МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забеспечення

**КУРСОВА РОБОТА**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни Інтернет програмування

на тему:

**«****CMS-система для Інтернет-магазину електронiки»**

студента ІІ курсу групи КН-21-1

спеціальності 122 «Комп’ютерні науки»

Левка Дениса Андрійовича

(прізвище, ім’я та по-батькові)

Керівник PhD, Марина ГРАФ

Дата захисту: " " 2023 р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Марина ГРАФ \_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Олександр КУЗЬМЕНКО . (підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Денис ФУРІХАТА \_

Житомир – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп’ютерних технологій

Кафедра інженерії програмного забеспечення

Освітній рівень: бакалавр

Спеціальність 122 «Комп’ютерні науки»

«ЗАТВЕРДЖУЮ»

Зав. кафедри

\_\_\_\_\_\_\_\_\_\_\_\_ Марина ГРАФ

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_2023 р.

ЗАВДАННЯ

НА КУРСОВИЙ ПРОЕКТ СТУДЕНТУ

Левку Денису Андрійовичу

1. Тема роботи: CMS-система для Інтернет-магазину електронiки,

керівник курсового проекту: к.т.н., доцент А.В. Морозов

1. Строк подання студентом: “ 23 ” травня 2023р.
2. Вихідні дані до роботи: CMS-система для Інтернет-магазину електронiки.
3. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

1. Аналіз аналогічних розробок;

2. Алгоритми роботи програми;

3. Опис роботи програми.

1. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Презентація\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Посилання на репозиторій: <https://github.com/denyafreak/qq>

Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання “ 14 ” вересня 2022 р.

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Левко Д.А.

(підпис) (прізвище та ініціали)

**Керівник проекту** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фуріхата. Д.В.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Завданням курсового проекту є розробка програмного забезпечення «Комплектація комп'ютера»

Пояснювальна записка до курсового проекту (роботи) на тему «Комплектація комп'ютера» складається з вступу, трьох розділів, висновків, списку використаної літератури та додатків.

Текстова частина викладена на 14 сторінках друкованого тексту. Пояснювальна записка має 5 сторінок додатків. Список використаних джерел містить 4 найменування і займає 1 сторінку. В роботі наведено 15 рисунків Загальний обсяг роботи – 25 сторінок.

Ключові слова: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ІНТЕРНЕТ МАГАЗИН, ТАБЛИЦЯ, ДОДАТОК, PHP, ТЕХНІКА.

ЗМІСТ

[РОЗДІЛ 1 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 6](#_Toc92738106)

[1.1. Проектування загального алгоритму роботи програми 6](#_Toc92738107)

[1.2. Розробка функціональних алгоритмів роботи програми 6](#_Toc92738108)

[2.3. Розробка контролерiв та моделей 8](#_Toc92738109)

[Висновки до другого розділу 11](#_Toc92738110)

[РОЗДІЛ 2 ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ 12](#_Toc92738111)

[2.1. Опис роботи з програмним додатком (Опис інтерфейсу) 12](#_Toc92738112)

[ВИСНОВКИ 18](#_Toc92738113)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 19](#_Toc92738114)

[Додатки 20](#_Toc92738115)

ВСТУП

**Актуальність теми.**

Тема і мова PHP є актуальною, і практичні навички, які були отримані під час виконання цієї роботи будуть викорастані в інших мовах програмування чи у майбутньому.

**Метою курсової роботи** є дослідження особливостей проектування та

реалізації cистеми адміністрування веб-сайтом за визначеним темою курсової роботи

**Завданням на курсову роботу є:**

- аналіз теоретичних засад проектування та реалізації CMS;

- аналіз існуючих веб-сайтів на обрану вами тематику, визначення унікальності;

- визначення інформаційних потреб предметної області дослідження;

- розробка адаптивного інтерфейсу веб-сайту за допомогою HTML5, CSS3, JavaScript;

- розробка математичної та алгоритмічної моделі функціонування системи на основі БД;

- проектування бази даних за визначеною предметною областю;

- написання серверної частини; - реалізація CMS.

**Об’екст дослідження.** є методи та засоби розробки CMS за

визначеними предметними областями.

**Предмет дослідження.** є використання веб-технологій для

забезпечення інформаційних потреб предметної області. В процесі роботи над курсовою роботою студент може використовувати монографічні, аналітичні, математичні, графічні методи, методи об‘єктно-орієнтованого проектування та програмування та інші методи дослідження.

# РОЗДІЛ 1 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 1.1. Проектування загального алгоритму роботи програми

На рис. 1.1. зображено загальний принцип роботи майбутньої програми

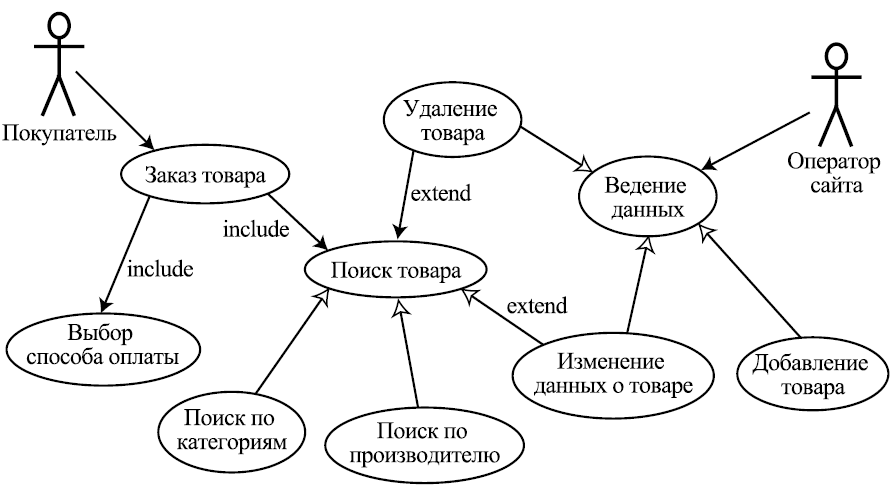


Рис. 1.1. Діаграма прецедентів

Перед нами поставлена задача складення загального алгоритму роботи сайту, що надасть можливість оцінити його складність та зробити висновки щодо подальшої розробки.

