|  |
| --- |
| **Міністерство освіти і науки України**  **Відокремлений структурний підрозділ «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя»** |
|  |
| відділення електронних апаратів |
|  |
| циклова комісія програмних систем і комплексів |
|  |

ПОЯСНЮВАЛЬНА ЗАПИСКА

**до курсової роботи**

Beginner

|  |
| --- |
|  |
| **на тему:**  Розробка веб-додатку управління компонентами інтернет-магазину карт пам’яті |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Виконав: студент (ка) | 3 | курсу, | групи | КН-321 |
| напряму підготовки (спеціальності) | | | | |
| 122 «Комп’ютерні науки» | | | | |
|  | | | | |
| **Данилейко. В. Р** | | | | |
|  | | | | |
| Керівник | Лісовий В.М. | | |  |
|  | | | | |
| Оцінка: |  |  |  |  |
| Члени комісії: | | | | |
| (підпис) | | (прізвище та ініціали) | | |

м. Тернопіль – 2022

Методичні вказівки до виконання курсових робіт з дисципліни «Web технології та web дизайн» для студентів напряму підготовки 122 «Комр’ютерні науки». – Тернопіль: ВСП «ТФК ТНТУ ім. І.Пулюя», 2023. – 47с.

Методичні вказівки дають змогу студентам підготувати курсовий проект відповідно до прийнятих вимог та положень. У методичних вказівках поетапно розписано пункти виконання курсового проекту. Подано вимоги до оформлення проекту із необхідними зразками.

Укладачі: Капаціла І.Б., спеціаліст вищої категорії, викладач-методист,

Лісовий В.М., спеціаліст вищої категорії

Методичні вказівки розглянуто і схвалено на засіданні циклової комісії програмних систем і комплексів.

Протокол №\_\_ від « \_\_\_» \_\_\_\_\_\_\_\_\_ 2023р.

Голова циклової комісії програмних систем і комплексів \_\_\_\_\_\_\_\_ Г.Я. Марціяш

Схвалено методичною радою Відокремленого структурного підрозділу «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя»

Протокол від « \_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 року № \_\_\_

« \_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 року

Голова \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Я.Д. Шевчук

ЗМІСТ

[ВСТУП 5](#_Toc57644750)

[1 ЗАГАЛЬНИЙ РОЗДІЛ 6](#_Toc57644751)

[1.1 Аналітичний огляд існуючих рішень 6](#_Toc57644752)

[1.2 Технічне завдання 7](#_Toc57644753)

[2 РОЗРОБКА ТЕХНІЧОГО ТА РОБОЧОГО ПРОЕКТУ 11](#_Toc57644754)

[2.1 Розробка структури сайту і web-сторінок 11](#_Toc57644755)

[2.2 Створення та верстка сторінок сайту 17](#_Toc57644756)

[2.3 Розробка структури бази даних сайту 21](#_Toc57644757)

[2.4 Програмування сайту 24](#_Toc57644758)

[2.5 Тестування web-сайту 31](#_Toc57644759)

[3 СПЕЦІАЛЬНИЙ РОЗДІЛ 37](#_Toc57644760)

[3.1 Інструкція з розміщення сайту в Інтернеті 37](#_Toc57644761)

[3.2 Інструкція з обслуговування та наповнення сайту 38](#_Toc57644762)

[3.3 Інструкція з популяризації та підтримки сайту 40](#_Toc57644763)

[ВИСНОВКИ 43](#_Toc57644768)

[ПЕРЕЛІК ПОСИЛАНЬ 44](#_Toc57644769)

[Додаток А Лістинг файлу «index.html» 45](#_Toc57644770)

ВСТУП

У сучасному цифровому світі електронна комерція стає все більш популярною і затребуваною серед споживачів. Інтернет-магазини надають зручну можливість купувати товари та послуги в будь-який час та з будь-якого місця, що робить їх незамінними для сучасного способу життя. Зокрема, ринок компонентів комп'ютерів та електроніки має особливе значення, адже швидкий розвиток технологій приводить до постійного оновлення та покращення цих компонентів.

Одним з найпоширеніших компонентів комп'ютерів є картки пам'яті, які використовуються для зберігання даних у цифровому форматі. За останні кілька років попит на картки пам'яті значно зрос, оскільки вони є необхідними для смартфонів, фотоапаратів, планшетів та багатьох інших електронних пристроїв. Це створює потребу в ефективних інтернет-магазинах, які забезпечують зручну покупку та управління цими картками пам'яті.

У процесі розробки будуть використовуватися сучасні веб-технології, такі як HTML, CSS, JavaScript і бази даних для зберігання інформації про продукти, клієнтів та замовлення. Очікується, що розроблений веб-додаток забезпечить зручну та ефективну роботу інтернет-магазину з картами пам'яті, підвищить його конкурентоспроможність та задовольнить потреби клієнтів у виборі та придбанні необхідних продуктів.

Заключно, ця курсова робота має на меті детально розглянути процес розробки веб-додатку управління компонентами інтернет-магазину карт пам'яті та продемонструвати його практичну застосовність. Вона також спрямована на вивчення сучасних веб-технологій та засобів для розробки інтернет-магазинів та відповідних аспектів управління та клієнтського сервісу.

# ЗАГАЛЬНИЙ РОЗДІЛ

## Аналітичний огляд існуючих рішень

Розробка веб-додатку є комплексним процесом, який включає в себе як візуальну, так і програмну частину. Чітке розділення програмного продукту на складові частини дозволяє ефективно розробити схему взаємодії між ними. При реалізації проекту важливо обирати ефективні інструменти розробки, враховуючи поставлене завдання.

На щастя, сьогодні існує багато сучасних інструментів для ефективної та швидкої розробки програмних рішень. Один з таких інструментів, який може бути використаний для збереження даних на клієнтському боці, це localStorage. localStorage - це механізм, який дозволяє зберігати дані у вигляді ключ-значення на браузері користувача. Він надає можливість зберігати та отримувати дані навіть після закриття веб-сторінки.

Для розробки візуальної частини веб-додатку можна використовувати HTML та CSS. HTML визначає структуру сторінки та розмітку елементів, а CSS відповідає за стилізацію та зовнішній вигляд веб-додатку. Сучасні версії HTML5 та CSS3 надають багато нових можливостей для створення багатофункціональних інтерфейсів.

Для програмної частини веб-додатку можна використовувати JavaScript. JavaScript є мовою програмування, яка дозволяє створювати динамічні та інтерактивні елементи на веб-сторінці. Використання JavaScript дозволяє реалізувати різноманітні функціональні можливості, такі як обробка подій, маніпуляція даними та взаємодія з користувачем.

localStorage можна використовувати в JavaScript для збереження та отримання даних на клієнтському боці. Ви можете зберігати дані у вигляді ключ-значення за допомогою методів, таких як setItem() та getItem(). Наприклад, ви можете зберігати налаштування користувача або інші потрібні дані у localStorage, щоб забезпечити постійну доступність цих даних.

Отже, для розробки веб-додатку ви можете використовувати localStorage для збереження даних на клієнтському боці, HTML та CSS для створення візуальної частини, а JavaScript для програмної реалізації і взаємодії з користувачем. Пам'ятайте про обмеження localStorage та адаптуйте рішення під вимоги вашого проекту.

## Технічне завдання

Найменування та область застосування

Найменування програмного вибору – «Веб-додаток для управління компонентами інтернет-магазину карт пам’яті».

Стисла назва – «MemoryCards».

Область застосування програми – ведення обліку та управління компонентами інтернет-магазину карт пам'яті користувачами веб-додатку, зберігання та впорядкування даних, що безпосередньо стосуються характеристик карт пам'яті.

Вимоги до функціоналу web-сайту

**Вимоги до функціональних характеристик.** Для початку окреслимо функції, які буде містити веб-додаток.

Веб-додаток повинен забезпечувати такі функції як:

* Внесення нових даних про карту пам'яті.
* Редагування вже існуючих компонентів.
* Видалення існуючих даних про карту пам'яті.
* Швидке впорядкування існуючих карт пам'яті.
* Пошук і виокремлення карт пам'яті за характеристиками.

Побудуємо UML-діаграму варіантів використання програми, що зображена на рисунку 1.1.

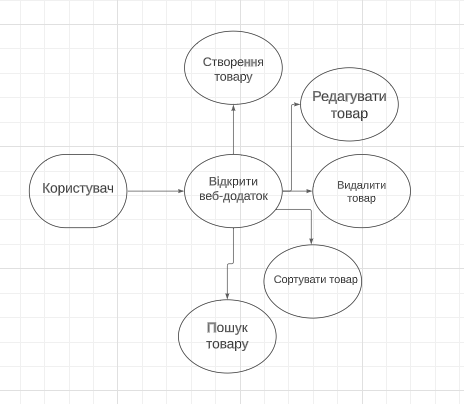


Рисунок 1.1 – UML-діаграма варіантів використання

веб-додатку“MemoryCards ”

Окресливши функціонал, що повинен містити веб-додаток, перейдемо до організації вхідних та вихідних даних.

