ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розроблення веб-додатку соціальної мережі за допомогою React.js з інтелектуальною системою розпізнавання зображень»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
освітньо-професійної програми Штучний інтелект

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Карина ЛЕБЕДИНЧЕНКО

Виконала:
здобувачка вищої освіти Карина ЛЕБЕДИНЧЕНКО
група ШІД-41

Керівник: Максим ФЕСЕНКО
к.т.н., доцент

Рецензент:
_____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту

_____ Ольга ЗІНЧЕНКО
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Лебединченко Карині Олексіївні

1. Тема кваліфікаційної роботи: Розроблення веб-додатку соціальної мережі за допомогою React.js з інтелектуальною системою розпізнавання зображень

керівник кваліфікаційної роботи Максим ФЕСЕНКО к.т.н., доцент,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» 02.2024р. № 36

2. Строк подання кваліфікаційної роботи «31» травня 2024р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, вимоги до розробки веб-застосунків.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження принципів розробки веб-технологій
Аналіз технологій створення веб-застосунків
5. Перелік графічного матеріалу: *презентація*
1. Постановка задачі
2. Аналіз сучасних веб-технологій

3. Аналіз проблеми
4. Демонстрація готового проекту

6. Дата видачі завдання «27» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	27.02-05.03.24	Виконано
2	Вивчення та аналіз поточної ситуації наявної проблеми та сучасних веб-технологій	06.03-30.03.24	Виконано
3	Розробка веб-застосунку	01.04-25.04.24	Виконано
4	Оформлення роботи: вступ, висновки, реферат	20.11-25.11.23	Виконано
5	Опис кваліфікаційної роботи	25.04-05.05.24	Виконано
6	Розробка демонстраційних матеріалів	05.12-12.05.24	Виконано

Здобувачка вищої освіти _____ Карина ЛЕБЕДИНЧЕНКО

Керівник
кваліфікаційної роботи _____ Максим ФЕСЕНКО

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня, бакалавра: 53 стор., 44 рис., 17 джерел.

Мета роботи – підвищення ефективності інформування та обізнаності людей в питаннях утримання домашніх улюбленців.

Об'єкт дослідження – процес набуття корисних знань без суворих та нав'язливих методів навчання.

Предмет дослідження – система соціальної мережі.

Короткий зміст роботи: У роботі проведено детальний опис всіх веб-технологій для розробки веб-застосунків. Було проведено дослідження та тестування технологій для ретельного вибору технологій під розробку проекту кваліфікаційної роботи.

Проведено аналіз статистики використання конструкторів CMS для розробки веб-сайтів та порівняння доцільності їх використання порівняно з фреймворком React.js, а також технологіями HTML, CSS, JavaScript.

В ході розробки використано фреймворк React.js, бібліотеку для зберігання стану додатку Redux, мову розмітки сторінок HTML, каскадні таблиці стилей CSS, мову програмування для функціоналу JavaScript та середовище розробки VS Code.

КЛЮЧОВІ СЛОВА: FRONTEND, РОЗРОБКА, ВЕБ-ДОДАТОК, ВЕБ-ЗАСТОСУНОК, REACT.JS, HTML, CSS, JAVASCRIPT, ФРЕЙМВОРК.

ЗМІСТ

ВСТУП

Наразі сучасна сучасні технології дають потужний поштовх до швидкого розвитку IT-сфери та її урізноманітнення. IT-сфера поділилась на багато галузей і в порівнянні з попередніми роками вони набирають обертів. Доступність стала ключовою ознакою для такого розвитку. Якщо ще 15 років тому потреби у розробці та організації серверної (backend) і візуальної (frontend) частин міг задовольнити й один фахівець, то зараз як на розробку backend, так і у frontend-розробці є потреба в залученні декількох спеціалістів різних рівнів на кожну з частин проекту.

Світ змінюється, а отже, з'являються нові методи вирішення актуальних проблем та задач у сфері розробки. Зараз є незлічена кількість додаткових бібліотек, матеріалів та фреймворків для реалізації будь-яких задач у сфері розробки. Безперечно, є ті, що набули більшої популярності та розповсюдженості у світі. Серед таких відзначається фреймворк React.js, який набув більшої популярності серед Європи та Америки.

У сучасному світі велику роль відіграє час. Отже, перед компаніями, що надають послуги з розроблення будь-якого продукту, постає питання в розробленні якісного продукту за найшвидший термін. Ще декілька років тому перед розробниками постала задача створити продукт, який буде відповідати сучасним вимогам, з урахуванням потреб та вподобань користувачів, але при цьому продукт має включати в себе швидкісне завантаження контенту, гарну оптимізацію та обширний функціонал. І все це повинно бути реалізоване якомога швидше. Задача перед розробниками (на той момент) стояла складна, адже можливості мов програмування та технологій були досить обмеженими, а вимоги замовників ставили все більш вибагливими.

В наш час жоден бізнес не обходиться без сучасного веб-застосунку, адже це значно розширює ринок споживачів для компанії та підвищує якість бізнес-процесів, розширює горизонти, а також безпосередньо впливає на його процвітання. Тому для комерції вкрай важливо мати сучасний веб-додаток зі своєї продукцією. По-перше, це підвищує продажі. По-друге, веб-застосунок, надає

споживачам доступність до товару. Та, по-третє, це безпосередньо впливає на розповсюдження підприємства, адже завдяки доступності в кожного споживача є можливість отримати товар/продукт/послугу та ознайомитись з умовами придбання в будь-який час, незалежно від статусу, місцезнаходження тощо. І це стосується різних типів бізнесу: роздрібний, оптовий, з надання послуг, виробничий, технологічний, франчайзинг тощо. Завдяки інтернету навіть з'явилась можливість побудувати онлайн бізнес, що надає змогу започаткувати свою справу без зайвих витрат на оренду приміщення, ремонт, обладнання та працівників.

В підсумку зрозуміло, що розробка веб-додатків має великий вплив на весь світ та сприяє процвітанню комерції в наш час, завдяки автоматизації більшості бізнес-процесів, їх оптимізації, розширенні ринку споживачів, підвищенні пізнаванності та доступності. Особливо в умовах війни в України, веб-додатки мають не аби який вплив на вирішення багатьох питань, такі як: збори на потреби армії чи гуманітарну допомогу, привернення уваги світу до проблем в країні, підтримання комерції та виробництв, що вкрай важливо для підтримання економіки, доступність до подій та інформації, а також тих чи інших товарів для будь-яких потреб.

Ціль проекту кваліфікаційної роботи полягає в тому, щоб створити веб-застосунок соціальної мережі для тварин. Такий додаток наддасть змогу людям по всьому світу спілкуватись з приводу утримання домашніх улюбленців. Особливо це актуально для власників екзотичних та нестандартних вихованців, бо інформацію про утримання та дії власників в критичних ситуаціях щодо здоров'я знайти дуже складно, а то й неможливо.

Для досягнення бажаних результатів потрібно виконати наступні задачі:

1. Проаналізувати основні веб-технологій для створення веб-застосунку:
   - дослідити історію виникнення основних технологій для кращого розуміння їх використання;
   - порівняти різні веб-технології для використання найбільш доцільних;

– провести аналіз принципів, підходів, методологій, які використовуються у веб-розробці;

2. Проаналізувати поточний ринок подібних веб-застосунків
3. Визначитись з використанням технологій
4. Спроектувати веб-додаток:

– визначитись з UI/UX принципами;
– обрати структуру проекту;
– визначитись з ТЗ;

5. Розробити веб-додаток з урахування всіх попередніх кроків.

1 ТЕОРЕТИЧНА ЧАСТИНА

1.1 Історія виникнення HTML та принцип роботи

Свій початок технології frontend розробки беруть ще три десятиліття тому. Перший сайт світ побачив 6 серпня 1991 року. Це був примітивний сайт з гіперпосиланнями на різну інформацію про те, що ж він таке та для чого потрібний (рис. 1.1). Тоді, власне, і відбулася презентація всесвітньої павутини – World Wide Web. Наразі, цей сайт все ще доступний за тим самим посиланням, що і у 1991 році.

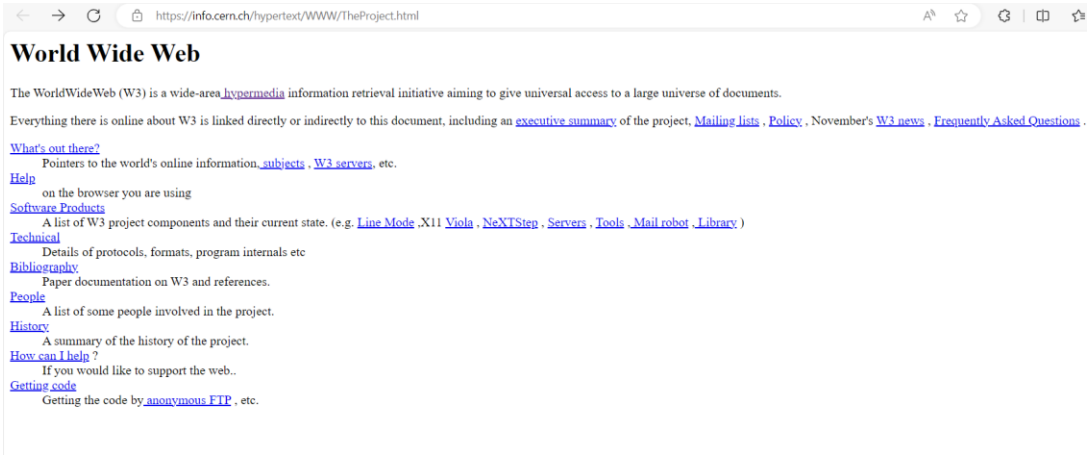

Рис. 1.1 Перший веб-сайт

Якщо подивитись код сторінки (рис. 1.2), то можна виявити, що структура та концепція значно відрізняються від того, що використовують в сьогоднішні дні, а деяких тегів навіть немає й зовсім. Раніше за стандартами HTML було рекомендовано написання тегів у верхньому регістрі (капсом) для покращення читабельності та консистентності коду. Сьогодні ж написання тегів є більш прийнятним у розробці веб-застосунків, хочу й наразі HTML підтримує написання тегів як у верхньому регістрі, так і в нижньому.

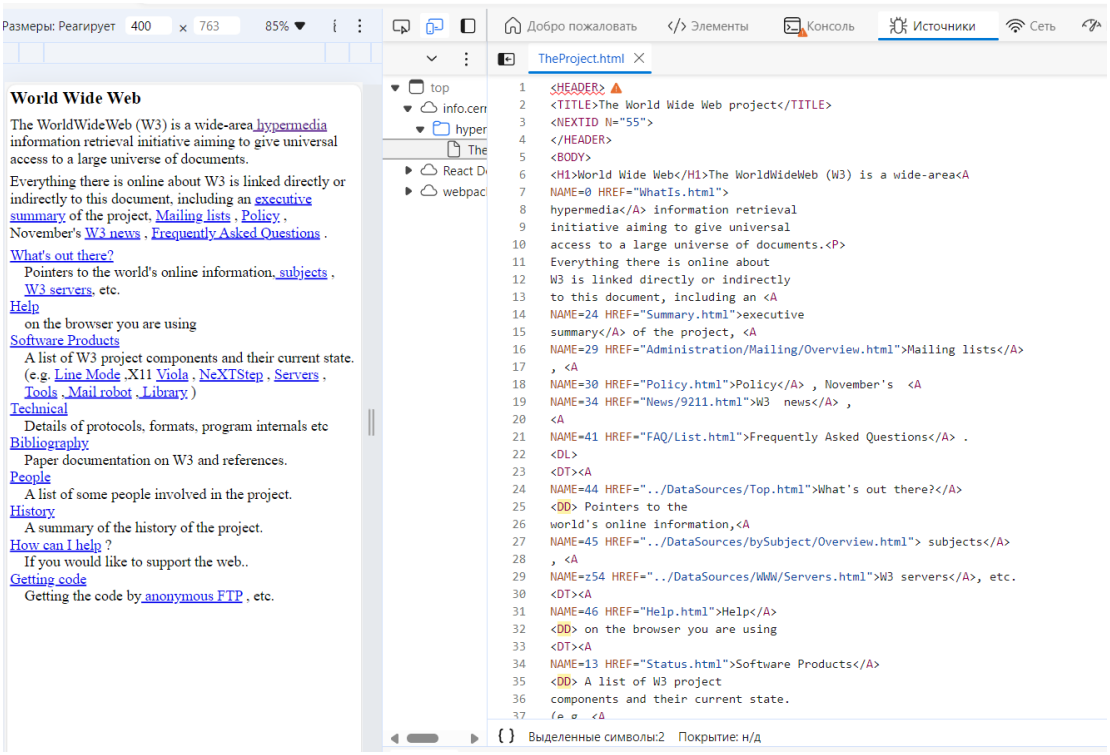

Рис. 1.2 Структура елементів HTML сторінки першого веб-сайту

Сама структура розташування та ієрархія елементів далека від сучасних загальноприйнятих норм написання HTML структури. Хоча це ніяк не впливає на працездатність коду. Саме в цьому й полягає задача frontend-спеціалістів – створювати веб-сайти з подальшим їх відображенням в Інтернеті, незважаючи на постійний розвиток та зміни IT-технологій.

Мова гіпертекстової розмітки HTML (HyperText Markup Language) була розроблена британським вченим Тімом Бернерсом-Лі приблизно в 1986-1991 роках в Женеві (Швейцарія).

Початкова задача HTML полягала в обміні технічної та наукової документації між звичайними людьми, які не були спеціалістами у верстанні сторінок. На той час за допомогою HTML сторінок можна було створити відносно гарний документ без додаткових знань чи вмінь. Задачею HTML було ввідтворення документів на будь-якому обладнанні з різними технічними можливостями.

Варто відмітити, що Тім Бенерс-Лі не був автором ні самої ідеї, ні самого терміна «гіпертекст». Термін винайшов ще у 1963 році Тед Нельсон, який працював над проектом Xanadu (рис. 1.3). Ідеї Нельсона випередили час, що нерідко

трапляється в науці, і, зазвичай, вся слава дістається тому, хто зміг втілити ідею у життя.[5]

Проект Xanadu був першим гіпертекстовим проектом, але зазнало поразки у 1960 році через купу недоліків. Наступна спроба презентувати світу свою ідею припала аж на 1998 рік. Тоді вийшла лише неповна версія, а от вже робоча версія вийшла в 2014 році. Після публікації останньої версії OpenXanadu на сайті авторами зазначено, що Тім Бенерс-Лі вкрав їхню ідею, а її реалізація взагалі має жахливе втілення.

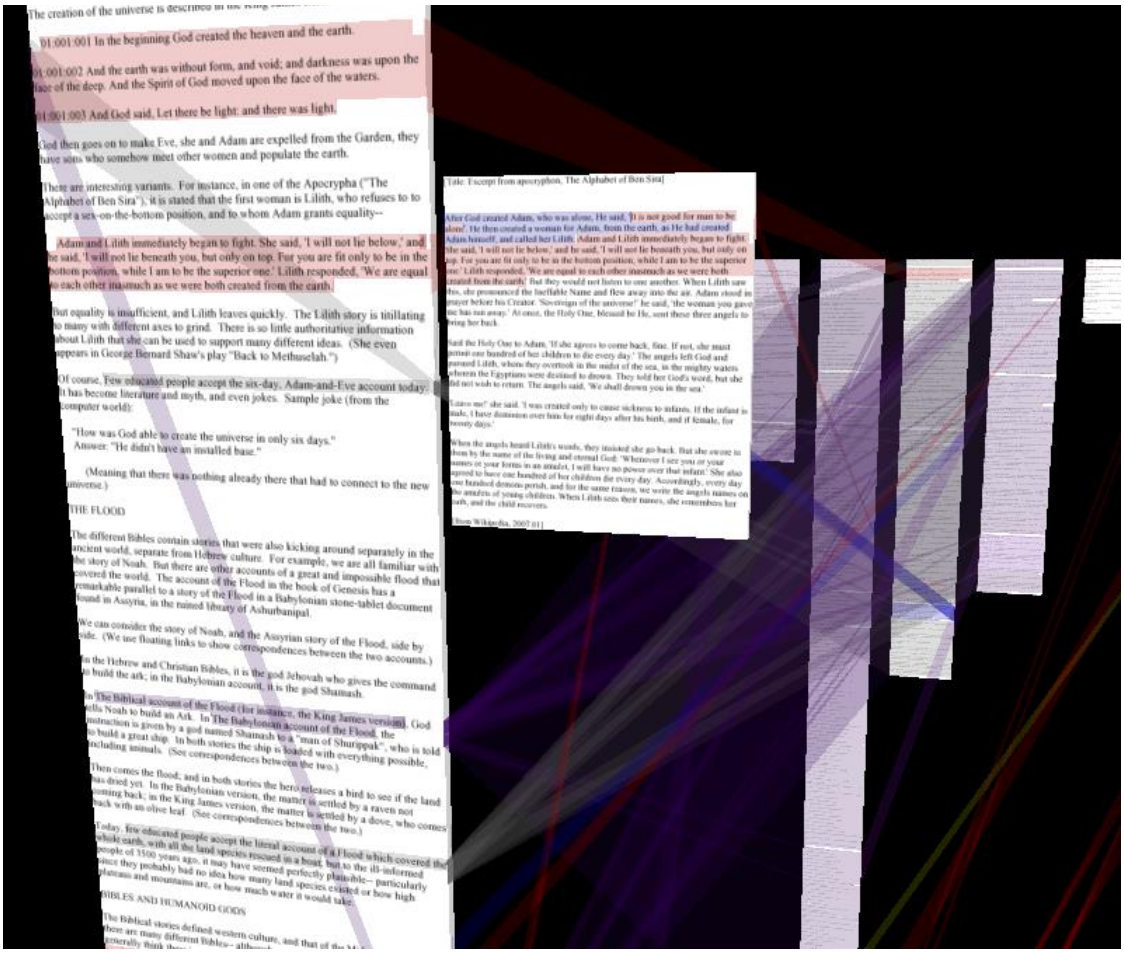

Рис. 1.3 – Проект Xanadu – прототип «ідеального Інтернету»

Xanadu називають концепцією «ідеального інтернету», адже за версією авторів кожний шматок тексту має посилання на документ, з якого його було взято; має захищені протоколи безпеки та видимі посилання; відсутність видалення документу вважається перевагою тощо. Цей проект зайняв 54 роки і як

відзначається в багатьох джерелах, Тед Нельсон мав гану ідею, але йому бракувало технічних знань для демонстрації своєї ідеї.

