**Storefront**

Benjamin Kwon

CSC-17B-44085

June 3, 2019

**Table of Contents**

1. Introduction..........................................................................................................1

2. Description............................................................................................................1

3. Folder Layout........................................................................................................7

4. Concepts...............................................................................................................8

5. Diagrams.............................................................................................................14

6. Code/GitHub Link................................................................................................15

**1. Introduction**

The purpose of this project was to create a functional online shopping website containing a shopping cart. The website also contains separate user and admin interfaces. HTML, CSS, JavaScript, PHP, and MySQL queries were used to create the front and back end of the website.

**2. Description**

**Storefront**

This project simulates a website dealing in e-commerce. The theme for this storefront is related to fashion. As such, products such as t-shirts, shirts, pants, accessories, and more are sold. In its implementation, as many functionalities possible were included such as sorts, categories, streamlined visual cues, and more. Products can be sorted by categories and by price, both ascending and descending.

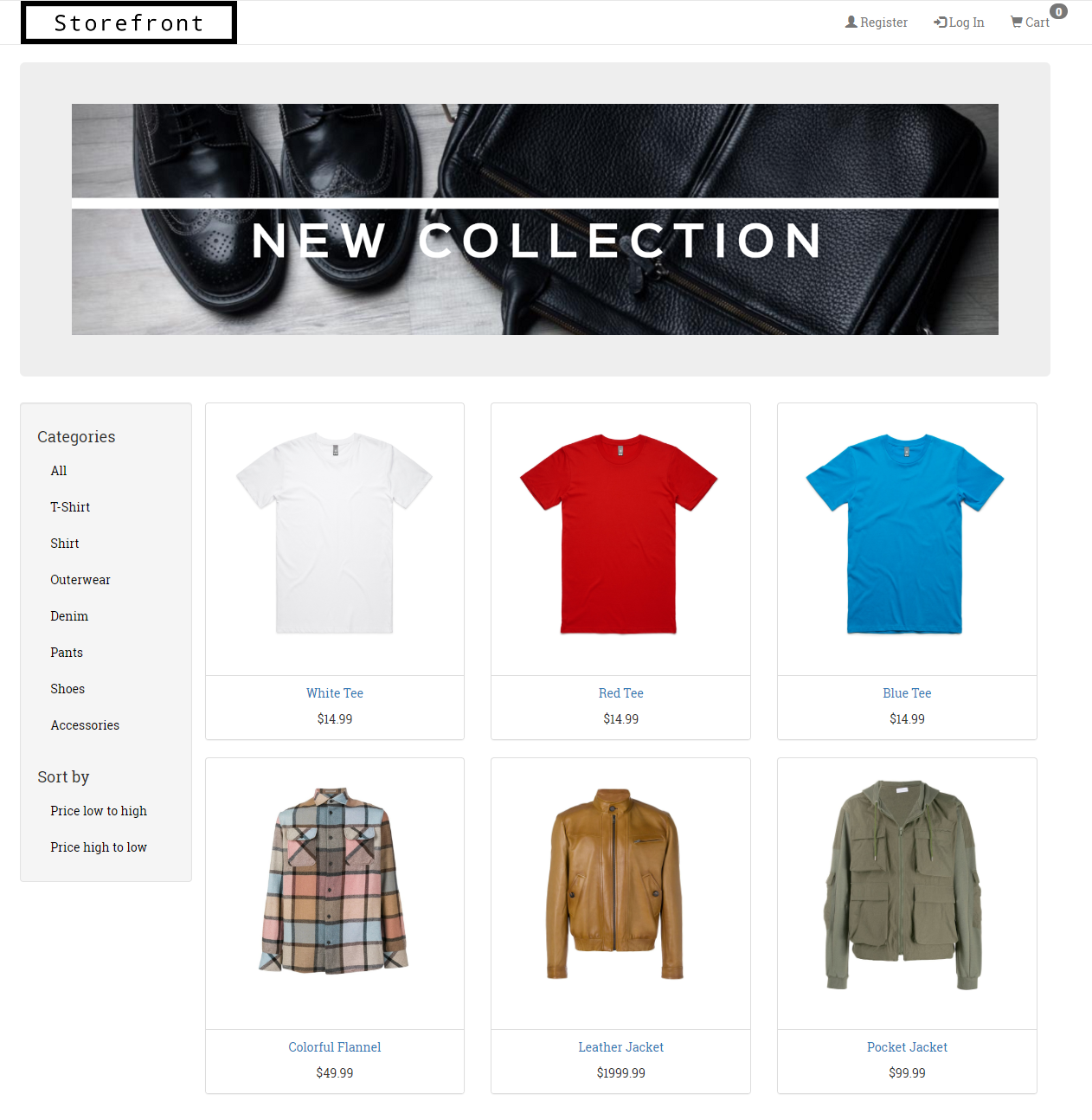
On this website, the user can browse products and place them into a shopping cart. Afterwards, the user can checkout these items and place an order. In order to place an order, a user must log in to the system first. These orders are stored in a database and can be later seen in the user’s account page.

