Advanced Programming CIT 3009

Research Project - John’s Snack Shop

Author: Daniel Pinto 1306537

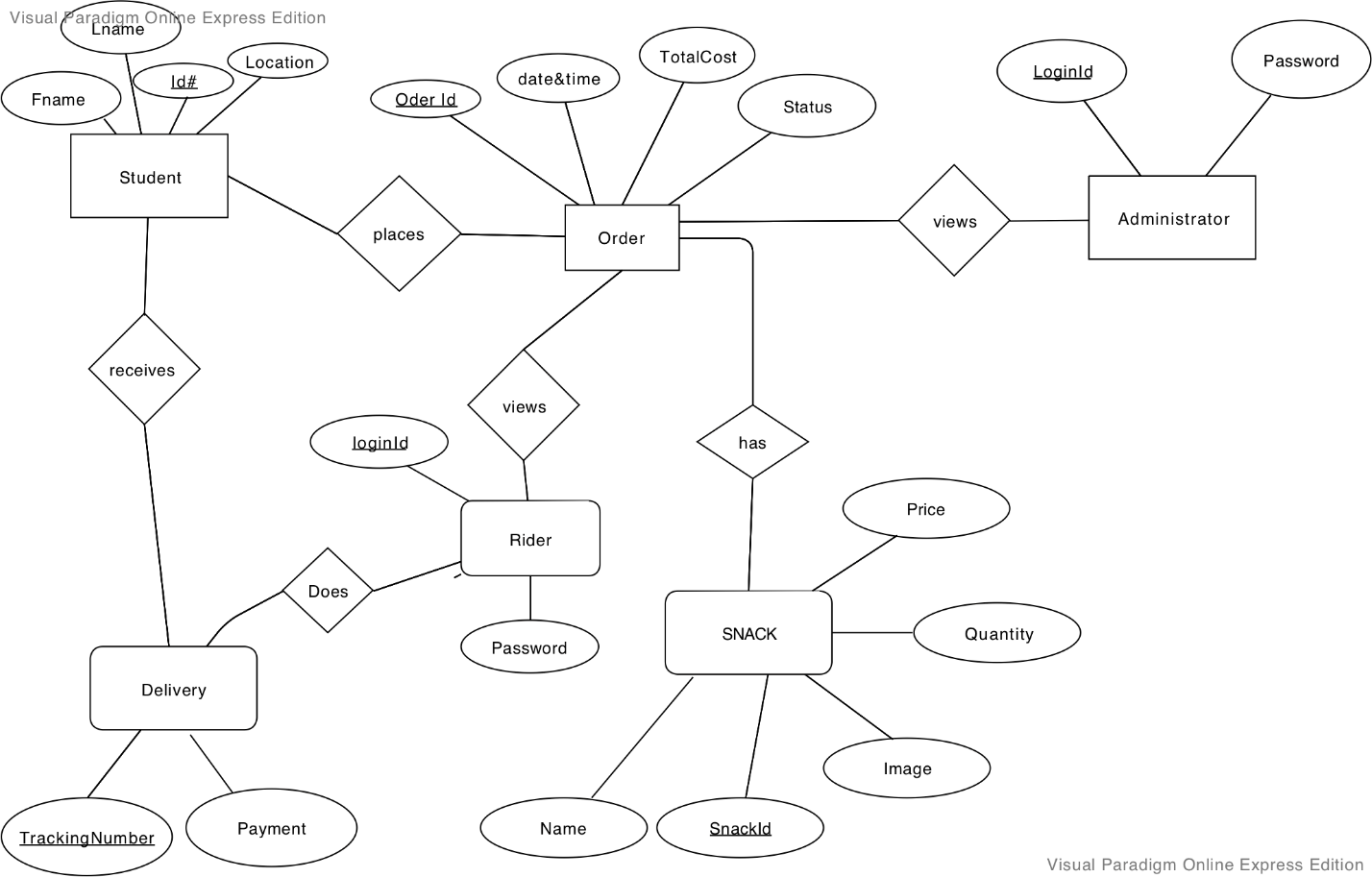
Rory Adamson 0903081

University of Technology Jamaica

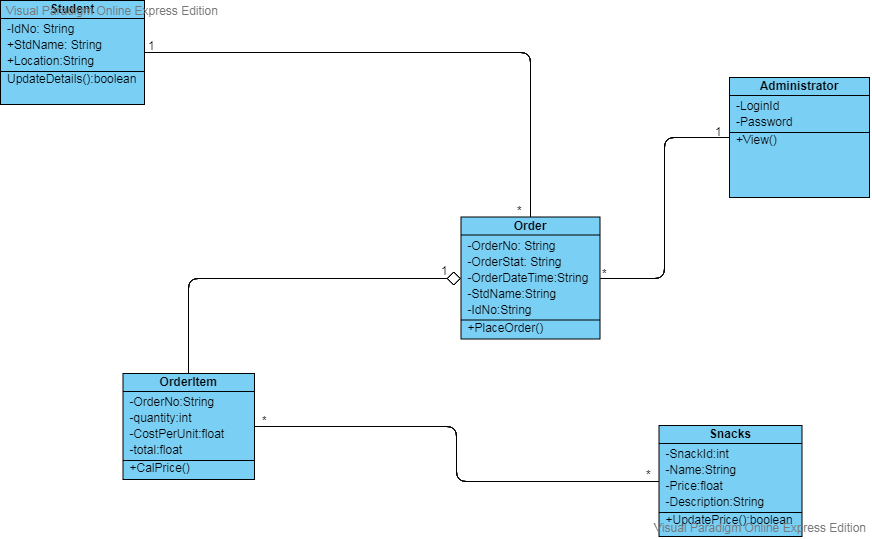
Abstract

This project focuses on a web-based solution for the proprietors of ‘John’s Snack Shop’, which is located on the campus of the University of Technology Jamaica. John’s Snack Shop is a popular entity which provides students with a variety of snack options. Currently students have to attend the location physically in order to purchase their desired snack. The solution will create a platform where students of the university who are desirous of purchasing snacks, can do so online and have it delivered to them. The solution will demonstrate the use of the following; S.O.L.I.D principles, Repository Pattern, Model-View-Controller Pattern, Singleton Pattern, Factory Pattern, Code Generation Tool, Source Control Management Tool, Package Management Tool, Unit Testing and Test Automation. The solution

Diagrams

Figure 1 represents the ERD diagram for the coded solution.

**Figure 1**

Figure 2 is a representation of the class diagram used for the solution.

**Figure 2**

Solid principles

The SOLID principles are a dependency management for object oriented programming design and was introduced by Robert Cecil Martin (Carr, 2010). The five principles are listed below and explained with the aid of pictorial evidence from the source code.

1. Single Responsibility principle

The Single Responsibility Principle (SRP) states that there should never be more than one reason for a class to change (Carr, 2010). This means that every class, or similar structure, in your code should have only one job to do.

1. Open-closed principle

This principle speaks to the fact that classes should be open for extension but closed for modification (Carr, 2010). Open to extension means that you should design your classes so that new functionality can be added as new requirements are generated while closed for modification means that once you have developed a class you should never modify it, except to correct bugs (Carr, 2010).

