Advanced Programming CIT3009

University of Technology, Jamaica

Name: Shavaun Mitchell ID: 1608207

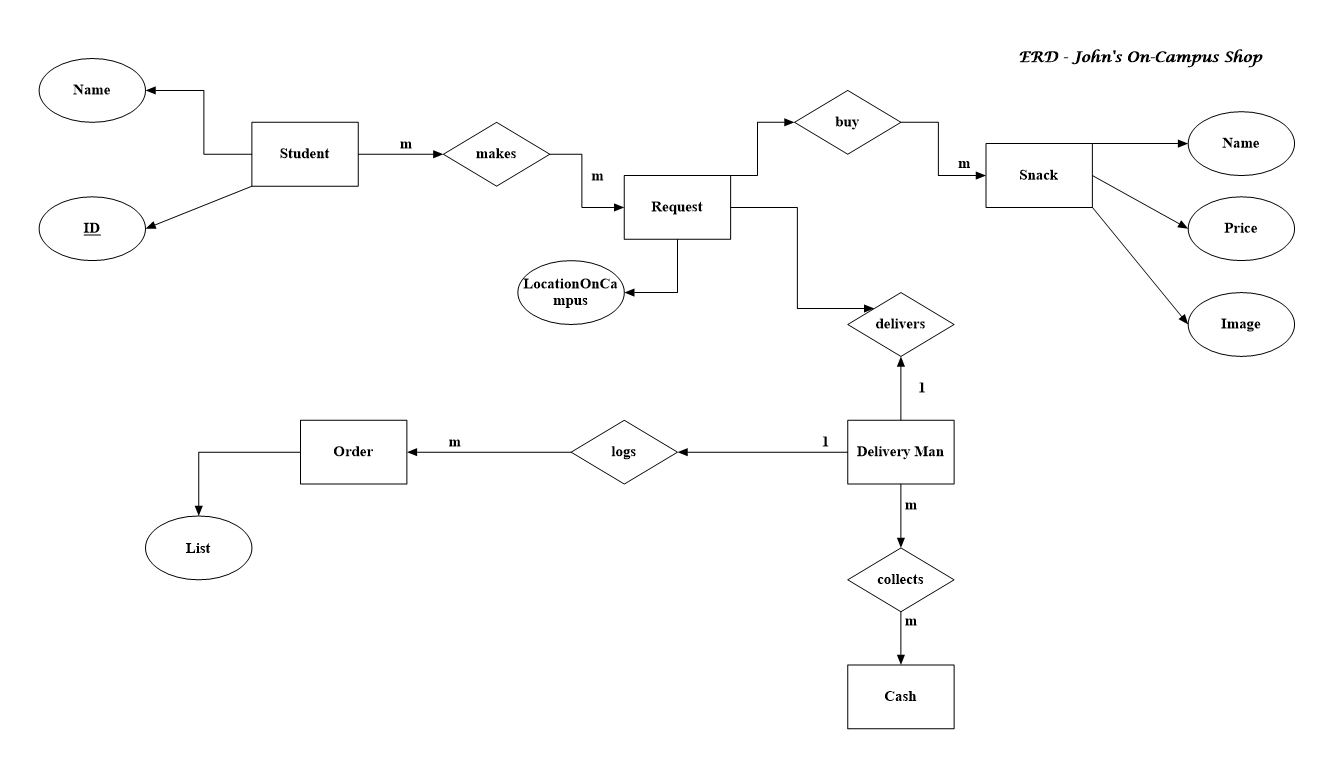
Name: Dominique Griffiths ID: 1608205

Tutor: Mr. Gilroy Gordon

Semester: 2019/2

April 11, 2019

**ERD - Entity Relationship Diagram**



**Wire Frame**

Footer

Sign Up

Context

Admin Login

Context

Background Picture

Contact

Request

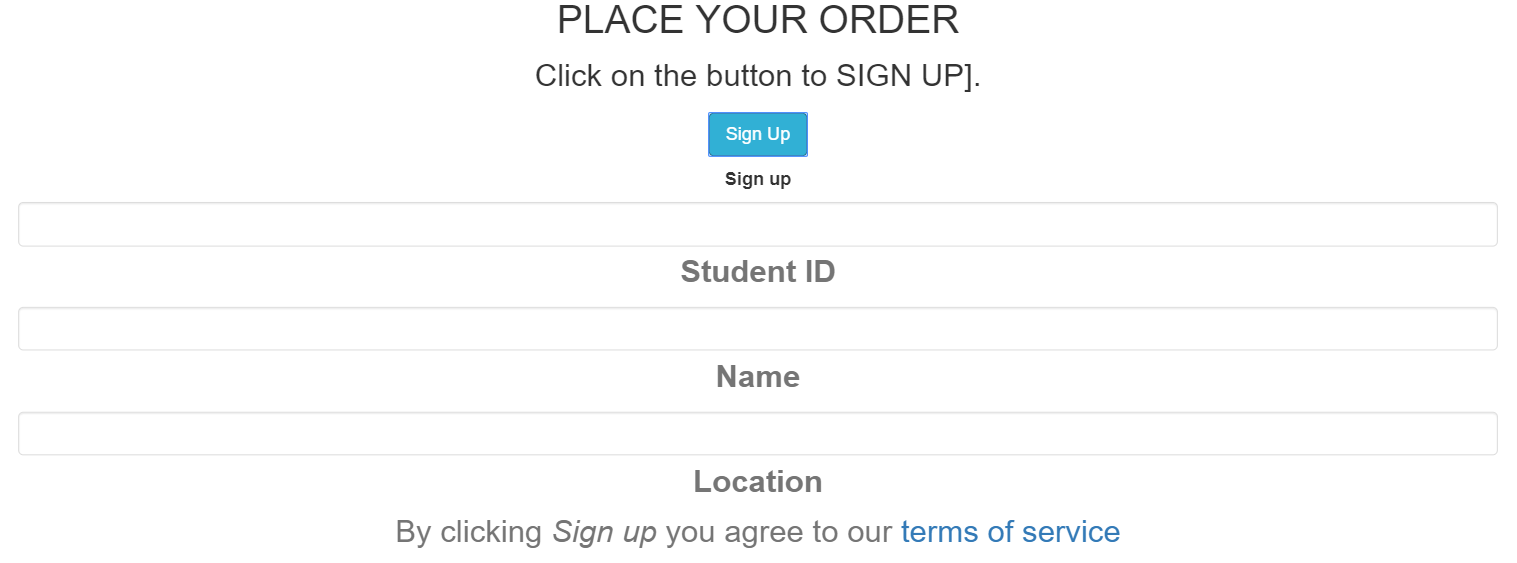
About Us

Home

**Demonstrate the use of SOLID principles**

**S – Single Responsibility Principle -** An entity (class/method) should only have ONE responsibility.

The Login to the John’s Shop App is said to be a single responsibility principle because it carries out one task which should enable the user to enter the software application. User uses this method to enter grocery/snack section in order to place their order. Also each main action that need to be done by the app is done by specific controller each controller doing only one job for instance my stack controller only add snacks to the cart, my request controller only gets the requests from the database etc.



**O – Open/Closed Principle -** Open for extension, closed for modification.

In the database and service entity you can add more functionalities.

More variables or extended functionality can easily be added to our database classes with little no effort however if anything is changes/modified it will be much harder to do so.

In all my controllers and factory other functionalities can easily by added by just adding an extra function to them however if one were to change how the factory Controller works that would be much more difficult so its open to extension but not to modification

**L – Liskov Substitution Principle -** Objects in a program should be replaceable without altering the correctness of the program.