## 1.2. Розробка функціональних алгоритмів роботи програми

Аналізуючи стає зрозуміло, що спочатку відкривається сторiнка на якiй розташовуються елементи управління і виводу інформації, можливість вибрати товар, категорiю, зробити замовлення та аккаунт. Після ініціалізації вiкна автоматчно скриваються деякі елементи програмного додатку, які на даний момент не потрібні.

Далі користувач має такі можливості:

1. Вибрати товар
2. Подивитися iнформацiю про товар
3. Авторизуватися
4. Зареєструватися
5. Зробити замовлення
6. Написати вiдгук
7. Адмiнiстративна панель
8. Додавання товарiв, категорiй
9. Редагування замовлення
10. Редагування аккаунта
11. При вiдкриваннi сайту відбуваеться підключення до бази даних

class Db  
{  
 public static function getConnection()  
 {  
 $db = new PDO("mysql:host=localhost;dbname=phpshop","root","root");  
 $db->exec("set names utf8");  
 return $db;  
 }  
}

1. Методи перевiрки користувача на роль адмiна

public static function checkAdmin()  
{  
 $userId = User::*checkLogged*();  
 $user = User::*getUserById*($userId);  
  
 if ($user['role'] == 'admin') {  
 return true;  
 }  
 die('Access denied');  
}

перевіряємо чи є користувач адміном, якщо ні, він буде переадресован. Отримаємо інформацію про користувача якщо його роль адмін, допускаємо до адмін панелі, в іншому випадку завершуємо роботу з повідомленням

1. Метод автопiдключення класiв

function \_\_autoload($class\_name)  
{  
 $array\_paths = array(  
 '/models/',  
 '/components/',  
 '/controllers/',  
 );  
 foreach ($array\_paths as $path) {  
 $path = *ROOT* . $path . $class\_name . '.php';  
 if (is\_file($path)) {  
 include\_once $path;  
 }  
 }  
}

отримуємо масив папок де можуть знаходитись необхідні класи. Проходимось по масиву та формуєму шляхи якщо шлях є, підключаємо

1. Метод додавання товару, рахунку товарiв, отримання товарiв, отримання повної цiни, очищення товарiв та видалення

public static function addProduct($id)

public static function countItems()

public static function getProducts()

public static function getTotalPrice($prod)

public static function clear()

public static function deleteProduct($id)

додавання товару до кошика (з використанням сессій)

рахунок кількості товарів у кошику (з використанням сессій)

повертання масиву з ідентифікаторами та кількістью товару у кошику

рахунок ціни всіх товарів у кошику

очищення кошика

відаляє товар з кошика по id

1. Методи пагiнацiї та маршрутизацiї

Клас пагінації для генерації посторінкової навігації та компонент для праці з маршрутами

## 2.3. Розробка контролерiв та моделей

**Опис функціональної частини програмного забезпечення**

1. Контролер AdminCategoryController, управління категоріями товарів в адмінпанелі з методами

public function actionIndex()

public function actionCreate()  
{  
 // Проверка доступа  
 self::*checkAdmin*();  
 // Обработка формы  
 if (isset($\_POST['submit'])) {  
 // Если форма отправлена  
 // Получаем данные из формы  
 $name = $\_POST['name'];  
 $sortOrder = $\_POST['sort\_order'];  
 $status = $\_POST['status'];  
 // Флаг ошибок в форме  
 $errors = false;  
 // При необходимості можно валідувати значения потрібним чином  
 if (!isset($name) || empty($name)) {  
 $errors[] = 'Заполните поля';  
 }  
 if ($errors == false) {  
 // Якщо все ок, додаємо нову категорію  
 Category::*createCategory*($name, $sortOrder, $status);  
 // Перенаправляємо користувача на сторінку керування категоріями  
 header("Location: /admin/category");  
 }  
 }  
 require\_once(*ROOT* . '/views/admin\_category/create.php');  
 return true;  
}

public function actionUpdate($id)

public function actionDelete($id)

керування категоріями з перевіркою доступу

додавання категорій з перевіркою доступу

редагування категорій з перевіркою доступу

видалення категорій з перевіркою доступу

1. Контролер AdminController головна сторінка в адмінпанелі

public function actionIndex()  
{

// Перевірка доступу  
 self::*checkAdmin*();  
 require\_once(*ROOT* . '/views/admin/index.php');  
 return true;  
}

Action для стартовою сторінки "Панель адміністратора"

1. Контролер AdminOrderController управління замовленнями в адмінпанелі

public function actionIndex()  
{  
 self::*checkAdmin*();  
 $ordersList = Order::*getOrdersList*();  
 require\_once(*ROOT* . '/views/admin\_order/index.php');  
 return true;  
}

public function actionUpdate($id)

public function actionView($id)

public function actionDelete($id)

керування замовленями з перевіркою доступу

редагування замовлення з перевіркою доступу

перегляд замовлення з перевіркою доступу

видалення замовлення з перевіркою доступу

1. Контролер AdminProductController управління товарами в адмінпанелі

public function actionIndex()

public function actionCreate()

public function actionUpdate($id)

public function actionDelete($id)

керування товарів з перевіркою доступу

додавання товарів з перевіркою доступу

редагування товарів з перевіркою доступу

видалення товарів з перевіркою доступу

1. Контролер CabinetController кабінет користувача

public function actionIndex()  
{  
 $userId = User::*checkLogged*();  
 $user = User::*getUserById*($userId);  
 require\_once(*ROOT* . '/views/cabinet/index.php');  
 return true;  
}  
  
public function actionEdit()