Наразі, можна зрозуміти, що видана версія 2014 року вже мало кого цікавить, адже з того часу багато що змінилось і вдосконалилось, а ідея Нельсона дуже запізнилась з своєю реалізацією. І хоча Тіма Бенерсона-Лі можна вважати крадієм, але саме він довів цю ідею до прийнятної реалізації, завдяки чому ми маємо саме такий Інтернет, яким ми його бачимо.

Першою формальною версією HTML був HTML 2.0 у форматі RFC, випущений у 1995 році (рис. 1.4). Тобто HTML 2.0 був документом Інтернету, що містив в собі технічні специфікації та стандарти, широко застосовані по всій Інтернет павутині, які декларують внутрішню роботу Інтернету. З початку свого існування та аж до HTML 4.0 мова гіпертекстової розмітки була побудована на SGML. SGML в свою чергу є метамовою, завдяки якій безпосередньо визначають мову розмітки для документів. З другої по четверту версію HTML веб-сторінки будувались як SGML-документи, хоча через вільну інтерпретацію Бернерсом-Лі, веб-сторінки не були коректними SGML-документами.

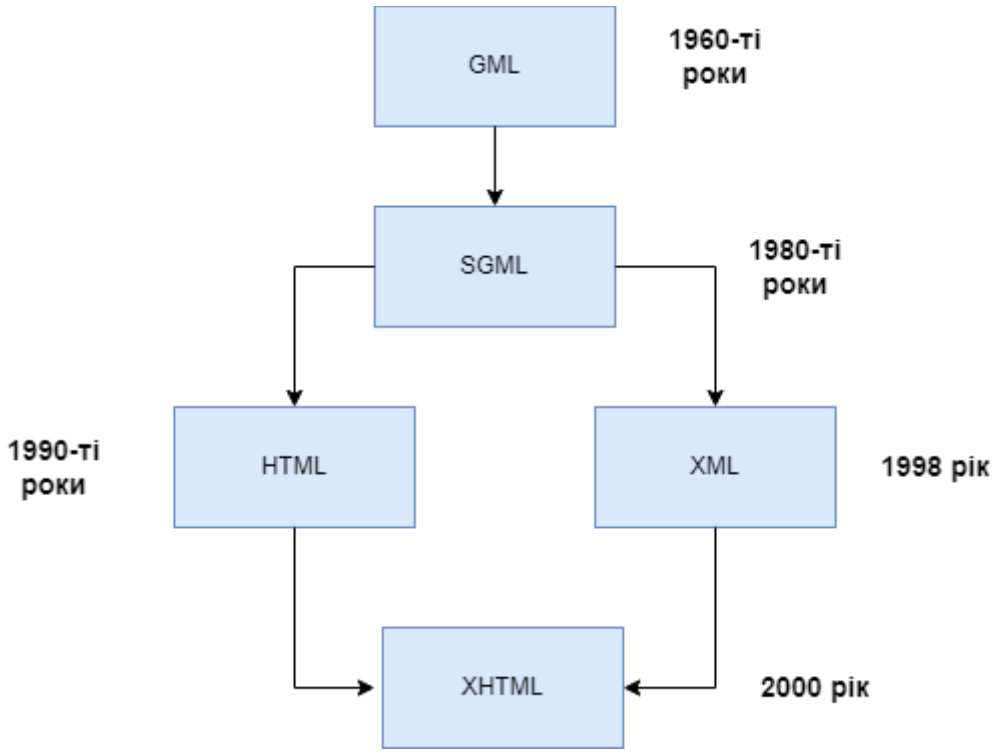

Рис. 1.4 – Історія HTML та взаємозв’язок з іншими форматами текстових розміток

Вже в 5 версії HTML відмовилися від побудови веб-сторінок, як SGML-документів. У жовтні 2014 року виходить нова версія HTML 5, яка розглядається не як додаток до стандартів SGML, а як окремий мовний стандарт маркування, що спрощує його використання. З появою п'ятої версії додаються нові елементи, покращується мультимедійна підтримка та адаптивний дизайн, а головною родзинкою стає поява JavaScript API.

Завзяті розробники й досі можуть використовувати старі версії HTML, зокрема XHTML, хоча необхідності в цьому немає, бо виникає багато проблем з адаптивною стилізацією та стилізацію загалом, а також інші побічні дефекти.

1.2 Історія виникнення CSS та принцип роботи

Відходячи від HTML, розробники неодмінно натрапляють на CSS (Cascading Style Sheets) – каскадні таблиці стилей. Якщо HTML – це скелет веб-сторінок, то CSS його оболонка. CSS дає можливість розробникам створити зручний та гарний інтерфейс з урахуванням всіх UI/UX вимог. Навіть після впровадження CSS тодішні веб-сайти були далекі від сьогоднішніх, але це поклало початок новій ері у світі веб-сайтів.

Ідея Cascading Style Sheets належить норвежцю Гокону Лі, який запропонував ввести можливість застосування стилів до HTML ще у 1994 році, згодом у 1996 році вийшла перша версія стандарту CSS. Варто зазначити, що першими браузерами, адаптованими до каскадних таблиць стилей, стали Internet Explorer та Opera, а вже у першому році нинішнього тисячоліття перша версія була об'явлена як частина інтернет-браузера.[6]

Перша версія включала в себе:

– Параметри шрифтів – гарнітуру, розмір, нахил, курсив, жирність тощо;

– Визначення кольору тексту, фону елементів чи сторінки, обводки, границь та ін;

– Атрибути тексту – його висота, міжстроковий інтервал, міжсимвольний інтервал;

– Вирівнювання тексту, зображень, блоків та більшість інших існуючих елементів;

– Внутрішні та зовнішні відступи блоку та задання його розмірам.

Здавалося б, що з появою CSS створення веб-сайтів повинно стати легшим та цікавішим, але в новій знахідці не було жодних “правил” чи рекомендацій щодо застосування стилей. Це призвело до того, що розробникам доводилось застосовувати набір правил методом “костилів”. Тобто кожен викручувався як міг й використовувати властивості не за призначенням було звичайною справою. Навіть сайт Гокона Лі (рис. 1.5), що був створений за допомогою CSS далекий від гарного прикладу застосування каскадних таблиць стилей.

Рис. 1.5. Сайт-приклад з застосування стилей винахідника CSS Гокона Лі

Попри різноманітність функціональності CSS підтримки єдиного та якісного підходу щодо розміщень елементів на сторінці довго була відсутня. Один із прикладів – застосування властивості float для адаптивного дизайну сайту, хоча його початковим призначенням є розміщення зображень в залежності від обраного значення.

Однією з головних проблем CSS стало різне відображення сторінки в різних браузерах через що розробникам й досі доводиться прописувати окремі правила для різних браузерів.

З часом все більш актуальним поставало питання в адаптивності веб-сайту, особливо для екранів з меншою здатністю розширення екрану, таких як смартфони, планшети, тощо. З появою таких інструментів як Flexible та Grid Layout цю проблему було вирішено, але це вже відбулося за довгий час після виходу першої версії CSS.

З появою останньої третьої версії все більш менш стало на свої місця, і зараз спільнота розробників сформувала більш менш єдиний підхід щодо використання тих чи інших властивостей селекторів, хоча їх “некоректне” застосування можна побачити на багатьох сайтах й дотепер.

Застосовуючи таблиці стилей розробник перш за все повинен думати не тільки про відображення контенту на сторінці, а й про його оптимізацію. Отож, використовуючи CSS багато новачків (і навіть досвідчені спеціалісти), прописуючи стилі, не задумуються про наслідки повторень одних й тих самих стилей, які можна узагальнити та створити шаблони, чи завдяки додатковим можливостям пріоритетності (рис. 1.6) значно зменшити повторні застосування одних й тих самих стилей.

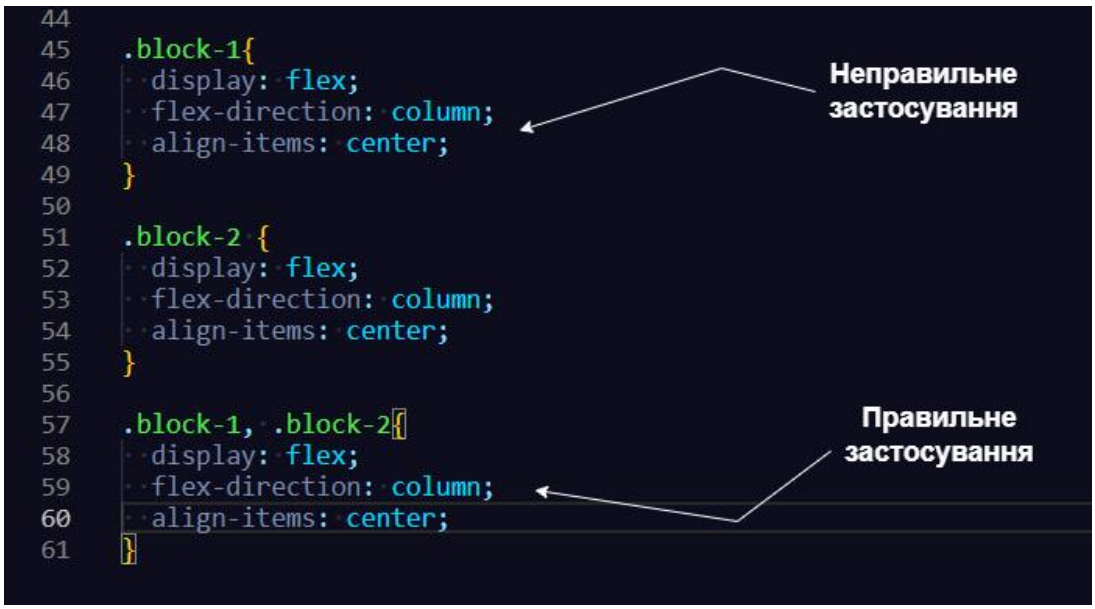

Рис. 1.6 – Приклад оптимізованого та неоптимізованого застосування стилей

І перший, і другий варіанти є валідними з точки зору CSS, але завантаження сторінки буде швидшим при дотриманні другого варіанту написання правил відображення елементів. І здавалося б, що декілька рядків в даному випадку не

мають ніякого значення, але при дотриманні такого підходу можна заощадити навіть до 1000 рядків при створенні багатосторінкового веб-сайту. А це вже в свою чергу має не аби яку перевагу при завантаженні сторінок та сайту в цілому.

Або ж є інший набір правил при якому можна значно заощадити ресурси – використання послідовності селекторів або комбінація селекторів (рис. 1.8).

```
.item-1{
  padding: 20px 20px 20px 20px;
  margin: 0 20px 0 20px;
}

.item-2 {
  padding: 20px 20px 20px 20px;
  margin: 0 20px 0 20px;
}

.item-3 {
  padding: 20px 20px 20px 20px;
  margin: 0 20px 0 20px;
}
```

Рис. 1.7 – Приклад застосування стилей без «оптимізаційного» підходу

```
.block-1>div {
  padding: 20px 20px 20px 20px;
  margin: 0 20px 0 20px;
}
```

Рис. 1.8 – Приклад застосування комбінації двох селекторів

```
<div className="block-1">
  <div className="item-1"></div>
  <div className="item-2"></div>
  <div className="item-3"></div>
</div>
```

Рис. 1.9 – Елементи до яких застосовуються стилі з рис. 1.7 та рис.1.8

На рис. 1.9 представлений HTML код з батьківським блоком та дочірніми до нього. До кожного з цих блоків CSS дозволяє застосувати різні стилі, при цьому отримавши однаковий результат. Але при цьому на Рис. 1.8 застосовується більш «правильний» підхід, як з точки зору чистоти коду, так і з боку оптимізації. І навіть код з більш оптимальним підходом можна замінити на зовсім інший, отримавши при цьому один й той самий результат.

В цьому й полягає одночасно перевага та недолік каскадних таблиць стилей – один і той самий результат можна отримати багатьма шляхами, але кожен розробник обирає той, який вважає за потрібним. Через це в frontend-розробці відсутній єдиний підхід щодо стилізації елементів в тому чи іншому випадку, але це в свою чергу надає розробникам вільність у своїх діях.

1.3 Історія виникнення JavaScript

Як висловлюються у багатьох джерелах, то історія виникнення цієї мови програмування схожа на детектив. Під час створення HTML автор не передбачав ніякої мови програмування у браузері, тому й не дивно, що так все складається.

Починається ця історія з компанії Netscape – американської корпорації, засновником якої є Джеймс Кларк. Далекоглядний підприємець одразу зрозумів, що глобальна мережа Інтернет набирає великих обертів, тому терміново потрібно зайняти своє місце в створенні програмного забезпечення для роботи з нею. Тож в першу чергу він вирішив зібрати команду з молодих талановитих вчених, які допоможуть йому в досягнені поставленої мети.[2]

Так і з’явився Netscape Navigator (рис. 1.10) у квітні 1994 року. Браузер, який стрімко почав набирати обертів і аж до четвертої версії був головним конкурентом Internet Explorer.

Перша публічна версія програмного забезпечення вийшла під номером 0.9 у листопаді 2024 року. Але просто браузера йому не вистачало, тож він вирішив, що великою перевагою HTML буде якась легка скриптова мова, яку можна вписувати безпосередньо в рядки розмітки документа. Недовго думаючи, для виконання

поставленої задачі, був найнятий Брендон Айк – на той час дуже відомий спеціаліст з мов програмування.

Рис. 1.10 – Вигляд Netscape Navigator 9.0.0.4

До початку своєї роботи в Netscape Corporation Брендон працював декілька років в різних компаніях. До речі, саме його діяльність в Netscape та в подальшому в Mozilla, принесла йому не аби який успіх.

Перед ним була поставлена задача вбудувати у Netscape Navigator таку мову програмування, як Scheme. Це функціональна мова програмування, створена в середині 1970-х років. Досить проста мова, що передбачає собою реалізацію всього необхідного через “надбудови” над конструкціями, але треба відзначити, що має досить специфічний синтаксис (рис. 1.11), тому було відхилено пропозицію її імплементування в браузер.[13]

(read-byte [in])	повертає код першого знаку з потоку введення. Наприклад: > (read-byte) 1 **49**
(read-line [in mode])	повертає перший рядок з потоку введення; знаки зчитуються, доки не зустрінеться кінець рядка або кінець файлу. Наприклад: > (read-line) 1 2 3 4 5 6 7 8 9 10 **"1 2 3 4 5 6 7 8 9 10"**
(read-string n [in])	повертає рядок з n знаків із потоку введення. Наприклад: > (read-string 10) 1 2 3 4 5 6 7 8 9 10 **"1 2 3 4 5 "**
(read-string! str [in start-pos end-pos])	рядку str надається значення вилученого з потоку введення рядка; необов'язковий аргумент – потік введення та позиція початку start-pos та кінця end-pos зчитування знаків. Наприклад: > (define рядок (make-string 5)) > (read-string! рядок) 12345 **5** > (write рядок) **"12345"**

Рис. 1.11 – Приклад синтаксису мови програмування Scheme

Наразі, в Інтернеті мало інформації щодо цієї функціональної мови програмування. Навіть на офіційному сайті при переході на будь-які посилання документації виводить помилку, тож інформації небагато та знайти про мову можна лише в небагатьох джерелах, де коротко пробігаються по ній.

Повертаючись до Netscape, погляд пав на мову програмування Java (що посягає велику популярність й досі), яку активно просувала компанія Sun. Американська компанія, створена в 1982 році, займалась розробкою програмного та створенням апаратного забезпечень. Пізніше була передана до рук компанії Oracle. Важливим фактором цієї компанії в frontend розробці стала їх "винаходом" мова програмування Java (рис. 1.12).

