Technical report of project Bio & Health

## **Andreas Nalmpantis – 3699, Ioannis Oikonomidis – 3668,**

## **Chris Papakonstantinou – 2499, Iordanis Kokkinidis – 3412**

## **29th of May 2022**

# Chosen Topic

Our web app is an e-shop where various organic products are sold as well as various other products related to the improvement of physical and mental health. We chose this project because we believe e-shops are a very important type of web app, especially in the aftermath of the pandemic. We are also all passionate about nutrition and organic products. Additionally, we feel that gaining the experience of creating a website for an e-shop is very useful in the current market and will help us in our careers.

# Web App Description

* Our web app supports two types of accounts, administrators, and users/consumers.
* Administrators can add, delete, and edit products and categories to our web app’s database as well as do many other activities.
* Users can also add products to their cart and then place orders based on the contents of their cart. Users also have many more capabilities which will not be analyzed in this section of the report.
* Our web app has a navbar with different content based on if the account is an administrator or a consumer or the visitor is not signed in.
* The main page which is named Index.php contains image links to the different types of categories that lead to the search page showing only the chosen category as well as image links to the search page showing only products on offer, new products and an image link leading to the page with more information about the web app.
* A contact page is also included where consumers can submit a contact form and send it to the owners of the web app.
* There are also a lot more forms in the web app, such as one for editing the characteristics of products, one for the checkout when ordering, ones for signing in and creating an account and many more.
* On the navbar there is always a search bar that searches in the app’s database for the search term and links to the search page which shows all the products that contain the search term in their name or in their category name.
* Additionally, the app has a footer which is the same for every page and contains four icons that link to the main pages of the respective social media platforms they show. Since we don’t actually have any social media pages for the shop, we have set them up to just link to the social medias’ main pages.
* Lastly, our web app is optimized for mobile devices by using bootstrap as the main library for developing the CSS of the front end as mentioned in the report of the second part of the project.

# System Architecture

The frontend of the web app is written using html 5, CSS, Bootstrap and Javascript. More details about the structure of the front end have already been given in the report of the second part of the project.

The backend of the project is written using PHP. We also used a few lines of Jquery code to send requests to php scripts. The database is written in MySQL and we have used XAMPP as a server and its database system.

# Database description

ER Diagram:

Graphical user interface, application

Description automatically generated

Table description:

* **user**: a table that holds the main information about the app’s accounts. This is where the app’s owners can add a record with the role ‘Administrator’ to create a new admin account.
* **user\_info**: a table that holds the info users with the role ‘Customer’ input when they make their first order to auto fill it in when they make more orders. This info can be edited by the customer (only their info can be changed) from the ‘Order Information’ option of the navbar’s pop up menu shown when hovering over the user icon.
* **shipping**: table that saves the cost and max delivery time of the shipping method the customer with the corresponding ‘user\_id’ has currently selected in their cart.
* **cart\_item**: table that saves the information of all the products that are currently placed in the customer’s cart with the corresponding ‘user\_id’.
* **product**: table that saves each product’s characteristics. The ‘image’ field is a relative link to the product’s image.
* **offers**: table that saves the currently available offers by saving the ‘product\_id’ that is on offer as well as the percentage of the offer.
* **category**: table that saves the information of each of the current product categories in the database.
* **order**: table that saves each order in the database. Pending orders have the field ‘completed’ set to 0 and are shown in the pending orders page, while completed orders have that field set to 1 and are not shown in the pending orders page, but are not deleted from the database and can be accessed by looking at the contents of this table.
* **billing\_address**: table that saves the data inputted by the consumer in the ‘Billing Address’ part of the checkout when making an order. The primary key is the foreign key ‘order\_id’ of the ‘order’ table to bind the data to the specific order with that id.
* **payment\_method**: table that saves the data inputted by the consumer in the ‘Payment’ part of the checkout when making an order. The primary key is the foreign key ‘order\_id’ of the ‘order’ table to bind the data to the specific order with that id.
* **shipping\_address**: table that saves the data inputted by the consumer in the ‘Shipping Address’ part of the checkout when making an order. The primary key is the foreign key ‘order\_id’ of the ‘order’ table to bind the data to the specific order with that id.
* **order\_item**: table that saves the information of all the products that are included in the customer’s order with the corresponding ‘order\_id’.

