COSC 2100 – Assignment 5

This assignment is an individual assignment. Submit your work via DC Connect by the due date provided.

# Requirements

Create a college’s admissions application, for new students using C#. To register any student the admission application will need to check and manipulate the following data, until reach answer for the student which program can be admitted to and how much the annual fees (plus Tax):

* Student information:
  + First name
  + Last name
  + Email.
  + SIN (unique value).
  + High school Grade (values: 10,20, ….,90).
  + Admission Test score (values: 10,20, ….,90).
* Admission information:
  + Location of Campus (provinces) {ON, QC, NS, NB, MB, BC, PE, SK, AB, NL }.
  + Program Name {Architectural, Science and Art, Law, Health, Engineering, Music, Business }.
  + Program Fees (per year).
  + Program Registration Fees (per year).
  + Program Study Duration (year:1 - 3).

**Step 01:** The registration officer is capable of inputting the student’s information in section one; student’s first name, student’s last name, student’s ID, student’s email in the text boxes, and select value of High school Grade, and Admission Test score from Combobox.

**Step 02:** Each campus has a different admission requirement and registration fee, each campus has all/some of the programs, and each program has different fees and Study Duration.

At the beginning section two (Available programs) is inactivated, based on student inputs displays the options that are available for this student, starting by active the combo box of campus locations that the student can be admitted to, if there are no available options display a message to inform the student that his application has been rejected, then keep section two inactive and ignore next step, furthermore, Based on student SIN the application should able to check if the student already admitted or not, if yes, display a message that informs the student you have a record in the College system.

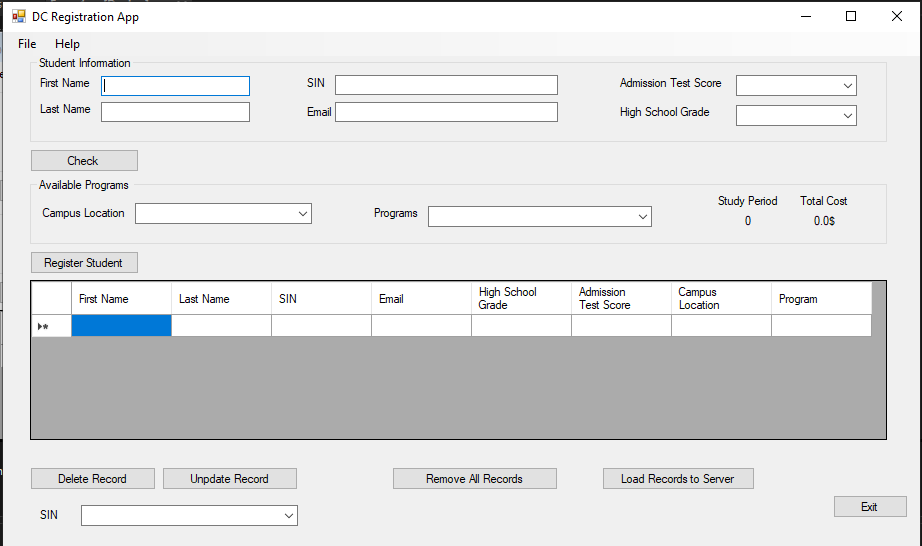
* Possible provinces: {Ontario, Quebec, Nova Scotia, New Brunswick, Manitoba, British Columbia, Prince Edward Island, Saskatchewan, Alberta, Newfoundland and Labrador}

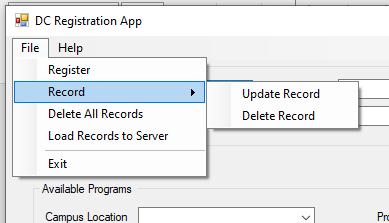
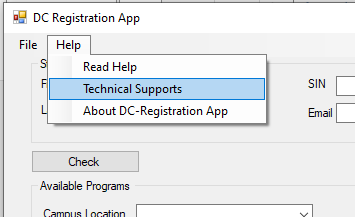
**Step 03:** now the student can be admitted to the College, based on selecting a campus, and activating the 'programs' combo box, which lists the available programs on the selected campus.

