COMPX223 Intro and Project Guide (solo work)

**Note these instructions are for the Windows lab machines*

Locating Software:

First off, we are going to need to know the software we will use in COMPX223. We will be using three main software over the duration of this course. They are "Database Manager", "SQL Server Management Studio" and "Visual Studio".

Left-click the **Start button** in the bottom left hand corner of your screen. In the **Search bar** type in **Database Manager**. Above the search bar there will be a **Programs** section that has the program we want to open.

Repeat the above paragraph with **SQL Server Management Studio** and latest version of **Visual Studio**.

Creating a Database:

For all of your projects, tests and practical exercises you will generally need to create a new database for each. So make sure that the name is useful and unique so you do not get confused later on down the track.

Open the program **Database Manager** by Left-clicking the icon on your **Task Bar** or **Start Menu**. When opened you will be in the **Manage Database** section, switch to the other section by Left-clicking the **Add New Database** button at bottom right of program.

In the **Create Database** section, type in the New Database Name textbox type in **INTRO_USERS** and make sure Database Model is empty (is the default). Note that the database will be created with your username at the front, e.g. **username_INTRO_USERS**. The underscores are in use because the program does not allow database names with spaces. (Try it with a space to see the error)

Left-click **Create New Database** button and let it load. It should automatically take you back to the **Manage Databases** section. Alternatively, you can switch sections by Left-clicking the **Manage Databases** tab or **Create Databases** tabs near the top.

If you make a mistake when creating a database, navigate to the **Manage Databases** section and Left-click on the database you want to delete. Once you are certain the highlighted database is the one you want to delete, Left-click the **Delete Selected Database** button. A pop up window making sure you actually wanted to delete it will show, giving you the option to delete it or take back your decision. If you clicked **yes**, you will notice that the database has been removed from your list of databases.

Using SQL Server Management Studio:

Once we have created a database, we now want to do something with that database. Remember that if you do not understand something use the internet and try find it out yourself, you might learn a few extra tips to help you with your work.

Open **SQL Server Management Studio**. In the **Connect to Server** window that pops up leave Server type as **Database Engine**, in the Server name type in **cairo.cms.waikato.ac.nz** and leave the Authentication as **Windows Authentication**. Left-click the **Connect** button to access the server where you created your database earlier.

We now want to create a new query, Left-click the **New Query** item near the top left of the program below **File** etc. This will create a new query for you to write SQL code on using the current **Cairo** connection. Alternatively you can select from **File > New > Query with Current Connection**. You can also use the keyboard shortcut pressing both **CTRL + N**.

Below and to the right of the **New Query** menu item you can see a drop down box with **master** in it. Left-click to make it drop down and show all the connected databases, look for the database **username_INTRO_USERS**. For example, my ones are named wjkw1_INTRO_USERS.

Once you know the name of the database you are using, you can put a small piece of code at the top of your query to make it easier to find the database that this query will use. Now type in the following statement at the top of your query page:

```
USE username_INTRO_USERS
```

Now when you first open this script you will be able to Execute this first line to save you the hassle of searching through all the databases to find your one. This trick will be handy when you have more databases to manage.

Copy, Paste, Cut, Undo and Save are all the same keyboard shortcuts we are used to. I.e., **CTRL + C, CTRL + V, CTRL + X, CTRL + Z, CTRL + S**.

If you have created a document previously you can open it from SQL Server Management Studio, Left-click on **File > Open > File…**and alternatively you can use the keyboard by pressing **CTRL + O**.

A useful tip is that you can execute part of the script instead of all of it. If you highlight an area of script you want to **execute** (can be one line to as many as you want) and then press **ALT + X**.

Creating Tables:

We should be familiar with ER Diagrams and their transformation to tables. A lot of the learning received in this course when creating tables and running queries is from practice. Make sure that doing the practical's is a priority, as it will help you throughout the course and in tests. When creating tables it is helpful to be aware of all the different types of data we can store, check lectures slides for info.