Вихідні дані формує безпосередньо користувач додадку. Дані, що він вносить, є жорстко обмежені. Від користувача вимагаються всі дані, зазначені нище.

Таблиця 1.1 – Формат вихідних даних веб-додатку

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Виробник | Назва моделі | Тип | Об’єм пам’яті | Speed Class | Макс.швидкість  читання | Макс.швидкість запису |
| SanDisk | Extreme PRO | SD | 64 гб | 10 | 170 МБ/с | 170 МБ/С |

Вхідна інформація буде формуватися на основі вихідних даних, що були одержані від користувача під час роботи веб-додатку.

**Вимоги до надійності.** Додаток забезпечує гнучку систему контролю вихідних та вхідних даних. Контроль вихідних даних полягає у тому, що користувачу, тим чи іншим чином, повідомляється про правильність або неправильність вхідних даних. Таким чином, поки користувач не введе всі необхідні данні, веб-додаток буде повідомляти про помилку в заповненні полів та вкаже де саме потрібно заповнити.

**Умови експлуатації.** Розроблений програмний продукт не потребує жодних видів обслуговування. Для його використання необхідна лише реєстрація нових карт пам’яті, та робота з уже існуючими. Інформація, яка буде зберігатися на сервері, змінюватиметься користувачем при кожній його взаємодії з нею. Тобто користувач може змінювати інформацію як він захоче.

**Часові характеристики.** Вимоги до часових характеристик в інформаційно-пошукових системах становлять до 10с на відповідь на запит або меньше.

# РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЕКТУ

## Розробка структури сайту і web-сторінок

В даному розділі здійснюється проектування інтерфейсу майбутнього програмного застосунку. Перш за все, необхідно визначитися з функціональними елементами, які будуть присутні на веб-ресурсі. Наявність кожного елемента має бути обгрунтована, оскільки надлишковий функціонал може розсіювати увагу розробника, тому підбір компонентів потрібно здійснювати обережно. Необхідно виокремити лише необхідні та важливі елементи.

Майбутній веб-додаток повинен складатися з таких функціональних елементів:

− Головне меню

− Поле пошуку

− Сортування карт пам’яті

− Блок додавання карти пам’яті

− Блок видалення та редагування карти пам’яті

− Список карт пам’яті

Після узгодження та визначення усіх функціональних елементів веб-ресурсу, переходим до детального опису та принципу користування, після чого можна навести рисунки майбутніх елементів.

**Головне меню.** Дуже важлива складова, адже ця частина веб-додатку виступає у ролі провідника, що допомагає навести користувача на потрібний функціонал. Це допоможе користувачеві зрозуміти, яким саме веб-ресурсом він користується. Саме через навігаційне меню користувач здійснює основну роботу на веб-ресурсі.

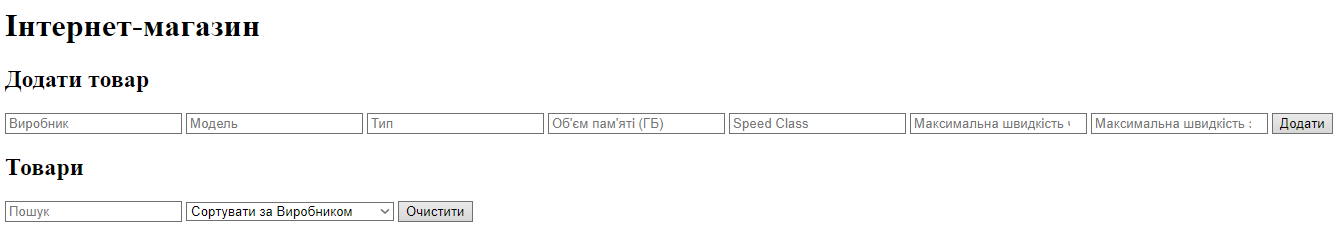


Рисунок 2.1 – Головне меню веб-додатку “MemoryCards”

**Поле пошуку.** Інструмент для виокремлення необхідної інформації є поле пошуку. Він допомагає користувачеві відшукати потрібний йому товар. Власне, саме поле пошуку повинно сигналізувати користувачеві, що з його допомогою можна здійснити пошуковий запит. Принцип користування досить простий. Користувач, бажаючи знайти необхідну інформацію, вводить у поле свій запит, що може складатися з одного або більше слів, після чого веб-додаток повинен відобразити лише ті дані, які точно мають збіги з введеним запитом. Таким чином, наявністю цього елемента дозволяє прискорити та спростити процес пошуку необхідних даних.

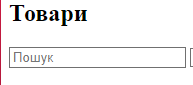


Рисунок 2.2 – Поле пошуку веб-додатку “MemoryCards”

**Сортування карт пам’яті.** Сортування та фільтрування є невід’ємною частиною будь-якого сучасного веб-додатку. Призначення цього блоку є очевиндим. Воно виводить в вниз товари, які відбовідають бажанням користувача.

Цей функціональний елемент обов’язково повинен бути присутнім на веб-ресурсі.

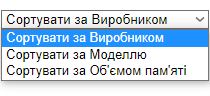
Принцип користування полягає у виборі користувачем певних критеріїв. Після чого веб-додаток повинен у динамічному режимі посортувати кожен елемент та відобразити його у тому порядку, який вибрав користувач. Таким чином, налагоджується швидка вибірка серед компонентів мобільних пристроїв. 

Рисунок 2.3 – Блок сортування веб-додатку “Phones”

**Блок додавання карти пам’яті.** У цьому функціональному елементі відбувається основна робота при заповненні списку с товарами. Користувачу потрібно вписати в певні поля інформацію, що привиде до появи в списку товарів тих карт пам’яті які він добавив, та дозволить виконувати пошук і сортування за допомогою цієї інформації



Рисунок 2.4 – Блок додавання товару веб-додатку “MemoryCards”

**Блок видалення та редагування карти пам’яті.** Функціональний елемент «Видалення» надає користувачу можливість, як вже зрозуміло з назви, видаляти зі списку товарів карти пам’яті, які вибере користувач.

Після видалення карта пам’яті повністю зникне із списка товарів, і відновити її получиться лише тільки при повторному вводі інформації завдавяки блоку додавання.

Редагування ж в свою чергу дозволяє змінювати вже введені дані про карту яку ввів користувач. Таким чином можна виправити помилки при вводі, або добавити нову карту не використовуючи блок додавання.



Рисунок 2.5 –Вікно видалення веб-додатку “MemoryCards”

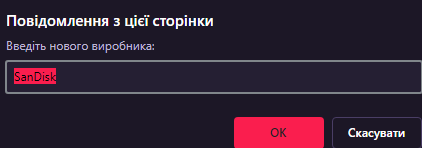


Рисунок 2.6 –Вікно редагування веб-додатку “MemoryCards”

**Список карт пам’яті.** Цей функціональний елемент є візуальним представленням всіх карт пам’яті інформацію про які було внесено в веб-додаток.

Завдяки йому також і виконується редагування, сортування та видалення, тобто взаємодії с картами пам’яті після їх введення в веб-дотаток.

В цьому списку знаходиться вся інформація яку надав користувач, тобто:  
Виробник, Модель, тип, Speed Class, максимальна швидкість читання та запису.





Рисунок 2.7 – Список товару веб-додатку “MemoryCards”

Описавши майбутні функціональні елементи веб-додатку, побудуємо структурну схему його головної сторінки. Схема зображена на рисунку 2.8.



Рисунок 2.8 – Структурна схема головної сторінку веб-додатку “MemoryCards”

## Створення та верстка сторінок сайту

Розробка веб-додатку для управління компонентами інтернет-магазину карт пам'яті вимагає певних кроків та інструментів, щоб забезпечити оптимальну функціональність та користувацький досвід. Перш за все, перед початком розробки необхідно визначитися з вибором середовища розробки. Один з рекомендованих варіантів - використання IDE (інтегрованого середовища розробки) Visual Studio Code, яке надає розширений набір інструментів для редагування веб-додатків.

Опис функціональних елементів та їх використання базується на конструкціях, що використовують певні селектори. Це дозволяє точно визначити, які елементи сторінки пов'язані з певними функціями та стилізувати їх відповідно до вимог дизайну.

Загальний підхід до створення та верстки сторінок сайту залежить від вибраної технології та потреб проекту, проте розглянуті основні концепції та інструменти допоможуть забезпечити ефективну розробку веб-додатку для управління компонентами інтернет-магазину карт пам'яті.

