University of Technology, Jamaica

Faculty: Engineering and Computing

School: School of Computing and Information and Technology

Academic Year: 2018/19

Semester: 2

Module: CIT 3009 Advance Programming

*Research Group Project for John Shop*

Group Members:

Duran Thomas - 1401134  
Alicia Harriott – 1104334

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Tutor: Mr. Gilroy Gordon

Table of Contents

[ERD Diagram 3](#_Toc6539045)

[Class Diagram 4](#_Toc6539046)

[S.O.L.I.D Design Principles 5](#_Toc6539047)

[Repository Pattern 6](#_Toc6539048)

[Model-View-Controller (MVC) Pattern 8](#_Toc6539049)

[Singleton Pattern 10](#_Toc6539050)

[Factory Pattern 11](#_Toc6539051)

[Code Generation Tool 12](#_Toc6539052)

[Source Control Management Tool 13](#_Toc6539053)

[Reference 14](#_Toc6539054)

## ERD Diagram

**Orders**

Views

Items\_Ordered

Generates  
from

Monitors

Supplier

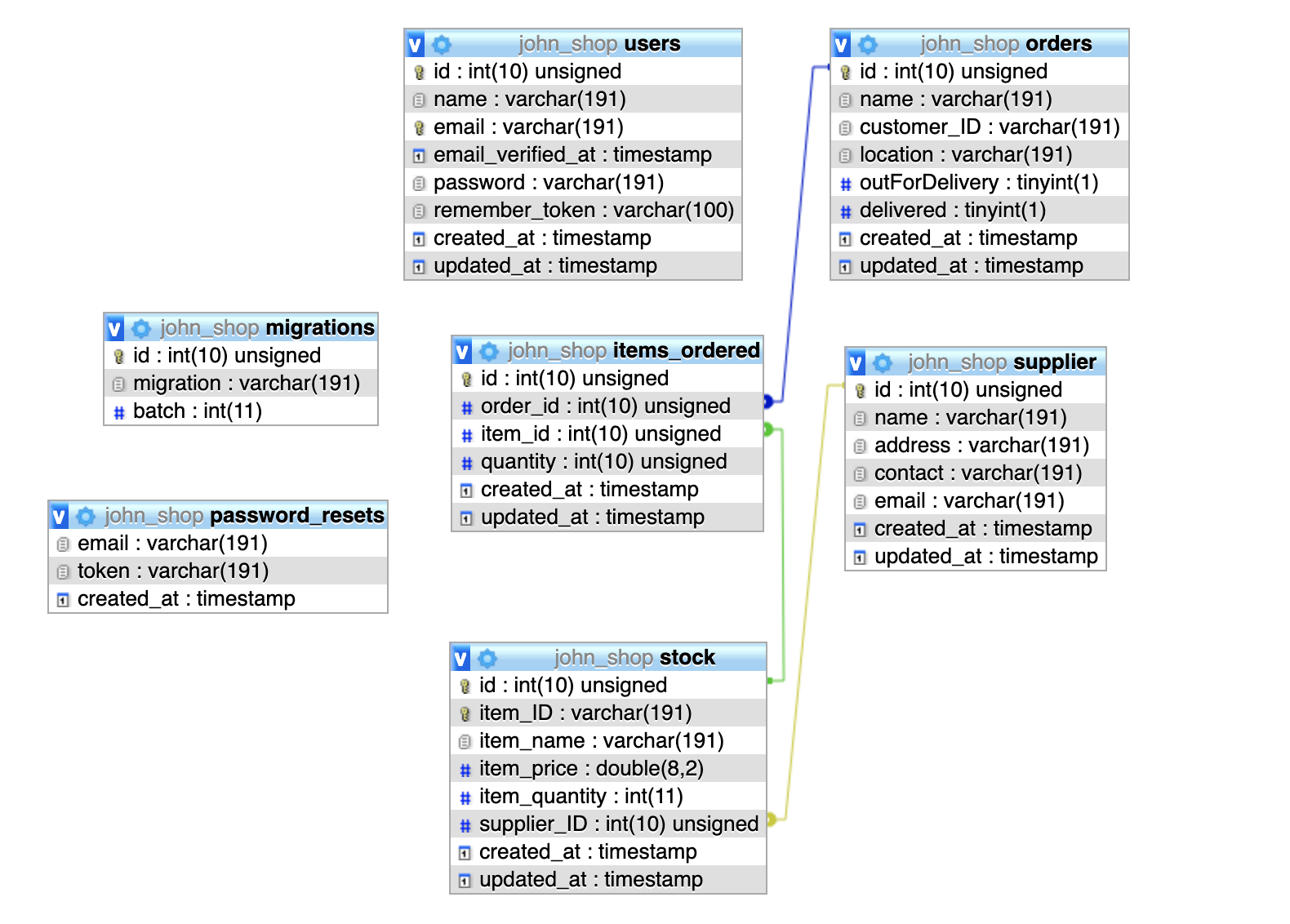
Purchase  
from

Administrator

Stock

Customer

## Class Diagram

****

## S.O.L.I.D Design Principles

What are these?

These principles, when combined together, make it easy for a programmer to develop software that are easy to maintain and extend. They also make it easy for developers to avoid code smells, easily refactor code, and are also a part of the agile or adaptive software development.

