In this document we will be covering the first part of the Project, Milestone 1

Project 1 Milestone 1

PRG282

Jacques Van Niekerk 577343  
Maryam Jhavary 577869  
Ruben Venter 577663  
Ruan Jansen Van Rensburg 577811

Contents

[Introduction: 2](#_Toc150289193)

[Topic: 2](#_Toc150289194)

[The Team working on project: 2](#_Toc150289195)

[Jacques Van Niekerk 577343 2](#_Toc150289196)

[Maryam Jhavary 577869 2](#_Toc150289197)

[Ruben Venter 577663 2](#_Toc150289198)

[Ruan Jansen Van Rensburg 577811 2](#_Toc150289199)

[Declaration: 2](#_Toc150289200)

[Presentation Layer 3](#_Toc150289201)

[Login and Registration Form: 3](#_Toc150289202)

[Form Features: 3](#_Toc150289203)

[Home Form: 4](#_Toc150289204)

[Form Features: 4](#_Toc150289205)

[Student Details Form: 4](#_Toc150289206)

[Form Features: 5](#_Toc150289207)

[Module Details Form: 6](#_Toc150289208)

[Form Features: 7](#_Toc150289209)

[Business Logic Layer 8](#_Toc150289210)

[General Description 8](#_Toc150289211)

[Class 8](#_Toc150289212)

[Detailed Method Description 8](#_Toc150289213)

[Fields 9](#_Toc150289214)

[Data Access Layer 9](#_Toc150289215)

[Classes: 9](#_Toc150289216)

[Methods For Modules: 9](#_Toc150289217)

[Methods For Students: 9](#_Toc150289218)

[Fields for Modules: 9](#_Toc150289219)

[Fields for Students: 10](#_Toc150289220)

[Extra Information Regarding the Creation and Implementation of Data Access Layer 10](#_Toc150289221)

[Creating the Database/Database design 10](#_Toc150289222)

[Data Layer 14](#_Toc150289223)

[General Information 14](#_Toc150289224)

[Class 14](#_Toc150289225)

# Introduction:

In this document is a detailed report of how we plan to use multi layered architecture inside the application. We are designing an application for Belgium campus that will be responsible for managing student information. We will take an in-depth look at, Presentation Layer, Business Logic Layer, Data Layer, Data Access Layer.

# Topic:

Milestone 1 – Detailed report of how multi layered architecture will be possibly utilize in the application

# The Team working on project:

## Jacques Van Niekerk 577343

* Business Logic Layer
* Word Document

## Maryam Jhavary 577869

* Presentation Layer

## Ruben Venter 577663

* Data Layer

## Ruan Jansen Van Rensburg 577811

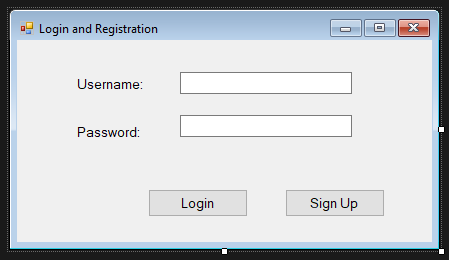
* Data Access Layer

# Declaration:

We, Jacques Van Niekerk, Maryam Jhavary, Ruben Venter, Ruan Jansen Van Rensburg, declare that the work that has been done on this document has been done by us and that no cheating or plagiarism has been done.

# Presentation Layer

## Login and Registration Form:



A screenshot of a computer registration

Description automatically generated

The “Login and Registration Form” is the user's entry point to the program, providing security and ease. Both new users and returning users can easily register and log in to start using the application. Returning users can immediately access their accounts.

### Form Features:

* Username Entry: Login with your distinct username to gain access to your account. You can quickly register a new account if you don't already have one.
* Password Entry: Use a strong password to safeguard your account.
* ‘Login’ Button: Registered users can log in to their accounts with this button.
* ‘Sign Up’ Button: New users can easily register their information then login afterwards with those details.

