



Beverage Ordering App Project

<http://18.191.207.218/CSC4610/webpage/login.html>

Created By: Wen Zhe Chuah

Date: December 3rd, 2020

Tables of Contents

Introduction.....	3
System Architectural Design.....	3
ER Diagram.....	4
Relational Table.....	5
Detailed Description of each component.....	8
Account Registration.....	9
Login System.....	10
Admin's View.....	10
Menu View.....	11
Cart View.....	12
Review View.....	13
Delivery View.....	14
Conclusion.....	15

Introduction

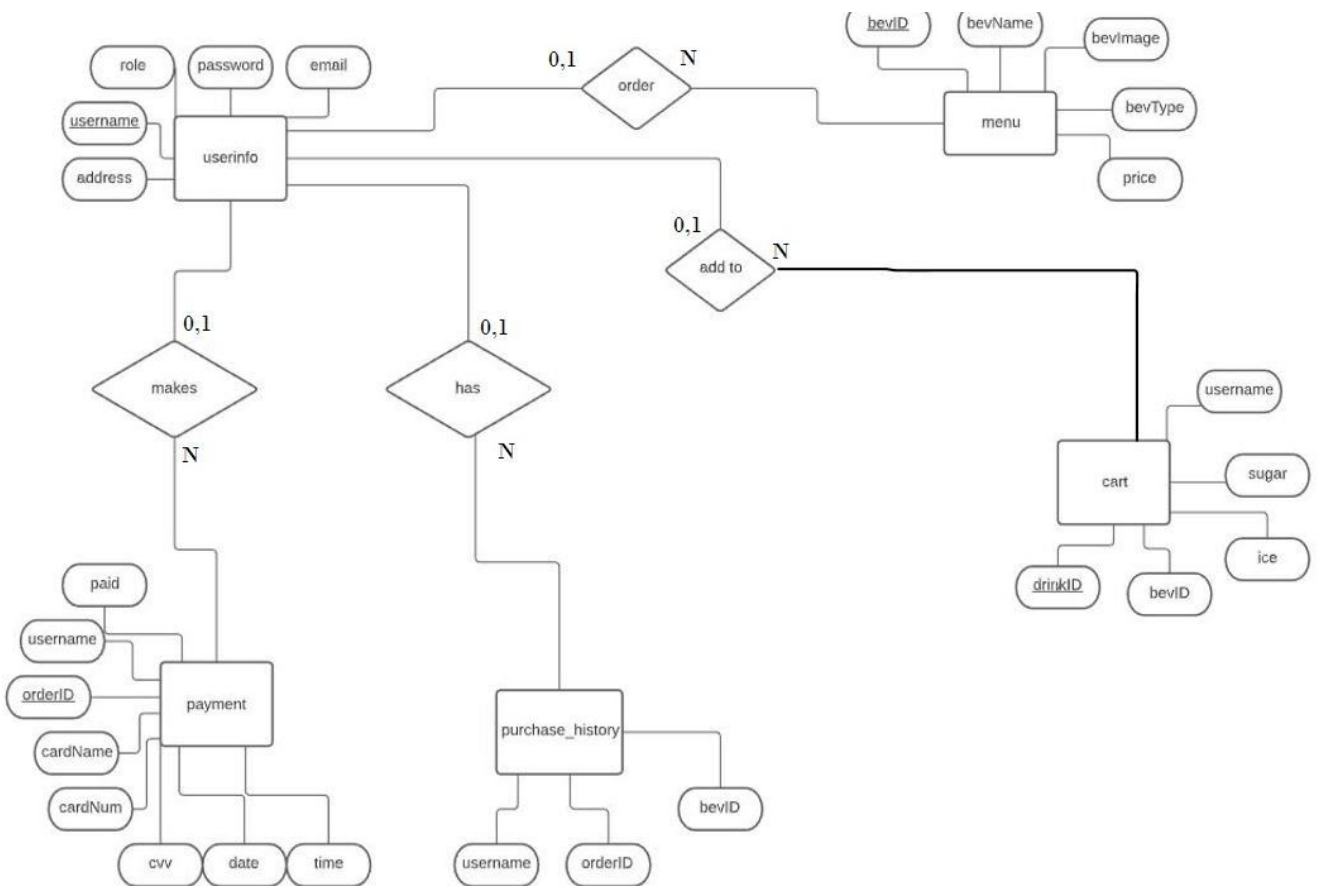
Bubble Tea Shoppe is an online web beverage ordering app that allows users to order their favorite drinks online and get delivered to them. This project aims to create a user-friendly and easy-to-use user interface for the user. The reason I decide to work on this application is that I enjoy drinking all types of beverages especially bubble tea, and also there isn't any application that only focuses on beverage orders.

