



SILESIA UNIVERSITY OF TECHNOLOGY
FACULTY OF AUTOMATIC CONTROL, ELECTRONICS AND
COMPUTER SCIENCE

Internet Technologies – project work

e-commerce site

Authors:

Wojciech Obrzut

Justyna Neblik

Year 3, sem. V, group 7 , section:

Project supervisor: mgr inż. Oliwia Krauze

Gliwice, December 2021

Table of Content

1. Introduction.....	3
2. Aim and scope of the project	3
3. Schedule	3
3.1. Schedule approved at the beginning.....	3
3.2. Schedule reflecting actual work	3
4. Software implementation.....	4
4.1. Defining the problem.....	4
4.2. Analysis of possible solutions	4
4.3. The proposed solution.....	4
4.4. Implementation	5
4.5. Problems during application development	15
5. Summary.....	15
6. Literature	15

1. Introduction

The goal of our project is to create an e-commerce website. The theme of our site is a fashion brand shop for different age groups and genders. To complete this project we had to create all sorts of functions and databases to make the site functional. The foundation of our project is to create a website that allows you to purchase items, enter and export data into databases for the store's operation.

2. Aim and scope of the project

Before we started the project, we considered what functions an online store should have. It was important that our project should include elements known to us from the websites of international fashion brands.

Our prototype was other websites that we use to make our purchases. We have identified a few mandatory page elements:

- home page
- Login / registration panel
- Products page
- Product catalog
- Wishlist
- Cart
- Checkout

To incorporate these elements into our site, we had to read tutorials to get acquainted with the technologies used to create these pages. After searching for what we would need to write this page, we decided to make a schedule of the stages of our work on the project. Determining what solutions we will use was challenging for us because we had little to no knowledge of creating websites with an extensive database.

3. Schedule

3.1. Schedule approved at the beginning

- creating a frame for the site with HTML and CSS (15.10-28.10)
- creating customer account formula, registration, and sign up PHP (29.10-11.11)
- adding products to the site, price, etc (12.11-25.11)
- creating basket and checkout, payment methods (26.11-09.12)
- adding filtering criteria ect(10.12-17.12)
- taking care of the esthetic side of the site (Bootstrap) (05.01)

3.2. Schedule reflecting actual work

We simultaneously developed the backend as well as the styling part of this site, therefore, we used the last week to touch up issues with our checkout system e.g. improving the checkout fill-out form, saving data to a database. Also, we didn't add payment methods to our final version of this checkout. Every part was made on time according to the schedule.

4. Software implementation

4.1. Defining the problem

The first feature is the ability to log in and register new customers. To create this function we had to link our website so that we could both enter and read data from the database. For the registration and login form to work correctly, we needed to set some restrictions on the data entered into the fill-out form. Examples of such requirements in the registration fill-out form are the requirement to enter certain characters in the login field, the correct format of the e-mail address (FILTER_VALIDATE_EMAIL), entering the same password twice, or checking whether all fields are filled out. For the login form, we check if the login and password match the data in the database. We have implemented safeguards against data leakage, a hashed password which makes reading the password without the proper tools impossible. Hashing the password turns a normal password into an unreadable string of characters.

An important step in creating our project was a system for filtering products by color, price, gender, and clothing type. To filter the correct price range we created a slide bar with two sliders, and for the rest we used checkboxes. The backend part consists of if statements that filter out some of the products that do not match the selected values.

The final part and culmination of our project is the basket, checkout, and wishlist page. After pressing the "add to cart" button an item goes to the cart where its quantity, price, and price for the whole shopping is counted. When we go to checkout, we fill out the form and on the side, there is a preview of our products in the shopping cart. After pressing the button, our data is sent to the database and we get a message about the successful order along with a summary. The wishlist is a simplified version of a shopping cart with a "delete", "add to cart", link to the product page button.

4.2. Analysis of possible solutions

One of the possible technical solutions was to use firebase as a database program, but we choose PHPMyAdmin because this application uses MySQL, which we got acquainted with during our studies. Another advantage of PHPMyAdmin was the popularity of this tool.

One of the next choices was whether to use plain CSS or SASS. CSS seemed more understandable to us as we are beginners. For our project SASS seemed unnecessary for our simple project.

