Вищий навчальний заклад

«Університет економіки та права «КРОК»
Фаховий коледж

Циклова комісія з інформаційних технологій

Кваліфікаційна робота фахового молодшого бакалавра
на тему Розробка інтернет-магазину з продажу атрибутики для комп'ютерних ігор

Виконав_____
(Підпис)

_____Бичков Лев Генадійович_____
(прізвище, ім'я, по батькові)

_____

Науковий керівник_____
_____Пантєєв Роман Леонідович_____
(прізвище, ім'я, по батькові)

_____
_____
(Резолюція «До захисту»)

Попередній захист:

_____
___
(Висновок: "До захисту в екзаменаційній комісії ")

Голова циклової комісії _____ _____ _____
(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2025 року

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

Фаховий коледж

Циклова комісія з інформаційних технологій

Спеціальність 121 інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Голова циклової комісії _____ Леонід УВАРОВ

(підпис)

«____» _____ 2025 року

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач освіти Бичков Лев Генадійович

Тема роботи Розробка інтернет-магазину з продажу атрибутики для комп’ютерних ігор

1. від «___» _____ 202__ р. № _____

2. Термін здачі закінченої роботи «30» травня 2025 року

3. Вихідні дані до роботи:

1) Провести аналіз ринку та наявних рішень у сфері онлайн-торгівлі ігровою атрибутикою.

2) Сформувати функціональні та нефункціональні вимоги до майбутньої системи.

3) Розробити архітектуру інтернет-магазину (структура бази даних, модулі, API).

4) Створити дизайн користувацького інтерфейсу з урахуванням принципів UX/UI.

5) Реалізувати веб-додаток з використанням сучасних технологій фронтенду та бекенду

6) Здійснити інтеграцію платіжної системи та засобів авторизації/реєстрації користувачів.

7) Забезпечити захист персональних даних та відповідність базовим вимогам кібербезпеки.

8) Провести тестування функціональності, продуктивності та зручності користування.

4. Зміст пояснювальної записки:

1) Розділ 1. Аналіз предметної області

Вивчення існуючих рішень, аналіз конкурентів та визначення ключових вимог і можливостей для розробки інтернет-магазину.

2) Розділ 2. Постановка завдань і вимог

Формулювання цілей проекту, визначення функціональних та нефункціональних вимог до майбутнього інтернет-магазину.

3) Розділ 3. Розробка дизайну користувацького інтерфейсу

Створення зручного та привабливого інтерфейсу, що враховує потреби користувачів і сприяє легкому навігаційному досвіду.

4) Розділ 4. Реалізація інтернет-магазину

Програмна розробка та впровадження всіх функцій інтернет-магазину відповідно до технічних вимог.

5) Розділ 5. Забезпечення безпеки та захисту даних

Впровадження сучасних засобів захисту даних користувачів і забезпечення безпеки транзакцій.

6) Розділ 6. Тестування та оцінка якості

Перевірка функціональності сайту, пошук і виправлення помилок, а також оцінка загальної якості продукту.

5. **Перелік графічного матеріалу:**

- Скріншоти інтерфейсу сайту
- Фрагменти коду та конфігураційні файли

Дата видачі завдання «12» лютого 2025 року

Науковий керівник _____ _Пантєєв Р. Л._

(підпис) (прізвище, ім'я, по батькові)

Завдання прийняв до виконання _____ __Бичков Л.Г.

(підпис) (прізвище, ім'я, по батькові)

РЕФЕРАТ

Пояснювальна записка: 57 сторінок, 2 додатків, 16 джерел.

Об'єкт дослідження: Об’єктом дослідження у кваліфікаційній роботі є процеси розробки та впровадження веб-додатку інтернет-магазину для роздрібної торгівлі. Зокрема, досліджуються аспекти проєктування архітектури клієнтської частини, інтеграції зі сторонніми сервісами (зокрема, платіжними системами та сервісами аутентифікації), а також особливості забезпечення безпеки та захисту даних у безсерверних веб-рішеннях.

Мета роботи: Розробити функціональний інтернет-магазин, призначений для реалізації атрибутики, пов’язаної з комп’ютерними іграми, з урахуванням вимог сучасної веб-розробки, безпеки, зручності користування та адаптивності інтерфейсу.

У кваліфікаційній роботі проведено аналіз сучасних рішень у сфері електронної комерції та визначено актуальні тенденції розвитку. Було розроблено односторінковий веб-додаток інтернет-магазину, що включає каталог товарів, кошик покупок, функціонал списку бажань та фільтрацію товарів за ціною. Реалізація здійснена з використанням HTML, CSS та JavaScript, без серверної частини, зберігання даних відбувається у LocalStorage. Надано інструкції для користувачів, описано архітектуру додатку та особливості інтеграції платіжної системи (Stripe, PayPal) для безпечної оплати товарів. Також розглянуто аспекти захисту персональних даних і проведено тестування зручності та продуктивності інтерфейсу.

Результати розробки можуть бути повноцінним інтернет магазином для продажу та реклами.

Ключові слова: Сайт, Інтернет магазин, HTML, атрибутика для ігор.

ABSTRACT

Explanatory note: 56 pages, 2 appendices, 16 sources.

Object of research: The object of research in this qualification work is the processes of developing and implementing a web application for an online store focused on retail sales. In particular, it studies the aspects of designing the client-side architecture, integrating third-party services (such as payment systems and authentication services), and ensuring data security and protection in serverless web solutions.

Purpose of the work: To develop a functional online store for selling merchandise related to computer games, considering modern web development standards, security, user convenience, and adaptive interface design.

The qualification work includes an analysis of modern solutions in the field of e-commerce and identifies current development trends. A single-page web application was developed that includes a product catalog, shopping cart, wishlist functionality, and product price filtering. The implementation is carried out using HTML, CSS, and JavaScript, without a server component, and data is stored in LocalStorage. User instructions are provided, the application architecture is described, and the specifics

of integrating payment systems (Stripe, PayPal) for secure payment processing are outlined. Additionally, the aspects of protecting personal data are considered, and testing of the interface's usability and performance has been carried out.
The results of the development can be a fully functional online store for sales and marketing.

Keywords: Website, Online store, HTML, Game merchandise.

ЗМІСТ

ВСТУП

У сучасному світі стрімко зростає популярність комп’ютерних ігор, що обумовлено розвитком новітніх технологій, зростанням доступності інтернету та загальною діджиталізацією суспільства. Ігрова індустрія перетворилася з вузькоспеціалізованої сфери у величезний ринок з мільярдними оборотами та постійно зростаючою кількістю користувачів. У зв’язку з цим паралельно розвивається індустрія супутньої продукції - атрибутики для геймерів, яка включає широкий спектр товарів: одяг, аксесуари, предмети колекціонування, сувенірну продукцію тощо.

Інтернет-магазини стали одним із основних інструментів реалізації подібної атрибутики. Вони дозволяють не лише значно розширити географію продажів, а й забезпечити швидкий доступ до продукції для цільової аудиторії, створюючи зручний та інтуїтивно зрозумілий інтерфейс для покупців. Саме створення ефективного веб-додатку для продажу ігрової атрибутики є актуальною задачею в умовах сучасного ринку.

Метою цієї дипломної роботи є **розробка функціонального інтернет-магазину**, орієнтованого на продаж атрибутики, пов’язаної з комп’ютерними іграми. Для досягнення цієї мети необхідно виконати низку завдань, зокрема: провести детальний аналіз ринку та конкурентних рішень, визначити функціональні та нефункціональні вимоги до системи, розробити архітектуру веб-додатку та спроектувати зручний і привабливий інтерфейс користувача з урахуванням принципів UX/UI дизайну.

Важливими аспектами розробки інтернет-магазину є забезпечення безпеки персональних даних користувачів та відповідність базовим вимогам кібербезпеки. Не менш значущим є тестування створеного веб-додатку з точки зору продуктивності та зручності користування, що дозволить гарантувати високу якість роботи системи та задоволеність клієнтів.

Актуальність даної роботи полягає у необхідності створення зручного, надійного та естетично привабливого веб-сервісу, здатного задовольнити

потреби сучасних споживачів, які шукають атрибутику для комп'ютерних ігор. Отже, результати дипломної роботи можуть мати практичне застосування у сфері електронної комерції та слугувати платформою для подальшого розвитку проєктів, спрямованих на задоволення потреб геймерської спільноти.

Крім того, у ході роботи передбачається дослідження сучасних технологій веб-розробки, таких як HTML5, CSS3, JavaScript, а також фреймворків і бібліотек для фронтенду та бекенду (наприклад, React, Vue.js, Node.js). Це дозволить створити адаптивний та швидкий веб-застосунок, який відповідатиме високим вимогам сучасних користувачів.

Також важливо врахувати питання інтеграції платіжних систем та розробки зручної системи авторизації та реєстрації користувачів. Сучасні тенденції веб-розробки зосереджені не лише на зручності користувача, а й на високому рівні захисту даних, що є критично важливим для комерційних проєктів. Таким чином, ця дипломна робота передбачає всебічне опрацювання теми - від етапу планування та проектування до кінцевої реалізації та тестування готового продукту.

собливу увагу в дипломній роботі буде приділено аналізу сучасних конкурентних рішень. Це дозволить визначити найбільш затребувані функції та інтерфейсні рішення, що актуальні для користувачів у цій сфері. Аналіз наявних рішень також допоможе уникнути поширених помилок та обрати найкращі практики для реалізації веб-додатку. Таке дослідження дозволить не тільки краще зрозуміти очікування користувачів, а й закласти основу для розробки конкурентоспроможного продукту.

Під час розробки інтернет-магазину буде здійснено побудову логічної архітектури проєкту. Це включає проектування бази даних, визначення основних модулів та компонентів системи, а також підготовку API для взаємодії фронтенду та бекенду. Такий підхід дозволить забезпечити масштабованість та гнучкість майбутньої системи, що є важливими критеріями для довгострокового розвитку та обслуговування веб-додатку.

У процесі створення інтернет-магазину значна роль відведена реалізації зручного користувацького інтерфейсу, що ґрунтується на сучасних принципах UX/UI дизайну. Це дозволить зробити веб-застосунок інтуїтивно зрозумілим для відвідувачів, а також підвищити рівень задоволення користувачів та їх лояльність. Важливо, щоб інтерфейс був не лише візуально привабливим, а й функціональним, з оптимальним розміщенням основних елементів.

Крім того, дипломна робота передбачає дослідження аспектів забезпечення безпеки веб-додатку, адже захист персональних даних користувачів є ключовою вимогою для будь-якого сучасного інтернет-магазину. Будуть розглянуті базові принципи безпеки, такі як шифрування даних, захист від атак типу CSRF та XSS, а також інші заходи для запобігання несанкціонованого доступу.

Таким чином, об’єктом дослідження дипломної роботи є інтернет-магазин з продажу атрибутики для комп’ютерних ігор, а предметом дослідження — процеси розробки, інтеграції, тестування та захисту веб-додатку. Усі етапи виконання проєкту будуть ретельно задокументовані та супроводжуватимуться пояснювальною запискою, яка описує теоретичну базу та практичні аспекти розробки.

Реалізація цього дипломного проєкту дозволить продемонструвати володіння сучасними технологіями веб-розробки, а також вміння застосовувати набуті знання у практичних завданнях, що є надзвичайно важливо для майбутньої професійної діяльності.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасні тенденції онлайн-торгівлі

Онлайн-торгівля (електронна комерція) за останні роки стала однією з найдинамічніших і найшвидше зростаючих сфер сучасної економіки. Завдяки стрімкому розвитку інтернет-технологій, збільшенню проникнення смартфонів та широкосмугового інтернету, все більше споживачів віддають перевагу покупкам через мережу. Це не лише забезпечує зручність і швидкість, а й відкриває нові можливості для бізнесу.

Однією з ключових тенденцій онлайн-торгівлі є зростання мобільної комерції (m-commerce). Зараз більшість користувачів здійснюють покупки зі смартфонів, що змушує власників сайтів адаптувати свої ресурси під мобільні пристрої. Мобільна оптимізація включає швидкість завантаження, зручність інтерфейсу та інтеграцію з мобільними платіжними системами. Це дозволяє підвищити рівень конверсії та утримати увагу клієнтів.

Важливою тенденцією є персоналізація контенту. Завдяки штучному інтелекту та алгоритмам машинного навчання, онлайн-магазини можуть пропонувати користувачам індивідуальні рекомендації товарів на основі їхньої поведінки, історії переглядів і покупок. Це підвищує рівень задоволення клієнтів і стимулює повторні продажі.