Classes created by extending our class should be able to fit in application without failure. This requires the objects of your subclasses to behave in the same way as the objects of your superclass. This is mostly seen in places where we do run time type identification and then cast it to appropriate reference type. This is show case in our backend within our DB Controllers where they both extend the entities and changing these entities would not affect the correctness of the code.

**I – Interface Segregation Principle -** Many specific interfaces are better than one general interface.

**D – Dependency Inversion Principle -** One should depend on abstractions not concrete types.

These separate components are so well closed in their boundaries that we can use them in other software modules apart from spring with same ease. This has been achieved by dependency inversion and open closed principles. All modules expose only abstraction which is useful in extending the functionality or plug-in in another module.

**Demonstrate the use of the Repository Pattern**

The repository pattern is intended to create an abstraction layer between the data access layer and the business logic layer of an application. It is a data access pattern that prompts a more loosely coupled approach to data access. We create the data access logic in a separate class, or set of classes, called a repository with the responsibility of persisting the application's business model.

This was demonstrated where we use a generic class to ascertain all database connection and each database class inherits from it also we used the repository pattern in our all request and all snacks service where these provide all the request and all the snack data to any controller that might need it.

**Demonstrate the use of the Singleton Pattern**

Allows us to maintain only one instance of a resource to be utilized by multiple objects.

* Singleton pattern restricts the instantiation of a class and ensures that only one instance of the class exists in the java virtual machine.
* The singleton class must provide a global access point to get the instance of the class.

This is demonstrated where in our snackDB and requestDB file where if there is no table, we create one however if the table already exist, we don’t create any. Also this is demonstrated on the front in our services which are automatically singleton in angularjs so is so it basically on creates one instance of each.

**Demonstrate the use of the Model-View-Controller Pattern**

[AngularJS](https://www.upwork.com/hiring/development/writing-job-description-find-great-angularjs-developer/) is a full-featured JavaScript framework, with the aim of**simplification making it easier to create an open-source, front-end JavaScript framework**. It excels at building dynamic, single page web apps (SPAs) and supports the Model View Controller (MVC) programming structure. An example of the model view control is where we have the database and objects for snacks and request as our model when then have controllers for each database that interacts with them to perform crud operations and also in the front end where we have controllers interact with the object in order to push what is needed to our view.



