CMPG323 – Document

Project 2

A picture containing logo

Description automatically generated

Web Application

Francois Scholtz

23269227

19 November 2020

Table of Contents

[Introduction 4](#_Toc56677994)

[System overview 4](#_Toc56677995)

[Technologies used 4](#_Toc56677996)

[Hosting and Source Control 4](#_Toc56677997)

[Design and Architectural Patterns 5](#_Toc56677998)

[Design Patterns 5](#_Toc56677999)

[Architectural Pattern 5](#_Toc56678000)

[Solid principles 7](#_Toc56678001)

[System requirements 9](#_Toc56678002)

[Entity Relationship Diagram (ERD) 9](#_Toc56678003)

[Use cases 9](#_Toc56678004)

[Data flow diagram 10](#_Toc56678005)

[Web Developer 10](#_Toc56678006)

[How to guide 10](#_Toc56678007)

[CRUD 11](#_Toc56678008)

[C – Create 11](#_Toc56678009)

[R – Retrieve/Read 11](#_Toc56678010)

[U – Update 11](#_Toc56678011)

[D – Delete 11](#_Toc56678012)

[Process flow diagrams 12](#_Toc56678013)

[Registration and login process 12](#_Toc56678014)

[Guest Role processes 12](#_Toc56678015)

[Employee Role processes 13](#_Toc56678016)

[Manager Role process 13](#_Toc56678017)

[Administrator Role process 13](#_Toc56678018)

Table of figures

[Figure 1: Table of Technology used 4](#_Toc56677972)

[Figure 2: Microsoft Azure hosting services provided 4](#_Toc56677973)

[Figure 3: Layout of the Repository architecture courtesy of Jacqui from Dimension data 5](#_Toc56677974)

[Figure 4: Focus on GET and POST requests courtesy of Jacqui from Dimension data 6](#_Toc56677975)

[Figure 5: Example of SRP in code – HomeController.cs 6](#_Toc56677976)

[Figure 6: Example of OCP in code - ApplicationDBContextModelSnapshot.cs 7](#_Toc56677977)

[Figure 7: Example of LSP in code – Program.cs 7](#_Toc56677978)

[Figure 8: Example of DIP in code -UserRoles.Controller.cs 8](#_Toc56677979)

[Figure 9: Solution system ERD 8](#_Toc56677980)

[Figure 10: Solution system use case diagram 9](#_Toc56677981)

[Figure 11: Solution system DFD 9](#_Toc56677982)

[Figure 12: Registration and Login process 11](#_Toc56677983)

[Figure 13: Guest Role process 12](#_Toc56677984)

[Figure 14: Employee Role process 12](#_Toc56677985)

[Figure 15: Manager Role process 12](#_Toc56677986)

[Figure 16: Administrator Role process 13](#_Toc56677987)

# Introduction

The DimDatPro application was developed as part of the CMPG323 Project 2 requirements. It is individual industry project with a brief received in collaboration with Dimension Data. The project scope was open ended and limited to retrieving, capturing and maintaining data. The web application development route was taken to address proof of concept. The solution system is outlined beginning with a system overview describing the system briefly.

# System overview

## Technologies used

Various technologies were used to during the development of the solution application these are discussed outlined below.

|  |  |
| --- | --- |
| **Technology** | **Application** |
| Microsoft Visual Studio 2019 | Development Tool used to develop the project. |
| ASP.NET MVC CORE | As required by brief. Framework used for the web development. |
| SQL Server 2017 Express Edition | csv Excel file provided was imported to create a database. This database used as operational database. |
| Entity Framework Identity | User data management (registrations and profile management) implementation. |
| Entity Framework MVC Core | Operational data management (records CRUD processing) |
| Power BI | Business data analytics dashboard tool used to analyse the operational data. |

Figure 1: Table of Technology used

## Hosting and Source Control

The solution application is hosted using cloud services provided by Microsoft Azure. The project was published directly from Visual Studio to Azure. The following is layout of the hosting services accessed from Azure. The application URL is <https://dimdatpro2020201118135626.azurewebsites.net>

