

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМА-ПОМІЧНИК З ПІДБОРУ СУМІСНИХ
КОМП'ЮТЕРНИХ КОМПЛЕКТУЮЧИХ ДЛЯ ЗБАЛАНСОВАНОЇ
ЗБІРКИ ПЕРСОНАЛЬНОГО КОМП'ЮТЕРА
SOFTWARE ASSISTING THE SELECTION OF COMPATIBLE
COMPUTER COMPONENTS FOR A BALANCED ASSEMBLY OF A PC

Виконав(ла): студент(ка) 4 курсу, групи КНТ-229

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

МОМОТ В.І.

(ПРИЗВИЩЕ та ініціали)

Керівник КОЛПАКОВА Т.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОРОТКИЙ О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 122 Комп'ютерні науки
(код і найменування)
 Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
 Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 «_____» _____ 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

МОМОТА Владислава Ігоровича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програма-помічник з підбору сумісних комп'ютерних комплектуючих для збалансованої збірки персонального комп'ютера. Software Assiting the Selection of Compatible Computer Components for a Balanced Assembly of a PC

керівник проєкту (роботи) к.т.н, доцент, КОЛПАКОВА Тетяна Олексіївна,
(науковий ступінь, вчене звання, прізвище, ім'я, по батькові,)

затверджені наказом закладу вищої освіти від «04» квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 1 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування та моделювання вебзастосунку. 3. Розробка вебзастосунку. 4. Керівництво програміста. 5. Керівництво оператора. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	КОЛПАКОВА Т.О., доцент		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання « 04 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання
2	Аналіз предметної області.	2 тиждень	Розділ 1
3	Розробка архітектури програми.	3 тиждень	Розділ 2
4	Розробка програми.	4–5 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми.	6 тиждень	Розділ 4, 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
7	Захист роботи.	8 тиждень	

Студент(ка)

_____ Владислав МОМОТ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Тетяна КОЛПАКОВА
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 137 с., 6 табл., 59 рис., 23 ліст., 2 дод., 28 джерел.

ПРОГРАМА-ПОМІЧНИК, ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР, СУМІСНІСТЬ КОМПЛЕКТУЮЧИХ, ЗБАЛАНСОВАНА ЗБІРКА, ВЕБЗАСТОСУНОК, MONGO DB, REACT, NODE JS, КЛІЄНТ-СЕРВЕР.

Об'єкт дослідження – процес підбору комплектації при збірці ПК з урахуванням сумісності комплектуючих для потреб користувача.

Предмет дослідження – вебзастосунок для аналізу сумісності комплектуючих на основі специфіки їх характеристик.

Мета роботи – розробка зручного та ефективного вебзастосунку для автоматизації процесу підбору сумісних комп'ютерних комплектуючих для скорочення часу, витраченого користувачем під час збірки ПК.

Матеріали, методи та технічні засоби: функціональне програмування, технологія React для створення вебзастосунку, мова програмування JavaScript, крос-платформне середовище виконання JavaScript – Node.js, база даних Mongo DB та операційна система Microsoft Windows 10.

Результати. Створено програмний вебзастосунок, який розділяється на клієнтську та серверну частину. Він допомагає користувачу у підборі сумісних комплектуючих для збалансованої збірки ПК.

Висновки. Розроблено програму-помічник для підвищення ефективності взаємодії розділу комп'ютерних комплектуючих інтернет-магазину “Rozetka” з користувачем або для індивідуального використання для скорочення часу користувача під час збірки ПК.

Галузь використання – комп'ютерне ПЗ для домашнього використання або інтернет-магазини, які реалізують комп'ютерні комплектуючі.

ABSTRACT

Explanatory note to the bachelor's thesis: 137 pages, 6 tables, 59 figures, 23 listings, 2 appendices, 28 sources.

ASSISTANT PROGRAM, PERSONAL COMPUTER, COMPONENT COMPATIBILITY, BALANCED ASSEMBLY, WEB APPLICATION, MONGO DB, REACT, NODE JS, CLIENT-SERVER.

Object - process of selecting a complete set when assembling a PC, taking into account the compatibility of components for the user's needs.

Subject - web application for analyzing the compatibility of components based on the specifics of their characteristics.

The purpose of the study is to develop a convenient and efficient web application to automate the process of selecting compatible computer components to reduce the time spent by the user during PC assembly.

Materials, methods and techniques: functional programming, React technology for creating a web application, JavaScript programming language, cross-platform JavaScript runtime environment – Node.js, Mongo DB database and Microsoft Windows 10 operating system.

Results. A web application has been created that is divided into client and server parts. It helps the user in the selection of compatible components for a balanced PC assembly.

Conclusions. An assistant program has been developed to increase the efficiency of interaction between the computer components section of the Rozetka online store and the user or for individual use to reduce user time during PC assembly.

The field of application is computer software for home use or online stores that sell computer components.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області	11
1.1 Аналіз об'єкту розробки	11
1.2 Аналіз задачі розробки.....	13
1.3 Огляд існуючих аналогів програмного забезпечення	14
1.4 Розробка технічного завдання	18
1.4.1 Підстава для розробки.....	18
1.4.2 Мета розробки та її призначення	18
1.4.3 Основні вимоги до програмного продукту	18
1.4.4 Вимоги до технічних засобів і мови програмування.....	19
1.4.5 Вимоги до програмної документації	19
1.4.6 Порядок приймання і тестування	19
2 Проєктування та моделювання вебзастосунку	20
2.1 Вибір інструментів програмування	20
2.1.1 Аналіз мов програмування.....	20
2.1.2 Аналіз IDE для розробки.....	22
2.1.3 Аналіз баз даних	25
2.1.4 Аналіз інструменту для написання клієнтської частини.....	26
2.1.5 Аналіз інструменту для написання серверної частини	28
2.2 Розробка структурної схеми програми.....	29
2.3 Розробка функціональної схеми	30
2.4 Структура бази даних системи	32
2.5 Вимоги до вебзастосунку.....	33
3 Розробка вебзастосунку	35
3.1 Створення проєкту	35
3.2 Розробка клієнтської частини	36
3.3 Розробка серверної частини.....	51

3.3.1 Опис вхідних та вихідних даних.....	51
3.3.2 Розробка серверу на Node.js.....	57
4 Керівництво програміста	62
4.1 Призначення та умови застосування	62
4.2 Характеристика програми.....	62
4.3 Звертання до програми.....	63
4.4 Вхідні та вихідні дані	63
4.5 Повідомлення для програміста	64
5 Керівництво оператора	66
5.1 Призначення програми.....	66
5.2 Умови виконання.....	66
5.3 Виконання програми	67
5.4 Повідомлення оператору.....	76
Висновки.....	80
Перелік джерел посилань.....	82
Додаток А Текст програми	85
Додаток Б Слайди презентації.....	131

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
ID	– Identification;
IDE	– Integrated Development Environment,
JSON	– JavaScript Object Notation;
NPM	– Node Package Manager;
NPX	– Node Package eXecute;
NVM	– Node Version Manager;
SSD	– Solid-state drive;
SQL	– Structured Query Language;
TDP	– Thermal design power;
URL	– Uniform Resource Locators;
ПЗ	– програмне забезпечення;
ПК	– персональний комп'ютер.

ВСТУП

Дипломна кваліфікаційна робота бакалавра присвячена розробці програми-помічника для автоматичного процесу підбору сумісних комп'ютерних комплектуючих для збалансованої збірки ПК (персонального комп'ютера).

Сучасне життя створює необхідність вивчати світ комп'ютерної техніки, а тому виникає потреба допомогти кожному користувачу у вирішенні задачі збірки ПК.

Вже багато часу людям доводиться вирішувати задачу вибору комп'ютерної техніки як для офісу так і для власного використання. Персональний комп'ютер є майже в кожного, але не кожному легко розібратися у багатоваріантності комплектуючих. Для цього необхідно вивчати переваги комплектуючих, стежити за технічним прогресом, чітко уявляти собі коло розв'язуваних завдань. В теперішній час кожна людина стикається з певними інтернет-ресурсами. Вебсистеми використовуються для вирішення багатьох завдань. Для зручності було розроблено дану програму, яка має допомогти кожному користувачу без витрачання часу на аналіз та особливих знань характеристик комп'ютерних комплектуючих зробити збалансовану збірку свого домашнього персонального комп'ютеру. Така програма повинна робити автоматичний аналіз сумісності комплектуючих, які обирає користувач під час роботи з програмою.

Метою роботи є скорочення часу, витраченого користувачем під час збірки ПК, шляхом розроблення автоматичного процесу підбору сумісних комп'ютерних комплектуючих.

Для досягнення поставленої мети були визначені такі задачі:

- дослідити принципи створення веборієнтованих систем;
- дослідити види комп'ютерних комплектуючих;

- дослідити API (Application Programming Interface) інтернет-магазину “Rozetka” та систематизувати отримані дані щодо наявності та популярності;

- спроектувати базу даних та архітектуру програми;
- визначити особливості характеристик комплектуючих;
- обрати засіб аналізу сумісності комплектуючих;
- програмно реалізувати серверну частину програми;
- програмно реалізувати зовнішній інтерфейс програми.

Розроблено програмну систему для підвищення ефективності взаємодії розділу комп’ютерних комплектуючих інтернет-магазину “Rozetka” з користувачем або для індивідуального використання для скорочення часу користувача під час збірки ПК.

Мова гіпертекстової розмітки HTML, каскадна таблиця стилів SASS, мова програмування JavaScript, фреймворк React, крос-платформне середовище виконання JavaScript – Node.js, фреймворк вебдодатків для Node.js – Express, база даних Mongo DB.

Галуззю використання є комп’ютерне ПЗ (програмне забезпечення) для домашнього використання або інтернет-магазини, які реалізують комп’ютерні комплектуючі.

Результатом роботи є вебзастосунок для підбору конфігурації ПК, з реалізацією аналізу сумісності комплектуючих.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз об'єкту розробки

Надійна робота комп'ютерної техніки, її швидкодія та обчислюванні можливості залежать від конфігурації, правильного підбору, налаштування та узгодження компонентів комп'ютера.

На сучасному ринку зараз пропонується багато різноманітних компонентів для комп'ютера, тому легко помилитись, обравши вже не актуальний або не підходящий. Є багато людей, які не працюють в галузі комп'ютерної техніки, не слідкують за технічним процесом, іноді, навіть, не знають архітектуру ПК, а отже вони не зможуть вирішити, які саме елементи треба обирати при збірці ПК. Дану задачу вирішують конфігуратори. Конфігуратор дає можливість користувачеві без технічного досвіду та навичок замовити деталі та складання ПК: для дому, офісу, ігор, роботи з графікою.

Окрім того, будь-який користувач може заощадити, якщо замість покупки готового ПК від виробника збере комп'ютер частинами. Важливо розуміти, що для формування єдиної системи всі елементи мають бути сумісні. Пересічному користувачеві досить складно самостійно підібрати відповідний набір комплектуючих. Тому не помилитися з вибором допоможе конфігуратор ПК – інструмент, що дозволяє зібрати комп'ютер онлайн. Сервіс дозволяє підібрати оптимальний набір комплектуючих із потрібними характеристиками.

Отже створення вебресурсу, який призначений саме для того, щоб допомогти як людям, що знайомі з комп'ютерною технікою, так і людям, що практично не мають уявлення про сучасні технології знайти для себе найкращий варіант конфігурації ПК є актуальною задачею. Головним завданням даної роботи є створення ресурсу орієнтованого на інформативність, функціональність, доступність і простоту використання ресурсу та адаптивність відображення на різних пристроях.

Для створення даної програми було досліджено характеристики комп'ютерних комплектуючих.

Умовна схема конфігурації ПК представлена на рисунку 1.1 [1].

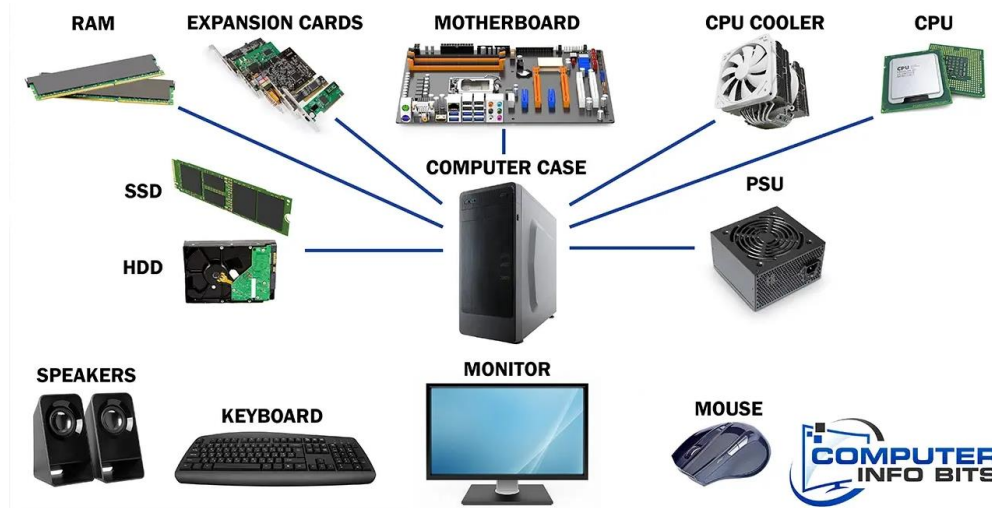


Рисунок 1.1 – Умовна схема конфігурації ПК [1]

Функціональна частина програми реалізована на основі підбору наступних комплектуючих [2]:

- центральний процесор – один з ключових компонентів комп'ютера, який відповідає за інтерпретацію команд та керування різними частинами системи;
- материнська плата – плата, на якій містяться основні компоненти комп'ютера, що забезпечують логіку його роботи;
- відеокарта – ключовий компонент комп'ютера, призначений для генерації та обробки зображень з подальшим їхнім виведенням на екран монітору;
- блок живлення – компонент комп'ютера, призначений для забезпечення вузлів комп'ютера електричною енергією;
- оперативна пам'ять – тип комп'ютерної пам'яті, яка призначена для зберігання даних програм під час їхнього виконання;
- жорсткий диск – комп'ютерний запам'ятовувальний пристрій, магнітний диск, основа якого виконана з жорсткого твердого матеріалу;

– твердотілий накопичувач – комп'ютерний запам'ятовувальний пристрій на основі мікросхем пам'яті та контролера керування ними, що не містить рухомих механічних частин.

1.2 Аналіз задачі розробки

Для вирішення поставленої задачі необхідно:

- провести пошук комп'ютерних комплектуючих, які відповідають сучасним вимогам;
- перевірити їх наявність в інтернет-магазині;
- відфільтрувати товари за популярністю з можливістю вибору комплектуючих різних виробників;
- вивчити характеристики комп'ютерних комплектуючих;
- проаналізувати сумісність для досягнення збалансованості збірки.

На першому етапі роботи був проведений пошук усіх комп'ютерних комплектуючих:

- процесор;
- кулер процесора;
- материнська плата;
- оперативна пам'ять;
- відеокарта;
- блок живлення;
- жорсткий диск;
- SSD (Solid-state drive) диск;
- корпус.

Під час реалізації даної програми була досліджена сумісність наступних комплектуючих: процесора та материнської плати, процесора та кулера, процесора та оперативної пам'яті, відеокарти та процесора, корпусу

та відеокарти, корпусу та кулера. Також досліджена сумісність блока живлення з загальним споживанням системи.

Для перевірки сумісності були використанні такі параметри комплектуючих: сокет процесора, чипсет материнської плати, тип оперативної пам'яті, TDP (Thermal design power) кулера, назва відеокарти, ширина відеокарти, висота кулера, потужність блока живлення.

1.3 Огляд існуючих аналогів програмного забезпечення

Конфігуратор ПК від CAN.ua – конфігуратор, який допомагає зібрати комп'ютер онлайн з перевіркою фізичної сумісності комплектуючих.

В нього вбудовані функції перевірки сумісності обраних компонентів, які інформують про помилки або проблемних місцях обраної збірки, дають підказки щодо їх усунення перед ухваленням рішення про покупку або збереженні конфігурації, при цьому носять виключно рекомендаційний характер і не обмежують у процесі складання майбутнього ПК [3].

Переваги:

- можна обрати не тільки комплектуючі, а ще є багатий вибір з комп'ютерної периферії (монітор, мишка, клавіатура, акустика, джерело безперебійного живлення, навушники);
- зручний та простий інтерфейс на українській мові;
- можливість зберегти збірку ПК та надіслати на електронну пошту;
- підказки щодо сумісності та інших аспектів під час збірки ПК;
- є можливість порівняти товари між собою.

Недоліки:

- немає підказок для створення саме збалансованої збірки;
- невелика кількість товарів для вибору (товари тільки цього магазину);
- немає перевірки сумісності для корпусів та кулерів;

– немає перевірки сумісності для блоків живлення (є тільки підказки, які можна не побачити) .

Сторінку конфігуратора CAN.ua наведено на рис. 1.2 [3].

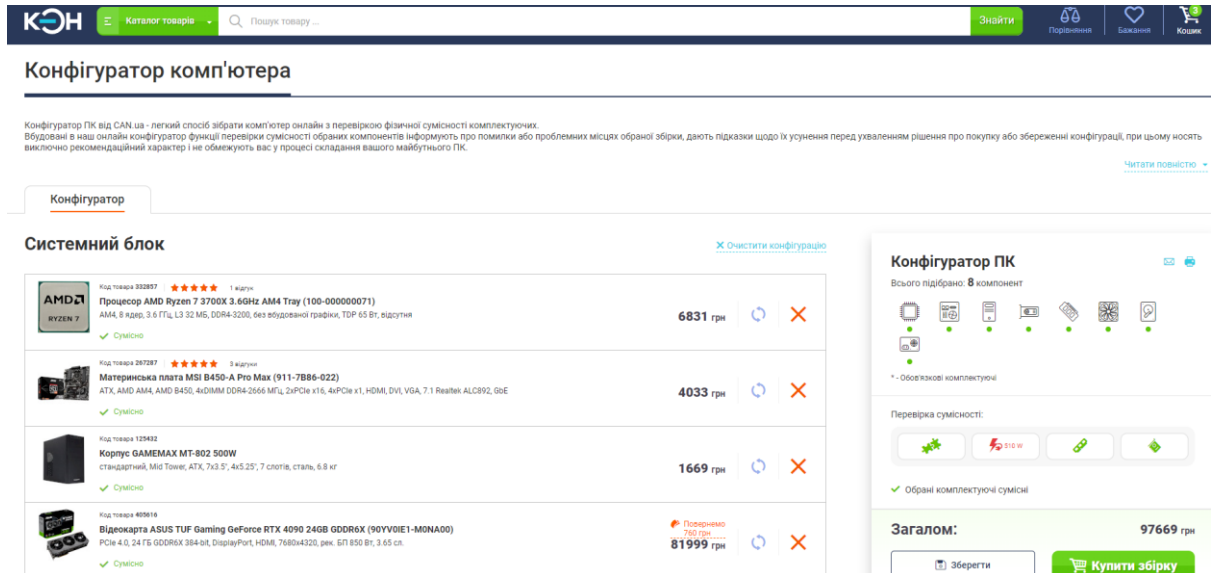


Рисунок 1.2 – Сторінка конфігуратора CAN.ua [3]

PCPartPicker – цей сайт дозволяє вибрати окремі компоненти для вашої збірки і гарантує, що вони сумісні один з одним. Він також надає відгуки користувачів і рейтинги для кожного компонента [4].

Переваги:

- широкий вибір комплектуючих у багатьох магазинах;
- детальний опис з графіком зміни цін та відгуками;
- детальні підказки та опис проблем сумісності;
- сумісність перевіряється за багатьма показниками;
- можливість зберегти збірку;
- свій форум для обговорення сумісності.

Недоліки:

- тільки американський ринок, та інтерфейс на англійській мові;
- перевантажений інтерфейс, дуже багато інформації на одній сторінці;

– опис деяких проблем не буде зрозумілий користувачу, який має недостатньо досвіду в цій області;

– немає підказок для створення саме збалансованої збірки.

Сторінку конфігуратора PCPartPicker наведено на рис. 1.3 [4].

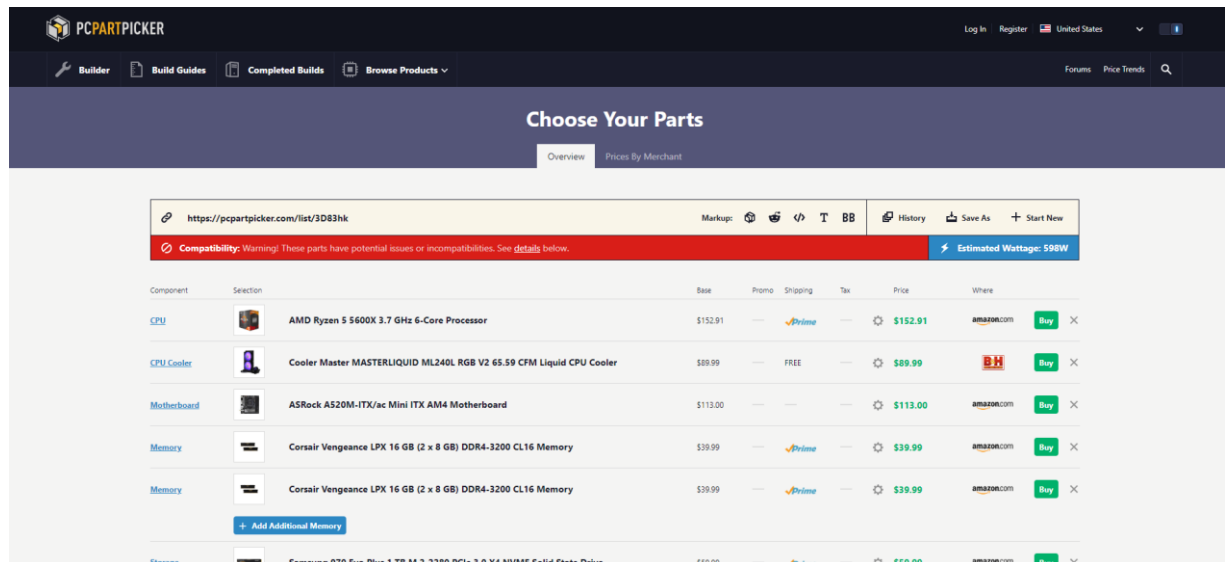


Рисунок 1.3 – Сторінка конфігуратора PCPartPicker [4]

Newegg – популярний сайт для купівлі комп'ютерних компонентів, який також пропонує інструмент для складання ПК, що дозволяє вибрати компоненти для вашої збірки [5].

Переваги:

– багатофункціональний, є можливість не тільки збірки ПК, а ще обрати готові збірки, периферію та інші комп'ютерні девайси;

– при виборі комплектуючих, вони вже відфільтровані за сумісністю;

– можливість надіслати збірку на електронну пошту.

Недоліки:

– тільки іноземний ринок, та інтерфейс на англійській мові;;

– товари тільки цього магазину;

– для замовлення потрібна реєстрація;

– перевантажений інтерфейс, багато непотрібної, саме для збірки, інформації на сайті;

– немає підказок для створення саме збалансованої збірки та підказок щодо сумісності.

Сторінку конфігуратора Newegg наведено на рис. 1.4 [5].