Action для сторінки "Кабінет користувача"

Редагування даних користувача

1. Контролер CartController кошик

public function actionIndex()

public function actionCheckout()

public function actionAdd($id)

public function actionDelete($id)

Action для додавання товару до кошика синхронним запитом

Action для додавання товару до корзини за допомогою асинхронного запиту (ajax)

Action для сторінки "Оформлення покупки" та "Кошик"

1. Контролер CatalogController каталог товарів

public function actionIndex()

public function actionCategory($categoryId, $page = 1)  
{  
 // Список категорій для лівого меню  
 $categories = Category::*getCategoriesList*();  
 // Список товарів в категорії  
 $categoryProducts = Product::*getProductsListByCategory*($categoryId, $page);  
 // Вся кількість товарів (для навигації)  
 $total = Product::*getTotalProductsInCategory*($categoryId);  
 // Створємо обєкт Pagination  
 $pagination = new Pagination($total, $page, Product::*SHOW\_BY\_DEFAULT*, 'page-');  
 // вид  
 require\_once(*ROOT* . '/views/catalog/category.php');  
 return true;  
}

1. Контролер ProductController товар

public function actionView($productId)  
{  
 // Список категорій для лівого меню  
 $categories = Category::*getCategoriesList*();  
 // Получаем инфомрацію про товар  
 $product = Product::*getProductById*($productId);  
 // вид  
 require\_once(*ROOT* . '/views/product/view.php');  
 return true;  
}

1. Контролер SiteController

public function actionIndex()

public function actionContact()

public function actionAbout()

Action для сторінки "Головна", "Контакти" та "Про нас"

1. Контролер UserController

public function actionRegister()

public function actionLogin()

public function actionLogout()

метод для реєстрації

метод для входу у аккаунт

метод виходу з аккаунта

1. Моделі для роботи з інформацією з бази даних та методи для них

## Висновки до другого розділу

В цьому розділі було спроектувано загальний алгоритм роботи програми. Вибрано головні елементи даної програми. Програма готова для подальшого тестування.

# РОЗДІЛ 2 ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ

## 2.1. Опис роботи з програмним додатком (Опис інтерфейсу)

При запуску сайту спочатку відкривається вікно програми (Рис. 2.1).

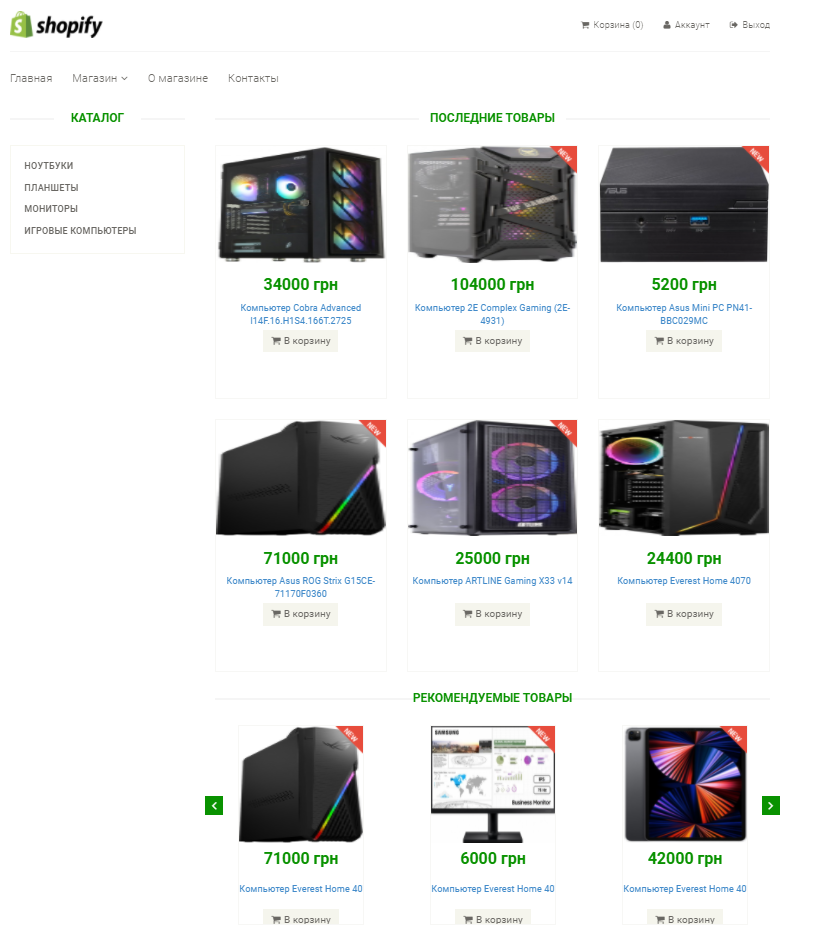


Рис 2.1. Скріншот головної сторінки

Далі можна починати роботу з сайтом. Виберемо категорію.

