The prototype of the group project will consist of a preliminary design of the UI, how each person’s code will interface with other code, and the SQL for the assignment in a class.  Again make sure to read the final group project requirements to full understand what the UI will look like and how the program will work when completed.

For the “Search” folder, there should be a XAML file for the UI called “wndSearch.xaml”, another file named “clsSearchSQL” which contains all SQL statements for the Search Window, and the last file should be “clsSearchLogic” which will contain all business logic for the Search Window.

The user also needs to be able to search for invoices, which will be a choice from the menu.  On the search screen, all invoices should be displayed in a list (like a DataGrid) for the user to select.  The user may limit the invoices displayed by choosing an Invoice Number from a drop down, selecting an invoice date, or selecting the total charge from a drop-down box.  The total charges in the drop-down box should be the unique set of total charges sorted from smallest to largest.  When a limiting item is selected, the list should only reflect those invoices that match the criteria.  So, the user should be able to select a date and a total charge, and only invoices that match both will be displayed.  A clear selection button should reset the form to its initial state.  Once an invoice is selected the user will click a “Select Invoice” button, which will close the search form and open the selected invoice up for viewing on the main screen.  From there the user may choose to Edit or Delete the invoice.

For the “Main” folder, there should be a XAML file for the UI called “wndMain.xaml”, another file named “clsMainSQL” which contains all SQL statements for the Main Window, and the last file should be “clsMainLogic” which will contain all business logic for the Main Window.

The main window should allow the user to create new invoices, edit existing invoices, or delete existing invoices.  There should be just one window for all functionality of the main window.  So, the main window will NOT open other windows to add/edit/delete invoices.  It will also have a menu (at the top, use a menu control) that will have two functionalities.  The first will allow the user to update a def table that contains the items.  The next will be to open a search screen used to search for invoices.

If a new invoice is created the user may enter data pertaining to that invoice.  Since an auto-generated number is used in the database for the invoice number, when a user creates a new invoice, just display “TBD” for the Invoice Number.  An invoice date will also be assigned by the user.  Next different items will be entered by the user.  The items will be selected from a drop-down box and the cost for that item will be put into a read only textbox.  This will be the default cost of an item. Once the item is selected, the user can add the item.  As many items as needed should be able to be added.  All items entered should be displayed for viewing in a list (something like a DataGrid).  Items may be deleted from the list.  A running total of the cost of all items should be displayed as items are entered or deleted.

Once all the items are entered the user can save the invoice.  Once the Invoice is saved, query the max invoice number from the database, to display for the invoice number (for our project, this will work, since the last inserted invoice, will be the max).  This will lock the data in the invoice for viewing only.  From here the user may choose to Edit the Invoice or Delete the Invoice.

**GUI**

            All of the screens should be created with all controls needed to complete the requirements.  For instance, on the search screen, there should be 3 drop down boxes for selection, a DataGrid, and select and cancel buttons.  Once each screen has been created the flow of the program needs to be completed.  So, for example, on the main form, there should be a menu with the selection of “Search for Invoice” that when clicked should open the search window, then when the user clicks the “Select” or “Cancel” buttons the search window should close and the main form get focus.

**Interfaces**

            This part of the assignment is to put together a plan on how each screen will pass the data to the other screens.  This will be done by putting the appropriate comments in the sections of stubbed out code to explain how the data will be passed around.  This will get you thinking about how each screen will interface with the others.  So for example, on the search screen, behind the button click event for the “Select” button, there should be a detailed comment about how the selected InvoiceID will be passed back to the main form.  For example, if a property is set in the Search screen window with the selected Invoice ID, then the comment will explain how the variable is set and the Main screen may access this data via a property.

**SQL**

            This part of the assignment is to create a class that contains the main pieces of SQL used throughout the project.  This class will be nothing but methods that contain different statements of SQL.  Make sure to create SQL statements that will help in meeting all requirements that use the database.  So, SQL statements needed will be to select different types of data on each window, to update/insert/delete data on each form.  Use Microsoft Access to run the queries ahead of time to make sure the queries give you the expected data and work correctly.  Your SQL statements should be tested and working.  Below is an example of a class/method that should be used as a guide for your code.

    class clsSQL

    {

        /// <summary>

        /// This SQL gets all data on an invoice for a given InvoiceID.

        /// </summary>

        /// <param name="sInvoiceID">The InvoiceID for the invoice to retrieve all data.</param>

        /// <returns>All data for the given invoice.</returns>

        public string SelectInvoiceData(string sInvoiceID)

        {

            string sSQL = "SELECT \* FROM Invoices WHERE InvoiceNum = " + sInvoiceID;

            return sSQL;

        }

    }