Активно розвивається соціальна комерція - продаж товарів безпосередньо через соціальні мережі, такі як Instagram, Facebook, TikTok. Це дозволяє брендам швидко охоплювати широку аудиторію, використовувати впливових блогерів і створювати інтерактивні рекламні кампанії.

Ще одна важлива складова - логістика і доставка. Сучасні споживачі очікують швидкої, надійної та, бажано, безкоштовної доставки. Це стає одним із основних факторів вибору інтернет-магазину. Розвиток автоматизації складських процесів і використання сервісів кур’єрської доставки сприяє підвищенню рівня обслуговування клієнтів.

Способи оплати також зазнають змін: поряд з традиційними банківськими картками популярні електронні гаманці (PayPal, Apple Pay, Google Pay), мобільні платежі, а також криптовалюти. Безпека фінансових операцій - один із ключових факторів довіри покупців.

В останні роки в онлайн-торгівлю активно впроваджуються технології доповненої (AR) та віртуальної реальності (VR), які дозволяють клієнтам віртуально «приміряти» одяг, розглянути товар у 3D або переглянути, як меблі виглядатимуть у приміщенні. Це значно покращує користувацький досвід і знижує ризик повернень.

Разом з перевагами, онлайн-торгівля стикається і з низкою викликів. Серед них - забезпечення безпеки персональних даних, боротьба з шахрайством, підтримка високого рівня обслуговування клієнтів, конкурентна боротьба в умовах великої кількості гравців на ринку.

За даними статистики, обсяг світового ринку електронної комерції щорічно зростає в середньому на 15-20%. В Україні також спостерігається значний розвиток цієї сфери - за останні п’ять років кількість покупців в інтернеті зросла вдвічі. Найпопулярнішими категоріями товарів є електроніка, одяг, косметика, продукти харчування.

Отже, сучасні тенденції онлайн-торгівлі диктують нові вимоги до веб-ресурсів, роблячи акцент на зручності, персоналізації, мобільності та безпеці. Врахування цих факторів є критично важливим при розробці власного інтернет-магазину або сайту, що займається продажем товарів чи послуг.

1.2 Аналіз конкурентів та існуючих рішень.

У сучасних умовах розвитку електронної комерції конкурентний аналіз є одним із ключових етапів при створенні власного інтернет-магазину. Він дозволяє визначити ринкові стандарти, оцінити сильні та слабкі сторони існуючих платформ, виявити нішеві можливості та сформувати унікальні пропозиції для цільової аудиторії.

Для більш глибокого розуміння ринку, розглянемо декілька відомих прикладів, які користуються популярністю серед споживачів, зокрема Fanatical та J!NX, а також коротко згадаємо інших гравців у цій сфері.

Fanatical - міжнародний онлайн-магазин цифрових ігор і програмного забезпечення. Він орієнтований на геймерів і технічних користувачів, пропонуючи великий каталог товарів за конкурентними цінами.

Основні характеристики Fanatical:

- **Великий асортимент:** магазин пропонує тисячі різних ігор і програм, включаючи новинки і класичні хіти.
- **Гнучкі фільтри і пошук:** дозволяють швидко знаходити потрібні товари за жанром, платформою, рейтингом тощо.
- **Безпека:** Fanatical використовує сучасні протоколи захисту даних та платежів, що підвищує довіру користувачів.
- **Система знижок та акцій:** постійні розпродажі, пакетні пропозиції і промокоди стимулюють продажі та залучають нових клієнтів.
- **Зручність користування:** сайт має простий і зрозумілий інтерфейс, швидке завантаження сторінок і адаптивний дизайн для мобільних пристроїв.

Таким чином, Fanatical є прикладом добре налагодженої платформи, що успішно поєднує широкий вибір товарів, безпеку і зручність.

J!NX - інтернет-магазин, який спеціалізується на продажу одягу, аксесуарів і сувенірної продукції для геймерів і фанатів поп-культури. Магазин має яскравий брендовий стиль, що чітко відображає інтереси цільової аудиторії.

Ключові особливості J!NX:

- **Унікальний дизайн:** сайт і товари орієнтовані на фанатів ігор, коміксів та кіно, що робить бренд впізнаваним і затребуваним.
- **Активна маркетингова стратегія:** J!NX активно використовує соціальні мережі, співпрацює з блогерами та проводить конкурси, що підвищує лояльність клієнтів.
- **Інтуїтивний інтерфейс:** простота оформлення замовлень, наявність докладних описів товарів і високоякісних фото.

- **Персоналізація:** пропозиції товарів враховують інтереси користувача, що збільшує ймовірність покупки.

J!NX демонструє, як можна ефективно працювати в нішевому сегменті, використовуючи сильний бренд і активну взаємодію з аудиторією.

На ринку онлайн-торгівлі існує безліч інших платформ, які можна розглядати як конкуренти або джерела ідей:

- **Amazon:** найбільший у світі маркетплейс з величезним асортиментом товарів, розвиненою логістикою і системою відгуків. Відрізняється високою надійністю і широкими можливостями для продавців.
- **eBay:** популярний майданчик для продажу нових і вживаних товарів з можливістю проведення аукціонів.
- **Etsy:** платформа для продажу унікальних хендмейд товарів і творчих виробів, орієнтована на творчу аудиторію.

При оцінці конкурентів важливо враховувати такі аспекти:

- **Функціональність сайту:** пошук, фільтри, сортування, процес оформлення замовлення, можливість порівняння товарів.
- **Дизайн і юзабіліті:** зручність навігації, адаптивність під різні пристрої, привабливість інтерфейсу.
- **Технології:** використання сучасних рішень, швидкість роботи сайту, інтеграції з платіжними системами, системами обробки замовлень і логістики.
- **Маркетингові інструменти:** персоналізація пропозицій, рекламні акції, програми лояльності.
- **Підтримка клієнтів:** якість і швидкість обслуговування, наявність онлайн-консультантів, FAQ.
- **Безпека:** захист персональних даних і фінансових транзакцій.

1.3 Проблеми та недоліки існуючих сайтів

Незважаючи на значне зростання та розвиток електронної комерції, багато існуючих інтернет-магазинів стикаються з рядом проблем, які знижують їхню

ефективність та погіршують досвід користувачів. Однією з ключових проблем є відсутність або недостатня локалізація. Багато сайтів пропонують інтерфейс лише однією чи кількома іноземними мовами, що створює мовний бар'єр для частини аудиторії. Відсутність підтримки місцевих валют, способів оплати та умов доставки обмежує зручність користування та зменшує довіру потенційних клієнтів. Непрофесійний переклад і ігнорування культурних особливостей регіону також негативно впливають на сприйняття ресурсу та знижують його привабливість для локального ринку.

Крім того, суттєвим недоліком багатьох інтернет-магазинів є незадовільний користувацький досвід (UX). Складна й заплутана навігація, погано структуроване меню, неефективний пошук товарів змушують користувачів витрачати багато часу на пошук потрібного продукту, що призводить до розчарування й відходу з сайту. Процес оформлення замовлення часто занадто довгий, містить багато обов'язкових полів, вимагає обов'язкової реєстрації або не надає можливості зберегти кошик — це викликає негативні емоції та знижує конверсію. Важливою проблемою є відсутність адаптивного дизайну: сайти, які погано оптимізовані під мобільні пристрої, втрачають значну частину мобільних покупців. Також часто зустрічається низька якість візуального контенту - фотографії товарів низької роздільної здатності, відсутність демонстраційних відео чи оглядів знижують довіру користувачів. Ще один важливий момент - відсутність персоналізації, коли сайт не аналізує поведінку користувачів і не пропонує індивідуальні рекомендації, що обмежує можливості для повторних продажів. До того ж повільне завантаження сторінок суттєво впливає на зручність користування та може призвести до значного відтоку клієнтів.

Безпека є ще однією слабкою стороною багатьох онлайн-платформ. Відсутність надійного шифрування даних через відсутність SSL-сертифікатів ставить під загрозу конфіденційність інформації користувачів і їхні фінансові транзакції. Недостатньо надійна автентифікація, відсутність двофакторної перевірки створюють ризик злому акаунтів. Крім того, багато сайтів є

вразливими до різних кібератак, таких як SQL-ін’єкції, міжсайтовий скриптинг та інші, що може призвести до витоку даних або порушення роботи ресурсу. Деякі платформи неправильно зберігають персональні дані, не дотримуються вимог законодавства у сфері захисту інформації, що може спричинити юридичні проблеми. Помилки при інтеграції з платіжними системами збільшують ризик шахрайства та фінансових втрат.

Додатково варто зазначити, що багато сайтів недостатньо інтегровані з соціальними мережами, що обмежує маркетингові можливості та взаємодію з аудиторією. Слабка служба підтримки клієнтів — відсутність оперативної реакції, консультацій або зручних способів зв’язку — також негативно впливає на репутацію магазину. Брак аналітичних інструментів ускладнює розуміння поведінки користувачів та оптимізацію роботи сайту. У деяких випадках технічна база інтернет-магазинів не дозволяє масштабувати бізнес без втрати продуктивності, що стримує зростання та розвиток.

У підсумку, проблеми з локалізацією, незручний користувацький інтерфейс і недостатній рівень безпеки суттєво обмежують потенціал існуючих онлайн-майданчиків. Для створення успішного, конкурентоспроможного та надійного інтернет-магазину надзвичайно важливо комплексно вирішувати ці питання, впроваджувати сучасні технології, орієнтуватися на потреби користувачів і забезпечувати високий рівень захисту даних.

РОЗДІЛ 2. ПОСТАНОВКА ЗАВДАНЬ ТА ВИМОГ

2.1 Функціональні вимоги до системи

Функціональні вимоги визначають набір основних можливостей, які повинна мати розроблювана система - інтернет-магазин. Вони формують основу для розробки та тестування, а також допомагають забезпечити відповідність продукту очікуванням користувачів і бізнес-потребам. Враховуючи сучасні тенденції та особливості безсерверної архітектури, важливо детально прописати функції, які мають бути реалізовані.

Каталог є серцем будь-якого інтернет-магазину, тому його функціональність повинна бути максимально повною та зручною. Каталог повинен підтримувати багаторівневу ієрархію категорій для логічного структурування товарів, що допомагає користувачам швидко знаходити потрібні позиції. Кожен товар у каталозі має мати детальний опис, який включає назву, характеристики (розмір, колір, матеріал тощо), фотографії високої якості, ціну, наявність на складі, а також інформацію про акції або знижки. Для полегшення навігації необхідно реалізувати функції сортування товарів за різними критеріями (ціна, популярність, рейтинг, дата додавання), а також фільтрацію за параметрами (розмір, бренд, ціна тощо).

Кошик - це тимчасове сховище для товарів, які користувач планує придбати. Його функціональність має включати можливість додавання товарів із каталогу, змінювання кількості одиниць кожного товару, видалення позицій, а також підрахунок загальної вартості замовлення з урахуванням знижок та акцій. Для зручності користувачів, особливо зареєстрованих, важливо зберігати стан кошика між сесіями, щоб покупець міг повернутися і завершити покупку пізніше. Інтерфейс кошика повинен бути простим, інтуїтивно зрозумілим і адаптивним під різні пристрої.

Реалізація системи реєстрації та авторизації користувачів є ключовою для персоналізації сервісу і управління замовленнями. Користувачі повинні мати можливість створити обліковий запис, вказавши основні дані: ім’я, електронну

пошту, пароль. Для спрощення входу слід передбачити авторизацію через соціальні мережі (Google, Facebook, Apple ID) або зовнішні сервіси. Після входу користувач отримує доступ до особистого кабінету, де можна переглядати історію замовлень, змінювати профільні дані, керувати адресами доставки та налаштовувати сповіщення. Важливо також реалізувати механізми відновлення паролю і підтвердження електронної пошти для підвищення безпеки.

Процес оформлення замовлення має бути простим, зрозумілим і мінімально тривалим, щоб не втрачати потенційних покупців. Після додавання товарів у кошик користувач переходить до оформлення замовлення, де обирає спосіб доставки (кур'єр, пошта, самовивіз), вводить контактні дані та адресу, а також може залишити додаткові коментарі. Важливо реалізувати перевірку коректності введеної інформації, щоб уникнути помилок при доставці. Система має відображати підсумкову вартість із урахуванням усіх зборів і податків перед підтвердженням.