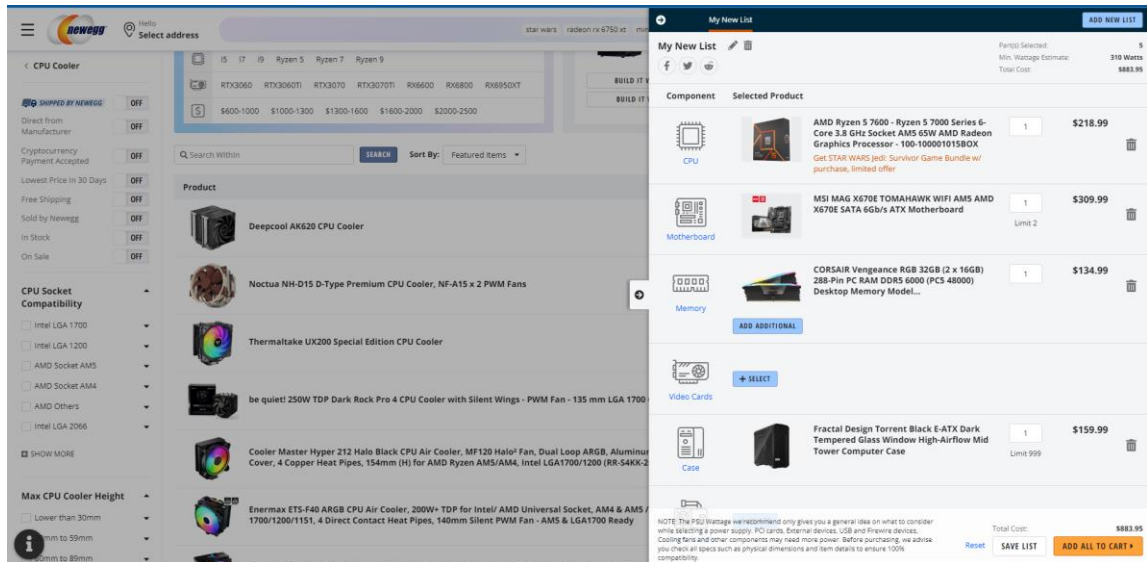


Рисунок 1.4 – Сторінка конфігуратора Newegg [5]

В результаті огляду аналогів було виявлено, що вони мають наступну функціональність:

- вибір комплектуючих зі списку товарів;
- перевірка комплектуючих на сумісність;
- можливість додати вибрані комплектуючі до корзини та відправити збірку на електронну пошту;
- зрозумілий та простий інтерфейс програми.

Серед переважаючих недоліків у застосунках-аналогах спостерігаються відсутність підказок щодо збалансованості збірки та орієнтованість тільки на свої товари або товари на українського ринку.

Беручи до уваги усі перераховані вище переваги та недоліки існуючого програмного забезпечення, було створено технічне завдання на застосунок, що розробляється.

1.4 Розробка технічного завдання

1.4.1 Підстава для розробки

Підставою для розробки служить завдання на дипломну кваліфікаційну роботу бакалавра на тему "Програма-помічник з підбору сумісних комп'ютерних комплектуючих для збалансованої збірки персонального комп'ютера", затверджене наказом № 87 від 4 квітня 2023 р. по НУ «Запорізька політехніка».

1.4.2 Мета розробки та її призначення

Метою є розробка програмного забезпечення для автоматичного процесу підбору сумісних комп'ютерних комплектуючих, що призначене для скорочення часу користувача під час збірки ПК.

1.4.3 Основні вимоги до програмного продукту

Вебзастосунок, що розробляється, має мати наступну функціональність:

- можливість вибрати комплектуючі зі списку;
- фільтрування списків, з функцією виведення тільки сумісних комплектуючих;
- виведення інформації о товарах інтернет-магазину “Rozetka”;
- повідомлення користувачеві про відсутність сумісності відповідних комплектуючих;
- алгоритм перевірки збалансованості збірки (процесора та відеокарти);
- збереження готової збірки у програмі;
- можливість відправити готову збірку на електронну пошту.

Вхідними даними є обрані комп'ютерні комплектуючі.

Вихідними даними є готова збірка ПК з урахуванням сумісності обраних комп'ютерних комплектуючих.

1.4.4 Вимоги до технічних засобів і мови програмування

Програма, що розробляється у вигляді вебзастосунку має бути написана на мові JavaScript, з використанням фреймворку React у середовищі розробки Visual Studio Code. Програма має бути кросплатформною та виконуватися у будь-якій системі, яка підтримує сучасні браузері.

1.4.5 Вимоги до програмної документації

Програмна документація має включати: керівництво програміста, керівництво оператора, опис програми та тексти основних програмних модулів.

1.4.6 Порядок приймання і тестування

Тестування даного ПЗ проводиться у вигляді виконання усіх можливих сценаріїв роботи оператора. Дане ПЗ вважається прийнятним, якщо воно задовольняє усім висунутим вимогам до нього і успішно пройшло етап тестування.

2 ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ

2.1 Вибір інструментів програмування

Для розробки вебзастосунку треба використати такі типи технологій та засобів:

- веборієнтовану мову програмування;
- IDE (Integrated Development Environment) для розробки;
- базу даних;
- фреймворк для клієнтської частини;
- інструмент для серверної частини.

Вибір технологій розробки є суттєвим етапом у розробці будь-якої системи і може суттєво впливати на швидкість, якість та майбутні можливості проєкту. Особливо важливо правильно підібрати технології для веборієнтованих систем, оскільки вони залежать від вимог проєкту та потреб користувачів.

2.1.1 Аналіз мов програмування

JavaScript – динамічна, об'єктно-орієнтована, прототипна, мова програмування, найчастіше використовується для написання вебсторінок [6].

Переваги використання JavaScript:

- висока швидкість;
- розширена функціональність;
- інтерактивність на стороні клієнта;
- легка інтеграція з API;
- універсальність.

Недоліки використання JavaScript:

- код завжди видно;
- несумісність з деякими старими браузерами.

RНР – скриптова мова програмування, була створена для генерації HTML сторінок на стороні вебсервера [7].

Переваги використання RНР:

- не потрібно писати додатковий код;
- ефективно працює з базами даних;
- має високий рівень безпеки.

Недоліки використання RНР:

- не підходить для великих вебдодатків на основі контенту;
- використання фреймворку та інструментів RНР призводить до погіршення продуктивності вебдодатку;
- не дозволяє змінювати основну поведінку вебдодатків.

Python – інтерпретована об'єктно-орієнтована мова програмування зі строгою динамічною типізацією, яка може бути використана для написання серверної частини вебдодатку [8].

Переваги використання Python:

- легко писати, читати та вивчати;
- інтерпретована мова;
- динамічно-типізована мова;
- відкритий вихідний код;
- підтримує велику кількість бібліотек;
- переносимість.

Недоліки використання Python:

- низька швидкість роботи;
- помилки під час виконання;
- велике споживання пам'яті;
- обмежений доступ до даних.

Оскільки розробляється динамічний клієнт-серверний вебдодаток, для цієї задачі краще за все підходить JavaScript, бо це універсальна мова

програмування, яку можна використовувати як для фронтенд-, так і для бекенд-розробки.

Порівняння JavaScript з іншими мовами програмування наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння мов програмування

Критерій	Python	PHP	JavaScript
Функціональна	так	ні	так
Спектр застосувань	AI, web, mobile apps, big data	web, server side	web, frontend, backend
Виконує програми на стороні клієнта	ні	ні	так
Виконується в браузері	ні	ні	так
Швидкість	низька	висока	висока
Затребуваність	2 місце	7 місце	1 місце
Підтримка спільноти	Має велику підтримку спільноти	Має велику підтримку спільноти	Має велику підтримку, існує безліч фреймворків та бібліотек

2.1.2 Аналіз IDE для розробки

IDE – це інтегроване середовище для розробки програмного забезпечення. Воно зазвичай містить компілятор, дебаггер, редактор коду, засоби автоматизації тестування та інші інструменти, які допомагають програмістам вести розробку швидше та ефективніше [9].

Перевага IDE полягає в тому, що вони можуть допомагати відслідковувати помилки в коді, забезпечувати автодоповнення та відживлення коду під час виконання програми. Також IDE можуть надавати доступ до бібліотек та інших ресурсів, що сприяє швидкому та ефективному розвитку програмного забезпечення.

Для розробки вебзастосунку на основі JavaScript можна використовувати багато IDE або текстових редакторів, таких як:

- Visual Studio Code;
- WebStorm;
- Sublime Text;
- NetBeans;
- Notepad++;
- Eclipse;
- Atom.

Найпопулярнішими з вище описаних є WebStorm та Visual Studio Code.

WebStorm є інтегрованим середовищем розробки (IDE), призначеним для розробки вебдодатків, особливо з фокусом на JavaScript, HTML та CSS. Він розроблений компанією JetBrains, яка також створила платформу IntelliJ IDEA, на базі якої побудований WebStorm. Основні функції та переваги WebStorm включають: автодоповнення, аналіз коду, навігація по коду, рефакторинг, зневадження [10].

Visual Studio Code є безкоштовним текстовим редактором та інтегрованою середовищем розробки (IDE), яке надає зручний і продуктивний інтерфейс для створення, редагування та зневаження сучасних вебзастосунків та програм для хмарних систем [11].

VS Code доступний для платформ Windows, Linux і macOS, що робить його популярним серед розробників незалежно від їх операційної системи. Він пропонує широкий набір функціональності, яка сприяє зручному та продуктивному процесу розробки вебдодатків [11].

За допомогою VS Code ви можете працювати з різними мовами програмування, такими як HTML, CSS, JavaScript, TypeScript та інші. Він підтримує підсвічування синтаксису, автодоповнення коду, вбудовані термінали, та багато іншого. Крім того, завдяки його розширюваній архітектурі, ви можете встановлювати додатки для розширення його функціональності відповідно до своїх потреб [11].

Обидва інструменти мають можливість працювати з розподіленою системою контролю версій.

Система контролю версій – інструмент, призначений для відстеження змін у файлі або наборі файлів протягом часу. Вона дозволяє зберігати і керувати версіями документів, програмного коду або будь-якого іншого виду файлів [12].

Git – одна з таких систем для управління та зберігання проєкту. Git є розподіленою системою контролю версій, яка полегшує зберігання, відстеження та управління проєктом у команді [12].

Хоча ці редактори коду є дуже схожими за своїм функціоналом, виходячи з порівняльної таблиці 2.2 для вебдодатку краще використовувати Visual Studio Code через те, що ця IDE є більш зрозумілою та простою для створення невеликих додатків, що прискорює розробку. Також вона є безкоштовною.

Таблиця 2.2 – Порівняння IDE для розробки

Критерій	Visual Studio Code	WebStorm
Підтримка Git	Так	Так
Підтримка мови JavaScript	Так	Так
Більш висока швидкість	Так	Ні
Можливість розширення	Так	Так
Безкоштовна?	Так	Ні

2.1.3 Аналіз баз даних

Кожен додаток який працює з даними повинен десь їх зберігати. Для таких цілей створюється база даних, в якій зберігаються статичні дані, які можна створювати, редагувати та видаляти з неї. Дані зберігаються на зовнішніх носіях інформації або на онлайн серверах бази даних.

Для нашої задачі було порівняно такі бази даних: MySQL та Mongo DB.

MySQL – це відкрита реляційна система управління базами даних, розроблена та підтримувана компанією Oracle. Вона надає можливості для зберігання та керування структурованими даними за допомогою SQL (Structured Query Language) [13].

MongoDB – це нереляційна база даних на основі документів, також відома як об'єктна система, що дає змогу розробникам програмного забезпечення масштабувати свої програмні додатки, спрощуючи їхню розробку [14].

Для зручної роботи бази даних Mongo DB у середовищі Node JS використовується бібліотека Mongoose.

Mongoose – це бібліотека для роботи з MongoDB у середовищі Node.js. Вона надає зручні інструменти для моделювання схем даних, виконання запитів та взаємодії з базою даних MongoDB [15].

Mongoose дозволяє моделювати дані у вигляді об'єктів JavaScript, які відповідають документам MongoDB. Це дозволяє зручно працювати з даними, використовуючи звичні об'єктно-орієнтовані підходи [15].

Під час розгляду переваг та недоліків баз даних для вебдодатку на основі JavaScript, порівняння яких відображено в таблиці 2.3 було вирішено, що MongoDB є найбільш підходящим вибором завдяки гнучкості, масштабованості та простоті використання.

Таблиця 2.3 – Порівняння баз даних

Критерій	Mongo DB	MySQL
Схема	MongoDB зберігає дані в колекціях без обов'язкової схеми.	MySQL вимагає визначення схеми для таблиць у базах даних.
Структура даних	Дані зберігаються та подаються у форматі JSON (JavaScript Object Notation) (не реляційна)	Кожен запис зберігається у вигляді структури таблиці з рядками і стовпчиками (реляційна)
Маштабованість	По вертикалі та горизонталі	Тільки по вертикалі
Гібкість	Висока	Середня
Обробка неструктурованих даних	Присутня	Відсутня
Безпека	Менш уразлива до кібератак	Більш уразлива до кібератак
Підтримка спільноти	Середня	Висока

2.1.4 Аналіз інструменту для написання клієнтської частини

Для створення вебзастосунків зараз часто використовуються різні фреймворки та бібліотеки, які пришвидшують розробку та розширюють функціонал мови.

Для написання клієнтської частини на мові програмування JavaScript найпопулярнішими є такі бібліотеки:

- Angular;
- React;
- Vue.

React – це відкрита бібліотека JavaScript, розроблена компанією Facebook. Вона з'явилася у 2013 році і швидко набула популярності у розробці сучасних вебдодатків, зокрема односторінкових додатків [16].

React дає змогу поєднати html-розмітку та JavaScript код за допомогою JSX-розмітки, яку React підтримує з коробки.

JSX – це комбінація розмітки UI та програмного коду, яку можна написати прямо всередині JavaScript файлів. Це дозволяє розробникам описувати як структуру, так і поведінку компонентів в одному місці, що сприяє прискоренню розробки [17].

Основний принцип React – створення компонентів. У React компонент є основною будівельною одиницею додатку. Компонент отримує вхідні дані (властивості) і повертає розмітку (відображення), яку браузер рендерить. Компоненти треба використовувати повторно, щоб не дублювати код [16].

Angular – це повноцінний фреймворк для створення клієнтських додатків, розроблений компанією Google. Він надає інструменти та структуру для побудови масштабних та динамічних вебдодатків, які працюють в браузері [18].

Vue – це прогресивний JavaScript-фреймворк з відкритим вихідним кодом для побудови користувацьких інтерфейсів. Він став популярним завдяки своїй простоті вивчення, гнучкості та ефективності [19].

З порівняльної таблиці 2.4 видно, що найбільшими конкурентами є Angular та React, тому що вони вже давно на ринку та мають велику підтримку суспільства, просту в освоєнні документацію та популярність. Vue в цьому плані відстає тому, що з'явився на ринку нещодавно і поки не має великої підтримки. Порівнюючи Angular та React, для розроблюваного вебдодатку, який є не дуже складним у розробці більш підходящим є React, через легкість освоєння та тому, що код React пишеться з використанням мови JavaScript, а Angular – TypeScript.

Таким чином ми проаналізували найпопулярніші бібліотеки та фреймворки для створення клієнтської частини вебдодатків та обрали бібліотеку React.

Таблиця 2.4 – Порівняння бібліотек та фреймворків React, Angular, Vue

Критерій	React	Angular	Vue
Рік випуску	2013	2010	2014
Мова	JavaScript	TypeScript	JavaScript
Підтримка суспільства	Висока	Висока	Низька
Легкість освоєння	Висока	Низька	Висока
Гнучкість	Середня	Низька	Висока

2.1.5 Аналіз інструменту для написання серверної частини

Для створення серверної частини вебзастосунку буде використана мова Javascript, вона була обрана завдяки її універсальності, але універсальною вона стає за допомогою Node.js.

Node.js – це середовище виконання JavaScript, побудоване на движку V8, що дозволяє виконувати JavaScript код поза браузером, на сервері. Основне призначення Node.js полягає у створенні швидких та масштабованих мережових додатків [20].

Раніше на серверах використовувалася традиційна схема, де клієнт відправляв запит до сервера, сервер обробляв запит і відправляв відповідь, після чого з'єднання припинялося. Це означало, що сервер повинен був завершити обробку попереднього запиту, перш ніж приступити до наступного. Це призводило до проблеми, де сервер не міг обробляти багато запитів одночасно, що обмежувало продуктивність та швидкодію системи.

Node.js використовує подієво-орієнтовану модель та неблокуючу архітектуру вводу/виводу, що дозволяє обробляти багато запитів одночасно,

не блокуючи основний потік операціями вводу-виводу. Це робить його ефективним та швидким в роботі з великими навантаженнями [20].

Node.js є фаворитом для розробки серверної частини вебдодатків. Він використовує вебфреймворк Express, який є дуже популярним і надає гнучкість та широкий спектр можливостей для створення серверів та обробки запитів. За допомогою Node.js також можна створювати додатки для персональних комп'ютерів та навіть побудувати мережі [20].

Завдяки своїй асинхронності та ефективному використанню ресурсів, Node.js дозволяє забезпечити високу продуктивність серверних додатків та оптимальне використання апаратних ресурсів. Він став популярним вибором для розробки сучасних серверних додатків.

2.2 Розробка структурної схеми програми

Структурна схема програми відображає організацію та взаємозв'язок між компонентами програмного забезпечення. Вона відображає ієрархію модулів, підпрограм, функцій та їх зв'язки.

Веборієнтовані системи працюють за архітектурою клієнт-сервер. Ця архітектура використовується для організації взаємодії між вебдодатком та користувачем через мережу Інтернет.

Веборієнтована система, яка розроблюється, буде працювати за такою схемою:

- користувач відкриває сторінку вебзастосунку та обирає необхідні значення комплектуючих зі списку;
- обрана користувачем інформація трансформується в дані, за допомогою яких формується запит, який буде відправлений на сервер;
- сервер отримує запит, оброблює його шляхом пошуку логіки та валідації цих даних;
- сервер виконує запит з параметрами пошуку до бази даних;
- сервер повертає результат клієнтській частині;

- клієнт отримує список комплектуючих необхідного типу, які відповідають обраним значенням;
- клієнт обирає ці комплектуючі, їх ID (Identification) відправляються на сервер;
- сервер отримує запит та генерує запит до стороннього API;
- сервер повертає результат, який повернула API, клієнтській частині;
- клієнтська частина отримує інформацію про товар та за допомогою гіпертекстової розмітки відображає її клієнту.

Структурна схема програми відображена на рисунку 2.1.



Рисунок 2.1 – Структурна схема програми

2.3 Розробка функціональної схеми

Функціональна схема є важливим етапом розробки системи або додатку і допомагає уявити, як компоненти системи взаємодіють між собою та зовнішнім середовищем. Основна мета функціональної схеми полягає в тому, щоб відобразити логіку роботи системи та ідентифікувати взаємодію між її компонентами.

В даному вебзастосунку процес функціонування базується на аналізі сумісності для кожної комплектуючої.

Програма працює за алгоритмом фільтрування комплектуючих виходячи з їх сумісності або збалансованості для відеокарти та процесора. Якщо пошук сумісних комплектуючих не дав результату, то програма виводить про це повідомлення та користувачеві потрібно змінити або тип цієї комплектуючої, або змінити ту комплектуючу з якою є проблеми сумісності, щоб порівнювальні параметри пошуку дали необхідний результат. Якщо всі комплектуючі є сумісними то користувач зможе додати збірку до корзини та відправити на електронну пошту.

Функціональна схема програми відображена на рисунку 2.2.

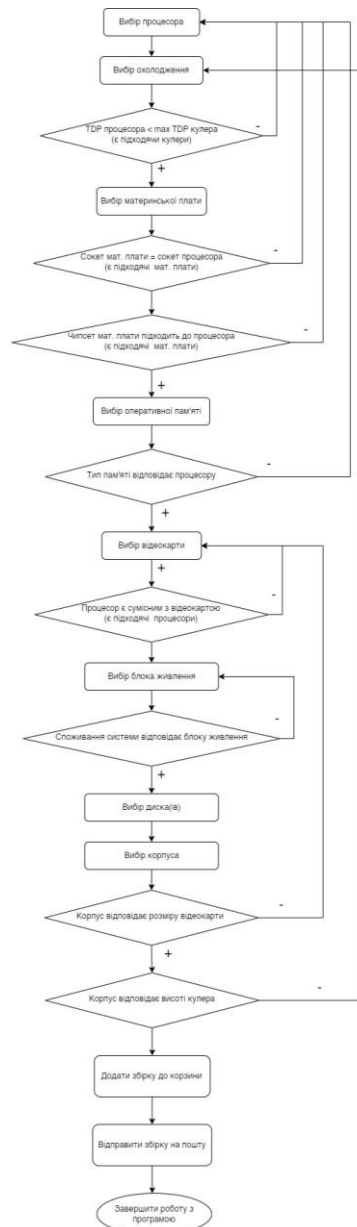


Рисунок 2.2 – Функціональна схема програми

2.4 Структура бази даних системи

Структура бази даних системи визначає, як дані організовані та зберігаються в системі. Вона включає в себе таблиці, схеми, відношення між таблицями та інші елементи, що визначають структуру даних.

Структура бази даних визначається під час проєктування системи та залежить від конкретних потреб додатку. Вона повинна бути гнучкою, ефективною та добре організованою для забезпечення оптимальної роботи системи та зручного доступу до даних.

MongoDB є документ-орієнтованою базою даних, тому її структура відрізняється від традиційних реляційних баз даних. У MongoDB дані зберігаються у вигляді документів у форматі JSON. Документи в MongoDB не вимагають фіксованої структури і можуть мати різні поля та значення [14].

Основними складовими структури бази даних MongoDB є колекції та документи.

Колекції: вони аналогічні таблицям в реляційних базах даних і містять набір документів, які групуються за певним типом або категорією даних. Кожна колекція може мати унікальне ім'я і може містити будь-яку кількість документів.

Документи: вони представляють собою основні одиниці даних в MongoDB. Кожен документ є окремим записом, який містить поля зі значеннями. Документи можуть бути різної структури, і вони можуть мати вкладені документи або масиви значень.

Колекціями у створюваному вебзастосунку є комп'ютерні комплектуючі, а документами товари з інтернет-магазину з їх характеристиками та ID товару.

Виходячи з цього була розроблена схема структури бази даних, яка відображена на рисунку 2.3.

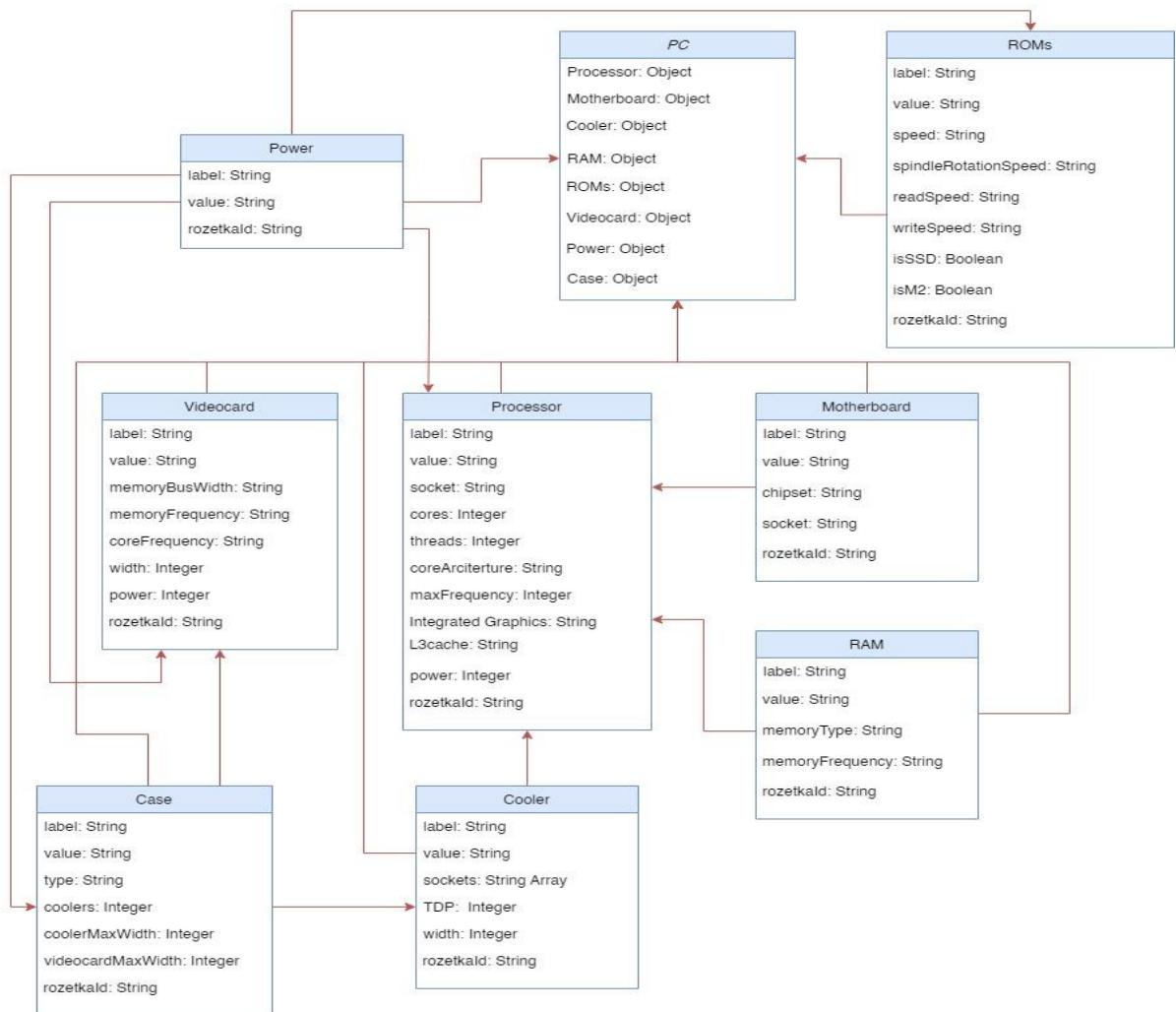


Рисунок 2.3 – Структурна схема бази даних

2.5 Вимоги до вебзастосунку

Система повинна відповідати усім сучасним вимогам розробки вебзастосунків та усім вимогам обраної тематики системи.

