

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 «Веб-технології»)

на тему:

«Вебзастосунок адопції домашніх тварин»

Виконав студент групи ВТ-21-1
ПАРХОМЕНКО Олександр Євгенійович

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
ФАНТ Микола Олександрович

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
«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: ПАРХОМЕНКО Олександр Євгенійович

Керівник роботи: САВІЦЬКИЙ Роман Святославович

Тема роботи: «Вебзастосунок адопції домашніх тварин»,

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

Вихідні дані для роботи: вебзастосунок для оптимізації процесів адопції, процес організації адопції домашніх тварин.

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

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра складається з вступу, трьох розділів, висновку і списку використаних джерел.

Текстова частина викладена на 65 сторінках друкованого тексту. Список використаних джерел містить 17 найменувань і займає 2 сторінки. В роботі наведено 13 таблиць, 38 рисунків і 4 додатки. Загальний обсяг роботи – 86 сторінок.

У вступі було описано актуальність теми, мету дослідження, завдання випускної кваліфікаційної роботи бакалавра, об'єкт дослідження, предмет дослідження та практичне значення отриманих результатів.

У першому розділі було виконано аналіз аналогів системи для дослідження ринку і визначення основного функціоналу, а також визначено архітектуру програмного продукту й обгрунтовано вибір інструментальних засобів.

У другому розділі було проведено архітектурне й функціональне проєктування вебзастосунку, а також наведено UML-діаграми для використання системи, діаграму БД і діаграми діяльності для основних алгоритмів програми.

У третьому розділі було реалізовано кінцевий програмний продукт у відповідності до поставлених вимог. Було описано розгортання вебзастосунку, а також всіх його компонентів. Також було проведено тестування для уникнення критичних помилок на етапі розробки.

Ключові слова: АДОПЦІЯ, Next.js, TS, PostgreSQL, Prisma, Clerk.

					ІПЗ.КР.Б – 121 – 25 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з випускної кваліфікаційної роботи	Літ.	Арк.	Аркушів	
Розроб.		Пархоменко О.Є.							
Керівник		Савіцький Р.С.					2	86	
						ФІКТ Гр. ВТ-21-1			
Н. контр.									
Зав. каф.									

ABSTRACT

The Bachelor's qualification work report consists of an introduction, three chapters, a conclusion, and a list of references. The textual part is presented on 65 pages of printed text. The list of references contains 17 sources and takes up 2 pages. The report includes 13 tables, 38 figures and 4 appendices. The total length of the report is 86 pages.

The introduction describes the relevance of the topic, the purpose of the research, the tasks of the Bachelor's qualification work, the object and subject of the research, and the practical significance of the obtained results.

In the first chapter, an analysis of similar systems for market research was conducted, the main functionality was defined, the software architecture was determined, and the choice of development tools was justified.

In the second chapter, the architectural and functional design of the web application was carried out, including UML diagrams for system usage, a database diagram, and activity diagrams for the main algorithms of the program.

In the third chapter, the final software product was implemented according to the defined requirements. The deployment of the web application and all its components was described. Testing was also carried out to avoid critical errors during the development stage.

Keywords: ADOPTION, Next.js, TS, PostgreSQL, Prisma, Clerk.

					ІПЗ.КР.Б – 121 – 25 – ПЗ								
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з випускної кваліфікаційної роботи				Літ.		Арк.	Аркушів	
Розроб.		Пархоменко О.Є.											
Керівник		Савіцький Р.С.										2	86
									ФІКТ Гр. ВТ-21-1				
Н. контр.													
Зав. каф.													

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	10
1.1 Постановка задачі.....	10
1.2 Аналіз аналогів програмного продукту	11
1.3 Вибір архітектури системи вебзастосунку адопції домашніх тварин ..	14
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	17
Висновки до першого розділу	23
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ АДОПЦІЇ ДОМАШНІХ ТВАРИН	24
2.1 Визначення варіантів використання та об’єктно-орієнтованої структури системи	24
2.2 Розробка бази даних системи	26
2.3 Проектування та реалізація алгоритмів роботи системи	30
2.4 Реалізація вебзастосунку адопції домашніх тварин	36
Висновки до другого розділу.....	45
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБЗАСТОСУНКОМ АДОПЦІЇ ДОМАШНІХ ТВАРИН	46
3.1 Розгортання вебзастосунку.....	46
3.2 Структура інтерфейсу вебзастосунку адопції домашніх тварин	47

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3 Тестування роботи програмного продукту	57
Висновки до третього розділу.....	61
ВИСНОВКИ	62
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	64
ДОДАТКИ.....	66
Додаток А.....	67
Додаток Б.....	80
Додаток В.....	81
Додаток Г.....	84

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTML (HyperText Markup Language) – мова гіпертекстової розмітки

JSX (JavaScript XML) – синтаксичне розширення для написання HTML-подібного коду всередині JavaScript

CSS (Cascade Style Sheets) – каскадні таблиці стилів

TS – мова програмування TypeScript

СУБД – Система управління базами даних

ORM (англ. Object-Relational Mapping) – технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних»

БД – база даних

ПЗ – програмне забезпечення

ШІ – штучний інтелект

API (англ. Application Programming Interface) – інтерфейс прикладного програмування, набір чітко визначених методів для взаємодії різних компонентів

SSR (англ. Server-Side Rendering) – рендеринг на стороні сервера

SPA (англ. Single-page Application) – односторінковий застосунок

SDK (англ. Software Development Kit) – набір інструментів, бібліотек і документацій для розробки програмного забезпечення

SQL (англ. Structured Query Language) – стандартна мова запитів для роботи з реляційними базами даних

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Процес адопції домашніх тварин має низку проблем, таких як перевантажені притулки, складність пошуку нових власників та недостатня прозорість процедур. Вебзастосунок може розв'язати ці проблеми, забезпечивши автоматизацію, зручність та прозорість процесів. Розробка вебзастосунку адопції дозволить оптимізувати управління базами тварин, а також полегшити взаємодію між користувачами та центром адопції.

Адопція відіграє ключову роль у зменшенні кількості безпритульних тварин і поширенні відповідального ставлення до домашніх улюбленців, що є важливим кроком до більш свідомого суспільства.

Створення вебзастосунку адопції домашніх тварин може допомогти у виборі ідеального улюбленця за рахунок штучного інтелекту. Подібних функціональних можливостей не було реалізовано у наявних системах, а отже, такий підхід може знизити кількість тварин, що повертають у центри адопції через складність утримання і невідповідність умовам життя власника.

Мета дослідження – розробка вебзастосунку для автоматизації процесів адопції домашніх тварин, покращення взаємодії з центром адопції та збільшення кількості успішних усиновлень.

Завдання випускної кваліфікаційної роботи бакалавра:

- проаналізувати існуючі системи адопції;
- дослідити потреби користувачів;
- розробити архітектуру системи;
- обґрунтувати вибір інструментальних засобів системи;
- визначити варіанти використання та структуру системи;
- розробити базу даних;
- провести проектування та реалізацію алгоритмів роботи програми;
- провести проектування структури інтерфейсу вебзастосунку;
- розгорнути вебзастосунок у хмарному середовищі;

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

- провести тестування програмного продукту на предмет критичних помилок.

Об’єкт дослідження. Процес організації адопції домашніх тварин.

Предмет дослідження. Вебзастосунок для оптимізації процесів адопції.

Практичне значення одержаних результатів. Розроблений вебзастосунок допомагає спростити процес підбору відповідних власників для тварин, що шукають новий дім. Використання автоматизованого аналізу анкет дозволяє зменшити ризик повернення тварин у притулок і підвищити рівень відповідального власництва. Крім того, система може бути використана центрами адопції для більш ефективного управління процесом адопції.

За темою випускної кваліфікаційної роботи бакалавра було опубліковано тезу на тему “Використання штучного інтелекту у Next.js” [1].

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1 Постановка задачі

Основна мета роботи є розробка вебзастосунку адопції домашніх тварин задля підвищення ефективності і спрощення процесу взяття відповідальності на домашніх тварин з притулків.

Система вебзастосунку адопції домашніх тварин включає низку ключових програмних модулів, які забезпечують ефективну роботу сервісу. Основним є модуль каталогу тварин, що дозволяє користувачам переглядати доступних до адопції тварин із можливістю фільтрації за категоріями та на основі особистих даних користувача. Модуль адопції охоплює форму для подання заявки на адопцію, після заповнення якої штучний інтелект аналізує введені дані і надає адміністратору висновок із поясненням, чи можна користувачу взяти відповідальність на конкретну тварину. Модуль адміністрування центру включає систему управління вмістом додатку. Освітній модуль містить базу знань зі статтями про догляд за тваринами, а також рекомендації щодо використання кожної сторінки вебзастосунку, що працює на основі штучного інтелекту і аналізує дані, такі як шлях поточної сторінки і роль користувача. Модуль інтеграцій і комунікацій забезпечує розсилку сповіщень на електронну пошту після прийняття або відхилення заявки на адопцію адміністратором, а також інтеграцію чат-бота для швидких відповідей. Така система дозволяє задовольнити потреби всіх категорій користувачів і автоматизувати основні процеси діяльності центру адопції.

Основними етапами розробки вебзастосунку є аналіз вимог і проектування, розробка технічного стеку, створення модулів, тестування та вдосконалення. На першому етапі визначаються цілі вебзастосунку й потреби користувачів, збираються вимоги до функціоналу, проектується архітектура системи й бази даних, створюються прототипи інтерфейсів. Далі обираються технології для фронтенду й бекенду, налаштовується середовище розробки, визначаються структура API та підхід до безпеки. Після цього розробляються основні модулі:

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

бекенд-логіка, інтерфейси для користувачів, а також реалізуються інтеграції з зовнішніми сервісами. На етапі тестування проводиться мануальне тестування, перевіряється виконання основного функціоналу, взаємодія між компонентами вебзастосунку, оцінюється безпека, продуктивність і зручність користування. Завершальним етапом є виправлення недоліків, оптимізація продуктивності та впровадження нових функцій. Така послідовність забезпечує поступовий розвиток системи та її ефективну роботу.

1.2 Аналіз аналогів програмного продукту

Основна мета вебзастосунку адопції домашніх тварин полягає в автоматизації та оптимізації процесу взяття відповідальності на домашніх улюбленців різних видів, а також у покращенні взаємодії між користувачем і центром адопції. Для цього розглянемо аналоги системи для дослідження ринку.

Harpu Paw [2] (рис. 1.1) – благодійний фонд допомоги безпритульним тваринам, що займається адопцією домашніх тварин, а також допомогою загубленим тваринам, тісно взаємодіє із проєктом біоетики. Аналіз цього аналогу надихнув на ідею інтегрувати в проєкт рекомендації штучного інтелекту для підбору улюбленців, що враховують як зовнішні параметри, так і поведінкові особливості.

ЛКП “Лев” [3] (рис. 1.2) – це ветеринарно-стерилізаційний центр для тварин, що використовує досвід Європи і застосовує власну модель регулювання чисельності безпритульних тварин. Вивчення діяльності цього центру допомогло зрозуміти, наскільки важливо надавати чітку й структуровану інформацію. Це стало основою для інтеграції в проєкт чат-бота з ШІ-асистентом, який може надавати необхідну інформацію, а також для впровадження системи контекстних підказок для зручного користування вебзастосунком.

GladPet [4] (рис. 1.3) – це онлайн-ресурс з прилаштування (адопції) та допомоги безхатнім тваринам. Аналіз даного ресурсу дав можливість краще зрозуміти важливість зручного інтерфейсу й фільтрації тварин за різними

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

критеріями, що допомогло під час розробки користувацького інтерфейсу та реалізації рекомендацій на основі ШІ.

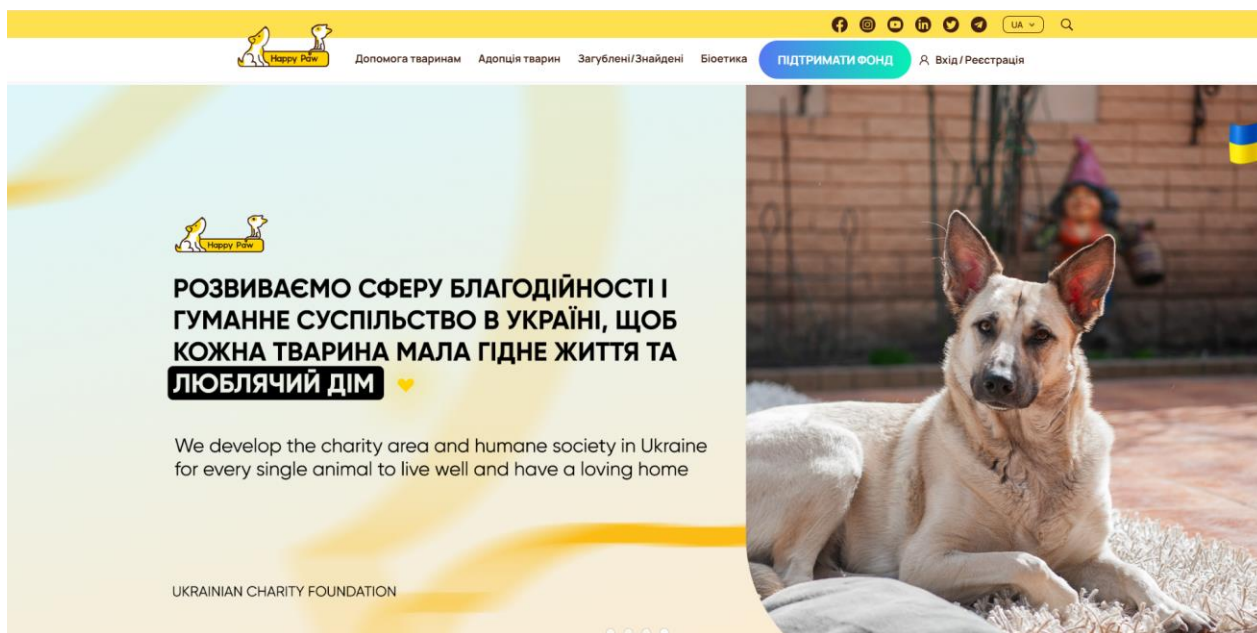


Рисунок 1.1. Happy Paw

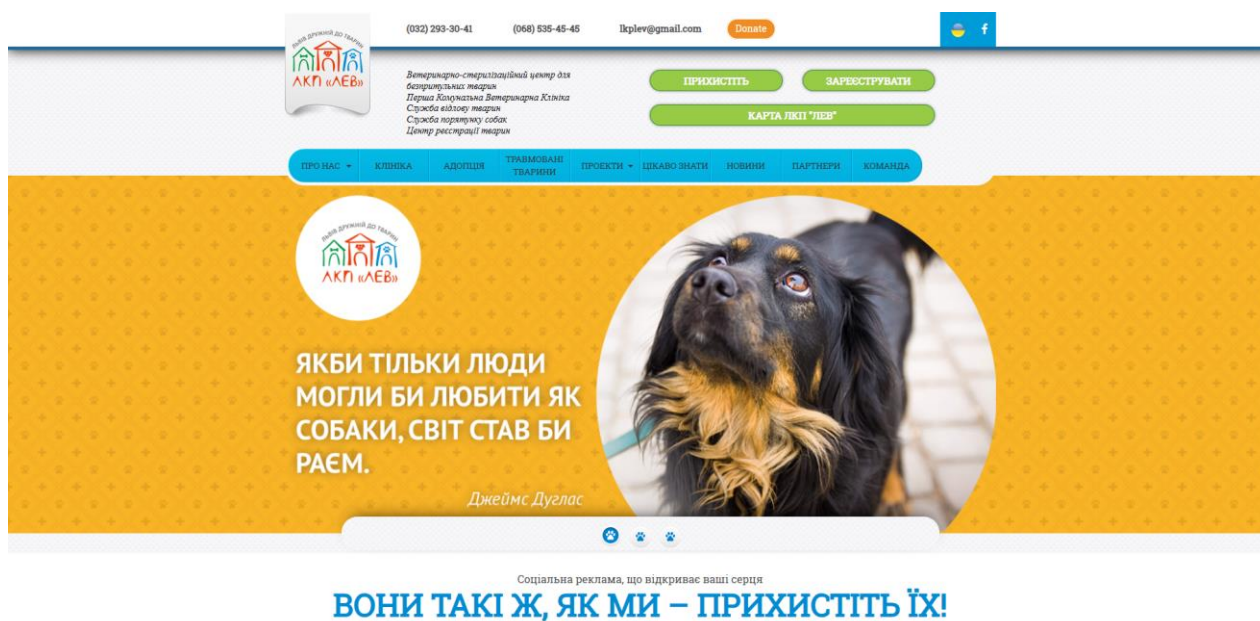


Рисунок 1.2. ЛКП “Лев”

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

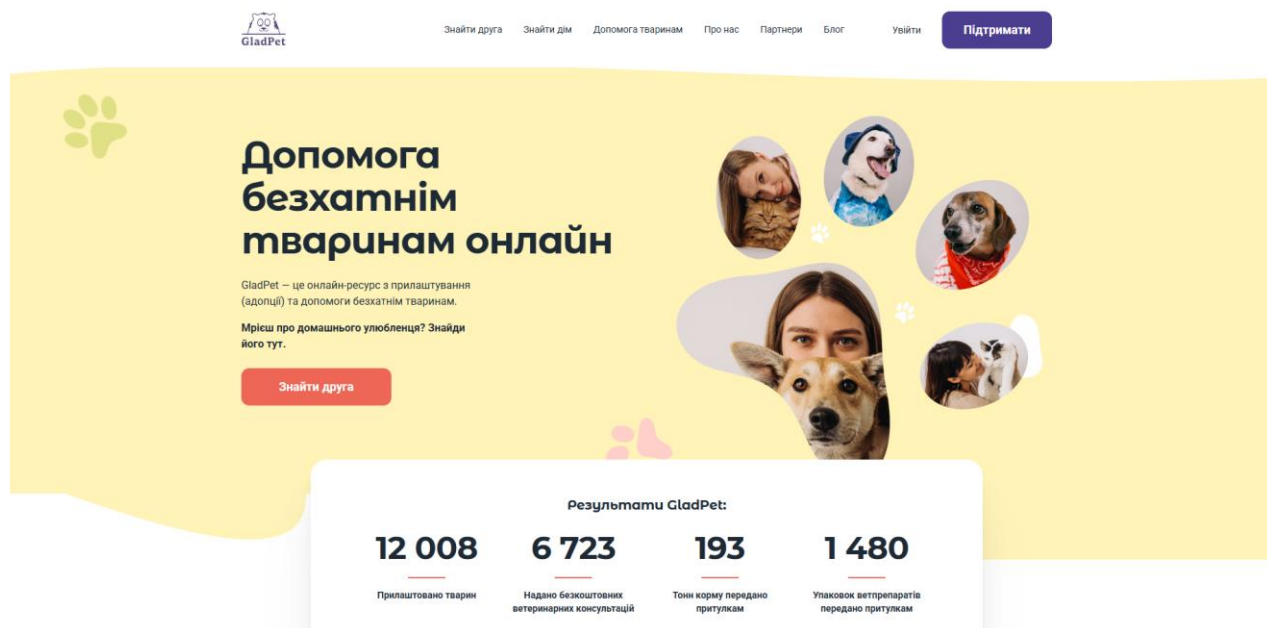


Рисунок 1.3. GladPet

Порівняння функціональних можливостей продуктів приведено нижче (табл. 1.1).