Для забезпечення безперебійної роботи інтернет-магазину необхідно інтегрувати платіжні системи, що підтримують безпечні та зручні способи оплати. Це можуть бути платіжні шлюзи, такі як Stripe, PayPal, а також локальні сервіси. Система має підтримувати оплату кредитними та дебетовими картками, електронними гаманцями, а також, за необхідності, оплату при отриманні. Під час платежу має забезпечуватись шифрування даних, а користувач має отримувати підтвердження про успішну оплату. Важливо також передбачити обробку відмовлених платежів і можливість повторної спроби.

Для підтримки актуальності сайту та контролю над бізнес-процесами необхідно створити зручний інтерфейс адміністратора. Адміністратор повинен мати можливість додавати, редагувати та видаляти товари, управляти категоріями, керувати цінами та акціями. Крім того, він має отримувати доступ до списку замовлень із можливістю змінювати їхній статус (обробка, відвантажено, доставлено), а також керувати користувачами (блокування, редагування профілів). Всі дії адміністратора повинні бути безпечними і захищеними від несанкціонованого доступу.

Для зручності користувачів необхідно реалізувати ефективний механізм пошуку товарів по ключових словах, категоріях, брендах та інших параметрах. Пошукова система має підтримувати автозаповнення, підказки та корекцію помилок, що значно спрощує процес пошуку і підвищує задоволеність клієнтів.

Важливим елементом, що впливає на вибір товару, є можливість залишати відгуки та оцінки. Система має надати користувачам інструменти для публікації відгуків з текстом і зірковим рейтингом, а також механізми модерування для запобігання спаму і некоректних коментарів. Відгуки сприяють підвищенню довіри до магазину і стимулюють активність клієнтів.

Для підтримки ефективного зв’язку з користувачами система повинна надсилати сповіщення про зміни статусу замовлень, акції, нові надходження та інші важливі події. Сповіщення можуть надходити у вигляді email-листів, SMS-повідомлень або push-повідомлень у браузері. Це допомагає утримувати клієнтів і підвищувати рівень їхньої лояльності.

Таким чином, реалізація перерахованих функціональних вимог дозволить створити повноцінний і конкурентоспроможний інтернет-магазин, що забезпечить комфортний і безпечний досвід користувачів, а також ефективне управління для власників бізнесу.

2.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи, що впливають на її зручність, надійність, безпеку та ефективність роботи. Вони є важливими для забезпечення високого рівня користувацького досвіду, стабільності та захисту даних. Нижче описані основні нефункціональні вимоги, які ставляться до розроблюваного інтернет-магазину.

Зручність користування (Usability)

Одним із ключових аспектів успіху будь-якого веб-додатку є зручність інтерфейсу для кінцевого користувача. Сайт повинен бути інтуїтивно зрозумілим, простим у навігації, з логічною структурою і чіткими підказками.

Важливо, щоб користувач міг швидко знайти потрібний товар, легко додати його до кошика і без зусиль оформити замовлення. Інтерфейс повинен бути адаптивним, тобто однаково добре відображатися і працювати як на великих екранах десктопів, так і на мобільних пристроях. Респонсивність дизайну дозволить залучити широку аудиторію та підвищити конверсію. Крім того, час відгуку інтерфейсу має бути мінімальним, щоб не викликати роздратування користувачів.

Безпека (Security)

Безпека є критично важливою вимогою, особливо для інтернет-магазинів, які працюють з персональними даними та фінансовою інформацією користувачів. Система має забезпечувати захист даних від несанкціонованого доступу, втрати, пошкодження або викрадення. Обов'язковим є використання протоколу HTTPS для шифрування переданих даних між клієнтом і сервером або зовнішніми сервісами. Система авторизації має впроваджувати надійні механізми аутентифікації, зокрема, хешування паролів, багатофакторну автентифікацію (2FA) та обмеження кількості спроб входу для запобігання brute-force атак. Також важливо передбачити валідацію даних на клієнтській та серверній (або зовнішній) стороні для запобігання ін'єкціям, XSS-атакам та іншим загрозам. Всі сторонні бібліотеки і API мають бути перевірені на безпеку та регулярного оновлюватися.

Продуктивність (Performance)

Швидкість роботи інтернет-магазину значною мірою впливає на задоволеність користувачів і конверсію. Сайт повинен завантажуватися максимально швидко, незалежно від кількості товарів у каталозі або трафіку на ресурсі. Для цього слід оптимізувати всі ресурси: зображення, скрипти, стилі та кешування.

Особливо важливо забезпечити швидке відображення головної сторінки, каталогу та кошика, оскільки саме на цих етапах користувачі приймають рішення про покупку. Використання сучасних фронтенд-технологій, CDN, а також

асинхронне завантаження даних допоможе досягти високої продуктивності. Система повинна коректно працювати навіть при високих навантаженнях, без затримок або збоїв. Враховуючи, що сайт реалізується без власного серверного боку, важливо оптимально використовувати зовнішні сервіси та API, а також мінімізувати обсяг переданих даних.

Надійність і доступність (Reliability and Availability)

Інтернет-магазин повинен бути доступним для користувачів 24/7 із мінімальним часом простою. Важливо, щоб система не зависала і не припиняла роботу під час пікових навантажень або технічних збоїв. Для цього необхідно передбачити резервні копії даних, механізми відновлення після збоїв, а також моніторинг роботи сайту і швидке реагування на проблеми. Особливо це актуально для безсерверних рішень, де контроль над інфраструктурою частково залежить від сторонніх сервісів.

Масштабованість (Scalability)

Система повинна легко адаптуватися до зростання кількості користувачів і товарів без значного падіння продуктивності. Обрана архітектура і технології мають підтримувати швидке розширення функціоналу і збільшення обсягів даних. Це дозволить розвивати проект у майбутньому без необхідності кардинальної переробки.

Таким чином, виконання зазначених нефункціональних вимог забезпечить стабільну, безпечну та комфортну роботу інтернет-магазину, що сприятиме залученню і утриманню клієнтів, а також успішному розвитку бізнесу.

2.3 Структура веб-додатку

Архітектура веб-додатку визначає спосіб організації його складових частин та характер їх взаємодії. В умовах реалізації сайту без серверної частини структура проєкту ґрунтується на клієнтській логіці та використанні сторонніх сервісів.

Основним компонентом додатку є клієнтська частина (frontend), яка реалізована з використанням стандартних веб-технологій: HTML для структури сторінок, CSS для візуального оформлення та JavaScript для взаємодії з користувачем та реалізації логіки.

Клієнтська частина відповідає за такі функції:

- відображення каталогу товарів;
- взаємодія з кошиком, списком бажань (wishlist);
- фільтрація та сортування;
- взаємодія з локальним сховищем браузера.

Обробка чутливих або складних процесів, таких як аутентифікація або оплата, делегується зовнішнім сервісам. Наприклад, для здійснення платежів використовується API платіжної системи (Stripe або PayPal), а для авторизації – Firebase Authentication. Таким чином, додаток може функціонувати без власного серверу або бази даних.

Замість серверного зберігання даних, додаток використовує LocalStorage або IndexedDB, які дозволяють тимчасово зберігати інформацію про кошик або список бажань безпосередньо у браузері користувача.

Взаємодія між елементами додатку виглядає наступним чином:

- користувач працює з веб-інтерфейсом у браузері;
- браузер зберігає дані локально або надсилає запити до сторонніх сервісів;
- відповіді від сервісів обробляються JavaScript-кодом і відображаються на сторінці.

Така архітектура має низку переваг: швидкість розробки, простота підтримки, відсутність витрат на серверне середовище. Проте вона також має обмеження – зокрема, відсутність централізованого зберігання даних і обмежені можливості масштабування.

Отже, структура веб-додатку базується на сучасному безсерверному підході, який дозволяє створити повноцінний інтернет-магазин із базовими функціональними можливостями без складної інфраструктури.

РОЗДІЛ 3. РОЗРОБКА ДИЗАЙНУ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

3.1 UX/UI дизайн: принципи, макети, прототипи із фокусом на швидкість та простоту

UX/UI дизайн є важливою складовою успішного інтернет-магазину, оскільки він визначає, наскільки легко користувачі взаємодіють із системою, наскільки швидко вони можуть знайти потрібну інформацію та оформити замовлення. Для сайту без серверної частини (без власного бекенду) UX/UI дизайн відіграє ще більш значну роль, оскільки саме від нього залежить ефективність клієнтської частини.

Принципи UX/UI дизайну

Основними принципами UX/UI дизайну для цього проекту є:

- Простота: Інтерфейс має бути інтуїтивно зрозумілим, без перевантаження користувача зайвими елементами. Це особливо важливо для швидких SPA-рішень (Single Page Application), де важлива мінімальна кількість переходів між сторінками.
- Швидкість: Кожна взаємодія користувача з сайтом має бути максимально швидкою. Для цього необхідно мінімізувати обсяг елементів сторінки, використовувати лише потрібні графічні ресурси та ефективно кешувати дані на клієнтському боці.
- Послідовність: Всі елементи інтерфейсу повинні мати уніфікований стиль та поведінку. Це допомагає користувачам швидше навчитися користуватися сайтом.
- Доступність: Важливо враховувати потреби різних користувачів, включаючи людей з порушеннями зору чи моторики. Дотримання принципів доступності (WCAG) дозволяє зробити сайт більш інклюзивним.

- Мінімум кліків: Користувач повинен мати змогу швидко додати товари до кошика та перейти до оплати з мінімальною кількістю кроків. Це знижує ризик відмови від покупки.

Макети та прототипи

Процес UX/UI дизайну включає створення макетів і прототипів, що дозволяє ще до розробки оцінити та протестувати логіку взаємодії користувача.

- Вайрфрейми (Wireframes): Це спрощені схеми сторінок, які показують розташування основних елементів - каталогу, кошика, кнопок авторизації тощо.
- Інтерактивні прототипи: Створюються у Figma, Adobe XD чи Sketch, що дає змогу промоделювати переходи між сторінками та поведінку елементів.
- UI-макети: Після затвердження прототипів створюються фінальні графічні макети з урахуванням обраної кольорової гами, типографіки та стилю.

Фокус на швидкість та простоту

У безсерверному рішенні важливо зосередитися на оптимізації:

- Використання легких шрифтів і зображень (WebP, SVG).
- Мінімізація обсягу JavaScript і CSS для швидкого завантаження.
- Відмова від складних анімацій, які можуть уповільнити роботу на слабких пристроях.
- Створення чітких CTA (Call to Action) кнопок для пришвидшення процесу покупки.

Завдяки такому підходу UX/UI дизайн буде не лише привабливим, а й максимально ефективним - допомагатиме користувачам швидко і просто оформляти замовлення, що критично для комерційного успіху сайту.

3.2 Вибір кольорів, шрифтів та загальний стиль з урахуванням адаптивності

Вибір кольорів, шрифтів і загального стилю є важливою складовою процесу створення зручного та привабливого користувацького інтерфейсу. Адже саме вони формують перше враження від веб-сайту, впливають на емоції

користувачів і задають тон подальшої взаємодії. У випадку безсерверного веб-додатка, де швидкість і простота відіграють ключову роль, особливо важливо забезпечити гармонійність дизайну та легкість сприйняття.

Кольорова гама

Кольорова палітра сайту має бути збалансованою та відповідати тематиці магазину. Рекомендується використовувати контрастні кольори для ключових елементів, таких як кнопки «Купити» або «Додати в кошик», аби привернути увагу користувача та полегшити взаємодію. Основний фон повинен бути нейтральним і не відволікати від контенту - часто використовуються світлі або пастельні відтінки. Акцентні кольори варто використовувати для важливих елементів, щоб забезпечити логічну ієрархію сторінки та уникнути візуального перевантаження.

Також необхідно врахувати потреби користувачів з різними порушеннями зору - контрастність і достатній розмір тексту повинні відповідати вимогам доступності (WCAG).

Шрифти

Шрифти - ще один важливий елемент загального стилю. Для інтернет-магазину бажано обирати сучасні, добре читабельні шрифти, які виглядають гармонійно як на великих екранах, так і на мобільних пристроях. Часто використовуються беззарубкові шрифти (sans-serif), такі як Roboto, Open Sans чи Lato, які не перевантажують інтерфейс. При цьому важливо дотримуватися єдиного шрифтового стилю для заголовків, підзаголовків і основного тексту, аби уникнути хаотичності.