1. Liskov substitution principle

This principle specifies that classes should be designed so that client dependencies can be substituted with subclasses without the client knowing about the change (Carr, 2010).

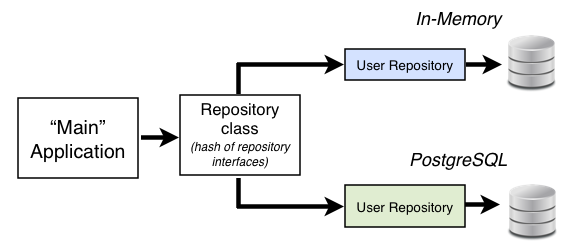
1. Interface Segregation principle

This principle states that clients should not be forced to depend on interfaces that they do not use (Carr, 2010).

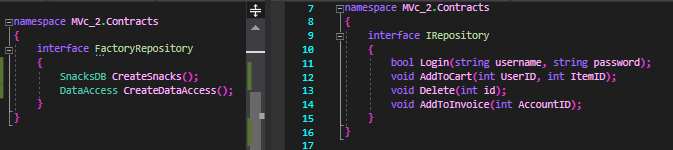
1. Dependency Inversion Principle

This states that high level modules should not depend upon low level modules. Both should depend upon abstractions additionally, abstractions should not depend upon details. Details should depend upon abstractions (Carr, 2010).

Repository Pattern

The aim of the repository pattern is to create an abstraction layer between the data access layer and the business logic layer of an application (Dykstra, 2013). Implementing these patterns can help isolate your application from changes in the data store and can facilitate automated unit testing or test-driven development (TDD) (Dykstra, 2013). The below figure 3 shows the concept this principle pushes.

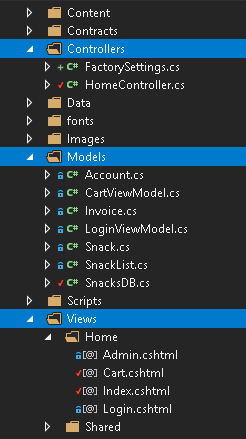
**Figure 3**



**Figure 3.1 Repository pattern use in the source code**

Model View Controller Pattern

MVC is a design pattern used to decouple user-interface (view), data (model), and application logic (controller). This pattern helps to achieve separation of concerns. Using the MVC pattern for websites, requests are routed to a Controller which is responsible for working with the Model to perform actions and/or retrieve data. The Controller chooses the View to display, and provides it with the Model. The View renders the final page, based on the data in the Model ("ASP.NET MVC Pattern | .NET"). Figure 4 represents a generic representation of the MVC pattern.