The next thing we had to consider was whether to make our carousel or use an owl carousel. We agreed to have a self-written carousel because we found out about the owl carousel after we had written the code for our carousel. We didn't want to change something that worked well.

4.3. The proposed solution

One of the assumptions of our project is that the website should not only work but also be clear and pleasing to the eye. That's why we used a range of programming languages as well as libraries. To write our project, we used Visual Studio Code, xampp, MySQL database, phpMyAdmin, programming languages PHP, javascript, AJAX, HTML, CSS, jquery library, bootstrap library. We have added libraries to facilitate the design of the visual and functional aspects of the website, especially the product catalog and checkout.

4.4. Implementation

a) home page



Image Nº 1, navigation of the site while being logged out



Image Nº 2, navigation of the site while being logged in,
visible hover effect in categories

```
<?php
if (isset($_SESSION["userid"])) {
    echo "<li><a href='wishlist.php'> Profile </a></li>";
    echo "<li><a href='includes/logout.inc.php'> Log out </a></li>";
}
else {
    echo "<li><a href='signup.php'> Sign up </a></li>";
    echo "<li><a href='login.php'> Log in </a></li>";
}
?>
```

Image Nº 3, code for the change of links according to the session status

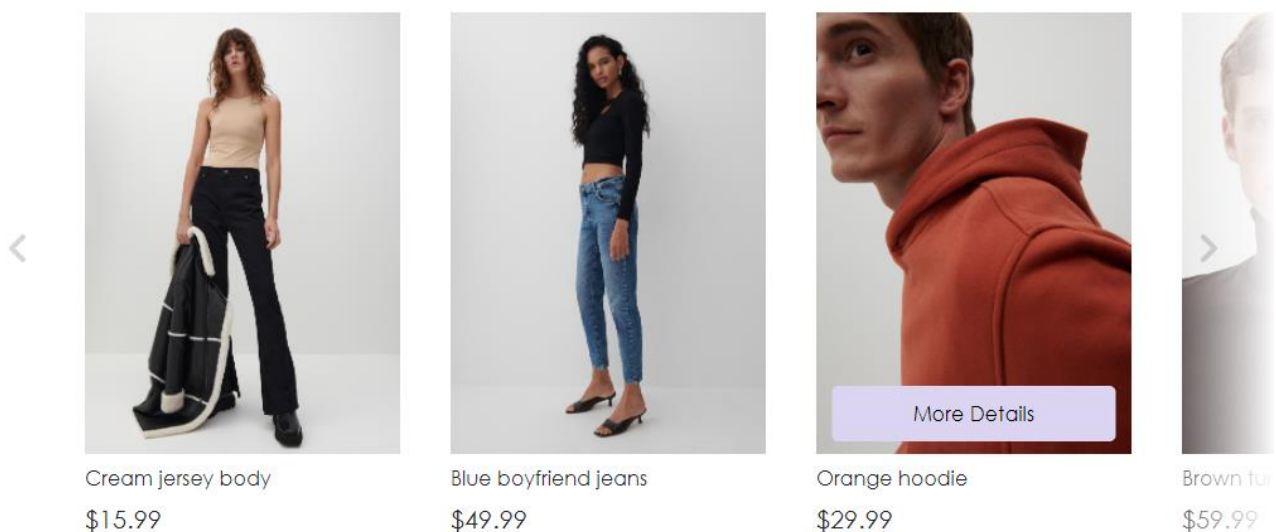


Image Nº 4, our product carousel, visible "more details" hover tag

To create this image carousel, we used a separate PHP file for extracting information from the database and a JS file for animation.

```
const productContainers = [...document.querySelectorAll('.product-container')];
const nxtBtn = [...document.querySelectorAll('.nxt-btn')];
const preBtn = [...document.querySelectorAll('.pre-btn')];

productContainers.forEach((item, i) => {
  let containerDimensions = item.getBoundingClientRect();
  let containerWidth = containerDimensions.width;

  nxtBtn[i].addEventListener('click', () => {
    item.scrollLeft += containerWidth;
  })

  preBtn[i].addEventListener('click', () => {
    item.scrollLeft -= containerWidth;
  })
})
```

