

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

**«Вебплатформа для пошуку господарів для
безпритульних тварин з персоналізованими
обліковими записами»**

Виконала студентка групи ВТ-21-1
ГАМАРСЬКА Дар'я Ігорівна

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
ЧИЖМОТРЯ Олексій Володимирович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«26» лютого 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу

Здобувач вищої освіти: **ГАМАРСЬКА Дар'я Ігорівна**

Керівник роботи: **САВІЦЬКИЙ Роман Святославович**

Тема роботи: **«Вебплатформа для пошуку господарів для безпритульних тварин з персоналізованими обліковими записами»**,

затверджена Наказом закладу вищої освіти №61/с від **«14» лютого 2025 р.**

Вихідні дані для роботи: **Об'єктом дослідження** є інформаційна система, що забезпечує взаємодію між користувачами з метою пошуку та адаптації тварин.

Предметом дослідження є архітектура, функціональні можливості та програмна реалізація платформи, що забезпечує роботу з оголошеннями, медичними даними тварин, інтеграцією з платіжними сервісами та підтримкою багатомовного інтерфейсу.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Савіцький Р.С.	27.02.2025р.	27.02.2025р.
2	Савіцький Р.С.	27.02.2025р.	27.02.2025р.
3	Савіцький Р.С.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебзастосунку для пошуку та адаптації тварин і пояснювальної записки. Пояснювальна записка до випускної роботи містить 86 сторінок, 39 ілюстрацій, 8 таблиць та 5 додатків.

Метою роботи є створення платформи, що полегшує пошук і адаптацію тварин, з можливістю публікації оголошень, здійснення донатів, фільтрації за категоріями, а також зручної взаємодії для всіх типів користувачів.

Архітектура застосунку передбачає розділення на серверну та клієнтську частину. Реалізовано підтримку автентифікації, багатомовного інтерфейсу, додавання та перегляду оголошень, фільтрації за категоріями (“Віддам безкоштовно”, “Загублені/знайдені”), функції обраного, а також інтеграцію з Stripe API для прийому благодійних внесків.

Форма оголошення поділена на етапи для зручності користувача, підтримується валідація, завантаження фотографій та історії щеплень тварин. Інтерфейс є адаптивним і забезпечує комфортну роботу з будь-яких пристроїв – мобільних, планшетів і десктопів.

КЛЮЧОВІ СЛОВА: ТВАРИНИ, ОГОЛОШЕННЯ, ДОПОМОГА, ДОНАТ, ПОШУК, УСИНОВЛЕННЯ, ПОШУК, ВЕБПЛАТФОРМА.

					ІПЗ.КР.Б. – 121 – 25 – ПЗ			
		№ докум.	Підпис					
Розробник.	Гамарська Д.І.				Вебплатформа для пошуку господарів для безпритульних тварин з персоналізованими обліковими записами	Літ.	Арк.	Аркуші
Керівник	Савіцький Р.С.						3	87
Н. контр.	Тичина О. П.					Житомирська політехніка, група Вм-21-1		
Зав. каф.	Вакалюк Т. А.							

ABSTRACT

The bachelor's qualification thesis consists of a web application for pet search and adoption, as well as an explanatory note. The explanatory note contains 86 pages, 39 illustrations, 8 tables and 5 applications.

The aim of this project is to develop a user-friendly platform that facilitates the search and adoption of animals, allowing users to publish notices, make donations, filter by categories, and interact efficiently with the system.

The application is built using React, SCSS, and a Node.js / Express backend, with MongoDB as the database. The platform includes authentication, multilingual support, creation and browsing of notices, category filtering (“For Free”, “Lost/Found”), a favorites system, and integration with the Stripe API for receiving charitable donations.

The notice form is divided into steps for better usability and includes support for validation, photo uploads, and entry of vaccination history. The interface is fully responsive and optimized for use across devices: mobile phones, tablets, and desktops.

KEYWORDS: ANIMALS, NOTICE, ADOPTION, DONATION, SEARCH, PLATFORM, WEB APPLICATION.

					ІІЗ.КР.Б. – 121 – 25 – ІІЗ	
		№ докум.	Підпис			4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕХНІЧНИХ РІШЕНЬ.....	10
1.1 Огляд стану проблеми адаптації та пошуку тварин	10
1.2 Аналіз існуючих платформ-аналогів	13
1.3 Визначення загальних вимог до платформи	19
1.4 Постановка технічного завдання	20
Висновки до першого розділу.	22
РОЗДІЛ 2. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	23
2.1 Загальна структура та архітектура платформи	23
2.2 Структура бази даних	26
2.3 Реалізація API для взаємодії фронтенду та бекенду	32
Висновки до другого розділу.....	42
РОЗДІЛ 3. ІНТЕРФЕЙС ТА КОРИСТУВАННЯ ПРОГРАМНИМ ПРОДУКТОМ.....	43
3.1 Користувацький інтерфейс і адаптивність верстки	43
3.2 Система розгортання застосунку.....	60
3.3 Тестування функціоналу платформи	62
Висновки до третього розділу	67
ВИСНОВКИ.....	69
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	71

					ІІЗ.КР.Б. – 121 – 25 – ПЗ	
		№ докум.	Підпис			6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Back-end – серверна частина.

CRUD (англ. create read update delete) – основні функції управління даними «створення, читання, оновлення і вилучення».

CSS (англ. Cascading Style Sheets) – мова стилів.

DOM (англ. Document Object Model) – об'єктна модель документа.

Інтерфейс користувача (анг. Front-end) – це інтерфейс для взаємодії між користувачем і серверною частиною.

HTML (англ. HyperText Markup Language) – мова розмітки документів для перегляду вебсторінок у браузері.

HTTP (англ. HyperText Transfer Protocol) – протокол передачі даних, що використовується в комп'ютерних мережах.

JSX – розширення синтаксису JavaScript

JWT (JSON Web Token) – формат токенів для безпечної передачі інформації та автентифікації.

REST API – стиль архітектури API, що ґрунтується на використанні HTTP-запитів.

Stripe – платіжна система, яка дозволяє приймати онлайн-оплати.

UI (англ. User Interface) – інтерфейс користувача.

UX (англ. User Experience) – досвід користування.

БД – база даних.

СКБД – система керування базою даних.

UPAW (англ. World Animal Association of Ukraine) – Всесвітня асоціація з тварин України UPAW

ВСТУП

Актуальність теми зумовлена необхідністю удосконалення способів пошуку, адаптації та допомоги тваринам у цифровому середовищі. Щороку тисячі тварин опиняються без домівки або втрачаються, а наявні сервіси часто не забезпечують достатньої ефективності, зручності та доступності у взаємодії між волонтерами, власниками та потенційними новими господарями. В умовах стрімкого розвитку вебтехнологій зростає потреба в єдиній цифровій платформі, яка б об'єднала ключові функції: публікацію оголошень, підтримку пошуку за категоріями, прийом благодійних внесків, а також зручний інтерфейс для всіх учасників процесу.

Практична складова теми полягає у створенні інструменту, який об'єднає зусилля волонтерів, власників тварин, ветеринарів та небайдужих громадян. Вебплатформа забезпечить доступ до інформації і зменшить час на пошук втрачених тварин. Також полегшить процес адаптації безпритульних тварин. Інтеграція можливостей для донатів допоможе допомагати небайдужим людям.

Метою випускної кваліфікаційної роботи бакалавра є розробка вебплатформи, яка дозволяє користувачам додавати оголошення про тварин, здійснювати їх пошук, фільтрувати за категоріями, додавати улюблені, переглядати інформацію про вакцинації, а також робити донати для підтримки волонтерських ініціатив.

Для реалізації поставленої мети необхідно вирішити завдання:

1. Провести аналіз існуючих рішень для пошуку, прилаштування та допомоги тваринам, виявити їхні переваги та недоліки.
2. Визначити технології, найбільш придатні для реалізації інтерактивної платформи з урахуванням вимог до безпеки, масштабованості, адаптивності та доступності.
3. Реалізувати платформу для створення та перегляду оголошень, розробити фільтрацію, підтримку категорій, мультимовність та функцію

“обране”, інтегрувати платіжну систему (Stripe API) для збору благодійних внесків.

4. Розробити архітектуру проекту. Ключовими моментами є база даних, серверна та клієнтська частина та API для інтеграції Stripe.

5. Забезпечити адаптивний інтерфейс, щоб могли працювати на різних пристроях.

6. Розробити механізми автентифікації та авторизації для забезпечення безпеки даних користувачів вебплатформи.

7. Провести аналіз продуктивності платформи.

8. Провести тестування платформи.

Об’єктом дослідження є інформаційна система, що забезпечує взаємодію між користувачами з метою пошуку та адаптації тварин.

Предметом дослідження є архітектура, функціональні можливості та програмна реалізація платформи, що забезпечує роботу з оголошеннями, медичними даними тварин, інтеграцією з платіжними сервісами та підтримкою багатомовного інтерфейсу.

Під час реалізації використано стек технологій: React, SCSS, Node.js / Express, MongoDB, Stripe API, react-hook-form, з урахуванням потреб у швидкодії, безпеці та зручності користувача.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕХНІЧНИХ РІШЕНЬ

1.1. Огляд стану проблеми адаптації та пошуку тварин

Проблема безпритульних тварин є однією з актуальних соціальних та екологічних викликів сучасного світу. Щороку тисячі собак і котів опиняються на вулиці через безвідповідальність власників, відсутність ефективних регулюючих механізмів та низький рівень культури адопції [1]. Ці тварини стикаються з небезпекою голоду, хвороб та жорстокого поводження, що робить пошук нових власників критично важливим завданням.

Водночас багато людей прагнуть завести домашнього улюбленця, але через відсутність єдиної централізованої системи часто звертаються до неофіційних джерел або купують тварин у розплідниках, замість того, щоб надати шанс безпритульним. Це підсилює проблему перенасичення притулків і зменшує рівень адопції.

У світі існують різні ініціативи та технологічні рішення для сприяння пошуку домівок для тварин, зокрема спеціалізовані онлайн-платформи, мобільні застосунки та волонтерські програми. Проте багато з них мають обмежену функціональність, низький рівень охоплення або проблеми з оновленням бази тварин.

Проблема безпритульних тварин в Україні залишається гострою соціальною та екологічною загрозою. За оцінками Всесвітньої асоціації з тварин України кількість безпритульних тварин сягає приблизно 140 тисяч [2]. Україна входить до десяти країн світу з найбільшою чисельністю безпритульних тварин, що свідчить про значні прогалини в системі регулювання, обліку та контролю за домашніми улюбленцями.

Особливо різке зростання чисельності безпритульних тварин відбулося внаслідок повномасштабного вторгнення. За даними організації UAnimals, тільки за перший рік війни кількість котів і собак у притулках зросла на 60%. Щоденна евакуація тварин із зон бойових дій не встигає компенсувати їхній неконтрольований приріст, а місць у притулках не вистачає.

Основними причинами збільшення кількості безпритульних тварин є безвідповідальність власників – за даними комунальних служб Львова, 84% тварин у муніципальному центрі відлову були викинуті господарями. Втрата тварин – близько 11% тварин опинилися на вулиці через те, що загубилися. Природне розмноження безпритульних тварин становить лише 5% тварин є народженими на вулиці.

Низький рівень обізнаності власників про необхідність реєстрації та вакцинації домашніх улюбленців погіршує ситуацію. За даними місцевих ініціатив, 40% власників зареєстрованих тварин не знали про обов’язкову вакцинацію від сказу або ніколи її не робили.

В Україні досі немає централізованого державного реєстру домашніх тварин. Це ускладнює контроль за їхнім переміщенням, ідентифікацію загублених тварин та боротьбу з безвідповідальним ставленням власників. Деякі міста, зокрема Львів, створили локальні бази даних, однак на загальнонаціональному рівні така система відсутня.

Лікар ветеринарної медицини Ярослав Тищенко наголошує, що без належного контролю Україна залишається лідером у Європі за рівнем захворюваності тварин на сказ. Щорічно реєструються тисячі випадків нападів заражених тварин на людей. Наприклад, у Харківській області у 2023 році зафіксовано понад 2000 звернень щодо укусів тваринами, хворими на сказ, що на 25% більше, ніж у попередньому році [3].

Попри складну ситуацію, в Україні діють різні волонтерські та технологічні ініціативи, спрямовані на допомогу тваринам такі, як притулки та волонтерські організації. Багато громадських об’єднань займаються порятунком та пристроєм тварин у нові родини. Онлайн-платформи для адопції, цифрові сервіси, що допомагають знаходити власників для безпритульних тварин, проте вони не завжди ефективні через обмежену функціональність і відсутність єдиної бази даних. Також наявні освітні кампанії, які проводять заходи щодо популяризації відповідального ставлення

до тварин, однак їхня ефективність поки що недостатня для кардинальних змін.

Розглянемо основні труднощі та бар'єри, з якими стикаються як власники тварин, так і організації, що займаються їхнім утриманням та розміщенням (рис.1.1).

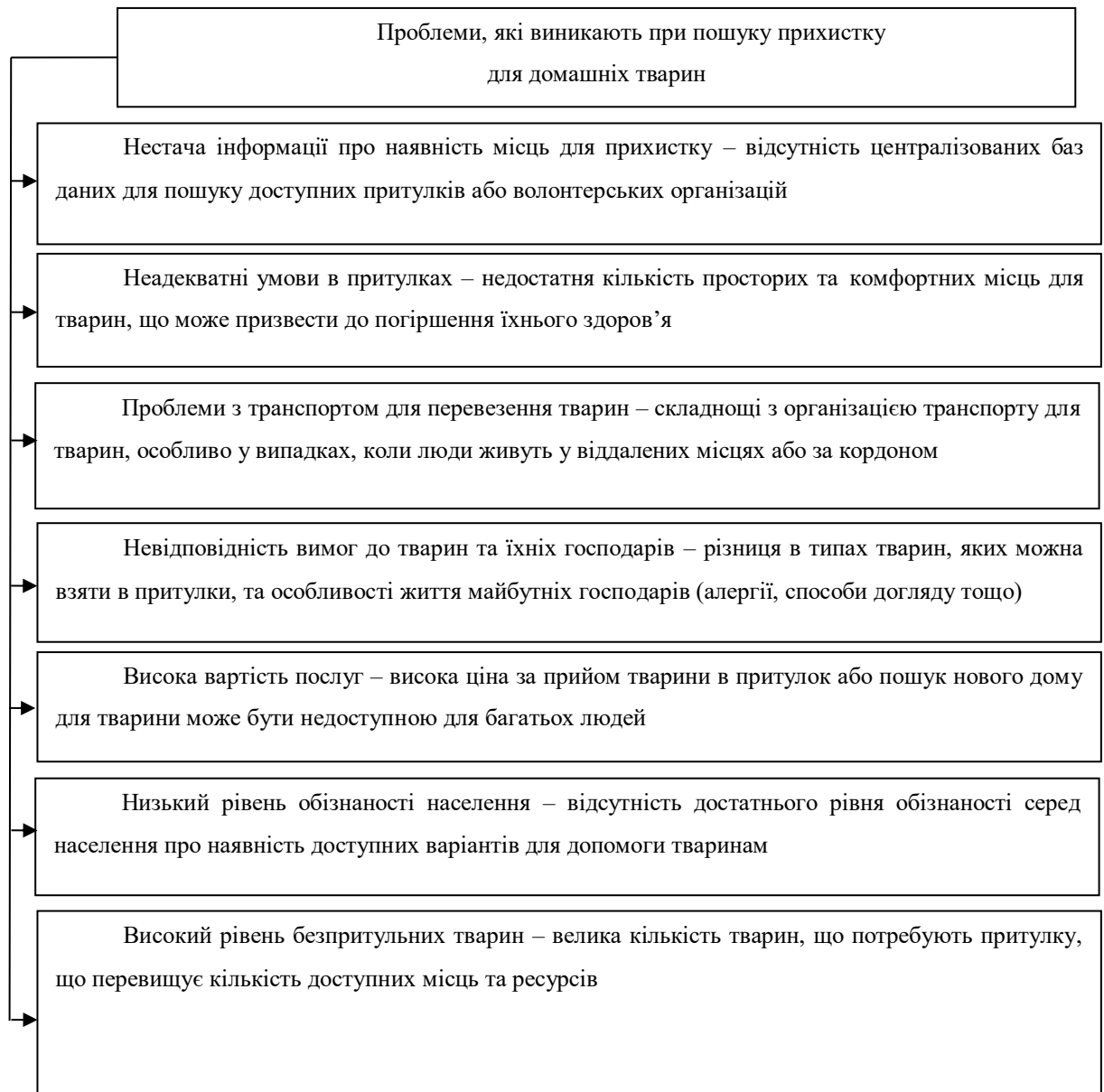


Рис.1.1 - Проблеми, які виникають при пошуку притулку для домашніх тварин

Розробка платформи для купівлі, продажу та пошуку нових господарів для домашніх тварин, а також для налагодження зв'язку між ними, є важливим кроком з таких причин, як зручність із зростанням популярності електронної

комерції все більше людей використовують Інтернет для здійснення покупок, включаючи купівлю або пошук прихистку для домашніх тварин. Платформа дає можливість любителям тварин знайти підходящий варіант без необхідності виходити з дому. Розширене охоплення платформа може охопити значно більшу аудиторію, ніж традиційний магазин. Потенційні власники домашніх тварин з будь-якої точки світу мають змогу знайти свого майбутнього друга. Поліпшений догляд за тваринами добре спроектована платформа може надавати корисну інформацію про догляд, хвороби та харчування тварин, що дозволить потенційним власникам приймати більш усвідомлені рішення, вибираючи тварину, яка підходить їхньому способу життя. Інформація про загублених та знайдених тварин: власники загублених тварин можуть оперативно створювати оголошення про пошук своїх улюбленців, а ті, хто знайде тварину, можуть розмістити повідомлення про її знаходження.

Отже, створення платформи, який об'єднує зацікавлені сторони, зокрема для пошуку прихистку, купівлі-продажу тварин та покращення комунікації між ними, є надзвичайно актуальним.

1.2. Аналіз існуючих платформ-аналогів

Аналіз існуючих платформ-аналогів є важливим етапом у процесі розробки нової платформи, оскільки він дозволяє визначити функціонал, який є найбільш затребуваним, виявити недоліки конкурентних рішень та адаптувати найкращі практики. Мета цього аналізу полягає в тому, щоб створити продукт, який би відповідав очікуванням користувачів і вирішував їхні потреби більш ефективно, ніж існуючі рішення.

Для аналізу ми обрали три платформи, що мають схожий функціонал: Petfinder, Сіріус та OLX (категорія “Тварини”). Критерії аналізу включають оцінку функціональності, переваг, недоліків, а також доступності платформ на різних пристроях.

Petfinder – це міжнародна платформа, яка надає користувачам доступ до великої бази тварин із притулків, переважно в США та Канаді. Основна

аудиторія — люди, які шукають домашніх улюбленців. Платформа доступна як через вебсайт, так і через мобільний додаток. Ця платформа має один із найширших наборів функцій серед аналізованих рішень. Вона дозволяє користувачам використовувати розширений пошук із фільтрами за віком, породою, статтю, розміром та місцезнаходженням. Інтегрувати пошук із картою, що значно спрощує визначення місцезнаходження притулків. Зберігати тварин у "список улюбленців" для подальшого перегляду. Подавати заявки на адаптацію через інтерактивну форму. Працювати через мобільний додаток, що підвищує доступність платформи (Рис.1.2).

