**CIS 3325 Fall 2018**

**Project # 3A**

**Due Date: Beginning of class on Thursday, November 1, 2018**

**To be done by each student individually**

***Introduction:***

In each assigned project during the semester, you will develop a part of a fully functional business application for a company named SportsPro. SportsPro is a hypothetical software company that develops software for sports league. This application is designed for the technical support department of SportsPro. The purpose of the application is to track technical support service calls (referred to as *incidents*) in a database that also stores information about the company’s customers, software products, and technicians. When you complete the last project, you will have a fully functional application that follows the MVC design paradigm discussed in class.

***Requirements for Project # 3A:*** (***Required reading: chapters 6 and 7, Murach Textbook, Class Notes on using 3-tier architecture and ComboBox, Gaddis: Appendix B, page 722 Mastery: Class concepts and usage***).

***Naming Requirements***

1. Solution: Assignment3YourName e.g.Assignment2JohnDoe
2. User Interface Project: SportsProUserInterfaceLayer
3. Business Class Library Project: SportsProBLLClassLibrary
4. Database Class Library Project: SportsProDALClassLibrary
5. **Please follow ALL naming conventions discussed in class. (E.g. names of Variable/constant and UI controls are always lowercaseCamelCase; Method names, property names, class names (including Form classes in your solution explorer window), and module/function names are always UppercaseCamelCase.**

Please note that Project # 3 is divided into two parts, each with its own submission deadline. Since you will need your completed project # 3A to tackle Project # 3B, **PLEASE RETAIN A COPY OF YOUR SUBMITTED PROJECT # 3A SO THAT YOU ARE ABLE TO COMPLETE PROJECT # 3B, while your instructor is grading your submitted Project # 3A. Failure to do so will require you to start from scratch and force you to redo Project # 3A in order to complete project # 3B!**

1. **User Interface Requirements (SportsProUserInterfaceLayer):**

For project # 3A, you’ll modify the SportsPro application you created as a part of Projects # 2A and 2B. You are asked to modify the **FrmMain** menu structure and add “**Display Open Incidents By Technician**” option to the **Incidents** Menu. Selecting this option from the main menu will display the form **FrmOpenIncidentsByTechnician** shown in Figures 1 and 2 below:

Figure 1: FrmOpenIncidentsByTechnician at runtime. Selecting a Technician

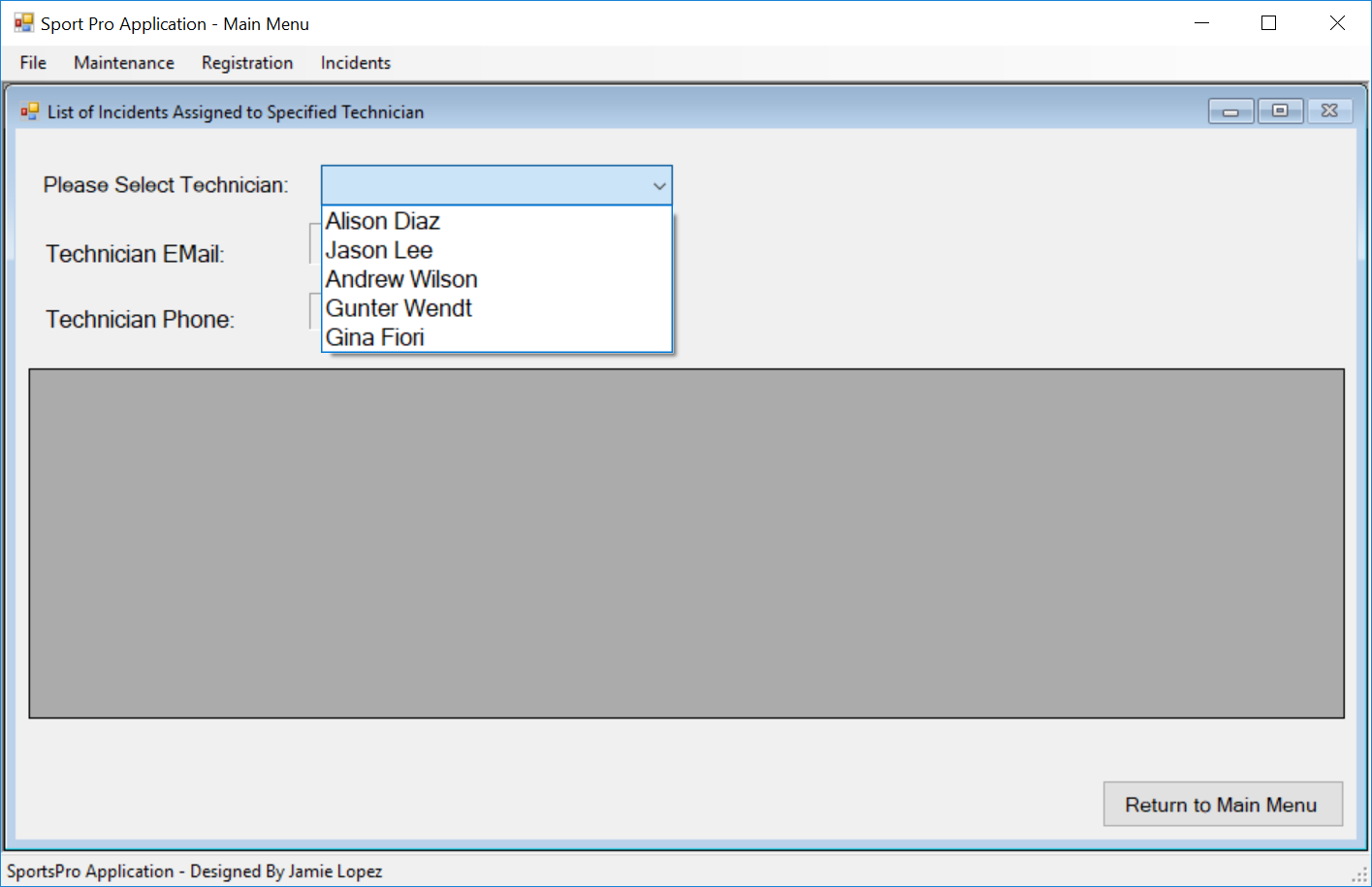
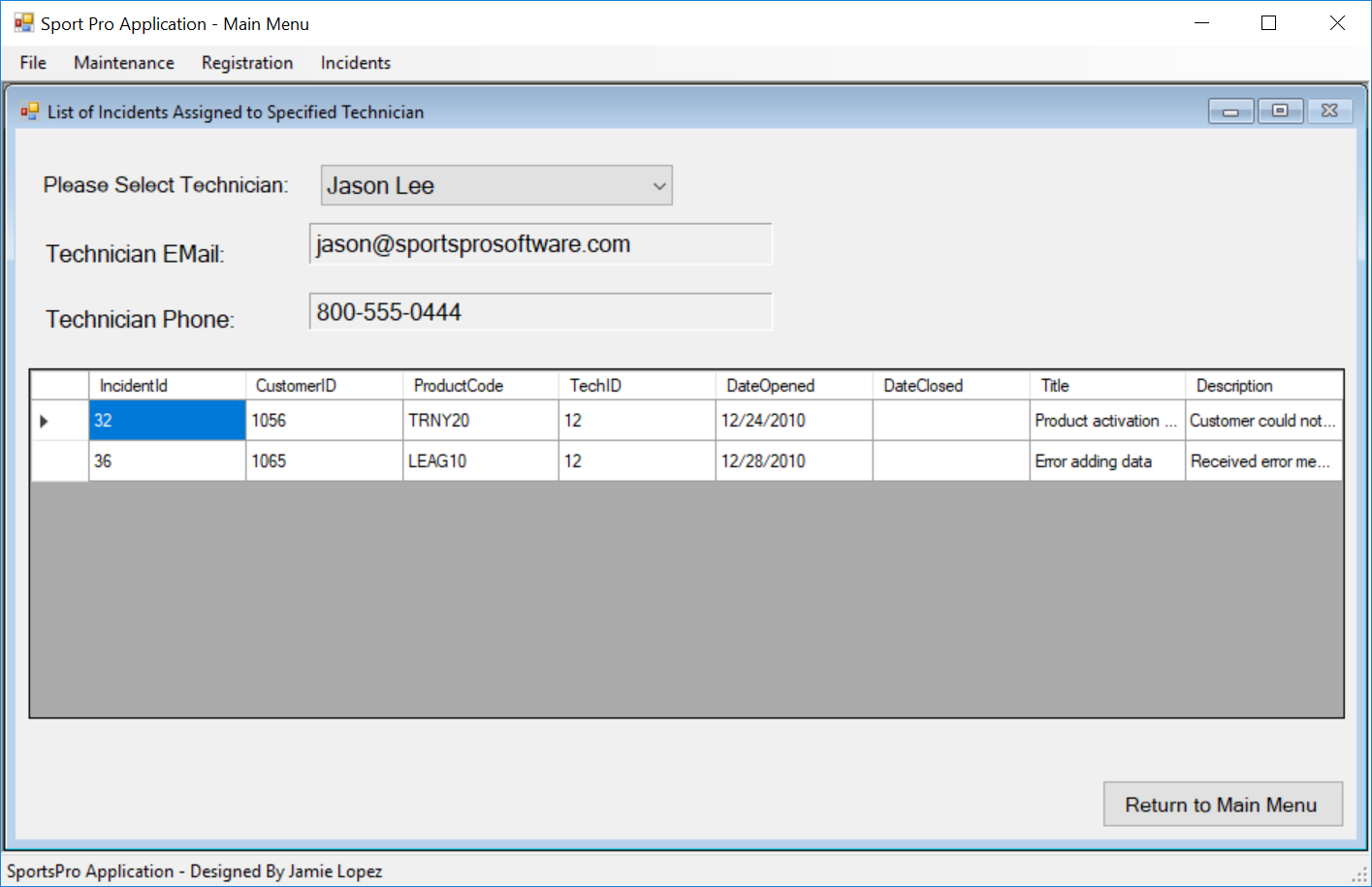
(Note: The text portion of the ComboBox is empty as technician is NOT preselected.)