Таблиця 1.1

Функціональні можливості продуктів

Критерій	Happy Paw	ЛКП “Лев”	GladPet	Paw in Paw
Зручність вебінтерфейсу	Так	Так	Так	Так
Доступність	Так	Так	Так	Так
Звітність	Так	Ні	Так	Ні
Адопція	Так	Так	Так	Так
Прилаштування	Так	Так	Так	Ні
Допомога	Так	Так	Так	Ні
Освітні матеріали	Так	Так	Так	Так
Візуальне сприйняття	Так	Ні	Так	Так
Рекомендації ІІІ	Ні	Ні	Ні	Так
Аналіз анкети адопції	Ні	Ні	Ні	Так
Чат-бот на основі ІІІ	Ні	Ні	Ні	Так
Система підказок	Ні	Ні	Ні	Так

У кожного з представлених ресурсів є незначні відмінності, що не впливають на роботу основного функціоналу.

Отже, основними характеристиками нового вебзастосунку адопції домашніх тварин мають бути:

- зручний вебінтерфейс для забезпечення доступності і візуального сприйняття;
- можливість надіслати заяву на адопцію або прилаштувати тварину;
- можливість фільтрації і отримання рекомендацій;
- освітні матеріали для ознайомлення;
- можливість керування вмістом БД через панель адміністратора;
- підтримка штучного інтелекту для допомоги вибору домашнього улюбленця і перевірки заповнених анкет, а також наявність чат-боту і підмережі для допомоги користувачам вебзастосунку;
- можливість комунікації і отримання сповіщень через email.

1.3 Вибір архітектури системи вебзастосунку адопції домашніх тварин

Для розробки вебзастосунку адопції домашніх тварин було обрано клієнт-серверну архітектуру, яка дозволяє ефективно обробляти запити користувачів і забезпечує масштабованість платформи. У цій архітектурі клієнтські додатки взаємодіють із сервером для отримання, обробки та збереження даних. Ролі чітко розподілені: клієнт відповідає за вебінтерфейс і взаємодію з користувачами, відображає інформацію про тварин, надає форми для заповнення анкет, сервер обробляє запити, виконує бізнес-логіку, взаємодіє з базою даних і забезпечує безпечне збереження інформації. Серверна частина платформи виконує такі ключові функції, як обробка запитів на перегляд каталогу тварин із фільтрами, обробка та анкет на адопцію, збереження й оновлення інформації про тварин тощо. Клієнтська частина відповідає за відображення даних, зокрема каталогу тварин із фільтрацією за категорією та на основі рекомендацій штучного інтелекту, забезпечує можливість заповнення анкет на адопцію, відображає інформаційні сторінки з порадами щодо догляду, а також інформує користувачів про статуси

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

їхніх заявок. Детальну архітектуру вебзастосунку адопції домашніх тварин наведено на рисунку 1.4.

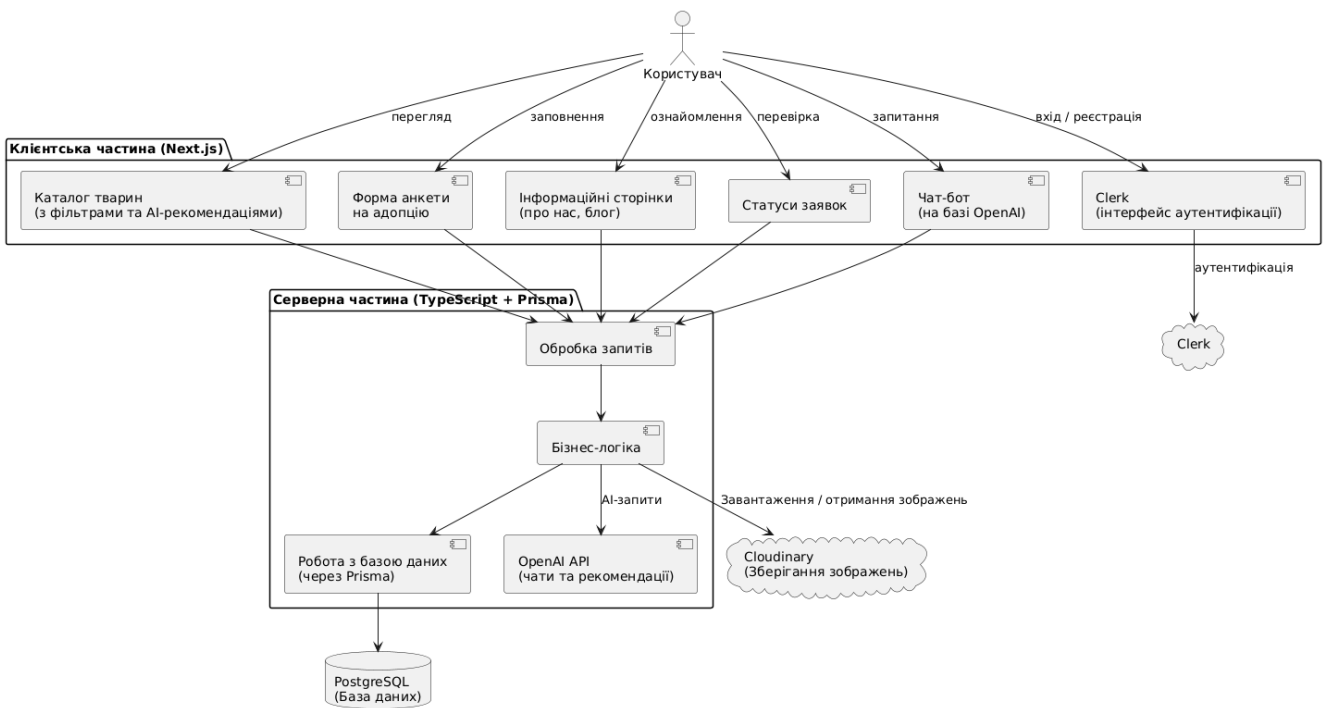


Рисунок 1.4. Архітектура вебзастосунку адопції домашніх тварин

Клієнтську частину реалізовано за допомогою SSR-фреймворку Next.js [5]. Головною особливістю підходу є рендеринг сторінки на стороні сервера при першому запиті, а далі додаток працює як SPA, виконуючи за потреби запити на сервер. Перевагою даного підходу є постійна актуальність інформації, висока швидкість завантаження та відображення контенту, можливість налаштувати кешування, висока реалізація SEO. Також даний фреймворк надає можливість поєднувати логіку сервера та клієнта. Для стилізації користувацького інтерфейсу було використано бібліотеку Tailwind CSS [6], що дає змогу писати стилі у класах до тегів JSX, а також Chakra UI [7], що надає можливість використовувати готові UI-компоненти на сторінках.

Серверну частину реалізовано за допомогою API-роутів і Server Actions, написаних мовою програмування TypeScript [8]. Взаємодія з базою даних PostgreSQL [9] відбувається за допомогою ORM Prisma [10], що дозволяє виконувати операції без написання команд до SQL.

Для підвищеного рівня типізації вебзастосунку використано мову програмування TypeScript [8], що дозволяє вчасно виявити й усунути критичні помилки, що підвищує безпеку програмного продукту.

Для впровадження аутентифікації й авторизації у вебзастосунок інтегровано незалежний провайдер Clerk [11], що є повністю керованим сервісом, містить вбудовані UI-компоненти і панель адміністратора, надає можливість увійти в додаток використовуючи сторонні провайдери, надає можливість керування профілем користувача, гарантує захищеність маршрутів, що значно пришвидшує розробку. Даний сервіс працює на основі токенів JWT і зберігає стан аутентифікації на стороні клієнта за допомогою cookies і localStorage. Таким чином Clerk забезпечує безпечну й оновлювану сесію.

Для інтеграції штучного інтелекту у вебзастосунок було використано OpenAI API [12]. Порівнявши різні моделі [13], було обрано деякі з них, що найкраще підходять для реалізації визначеного функціоналу. Для системи рекомендацій каталогу тварин було обрано модель gpt-4, оскільки вона найкраще підходить для виконання комплексних задач і відрізняється високою точністю відповідей. Для аналізу анкети на адопцію і системи підказок по сайту обрано модель gpt-3.5-turbo, тому що дана модель є швидкою, добре підходить для обробки великих обсягів даних і є найдешевшою серед моделей, а отже, є найкращим варіантом коли є необхідність часто робити запити до штучного інтелекту. Для реалізації асистенту чат-боту обрано модель gpt-4-turbo, тому що дана модель зберігає високу точність відповідей, проте є швидшою за gpt-4.

Для зберігання зображень використано хмарний сервіс Cloudinary [14]. Цей сервіс зручний для інтеграції у вебзастосунок і гарантує безпеку зображень, тому що вони зберігаються у сховищі і вимагають секретних ключів для доступу. Даний підхід забезпечує вищу продуктивність додатку і зменшує навантаження на систему.

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Правильний вибір архітектури дозволяє створити гнучкий, надійний і масштабований вебзастосунок, що надає користувачам зручний та інтерактивний вебінтерфейс з широкими функціональними можливостями.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки системи було використано такі технології: JSX як формат розмітки, каскадні таблиці стилів CSS, мова програмування TypeScript [8], а також SSR-фреймворк Next.js [5]. Для стилізації інтерфейсу застосовано бібліотеки TailwindCSS [6] та Chakra UI [7]. На серверній частині також використовується TypeScript [8]. Для роботи з базою даних обрано ORM Prisma [10] та СУБД PostgreSQL [9]. Для зберігання зображень використано хмарний сервіс Cloudinary [14]. Додатково інтегровано OpenAI API [13] для реалізації функціональності штучного інтелекту.

Як сервіс аутентифікації було обрано Clerk [11], який має низку переваг над NextAuth (табл. 1.2). Clerk — це незалежний провайдер аутентифікації, який має власні API, SDK та інфраструктуру для управління користувачами. Він інтегрується безпосередньо з Next.js. Clerk надає можливість зареєструватись та увійти у вебзастосунок використовуючи зручні користувачам методи, наприклад, через обліковий запис Google або Github, а також отримавши код підтвердження на електронну пошту.

Таблиця 1.2

Порівняння сервісів аутентифікації

Функція	Clerk	NextAuth.js
Хостинг	Повністю керований сервіс (SaaS)	Самохостинг, працює у вашому додатку
Управління користувачами	Вбудовані UI-компоненти та адмінпанель	Немає вбудованої панелі
База даних	Clerk самостійно зберігає	Потрібно підключати власну базу

Продовження таблиці 1.2

Соціальні мережі та OAuth	Попередньо налаштовані провайдери (Google, GitHub тощо)	Потрібно налаштовувати вручну
Мультиоренда (multi-tenancy)	Підтримується з коробки	Потрібно реалізовувати самостійно
Управління сесіями	Clerk обробляє автоматично	Потрібно налаштовувати вручну
Налаштовуваний UI	Готові UI-компоненти	Потрібно створювати власний UI

Для визначення бази, яка відповідає потребам системи, було проведено порівняння декількох СУБД. Почнемо з порівняння максимального обсягу даних, які можливо зберегти (табл. 1.3). MySQL добре підходить для середніх і великих проєктів та ефективно працює з реляційними структурами даних. PostgreSQL має дещо нижчий поріг за обсягом даних, але забезпечує кращу підтримку складних запитів і великих наборів даних. MongoDB не має чітко визначених обмежень, однак максимальний обсяг залежить від файлової системи. SQL Server і Oracle мають найвищі ліміти та найкраще підходять для масштабних корпоративних застосунків.

Таблиця 1.3

Порівняння СУБД за обсягом даних

СУБД	Максимальний обсяг даних
MySQL	До 256 ТБ на таблицю (InnoDB)
PostgreSQL	До 32 ТБ на таблицю
MongoDB	Обмежено файловою системою (теоретично – петабайти)
Microsoft SQL Server	До 524 ПБ
Oracle Database	До 8 ЕБ

Проведено аналіз алгоритмів забезпечення безпеки даних в аналогах СУБД (табл. 1.4). PostgreSQL надає широкий набір інструментів безпеки, зокрема підтримку політик на рівні рядків (RLS), однак повне шифрування даних (TDE) наразі офіційно не реалізовано. У MySQL TDE доступне лише у платній Enterprise-версії, а у безкоштовній функціональність безпеки значно обмежена. MongoDB підтримує шифрування даних на диску, але ефективність захисту значною мірою залежить від правильного налаштування. SQL Server і Oracle DB забезпечують найпотужніші засоби захисту, проте є комерційними продуктами.

Таблиця 1.4

Порівняння алгоритмів забезпечення безпеки в СУБД

СУБД	Механізм шифрування	Управління доступом	Аудит та логування	Захист від SQL-ін'єкцій	Реплікація та резервне копіювання
PostgreSQL	AES-128/256, Transparent Data Encryption	Рольова система, Row-Level Security	Вбудовані журнали логування	Підготовлені запити (Prepared Statements)	WAL-реплікація, pg_basebackup
MySQL	AES, TDE (у версії Enterprise)	Полі, GRANT/REVOKE	Вбудований лог аудиту (Enterprise)	Підготовлені запити	binlog-реплікація, mysqldump
MongoDB	AES-256, TDE (Enterprise)	Полі, RBAC	Вбудовані журнали операцій	ORM-бібліотеки або фреймворки	Oplog-реплікація, snapshot backup
Microsoft SQL Server	AES-256, TDE, Always Encrypted	Розширене RBAC, маскування даних (Data Masking)	Вбудований аудит	Захист на рівні API	Log Shipping, Mirroring, Always On

Продовження таблиці 1.4

Oracle DB	AES, TDE, Data Redaction	Гнучка система ролей	Потужний аудит	Вбудовані механізми захисту	RMAN, Data Guard, Flashback
-----------	--------------------------------	-------------------------	-------------------	-----------------------------------	-----------------------------------

Також проведено детальне порівняння аналогів за архітектурними, функціональними та експлуатаційними характеристиками (табл. 1.5). PostgreSQL є найбільш гнучкою та функціонально багатою серед відкритих СУБД, що добре підходить для вебзастосунків. MySQL вирізняється високою швидкістю роботи з простими додатками, але має обмеження при обробці складних запитів. MongoDB є найкращим вибором для NoSQL-проектів, де потрібне гнучке зберігання даних і можливість горизонтального масштабування. Microsoft SQL Server орієнтований на корпоративні рішення, добре інтегрується з платформою .NET, проте є платним. Oracle DB забезпечує найвищий рівень безпеки та потужності, тому підходить для фінансових установ, банківських систем і критично важливих задач.

Таблиця 1.5

Порівняння СУБД за архітектурними, функціональними та експлуатаційними характеристиками

Критерій	PostgreSQL	MySQL	MongoDB	Microsoft SQL Server	Oracle DB
Логічна модель даних	Реляційна (ACID)	Реляційна (ACID)	Документна (NoSQL)	Реляційна (ACID)	Реляційна (ACID)
Платформи	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux (обмежено)	Windows, Linux
Область застосування	Веб, аналітика, OLTP/OLAP	Веб, мобільні додатки, OLTP	Big Data, аналітика, NoSQL	Корпоративні додатки, фінанси, OLTP/OLAP	Корпоративні системи, банки, OLTP/OLAP
Особливості	JSONB, CTE, MVCC, RLS	Простота, швидкість, JSON підтримка	Гнучкість, горизонтальне масштабування	Always Encrypted, Data Masking	Потужна безпека, Data Guard, Flashback

Продовження таблиці 1.5

Мови маніпулювання	SQL (PL/pgSQL), JSONB	SQL, JSON	Mongo Query Language (MQL)	SQL (T-SQL)	SQL (PL/SQL)
Шифрування	AES-128/256 (на рівні розширень)	AES-256 (Enterprise TDE)	AES-256 (на рівні диска)	TDE, Always Encrypted	TDE, Data Redaction
Розширюваність	Підтримка розширень та кастомних типів	Мінімальна	Вбудовані плагіни	Розширення через .NET	PL/SQL розширення
Масштабування	Вертикальне + шардинг	Вертикальне	Горизонтальне (sharding)	Вертикальне (Always On)	Вертикальне + кластеризація
Ліцензія	Open Source (PostgreSQL License)	Open Source	Open Source (SSPL, але з обмеженнями)	Пропрієтарна (Microsoft)	Пропрієтарна (Oracle)

Для визначення ORM, що найбільше відповідає вимогам системи, проведено порівняння деяких з них (табл. 1.6). Prisma є найзручнішою ORM з відмінною підтримкою типізації, що робить її популярною серед розробників TypeScript. Водночас через високий рівень абстракції вона може незначно впливати на продуктивність. Drizzle є легшою та продуктивнішою альтернативою, ближчою до звичайного SQL, і підходить для тих, хто потребує максимального контролю над запитамі. TypeORM часто обирають для об'єктно-орієнтованих застосунків, проте вона має певні проблеми з типізацією. Knex.js не є повноцінною ORM, а радше конструктором запитів, що забезпечує велику гнучкість, але вимагає більше ручного кодування.

Таблиця 1.6

Порівняння ORM

ORM	Типізація	Міграції	Продуктивність	Зручність у Next.js	Гнучкість
Prisma	Сильна (TypeScript)	Вбудовані	Добра, але не найшвидша	Чудова інтеграція	Деколи менш гнучка через абстракції
Drizzle	Сильна (TypeScript, SQL-like API)	Вбудовані	Висока (SQL-запити ближчі до "чистого" SQL)	Легка інтеграція	Дуже гнучка
TypeORM	Середня (не завжди точна)	Вбудовані	Середня	Добре працює	Досить гнучка
Knex.js	Відсутня (потребує додаткових типів)	Вимагає ручного налаштування	Висока	Потребує додаткової інтеграції	Дуже гнучка

Після детального аналізу різних СУБД і ORM було вирішено використовувати PostgreSQL разом із Prisma для реалізації проєкту. Це рішення обумовлене низкою переваг. PostgreSQL забезпечує гнучкість, безпеку та потужність, зокрема підтримку складних запитів, реляційних зв'язків, що є важливим для масштабних систем. Водночас Prisma дозволяє значно спростити роботу з базою завдяки автоматичній генерації типізованих моделей, зручному управлінню міграціями без необхідності писати SQL вручну, а також глибокій інтеграції з TypeScript, що підвищує безпеку та зручність коду. Окрім того, використання Prisma спрощує розгортання та супровід: Prisma Studio надає зручний інтерфейс для взаємодії з базою, а сама ORM добре інтегрується з такими

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

фреймворками, як Next.js [15], а також підтримує популярні хмарні платформи, наприклад, Vercel.

