Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ БЛАГОДІЙНОЇ ДОПОМОГИ БЕЗПРИТУЛЬНИМ ТВАРИНАМ З ВИКОРИСТАННЯМ REACT, TYPESCRIPT ТА REDUX TOOLKIT

DEVELOPMENT OF A WEB APPLICATION FOR CHARITY ASSISTANCE TO HOMELESS ANIMALS USING REACT, TYPESCRIPT AND REDUX TOOLKIT

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗз-41
Рудика Андрій Степанович

Керівник:
Суринович Олена Миколаївна

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«20» 12 2024р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Рудиці Андрію Степановичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка веб-додатку для благодійної допомоги безпритульним тваринам з використанням React, TypeScript та Redux Toolkit»*
Керівник роботи: *к.т.н., доцент Суринович О. М.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *мови програмування: TypeScript та JavaScript, система стилізації: TailwindCSS, фреймворк: React, система керування станом: Redux Toolkit, середовище розробки: Visual Studio Code, методичні вказівки до виконання кваліфікаційної роботи бакалавра*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз предметної області, формування вимог до розробки веб-додатку, розгляд технологій для створення веб-додатку, розробка функціоналу перегляду тварин з можливістю фільтрації, реалізація інтуїтивної навігації, створення сторінки з інформацією про притулок*

5. Перелік графічного матеріалу: *12 рисунків*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Суринович О. М*		
Специфікація вимог до розробленої системи	*Суринович О. М*		
Розробка об'єкта проектування	*Суринович О. М*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	1,51 %		
Академічна доброчесність	*Суринович О. М*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	Вик.
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	Вик.
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	Вик.
4	Розробка функціонально-структурної схеми роботи об'єкта проектування	до 29.03.2025 р.	Вик.
5	Практична реалізація об'єкта проектування	до 26.04.2025 р.	Вик.
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	Вик.
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	Вик.

Здобувач вищої освіти _____ Андрій РУДИКА

Керівник кваліфікаційної роботи _____ Олена СУРИНОВИЧ

АНОТАЦІЯ

Рудика А. С. Розробка веб-додатку для благодійної допомоги безпритульним тваринам з використанням React, TypeScript та Redux Toolkit. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел.

У першому розділі здійснено здійснено аналіз сучасного стану проблеми, порівняльний аналіз проектів що вирішують проблему і сформульовано завдання. В другому розділі були сформовані вимоги до розроблювальної системи та визначено інструменти, які найкраще підходять для цього. У третьому розділі продемонстровано практичну реалізацію системи об’єктування та проведення тестування готового продукту. У висновках описано досягнення поставлених цілей кваліфікаційної роботи бакалавра.

Ключові слова: веб-додаток, тварини, алгоритм, всиновлення, функціонал.

ABSTRACT

Rudyka A. Development of a web application for charity assistance to homeless animals using React, TypeScript and Redux Toolkit. Manuscript. Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions, a list of references.

The first chapter presents analysis of the current state of the problem, comparative analysis of projects that solve the problem and formulates the objectives. The second chapter requirements for the development system were formed and the tools that are best suited for this were identified. The third chapter demonstrated the practical implementation of the objectification system and testing of the finished product. The conclusions summarize the achievement of the set goal of the bachelor’s qualification work.

Keywords: web application, animals, algorithm, adoption, functionality.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі складно уявити життя без Інтернету. Щодня мільярди людей користуються веб-ресурсами з різноманітними цілями – від отримання інформації до здійснення покупок. Стрімкий розвиток цифрових технологій створив можливість розв’язання не лише комерційних, а й соціально важливих задач. Особливої уваги заслуговують веб-додатки, що спрямовані на допомогу навколишньому середовищу, людям та безпритульним тваринам.

На сьогоднішній день у світі існує велика кількість тварин, які потребують опіки, лікування, житла чи нового господаря. Благодійні організації та волонтери активно створюють веб-додатки, які мають на меті допомогти цим тваринам.

Однак, більшість з них мають застарілий або незручний інтерфейс, що ускладнює взаємодію користувача з ресурсом. Саме тому створення зручного, функціонального та сучасного веб-додатку зосередженого на допомозі тваринам є актуальним завданням.

Розробка подібної платформи дозволить:

– ефективно організовувати збори коштів на потреби тварин;

– забезпечити швидкий доступ до інформації про тварин, які потребують допомоги або шукають дім;

– інформувати населення щодо гуманного ставлення до тварин;

– сприяти об’єднанню небайдужих людей навколо вирішення проблем безпритульних тварин.

Таким чином, метою кваліфікаційної роботи є створення веб-додатку для благодійної допомоги безпритульним тваринам, який дозволить користувачам переглядати список доступних для всиновлення тварин, використовувати зручну фільтрацію та здійснювати інші взаємодії із платформою.

Об’єктом кваліфікаційної роботи є процес створення інформаційних систем для підтримки діяльності благодійних організацій.

Предметом кваліфікаційною роботи є веб-додаток для всиновлення та підтримки безпритульних тварин, зокрема її функціональні можливості, інтерфейс користувача та засоби реалізації.

Для досягнення поставленої мети кваліфікаційної роботи були визначені такі завдання:

– проаналізувати існуючі веб-додатки, спрямовані на допомогу тваринам, та виявити їхні сильні і слабкі сторони;

– розглянути сучасні технології та інструменти, які підходять для створення веб-додатку;

– реалізувати функціонал перегляду тварин з можливістю фільтрації за категоріями;

– створити сторінку з інформацією про притулок, з яким пов’язаний веб-додаток;

– розробити інтуїтивно зрозумілу та приємну навігацію між сторінками веб-додатку.

РОЗДІЛ 1
АНАЛІЗ ВЕБ-ДОДАТКІВ З ВСИНОВЛЕННЯ ТА ДОПОМОГИ БЕЗПРИТУЛЬНИМ ТВАРИНАМ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Проблема безхатніх тварин появилась давно, ще до появи інтернету. Люди клеїли оголошення, знаходили найближчі притулки через сарафанне радіо або близьких людей які володіли цією інформацією, або просто ігнорували цю проблему. Проте, теперішні технології допомогли зробити крок вперед, оскільки тепер людина завжди може оволодіти інформацією про притулки, достатньо їй цього захотіти.

Але, не дивлячись на це, дуже мала кількість людей приділяє роботу над цим. Наприклад, якщо переглянути веб-додатки які формують себе як веб-додатки для допомогти безпритульним тваринам, більшість з них буде неприємна для користувача, або навпаки – недостатньо функціональна.

Веб-додаток для всиновлення тварин повинен бути приємним для користувача, містити сучасні тенденції дизайну, оскільки який би не був функціонал на веб-додаток – зовнішність в 90 відсотках відіграє основну роль.

В чому ще проявляється велика проблема таких веб-додатків? Їх просто мало. Яким би не був великий притулок для всиновлення – рано чи пізно він буде повністю завантажений, і не зможе приймати нових тварин. Після цього появляється питання – а що буде, коли всі притулки будуть повністю завантажені?

Якщо таке трапиться, це свідчить про наступну проблему – притулків просто недостатньо. Але навіть якщо поглянути на проблему з іншої сторони, коли притулки неперевантажені – ми в будь-якому випадку можемо побачити безпритульних тварин на вулиці. Чому так трапляється? Тому що людей, які цим займаються недостатньо. Отже, аби подолати відразу дві проблеми, ми маємо

просте рішення – створити ще більше притулків, і стимулювати врятування безпритульних тварин таким чином.

Але, які переваги від цього людям? На вулицях з'являється все більше і більше безпритульних тварин, які формують зграї, а потім починають шукати собі жертв. Інколи, це інші тварини, а інколи – це самі люди. Наприклад, в Україні ми можемо побачити збільшення кількості людей, яких кусають безпритульні тварини [1]. І, здається, що поки тебе ця проблема не стосується – все добре, проте достатньо уявити ситуацію, як твоя дитина побігла гуляти у двір, а додому прийшла вся заплакана, оскільки її налякала зграя диких собак, або ще гірше – ця зграя напала на неї.

Сотні тисяч людей стосуються цієї проблеми. Деякі навіть віддають все своє життя, аби подолати цю проблему. Відомі зірки завжди раді зробити дарунки для притулків, щоб допомогти тваринкам [2]. Якщо говорити про Україну – то тут все складніше. Приходиться залучати безліч інвесторів, аби притулок функціонував повністю. Проте, все більше і більше почали залучати владу, тому що проблема безпритульних тваринах зростає все більше. Тобто, люди бачать проблему безпритульності, і розуміють, наскільки це актуально.

На щастя, активісти, волонтери та небайдужі люди об'єднуються, аби вирішити цю проблему. Вони організовують благодійні заходи, шукають нові домівки для тваринок, проводять інформаційні кампанії серед населення, що значно допомагає вирішити проблему.

Варто пам'ятати, що якщо ти робиш щось волонтерське – ти не повинен думати про копійку собі. Я мрію зробити великий вклад у вирішення проблеми безпритульності тварин своїм веб-додатком, тому цей проект буде повністю волонтерським. Після створення веб-додатку буде проведено переговори з притулками, які будуть відкриватись, та передача цього веб-додатку їм. Тепер проведемо аналіз вже наявних веб-додатків, та виявимо переваги та недоліки.

Одним з популярних веб-додатків є Petfinder – веб-додаток для всиновлення тварин в США, який ми можемо побачити на рисунку 1.1.

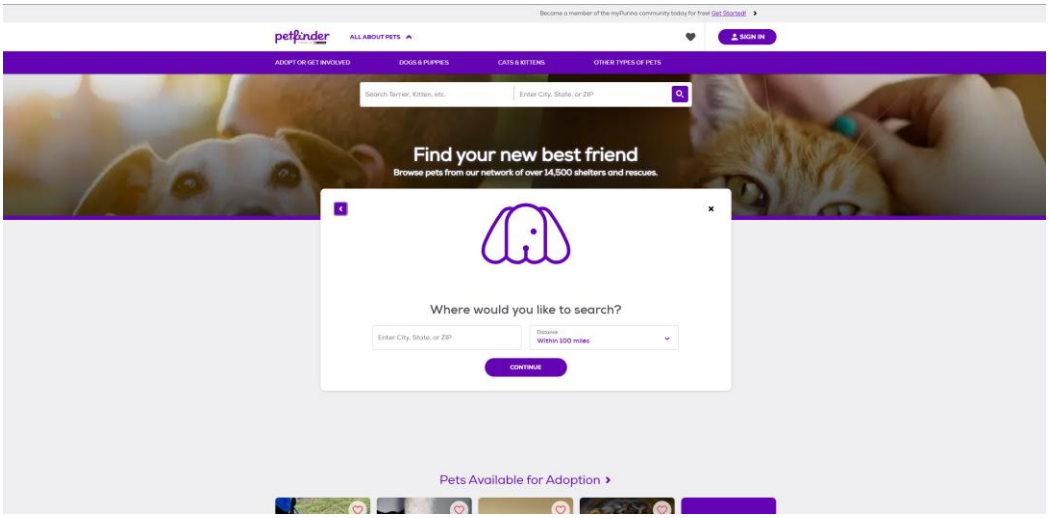

Рисунок 1.1 – Головна сторінка веб-додатку Petfinder [3]

Petfinder має безліч плюсів – наприклад, ми можемо побачити найближчі пункти притулків та інформацію про тварин. Але якщо ми говоримо про мінуси – це дизайн. Веб-додаток виглядає занадто пустим, а коли ми переходимо на сторінку з описом тварин – шрифт дуже поганий для прочитання, він дуже маленький, а інформація про тварину не є «цікавою», що штовхає від всиновлення. Також велика проблема в тому, що веб-додаток врахований лише для аудиторії США, що робить неможливим всиновлення за межами країни.