While users can shop and place orders, admins have more functionality regarding database manipulation. Admins can add products to the store, add product variants (different sizes and quantities of the same product), edit products, edit product variants, delete products, and delete product variants. In addition, they can modify and delete users stored in the database as well (see screenshots below).

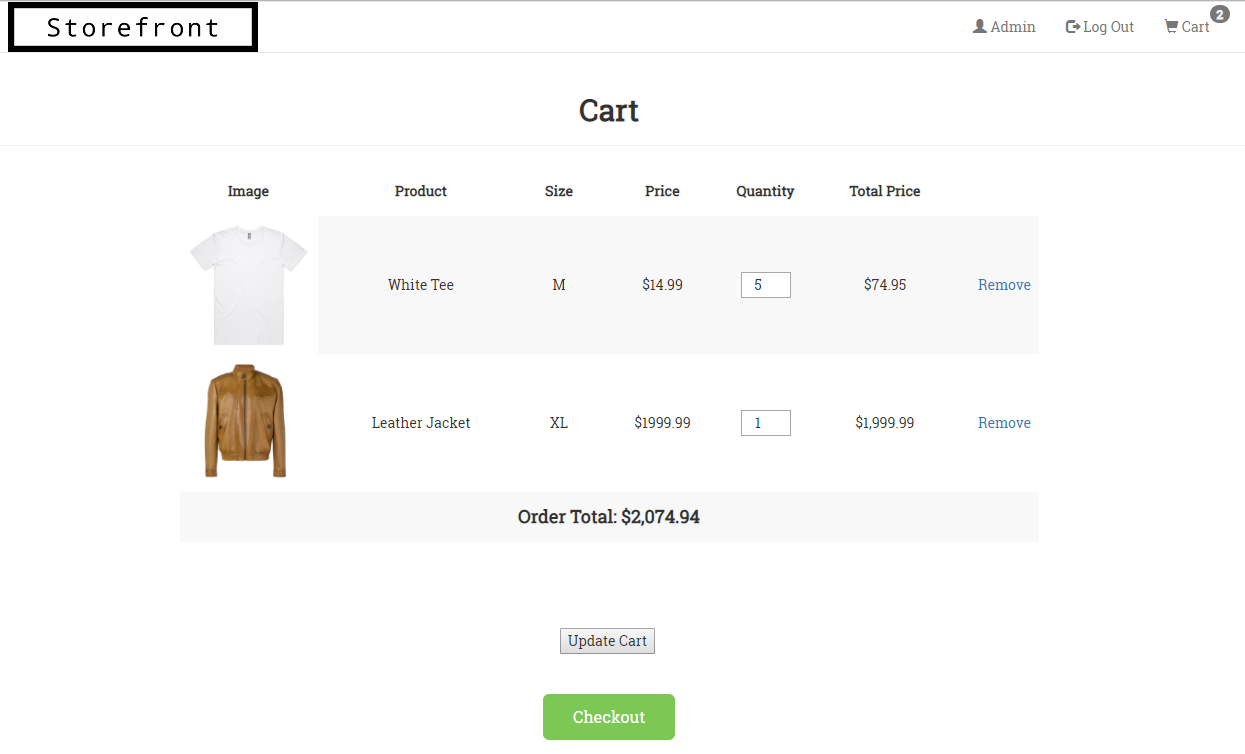
By default, the database is prepopulated with data including several products and users. The following accounts can be used if needed. Otherwise, new accounts can be created as either admins or regular users.