Отже, використання PostgreSQL як системи управління базами даних забезпечує високу продуктивність, гнучкість та надійність зберігання даних, тоді як Prisma спрощує процес розробки завдяки зручному API, автоматичній генерації типізованих запитів та інтеграції з сучасними фреймворками.

При розробці використовувалися інтегровані середовища та програмні інструменти: Visual Studio Code і JetBrains WebStorm.

Висновки до першого розділу

У цьому розділі було сформульовано завдання випускної кваліфікаційної роботи бакалавра та визначено функціональні вимоги до системи. Проведено аналіз ринку та аналогічних рішень, що дозволило виокремити основні функціональні можливості вебзастосунку: зручний користувацький інтерфейс, система реєстрації та аутентифікації користувачів, обробка заявок на адопцію, фільтрація каталогу і отримання рекомендацій, підтримка чат-боту і системи підказок на сторінках, сповіщення користувачів через електронну пошту. Також було обґрунтовано та підтверджено вибір архітектури системи. Для розробки клієнтської частини використано Next.js, TypeScript, TailwindCSS та Chakra UI. Серверна частина реалізована на Next.js (App Router) із використанням ORM Prisma для роботи з базою даних PostgreSQL. Для аутентифікації користувачів інтегровано Clerk. Для зберігання зображень використано хмарне сховище Cloudinary. Для реалізації підтримки штучного інтелекту було використано OpenAI API.

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ АДОПЦІЇ ДОМАШНІХ ТВАРИН

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

У цьому розділі проведено детальний огляд варіантів використання системи, а також вимог, які повинен виконувати програмний продукт для забезпечення виконання ним основних функцій.

Система складається з двох основних модулів: клієнтської, що надає інтерфейс для користування системою, та адміністративної, що надає можливість керувати системою.

Система передбачає наявність різних користувачів, кожен з яких має свою роль, відповідно й певні права використання системи. Ці ролі включають в себе неавторизованого користувача, авторизованого користувача, а також адміністратора. Неавторизований користувач має гостьовий доступ до вебзастосунку. Він має можливість переглядати тварин, доступних для адопції, проте не може надіслати заявку на адопцію. Авторизований користувач має можливість надіслати заявку на адопцію, заповнивши перед цим відповідну форму. Адміністратор має доступ до адміністративної панелі, де він може керувати вмістом БД і обробляти заявки користувачів на адопцію.

Проєкт створюється для того, щоб спростити процес адопції домашніх тварин. Основна мета полягає у тому, щоб зробити його зручним і зрозумілим як для користувачів, так і для притулків, а також популяризувати відповідальне ставлення до тварин. Реалізація відбувається у вигляді вебсайту, де буде зібрана інформація про тварин, яких можна всиновити, реалізовано систему фільтрації за категоріями та за рекомендаціями, а також інші необхідні функції.

Для майбутніх власників тварин важливо мати можливість переглядати каталог доступних тварин, шукати тварин за категоріями, бачити повну інформацію про кожну тварину включно з фото і описом, подавати заявки на адопцію, отримувати сповіщення про результат обробки надісланої заявки.

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

Адміністратори повинні мати змогу керувати всією базою даних, архівувати анкети й модерувати контент.

Інтерфейс розробляється з урахуванням зручності, щоб навіть користувачі без технічного досвіду могли швидко зорієнтуватися і почати користуватися сайтом.

Функціонально система повинна дозволяти реєстрацію та вхід для користувачів з різними ролями (звичайний користувач і адміністратор), а також зберігати та обробляти інформацію про користувачів, тварин і подані заявки.

Нефункціонально система має бути простою в освоєнні, для новачка час освоєння повинен бути до 30 хвилин, а для досвідченого користувача не більше 5 хвилин. Час простою не повинен перевищувати 10% від загального часу, середній безперервний період роботи повинен тривати 20 робочих днів, повинно бути не більше 1 помилки на 1000 запитів. Система повинна витримувати щонайменше 100 одночасних користувачів без втрати якості роботи та бути здатною до масштабування.

Платформа має коректно працювати у браузері Chrome та операційній системі Windows 10.

На рисунку 2.1 зображено діаграму варіантів використання системи вебзастосунку адопції домашніх тварин.

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				25
Змн.	Арк.	№ докум.	Підпис	Дата		



Рисунок 2.1. Діаграма використання вебзастосунку адопції домашніх тварин

2.2 Розробка бази даних системи

В створюваному вебзастосунку буде використана база даних PostgreSQL [16]. Такі бази даних широко використовуються для зберігання даних у сучасних вебзастосунках.

Реалізована база даних складається з 6 таблиць. Найменування та призначення кожної з них представлено у таблиці 2.1.

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

Інформація про таблиці бази даних

Назва таблиці	Призначення
Pet	Таблиця містить тварин
Category	Таблиця містить категорії тварин
Post	Таблиця містить пости блогу
Adoption	Таблиця містить запити на адопцію
Message	Таблиця містить повідомлення чатів
Chat	Таблиця містить чати

На рисунку 2.2 представлена діаграма бази даних.

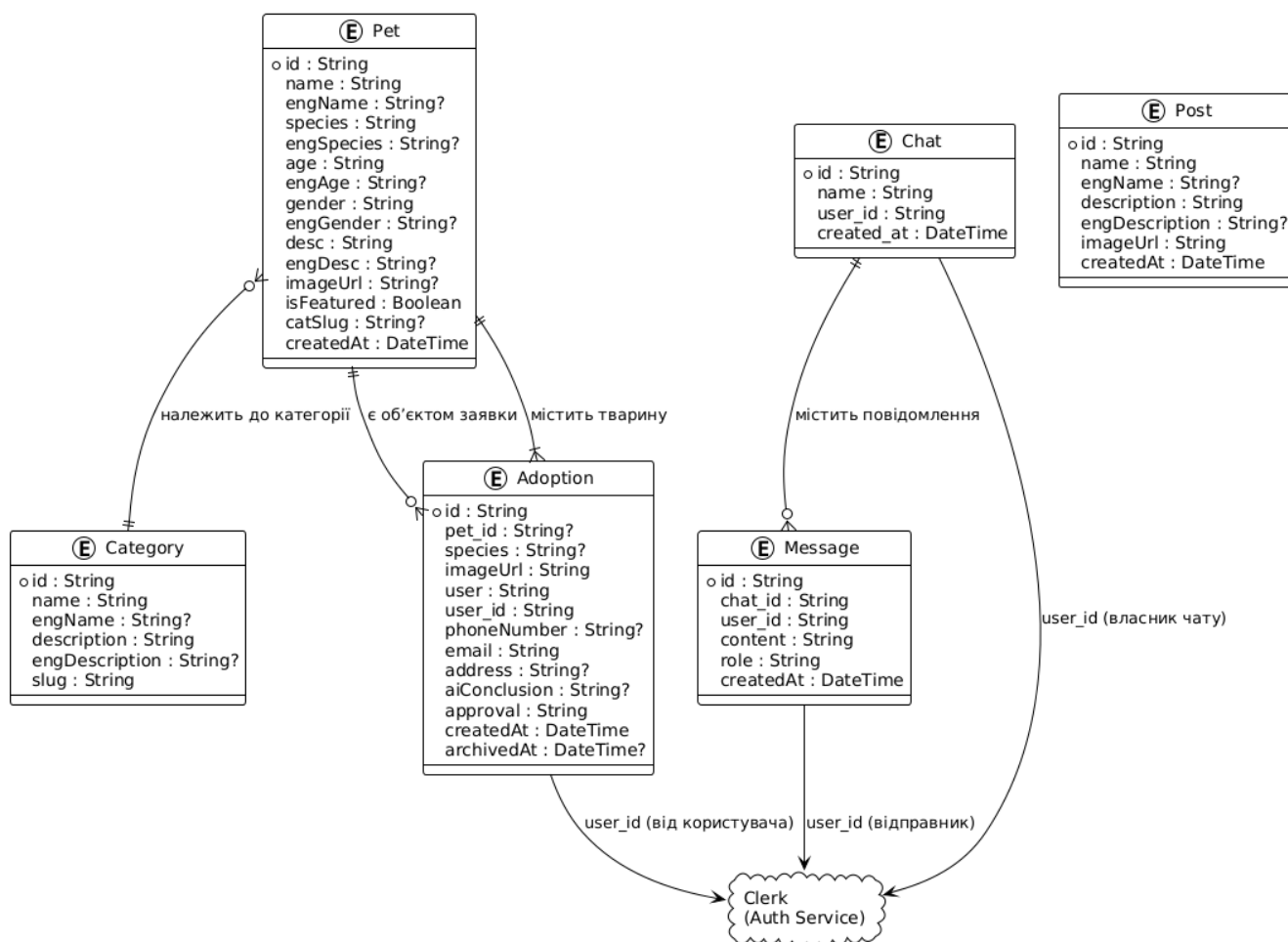


Рисунок 2.2. Діаграма БД

Далі наведемо опис таблиць бази даних (таблиці 2.2-2.7).

Таблиця 2.2

Опис полів таблиці “Pet”

Назва	Тип даних	PK	FK	Опис поля
id	String	+	-	Ідентифікатор
ukName	String	-	-	Ім'я українською
engName	String	-	-	Ім'я англійською
ukSpecies	String	-	-	Вид українською
engSpecies	String	-	-	Вид англійською
ukAge	String	-	-	Вік українською
engAge	String	-	-	Вік англійською
ukGender	String	-	-	Стать українською
engGender	String	-	-	Стать англійською
ukDesc	String	-	-	Опис українською
engDesc	String	-	-	Опис англійською
imageUrl	String	-	-	Шлях до зображення
isFeatured	Boolean	-	-	Чи є рекомендованим
category	Category	-	-	Категорія
catSlug	String	-	-	Слаг категорії
adoptions	Adoption[]	-	-	Адопції
createdAt	DateTime	-	-	Дата і час створення

Таблиця 2.3

Опис полів таблиці “Category”

Назва	Тип даних	PK	FK	Опис поля
id	String	+	-	Ідентифікатор
ukName	String	-	-	Назва українською
engName	String	-	-	Назва англійською
ukDescription	String	-	-	Опис українською
engDescription	String	-	-	Опис англійською
pets	Pet[]	-	-	Тварини
slug	String	-	+	Слаг

Таблиця 2.4

Опис полів таблиці “Post”

Назва	Тип даних	PK	FK	Опис поля
id	String	+	-	Ідентифікатор
ukName	String	-	-	Заголовок українською
engName	String	-	-	Заголовок англійською
ukDescription	String	-	-	Контент українською
engDescription	String	-	-	Контент англійською
imageUrl	String	-	-	Шлях до зображення
createdAt	DateTime	-	-	Дата і час створення

Таблиця 2.5

Опис полів таблиці “Adoption”

Назва	Тип даних	PK	FK	Опис поля
id	String	+	-	Ідентифікатор
pet_id	String	-	-	ID тварини
pet	Pet	-	-	Тварина
species	String	-	-	Вид
imageUrl	String	-	-	Зображення
user	String	-	-	Користувач
user_id	String	-	-	Ідентифікатор користувача
phoneNumber	String	-	-	Номер телефону
email	String	-	-	Електронна пошта
address	String	-	-	Адреса
aiConclusion	String	-	-	Висновок ІІІ
approval	String	-	-	Схвалення адміністратором
createdAt	DateTime	-	-	Дата і час створення
archivedAt	DateTime	-	-	Дата архівування

Таблиця 2.6

Опис полів таблиці “Message”

Назва	Тип даних	PK	FK	Опис поля
id	String	+	-	Ідентифікатор
chat_id	String	-	+	ID чату
chat	Chat	-	-	Чат
user_id	String	-	-	ID користувача
content	String	-	-	Повідомлення
role	String	-	-	Роль
createdAt	DateTime	-	-	Дата і час створення

Таблиця 2.7

Опис полів таблиці “Chat”

Назва	Тип даних	PK	FK	Опис поля
id	String	+	-	Ідентифікатор
name	String	-	-	Назва чату
user_id	String	-	-	ID користувача
messages	Message[]	-	-	Повідомлення чату
created_at	DateTime	-	-	Дата і час створення

2.3 Проєктування та реалізація алгоритмів роботи системи

В даному розділі буде побудована діаграма діяльності системи, що є важливим інструментом на різних етапах розробки ПЗ. Діаграми діяльності можуть використовуватись для моделювання бізнес-процесів, де вони допомагають визначити послідовність подій та вимог. Діаграма діяльності має кілька важливих переваг. Вона дозволяє наочно відобразити логіку роботи системи, що значно полегшує розуміння процесів усіма учасниками проєкту. Такий підхід допомагає виявити можливі неефективності в діях, сприяє їх оптимізації та подальшій автоматизації. Діаграма також може слугувати основою для створення технічного завдання або формалізації алгоритмів. Вона зручна для аналізу поведінки

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

користувачів у системі, добре поєднується з іншими типами UML-діаграм і є корисною як для бізнес-аналітиків, так і для розробників.

Процес подання заявки на адопцію тварини починається з того, що користувач відкриває сайт і переглядає каталог тварин. Він може скористатися фільтрами для зручного пошуку та обрати тварину, яка його зацікавила.

Неавторизований та авторизований користувач має змогу обмежити каталог тварин за категоріями, а також отримати рекомендації на основі штучного інтелекту. Після заповнення форми штучний інтелект надає список тварин, що найкраще відповідають стилю життя майбутнього власника. Також користувач може обмежити рекомендації за конкретними категоріями.

Після того, як користувач обрав найкращий для себе варіант, він повинен впевнитись, що авторизувався в системі, інакше кнопка подачі заявки буде заблокованою. Після успішної авторизації користувач натискає кнопку «Подати заявку». Користувач переходить на сторінку з анкетою, яка містить перелік запитань різного характеру. Анкета містить загальні питання, питання щодо стилю життя потенційного власника, питання психологічного характеру. Користувач заповнює та надсилає дану анкету. Відповідна діаграма діяльності зображена на рисунку 2.3.

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

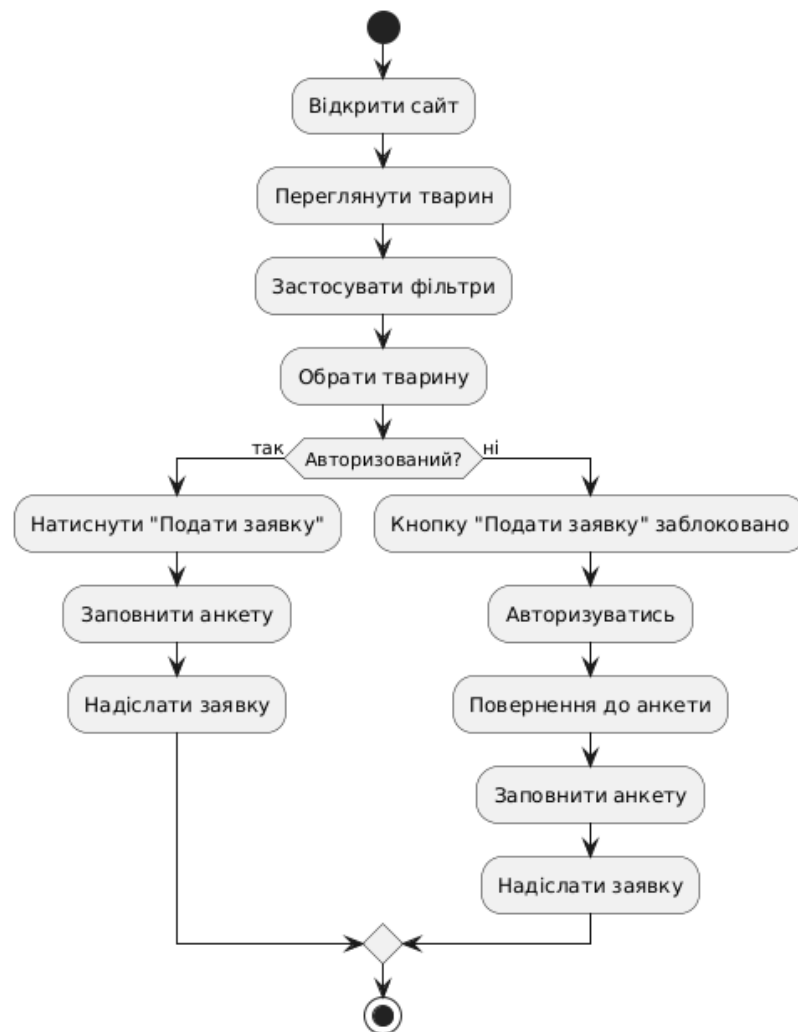


Рисунок 2.3. Діаграма діяльності подання заявки на адопцію

Процес обробки заявки адміністратором передбачає, що адміністратор переглядає список нових запитів в адмінпанелі і обирає конкретну заявку для розгляду. Система відображає всі деталі анкети, такі як фото, ім'я та вид тварини, на яку було подано заявку. Також адміністратор може переглянути висновок, сформований на основі аналізу штучного інтелекту, який містить пояснення, чи можна надавати даного улюбленця конкретному власнику.

Штучний інтелект порівнює реальні потреби зазначеного улюбленця із можливостями кандидата на адопцію. Потреби включають в себе витрати на утримання, включно із харчуванням, послугами ветеринара і витратами на встановлення необхідного обладнання. Також враховується кількість вільного часу, оскільки деякі тварини потребують значно більше часу на догляд і фізичну активність, ніж інші. Крім того, важливим аспектом є сімейний стан кандидата,

тому що існують тварини, які не приймають інших членів родини, крім господаря або можуть бути агресивними і становити небезпеку. Додатково враховується і безпека самої тварини. Важливим аспектом є й тип житла, тому що деякі тварини потребують більше місця, ніж інші. До того ж, власне житло й житло в оренді становлять значну різницю у результатах.