Наступний веб-додаток для аналізу – це український Dogcat, найпопулярніший веб-додаток по всиновленню тварин в Україні, який ми можемо побачити на рисунку 1.2.

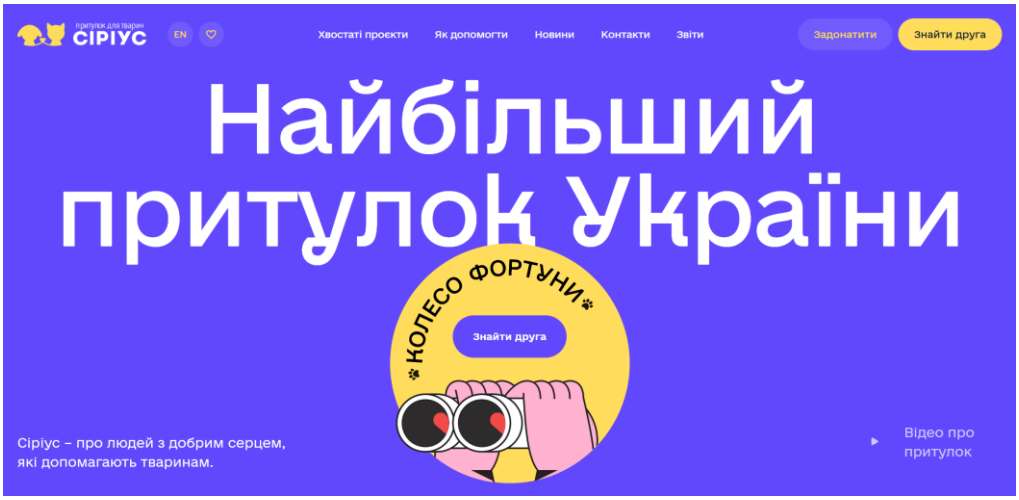

Рисунок 1.2 – Головна сторінка веб-додатку Dogcat [4]

Dogcat – це веб-додаток, на який повинні рівнятись інші веб-додатки які базують себе на всиновленні. Тут ми можемо побачити прекрасний інтерактивний список для всиновлення, чудовий дизайн, зворотній зв’язок, повну інформацію про тварину, та найголовніше – можливість фільтрації за типом тварини, розміром, та особливостями. Має повну адаптацію, під будь-який розмір екрану. Із мінусів – «занадто великий» тип дизайну, адже деякі компоненти було можливо зробити в рази меншими. Також деякі кнопки абсолютно не функціонують, тому лише залишається питання – навіщо вони у веб-додатку.

Далі розглянемо ще один український веб-додаток Happypaw, який ми можемо побачити на рисунку 1.3.

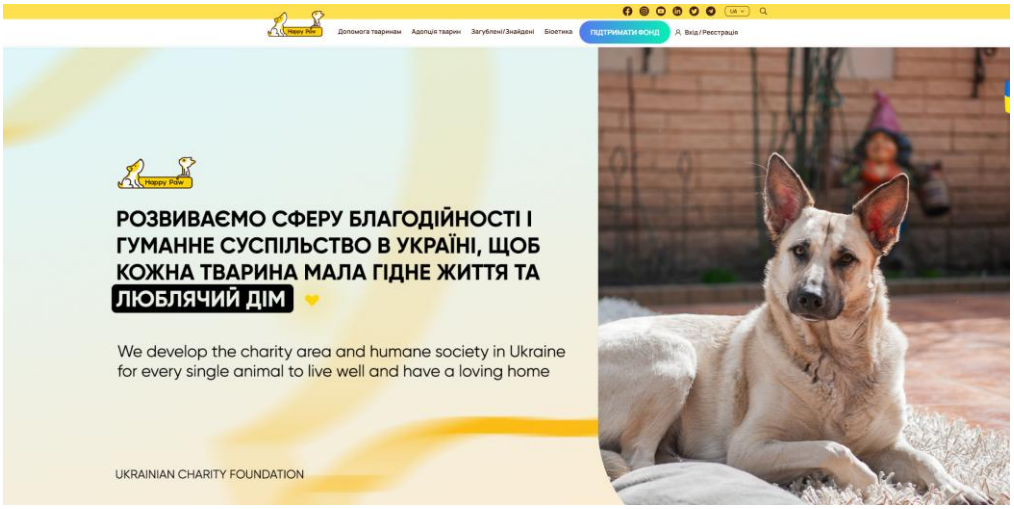

Рисунок 1.3 – Головна сторінка веб-додатку Happypaw [5]

Happypaw користується своєю репутацією, оскільки це старий веб-додаток. Проте, те що буде відразу кидатись в очі – це застарілий дизайн. Єдине де ми можемо побачити його оновленим – це на головній сторінці веб-додатку. Якщо ми говоримо про функціонал – веб-додаток має можливість фільтрації, проте ця фільтрація стандартна, не розширена. Проте веб-додаток має багато зворотнього зв’язку, що дає величезний плюс.

Проаналізувавши найпопулярніші веб-додатку, можемо винести наступне:

– більшість веб-додатків не мають сучасного дизайну, який зацікавив би користувача;

– відсутня або доволі маленька можливість фільтрації, що не дає можливості користувачу знайти потрібну для нього тваринку;

– інформація про зворотній зв’язок прихована в певному місці, яке потрібно найти користувачу, або повністю відсутній, що відлякує його від всиновлення;

– відсутність зручної системи навігації;

– відсутність системи уподобання тварин.

Отже, виходячи з цих даних, ми можемо постановити задачу, яким повинен виглядати наш веб-додатку.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Виходячи з інформації яку ми отримали в розділі 1.1, ми можемо сформувати наступні задачі:

– проаналізувати існуючі веб-додатки, спрямовані на допомогу тваринам, та виявити їхні сильні і слабкі сторони;

– розглянути сучасні технології та інструменти, які підходять для створення веб-додатку;

– реалізувати функціонал перегляду тварин з можливістю фільтрації за категоріями;

– створити сторінку з інформацією про притулок, з яким пов’язаний веб-додаток;

– розробити інтуїтивно зрозумілу та приємну навігацію між сторінками веб-додатку.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАЛЬНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблювального програмного забезпечення та проектування програмного забезпечення

Проаналізувавши веб-додатки в розділі 1, ми можемо винести для себе цілі та основні задачі. Ціллю нашої розробки є створення сучасної веб-платформи для перегляду та усиновлення тварин із притулку. Веб-додаток дозволятиме користувачам зручно переглядати доступних тварин, фільтрувати їх за різними параметрами та додавати їх в обране. Основна мета цього проекту – це підвищити ефективність усиновлення тварин шляхом створення зручного цифрового середовища.

Основними задачами будуть:

– надати користувачам інтуїтивно зрозумілий інтерфейс;

– реалізувати фільтрування за параметрами: розмір, стать, вік, стерилізація та наявність специфічних навичок;

– створення можливості додавання тварин до списку вподобань;

– забезпечення адаптивності веб-додатку для будь-якого екрану – від мобільних пристроїв до великих моніторів;

– забезпечення захисту даних.

Тепер варто сформувати вимоги до розроблювального програмного забезпечення. Поділимо їх на функціональні та нефункціональні. До функціональних віднесемо наступне:

– користувач матиме змогу переглядати список тварин у вигляді карток, на яку можна натиснути;

– під час натискання на карточку буде відкриватись сторінка, де користувач зможе побачити фото, ім’я, вік, стать, розмір та навички тварини;

– користувач матиме змогу вподобати тварину як на її сторінці, так і за допомогою карточки, після чого тварина автоматично буде переміщуватись на початок списку;

– вподобання буде зберігатись в локальному сховищі, аби після оновлення сторінки вони залишались;

– функціонал вподобань буде реалізовано за допомогою глобального стану через Redux;

– користувач матиме змогу скинути фільтри, вподобання, та переглянути список таким, яким він був з самого початку;

– користувач зможе інтуїтивно зрозуміти як отримати зворотній зв’язок та чи активний певний компонент;

– можливість перегляду сторінки із контактною інформацією про притулок;

– можливість подання заявки на усиновлення через відповідність форму на веб-додаток, вказавши необхідну інформацію.

Якщо ми говоримо про нефункціональні вимоги, то до них ми можемо віднести наступне:

– веб-додаток повинен бути повністю адаптивний для будь-яких розмірів екрану, включаючи мобільні пристрої;

– повинна бути анімація переміщення карток, аби користувач розумів, чи зміг він добавити картку до вподобаних;

– повинна бути висока швидкодія, аби фільтрація та відображення списку працювали без затримок;

– потрібна безпечна робота з даними, аби користувач не вводив у певні поля неправильну інформацію;

– система повинна бути масштабованою, аби в майбутньому була можливість додавання нового функціоналу;

– код проекту повинен бути легко підтримуваним і розширюваним, з чітким розділенням відповідальності між компонентами;

– важливі помилки мають бути оброблені та супроводжуватись повідомленнями для користувача.

Тепер розглянемо сценарій використання нашого проекту на рисунку 2.1, та як він вирішить проблему всиновлення.

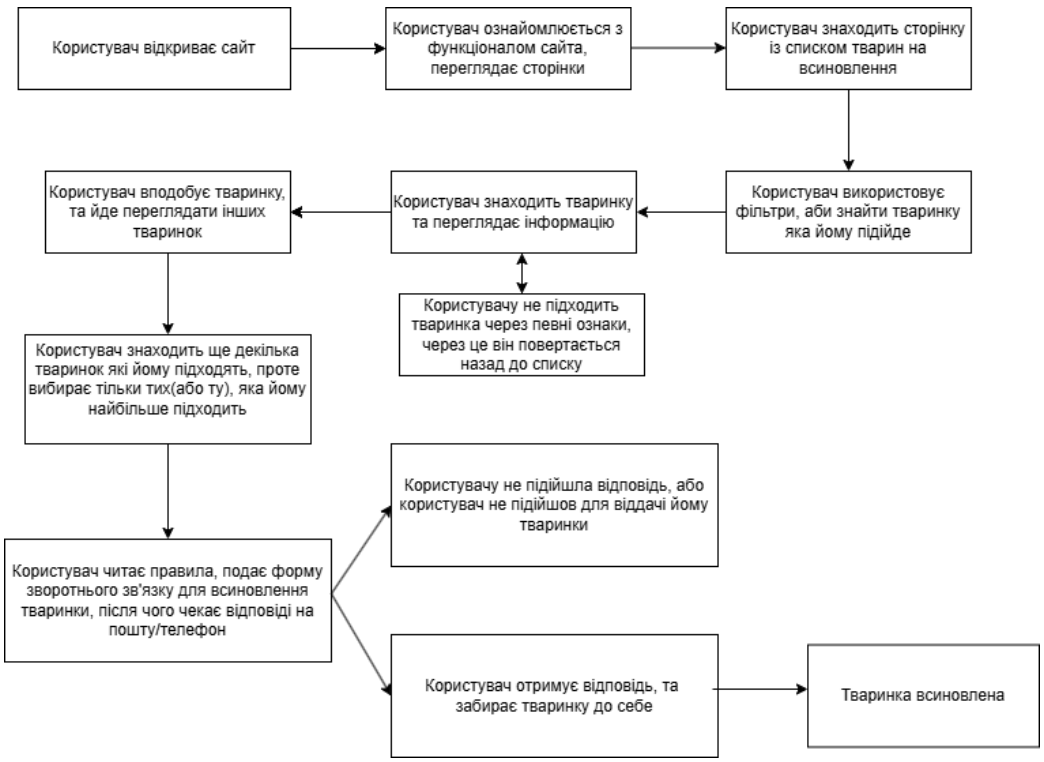

Рисунок 2.1 – Схема відвідування веб-додатку користувачем який хоче всиновити тваринку

Отже, ми можемо побачити що наш веб-додаток дійсно може вирішувати проблему всиновлення тварин, а отже ціль буде досягнута. Тепер розглянемо UX/UI-дизайн, оскільки нам важливо аби дизайн був сучасним та інтерактивним:

– інтерфейс розроблений згідно з стандартами доступності та простоти;

– всі дії користувача будуть відображатись одразу, тобто не прийдеться перезавантажувати сторінку;

– Navbar буде рухатись з користувачем протягом сторінки, що забезпечить доступ до навігації в будь-яку секунду;

– завдяки Framer Motion ми зможемо зробити анімації для зміни списку, вподобань, тощо;

– аби дизайн виглядав цікавим, будуть використовуватись відеоролики замість статичних зображень.