Вимоги до основної сторінки:

- можливість вибрати комп'ютерні комплектуючі;
- наявність повідомлень щодо сумісності;
- докладна інформація про вибрані комплектуючі;
- валідація форми збірки ПК.

Вимоги до сторінки кошика:

- збереження збірки у системі;

- можливість відправити збірку на електронну пошту;
- можливість зберегти електронну пошту клієнта в системі.

Вимоги до дизайну:

- мінімалістичний, сучасний дизайн;
- інформативність інтерфейсу;
- інтуїтивність інтерфейсу;
- адаптивність відображення на різних пристроях.

Вимоги до серверної частини:

- можливість додавати нові комплектуючі до системи;
- можливість оновлювати ID товару інтернет-магазину, якщо він перестане бути в наявності;
- можливість видаляти застарілі комплектуючі з системи.

Вимоги оптимізації:

- кроссплатформність;
- підтримання великої кількості одночасних запитів;
- повинна мати високу швидкість запитів.

3 РОЗРОБКА ВЕБЗАСТОСУНКУ

3.1 Створення проєкту

Після аналізу предметної області, створення технічного завдання та визначенням технічних засобів, які будуть використані можна починати розробку програми.

Для успішної роботи з React та Node.js необхідно встановити платформу Node.js. Для цього можна перейти на офіційний вебсайт Node.js [21] і завантажити останню стабільну версію цього програмного забезпечення, або скористатися утилітою NVM (Node Version Manager), яка дозволяє встановлювати та керувати різними версіями Node.js.

Після встановлення Node.js, ви отримуєте в комплекті з ним два важливих інструменти – NPM (Node Package Manager) та NPX (Node Package eXecute).

NPM – це інструмент для управління пакетами Node.js. Він дозволяє встановлювати, оновлювати та видаляти пакети з реєстру NPM. NPM є найбільшим реєстром пакетів програмного забезпечення в світі і надає доступ до безлічі корисних пакетів, які можна використовувати у ваших проєктах [22].

NPX – це новий інструмент, який постачається разом з Node.js. Він дозволяє виконувати команди та встановлювати пакети програмного забезпечення. NPX дозволяє запускати пакети безпосередньо з реєстру NPM, без необхідності встановлювати їх локально на вашому комп'ютері [22].

Використання NPM та NPX дозволяє легко керувати пакетами та виконувати різні завдання в процесі розробки вашого проєкту з використанням React та Node.js.

Клієнтська частина починається зі створення проєкту React у командному рядку. Для цього є спеціальна команда create-react-app, текст якої наведено у лістингу 3.1, з її допомогою можна швидко та легко створити новий проєкт.

Лістинг 3.1 – Команда створення нового проєкту

```
npx create-react-app project_name
```

Після створення проєкту React буде створена початкова структура, яка включає дві папки: “public” і “src”. У папці “public” знаходиться головний файл “index.html”, який відкривається в браузері та є точкою входу до додатку. У папці “src” розміщуються файли, необхідні для розробки додатку.

У файлі “index.js” в папці “src” відбувається підключення React до “index.html”. Файл “App.js” є головним компонентом React, який відображається у нашому додатку і з якого починається розробка. У React додатку компоненти використовуються для побудови інтерфейсу. Усі файли, необхідні для роботи додатку, рекомендується створювати в папці “src”.

3.2 Розробка клієнтської частини

React створює односторінкові додатки, тому у проєкті React зазвичай присутній лише один файл “index.html”, який є початковим шаблоном сторінки. Цей файл містить мінімальну розмітку HTML і підключає JavaScript код React додатку за допомогою елемента “root” , у ньому вказуються зазвичай назва додатку, лого та метадані необхідні для пошукових систем.

Для розробленого застосунку структура складається з таких компонентів:

- Assets – зображення та шрифти;
- Components – загальні компоненти програми, які можуть бути повторно використані;
- Globals – тут зберігаються файли дій, за допомогою яких програма з’єднується з серверною частиною за допомогою запитів та зберігає дані відповідей з сервера до сховища програми;
- Pages – сторінки програми (головна сторінка конфігуратора та сторінка корзини);

- Stores – функціонал сховища, за допомогою якого зберігаються дані відповідей з сервера;
- Theme – основні стилі програми.

Структура вебзастосунку представлена на рисунку 3.1.

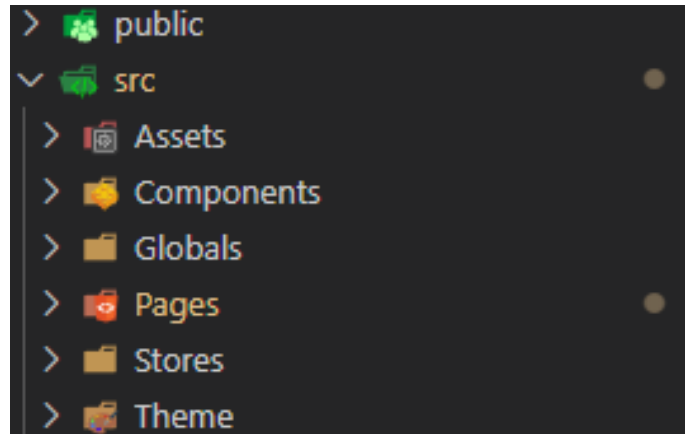


Рисунок 3.1 – Структура клієнської частини програми

Для того, щоб в React додатку можна було створювати декілька сторінок, використовується бібліотека react-router-dom.

Розробка починається з основного файлу програми App.js, в якому розміщується оболонка сховища програми, компонент меню та навантажувача, роутер з навігацією по сторінкам.

Для створення веб застосунку була використана функціональна парадигма програмування.

Функціональне програмування – це парадигма програмування, в якій програми структуруються та будуються навколо функцій. Основна ідея полягає в тому, що функції розглядаються як основний будівельний блок програм, і вони повинні бути безсторонніми, незмінними та повторно використовуваними [23].

Функціональне програмування в контексті React відноситься до підходу, де в основі структури додатка лежать функціональні компоненти. У традиційному підході до розробки React додатків використовуються класові

компоненти, але починаючи з React 16.8, була введена можливість використання функціональних компонентів за допомогою хуків (hooks).

У функціональному підході до React, компоненти визначаються як чисті JavaScript функції, які приймають вхідні дані (props) і повертають відображення на основі цих даних. Функціональні компоненти мають більш просту структуру і вони зазвичай є більш декларативними, що полегшує розробку і робить код більш зрозумілим.

Хуки в React – це нововведення, які дозволяють використовувати стан, контекст та інші можливості React без написання класів, як це робилось раніше. Використання хуків включає в себе використання спеціальних функцій, таких як `useState`, `useEffect`, `useContext` та багатьох інших [24].

В React для маршрутизації та відображення різних сторінок в залежності від шляху на сайті використовується компонент Router разом з компонентом Route. Компонент Route встановлює відповідність між шляхом та компонентом, який буде відображений, коли шлях відповідає поточному URL (Uniform Resource Locators). Шлях може бути заданий у компонента Route за допомогою атрибуту `path`, а компонент, який має бути відображений, передається за допомогою атрибуту `component` або `render`.

Приклад використання React Router відображений у лістингу 3.2.

Лістинг 3.2 – Використання Router у програмі

```
<Router>
  <Routes>
    <Route exact path={'/'} element={<MainPage/>} />
    <Route exact path={'/cart'} element={<CartPage/>} />
  </Routes>
</Router>
```

Для того, щоб застосунок не зависав під час виконання запиту, потрібно було додати компонент завантаження у вигляді колеса. Код цього компоненту наведено у лістингу 3.3.

Лістинг 3.3 – Код власного компоненту завантаження.

```
import React, { useEffect } from 'react';
import { useSelector } from 'react-redux';
import './style.scss';
const CustomLoader = () => {
  const loading = useSelector((state) => state.app.loading);
  useEffect(() => {
    if (loading) {document.body.style.overflow = 'hidden';
    } else {document.body.style.overflow = 'unset';}
  }, [loading]);
  return (loading) && (
    <div className="loaderOverlay">
      <div className="loader">
        <svg className="circular" viewBox="25 25 50
        50">
          <circle className="path" cx="50"
          cy="50" r="20" fill="none"
          strokeWidth="3"
          strokeMiterlimit="10"/>
        </svg>
      </div>
    </div>
  );
}
export default CustomLoader;
```

Вебдодатку потрібно мати можливість швидкої та зручної роботи з серверною частиною та вміти зберігати дані про комплектуючі, які будуть надходити. Для виконання цих задач була використана зв'язка Redux + Saga.

Redux – це бібліотека управління станом, яка дозволяє отримати збережені в ній дані з будь-якої частини програми. Основна ідея Redux полягає в тому, щоб усі дані, які визначають стан додатка, зберігати в одній централізованій структурі даних, яку називають “store”. Компоненти додатка взаємодіють з цим стором, виконуючи дії, які називаються “actions”, та оновлюючи стан через функції, які називаються “reducers” [25].

Redux Saga – це бібліотека для керування побічними ефектами в додатку, зокрема виконання асинхронних запитів, обробки даних з сервера, а також управління потоками даних [26]. Щоб працювати з сагою в додатку потрібно створювати “sagas”. Вони слідкують за діями (actions), які відбуваються в додатку, і відповідають на них виконанням певних операцій. Також вони взаємодіють з серверною частиною за допомогою виклику запитів з компоненту “services” передаючи дані запиту (payload) клієнтської частини та зберігаючи дані відповідей (response) у сховищі програми.

Redux необхідно підключити до проєкту для цього створюється папка store з файлом index.ts, у якому створюється сховище даних. Код створення store наведено у лістингу 3.4.

Лістинг 3.4 – Код створення store за допомогою Redux

```
import { applyMiddleware, createStore, compose } from 'redux';
import { createBrowserHistory } from 'history'
import { persistStore, persistReducer } from 'redux-persist';
import storage from 'redux-persist/lib/storage';
import rootSaga from './sagas';
import rootReducer, { blacklistReducers } from './reducers';
import middleware, { sagaMiddleware } from './middleware';
export const history = createBrowserHistory()
const reducer = persistReducer<RootStateType>(
  {
    key: 'root',
    storage,
    blacklist: Object.keys(blacklistReducers),
  }, rootReducer
);
type PropertyTypes<T> = T extends {[key: string]: infer U} ? U
: never
export type InferActionTypes<T extends {[key: string]: (...args: any[])=> any}> = ReturnType<PropertyTypes<T>>
declare global {
  interface Window {
    __REDUX_DEVTOOLS_EXTENSION_COMPOSE__?: typeof compose;
  }
}
```

```

    }

    const composeEnhancer =
window.__REDUX_DEVTOOLS_EXTENSION_COMPOSE__ || compose;
export type RootStateType = ReturnType<typeof rootReducer>;
export type AppDispatch = typeof store.dispatch;
const store = createStore(
    reducer,
    composeEnhancer(applyMiddleware(...middleware)),
);
const persistor = persistStore(store);
sagaMiddleware.run(rootSaga);
export { store, persistor };

```

Усі ред'юсери необхідно з'єднувати, цим займається функція `combineReducers`. Код з використанням цієї функції наведено у лістингу 3.5.

Лістинг 3.5 – Код застосування функції `combineReducers`

```

import { combineReducers } from 'redux';
import mainReducer from '../Globals/MainPage/reducer';
import appReducer from '../Globals/App/reducer';
export const blackListReducers = {
    app: appReducer,
    main: mainReducer,
};
const rootReducer = combineReducers({...blackListReducers});
export default rootReducer;

```

Після цього потрібно створити та під'єднати сагу до Redux, для цього використана функція `createSagaMiddleware`. Код її використання наведений у лістингу 3.6.

Лістинг 3.6 – Код застосування функції `createSagaMiddleware`

```

import createSagaMiddleware from 'redux-saga';
export const sagaMiddleware = createSagaMiddleware();
const middleware = [sagaMiddleware];
export default middleware;

```

Для спостереження за певними подіями або діями та виконання відповідних саг була додана функція `WatcherSaga`, в додатку вона одна, та відстежує дії основної сторінки. Код цієї функції показано у лістингу 3.7.

Лістинг 3.7 – Код застосування функції `WatcherSaga`

```
export default function* WatcherSaga() {
  yield takeLatest(mainConstants.GET_PAGE, getPage);
  yield takeLatest(mainConstants.GET_PROCESSORS, getProcessors);
  yield takeLatest(mainConstants.GET_COOLERS, getCoolers);
  yield takeLatest(mainConstants.GET_MOTHERBOARDS,
getMotherboards);
  yield takeLatest(mainConstants.GET_RAMs, getRams);
  yield takeLatest(mainConstants.GET_HDDS, getHDDs);
  yield takeLatest(mainConstants.GET_SSDS, getSSDs);
  yield takeLatest(mainConstants.GET_POWERs, getPowers);
  yield takeLatest(mainConstants.GET_VIDEOCARDS, getVideocards);
  yield takeLatest(mainConstants.GET_CASES, getCases);
  yield takeLatest(mainConstants.CREATE, create);
}
```

Код з'єднання всіх цих функцій саги з `Redux` наведено у лістингу 3.8.

Лістинг 3.8 – Код відстеження всіх дій у додатку

```
import { all } from 'redux-saga/effects';
import WatcherMainSaga from '../Globals/MainPage/sagas';
export default function* rootSaga() {
  yield all([
    WatcherMainSaga(),
  ]);
}
```

Однією з задач є створення адаптивного дизайну для вебзастосунку, для цього було використано правило медіа-запитів мови `CSS`. Воно дозволяє налаштовувати стилі відповідно до різних пристроїв, розмірів екрану,

орієнтації тощо. Приклад медіа-запитів `@media` у програмі наведено у лістингу 3.9.

Лістинг 3.9 – Код застосування медіа-запитів `@media`

```
@media (max-width: 870px) {
    .select {
        width: 100%;
    }
    .accessoriesCharacteristics {
        width: 100%;
        flex-direction: column;
        justify-content: left;
    }
    .CharSubTitle {
        margin-top: 5px;
        margin-left: 0px;
    }
}
```

Збоку конфігуратора знаходиться меню, за допомогою якого виконується навігація між головною сторінкою та корзиною. Меню можна розгорнути та згорнути, воно також адаптовано під мобільні додатки. В меню відображене лого програми та іконки сторінок з підписами.

В лівій частині сторінки конфігуратора відображені назва комп'ютерної комплектуючої та випадаюче меню з різними параметрами. Після вибору знизу з'являється список назв комплектуючих, які також відповідають назвам товарів інтернет-магазину “Rozetka”. Коли була натиснута радіо-кнопка з назвою, то у правій частині екрану відображається інформація про вибрані комплектуючі, а саме назва, основні характеристики, картинка, опис (який можна розгорнути та згорнути) та ціна товару.

Компонент випадаючого списку було створено за допомогою бібліотеки `react-multi-select-component`, яка надає широкі можливості налаштування та відображення, такі як кастомізація стилів, можливість

відображення попереднього перегляду обраних елементів та інше. Код випадяючого списку представлено у лістингу 3.10.

Лістинг 3.10 – Код компоненту CustomSelect

```
import React, {useState, useEffect} from 'react';
import { MultiSelect } from "react-multi-select-component";
import './styles.scss';
const CustomMultiSelect = ({
  selected = [],
  setSelected,
  title,
  options = [],
  oneRequired = false,
  style,
  multi = false,
  placeholder = 'Не вибрано',
  titleStyle,
  disabled = false,
  className = "",
}) => {
  const [availableOptions, setAvailableOptions] =
    useState(options);
  const [allOptions, setAllOptions] = useState(options);
  const onChange = (value) => {
    if (oneRequired && value?.length === 0) {
      return;
    }
    if (!multi && value?.length >= 1) {
      setSelected(value.slice(-1));
      return;
    }
    setSelected(value);
  }
  useEffect(() => { setAvailableOptions(allOptions); },
    [selected, allOptions]);
  useEffect(() => { setAllOptions(options); }, [options]);
  const customValueSingleRenderer = (selected, _options) => {
    return selected?.length
```

```

      ? <>{selected.map(({ label }) =>
        <div                                className="customValueTextSingle"
key={label}>{label}</div>
      )} </>
      :
      <div                                <div
        className="customValueTextSingle">{placeholder}</div>;
    };

    const customItemRenderer = ({ option, onClick, disabled,
checked }) => {
      return (
        <div className={`customItem ${checked ? 'active'
          : ''}` onClick={onClick} disabled={disabled}>
          {option.label}
        </div>
      )
    };
    return (
      <div className="selectContainer" style={style}>
        {title ? <div className="selectTitle"
style={titleStyle}>
          {title}
        </div> : <></>}

      <MultiSelect
        options={availableOptions}
        value={selected}
        onChange={onChange}
        hasSelectAll={false}
        disableSearch={true}
        valueRenderer={customValueSingleRenderer}
        ItemRenderer={customItemRenderer}
        disabled={disabled}
        className={className}
      />
    </div>
  );
}
export default CustomMultiSelect;

```

Для перевірки сумісності різних комплектуючих по їх параметрам були створені окремі функції. Наприклад, щоб перевірити сумісність процесора та материнської плати треба перевірити відповідність сокетів (sockets) процесора та материнської плати. Після цього виходячи з покоління або назви процесора сформувати список чипсетів, які є найбільш сумісними з процесором та збалансованими по ціні материнської плати. Повернути материнські плати, чипсет яких є у масиві сформованих назв чипсетів. Код цієї перевірки можна побачити у лістингу 3.11.

Лістинг 3.11 – Код перевірки чипсету та сокету материнської плати при виборі процесора

```
const intelValues = ['i3', 'i5', 'i7', 'i9'];
let chipsetNumber = 3;
let chipsets = '';
if (intelValues.includes(data.value)) {
    chipsetNumber =
    data.label === 'Intel Core i3 8350K 4000 МГц' ? 3 :
    data.label.split('-')[1].slice(0,2) - 6
    chipsets =
    data.value === 'i3' ? ['H${chipsetNumber}10`,
`H${chipsetNumber}70`, `B${chipsetNumber}60`, `Z${chipsetNumber}90`] :
    data.value === 'i5' ? ['H${chipsetNumber}70`,
`B${chipsetNumber}60`, `Z${chipsetNumber}90`] : data.value === 'i7' ?
['B${chipsetNumber}60`, `Z${chipsetNumber}90`] : data.value === 'i9' ?
['Z${chipsetNumber}90`] : ''
    else if (data.socket === 'AM4'){
        chipsets = data.label === 'AMD Ryzen 3 3200G 3600 МГц' ||
data.label === 'AMD ` Ryzen 3 1200AF 3200 МГц' ? ['X470`,
`B450`, `A320`] :
        data.value === 'ryzen3' || data.label === 'AMD Ryzen 5
3600 3600 МГц' ? ['X470`, 'X570', `B450`, `B550`, 'A520'] :
        data.value === 'ryzen5' || data.value === 'ryzen7' ?
['X470`, 'X570', `B450`, `B550`] : data.value === 'ryzen9' ?
['X570', `B550`] : ''
    };
if (motherboardValue?.length > 0) {
```

```

        dispatch(mainActions.getMotherboards({value:
motherboardValue[0].value, socket: data.socket,
chipsets:chipsets})));
    };
    If (motherboardData?.data?.socket !== data.socket ||
!chipsets.includes((motherboardData?.data?.chipset))) {
        selectedMotherboard();
    };

```

Розрахунок необхідної потужності блока живлення виконується після вибору або зміни кожною з комп'ютерних комплектуючих. Динамічними є дані потужності процесора та відеокарти, інші дані статичні та додаються до загальної потужності, якщо ці комплектуючі були обрані або якщо вони відповідають необхідним параметрам. Наприклад кількість вентиляторів у корпусі або тип SSD диску. Код розрахунку загальної потужності системи наведено у лістингу 3.12.

Лістинг 3.12 – Код розрахунку загальної потужності системи

```

useEffect(() => {
    const ramPower = 10;
    const hddPower = hddValue?.length > 0 && hdd ? 15 : 0;
    const ssd1Power = ssdValue?.length > 0 && ssd ?
ssd1Data?.data?.isM2 ? 10 : 15 : 0;
    const ssd2Power = ssd2Value?.length > 0 && ssd2 ?
ssd2Data?.data?.isM2 ? 10 : 15 : 0;
    const coolerPower = pcCaseValue?.length > 0 && pcCase ?
caseData?.data?.coolers * 5 : 0;
    const motherboardPower = motherboardValue?.length > 0 &&
motherboard && processorData?.data?.label?.slice(0, 3) ===
'AMD' ? 50 : 0;
    const processorCoolerPower = coolerValue?.length > 0 &&
cooler ? coolerData?.data?.value === 'water_cooler' ? 25 :
5 : 0;
    const processorPower = processorValue?.length > 0 &&
processor ? processorData?.data?.power +
processorCoolerPower : 0;

```

```

    const videocardPower = videocardValue?.length > 0 &&
    videocard ? videocardData?.data?.power : 0;
    const total_power = ramPower + hddPower + ssd1Power +
    ssd2Power + coolerPower + motherboardPower + processorPower
    + videocardPower;
    setTotalPower(total_power);
    if (powerData?.data?.value < total_power * 1.2) {
        selectedPower();
    };
}, []
);

```

Готову збірку треба десь зберігати, для цього була створена корзина. Щоб зберегти збірку та щоб вона не зникла після оновлення сторінки або закриття браузера було використано localStorage. Цей об'єкт браузера надає методи для збереження отримання та видалення даних. Приклад використання наведено у лістингу 3.13.

Лістинг 3.13 – Метод збереження комплектуючих в корзині

```

localStorage.setItem('cartData', JSON.stringify(
    {
        processor: processorData,
        cooler: coolerData,
        motherboard: motherboardData,
        ram: ramData,
        power: powerData,
        case: caseData,
        videocard: videocardData,
        hdd: hddData,
        ssd1: ssd1Data,
        ssd2: ssd2Data,
    }
));

```

Для створення візуальної частини корзини було використано таблицю з бібліотеки rc-table та створено свій окремий компонент, з додаванням стилів

для його адаптації під різні пристрої та дизайн сайту. Компонент відображений у лістингу 3.14.

Лістинг 3.14 – Компонент таблиці корзини

```
const CustomTable = ({ columns, data, scroll = { y: 800, x: 450
}, onClick }) => {
  const onRowClick = (state) => {
    return {onClick: e => {onClick(state)}}
  }
  return (
    <Table
      columns={columns}
      data={data}
      scroll={scroll}
      onRow={onRowClick}
    />
  )
}
export default CustomTable;
```

Для додавання можливості відправки повідомлення з готовою збіркою на електронну пошту було створено форму заповненням адреси електронної пошти, використовуючи react-hook-form.

react-hook-form – це бібліотека для управління формами в React за допомогою хуків. Вона надає простий та потужний спосіб роботи з формами, дозволяючи легко валідувати та збирати дані з форм, обробляти помилки та здійснювати інші операції пов'язані з формами [27].

Після подавання форми дані комплектуючих відправляються на електронну пошту. Для розробки функції відправки на електронну пошту було взято бібліотеку email.js, яка дозволяє легко відправити електронний лист з клієнтської частини використовуючи API обраної електронної пошти, в нашому випадку gmail.com. Код функції приймання форми та відправки електронного листа відображено у лістингу 3.15.