Open **SQL Server Management Studio** if it is not open already. Connect the database to your **INTRO_USERS** database. The code shown below is explained in the COMP223: Intro and Project Guide - video_02. Now type in the following lines of code:

```
CREATE TABLE users
  (
     username   VARCHAR(20) NOT NULL PRIMARY KEY,
     firstname  VARCHAR(20) NOT NULL,
     middlename VARCHAR(20),
     lastname   VARCHAR(20) NOT NULL,
     password   VARCHAR (20) NOT NULL
  )

CREATE TABLE socialmedia
  (
     socmediaid      VARCHAR(3) NOT NULL PRIMARY KEY,
     socialmedianame VARCHAR(30) NOT NULL
  )
```

```
CREATE TABLE usersession
  (
     sessionid   INT NOT NULL IDENTITY(1, 1) PRIMARY KEY,
     socmediaid  VARCHAR(3),
     username    VARCHAR(20),
     sessiontime INT, --assumed as minutes for simplicity
     FOREIGN KEY(socmediaid) REFERENCES socialmedia,
     FOREIGN KEY(username) REFERENCES users
  )
```

If you ever make a mistake when running the table creation script, the easiest way to fix it is to delete the table and re-make it correctly. This is the same if you insert data into the table and that data turns out to be wrong also. If you want to delete Users table, then you must delete UserSession table first since it foreign keys from Users and Social Media. This means you will need to re-create both of them after deleting them. An example of the code we would use in our case is:

```
DROP TABLE usersession
DROP TABLE users
DROP TABLE socialmedia
```

Inserting Data into Tables:

Now that we have some tables to experiment with, it is a good idea to insert some data into them. This is just an intro to what the code looks like, in order to understand what it is doing be sure to go to lectures and complete all given practicals.

The following script will insert some data into the **Users**, **SocialMedia** and **UserSession** tables. There are many ways you can insert data into tables, you can even update existing data by using the **UPDATE** keyword. The lectures and practicals will show you how to do this. Execute the following script:

```
--This inserts data into Users table

INSERT INTO users VALUES ('jkc1', 'John', 'Middle', 'Carter', 'pass1')
INSERT INTO users VALUES ('amo1', 'Amos', 'Carter', 'Orange', 'pass2')
INSERT INTO users VALUES ('wkc1', 'Wong', 'Caleb', 'Cartel', 'pass3')
INSERT INTO users VALUES ('ddj1', 'Daisy', 'Day', ' Johnson ', 'pass4')
INSERT INTO users VALUES ('dps1', 'Dayne', 'Pint', ' Shipper ', 'pass5')
```

```
--This table links the two tables together
INSERT INTO socialmedia VALUES ('FB', 'Facebook')
INSERT INTO socialmedia VALUES ('TWT', 'Twitter')
INSERT INTO socialmedia VALUES ('IG', 'Instagram')
INSERT INTO socialmedia VALUES ('YT', 'YouTube')

--This stores data into UserSession linking Users and SocialMediaa
INSERT INTO usersession VALUES ('FB', 'jkc1', 256)
INSERT INTO usersession VALUES ('TWT', 'amo1', 20)
INSERT INTO usersession VALUES ('IG', 'wkc1', 60)
INSERT INTO usersession VALUES ('YT', 'ddj1', 180)
INSERT INTO usersession VALUES ('FB', 'dps1', 25)
```

Checking the Data in the Tables:

Now that each table holds data, we need a way to see the data. We will use some basic select statements to see our data inside the tables. As you will find as the course goes on, these statements can be very complex. So make sure you complete the practicals, as they will greatly increase how well you do in the tests. Execute the following statements:

```
SELECT * FROM users
SELECT * FROM socialmedia
SELECT * FROM usersession
```