# Run instructions

1. First, set up a XAMPP server following the exact instructions you gave us in the e-Learning page.
2. Move the contents of the ‘bio-health’ folder included in the zip file we uploaded to e-Learning to the ‘htdocs’ folder located inside the installation folder of the XAMPP server.
3. Make sure to overwrite any files when prompted during the moving process.
4. Start the XAMPP Control Panel with administrator privileges and start the Apache and MySQL modules as shown in the picture:

Graphical user interface

Description automatically generated

1. Head to the PhpMyAdmin page by clicking on the ‘Admin’ button of the MySQL module of the XAMPP Control Panel.
2. Once there, click on the ‘SQL’ tab, copy the contents of the ‘db.sql’ file located in the ‘bio-health’ folder you moved to the ‘htdocs’ folder of your XAMPP installation (this is the source file of the app’s database which includes both the MySQL code that builds the database as well as ‘INSERT’ queries that add data such as products to the database) and paste them in the text field that runs queries.
3. If done successfully, there should be a new database named ‘biohealth’ on the list of databases shown on the left, as shown on the image below:

Text

Description automatically generated with medium confidence

1. Next, either click on the ‘Admin’ button of the Apache module of the XAMPP Control Panel or head to the browser of your choice and type ‘localhost’ in the url box. If done correctly this should take you to the ‘Index.php’ page which is the main page of the app (the URL will still say ‘localhost’, you can click the ‘Bio & Health’ logo of the navbar to go to the URL ‘http://localhost/Index.php’, which is the actual URL of the main page). If for some reason this doesn’t work (for one of us the ‘localhost’ page redirected to ‘localhost/dashboard’ instead of our ‘Index.php’ page), just copy the URL ‘http://localhost/Index.php’ to the URL box.
2. Now both the Apache server and the MySQL database are up and running and you are placed at the main page of our web app.

# Περιγραφή υλοποιήσεων του 3ου μέρους

## Search bar

The search bar located in the navbar is now fully operational and upon clicking the magnifying glass button, you are redirected to the ‘Search.php’ page with the applicable get parameters where all the products found in the database with a matching name or category name are listed. The products listed are refreshed upon changing the input of the search bar if you are on the ‘Search.php’ page.

If an account is not logged in, the visitor can only see the products and go to the information page by clicking on their image or name and stock information. If the account is logged in as a customer, they can also add each product to their cart by clicking the applicable button and this is also now operational. Lastly, if logged in as an administrator, you can edit each product’s information or delete it from the database altogether. There are also two filters available, a filter for the categories of the products and a filter to sort the products.

Updating the search results when coming from a different page is done with the php script located inside the div element with the id ‘products’. Updating the search results when already in the ‘Search.php’ page is handled by the four javascript scripts located at the end of the head of the ‘Search.php’ file by using jquery and making a get request to the php script located in ‘PHP\_Back\_End/search.php’ which updates the search results. The categories and sort types are echoed by the scripts located inside the select tags with ids ‘category’ and ‘order’.

Lastly, when a product is added to the cart or is to be deleted, the applicable javascript function from the file ‘JS/search.js’ which also makes a get request to the ‘PHP\_Back\_End/search.php’ script for adding to cart and to the ‘PHP\_Back\_End/delete\_product.php’ script for deleting the product. All of the search functionalities use the function ‘updateSearchResults’ located in the ‘PHP\_Back\_End/search\_functions.php’ script by including it to search for the products and echo them.

## Cart

The cart is now fully operational. The products added to it are shown based on the database’s ‘cart\_item’ table with rows that have the same ‘user\_id’ as the account that is logged in. This is all achieved by the php script located in the div element with id ‘products’ in the ‘UserCart.php’ file.

Cart items can have their amount changed on the cart page, this is done by calling the applicable javascript functions located in the ‘JS/cartpage.js’ file which send get requests to the ‘PHP\_Back\_End/edit\_cart.php’ php script with the needed parameters which in turn makes the needed changes to the database.

When the ‘Proceed to checkout’ button is clicked, the javascript function ‘updateShippingType’ sends a get request to the ‘PHP\_Back\_End/edit\_cart.php’ php script which inserts into the database ‘shipping’ table or updates the cost and max delivery time of the chosen shipping method.

## Checkout

The checkout is also now fully operational. We have also added a ‘Phone number’ field. The fields ‘Full name’, ‘Email’, ‘Phone number’, ‘Address’, ‘City’, ‘State’, and ‘Zip code’ are auto completed after the account’s first order.

The autocompletion is done by the php script located at the start of the body of the ‘UserCheckout.php’ file. If the info has not been saved yet, the fields that have no data are left empty. The total cost of the order is calculated and echoed by the php script inside the span element with id ‘pay\_val’ in the ‘UserCheckout.php’ file.

Once the form of the ‘UserCheckout.php’ is submitted by pressing the ‘Place order’ button, a post request is sent to the "PHP\_Back\_End/checkout.php" php script which inserts all the data inputted by the customer into the database and then redirects to the ‘UserOrderCompletion.php’ page with the id of the order as a get parameter in the url which shows a message confirming the completion of the order as well as the id of the order.