**Головне меню.** Головне навігаційне меню є ключовим компонентом веб-додатку, оскільки воно допомагає користувачеві орієнтуватись і знаходити необхідний функціонал. Це забезпечує користувачеві розуміння, яким саме веб-ресурсом він користується, і через нього він здійснює основну роботу на сайті.

Головне меню було створене в програмі Index.html, базовою мовою на Visual Studio Code.

**Поле пошуку.** Поле пошуку є важливим інструментом для знаходження необхідної інформації. Воно дозволяє користувачеві швидко знайти потрібний товар шляхом введення пошукового запиту. Принцип його використання полягає в тому, що користувач вводить запит у поле, і веб-додаток відображає лише ті дані, які точно відповідають введеному запиту. Це значно спрощує процес пошуку необхідних даних.

Поле пошуку було створенно завдяки команди placeholer, а сам пошук відбувається завдяки командам search в частині програми app.js.

**Блок сортування.** Сортування та фільтрування є невід'ємною частиною сучасного веб-додатку. Цей блок виводить вниз карти, які відповідають бажанням користувача. Користувач може вибрати певні критерії, за якими веб-додаток динамічно сортує та відображає кожен елемент. Це дозволяє швидко вибрати серед компонентів карт пам’яті.

**Блок додавання.** Блок додавання карт пам'яті є основним елементом для заповнення списку товарів. Користувачу потрібно вказати інформацію в певні поля, що дозволяє додати карту пам'яті до списку товарів і використовувати її для подальшого пошуку та сортування.

Додавання виконується завдяки команді function addproduct getElementById

**Блок видалення, редагування компонентів.** Блок видалення карт пам'яті надає користувачеві можливість видаляти карти пам'яті зі списку товарів. Після видалення, карта повністю зникає зі списку і може бути відновлена лише за допомогою блока додавання.

Видалення відбувається завдяки команді deleteProduct в app.js, який має доступ до Index.html звідки і видаляє компонент

Редагування відкриває можливість вносити зміни вже введених даних про карту пам'яті, яку вказав користувач. Це дозволяє виправити помилки, допущені при введенні, або додати нову карту без необхідності використовувати блок додавання. Таким чином, редагування надає можливість коригувати та оновлювати існуючі дані про карту пам'яті, що сприяє точності та актуальності інформації в базі даних.

Редагування відбувається завдяки команді editProduct, вона як і видалення має доступ до html файлу, компонент в якому і проходить через редагування.

**Список карт пам’яті.** Цей функціональний елемент відображає всі картки пам'яті, інформація про які була введена в веб-додаток. Його візуальне представлення дозволяє взаємодіяти з картками пам'яті, здійснюючи редагування, сортування та видалення після їх введення в додаток. У цьому списку міститься повна інформація, яку надав користувач, включаючи виробника, модель, тип, Speed Class, а також максимальну швидкість читання та запису. Цей функціональний елемент дозволяє зручно переглядати та керувати всіма даними про картки пам'яті, які були внесені в додаток.

## Програмування сайту

При розробці веб-додатка для управління компонентами інтернет-магазину карт пам'яті, важливо організувати функціонал веб-сайту відповідно до його структури, розділивши програмний код на окремі файли, які виконують певні функціональні завдання. Це дозволить забезпечити більш зручне та ефективне супроводження додатка у майбутньому. При цьому важливо, щоб структура файлів веб-додатка була зрозумілою, а назви файлів відображали їх функціональне призначення. Наголошується, що усі скрипти були написані у редакторі коду Visual Studio Code.

Для веб-додатка "MemoryCards" було створено структуру сценаріїв, яка описана нижче.

**Додавання нового товару:**

* За допомогою addProductForm.addEventListener('submit', function(event) {...}) відбувається відслідковування події подання форми.
* Функція addProduct() отримує значення полів форми та створює об'єкт product.
* Об'єкт product додається до масиву products.
* Масив products зберігається у локальному сховищі (localStorage).
* Функція addProductForm.reset() очищає поля форми.
* Функція updateProductList() оновлює список товарів.

**Редагування товару:**

* Функція editProduct(index) відображає вікно спливаючого повідомлення зі змінними значеннями товару.
* Значення товару оновлюються на основі введених даних.
* Оновлений масив products зберігається у локальному сховищі.
* Функція updateProductList() оновлює список товарів.

**Видалення товару:**

* Функція deleteProduct(index) відображає вікно спливаючого повідомлення для підтвердження видалення товару.
* При підтвердженні, товар видаляється з масиву products.
* Оновлений масив products зберігається у локальному сховищі.
* Функція updateProductList() оновлює список товарів.

**Пошук товару:**

* За допомогою searchInput.addEventListener('input', function() {...}) відбувається відслідковування введення користувачем значення у поле пошуку.
* Введене значення перетворюється у нижній регістр (toLowerCase()).
* Масив products фільтрується на основі введеного значення.
* Функція updateProductList() оновлює список товарів з відфільтрованими результатами.

**Сортування товарів:**

* За допомогою sortSelect.addEventListener('change', function() {...}) відбувається відслідковування зміни вибраного значення в полі сортування.
* Значення сортування зберігається у змінній sortBy.
* Масив products сортується за вибраним полем.
* Функція updateProductList() оновлює список товарів з відсортованими результатами.

**Очищення списку товарів:**

* Функція clearButton.addEventListener('click', function() {...}) відслідковує натискання кнопки очищення.
* Відображається вікно спливаючого повідомлення для підтвердження очищення списку.
* При підтвердженні, масив products очищається, і його збереження у локальному сховищі видаляється.
* Список товарів очищається на сторінці.

**Оновлення списку товарів:**

* Функція updateProductList() очищує список товарів на сторінці.
* Для кожного товару у масиві products створюється відповідний елемент div з необхідними дочірніми елементами.
* Елементи товарів додаються до списку товарів на сторінці.

Отже, структура сценаріїв веб-додатку складається з семи функціональних файлів, які розділяють загальний програний код

## Тестування сайту

Цей розділ детально описує процес тестування кінцевого веб-додатку. Під час тестування було виявлено й виправлено незначні помилки, пов'язані з версткою та програмуванням веб-додатку. В процесі тестування були перевірені різні частини та аспекти програмного продукту.

Тестування складалося з таких етапів:

1. **Функціональне тестування:**

* Перевірка користувацьких форм, форм пошуку та сортування, посилань тощо.
* Перевірка навігаційної панелі та посилань.
* Перевірка функції пошуку за допомогою введення дивної послідовності символів, яка не має збігів.
* Перевірка функції пошуку за введенням параметру, що точно збігається з існуючими елементами.
* Перевірка функціоналу сортування та підрахунку певної властивості.

1. **Тестування на сумісність:**

* Перевірка вигляду веб-додатку на різних браузерах, таких як Google Chrome, Mozilla Firefox, Internet Explorer.
* Перевірка відображення веб-додатку на різних пристроях, включаючи мобільні пристрої з малими екранами.

1. **Тестування стійкості веб-додатку до помилок користувача:**

* Перевірка реакції додатку на передбачувані дії користувача.
* Перевірка реакції додатку при залишенні полів незаповненими.

**Функціональне тестування.**

Проведене тестування навігаційної панелі та функції пошуку веб-додатку показало наступні результати:

Навігаційна панель: Під час перевірки кожного посилання в головному меню було виявлено, що вони працюють коректно.

Функція пошуку: Виконана перевірка функції пошуку веб-додатку. Спроба ввести дивну послідовність символів, яка не має збігів з існуючими елементами, не видала жодного результату, що свідчить про правильну роботу функції пошуку. Далі, при введенні параметру, який точно збігається з існуючими елементами (наприклад, виробника), результат пошуку показав два компоненти з такими збігами. Це підтверджує, що функція пошуку працює коректно.

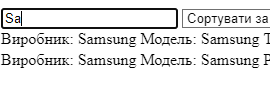


Рисунок 2.9 – Результат перевірки функції пошуку.

Протестуємо роботу кнопок, для відкриття модального вікна веб-додатку. При активації пункту “Додати”, отримаємо повідомлення про пусте поле, що пропонує користувачеві ввести дані, для створення нової карти пам’яті. Натиснувши кнопку “Редагувати” на довільному компоненті карти, отримаємо модальне вікно, в яку нам веб-дотаток запропонує вести нову інформацію, при введені пустого поля виведе null. Отже, функціонал кнопок для взаємодії з веб-дотаткой працює нормально.

Тепер перевіримо роботу функціоналу сортування. Для цього в меню сортування виберемо – Сортувати за Об’ємом пам’яті, веб-додаток повинен вивести вниз карти с найбільшим об’ємом.

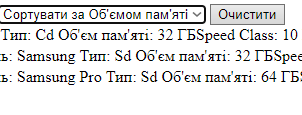


Рисунок 2.10 – Результат перевірки функцій сортування.

