НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
__інформаційних систем і технологій__
(назва кафедри)

_____ **/ Швиденко М.З. /**
(підпис) (ПІБ)

“___” _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Чат бот для пошуку нових власників тварин»
Спеціальність 122 – «Комп’ютерні науки»

Гарант освітньої програми

_____ _____ _____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____ _____ __Мокрієв М.В.__
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Онищук Максим Сергійович_____
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
___інформаційних систем і технологій___
(назва кафедри)

___к.е.н., доцент_____**Швиденко М.З**.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"___" _____ 2025 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Онищук Максим Сергійович

Спеціальність 122 – «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Чат бот для пошуку нових власників тварин» затверджена наказом ректора НУБіП України від 26.02.2025 №238с
2. Термін подання завершеної роботи на кафедру_____
(рік, місяць, число)
3. Вихідні дані до роботи: створення чат-боту для пошуку нових власників тварин.
4. Перелік питань, що розглядаються:
   - Аналіз проблемної області
   - Моделювання предметної області
   - Проєктування програмної системи
   - Впровадження та експлуатація системи

Дата видачі завдання "___"_____ 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____/ Мокрієв М.В. /
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання:_____/Онищук М.С. /
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська робота присвячена розробці **Telegram-бота** для автоматизованого пошуку нових власників для тварин або пухнастих друзів для людей. Мета проєкту — створити **корисний інструмент** для реєстрації профілів, перегляду анкет, обміну вподобаннями та пошуку взаємних симпатій для отримання контактів і подальшого пошуку власників для тварин.

Рішення реалізовано на **Telegram** з використанням **Python** та бібліотеки **pyTelegramBotAPI**, а також **SQLite**. Система включає реєстрацію користувачів (тварина/власник), заповнення та фільтрацію анкет, механізм "вподобайок", обробку взаємних симпатій та скарг.

Інтерфейс повністю інтегрований у Telegram, що забезпечує зручність на мобільних пристроях. Ця система підвищує ефективність пошуку домівок для тварин, скорочує час комунікації та полегшує роботу волонтерів.

ABSTRACT

This bachelor's thesis focuses on developing a **Telegram bot** for automated pet adoption or finding furry companions. The project's goal is to create an **intuitive tool** for profile registration, Browse, exchanging likes, and finding mutual matches to reveal contact information.

The solution is implemented as a **Telegram chatbot** using **Python** and the **pyTelegramBotAPI** library, with **SQLite** for data storage. The system features user registration (pet/owner), profile creation and filtering, a like-based interaction system, mutual match handling, and complaint reporting.

All user interactions occur within Telegram, ensuring a convenient mobile-friendly interface. This system enhances pet adoption efficiency, reduces communication time, and assists volunteers.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання бакалаврської роботи	Строк виконання етапів бакалаврської роботи	Примітка
1	Отримання завдання	26 лютого 2025	
2	Системний аналіз предметної області	березень 2025	
3	Інформаційне забезпечення	квітень 2025	
4	Прикладне програмне забезпечення	квітень 2025	
5	Рекомендації щодо впровадження та експлуатації системи	травень 2025	
6	Оформлення записки	травень 2025	
7	Перевірка на плагіат	травень 2025	
8	Проходження нормо контролю	травень 2025	
9	Проходження передзахисту	2 червня 2025	
10	Захист роботи	червень 2025	

Студент _____ Онищук М.С.

(підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

доцент к.т.н. _____ Мокрієв М.В.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

ЗМІСТ

ВСТУП

У сучасному цифровому світі автоматизація процесів взаємодії між людьми та організаціями значно підвищує ефективність комунікацій і скорочує витрати часу. Однією з соціально важливих сфер, яка все ще потребує широкого впровадження сучасних інформаційних технологій, є пошук нових власників для безпритульних тварин або нових домашніх улюбленців для людей. Проблема безпритульних тварин, а також складність у налагодженні ефективного обміну інформацією між потенційними власниками та власниками тварин, залишається актуальною.

Більшість процесів з пошуку нових родин для тварин відбуваються вручну через соціальні мережі, публікації, форуми або паперові анкети, що ускладнює фільтрацію даних, супровід тварин, фіксацію історій та звітність. Такі методи нерідко призводять до втрати інформації, дублювання запитів та зниження довіри між сторонами. Це знижує ефективність зусиль волонтерів, ускладнює адопцію тварин і затримує процес їхнього влаштування.

Актуальним рішенням є створення сучасної інформаційної системи, яка автоматизує процеси пошуку, реєстрації, підбору анкет і комунікації між користувачами, заснованої на доступних цифрових інструментах. Зокрема, платформа Telegram забезпечує ідеальне середовище для швидкої взаємодії, зручного доступу з мобільних пристроїв і високої надійності в обробці даних.

Метою цієї бакалаврської роботи є розробка Telegram-бота "PawLink", який забезпечує двосторонню взаємодію між потенційними власниками тварин та тими, хто шукає новий дім для своїх улюбленців. Система надає можливість створення анкет, встановлення фільтрів пошуку, здійснення взаємних вподобань, перегляду запитів та обміну контактними даними при наявності сходження між сторонами.

Рішення реалізовано за допомогою мови програмування Python із використанням бібліотеки pyTelegramBotAPI. Для зберігання всіх необхідних даних (анкет, фотографій, запитів, вподобайок, скарг) використовується

реляційна база даних SQLite. Інтерфейс системи побудований у вигляді чат-бота з інтерактивними кнопками, повідомленнями та клавіатурами Telegram.

Актуальність роботи полягає у використанні сучасних технологій для вирішення соціально важливого завдання — пошуку родини для тварин або нових улюбленців для людей. Telegram-бот дозволяє централізувати дані, уникати дублювання, прискорює процес комунікації та забезпечує прозорість та зручність для всіх учасників процесу.

Об’єктом дослідження є процес комунікації між власниками тварин, потенційними власниками та власниками тварин. Предметом дослідження є процес автоматизації цієї комунікації за допомогою Telegram-бота з функціональністю реєстрації, перегляду анкет, застосування фільтрів, взаємних вподобань та обміну контактами.

Для досягнення поставленої мети були визначені такі завдання:

- проаналізувати наявні інструменти автоматизації пошуку нових власників тварин;
- визначити функціональні та нефункціональні вимоги до Telegram-бота;
- розробити структуру бази даних для збереження профілів, фотографій, лайків та звітів;
- реалізувати логіку взаємодії з користувачами за допомогою інтерфейсу Telegram;
- забезпечити обробку та фільтрацію анкет відповідно до вибраних критеріїв;
- протестувати бот на надійність, зручність користування та функціональність.

Результатом роботи є повноцінно функціональний Telegram-бот, який може використовуватися для організації процесу взаємного пошуку між людьми та тваринами. Система сприятиме зменшенню кількості безпритульних тварин, покращенню комунікації між сторонами та забезпечить ефективний і масштабований інструмент для волонтерських та соціальних ініціатив.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

Проблема безпритульних тварин та складність у пошуку для них нових родин є соціально важливим питанням, що вимагає сучасних технологічних рішень. Щороку тисячі тварин опиняються на вулиці або в притулках. Їх подальша доля часто залежить від ефективності пошуку відповідального власника. У цій сфері вкрай необхідно забезпечити зручну комунікацію між волонтерами, притулками, потенційними власниками та людьми, які з певних причин змушені шукати новий дім для своїх улюбленців.

На сьогодні більшість процесів відбувається вручну — через публікації в соціальних мережах, оголошення на форумах або навіть через паперові анкети. Такі методи не мають централізованої бази даних, відсутні фільтри пошуку, а комунікація є неструктурованою. Усе це призводить до дублювання інформації, втрати даних, затримок у підборі анкет і труднощів у взаємодії між сторонами.

У сучасних умовах Telegram є одним із найзручніших і найпоширеніших каналів зв’язку. Його функціонал дозволяє створювати інтерактивних ботів, які автоматизують процеси реєстрації, перегляду анкет, застосування фільтрів та взаємного зв’язку. Саме це дозволяє значно спростити пошук нових родин для тварин або нових улюбленців для потенційних власників.

Telegram-бот може вирішити основні проблеми: збір даних, зручне подання анкет, автоматичну фільтрацію за віком, типом тварини або статтю, надання зворотного зв’язку, обробку скарг і формування запитів на знайомство. Усі дані зберігаються в централізованій базі, що виключає дублювання та втрату інформації.

Більшість існуючих систем у цій сфері або застарілі, або не пристосовані до вимог сучасного користувача, не мають мобільного або інтерактивного інтерфейсу, не дозволяють вести персоналізований пошук та не передбачають захисту персональних даних. Також відсутні можливості для автоматичного обміну контактами при взаємній згоді або взаємодії з ботом без посередництва людини.

Необхідність у впровадженні зручного, швидкого і безпечного засобу комунікації між тими, хто шукає тварину, та тими, хто її пропонує, є очевидною. Telegram-бот може стати універсальним інструментом для таких взаємодій, забезпечуючи легке керування даними, швидке прийняття рішень та підвищення якості сервісу.

Класичними учасниками предметної області є: користувачі (люди або тварини), власники тварин, адміністратори, а також програмний бот, що відповідає за логіку взаємодії.

Приклад структури класів предметної області наведено в таблиці.

Таблиця 1.1

Опис атрибутів класів предметної області

Клас предметної області	Атрибут	Опис
Користувач	Ім’я	Ім’я користувача або контактної особи
	Вік	Вік користувача
	Телеграм-нікнейм	Telegram username (за наявності)
	Телефон	Номер телефону для зв’язку
	Тип акаунту	Потенційний власник або "Я хвіст"
Тварина	Тип тварини	Собака / Кіт / Інше
	Вік	До 1 року / 1–5 років / Старше 5 років
	Стать	Чоловіча / Жіноча
	Опис	Коротка інформація про тварину
	Фото 1–3	Фотографії тварини

Взаємодія (вподобайки)	ID користувача	Хто поставив вподобайку
	ID користувача анкети	Кому поставлено вподобайку
	Статус	Відправлено / Підтверджено / Відхилено
	Дата та час	Дата взаємодії
Скарга	Хто поскаржився	Ідентифікатор користувача
	На кого подано скаргу	Ідентифікатор користувача
	Причина	Текст скарги
	Дата подання	Дата й час скарги
Повідомлення / Подія	Вміст	Текст повідомлення або дії
	Тип взаємодії	Реєстрація, перегляд, скарга тощо
	Дата й час	Дата створення повідомлення або дії

Ця таблиця забезпечує чітке уявлення про структуру інформації в системі Telegram-бота для допомоги у пошуку нових родин для тварин.

1.2 Огляд існуючих рішень

На сьогодні в Україні та за кордоном існує чимало рішень, що використовуються у сфері пошуку нових власників для тварин, які перебувають у притулках або тимчасових домівках. Найчастіше це веб-сайти або мобільні застосунки з оголошеннями, однак ефективність цих систем часто обмежена через складність користування, необхідність реєстрації, або відсутність зручного зв’язку між потенційними власниками та тими, хто опікується тваринами.

Більшість існуючих сервісів (наприклад, OLX, Animal ID, Petfinder тощо) не мають гнучкої комунікаційної інфраструктури, яка дозволяє швидко відповідати на запити користувачів, надавати актуальну інформацію про тварин або проводити попереднє анкетування потенційних власників. Часто

адміністратори або волонтери повинні вручну обробляти вхідні запити, що значно знижує ефективність і збільшує навантаження.

Одним із перспективних напрямків у вирішенні цієї проблеми є використання чат-ботів у месенджерах, зокрема в Telegram. Telegram-боти дозволяють реалізувати автоматичну обробку запитів, фільтрацію тварин за параметрами (вік, порода, розмір, стать, місцезнаходження), автоматизовану реєстрацію заявок, а також інтеграцію з базами даних притулків або окремих волонтерів.

Наприклад, деякі українські волонтерські ініціативи використовують базові Telegram-боти, які дозволяють показати фото тварин або дати контакт волонтера. Проте функціонал таких ботів обмежується лише переглядом інформації або надсиланням оголошення вручну. Вони не забезпечують повноцінний пошук за параметрами, автоматичне оновлення інформації, або систему обробки анкет потенційних власників.

Крім того, більшість ботів не мають системи модерації, оцінки надійності заявників або форм попереднього опитування. Також відсутні функції зворотного зв’язку, аналітики та керування базою тварин. У результаті — процес пошуку нового власника залишається повільним, неструктурованим та трудомістким.

Інтеграція чат-бота Telegram як веб-орієнтованої інформаційної системи для автоматизації пошуку власників тварин дозволить значно покращити процеси взаємодії між користувачами та тими, хто хоче знайти пухнастому нового господара. Такий бот може містити:

- анкетування користувача;
- підбір тварин за заданими параметрами;
- функції перевірки та модерації;
- автоматичне надсилання повідомлень або запитів адміністраторам;
- формування звітів щодо заявок.

Таким чином, розробка спеціалізованого Telegram-бота для пошуку нових власників тварин є актуальним і необхідним кроком для підвищення

ефективності процесу прилаштування тварин. Це дозволить волонтерам зосередитись на живому спілкуванні з людьми, а технічні функції (пошук, фільтрація, облік) передати системі, що працює 24/7.

1.3 Постановка завдання

Основною метою розробки програмного забезпечення є створення Telegram-бота, який забезпечить зручний, швидкий та ефективний процес пошуку нових власників для безпритульних або покинутих тварин. Такий чат-бот має спростити взаємодію між користувачами та волонтерами, централізувати інформацію про доступних тварин, автоматизувати облік анкет і комунікацію, а також допомогти в прийнятті рішень щодо прилаштування.

Система має забезпечити структуровану базу даних, яка зберігатиме детальну інформацію про тварин (ім’я, вік, стать, тип, фото), а також анкети користувачів, включаючи їхні контактні дані, відповіді на запитання анкети, тощо. Бот повинен дозволяти швидкий пошук тварин за параметрами, зручно показувати результати у форматі повідомлень Telegram та дозволяти користувачам переглядати та обробляти анкети.

Окрім основної інформаційної функції, бот повинен містити модулі сповіщень — наприклад, про появу нових вподобайок від інших користувачів, зміну статусу вподобайки(у разі позитивного рішення). Це дасть змогу оперативно реагувати на події та покращить ефективність роботи волонтерів і притулків.

Інтерфейс Telegram-бота має бути максимально простим для користувача: послідовна логіка взаємодії, зрозумілі кнопки, автоматичні підказки та перевірки даних. Для власників тварин має бути передбачено зручне додавання та редагування інформації про тварин, а також обробка анкет користувачів.

Загалом, реалізація цього проєкту сприятиме більш структурованій, швидкій та безпечній комунікації між тими, хто шукає тварину, та тими, хто

опікується їх прилаштуванням. Telegram-бот зменшить навантаження на волонтерів, дозволить уникнути плутанини з заявками, покращить облік і дозволить ефективно приймати рішення щодо передачі тварин у нові родини.