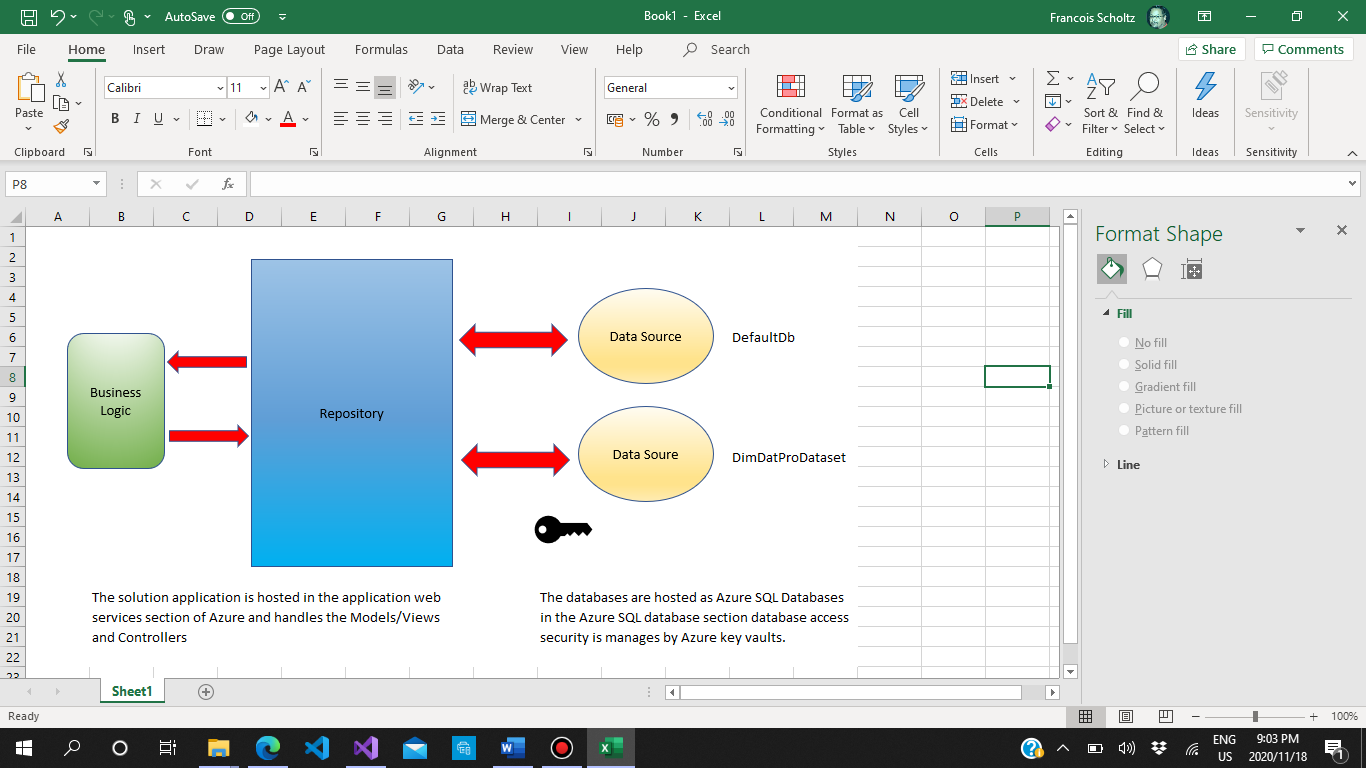


Figure 2: Microsoft Azure hosting services provided

Source control was done using GitHub through commits made from GitBash regularly. The URL for the repository is <https://github.com/FAScholtz/CMPG323---Projecct2---Second>.

## Design and Architectural Patterns

### Design Patterns

Various design patterns were used in the solution application. These design patterns are described by Nikko Bautista (Bautista, 2010) as having been optimised and are reusable solutions to computer programming problems that can be implemented in the correct situations. Design patterns implemented are:

* Structural patterns

They make it easier for the various entities to work together.

* Creational patterns

They make it easier for the creation of object through instantiation.

* Behavioural patterns

They make it easier for the various entities to communicate with each other.

The patterns above were implemented in the solution explorer through the use of the Entity Framework in creation of controllers.

### Architectural Pattern

Architectural design patters are like design patterns but are applied more broadly (Mallawaarachchi, 2017). The design patterns used in the solution explorer are:

* The standard MVC pattern
  + Models

They represent objects that carry data.