**Тестування на сумісність.** Тестування сумісності веб-додатку включає два етапи: перевірку кросбраузерності, щоб переконатися, що веб-сайт вірно відображається в більшості браузерів, і перевірку відображення на різних пристроях, зокрема на смартфонах з маленькими екранами.

Перевірка кросбраузерності включала запуск веб-додатку на кількох вибраних браузерах, таких як Google Chrome, Mozilla Firefox і Internet Explorer. Під час запуску на цих браузерах зовнішній вигляд сайту залишався практично незмінним, що свідчить про його належне відображення.

Другий етап тестування передбачав запуск веб-додатку на різних пристроях. Один з пристроїв, який був залучений до тестування, був телефон, також був вибраний ноутбук. Щоб емулювати запуск веб-додатку на мобільному пристрої, при цьому був використаний браузер Google Chrome. Після запуску веб-додатку було зроблено висновок, що він коректно відображається на комп’ютері та ноутбуці, але на телефоні виникла проблема с відображенням тексту, оскільки він був розрахований на англійський текст, а не на український.

Це означає що відображення було би нормальним, лише якби інтерфейс був написанний англійською. Про це свідчить «Speed Class» в якому не має жодних змін.

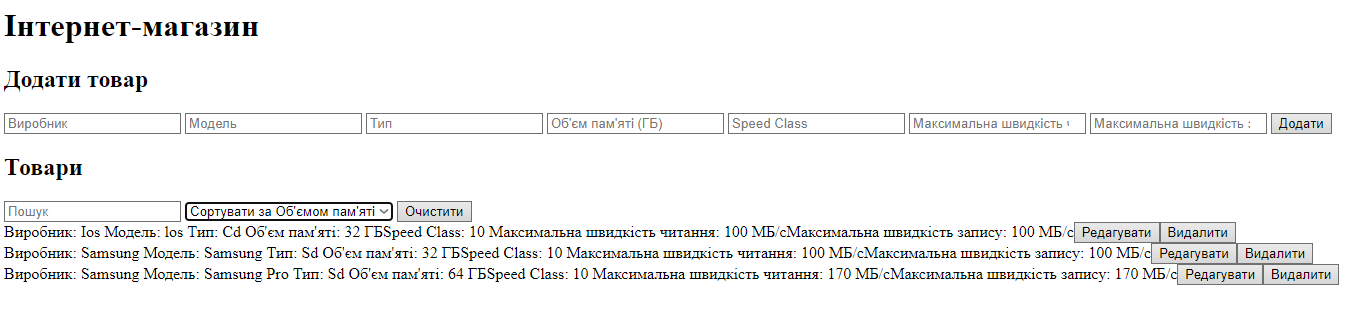


Рисунок 2.11 – Результат тестування додатку на комп’ютері та ноутбуці

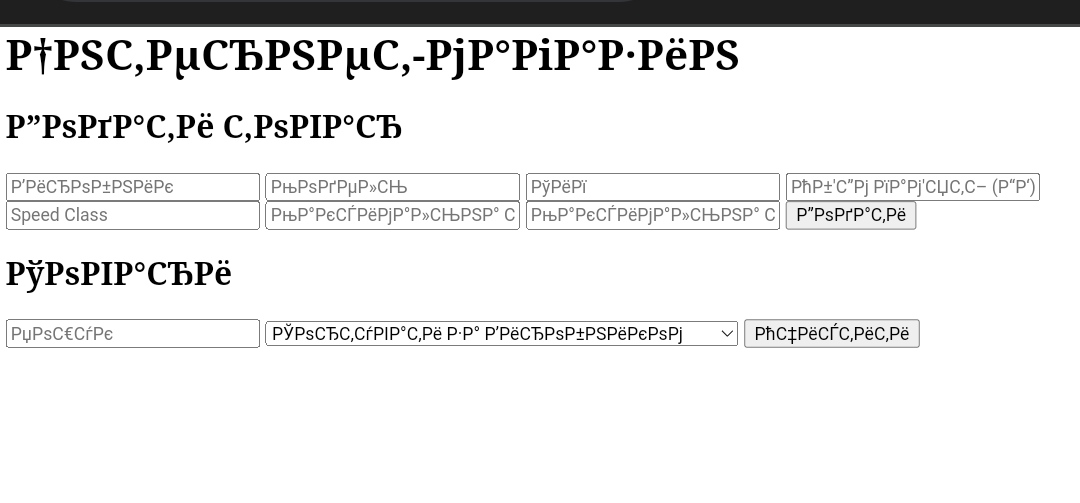


Рисунок 2.12 – Результат тестування додатку на телефоні

Отже, результати тестування сумісності свідчать про те, що веб-додаток успішно пройшов перевірку на кросбраузерність, проте має проблеми с відображенням на телефоні.

**Тестування стійкості до помилок користувача.** Тут перевірем реакцію сайту на не передбачувані дії від користувача. Перевіримо роботу компонентів, які найбільш вразливі до подібних дій.

Головний компонент, що підлягає тестуванню на цьому етапі – це модальне вікно. Для початку спробуємо заповнити поля форми невластивою їм інформацією, або інформацією у неправильному форматі. Результатом такої передбачуваної дії є неможливість виклику подібної помилки, оскільки веб-додаток не дозволяє вписувати символи або букви, в тих полях де повинні бути лише цифри.

Тепер спробуємо залишити поля незаповненими і подивимось на те, як відреагує додаток на цей крок. Виникло повідомлення, яке говорить про те, що вказаний додаток пустий, і має мати хоть якесь значення

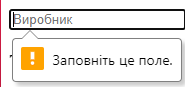


Рисунок 2.13 – Повідомлення про відсутність інформації.

# СПЕЦІАЛЬНИЙ РОЗДІЛ

## Розміщення сайту в інтернеті

Процедура розміщення веб-сайту у мережі інтернет складається з кількох послідовних етапів. Для початку необхідно визначитися із доменним ім’ям та доменною зоною. Доменне ім’я дасть змогу користувачам мережі інтернет потрапити на необхідний веб-ресурс, а завдяки правильно вибранній доменній зоні, пошуковики будуть віддавати пріоритет веб-ресурсу. Аби веб-сайт працював безперебійно, необхідно спеціальне, зарезервоване місце на віддаленому сервері. Такі зарезервовані місця надають хостингові сервіси. Наступним кроком необхідно обрати хостинговий сервіс, що надасть усі необхідні послуги для утримання та обслуговування веб-сайту.

Слід почати з вибору доменної зони. Необхідно проаналізувати, на який тип корстувачів буде орієнтуватися веб-додаток “Phones”. Так як інтерфейс веб-додатку написаний англійською мовою, то можна сказати, що веб-ресурс має бути орієнтований на користувачів зразу із декількох країн. Тому, доцільно вибрати загальну доменну зону – com. Після аналізу та вибору доменної зони потрібно обрати доменне ім’я, тобто та частина адреси, яка стоїть зліва від доменної зони. Коротке та просте доменне ім’я – це запорука популярності веб-ресурсу, саме тому необхідно добре проаналізувати усі можливі варіанти. Крім цього, ще однією умовою для доменного ім’я повинна бути зрозумілість. Тобто доменне ім’я повинно відображати суть та призначення веб-додатку. Таким чином користувач точно буде розуміти, що даний веб-ресурс – це саме те, що йому необхідно. Підберемо декілька варіантів доменних імен, тоді проаналізуємо кожне з них, звузимо коло варіантів до одного екземпляру.

Варіанти майбутнього доменного імені для веб-додатку “Phones”:

1. phones;
2. phones-designer;
3. phones-builder.

Розглянемо доменне ім’я “phones”. Це перше, що спадає на думку, коли мова йде про веб-ресурс, що напряму пов’язаний із мобільними пристроями. Однак, ця назва практично ні про що не говорить користувачеві. Лише про те, що веб-сайт якось пов’язаний зі смартфонами. Відкидаємо цей варіант.

Далі розглянемо ім’я “phones-designer”. Це значно кращий варіант у порівнянні із попереднім, однак розглянемо детальніше це ім’я. Друге слово “designer” не зовсім підходить під опис додатку “Phones”, адже ми не обираємо дизайн для смартфонів. Тож цей варіант також не підходить.

Останній варіант “phones-builder” чудово підходить для веб-додатку “Phones”, адже він короткий, а призначення сайту одразу стає зрозумілим для користувача.

Отже, доменне ім’я та доменна зона разом формують загальну адресу веб-ресурсу. Адреса веб-додатку “Phones” – phones-builder.com.

Вибравши адресу веб-сайту, необхідно зареєструвати її, а також розмістити розроблений сайт на хостинг.