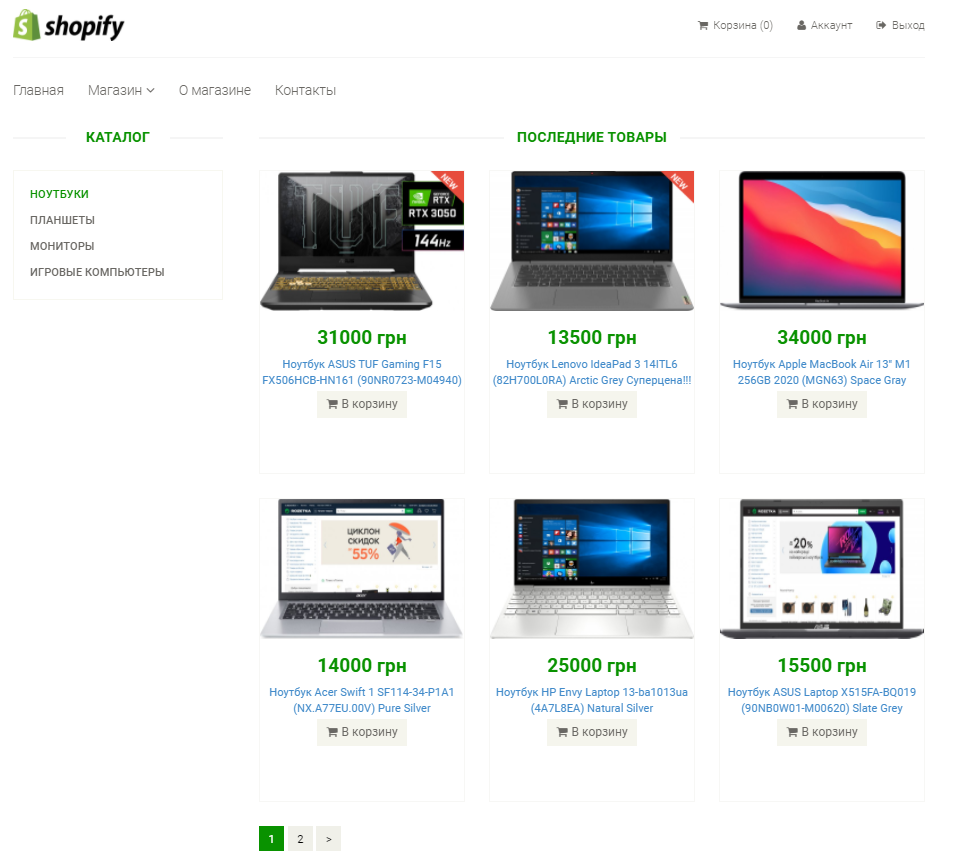


Рис. 2.2. Категорії

Як ми бачимо категорію «Ноутбуки» з різними моделями ноутбуків

Виберемо ноуттбук який нам подобається

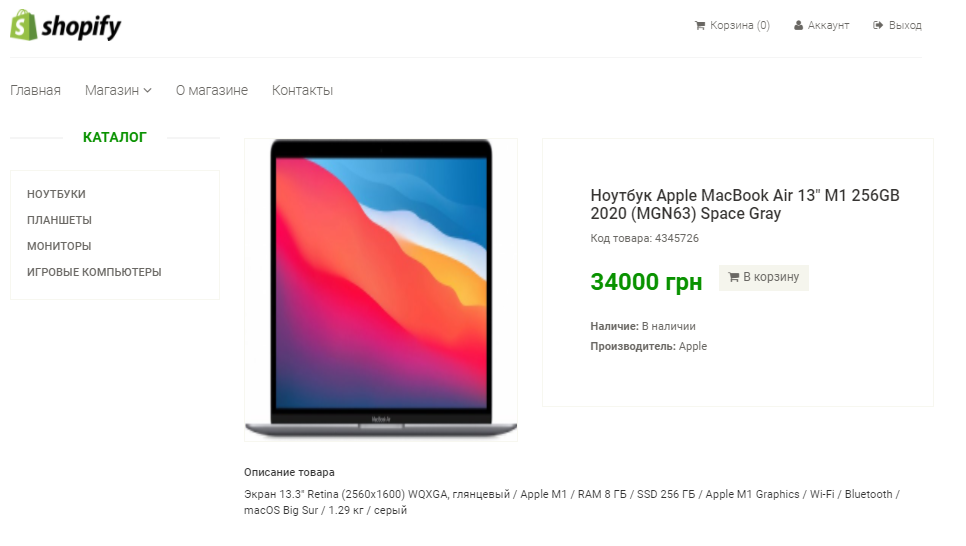


Рис. 2.3. Вибраний ноутбук

Натиснемо кнопку «В корзину» та побачимо, що товар доданий



Рис. 2.4. Кількість товарів у кошику

Перейдем на сторінку «Корзина»

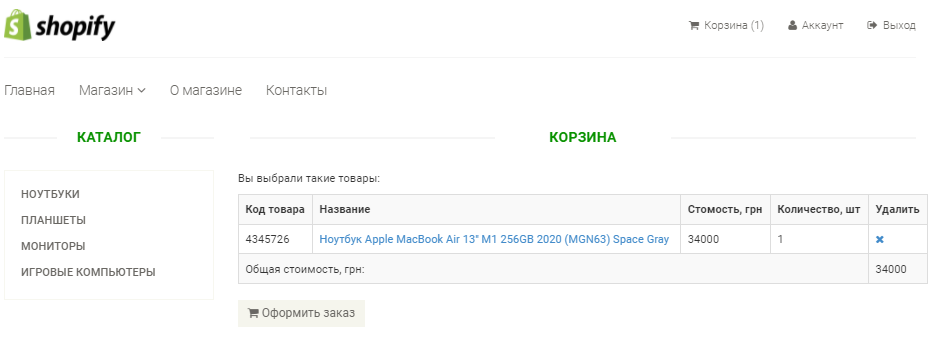


Рис. 2.5. Сторінка «Кошик»