* + View

They represent the visualisations of the data that the models contain and can also have logic to update controllers if data changes

* + Controllers

They act on both model and view controlling data flow into model objects and updates the view whenever data changes. They keep views and models separate

* The repository architectural pattern

The repository architectural pattern is used to create an abstraction layer between the business logic and data access layers therefore insulating the application from changes that are made in the data store. It is illustrated in the following diagram.

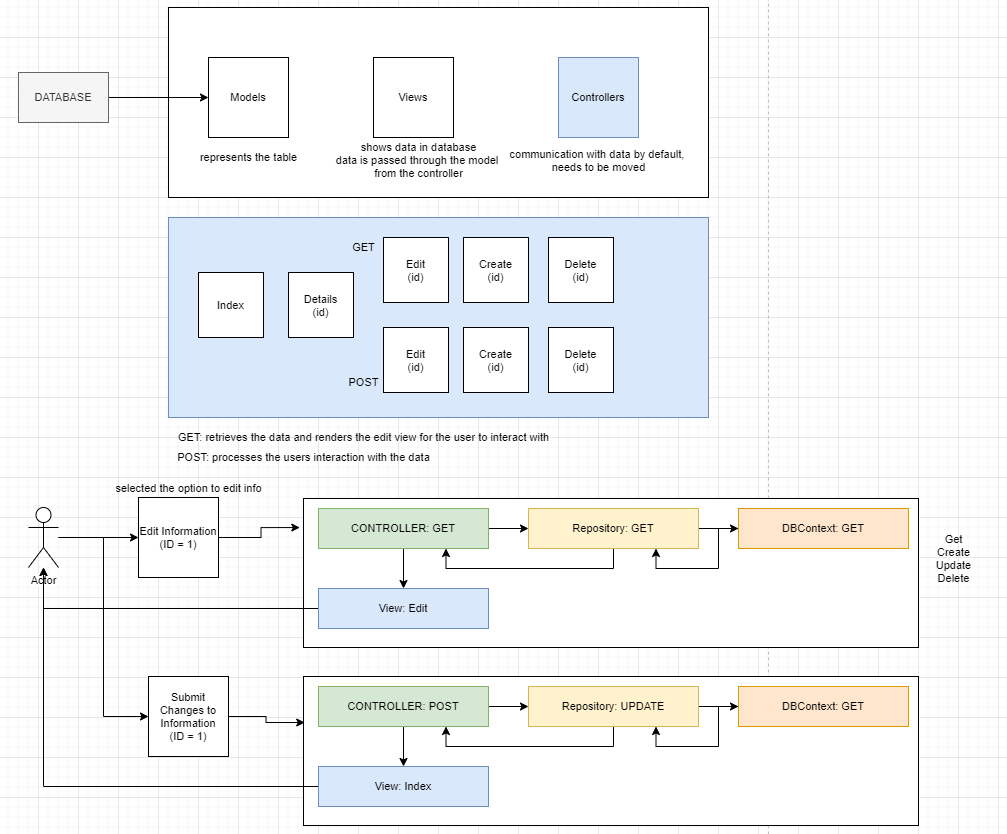


Figure 3: Layout of the Repository architecture courtesy of Jacqui Muller (Muller, 2020)

