Laborator 6-7

Problema 1. Activitate practică implementare aplicatie login

Să se creeze un proiect de autentificare în cadrul unui site. Acest proiect va avea pagina dedicată înregistrarii unui nou utilizator. Un utilizator inregistrat va putea să se logheze în site utilizând pagina de Login care îl va redirecta spre o pagina home.php în care este salutat și are opțiunea de logout cu ajutorul codului scris în pagina logout.php. Crearea unui cod sursă destinat autentificării presupune gestionarea tabelei utilizatori din baza de date magazin.

Rezolvare

Pagina autentificare.php

```
<?php
session_start();
// informatii conectare.
$DATABASE_HOST = 'localhost';
$DATABASE_USER = 'root';
$DATABASE_PASS = '';
$DATABASE_NAME = 'magazin';
// Încercați să vă conectați folosind informațiile de mai sus.
$con = mysqli_connect($DATABASE_HOST, $DATABASE_USER, $DATABASE_PASS, $DATABASE_NAME);
if ( mysqli_connect_errno() ) {
	// Dacă există o eroare la conexiune, opriți scriptul și afișați eroarea.
	exit('Esec conectare MySQL: ' . mysqli_connect_error());
}
// Acum verificăm dacă datele din formularul de autentificare au fost trimise, isset () va verifica dacă datele există.
if ( !isset($_POST['username'], $_POST['password']) ) {
	// Nu s-au putut obține datele care ar fi trebuit trimise.
	exit('Completati username si password !');
}
// Pregătiți SQL-ul nostru, pregătirea instrucțiunii SQL va împiedica injecția SQL.
if ($stmt = $con->prepare('SELECT id, password FROM utilizatori WHERE username = ?')) {
// Parametrii de legare (s = șir, i = int, b = blob etc.), în cazul nostru numele de utilizator este un șir, //așa că vom folosi „s”
	$stmt->bind_param('s', $_POST['username']);
	$stmt->execute();
	// Stocați rezultatul astfel încât să putem verifica dacă contul există în baza de date.
	$stmt->store_result();
```

```
if ($stmt->num_rows > 0) {
	$stmt->bind_result($id, $password);
	$stmt->fetch();
	// Contul există, acum verificăm parola.
// Notă: nu uitați să utilizați password_hash în fișierul de înregistrare pentru a stoca parolele hash.
	if (password_verify($_POST['password'], $password)) {
		// Verification success! User has loggedin!
// Creați sesiuni, astfel încât să știm că utilizatorul este conectat, acestea acționează practic ca cookie-//uri, dar rețin datele de pe server.
		session_regenerate_id();
		$_SESSION['loggedin'] = TRUE;
		$_SESSION['name'] = $_POST['username'];
		$_SESSION['id'] = $id;
		echo 'Bine ati venit' . $_SESSION['name'] . '!';
header('Location: home.php');
	} else {
		// password incorrect

echo 'Incorrect username sau  password!';
	}
} else {
	//  username incorect
	echo 'Incorrect username sau password!';
}
	$stmt->close();
}
?>
```

2. Pagina Index.html

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Login</title>
		<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
	</head>
	<body>
		<div class="login">
			<h1>Login</h1>
			<form action="authenticate.php" method="post">
```

```
                        <label for="username">
                            <i class="fas fa-user"></i>
                        </label>
        <input type="text" name="username" placeholder="Username"
id="username" required>
                        <label for="password">
                            <i class="fas fa-lock"></i>
                        </label>
                        <input type="password" name="password"
placeholder="Password" id="password" required>
                        <input type="submit" value="Login">
                    </form>
<div><a href="registration.html">Nou utilizator</a></div>
            </div>
        </body>
</html>
```

Pagina registration.html

