Final Project Report

Shopaholic

Jasmine Lao - (015465723)
(jasmine.lao@sjsu.edu)

Phuong Huynh - (015996955)
(nguyenlanphuong.huynh@sjsu.edu)

Gael Gil - (016107416)
(gael.gil@sjsu.edu)

August 11, 2023

CS 157A — Section 62

Table of Contents

1. Goals and Description of Application

Our web application, called Shopaholic, is an e-commerce website in which users can shop for products and merchants can upload their products to the website. In total, our web application has three types of users: users, merchants, and admin.

On our first page of the website, there will be a welcoming page where all types of users can sign in. All users have an ID, first name, last name, user name, and password. After entering these details, the user can select which type of user category they belong to. Upon successful login, the user will encounter a home page where products will be displayed.

The user, merchant, and admin views differ as follows:

- Users are able to view and interact with products that are displayed on the homepage. They can add products they like to their cart and place an order.
- Merchants are able to upload different products to the Shopaholic homepage. They can also edit or delete products as they wish.
- Admin is able to delete and edit users, merchants, products, and orders.

In addition, users are able to add a product review to the review page.

Overall, the goal of our Shopaholic application is to be able to interact with the elements on the pages of Shopaholic and be able to make those updates to our backend database on MySQL.

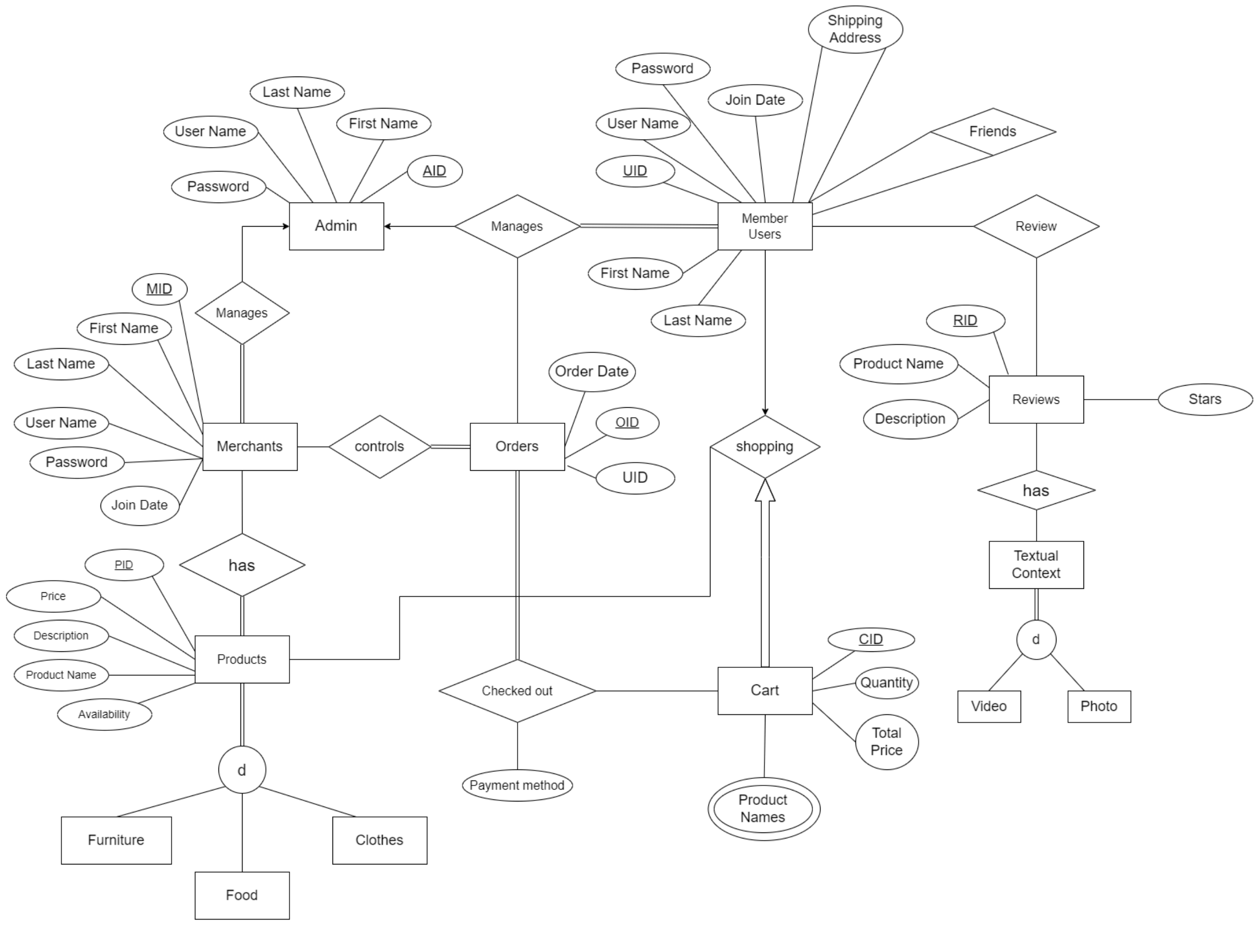
2. Application/Functionality Requirements and Architecture

