1. Retrieves the user roll number from the database based on the provided student ID.
2. Connects to the database and executes the following SQL query to fetch the roll number from the [Student Database] table:

```
SELECT roll_no FROM dbo.[Student Database] WHERE email_id =
        @studentID
```

Function Name: GetUserYear

Location

File: `loginHandle.h`

Input Parameters

Parameter	Description
`String^ ID`	User ID for which the academic year is to be retrieved
`String^ role`	User role (e.g., “Student”)

Output Parameters

Returns the user’s academic year.

Description

1. Retrieves the user’s academic year from the database based on the provided student ID and role.
2. Connects to the database and performs a role-specific query to fetch the academic year from the [Student Database] table.
3. Uses the provided email ID to uniquely identify the student in the query.
4. Handles exceptions and displays an error message if there are issues with the database connection or query execution.

SQL Query

Query: For Student Role

```
SELECT year FROM dbo.[Student Database] WHERE email_id =
        @studentID
```

Function Name: `ContainsNonNumericCharacters`

Location

File: `loginHandle.h`

Input Parameters

- **`String^ str`**: The string to be checked for non-numeric characters.

Output Parameters

Returns a boolean indicating whether the string contains non-numeric characters.

Description

1. Iterates through each character in the input string.
2. Checks if the character is not a digit using `Char::IsDigit`.
3. Returns `true` if a non-numeric character is found; otherwise, returns `false`.

Function Name: `FetchDetailsByRollNumber`

Location

File: `updateForm.h`

Input Parameters

- **`String^ rollNumber`**: Roll number for which details are to be fetched.
- **`String^ Role`**: Role (e.g., “Student” or “Professor”).

Output Parameters

Returns a `std::map<string, string>` containing details fetched based on the roll number and role.

Description

1. Initializes a `std::map` to store details.
2. Checks the provided role (“Student” or “Professor”).
3. Executes a role-specific SQL query to fetch details from the corresponding table based on the identifier (roll number or faculty ID).
4. Retrieves column names and values from the SQL result set, inserting them into the `std::map`. ### SQL Queries

Query: For Student Role

```
SELECT * FROM [Student Database] WHERE roll_no = @RollNumber
```

Query: For Professor Role

```
SELECT * FROM [Faculty] WHERE faculty_ID = @faculty_ID
```

Function Name: `ImageToBytes`

Location

File: `updateForm.h`

Input Parameters

- **`Bitmap^ image`**: Bitmap image to be converted to a byte array.

Output Parameters

Returns a `array<Byte>^` representing the byte array of the provided Bitmap image.

Description

1. Creates a `MemoryStream` to store the image data.
2. Saves the provided Bitmap image to the MemoryStream in JPEG format.
3. Converts the MemoryStream to a byte array using `ToArray` method.
4. Returns the resulting byte array.

Function Name: `updateUserDetails`

Location

File: `updateForm.h`

Input Parameters

- **`String^ rollnumber`**: Roll number or user ID for which details are to be updated.
- **`String^ address`**: New address value.
- **`String^ password`**: New password value.
- **`String^ phoneNo`**: New phone number value.
- **`String^ dateOfBirth`**: New date of birth value.
- **`Bitmap^ Image`**: New profile image in Bitmap format.
- **`String^ Role`**: Role of the user (e.g., “Student” or “Professor”).

Output Parameters

Doesn’t return anything.

Description

1. Converts the provided Bitmap image to a byte array.
2. Opens a connection to the database.
3. Executes a role-specific SQL query to update user details.
4. Sets parameters for the SQL query with the provided values, executes it, and prints success/error messages.

SQL Queries

Query: For Student Role

```
UPDATE [Student Database] SET Address = @Address, password =
        @Password, PhoneNo = @PhoneNo, DateOfBirth =
        @DateOfBirth, DP = @Image WHERE roll_no = @RollNumber
```

Query: For Professor Role

```
UPDATE [Faculty] SET Address = @Address, password = @Password,
        PhoneNo = @PhoneNo, DateOfBirth = @DateOfBirth, DP =
        @Image WHERE faculty_ID = @RollNumber
```

Function Name: `IsValidPhoneNumber`

Location

File: `updateForm.h`

Input Parameters

- **`String^ phoneNo`**: Phone number to be checked for validity.

Output Parameters

Returns a boolean indicating whether the provided phone number is valid.