Розглянемо макет нашої головної сторінки на рисунку 2.2.

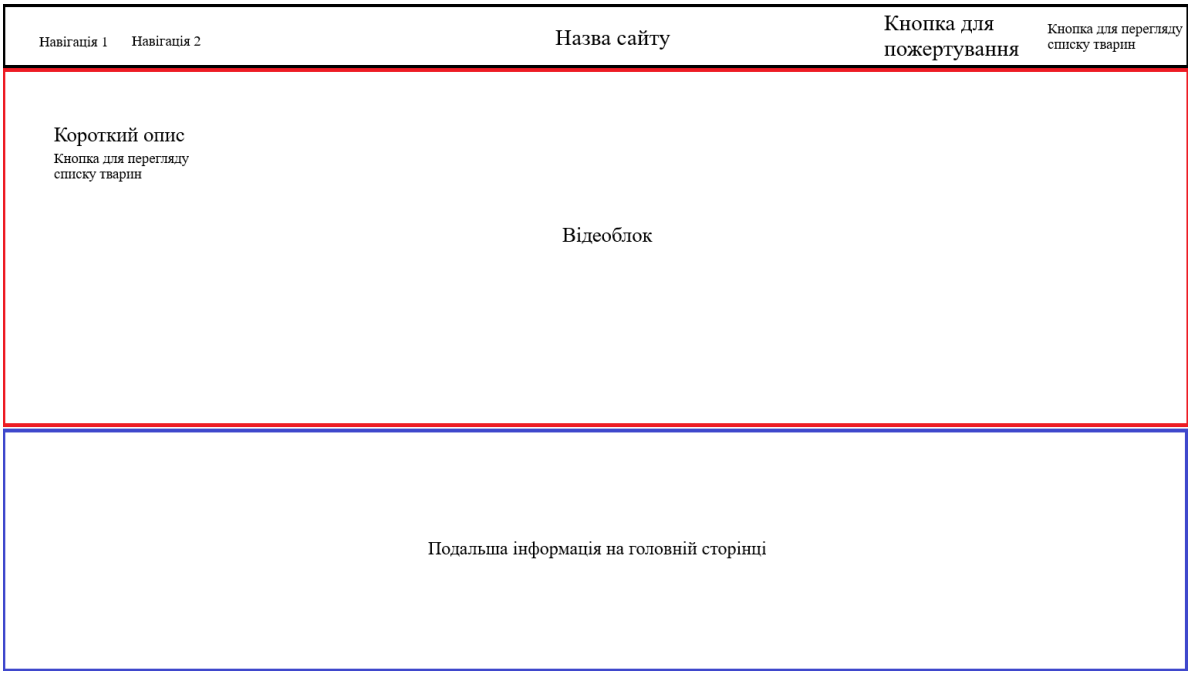

Рисунок 2.2 – Зображення макету веб-додатку

Самий верхній блок – Navbar, який включає в себе всю навігацію, середній блок – відеоролик, на який буде накладена текстова інформація, нижній блок – подальша інформація на головній сторінці, яка включатиме в себе текст та зображення.

Веб-додаток повинен бути виконаний в приємній для користувача кольоровій гаммі, тому варто важливо вибрати кольори. Наприклад, на головній сторінці було б чудово використовувати чорні та білі кольори, оскільки відео на першій сторінці – з чорним котиком. Тобто, ми можемо підбирати кольорову палітру під відео, яке буде на сторінці.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Під час розробки програмного забезпечення було прийнято використовувати React [6] в поєднанні з TypeScript [7], React Redux [8],

TailwindCSS [9] та Framer Motion [10]. Розглянемо, чому саме ми використовуємо ці технології:

– React є одним з найпопулярніших інструментів який використовують у розробці фронтенду. В цьому проекті я хотів використовувати компонентний підхід, що є основною перевагою React, оскільки саме він має компонентну архітектуру, що дозволяє розбивати інтерфейс на невеличкі частині (компоненти), які ми можемо багаторазово використовувати. Це дозволить краще організувати код та дозволить зосередитись на інших елементах, які ми запросто зможемо комбінувати та повторно використовувати. Також React забезпечує реактивну взаємодію, що оптимізує оновлення інтерфейсу. Проте саме важливе – це те, що React має велику кількість готових рішень та бібліотек, а також велику документацію;

– TypeScript є однією з ключових технологій, яку ми будемо використовувати у розробці клієнтської частини застосунку. Наш проект буде містити дуже багато станів, даних, які ми будемо передавати, тому важливо аби ми передача була безпомилкова. В такому випадку нам потрібна чітка типізація, завдяки чому ми зможемо уникнути велику кількість потенційних помилок, які могли б з’явитись внаслідок передачі неправильних параметрів або неправильного використання функції. Також, якщо ми зробимо помилку – ми завжди зможемо побачити де саме, оскільки система типів автоматично підкаже, що і де було порушено. Наприклад, якщо ми забудемо передати певний пропс. І саме головне – завдяки TypeScript ми зможемо краще організувати структуру даних та функцій у самому коді, що полегшить подальшу підтримку системи;

– React Redux буде використовуватись для ефективного керування глобальним станом застосунку. Наприклад, за допомогою React Redux ми можемо запросто створити систему вподобання тварин, яка буде синхронізуватись на всіх сторінках одночасно. Також завдяки React Redux ми зможемо краще контролювати потік даних, уникати неузгодженості та набагато легше відстежувати зміни. React Redux дуже просто інтегрується з React компонентами за допомогою спеціальних хуків, що дозволить більш ефективно

взаємодіяти з даними та зменшить кількість зайвих рендерів. Це дуже спростить нам роботу з фільтрами;

– TailwindCSS давно є улюбленцем для стилізації інтерфейсу багатьох людей, що включає і мене. TailwindCSS дозволяє створювати адаптивний та сучасний дизайн без написання великих обсягів CSS. Ми можемо писати код для CSS прямо в компонентах, що збільшує швидкість розробки та зменшує потребу в написанні додаткових стилів. Також TailwindCSS забезпечує гнучке налаштування тем, кольорів, відступів та інших параметрів. Але саме головне, через що ми будемо його використовувати – ми запросто можемо реалізувати адаптивний дизайн для будь-якого розміру екрану, що є дуже важливим для нас. У комбінації з React ми можемо створювати багаторазові, стилізовані компоненти без складної логіки CSS-файлів. Це значно спростить підтримку інтерфейсу, запросто дозволить масштабувати дизайн у майбутньому та пришвидшить розробку в декілька раз;

– Framer Motion є однією з найпопулярніших та потужних бібліотек для анімацій у React проектах, яка дозволить нам швидко створювати плавні переходи, рухи елементів, анімовану появу та зникнення. Ця бібліотека побудована спеціально для роботи з React-компонентами, тому ми запросто інтегруємо її в проект. Завдяки Framer Motion, ми зможемо зробити більшість візуальної частини проекту естетично привабливою, але й одночасно ми зробимо її більш інтуїтивною для користувача, оскільки анімації допомагають краще орієнтуватись в інтерфейсі та зміні станів. Також для вас важливо, що Framer Motion має внутрішній механізм для зменшення навантаження на DOM, що оптимізує продуктивність в порівнянні з іншими бібліотеками. Нам не прийдеться «винаходити велосипед», який може мати набагато гіршу оптимізацію, нам буде достатньо просто скористатись цією бібліотекою.

У нашій системі будуть використовуватись переважно логічні алгоритми для обробки користувацьких дій і забезпечення взаємодії з інтерфейсом. Проте, варто приділити увагу алгоритму сортування та алгоритму вподобання. Алгоритм фільтрації тварин у нашому проекті буде булевого типу, який буде

приймати в себе набір фільтрів (вік, стать, розмір, навички, стерилізація), та виводити з масив даних, що буде відповідати умовам. Якщо ми говоримо про алгоритм вподобань, тоді тут достатньо використати алгоритм сортування, який буде реалізований через оновлення порядку масиву, де елементи, які вподобали будуть ставати першими в списку.

Коли ми говоримо про реалізацію системи навігації, тоді варто зазначити, що ми будемо використовувати бібліотеку React Router, яка проста та максимально зручна в використанні. Вона дозволить будувати маршрути в односторінковому застосунку (SPA), не перезавантажуючи сторінку повністю, що суттєво покращить швидкодію та користувацький досвід [11]. Варто зазначити, що на деяких наших сторінках будуть застосовуватись динамічні маршрути, проте ми запросто можемо це зробити за допомогою хука, який пропонує нам ця бібліотека. Використовуватись вони будуть для переходу на сторінку з певною тваринкою, на яку натисне користувач. Для зручності навігація завжди буде доступна зверху сторінки, що дозволить користувачу в будь-який період часу перейти на сторінку, яка буде йому необхідна.

Оскільки в нашому додатку використовуються форми зворотнього зв’язку, нам важливо аби ця інформація доходила до власників веб-додатку та притулку. Саме через це в проекті вирішено використати Getform – сервіс, який спрощує обробку даних [12]. Цей сервіс ми зможемо запросто інтегрувати в наш проект, передаючи зручну панель з повною інформацією про відправника на сервер. Також, для нас важливо мати можливість отримувані інформацію зі схованих полів у формі, що нам і надає Getform. Завдяки Getform ми маємо змогу створити стабільну, безпечну та ефективну систему збору заявок, яка не буде вимагати додаткових витрат на серверне обладнання й відповідатиме всім вимогам щодо обробки персональних даних. З допомогою його використання ми будемо отримувати інформацію, на яку саме тваринку була вислана форма, без точного вказання назви тваринки користувачем. Ми можемо побачити принцип, який ми будемо використовувати, на рисунку 2.3.