1.4 Функціональні та нефункціональні вимоги

Функціональні вимоги

1. Реєстрація користувача — бот підтримує покрокову реєстрацію потенційного власника або тварини з автоматичним збереженням у базу.
2. Введення анкети тварини — у ролі тварини користувач проходить анкетування (тип, вік, стать, фото, опис).
3. Пошук анкет з фільтрами — користувачі можуть переглядати анкети з фільтрацією за віком, типом і статтю тварини або віком власника.
4. Система вподобайок — реалізована взаємна система "Подобається", яка відкриває контактну інформацію у разі збігу.
5. Скарги — користувачі можуть надсилати скарги на анкети з поясненням.
6. Перегляд запитів — користувачі можуть переглядати вхідні запити й підтверджувати або відхиляти їх.
7. Профіль користувача — кожен має змогу переглядати свою анкету та видаляти її.
8. Головне меню та довідка — бот надає зручну навігацію, повернення в меню, довідник та технічну підтримку.

Нефункціональні вимоги

1. Надійність — бот обробляє неочікувані помилки без аварійного завершення, логування у консолі.
2. Сумісність з Telegram — реалізовано використання inline-кнопок, клавіатур та команд згідно з API Telegram.
3. Мобільна адаптивність — бот підтримується на всіх пристроях, де працює Telegram.

4. Мінімізація порогу входу — немає потреби у реєстрації через сторонні сервіси — лише запуск бота.
5. Продуктивність — пошук і відображення анкет реалізовано через SQLite з фільтрами та кешуванням переглянутих профілів.
6. Безпека — базова перевірка користувачів, обмеження доступу до особистої інформації до моменту взаємного підтвердження.
7. Гнучкість — можливість змінити профіль або видалити його в будь-який момент.
8. Масштабованість — реалізовано розширювану структуру бази даних для подальшого розвитку (скарги, чорні списки, адміністрування).

1.5 Вимоги до інтерфейсу користувача

Інтерфейс бота в Telegram побудований у вигляді покрокових сценаріїв з контекстною клавіатурою. Всі дії користувача відбуваються через кнопки або відповіді в чаті.

- Головне меню включає основні дії: перегляд анкет, мій профіль, перегляд запитів, довідка.
- Реєстрація відбувається у вигляді діалогу з ботом: користувач послідовно вводить свої дані (або анкета тварини) — бот зберігає їх у базу.
- Профіль відображається з описом та фото. Передбачена можливість перегляду, редагування або видалення.
- Фільтрація анкет реалізована через кнопки: користувач вибирає критерії (тип, вік, стать) перед пошуком.
- Відображення анкет — у вигляді повідомлень із фото, описом і кнопками: «Подобається», «Наступна», «Поскаржитись».
- Взаємодія при збігу симпатій — обидва користувачі отримують повідомлення з контактами.
- Довідник і підтримка — доступні з меню, містять інструкції та контакт техпідтримки.

Інтерфейс витримує стиль Telegram: короткі інформативні повідомлення, обмеження за кількістю символів, використання Markdown для форматування.

Інтерфейс повинен враховувати обмеження Telegram — мінімум тексту в одному повідомленні, компактність і зрозумілі підписи до кнопок. Для важливих дій застосовуються підтвердження (наприклад, «Ви впевнені, що хочете видалити анкету?»).

Завдяки Telegram як платформі, бот автоматично адаптується до мобільних пристроїв, підтримує спливаючі підказки, реакцію на команди та inline-кнопки, що створює зручний користувацький досвід.

Загальна мета інтерфейсу — забезпечити максимально легкий процес знайомства з тваринами та подачі заявки без потреби сторонньої допомоги.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні відомості

У процесі створення Telegram-бота для пошуку нових власників тварин важливо використовувати уніфіковану мову моделювання UML, яка дозволяє формалізувати структуру й поведінку системи ще до етапу реалізації. UML є ефективним інструментом проєктування для будь-якої складної інформаційної системи, і в контексті розробки чат-бота вона допомагає чітко візуалізувати архітектуру, ролі користувачів, сценарії взаємодії, логіку обробки заявок і взаємозв’язки між даними.

UML-діаграми дозволяють описати основні об'єкти предметної області: тварини, анкети, вподобайки, користувачі (потенційні власники, власники тварин), а також способи взаємодії з ними через Telegram-бота. Це

дає змогу сформувати логічну модель системи, визначити функціональні блоки та оцінити масштабованість проєкту.

Діаграми варіантів використання ілюструють, як різні учасники — звичайні користувачі, адміністратори — взаємодіють із ботом. Наприклад, користувач створює анкету, переглядає тварин, шукає за фільтрами; власник тварини додає анкету тварини або змінює статус вподобайок; адміністратор переглядає загальну статистику або модерує доступ.

Діаграми класів відображають структуру даних, які зберігаються в системі. Наприклад, клас "Тварина" містить атрибути (ім’я, вік, стать, опис), клас "Користувач" — ім’я, Telegram ID, роль, контактна інформація; клас "Вподобайка" — посилання на користувача, статус, дату створення тощо. Всі ці об’єкти пов’язані між собою, і UML дозволяє відобразити ці зв’язки у зрозумілій формі.

Діаграма діяльності використовується для моделювання процесів, наприклад, «Додавання тварини власником». Вона демонструє логіку виконання дій у вигляді послідовності кроків — це особливо корисно для побудови діалогів у боті, де кожне повідомлення є етапом логічного сценарію.

Діаграми послідовності описують обмін повідомленнями між користувачем і ботом, а також між ботом і базою даних у межах конкретного сценарію. Наприклад, при зміні статусу вподобайки користувач взаємодіє з ботом, бот оновлює статус у БД, а потім надсилає повідомлення користувачу. Це дозволяє візуально зрозуміти, як функціонує система зсередини.

Для побудови загальної архітектури Telegram-бота також можна використати діаграми компонентів для опису модулів (бот, база даних, API обробки зображень тощо) та діаграми для відображення інфраструктури: сервер, хмарне сховище.

Використання UML-діаграм у розробці Telegram-бота дозволяє:

- формалізувати логіку взаємодії користувачів із системою;
- визначити точні вимоги до функціональності;
- покращити комунікацію між розробником, замовником і волонтерами;

- уникнути непорозумінь у реалізації;
- створити якісну технічну документацію, яка буде корисною в подальшій підтримці та масштабуванні проєкту.

Таким чином, UML виступає потужним інструментом для візуалізації й опису як технічної, так і функціональної структури Telegram-бота, що дозволяє створити ефективну, надійну й зрозумілу систему для пошуку нових власників тварин.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів

Діаграма прецедентів (use case diagram) Telegram-бота PawLink ілюструє основні сценарії взаємодії користувачів із системою.

Рис. 1 Діаграма прецедентів

Ключовими акторами є:

- Користувач (User) — фізична особа, яка може зареєструватися в одній із ролей: "потенційний власник" або "хвіст " (представник тварини).
- Telegram-платформа — посередник між користувачем і ботом, який передає команди, повідомлення, контакти, фото.
- Система (бот) — виконує бізнес-логіку: реєстрацію, фільтрацію, перегляд анкет, збереження даних, обробку взаємодій.

Прецеденти користувача (обидві ролі):

- Запустити бота (/start)
- Пройти реєстрацію (вибір типу акаунту, ім’я, вік, телефон, опис)
- Завантажити фото (1–3)
- Переглядати анкети протилежної ролі
- Застосувати фільтри (вік, стать, тип тварини або вік власника)
- Надіслати вподобайку (виконує функцію запиту)
- Переглядати вхідні запити
- Підтвердити або відхилити запит
- Отримати контакт у разі взаємної симпатії
- Подати скаргу на анкету
- Видалити профіль

Прецеденти Telegram-платформи:

- Приймати/надсилати повідомлення, інлайн-кнопки
- Приймати/передавати контакти
- Приймати/передавати фото
- Обробляти callback-запити (інлайн-фільтри)

Прецеденти системи (бота):

- Зберігати дані в базу (SQLite)
- Створювати профілі з фото
- Фільтрувати анкети за критеріями
- Реєструвати вподобайки як "запити"
- Визначати взаємну симпатію
- Розкривати контакти при взаємності
- Зберігати та обробляти скарги
- Видаляти профілі користувачів
- Відправляти повідомлення, картки анкет, запити

Сценарій: Пошук тварини

1. Користувач запускає бота та обирає «Я потенційний власник».

2. Проходить реєстрацію, вводить дані.
3. Обирає фільтри пошуку (тип, вік, стать тварини).
4. Переглядає анкети, ставить «вподобайки».
5. У разі взаємної симпатії обом відкриваються контакти.

Сценарій: Додавання тварини

1. Користувач обирає роль «Я хвіст».
2. Вказує тип тварини, вік, стать, опис.
3. Завантажує 1–3 фотографії.
4. Завершує реєстрацію, анкета зберігається.
5. Потенційні власники можуть переглядати та ставити вподобайки.

2.2.2 Діаграма послідовності

Діаграма послідовності є інструментом, що дозволяє відстежити хронологію викликів функцій, обміну повідомленнями та логіки між компонентами Telegram-бота PawLink. У цій системі основна взаємодія відбувається між користувачем, Telegram-ботом, бекенд-сервером (Python + SQLite) та базою даних.

Ця діаграма демонструє типовий сценарій — реєстрацію нового користувача типу "Я хвіст" (власник тварини), яка передбачає кілька кроків: вибір типу, введення даних, завантаження фото тощо.

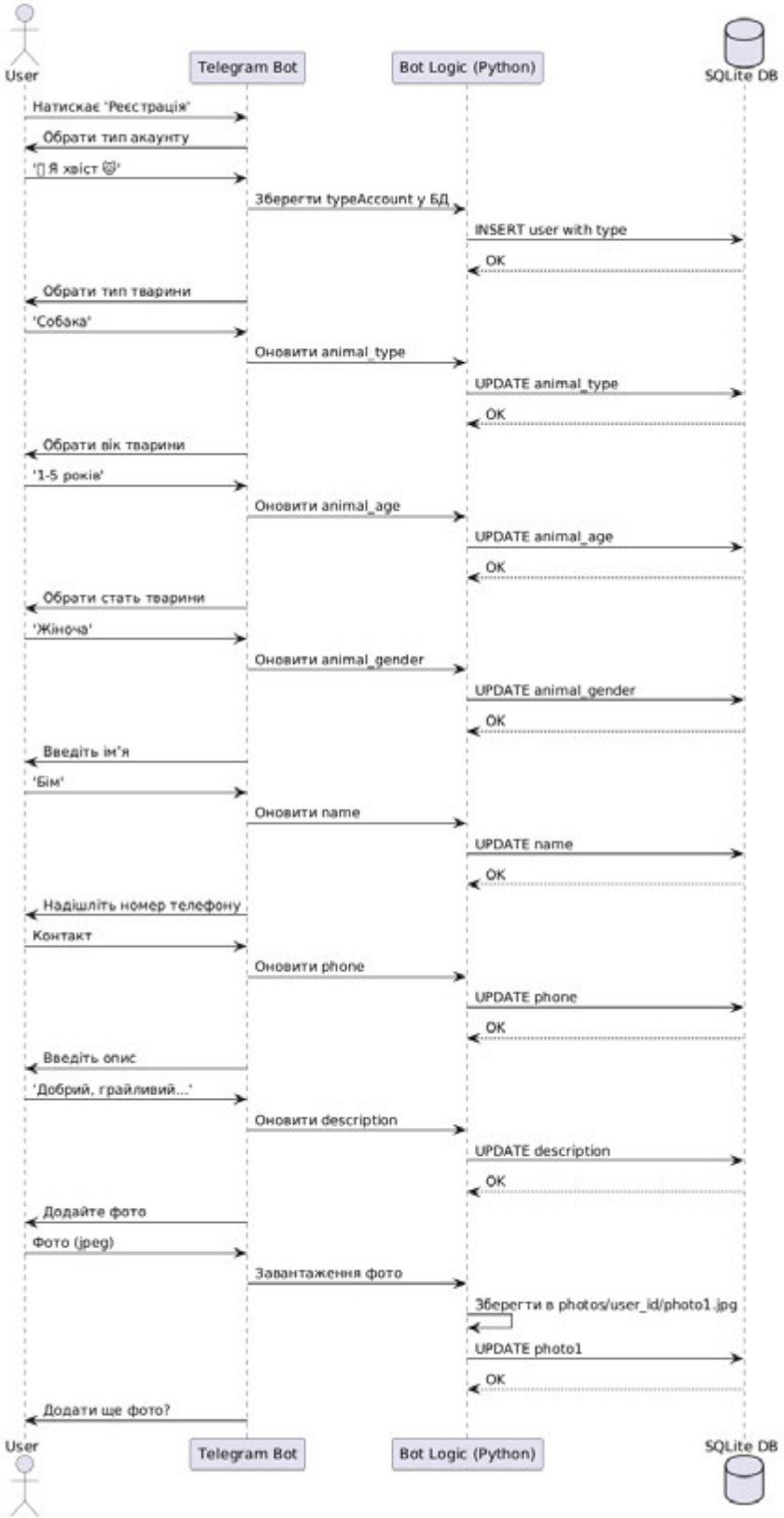

Рис. 2 Діаграма послідовності

Учасники:

- User (Користувач) — ініціює процес реєстрації.
- Telegram Bot — приймає команди, відображає інтерфейс.

- Bot Logic (Python) — обробляє повідомлення, оновлює стан.
- SQLite Database — зберігає дані анкети користувача.

Сценарій: реєстрація користувача типу "Я хвіст"

Послідовність дій:

1. Користувач натискає кнопку «Реєстрація».
2. Бот запитує тип акаунту (хвіст чи власник).
3. Користувач обирає «Я хвіст ».
4. Бот запитує тип тварини (собака/кіт/інше).
5. Користувач обирає «Собака».
6. Бот запитує вік тварини (до 1 року, 1–5, 5+).
7. Користувач обирає «1–5 років».
8. Бот запитує стать тварини.
9. Користувач вводить «Жіноча».
10. Бот запитує контактне ім’я.
11. Користувач вводить ім’я.
12. Бот запитує номер телефону.
13. Користувач надсилає контакт.
14. Бот запитує опис анкети.
15. Користувач вводить опис.
16. Бот запитує перше фото.
17. Користувач надсилає фото.
18. Бот зберігає фото у локальну папку.
19. Бот додає шлях до фото в БД.
20. Бот запитує, чи хоче додати ще фото.

2.2.3 Діаграма активності.

Діаграма активності — це один із типів діаграм UML, який дозволяє моделювати поведінку системи у вигляді послідовного або паралельного потоку дій. Вона особливо корисна для відображення логіки бізнес-процесів, сценаріїв взаємодії з користувачем або алгоритмів, що включають вибір, цикли та умови.

На відміну від діаграм послідовності, які зосереджуються на взаємодії об'єктів у часі, діаграми активності роблять акцент на загальному потоці управління процесом. Вони показують, які дії виконуються, в якій

послідовності, та які умови або рішення впливають на зміну маршруту виконання.

У контексті системи PawLink — Telegram-бота для пошуку нових власників для тварин або навпаки — діаграма активності дозволяє проаналізувати основний сценарій реєстрації користувача та додавання анкети до бази даних. Цей процес є критичним для функціонування системи, адже саме з нього починається будь-яка подальша взаємодія з ботом.

На діаграмі активності показано типовий потік взаємодії нового користувача з ботом. Сценарій починається з надсилання команди /start. Далі користувач вибирає тип облікового запису: або потенційний власник тварини, або тварина (яку реєструє власник тварини). Після цього, залежно від вибраного типу, бот пропонує заповнити відповідну анкету.