Description

1. Checks if the input phone number is `nullptr` and returns `true` in that case.
2. Uses a regular expression pattern to validate the format of the phone number.
3. Converts the managed String^ to a native std::string for regex matching.
4. Returns the result of the regex match indicating the validity of the phone number.

Function Name: `getNextDate`

Location

File: `updateForm.h`

Input Parameters

- **`const std::string &currentDate`**: Current date in the format “DD-MM-YYYY.”
- **`int daysAhead`**: Number of days to move ahead.

Output Parameters

Returns the next date in the format “DD-MM-YYYY” given the current date and days to move ahead.

Description

1. Parses the provided current date into day, month, and year components.
2. Checks the validity of the parsed date using the `isValidDate` function.
3. Adjusts the days in February based on whether the current year is a leap year using the `isLeapYear` function.
4. Iteratively advances the date by the specified number of days.
5. Formats the result into “DD-MM-YYYY” and returns the next date.

Helper Functions

- `isValidDate(int day, int month, int year)`: Checks if the given date is valid.
- `isLeapYear(int year)`: Checks if the given year is a leap year.

Function Name: `updateAdminDetails`

Location

File: `AdminSetDates.h`

Input Parameters

- **`String^ gradeCollection`**: Status for grade collection.
- **`String^ courseReg`**: Status for course registration.
- **`String^ feePayment`**: Status for fee payment.
- **`String^ gradeView`**: Status for viewing grades.

- **`String^ midSemDate`**: Mid-semester start date. (Can be “NULL” to indicate no change)
- **`String^ endSemDate`**: End-semester start date. (Can be “NULL” to indicate no change)

Output Parameters

Doesn’t return anything.

Description

1. Constructs the SQL query for updating Admin details based on the provided parameters.
2. Appends optional parameters (midSemDate and endSemDate) to the query if they are not “NULL.”
3. Retrieves the database connection string.
4. Creates a SQL command with the constructed query and connection.
5. Executes the SQL command to update the Admin details.

SQL Queries

```
UPDATE [Admin] SET
  is_course_registration = @courseReg,
  is_grade_submission = @gradeCollection,
  view_grades = @gradeView,
  start_fee_payment = @feePayment
  -- Optional Parameters
  @midSemDate
  @endSemDate
```

Function Name: `getDetails`

Location

File: `AdminSetDates.h`

Input Parameters

None

Output Parameters

Returns a `std::map<std::string, std::string>` containing details from the Admin table.

Description

1. Constructs and executes a SQL query to retrieve all columns from the Admin table.
2. Retrieves database connection string, creates SqlConnection, and SqlCommand.
3. Uses SqlDataReader to loop through each column, converts String^ to std::string, and inserts into a std::map.
4. Returns the std::map containing details from the database.

SQL Queries

```
SELECT * FROM Admin
```

Function Name: `ViewRecords_Load`

Location

File: `ViewRecords.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

Based on the default year value of comboBox, it fetches data from Admin database and renders on the screen.

SQL Queries

The following are the SQL queries used in this function:

Query 1: To fetch the fees paid in that year.

```
SELECT fees_paid FROM dbo.[Financial Records] WHERE year= year;
```

Query 2: To fetch the total number of students in that year.

```
SELECT no_of_students FROM dbo.[Financial Records] WHERE year=
        year;
```

Query 3: To fetch the total number of teachers in that year.

```
SELECT no_of_teachers FROM dbo.[Financial Records] WHERE year=
        year;
```

Function Name: comboBox1_SelectedIndexChanged

Location

File: `ViewRecords.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

Upon changing the selected yer from comboxBox, it fetches new data from Admin database and renders on the screen.

SQL Queries

The following are the SQL queries used in this function:

Query 1: To fetch the fees paid in that year.

```
SELECT fees_paid FROM dbo.[Financial Records] WHERE year= year;
```

Query 2: To fetch the total number of students in that year.

```
SELECT no_of_students FROM dbo.[Financial Records] WHERE year=
        year;
```

Query 3: To fetch the total number of teachers in that year.

```
SELECT no_of_teachers FROM dbo.[Financial Records] WHERE year=
        year;
```

Function Name: `getisFeePayment`

Location

File: `StudentHome.h`

Input Parameters

None

Output Parameters

Parameter	Description
`bool`	Returns `true` if Fee Payment has started, `false` otherwise.

Description

Retrieves the boolean value of the `start_fee_payment` field from the `Admin` table to determine if Fee Payment has started.