```
<!DOCTYPE html>
<html>
        <head>
            <meta charset="utf-8">
            <title>Inregistrare</title>
            <link                                                rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
        </head>
        <body>
            <div class="register">
                <h1>Register</h1>
                <form          action="register.php"          method="post"
autocomplete="off">
                        <label for="username">
                            <i class="fas fa-user"></i>
                        </label>
                        <input          type="text"          name="username"
placeholder="Username" id="username" required>
                        <label for="password">
                            <i class="fas fa-lock"></i>
                        </label>
                        <input type="password" name="password"
placeholder="Parola" id="password" required>
                        <label for="email">
```

```
                                <i class="fas fa-envelope"></i>
                            </label>
                            <input          type="email"          name="email"
placeholder="Email" id="email" required>
                            <input type="submit" value="Register">
                        </form>
                </div>
        </body>
</html>
```

3.Pagina registration.php

```
<?php
// Schimbați acest lucru cu informațiile despre conexiune.
$DATABASE_HOST = 'localhost';
$DATABASE_USER = 'root';
$DATABASE_PASS = "";
$DATABASE_NAME = 'magazin';
// Incerc sa ma conectez pe baza info de mai sus.
$con      =      mysqli_connect($DATABASE_HOST,      $DATABASE_USER,
$DATABASE_PASS, $DATABASE_NAME);
if (mysqli_connect_errno()) {
        // Dacă există o eroare la conexiune, opriți scriptul și afișați eroarea.
        exit('Nu se poate conecta la MySQL: ' . mysqli_connect_error());
}
//.
if (!isset($_POST['username'], $_POST['password'], $_POST['email'])) {
        // Nu s-au putut obține datele care ar fi trebuit trimise.
        exit('Complare formular registration !');
}
// Asigurați-vă că valorile înregistrării trimise nu sunt goale.
if    (empty($_POST['username'])    ||    empty($_POST['password'])    ||
empty($_POST['email'])) {
        // One or more values are empty.
        exit('Completare registration form');
}
if (!filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
        exit('Email nu este valid!');
}
if (preg_match('/[A-Za-z0-9]+/', $_POST['username']) == 0) {
    exit('Username nu este valid!');
}
if (strlen($_POST['password']) > 20 || strlen($_POST['password']) < 5) {
```

```
        exit('Password trebuie sa fie intre 5 si 20 charactere!');
}
// verificam daca contul userului exista.
if ($stmt = $con->prepare('SELECT id, password FROM utilizatori WHERE username = ?')) {
        // hash parola folosind funcția PHP password_hash.
        $stmt->bind_param('s', $_POST['username']);
        $stmt->execute();
        $stmt->store_result();
        // Memoram rezultatul, astfel încât să putem verifica dacă contul există în baza de date.
        if ($stmt->num_rows > 0) {
                // Username exista
                echo 'Username exists, alegeti altul!';
        } else {
                if ($stmt = $con->prepare('INSERT INTO utilizatori (username, parola, email) VALUES (?, ?, ?)')) {
        // Nu dorim să expunem parole în baza noastră de date, așa că hash parola și utilizați //password_verify atunci când un utilizator se conectează.
        $password = password_hash($_POST['password'], PASSWORD_DEFAULT);
        $stmt->bind_param('sss', $_POST['username'], $password, $_POST['email']);
        $stmt->execute();
        echo Success inregistrat!';
header('Location: indexl.html');
} else {
        // Ceva nu este în regulă cu declarația sql, verificați pentru a vă asigura că tabelul conturilor //există cu toate cele 3 câmpuri.
        echo 'Nu se poate face prepare statement!';
}
        }
        $stmt->close();
} else {
        // Ceva nu este în regulă cu declarația sql, verificați pentru a vă asigura că tabelul conturilor //există cu toate cele 3 câmpuri.
        echo 'Nu se poate face prepare statement!';
}
$con->close();
?>
```

4.Pagina Home.php

```
<?php
// We need to use sessions, so you should always start sessions using the
below code.
session_start();
// If the user is not logged in redirect to the login page...
if (!isset($_SESSION['loggedin'])) {
	header('Location: index.php');
	exit;
}
?>