Shopaholic is a simple shopping website with three types of views: users, merchants, and admin. This application allows users to create an account. They can create a user account or a merchant account. A user account can view products, add products to their cart, review products, place orders, and view order history. A merchant account can view their own products and create, edit, and delete their products.

When first launching the Shopaholic website a user will be given the login page where they will be able to create an account if they don't already have one. The two accounts available to create are merchants and users. Lastly, there is the admin account that allows someone to have complete control over the site. The admin can create, delete, and edit any user, merchant, and their products, and order history.

The project itself is a pretty straightforward web application project. We have a servlet to help with get/post requests. These allow us to manage the create, edit, and delete functions given from the front end. We also have views which are the front end of the application. This is what the users see. The MySQL tables are also accessed through the servlets. Overall, our web application follows the 3-tier architecture design: the clients can see the UI through user, merchant, or admin, the middleware is the Tomcat server and servlets, and the database is MySQL.

3. (E)ER Diagram



4. Database Design (tables and relationships)

Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update, and Deletion Anomalies.

1. 1st Normal Form
    a. Unnormalized relation

MemberUsers(UID: String, MemberUsersFriends, First Name: String, Last Name: String, Shipping Address: String, User Name: String, Password: String, Join Date: Date)

UID	MemberUsersFriends	FirstName	LastName	…
1222	1235	John	Doe	
1234	1222	Jane	Doe	
	1235	Alex	Smith	
1235	1245	Lady	Gaga	
	1222	Tom	Hank	
	1234	Taylor	Swift	

    b. Relation in 1st Normal Form

MemberUsers(UID: String, First Name: String, Last Name: String, Shipping Address: String, User Name: String, Password: String, Join Date: Date)

MemberUsersFriends(UID: String, FUID: String)
UID is Foreign Key that points to MemberUsers or GuestUsers UID.

MemberUsers table

UID	FirstName	LastName	…
1222	John	Doe	
1234	Jane	Doe	
1235	Lady	Gaga	

MemberUsersFriends table

UID	FUID
1222	1235
1234	1222
1234	1235
1235	1245
1235	1222
1235	1234

2. 2nd Normal Form

A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table. In other words, A relation that is in First Normal Form and every non-primary-key attribute is fully functionally dependent on the primary key, then the relation is in Second Normal Form (2NF).

a. Before decomposing

Merchants(MID: String, FirstName: String, LastName: String, UserName: String, Password: String, Join Date: Date, PID: String, ProductName: String, Price: Integer, Description: String, Availability: Boolean, Type: String)

Functional Dependencies:

- MID → FirstName, LastName, UserName, String, Password, Join
- PID → ProductName, Description, Type,Price, Availability

b. After decomposing

Merchants(MID: String, First Name: String, Last Name: String, UserName: String, Password: String, Join Date: Date)

MerchantProducts(MID: String, PID: String)

Products(PID: String, Product Name: String, Price: Integer, Description: String, Availability: Boolean, Type: String)

Normalization to BCNF:

Admin(AID: String, User Name: String, User Password: String)

- FD1: AID → User Name, User Password
- Since there is no partial or transitive dependency in this table, it is in 1NF, 2NF, and 3NF
- Since the LHS of FD1 is a candidate key, the Admin table is in BCNF.