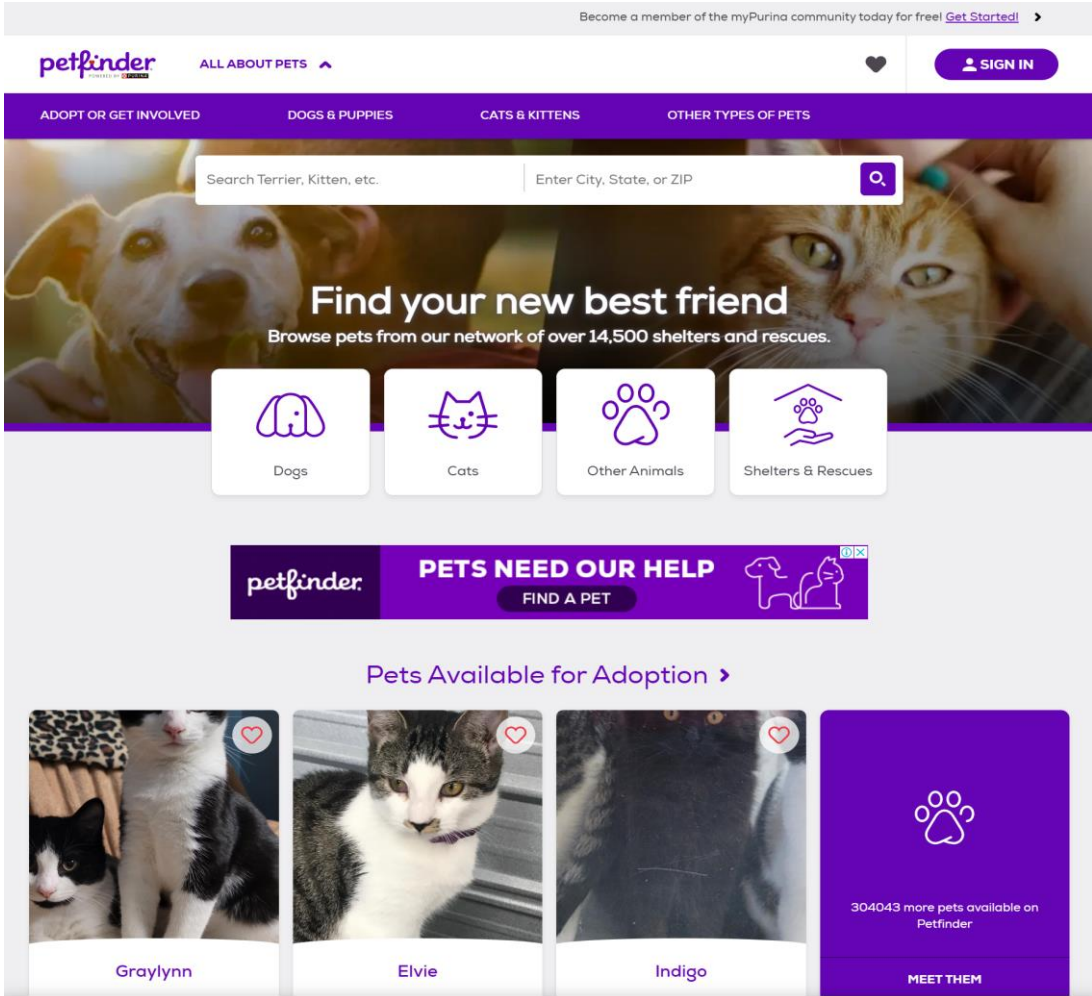


Рис.1.2. - Міжнародна платформа Petfinder

Сіріус – це українська платформа, яка зосереджується на пошуку та адаптації домашніх тварин. Основна цільова аудиторія – українські користувачі, які шукають улюбленців або хочуть допомогти притулкам із передачею тварин. Дана платформа пропонує доступ через вебсайт. Сіріус

платформою з базовим функціоналом, орієнтованою на локальний український ринок. Вона забезпечує простий пошук із базовими фільтрами (вік, стать, статус тварини). Сучасний адаптивний інтерфейс, який добре працює на різних пристроях. Зручну подачу інформації про тварин у вигляді карток із фото, описом та параметрами. А також має орієнтацію на українські притулки, що робить її актуальною для локальних користувачів. (Рис.1.3).

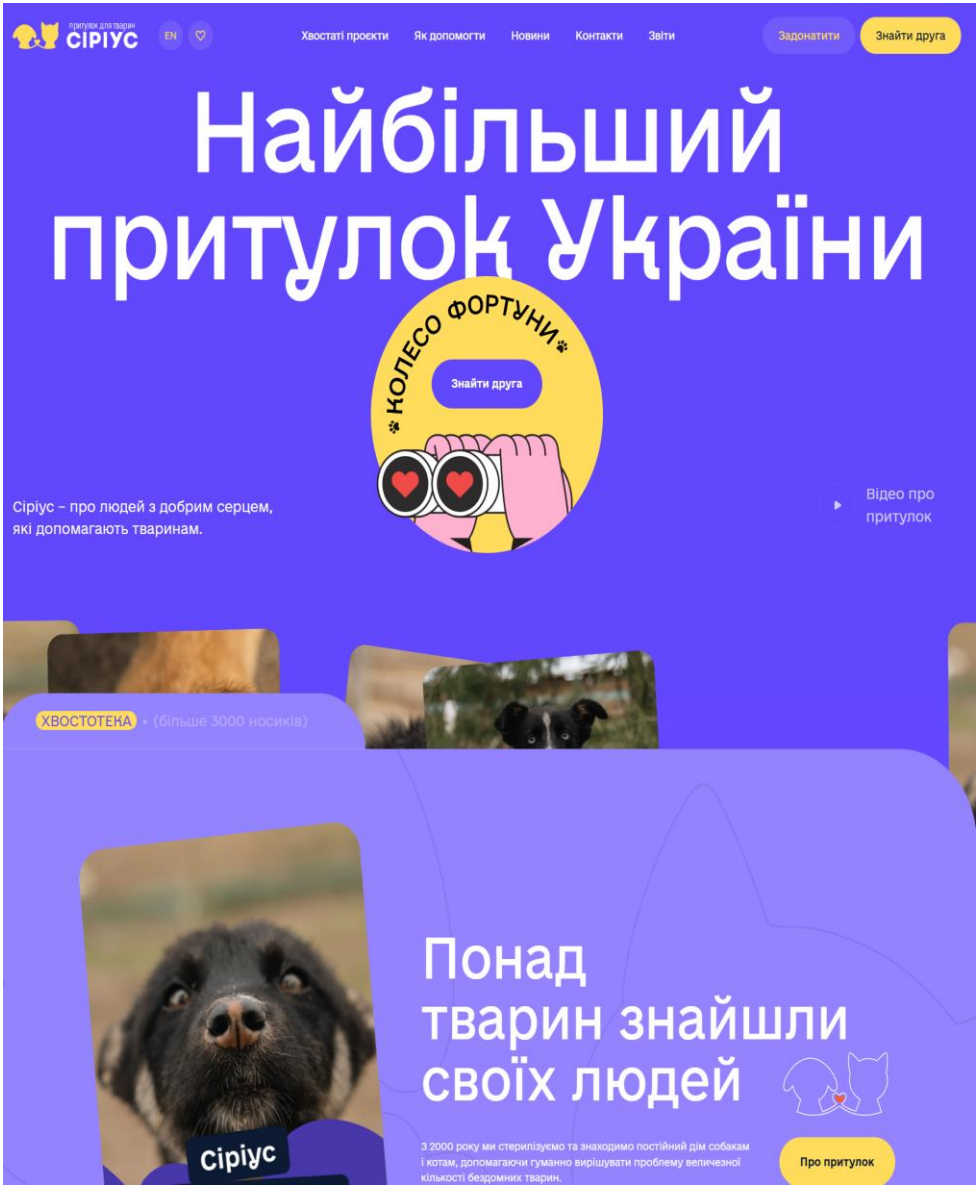


Рис.1.3. - Українська платформа Cіріус

OLX – це міжнародна платформа оголошень, яка включає категорію “Тварини”. Основна аудиторія – це приватні особи, які шукають або продають тварин. OLX має як вебсайт, так і мобільний додаток. OLX це універсальна платформа, що дозволяє створювати оголошення про продаж або передачу

тварин із мінімальною кількістю даних. Використовувати пошук за ключовими словами та базовими категоріями. Виконувати прямий зв’язок між користувачами через чат. І мати доступ до платформи через мобільний додаток. (Рис.1.4).

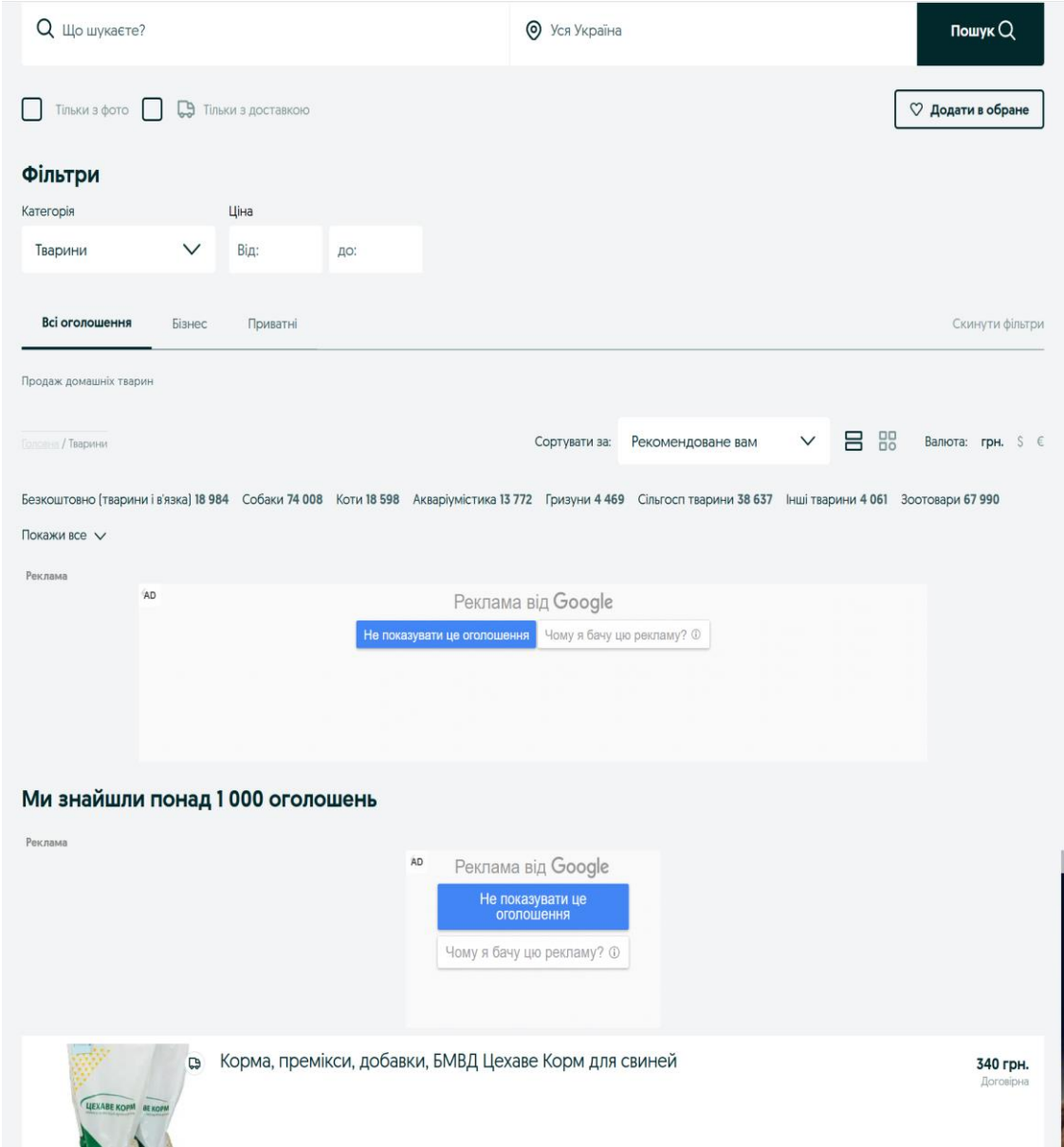


Рис.1.4. - Міжнародна платформа OLX

Home4pets є платформою адаптованою під український ринок, що дозволяє створювати оголошення з детальною інформацією, включаючи вік, стать, породу, місце розташування, контактні дані. Використовувати фільтри для пошуку оголошень. Можливість додавання історії щеплення тварин. Можливість відкривати карточку тварини для детального перегляду. І також можливість генерації документів для всиновлення тварини (Рис.1.5).

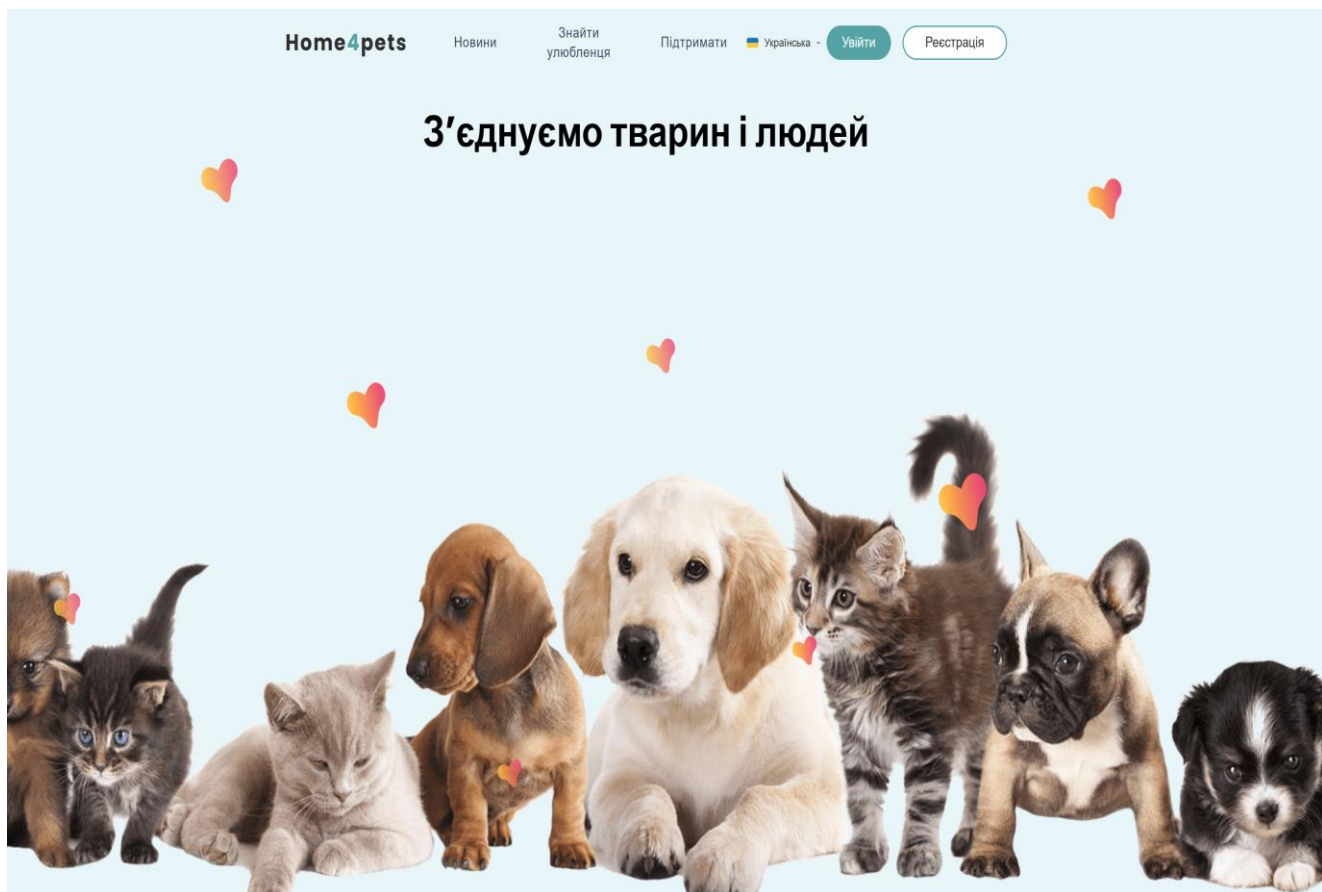


Рис.1.5. - Платформа Home4pets

Перед аналізом переваг та недоліків платформ, важливо зрозуміти функціональність кожної з них, оскільки саме вона визначає, наскільки ефективно платформа вирішує завдання пошуку та адаптації тварин.

Аналіз функціональності платформ дозволяє визначити їх сильні та слабкі сторони. На основі отриманих даних, ми можемо виділити ключові переваги, які слід врахувати під час розробки власної платформи, а також недоліки, яких потрібно уникнути. Порівняння здійснюється за основними критеріями, що впливають на ефективність роботи платформи, такими як зручність використання, гнучкість пошуку, інтеграція додаткових функцій, підтримка локальної аудиторії та доступність на різних пристроях.

Таблиця 1.1 відображає переваги та недоліки платформ Petfinder, DogCat та OLX (категорія “Тварини”).

Таблиця 1.1

Переваги та недоліки існуючих платформ аналогів

Критерії	Petfinder	Cipiyс	OLX	Home4Pets
Мобільний додаток	Є	Немає	Є	Немає
Інтерфейс	Зручний, сучасний	Адаптивний, зручний	Мінімальний	Сучасний, інтуїтивно зрозумілий
Пошукові фільтри	Базові	Базові	Відсутні	За статусом
Інтеграція з картою	Є	Немає	Немає	Немає
Орієнтація	Міжнародна з геообмеженням	Локальна	Міжнародна	Локальна
Аудиторія	Середня	Локальна	Широка	Середня
Функціонал подачі оголошень	Розширений	Зручний	Дуже простий	Повний з документами та щепленням

Таблиця демонструє, що кожна платформа має свої сильні сторони, але й певні обмеження. Petfinder виграє завдяки широкому функціоналу, але його географічне обмеження та складність фільтрів можуть бути проблемою. DogCat забезпечує зручність і локальну актуальність, але потребує мобільного додатку та розширеного функціоналу. OLX має широку аудиторію, але не є спеціалізованою платформою для тварин. Ці аспекти враховуватимуться при розробці нової платформи.

На основі аналізу можна зробити кілька важливих висновків, які допоможуть у розробці власної платформи:

1. Необхідний функціонал можливість створення оголошень як притулками, так і приватними особами. Збереження обраних тварин у “список улюбленців”. Підключення Stripe для збору благодійних внесків на допомогу тваринам. І також форма подачі оголошення з кількома кроками, включаючи завантаження фото, історії щеплень.
2. Варто уникати надмірну складність інтерфейсу, що може ускладнити користування платформою. Обмежити географічне охоплення, щоб зробити платформу доступною для більшої аудиторії.
3. Додатковими можливостями може бути адаптивний інтерфейс, що дозволяє комфортно працювати з мобільних і десктопних пристроїв. Багатомовність, для зручності користувачів із різними мовними вподобаннями. Освітній контент для майбутніх власників (наприклад, статті про догляд за тваринами).

Аналіз платформ-аналогів показав, що, незважаючи на існування рішень із подібним функціоналом, жодна з них не є ідеальною. Petfinder забезпечує широкий функціонал, але обмежений географічно; DogCat має зручний дизайн, але базову функціональність; OLX надає велику аудиторію, але відсутність спеціалізації створює проблеми. Розроблювана платформа має поєднати найкращі риси рішень, додавши унікальний, що зробить її зручною для українського ринку.

1.3. Визначення загальних вимог до платформи

Створення інтерактивної вебплатформи, яка забезпечить зручний доступ до інформації про тварин, що потребують адаптації, з можливістю пошуку, фільтрації, додавання нових записів і редагування існуючих. Платформа повинна бути доступною для використання як приватними особами, так і волонтерами.

Реалізувати такі функції:

1. Архітектура платформи реалізовується організацією Frontend та Backend із дотриманням принципів RESTful API [4]. Реалізацію аутентифікації користувачів із використанням JSON Web Token (JWT) [5].