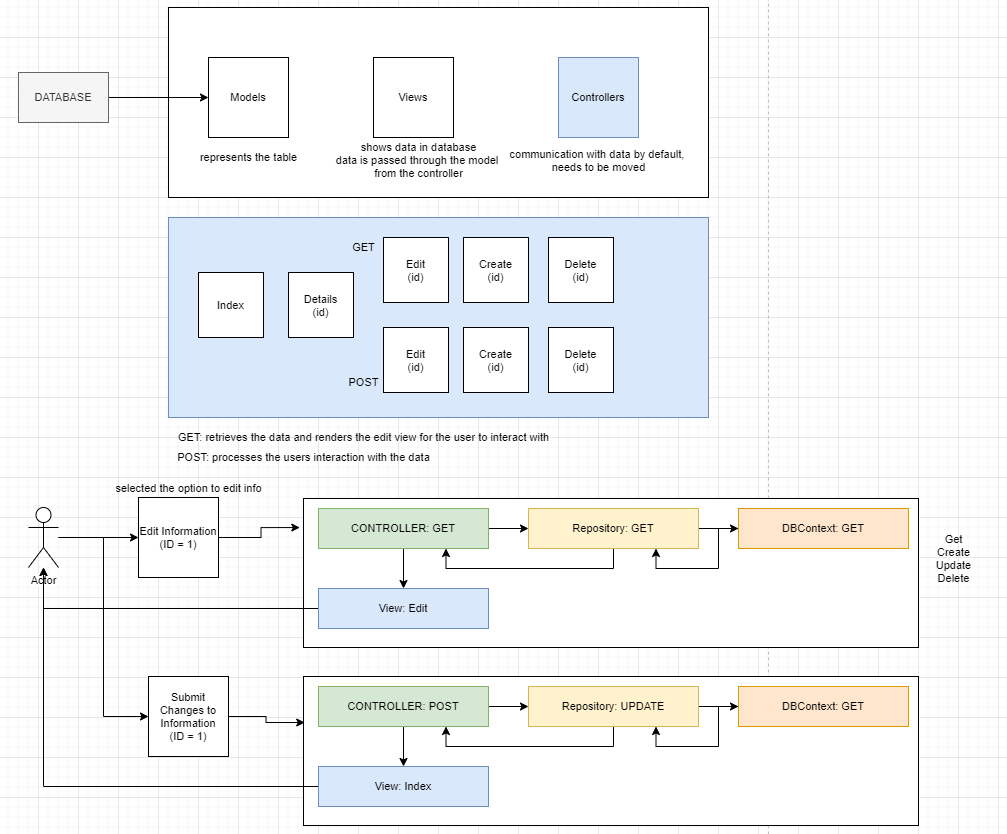


Figure 4: Focus on GET and POST requests courtesy of Jacqui Muller (Muller, 2020)

### Solid principles

In an attempt to make the solution application more maintainable, understandable, and flexible as elaborated on further by Sachin Gandhi (Gandhi, 2018). These five design principles constituting SOLID were used. They are

* Single Responsibility Principle (SRP):

Single classes should not be overloaded with multiple responsibilities similarly multiple classes should not share responsibilities or have mixed responsibilities.