Figure 2: FrmOpenIncidentsByTechnician at Runtime after a Technician is selected.



The top portion of the form will receive technician data from the Technician table in the TechSupport database based on the Technician class. The **read-only** ComboBox will be populated with technician names from the Technician table. The Email and Phone number fields will be populated after a technician is selected. The bottom portion, comprising of a DataGridView, will display **open** incidents for the technician selected in the ComboBox. An incident is considered an open incident when its DateClosed field contains a Null value. The DataGridView receives its data from the Incidents table based on a List<Incident> object built for the selected technician. The following design guidelines must be followed to populate **FrmOpenIncidentsByTechnician**.

1. **Please make sure that all forms, their caption bars, and all controls on these forms adhere to the design shown here. Pay attention to detail to ensure that the data are displayed as specified here. Proper names must be assigned to all controls and all naming conventions are strictly adhered to.**
2. **The DataGridView is anchored to the left, top, and right positions in the form.**
3. **The columns should auto size when the form size is altered at run time such that a horizontal scroll bar is never displayed. A vertical scroll bar may be displayed if the list of open incidents is longer than the DataGridView height.**
4. **The columns must be displayed in the order indicated in Figure 2. This, too a large extent, is controlled by the order in which the fields and properties are listed in the Incident class. If you are continuing from Project # 2B, you may want to modify the Incident class and rearrange the fields and properties to fit this sequence.**
5. **FrmOpenIncidentsByTechnician will be displayed as an MDI child form and should be anchored within the MDI parent on the left and top of the parent container. It should be docked to fill the entire MDI Parent.**
6. **A “Return to Main Menu” button must be placed anchored in the bottom right corner of FrmOpenIncidentsByTechnician. The button click should hide the form and return the user to the main form.**
7. **The cancel (X) button in the control box should be disabled so that while user can minimize/maximize the form, she/he cannot close the form. The only way the user can dismiss the form from view is by clicking the Return to Main Menu button.**
8. **Repeatedly selecting “Display Open Incidents By Technician” option should NOT result in a new instance of FrmOpenIncidentsByTechnician. Only one instance of this form should exist for the entire application.**

This UI will make use of additional business and data access classes and class methods to display the needed information. **These classes must be designed and coded in their respective class libraries as discussed in these specifications**. Please remember that you may have coded some of these classes for previous projects. Those should be reused here. You are essentially adding functionality to these classes and may have to modify some class methods for this project.

1. **Business Logic Class Library (Class Library Name: SportsProBLLClassLibrary)**

The following data access class will be used in this project.