Image № 5, JS code for moving this carousel using left-right arrow buttons

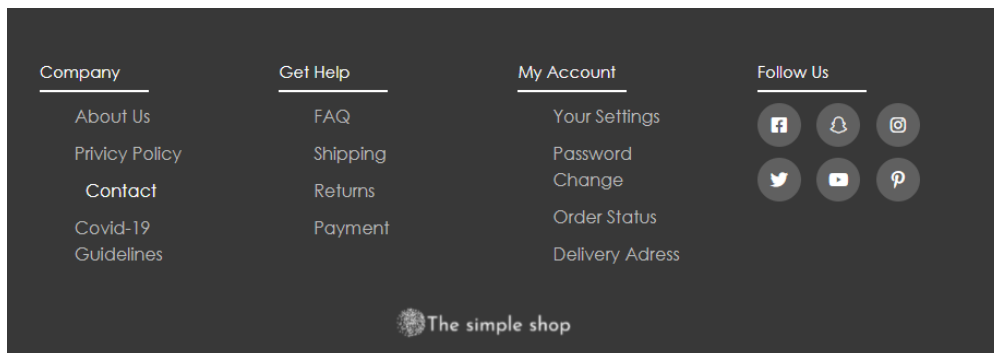


Image № 6, footer for full screen, visible hover effect on “contact”

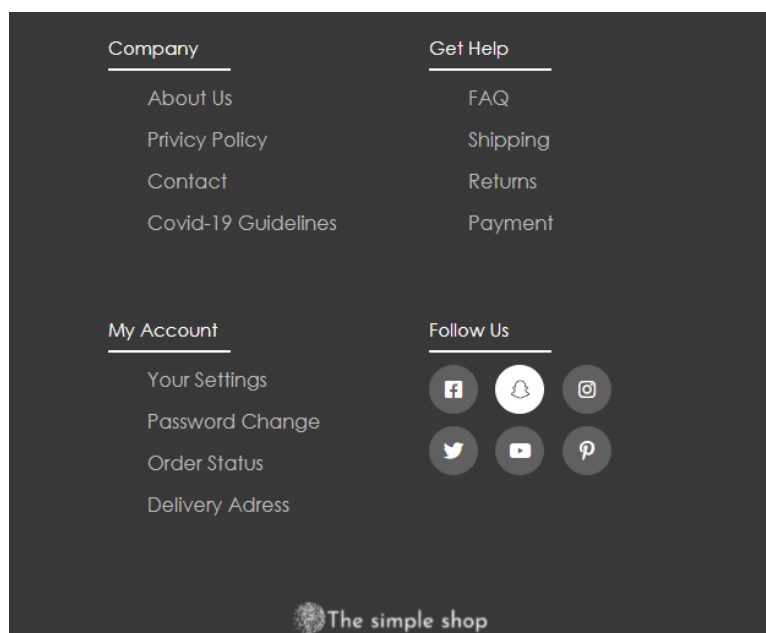
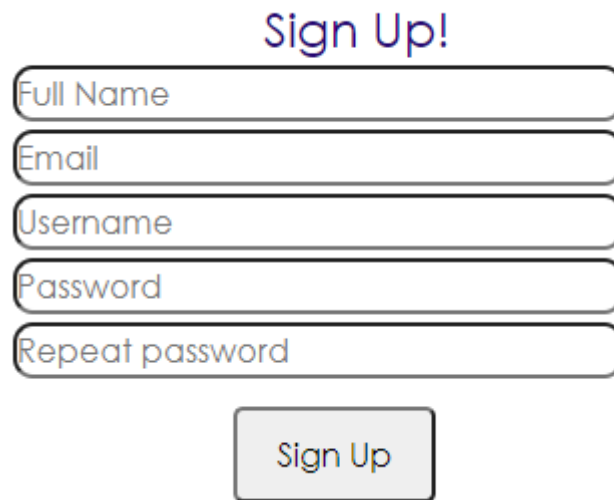