System Architectural Design

This web application is deployed on an Amazon EC2 instance with a xampp server. The reason why I decided to utilize a xampp server is that it is easier for me to use PHP on my web project. The direct link to the website is <http://18.191.207.218/CSC4610/webpage/login.html>. It is hosted with Apache Web Server that is included in the xampp server. For the database, I am using the PHPMyAdmin that are also included in the xampp server.

In this web application, I have used technologies like the basic HTML, CSS, JQuery, Bootstrap 4, PHP, and MySQL. HTML and CSS are used for the structure and presentation of the webpage. JQuery is used to make animation and change the value of an attribute of the static structure. Bootstrap 4 is used mainly for a pop up modal when I click on a button. PHP is utilized for interacting with the database and creating a session, sending and receiving session variables, redirecting to another webpage, handling validation, and backend logic. Lastly, I also use PHPMyAdmin which acts as a communicator to handle the administration of my database in MySQL.

In my database, there are five tables which are, userinfo, cart, menu, payment, and purchase_history tables. The image below is my ER Diagram for the web application. Userinfo table will have zero or one to many relationships to all tables including menu, payment, purchase_history, and cart. Zero or one user will order multiple drinks from the menu, make multiple payments, add multiple drinks to the cart, and have multiple purchase histories.

ER Diagram

From the ER diagram, I have derived some relational tables that allow the web server to store and update data to and delete data from the MySQL database. The headers of the relational table will be shown below with details and descriptions on how they are used in the web application.

Relational table: userinfo

The userinfo table will hold 5 columns of data which are username, password, email, address, and role respectively. Username will be the primary key of the table because username should be unique for all registered users. Password, email, and address columns are all self-explanatory. However, the role will identify if an account is for an admin or a regular user. When a user is login or sign up to the website, they will interact with this table to store the data or to validate if the username exists when signing in.

<u>Username</u>	Password	Email	Address	Role
varchar(255)	varchar(255)	varchar(255)	varchar(255)	varchar(255)

Relational table: menu

The menu table holds 5 columns of data which are bevID, bevName, bevImage, bevType, and the price. bevID is the primary key of this table so that each row of data will have unique data to interact with. bevName is the name of a beverage. bevImage is the id of an image such as "Chocolate-Sea-Salt-Crema". bevImage is the type of the beverage and also the subfolder name of the image folder. With the combination of

bevImage and bevType, I can simply access the image folder with bevType/bevImage + .png. This table is mainly used to print the beverage menu of the shop.

<u>bevID</u>	bevName	bevImage	bevType	Price
varchar(10)	varchar(255)	varchar(255)	varchar(20)	decimal(3,2)

Relational table: cart

The cart table holds 5 data columns which include drinkID, username, bevID, ice, and sugar. This table will store what the user orders. drinkID is the primary key of this table. The only usage for the drinkID is to let the user delete what they added to the cart. When they remove the item in the cart, the backend code will track the drinkID and remove the data row in the table. I also added a username and bevID because it gives me a better opportunity to get different types of SQL queries in the future. Ice and sugar will store the user preference of their sweetness and ice level.

