CAMPUSCONNECT DOCUMENTATION

Contents

[Project Scope 2](#_Toc179714486)

[Physical Data Model 3](#_Toc179714487)

[Physical Process Model 4](#_Toc179714488)

[All SQL used: 5](#_Toc179714489)

[Two Generated Reports 8](#_Toc179714490)

[Maintaining child entity of the data model 9](#_Toc179714491)

[User Manual 10](#_Toc179714492)

[Database Schema 11](#_Toc179714493)

[2. Administrator Table 14](#_Toc179714494)

[Academic Record Table 14](#_Toc179714495)

[Module Table 15](#_Toc179714496)

[Relationships Between Tables 15](#_Toc179714497)

[Detailed Time Diary: 17](#_Toc179714498)

[GitHub Collaboration 19](#_Toc179714499)

**Project Overview**

***CampusConnect*** is a student and lecturer management system designed to enhance the academic and administrative efficiency of educational institutions. It supports functionalities such as student registration, module management, mark maintenance, report generation and communication through announcements

**Programming Language**

* C#: The primary programming language used for the development of CampusConnect

**Database Management System (DBMS)**

* SQL Server: Used for creating, managing, and querying the database schema

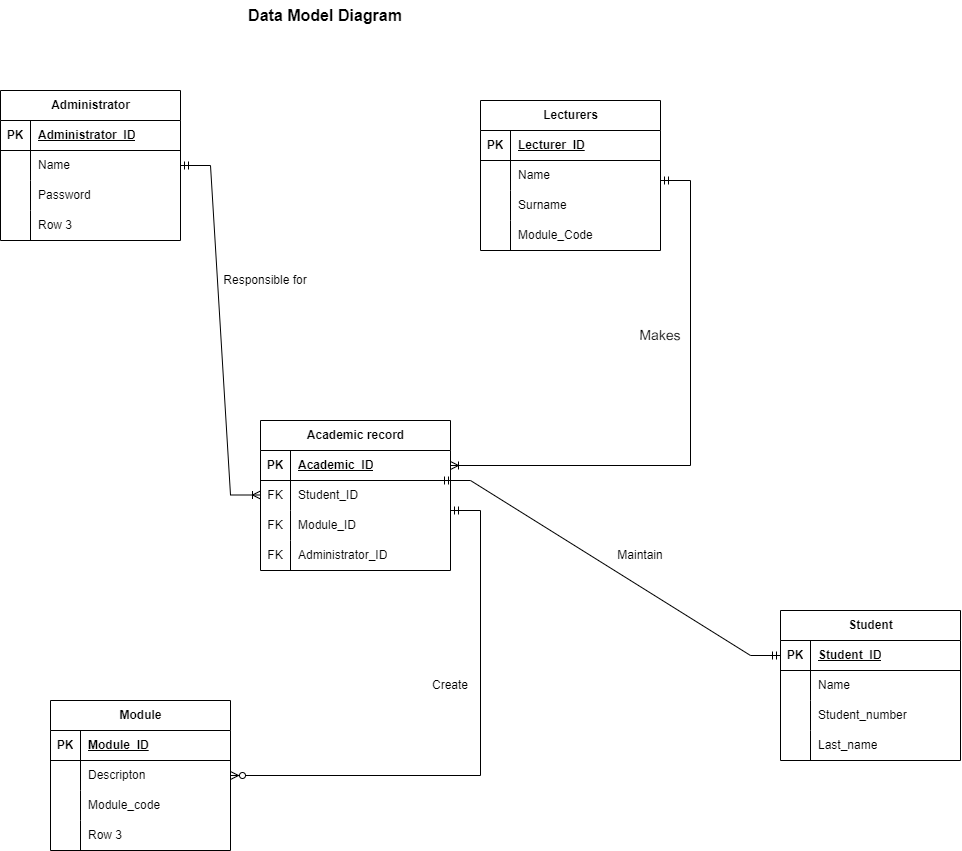
**System Access Credentials**

* Admin Username: 31546455
* Admin Password: 121012
* Lecturer and Student Login: Generated during user registration

