

PLAY THERAPY USER MANUAL

Developed by Christian van Zyl

1.	INTRODUCTION	3
2.	BASICS: ACCESSING YOUR WEBSITE AND ADMIN.....	4-12
3.	PRODUCTS: ADDING, REMOVING AND UPDATING PRODUCTS.....	13-29
4.	ACCESSING USER PROFILE	30-36
5.	SHIPPING OPTIONS.....	37-39
6.	ORDERS	40-51
7.	UPDATING A PAGE ON YOUR SITE.....	52-58
8.	COLLECTING MONEY FROM THE STORE	59-60
9.	CHECKING WEB TRAFFIC & STATISTICS.....	61-63
10.	APPENDIX	64-70
11.	REFERENCING.....	71

1. INTRODUCTION

1.1. Background

Play Therapy by Lenka De Villiers-Van Zyl is an ecommerce website commissioned for development by Lenka De Villiers-Van Zyl. The website had three objectives:

- Serving as an additional source of income for the client.
- Promoting Play Therapy and its importance as well as the client's practice.
- Promoting the client's written work on Play Therapy.

The site succeeded in meeting these objectives. The following documentation serves as an informative guide on how to use the website as the administrator.

1.2 About this website

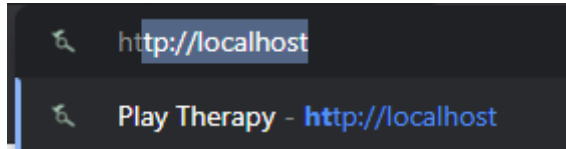
Play Therapy by Lenka De Villiers-Van Zyl has been coded in several languages. HTML5 (HyperText Markup Language) was used to a large extent, in conjunction with Bootstrap 4.4.1 and CSS (Cascading Style Sheets), for the front-end of the website. Bootstrap is an open-source CSS and JavaScript-based framework (Ouellette, 2021). Furthermore, icons from the website Font Awesome were used and font styles from Google Fonts were used.

JavaScript and JQuery 3.6.0 were used to a lesser extent and mainly used for validation and redirecting purposes. PHP 7.4.16 is used to form a connection to the database. It is further used to upload to, and download from, the database itself. The website was coded in Atom - an open-source text and source code editor (Kenlon, 2020).

2. BASICS: ACCESSING YOUR WEBSITE AND ADMIN

2.1 Accessing the website

In its current state, the website can be accessed by typing in “http://localhost” into the browser’s URL.



This will depend on the user’s Xampp setup and could also require an additional section after localhost i.e., “http://localhost/playtherapy”. When the website leaves the development environment and goes “live”, it will be accessible by the URL <https://www.playtherapybylenkadevilliersvanzyl.co.za>. The administrator section and user section are separated by using different usernames and passwords.

User:

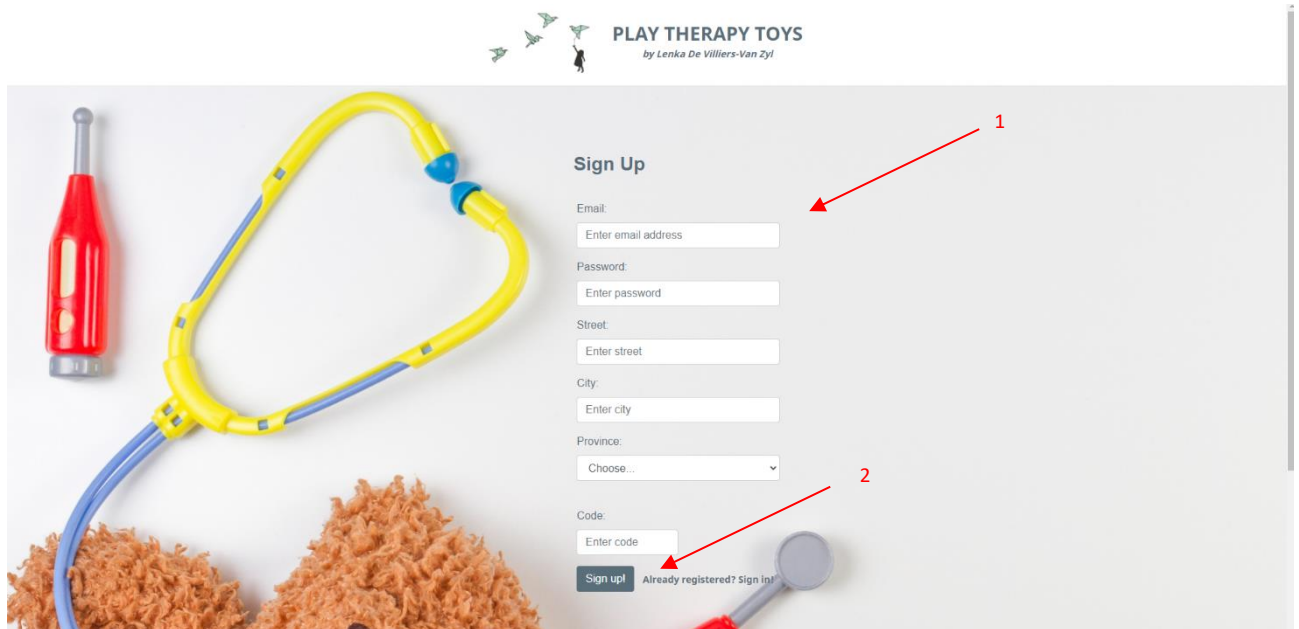
Users can currently access the “Home” page of the website by simply entering “http://localhost”. This will give the user “guest” privileges. The privileges allow for basic navigation and sending a message on the “Contact” page.

To access “customer” privileges, it is necessary for a user to login to the website. The user needs to navigate to the “Sign Up” page if they do not currently have an account, or to the “Sign In” page if they have an existing account.



The “Sign Up” page will require:

- 1) Basic information from the user to be entered into the given fields.
- 2) Clicking on the “Sign up” button. The user will then be redirected to the “Sign In” page.



The screenshot shows the 'Sign Up' page for 'PLAY THERAPY TOYS by Lenka De Villiers-Van Zyl'. The page features a background image of a yellow and blue stethoscope, a red toy stethoscope, and two piles of orange shredded material. The 'Sign Up' form includes the following fields and elements:

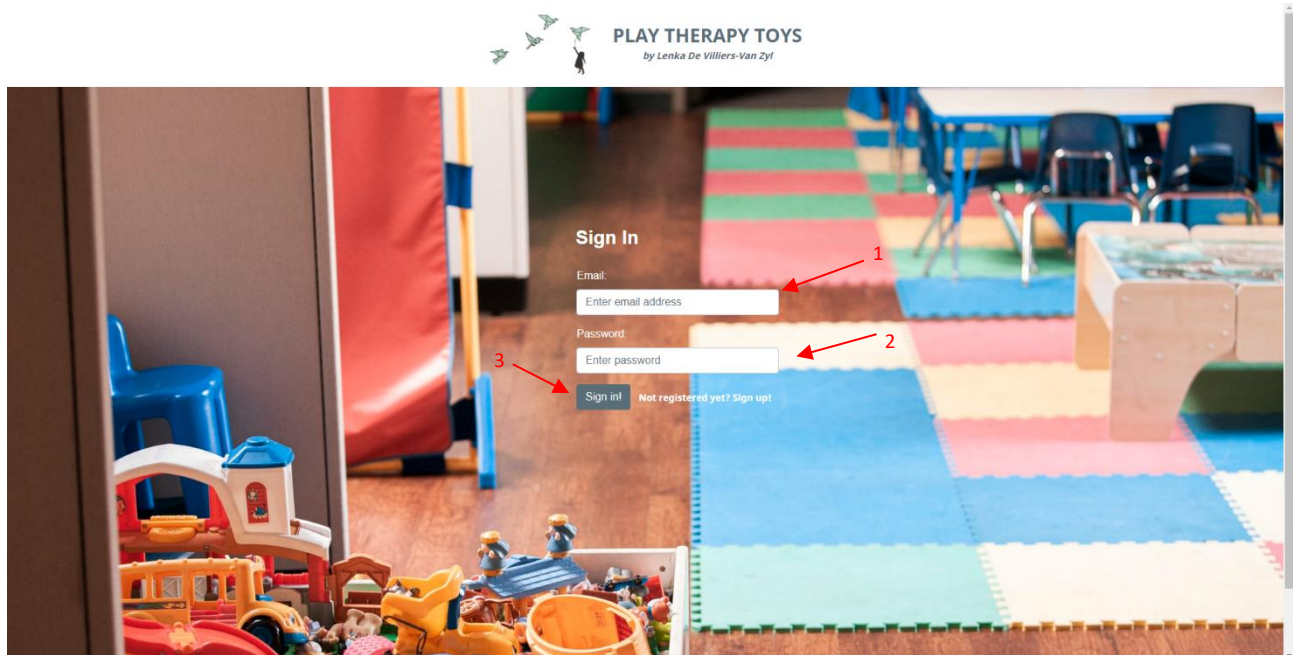
- Email:** A text input field with the placeholder 'Enter email address'. A red arrow labeled '1' points to this field.
- Password:** A text input field with the placeholder 'Enter password'.
- Street:** A text input field with the placeholder 'Enter street'.
- City:** A text input field with the placeholder 'Enter city'.
- Province:** A dropdown menu with the placeholder 'Choose...'.
- Code:** A text input field with the placeholder 'Enter code'.
- Sign up:** A dark blue button with white text.
- Already registered? Sign in:** A link in blue text.

A red arrow labeled '2' points to the 'Sign up' button.

The “Sign In” page requires the user to:

- 1) Enter their email address.
- 2) Enter their password.
- 3) Click on the “Sign in” button.

After submitting these details, clicking on the “Sign in” button will redirect them to the “Home” page, where additional privileges have now been granted.



This is immediately apparent by the change in the header section of the website, where the “Sign In” and “Sign Up” links have disappeared and been replaced by the “Profile” link. The “Cart” link also now displays a number next to it, indicating the number of items currently stored in the cart.



Administrator:

The administrator(s) can currently access the “Home” page of the website by simply entering “http://localhost”. To access administrative privileges, the administrator needs to click on the “Sign In” link provided in the header section



PLAY THERAPY TOYS
by Lenka De Villiers-Van Zyl

[Browse](#)

[Blog](#)

[Contact Us](#)

[About](#)

[Cart](#)

[Sign Up](#)

[Sign In](#)

After

being redirected to the “Sign In” page, the administrator will have to:

- 1) Enter their email address.
- 2) Enter their password.
- 3) Click on the “Sign in” button.



PLAY THERAPY TOYS
by Lenka De Villiers-Van Zyl

Sign In

Email:

Password:

[Not registered yet? Sign up!](#)

2.2. The Admin Area

The “Administrator panel” of the website is displayed below:



Edit Accounts



Edit Products



View Orders

The panel allows an administrator to access three main pages – “Edit Accounts”, “Edit Products” and “View Orders”. It is important to note that the “Sign Out” link is accessible even when navigating to these pages. Clicking on the “Sign Out” link will sign the administrator out of the session.

Edit Products

View Orders

Sign Out

To access these pages, the administrator can click on the links in the header section as displayed below:

Edit Accounts

Edit Products

View Orders

The pages can also be accessed by clicking on the image or heading associated with the page:



Edit Accounts




Edit Products



View Orders

Edit Accounts:

Navigating to the “Edit Accounts” page displays the following page:



PLAY THERAPY TOYS
by Lenka De Villiers-Van Zyl

[Edit Products](#)[View Orders](#)[Sign Out](#)

ACCOUNTS

< BACK

ID	Email	Address	Admin	Edit	Delete
37	lenkavanzyl@gmail.com	20 geelbos street cape town Mpumalanga 7530	✓		
41	admin@gmail.com	24 Rand Street Sandton Gauteng 9000	✓		
44	christianvanzyl@gmail.com	24 geelbos road cape town Gauteng 2342	✓		

The administrator can view all of the accounts registered to the website on this page. An account’s:

- 1) ID number
- 2) Email address
- 3) Address (Shipping Information)

ID	Email	Address
37	lenkavanzyl@gmail.com	20 Rand Street - Sandton - Gauteng - 9000

4) Privileges are displayed. A red checkmark indicates that the user does not have administrative privileges.

Admin
✓
✓
✓




The administrator has the options to:

5) Click on “Edit” to edit administator accounts.

For obvious ethical reasons, the customers’ accounts cannot be edited. However, the option to delete either customer accounts or administrator accounts exists.

6) Click on the “trash” icon to delete an account.

7) Click on the “Back” button to navigate to the previous page.

