НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних систем, мереж та кібербезпеки**
_____**Ігор БОЛБОТ**	_____**Дмитро КАСАТКІН**
(підпис)	(підпис)
«____» _____2026 р.	«____» _____2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Спеціалізований додаток для власників тварин»

Спеціальність «122 Комп’ютерні науки»
Освітньо-професійна програма «Комп’ютерні науки»

Гарант освітньої програми
д.е.н., професор _____ **Роман РУДЕНСЬКИЙ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи _____ **Володимир НАЗАРЕНКО**
(підпис)

Виконав _____ **Артур ПРИЛУЦЬКИЙ**
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри компю'терних систем, мереж та кібербезпеки

к.т.н., доцент _____Дмитро КАСАТКІН

(підпис)

«____» _____2026 р.

ЗАВДАННЯ ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Прилуцькому Артуру Валентиновичу

(прізвище, ім'я, по батькові)

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки» Тема бакалаврської кваліфікаційної роботи

« Спеціалізований додаток для власників тварин » затверджена наказом від «06» січня 2026 р.

№ 46 «С» Термін подання завершеної роботи на кафедру «_____» _____2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи: спільний вигул, оперативний обмін інформацією, геопросторові сервіси.

Перелік питань, що підлягають дослідженню:

1) Актуальність сервісів для власників тварин та аналіз існуючих аналогів.
2) Проєктування архітектури системи та реляційної моделі бази даних.
3) Програмна реалізація мобільного клієнта та серверного API системи.
4) Розробка та оптимізація алгоритму гіперлокального підбору анкет.
5) Реалізація аналітичного модуля та автоматизованої генерації PDF-звітів.
6) Функціональне тестування інтерфейсу додатка та верифікація API.

Дата видачі завдання «_____» _____2026 р.

Керівник бакалаврської кваліфікаційної роботи _____ **Володимир НАЗАРЕНКО**

(підпис)

Завдання взяв до виконання _____ **Артур ПРИЛУЦЬКИЙ**

(підпис)

Календарний план

№ з/п	Назва етапів виконання бакалаврської кваліфікаційної роботи	Строк виконання етапів бакалаврської кваліфікаційної роботи	Примітка
1	Отримання завдання, формування плану роботи	03.02.2026-07.02.2026	Виконано
2	Аналіз предметної області та джерел	08.02.2026-28.02.2026	Виконано
3	Формування вимог і моделювання системи	01.03.2026-15.03.2026	Виконано
4	Проєктування структури даних та архітектури застосунку	16.03.2026-30.03.2026	Виконано
5	Розробка основних програмних модулів	31.03.2026-30.04.2026	Виконано
6	Реалізація аналітики, сповіщень, нотаток і налаштувань	01.05.2026-10.05.2026	Виконано
7	Тестування та доопрацювання застосунку	11.05.2026-18.05.2026	Виконано
8	Оформлення пояснювальної записки та подання роботи	19.05.2026-25.05.2026	Подано на кафедру

Студент _____ Прилуцький А.В._____
(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи
_____ Назаренко В.А.__
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

В нинішні часи домашні тварини давно перестали бути просто «улюбленцями», ставши повноправними членами родин та джерелом психологічної підтримки для своїх власників. Проте події, що розпочалися 24 лютого 2022 року з повномасштабним вторгненням РФ на територію України, докорінно змінили життя не лише людей, а й їхніх чотирилапих друзів, створивши низку нових, гострих проблем, які потребують технологічного вирішення.

В умовах воєнного стану та масової вимушеної міграції (як внутрішньої, так і зовнішньої) власники тварин зіткнулися з безпрецедентними викликами. Люди, що переїхали до нових міст, часто опиняються в інформаційному вакуумі: вони не знають локацій для безпечного вигулу, розташування працюючих ветеринарних клінік, зоомагазинів чи бомбосховищ, куди пускають з тваринами. Втрата звичних соціальних зв'язків ускладнює отримання побутових порад або екстреної допомоги.

Окремою критичною проблемою стало різке збільшення кількості тварин, які залишилися без опіки. Внаслідок евакуацій, обстрілів та руйнування інфраструктури тисячі чотирилапих улюбленців загубилися, а найтрагічнішим є те, що багато з них залишилися сиротами через загибель своїх власників або вимушену розлуку, коли евакуація з тваринами була фізично неможливою. Існуючі методи пошуку та прилаштування через хаотичні репости в загальних групах Telegram, Instgram, Facebook часто є неефективними через величезний потік новин та відсутність структурованої інформації.