Адміністратор критично оцінює отримані дані, після чого ухвалює рішення щодо доцільності адопції. У разі позитивного рішення статус заявки оновлюється на “Схвалено”, а користувач отримує відповідне повідомлення на електронну пошту. Якщо рішення негативне, користувач також інформується про відмову. Після цього заявка архівується, і адміністратор завершує роботу з нею. Діаграму діяльності для обробки запиту адміністратором наведено на рисунку 2.4.

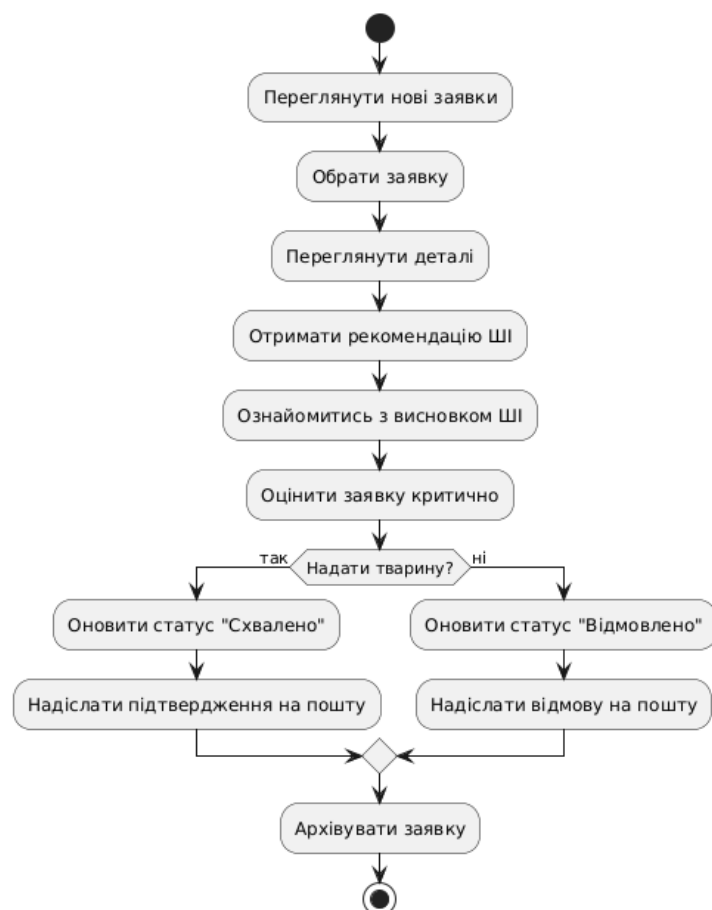


Рисунок 2.4. Діаграма діяльності для обробки запиту адміністратором

Для неавторизованого та авторизованого користувача передбачено можливість застосування фільтра. Процес отримання рекомендованих тварин у

каталозі передбачає заповнення користувачем форми даними, що можуть суттєво впливати на вибір улюбленця, такими як тип житла, щомісячний дохід, кількість вільного часу, наявність дітей та досвід утримання тварин. Також користувач може обмежити пошук конкретними категоріями.

Після цього ШІ аналізує дані з форми і повертає тварин, що найкраще відповідають способу життя майбутнього власника. Штучний інтелект бере до уваги реальні можливості користувача щодо утримання і порівнює їх зі списком доступних тварин. Для невеликого бюджету рекомендуються тварини із незначними витратами на утримання, включно із доглядом, купівлею необхідного обладнання і послугами ветеринара. Для маленького й орендованого житла рекомендуються невеликі й неактивні тварини, або ж тварини, що живуть у клітках, акваріумах і терраріумах. Також враховується час, який власник може витратити на улюбленця, що є особливо важливим для більшості порід собак. Крім того, важливими аспектами у виборі улюбленця є наявність дітей та їх вік, а також досвід утримання тварин. При відсутності досвіду і наявності дітей штучний інтелект буде надавати перевагу улюбленцям, що є менш небезпечними і простішими в утриманні.

При умові, що користувач додатково обрав категорії, тварини будуть рекомендовані лише за вказаними категоріями, а якщо обрано лише категорії і форму не було заповнено, то ШІ не буде надавати рекомендацій. Рекомендовані тварини будуть повернені із нумерацією від найкращого до найгіршого варіанту. Діаграма діяльності для отримання рекомендованих тварин наведено на рисунку 2.5.

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

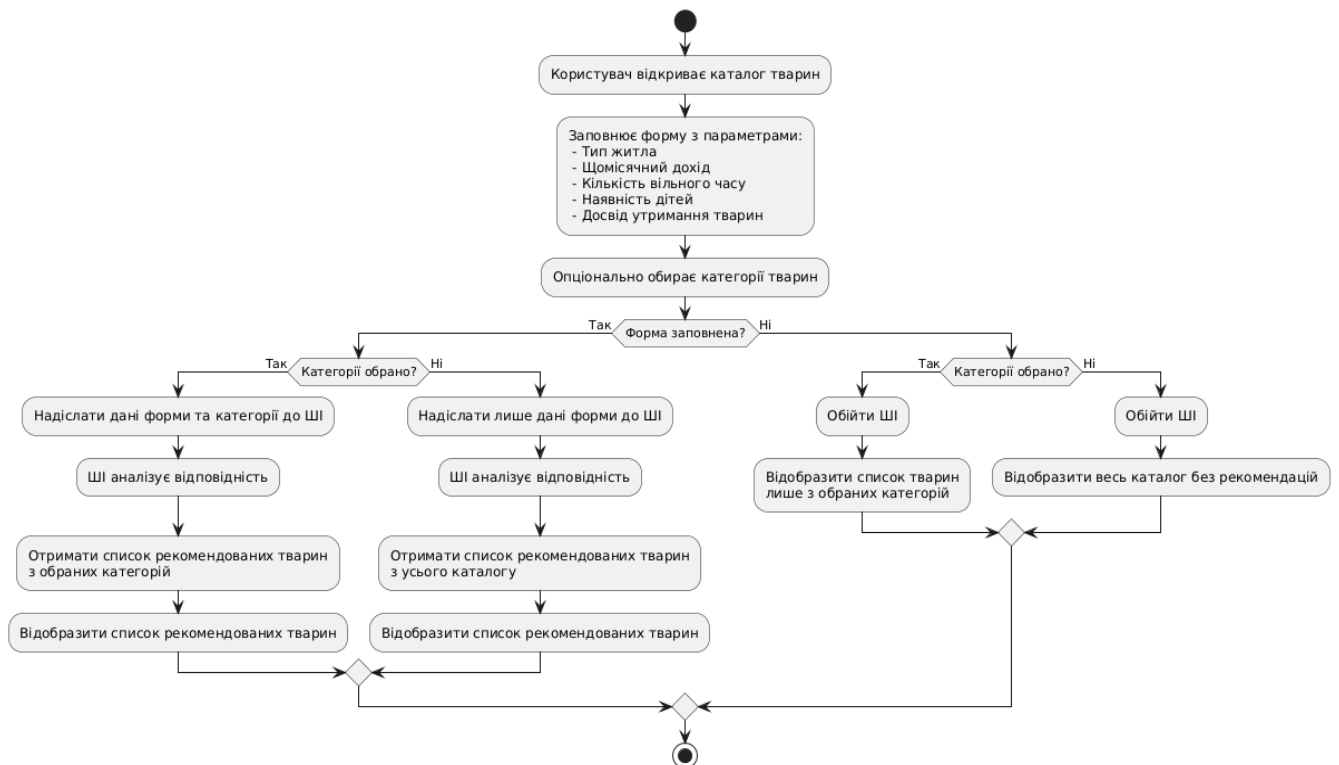


Рисунок 2.5. Діаграма діяльності для отримання рекомендованих тварин

Система підказок на сторінках вебзастосунку працює на основі поточного шляху URL та ролі користувача, оновлення повідомлення відбувається при перезавантаженні сторінки, переходу по навігації, зміні ролі користувача або мови вебзастосунку.

Штучний інтелект надає рекомендації відповідно до ролі користувача. Для неавторизованих і авторизованих користувачів надаються підказки і рекомендації щодо використання даної сторінки. Подібний функціонал може бути корисним для користувачів без досвіду використання застосунків адопції. Для адміністратора надаються рекомендації щодо покращення певних частин сторінки та функціоналу.

Створена система підказок і рекомендацій підвищує інтерактивність сторінки і покращує взаємодію між користувачем і центром адопції. Відповідна діаграма діяльності наведена на рисунку 2.6.

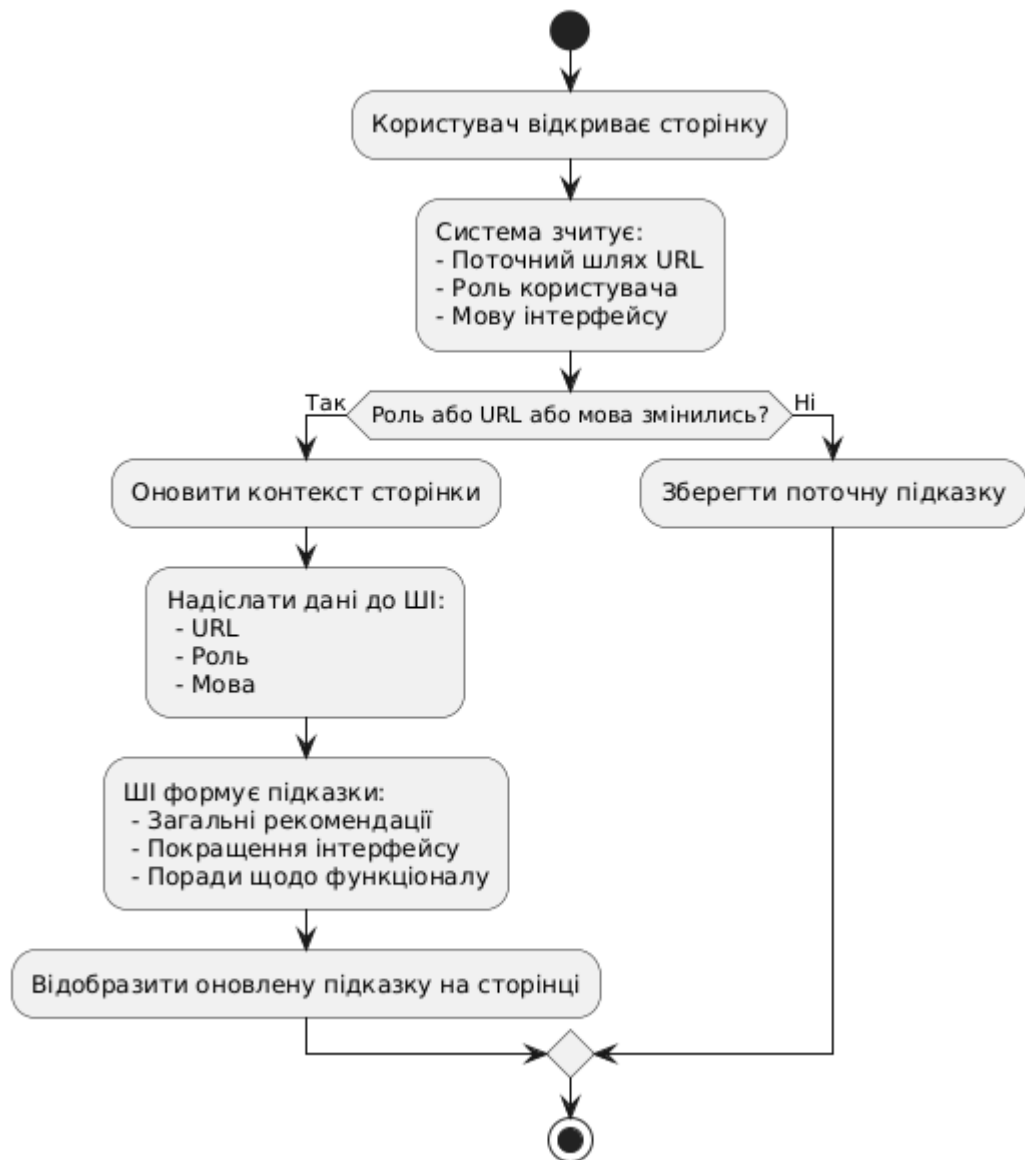


Рисунок 2.6. Діаграма діяльності до системи підказок

2.4 Реалізація вебзастосунку адопції домашніх тварин

У цьому розділі розглянуто діаграму компонентів вебзастосунку для адопції тварин, яка демонструє взаємозв'язки між основними структурними елементами системи, зокрема між модулями інтерфейсу, серверною логікою, базою даних та сторонніми сервісами. Кожен функціональний блок застосунку, такий як система аутентифікації, модуль керування анкетами, модуль блогу чи чат, представлено як окремий компонент. Варто зазначити, що діаграма компонентів зосереджується виключно на структурних аспектах системи.

Система складається з кількох основних компонентів. PostgreSQL виступає як база даних, у якій зберігається інформація про тварин, заявки на адопцію,

повідомлення, категорії, чати та інші сутності. PetLayout є контекстом, який обгортає каталог тварин і фільтр за категоріями, відповідаючи за відображення та структурування інформації. Компонент AIHint генерує підказки для користувачів, враховуючи поточний маршрут, роль користувача та мову інтерфейсу. Компонент Chat реалізує чат-бот, який дозволяє поставити питання до асистента. LanguageSwitcher дає змогу змінювати мову інтерфейсу та локалізувати контент. Операції всередині застосунку поділено на логічні блоки. До блоку Pet Management входять функції для додавання, редагування, перегляду та видалення тварин. Adoption Process охоплює процес подання заявки, її обробку за допомогою штучного інтелекту, зміну статусу та архівацію. Модуль Blog Module забезпечує перегляд статей, новин і історій адопції. Chat Module відповідає за взаємодію між користувачем і ШІ, а також зберігання історії повідомлень. Filters реалізує функціональність фільтрації тварин за категоріями та на основі рекомендацій. Hint System забезпечує показ динамічних підказок, які змінюються відповідно до поведінки користувача на сайті. На рисунку 2.7 наведено діаграму компонентів системи.

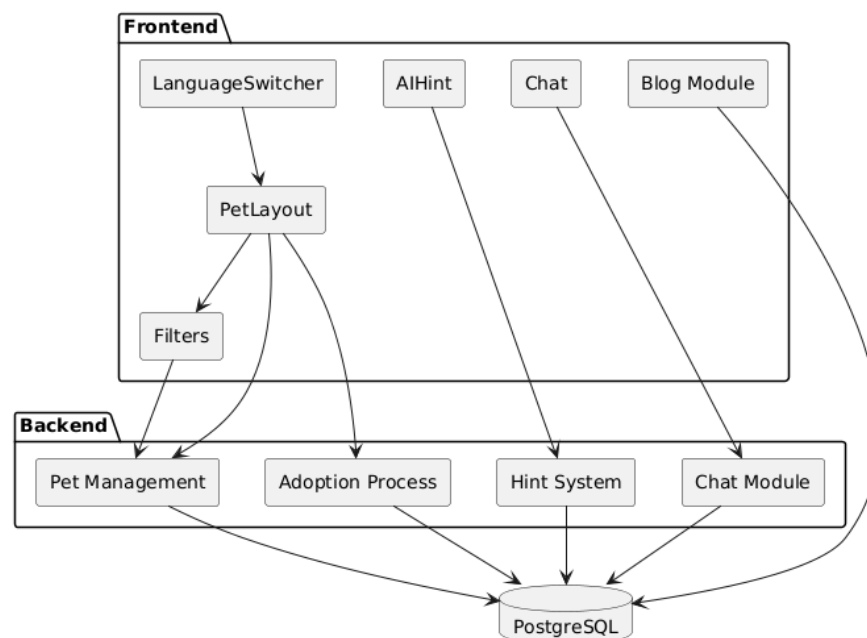


Рисунок 2.7. Діаграма компонентів системи

Важливим етапом розробки вебзастосунку є інтеграція аутентифікації та авторизації. Нижче наведено алгоритми впровадження цих процесів за допомогою стороннього сервісу Clerk [17].

Алгоритм аутентифікації за допомогою Clerk починається з того, що користувач вводить свої облікові дані на сторінці входу й отримує код підтвердження на пошту. Після цього Clerk приймає введені облікові дані та перевіряє їх правильність. Якщо облікові дані вірні, система видає токен аутентифікації, який браузер користувача зберігає у вигляді cookie. При кожному подальшому запиті до вебзастосунку цей токен автоматично надсилається до Clerk. Clerk перевіряє дійсність токена й надає доступ до відповідного ресурсу, якщо перевірка пройдена успішно.

Алгоритм OpenID-аутентифікації через Google або GitHub починається з натискання користувачем кнопки входу через відповідний сервіс. Після цього його перенаправляють на сторінку входу Google або GitHub, де користувач вводить свої облікові дані. Сторонній сервіс перевіряє ці дані й у разі успішної верифікації надсилає вебзастосунку токен аутентифікації. Цей токен використовується для створення нового облікового запису користувача або входу в наявний профіль.

Clerk надає готові компоненти, що виконують вищезазначені процеси, приклад використання даних компонентів наведено нижче.

```
"use client";
import { SignedIn, SignedOut, SignInButton, UserButton } from "@clerk/nextjs";
import { Box, Button } from "@chakra-ui/react";
import { useLocale } from 'next-intl';

const AuthButtons = () => {
  const locale = useLocale();
  return (
    <Box>
      <SignedOut>
        <SignInButton
          mode="modal"
          key={`signin-${locale}`}
        >

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савицький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

```

        <Button colorScheme="teal" size="sm">
          {locale === 'uk' ? "Увійти" : "Sign in"}
        </Button>
      </SignInButton>
    </SignedOut>
    <SignedIn>
      <UserButton />
    </SignedIn>
  </Box>
);
};

export default AuthButtons;

```

Алгоритм перевірки ролі користувача передбачає передачу сесії до функції `checkUserRole`, де визначається, чи є користувач учасником будь-якої з організацій Clerk. В залежності від результату перевірки функція повертає відповідну роль, яку можна використовувати для визначення прав доступу, наприклад, на сторінках, де потрібні права адміністратора. Лістинг функції `checkUserRole` наведено нижче.

```

const checkUserRole = (session: any) => {
  if (!session || !session.user) {
    return "guest";
  }

  if (
    (session &&
      session.user) &&
    (!session.user.organizationMemberships ||
      session.user.organizationMemberships.length === 0)
  ) {
    return "user";
  }

  const organizationMemberships = session.user.organizationMemberships;

  for (const membership of organizationMemberships) {
    if (membership.role) {

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return membership.role.toLowerCase();
    }
}

return null;
}

export { checkUserRole };

```

Для функціонування вебзастосунку найважливішим етапом є підключення до бази даних. Для цього було використано ORM Prisma, що є технологією зв'язування БД з програмним кодом. Тому розглянемо файл створення екземпляру Prisma-клієнта. Головною метою цього файлу є уникнення створення нових екземплярів під час розробки. Нижче наведено лістинг цього файлу.