*Table 1. Pre-stored accounts with their email and password*

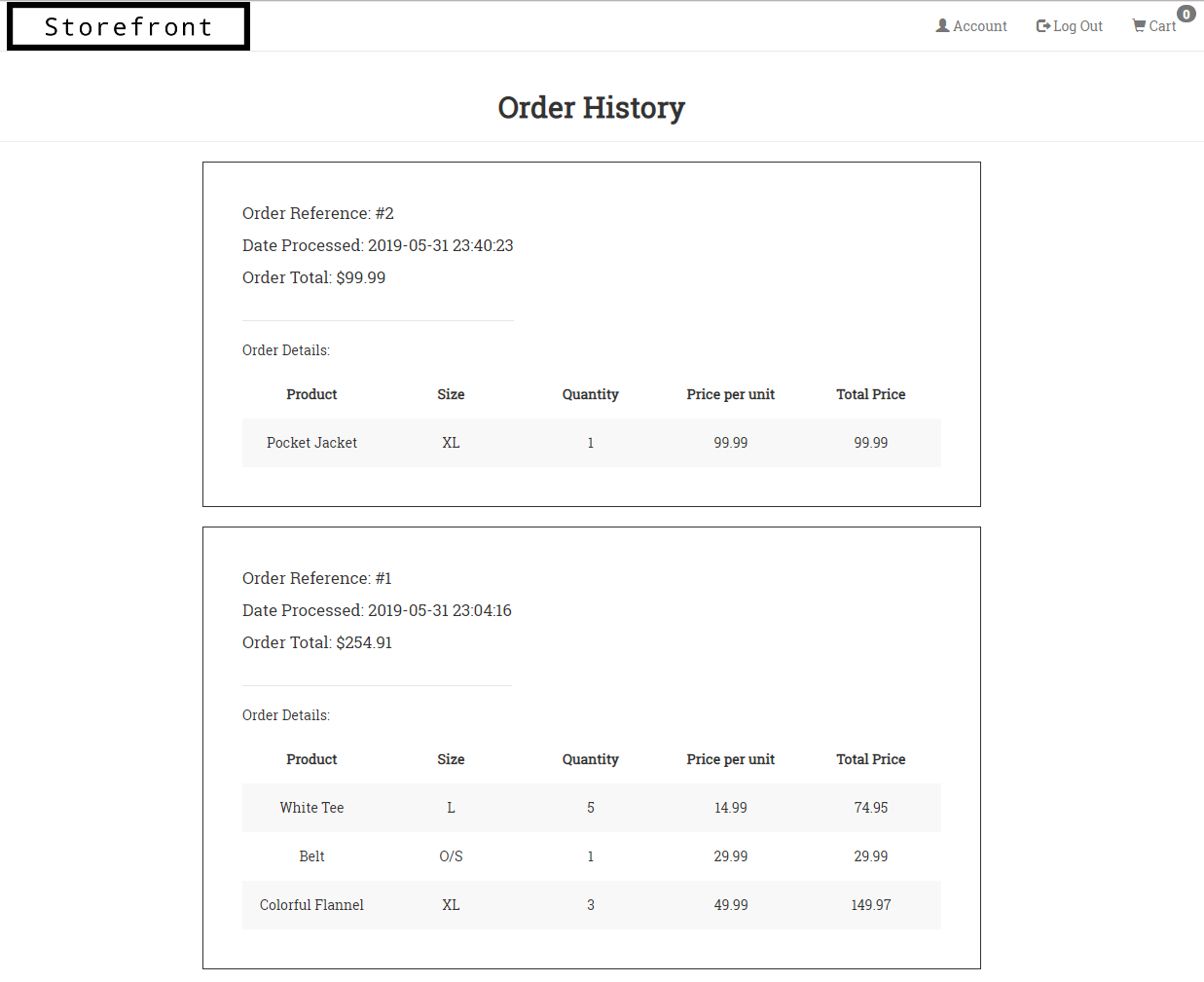
|  |  |  |
| --- | --- | --- |
| **User E-mail** | **Password** | **Role** |
| admin@storefront.com | admin | Admin |
| bkwon1@student.rccd.edu | Pass123! | User |
| user1@email.com | Pass123! | User |
| user2@email.com | Pass123! | User |
| user3@email.com | Pass123! | User |
| user4@email.com | Pass123! | User |
| user5@email.com | Pass123! | User |