The data that it returns should look like the following if you select the **Results to Text**. To set it to this, click on menu item **Query > Results to > Results to Text**. Alternatively, you can press **Ctrl + T** switch to results as text. The returned results will look like the following:

```
UserName        FirstName  MiddleName       LastName         Password
-------------------- -------------------- --------------------------
amo1            Amos       Carter           Orange           pass2
ddj1            Daisy      Day              Johnson          pass4
dps1            Dayne      Pint             Shipper          pass5
jkc1            John       Middle           Carter           pass1
wkc1            Wong       Caleb            Cartel           pass3
(5 row(s) affected)

SocMediaID      SocialMediaName
---------- ------------------------------
FB              Facebook
IG              Instagram
TWT             Twitter
YT              YouTube
(4 row(s) affected)
```

```
SessionID      SocMediaID     UserName     SessionTime
----------- ---------- ------------------- -----------
1              FB             jkc1         256
2              TWT            amo1         20
3              IG             wkc1         60
4              YT             ddj1         180
5              FB             dps1         25
(5 row(s) affected)
```

Starting on the Application:

From here, we will be using **Visual Studio** to create an application that will link to our database. It will allow us to log in to the program as a user, and let us to register to the database. Hopefully you are familiar with C# coding, if not then be sure you check out the C# tutorials on moodle (and let the tutor know).

Download the skeleton code from Moodle, the file is called **INTRO_USERS**. Run the Visual Studio file inside the main file by double clicking on it or right clicking it and selecting **Open**. Be sure to read the comments in each of the classes as they tell you what certain sections do. Also make a note of the control names, for example, **textBoxUserName** is the first text box on the login pages name.

Normally, we would first need to do is create an **SQL class**. This is achieved by Left-clicking **Project > Add class…** > changing class name to **SQL.cs** from **Class1.cs** > then left-click the **Add** button to create it. However, to save you time there is a **SQL class** already added to the project.

Changing the Database Connection:

Navigate to **SQL.cs**, if you check the right hand side near the top there is the **Solution Explorer**. Left-click on the little triangle to the left of **INTRO_USERS** text to expand out the folders. Leftclick on **SQL.cs** to get to the **SQL** code if it isn't tabbed already. Use the internet to see tutorials on getting to the **Solution Explorer** if it is not there for you.

Once you have the **SQL class** open, we will need to match the classes connection to your database **INTRO_USERS**. The first line of code where we create the **SqlConnection** object is the one we need to change.

We need to connect to your database within the **SQL class**. To do this change the **Database=wjkw1_INTRO_USERS** to your database name that will be similar except it will have your own username. Do not forget the semi colon or you will get an error. For example **Database=*your_username*_INTRO_USERS** or whatever your database is called. The methods within the **SQL class** are very similar to how you would use the **Math class**. Again, just make sure you have a read of the comments within the classes, as they are good with helping you understand the code.

Select Queries on the Login Page:

Navigate to the **LoginPage.cs**, not the designer page but the page with code. If you are on the designer the press **F7** to get to the code. Also, if you want to look at the designer but are on the code then press **Shift + F7**. Users are going to be able to log in using their username and password. For simplicity we will not take it to a separate page once logged in like you would in your project, instead we will display in a message box the result of each attempt. Use the comments in the code to help you. For a more comprehensive breakdown of the code be sure to watch the video where we create this same application.

Go ahead and type the following code below the comments with numbers into the **buttonLogin_Click** event method:

```
(1)
//Get the   username   and   password   and   store as    string
      username = textBoxUserName.Text.Trim();
      password = textBoxPassword.Text.Trim();
(2)
//Use a     select      query on    the   users table
      SQL.selectQuery("SELECT     *     from  Users");
(3)
//If  there exists      at     least one   User
if (SQL.read.HasRows)
{
      //cycle    through    all   users checking    if    the   username    exists     and
      if    the   password    matches
      while (SQL.read.Read())
      {
            //SQL.read[i]:  i=0   is    UserName    ...   i=4   is    Password
            if (username.Equals(SQL.read[0].ToString()) &&
password.Equals(SQL.read[4].ToString()))
            {
                  //Username and   Password    correct,    get   fname,     lname to
      display
                  loggedIn = true;
                  firstname = SQL.read[1].ToString();
                  lastname = SQL.read[3].ToString();
                  break; //stops   the   while loop  since they  have  logged     in
            }
      }
}
else
{
      //Error    message    to    show  that  no    users have  been  registered
      MessageBox.Show("No   users have  been  registered.");
      return;
}
```