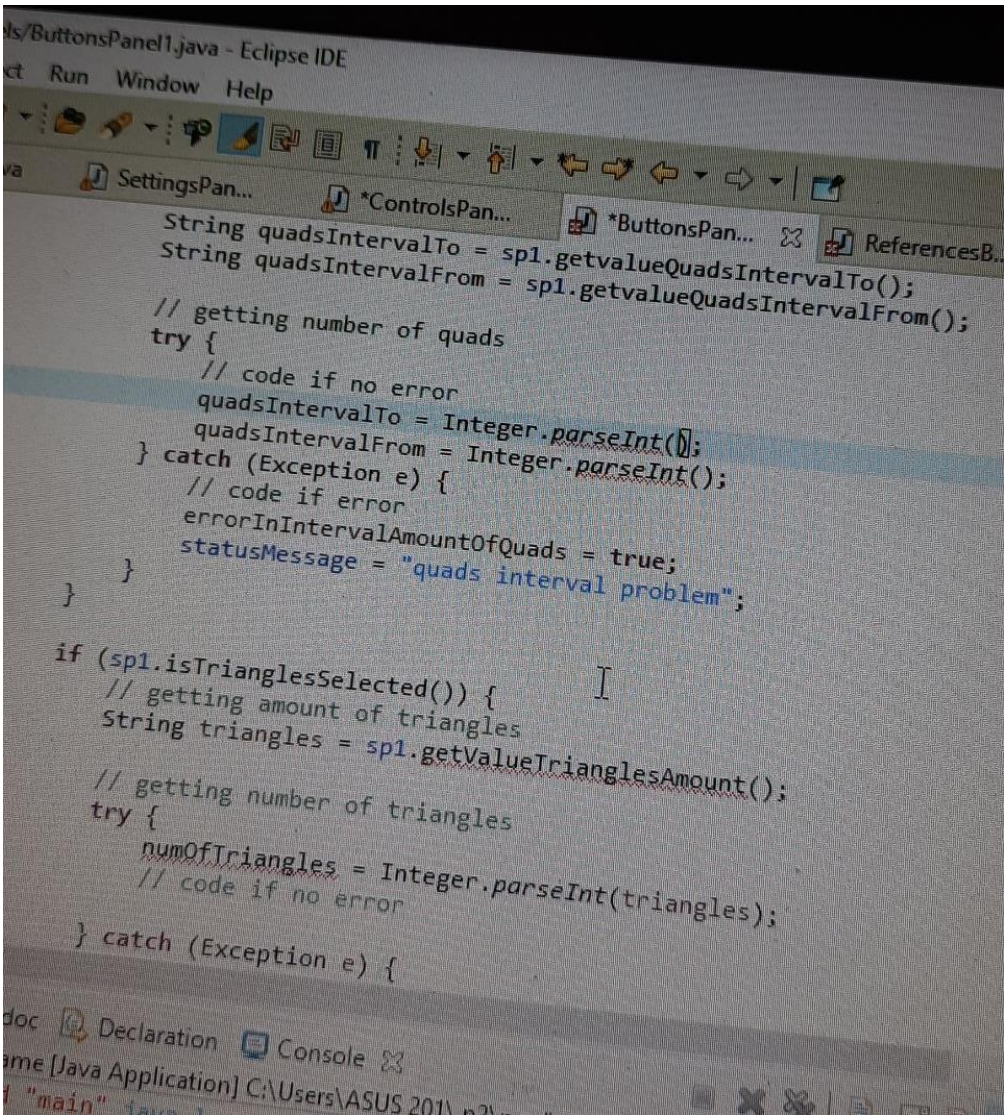

Рис. 1.12 – Приклад коду на мові програмування Java

Отож, першою вимогою, яку висунув Брендон Айк до команди розробників стала – синтаксична подібність до мови програмування Java. Іншою забаганкою (інакше це не назвати) стала реалізація ООП, але без класів. Можна тільки уявити наскільки дивною була ця ідея та не менш “вражаюча” її реалізація.

Відходячи трішки від історії створення, варто зазначити що представляє собою ООП. ООП – це об'єктно орієнтоване програмування в основі якого лежить опис типів (або моделей), представлених з прототипів або як екземпляри класів, що утворюють свою ієрархію наслідування (рис. 1.13). Тобто основою цієї методології є створення та опис будь-якого об'єкта з чітко прописаною послідовністю дій цього об'єкта за допомогою методів у мові програмування.

Рис. 1.13 – Схема Об'єктно Орієнтованого Програмування

Повертаючись до основної гілки історії тепер зрозуміло, що ідея реалізації ООП без класів порушує не просто принципи методології, а спотворює саме поняття про цей стиль програмування.

Через нетерпіння компанії Netscape, Брендону довелось розробити прототип мови всього за десять робочих днів. Отож, у травні 1995 року була створена мова програмування JavaScript (назва якої має зрозуміле значення) скриптового походження для використання у розмітці веб-сторінок. Її основою стали – синтаксис популярної мови програмування Java, функціональність мінімалістичної мови Scheme та ООП з Self.

Окрім вище перелічених недоліків, JavaScript мала ще купу інших проблем. Наприклад, вбудована стійкість до помилок, за думкою розробників мови, мала б залучати недосвідчених фахівців до її користування, а от досвідчені розробники стали насміхатися та не сприймали “винахід” серйозно. Навіть автор мови у 2013 році визнав багато рішень невдалими у своєму Твіттер-блозі.

Ще за тих часів виник тренд – уникати написання коду на frontend та використовувати сторонні програмні забезпечення, що дають змогу створювати візуальні сторінки, а код генерувати автоматично. ! За той час виникло багато відповідних програмних забезпечень, основними з них були: Microsoft FrontPage,

що вийшов 1995 року, та Macromedia Dreamweaver (Adobe Dreamweaver) 1997 року.[10]

Microsoft FrontPage (перша версія вийшла під назвою Vermeer FrontPage) це редактор HTML, що свого часу входив до складу пакету Microsoft Office. Програма на той час мала купу корисного функціоналу, ось його перелік:

1. Зручний редактор, що давав можливість перетворювати візуальне представлення сторінки в код, що значно полегшувало створення веб-сайтів.
2. Додаток містив в собі шаблони та теми, які використовувались як основа під веб-сторінки завдяки чому розробники витрачали менше часу на створення при цьому не втрачаючи візуально гарний результат по завершенню.
3. Інтеграція з програмним забезпеченням Microsoft дозволяла легко імпортувати будь-які файли.
4. Була вбудована підтримка інтерактивних елементів, що дозволяло створювати цікаві веб-сторінки без поглиблених знань у програмуванні.
5. Функція створення веб-форм надавала можливість швидко та якісно додавати функціональні форми на веб-сайти.

З вище переліченого функціоналу (рис. 1.14) можна тільки уявити який фурор зробило це програмне забезпечення. Такий різноманітний та корисний функціонал прискорював роботу в декілька раз, а все незадоволення розробників від, об'єктивно, не дуже коректних технологій веб-розробки зникало й зовсім.

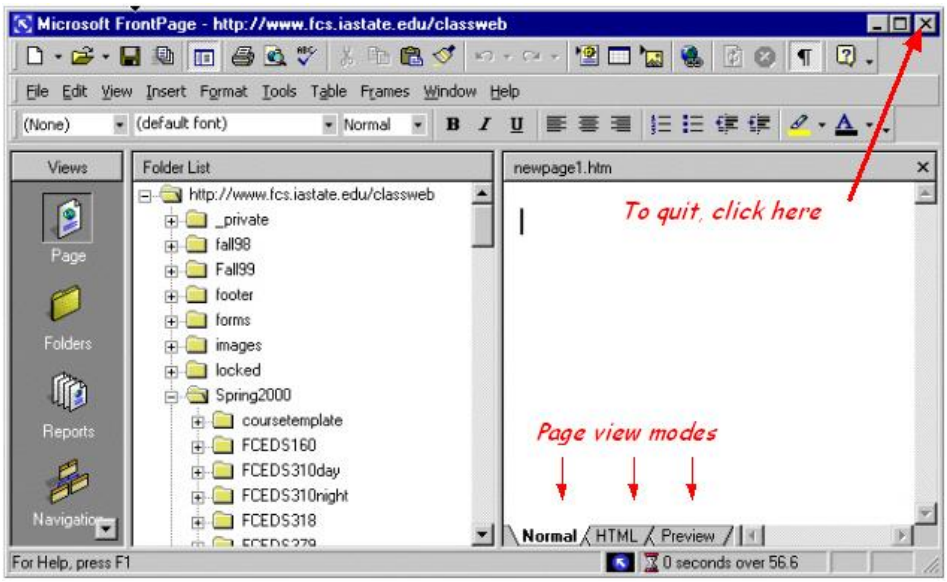

Рис. 1.14 – Інтерфейс Microsoft FrontPage

Macromedia Dreamweaver мав всі ті ж самі можливості, що й попередній конкурент (рис. 1.15), але ще мав в своєму арсеналі підтримку JavaScript, можливість працювати з таблицями, тісну інтеграцію з іншими продуктами свого власника та підтримку у різних браузерах. Останнє значно додавало переваги над Microsoft FrontPage.

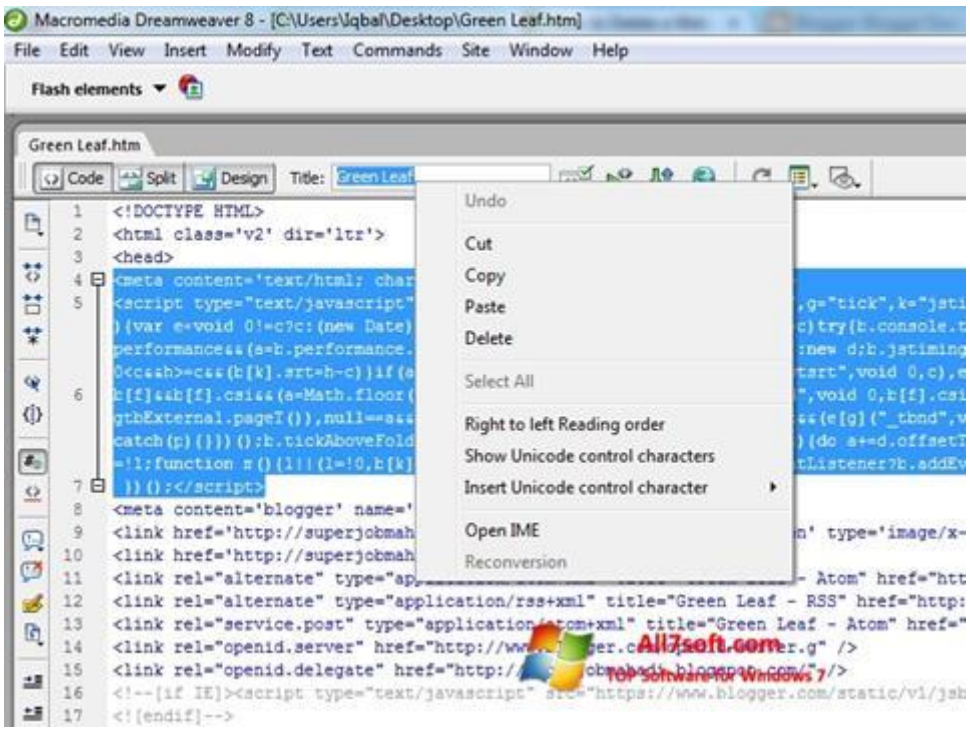

Рис. 1.15 – Інтерфейс Macromedia Dreamweaver 8

Як можна вже було зрозуміти, розробка веб-сайтів сприймалась більш як робота для дизайнерів, аніж для розробників.

Перед тим як мова піде про backend розробку у створенні веб-застосунків треба розібратись що він собою представляє. Це таке серверне середовище, що працює в невидимій частині будь-якого додатку, будь-то веб-застосунки або ж десктопні програмні забезпечення. Вся розробка в цій частині направлена на роботу з даними: їх обробку, перетворення, збереження, захист, організацію тощо. Backend так само, як і frontend поєднує між собою велику кіл-ть процесів та технологій, що на виході дають можливість гармонічно працювати всьому додатку.[17]

Повертаючись до історії становлення frontend мова піде про таку технологію як CGI (Common Gateway Interface), яка дозволяла динамічно генерувати HTML. Її поява припала на 1993 рік.

І початком нової ери в створенні веб-сайтів стає поява PHP в 1995 році. Вона була створена для frontend-розробки та органічно поєднувалась з мовою гіпертекстової розмітки. Багато хто, після появи серверної мови, намагався створити їй конкуренцію, але зусилля зазнали краху. До речі, мова PHP й досі залишається актуальної для створення веб-застосунків.

Підсумовуючи історію JavaScript, залишаються змішані думки, адже такий великий шлях довелось подолати до того, що мова представляє собою зараз. А дізнавшись як відбувались події від початку до кінця, стає зрозумілішим принцип роботи, функціонал та наскільки кропітким процесом є становлення кожного веб-додатку за допомогою цієї мови зараз.

1.4 Google MobileNet

Як вже зрозуміло з назви цей модуль розробила компанія Google у 2018 році, а у 2017 році була випущена перша версія модуля MobileNet.

Ще з якихось десять років тому будь-яка система розпізнавання зображень вважалась досить важким алгоритмом з великими затратами обчислювальних ресурсів, що значно обмежувало використання таких алгоритмів. Але як то буває зі зростом технологій, незабаром використання таких “важких” технологій стало доступним навіть на мобільних пристроях.

Так Google і випустила MobileNet, що являє собою глибоку нейронну мережу, створену для якісного розпізнавання зображень та об’єктів на пристроях з невеликими можливостями обчислювальних ресурсів. Годі й тільки уявити, яким це стало проривом в IT-технологіях створення такого винаходу, який ще декілька років тому був просто неможливим.

Перша версія представляла собою глибокі розділені згортки, що дають змогу зменшити обчислювальні складності у розпізнаванні зображень. Глибока розділена згортка включає в себе згортку за каналом та точкову згортку. Таким чином

зменшується кіл-ть параметрів та обчислень, що в свою чергу не потребує високої обчислювальної потужності.

Другу версію було вдосконалено, шляхом інтеграції блоків зворотних залишків та розширювальної згортки. Принцип дії перших полягає в тому, що застосовується велика кіл-ть каналів для операцій згортки та інших обчислювальних операцій. Такі операції значно зменшуються обчислювальний ресурс та мають широке застосування в сучасних архітектурах нейронних мереж глибокого навчання. А от розширювальні згортки збільшують кіл-ть каналів для застосування точкової згортки. Все це в купі покращило продуктивність та ефективність роботи в другій версії.[8]

Постає логічне питання: а як можна використати цю архітектуру у веб-додатку, якщо вона розроблена спеціально для мобільних пристроїв? Все просто. Існує така бібліотека, як TensorFlow.js. Вона розроблена для машинного навчання в браузері та дозволяє використовувати вже навчені моделі безпосередньо в браузері або на Node.js.[4]

Через офіційний сайт бібліотеки можна знайти документацію (рис. 1.16) короткого опису моделі та безпосередньо інструкцію по її використанню.

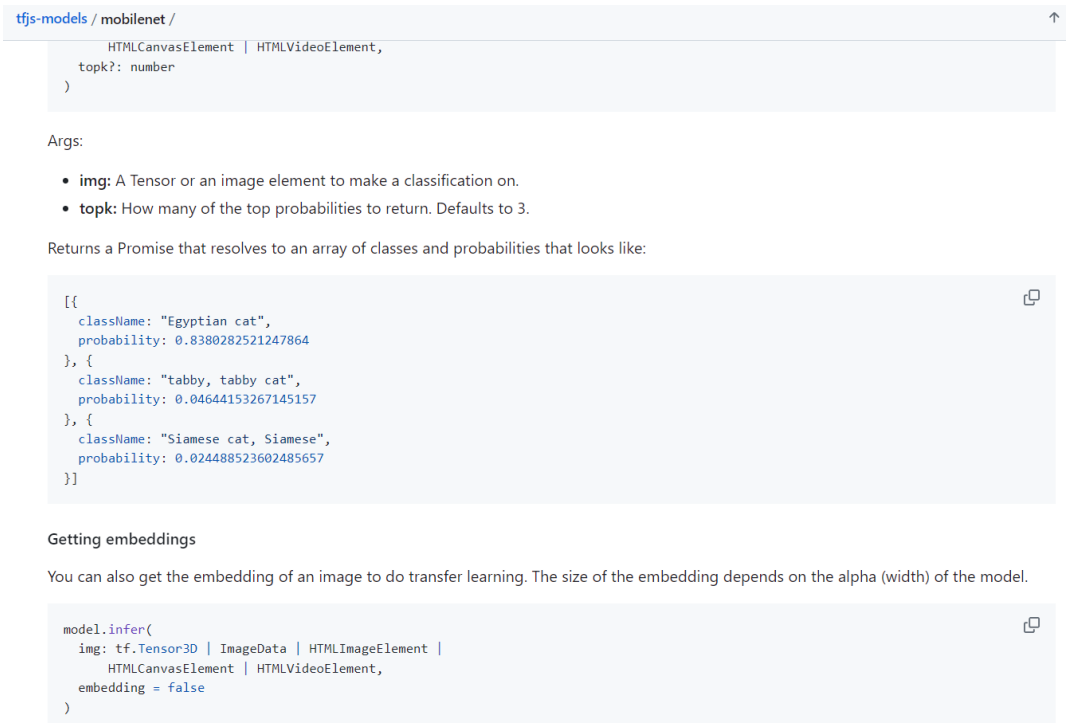

Рис. 1.16 – Документація по використанню MobileNet з GitHub

2 АНАЛІЗ АНАЛОГІВ ТА ТЕХНОЛОГІЙ

2.1 Загальна ситуація веб-розробки у світі та в Україні

Якщо проаналізувати веб-сайти, які створювались тридцять років тому та зараз, то одразу можна зрозуміти наскільки великий прорив пройшов у сфері створення веб-додатків. Але декілька років тому почало набирати обертів використання конструкторів чи шаблонів у створенні веб-додатків. З одного боку, це значно заощаджує час та ресурси на створення, а от з іншого боку – пригнічує розвиток веб-технологій. Веб-сайти дедалі стають шаблонними та схожими один на інший, а про різноманітність контенту та інтеграції цікавих інтерактивних частин мови взагалі і не йде. Таке широке застосування шаблонності пригнічує зріст та розвиток технологій.

Особливо це відображається на ситуації ринку праці. Адже все більше використовують усілякі конструктори для створення веб-сайтів, за примітивними шаблонами. Ось список популярних конструкторів для створення веб-сайтів:

– WIX;
– Squarespace;
– Webflow;
– Shopify;
– WordPress та ін.

Наразі, конструктори сайтів займають десь близько 30-40% від всіх веб-сайтів у світі. Інша частина (70-60%) припадає на веб-технології, такі як React.js, Vue.js, Angular та основний стек (рис. 2.1).