Edit	Delete
	 6
 5	

ACCOUNTS

 7

After clicking on the “Edit” button, one navigates to the page displayed below:



EDIT ADMIN ACCOUNT

[< BACK](#)

Email:	Password:
<input type="text" value="admin@gmail.com"/>	<input type="password"/>
Street:	City:
<input type="text" value="24 Geelbos road"/>	<input type="text" value="cape town"/>
Province:	Code:
<input type="text" value="Eastern Cape"/>	<input type="text" value="8000"/>
<input type="button" value="Change Details"/>	

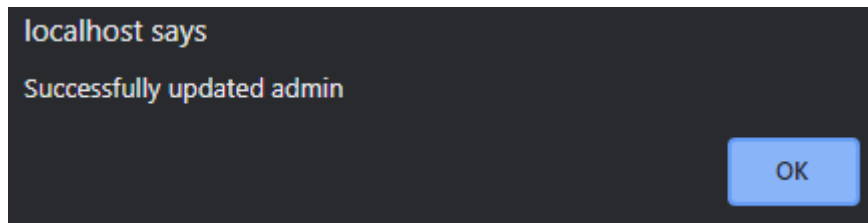
The

administrator can:

- 1) Change the details of the account by clicking on a field and re-entering the information.
- 2) Click on the “Change Details” button to confirm and apply the changes to the account.

Email:	Password:
<input type="text" value="admin@gmail.com"/>	<input type="password"/>
Street:	City:
<input type="text" value="24 Geelbos road"/>	<input type="text" value="cape town"/>
Province:	Code:
<input type="text" value="Eastern Cape"/>	<input type="text" value="8000"/>
<input type="button" value="Change Details"/>	

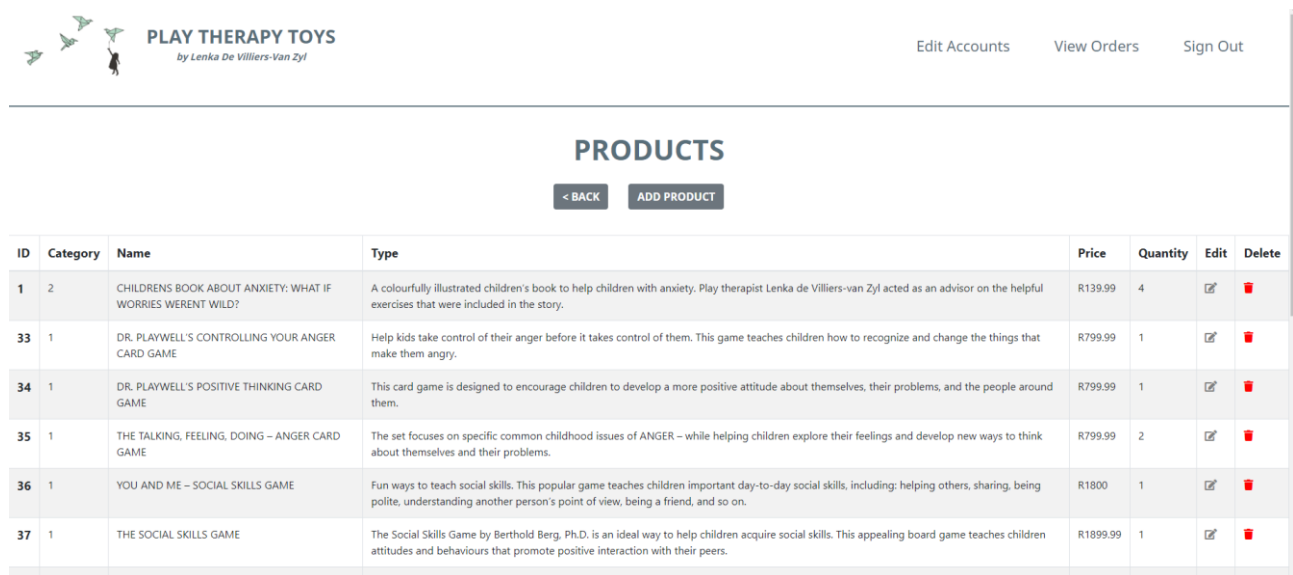
Changes are confirmed with a pop-up notification as shown below:



The other two pages available to the administrator, namely the “Edit Products” page and the “View Orders” page, will respectively explained in section 3 and section 6.

3. PRODUCTS: ADDING, REMOVING AND UPDATING PRODUCTS

Navigating to the “Edit Products” page displays the following page:



The screenshot shows the 'Edit Products' page for 'PLAY THERAPY TOYS by Lenka De Villiers-Van Zyl'. The page has a header with navigation links: 'Edit Accounts', 'View Orders', and 'Sign Out'. Below the header, the word 'PRODUCTS' is centered, with '< BACK' and 'ADD PRODUCT' buttons. A table lists products with columns: ID, Category, Name, Type, Price, Quantity, Edit, and Delete. The table contains 7 rows of product data.

ID	Category	Name	Type	Price	Quantity	Edit	Delete
1	2	CHILDRENS BOOK ABOUT ANXIETY: WHAT IF WORRIES WERENT WILD?	A colourfully illustrated children's book to help children with anxiety. Play therapist Lenka de Villiers-van Zyl acted as an advisor on the helpful exercises that were included in the story.	R139.99	4		
33	1	DR. PLAYWELL'S CONTROLLING YOUR ANGER CARD GAME	Help kids take control of their anger before it takes control of them. This game teaches children how to recognize and change the things that make them angry.	R799.99	1		
34	1	DR. PLAYWELL'S POSITIVE THINKING CARD GAME	This card game is designed to encourage children to develop a more positive attitude about themselves, their problems, and the people around them.	R799.99	1		
35	1	THE TALKING, FEELING, DOING – ANGER CARD GAME	The set focuses on specific common childhood issues of ANGER – while helping children explore their feelings and develop new ways to think about themselves and their problems.	R799.99	2		
36	1	YOU AND ME – SOCIAL SKILLS GAME	Fun ways to teach social skills. This popular game teaches children important day-to-day social skills, including: helping others, sharing, being polite, understanding another person's point of view, being a friend, and so on.	R1800	1		
37	1	THE SOCIAL SKILLS GAME	The Social Skills Game by Berthold Berg, Ph.D. is an ideal way to help children acquire social skills. This appealing board game teaches children attitudes and behaviours that promote positive interaction with their peers.	R1899.99	1		

The

administrator can view all the products on the database and the information associated with the products. The following information with regards to a product is displayed:

- 1) ID number of the product.
- 2) Category associated with the product – 1 indicates that the product is a toy, whilst 2 indicates that it is a book.
- 3) The product's name.
- 4) A description of the product.
- 5) The product's price.
- 6) The quantity of the product in stock

ID	Category	Name	Type
1	2	CHILDRENS BOOK ABOUT ANXIETY: WHAT IF WORRIES WERENT WILD?	A colourfully illustrated children's book to help children with anxiety. Play therapist Lenka de Villiers-van Zyl acted as an advisor on the helpful exercises that were included in the story.

Price	Quantity
R139.99	4

The administrator can add products, remove products and edit products on the database.

3.1 Adding and Removing Products

By clicking on the “ADD PRODUCT” button, the administrator is navigated to the “Add Product” page.

PRODUCTS

< BACK

ADD PRODUCT

The “Add Product” page is displayed below:



PLAY THERAPY TOYS
by Lenka De Villiers-Van Zyl

[Edit Accounts](#)

[Edit Orders](#)

[Sign Out](#)

ADD PRODUCTS

< BACK

Category:	Product Name:
<input type="text" value="Enter cate"/>	<input type="text" value="Enter product name"/>
Type:	Price:
<input type="text" value="Enter product type"/>	<input type="text" value="Enter price"/>
Quantity:	Image upload:
<input type="text" value="Select"/>	<input type="button" value="Choose File"/> No file chosen
<input type="button" value="Upload product"/>	

The administrator can enter the product's details into the various fields provided:

- 1) Adding the category number, where “1” indicates that the product is a toy and “2” indicates the the product is a book.
- 2) Enter the product's name.
- 3) Enter the product's description.
- 4) Enter the product's price.
- 5) Select the quantity of the product to be added to the database – thus the available stock of the product.
- 6) Upload an image to the database. The administrator needs to click on the “Choose File” button and add the chosen file.
- 7) Upload the product by clicking on the “Upload product” button.

The form contains the following fields and labels:

- Category:** (labeled 1) with a text input field containing "Enter cate".
- Product Name:** (labeled 2) with a text input field containing "Enter product name".
- Type:** (labeled 3) with a text input field containing "Enter product type".
- Price:** (labeled 4) with a text input field containing "Enter price".
- Quantity:** (labeled 5) with a dropdown menu showing "Select".
- Image upload:** (labeled 6) with a "Choose File" button and the text "No file chosen".
- Upload product** (labeled 7) is a dark blue button at the bottom.

A brief explanation on how the code facilitates this functionality will now be provided.

The “Add Product” page was created with HTML5, CSS, JavaScript and Bootstrap. The main part of the page is the form provided. The form's code follows below in order to add context to the explanation:

```

<form id="addproductform" method="post" action="backend/addProducts.php" onsubmit="return
    productSubmitHandler()">

    <div class="form-row">

        <div class="form-group col-6-md">

            <label class="label" for="inputCategoryID">Category:</label>

            <input name="categoryID" type="text" style="width: 100px" class="form-control"
            id="inputCategory" placeholder="Enter category"/>

            <div class="invalid-feedback">Please enter a valid category.</div>

        </div>

        <div class="form-group col-6-md">

            <label class="label" for="inputProdName">Product Name:</label>

            <input name="productName" type="text" style="width: 420px" class="form-control"
            id="inputProdName" placeholder="Enter product name"/>

            <div class="invalid-feedback">Please enter a product name.</div>

        </div>

    </div>

    <div class="form-row">

        <div class="form-group col-6-md">

            <label class="label" for="inputProdType">Type:</label>

            <input name="prodType" type="text" style="width: 400px" class="form-control"
            id="inputProdType" placeholder="Enter product type"/>

            <div class="invalid-feedback">Please enter product type.</div>

        </div>

        <div class="form-group col-6-md">

            <label class="label" for="inputPrice">Price:</label>

            <input name="prodprice" type="text" style="width: 120px" class="form-control" id="inputPrice"
            placeholder="Enter price" />

            <div class="invalid-feedback">Please enter a valid price.</div>

        </div>

    </div>

    <div class="form-row">

        <div class="form-group col-6-md">

```



```

<label class="label" for="inputQuantity">Quantity:</label>

<input name="prodQuantity" type="number" style="width: 120px" class="form-control"
id="inputQuantity" placeholder="Select"/>

<div class="invalid-feedback">Please select a valid quantity.</div>

</div>

<div class="form-group col-6-md">

<label class="label" for="inputQuantity">Image upload:</label>

<input name="prodImg" type="file" style="width: 300px" class="form-control" id="inputImg" />

<div class="invalid-feedback">Please select a valid file.</div>

</div>

</div>


<button name="submit" value="submit" type="submit" class="btn btn-primary "
id="signUpButton">Upload product</button>

</form>

```



By clicking on the “Upload product” button, the “onsubmit” element of the form must firstly receive a return value of “true” from the “productSubmitHandler()”. The “productSubmitHandler()” consists of basic JavaScript validation. It checks that the fields of the form have been correctly filled.

If a field has not been filled with the correct data, a feedback message appears:

Category: 

Please enter a valid category.

If a field has been filled with the correct data, it is outlined in green and receives a green checkmark.

Category:  Product Name: 

Please enter a product name.

If a field contains the incorrect data, the “onsubmit” element of the form triggers the “event.preventDefault” JavaScript function. This prevents the form from following its normal action of posting the form. The “productSubmitHandler()” will only return a value of “true” if all the fields have been filled in correctly. The default action, posting the form, is then performed and the “action” of posting to "backend/addProducts.php" triggers. The backend “Add Products” page performs numerous activities. It firstly links the “connDB.php” file with the “addProducts.php” file in order to create the “conn” object that can be used to establish a connection to the database.

The “connDB.php” file’s code is shown below:

```
<?php

$hostName = "localhost";
$databaseName = "playtherapy";
$username= "christian";
$password= "";
?>
```

The code above is then incorporated into the “addProduct.php” page in order to create the “conn” object for establishing a connection with the database.

```
<?php

// require once so that file is read once and not multiple times, prevent wasteful multiple disk access
require_once ("connDB.php");

error_reporting(E_ALL);
```

```

ini_set("display_errors", "1");

// create new object conn. Calls new instance of mysqli function, using values from loginDB

$conn = mysqli_connect($hostName, $username, $password, $databaseName);

// connect_error is a property of the conn object, if it has a value, the die function is called to terminate the
    program

if($conn->connect_error) {

    // the error is not written to console after dev is completed, because can give hackers information in
        certain circumstances

    die("Fatal Error");

}else {

    echo "connection success";

};

```

The data posted from the front-end's "addProduct.php" page is received on the backend's "addProduct.php" page and stored in variables:

```

if(isset($_POST["submit"])){

$category = $_POST["categoryID"];

$prodName = $_POST["productName"];

$prodType = $_POST["prodType"];

$prodPrice = $_POST["prodprice"];

$prodQuantity = $_POST["prodQuantity"];

```

The image file submitted needs to be processed and stored into a variable to upload:

```

if(!empty($_FILES["prodImg"]["name"])) {

    // Get file info

    $fileName = basename($_FILES["prodImg"]["name"]);

    $fileType = pathinfo($fileName, PATHINFO_EXTENSION);

```

```

// Allow certain file formats

$allowTypes = array('jpg','png','jpeg','gif');

if(in_array($fileType, $allowTypes)){

    $image = $_FILES['prodImg']['tmp_name'];

    $imgContent = addslashes(file_get_contents($image));

}

```

An SQL query is stored in a variable called “query”. The data received from the form, stored in variables, are used in the query:

```

$query = "INSERT INTO product (categoryID, productName, prodType, prodprice, prodQuantity,
    prodImg) VALUES ('$category', '$prodName', '$prodType', '$prodPrice', '$prodQuantity',
    '$imgContent')";

```

A PHP function, “mysqli_query”, is used with the created object “conn” and stored variable “query”. The function establishes a connection and then performs the stored SQL query, inserting the data received from the form into the “product” table of the database:

```

if(mysqli_query($conn, $query)){

    // closing the connection

    mysqli_close($conn);

    echo "<script type='text/javascript'>alert('Successfully added product to the database!');

    location='../editProducts.php';



    </script>";

    exit;
}

```

}

Removing a product from the database is as simple as clicking on the red “trash” icon located underneath the column labelled “delete”. Clicking on this button will delete the product in the same row as the icon.

