**Final Report**

**for**

**Nom Nom Express**

**By**

**Angela Ziesel,**

**Kamila Diyanova,**

**Kento Hyono,**

**Manny Sandhu**

Contents

[1. Project Overview 3](#_Toc149131042)

[2. Project Specifications 3](#_Toc149131043)

[2.1 Function Requirements 3](#_Toc149131044)

[2.2 Non-Function Requirements 3](#_Toc149131045)

[2.3 Group Roles 4](#_Toc149131046)

[3. Usability Guide 4](#_Toc149131047)

[4. Problems and Challenges 9](#_Toc149131048)

[5. Skills Learned 10](#_Toc149131049)

[6. Proposed Changes 11](#_Toc149131050)

[7. Code Snippets 12](#_Toc149131051)

[7.1 Back-end / middleware 12](#_Toc149131052)

[7.2 Middle-ware 13](#_Toc149131053)

[7.3 Front-end 14](#_Toc149131054)

**Table of Figures**

[Figure 1: Login Window 5](#_Toc149131056)

[Figure 2: Registration Page 5](#_Toc149131057)

[Figure 3: Main Page - logged in 6](#_Toc149131058)

[Figure 4: Account Info Page 7](#_Toc149131059)

[Figure 5: Order Cart Page 8](#_Toc149131060)

[Figure 6: Admin Page 9](#_Toc149131061)

[Figure 7: Back-end / middleware Code Snippet 12](#_Toc149131062)

[Figure 8: Middle-ware Code Snippet 13](#_Toc149131063)

[Figure 9: Front-end Code Snippet 14](#_Toc149131064)

## Project Overview

Our client, Nom Nom Express found itself in a situation where it lacked any discernible online presence. Therefore, they have expressed the imperative need for the development of a sophisticated website that will serve as a platform for showcasing their diverse menu offerings and facilitating seamless online orders for delivery, thereby enhancing the overall experience for their clients.

Our project directive therefore, is to produce a functional mobile responsive web storefront for Nom Nom Express to promote and sell their food menu items for delivery, with all in-scope requirements met using free open-source software, tested and deployed in 8 weeks.

## Project Specifications

### 2.1 Function Requirements

* R01 – The client must be able to log into an admin account
* R02 – The admin account must be able to add products to the web site
* R03 - Must be able to display products by categories
* R04 - Customers must register to place orders
* Must provide a fully functioning shopping cart utility where a customer can:
  + R05 - display the selected items currently in the cart
  + R06 - add selected products to the cart
  + R07 - delete products (individually or all) from the cart
  + R08 - adjust the quantity of a selected product currently in the cart
  + R11 (CR01) - Create a unique file for each order that contains the order confirmation information so that at a future date, the information in the file can be sent to the client as an email message
  + R12 - include a secure, online payment process
* R13 - Allow customers to view their order history
  + R50 (CR02)
    - A. All users must be shown a new privacy page informing them of this new law, what personal information is being collected on them and how that information will be used.
    - B. Users must either accept the terms or decline them.
    - C. Users must be able to change their mind on accepting or declining the terms later on.
    - D. Users that decline the terms will not be allowed to login to the main site.

### 2.2 Non-Function Requirements

* R09 - preserve the contents of the cart if the user’s session is disconnected accidentally
* R10 - have the cart accessible regardless from any machine the customer is logging in
* R14 - Provide an intuitive, consistent look and feel to the user interface
* R15 - Use free open-source software
* R16 - Have the system fully tested and up and running in 8 weeks (this is a firm deadline)

### 2.3 Group Roles

* Team Lead –This role has the final say when a decision “stalemate” situation arises. Also, the team member assigned to this role has to check every member’s progress and confirm that project is going smoothly by organizing the meeting and tasks assigned to each member.

