Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Криклі Володимира Андрійовича*
(ПІБ)

академічної групи *121-19-2*
(шифр)

напряму підготовки *121 Інженерія програмного забезпечення*
(код і назва напряму підготовки)

на тему: *Розробка веб-орієнтованого додатку з продажу настільних ігор та головоломок за допомогою React.JS, PHP у середовищі програмування Visual Studio Code*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*Проф. Бердник М.Г.*			
розділів:				
спеціальний	*Проф. Бердник М.Г.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*доц. Шедловський І.А*			
Нормоконтролер	*ст. викл. Мартиненко А.А.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 20 23 року

ЗАВДАННЯ
на дипломний проект

бакалавра
(назва освітньо-кваліфікаційного рівня)

студенту *121-19-2* *Криклі Володимиру Андрійовичу*
(група) (прізвище та ініціали)

Тема дипломного проекту *Розробка веб-орієнтованого додатку з продажу настільних ігор та головоломок за допомогою React.JS, PHP у середовищі програмування Visual Studio Code*

затверджена наказом ректора НТУ «ДП» від *16 . 05 .2023 р.* № *350 -с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПО й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *Проф. Бердник М.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Крикля В.А.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14 .01 .2023 р.*

Термін подання дипломного проекту до ДЕК ***12 .06 .2023 р.***

РЕФЕРАТ

Пояснювальна записка: 88 с., 34 рис., 3 дод., 29 джерел.

Об'єкт розробки: веб-додаток з продажу настільних ігор та головоломок.

Мета кваліфікаційної роботи: розробка інтернет-магазину за допомогою React.JS, PHP у середовищі програмування Visual Studio Code з інтуїтивно-зрозумілим інтерфейсом для задоволення потреб клієнтів.

У вступній частині кваліфікаційної роботи проведено аналіз та розглянуто сучасний стан проблеми, визначено конкретну мету дослідження та галузь її застосування, наведено обґрунтування актуальності обраної теми та деталізовано постановку задачі.

У першому розділі кваліфікаційної роботи було проведено детальний аналіз предметної галузі, розглянуто актуальність поставленої задачі та цілей розробки, сформульовано конкретну постановку завдання, а також визначено вимоги до програмної реалізації, використовуваних технологій та програмних засобів.

У другому розділі кваліфікаційної роботи було проаналізовано наявні рішення, обрані платформи для розробки, проведено проектування та розробка програми. Також була надана детальна інформація про роботу програми, включаючи опис алгоритму та структури її функціонування. В розділі було розглянуто процес виклику та завантаження програми, визначено вхідні та вихідні дані, а також надано характеристику параметрів технічних засобів.

У економічному розділі кваліфікаційної роботи була визначена трудомісткість розробленої інформаційної системи. Крім того, було проведено розрахунок вартості роботи зі створення програми та оцінено необхідний час для її реалізації.

Практичне значення полягає у розробці веб-додатку зі зрозумілим інтерфейсом, який дозволяє користувачам безпечно купувати необхідні товари.

Актуальність даного програмного продукту визначається значним попитом на здійснення дистанційних покупок у мережі інтернет, що значно спрощує процес отримання товару клієнтом та економить фінансові витрати на відкриття стаціонарного магазину.

Список ключових слів: КОМП’ЮТЕР, ІНТЕРНЕТ, МАГАЗИН, ТОВАРИ, REACT, PHP, НАСТІЛЬНІ ІГРИ, ГОЛОВОЛОМКИ.

ABSTRACT

Explanatory note: 88 p., 34 fig., 3 appx., 29 sources.

Development Object: Web application for selling board games and puzzles.

The objective of this qualification work is to develop an online store using React.JS and PHP in the Visual Studio Code programming environment with an intuitive user interface to meet the needs of customers.

The introduction section of the qualification work provides an analysis and discussion of the current state of the problem, identifies the specific research objective and the field of its application, justifies the relevance of the chosen topic, and specifies the task formulation in detail.

In the first chapter of the qualification work, a detailed analysis of the subject area was conducted, the relevance of the set task and development goals were discussed, the specific task formulation was formulated, and the requirements for software implementation, technologies, and tools used were determined.

The second chapter of the qualification work analyzed existing solutions, selected development platforms, performed project design and development. Detailed information about the program's operation, including the description of the algorithm and the structure of its functioning, was provided. The chapter also covers the process of program invocation and loading, determination of input and output data, and characterization of technical device parameters.

In the economic chapter of the qualification work, the complexity of the developed information system was determined. Additionally, the cost calculation for program development and the estimated time for its implementation were conducted.

The practical significance lies in the development of a web application with a user-friendly interface that allows users to securely purchase necessary products.

The relevance of this software product is determined by the significant demand for online shopping, which greatly simplifies the process of acquiring goods for customers and saves financial costs for opening a physical store.

Keywords: COMPUTER, INTERNET, STORE, PRODUCTS, REACT, PHP, BOARD GAMES, PUZZLES.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

СУБД – система управління базами даних.

HTML (HyperText Markup Language) – основна мова розмітки веб-сторінок.

JS (JavaScript) – високорівнева мова програмування, яка переважно використовується для веб-розробки.

React – відкрита бібліотека JavaScript для розробки інтерфейсів користувача.

JSX (JavaScript XML) – розширений синтаксис, що використовується в React для написання JavaScript-коду, який нагадує синтаксис HTML/XML.

PHP – мова програмування загального призначення, яка широко використовується для розробки веб-додатків і скриптів.

CSS – мова стилів, яка використовується для визначення вигляду і форматування веб-сторінок.

ВСТУП

Інтернет-торгівля стає все більш популярною, оскільки надає можливість швидко та зручно придбати товари без виходу з дому. Відкриття власного інтернет-магазину може бути перспективним і вигідним бізнесом, зокрема для тих, кому не вистачає коштів для відкриття стаціонарної торгової точки. Це також дозволяє швидко реагувати на нові тенденції та пропозиції на ринку і використовувати можливості реклами в інтернеті для просування бізнесу. Доступність віртуального магазину є важливою перевагою, оскільки він може бути відкритий для будь-якої категорії підприємців і не потребує великих знань та часових витрат для його управління.

Якщо для відкриття стаціонарного магазину не вистачає коштів, то інтернет-магазин може стати хорошою можливістю для успішного бізнесу. Багато перспективних ідей можна успішно реалізувати в Інтернеті, так як нові продукти та послуги швидко з'являються на ринку і привертають увагу покупців в Інтернеті. Часові рамки таких ідей можуть бути дуже короткими, і саме тому важливо відкрити онлайн-магазин вчасно, щоб привернути увагу нових клієнтів та розширити бізнес. Інтернет-ресурс також може використовуватися для рекламних заходів та представлення компанії на міжнародному рівні. Успіх проекту залежить від продукту, який обирається, а також від організаторських здібностей підприємця.

Один з головних плюсів інтернет-магазину полягає у його доступності для будь-яких підприємців. Якщо відсутній час або знання управління, можна купити готову торгову платформу та продовжувати її розвиток. Сьогодні створення інтернет-магазину не є дуже дорогим, а ця тактика є корисною для тих, хто має досвід роботи з віртуальними платформами. Якщо товари та послуги не обмежені однією територією, краще орієнтуватися на потенційних клієнтів у великому місті, де контингент користувачів значно ширший. Райони обласних центрів та мегаполісів мають багато потужностей виробників, що спрощує логістичні завдання при роботі з замовленнями.

Ця робота має на меті створення інтернет-магазину з використанням бібліотеки React.JS. Оскільки інтернет-магазини стали дуже популярними, особливо в період карантину, то розробка власного інтернет-магазину є важливим кроком для бізнесу в сучасному світі. Завдяки технологіям React.JS можна створити швидкий та зручний інтерфейс для користувачів, що підвищує ефективність роботи магазину та забезпечує задоволення користувачів від процесу купівлі. Інтернет-магазин можуть володіти як малі стартапи, так і великі корпорації, тому створення власного магазину за допомогою React.JS може бути вигідним та не вимагати великих витрат.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної області

Інтернет-магазини - це електронні ресурси, які дозволяють користувачам здійснювати покупки через Інтернет. Вони з'явилися в результаті зростання популярності онлайн-шопінгу і стали дуже популярними серед споживачів по всьому світу.

Основна перевага інтернет-магазинів полягає у тому, що вони дозволяють покупцям купувати товари з будь-якого місця, не виходячи з дому. Це особливо зручно для людей з обмеженими можливостями або тих, хто має зайнятий розклад.

Інтернет-магазини можуть пропонувати різноманітний товарний асортимент, включаючи електроніку, одяг, косметику, продукти харчування та інше. Крім того, у інтернет-магазинах часто діють знижки та акції, що дозволяє зекономити гроші на покупках.

Підприємці, які бажають створити свій власний інтернет-магазин, можуть скористатися різноманітними готовими платформами, які дозволяють створити сайт без додаткових знань програмування. Крім того, вони можуть скористатися послугами спеціалізованих компаній, які займаються розробкою та просуванням інтернет-магазинів.

Інтернет-магазини стали невід'ємною частиною сучасного ринку та продажів, і вони продовжують зростати в популярності. Якщо вони використовуються ефективно, то можуть стати вигідним та зручним варіантом для покупців та підприємців.

1.2. Призначення розробки та область застосування

Веб-додаток який розробляється, призначений для використання клієнтами по всій території України, які зацікавлені у придбанні товарів цієї категорії. Основною областю застосування є продаж та доставка головоломок та настільних ігор на території України. Також може бути запропонована консультація клієнтів щодо вибору товару, різноманітних акцій та знижок.

1.3. Підстава для розробки

Підставами для розробки та виконання кваліфікаційної роботи) є:

- освітня програма 121 «Інженерія програмного забезпечення»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;
- завдання на кваліфікаційну роботу на тему «Розробка веб-орієнтованого додатку з продажу настільних ігор та головоломок за допомогою React.JS, PHP у середовищі програмування Visual Studio Code».

1.4. Постановка завдання

Розробити веб-додаток з продажу настільних ігор та головоломок для задоволення потреб клієнтів за допомогою React.JS, PHP у середовищі програмування Visual Studio Code. Основні пункти:

- розробка зручного та інтуїтивно зрозумілого інтерфейсу для користувачів, який дозволяє швидко знайти потрібний товар та здійснити замовлення;
- розробка функціоналу для управління товарами, включаючи можливість додавання нових товарів, редагування цін та опису, видалення старих;

- розробка системи оплати та доставки, яка дозволяє користувачам зручно та безпечно оплачувати замовлення та отримувати товари у зручний для них спосіб;
- розробка функціоналу для управління особистим кабінетом користувача для додавання інформації, яка потрібна при обробці замовлення тощо.

1.4.1. Функціональні особливості, актуальність сайту та можливості

Функціональні особливості:

- каталог товарів: розміщення фото, опису і ціни товару;
- кошик покупок: можливість додавати товари в кошик, редагувати їх кількість та видаляти з нього;
- реєстрація: форма, що дозволяє користувачеві зареєструватися в базі магазину, здійснювати покупки та керувати особистою інформацією: додавати дані для доставки та редагувати їх;
- адміністративна панель: панель керування, яка дозволяє власникам магазину керувати товарами, редагувати їх, видаляти та додавати нові до бази товарів.

Ці функціональні особливості є ключовими для забезпечення зручності та ефективності роботи інтернет-магазину настільних ігор головоломок.

Створення інтернет-магазину з продажу головоломок та настільних ігор є актуальним у наш час і має потенційні переваги для бізнесу. Зростання популярності онлайн-шопінгу, особливо під час пандемії COVID-19, зробило інтернет-магазини все більш затребуваними споживачами. Крім того, продаж головоломок та настільних ігор має свій власний ринок, що забезпечує попит на такі товари серед любителів ігор та головоломок. Таким чином, створення інтернет-магазину з таким асортиментом товарів може бути ефективним рішенням для підприємства, яке має за мету продаж товарів у веб-просторі, що дає змогу охопити більшу аудиторії клієнтів та не витрачатися на відкриття

фізичного магазину. На мою думку зараз саме той момент, коли люди починають віддавати перевагу саме інтернет-магазинам через їх зручність та доступність.

Можливості сайту:

- моніторинг товарів на сайті, їх цін та опису;
- реєстрація у базі клієнтів;
- вхід до облікового запису, що надає можливість здійснювати покупки, редагувати додаткову інформацію про себе, поповнювати баланс;
- додавання товарів до кошику, видалення з нього;
- у адміністраторів: додавання нових товарів у базу прямо з інтерфейсу веб-додатку, видалення їх та редагування;
- перегляд вкладок з інформацією про сайт, у яких описуються: основна інформація про розробника, його контакти та адреса у формі Google Maps.

1.4.2. Опис інтерфейсу користувача

1) Користувач переходить на головну сторінку за посиланням.
2) Перше на що слід звернути увагу, це кнопки сайту та їх назви. За назвою кожної кнопки інтуїтивно можна здогадатися, яку функцію вона виконує.
3) Для користувача в верхній частині сайту є кнопки:

- Увійти – кнопка, при натисканні якої з’являється форма для авторизації у системі, якщо користувач зареєстрований у базі;
- Реєстрація – кнопка, при натисканні якої з’являється форма для реєстрації користувача у базі клієнтів магазину. Після успішної реєстрації користувача повертає на головну сторінку для подальшої авторизації;
- Про нас, логотип сайту – кнопки, при натисканні на які з’являється форма з основною інформацією про розробника та адресу у формі Google Maps;
- Контакти – кнопка, при натисканні якої з’являється форма з контактними телефонами розробника та посиланнями на його соціальні мережі;
- Особистий кабінет - кнопка, при натисканні якої з’являється одна з наступних форм:

- якщо користувач не авторизувався – форма з нагадуванням про потребу авторизації;
- якщо авторизований користувач – клієнт – форма з додатковою інформацією про клієнта та можливість її редагування;
- якщо авторизований користувач – адміністратор – форма з можливістю редагування, додавання та видалення товарів.

4) На головній сторінці користувачу надається інформація про товари, а саме фото товару, назва та ціна. Також при натисканні на фото товару з'являється додаткова форма з докладним описом товару.
5) Після авторизації користувачеві відкриваються нові об'єкти:

- текст з привітання та зареєстрованим іменем;
- кнопка Баланс та актуальний баланс клієнта. При натисканні відкривається форма для поповнення балансу;
- кнопка Вихід, при натисканні якої користувач виходить зі свого облікового запису.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Сайт повинен відкриватись на будь-яких комп'ютерах, які підключені до інтернету та мають браузер для перегляду веб-сторінок.

Інтернет-магазин має відповідати певним вимогам щодо своїх функціональних характеристик, щоб забезпечити максимальний комфорт та зручність для користувачів.

Основними функціональними характеристиками інтернет-магазину є:

- зручний та інтуїтивно зрозумілий інтерфейс, що дає можливість легко знаходити необхідні товари та виконувати інші операції в магазині;
- наявність інформації про кожен товар, включаючи назву опис, фотографію та ціну;

- можливість оформлення замовлення в режимі онлайн з доставкою до поштової адреси;
- зручний та безпечний спосіб оплати товарів та послуг.

Ці функціональні характеристики є основою успішної роботи інтернет-магазину та забезпечують високу якість обслуговування та задоволеність клієнтів.

1.5.2. Вимоги до інформаційної безпеки

Для здійснення покупок на сайті користувач повинен зареєструватись та авторизуватися, з використанням свого логіну та пароля(зашифрованого через відповідну строку), які зберігаються у базі даних для захисту від шахрайства. Крім того, на сайті існують дві ролі - користувач та адміністратор, кожен з яких має доступ до своєї відповідної панелі. Адміністратору відкривається можливість додавати, редагувати та видаляти товари.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для зручного використання веб-додатка технічний засіб користувача повинен відповідати таким технічним вимогам:

- мати операційну систему Windows 7-11, Linux або MacOS;
- мати не менше 2 ГБ оперативної пам'яті;
- мати не менше 5 ГБ вільного місця на жорсткому диску з системою.

1.5.4. Вимоги до інформаційної та програмної сумісності

На сайті користувач може зареєструватись і обрати будь-який товар, спосіб оплати та вказати адресу доставки та номер телефону. Сайт розроблено на мові JavaScript з використанням бібліотеки ReactJs, яка є дуже популярною на сьогодні. При завантаженні сторінки не повинно бути проблем, але користувач

повинен мати достатньо нову версію браузера, щоб забезпечити коректну роботу змін на сторінці. Сайт підтримує наступні версії браузерів:

- Windows: 7 версія та вище;
- Chrome: 113 версія та вище;
- Opera: 98 версія та вище;
- Edge: 112 версія та вище;
- Safari: 14.1 версія та вище;
- IE: 11 версія та вище;
- Android: 9 версія та вище;
- Firefox: 106 версія та вище;
- Opera Mobile: 54.3 версія та вище.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Метою даної кваліфікаційної роботи є створення інтернет-магазину, спрямованого на продаж настільних ігор та головоломок на мові програмування JavaScript з використанням бібліотеки ReactJS[1]. Основний функціонал додатка полягає в наданні клієнтам можливості знайти, обрати та придбати товари, використовуючи систему фільтрації.

Додаток повинен мати два рівні доступу до функціоналу:

1) Клієнт: на цьому рівні користувачі зможуть реєструватись та входити в систему, переглядати список товарів та отримувати інформацію про конкретний товар.
2) Адміністратор: адміністратори матимуть можливість додавати або видаляти товари з обраного списку, переглядати список обраних товарів та редагувати інформацію про товари.

Крім основного функціоналу, важливими складовими програми є такі додаткові можливості:

- можливість доповнювати та редагувати базу даних: ця функція дозволяє адміністраторам магазину додавати нові товари, виправляти інформацію про існуючі товари та здійснювати інші редагування бази даних;
- налаштування корзини: користувачам надається можливість додавати обрані товари до корзини або видаляти їх. Крім того, можуть бути реалізовані функції розрахунку загальної суми та оформлення покупки;
- налаштування авторизації на сайті: для забезпечення безпеки та персоналізації веб-додатка може бути реалізована система авторизації користувачів. Це дозволить клієнту створити обліковий запис, увійти у свій профіль та керувати інформацією, потрібною для покупки.

Усі ці функції додаткового функціоналу допомагають покращити роботу програми, забезпечують зручність користування та полегшують процес покупок в інтернет-магазині настільних ігор та головоломок.

2.2. Опис застосованих математичних методів

Під час проектування та розробки цього веб-додатка не використовувалися складні математичні методи або алгоритми. Розрахунки та обчислення, які використовувалися, обмежувалися простими арифметичними діями.

2.3. Опис використаної архітектури та шаблонів проектування

При проектуванні та розробці даного проекту використано архітектурний шаблон MVC (Model-View-Controller) та його варіацію, відому як MVVM (Model-View-ViewModel). У цьому шаблоні компонент ‘App’ виступає в ролі контролера, який керує взаємодією між моделлю (state) і представленням (відображенням).

MVC (Model-View-Controller) - це архітектурний шаблон проектування, який допомагає розділити компоненти додатку на три основні ролі: Модель (Model), Представлення (View) та Контролер (Controller). Кожна з цих ролей відповідає за свої відповідності в програмі та має чіткі обов'язки. Основна ідея MVC полягає в розділенні логіки програми, відображення даних та керування взаємодією користувача з додатком (рис. 2.1).