2. Базовий шар бази даних реалізовано на основі MongoDB [6]. Створення моделей даних для збереження інформації про тварин, користувачів та оголошень. Реалізацію CRUD-операцій (створення, читання, оновлення, видалення) для роботи з базою даних.

3. Рівень бізнес-логіки включає можливість пошуку тварин за ключовими словами. Категоризацію оголошень на типи (“Віддам безкоштовно”, “Загублені/знайдені”). І також додавання тварин до списку “улюбленців”.

4. Функціональність платформи охоплює створення та перегляд оголошень з тваринами, які містять фото, ім’я, вік, породу, стать, опис, інформацію про щеплення, особливості поведінки. Можливість додавання інформації про вакцинацію та формування PDF-документу про тварину для потенційного господаря або притулку. Додавання фото та редагування картки тварини. Та інтеграцію з Stripe API [7] для збору благодійних внесків. Підтримка двох мов інтерфейсу - української та англійської.

5. Нефункціональність платформи включає створення адаптивного дизайну, що дає змогу працювати на різних пристроях. Інтерфейс має бути інтуїтивно зрозумілим для широкого кола користувачів. Дані про тварин мають бути доступні без потреби авторизації, а саме перегляд оголошень. Також повинно бути централізоване збереження даних у базі MongoDB.

1.4. Постановка технічного завдання

Розробка вебплатформи для пошуку та адаптації тварин є актуальним напрямом, спрямованим на вирішення проблеми безпритульності тварин шляхом створення ефективного механізму взаємодії між притулками, волонтерами та потенційними власниками. Основною метою проєкту є створення інтуїтивно зрозумілої цифрової системи, яка дозволить автоматизувати та спростити процес пошуку нових домівок для тварин і водночас підвищити рівень соціальної відповідальності.

На платформі буде реалізовано можливість створення облікових записів користувачів із подальшою автентифікацією через електронну пошту.

Користувачі можуть створювати оголошення про тварин, редагувати їх, переглядати оголошення інших користувачів, а також додавати тварин до списку «улюбленців». Додатково передбачена функція формування PDF-документа на основі заповненої інформації про тварину – для зручного роздруку або пересилання. Платформа підтримує багатокрокову форму створення оголошення з валідацією даних, можливістю додати фотографії тварини, а також вказати її особливості поведінки, стан здоров'я, наявність щеплень та інші характеристики.

З метою сприяння волонтерським ініціативам та організаціям передбачено інтеграцію платіжної системи Stripe, яка дозволяє здійснювати благодійні внески безпосередньо через платформу. Пошук інформації на сайті реалізується через систему фільтрації за категоріями (наприклад, «віддам безкоштовно», «загублені/знайдені»), а також за ключовими словами. Архітектура платформи побудована з урахуванням принципів REST і поділена на клієнтську частину, реалізовану з використанням бібліотеки React.js [8] із застосуванням модульної SCSS-стилізації, та серверну частину на базі Node.js [9] із Express [10]. Вся інформація зберігається у нереляційній базі даних MongoDB, що дозволяє гнучко оперувати структурами даних.

Окрему увагу приділено питанням безпеки та продуктивності. Використання токенів JSON Web Token (JWT) забезпечує надійний захист особистих даних користувачів. Важливо також, що платформа є багатомовною та підтримує як українську, так і англійську мови, що розширює її аудиторію. Інтерфейс системи адаптовано під різні типи пристроїв, включаючи смартфони, планшети та комп'ютери, що гарантує комфортну взаємодію незалежно від технічних умов користувача.

У результаті реалізації цього технічного завдання буде створено сучасну вебплатформу, яка значно підвищить ефективність пошуку та адаптації тварин, дозволяючи швидко знаходити, переглядати й поширювати інформацію про них. Таким чином, розроблена система стане дієвим

інструментом у сфері допомоги безпритульним тваринам і сприятиме зменшенню їх кількості.

Висновки до першого розділу.

У результаті аналізу предметної області було з'ясовано, що проблема безпритульних тварин в Україні має значні соціальні та екологічні масштаби, зумовлені низьким рівнем культури адопції, відсутністю централізованих баз даних, слабкою інформованістю населення та браком ефективних технологічних інструментів. Значне зростання кількості тварин у притулках після повномасштабного вторгнення лише підсилило потребу в цифрових рішеннях для координації дій волонтерів, притулків і майбутніх власників.

У процесі дослідження проведено глибокий порівняльний аналіз існуючих аналогів – зокрема платформ Petfinder, DogCat і OLX. Встановлено, що хоча деякі з них мають окремі корисні функції (наприклад, пошук із фільтрами або адаптивний інтерфейс), жодна не поєднує в собі повний спектр можливостей, орієнтованих на українських користувачів, включно з локалізацією, донатами, системою адаптації й централізованим управлінням базою тварин.

На основі вивчення проблем і можливостей сформульовано вимоги до нової вебплатформи, серед яких: адаптивний дизайн та зручний UI/UX; реалізація особистого кабінету користувача; інтеграція з платіжною системою (Stripe) для прийому благодійних внесків; формування оголошення з можливістю прикріплення фото, медичної інформації, опису поведінки та особливих потреб тварини.

Також визначено набір технологій, оптимальних для реалізації проєкту, зокрема: React.js, Node.js, MongoDB, Mongoose, JWT, SCSS, REST API. З технічної точки зору особливу увагу приділено безпеці, продуктивності та масштабованості системи. Таким чином, результати першого розділу дали змогу сформулювати чітке бачення архітектури майбутнього застосунку, технічних рішень і функціональних модулів, що ляжуть в основу ефективної цифрової платформи для пошуку та адаптації тварин.

РОЗДІЛ 2. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Загальна структура та архітектура платформи

Архітектура програмного продукту є основою для його ефективного функціонування, масштабованості та підтримки. У процесі розробки було визначено, що система повинна забезпечувати зручний пошук, адаптацію та управління інформацією про тварин, а також надавати користувачам змогу швидко взаємодіяти з платформою. Основною метою створення продукту стало розроблення стабільного та зручного сервісу, який би поєднував гнучкість клієнтської частини та надійність серверної логіки.

На етапі аналізу вимог було визначено ключові функціональні та нефункціональні вимоги до системи. Вона повинна підтримувати створення профілів користувачів, управління оголошеннями, взаємодію між потенційними власниками та поточними власниками тварин. Важливою частиною стала безпека даних, швидкість роботи та можливість масштабування. Було враховано, що додаток має працювати у вебсередовищі, бути адаптивним і відповідати сучасним стандартам UI/UX.

Для реалізації архітектурного рішення було обрано серверно-клієнтську модель, яка передбачає розділення серверної частини, що займається обробкою запитів та управлінням базою даних, і клієнтської частини, що забезпечує взаємодію користувачів із платформою через браузерний інтерфейс. Основним підходом став REST API, який дозволяє обмінюватися даними між сервером і клієнтом у стандартному форматі JSON. Використання React.js на клієнтській стороні гарантує швидке оновлення інтерфейсу, динамічність компонентів та комфортну інтеграцію з іншими інструментами.

Серверна частина розроблена на базі Express.js [11] – легкого та гнучкого фреймворку для Node.js, що забезпечує обробку запитів, маршрутизацію та інтеграцію з базою даних. Використання Express.js дозволило створити ефективну серверну логіку з розподілом функціональності на окремі модулі, що покращує підтримку коду та його масштабованість. Для роботи з базою даних обрано MongoDB – NoSQL-сховище, що добре підходить для гнучких

структур даних і легко масштабується. Взаємодія з MongoDB здійснюється через бібліотеку Mongoose [12], яка спрощує роботу з моделями та надає зручний API для взаємодії з базою.

Клієнтська частина побудована за допомогою React.js [13], що дозволяє створювати багаторазово використовувані компоненти, які динамічно змінюються залежно від стану додатку. Для управління станом застосовується Redux Toolkit [14], що спрощує процес передачі даних між компонентами та забезпечує централізоване сховище інформації. Redux Toolkit використовується для оптимізації роботи зі станом, що зменшує складність роботи з Redux та робить код чистішим та зрозумілішим. Крім того, для маршрутизації в межах клієнтської частини застосовується React Router [15], що дозволяє зручно працювати з динамічними маршрутами та управляти переходами між сторінками.

База даних побудована за схемою з чітко визначеними колекціями для користувачів, оголошень, а також інформації про тварин. Використання MongoDB у поєднанні з Mongoose дозволяє визначати структуру документів, контролювати валідацію даних та реалізовувати ефективні запити для отримання необхідної інформації. Завдяки NoSQL-архітектурі забезпечується гнучкість у зміні схеми даних, що є важливим при розширенні функціональності додатку.

Щодо безпеки, було реалізовано використання JWT (JSON Web Token) [16] для аутентифікації користувачів. Кожен запит, що потребує авторизації, включає унікальний токен, що забезпечує безпечну ідентифікацію користувача. Також враховано механізми захисту від поширених атак, таких як SQL-ін'єкції [17], XSS (міжсайтовий скриптинг) [18] та CSRF (міжсайтовий підроблений запит) [19].

Система спроектована з урахуванням можливості масштабування. Це досягається використанням розподіленої бази даних, кешування запитів та горизонтального масштабування серверів. Використання Express.js дозволяє зручно впроваджувати балансування навантаження, а оптимізація запитів до MongoDB знижує навантаження на базу даних.

У процесі подальшого розвитку системи було реалізовано функціонал, спрямований на покращення взаємодії користувачів із платформою. Зокрема, було впроваджено можливість генерації індивідуальних документів у форматі PDF для кожної тварини. На серверній частині за допомогою бібліотеки PDFKit [20] створюється договір про усиновлення тварини (“Pet Adoption Agreement”), який користувач може завантажити безпосередньо з інтерфейсу сайту. Для цього було розроблено окремий маршрут API /api/notices/generate-document/:id, що забезпечує динамічне формування та відправлення документів клієнту.

На фронтенді, за допомогою React.js [21], було реалізовано інтеграцію кнопки завантаження в модальне вікно інформації про тварину. Крім того, для більш повного представлення даних, було додано секцію “Health History”, яка містить історію вакцинацій тварини у вигляді списку з датами та назвами вакцин. Таке рішення підвищує інформативність картки тварини та сприяє кращій обізнаності потенційних власників про стан здоров'я тварини.

Таким чином, розроблена архітектура поєднує в собі сучасні технології та підходи, які забезпечують високу продуктивність, масштабованість та зручність підтримки програмного продукту. Використання Express.js на сервері, React.js на клієнті, а також інтеграція з MongoDB забезпечують ефективну взаємодію між усіма частинами системи. На рисунку наведено схему архітектури (Рис.2.1).

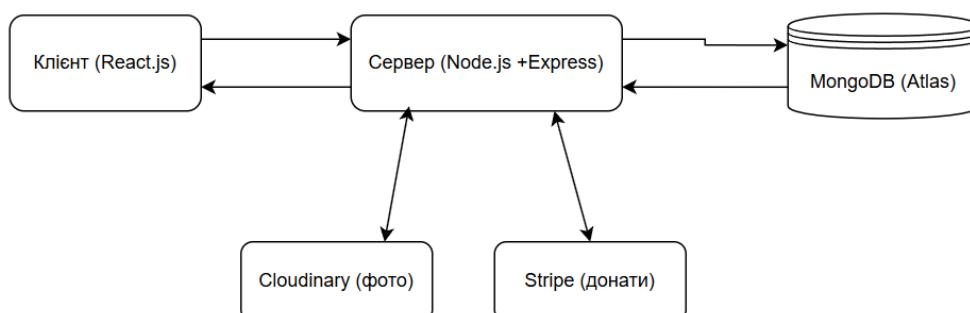


Рис. 2.1 - Схема архітектури

2.2. Структура бази даних

База даних, що використовується для функціонування платформи, є основним сховищем інформації, необхідної для забезпечення її роботи. Вона реалізована за допомогою MongoDB, яка є NoSQL-базою даних, що дозволяє зберігати інформацію у вигляді колекцій документів у форматі JSON. Для зручності взаємодії з базою використовується ORM-бібліотека Mongoose, яка надає можливість визначати схеми даних, проводити валідацію та забезпечувати взаємозв'язки між об'єктами (Рис.2.2).

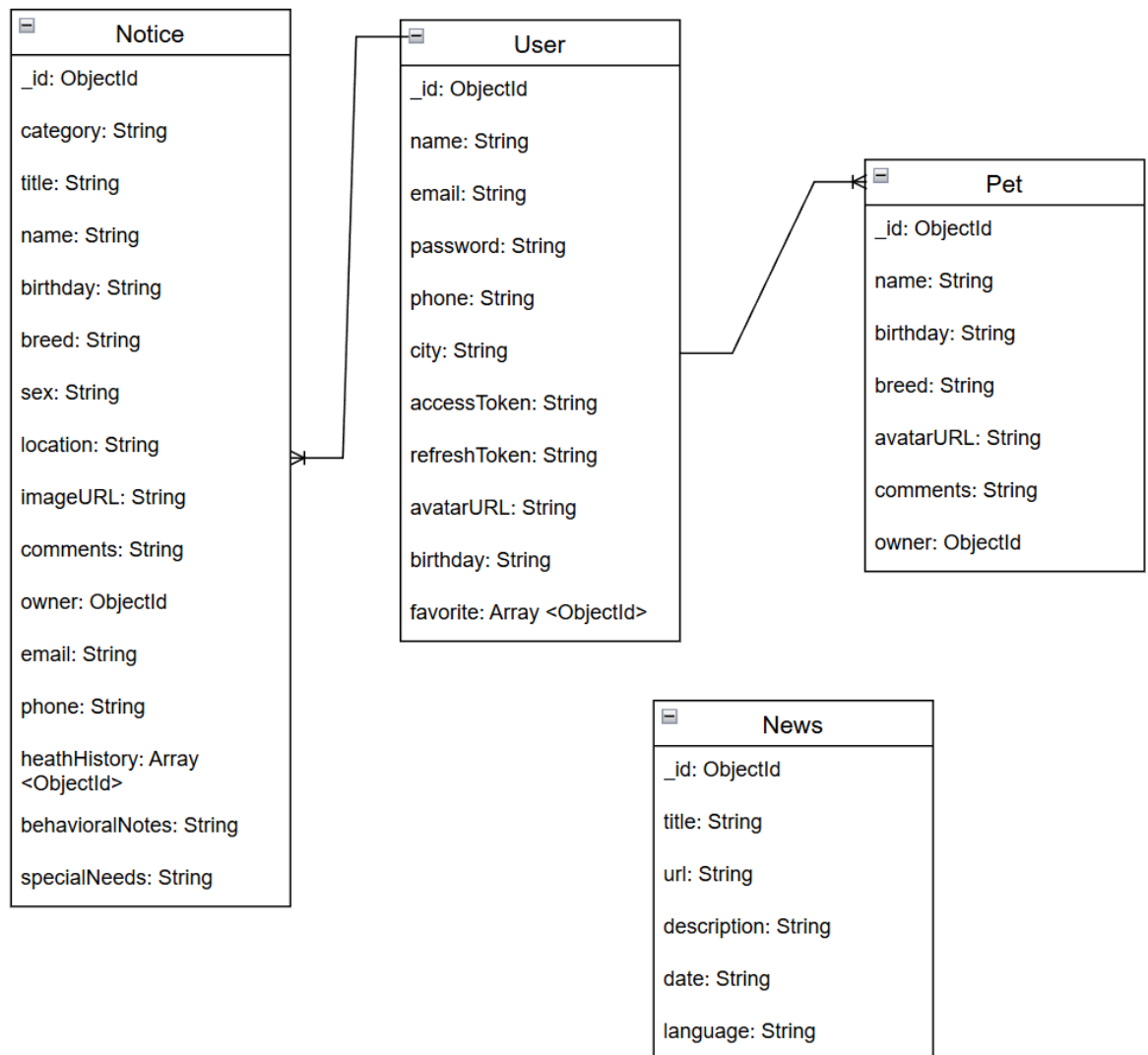


Рис. 2.2 - Структура бази даних

Ключовим компонентом бази є колекція користувачів, оскільки платформа орієнтована на взаємодію людей, які шукають або надають тварин

для адаптації. Структура моделі користувача містить у собі наступні дані: ім'я, електронну пошту, хешований пароль, номер телефону та місто проживання (табл.2.1).

Таблиця 2.1

Структура моделі користувача (User schema)

Поле	Тип	Призначення
name	String	Ім'я користувача. Обов'язкове поле, з перевіркою за регулярним виразом.
email	String	Електронна пошта. Обов'язкове унікальне поле, з валідацією за шаблоном.
password	String	Пароль користувача. Мінімум 6 символів, обов'язкове поле.
phone	String	Номер телефону у форматі +380XXXXXXXXXX. Обов'язкове поле.
city	String	Місто користувача. Обов'язкове поле з валідацією.
accessToken	String	Токен доступу. Генерується при авторизації, за замовчуванням порожній.
refreshToken	String	Токен оновлення. Генерується при авторизації, за замовчуванням порожній.
avatarURL	String	Посилання на аватар користувача. Має дефолтне зображення.
birthday	String	Дата народження у форматі DD.MM.YYYY. Необов'язкове поле.
favorite	Array of ObjectId	Список ID оголошень, доданих у «улюблені».

Модель користувача є однією з ключових структур бази даних, що відповідає за збереження персональної інформації про зареєстрованих користувачів платформи. Вона містить усі основні поля, необхідні для автентифікації, ідентифікації та персоналізації досвіду користувача.

Кожен запис у колекції користувачів включає обов'язкові поля, такі як ім'я, email, номер телефону, місто проживання та пароль. Email є унікальним і використовується як логін під час входу в систему. Пароль зберігається у зашифрованому вигляді й має мінімальну довжину для забезпечення базового рівня безпеки. Також передбачено можливість додавання дати народження,

аватару користувача та локації, які допомагають зробити профіль повноцінним.

Для підтримки безпечної авторизації реалізовано механізм токенів: accessToken і refreshToken. Вони дозволяють керувати сесіями користувачів, не вимагаючи повторного введення логіна та пароля.

Окремо в моделі реалізовано поле favorite, яке містить масив ідентифікаторів оголошень, доданих до улюблених. Це дозволяє користувачам зберігати тварин, що їх зацікавили, і швидко повертатись до них пізніше.

Загалом модель користувача забезпечує надійне та зручне зберігання інформації, а також підтримує функціонал, необхідний для персоналізованої роботи з платформою.

Модель Notice у системі відіграє ключову роль у представленні оголошень про тварин, що потребують адаптації. Вона охоплює повний набір полів, які дозволяють користувачам платформи створювати детальні картки тварин, включаючи інформацію про зовнішній вигляд, поведінку, стан здоров'я та контактні дані. Ця модель є основною для реалізації функціоналу публікації, пошуку тварин, а також слугує базою для подальшої інтеграції з іншими елементами системи, такими як користувачі, списки обраного та документи PDF (табл.2.2).

Таблиця 2.2

Структура модулі оголошення (Notice)

Поле	Тип	Призначення
category	String (enum)	Категорія оголошення: 'lost' або 'goodHands'
title	String	Заголовок оголошення
name	String	Ім'я тварини
birthday	String	Дата народження тварини у форматі dd.mm.yyyy
breed	String	Порода тварин
sex	String (enum)	Стать тварини: 'male' або 'female'
location	String	Місто, де знаходиться тварина
imageUrl	String	Посилання на зображення тварини
comments	String	Коментарі або додаткові зауваження

Продовження таблиці 2.2

owner	ObjectId	Посилання на користувача (власника оголошення)
email	String	Контактна електронна адреса власника
phone	String	Контактний номер телефону власника
healthHistory	Array<Object>	Масив з інформацією про щеплення тварини (назва вакцини, дата)
behavioralNotes	String	Опис поведінки тварини
specialNeeds	String	Особливі потреби тварини

Завдяки моделі Notice реалізується структуроване та зручне зберігання даних про тварин. Вона забезпечує гнучкість у наповненні оголошень, враховуючи обов'язкові та необов'язкові поля, включно з історією щеплень і особливими потребами тварини. Інтеграція з базою MongoDB через Mongoose дозволяє ефективно працювати з інформацією, гарантує цілісність даних, валідацію та масштабованість. Таким чином, модель Notice є фундаментальною частиною архітектури платформи, спрямованої на ефективну допомогу тваринам у пошуку нового дому.