<u>drinkID</u>	Username	bevID	Ice	Sugar
varchar(8)	varchar(255)	varchar(6)	varchar(4)	varchar(4)

Relational table: payment

The payment table will keep track of all the payment transactions a user did. This table holds orderID, username, cardName, cardNum, cvv, time, date and paid. orderID is the primary key of this table. Username, card name, card number, cvv, time, and date is also very self-explanatory. Paid column will be the data that will store the total price of the transaction. This table will interacted when a user makes a payment of their order.

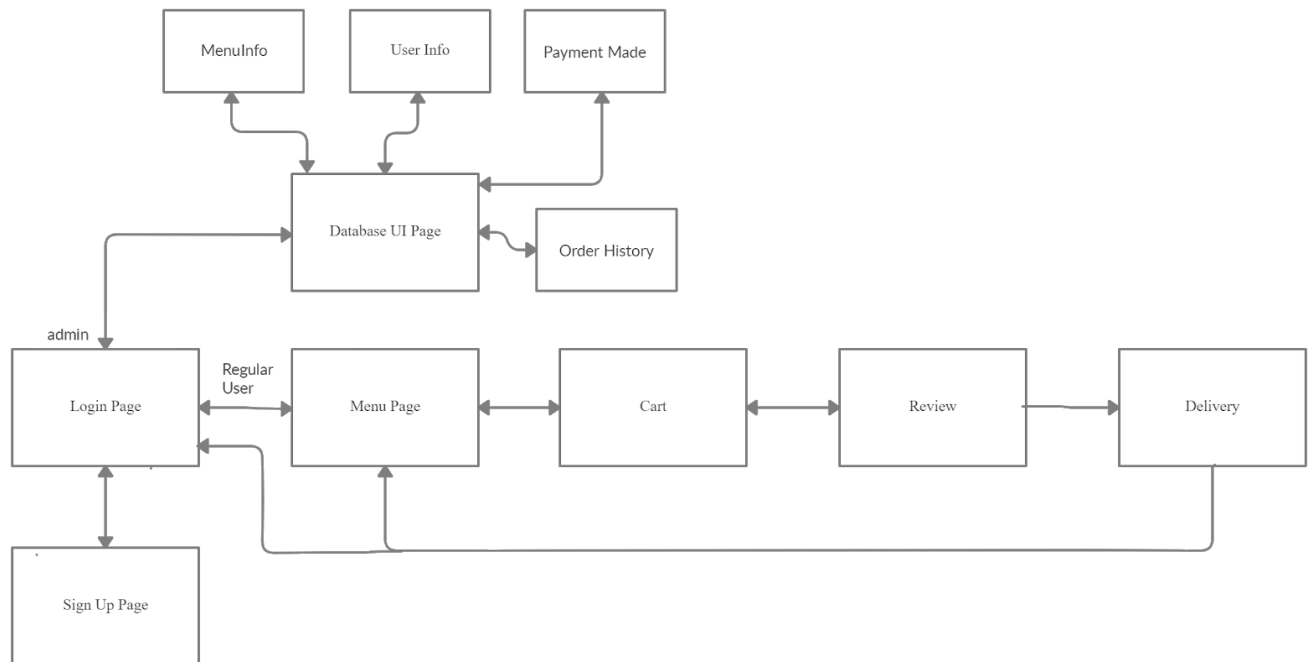
orderID	Username	cardName	cardNum	Cvv	Time	Date	Paid
Varchar	varchar	varchar	bigint	Smallint	Varchar	Varchar	Decimal
(7)	(255)	(255)	(12)	(3)	(10)	(10)	(6,2)

Relational table: purchase_history

The purchase_history table will keep track of every individual order of a transaction that includes the orderID, bevID, and username. This combination of data columns will allow the admin to figure out in which date and time, which customer has ordered what drinks. This table will interact only when the admin looks into the database webpage.

orderID	bevID	Username
varchar(7)	varchar(6)	varchar(255)