У пошуковій системі браузера, спробуємо знайти хостинг-сервіс. Зупинимось на сервісі hostiq.ua. Вибір данного хостинг-сервісу обгрунтований тим, що його сервера знаходяться найблище до Європи. Тепер необхідно перевірити, чи доступна адреса для реєстрації. Перевіривши, дізнаємося, що домен доступний для реєстрації. Тепер спробуємо розмістити веб-додаток на хостинг. Після вдалого розміщення веб-ресурсу на хостинг, необхідно перевірити його роботу за зареєстрованим доменним ім’ям. Робимо висновок, що веб-ресурс корректно працює за новим доменним ім’ям.

## Інструкція з обслуговування та наповнення сайту

Адміністративна частина веб-додатку “Phones” складається із кількох інших частин. Насамперед, це – файл сервера, на якому працює додаток. Він містить дерективи ініціалізації сервера та підключення його до бази даних. Друга частина – це сама база даних, яка містить усі дані користувачів веб-додатку, тобто компоненти мобільних пристроїв. Третя частина – це безпосередній контент веб-додатку, тобто набір файлів із розширеннями .html, .css, .js.

Аби додати новий серверний функціонал, адміністратору слід робити нові внесення саме у функціональному файлі index.js. Саме у ньому відбуваються усі процеси, що пов’язані із серверною частиною, включаючи підключення до бази даних, порт робочого сервера, тощо. Веб-додаток “Phones” містить підключення лише до однієї бази даних. Веб-додаток “Phones” використовує об’єктну базу даних MongoDB, проте, при необхідності, її можна замінити будь-якою іншою, наприклад – реляційною базою даних MySQL. Аби створити нове підключення до бази даних, або змінити наявну базу даних, потрібно встановити необхідні модулі для Node.js. У наявній базі даних передбачена лише одна колекція. Для створення нової колекції у функціональному файлі index.js необхідно визначити її схему, а також встановити нові поля для неї. Після створення схеми та визначення полів, необхідно створити модель бази даних, через яку і буде відбуватися подальша взаємодія із нею.

Файл головної сторінки веб-додатку “Phones” – index.html. Щоб відредагувати головну сторінку веб-ресурсу, слід робити зміни саме у цьому файлі. Веб-додаток не передбачає використання більше однієї сторінки, проте при необхідності можна визначити більше сторінок та зробити на них відповідні посилання у файлі index.html.

Файли стилізації головної сторінки знаходяться у папці з назвою “css”. Створюючи та визначаючи нові сторінки веб-додатку “Phones”, рекомендується користуватися вже визначеними стилями, аби загальна стилізація веб-сайту залишалася одинаковою. При додавані нових стилів для веб-сторінок, рекомендується використовувати саме цю папку для їх зберігання.

Основний функціонал, що не стосується бази даних та серверної частини знаходиться у папці js. Назви функціональних файлів повністю відповідають їх призначенню, тому для редагування вже існуючого функціоналу, слід вносити зміни у них. Аби додати новий функціонал, що ніяк не пов’язаний із уже існуючим, необхідно створити новий файл із назвою, що буде відповідати його призначенню.

Також слід визначити мінімальні характеристики сервера хостингу, на якому працює веб-сайт. Мінімальні харектеристики, що ставляться до хостингу, на якому розташовується веб-додаток “Phones” наведені у таблиці 3.1.

Таблиця 3.1 – Мінімальні характеристики сервера для хостингу додатку

|  |  |
| --- | --- |
| **Назва харектеристики** | **Значення характеристики** |
| Веб-сервер | Apache 1.3.0 або IIS 5.0 |
| Скриптова мова | PHP 7.2 |
| Система керуванням базами даних | MySQL 5.6 |
| Ліміт пам’яті | Не менше ніж 32 Мб |
| Час виконання скрипта | 60 секунд |

Однак, якщо функціонал веб-сайту буде зростати, відповідно навантаження на сервер також може значно зрости. Тому, при розширенні контенту веб-додатку “Phones” слід враховувати, що вимоги до сервера також потрібно підвищити.

## Інструкція з популяризації та підтримки сайту

Аби розповсюдити веб-ресурс у мережі інтернет існує безліч механізмів популяризації, що вже тривалий час використовуються. Популяризацію веб-ресурсу можна умовно поділити на внутрішню і зовнішню. Внутрішня популяризація веб-ресурсу реалізується шляхом впровадження зручного функціоналу, таким чином, користувачі, що вже відвідали веб-ресурс будуть мати стимул, аби і в подальшому працювати з ним. Зовнішня популяризація дещо відрізняється від внутрішньої. Тут мова йде про способи та механізми збільшення безпосередньої кількості відвідувань веб-ресурсу. Опишемо детальніше можливості та механізми внутрішньої та зовнішньої популяризації.

На зовнішню популяризацію впливає кілька чинників. Насамперед це простий та зрозумілий інтерфейс. Таким чином, необхідно створити умови для зручного користування інструментами веб-ресурсу, аби користувач мав достатній стимул для подальшої роботи з ним.

До внутрішньої популяризації також відноситься правильно побудована адреса веб-сайту. Правильно вибране доменне ім’я користувачі веб-ресурсу не сплутають з іншим, саме тому потрібно з обачністю його вибирати. Коротке та лаконічне – ось яким повинно бути доменне ім’я веб-ресурсу, аби зацікавити користувачів мережі інтеренет.

У свою чергу, доменна зона також впливає на популяризації веб-ресурсу. Наслідком невдало вибраної доменної зони є зменшення кількості відвідувань, саме тому правильна доменна зона – це запорука успіху веб-ресурсу.

Ще один фактор, який можна віднести до внутрішньої популяризації – це його стабільність. Ніхто з користувачів не захоче працювати із продуктом, який містить багато помилок та має безліч слабких місць, тому цей фактор також має великий вплив на популяризації веб-ресурсу.

Налогодження зворотного зв’язку на веб-русурсі також можна вважати фактором внутрішньої популяризації, тому важливо надати користувачеві можливість зв’язатися із адміністрацією сервісу. Це один із головних аспектів будь-якого достойного веб-ресурсу, адже користувач, знаючи, що він може у будь-який момент звернутись за допомогою до повноважених осіб, захоче і дальше працювати з цим веб-сайтом.

Те, що стосується безпосереднього збільшення трафіку відвідувань уже можна відносити до зовнішньої популяризації. Існує декілька перевірених механізмів просування свого веб-ресурсу у мережі інтернет, що вже тривалий час використовуються відповідними спеціалістами.

Перше, що необхідно зробити при зовнішній популяризації, це проаналізувати ключові слова веб-ресурсу. Слід звернути увагу на те, які ключові слова використовуються в інших популярних веб-ресурсах тієї ж тематики. Також необхідно звертати увагу на те, які запити здійснюють користувачі, що шукають подібні веб-ресурси. Впровадивши правильні ключові слова на веб-ресурс, трафік відвідувань повинен зрости.

Просування веб-ресурсу у соціальних мережах також впливає на трафік відвідування. Більшість популярних сайтів, крім підтримки основного веб-ресурсу, ведуть підтримку соціальних мереж. Таким чином користувачі будуть знаходити сайт не тільки у пошукових системах,а й про нього можна буде дізнатися із сторонніх джерел. Отже даний механізм популяризації також значно збільшує трафік відвідувань.

Не останнім по важливості методом популяризації є електронний маркетинг. Таким чином можна інформувати відвідувачів мережі інтернет про роботу веб-ресурсу. Аби користувачі не втратили інтерес до веб-ресурсу, час від часу необхідно відправляти користувачам електронні листи, що пов’язані з контентом веб-додатку. Відправляючи електронні бюлетені певній групі людей, потрібно зацікавити їх. Також, розсилаючи користувачам електронні листи, налагоджується зворотній зв’язок, що дає можливість ще більше збільшити трафік відвідувань.

ВИСНОВКИ

В даній курсовій роботі було поетапно описано розробку веб-додатку управління компонентами інтернет-магазину смартфонів. Розроблений веб-додаток відповідає усім поставленим вимогам.

Для розробки даного веб-додатку використовувалися базові принципи та методи розробки веб-сайтів.

Під час виконання курсової роботи було отримано цінний практичний досвід розробки веб-ресурсів, що у майбутньому повинно стати підгрунттям для створення нових проектів. Також закріплено і поглиблено знання, одержанні за час вивчення предмету «Web-технології», більш детально вивчено процес розробки програмних засобів із застосуванням таких технологій як HTML, CSS, JavaScript, Node.js.

ПЕРЕЛІК ПОСИЛАНЬ

1. W3SCHOOLS : вебсайт.URL: https://www.w3schools.com/html/
2. (дата звернення: 02.04.2022).
3. METANIT : вебсайт. URL: https://metanit.com/web/nodejs/ (дата звернення: 06.05.2022).
4. MDN WEB DOCS : вебсайт.URL: https://developer.mozilla.org/enUS/docs/Web (дата звернення: 12.05.2022).
5. REGEXONE: вебсайт.URL: https://regexone.com/ (дата звернення: 03.05.2022).