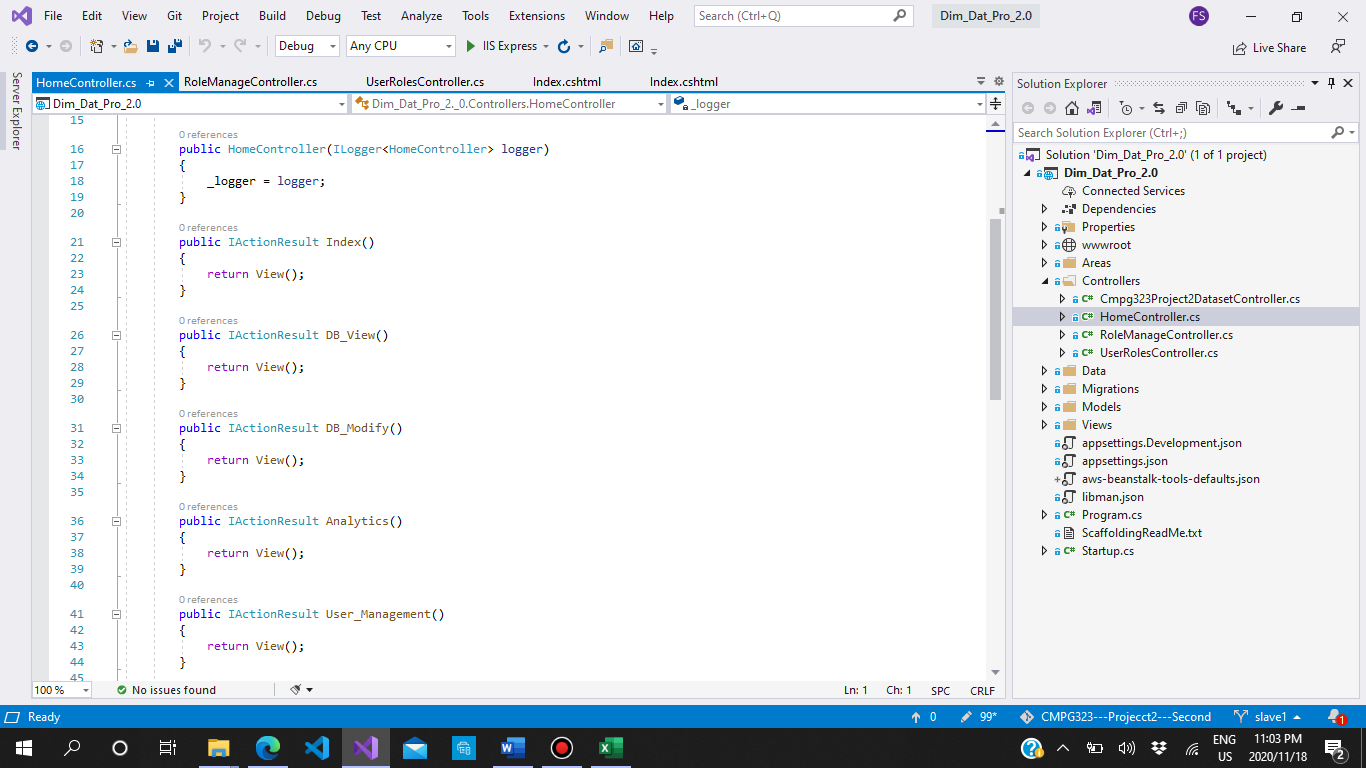


Figure 5: Example of SRP in code – HomeController.cs

* Open Closed Principle (OCP)

The ability to extend classes without changing its core implementations.