ID	Category	Name	Type	Price	Quantity	Edit	Delete
1	2	CHILDRENS BOOK ABOUT ANXIETY: WHAT IF WORRIES WERENT WILD?	A colourfully illustrated children’s book to help children with anxiety. Play therapist Lenka de Villiers-van Zyl acted as an advisor on the helpful exercises that were included in the story.	R139.99	4		

In terms of the code, the product’s ID is used to distinguish it from the other products on the database. The ID is then used in a simple MySQL query, executed by calling the function “mysqli_query” and passing the object “conn” and the variable “query” as parameters. The code displayed for the deletion process is shown below:

```
$id = $_GET["id"];

// sql to delete a record



$query = "DELETE FROM product WHERE productID = $id";

if (mysqli_query($conn, $query)) {
    mysqli_close($conn);
    header("Location: ../editProducts.php");
    exit;
} else {
    echo "Error deleting record";
    mysqli_close($conn);
    // terminate script
    exit;
}
```


3.2 Updating Products

[Explain how products are updated on your website]

By clicking on the “Edit products” button, the administrator is navigated to the “Edit Admin Product” page.

PRODUCTS				
< BACK		ADD PRODUCT		
	Price	Quantity	Edit	Delete
ly illustrated children’s book to help children with anxiety. Play enka de Villiers-van Zyl acted as an advisor on the helpful exercises cluded in the story.	R139.99	4		

The “Edit Product” page is displayed below:

**PLAY THERAPY TOYS**
by Lenka De Villiers-Van Zyl

Edit AccountsEdit OrdersSign Out

EDIT PRODUCTS

< BACK

Category:

2

Product Name:

CHILDRENS BOOK ABOUT ANXIETY: WHAT IF WORF

Type:

A colourfully illustrated children's book to help childre

Price:

139.99

Quantity:

4

Image upload:

Choose FileNo file chosen

Change Details

The administrator can now edit the various product details. The fields are populated with the current data related to the product. The administrator can edit:

- 1) the category number, where “1” indicates that the product is a toy and “2” indicates the the product is a book.
- 2) change the product’s name.

22 | Page

- 3) change the product's description.
- 4) change the product's price.
- 5) change the stock allocated to the product.
- 6) Upload a different image to the database. The administrator needs to click on the "Choose File" button and add the chosen file.
- 7) Upload the changed product details by clicking on the "Change Details" button.

The screenshot shows a web form for editing product details. It includes the following elements:

- Category:** A dropdown menu with the value '2' selected. (Annotation 1 points to the dropdown arrow)
- Product Name:** A text input field containing 'CHILDRENS BOOK ABOUT ANXIETY: WHAT IF WORF'. (Annotation 2 points to the text)
- Type:** A text input field containing 'A colourfully illustrated children's book to help childre'. (Annotation 3 points to the text)
- Price:** A text input field containing '139.99'. (Annotation 4 points to the text)
- Quantity:** A numeric input field with a spinner, showing the value '4'. (Annotation 5 points to the spinner)
- Image upload:** A section containing a 'Choose File' button and the text 'No file chosen'. (Annotation 6 points to the 'Choose File' button)
- Change Details:** A dark blue button at the bottom of the form. (Annotation 7 points to the button)

A brief explanation on how the code facilitates this functionality will now be provided.

The "Edit Admin Product" page was created with HTML5, CSS, JavaScript and Bootstrap. The main part of the page is the form provided. The form's code follows below in order to add context to the explanation:

```
<form id="editform" method="post" action="backend/updateProduct.php?id= ' . $row['productID'] . '"
enctype="multipart/form-data" onsubmit="return productSubmitHandler()">
```

```
<div class="form-row">
```

```
<div class="form-group col-6-md">
```

```
<label class="label" for="inputCategoryID">Category:</label>
```

```

        <input name="categoryID" type="text" style="width: 100px" class="form-control"
        id="inputCategory" value= "" . $row['categoryID'] . ""/>

        <div class="invalid-feedback">Please enter a valid category.</div>

    </div>

    <div class="form-group col-6-md">

        <label class="label" for="inputProdName">Product Name:</label>

        <input name="productName" type="text" style="width: 420px" class="form-control"
        id="inputProdName" value="" . $row['productName'] . ""/>

        <div class="invalid-feedback">Please enter a product name.</div>

    </div>

</div>

<div class="form-row">

    <div class="form-group col-6-md">

        <label class="label" for="inputProdType">Type:</label>

        <input name="prodType" type="text" style="width: 400px" class="form-control"
        id="inputProdType" value="" . $row['prodType'] . ""/>

        <div class="invalid-feedback">Please enter product type.</div>

    </div>

    <div class="form-group col-6-md">

        <label class="label" for="inputPrice">Price:</label>

        <input name="prodprice" type="text" style="width: 120px" class="form-control" id="inputPrice"
        value="" . $row['prodprice'] . "" />

        <div class="invalid-feedback">Please enter a valid price.</div>

    </div>

</div>

<div class="form-row">

    <div class="form-group col-6-md">

        <label class="label" for="inputQuantity">Quantity:</label>

        <input name="prodQuantity" type="number" style="width: 80px" class="form-control"
        id="inputQuantity" value="" . $row['prodQuantity'] . ""/>

        <div class="invalid-feedback">Please select a valid quantity.</div>

    </div>

```



```

<div class="form-group col-6-md">

  <label class="label" for="inputQuantity">Image upload:</label>

  <input name="prodImg" type="file" style="width: 300px" class="form-control" id="inputImg" />

  <div class="invalid-feedback">Please select a valid file.</div>

</div>

</div>


<button name="submit" value="submit" type="submit" class="btn btn-primary "
id="signUpButton">Change Details</button>

</form>



```

By clicking on the “Change Details” button, the “onsubmit” element of the form must firstly receive a return value of “true” from the “productSubmitHandler()”. The “productSubmitHandler()” consists of basic JavaScript validation. It checks that the fields of the form have been correctly filled.

If a field has not been filled with the correct data, a feedback message appears:

Category: 
Please enter a valid category.

If a field has been filled with the correct data, it is outlined in green and receives a green checkmark.

Category:  Product Name: 
Please enter a product name.

If a field contains the incorrect data, the “onsubmit” element of the form triggers the

“event.preventDefault” JavaScript function. This prevents the form from following its

normal action of posting the form. The “productSubmitHandler()” will only return a value of “true” if all the fields have been filled in correctly. The default action, posting the form, is then performed and the “action” of posting to "backend/updateProduct.php" triggers.

The backend “Update Product” page performs numerous activities. It firstly links the “connDB.php” file with the “updateProduct.php” file in order to create the “conn” object that can be used to establish a connection to the database.

The “connDB.php” file’s code is shown below:

```
<?php
$hostName = "localhost";
$databaseName = "playtherapy";
$username= "christian";
$password= "";
?>
```

The code above is then incorporated into the “updateProduct.php” page in order to create the “conn” object for establishing a connection with the database.

```
<?php
// require once so that file is read once and not multiple times, prevent wasteful multiple disk access
require_once ("connDB.php");
error_reporting(E_ALL);
ini_set("display_errors", "1");
// create new object conn. Calls new instance of mysqli function, using values from loginDB
```

```

$conn = mysqli_connect($hostName, $username, $password, $databaseName);

// connect_error is a property of the conn object, if it has a value, the die function is called to terminate the
// program

if($conn->connect_error) {

    // the error is not written to console after dev is completed, because can give hackers information in
    // certain circumstances

    die("Fatal Error");

}else {

    echo "connection success";

};

```

The data posted from the front-end's "editAdminProducts.php" page is received on the backend's "updateProduct.php" page and stored in variables:

```

if(isset( $_GET["id"])) {

    $id = $_GET["id"];

}

if(isset($_POST["submit"])){

$category = $_POST["categoryID"];

$prodName = $_POST["productName"];

$prodType = $_POST["prodType"];

$prodPrice = $_POST["prodprice"];

$prodQuantity = $_POST["prodQuantity"];

```

The “id” of the product being edited is allocated to the “id” variable. The “id”’s value is fetched from the URL, as it is embedded in the form’s action link, namely this code from the form code:

```
action="backend/updateProduct.php?id= ' . $row['productID']. "
```

The image file adjusted needs to be processed and stored into a variable to upload:

```
if(!empty($_FILES["prodImg"]["name"])) {  
    // Get file info  
  
    $fileName = basename($_FILES["prodImg"]["name"]);  
    $fileType = pathinfo($fileName, PATHINFO_EXTENSION);  
  
    // Allow certain file formats  
  
    $allowTypes = array('jpg','png','jpeg','gif');  
    if(in_array($fileType, $allowTypes)){  
        $image = $_FILES['prodImg']['tmp_name'];  
        $imgContent = addslashes(file_get_contents($image));  
    }  
}
```

An SQL query is stored in a variable called “updateReq”. The data received from the form, stored in variables, are used in the query:

```
$updateReq = "UPDATE product SET categoryID = '$category', productName = '$prodName', prodType  
= '$prodType', prodprice = '$prodPrice', prodQuantity = '$prodQuantity', prodImg = '$imgContent'  
WHERE productID = '$id'";
```

A PHP function, “mysqli_query”, is used with the created object “conn” and stored

variable “updateReq”. The function establishes a connection and then performs the stored SQL query, inserting the data received from the form into the “product” table, where the product’s ID matches that of the set “id” variable, of the database:

```
if(mysqli_query($conn, $updateReq )){  
    // closing the connection  
    mysqli_close($conn);  
    echo "<script type='text/javascript'>alert('Successfully updated product!');  
    location='../editProducts.php';  
    </script>";  
    exit;  
} else{  
  
    echo("Error description: " . mysqli_error($conn));  
    mysqli_close($conn);  
    exit;  
}};
```

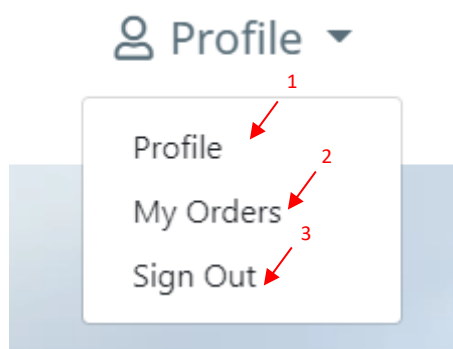
4. ACCESSING USER PROFILE

Customers, who have created an account and have signed in, can access their profile by clicking on the “Profile” button included in the header:



The “Profile” button has a drop-down menu when clicked. The user has three options and can:

- 1) view/edit their profile that includes their password and shipping details.
- 2) view the orders they have made and whether or not it has been marked as delivered.
- 3) sign out of the website.



Clicking on the “Profile” button in the drop-down menu redirects the user to the “Profile” page displayed below:

On this page the user can change their password by clicking on the “Edit password”. Please note that the “email” field, the password field as well as the “Change password” button is “greyed” out. This is to ensure that the user cannot change their email address or accidentally change their password. By clicking on the “Edit password” button, the password field as well as the “Change password” button become available.

Account Details:

Email:

lenkavanzyl@gmail.com

Password:

Edit password

Change password

JavaScript is used to disable and enable the greyed out fields. The comments above the code, displayed below, explains the logic and what the code does:

```
// the password inputfield is deactivated until the user clicks on
// checkIfChangePassword button. When they click on this button,
// the inputPassword field is changed to active and the changePasswordButton
// is also changed to an active state
const checkifchangepassword = document.getElementById("checkIfChangePassword");
if (checkifchangepassword) {
    checkifchangepassword.addEventListener("click", enablePasswordButton, false);
```

```
}  
  
function enablePasswordButton() {  
    document.getElementById("changePasswordButton").disabled = false;  
    document.getElementById("inputPassword").disabled = false;  
}
```

The “Profile” page also allows users to change their shipping details. The user can:

- 1) change their street’s name.
- 2) change their city’s name.
- 3) change the province by clicking on the drop down menu.
- 4) change their postal code.
- 5) click on the “Change shipping details” button to finalize the changes.

Shipping Address:

The form is titled "Shipping Address:" and contains four input fields and one button. Red arrows with numbers 1 through 5 point to specific elements: 1 points to the "Street" input field containing "10 Geelbos Street"; 2 points to the "City" input field containing "Cape Town"; 3 points to the "Province" dropdown menu showing "Gauteng" with a downward arrow; 4 points to the "Code" input field containing "7530"; and 5 points to the "Change shipping details" button.