When expanded the acronyms might seem complicated, but they are pretty simple to grasp.

* **S** - Single-Responsibility principle – this principle states that a class should have one and only one responsibility.
* **O** - Open-closed principle – this principle is saying that a class should be open for extension but closed for modification.
* **L** - Liskov substitution principle – Objects in a program should be replaceable without altering the correctness of the program.
* **I** - Interface segregation principle – This principle is saying that many different interfaces is better than one general interface.
* **D** - Dependency Inversion Principle – This is saying that one should be dependent on abstract classes and not concrete ones.

## Repository Pattern

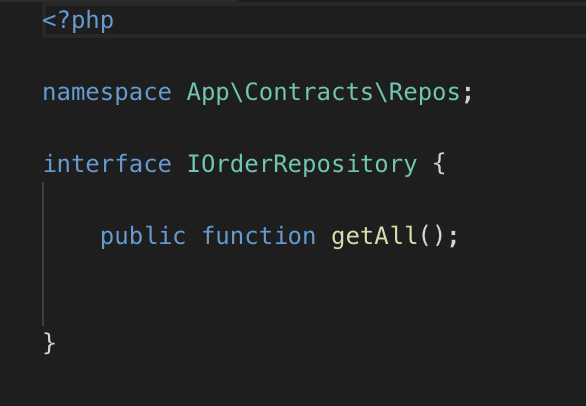
Repository pattern is one of the most popular patterns used to create large scale applications. It creates layers for database operations while restricting us from working directly with the data in the application. Advantages of working with repository pattern may include:

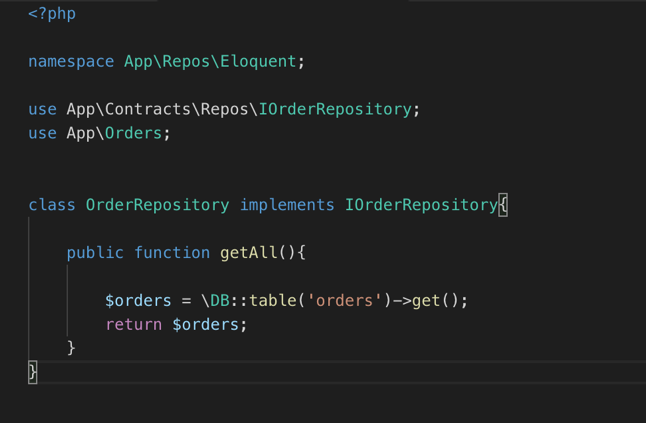
* Database access code is centrally managed which makes it easy to implement database access policies.
* Database access codes can be reused.