```

import { PrismaClient } from "@prisma/client";

const globalForPrisma = globalThis as unknown as {
  prisma: PrismaClient | undefined
}

export const prisma = globalForPrisma.prisma ?? new PrismaClient()

if (process.env.NODE_ENV !== 'production') globalForPrisma.prisma = prisma

```

Для взаємодії адміністратора і користувача зроблено інтеграцію з електронною поштою, тому розглянемо функцію sendEmail для реалізації email-сповіщень. В даній функції було використано бібліотеку nodemailer. Користувач отримує повідомлення з рішенням адміністратора щодо надісланої заявки на адопцію. Наведемо лістинг цієї функції.

```

export async function sendEmail(email: string, decision: string) {
  const t = getTranslations("email");

  const transporter = nodemailer.createTransport({
    host: 'smtp.gmail.com',
    port: 587,

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

```

    secure: false,
    auth: {
      user: process.env.GMAIL_USERNAME,
      pass: process.env.GMAIL_PASSWORD,
    },
  });

  try {
    await transporter.sendMail({
      from: process.env.GMAIL_USERNAME,
      to: email,
      subject: (await t)("header"),
      text: decision === "approve" ? (await t)("approve") : (await t)("deny"),
      replyTo: email,
    });
    return { success: true, message: 'Email sent successfully!' };
  } catch (error) {
    console.error(error);
    return { success: false, message: 'Failed to send email.' };
  }
}

```

На основі рішення адміністратора щодо конкретної заявки на адопцію відбувається зміна деяких полів запису, а саме approval та archivedAt, що являють собою схвалення або відмову щодо запиту і дату архівування відповідно, після чого заявка зникає з панелі адміністратора. Наведемо лістинг API запиту.

```

import {NextRequest, NextResponse} from "next/server";
import {prisma} from "@/utils/connect";

type Params = {
  params: {
    id: string;
  }
}

export const PATCH = async (req: NextRequest, { params }: Params) => {
  const id = params.id;

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

```

try {
  const body = await req.json();
  const adoptionRequest = await prisma.adoption.update({
    where: {id},
    data: {
      approval: body,
      archivedAt: new Date()
    }
  });
  return new NextResponse(JSON.stringify(adoptionRequest), { status: 201 });
} catch (err) {
  return new NextResponse(JSON.stringify({ message: "Something went wrong!" }),
{ status: 500 });
}
}

```

Далі наведемо пояснення контролю статусу адопції. Після того, як користувач надіслав заявку на адопцію, кнопка подачі має бути заблокованою. При завантаженні компонентів карток улюбленців виконується код, наведений нижче.

```

const [isLoading, setIsLoading] = useState(true);
const [hasPendingRequest, setHasPendingRequest] = useState(false);

useEffect(() => {
  const checkRequest = async () => {
    if (user && isLoaded) {
      const res = await
fetch(`/api/adoption/check?userId=${user.id}&petId=${pet_id}`);
      const data = await res.json();
      setHasPendingRequest(data.exists);
      setIsLoading(false);
    }
  };
  checkRequest();
}, [user, isLoaded, pet_id]);

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

У вищезазначеному коді відбувається запит на API route і передається ідентифікатор тварини і користувача, після чого перевіряється таблиця із запитами і повертається значення статусу.

```
import { NextRequest, NextResponse } from "next/server";
import { prisma } from "@utils/connect";

export const GET = async (req: NextRequest) => {
  const { searchParams } = new URL(req.url);
  const userId = searchParams.get("userId");
  const petId = searchParams.get("petId");

  if (!userId || !petId) {
    return new NextResponse(JSON.stringify({ message: "Missing parameters" })), {
      status: 400 };
  }

  const existing = await prisma.adoption.findFirst({
    where: {
      user_id: userId,
      pet_id: petId,
      approval: "inProcess"
    }
  });

  return NextResponse.json({ exists: !!existing });
};
```

Зважаючи на те, що кожна тварина відноситься до конкретної категорії, було доцільним створити запасну категорію на випадок видалення будь-якої з категорій і оновити категорію для відповідних улюбленців. Це рішення було прийняте для того, щоб уникнути помилок у роботі вебзастосунку і кожна тварина була прикріплена до певної категорії. Коли адміністратор видаляє якусь категорію відбувається перевірка резервної категорії, якщо її ще немає, то створює її.

```
const categoryToDelete = await tx.category.findUnique({
  where: { id }
```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

```

});

if (!categoryToDelete) {
    throw new Error("Category not found");
}

if (categoryToDelete.slug === fallbackSlug) {
    throw new Error("Cannot delete fallback category");
}

let fallbackCategory = await tx.category.findUnique({
    where: { slug: fallbackSlug }
});

if (!fallbackCategory) {
    fallbackCategory = await tx.category.create({
        data: {
            name: "Не визначено",
            slug: fallbackSlug,
            description: "Запасний варіант категорії для тварин",
            engName: "Not Defined",
            engDescription: "Default category for unassigned pets",
        },
    });
}