The role is also responsive for following tasks:

* Producing Weekly Status Report
* Compiling and Uploading Deliverables to D2L
* Ensuring demo is ready and presenting it
* Project coordination (setting / calling meetings)
* Update documentation each week
* Update Gantt Chart each week
* Database/Back-end Developer – This role is fully responsible for the MySQL database. It includes database design, setting up relationships among entities, normalization, and inserting and uploading data into them. In addition to it, this role needs to structure the code to interface to it.
* Middle-ware Developer – Middle-ware developer is the key role to build PHP code to communicate among web browser (where clients see), web server, and database server. Since this project is monolithic, all of the development fields are tightly connected, resulting that this role is assigned to team members more often than other roles.
* Front-end Developer – This role focuses on user interface coding, which includes HTML, CSS, JavaScript to build website that fits to our objective, food delivery, by choosing appropriate theme and design decisions and implementing them.
* Technical Writer – As a technical writer, they must produce the technical documentation required each week as a deliverable, such as meeting minutes, weekly status report. Since every member needs at least one documentation submitted to the lead every week, virtually all members are assigned to this role.
* Testing – This role is responsible for testing variety of units, and whether if the website work as expected, meets requirements, or cause any bug and behave undesired behaviour. At the point when the whole project is done, team member who assigned to this role must ensure entire application is tested as a whole.

## Usability Guide

1. User accesses [Nom Nom Express website](https://deepblue.camosun.bc.ca/~C0397554/cart/index.php) in browser of their choice
   * + - 1. Site prompts user to login

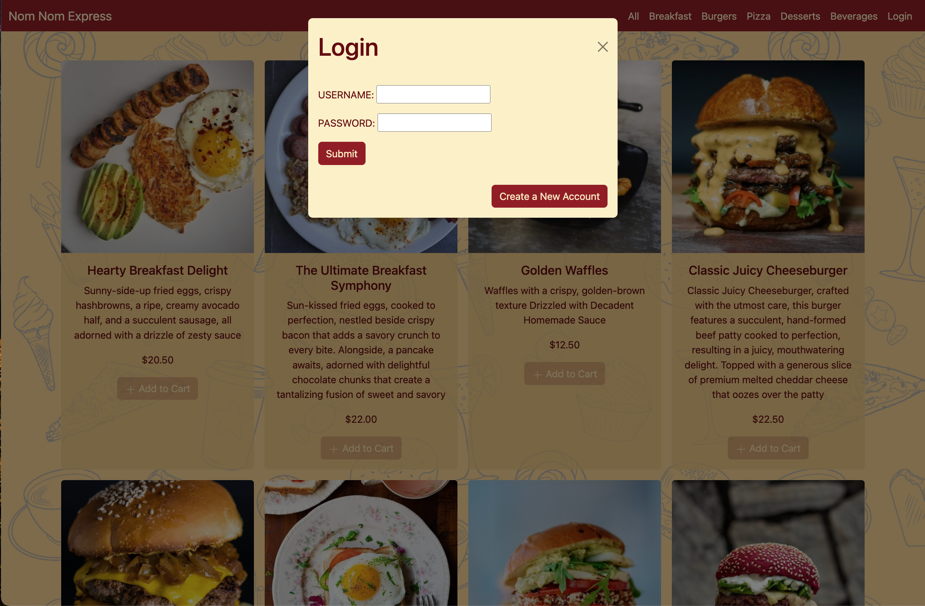
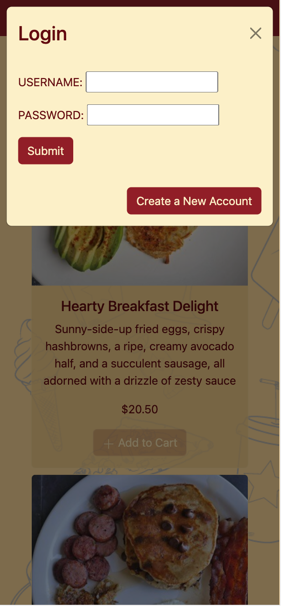
 

Figure 1: Login Window

(Desktop left, Mobile right)

* New User:

1. User selects  link
2. A General Data Protection Regulation window is displayed, User can accept or decline the privacy policy [CR02]
   * User clicks  button
     + 1. Window closes, redirected to main page, prompted to login
   * User clicks  button
     1. Window closes, redirects to registration page [R04]