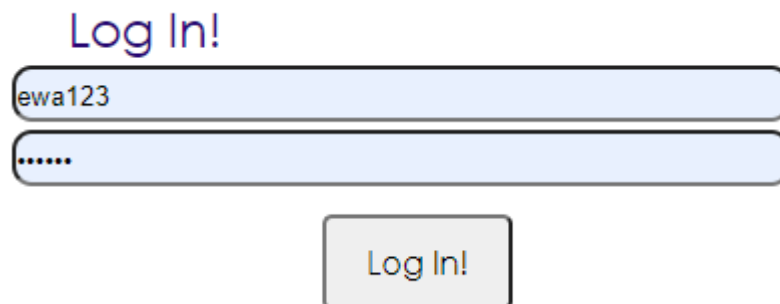
Image № 6, footer for the sized down screen, visible hover effect on Snapchat icon

b) Login / registration panel



The 'Sign Up!' form features a purple title at the top. Below it are five white input fields with rounded corners and thin borders, stacked vertically. The labels 'Full Name', 'Email', 'Username', 'Password', and 'Repeat password' are placed inside the first four fields respectively. A light gray button with rounded corners and a thin border, labeled 'Sign Up' in a dark font, is centered below the input fields.

Image Nº 7, sign up form



The 'Log In!' form has a purple title at the top. It consists of two white input fields with rounded corners and thin borders, stacked vertically. The first field contains the text 'ewa123', and the second field contains six dots, indicating a password. A light gray button with rounded corners and a thin border, labeled 'Log In!' in a dark font, is centered below the input fields.

Image Nº 8, login form

This function is possible thanks to several files which we will describe below:

- includes\dbh.inc.php – connection to the database
- includes\functions.inc.php – the backend file of signup and login form with requirements for filling out etc.
- signup.php – signup site, the visible part
- login.php - login site, the visible part
- includes\login.inc.php - connects dbh.inc.php, functions.inc.php, and login.php, triggered by clicking “login!” button, spits out error messages in HTTP header
- includes\logout.inc.php – logging out function, destroying the session
- includes\signup.inc.php – the same as login.inc.php only for signup

```
function invalidEmail($email) {
    $result;
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)){
        $result = true;
    }
    else {
        $result = false;
    }
    return $result;
}
```

Image № 9, an example of a requirement for the signup form, in this one it is for checking if we entered a valid email address

```
function uidExists($conn, $username) {
    $sql = "SELECT * FROM users WHERE usersUid = ? OR usersEmail = ?;";
    $stmt = mysqli_stmt_init($conn);
    if(!mysqli_stmt_prepare($stmt, $sql)) {
        header("location: ../signup.php?error=stmtfailed");
        exit();
    }
}
```

Image № 10, checking if the email or username is in the database for the login form

```
function createUser($conn, $name, $email, $username, $pwd) {
    $sql = "INSERT INTO users (userName, usersEmail, usersUid, UsersPwd) VALUES (?, ?, ?, ?);";

    $stmt = mysqli_stmt_init($conn);
    if(!mysqli_stmt_prepare($stmt, $sql)) {
        header("location: ../signup.php?error=stmtfailed");
        exit();
    }

    $hashedPwd = password_hash($pwd, PASSWORD_DEFAULT);

    mysqli_stmt_bind_param($stmt, "ssss", $name, $email, $username, $hashedPwd);
    mysqli_stmt_execute($stmt);
    mysqli_stmt_close($stmt);
    mysqli_close($conn);
    header("location: ../signup.php?error=none");
    exit();
}
```

Image № 11, if we successfully fill out the signup form, our site will import the data using the code visible in the image also the password is being hashed

```
if (uidExists($conn, $username) !== false) {
    header("location: ../signup.php?error=usernamealreadytaken");
    exit();
}
```

Image № 12, code from includes/signup.inc.php, if statement that displays error message in HTTP

usersId	usersName	usersEmail	usersUid	usersPwd
1	ewa	ewa@gmail.com	ewa123	\$2y\$10\$vUYT28RrZl1kxEQqJc6pF..Km9zpFx6HFKJa.X0/F/...
2	ala ali	alaali@gmail.com	alaala	\$2y\$10\$O1dyFf2ZX2Ludi05idqhtuBBMAoGLfZt.uXOYRMi6uy...
3	asdf	asdf@gmail.com	asdf	\$2y\$10\$mxr.3Jk8Yn4i8QowttfgAOI/7PCRVC5PEtneYFXmF/9...