<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Pagina proiect parolata</title>
		<link href="style.css" rel="stylesheet" type="text/css">
		<link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
	</head>
	<body class="loggedin">
		<nav class="navtop">
			<div>
				<h1>TITLU WEBSITE</h1>
				<a href="profile.php"><i class="fas fa-user-
circle"></i>Profile</a>
				<a href="logout.php"><i class="fas fa-sign-out-
alt"></i>Logout</a>
			</div>
		</nav>
		<div class="content">
			<h2>Pagina cu parola</h2>
			<p>Bine ati revenit, <?=$_SESSION['name']?>!</p>
		</div>
	</body>
</html>
```

5. Pagina logout.php

```
<?php
session_start();
session_destroy();
```

```
// Redirectare paginaprincipala produse:
header('Location: magazin.php');
?>
```

Problema 2: Să se creeze o aplicație de tip cos cumpărături. Pagina principală a coșului conține o lista de produse cu un buton numit *Adaugă în coș"*. Când utilizatorul face clic pe butonul *„Adaugă în coș",*se redirecteaza spre aplicatia de Login-Registration, acțiune urmată de memorarea în tabela a ID-ul produsului selectat, cantitatea și ID-ul membru înregistrat sunt stocate în tabelul tbl_cart (ID-ul de membru este predefinit la începutul programului, unde puteți integra modulul de autentificare pentru a obține ID-ul de membru conectat). Pagina coș de cumpărături conține funcționalitățile de a adăuga articole în coș, de a scoate un singur articol din coș și de a goli coșul complet. Baza de date utilizată este magazin2.

Se crează pagina magazin.php ce contine produsele. Pozele produselor se află în sub directorul product-images salvate cu numele.jpg

```
<?php
require_once "ShoppingCart.php";?>
<HTML>
<HEAD>
<TITLE>Creare cos cumparaturi </TITLE>
<link href="style.css" type="text/css" rel="stylesheet" />
</HEAD>
<BODY>
<div id="product-grid">
   <div class="txt-heading"><div class="txt-heading-
label">Products</div></div>
   <?php
$shoppingCart = new ShoppingCart();
   $query = "SELECT * FROM tbl_product";
   $product_array = $shoppingCart->getAllProduct($query);
   if (! empty($product_array)) {
      foreach ($product_array as $key => $value) {
         ?>
      <div class="product-item">
      <form method="post" action="cos.php?action=add&code=<?php
echo $product_array[$key]["code"]; ?>">
         <div class="product-image">
            <img src="<?php echo $product_array[$key]["image"]; ?>">
         </div>
```

```
        <div>
          <strong><?php echo $product_array[$key]["name"];
?></strong>
        </div>
        <div class="product-price"><?php echo
"$".$product_array[$key]["price"]; ?></div>
        <div>
          <input type="text" name="quantity" value="1" size="2" />
          <input type="submit" value="Add to cart"
class="btnAddAction" />
        </div>
      </form>
   </div>
   <?php
      }
   }
   ?>
</div>
</BODY>
</HTML>
```

CRUD pe tabela tblcos și produse-pagina ShoppingCart.php

```
<?php
require_once "DBController.php";

class ShoppingCart extends DBController
{

   function getAllProduct()
   {
      $query = "SELECT * FROM tbl_product";

      $productResult = $this->getDBResult($query);
      return $productResult;
   }

   function getMemberCartItem($member_id)
   {
      $query = "SELECT tbl_product.*, tbl_cart.id as
cart_id,tbl_cart.quantity FROM tbl_product, tbl_cart WHERE
        tbl_product.id = tbl_cart.product_id AND tbl_cart.member_id = ?";
```

```
        $params = array(
            array(
                "param_type" => "i",
                "param_value" => $member_id
            )
        );