Лістинг 3.15 – Функція відправки електронного листа

```

const onSubmit = (formData) => {
  const data = {
    processor: cartData.processor.href,
    cooler: cartData.cooler.href,
    motherboard: cartData.motherboard.href,
    ram: cartData.ram.href,
    power: cartData.power.href,
    pcCase: cartData.case.href,
    videocard: cartData?.videocard?.href || 'Без відеокарти',
    hdd: cartData?.hdd?.href || 'Без жорсткого диску',
    ssd: cartData?.ssd1?.href || 'Без SSD диску',
    ssd2: cartData?.ssd2?.href || 'Без SSD диску',
    email: formData.email,
  };

  emailjs.send('service_fzcpkxc', 'template_rp1o8xq',
data, 'user_xF9B2bI6TfnIIj3DRC5xs')
    .then(() => {
      if (emailSave) {
        localStorage.setItem('email',
JSON.stringify(formData.email));
        localStorage.setItem('emailSave',
JSON.stringify(emailSave));
      }
      else {
        localStorage.setItem('email', JSON.stringify(''));
        localStorage.setItem('emailSave',
JSON.stringify(false));};
        notify();
        setModalIsOpen(false);
      }, (error) => {
        alert(error.text);
      });
    });
};

```

Щоб виводити повідомлення користувачу, було розроблено компонент, основою якого є бібліотека react-hot-toast. Приклад створення повідомлення за допомогою цього компоненту наведено у лістингу 3.16.

Лістинг 3.16 – Створення повідомлення у вебзастосунку

```
const notify = () =>
toast.custom(
  <CustomToast
    title="Ваша збірка успішно додана до корзини"
    id="successCart"
    cart={true}
    theme='success'
  />,
  { id: 'successCart' }
);
```

Результатом розробки клієнтської частини є односторінковий сайт з адаптивним дизайном написаний за допомогою технології React з використанням сховища Redux. Для створення запитів до серверу була використана бібліотека Saga. Також у ході розробки необхідних компонентів були додані інші бібліотеки таблиці, форми, випадаючого списку та інші. Щоб зробити можливість відсилати електронного листа було вирішено взяти бібліотеку email.js.

3.3 Розробка серверної частини

3.3.1 Опис вхідних та вихідних даних

Перед розробкою серверної треба визначитися з вхідними та вихідними даними, які будуть надходити та відсилатися після виклику запиту. Вхідними даними є дані, які формуються алгоритмами перевірки на клієнтській частині вебдодатку. Вихідними є оброблені дані, взяті з бази даних, відфільтровані за параметрами пошуку, якими є вхідні дані, або сторінки товарів, надіслані API “Rozetka”, які відповідають назвам обраних комплектуючих.

Вхідні дані наведено у таблиці 3.1.

Таблиця 3.1 – Вхідні дані

Тип комплектуючої	Позначення змінної	Тип змінної	Значення змінної
1	2	3	4
Процесор	value	string	Покоління процесора
	processorsLabels	array	Назви процесорів сумісних з відеокартою
Охолодження (кулер процесора)	value	string	Тип охолодження
	TDP	integer	TDP процесора
	socket	string	Сокет процесора
Материнська плата	value	string	Виробник
	chipsets	array	Відфільтровані чипсети, сумісні з процесором
	socket	string	Сокет процесора
	isSSD	boolean	Відповідність типу SSD
Оперативна пам'ять	value	string	Об'єм пам'яті
	memoryType	string	Тип пам'яті
Постійний запам'ятовуючий пристрій	value	string	Об'єм пам'яті
Відеокарта	value	string	Назва відеокарти
Корпус	value	string	Виробник
	videocardWidth	integer	Розмір відеокарти
	coolerWidth	integer	Розмір кулера

Вихідні дані наведено у таблиці 3.2.

Таблиця 3.2 – Вихідні дані

Тип комплектуючої	Позначення змінної	Позначення параметра змінної	Тип параметра змінної	Значення параметра змінної
1	2	3	4	5
Процесор	Processor	value	string	Покоління процесора
		label	string	Назва товару
		socket	string	Сокет
		cores	integer	Кількість ядер
		threads	integer	Кількість потоків
		coreArhitecture	string	Архітектура
		maxFrequency	integer	Максимальна частота
		integrated Graphics	string	Назва інтегрованого відеочипу
		power	integer	Потужність
		L3cache	string	Кеш третього рівня
		rozetkaId	string	ID товару в інтернет-магазині “Rozetka”
Охолодження (кулер процесора)	Cooler	value	string	Тип охолодження
		label	string	Назва товару
		sockets	array	Підтримувані сокети

Продовження таблиці 3.2

1	2	3	4	5
		TDP	integer	Максимальна потужність
Материнська плата	Motherboard	width	integer	Розмір
		rozetkaId	string	ID товару в інтернет-магазині “Rozetka”
		value	string	Виробник
Охолодження (кулер процесора) Оперативна пам'ять	Cooler Ram	label	string	Назва товару
		chipset	string	Чипсет
		socket	string	Сокет
		rozetkaID	string	ID товару в інтернет-магазині “Rozetka”
		value	string	Об'єм пам'яті
		label	string	Назва товару
Материнська плата	Motherboard	memoryType	string	Тип пам'яті
		memoryFrequency	string	Частота
		rozetkaID	string	ID товару в інтернет-магазині “Rozetka”
Постійний запам'ятовуючий пристрій	Rom	value	string	Об'єм пам'яті
		label	string	Назва товару
		readSpeed	string	Швидкість читання (SSD)

Продовження таблиці 3.2

1	2	3	4	5
		writeSpeed	string	Швидкість запису (SSD)
		speed	string	Середня швидкість передачі даних
		spindleRotation Speed	string	Швидкість обертання шпинделю
		isSSD	boolean	Відповідність типу SSD
		isM2	boolean	Відповідність типу M2
		rozetkaID	string	ID товару в інтернет-магазині "Rozetka"
		value	string	Потужність
Блок живлення	Power	label	string	Назва товару
		rozetkaID	string	ID товару в інтернет-магазині "Rozetka"
Відеокарта	Videocard	value	string	Назва відеокарти
		label	string	Назва товару
		power	string	Потужність
		memoryBusWidth	string	Розрядність шини пам'яті
		memoryFrequency	string	Частота пам'яті
		coreFrequency	string	Частота ядра
		width	integer	Розмір

Продовження таблиці 3.2

1	2	3	4	5
		rozetkaID	string	ID товару в інтернет-магазині “Rozetka”
Корпус	Case	value	string	Виробник
		label	string	Назва товару
		type	string	Тип материнської плати
		coolers	integer	Кількість вентиляторів
		videocardMax Width	integer	Максимальний розмір відеокарти
		coolerMaxWidth	integer	Максимальний розмір кулера
		rozetkaID	string	ID товару в інтернет-магазині “Rozetka”

Так як програма працює зі стороннім API, то треба визначитися з основними даними, які потрібні для виконання запиту, та які потрібно взяти для використання у вебзастосунку:

а) вхідні дані API інтернет магазину “Rozetka”:

- 1) URL запиту;
- 2) front-type=xl – тип сторінки;
- 3) country=UA – країна запиту;
- 4) lang=ua – мова запиту;
- 5) goodsId=id – ідентифікатор товару;

б) вихідні дані API інтернет магазину “Rozetka”:

- 1) price – ціна товару;

- 2) description – опис товару;
- 3) images – картинки товару;
- 4) href – посилання на товар.

3.3.2 Розробка серверу на Node.js

Найголовніша ціль роботи сервера – відповідати на запити, які надходять від клієнтів та зберігати дані у БД. Для цих цілей була обрана середа NodeJs.

Розробка починається з команди `npm init`, для того щоб ініціалізувати роботу `npm`, після цього створиться файл `package.json` зі вмістом, який наведено у лістингу 3.17.

Лістинг 3.17 – `package.json` після команди `npm init`

```
{
  "name": "diplom",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

Для початку, необхідно встановити вебфреймворк Express, щоб це зробити потрібно виконати команду `npm install express`. Щоб можна було робити запити до бази даних, треба встановити бібліотеку `mongoose`, за допомогою команди `npm install mongoose`. Для того щоб виконувати HTTP запити до стороннього сервісу потрібно ще встановити `axios` – `npm install axios`.

Після того, як всі необхідні пакети встановлені, можна почати створювати програму. З початку треба створити файл `server.js` в якому потрібно з'єднатися з базою даних, підключити роути програми та хедери запитів, зробити обробник помилок та останнім шляхом запустити сервер. Під'єднання бази даних наведено у лістингу 3.18.

Лістинг 3.18 – З'єднання бази даних з сервером

```
const mongoose = require('mongoose');
const dotenv = require('dotenv');
dotenv.config();
const mongoDB
=`mongodb+srv://${process.env.DB_USER}:${process.env.DB_PASS}@cluster0.hyuxcsy.mongodb.net/configurator`;
mongoose.connect(mongoDB, {
  useCreateIndex: true,
  useNewUrlParser: true,
  useFindAndModify: false,
  useUnifiedTopology: true});
mongoose.Promise = global.Promise;
module.exports = mongoose;
```

Для створення серверу використана команда `createServer` та команда `listen` для прослуховування. Код запуску серверу можна побачити у лістингу 3.19.

Лістинг 3.19 – Запуск серверу

```
var http = require('http');
http.createServer(app).listen(process.env.PORT || 3000);
console.log(`Server has been started on port
${process.env.PORT}...`)
```

Після налаштування роботи серверу треба зайнятися обробкою запитів. Потрібно створити три папки, назви яких відповідають необхідним компонентам, які використовує кожен сервер: `routes`, `controllers`, `models`.

Ці компоненти працюють разом для створення структурованої та організованої архітектури вебдодатків.

Routes визначають кінцеві точки або URL-адреси, які вебпрограма надає для обробки вхідних запитів. Під кожну комп'ютерну комплектуючу створено окремі роути для читання, створення, оновлення та видалення (GET, POST, PUT DELETE) даних про ці комплектуючі. Приклад роутера показано у лістингу 3.20.

Лістинг 3.20 – Приклад роутера

```
const express = require('express');
const router = express.Router();
router.post('/create', caseController.create);
```

Controllers відповідають за обробку вхідних запитів, обробку необхідної логіки та повернення відповідної відповіді. Вони діють як посередники між маршрутами та моделями, отримуючи дані від клієнта, взаємодіючи з відповідними моделями чи службами та надсилаючи відповідь клієнту. Код контролера get для процесорів можна побачити у лістингу 3.21.

Лістинг 3.21 – Код контролера get

```
get: async function (req, res, next) {
  var value = req.query.value;
  var processorsLabels = req.query.processorsLabels;
  var processors;
  try {
    processors = await processorModel.find(processorsLabels &&
      value ? {label: { $in: processorsLabels.split(',') }, value:
        value} : value ? {value: value} : {});
    res.json({ status: 'success', message: 'Processors get
      successful', data: { processors: processors } });
  } catch (err) {
    res.json({ status: 'error', message: err, data: null });
  }
},
```

Models: представляють структури даних і взаємодію з базою даних. Код моделі для однієї з комплектуючих наведено у лістингу 3.22.

Лістинг 3.22 – Модель блока живлення

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const PowerSchema = new Schema({
  label: {
    type: String,
    required: true,
  },
  value: {
    type: String,
    required: true,
  },
  rozetkaId: {
    type: String,
    required: true,
  }
});
module.exports = mongoose.model('Power', PowerSchema);
```

Для роботи зі сторінками, які надходять з API інтернет-магазину “Rozetka” було додано роут `pages` з методом `GET`. Запит з параметрами виконується за допомогою бібліотеки `axios`. Код контролера для надсилання запиту та повернення необхідної інформації клієнту представлено у лістингу 3.23.

Лістинг 3.23 – Контролер `pages`

```
get: async function (req, res, next) {
  var id = req.query.id;
  try {
    const response = await
    axios.get(`${process.env.rozetka_url}get-main?front-
    type=xl&country=UA&lang=ua&goodsId=${id}`);
```

```

        res.json({ status: "success", message: `get data for ${id}
success`, data: response.data.data });
    }
    catch (err) {
        res.json({ status: "error", message: err, data: null
});
    }
},

```

В результаті ми отримали серверну частину вебзастосунку для перевірки сумісності комплектуючих та роботи зі стороннім API за допомогою технології NodeJS, вебфреймворку Express, з використанням бази даних Mongo DB та бібліотеки яка її з'єднує з кодом JavaScript – mongoose.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови застосування

Дане програмне забезпечення призначено для автоматизації процесу підбору сумісних комп'ютерних комплектуючих для збалансованої збірки ПК.

Для забезпечення коректної та безвідмовної роботи програми персональний комп'ютер повинен мати не менше 1 Gb оперативної пам'яті. Наявність клавіатури, миші, монітору.

Також версія браузера повинна бути не нижче ніж та що вказана у списку [28]:

- Google Chrome: Version 18;
- Mozilla Firefox: Version 14;
- Apple Safari: Version 6;
- Microsoft Edge: Version 12;
- Opera: Version 15.

4.2 Характеристика програми

Програма здійснює збалансовану збірку персонального комп'ютера за рахунок автоматичного підбору сумісних комплектуючих, що вирішує задачу користувача при виборі товарів, а саме відбору тільки кращих товарів у магазині (в даному вебзастосунку використовуються дані інтернет-магазину “Rozetka”).

Особливістю програми є проведення аналізу сумісності комплектуючих, які обирає користувач при здійсненні вибору під час функціонування програми. Перелік комплектуючих вказано в підрозділі 1.2.

4.3 Звертання до програми

Для запуску програми потрібно попередньо упевнитися в відповідності характеристик комп'ютера, на якому буде запускатись встановлена програма, до системних вимог програми. Другим кроком потрібно упевнитися в наявності всіх бібліотек для роботи програми та в наявності Node JS останньої версії. Третім кроком потрібно запустити серверну частину. Успішним запуском серверу є виведене у консолі речення `Server has been started on port 3001...`

Четвертим кроком потрібно запустити саму програму за допомогою командного рядка.

Звертання до програми передбачене одним способом: через вікно браузера.

Після запуску даного вебзастосунку відкривається браузер з вікном програми-помічника, в цьому вікні з'являється головна сторінка, що свідчить про початок роботи з програмою.

Критерієм оцінювання результатів та правильності виконання даної програми є наявність збірки ПК у кошику та вихідного файлу, який створюється вебзастосунком після передачі вихідних даних на поштову електронну адресу за бажанням користувача.

4.4 Вхідні та вихідні дані

Вхідними даними програми є комплектуючі, які аналізуються на предмет сумісності: процесор, кулер процесора, материнська плата, оперативна пам'ять, відеокарта, блок живлення, жорсткий диск, SSD диск, корпус. Вихідними даними програми є:

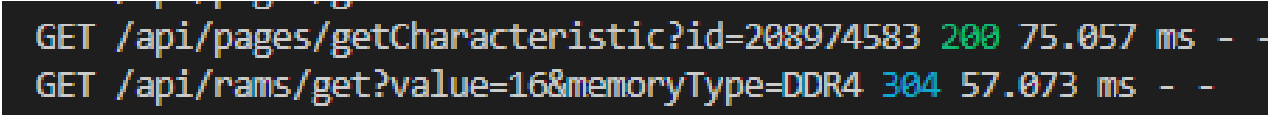
- збалансована збірка ПК, яку створив користувач у процесі функціонування програми, а саме шляхом вибору необхідних комплектуючих;

- інформація про комплектуючі з вказаними актуальними цінами та описом характеристик;
- корзина остаточного вибору користувача;
- відправлене повідомлення на поштову електронну адресу про створену збірку у корзині з необхідною інформацією щодо комплектуючих.

4.5 Повідомлення для програміста

Всі повідомлення для програміста вказані на рисунках 4.1 – 4.6.

Успішно виконаний запит з його статусом зображений далі на рисунку 4.1.



```
GET /api/pages/getCharacteristic?id=208974583 200 75.057 ms - -
GET /api/rams/get?value=16&memoryType=DDR4 304 57.073 ms - -
```

Рисунок 4.1 – Повідомлення успішного запиту

Помилка на стороні серверу або підключеної API має такий вигляд, як на рисунку 4.2.



```
{
  "data": null,
  "message": {
    "message": "getaddrinfo ENOTFOUND roetka.com.ua",
    "name": "Error",
    "code": "ENOTFOUND"
  },
  "config": {
    "transitional": {
      "silentJSONParsing": true,
      "forcedJSONParsing": true,
      "clarify": false
    },
    "message": "getaddrinfo ENOTFOUND roetka.com.ua",
    "name": "Error",
    "stack": "Error: getaddrinfo ENOTFOUND roetka.com.ua\n    at AxiosError.from (D:\\",
    "status": null
  },
  "status": "error"
}
```

Рисунок 4.2 – Повідомлення про помилку на стороні серверу

Якщо помилка на стороні бази даних, то видається повідомлення, зображене на рисунку 4.3.

```
MongoDB connection error: MongoError: bad auth : Authentication failed.
at MessageStream messageHandler (D:\ZNTU\Program\assistant for the col
```

Рисунок 4.3 – Повідомлення про помилку на стороні БД

Якщо помилка на стороні програми, то видається повідомлення, зображене на рисунку 4.4.

```
WebSocket connection to 'ws://localhost:3000/ws' failed:
WebSocket connection to 'ws://localhost:3000/ws' failed:
```

Рисунок 4.4 – Повідомлення про помилку на стороні програми

Успішно запущена програма має вигляд, який зображено на рисунку 4.5.

```
Compiled successfully!
You can now view diplom in the browser.

Local:      http://localhost:3000
On Your Network: http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
Files successfully emitted, waiting for typecheck results...
Issues checking in progress...
No issues found.
```

Рисунок 4.5 – Повідомлення про успішний запуск програми

Успішно запущений сервер має наступний вигляд, який зображено на рисунку 4.6.

```
> node_rest_api_jwt@1.0.0 start
> yarn serve

yarn run v1.22.4
$ node server.js
Server has been started on port 3001...
```

Рисунок 4.6 – Повідомлення про успішний запуск серверу

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Дане програмне забезпечення призначено для автоматизації процесу підбору сумісних комп'ютерних комплектуючих для збалансованої збірки ПК.

Процес функціонування програми полягає в обиранні користувачем:

- з випадającego списку типу комплектуючої ;
- шляхом натискання радіо-кнопки назви комплектуючої (після чого видаються опис комплектуючої, її характеристики, ціна та візуальне відображення).

Під час автоматичного аналізу у разі несумісності комплектуючих користувачу видається повідомлення для подальшої коректної роботи програми. В результаті послідовних дій користувача створюється збірка ПК, яка відображається в корзині, з можливістю подальшої відправки даної інформації на електронну пошту.

5.2 Умови виконання

Для виконання даної програми потрібно:

- встановити останню версію Node JS на офіційному сайті;
- перейти у терміналі до папки програми або запустити програму у середі розробки Visual Studio Code;
- встановити залежності для backend-частини (необхідні бібліотеки), для цього потрібно ввести команду `npm install` у терміналі;
- додати `.env` файл для запуску проєкту з необхідними змінними;
- (`secret_key = 'diplom', base_url = 'http://localhost/' front_url = 'http://localhost:3000/', rozetka_url = 'https://rozetka.com.ua/api/product-api/v4/goods/', PORT=3001, DB_USER=vladmomot, DB_PASS=v8l5a2d3`);
- запустити сервер, виконавши команду `npm start`;

- встановити залежності для frontend-частини (необхідні бібліотеки та фреймворк), для цього потрібно ввести команду `npm install` у терміналі;
- додати `.env` файл для запуску проєкту з необхідними змінними (`REACT_APP_API_URL = http://localhost:3001/api/`);
- запустити програму виконавши команду `npm start`.

5.3 Виконання програми

Візуальний вигляд форм для послідовного вибору користувачем всіх необхідних комп'ютерних комплектуючих у даному вебзастосунку зображено на рисунках 5.1-5.16. Форма вихідної інформації з описом характеристик комп'ютерних комплектуючих зображена на рисунку 5.17, а візуальне відображення корзини підсумкового вибору користувача має вигляд, зображений на рисунку 5.20. Якщо під час функціонування програми виникає несумісність комплектуючих, то є приклад такої ситуації, зображений на рисунку 5.18. Необхідна інформація, яка стосується можливості відправлення на адресу електронної пошти, вказана на рисунках 5.21-5.23.

Нижче наведено порядок виконання програми користувачем:

Першим кроком треба обрати виробника та покоління процесору у випадуючому списку, що розкривається (рис. 5.1).

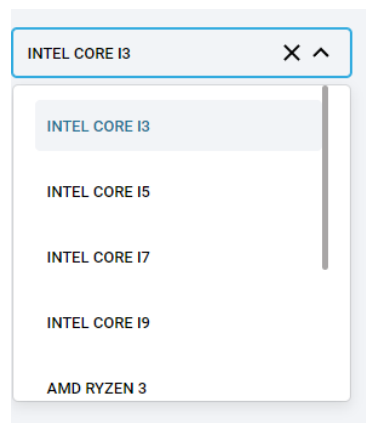


Рисунок 5.1 – Форма вибору типу процесору

Другим кроком обрати назву процесора, натиснувши на неї або на перемикач (рис. 5.2).

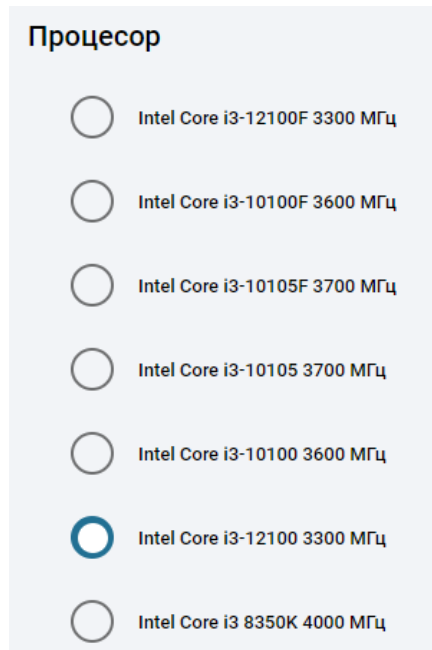


Рисунок 5.2 – Форма вибору назви процесору

Третім кроком обрати тип охолодження у випадяючому списку, що розкривається (рис. 5.3).

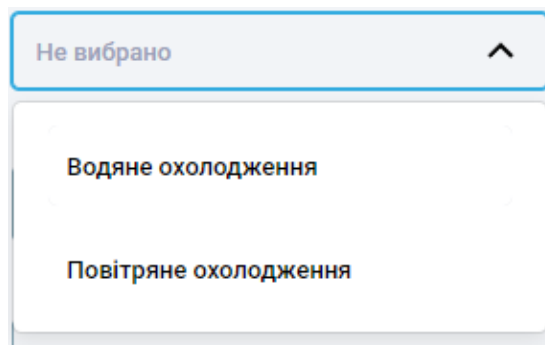


Рисунок 5.3 – Форма вибору типу охолодження

Четвертим кроком обрати назву охолодження, натиснувши на неї або на перемикач (рис. 5.4).

Охолодження

- ☐ be quiet! Silent Loop 2 280 мм
- ☐ ARCTIC Liquid Freezer II 280
- ☒ DeepCool Gammaxx L240 V2
- ☐ Fractal Design Celsius S24 Black
- ☐ Zalman Alpha 24 Black

Рисунок 5.4 – Форма вибору назви охолодження

П'ятим кроком обрати виробника материнської плати у випадяючому списку, що розкривається (рис. 5.5).

ASUS

- ASUS
- MSI
- GIGABYTE

Рисунок 5.5 – Форма вибору типу материнської плати

Шостим кроком обрати назву материнської плати, натиснувши на неї або на перемикач (рис. 5.6).

Материнська плата

- ☒ Asus ROG STRIX Z690-E Gaming Wi-Fi
- ☐ Asus PRIME H610M-A D4

Рисунок 5.6 – Форма вибору назви материнської плати

Сьомим кроком обрати кількість оперативної пам'яті у випадяючому списку, що розкривається (рис. 5.7).

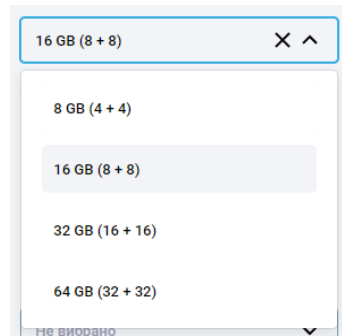


Рисунок 5.7 – Форма вибору кількості оперативної пам'яті

Восьмим кроком обрати назву оперативної пам'яті, натиснувши на неї або на перемикач (рис. 5.8).