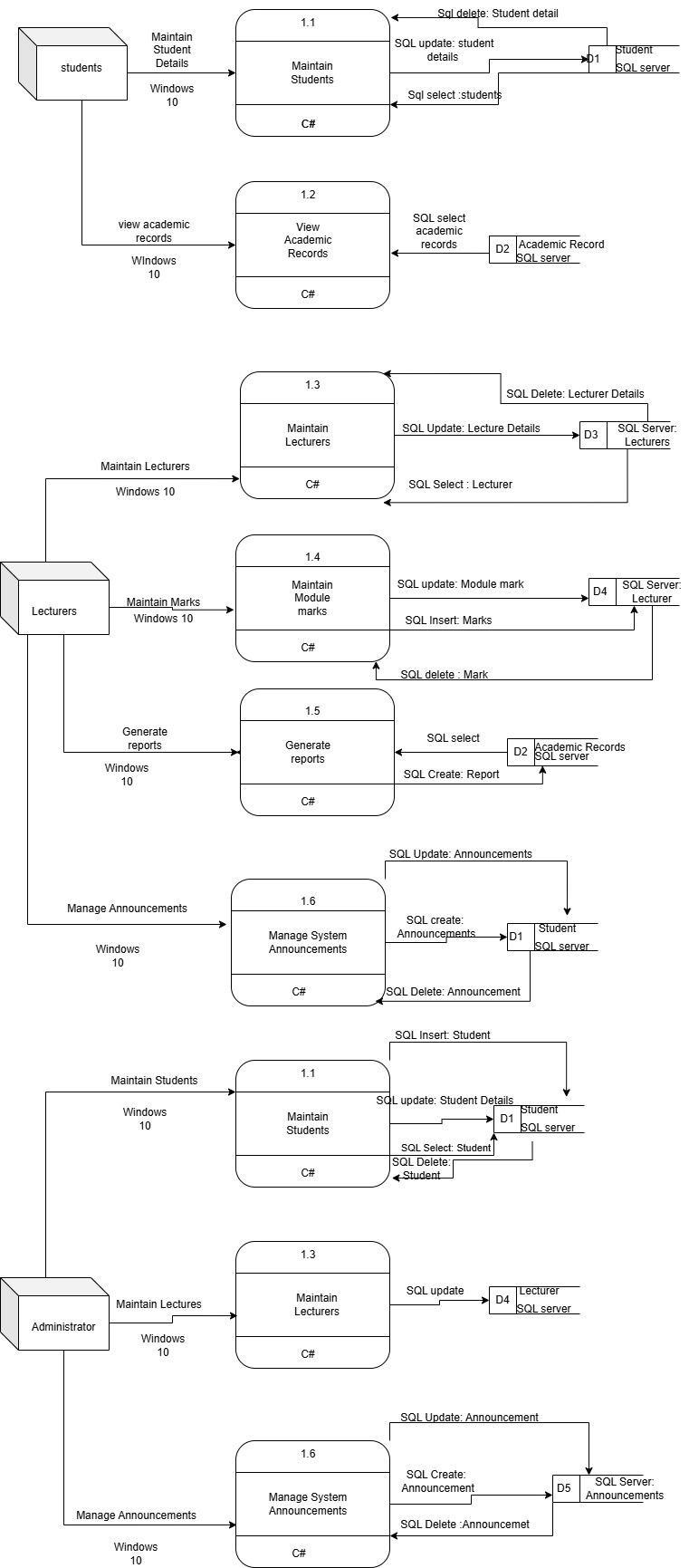
# Project Scope

* Maintains students.
* Maintains module marks.
* Maintains lecturers.
* Generates reports.
* Maintains academic records.