Модель Pet використовується для збереження інформації про тварин, які належать до певного користувача. Вона слугує окремою сутністю для обліку домашніх тварин у системі, дозволяючи користувачам додавати своїх улюбленців, наприклад, для особистого кабінету. Кожен запис у цій моделі пов'язаний із користувачем (owner) та містить базову інформацію про тварину – ім'я, породу, дату народження, фото та текстовий опис або зауваження (табл.2.3).

Таблиця 2.3

Структура моделі Pet

Поле	Тип даних	Призначення
name	String	Ім'я тварини; обов'язкове поле, проходить перевірку на формат і довжину.
birthday	String (дата у форматі DD.MM.YYYY)	Дата народження тварини; обов'язкове поле.
breed	String	Порода тварини; обов'язкове поле.

Продовження таблиці 2.3

avatarURL	String	Посилання на зображення тварини; обов'язкове, але допускається порожнє.
comments	String	Коментар або опис про тварину; обов'язкове поле з валідацією.
owner	ObjectId (посилання на модель User)	Ідентифікатор користувача, який додав тварину; обов'язкове поле.

Модель Pet реалізована з урахуванням обов'язкових атрибутів, необхідних для повного профілю тварини, при цьому забезпечено гнучкість через можливість залишити фото порожнім. Поля моделі проходять валідацію за допомогою регулярних виразів, що гарантує коректність і чистоту введених даних. Зв'язок з моделлю User дозволяє точно відслідковувати, хто саме додав інформацію про тварину. Завдяки цьому модель є зручною для реалізації як особистих кабінетів, так і загального обліку тварин у системі. Вона гармонійно інтегрується з іншими частинами платформи, підтримуючи її модульність та розширюваність.

Модель News використовується для збереження та відображення актуальних новин, які мають інформативне значення для користувачів платформи. Вона є частиною контентного блоку сайту та дозволяє централізовано керувати інформаційними повідомленнями, що стосуються захисту тварин, діяльності притулків, оновлень платформи або соціальних ініціатив. Новини можуть бути представлені кількома мовами, що забезпечує багатомовну підтримку та охоплення ширшої аудиторії (табл.2.4).

Таблиця 2.4

Структура моделі News

Поле	Тип даних	Призначення
title	String	Заголовок новини; коротко описує зміст статті; обов'язкове поле.
url	String	Посилання на повну версію новини або джерело; обов'язкове поле.

description	String	Короткий опис або анонс новини; обов'язкове поле.
date	String	Дата публікації новини у форматі DD.MM.YYYY; обов'язкове поле.
language	String	Мова, якою написано новину (наприклад, uk, en); обов'язкове поле.

Модель News дозволяє ефективно управляти новинним контентом, зберігаючи найважливіші параметри кожної публікації – заголовок, опис, дату, мову та джерело. Завдяки чітко визначеним обов'язковим полям забезпечується цілісність і повнота інформації, що відображається користувачам. Підтримка різних мов розширює функціональність платформи, роблячи її зручною для іноземних відвідувачів. Модель легко інтегрується у вебінтерфейс і може слугувати основою для реалізації інформаційного розділу, push-сповіщень або email-розсилок.

У межах проектування платформи було розроблено логічно структуровану, масштабовану та зручну для підтримки базу даних, що відповідає функціональним потребам системи та вимогам до збереження інформації про тварин, користувачів, оголошення, особистих улюбленців та новини.

Основні моделі – User, Notice, Pet, News – чітко розмежовані за призначенням і відображають ключові аспекти роботи платформи. Вони охоплюють як базову контактну та ідентифікаційну інформацію, так і спеціалізовані поля: історія щеплень, особливості поведінки тварин, прикріплені зображення, геолокаційні та категорійні дані. Це дозволяє реалізувати повноцінний функціонал системи – від створення оголошень до генерації PDF-документів.

Використання нереляційної бази даних MongoDB у поєднанні з бібліотекою Mongoose забезпечило гнучкість у моделюванні документів, підтримку вкладених об'єктів та масивів (наприклад, історії вакцинацій), що є критично важливим для складної, динамічної структури з різними типами об'єктів. Архітектура бази даних

також враховує можливість масштабування в майбутньому, завдяки чому вона є придатною для подальшого розширення функціоналу.

Загалом, побудована структура бази даних гарантує цілісність, безпечне зберігання, зручність обробки та швидкий доступ до критично важливої інформації, що робить її надійною основою для всієї роботи платформи.

2.3. Реалізація API для взаємодії фронтенду та бекенду

Розробка програмного забезпечення є ключовим етапом створення платформи, що забезпечує функціональність для користувачів, управління оголошеннями про тварин, а також інтеграцію бази даних із серверною та клієнтською частинами.

У межах серверної частини платформи реалізовані кілька основних модулів: аутентифікація користувачів, управління оголошеннями, управління тваринами, робота з новинами та обліковими записами користувачів. Кожен модуль включає набір роутів, контролерів і моделей даних, що відповідають за певний функціонал.

Серверна частина проєкту реалізована на Express.js, що забезпечує обробку запитів, підключення до бази даних і управління ресурсами. Основними компонентами бекенду є модулі аутентифікації, що включають в себе реєстрацію, авторизацію та вихід користувачів. Роутинг API визначає правила доступу до ресурсів, таких як оголошення, новини, користувачі та тварини [22]. Контролери логіка яких обробки запитів, валідація даних і взаємодія з моделями. Моделі бази даних, що описують структури даних у MongoDB. Middleware для перевірки прав доступу, валідації запитів і завантаження файлів [23]. Хмарне зберігання включає використання Cloudinary для збереження зображень [24].

Для взаємодії між клієнтською частиною та сервером розроблено REST API, яке забезпечує операції CRUD (створення, читання, оновлення, видалення) для різних ресурсів. Нижче наведено опис основних ендпоінтів.

Реалізовані маршрути для роботи з користувачами: реєстрація, вхід, вихід та отримання інформації про поточного користувача. Використовується JWT-аутентифікація для захисту даних.

```
router.post("/register", validateBody(userSchemas.registerSchema), ctrlWrapper(ctrl.register));
router.post("/login", validateBody(userSchemas.loginSchema), ctrlWrapper(ctrl.login));
router.post("/logout", authenticate, ctrlWrapper(ctrl.logout));
router.get("/current", authenticate, ctrlWrapper(ctrl.getCurrentUser));
```

Кожен запит проходить через middleware `authenticate`, що перевіряє JWT-токен користувача.

Реалізовано маршрути для створення, отримання, видалення та додавання оголошень у список улюблених. Використовується middleware `authenticate` для перевірки автентичності запитів, а `isValidId` забезпечує перевірку коректності ідентифікаторів у базі даних.

```
router.post("/", authenticate, upload.single("avatar"), validateBody(noticesSchemas.noticeAddSchema),
ctrlWrapper(ctrl.addNotice));
router.patch("/toFavorite/:id", authenticate, isValidId, ctrlWrapper(ctrl.addNoticeToFavorite));
router.patch("/removeFavorite/:id", authenticate, isValidId, ctrlWrapper(ctrl.removeNoticeFromFavorite));

router.get("/users", authenticate, ctrlWrapper(ctrl.getUserNotices));
router.get("/users/favorite", authenticate, ctrlWrapper(ctrl.getFavoriteNotices));
router.get("/:id", isValidId, ctrlWrapper(ctrl.getOneNotice));
router.get("/", ctrlWrapper(ctrl.getNoticeByCategory));
router.delete("/:id", authenticate, isValidId, ctrlWrapper(ctrl.removeNotice));
```

Користувач може додавати та видаляти своїх улюбленців, а також отримувати інформацію про свій профіль. Дані про тварину зберігаються у вигляді об'єкта з характеристиками, такими як вік, ім'я, порода та зображення.

Система дозволяє отримувати актуальні новини, що стосуються тварин, догляду за ними та процесу адаптації. Дані отримуються із зовнішнього API за допомогою `NEWS_API_KEY` [25].

Контролери обробляють запити та взаємодіють із моделями бази даних через `Mongoose`. Наприклад, додавання нового оголошення відбувається наступним чином:

```
const addNotice = async (req, res) => {
  const { _id } = req.user;
```

```

const { path: avatarPath } = req.file;
const avatarURL = await uploadImageToCloudinary(avatarPath);

const newNotice = await Notice.create({ ...req.body, owner: _id, avatarURL });
res.status(201).json(newNotice);
};

```

Для взаємодії з базою даних використовується асинхронний виклик `create`, що додає новий запис.

Для безпеки додано JWT-аутентифікацію, що дозволяє користувачам отримувати доступ до приватних ресурсів після входу. Перевірку даних `middleware validateBody`, який гарантує, що дані відповідають заданій схемі. Обмеження прав доступу завдяки `middleware authenticate` перевіряє, чи має користувач доступ до певних дій.

Таким чином, серверна частина проєкту забезпечує злагоджену взаємодію між користувачем, базою даних та клієнтським інтерфейсом, гарантуючи безпеку та зручність використання платформи.

Фронтенд-платформа була розроблена на основі `React.js`, що дозволяє створювати динамічні та інтерактивні вебдодатки. Вона складається зі структурованої файлової системи, що включає основні сторінки, компоненти, маршрутизацію та управління станом за допомогою `Redux Toolkit`. Завдяки такому підходу забезпечується висока продуктивність, масштабованість і зручність використання. Наведено структуру сайту (Рис.2.2).



Рис. 2.3 - Структура сайту

Кожна зі сторінок містить відповідні React-компоненти, які відображають інтерфейс користувача та взаємодіють із серверною частиною за допомогою API-запитів.

Для організації переходів між сторінками використовується React Router, що дозволяє динамічно змінювати контент без перезавантаження сторінки. Навігація реалізована через компонент BrowserRouter, що включає основні маршрути та захищені маршрути для обмеження доступу до приватних сторінок [26].

Redux використовується для управління глобальним станом додатка, що забезпечує зручну взаємодію між компонентами та ефективну обробку асинхронних запитів. В даному проєкті зберігаються кілька основних категорій даних: автентифікація користувачів, інформація про користувача, оголошення, новини (табл.2.5).

Таблиця 2.5

Збережені дані в Redux та відповідні методи

Категорія даних	Збережені дані	Методи
Автентифікація користувача	JWT-токен: використовується для автентифікації користувачів після входу в систему	login, logout, current
	Дані про поточного користувача (ID, ім'я, email): містять основну інформацію про авторизованого користувача	login, current
	Статус авторизації: визначає, чи користувач залогінений	login, logout
Дані користувача	Аватар, персональні дані, список улюбленців: містять особисту інформацію користувача	getUserData, updateUserData
	Статус завантаження та редагування профілю: показує, чи виконується запит на зміну профілю	updateUserData
	Флаг додавання нового улюбленця: визначає, чи був успішно доданий новий улюбленець	addUserPet
Оголошення	Актуальні оголошення: список оголошень за вибраною категорією	getNoticeByCategory, getUserNotices
	Список улюблених оголошень: ID обраних користувачем оголошень	addNoticeToFavorite, removeNoticeFromFavorite
	Фільтри, пагінація, статус завантаження: визначають параметри пошуку та навігації	setQueryParams, setTotalPages

Новини	Масив отриманих статей: список новин, які отримав користувач	<code>getAllNews</code>
	Кількість сторінок: загальна кількість доступних сторінок з новинами	<code>setTotalPages</code>
	Фільтри та параметри запити: пошукові параметри новин	<code>setQueryParams</code>

Загалом Redux виконує важливу роль у додатку, забезпечуючи збереження критично важливої інформації, зменшуючи кількість запитів до сервера та покращуючи загальну продуктивність. Використання Redux Toolkit дозволяє зручно управляти станом, а `redux-persist` забезпечує збереження ключових параметрів, що сприяє кращій взаємодії користувача з платформою.

Асинхронна взаємодія між клієнтом і сервером здійснюється через `createAsyncThunk` [27], що дозволяє відправляти запити на бекенд і змінювати глобальний стан після їх виконання. Наприклад, у файлі `auth-operations.js` реалізовано основні методи, такі як `register()`, що надсилає дані користувача для створення нового акаунта. `Login()` перевіряє автентичність користувача та зберігає токен доступу. `Logout()` завершує сесію користувача та видаляє його дані з локального стану. `Current()` перевіряє поточний статус користувача на основі збереженого токена.

Кожен Redux-слайс взаємодіє з API через модуль `shared/api/`, що містить окремі файли для роботи з різними типами даних. API-запити реалізовані через `fetch` або `axios`, що дозволяє легко обробляти відповіді та помилки.

Візуальна частина розроблена за допомогою SCSS-модулів. Всі компоненти використовують модульний підхід до стилів, що дозволяє легко змінювати зовнішній вигляд окремих частин додатка без впливу на інші.

Також у клієнтській частині реалізована можливість завантаження офіційних PDF-документів про адаптацію тварин за допомогою інтеграції з

серверним API. Це розширює функціональність платформи та забезпечує зручність для користувачів у роботі з документами.

Ключові дані, такі як інформація про користувача, параметри пошуку та вибрані оголошення, зберігаються за допомогою бібліотеки `redux-persist`. Це дозволяє уникнути втрати важливої інформації при оновленні сторінки або перезапуску браузера. Наведено діаграму послідовностей, яка загальноно показує взаємодію користувача з вебсайтом для перегляду оголошень, створення нових публікацій та здійснення донатів. Логіка є такою, що користувач шукає, переглядає оголошення та створює нові. Платформа надсилає відповідні запити на сервер для обробки інформації. Також користувач може зареєструватися/увійти в особистий кабінет і також зробити пожертву. Донат обробляється через бекенд і якщо виникне помилка то вона передається назад користувачу (Рис.2.3).

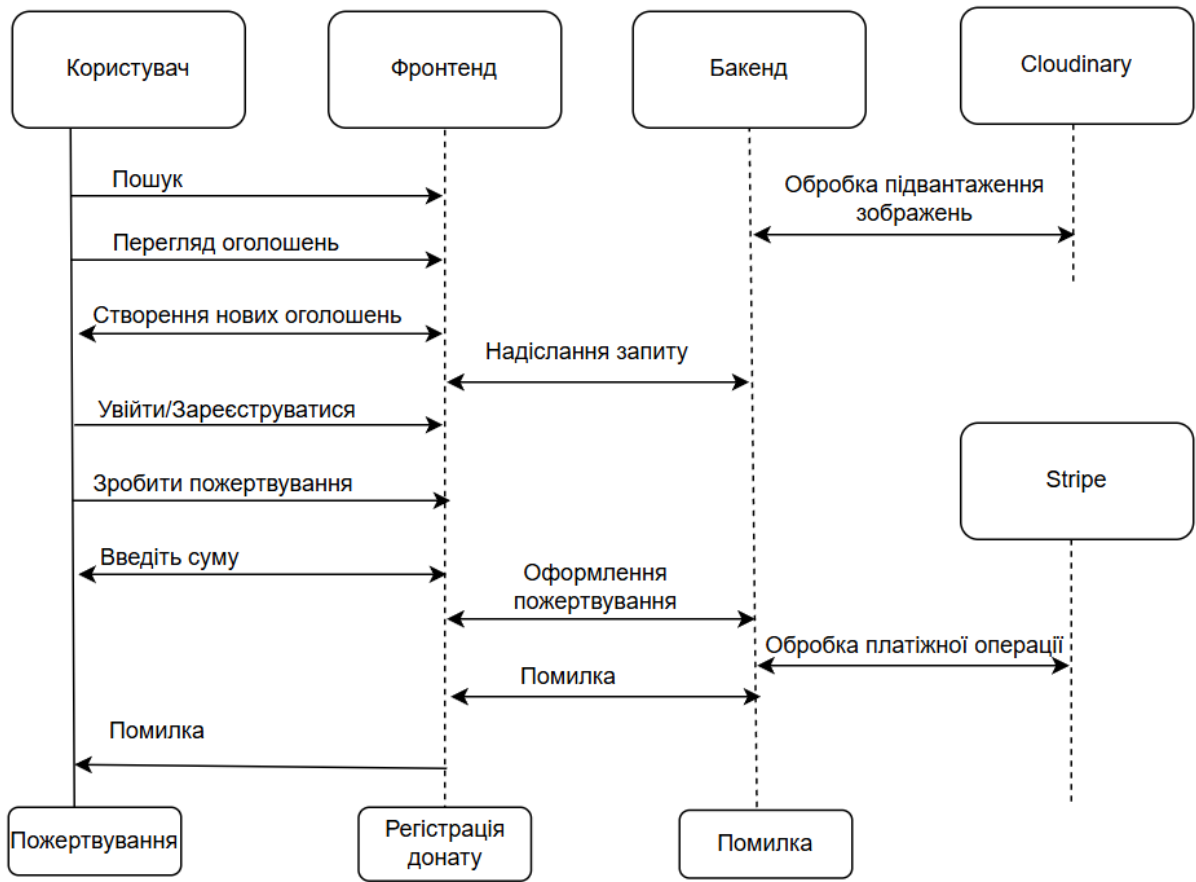


Рис.2.4 - Діаграма послідовностей

Таким чином, створена інтеграція між клієнтською та серверною частинами платформи забезпечує повний цикл обробки даних, що відповідає сучасним вимогам до розробки вебдодатків.

Фронтенд - додаток розроблений із використанням сучасних технологій та забезпечує зручний і продуктивний інтерфейс для користувачів. Використання React.js, Redux Toolkit та React Router забезпечує ефективне керування станом, навігацію та інтерактивність. Завдяки модульному підходу та використанню SCSS, проєкт є легко підтримуваним та розширюваним.

Функціонал вебплатформи для адаптації та пошуку тварин орієнтований на простоту користування, ефективну взаємодію з базою даних та підтримку ключових сценаріїв: створення оголошень, перегляд карток тварин, фільтрація, генерація документів та здійснення благодійних внесків.

Реалізацію ініціалізації серверного застосунку та підключення до бази даних наведено у Додатку А.

Користувачі мають можливість створювати облікові записи, проходити авторизацію та керувати своїм профілем. Для безпечного доступу до захищених розділів системи реалізовано автентифікацію через JWT-токени.

Після входу користувач отримує доступ до персонального кабінету, в якому можна редагувати особисту інформацію, переглядати список улюблених тварин та додавати нові оголошення. Додавання оголошення реалізовано через багатоступеневу форму (AddPetForm.jsx), що містить поля для категорії, опису, дати народження тварини, фото, місця перебування, а також медичної інформації (історії щеплень, особливих потреб). Логіку асинхронних запитів до оголошень у Redux наведено у Додатку Д.1. Селектори для вибірки стану оголошень у Redux наведено у Додатку Д.2. Визначення Redux slice для оголошень із початковим станом та редюсерами наведено у Додатку Д.3.

Перед відправкою дані обробляються функцією createFormData, яка трансформує об'єкт у формат FormData, включаючи вкладені масиви. Цей

підхід дозволяє передавати файли зображень та серіалізовану історію вакцинації в одному запиті.

Оголошення передаються на бекенд через маршрут: `POST /api/notices`

Файл зображення за потреби завантажується в Cloudinary (Додаток Е), а медична інформація з поля `healthHistory` парситься на сервері.

