Tristan Cole McDonald

Student Number: 20116247

PATHWAY: CLDV6211

Lecturer: Isaac Leshaba

05 July 2021

POE

Table of Contents

Table of Figures

1. Specifications

1.1. Database Specifications for Domingo Roof Works

The database for Domingo Roof Works was created in Microsoft SQL Server Management Studio using the SQL language.

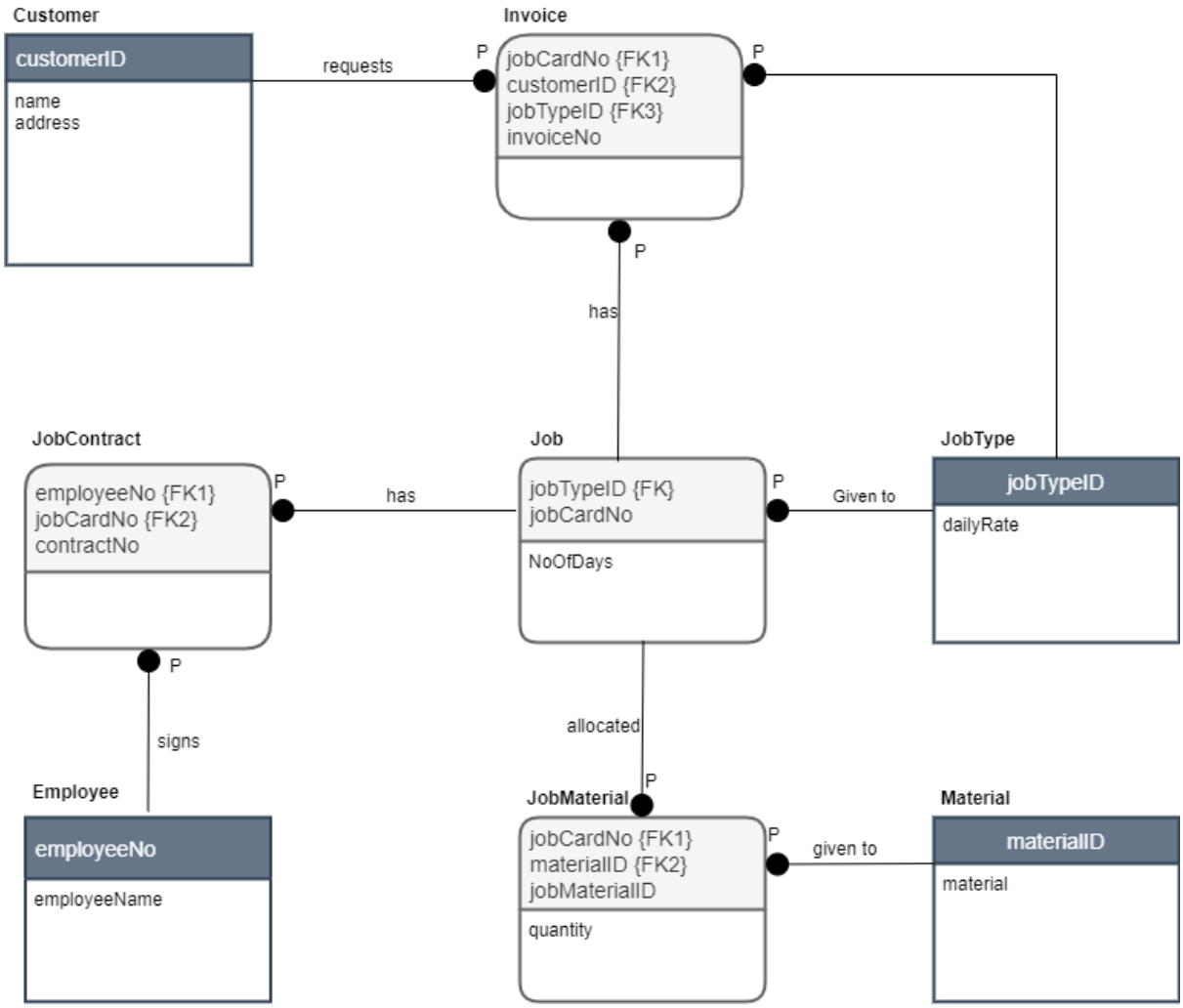


Figure 1: Domingo Roof Works ERD

Figures 2, 3, and 4 below illustrate how the 8 tables(entities) for the database were created, with reference to (Chao, 2014) and (Tik, 2021). Each table has a primary key and the tables (child entities) which contain foreign keys (primary keys from parent entities) are illustrated in figure 1 (above) with rounded corners.

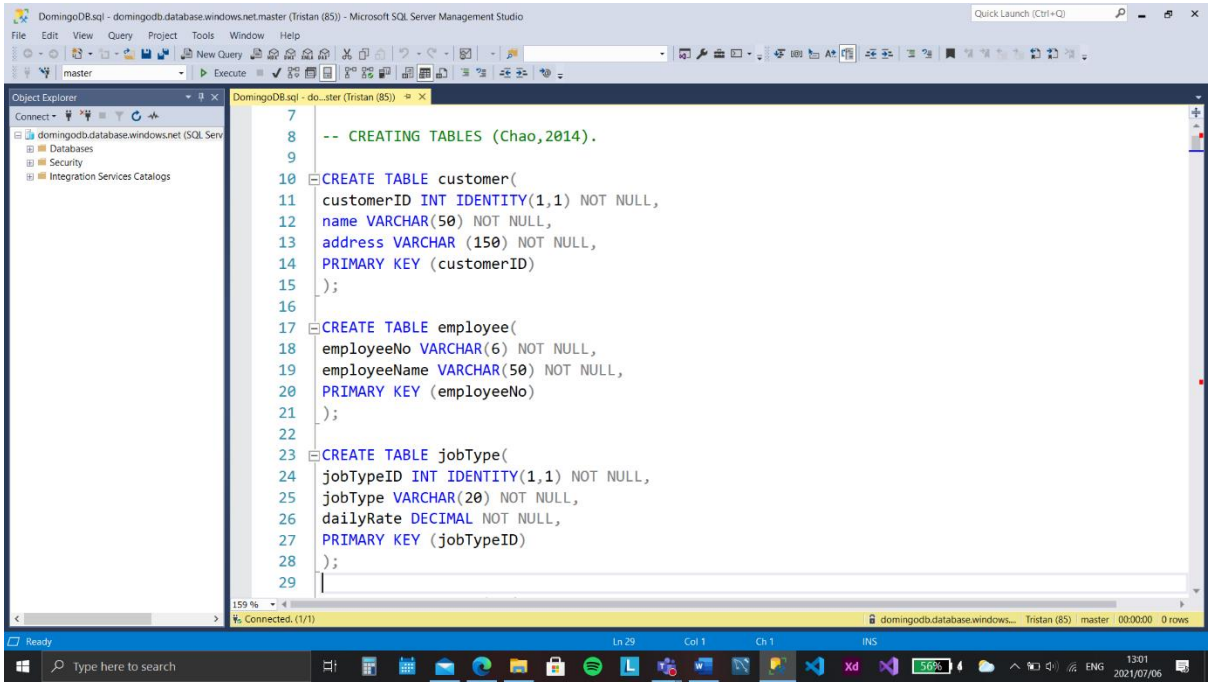


Figure 2: Table creation.

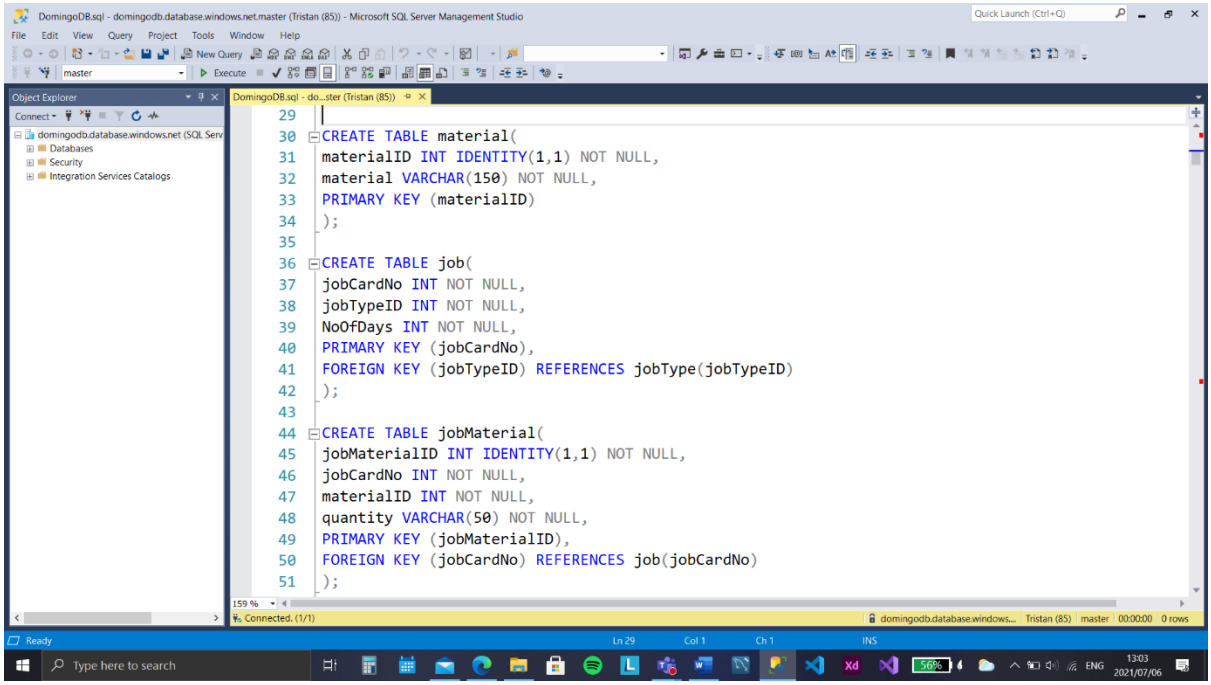


Figure 3: Table creation 2.

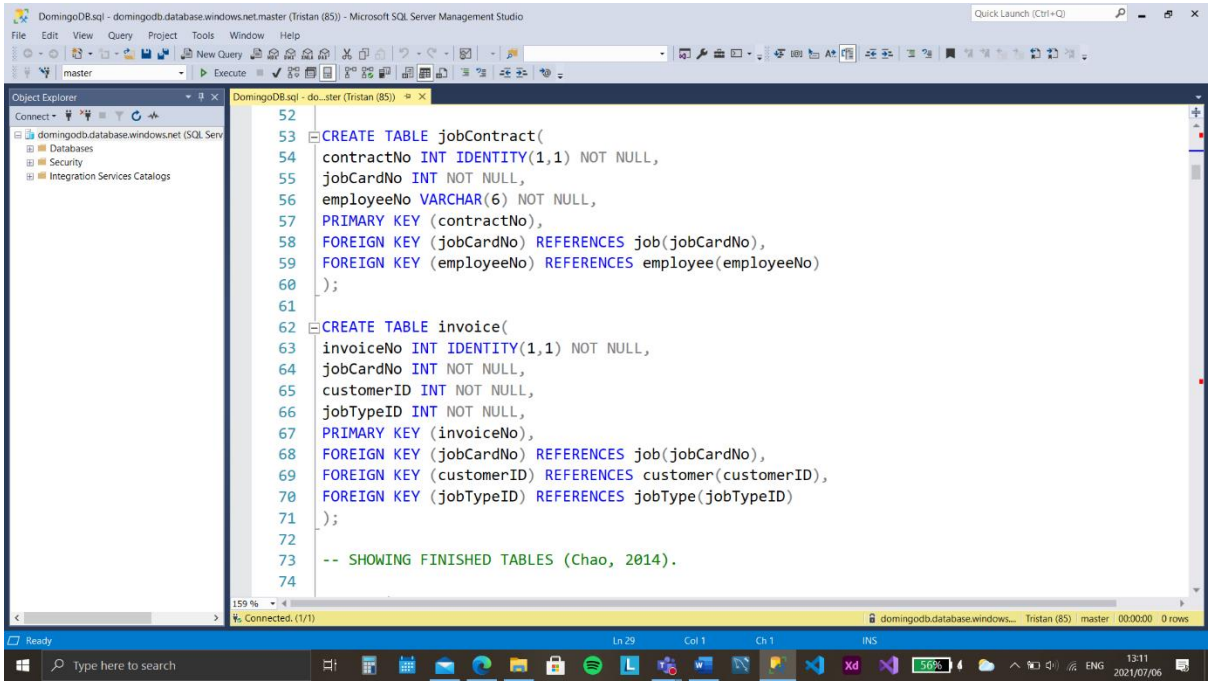


Figure 4: Table creation 3.