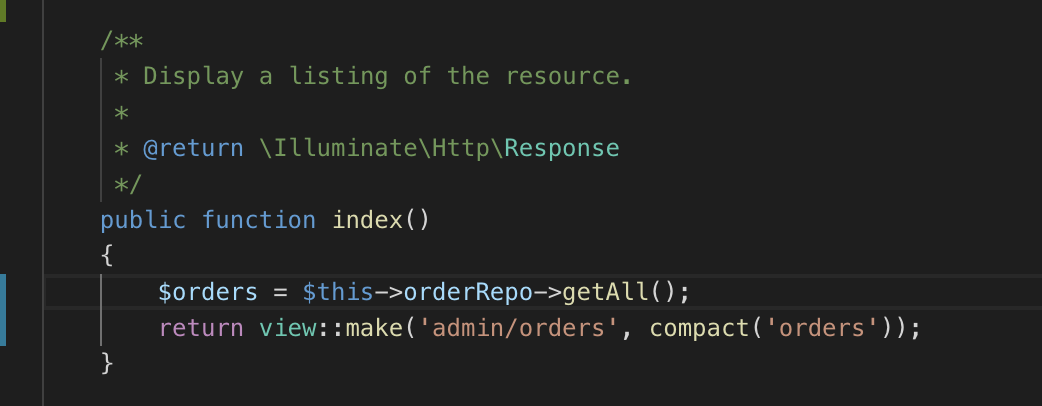
Applications that don’t follow repository patterns may suffer from:

* Duplicate database operations code.
* Need for external dependencies to unit test business logic.

An example of Repository Pattern in our application can be seen in the following screenshots.

****Figure 1.

****Figure 2.

****Figure 3.

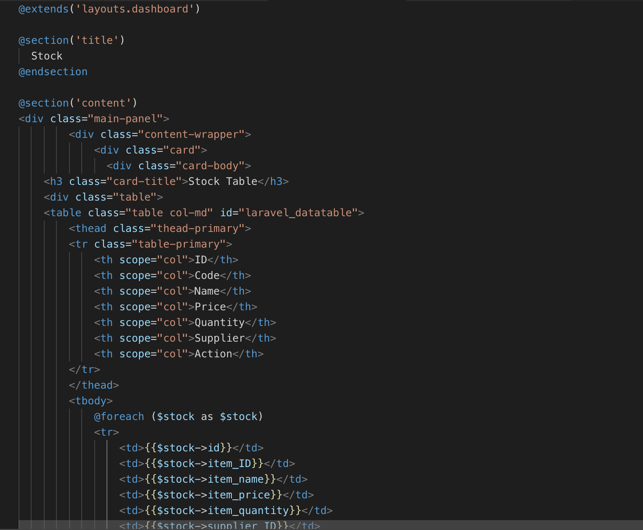
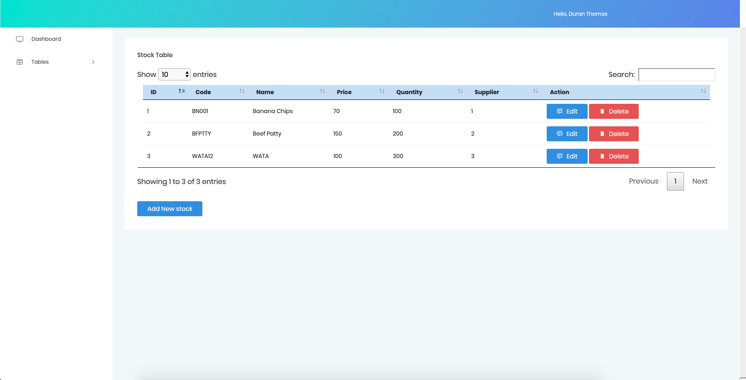
## Model-View-Controller (MVC) Pattern