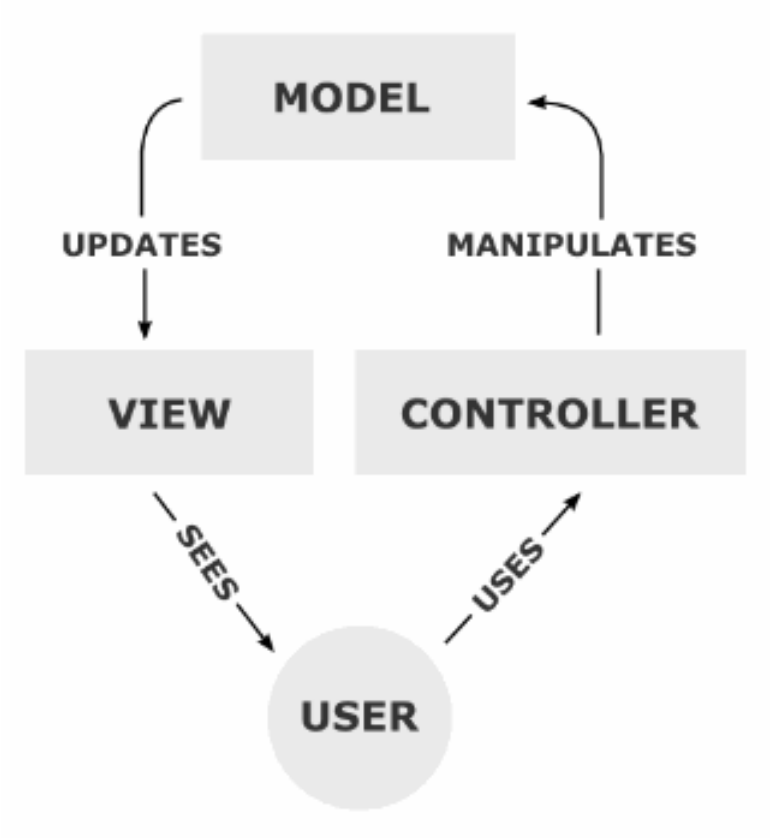

Рис. 2.1. Графічне представлення шаблону MVC

Основні компоненти MVC:

1) Модель (Model): модель представляє дані і бізнес-логіку додатку. Вона відповідає за збереження та обробку даних, виконання операцій з даними і відправку сповіщень (подій) про зміни даних. Модель не залежить від інших компонентів і може існувати незалежно від них.
2) Представлення (View): представлення відповідає за візуалізацію даних та інтерфейс користувача. Воно отримує дані від моделі та відображає їх у відповідному форматі, щоб користувач міг взаємодіяти з додатком. Представлення також може відправляти події (наприклад, натискання кнопок) контролеру для обробки.
3) Контролер (Controller): контролер відповідає за обробку подій, взаємодію з користувачем та оновлення моделі і представлення. Він слухає події від представлення, виконує відповідні дії та змінює стан моделі або представлення. Контролер також може відправляти команди моделі для виконання розрахунків або оновлення даних.

В контексті React, компоненти можна розглядати як комбінацію представлення та контролера, оскільки вони відповідають як за відображення даних, так і за обробку подій. Реакт-компоненти (наприклад, Header, Items, Categories) можуть відображати дані з моделі (state) і сприймати взаємодію користувача для подальшої обробки контролером (App).

Переваги MVC:

1) Розділення відповідальностей: модульність шаблону MVC дозволяє розділити відповідальності між компонентами, що полегшує розробку та підтримку коду.
2) Покращена повторне використання: шаблон MVC сприяє повторному використанню компонентів, оскільки модель та представлення можуть бути пере використані в різних контекстах.
3) Тестованість: Компоненти Моделі, Представлення та Контролера можуть бути тестовані окремо, що сприяє полегшенню процесу тестування додатку.

Мінуси MVC:

1) Потенційна складність: реалізація MVC може вимагати додаткового зусилля та зростання кількості коду порівняно з менш складними шаблонами проектування.
2) Надмірна залежність: невірне розділення відповідальностей або злиття ролей може призвести до надмірної залежності між компонентами, ускладнюючи розробку та підтримку.

Загалом, MVC є потужним шаблоном проектування, який допомагає впоратися зі складними додатками та забезпечити чітку розділення логіки, представлення і взаємодії з користувачем. Проте варто враховувати контекст використання та особливості конкретної технології чи фреймворку, так як реалізація MBC може розрізнятися в залежності від конкретного середовища розробки.

У даному проекті також використовується шаблон "Контейнер-представлення" (Container-Presentation). Компонент App виступає в ролі

контейнера, який містить логіку додатку, обробку подій та керування станом. Він також передає дані і зворотні виклики до дочірніх компонентів, таких як Header, Items, Categories та інші, які виступають у ролі представлення.

Шаблон "Контейнер-представлення" (Container-View) є розширенням шаблону проектування MBC (Model-View-Controller) і широко використовується в реактивних фреймворках, зокрема в React. Він допомагає зберегти компоненти React чистими та керованими та полегшує управління станом додатку.

У шаблоні "Контейнер-представлення" використовуються два типи компонентів:

1) Контейнери (Containers або Smart Components): контейнери відповідають за керування станом та логікою додатку. Вони отримують дані з моделі (або зовнішніх джерел даних) і передають їх до представлення через пропси (props). Контейнери також відповідають за обробку подій та взаємодію з іншими компонентами.
2) Представлення (Views або Dumb Components): представлення відповідають за візуалізацію даних та відображення їх користувачу. Вони отримують дані через пропси від контейнерів і рендерять їх на екрані. Представлення не мають власного стану і не виконують складної логіки.

Основна ідея шаблону "Контейнер-представлення" полягає в розділенні відповідальностей між контейнерами та представленнями. Контейнери беруть на себе завдання керування станом додатку, виконання дій та взаємодії зі зовнішнім середовищем. Представлення ж відповідають тільки за візуалізацію даних і зазвичай мають просту структуру.

Переваги шаблону "Контейнер-представлення":

1) Розділення відповідальностей: шаблон дозволяє чітко розділити логіку додатку від візуалізації. Контейнери відповідають за стан і логіку, тоді як представлення лише відображають дані.
2) Покращена керованість: контейнери дозволяють керувати станом додатку та управляти взаємодією з іншими компонентами. Це полегшує розробку та розуміння додатку.

3) Повторне використання та тестованість: представлення можна повторно використовувати, оскільки вони не мають залежності від стану додатку. Крім того, компоненти можуть бути легко тестовані окремо.

Мінуси шаблону "Контейнер-представлення":

1) Збільшена кількість компонентів: використання шаблону "Контейнер-представлення" може призвести до збільшення кількості компонентів в додатку, що може стати проблемою для простих або менш складних проектів.
2) Додатковий шар абстракції: шаблон вводить додатковий шар абстракції, який потребує розуміння його концепцій та правильної організації компонентів.

Загалом, шаблон "Контейнер-представлення" є корисним інструментом для керування станом та розподілу відповідальностей у реактивних додатках. Використовуючи цей шаблон, розробники можуть створювати чистий, модульний та добре керований код.

За допомогою функції ‘fetch’ здійснюється взаємодія з сервером для отримання даних. Запити на сервер виконуються для отримання даних про користувача (fetch('http://localhost/getuser.php?id=${savedUserId}')) та відправлення даних з форм (fetch('http://localhost/additem.php', ...)).

‘fetch’ - це вбудована функція в браузерах, яка дозволяє здійснювати асинхронні мережеві запити і отримувати відповіді з сервера. Вона надає простий та потужний спосіб взаємодії з API або віддаленими ресурсами.

Основні риси та використання fetch:

1) Асинхронність: fetch виконується асинхронно, що означає, що вона не блокує виконання інших частин коду під час виконання запиту і очікування відповіді сервера.
2) Синтаксис та ланцюжковість: fetch використовує проміси для обробки результату запиту. Це дозволяє виконувати ланцюжки методів, такі як .then() та .catch(), для обробки успішної відповіді або помилки.

3) Параметри запиту: за допомогою fetch можна налаштовувати різні параметри запиту, такі як метод (GET, POST, PUT, DELETE), заголовки (headers), дані в тілі запиту (body), кешування та інші.
4) Обробка відповіді: fetch повертає об'єкт Response, який містить інформацію про відповідь сервера. Ви можете використовувати методи, такі як .json(), .text(), .blob(), для отримання даних з відповіді у зручному форматі.

Приклад використання fetch для виконання GET-запиту і отримання даних в форматі JSON приведено на рисунку 2.2:

```
fetch("http://localhost/getitems.php")
  .then((response) => response.json())
  .then((data) => {
    const items = data.map((item) => ({
      id: item.id,
      title: item.title,
      description: item.description,
      price: item.price,
      img: item.img,
      category: item.category,
    }));
    this.setState({ allitems: items, currentItems: items, isLoading: false });
  })
  .catch((error) => console.log(error));
```

Рис. 2.2. Приклад використання ‘fetch’

У цьому прикладі ми виконуємо GET-запит на http://localhost/getitems.php, отримуємо відповідь сервера, перетворюємо її в форматі JSON і обробляємо отримані дані в функції .

2.4. Опис використаних технологій та мов програмування

Технології та мови програмування, які були використані при створенні даного проекту:

- HTML (HyperText Markup Language);
- CSS (Cascading Style Sheets);
- JavaScript[7];

- React;
- PHP[13];
- MySQL[18].

HTML (HyperText Markup Language) є мовою розмітки, яка використовується для створення структури та вмісту веб-сторінок. HTML визначає, які елементи і як вони повинні бути відображені на веб-сторінці, а також надає їм семантичне значення.

Основними концепціями HTML є елементи (tags), атрибути (attributes) і текстовий вміст (text content). Елементи використовуються для створення різних елементів на сторінці, таких як заголовки, абзаци, списки, таблиці, зображення і багато іншого. Кожен елемент має свою семантику і виконує певну функцію на сторінці.

Атрибути використовуються для надання додаткових властивостей елементам. Наприклад, атрибут src використовується для вказівки джерела зображення, атрибут href - для посилань. Атрибути допомагають змінювати поведінку або вигляд елементів.

Текстовий вміст використовується для вставки тексту всередину елементів. Він може бути простим текстом або форматованим з використанням різних тегів для надання стилів, розмітки або структури.

HTML є стандартом, який розробляється і підтримується Консорціумом Всесвітньої павутини (W3C). Існує кілька версій HTML, і на сьогоднішній день найпоширенішою є HTML5. HTML5 включає багато нових функцій і покращень, які дозволяють створювати більш динамічні та інтерактивні веб-сторінки.

Окрім основної структури сторінки, HTML також підтримує форми, які дозволяють користувачам взаємодіяти зі сторінкою, вводити дані, відправляти їх на сервер та отримувати відповідь.

CSS (Cascading Style Sheets) є мовою стилів, яка використовується для визначення вигляду і форматування веб-сторінок, написаних на мові розмітки, такій як HTML. CSS дозволяє змінювати кольори, шрифти, розміри, розташування елементів і багато іншого[24-28].

Основна ідея CSS полягає в тому, що вона розділяє структуру сторінки (визначену HTML) від її вигляду і стилів. За допомогою CSS можна створювати стильові правила, які визначають, як конкретні елементи повинні бути відображені на сторінці.

CSS використовує селектори для вибору елементів, до яких будуть застосовуватись стилі. Селектори можуть бути базовими (наприклад, назви тегів або класи елементів) або більш складними за допомогою комбінування селекторів (наприклад, елементи з певним атрибутом або вкладені елементи). Після вибору елементів за допомогою селекторів, до них застосовуються властивості стилів.

Властивості стилів визначають конкретні атрибути елементів, які треба змінити. Вони мають значення, яке вказує, як елемент повинен бути відображений. Наприклад, властивість color визначає колір тексту, font-size - розмір шрифту, margin - відступи, і так далі.

CSS також підтримує різні способи вкладення стилів, включаючи внутрішні стилі, зовнішні файли CSS і навіть встроєні стилі безпосередньо в HTML-коді. Зовнішні файли CSS рекомендується використовувати для розділення стилів від HTML-коду, що полегшує підтримку і редагування стилів.

CSS є потужним інструментом для веб-дизайну, оскільки вона надає широкі можливості для створення різноманітних макетів, стилів і ефектів. За допомогою CSS можна створювати адаптивний дизайн, анімацію, градієнти, тіні, переходи, розташування елементів на сторінці і багато іншого.

CSS є стандартом, розробленим і підтримуваним Консорціумом Всесвітньої павутини (W3C). На сьогоднішній день найпоширенішою версією є CSS3, яка включає багато нових функцій і можливостей.

JavaScript є високорівневою, інтерпретованою мовою програмування, що зазвичай використовується для розробки веб-додатків. Вона дозволяє взаємодіяти з користувачем, маніпулювати веб-сторінками і забезпечувати динамічність та інтерактивність веб-додатків[8-12].

Основні риси та можливості JavaScript:

1) Веб-браузерний скриптінг: JavaScript використовується для створення веб-додатків, які взаємодіють з користувачем та маніпулюють вмістом веб-сторінок. Вона може динамічно змінювати елементи сторінки, реагувати на події (натискання кнопок, введення тексту) і виконувати асинхронні запити на сервер.
2) Кросс-платформеність: JavaScript підтримується всіма сучасними веб-браузерами і може виконуватись на різних платформах, включаючи Windows, macOS та Linux.
3) Об'єктно-орієнтований підхід: JavaScript базується на об'єктно-орієнтованому програмуванні, що дозволяє створювати об'єкти з властивостями і методами. Це дозволяє розбивати код на модулі і повторно використовувати код.
4) Функціональне програмування: JavaScript також підтримує функціональне програмування, де функції є об'єктами першого класу і можуть бути передані як аргументи, збережені в змінних і повернуті з функцій.
5) Асинхронність: JavaScript має підтримку асинхронного програмування, що дозволяє виконувати операції без блокування потоку виконання. Це особливо корисно для виконання мережевих запитів та обробки відповідей з сервера без затримки виконання інших частин програми.
6) Багатофункціональність: JavaScript може бути використаний для розробки різноманітних додатків, включаючи веб-додатки, мобільні додатки, настільні додатки, ігри та навіть серверну частину за допомогою платформи Node.js.
7) Велика екосистема: JavaScript має велику спільноту розробників і багато сторонніх бібліотек і фреймворків, таких як React, Angular, Vue.js, для спрощення розробки веб-додатків і покращення продуктивності.

JavaScript використовується для створення динамічних веб-сторінок, валідації форм, реалізації ефектів анімації, маніпулювання DOM, взаємодії з веб-сервером за допомогою AJAX і багато іншого. Це мова, яку багато розробників

вивчають і використовують для створення потужних інтерактивних веб-додатків.

React є відкритою бібліотекою JavaScript для розробки інтерфейсів користувача. Вона створена компанією Facebook і широко використовується для побудови ефективних та масштабованих веб-додатків[2-6].

Основні риси та можливості React:

1) Компонентна структура: React розбиває веб-інтерфейс на незалежні компоненти, які можуть бути повторно використані і забезпечують легку організацію коду. Компоненти можуть мати свій стан (state) і властивості (props), що дозволяє створювати динамічний контент.
2) Віртуальний DOM: React використовує внутрішню структуру даних, відому як віртуальний DOM. Він є легковагим представленням реального DOM і дозволяє React ефективно виконувати оновлення і перерисування компонентів. React порівнює стару та нову версії віртуального DOM, знаходить різницю і застосовує тільки необхідні зміни до реального DOM.
3) Односторінкові додатки: React дозволяє створювати односторінкові додатки, де весь контент завантажується один раз, а потім динамічно оновлюється без перезавантаження сторінки. Це забезпечує швидку та зручну взаємодію з додатком.
4) JSX: React використовує JSX (JavaScript XML) для опису компонентів і їх структури. JSX поєднує JavaScript і HTML-подібний синтаксис, що дозволяє легко описувати компоненти та їх взаємодію з даними.
5) Універсальність: React може працювати як на клієнтській стороні, так і на серверній стороні. На клієнтській стороні React може бути використаний для створення веб-додатків, тоді як на серверній стороні він може генерувати HTML на основі компонентів для передачі на клієнт.
6) Розширюваність: React можна поєднувати з іншими бібліотеками та фреймворками, такими як Redux, для керування станом додатків, або React Router, для реалізації маршрутизації в односторінкових додатках.

7) Розробка на основі компонентів: React підтримує підхід розробки на основі компонентів, де кожен компонент відповідає за свою функціональність і може бути незалежно розроблений, тестований та підтримуваний.

React є дуже популярним веб-фреймворком і використовується для розробки великої кількості веб-додатків. Він надає зручні інструменти для побудови модульних, швидких і ефективних інтерфейсів користувача.

PHP (Hypertext Preprocessor) - це мова програмування загального призначення, яка широко використовується для розробки веб-додатків і скриптів. Вона зазвичай виконується на стороні сервера, і її скрипти виконуються на веб-сервері, що генерує HTML-сторінки, які відправляються до веб-браузеру користувача[14-17].

Особливості та можливості PHP:

1) Веб-розробка: PHP спеціально розроблена для роботи з веб-серверами і створення динамічних веб-сторінок. Вона має вбудовану підтримку для взаємодії з серверами баз даних, обробки форм, керування сеансами користувачів та іншими веб-функціями.
2) Простота вивчення: PHP має простий і легкий для вивчення синтаксис, який нагадує синтаксис мови C. Це робить його доступним для початківців і швидко засвоюється розробниками з досвідом інших мов програмування.
3) Розширюваність: PHP має велику кількість вбудованих функцій і бібліотек, які спрощують розробку веб-додатків. Крім того, PHP дозволяє розробникам створювати власні функції та розширення, що значно розширює можливості мови.
4) Підтримка різноманітних баз даних: PHP має вбудовану підтримку для багатьох популярних баз даних, таких як MySQL, PostgreSQL, Oracle і інших. Це дозволяє легко працювати з даними і здійснювати операції збереження, отримання та оновлення інформації у базі даних.

5) Масштабованість: PHP підтримує різні архітектурні шаблони, такі як Model-View-Controller (MVC), що сприяє розподіленню відповідальностей і полегшує підтримку та розширення додатків.
6) Велике співтовариство: PHP має велику та активну спільноту розробників, яка надає підтримку, документацію, бібліотеки і фреймворки. Це забезпечує швидкий доступ до рішень проблем, оновлення і нових можливостей.

PHP широко використовується для створення різноманітних веб-додатків, від невеликих сайтів і блогів до складних корпоративних систем. Вона є однією з найпопулярніших мов програмування для веб-розробки і продовжує активно розвиватись та вдосконалюватись.

MySQL є однією з найпопулярніших відкритих систем управління базами даних (СУБД). Вона використовується для зберігання та управління структурованою інформацією в базах даних[19-23].

Основні особливості та можливості MySQL:

1) надійність: MySQL є дуже надійною СУБД з високим рівнем стабільності. Вона може обробляти великі обсяги даних і має механізми відновлення в разі відмови.
2) Простота використання: MySQL має простий і зрозумілий SQL-синтаксис, який дозволяє легко створювати, змінювати та запитувати дані в базі даних. Вона також має добре документовану інструкцію та широку спільноту розробників, що сприяє її використанню.
3) Масштабованість: MySQL може масштабуватись від невеликих проектів до великих підприємств. Вона підтримує розподілені системи баз даних, реплікацію даних та кластеризацію для підвищення продуктивності та доступності.
4) Підтримка стандартів: MySQL дотримується багатьох стандартів SQL, що дозволяє переносити бази даних між різними СУБД і легко інтегруватись з іншими додатками та інструментами.

5) Багатофункціональність: MySQL підтримує широкий спектр функцій, таких як транзакції, відкати, вирази згортки (aggregates), підзапити, індексацію, процедури, функції, тригери та інше. Вона також підтримує різні типи даних, включаючи числа, рядки, дати, зображення та бінарні дані.
6) Безпека: MySQL надає механізми захисту даних, включаючи автентифікацію, авторизацію, шифрування та захист від вторгнень. Вона також підтримує SSL для безпечного обміну даними між клієнтами і сервером.

MySQL може використовуватись в поєднанні з різними мовами програмування та платформами розробки. Її можна легко інтегрувати з веб-додатками, серверними програмами, мобільними додатками та іншими системами для забезпечення зберігання та обробки даних.

Взагалі, MySQL є потужним та надійним інструментом для роботи з базами даних, який широко використовується в індустрії для забезпечення зберігання та управління даними.

2.5. Опис структури програми та алгоритмів її функціонування

При створенні структури сайту було використано шаблон односторінкового додатка (Single-Page Application, SPA).

SPA є архітектурним підходом, при якому увесь контент та логіка розміщуються на одній сторінці, яка завантажується лише один раз при початковому завантаженні. Він використовує JavaScript для динамічного взаємодії з користувачем, без необхідності перезавантаження сторінки.

Структура цього проекту має такий вигляд::

1) index.html – це головний HTML-файл, який містить загальну структуру сторінки та основний контент.
2) App.js – це JavaScript-файл, який відповідає за керування логікою і взаємодією на сторінці. Він містить функції та події, які обробляють

натискання на кнопки та відкривають відповідні форми або елементи на сторінці.

3) index.css – це файл стилів, який містить CSS-стилі для оформлення сторінки.
4) Форми: кожна кнопка на головній сторінці відкриває відповідну форму або елемент на сторінці. Форми можуть бути реалізовані як вбудовані елементи на сторінці або включати в себе модальне вікно, яке з'являється поверх основного контенту.

Ця структура дозволяє створювати взаємодію з користувачем без перезавантаження сторінки. Користувачі можуть взаємодіяти з формами, заповнювати дані та відправляти їх, а результати можуть оброблятись за допомогою JavaScript.