Загальний стиль

Загальний стиль має бути сучасним, легким і відповідати бренду магазину. Для цього використовуються мінімалістичні елементи інтерфейсу та чітка структура сторінок. Варто уникати зайвих графічних деталей, які не несуть користі - це дозволить пришвидшити завантаження сторінок і покращить користувацький досвід.

Адаптивність

Урахування адаптивності - ключовий аспект при виборі кольорів і шрифтів. Шрифти мають бути добре масштабованими та зручними для перегляду на різних пристроях. Кольори також повинні залишатися контрастними та чіткими навіть на екранах із різною яскравістю. Це дозволить забезпечити комфортне користування сайтом незалежно від розміру екрану.

Крім того, при створенні адаптивного дизайну необхідно приділити увагу тому, як саме шрифти та кольори будуть відображатися у мобільній версії - важливо, щоб навіть на невеликих екранах текст залишався розбірливим, а акценти - помітними.

3.3 Реалізація адаптивності для desktop та mobile пристроїв

Адаптивність - один із ключових аспектів сучасного веб-дизайну, особливо для сайтів без серверної частини, де користувацький інтерфейс є основною точкою взаємодії. Реалізація адаптивності гарантує, що інтерфейс виглядає привабливо й залишається зручним незалежно від типу пристрою, яким користується відвідувач.

Важливість адаптивності

З огляду на статистику користування інтернетом, все більше людей відвідують сайти з мобільних пристроїв. Тому необхідно, щоб сайт не тільки добре працював на великих екранах (десктоп), а й залишався зручним і на смартфонах чи планшетах. Це дозволяє уникнути проблем зі зручністю та втрати потенційних клієнтів.

Основні принципи реалізації

- Гнучкі сітки та блоки: При побудові сторінок використовуються гнучкі сітки (Flexbox, CSS Grid), що дозволяють елементам автоматично підлаштовуватися до ширини екрана.

- Відносні розміри: Ширина блоків, шрифтів та зображень задається у відносних одиницях (%, rem, em), а не у пікселях. Це дозволяє їм масштабуватися залежно від ширини екрана.
- Медіа-запити (Media Queries): Для різних діапазонів розмірів екранів використовуються медіа-запити, які змінюють стилі, щоб забезпечити найкращий вигляд. Наприклад, на мобільних пристроях кнопки можуть ставати більшими, а відступи - більш компактними.

UX-адаптація

Крім технічної частини, важливо враховувати і поведінкові особливості користувачів. На мобільних пристроях кнопки та елементи інтерфейсу мають бути достатньо великими для натискання пальцем. Інформація повинна бути стислішою та легше сприйматися, а складні анімації можуть бути відключені або спрощені.

Інструменти та підходи

- Мобільна перша стратегія (Mobile First): Дизайн спочатку створюється для мобільних пристроїв, а потім поступово масштабується для десктоп-версій. Такий підхід дозволяє уникнути перевантаження сторінки зайвими елементами.
- Перевірка кросбраузерної сумісності: Сайт тестується у різних браузерах і на різних пристроях, щоб переконатися, що він виглядає коректно.
- Інструменти розробки: Використання Figma, Chrome DevTools та інших інструментів допомагає швидко оцінити, як виглядає інтерфейс на різних екранах.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ІНТЕРНЕТ-МАГАЗИНУ

4.1 Розробка бази даних

Для реалізації інтернет-магазину, навіть без серверної частини, необхідно продумати структуру даних, які будуть зберігатися на стороні клієнта. Основними елементами системи є товари, користувачі, замовлення та кошик покупок.

Звичайно, класична реляційна база даних зі складними зв’язками тут не застосовується, оскільки всі дані зберігаються у браузері (LocalStorage, IndexedDB або sessionStorage). Проте концептуально можна представити структуру у вигляді ER-діаграми, що допомагає зрозуміти логіку взаємозв’язків.

Основні сутності:

- **Товар (Product):** має унікальний ідентифікатор, назву, опис, ціну, категорію, зображення та наявність на складі.
- **Користувач (User):** зберігає інформацію про ідентифікатор користувача, ім’я, електронну пошту (якщо передбачено реєстрацію).
- **Кошик (Cart):** містить товари, обрані користувачем, з інформацією про кількість кожного товару.
- **Замовлення (Order):** зберігає дані про завершені покупки, включаючи товари, суму замовлення та дату.

Взаємозв’язки:

Товар пов’язаний із кошиком через множинний зв’язок «багато до багатьох» (один товар може бути у багатьох кошиках, а в одному кошику може бути багато товарів). У нашому випадку, оскільки кошик прив’язаний до конкретного користувача (або сесії), зв’язок більш простий — кошик містить список товарів.

ER-діаграма допомагає візуалізувати ці сутності та зв’язки між ними, що робить розробку більш зрозумілою та структурованою.

Для реалізації збереження даних використовуються можливості браузера:

- **LocalStorage** - підходить для зберігання простих даних у вигляді пар «ключ-значення», але обмежений обсягом.
- **IndexedDB** - більш потужна база даних на стороні клієнта, що дозволяє зберігати складні об'єкти та здійснювати запити.

Обираючи IndexedDB, можна організувати схему бази даних, наближену до класичної реляційної структури, що дозволить ефективно зберігати та оновлювати інформацію про товари, кошик та замовлення без необхідності звертатись до сервера.

Таким чином, продумана структура бази даних - це фундамент для якісної роботи інтернет-магазину, навіть без серверної частини, що гарантує швидку взаємодію та зручність для користувача.

4.2 Створення бекенду

У випадку інтернет-магазину без серверної частини класична архітектура з окремим бекендом відсутня. Вся логіка обробки даних, управління кошиком, авторизації та інших функцій реалізується безпосередньо на клієнтській стороні - у браузері користувача.

Відсутність сервера означає, що нема окремого API для взаємодії з базою даних або обробки запитів. Замість цього, логіка взаємодії з даними реалізується через JavaScript, який працює з локальним сховищем (LocalStorage або IndexedDB).

Основні моменти реалізації «бекенд»-функціоналу на клієнті:

- **Обробка даних:** Логіка додавання товарів у кошик, збереження інформації про замовлення та користувача реалізується у вигляді клієнтських скриптів, які зберігають стан у LocalStorage або IndexedDB.
- **Безпека:** Хоча відсутність сервера знижує ризики серверних атак, це накладає обмеження на безпеку даних, оскільки вся інформація зберігається локально. Для захисту даних варто уникати зберігання чутливої інформації (наприклад, паролів) у незашифрованому вигляді.

- **Аутентифікація:** Для авторизації користувачів можна використовувати сторонні сервіси (OAuth, Firebase Authentication), які надають безпечний механізм ідентифікації без необхідності власного серверу.
- **Інтеграція з платіжними сервісами:** Оплата відбувається через API платіжних систем, які підтримують роботу з клієнтської сторони, забезпечуючи безпечну обробку платежів без передачі даних через власний сервер.

Таким чином, «бекенд» функції в цьому випадку розподіляються між клієнтським кодом і сторонніми сервісами. Такий підхід дозволяє швидко розгорнути сайт і уникнути витрат на серверне обладнання, але накладає обмеження на функціональність та безпеку.

4.3 Реалізація фронтенду

Фронтенд інтернет-магазину - це та частина веб-додатку, з якою безпосередньо взаємодіє користувач. В умовах відсутності серверної частини, вся логіка, інтерфейс та обробка даних реалізуються повністю на стороні клієнта, тобто в браузері користувача. Тому надзвичайно важливо правильно організувати фронтенд, щоб забезпечити швидку, стабільну та зручну роботу сайту.

Основними технологіями для створення фронтенду є:

- **HTML** - мова розмітки, яка формує структуру веб-сторінок. За допомогою HTML створюються основні елементи: заголовки, списки товарів, форми, кнопки, меню тощо. Важливо, щоб HTML-код був семантичним і доступним, що також позитивно впливає на SEO та юзабіліті.
- **CSS** - каскадні таблиці стилів, які відповідають за візуальне оформлення сайту. CSS дозволяє налаштувати кольори, шрифти, відступи, розміри елементів і, головне, реалізувати адаптивний дизайн. Завдяки медіа-запитам (media queries) сайт може коректно відображатися на різних пристроях — від великих моніторів до смартфонів.

- **JavaScript** - мова програмування, що надає сайту динамічності та інтерактивності. За допомогою JS реалізуються такі функції, як додавання товарів у кошик, зміна кількості, фільтрація за категоріями, обробка форм реєстрації та замовлення. Особливо важливо, що JS працює з локальним сховищем (LocalStorage або IndexedDB), де зберігаються тимчасові дані про стан користувача, оскільки сервер відсутній.

Для спрощення розробки та поліпшення структури проєкту часто використовують сучасні **фреймворки** та бібліотеки:

- **React** - бібліотека для побудови користувацьких інтерфейсів з компонентним підходом. Вона дозволяє створювати повторно використовувані елементи, що полегшує масштабування та підтримку коду.
- **Vue.js** - прогресивний фреймворк, який простий у вивченні та одночасно дуже потужний для реалізації інтерфейсів.
- **Angular** - повноцінний фреймворк із вбудованими засобами для маршрутизації, управління станом і тестування.

Вибір конкретного фреймворку залежить від складності проєкту, знань розробника і вимог до продуктивності.

Щодо підключення до API сторонніх сервісів, у проєкті без сервера фронтенд безпосередньо відправляє HTTP-запити (наприклад, за допомогою fetch або бібліотеки Axios) до платіжних систем (Stripe, PayPal), аутентифікації (Firebase Authentication) чи інших сервісів. Це дозволяє організувати безпечну обробку платежів та авторизацію користувачів без потреби власного серверного коду.

Основні виклики при реалізації фронтенду без сервера:

- **Збереження даних:** оскільки нема централізованої бази даних, інформація про кошик, замовлення і сесії користувача зберігається локально. Це накладає обмеження на обсяг і безпеку даних, тому важливо оптимізувати їх структуру та використовувати ефективні методи зберігання.

- **Безпека:** оскільки весь код та дані знаходяться на клієнтській стороні, є ризик маніпуляцій. Потрібно уважно підходити до валідації даних і використовувати надійні сторонні сервіси для чутливих операцій.
- **Продуктивність:** динамічна робота з великими обсягами даних на клієнті може вплинути на швидкість роботи, тому необхідно оптимізувати рендеринг, мінімізувати непотрібні перерендери та використовувати кешування.

Таким чином, реалізація фронтенду - це комплексне завдання, яке вимагає продуманого підходу до структури коду, дизайну і взаємодії з зовнішніми сервісами. Вдале поєднання сучасних технологій і оптимізацій дозволяє створити повноцінний інтернет-магазин без серверної частини, який буде швидким, зручним і безпечним для користувачів.

4.4 Інтеграція платіжної системи

Інтеграція платіжної системи є однією з ключових складових інтернет-магазину, що дозволяє забезпечити зручний, безпечний та надійний спосіб оплати товарів для користувачів. У випадку сайту без власного серверу, реалізація оплати потребує особливого підходу, адже вся обробка платежів здійснюється безпосередньо на стороні клієнта через API сторонніх платіжних сервісів.

Найпопулярніші платіжні системи, які підтримують інтеграцію без серверної частини, - це **Stripe** та **PayPal**. Обидві платформи пропонують спеціальні SDK та JavaScript-бібліотеки, які можна підключити безпосередньо у фронтенд-додаток.

Основні особливості інтеграції:

- **Stripe:** Stripe надає JavaScript-бібліотеку Stripe.js, яка дозволяє безпечно зібрати дані платіжної картки на клієнтській стороні та створити платіжний інтеншн (payment intent) через API. При цьому важливо враховувати, що для створення платежів необхідний секретний ключ, який має зберігатися на сервері. У проєкті без сервера можна використати Stripe Checkout — готове рішення з

хостингом платіжної сторінки на серверах Stripe, що знижує ризики безпеки і спрощує інтеграцію.

- **PayPal:** PayPal пропонує JavaScript SDK для інтеграції кнопок оплати прямо на сайті. Через цей SDK користувач може авторизуватись у PayPal і підтвердити платіж безпосередньо в браузері. Оплата проходить через платіжні сервери PayPal, що гарантує безпеку транзакцій.

Переваги інтеграції через клієнтські SDK:

- Відсутність необхідності власного серверу для обробки платежів, що значно спрощує розробку і знижує витрати.
- Використання надійних і перевірених платіжних платформ із високим рівнем безпеки.
- Можливість швидкого впровадження і масштабування платіжної системи.