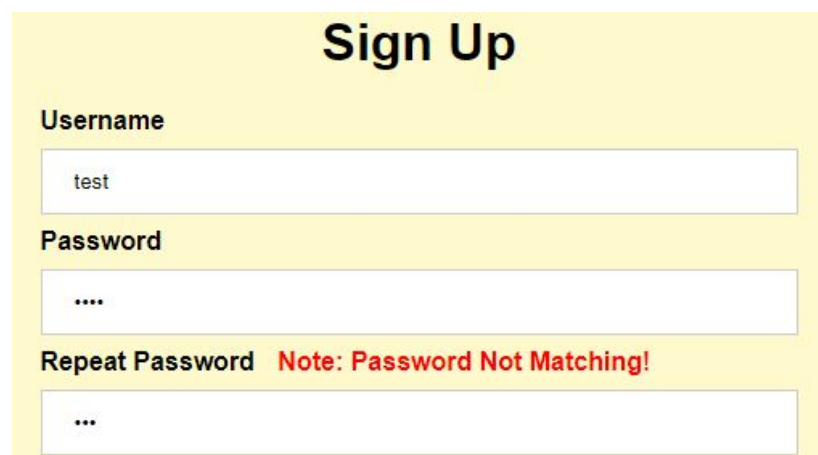
Detailed Description of Each Component



The image above is the website flow of the web application. Each square represents a file that contains HTML and CSS, which means it doesn't include the back end processes file. All the files are in .php format but the login and signup page are in .html. There are five more php files in the webpage folder that handles the process of redirection of pages, session control, and action of MySQL. There is also one CSS folder that has an additional 7 .css file, and an image folder that has all the images on the website.

Account Registration

The account registration function happens in the signup.html. Users will have to register an account with a unique username, password, repeated password, email address, and delivery address. There are 2 types of validation for the registration process. First, if the webpage will listen to events to see if the password and the repeated password are the same. If they are different, the sign up button will be disabled.



The image shows a 'Sign Up' form with a yellow background. It contains three input fields: 'Username' with the value 'test', 'Password' with masked characters '....', and 'Repeat Password' with masked characters '...'. A red error message 'Note: Password Not Matching!' is displayed next to the 'Repeat Password' label.

Sign Up

Username

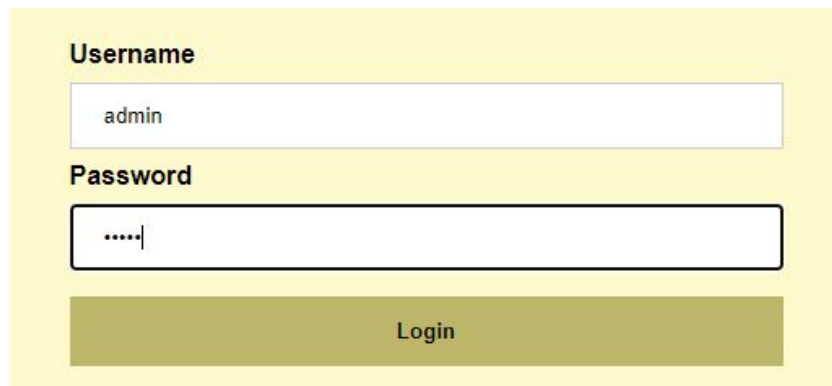
Password

Repeat Password **Note: Password Not Matching!**

The second validation will happen in registration.php. Once the user clicks on the signup button, name, password, email, and address will be retrieved using the post method. In registration.php, it will check if the username exists in the database. If it exists, the user will be returned to the signup page and alert the user to choose another username because it has to be unique. On the other hand, the user's information will be inserted into the userinfo table and will be redirected to the login page.

Login System

The login page will let the user input a username and password. Once the user enters their information, the validation will process in `validate.php`.



The image shows a login form with a yellow background. It contains two input fields: 'Username' with the text 'admin' and 'Password' with masked characters '.....'. Below the fields is a green 'Login' button.

In `validate.php`, the name and password will be retrieved from the post method. Then, it will check if the name and password exist in the database. If it doesn't exist, the user will be redirected to the login page with an invalid alert. If it exists, the role of the account user will be checked. If the role is an admin, the user will be redirected to `databaseUI.php`. If the user is a regular customer, they will be redirected to `menu.php`.