# Physical Data Model

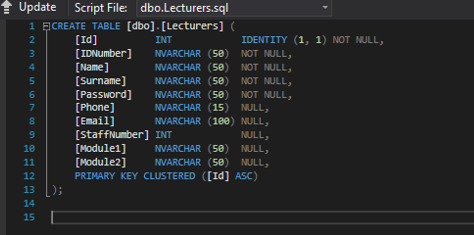


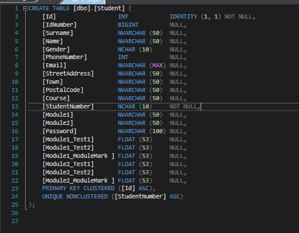
# Physical Process Model

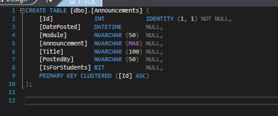


# All SQL used:

CREATE







A screenshot of a computer

Description automatically generated

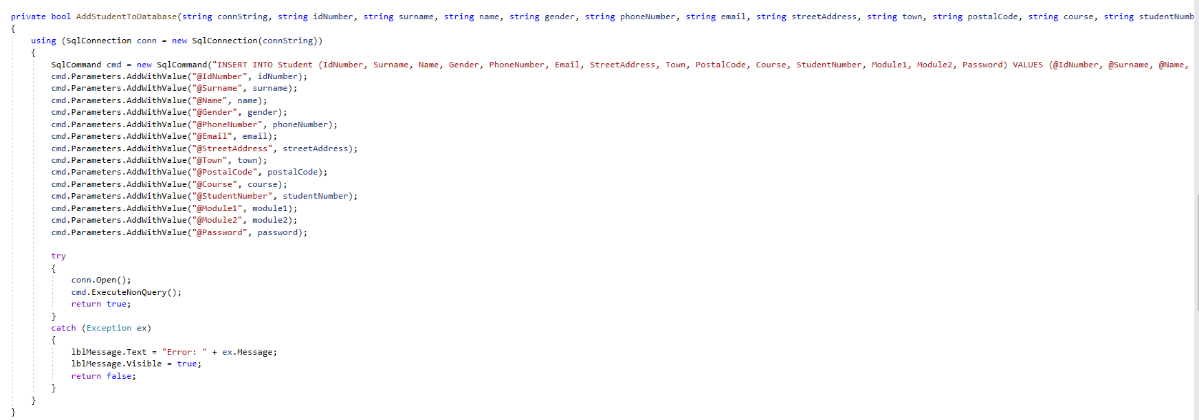
A screenshot of a computer

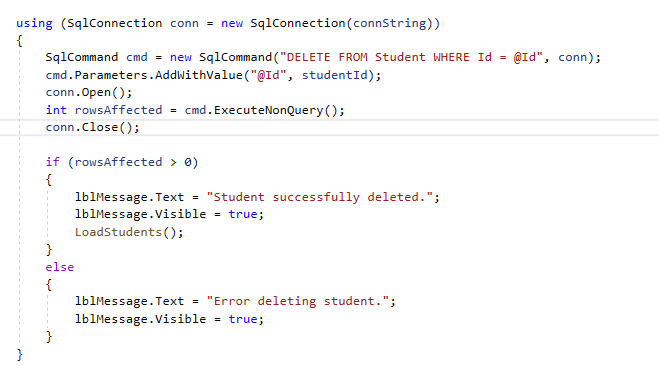
Description automatically generated

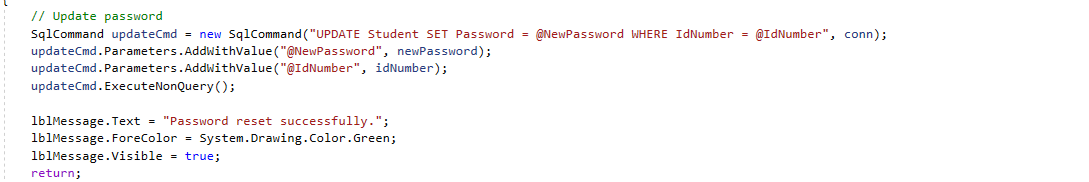
A screenshot of a computer

Description automatically generated

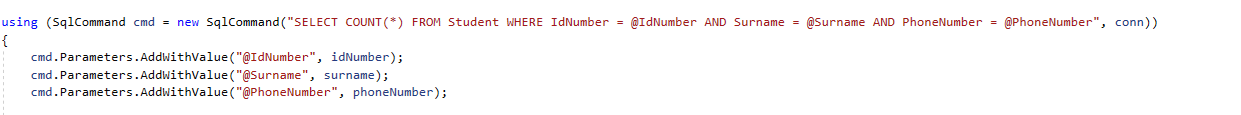
Maintaining All tables in the database (insert, update and delete)