        $cartResult = $this->getDBResult($query, $params);
        return $cartResult;
    }

    function getProductByCode($product_code)
    {
        $query = "SELECT * FROM tbl_product WHERE code=?";

        $params = array(
            array(
                "param_type" => "s",
                "param_value" => $product_code
            )
        );

        $productResult = $this->getDBResult($query, $params);
        return $productResult;
    }

    function getCartItemByProduct($product_id, $member_id)
    {
        $query = "SELECT * FROM tbl_cart WHERE product_id = ? AND
member_id = ?";

        $params = array(
            array(
                "param_type" => "i",
                "param_value" => $product_id
            ),
            array(
                "param_type" => "i",
                "param_value" => $member_id
            )
        );
```

```
        $cartResult = $this->getDBResult($query, $params);
        return $cartResult;
    }

    function addToCart($product_id, $quantity, $member_id)
    {
        $query = "INSERT INTO tbl_cart (product_id,quantity,member_id)
VALUES (?, ?, ?)";

        $params = array(
            array(
                "param_type" => "i",
                "param_value" => $product_id
            ),
            array(
                "param_type" => "i",
                "param_value" => $quantity
            ),
            array(
                "param_type" => "i",
                "param_value" => $member_id
            )
        );

        $this->updateDB($query, $params);
    }

    function updateCartQuantity($quantity, $cart_id)
    {
        $query = "UPDATE tbl_cart SET  quantity = ? WHERE id= ?";

        $params = array(
            array(
                "param_type" => "i",
                "param_value" => $quantity
            ),
            array(
                "param_type" => "i",
                "param_value" => $cart_id
            )
        );
```

```
        $this->updateDB($query, $params);
    }

    function deleteCartItem($cart_id)
    {
        $query = "DELETE FROM tbl_cart WHERE id = ?";

        $params = array(
            array(
                "param_type" => "i",
                "param_value" => $cart_id
            )
        );

        $this->updateDB($query, $params);
    }

    function emptyCart($member_id)
    {
        $query = "DELETE FROM tbl_cart WHERE member_id = ?";

        $params = array(
            array(
                "param_type" => "i",
                "param_value" => $member_id
            )
        );

        $this->updateDB($query, $params);
    }
}
```

Pagina DBController.php

```
<?php

class DBController
{

    private $host = "localhost";

    private $user = "root";
```

```
    private $password = "";

    private $database = "magazin";

    private static $conn;

    function __construct()
    {
        $this->conn = mysqli_connect($this->host, $this->user, $this->password, $this->database);
    }

    public static function getConnection()
    {
        if (empty($this->conn)) {
            new Database();
        }
    }

    function getDBResult($query, $params = array())
    {
        $sql_statement = $this->conn->prepare($query);
        if (! empty($params)) {
            $this->bindParams($sql_statement, $params);
        }
        $sql_statement->execute();
        $result = $sql_statement->get_result();

        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $resultset[] = $row;
            }
        }

        if (! empty($resultset)) {
            return $resultset;
        }
    }

    function updateDB($query, $params = array())
    {
```

```
        $sql_statement = $this->conn->prepare($query);
        if (! empty($params)) {
            $this->bindParams($sql_statement, $params);
        }
        $sql_statement->execute();
    }

    function bindParams($sql_statement, $params)
    {
        $param_type = "";
        foreach ($params as $query_param) {
            $param_type .= $query_param["param_type"];
        }