**Step 04:** If the student selects the campus and the available program, enter the student's information and admission details into the college records as they appear in DataGridView (Below).

**Step 05:** After the registration officer completes his/her long day should load the entered records to the server (.JSON file).

# Design



# Instructions from the UX (User Experience) Department

* The form should not allow minimizing or maximizing the screen
* The form should have its Exit Button, which is placed on the form and inside the file menu as shown in the previous design.
* The form should open in the centre of the screen
* The 'cost' label should show in currency format (using string formats)
* The ‘Study period’ label should show in decimal formats (0.00).
* ComboBoxes should have their dropdown style set to DropDownList
* ComboBoxes (Campus locations, and available programs) should have their Items collection populated with appropriate data for that list as mentioned above.
* Section Two ‘Available Programs’ should be inactive, before checking the students’ inputs..
* Where possible, all interactive controls (including TextBoxes) should have access keys (hotkeys) set up.
* The Data Grid View should have a number of columns based on the entered information, each student has a unique SIN, and the Data Grid View displays all Details (as shown in the design)
  + “**Read Help**” opens a new form that displays all the details of controls and their functionalities, and this form has its exit button.
  + “**Technical Support**” opens a new form that displays all your details your name, ID, and DC email, and this form has its exit button.
  + “**About DC- Registration App**” opens a new form that displays a simple description of your application, and this form has its exit button.
* All controls (Buttons) on the form and which corresponding controls on menus should have the same functionality

**DOCUMENTAR TODO, SI LOS CAMPUS TIENEN COERTOS PROGRAMAS, ESCRIBIRLO EN EL HEADER, SE PUEDE QUE TODOS LOS CAMPUS TENGA TODOS LOS PROGRRAMAS O QUE CADA CAMPUS TENGA PROGRAMAS DIFERENTES. NO IMPORTA.**

**SI ES ACEPTADO, PONER LA INFO EN EL DATAGRIDVIEW, NO EN EL JSON, EL JSON ES AL FINAL QUE OPRIMA EL BOTON. EL DELETE RECORD DEBE BORRAR SOLAMENTE A UN ESTUDIANTE, ESTO DEBE PASAR EN EL JSON FILE**

**SE PUEDE REMOVER EL SIN CON EL DROPDOWN**

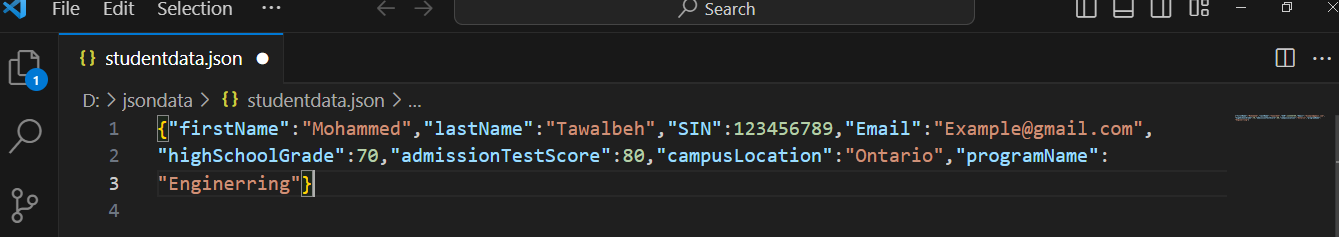
**PARA CADA BOTON, ES MAS O MENOS HACER DOS FUNCIONES, SI EXISTE EL JSON FILE, Y SI NO EXISTE EL JSON FILE**