## Home Form:

A screenshot of a computer

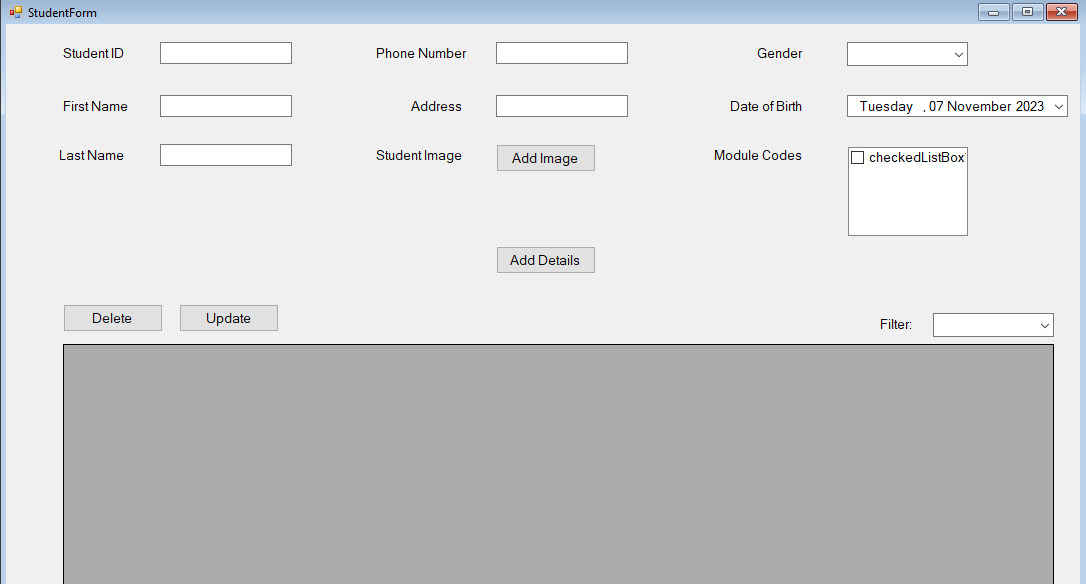
Description automatically generated

The user-friendly style of this form makes it simple for users to browse to various sections of the application, where they can view and modify information on modules and students.

### Form Features:

* "Student Details": This button will take you to a page where you can view comprehensive student data as well as modify them.
* "Module Details": This button will take you to a page where you can view comprehensive module data as well as modify them.

## Student Details Form:



A screenshot of a computer

Description automatically generated

The Student Details form provides a robust platform for viewing, capturing, organizing, and maintaining student data, facilitating effective student administration within our application.

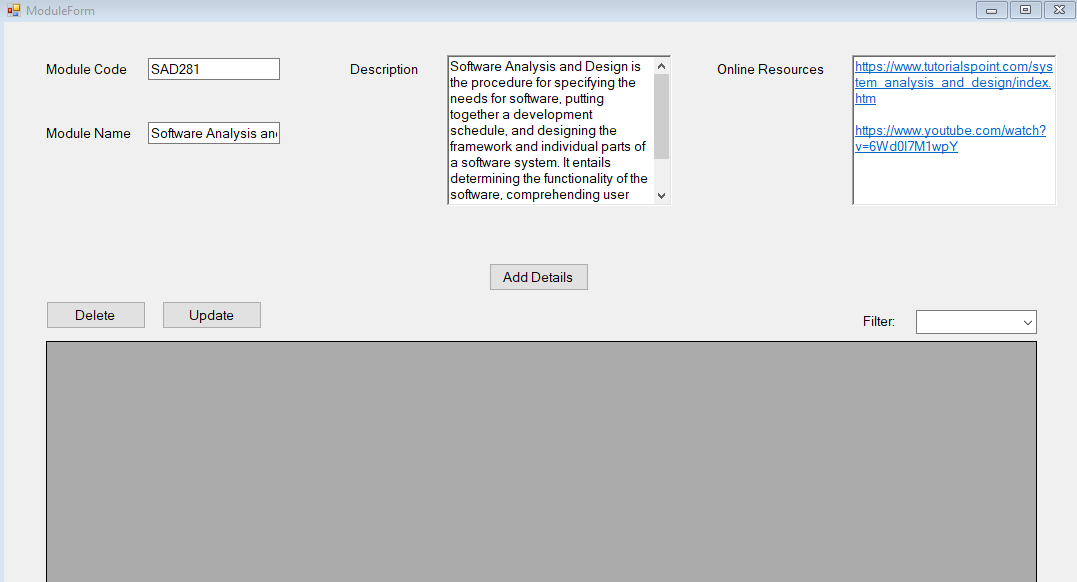
### Form Features:

* Student Number Field: For simple tracking and record-keeping, enter the student's unique identification number.
* Name and Surname Fields: To ensure proper identification, enter the student's entire name and surname.
* Phone Number Field: Enter the student’s phone number in this field
* Gender Field: Choose the gender of the student from the list of choices, which includes "Male," "Female," and "Prefer Not to Say.
* Student Image Field: Provide a picture of the student so that they can be recognized visually.
* Address Field: Keep a record of the student's address for correspondence and future use.
* Date of Birth: To ensure an accurate age calculation, enter the student's birthdate using the date-time picker.
* Module Codes: From a list of options, select the applicable module codes that pertain to the student.
* Add Button: The entered data is saved to the database and is easily accessible for later use.
* Update and Delete Buttons: Easily maintain student information by adding or removing records as needed.
* Options for Filtering: Sort and view student information in either ascending or descending order.
* Data Grid View: All student data is shown in a tabular display for convenient management and reference.