MemberUsers(UID: String, First Name: String, Last Name: String, Shipping Address: String, User Name: String, Password: String, Join Date: Date)

- FD1: UID → First Name, Last Name, Shipping Address, User Name, Password, Join Date
- Since there is no partial or transitive dependency in this table, it is in 1NF, 2NF, and 3NF
- Since the LHS of FD1 is a candidate key, the MemberUsers table is in BCNF.

MemberUsersFriends(UID: String, FUID: String)

- UID and FUID are the candidate keys that have no composite key; therefore, the process is done.

Merchants(MID: String, First Name: String, Last Name: String, User Name: String, Password: String, Join Date: Date)

- MID → FirstName, LastName, UserName, Password, JoinDate
- LHS contains a candidate key and all non-candidate keys on the RHS are fully functionally dependent on the candidate key; therefore, the table is in 1NF, 2NF, 3NF, and BCNF

MerchantProducts(MID: String, PID: String)

- MID and PID are the candidate keys that have no composite key; therefore, the process is done.

Orders(OID: String, CID: String, Order Date: Date)

- OID → CID, OrderDate
- OID is a super key for the RHS functional dependencies, so the table is in all forms of normalization.

MerchantOrders(MID: String, OID: String)

- MID and OID are the candidate keys that have no composite key; therefore, the process is done.

Products(PID: String, Product Name: String, Price: Integer, Description: String, Availability: Boolean, Type: String)

- PID → ProductName, Price, Description, Availability, Type
- RHS is fully functionally dependent (2NF) on the super key (1NF) on the LHS (BCNF) and they are not transitive dependencies (3NF)

Reviews(RID: String, PID: String, Author: String, Product Name: String, Description: String, Stars: Integer, Likes: Integer, Date: Date)

- RID → PID, Author, ProductName, Description, Stars, Likes, Date
- RID is a super key (BCNF) that uniquely defines (1NF) each row in the table. The RHS is fully functionally dependent on the LHS (2NF) and the RHS are not transitively dependent on each other

ReviewsTextualContext(RID: String, Type: String)

- RID and Type are the candidate keys that have no composite key; therefore, the process is done.

Cart(CID: String, UID: String, Quantity: Integer, Total Price: Integer)

- CID → UID, Quantity, TotalPrice
- LHS is a super key that each attribute has a single, atomic value, and non-transitively dependent RHS is fully functionally dependent on each other.

CartProducts(CID: String, PID: String, Product Name: String)

- CID, PID → ProductName
- LHS are candidate keys and RHS is an attribute that is fully dependent on the LHS, so it is in all of the normalization forms

Final Schema Design

Admin(AID: String, User Name: String, User Password: String)

MemberUsers(UID: String, First Name: String, Last Name: String, Shipping Address: String, User Name: String, Password: String, Join Date: Date)

MemberUsersFriends(UID: String, FUID: String)
UID is Foreign Key that points to MemberUsers or GuestUsers UID.

Merchants(MID: String, First Name: String, Last Name: String, User Name: String, Password: String, Join Date: Date)

MerchantProducts(MID: String, PID: String)
MID is Foreign Key that points to Merchants MID.
PID is Foreign Key that points to Products PID.

Orders(OID: String, UID: String, Order Date: Date)

MerchantOrders(MID: String, OID: String)
MID is Foreign Key that points to Merchants MID. OID is Foreign Key that points to Orders OID.

Products(PID: String, Product Name: String, Price: Integer, Description: String, Availability: Boolean, Type: String)

Reviews(RID: String, PID: String, Author: String, Product Name: String, Description: String, Stars: Integer, Likes: Integer, Date: Date)

ReviewsTextualContext(RID: String, Type: String)
RID is Foreign Key that points to Reviews RID.