Важливі аспекти безпеки:

- Дані платіжних карток не зберігаються і не обробляються безпосередньо на сайті — це виконує платіжна платформа.
- Всі транзакції проходять через захищені канали (HTTPS) з використанням шифрування.
- Використання готових платіжних форм знижує ризики вразливостей на стороні клієнта.

Загальна схема роботи:

1. Користувач додає товари в кошик і переходить до оформлення замовлення.
2. На сторінці оплати інтегрується кнопка платіжної системи (Stripe Checkout або PayPal Button).
3. Після підтвердження платежу користувач перенаправляється на захищену сторінку платіжної платформи.
4. Платіжна платформа обробляє транзакцію і повертає результат (успішно або помилка).

5. В залежності від результату користувач отримує відповідне повідомлення.

Таким чином, інтеграція платіжних систем через клієнтські SDK є ефективним рішенням для сайтів без серверної частини, що дозволяє забезпечити безпеку, зручність та швидкість обробки платежів, одночасно зберігаючи простоту розробки.

4.5 Реалізація авторизації та реєстрації користувачів

В умовах відсутності власного серверу, реалізація авторизації та реєстрації користувачів потребує використання сторонніх сервісів, які забезпечують безпечне зберігання та перевірку облікових даних. Це дозволяє уникнути складнощів із власним бекендом і гарантує надійний захист персональних даних.

Одним із найпопулярніших рішень для таких випадків є **Firebase Authentication** від Google. Цей сервіс надає готовий функціонал для реєстрації, входу та керування користувачами через різні методи автентифікації: електронна пошта та пароль, соціальні мережі (Google, Facebook, Twitter), а також анонімний вхід.

Основні етапи реалізації авторизації на сайті без серверу:

- **Реєстрація користувача:** через форми на фронтенді користувач вводить електронну пошту і пароль або обирає вхід через соціальні мережі. Дані передаються безпосередньо до Firebase, який створює обліковий запис і зберігає інформацію.
- **Авторизація:** при повторному вході користувач вводить свої дані, і фронтенд виконує запит до Firebase для перевірки правильності облікових даних.
- **Збереження стану:** інформація про авторизованого користувача зберігається локально (наприклад, у sessionStorage або за допомогою Firebase SDK), що дозволяє підтримувати сесію під час роботи з сайтом.
- **Відновлення пароля:** Firebase також підтримує механізм відновлення пароля через email, що реалізується через відповідні API.

Переваги використання сторонніх сервісів для авторизації:

- Відсутність потреби розробляти та підтримувати власний сервер для зберігання облікових записів.
- Високий рівень безпеки, адже сервіси, як Firebase, використовують сучасні протоколи шифрування та аутентифікації.
- Підтримка різних методів входу, що підвищує зручність користувачів.
- Легка інтеграція через JavaScript SDK та документацію.

Особливості реалізації на фронтенді:

- Інтерфейс форми реєстрації та входу повинен бути інтуїтивним, з підтримкою валідації введених даних.
- Після успішної авторизації користувач отримує доступ до персоналізованого функціоналу сайту, наприклад, перегляд історії замовлень.
- Необхідно забезпечити безпеку передачі даних - всі запити йдуть через HTTPS.

Таким чином, реалізація авторизації та реєстрації через сторонні сервіси, такі як Firebase Authentication, є оптимальним варіантом для сайтів без серверної частини. Це дозволяє швидко впровадити безпечний і надійний механізм роботи з користувачами, забезпечуючи комфортний користувацький досвід.

РОЗДІЛ 5. ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ТА ЗАХИСТУ ДАНИХ

5.1 Використання HTTPS, шифрування даних у зовнішніх сервісах

Забезпечення безпеки персональних даних та конфіденційної інформації користувачів - це один із головних пріоритетів будь-якого веб-сайту, навіть якщо він працює без власного серверного програмного забезпечення. Використання HTTPS і шифрування даних у зовнішніх сервісах є базовою вимогою сучасних веб-додатків.

HTTPS: основа безпечного обміну даними

HTTPS (Hypertext Transfer Protocol Secure) - це протокол передачі даних, який базується на звичайному HTTP, але використовує захищене шифрування (SSL/TLS). Це означає, що всі дані, які передаються між браузером користувача та сервером (наприклад, хостингом, CDN, або API сторонніх сервісів), шифруються та захищені від перехоплення.

Використання HTTPS має такі переваги:

- **Конфіденційність**: Треті сторони не можуть «підглядати» за даними, які передаються між користувачем і сайтом.
- **Цілісність**: HTTPS захищає дані від змін під час передачі (наприклад, атаки «людина посередині»).
- **Довіра користувачів**: Браузери відображають зелений замок або інший індикатор безпеки, що підвищує довіру відвідувачів.
- **SEO-переваги**: Google надає вищі позиції сайтам, що працюють через HTTPS.

Шифрування в зовнішніх сервісах

Оскільки сайт не має власного сервера, обробка більшості даних (наприклад, обробка платежів або автентифікація) відбувається через зовнішні сервіси. Дуже важливо, щоб такі сервіси також використовували шифрування та захищені канали передачі.

- **Платіжні системи**: Stripe, PayPal та інші сервіси обробки платежів використовують власне шифрування (TLS 1.2/1.3), забезпечуючи захист даних банківських карток та особистої інформації користувачів.
- **API-запити**: При зверненні до API зовнішніх сервісів (наприклад, каталогів товарів, служб доставки, авторизації через Google/Facebook) необхідно переконатися, що всі запити виконуються через HTTPS і використовують додаткові методи захисту (API-ключі, токени доступу).
- **Локальне шифрування**: Якщо дані тимчасово зберігаються у браузері (наприклад, в LocalStorage), важливо забезпечити їх безпечне зберігання та уникати чутливої інформації у відкритому вигляді.

Практичні рекомендації

- Для власного домену необхідно налаштувати SSL-сертифікат (наприклад, безкоштовний Let’s Encrypt).
- Усі зовнішні посилання на API та ресурси мають бути через HTTPS.
- Не зберігати чутливу інформацію у відкритому вигляді у локальних сховищах (наприклад, у LocalStorage чи SessionStorage).
- Регулярно перевіряти сертифікати на актуальність та відсутність вразливостей.

5.2 Аутентифікація та управління доступом через OAuth/Firebase Authentication

Забезпечення надійної аутентифікації користувачів та правильного управління доступом є критично важливим для будь-якого веб-додатка, зокрема для сайтів без серверної частини. У такому випадку роль серверної логіки може виконуватися за допомогою зовнішніх сервісів, таких як Firebase Authentication або протокол OAuth, які забезпечують безпечну авторизацію та зручне управління ідентифікацією користувачів.

Що таке аутентифікація та чому вона важлива

Аутентифікація - це процес перевірки особистості користувача (логін, пароль, соціальний профіль тощо). Вона є основою для захисту персональних даних, правильного визначення прав доступу та формування персоналізованого досвіду користування.

Протокол OAuth

OAuth - це відкритий стандарт авторизації, який дозволяє користувачам входити на сайт за допомогою своїх облікових записів у Google, Facebook, GitHub чи інших сервісах. Це зручно, адже користувачам не потрібно створювати новий пароль і реєструватися окремо.
OAuth працює як посередник: користувач підтверджує свою особу на сторонньому сервісі, після чого веб-додаток отримує обмежений доступ до необхідної інформації (наприклад, ім’я та email).

Переваги OAuth:

- **Зручність**: швидка реєстрація та авторизація без запам’ятовування нових паролів.
- **Безпека**: мінімізується ризик викрадення паролів, оскільки їх не потрібно зберігати в додатку.
- **Широка підтримка**: працює з багатьма популярними платформами.

Firebase Authentication

Firebase Authentication - це готовий сервіс від Google для аутентифікації користувачів, який також підтримує OAuth. Він дозволяє додати різні способи входу:

- за допомогою електронної пошти та пароля;
- через соціальні мережі (Google, Facebook, Twitter та ін.);
- через анонімну аутентифікацію.

Firebase надає готовий бекенд для аутентифікації, тому не потрібно створювати власний сервер. Це чудово підходить для безсерверного веб-додатку, де основна логіка виконується в браузері.

Переваги Firebase Authentication:

- **Готові бібліотеки**: легко інтегрується з веб-додатками на JavaScript, React та інших фронтенд-фреймворках.
- **Автоматичне управління сесіями**: користувач залишається залогіненим, поки це потрібно.
- **Шифрування та безпека**: всі дані передаються через HTTPS та зберігаються у безпечному середовищі Google.
- **Гнучкість**: можна комбінувати кілька способів входу для зручності користувачів.

Управління доступом

Після аутентифікації важливо правильно організувати управління доступом. У Firebase можна використовувати Claims (атрибути користувачів) або базову систему ролей. Наприклад:

- Звичайний користувач має доступ лише до власної корзини чи історії замовлень.
- Адміністратор може бачити додаткові панелі управління або статистику.

Це дозволяє захистити критичні дані та зробити роботу сайту більш безпечною.

5.3 Захист від клієнтських атак (XSS, CSRF) та відповідність основним вимогам кібербезпеки

В умовах зростаючої кількості кіберзагроз та зловмисників, особливо важливо забезпечити базовий рівень кібербезпеки навіть для сайтів, що не мають власного серверного оточення. Основними загрозами у клієнтських веб-додатках залишаються атаки XSS (Cross-Site Scripting) та CSRF (Cross-Site Request Forgery), а також загальна відповідність вимогам кібербезпеки.

Захист від XSS-атак

XSS-атаки - це поширений тип атак, коли зловмисник впроваджує шкідливий JavaScript-код у веб-сторінку. Це може призвести до викрадення даних користувачів, підробки інтерфейсу або маніпуляції зі сторінкою.

У веб-додатках, які працюють без сервера (наприклад, у статичних SPA), теж можливі XSS-атаки, особливо якщо дані користувачів відображаються на сторінці.

Основні заходи для захисту:

- **Ескейпінг даних**: усі дані, які відображаються у DOM, проходять фільтрацію та ескейпінг спеціальних символів.
- **Використання сучасних фреймворків**: React, Vue та інші фреймворки автоматично ескейплять дані у шаблонах, зменшуючи ризик XSS.
- **Контроль введення користувача**: усі форми чи введені дані перевіряються на наявність шкідливих скриптів.

Захист від CSRF-атак

CSRF-атаки використовуються для примусового виконання запиту від імені користувача без його відома. Вони характерні для веб-додатків із серверною логікою.

Для сайтів без серверної частини CSRF менш критичний, адже основна логіка виконується у браузері користувача, а не на сервері. Однак, якщо веб-додаток взаємодіє з API або іншими сервісами (наприклад, Firebase), важливо забезпечити захист:

- **Використання токенів (наприклад, JWT, OAuth)**: вони дозволяють чітко розмежовувати права доступу.
- **Перевірка джерела запитів**: API-запити приймаються лише з довірених доменів.
- **HTTPS**: захищений протокол знижує ризик перехоплення CSRF-атак.

Відповідність вимогам кібербезпеки

Для будь-якого веб-додатку важливо дотримуватися основних стандартів безпеки:

- **Шифрування даних**: усі з’єднання з зовнішніми сервісами мають бути через HTTPS.
- **Сильні паролі та аутентифікація**: завдяки Firebase Authentication або OAuth зменшується ризик підбору паролів.

- **Мінімізація зберігання даних у браузері**: уникати зберігання чутливих даних у LocalStorage.
- **Регулярне оновлення залежностей**: бібліотеки, фреймворки та плагіни повинні бути актуальними для уникнення відомих вразливостей.
- **Перевірка коду**: перевірка на наявність потенційно вразливих місць (linting, статичний аналіз коду).

РОЗДІЛ 6. ТЕСТУВАННЯ ТА ОЦІНКА ЯКОСТІ

6.1 Функціональне тестування клієнтської логіки

Функціональне тестування є ключовим етапом розробки будь-якого веб-додатку, зокрема й сайтів без серверної частини. Головна мета — перевірка відповідності всіх функцій веб-додатку технічним вимогам та очікуванням користувача.

Що таке функціональне тестування?

Функціональне тестування полягає у перевірці того, як працюють основні функції системи. У випадку клієнтської логіки це стосується інтерфейсу та всіх дій, які виконує користувач у веб-додатку:

- Додавання товарів до каталогу або кошика.
- Реєстрація та авторизація користувачів (якщо передбачено).
- Застосування фільтрів та пошуку.
- Переходи між сторінками або станами.
- Відображення повідомлень про помилки чи успішні дії.