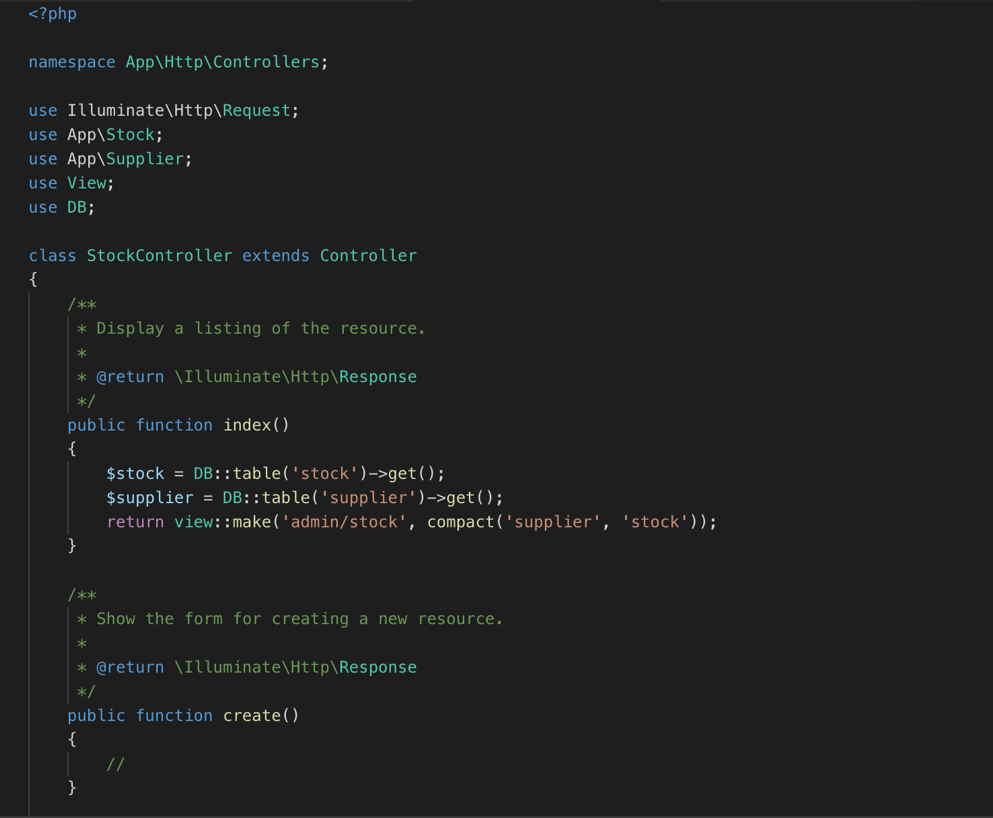
MVC pattern is used as a successful and efficient way to relate the user interface with the underlying data models. This pattern consists of three parts:

1. The Model: The model represents the object which carries the data.
2. The View: The view represents the visualization of the data that the model contains.
3. The Controller: The controller acts on both the model and the view, it controls the flow of data into the model and also updates the view whenever there are changes to the data.

Example of the MVC Pattern in our project:

  
Figure 4: The Model

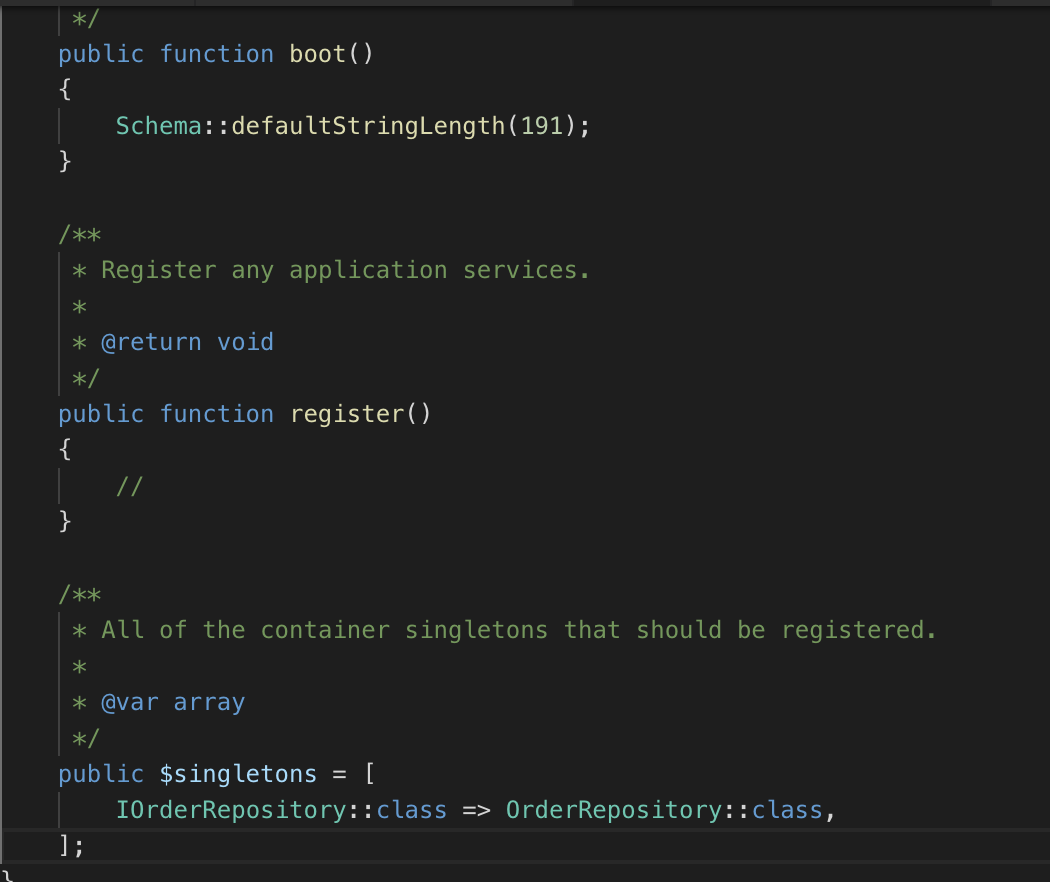
   
Figure 5: The View

****Figure 6: The Controller

## Singleton Pattern

The singleton pattern is a design pattern that restricts the instantiation of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system.

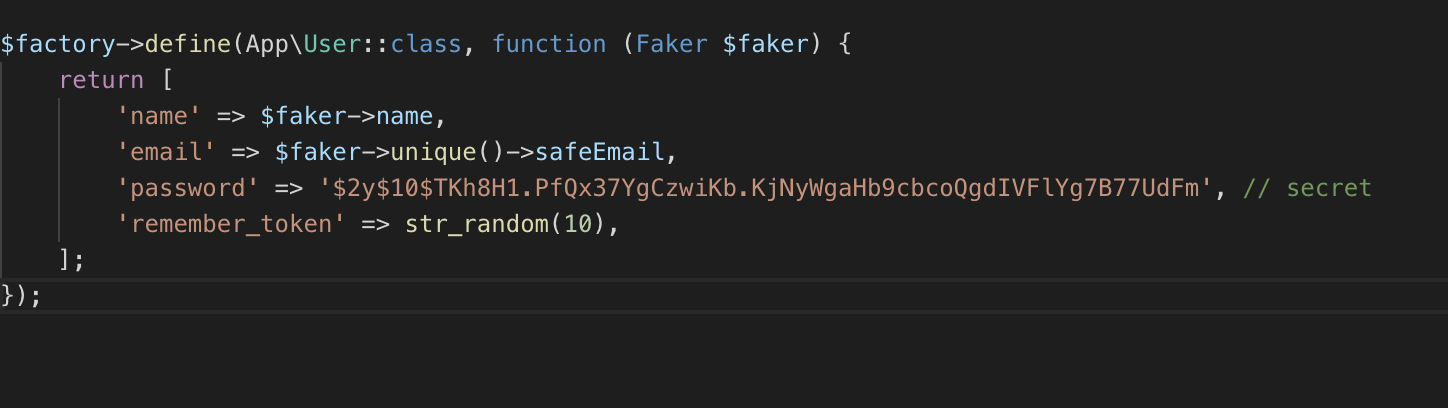
Example of singleton in our application:

  
Figure 7.