This is the main part of the code on the **Login Page** that allows us to query data from the database and log in by comparing what we have in our textboxes. The code to display the result of the log in is not complicated and you should already know how to code it. Make sure to have a look at the comments and making sure you understand the code. Try logging in using username: **jkc1** password: **pass1** and see if it works. If it doesn't then check the connected database is correct.

Insert Statements on the Register Page:

Now we are going to register someone and add them to the database. We do this using the **SQL** class's **executeQuery** method. We will get the user data from textboxes and store it in the database using an **insert** statement. It is best to make sure you know the how to code SQL **insert** statements on **SQL Server Management Studio**, the way we do it in code is very similar, except we replace the text for usernames as variables so it changes based on the user input.

The actual **insert statement** in code will need a lot of back slashes because of quote marks within speech marks. First, navigate to **RegisterPage.cs** using the **solution explorer** or using the tabs along the top. Now type the following code into **buttonRegister_Click** event method using the comments for help:

```
(1)
//GET the   users data  and   store into  variables
      username = textBoxUserName.Text.Trim();
      password = textBoxPassword.Text.Trim();
      firstname = textBoxUserName.Text.Trim();
      middlename = textBoxPassword.Text.Trim();
      lastname = textBoxUserName.Text.Trim();
(2)
//Using    the   SQL   class executeQuery,    INSERT     statement  to    put   data  into
      database
// INSERT  INTO  Users VALUES ('jkc1',  'John',    'Middle',  'Carter',  'pass1')
      SQL.executeQuery("INSERT INTO Users VALUES ('" + username + "', '" + firstname +
"', '" + middlename + "', '" + lastname + "', '" + password + "')");
```

When coding the execute statements this way a lot can go wrong if you do not look carefully at the statement. If you are getting errors from the statement then try **back slashing** the quotation marks e.g. "… \' …" this will ensure that the quotes are not being confused as **char quotes**.

Adding to the Browse Page for the Application:

We have already added a **BrowsePage** form on our program. Normally to add a new form we would just Left-click **Project > Add Windows Form…** give it the name **BrowsePage.cs** and then Left-click **Add**. So be aware of how to, but know that in this case it is added for you.

You should know how to add controls to the form and how to rename them also, if you do not then refer to the C# help on Moodle before continuing.

On the **BrowsePage.cs (Design)** we have an user interface that will allow the end user to search through the **UserSessions**. Note that this interface makes use of two **combo boxes** for user input, and has a **list box** to display our search results. The idea behind this page is something you would normally create for an admin, not any user. However, for simplicity we will allow any user to access this information. This page is going to let us see all **UserSessions** for a particular **User** and **Social Media**.

The **SQL.cs** class has a method that allows us to update a combo box based on a query given. Because a combo box can only hold one piece of data at a time, the update method will only take into consideration the first column of data returned from the query. If we query **"SELECT * from Users"**, which returns the complete **Users** table, we would get a combo box full of usernames i.e. the **Users** first column. However, if we were to query **"SELECT * from UserSession"** which has a dynamic first column, we would be given just a whole bunch of numbers in the combo box. So be careful when using this method.

We will be pulling data from the database and inserting it into our combo boxes. In the **public BrowsePage() { … }** method below the **InitializeComponent();** line of code type out the following code:

```
string usernameQuery = "SELECT  UserName  FROM Users";
string socialMediaQuery = "SELECT    SocialMediaName FROM SocialMedia";
SQL.editComboBoxItems(comboBoxUsername, usernameQuery);
SQL.editComboBoxItems(comboBoxSocialMedia, socialMediaQuery);
```

If the queries are at any time left blank you will run into errors. Make sure that in your own programs you put in place try-catches to keep the program running smoothly.