# General Operation

* A Registration employee can exit by:
  + Clicking the “**Exit”** button (on Form OR from File Menu) with a mouse cursor
  + Using an access key to activate the Exit button
  + Tabbing to the Exit button and tapping Enter on the keyboard
  + **When Click on Exit button:**
    - Ends the application.
* A Registration employee can check the eligibility of a student to be admitted by:
  + Clicking the **“Check”** button with a mouse cursor
  + Using an access key to activate the “Check” button
  + Tabbing to the “Check” button and tapping Enter on the keyboard
  + **When Click on Check button:**
    - Check the eligibility of the student (yes/no).
    - If no, clear the section one “Student information” and display a message that the student application is rejected.
    - if yes, Active the ‘Campus Location’ combobox that list all campuses that student can be admitted to,
    - once the user selects the campus active the ‘programs’ combobox then lists all programs that are available in this campus.
    - After the student selects the program display the program’s study period and the cost of the study period
* A Registration employee can register the student by:
  + Clicking the **“Register”** button (on Form OR from File Menu) button with a mouse cursor
  + Using an access key to activate the “**Register**” button
  + Tabbing to the “**Register**” button and tapping Enter on the keyboard.
  + **When Click on the Register button:**
    - Add the student data to DataGridView as one Record (Row).
    - Add the student’s SIN to the SIN Combobox that appears bottom of the form.
  + **When selecting any row from the DataGrid view:** 
    - display the selected student information in section one “Student Information”
* A Registration employee can Update the student’s Record that corresponds with the SIN selected from ComboBox that appears in the form bottom by:
  + Clicking the **“Update Record”** button(on Form OR from File Menu) with a mouse cursor
  + Using an access key to activate the “**Update Record**” button
  + Tabbing to the “**Update Record**” button and tapping Enter on the keyboard.
  + **When Click on the Update Button:**
    - The application should ask to select the SIN that needs to be updated.
    - Update student information from new entries from section one "Student Information".
* A Registration employee can Delete the student’s Record that corresponds with the SIN selected from ComboBox that appears in the form bottom by:
  + Clicking the **“Delete Record”** button(on Form OR from File Menu) button with a mouse cursor
  + Using an access key to activate the “**Delete Record**” button
  + Tabbing to the “**Delete Record**” button and tapping Enter on the keyboard.
  + **When Click on Delete Button:**
    - The application should ask to select the SIN that needs to be Deleted.
* A Registration employee can Clear the form by:
  + Clicking the “**Remove all Records**” button (on Form OR from File Menu) with a mouse cursor
  + Pressing the **Esc key** on the keyboard
  + Using an access key to activate the **Remove all Records** button
  + Tabbing to the “**Remove all Records**” button and tapping Enter on the keyboard
  + **When Click Remove All button:**
    - All Students and Program information will be removed.
    - All records in DataGrid View will be removed
    - All input controls will clear or be set to their default state
* A Registration employee can upload the records to Database Server (JSON file) by:
  + Clicking the “**Load Records to Server**” button (on Form OR from File Menu) with a mouse cursor
  + Using an access key to activate the ‘Load Records to Server’ button
  + Tabbing to the “**Load Records to Server**” button and tapping Enter on the keyboard
  + **When Click on “Load Records to Server” button:** 
    - Upload the records to the (.JSON) file.
    - All Students and Program information will be removed.
    - All records in DataGrid View will be removed
    - All input controls will clear or be set to their default state
    - Show a message that inform the employee that the data successfully loaded