У випадку з «потенційним власником» заповнюється інформація про ім’я, вік, контактний номер телефону та короткий опис. Для облікового запису типу «тварина» вказується тип тварини (кіт, собака, інше), її вік, стать, а також опис і фотографії.

Після збору всіх даних бот здійснює перевірку введеної інформації та реєструє користувача в базі даних. У разі успішної реєстрації користувач отримує підтвердження та переходить до головного меню, де може переглядати анкети інших користувачів, ставити «Подобається»(вподобайку), переглядати запити, змінювати фільтри тощо.

Ця діаграма допомагає проаналізувати користувацький досвід на етапі onboarding, виявити слабкі місця та переконатись, що усі критичні сценарії враховано. Крім того, вона є основою для реалізації контролю потоку в логіці Telegram-бота.

Розроблена діаграма активності наведена нижче:

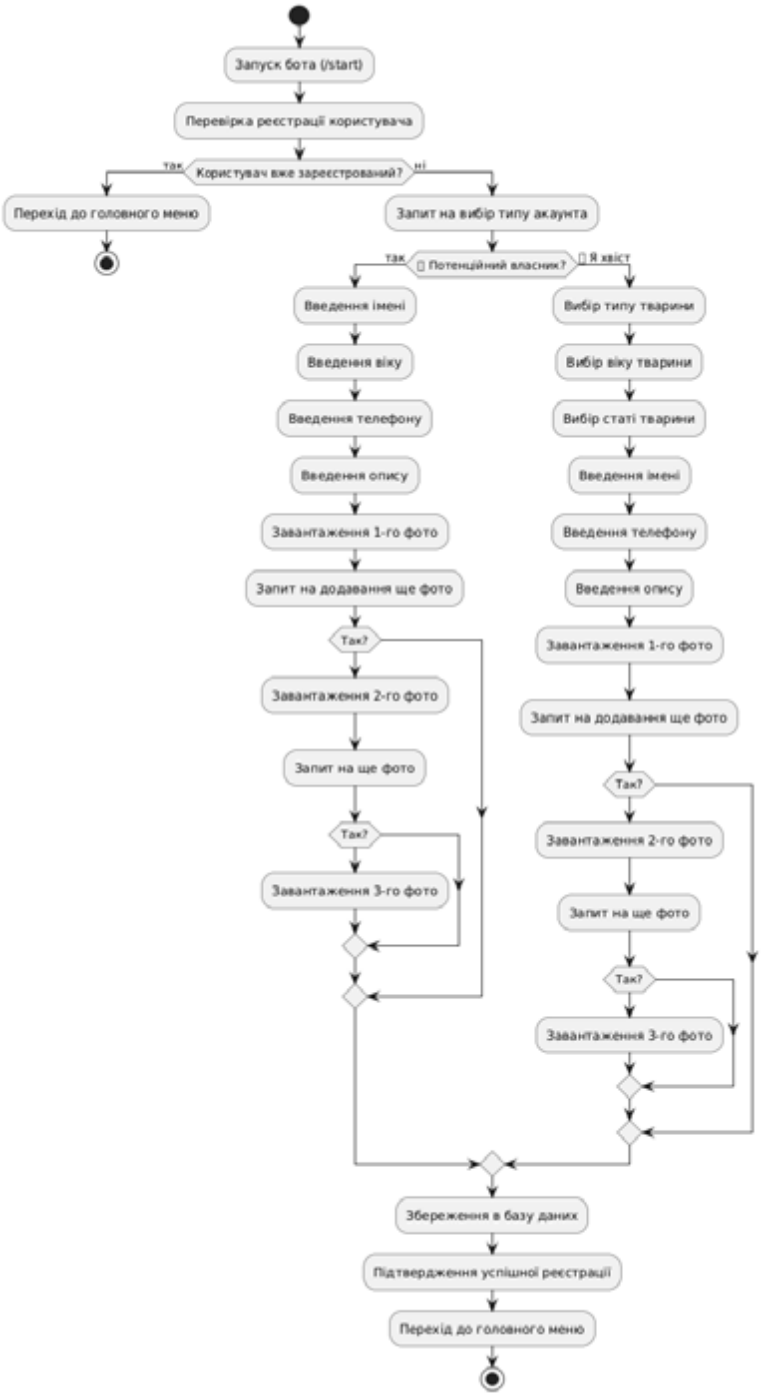

Рис. 3 Діаграма активності

Ця діаграма чітко описує логіку дій користувача з моменту запуску бота до завершення реєстрації. Вона буде корисною для всіх учасників розробки системи, адже дозволяє побачити сценарій без потреби заглиблюватися у код.

2.3 Абстракції предметної області

У сфері розробки програмного забезпечення поняття "абстракції" передбачає спрощене представлення складних систем за допомогою виокремлення ключових сутностей, їхніх властивостей та взаємозв’язків. Абстракції дозволяють створити концептуальну модель предметної області, що забезпечує ефективне проєктування логіки роботи додатку та покращує розуміння загальної структури системи.

У випадку з PawLink — Telegram-ботом для пошуку нових власників для тварин або майбутніх улюбленців для людей — предметна область зосереджена на взаємодії між користувачами двох типів: "потенційний власник" та "тварина". Кожна роль має власну поведінку, атрибути та функціональність, що впливають на подальші бізнес-процеси. Користувачі створюють профілі, додають анкети, переглядають інші анкети, встановлюють фільтри, надсилають запити, обмінюються контактами при взаємній вподобайці та можуть залишати скарги на інші анкети у разі порушення правил.

Основні абстракції, виділені у системі:

Кожен об’єкт має чітко визначені атрибути та методи, які дозволяють реалізовувати функціональність відповідно до бізнес-логіки. Наприклад, клас Pet включає поля animal_type, animal_age та animal_gender, а також методи для заповнення анкети. Клас Owner має поле age і методи для взаємодії з анкетами тварин.

Використання абстракцій дозволяє ефективно масштабувати систему, зменшити дублювання коду та покращити її підтримку. Наприклад, логіка запитів або лайків реалізована однаково для обох ролей, що досягається завдяки спільній батьківській структурі User.

Таким чином, абстракції предметної області в системі *PawLink* дають змогу зосередитись на головних функціональних аспектах, забезпечують гнучкість при розширенні та роблять систему прозорою як для розробників, так і для користувачів.

2.4 Діаграма класів

Діаграма класів є основоположним інструментом об’єктно-орієнтованого проектування, який відображає статичну структуру системи. У контексті розробки Telegram-бота PawLink, ця діаграма демонструє основні сутності, їх атрибути, методи та взаємозв’язки, що забезпечують роботу системи пошуку нових власників для тварин.

У системі усі користувачі представлені єдиним класом User, який включає інформацію як про потенційних власників, так і про тварин, що шукають нову домівку. Тип акаунту (typeAccount) визначає роль користувача: "Я хвіст" або "Я потенційний власник". Атрибути користувача адаптовані відповідно до обраної ролі — наприклад, для тварини зберігається тип, вік, стать; для власника — людський вік, контактна інформація тощо.

Ключові класи:

- User — центральна сутність, що містить основні атрибути профілю: ID, ім’я, опис, контакт, тип акаунту, інформацію про тварину або власника, а також шляхи до трьох фото.
- Like — реалізує взаємодію типу "вподобайка", яка одночасно виконує функцію запиту на знайомство. Містить ID того, хто поставив лайк, кому він адресований, статус ("Отправлен", "Підтверджено"), а також часову мітку.
- Report — фіксує скарги. Включає ID того, хто подає скаргу, ID користувача, на якого подано скаргу, причину та час.

- Blacklist — допоміжний клас, що зберігає користувачів, які були заблоковані.
- Filter — представлений у вигляді словника в коді, але у діаграмі фіксується як окремий клас, який зберігає тимчасові критерії пошуку (вік, стать, тип тощо).

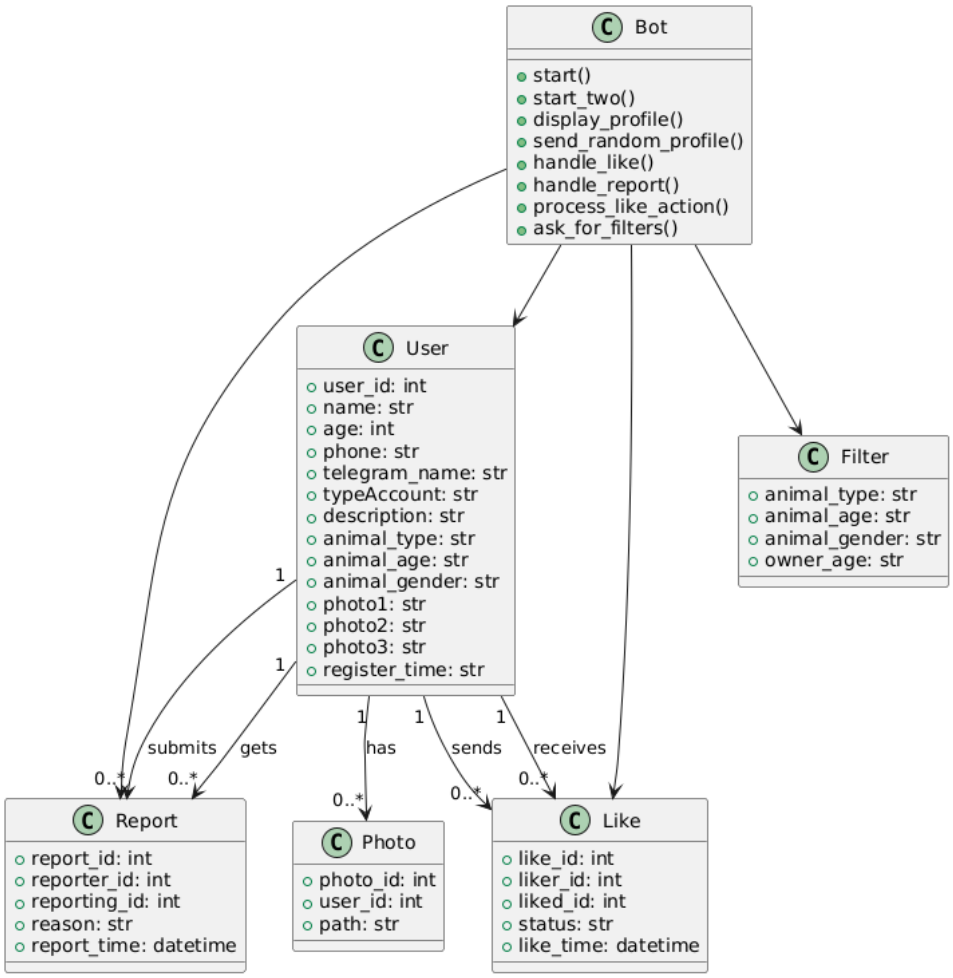

Рис. 4 Діаграма класів

У діаграмі класів зв’язки між User та іншими класами визначають такі відношення:

- `User → Like: один користувач може створити багато лайків (асоціація 1:N).`

- User → Report: один користувач може подати кілька скарг або бути об’єктом скарг.
- User → Photo: фото є частиною атрибутів користувача, зберігаються як photo1, photo2, photo3.

Таким чином, діаграма класів PawLink наочно демонструє основні компоненти системи, їх взаємозв’язки та підтримує прозоре розуміння логіки додатку. Вона служить не лише документаційною основою, а й інструментом перевірки повноти реалізації архітектури Telegram-бота.

2.5 Функціональна модель

Функціональна модель системи, заснована на методі структурного аналізу та проєктування (SADT), дозволяє представити Telegram-бот PawLink у вигляді ієрархічної структури з логічним розділенням на основні функціональні блоки. Такий підхід забезпечує чітке розуміння того, як окремі частини системи взаємодіють між собою, як дані переміщуються між компонентами, та які механізми беруть участь у виконанні тих чи інших задач.

SADT-метод ідеально підходить для моделювання складних інформаційних систем, таких як PawLink, де необхідно враховувати не лише взаємодію користувачів з ботом, але й логіку зберігання, обробки даних, перевірки дій, реагування на запити та модерацію контенту.

На верхньорівневій SADT-діаграмі зображено ключовий процес: Організація взаємодії користувачів у системі PawLink. Центральним об’єктом виступає інформаційна система Telegram-бота. До неї надходять вхідні дані (анкета, параметри фільтрації, скарги), система керується певними правилами (бізнес-логіка, обмеження Telegram API, умови пошуку), функціонує за допомогою визначених механізмів (користувачі, Telegram-бот, база даних, адміністратор) і генерує результати взаємодії (профіль користувача, відібрані анкети, вподобайки, обмін контактами).

Рис. 5 Функціональна модель

Telegram-бот PawLink виконує низку основних функцій:

- Реєстрація користувачів (профілювання у ролі тварини або майбутнього власника).
- Створення анкети з фотографіями.
- Встановлення фільтрів пошуку.
- Перегляд анкет інших користувачів.
- Надсилання “вподобайок” або запитів.
- Визначення взаємності та відкриття контактів.
- Повідомлення про дії в системі (вподобайки, запити, нові профілі).
- Можливість подання скарг.
- Збереження історії взаємодій.

Ключові функціональні блоки Telegram-бота, представлені на SADT-діаграмі:

- Реєстрація профілю: збереження базових даних користувача.
- Створення анкети**:** введення параметрів анкети та додавання фото.

- Фільтрація анкет: застосування пошукових критеріїв.
- Надсилання запитів: взаємодія у вигляді “вподобайок”.
- Обробка відповідей: відкриття контактів при взаємності.
- Обробка скарг: фіксація порушень та надсилання на розгляд.

Перевага SADT-моделі полягає в її здатності забезпечити як загальне уявлення про архітектуру, так і деталізацію ключових сценаріїв взаємодії. Ієрархічна структура дозволяє виділити логіку, залежну від ролі користувача, а також оцінити ділянки, які потребують особливої уваги — наприклад, модерація скарг або уникнення дублювання анкет.

Таким чином, функціональна модель, побудована за методом SADT, дозволяє краще зрозуміти логіку Telegram-бота PawLink, спростити управління архітектурою, забезпечити масштабованість та підтримувати високий рівень користувацького досвіду.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

Логічна модель даних чат-бота PawLink, призначеного для пошуку нових власників тварин, визначає структуру ключових сутностей, зв’язків між ними та правила цілісності даних, які реалізують основну бізнес-логіку системи. Вона розробляється на концептуальному рівні, не залежить від конкретної СУБД чи технології зберігання, але служить базою для побудови фізичної моделі.