Making a More Complex Query:

When querying in visual studio using the coding method, make sure you test that the queries work on **SQL Server Management Studio** first. This will ensure that the query is fine, but any errors that arise are from some misspelling or elsewhere in the code. Any Query that works on **SQL Server Management Studio** will work as a query in C# code.

First off, we will need a private method that gets our **SocMediaID** from the **SocialMediaName** in the combo box. To make it easier for us, this method is already created. It is called **getMediaID(string socialMedia)** and takes the social media name, and returns us the ID. Be sure to locate this method in the **BrowsePage.cs** and have a look at the code and comments to understand it. The video created alongside this script will help you understand the code too.

We will now be querying the databases **UserSession** table. Before we wrote code that helps the user to select a username and social media from the database easier. Now we will use that data to search through the **UserSession** table and display that data. Now navigate to the **BrowsePage.cs** and scroll until you find the method **buttonSearch_Click(...){...}** click event method. In here type in the following code:

```
//(1)
//give the username and socialMedia variables proper data
username = comboBoxUsername.SelectedItem.ToString();
socialMedia = comboBoxSocialMedia.SelectedItem.ToString();
//calls method to get ID based on it
socialMediaID = getMediaID(socialMedia);
.
.
.
//(2)
//Query to get the data we want from a table using username and ID
query = "SELECT * FROM UserSession WHERE UserName LIKE \'" + username
       + "\' AND SocMediaID LIKE \'" + socialMediaID +"'";
//The SQL select query, using above string
SQL.selectQuery(query);
.
.
.
//(3)
//Similar to the login page query, we will check that data has been returned
//Then loop through each row, printing the data to the list box
```

```
//Check that there is something to write into list box
                if (SQL.read.HasRows)
                {
                    listBoxDisplay.Items.Add("Results for: " + username);
                    //loop through each table row from the database
                    while (SQL.read.Read())
                    {
                        //get the data values and store them in variables
                           sessionID = SQL.read[0].ToString();
                           socialMediaID = SQL.read[1].ToString();
                           sessionTime = int.Parse(SQL.read[3].ToString());
                        //display each of the rows in a nice way
                           listBoxDisplay.Items.Add("Session ID: " + sessionID
                            + ",  Social Media: " + socialMediaID
                            + ",  Session Time: " + sessionTime);
                    }
                }
                else //where it doesnt have any successful searches
                {
                    listBoxDisplay.Items.Add("No User Sessions found.");
                 }
```

Hopefully you will notice how similar the pseudo for this piece of code is to the **LoginPage** queries. It is very similar, the main change you will need to do for your own programs is error checking and displaying the data nicer.

Example Program:

You will find an example program and video explaining the fundamentals of it on the Moodle page also. I recommend checking out this video and database as it will no doubt be similar to what you will have to do.

Extra Hints and Tips for the Deliverable:

Best thing to do when writing queries in **Visual Studio** is to check it on the **SQL Server Management Studio** before you put it as a **string** in C# code.

Comment your code as you go so you know what is happening in your code, especially for when you look back at it. Throughout the code given, there is comments that will help you to understand what the code is doing.

There is a lot of extra stuff you will want to do for your own deliverable compared to this intro guide. I have shown you the basics of what is possible using the **SQL class**. You will now need to take the initiative in learning the more complex **Select** statements. Usings these basics you will be able to make a more complex database application as long as you learn how to **query a database** effectively.

Using **String concatenation** to construct your SQL queries can enable **SQL Injection attacks**. See the lecture material to learn how to avoid this problem.

If you do not know something then search for it on the internet; this will help you become more self-driven in your learning and in some cases will even help you remember it better.