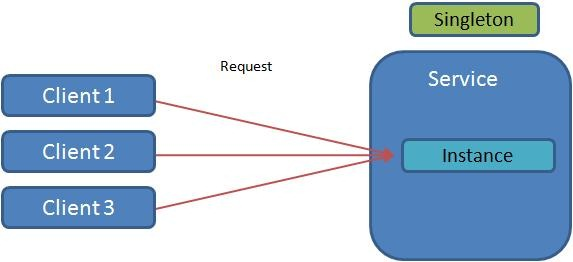
**Figure 4.1 MVC Pattern representation in our Source Code**



**Figure 4**

Singleton Pattern

Singleton pattern defines an instance operation that lets clients access its unique instance.It's important for some classes to have exactly one instance("Singleton", 2019). Although there can be many printers in a system, there should be only one printer spooler("Singleton", 2019). There should be only one file system and one window manager and a digital filter will have one A/D converter as well as an accounting system will be dedicated to serving one company("Singleton", 2019).



**Figure 5 pictorial representation of the singleton pattern.**

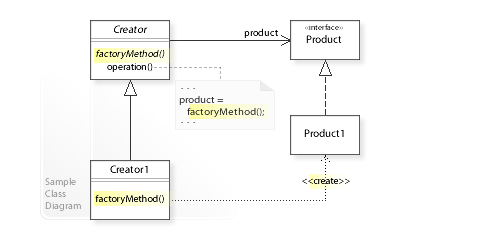


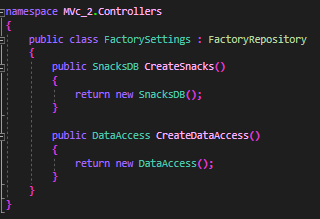
**Figure 5.1 singleton representation in the source code.**

Factory Pattern

The factory method design pattern defines an interface for creating an object, but lets subclasses decide which class to instantiate(Anderson, 2007). Factory Method lets a

class defer instantiation to subclasses (Anderson, 2007).

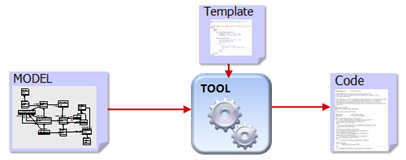
**Figure 6 Factory Pattern pictorial representation**



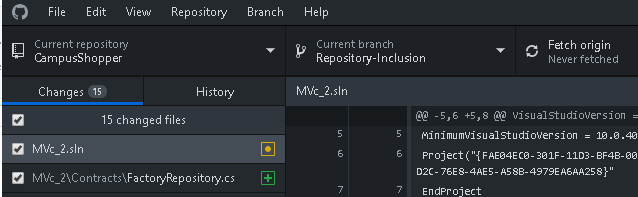
**Figure 6.1 Factory pattern representation in the source Code**

Code Generation Tool

A code generator allows you to leverage the models created with your modelling tools ("Custom code generation tools - Obeo", n.d.). They ensure continuity between your design and development activities ("Custom code generation tools - Obeo", n.d.).