Cart(CID: String, UID: String, Quantity: Integer, Total Price: Integer)

CartProducts(CID: String, PID: String, Product Name: String)
CID is Foreign Key that points to Cart CID.
PID is Foreign Key that points to Products PID.

5. Major Design Decisions

When determining what type of software technologies to use we initially decided to use Javascript and VScode to create the web application. We started with Nodejs and used libraries such as Express, Sequelize, and Postgres. However, this turned out to be a big challenge because it was not compatible with each other's systems. Additionally, not every team member was comfortable with writing Javascript. Since we could not figure out how to get the web application running we decided to use Java, JDBC, Tomcat server, and MySQL. This turned out to be the right choice as it would lead to the product we have now.

Secondly, when designing the user interface we planned to create a homepage where products would be displayed and where users can log in or sign up from. However, due to complexity issues, we decided to go in a different direction. In the end, we decided to have a login starting page where the user can log in or signup form. Upon successful login/signup, it will redirect to their corresponding pages where users can perform their tasks. Once we figured out what happens at the start we could not continue implementing the features to our web application.

6. Implementation Details

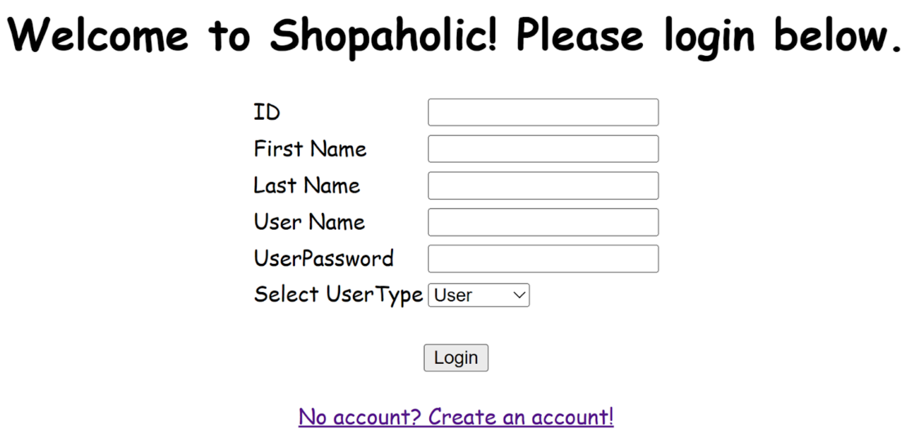
As mentioned above in Application/Functionality Requirements and Architecture we used Java, JDBC, JSP, Tomcat server, and MySQL. First, we started with Nodejs on VScode however that did not work and we moved to Java on Eclipse. We then followed a tutorial on how to create a web application using Servlet and Tomcat. So then we started by creating a simple welcome page using Servlet and Tomcat to display a welcomePage.jsp file.

However, it took a while for all of us to get on the same page and get the server running on our systems. Once we had it on our systems running we then continued creating our application. Then we worked on creating the tables and creating the backend and front end for the application. We made sure that we could edit, create, and delete from the database using the front end.

Once our team was able to connect with the database, we continued to work on implementing the relationships between the tables. We had already created the schema and the table however we had not yet implemented the front end of the application for displaying a user's orders or a merchant's products. When we finished, we then cleaned up the project structure and created a class that can be called whenever we need to connect to the database. This made it so that we don't always have to write the connection statement.