        $bind_params[] = & $param_type;
        foreach ($params as $k => $query_param) {
            $bind_params[] = & $params[$k]["param_value"];
        }

        call_user_func_array(array(
            $sql_statement,
            'bind_param'
        ), $bind_params);
    }
}
```

Dacă userul este nou trebuie să se înregistreze în pagina registration.html pentru ca apoi să își genereze cosul.
Pagina indexl.html din care se poate alege pagina de registration.html pentru client nou dacă este client vechi se completeaza in aceasta pagina campurile Username si parola.

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <title>Login</title>
                <link                                                rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
        </head>
        <body>
                <div class="login">
```

```
                    <h1>Login</h1>
                    <form action="authenticate.php" method="post">
                        <label for="username">
                            <i class="fas fa-user"></i>
                        </label>
                        <input type="text" name="username"
placeholder="Username" id="username" required>
                        <label for="password">
                            <i class="fas fa-lock"></i>
                        </label>
                        <input type="password" name="password"
placeholder="Password" id="password" required>
                        <input type="submit" value="Login">
                    </form>
<div><a href="registration.html">Nou utilizator</a></div>
                </div>
        </body>
</html>
```

Pagina autentificare.php

```
<?php
session_start();
// informatii conectare.
$DATABASE_HOST = 'localhost';
$DATABASE_USER = 'root';
$DATABASE_PASS = '';
$DATABASE_NAME = 'magazin';
// Încercați să vă conectați folosind informațiile de mai sus.
$con = mysqli_connect($DATABASE_HOST, $DATABASE_USER,
$DATABASE_PASS, $DATABASE_NAME);
if ( mysqli_connect_errno() ) {
        // Dacă există o eroare la conexiune, opriți scriptul și afișați eroarea.
        exit('Failed to connect to MySQL: ' . mysqli_connect_error());
}
// Acum verificăm dacă datele din formularul de autentificare au fost trimise,
isset () va verifica dacă datele există.
if ( !isset($_POST['username'], $_POST['password']) ) {
        // Nu s-au putut obține datele care ar fi trebuit trimise.
        exit(‘Completati username si password !');
}
```

```
// Pregătiți SQL-ul nostru, pregătirea instrucțiunii SQL va împiedica injecția SQL.
if ($stmt = $con->prepare('SELECT id, password FROM users WHERE username = ?')) {
// Parametrii de legare (s = șir, i = int, b = blob etc.), în cazul nostru numele de utilizator este un șir, //așa că vom folosi „s”
	$stmt->bind_param('s', $_POST['username']);
	$stmt->execute();
	// Stocați rezultatul astfel încât să putem verifica dacă contul există în baza de date.
	$stmt->store_result();
if ($stmt->num_rows > 0) {
	$stmt->bind_result($id, $password);
	$stmt->fetch();
	// Contul există, acum verificăm parola.
// Notă: nu uitați să utilizați password_hash în fișierul de înregistrare pentru a stoca parolele hash.
	if (password_verify($_POST['password'], $password)) {
		// Verification success! User has loggedin!
// Creați sesiuni, astfel încât să știm că utilizatorul este conectat, acestea acționează practic ca cookie-//uri, dar rețin datele de pe server.
		session_regenerate_id();
		$_SESSION['loggedin'] = TRUE;
		$_SESSION['name'] = $_POST['username'];
		$_SESSION['id'] = $id;
		echo 'Welcome ' . $_SESSION['name'] . '!';
header('Location: cos.php');
	} else {
		// password incorrect

echo 'Incorrect username sau  password!';
	}
} else {
	//  username incorect
	echo 'Incorrect username sau password!';
}
	$stmt->close();
}
?>
```

Pagina registration.html

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Register</title>
		<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
	</head>
	<body>
		<div class="register">
			<h1>Register</h1>
			<form action="register.php" method="post" autocomplete="off">
				<label for="username">
					<i class="fas fa-user"></i>
				</label>
				<input type="text" name="username" placeholder="Username" id="username" required>
				<label for="password">
					<i class="fas fa-lock"></i>
				</label>
				<input type="password" name="password" placeholder="Password" id="password" required>
				<label for="email">
					<i class="fas fa-envelope"></i>
				</label>
				<input type="email" name="email" placeholder="Email" id="email" required>
				<input type="submit" value="Register">
			</form>
		</div>
	</body>
</html>
```

Pagina registration.php