*A view of the home page where users can browse the product catalog.*

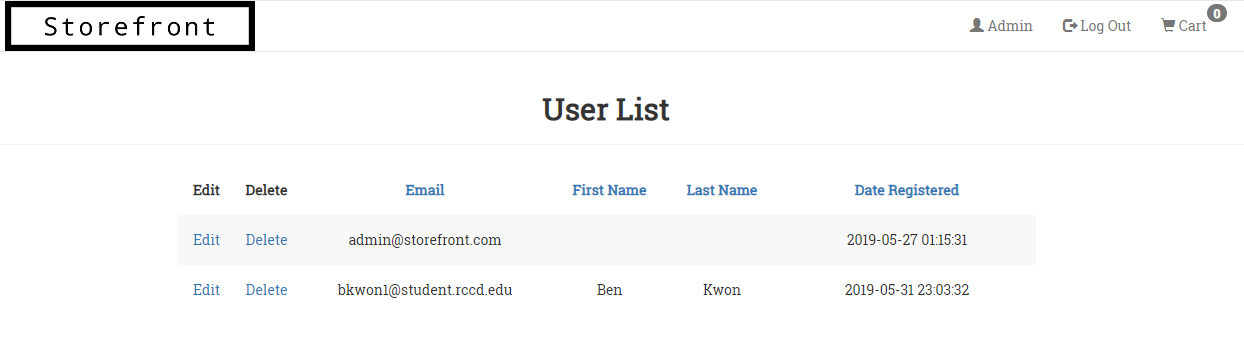
*Six products per page are displayed.*

****

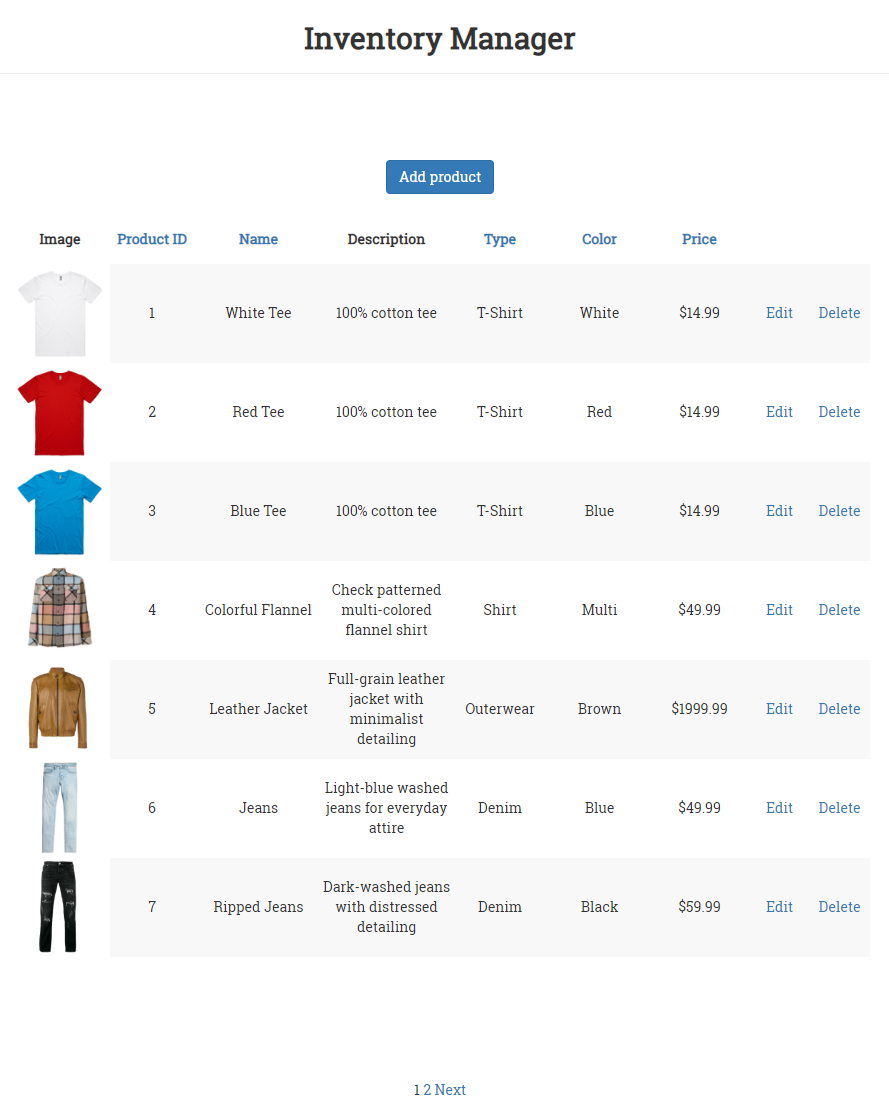
*A view of the cart after the user adds products. The order total is calculated here.*