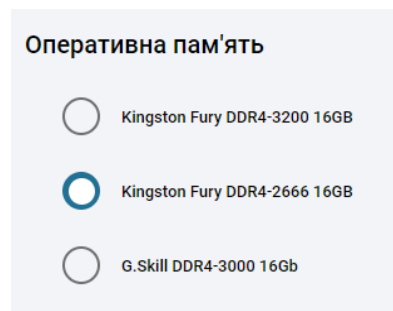


Рисунок 5.8 – Форма вибору назви оперативної пам'яті

Дев'ятим кроком обрати назву відеокарти у випадяючому списку, що розкривається (рис. 5.9).

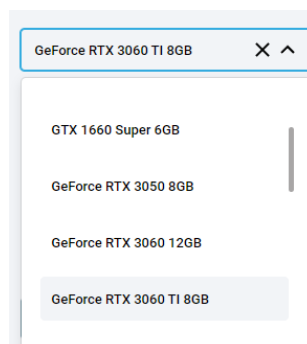


Рисунок 5.9 – Форма вибору назви відеокарти

Десятим кроком обрати виробника відеокарти, натиснувши на неї або на перемикач (рис. 5.10).

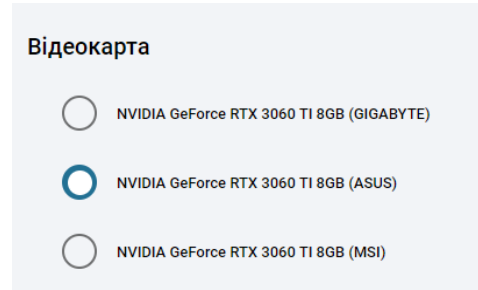


Рисунок 5.10 – Форма вибору виробника відеокарти

Одинадцятим кроком обрати потужність блока живлення у випадяючому списку, що розкривається (рис. 5.11).

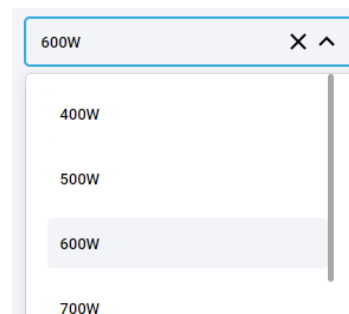


Рисунок 5.11 – Форма вибору потужності блока живлення

Дванадцятим кроком обрати назву блока живлення, натиснувши на неї або на перемикач (рис. 5.12).

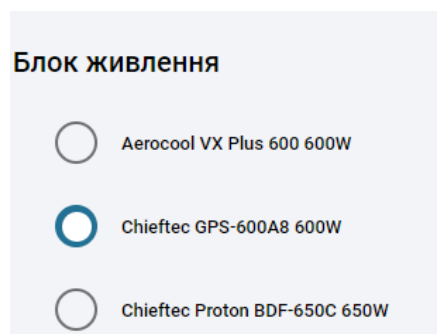


Рисунок 5.12 – Форма вибору назви блока живлення

Після цього обрати об'єм диску у випадаючому списку, що розкривається (для полів Жорсткий диск, SSD диск, SSD диск 2) або опцію «без HDD диску», якщо цей диск не потрібен (обов'язково повинен бути вибраний хоча б один диск) (рис. 5.13).

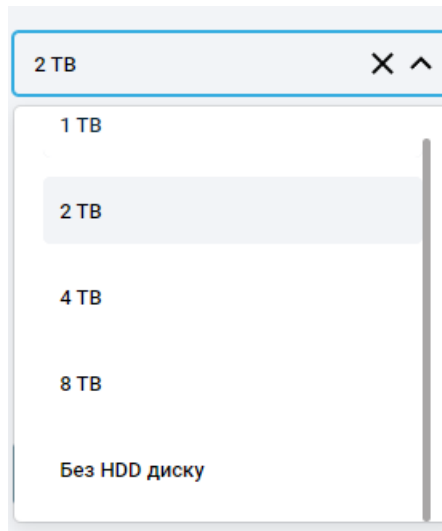


Рисунок 5.13 – Форма вибору об'єму HDD диску

Коли об'єм був вибраний, треба обрати назву диска, натиснувши на неї або на перемикач (рис. 5.14).



Рисунок 5.14 – Форма вибору назви HDD диску

Далі треба обрати виробника корпусу у випадаючому списку, що розкривається (рис. 5.15).

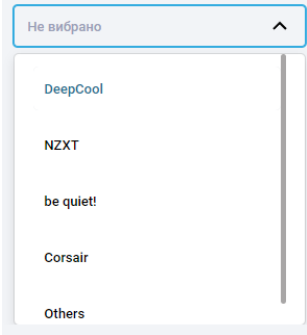


Рисунок 5.15 – Форма вибору виробника корпусу

Обрати назву корпусу, натиснувши на неї або на перемикач (рис. 5.16).



Рисунок 5.16 – Форма вибору назви корпусу

Переглянути вибрані комплектуючі та порівняти з іншими, дивлячись на основні характеристики та опис.

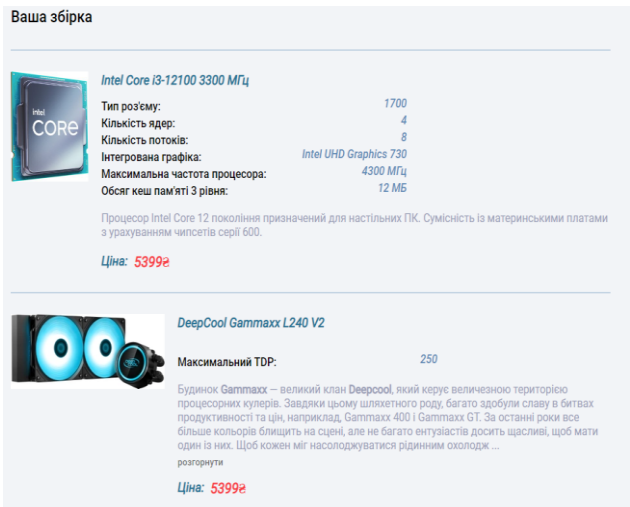


Рисунок 5.17 – Форма вихідної інформації опису характеристик комплектуючих

Виходячи з цього обрати інший процесор або відеокарту, якщо вони не є сумісними (рис. 5.18).

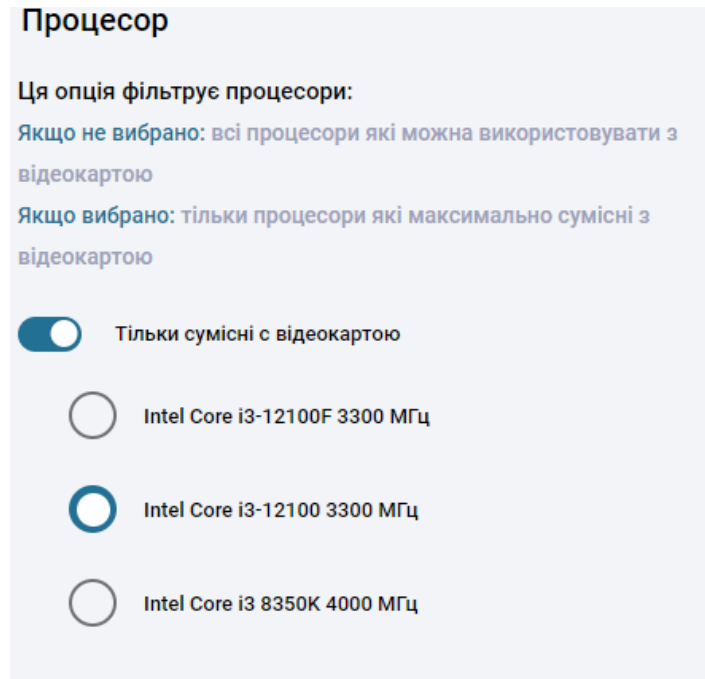


Рисунок 5.18 – Приклад несумісності комплектуючих

Якщо все обрано правильно в нижньому правому куті натиснути на кнопку «Додати вашу збірку до корзини» (рис. 5.19).

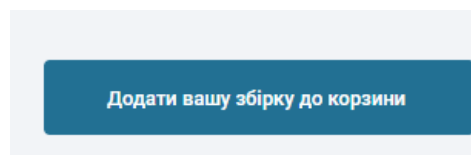


Рисунок 5.19 – Вигляд кнопки «Додати вашу збірку до корзини»

Після повідомлення о успішності додавання збірки перейти до корзини. Якщо натиснути на назви комплектуючих, то здійснюється перехід на сторінку покупки товару інтернет-магазину “Rozetka” (рис. 5.20).

Корзина

Натисніть на елемент таблиці для того щоб перейти на сторінку покупки товару

Тип	Назва	Ціна
Процесор	Intel Core i3-12100 3300 MTq	5399e
Охолодження	DeepCool Gammax L240 V2	3149e
Материнська плата	Asus ROG STRIX Z690-E Gaming Wi-Fi	17699e
Оперативна пам'ять	Kingston Fury DDR4-2666 16GB	1799e
Відеокарта	NVIDIA GeForce RTX 3060 Ti 8GB (ASUS)	19969e
Блок живлення	be quiet! Straight Power 11 750W	6619e
SSD диск	Samsung 870 Evo-Series 250GB	1409e

Рисунок 5.20 – Візуальне відображення корзини підсумкового вибору користувача

Якщо потрібно надіслати збірку на електронну пошту, для цього користувачу треба натиснути на кнопку «Надіслати збірку на пошту» (рис. 5.21).

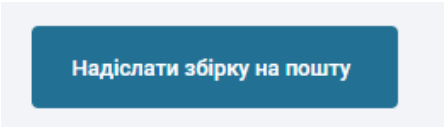


Рисунок 5.21 – Вигляд кнопки «Надіслати збірку на пошту»

Перед відправленням треба заповнити форму з електронною поштою та натиснути кнопку «Надіслати» (за необхідністю ввімкнути перемикач «Зберегти пошту», щоб не заповнювати її знову) (рис. 5.22).

← Закрити

Введіть вашу пошту

✉

☐

Зберегти пошту

Надіслати

Рисунок 5.22 – Форма для заповнення адреси електронної пошти

Останнім кроком перевірити електронну пошту на наявність повідомлення.

Дане повідомлення містить перелік назв комплектуючих та URL-адреси товарів інтернет-магазину “Rozetka” (рис. 5.23).

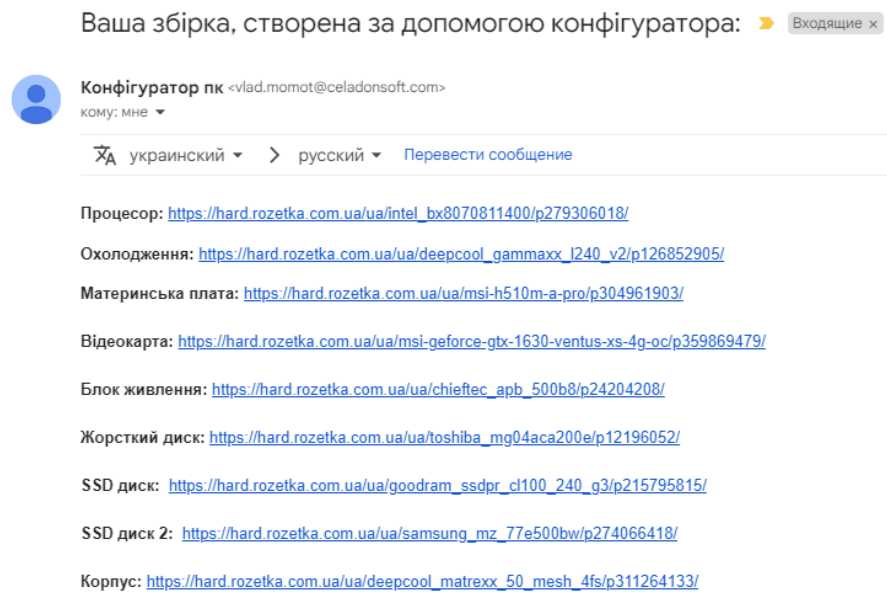


Рисунок 5.23 – Відображення повідомлення створеної збірки ПК, отримане користувачем на адресу електронної пошти

5.4 Повідомлення оператору

В процесі функціонування програми користувач може отримати одне з наступних повідомлень, які вказані на рисунках 5.24-5.33.

Коли всі обов’язкові комплектуючі вибрані та є сумісними, видається повідомлення, вигляд якого зображено на рисунку 5.24.



Рисунок 5.24 – Повідомлення успішності створення корзини

Якщо всі обов'язкові комплектуючі не вибрані або не є сумісними, то повідомлення має наступний вигляд, зображений на рисунку 5.25.

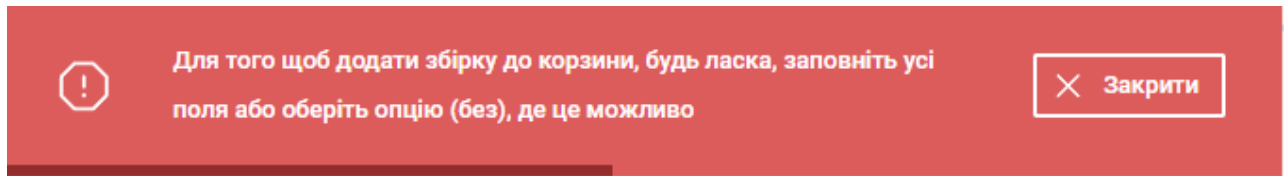


Рисунок 5.25 – Повідомлення неуспішності створення коззину

Коли введена електронна пошта є існуючою, та не виникло інших помилок при надсиланні електронного листа, видається таке повідомлення, як на рисунку 5.26.

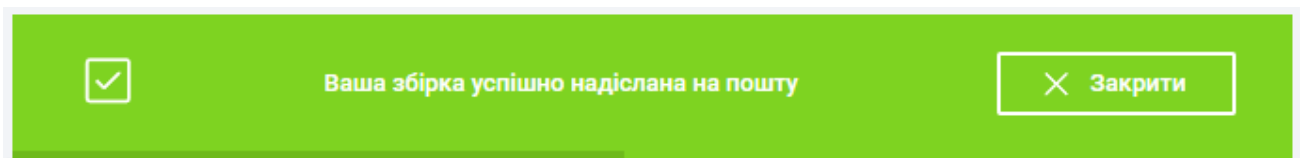


Рисунок 5.26 – Повідомлення успішності відправки вмісту коззину на адресу електронної пошти

Якщо користувач при введенні адреси електронної пошти зробив помилку, і така адреса не існує, то програмою передбачено повідомлення, зображене на рисунку 5.27.

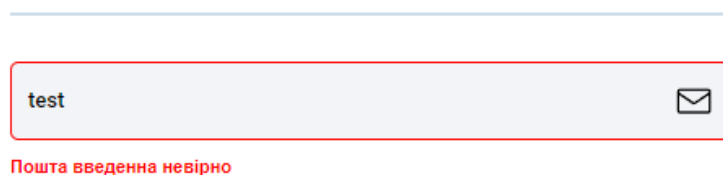
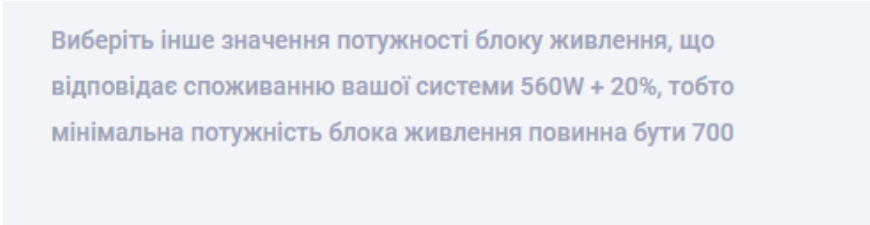


Рисунок 5.27 – Повідомлення невірної заповнення форми адреси електронної пошти

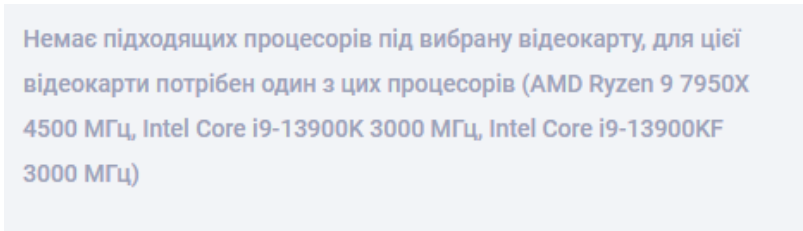
Якщо вибрана потужність блока живлення з випадуючого списку менше загальної потужності збірки, то видається повідомлення, вигляд якого зображено на рисунку 5.28.



Виберіть інше значення потужності блока живлення, що відповідає споживанню вашої системи 560W + 20%, тобто мінімальна потужність блока живлення повинна бути 700

Рисунок 5.28 – Повідомлення невірного вибору потужності блока живлення

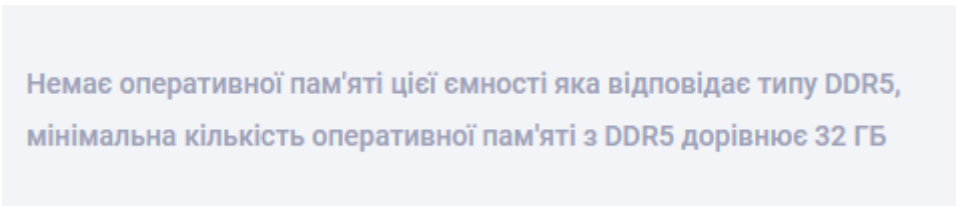
Коли для обраної відеокарти немає процесорів вибраного покоління (з'являється тільки тоді, коли обрана функція підбору збалансованих процесорів під відеокарту) програмою передбачено повідомлення, зображене на рисунку 5.29.



Немає підходящих процесорів під вибрану відеокарту, для цієї відеокарти потрібен один з цих процесорів (AMD Ryzen 9 7950X 4500 МГц, Intel Core i9-13900K 3000 МГц, Intel Core i9-13900KF 3000 МГц)

Рисунок 5.29 – Повідомлення специфіки підбору для відеокарти

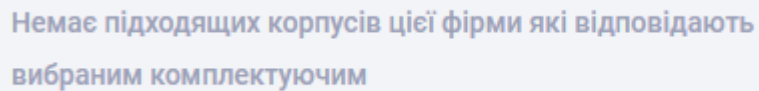
Повідомлення відсутності оперативної пам'яті вибраного об'єму для типу оперативної пам'яті, яку підтримує процесор, зображено на рисунку 5.30.



Немає оперативної пам'яті цієї ємності яка відповідає типу DDR5, мінімальна кількість оперативної пам'яті з DDR5 дорівнює 32 ГБ

Рисунок 5.30 – Повідомлення сумісного підбору для оперативної пам'яті

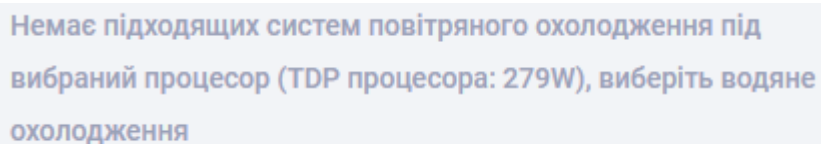
Повідомлення відсутності корпусів вибраного виробника, які відповідають розміру обраного охолодження та/або відеокарти, має вигляд, зображений на рисунку 5.31.



Немає підходящих корпусів цієї фірми які відповідають
вибраним комплектуючим

Рисунок 5.31 – Повідомлення відсутності вибраного виробника для корпусів

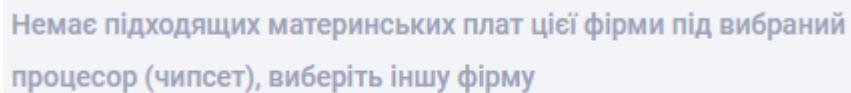
При відсутності систем охолодження обраного типу, які відповідають TDP процесора, видається повідомлення зображене на рисунку 5.32.



Немає підходящих систем повітряного охолодження під
вибраний процесор (TDP процесора: 279W), виберіть водяне
охолодження

Рисунок 5.32 – Повідомлення відсутності систем охолодження обраного
типу

Повідомлення відсутності материнських плат вибраного виробника, які відповідають чипсету процесора, зображено на рисунку 5.33.



Немає підходящих материнських плат цієї фірми під вибраний
процесор (чипсет), виберіть іншу фірму

Рисунок 5.33 – Повідомлення відсутності материнських плат вибраного
виробника

ВИСНОВКИ

В ході виконання даної роботи вирішено актуальну задачу розроблення автоматичного процесу підбору сумісних комп'ютерних комплектуючих з метою скорочення часу користувача під час збалансованої збірки ПК.

На етапі аналізу предметної області були вирішені наступні задачі:

- проведено пошук усіх комп'ютерних комплектуючих: процесор, кулер процесора, материнська плата, оперативна пам'ять, відеокарта, блок живлення, жорсткий диск, SSD диск, корпус;
- досліджено сумісність комплектуючих: процесора та материнської плати, процесора та кулера, процесора та оперативної пам'яті, відеокарти та процесора, корпусу та відеокарти, корпусу та кулера (окремо досліджено сумісність блока живлення з загальним споживанням системи).

Під час реалізації даного вебзастосунку:

- досліджено принципи створення веборієнтованих систем, види комп'ютерних комплектуючих, API інтернет-магазину “Rozetka”;
- визначено особливості характеристик комплектуючих;
- обрано засіб аналізу сумісності комплектуючих;
- спроектовано базу даних та архітектуру програми;
- проведено огляд існуючих аналогів програмного забезпечення;
- зроблено порівняльний аналіз мов та середовищ програмування, баз даних, фреймворків для створення клієнтської частини.

Даний вебзастосунок створено з використанням база даних Mongo DB, мова програмування JavaScript, які були обрані як найкращі після проведеного аналізу та порівняння з іншими існуючими на даний момент базами даних та мов програмування. Проаналізувавши найпопулярніші бібліотеки та фреймворки для створення клієнтської частини вебдодатків було обрано бібліотеку React. Для розробки серверної частини вебзастосунку було використано Node.js, бо воно є фаворитом при створенні швидких та

масштабованих мережових додатків. Завдяки своїй асинхронності та ефективному використанню ресурсів, Node.js дозволяє забезпечити високу продуктивність серверних додатків та оптимальне використання апаратних ресурсів. Отже, в процесі виконання роботи були використані такі методи та технічні засоби: мова гіпертекстової розмітки HTML, база даних Mongo DB, мова програмування JavaScript, каскадна таблиця стилів SASS, фреймворк React, крос-платформне середовище виконання JavaScript – Node.js, фреймворк вебдодатків для Node.js – Express.

Реалізовані наступні функції програмного забезпечення:

- можливість вибрати комплектуючі зі списку та фільтрування списків, з функцією виведення тільки сумісних комплектуючих;
- виведення інформації о товарах інтернет-магазину “Rozetka”;
- повідомлення користувачеві про відсутність сумісності відповідних комплектуючих;
- перевірка збалансованості збірки (процесора та відеокарти);
- збереження готової збірки у програмі та можливість її відправлення на електронну пошту.

Було розроблено адаптивний вебзастосунок для підвищення ефективності взаємодії розділу комп’ютерних комплектуючих інтернет-магазину “Rozetka” з користувачем або для індивідуального використання для скорочення часу користувача під час збірки ПК, вирішені поставлені задачі проєкту. Практична цінність даного програмного забезпечення полягає в тому, що воно надає можливості покращити інформаційний обмін між інтернет-магазинами та користувачами. Галуззю використання є комп’ютерне ПО для домашнього використання або інтернет-магазини, які реалізують комп’ютерні комплектуючі.