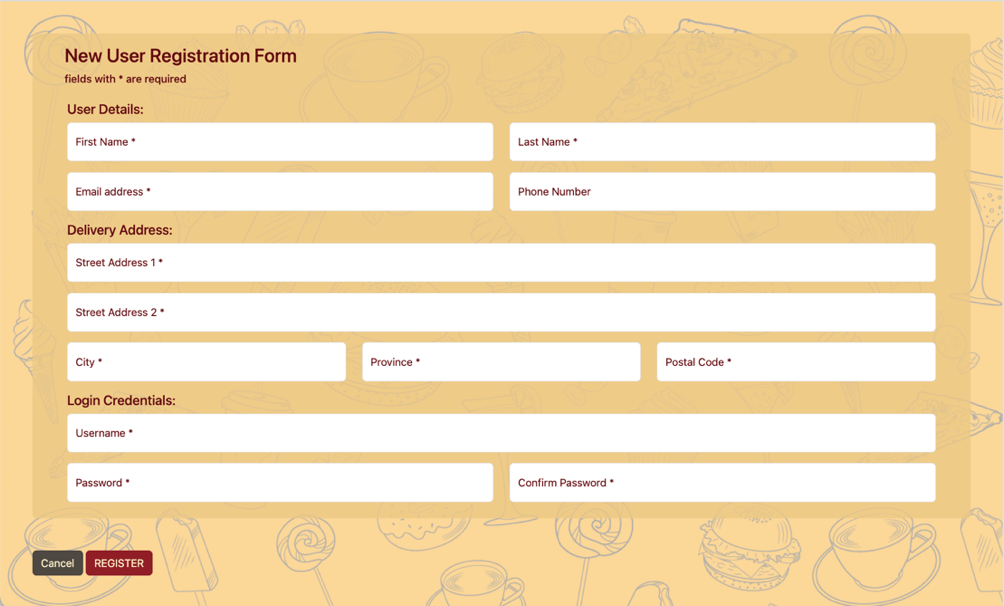
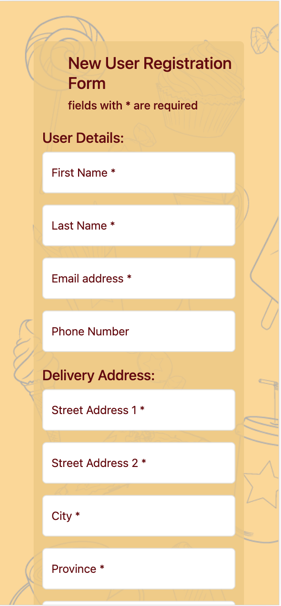
 

Figure 2: Registration Page

(Desktop left, Mobile right)

* + 1. Enter information into appropriate fields, click  button, correct any invalid entries or missing fields
    2. Site saves registration information in database, defaulted with customer role
  + Existing User:

1. User enters username and password, clicks  button, correct any invalid entry
2. User is logged in as either Customer or Admin, based on their account role [R01]

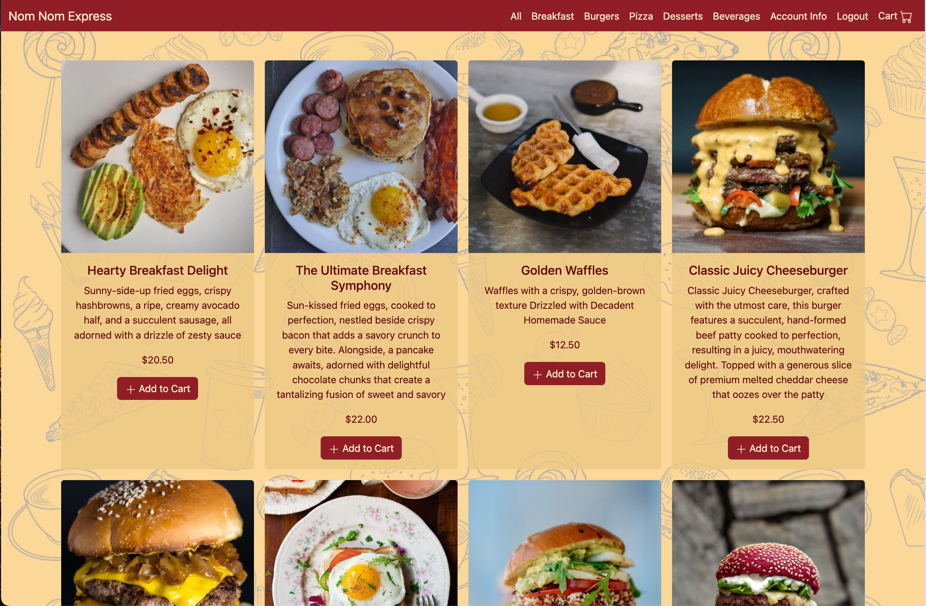
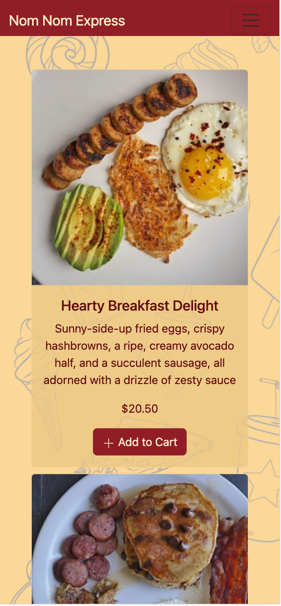
 