Figures 5 to 8 below illustrate how the predetermined data (given by Domingo Roof Works) was inserted into the database tables that were created above.

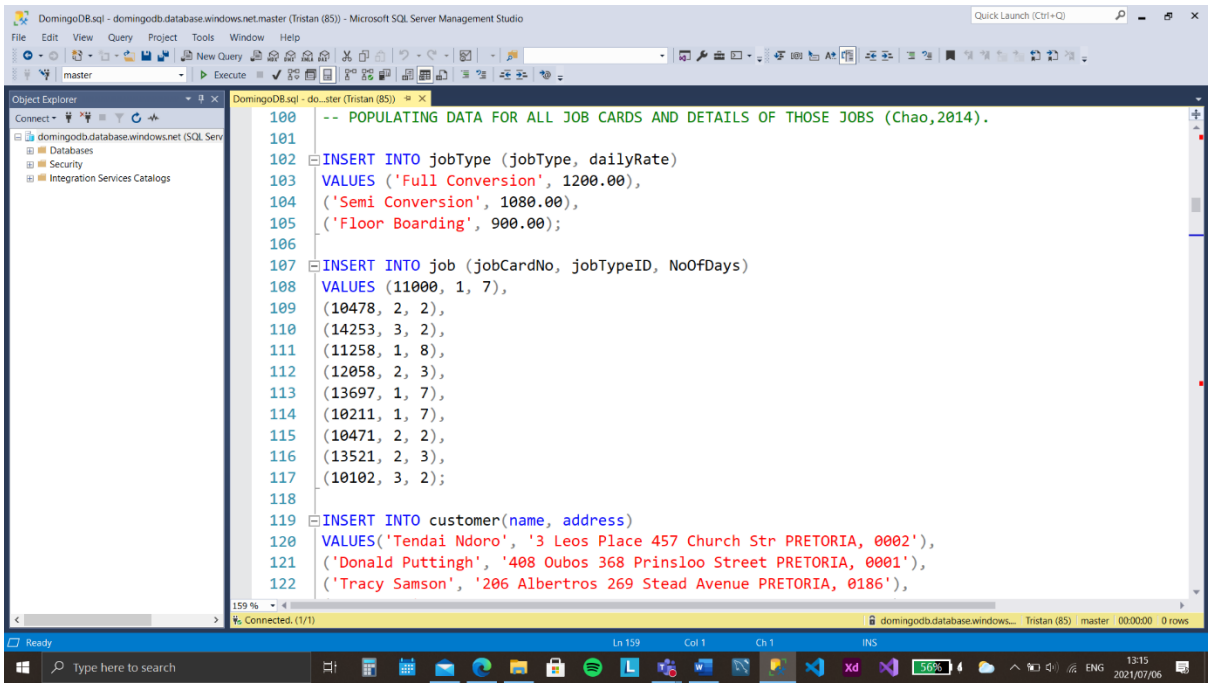


Figure 5: Inserting data into tables.

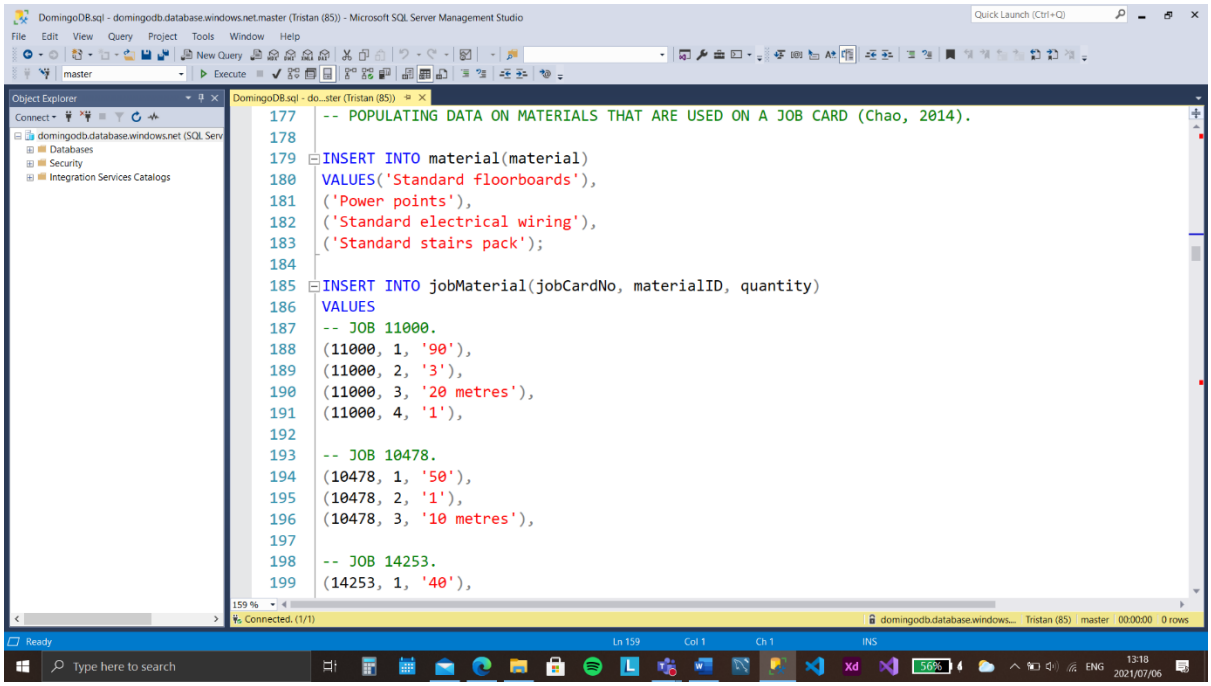


Figure 6: Inserting data into tables 2

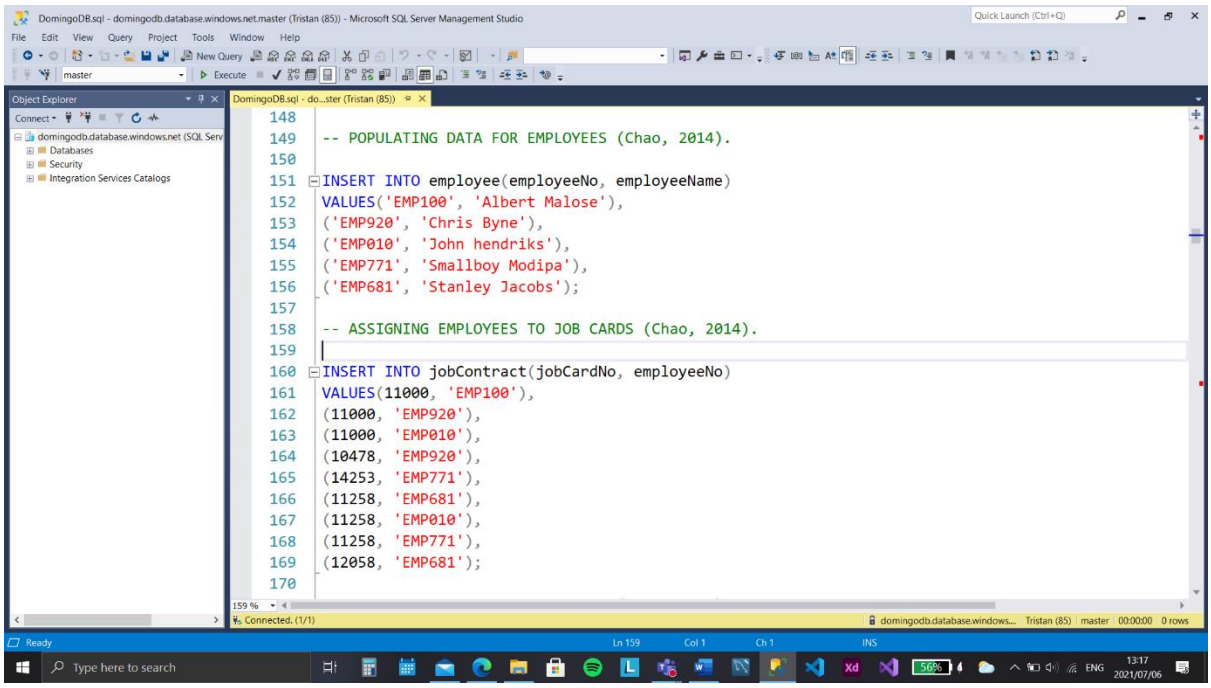


Figure 7: Inserting data into tables 3.

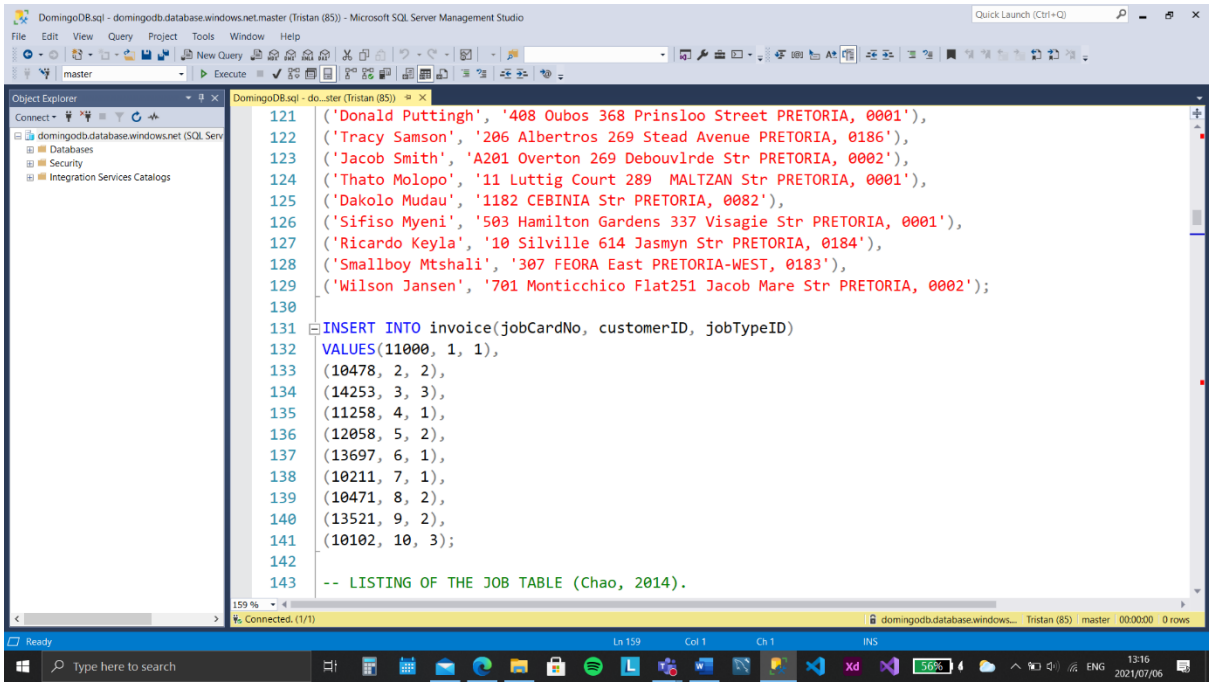


Figure 8: Inserting data into tables 4.