```

Потім тварини, що були прив'язані до видаленої категорії, автоматично переносяться до запасної, потім небажана категорія видаляється.

```

await tx.pet.updateMany({
    where: { catSlug: categoryToDelete.slug },
    data: { catSlug: fallbackSlug },
});

await tx.category.delete({
    where: { id },
});

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

Ці кроки виконуються у транзакції, що дозволяє надійно і послідовно виконувати операції. Такий підхід дозволяє уникнути ситуацій, коли тварини залишаються без категорій, що може спричинити помилки і збій у роботі вебзастосунку.

Висновки до другого розділу

У процесі написання другого розділу було розроблено діаграми варіантів використання, діяльності та компонентів. Досліджено основні сценарії взаємодії користувачів із системою та визначено ключовий функціонал, який має бути реалізований у вебзастосунку для забезпечення ефективного процесу адопції тварин.

Також було спроектовано структуру бази даних, яка використовується для зберігання інформації про тварин, заявки на адопцію, повідомлення та інші важливі дані. Побудовано діаграму бази даних і надано опис кожної таблиці та їхніх взаємозв'язків.

На завершення, було реалізовано усі спроектовані функції, внаслідок чого створено повноцінний функціональний вебзастосунок. Представлено набір методів і механізмів, що забезпечують виконання ключових задач системи та підтримують зручну взаємодію користувачів із платформою.

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБЗАСТОСУНКОМ
АДОПЦІЇ ДОМАШНІХ ТВАРИН

3.1 Розгортання вебзастосунку

Під час розробки вебзастосунку його компоненти розгортаються локально, а для глобального користування вебзастосунок було розгорнуто на хмарному сервісі Vercel. Для повноцінного деплою бази даних PostgreSQL використано Neon, що добре інтегрується з Vercel. Як сховище зображень було використано хмарний сервіс Cloudinary. Для підтримки штучного інтелекту було використано OpenAI API, а для аутентифікації та авторизації – сервіс Clerk. Діаграму розгортання вебзастосунку зображено на рисунку 3.1.

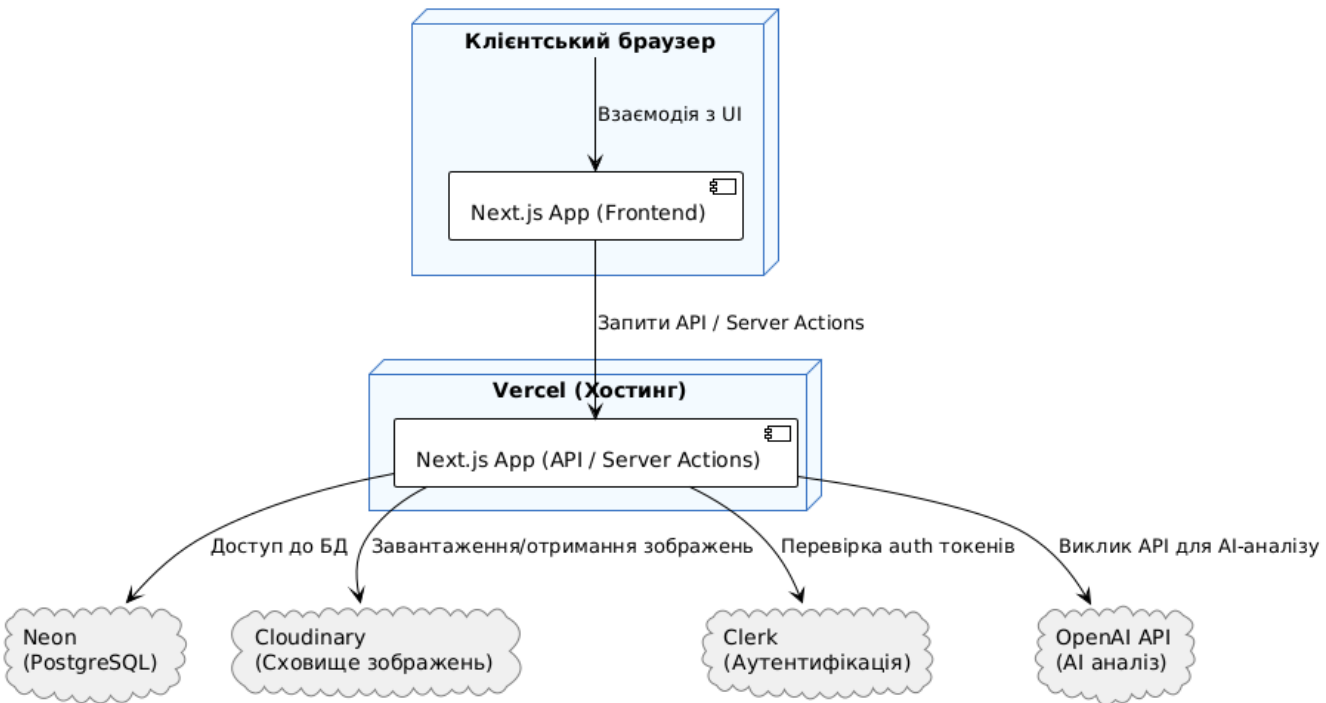


Рисунок 3.1. Діаграма розгортання вебзастосунку

Ця діаграма показує структуру вебзастосунку та описує його роботу. Користувач бачить інтерфейс, зроблений за допомогою Next.js. Щоб увійти або зареєструватись в системі він проходить процес аутентифікації через окремий хмарний сервіс Clerk. Після виконання певних дій дані йдуть на серверну частину вебзастосунку, де використовується ORM Prisma для виконання операцій з БД. Якщо адміністратор хоче створити новий запис у БД, то файли зображень зберігаються у хмарному середовищі Cloudinary, після чого він зможе бачити ці

фото в панелі адміністратора, а користувач – на доступних йому сторінках. У деяких випадках вебзастосунок звертається до OpenAI, наприклад, для аналізу анкети на адопцію або надання рекомендацій для користувачів. Процеси, описані вище, працюють як єдина захищена система, що зберігає свої дані в різних регіонах світу.

3.2 Структура інтерфейсу вебзастосунку адопції домашніх тварин

Коли неавторизований користувач заходить на сайт він бачить головну сторінку з невеликою галереєю з тваринами, а також панель навігації, що містить посилання на сторінки, перемикач мови та кнопку для аутентифікації (рис. 3.2).

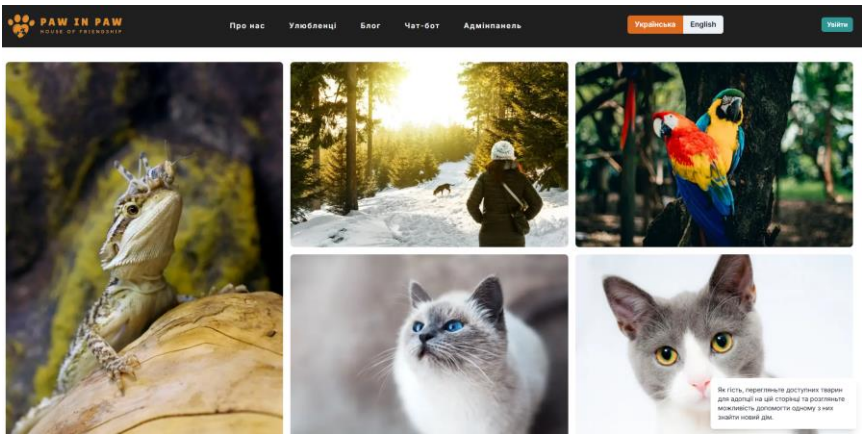


Рисунок 3.2. Головна сторінка вебзастосунку

Сторінка з улюбленцями є найважливішою для вебзастосунку адопції, оскільки вона містить каталог тварин, доступних для адопції (рис. 3.3).

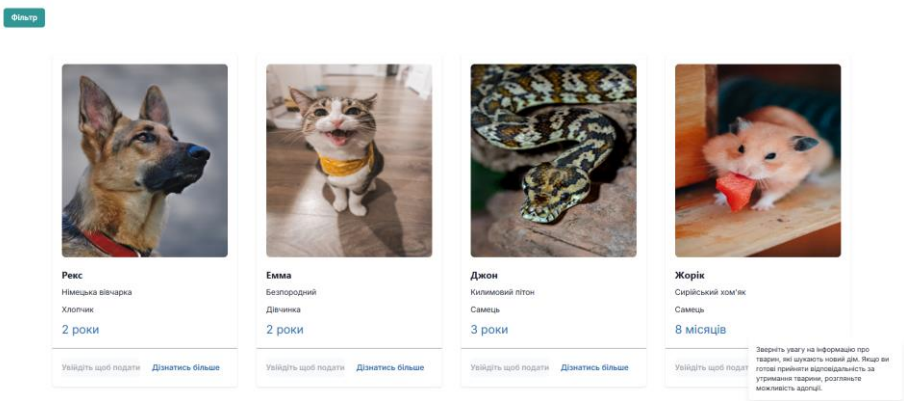


Рисунок 3.3. Сторінка з каталогом тварин

Неавторизований користувач не має змоги надіслати заявку на адопцію, тому відповідна кнопка на картці тварини є заблокованою, проте він може побачити коротку інформацію про тварину на картці (рис. 3.4), а також більш детальну

інформацію, якщо перейти за посиланням “Дізнатись більше” або натиснувши на ім’я тварини (рис. 3.5).



Мідорі

Зелена ігуана

Самка

2 роки

Увійдіть щоб подати

Дізнатись більше

Рисунок 3.4. Картка тварини



Рекс

Вид: Німецька вівчарка

Вік: 2 роки

Стать: Хлопчик

Німецька вівчарка Рекс служила у прикордонних військах, де допомагала в затриманні порушників і пошуку заборонених речовин. Відзначалась слухняністю, витривалістю та винятковим нюхом.

Увійдіть щоб подати
Назад

Рисунок 3.5. Сторінка тварини

Також для неавторизованих користувачів є можливість застосувати фільтр для каталогу (рис. 3.6). Можна відфільтрувати тварин за категоріями або скористатись штучним інтелектом для отримання рекомендацій на основі даних з форми. Якщо було обрано лише категорії, то відбудеться запит до БД і користувач побачить улюбленців за відповідними категоріями (рис. 3.7). Якщо крім категорій користувач заповнив поля форми, то відбудеться запит до штучного інтелекту, на основі якого користувач дізнається, які улюбленці з обраних категорій найкраще відповідають стилю життя майбутнього власника (рис. 3.8). При умові, що користувач заповнив лише форму, рекомендації будуть надаватись з усіх доступних тварин (рис. 3.9).

Знайдіть ідеального улюбленця

Категорії

Оберіть категорії

▼

Розкажіть про себе для отримання рекомендацій

Ваш щомісячний дохід

Оберіть діапазон доходу

▼

Ваше житлове становище

Оберіть тип житла

▼

Ваш вільний час на тиждень

Оберіть кількість вільного часу

▼

Ваш досвід з тваринами

Оберіть рівень досвіду

▼

Чи маєте дітей?

Оберіть один варіант

▼

Очистити

Відмінити

Застосувати фільтри

Рисунок 3.6. Фільтр каталогу тварин

Фільтр

Гризуни

Птахи



Жорік

Сирійський хом'як

Самець

8 місяців



Артур

Папуга Ара

Самець

10 років

Рисунок 3.7. Фільтрація за категоріями

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

Для того, щоб процес адопції став доступним, необхідно авторизуватися. Щоб увійти в додаток використовуючи сервіс Clerk, потрібно натиснути кнопку “Увійти” і побачити модальне вікно для реєстрації або входу (рис. 3.10).

Рисунок 3.10. Модальне вікно Clerk

Користувач має змогу авторизуватись отримавши код підтвердження на електронну пошту (рис. 3.11), а також за допомогою обраних адміністратором провайдерів, таких як GitHub (рис. 3.12) або Google (рис. 3.13).

Рисунок 3.11. Авторизація через код

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

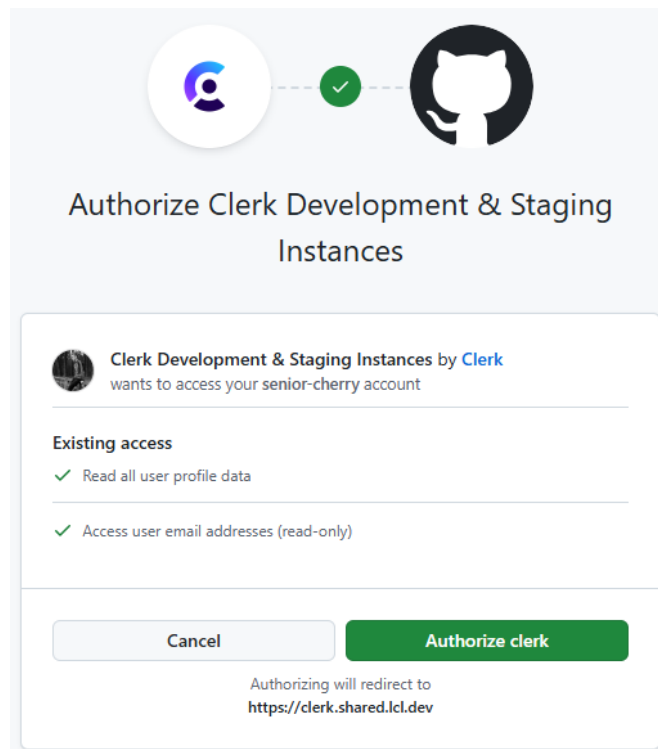


Рисунок 3.12. Авторизація через GitHub

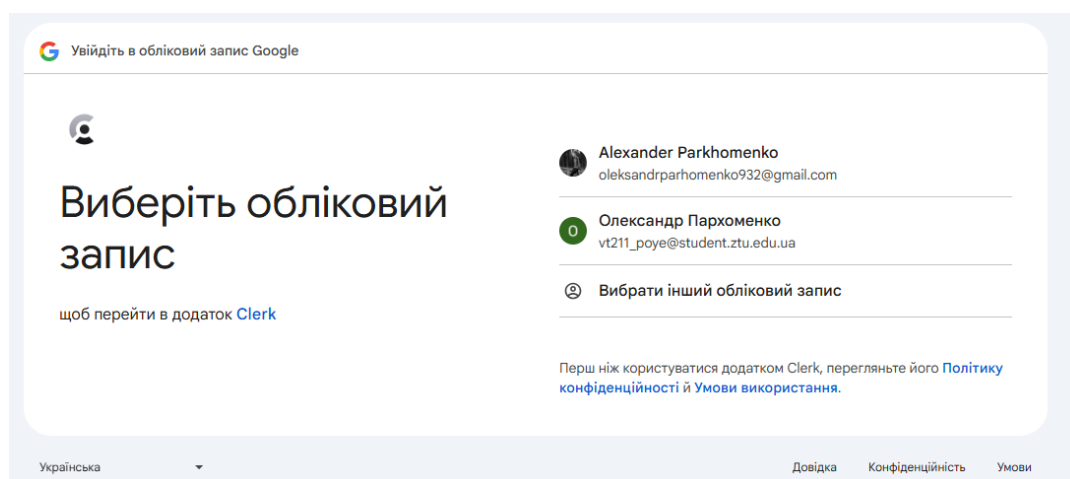


Рисунок 3.13. Авторизація через Google

Після того, як користувач авторизувався, він отримує доступ до процесу адопції. Щоб перейти до анкети на адопцію необхідно натиснути на відповідну кнопку на картці улюбленця (рис. 3.14). Після чого потенційний власник переходить до анкети, що містить питання різного характеру, аналіз відповідей на які допоможе зробити висновок щодо придатності користувача для адопції даної тварини (рис. 3.15).

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савицький Р.С.				53
Змн.	Арк.	№ докум.	Підпис	Дата		



Джон

Килимовий пітон

Самець

3 роки

Взяти тварину

Дізнатись більше

Рисунок 3.14. Картка тварини з можливістю адопції

Вік кандидата

20

Контактний номер телефону

Контактний номер телефону

Електронна пошта

vt211_poye@student.ztu.edu.ua

Адреса проживання

Адреса проживання

Тип житла

Оберіть...

Чи дозволено утримання тварин у вашому житлі?

Оберіть...

Рисунок 3.15. Анкета адопції

		Пархоменко О.Є.		
		Савіцький Р.С.		
Змн.	Арк.	№ докум.	Підпис	Дата

ІПЗ.КР.Б – 121 – 25 – ПЗ

Арк.

54

Якщо форма пройшла валідацію і запит було успішно надіслано, на картці тварини буде видно повідомлення, що заявка вже на розгляді в адміністратора (рис. 3.16).

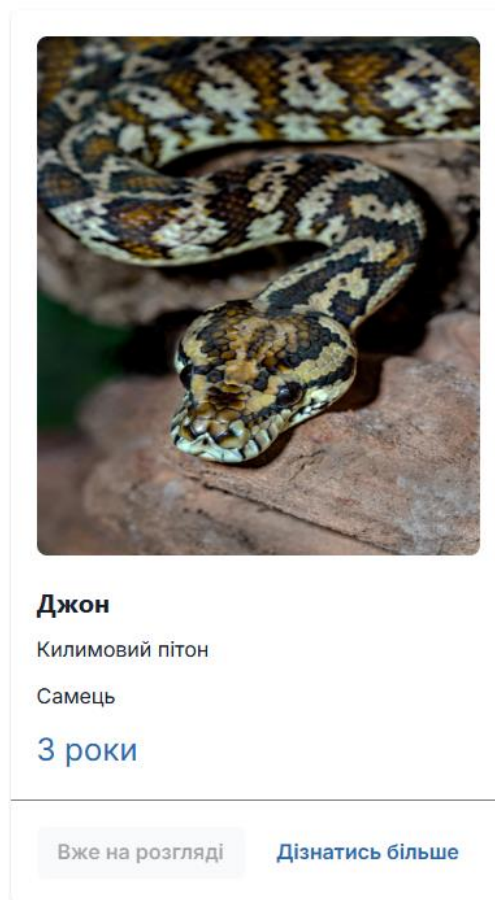


Рисунок. 3.16. Картка тварини на момент розгляду заявки

Крім того, незалежно від ролі користувача, у вебзастосунку працює система підказок на основі штучного інтелекту, що приймає шлях сторінки з рядка адреси сайту і роль користувача, і надає деякі рекомендації щодо використання або покращення поточної сторінки. На рисунку 3.17 наведено приклад повідомлення для авторизованого користувача на сторінці з улюбленцями.

Ознайомтесь з інформацією про тварин, які потребують домівки. Якщо ви готові прийняти відповідальність за утримання тварини, розгляньте можливість адопції.

Рисунок 3.17. Підказка на сторінці з улюбленцями

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

Авторизовані користувачі також мають підтримку чат-боту, що надає послуги ШІ-асистенту, можна створювати і видаляти окремі чати, що прив'язані до поточного облікового запису. Приклад роботи чат-боту наведено на рисунку 3.18.

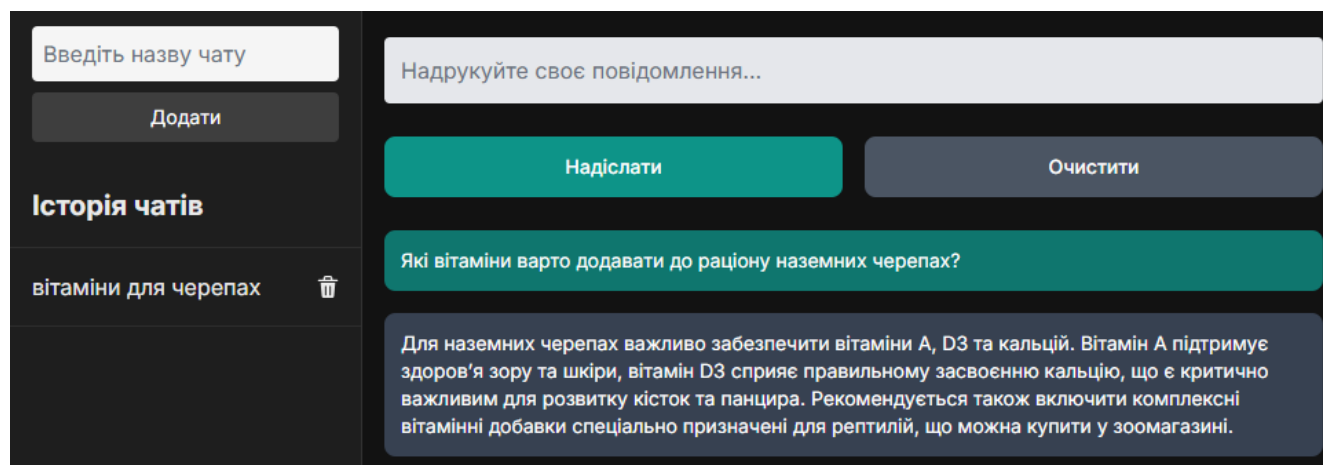


Рисунок 3.18. Сторінка чат-боту

Для користувача з правами адміністратора є доступ до спеціальної панелі, де він має змогу керувати вмістом БД, а також керувати заявками на адопцію, а саме прийняти або відхилити після аналізу результатів отриманих від штучного інтелекту (рис. 3.19).

Домашні улюбленці						▼
Категорії						▼
Статті						▼
Заявки на адопцію						▲
ФОТО	ВИД	КОРИСТУВАЧ	ЕЛЕКТРОННА ПОШТА	ВИСНОВОК ШІ	ДІЇ	
	Килимовий пітон	Олександр Пархоменко	vt211_poye@student.ztu.edu.ua	Висновок ШІ	Прийняти	Відхилити
Дані заявок на адопцію				Рекомендується. Кандидат підготовлений і має досвід у догляді за домашніми тваринами. Він виявляє високу емоційну стабільність, готовий вкласти час та ресурси у догляд за твариною, та позитивно налаштований до виховання. Враховуючи всі ці аспекти, кандидат може бути відповідальним власником Килимового пітона.		

Рисунок 3.19. Дані заявок на адопцію в адмінпанелі

Після прийняття або відхилення заявки на адопції користувач отримує відповідне сповіщення на електронну пошту (рис. 3.20).



oleksandrparhomenko932@gmail.com

кому мені ▼

Вітаємо! Вашу заявку на адопцію тварини схвалено. Наша команда незабаром зв'яжеться з вами для подальших дій.

← Відповісти

→ Переслати

Рисунок 3.20. Сповіщення про результати адопції.

3.3 Тестування роботи програмного продукту

Роботу вебзастосунку адопції домашніх тварин було протестовано різними способами, що дозволило переконатися у його стабільності та надійності, коректності виконання запитів на API та обробки помилок, правильній взаємодії між клієнтською та серверною стороною. Запити на сервер було протестовано за допомогою інструменту для створення й тестування API Postman. Це дозволило надіслати запити до окремих ендпоінтів, перевірити правильність роботи створених методів для роботи з БД, а також переглядати відповіді і статус запитів у зручному для сприйняття форматі. На рисунку 3.21 зображено приклад виконання запиту на отримання списку зі статей блогу.

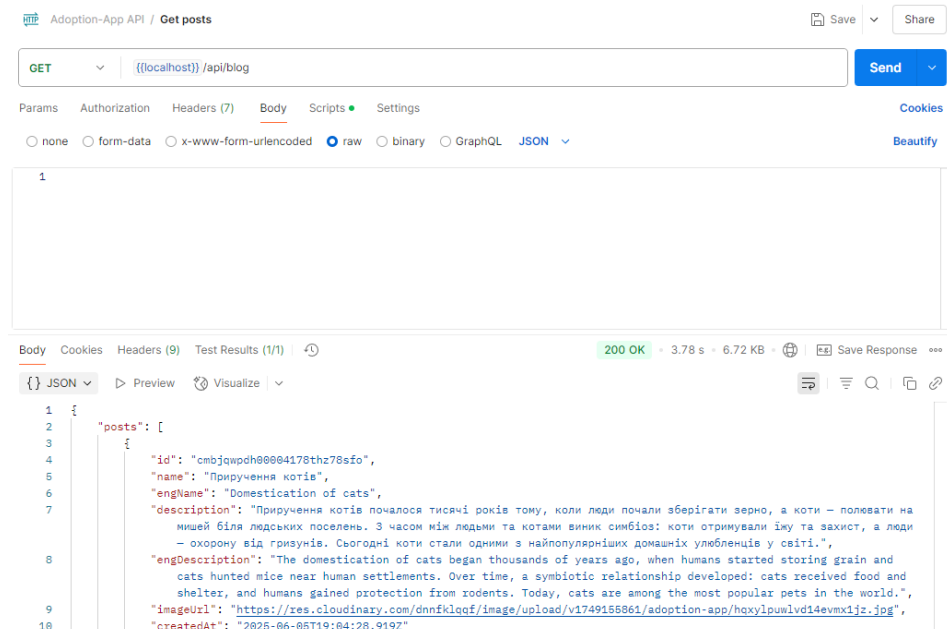


Рисунок 3.21. Тестування запиту на отримання статей блогу

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

Також за допомогою Postman було перевірено роботу пагінації, разом із параметрами в рядку адреси (рис. 3.22).

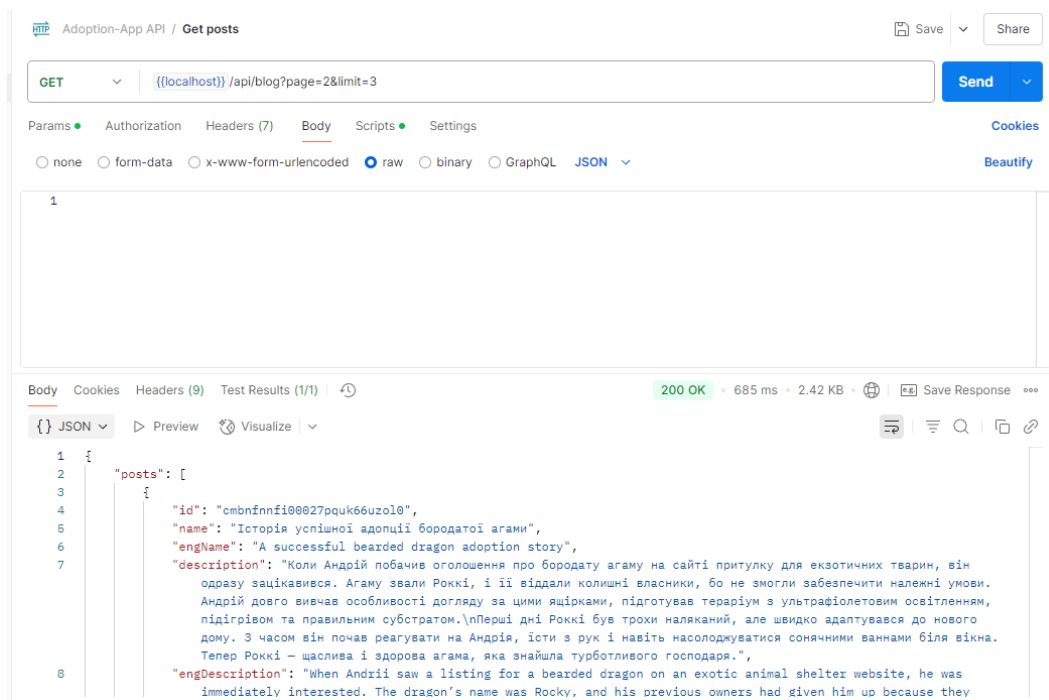


Рисунок 3.22. Тестування пагінації із параметрами

Для перевірки клієнтської частини вебзастосунку було використано мануальне тестування. Для форми адопції було додано валідацію електронної пошти і номеру телефону, а також перевірку заповнення обов’язкових полів з варіантом відповіді. Для перевірки електронної пошти і номеру телефону було використано регулярні вирази, що перевіряють введені дані за певними синтаксичними правилами, а для валідації заповнення полів робиться перевірка наявності порожніх значень (рис. 3.23).

```

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
if (!emailRegex.test(inputs.email)) {
  toast( options: { title: tf( key: "error"), description: tf( key: "email"), status: 'error' });
  return;
}

const phoneRegex = /^\+380\d{3}\d{2}\d{2}\d{2}$/;
if (!phoneRegex.test(inputs.phoneNumber)) {
  toast( options: { title: tf( key: "error"), description: tf( key: "phone"), status: 'error' });
  return;
}

const invalidSelects = requiredSelects.filter((field :string ) => inputs[field as keyof Inputs] === "");
if (invalidSelects.length > 0) {
  toast( options: { title: tf( key: "error"), description: tf( key: "fields"), status: 'error' });
  return;
}
  
```

Рисунок 3.23. Валідація анкети на адопцію

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

Для відображення помилки використовується тост, приклад якого можна побачити на рисунку 3.24.



Рисунок 3.24. Тост з відображенням помилки валідації

Елемент тост був також використаний для відображення результатів виконання запитів до API (рис. 3.25).