Саме тому розробка спеціалізованого соціального додатку для власників тварин набуває особливої актуальності. Такий продукт перестає бути лише розважальною платформою для публікації фотографій, а трансформується у важливий інструмент соціальної взаємодії та безпеки. Він покликаний вирішити такі задачі:

- координація допомоги та адаптація - створення єдиного простору для комунікації волонтерів та небайдужих людей задля пошуку нових родин для тварин, які втратили власників через війну, або тимчасового прихистку (перетримки) для евакуйованих улюбленців;
- пошук зниклих - оперативне сповіщення спільноти про загублених тварин з використанням геолокації для швидкого реагування користувачів, що знаходяться поруч;
- адаптація ВПО - допомога людям, що переїхали, швидко інтегруватися в нову спільноту власників тварин, знаходити компанію для вигулу та перевірені сервіси;

Таким чином, об'єктом розробки є не просто соціальна мережа, а екосистема взаємодопомоги, що використовує сучасні технології для вирішення гуманітарних проблем власників тварин у кризових умовах.

1. ОСОБЛИВОСТІ СПЕЦІАЛІЗОВАНОГО ДОДАТКУ ДЛЯ ВЛАСНИКІВ ТВАРИН

1.1 Аналіз предметної області та сучасних потреб власників домашніх тварин

Під час повномасштабного вторгнення Росії в Україну зростає потреба у пошуку нового господаря для чотирилапих улюбленців. Особливо якщо побачимо зростаючу кількість груп з пошуком тварин у популярних соцмережах та велику кількість безпритульних тваринок у зоні бойових дій.

І саме тому зароджується стійкий попит на спеціалізовані програмні рішення для вигулу або адопції (прилаштування) домашніх тварин. Адже моніторинг сотень розрізнених груп у популярних соціальних мережах (Facebook, Telegram, Viber) є вкрай неефективним через хаотичність інформації та відсутність інструментів структурування.

Наприклад, у загальнодоступних чатах оголошення про зниклу тварину може "потонути" у стрічці повідомлень вже за 10–15 хвилин через активне листування учасників на інші теми. Крім того, відсутність чіткої географічної прив'язки змушує користувача вручну переглядати контент, який часто не стосується його району чи навіть міста.

Натомість, спеціалізований додаток із системою розумного пошуку дозволяє вирішити цю проблему комплексно:

- геолокація замість хаотичної стрічки: Користувач отримує сповіщення лише про події (наприклад, "Загублено собаку") у радіусі його фактичного перебування, що дозволяє миттєво залучити до пошуку сусідів, а не випадкових людей з інтернету.
- гнучка фільтрація: Можливість відсортувати оголошення за конкретними параметрами (тип тварини, порода, стать, наявність чіпа), що практично неможливо реалізувати у звичайному месенджері.

Така консолідація даних в єдиному інформаційному просторі значно скорочує час реакції спільноти та підвищує шанси на успішний порятунок тварини.

1.2 Огляд та порівняльний аналіз існуючих програмних рішень на ринку.

Для обґрунтування доцільності розробки власної системи необхідно проаналізувати існуючі рішення, які наразі використовуються власниками тварин для комунікації, моніторингу та пошуку. Ринок можна умовно поділити на два сегменти: апаратні GPS-трекери з власним ПЗ та загальні соціальні мережі. Детальне порівняння можна знайти у табл. 1.1.

1.2.1 Апаратні рішення з програмним супроводом (на прикладі Tractive)

Додатки типу Tractive GPS працюють у зв'язці з фізичним GPS-трекером, що кріпиться на нашийник.

Рис. 1.1 - Tractive GPS

Переваги це - висока точність відстеження в реальному часі, історія переміщень, можливість налаштування "віртуальної огорожі" (geofence).

Недоліком є висока вартість. Необхідність купівлі дороговартісного пристрою та оплати щомісячної підписки.

Вузька спеціалізація запропонованого рішення - додаток корисний лише для моніторингу власної тварини. Він не дозволяє комунікувати з іншими власниками, шукати компанію для вигулу або допомагати у пошуку чужих тварин, якщо у них немає трекера.

1.2.2 Групи у соціальних мережах (Facebook, Instagram)

Найпоширеніший спосіб комунікації - тематичні групи ("ANIMALS/Київ", "Власники собак").