Зробимо оформлення заказу

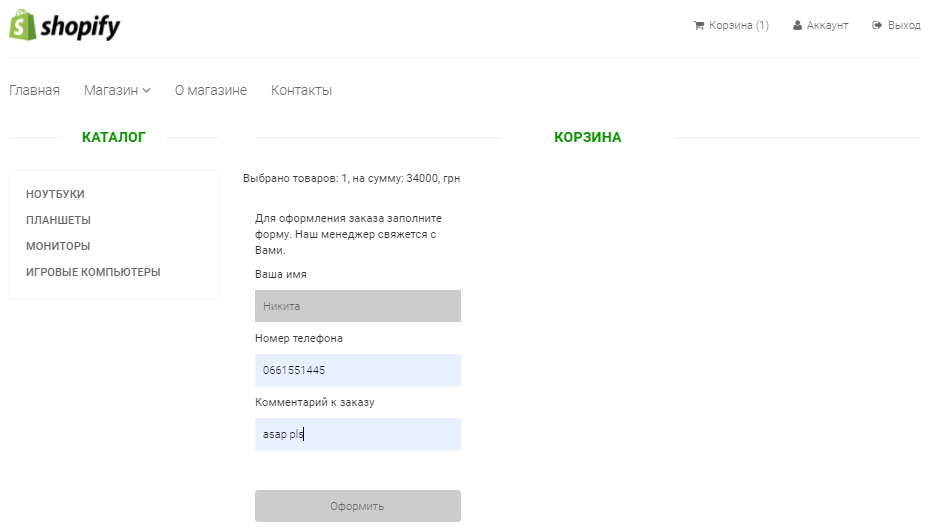


Рис. 2.6. Сторінка оформлення заказу



Рис. 2.7. Повідомлення про прийняття заказу

Увійдем на аккаунт Адміна та подивимось закази

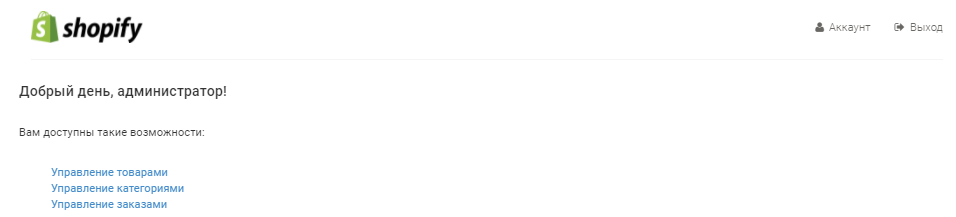


Рис. 2.8. Сторінка Адмінпанелі

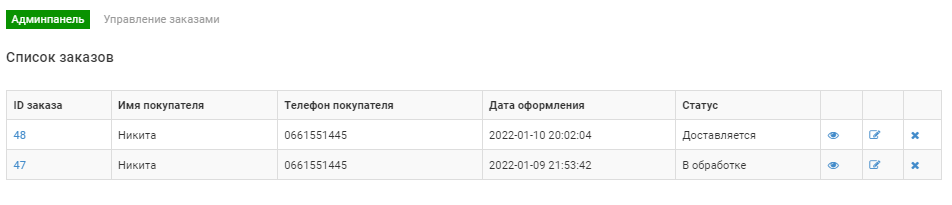


Рис. 2.9. Список заказів

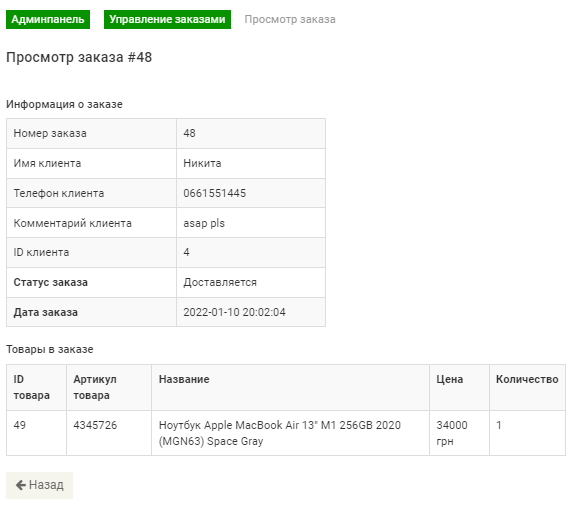


Рис. 2.10. Повна інфорація про заказ

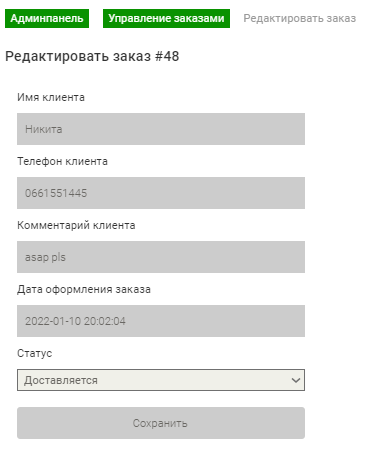
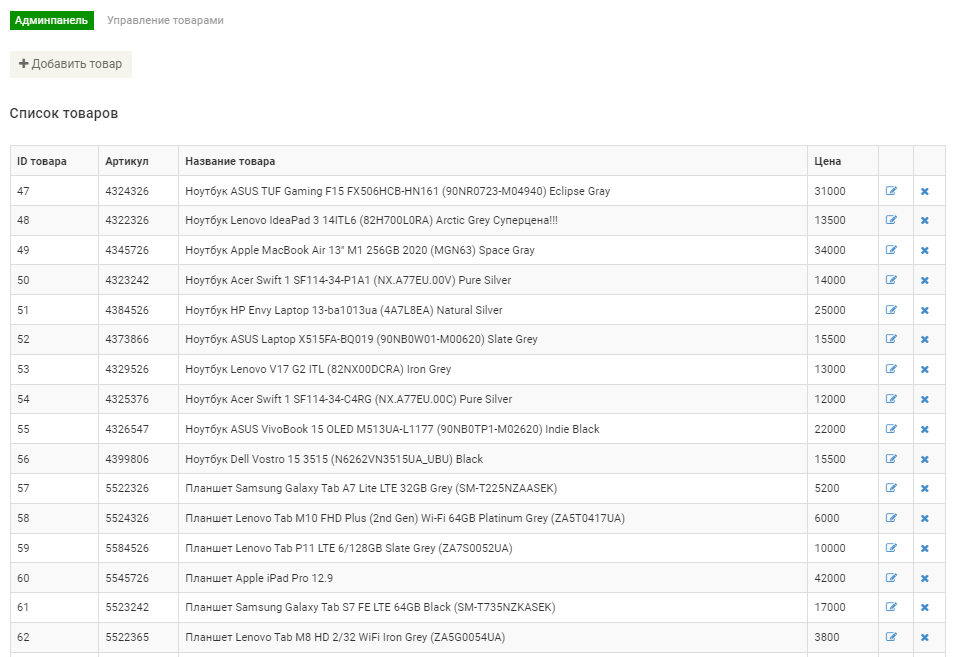