Додаток А  
Лістинг файлу «index.html»

<!DOCTYPE html>

<html>

<head>

<title>Інтернет-магазин</title>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<h1>Інтернет-магазин</h1>

<form id="addProductForm">

<h2>Додати товар</h2>

<input type="text" id="виробник" placeholder="Виробник" required>

<input type="text" id="модель" placeholder="Модель" required>

<input type="text" id="тип" placeholder="Тип" required>

<input type="number" id="обсяг" placeholder="Об'єм пам'яті (ГБ)" required>

<input type="text" id="speedClass" placeholder="Speed Class" required>

<input type="number" id="швидкістьЧитання" placeholder="Максимальна швидкість читання (МБ/с)" required>

<input type="number" id="швидкістьЗапису" placeholder="Максимальна швидкість запису (МБ/с)" required>

<button type="submit">Додати</button>

</form>

<h2>Товари</h2>

<input type="text" id="пошук" placeholder="Пошук">

<select id="сортування">

<option value="виробник">Сортувати за Виробником</option>

<option value="модель">Сортувати за Моделлю</option>

<option value="обсяг">Сортувати за Об'ємом пам'яті</option>

</select>

<button id="очистити">Очистити</button>

<div id="списокТоварів"></div>

<script src="app.js"></script>

</body>

</html>

Додаток Б  
Лістинг файлу «buttons.css»

.editbtn{

    border: 1px solid rgb(75, 212, 75);

    border-radius: 10px;

    margin: 10px;

    padding:10px;

    padding-left: 20px;

    padding-right: 20px;

    color: rgb(75, 212, 75);

}

.deletebtn{

    border: 1px solid rgb(216, 100, 65);

    border-radius: 10px;

    margin: 10px;

    padding:10px;

    padding-left: 20px;

    padding-right: 20px;

    color: rgb(216, 100, 65);

}

.editbtn:hover,.deletebtn:hover{

    color: skyblue;

    border-color: skyblue;

}

Додаток В  
Лістинг файлу «components.css»

.component{

    background-color: white;

    border:3px;

    border-style: solid;

    border-color: NavajoWhite;

    border-radius: 10px;

    width: auto;

    margin: 10px;

    text-align: center;

    transition: 0.6s ease-in-out;

}

.component:hover{

    background-color: lightgray;

    transition: 0.3s ease-in;

}

p.info{

    font-family: 'Geneva', 'Verdana', 'sans-serif';

    text-align: center;

}

p.producerandmodel{

    font-weight: bold;

    font-family: 'Geneva', 'Verdana', 'sans-serif';

    text-align: center;

}

.picture{

    width: 300px;

    height: 300px;

    margin-top: 10px;

    margin-bottom: 10px;

    display: block;

    margin-left: auto;

    margin-right: auto;

    padding: 5px;

}

.hiden{

    transform: scale(0%);

}

.hiden2{

    display: none;

}

Додаток Г  
Лістинг файлу «inputs.css»

.external-fix, .form-field, .toggler-button, .search-button, .search-line{

    border: none;

    outline: none;

    -webkit-appearance: none;

}

.button-border{

    border: 1px solid black;

    border-radius: 10px;

}

.elementHoverEffects, .form-field:hover, .search-block:hover{

    -webkit-transition: 0.3s all;

    transition: 0.3s all;

    border: 3px solid skyblue;

    border-radius: 5px;

}

.search-block{

    -webkit-transition: 0.3s all;

    transition: 0.3s all;

    border: 1px solid black;

    border-radius: 5px;

}

.search-button{

    width: 30px;

    background: url(../src/images/search\_icon.png) no-repeat center;

    background-color: white;

}

.toggler-button{

    width: 50px;

    height: 50px;

    background: url(../src/images/toggler\_icon.png) no-repeat center;

    background-color: NavajoWhite;

}

.sort-functions-block, .count-functions-block{

    border: 3px solid lightgray;

    border-radius: 10px;

}

.form-field{

    -webkit-transition: 0.3s all;

    transition: 0.3s all;

    border: 2px solid lightgray;

    border-radius: 10px;

    text-align: center;

}

input:invalid{

    border: red solid 3px;

}

input:invalid:focus{

    border: red solid 3px;

}

Додаток Д  
Лістинг файлу «modal.css»

.modal{

    display: flex;

    position: fixed;

    align-items: center;

    justify-content: center;

    width: 100%;

    height: 100%;

    transform: translateY(-50%) translateX(50%) scale(0);

    transition: 0.3s ease-in-out;

}

.active{

    transform: translateY(0%) translateX(0) scale(100%);

}

.overlay{

    position: fixed;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    z-index: 0;

    background-color: rgba(0,0,0,0.5);

}

.modalwindow{

    position: relative;

    text-align: center;

    top: 0;

    left: 0;

    width: 50%;

    height: 80%;

    border: 1px solid black;

    border-radius: 10px;

    overflow: scroll;

    scroll-behavior: smooth;

    background-color: white;

}

.modalwindow-phone{

    width: 90%;

}

.close{

    display: flex;

    position: absolute;

    justify-content: center;

    right: 10px;

    top: 10px;

    width: 20px;

    height: 25px;

    border: 1px solid black;

    border-radius: 5px;

    font-weight: bold;

    background-color: lightgray;

}

.close:hover{

    cursor: pointer;

    background-color: gray;

}

.modal-window-minuature{

    display: block;

    width: 40%;

    height: auto;

    margin-left: auto;

    margin-right: auto;

    margin-top: 10px;

    margin-bottom: 10px;

}

.modal-header-sign{

    font-weight: bold;

    color: gray;

}

.page-scroll-off{

    overflow: hidden;

}

/\* ПРИХОВАТИ SCROLLBAR У МОДАЛЬНОГО ВІКНА\*/

.modalwindow::-webkit-scrollbar { width: 0; }

/\* ПРИХОВАТИ SCROLLBAR НА СТОРІНЦІ\*/

body::-webkit-scrollbar { width: 0; }

Додаток Е  
Лістинг файлу «navigation.css»

html{

    height: 100%;

}

html body{

    height: 100%;

    display: -webkit-box;

    display: -ms-flexbox;

    display: flex;

    -webkit-box-orient: vertical;

    -webkit-box-direction: normal;

    -ms-flex-direction: column;

    flex-direction: column;

}

.content{

    -webkit-box-flex: 1;

    -ms-flex: 1 0 auto;

    flex: 1 0 auto;

    display: -webkit-box;

    display: -ms-flexbox;

    display: flex;

    -ms-flex-wrap: wrap;

    flex-wrap: wrap;

    -webkit-box-pack: center;

    -ms-flex-pack: center;

    justify-content: center;

    -webkit-box-orient: horizontal;

    -webkit-box-direction: normal;

    -ms-flex-direction: row;

    flex-direction: row;

}

.main-navbar-theme{

    background-color: NavajoWhite;

    border-bottom: 1px solid lightgray;

}

.main-navbar-theme ul li a, .navbar-menu-items-text{

    color: black;

    -webkit-transition: 0.3s all;

    transition: 0.3s all;

}

.main-navbar-theme ul li a:hover, .navbar-menu-items-text:hover{

    color: white;

    -webkit-transition: 0.3s all;

    transition: 0.3s all;

}

.navbar-menu-items-text{

    font-family: "Gill Sans", "Gill Sans MT", "Calibri", "Trebuchet MS", "sans-serif";

    font-weight: bold;

}

.footer-theme{

    background-color: NavajoWhite;

    border-top: 1px solid lightgray;

}

Додаток Ж  
Лістинг файлу «count.js»

//COUNT FUNCTIONS CODE

function countСertainField(value){

    let countedFields = document.querySelectorAll(`p[value='${value}']`);

    return Object.values(countedFields)

    .map(p=>p.innerHTML.match(/(\d+\.\d+|\d+)/)[0]).reduce((n1,n2)=>Number(n1) + Number(n2),0)

    .toFixed(1);

}

function countComponents(){

    return document.getElementsByClassName("component").length;

}

function displayCertainPropertySum(){

    let span = document.getElementById("countFrame");

    let countFlag = document.getElementById("count").value;

    span.innerHTML = `Sum of all component's ${countFlag} is ${countСertainField(countFlag)}`;

}

function displayComponentsCount(){

    let span = document.getElementById("countFrame");

    span.innerHTML = `Number of all components is ${countComponents()}`;

}

//EVENT HANDLERS

document.getElementById("propertyCountBtn").addEventListener("click",displayCertainPropertySum);

document.getElementById("componentsCountBtn").addEventListener("click",displayComponentsCount);

Додаток И  
Лістинг файлу «index.js»