Admin's View

There is only one admin account right now, which username and password are both 'admin'. In `databaseUI`, PHP, there are 5 buttons including sign out. The user info button will redirect the admin to `userDB.php` to view the list of the user's record. The menu info button will redirect the admin to `menuDB.php` to view all the beverages in the menu. The payment made button will redirect the admin to `paymentDB.php` and show all the transactions. Lastly, the order history button will show every single unique drink purchase.

Menu view

In menu.php, there is a list of buttons with the beverage image and name on it. Each button clicked will pop out a modal that shows what sugar and ice level the user prefers for their drinks. After that, the user will have to click on the add to cart button to add their order to the cart.



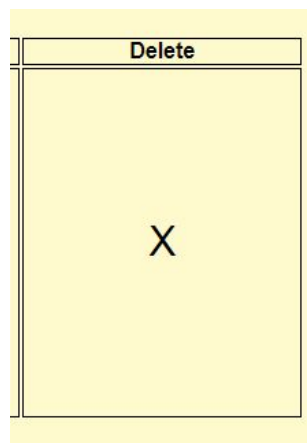
When the add to cart button is clicked, the orderProcess.php will receive bevId, sugar, and name from the post form and username from the session. Next, it will randomly generate a drinkID for the order and then check if the drinkID exists in the database. If it exists, the PHP will regenerate a drinkID. After that, drinkID, username, bevId, sugar, and ice level will be added to the cart database and redirect users to the menu page.

Cart View

Once the user decides to move on with their order, they will have to click on the cart image button on the top right corner of the page, right below the sign up button.



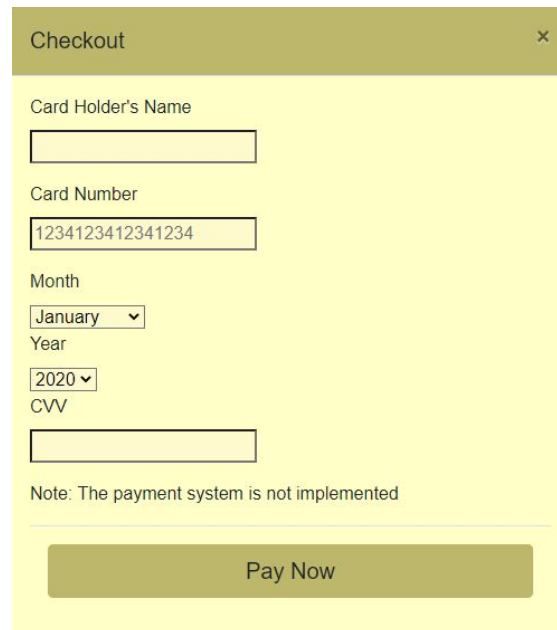
This button will redirect users to `cart.php`, where all the beverages the user added to cart is listed. In this page, it will read the cart database and portray the beverage image, name, sugar level, ice level, and price. The price is then repeatedly added and forms a total price. Another function in this page is the delete button.



This button is to allow the user to remove their cart item from the cart. When this button is clicked, it will send a request to `otherProcess.php`, which is a php page which does loose processes. At this point, `orderProcess.php` will take in the deleted `drinkID` and delete the row of data from the cart table. Then it will redirect the user back to the cart page and the deleted row will not be there anymore.

Review View

When the user is ready to order their drinks, they will click on the review button and will be redirected to checkout.php. In this php file, the user will review again what they order, subtotal, 9.95% tax, and the total that they have to pay. After that, the user can click on the checkout button and a checkout modal will pop out.