Figure 3: Main Page - logged in

(Desktop left, Mobile right)

* + - * Customer:

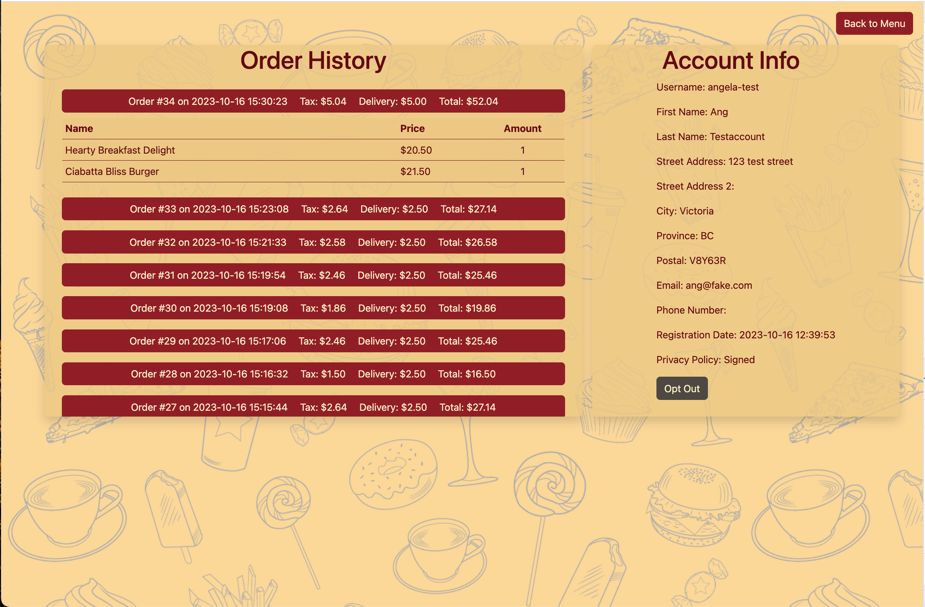
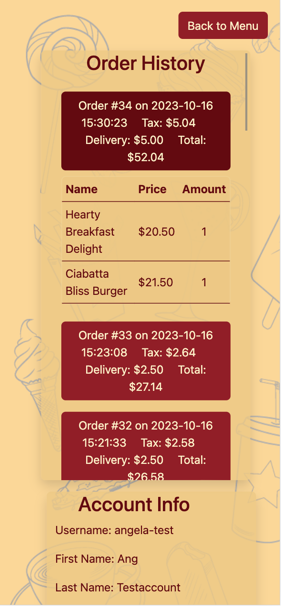
1. Site redirects to main page with updated navigation bar and activated add to cart buttons
2. Menu item display can be filtered by clicking on the category in the navigation bar [R03]
3. User can review account information and history by clicking  link in the navigation bar  

Figure 4: Account Info Page

(Desktop left, Mobile right)

* + 1. Order history is displayed, clicking on an order row will expand the details to display each item ordered [R13]
    2. Account Info is also displayed showing all the registration information
    3. Click on  button to decline privacy policy. User will no longer be able to add items to cart [CR02]

1. Add item to cart [R06]
   * 1. Menu item can be added to cart by clicking the  button. If clicked more than once, quantity of item in cart is increased
2. Display cart [R05]
   * 1. Cart can be displayed by clicking on  link in navigation bar
     2. Cart contents are retained regardless if User logs out, is disconnected, or logs in from another machine [R09, R10]