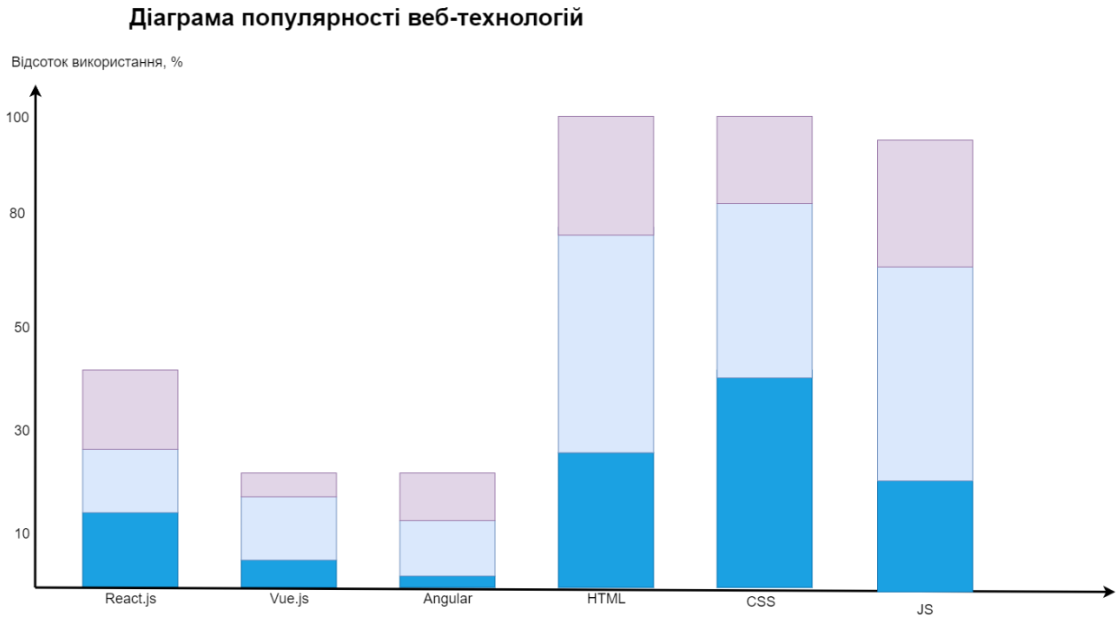

Рис. 2.1 – Статистика використання веб-технологій у світі

React.js займає 30-40% від усього ринку в той час як Vue.js та Angular поділили між собою по 20% (рис. 2.1). Така популярність React.js зумовлена простотою свого використання, що чудово підходить як дня початківців, так і для професіоналів. В багатьох джерелах про веб-технології зазначається, що «React є чудовим стартом для знайомства з фреймворками після опанування базисних технологій». Vue.js та Angular мають дещо складнішу архітектуру вибудови компонентів та в цілому більш нагромаджений функціонал, що в свою чергу впливає зменшення швидкості створення застосунків та більшої кількості помилок. А для новачків та простих веб-застосунків це однозначно складний варіант.[12]

2.2 Аналіз CMS

CMS (Content Management System) – це веб-додатки або ж додатки, що допомагають швидко створювати веб-сайти за шаблонами. З CMS більш менш все зрозуміло. Лідерами є WordPress та Shopify, що мають від свого ринку 38% та 20% відповідно. Також слід зазначити, що існують ще конструктори для створення веб-сайтів, але вони мають ще більш обмежені можливості та менший функціонал, аніж вище перелічені. Наприклад українська торгова платформа Prom.ua дає можливість створювати сайти за своїм шаблоном.

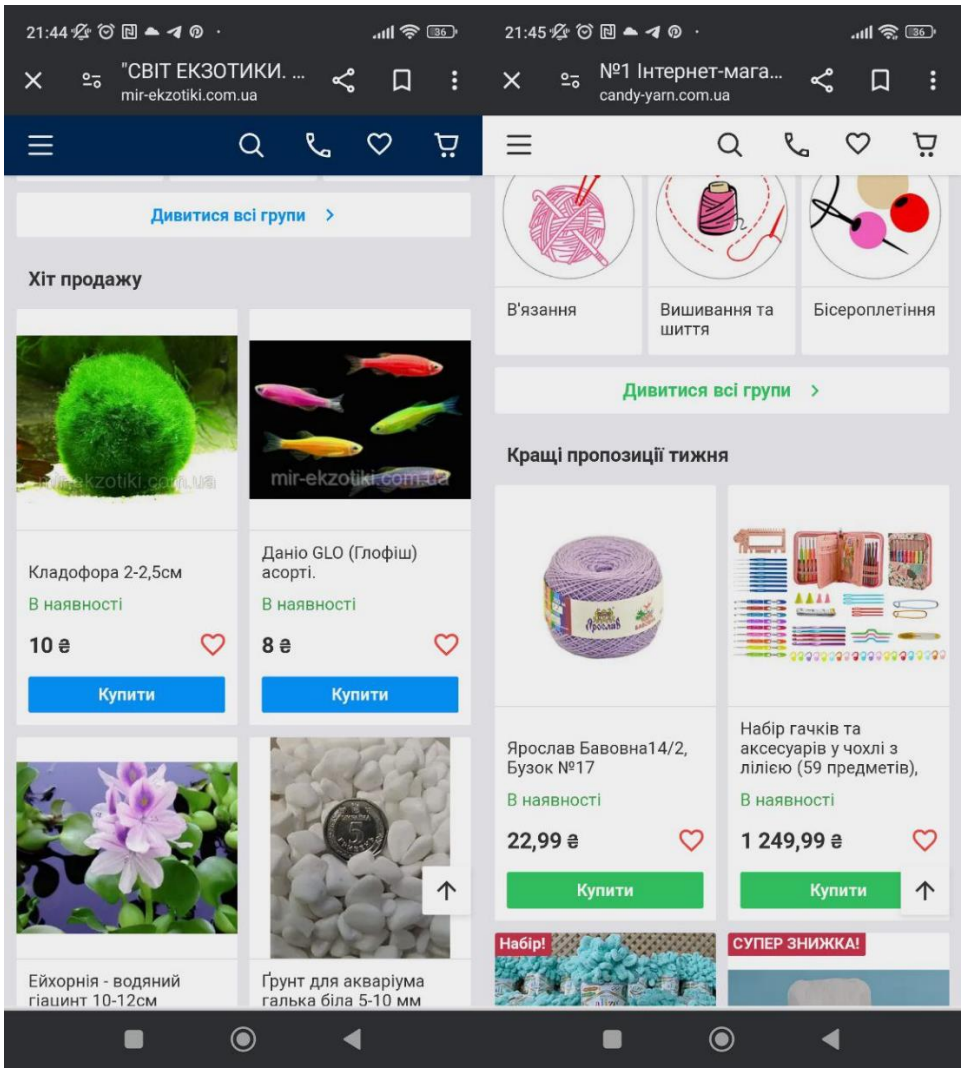

Рис. 2.2 – Приклад сайтів, створених на Prom.ua

На рис. 2.2 можна побачити на скільки схожі ці інтернет-магазини, хоча мають зовсім різну тематику. Веб-сайти на основі Prom використовують виключно для створення інтернет-магазинів. І хоча такий спосіб не потребує ніяких знань веб-технологій та в більшості випадків такі веб-магазини розробляють їх власники, але магазин одразу втрачає свою унікальність та не справить враження на жодного користувача. Такий довгий час від початку 1990-х років до сьогодні було пройдено, для того щоб люди створювали сайти в стилі 2010-х? Таке широке використання конструкторів приводить до ряду проблем, хоча має й переваги.

Як вище було зазначено – веб-сайти не мають унікальності. Сфера маркетингу значно зросла та зараз ні для кого не секрет, що велика кількість прибутку та продажів в того, хто вміє зацікавити клієнта. Кожен власник бізнесу

повинен в першу чергу задуматись про створення унікальної пропозиції та гарної подачі. А інтернет-магазини, створені за допомогою CMS, не дають можливості себе зарекомендувати та зацікавити. Можливо клієнт і згадає про магазин згодом, але з яскравою подачею свого продукту чи послуг на веб-сторінці цей шанс збільшується в рази. Також, іншою проблемою такого масового застосування конструкторів є зниження довіри в користувачів, адже, в більшості випадків, веб-сайт зроблений за шаблоном видно неозброєним оком. Отож, коли користувачі переходять за посиланням і бачать будь-яку шаблонну сторінку – рівень довіри зменшується, бо сприймається це як халатне відношення до свого бізнесу або ж низький рівень репутації.

Є дійсно унікальні сайти з гарним інтерфейсом побудовані за допомогою конструкторів, але їх лічені одиниці і зазвичай вони потребують значно кількість часу та знань CSS чи JavaScript для того, щоб зробити з надаваємого шаблону більш унікальний.

До того ж, CMS не є панацею, адже мають ще ряд суттєвих недоліків, крім зазначених вище. До таких недоліків відносяться прив’язаність до цих самих конструкторів, бо при створенні веб-сайтів на подібних платформах клієнти вимушені користуватись технічною підтримкою так сервісів, хостингом та підлаштовуватись під їх оновлення, що може призвести до подальших проблем з вже створеними веб-сторінками. Також, CMS суттєво впливає на пошукову оптимізацію, тому завчасно потрібно готуватись до того, що покази веб-сайтів у браузерах матимуть низькі результати, адже конструктори нерідко дають не найкращі пропозиції щодо SEO-оптимізації.

Іншими суттєвими недоліками конструкторів є примусове розміщення реклами на створених веб-сторінках та неякісна безпека. Часті випадки, коли конструктори-сервіси розміщують свою рекламу або ж примушують платити додаткові кошти за відсутність такої реклами на веб-сторінках користувачів.

Ну і на додачу до всього цього якщо користувач вирішить скасувати свій план плати чи перейти на інший сервіс, то він запросто може втратити доступ до даних веб-сайту, сам веб-додаток, не кажучи про те, що перенесення свого веб-сервісу на

іншу платформу потребує не просто редагування, а його створення майже з самого початку. В той час, як веб-додатки створенні за допомогою фреймворків чи основного стеку веб-технологій досить легко змінити, щось додати/видалити або перенести на інший хостинг.

2.3 Ринок праці в IT-технологіях

Ринок праці в IT зазнає великої кризи, ситуація з розповсюдженням конструкторів лише погіршує ситуацію. Ось список факторів, що спричинили нестабільну ситуацію на ринку праці в Україні:

– підвищення кількості кандидатів на одне вакантне місце, спричинене covid-19 та дестабілізацією карантином у світі;

– підвищення кількості кандидатів на одне вакантне місце після початку повномасштабного вторгнення у 2022 році, що спричинило вкрай важку нестабільну ситуацію на ринку праці;

– вихід великої кількості закордонних компаній з українського ринку

– передислокація українських IT компаній закордон;

– економія компаній на кількості співробітників та небажання навчати кандидатів без досвіду;

– розповсюдження CMS та ін.

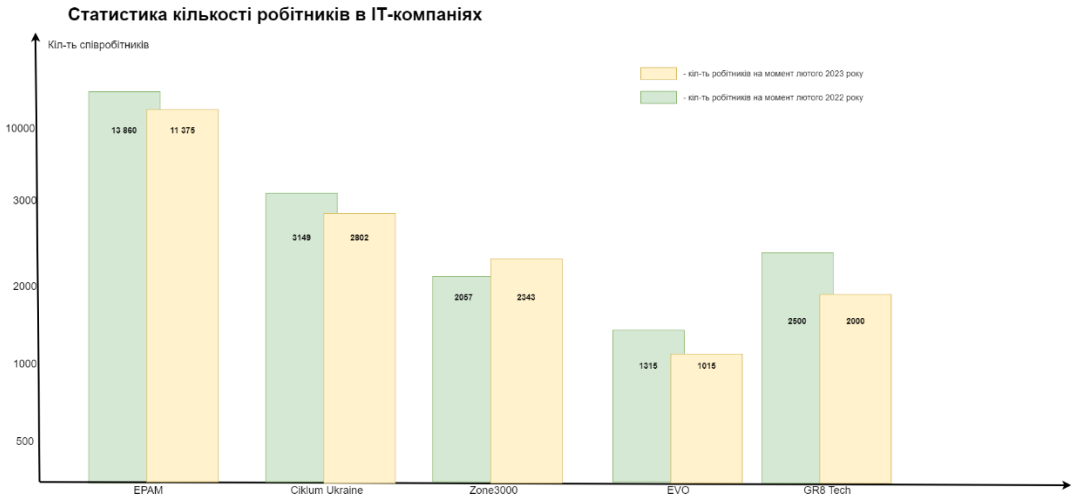

Рис. 2.3 – Статистика кількості робітників в українських IT-компаній

На рис. 2.3 можна побачити, що кількість робітників в українських IT-компаній значно зменшилась за рік від початку повномасштабного вторгнення у

лютому 2022 року. Такі компанії як EPAM, Ciklum Ukraine, EVO та GR8 Tech зменшилися в штаті від 20% до 12%. Єдиним виключенням в цій статистиці стала компанія Zone3000, їй вдалось розширити свій штат майже на 200 осіб. За оцінками DOU.ua таке суттєве зменшення кількості робітників пов'язане з тим, що близько 60% фахівців були вимушені покинути свої домівки та переїхати в більш безпечне місце, а п'яти тисячам спеціалістів довелось виїхати за кордон.[9]

Починаючи з кінця списку, значну дестабілізацію вносить факт розповсюдження конструкторів для веб-сайтів, особливо в умовах війни через те, що малий та середній класи підприємств для того щоб вижити намагаються заощадити кошти через це надають перевагу конструкторам сайтів та, в більшості випадків, отримають неякісний результат як функціональний, так і візуальний. За даними Wix, Nethouse та OkayCMS попит на використання їх продуктами зріс на 35%, 45% та 20% відповідно з початку 2022 року, що свідчить про перевагу вартості через нестабільну економічну ситуацію в країні. Через зростання переваги конструкторів над веб-технологіями IT-компаніям доводиться також доводиться скорочувати штат через нерентабельність утримання кількості спеціалістів.

Підсумовуючи все вище описане треба сказати, що немає нічого поганого в тому, що створюються додаткові рішення для прискорення, оптимізації та полегшення роботи не тільки у frontend розробці, а й в будь-якій сфері взагалі. Але це може бути гарним рішенням тільки в тому випадку, коли ці нові технології дають дійсно якісний результат.

2.4 Аналіз веб-додатків/додатків конкурентів

Проаналізувавши ринок конкурентів, які мають схожу реалізацію ідеї, були отримані наступні результати:

1. Вдалось знайти лише один додаток «Petzbe» (рис. 2.4) для мобільних пристроїв, який можна завантажити з Google Play (для Android) та App Store (для iOS).

2. В одній з статей 2011 року була знайдена інформація щодо веб-сайту “Мордашки.com” від харківських розробників, але знайти даний веб-додаток не вдалось.

3. На просторах інтернету було знайдено ще декілька згадувань про різні додатки для тварин, але всі посилання були неактивні.

По-перше, відсутність веб-додатків/додатків наштовхує на думку про те, що цей ринок вільний та є можливість стати лідером в цій сфері. Всього лише один додаток, який не має великого попиту. Протестувавши додаток «Petzbe» знайдені наступні переваги:

– великий функціонал;

– тематичність.

Рис. 2.4 – Мобільний додаток Petzbe

На жаль, на цьому переваги додатку завершуються. Переходячи до недоліків, можна сказати наступне:

– додаток має проблеми з адаптацією під пристрої, бо на один з смартфонів не було можливості завантаження (рис. 2.5), хоча девайс має не найновішу версію Android (рис. 2.6);

– відсутність веб-версії;

– додаток хоч і містить в собі великий функціонал з різними можливостями, але інтерфейс виглядає дещо застарілим та неякісним та доволі незрозумілим;

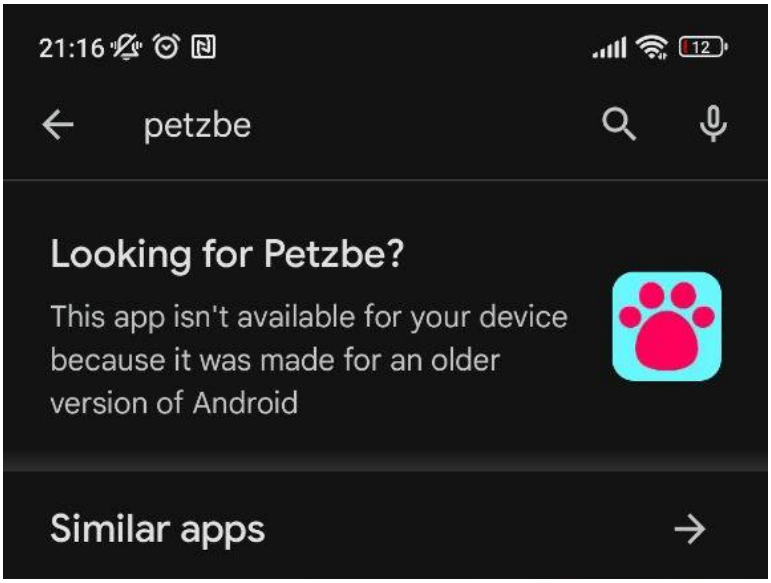

Рис. 2.5 – Результати пошуку додатку

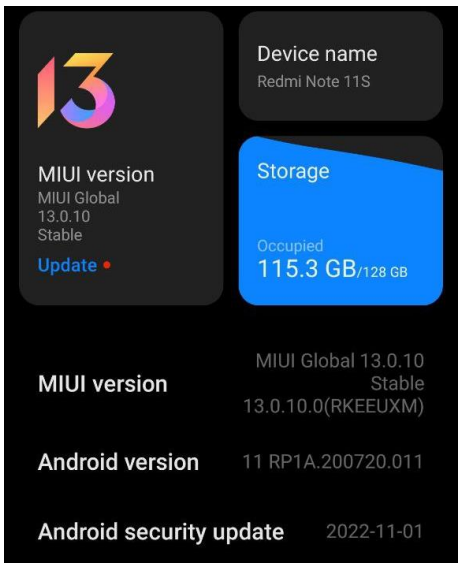

Рис. 2.6 – Версія ПО смартфорну

– також при реєстрації виникли складності, поля для вводу інформації неперепрацьовані та й зареєструватись вийшло лише з другого разу під іншою поштою без зрозумілих на то причин.