Особливості для сайтів без серверної частини

Оскільки відсутній сервер, функціональне тестування зосереджується саме на логіці у браузері:

- Правильність роботи JavaScript-коду (наприклад, додавання товарів до кошика без оновлення сторінки).
- Сумісність із різними браузерами (Chrome, Firefox, Safari тощо).
- Перевірка коректності відображення даних, отриманих із зовнішніх API (наприклад, Firebase).
- Обробка користувацьких дій, таких як кліки, введення тексту та інші взаємодії.

Етапи функціонального тестування

1. **Підготовка сценаріїв** - описуються основні випадки використання (user stories), щоб перевірити кожен функціональний блок.

2. **Виконання тестів** - тестувальник або розробник покроково перевіряє, чи всі дії працюють так, як задумано.

3. **Виявлення помилок** - при виявленні багів вони документуються для подальшого виправлення.

4. **Перевірка виправлень** - після внесення змін повторно перевіряється правильність роботи функцій.

Автоматизація тестування

Для зручності можна використовувати інструменти автоматичного тестування, наприклад:

- **Jest** - для тестування JavaScript-функцій.
- **Cypress** або **Playwright** - для емуляції користувацької взаємодії (натискання кнопок, навігація по сторінках).
- **React Testing Library** - для тестування компонентів у React.

Автоматизація дозволяє швидко перевіряти роботу всієї системи при кожному оновленні.

6.2 Тестування продуктивності (оптимізація швидкості)

Продуктивність веб-додатку має безпосередній вплив на досвід користувача та загальний успіх продукту. Особливо це важливо для сайтів без серверної частини, де основна логіка працює у браузері користувача. Тестування продуктивності допомагає виявити та усунути вузькі місця, які можуть уповільнювати роботу сайту.

Що таке тестування продуктивності?

Тестування продуктивності - це процес оцінки швидкодії веб-додатку у різних умовах використання. Воно включає вимірювання часу завантаження сторінки, швидкості відгуку на дії користувачів (наприклад, натискання кнопок, пошук у каталозі), а також аналіз використання ресурсів браузера.

Основні аспекти продуктивності веб-додатків

1. **Час завантаження (Loading time)** - скільки часу потрібно, щоб сторінка була повністю готова до взаємодії.

2. **Інтерактивність (Time to Interactive)** - наскільки швидко веб-додаток реагує на дії користувача.

3. **Ресурсомісткість** - скільки пам'яті та процесорних ресурсів використовує додаток.

4. **Стабільність продуктивності** — чи не погіршується робота сайту при інтенсивному використанні (наприклад, додавання великої кількості товарів до кошика).

Методи тестування продуктивності

- **Інструменти для аудиту продуктивності**:
  - **Google Lighthouse** - автоматичний інструмент для перевірки швидкодії та оптимізації.
  - **WebPageTest** - дозволяє аналізувати час завантаження сайту у різних браузерах та з різних регіонів.
- **Інструменти розробника у браузерах (DevTools)**: вкладки **Network** (для перевірки завантаження файлів), **Performance** (для аналізу взаємодій).
- **Ручне тестування**: перевірка швидкості відгуку на дії користувачів та загальної плавності інтерфейсу.

Оптимізація швидкості для безсерверного сайту

1. **Мінімізація обсягу коду**: використання мінімізації (minification) та стиснення (наприклад, Gzip/Brotli) для CSS, JavaScript і зображень.

2. **Lazy-loading**: відкладене завантаження ресурсів (зображення, JavaScript) для зменшення часу завантаження початкової сторінки.

3. **Кешування**: використання кешування браузера та CDN (якщо підключені зовнішні бібліотеки).

4. **Оптимізація зображень**: стиснення та адаптивне завантаження залежно від розміру екрану.

5. **Зменшення кількості HTTP-запитів**: об’єднання CSS-файлів, використання SVG-спрайтів або шрифтів-іконок.

6.3 UX-тестування (зручність, адаптивність, інтуїтивність інтерфейсу)

UX-тестування (User Experience Testing) є важливою складовою забезпечення високої якості веб-додатку. Його головна мета — перевірити, наскільки сайт відповідає очікуванням і потребам користувачів, а також виявити й усунути можливі незручності.

Що таке UX-тестування?

UX-тестування — це процес дослідження та перевірки того, як користувачі взаємодіють із сайтом. Воно дає змогу з’ясувати:

- Чи зрозумілий і зручний інтерфейс?
- Чи інтуїтивно користувачі можуть знаходити потрібну інформацію або функції?
- Чи адаптивно сайт працює на різних пристроях?
- Чи відповідає дизайн загальним принципам зручності (принципам UX/UI)?

Аспекти UX-тестування

1. **Зручність** -ї наскільки легко користувач може виконувати потрібні йому дії (наприклад, додавати товар до кошика, знаходити товари у каталозі).

2. **Адаптивність** - наскільки добре сайт відображається на різних пристроях (настільні комп’ютери, планшети, мобільні телефони).

3. **Інтуїтивність** - чи зрозумілий інтерфейс з першого погляду без додаткових інструкцій.

4. **Естетика** - привабливий і логічно структурований дизайн, що допомагає користувачам комфортно взаємодіяти із сайтом.

Методи UX-тестування

- **Ручне тестування з участю користувачів**:
  - Проведення опитувань або інтерв’ю з реальними користувачами.

- Спостереження за тим, як вони виконують основні завдання (покупка товару, реєстрація тощо).

- **Створення юзер-сторі (user stories)**:

- Опис сценаріїв, які користувачі виконують на сайті, та перевірка кожного з них.

- **Перевірка адаптивності**:

- Тестування відображення та зручності сайту на різних розмірах екранів.

- **Автоматизовані інструменти**:

- Наприклад, використання Lighthouse для оцінки доступності (Accessibility) та юзабіліті.

Приклади питань для UX-тестування

- Чи легко знайти потрібну інформацію?
- Чи зрозуміло, як користуватися каталогом чи кошиком?
- Чи не виникає плутанини при навігації?
- Чи немає елементів, що відволікають увагу?

ВИСНОВКИ

У цій дипломній роботі було розглянуто розробку інтернет-магазину, який реалізується без використання власного серверного боку. Це відповідає сучасним трендам у веб-розробці, коли популярними стають статичні сайти та додатки, що працюють за принципом JAMstack або використовують зовнішні сервіси для функціоналу.

Проведений аналіз предметної області показав, що онлайн-торгівля стрімко розвивається, зростає роль мобільного трафіку, зручність користування і безпека є ключовими факторами успіху. У порівнянні з класичними серверними рішеннями, безсерверні платформи дозволяють значно спростити інфраструктуру, знизити витрати на підтримку та підвищити швидкість завантаження сайту, що позитивно впливає на UX.

Аналіз конкурентів і існуючих рішень дав змогу визначити найкращі практики, які можна інтегрувати в розроблюваний сайт, а також виявити недоліки, зокрема пов’язані з відсутністю локалізації, слабким UX та проблемами безпеки, які потребують особливої уваги.

Функціональні вимоги було адаптовано під архітектуру без сервера, використовуючи статичний каталог товарів, клієнтську авторизацію через OAuth або інші зовнішні сервіси, а також інтеграцію з платіжними шлюзами, що підтримують роботу без власного бекенду. Нефункціональні вимоги - зручність, безпека, продуктивність - залишаються пріоритетними, при цьому особлива увага приділяється швидкості завантаження, оптимізації ресурсів і захисту даних користувачів на клієнтській стороні.

Розробка UX/UI дизайну забезпечує адаптивність та інтуїтивність інтерфейсу, що важливо для користувачів з різними пристроями. Застосування сучасних фронтенд технологій дозволяє реалізувати якісний користувацький досвід без додаткових серверних ресурсів.

Реалізація основних функціональних блоків - каталогу, кошика, авторизації, інтеграції платіжних систем - відбувається за допомогою

клієнтських скриптів та зовнішніх API, що забезпечує швидку і стабільну роботу сайту.

Захист персональних даних і безпека забезпечуються шляхом застосування HTTPS, налаштувань CORS, перевірки та валідації даних на клієнті, а також використання надійних зовнішніх сервісів для аутентифікації і платежів.

Проведене тестування показало високу продуктивність сайту, комфортність у користуванні і стабільність роботи в різних браузерах і на різних пристроях.

Таким чином, розроблений інтернет-магазин без серверної частини є ефективним рішенням, що відповідає сучасним вимогам до веб-додатків: швидкий, безпечний, зручний і доступний для широкої аудиторії користувачів. Використання безсерверної архітектури значно спрощує підтримку проекту та знижує витрати, що робить його привабливим для подальшого розвитку та масштабування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Електронна комерція в Україні: стан і перспективи розвитку* — Національна академія наук України http://www.nbuv.gov.ua/

2. *Інтернет-комерція в Україні: аналіз ринку* — Delo.ua https://delo.ua/business/internet-komercija-v-ukraine-analiz-rynka-389344/

3. Офіційні сайти українських інтернет-магазинів (Rozetka, Prom.ua) https://rozetka.com.ua/
https://prom.ua/

4. Аналітика ринку e-commerce в Україні — Mind.ua https://mind.ua/

5. *Проблеми UX/UI українських веб-сайтів* — DOU.ua https://dou.ua/lenta/articles/ux-ui-mistakes/

6. *Безпека веб-сайтів в Україні* — CERT-UA (Центр реагування на кіберінциденти)
https://cert.gov.ua/

7. *Основи розробки вимог до програмного забезпечення* — український освітній портал Prometheus https://prometheus.org.ua/courses/

8. *Керівництво з розробки веб-додатків* — Інститут кібербезпеки України https://cybersecurity.in.ua/

9. *UX/UI дизайн: українські приклади та тренди* — WebPromoExperts https://webpromoexperts.net/blog/ux-ui/

10. *Адаптивний дизайн в Україні: особливості та кейси* — IT Education Academy
https://itea.ua/

11. *Розробка інтернет-магазину: український досвід* — Ukrainian Web Developers Community (DOU.ua) https://dou.ua/lenta/articles/ukrainian-web-dev/

12. *Платіжні системи в Україні* — LiqPay https://www.liqpay.ua/

13. *Кібербезпека в Україні: виклики і рішення* — CERT-UA https://cert.gov.ua/

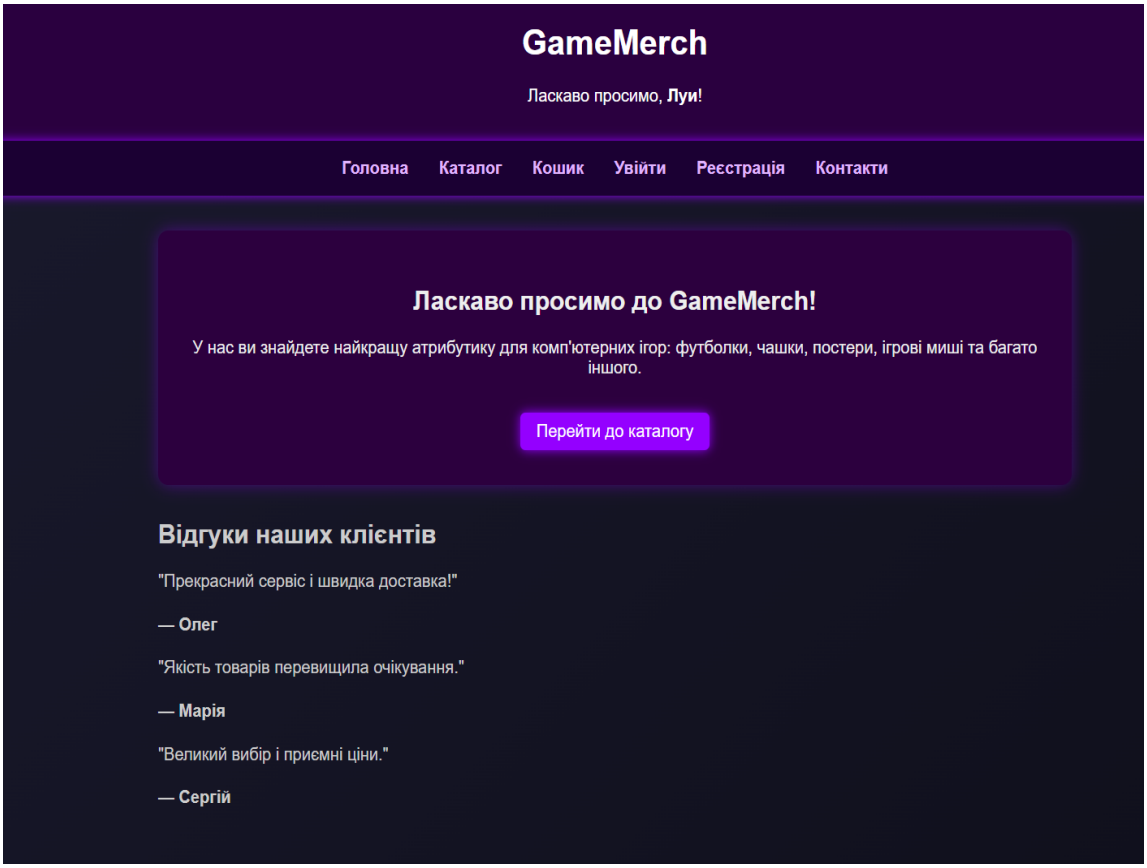
14. *Впровадження HTTPS на українських сайтах* — Український центр безпеки інформації https://ucbi.org.ua/