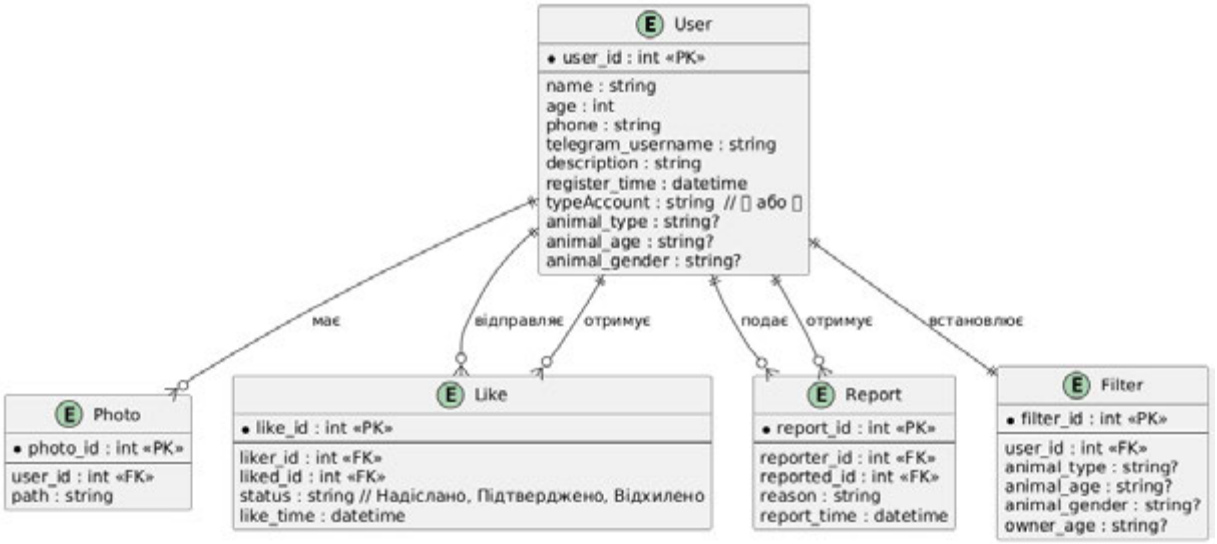

Рис. 6 Логічна модель даних

Основні завдання моделі**:**

- забезпечити відповідність структури даних функціональним вимогам (реєстрація, пошук, знайомства, обмін контактами);
- реалізувати логіку двосторонньої взаємодії (вподобайки-запити);
- підтримувати збереження скарг, фільтрів і медіа;
- гарантувати консистентність зв’язків між користувачами.

Основні сутності системи:

1. Користувач (User)

Усі користувачі системи — і ті, хто шукає тварину, і ті, хто представляє тварину — зберігаються в одній таблиці. Тип акаунту (потенційний власник чи хвіст) вказується у полі typeAccount. Для типу "хвіст" додатково вказуються характеристики тварини: вид, вік, стать. Атрибути:

- Telegram ID (унікальний ідентифікатор),
- ім’я, вік, номер телефону, опис,
- тип акаунту (typeAccount),
- характеристики тварини (якщо це хвіст).

2. Фото (Photo)

До кожного користувача може бути прикріплено від 1 до 3 фотографій, що зберігаються у файловій системі, а в базі зберігаються посилання.

3. Вподобайка (Like)

Користувач може поставити вподобайку (фактично — запит) іншому користувачу. Якщо обидві сторони поставили вподобайки — бот надає контактну інформацію. Вподобайка містить:

- ID відправника (liker_id) і отримувача (liked_id),
- статус (надіслано, підтверджено, відхилено),
- час дії.

4. Скарга (Report)

Дозволяє подати скаргу на користувача у разі неприйнятної поведінки. Містить:

- ID скаржника,
- ID користувача, на якого скарга,
- причину, дату подання.

5. Фільтр пошуку (Filter)

Зберігає обрані користувачем фільтри пошуку. Якщо це потенційний власник — фільтрує анкети хвостів (тип тварини, вік, стать). Якщо це хвіст — фільтрує за віком власника.

Зв’язки між сутностями:

- Один користувач може мати багато фотографій (1:N).
- Один користувач може надіслати або отримати багато вподобайок (1:N в обидва боки).
- Один користувач може подати кілька скарг і отримати скарги від інших.
- Кожен користувач має лише один набір фільтрів (1:1).

Переваги такої структури:

- Єдина таблиця користувачів дозволяє уникнути дублювання структур.
- Вподобайки реалізують знайомства з умовною взаємністю.
- Гнучка підтримка фільтрів для обох ролей.
- Масштабована структура для збереження медіа й запитів.

3.2 Вибір системи управління базою даних та її реалізація

У межах створення чат-бота для пошуку нових власників тварин важливою складовою є розробка надійної та ефективної системи зберігання даних. Оскільки бот працює з обмеженим обсягом структурованої інформації та орієнтований на локальне розгортання без використання серверної інфраструктури, доцільним є використання реляційної вбудованої системи управління базами даних SQLite.

Серед основних причин вибору SQLite слід відзначити її простоту у використанні, відсутність потреби у зовнішньому сервері, підтримку мови

SQL, наявність транзакцій та цілісності даних. Крім того, SQLite є сумісною з мовою програмування Python, у якій реалізовано логіку функціонування бота, що дозволяє безпосередньо виконувати запити до бази даних за допомогою стандартного модуля sqlite3.

Фізична модель бази даних розроблена на основі логічної структури, узгодженої з функціональними вимогами до системи. Центральною сутністю є таблиця users, яка містить повну інформацію як про користувачів, що шукають тварин, так і про тих, хто публікує анкети тварин. Це дозволило уникнути дублювання таблиць для різних типів акаунтів. Роль користувача визначається полем typeAccount, що має два можливі значення: «Я потенційний власник» або «Я хвіст». У межах одного запису зберігається інформація про ім’я, вік, контактні дані, тип і характеристики тварини (за потреби), а також шляхи до фотографій.

Додатково реалізовано три допоміжні таблиці:

- likes — фіксує взаємодії між користувачами (зокрема лайки та підтверджені запити);
- reports — містить скарги на некоректні або фейкові анкети;
- blacklist — використовується для зберігання заблокованих користувачів або спамерів.

Встановлення зв’язків між сутностями здійснюється за допомогою зовнішніх ключів: наприклад, у таблиці likes зберігаються liker_id та liked_id, які є посиланнями на user_id з таблиці users.

Дані з Telegram (ідентифікатор користувача, нікнейм, контактний номер) автоматично зберігаються в базу під час реєстрації або взаємодії з ботом. Усі SQL-операції — вставка, оновлення, вибірка та фільтрація — здійснюються через код Python, інтегрований з Telegram Bot API. Основні дії

користувачів (реєстрація, перегляд анкет, надсилання запитів або скарг) викликають відповідні запити до бази даних.

Реалізована структура забезпечує цілісність, ефективність пошуку, підтримку фільтрації та модульність. Таким чином, SQLite повністю задовольняє потреби даного проєкту як у технічному, так і в архітектурному аспектах, дозволяючи зосередитись на логіці функціонування бота без додаткових витрат на інфраструктуру.

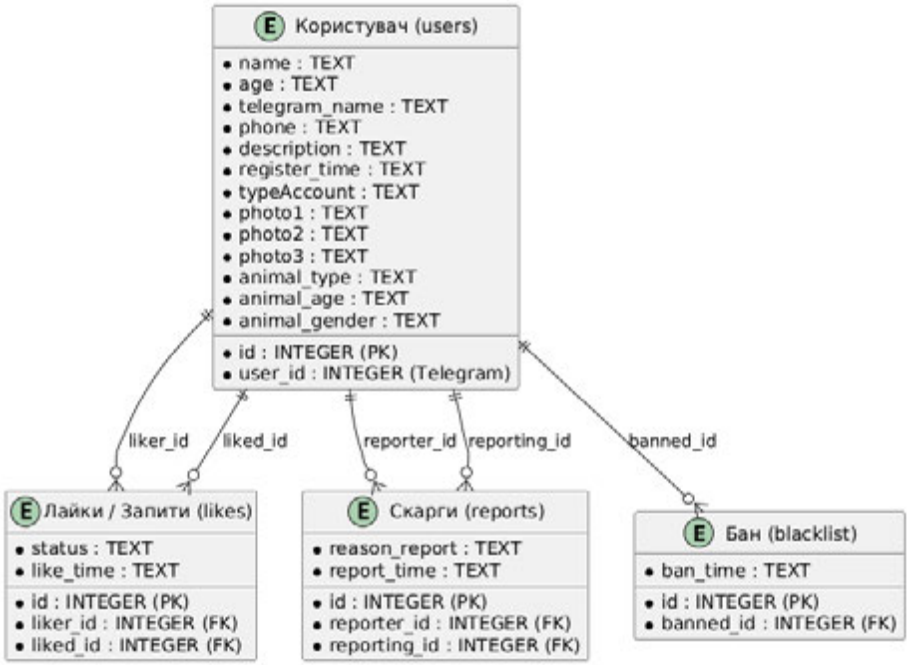

Рис. 7 ER-діаграма

Пояснення до ER-діаграми

1. Користувач (users)

Центральна таблиця, що містить об’єднану інформацію про всіх користувачів системи — як тих, хто шукає тварину, так і тих, хто її пропонує. Роль користувача визначається полем typeAccount. У таблиці також зберігаються персональні дані (ім’я, вік, контактна інформація), опис профілю, тип тварини (якщо актуально), її характеристики (вік, стать), а також шляхи до фотографій, пов’язаних із профілем.

2. Вподобайки (likes)

Таблиця, що фіксує взаємодії між користувачами. Кожен запис представляє один запит або лайк — хто на кого його надіслав (liker_id → liked_id), поточний статус взаємодії (наприклад, "Отправлен", "Підтверджено") та час дії. Всі зовнішні ключі посилаються на унікальні user_id у таблиці users.

3. Скарги (reports)

Таблиця містить інформацію про подані скарги. Кожен запис включає ідентифікатор користувача, що подав скаргу (reporter_id), користувача, на якого скарга надійшла (reporting_id), текст причини скарги та час її подання. Як і в попередньому випадку, всі ідентифікатори є зовнішніми ключами до таблиці users.

4. Бан (blacklist)

Таблиця зберігає інформацію про заблокованих користувачів. Кожен запис містить banned_id, який вказує на користувача, що був доданий до чорного списку, та дату блокування. Ця структура дозволяє реалізовувати базову модерацію та обмеження доступу в системі.

Ця діаграма чітко показує, як різні частини системи взаємодіють з обраною СУБД.

3.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення Telegram-бота для взаємодії між користувачами (потенційними власниками та тими, хто шукає новий дім для тварин) спроєктована відповідно до принципів чіткої багаторівневої організації, модульності, масштабованості та гнучкості. Така архітектура дозволяє забезпечити легке розширення функціональності, зручність у супроводі, ефективну обробку даних і зрозумілу структуру коду.

Функціональність бота реалізована за багаторівневою логічною структурою, яка включає:

Рівень представлення (Presentation Layer)

Цей рівень відповідає за взаємодію користувача з ботом через Telegram. Основне завдання цього шару — приймати повідомлення користувача, передавати їх для обробки на прикладний рівень, а також формувати відповідь. В рамках реалізації використовується бібліотека `telebot`, яка абстрагує Telegram Bot API та надає інструменти для обробки команд, повідомлень, фотографій і контактної інформації. Рівень представлення не містить бізнес-логіки — лише обробку подій і генерацію інтерфейсної відповіді (текстові повідомлення, кнопки, медіа).

Прикладний рівень (Application Layer)

Цей рівень є центральним у всій системі. Він реалізує усі процеси, зокрема:

- керування станами взаємодії з користувачем (`BotState`, `user_states`),
- реєстрацію користувачів та збір їхніх анкетних даних,
- валідацію введених даних (вік, номер телефону, опис тощо),
- завантаження фотографій,
- обробку лайків і взаємних симпатій,
- формування фільтрів пошуку,
- генерацію запитів між користувачами,
- обробку скарг (репортів) на анкети.

Уся логіка представлена у вигляді обробників повідомлень, які визначають подальші дії залежно від поточного стану користувача в контексті

сесії. Це забезпечує "контекстну пам’ять" для кожного користувача на основі подій, які відбуваються в процесі роботи.

Рівень доступу до даних (Data Access Layer)

Цей шар відповідає за взаємодію з реляційною базою даних SQLite. У системі використовується файлова база даних `main.sql`, яка зберігає інформацію про користувачів, лайки, репорти, чорний список. Доступ до бази реалізується за допомогою бібліотеки `sqlite3`, з використанням параметризованих запитів для уникнення SQL-ін'єкцій.

Усі запити на створення, читання, оновлення чи видалення даних (CRUD-операції) реалізуються виключно в цьому шарі, що забезпечує чітку ізоляцію логіки збереження інформації від прикладного коду.

Файлове сховище (File Storage Layer)

Оскільки система працює з користувацькими фотографіями, важливою є підтримка окремого шару зберігання файлів. Для цього реалізовано файлову структуру, де для кожного користувача створюється окрема директорія з фотографіями. У базі зберігаються лише шляхи до відповідних файлів. Це дозволяє зменшити навантаження на базу даних, спростити управління файлами та забезпечити гнучке масштабування.

Взаємодія між компонентами

Типова послідовність дій виглядає так:

1. Користувач надсилає повідомлення боту через Telegram.
2. Повідомлення передається через Telegram Bot API до рівня представлення.
3. Обробник повідомлень передає запит прикладному рівню.

4. Залежно від стану взаємодії, виконується відповідна бізнес-логіка: зчитування/запис у базу даних, перевірка коректності, оновлення сесії, збереження фотографій.

5. Сформована відповідь надсилається назад через Telegram Bot API користувачу.

Захист і продуктивність

Важливою частиною архітектури є захист персональних даних користувачів, таких як номер телефону, нікнейм, фотографії. Захист реалізовано шляхом:

- параметризованих SQL-запитів;
- валідації всіх вхідних даних;
- обмеження доступу до файлової системи;
- ізоляції логіки взаємодії з Telegram від бізнес-логіки та зберігання даних.

Продуктивність системи досягається за рахунок легкості використання SQLite, локального зберігання фотографій і відсутності зовнішніх залежностей. Система здатна обробляти значну кількість паралельних сесій без істотних затримок, завдяки ізольованому зберіганню сесій користувачів у оперативній пам'яті.

Технологічний стек

- Python 3.11 — основна мова реалізації системи;
- telebot (PyTelegramBotAPI) — бібліотека для роботи з Telegram Bot API;
- sqlite3 — вбудована база даних;
- OS, datetime, re, traceback — стандартні модулі для роботи з файлами, часом, текстом і обробкою помилок.

Таким чином, архітектура системи дозволяє ефективно реалізувати цілісну логіку чат-бота, який легко підтримувати, масштабувати та оновлювати з урахуванням нових вимог користувачів і функціоналу.