Для кожної тварини передбачено можливість перегляду картки з детальною інформацією. В ній відображаються: порода, вік, дата народження, місце перебування, контактні дані, поведінкові особливості, список щеплень. Додатково користувач може додати тварину до улюблених або завантажити PDF-документ про всиновлення.

Генерація PDF реалізована за допомогою бібліотеки PDFKit. Створення документа відбувається на сервері динамічно при зверненні до маршруту:

`GET /api/notices/generate-document/:id`

Документ містить основну інформацію про тварину, історію вакцинації та поля для підпису. Повна реалізація наведена у Додатку Б.

Для підтримки фінансової допомоги впроваджено Stripe Checkout. Після натискання на кнопку “Зробити донат” на фронтенді надсилається запит до:

`POST /api/stripe/session`

Де створюється сесія Stripe, яка редиректить користувача на сторінку оплати. Реалізацію цього функціоналу наведено у Додатку В.

```
const session = await stripe.checkout.sessions.create({
  payment_method_types: ['card'],
  line_items: [...],
  success_url: `${domain}/donate?status=success`,
});
```

Також доступна вкладка новин, де публікуються актуальні статті, поради щодо догляду за тваринами, анонси подій тощо.

Окремий модуль новин реалізовано для інформування користувачів про актуальні події, пов’язані з тваринами, захистом прав тварин, новинами з притулків, порадами щодо догляду та іншими тематичними матеріалами.

Новини доступні двома мовами – українською та англійською, що забезпечує багатомовну підтримку платформи.

Інформація надходить з двох джерел: англomовного і українomовного. Англomовні новини отримуються через інтеграцію з NewsAPI. Для кожного запиту здійснюється фільтрація статей за ключовими словами `pets` OR `animals`, встановлюється обмеження до 50 новин, і результати зберігаються в базу даних, якщо URL ще не існує. Дані включають заголовок, опис, дату публікації, посилання та мову новини.

Україномовні новини парсяться з RSS-стрічки сайту [TSN.ua](https://tsn.ua) за допомогою бібліотеки `rss-parser`. Кожен елемент, який містить унікальне посилання, зберігається в базу новин (колекція `news`) з відповідною мовною позначкою `"uk"`.

Код реалізації українського парсингу:

```
const feed = await parser.parseURL("https://tsn.ua/rss");
const items = feed.items.slice(0, 50);
await News.create({ title, url, description, date, language: "uk" });
```

Для користувача реалізовано ендпоінт `/api/news` із підтримкою наступних параметрів: `page` (поточна сторінка пагінації); `limit` (кількість новин на сторінці); `search` (ключове слово для пошуку в заголовку або описі); `lang` (мова новин `"uk"` або `"en"`).

Пошук здійснюється за допомогою `$regex` з опцією нечутливості до регістру (`$options: "i"`), що дозволяє шукати як у назвах, так і в описах новин. Пагінація реалізована динамічно на сервері з урахуванням загальної кількості новин та параметрів користувача. Автоматичне оновлення новин – при кожному запиті новини синхронізуються з джерелами, і нові елементи додаються в базу тільки якщо їх ще не існує.

Модуль новин платформи є інформативним і динамічним, дозволяє користувачам бути в курсі важливих подій, а також підвищує загальну залученість аудиторії. Можливість пошуку та фільтрації новин за мовою й ключовими словами робить інтерфейс зручним і ефективним.

Таким чином, основний функціонал платформи орієнтований на всебічну підтримку процесу всиновлення тварин: від створення оголошень і пошуку нових власників до зручного юридичного оформлення документів. Завдяки сучасному підходу до розробки, платформа є надійним інструментом для ефективної взаємодії між людьми, які прагнуть допомогти тваринам знайти новий дім.

Висновки до другого розділу

У результаті реалізації програмного забезпечення було побудовано повноцінну вебплатформу, яка забезпечує зручну взаємодію між користувачами, притулками та волонтерськими організаціями з метою адаптації тварин.

Було створено багаторівневу архітектуру платформи, яка включає: особистий кабінет користувача; сторінку з оголошеннями тварин; модуль додавання картки тварини з фото, описом та медичною історією; систему автентифікації з JWT; багатомовну підтримку (українська/англійська); адаптивний дизайн інтерфейсу.

Ключовим функціональним досягненням стало впровадження Stripe API, що забезпечило можливість прийому благодійних внесків для підтримки діяльності платформи.

У межах другого розділу також: реалізовано REST API для взаємодії між фронтендом і бекендом; розроблено базу даних із гнучкою схемою, що включає дані про тварин, користувачів, історію вакцинацій, поведінкові особливості; реалізовано можливість формування PDF-документів для усиновлення тварин.

Завдяки адаптивній верстці та структурованому підходу до розробки, платформа забезпечує високу продуктивність, масштабованість і безпеку. Таким чином, виконані роботи повністю відповідають технічному завданню й забезпечують ефективну цифрову підтримку процесу адопції тварин в Україні.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА КОРИСТУВАННЯ ПРОГРАМНИМ ПРОДУКТОМ

3.1 Користувацький інтерфейс і адаптивність верстки

Користувацький інтерфейс (UI) платформи для адаптації та пошуку тварин розроблений із урахуванням принципів доступності, простоти взаємодії, візуальної привабливості та ефективної адаптації під різні типи пристроїв. Основна мета інтерфейсу – забезпечити максимально комфортний шлях користувача до взаємодії з оголошеннями, пошуком тварин, публікацією нової інформації, переглядом новин та підтримкою проєкту через донати.

Верстка платформи побудована за принципами mobile-first, що означає пріоритетну розробку під мобільні пристрої з поступовим масштабуванням для планшетів і десктопів. Застосовано Flexbox [28] та CSS Grid [29] для формування адаптивних макетів.

Медіа-запити (@media) використовуються для зміни стилів відповідно до ширини вікна браузера. Наприклад, на мобільних пристроях картки оголошень розташовуються вертикально в одну колонку, а на великих екранах – у триколонну сітку.

Візуальне представлення головної сторінки платформи є важливим елементом користувацького інтерфейсу, оскільки саме з неї починається перша взаємодія користувача із застосунком. Адаптивність макету забезпечує коректне та зручне відображення сторінки незалежно від розміру екрана, що особливо важливо для користувачів смартфонів і планшетів (рис.3.1, 3.2).

Адаптація охоплює:

- перемикання бургер-меню замість горизонтальної навігації;
- збільшені кнопки та поля для зручності натискання;
- автоматичне масштабування зображень;
- вирівнювання тексту та інтерфейсних елементів для зручного читання.

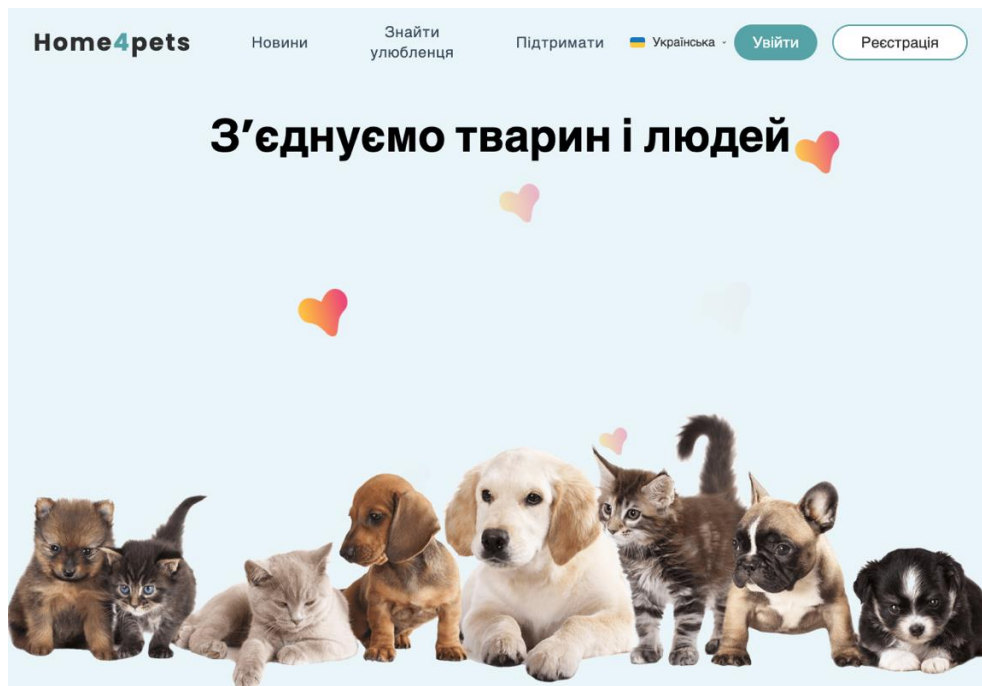


Рис. 3.1 - Головна сторінка платформи на десктопному пристрої

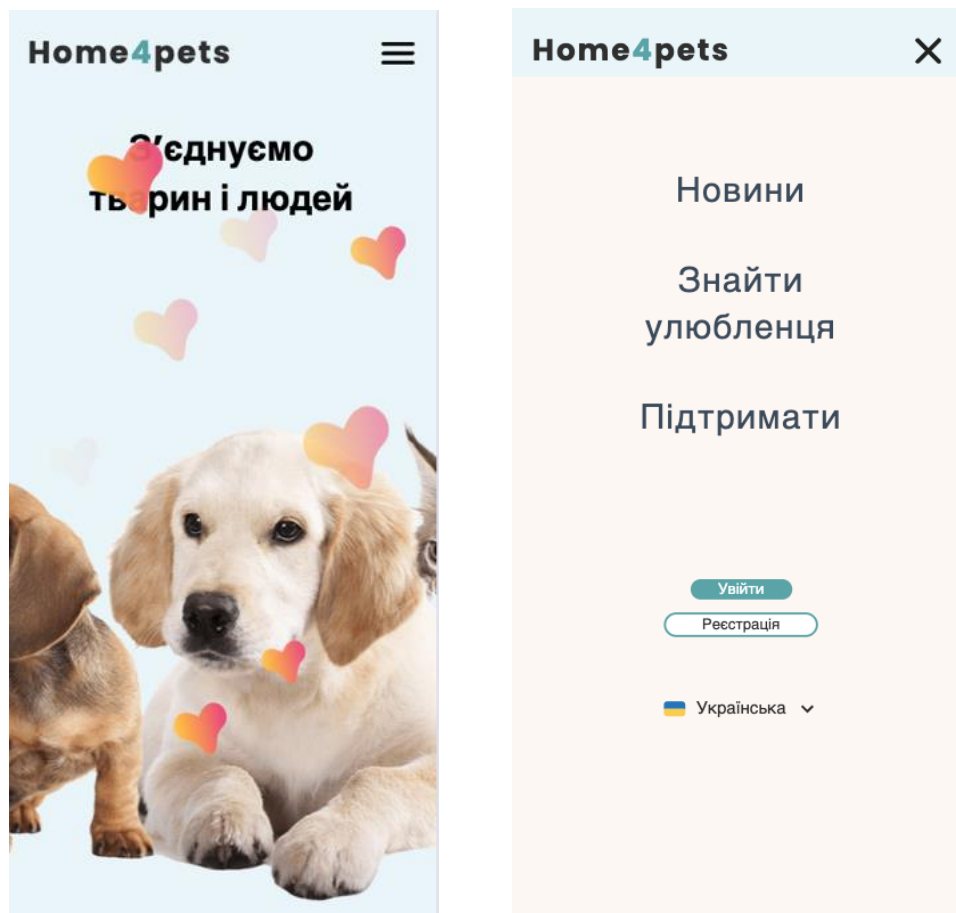


Рис. 3.2 - Головна сторінка платформи на мобільних пристроях

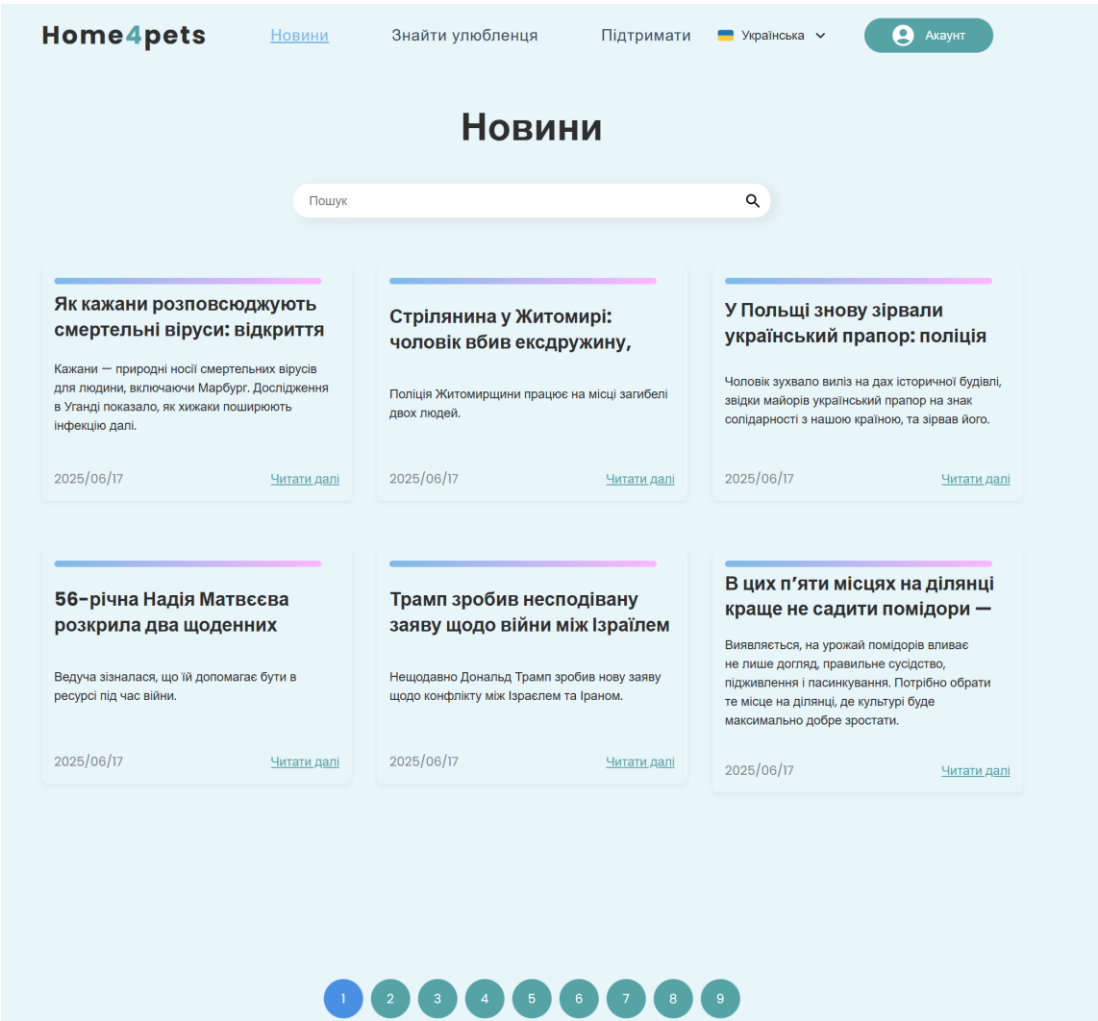


Рис.3.4 - Сторінка новин українською мовою

Реалізація новинного модуля у двох мовних варіантах демонструє високий рівень адаптивності платформи до потреб користувачів з різною мовною орієнтацією. Завдяки динамічному завантаженню контенту з перевірених джерел та можливості пошуку за ключовими словами, сторінка новин виконує не лише інформаційну, але й освітню функцію. Інтеграція з зовнішніми джерелами дозволяє системі бути незалежною від ручного наповнення контенту, що оптимізує витрати ресурсів і підвищує ефективність її функціонування.

Залежно від мови, запити надсилаються на відповідне джерело, що реалізовано у функції `getNews`, з подальшою фільтрацією, пагінацією та виведенням даних на інтерфейс. Такий підхід забезпечує релевантність, масштабованість і актуальність представленої інформації на сторінці новин.

Крім базових функцій, інтерфейс надає:

Генерацію PDF-документа про всиновлення тварини (інтегровано в картку тварини) (Рис.3.5).

Pet Adoption Agreement

Pet Information

Name: Рудик
Breed: Дворняга
Birthday: 01.01.2023
Location: Київ

Health History

1. Vaccine: Rabies, Adoption Date: 5/28/2025

Behavioral Notes

Дуже ніжний та ручний, граціозний, виважений, акуратний та досить ледачий. До вулиці ніякої цікавості не має. В туалет ходить виключно на лоток. Гарно співіснує з іншими котами, та навіть собаками.

New Owner Information

Name: Дар'я
Email: dgamarskayaa2@gmail.com
Phone: +380985771378

Adoption Date: 6/17/2025

Signature of New Owner

Рис.3.5 - Згенерований договір всиновлення тварини

Функцію “Додати до улюблених” через інтерактивну кнопку з іконкою (Рис.3.6).

100 грн, 200 грн та 500 грн. Кожна кнопка викликає Stripe Checkout-сесію з відповідною сумою (Рис.3.8).

← New business sandbox Sandbox

Donation for Animal Support
UAH 200.00

Email
email@example.com

Card information
1234 1234 1234 1234 VISA MASTERCARD AMERICAN EXPRESS DISCOVER
MM / YY CVC

Cardholder name
Full name on card

Country or region
Ukraine

☐ Securely save my information for 1-click checkout
Pay faster on New business sandbox and everywhere Link is accepted.

Pay

Powered by stripe
Terms Privacy

Рис. 3.8 - Інтерфейс Stripe Checkout для здійснення благодійного донату на платформі Home4Pets

Функціонал Stripe-донатів є ефективним і безпечним інструментом підтримки ініціативи, що значно підвищує залученість користувачів. А простий і зрозумілий інтерфейс сприяє швидкому прийняттю рішення на користь підтримки. Таким чином, сторінка "Support" виконує як функціональну, так і мотиваційну роль.

Отже, користувацький інтерфейс платформи розроблено з акцентом на зручність, інтуїтивну взаємодію та доступність із будь-яких пристроїв. Використання сучасного стеку технологій (React, Redux, SCSS, i18n) дозволило створити інтерфейс, який задовольняє потреби користувачів як із технічної, так і з емоційної точки зору. Завдяки адаптивній верстці та продуманому UX, платформа забезпечує приємний досвід навіть для нових користувачів і залишається стабільною при розширенні функціоналу в майбутньому.

Однією з ключових складових сучасної вебплатформи, орієнтованої на взаємодію з користувачем, є система автентифікації та особистий кабінет. У рамках проєкту «Home4Pets» реалізовано повноцінну підтримку користувацьких облікових записів, що дозволяє зберігати індивідуальні дані, управляти власними оголошеннями та взаємодіяти з іншими функціями сайту. Це сприяє персоналізації досвіду користувача та підвищує рівень безпеки, контроль доступу до функціоналу та ефективність використання платформи.

Сторінка реєстрації RegisterPage містить обов'язкові поля: ім'я, електронну пошту, пароль, номер телефону та місто. На стороні клієнта валідація реалізована за допомогою бібліотеки react-hook-form у поєднанні з кастомним валідаційним шаблоном (регулярні вирази). На бекенді всі дані додатково перевіряються через Joi-схему (registerSchema).

Кнопка “Next” активується лише після проходження базової валідації на фронтенді. У разі некоректного або неповного заповнення обов'язкових полів система видає підказки (error-меседжі) червоним кольором, інформуючи користувача про помилки. Це дозволяє швидко виправити введені дані (Рис.3.9).

Рис. 3.9 - Форма реєстрації користувача (етап 1)

Після успішного введення email і паролю користувач переходить до заповнення другої частини форми, яка включає ім'я, місто та номер телефону. Реалізована перевірка на обов'язковість заповнення (Рис.3.10).