SQL Queries

Query 1: Get start_fee_payment value from Admin table

```
SELECT start_fee_payment FROM Admin;
```

Function Name: `getisCourseReg`

Location

File: `StudentHome.h`

Input Parameters

None

Output Parameters

Parameter	Description
`bool`	Returns `true` if Course Registration has started, `false` otherwise.

Description

Retrieves the boolean value of the `is_course_registration` field from the `Admin` table to determine if Course Registration has started.

SQL Queries

Query 1: Get is_course_registration value from Admin table

```
SELECT is_course_registration FROM Admin;
```

Function Name: `getisFeesPaid`

Location

File: `StudentHome.h`

Input Parameters

None

Output Parameters

Parameter	Description
bool	Returns `true` if student has paid the fees, `false` otherwise.

Description

Retrieves the boolean value of the `fees_paid` field from the `Student Database` table to determine if student has paid the fees.

SQL Queries

Query 1: Get fees_paid value from [Student Database] table

```
SELECT fees_paid FROM [Student Database] where roll_no =
        RollNumber;
```

Function Name: `Button3_Click`

Location

File: `StudentHome.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

Renders `StudentCourseReg.h` if the student has paid the fees else displays an error message.

Function Name: `Button2_Click`

Location

File: `StudentHome.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. If student has not already paid fees then clicking on Button2 a message will be shown of successful fee payment.
2. It updates [Student Database] table by setting fees_paid to 1.
3. It updates [Financial Records] table by adding the fees_paid by the student to fees_paid attribute of table.

SQL Queries

Query 1: Set fees_paid value to 1 of [Student Database] table

```
Update [Student Database] set fees_paid = 1 where roll_no =
        RollNumber;
```

Query 2: Select fees value from [Student Database] table

```
SELECT fees FROM dbo.[Student Database] WHERE roll_no=
        RollNumber;
```

Query 3: Select fees value from [Student Database] table

```
Update [Financial Records] set fees_paid = fees_paid + fees where
        year = 2023;
```

Function Name: `getisViewTimeTable`

Location

File: `ProfDashboard.h`

Input Parameters

None

Output Parameters

Parameter	Description
`bool`	Returns `true` if admin has generated the timetable, `false` otherwise.

Description

Retrieves the boolean value of the `view_timetable` field from the `Admin` table to determine if admin has generated the timetable.

SQL Queries

Query 1: Get view_timetable value from Admin table

```
SELECT view_timetable FROM Admin;
```

Function Name: `getisMidEndDateSet`

Location

File: `StudentDashboard.h`

Input Parameters

None

Output Parameters

Parameter	Description
`bool`	Returns `true` if admin has set the midsem and endsem date, `false` otherwise.

Description

Retrieves the date value of the `midsem_start_date` field and `endsem_start_date` from the `Admin` table to determine if admin has set the midsem and endsem start dates.

SQL Queries

Query 1: Check midsem_start_date and endsem_start_date for NOT NULL

```
SELECT CASE
WHEN midsem_start_date IS NOT NULL
AND endsem_start_date IS NOT NULL
THEN 1 else 0
END
AS Result From [Admin];
```

Function Name: `getisViewTimeTable`

Location

File: `StudentDashboard.h`

Input Parameters

None

Output Parameters

Parameter	Description
bool	Returns `true` if admin has generated the timetable, `false` otherwise.

Description

Retrieves the boolean value of the `view_timetable` field from the `Admin` table to determine if admin has generated the timetable.

SQL Queries

Query 1: Get view_timetable value from Admin table

```
SELECT view_timetable FROM Admin;
```

Function Name: `getisFeesPaid`

Location

File: `StudentDashboard.h`

Input Parameters

None

Output Parameters

Parameter	Description
bool	Returns `true` if student has paid the fees, `false` otherwise.

Description

Retrieves the boolean value of the `fees_paid` field from the `Student Database` table to determine if student has paid the fees.

SQL Queries

Query 1: Get fees_paid value from [Student Database] table

```
SELECT fees_paid FROM [Student Database] where roll_no =
        RollNumber;
```

Function Name: `Button4_Click`

Location

File: `StudentDashboard.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

Renders `StudentTimetable.h` if the student has paid the fees and admin has generated the timetable else displays an error message.

Function Name: `Button5_Click`

Location

File: `StudentDashboard.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

Renders `StudentExamScedule.h` if the student has paid the fees and admin has set the midsem and endsem start dates else displays an error message.