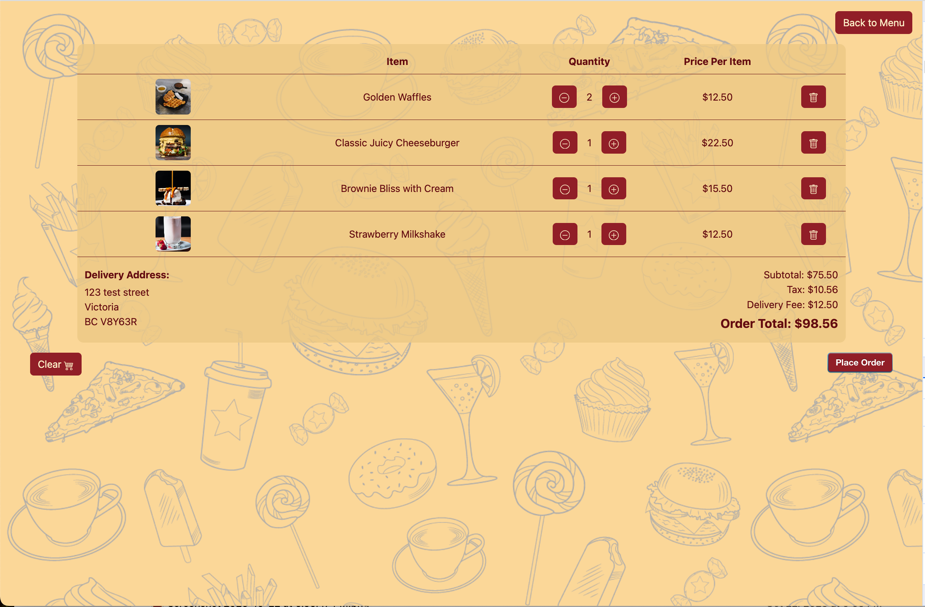
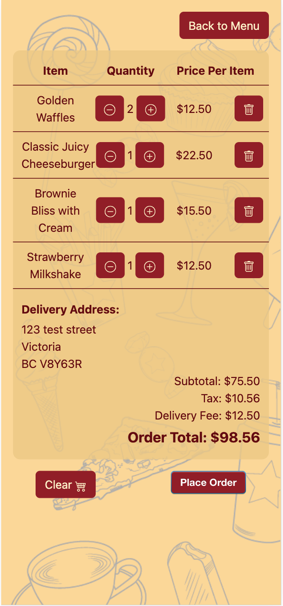
 

Figure 5: Order Cart Page

(Desktop left, Mobile right)

1. Update order cart
   * 1. Quantity of item can be adjusted by clicking the  or  buttons [R08]
     2. Item can be deleted by clicking on the  button [R07]
     3. Cart can be emptied by clicking on the  button, page will update to display message ‘No Items in the Cart’ [R07]
2. Order is placed by clicking on the  button [R12]
   * 1. Payment window opens, User enters credit card information and clicks Pay button
     2. Site redirects to Order Complete page, displaying order details and thank you message to User
     3. Site creates an Order Confirmation file in orders folder simulating an Order Confirmation email [R11/CR01]
3. User clicks  button to return to main page

* Admin:

1. Site redirects to Admin page, displaying Store Info, Add Menu Item form, and list of Existing Menu Items that can be edited or enabled/disabled