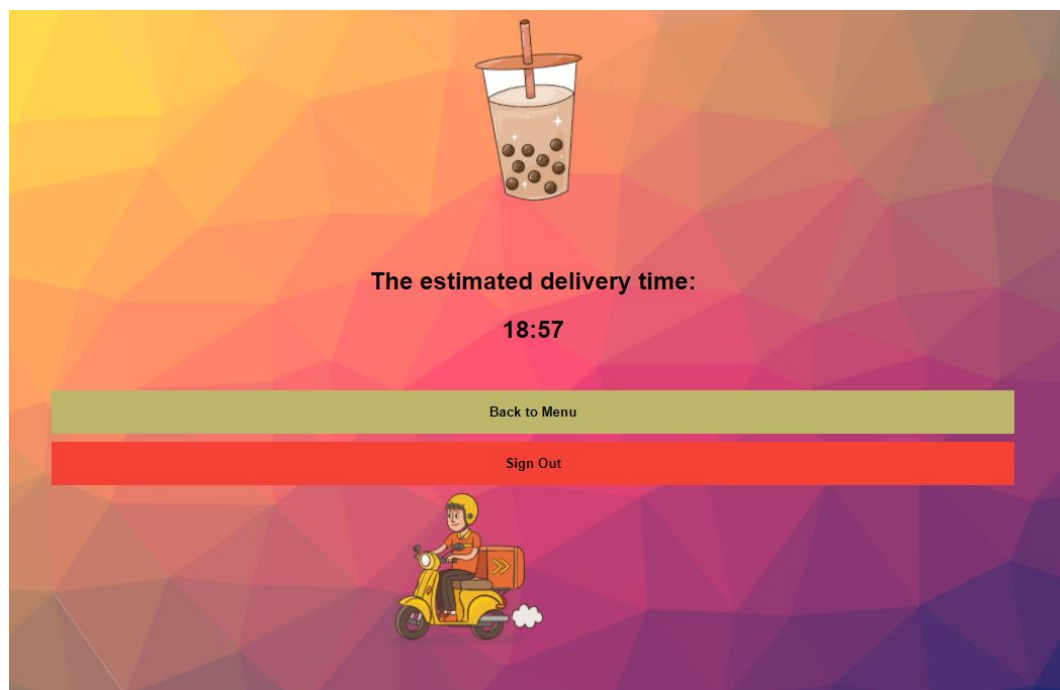
A screenshot of a checkout modal window. The modal has a title bar with the text 'Checkout' and a close button (X). The form is light yellow and contains the following fields: 'Card Holder's Name' with a text input; 'Card Number' with a text input containing '1234123412341234'; 'Month' with a dropdown menu showing 'January'; 'Year' with a dropdown menu showing '2020'; and 'CVV' with a text input. Below these fields is a note: 'Note: The payment system is not implemented'. At the bottom of the modal is a large green button labeled 'Pay Now'.

Users will have to enter the card holder's name, 12-digit card number, expired month and year, and 3-digit cvv. Since this is just a prototype project, therefore the payment system is not implemented. Once the user is ready to pay, they will click on the pay now button and the validation process will be done in payProcess.php. payProcess page will get card name, number, and cvv from post method and will check if the user entered the correct 12-digit combination for the card number and 3-digit combination for cvv. If either of them is incorrect, the user will be redirected to the checkout page. If no errors happened, orderID will be generated and will be inserted to the payment database together with username, card name, card number, cvv, time, date, and price paid. Also,

orderId, bevID, and username will be inserted to purchase_history table for additional records. Lastly, the cart table in the database will be removed for the user to prevent duplication.

Delivery View

After the process, the user will be redirected to delivery.php. In this page, the user will see a countdown timer for the delivery and the animated motorcyclist moving from right to left.



The countdown timer is functioned by javascript. The timer is randomly picked from a timeslot array which contains 5 - 30 minutes. When the countdown is done, the timer will be replaced by "Delivered or Reaching Soon =)". The animation is also done by javascript like how we did in CSC2610. After that, the user can click back to the menu or sign out.

Conclusion

In conclusion, the web application project is completed with no visible bugs under normal usage and runs responsively and efficiently with a standard Internet speed. The only flaw for this website is that it will not fit all monitors and mobile web screens because of my inexperience in web design. There are more functions and improvements that this website can be. For instance, admin users can have more power alternating the data in the database, a working payment system, adjusting the website to fit all types of resolution, and more.