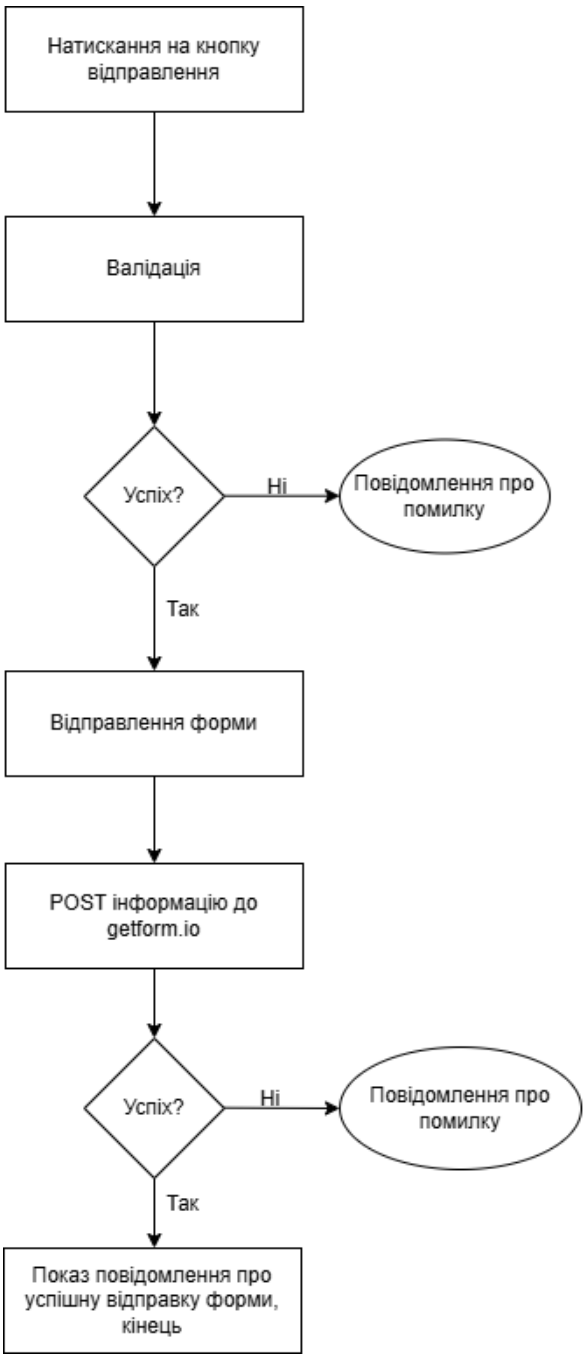

Рисунок 2.3 – Алгоритм відправки форми через сервіс Getform

Також у нас будуть компоненти, які важливі для логіки сторінок. Наприклад, одним з таких буде компонент для анімації переходу між сторінками. Він буде оформлений завдяки Framer Motion, що зробить перехід максимально оптимізованим, та не буде використовувати зайві процеси. Варто ще зазначити про компонент, який буде запам’ятовувати позицію користувача на сторінці, якій він був, оскільки після переходу на іншу сторінку і переміщення назад –

користувач мусить бути на тому самому місці, з якого було здійснене переміщення.

Загалом, нам ще будуть потрібні іконки, які ми будемо використовувати в деяких кнопках або просто як інформацію на сторінках. В цьому нам допоможе React Icons, який надає сучасні та різноманітні іконки, в яких можна запросто міняти стилі на потрібні нам. Також, варто зазначити, що ці іконки дуже легко інтегрувати в наш проект, саме через це наш вибір впав саме на них.

Що стосується сторінок – у нас їх буде 7. Нам важливо аби була сторінка з можливістю фінансової підтримки притулку, сторінка де буде показаний список тваринок, динамічна сторінка з інформацією про певну тваринку, головна сторінка, яка буде показуватись кожному користувачу при заході на веб-додатку, сторінка з правилами, де буде вказана коротка інформація про те, кому і як можна всиновити тваринку, сторінка з розповіддю про притулок, сторінка з контактною інформацією. Важливо, аби окрім Navbar на сторінці був і Footer, який мав важливу інформацію для користувача.

Отже, сформувавши перелік потрібних алгоритмів, інформацію про технології, їх використання, та список необхідних для нас компонентів та сторінок, ми можемо перейти до розділу 3, де ми опишемо створення нашого продукту.

РОЗДІЛ 3
РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ БЛАГОДІЙНОЇ ДОПОМОГИ ТВАРИНАМ

3.1 Практична реалізація об’єкта проектування

Реалізація нашого об’єкта проектування передбачає написання клієнтської частини програмного забезпечення з використанням технологій, які ми розглянули в другому розділі.

Доцільно було б встановити проект відразу з TypeScript, проте, оскільки наш проект з самого початку був на React, замість того аби створювати його з нуля, просто встановимо TypeScript на нього, поміняємо розширення всіх файлів да зробимо типізацію. Проте, спочатку розберемось зі встановленням всього, що нам потрібно у проект. Розглянемо процес встановлення файлів у лістингу 3.1.

Лістинг 3.1 – Демонстрація процесу встановлення файлів

```
npm install --save typescript @types/node @types/react @types/react-dom
@types/jest
npm install react-redux @reduxjs/toolkit tailwindcss framer-motion react-
router-dom
```

кінець лістингу 3.1

Встановивши бібліотеки, які нам потрібні, можемо перейти до створення нашого продукту. Далі, давайте створимо нашу загальну систему навігації та отримування інформації через Provider, який надає нам React Redux. Ми можемо побачити це в лістингу 3.2.

Лістинг 3.2 – Демонстрація структури навігації в App.tsx, передача інформації між сторінками завдяки Provider

```
function App() {
  return (
    <div className="bg-black">
      <Provider store={store}>
      <AnimatePresence mode="wait">
      <Router>
```

```
        <Navbar/>
        <Routes>
          <Route path="/" element={<MainPage/>} />
          <Route path="/about" element={<AboutPage/>} />
          <Route path="/donation" element={<DonatePage/>}/>
          <Route path="/pet/:id" element={<PetPage />} />
          <Route path="/adoptation" element={<AdoptationPage/>}/>
          <Route path="/rules" element={<RulesPage/>}/>
          <Route path="/contacts" element={<ContactPage/>}/>
        </Routes>
        <Bottom/>
      </Router>
      </AnimatePresence>
      </Provider>
    </div> );}
export default App;
```

кінець лістингу 3.2

Як ми можемо побачити, у нас встановлені шляхи до сторінок, а якщо ми беремо PetPage – ми бачимо динамічну систему шляху. Також, варто підмітити, що наш продукт огорнутий в AnimatePresence, що необхідний йому для створення анімацій між сторінками.

Тепер, давайте будемо йти зверху до низу. Store.tsx ми обговоримо пізніше, в іншому розділі, а зараз розглянемо звичайні компоненти. Давайте розпочнемо з Navbar.

Пам’ятаємо, що в розділі 2 ми зазначили, що Navbar повинен завжди бути з користувачем, незалежно від стану та локації користувача на сторінці. Ми можемо побачити, що в нашій структурі Navbar рендериться незалежно від того, на якій він сторінці. Проте, цього недостатньо. Завдяки TailwindCSS ми можемо зробити просту логіку, яка змусить Navbar рухатись за користувачем. Принцип дії ми можемо побачити в лістингу 3.3.

Лістинг 3.3 – Принцип фіксації позиції Navbar, зміна стилю залежно від цього

```
const [color, setColor] = useState(false);

const changeColor = () => {
    if (window.scrollY >= 20) {
      setColor(true);
    } else {
```

```
      setColor(false);
    }
  };

 useEffect(() => {
    window.addEventListener("scroll", changeColor);
    return () => {
      window.removeEventListener("scroll", changeColor);
    };
}, []);


className={
color      ? "w-full flex items-center h-[75px] bg-[#181414] fixed z-20
top-0 duration-200 border-b border-b-transparent"
          : "w-full fixed flex items-center h-[75px] z-20 border-b-white
border-b top-0 duration-200"
      }
```

кінець лістингу 3.3

Як ми бачимо, окрім того що Navbar рухається до користувача коли наше прокручування більше ніж 20 пікселів, ми також динамічно міняємо колір. Також ми маємо useEffect, яким контролюємо стан прокручування, та постійно обновляємо його залежно від позиції прокручування на сторінці. Тепер давайте розглянемо можливість навігації на інші сторінки в лістингу 3.4.

Лістинг 3.4 – Можливість навігації на іншу сторінку при натисканні на кнопку

```
     <Link to="/adoptation">
           <button className={
                color ? "uppercase text-center items-center text-[13px]
bg-transparent border hover:bg-white hover:text-black text-white rounded-
3xl px-[30px] py-[6px] font-semibold duration-300"
                     : "uppercase text-center items-center text-[13px]
hover:bg-[#123B83] hover:text-white hover:border-[#123B83] border bg-
transparent text-white rounded-3xl px-[30px] py-[6px] font-semibold
duration-300"}> Adopt
           </button></Link>
```

кінець лістингу 3.4

Варто зазначити, що компонент Link ми отримуємо від бібліотеки React Router, використання якої ми обговорювали в розділі 2. Як ми бачимо, при натисканні на кнопку ми маємо змогу перемістити на сторінку з всиновленням

тварин. Також, бачимо, наскільки TailwindCSS спрощує роботу зі стилями, дозволяючи задавати їх прямо в компонентах. Наприклад, наша кнопка змінює колір при наведенні на неї, що значно підвищує інтуїтивність інтерфейсу. Разом зі зміною кольору відбувається і зміна кольору тексту, обведення і так далі. Подібний принцип навігації ми маємо на інших кнопках, тому, вважаю, що нам не потрібно описувати кожен з них, оскільки логіка залишається такою самою. Варто зазначити, що в Navbar ще є мобільна версія, проте її винесемо в окремий розділ.

Після того як ми розглянули Navbar, вважаю за необхідне розглянути Footer. На ньому також є кнопки, за допомогою яких ми можемо перейти на іншу сторінку. Із важливого – ми можемо переміститись на самий верх сторінки завдяки зручній кнопці, яка плавно прокрутить сторінку. Ми можемо побачити принцип роботи в лістингу 3.5.

Лістинг 3.5 – Можливість плавного переміщення на самий верх сторінки

```
const scrollToTop = () => {
    window.scrollTo({ top: 0, behavior: 'smooth' });
  };
<FaLongArrowAltUp onClick={scrollToTop} className="bg-white lg:text-[42px] text-[36px] md:text-[44px] text-black p-2 rounded-full cursor-pointer hover:bg-[black] hover:text-white duration-300" />
```

кінець лістингу 3.5

Разом з цим, ми бачимо використання іконки, яка описана як FaLongArrowAltUp. По натисканню на кнопку відбувається виконання функції, після чого ми переміщуємся на самий верх, незалежно від розміру екрану на якому ми переглядаємо веб-додаток. Footer зроблений так само як і Navbar, тому тут немає необхідності детального розгляду. Єдине, що є ще важливим – внизу написана контактна інформація, оскільки вона має бути розміщена в інтуїтивно зрозумілому місці для користувача.

Наступним, що є важливим, потрібно розглянути як будуть виглядати наші сторінки. Загалом, структура кругом є однаковою – відеоролик який займає

певну висоту сторінки, і текст на ньому, а вже після цього йде подальша інформація. Побачити це ми можемо в лістингу 3.6.

Лістинг 3.6 – Структура коду головної сторінки на більшості сторінок

```
<div className="h-full w-full relative bg-black">
      <video
         className="w-full md:h-[800px] h-[800px] object-cover justify-
center flex"
        src={bg}
        autoPlay
        loop
        muted
      />
      <div className="absolute top-[140px] md:left-16 left-8 text-white">
          <p  className="md:text-[30px] text-[26px] font-medium">He also
wants</p>
        <p className="md:text-[30px] text-[26px] font-medium mb-2">
          to feel love and have family!
        </p>
        <p className="text-[14px] max-w-[370px]">
           Adopting a pet gives him the opportunity to find a home with
loving
           owners, where he will receive the necessary care, comfort and
warmth.
        </p>
        <Link to="/adoptation">
          <button className="mt-4 link link-underline underline-offset-8
uppercase text-[13px] tracking-wide font-semibold">
            adopt a pet
          </button>
        </Link>
      </div>
</div>
```

кінець лістингу 3.6

Як ми можемо побачити, код вже містить адаптивність під різні екрани, проте, більш детально ми це розглянемо в розділі 3.4. Ми можемо змогу побачити розміщення елементів на сторінці, що текст має позицію absolute, що дає йому можливість розміститись на відеоролику. Сам відеоролик зациклений, автоматично запускається та не має звуку. Окрім цього, поверху основного блоку інформації у нас завжди лежить Navbar, неважливо на якій сторінці ми

перебуваємо. Давайте побачимо, як буде виглядати фрагмент нашої головної сторінка з таким кодом на рисунку 3.1.