## Module Details Form:

A screenshot of a computer

Description automatically generated



The Module Details form streamlines the process of recording and managing module data, making it a valuable resource for users. Its intuitive design ensures efficient data input, retrieval, enabling users to access and view module information with ease.

### Form Features:

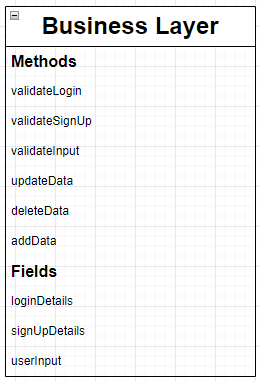
* Module Code: Enter the unique code associated with each module, simplifying identification and retrieval of module information.
* Module Name: Input the name of the module, ensuring clear and concise recognition.
* Module Description: Utilize the rich text capabilities of this field to provide comprehensive descriptions of the modules, facilitating a deeper understanding of their content and purpose.
* Online Resources: This field allows you to insert hyperlinks to online materials, enhancing the learning experience by providing quick access to valuable resources.
* Add Button: The entered data is saved to the database and is easily accessible for later use.
* Update and Delete Buttons: Easily maintain module information by adding or removing records as needed.
* Options for Filtering: Sort and view module information in either ascending or descending order.
* Data Grid View: All module data is shown in a tabular display for convenient management and reference.

# Business Logic Layer

## General Description

The business Logic layer will be responsible for the core functionality of the application. Underneath you can find what are the possible methods and fields that will be used inside the business logic Layer. These methods will be used to validate data and make changes. The business logic layer will take data from the presentation layer and will send data to the data layer.

## Class



## Detailed Method Description

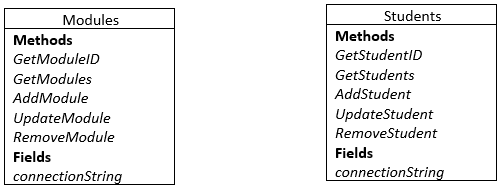
|  |  |
| --- | --- |
| Method Name | Description |
| validateLogin | This method will be used to crosscheck the Login info that the user inputs into the  login/sign-up page with existing user information  This method will contain |
| validateSignUp | This method will be used to validate the Login information that the user inputs into the  sign-up/login page |
| validateInput | This method will be used to store data into variables and to check whether the data that was entered is valid. Validation will check if data types and numbers, letters or characters entered are acceptable. |
| updateData | This method will be used to acquire the new data that needs to be updated. The validateInput method will also be called to store data and validate data. |
| deleteData | This method will be used to acquire the data that needs to be deleted. |
| addData | This method will be used to acquire the new data that needs to be added. The validateInput method will also be called to store data and validate data. |

## Fields

|  |  |
| --- | --- |
| Field Name | Description |
| loginDetails | This is an array. It will be used to store the data that the user enters into the fields of the Login/Sign-up page. Thereafter the array will be split into variables and those variables will be used for validation and to check data. |
| signUpDetails | This is an array. It will be used to store the data that the user enters into the fields of the Login/Sign-up page. Thereafter the array will be split into variables and those variables will be used for validation and to check data. |
| userInput | This is an array. It will be used to store data that the user has added or changed. This data will be then divided into different variables and those will be validated according to data type, values entered, string entered, and characters entered. This will check whether the data is valid or not. Afterwards this data will be used to update or change existing data or add new data. |

# Data Access Layer

## Classes:



## Methods For Modules:

|  |  |
| --- | --- |
| **Method** | **Description** |
| GetModuleID | Retrieves a specific module's information by its unique ID. |
| GetModules | Retrieves a list of all available modules. |
| AddModule | Adds a new module to the database. |
| UpdateModule | Updates an existing module's information in the database. |
| RemoveModule | Deletes a module from the database based on its ID. |

## Methods For Students:

|  |  |
| --- | --- |
| **Method** | **Description** |
| GetStudentID | Retrieves a specific student's information by their unique student ID. |
| GetStudents | Retrieves a list of all available students. |
| AddStudent | Adds a new student to the database. |
| UpdateStudent | Updates an existing module's information in the database. |
| RemoveStudent | Deletes a student from the database based on their student ID. |