**Demonstrate the use of the Factory Pattern**

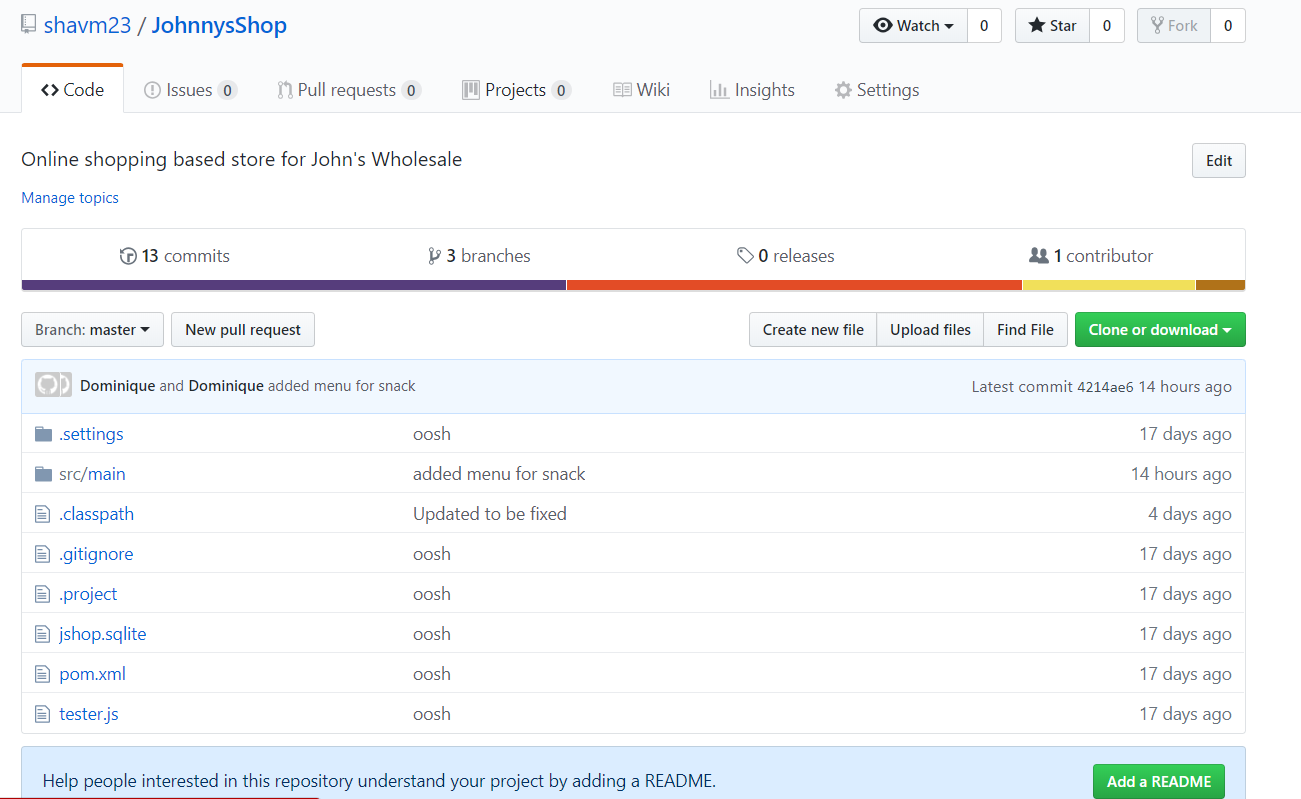
JDBC - Java Database Connectivity

An example of factory patter is a in our backend where we have a controller that creates the database objects for snack and request.

1. Factory design pattern provides approach to code for interface rather than implementation.
2. Factory pattern removes the instantiation of actual implementation classes from client code. Factory pattern makes our code more robust, less coupled and easy to extend. For example, we can easily change PC class implementation because client program is unaware of this.
3. Factory pattern provides abstraction between implementation and client classes through inheritance. (Pankaj, 2019)
4. We used the factory pattern in our front end to automatically create cart items with a function add so that we could use this function at anytime in our cart item object

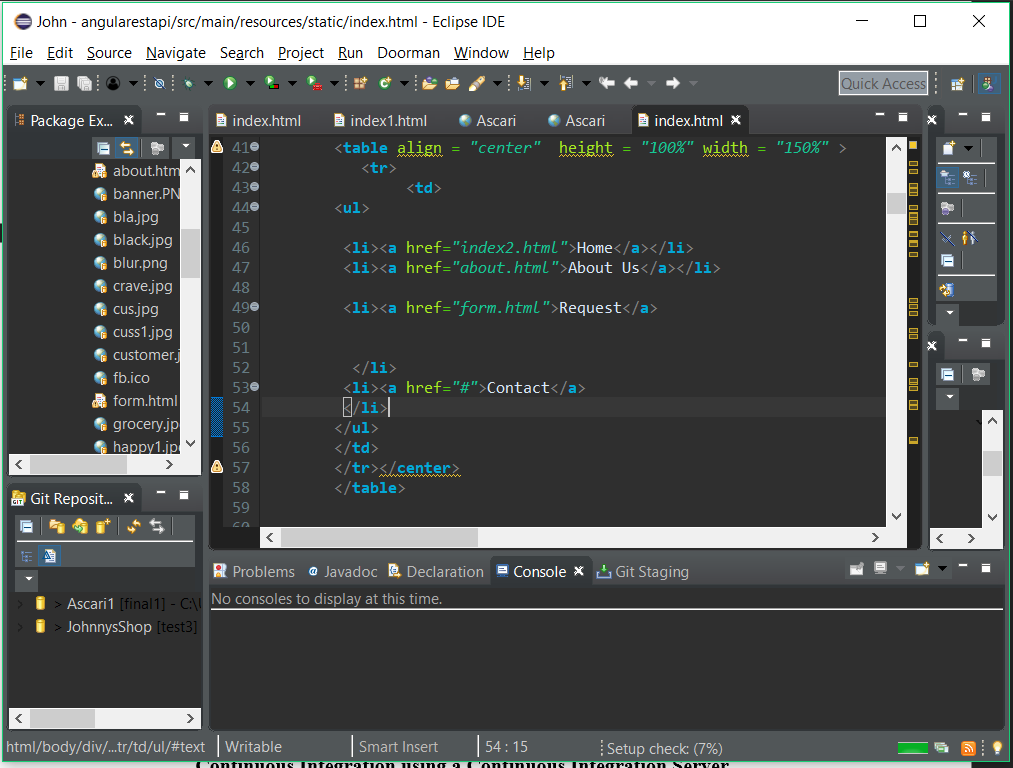
**Demonstrate the use of Source Control Management Tool**