Image N° 13, example of user data, example of how a hashed password looks like

c) Products page

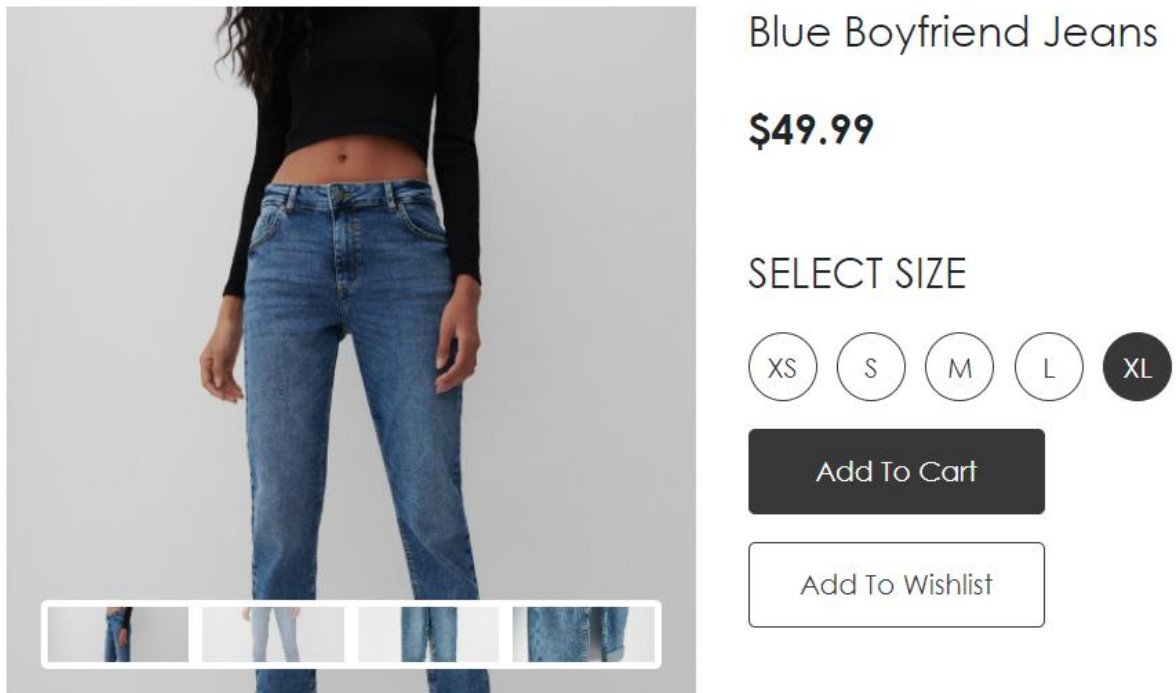


Image N° 14, part of the product page, besides there is navbar, footer, slider and description

```

/*img carousel*/
const productImages = document.querySelectorAll(".product-images img");
const productImageSlide=document.querySelector(".image-slider");
let activeImageSlide = 0;

productImages.forEach((item, i) => {
    item.addEventListener('click', () => {
        productImages[activeImageSlide].classList.remove('active');
        item.classList.add('active');
        productImageSlide.style.backgroundImage=`url('${item.src}')`;
        activeImageSlide=i;
    })
})

// size button
const sizeBtns=document.querySelectorAll('.size-radio-btn');
let checkedBtn =0;

sizeBtns.forEach((item, i) => {
    item.addEventListener('click',() =>{
        sizeBtns[checkedBtn].classList.remove('check');
        item.classList.add('check');
        checkedBtn=i;
    })
})

```

Image № 15, JS for image slideshow and size button selector

```

<a href="cartAction.php?action=addToCart&id=<?=$row["item_id"]; ?>" class="btn cart-btn">Add to Cart</a>

<form action="wishlist.php" method="post">
<input type="hidden" type="number" name="quantity" value="1" placeholder="Quantity" required>
<input type="hidden" name="product_id" value="<?=$row['item_id']?>">
<input type="submit" class="btn wish-btn" value="Add To Wishlist">
</form>

```

Image № 16, code for “add to cart” and “add to wishlist”

item_id	item_gender	item_type	item_name	item_color	item_price	item_actual_price	item_image	item_image2
10	boy	pants	Knaki pants with neon green stripe	khanki	14.99	50.59	./images/1/5.jpg	./images/2/5.jpg