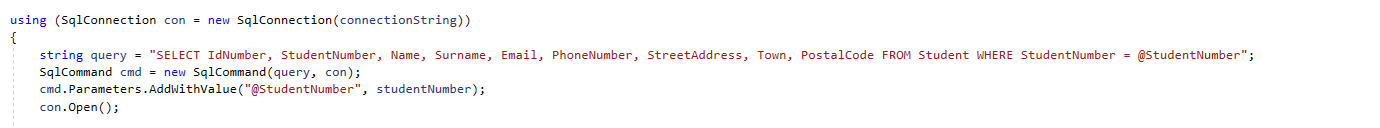






Querying the database(SELECT)

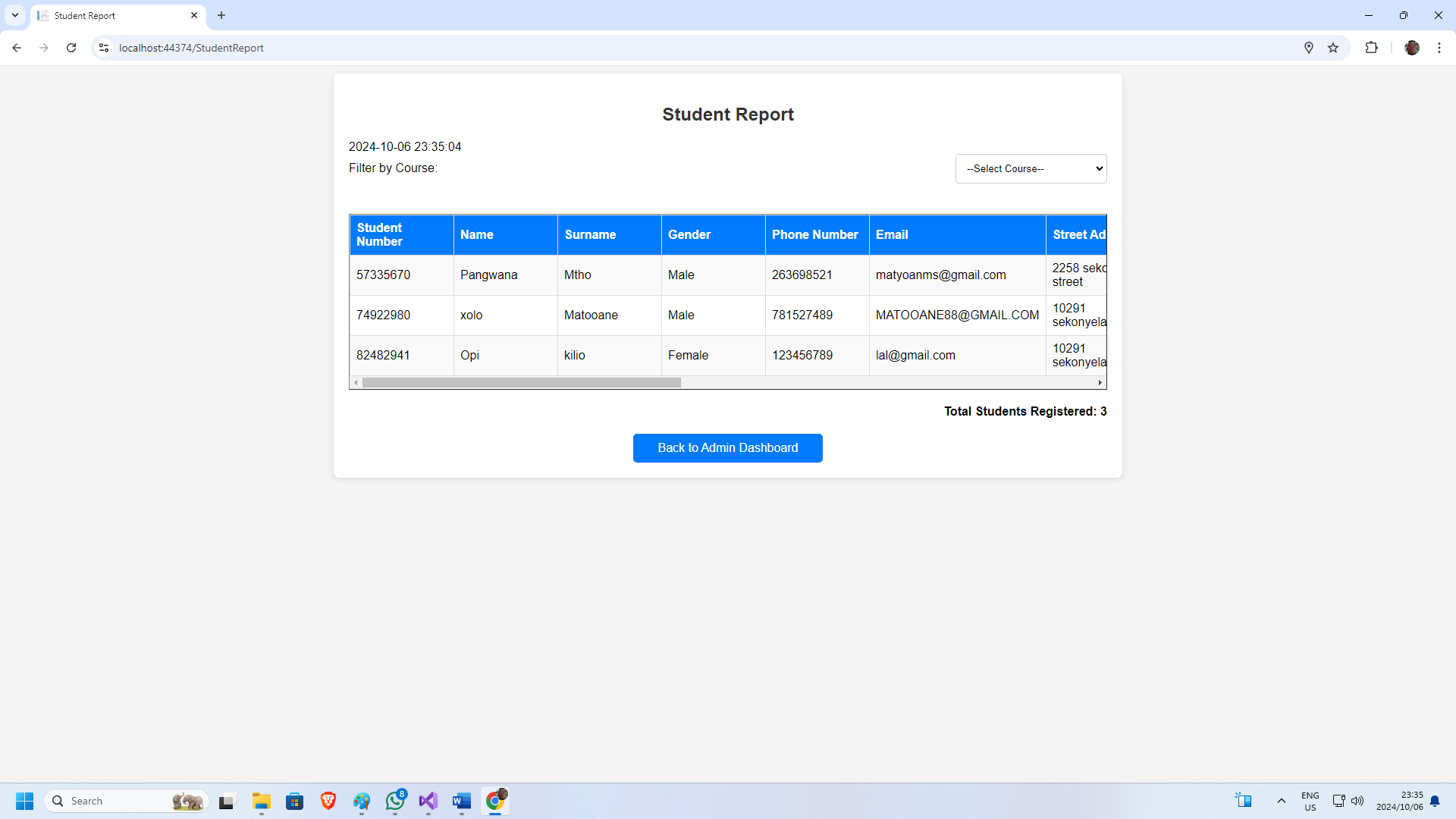




# Two Generated Reports

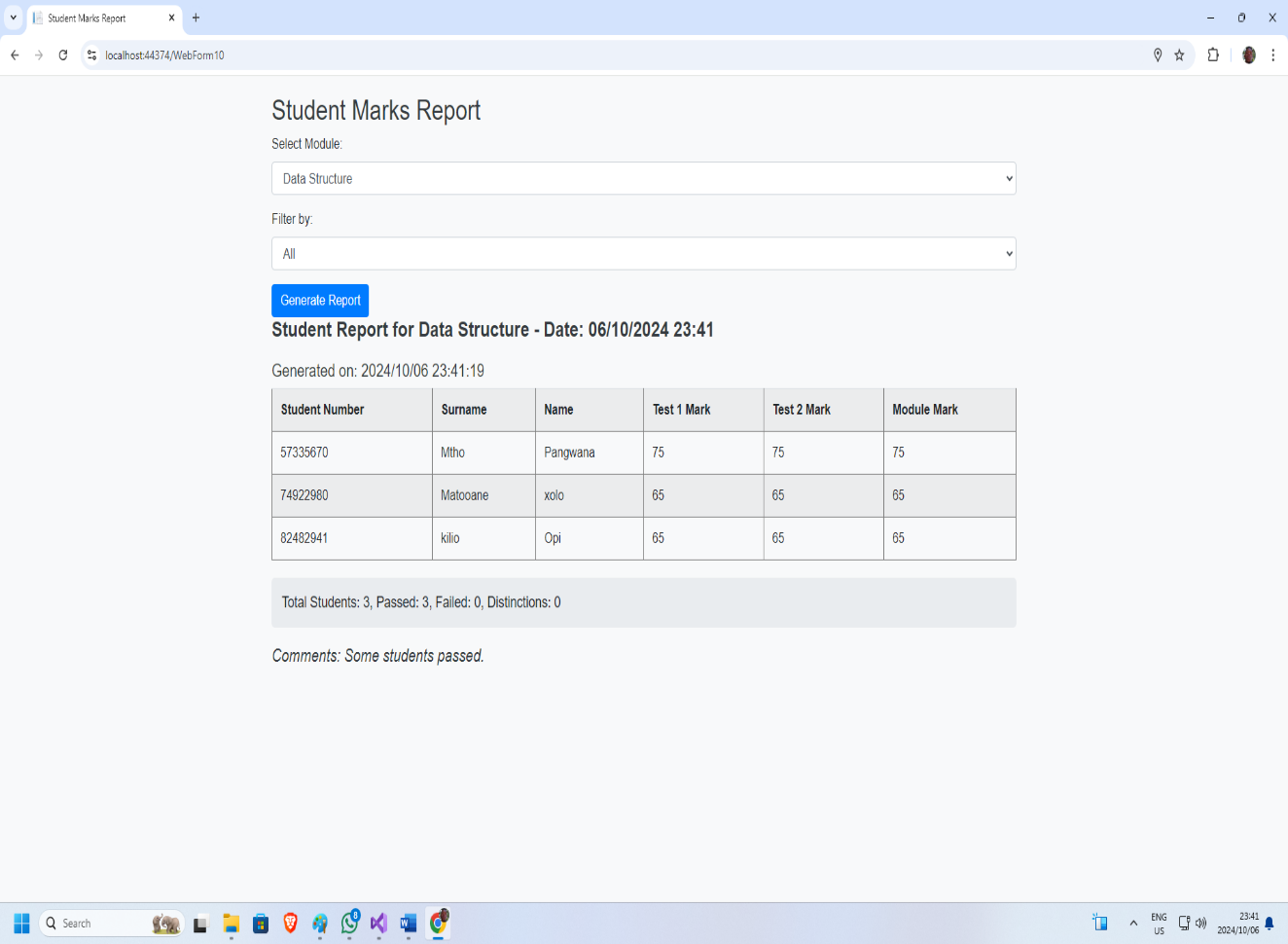
**Summary Report**:

* Uses GROUP BY, SUM, or similar aggregate functions.
* Displayed in a professional layout with clear, well-organized data.