Даний проєкт в майбутньому можливо модернізувати, додавши окремо візуальне відображення адмін-панелі для створення актуалізації вебзастосунку з можливістю оновлення інформації шляхом додавання і редагування комплектуючих.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Scheme of the main components of the pc? [Electronic resource]. – Access mode: <https://computerinfobits.com/parts-of-computer-and-their-functions/>.
2. Introduction to Computer Hardware [Electronic resource]. – Access mode: <https://www.crucial.com/articles/pc-builders/what-is-computer-hardware>.
3. CAN [Електрон. ресурс]. – Режим доступу: <https://can.ua/configurator/>.
4. PCPartPicker [Electronic resource]. – Access mode: <https://pcpartpicker.com/list/>.
5. Newegg [Electronic resource]. – Access mode: <https://www.newegg.com/tools/custom-pc-builder/>.
6. Jones D. JavaScript: Novice to Ninja / Darren Jones. – [S. l.] : SitePoint, 2017. – 664 p.
7. PHP [Electronic resource]. – Access mode: <https://www.sitepoint.com/php-beginners-guide/>.
8. Python [Electronic resource]. – Access mode: <https://pythoninstitute.org/about-python>.
9. What is an IDE [Electronic resource]. – Access mode: <https://www.redhat.com/en/topics/middleware/what-is-ide>.
10. WebStorm [Electronic resource]. – Access mode: <https://www.jetbrains.com/webstorm/>.
11. Bruce J. Visual Studio Code: End-To-End Editing and Debugging Tools for Web Developers / Johnson Bruce. – [S. l.] : Wiley & Sons, Incorporated, John, 2019. – 312 p.
12. Система контролю версій [Електрон. ресурс]. – Режим доступу: <https://git-scm.com/book/uk/v2/Вступ-Про-систему-контролю-версій>.
13. MySQL [Electronic resource]. – Access mode: <https://www.oracle.com/mysql/what-is-mysql/>.

14. Chodorow K. MongoDB: The Definitive Guide / Kristina Chodorow, Michael Dirolf. – [S. l.] : O'Reilly Media, Incorporated, 2010. – 511 p.
15. Mongoose [Electronic resource]. – Access mode: <https://mongoosejs.com/>.
16. Banks A. Learning React: Modern Patterns for Developing React Apps / Alex Banks, Eve Porcello. – [S. l.] : O'Reilly Media, 2020. – 310 p.
17. JSX [Electronic resource]. – Access mode: https://www.w3schools.com/react/react_jsx.asp.
18. Fain Y. Angular Development with TypeScript / Yakov Fain, Anton Moiseev. – [S. l.] : Manning Publications Co. LLC, 2018. – 560 p.
19. Kyriakidis A. Majesty of Vue. Js / Alex Kyriakidis, Kostas Maniatis. – [S. l.] : Packt Publishing, Limited, 2016. – 240 p.
20. Herron D. Node Web Development / David Herron. – [S. l.] : Packt Publishing, Limited, 2013. – 376 p.
21. Node.js [Электрон. ресурс]. – Режим доступа: <https://nodejs.org/en>.
22. NPM and NPX [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/what-are-the-differences-between-npm-and-npx/#>.
23. Atencio L. Functional Programming in JavaScript: How to improve your JavaScript programs using functional techniques / Luis Atencio. – [S. l.] : Manning Publications, 2016. – 272 p.
24. Hooks in react [Electronic resource]. – Access mode: <https://legacy.reactjs.org/docs/hooks-intro.html>.
25. Chinnathambi K. Learning React: A Hands-On Guide to Building Web Applications Using React and Redux / Kirupa Chinnathambi. – [S. l.] : Pearson Education, Limited, 2018. – 286 p.
26. Redux Saga [Electronic resource]. – Access mode: <https://github.com/redux-saga/redux-saga>.
27. What is react-hook-form? [Electronic resource]. – Access mode: <https://blog.logrocket.com/react-hook-form-complete-guide/>.

28. React browser support [Electronic resource]. – Access mode:
<https://caniuse.com/?search=react>

ДОДАТОК А
Текст програми

```

/src/MainPage/index.js
import React, { useState, useEffect } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import CustomMultiSelect from
'../../Components/CustomMultiSelect';
import RadioInput from '../../Components/RadioInput';
import { mainActions } from '../../Globals/MainPage/actions';
import {
  processors,
  coolers,
  motherboards,
  rams,
  videocards,
  ssds,
  hdds,
  powers,
  cases,
  maxVideocard1660,
  maxVideocard3050,
  maxVideocard3060ti,
  maxVideocard3070ti,
  maxVideocard4070,
  maxVideocard4070ti,
  maxVideocard4080,
  maxVideocard4090,
  maxVideocard1660_All,
  maxVideocard3050_All,
  maxVideocard3060ti_All,
  maxVideocard3070ti_All,
  maxVideocard4070_All,
  maxVideocard4070ti_All,
  maxVideocard4080_All,
  maxVideocard4090_All
} from './options';
import './styles.scss';
import CustomButton from '../../Components/CustomButton';
import LineDivider from '../../Components/LineDivider';
import CustomToast from '../../Components/CustomToast';
import toast, { Toaster } from 'react-hot-toast';

```

```

import Switch from '../../Components/Switch';

const MainPage = () => {
  const dispatch = useDispatch();
  const processorData = useSelector((state) =>
state.main.processorData);
  const processorsData = useSelector((state) =>
state.main.processors);
  const coolerData = useSelector((state) =>
state.main.coolerData);
  const coolersData = useSelector((state) =>
state.main.coolers);
  const motherboardData = useSelector((state) =>
state.main.motherboardData);
  const motherboardsData = useSelector((state) =>
state.main.motherboards);
  const ramData = useSelector((state) => state.main.ramData);
  const ramsData = useSelector((state) => state.main.rams);
  const powerData = useSelector((state) =>
state.main.powerData);
  const powersData = useSelector((state) =>
state.main.powers);
  const videocardData = useSelector((state) =>
state.main.videocardData);
  const videocardsData = useSelector((state) =>
state.main.videocards);
  const ssd1Data = useSelector((state) =>
state.main.ssd1Data);
  const ssd2Data = useSelector((state) =>
state.main.ssd2Data);
  const hddData = useSelector((state) => state.main.hddData);
  const HDDsData = useSelector((state) => state.main.HDDs);
  const SSDsData = useSelector((state) => state.main.SSDs);
  const SSD2sData = useSelector((state) => state.main.SSD2s);
  const caseData = useSelector((state) =>
state.main.caseData);
  const casesData = useSelector((state) => state.main.cases);
  const [processor, selectedProcessor] = useState();
  const [processorValue, setProcessorValue] = useState();

```

```

const [cooler, selectedCooler] = useState();
const [coolerValue, setCoolerValue] = useState();
const [motherboard, selectedMotherboard] = useState();
const [motherboardValue, setMotherboardValue] = useState();
const [ram, selectedRam] = useState();
const [ramValue, setRamValue] = useState();
const [power, selectedPower] = useState();
const [powerValue, setPowerValue] = useState();
const [videocard, selectedVideocard] = useState();
const [videocardValue, setVideocardValue] = useState();
const [hdd, selectedHdd] = useState();
const [hddValue, setHddValue] = useState();
const [ssd, selectedSsd] = useState();
const [ssdValue, setSsdValue] = useState();
const [ssd2, selectedSsd2] = useState();
const [ssd2Value, setSsd2Value] = useState();
const [pcCase, selectedCase] = useState();
const [pcCaseValue, setCaseValue] = useState();
const [processorDescriptionLength,
setProcessorDescriptionLength] = useState(600);
    const [coolerDescriptionLength, setCoolerDescriptionLength]
= useState(400);
    const [motherboardDescriptionLength,
setMotherboardDescriptionLength] = useState(600);
    const [ramDescriptionLength, setRamDescriptionLength] =
useState(400);
    const [videocardDescriptionLength,
setVideocardDescriptionLength] = useState(600);
    const [powerDescriptionLength, setPowerDescriptionLength] =
useState(400);
    const [hddDescriptionLength, setHddDescriptionLength] =
useState(400);
    const [ssd1DescriptionLength, setSsd1DescriptionLength] =
useState(400);
    const [ssd2DescriptionLength, setSsd2DescriptionLength] =
useState(400);
    const [caseDescriptionLength, setCaseDescriptionLength] =
useState(400);

```

```

    const [processorsForVideocard, setProcessorsForVideocard] =
useState([]);
    const [totalPower, setTotalPower] = useState();
    const [onlyBestVideocards, setOnlyBestVideocards] =
useState(false)

    useEffect(() => {
        const ramPower = 10;
        const hddPower = hddValue?.length > 0 && hdd ? 15 : 0;
        const ssd1Power = ssdValue?.length > 0 && ssd ?
ssd1Data?.data?.isM2 ? 10 : 15 : 0;
        const ssd2Power = ssd2Value?.length > 0 && ssd2 ?
ssd2Data?.data?.isM2 ? 10 : 15 : 0;
        const coolerPower = pcCaseValue?.length > 0 && pcCase ?
caseData?.data?.coolers * 5 : 0;
        const motherboardPower = motherboardValue?.length > 0 &&
motherboard && processorData?.data?.label?.slice(0, 3) ===
'AMD' ? 50 : 0;
        const processorCoolerPower = coolerValue?.length > 0 &&
cooler ? coolerData?.data?.value === 'water_cooler' ? 25 : 5
: 0;
        const processorPower = processorValue?.length > 0 &&
processor ? processorData?.data?.power +
processorCoolerPower : 0;
        const videocardPower = videocardValue?.length > 0 &&
videocard ? videocardData?.data?.power : 0;
        const total_power = ramPower + hddPower + ssd1Power +
ssd2Power + coolerPower + motherboardPower + processorPower
+ videocardPower;
        setTotalPower(total_power);
        if (powerData?.data?.value < total_power * 1.2) {
            selectedPower();
        };
    }, [
        processorData,
        videocardData,
        coolerData,
        caseData,
        ssd1Data,

```

```

        ssd2Data,
        hdd,
        ssd,
        ssd2,
        pcCase,
        motherboard,
        cooler,
        processor,
        videocard,
        powerData,
        pcCaseValue,
        hddValue,
        ssdValue,
        ssd2Value,
        videocardValue,
        processorValue,
        motherboardValue,
        coolerValue
    ]
);

const setProcessor = (data) => {
    setProcessorValue(data);
    setProcessorDescriptionLength(600);
    selectedProcessor();
    if (ramValue?.length > 0 && ramValue[0].value === '8'
        && ['i5', 'i7', 'i9', 'ryzen5', 'ryzen7',
'ryzen9'].includes(data[0].value)) {
        selectedRam();
    };

    const maxVideocard_Best =

maxVideocard1660.videocards.includes(videocardData?.data?.value) ?
maxVideocard1660 :

maxVideocard3050.videocards.includes(videocardData?.data?.value) ?
maxVideocard3050 :

```

```

maxVideocard3060ti.videocards.includes(videocardData?.data?.
value) ? maxVideocard3060ti :
    maxVideocard3070ti.videocards.includes(videocardData?.data
?.value) ? maxVideocard3070ti :

maxVideocard4070.videocards.includes(videocardData?.data?.value) ?
maxVideocard4070 :

    maxVideocard4070ti.videocards.includes(videocardData?.data?.
value) ? maxVideocard4070ti :

maxVideocard4080.videocards.includes(videocardData?.data?.value) ?
maxVideocard4080 :

    maxVideocard4090.videocards.includes(videocardData?.data?.va
lue) ? maxVideocard4090 : '';

    const maxVideocard_All =

maxVideocard1660_All.videocards.includes(videocardData?.data
?.value) ? maxVideocard1660_All :
    maxVideocard3050_All.videocards.includes(videocardData?.da
ta?.value) ? maxVideocard3050_All :
    maxVideocard3060ti_All.videocards.includes(videocardData?.
data?.value) ? maxVideocard3060ti_All :
    maxVideocard3070ti_All.videocards.includes(videocardData?.
data?.value) ? maxVideocard3070ti_All :
    maxVideocard4070_All.videocards.includes(videocardData?.da
ta?.value) ? maxVideocard4070_All :
    maxVideocard4070ti_All.videocards.includes(videocardData?.
data?.value) ? maxVideocard4070ti_All :
    maxVideocard4080_All.videocards.includes(videocardData?.da
ta?.value) ? maxVideocard4080_All :
    maxVideocard4090_All.videocards.includes(videocardData?.da
ta?.value) ? maxVideocard4090_All : '';

    const maxVideocard = onlyBestVideocards ?
maxVideocard_Best : maxVideocard_All;

```

```

        setProcessorsForVideocard( maxVideocard?.processors ||
[]);

        dispatch(mainActions.getProcessors({value: data[0].value,
processorsLabels: maxVideocard?.processors || ''}));
    };

    const handleChangeProcessor = (data) => {
        selectedProcessor(data.rozetkaId);
        dispatch(mainActions.getPage({id: data.rozetkaId, data:
data, pageType: 'processor'}));
        setProcessorDescriptionLength(600);

        if (coolerValue?.length > 0) {
            dispatch(mainActions.getCoolers({
                value: coolerValue[0].value,
                socket: data.socket,
                tdp: (data.power * 0.9).toFixed(0) || ''
            }));
        };

        if (!coolerData?.data?.sockets.includes(data.socket)) {
            selectedCooler();
        };

        const intelValues = ['i3', 'i5', 'i7', 'i9'];
        let chipsetNumber = 3;
        let chipsets = '';
        if (intelValues.includes(data.value)) {
            chipsetNumber =
                data.label === 'Intel Core i3 8350K 4000 МГц' ? 3 :
                data.label.split('-')[1].slice(0,2) - 6
            chipsets =
                data.value === 'i3' ? [`H${chipsetNumber}10`,
`H${chipsetNumber}70`, `
                B${chipsetNumber}60`,
`Z${chipsetNumber}90`] :
                data.value === 'i5' ? [`H${chipsetNumber}70`,
`B${chipsetNumber}60`, `Z${chipsetNumber}90`] :

```

```

        data.value === 'i7' ? [`B${chipsetNumber}60`,
`Z${chipsetNumber}90`] :
        data.value === 'i9' ? [`Z${chipsetNumber}90`] : ''
    }
    else if (data.socket === 'AM4'){
        chipsets =
        data.label === 'AMD Ryzen 3 3200G 3600 МГц' || data.label
=== 'AMD Ryzen 3 1200AF 3200 МГц' ? [`X470`, `B450`, `A320`]
:
        data.value === 'ryzen3' || data.label === 'AMD Ryzen 5
3600 3600 МГц' ? [`X470`, 'X570', `B450`, `B550`, 'A520'] :
        data.value === 'ryzen5' || data.value === 'ryzen7' ?
[`X470`, 'X570', `B450`, `B550`] :
        data.value === 'ryzen9' ? ['X570', `B550`] : ''
    };
    if (motherboardValue?.length > 0) {
dispatch(mainActions.getMotherboards({value:
motherboardValue[0].value, socket: data.socket, chipsets:
chipsets}));
    };

    if (motherboardData?.data?.socket !== data.socket ||
!chipsets.includes((motherboardData?.data?.chipset))) {
        selectedMotherboard();
    };

    const memoryType =
        data.socket === 'AM5' ||
        data.label.split('-')[1].slice(0,2) === '13' ? 'DDR5' :
'DDR4'

    if (ramValue?.length > 0) {
        dispatch(mainActions.getRams({value: ramValue[0].value,
memoryType: memoryType}));
    };

    if (memoryType !== ramData?.data?.memoryType) {
        selectedRam();
    };

```

```

        if (data?.integratedGraphics !== undefined) {
            if (videocards.filter(item => item.value ===
'without_videocard')?.length > 0) {
                videocards.pop()
            }
            videocards.push({
                label: 'Без відеокарти', value: 'without_videocard'
            });
        }
        else {
            if (videocards.filter(item => item.value ===
'without_videocard')?.length > 0) {
                videocards.pop()
            }
        }
    };

    const setCooler = (data) => {
        setCoolerValue(data);
        dispatch(mainActions.getCoolers({
            value: data[0].value,
            socket: processorData?.data?.socket || '',
            tdp: (processorData?.data?.power * 0.9).toFixed(0) || ''
        }));
        selectedCooler();
        setCoolerDescriptionLength(400);
    };

    const handleChangeCooler = (data) => {
        selectedCooler(data.rozetkaId);
        dispatch(mainActions.getPage({id: data.rozetkaId, data:
data, pageType: 'cooler'}));
        setCoolerDescriptionLength(400);

        if (caseData?.data?.coolerMaxWidth <= data.width) {
            selectedCase();
        };

        if (pcCaseValue?.length > 0) {

```

```

        dispatch(mainActions.getCases({value:
pcCaseValue[0].value, videocardWidth:
videocardData?.data?.width || '', coolerWidth: data.width
}));
    };
};

const setMotherboard = (data) => {
    setMotherboardValue(data);
    setMotherboardDescriptionLength(600);
    selectedMotherboard();

    let chipsets = '';
    if (processorData?.data) {
        const intelValues = ['i3', 'i5', 'i7', 'i9'];
        let chipsetNumber = 3;

        if (intelValues.includes(processorData?.data?.value)) {
            chipsetNumber =
                processorData?.data?.label === 'Intel Core i3 8350K
4000 МГц' ? 3 :
                processorData?.data?.label.split('-')[1].slice(0,2)

            chipsets =
                processorData?.data?.value === 'i3' ?
                [`H${chipsetNumber}10`, `H${chipsetNumber}70`,
                `B${chipsetNumber}60`, `Z${chipsetNumber}90`] :
                processorData?.data?.value === 'i5' ?
                [`H${chipsetNumber}70`, `B${chipsetNumber}60`,
                `Z${chipsetNumber}90`] :
                processorData?.data?.value === 'i7' ?
                [`B${chipsetNumber}60`, `Z${chipsetNumber}90`] :
                processorData?.data?.value === 'i9' ?
                [`Z${chipsetNumber}90`] : ''
        }
    }
    else if (data.socket === 'AM4') {
        chipsets =
            processorData?.data?.label === 'AMD Ryzen 3 3200G 3600
МГц' || processorData?.data?.label === 'AMD Ryzen 3
1200AF 3200 МГц' ? [`X470`, `B450`, `A320`] :

```

```

        processorData?.data?.value === 'ryzen3' ||
        processorData?.data?.label === 'AMD Ryzen 5 3600 3600
        МГц' ? ['X470`, 'X570`, `B450`, `B550`, 'A520'] :
        processorData?.data?.value === 'ryzen5' ||
        processorData?.data?.value === 'ryzen7' ? ['X470`,
        'X570`, `B450`, `B550`] :
        processorData?.data?.value === 'ryzen9' ? ['X570`,
        `B550`] : ''
    };
};

dispatch(mainActions.getMotherboards({value: data[0].value,
socket: processorData?.data?.socket || '', chipsets:
chipsets})));
};

const handleChangeMotherboard = (data) => {
    selectedMotherboard(data.rozetkaId);
    dispatch(mainActions.getPage({id: data.rozetkaId, data:
data, pageType: 'motherboard'})));
    setMotherboardDescriptionLength(600);
};

const setVideocard = (data) => {
    setVideocardValue(data);
    dispatch(mainActions.getVideocards({value:
data[0].value})));
    selectedVideocard();
    setVideocardDescriptionLength(600);
};

const handleChangeVideocard = (data) => {
    selectedVideocard(data.rozetkaId);
    dispatch(mainActions.getPage({id: data.rozetkaId, data:
data, pageType: 'videocard'})));
    setVideocardDescriptionLength(600);

    if (caseData?.data?.videocardMaxWidth <= data.width) {
        selectedCase();
    }
};

```

```

};

if (pcCaseValue?.length > 0) {
  dispatch(mainActions.getCases({value:
pcCaseValue[0].value, videocardWidth: data.width,
coolerWidth: coolerData?.data?.width || '' }));
};

const maxVideocard_Best =
  maxVideocard1660.videocards.includes(data.value) ?
maxVideocard1660 :
  maxVideocard3050.videocards.includes(data.value) ?
maxVideocard3050 :
  maxVideocard3060ti.videocards.includes(data.value) ?
maxVideocard3060ti :
  maxVideocard3070ti.videocards.includes(data.value) ?
maxVideocard3070ti :
  maxVideocard4070.videocards.includes(data.value) ?
maxVideocard4070 :
  maxVideocard4070ti.videocards.includes(data.value) ?
maxVideocard4070ti :
  maxVideocard4080.videocards.includes(data.value) ?
maxVideocard4080 :
  maxVideocard4090;

const maxVideocard_All =
  maxVideocard1660_All.videocards.includes(data.value) ?
maxVideocard1660_All :
  maxVideocard3050_All.videocards.includes(data.value) ?
maxVideocard3050_All :
  maxVideocard3060ti_All.videocards.includes(data.value) ?
maxVideocard3060ti_All :
  maxVideocard3070ti_All.videocards.includes(data.value) ?
maxVideocard3070ti_All :
  maxVideocard4070_All.videocards.includes(data.value) ?
maxVideocard4070_All :
  maxVideocard4070ti_All.videocards.includes(data.value) ?
maxVideocard4070ti_All :

```

```

        maxVideocard4080_All.videocards.includes(data.value) ?
maxVideocard4080_All :
        maxVideocard4090_All;

    const maxVideocard = onlyBestVideocards ?
maxVideocard_Best : maxVideocard_All;
    setProcessorsForVideocard(maxVideocard.processors);
    if (processorValue?.length > 0) {
        dispatch(mainActions.getProcessors({value:
processorValue[0].value, processorsLabels:
maxVideocard.processors}));
    };
    if
(!maxVideocard.processors.includes(processorData.data.label)) {
        selectedProcessor();
    };
};

const setRam = (data) => {
    setRamValue(data);
    selectedRam();
    setRamDescriptionLength(400);
    let memoryType = '';
    if (processorData?.data) {
        memoryType =
            processorData?.data?.socket === 'AM5' ||
            processorData?.data?.label.split('-')[1].slice(0,2)
=== '13' ? 'DDR5' : 'DDR4'
    };
    dispatch(mainActions.getRams({value: data[0].value,
memoryType: memoryType}));
};

const handleChangeRam = (data) => {
    selectedRam(data.rozetkaId);
    dispatch(mainActions.getPage({id: data.rozetkaId, data:
data, pageType: 'ram'}));
    setRamDescriptionLength(400);
};

```

```

const setPower = (data) => {
  setPowerValue(data);
  dispatch(mainActions.getPowers({value: data[0].value}));
  selectedPower();
  setPowerDescriptionLength(400);
};

const handleChangePower = (data) => {
  selectedPower(data.rozetkaId);
  dispatch(mainActions.getPage({id: data.rozetkaId, data:
data, pageType: 'power'}));
  setPowerDescriptionLength(400);
};

const setHDD = (data) => {
  setHddValue(data);
  dispatch(mainActions.getHDDs({value: data[0].value, isSSD:
false}));
  selectedHdd();
  setHddDescriptionLength(400);
};

const handleChangeHDD = (data) => {
  selectedHdd(data.rozetkaId);
  dispatch(mainActions.getPage({id: data.rozetkaId, data:
data, pageType: 'hdd'}));
  setHddDescriptionLength(400);
};

const setSSD = (data) => {
  setSsdValue(data);
  dispatch(mainActions.getSSDs({value: data[0].value, isSSD:
true}));
  selectedSsd();
  setSsd1DescriptionLength(400);
};

const handleChangeSSD = (data) => {

```

```

        selectedSsd(data.rozetkaId);
        dispatch(mainActions.getPage({id: data.rozetkaId, data:
data, pageType: 'ssd1'}));
        setSsd1DescriptionLength(400);
    };

    const setSSD2 = (data) => {
        setSsd2Value(data);
        dispatch(mainActions.getSSD2s({value: data[0].value,
isSSD: true}));
        selectedSsd2();
        setSsd2DescriptionLength(400);
    };

    const handleChangeSSD2 = (data) => {
        selectedSsd2(data.rozetkaId);
        dispatch(mainActions.getPage({id: data.rozetkaId, data:
data, pageType: 'ssd2'}));
        setSsd2DescriptionLength(400);
    };

    const setCase = (data) => {
        setCaseValue(data);
        selectedCase();
        setCaseDescriptionLength(400);
        dispatch(mainActions.getCases({
            value: data[0].value,
            coolerWidth: coolerData?.data?.width || '',
            videocardWidth: videocardData?.data?.width || '',
        }));
    };

    const handleChangeCase = (data) => {
        selectedCase(data.rozetkaId);
        dispatch(mainActions.getPage({id: data.rozetkaId, data:
data, pageType: 'case'}));
        setCaseDescriptionLength(400);
    };