```
<?php
// Schimbați acest lucru cu informațiile despre conexiune.
$DATABASE_HOST = 'localhost';
$DATABASE_USER = 'root';
$DATABASE_PASS = "";
$DATABASE_NAME = 'magazin';
```

```
// Incerc sa ma conectez pe baza info de mai sus.
$con = mysqli_connect($DATABASE_HOST, $DATABASE_USER, $DATABASE_PASS, $DATABASE_NAME);
if (mysqli_connect_errno()) {
	// Dacă există o eroare la conexiune, opriți scriptul și afișați eroarea.
	exit(‘Nu se poate conecta la MySQL: ' . mysqli_connect_error());
}
//.
if (!isset($_POST['username'], $_POST['password'], $_POST['email'])) {
	// Nu s-au putut obține datele care ar fi trebuit trimise.
	exit('Complare formular registration !');
}
// Asigurați-vă că valorile înregistrării trimise nu sunt goale.
if (empty($_POST['username']) || empty($_POST['password']) || empty($_POST['email'])) {
	// One or more values are empty.
	exit(‘Completare registration form');
}
if (!filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
	exit('Email nu este valid!');
}
if (preg_match('/[A-Za-z0-9]+/', $_POST['username']) == 0) {
   exit('Username nu este valid!');
}
if (strlen($_POST['password']) > 20 || strlen($_POST['password']) < 5) {
	exit('Password trebuie sa fie intre 5 si 20 charactere!');
}
// verificam daca contul userului exista.
if ($stmt = $con->prepare('SELECT id, password FROM users WHERE username = ?')) {
	// hash parola folosind funcția PHP password_hash.
	$stmt->bind_param('s', $_POST['username']);
	$stmt->execute();
	$stmt->store_result();
	// Memoram rezultatul, astfel încât să putem verifica dacă contul există în baza de date.
	if ($stmt->num_rows > 0) {
		// Username exista
		echo 'Username exists, alegeti altul!';
	} else {
		if ($stmt = $con->prepare('INSERT INTO users (username, password, email) VALUES (?, ?, ?)')) {
```

```
        // Nu dorim să expunem parole în baza noastră de date, așa că hash parola și utilizați //password_verify atunci când un utilizator se conectează.
        $password = password_hash($_POST['password'], PASSWORD_DEFAULT);
        $stmt->bind_param('sss', $_POST['username'], $password, $_POST['email']);
        $stmt->execute();
        echo Success inregistrat!';
header('Location: indexl.html');
} else {
        // Ceva nu este în regulă cu declarația sql, verificați pentru a vă asigura că tabelul conturilor //există cu toate cele 3 câmpuri.
        echo 'Nu se poate face prepare statement!';
}
        }
        $stmt->close();
} else {
        // Ceva nu este în regulă cu declarația sql, verificați pentru a vă asigura că tabelul conturilor //există cu toate cele 3 câmpuri.
        echo 'Nu se poate face prepare statement!';
}
$con->close();
?>
```

Pagina Cos.php

```
<?php
require_once "ShoppingCart.php";
session_start();
// Dacă utilizatorul nu este conectat redirecționează la pagina de autentificare ...
if (!isset($_SESSION['loggedin'])) {
        header('Location: indexl.html');
        exit;
}
// pt membrii inregistrati
$member_id=$_SESSION['loggedin'];

$shoppingCart = new ShoppingCart();
if (! empty($_GET["action"])) {
    switch ($_GET["action"]) {
        case "add":
```

```
            if (! empty($_POST["quantity"])) {

                $productResult = $shoppingCart->getProductByCode($_GET["code"]);