With reference to (Chao, 2014), (Tik, 2021), and (w3schools.com, 2021), various Stored Procedures were created for the Domingo Roof Works Web Application to Insert, read data, update data, and delete data to and from the Domingo Roof Works database. The scripts for these Stored Procedures are included in the POE folder and the document from task 2 (included in the Task 2 Revised folder) provides an in-depth illustration of how this database was created and manipulated.

1.2. Web App Specifications for Domingo Roof Works

The Web Application for Domingo Roof Works was created in Visual Studio Using the ASP.NET Core Model View Controller (MVC) Framework. This Web App is hosted in the Microsoft Azure cloud and has been connected to the database which is also hosted in the Microsoft Azure cloud.

With reference to (Andrew Troelsen & Philip Japikse, 2017), (IAmTimCorey, 2018), and (Tik, 2021), a Model for each required table(entity) in the database has been created which serves as a template for the data transferred between the database tables and the Web Application. An example of a model created for an employee is in Figure 9 shown below. Note that all other models follow this or a similar structure.

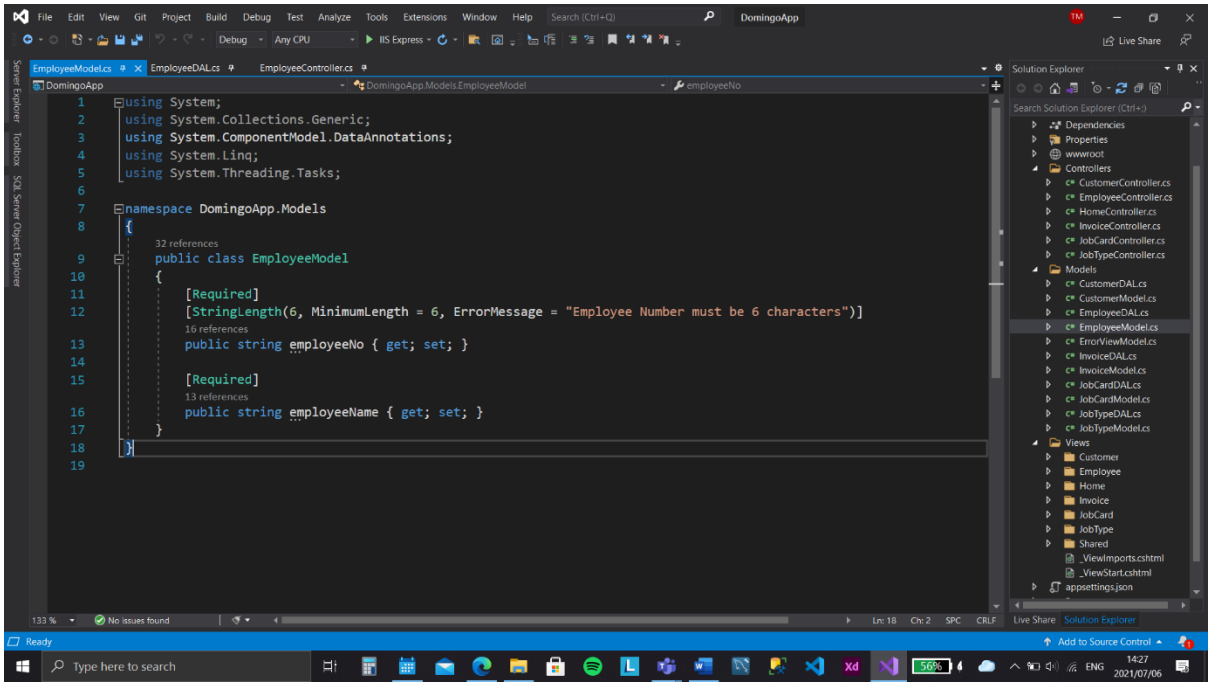
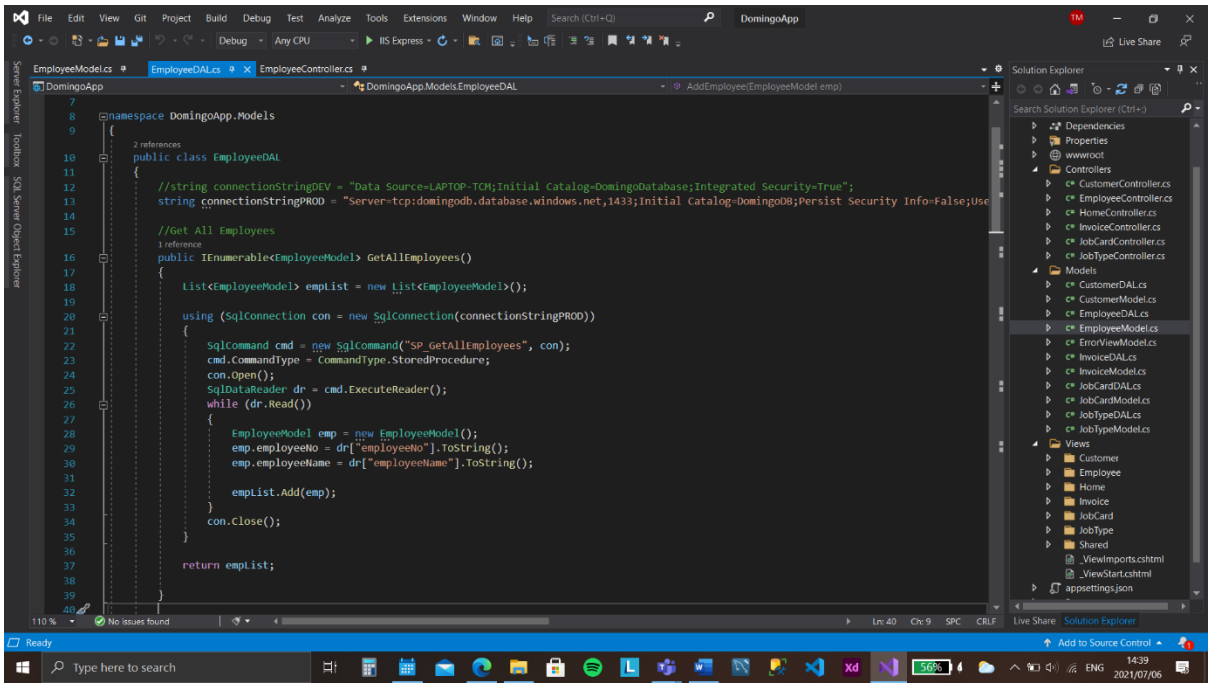


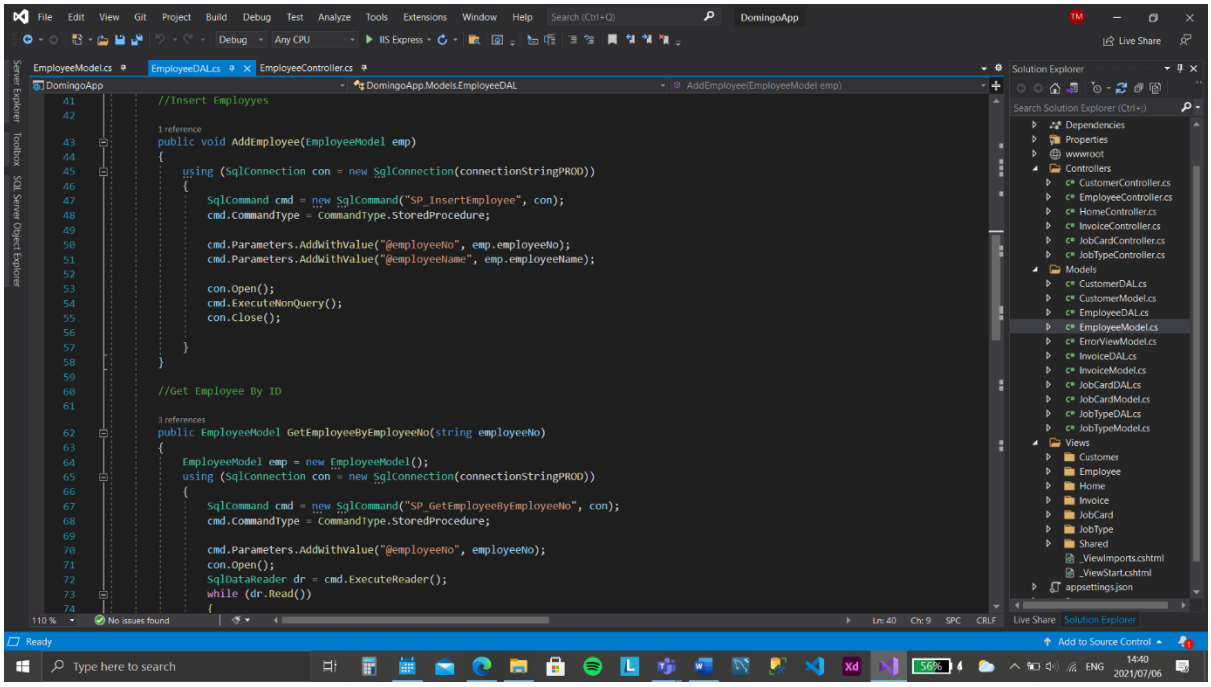
Figure 9: Employee model example.