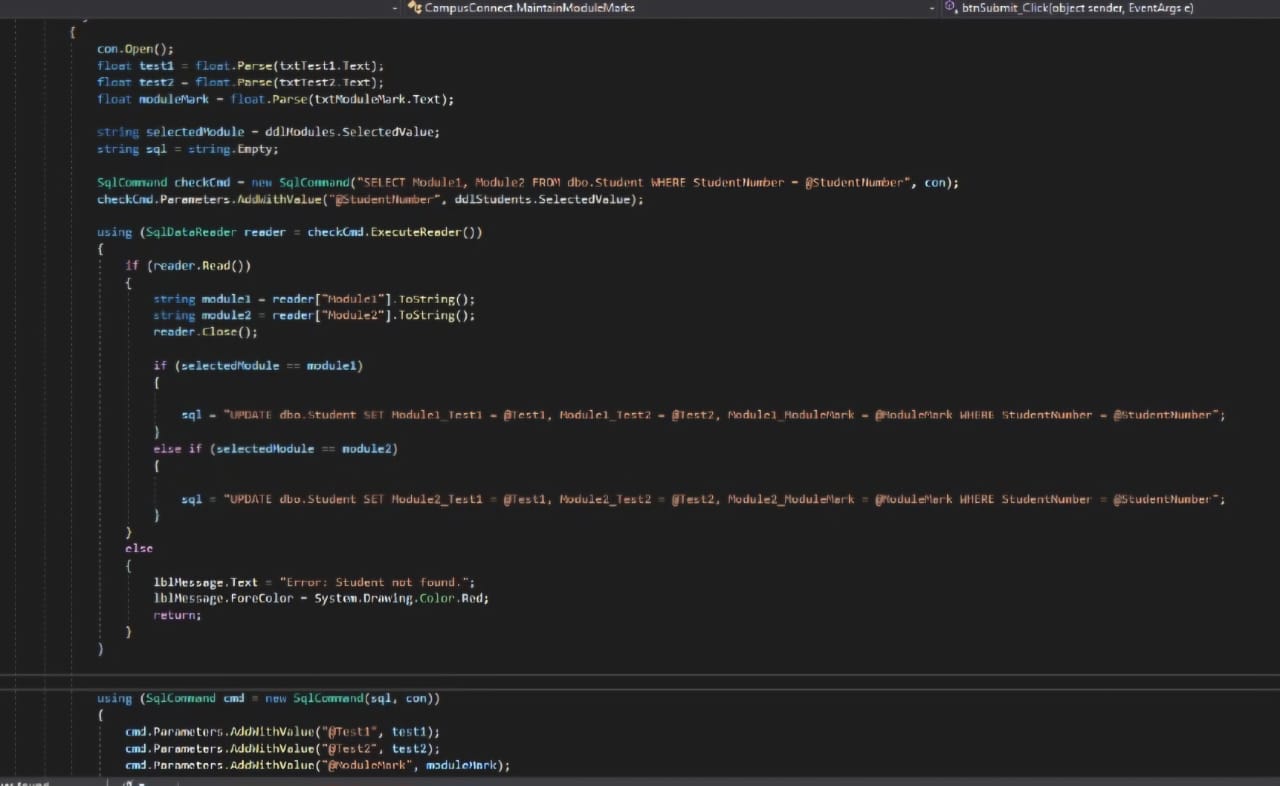


**Detailed Student Performance Report**:

* Shows detailed marks and pass/fail status.
* Includes sorting and filtering capabilities.



# Maintaining child entity of the data model



**Explanation of the Code:**

* **Reusability**: This method makes efficient use of code reusability by defining a function that can be called whenever student marks need to be updated
* **Parameters**: The method accepts parameters like studentId, module1Test1, module1Test2, and module1Exam, making it adaptable to different student records
* **SQL Command**: The SQL UPDATE command is used to modify the relevant fields in the Student table