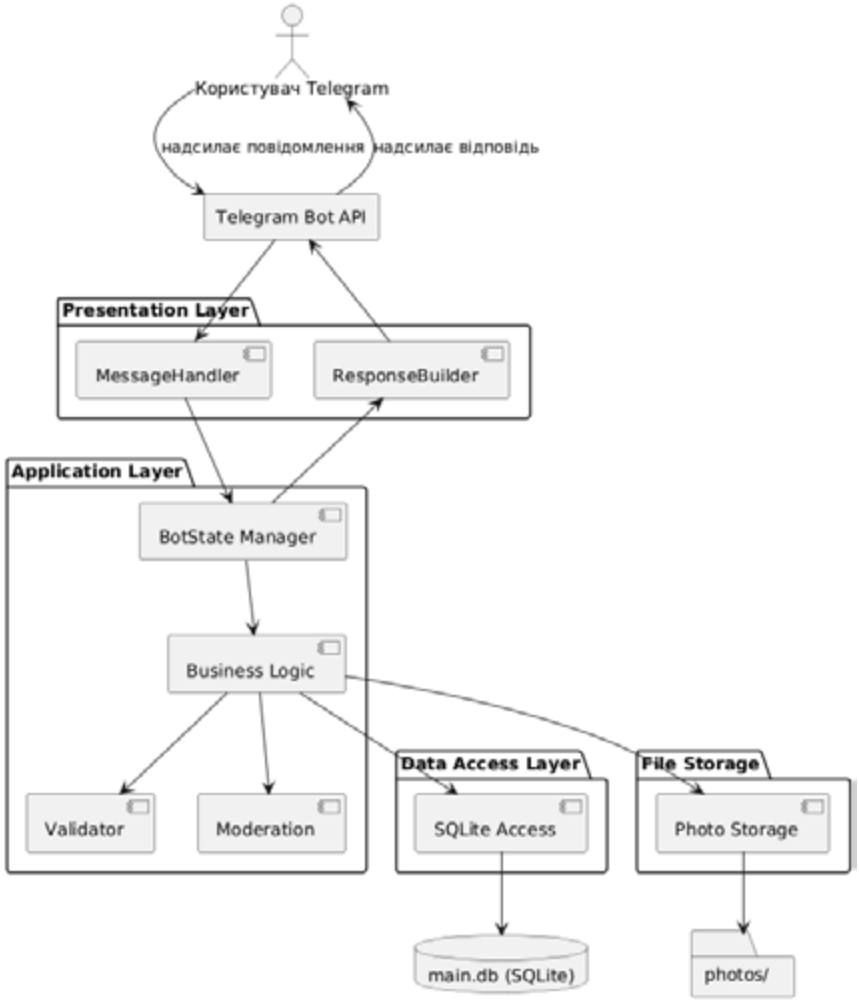

Рис. 8 Діаграма архітектури програмного забезпечення

Ця діаграма відображає основні логічні шари та їхню взаємодію, що відповідає архітектурі, описаній у тексті.

3.4 Організаційна структура програмного забезпечення

3.4.1 Діаграма пакетів

Ця діаграма пакетів ілюструє модульну архітектуру програмного забезпечення чат-бота, розділену на логічні пакети (модулі). Вона демонструє

взаємозв’язки між компонентами системи, їх ролі та напрями обміну даними. Такий підхід сприяє модульності, масштабованості та спрощенню підтримки програмного коду.

Основні пакети включають:

- **Telegram API Interaction:** Відповідає за комунікацію з Telegram Bot API.
- **Application Logic:** Містить основну бізнес-логіку бота, керування станами та обробку запитів користувачів.
- **Data Access:** Реалізує доступ до бази даних SQLite.
- **Domain Model:** Логічно містить визначення сутностей та ключових бізнес-правил, хоча в Python-боті це може бути інтегровано в Application Logic.
- **File Storage:** Відповідає за зберігання та керування фотографіями анкет у файловій системі.

Взаємозв’язки між пакетами позначені стрілками, які показують напрямок викликів та залежностей. Це дає змогу чітко відокремлювати відповідальність кожного компонента, що полегшує тестування, зміну або заміну окремих частин системи без впливу на інші.

Рис. 9 Діаграма пакетів

3.5 Вибір інструментарію для створення програмного забезпечення

Розробка чат-бота для пошуку нових власників тварин вимагає ретельного підбору технологій, які забезпечують простоту розробки, гнучкість, стабільну роботу та зручний інтерфейс взаємодії з користувачами через Telegram. Після аналізу вимог до системи, враховуючи потреби в управлінні профілями користувачів та тварин, обробці взаємодій (лайки, запити), а також адміністративних функціях модерації, для реалізації проєкту були обрані такі інструменти: Python 3, бібліотека PyTelegramBotAPI (Telebot) та вбудована СУБД SQLite.

Python 3 виступає основною мовою програмування завдяки своїй універсальності, лаконічності синтаксису та широкій екосистемі бібліотек. Його об'єктно-орієнтований підхід та можливість швидкої розробки дозволяють ефективно організувати структуру коду, що сприяє легкій підтримці та масштабуванню системи в майбутньому. Простота Python робить його відмінним вибором для розробки чат-ботів.

PyTelegramBotAPI (Telebot) було обрано як основний фреймворк для взаємодії з Telegram Bot API. Ця бібліотека є високопродуктивною, зручною у використанні та дозволяє легко обробляти вхідні повідомлення, команди, файли (такі як фотографії) та формувати відповіді. Вона надає необхідні інструменти для реалізації логіки діалогів, керування станами користувачів та інтеграції з функціоналом Telegram, таким як inline-клавіатури та кнопки.

SQLite забезпечує зручну та ефективну роботу з базою даних, дозволяючи зберігати структуровані дані без необхідності розгортання окремого серверу баз даних. Його "безсерверна" природа (база даних зберігається у єдиному файлі main.sql) значно спрощує розгортання,

тестування та перенесення бота. SQLite ідеально підходить для даного проєкту, оскільки він дозволяє легко реалізувати CRUD-операції для управління профілями користувачів, анкетами тварин, записами про лайки, запити та репорти.

Додатково, для роботи з файловою системою (збереження фотографій анкет) використовуються вбудовані модулі Python, такі як os та shutil. Це дозволяє ефективно керувати медіафайлами, які пов'язані з анкетами, зберігаючи їх на диску та посилаючись на них з бази даних.

Загалом, обрані інструменти створюють надійну основу для реалізації ефективної, безпечної та масштабованої системи чат-бота, орієнтованого на кінцевих користувачів Telegram.

Python 3 обрано як основну мову програмування для розробки чат-бота. Python є сучасною, високопродуктивною та кросплатформенною мовою, що дозволяє створювати масштабовані застосунки. Його модульна структура, підтримка численних бібліотек та легка інтеграція з іншими технологіями робить його ідеальним вибором для створення системи з чітким розділенням відповідальностей. У рамках цього проєкту Python забезпечує надійну логіку для обробки запитів користувачів, управління даними та реалізації бізнес-логіки бота.

PyTelegramBotAPI (Telebot) обрано як бібліотеку для взаємодії з Telegram API. Telebot спрощує процес розробки бота, надаючи зручний інтерфейс для:

- **Обробки повідомлень:** Легке розпізнавання команд та текстових повідомлень від користувачів.
- **Керування станами:** Реалізація логіки діалогів через user_states, що дозволяє боту "пам'ятати" контекст розмови.

- **Надсилання медіа:** Простота надсилання фотографій, документів та інших файлів.
- **Використання клавіатур:** Зручне створення кнопок та inline-клавіатур для інтерактивної взаємодії.
- **Обробка зворотних викликів:** Ефективна обробка натискань на кнопки та інших взаємодій.

SQLite виступає як основна система керування базами даних для проєкту. Це легка, вбудована та безсерверна платформа для зберігання даних. Вона забезпечує підтримку простих запитів та транзакцій, що є критично важливим для обробки профілів, лайків та репортів. Завдяки простоті розгортання та відсутності необхідності налаштування серверу, SQLite ідеально підходить для даного застосунку. Хоча SQLite не надає розширених функцій безпеки, як корпоративні СУБД, для бота з прямою взаємодією через API та без публічного доступу до БД це є прийнятним рішенням за умови належного контролю доступу до файлу main.sql.

Використання **Python 3, PyTelegramBotAPI та SQLite** забезпечує надійну технологічну основу для створення безпечного, масштабованого та зручного чат-бота. Усі ці інструменти добре інтегруються між собою та відповідають сучасним стандартам розробки. Вони дозволяють ефективно реалізувати функціонал бота, з дотриманням принципів модульної архітектури та захисту даних (наскільки це можливо для вбудованої БД).

Завдяки цьому стеку технологій, система здатна ефективно обслуговувати значну кількість користувачів, витримувати зростання функціональності та забезпечувати стабільну роботу в умовах реального навантаження. Такий підхід дозволяє легко масштабувати рішення, адаптуючи його до змін бізнес-вимог та зростання обсягу взаємодій.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Успішна робота чат-бота для пошуку нових власників тварин значною мірою залежить від відповідності апаратного та програмного забезпечення певним вимогам. Ці вимоги гарантують стабільність, безпеку, швидкодію та безперебійну роботу бота, забезпечуючи зручну взаємодію для всіх користувачів.

4.1.1 Апаратні вимоги

Чат-бот, розроблений на Python з використанням PyTelegramBotAPI та SQLite, має відносно низькі вимоги до апаратного забезпечення, що робить його гнучким у розгортанні.

- **Для розгортання та роботи бота (серверна частина):**
  - **Процесор (CPU):** Мінімально - 1 ядро, рекомендовано - 2 ядра або більше (наприклад, Intel Core i3 або аналогічний AMD Ryzen). Цього буде достатньо для обробки паралельних запитів від кількох користувачів.
  - **Оперативна пам'ять (RAM):** Мінімально - 1 ГБ, рекомендовано - 2 ГБ або більше. Python-додаток та бібліотеки потребують певної кількості пам'яті для ефективної роботи.
  - **Дисковий простір:** Мінімально - 1 ГБ вільного місця. Цього достатньо для зберігання файлів бота, бази даних SQLite (main.sql) та папки для фотографій анкет (photos/). З ростом кількості анкет та фотографій цей обсяг може збільшуватися. Рекомендовано використовувати SSD-накопичувачі для кращої швидкодії доступу до файлів.
  - **Підключення до Інтернету:** Стабільне та надійне підключення з пропускною здатністю не менше 10 Мбіт/с. Це критично важливо для

безперебійного обміну даними з серверами Telegram API в режимі реального часу.

- **Електроживлення:** Стабільне джерело живлення, бажано з джерелом безперебійного живлення (ДБЖ) для серверів, що забезпечить безперервну роботу.

- **Для адміністративної роботи (запуск локально або моніторинг):**
  - Будь-який сучасний комп'ютер або ноутбук (відповідний для офісної роботи) з доступом до інтернету буде достатнім.

4.1.2 Програмні вимоги

Програмне забезпечення для чат-бота повинно забезпечити середовище для виконання Python-коду, а також доступ до необхідних бібліотек.

- **Операційна система:**
  - **Для розгортання бота (серверна частина):** Будь-яка сучасна серверна версія Linux (наприклад, Ubuntu Server, Debian), Windows Server або macOS. Linux є кращим вибором через легкість розгортання, стабільність та низьке споживання ресурсів.
  - **Для розробки:** Будь-яка сучасна версія Windows, macOS або Linux, що підтримує встановлення Python 3.
- **Середовище виконання:**
  - **Python 3.8+:** Рекомендується використовувати останню стабільну версію Python 3 для доступу до нових функцій та оптимізацій.
- **Необхідні Python-бібліотеки:**
  - **pytelegrambotapi (telebot):** Бібліотека для взаємодії з Telegram Bot API. Встановлюється через pip install pytelegrambotapi.
  - **sqlite3:** Вбудований модуль Python для роботи з базою даних SQLite. Не потребує додаткового встановлення.

- **datetime, os, shutil, random, re, enum:** Стандартні модулі Python, які використовуються для обробки дати/часу, роботи з файловою системою, генерації випадкових чисел, регулярних виразів та перелічень. Не потребують додаткового встановлення.

- **Інші вимоги:**

- **Доступ до мережі:** Дозволити вихідні з'єднання на порти Telegram Bot API (зазвичай 443 для HTTPS).

- **Програмне забезпечення для розробки:**

  - **Інтегроване середовище розробки (IDE):** PyCharm, VS Code, Sublime Text або інше для написання та налагодження коду.

  - **Система контролю версій:** Git (для керування версіями коду).

Ці апаратні та програмні компоненти обрано з метою формування стабільного, безпечного та зручного робочого середовища для чат-бота. Вони дозволяють ефективно обробляти запити користувачів, керувати даними та забезпечувати стабільну роботу системи в реальному часі. Такий підхід гарантує цілісність, конфіденційність та доступність критичних даних, що є ключовими чинниками для безперервної та ефективної роботи бота.

4.2 Тестування системи

Тестування чат-бота для пошуку нових власників тварин є критично важливим етапом розробки, який забезпечує коректну роботу всіх функцій, зручність взаємодії з користувачем та відповідність заданим вимогам. Тестування проводилося шляхом імітації дій кінцевого користувача в середовищі Telegram, перевіряючи основні сценарії використання бота.

4.2.1 Перевірка основних функцій

1. **Початковий екран та Реєстрація:**

- Після запуску бота командою /start користувач потрапляє на початковий екран. Бот пропонує варіанти дій: "Створити анкету" або "Мій профіль".

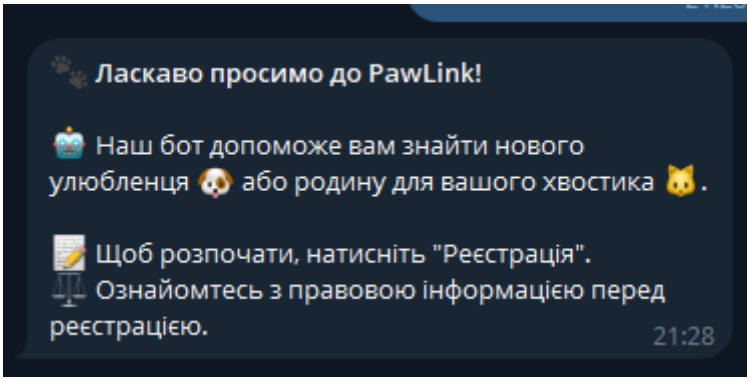

Рис. 10 Стартове меню після запуску бота

- Якщо користувач не зареєстрований, він має можливість перейти до реєстрації через кнопку "Створити анкету". Бот послідовно запитує необхідну інформацію: Telegram ID (автоматично), тип акаунта (власник чи шукач), ім'я, вік, номер телефону, опис.