Utilizing information from (Tik, 2021) and (Andrew Troelsen & Philip Japikse, 2017), a Data Access Layer (DAL) for each table(Model) has also been created which allows the Web Application to Insert, Read, Update, or Delete data to or from the Domingo database. For the application to perform the Create, Read, Update, and Delete (CRUD) functions with the database each function required in the POE has a stored procedure created for it in the database (See Task 2 Revision Folder for the Stored Procedures). An example of the DAL for an employee is provided in Figures 10 to 13 below. Note that all other DAL's follow this or a similar structure.



```
namespace DomingoApp.Models
{
    2 references
    public class EmployeeDAL
    {
        //string connectionStringDEV = "Data Source=LAPTOP-TCM;Initial Catalog=DomingoDatabase;Integrated Security=True";
        string connectionStringPROD = "Server=tcp:domingodb.database.windows.net,1433;Initial Catalog=DomingoDB;Persist Security Info=False;Use

        //Get All Employees
        1 reference
        public IEnumerable<EmployeeModel> GetAllEmployees()
        {
            List<EmployeeModel> empList = new List<EmployeeModel>();

            using (SqlConnection con = new SqlConnection(connectionStringPROD))
            {
                SqlCommand cmd = new SqlCommand("SP_GetAllEmployees", con);
                cmd.CommandType = CommandType.StoredProcedure;
                con.Open();
                SqlDataReader dr = cmd.ExecuteReader();
                while (dr.Read())
                {
                    EmployeeModel emp = new EmployeeModel();
                    emp.employeeNo = dr["employeeNo"].ToString();
                    emp.employeeName = dr["employeeName"].ToString();

                    empList.Add(emp);
                }
                con.Close();
            }

            return empList;

        }
```

Figure 10: EmployeeDAL example.



```
        //Insert Employyes

        1 reference
        public void AddEmployee(EmployeeModel emp)
        {
            using (SqlConnection con = new SqlConnection(connectionStringPROD))
            {
                SqlCommand cmd = new SqlCommand("SP_InsertEmployee", con);
                cmd.CommandType = CommandType.StoredProcedure;

                cmd.Parameters.AddWithValue("@employeeNo", emp.employeeNo);
                cmd.Parameters.AddWithValue("@employeeName", emp.employeeName);

                con.Open();
                cmd.ExecuteNonQuery();
                con.Close();

            }
        }

        //Get Employee By ID

        3 references
        public EmployeeModel GetEmployeeByEmployeeNo(string employeeNo)
        {
            EmployeeModel emp = new EmployeeModel();
            using (SqlConnection con = new SqlConnection(connectionStringPROD))
            {
                SqlCommand cmd = new SqlCommand("SP_GetEmployeeByEmployeeNo", con);
                cmd.CommandType = CommandType.StoredProcedure;

                cmd.Parameters.AddWithValue("@employeeNo", employeeNo);
                con.Open();
                SqlDataReader dr = cmd.ExecuteReader();
                while (dr.Read())
```

Figure 11: EmployeeDAL example 2.

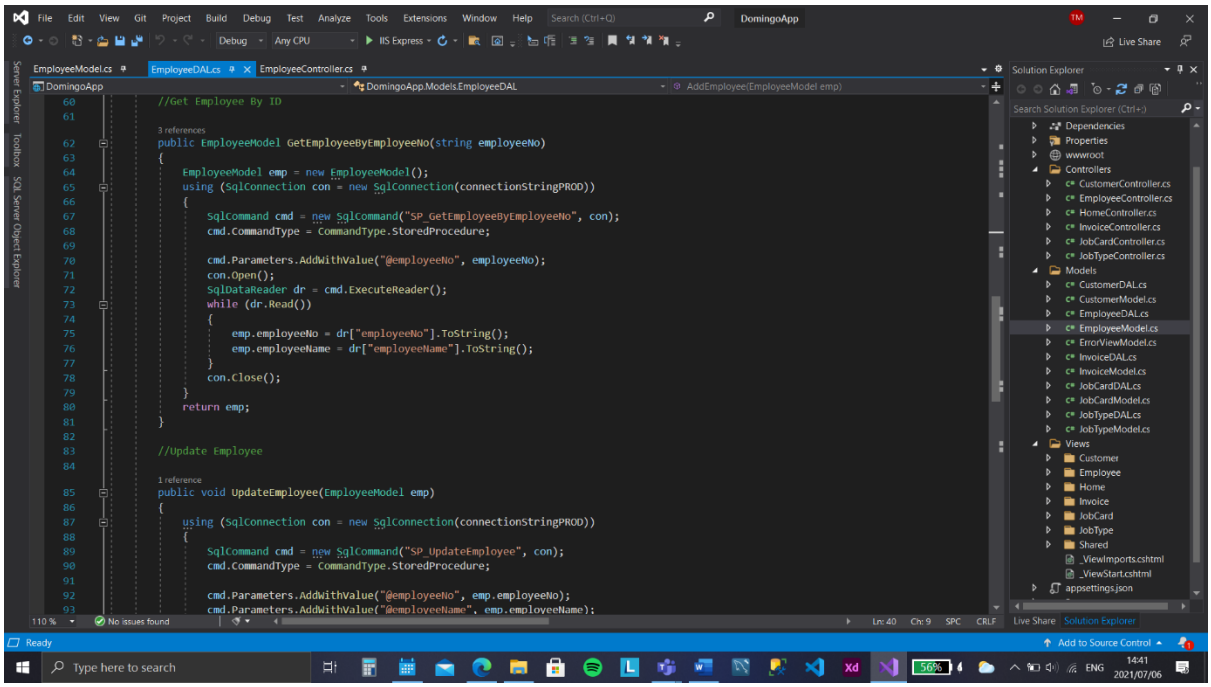


Figure 12: EmployeeDAL example 3.

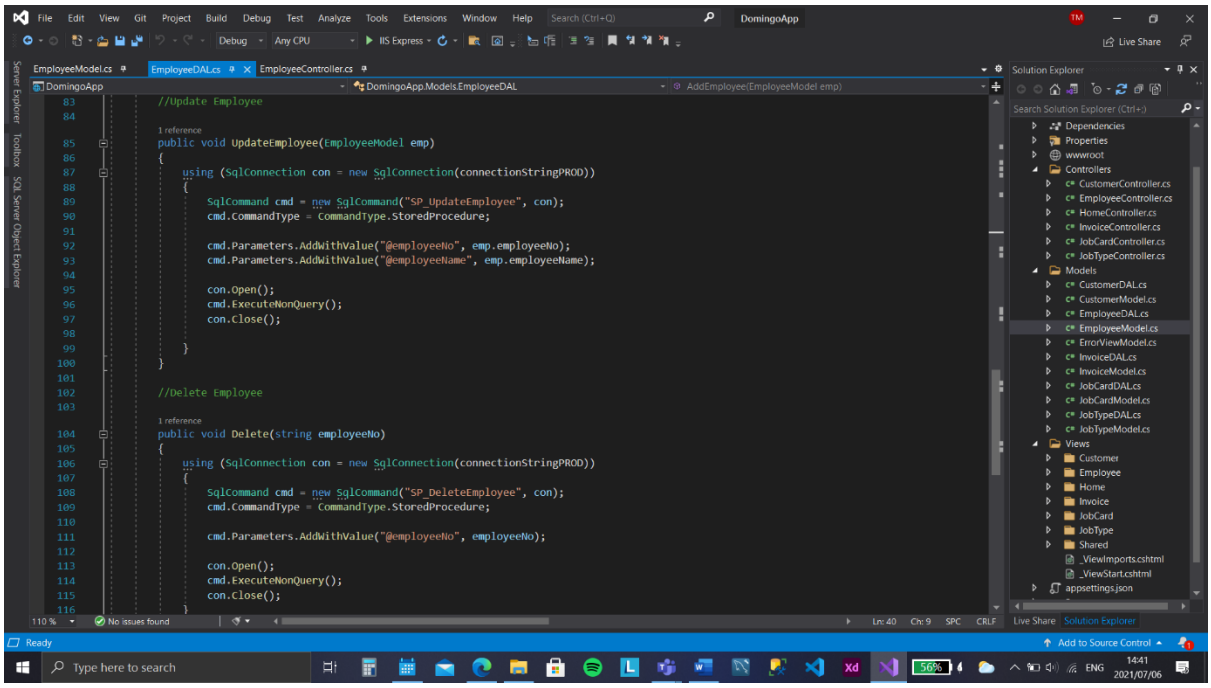


Figure 13: EmployeeDAL example 4.

Using information provided by (Tik, 2021) and (Andrew Troelsen & Philip Japikse, 2017), a controller for each table(Model) has been created which handles the requests sent by the user to the web application. With reference to (tutorialspoint.com, 2021), the controllers for each of the functions perform various duties requested by the user such as returning a details page, displaying error pages when necessary, and returning other pages(views) associated with the CRUD operations when requested. The controllers also work in conjunction with the DAL's to submit or receive data to or from the database and display the outcomes. An example of the controller for an employee is provided in Figures 14 to 18 below. Note that all other controller's follow this or a similar structure.

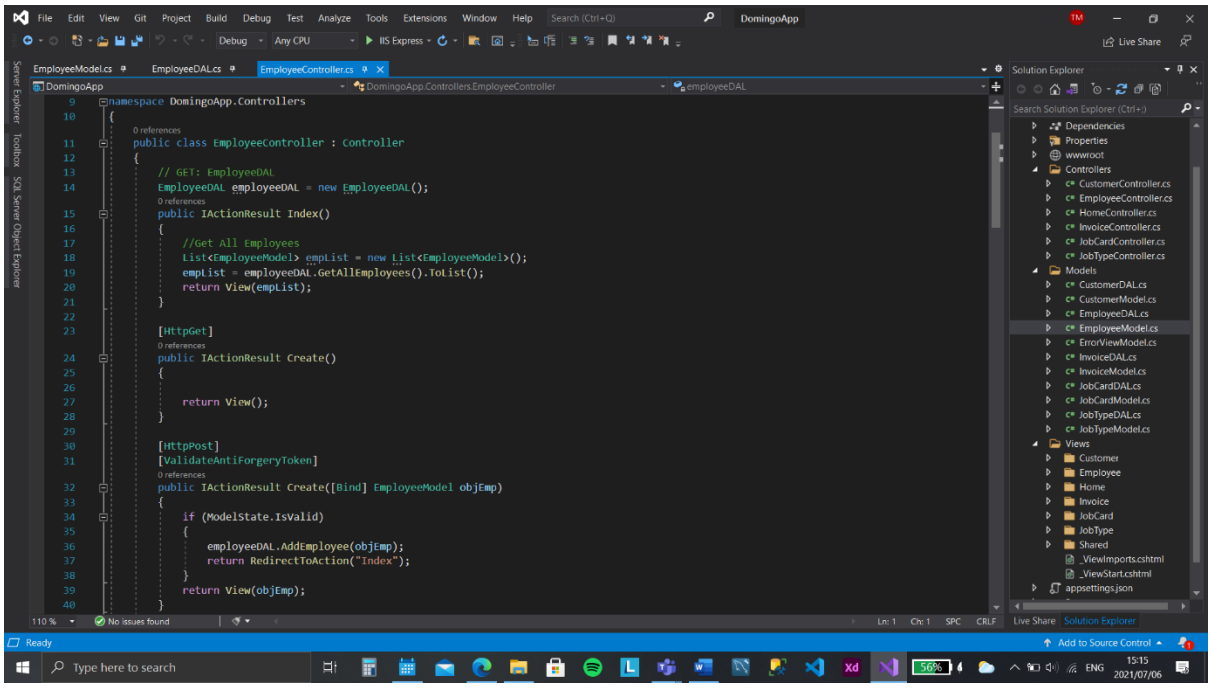


Figure 14: Employee controller example.