Street:	<input type="text" value="10 Geelbos Street"/>	City:	<input type="text" value="Cape Town"/>
Province:	<input type="text" value="Gauteng"/>	Code:	<input type="text" value="7530"/>
<input type="button" value="Change shipping details"/>			

The password details as well as the shipping details are changed in the same fashion. A form is used on the front-end, which on submit first performs a validation check as shown below:

Password:

Please enter a password with at least 8 characters, 1 digit and 1 uppercase, lowercase & special character.

Shipping Address:

Street:

Please enter a street name.

City:

The forms post to backend PHP files that performs the same coding logic. For brevity's sake, only the password changes will be shown.

The code below displays PHP code inserted inbetween the HTML code on the front-end page "Profile". PHP has to be included in this section, as the data to fill the forms need to be fetched from the database and according to the set account ID. The account ID is set when the user signs in.

The user's account ID is then stored in the variable "\$_SESSION['uid']". This user ID is fetched in the code below and stored in the "id" variable. A query is created and stored in the variable "accountInfo". The query will select all the details of the particular account, using the "id" to identify the particular account. The user's email address and associated details can then be fetched through an SQL query using the "conn" object, as well as the stored query "accountInfo".

Once the form is submitted, validation checks are performed through the "accountSubmitHandler" JavaScript function. If the checks return a value of "true" the form's action, of posting the data to the "backend/changeAccountDetails.php", will trigger.

```
<?php
```

```

// require once so that file is read once and not multiple times, prevent wasteful multiple disk access
require_once ("backend/connDB.php");

// create new object conn. Calls new instance of mysqli function, using values from loginDB

$conn = mysqli_connect($hostName, $username, $password, $databaseName);

```

```

// connect_error is a property of the conn object, if it has a value, the die function is called to terminate
the program

if($conn->connect_error) {

    // the error is not written to console after dev is completed, because can give hackers information in
    certain circumstances

    die("Fatal Error");

};

if(isset($_SESSION['uid'])) {

    $id = $_SESSION['uid'];

}

else {

    echo "Index not present";

}

$accountInfo = "SELECT * FROM account WHERE accountID = $id";

$result = $conn->query($accountInfo);

if ($result) {

    while($row = $result->fetch_assoc()) {

        echo

        '<form id="profileAccountform" method="post" action="backend/changeAccountDetails.php"
onsubmit="return accountSubmitHandler()">

        <div class="form-row">

            <div class="form-group col-6-md">

                <label class="label" for="inputEmail">Email:</label>

                <input disabled="disabled" name="email" type="email" class="form-control" id="inputEmail"
value= " . $row['accountEmail'] . "'/>

            </div>

            <div class="form-group col-6-md">

                <label class="label" for="inputPassword">Password:</label>

                <input disabled="disabled" name="password" type="password" class="form-control"
id="inputPassword"/>

                <div class="invalid-feedback">Please enter a password with at least 8 characters, <br />1 digit
and 1 uppercase, lowercase & special character.</div>

            </div>

```

```

        </div>

        <button name="button" type="button" class="btn btn-primary" id="checkIfChangePassword">Edit
password</button>

        <button disabled="disabled" name="submit" value="submit" type="submit" class="btn btn-
primary " id="changePasswordButton">Change password</button>

        </form>

        ';

    }; };

?>

```

On the backend's side, the "changeAccountDetails.php" page performs numerous functions. The set session ID is firstly fetched. The session information is obtained on all PHP pages by including the function "session_start();" at the beginning of the PHP file. The data can then be accessed by making use of the "\$_SESSION variable".

The rest of the code is also straight-forward. The posted data is stored in a variable called "password". The data is sanitized by using the PHP function "mysqli_real_escape_string). The newly entered password will also need to be hashed for security reasons by using the PHP function md5().

Hashing the password ensures that the entered password is not stored in the database, but rather its hash value. On signing in, this hash value is compared to the entered password (which is then also hashed). If the two hash values coincide, access will be granted. The password is updated through the creation of a query stored in "updateReq" and then by invoking the PHP function "mysqli_query" and passing the values of "conn" and "updateReq" to it.

```

// account info

if(isset($_SESSION['uid'])) {

    $id = $_SESSION['uid'];

}

else {

    echo "Index not present";

}

```

```

if(isset($_POST["submit"])){

    $password = $_POST["password"];

    // Data sanitization

    // protection from mysql injection by using realscapestring

    $password = mysqli_real_escape_string($conn, $password );

    $hash = md5($password);

    $updateReq = "UPDATE account SET accountPasswordHash = '$hash' WHERE accountID = '$id'";

    if(mysqli_query($conn, $updateReq )){

        // closing the connection

        mysqli_close($conn);

        echo "<script type='text/javascript'>alert('Successfully updated password!');

        location='../profileScreen.php';

        </script>";

        exit;

    } else{

        echo("Error description: " . mysqli_error($conn));

        mysqli_close($conn);



        exit;

    }
}

```

5. SHIPPING OPTIONS

Play Therapy by Lenka De Villiers-Van Zyl makes use of *PostNet* to ship and deliver their products. As they offer affordable prices, a standard rate of R180.00 is added to all orders for shipping. Customers are informed of the additional cost at the “Shipping” page. Below the cart checkout page displays one item priced at R499.00.

Product Name		Category	Quantity	Price	Remove
BEAR STAGE PUPPET		Books	1	R499.99	


TOTAL: R499.99
[Place Order](#)

Clicking on the “Place Order” button navigates to the “Shipping” page, where the following is displayed before confirming shipping details:

A shipping cost of R180.00 is added to all orders.
Shipping is done by Postnet.

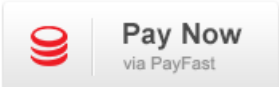
[Confirm shipping details](#)

The final screen before payment reflects the added cost:

BEAR STAGE PUPPET 

Total: R679.99

* Price



A small product costs approximately R99.99 to ship, whilst larger orders can reach R200.00. By submitting an average rate, “losses” incurred for larger products will be nullified by the average rate that is more expensive for smaller products.

The following code is used to set and insert the shipping price into a transaction:

```

$shippingPrice = 180.00;

$orderTotalPrice = $_SESSION['totalCart'] + $shippingPrice;

$orderPrice = $_SESSION['totalCart'];

// date function of php

$orderDate = date('Y-m-d');


// inserting the order details from above into the ordertransaction table.


$orderInsert = "INSERT INTO ordertransaction (accountID, shippingPrice, orderTotalPrice, orderPrice,
orderDate, orderDelivered) VALUES ('$id', '$shippingPrice', '$orderTotalPrice', '$orderPrice', '$orderDate',
'$orderDelivered')";

```

The shipping price is added in the “Order.php” page. A variable of “shippingPrice” is set to 180.00 and added to the total order’s price. The individual shipping price and total order’s price is then inserted into the database to log the transaction. The order process is discussed in depth in the following section.

Please note that the administrator should view the placed orders on a day-to-day basis.

By viewing the orders, the administrator can identify the account ID linked to the transaction.



PLAY THERAPY TOYS
by Lenka De Villiers-Van Zyl

[Edit Products](#)
[Edit Accounts](#)

ORDERS

< BACK

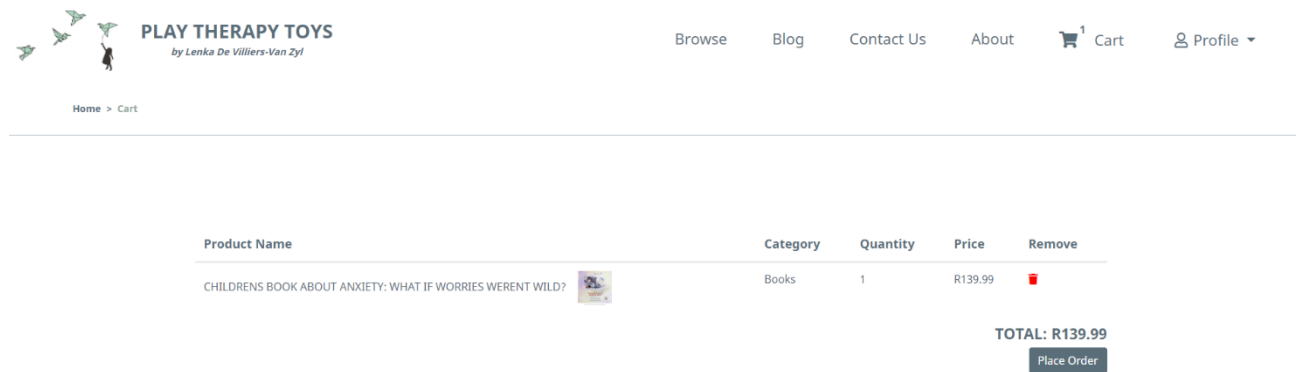
Order ID	AccountID	Order Price	Date	Delivered	Change Delivery Status
61	37	979.99	2021-06-15	✓	
62	37	319.99	2021-06-15	✓	

By viewing the accounts section of the administrator panel, as discussed in section 2, the shipping address of the customer can be noted for the purpose of posting the parcel via *PostNet*.

ACCOUNTS					
< BACK					
ID	Email	Address	Admin	Edit	Delete
37	lenkavanzyl@gmail.com	20 Rand Street - Sandton - Gauteng - 9000	✓		✖

6. ORDERS

The order process is complex and involves several PHP pages. After a user has selected the products that they want to buy, by adding it to their cart, they will navigate to the “Cart” page to check out. The “Cart” page is displayed below:



The “Cart” page is important to the placement of the order, as it already contains important information used during the order process. The code below displays a small section of the “Cart” page’s code, which is necessary to understand and discuss before continuing with the order process.

Navigating to the “Cart” page initializes the session variable “totalCart”. This variable is used to determine the overall price of the order. It is initially set to 0. The next part of the code uses the data stored in the session variable, which is an array, “cart”. This variable is initialized when a user signs in as an empty array.

The “cart” array contains all the product ID’s of products added to the cart. The array is looped through using a “for loop”. For each product ID present in the array, the product information pertaining to that product ID is now fetched. This is done through the use of a standard SQL query. The product information is then used to populate the “Cart” page’s table. Furthermore, the product price of each product in the array will be added to “totalCart” in order to determine the price of the order.


```

// checks if the session variable of cart has been initiated. Initiates and sets the totalcart variable to 0

if(isset($_SESSION['cart'])) {

    $_SESSION['totalCart'] = 0;


    // for loop, looping through the array of the session variable cart. The cart array contains the product ID's
    // of the products added to cart.


    foreach ($_SESSION['cart'] as $prodID){


        // during a loop, the productID from the array cart is used to find the matching product in the database
        // with the same product ID.

        $productInfo = "SELECT * FROM product WHERE productID = $prodID";

        // query to the database is made

        $result = $conn->query($productInfo);


        // all the row information pertaining to fetched product is now "fed" into the table

        while($row = $result->fetch_assoc()) {

            // checks the category of product, either a toy or a book

            if($row['categoryID'] === 1)

            {

                $insertType = "Toys";

            }else{

                $insertType = "Books";

            }


            if($row['prodQuantity'] === 0)

            {

                $insertStock = "Out of Stock";

```

```

}else {

$insertStock = 1;

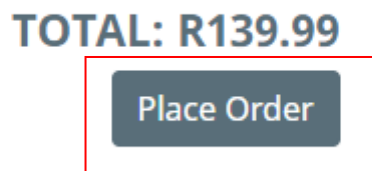
// totalcart variable is used to store all the product prices combined

$_SESSION['totalCart'] = $_SESSION['totalCart'] + $row['prodprice'];

}

```

The order process can now be further explained. An order is placed by clicking on the “Place Order” button:



This button makes use of a JavaScript event listener. The PHP code shown below is included in the “Cart” page. The JavaScript is shown below the PHP code and the comments explain the code in more depth. If the cart is not empty, the JavaScript ultimately redirects to the “backend/order.php” page.

```

<?php if(isset($_SESSION['totalCart']) && $_SESSION['totalCart'] > 0){

    // form redirects to order php script

    echo " <form method='post' action='backend/order.php?>

        <button name='submit' value='submit' type='submit' class='btn btn-primary'
id='cartButtonRedirect'>Place Order</button></form>";

    }else{

        // alert that shows that the cart is empty and needs to be filled before check out

        echo "<button class='btn btn-primary' id='alertEmpty'>Place Order</button>";

    }

?>

```

```

//      Cart  Screen functions      //

// this is an event listener for activating a redirect function to a php page
// adding a product to the cart

const redirect = document.getElementById("cartButtonRedirect");

if (redirect) {

    redirect.addEventListener("click", reDirectCart, false);

}

// eventlistener attached to button alerting the user that their cart is empty

const cartEmpty = document.getElementById("alertEmpty");

if (cartEmpty) {

    cartEmpty.addEventListener("click", cartEmptyMessage, false);

}

// function alerting user that their cart is empty

function cartEmptyMessage() {

    alert(

        "Your cart is empty. Please fill your cart with at least one item before checking out"

    );

}