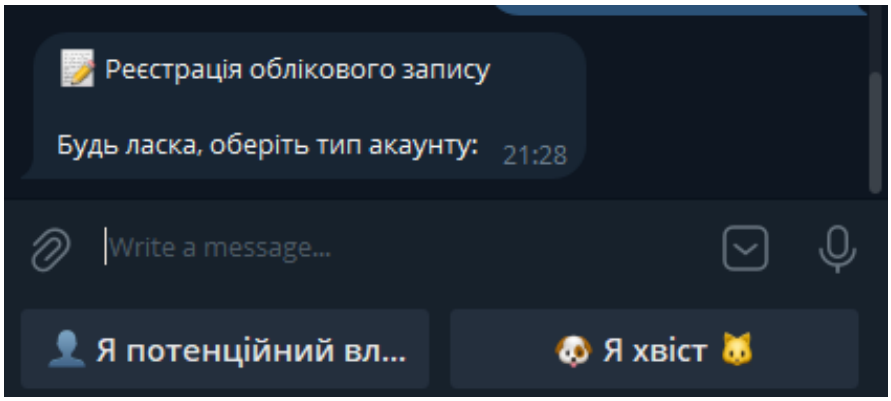

Рис. 11 Початок реєстрації акаунта

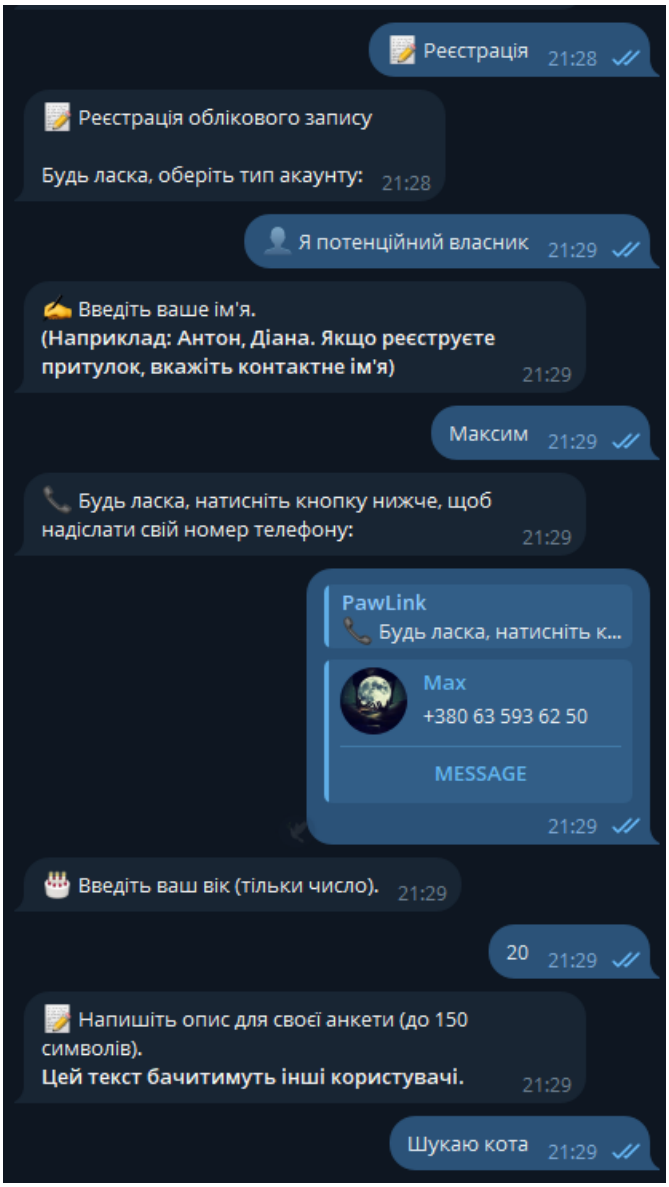

Рис. 12 Реєстрація як потенційного власника

- Перевіряється коректність збереження цих даних у базі даних SQLite.

2. **Додавання фотографій до анкети:**

- Після введення текстових даних, бот просить додати фотографії (1-3 фото). Перевіряється коректність завантаження фотографій та їх збереження у відповідній папці файлового сховища, а також збереження шляхів до фото у базі даних.

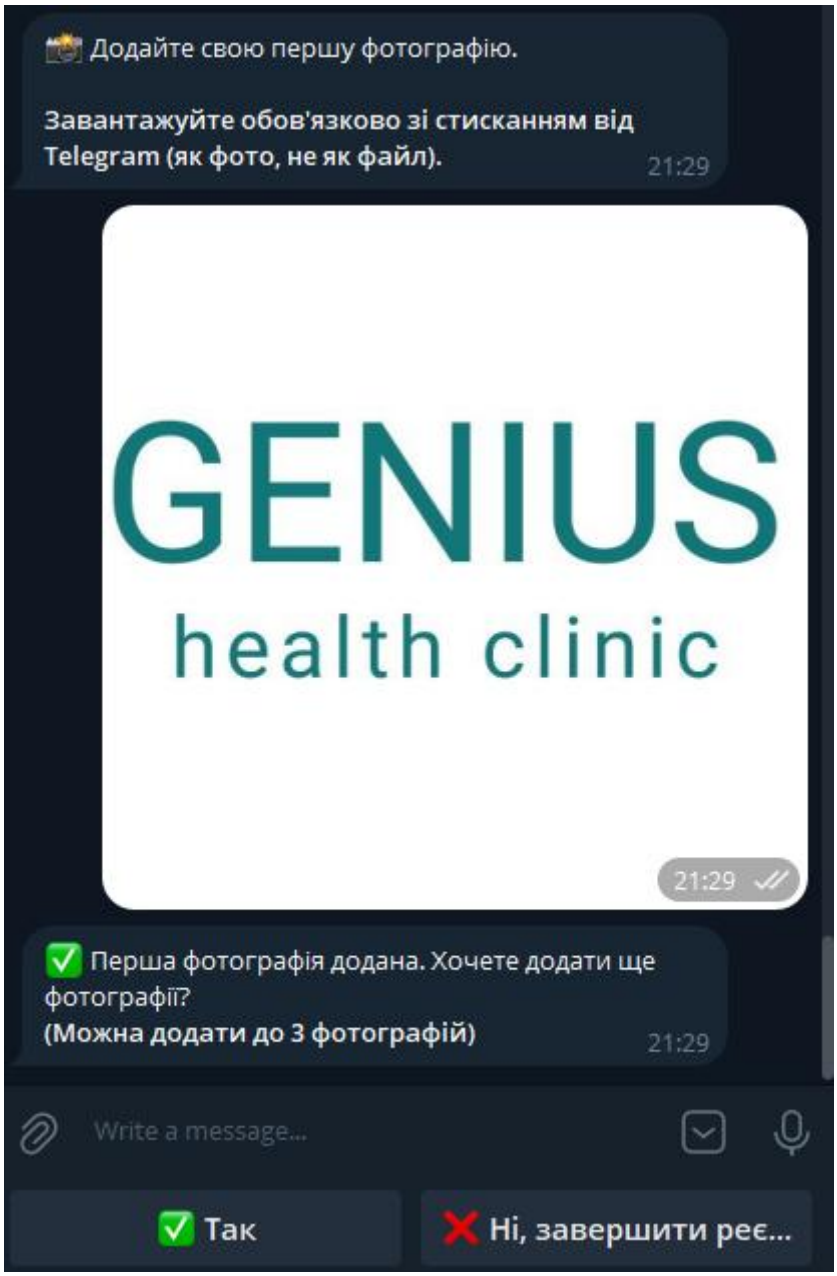

Рис. 13 Додавання фотографій при реєстрації

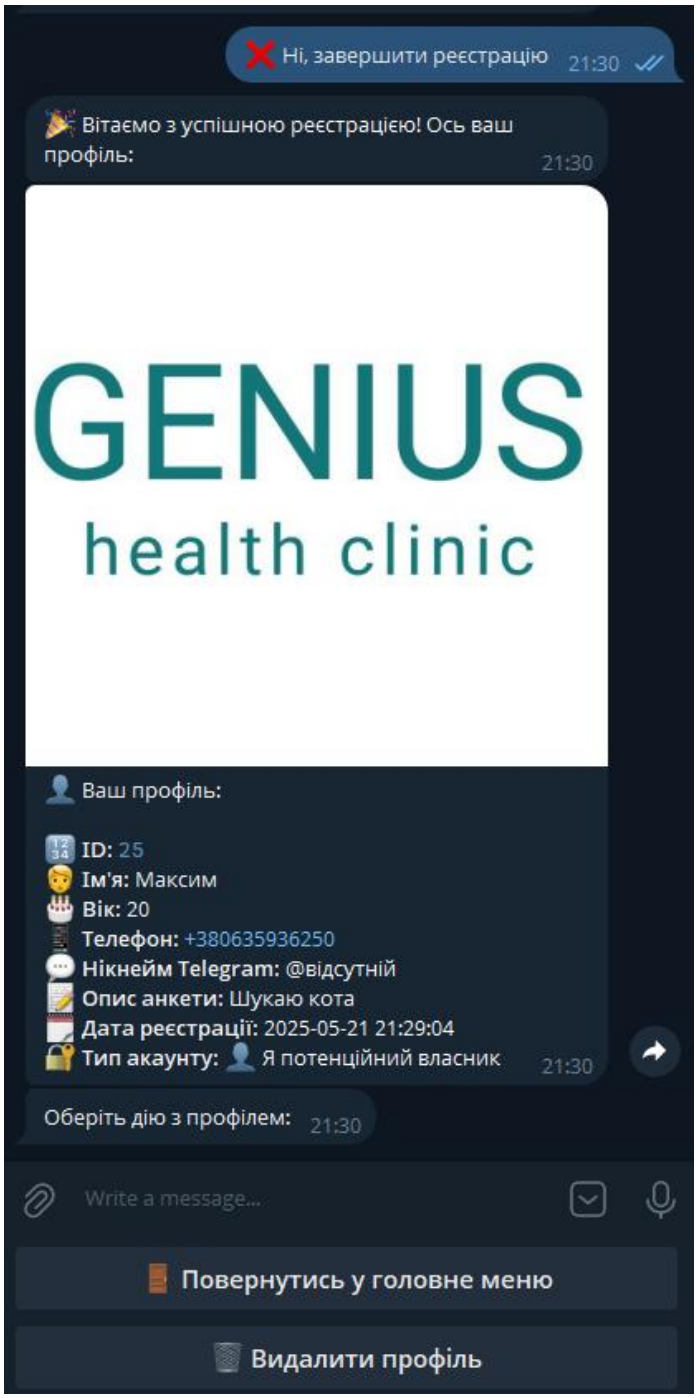

Рис. 14 Закінчення реєстрації та вивід профіля користувача

3. **Перегляд та редагування профілю:**

- Користувач може перейти до "Мій профіль" для перегляду своєї анкети.

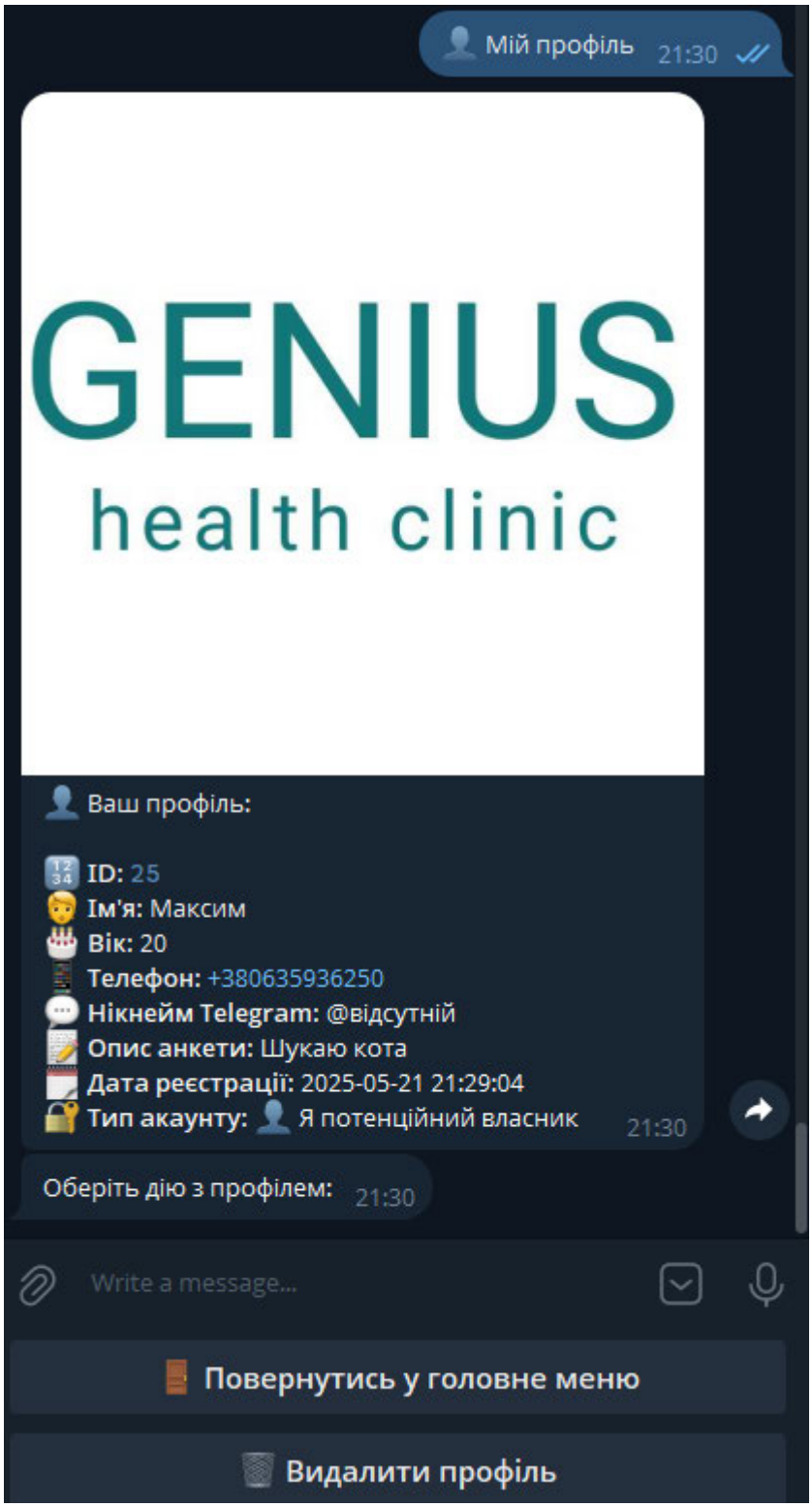

Рис. 15 Перегляд профілю користувача

- Функція видалення профілю та очищення пов'язаних даних (з БД та файлів) також перевіряється.

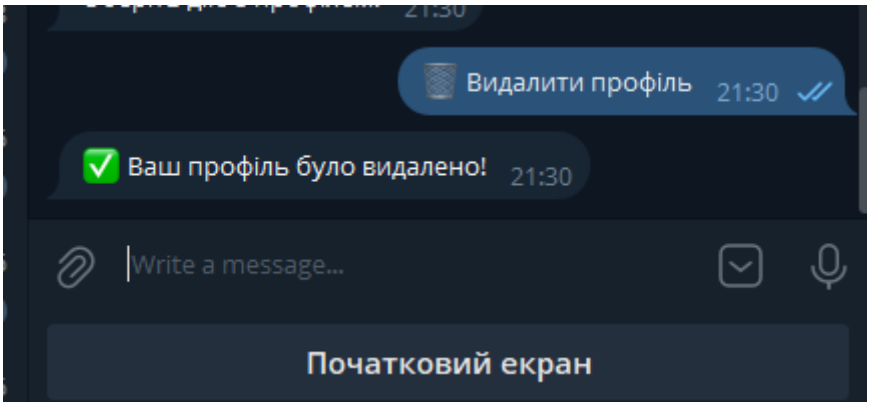

Рис. 16 Видалення профілю користувача

4. **Пошук профілів (анкет):**

- Користувач може ініціювати пошук анкет (тварин або власників) за допомогою кнопки "Пошук анкет".
- Бот пропонує фільтри: тип тварини (собака, кішка тощо), вік, стать.
- Перевіряється коректність застосування фільтрів та відображення відповідних анкет.