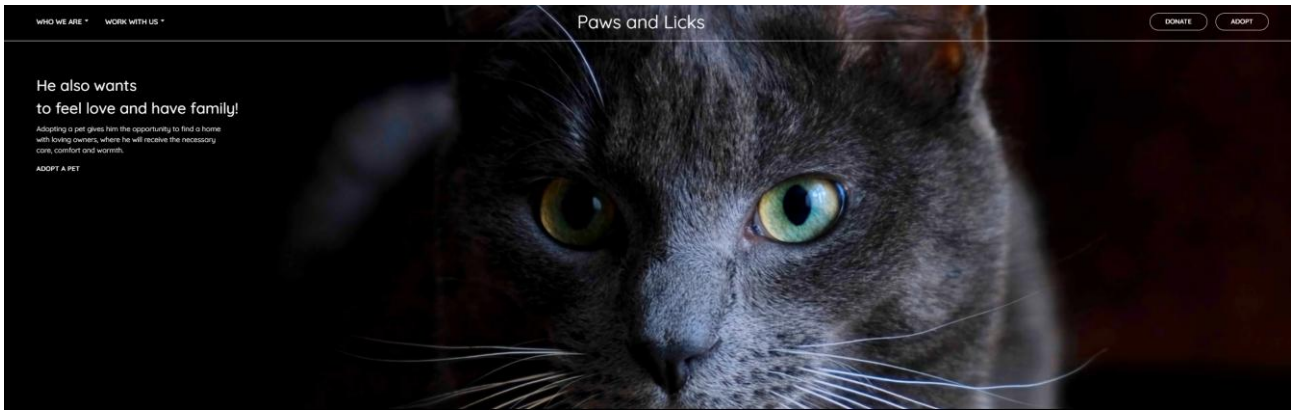

Рисунок 3.1 – Зображення фрагменту головної сторінки

Наступним важливим кроком є створення коду на сторінці з всиновленням тварин. Створення системи фільтрів ми розглянемо в розділі 3.5, проте саму структуру, оскільки вона є найважливішою на нашому веб-додатку, ми маємо розглянути.

Як працює наша сторінка? Ми будемо рендерити картки з тваринками, які перебувають в притулку. Після цього у користувача буде змогу відфільтрувати список за власним вподобанням, а після цього – натиснути на картку з тваринкою, яка йому сподобалась. Отже, з самого початку розглянемо код до карток.

Наша картка містить фотографію, базову інформацію, а також інтерактивні кнопки: перегляд подальшої інформації та можливість вподобання. Разом з цим, користувач може просто натиснути на картку, аби перейти на сторінку з тваринкою. Із основного тут варто розглянути, які пропси приймає компонент з карточкою, що ми можемо побачити в лістингу 3.7.

Лістинг 3.7 – Пропси, які приймає компонент AdoptionCard

```
interface AdoptionCardProps {
  animal: Animal;
  onClick: () => void;
  onLike: () => void;
```

```
  index: number;
  isLiked: boolean;
}
```

кінець лістингу 3.7

Як ми бачимо, ми отримуємо інтерфейс тварини (Animal), певні функції, індекс, та перевірку, чи вподобана карточка. Разом з цим, ми використовуємо масиви різноманітних кольорів та фігур для динамічного оформлення карток, аби зробити відображення карток на сторінці набагато цікавішим, що ми можемо побачити в лістингу 3.8.

Лістинг 3.8 – Масив кольорів на фігур, відображення кольору і форми на основі індексу картки

```
const colors = ["#A2F5BF", "#F5E1A2", "#F5A2C2", "#B2A2F5", "#F5A2A2"];

const shapes = [
  "polygon(0% 30%, 100% 10%, 100% 100%, 0% 70%)", // Хвилястий верх
  "polygon(0% 40%, 100% 0%, 100% 100%, 0% 60%)", // Діагональ
   "polygon(10% 0%, 90% 0%, 100% 50%, 90% 100%, 10% 100%, 0% 50%)", // 
Шестикутник
  "ellipse(40% 50% at 30% 50%)", // Овал збоку
  "polygon(0% 10%, 100% 0%, 100% 90%, 0% 100%)", // Нахилений прямокутник
  "polygon(0% 50%, 100% 0%, 100% 100%, 20% 80%)", // Кутовий зріз
  "polygon(0% 100%, 100% 60%, 100% 100%)", // Трикутник внизу
  "polygon(10% 20%, 100% 0%, 100% 80%, 0% 100%)", // Незвична фігура
];
  const bgColor = colors[index % colors.length];
  const shape = shapes[index % shapes.length];
```

кінець лістингу 3.8

Далі йде звичайна верстка, відображення кнопки вподобання різними кольорами (в залежності від стану кнопки), та анімація на вподобання. Перейдемо відразу до розгляду рендеринга карток.

Наш список рендериться на сторінці AdoptationPage, де присутня система фільтру цих карток, проте про це розглянемо в розділі 3.5. Сам список відображає лише 12 карток на сторінці, проте ми маємо можливість перейти на відображення наступних 12 карток завдяки пагінації, яка винесена у нас в

окремий компонент, для комфортної роботи з нею. Загалом, це дозволяє не створювати на сторінці «нескінченний список». Продемонстрований код ми можемо побачити в лістингу 3.9.

Лістинг 3.9 – Пагінація при відображенні списку із карток

```
const [currentPage, setCurrentPage] = useState<number>(1);
const animalsPerPage = 12;
const indexOfLastAnimal = currentPage * animalsPerPage;
const indexOfFirstAnimal = indexOfLastAnimal - animalsPerPage;
const    currentAnimals    =    orderedAnimals.slice(indexOfFirstAnimal,
indexOfLastAnimal);
const totalPages = Math.ceil(filteredAnimals.length / animalsPerPage);

const handlePageChange = (pageNumber: number) => {
  setCurrentPage(pageNumber);
  localStorage.setItem("currentPage", pageNumber.toString());

  if (filterButtonRef.current) {
    const                          buttonPosition                          =
filterButtonRef.current.getBoundingClientRect().top  +  window.scrollY  -
100;
    window.scrollTo({ top: buttonPosition, behavior: "smooth" });
  }
};
```

кінець лістингу 3.9

На кожну з карток у нас є можливість натиснути, після чого ми переходимо на сторінку з натиснутою тваринкою. Відбувається це завдяки передачі animalId, яке ми також використовуємо при вподобанні. Розглянемо рендеринг карток в лістингу 3.10.

Лістинг 3.10 – Рендеринг карток на сторінці AdoptationPage

```
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-
cols-4 gap-8 mt-6 mx-8">
  <AnimatePresence initial={false}>
  {currentAnimals.map((animal: Animal) => (
    <motion.div
      layout
      key={animal.id}
      initial={{ opacity: 0, scale: 0.9 }}
      animate={{ opacity: 1, scale: 1 }}
      exit={{ opacity: 0, scale: 0.9 }}
```

```
      transition={{ duration: 0.3 }}
      onClick={() => handleCardClick(animal.id)}
    >
      <AdoptationCard
        index={animal.id}
        animal={animal}
        onClick={() => handleCardClick(animal.id)}
        onLike={() => handleLike(animal.id)}
        isLiked={likedAnimalIds.includes(animal.id)}
      />
    </motion.div>
  ))}
</AnimatePresence>
  </div>
```

кінець лістингу 3.10

Після натискання на картку, ми переносимось на сторінку з тваринкою, де ми отримуємо повну інформацію про неї, оскільки інформація про неї передається між компонентами. Ми можемо побачити детальну інформацію про тваринку, кнопку для вподобання та кнопку для подачі заявки на всиновлення, що являє собою модальну форму, яку ми розглянемо в розділі 3.3. Загалом, сторінка представляє собою звичайну верстку, проте варто розглянути як саме ми передаємо інформацію з сторінки AdoptationPage на сторінку PetPage, оскільки ми використовуємо не лише динамічний роутинг, а й глобальний стан через Redux, що продемонстровано в лістингу 3.11.

Лістинг 3.11 – Передача інформації з AdoptationPage на Petpage

```
  const { id } = useParams<{ id: string }>();
  const animals = useSelector((state: { animals: { animals: PetDetails[]
} }) => state.animals.animals);
  const [petDetails, setPetDetails] = useState<PetDetails | null>(null);
useEffect(() => {
    window.scrollTo(0, 0);
    const pet = animals.find((animal) => animal.id === parseInt(id ?? ""));
    if (pet) {
      setPetDetails(pet);
    }
  }, [id, animals]);

  if (!petDetails) {
    return <div>Loading...</div>; }
```

кінець лістингу 3.11

Завдяки цьому ми зрозуміли, як здійснюється процес відображення та передачі інформації між компонентами та сторінками, від карток до сторінки з тваринками.

Наш веб-додаток також має безліч інших сторінок, наприклад, контактна інформація, інформація про притулок та правила всиновлення, проте їх структура схожа на те, що ми вже розглядали, тому нічого нового ми не побачимо.

Написавши код до нашого проекту, ми можемо перейти до тестування та розгляду, чи дійсно все працює так як задумано.

3.2 Тестування налагодження інформаційно-комп'ютерної системи

Тестування є невід'ємною частиною розробки нашого проекту, оскільки завдяки ньому ми можемо виявити помилки та усунути їх до впровадження у реальне середовище. Для нас буде важливим перевірити відправку форми (чи передається інформація на сервер), відображення карток, відображення правильної інформації на сторінці з тваринкою та функціонал фільтрів.

Розпочати розгляд варто з відправки форми, оскільки вона є основою для подання заявки на всиновлення. Саме через цей процес ми будемо отримувати контактну інформацію та найважливіше – інформацію про тваринку, що буде отримувати заявку на всиновлення.

Після того як користувач натискає на кнопку «Take the tail home» всередині модального вікна – він отримує повідомлення про відправку форми (в разі успіху), після чого він може закрити це модальне вікно. Це створює ефект завершеності дії й додає користувачеві впевненості, що система працює коректну та його інформація приймається в обробку. Відправка даних відправляється на спеціальну пошту, яка буде передана власникам веб-додатку. Давайте розглянемо, чи правильно передались дані на сервер, та чи є номер тваринки, що ми можемо побачити на рисунку 3.2.