```

The “backend/order.php” page inputs the order information into the order table of the database. An excerpt of the code is shown below and the comments explain the code in depth:

```

// account info, if logged in, is fetched.

if(isset($_SESSION['uid'])) {

    $id = $_SESSION['uid'];

}

else {

```

```

    echo "Index not present";
}

// payment info and product

// for loop, looping through the array of the session variable cart. The cart array contains the product ID's of
the products added to cart.

if(isset($_SESSION['cart'])) {

    $_SESSION['totalCart'] = 0;

    foreach ($_SESSION['cart'] as $prodID){

        // during a loop, the productID from the array cart is used to find the matching product in the database with
        the same product ID.

        $productInfo = "SELECT * FROM product WHERE productID = $prodID";

        // query to the database is made on above statement

        $result = $conn->query($productInfo);

        // fetching the product price of the product matching the id of the prodID in the array and adding it to
        totalCart,

        // the session variable for the total cost of the order

        while($row = $result->fetch_assoc()) {

            $_SESSION['totalCart'] = $_SESSION['totalCart'] + $row['prodprice'];

        }

    }}

// setting the order delivery status to false

$orderDelivered = 0;

// adding the shipping cost

$shippingPrice = 180.00;

$orderTotalPrice = $_SESSION['totalCart'] + $shippingPrice;

$orderPrice = $_SESSION['totalCart'];

// date function of php

$orderDate = date('Y-m-d');
```

```
// inserting the order details from above into the ordertransaction table.
```

```
$orderInsert = "INSERT INTO ordertransaction (accountID, shippingPrice, orderTotalPrice, orderPrice,
orderDate, orderDelivered) VALUES ('$id', '$shippingPrice', '$orderTotalPrice', '$orderPrice', '$orderDate',
'$orderDelivered')";
```

```
if(mysqli_query($conn, $orderInsert)){
```

```
    // closing the database connection
```

```
    mysqli_close($conn);
```

```
    //redirection back to the shipping screen
```

```
    header("Location: ../shippingScreen.php");
```

```
    //terminating the script
```


```
    exit;
```

```
} else{
```


```
    echo "Failure: $orderInsert. " . mysqli_error($conn);
```


```
}
```

The “backend/order.php” page ultimately redirects to the “Shipping” page. This page is displayed below:



[Browse](#)[Blog](#)[Contact Us](#)[About](#)

 ¹ Cart

 Profile ▾

Home > Cart > Shipping

Shipping Address:

Street:

20 Rand Street

City:

Sandton

Province:

Gauteng ▾

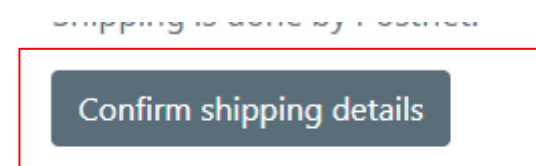
Code:

9000

A shipping cost of R180.00 is added to all orders.
Shipping is done by Postnet.

Confirm shipping details

The user can change their shipping details on this page. Clicking on the “Confirm shipping details” button redirects to the “backend/orderitem.php” page, which fulfills multiple roles.



The “orderitem.php” page firstly makes any changes to the shipping details by using the form’s posted data. The data is stored in variables and then a SQL query is made to update the account’s shipping details:

```
// account info

if(isset($_SESSION['uid'])) {

    $id = $_SESSION['uid'];

}

else {

    echo "Index not present";

}

if(isset($_POST["submit"])){

    $street = $_POST["street"];

    $city = $_POST["city"];

    $province = $_POST["province"];

    $code = $_POST["code"];

    // Data sanitization

    $province = mysqli_real_escape_string($conn, $province );

    $code = mysqli_real_escape_string($conn, $code);

    $updateReq = "UPDATE account SET accountStreet = '$street', accountCity = '$city', accountProvince = '$province', accountPostalCode = '$code' WHERE accountID = '$id'";

    if(mysqli_query($conn, $updateReq )){}
```

Following this code, is an important section of code. In order to solve the many to many relationship between the “Ordertransaction” table and the “Product” table, another table called “OrderItem” was created as a bridge between the two tables. The table fulfills a simple, but important role. It links the product ID to a specific order ID. This is crucial to maintain data integrity, as an order might contain multiple product ID’s and they all need to be linked to the same order ID, so that the following order’s product ID’s link correctly and data duplication does not take place.

To populate this table with the correct data, the order ID is firstly fetched from the order placed, according to the customer’s account ID. Following this, the shipping price variable is initiated with the set price of 180. The cart variable/array is then looped through in order to match all the product ID’s in the cart to the specific order ID. Having this information, it is now possible to populate the “OrderItem” table with a SQL insert query, namely “orderItemInsert”. After the data has been inserted into the table, the script redirects to the “Payment” page.

```
$orderItemIDInfo = "SELECT orderID FROM ordertransaction WHERE accountID = $id";  
$result = $conn->query($orderItemIDInfo);  
if(mysqli_num_rows($result) == 0) {  
    echo "<script type='text/javascript'>alert('Order ID not found.');}  
else{  
    $numResults = mysqli_num_rows($result);  
    $counter = 0;  
    while ($row = $result->fetch_assoc()) {  
        if (++$counter == $numResults) {  
            $orderID = $row['orderID'];  
        }  
    }  
}  
$shippingPrice = 180;
```

```

if(isset($_SESSION['cart'])) {
foreach ($_SESSION['cart'] as $prodID){

$orderItemProdIDInfo = "SELECT productID, prodprice FROM product WHERE productID = $prodID";

$result = $conn->query($orderItemProdIDInfo);

while($row = $result->fetch_assoc()) {

$totalLinePrice = $shippingPrice + $row['prodprice'];

$productID = $row['productID'];

$orderItemInsert = "INSERT INTO orderitem (orderID, productID, quantityOrdered, totalLinePrice)
VALUES ('$orderID', '$productID', '$quantityOrdered', '$totalLinePrice')";

if(mysqli_query($conn, $orderItemInsert)){

echo "Success";

} else{

echo "Failure";

mysqli_close($conn);

}}

}

}

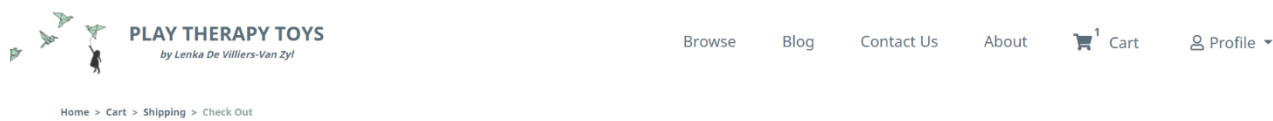
mysqli_close($conn);

header("Location: ../paymentScreen.php");

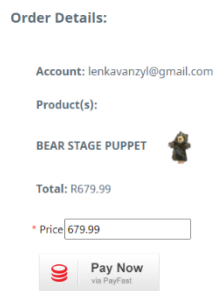
exit;

```

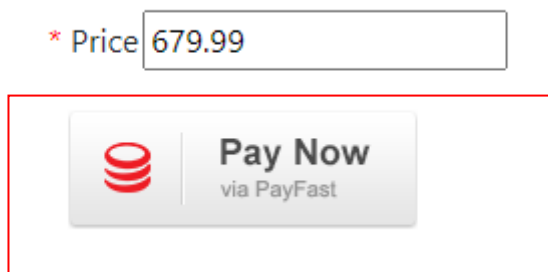

The last step of the order process takes place on the “Payment” page. The page is displayed below:




The customer is shown vital



information of the order and the final price (with the shipping price now included). By clicking on the “Pay Now” button, the customer is redirected to “Payfast”’s payment page.



The customer completes the payment by clicking on “Complete Payment” and is then redirected to “*Play Therapy by Lenka De Villiers-Van Zyl*”’s “Home” page.

 You're in the sandbox. This is a **testing environment** used to ensure correct integration with PayFast. If unexpected, please contact the merchant you are trying to pay to resolve the matter.

Play Therapy Products

Payment total	R 679.99 ZAR
---------------	--------------

You have a balance of R 99,590,876.57

COMPLETE PAYMENT

Payment total	R 679.99
From balance	-R 679.99
Remaining balance	R 99,590,196.58

CANCEL

Having trouble with your transaction? Let us help you
+27 (0)21 300 4455 support@payfast.co.za

The payment process and data supplied to *PayFast* is explained in depth in section 8 of this document. In terms of the order process, it is simply important to note that the “Payment” page includes PHP code that executes after being redirected. Firstly, the “cart” session variable is looped through in order to find the products sold according to their ID. The ID is then used in a query to fetch the current quantity of stock linked to each product. Following this, the quantity of stock of each product is adjusted by a SQL query in order to take the transaction into account. Finally, the “cartQuantity”, “totalCart” as well as the “cart” variable are reset for the next transaction.

```
if(isset($_SESSION['cart'])) {
```

```
foreach ($_SESSION['cart'] as $prodID){
```

```
$productInfo = "SELECT prodQuantity FROM product WHERE productID = $prodID";
```

```
$result = $conn->query($productInfo);
```

```
while($row = $result->fetch_assoc()) {
```

```
$productQ = $row['prodQuantity'] - 1;

$updateProductQuantity = "UPDATE product SET prodQuantity = '$productQ' WHERE productID =
$prodID";

if(mysqli_query($conn, $updateProductQuantity )){

}

}

}

}

$_SESSION['cartQuantity']= 0;
$_SESSION['totalCart'] = 0;
$_SESSION['cart']= [];

mysqli_close($conn);

exit;
```

7. UPDATING A PAGE ON YOUR SITE

In order to understand how pages on the website are updated, the concept of “session” and “session variables” need to be briefly explained. A “session” can be added to PHP code in order to store “session variables”. As soon as a session is created, a unique session ID is generated for the user (Guru99, 2021). Depending on server settings, the session lasts for a default amount set by the server (usually 24 minutes) or until the browser is closed (Guru99, 2021).

This feature is useful for storing information such as:

- 1) the current status of a created Boolean login variable, thus either “true” or “false”
- 2) the authority of the user currently using the website.
- 3) the number and/or content of items in a created cart variable.

When one navigates to a website programmed in PHP, it is likely that session variables are initialized or set at some point. In the case of the website in question, most of the session variables are set on signing in.

To access session variables on a PHP page, one needs to invoke the “session_start()” function. Session variables can then be instantiated or those previously set can be accessed.

The most important session variables for the website in question is displayed below. The variables are set on signing in on the “backend/signIn.php” page. The session function is firstly called in order to set and store session variables. The user ID is stored in the variable “id”. This is used to identify the current user’s account on all pages.

```
<?php
error_reporting(E_ALL);
ini_set("display_errors", "1");
session_start();
?>
```

```

if(isset($_SESSION['uid'])) {
    $id = $_SESSION['uid'];
}
else {
    echo "Index not present";
}

```

The variable “id” is set according to the user’s account ID. Other important variables set are listed below as well and the comments explain their purpose:

```

{
    // password and email matched, now get data of user to set session variables
    $row = mysqli_fetch_array($resultSet2);
    // set session uid to the user's account ID
    $_SESSION["uid"] = $row["accountID"];
    // set the auth level accordingly
    $_SESSION["authLVL"] = $row["accountAcclVL"];
    // set session login variable to true
    $_SESSION["login"] = true;
    // create variable to hold cart data
    $_SESSION["cart"] = [];
    // create variable to hold cart data quantity
    $_SESSION["cartQuantity"] = 0;
    // create variable holding cart total
    $_SESSION["totalCart"] = 0;
    // now check and set level of authentication.
    if ($_SESSION["authLVL"] == 0) {
        // login verified as user, direct to home screen where associated accountlevel is 0
        // set the session name to the user's email
    }
}

```

```

$_SESSION["name"] = "user";

echo "<script type='text/javascript'>

location='../index.php';

</script>";

// closing the connection

mysqli_close($conn);

// terminate script

exit;

}else{

    $_SESSION["name"] = "admin";

    // login verified as admin, direct to admin screen where associated accountlevel is 1

    // account level checked again at admin screen

    echo "<script type='text/javascript'>

location='../admin.php';

</script>";

    // closing the connection

    mysqli_close($conn);

    // terminate script

    exit;

};

};

```

The “cart” session variable is extremely important for the website to function effectively. It is set to an empty array on signing in (as displayed on the previous page). Every time a user clicks on the “addToCart” button, the following PHP code is executed:

```

if(isset($_SESSION["cart"])) {

    array_push($_SESSION['cart'], $_GET['id']);

    $_SESSION["cartQuantity"] ++;

```

```

}

header("Location: ../index.php");

exit;

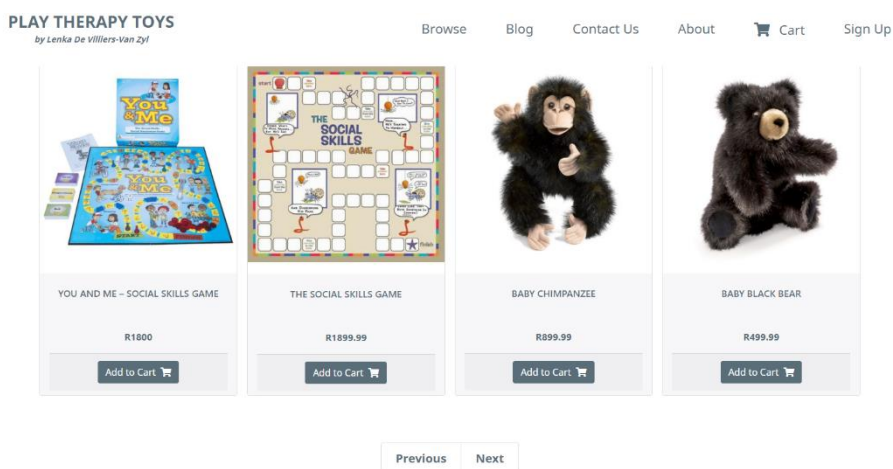
```

The code firstly fetches the session “cart” and then pushes the product ID (sent on clicking on the “addToCart” button) to the “cart” array. The session variable “cartQuantity” also increases. This variable is simply used to indicate the number of items currently in the cart as displayed below:

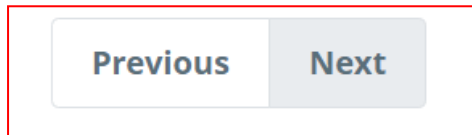


In terms of “adding new pages” to the website, unless one creates a website similar to WordPress (a daunting task on its own), it will only be possible to add pages to the website manually. The “Home” page contains links to all of the pages on the website (depending on the user level i.e. “guest”, “customer”, “administrator”). Expanding on the “Home” page or any of the other pages on the website would only be possible by adding new links to newly created pages.