В цілому, задум гарний, але допрацювати треба багато чого. З цікавих можливостей в цьому додатку знайдено:

– можливість створювати опитування, ставити питання на форумі (рис. 2.7);

– є категорії зі статтями на різну тематику;

– реалізовано пошук користувачів за різними критеріями (рис. 2.8).

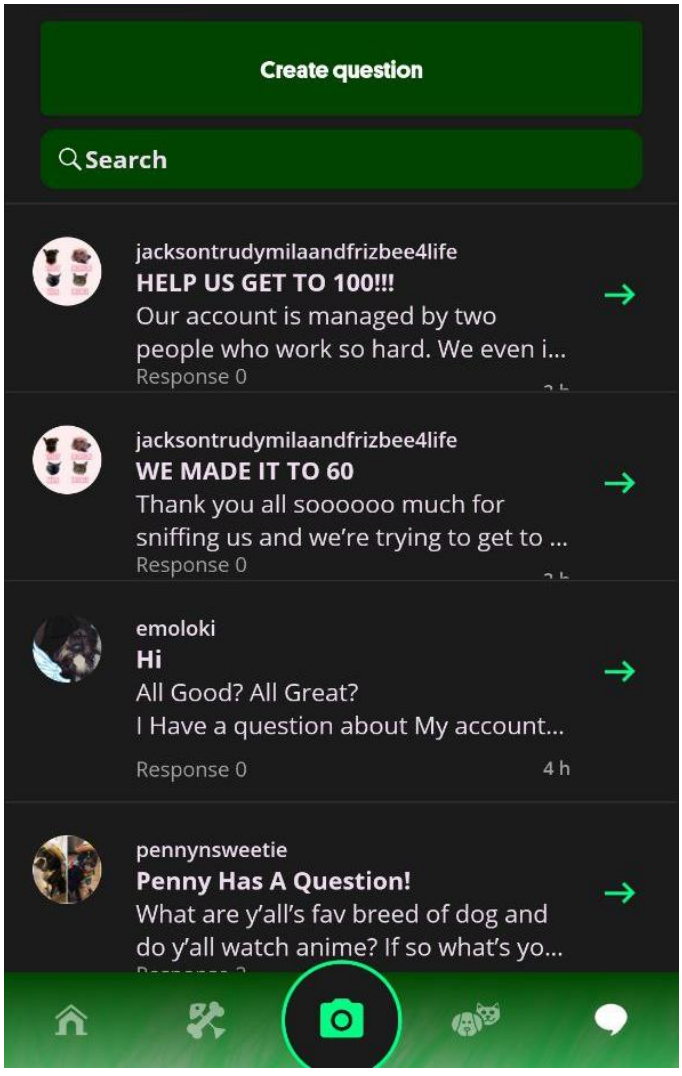

Рис. 2.7 – Форум «Petzbe»

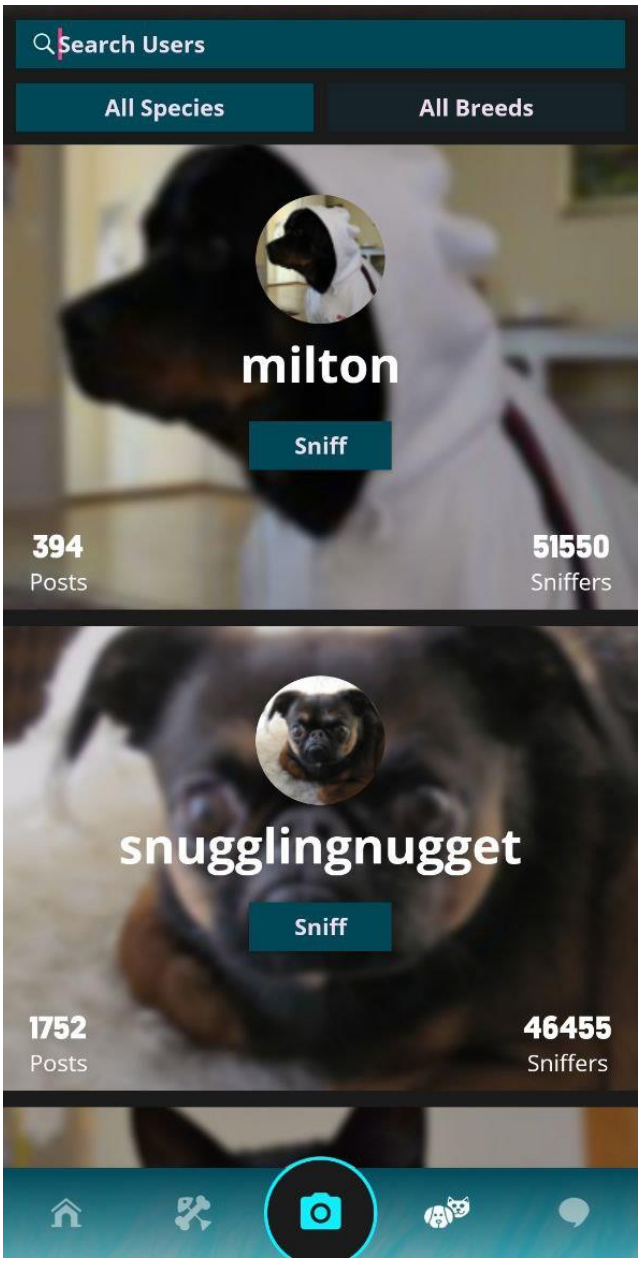

Рис. 2.8 – Пошук користувачів в «Petzbe»

2.5 Аналіз використання HTML

З HML все зрозуміло. Не існує на даний момент іншої мови розмітки веб-сторінок, які б використовувались у світі для створення веб-застосунків. Для створення веб-контенту на кожній сторінці потрібно застосовувати мови HML. Оскільки в проекті буде використано такий фреймворк, як React.js, то розробнику надаються можливості використання формату JSX (рис. 2.8). Це синтаксичне розширення JavaScript, яке дозволяє створювати HML-подібний код, що значно спрощує створення всього контенту (особливо динамічного).

```
return (
  <>
    <TabNavForGallery
      onChangeHandlerType={onChangeRequest}
      onChangeHandlerBreeds={onChangeRequest}
      onChangeHandlerLimit={onChangeRequest}
      onChangeHandlerSort={onChangeRequest}
    ></TabNavForGallery>
    <div className="gallery-content">
      <GridBody
        gridContent={historyGallery[counter]}
        handlerFav={handlerClickGalleryFav}
      ></GridBody>
      <div className="page-switch">
        <button
          className={counter === 0 ? "sw-prev-dis" : "sw-prev"}
          onClick={handlerClickPrev}
        ></button>
        <button
          className="sw-next"
          onClick={handlerClickNext}
        ></button>
      </div>
    </div>
  </>
```

Рис. 2.8 – Приклад JSX формату в коді

2.6 Аналіз використання CSS

З теоретичної частини зрозуміло, що CSS є ще однією основною складовою при розробці будь-яких веб-застосунків. І хоча існують такі препроцесори, як SASS та SCSS, які також можна інтегрувати з React.js, але така інтеграція несе в собі ряд недоліків:

– використання препроцесорів потребує додаткового встановлення необхідних інструментів;

– складна інтеграція (особливо у великих проектах або при використанні специфічних інструментів);

– додаткове навантаження для браузера (особливо відображається на мало потужних пристроях);

– складна сумісність з CSS модулями та додаткові налаштування, через що виникають постійні конфлікти.

Тому, було прийняте рішення – не ускладнювати роботу та не наражатись на непотрібні додаткові проблеми при розробці, обмежившись лише використанням CSS. Використання стандартної стилізації і без зайвих проблем забезпечить відмінну якість інтерфейсу веб-додатку.

Основні переваги використання CSS буде полягати в створенні адаптивного веб-дизайну, яке забезпечить якісне відображення контенту на екранах з різною роздільною здатністю. Такого ефекту можна досягнути за допомогою використання Flex та Grid Layout.

2.7 Аналіз використання JavaScript

JavaScript є однією з основних мов програмування у веб-розробці. Її використання повністю покриває створення всього функціоналу веб-застосунку з боку frontend-частини. Хоча не рідко підмічається використання JavaScript й у деяких випадках, де зазвичай застосовується backend.

Ще декілька років тому почала набирати популярність у веб-розробці така мова, як TypeScript. Вона вважається покращеною версією JavaScript, що, на думку багатьох, вплинуло на більш серйозне відношення до мови, ніж то було раніше. Головною перевагою TypeScript стала типізація даних, адже JavaScript має динамічну типізацію, на відмінно від більшості мов програмування. Але використання TypeScript в рамках даного проекту не є доцільним за декількома причинами:

– вимагає додаткового налаштування для проекту;

– збільшує час на компіляцію, що суттєво може сповільнити розробку;

– виникнення проблем при використання бібліотек через не передбачення в них типізації;

– в невеликих проектах не є доцільним таке використання.

Та й в загалі, використання TypeScript потребує додаткового опанування стороннього програмного забезпечення. Його використання є більш доцільним для розробників (або групи розробників), які пишуть frontend та backend частини.

Оскільки проект кваліфікаційної роботи на даному етапі не передбачає використання backend технологій, то використання JavaScript буде цілком достатньо для виконання поставлених задач.

2.8 Аналіз використання React.js

React.js є одним з найпопулярніших фреймворків у світі. Він вважається одним з найфункціональнішим та водночас простою бібліотекою для молодих спеціалістів. Його переваги полягають в наступному:

– компоненти та перивокористовувані компоненти;
– інкапсуляція компонент;
– віртуальне DOM–дерево;
– зручний потік даних;
– гнучкість JavaScript;
– вбудовані інструменти;
– інтеграція з іншими бібліотеками;
– можливість розробки для мобільних додатків та ін.

Компоненти в React.js – це такі самостійні окремі частини веб-контенту, які мають свою логіку та HTML–шаблон, завдяки чому можна створювати цілі окремі частини веб-застосунку не переймаючись через глобалізацію логіку, функціоналу тощо.

Завдяки Virtual DOM відбувається “екологічний” рендеринг сторінок, що дозволяє фреймворку спочатку відслідковувати зміни в віртуальній структурі веб-сторінки, а вже потім вносили лише необхідні зміни до справжнього DOM-дерева. Завдяки такому підходу, розробникам не доводиться продумувати логіку постійних змін на сторінці, а обчислювальні потужності пристроїв не перенавантажуються.[1]

Потік даних від компоненти до іншої відбувається від батьківської компоненти до дочірньої. Але з часом такий підхід став досить не функціональним у розробці (особливо великих проектах), тож майже в кожному проекту прийнято використовувати додаткову бібліотеку Redux за замовчуванням.

Вбудовані інструменти фреймворку дозволяють ефективно використовувати час при розробці будь-якого веб-застосунку. Наприклад, та ж сама конструкція «useEffect» (рис. 2.9) дозволяє відслідковувати зміни в даних й при їх зміні запускати певну логіки самостійно. Або ж інший інструмент «useState» дозволяє слідкувати за змінними і при їх зміні невідкладно ререндерити весь контент та код, де використовується ця змінна.[7]

Завдяки React Native бібліотека надає можливість створення застосунків під мобільні пристрої, що не потребує додаткового навчання та навичок, а дозволяє використовувати ті ж самі методи та принципи, як і при звичайній розробці веб-застосунків.

```
useEffect(
  () =>
    function breeds() {
      let arrAPI = [];
      if (counter >= historyGallery.length) {
        new Promise((resolve, reject) => {
          fetch(link, {
            headers: { "x-api-key": userId },
          }).then((data) => resolve(data.json()));
        }).then((result) => {
          const arrFaves = [];
          new Promise((resolveFaves, reject) => {
            fetch("https://api.thecatapi.com/v1/favourites", {
              method: "GET",
              headers: {
                "content-type": "application/json",
                "x-api-key": userId,
              },
            }).then((data) => resolveFaves(data.json()));
          })
            .then((resultFaves) => {
              result.forEach((item) => {
                item.isFav = resultFaves.some(
                  (itemFav) => itemFav.image.id === item.id
                );
                arrAPI.push(item);
              });
              resultFaves.forEach((itemFav) => {
                arrFaves.push({
                  imageId: itemFav.image.id,
                  favId: itemFav.id,
                });
              });
              return arrAPI;
            })
            .then((result) => {
              setAPIArr(result);
              setArrFaves(arrFaves);
              dispatch(addHistory([result]));
```

Рис. 2.9 – Приклад інструменту «useEffect»

2.9 Аналіз використання Redux Toolkit

Дана бібліотека є невід'ємною частиною інструментів при розробці веб-застосунків на React.js. Оскільки потік даних в самому фреймворку є дещо специфічним й обмеженим, то використання Redux Toolkit є вирішенням цієї проблеми. З її допомогою створюються сховище для зберігання змінних, що надає змогу не передавати дані через великий ланцюжок компонент від батьківської до дочірньої, а просто створити сховище з даними, до якого будуть мати доступ усі компоненти веб-додатку.[3]

3 ДЕМОНСТРАЦІЯ ПРОЕКТУ ТА ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

3.1 Головна сторінка проекту

Веб-додаток має назву «PetsPaw» та має три основні вкладки:

– голосування;

– породи;

– галерея.

Ця сторінка не містить в собі якихось особливих функцій, окрім основних вкладок. На рис. 3.1 відображається ліва частину веб-сторінки. Логотип є клікабельним та повертає сторінку при натисканні до початкового стану. Всі вкладки мають однакову логіку по перенаправленні користувача на інший контент додатку, а також мають свою анімацію за принципами UI/UX, що підсвідомо дає користувачам зрозуміти яка сторінка наразі активна, на яку кнопку буде здіснено клік тощо.

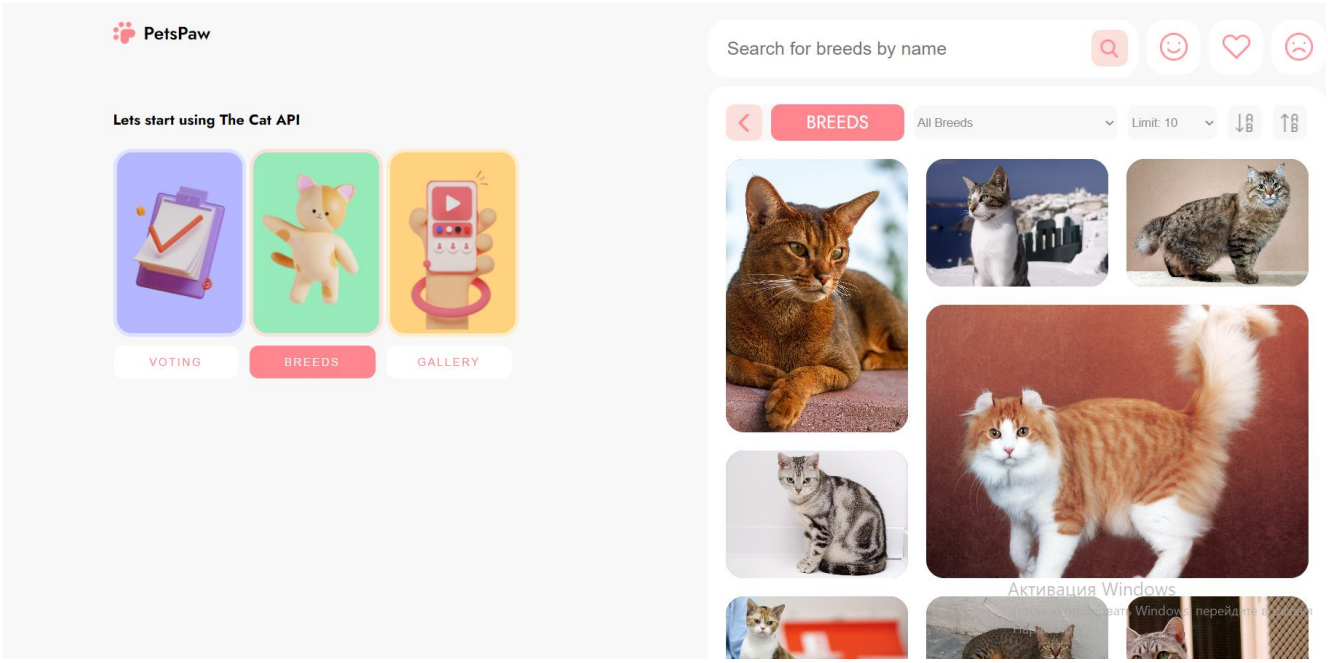

Рис. 3.1 – Основні вкладки

3.2 Сторінка «Голосування»

Сторінка «Голосування» (рис. 3.2) призначена для голосування (реакцію) на певне зображення, яке надходить з API. Користувач може обирати чи сподобалось

йому дане фото або не сподобалось. Також зображення можна додавати до списку «Обране».