//SERVER AND MONGODB CODE

const express = require("express");

const mongoose = require("mongoose");

const { createServer } = require("http");

const app = express();

const serverPort = 3000;

var cors = require("cors");

app.use(cors());

app.use(express.json({ type: "\*/\*" }));

const dataBaseUrl = "mongodb+srv://admin:admin123@cluster0.ok61k.mongodb.net/Kursova?retryWrites=true&w=majority";

mongoose.connect(dataBaseUrl, { useNewUrlParser: true })

  .then(() => console.log("MongoDb connected"))

  .catch(err => console.log(err));

const componentsScheme = new mongoose.Schema({

  brand: {

    type: String,

    required: true,

  },

  model: {

    type: String,

    required: true,

  },

  diagonal: {

    type: Number,

    required: true,

  },

  cpu: {

    type: String,

    required: true,

  },

  ram: {

    type: String,

    required: true,

  },

  rom: {

    type: String,

    required: true,

  },

  icon:{

    type: String,

    required: true,

  }

});

const Components = mongoose.model("Components", componentsScheme);

app.get("/components", (req, res) => {

    Components.find().then(array=> res.send(array)).catch(err => res.send(err));

});

app.post("/components/new",(req,res)=>{

        req = req.body;

        Components.create({

        brand: req.brand,

        model: req.model,

        cpu: req.cpu,

        diagonal: Number(req.diagonal),

        ram: req.ram,

        rom: req.rom,

        icon: req.icon

      }).then(obj=>res.send(obj)).catch(err=> res.send(err));

});

app.post("/components/find",(req,res)=>{

  req = req.body;

  Components.findById(req.\_id).then(obj=>res.send(obj)).catch(err=> res.send(err));

});

app.post("/components/delete",(req,res)=>{

  req = req.body;

  Components.deleteOne({\_id: req.\_id}).then(obj=>res.send(obj)).catch(err=> res.send(err));

});

app.post("/components/change",(req,res)=>{

  req = req.body;

  Components.updateOne({\_id: req.\_id},req).then(obj=>res.send(obj)).catch(err=> res.send(err));

});

const server = createServer(app);

server.listen(serverPort, () => console.log(`Server is ready!\nPort: ${serverPort}`));

Додаток К  
Лістинг файлу «input-validation.js»

//INPUTS VALIDATION CODE

function setInputsRestrictions(){

    let inputs = document.querySelectorAll("form[name='modal\_form'] input[type='text']");

    inputs[0].pattern = "[A-Za-z]{3,15}";

    inputs[2].pattern = "([0-9][0-9]?\\.[0-9]|[0-9][0-9]?)";

    inputs[3].pattern = "[A-Za-z0-9 ]{3,20}";

}

setInputsRestrictions();

Додаток Л  
Лістинг файлу «main.js»

//MODAL AND SERVER REQUESTS CODE

let modal = document.getElementById("mod");

let overlay = document.getElementById("over");

let close = document.getElementById("closeBtn");

let modalD = document.getElementById("modalDialog");

let miniatureFrame = document.getElementById("miniature-frame");

let modalForm = document.querySelector("form[name='modal\_form']");

//ВІДКРИТТЯ МОДАЛЬНОГО ВІКНА

function openModal(e){

    e.preventDefault();

    modal.classList.add("active");

    document.body.classList.add("page-scroll-off");

}

function openModalAsFunction(){

    modal.classList.add("active");

    document.body.classList.add("page-scroll-off");

}

//ВІДКРИТТЯ МОДАЛЬНОГО ВІКНА

//ЗАКРИТТЯ ТА ОЧИЩЕННЯ МОДАЛЬНОГО ВІКНА

function closeAndClearModal(){

    modal.classList.remove("active");

    document.body.classList.remove("page-scroll-off");

    clearForm();

}

//ЗАКРИТТЯ ТА ОЧИЩЕННЯ МОДАЛЬНОГО ВІКНА

//ОТРИМАННЯ ДАНИХ З ФОРМИ

function retrieveDataFromForm(){

    const {brand,model,diagonal,cpu,ram,rom,icon} = modalForm;

    let formData = {

        brand: brand.value,

        model: model.value,

        diagonal: diagonal.value,

        cpu: cpu.value,

        ram: ram.value,

        rom: rom.value,

        icon: icon.src

    };

    return formData;

}

//ОТРИМАННЯ ДАНИХ З ФОРМИ

//ПЕРЕВІРКИ ВАЛІДНОСТІ КАРТИНКИ

function setNoImageIfFileInvalid(fileName){

    if(!/.+\.jpg$/.test(fileName) && !/.+\.png$/.test(fileName)){

        miniatureFrame.src = "src/images/noImage\_photo.png";

        return false;

    } else return true;

}

//ПЕРЕВІРКИ ВАЛІДНОСТІ КАРТИНКИ

//ВІДОБРАЖЕННЯ ІКОНКИ У МОДАЛЬНОМУ ВІКНІ

function previewFile(e) {

    let file = e.target.files[0];

    let reader  = new FileReader();

    reader.onloadend = function () {

    miniatureFrame.src = reader.result;

    }

    if(!setNoImageIfFileInvalid(file.name)) return;

    reader.readAsDataURL(file);

}

//ВІДОБРАЖЕННЯ ІКОНКИ У МОДАЛЬНОМУ ВІКНІ

//ГЕНЕРУВАННЯ HTML КОМПОНЕНТА

function createNewComponent(formData){

    const newComponent = document.createElement("div");

    newComponent.id = formData.\_id;

    newComponent.classList.add("component");

    newComponent.insertAdjacentHTML("afterbegin",`

    <img class = "picture" src="${formData.icon}" alt = "phone\_picture">

    <p value = "brand" class="producerandmodel">${formData.brand}</p>

    <p value = "model" class="producerandmodel">${formData.model}</p>

    <p value = "diagonal" class="info">Diagonal: ${formData.diagonal}</p>

    <p value = "cpu" class="info">CPU: ${formData.cpu}</p>

    <p value = "ram" class="info">RAM: ${formData.ram}</p>

    <p value = "rom" class="info">ROM: ${formData.rom}</p>

    <input type="button" class = "editbtn" value="Edit" onclick="getDocumentFromDB(this.parentNode.id)">

    <input type="button" class = "deletebtn" value="Delete" onclick="deleteComponentFromDB(this.parentNode.id)">

    `);

    return newComponent;

}

//ГЕНЕРУВАННЯ HTML КОМПОНЕНТА

//ЗАПОВНЕННЯ ФОРМИ ПРИ РЕДАГУВАННІ ВІКНА

function fillFieldsWithComponentData(formData){

    let texts = document.querySelectorAll("form[name='modal\_form'] input");

    let selects = document.querySelectorAll("form[name='modal\_form'] select");

    texts[0].value = formData.brand;

    texts[1].value = formData.model;

    texts[2].value = formData.diagonal;

    texts[3].value = formData.cpu;

    selects[0].value = formData.ram;

    selects[1].value = formData.rom;

    miniatureFrame.src = formData.icon;

}

//ЗАПОВНЕННЯ ФОРМИ ПРИ РЕДАГУВАННІ ВІКНА

//ВІДКРИТИ МОДАЛЬНЕ ВІКНО У РЕЖИМІ РЕДАГУВАННЯ

function openEditWindow(formData){

    fillFieldsWithComponentData(formData);

    function tempFunction(e){

        processTheEditForm(e,formData.\_id);

    }

    modalForm.removeEventListener("submit",processTheCreateForm);

    modalForm.addEventListener("submit",tempFunction);

    overlay.addEventListener("click",function(e){

        modalForm.removeEventListener("submit",tempFunction);

        modalForm.addEventListener("submit",processTheCreateForm);

    });

    close.addEventListener("click",function(e){

        modalForm.removeEventListener("submit",tempFunction);

        modalForm.addEventListener("submit",processTheCreateForm);

    });

    openModalAsFunction();

}

//ВІДКРИТИ МОДАЛЬНЕ ВІКНО У РЕЖИМІ РЕДАГУВАННЯ

//ОЧИЩЕННЯ ФОРМИ ПІСЛЯ РЕДАГУВАННЯ

function clearForm(){

    Object.values(document.querySelectorAll("form[name='modal\_form'] input[type='text']"))

    .forEach(p=>p.value = "");

    document.getElementById("emptyMessage").innerHTML = "";

    document.querySelectorAll("form[name='modal\_form'] select")[0].value = "2 GB";

    document.querySelectorAll("form[name='modal\_form'] select")[1].value = "16 GB";

    miniatureFrame.src = "src/images/noImage\_photo.png";

    document.getElementById("miniature").value = "";

    modalD.scrollTo(0,0);