Class Name Description

Incident A business class that represents an incident entity.

Technician A business class that represents a technician entity.

IncidentBLL A business logic class that includes methods for working with the Incidents data retrieved from the TechSupport database. The methods will specify requests for incidents data to the Data Access layer classes, which in turn will return the incidents data retrieved from the database.

TechnicianBLL A business logic class that includes methods for working with technician data retrieved from the Technician table in TechSupport database. The methods will specify requests for technician data to the Data Access layer classes, which in turn will return the technician data retrieved from the database.

1. **Data Access Layer Class Library (Class Library Name: SportsProDALClassLibrary)**

The following data access class will be used in this project.

Class Name Description

TechSupportDB A database class that contains a method that returns a SqlConnection object to the TechSupport database.

IncidentDAL Database access class that will contain methods to work with the Incidents table in the TechSupport database such as methods to retrieve all incidents and incidents for a specified technician; add a new incident or update an existing incident.

TechnicianDAL Database access class that will contain methods to work with the Technicians table in the TechSupport database such as methods to retrieve technician names; email and phone number for selected technician; add a new technician or update/Delete an existing technician.

Class diagrams shown in Figures 3 and 4 illustrate properties/methods that belong to each Business and Data Access class library. When coding business entity classes (Incident and Technician), please ensure that you list the private fields and respective property methods in the same order as the order in which the corresponding fields in the database tables are listed. This will impact the order in which columns are displayed in the DataGridView. Unfortunately, when displayed in class diagram format, these fields and properties are listed in alphabetic order.

Note: the addition of a new method GetOpenIncidentsByTechnician(int techID) in the IncidentBLL class.

**Figure 3: SportsProBLLClassLibrary Classes**

**Figure 4: SportsProDALClassLibrary Classes**



Please note that when a property can be assigned a null value (i.e. the property is nullable and it can contain a null value), its datatype includes a “?” symbol. For example, DateClosed in the Incident class has a Datatype DateTime? This indicates that DateClosed field may not be assigned a value and may contain a “null”. An incident with a null value for its DateClosed property is considered an “Open” incident in that the incident has yet to be resolved by tech support.

Additionally, if a method returns a value, its return value datatype is indicated in the method signature. For example, the GetConnection():SqlConnection method of the TechSupportDB class returns an object of type “SqlConnection.”

1. **Processing Tasks**

Operation

* The **FrmOpenIncidentsByTechnician** form should be displayed when the user chooses the Incidents🡪Display Open Incidents by Technician option from the main menu on FrmMain.
* When the **FrmOpenIncidentsByTechnician** is loaded, the form load event should populate the ComboBox with technician names. The technician names are assigned to be displayed in the display list portion while technician IDs will be assigned to the Value/Index portion of the ComboBox. So, when the user selects a technician name in the ComboBox, his/her techID is available to be extracted and passed on to appropriate method(s) in the Business Logic Layer classes. Upon form load, no technician should be pre-selected (text portion of the ComboBox is empty) Email and phone fields as well as the DataGridView should be empty as well – see Figure 1. .
* To display the open incidents for a selected technician, the user selects the technician name in the **read-only** ComboBox. Each time a technician is selected, the selected technician’s detail (email and phone) and open incidents assigned to the selected technician are retrieved and displayed in the respective Label fields and DataGridView.
* Clicking the “Return to Main Menu” button will hide the current form and return the user to the FrmMain.
* The child form must be disposed off appropriately before the user exits the application.
* To populate the ComboBox with technician names and IDs:
* Invoke the GetTechnicianNames method of the TechnicianBLL class in the form load event.
* This method will in turn invoke the RetrieveTechnicianNames method of the TechnicianDAL class.
* This DAL class method will retrieve technician name and ID data from the database via SQL and return these to the BLL class method that invoked it.
* The BLL class method will return the data it received to the form to populate the ComboBox.
* To populate technician email, phone and open incidents assigned to the selected technician:
* SelectionChangeCommitted event is triggered when the user-selected technician name is displayed in the text portion of ComboBox signaling that the selection of a technician is completed.
* The code to this event will first extract the ID of the selected technician (Hint. Think SelectedValue property of the ComboBox). Ensure that the extracted value is converted to an integer value since techID is declared as an integer value.
* Invoke the GetTechnicianDetails method of the TechnicianBLL class and GetOpenIncidentsByTechnician method of the IncidentBLL class, passing in the extracted and converted techID value as an argument in each case. .
* These methods, in turn, will invoke their respective counterpart methods in the TechnicianDAL and IncidentDAL classes, passing the techID it received.
* The TechnicianDAL and IncidentDAL class methods will configure the necessary SQL to retrieve the data requested. The methods will return the retrieved data in the form defined by the respective class methods to the BLL class methods. To get open incidents for the selected technician, use a command object with a parameter and a select statement that retrieves the requested data from the Incidents table. Similarly, to get technician details, use a command object with a parameter and a select statement that retrieves the requested data from the Technician table. The techID value being passed as an argument will be assigned to the parameter before executing the SQL query.
* Each of the BLL class methods will further package the data it receives as specified by the class method declaration for transport back to the form. For example, GetOpenIncidentsByTechnician method returns a List<Incident> object while GetTechnicianNames method returns a DataTable and GetTechnicianDetails method returns a Technician object.

