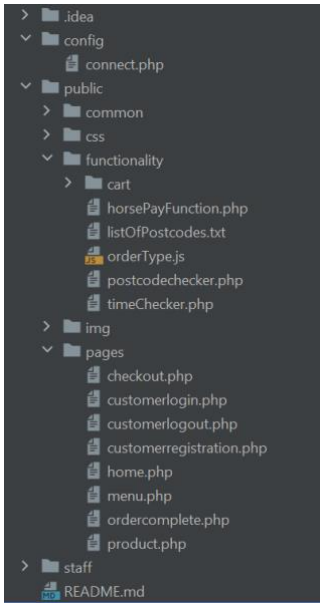
CSC8019 Team Project- Group Report

Description

The product developed enables customers to purchase ice cream through an online ordering system for an ice cream shop called Cavallo. This product comprises three basic elements, a MySQL database to store all the orders, a front-end website that incorporates JavaScript, HTML and CSS for styling. The third element is the back end, written in PHP with MySQLi which communicates with the database and creates the interactive functional elements of the website, such as the HorsePay API.

Architecture, size and complexity

In terms of the structure where there are several folders within the public folder, this allows a clear distinction between the functional aspects of the website for instance the postcode checker, the cart functions and HorsePay API functions from the main pages such as the checkout, customer login and the home page.



To, illustrate this more clearly, a screenshot has been provided. These are also linked to the database through the connect.php file in the config folder.

Development process model

Given the small size of this system the team collectively thought Agile would be the most appropriate to implement and involves users at every step as there aren't many stakeholders apart from the owners and employees of the shop, customers and delivery drivers. Another reason for this choice is because the key principle in Agile is to satisfy the customers need through early and continuous delivery of valuable software.

This process allows the deployment of the software to be much more streamlined as one way of working in Agile is the Scrum framework which comprises of assess, select, develop and review. This essentially is done through sprints lasting a fixed time between 2-4 weeks, given the time constraint in the project the team had to alter this to every 1-2 weeks.

The team also incorporated pair programming as this allowed more members to both write and supervise the code. This was managed through two processes. Firstly, the team created a Discord channel where members could all congregate together and work in pairs by sharing their screen and working on separate areas such as the Database setup, once this was completed this was then used in tandem with a project management tool Trello, which would then be updated as completed or any issues that would arise would then be posted on there.

Team coding standards

To make the written code more easily understandable by everyone in the team, coding standards were agreed on at the start of the project. The project's source code was stored in the team's GitLab repository and each team member would use a sufficient amount of commits to differentiate parts of the code or functionality with a relevant commit message explaining what was written or changed. This allowed for easier problem-solving in case parts of the system would break and for merging code.

In terms of writing code, naming conventions and commenting standards were agreed on. For all languages used, the names of variables and functions are short but descriptive. In HTML lower case with underscores and hyphens is used. In PHP, JavaScript and jQuery the variables are written in the same way *($variable_names)*, constants are written in caps with underscores *($CONSTANT_NAMES)* and functions use lowerCamelCase *(functionNames()).* In code comments, the author of the code and a description of its purpose is given.

Functionality

Initially, the user starts on the Home page where they are welcomed with a brief description of the shop including details of the opening hours and the address. The users are then encouraged to select either collection or delivery. Then in the case of delivery, to also insert their postcode to check whether it is eligible for delivery.

For this function, the team incorporated a text file with all the eligible postcodes, then created a postcodechecker.php file which contains a function with a while loop and if statement to iterate through the list of postcodes and if the postcode exists in this list it returns true, thus is accepted as a valid postcode. If the following postcode is not eligible then an error stating the postcode is not within the 5 -mile radius will be thrown. After completing this postcode checker, the user is then encouraged to click on the view menu button as shown below.

Menu and product pages

Both the menu page and product page are dynamic. This means that the system will retrieve the ice cream flavours from the database rather than it being hardcoded in HTML. Because of this, the ice cream shop can easily change the menu, remove or add flavours, change prices and surcharges or manage dairy-free options in the database without having to update the website. This allows for expandability and changing requirements.

Labels on the products indicate how many products of a specific flavour are currently in the basket. This has a positive effect on the user experience as it gives the user confirmation of what is in the basket without having to open the basket.

When a product has a dairy-free option, this is indicated with a label. To make the process of finding the desired products easier and more user friendly, a dairy-free filter has been added. The user is able to check the dairy-free checkbox, after which the menu page will only show products that are also available dairy-free.

The product page dynamically loads information on the item that has been clicked on, on the menu page. If the product has a dairy-free option, a checkbox appears which allows the user to select whether it wants the ice cream flavour dairy-free or not. The user is able to select a size using the dropdown after which the price will be updated in real-time.