7. Demonstration of System Run (with screenshots)

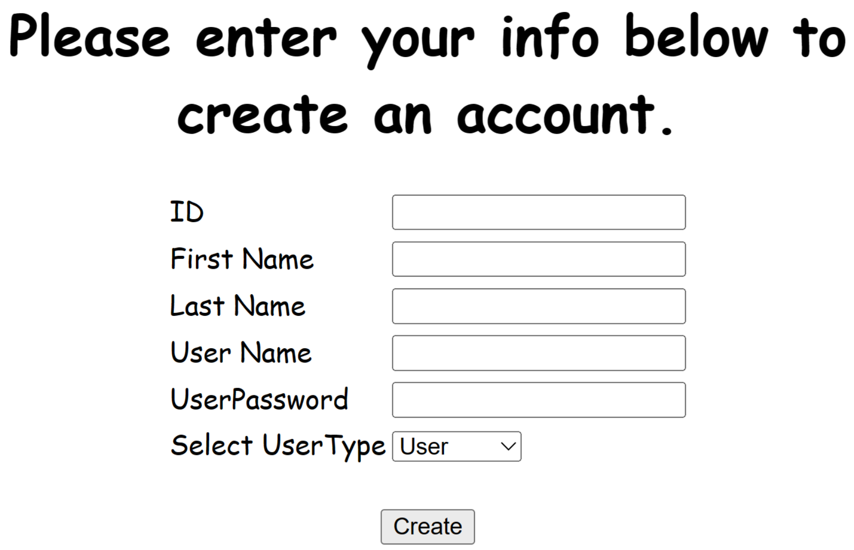
Login Page:



Welcome to Shopaholic! Please login below.

ID	
First Name	
Last Name	
User Name	
UserPassword	
Select UserType	User

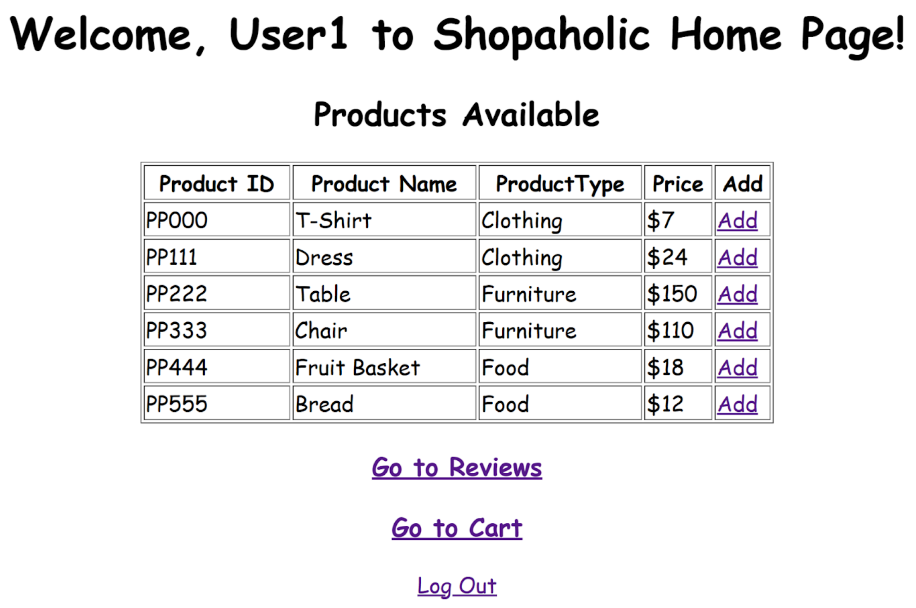
Sign Up Page:



Please enter your info below to create an account.

ID	
First Name	
Last Name	
User Name	
UserPassword	
Select UserType	User

User View:



Welcome, User1 to Shopaholic Home Page!

Products Available

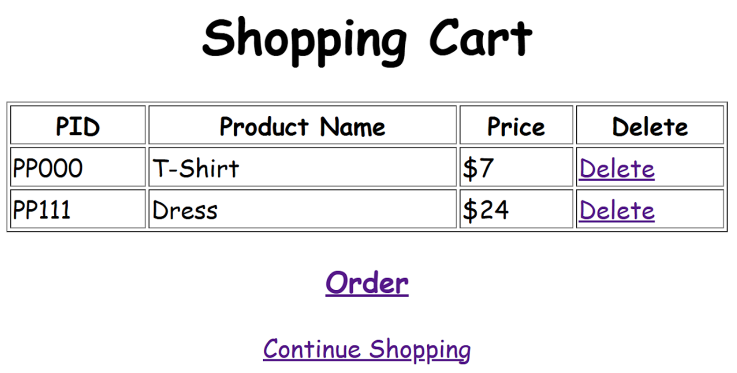
Product ID	Product Name	ProductType	Price	Add
PP000	T-Shirt	Clothing	$7	Add
PP111	Dress	Clothing	$24	Add
PP222	Table	Furniture	$150	Add
PP333	Chair	Furniture	$110	Add
PP444	Fruit Basket	Food	$18	Add
PP555	Bread	Food	$12	Add