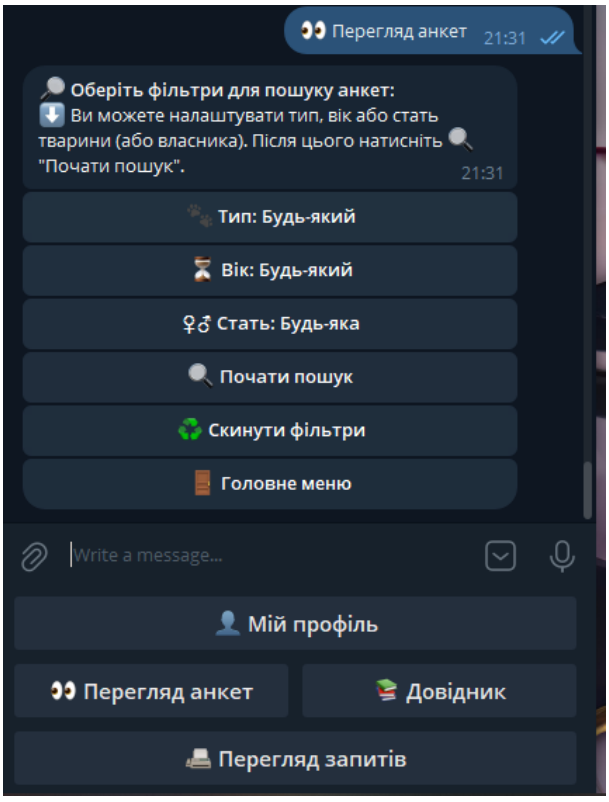

Рис. 17 Головне меню з фільтрами для пошуку анкет

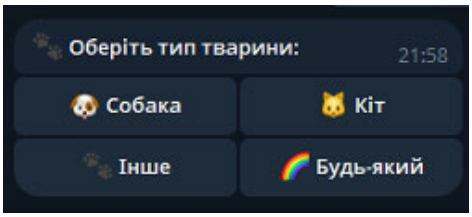

Рис. 18 Меню обрання типу тварини

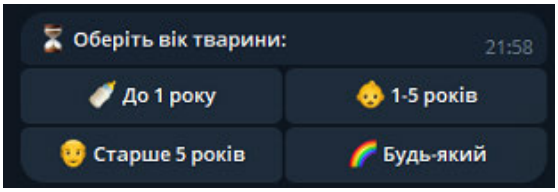

Рис. 20 Меню обрання типу тварини

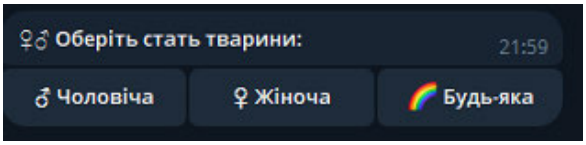

Рис. 21 Меню обрання статі тварини

Рис. 22 Пошук анкет

5. **Взаємодія з анкетами (Вподобайки, Запити, Репорти):**

- Під час перегляду анкет користувач може поставити "Лайк", "Надіслати запит" або "Поскаржитись".

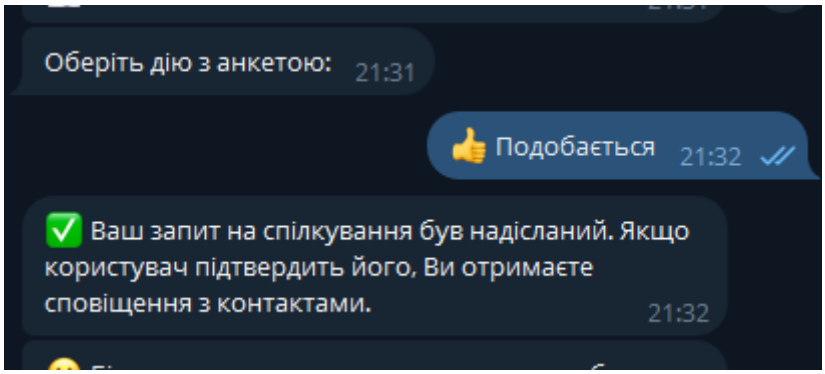

Рис. 23 Механізм відправки вподобайки

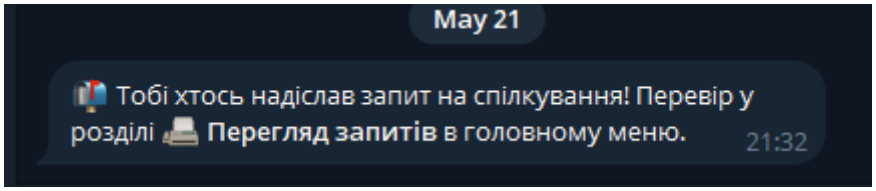

Рис. 24 Механізм отримання вподобайки від іншого користувача

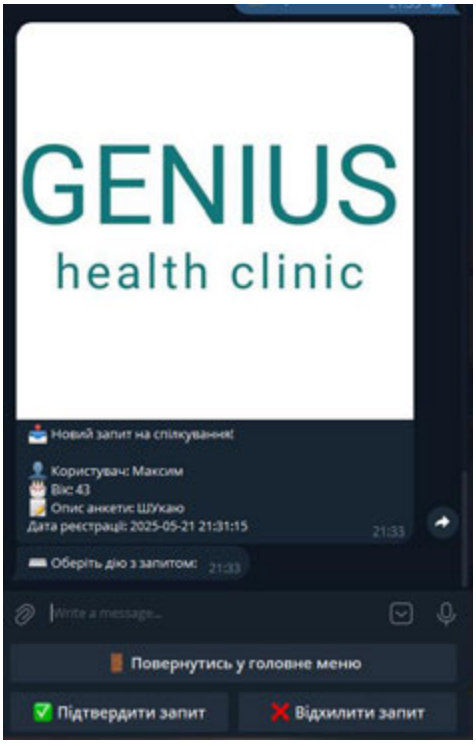

Рис. 25 Інтерфейс обробки запиту для користувачів

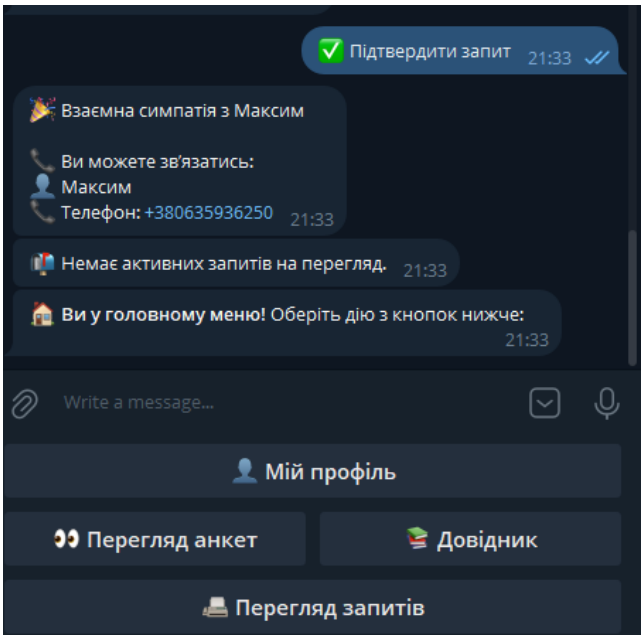

Рис. 26 Інтерфейс підтвердження

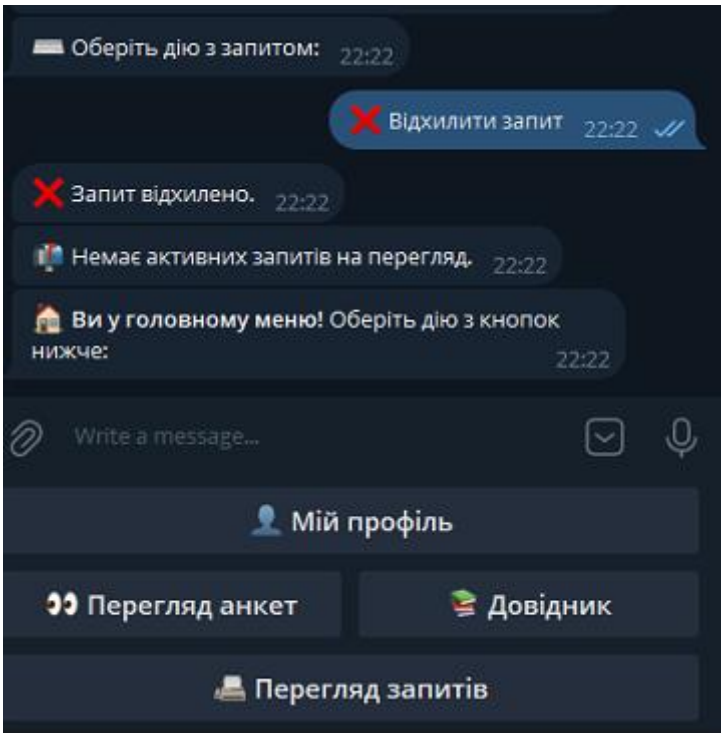

Рис. 27 Інтерфейс відхилення

- Перевіряється коректність запису цих дій у базу даних.
- Тестується логіка взаємного "матчу": якщо обидва користувачі надсилають "Запит", бот має відкрити їхні контакти та сповістити про матч.
- Перевіряється функція подання репортів та їх реєстрація для подальшої модерації.

4.2.2 Адміністрування та модерація:

Хоча бот не має окремої веб-панелі адміністратора, модерація може здійснюватися через прямий доступ до бази даних.

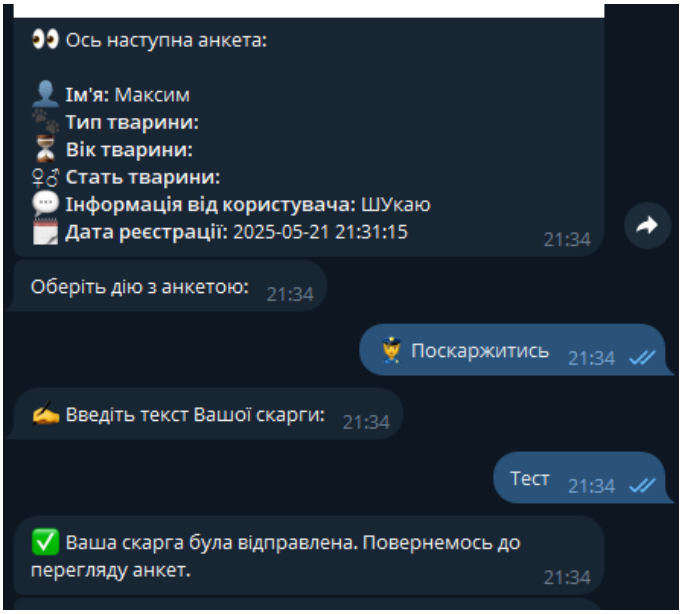

Рис. 28 Інтерфейс відправлення скарги

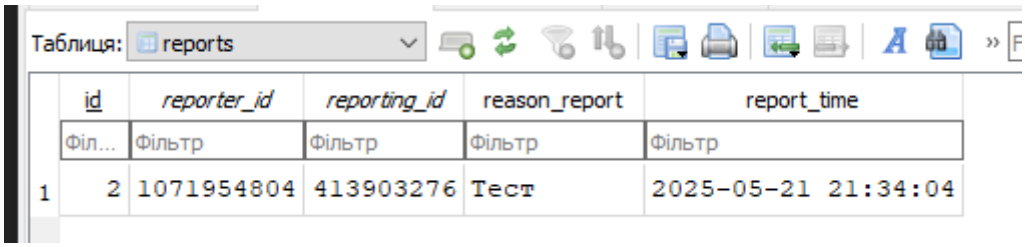

Рис. 29 Підтвердження оформлення скарги в базі даних

- **Перегляд скарги:** Модератор може переглядати подані користувачами репорти, аналізувати їхній зміст.
- **Блокування користувачів:** Перевіряється можливість блокування користувачів або окремих анкет за їх ID, що приховує їх від загального пошуку.

4.2.3 Тестування на стійкість та обробку помилок:

- **Некоректний ввід:** Перевіряється реакція бота на некоректний ввід даних (наприклад, введення тексту замість числа для віку, завантаження

неграфічних файлів як фото). Бот має коректно обробляти такі ситуації та надавати зрозумілі повідомлення про помилку.

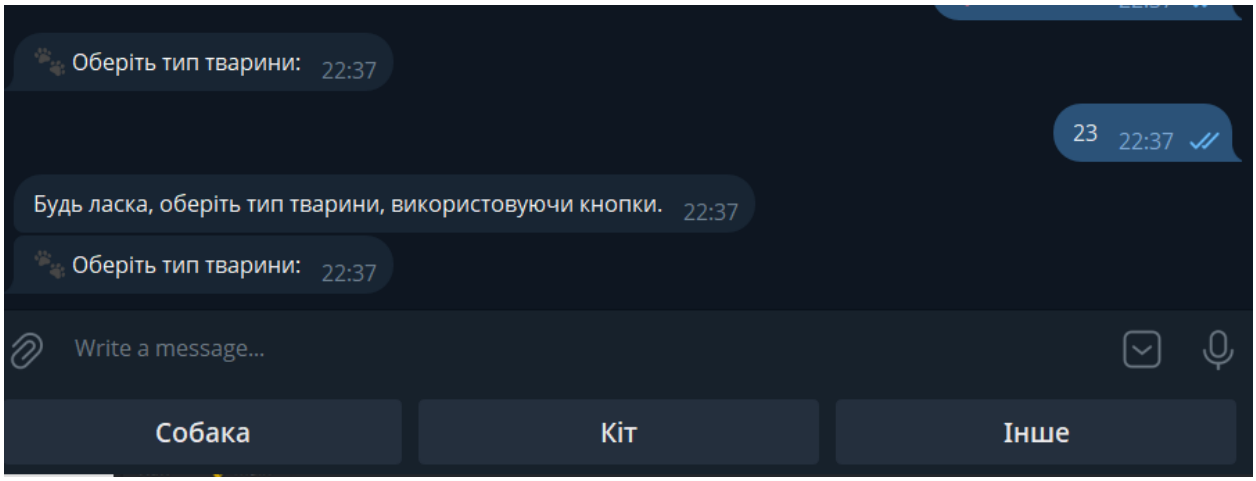

Рис. 31 Некоректний ввід типу тварини

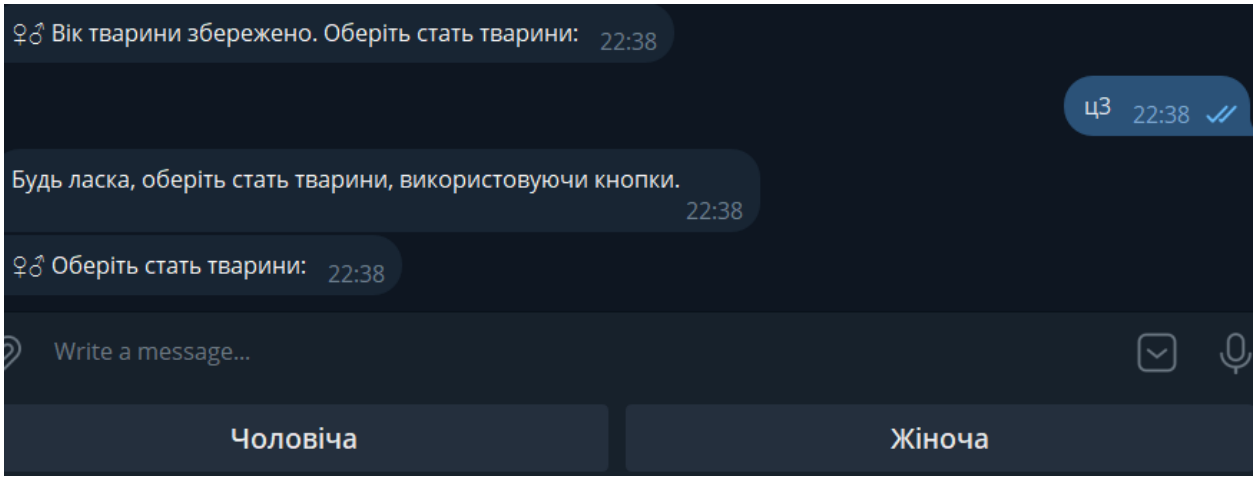

Рис. 31 Некоректний ввід статі тварини

- **Відсутність підключення до БД:** Перевіряється поведінка бота у випадку, якщо файл бази даних пошкоджений або недоступний.