**Figure 7 Code generation tool pictorial representation**

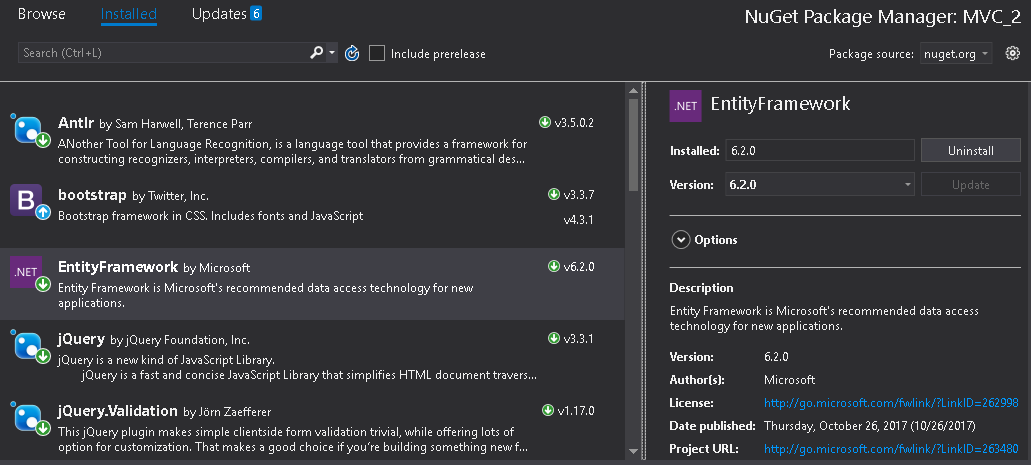
Source Control Management Tool

Source Code Management primarily involves tracking the modifications to code("Source Code Management"). Tracking modifications assists development and collaboration by providing a running history of development and helping to resolve conflicts when merging contributions from multiple sources ("Source Code Management"). Software tools for source code management are sometimes referred to as "Source Code Management Systems" (SCMS), "Version Control Systems" (VCS), "Revision Control Systems" (RCS) or simply "code repositories" depending on what features they provide or how they are being used("Source Code Management").

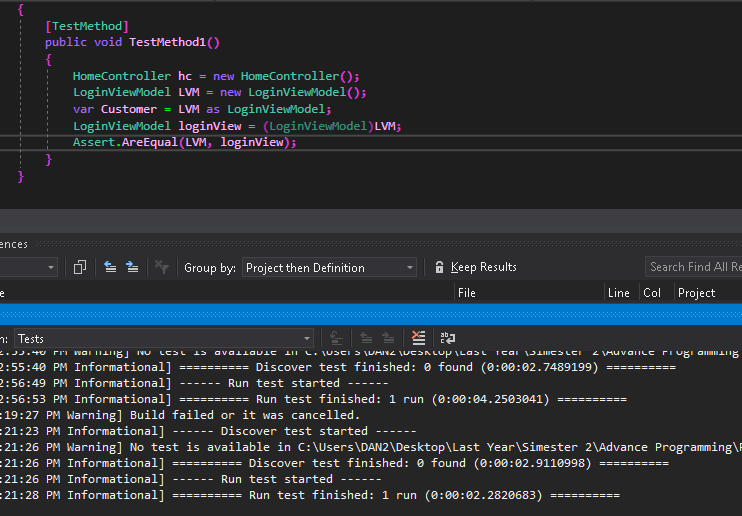
**Figure 8 example of source control management tool in the source code**

Package Management Tool

A Package Manager allows tracking the installation of files making it easy to remove and upgrade packages ("6.3. Package Management", 2019). As well as the binary and library files, a package manager will handle the installation of configuration files ("6.3. Package Management", 2019).

**Figure 9 shows the package manage tool which was used**

Unit Testing

Unit testing is a level of software testing where the individual units or components of a software or web applications are tested (Leo, n.d.). It is an important part of software testing and a component of test driven development (Leo, n.d.).

**Figure 10 representation of unit testing in the source code**

Continuous Integration

Continuous Integration refers to a number of things;

* A practice of always building a working version of your system each day(Anderson, 2012).
* A practice of always running your test cases each day. A practice of committing changes to your version control system each day(Anderson, 2012).
* An automated system that monitors changes to your version control system such that when it detects the commit it checks out that version of your system builds it, runs all tests, and verifies that it passes(Anderson, 2012).
* if it does, it creates a new official release if it doesn’t, it notifies the appropriate developers(Anderson, 2012).

References

6.3. Package Management. (2019). Retrieved from <http://ftp.oregonstate.edu/.2/lfs-website/lfs/view/development/chapter06/pkgmgt.html>

Anderson, K. (2007). Retrieved from <https://www.cs.colorado.edu/~kena/classes/6448/f07/lectures/21/21-Factory.pdf>

Anderson, K. (2012). Retrieved from <https://www.cs.colorado.edu/~kena/classes/5828/s12/lectures/27-continuousintegration.pdf>

Carr, R. (2010, December 12). Retrieved April 11, 2019, from <http://www.blackwasp.co.uk/SOLID.aspx>

Custom code generation tools - Obeo. Retrieved from <https://www.obeo.fr/en/services/code-generation-tools>

Dykstra, T. (2013). Implementing the Repository and Unit of Work Patterns in an ASP.NET

MVC Application (9 of 10). Retrieved from <https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application>

Garnaik, R. (2019). Singleton Design Pattern- Java Tech Guy must read!. Retrieved from <https://javainsider.wordpress.com/2012/10/25/singleton-design-pattern-java-tech-guy-must-read/>

Leo, M. Unit Testing significance in Software Development. Retrieved from <http://www.academia.edu/Documents/in/Unit_Testing>

Singleton. (2019). Retrieved from <http://www.cs.unc.edu/~stotts/GOF/hires/pat3efso.htm>

Source Code Management. (n.d.). Retrieved from <https://its.unl.edu/bestpractices/source-code-management>