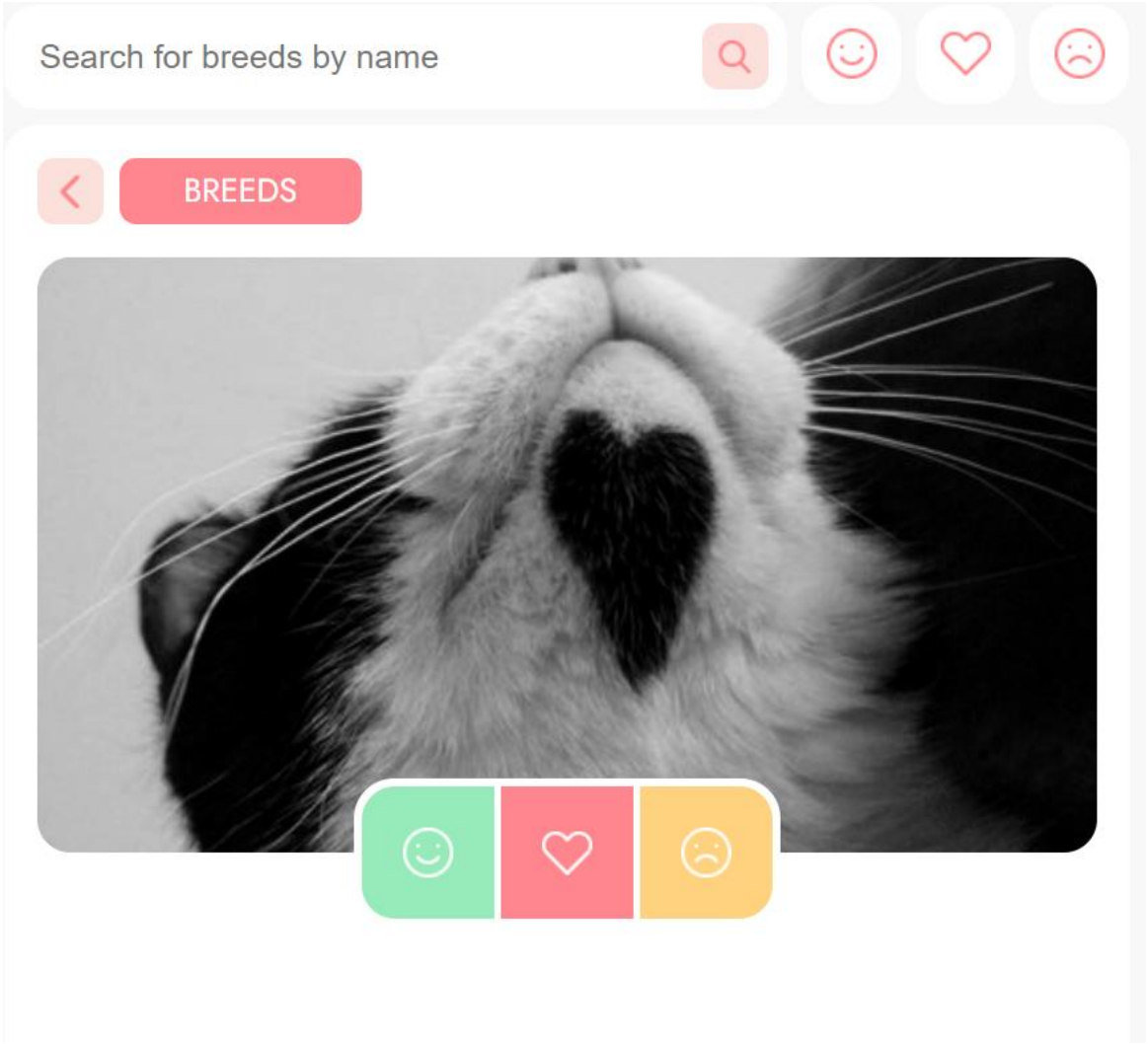

Рис. 3.2 – Сторінка «Голосування»

Сторінка містить історію дій користувача, де зазначається час, дія та id зображення, над яким відбулась дія (рис. 3.3). Також реалізовано певне повідомлення, якщо користувач ще не приймав участь в голосуванні (рис. 3.4).

Дані про вибір користувача відправляються до API, де вони оброблюються та зберігаються під індивідуальний id кожного користувача. Така логіка реалізовується за допомогою HTTP-методів. Це стандартний протокол у кожній мові програмування, завдяки якому можна здійснювати запити на будь-які дії з інформацією на сервері. А документація API пояснює як з цим працювати.

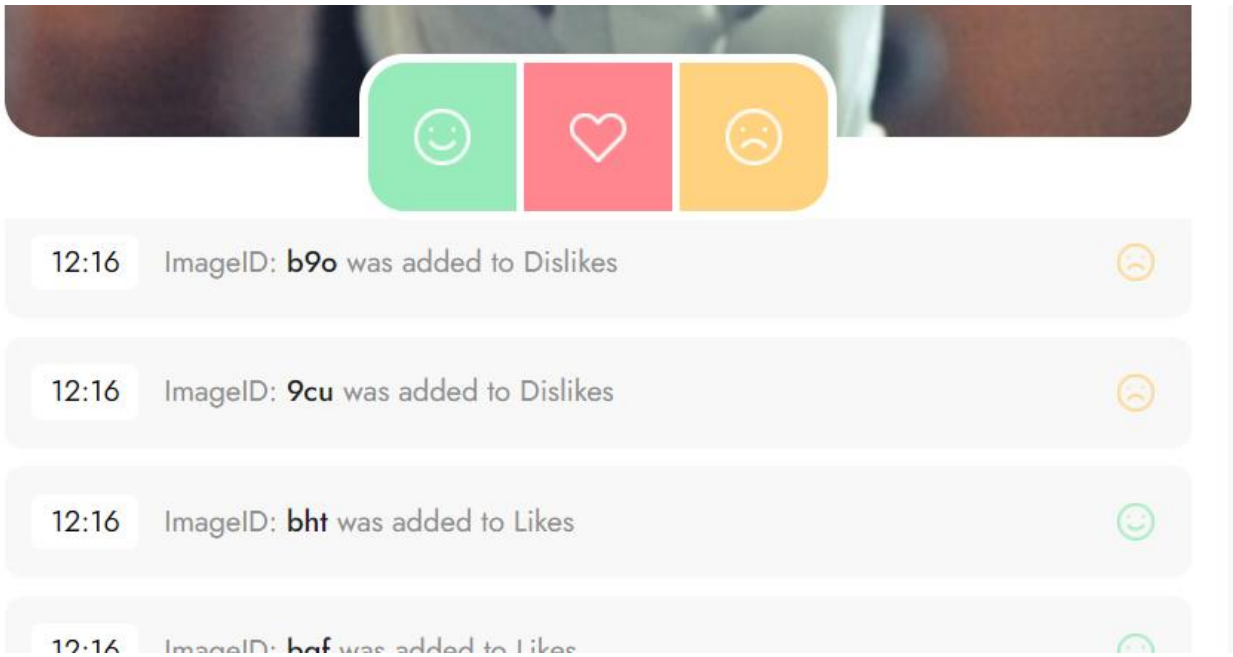

Рис. 3.3 – Історія дій на сторінці «Голосування»

History actions is empty.
Make new actions.

Рис. 3.4 – Відображення повідомлення в пустій історії подій

Отримання зображення з API (рис. 3.5) відбувається наступним чином:

1. У окремій компоненті під сторінку «Голосування» створюється зміна під унікальний код користувача.

2. За допомогою хука useState створюється змінні imgUrl та imgId через деструктурізацію для того, щоб при оновлені змінних вся компонента оновлювалась (відповідно й оновлюватиметься контент на веб-сторінці).

3. Створюються функція, в якій відбувається отримання даних з API та їх обробка.

4. Ця функція оновлюється завдяки хуку useEffect, який відслідковує зміни компоненти.

```
export default function Voting({ history, onVoteClick }) {
  const userId =
    "live_tjRhG76aZqVgTyDzKxFDZl4qGTeFx4IXrOemhE7D6IrsfY75X8QBC6THPXFa0MPe";
  const [imgId, setImgId] = useState();
  const [imgUrl, setImgUrl] = useState();
  const [loading, setLoading] = useState(true);

  async function fetchDataFromAPI() {
    try {
      const response = await fetch(
        "https://api.thecatapi.com/v1/images/search",
        {
          headers: { "x-api-key": userId },
        }
      );
      const data = await response.json();
      setImgId(data[0].id);
      setImgUrl(data[0].url);
    } catch (error) {
      console.error("There was a problem with the fetch operation:", error);
    }
    setLoading(false);
  }

  useEffect(() => fetchDataFromAPI, []);
```

Рис. 3.5 – Код за яким отримується зобаження для сторінки «Голосування»

Саме оновлення зображення після голосування користувача відбувається наступним чином (рис. 3.6):

1. На всі кнопки для голосування навішується функція, яка спрацьовує при кліці.
2. В функції handlerVote оновлюється історія подій.
3. Створюється об’єкт, що містить в собі дані для відправки.
4. По завершенню об’єкт відправляється до API через функцію fetch.
5. Компонента оновлюється для отримання нового зображення.

```
function handlerVote(value) {
    console.log(value)
    handlerHistoryOnClick(value);
    let url = "https://api.thecatapi.com/v1/votes";
    const body = {
        image_id: imgId,
        sub_id: "user-123",
        value: value,
    };

    if (value === "fav") {
        url = "https://api.thecatapi.com/v1/favourites";
        delete body.value;
    }

    fetch(url, {
        method: "POST",
        headers: { "Content-Type": "application/json", "x-api-key": userId },
        body: JSON.stringify(body),
    });

    setLoading(true);
    fetchDataFromAPI();
}
```

Рис. 3.6 – Код, що спрацьовує при голосуванні

3.3 Сторінка «Породи»

Сторінка «Породи» (рис. 3.7) призначена для знайомства користувача з різними породами улюбленців. На сторінці присутня фільтрація за породами, кількістю відображаємих зображень на сторінку, а також сортування по назвам.

При наведені курсора користувачем відображається кнопка з назвою порода. При кліці на цю кнопку користувачу відкривається сторінка з детальною інформацією про породу (рис. 3.8).

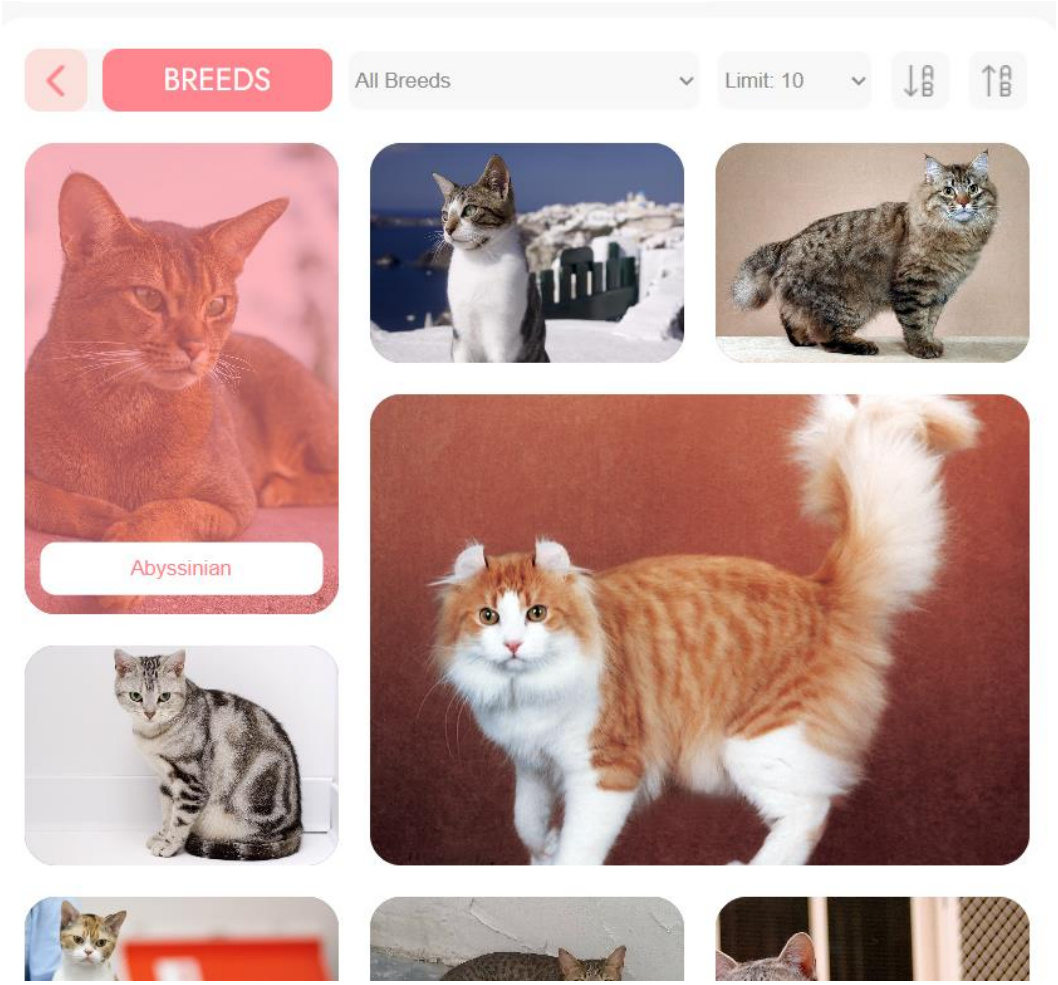

Рис. 3.7 – Сторінка «Породи»

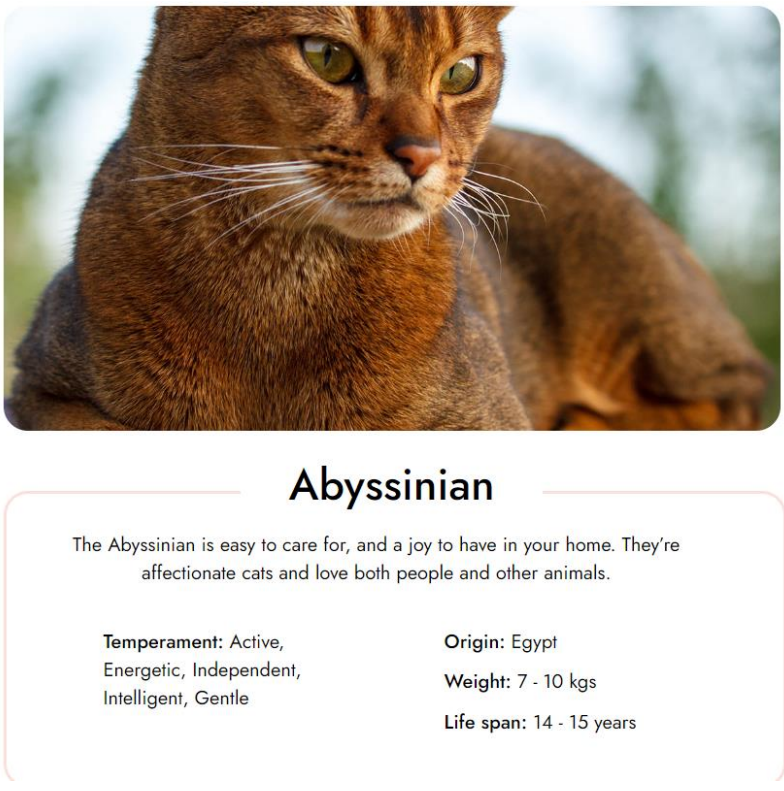

Рис. 3.8 – Сторінка з детальною інформацією про породу

Завдяки детальній інформації користувач може дізнатись про особливості кожної породи, характер, стандартні характеристики тіла та багато чого іншого, що в свою чергу допоможе краще зрозуміти кожну породу та убезпечить від хибних уявлень про той чи інший вид.

Завантаження даних все таким же чином відбувається як і в попередній сторінці. Надходить масив з об'єктами, в яких міститься вся необхідна інформація для реалізації. Після цього в функції breeds ці дані сортуються наступним чином (рис. 3.9):

1. В кожному об'єкті перевіряється чи має цей об'єкт зображення.
2. Елемент додається в внутрішній масив функції для тимчасового зберігання даних.
3. Далі дані з масиву arrAPI завантажуються до основного масиву компоненти, де в ньому зберігаються.

```
function breeds() {
  const arrForBreeds = [];
  let arrAPI = [];

  new Promise((resolve, reject) => {
    fetch("https://api.thecatapi.com/v1/breeds", {
      headers: { "x-api-key": userId },
    }).then((data) => resolve(data.json()));
  })
    .then((result) => {
      result.forEach((item) => {
        if (item.image) arrAPI.push(item);
      });
      return arrAPI;
    })
    .then((result) => {
      setAPIArr(result);
      result.forEach((catInfo) => {
        if (catInfo.id && catInfo.name) {
          arrForBreeds.push({ id: catInfo.id, name: catInfo.name });
        }

      });
      setBreeds(arrForBreeds);
    });
}

useEffect(() => breeds, []);
```

Рис. 3.9 – Код завантаження даних на сторінку «Породи»

Показ контенту на сторінці реалізується за допомогою наступного коду:

1. В змінну gridItems присвоюється результат методу map.
2. В методі створюється DOM-елементи, в яких розподіляється інформація з кожного елемента масиву.
3. До кожного елемента додається обробник подій при кліці на кнопку для відкриття детальної інформації про породу.
4. Змінна gridItems присвоюється до основної частини елементів сторінки.

```
if (scrollArray) {
  gridItems = scrollArray.map((cat, index) => (
    <div
      className={"grid-item gr-i-" + (index > 9 ? index - 10 : index)}
      key={index}
    >
      <img src={cat.image.url} alt="" />
      <div className="gr-i-hover"></div>
      <button
        className="gr-i-hbtn"
        onClick={() => {dispatch(changePage("breed-info")); dispatch(addInfoBreed(cat))} }
      >
        {cat.name}
      </button>
    </div>
  ));
}
```

Рис. 3.10 – Реалізація даних, що надійшли з API

Також сторінка має ще одну внутрішню компоненту, яка відповідає за реалізацію панель фільтрування (рис. 3.11). В ній є своя структура елементів, а основна частина логіки відповідає за відправку даних, якщо користувач змінив хоч одну категорію фільтрування.

Рис. 3.11 – Панель фільтрації сторінки «Породи»

3.4 Сторінка «Галерея»

Сторінка «Галерея» (рис. 3.12) містить унікальну можливість подивитись на різні зображення інших користувачів, проявити своє враження щодо кожного

зображення. Також цікавим функціоналом є можливість фільтрації зображень за різними критеріями. Це допоможе користувачам відображати лише цікавий для них контент, а також допоможе в пошуку цікавого зображення, обмежуючи коло пошуку.