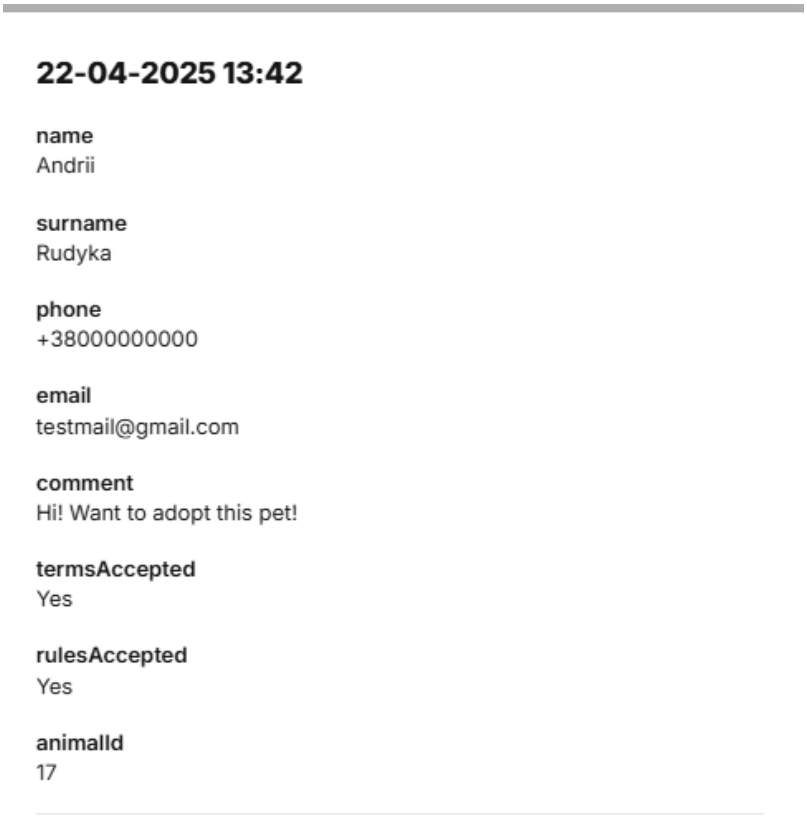

Рисунок 3.2 – Дані, які передались з заповненої форми на сервер

Як ми бачимо, дані на сервер передаються успішно, а саме головне – що ми бачимо номер тваринки, який передається автоматично. Це дозволяє чітко ідентифікувати, про яку саме тваринку ідеться в заявці. В коментарях спеціально не було написано яка саме тваринка, адже 95 відсотків користувачів не будуть цього писати. Отже, перший тест успішно пройдено.

Далі розглянемо відображення карток на сторінці. Це є критично важливий аспект, який визначає зручність взаємодії користувача з веб-додатком. Перш за все, ми повинні переконатись, що картки з тваринками відображаються корректно. Особливу увагу приділяємо механізму вподобання тварини, оскільки нам важливо, щоб тваринка, яку вподобав користувач, автоматично переміщувалась на початок списку. Це дозволяє підкреслити вибір користувача, та дати йому змогу швидше повернутись до тваринок, які йому дійсно сподобались. Така система підвищує шанси всиновлення тваринки, оскільки користувач користується зручним функціоналом, а не гортає список з самого початку. Розглянемо це на рисунку 3.3.

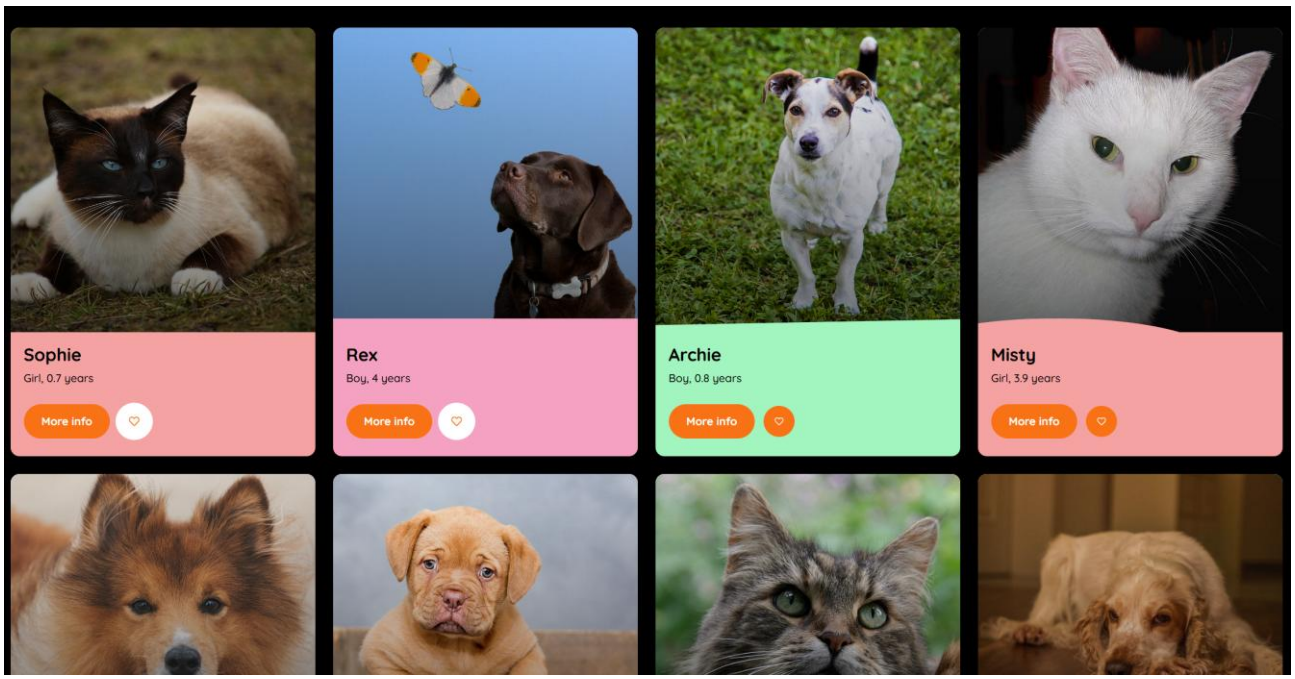

Рисунок 3.3 – Відображення карток з тваринами на сторінці

Як ми бачимо, список успішно рендериться та відображається, ми успішно бачимо зображення тваринок, коротку інформацію про них, та маємо можливість натиснути на карточку. Також, як бачимо, вподобання тваринок успішно працює, та остання вподобана тваринка переноситься на самий початок списку, неважливо на якій сторінці до цього вона перебувала. Тепер давайте розглянемо сторінку з інформацією про тваринок, яка продемонстрована на рисунку 3.4.

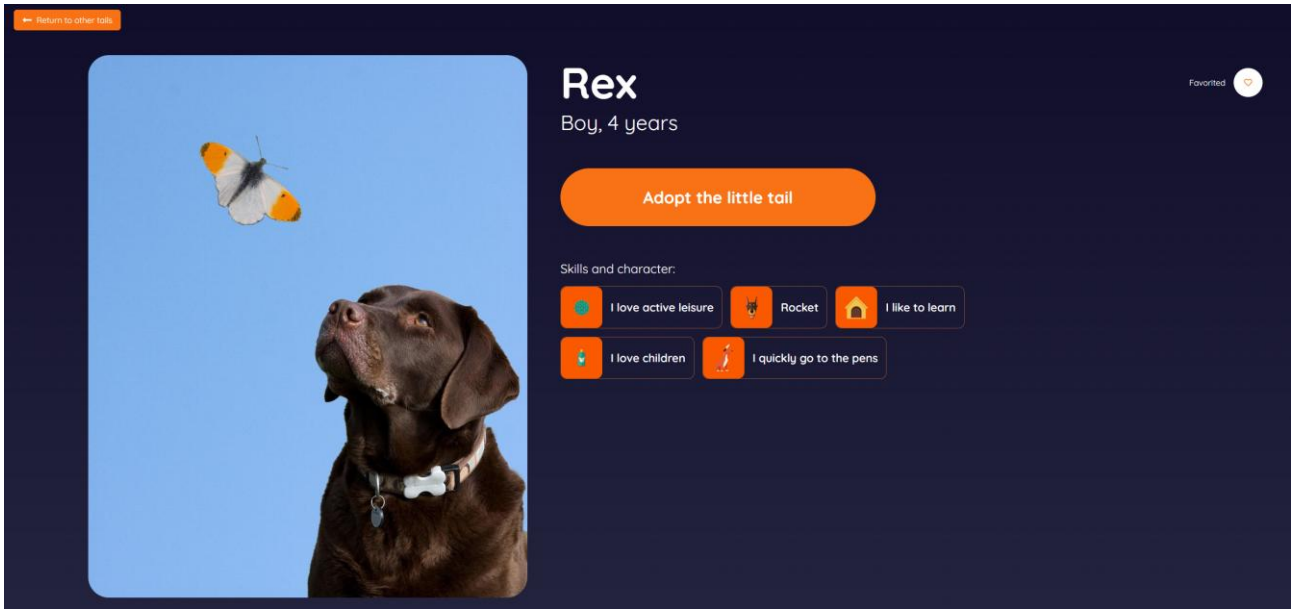

Рисунок 3.4 – Відображення інформації про тваринку на її сторінці

Ми можемо побачити збільшену картинку, на яку користувач має змогу натиснути та побачити повну картинку, кнопку для всиновлення, навички, які притаманні тварині та саме головне – що тваринка вподобана. Тобто, стан вподобання успішно передається між компонентами. Коли ми натискаємо на кнопку – відкривається модальна форма, тобто все працює успішно.

Наступним для розглядання є фільтрація. Важливо сказати для потрапляння у відфільтрований список, вона повинна відповідати усім критеріям, а не лише одному із них. Такий підхід забезпечує більш точний і релевантний результат для користувача. Давайте спочатку розглянемо меню з фільтрацією. Воно відкривається під час натискання на кнопку, після чого перед нами з'являється модальне вікно, яке продемонстроване на рисунку 3.5.

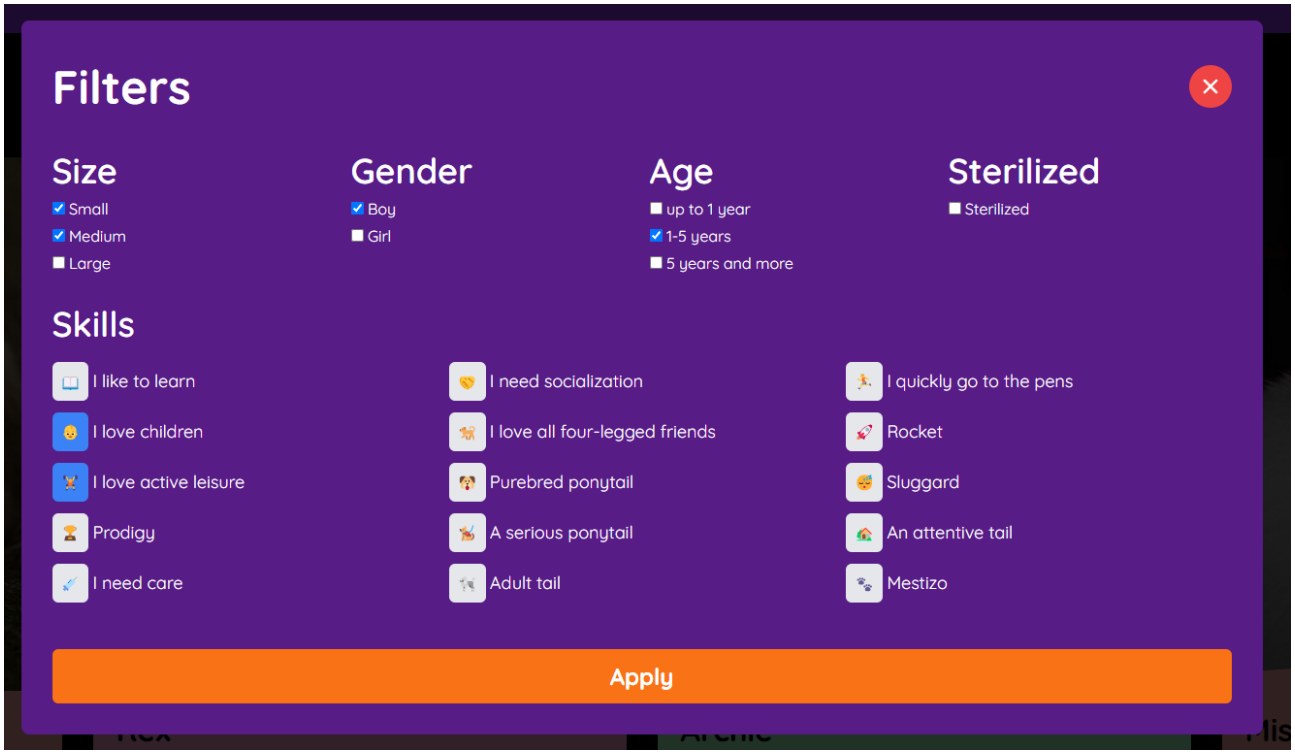

Рисунок 3.5 – Відображення модального вікна з фільтрами

Після того як ми виберемо фільтри, які потрібні для нас – натиснемо Apply, та дивимось, чи перерендериться наш список з тваринками. Важливо, аби тваринка чітко підходила під всі фільтри. Результат нажаття ми можемо побачити на рисунку 3.6.