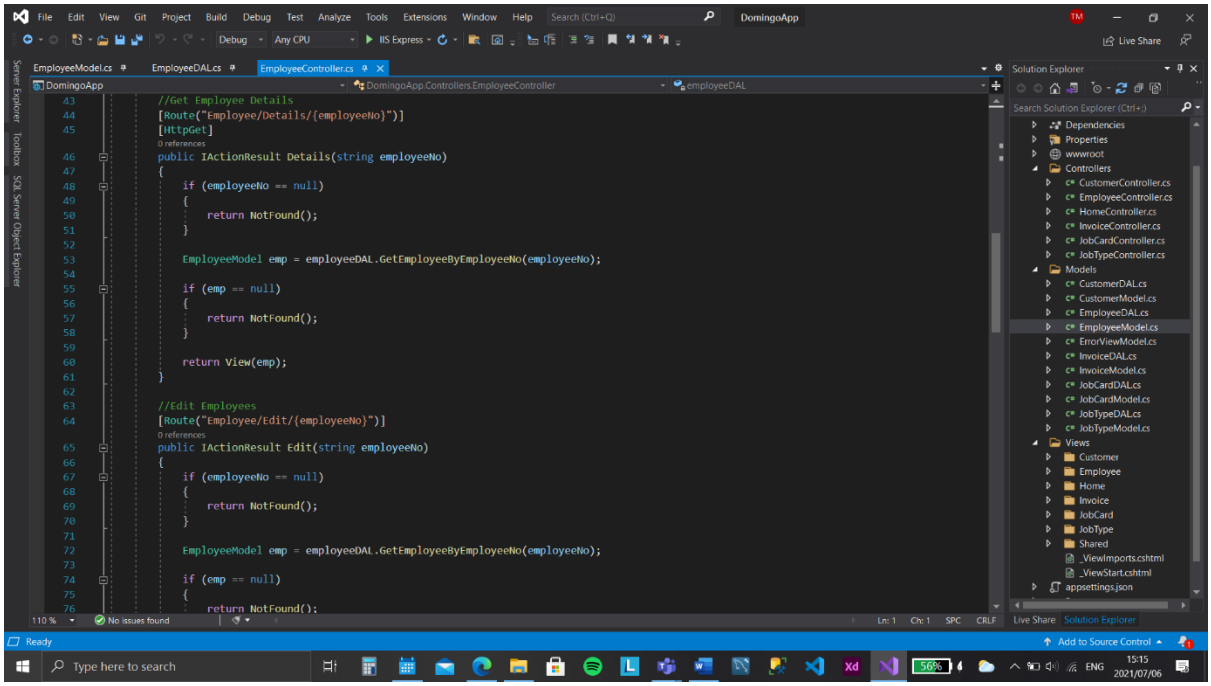


Figure 15: Employee controller example 2.

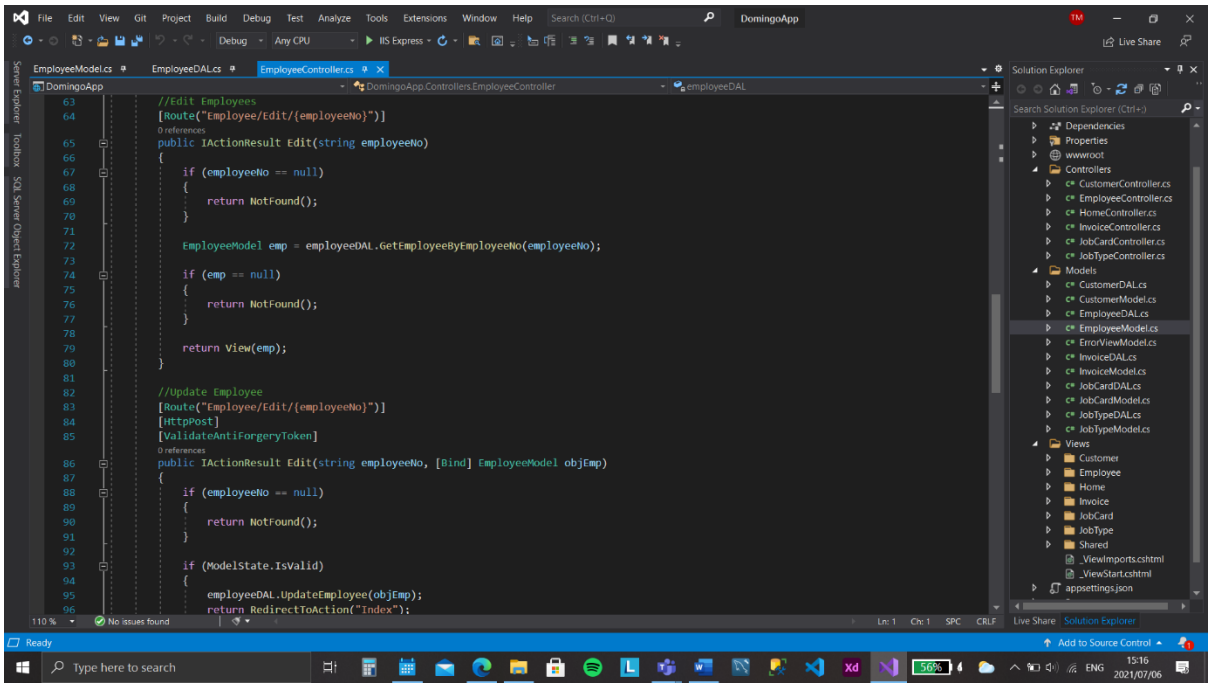


Figure 16: Employee controller example 3.

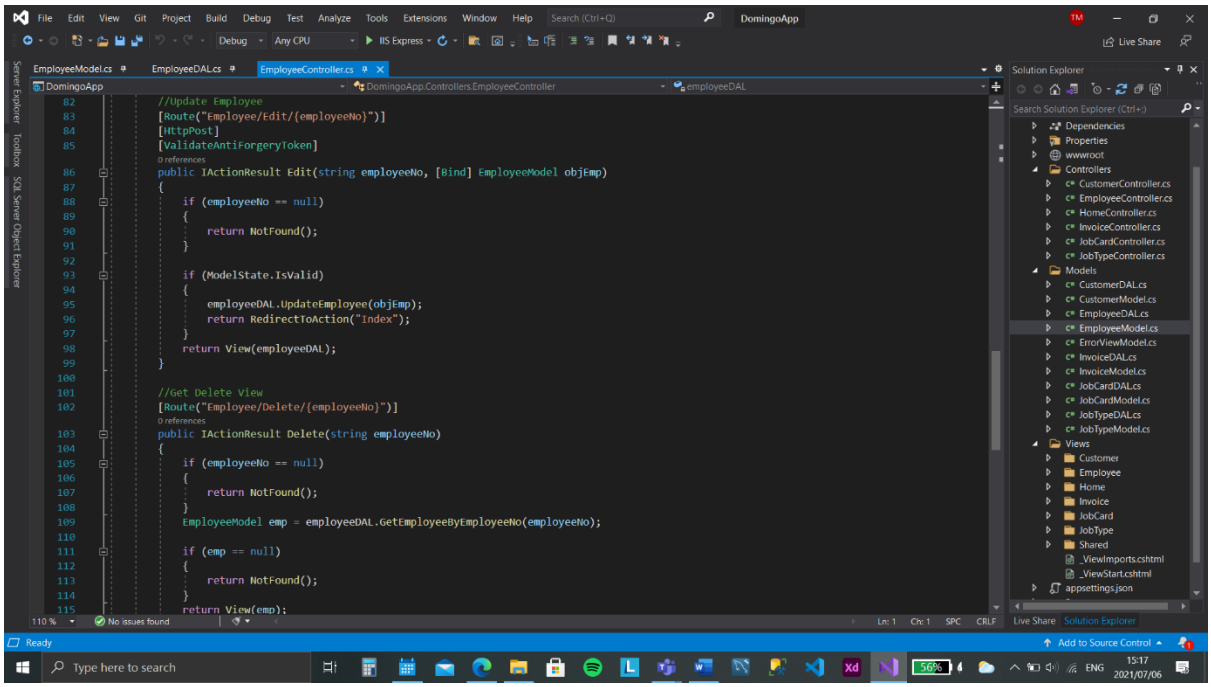


Figure 17: Employee controller example 4.

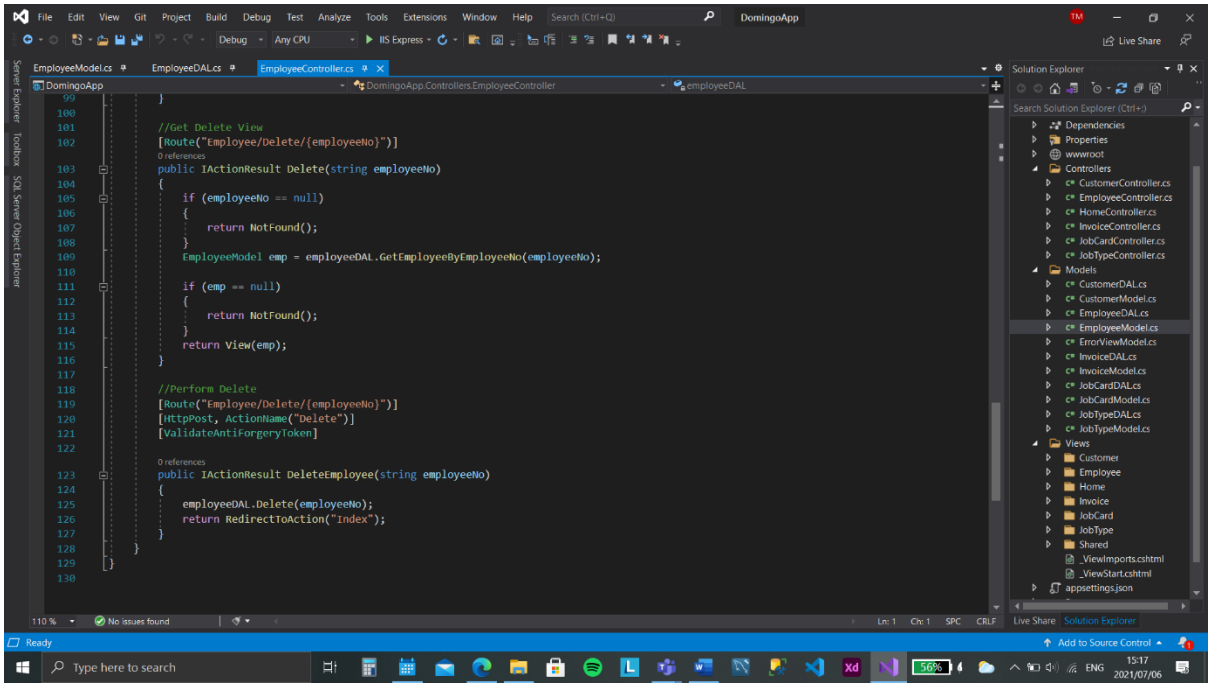


Figure 18: Employee controller example 5.

Finally, with reference to (Tik, 2021) and (IAmTimCorey, 2018), a few views were created for each of the CRUD operations and error pages required in each of the tables(Models). The views utilize a mixture of HTML, CSS, JavaScript, and C# which is called Razor pages (IAmTimCorey, 2018). The front-end styling/presentation was created using these Razor page views, and each page provided a functionality for the user. An example of 1 of the 4 views created for the employee table(Model) is shown in Figure 19 below. Note that all other views follow this or a similar structure.