Головний файл index.html розташований у папці public проекту та відповідає за завантаження на сторінку div з id=’root’ та встановлення деяких елементів, таких як назва сайту на вкладці браузера, підключення скриптів та інші (рис. 2.3).

```
public > <> index.html > ...
1   <!DOCTYPE html>
2   <html lang="en">
3     <head>
4       <meta charset="utf-8" />
5       <meta name="viewport" content="width=device-width, initial-scale=1" />
6       <link rel="shortcut icon" href="img/icons8-brain-64.png" type="image/png">
7       <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
8       <title>МізкоГрай - магазин головоломок та настільних ігор</title>
9     </head>
10    <body>
11      <noscript>You need to enable JavaScript to run this app.</noscript>
12      <div id="root"></div>
13    </body>
14  </html>
```

Рис. 2.3. Вміст файлу index.html

Після цього завантажується файл index.js (з папки src), який підключає основні бібліотеки React та ReactDOM, а також файли index.css та App.js, далі створює компонент ‘root’, у якому обробляється увесь веб-додаток (рис. 2.4).

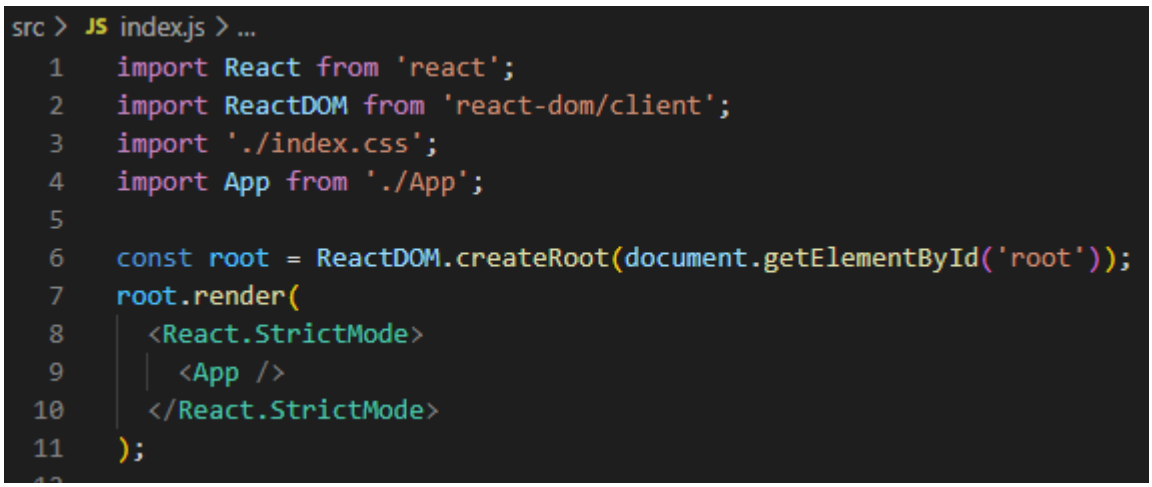

Рис. 2.4. Вміст файлу index.js

ReactDOM - це пакет, який надає методи для взаємодії з DOM (Document Object Model) у контексті React-додатків. Він дозволяє рендерити компоненти React у віртуальному DOM та вставляти їх у справжній DOM сторінки.

У випадку цього проекту застосовано ReactDOM.render(element, container[, callback]): Цей метод використовується для рендерингу компонентів React. Він приймає кореневий елемент (React-компонент або JSX-вираз) та контейнер (DOM-елемент), в який потрібно вставити рендеринг. Можна також вказати опціональний зворотний виклик (callback), який виконається після успішного рендерингу.

Файл index.css підключається для того щоб усі стилі елементів веб-сторінки могли бути налаштовані у цьому файлі.

Усі елементи веб-додатку завантажуються з компоненту App.js, у якого також є свої дочірні компоненти, такі як Header, User, Items та інші. Директорія з різними компонентами має такий вигляд (рис. 2.5):

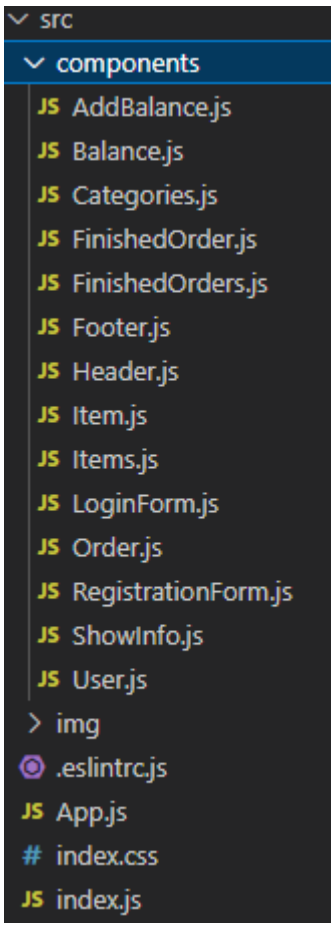

Рис. 2.5. Директорія компонентів веб-додатку

У цій директорії розташовані усі компоненти, які використовуються при завантаженні сторінки та будь-яких діях на ній.

За зв’язок з базою даних відповідає набір PHP-файлів, який представлено нижче (рис. 2.6):

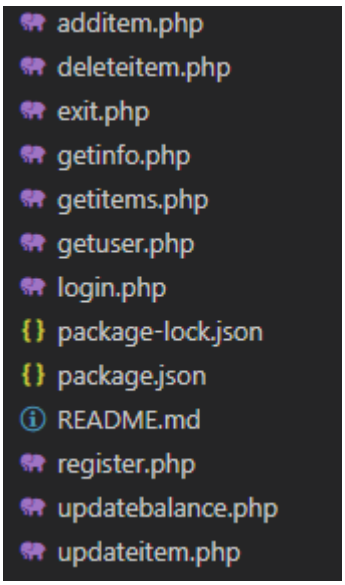

Рис. 2.6. Список PHP-файлів

Кожен з цих файлів відповідає за свою дію у зв’язку з базою даних. Наприклад, файл login.php, який відповідає за авторизацію користувача на сайті, має такий вигляд (рис. 2.7):

login.php

```php
<?php

    header("Access-Control-Allow-Origin: *");

    $login = filter_var(trim($_POST['login']), FILTER_SANITIZE_STRING);
    $pass = filter_var(trim($_POST['password']), FILTER_SANITIZE_STRING);

    $pass = md5($pass."sfdfsd145");

    $mysql = new mysqli("localhost", "root", "", "my-shop");

    $result = $mysql->query("SELECT * FROM users WHERE login='$login' AND password='$pass'");
    $user = $result->fetch_assoc();

    if(!$user){
        echo "Користувача з таким логіном та паролем не знайдено!";
        exit();
    }

    setcookie('user_id', $user['id'], time() + 3600, "/");

    $mysql->close();

    header('Location: http://localhost:3000');

?>
```

Рис. 2.7. Файл з авторизацією

Тут ми можемо бачити, як на мові PHP реалізовано вхід до облікового запису, дані про який містяться у базі даних у таблиці ‘users’. Користувач вводить свій логін та пароль до відповідних полей у формі реєстрації на сайті, натискає кнопку Підтвердження, файл на сервері обробляє введені дані та повертає відповідь у вигляді ‘id’ клієнта, або помилки, якщо такого не знайдено. Приблизно по такому ж алгоритму працюють й інші файли, які відповідають за з’єднання за базою даних.

Короткий опис інших компонентів проекту:

- img: директорія, звідки на сторінку завантажуються фото усіх товарів;
- AddBalance.js: форма для поповнення особистого балансу;
- Balance.js: елемент, що містить число з балансом;
- Categories.js: список категорій товарів;

- FinishedOrder.js: елемент, що виводить інформацію про куплений товар;
- FinishedOrders.js: компонент, який виводить усі куплені товари;
- Footer.js: попередження про авторські права;
- Header.js: один з найбільших компонентів, який містить: логотип, куток користувача (інформація про сайт, контакти розробника, особистий кабінет) та фонове зображення;
- Item.js: елемент, що виводить інформацію про товар;
- Items.js: компонент, який виводить усі товари;
- LoginForm.js: форма авторизації;
- Order,js: компонент, що виводить товар у кошику для покупок;
- RegistrationForm.js: форма реєстрації;
- ShowInfo.js: форма з докладною інформацією про товар;
- User.js: компонент, який містить кнопки реєстрації, авторизації для неавторизовагих користувачів, а для авторизованих: Привітання, Баланс та Вихід;
- additem.php: додавання товару до бази даних;
- deleteitem.php: видалення товару з бази даних;
- exit.php: вихід з облікового запису;
- getinfo.php: додавання або редагування додаткової інформації про користувача;
- getitems.php: отримання списку товарів з бази даних;
- getuser.php: отримання інформації про користувача;
- register.php: файл реєстрації;
- updatebalance.php: оновлення балансу в базі даних;
- updateitem.php: оновлення інформації про товар у базі даних.

2.6. Обгрунтування та організація вхідних та вихідних даних програми

У проекті даної кваліфікаційної роботи вхідними даними є:

- логін, пароль, ім’я, додаткова інформація про користувача, сума поповнення балансу (якщо користувач - клієнт);
- логін, пароль, інформація про новий товар та нові дані про старий (якщо користувач - адміністратор).

Вихідними даними є дані про товар або користувача, які зберігаються у базі даних та виводяться у зручному для користувача графічному вигляді при відправленні запиту.

Обмін даними між клієнтською та серверною частинами здійснюється за допомогою формату JSON. JSON є популярним форматом обміну даними, який дозволяє передавати структуровану інформацію у вигляді об'єктів та масивів. Він легко зрозумілий як для людей, так і для комп'ютерів, що сприяє ефективному обміну даними між клієнтом та сервером.

2.7. Опис роботи розробленого програмного продукту

Для початку роботи з сайтом, необхідно перейти за його посиланням. У моєму випадку проект запускається через локальний сервер Open Server Panel.

OpenServerPanel (також відомий як OpenServer Control Panel або просто OpenServer) - це програмне забезпечення, яке надає інтерфейс для управління локальним сервером на основі стеку LAMP (Linux, Apache, MySQL, PHP) або WAMP (Windows, Apache, MySQL, PHP). Він призначений для розробки та тестування веб-сайтів або веб-додатків на локальному комп'ютері.

Зовнішній вигляд сайту (рис. 2.8):

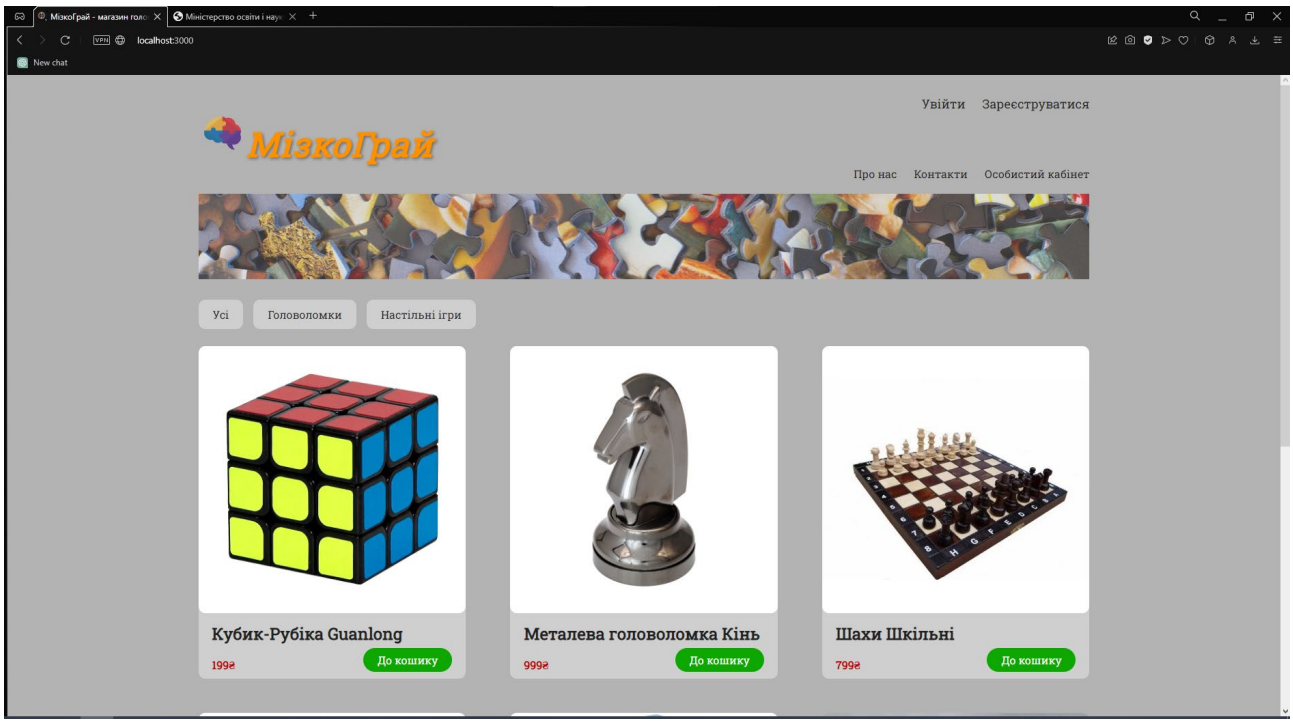

Рис. 2.8. Головна сторінка інтернет-магазину «Мізкограй»

2.7.1. Використані технічні засоби

Для забезпечення користувачу зручної роботи з сучасними браузерами, розробники рекомендують мати наступні мінімальні параметри комп'ютера:

1) Центральний процесор (ЦП): Pentium 4 або еквівалент.
2) Відеоадаптер: 3D адаптер від nVidia, Intel, AMD/ATI.
3) Відеопам'ять: мінімум 128 МБ відеопам'яті.
4) Накопичувач: щонайменше 150 ГБ вільного простору на жорсткому диску.
5) Оперативна пам'ять: мінімум 2 ГБ оперативної пам'яті.

Ці рекомендації допоможуть забезпечити плавну роботу браузера і відтворення веб-контенту на вашому комп'ютері. Звичайно, якщо ви плануєте використовувати важкі веб-додатки або мультимедійний контент, може знадобитися більш потужний комп'ютер з більшим обсягом пам'яті та швидшим процесором.

Технічні засоби, використані мною:

1) Центральний процесор (ЦП): Intel Core i3.

2) Відеоадаптер: 3D адаптер від nVidia.
3) Відеопам'ять: 3 ГБ.
4) Накопичувач: 1 ТБ.
5) Оперативна пам'ять: 16 ГБ.
6) Клавіатура, миша, монітор.

2.7.2. Використані програмні засоби

При розробці цього проекту було використано наступні програмні засоби:

- Visual Studio Code;
- Open Server Panel;
- PhpMyAdmin.

Visual Studio Code (VS Code) - це безкоштовний, легкий у користуванні текстовий редактор, розроблений компанією Microsoft. Він призначений для роботи з програмним кодом у різних мовах програмування та надає широкий спектр функцій, що полегшують розробку програмного забезпечення.

Основні особливості Visual Studio Code:

1) Підтримка різних мов програмування: VS Code надає підтримку для багатьох популярних мов програмування, таких як JavaScript, Python, C#, Java, HTML, CSS і багато інших. Завдяки цьому, розробники можуть працювати над проектами різного типу в одному редакторі.
2) Розширення та налаштування: VS Code дозволяє розширити його функціональність за допомогою розширень, які розробляються спільнотою. Розширення дозволяють додавати нові можливості, підтримку для конкретних технологій, інструменти для розробки та інше. Крім того, користувачі можуть налаштовувати редактор за своїми потребами.
3) Інтеграція з Git: VS Code має вбудовану підтримку системи контролю версій Git. Це дозволяє розробникам здійснювати коміти, переглядати та

порівнювати зміни, створювати та об'єднувати гілки, працювати з репозиторіями та багато іншого, все безпосередньо з редактора.

4) Розширені можливості пошуку та заміни: VS Code надає потужні інструменти для пошуку та заміни тексту в проекті. За допомогою регулярних виразів та різних параметрів пошуку, можна легко знаходити та замінювати текст у великих обсягах коду.
5) Відладка коду: редактор підтримує можливість відладки коду для різних мов програмування. За допомогою спеціальних розширень та налаштувань, розробники можуть ставити точки зупинки, переглядати значення змінних, виконувати кроки виконання коду та інше.
6) Live Share: функція, що дозволяє розробникам спільно працювати над проектом в реальному часі. За допомогою Live Share, ви можете запрошувати інших розробників до свого редактора, щоб вони могли бачити ваш код, робити зміни та спілкуватися.

Visual Studio Code є популярним вибором серед розробників завдяки своїй легкості використання, гнучкості та багатому екосистемі розширень. Він доступний на різних операційних системах, включаючи Windows, macOS та Linux (рис. 2.9).

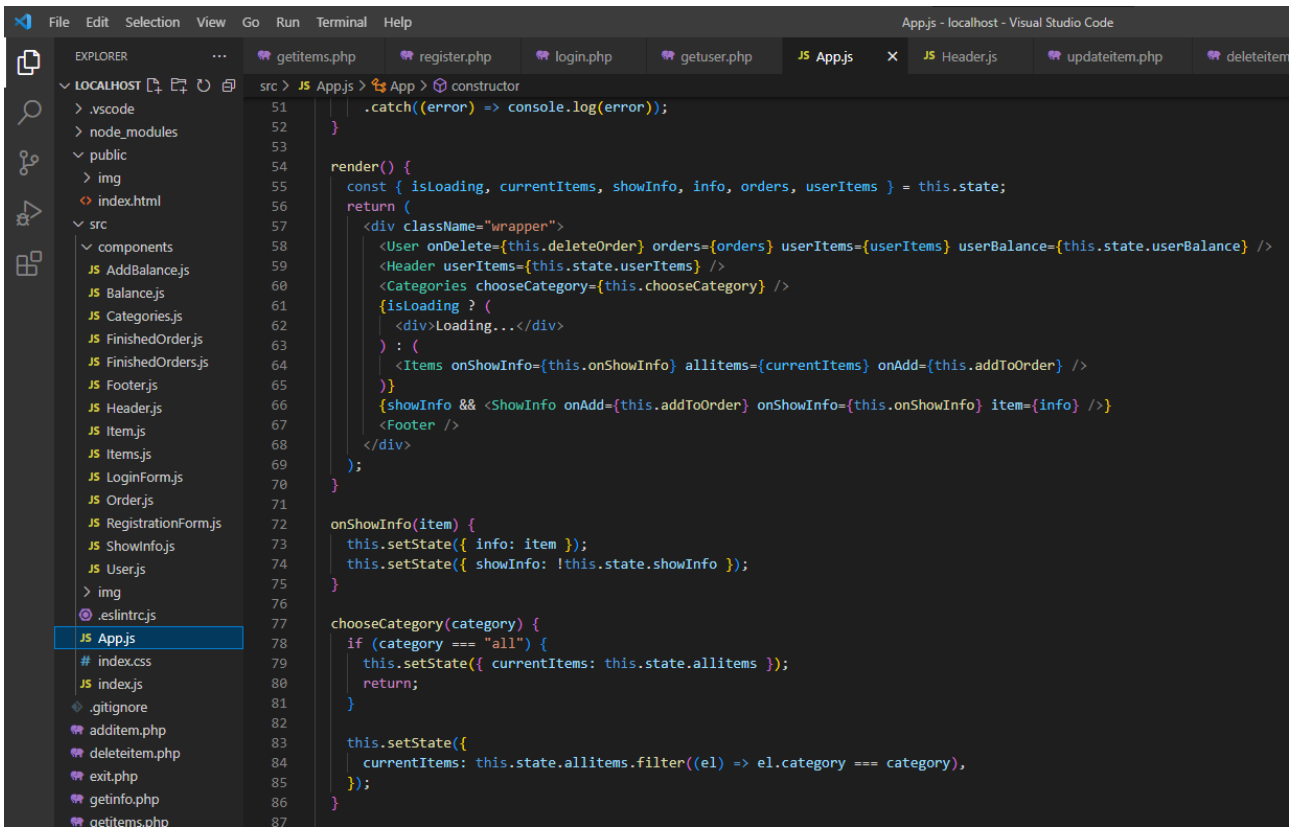

Рис. 2.9. Зовнішній вигляд VS Code

Open Server Panel надає зручний спосіб налаштування серверного середовища, включаючи веб-сервер Apache, базу даних MySQL, мову програмування PHP та інші складові веб-стеку. Він дозволяє встановлювати, конфігурувати та керувати різними версіями цих компонентів з легкістю.

Open Server Panel також має функції для керування віртуальними хостами, доменами, базами даних, налаштування PHP-параметрів, логування та моніторингу сервера. Він дозволяє зручно перемикатися між різними конфігураціями сервера для розробки різних проектів.