Home4pets

Новини

Знайти
улюбленця

Підтримати

 Українська ▾

Увійти

Реєстрація

Створити акаунт

[Вже маєте акаунт? Увійдіть](#)

Рис. 3.10 - Форма завершення реєстрації (етап 2)

Після реєстрації та авторизації користувач потрапляє в персональний кабінет. Тут відображається його фото, контактні дані, місто, дата народження. Також користувач може додавати інформацію про власних тварин або перейти до редагування існуючих даних (Рис.3.11).

The screenshot shows the Home4pets website interface. At the top, there is a navigation bar with the logo 'Home4pets' on the left, and links for 'Новини', 'Знайти улюбленця', 'Підтримати', and a language selector set to 'Українська'. A 'Акурит' button with a user icon is on the right. Below the navigation bar, the main content area has a light blue background. On the left, a white card displays the user's profile information: a placeholder for a profile picture, the name 'Ім'я: Дар'я', email 'Електронна Пошта: dgamarskayaa@gmail.com', date 'Дата Народження: Невідомо', phone 'Телефон: +380965881496', and location 'Місто: Бердів'. Each field has a small blue pencil icon for editing. At the bottom of the card is a 'Вийти' button. To the right of the profile card, a large white card contains the text 'У вас ще немає улюбленців' and 'Додати улюбленця:' above a large teal circle with a white plus sign.

Рис. 3.11 - Головна секція особистого кабінету

У режимі редагування користувач може змінити особисту інформацію (наприклад, дату народження). Поле автоматично набуває активного стану, і після внесення змін необхідно натиснути галочку для підтвердження. Інтерфейс реалізовано інтуїтивно, з іконками редагування біля кожного поля (Рис.3.12).

Рис. 3.12 - Режим редагування профілю

Форма авторизації дозволяє зареєстрованим користувачам увійти до системи, ввівши email і пароль. Передбачено можливість візуалізації пароля через відповідну іконку. У разі неправильного введення або відсутності облікового запису користувач може перейти за посиланням на реєстрацію. Інтерфейс адаптовано до мобільних пристроїв, забезпечено валідацію введених даних. (Рис.3.13).

Рис. 3.13 - Сторінка авторизації користувача

Після успішної авторизації користувач переходить до свого профілю, де зображено його аватар, ім'я, email, дату народження, номер телефону та місто. Поруч розміщено секцію для додавання інформації про власних тварин. Фото профілю можна змінити, натиснувши на іконку «Edit Photo» (Рис.3.14).

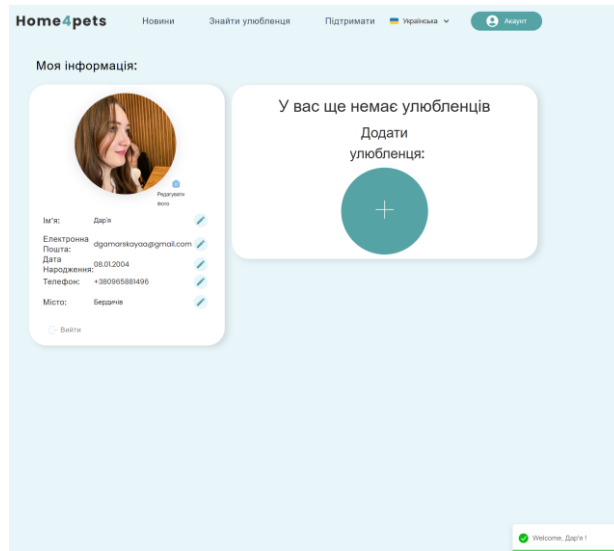


Рис. 3.14 - Повідомлення про успішну авторизацію

У нижній частині екрана з'являється toast-сповіщення з підтвердженням успішного входу до системи: «Welcome, [UserName]». Це сприяє позитивному досвіду користувача та підтверджує коректність виконаної операції.

На рисунку показано приклад збереженого запису користувача у базі даних MongoDB. Об'єкт містить унікальний ідентифікатор `_id`, основну інформацію про користувача (ім'я, email, пароль у хешованому вигляді, номер телефону, місто проживання), а також поля для `accessToken`, `refreshToken`, посилання на аватар (`avatarURL`) і дату народження. Також присутній масив `favorite`, що відповідає за список обраних оголошень. Метадані `createdAt` і `updatedAt` автоматично зберігають дати створення та останнього оновлення документа. Ця структура є типовою для моделі `User`, створеної за допомогою бібліотеки `Mongoose` у `Node.js` (Рис.3.15).

```

_id: ObjectId('68519f58a3f13fbbe5b6d2f7')
name : "Дар'я"
email : "dgamarskayaa@gmail.com"
password : "$2a$10$fk7faHyokD7ipb.LgwWeJ060ECZdsh5BHYfC2IuQdJfDYkwe17F9G"
phone : "+380965881496"
city : "Бердичів"
accessToken : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY4NTE5ZjU4YTNmMTNmYmJlN..."
refreshToken : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY4NTE5ZjU4YTNmMTNmYmJlN..."
avatarURL : "https://res.cloudinary.com/djcuhdps6/image/upload/v1750179894/ysbi6ors..."
birthday : "08.01.2004"
favorite : Array (empty)
createdAt : 2025-06-17T17:01:12.462+00:00
updatedAt : 2025-06-17T17:06:15.105+00:00

```

Рис. 3.15 - Структура документа користувача в базі даних MongoDB

Отже, реалізація особистого кабінету, а також функціоналу реєстрації та авторизації користувачів у платформі Home4Pets дозволила забезпечити індивідуалізований досвід взаємодії з сервісом. Завдяки використанню технологій JWT для автентифікації та зберіганню інформації в MongoDB через Mongoose, система гарантує надійний захист персональних даних, а також гнучкість і масштабованість зберігання користувацьких профілів.

Однією з ключових можливостей платформи Home4Pets є функціонал, що дозволяє користувачам самостійно створювати оголошення про тварин, а також зберігати вподобані профілі у "список улюбленців". Це дозволяє ефективно організувати процес пошуку нових господарів для тварин, зберігаючи релевантну інформацію та забезпечуючи зручний доступ до неї для авторизованих користувачів.

Користувачі платформи мають змогу не лише переглядати оголошення, а й створювати особисті картки своїх домашніх улюбленців у межах особистого кабінету. Така функція дозволяє зручно зберігати інформацію про власних тварин і, за потреби, швидко створювати оголошення про їх пошук або передачу в добрі руки.

Процес додавання улюбленця відбувається поетапно у вигляді модального вікна. На першому кроці користувач заповнює базові дані: ім'я тварини, дату народження та породу. Після переходу до наступного етапу система запитує завантаження фотографії та внесення коментарів про особливості улюбленця — характер, звички, тощо. Усі дані зберігаються у

базі, а додана тварина одразу з’являється у секції “My Pets” персонального кабінету користувача (Рис.3.16, Рис.3.17, Рис. 3.18).

Додати улюбленця

Кличка

Багіра

Дата народження

01.01.2024

Порода

Бомбей

Скасувати

Далі

Рис. 3.16 - Додавання улюбленця: крок 1: ім’я, дата народження, порода

Додати улюбленця

Додайте фото та коментар



Дуже грайлива та мила кішечка

Назад

Готово

Рис. 3.17 - Додавання улюбленця крок 2: фото та опис тварини

					ІПЗ.КР.Б. – 121 – 25 – ПЗ	55
		№ докум.	Підпис			

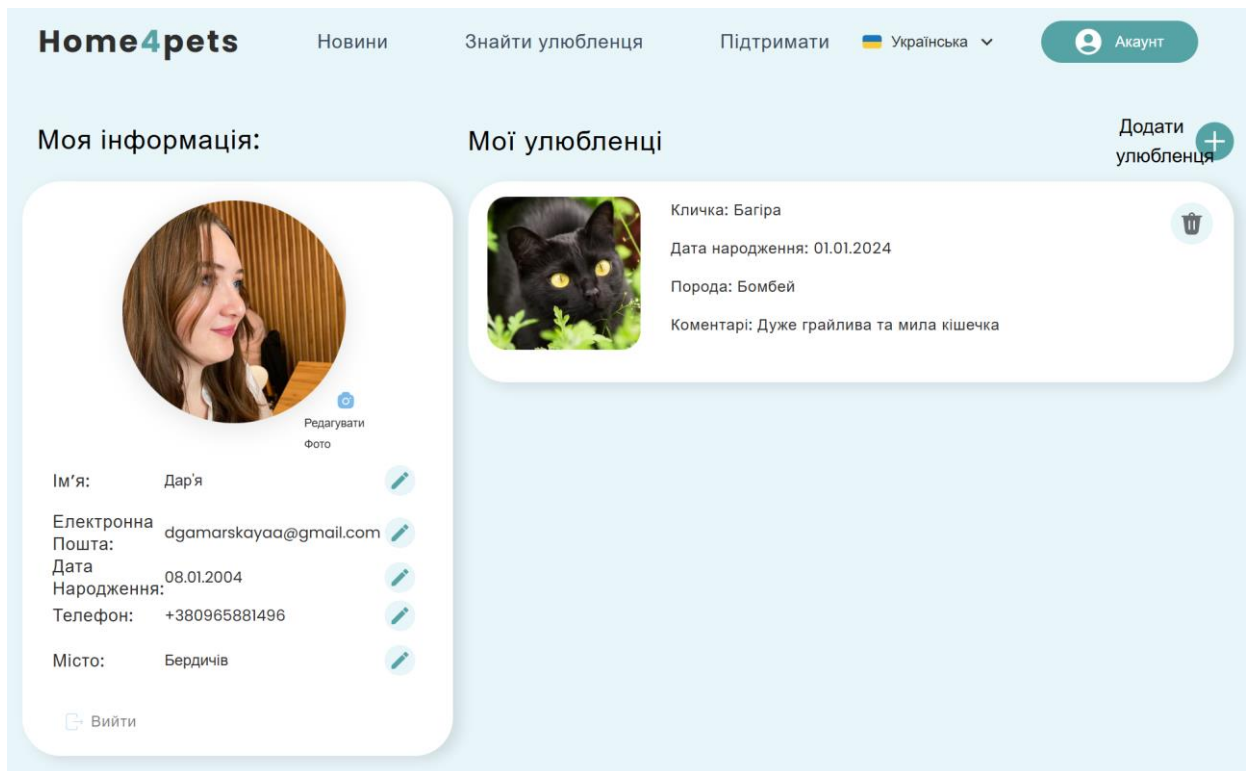


Рис. 3.18 - Сторінка особистого кабінету з доданим улюбленцем

Завдяки такій функціональності платформа дозволяє користувачам вести облік своїх тварин, зберігати важливу інформацію про них і забезпечує основу для подальшої взаємодії (наприклад, створення оголошення або збереження історії профілю). Інтерфейс інтуїтивний і реалізований у стилі крок за кроком, що робить процес додавання максимально зручним навіть для новачків.

Опис процесу додавання нового оголошення до платформи складається з трьох ключових кроків, кожен із яких реалізований у вигляді окремого етапу введення даних із валідацією та зручним UI-інтерфейсом. Користувач поетапно вводить інформацію, яка відображатиметься в публічному оголошенні про тварину.

На першому кроці користувач вибирає категорію оголошення: «Загублені / Знайдені» або «В добрі руки», після чого заповнює поля, що містять заголовок оголошення, кличку тварини, дату народження та породу. Це дозволяє коротко описати тварину та її основні характеристики, що буде видно іншим користувачам при перегляді картки (Рис.3.19).

Додати оголошення

Загублені / Знайдені
Віддам безкоштовно

Заголовок оголошення*
Лагідний кіт шукає родину

Кличка
Фог

Дата народження
05.02.2022

Порода
хайленд-страйт

Скасувати
Далі

Рис. 3.19 - Заповнення основної інформації для оголошення (заголовок, кличка, дата народження, порода)

Другий етап включає визначення статі тварини, вибір міста, завантаження зображення та заповнення блоку «Історія щеплень». Це важливо з точки зору інформування потенційного власника про стан здоров'я тварини. Для кожного щеплення зазначається назва та дата (Рис.3.20).

Додати оголошення

Стать*:

Самець
Самка

Місто*
Житомир

Завантажте зображення улюбленця:

Історія щеплень:

Rabies
28.05.2025

FeLV
04.06.2025

+ Додати щеплення

Скасувати
Далі

Рис. 3.20 - Введення інформації про стать, місто, завантаження фото та історію щеплень тварини

На фінальному кроці користувач заповнює текстові поля щодо поведінкових особливостей тварини (наприклад, спокійна, лагідна, грайлива), вимог до догляду (особлива дієта, алергії, потреба в ліках), а також загальний коментар до оголошення – наприклад, рекомендації для нових власників. Така структура дозволяє зібрати комплексну та корисну інформацію про улюбленця (Рис.3.21).

Додати оголошення

Особливості поведінки

Неймовірно ласкавий та люблячий. Любить щоб його гладили

Особливості догляду

Має алергію на курятину

Коментар до оголошення

Віддамо лише у відповідальні руки

НазадГотово

Рис. 3.21 - Додавання інформації про поведінку, догляд і коментарі до оголошення

Форма додавання оголошення розроблена з урахуванням принципів зручності, доступності та повноти даних. Вона забезпечує послідовне введення інформації, валідацію, а також можливість додати фото, вакцинальні дані та особливості поведінки. Це підвищує якість поданих оголошень і збільшує ймовірність успішної адаптації тварини.

На рисунку зображено сторінку з оголошенням, створеним користувачем. В інтерфейсі передбачено фільтрацію за категоріями («Віддам безкоштовно», «Загублені/знайдені», «Улюблені», «Мої оголошення») та рядок пошуку. Візуально представлено картку тварини з фото, короткою інформацією про породу, місце знаходження та вік. Доступні дії: переглянути більше деталей

або видалити оголошення. Також передбачена кнопка «Додати», що відкриває форму створення нового оголошення (Рис.3.22).

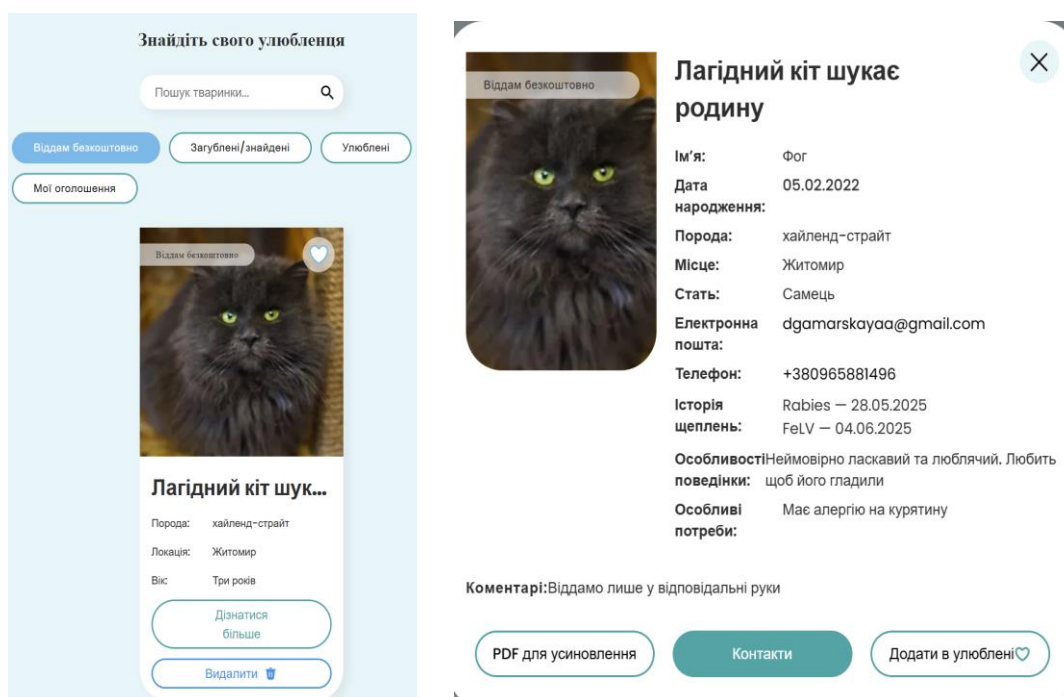


Рис. 3.22 - Перегляд оголошень користувача

Рисинок демонструє, як зберігається інформація про користувача в базі даних MongoDB. Дані включають: ім'я, email, хешований пароль, номер телефону, місто, токени для авторизації (accessToken, refreshToken), URL аватарки, дату народження, масив улюблених тварин (favorite), дату створення та оновлення профілю. Структура зберігання забезпечує безпечну роботу з персональними даними та дозволяє ефективно керувати профілем користувача (Рис.3.23).

```

_id: ObjectId('68519f58a3f13fbbe5b6d2f7')
name: "Дар'я"
email: "dgamarskayaa@gmail.com"
password: "$2a$10$fk7faHyokD7ipb.LgwWeJ060ECZdsh5BHYfC2IuQdJfDYkwe17F9G"
phone: "+380965881496"
city: "Бердичів"
accessToken: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY4NTE5ZjU4YTNmMTNmYmJlLn..."
refreshToken: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY4NTE5ZjU4YTNmMTNmYmJlLn..."
avatarURL: "https://res.cloudinary.com/djcuhdps6/image/upload/v1750179894/ysbi6ors..."
birthday: "08.01.2004"
favorite: Array (1)
  0: ObjectId('68519743a3f13fbbe5b6d159')
createdAt: 2025-06-17T17:01:12.462+00:00
updatedAt: 2025-06-17T17:57:29.647+00:00

```

Рис. 3.23 – Дані користувача в базі даних MongoDB

Отже, у результаті реалізації функціоналу додавання улюбленців та нових оголошень було створено зручний, інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам легко передавати інформацію про тварин у систему. Через покрокові форми користувач має змогу ввести всі необхідні дані – від базової інформації (ім'я, порода, вік, місто) до специфічних характеристик поведінки, потреб у догляді, історії щеплень і фотографій.

3.2 Система розгортання застосунку

Система розгортання забезпечує публікацію серверної та клієнтської частин платформи.

Для розгортання клієнтської частини виконано аналіз хостингів для фронтенду. Розглянемо порівняння таких хостингів, як Vercel, Netlify, Firebase Hosting, GitHub Pages (табл. 3.1)

Таблиця 3.1

Сервіс	Переваги	Недоліки
Vercel	Швидке розгортання Оптимізовано під React Безкоштовний тариф	Обмежена настройка бекенду
Netlify	Просте розгортання Вбудовані форми	Повільніший build у безкоштовному тарифі
Firebase Hosting	Інтеграція з іншими сервісами Firebase	Необхідність налаштувань через CLI
GitHub Pages	Безкоштовний Пряме розгортання з GitHub	Лише статичні сайти Немає бекенду Менше налаштувань та функцій

Після аналізу було вирішено вибрати для розгортання клієнтської частини Vercel, адже він безкоштовний, має швидке розгортання та гарно оптимізований під React. На рисунку наведено задеплоїну клієнтську частину на Vercel (Рис. 3.24).

