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/customers.
* 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 customer 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 customers 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 front-end 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 back-end 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 database management system.

# Database description

ER Diagram:

Diagram

Description automatically generated with medium confidence

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 customer 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 customer 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 customer 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’.
* **new**: table that saves the product ids of the products that are new.
* **user\_favourites**: table that saves the ids of the favorite products of each user.

# 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 ‘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 ‘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.

# Description of developed functions of the 3rd part

## Navbar and footer

In the third part we have merged the previous two navbar files, the ‘AdminNavbar.html’ and the ‘UserNavbar.html’ to a single file named ‘Navbar.php’. This file is included in each page file around the start of the body and at the top of it starts a session so the session variables are always available on every page.

Inside the ‘Navbar.php’ file there are several different php scripts which among other functions, control what icons are echoed based on the role of the signed in account or the lack of signed in account, what drop down menu options are echoed based on the sign in status, which options are echoed on the ‘burger’ menu based on whether the account is logged in as an administrator or not.

The different category options are also dynamically echoed using the categories saved in the database when not signed in as an administrator. The ‘Offers’ and ‘New Products’ are statically shown since they can’t be removed as categories. Lastly, the search bar has some differences in the html code based on whether the page the navbar file is included in is the search page or not.

We also merged the previously different footer files into a single file named ‘Footer.php’ which does not have any php code inside it since it is the same for all pages.

## The ‘PHP\_Back\_End/db\_connection.php’ file

This php script is included wherever a connection to the database needs to be established. It simply connects to the web app’s MySQL database using the XAMPP’s default account with the hostname being ‘localhost’, the username being ‘root’, the password being an empty string and the database name being ‘biohealth’. If for some reason you cannot connect to the database because of the account info being wrong, you can change it in this php script and the change will apply to the whole web app.

## Sign Up and Sign In

Sign up and sign in functionality is now connected to a database and creates and stores the information given by the user.

In order to sign up, the user has the click the “Create an account” link, which is in the dropdown menu of the navbar and appears only when the user is not signed in. He is then redirected to the ‘UserSignUp.php’ page, where he fills all the necessary inputs and submits.

The information submitted is sent to the ‘sign\_up.php’ file which is in the ‘PHP\_Back\_End’ folder. There, a connection gets established along with a session, and if those credentials don’t already exist in the database, then a new entry gets added in the ‘user’ table with the credentials given, along with a new (empty) entry in the ‘user\_info’ table which will later contain the user’s information required to make an order. After signing up, the user’s id, username and role is stored in session variables which allows his information to be distributed easier to the rest of the files.

In order to sign in, the user has the click the “Sign In” link, which is also in the dropdown menu of the navbar and appears only when the user is not signed in. He is then redirected to the ‘UserSignIn.php’ page, where he fills all the necessary inputs and submits.

The information is sent to the ‘sign\_in.php’ file, which checks if there is an entry in the database that matches the credentials given. If there is, then the user is signed in, and his information is stored in the session variables.

If you want to sign in with an account that already exists in the database, take a look at the values inserted into the ‘user’ table of the ‘db.sql’ file which creates the database. From there you can copy the username/email and the password of the account you want to sign in as and paste them in the sign in page to sign in as that account.

## Admin sign in/sign up

We decided not to allow visitors of the site to create admin accounts on from the web app as that would obviously be a major security risk. Instead we have added a single account with administrative rights in the file which creates the database ‘db.sql’.

In the ‘user’ table of the database, a user with the following values is inserted when the database is creating using the ‘db.sql’ file:

* user\_id: ‘7929791AD63B5’
* username: ‘admin’
* email: ‘admin@bioandhealth.gr‘
* password: ‘5rUWqHq2jH’
* role: ‘Administrator’

This is the only account with administrative rights in the database when it is created. If you want to sign in as an admin and take advantage of the administrative functions, you need to login with the above email or username and the above password.

Once logged in, the username, email and password of the admin account can be changed from the ‘Account Management’ option of the navbar.

Additionally, if you want to create another account with administrative rights, you can insert a row to the ‘user’ table of the database with the field ‘role’ set to ‘Administrator’ from a page that handles the web app’s database, such as the PhpMyAdmin page of XAMPP. The ‘user\_id’ of any new accounts inserted manually must be a 13 character string containing only numbers and the capital letters A – F.