Open Server Panel є популярним інструментом серед веб-розробників, оскільки він спрощує налаштування серверного середовища та забезпечує зручні інструменти для розробки та тестування веб-додатків на локальному комп'ютері (рис. 2.10).

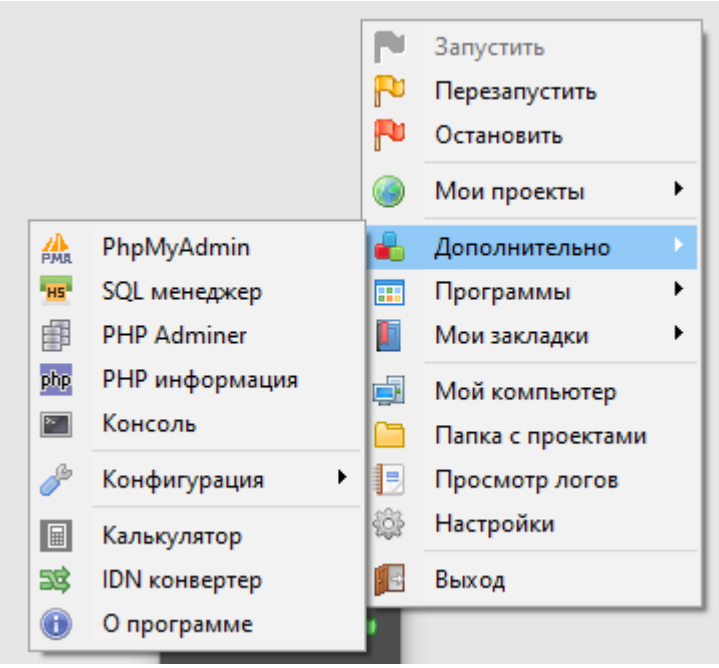

Рис. 2.10. Зовнішній вигляд OS Panel

PhpMyAdmin - це безкоштовний веб-інтерфейс для управління базами даних MySQL. Він надає зручні інструменти для роботи з базами даних, такі як створення, видалення та зміна таблиць, виконання SQL-запитів, імпорт та експорт даних, управління користувачами та дозволами, створення резервних копій та багато іншого.

PhpMyAdmin працює через веб-браузер і дозволяє зручно керувати базами даних без необхідності встановлювати додаткові клієнтські програми. Він має інтуїтивний інтерфейс, що робить його доступним для користувачів з будь-яким рівнем досвіду.

За допомогою PhpMyAdmin ви можете легко створювати нові таблиці та змінювати їх структуру, виконувати SQL-запити для вибірки, вставки, оновлення та видалення даних. Ви також можете експортувати дані у різних форматах, таких як SQL, CSV, XML, а також імпортувати дані з цих форматів.

PhpMyAdmin надає можливість керування користувачами і дозволами доступу до баз даних. Ви можете створювати нових користувачів, встановлювати їх права доступу та керувати рівнями безпеки.

Загалом, PhpMyAdmin є потужним інструментом для управління базами даних MySQL, який робить процес роботи з базами даних зручним та ефективним, незалежно від вашого рівня досвіду (рис. 2.11).

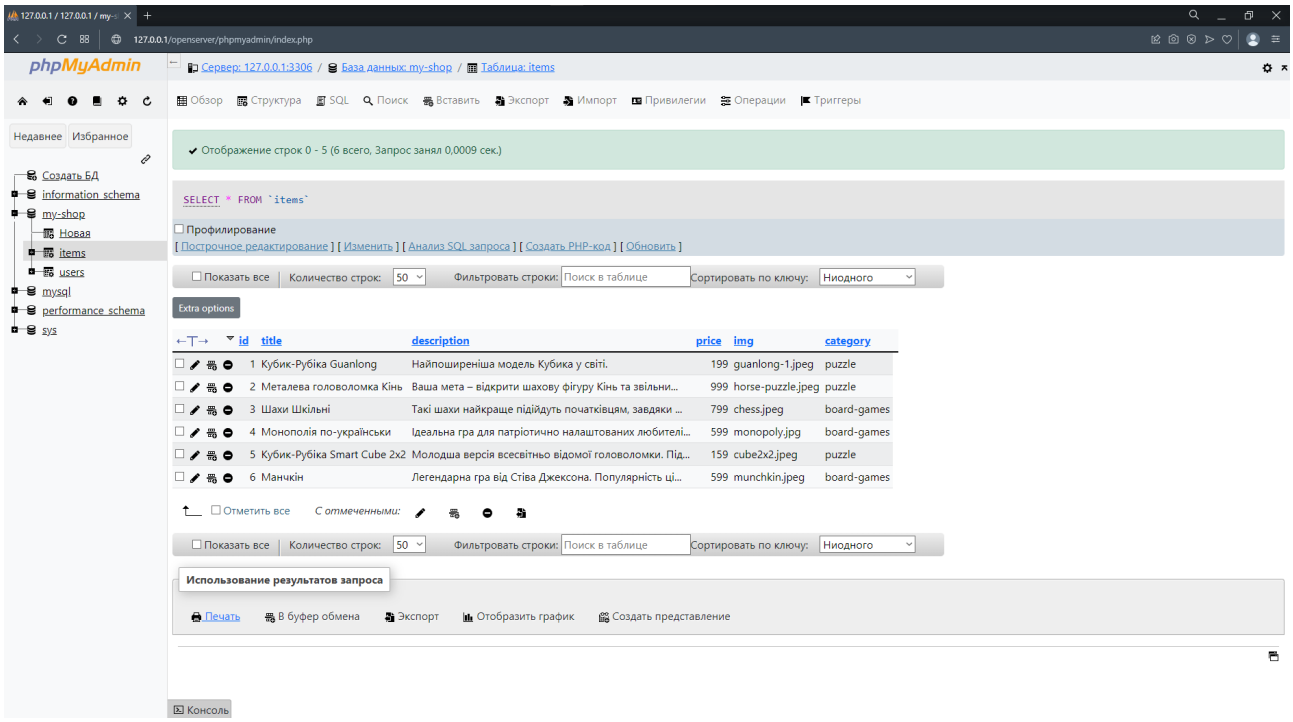

Рис. 2.11. Зовнішній вигляд PhpMyAdmin

2.7.3. Виклик та завантаження програми

Розроблений веб-застосунок може бути доступний в мережі Інтернет за допомогою стандартизованої адреси (URL: http://localhost:3000) через сучасні браузери, що підтримуються на різних операційних системах. Наприклад, для десктопних операційних систем, таких як Windows 10, Linux та macOS, підтримуються популярні браузери, такі як Google Chrome, Firefox та Brave. На мобільних платформах підтримуються браузери, такі як Google Chrome, Dolphin, Opera Mobile, Mozilla Firefox та Safari.

Додаток гарантовано запускається на останніх двох версіях усіх підтримуваних браузерів, які визначені офіційними розробниками. Зазвичай, додаток також може працювати на старіших версіях браузерів, але не надається гарантія успішної роботи на них.

Таким чином, залежно від операційної системи та пристрою, користувачі можуть отримати доступ до веб-застосунку через обраний браузер, забезпечуючи широку сумісність та доступність на різних платформах.

2.7.4. Опис інтерфейсу користувача

Після запуску веб-додатку перед користувачем відкривається головна сторінка інтернет-магазину. Так як структура сайту є односторінковою, на ній будуть з’являтися додаткові форми, поля та кнопки при натисканні на відповідні елементи (рис. 2.8).

Для початку роботи на сайті, необхідно зареєструватися у базі користувачів. За це відповідає кнопка у правому верхньому куті екрану з надписом: «Зареєструватися» (рис. 2.12).

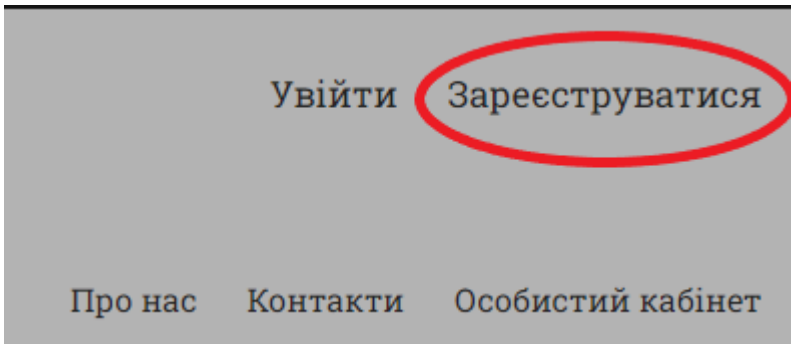

Рис. 2.12. Кнопка реєстрації

Після натискання цієї кнопки з’являється форма для реєстрації (рис. 2.13).

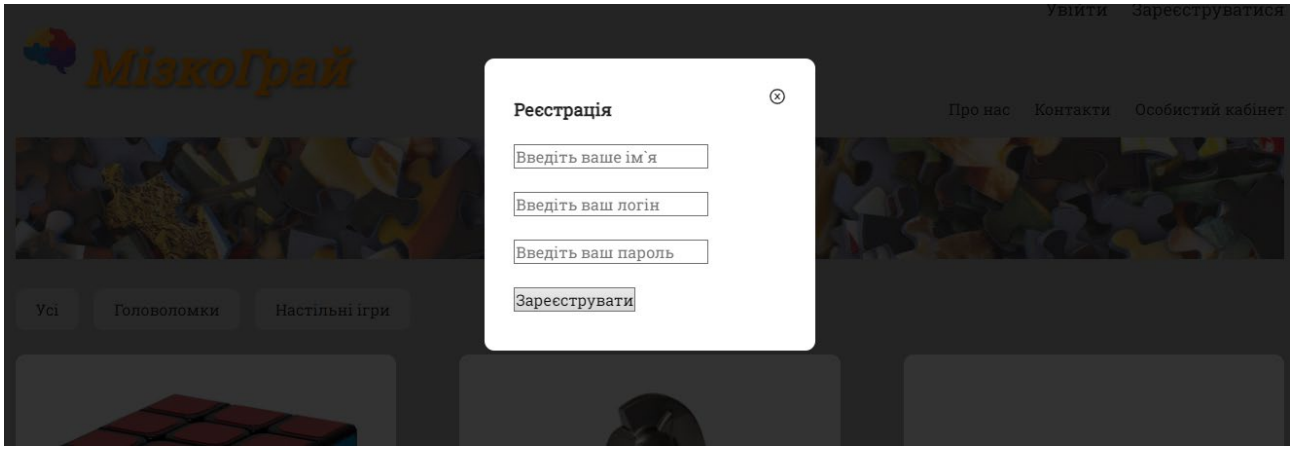

Рис. 2.13. Форма реєстрації

Якщо користувач успішно зареєструвався, його перенаправляє назад на головну сторінку, де наступним кроком слідує авторизація. За це відповідає кнопка у правому верхньому куті з надписом «Увійти» (рис. 2.14).

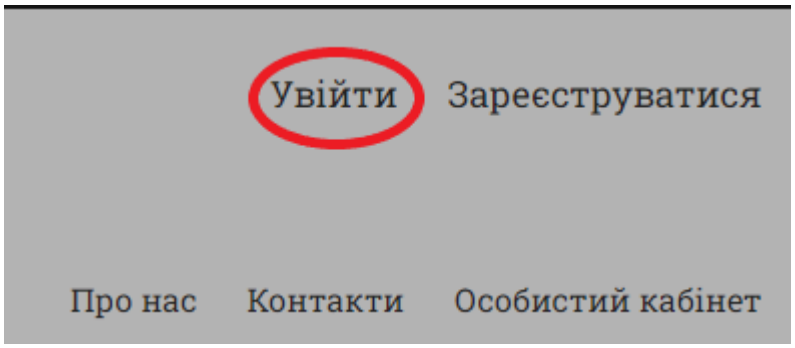

Рис. 2.14. Кнопка авторизації

Після натискання цієї кнопки з’являється форма для авторизації (рис. 2.15).

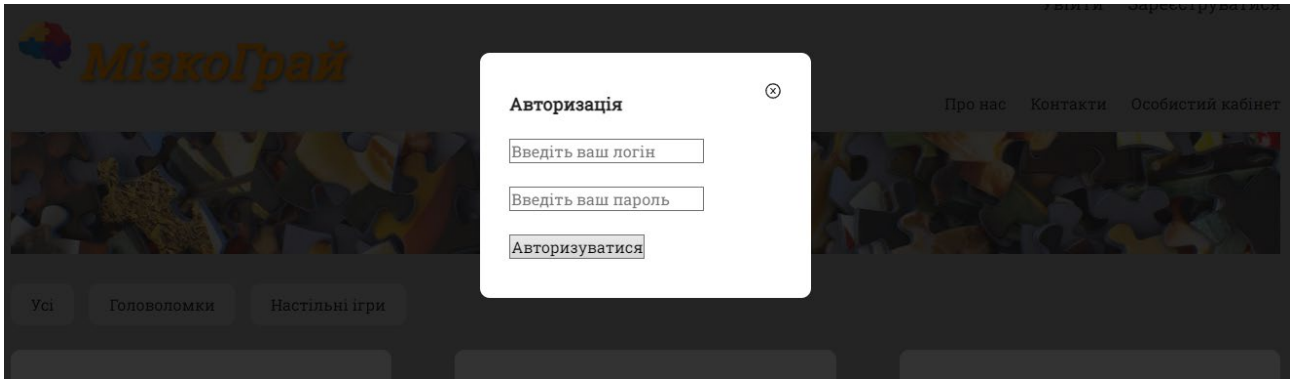

Рис. 2.15. Форма авторизації

Подальші можливості у функціоналі користувача залежать від його ролі вказаної у базі. Розглянемо варіанти:

1) Якщо авторизований користувач – адміністратор:

Для прикладу розглянемо інтерфейс, що відкривається адміністратору сайту (рис. 2.16).

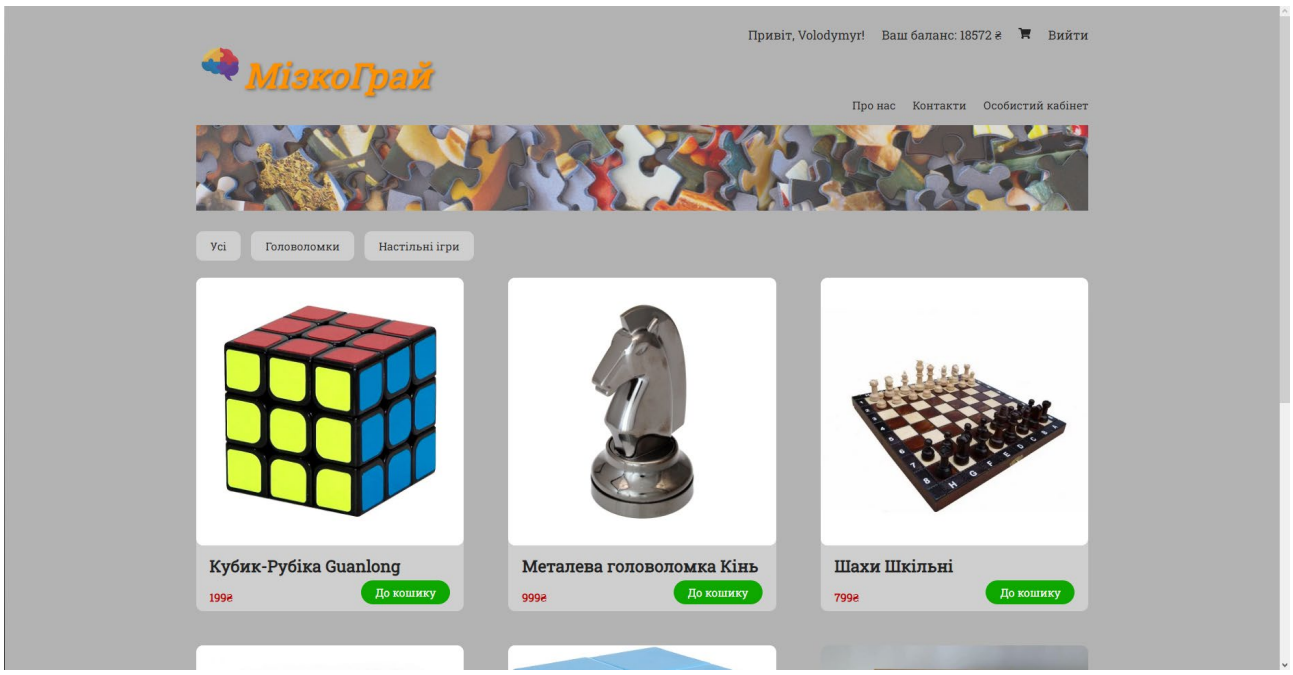

Рис. 2.16. Інтерфейс адміністратора

На рисунку ми бачимо наступні зміни: на місці кнопок «Увійти» та «Зареєструватися» з’явилися інші елементи – Привітання, кнопка «Баланс», значок «Кошик» та кнопка «Вийти».

У випадку адміністратора нас цікавить кнопка «Особистий кабінет», яка розташована трохи нижче з правого боку сторінки (рис. 2.17).

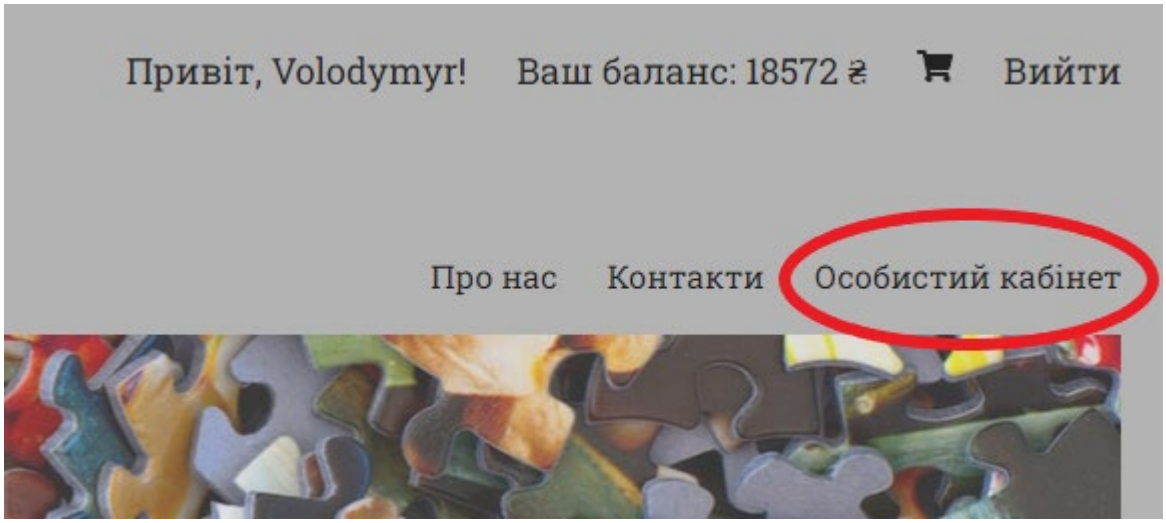

Рис. 2.17. Кнопка «Особистий кабінет»

При натисканні на дану кнопку з’являться форма з адміністративною панеллю (рис. 2.18).

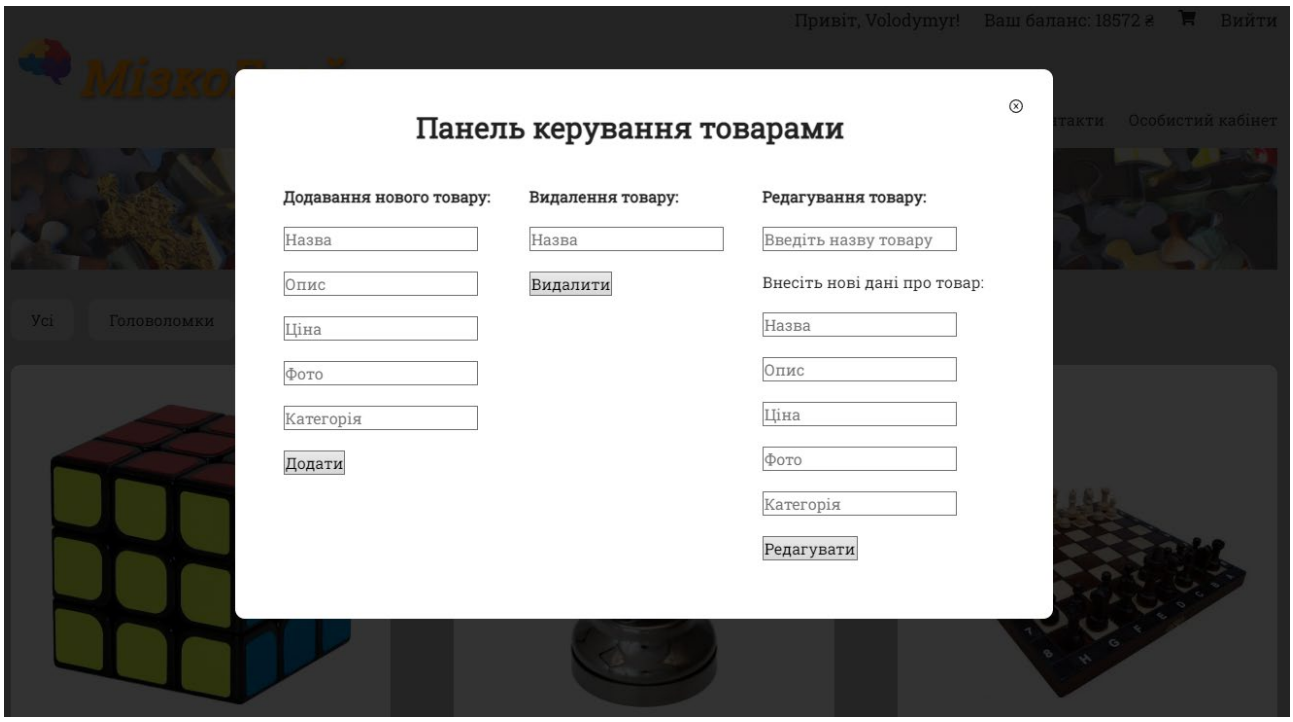

Рис. 2.18. Адміністративна панель

Тут адміністратор може керувати товарами: додавати нові, редагувати та видаляти існуючі.