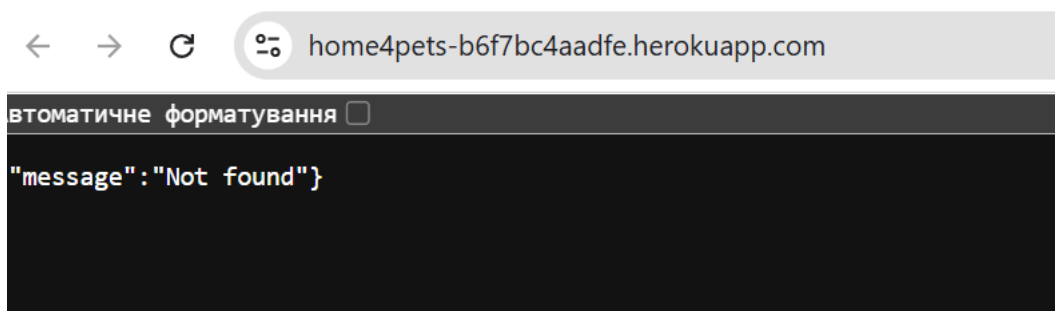


Рис. 3.25 - Задеплойна Серверна частина на Vercel

Після розгортання серверної та клієнської частини маємо задеплойну вебплатформу (Рис. 3.26)

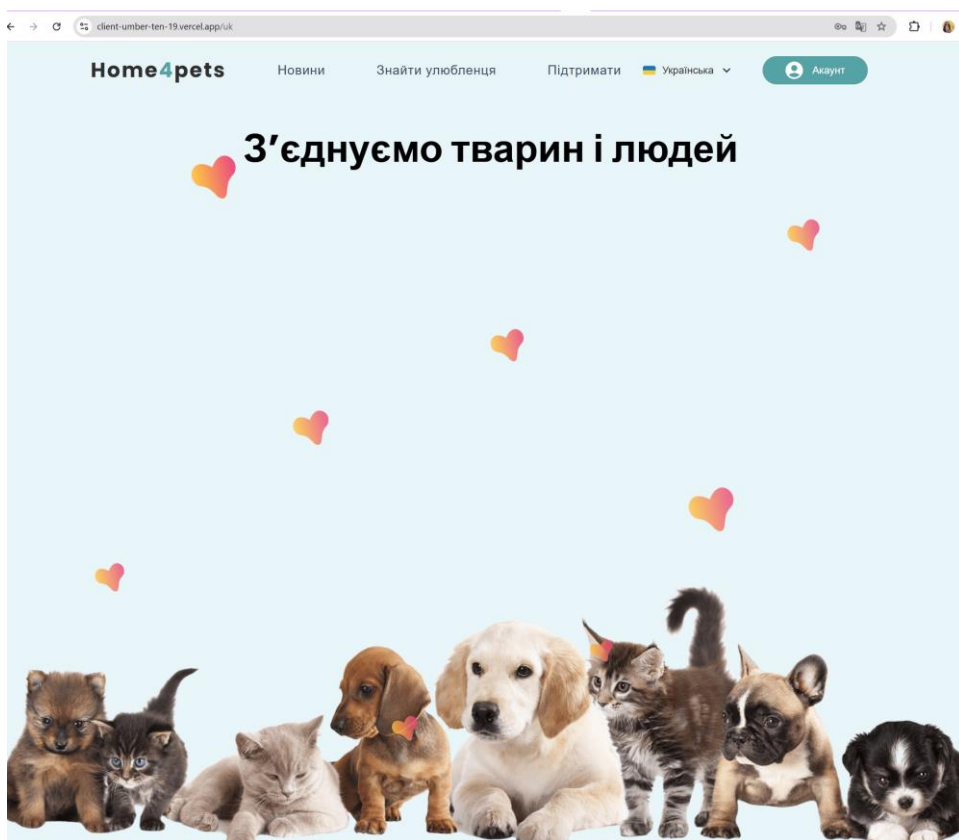


Рис. 3.26 - Розгорнута вебплатформа Home4Pets

3.3 Тестування функціоналу платформи

На завершальному етапі розробки вебплатформи для всиновлення тварин було здійснено комплексне тестування її функціональних і нефункціональних характеристик. Основною метою тестування є перевірка швидкодії, надійності та відповідності вебзастосунку сучасним стандартам продуктивності та доступності. Також важливо було визначити потенційні вузькі місця в інтерфейсі або коді, які можуть вплинути на досвід кінцевого користувача.

Для проведення тестування було використано два провідних інструменти:

1. GTMetrix – це популярний онлайн-інструмент для оцінки продуктивності сайтів, який надає розширену аналітику завдяки поєднанню технологій Google Lighthouse та Web Vitals. GTMetrix генерує глибокі звіти про час повного завантаження сторінки (Time to Fully Loaded), показник Largest Contentful Paint (LCP), час до повної взаємодії (Time to Interactive), об'єм ресурсів, Waterfall-діаграму запитів і багато іншого. Це дозволяє виявити затримки, надлишкові скрипти, блокування рендерингу та інші критичні фактори [32].

2. Lighthouse – вбудований інструмент в Google Chrome, який дозволяє локально перевірити основні метрики сайту: продуктивність, доступність, SEO-оптимізацію, найкращі практики програмування. Lighthouse виконує аудит вебсторінки та генерує оцінку за кількома ключовими параметрами, що є особливо корисним при оптимізації SPA-додатків (single-page application), зокрема таких, як наш React-застосунок [33].

На рис. 3.23 представлено головну панель результатів тестування веб-додатку, проведеного за допомогою інструменту GTMetrix. Сканування було здійснено через публічне посилання, створене засобом тунелювання Ngrok, що дозволяє тимчасово зробити локальний сервер доступним для зовнішніх сервісів [34].

У верхній частині звіту вказано URL сторінки (<https://ffc0-77-75-150-169.ngrok-free.app>), а також дата та час проведення тестування, локація сервера (Vancouver, Canada) і браузерна конфігурація (Chrome 125.0.0.0, Lighthouse 12.3.0).

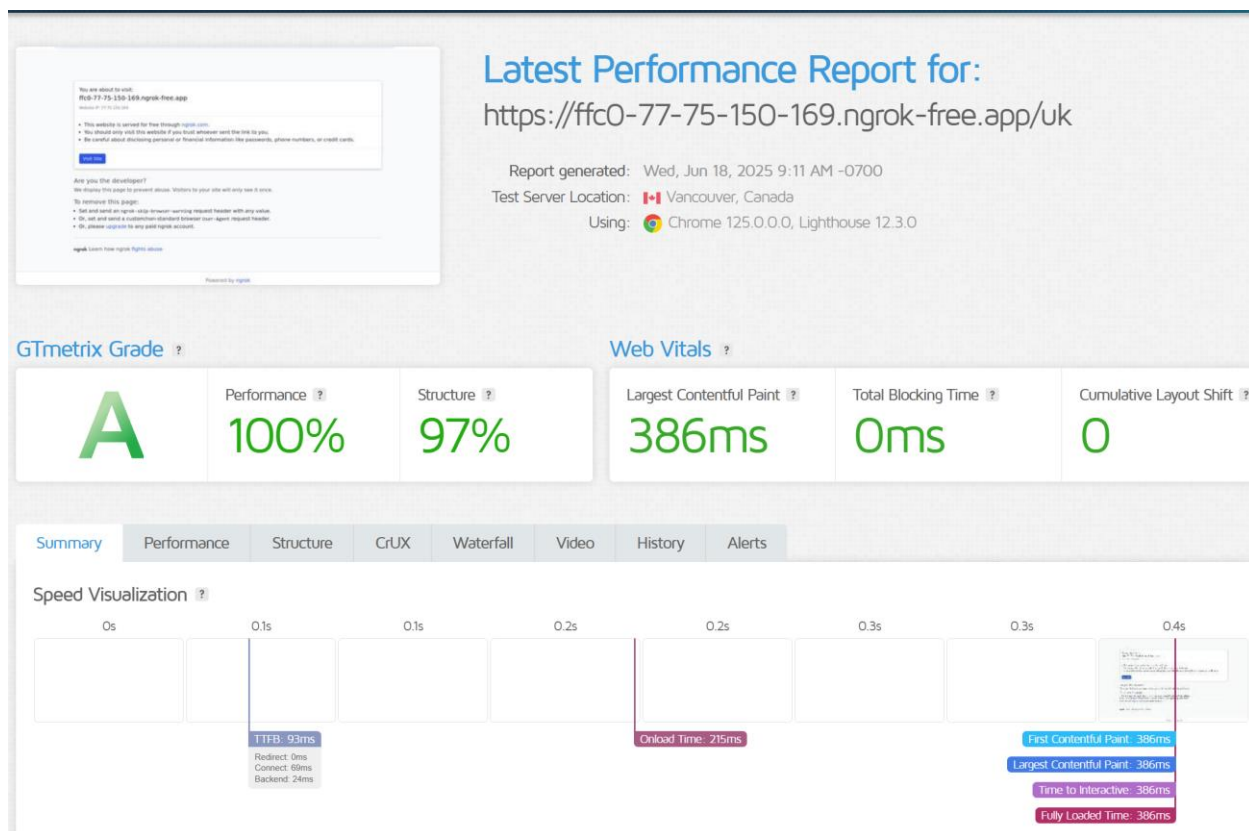


Рис. 3.23 - Загальна оцінка продуктивності сторінки за GTMetrix

GTMetrix надає інтегральну оцінку сторінки у вигляді GTMetrix Grade, яка у нашому випадку дорівнює A (найвищий клас якості). Складові оцінки:

- Performance: 100% – свідчить про миттєве завантаження та високу ефективність JavaScript, CSS та зображень;
- Structure: 97% – показує, що верстка, HTML-структура та компонування DOM реалізовані відповідно до найкращих практик;
- Web Vitals:
 - o Largest Contentful Paint (LCP) – 386 мс;
 - o Total Blocking Time (TBT) – 0 мс;
 - o Cumulative Layout Shift (CLS) – 0.

Ці метрики відповідають ідеальним показникам, які вказують на повну відсутність блокувань, зміщень верстки або затримок завантаження вмісту.

Рис. 3.24 ілюструє секцію Performance GTMetrix, яка деталізує продуктивність за ключовими метриками, отриманими від ядра Lighthouse:

- First Contentful Paint (FCP) – 386 мс: час, коли вперше відображається вміст на екрані користувача;
- Speed Index – 385 мс: показник, що відображає швидкість завантаження візуального вмісту;
- Largest Contentful Paint (LCP) – 386 мс: момент, коли завантажуються найбільший блок контенту (як правило, велике зображення або заголовок);
- Time to Interactive (TTI) – 386 мс: час до моменту, коли сторінка стає повністю інтерактивною;

Рис. 3.24 - Деталізація продуктивності за показниками Lighthouse у GTMetrix

- Total Blocking Time (TBT) – 0 мс: повна відсутність JavaScript-блокувань під час рендерингу;
- Cumulative Layout Shift (CLS) – 0: показник, що вказує на повну стабільність верстки без візуальних зміщень під час завантаження.

Кожна з цих метрик отримала позначку “Good – Nothing to do here”, що означає, що немає потреби в оптимізації на жодному з етапів.

Latest Performance Report for:

<https://ffc0-77-75-150-169.ngrok-free.app/uk>

Report generated: Wed, Jun 18, 2025 9:11 AM -0700

Test Server Location: Vancouver, Canada

Using: Chrome 125.0.0.0, Lighthouse 12.3.0

GTmetrix Grade ?

Performance ? **100%**

Structure ? **97%**

Web Vitals ?

Largest Contentful Paint ? **386ms**

Total Blocking Time ? **0ms**

Cumulative Layout ? **0**

Summary Performance Structure CrUX **Waterfall** Video History Alerts

Waterfall Chart

A request-by-request visualization of the page load. [Learn how to read a waterfall chart.](#)

Filter requests...

All HTML JS CSS Images Video XHR Fonts Other

ERR_NGROK_6024 - You are about to visit ffc0-77-75-150-169...website if you trust whoever sent the link to you.

URL	Status	Domain	Size	Timeline
* uk	200	ffc0-77-75-150-169.n...	2.83KB	94ms
* EuclidSquare-Regular-WebS.woff	200	cdn.ngrok.com	23.5KB	176ms
* EuclidSquare-RegularItalic-WebS.woff	200	cdn.ngrok.com	24.3KB	169ms
* EuclidSquare-Medium-WebS.woff	200	cdn.ngrok.com	23.4KB	198ms
* EuclidSquare-Semibold-WebS.woff	200	cdn.ngrok.com	23.4KB	198ms
* EuclidSquare-MediumItalic-WebS.woff	200	cdn.ngrok.com	24.5KB	198ms
* IBM Plex Mono-Text.woff	200	cdn.ngrok.com	49.6KB	198ms
* IBM Plex Mono-TextItalic.woff	200	cdn.ngrok.com	56.3KB	198ms
* IBM Plex Mono-SemiBold.woff	200	cdn.ngrok.com	51.1KB	198ms
* IBM Plex Mono-SemiBoldItalic.woff	200	cdn.ngrok.com	56.7KB	199ms
* error.css	200	cdn.ngrok.com	252B	108ms
* error.js	200	cdn.ngrok.com	467B	108ms
* gaopout_signal.js	200	flaagicoojcltmeiodndf...	181B	76ms
* allerrors.js	200	cdn.ngrok.com	84.9KB	29ms
* allerrors.css	200	cdn.ngrok.com	18.2KB	

Рис. 3.25 - Графік завантаження ресурсів (Waterfall Chart) GTMetrix

У нашому випадку:

- HTML-документ (uk) завантажується за ~94 мс;
- шрифти EuclidSquare та IBM Plex Mono завантажуються з CDN з мінімальною затримкою до 199 мс;
- основні стилі CSS (error.css, allerrors.css) мають розмір до 20 КБ і завантажуються швидко;
- загальна тривалість завантаження всіх ресурсів – до 386 мс, що є кращим результатом.

Жоден із запитів не має статусу помилки або значного очікування (blocking), що свідчить про добру оптимізацію як фронтенд-коду, так і мережевої архітектури.

Таким чином, додаток повністю відповідає сучасним рекомендаціям для доступності, SEO та UX-дизайну, що робить його інклюзивним та зручним для широкого кола користувачів (рис.3.26).

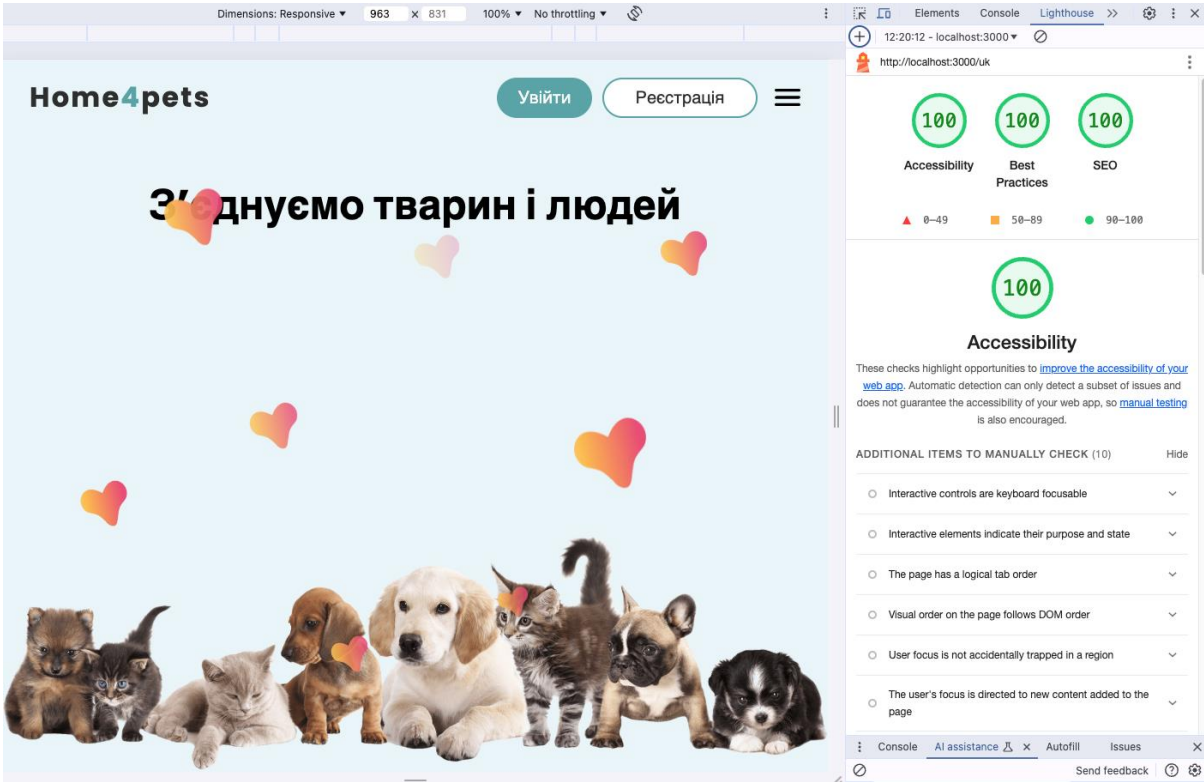


Рис. 3.26 - Результати Lighthouse-аудиту в Chrome DevTools

Результати тестування підтвердили, що вебплатформа є високопродуктивною, швидко завантажується, не має блокуючих скриптів і відповідає сучасним вебстандартам. GTMetrix і Lighthouse показали стабільну роботу, відмінну адаптивність та зручність для користувачів. Всі виявлені показники відповідають оптимальним значенням для продуктивного веб-додатку. Це забезпечує позитивний користувацький досвід як на мобільних, так і на десктопних пристроях.

Висновки до третього розділу

У третьому розділі роботи було надано всебічний опис практичної реалізації вебплатформи для пошуку та всиновлення домашніх тварин.

Розроблений застосунок охоплює повний функціонал сучасного вебресурсу — від реєстрації та авторизації користувачів до додавання оголошень і прийому фінансової підтримки через Stripe. Кожен компонент було реалізовано з урахуванням принципів масштабованості, зручності використання, безпеки та продуктивності.

У рамках реалізації користувацького інтерфейсу було впроваджено адаптивну верстку, що забезпечує коректне відображення інтерфейсу на пристроях із різною шириною екрана. Інтерфейс особистого кабінету надає користувачам можливість редагувати свої дані, додавати улюбленців, створювати оголошення з фотографіями, поведінковими та медичними особливостями, що значно розширює функціональність платформи.

На бекенді забезпечено обробку запитів, збереження інформації до бази MongoDB, а також реалізовано механізми авторизації на основі JWT-токенів. Важливим аспектом стало тестування продуктивності – платформа була проаналізована за допомогою GTMetrix та Google Lighthouse, що дозволило підтвердити високу швидкість завантаження, відсутність блокувань та відповідність сучасним стандартам доступності й оптимізації.

Таким чином, розроблена платформа є технічно завершеним, інклюзивним і продуктивним рішенням, що готове до розгортання у публічному середовищі, масштабування та подальшого функціонального розширення відповідно до цілей організації.

ВИСНОВКИ

Нами розроблено вебплатформу для пошуку, адаптації та прилаштування домашніх тварин, що відповідає сучасним вимогам зручності, швидкодії та доступності. Реалізація платформи дозволяє забезпечити ефективну взаємодію між користувачами, які бажають знайти або прилаштувати тварин, а також організаціями, що займаються порятунком і доглядом за ними.

Відповідно до поставленої мети, було проведено аналіз існуючих платформ з подібним функціоналом, що дозволило визначити ключові переваги та недоліки існуючих рішень і сформувати концепцію оптимального функціоналу для нашої системи. Було розроблено архітектуру програмного забезпечення, що забезпечує стабільність роботи платформи, ефективне управління даними та безпечну взаємодію користувачів.

Результати досліджень та розробки дозволили отримати наступні результати - створено базу даних у MongoDB, що містить інформацію про користувачів, оголошення, новини та збережені улюблені тварини. Завдяки використанню Mongoose забезпечено гнучке управління даними та взаємозв'язки між сутностями. Реалізовано серверну частину (Backend) на основі Express.js, що відповідає за обробку запитів, взаємодію з базою даних та авторизацію користувачів. Впроваджено JWT-аутентифікацію для безпечного зберігання токенів доступу та управління правами користувачів. Розроблено клієнтську частину (Frontend) за допомогою React.js, що забезпечує зручний користувацький інтерфейс. Використано Redux Toolkit для ефективного управління станом додатку, а React Router для організації навігації між сторінками. Здійснено інтеграцію з хмарним сховищем Cloudinary, що дозволяє зберігати та оптимізувати зображення тварин для оголошень, забезпечуючи швидке завантаження та високу якість контенту. Виконано тестування платформи, що включає перевірку працездатності основних функцій, аутентифікації, обробки запитів, коректності збереження даних та адаптивності інтерфейсу. Було усунуто виявлені недоліки та покращено продуктивність роботи.