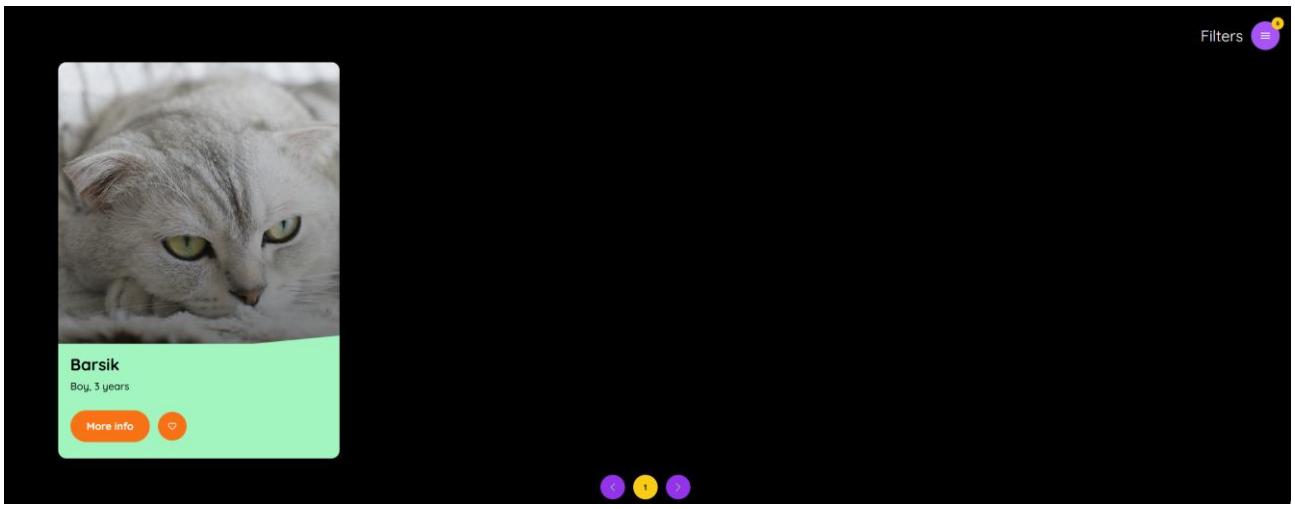

Рисунок 3.6 – Відображення списку після застосування фільтрів

Як ми бачимо – нам вибило конкретну тваринку, яка підходить по всім фільтрам. Разом з цим, ми можемо побачити скільки саме фільтрів було застосовано, оскільки у нас відображається цифра активних фільтрів на кнопці. Після перезавантаження сторінки фільтри не скидаються, тобто все працює як було задумано. Отже, тут тест теж було пройдено успішно.

Проте, тестування допомогло знайти нам і баги. Наприклад, були проблеми при вподобанні тваринки, оскільки прокручування переносився на самий верх. Завдяки цьому ми зрозуміли, що потрібно зберігати позицію прокручування. Або, коли дані просто не завантажувались в PetPage, через що ми виправляли логіку UseEffect.

Загалом, тестування допомогло нам виправити багато помилок, проінспектувати функціонал нашого продукту, та зрозуміти, що потрібно ще добавити/виправити, а що все знаходиться в готовому стані.

3.3 Створення та реалізація модальної форми із валідацією та надсиланням даних на сервер

Реалізація модальної форми є однією з важливих деталей у нашому проекті, оскільки користувач мусить мати змогу відправити форму для

зворотнього зв’язку. Без цього ми не зможемо взнати, яку саме тваринку для адаптації було вибрано та дізнатись контактну інформацію користувача.

Компонент ми створюємо за допомогою хуків стану (useState, useEffect). Також, створили клієнтську валідацію введених даних, а також надсилання форми на сервер через API сервісу Getform [13]. Для початку, давайте розглянемо пропси, які приймає компонент, що ми можемо побачити в лістингу 3.12.

Лістинг 3.12 – Пропси, які приймає компонент AdoptionModal

```
interface AdoptionModalProps {
  isOpen: boolean;
  onClose: () => void;
  animalId: number;
}
```

кінець лістингу 3.12

Наступним, що потрібно розглянути, це ініціалізація стану, та типи для даних форми. Розглянемо їх в лістингу 3.13.

Лістинг 3.13 – Ініціалізація стану форми

```
const [formData, setFormData] = useState<FormData>({
  name: "",
  surname: "",
  phone: "",
  email: "",
  comment: "",
  termsAccepted: false,
  rulesAccepted: false,
  animalId: animalId,
});
```

кінець лістингу 3.13

Важливим кроком буде реалізація синхронізації animalId при оновленні пропсів, оскільки ми працюємо відразу з декількома тваринками, що створює потенційні ризики некоректного відображення або передачі інформації. Код для цього ми можемо побачити в лістингу 3.14.

Лістинг 3.14 – Синхронізація animalId при оновленні пропсів

```
useEffect(() => {
  setFormData((prevData) => ({
    ...prevData,
    animalId: animalId,
  }));
}, [animalId]);
```

кінець лістингу 3.14

Далі, давайте розглянемо валідацію [14]. Чому вона важлива? Оскільки користувач може допустити помилки в деяких полях, і нам важливо аби він зрозумів, в чому саме помилка, та як її виправити. Також, ми блокуємо деякі можливості, наприклад аби у поле з номером телефона користувач не міг ввести текст. Розглянемо валідацію у лістингу 3.15.

Лістинг 3.15 – Валідація у формі зворотнього зв’язку

```
const validateForm = () => {
  const newErrors: Errors = {};
  if (!formData.name.trim()) newErrors.name = "Name is required";
  if (!formData.surname.trim()) newErrors.surname = "Surname is required";
  if (!formData.email.trim()) {
    newErrors.email = "Email is required";
  } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
    newErrors.email = "Invalid email format";
  }
  if (!formData.termsAccepted)
    newErrors.termsAccepted = "You must accept the Terms of Use";
  if (!formData.rulesAccepted)
    newErrors.rulesAccepted = "You must accept the Rules for tail care";

  setErrors(newErrors);
  return Object.keys(newErrors).length === 0;
};
```

кінець лістингу 3.15

Після цього перейдемо до найважливішого пункту – відправки форми. Ми будемо відслідковувати відправку форми, і в випадку чого виводити повідомлення про успіх або помилку користувачу, аби він розумів, чи все успішно, чи ні. Також у відправці форми присутній номер тваринки, який

відправляється автоматично, тобто користувач не вводить ніякої інформації про цю тваринку. Це є необхідним для реалізації системи всиновлення, оскільки користувачі інтуїтивно завжди враховують, що система сама відправить інформацію про тваринку, яка їм потрібна. Розглянемо це в лістингу 3.16.

Лістинг 3.16 – Реалізація функціоналу відправки форми

```
const handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    setSubmissionMessage("");
    setIsSubmitting(true);
    if (validateForm()) {
      const formDataToSend = new FormData();
      formDataToSend.append("name", formData.name);
      formDataToSend.append("surname", formData.surname);
      formDataToSend.append("phone", formData.phone);
      formDataToSend.append("email", formData.email);
      formDataToSend.append("comment", formData.comment);
        formDataToSend.append( "termsAccepted", formData.termsAccepted ?
"Yes" : "No );
        formDataToSend.append( "rulesAccepted", ormData.rulesAccepted ?
"Yes" : "No" );
      formDataToSend.append("animalId", String(formData.animalId ?? ""));
        fetch("https://getform.io/f/aqoklyga", { method: "POST", body:
formDataToSend, })
        .then((response) => {
          if (response.ok) { setSubmissionMessage( "Thank you! Your form
has been successfully submitted." );
           setIsSubmitting(false); setIsFormSubmitted(true);
          } else { setSubmissionMessage ( "There was an issue submitting
your form. Please try again." );
           setIsSubmitting(false); setIsFormSubmitted(true); } })
          .catch((error) => { console.error("An error occurred while
submitting the form:", error);
         setSubmissionMessage("There was an error. Please try again.");
          setIsSubmitting(false); setIsFormSubmitted(true); });
    } else {
      setIsSubmitting(false); }};
```

кінець лістингу 3.16

Модальне вікно відкривається лише коли користувач жме на кнопку «Adopt the little tail», і має можливість закриватись. Має воно розмітку з полями вводу, текстовим контейнером для введення користувачем певних коментарів, чекбоксами та кнопкою підтвердження. Разом з цим, при відправці користувачу

вказується, що повідомлення відправляється, а при успішній або неуспішній відправці – відповідне повідомлення. Розглядати це окремо не є важливим, оскільки похожі принципи ми бачили у розділі 3.1.

3.4 Розробка адаптивності під будь-який екран

Адаптивність в цьому проекті присутня в будь-якому місці веб-додатку. Візьмемо за приклад відображення елементів на сторінці AdoptationPage, код якої ми можемо побачити в лістингу 3.17.

Лістинг 3.17 – Демонстрація можливостей адаптивності деяких елементів на сторінці AdoptatonPage

```
    <div className="px-6 md:px-12 2xl:px-16 3xl:px-48 py-12 w-full flex
flex-col  md:flex-row  justify-between  bg-purple-600  rounded-b-[60px]
lg:rounded-b-[120px] lg:min-h-screen">
           <div className="mt-16 md:mt-20 lg:mt-56 text-center lg:text-
left">
              <h1 className="text-[48px] sm:text-[64px] md:text-[50px]
xl:text-[84px] lg:text-[64px] 2xl:text-[114px] font-bold xl:max-w-[900px]
md:max-w-[490px] leading-[1.2]">
             Here you can find your four-legged friend </h1>
               <p  className="text-[20px]  sm:text-[24px]  md:text-[24px]
lg:text-[36px]  xl:text-[40px]  xl:max-w-[750px]  md:max-w-[450px]  mt-6
md:mt-8 lg:mt-12">
              In the shelter, there will be friends for everyone - big,
small,
             guards, companions, lazy and restless.
           </p>
         </div>
           <div className="zoom-container rotate-[0.08rad] w-full sm:w-
[500px]  md:w-[500px]  lg:w-[400px]  xl:w-[700px]  h-[400px]  sm:h-[500px]
md:h-[500px] 2xl:h-[900px] lg:h-[700px] xl:h-[800px] 3xl:w-[800px] 3xl:h-
[1000px] overflow-hidden mt-12 md:mt-16 lg:mt-32 mb-8 md:mb-12 lg:mb-16
lg:mx-0 md:mx-4 mx-auto">
           <img
              className="zoom-image w-full h-full object-cover rounded-
[60px] sm:rounded-[80px] md:rounded-[100px] lg:rounded-[120px] transition-
transform"
             src={catSam}
             alt="Pet"
           />
         </div>
       </div>
```

кінець лістингу 3.17

Як ми можемо побачити – зображення та текст завжди змінюють свій розмір, а на маленьких екранах – повністю міняють структуру відображення. Так відбувається абсолютно на будь-якому елементі веб-додатку, оскільки нам важливо, щоб користувачі почували себе зручно на будь-якому пристрої. Проте, давайте розглянемо можливість адаптивності більш широко, на прикладі мобільних девайсів.