Go to Reviews

Go to Cart

Log Out

User Shopping Cart:



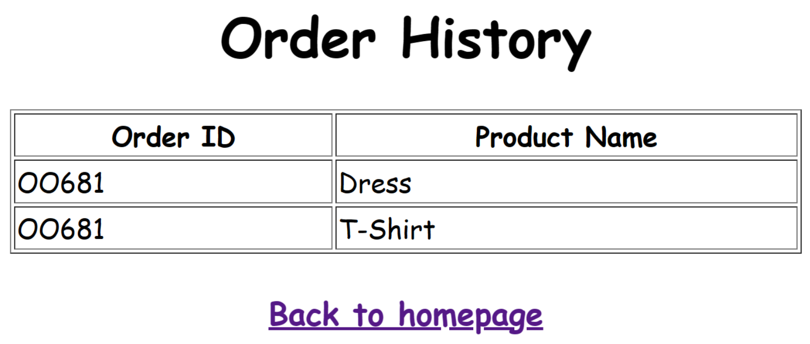
Shopping Cart

PID	Product Name	Price	Delete
PP000	T-Shirt	$7	Delete
PP111	Dress	$24	Delete

Order

Continue Shopping

User Order History:

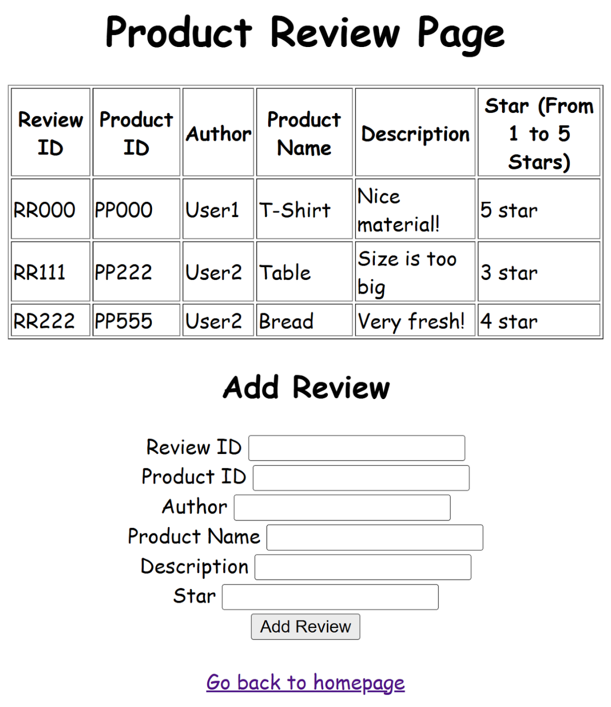


Order History

Order ID	Product Name
OO681	Dress
OO681	T-Shirt

Back to homepage

Product Reviews:



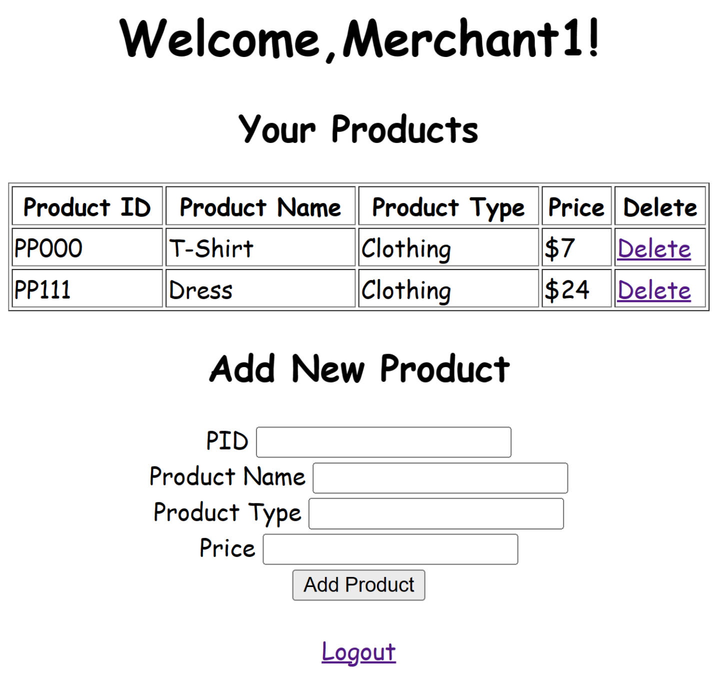
Product Review Page

Review ID	Product ID	Author	Product Name	Description	Star (From 1 to 5 Stars)
RR000	PP000	User1	T-Shirt	Nice material!	5 star
RR111	PP222	User2	Table	Size is too big	3 star
RR222	PP555	User2	Bread	Very fresh!	4 star

Add Review

Review ID
Product ID
Author
Product Name
Description
Star

Merchant View:



Welcome,Merchant1!

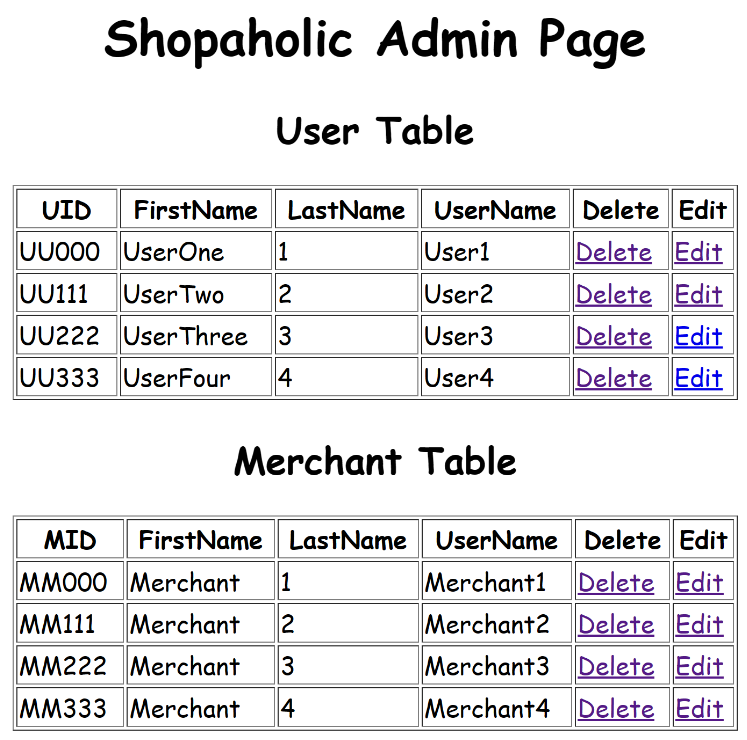
Your Products

Product ID	Product Name	Product Type	Price	Delete
PP000	T-Shirt	Clothing	$7	Delete
PP111	Dress	Clothing	$24	Delete

Add New Product

PID
Product Name
Product Type
Price

Admin View (users, merchants):