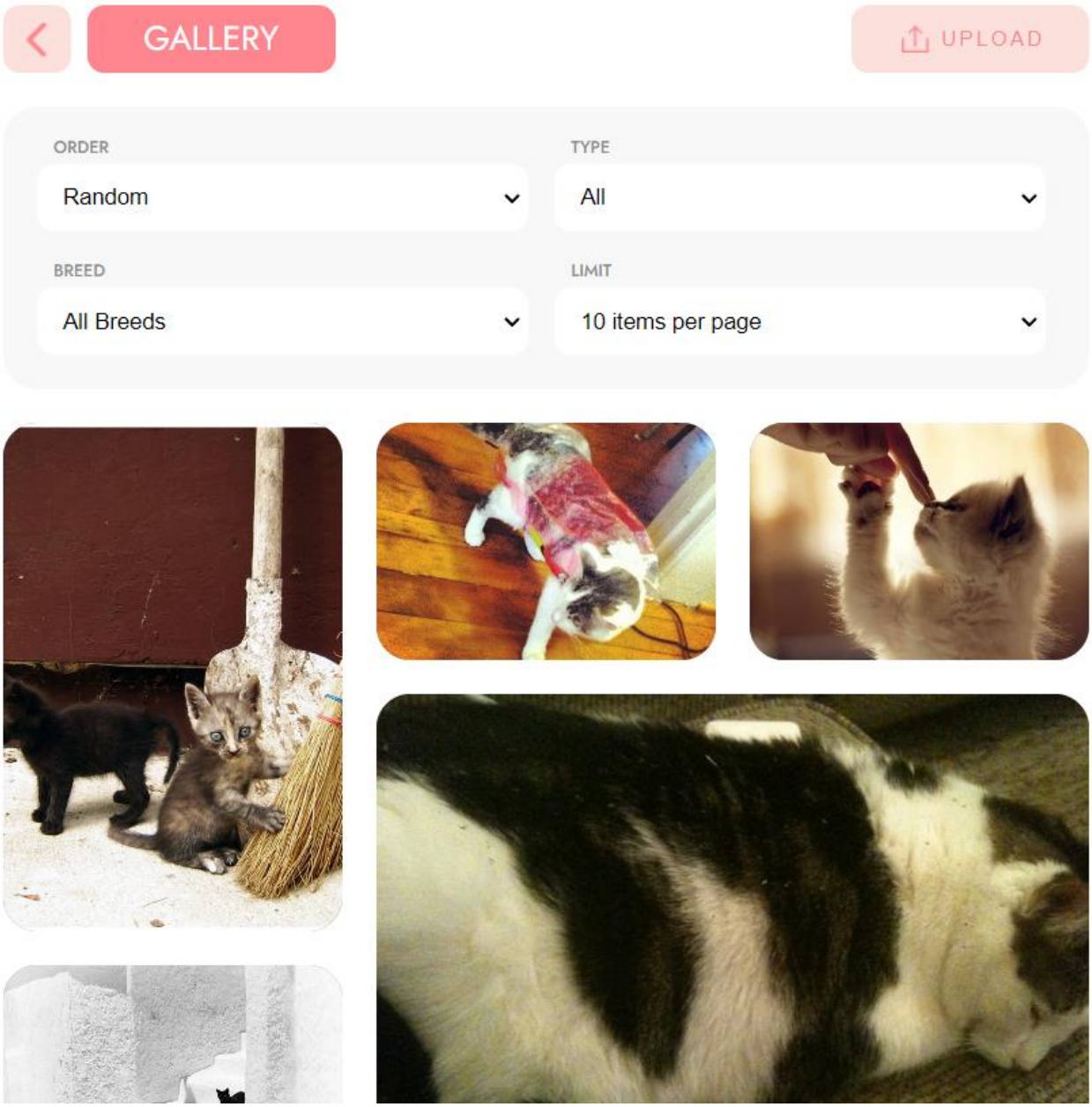

Рис. 3.12 – Сторінка «Галерея»

Реалізація в цієї сторінки дещо складніша, ніж в інших. По-перше, інформація таким самим чином завантажується з API (рис. 3.13), але разом з тим під час завантаження функція gallery відправляє запит до API на кожний елемент для того, щоб прослідкувати чи не знаходиться зображення у списку «Обране».

```
useEffect(
  () =>
    function gallery() {
      let arrAPI = [];
      if (counter >= historyGallery.length) {
        new Promise((resolve, reject) => {
          fetch(link, {
            headers: { "x-api-key": userId },
          }).then((data) => resolve(data.json()));
        }).then((result) => {
          const arrFaves = [];
          new Promise((resolveFaves, reject) => {
            fetch("https://api.thecatapi.com/v1/favourites", {
              method: "GET",
              headers: {
                "content-type": "application/json",
                "x-api-key": userId,
              },
            }).then((data) => resolveFaves(data.json()));
          })
            .then((resultFaves) => {
              result.forEach((item) => {
                item.isFav = resultFaves.some(
                  (itemFav) => itemFav.image.id === item.id
                );
                arrAPI.push(item);
              });
              resultFaves.forEach((itemFav) => {
                arrFaves.push({
                  imageId: itemFav.image.id,
                  favId: itemFav.id,
                });
              });
              return arrAPI;
            })
            .then((result) => {
              setAPIArr(result);
              setArrFaves(arrFaves);
              dispatch(addHistory([result]));
            });
        });
      }
```

Рис. 3.13 – Код реалізації даних сторінки «Галерея»

Така додаткова логіка необхідна для того, аби у веб-додатку при наведені користувачем (рис. 3.14) відмічалось чи є дане зображення у списку «Обране», адже тоді при натисканні на кнопку зображення буде видалятись зі списку, а не додаватись в нього.

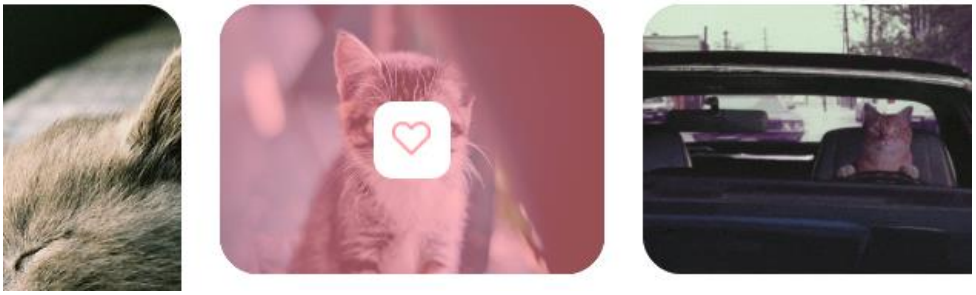

Рис. 3.14 – Результат наведення користувачем на зображення

Наступною досить складною логікою є реалізація логіки при додаванні/видаленні зображення до/з списку «Обране». На кожну кнопку, яка прив'язується до зображення навішується обробник handlerClickGalleryFav (рис. 3.15). В цю функцію передається подія, через яку витягується id зображення. Далі створюється об'єкт з необхідними даними для відправки.

За допомогою циклу, який проходиться по відображаємому масиву елементів, знаходиться потрібний елемент та змінюється його статус. Після цього необхідні дані відправляються до API для того, щоб внести зміни у списку «Обране».

```
function handlerClickGalleryFav(event) {
  //Function for adding fav cat by click on item
  let idFav = event.target.id;
  let bodyFav = JSON.stringify({
    image_id: idFav,
    sub_id: "user-123",
  });

  let newData = arrFromAPI.slice();
  console.log(newData);

  for (const value of newData) {
    if (value.id === idFav) {
      if (value.isFav) {
        const itemFav = arrFaves.find((item) => item.imageId === idFav);
        fetch(`https://api.thecatapi.com/v1/favourites/${itemFav.favId}`, {
          method: "DELETE",
          headers: {
            "content-type": "application/json",
            "x-api-key": userId,
          },
        });
      } else {
        fetch("https://api.thecatapi.com/v1/favourites", {
          method: "POST",
          headers: {
            "content-type": "application/json",
            "x-api-key": userId,
          },
          body: bodyFav,
        });
      }
      value.isFav = !value.isFav;
    }
  }
  setAPIArr(newData);
}
```

Рис. 3.15 – Функція для відправки даних при зміні списку «Обране»

У компоненті фільтрації реалізовується певна логіка, яка відповідає за передачу даних при зміні користувачем будь-якого критерія. Ці дані обробляються вже в основній компоненті, адже саме звідси відбувається вся підгрузка контенту з API.

При будь-якій зміні фільтру дані оброблюються функцією onChangeReauest (рис. 3.16), яка створює нове посилання для запиту зі зміненими параметрами. Далі за допомогою регулярних виразів до посилання додаються необхідні параметри, і вже після того оновлюється стан компоненти. Таким чином при оновленні відображається контент з встановленими параметрами користувачем.

```
function onChangeRequest(value, param) {            //Function for
  let newLink;
  setCounter(0);
  dispatch(clearHistory());

  if (link.includes(param) && param + value === "breed_idsall") {
    let regex = new RegExp(`(\\?|&)${param}=([^&]*)`);
    newLink = link.replace(regex, '');
    setLink(newLink);
  } else if (link.includes(param)) {
    if (!link.includes(param + value)) {
      let regex = new RegExp(`(\\?|&)${param}=([^&]*)`);
      newLink = link.replace(regex, `$1${param}=${value}`);
      setLink(newLink);
    }
  } else {
    newLink = link + `&${param}=${value}`;
    setLink(newLink);
  }
}
```

Рис. 3.16 – Код функції для зміни контенту при фільтрації на сторінці «Галерея»

3.5 Сторінка «Лайки»

Ця сторінка відображає контент, який раніше був вподобаним користувачем та відображає повний список зображень за весь час активності користувача у веб-додатку (рис. 3.7). Завдяки цій вкладці користувач може з легкістю знайти контент, який його зацікавив раніше.

Особливого функціоналу ця сторінка не містить. Лише реалізована підгрузка даних та перемикання між сторінками, але в будь-якому випадку до сторінки можна додавати цікавий функціонал. Вона гнучка та адаптивна до будь-яких змін.

Таким же чином реалізовані й сторінки «Дизлайки» та «Обране.»

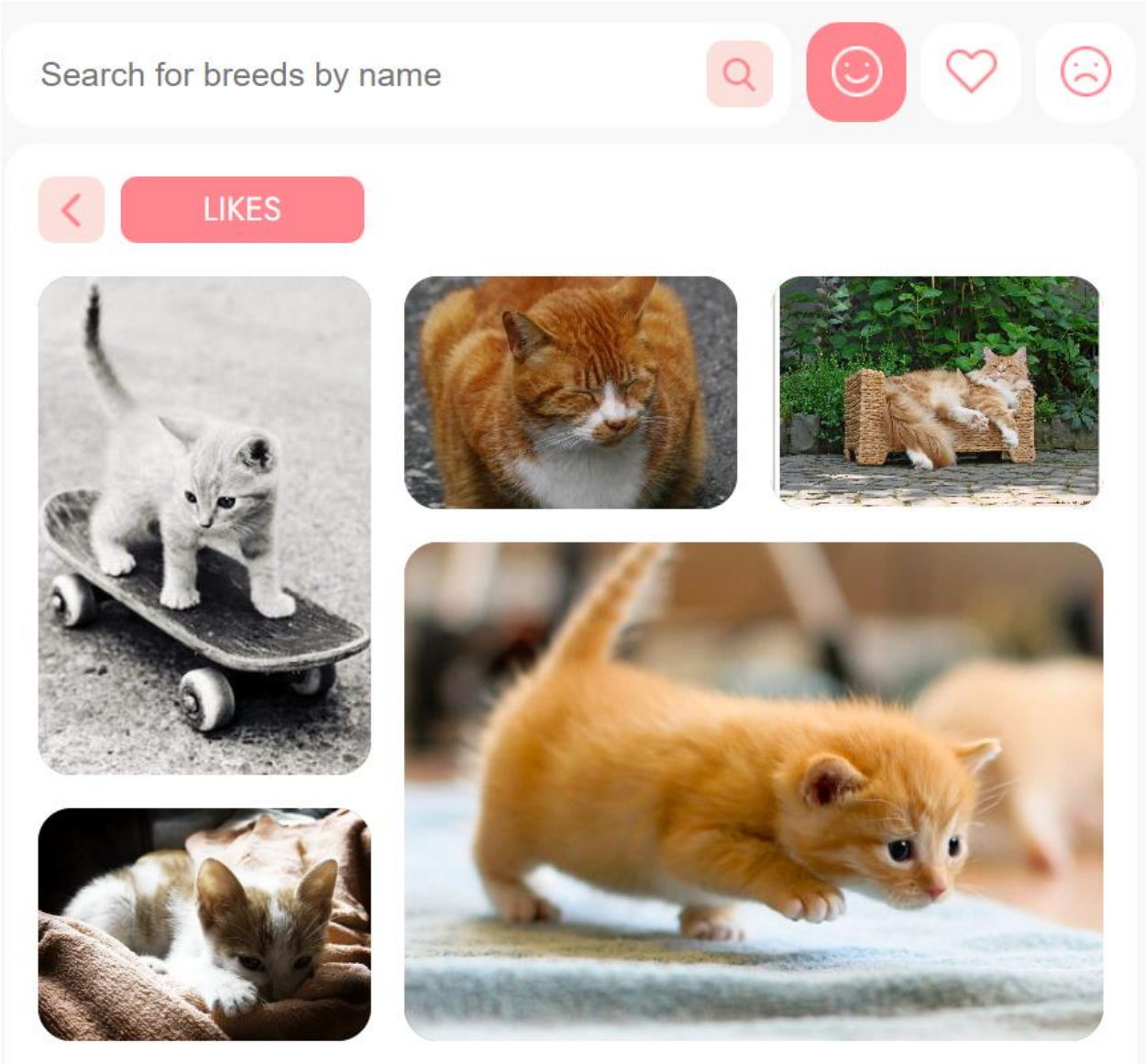

Рис. 3.17 – Сторінка «Лайки»

3.6 Панель «Навігація»

Це головна панель для навігації користувача, містить в собі кнопки для переходу на сторінки «Лайки», «Дизлайки», «Особливе», а також має рядок пошуку зображень за породами (рис. 3.18). Функціонал пошуку дозволяє відфільтрувати потрібну породу та знайти необхідні зображення певної породи, що значно полегшує пошук необхідного контенту.

Панель навігації містить в собі анімації для покращення досвіду користувача при використання веб-додатку, а швидка оптимізація не буде обтяжувати користувачів довгим часом очікування при завантаженні.

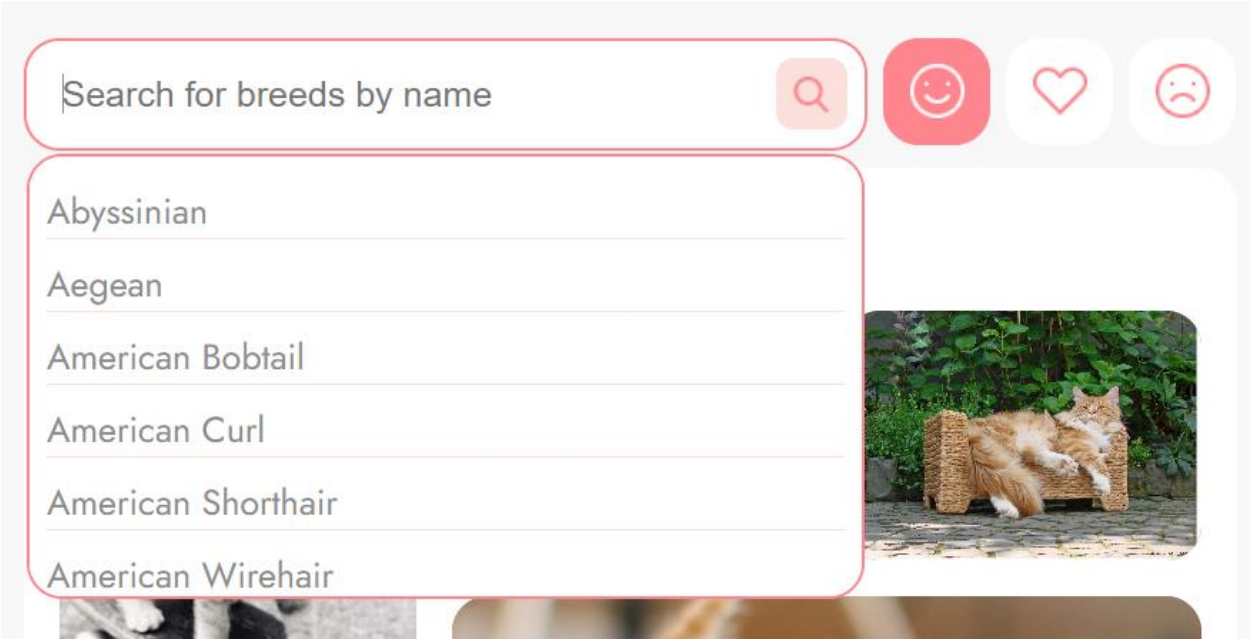

Рис. 3.18 – Панель «Навігації»

Панель містить в собі доволі цікаву логіку. За допомогою хука useEffect (рис. 3.19), який відстежує будь-які зміни в рядку пошуку, компонента автоматично оновлюється разом з результатами пошуку. Такий підхід надає можливість реалізувати динамічність пошуку та додати цікавості.

Функції handlerFocusInput та handlerBlurInput створені для того, щоб виводити на сторінці та скривати результати пошуку під час дій користувача. Коли користувач натискає на рядок, то вікно з'являється, аналогічно й при виході з панелі. На жаль, засоби CSS не містять достатнього функціоналу, щоб реалізовувати такий підхід, тому довелось створювати JavaScript-код для реалізації цієї задумки. Дані фунції навішуються на input для прослуховування подій onFocus та onBlur відповідно.

