Advanced Programming

Research Report

University of Technology, Jamaica

Done By:

Demar Johnson ID#: 1608193

Danyel Roper ID#: 1608198

Tutor: Mr Gilroy Gordon

April 10, 2019

John Shop Documentation

# ERD

# UML (Class Diagram)

# WIREFRAME

# 

# Design Patterns

This project utilized the following design patterns:

* Factory
* Repository
* Model-View-Controller
* Singleton

The SOLID principle was also utilized.

## Singleton Pattern

Singleton pattern is one of the simplest design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class. (tutorialspoint, Design Pattern - Singleton Pattern, 2019)

## Implemented:

This pattern can be observed in our ‘entities’ package as all our domain classes are given a single responsibility; creating an object and nothing else.

## Model-View-Controller

MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

* **Model** - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
* **View** - View represents the visualization of the data that model contains.
* **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate. (tutorialspoint, Design Patterns - MVC Pattern, 2019)

## Implemented:

This pattern can be observed both in the backend and the front end of our code.

The package names would give this pattern away.

The Model in our project would be our database and entities packages, the Controller would be both the controllers package in our backend and the controllers package found in our frontend. That then leaves us with our view which is the index page working with our angularjs frontend framework.

## Repository

Essentially, it provides an abstraction of data, so that your application can work with a simple abstraction that has an interface approximating that of a collection. Adding, removing, updating, and selecting items from this collection is done through a series of straightforward methods, without the need to deal with database concerns like connections, commands, cursors, or readers. (Fadatare, 2018)Using this pattern can help achieve loose coupling and can keep domain objects persistence ignorant.

## Implemented:

### Generic Repository Implementation

The repository pattern was implemented using a generic class to attain all connections to our specific SQl Provider,SQLite, this class is then extended by our business model classes to attain a database table for each business model.

## Factory

Factory pattern is one of the most used design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object. In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface. (tutorialspoint, 2019)

## Implemented:

This pattern was observe in the frontend of our code, specifically the structure which allowed for the login functionality of our application. An angular service called SharedLogin was responsible for creating objects used to hold the ng-hide statuses of our variables. This object was then used by our LoginController and our OptionController both found in the js folder of our frontend code.

# References

Fadatare, R. (2018, April 14). *Repository Pattern*. Retrieved from ramesh-java-design-patterns.blogspot.com: https://ramesh-java-design-patterns.blogspot.com/2018/04/repository-pattern.html

tutorialspoint. (2019). *Design Pattern - Factory Pattern*. Retrieved from www.tutorialspoint.com: https://www.tutorialspoint.com/design\_pattern/factory\_pattern.htm

tutorialspoint. (2019). *Design Pattern - Singleton Pattern*. Retrieved from www.tutorialspoint.com: https://www.tutorialspoint.com/design\_pattern/singleton\_pattern.htm

tutorialspoint. (2019). *Design Patterns - MVC Pattern*. Retrieved from www.tutorialspoint.com: https://www.tutorialspoint.com/design\_pattern/mvc\_pattern.htm