На цьому функціонал, призначений лише для адміністратора, закінчується. Усі інші можливості доступні не тільки адміністраторам.

2) Якщо користувач – клієнт:

Для прикладу розглянемо інтерфейс, що відкривається клієнту (рис. 2.19).

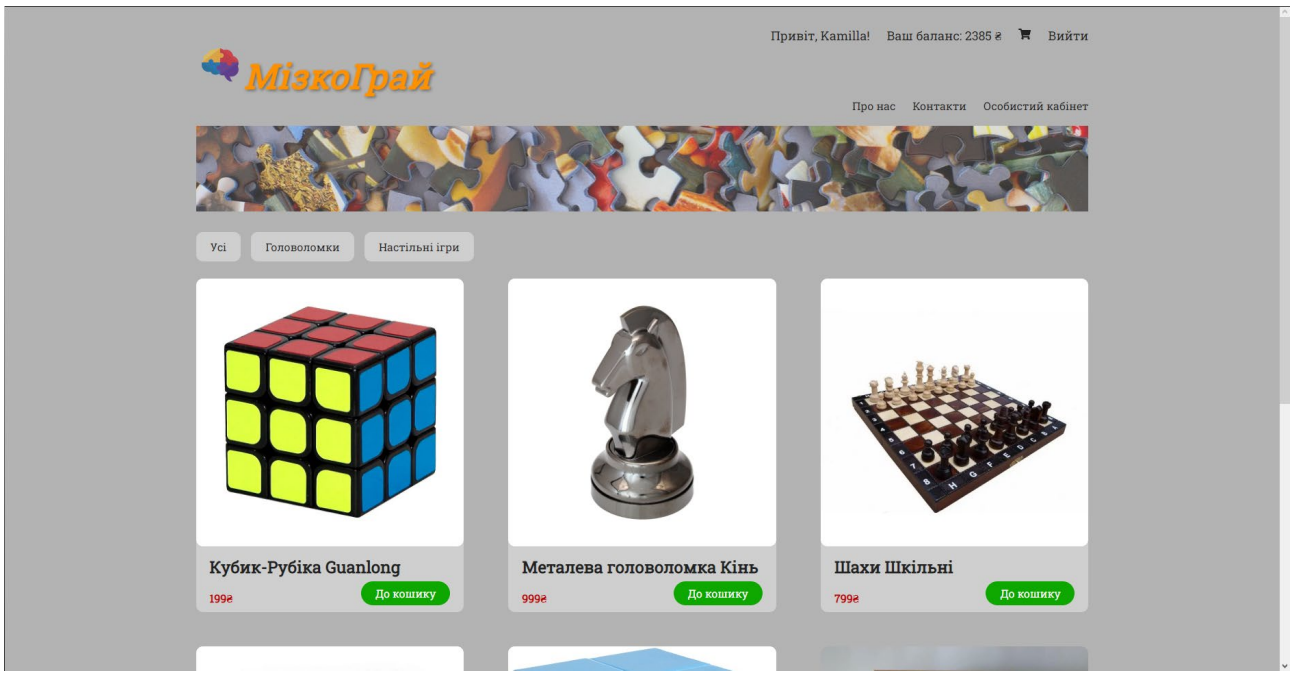

Рис. 2.19. Інтерфейс клієнта

На перший погляд ми не бачимо різниці між інтерфейсами клієнта та адміністратора, але їх принциповою відмінністю є доступ до адміністративної панелі. У клієнта такої можливості немає. Тож розберемо функціонал, доступний клієнту.

Вгорі сторінки розташовані кнопки: «Баланс», «Вийти», «Кошик», «Про нас», «Контакти» та «Особистий кабінет» (рис. 2.20).

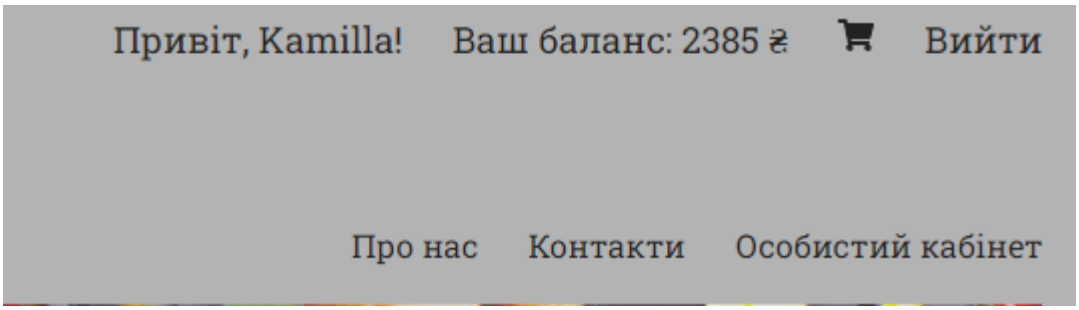

Рис. 2.20. Клієнтська частина

На відміну від адміністратора, після натискання кнопки «Особистий кабінет», клієнту відкривається форма з додатковою інформацією та можливістю її редагування (рис. 2.21).

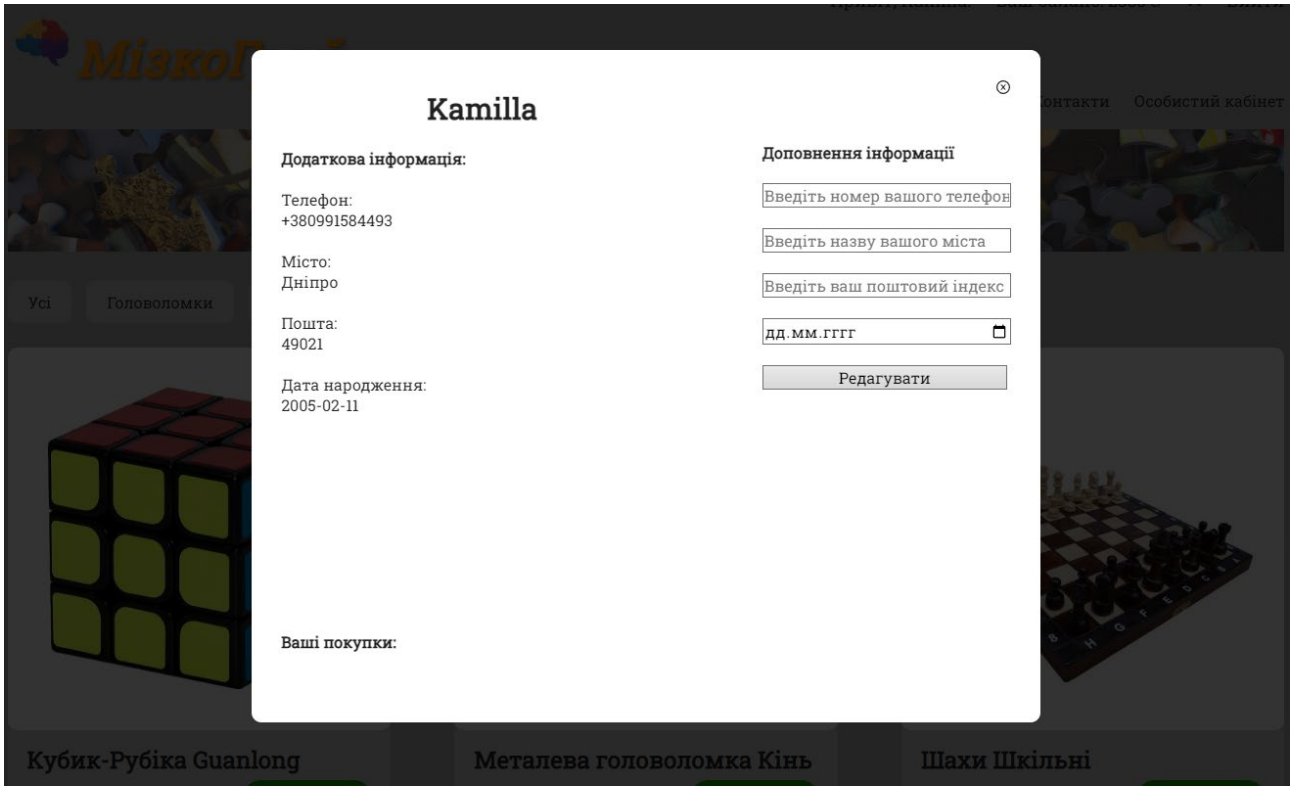

Рис. 2.21. Особистий кабінет клієнта магазину

Кнопка «Про нас» відкриває форму з основною інформацією про розробника веб-додатку та Google картою з адресою (рис. 2.22).

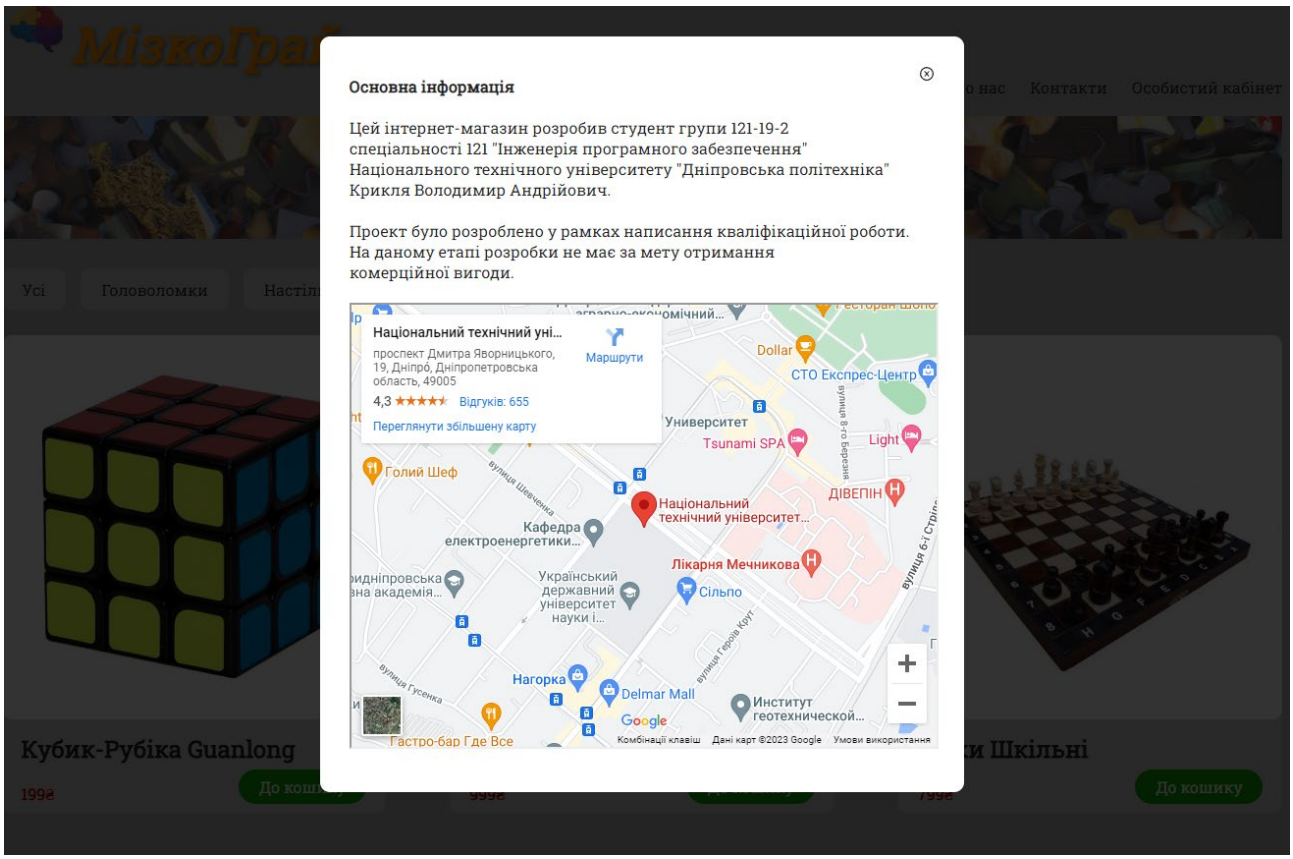

Рис. 2.22. Форма з основною інформацією

Кнопка «Контакти» відкриває форму з контактами розробника магазину: мобільні телефони, соціальні мережі (Telegram, Instagram, Facebook) (рис. 2.23).

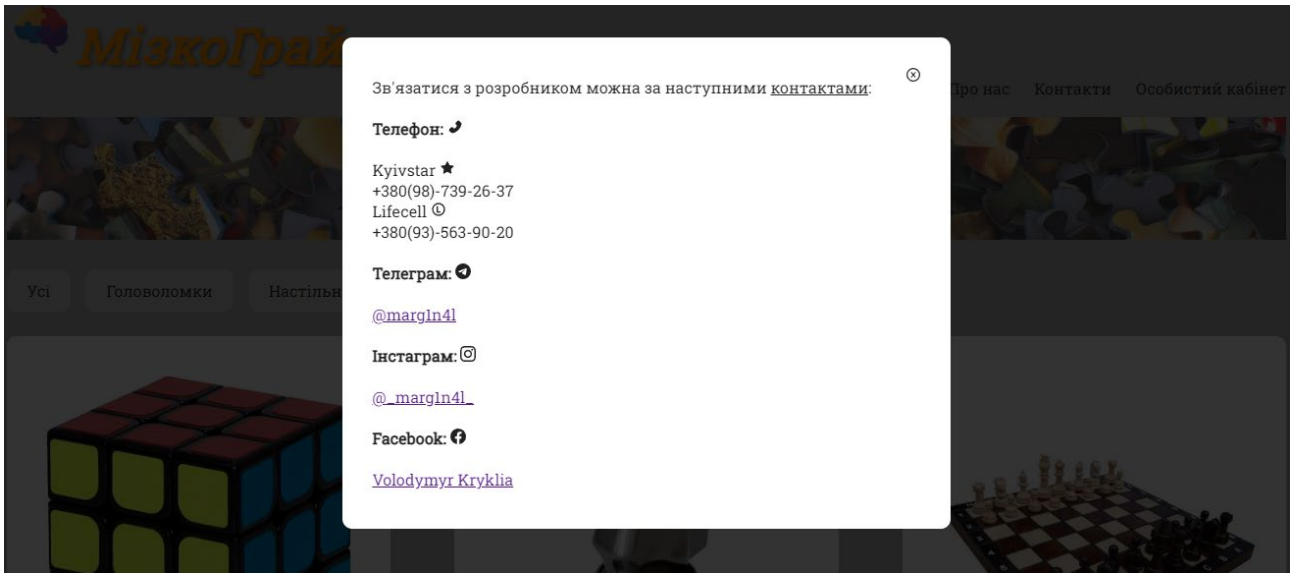

Рис. 2.23. Форма з контактами розробника

Кнопка «Баланс» відкриває форму поповнення балансу користувача на сайті (рис. 2.24).

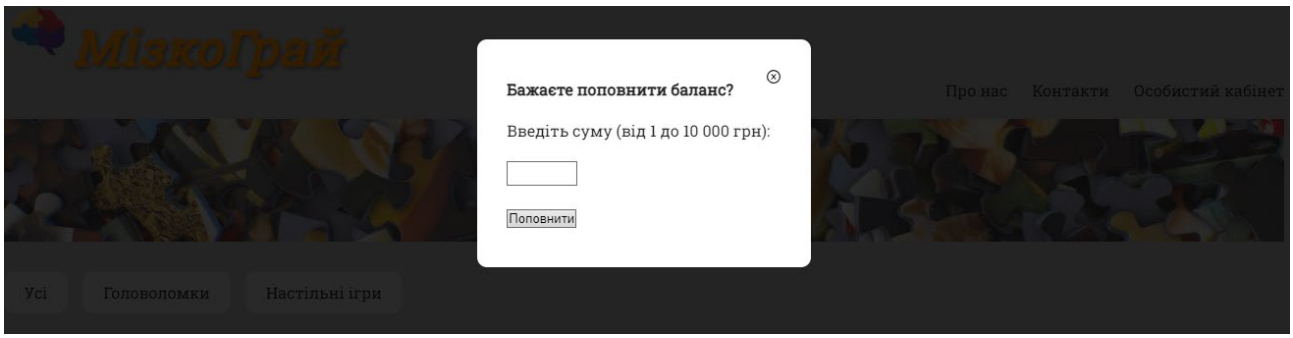

Рис. 2.24. Форма поповнення балансу

Натискання кнопки «Кошик» створює вікно з кошиком, у якому відображаються товари, обрані користувачем для покупки. Додані у кошик товари можна легко видалити звідти, за допомогою кнопки « » (рис. 2.25-26).

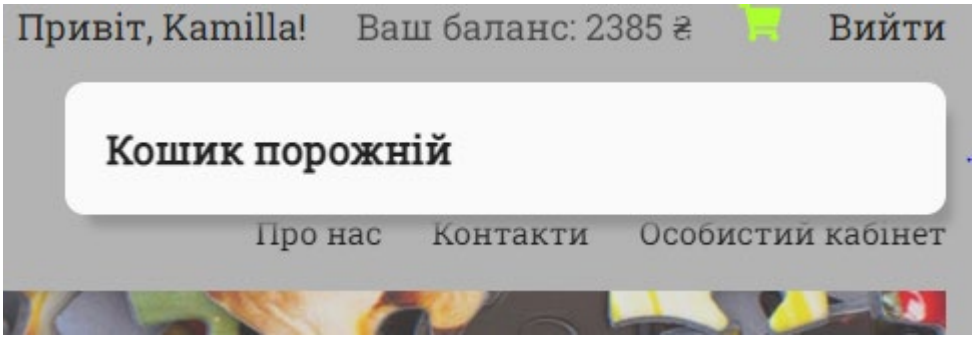

Рис. 2.25. Порожній кошик

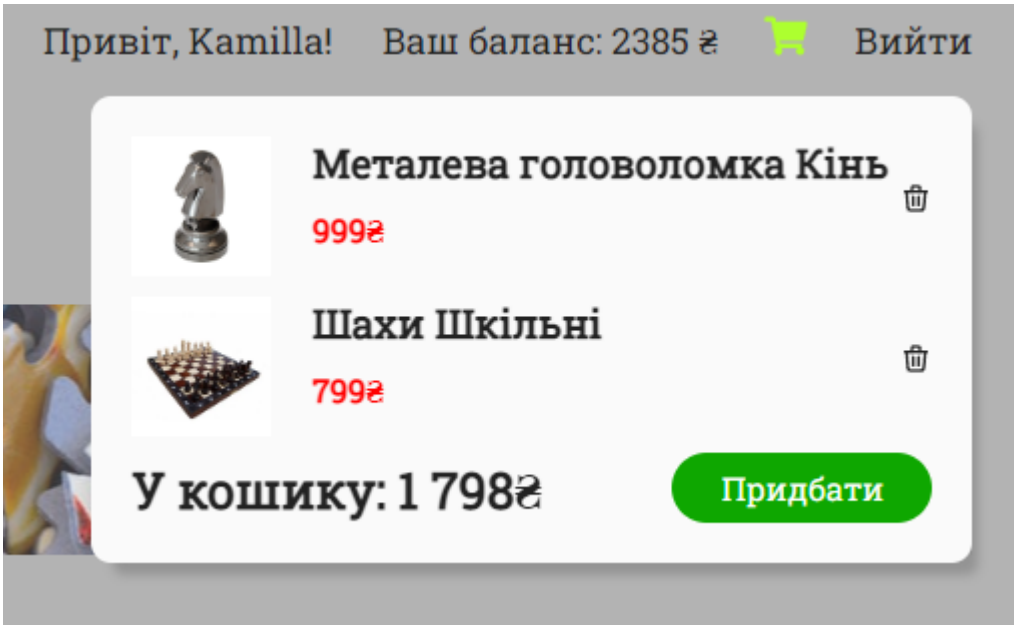

Рис. 2.26. Кошик, наповнений товарами

Кнопка «Вийти» завершує сесію користувача та виходить з облікового запису. Після її натискання, для користування сайтом необхідна нова авторизація.