Shopaholic Admin Page

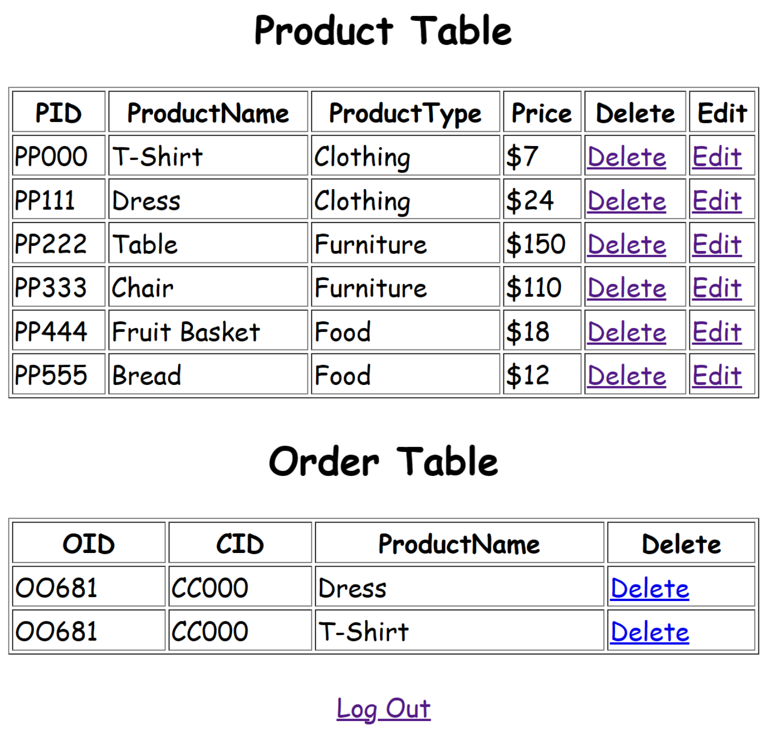
User Table

UID	FirstName	LastName	UserName	Delete	Edit
UU000	UserOne	1	User1	Delete	Edit
UU111	UserTwo	2	User2	Delete	Edit
UU222	UserThree	3	User3	Delete	Edit
UU333	UserFour	4	User4	Delete	Edit

Merchant Table

MID	FirstName	LastName	UserName	Delete	Edit
MM000	Merchant	1	Merchant1	Delete	Edit
MM111	Merchant	2	Merchant2	Delete	Edit
MM222	Merchant	3	Merchant3	Delete	Edit
MM333	Merchant	4	Merchant4	Delete	Edit

Admin View continued (products, orders):



Product Table

PID	ProductName	ProductType	Price	Delete	Edit
PP000	T-Shirt	Clothing	$7	Delete	Edit
PP111	Dress	Clothing	$24	Delete	Edit
PP222	Table	Furniture	$150	Delete	Edit
PP333	Chair	Furniture	$110	Delete	Edit
PP444	Fruit Basket	Food	$18	Delete	Edit
PP555	Bread	Food	$12	Delete	Edit

Order Table

OID	CID	ProductName	Delete
OO681	CC000	Dress	Delete
OO681	CC000	T-Shirt	Delete

Log Out

8. Conclusions

In the time working on this project we faced many challenges and learned a lot of lessons. To start with we struggled with cloning a git repo locally. Even though we had read and write permissions, sometimes it would not allow us to push up.

Another challenge was learning Servlets, Tomcat server, and JSP files. This was our group's first time working with Java to create a web application so we all had to learn from the beginning on how to use these tools.

Lastly, learning to use Eclipse was fairly difficult. Some of the team members had previously not worked with Eclipse so it was completely new. This was honestly the biggest challenge I (Gael) personally dealt with.

In the end, though, we learned a lot through these challenges. We were able to learn how to set up a server using a Tomcat server and load our website onto it. We also learned to use Eclipse comfortably by the end of the project.

Overall, we learned how to create a full-stack web application to create, update, and delete records from a database. Some things we would do in a future version of our Shopaholic project would be to implement a better UI. Since our team spent most of the time working on the backend of the project, we were unable to spend time designing the look and feel of the user interface.

Another improvement could be minor features such as users such as being able to upload profile pictures and products being shown as images instead of text. If we had another opportunity to create another web application, our team would try to use Nodejs to create the project. Luckily, our web application turned out successful and our team was able to gain new skills during the creation of Shopaholic.