# User Manual

**Installation Steps:**

1. Download the Project Files: Obtain the CampusConnect project files from the provided link or your project repository.
2. Open Visual Studio: Launch Visual Studio on your computer.
3. **Open the Project:**
   * Click on "File" in the top menu.
   * Select "Open" > "Project/Solution".
   * Navigate to the location where you saved the CampusConnect project files and select the .sln (solution) file.
4. **Configure the Database Connection:**
   * Open the Web.config file in the project.
   * Ensure the connection string is correctly set up with your SQL Server details under ConfigurationManager.ConnectionStrings["Database1"].ConnectionString.
   * Update the server name, database name, username and password as needed to match your database configuration.
5. **Build the Solution:**
   * Click on "Build" in the top menu and select "Build Solution".
   * Wait for Visual Studio to compile the project and confirm that there are no errors.
6. **Run the Application:**
   * Click on the "Start" button or press F5 to run the project.
   * The application should open in your default web browser, displaying the CampusConnect interface.

**System Features**

* **Data Validation**: Ensures all form inputs meet the required formats.
* **Integration Testing**: Validates data entry and output consistency.
* **Forms and Reports**: Designed using user-friendly principles from B&W guidelines.
* **Help and Tooltips**: Built-in assistance and explanations for all forms.

**Extra Information**

* The system has been optimized for performance, ensuring quick data retrieval and seamless navigation.
* User experience enhancements include interactive UI elements like tooltips and guided forms.

**Complexity**

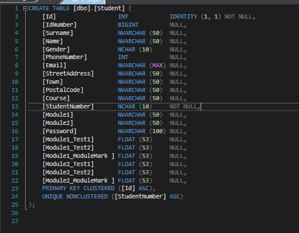
The project complexity is highlighted by the integration of various functionalities such as dynamic report generation, real-time data updates, and efficient data handling.

# Database Schema

The *CampusConnect* database schema consists of three primary tables: **Student**, **Lecturer**, and **Announcement**. Each table is designed to manage essential data related to the project's functionalities effectively.

**1.** Student Table

The **Student** table stores all relevant information about students enrolled in the institution.

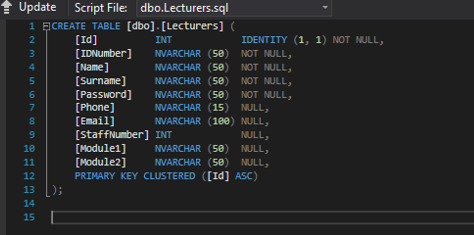


**Attributes:**

* Id: Unique identifier for each student (Primary Key).
* IdNumber: National ID number of the student.
* Surname: Student's surname.
* Name: Student's first name.
* Gender: Gender of the student.
* PhoneNumber: Contact number of the student.
* Email: Student's email address.
* StreetAddress: Home address of the student.
* Town: Town of residence.
* PostalCode: Postal code.
* Course: The course in which the student is enrolled.
* StudentNumber: Unique student number (Unique constraint).
* Module1: First module registered by the student.
* Module2: Second module registered by the student.
* Password: Account password for the student.
* Module1\_Test1, Module1\_Test2, Module1\_ModuleMark: Test scores and final mark for Module 1.
* Module2\_Test1, Module2\_Test2, Module2\_ModuleMark: Test scores and final mark for Module 2.

**2.** Lecturers Table

The **Lecturers** table contains details about academic staff members responsible for teaching and managing modules.

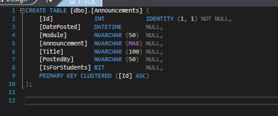


**Attributes**:

* Id: Unique identifier for each lecturer (Primary Key).
* IDNumber: National ID number of the lecturer.
* Name: First name of the lecturer.
* Surname: Last name of the lecturer.
* Password: Account password for the lecturer.
* Phone: Contact number of the lecturer.
* Email: Lecturer's email address.
* StaffNumber: Unique identifier for the lecturer.
* Module1: First module assigned to the lecturer.
* Module2: Second module assigned to the lecturer.

#### **3.** Announcements Table

The **Announcements** table stores messages and updates shared by lecturers and administrators regarding modules



**Attributes**:

* **Id**: Unique identifier for each announcement (Primary Key)
* **DatePosted**: Date and time when the announcement was posted
* **Module**: The module related to the announcement
* **Announcement**: Content of the announcement
* **Title**: Title of the announcement
* **PostedBy**: Identifier for who posted the announcement (could be linked to the lecturer)
* **IsForStudents**: Boolean indicating if the announcement is directed towards students

## 2. Administrator Table

The Administrator table holds information about system administrators responsible for managing the CampusConnect platform.