```
if (res.status === 200) {  
  toast( options: {  
    title: t( key: "toastSuccessTitle"),  
    description: t( key: "toastSuccessDescription"),  
    status: 'success',  
    duration: 3000,  
    isClosable: true,  
  });  
  setIsDeleted( value: true);  
} else {  
  const data = await res.json();  
  toast( options: {  
    title: t( key: "toastErrorTitle"),  
    description: data.message,  
    status: 'error',  
    duration: 5000,  
    isClosable: true,  
  });  
}
```

Рисунок 3.25. Відображення результату API-запиту за допомогою тосту

Крім вищезазначених способів тестування програми була здійснена перевірка сайту на продуктивність за допомогою інструменту Lighthouse. Перевірка сторінки з улюбленицями зображена на рисунку 3.26, а сторінки чат-боту на рисунку 3.27.

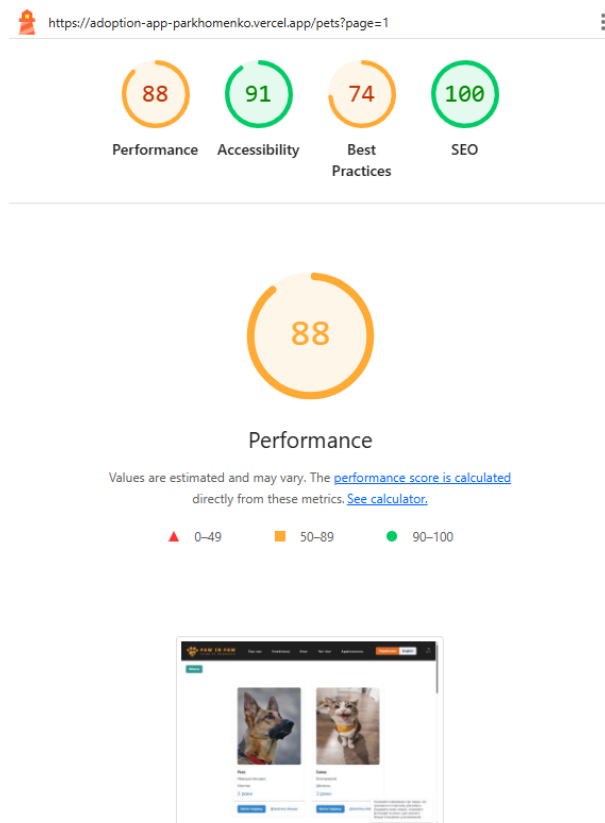


Рисунок 3.26. Перевірка продуктивності сторінки з улюбленцями

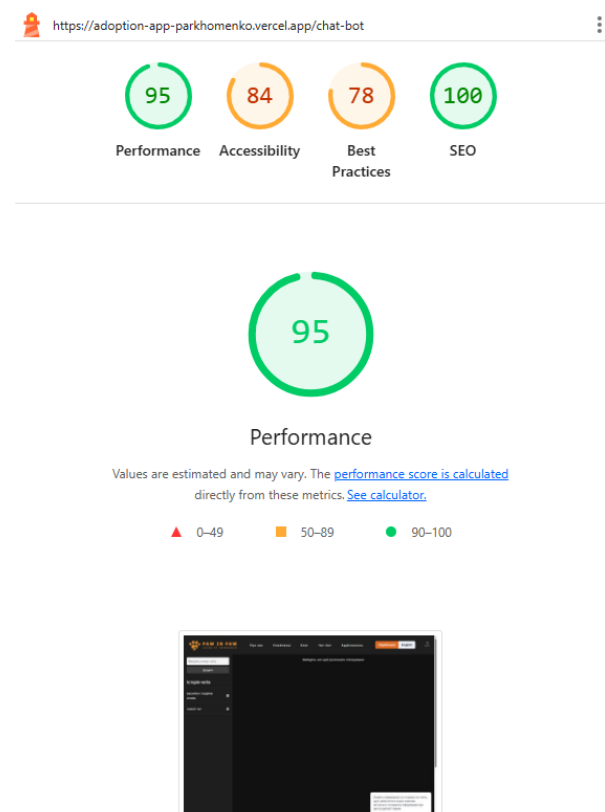


Рисунок 3.27. Перевірка продуктивності сторінки чат-боту

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до третього розділу

В даному розділі був описаний повний цикл роботи вебзастосунку адопції домашніх тварин, включно з розгортанням і тестуванням функціоналу. Було спроектовано діаграму розгортання даної системи, де було зображено її компоненти, а також взаємодію між ними.

Було представлено структуру інтерфейсу вебзастосунку, що є зручною та інтуїтивно зрозумілою для використання різними користувачами. Для кожної з можливих ролей продемонстровано доступні сторінки та можливості використання функціоналу системи.

Також було приділено увагу тестуванню вебзастосунку, яке дозволило вчасно виявити й усунути помилки та переконатися у працездатності системи. Було проведено тестування API-запитів за допомогою інструменту Postman, мануальне тестування клієнтської частини і перевірка продуктивності сторінок вебзастосунку за допомогою інструменту LightHouse.

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

ВИСНОВКИ

В результаті написання випускної кваліфікаційної роботи бакалавра було виконано основні етапи створення вебзастосунку для автоматизації процесу адопції і покращення взаємодії між майбутніми власниками і центром адопції. У процесі розробки ПЗ було використано ефективні й сучасні інструменти та рішення. Кожен розділ випускної кваліфікаційної роботи бакалавра мав чітко визначену мету й завдання, в результаті чого було спроектовано повноцінний вебзастосунок із визначеною функціональністю та інтуїтивно зрозумілим інтерфейсом.

У першому розділі було проведено аналіз аналогів вебзастосунку для дослідження ринку, що дало змогу визначити вимоги до функціональних можливостей майбутньої системи. Основною перевагою стала інтеграція штучного інтелекту для створення системи рекомендацій, аналізу заявок на адопцію, створення чат-боту та системи підказок. Обгрунтовано вибір клієнт-серверної архітектури і технологій для розробки, проведено порівняння різних СУБД і ORM. Розроблено архітектурну концепцію проєкту, визначено його модульну організацію, а також засоби впровадження залежностей та взаємодії між компонентами.

У другому розділі було проведено архітектурне й функціональне проєктування вебзастосунку. Було розроблено діаграми варіантів використання системи для різних ролей, а саме для неавторизованого й авторизованого користувачів, а також для адміністратора. Також було створено діаграму бази даних для зображення таблиць й взаємозв'язків між ними. Сформовано діаграми активності для демонстрації основних алгоритмів роботи програми, таких як подача заявки на адопцію, обробка заявки адміністратором і отримання рекомендованих тварин.

У третьому розділі представлено реалізацію програмного продукту. Вебзастосунок розгорнуто на хмарному сервісі Vercel, що гарантувало стабільність і масштабованість системи. Було успішно налагоджено взаємодію між клієнтською

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

і серверною частинами, базою даних, сервісом для аутентифікації і авторизації, API штучного інтелекту й хмарним сховищем для зберігання зображень. Створено привабливий й інтуїтивно зрозумілий інтерфейс, що надає можливість використання функціоналу системи. Реалізовано можливість фільтрації за категоріями й отримання рекомендацій на основі штучного інтелекту, реєстрації або входу у систему, подачі заявки на адопцію, використання чат-боту і перегляду освітніх матеріалів. Проведено тестування API-запитів за допомогою інструменту Postman, клієнтської частини за допомогою засобів мануального тестування, продуктивності за допомогою інструменту LightHouse. Це дозволило виявити та усунути критичні помилки на етапі розробки, перевірити роботу основного функціоналу та відповідність вимогам.

В результаті випускної кваліфікаційної роботи бакалавра було реалізовано технічно досконалий програмний продукт з максимальною орієнтацією на користувача. Виконана робота демонструє системний підхід до створення вебзастосунку адопції домашніх тварин, що забезпечує зручність використання відповідно до початкових вимог.

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Пархоменко О.Є., Савіцький Р.С. Використання штучного інтелекту у Next.js. Тези XV Міжнародної науково-технічної конференції «Інформаційно-комп'ютерні технології», 28-29 березня 2025 р. Житомир: Державний університет “Житомирська політехніка”, 2025. С. 27-28. [Електронний ресурс] – Режим доступу до ресурсу: <https://conf.ztu.edu.ua/wp-content/uploads/2025/04/povnyj-tekst.pdf>
2. Офіційний сайт Happy Paw [Електронний ресурс] – Режим доступу до ресурсу: <https://happypaw.ua/>
3. Офіційний сайт ЛКП “Лев” [Електронний ресурс] – Режим доступу до ресурсу: <https://lkplev.com/>
4. Офіційний сайт GladPet [Електронний ресурс] – Режим доступу до ресурсу: <https://gladpet.org/>
5. Офіційна документація Next.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs>
6. Офіційна документація Tailwind CSS [Електронний ресурс] – Режим доступу до ресурсу: <https://v2.tailwindcss.com/docs>
7. Офіційна документація Chakra UI [Електронний ресурс] – Режим доступу до ресурсу: <https://chakra-ui.com/docs/get-started/frameworks/next-app>
8. Офіційна документація до мови програмування TypeScript [Електронний ресурс] – Режим доступу до ресурсу: <https://www.typescriptlang.org/docs/>
9. Офіційна документація PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/>
10. Офіційна документація ORM Prisma [Електронний ресурс] – Режим доступу до ресурсу: <https://www.prisma.io/docs>
11. Офіційна документація Clerk [Електронний ресурс] – Режим доступу до ресурсу: <https://clerk.com/docs>
12. Офіційна документація OpenAI API [Електронний ресурс] – Режим доступу до ресурсу: <https://platform.openai.com/docs/overview>

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				64
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Порівняння моделей OpenAI [Електронний ресурс] – Режим доступу до ресурсу: <https://platform.openai.com/docs/models/compare>
14. Офіційна документація Cloudinary [Електронний ресурс] – Режим доступу до ресурсу: <https://cloudinary.com/documentation>
15. PostgreSQL with Next.js and Prisma [Електронний ресурс] – Режим доступу до ресурсу: <https://medium.com/@itsuki.enjoy/postgresql-with-next-js-and-prisma-44f66a05378a>
16. Which Database Should You Use for Your Next.js App? [Електронний ресурс] – Режим доступу до ресурсу: <https://javascript.plainenglish.io/which-database-should-you-use-for-your-next-js-app-f667e63677ce>
17. Clerk Use Case: Sample Code [Електронний ресурс] – Режим доступу до ресурсу:
<https://medium.com/@linglijunmail/clerk-use-case-sample-code-4cfc8169a5cb>

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

Лістинг коду з файлу adoption/create/page.tsx:

```
"use client";
import React, {useEffect, useState} from "react";
import {useSearchParams, useRouter} from "next/navigation";
import {useUser} from "@clerk/nextjs";
import {useTranslations} from "next-intl";
import {useToast} from "@chakra-ui/react";
import {Inputs} from "@/types/types";

const requiredSelects = [
  'age',
  'phoneNumber',
  'email',
  'address',
  'aptType',
  'petsAllowed',
  'kids',
  'otherAnimals',
  'liveWith',
  'employmentStatus',
  'incomeLevel',
  'animalsBefore',
  'nutritionalNeeds',
  'reasonToAdoptPet',
  'characterPreferences',
  'lifeChangingActions',
  'psychologicalDiseases',
  'tendencyToBeImpulsive',
  'emotionalStability',
  'reactionToStress',
  'uncontrolledAnger',
  'blamesForAnger',
  'attitudeToSpending',
  'attitudeToBadBehaviour',
  'requireLove',
  'isAbleToFollowSchedule',
  'friendlySupport',
  'allergyToFur',
  'howToRaisePet',
  'reactionToBadAction',
  'isPunishmentNecessary',
];

const AdoptionForm = () => {
  const toast = useToast();
  const q = useTranslations("adoptionFormQuestions");
  const o = useTranslations("adoptionFormOptions");

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const tf = useTranslations("adoption-form-toast");
const router = useRouter();
const {isLoading, user} = useUser();
const searchParams = useSearchParams();
const pet_id = searchParams.get("id") || "";
const imageUrl = searchParams.get("imageUrl") || "";
const species = searchParams.get("species") || "";

const loadSavedFormData = () => {
  try {
    const savedData = localStorage.getItem('adoptionFormData');
    if (savedData) {
      return JSON.parse(savedData);
    }
  } catch (error) {
    console.error('Error loading saved form data:', error);
  }
  return null;
};

const saveFormData = (data: Inputs) => {
  try {
    localStorage.setItem('adoptionFormData', JSON.stringify(data));
  } catch (error) {
    console.error('Error saving form data:', error);
  }
};

const [inputs, setInputs] = useState<Inputs>(() => {
  const savedData = loadSavedFormData();
  return {
    pet_id,
    imageUrl,
    species,
    user: user?.fullName || "",
    user_id: user?.id || "",
    age: savedData?.age || 14,
    phoneNumber: savedData?.phoneNumber || "",
    email: savedData?.email || user?.emailAddresses?.[0]?.emailAddress || "",
    address: savedData?.address || "",
    aptType: savedData?.aptType || "",
    petsAllowed: savedData?.petsAllowed || "",
    kids: savedData?.kids || "",
    otherAnimals: savedData?.otherAnimals || "",
    liveWith: savedData?.liveWith || "",
    employmentStatus: savedData?.employmentStatus || "",
    incomeLevel: savedData?.incomeLevel || "",
    animalsBefore: savedData?.animalsBefore || "",
    nutritionalNeeds: savedData?.nutritionalNeeds || "",
    reasonToAdoptPet: savedData?.reasonToAdoptPet || "",
  };
});

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савицький Р.С.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        characterPreferences: savedData?.characterPreferences || "",
        lifeChangingActions: savedData?.lifeChangingActions || "",
        psychologicalDiseases: savedData?.psychologicalDiseases || "",
        tendencyToBeImpulsive: savedData?.tendencyToBeImpulsive || "",
        emotionalStability: savedData?.emotionalStability || "",
        reactionToStress: savedData?.reactionToStress || "",
        uncontrolledAnger: savedData?.uncontrolledAnger || "",
        blamesForAnger: savedData?.blamesForAnger || "",
        attitudeToSpending: savedData?.attitudeToSpending || "",
        attitudeToBadBehaviour: savedData?.attitudeToBadBehaviour || "",
        requireLove: savedData?.requireLove || "",
        isAbleToFollowSchedule: savedData?.isAbleToFollowSchedule || "",
        friendlySupport: savedData?.friendlySupport || "",
        allergyToFur: savedData?.allergyToFur || "",
        howToRaisePet: savedData?.howToRaisePet || "",
        reactionToBadAction: savedData?.reactionToBadAction || "",
        isPunishmentNecessary: savedData?.isPunishmentNecessary || ""
    }
});

const [isLoading, setIsLoading] = useState(false);

useEffect(() => {
    if (isLoading && user) {
        setInputs((prev) => ({
            ...prev,
            user: user.fullName || prev.user,
            user_id: user.id || prev.user_id,
            email: user.emailAddresses[0]?.emailAddress || prev.email,
        }));
    }
}, [isLoading, user]);

const handleChange = (
    e: React.ChangeEvent<HTMLInputElement | HTMLTextAreaElement |
HTMLSelectElement>
) => {
    setInputs((prev) => ({
        ...prev,
        [e.target.name]: e.target.value === "yes" ? true : e.target.value ===
"no" ? false : e.target.value
    }));
};

const handleSubmit = async (e: React.FormEvent<HTMLFormElement>) => {
    e.preventDefault();

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
if (!emailRegex.test(inputs.email)) {
  toast({ title: tf("error"), description: tf("email"), status: 'error' });
  return;
}

const phoneRegex = /^+380\d{3}\d{2}\d{2}\d{2}$/;
if (!phoneRegex.test(inputs.phoneNumber)) {
  toast({ title: tf("error"), description: tf("phone"), status: 'error' });
  return;
}

const invalidSelects = requiredSelects.filter((field) => inputs[field as
keyof Inputs] === "");
if (invalidSelects.length > 0) {
  toast({ title: tf("error"), description: tf("fields"), status: 'error'
});
  return;
}

setIsLoading(true);

await saveFormData(inputs);
try {
  const res = await fetch(`/api/adoption`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
      ...inputs
    }),
  });

  if (res.ok) {
    router.push('/pets');
    toast({
      title: tf("success"),
      description: tf("successDesc"),
      status: 'success',
      duration: 5000,
      isClosable: true,
    })
  }
} catch (err) {
  const message = err instanceof Error ? err.message : tf("unknownError");

  toast({
    title: tf("error"),
    description: message,
    status: 'error',
    duration: 5000,
  })
}

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        isClosable: true,
    });
} finally {
    setIsLoading(false);
}
};

return (
    <div className="form">
        <form onSubmit={handleSubmit} className="flex flex-wrap gap-6">

            <div className="w-full flex flex-col gap-2">
                <label className="text-sm">{q("age")}</label>
                <input
                    className="ring-1 ring-orange-700 p-4 rounded-sm
placeholder:text-orange-700 outline-none"
                    type="number"
                    placeholder={q("age")}
                    name="age"
                    min="14"
                    max="100"
                    onChange={handleChange}
                    value={inputs.age}
                    required
                />
            </div>

            <div className="w-full flex flex-col gap-2">
                <label className="text-sm">{q("phoneNumber")}</label>
                <input
                    className="ring-1 ring-orange-700 p-4 rounded-sm
placeholder:text-orange-700 outline-none"
                    type="text"
                    placeholder={q("phoneNumber")}
                    name="phoneNumber"
                    onChange={handleChange}
                    value={inputs.phoneNumber}
                    required
                />
            </div>

            <div className="w-full flex flex-col gap-2">
                <label className="text-sm">{q("email")}</label>
                <input
                    className="ring-1 ring-orange-700 p-4 rounded-sm
placeholder:text-orange-700 outline-none"
                    type="text"
                    placeholder={q("email")}
                    name="email"
                    onChange={handleChange}

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        value={inputs.email}
        required
      />
    </div>

    <div className="w-full flex flex-col gap-2">
      <label className="text-sm">{q("address")}</label>
      <input
        className="ring-1 ring-orange-700 p-4 rounded-sm
placeholder:text-orange-700 outline-none"
        type="text"
        placeholder={q("address")}
        name="address"
        onChange={handleChange}
        value={inputs.address}
        required
      />
    </div>

    <div className="w-full flex flex-col gap-2">
      <label className="text-sm">{q("aptType")}</label>
      <select
        className="ring-1 ring-orange-700 p-4 rounded-sm outline-
none"
        name="aptType"
        value={inputs.aptType}
        onChange={handleChange}
      >
        <option value="">{o("select")}</option>
        <option value={o("rentFlat")}>{o("rentFlat")}</option>
        <option value={o("rentHouse")}>{o("rentHouse")}</option>
        <option value={o("ownFlat")}>{o("ownFlat")}</option>
        <option value={o("ownHouse")}>{o("ownHouse")}</option>
      </select>
    </div>

    <div className="w-full flex flex-col gap-2">
      <label className="text-sm">{q("petsAllowed")}</label>
      <select
        className="ring-1 ring-orange-700 p-4 rounded-sm outline-
none"
        name="petsAllowed"
        value={inputs.petsAllowed}
        onChange={handleChange}
      >
        <option value="">{o("select")}</option>
        <option value={o("yes")}>{o("yes")}</option>
        <option value={o("no")}>{o("no")}</option>
      </select>
    </div>

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

none"

```
<div className="w-full flex flex-col gap-2">
  <label className="text-sm">{q("kids")}</label>
  <select
    className="ring-1 ring-orange-700 p-4 rounded-sm outline-
    name="kids"
    value={inputs.kids}
    onChange={handleChange}
  >
    <option value="">{o("select")}</option>
    <option value={o("yes")}>{o("yes")}</option>
    <option value={o("no")}>{o("no")}</option>
  </select>
</div>
```

placeholder:text-orange-700 outline-none"

```
<div className="w-full flex flex-col gap-2">
  <label className="text-sm">{q("otherAnimals")}</label>
  <input
    className="ring-1 ring-orange-700 p-4 rounded-sm
    type="text"
    placeholder={q("otherAnimals")}
    name="otherAnimals"
    onChange={handleChange}
    value={inputs.otherAnimals}
    required
  />
</div>
```

none"

```
<div className="w-full flex flex-col gap-2">
  <label className="text-sm">{q("liveWith")}</label>
  <select
    className="ring-1 ring-orange-700 p-4 rounded-sm outline-
    name="liveWith"
    value={inputs.liveWith}
    onChange={handleChange}
  >
    <option value="">{o("select")}</option>
    <option value={o("alone")}>{o("alone")}</option>
    <option value={o("family")}>{o("family")}</option>
    <option value={o("friends")}>{o("friends")}</option>
  </select>
</div>
```

```
<div className="w-full flex flex-col gap-2">
  <label className="text-sm">{q("employmentStatus")}</label>
  <select
```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

none"

```
className="ring-1 ring-orange-700 p-4 rounded-sm outline-  
  
name="employmentStatus"  
value={inputs.employmentStatus}  
onChange={handleChange}
```

>

```
<option value="">{o("select")}</option>  
<option value={o("employed")}>{o("employed")}</option>  
<option value={o("student")}>{o("student")}</option>  
<option value={o("unemployed")}>{o("unemployed")}</option>  
<option value={o("retired")}>{o("retired")}</option>
```

</select>

</div>

```
<div className="w-full flex flex-col gap-2">
```

```
<label className="text-sm">{q("incomeLevel")}</label>
```

```
<select
```

```
className="ring-1 ring-orange-700 p-4 rounded-sm outline-
```

none"

```
name="incomeLevel"
```

```
value={inputs.incomeLevel}
```

```
onChange={handleChange}
```