Lastly, in order to delete an account with administrative rights, simply delete the row with the associated account from the ‘user’ table of the database. If you do this, also make sure to delete the row with the same ‘User\_ID’ as the deleted account’s ‘user\_id’ from the ‘user\_info’ table of the database in case there is such a row.

## Sign Out

To sign out, the user must click the ‘Log out’ button, which is in the dropdown menu of the navbar and only appears when a user is signed in. The ‘sign\_out.php’ file handles the sign out functionality. It unsets and destroys the current session.

## Account Information

We used account information to describe the email, username and password given by the user when they create their account. These are the credentials that they needs to sign into their account.

If they want to view or change that information, the users can click the ‘Account Management’ link, which is also located in the dropdown menu of the navbar. After the users have made the changes they want, they can click the “Save changes” button, for the information to be updated in our database.

This link opens the ‘UserAccountManagement.php’ page, which is a new page that was not created at the 2nd part of the project. There, they can change their email, username, and password to whatever they want.

We understand that this is not really secure, but we decided to proceed with it, since we did not have a lot of time to do something more complex.

## Order Information

The order information is the information required when the user is making an order. For example, their address, zip code etc...

To view, change, or delete that information from the database, the user must click the “Order Information” link, located in the dropdown menu of the navbar. They are then redirected to the ‘UserOrderInfo.php’ page. Then, the user can make any change that they want and save it by clicking the “Save changes” button.

The information is sent to ‘order\_info.php’ file which is in the back-end folder. There, a query which updates all the information accordingly gets executed and the user is sent back to the main page.

## Add products

Only the administrator can add products. In order to do that, they have to click the “Add products” link, which is located in the ‘burger’ menu of the navbar. These options get displayed only when the account is logged in as an admin.

They are then redirected to the ‘AdminNewProduct.php’ page. In this page, the admin can add both a new product and a new category to the database. On the left of the page, a list with all the categories is displayed along with a link to add a new one. If they click one of the category links, then the appropriate text gets typed in the ‘Category’ input of the “Add a new product” form. Just a small shortcut to help the admin.

When they click the “Add a new category link”, a small pop-up form appears, which requires a name and an image for the category. This is done using a little bit of Javascript, located in the ‘new\_product.js’ file in the ‘JS’ folder. The ‘openForm’ function makes the pop-up appear and the ‘closeForm’ makes it disappear. Other than that, this file is also responsible for filling the text in the ‘Category’ input when the user clicks one of the category links.

After this information is given, the admin clicks ‘OK’, which submits the information to the ‘new\_category.php’ file which adds a new entry in the ‘category’ table of the database.

In order to add a new product, the admin has to fill all the form’s inputs and then click the “Submit” button. The information is sent to the ‘new\_product.php’ file which adds a new entry to the ‘product’ table of the database.

## Edit a product’s information

This functionality can also only be used from the admin. In order to edit the information of a product, the admin can click the ‘Edit products’ link, which is located in the burger menu of the navbar, right next to the ‘Add products’ link.

They are then redirected to the Search page, which displays all the available products of the shop. If they want to edit one, they can click the ‘Edit’ button which appears under the product’s price.

Then, they are taken to the ‘AdminEditProduct.php’ page which contains a form which gets automatically filled with the information already submitted for the product when it was added or last edited. After all the changes are made, they are submitted by clicking the ‘Submit’ button.

The information is sent to the ‘edit\_product.php’ file, located in the back-end folder, which updates the appropriate entry in the ‘product’ table.

## Search bar and search page

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.

Pressing the ‘Finish’ button after the ‘Add to cart’ button, updates the amount of the given product on the cart to the amount chosen on the modal. So if there were 3 units of a product, changing the amount on the modal from 3 to 5 would make it so the cart has 5 total units of that product. This is why when a product is not already in the cart, it has 0 on the amount in the modal.

## 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.

## Pending orders

The pending orders page the administrators can access by clicking the ‘Pending orders’ option of the navbar’s menu is now operational. Clicking on the arrows next to the table header names sorts that column in the opposite order of what the arrow is currently showing. If the arrow is pointing up, clicking it will make the arrow point down and sort that column by descending order. This is done by the php script inside the ‘thead’ element of the ‘AdminPendingOrders’ file.