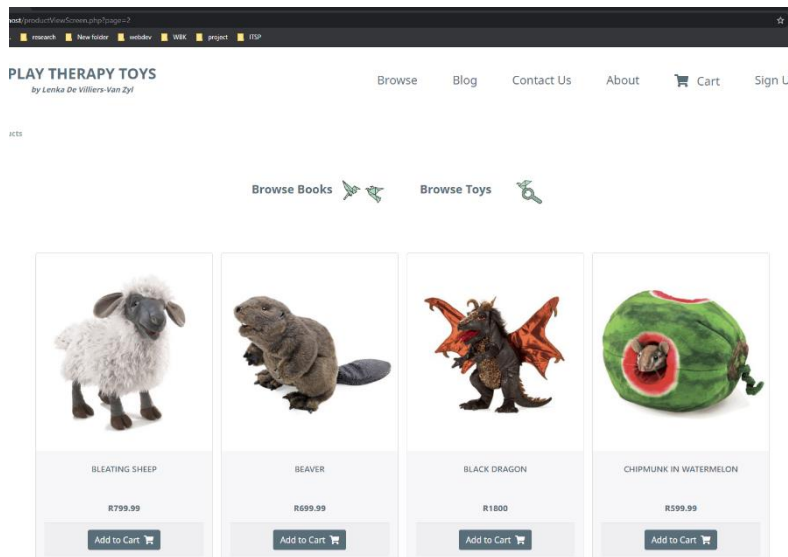
The only “new” pages created are related to the listing of products. The pagination system is used on the “Browse” page as well as the “Toys” and “Books” pages.



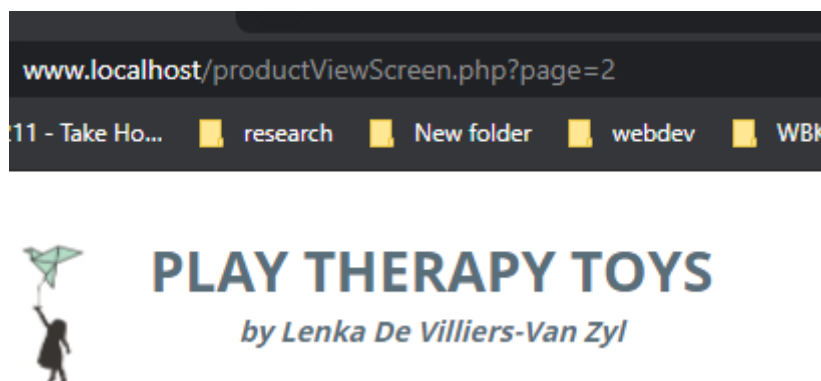
If the end of the page is reached on those pages, the “Next” or “Previous” buttons can be used to either show the products following those shown, or the previously listed products.



Clicking on either of the buttons, will result in a “new” page being loaded.



The URL indicates the current page number. The image below shows that the current page navigated to is the second page.



The code related to pagination is displayed below. It also makes use of session variables. The excerpt shown does not include the calling of the “session_start()” function, as it is invoked at the top of the page. The excerpt shown is used on all three pages where pagination is used, namely the “Browse” page, “Toys” page and “Books” page.

Firstly, a variable called “counter” is created and given the value of 0. A “page” variable is created after this. If it is already initialized, the value of the page is taken from the number next to the word “page” in the URL as shown in the image above. Otherwise, the page is automatically assigned a value of 1.

A variable called “limit” is created in order to specify the amount of results a query may return. This is used in conjunction with the “offset” variable, to keep track of what results have been displayed and should be displayed next.

The “amountOfProducts” variable uses a MySQL query to determine the number of products stored in the product table according to ID. The result of the query is then used to populate the variable “totalRows”.

With the total number of rows and a limit to the number of products displayed on each page set, the total number of pages can be calculated. This is stored in the “totalPagesNeeded” variable.

It is calculated by dividing the total number of rows by the limit set and then rounding the number up to the nearest integer. This is done by making use of the PHP function ceil.

The query to populate the page can now finally be created, namely the “productInfo” query. The result of the query is then used to populate the page with the specified products.

```
$counter = 0;

if (isset($_GET['page']) ) {

    $page = $_GET['page'];

} else {
```

```

$page = 1;

}

$limit = 8;

$offset = ($page - 1) * $limit;

$amountOfProducts = "SELECT COUNT(productID) FROM product";

$resultSet = mysqli_query($conn, $amountOfProducts);

$totalRows = mysqli_fetch_array($resultSet)[0];

$totalPagesNeeded = ceil($totalRows/$limit);

$productInfo = "SELECT * FROM product LIMIT $offset, $limit";

$result = mysqli_query($conn, $productInfo);

```

The navigation buttons at the bottom of the page are formed through the use of HTML and PHP. The Bootstrap class of “pagination” is used. One can simply populate a list with page numbers or, as in this case, with “previous” and “next” buttons. The buttons are links to the next and previous pages. The page number is determined by accessing the value of the “page” variable.

```

<nav aria-label="pageNav">

    <ul class="pagination pagination-lg justify-content-center" >

        <li class="page-item"><a class="page-link" href="<?php if($page <= 1){ echo '#'; } else { echo
"?page=" . ($page - 1); } ?>" id="pageNav">Previous</a></li>

        <li class="page-item"><a class="page-link" href="<?php if($page >= $totalPagesNeeded){ echo '#'; }
else { echo "?page=" . ($page + 1); } ?>" id="pageNav">Next</a></li>

    </ul>

</nav>

```

8. COLLECTING MONEY FROM THE STORE

Play Therapy by Lenka De Villiers-Van Zyl makes use of *PayFast* to facilitate all transactions.

PayFast allows customers to make payments in various ways. The customer can for example make payments through the use of credit or cheque cards (*Visa, Mastercard*), debit card, instant EFT, *Mobicred, Masterpass, SnapScan* and similar payment applications (*PayFast, 2021*).

PayFast takes developers into account and provides source code that can be directly implemented into the website (*PayFast, 2021*). It is important to note that one has to create a merchant's account with *PayFast* to make use of these facilities (*PayFast, 2021*). *PayFast* has various methods that can be used to integrate the application with a website (*PayFast, 2021*).

For the website in question, a simple JavaScript paragraph was used that is linked to a HTML form (*PayFast, 2021*). The JavaScript is freely viewable on the web and for brevity's sake is not included in this document. It is important to show how it is incorporated into the HTML form.

```
<div class='container-fluid' style='padding-top: 2%>
```

```
    <form action='https://sandbox.payfast.co.za/eng/process'
name='form_c741fd0b9c9c00c2b842839588803224' onsubmit='return
click_c741fd0b9c9c00c2b842839588803224( this );' method='post'>
```

```
        <input type='hidden' name='cmd' value='_paynow'>
```

```
        <input type='hidden' name='receiver' value='10022985'>
```

```
        <input type='hidden' name='item_name' value='Play Therapy by Lenka De Villiers-Van Zyl'>
```

```
        <input type='hidden' name='amount' value= " ".$_SESSION['totalCart']. ">
```

```
        <input type='hidden' name='item_description' value='Play Therapy Products'>
```

```
        <input type='hidden' name='return_url' value='http://www.localhost/'>
```

```
    <table>
```

```
        <tr><td><font color='red'>*</font>&nbsp;Price</td><td><input type='text' name='custom_amount'
class='pricing' value= " ".$_SESSION['totalCart']. "></td></tr>
```

```
<tr><td colspan=2 align=center style='padding-top: 8%; '><input type='image'
src='https://www.payfast.co.za/images/buttons/light-large-paynow.png' width='174' height='59' alt='Pay Now'
style='margin-right: 15%' title='Pay Now with PayFast'></td></tr></table>
```

```
</form>
```

```
</div>
```

PayFast supplies a generic form with certain values that need to be filled in by the developer (PayFast, 2021). The developer firstly needs to link the form action to the appropriate *PayFast* link. For development purposes, *PayFast* provides a sandbox in which interaction with the application can be tested (PayFast, 2021). This link is shown to be used in the code provided. The “onsubmit” element calls the JavaScript function provided by *PayFast*.

The fields that need to be given values include:

- the merchant’s ID, in this case it is the value of “10022985” on the second input of the HTML form.
- the website’s name needs to be provided.
- the item(s)’s name and description need to be provided.
- the total cost of the transaction needs to be provided.
- the URL to which *PayFast* should redirect after the transaction must also be supplied.


(PayFast, 2021).

The client (administrator) receives the money of a transaction by it being deposited directly into their account by *PayFast*. The merchant’s account that one needs to create requires the necessary banking details in order to perform this transaction (PayFast, 2021).

9. CHECKING WEB TRAFFIC & STATISTICS

Google Analytics is used to track website traffic and for detailed analysis of *Play Therapy by Lenka De Villiers-Van Zyl*.

A unique measurement ID is supplied on the creation of a *Google Analytics* account (Google Analytics, 2021). The measurement ID is incorporated into a supplied JavaScript function that can then be added to every webpage that one wants to monitor (Google Analytics, 2021). The image below shows the created stream for *Play Therapy by Lenka De Villiers-Van Zyl* as well as the measurement ID to be used.

STREAM URL	STREAM NAME	MEASUREMENT ID
https://www.playtherapybyldvz.co.za	play	G-Q817YVX2Y3 
STREAM ID	STATUS	
2670497126	No data received in past 48 hours. Learn more	

(Google Analytics, 2021).

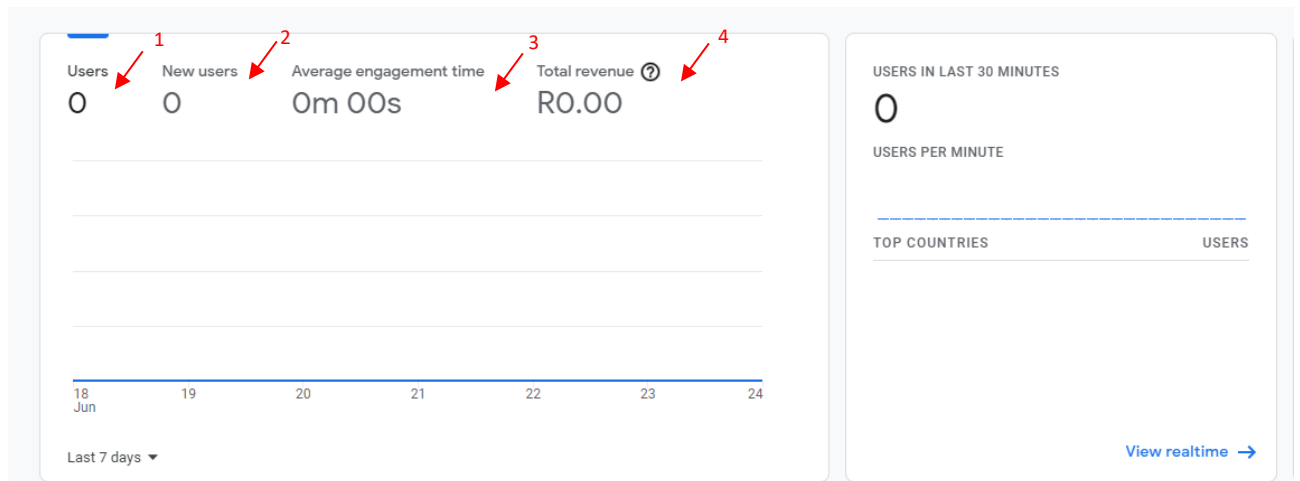
The following JavaScript, supplied by *Google Analytics*, is added to the header section of all pages that one wants to monitor (Google Analytics, 2021).

```
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id=G-Q817YVX2Y3"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'G-Q817YVX2Y3');
</script>
```

Google Analytics provides one with vital traffic statistics such as:

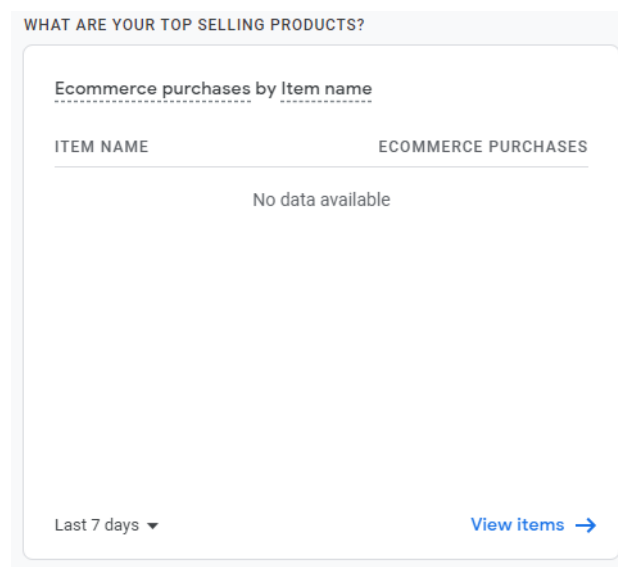
- 1) the number of users that the site has seen
- 2) number of new users
- 3) how long a user uses the site on average
- 4) the revenue generated by the website



(Google Analytics, 2021).