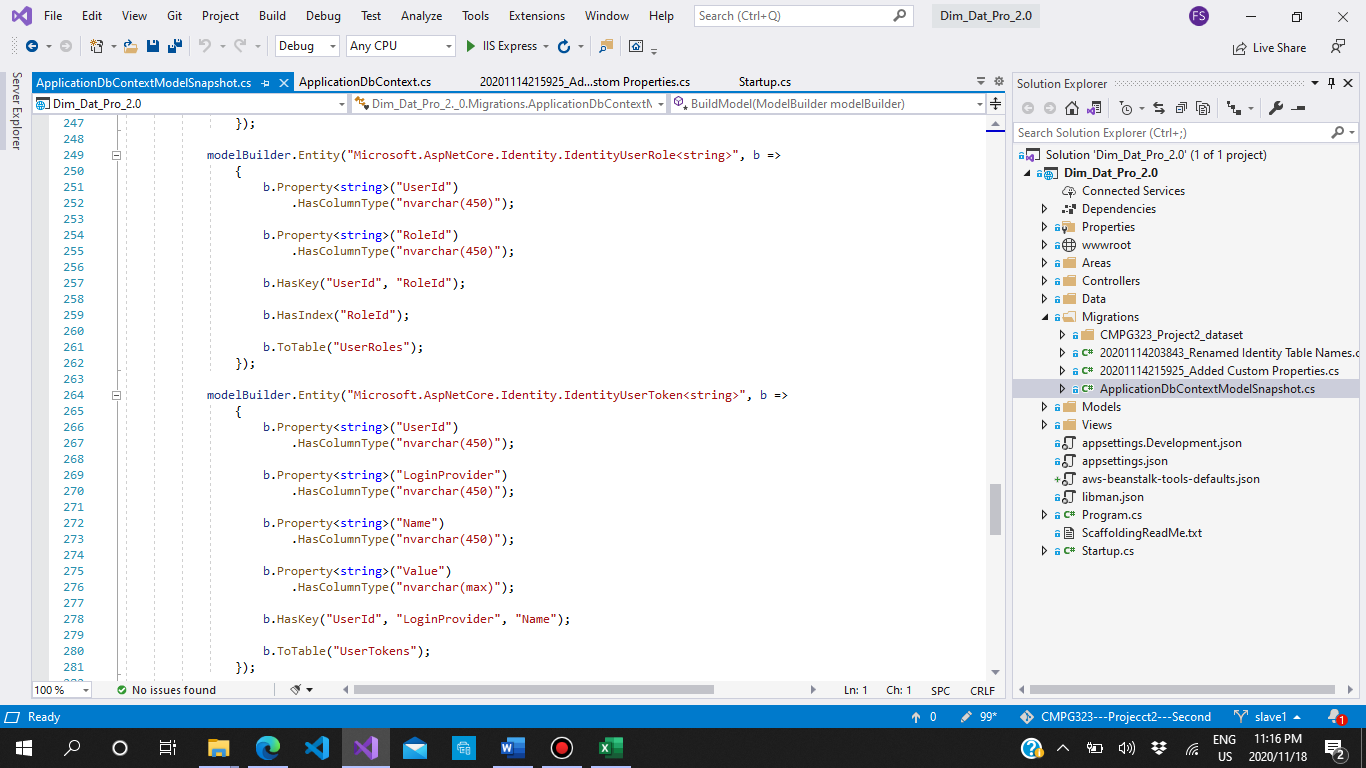


Figure 6: Example of OCP in code - ApplicationDBContextModelSnapshot.cs

* Liskov Substitution Principle (LSP)

child class should be perfectly substitutable for their parent class. If class B is a child, or is derived from, parent class A, then class B should be substitutable for class A.

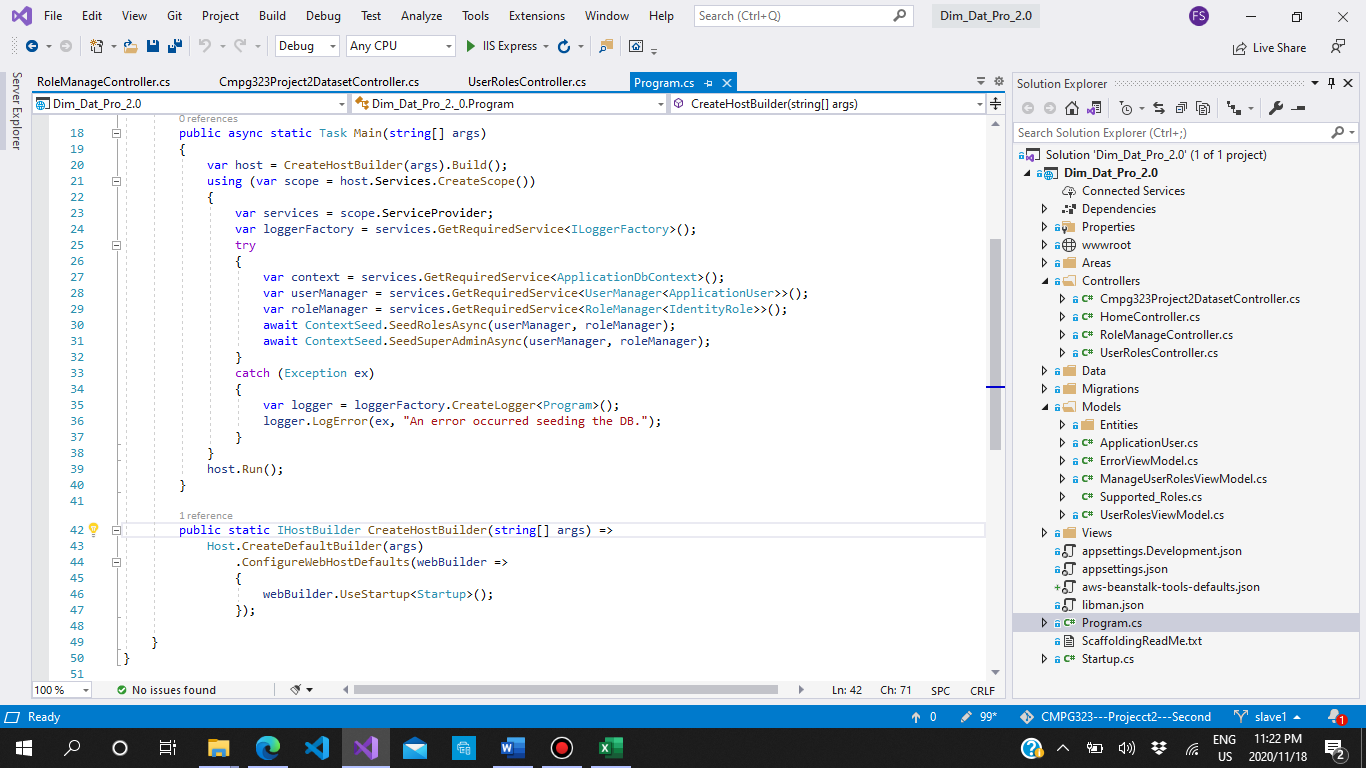


Figure 7: Example of LSP in code – Program.cs

* Interface Segregation Principle (ISP)

The creation separate interfaces for each operation or requirement rather than having a single class to do the same work. No example found in code however, no violations either

* Dependency Inversion Principle (DIP)

There should not be a tight coupling among components of software and to avoid that, the components should depend on abstraction.