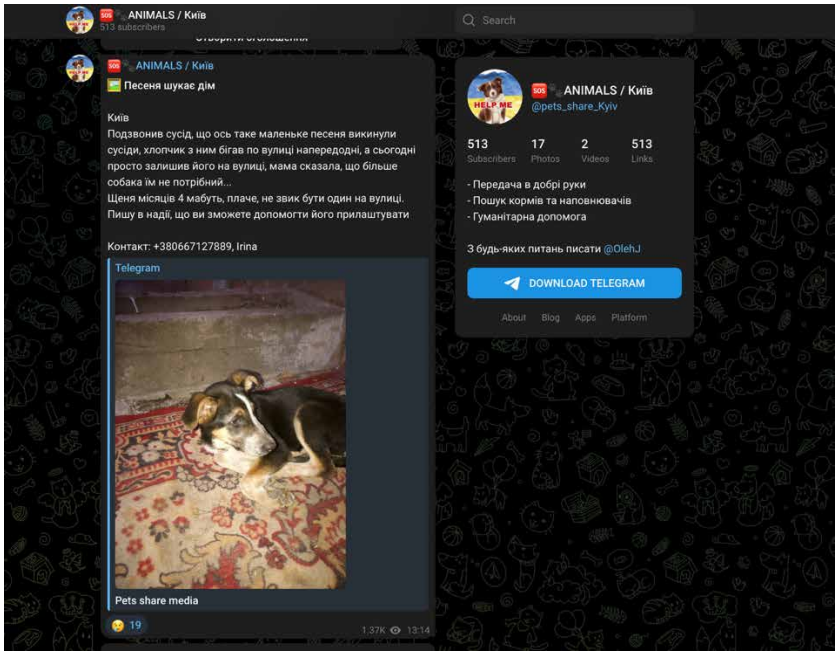

Рис. 1.2 - Telegram-канал “ANIMALS/Київ”

Переваги:

- Велика аудиторія: Можливість зв’язатись з автором оголошення або задати будь-які незрозумілі питання спільного даної групи

Недоліки:

- Алгоритмічна стрічка: Пости відображаються не в хронологічному порядку. Оголошення "Терміново! Собака бігає по проїжджій частині" користувачі можуть побачити через 2 дні, коли це вже неактуально.
- Відсутність структури: Неможливо відфільтрувати пости за породою або конкретною вулицею.
- Низька ефективність пошуку: Інформація губиться серед реклами та спаму.

Таблиця 1.1

Порівняльна характеристика

Критерій порівняння	**Tractive GPS**	**Групи в соц. мережах**	**Проєктований додаток**
Геолокація подій	Висока (GPS)	Відсутня/текстова	Інтерактивна мапа
Соціальна взаємодія	Відсутня	Висока	Висока
Швидкість сповіщення	Миттєва (для власника)	Низька (алгоритми)	Миттєва (Push за радіусом)
Структурованість даних	Висока	Низька	Висока (фільтри)
Вартість використання	Висока (пристрій + підписка)	Безкоштовно	Безкоштовно

1.3 Визначення особливостей взаємодії користувачів у спеціалізованих соціальних мережах

При проєктуванні архітектури соціального додатку для власників тварин необхідно враховувати, що патерни поведінки користувачів у нішевих спільнотах суттєво відрізняються від взаємодії у загальних соціальних мережах (Facebook, Twitter). Ці відмінності базуються на переході від «Social Graph» (зв'язки між знайомими людьми) до «Interest Graph» (зв'язки на основі спільних інтересів та локацій).

Можна виділити три ключові моделі взаємодії, які має забезпечувати система:

1) Гіперлокальна соціалізація (Co-walking). На відміну від онлайн-спілкування, кінцевою метою взаємодії власників тварин часто є офлайн-активність - спільний вигул.
   - Особливість: Користувачі шукають компаньйонів не за спільними друзями, а за географічною близькістю (радіус 1–2 км) та сумісністю тварин (наприклад, розмір собаки, рівень активності).
   - Вимога до системи: Необхідність реалізації алгоритму метчингу (співставлення), який враховує не лише параметри користувача, а й "анкету" тварини (агресивність, стерилізація тощо).
2) Екосистема адопції та кураторства. Універсальна модель, що об'єднує професійний менеджмент притулку з користувацьким інтерфейсом для майбутніх власників.
   - Особливість: Створення єдиного інформаційного простору, де анкета тварини є динамічним об'єктом, що може змінювати власника або статус (у притулку / на перетримці / вдома).
   - Вимога до системи: Реалізація розширеного керування акаунтом. Для притулків це інструмент обліку та візуалізації аналітики (кількість підопічних, розподіл за статтю та типом). Для користувачів - це можливість зручного пошуку пари («Matching») та прямого спілкування з кураторами через вбудований чат.
   - Результат: Притулки отримують можливість автоматизованого експорту звітів (PDF Report), а користувачі - прозорий шлях від перегляду анкети до успішного прилаштування тварини.