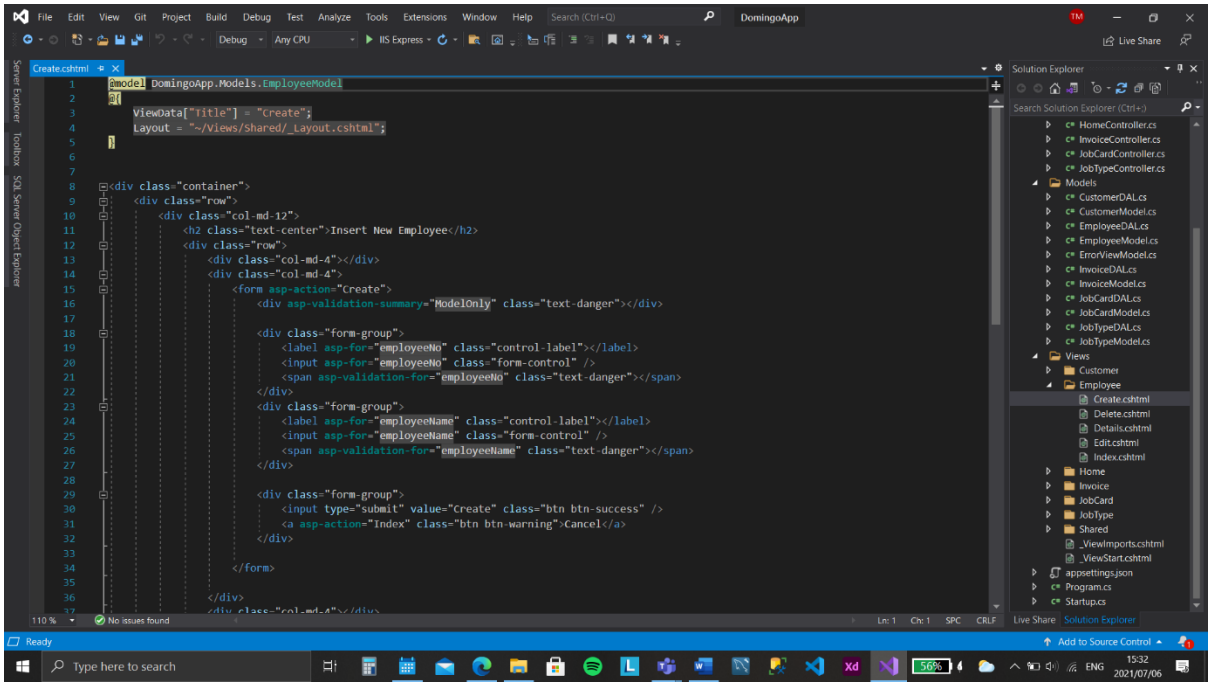


Figure 19: Employee create view example.

1.3. How a user can navigate around and perform the different functions in the Domingo Roof Works Web Application.

Upon arrival to the Domingo Roof Works website the user will be greeted with this home page in Figure 20 below.

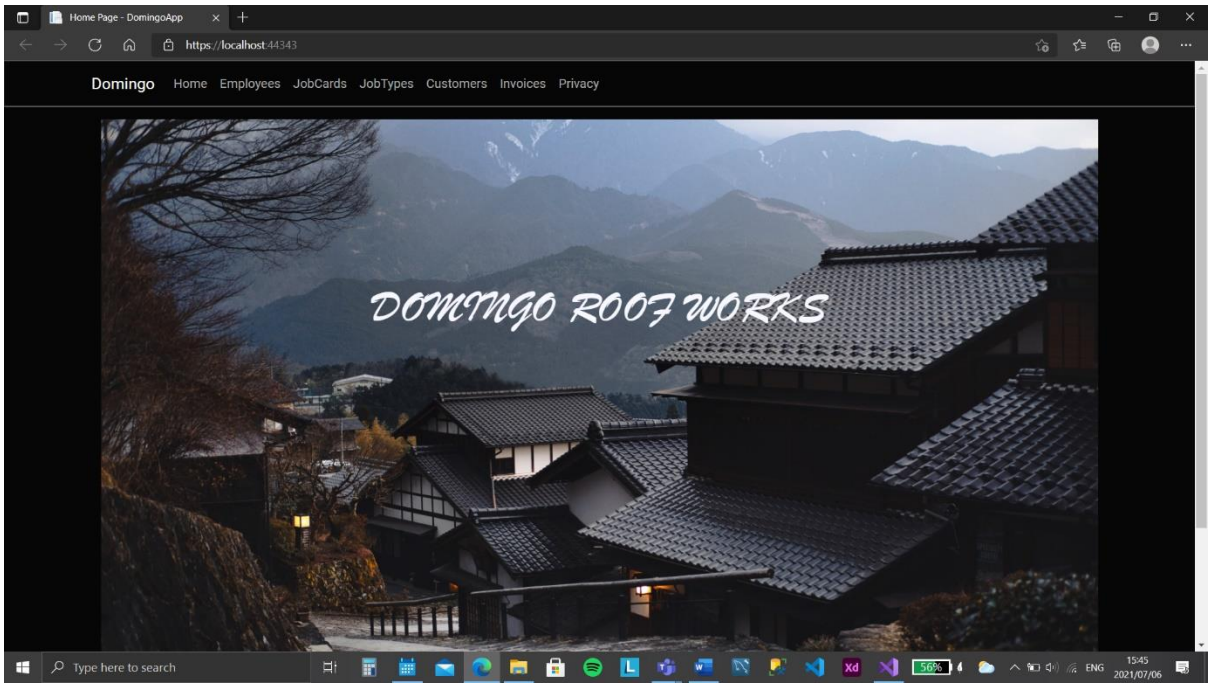


Figure 20: Domingo home page.

Referencing (Tik, 2021) and (IAmTimCorey, 2018), the user can click on any of the titles displayed in the top navigation bar to navigate to the various pages(functions) that they would like to access. For this example, the employees title(page) in the navigation bar will be utilized. See figure 21 below for the displayed page after the employees title(page) in the navigation bar was clicked.

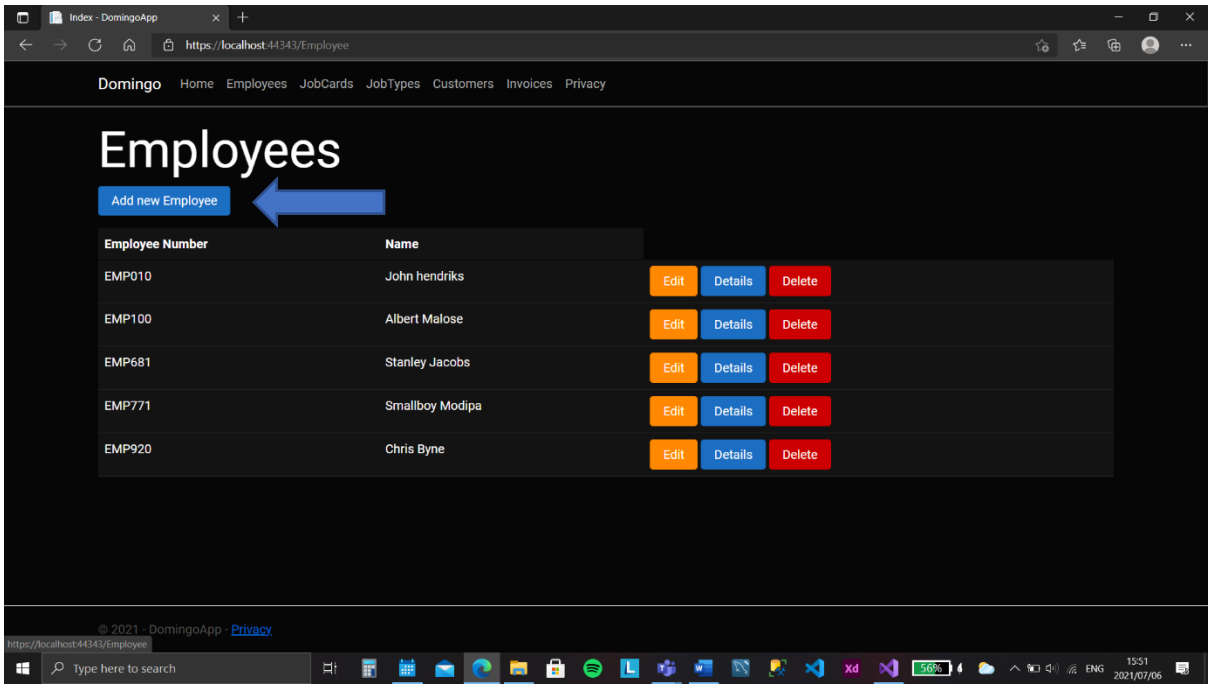


Figure 21: Employees page and add button.

Once the page is displayed the user is provided the necessary information for the operations that they would like to perform within the specific page such as Inserting, Reading, Updating, and Deleting data. In this example with the employees a new employee will be created by clicking the Add New Employee button. The user is then prompted to enter the employee's details as shown in Figure 22 below. For this example, an employee with an employeeNo of EMP123 and a name of Jeff Smith will be created.

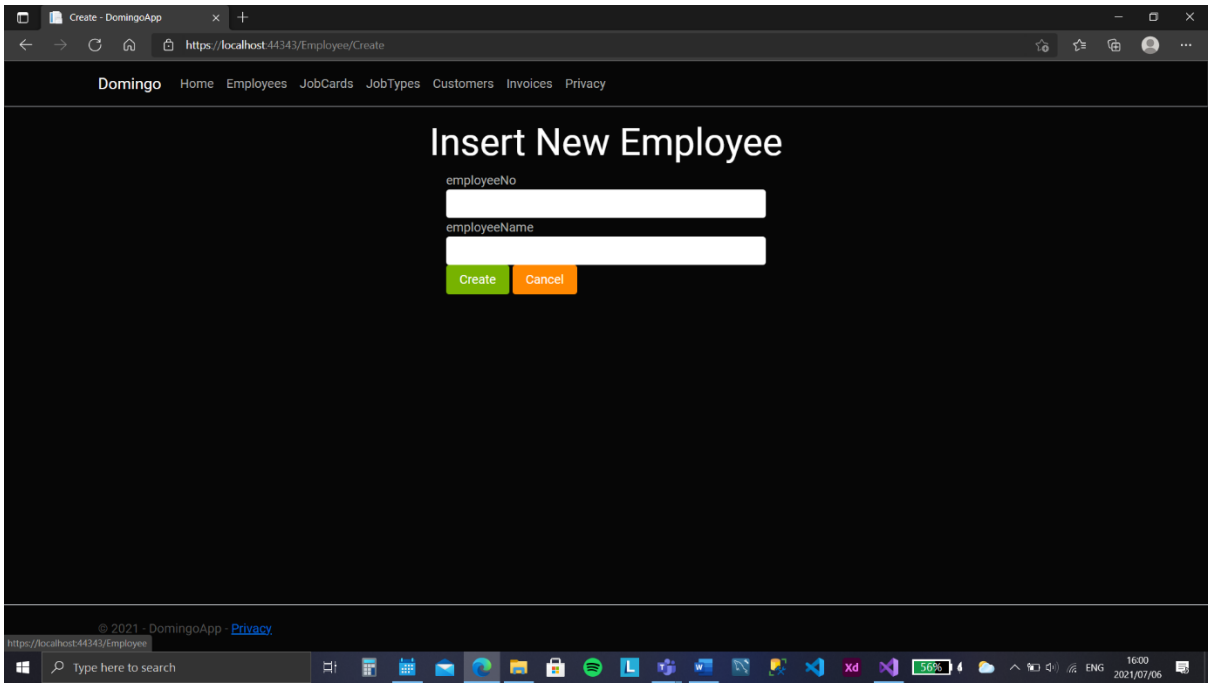


Figure 22: Add new employee page.

The user can also edit the details of an item by clicking the Edit button indicated on the right-hand side of the item's row of information (see Figure 23 below). The user will then be provided with the necessary page for updating the item's information, as displayed in Figure 24 below. For this example, the employee's name will be changed to Jeff Jackson.