The loading of the rows of the orders table is done by the ‘PHP\_Back\_End/pending\_orders.php’ php script which is included inside the ‘tbody’ element of the ‘AdminPendingOrders’ file. This script handles updating the shown orders based on the selected column for sorting and the sort type as well as updating the order’s ‘completed’ field to 1.

The get requests to set an order to complete and to update the orders are made by the ‘completeOrder’ and ‘updateOrders’ functions of the ‘JS/completedorderspage.js’ javascript file which are called when clicking on the respective elements of the table. Clicking on the ‘Details’ button redirects to the order details page with the ‘order\_id’ of the specific order as a get parameter.

## Admin order details

The admin order details page now shows all the info stored about an order instead of showing some default text like it did in the second part. The ‘Billing Address’, ‘Payment’ and ‘Shipping Address’ sections have their field values loaded from the database by the php script inside the section element with id ‘form’ of the ‘AdminCart.php’ file.

The ‘Shopping Cart’ and ‘Pricing’ sections are echoed by the php script inside the section element with id ‘cart’ of the ‘AdminCart.php’ file.

## Favorites

The favorites page now correctly displays the products the signed in user has added to their list of favorites and allows them to add them to the cart or remove them from the favorites list. Fetching and echoing the favorited products is done using the script inside the div element with the id ‘products’ of the ‘UserFavorites.php’ file.

The removal of a product from the favorites is done using the script ‘PHP\_Back\_End/favorites.php’ which handles removing the product with the given ‘product\_id’. This is done by sending a get request to this file with the ‘product\_id’ parameter with the javascript function ‘removeFromFavorites’ of the ‘JS/addToFavorites.js’ file.

## Product info

The product info page now correctly displays the information of the product whose ‘productID’ is passed as a get parameter to the page. The information is fetched from the database by the php script at the start of the body element of the ‘UserProductInfo.php’ file.

Additionally, the ‘Add to cart’ button and the cart and favorites icons are only visible when logged in as a customer. The add to cart mechanism works the exact same way as the add to cart mechanism of the search page.

## Contact Form

The user can submit a contact form to the website’s owners by clicking the ‘Contact Form’ link, located in the footer of any page.

They are then redirected to the ‘UserContactForm.php’ page. There, they fill in all the required inputs, type in their message and click ‘Submit’.

The information is sent to the ‘UserContactFormCompletion.php’ page which sents an email, using the php mail function. Also, an appropriate response is echoed for the user, to notify them whether the message was sent successfully or not.

Then, they can go back to the main page by clicking the ‘Back to shop’ button.

## Information about the shop

The ‘UserMoreInfo.php’ page contains information about the Bio&Health shop. The user can go to that page by clicking the ‘ABOUT US’ image located at the bottom of the index page, along with the offers and new products links or by clicking the ‘More Info’ link in the footer.

The ‘Terms of use’ and ‘Personal data’ also lead to this page, since we don’t actually have any terms of use or personal data policies to link to.

## Index

The index page has the correct images in place of the previous placeholder pictures and all the links that previously led to the search page, have been updated to lead to the search page with the correct category filters.

# Functions we had not implemented in the 2nd part

Pretty much everything mentioned in the previous part was already there in the 2nd part but it was filled with placeholder assets. For instance, instead of the products in the search being loaded dynamically, we had some set products hard coded into the html code.

This has changed now and everything that was previously cosmetic only, now works. Thus, the previous part also acts as a list of functions that we fixed in the 3rd part which did not work correctly in the 2nd part.

# Functions we did not have time to implement/decided to leave out

We decided not to implement the ability to sign up or sign into the web app with social media accounts, since after researching how to actually do it, we found it hard to do and thus removed it altogether from the sign up and sign in screens.

# Notes

Some of the names of the files we have used would be better changed in the 3rd part, however because we already used them in the 2nd pard we decided to keep them as they were so that it would be easier to identify how our 3rd part pages match up with our 2nd part pages.

We also used the info about products from the website ‘biologiko xorio’. This is because it would be incredibly time consuming to find correctly sized pictures and information for original products and it would not anything to improve our site.

Since this web app will not be uploaded to a server and go online, there should be no issues with copyright.

# References

Site of ‘biologiko xorio’: <https://www.biologikoxorio.gr/en>