Також, при наведенні курсору на назву інтернет-магазину «МізкоГрай», напис змінює колір, а при натисканні з’являється форма з особистою інформацією, як у кнопки «Про нас» (рис. 2.27).

Рис. 2.27. Назва інтернет-магазину

Нижче, під фоновим зображенням, розташовані кнопки з назвами категорій товарів (рис. 2.28).

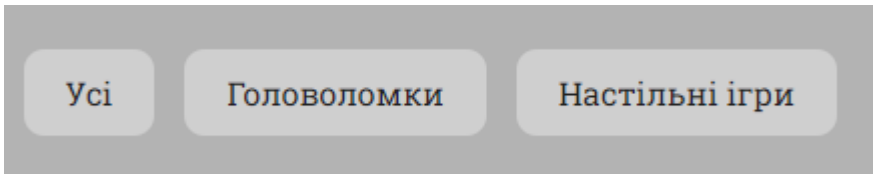

Рис. 2.28. Категорії товарів

Кнопка «Усі» відображає список усіх, доступних для покупки товарів (рис. 2.29).

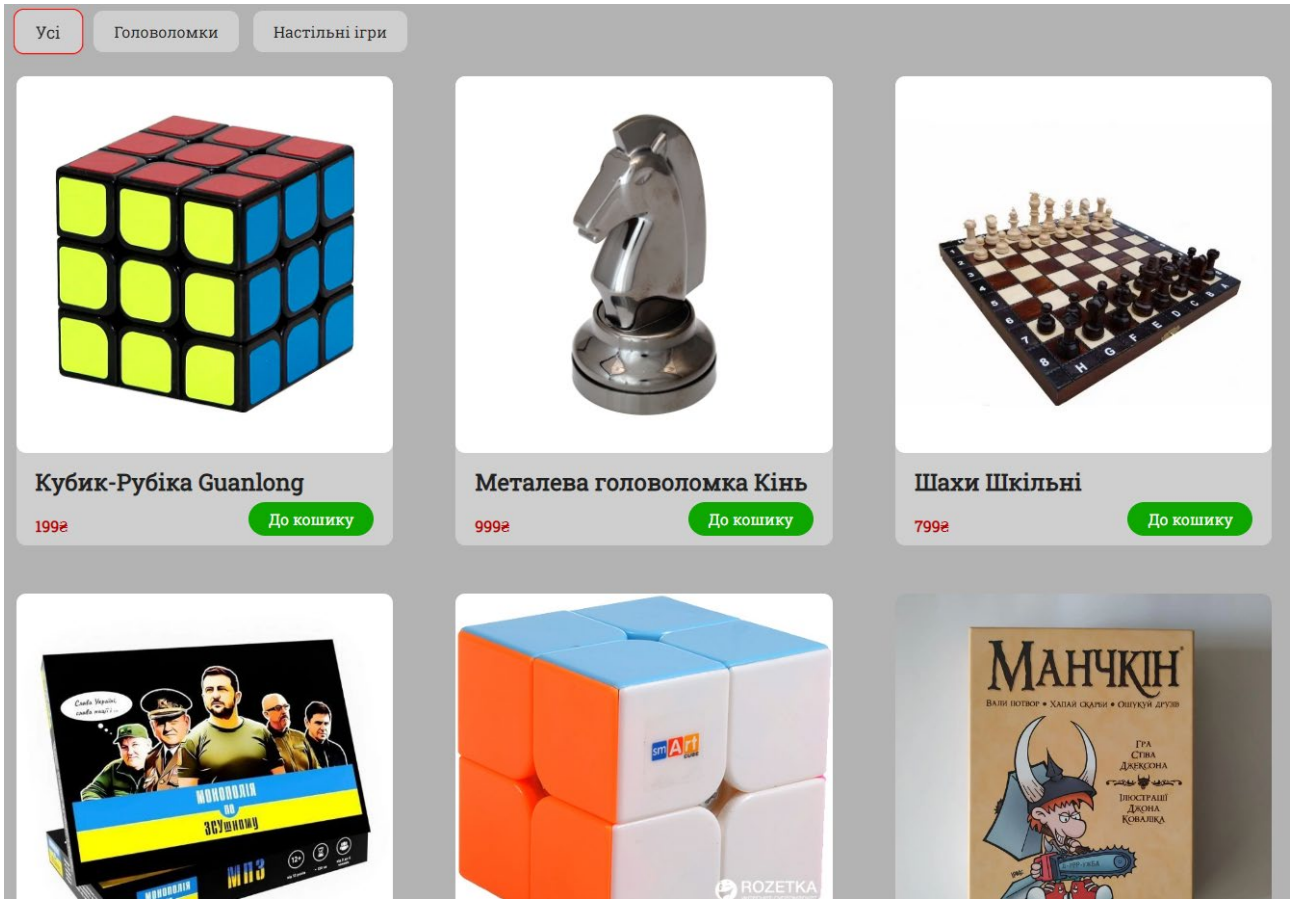

Рис. 2.29. Список усіх товарів

Кнопка «Головоломки» відображає список товарів, які відносяться до категорії головоломок (рис. 2.30).

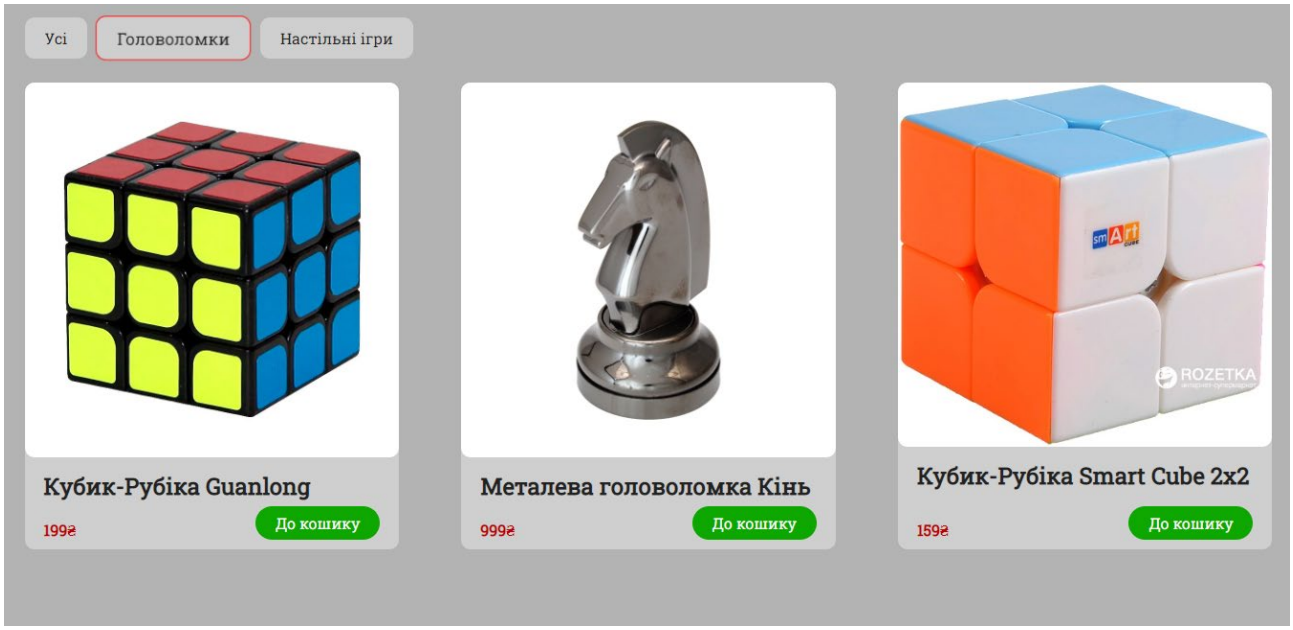

Рис. 2.30. Категорія «Головоломки»

Кнопка «Настільні ігри» відображає список товарів, які відносяться до категорії настільних ігор (рис. 2.31).

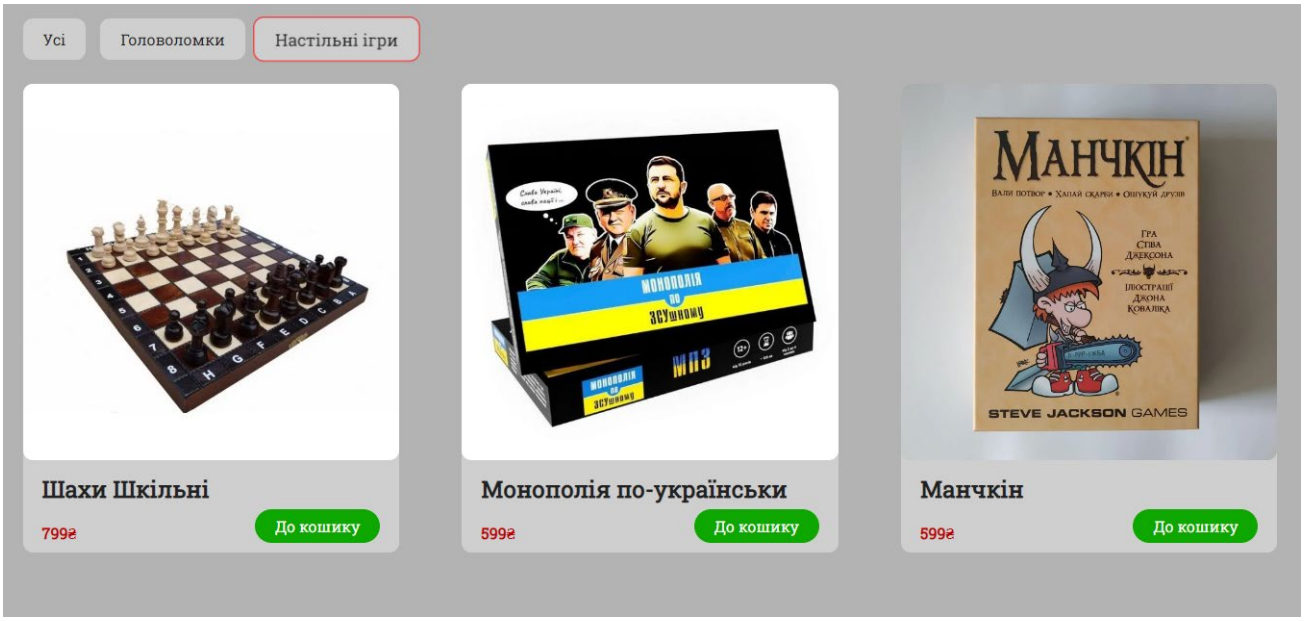

Рис. 2.31. Категорія «Настільні ігри»

Кожен товар має свій окремий блок, в якому знаходиться наступна інформація: фото товару, назва, ціна та кнопка «До кошику», яка додає товар у кошик (рис. 2.32).

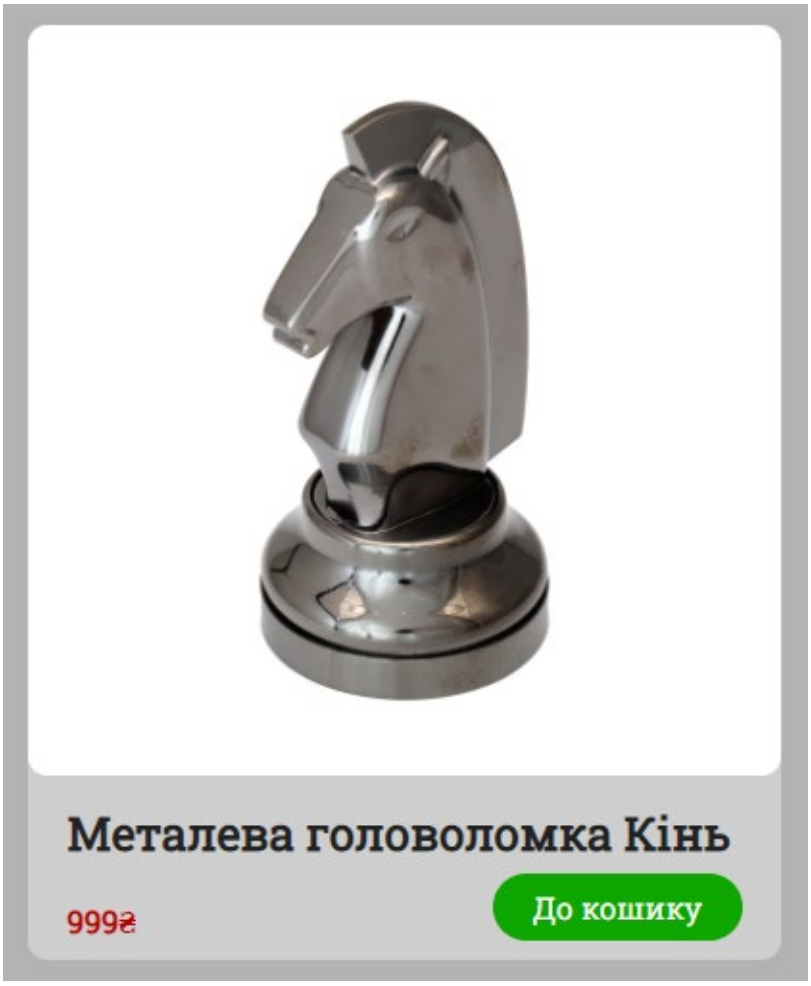

Рис. 2.32. Блок товару

При наведенні курсору на фото товару, воно збільшується, а при натисканні відкривається форма з повною інформацією про товар, яка включає повний опис (рис. 2.33).

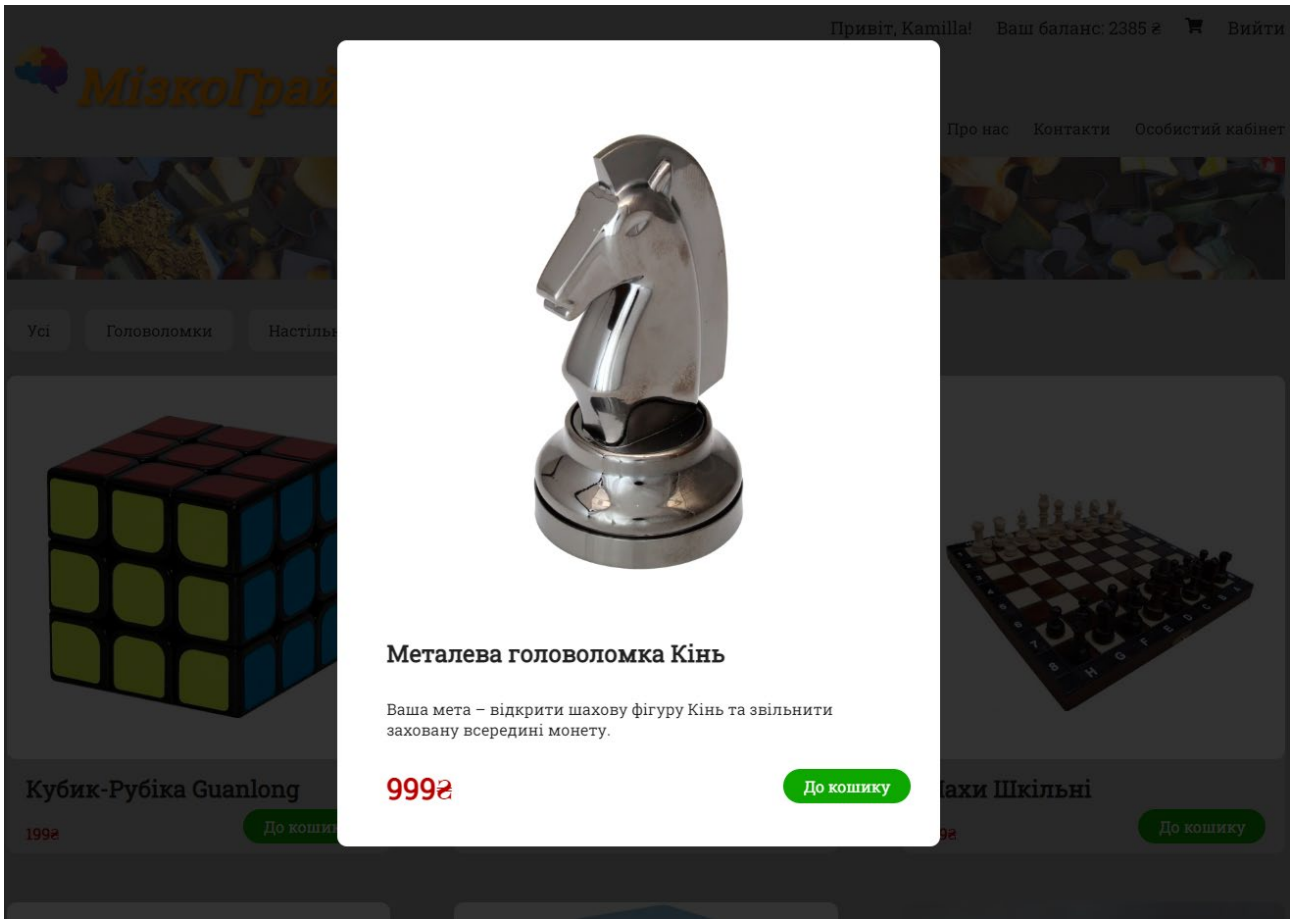

Рис. 2.33. Форма з повним описом товару

У правому нижньому куті сторінки знаходиться попередження про авторські права створеного веб-додатку (рис. 2.34).

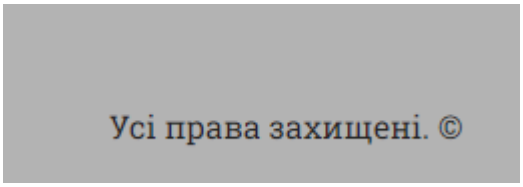

Рис. 2.34. Авторські права

Таким чином, створений інтерфейс користувача є легким у освоєнні, інтуїтивно зрозумілим, забезпечує високу якість обслуговування та задоволеність, як клієнтів, так і адміністраторів цього інтернет-магазину.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вихідні дані:

1. передбачуване число операторів програми – 1632;

2. коефіцієнт складності програми – 1,25;

3. коефіцієнт корекції програми в ході її розробки – 0,2;

4. годинна заробітна плата розробника – 115 грн/год (за версією сайта WorkUA) - https://www.work.ua/salary-dnipro-it/;

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;

6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,1;

7. вартість машино-години ЕОМ – 18 грн/год (1 грн – е/е, 10 грн – ПЗ, 7 грн -амортизація).

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки. Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,}$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_{д}$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q – передбачуване число операторів (1632);

C – коефіцієнт складності програми (1,25);

p – коефіцієнт корекції програми в ході її розробки (0,2).

$$Q = 1632 \cdot 1{,}25 \cdot (1 + 0{,}2) = 2448;$$

Витрати праці на вивчення опису задачі t_{u} визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_{u} = \frac{Q \cdot B}{(75...85) \cdot K}, \text{ людино-годин}$$

де В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,2);

К – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності (1,1);

$$t_{u} = \frac{2448 \cdot 1{,}2}{85 \cdot 1{,}1} = 31{,}42 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_{a} = \frac{Q}{(20...25) \cdot K};$$

$$t_{a} = \frac{2448}{20 \cdot 1{,}1} = 111{,}27 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)\cdot K};$$

$$t_n = \frac{2448}{25\cdot 1,1} = 89 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)\cdot K};$$

$$t_{отл} = \frac{2448}{5\cdot 1,1} = 445 \text{ людино-годин.}$$

– за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1,2 \cdot t_{отл};$$

$$t_{отл}^{к} = 1,2 \cdot 445 = 534 \text{ людино-годин.}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до};$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису;

$$t_д = \frac{Q}{(15...20)\cdot K};$$

$$t_{др} = \frac{2448}{20\cdot 1,1} = 111,27 \text{ людино-годин.}$$

де $t_{до}$ – трудомісткість редагування, печатки й оформлення документації;

$$t_{до} = 0,75 \cdot t_{др};$$

$$t_{до} = 0{,}75 \cdot 111{,}27 = 83{,}45 \text{ людино-годин.}$$

$$t_{д} = 111{,}27 + 83{,}45 = 194{,}72 \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 31{,}42 + 111{,}27 + 89 + 445 + 194{,}72 = 921{,}41 \text{ людино-годин.}$$

У результаті ми розрахували, що в загальній складності необхідно 921,41 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $Кпо$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$Кпо = З_{ЗП} + З_{МВ}, \text{ грн,}$$

$З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

З урахуванням того, що середня годинна зарплата розробника становить 115 грн/год, то отримаємо:

$$З_{ЗП} = 921{,}41 \cdot 115 = 105\,962{,}15 \text{ грн}$$

Вартість машинного часу $З_{МВ}$, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{МВ} = t_{отл} \cdot C_М \text{, грн,}$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год;

$C_М$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 445 \cdot 18 = 8010 \text{ грн}$$

Звідси витрати на створення програмного продукту:

$$Кпо \quad 105\,962{,}15 + 8010 = 113\,972{,}15 \text{ грн}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{, мес.}$$

де B_k – число виконавців;

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p = 176 годин).