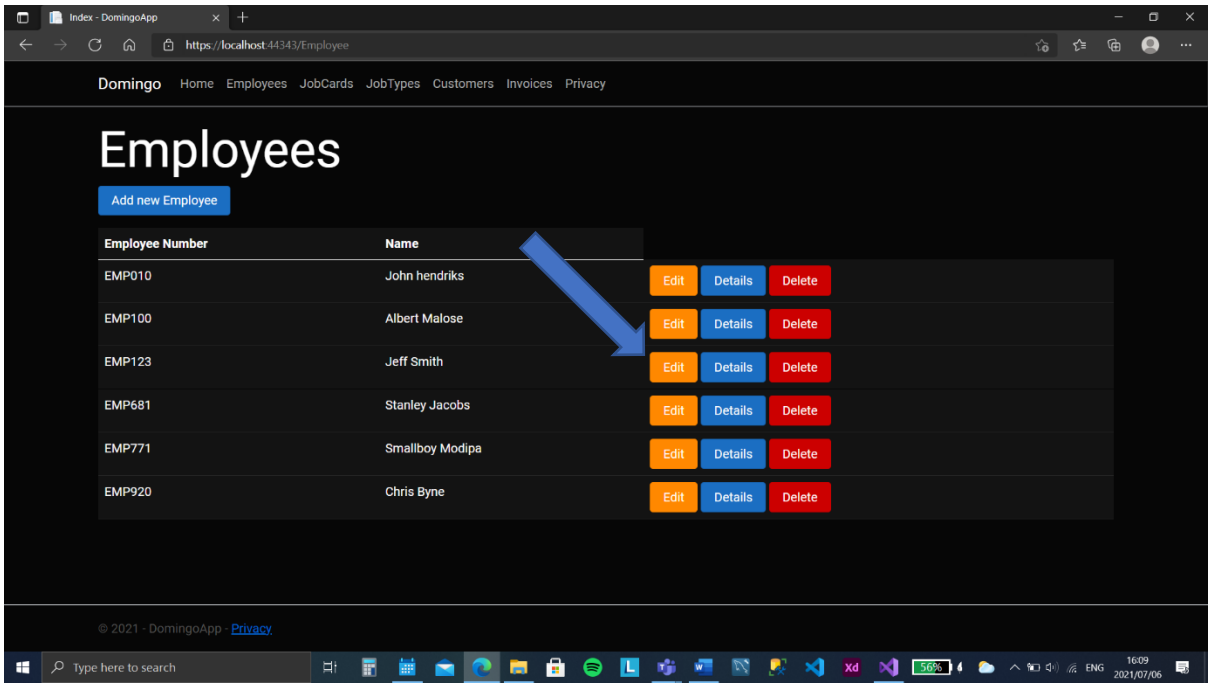


Figure 23: Edit button example.

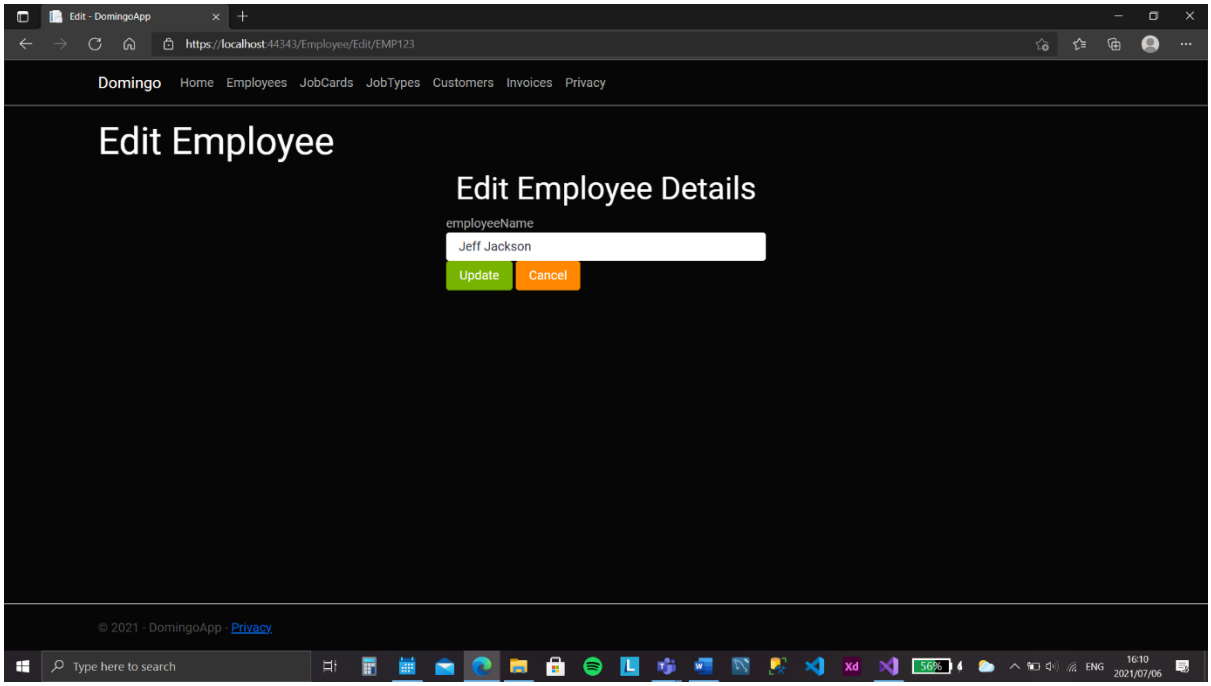


Figure 24: Edit page example.

Viewing details for a specific item is also possible by clicking on the blue Details button as indicated in Figure 25 below. In this example we will view the details of Jeff Jackson.

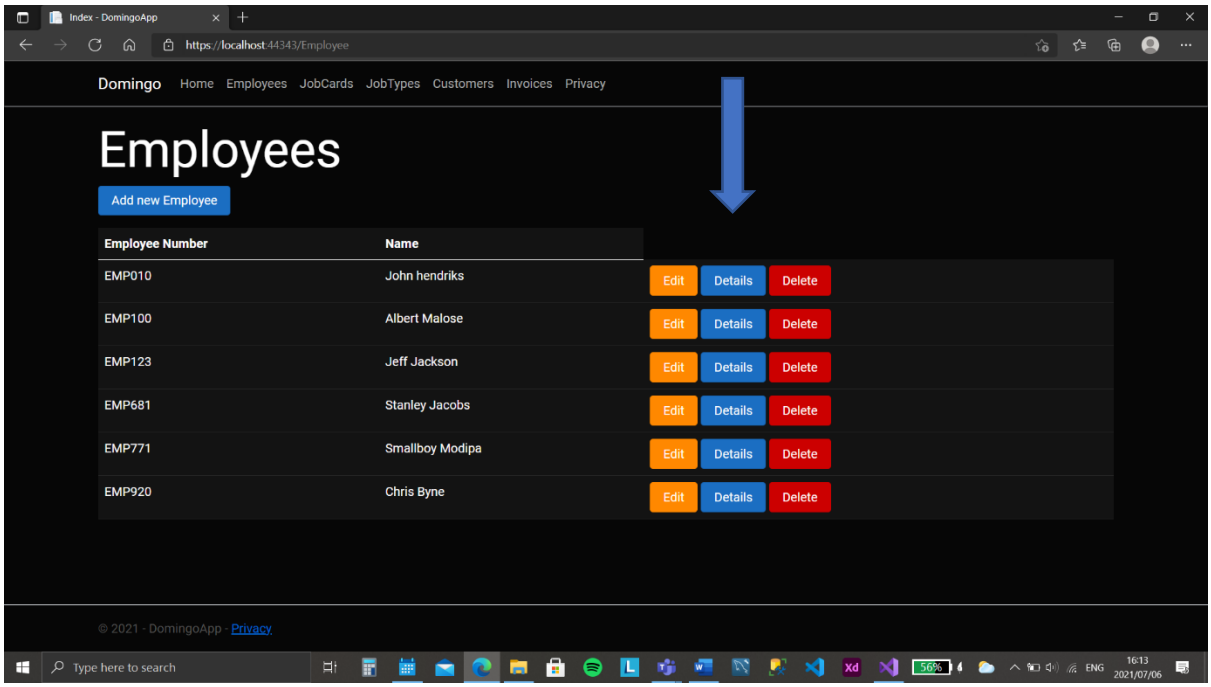


Figure 25: Details button example.

The user is presented with a new page displaying all the necessary details(see figure 26 below).

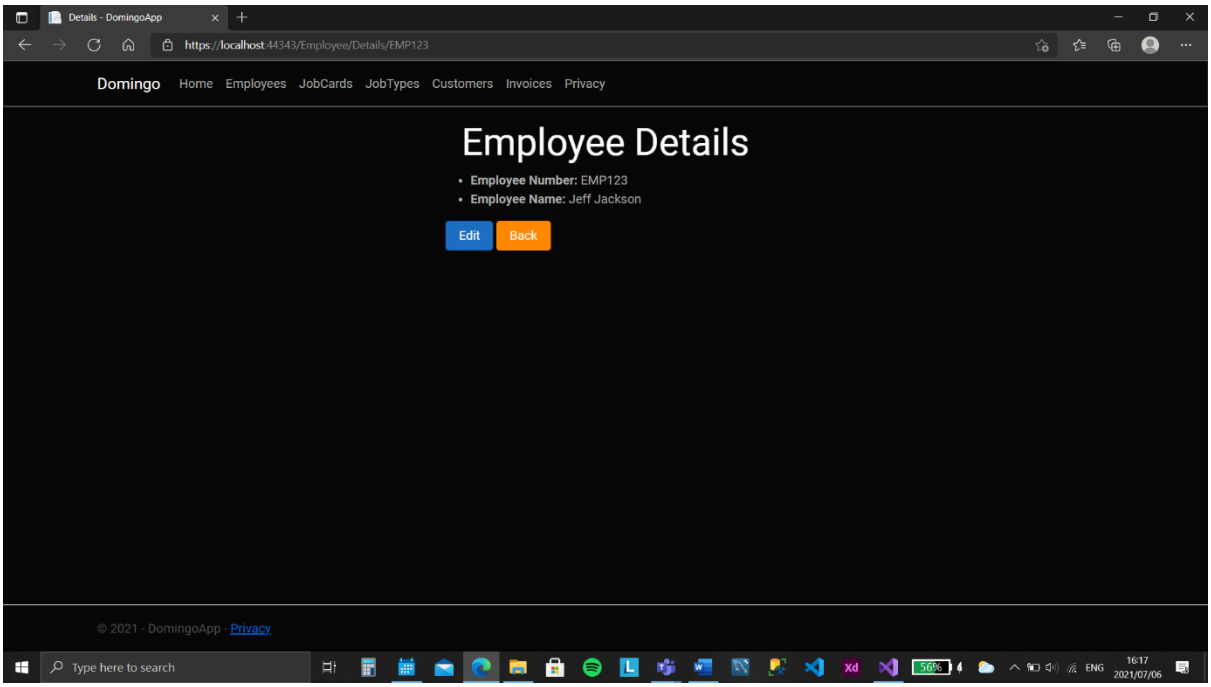


Figure 26: Details page example.

Finally, a specific item can be deleted from the database by clicking on the red Delete button indicated in Figure 27 below. For this example, we will be deleting Jeff Jackson from the database as he is no longer an employee.