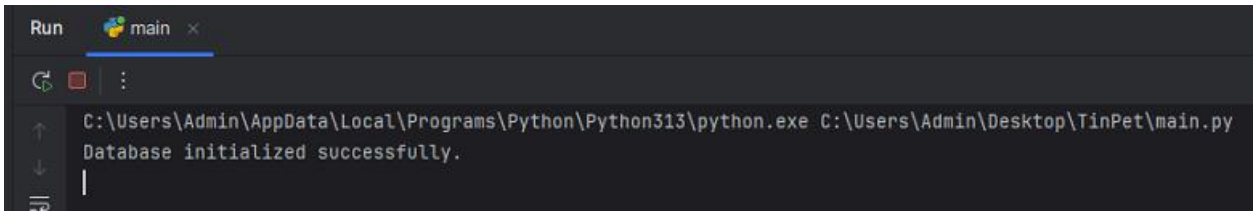

Рис. 32 Повідомлення в консолі про наявність підключення до бд

4.2.4 Результати тестування:

У процесі тестування було виявлено, що чат-бот коректно виконує всі заявлені функції, забезпечує логічну послідовність діалогів та ефективно взаємодіє з базою даних SQLite та файловим сховищем. Система продемонструвала стабільність під час нормального використання та здатність обробляти типові помилки вводу. Деякі дрібні недоліки, виявлені під час тестування, були усунені.

Таким чином, розроблена система чат-бота відповідає поставленим вимогам і готова до використання.

ВИСНОВКИ

Розробка чат-бота для пошуку нових власників тварин вирішує актуальну потребу у зручному та ефективному механізмі зв'язку між тими, хто шукає тварину, і тими, хто шукає для неї дім. Система покликана автоматизувати ключові процеси, зокрема створення та управління анкетами тварин та потенційних власників, механізм взаємних "вподобайок", а також базову модерацію. Це дозволяє значно підвищити ефективність пошуку, зменшити часові затрати на традиційні методи та забезпечити зручну взаємодію в звичному онлайн-середовищі Telegram.

Запропоноване рішення, створене з використанням **Python 3** та сучасної бібліотеки **PyTelegramBotAPI (Telebot)**, у поєднанні з легкою вбудованою базою даних **SQLite**, забезпечує стабільну та гнучку роботу чат-бота. Використання SQLite спрощує взаємодію з базою даних, а продумана логіка керування станами користувачів гарантує послідовну та інтуїтивно зрозумілу взаємодію. Інтуїтивно зрозумілий інтерфейс взаємодії через Telegram, реалізований з урахуванням можливостей платформи, забезпечує зручний доступ до функціоналу як для опікунів тварин, так і для тих, хто бажає їх прихистити.

Проведений аналіз існуючих підходів до пошуку власників тварин засвідчив необхідність створення спеціалізованої системи, яка б об'єднала можливості зручної реєстрації, фільтрації за параметрами та інтерактивної взаємодії. Розроблений чат-бот відповідає цим вимогам, забезпечуючи централізоване зберігання інформації про анкети, підтримку взаємних "метчів" та базові інструменти для модерації контенту.

Окрім виконання функціональних завдань, система сприяє модернізації процесу пошуку нових власників, переходячи від застарілих та розрізнених методів до централізованого, цифрового рішення. Це сприяє підвищенню

прозорості процесів, покращенню комунікації між сторонами та оптимізації пошуку. Зокрема, функціонал подання репортів та модерації контенту допомагає підтримувати безпечне середовище для користувачів.

У перспективі система пропонує можливості для подальшого розвитку – впровадження розширених фільтрів пошуку, інтеграції з зовнішніми сервісами (наприклад, для верифікації користувачів або модерації фотографій), розширеної статистики та аналітики взаємодій, а також потенційне впровадження елементів штучного інтелекту для більш точного підбору "матчів". Таким чином, розроблене рішення не тільки задовольняє поточні потреби у пошуку нових власників тварин, але й формує основу для стратегічного розвитку подібних соціальних проєктів у цифровому просторі.

Підсумовуючи, система чат-бота, створена в рамках цієї бакалаврської роботи, повністю відповідає поставленим цілям і демонструє високу ефективність у вирішенні завдань пошуку нових власників тварин. Вона є вагомим кроком на шляху до автоматизації та цифровізації процесів, пов'язаних з прилаштуванням тварин, і може бути використана як основа для подальших розробок у сфері соціально орієнтованих чат-ботів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Microsoft. ASP.NET Core Documentation – [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/aspnet/core
2. The Clean Architecture – [Електронний ресурс] – Режим доступу: https://medium.com/clean-code-channel/clean-architecture-the-solution-to-have-a-reusable-flexible-and-testable-code-ac7e296d1a75
3. Типи архітектури програмного забезпечення – [Електронний ресурс] – Режим доступу: https://medium.com/nuances-of-programming/4-типа-архитектуры-программного-обеспечения-917133174724
4. What is Unified Modeling Language (UML)? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
5. ПРОЄКТУВАННЯ ER-ДІАГРАМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ru/skills/proektirovanie-er-diagrammy/
6. Діаграма варіантів використання (UseCase diagram) – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ru/fd_use-case-diagram.html
7. ПРОЄКТУВАННЯ USE CASE ДІАГРАМИ. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ru/skills/proektirovanie-use-case-diagrammy-opredelenie-funktsionalnykh-vozmozhnostey-sistemy/
8. What is Sequence Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
9. UML - Activity Diagrams – Tutorialspoint – [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm
10. Побудова діаграми класів – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ru/gpg_class-diagram.html
11. UML-діаграми класів – [Електронний ресурс] – Режим доступу: https://prog-cpp.ru/uml-classes/
12. Entity Relationship Diagram - Data Modeling – [Електронний ресурс] – Режим доступу: https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html

13. UML 2 Tutorial - Package Diagram - Sparx Systems – [Електронний ресурс] – Режим доступу: https://sparxsystems.com/resources/tutorials/uml2/package-diagram.html
14. Microsoft Docs. (2021). Layered architecture pattern – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/azure/architecture/patterns/layered
15. What is Component Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/
16. Deployment Diagram in UML: Definition, Examples & Components – [Електронний ресурс] – Режим доступу: https://study.com/academy/lesson/deployment-diagram-in-uml-definition-examples-components.html
17. What is a data flow diagram? – [Електронний ресурс] – Режим доступу: https://www.lucidchart.com/pages/data-flow-diagram
18. OpenCart Documentation – [Електронний ресурс] – Режим доступу: https://docs.opencart.com
19. PrestaShop Developer Guide – [Електронний ресурс] – Режим доступу: https://devdocs.prestashop-project.org
20. Shopify Help Center – [Електронний ресурс] – Режим доступу: https://help.shopify.com
21. Соммервіль, І. (2016). Інженерія програмного забезпечення (10-е вид.). Pearson Education.
22. Прессман Р. С. та Максим Б. Р. (2015). Software Engineering: A Practitioner's Approach (8-е видання). Освіта McGraw-Hill.
23. Пун, А., Лоу, К. (2017). «Приплив електронної комерції в індустрію гостинності: дослідження онлайн-туристичних агентств». Міжнародний журнал досліджень туризму, 19(6), 703–710.
24. Коннолі, Т., і Бегг, К. (2014). Системи баз даних: практичний підхід до проєктування, впровадження та управління (6-е видання). Pearson Education.

25. Фріман, Е., Робсон, Е. (2021). Head First Design Patterns (2-е видання). O'Reilly Media.
26. Корпорація Microsoft. (2023). Документація Microsoft SQL Server. [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/sql
27. Ріхтер, Дж. (2012). CLR через C# (4-е видання). Microsoft Press.
28. Альбахарі, Дж., і Альбахарі, Б. (2021). C# 10 in a Nutshell (8-е видання). O'Reilly Media.
29. Буч, Г., Рамбо, Дж., і Якобсон, І. (2005). Посібник користувача з уніфікованої мови моделювання (2-е видання). Аддісон-Уеслі.
30. Якобсон І., Крістерсон М., Йонссон П. та Овергаард Г. (1992). Об'єктно-орієнтоване програмне забезпечення: підхід, орієнтований на використання. Аддісон-Уеслі.
31. Васильєв, А. Н. (2020). «Автоматизація кадрового діловодства в державних установах». Журнал інформаційних систем та державного управління, 7(2), 122–130.
32. Парсонс, Д. та Оджа, Д. (2016). Нові погляди на комп'ютерні концепції 2016. Cengage Learning.
33. Беннетт С., МакРобб С. та Фармер Р. (2010). Об'єктно-орієнтований системний аналіз і проєктування з використанням UML (4-е видання). Макгроу-Хілл
34. Telegram Bot API. (2024). *Офіційна документація Telegram Bot API*. Отримано з: https://core.telegram.org/bots/api.
35. SQLite Documentation. (2024). *SQLite — офіційна документація*. Отримано з: https://www.sqlite.org/docs.html.
36. Бочкарьов С. В. (2018). *Основи моделювання програмного забезпечення з використанням UML*. Харків: ХНУРЕ.

ДОДАТОК А

Ініціалізація та структура бази даних

```
def init_db():
    conn = None
    try:
        conn = sqlite3.connect('main.sql')
        cur = conn.cursor()
        cur.execute('''CREATE TABLE IF NOT EXISTS users (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    user_id INTEGER UNIQUE,
                    account_type TEXT,
                    name TEXT,
                    age INTEGER,
                    phone TEXT,
                    description TEXT,
                    photo_count INTEGER DEFAULT 0,
                    last_active_time DATETIME,
                    registration_time DATETIME
                )''')
```

```
    cur.execute('''CREATE TABLE IF NOT EXISTS likes (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    liker_id INTEGER,
                    liked_id INTEGER,
                    timestamp DATETIME
                )''')
    cur.execute('''CREATE TABLE IF NOT EXISTS reports (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    reporter_id INTEGER,
                    reporting_id INTEGER,
                    context TEXT,
                    timestamp DATETIME
                )''')
    conn.commit()
except sqlite3.Error as e:
    print(f"Помилка при роботі з базою даних: {e}")
finally:
    if conn:
        conn.close()
```

ДОДАТОК Б

Отримання та відображення профілів (пошук анкет)

```
def get_next_profile_to_view(current_user_id):
    conn = sqlite3.connect('main.sql')
    cur = conn.cursor()

    cur.execute('SELECT account_type FROM users WHERE user_id = ?', (current_user_id,))
    current_user_type = cur.fetchone()[0]

    opposite_account_type = 'human' if current_user_type == 'animal' else 'animal'

    viewed_profiles = user_profile_view_state.get(current_user_id, [])

    # Filter for the opposite account type, and not liked or reported by the current user
    # and not already liked the current user
    # and not already viewed
    query = """
```

```
        SELECT user_id, name, age, description, photo_count

        FROM users

        WHERE account_type = ?

        AND user_id != ?

        AND user_id NOT IN (SELECT liked_id FROM likes WHERE liker_id
= ?)

        AND user_id NOT IN (SELECT reporting_id FROM reports WHERE
reporter_id = ?)

        AND user_id NOT IN (SELECT liker_id FROM likes WHERE liked_id
= ?)

    """

    params = [opposite_account_type, current_user_id, current_user_id,
current_user_id, current_user_id]



    # Apply filters if they exist for the current user

    filters = user_filters.get(current_user_id, {})

    if 'type' in filters and filters['type'] != 'Будь-який':

        query += " AND name = ?" # Assuming 'name' stores animal type for
'animal' profiles

        params.append(filters['type'])

    if 'age' in filters and filters['age'] != 'Будь-який':
```

```
        if filters['age'] == 'До 1 року':
            query += " AND age < 1"
        elif filters['age'] == '1-3 роки':
            query += " AND age >= 1 AND age <= 3"
        elif filters['age'] == '3-7 років':
            query += " AND age >= 3 AND age <= 7"
        elif filters['age'] == 'Більше 7 років':
            query += " AND age > 7"
    if 'gender' in filters and filters['gender'] != 'Будь-яка':
        # Assuming gender is part of description or a separate field, adjust as needed
        # For this example, let's assume it's in the description for simplicity, or we need a 'gender' column
        pass # No explicit gender filter in the provided schema, would need a 'gender' column in 'users' table

    query += " ORDER BY RANDOM()"

    cur.execute(query, params)
    available_profiles = cur.fetchall()
```

```
    for profile in available_profiles:
        if profile[0] not in viewed_profiles:
            conn.close()
            return profile

    conn.close()
    return None # No new profiles found


def show_profile(chat_id, profile_data, back_button=False):
    if profile_data:
        profile_user_id, name, age, description, photo_count = profile_data

        caption = f"*{name}*, {age} років\n\n{description}"

        photo_path = None
        if photo_count > 0:
```

```
        photo_path = f'photos/{profile_user_id}/photo_1.jpg' # Assume photo_1.jpg is the main photo


    if photo_path and os.path.exists(photo_path):
        with open(photo_path, 'rb') as photo:
            markup = types.InlineKeyboardMarkup()
            like_button = types.InlineKeyboardButton("❤️ Подобається", callback_data=f"like_{profile_user_id}")
            next_button = types.InlineKeyboardButton("➡️ Далі", callback_data="next_profile")
            report_button = types.InlineKeyboardButton("❗ Поскаржитись", callback_data=f"report_{profile_user_id}")

            if back_button:
                back_btn = types.InlineKeyboardButton("⬅️ Назад", callback_data="prev_profile")
                markup.row(back_btn, like_button, next_button)
            else:
                markup.row(like_button, next_button)
```

```
            markup.row(report_button)

            bot.send_photo(chat_id, photo, caption=caption, reply_markup=markup, parse_mode='Markdown')

        else:

            markup = types.InlineKeyboardMarkup()

            like_button = types.InlineKeyboardButton("💔 Подобається", callback_data=f"like_{profile_user_id}")

            next_button = types.InlineKeyboardButton("➡️ Далі", callback_data="next_profile")

            report_button = types.InlineKeyboardButton("❗ Поскаржитись", callback_data=f"report_{profile_user_id}")


            if back_button:

                back_btn = types.InlineKeyboardButton("⬅️ Назад", callback_data="prev_profile")

                markup.row(back_btn, like_button, next_button)

            else:

                markup.row(like_button, next_button)


            markup.row(report_button)
```

```
            bot.send_message(chat_id, caption, reply_markup=markup, parse_mode='Markdown')
        else:
            markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
            btn_back_to_main = types.KeyboardButton('На головну')
            markup.add(btn_back_to_main)
            bot.send_message(chat_id, "На жаль, нових анкет за вашими критеріями не знайдено. Спробуйте змінити фільтри або зачекайте на нові профілі.", reply_markup=markup)
            user_states[chat_id] = BotState.REGISTRATION_COMPLETED
```