Рис. 2.11. Редагування заказу

Подивимось товари та категорії



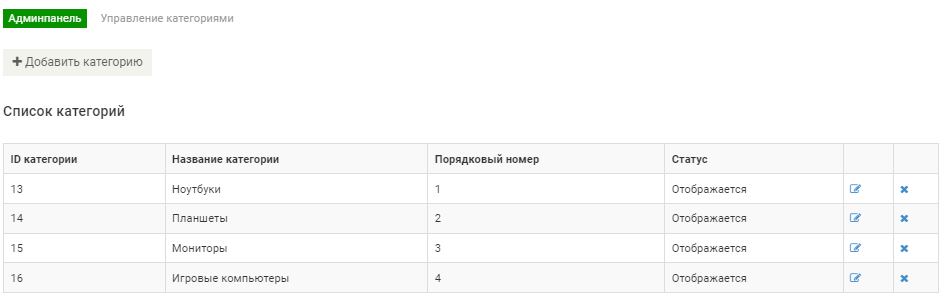


Рис. 2.12. Список товарів/категорій

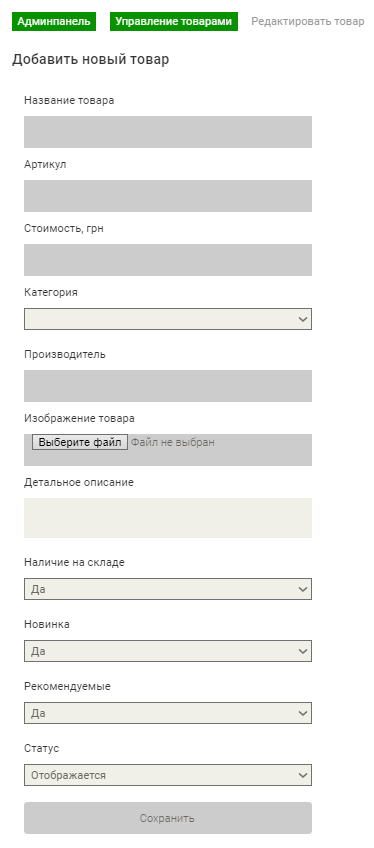
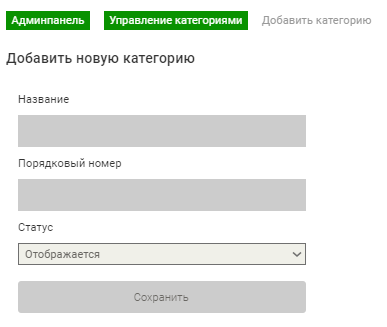
 

Рис. 2.13. Додавання товару/категорію

# ВИСНОВКИ

Під час виконання курсової роботи були визначенна основна мета курсової роботи, було обмірковане завдання, була обміркована актуальність даної задачі, проведений аналіз задачі, і в даному програмному додатку реалізовані наступні функції.

* Стабільна робота
* Вибор потрібної комплектаціїї
* Дизайнерські можливості
* Заказ товару, та керування ним
* Адмінпанель

Були представлені описи елементів програмного забеспечення, була написана інструкція. Був створений функціонуючий програмний додаток, були отримані практичні новички роботи з текстовими та бінарними файлами.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Руководство по PHP 8 - METANIT.COM:  
   <https://metanit.com/php/tutorial/>
2. Web-технології. Ч.2  
   <https://learn.ztu.edu.ua/course/view.php?id=3240>
3. Андрій Морозов Розробка CMS-системи на мові PHP

<https://www.youtube.com/channel/UCyQiI9vbRVvIug0V9Y7C76Q>

1. Victor Zinchenko PHP Start | Практика

<https://www.youtube.com/c/ВикторЗинченкоА/featured>

# Додатки

<?php  
class AdminCategoryController extends AdminBase  
{  
  
 public function actionIndex()  
 {  
 self::*checkAdmin*();  
 $categoriesList = Category::*getCategoriesListAdmin*();  
 require\_once(*ROOT* . '/views/admin\_category/index.php');  
 return true;  
 }  
  
 public function actionCreate()  
 {  
 self::*checkAdmin*();  
  
 if (isset($\_POST['submit'])) {  
 $name = $\_POST['name'];  
 $sortOrder = $\_POST['sort\_order'];  
 $status = $\_POST['status'];  
 $errors = false;  
  
 if (!isset($name) || empty($name)) {  
 $errors[] = 'Заполните поля';  
 }  
  
 if ($errors == false) {  
 Category::*createCategory*($name, $sortOrder, $status);  
  
 header("Location: /admin/category");  
 }  
 }  
  
 require\_once(*ROOT* . '/views/admin\_category/create.php');  
 return true;  
 }  
 public function actionUpdate($id)  
 {  
 self::*checkAdmin*();  
  
 $category = Category::*getCategoryById*($id);  
  
 if (isset($\_POST['submit'])) {  
 $name = $\_POST['name'];  
 $sortOrder = $\_POST['sort\_order'];  
 $status = $\_POST['status'];  
  
 Category::*updateCategoryById*($id, $name, $sortOrder, $status);  
  
 header("Location: /admin/category");  
 }  
  
 require\_once(*ROOT* . '/views/admin\_category/update.php');  
 return true;  
 }  
  
 public function actionDelete($id)  
 {  
 self::*checkAdmin*();  
  
 if (isset($\_POST['submit'])) {  
 Category::*deleteCategoryById*($id);  
  
 header("Location: /admin/category");  
 }  
  
 require\_once(*ROOT* . '/views/admin\_category/delete.php');  
 return true;  
 }  
  