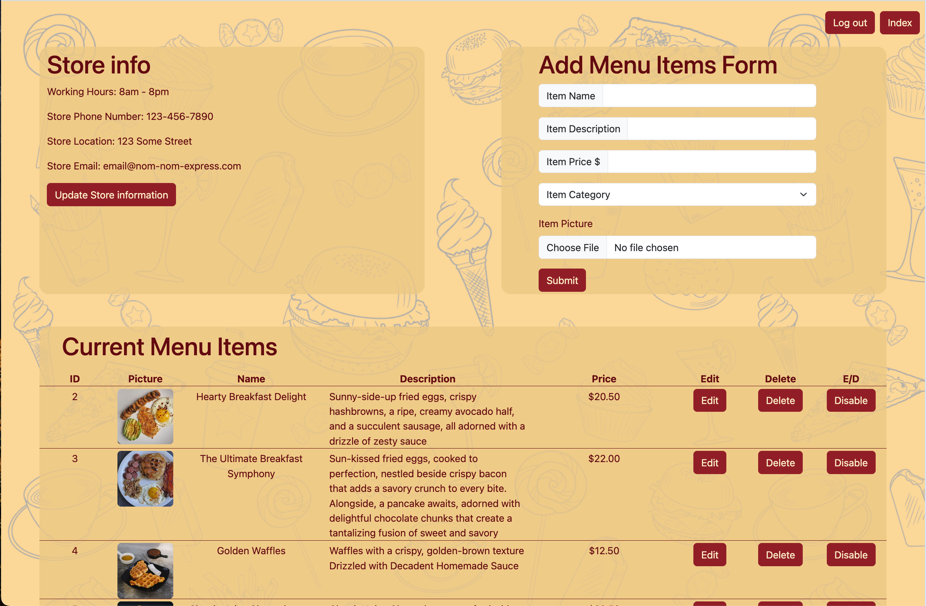
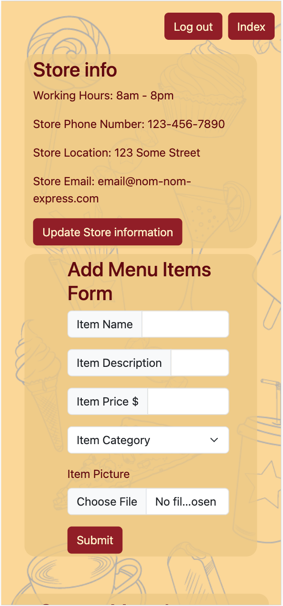
 

Figure 6: Admin Page

(Desktop left, Mobile right)

1. Update Store info
   * 1. Click  button
     2. Window opens with Update Form, User enters new information, clicks  button
     3. Window closes, new information displayed in Store info section
2. Add new Menu item [R02]
   * 1. Enter item information into Add Menu Items Form, select picture from file, click  button
     2. Site adds item to database and displays on main page
3. Update Current Menu Items
   * 1. Click  button for a window to modify the data fields for that menu item. Update appropriate field and click  button. Item is updated in database and changes reflect on main page
     2. Clicking  button asks for confirmation before the item is deleted. Click  button to confirm ( button to cancel). Item is deleted from database and removed from main page
     3. Clicking  button asks for confirmation before the item is disabled. Click  button to confirm ( button to cancel). Item remains in database but is not visible/available for order on main page
     4. Clicking  button (on item previously disabled) asks for confirmation before the item is enabled. Click  button to confirm ( button to cancel). Item is visible/available for order on main page.
4. User logs out
   * + - 1. Click  button from navigation bar
         2. Site redirects to main page and prompts for login

## Problems and Challenges

* We ran into a few technical issues that we didn’t have any control over with the database and GitLab. Our database was generating code that wasn’t usable, so we had to manually write the sql code to create our entire data model. With GitLab the college hadn't renewed the security certificate, making our college provided GitLab accounts unusable for a few days. We came together as a team to coordinate what to do if the issue was not fixed immediately; however, that issue was solved quickly did not impact our project significantly.
* Our team struggled with general consistency during the first few weeks of the project. We all came in with our own ideas and styles, that lead to a few conflicts that we had to work out. By setting aside time during a few meetings we were all able to come to an understanding what the website would look like and how to generally implement the code.
* We had a problem connecting to the Deepblue server because we didn’t have the correct permissions to upload files. For this issue we had to work with Shohreh who was communicating with the college network admins on our behalf. As soon as she had an answer, she walked us through on how to update the production site.
* We had quite a few general programing problems come up. With PHP being new to all of us things like syntax issues would pop up here and there but with files being large they were hard to spot sometimes. We overcame this problem by getting help from each other when needed to find those small issues. It also made us rethink how we needed to implement certain features and if needed to add or remove additional features. This issue was solved by updating our BRD document anytime we implemented a feature in a more efficient way then the use case asked for.
* Our team had a few issues when it came to MS Project. It was a program we had almost no experience with, and it was inaccessible on our home computers. We were able to finally access the program by remoting into an available computer on the Interurban campus. W still struggle a bit with the program itself, but we have improved by using it for scheduling tasks every week.
* We had a few problems when trying to create the Use Case writeups such as complexity, consistency, and workload. All these issues were caused by a lack of understanding of what a proper use case should look like and how to represent it. We overcame this challenge by working as a group to talk through our ideas and consistently updating use cases as needed.