Розроблена вебплатформа має практичну цінність, оскільки дозволяє значно спростити процес пошуку та адаптації тварин, підвищуючи ефективність комунікації між потенційними власниками та організаціями. Вона може бути використана громадськими ініціативами, притулками, ветеринарними клініками та окремими користувачами, що бажають знайти чи прилаштувати тварину.

Таким чином, розроблена платформа є сучасним і ефективним рішенням у сфері допомоги безпритульним тваринам, що сприяє їх швидкому прилаштуванню та покращенню умов пошуку нових власників.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. В Україні нараховується мінімум 140 тисяч безпритульних тварин. URL: <https://life.pravda.com.ua/society/2023/11/1/257378/>
2. Україна – серед 10 країн, де найбільше безпритульних тварин. URL: <https://suspilne.media/607399-ukraina-sered-10-krain-de-najbilse-bezpritulnih-tvarin/>
3. Протягом 2023 року в Харківській області зареєстрували 107 випадків сказу тварин. URL: https://suspilne.media/kharkiv/645044-za-tizden-skaz-viavili-u-gromadah-de-jogo-ne-bulo-z-2018-roku-na-harkivsini-mozut-provesti-vidstril-dikih-tvarin/?utm_source=chatgpt.com
4. Fernando Doglio. REST API Development with Node.js. Apress, 2018.
5. Introduction to JSON Web Tokens. URL: <https://auth0.com/docs/secure/tokens/json-web-tokens>
6. MongoDB Documentation. URL: <https://www.mongodb.com/docs/>
7. Stripe API Reference & Docs. URL: <https://docs.stripe.com/>
8. React Tutorial for Beginners. URL: <https://react.dev/learn>
9. Node.js – Official Website. URL: <https://nodejs.org/en/>
10. Express.js – Official website. URL: <https://expressjs.com/>
11. Ethan Brown. Web Development with Node and Express: Leveraging the JavaScript Stack. 2nd Edition. O'Reilly Media, 2019.
12. Mongoose: Elegant MongoDB object modeling for Node.js URL: <https://mongoosejs.com/>
13. React.js – Official Website. URL: <https://reactjs.org/>
14. Redux Toolkit Documentation URL: <https://redux-toolkit.js.org/>
15. Офіційна документація React Router URL: <https://reactrouter.com/>
16. Офіційний сайт JWT URL: <https://jwt.io/>
17. OWASP SQL Injection URL: https://owasp.org/wwwcommunity/attacks/SQL_Injection
18. OWASP-Cross-siteScripting(XSS) URL: <https://owasp.org/wwwcommunity/attacks/xss/>

19. OWASP-Cross-SiteRequestForgery (CSRF) URL: <https://owasp.org/www-community/attacks/csrf>
20. Офіційний сайт та документація PDFKit (Node.js) URL: <https://pdfkit.org/>
21. Stoyan Stefanov. React - Up & Running. 2nd Edition, 2018.
22. Офіційна документація Express.js — Routing URL: <https://expressjs.com/en/guide/routing.html>
23. Офіційна документація Express.js (Middleware) URL: <https://expressjs.com/en/guide/using-middleware.html>
24. Офіційна документація Cloudinary URL: <https://cloudinary.com/documentation>
25. NewsAPI - офіційний сайт і документація URL: <https://newsapi.org/docs/get-started>
26. React Router Tutorial: Getting Started with BrowserRouter URL: <https://www.freecodecamp.org/news/react-router-tutorial/>
27. Офіційна документація Redux Toolkit — createAsyncThunk URL: <https://redux-toolkit.js.org/api/createAsyncThunk>
28. MDNWebDocs-CSSFlexibleBoxLayoutURL: [https://developer.mozilla.org/enUS/docs/Web/CSS/CSS_flexible_box layout/Basic concepts of flexbox](https://developer.mozilla.org/enUS/docs/Web/CSS/CSS_flexible_box_layout/Basic_concepts_of_flexbox)
29. MDNWeb Docs — CSS Grid Layout URL: [https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_grid layout](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_grid_layout)
30. Офіційна документація React Hook Form URL: <https://react-hook-form.com/>
31. Офіційна документація i18next URL: <https://www.i18next.com/>
32. Офіційний сайт GTMetrix URL: <https://gtmetrix.com/>
33. Офіційна документація Google Lighthouse URL: <https://developer.chrome.com/docs/lighthouse/overview/>
34. Офіційний сайт Ngrok URL: <https://ngrok.com/>

ДОДАТКИ

Ініціалізація серверного застосунку та підключення до бази даних (server.js)

```
const app = require("./app");
const { exit } = require("node:process");
require("dotenv").config();

const { PORT = 8080 } = process.env;

const { connectionMongo } = require("./dbMongo/connectionMongodb");

const start = async () => {
  try {
    await connectionMongo();
    app.listen(PORT, (err) => {
      if (err) {
        console.error(`Error at server launch:`, err);
        exit(1);
      }
      console.log(
        `Database connection successful. Server running. Use our API on port: ${PORT}`
      );
    });
  } catch (err) {
    console.error(
      `Server not running. Failed to launch application with error: ${err.message}`
    );
    exit(1);
  }
};

start();
```

Функція автоматичної генерації документа всиновлення тварини у форматі PDF

```

const PDFDocument = require("pdfkit");
const fs = require("fs");
const path = require("path");

const generateAdoptionDocument = (petData, userData) => {
  return new Promise((resolve, reject) => {
    const doc = new PDFDocument();
    const fileName = `adoption_${petData._id}_${userData._id}.pdf`;
    const filePath = path.join(__dirname, `../temp/${fileName}`);

    const stream = fs.createWriteStream(filePath);
    doc.pipe(stream);

    doc.fontSize(26).text("Pet Adoption Agreement", { align: "center" });
    doc.moveDown(1.5);

    doc.moveTo(50, doc.y).lineTo(550, doc.y).stroke();
    doc.moveDown(1);

    doc.fontSize(18).text("Pet Information", { underline: true });
    doc.moveDown(0.5);
    doc
      .fontSize(12)
      .text(`Name: ${petData.name || "N/A"}`)
      .text(`Breed: ${petData.breed || "N/A"}`)
      .text(`Birthday: ${petData.birthday || "N/A"}`)
      .text(`Location: ${petData.location || "N/A"}`)
      .moveDown(1);

    if (petData.healthHistory && petData.healthHistory.length > 0) {
      doc.fontSize(18).text("Health History", { underline: true });
      doc.moveDown(0.5);

      petData.healthHistory.forEach((vaccine, index) => {
        doc
          .fontSize(12)
          .text(
            `${index + 1}. Vaccine: ${vaccine.vaccine}, Date: ${new Date(
              vaccine.date
            ).toLocaleDateString()}`
          );
      });
    }
  });
};

```

```

        doc.moveDown(1);
    }

    if (petData.behavioralNotes) {
        doc.fontSize(18).text("Behavioral Notes", { underline: true });
        doc.moveDown(0.5);
        doc.fontSize(12).text(`${petData.behavioralNotes}`);
        doc.moveDown(1);
    }

    if (petData.specialNeeds) {
        doc.fontSize(18).text("Special Needs", { underline: true });
        doc.moveDown(0.5);
        doc.fontSize(12).text(`${petData.specialNeeds}`);
        doc.moveDown(1);
    }

    doc.moveTo(50, doc.y).lineTo(550, doc.y).stroke();
    doc.moveDown(1);

    doc.fontSize(18).text("New Owner Information", { underline: true });
    doc.moveDown(0.5);
    doc
        .fontSize(12)
        .text(`Name: ${userData.name || "N/A"}`)
        .text(`Email: ${userData.email || "N/A"}`)
        .text(`Phone: ${userData.phone || "N/A"}`)
        .moveDown(1);

    doc.fontSize(12).text(`Adoption Date: ${new Date().toLocaleDateString()}`);
    doc.moveDown(2);

    doc.text("_____", { align: "left" });
    doc.text("Signature of New Owner", { align: "left" });

    doc.end();

    stream.on("finish", () => resolve(filePath));
    stream.on("error", (err) => reject(err));
});
};

module.exports = generateAdoptionDocument;

```

Інтеграція з платіжною системою Stripe

```

const express = require("express");
const router = express.Router();

const stripe = require("stripe")(process.env.STRIPE_SECRET_KEY);

router.post("/create-session", async (req, res) => {
  try {
    const { items } = req.body;

    if (!items || !Array.isArray(items) || !items[0]?.amount) {
      return res.status(400).json({ error: "Invalid donation data" });
    }

    const amount = parseInt(items[0].amount);

    const session = await stripe.checkout.sessions.create({
      payment_method_types: ["card"],
      mode: "payment",
      locale: "auto",
      line_items: [
        {
          price_data: {
            currency: "uah",
            product_data: {
              name: "Donation for Animal Support",
            },
            unit_amount: amount * 100,
          },
          quantity: 1,
        },
      ],
      success_url: `${process.env.CLIENT_URL}/support?donation=success`,
      cancel_url: `${process.env.CLIENT_URL}/support?donation=cancel`,
    });

    res.status(200).json({ id: session.id });
  } catch (error) {
    console.error("Stripe session error:", error);
    res.status(500).json({ error: "Failed to create Stripe session" });
  }
});

module.exports = router;

```

Додаток Д.1. Redux Operations для оголошень

```

import { createAsyncThunk } from '@reduxjs/toolkit';
import * as api from '../shared/api/notices';

export const getNoticeByCategory = createAsyncThunk(
  'notices/getNoticeByCategory',
  async (params, { rejectWithValue, getState }) => {
    const {
      notices: { queryParams },
    } = getState();
    try {
      const result = await api.getNoticeByCategory(queryParams);
      if (!['favorite', 'own'].includes(queryParams.category)) {
        return { notices: result };
      }
      return result;
    } catch ({ response }) {
      const { status, data } = response;
      const error = {
        status,
        message: data.message,
      };
      return rejectWithValue(error);
    }
  }
);

export const getOneNotice = createAsyncThunk(
  'notices/getOneNotice',
  async (id, { rejectWithValue }) => {
    try {
      const result = await api.getOneNotice(id);
      return result;
    } catch ({ response }) {
      const { status, data } = response;
      const error = {
        status,
        message: data.message,
      };
      return rejectWithValue(error);
    }
  }
);

export const addNoticeToFavorite = createAsyncThunk(
  'notices/addNoticeToFavorite',

```

```

async (id, { rejectWithValue }) => {
  try {
    const result = await api.addNoticeToFavorite(id);
    return result;

  } catch ({ response }) {
    const { status, data } = response;
    const error = {
      status,
      message: data.message,
    };
    return rejectWithValue(error);
  }
});

export const getFavoriteNoticeForCategories = createAsyncThunk(
  'notices/getFavoriteNoticeForCategories',
  async (_, { rejectWithValue, getState }) => {
    try {
      const result = await api.getFavoriteNotices();
      return result;
    } catch ({ response }) {
      const { status, data } = response;
      const error = {
        status,
        message: data.message,
      };
      return rejectWithValue(error);
    }
  });

export const getFavoriteNoticeByUser = createAsyncThunk(
  'notices/getFavoriteNoticeByUser',
  async (_, { rejectWithValue }) => {
    try {
      const result = await api.getFavoriteNotices();
      return result;
    } catch ({ response }) {
      const { status, data } = response;
      const error = {
        status,
        message: data.message,
      };
      return rejectWithValue(error);
    }
  });

```

```

    }
  }
);

export const removeNoticeFromFavorite = createAsyncThunk(
  'notices/removeNoticeFromFavorite',
  async (id, { rejectWithValue }) => {
    try {
      const result = await api.removeNoticeFromFavorite(id);
      return result;

    } catch ({ response }) {
      const { status, data } = response;
      const error = {
        status,
        message: data.message,
      };
      return rejectWithValue(error);
    }
  }
);

```

```

export const addNotice = createAsyncThunk(
  'notices/addNotice',
  async (data, { rejectWithValue }) => {
    try {
      const result = await api.addNotice(data);
      return result;
    } catch ({ response }) {
      const { status, data } = response;
      const error = {
        status,
        message: data.message,
      };
      return rejectWithValue(error);
    }
  }
);

```

```

export const getUserNotices = createAsyncThunk(
  'notices/getUserNotices',
  async (_, { rejectWithValue }) => {
    try {
      const result = await api.getUserNotices();
      return result;
    } catch ({ response }) {

```

```

        const { status, data } = response;
        const error = {
            status,
            message: data.message,
        };
        return rejectWithValue(error);
    }
}
);

export const removeNotice = createAsyncThunk(
    'notices/removeNotice',
    async (id, { rejectWithValue }) => {
        try {
            const result = await api.removeNotice(id);
            return result;
        } catch ({ response }) {

            const { status, data } = response;
            const error = {
                status,
                message: data.message,
            };
            return rejectWithValue(error);
        }
    }
);

```

Додаток Д. 2. Redux Selectors для оголошень

```

export const getNoticesError = ({ notices }) => notices.error;
export const getNoticesLoading = ({ notices }) => notices.loading;
export const getCurrentNotices = ({ notices }) => notices.currentNotices;
export const getFavoriteNotices = ({ notices }) => notices.favorite;
export const getTotalPages = ({ notices }) => notices.totalPages;
export const getFilter = ({ notices }) => notices.queryParams.filter;
export const getCategory = ({ notices }) => notices.queryParams.category;
export const getPage = ({ notices }) => notices.queryParams.page;
export const getIsAddedSuccess = ({ notices }) => notices.isAddedSuccess;

```

Додаток Д. 3. Redux Slice для оголошень

```

import { createSlice } from '@reduxjs/toolkit';

```

```

import storageSession from 'redux-persist/lib/storage/session';
import { persistReducer } from 'redux-persist';

import {
  getNoticeByCategory,
  addNotice,
  addNoticeToFavorite,
  getFavoriteNoticeForCategories,
  getFavoriteNoticeByUser,
  removeNoticeFromFavorite,
  getUserNotices,
  removeNotice,
} from './notices-operations';

const initialState = {
  currentNotices: [],
  favorite: [],
  loading: false,
  error: null,
  totalPages: 1,
  isAddedSuccess: false,
  queryParams: {
    filter: '',
    category: 'for-free',
    page: 1,
  },
};

const noticesSlice = createSlice({
  name: 'notices',
  initialState,
  reducers: {
    setQueryParams: (state, { payload }) => {
      const {
        page = 1,
        filter = '',
        category = state.queryParams.category,
      } = payload;
      state.queryParams.filter = filter;
      state.queryParams.category = category;
      state.queryParams.page = page;
    },
    setTotalPages: (state, { payload }) => {
      state.totalPages = payload;
    },
    resetIsAddedSuccess: state => {
      state.isAddedSuccess = false;
    },
  },
});

```

```

},
extraReducers: builder => {
  builder
    .addCase(getNoticeByCategory.pending, store => {
      store.loading = true;
      store.error = null;
    })
    .addCase(getNoticeByCategory.fulfilled, (store, { payload }) => {
      store.loading = false;
      store.currentNotices = payload.notices;
      store.totalPages = payload.totalPages;
    })
    .addCase(getNoticeByCategory.rejected, (store, { payload }) => {
      store.loading = false;
      store.error = payload;
    })
    .addCase(getUserNotices.pending, store => {
      store.loading = true;
      store.error = null;
    })
    .addCase(getUserNotices.fulfilled, (store, { payload }) => {
      store.loading = false;
      store.currentNotices = payload;
    })
    .addCase(getUserNotices.rejected, (store, { payload }) => {
      store.loading = false;
      store.error = payload;
    })
    .addCase(getFavoriteNoticeForCategories.pending, store => {
      store.loading = true;
      store.error = null;
    })

    .addCase(
      getFavoriteNoticeForCategories.fulfilled,
      (store, { payload }) => {
        store.loading = false;
        store.currentNotices = payload;
        store.favorite = payload.map(el => el._id);
      }
    )
    .addCase(
      getFavoriteNoticeForCategories.rejected,
      (store, { payload }) => {
        store.loading = false;
        store.error = payload;
      }
    )
}

```

```

    )
    .addCase(getFavoriteNoticeByUser.pending, store => {
      store.loading = true;
      store.error = null;
    })
    .addCase(getFavoriteNoticeByUser.fulfilled, (store, { payload }) => {
      store.loading = false;
      store.favorite = payload.map(el => el._id);
    })
    .addCase(getFavoriteNoticeByUser.rejected, (store, { payload }) => {
      store.loading = false;
      store.error = payload;
    })
    .addCase(addNoticeToFavorite.pending, store => {
      store.loading = true;
      store.error = null;
    })
    .addCase(addNoticeToFavorite.fulfilled, (store, { payload }) => {
      store.loading = false;
      const addedNotice = payload.favorite.length - 1;
      store.favorite.push(payload.favorite[addedNotice]);
    })
    .addCase(addNoticeToFavorite.rejected, (store, { payload }) => {
      store.loading = false;
      store.error = payload;
    })
    .addCase(removeNoticeFromFavorite.pending, store => {
      store.loading = true;
      store.error = null;
    })
    .addCase(removeNoticeFromFavorite.fulfilled, (store, { payload }) => {
      store.loading = false;
      store.favorite = store.favorite.filter(id => id !== payload.id);
    })
    .addCase(removeNoticeFromFavorite.rejected, (store, { payload }) => {
      store.loading = false;
      store.error = payload;
    })

    .addCase(addNotice.pending, store => {
      store.loading = true;
      store.error = null;
    })
    .addCase(addNotice.fulfilled, (store, { payload }) => {
      store.isAddedSuccess = true;
      store.loading = false;
      if ([payload.category, 'own'].includes(store.queryParams.category)) {

```

```

        store.currentNotices = [payload, ...store.currentNotices];
    }
})
.addCase(addNotice.rejected, (store, { payload }) => {
    store.loading = false;
    store.error = payload;
})
.addCase(removeNotice.pending, store => {
    store.loading = true;
    store.error = null;
})
.addCase(removeNotice.fulfilled, (store, { payload }) => {
    store.loading = false;
    store.currentNotices = store.currentNotices.filter(
        ({ _id }) => _id !== payload.id
    );
})
.addCase(removeNotice.rejected, (store, { payload }) => {
    store.loading = false;
    store.error = payload;
});
},
});

export const { setQueryParams, setTotalPages, resetIsAddedSuccess } =
    noticesSlice.actions;

const noticesReducer = noticesSlice.reducer;

const persistConfig = {
    key: 'noticesQueryParams',
    storage: storageSession,
    whitelist: ['queryParams'],
};

const persistedNoticesReducer = persistReducer(persistConfig, noticesReducer);

export default persistedNoticesReducer;

```

```
const cloudinary = require("cloudinary").v2;
const { CLOUD_NAME, API_KEY, API_SECRET } = process.env;

cloudinary.config({
  cloud_name: CLOUD_NAME,
  api_key: API_KEY,
  api_secret: API_SECRET,
});

const uploadToCloudinary = async (path) => {
  try {
    const result = await cloudinary.uploader.upload(path, {
      width: 450,
      height: 450,
      crop: "fill",
    });

    const fs = require("fs");
    fs.unlinkSync(path);

    return result;
  } catch (error) {
    console.error("Cloudinary Error:", error.message);
    throw new Error(error.message);
  }
};

module.exports = uploadToCloudinary;
```