# Instructions

* For input to be considered valid:
  + All fields in Section one “Student Information” must be entered; these fields cannot be blank or filled only with spaces
  + A comboboxes in this section should be selected do not leave it in default status.
  + A SIN should be unique.
* Campus and Programs inputs:
  + Campus and program details are not inputted values and are not visible to the employee, these values are in the background, values are used to check the eligibility of applicant students, calculate the cost, and fill in some inputs.
  + The class “Program” must be defined in a separate file. the following are class’s properties must exist with these exact names and casing:  
    program {program name, specific fees, study duration} no methods
    - programName (String)
    - programFees (int)
    - programDuration (int)
  + The class “campus” must be defined in a separate file. the following are class’s properties must exist with these exact names and casing:  
    A campus {admission requirement (high school grade, admission test score), registration fee, Available Programs}. No Mothes
    - HSGradeReq (int)
    - admissionTSReq (int)
    - regFees (int)
    - listPrograms (Program) *should be Array List*
* Validation should take place in a Method that takes the input fields - from section one “Student Information” - as parameters, this function will be triggered when check button is pressed.
* For each field that is invalid or empty when the Validation Method is called, an error message specific to that field should be displayed and highlighted the field by red colour.
* The Validation method should also check if the student is eligible to be admitted to one or more campuses, if yes (true) then activate the “campuses” combobox lists these campuses.
* If (false) an error message should be displayed to inform the employee that the student application is rejected, and reset the compoboxes (High school grade and Admission test score) input fields to their default state
* Create an event handler “SelectedIndexChanged” for the “campuses” combobox, once the employee selects the campus, activate the “Programs” combobox and add the available programs in the selected campus, displays the program’s duration of study, and the total cost plus registration fees.
* The class “Student” must be defined in a separate file, this class represent each record, the following are class’s properties must exist with these exact names and casing:
  + - firstName (String)
    - lastName (String)
    - SIN (int)
    - Email (String)
    - highSchoolGrade (int)
    - admissionTestScore (int)
    - campusLocation (string)
    - programName (String)
  + One parameterized constructor is required and includes needed parameters representing all previous properties.
  + Setter and getter methods for all class’s properties.
* An Array List of "Student" data types must be created inside your main form Class, and should be public.
* If the student accepts this offer, a list of functions should be done when the Register button is pressed, these functions are:
  + check if the student does not have another admission, and if yes (true) display a message that informs the employee that the student already has a record in the college’s system, and can’t be admitted in more programs, and ask if he would like to delete or update the previous record.
  + if no (false) add the students to college records, and add a student to the DataGrid view as one record, in addition, create a student object with these entries and add this object to the student array list.
  + furthermore, add student’s SIN to SIN’s Combobox for any future change whatever Update/Delete.
* For any requested update, when the Update button is pressed the application should ask the employee to select the Student’s SIN from SIN’s Combobox then based on the student’s SIN retrieve the student’s information from DataGrid View to the section one "Student information" except campus location and program and return the compoboxes (Campuses and Programs) to their inactivate status, delete this object from student array list, and start from the beginning to double check if the student still eligible to be admitted with new entries.
* For any requested delete, when the delete button is pressed the application should ask the employee to select the Student’s SIN from SIN’s Combobox then delete the student’ record from DataGrid View, delete this object from student array list, and delete the SIN from SIN’s Combobox,
* There must be a function (method) to reset the form's input fields to their default state. This function will be triggered when either the “Remove all records” or “Upload Records to Server” buttons is pressed.
* For loading data you need to download and install Json.NET and Newtonsoft.JSON packages, when "Load records to Server" is pressed call function (method), this function takes an ArrayList of students as a parameter, which does a loop to pass all items in this ArrayList, this function call “SerializeObject(studentObj)” located under JsonConvert class, is used to convert object data to special string format, this format is used to write in JSON format as showed below.
* Ensure that your design incorporates suitable exception handling.



# Additional Considerations

* The program must be adequately documented:
  + methods, functions and event handlers should all have block comments
  + calculations, decisions and iteration should be explained with brief comments
  + there should be a header at the top of each human-generated code files, including your name, the last modified date, and a description
* Adhere to an approved style guide and ensure your variable names, form controls, and other elements are properly cased and adequately descriptive.

# Assessment

Each student’s work will be submitted individually as a compressed folder containing the entire solution. The instructor will assess Your completed work using the COSC2100 Assignment 4 Rubric available on DC Connect. Reasonably detailed feedback will be provided.