## Skills Learned

* Our team developed an understanding of PHP as a middleware tool that allows interact with our database. We used PHP to create dynamic and responsive website, ensuring a robust connection between our frontend and the backend database.
* We accomplished the seamless integration of Stripe into our project, providing a secure and user-friendly solution for online payment processing.
* We developed the skill of creating comprehensive use cases and other project-related documentation. Creating use cases helped us to clearly define and document the functional requirements of our project and ensure that all team members understand what the system should do, thus reducing misunderstandings and miscommunications.
* Our team became skilled at using GitLab for collaboration. We learned how to work together efficiently by creating, merging, and managing different versions of our code, making our teamwork smoother.
* We got better at understanding and working with code written by different team members, even when they had their unique ways of writing code. This made our code collaboration more effective.
* We gained practical experience in managing MySQL, learned how to effectively use and manage MySQL database to store, organize, and retrieve data efficiently, which is crucial for our project's success.
* We improved our understanding of all stages a project goes through, better anticipating needs further down, saving time and avoiding rework.

## Proposed Changes

* Naming Conventions: In future if we work together as a team, we will decide how we name things like files and variables. This way, we'll ensure that the names are consistent and clear for everyone. When everyone knows what things are called, it makes it easier for us to understand and work on the project.
* Stable Leadership: We'll aim to have more stable roles and leadership within our team. Changing roles every week caused disruptions, so we'll try to avoid that. So, next time, we'll try to have more consistent leaders and roles in our team. This means that the same people will lead and do specific tasks for longer periods, which should make things run more smoothly.
* In the future, we'll spend more time talking and planning before we start. This helps us understand things better and make sure we're on the right track from the beginning. One important part of this planning is creating Use Cases. These will help us outline how the system should work and what it should do from a user's point of view.
* In the future, we will enhance our collaboration by planning how we use GitLab right from the beginning. This means we will establish a clear structure for branching and merging. This includes things like when to create different branches for different tasks and when merging. By making these decisions early, we can avoid confusion and work together more smoothly. We will decide when and how to create separate branches for different tasks. Each branch will focus on specific aspects of our project, making it easier for team members to work independently without affecting the main project until their work is ready. Also, we will determine when it's the right time to merge these branches back into the main project. This is important because it's when individual contributions come together to create the final product.

## Code Snippets

### 7.1 Back-end / middleware

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Figure 7: Back-end / middleware Code Snippet

The piece of code checks if a POST request with an 'action' parameter is received (indicates that a user wants to adjust the quantity of an item in their cart). It fetches the current quantity of the item from the database. Depending on the user's 'action' (increase, decrease, or no change), it calculates the new quantity, then updates the database with the new quantity for that item. If the update is successful, the user is redirected to the cart page, reflecting the updated item quantity, if not - it will display an error message.

Our team successfully implemented dynamic cart management, allowing users to easily adjust the quantity of items in their cart. This enhances the user experience by giving them more control over their shopping cart. This code demonstrates a solid understanding of interacting with the database using MySQL and PHP. We have included an error handling to provide informative error messages in case something goes wrong, which can help us with debugging.

We proud of this code as it effectively manages the shopping cart, enhancing the user experience and demonstrating proficiency in key back-end /middle-ware development skills.