Витрати на створення програмного продукту:

$$T = \frac{921{,}41}{1 \cdot 176} = 5{,}23 \text{ міс.}$$

Вартість розробки інтернет-магазину становить 113 972,15 грн. Час розробки очікується приблизно 5,23 місяці при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Цей термін включає час, необхідний для дослідження, розробки алгоритму, дизайну і документування. Загальна кількістю людино-годин, яку буде витрачено на розробку – 921,41.

ВИСНОВКИ

Метою кваліфікаційної роботи є розробка веб-орієнтованого додатку з продажу настільних ігор та головоломок за допомогою React.JS, PHP у середовищі програмування Visual Studio Code.

Актуальність поставленої задачі обумовлюється тим, що створення інтернет-магазину з таким асортиментом товарів може бути ефективним рішенням для підприємства, яке має за мету продаж товарів у веб-просторі, що дає змогу охопити більшу аудиторії клієнтів та не витрачатися на відкриття фізичного магазину.

Розроблена програма являє собою інтернет-магазин, що реалізує інтуїтивно зрозумілий інтерфейс, дає змогу користувачам купувати товари, реєструватися у базі клієнтів та керувати своїми особистими даними.

Сайт призначений для забезпечення діяльності інтернет-магазину і надає можливість виконувати наступні дії:

1) Моніторинг товарів на сайті, їх цін та опису.
2) Реєстрація у базі клієнтів.
3) Вхід до облікового запису, що надає можливість здійснювати покупки, редагувати додаткову інформацію про себе, поповнювати баланс.
4) Додавання товарів до кошику, видалення з нього.
5) У адміністраторів: додавання нових товарів у базу прямо з інтерфейсу веб-додатку, видалення їх та редагування.
6) Перегляд вкладок з інформацією про сайт, у яких описуються: основна інформація про розробника, його контакти та адреса у формі Google Maps.

Ці функціональні характеристики є основою успішної роботи інтернет-магазину та забезпечують високу якість обслуговування та задоволеність клієнтів.

Веб-додаток призначений для використання клієнтами по всій території України, які зацікавлені у придбанні товарів цієї категорії. Областю застосування є продаж та доставка головоломок та настільних ігор на території України.

Структура програми являє собою веб-додаток, написаний на мовах програмування JavaScript, що взаємодіє з базою даних «my-shop». База даних розроблена мовою SQL в системі управління базами даних MySQL.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення 921,41 люд-год, підраховані витрати на створення програмного забезпечення - 113 972,15 грн і гаданий період розробки 5,23 міс при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ReactJS. URL: https://reactjs.org/
2. Alex Banks and Eve Porcello. Learning React: Modern Patterns for Developing React Apps. — O'Reilly Media, 2017. — 350 с.
3. Eric Elliott. Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries. — O'Reilly Media, 2014. — 254 с.
4. Cay Horstmann. React for the Real World: Large-scale, Fast, and Scalable Web Applications. — Packt Publishing, 2020. — 432 с.
5. Robin Wieruch. The Road to React: Your journey to master plain yet pragmatic React.js. — Independently published, 2018. — 394 с.
6. Adam Freeman. Pro React 16. — Apress, 2019. — 1079 с.
7. JavaScript. URL: https://uk.javascript.info
8. Douglas Crockford. JavaScript: The Good Parts. — O'Reilly Media, 2008. — 176 с.
9. Eric Freeman, Elisabeth Robson. Head First JavaScript Programming: A Brain-Friendly Guide. — O'Reilly Media, 2014. — 704 с.
10. Marijn Haverbeke. Eloquent JavaScript: A Modern Introduction to Programming. — No Starch Press, 2018. — 472 с.
11. David Flanagan. JavaScript: The Definitive Guide. — O'Reilly Media, 2020. — 706 с.
12. Kyle Simpson. You Don't Know JS: Up & Going. — O'Reilly Media, 2015. — 88 с.
13. PHP. URL: https://www.php.net/manual/en/index.php
14. David Sklar, Adam Trachtenberg. PHP Cookbook. — O'Reilly Media, 2014. — 820 с.
15. Matt Zandstra. PHP Objects, Patterns, and Practice. — Apress, 2017. — 768 с.
16. Rasmus Lerdorf, Kevin Tatroe, Peter MacIntyre. Programming PHP. — O'Reilly Media, 2013. — 540 с.

17. Larry Ullman. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide. — Peachpit Press, 2017. — 704 с.
18. Robin Nixon. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. — O'Reilly Media, 2014. — 812 с.
19. MySQL. URL: https://www.mysql.com
20. Paul DuBois. MySQL Cookbook. — O'Reilly Media, 2014. — 866 с.
21. Robin Nixon. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. — O'Reilly Media, 2014. — 812 с.
22. Bill Karwin. SQL Antipatterns: Avoiding the Pitfalls of Database Programming. — Pragmatic Bookshelf, 2010. — 328 с.
23. Charles Bell, Mats Kindahl, Lars Thalmann. MySQL High Availability: Tools for Building Robust Data Centers. — O'Reilly Media, 2014. — 746 с.
24. Jeremy D. Zawodny, Derek J. Balling. High Performance MySQL: Optimization, Backups, and Replication. — O'Reilly Media, 2012. — 826 с.
25. Eric A. Meyer, Estelle Weyl. CSS: The Definitive Guide. — O'Reilly Media, 2017. — 1090 с.
26. Jon Duckett. HTML & CSS: Design and Build Websites. — Wiley, 2011. — 512 с.
27. David McFarland. CSS: The Missing Manual. — O'Reilly Media, 2015. — 718 с.
28. Rachel Andrew. The New CSS Layout. — Smashing Magazine, 2017. — 218 с.
29. Dan Cederholm. CSS3 For Web Designers. — A Book Apart, 2010. — 112 с.

ДОДАТОК А

КОД ПРОГРАМИ

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="shortcut icon" href="img/icons8-brain-64.png" type="image/png">
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <title>МізкоГрай - магазин головоломок та настільних ігор</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>

import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

import React from 'react';
import Header from './components/Header';
import Footer from './components/Footer';
import Items from './components/Items';
import Categories from './components/Categories';
import ShowInfo from './components/ShowInfo';
import User from './components/User';

class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      isLoading: true,
      allitems: [],
      currentItems: [],
      showInfo: false,
      info: {},
      orders: [],
      userItems: [],

    };
    this.state.currentItems = this.state.allitems;
    this.addToOrder = this.addToOrder.bind(this);
    this.deleteOrder = this.deleteOrder.bind(this);
    this.chooseCategory = this.chooseCategory.bind(this);
    this.onShowInfo = this.onShowInfo.bind(this);
  }
```

```
  componentDidMount() {
    const userBalance = localStorage.getItem('userBalance');
    if (userBalance) {
      this.setState({ userBalance: parseInt(userBalance) });
    }


    fetch("http://localhost/getitems.php")
      .then((response) => response.json())
      .then((data) => {
        const items = data.map((item) => ({
          id: item.id,
          title: item.title,
          description: item.description,
          price: item.price,
          img: item.img,
          category: item.category,
        }));
        this.setState({ allitems: items, currentItems: items, isLoading: false });
      })
      .catch((error) => console.log(error));
  }

  render() {
    const { isLoading, currentItems, showInfo, info, orders, userItems } = this.state;
    return (
      <div className="wrapper">
        <User onDelete={this.deleteOrder} orders={orders} userItems={userItems}
userBalance={this.state.userBalance} />
        <Header userItems={this.state.userItems} />
        <Categories chooseCategory={this.chooseCategory} />
        {isLoading ? (
          <div>Loading...</div>
        ) : (
          <Items onShowInfo={this.onShowInfo} allitems={currentItems}
onAdd={this.addToOrder} />
        )}
        {showInfo && <ShowInfo onAdd={this.addToOrder} onShowInfo={this.onShowInfo}
item={info} />}
        <Footer />
      </div>
    );
  }

  onShowInfo(item) {
    this.setState({ info: item });
    this.setState({ showInfo: !this.state.showInfo });
  }

  chooseCategory(category) {
    if (category === "all") {
      this.setState({ currentItems: this.state.allitems });
      return;
    }

    this.setState({
      currentItems: this.state.allitems.filter((el) => el.category === category),
    });
  }

  deleteOrder(id) {
```

```
    this.setState({ orders: this.state.orders.filter((el) => el.id !== id) });
  }

  addToOrder(item) {
    let isInArray = false;
    this.state.orders.forEach((el) =>{
      if(el.id === item.id)
        isInArray = true
    })
    if(!isInArray)
      this.setState({orders: [...this.state.orders, item]})
  }

}

export default App;
```

```
@import url('https://fonts.googleapis.com/css2?family=Roboto+Slab&display=swap');

* {
  margin: 0;
  padding: 0;
}

body {
  background: rgb(179, 179, 179);
  color: #222;
  font-family: 'Roboto Slab', serif;
  font-weight: 300;

}

header{
  position: relative;

}

.wrapper {
  width: 70%;
  margin: 50px auto;
}

.brain{
  width: 70px;
  opacity: 80%;
}

.name{
  font-size: 30px;
  padding-left: 20%;
}


header .logo {
  font-weight: 1000;
  font-size: 50px;
  color: rgb(255, 145, 0);
  text-shadow: 2px 2px 4px rgba(0,0,0,0.4);
  font-style: italic;
  letter-spacing: 1px;
  transition: all 0.5s ease;
  cursor: pointer;
```

```
}

header .logo:hover{
  color: rgb(116, 67, 250);
    text-shadow: 2px 2px 8px rgba(255,0,0,0.6);
    transform: scale(1.1);
}

ul.nav {
  float: right;
  list-style: none;
  position: relative;
}

ul.nav li {
  display: inline;
  margin-left: 25px;
  cursor: pointer;
  transition: opacity 300ms ease;
}

ul.nav li:hover {
  opacity: 0.5;
}

ul.log {
  float: right;
  position: relative;
  bottom: 20px;
  font-size: large;
}

ul.log li {
  display: inline;
  margin-left: 25px;
  cursor: pointer;
  transition: opacity 300ms ease;
}

ul.log li:hover {
  opacity: 0.5;
}

ul.user{
  list-style: none;
  float: right;
  position: relative;
  bottom: 20px;
  font-size: large;
}

ul.user li {
  display: inline;
  margin-left: 25px;

}

ul.user li.exit{
  cursor: pointer;
  transition: opacity 300ms ease;
}
```

```
ul.user li.exit:hover{
  opacity: 0.5;
}


.shop-cart-button{

  cursor: pointer;
  position: relative;
  transition: all 500ms ease;

}

.shop-cart-button:hover,
.shop-cart-button.active{
  color: greenyellow;
  transform: scale(1.2);
}

.shop-cart{
  position: absolute;
  top: 70px;
  right: 15%;
  width: 400px;
  border-radius: 10px;
  background:#fafafa;
  -webkit-box-shadow: 7px 10px 5px -4px rgba(148,148,148,1);
  -moz-box-shadow: 7px 10px 5px -4px rgba(148,148,148,1);
  box-shadow: 7px 10px 5px -4px rgba(148,148,148,1);
  z-index: 100;
  padding: 20px;
  animation: ani 0.5s forwards;
}

@keyframes ani{
  0% {opacity: 0;};
  100% {opacity: 1;}
}

.order{
  width: 50%;
  float: left;
  margin-bottom: 10px;
}

.order img{
  width: 100px;
  float: left;
  margin-right: 20px;
}

.order h2{
  font-size: 20px;
  margin-bottom: 10px;
}

.order b{
  color: red;
  font-weight: 600;
}

.shop-cart .empty h2{
```

```
  font-size: 20px;
}

.shop-cart .item{
  width: 100%;
  float: left;
  margin-bottom: 10px;
}

.shop-cart .item img{
  width: 70px;
  float: left;
  margin-right: 20px;
}

.shop-cart .item h2{
  font-size: 20px;
  margin-bottom: 10px;
}

.shop-cart .item b{
  color: red;
  font-weight: 600;
}

.shop-cart .item .delete-icon{
  color: #222;
  float: right;
  position: relative;
  bottom: 13px;
  cursor: pointer;
  transition: all 500ms ease;
}

.shop-cart .item .delete-icon:hover{
  color: yellowgreen;
  transform: scale(1.3);
}

.shop-cart .summ{
  float: right;
  width: 100%;
  font-weight: 600;
  font-size: 25px;
}

.red{
  float: right;
  position: relative;
  top: -300px;
}

.red > input {
  width: 250px;
}

.account .admin {
  width: 42%;
}

.account .admin > div{
  display: inline-flex;
```

```
  padding: 20px;
}

.admin-item{
  position: relative;
}


header .presentation {

  margin-top: 40px;
  background-image: url('./img/photo-1494059980473-813e73ee784b.jpg');
  background-position: center;
  height: 125px;
  width: 100%;
  opacity: 70%;
  margin-bottom: 30px;
}

.categories div{
  display: inline-block;
  background: #cfcfcf;
  border-radius: 10px;
  padding: 10px 20px;
  margin-bottom: 25px;
  margin-right: 15px;
  cursor: pointer;
  border: 1px solid transparent;
  transition: all 500ms ease;
}

.categories div:hover{
  border-color: red;
  transform: scale(1.1);
}

.info, .about, .contacts, .login, .reg, .balance, .account{
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(0, 0, 0, 0.8);
  z-index: 1000;
  overflow: auto;
}

.info > div {
  width: 30%;
  position: relative;
  margin: 5% auto;
  padding: 40px 30px;
  background: #fff;
  border-radius: 10px;
  animation: ani 0.5s forwards;

}

.contacts > div {
  width: 30%;
  position: relative;
  margin: 5% auto;
```

```
  padding: 40px 30px;
  background: #fff;
  border-radius: 10px;
  animation: ani 0.5s forwards;

}

.about > div {
  width: 32%;
  position: relative;
  margin: 5% auto;
  padding: 40px 30px;
  background: #fff;
  border-radius: 10px;
  animation: ani 0.5s forwards;

}

.login > div {
  width: 15%;
  position: relative;
  margin: 5% auto;
  padding: 40px 30px;
  background: #fff;
  border-radius: 10px;
  animation: ani 0.5s forwards;

}

.reg > div {
  width: 15%;
  position: relative;
  margin: 5% auto;
  padding: 40px 30px;
  background: #fff;
  border-radius: 10px;
  animation: ani 0.5s forwards;

}

.balance > div {
  width: 15%;
  position: relative;
  margin: 5% auto;
  padding: 40px 30px;
  background: #fff;
  border-radius: 10px;
  animation: ani 0.5s forwards;

}

.account > div {
  width: 40%;
  position: relative;

  margin: 5% auto;
  padding: 40px 30px;
  background: #fff;
  border-radius: 10px;
  animation: ani 0.5s forwards;
}
```

```
input{
  font-size: medium;
  font-family: 'Roboto Slab', serif;
}



main{
  display: flex;
  width: 100%;
  flex-wrap: wrap;
  justify-content: space-between;

}

main .item{
  width: 30%;
  height: auto;
  margin-bottom: 50px;
  background: #cfcfcf;
  overflow: hidden;
  position: relative;
  padding-bottom: 20px;
  border-radius: 10px;
}

main .item img, .info img{
  width: 100%;
  border-radius: 10px 10px 10px 10px;
  transition: transform 300ms ease;
  cursor: pointer;
}



main .item img:hover,
.info img:hover {
  transform: scale(1.05);
}



main p, main h2 ,
.info b, .info p, .info h2{
  margin-top: 10px;
  margin-right: 10px;
  margin-left: 20px;
  margin-bottom: 30px;
  color: #222;
}

main b {
  color: rgb(187, 0, 0);
  position: absolute;
  left: 0px;
  bottom: 0px;
  margin-bottom: 10px;
  margin-left: 20px;
}

.youror{
  top: 200px;
```

```
  position: relative;
}

.info b {
  color: rgb(187, 0, 0);
  font-size: 30px;
}

main .add-to-cart {
  position: absolute;
  right: 20px;
  bottom: 10px;
  background: rgb(14, 167, 0);
  width: 130px;
  height: 35px;
  text-align: center;
  line-height: 35px;
  color: #fff;
  border-radius: 20px;
  cursor: pointer;
  transition: transform 300ms ease;
}

.buy-button{
  position: absolute;
  right: 20px;
  bottom: 20px;
  background: rgb(14, 167, 0);
  width: 130px;
  height: 35px;
  text-align: center;
  line-height: 35px;
  color: #fff;
  border-radius: 20px;
  cursor: pointer;
  transition: transform 300ms ease;
}

.buy-button:hover{
  transform: scale(1.2);
}

.info .add-to-cart{
  float: right;
  right: 20px;
  bottom: 10px;
  background: rgb(14, 167, 0);
  width: 130px;
  height: 35px;
  text-align: center;
  line-height: 35px;
  color: #fff;
  border-radius: 20px;
  cursor: pointer;
  transition: transform 300ms ease;
}

main .add-to-cart:hover,
.info .add-to-cart:hover{
  transform: scale(1.2);
}
```

```
.close-button{
  float: right;
  position: relative;
  top: -10px;
  cursor: pointer;
  transition: transform 500ms ease;
}

.close-button:hover {
  transform: scale(1.3);
  color: rgb(14, 167, 0);
}

footer {
  text-align: right;
  margin-top: 100px;
  margin-right: -250px;
  margin-bottom: -30px;
}

import React, { useState }  from 'react'
import { AiOutlineCloseCircle } from 'react-icons/ai';

export default function AddBalance(props) {

    const [balanceToAdd, setBalanceToAdd] = useState('');

    const handleInputChange = (event) => {
        setBalanceToAdd(event.target.value);
    };

    const handleSubmit = (event) => {
        event.preventDefault();
        const newBalance = parseInt(balanceToAdd);
        if (!newBalance || newBalance <= 0 || newBalance > 10000) {
          alert('Введіть коректну суму для поповнення!');
          return;
        }
        props.onAddBalance(newBalance);
        setBalanceToAdd('');
      };



  return (
    <div className='balance'>
      <div>
        <AiOutlineCloseCircle className='close-button' onClick={props.onClose} />
        <b>Бажаєте поповнити баланс?</b> <br /><br />
        <form onSubmit={handleSubmit}>
          <label>
            Введіть суму (від 1 до 10 000 грн): <br /><br />
            <input type='number' min='1' max='10000' value={balanceToAdd}
onChange={handleInputChange}/>
          </label> <br /><br />
          <button type='submit'>Поповнити</button>
        </form>
      </div>
    </div>
  )
}
```

```
import React from 'react';

function Balance({ userBalance }) {
  return (
    <span>{userBalance}</span>
  );
}

export default Balance;

import React, { Component } from 'react'

export class Categories extends Component {
    constructor(props) {
        super(props)
        this.state ={
            categories: [
                {
                    key: 'all',
                    name: 'Усі'
                },
                {
                    key: 'puzzle',
                    name: 'Головоломки'
                },
                {
                    key: 'board-games',
                    name: 'Настільні ігри'
                }

            ]
        }
    }
  render() {
    return (
      <div className='categories'>
        {this.state.categories.map(el => (
            <div key={el.key} onClick={() =>
this.props.chooseCategory(el.key)}>{el.name}</div>
        ))}
      </div>
    )
  }
}

export default Categories

import React, { Component } from 'react'

export class FinishedOrder extends Component {
  render() {
    return (
      <div className='item'>
        <img src={"./img/" + this.props.item.img}/>
        <h2>{this.props.item.title}</h2>
        <b>{this.props.item.price}₴</b>
      </div>
    )
  }
}

export default FinishedOrder
```

```
import React, { Component } from 'react'
import FinishedOrder from './FinishedOrder'

export class FinishedOrders extends Component {
  render() {
    return (
      <main>