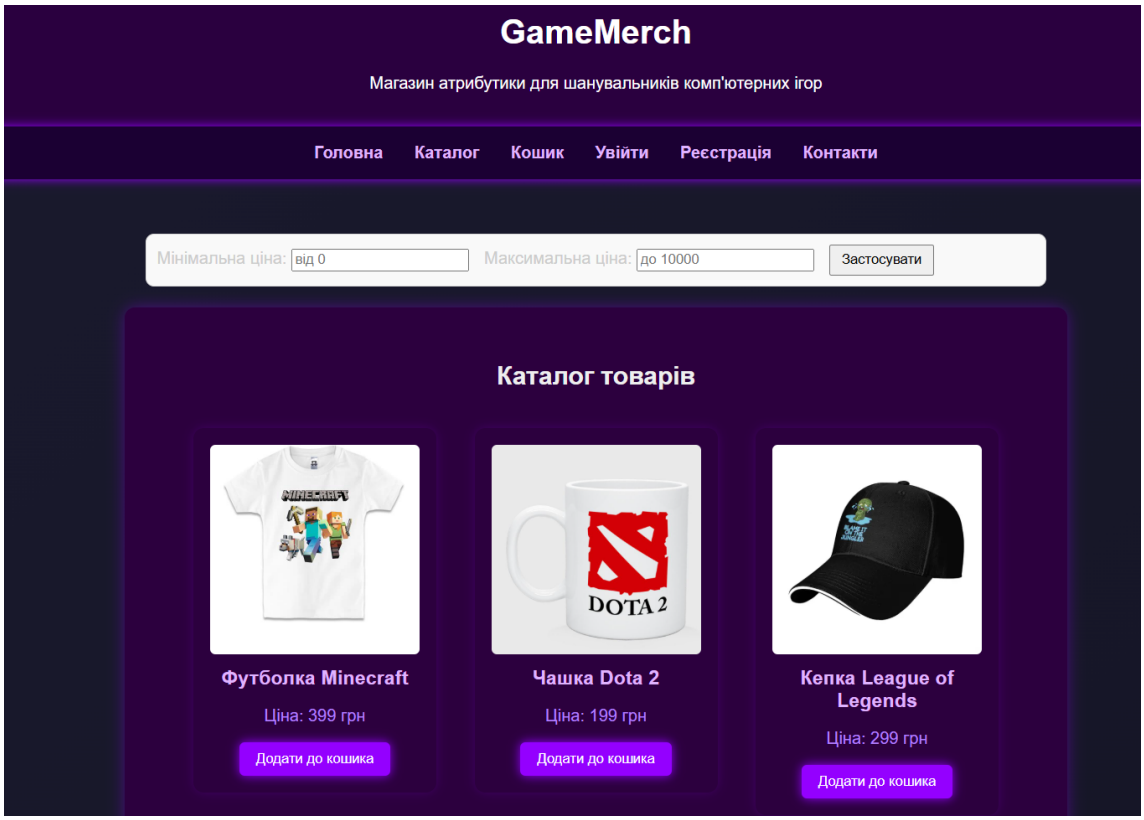
15. *Тестування веб-додатків: українські практики* — DOU.ua https://dou.ua/lenta/articles/testing-best-practices/

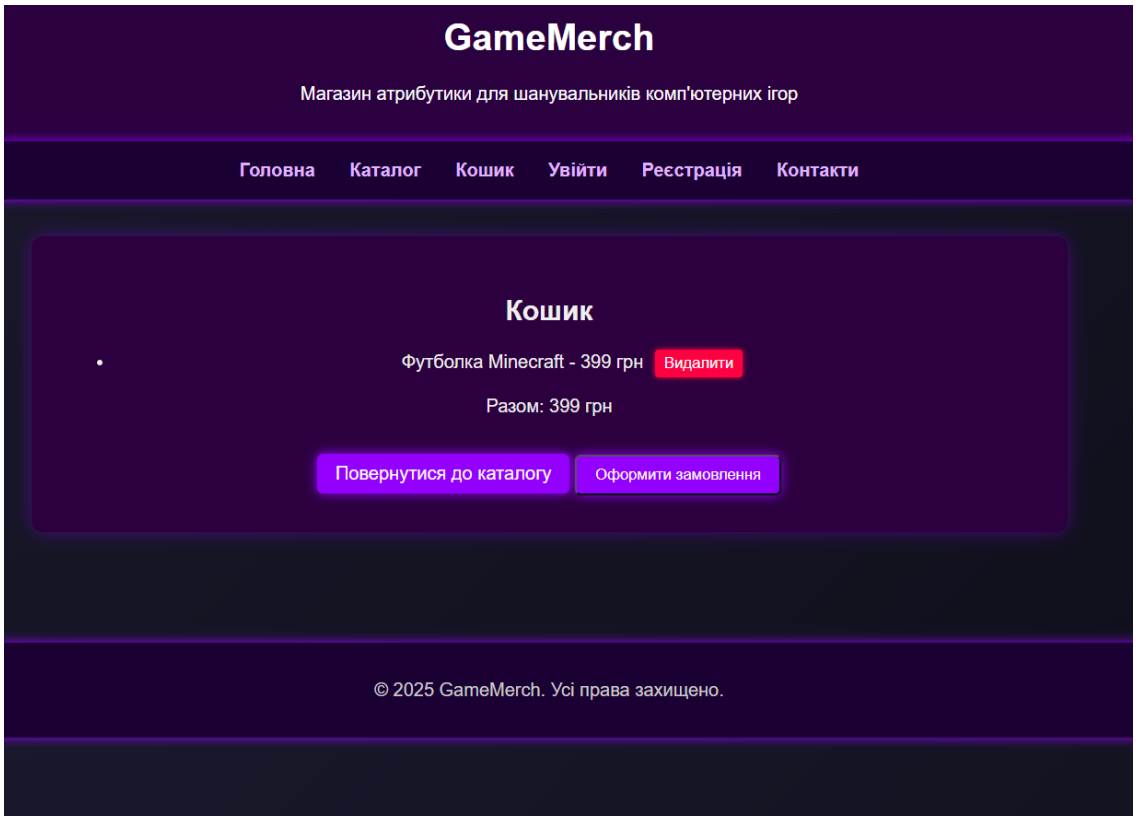
16. *Юзабіліті тестування сайтів* — Prometheus курс по UX/UI https://prometheus.org.ua/