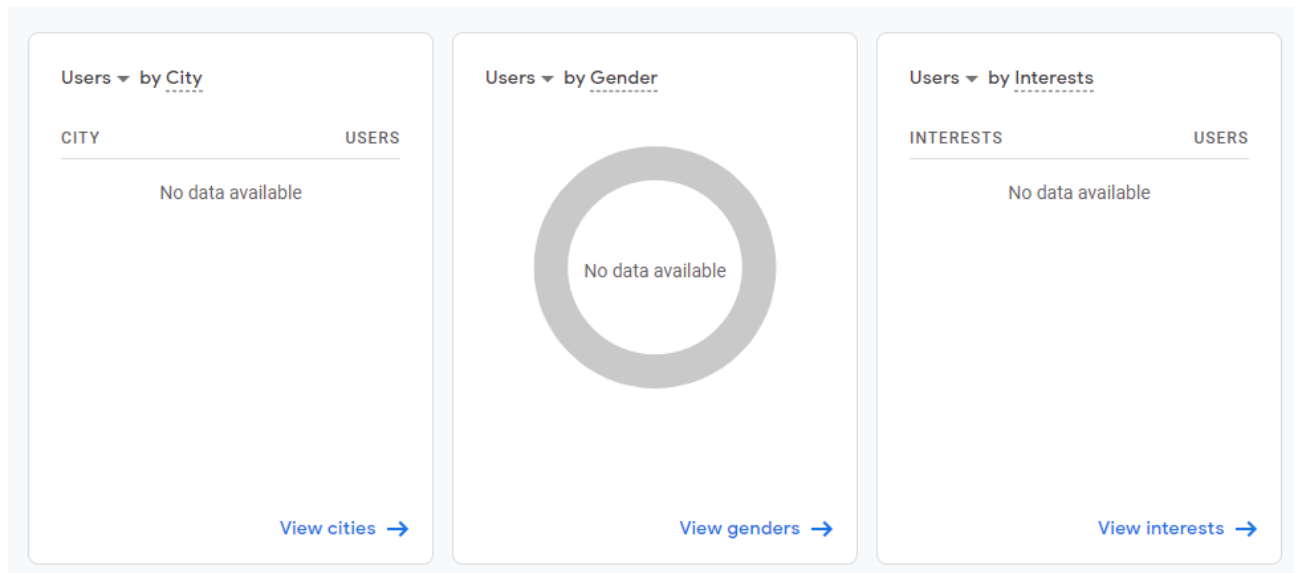
The above-mentioned information will be vital for *Play Therapy by Lenka De Villiers-Van Zyl*, as a start-up needs to be closely monitored.

Furthermore, *Google Analytics* can be used to track top selling products, vital for any ecommerce business. *Play Therapy by Lenka De Villiers-Van Zyl*'s owner can use this information to discover trends and gain an understanding of the products most popular with customers.



(Google Analytics, 2021).

Understanding one's target market is an important aspect of any business and Google Analytics makes it possible for *Play Therapy by Lenka De Villiers-Van Zyl's* owner to gain more knowledge of the target demographic visiting her website.

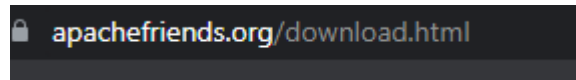


(Google Analytics, 2021).

10. APPENDIX

In order to run *Play Therapy by Lenka De Villiers-Van Zyl* the following steps need to be followed.

1. Navigate to the link displayed in the image below using any browser:



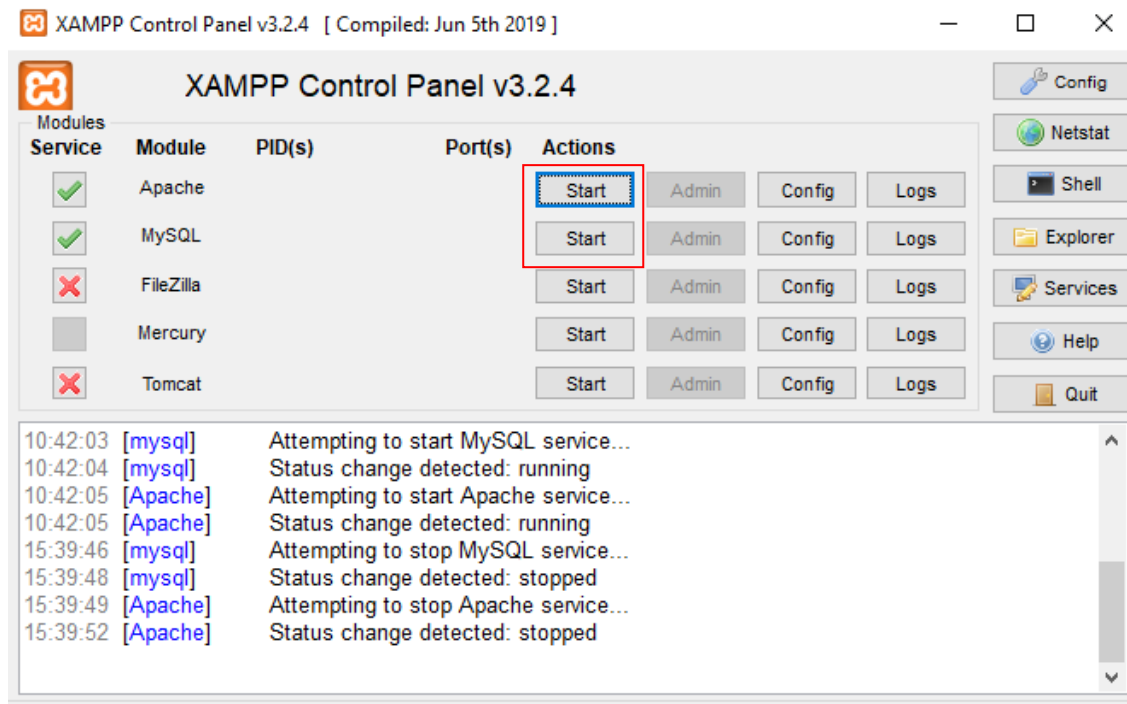
2. On this webpage, click on the download link that will download *Xampp* with PHP version 7.4

XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy.

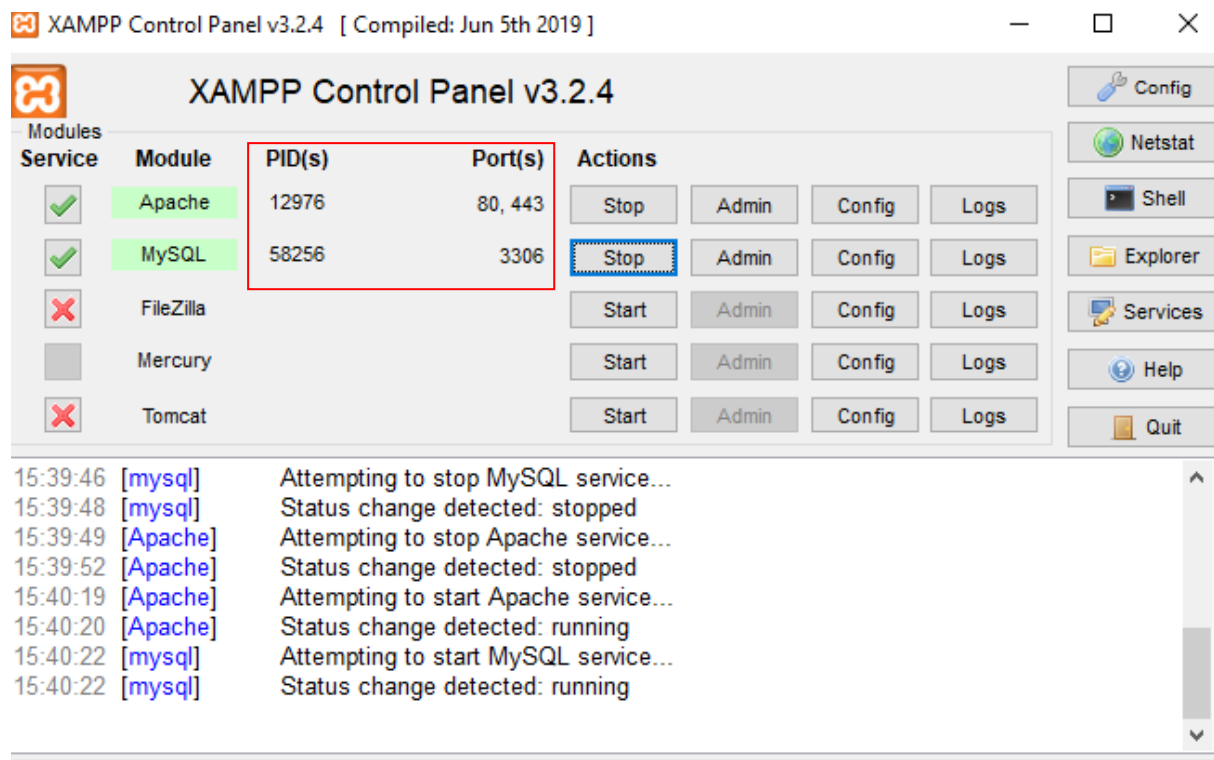
A screenshot of the XAMPP for Windows download page. The page title is 'XAMPP for Windows 7.3.28, 7.4.20 & 8.0.7'. Below the title is a table with three columns: 'Version', 'Checksum', and 'Size'. The table lists three versions: 7.3.28 / PHP 7.3.28, 7.4.20 / PHP 7.4.20, and 8.0.7 / PHP 8.0.7. The row for version 7.4.20 is highlighted with a red border. Below the table are links for 'Requirements', 'Add-ons', and 'More Downloads »'. At the bottom, a note states 'Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these'.

3. After the file has been downloaded, run the file and follow the installation steps.
4. After the installation has completed, run *Xampp* from the Windows “start menu”, shortcut on the desktop or from the *Xampp* folder within the C drive. It is important to right click on the *Xampp* icon when attempting to run the program and run it as administrator (or set administrative rights for the program in the folder).

5. Start the Apache and MySQL server by clicking on the respective “Start” buttons.



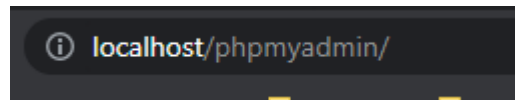
6. If the services are correctly configured, the *Xampp* window should display active PID(s) and Port(s) as shown below.