        {this.props.userItems.map(el => (
            <FinishedOrder key={el.id} item={el}/>
        ))}
      </main>
    )
  }
}

export default FinishedOrders

import React from 'react'

export default function Footer() {
  return (
    <footer>
        Усі права захищені. &copy;
    </footer>
  )
}

import React, { useState, useEffect} from 'react'
import { AiOutlineCloseCircle } from 'react-icons/ai'
import { AiFillStar } from 'react-icons/ai'
import { AiFillPhone } from 'react-icons/ai'
import { TbCircleLetterL } from 'react-icons/tb'
import { BsTelegram } from 'react-icons/bs'
import { BsInstagram } from 'react-icons/bs'
import { BsFacebook } from 'react-icons/bs'
import FinishedOrders from './FinishedOrders'


export default function Header(props) {
  let [aboutUs, openAboutUs] = useState(false)
  let [contacts, openContacts] = useState(false)
  let [account, openAccount] = useState(false)
  let [userId, setUserId] = useState(null);
  let userItems = props.userItems;

  let username = localStorage.getItem('username');
  const phone = localStorage.getItem('phone');
  const city = localStorage.getItem('city');
  const date = localStorage.getItem('date');
  const postal = localStorage.getItem('postal');
  const role = localStorage.getItem('role');


  useEffect(() => {
    const savedUserId = getCookie('user_id');
    setUserId(savedUserId);
    if (savedUserId) {
      fetch(`http://localhost/getuser.php?id=${savedUserId}`)
        .then(response => response.json())
        .then(data => {
```

```
          localStorage.setItem('role', data.role)
          localStorage.setItem('phone', data.phone);
          localStorage.setItem('city', data.city);
          localStorage.setItem('date', data.date);
          localStorage.setItem('postal', data.postal);
        })
    }

  }, []);

  function getCookie(name) {
    const cookies = document.cookie.split(';');
    for (let i = 0; i < cookies.length; i++) {
      const cookie = cookies[i].trim();
      if (cookie.startsWith(name + '=')) {
        return cookie.substring(name.length + 1);
      }
    }
    return null;
  }

  const showInfo = (el) => {
    if (el === "null"){
      el = "-----"
    }
    return el;
  }

  const checkAdmin = (user) => {
    if (user === 'admin'){
      username = false;
      return true;
    } else {
      return false;
    }
  }

  const checkUser = (name, role) => {
    if (name === null){
      if (checkAdmin(role) === false){
        return true;
      }
    }
    return false;
  }

  return (
    <header>
        <img src='./img/brain.png' className='brain'/>
        <span className='logo' onClick={() => openAboutUs(aboutUs =
!aboutUs)}>МіякоГрай</span>
        <div>

            <ul className='nav'>
                <li onClick={() => openAboutUs(aboutUs = !aboutUs)}>Про нас</li>
                {aboutUs && (
                  <div className='about'>
                    <div>
                      <AiOutlineCloseCircle className='close-button' onClick = {() =>
openAboutUs(aboutUs = !aboutUs)} />
                      <b>Основна інформація</b> <br /> <br />
                      Цей інтернет-магазин розробив студент групи 121-19-2 <br />
```

```
                    спеціальності 121 "Інженерія програмного забезпечення" <br />
                    Національного технічного університету "Дніпровська
політехніка"<br />
                    Крикля Володимир Андрійович.<br /> <br />
                    Проект було розроблено у рамках написання кваліфікаційної
роботи.<br />
                    На даному етапі розробки не має за мету отримання<br />
                    комерційної вигоди. <br /> <br />
                    <iframe
src="https://www.google.com/maps/embed?pb=!1m14!1m8!1m3!1d5292.216262417401!2d35.064830
1!3d48.4544548!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x40dbe2d6b66f138d%3A0x98b973a9561df40
f!2z0J3QsNGG0ZbQvtC90LDQu9GM0L3QuNC5INGC0LXRhdC90ZbRh9C90LjQuSDRg9C90ZbQstC10YDRgdC40YL
QtdGCIMKr0JTQvdGW0L_RgNC-0LLRgdGM0LrQsCDQv9C-
0LvRltGC0LXRhdC90ZbQutCwwrs!5e0!3m2!1suk!2sua!4v1683044484353!5m2!1suk!2sua"
width="100%" height="450" border="0" allowfullscreen="" loading="lazy"
referrerpolicy="no-referrer-when-downgrade"></iframe>                    </div>
                  </div>
                )}
                <li onClick={() => openContacts(contacts = !contacts)}>Контакти</li>
                {contacts && (
                  <div className='contacts'>
                    <div>
                      <AiOutlineCloseCircle className='close-button' onClick = {() =>
openContacts(contacts = !contacts)} />
                      Зв'язатися з розробником можна за наступними
<u>контактами</u>:<br /> <br />

                      <b>Телефон:</b> <AiFillPhone/> <br /> <br />
                      Kyivstar <AiFillStar/> <br />
                      +380(98)-739-26-37 <br />
                      Lifecell <TbCircleLetterL/> <br />
                      +380(93)-563-90-20 <br /> <br />

                      <b>Телеграм:</b> <BsTelegram/> <br /> <br />
                      <a href="https://t.me/marg1n4l">@marg1n4l</a> <br /> <br />

                      <b>Інстаграм:</b> <BsInstagram/> <br /> <br />
                      <a href="https://www.instagram.com/_marg1n4l_">@_marg1n4l_</a>
<br /> <br />

                      <b>Facebook:</b> <BsFacebook/> <br /> <br />
                      <a
href="https://www.facebook.com/profile.php?id=100037813683424">Volodymyr Kryklia</a>


                    </div>
                  </div>
                  )}
                <li onClick={() => openAccount(account = !account)}>Особистий
кабінет</li>
                {account && (
                  <div className='account'>
                  {checkUser(username, role) && (
                      <div>
                      <AiOutlineCloseCircle className='close-button' onClick = {() =>
openAccount(account = !account)} />
                      <b>Увійдіть до аккаунту для відображення додаткової
інформації.</b>
                      </div>
                    )}
                  {checkAdmin(role) && (
                    <div className='admin'>
```

```
                      <AiOutlineCloseCircle className='close-button' onClick = {() =>
openAccount(account = !account)} />
                      <b className='name'>Панель керування товарами</b> <br /> <br />
                      <div>
                        <form className='admin-item'
action='http://localhost/additem.php' method='post'>

                          <b>Додавання нового товару:</b> <br /> <br />

                          <input type="text" name="title" placeholder='Назва' />

                          <br /> <br />

                          <input type="text" name="description" placeholder='Опис' />

                          <br /> <br />

                          <input type="number" name="price" placeholder='Ціна' />

                          <br /> <br />

                          <input type="text" name="img" placeholder='Фото' />

                          <br /> <br />

                          <input type="text" name="category" placeholder='Категорія' />

                          <br /> <br />

                          <input className='submit' type="submit" value="Додати" />

                        </form>
                      </div>
                      <div>
                        <form className='admin-item'
action='http://localhost/deleteitem.php' method='post'>

                          <b>Видалення товару:</b> <br /> <br />

                          <input type="text" name="title" placeholder='Назва' />

                          <br /> <br />

                          <input className='submit' type="submit" value="Видалити" />

                        </form>
                      </div>
                      <div>
                        <form className='admin-item'
action='http://localhost/updateitem.php' method='post'>

                          <b>Редагування товару:</b> <br /> <br />

                          <input type="text" name="ontitle" placeholder='Введіть назву
товару' />

                          <br /> <br />

                          Внесіть нові дані про товар:

                          <br /> <br />
```

```
                        <input type="text" name="title" placeholder='Назва' />

                        <br /> <br />

                        <input type="text" name="description" placeholder='Опис' />

                        <br /> <br />

                        <input type="number" name="price" placeholder='Ціна' />

                        <br /> <br />

                        <input type="text" name="img" placeholder='Фото' />

                        <br /> <br />

                        <input type="text" name="category" placeholder='Категорія' />

                        <br /> <br />

                        <input className='submit' type="submit" value="Редагувати" />

                        </form>
                      </div>
                    </div>
                  )}
                    {username && (
                        <div>
                          <AiOutlineCloseCircle className='close-button' onClick = {()
=> openAccount(account = !account)} />
                          <b className='name'>{username}</b> <br /> <br />
                          <b>Додаткова інформація:</b> <br /> <br />
                          Телефон:<br />
                          {showInfo(phone)}
                          <br /> <br />
                          Місто:<br />
                          {showInfo(city)}
                          <br /> <br />
                          Пошта:<br />
                          {showInfo(postal)}
                          <br /> <br />
                          Дата народження:<br />
                          {showInfo(date)}
                          <br /> <br />

                          <form className='red'
action={`http://localhost/getinfo.php?userId=${getCookie('user_id')}`} method='post'>

                            <b>Доповнення інформації</b> <br /> <br />

                            <input type="text" name="phone" placeholder='Введіть номер
вашого телефону' />

                            <br /> <br />

                            <input type="text" name="city" placeholder='Введіть назву
вашого міста' />

                            <br /> <br />

                            <input type="text" name="postal" placeholder='Введіть ваш
поштовий індекс' />
```

```
                            <br /> <br />

                            <input type="date" name="date" placeholder='Введіть дату
вашого народження' />

                            <br /> <br />

                            <input className='submit' type="submit" value="Редагувати"
/>

                          </form>
                          <b className='youror'>Ваші покупки:</b>
                          <FinishedOrders userItems={userItems}/>
                        </div>
                      )}

                  </div>
                  )}
            </ul>

        </div>
        <div className='presentation'></div>
    </header>
  )
}

import React, { Component } from 'react'

export class Item extends Component {
  render() {
    return (
      <div className='item'>

        <img src={"./img/" + this.props.item.img} onClick={() =>
this.props.onShowInfo(this.props.item)} />
        <h2>{this.props.item.title}</h2>
        <b>{this.props.item.price}₴</b>
        <div className='add-to-cart' onClick={() =>
this.props.onAdd(this.props.item)}>До кошику</div>
      </div>
    )
  }
}

export default Item

import React, { Component } from 'react'
import Item from './Item'

export class Items extends Component {
  render() {
    return (
      <main>
        {this.props.allitems.map(el => (
            <Item onShowInfo={this.props.onShowInfo} key={el.id} item={el}
onAdd={this.props.onAdd}/>
        ))}
      </main>
    )
  }
}
```

```
export default Items

import React from 'react';
import { AiOutlineCloseCircle } from 'react-icons/ai';

export default function LoginForm(props) {
  return (
    <div className='login'>
      <div>
        <AiOutlineCloseCircle className='close-button' onClick={props.onClose} />
        <b>Авторизація</b> <br /><br />
        <form action='http://localhost/login.php' method='post'>

          <input type="text" name="login" placeholder='Введіть ваш логін' required/>

          <br /> <br />

          <input type="password" name="password" placeholder='Введіть ваш пароль'
required/>

          <br /> <br />

          <input className='submit' type="submit" value="Авторизуватися" />

        </form>
      </div>
    </div>
  );
}

import React, { Component } from 'react'
import { TbTrash } from 'react-icons/tb'

export class Order extends Component {
  render() {
    return (
      <div className='item'>
        <img src={"./img/" + this.props.item.img}/>
        <h2>{this.props.item.title}</h2>
        <b>{this.props.item.price}₴</b>
        <TbTrash className='delete-icon' onClick={() =>
this.props.onDelete(this.props.item.id)}/>
      </div>
    )
  }
}

export default Order

import React from 'react';
import { AiOutlineCloseCircle } from 'react-icons/ai';

export default function RegistrationForm(props) {
  return (
    <div className='reg'>
        <div>
            <AiOutlineCloseCircle className='close-button' onClick={props.onClose} />
            <b>Реєстрація</b> <br></br> <br></br>
            <form action='http://localhost/register.php' method='post'>
```

```
                <input type="text" name="name" minLength="2" placeholder='Введіть ваше
ім`я' required/>

                <br /> <br />

                <input type="text" name="login" placeholder='Введіть ваш логін'
required/>

                <br /> <br />

                <input type="password" name="password" placeholder='Введіть ваш пароль'
required/>

                <br /> <br />

                <input className='submit' type="submit" value="Зареєструвати" />

            </form>
        </div>
    </div>
  );
}

import React, { Component } from 'react'
export class ShowInfo extends Component {
  render() {
    return (
      <div className='info'>
        <div>
            <img src={"./img/" + this.props.item.img} onClick={() =>
this.props.onShowInfo(this.props.item)} />
            <h2>{this.props.item.title}</h2>
            <p>{this.props.item.description}</p>
            <b>{this.props.item.price}₴</b>
            <div className='add-to-cart' onClick={() =>
this.props.onAdd(this.props.item)}>До кошику</div>
        </div>
      </div>
    )
  }
}

export default ShowInfo

import React, { useState, useEffect } from 'react';
import LoginForm from './LoginForm';
import RegistrationForm from './RegistrationForm';
import Balance from './Balance';
import AddBalance from './AddBalance';
import Order from './Order';
import { FaShoppingCart } from "react-icons/fa";


export default function User(props) {
  const [loginOn, openLogin] = useState(false);
  const [registerOn, openRegister] = useState(false);
  const [username, setUsername] = useState('');
  const [userBalance, setUserBalance] = useState(0);
  const [balanceToAdd, setBalanceToAdd] = useState(0);
  const [showAddBalanceForm, setShowAddBalanceForm] = useState(false);
  const [userId, setUserId] = useState(null);
  let [cartOpen, setCartOpen] = useState(false);
```

```
  let { orders, userItems } = props;
useEffect(() => {
    const savedUserId = getCookie('user_id');
    const savedUsername = localStorage.getItem('username');
    const savedBalance = localStorage.getItem('userBalance');

    if (savedUserId) {
      fetch(`http://localhost/getuser.php?id=${savedUserId}`)
        .then(response => response.json())
        .then(data => {
          const userBalance = parseInt(data.balance, 10);
          const username = data.name;
          setUserBalance(userBalance);
          setUsername(username);
          setUserId(savedUserId);
          localStorage.setItem('username', username);
          localStorage.setItem('userBalance', userBalance.toString());
        })
        .catch(() => {
          if (savedBalance) {
            setUserBalance(parseInt(savedBalance));
          }
        });
    } else if (savedUsername) {
      setUsername(savedUsername);
      if (savedBalance) {
        setUserBalance(parseInt(savedBalance));
      }
    }
  }, []);
 useEffect(() => {
    if (userBalance !== null && userId) {
      fetch(`http://localhost/updatebalance.php?id=${userId}&balance=${userBalance}`)
        .then(response => response.json())
        .then(data => {
          console.log('Balance updated:', data);
        })
        .catch(error => {
          console.error('Error updating balance:', error);
        });
      localStorage.setItem('userBalance', userBalance.toString());
    }
  }, [userBalance, userId]);


  const handleLoginClose = () => {
    openLogin(false);
  };

  const handleRegisterClose = () => {
    openRegister(false);
  };
  const logout = () => {
    Promise.all([
      fetch('http://localhost/exit.php'),
      setUsername(''),
      document.cookie = 'user_id=;expires=Thu, 01 Jan 1970 00:00:01 GMT;',
      localStorage.clear(),
    ]);
  };

  const handleAddBalanceFormClose = () => {
```

```
  setShowAddBalanceForm(false);
};

const handleBalanceInputChange = (event) => {
  setBalanceToAdd(Number(event.target.value));
};

const handleAddBalance = (newBalance) => {
  const updatedBalance = userBalance + newBalance;
  setUserBalance(updatedBalance);
  localStorage.setItem('userBalance', updatedBalance.toString());
  setShowAddBalanceForm(false);
};

function getCookie(name) {
  const cookies = document.cookie.split(';');
  for (let i = 0; i < cookies.length; i++) {
    const cookie = cookies[i].trim();
    if (cookie.startsWith(name + '=')) {
      return cookie.substring(name.length + 1);
    }
  }
  return null;
}

const buyAllInCart = (summ) => {
  if (userBalance >= summ){
    const updatedBalance = userBalance - summ;
    setUserBalance(updatedBalance);
    localStorage.setItem('userBalance', updatedBalance.toString());
    for (var i = 0; i < orders.length; i++) {
      userItems.push(orders[i]);
    }
    orders = [];
    props.orders.length = 0;
  }
  else{
    alert('Недостатньо коштів!')
  }
};

const showOrders = (props) => {
  let summ = 0
  props.orders.forEach(el => summ += Number.parseFloat(el.price))
  return (<div>
    {props.orders.map(el =>(
    <Order onDelete={props.onDelete} key={el.id} item={el} />
    ))}
    <p className='summ'>У кошику: {new Intl.NumberFormat().format(summ)}₴</p>
    <div className='buy-button' onClick={() => buyAllInCart(summ)}>Придбати</div>
  </div>)
}



const showNothing = () => {
  return (<div className='empty'>
    <h2>Кошик порожній</h2>
  </div>

  )
}
```

```
  return (
    <div>
      {username ? (

      <div>

      {showAddBalanceForm ? (
            <AddBalance onClose={handleAddBalanceFormClose}
onAddBalance={handleAddBalance} onBalanceChange={handleBalanceInputChange} />
      ) : (
        <ul className='user'>
          <li>Привіт, {username}!</li>
          <li className='exit' onClick={() => setShowAddBalanceForm(true)}>Ваш баланс:
<Balance userBalance={userBalance}/> ₴</li>
          <li><FaShoppingCart onClick={() => setCartOpen(cartOpen = !cartOpen)}
className={`shop-cart-button ${cartOpen ? 'active' : ''}`} /></li>
          <li className='exit' onClick={logout}>Вийти</li>
        </ul>
      )}
      {cartOpen && (
              <div className={`shop-cart ${cartOpen ? 'show' : ''}`}>
                {props.orders.length > 0 ?
                  showOrders(props) : showNothing()
                }
              </div>
            )}
    </div>
      ) : (
        <ul className='log'>
          <li onClick={() => openLogin(!loginOn)}>Увійти</li>
          {loginOn && <LoginForm onClose={handleLoginClose} />}
          <li onClick={() => openRegister(!registerOn)}>Зареєструватися</li>
          {registerOn && <RegistrationForm onClose={handleRegisterClose} />}
        </ul>
      )}
    </div>
  );
}
```

```
<?php

    header("Access-Control-Allow-Origin: *");

    $host = 'localhost';
    $username = 'root';
    $password = '';
    $dbname = 'my-shop';
    $conn = mysqli_connect($host, $username, $password, $dbname);

    if (!$conn) {
        die("Connection failed: " . mysqli_connect_error());
    }

    $title = mysqli_real_escape_string($conn, $_POST['title']);
    $description = mysqli_real_escape_string($conn, $_POST['description']);
    $price = mysqli_real_escape_string($conn, $_POST['price']);
    $img = mysqli_real_escape_string($conn, $_POST['img']);
    $category = mysqli_real_escape_string($conn, $_POST['category']);

    if (!$img){
        $img = 'default.png';
```

```
    }


    $sql = "INSERT INTO items (title, description, price, img, category) VALUES
('$title', '$description', '$price', '$img', '$category')";

    if (mysqli_query($conn, $sql)) {
        echo "Данные успешно добавлены в таблицу";
    } else {
        echo "Ошибка: " . $sql . "<br>" . mysqli_error($conn);
    }

    mysqli_close($conn);

    header('Location: http://localhost:3000');

    exit();

?>

<?php

header("Access-Control-Allow-Origin: *");

$title = filter_var(trim($_POST['title']), FILTER_SANITIZE_STRING);

$mysqli = new mysqli("localhost", "root", "", "my-shop");
if ($mysqli->connect_error) {
    die("Error: failed to connect to the database");
}

$stmt = $mysqli->prepare("DELETE FROM items WHERE title=?");
if (!$stmt) {
    die("Error: " . $mysqli->errno . " - " . $mysqli->error);
}

$stmt->bind_param("s", $title);
$stmt->execute();

$stmt->close();
$mysqli->close();

header('Location: http://localhost:3000');

exit();

?>

<?php

    header("Access-Control-Allow-Origin: *");

    setcookie('user_id', $user['id'], time() - 3600, "/");

    header('Location: http://localhost:3000');

    exit();

?>
. . .
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ Крикля.docx	Пояснювальна записка до дипломного проекту. Документ Word.
Диплом_ Крикля.pdf	Пояснювальна записка до дипломного проекту в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і відкомпільовану програму
Презентація	
Презентація_Крикля.ppt	Презентація дипломного проекту