```

```

const checkFormValidation = () => {
  const rom = hdd || ssd || ssd2
  return processor && cooler && motherboard && pcCase &&
power && ram && rom
};

const notify = () => toast.custom(
  <CustomToast
    title="Ваша збірка успішно додана до корзини"
    id="successCart"
    cart={true}
    theme='success'
  />, { id: 'successCart' }
);

const notifyError = () => toast.custom(
  <CustomToast
    title="Для того щоб додати збірку до корзини, будь
ласка, заповніть усі поля або оберіть опцію (без), де
це можливо"
    id="errorCart"
  />, { id: 'errorCart' }
);

const addToCart = () => {
  localStorage.setItem('cartData', JSON.stringify(
    {
      processor: processorData,
      cooler: coolerData,
      motherboard: motherboardData,
      ram: ramData,
      power: powerData,
      case: caseData,
      videocard: videocardData,
      hdd: hddData,
      ssd1: ssd1Data,
      ssd2: ssd2Data,
    }
  ));
};

```

```

        notify();
    };

    const handleToggleOnlyBestVideocards = (data) => {
        setOnlyBestVideocards(data)
        const maxVideocard_Best =

maxVideocard1660.videocards?.includes(videocardData?.data?.value) ?
maxVideocard1660 :

maxVideocard3050.videocards?.includes(videocardData?.data?.value) ?
maxVideocard3050 :
        maxVideocard3060ti.videocards?.includes(videocardData?
            .data?.value) ? maxVideocard3060ti :
        maxVideocard3070ti.videocards?.includes(videocardData?.dat
            a?.value) ? maxVideocard3070ti :

maxVideocard4070.videocards?.includes(videocardData?.data?.value) ?
maxVideocard4070 :

        maxVideocard4070ti.videocards?.includes(videocardData?
            .data?.value) ? maxVideocard4070ti :

maxVideocard4080.videocards?.includes(videocardData?.data?.value) ?
maxVideocard4080 :

        maxVideocard4090.videocards?.includes(videocardData?.data?.v
            alue) ? maxVideocard4090 : '';

        const maxVideocard_All =

maxVideocard1660_All.videocards?.includes(videocardData?.dat
            a?.value) ? maxVideocard1660_All :
        maxVideocard3050_All.videocards?.includes(videocardData?.d
            ata?.value) ? maxVideocard3050_All :
        maxVideocard3060ti_All.videocards?.includes(videocardData?
            .data?.value) ? maxVideocard3060ti_All :
        maxVideocard3070ti_All.videocards?.includes(videocardData?
            .data?.value) ? maxVideocard3070ti_All :

```



```

<span style={{color: 'black', fontSize:
'15px'}}>Ця опція фільтрує процесори:</span>
<br></br><span style={{color: '#227093'}}>Якщо
не вибрано:</span> всі процесори які можна
використовувати з відеокартою
<br></br><span style={{color: '#227093'}}>Якщо
вибрано:</span> тільки процесори які максимально
сумісні з відеокартою
    </p>
    <br></br>
    <Switch
        title='Тільки сумісні с відеокартою'
        isOn={onlyBestVideocards}
        handleToggle={ (data) =>
handletoggleOnlyBestVideocards(data) }
        id={'onlyBestVideocards'}
    />
</div> : <></>
}
<CustomMultiSelect
    className='select'
    options={processors}
    selected={processorValue}
    setSelected={ (value) => setProcessor(value) }
/>
</div>
{processorValue?.length > 0 ?
processorsData.length > 0 ?
    <div className='tableRadio'>
        {processorsData.map (processorData => (
            <RadioInput
                label={processorData.label}
                value={processorData.rozetkaId}
                id={processorData._id}
                handleChange={ () =>
handleChangeProcessor (processorData) }
                isChecked={processor}
            />
        ) ) }
    </div>
}

```

```

        </div> :
        <div className='subTitle' style={{ maxWidth:
'440px', marginTop: '20px' }}>
        Немає підходящих процесорів під вибрану
відеокарту,

        для цієї відеокарти потрібен один з цих
процесорів ({processorsForVideocard?.join(', ')}
        </div> : <></>
    }
</div>
<div className='tableCollumn'>
    <div className='tableItem'>
        <div className="secondTitle">Охолодження</div>
        <CustomMultiSelect
            className='select'
            options={coolers}
            selected={coolerValue}
            setSelected={(value) => setCooler(value)}
        />
    </div>
    {coolerValue?.length > 0 ? coolersData.length >
0 ?

        <div className='tableRadio'>
            {coolersData.map(coolerData => (
                <RadioInput
                    label={coolerData.label}
                    value={coolerData.rozetkaId}
                    id={coolerData._id}
                    handleChange={() =>
handleChangeCooler(coolerData)}
                    isChecked={cooler}
                />
            ))}
        </div> :
        <div className='subTitle' style={{ maxWidth:
'440px', marginTop: '20px' }}>
        Немає підходящих систем повітряного охолодження
під вибраний процесор (TDP процесора:
{processorData.data.power}W),

```

```

        виберіть водяне охолодження
    </div> : <></>
    }
</div>
<div className='tableCollumn'>
    <div className='tableItem'>
        <div className="secondTitle">Материнська
плата</div>

        <CustomMultiSelect
            className='select'
            options={motherboards}
            selected={motherboardValue}
            setSelected={ (value) =>
setMotherboard(value) }
        />
    </div>
    {motherboardValue?.length > 0 ?
motherboardsData.length > 0 ?
        <div className='tableRadio'>
            {motherboardsData.map (motherboardData => (
                <RadioInput
                    label={motherboardData.label}
                    value={motherboardData.rozetkaId}
                    id={motherboardData._id}
                    handleChange={ () =>
handleChangeMotherboard(motherboardData) }
                    isChecked={motherboard}
                />
            ) ) }
        </div> :
        <div className='subTitle' style={{ maxWidth:
'440px', marginTop: '20px' }}>
            Немає підходящих материнських плат цієї фірми під
            вибраний процесор (чипсет),
            виберіть іншу фірму
        </div> : <></>
    }
</div>
<div className='tableCollumn'>

```

```

<div className='tableItem'>
  <div className="secondTitle">Оперативна
пам'ять</div>

  <CustomMultiSelect
    className='select'
    options={rams}
    selected={ramValue}
    setSelected={(value) => setRam(value)}
  />
</div>
{ramValue?.length > 0 ? ramsData.length > 0 ?
  processorValue?.length > 0 &&
ramValue[0].value === '8'
  && ['i5', 'i7', 'i9', 'ryzen5', 'ryzen7',
'ryzen9'].includes(processorValue[0].value) ?
  <div className='subTitle' style={{ maxWidth:
'440px', marginTop: '20px' }}>
    Для вашої системи мінімальна кількість
оперативної пам'яті дорівнює 16 ГБ
  </div> :
  <div className='tableRadio'>
    {ramsData.map(ramData => (
      <RadioInput
        label={ramData.label}
        value={ramData.rozetkaId}
        id={ramData._id}
        handleChange={() =>
handleChangeRam(ramData)}
        isChecked={ram}
      />
    ))}
  </div> :
  <div className='subTitle' style={{ maxWidth:
'440px', marginTop: '20px' }}>
    Немає оперативної пам'яті цієї ємності яка
відповідає типу DDR5,
    мінімальна кількість оперативної пам'яті з
DDR5 дорівнює 32 ГБ
  </div> : <></>

```

```

    }
  </div>
  <div className='tableCollumn'>
    <div className='tableItem'>
      <div className="secondTitle">Відеокарта</div>
      <CustomMultiSelect
        className='select'
        options={videocards}
        selected={videocardValue}
        setSelected={(value) => setVideocard(value)}
      />
    </div>
    {videocardValue?.length > 0 ?
      <div className='tableRadio'>
        {videocardsData.map(videocardData => (
          <RadioInput
            label={videocardData.label}
            value={videocardData.rozetkaId}
            id={videocardData._id}
            handleChange={() =>
              handleChangeVideocard(videocardData) }
            isChecked={videocard}
          />
        ))}
      </div> : <></>
    }
  </div>
  <div className='tableCollumn'>
    <div className='tableItem'>
      <div className="secondTitle">Блок
        живлення</div>
      <CustomMultiSelect
        className='select'
        options={powers}
        selected={powerValue}
        setSelected={(value) => setPower(value)}
      />
    </div>
    {powerValue?.length > 0 ?

```

```

        powerValue[0].value < totalPower * 1.2 - 10 ?
        <div className='subTitle' style={{ maxWidth:
'440px', marginTop: '20px' }}>
            Виберіть інше значення потужності блоку
            живлення, що відповідає споживанню вашої системи
            {totalPower}W + 20%,
            тобто мінімальна потужність блока живлення
            повинна бути {Math.round(totalPower * 1.2 > 1050
            ? 1200 / 100 : totalPower * 1.2 / 100) * 100}
        </div>
        :
        <div className='tableRadio'>
            {powersData.map(powerData => (
                <RadioInput
                    label={powerData.label}
                    value={powerData.rozetkaId}
                    id={powerData._id}
                    handleChange={() =>
handleChangePower(powerData)}
                    isChecked={power}
                />
            ))}
        </div> : <></>
    }
</div>
<div className='tableCollumn'>
    <div className='tableItem'>
        <div className="secondTitle">Жорсткий
диск</div>

        <CustomMultiSelect
            className='select'
            options={hdds}
            selected={hddValue}
            setSelected={(value) => setHDD(value)}
        />
    </div>
    {hddValue?.length > 0 ?
        <div className='tableRadio'>
            {HDDsData.map(hddData => (

```

```

        <RadioInput
          label={hddData.label}
          value={hddData.rozetkaId}
          id={hddData._id}
          handleChange={() =>
handleChangeHDD(hddData)}
          isChecked={hdd}
        />
      )}
    </div> : <></>
  }
</div>
<div className='tableCollumn'>
  <div className='tableItem'>
    <div className="secondTitle">SSD диск</div>
    <CustomMultiSelect
      className='select'
      options={ssds}
      selected={ssdValue}
      setSelected={(value) => setSSD(value)}
    />
  </div>
  {ssdValue?.length > 0 ?
    <div className='tableRadio'>
      {SSDsData.map(ssdData1 => (
        <RadioInput
          label={ssdData1.label}
          value={ssdData1.rozetkaId}
          id={ssdData1._id}
          handleChange={() =>
handleChangeSSD(ssdData1)}
          isChecked={ssd}
        />
      )}
    </div> : <></>
  }
</div>
<div className='tableCollumn'>
  <div className='tableItem'>

```

```

<div className="secondTitle">SSD диск 2</div>
<CustomMultiSelect
  className='select'
  options={ssds}
  selected={ssd2Value}
  setSelected={ (value) => setSSD2 (value) }
/>
</div>
{ssd2Value?.length > 0 ?
  <div className='tableRadio'>
    {SSD2sData.map(ssd2Data => (
      <RadioInput
        label={ssd2Data.label}
        value={ssd2Data.rozetkaId}
        id={ssd2Data._id + '1'}
        handleChange={ () =>
handleChangeSSD2 (ssd2Data) }
        isChecked={ssd2}
      />
    ))}
  </div> : <></>
}
</div>
<div className='tableCollumn'>
  <div className='tableItem' style={{
marginBottom: '20px' }}>
    <div className="secondTitle">Копируй</div>
    <CustomMultiSelect
      className='select'
      options={cases}
      selected={pcCaseValue}
      setSelected={ (value) => setCase (value) }
    />
  </div>
  {pcCaseValue?.length > 0 ? casesData.length > 0
?
    <div className='tableRadio'>
      {casesData.map(caseData => (
        <RadioInput

```

```

        label={caseData.label}
        value={caseData.rozetkaId}
        id={caseData._id}
        handleChange={() =>
handleChangeCase(caseData)}

        isChecked={pcCase}
      />
    )))
  </div> :
  <div className='subTitle' style={{ maxWidth:
'440px', marginTop: '20px' }}>
    Немає підходящих корпусів цієї фірми які
    відповідають вибраним комплектуючим
  </div> : <></>
}
</div>
</div>
<div className='buildContainer'>
  <div className="secondTitle" style={{ fontSize: 22
}}>Ваша збірка</div>
  <div style={{ marginTop: '20px' }}>
    {processorValue?.length > 0 && processor ?
    <>
      <LineDivider/>
      <div className='accessoriesContainer'>
        <div className='accessoriesImage'>
          <img alt='processor' style={{ maxHeight: '150px'
          }} src={processorData.images ?
processorData?.images[0].base_action.url : ''}/>
        </div>
        <div style={{ marginLeft: '20px' }}>
          <div
className="accessoriesTitle">{processorData?.data?.label}</div>
          <div
className="accessoriesCharacteristics">
            <p className="CharTitle">Тип роз'єму:
</p>
            <p
className="CharSubTitle">{processorData?.data?.socket}</p>

```

```

        </div>
        {processorData?.data?.coreArchitecture ? <div
        className="accessoriesCharacteristics">
            <p className="CharTitle">Архітектура
процесора: </p>
            <p
            className="CharSubTitle">{processorData?.data?.cor
eArchitecture}</p>
            </div> : <></>}
        <div
className="accessoriesCharacteristics">
            <p className="CharTitle">Кількість ядер:
</p>
            <p
className="CharSubTitle">{processorData?.data?.cores}</p>
            </div>
            <div
className="accessoriesCharacteristics">
                <p className="CharTitle">Кількість
потоків: </p>
                <p
className="CharSubTitle">{processorData?.data?.threads}</p>
                </div>
                <div
className="accessoriesCharacteristics">
                    <p className="CharTitle">Інтегрована
графіка: </p>
                    <p className="CharSubTitle">
{processorData?.data?.integratedGraphics ?
processorData?.data.integratedGraphics :
'Немає'}
                    </p>
                    </div>
                    <div
className="accessoriesCharacteristics">
                        <p className="CharTitle">Максимальна
частота процесора: </p>

```

```

        <p
            className="CharSubTitle">{processorData?.data?.maxFrequency} МГц</p>
        </div>
        <div
            className="accessoriesCharacteristics">
                <p className="CharTitle">Обсяг кеш
пам'яті 3 рівня: </p>
                <p
                    className="CharSubTitle">{processorData?.data?.L3cache}</p>
                </div>
                <br></br>
                <div
                    dangerouslySetInnerHTML={{ __html:
`$${processorData?.description?.text.slice(0,
processorDescriptionLength)}
    ${processorData?.description?.text.length >=
600 && processorDescriptionLength === 600 ?
    '...' : '' }` }}
                    style={{fontFamily: 'RobotoRegular', color:
'#9aa1b8', maxWidth: '800px' }}
                    />
                    {processorData?.description?.text.length
>= 600 ?

                    <button className='accessoriesButton'
                        onClick={() =>
                            setProcessorDescriptionLength(processorDescriptionLength === 600 ? 10000 : 600)}>
                        {processorDescriptionLength === 600 ?
'розгорнути' : 'згорнути' }
                    </button>
                    : <></>
                }
                <br></br>
                <div className="accessoriesPrice">
                    <p
                        className="accessoriesTitle">Ціна:</p>
                    <p
                        className="accessoriesPriceTitle">{processorData?.price}&</p>

```

```

        </div>
    </div>
</div>
</> : <></>
}
{coolerValue?.length > 0 && cooler ?
<>
    <LineDivider/>
    <div className='accessoriesContainer'>
        <div className='accessoriesImage'>
            <img alt='cooler' style={{ maxHeight: '150px' }}
            src={coolerData.images ?
            coolerData?.images[0].base_action.url : ''}/>
        </div>
        <div style={{ marginLeft: '20px' }}>
            <div
className="accessoriesTitle">{coolerData?.data?.label}</div>
                <br></br>
                <div
className="accessoriesCharacteristics">
                    <p className="CharTitle">Максимальный
TDP: </p>
                        <p
className="CharSubTitle">{coolerData?.data?.TDP}</p>
                            </div>
                            <br></br>
                            <div
                                dangerouslySetInnerHTML={{ __html:
                                `${coolerData?.description?.text.slice(0,
                                coolerDescriptionLength) }
                                ${coolerData?.description?.text.length >= 600
                                && coolerDescriptionLength === 400 ? '...' :
                                '' }` }}
                                style={{ fontFamily: 'RobotoRegular', color:
                                '#9aa1b8', maxWidth: '800px' }}
                                />
                                {coolerData?.description?.text.length >=
600 ?

```

```

<button className='accessoriesButton' onClick={()
=>
  setCoolerDescriptionLength(coolerDescriptionLengt
h === 400 ? 10000 : 400)}>
    {coolerDescriptionLength === 400 ?
'розгорнути' : 'згорнути' }
    </button> : <></>
  }
  <br></br>
  <div className="accessoriesPrice">
    <p
className="accessoriesTitle">Ціна:</p>
    <p
className="accessoriesPriceTitle">{processorData?.price}&</p>
    </div>
  </div>
</div>
</> : <></>
}
{motherboardValue?.length > 0 && motherboard ?
<>
  <LineDivider/>
  <div className='accessoriesContainer'>
    <div className='accessoriesImage'>
      <img alt='motherboard' style={{ maxHeight:
'150px' }} src={motherboardData.images ?
motherboardData?.images[0].base_action.url :
''}/>
    </div>
    <div style={{ marginLeft: '20px' }}>
      <div
className="accessoriesTitle">{motherboardData?.data?.label}</div>
      <br></br>
      <div
className="accessoriesCharacteristics">
        <p className="CharTitle">Сокет: </p>
        <p
className="CharSubTitle">{motherboardData?.data?.socket}</p>
      </div>

```

```

        <div
className="accessoriesCharacteristics">
            <p className="CharTitle">Чипсет: </p>
            <p
className="CharSubTitle">{motherboardData?.data?.chipset}</p>
            </div>
            <br></br>
            <div
                dangerouslySetInnerHTML={{ __html:
                `${motherboardData?.description?.text.slice(0,
                motherboardDescriptionLength)}
                ${motherboardData?.description?.text.length >=
                600 && motherboardDescriptionLength === 600 ?
                '...' : '' }` }}
                style={{fontFamily: 'RobotoRegular', color:
                '#9aa1b8', maxWidth: '800px' }}
                />
                {motherboardData?.description?.text.length
                >= 600 ?
                <button className='accessoriesButton'
                onClick={() =>
                setMotherboardDescriptionLength(motherboardD
                escriptionLength === 600 ? 10000 : 600)}>
                    {motherboardDescriptionLength === 600
                ? 'розгорнути' : 'згорнути' }
                </button> : <></>
                }
                <br></br>
                <div className="accessoriesPrice">
                    <p
className="accessoriesTitle">Ціна:</p>
                    <p
className="accessoriesPriceTitle">{motherboardData?.price}</p>
                </div>
                </div>
            </div>
        </> : <></>
    }
    {ramValue?.length > 0 && ram ?

```

```

<>
    <LineDivider/>
    <div className='accessoriesContainer'>
        <div className='accessoriesImage'>
            <img alt='ram' style={{ maxHeight: '150px' }}
            src={ramData.images ?
            ramData?.images[0].base_action.url : ''}/>
        </div>
        <div style={{ marginLeft: '20px' }}>
            <div
className="accessoriesTitle">{ramData?.data?.label}</div>
                <br></br>
                <div
className="accessoriesCharacteristics">
                    <p className="CharTitle">Обсяг пам'яті:
</p>
                        <p
className="CharSubTitle">{ramData?.data?.value} ГБ</p>
                    </div>
                    <div
className="accessoriesCharacteristics">
                        <p className="CharTitle">Тип пам'яті:
</p>
                            <p
className="CharSubTitle">{ramData?.data?.memoryType}</p>
                        </div>
                        <div
className="accessoriesCharacteristics">
                            <p className="CharTitle">Частота
пам'яті: </p>
                                <p
className="CharSubTitle">{ramData?.data?.memoryFrequency}</p>
                            </div>
                            <div
className="accessoriesCharacteristics">
                                <p className="CharTitle">Кількість
планок: </p>
                                    <p className="CharSubTitle">2</p>
                                </div>

```

```

        <br></br>
        <div
            dangerouslySetInnerHTML={{ __html:
                `${ramData?.description?.text.slice(0,
                    ramDescriptionLength)}
                ${ramData?.description?.text.length >= 400 &&
                    ramDescriptionLength === 400 ? '...' : ''}`
            }}
            style={{fontFamily: 'RobotoRegular', color:
                '#9aa1b8', maxWidth: '800px' }}
        />
        {ramData?.description?.text.length >= 400
?
        <button className='accessoriesButton'
            onClick={() =>
                setRamDescriptionLength(ramDescriptionLength
                    === 400 ? 10000 : 400)}>
            {ramDescriptionLength === 400 ?
'розгорнути' : 'згорнути' }
            </button> : <></>
        }
        <br></br>
        <div className="accessoriesPrice">
            <p
className="accessoriesTitle">Ціна:</p>
            <p
className="accessoriesPriceTitle">{ramData?.price}€</p>
            </div>
        </div>
    </div>
</> : <></>
}
{videocardValue?.length > 0 && videocard ?
<>
    <LineDivider/>
    <div className='accessoriesContainer'>
        <div className='accessoriesImage'>

```

```

<img alt='videocard' style={{ maxHeight: '150px'
}} src={videocardData.images ?
videocardData?.images[0].base_action.url : ''}/>
</div>
<div style={{ marginLeft: '20px' }}>
  <div
className="accessoriesTitle">{videocardData?.data?.label}</div>
    <br></br>
    <div
className="accessoriesCharacteristics">
      <p className="CharTitle">Розрядність
шини пам'яті: </p>
      <p
className="CharSubTitle">{videocardData?.data?.memoryBusWidth}</p>
    </div>
    <div
className="accessoriesCharacteristics">
      <p className="CharTitle">Частота
пам'яті: </p>
      <p
className="CharSubTitle">{videocardData?.data?.memoryFrequency}</p>
    </div>
    <div
className="accessoriesCharacteristics">
      <p className="CharTitle">Частота ядра:
</p>
      <p
className="CharSubTitle">{videocardData?.data?.coreFrequency}</p>
    </div>
    <br></br>
    <div
      dangerouslySetInnerHTML={{ __html:
`${videocardData?.description?.text.slice(0,
videocardDescriptionLength)}
${videocardData?.description?.text.length >=
600 && videocardDescriptionLength === 600 ?
'...' : '' }` }}
      style={{ fontFamily: 'RobotoRegular', color:
'#9a1b8', maxWidth: '800px' }}

```

```

        />
        {videocardData?.description?.text.length
    >= 600 ?

    <button className='accessoriesButton'
    onClick={() =>
    setVideocardDescriptionLength(videocardDescr
    iptionLength === 600 ? 10000 : 600)}>
        {videocardDescriptionLength === 600 ?
    'розгорнути' : 'згорнути' }

        </button> : <></>
    }
    <br></br>
    <div className="accessoriesPrice">
        <p
    className="accessoriesTitle">Ціна:</p>
        <p
    className="accessoriesPriceTitle">{videocardData?.price}&</p>
        </div>
    </div>
    </div>
    </> : <></>
    }
    {powerValue?.length > 0 && power ?
    <>
        <LineDivider/>
        <div className='accessoriesContainer'>
            <div className='accessoriesImage'>
                <img alt='power' style={{ maxHeight: '150px' }}
                src={powerData.images ?
                powerData?.images[0].base_action.url : ''}/>
            </div>
            <div style={{ marginLeft: '20px' }}>
                <div
    className="accessoriesTitle">{powerData?.data?.label}</div>
                <br></br>
                <div
    className="accessoriesCharacteristics">
                    <p className="CharTitle">Потужність:
    </p>

```

```

        <p
className="CharSubTitle">{powerData?.data?.value} W</p>
        </div>
        <br></br>
        <div
            dangerouslySetInnerHTML={{ __html:
                `${powerData?.description?.text.slice(0,
                    powerDescriptionLength)}
                ${powerData?.description?.text.length >= 400
                    && powerDescriptionLength === 400 ? '...' : ''
                }` }}
            style={{fontFamily: 'RobotoRegular', color:
                '#9a1b8', maxWidth: '800px' }}
        />
        {powerData?.description?.text.length >=
400 ?

        <button className='accessoriesButton'
            onClick={() =>
                setPowerDescriptionLength(powerDescriptionLe
                    ngth === 400 ? 10000 : 400)}>
            {powerDescriptionLength === 400 ?
'розгорнути' : 'згорнути' }

            </button> : <></>
        }
        <br></br>
        <div className="accessoriesPrice">
            <p
className="accessoriesTitle">Ціна:</p>
            <p
className="accessoriesPriceTitle">{powerData?.price}€</p>
            </div>
        </div>
    </div>
</> : <></>
}
{hddValue?.length > 0 && hdd ?
<>
    <LineDivider/>
    <div className='accessoriesContainer'>