A screenshot of a computer

Description automatically generated

**Attributes:**

* **Administrator\_ID**: Unique identifier for each administrator (Primary Key).
* **Name**: Administrator's first name.
* **Password**: Account password for the administrator.

## Academic Record Table

The Academic Record table stores the academic performance of students in their respective modules.

A screenshot of a computer

Description automatically generated

**Attributes:**

* **Academic\_ID**: Unique identifier for each academic record (Primary Key).
* **Student\_ID**: Foreign Key linking to the Student table.
* **Module\_ID**: Foreign Key linking to the Module table.
* **Administrator\_ID**: Foreign Key linking to the Administrator table responsible for maintaining this record.

### Module Table

The Module table contains information about the different modules offered by the institution.

A screenshot of a computer

Description automatically generated

**Attributes:**

* **Module\_ID**: Unique identifier for each module (Primary Key).
* **Description**: Detailed description of the module.
* **Module\_code**: Code representing the module in the system.

### Relationships Between Tables

**Student Table and Module Table**:

* + **Relationship**: Many-to-Many
  + **Explanation**: A student can be enrolled in multiple modules, and each module can have many students registered for it. This relationship is typically managed through the academic records, where module registrations and marks are tracked for each student.

**Lecturer Table and Module Table**:

* + **Relationship**: One-to-Many
  + **Explanation**: A lecturer can teach multiple modules, but each module is generally managed by one or more lecturers. This relationship indicates that a lecturer has authority over certain modules, allowing them to manage content, grades, and other academic activities.

**Lecturer Table and Announcements Table**:

* + **Relationship**: One-to-Many
  + **Explanation**: A lecturer can post multiple announcements related to the modules they manage. Each announcement is associated with a single lecturer, reflecting their communication about the relevant module.

**Student Table and Academic Record Table**:

* + **Relationship**: One-to-Many
  + **Explanation**: Each student has a unique academic record for every module they are registered for. This allows for storing individual performance data (test scores and module marks) for each module the student is enrolled in.

**Module Table and Academic Record Table**:

* + **Relationship**: One-to-Many
  + **Explanation**: Each module can have many academic records associated with it, as multiple students are assessed for their performance in each module. This relationship links the module's information with the performance data of the students taking it.

**Administrator Table and Academic Record Table**:

* + **Relationship**: One-to-Many
  + **Explanation**: An administrator is responsible for overseeing the maintenance of academic records. The administrator's actions in updating or managing the records are logged, linking their identifier to the specific records they maintain.

**Administrator Table and Announcements Table**:

* + **Relationship**: Many-to-Many (Potential)
  + **Explanation**: While primarily lecturers post announcements, administrators can also have the ability to manage or oversee announcements in some cases. If administrators directly interact with announcements, this would create a many-to-many relationship.

**Students and Lecturers** interact indirectly through modules since students enroll in modules taught by lecturers.

**Lecturers** use the **Announcements** table to communicate important updates to students regarding their respective modules.

**Academic Records** are crucial for linking students' performance with their registered modules, which are overseen by lecturers.

**Administrator** have control over maintaining these records, ensuring data accuracy and consistency across the system.

* **Normalization**

The database schema adheres to normalization principles to ensure:

* **Minimized Redundancy**: Each attribute is stored in its relevant table without unnecessary duplication.
* **Data Integrity**: All attributes are functionally dependent on their primary keys, ensuring accurate data retrieval and updates.

# GitHub Collaboration

**Project Development and Collaboration Tools**

**GitHub Version Control**

This project utilized GitHub extensively for version control and collaboration. The use of GitHub is highlighted as follows:

* **Version Control**: All code changes were tracked using GitHub, allowing us to manage different versions of the project efficiently.
* **Collaboration**: GitHub facilitated teamwork by enabling multiple contributors to work on the project without conflicts, using branches and pull requests.
* **Project Management**: Issues and tasks were tracked through GitHub's project boards to streamline our development process.

**Link to GitHub Repository**

* The GitHub repository for the *CampusConnect* project can be accessed at:

https://github.com/Phutza/CampusConnect/tree/master/CampusConnect