}

<?php  
class AdminOrderController extends AdminBase  
{  
 public function actionIndex()  
 {  
 self::*checkAdmin*();  
 $ordersList = Order::*getOrdersList*();  
  
 require\_once(*ROOT* . '/views/admin\_order/index.php');  
 return true;  
 }  
 public function actionUpdate($id)  
 {  
 self::*checkAdmin*();  
 $order = Order::*getOrderById*($id);  
  
 if (isset($\_POST['submit'])) {  
 $userName = $\_POST['userName'];  
 $userPhone = $\_POST['userPhone'];  
 $userComment = $\_POST['userComment'];  
 $date = $\_POST['date'];  
 $status = $\_POST['status'];  
  
 Order::*updateOrderById*($id, $userName, $userPhone, $userComment, $date, $status);  
  
 header("Location: /admin/order/view/$id");  
 }  
  
 require\_once(*ROOT* . '/views/admin\_order/update.php');  
 return true;  
 }  
 public function actionView($id)  
 {  
 self::*checkAdmin*();  
  
 $order = Order::*getOrderById*($id);  
  
 $prodQuant = json\_decode($order['products'], true);  
  
 $prodIds = array\_keys($prodQuant);  
 $products = Product::*getProdustsByIds*($prodIds);  
  
 require\_once(*ROOT* . '/views/admin\_order/view.php');  
 return true;  
 }  
 public function actionDelete($id)  
 {  
 self::*checkAdmin*();  
  
 if (isset($\_POST['submit'])) {  
 Order::*deleteOrderById*($id);  
  
 header("Location: /admin/order");  
 }  
  
 require\_once(*ROOT* . '/views/admin\_order/delete.php');  
 return true;  
 }  
  
}

<?php  
class AdminProductController extends AdminBase  
{  
 public function actionIndex()  
 {  
 self::*checkAdmin*();  
 $productsList = Product::*getProductsList*();  
  
 require\_once(*ROOT* . '/views/admin\_product/index.php');  
 return true;  
 }  
 public function actionCreate()  
 {  
 self::*checkAdmin*();  
 $categList = Category::*getCategoriesListAdmin*();  
  
 if (isset($\_POST['submit'])) {  
 $options['name'] = $\_POST['name'];  
 $options['code'] = $\_POST['code'];  
 $options['price'] = $\_POST['price'];  
 $options['category\_id'] = $\_POST['category\_id'];  
 $options['brand'] = $\_POST['brand'];  
 $options['availability'] = $\_POST['availability'];  
 $options['description'] = $\_POST['description'];  
 $options['is\_new'] = $\_POST['is\_new'];  
 $options['is\_recommended'] = $\_POST['is\_recommended'];  
 $options['status'] = $\_POST['status'];  
  
 $errors = false;  
  
 if (!isset($options['name']) || empty($options['name'])) {  
 $errors[] = 'Заполните поля';  
 }  
  
 if ($errors == false) {  
 $id = Product::*createProduct*($options);  
  
 if ($id) {  
 if (is\_uploaded\_file($\_FILES["image"]["tmp\_name"])) {  
 move\_uploaded\_file($\_FILES["image"]["tmp\_name"], $\_SERVER['DOCUMENT\_ROOT'] . "/upload/images/products/{$id}.jpg");  
 }  
 };  
  
 header("Location: /admin/product");  
 }  
 }  
  
 require\_once(*ROOT* . '/views/admin\_product/create.php');  
 return true;  
 }  
 public function actionUpdate($id)  
 {  
 self::*checkAdmin*();  
  
 $categoriesList = Category::*getCategoriesListAdmin*();  
  
 $product = Product::*getProductById*($id);  
  
 if (isset($\_POST['submit'])) {  
 $options['name'] = $\_POST['name'];  
 $options['code'] = $\_POST['code'];  
 $options['price'] = $\_POST['price'];  
 $options['category\_id'] = $\_POST['category\_id'];  
 $options['brand'] = $\_POST['brand'];  
 $options['availability'] = $\_POST['availability'];  
 $options['description'] = $\_POST['description'];  
 $options['is\_new'] = $\_POST['is\_new'];  
 $options['is\_recommended'] = $\_POST['is\_recommended'];  
 $options['status'] = $\_POST['status'];  
  
 if (Product::*updateProductById*($id, $options)) {  
  
 if (is\_uploaded\_file($\_FILES["image"]["tmp\_name"])) {  
 move\_uploaded\_file($\_FILES["image"]["tmp\_name"], $\_SERVER['DOCUMENT\_ROOT'] . "/upload/images/products/{$id}.jpg");  
 }  
 }  
 header("Location: /admin/product");  
 }  
  
 require\_once(*ROOT* . '/views/admin\_product/update.php');  
 return true;  
 }  
 public function actionDelete($id)  
 {  
 self::*checkAdmin*();  
 if (isset($\_POST['submit'])) {  
 Product::*deleteProductById*($id);  
  
 header("Location: /admin/product");  
 }  
 require\_once(*ROOT* . '/views/admin\_product/delete.php');  
 return true;  
 }  
  
}

<?php  
class Category  
{  
 public static function getCategoriesList()  
 {  
 $db = Db::*getConnection*();  
 $result = $db->query('SELECT id, name FROM category WHERE status = "1" ORDER BY sort\_order, name ASC');  
  
 $i = 0;  
 $categoryList = array();  
 while ($row = $result->fetch()) {  
 $categoryList[$i]['id'] = $row['id'];  
 $categoryList[$i]['name'] = $row['name'];  
 $i++;  
 }  
 return $categoryList;  
 }  
  
 public static function getCategoriesListAdmin()  
 {  
 $db = Db::*getConnection*();  
 $result = $db->query('SELECT id, name, sort\_order, status FROM category ORDER BY sort\_order ASC');  
  