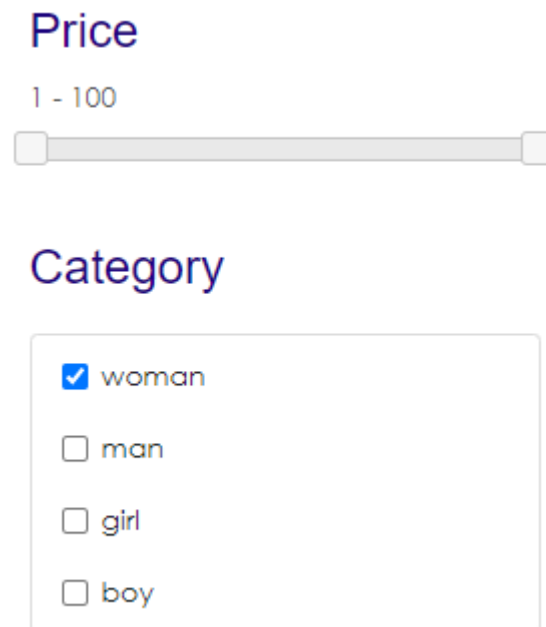
Image № 17, example of product in our database

d) Product catalog

To create this page we used two types of php files:

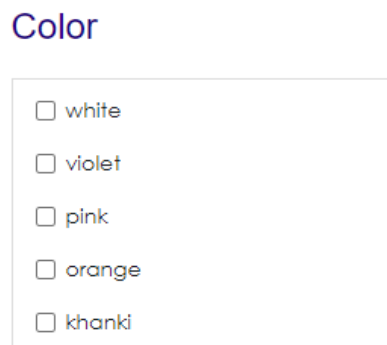
- fetch_data.php – filters products and displays all products that meet the requirements
- man.php, woman.php ... - used to display filters and contains most of the mechanism of filter

Additionally, the product catalog needs jquery and bootstrap.



The image shows two UI elements. The top one is a price filter section with the title "Price" in purple, a range "1 - 100" in orange, and a horizontal slider bar with a light gray track and white end caps. The bottom one is a category selection section with the title "Category" in purple. It contains a list of four items: "woman" (checked with a blue checkbox), "man" (unchecked), "girl" (unchecked), and "boy" (unchecked).

Image № 18, fetched gender category and price slider



The image shows a color selection section with the title "Color" in purple. It contains a list of five items, each with an unchecked checkbox: "white", "violet", "pink", "orange", and "khanki".

Image № 19, fetched color categories

```
<?php
$query = "SELECT DISTINCT(item_gender) FROM product WHERE item_status = '1' ORDER BY item_gender DESC";
$statement = $connect->prepare($query);
$statement->execute();
$result = $statement->fetchAll();
foreach($result as $row)
{
?>
<div class="list-group-item checkbox">
  <label><input type="checkbox" class="common_selector gender" value="<?php echo $row['item_gender']; ?>">
  <?php echo $row['item_gender']; ?></label>
</div>
<?php
}
?>
```

Image № 20, man.php, example of code for fetching all item_gender, setting type of selector (checkbox)

```

<script>
  $(function(){
    $('.gender').each(function(e){
      if($(this).val() == 'man'){
        $(this).attr("checked", "checked");
      }
    });
  });
</script>

```

Image Nº 21, man.php, script that allows us to prechecking a checkbox, that way we can create separate pages for men's clothes

```

$('#price_range').slider({
  range:true,
  min:1,
  max:100,
  values:[1, 100],
  step:1,
  stop:function(event, ui)
  {
    $('#price_show').html(ui.values[0] + ' - ' + ui.values[1]);
    $('#hidden_minimum_price').val(ui.values[0]);
    $('#hidden_maximum_price').val(ui.values[1]);
    filter_data();
  }
});

```

Image Nº 22, man.php, code responsible for price slide bar

e) Wishlist

Shopping wishlist





Product		add to cart now	Price
	Grey jeans Remove	Add To Cart	\$35.99
	Rib knit jersey crop top Remove	Add To Cart	\$12.99

Image Nº 23, wishlist, has 2 buttons: “add to cart”, “remove” and a link to product details, this wishlist was based on a cart tutorial (literature, 5th subsection) that we tailored for this exact purpose

f) Cart

Shopping Cart

Image	Product	Price	Quantity	Total	
	Grey jeans	\$35.99	<input type="text" value="1"/>	\$35.99	<button>Delete</button>
	Blue boyfriend jeans	\$49.99	<input type="text" value="1"/>	\$49.99	<button>Delete</button>
Cart Total			\$85.98		