GitHub was used as a source control management tool by the developers to create and code an app. We chose this management tool because when there are multiple people collaborating on a project, it’s hard to keep track to see who changed what, when, and where those files are stored. GitHub takes care of this problem by keeping track of all the changes that have been pushed to the repository.



**Demonstrate the use of a Package Management Tool**

Eclipse is an integrated development environment (IDE) that has been designed from the ground up for building integrated web and application development tooling. By design, the platform does not provide a great deal of end user functionality by itself. It was used because the usage of an IDE (like Eclipse) in Automation Testing is to build framework libraries and modules that are easily accessible and provides a better view of the project-framework.



**Demonstrate Unit Testing and Test Automation**

Testing is the process of checking the functionality of an application to ensure it runs as per requirements. JUnit testing comes into picture at the developers’ level; it is the testing of single entity (class or method). JUnit testing plays a critical role in helping John’s Shop software to be delivered out efficient to its client to ensure that everything is working.

This was demonstrated by using junit to test our DB classes.

**Continuous Integration using a Continuous Integration Server**

Developers practicing continuous integration merge their changes back to the main branch as often as possible. The developer's changes are validated by creating a build and running automated tests against the build. By doing so, you avoid the integration stress that usually happens when people wait for release day to merge their changes into the release branch (Pittet, n.d.).Continuous integration was demonstrated through the use of GitHub where the project was continuously improved up on a regular basis.

**YouTube Link**

This YouTube link demonstrates the way in which the application is executed.

<https://www.youtube.com/watch?v=b1T_nBSYqxE&list=PLv-lvl4AcMQiD02IygYOsqL4Cc4r839gY>

# References

Pankaj. (2019). *JornalDev*. Retrieved from www.journaldev.com: https://www.journaldev.com/author/pankaj

Pittet, S. (n.d.). *Continuous integration vs. continuous delivery vs. continuous deployment*. Retrieved from Atlassian CI/CD: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

Feature Review  
  - As a student, I may place an order for several snacks with my student ID, first name, last name and location.  
  - As a rider or owner I may access a list of orders made by students  
 Code Review (will be conducted from your online repository)  
- Demonstrate the use of SOLID principles  
- Demonstrate the use of the Repository Pattern  
- Demonstrate the use of the Model-View-Controller Pattern  
- Demonstrate the use of the Singleton Pattern  
- Demonstrate the use of the Factory Pattern  
-   
Tool Review  
- Demonstrate the use of a Code/Document Generation Tool  
- Demonstrate the use of Source Control Management Tool  
- Demonstrate the use of a Package Management Tool  
- Demonstrate Unit Testing and Test Automation  
- Continuous Integration using a Continuous Integration Server

Documentation Review  
- ERD   
- Wireframe  
- APA formatted report  
        -Contents  
        - APA formatting  
- YouTube Video

Quality Review

Code and Presentation