**

*An example of a user’s order history.*

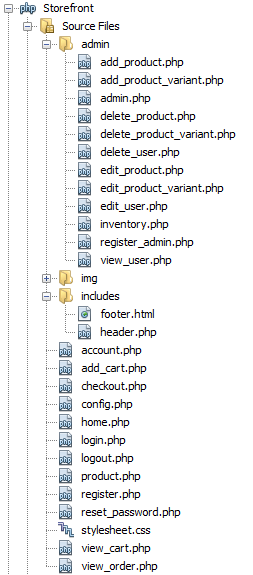
****

*Admins can modify and delete users from the database.*

****

*A view of the inventory manager page. Here, the admin can add, edit, and delete products and their variants.*

**3. Folder Layout**

****

Pages that are specific to admins are in a separate folder titled ‘admin.’ The folder ‘img’ contains any images used in the website including the banner, logo, and the pictures for each product. The folder ‘includes’ contains the header and footer for pages while the rest of the files are pages that can be accessed by any user.

**4. Concepts**

**Model-View-Controller**

*Model*:

The ‘storefront’ database contains tables representing data including products, product variants, order items, and orders. User information including emails, names, and passwords are also stored. After receiving a command, the model updates, which can then be displayed by the view.

*View*:

The view allows the visualization of the model state. Through HTML, CSS, and JavaScript, the user can see the data displayed from the model in a presentable way. For example, admins can see a maintained list of the product inventory as well as information from registered users directly from the database.

*Controller*:

By reading user inputs such as keyboard inputs, PHP can take the data from methods such as HTTP GET and POST requests. The data can then be manipulated and/or stored into the model through commands. Thus, the controller manages any user interaction with the model. Information such as products placed in the cart by the user is sent to the database when an order is placed.

**Objects**

File location: /Storefront/product.php (line 19-55)

/Storefront/add\_cart.php (line 7-11)

Using JavaScript class syntax, a Product class is constructed when a user chooses to add a product to their cart. The class includes variables that stores the products variant identification number and the quantity chosen. After converting this information to a JSON string, it is placed inside a cookie. The cookie is then passed into, decoded, and placed in a PHP object.