>

```
<option value="">{o("select")}</option>
```

```
<option
```

```
value={o("lessThanFiveHundred")}>{o("lessThanFiveHundred")}</option>
```

```
<option value="500-1000$" >500-1000$</option>
```

```
<option value="1000-2000$" >1000-2000$</option>
```

```
<option
```

```
value={o("moreThanTwoThousand")}>{o("moreThanTwoThousand")}</option>
```

</select>

</div>

```
<div className="w-full flex flex-col gap-2">
```

```
<label className="text-sm">{q("animalsBefore")}</label>
```

```
<select
```

```
className="ring-1 ring-orange-700 p-4 rounded-sm outline-
```

none"

```
name="animalsBefore"
```

```
value={inputs.animalsBefore}
```

```
onChange={handleChange}
```

>

```
<option value="">{o("select")}</option>
```

```
<option value={o("yes")}>{o("yes")}</option>
```

```
<option value={o("no")}>{o("no")}</option>
```

</select>

</div>

```
<div className="w-full flex flex-col gap-2">
```

```
<label className="text-sm">{q("nutritionalNeeds")}</label>
```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савицький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

        <input
            className="ring-1 ring-orange-700 p-4 rounded-sm
placeholder:text-orange-700 outline-none"
            type="text"
            placeholder={q("nutritionalNeeds")}
            name="nutritionalNeeds"
            onChange={handleChange}
            value={inputs.nutritionalNeeds}
            required
        />
    </div>

    <div className="w-full flex flex-col gap-2">
        <label className="text-sm">{q("reasonToAdoptPet")}</label>
        <textarea
            className="ring-1 ring-orange-700 p-4 rounded-sm
placeholder:text-orange-700 outline-none"
            name="reasonToAdoptPet"
            onChange={handleChange}
            value={inputs.reasonToAdoptPet}
            required
        ></textarea>
    </div>

    <div className="w-full flex flex-col gap-2">
        <label className="text-sm">{q("characterPreferences")}</label>
        <textarea
            className="ring-1 ring-orange-700 p-4 rounded-sm
placeholder:text-orange-700 outline-none"
            name="characterPreferences"
            onChange={handleChange}
            value={inputs.characterPreferences}
            required
        ></textarea>
    </div>

    <div className="w-full flex flex-col gap-2">
        <label className="text-sm">{q("lifeChangingActions")}</label>
        <textarea
            className="ring-1 ring-orange-700 p-4 rounded-sm
placeholder:text-orange-700 outline-none"
            name="lifeChangingActions"
            onChange={handleChange}
            value={inputs.lifeChangingActions}
            required
        ></textarea>
    </div>

    <div className="w-full flex flex-col gap-2">
        <label className="text-sm">{q("psychologicalDiseases")}</label>

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

none"

```
<select
  className="ring-1 ring-orange-700 p-4 rounded-sm outline-

  name="psychologicalDiseases"
  value={inputs.psychologicalDiseases}
  onChange={handleChange}
>
  <option value="">{o("select")}</option>
  <option value={o("yes")}>{o("yes")}</option>
  <option value={o("no")}>{o("no")}</option>
</select>
</div>
```

none"

```
<div className="w-full flex flex-col gap-2">
  <label className="text-sm">{q("tendencyToBeImpulsive")}</label>
  <select
    className="ring-1 ring-orange-700 p-4 rounded-sm outline-

    name="tendencyToBeImpulsive"
    value={inputs.tendencyToBeImpulsive}
    onChange={handleChange}
  >
    <option value="">{o("select")}</option>
    <option value={o("yes")}>{o("yes")}</option>
    <option value={o("no")}>{o("no")}</option>
  </select>
</div>
```

none"

```
<div className="w-full flex flex-col gap-2">
  <label className="text-sm">{q("emotionalStability")}</label>
  <select
    className="ring-1 ring-orange-700 p-4 rounded-sm outline-

    name="emotionalStability"
    value={inputs.emotionalStability}
    onChange={handleChange}
  >
    <option value="">{o("select")}</option>
    <option value={o("yes")}>{o("yes")}</option>
    <option value={o("no")}>{o("no")}</option>
  </select>
</div>
```

none"

```
<div className="w-full flex flex-col gap-2">
  <label className="text-sm">{q("reactionToStress")}</label>
  <select
    className="ring-1 ring-orange-700 p-4 rounded-sm outline-

    name="reactionToStress"
    value={inputs.reactionToStress}
  >
    <option value="">{o("select")}</option>
    <option value={o("yes")}>{o("yes")}</option>
    <option value={o("no")}>{o("no")}</option>
  </select>
</div>
```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савицький Р.С.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

none"

```
        onChange={handleChange}
      >
        <option value="">{o("select")}</option>
        <option value={o("calm")}>{o("calm")}</option>
        <option value={o("worry")}>{o("worry")}</option>
        <option value={o("panic")}>{o("panic")}</option>
      </select>
    </div>

    <div className="w-full flex flex-col gap-2">
      <label className="text-sm">{q("uncontrolledAnger")}</label>
      <select
        className="ring-1 ring-orange-700 p-4 rounded-sm outline-

        name="uncontrolledAnger"
        value={inputs.uncontrolledAnger}
        onChange={handleChange}
      >
        <option value="">{o("select")}</option>
        <option value={o("yes")}>{o("yes")}</option>
        <option value={o("no")}>{o("no")}</option>
      </select>
    </div>

    <div className="w-full flex flex-col gap-2">
      <label className="text-sm">{q("blamesForAnger")}</label>
      <select
        className="ring-1 ring-orange-700 p-4 rounded-sm outline-

        name="blamesForAnger"
        value={inputs.blamesForAnger}
        onChange={handleChange}
      >
        <option value="">{o("select")}</option>
        <option value={o("yes")}>{o("yes")}</option>
        <option value={o("no")}>{o("no")}</option>
      </select>
    </div>

    <div className="w-full flex flex-col gap-2">
      <label className="text-sm">{q("attitudeToSpending")}</label>
      <select
        className="ring-1 ring-orange-700 p-4 rounded-sm outline-

        name="attitudeToSpending"
        value={inputs.attitudeToSpending}
        onChange={handleChange}
      >
        <option value="">{o("select")}</option>
        <option value={o("ready")}>{o("ready")}</option>
      </select>
    </div>
```

none"

none"

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

none"

```

        <option value={o("notSure")}>{o("notSure")}</option>
        <option value={o("notReady")}>{o("notReady")}</option>
    </select>
</div>

<div className="w-full flex flex-col gap-2">
    <label className="text-sm">{q("attitudeToBadBehaviour")}</label>
    <select
        className="ring-1 ring-orange-700 p-4 rounded-sm outline-

        name="attitudeToBadBehaviour"
        value={inputs.attitudeToBadBehaviour}
        onChange={handleChange}
    >
        <option value="">{o("select")}</option>
        <option value={o("patience")}>{o("patience")}</option>
        <option value={o("refuse")}>{o("refuse")}</option>
        <option value={o("noIdea")}>{o("noIdea")}</option>
    </select>
</div>

<div className="w-full flex flex-col gap-2">
    <label className="text-sm">{q("requireLove")}</label>
    <select
        className="ring-1 ring-orange-700 p-4 rounded-sm outline-

        name="requireLove"
        value={inputs.requireLove}
        onChange={handleChange}
    >
        <option value="">{o("select")}</option>
        <option value={o("yes")}>{o("yes")}</option>
        <option value={o("no")}>{o("no")}</option>
    </select>
</div>

<div className="w-full flex flex-col gap-2">
    <label className="text-sm">{q("isAbleToFollowSchedule")}</label>
    <select
        className="ring-1 ring-orange-700 p-4 rounded-sm outline-

        name="isAbleToFollowSchedule"
        value={inputs.isAbleToFollowSchedule}
        onChange={handleChange}
    >
        <option value="">{o("select")}</option>
        <option value={o("yes")}>{o("yes")}</option>
        <option value={o("no")}>{o("no")}</option>
    </select>
</div>
```

none"

none"

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

none"

```
<div className="w-full flex flex-col gap-2">
  <label className="text-sm">{q("friendlySupport")}</label>
  <select
    className="ring-1 ring-orange-700 p-4 rounded-sm outline-
    name="friendlySupport"
    value={inputs.friendlySupport}
    onChange={handleChange}
  >
    <option value="">{o("select")}</option>
    <option value={o("yes")}>{o("yes")}</option>
    <option value={o("no")}>{o("no")}</option>
  </select>
</div>
```

none"

```
<div className="w-full flex flex-col gap-2">
  <label className="text-sm">{q("allergyToFur")}</label>
  <select
    className="ring-1 ring-orange-700 p-4 rounded-sm outline-
    name="allergyToFur"
    value={inputs.allergyToFur}
    onChange={handleChange}
  >
    <option value="">{o("select")}</option>
    <option value={o("yes")}>{o("yes")}</option>
    <option value={o("no")}>{o("no")}</option>
  </select>
</div>
```

none"

```
<div className="w-full flex flex-col gap-2">
  <label className="text-sm">{q("howToRaisePet")}</label>
  <select
    className="ring-1 ring-orange-700 p-4 rounded-sm outline-
    name="howToRaisePet"
    value={inputs.howToRaisePet}
    onChange={handleChange}
  >
    <option value="">{o("select")}</option>
    <option value={o("positive")}>{o("positive")}</option>
    <option value={o("strict")}>{o("strict")}</option>
    <option value={o("noIdea")}>{o("noIdea")}</option>
  </select>
</div>
```

```
<div className="w-full flex flex-col gap-2">
  <label className="text-sm">{q("reactionToBadAction")}</label>
  <select
```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савицький Р.С.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        className="ring-1 ring-orange-700 p-4 rounded-sm outline-
none"

        name="reactionToBadAction"
        value={inputs.reactionToBadAction}
        onChange={handleChange}
      >
        <option value="">{o("select")}</option>
        <option value={o("fix")}>{o("fix")}</option>
        <option value={o("insult")}>{o("insult")}</option>
        <option value={o("punish")}>{o("punish")}</option>
      </select>
    </div>

    <div className="w-full flex flex-col gap-2">
      <label className="text-sm">{q("isPunishmentNecessary")}</label>
      <select
        className="ring-1 ring-orange-700 p-4 rounded-sm outline-
none"

        name="isPunishmentNecessary"
        value={inputs.isPunishmentNecessary}
        onChange={handleChange}
      >
        <option value="">{o("select")}</option>
        <option value={o("yes")}>{o("yes")}</option>
        <option value={o("no")}>{o("no")}</option>
      </select>
    </div>

    <button
      type="submit"
      className="bg-orange-500 p-4 text-white w-48 rounded-md relative
h-14 flex items-center justify-center"
    >
      {isLoading ? o("creating") : o("submit")}
    </button>
  </form>
</div>

);
};

export default AdoptionForm;

```

Додаток Б.

Лістинг коду з файлу components/AIHelper.tsx:

```

"use client";
import { useEffect, useState } from "react";
import { useSession, useUser } from "@clerk/nextjs";
import { handleNewMessage } from "@/actions/handleNewMessage";

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import { checkUserRole } from "@/utils/userUtils";
import { usePathname } from "next/navigation";
import { useLocale, useTranslations } from "next-intl";

export default function AIHelper() {
  const { session } = useSession();
  const { isLoading } = useUser();
  const [message, setMessage] = useState("");
  const locale = useLocale();
  const t = useTranslations("ai-helper");
  const path = usePathname();

  const userRole = !isLoading ? "guest" : checkUserRole(session) || "guest";

  useEffect(() => {
    const fetchHint = async () => {
      const formData = new FormData();
      formData.append("chatId", "ai-helper");
      formData.append("newMessage", t("message"));
      formData.append("pageContext", path);
      formData.append("userRole", userRole);

      const response = await handleNewMessage(formData);
      if (response?.message) {
        setMessage(response.message);
      }
    };

    fetchHint();

  }, [userRole, path, locale]);

  if (!message) return null;

  return (
    <div className="fixed bottom-4 right-4 bg-white p-4 shadow-lg rounded-lg w-80">
      <p className="text-sm">{message}</p>
    </div>
  );
}

```

Додаток В.

Лістинг коду з файлу actions/handleNewMessage.ts:

```

"use server";
import OpenAI from "openai";
import { auth } from "@clerk/nextjs/server";
import { v4 as uuidv4 } from "uuid";

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import { revalidatePath } from "next/cache";
import { prisma } from "../../lib/prisma";
import { getTranslations } from "next-intl/server";
import { ChatCompletionMessageParam } from "openai/resources/chat/completions";

export const handleNewMessage = async (formData: FormData) => {
  const t = getTranslations("ai-helper");

  const openai = new OpenAI({
    apiKey: process.env.OPENAI_API_KEY!,
  });

  const userId = auth().userId || "guest";
  const chatId = formData.get("chatId") as string;
  const rawMessage = (formData.get("newMessage") as string)?.trim();
  const pageContext = formData.get("pageContext") as string;
  const userRole = formData.get("userRole") as string;

  if (!rawMessage) return;

  const userInput = rawMessage;

  const systemMessage = `${(await t)("systemMessage")}
  ${(await t)("role")}: ${userRole}.
  ${(await t)("page")}: ${pageContext}.
  ${(await t)("task")}`;

  if (chatId === "ai-helper") {
    try {
      const response = await openai.chat.completions.create({
        model: "gpt-3.5-turbo",
        temperature: 0.2,
        messages: [
          { role: "system", content: systemMessage },
          { role: "user", content: userInput },
        ],
      });

      return { message: response.choices[0].message.content || "No response from AI" };
    } catch (error) {
      console.error("Error generating AI response", error);
      return { message: "Error generating AI response." };
    }
  }

  try {
    await prisma.message.create({
      data: {
        id: uuidv4(),

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        chat_id: chatId,
        user_id: userId,
        content: userInput,
        role: "user",
        createdAt: new Date(),
    },
});

revalidatePath("/");

const existingMessages = await prisma.message.findMany({
    where: { chat_id: chatId, user_id: userId },
    orderBy: { createdAt: "asc" },
    select: { role: true, content: true },
}) as ChatCompletionMessageParam[];

const response = await openai.chat.completions.create({
    model: "gpt-4-turbo",
    messages: [
        { role: "system", content: systemMessage },
        ...existingMessages,
        { role: "user", content: userInput },
    ],
});

const botResponse = response.choices[0].message.content || "No response from
bot";

await prisma.message.create({
    data: {
        id: uuidv4(),
        chat_id: chatId,
        user_id: userId,
        content: botResponse,
        role: "assistant",
        createdAt: new Date(),
    },
});

return { message: botResponse };
} catch (error) {
    console.error("Error processing message", error);
    return { message: "Error processing your request." };
} finally {
    await prisma.$disconnect();
}
};

```

Лістинг коду з файлу /api/pets/recommend/route.ts:

```
import { NextRequest, NextResponse } from "next/server";
import { prisma } from "../../../../../lib/prisma";
import OpenAI from "openai";

const openai = new OpenAI({
  apiKey: process.env.OPENAI_API_KEY,
});

export async function POST(req: NextRequest) {
  try {
    const body = await req.json();
    const { income, space, freeTime, experience, kids, categories } = body;

    const hasFormData = income || space || freeTime || experience || kids;

    if (!hasFormData) {
      return NextResponse.json([]);
    }

    let petsQuery: any = {
      where: {
        isFeatured: true
      },
      select: {
        id: true,
        engName: true,
        engSpecies: true,
        category: {
          select: {
            engName: true,
            slug: true
          }
        }
      }
    };

    if (categories && categories.length > 0) {
      petsQuery.where = {
        AND: [
          { isFeatured: true },
          { catSlug: { in: categories } }
        ]
      };
    }

    const allPets = await prisma.pet.findMany(petsQuery);
```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савіцький Р.С.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (allPets.length === 0) {
    return NextResponse.json([]);
}

const filters = {
    income,
    space,
    freeTime,
    experience,
    kids
};

const prompt = `
    You are a pet recommendation expert. Based on the user's lifestyle and
    preferences, recommend the most suitable pets from the available options.

    User Information:
    - Monthly Income: ${filters.income || 'Not specified'}
    - Living Situation: ${filters.space || 'Not specified'}
    - Free Time per Week: ${filters.freeTime || 'Not specified'}
    - Pet Experience Level: ${filters.experience || 'Not specified'}
    - Has Children: ${filters.kids || 'Not specified'}

    Available Pets (JSON format):
    ${JSON.stringify(allPets, null, 2)}

    Instructions:
    1. Consider the user's income for pet care costs (food, vet bills, supplies)
    2. Match living space with pet size and exercise needs
    3. Consider time commitment required for different pets
    4. Factor in experience level - recommend easier pets for beginners
    5. Consider child safety and pet temperament if children are present

    Return ONLY a JSON array of the pet IDs that are most suitable, ordered by
    best match first.

    Do not include any explanations or additional text - just the JSON array of
    string Ids, not more than 8.
    `;

const response = await openai.chat.completions.create({
    model: "gpt-4",
    messages: [{ role: "user", content: prompt }],
    temperature: 0.2,
});

const aiResponse = response.choices[0].message.content?.trim() || "";

if (!aiResponse) {

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савицький Р.С.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return NextResponse.json({ message: "No response from AI" }, { status:
500 });
    }

    let recommendedPetIds;
    try {
        recommendedPetIds = JSON.parse(aiResponse);
    } catch (error) {
        console.error("Failed to parse AI response:", aiResponse);

        const arrayMatch = aiResponse.match(/\[\d,\s]+\]/);
        if (arrayMatch) {
            try {
                recommendedPetIds = JSON.parse(arrayMatch[0]);
            } catch (e) {
                return NextResponse.json({ message: "Invalid AI response format"
}, { status: 500 });
            }
        } else {
            return NextResponse.json({ message: "Invalid AI response format" }, {
status: 500 });
        }
    }

    if (!Array.isArray(recommendedPetIds) || recommendedPetIds.length === 0) {
        return NextResponse.json([]);
    }

    const processedIds = recommendedPetIds.map(id => String(id));

    const recommendedPets = await prisma.pet.findMany({
        where: {
            id: { in: processedIds }
        },
        include: {
            category: true
        }
    });

    const sortedPets = processedIds.map(id => recommendedPets.find(pet => pet.id
=== id)).filter(Boolean);
    return NextResponse.json(sortedPets);
} catch (error) {
    console.error("Error recommending pets:", error);
    return NextResponse.json({ message: "Failed to get pet recommendations" }, {
status: 500 });
}
}

```

		Пархоменко О.Є.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Савицький Р.С.				86
Змн.	Арк.	№ докум.	Підпис	Дата		