## Fields for Modules:

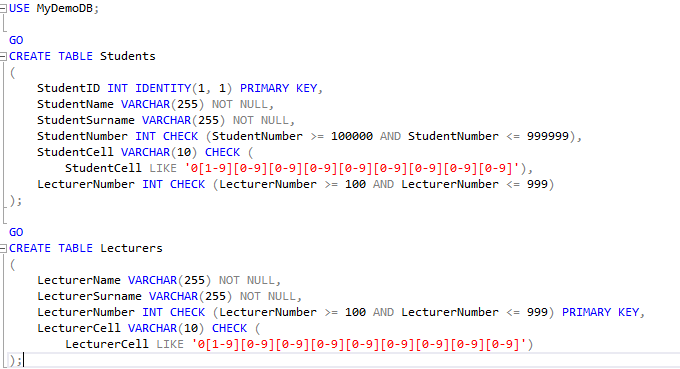
|  |  |
| --- | --- |
| **Field** | **Description** |
| connectionString: | Stores the connection string for the "Modules" database, allowing access and manipulation of module data. |

## Fields for Students:

|  |  |
| --- | --- |
| **Field** | **Description** |
| connectionString: | Contains the connection string for the "Students" database, enabling access and management of student data. |

## Extra Information Regarding the Creation and Implementation of Data Access Layer

### Creating the Database/Database design

* A screenshot of a computer

  Description automatically generatedWe will create tables and have relationships set to them.
* We used SQL to create the tables:

Students Table Creation:

StudentID: This is an auto-incrementing integer (IDENTITY) column, serving as the primary key for the "Students" table.

StudentName and StudentSurname: These are VARCHAR(255) columns for storing student names and surnames. They are marked as NOT NULL, ensuring that they cannot contain NULL values.

StudentNumber: This INT column is used to store 6-digit random student numbers within the range of 100,000 to 999,999. It is protected by a CHECK constraint.

StudentCell: This VARCHAR(10) column stores 10-digit cellphone numbers that start with '0' and have nine digits in the range of 1 to 9.

LecturerNumber: This INT column is included, but it's not described in the comments. It seems like you intended to include it, but there's no information provided regarding its usage in the "Students" table.

Lecturers Table Creation:

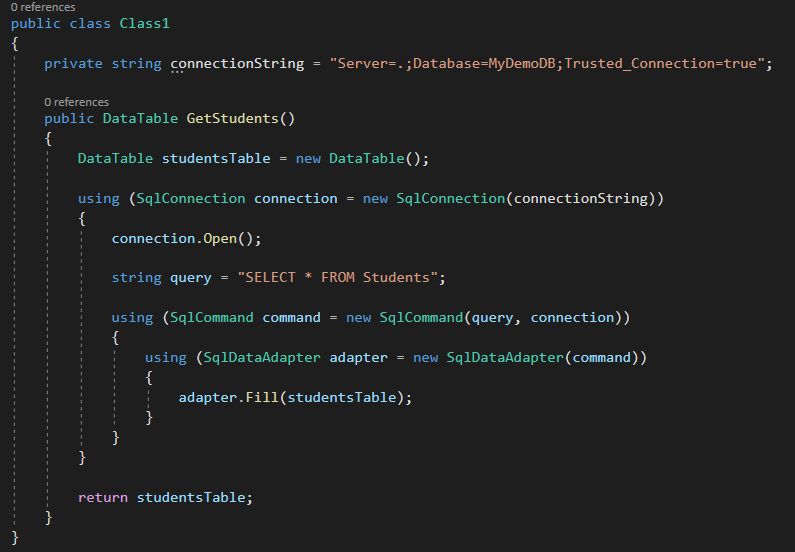
In the "Lecturers" table creation part:

LecturerName and LecturerSurname: These are VARCHAR(255) columns for storing lecturer names and surnames, marked as NOT NULL.

LecturerNumber: This INT column is used to store 3-digit random lecturer numbers within the range of 100 to 999. It is marked as the primary key for the "Lecturers" table, indicating that it uniquely identifies each lecturer.

LecturerCell: This VARCHAR(10) column stores 10-digit cellphone numbers that start with '0' and have nine digits in the range of 1 to 9.

1. Setting up the connection to the database with ADO.net



* Connection String (connectionString): This private field stores the connection string, which specifies the database server, and database name, and uses Windows-integrated security for authentication.
* GetStudents Method:

Method Purpose: The GetStudents method is responsible for retrieving student data from the database.