**Reading/Writing files/JSON**

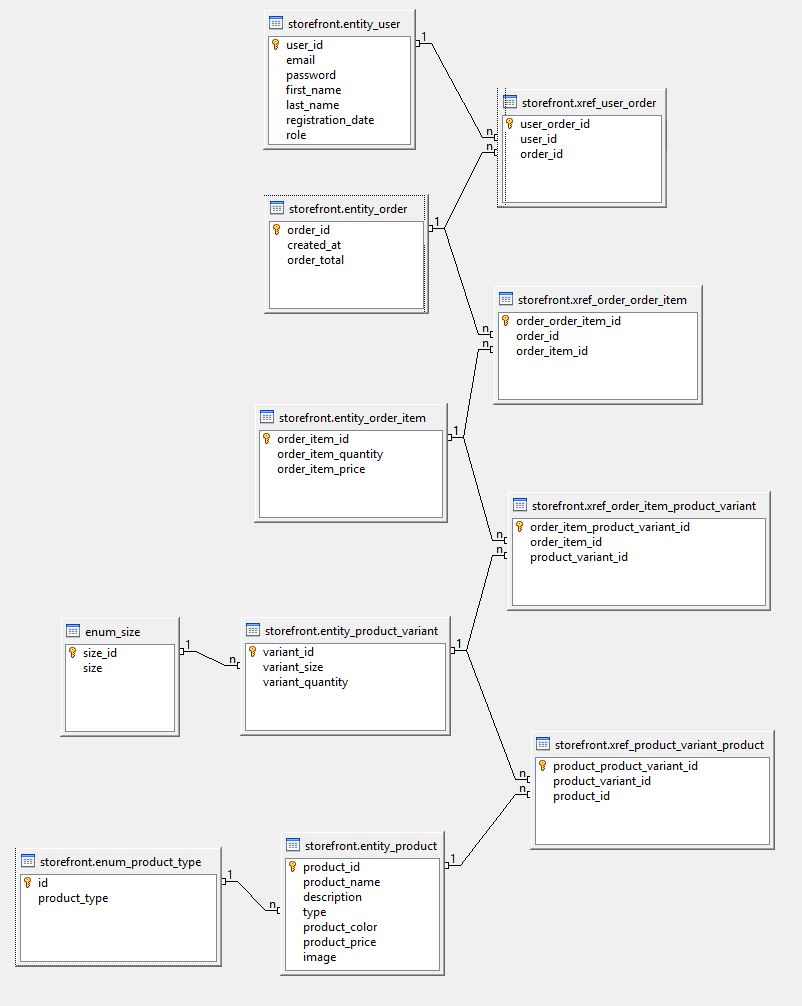
File location: /Storefront/product.php (lines 45-48)

/Storefront/add\_cart.php (line 7)

When a user adds an item to their cart, an object of the Product class is created. After information regarding the product is stored, JSON is used to stringify the object. This string is then stored into a cookie to be read and decoded later.

**Database SQL**

Diagram of database tables (entities, xrefs, and enums):



**Form Validation – Regular Expressions**

File location: /Storefront/register.php (lines 10-23)

Regular expressions were used for input validation when a user registers an account on the website. Details are as follows:

Email – Must match syntax of an email address including the @

Password – 6 to 20 characters that includes at least one character of each of the following: digit, lower case character, upper case character, symbol

**User-Admin-Login**

File location: /Storefront/admin/register\_admin.php (Register admin account)

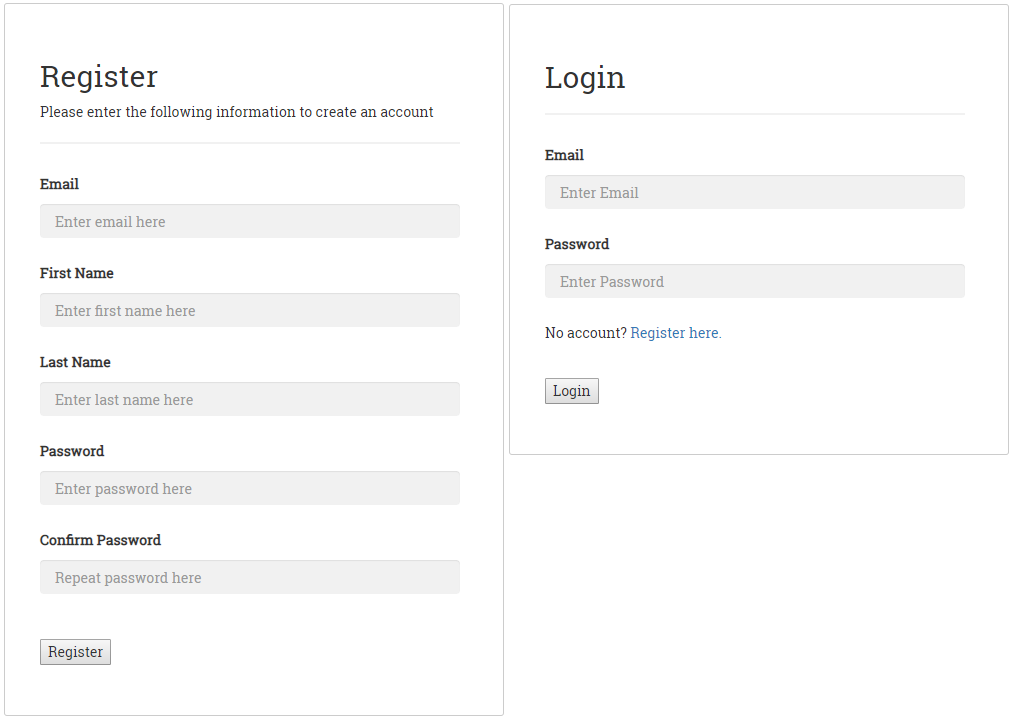
/Storefront /register.php (Register user account)