### 7.2 Middle-ware

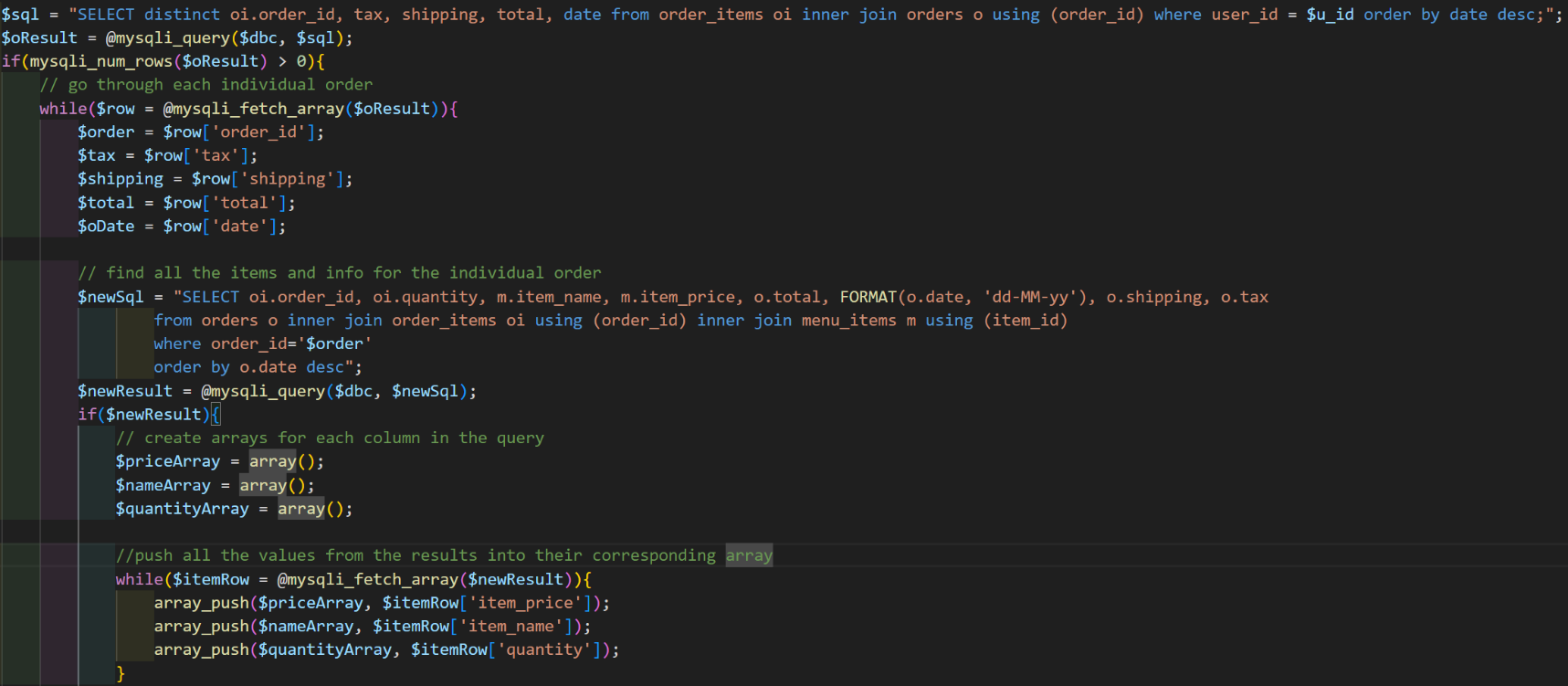


Figure 8: Middle-ware Code Snippet

This code snippet is taken from the history.php file and shows two different queries that are used to populate variables that will then be used to dynamically create an accounts order history. It uses two loops to go through both sets of data and at the bottom of the second loop there is a print statement that uses all the filled variables. Depending on how many orders the current user has this process will repeat for each individual order they have made.

We use this same loop structure every time there is a call to the database to fill variables. Most of the time it is simpler but this structure helped us implement dynamic html in many parts of the website. This particular one needs a second loop to create a few arrays because each individual order could have multiple items.

We are proud of this code because it is an example of how we have efficiently reused ideas and changed them to fit specific needs. It demonstrates our general resourcefulness and ability to efficiently tackle a task.

### 7.3 Front-end

*A computer screen shot of text

Description automatically generated*

Figure 9: Front-end Code Snippet

This is code viewed from the page source as rendered for display on the web page. We utilized Bootstrap to easily implement sophisticated design and simplify our CSS and JavaScript code by including classes into the HTML tags. This code uses Bootstrap’s simple grid design that behaves as a table by having three headers, Item, Quantity and Price Per Item. The reason we used the grid structure instead of using HTML table is to implement mobile responsive web page easily and enhance performance of PHP form that sometimes does not work depending on browser used.

We were able to implement sophisticated mobile responsive user interface that fits to this project’s product, which is food delivery, by choosing appropriate theme using warm colours and pretty background image. This site will enhance sales by making customers who visit this website want to order food here. Our knowledge of front-end became more solid through utilizing what we previously learned in the course.

Since we brainstormed over and over how website should look like to deliver the product that customer needs, we are proud of our overall look and feel of our website.