Continue Shopping
Checkout

Image № 24, viewCart.php, cart, has three buttons: “continue shopping” redirects to unisex.php, “Delete” deletes product from the cart, “checkout” redirects to checkout, link to product details, quantity box

g) Checkout

- checkout.php – displaying checkout page
- orderSuccess.php – displaying a page with an order summary
- cartAction.php – backend file for viewCart.php, checkout.php, and orderSuccess.php, adding products to cart, filling and sending information from checkout page to database, error messages
- Cart.class.php – creating 'cart_contents' session, transfers the content from cart to checkout

Checkout

Contact Details

First Name

Last Name

Email

Phone

Address

Your Cart

3

Grey jeans \$35.99(2)	\$71.98
Blue boyfriend jeans \$49.99(1)	\$49.99
Total	\$121.97

Image № 25, checkout.php, checkout fill-out form, on the right side summary of our shopping cart, clicking “place order” redirects to orderSuccess.php

ORDER STATUS

Your order has been placed successfully.

Order Info

Reference ID: #21

Total: \$121.97

Placed On: 2022-01-14 22:39:32

Buyer Name: Justyna Neblik

Email: JustynaNeblik@gmail.com

Phone: 787878565

Product	Price	QTY	Sub Total
Grey jeans	\$35.99	2	\$71.98
Blue boyfriend jeans	\$49.99	1	\$49.99

Image № 26, orderSuccess.php, a summary of your order

id	first_name	last_name	email	phone	address	created	modified	status
17	Andrew	Hopkins	world55@gmail.com	456789456789	greenville 7, Glasgow	2022-01-06 20:22:44	2022-01-06 20:22:44	1
21	Justyna	Neblik	JustynaNeblik@gmail.com	787878565	Gliwce, Kujawska, 2	2022-01-14 22:39:32	2022-01-14 22:39:32	1

Image № 27, clients database, example order is marked

id	customer_id	grand_total	created	status
17	17	135.98	2022-01-06 20:22:44	Pending
21	21	121.97	2022-01-14 22:39:32	Pending

Image № 28 order database

id	order_id	item_id	quantity
21	17	3	1
22	17	16	1
23	17	1	1
31	21	9	2
32	21	3	1

Image № 29 order details

4.5. Problems during application development

Most of the errors resulted from typos, and it was possible to catch them relatively quickly. One of the problems we ran across was following a cart tutorial which we didn't understand. We decided to take a chance and try to rewrite it again, this time with a tutorial that was more like our way of thinking. It was a great idea because not only did we understand what was happening in our code, but it also started to work well. Another problem we encountered was sign-up login links, which didn't work if we were on one of these sites. The problem was the overlapping CSS style, we fixed the problem and it works fine now. A proven method of solving problems was to write the code step by step so that you know exactly where the source of our problem is.

5. Summary

The website meets the basic assumptions that we established at the beginning of creating the project. We used various platforms, tools, and programming languages to create the project.

To further develop the project, you could add online payments, email with an invoice, types of delivery, also in filters, you could specify the number of items that meet the condition, sorting products, displaying in 3 or 4 tiles per row. Another thing that can be expanded is the customer profile with orders, password change, and shipping status. Despite the fact that these are great ideas, we cannot implement them into the project because we would not have time to take these steps in this time frame. Although it differs from a professionally made website, we are satisfied with the final product we managed to create in a limited time during this semester.

6. Literature

1. <https://www.youtube.com/watch?v=0-NF3JMs4E8> (home page, part of product page)
2. Reserved.com (imgaes)
3. <https://www.youtube.com/watch?v=gCo6JqGMi30> (login and signup)
4. <https://github.com/iamshaunjp/php-mysql-tutorial/tree/lesson-31> (fetching product, product details page)
5. <https://codeshack.io/shopping-cart-system-php-mysql/> (wishlist)
6. <https://www.webslesson.info/2018/08/how-to-make-product-filter-in-php-using-ajax.html> (filtering system, product catalog)
7. <https://www.codexworld.com/simple-php-shopping-cart-using-sessions/> (cart, checkout)