Specifications

* The best way to accomplish this project is to start with **a fully working solution** from Project # 2B. Modify the solution to comply with specifications provided here. For example, if your MDI child was not anchored correctly within FrmMain, correct that to meet the anchoring position described in these specifications.
* Review the class libraries as declared in Project # 2B and add classes that you need for project # 3A if they do not currently exist in the library. Also review class methods contained in each business class (including the ones you used in previous projects) to see if they need to be modified to meet specifications prescribed in this document.
* For Connection string:
* Database server name: CISSQL.MATRIX.TXSTATE.EDU**\\**CIS3325SUMMR2018 (Note the two”\”)
* Database name: TechSupport
* Authentication: Windows authentication.
* Don’t forget to include code to handle nulls for fields that are nullable – TechID and DateClosed.
* Don’t forget to name your projects/solution/classes/controls/forms/variables/class methods using the naming conventions discussed in class and prescribed here.
* Finally, you have seen project evaluation forms for two projects so far. The evaluation form for Project #3 and others that follow will be very similar and looking for similar level of detail. Use those as guidelines (particularly the Yes/No segment at the top of the evaluation form) when submitting your project. Going forward, a single check mark in the NO column would mean a grade of zero. Therefore, make sure that your submission complies with this section. **Above all, DO NOT submit if your project does not run for ANY reason. If you are concerned about not losing points on your grade, my recommendation is to start early and seek assistance early enough to ensure you have a fully working solution at least one day prior to the submission deadline. This will give you time to put your submission package together correctly, go through these specifications to ensure your submission meets each specification requirement, and ensure you are submitting a working solution.**

***Submission Requirements:***

1. The assignment is to be completed individually by each student. This is a requirement. Collaborative submissions and work will be treated as academic dishonesty and handled accordingly.
2. Your **complete and working Solution** should be available at the ***root level of your USB drive.***
3. Submit the USB drive with your completed and running solution as well as the following printouts **arranged per sequence below**:
   1. A cover page with your name, project name, and submission date.
   2. A screenshot of the open incidents by technician form at run time showing a list of all open incidents for a selected technician.
   3. Printout of the code page for FrmOpenIncidentsByTechnician. (screenshot is not acceptable)
   4. Printouts in **Full** of **all** business entity, BLL and DAL classes. Please arrange them in the order of being used going from the form to database. (again, screenshots are NOT acceptable):
      1. Technician Class in the business class library.
      2. Incident class in the business class library
      3. All methods in TechnicanBLL class
      4. All methods in IncidentBLL class
      5. All methods in TechnicianDAL class
      6. All methods in IncidentDAL class
      7. TechSupportDB class.
   5. **Staple the printouts in the above order and hook in your USB drive with a paper clip or a string into the staple.**
4. Place the USB drive and your printouts on the instructor’s desk before the start of class on the due date. Once the instructor starts class, the assignment is considered late and will not be accepted for grading.

***Academic Dishonesty:***

1. **This is an individual assignment. Each individual is to do his/her own work.**
2. **Any collusion or sharing of work among individuals will be considered an academic dishonesty and will be handled in accordance with the Texas State’s Honor Code.**
3. **This instructor is serious about enforcing this policy to the fullest extent possible.**