/Storefront /admin/admin.php (Admin panel)

/Storefront /account.php (User panel)

/Storefront /login.php (Login page)

When no user is logged in, a button for registering an account as well as logging in is displayed on the top navigation bar. When logged in, the register and login buttons disappear and instead show a button that directs the user to their main account page. Here, users can view orders they have placed or reset their password. On the other hand, admins can view the admin panel which grants access to the product inventory and the ability to view and modify any user account.

*A view of the login and register pages*

The `entity\_user` table inside the storefront database contains both users and admins. The difference between the two is the role column, which signals whether the user is an admin or not. Admins have a wider functionality access such as the ability to directly manipulate database items as well as user information. Users only have viewing access to their orders stored in the database

*Table 2. Roles of the two different types of users*

|  |  |
| --- | --- |
| **Admin** | **User** |
| Add products and product variants  Delete products and product variants  Edit products and product variants  Edit users  Delete users | Place orders  View orders |

**Cookies-Sessions-Securing Pages**

File location: Cookies: /Storefront/product.php & /Storefront/add\_cart.php

Sessions: Most files in project

Securing Pages: Most files in project

In this project, cookies are used to store the product object after it has been converted to a JSON string. This cookie is then decoded into a PHP object. This information is used to send the product information to a session variable named cart. This will hold the information of each product in the users ‘shopping cart.’ The information here is used to calculate cart totals and for the confirmation of an order if the user decides to place an order.

Sessions are used on many pages because of the nature of the website. These variables hold information for easy tracking such as the user’s identification number. The storefront needs continual access to the data stored in the database and many connections are started and closed when accessing different pages.

All pages are secured using sessions as well as valid GET or POST parameters. Depending on the page, users will be redirected to another more appropriate page or be denied access. For example, if a user wants to place an order, the session will check if he or she is logged in. If not, the user will be redirected to the login page. If a user tries to get into pages only intended for admins, they will be denied access and the script will terminate.

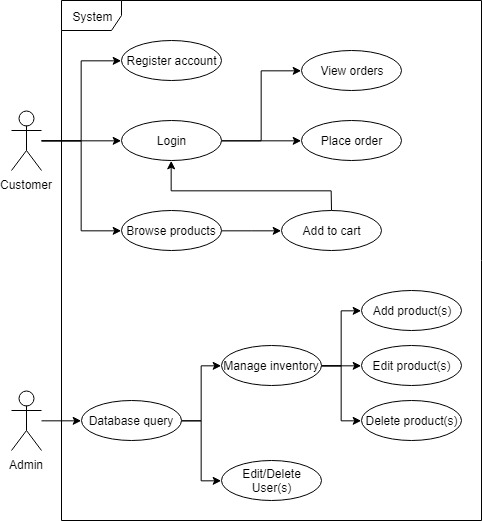
Pages that only grant access to admins are secured by using a session that tracks whether the user has admin privileges. Most of these session variables are set when a user logs in to the system.

**Parallel processing – pthreads**

Unfortunately, pthreads were not implemented into this version of the project. But the idea was entertained on how the concept can be used where multiple threads can be created to do their own separate task. For example, multiple different queries can be sent to and received from the MySQL database at the same time. This concept can be used wherever you need parallelism whether it be due to efficiency or when communication between threads are needed.

**5. Diagrams**

UML Use Case Diagram



UML Class Diagram



**6. Code**

The source code for each file of this project can be found at the following GitHub link: <https://github.com/bhknx3/CSC-17B/tree/master/Storefront>

OR

The code can be viewed in the file labeled Storefront\_Code.docx.