Елементи в результатах пошуку відображаються завдяки методу map, який фільтрує список порід відповідно того, що вводить користувач. Варто зазначити, що на кожен елемент результату навішується обробник при кліці, який і направляє користувача на відповідну галерею з зображеннями тієї породи, яка була обрана.

```
function NavigationPanel() {
  const dispatch = useDispatch();
  const breedsArr = useSelector(selectBreedsArray);
  const [searchResult, setSearchResult] = useState([]);

  useEffect(() => { setSearchResult(breedsArr); console.log("yes"); }, [breedsArr]);
  console.log(searchResult)

  function handlerClickSearchResult(idBreed) {
    dispatch(changePage('search'));
    dispatch(changeSearchRequest(idBreed));
  }

  function handlerSearch(event) {
    const result = breedsArr.filter(item => item.name.toLowerCase().includes(event.target.value.toLowerCase()));
    setSearchResult(result);
  }

  const searchResultList = searchResult.map((item) => (
    <li key={item.id} onClick={(e) => { handlerClickSearchResult(item.id); handlerClickNavBtns(e)}}>
      {item.name}
    </li>
  ));

  function handlerFocusInput() {
    const searchInput = document.querySelector('.inp-box');
    const resultList = document.querySelector(".search-results");
    searchInput.classList.add('inp-box-f');
    resultList.style.visibility = 'visible';
  }

  function handlerBlurInput(e) {
    const searchInput = document.querySelector(".inp-box");
    setTimeout(function () {
      const resultList = document.querySelector(".search-results");
      resultList.style.visibility = "hidden";
      e.target.textContent = '';
    }, 1000);
    searchInput.classList.remove("inp-box-f");
```

Рис. 3.19 – Код функціоналу Навігаційної панелі

3.7 Сторінка завантаження користувачем зображення

Дана сторінка включає в себе модульне вікно, яке пропонує користувача перенести зображення до відповідної зони або відкрити менеджер файлів та обрати потрібне зображення.

Якщо зображення проходить перевірку, то з’являється кнопка при натисканні якої, користувач додає зображення в базу (рис. 3.20). Якщо програма не розпізнає на зображенні потрібний об’єкт, то виводить помилку та пропонує користувачу спробувати завантажити інше зображення (рис. 3.21).

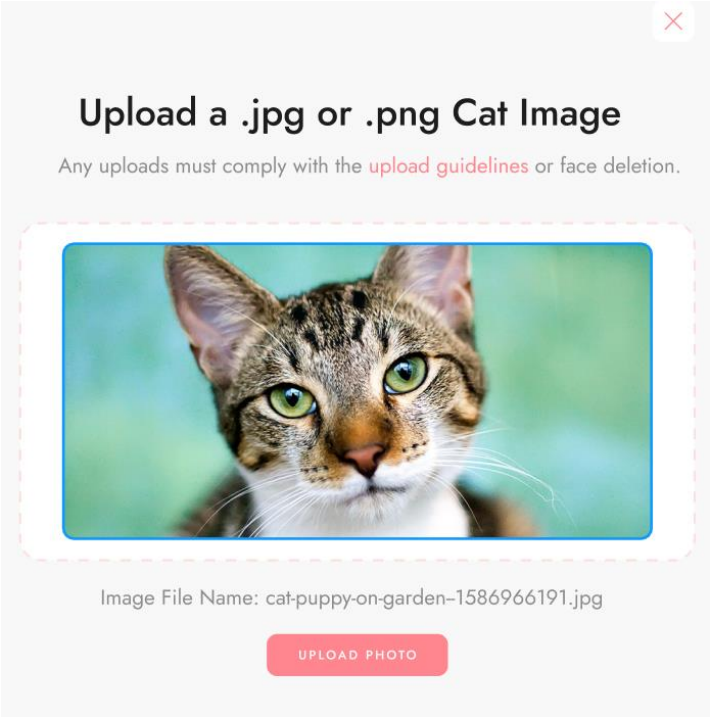

Рис. 3.20 – Модальне вікно для завантаження зображень

Рис. 3.20 – Модальне вікно при невдалому завантаженні зображень

Така логіка відбувається завдяки використанню модель Google MobileNetV2. Для того, щоб працювати з моделлю було встановлено бібліотеку TensorFlow.js, на основі якої і написана модель. За стандартом MobileNetV2 була навчена на наборі

даних ImageNet, що не є підходящим рішенням для даного веб-додатку та функцій, які потрібні. Тому було створено власний набір даних за яким була навчена модель, цей набір даних наразі підлаштований лише під класифікацію котів.

ВИСНОВКИ

В ході дипломної роботи було проведено аналіз сучасних веб-технологій та історії зародження в цілому. Були встановлені можливі проблеми та ризики, що можуть вплинути на створення веб-застосунку.

Також в ході роботи було проведено аналіз поточної ситуації всіх можливих технологій, як вони впливають на ринок праці, культуру створення веб-додатків та поточну ситуацію у світі.

Додатково було досліджено сферу соціальних мереж чи будь-яких подібних додатків/веб-додатків, які об'єднують власників тварин (особливо екзотичних чи специфічних) і допомагають обмінюватись знаннями та досвідом щодо їх утримання.

В ході дипломної роботи було розроблено соціальну мережу, використовуючи сучасні веб-технології, такі як фреймворк React.js для ефективнішої розробки веб-додатку, мову розмітки HTML, стилізацію контенту за допомогою CSS, мову програмування JavaScript для розробки функціоналу додатку і бібліотеку Redux, що допоможе керувати станом додатку.

Також в ході дослідження було проведено аналіз актуальних технологій для класифікації зображень. Було обрано модель Google MobileNetV2 для даної задачі, завдяки чому вдалось інтегрувати можливість аналізу та допуску зображень при їх завантаженні до бази через веб-застосунок.

Цей проект демонструє можливості створення високоякісного і масштабованого веб-додатку з інтуїтивно зрозумілим інтерфейсом та високою продуктивністю.

Основні результати та досягнення проекту включають:

1. Створення компонентного інтерфейсу
2. Ефективне управління станом
3. Привабливий і функціональний інтерфейс
4. Динамічність та інтерактивність
5. Забезпечення надійності та масштабованості

Успішне завершення даного проекту підтвердило ефективність використання React.js та пов'язаних технологій для розробки складних веб-додатків. Досвід, отриманий під час розробки, надає цінні знання та навички, які можуть бути використані в майбутніх проектах і професійній діяльності.

Розроблена соціальна мережа відповідає сучасним вимогам до веб-додатків, демонструючи високу продуктивність, надійність та зручність використання. Цей проект може бути основою для подальшого розвитку та вдосконалення, а також для розробки нових функціональностей, що сприятиме підвищенню його конкурентоспроможності на ринку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Banks A., Porcello E. Learning React: Functional Web Development with React and Redux. O'Reilly Media, 2017. 350 p. (дата звернення: 07.03.2024).

2. Crockford D. JavaScript: The good parts. Sebastopol, CA : O'Reilly, 2008. 153 p. (дата звернення: 05.03.2024).

3. Getting Started with React Redux | React Redux. *React Redux | React Redux*. URL: https://react-redux.js.org/introduction/getting-started (дата звернення: 20.03.2024).

4. Introduction to TensorFlow. *TensorFlow*.URL: https://www.tensorflow.org/learn (дата звернення: 08.03.2024).

5. MacDonald M. HTML5: The Missing Manual. O'Reilly Media, Incorporated, 2013 (дата звернення: 02.03.2024).

6. Meyer E. A., Weyl E. CSS: The Definitive Guide. 4th ed. O'Reilly Media, 2017. 1090 p (дата звернення: 04.03.2024).

7. Quick Start – React. *React*. URL: https://react.dev/learn (дата звернення: 15.03.2024).

8. tf.keras.applications.MobileNetV2 | TensorFlow v2.16.1. *TensorFlow*. URL: https://www.tensorflow.org/api_docs/python/tf/keras/applications/MobileNetV2 (дата звернення: 10.03.2024).

9. Бровінська М. Що в українського IT забрала війна: люди, гроші, можливості. *Війна*. URL: https://dev.ua/news/aiti-1677216528 (дата звернення: 12.03.2024).

10. Завантажити Macromedia Dreamweaver для Windows 7 (32/64 bit) Українською. *Каталог програмного забезпечення для Windows 7 (32/64 bit) Українською*. URL: https://uk.all7soft.com/macromedia-dreamweaver-windows-7/ (дата звернення: 04.03.2024).

11. Лебединченко К.О., Піонтківський Є.Р., Фесенко М.А., Використання композиційного візуального мислення без тренувань в області різноманітних маніпуляцій з зображеннями / Науково-практична конференція «Проблеми комп’ютерної інженерії». К.: ДУІКТ, 2023 р. с. 34,35

12. Леся. Vue vs React у 2022 році - який фреймворк вибрати - Wezom. *IT-компания полного цикла разработки программных продуктов WEZOM - Киев, Украина*. URL: https://wezom.com.ua/ua/blog/vue-vs-react (дата звернення: 02.03.2024).

13. Мітній І.С. Схематичне програмування. Київ : НПУ ім. М. П. Драгоман., 2010. 147 с (дата звернення: 03.03.2024).

14. Фесенко М.А., Лебединченко К.О., Піонтківський Є.Р. ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В МАРКЕТИНГУ / Науково-практична конференція «Сучасні досягнення компанії HEWLETT PACKARD ENTERPRISE в галузі ІТ та нові можливості їх вивчення і застосування». Збірник тез. – К.: ДУІКТ, 2023р., с. 39,40

15. Фесенко М.А., Лебединченко К.О., Піонтківський Є.Р. ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ У ВІЙСЬКОВІЙ СФЕРІ В УКРАЇНІ / Науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях ». Збірник тез. – К.: ДУІКТ, 2024р., с. 265,266

16. Фесенко М.А., Лебединченко К.О., Піонтківський Є.Р. ВИКОРИСТАННЯ БІБЛІОТЕКИ TENSORFLOW.JS ПРИ РОЗРОБЦІ ВЕБ-ДОДАТКУ/ Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. – К.: ДУІКТ, 2024р.,

17. Що таке backend-розробка і чим вона відрізняється від frontend. *Б24*. URL: https://www.buh24.com.ua/shho-take-backend-rozrobka-i-chym-vona-vidriznyayetsya-vid-frontend/ (дата звернення: 06.03.2024).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Кваліфікаційна робота
на тему: «Розроблення веб-
додатку соціальної мережі за допомогою React.js
з інтелектуальною системою розпізнавання зображень»

Виконала: здобувачка вищої освіти гр. ШІД-41
Карина ЛЕБЕДИНЧЕНКО
Керівник: кандидат технічних наук, доцент
Максим ФЕСЕНКО

2024 рік

Мета роботи: підвищення ефективності інформування та обізнаності людей в питаннях утримання домашніх улюбленців.

Об'єкт дослідження: процес набуття корисних знань без суворих та нав'язливих методів навчання.

Предмет дослідження: система соціальної мережі.

Основні завдання кваліфікаційної роботи

1. Проаналізувати основні веб-технологій для створення веб-застосунку.
2. Проаналізувати поточний ринок подібних веб-застосунків.
3. Визначитись з використанням технологій.
4. Спроектувати веб-додаток.
5. Розробити веб-додаток з урахування всіх попередніх кроків.

АКТУАЛЬНІ FRONTEND ТЕХНОЛОГІЇ

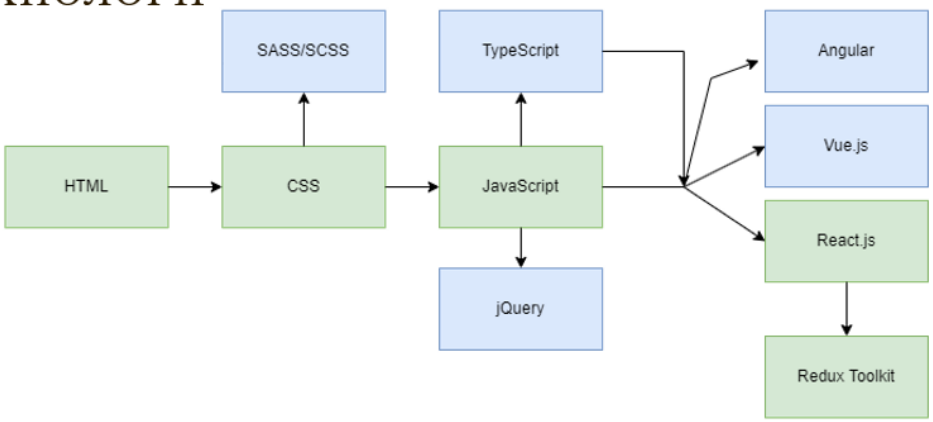

Рис. 1.1 – Актуальні технології

ДЕМОНСТРАЦІЯ ПРОЕКТУ

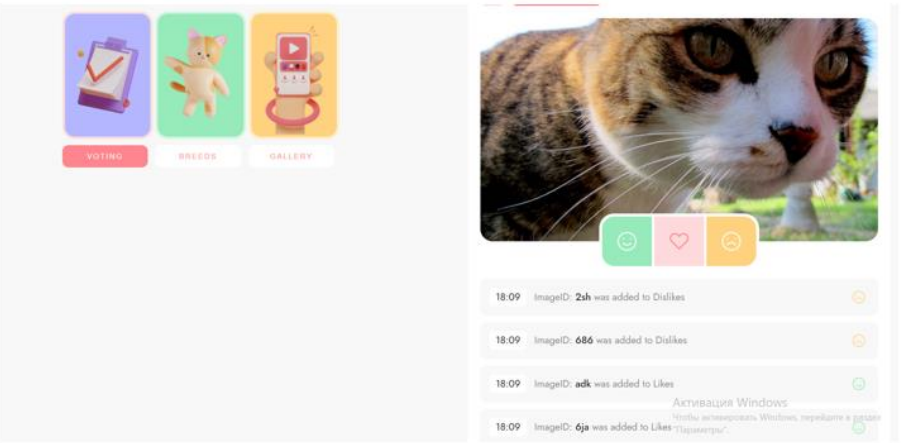

Рис. 1.2 – Сторінка «Голосування»

ДЕМОНСТРАЦІЯ ПРОЕКТУ

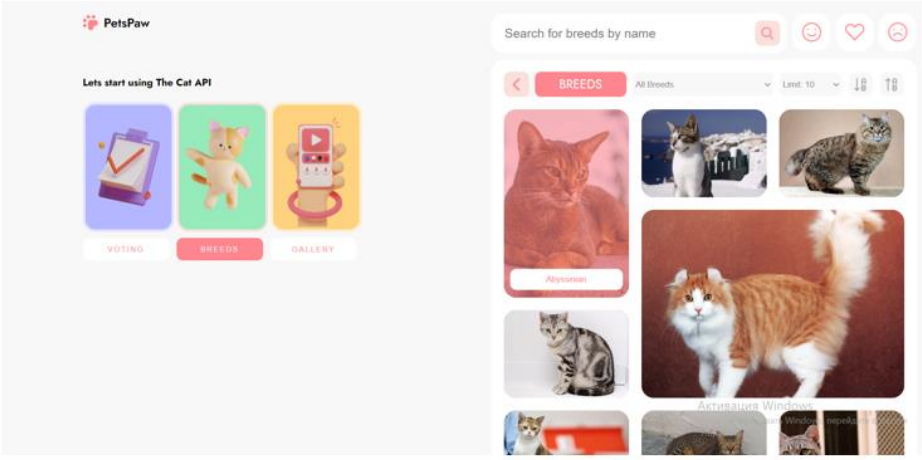

Рис. 1.3 – Сторінка «Породи»

ДЕМОНСТРАЦІЯ ПРОЕКТУ

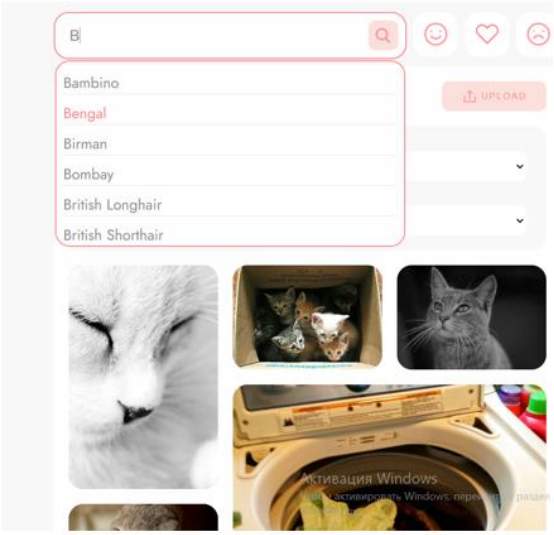

Рис. 1.4 – Реалізація пошуку

ВИКОРИСТАНІ ТЕХНОЛОГІЇ

1. HTML
2. CSS
3. JavaScript
4. JSX
5. React.js
6. Redux
7. MobileNetV2
8. TensorFlow.js

Висновки

В ході дипломної роботи було проведено аналіз сучасних веб-технологій та історії зародження в цілому. Були встановлені можливі проблеми та ризики, що можуть вплинути на створення веб-застосунку.

Також в ході роботи було проведено аналіз поточної ситуації всіх можливих технологій, як вони впливають на ринок праці, культуру створення веб-додатків та поточну ситуацію у світі.

Було розроблено соціальну мережу, використовуючи сучасні веб-технології, такі як фреймворк React.js для ефективнішої розробки веб-додатку, мову розмітки HTML, стилізацію контенту за допомогою CSS, мову програмування JavaScript для розробки функціоналу додатку і бібліотеку Redux, що допоможе керувати станом додатку.