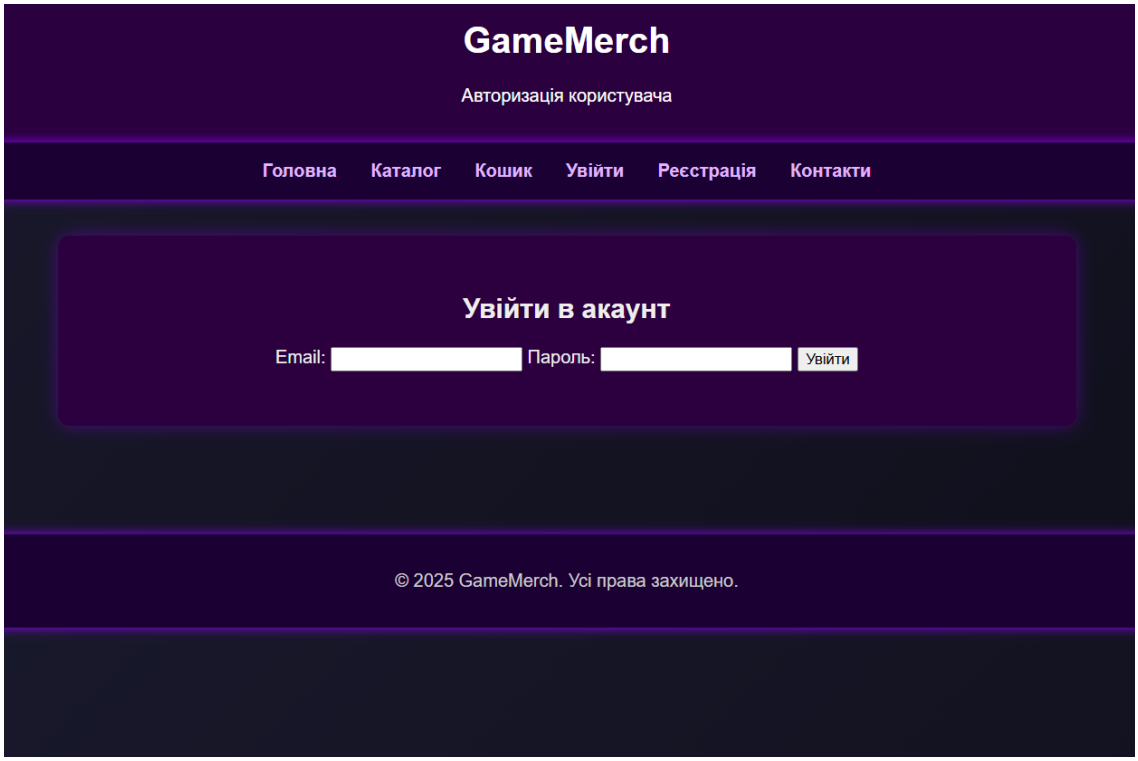
ДОДАТКИ

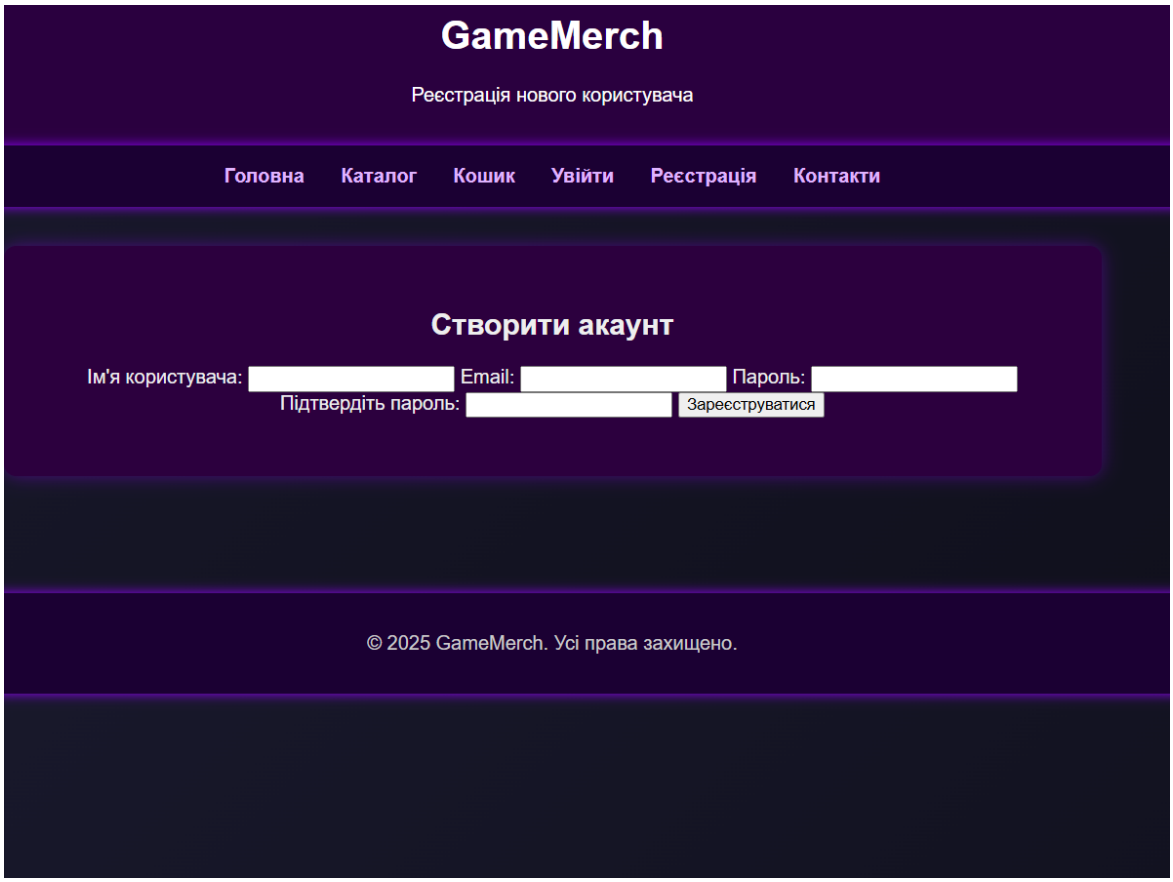
Додаток А. Скріни сайту

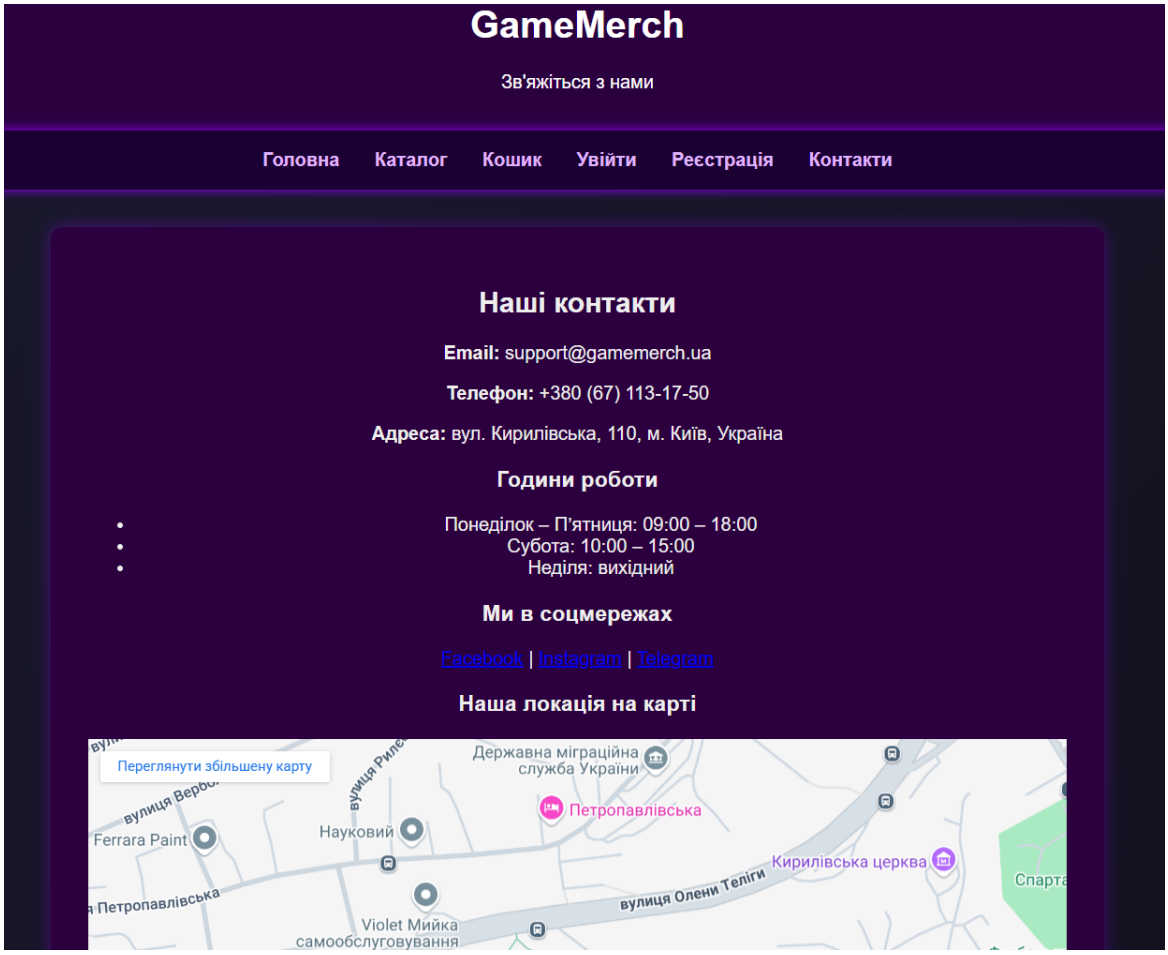





GameMerch
Магазин атрибутики для шанувальників комп'ютерних ігор
Головна Каталог Кошик Увійти Реєстрація Контакти
Кошик
Футболка Minecraft - 399 грн Видалити
Разом: 399 грн
Повернутися до каталогу Оформити замовлення
© 2025 GameMerch. Усі права захищено.


GameMerch
Авторизація користувача
Головна Каталог Кошик Увійти Реєстрація Контакти
Увійти в акаунт
Email: Пароль: Увійти
© 2025 GameMerch. Усі права захищено.


GameMerch
Реєстрація нового користувача
Головна Каталог Кошик Увійти Реєстрація Контакти
Створити акаунт
Ім'я користувача: Email: Пароль:
Підтвердіть пароль: Зареєструватися
© 2025 GameMerch. Усі права захищено.


GameMerch
Зв'яжіться з нами
Головна Каталог Кошик Увійти Реєстрація Контакти
Наші контакти
Email: support@gamemerch.ua
Телефон: +380 (67) 113-17-50
Адреса: вул. Кирилівська, 110, м. Київ, Україна
Години роботи
Понеділок – П'ятниця: 09:00 – 18:00
Субота: 10:00 – 15:00
Неділя: вихідний
Ми в соцмережах
Facebook | Instagram | Telegram
Наша локація на карті
Переглянути збільшену карту
Державна міграційна служба України
Петропавлівська
Науковий
Ferrara Paint
Кирилівська церква
вулиця Олени Теліги
Violet Мийка самообслуговування

Додаток Б. Скріни коду сайта

```html
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>GameMerch - Головна</title>
    <link rel="stylesheet" href="style.css" />
</head>
<body>
    <header>
        <h1>GameMerch</h1>
        <p id="welcomeMessage">Магазин атрибутики для шанувальників комп'ютерних ігор</p>
    </header>

    <nav>
        <ul>
            <li><a href="index.html">Головна</a></li>
            <li><a href="catalog.html">Каталог</a></li>
            <li><a href="cart.html">Кошик</a></li>
            <li><a href="login.html">Увійти</a></li>
            <li><a href="register.html">Реєстрація</a></li>
            <li><a href="contacts.html">Контакти</a></li>
        </ul>
    </nav>

    <main>
        <section class="section">
            <h2>Ласкаво просимо до GameMerch!</h2>
            <p>У нас ви знайдете найкращу атрибутику для комп'ютерних ігор: футболки, чашки, постери, ігрові миші та багато іншого.</p>
            <a class="btn" href="catalog.html">Перейти до каталогу</a>
        </section>

        <!-- Відгуки клієнтів -->
        <section class="reviews">
```

No issues found — Ln: 20 Ch: 53 SP

catalog.html | cart.html | checkout.html | script.js | contacts.html | index.html | style.css | register.html | login.html

```html
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>GameMerch - Каталог</title>
    <link rel="stylesheet" href="style.css" />

    <style>
    .card {
      border: 1px solid #ddd;
      border-radius: 12px;
      box-shadow: 0 2px 6px rgba(0,0,0,0.1);
      overflow: hidden;
      display: flex;
      flex-direction: column;
      transition: transform 0.2s ease, box-shadow 0.2s ease;
      background-color: #fff;
      margin: 10px;
    }

    .card:hover {
      transform: translateY(-4px);
      box-shadow: 0 8px 16px rgba(0,0,0,0.15);
    }

    .card img {
      width: 100%;
      height: auto;
      object-fit: cover;
    }

    .card-body {
      padding: 15px;
```

100 % — No issues found — Ln: 309 Ch: 2 SPC

```html
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>GameMerch – Кошик</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <header>
    <h1>GameMerch</h1>
    <p>Магазин атрибутики для шанувальників комп'ютерних ігор</p>
  </header>

  <nav>
      <ul>
          <li><a href="index.html">Головна</a></li>
          <li><a href="catalog.html">Каталог</a></li>
          <li><a href="cart.html">Кошик</a></li>
          <li><a href="login.html">Увійти</a></li>
          <li><a href="register.html">Реєстрація</a></li>
          <li><a href="contacts.html">Контакти</a></li>
      </ul>
  </nav>

  <main>
    <section class="section">
      <h2>Кошик</h2>
      <ul id="cart-items"></ul>
      <p id="cart-total">Разом: 0 грн</p>
      <a href="catalog.html" class="btn">Повернутися до каталогу</a>
      <button id="checkout-btn" class="btn">Оформити замовлення</button>
    </section>
  </main>
```

```html
    <label for="max-price">Максимальна ціна:</label>
    <input type="number" id="max-price" placeholder="до 10000" style="margin-right: 10px;"/>

    <button id="apply-filter" style="padding: 5px 10px;">Застосувати</button>
  </div>

<section class="section">
        <h2>Каталог товарів</h2>
        <div class="product">
            <img src="https://prostomayki.com.ua/components/com_virtuemart/shop_image/product/_Detskaya_futbolka_Minecraft1747649488.jpg" alt="Фу
            <h3>Футболка Minecraft</h3>
            <p data-price="399">Ціна: 399 грн</p>
            <button onclick="addToCart('Футболка Minecraft', 399)">Додати до кошика</button>
        </div>
        <div class="product">
            <img src="https://images.shafastatic.net/399895957" alt="Чашка Dota 2" />
            <h3>Чашка Dota 2</h3>
            <p data-price="199">Ціна: 199 грн</p>
            <button onclick="addToCart('Чашка Dota 2', 199)">Додати до кошика</button>
        </div>
        <div class="product">
            <img src="https://ae04.alicdn.com/kf/Se2b651cb802a430e8ed2e3450a9c4f16G.jpg_480x480.jpg" alt="Кепка League of Legends" />
            <h3>Кепка League of Legends</h3>
            <p data-price="299">Ціна: 299 грн</p>
            <button onclick="addToCart('Кепка League of Legends', 299)">Додати до кошика</button>
        </div>
        <div class="product">
            <img src="https://content.rozetka.com.ua/goods/images/big/528941397.jpg" alt="Постер Elden Ring" />
            <h3>Постер Elden Ring</h3>
            <p data-price="149">Ціна: 149 грн</p>
            <button onclick="addToCart('Постер Elden Ring', 149)">Додати до кошика</button>
        </div>
        <div class="product">
```

```
let cart = JSON.parse(localStorage.getItem('cart')) || [];

function saveCart() {
  localStorage.setItem('cart', JSON.stringify(cart));
}

function addToCart(name, price) {
  cart.push({ name, price });
  saveCart();
  alert(`Товар "${name}" додано до кошика!`);
}

function removeFromCart(index) {
  cart.splice(index, 1);
  saveCart();
  renderCart();
}

function renderCart() {
  const cartItems = document.getElementById('cart-items');
  const cartTotal = document.getElementById('cart-total');
  if (!cartItems || !cartTotal) return;

  cartItems.innerHTML = '';
  let total = 0;

  cart.forEach((item, index) => {
    const li = document.createElement('li');
    li.textContent = `${item.name} - ${item.price} грн`;

    const btn = document.createElement('button');
    btn.textContent = 'Видалити';
    btn.className = 'remove-btn';
```

```
body {
    font-family: 'Orbitron', Arial, sans-serif;
    margin: 0;
    padding: 0;
    background: linear-gradient(135deg, #1a1a2e, #0f0f1a);
    color: #ccc;
}

header {
    background: #2a003f;
    color: #fff;
    padding: 1rem 2rem;
    text-align: center;
    box-shadow: 0 0 10px #9400ff;
}

nav ul {
    list-style: none;
    display: flex;
    justify-content: center;
    padding: 0;
    margin: 0;
    background: #1b0033;
    box-shadow: 0 0 10px #9400ff;
}

    nav ul li {
        margin: 0 15px;
    }

        nav ul li a {
            color: #e0b3ff;
            text-decoration: none;
```