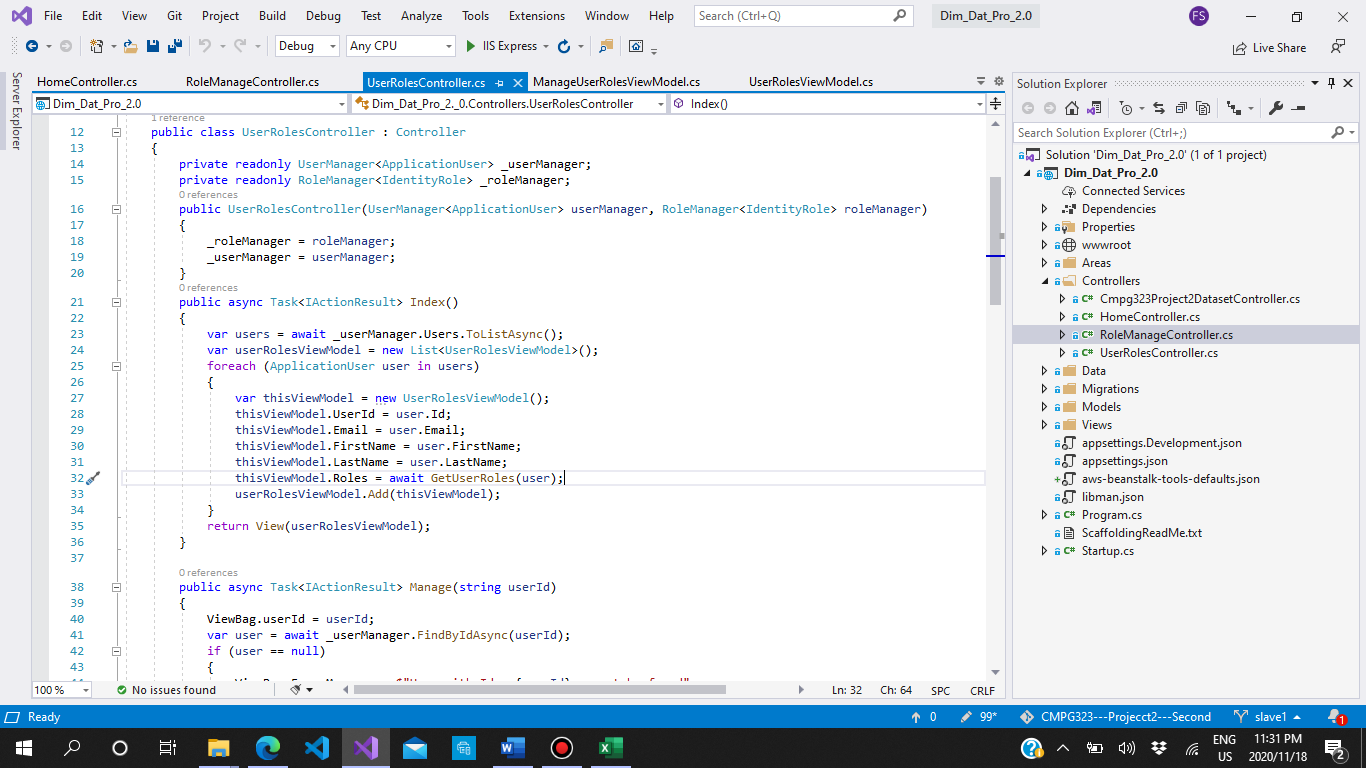


Figure 8: Example of DIP in code -UserRoles.Controller.cs

# System requirements

## Entity Relationship Diagram (ERD)

Diagram

Description automatically generated

Figure 9: Solution system ERD

## Use cases

Diagram, schematic

Description automatically generated

Figure 10: Solution system use case diagram

## Data flow diagram

Diagram

Description automatically generated

Figure 11: Solution system DFD

# Web Developer

## How to guide

The landing page, “Home”, provides brief information on the solution system. The user than has an option to either “Register’, if new to the system, or “Login”, if previously registered. If the user chooses to register a form requiring their information is required will be presented to be completed validation is used for email address and password inputs. Once registered a page for confirmation is presented. If the user chooses login the form the presented required the username or password to be inserted the username can be their email address or the prefix to the email address, e.g. [abc@gmail.com](mailto:abc@gmail.com) or just “abc”. Depending on the roles assigned to users by the administrator, the default being “guest”, access will be granted to the system. The role level access provided to the users are from lowest to highest is:

* “Guest”

This is the default access provided. Only access provided to the pages for “Home”, “Register”, and “Login/Log out”.

* “Employee”

The users have access to “Home”, “View Database” (with limited functionality), “Business Analytics”, “Register”, and “Login/Log out”.

* “Manager”
* The users have access to “Home”, “View Database” (with moderate functionality), “Business Analytics”, “Register”, and “Login/Log out”.
* Administrator”
* The users have access to “Home”, “View Database”, “Business Analytics”, “User Role Management”, “Role Management”, “Register”, and “Login/Log out”.

## CRUD

It is an acronym used in computer programming that four basic functions of persistent storage. They refer to all the functions mapped to HTML, SQL and executed in relational databases applications. The implementation of each the functions in the user database and the operational database are as follows:

### C – Create

In the user database, the registration of users is the process of creating new user records in the databases. In the operational database the creation of new records is done in the View Database page.