                $cartResult = $shoppingCart->getCartItemByProduct($productResult[0]["id"], $member_id);

                if (! empty($cartResult)) {
                    // Modificare cantitate in cos
                    $newQuantity = $cartResult[0]["quantity"] + $_POST["quantity"];
                    $shoppingCart->updateCartQuantity($newQuantity, $cartResult[0]["id"]);
                } else {
                    // Adaugare in tabelul cos
                    $shoppingCart->addToCart($productResult[0]["id"], $_POST["quantity"], $member_id);
                }
            }
            break;
        case "remove":
            // Sterg o sg inregistrare
            $shoppingCart->deleteCartItem($_GET["id"]);
            break;
        case "empty":
            // Sterg cosul
            $shoppingCart->emptyCart($member_id);
            break;
    }
}
?>
<HTML>
<HEAD>
<TITLE>creare cos permament in PHP</TITLE>
<link href="style.css" type="text/css" rel="stylesheet" />
</HEAD>
<BODY>
    <div id="shopping-cart">
        <div class="txt-heading">
```

```
        <div class="txt-heading-label">Cos Cumparaturi</div> <a
id="btnEmpty" href="cos.php?action=empty"><img src="empty-cart.png"
alt="empty-cart" title="Empty Cart" /></a>
      </div>
<?php
$cartItem = $shoppingCart->getMemberCartItem($member_id);
if (! empty($cartItem)) {
   $item_total = 0;
   ?>
<table cellpadding="10" cellspacing="1">
         <tbody>
            <tr>
               <th style="text-align: left;"><strong>Name</strong></th>
               <th style="text-align: left;"><strong>Code</strong></th>
               <th                                          style="text-align:
right;"><strong>Quantity</strong></th>
               <th                                          style="text-align:
right;"><strong>Price</strong></th>
               <th                                          style="text-align:
center;"><strong>Action</strong></th>
            </tr>
<?php
   foreach ($cartItem as $item) {
      ?>
                        <tr>
               <td
                  style="text-align: left; border-bottom: #F0F0F0 1px
solid;"><strong><?php echo $item["name"]; ?></strong></td>
               <td
                  style="text-align: left; border-bottom: #F0F0F0 1px
solid;"><?php echo $item["code"]; ?></td>
               <td
                  style="text-align: right; border-bottom: #F0F0F0 1px
solid;"><?php echo $item["quantity"]; ?></td>
               <td
                  style="text-align: right; border-bottom: #F0F0F0 1px
solid;"><?php echo "$".$item["price"]; ?></td>
               <td
                  style="text-align: center; border-bottom: #F0F0F0 1px
solid;"><a
                  href="cos.php?action=remove&id=<?php             echo
$item["cart_id"]; ?>"
```

```
                    class="btnRemoveAction"><img      src="icon-delete.png"
alt="icon-delete" title="Remove Item" /></a></td>
            </tr>
                                <?php
      $item_total += ($item["price"] * $item["quantity"]);
   }
   ?>

<tr>
               <td                                                    colspan="3"
align=right><strong>Total:</strong></td>
               <td align=right><?php echo "$".$item_total; ?></td>
               <td></td>
            </tr>
         </tbody>
      </table>
  <?php
}
?>
</div>
<div><a href="magazin.php">Alegeti alt produs</a></div>
<div><a          href="logout.php">Abandonati          sesiunea          de
cumparare</a></div>
<?php //require_once "product-list.php"; ?>

</BODY>
</HTML>
```

Pagina logout.php

```
<?php
session_start();
session_destroy();
// Redirectare paginaprincipala produse:
header('Location: magazin.php');
?>
```

Muncă individuală

Problema3. Să se regândească aplicația asfel încât să se poată gestiona o bază de date ce să conțină tabelele

Clienți
(Client_id,username,parola,email,strada,oras,tara,codpostal,nrcard,tipcard,dataexpcard,acceptareemail,nume,nrinregRC,cod_fiscal)

Produse(produs_id, produs_nume,pret, imagine, categorie, descriere, desc_completa, stare, oferta, noutati)
Cos(cos_id, Client_id, produs_id,cos_cantitate)
Comenzi(comenzi_id,produ_id,cantitate,clien_id,dataintrod,stare_comanda,data_cumparare)