 $categoryList = array();  
 $i = 0;  
 while ($row = $result->fetch()) {  
 $categoryList[$i]['id'] = $row['id'];  
 $categoryList[$i]['name'] = $row['name'];  
 $categoryList[$i]['sort\_order'] = $row['sort\_order'];  
 $categoryList[$i]['status'] = $row['status'];  
 $i++;  
 }  
 return $categoryList;  
 }  
  
 public static function deleteCategoryById($id)  
 {  
 $db = Db::*getConnection*();  
  
 $sql = 'DELETE FROM category WHERE id = :id';  
  
 $result = $db->prepare($sql);  
 $result->bindParam(':id', $id, PDO::*PARAM\_INT*);  
 return $result->execute();  
 }  
  
 public static function updateCategoryById($id, $name, $sortOrder, $status)  
 {  
 $db = Db::*getConnection*();  
 $sql = "UPDATE category  
 SET name = :name, sort\_order = :sort\_order, status = :status  
 WHERE id = :id";  
  
 $result = $db->prepare($sql);  
 $result->bindParam(':id', $id, PDO::*PARAM\_INT*);  
 $result->bindParam(':name', $name, PDO::PARAM\_STR);  
 $result->bindParam(':sort\_order', $sortOrder, PDO::*PARAM\_INT*);  
 $result->bindParam(':status', $status, PDO::*PARAM\_INT*);  
 return $result->execute();  
 }  
  
 public static function getCategoryById($id)  
 {  
 $db = Db::*getConnection*();  
  
 $sql = 'SELECT \* FROM category WHERE id = :id';  
  
 $result = $db->prepare($sql);  
 $result->bindParam(':id', $id, PDO::*PARAM\_INT*);  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $result->execute();  
  
 return $result->fetch();  
 }  
  
 public static function getStatusText($status)  
 {  
 switch ($status) {  
 case '1':  
 return 'Отображается';  
 break;  
 case '0':  
 return 'Скрыта';  
 break;  
 }  
 }  
  
 public static function createCategory($name, $sortOrder, $status)  
 {  
 $db = Db::*getConnection*();  
  
 $sql = 'INSERT INTO category (name, sort\_order, status) '  
 . 'VALUES (:name, :sort\_order, :status)';  
  
 $result = $db->prepare($sql);  
 $result->bindParam(':name', $name, PDO::PARAM\_STR);  
 $result->bindParam(':sort\_order', $sortOrder, PDO::*PARAM\_INT*);  
 $result->bindParam(':status', $status, PDO::*PARAM\_INT*);  
 return $result->execute();  
 }  
}

<?php  
class User  
{  
 public static function register($name, $email, $password)  
 {  
 $db = Db::*getConnection*();  
  
 $sql = 'INSERT INTO user (name, email, password) '  
 . 'VALUES (:name, :email, :password)';  
  
 $result = $db->prepare($sql);  
 $result->bindParam(':name', $name, PDO::PARAM\_STR);  
 $result->bindParam(':email', $email, PDO::PARAM\_STR);  
 $result->bindParam(':password', $password, PDO::PARAM\_STR);  
 return $result->execute();  
 }  
  
 public static function edit($id, $name, $password)  
 {  
 $db = Db::*getConnection*();  
  
 $sql = "UPDATE user   
 SET name = :name, password = :password   
 WHERE id = :id";  
  
 $result = $db->prepare($sql);  
 $result->bindParam(':id', $id, PDO::*PARAM\_INT*);  
 $result->bindParam(':name', $name, PDO::PARAM\_STR);  
 $result->bindParam(':password', $password, PDO::PARAM\_STR);  
 return $result->execute();  
 }  
  
 public static function checkUserData($email, $password)  
 {  
 $db = Db::*getConnection*();  
 $sql = 'SELECT \* FROM user WHERE email = :email AND password = :password';  
  
 $result = $db->prepare($sql);  
 $result->bindParam(':email', $email, PDO::*PARAM\_INT*);  
 $result->bindParam(':password', $password, PDO::*PARAM\_INT*);  
 $result->execute();  
  
 $user = $result->fetch();  
  
 if ($user) {  
 return $user['id'];  
 }  
 return false;  
 }  
  
 public static function auth($userId)  
 {  
 $\_SESSION['user'] = $userId;  
 }  
  
 public static function checkLogged()  
 {  
 if (isset($\_SESSION['user'])) {  
 return $\_SESSION['user'];  
 }  
  
 header("Location: /user/login");  
 }  
  
 public static function isGuest()  
 {  
 if (isset($\_SESSION['user'])) {  
 return false;  
 }  
 return true;  
 }  
  
 public static function checkName($name)  
 {  
 if (strlen($name) >= 2) {  
 return true;  
 }  
 return false;  
 }  
  
 public static function checkPhone($phone)  
 {  
 if (strlen($phone) >= 10) {  
 return true;  
 }  
 return false;  
 }  
  
 public static function checkPassword($password)  
 {  
 if (strlen($password) >= 6) {  
 return true;  
 }  
 return false;  
 }  
  
 public static function checkEmail($email)  
 {  
 if (filter\_var($email, *FILTER\_VALIDATE\_EMAIL*)) {  
 return true;  
 }  
 return false;  
 }  
  
 public static function checkEmailExists($email)  
 {  
 $db = Db::*getConnection*();  
  
 $sql = 'SELECT COUNT(\*) FROM user WHERE email = :email';  
  
 $result = $db->prepare($sql);  
 $result->bindParam(':email', $email, PDO::PARAM\_STR);  
 $result->execute();  
  
 if ($result->fetchColumn())  
 return true;  
 return false;  
 }  
  
 public static function getUserById($id)  
 {  
 $db = Db::*getConnection*();  
  
 $sql = 'SELECT \* FROM user WHERE id = :id';  
  
 $result = $db->prepare($sql);  
 $result->bindParam(':id', $id, PDO::*PARAM\_INT*);  
  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $result->execute();  
  
 return $result->fetch();  
 }  
}