Для розгляду такої можливості нам допоможе Navbar, а саме – його мобільна версія. Ми містимо його в окремому компоненті, який називається MobileNav, та рендеримо його в самому Navbar. Проте, він відображається лише коли екран менше певного розміру, і разом з тим пропадають інші елементи Navbar, які переносяться всередину мобільної версії.

Коли наш екран стає менший певного розміру, наш Navbar має лише одну кнопку та назву веб-додатку. Під час натискання на кнопку, ми відкриваємо вікно, де зберігається вся інформація з Navbar для маленьких екранів, зображення якого ми можемо побачити на рисунку 3.7.

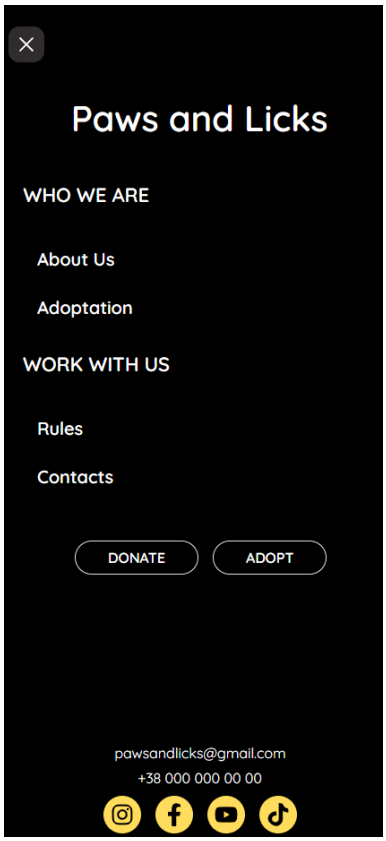

Рисунок 3.7 – Відображення меню мобільного Navbar

Ми маємо кнопки «Who we are» та «Work with us» які з самого початку закриті. Під час натискання у нас ці меню відкриваються, та ми можемо побачити там кнопки для навігації між сторінками. При навігації – ми переміщуємось на потрібну нам сторінку, а після цього цей Navbar закривається. Ми завжди маємо змогу відкрити його.

Тобто, адаптивність присутня абсолютно кругом, що покращує юзабіліті серед користувачів.

3.5 Створення фільтраційної системи з автооновленням результатів

Система фільтрації особливо важлива в нашому проекті, оскільки вона допоможе підвищити можливість всиновлення тваринки. Вважаю, що ніякому користувачу не було б зручно натискати на кожну тваринку, і шукати ту, що йому найбільше сподобається. Цю проблему і допомагає вирішити система фільтрації. Загальна структура реалізована через глобальний стан Redux, що дозволяє зберігати фільтри по всьому веб-додатку, навіть після оновлення або переходом між сторінками.

Давайте розглянемо, які змінні використовуються у фільтрах, код яких ми можемо побачити в лістингу 3.18.

Лістинг 3.18 – Змінні, які використовуються у фільтрах та знаходяться у store.ts

```
filters: {
  size: string[];
  gender: string[];
  sterilized: boolean;
  skills: string[];
  ageCategory: string[];
}
```

кінець лістингу 3.18

Ці змінні зберігаються у Redux-слайсі animals, що керує станом застосунку. Далі, перейдемо до сторінки AdoptationPage. На цій сторінці у нас є локальні стани, в які зчитується значення з Redux, і стани, які зберігають застосовані

фільтри, для розрахування кількості активних фільтрів. Розглянемо що відбувається при відкритті сторінки у лістингу 3.19.

Лістинг 3.19 – Стани при відкритті сторінки

```
useLayoutEffect(() => {
   const savedFilters = localStorage.getItem("savedFilters");
   if (savedFilters) {
     const parsedFilters: FiltersType = JSON.parse(savedFilters);
     parsedFilters.skills.forEach((skill: string) => {
       dispatch(setSkills({ skill, checked: true }));
     });
     dispatch(setSize(parsedFilters.size[0] || ""));
     dispatch(setGender(parsedFilters.gender[0] || ""));
     dispatch(setSterilized(parsedFilters.sterilized));
     dispatch(setAgeCategory(parsedFilters.ageCategory[0] || ""));
     setSelectedSize(parsedFilters.size);
     setSelectedGender(parsedFilters.gender);
     setSelectedSterilized(parsedFilters.sterilized);
     setSelectedSkills(parsedFilters.skills);
     setSelectedAgeCategory(parsedFilters.ageCategory);
     setAppliedSize(parsedFilters.size);
     setAppliedGender(parsedFilters.gender);
     setAppliedSterilized(parsedFilters.sterilized);
     setAppliedSkills(parsedFilters.skills);
     setAppliedAgeCategory(parsedFilters.ageCategory);
   }
 }, []);
```

кінець лістингу 3.19

Фільтри відкриваються під час натискання на спеціальну кнопку, яку добре видно на будь-якому екрані. Ми отримуємо модальне вікно, де можемо вибрати потрібні для нас фільтри. При натисканні на кнопку Apply у нас оновлюється глобальний стан для кожного з фільтрів. Значення зберігаються в localStorage, аби зберігатись навіть при перезавантаженні сторінки [15]. Давайте розглянемо алгоритм фільтрації тварин в лістингу 3.20.

Лістинг 3.20 – Алгоритм фільтрації тварин

```
const filteredAnimals = animals.filter((animal) => {
  const matchesSize = appliedSize.length === 0 ||
appliedSize.includes(animal.size);
```

```
  const matchesGender = appliedGender.length === 0 ||
appliedGender.includes(animal.gender);
  const matchesSterilized = appliedSterilized ? animal.sterilized === true
: true;
  const matchesSkills = appliedSkills.length === 0 ||
appliedSkills.every(skill => animal.skills?.includes(skill));
  const matchesAge = appliedAgeCategory.length === 0 ||
appliedAgeCategory.includes(animal.ageCategory);
  const matchesCategory = animalCategory === "All" || animal.category ===
animalCategory;

  return matchesSize && matchesGender && matchesSterilized &&
matchesSkills && matchesAge && matchesCategory;
});
```

кінець лістингу 3.20

Бачимо, що всі фільтри комбінуються з логікою AND, тобто тваринка має ідеально підходити під кожен з фільтрів. Коли ми говоримо про навички – нам показується тільки тваринка, в якої наявні конкретні навички з вибраних.

Завдяки цій системі фільтрів користувач може зручно керувати відображенням тварин, що прискорить всиновлення тварин в декілька разів.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було успішно реалізовано мету, ціль якої – створення сучасного, зручного та функціонального веб-додатку, спрямованого на допомогу безпритульним тваринам. Платформа дозволяє користувачам перегляди доступних до всиновлення тварин, фільтрувати їх за різними параметрами, а також дізнаватись більше про діяльність притулку.

Було проаналізовано наявні веб-додатки, які спеціалізуються в цій сфері, такі як Dogcat, Happypaw, Petfinder. Це дозволило краще зрозуміти, які рішення вже існують, а також дозволило виявити як слабкі, так і сильні сторони. Цей аналіз став основою для формування вимог до власної розробки.

Для реалізації веб-додатку було обрано сучасні технології: React для створення інтерфейсу, TypeScript для забезпечення стабільності й передбачуваності коду, TailwindCSS для швидкої стилізації, а також React Redux, який був вибраний для зручного керування глобальним станом додатку. Використання цих технологій дозволило створити швидкий, масштабований і зручний у підтримці додаток.

Одним з ключових функціональних елементів веб-додатку став розширений фільтр, який дозволяє користувачам легко знаходити тварин за різними критеріями. Його реалізовано у вигляді зручного модального вікна, яке відкривається за потребою і дозволяю комбінувати декілька параметрів одночасно. Такий підхід полегшує процес пошуку і збільшує шанси тварин знайти новий дім.

Також було створено окрему сторінку з інформацією про притулок, з яким пов’язаний веб-додаток. Це дозволяє користувачам ознайомитись не лише із загальною інформацією про притулок, але й переглянути його місцерозташування на карті.

Особливу увагу приділено навігації та загальному UX/UI-дизайну. Просте, інтуїтивне перемикання між сторінками та приємний візуальний стиль створюють позитивний досвід користувача. Навігація побудована на основі

логічної структурованості компонентів, що робить взаємодію з веб-додатком максимально комфортною.

Отже, розроблена система повністю відповідає постановленим завданням і вимогам. Вона є стабільною, оптимізованою, зручною для користувача, та готовою до інтеграції з додатковими сервісами. Проект має потенціал до масштабування й подальшого розвитку, зокрема через додавання особистих кабінетів користувачів або впровадження системи сповіщень. Загалом, всі поставлені задачі на кваліфікаційну роботу були успішно виконані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Укуси, нападі, сказ: чому у Кропивницькому влада не будує притулок для тварин. URL: https://suspilne.media/kropyvnytskiy/724520-ukusi-napadi-skaz-kropivnicka-miskrada-poki-ne-moze-rozvazati-problemu-bezpritulnih-sobak/ (дата звернення: 24.03.2025).

2. Taylor Swift donates to Tennessee animal shelter. URL: https://www.axios.com/local/nashville/2023/01/19/taylor-swift-donates-tennessee-animal-shelter?utm_source=chatgpt.com (дата звернення: 24.03.2025).

3. Petfinder. URL: https://www.petfinder.com/ (дата звернення: 26.03.2025).

4. Dogcat. URL: https://dogcat.com.ua/ (дата звернення: 26.03.2025).

5. Happypaw. URL: https://happypaw.ua/ua (дата звернення: 26.03.2025).

6. React. URL: https://react.dev/ (дата звернення: 02.04.2025).

7. TypeScript. URL: https://www.typescriptlang.org/ (дата звернення: 02.04.2025).

8. React Redux. URL: https://react-redux.js.org/ (дата звернення: 02.04.2025).

9. TailwindCSS. URL: https://tailwindcss.com/ (дата звернення: 02.04.2025).

10. Framer Motion. URL: https://motion.dev/ (дата звернення: 02.04.2025).

11. Single Page App. URL: https://reactrouter.com/how-to/spa (дата звернення: 04.04.2025).

12. GetForm. URL: https://getform.io/ (дата звернення: 04.04.2025).

13. Application Programming Interface. URL: https://corefy.com/uk/glossary/api (дата звернення: 06.04.2025).

14. Поняття валідації в програмній інженерії. URL: https://en.wikipedia.org/wiki/Data_validation (дата звернення: 22.04.2025).

15. Localstorage та sessionStorage. URL: https://uk.javascript.info/localstorage (дата звернення: 23.04.2025).