## Factory Pattern

Factory pattern can be defined as a simple design pattern that give us a convenient way to instantiate objects.

Factory pattern in our app is used to create a default admin account.

  
Figure 8.

## Code Generation Tool

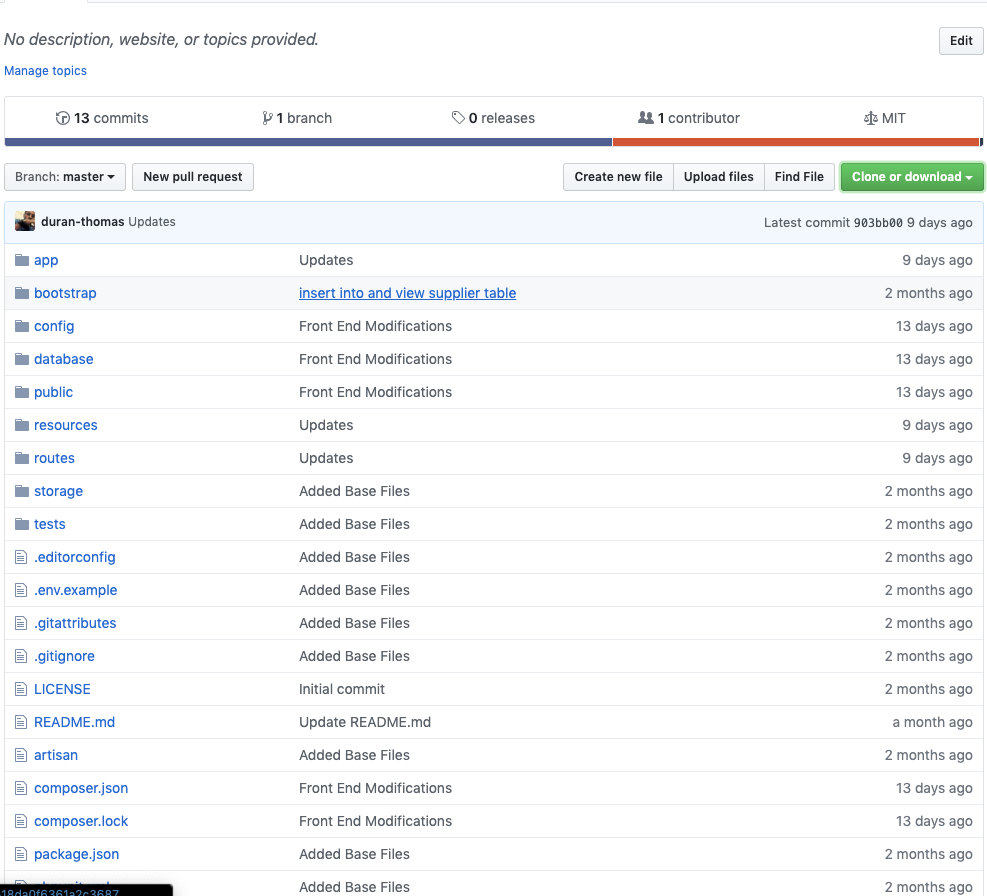
Code generation tool that was used in this app is composer. Composer was used to generate the basic codes required to start a Laravel project.

## Source Control Management Tool

For our source control, we utilized services from GitHub.

Here is the link to our GitHub repository along with a screenshot of the repo.

Link: https://github.com/duran-thomas/John-Shop.git



## Reference

Learning Laravel from scratch retrieved from<https://www.youtube.com/results?search_query=laravel+from+scratch>

YouTube Video Of Demo: https://youtu.be/QTgnBF6xXFs