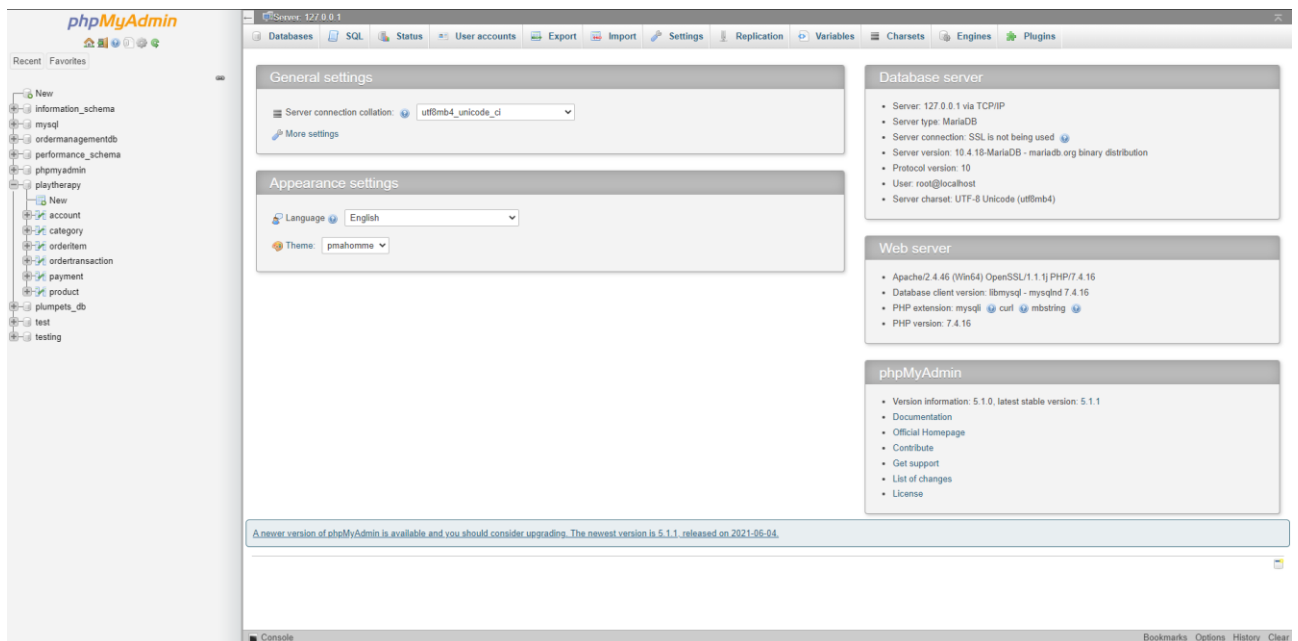
7. The next involves opening the *phpMyAdmin* panel. This can be done by either clicking on the admin button of the MySQL module:

Service	Module	PID(s)	Port(s)	Actions	
✓	Apache	12976	80, 443	Stop	Admin
✓	MySQL	58256	3306	Stop	Admin

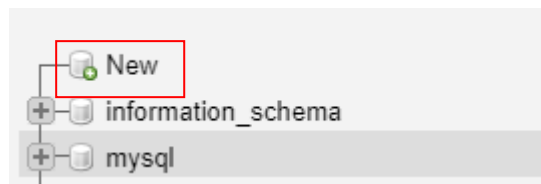
or by entering the URL displayed below in any browser:



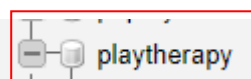
8. The following page should now be displayed:



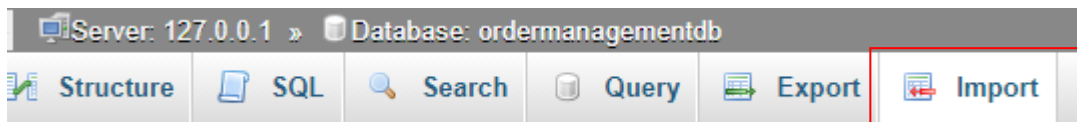
9. Click on the “new database” button and create a database with the name “playtherapy”. It is important to use this exact word and all lower-case lettering, otherwise the database will malfunction:



10. In the databases listed on the left panel, the database should now show:



11. Click on the database icon shown in step 10 to select it. Afterwards, click on the “Import” button on the top pane



12. The page displayed below should now appear. Click on the “Choose File” button.

Importing into the database "ordermanagementdb"

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in **[format]** [compression]. Example: .sql.zip

Browse your computer: Choose File No file chosen (Max: 40MiB)

You may also drag and drop a file on any page.

Character set of the file: utf-8

Partial import:

☒ Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. (This might be a good way to import large files, however it can break transactions.)

Skip this number of queries (for SQL) starting from the first one: 0

Other options:

☒ Enable foreign key checks

Format:

SQL

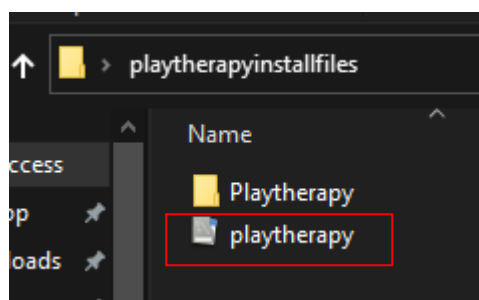
Format-specific options:

SQL compatibility mode: NONE

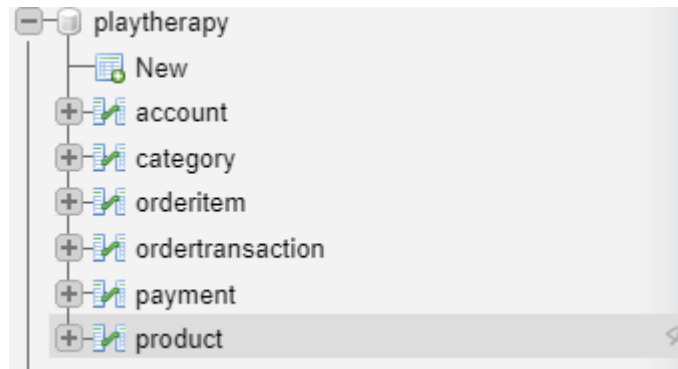
☒ Do not use AUTO_INCREMENT for zero values

Go

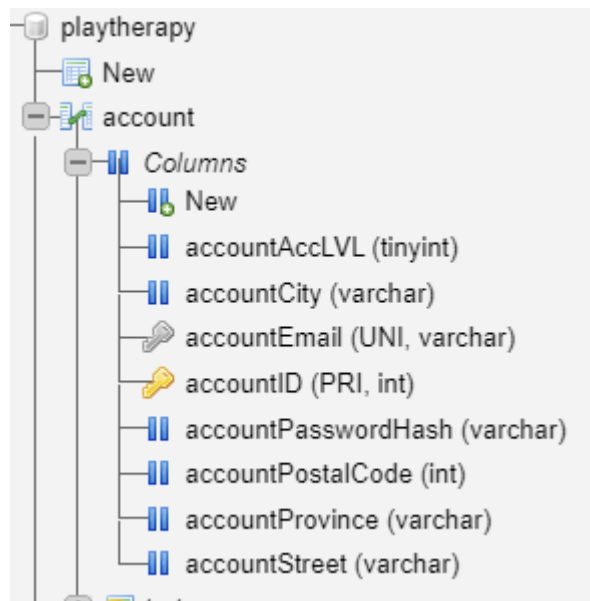
13. From the pop-up window, navigate to the stored folder labelled “playtherapyinstallfiles” and select the SQL file labelled “playtherapy”. This folder is supplied with the CD handed in during the presentation.



14. The database will now have been populated with the correct data. Clicking on the “playtherapy” database in the left pane should display the following:



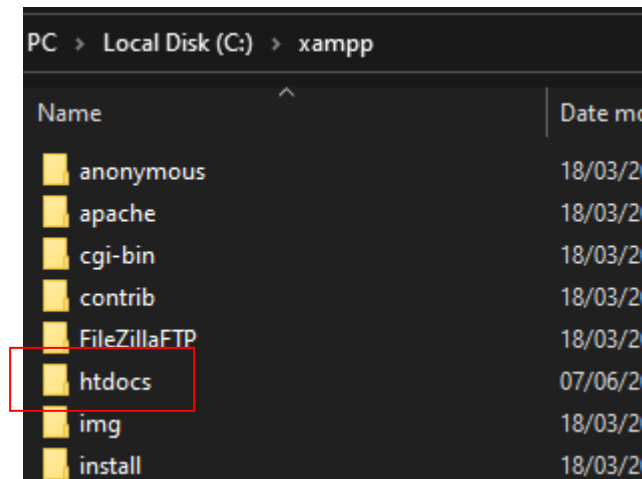
15. Ensure that the tables are all filled with columns, otherwise the import might have failed.



16. Furthermore, check that all the tables have entries in them.

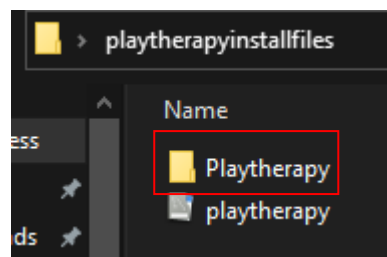
Options										
				productID	categoryID	productName	prodType	prodprice	prodQuantity	prodImg
<input type="checkbox"/>	Edit	Copy	Delete	1	2	CHILDRENS BOOK ABOUT ANXIETY: WHAT IF WORRIES WERE...	A colourfully illustrated children's book to help ...	139.99	3	[BLOB - 75.3 KiB]
<input type="checkbox"/>	Edit	Copy	Delete	33	1	DR. PLAYWELL'S CONTROLLING YOUR ANGER CARD GAME	Help kids take control of their anger before it ta...	799.99	1	[BLOB - 75.9 KiB]
<input type="checkbox"/>	Edit	Copy	Delete	34	1	DR. PLAYWELL'S POSITIVE THINKING CARD GAME	This card game is designed to encourage children t...	799.99	0	[BLOB - 78.0 KiB]

17. The next step involves navigating to the *Xampp* folder on the PC. Default installation should store it in the C drive. In the *Xampp* folder, navigate to the folder labelled "htdocs":

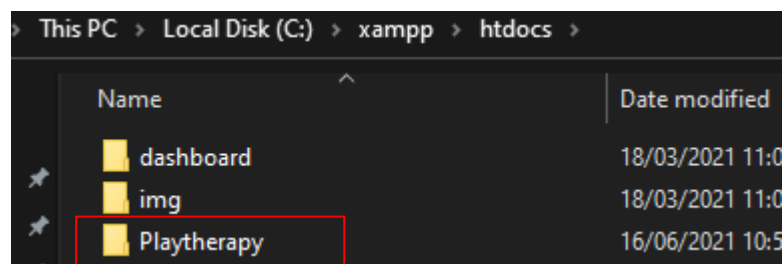


18. Open the folder and paste the folder labelled “Playtherapy”, found in the “playtherapyinstallfiles”, into the “htdocs” folder:

Below the labelled folder “Playtherapy”, found in the “playtherapyinstallfiles” folder is displayed:



Below, the folder “htdocs” is displayed after having pasted the “Playtherapy” folder into the folder.



19. Opening the “Playtherapy” folder located in the “htdocs” folder should display the following:

Name	Date modified	Type	Size
.git	19/06/2021 21:35	File folder	
backend	16/06/2021 10:40	File folder	
css	19/05/2021 11:22	File folder	
images	19/06/2021 19:39	File folder	
aboutUs	14/06/2021 08:10	PHP File	4 KB
addProduct	14/06/2021 08:11	PHP File	7 KB
admin	15/06/2021 19:43	PHP File	5 KB
cartScreen	20/06/2021 15:13	PHP File	7 KB
contactUs	20/06/2021 16:32	PHP File	5 KB
editAccounts	26/06/2021 12:44	PHP File	5 KB
editAdminOrders	15/06/2021 20:15	PHP File	5 KB
editAdminProducts	15/06/2021 19:41	PHP File	4 KB
editAdminUser	15/06/2021 19:41	PHP File	4 KB
editProducts	15/06/2021 20:07	PHP File	5 KB
footer	20/06/2021 16:19	PHP File	3 KB
header	20/06/2021 16:53	PHP File	3 KB
index	16/06/2021 18:01	JavaScript File	12 KB
index	20/06/2021 16:59	PHP File	11 KB
notExistScreen	19/06/2021 20:11	PHP File	3 KB
paymentScreen	25/06/2021 14:41	PHP File	9 KB
placeOrderScreen	15/06/2021 16:55	PHP File	6 KB
privacyPolicy	14/06/2021 08:24	PHP File	10 KB
productScreen	20/06/2021 17:17	PHP File	7 KB
productViewScreen	19/06/2021 18:48	PHP File	7 KB
productViewScreenBooks	19/06/2021 19:25	PHP File	7 KB
productViewScreenToys	19/06/2021 19:23	PHP File	7 KB
profileOrderScreen	15/06/2021 21:08	PHP File	5 KB
profileScreen	15/06/2021 21:08	PHP File	9 KB
shippingScreen	15/06/2021 20:42	PHP File	7 KB
signInScreen	14/06/2021 08:13	PHP File	4 KB
signUpScreen	16/06/2021 19:27	PHP File	9 KB
termsandconditions	14/06/2021 08:16	PHP File	14 KB

20. One can now navigate to the localhost to interact with the website. Please note that the server (MySQL and Apache) needs to be started in *Xampp* before opening the website. Depending on how *Xampp* is set up on the computer in question, one would either need to navigate to the link displayed:

www.localhost/index.php

or the link displayed here:

www.localhost/playtherapy/index.php

11. REFERENCING

Google Analytics. 2021. Analytics. *Google Analytics*. [Online] Available at: <https://analytics.google.com/analytics/> [Accessed 25 June 2021].

Guru99. 2021. PHP Session and PHP Cookies with Example. *Guru99*. [Online] Available at: <https://www.guru99.com/cookies-and-sessions.html> [Accessed 26 June 2021].

Kenlon, S. 2020. 5 reasons to use the Atom text editor. *OpenSource*, 24 December 2020. [Online] Available at: <https://opensource.com/article/20/12/atom> [Accessed 25 June 2021].

Ouellette, A. 2021. Bootstrap: A Beginner's Guide. *CareerFoundry*, 26 March 2021. [Online] Available at: <https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/> [Accessed 25 June 2021].

PayFast. 2021. Developer Documentation. *PayFast*. [Online] Available at: <https://developers.payfast.co.za/docs> [Accessed 25 June 2021].