}

//ОЧИЩЕННЯ ФОРМИ ПІСЛЯ РЕДАГУВАННЯ

//ПЕРЕВІРКА ЧИ ПУСТІ ПОЛЯ ФОРМИ

function emptyCheck(){

    let inputs = document.querySelectorAll("form[name='modal\_form'] input[type='text']");

    if(Object.values(inputs).filter(i=>i.value.length === 0).length > 0){

    document.getElementById("emptyMessage").innerHTML = "You must fill all fields!";

    modalD.scrollTo(0,0);

    return false;

    } else return true;

}

//ПЕРЕВІРКА ЧИ ПУСТІ ПОЛЯ ФОРМИ

//ОБРОБКА ФОРМИ МОДАЛЬНОГО ВІКНА ПРИ ДОДАВАННІ КОМПОНЕНТА

function processTheCreateForm(e){

    e.preventDefault();

    if(!emptyCheck()) return;

    let formData = retrieveDataFromForm();

    writteNewComponentToDB(formData);

    closeAndClearModal();

}

//ОБРОБКА ФОРМИ МОДАЛЬНОГО ВІКНА ПРИ ДОДАВАННІ КОМПОНЕНТА

//ОБРОБКА ФОРМИ МОДАЛЬНОГО ВІКНА ПРИ РЕДАГУВАННІ КОМПОНЕНТА

function processTheEditForm(e,id){

    e.preventDefault();

    if(!emptyCheck()) return;

    let formData = retrieveDataFromForm();

    formData.\_id = id;

    changeComponentFromDB(formData);

    closeAndClearModal();

}

//ОБРОБКА ФОРМИ МОДАЛЬНОГО ВІКНА ПРИ РЕДАГУВАННІ КОМПОНЕНТА

//АДАПТАЦІЯ МОДАЛЬНОГО ВІКНА ПІД ПЕВНИЙ ПРИСТРІЙ

function adaptieeModalWindow(){

    let len = document.getElementsByClassName("modalwindow").length;

    let modals = document.getElementsByClassName("modalwindow");

    for(let i = 0; i < len; i++) {

        if(this.innerWidth <= 600) modals[i].classList.add("modalwindow-phone");

        else modals[i].classList.remove("modalwindow-phone");

    }

}

//АДАПТАЦІЯ МОДАЛЬНОГО ВІКНА ПІД ПЕВНИЙ ПРИСТРІЙ

//WORK WITH REQUESTS TO SERVER

function readFromDB(){

    const xhr = new XMLHttpRequest();

    xhr.open("GET","http://localhost:3000/components");

    xhr.onload = () => {

        JSON.parse(xhr.response).forEach(c=>document.getElementById("componentsContainer").append(createNewComponent(c)));

    }

    xhr.send();

}

function writteNewComponentToDB(component){

    const xhr = new XMLHttpRequest();

    xhr.open("POST","http://localhost:3000/components/new");

    xhr.setRequestHeader("Content-Type","application/json");

    xhr.onload = () => {

        if(xhr.status === 413){

             alert("The request is overloaded.\nThe image size must be lesser than 35 KB");

             return;

        }

        else if(xhr.status >= 400){

         alert("Something wrong!.\nComponent wasn't added!\nPage will reload.");

         location.reload();

        }

        else if(xhr.status === 200) alert("Component was succesful added!");

        document.getElementById("componentsContainer").append(createNewComponent(JSON.parse(xhr.response)));

        sortComponents();

    }

    xhr.send(JSON.stringify(component));

}

function changeComponentFromDB(component){

    const xhr = new XMLHttpRequest();

    xhr.open("POST","http://localhost:3000/components/change");

    xhr.setRequestHeader("Content-Type","application/json");

    xhr.onload = () => {

        if(xhr.status === 413){

             alert("The request is overloaded.\nThe image size must be lesser than 35 KB");

             return;

        }

        else if(xhr.status >= 400){

         alert("Something wrong!.\nPage will reload.");

         location.reload();

        }

        let result = JSON.parse(xhr.response);

        if(result.acknowledged && JSON.parse(xhr.response).modifiedCount > 0){

        alert("Request successfully processed.Component was changed.Page will reload.");

        location.reload();

        }

        else if(result.acknowledged && JSON.parse(xhr.response).modifiedCount === 0){

        alert("Request successfully processed but component wasn't changed.Page will reload.");

        location.reload();

        }

        else if(!result.acknowledged){

        alert("Error. Component wasn't changed.Page will reload.");

        location.reload();

        }

    }

    xhr.send(JSON.stringify(component));

}

function deleteComponentFromDB(id){

    if(!confirm("Are you sure?")) return;

    const xhr = new XMLHttpRequest();

    xhr.open("POST","http://localhost:3000/components/delete");

    xhr.setRequestHeader("Content-Type","application/json");

    xhr.onload = () => {

        let deletedInfo = JSON.parse(xhr.response);

        if(deletedInfo.acknowledged === true){

        alert("Component was deleted! Page will reload.");

        location.reload();

        }

        else alert("Something wrong, component wasn't deleted.Try again");

    }

    xhr.send(JSON.stringify({\_id:id}));

}

function getDocumentFromDB(id){

    const xhr = new XMLHttpRequest();

    xhr.open("POST","http://localhost:3000/components/find");

    xhr.setRequestHeader("Content-Type","application/json");

    xhr.onload = () => {

        openEditWindow(JSON.parse(xhr.response));

    }

    xhr.send(JSON.stringify({\_id:id}));

}

//EVENT HANDLERS

modalForm.addEventListener("submit",processTheCreateForm);

window.addEventListener("load",readFromDB);

document.getElementById("miniature").addEventListener("change",previewFile);

document.getElementById("createPhone").addEventListener("click",openModal);

overlay.addEventListener("click",closeAndClearModal);

close.addEventListener("click",closeAndClearModal);

window.addEventListener("load",adaptieeModalWindow);

window.addEventListener("resize",adaptieeModalWindow);

Додаток М  
Лістинг файлу «sort-search.js»

//SORT AND SEARCH CODE

function sortAsNumber(components,value){

    components.sort(function(c1,c2){

        c1 = c1.querySelector(`p[value='${value}']`).innerHTML;

        c2 = c2.querySelector(`p[value='${value}']`).innerHTML;

        c1 = c1.match(/(\d+\.\d+|\d+)/)[0];

        c2 = c2.match(/(\d+\.\d+|\d+)/)[0];

        return Number(c1) - Number(c2);

    });

    return components;

}

function sortAsText(components,value){

    components.sort(function(c1,c2){

        c1 = c1.querySelector("p[value='" + value + "']").innerHTML.toLowerCase();

        c2 = c2.querySelector("p[value='" + value + "']").innerHTML.toLowerCase();

        return (c1 > c2) - (c1 < c2)

    });

    return components;

}

function sortComponents(){

    let sortFlag = document.getElementById("sortBy").value;

    let orderFlag = document.getElementById("order").value;

    let container = document.getElementById("componentsContainer");

    let containerCopy = container.cloneNode(true);

    let components = containerCopy.getElementsByClassName("component");

    components = Object.values(components).filter(e=>!e.classList.value.includes("hiden"));

    if(sortFlag === "brand"|sortFlag === "model"|sortFlag === "cpu")

        components = sortAsText(components,sortFlag);

    else if(sortFlag === "diagonal"|sortFlag === "ram"|sortFlag === "rom")

        components = sortAsNumber(components,sortFlag);

    else return;

    if(orderFlag === "desc") components.reverse();

    components.forEach(function(c, pos){

        c.style.order = pos;

    });

    document.body.replaceChild(containerCopy,container);

}

function extractAndAddInnerText(titles){

    let result = "";

    for(let j = 0;j < titles.length;j++) result+=titles[j].innerHTML;

    return result;

}

function convertWordsIntoRegExp(str){

    str = str.replace(/[^\w\s:]/g,'\\'+'$&');

    words = str.split(" ");

    let regExpStr = words[0];

    for(let i = 1;i < words.length;i++){

        regExpStr += "|" + words[i];

    }

    return new RegExp(regExpStr,"i");

}

function search(){

    let text = document.querySelector("input[type='search']").value;

    let components = document.getElementsByClassName("component");

    let regExp = convertWordsIntoRegExp(text);

    for(let i = 0;i < components.length;i++){

            let titles = components[i].querySelectorAll("p");

            let wholeComponentText = extractAndAddInnerText(titles);

            if(!regExp.test(wholeComponentText)){

                components[i].classList.add("hiden");

                setTimeout(function(){

                    components[i].classList.add("hiden2");

                },600);

            }

            else {

                components[i].classList.remove("hiden2");

                setTimeout(function(){

                    components[i].classList.remove("hiden");

                },600);

            }

    }

}

function processTheSearchRequest(e){

    e.preventDefault();

    search();

    sortComponents();

}

//EVENT HANDLERS

document.forms.search\_form.addEventListener("submit",processTheSearchRequest);

document.getElementById("sortBy").addEventListener("change",sortComponents);

document.getElementById("order").addEventListener("change",sortComponents);