### R – Retrieve/Read

When logging into the system the page posts data to the database to enquire if the record exists. If it does, the password for the user is matched to the inputted password and if its matched access is granted. Also, there is a search function when the User Role Management page is being loaded where the user details is posted to the user database and the roles assigned to the user is displayed. Furthermore, a search function on the View Database page retrieves the department records based on the search text as well as display the individual record details.

### U – Update

The user can log into their Profile page to amend and save any of their details including adding a picture. In the Role Management page, the administrator can add more roles. Similarly, the User Role Management page can be used by the administrator to add roles to users. In the operational database, under the View Database page, managers can update records

### D – Delete

In the User Role Management page, the administrator dan remove roles assigned to users. The administrator can also delete individual records.

# Process flow diagrams

## Registration and login process

Diagram

Description automatically generated

Figure 12: Registration and Login process

## Guest Role processes

A picture containing object, lamp, table

Description automatically generated

Figure 13: Guest Role process

## Employee Role processes

Diagram

Description automatically generated

Figure 14: Employee Role process

## Manager Role process

Diagram

Description automatically generated

Figure 15: Manager Role process

## Administrator Role process

Diagram

Description automatically generated

Figure 16: Administrator Role process

# References

Bautista, N. 2010, July. *A Beginner’s Guide to Design Patterns*. <https://code.tutsplus.com/articles/a-beginners-guide-to-design-patterns--net-12752#:~:text=%20There%20are%20three%20basic%20kinds%20of%20design,1%20structural%202%20creational%203%20behavioral%20More%20>. Date of access: 16 Nov 2020.

Mallawaarachchi, V. 2017. *10 Common Software Architectural Patterns in a nutshell*. <https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013#:~:text=%2010%20Common%20Software%20Architectural %20Patterns%20in%20a,consists%20of%20two%20parties%3B%20master%20and...%20More%20>. Date of access: 16 Nov. 2020.

Muller, J. 2020. Project 2. Unpublished lecture notes on eFundi, CMPG323. Potchefstroom: NWU.

Gandhi, S., 2018. *SOLID Design Principles Explained - C#*. <https://www.dotnettricks.com/learn/designpatterns/solid-design-principles-explained-using-csharp#:~:text=There%20are%20five%20SOLID%20principles%3A%201% 20Single%20Responsibility,Segregation%20Principle%20%28ISP%29%205%20Dependency%20Inversion%20Principle%20%28DIP%29>. Date of access: 16 Nov. 2020.