```

```

        <div className='accessoriesImage'>
        <img alt='hdd' style={{ maxHeight: '150px' }}
        src={hddData.images ?
        hddData?.images[0].base_action.url : ''}/>
        </div>
        <div style={{ marginLeft: '20px' }}>
        <div
className="accessoriesTitle">{hddData?.data?.label}</div>
        <br></br>
        <div
className="accessoriesCharacteristics">
                <p className="CharTitle">Обсяг пам'яті:
</p>

                <p
className="CharSubTitle">{hddData?.data?.value} ГБ</p>
                </div>
                <div
className="accessoriesCharacteristics">
                        <p className="CharTitle">Швидкість
передавання даних: </p>

                        <p
className="CharSubTitle">{hddData?.data?.speed}</p>
                        </div>
                        <div
className="accessoriesCharacteristics">
                                <p className="CharTitle">Швидкість
обертання шпинделя: </p>

                                <p
className="CharSubTitle">{hddData?.data?.spindleRotationSpeed}</p>
                                </div>
                                <br></br>
                                <div
dangerouslySetInnerHTML={{ __html:
` ${hddData?.description?.text.slice(0,
hddDescriptionLength)}
${hddData?.description?.text.length >= 400 &&
hddDescriptionLength === 400 ? '...' : '' } `
                                }}

```

```

style={{fontFamily: 'RobotoRegular', color:
'#9aa1b8', maxWidth: '800px' }}
    />
    {hddData?.description?.text.length >= 400
?

    <button className='accessoriesButton'
    onClick={() =>
    setHddDescriptionLength(hddDescriptionLength
=== 400 ? 10000 : 400)}>
        {hddDescriptionLength === 400 ?
'розгорнути' : 'згорнути' }

        </button> : <></>
    }
    <br></br>
    <div className="accessoriesPrice">
        <p
className="accessoriesTitle">Ціна:</p>
        <p
className="accessoriesPriceTitle">{hddData?.price}&</p>
        </div>
    </div>
</div>
</> : <></>
}
{ssdValue?.length > 0 && ssd ?
<>
    <LineDivider/>
    <div className='accessoriesContainer'>
        <div className='accessoriesImage'>
            <img alt='ssd1' style={{ maxHeight: '150px' }}
            src={ssd1Data.images ?
            ssd1Data?.images[0].base_action.url : ''}/>
        </div>
        <div style={{ marginLeft: '20px' }}>
            <div
className="accessoriesTitle">{ssd1Data?.data?.label}</div>
            <br></br>
            <div
className="accessoriesCharacteristics">

```

```

        <p className="CharTitle">Форм-фактор:
    </p>

        <p
            className="CharSubTitle">{ssdlData?.data?.isM2 ?
            'M.2' : '2.5'}</p>
        </div>
        <div
            className="accessoriesCharacteristics">
            <p className="CharTitle">Інтерфейс
підключення: </p>

            <p
                className="CharSubTitle">{ssdlData?.data?.isM2 ?
                'PCI Express 3.0 x4' : 'SATAIII'}</p>
            </div>
            <div
                className="accessoriesCharacteristics">
                <p className="CharTitle">Обсяг пам'яті:
    </p>

                <p
                    className="CharSubTitle">{ssdlData?.data?.value} ГБ</p>
                </div>
                <div
                    className="accessoriesCharacteristics">
                    <p className="CharTitle">Максимальна
швидкість запису: </p>

                    <p
                        className="CharSubTitle">{ssdlData?.data?.writeSpeed}</p>
                    </div>
                    <div
                        className="accessoriesCharacteristics">
                        <p className="CharTitle">Максимальна
швидкість зчитування: </p>

                        <p
                            className="CharSubTitle">{ssdlData?.data?.readSpeed}</p>
                        </div>
                        <br></br>
                    </div>

```

```

        dangerouslySetInnerHTML={{ __html:
        `${ssd1Data?.description?.text.slice(0,
        ssd1DescriptionLength)}
        ${ssd1Data?.description?.text.length >= 400
        && ssd1DescriptionLength === 400 ? '...' :
        '' }` }}
        style={{fontFamily: 'RobotoRegular', color:
        '#9aa1b8', maxWidth: '800px' }}
        />
        {ssd1Data?.description?.text.length >= 400
?

        <button className='accessoriesButton'
        onClick={() =>
        setSsd1DescriptionLength(ssd1DescriptionLeng
        th === 400 ? 10000 : 400)}>
                {ssd1DescriptionLength === 400 ?
'розгорнути' : 'згорнути' }

                </button> : <></>
        }
        <br></br>
        <div className="accessoriesPrice">
                <p
className="accessoriesTitle">Ціна:</p>
                <p
className="accessoriesPriceTitle">{ssd1Data?.price}€</p>
                </div>
        </div>
        </div>
        </> : <></>
        }
        {ssd2Value?.length > 0 && ssd2 ?
        <>
        <LineDivider/>
        <div className='accessoriesContainer'>
                <div className='accessoriesImage'>
        <img alt='ssd1' style={{ maxHeight: '150px' }}
        src={ssd2Data.images ?
        ssd2Data?.images[0].base_action.url : ''}/>
                </div>

```

```

        <div style={{ marginLeft: '20px' }}>
            <div
className="accessoriesTitle">{ssd2Data?.data?.label}</div>
            <br></br>
            <div
className="accessoriesCharacteristics">
                <p className="CharTitle">Форм-фактор:
</p>

                <p
                    className="CharSubTitle">{ssd2Data?.data?.isM2 ?
'M.2' : '2.5'}</p>
                </div>
                <div
className="accessoriesCharacteristics">
                    <p className="CharTitle">Інтерфейс
підключення: </p>

                    <p
                        className="CharSubTitle">{ssd2Data?.data?.isM2 ?
'PCI Express 3.0' : 'SATAIII'}</p>
                    </div>
                    <div
className="accessoriesCharacteristics">
                        <p className="CharTitle">Обсяг пам'яті:
</p>

                        <p
                            className="CharSubTitle">{ssd2Data?.data?.value} ГБ</p>
                        </div>
                        <div
className="accessoriesCharacteristics">
                            <p className="CharTitle">Максимальна
швидкість запису: </p>

                            <p
                                className="CharSubTitle">{ssd2Data?.data?.writeSpeed}</p>
                            </div>
                            <div
className="accessoriesCharacteristics">
                                <p className="CharTitle">Максимальна
швидкість зчитування: </p>

```

```

        <p
className="CharSubTitle">{ssd2Data?.data?.readSpeed}</p>
    </div>
    <br></br>
    <div
        dangerouslySetInnerHTML={{ __html:
`$${ssd2Data?.description?.text.slice(0,
ssd2DescriptionLength)}
    ${ssd2Data?.description?.text.length >= 400 &&
ssd2DescriptionLength === 400 ? '...' : '' }`
    }}
        style={{fontFamily: 'RobotoRegular', color:
'#9aa1b8', maxWidth: '800px' }}
    />
        {ssd2Data?.description?.text.length >= 400
?
        <button className='accessoriesButton'
onClick={() =>
setSsd2DescriptionLength(ssd2DescriptionLeng
th === 400 ? 10000 : 400)}>
            {ssd2DescriptionLength === 400 ?
'розгорнути' : 'згорнути' }
        </button> : <></>
    }
    <br></br>
    <div className="accessoriesPrice">
        <p
className="accessoriesTitle">Ціна:</p>
        <p
className="accessoriesPriceTitle">{ssd2Data?.price}&</p>
    </div>
    </div>
</div>
</> : <></>
}
{pcCaseValue?.length > 0 && pcCase ?
<>
    <LineDivider/>
    <div className='accessoriesContainer'>

```

```

        <div className='accessoriesImage'>
        <img alt='ssd1' style={{ maxHeight: '150px' }}
        src={caseData.images ?
        caseData?.images[0].base_action.url : ''}/>
        </div>
        <div style={{ marginLeft: '20px' }}>
        <div
className="accessoriesTitle">{caseData?.data?.label}</div>
        <br></br>
        <div
className="accessoriesCharacteristics">
            <p className="CharTitle">Форм-фактор:
</p>

            <p
className="CharSubTitle">{caseData?.data?.type}</p>
            </div>
            <div
className="accessoriesCharacteristics">
                <p className="CharTitle">Кількість
вентиляторів: </p>

                <p
className="CharSubTitle">{caseData?.data?.coolers}</p>
                </div>
                <br></br>
                <div
                    dangerouslySetInnerHTML={{ __html:
                    `${caseData?.description?.text.slice(0,
                    caseDescriptionLength)}
                    ${caseData?.description?.text.length >= 400
                    && caseDescriptionLength === 400 ? '...'
                    : '' }` }}
                    style={{fontFamily: 'RobotoRegular', color:
                    '#9a1b8', maxWidth: '800px' }}
                />
                {caseData?.description?.text.length >= 400
?

                <button className='accessoriesButton'
                onClick={() =>

```

```

        setCaseDescriptionLength(caseDescriptionLength
        === 400 ? 10000 : 400)}>
            {caseDescriptionLength === 400 ?
'розгорнути' : 'згорнути' }

            </button> : <></>
        }
        <br></br>
        <div className="accessoriesPrice">
            <p
className="accessoriesTitle">Ціна:</p>
            <p
className="accessoriesPriceTitle">{caseData?.price}&</p>
            </div>
        </div>
        <div>
            <LineDivider/>
        </div> : <></>
    }
</div>
</div>
<br></br>
<div style={{ display: 'flex', justifyContent: 'flex-
end' }} >
    <CustomButton
        onClick={checkFormValidation() ? () => addToCart()
: () => notifyError()}
        title='Додати вашу збірку до кошика'
        style={checkFormValidation() ? { width: '240px',
opacity: '1' } : { width: '240px', opacity: '0.5' }}
    />
</div>
<br></br>
<Toaster/>
</div>
)
}

export default MainPage;

```

ДОДАТОК Б
Слайди презентації

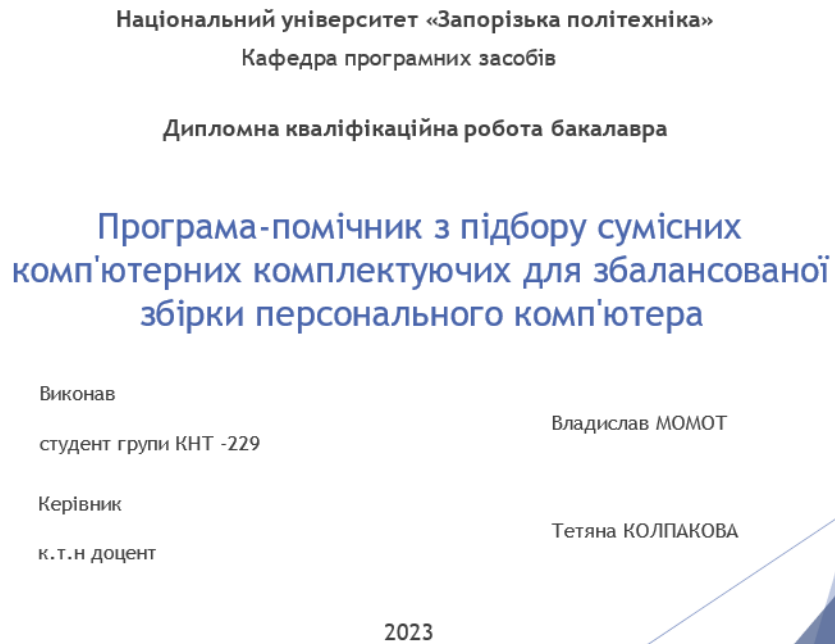


Рисунок Б.1 – Слайд 1

2

- ▶ **Метароботи** – розробка зручного та ефективного вебзастосунок для автоматизації процесу підбору сумісних комп'ютерних комплектуючих для скорочення часу, витраченого користувачем під час збірки ПК.
- ▶ **Предмет дослідження** – вебзастосунок для аналізу сумісності комплектуючих на основі специфіки їх характеристик.
- ▶ **Об'єкт дослідження** – процес підбору комплектації при збірці ПК з урахуванням сумісності комплектуючих для потреб користувача.

Рисунок Б.2 – Слайд 2

3

Завдання роботи

- дослідити принципи створення веборієнтованих систем;
- дослідити види комп'ютерних комплектуючих;
- дослідити API інтернет-магазину “Rozetka” та систематизувати отримані дані щодо наявності та популярності;
- спроектувати базу даних та архітектуру програми;
- визначити особливості характеристик комплектуючих;
- обрати засіб аналізу сумісності комплектуючих;
- програмно реалізувати серверну частину програми;
- програмно реалізувати зовнішній інтерфейс програми.

Рисунок Б.3 – Слайд 3

4

Аналіз предметної області

Критерій	CAN.ua	PCPartPicker	Newegg
Перевірка сумісності	так	так	так
Багатофункціональний	ні	так	ні
Перевантажений інтерфейс	ні	так	так
Тільки свої товари	так	ні	так
Потрібна реєстрація	ні	так	ні
Можливість відправити збірку на електрону пошту	ні	так	так
Підказки для збалансованої збірки	ні	ні	ні

Рисунок Б.4 – Слайд 4

5

Схема структури програми



Вхідні дані

Назва запити	Вхідні параметри
api/getprocessors	value, processorsLabels
api/getcoolers	value, socket, TDP
api/getmotherboards	value, socket, chipset
api/getrams/	value, memoryType
api/getvideocards	value
api/getpowers	value
api/gettrons/	isSSD, value
api/getcases	value, videocardWidth, coolerWidth
api/getpage/	rozetkaid

Рисунок Б.5 – Слайд 5

6

Схема функціонування програми

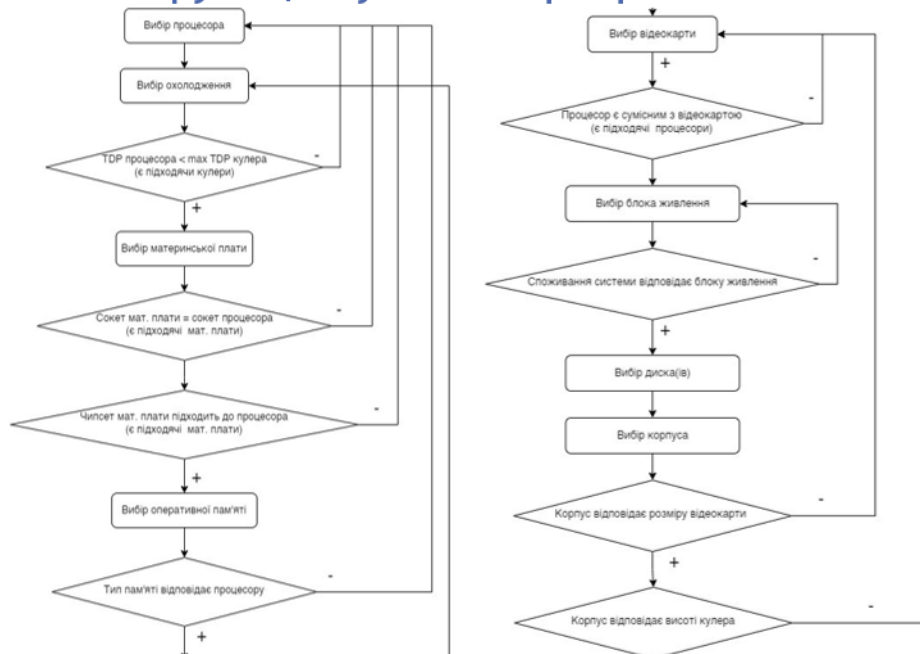


Рисунок Б.6 – Слайд 6

7 Схема даних

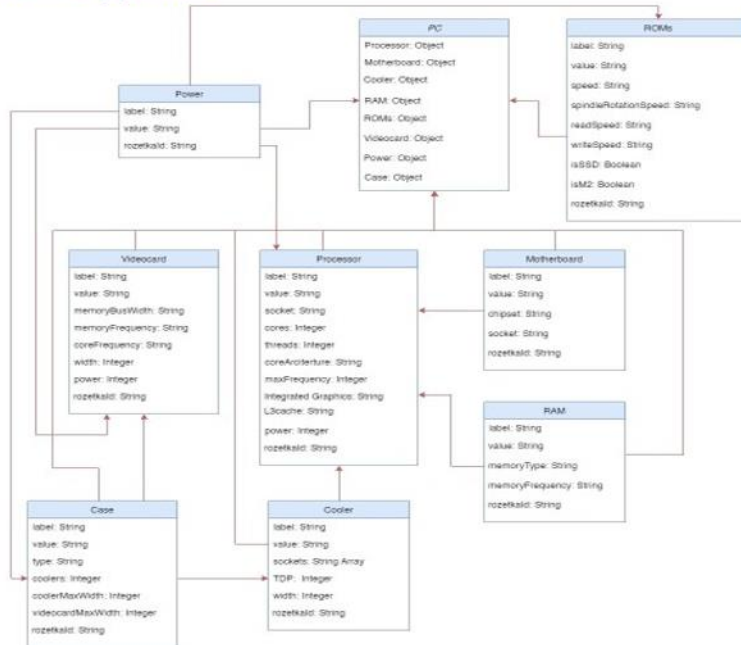


Рисунок Б.7 – Слайд 7

8

Вибір мови програмування

Критерій	Python	PHP	JavaScript
Функціональна	так	ні	так
Спектр застосувань	AI, web, mobile apps, big data	web, server side	web, frontend, backend
Виконує програми на стороні клієнта	ні	ні	так
Виконується в браузері	ні	ні	так
Швидкість	низька	висока	висока
Затребуваність	2 місце	7 місце	1 місце
Підтримка спільноти	Має велику підтримку спільноти	Має велику підтримку спільноти	Має велику підтримку, існує безліч фреймворків та бібліотек

Рисунок Б.8 – Слайд 8

9

Вибір методів (фреймворк і база даних)

Критерій	React	Angular	Vue	Критерій	Mongo DB	MySQL
Рік випуску	2013	2010	2014	Схема	Не вимагає визначення схеми	Вимагає визначення схеми
Мова	JavaScript	TypeScript	JavaScript	Структура даних	не реляційна	реляційна
Підтримка спільноти	Висока	Висока	Низька	Маштабованість	По вертикалі та горизонталі	Тільки по вертикалі
Легкість	Висока	Низька	Висока	Гібкість	Висока	Середня
Гнучкість	Середня	Низька	Висока	Обробка неструктурованих даних	Присутня	Відсутня
Критерій	React	Angular	Vue	Безпека	Менш уразлива до кібератак	Більш уразлива до кібератак
				Підтримка спільноти	Середня	Висока

Рисунок Б.9 – Слайд 9

10

Інтерфейс програми

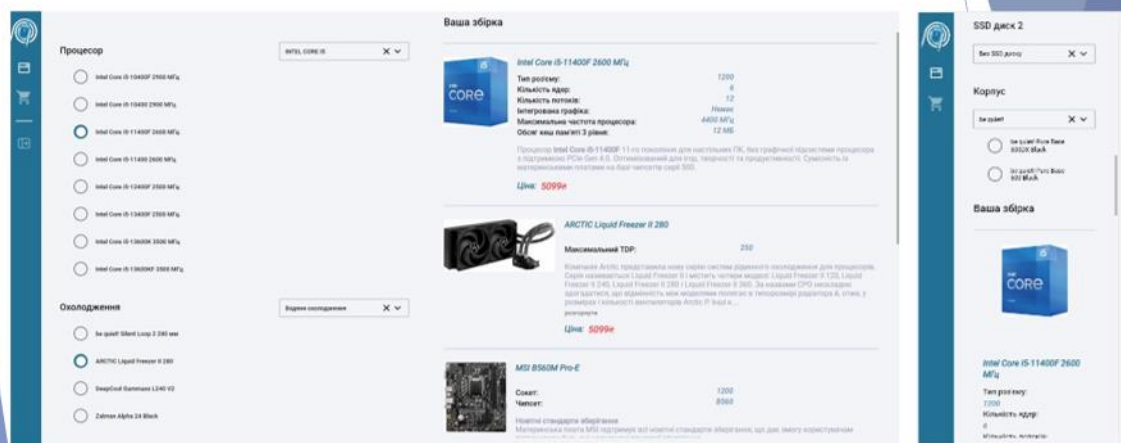


Рисунок Б.10 – Слайд 10

11

Приклади застосування програми

Конфігуратор ПК
Помічник в підборі комплектуючих

Процесор
Ця опція фільтрує процесори:
Якщо не вибрано: всі процесори які можна використувати в апаратному.
Якщо обрано: тільки процесори які максимально сумісні з апаратною.

☒ Тільки сумісні з відеокартою

Новий підбірка процесорів під вибрану відеокарту, для цит. відеокарти потрібен один з цих процесорів (AMD Ryzen 5 2600X 4700 MT/s, AMD Ryzen 7 5800X 3800 MT/s, Intel Core i5-13600K 3500 MT/s, Intel Core i5-13600KF 3500 MT/s, Intel Core i7-13700K 3400 MT/s, Intel Core i7-13700KF 3400 MT/s)

Охолодження
Новий підбірка систем водяного охолодження під вибраний процесор (TDP процесор: 270W), вибрати водне охолодження

Материнська плата
☐ ASUS PRIME Z790-P ATX

Корзина
Ваша збірка, створена за допомогою конфігуратора:

Тип	Назва	Ціна
Процесор	Intel Core i5-14600 (24M 90T)	53700
Охолодження	BeastCool Силикон 120M V2	91800
Материнська плата	MSI MAG Z790M-A Pro	22700
Оперативна пам'ять	Kingston Fury DDR5 20GB 16GB	17800
Дисковод	MSI RTX 1650 4GB (MSI)	12400
Блок живлення	SteelSeries Apex 650W 80PLUS 80W 600W	13800
Жорсткий диск	Seagate 2TB 7200rpm 128MB	41200
SSD диск	Seagate 1TB 100TB 3.5 16000	6000
SSD диск 2	Seagate 4TB 100TB 3.5 16000	18700
Корпус	BeastCool Silhouette 30 Black ATX	10700

Ваша збірка, створена за допомогою конфігуратора: **319720**

Конфігуратор по: <https://github.com/sergey-shesterikov/pc-builder>

Процесор: <https://github.com/sergey-shesterikov/pc-builder/blob/main/processors/processors.json>

Охолодження: <https://github.com/sergey-shesterikov/pc-builder/blob/main/cooling/cooling.json>

Материнська плата: <https://github.com/sergey-shesterikov/pc-builder/blob/main/motherboards/motherboards.json>

Відеокарта: <https://github.com/sergey-shesterikov/pc-builder/blob/main/gpus/gpus.json>

Блок живлення: <https://github.com/sergey-shesterikov/pc-builder/blob/main/power/power.json>

Жорсткий диск: <https://github.com/sergey-shesterikov/pc-builder/blob/main/hdd/hdd.json>

SSD диск: <https://github.com/sergey-shesterikov/pc-builder/blob/main/ssd/ssd.json>

SSD диск 2: <https://github.com/sergey-shesterikov/pc-builder/blob/main/ssd/ssd2.json>

Корпус: <https://github.com/sergey-shesterikov/pc-builder/blob/main/cases/cases.json>

Рисунок Б.11 – Слайд 11

12

Висновки

- В ході виконання даної дипломної роботи вирішено актуальну задачу розроблення автоматичного процесу підбору сумісних комп'ютерних комплектуючих з метою скорочення часу користувача під час збалансованої збірки ПК.
- Даний вебзастосунок створено з використанням бази даних MongoDB, мови програмування JavaScript, які були обрані як найкращі після проведеного аналізу та порівняння з іншими існуючими на даний момент базами даних та мов програмування. Проаналізувавши найпопулярніші бібліотеки та фреймворки для створення клієнтської частини веб-додатків було обрано бібліотеку React. Для розробки серверної частини веб-застосунку було використано Node.js.
- Практична цінність даного програмного забезпечення полягає в тому, що воно надає можливість покращити інформаційний обмін між інтернет-магазинами та користувачами. Галуззю використання є комп'ютерне ПО для домашнього використання або інтернет-магазини, які реалізують комп'ютерні комплектуючі.
- Даний проект в майбутньому можливо модернізувати, додавши окремо візуальне відображення адмін-панелі для створення актуалізації вебзастосунку з можливістю оновлення інформації шляхом додавання редагування комплектуючих.

12

Рисунок Б.12 – Слайд 12