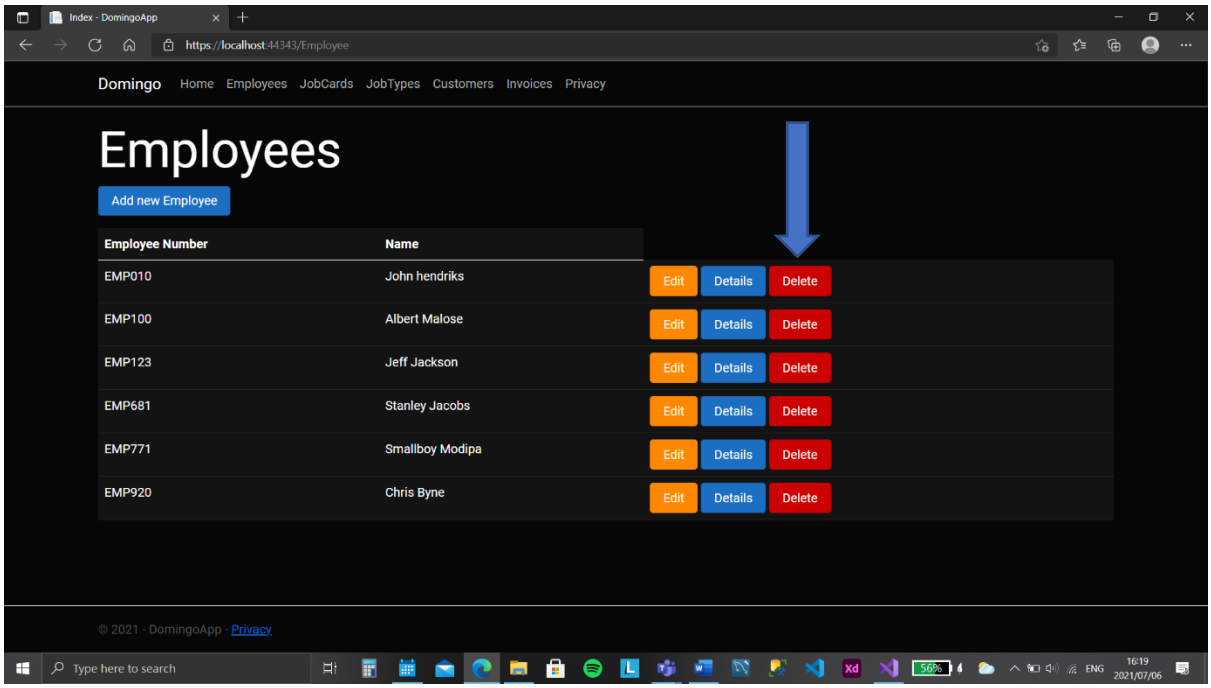


Figure 27: Delete button example.

Once the delete button is clicked the user will be redirected to the delete page which displays the information to be deleted and allows the user to cancel the delete operation if necessary(see Figure 28 below).

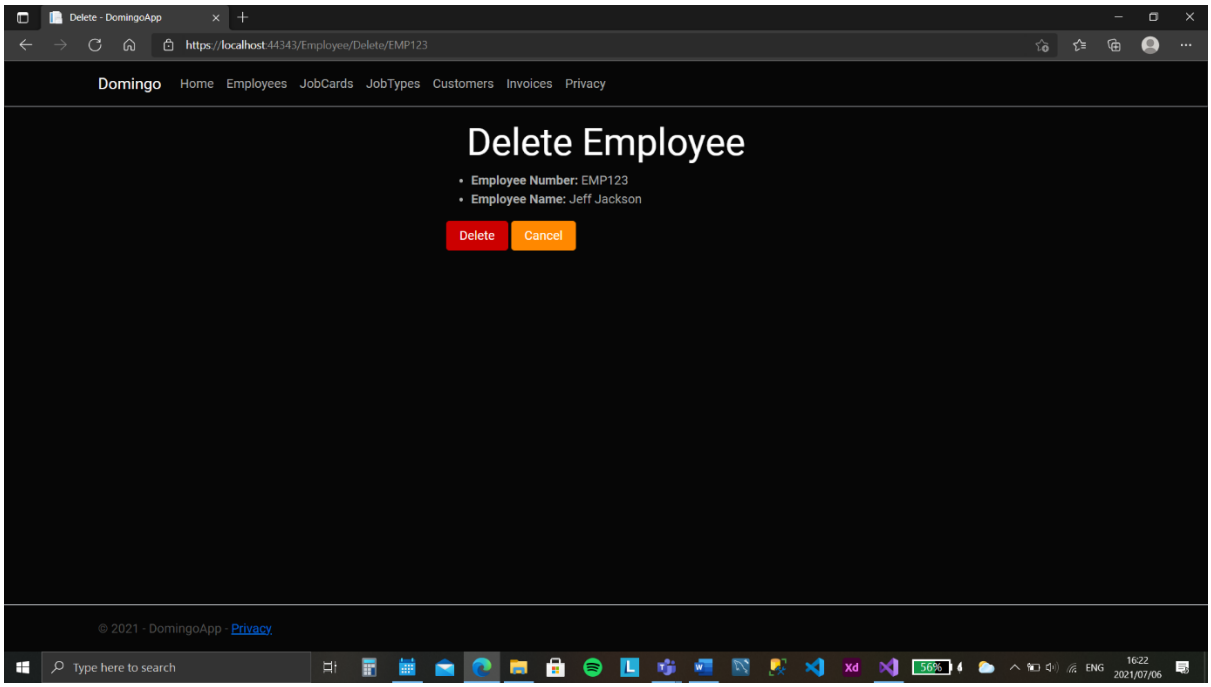


Figure 28: Delete page example.

2. Self-Evaluation

2.1. What were the most and least beneficial aspects of this course?

The most beneficial aspect of this course would be the knowledge gained on architectures, such as MVC and relational databases, as well as how various databases, and websites are designed and created. I would say there were no least beneficial aspects as all content was important and contributed to a deeper understanding of web development and cloud development.

2.2. How did each of the learning units contribute to your knowledge of web development?

Learning units 1 to 5 (Chao, 2014), in the cloud development and databases modules, taught me the database design, database implementation, and migration aspects of web development and the learning units provided in the programming module (Andrew Troelsen & Philip Japikse, 2017), enabled me to physically code the web application using C#, HTML, and CSS.

2.3. How did the knowledge gained from each of the learning units assist you in completing the tasks in this POE?

Learning unit 1 (Chao, 2014), helped me understand the fundamentals of what a database is and how it works as well as how a database interacts, or works, with the cloud.

Learning unit 2 (Chao, 2014), provided me the knowledge of the design levels of a database (conceptual design, logical design, and physical design) which enabled me to successfully design the database for Domingo Roof Works.

Learning unit 3 (Chao, 2014), taught the structure of database tables, normalization of tables, as well as the steps required to convert a database model into a relational database. Learning unit 4 (Chao, 2014), equipped me with the various SQL statements and keywords necessary to physically create the database, manipulate it, and manage it. Learning unit 3 and 4 enabled me to successfully create the database for Domingo Roof Works.

Finally learning unit 5 (Chao, 2014) as well as the tutorial provided by (Tik, 2021), provided knowledge on database migration between cloud and local servers which enabled me to migrate my local database and web app to the Azure cloud.

2.4. Do you think you will use the skills in this course in your career? If yes, explain how.

Yes. Within this career path (software engineering/IT) I see many jobs on sites like LinkedIn that require most, if not all, of the concepts(skills) provided in this course. Even jobs that do not use the exact frameworks used in this course will have a fundamentally similar layout/structure. The skills learned, such as the SQL language, and frameworks, such as MySQL and Microsoft SQL Server Management Studio, will allow me to create and manipulate databases for my employer. Architecture such as the MVC architecture, used in creating the web app, has equipped me with the necessary skills for web development with many frameworks which use this architecture.

2.5. If you were to rewind life one semester, what would you do differently in this course and what would you do the same?

I would have begun the POE Task 1 sooner as I feel I could have done better for that task, and I would have studied the theory sooner as it would have made the processes of coding and design more efficient for me as less mistakes would have been made. The process of learning the MVC architecture for the web app is something that I would have done the same as very little time was given to learn this framework and I think that I have successfully implemented it.

3. Link To Web App (Domingo Roof Works)

My Domingo Roof Works Web App URL:

https://domingoapplication.azurewebsites.net

References

Andrew Troelsen & Philip Japikse, 2017. *Pro C#7.* 8th ed. New York: Apress.

app.termly.io, 2021. *app.termly.io.* [Online]
Available at: https://app.termly.io/dashboard/website/c042eab6-af24-410e-8142-ee4bea06d9e5/privacy-policy#infocollect
[Accessed 01 July 2021].

bootswatch.com, 2021. *bootswatch.com.* [Online]
Available at: https://bootswatch.com/
[Accessed 23 June 2021].

Chao, L., 2014. *Cloud Database Development & Management.* Boca Raton, FL: CRC Press.

dotnettutorials.net, 2021. *dotnettutorials.net.* [Online]
Available at: https://dotnettutorials.net/lesson/variables-and-query-strings-in-routing/
[Accessed 19 June 2021].

IAmTimCorey, 2018. *youtube.com.* [Online]
Available at: https://www.youtube.com/watch?v=bIiEv__QNxw&t=2950s
[Accessed 17 June 2021].

Palmer, J., 2020. *liquidweb.com.* [Online]
Available at: https://www.liquidweb.com/kb/troubleshooting-microsoft-sql-server-error-18456-login-failed-user/
[Accessed 04 July 2021].

Rijn, B. v., 2020. *stackoverflow.com.* [Online]
Available at: https://stackoverflow.com/questions/58476819/entity-framework-error-invalidcastexception-the-field-of-type-system-int32-m
[Accessed 21 June 2021].

techfunda.com, 2021. *techfunda.com.* [Online]
Available at: https://techfunda.com/howto/958/model-attributes
[Accessed 20 June 2021].

Terevinto, C., 2018. *stackoverflow.com.* [Online]
Available at: https://stackoverflow.com/questions/51388735/the-model-item-passed-

into-the-viewdatadictionary-is-of-type-x-but-this-viewda
[Accessed 22 June 2021].

Tik, 2021. *youtube.com.* [Online]
Available at: https://www.youtube.com/watch?v=3ItcTmkjLsQ
[Accessed 12 June 2021].

Tik, 2021. *youtube.com.* [Online]
Available at: https://www.youtube.com/watch?v=OH8RXp8aEcU
[Accessed 18 June 2021].

Tik, 2021. *youtube.com.* [Online]
Available at: https://www.youtube.com/watch?v=kmtg7SUK718
[Accessed 23 June 2021].

tutorialbrain.com, 2020. *tutorialbrain.com.* [Online]
Available at: https://www.tutorialbrain.com/css_tutorial/css_font_family_list/
[Accessed 01 July 2020].

tutorialspoint.com, 2021. *tutorialspoint.com.* [Online]
Available at:
https://www.tutorialspoint.com/mvc_framework/mvc_framework_controllers.htm
[Accessed 06 July 2021].

w3schools.com, 2021. *w3schools.com.* [Online]
Available at: https://www.w3schools.com/SQL/sql_stored_procedures.asp
[Accessed 06 July 2021].

wdexplorer.com, 2021. *wdexplorer.com.* [Online]
Available at: https://wdexplorer.com/20-examples-beautiful-css-typography-design/
[Accessed 22 June 2021].