DataTable (studentsTable): A DataTable object is created to store the results of the SQL query.

* Connection and Command:

Blocks are used to ensure that resources like the SqlConnection, SqlCommand, and SqlDataAdapter are properly disposed of when they're no longer needed.

A SqlConnection is created with the connection string and opened to establish a connection to the database.

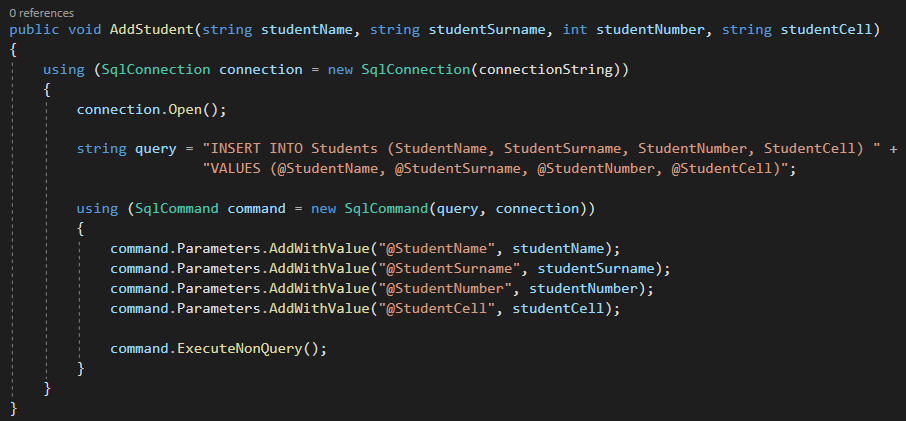
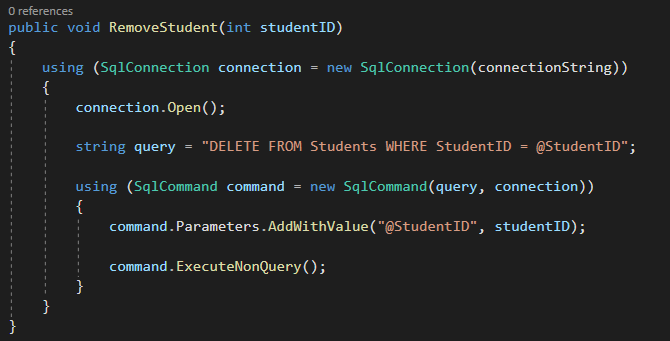
A SQL query, "SELECT \* FROM Students", is defined to retrieve all columns and rows from the "Students" table.

A SqlCommand is created with the query and the open connection.

A SqlDataAdapter is used to execute the query and fill the studentsTable with the results from the database.

* Result: The method returns the studentsTable, which now contains the student data retrieved from the database. The calling code can then work with this data for further processing or display.

1. Add or Remove records from the Students table

* Adding a Record:
  + Purpose: This method is used to add a new student record to the "Students" table in the database.
  + Technology: It is part of ADO.NET, specifically using SqlConnection and SqlCommand to execute SQL queries against the database.
  + Steps:
    - A SqlConnection object is created to establish a connection to the database.
    - The connection is opened.
    - A SQL INSERT query is defined to insert a new record with the specified values for student name, surname, number, and cell.
    - A SqlCommand object is created with the query and connection.
    - Parameters are added to the command to prevent SQL injection and provide values for the query.
    - The ExecuteNonQuery method is called to execute the INSERT query, adding the new student record to the database.
* Removing a Record:
  + Purpose: This method is used to remove a student record from the "Students" table based on the provided studentID.
  + Technology: It is also part of ADO.NET and utilizes SqlConnection and SqlCommand to execute a DELETE SQL query.
  + Steps:
    - A SqlConnection object is created to establish a connection to the database.
    - The connection is opened.
    - A SQL DELETE query is defined to remove a record from the "Students" table where the StudentID matches the provided studentID.
    - A SqlCommand object is created with the query and connection.
    - A parameter is added to the command to specify the studentID to delete.
    - The ExecuteNonQuery method is called to execute the DELETE query, removing the student record from the database.

# Data Layer

## General Information

The user can open the “Samples.txt” text file in order to find the sample usernames and passwords for the application. (The user can also make their own)

The database we are going to be using, is called BelgiumCampusDB. This database has 2 tables for Students and Modules respectively. The student table will contain all the information about every student attending the college, while the Modules table will contain all the information about every module the college is offering.

Below are diagrams containing all the fields in each table and their data types:

## Class