Then the customer is returned to the menu page if they click on the < button and if they would like to continue to add more items they can or they can click on the basket icon and edit the number of products before going to the checkout. When an item is added to the basket uses Cookies to store the items and passes them through to the checkout.

Checkout process

Once at the checkout, the user can view all the items in the basket. If the user is already logged in the address details are filled in automatically. However, if the user is not logged in, they will be redirected to customer login once they click the checkout with Horsepay button. If the customer is not registered there is a customer registration button which will allow the customer to create an account that is saved to the database.

When the customer clicks the checkout with HorsePay button whilst logged in, several actions occur. First, a JSON object is created with the required attributes for the HorsePay API. If the response object contains information that states the transaction was unsuccessful, the user receives a notification which tells them why the transaction has failed, and they remain on the checkout page. A successful transaction will result in the user being redirected the order confirmation page whilst the following process occurs in the back end.

A session variable is checked to see if the order is for collection or delivery. If the order is for delivery, a new order is created in the database and with information about who placed it, when it was placed, and the total price. During the creation of this order, a function is called to create an entry in the delivery table of the

database with the address information. When this delivery is created, another function is called to assign this delivery to a driver and their driverID is entered into the delivery table. After this, the items from the user's basket are entered into the database and linked to the order via an orderID. Information stored about these items come from the cookie-based cartfunction and includes size, quantity, flavour, and whether the product is dairy-free. For collection orders the process is the same but without the creation of a delivery and assignment of a driver.

On the order confirmation page, the user can see a summary of their order along with an estimated delivery/pick-up time. The page uses session variables as well as database data to create a dynamic order complete page and dynamic order confirmation email that is sent to the customer. Upon reaching this page, the cookies which contained the items in the basket are cleared and therefore the customer's basket is emptied. One way in which this functionality could be improved is the use of tracking order distance real time.

Analytics and Staff Login

The analytics page created is an element on the website only accessible by staff through the single login information. This page displays the statistics of the company's orders through tables and graphs. These include: the number of orders made in the last day/week/month; the popularity of the offered flavours; the weighting between orders placed for collection and those for delivery; how many orders are made on specific days of the week; and finally, a date selector which shows you all the orders made in the time frame specified. Chart.js was used for data visualisations including pie charts and a bar charts.

The use of PHP allowed the statistics to update in real time, allowing staff to monitor the daily progress Cavallo is making. The analytics page provides insight into consumer habits which can be valuable for future business growth.

Testing

Different types of testing were done on the system to ensure functionality and quality of the code. Mostly if-else statements were used together with echo statements as well as entering false values. The system has been tested during different times to ensure checkout is disabled outside opening hours. Responsiveness of the site was also taken into account by testing the system on different devices such as desktop and mobile.

Unit testing was done through the pairs which worked on different sections. For instance, the surcharge was not being added to the price on the Product page, so the team tried to find the error by using echo statements to print the price and the surcharge separate and to discover which one was not working. This testing was successful because a problem was found and solved.

Another bug found during testing was, on the menu page, the label that indicates the quantity of a certain flavour in the basket did not reflect the correct amount when the quantity was increased in the basket. This has also been fixed by using echo statements to print out Array values and find the problem.

The login forms for staff and driver have been tested by entering invalid logins and passwords and ensuring the staff and driver pages aren't accessible to a user that does not have the correct login details.

The analytics dashboard was tested by entering fictional data in the database and ensuring the different dates are illustrated on the diagram. The system has been tested in multiple browsers including Safari, Chrome and Firefox and the team had come to the conclusion that the initial functionality of advanced statistics did not work in Safari. This has then been resolved.

In terms of the HorsePay functionality, testing has been done to ensure the JSON objects were passing through and printing on HorsePayFunction and that transaction declined was operating. Mostly, echo statements were used to see how far the function would run through the code.

The team also tested that the collection/delivery selection was saved in the session variable and accessible throughout the entire system.

At the end, Big Bang testing ensured all the individual components of the system work together.

Peer Review

A	B	C	D	E	F
Student full name	Work contributed	Team participation	Professional Behaviour	Communication	**Final Peer Review**
Alan Cherian	10	10	10	10	10
Alice Hughes	10	10	10	10	**10**
Danyang Zhang	10	10	10	10	**10**
Du Hao	10	10	10	10	**10**
Dylan Graham	10	10	10	10	**10**
Josh Cairns	10	10	10	10	**10**
Kimberly Dijkmans	10	10	10	10	**10**
Mingwei Li	10	10	10	10	**10**
Manon Myung	10	10	10	10	**10**
Zhongyuan Dai	10	10	10	10	**10**
Total	100	100	100	100	**100**

The team decided an equal weighting of 10 across the board, this decision was brought about through an extensive conversation through Teams. Each member had to provide evidence of what they contributed to the team and which sections they worked on, this was checked through our project management tool Trello which recorded all the tasks completed and which members completed them.