3) Аналітичний моніторинг та верифікація (Адміністрування та довіра). Ця модель забезпечує контроль якості даних, безпеку взаємодії та аналіз ефективності роботи притулків.
   - Особливість: Взаємодія користувачів із системою через інструменти звітності та механізми підтвердження статусу організацій.
   - Функціонал для притулків: Формування PDF-звітів про стан бази підопічних та автоматизована візуалізація статистики за типом і статтю тварин («Pets by type», «Pets by gender»).
   - Функціонал для адміністратора: Верифікація нових організацій та модерація анкет, що створює безпечне середовище для всіх учасників процесу прилаштування.

1.4 Моделювання предметної області

Проектування концептуальної схеми предметної області є фундаментальним етапом розробки спеціалізованого програмного забезпечення, що дозволяє чітко формалізувати внутрішні процеси, інформаційні об'єкти та логічні зв’язки всередині системи. Для побудови детального опису архітектури додатку було застосовано інструментарій уніфікованої мови моделювання UML.

Зокрема, для концептуального визначення меж системи, взаємодії акторів та формалізації функціональних вимог було розроблено Use Case діаграму (рис. 1.3), яка візуалізує ключові прецеденти взаємодії користувачів, власників або притулків та адміністраторів із сервісом. Детальний опис акторів можна знайти у табл. 1.2.

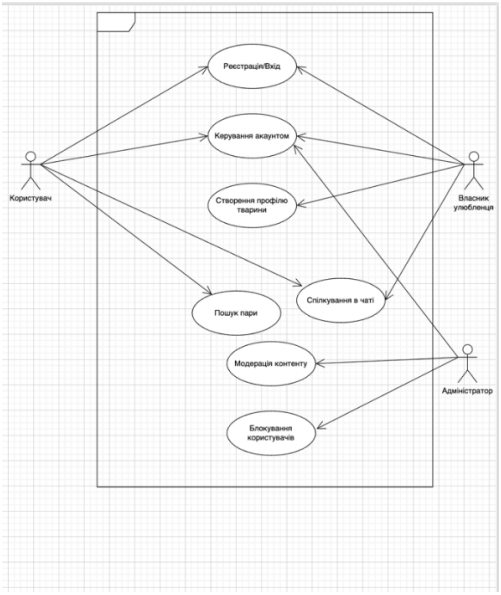

Рис. 1.3 – Use Case діаграма

Основні прецеденти включають:

- Реєстрація/Вхід
- Керування акаунтом
- Створення профілю тваринки
- Спілкування в чаті
- Пошук пари
- Модерація контенту
- Блокування користувачів

Таблиця 1.2

Основні актори системи

№	Назва	Опис
1	Користувач	Фізична особа, яка взаємодіє з додатком переважно з метою пошуку. Проходить автентифікацію, керує особистими

		налаштуваннями та ініціює підбір релевантних анкет за заданими критеріями.
2	Власник улюбленця	Суб'єкт системи (приватний власник або представник притулку), чиєю метою є наповнення бази даних та координація прилаштування. Має ширші права щодо маніпуляції контентом тварин.
3	Адміністратор	Внутрішній актор системи, який відповідає за модерацію контенту, дотримання правил спільноти та забезпечення загальної безпеки інформаційного середовища.

Для деталізації динаміки функціонування мобільного додатка побудовано діаграму активності UML (рис. 1.4). Вона відображає послідовність операцій та логічні розгалуження від запуску програми до фінальної взаємодії користувачів.

Процес починається з відкриття додатка, після чого система перевіряє статус автентифікації. Якщо активний токен сесії відсутній, користувач спрямовується до блоку реєстрації чи входу. Для авторизованих користувачів виконується обов'язкова перевірка наявності створеної анкети тварини. Якщо профіль вихованця в базі даних відсутній, система блокує доступ до стрічки та зобов'язує пройти процедуру його створення, що є необхідною умовою для роботи алгоритму підбору.

Після успішної верифікації даних ініціюється завантаження стрічки через асинхронний запит до бекенду на FastAPI. На екрані пристрою реалізується циклічний перегляд карток: якщо анкета не відповідає вподобанням, система завантажує наступний об'єкт, а у разі зацікавленості - надсилається лайк із фіксацією запису в базі даних Supabase.

Наприкінці система автоматично перевіряє наявність зустрічного лайка від власника переглянутої тварини. За його відсутності користувач повертається до стрічки виявлення. При підтвердженні взаємного інтересу («Метч») додаток

відображає відповідне візуальне сповіщення, ініціює створення унікального ідентифікатора сесії чату та переводить користувача до фінальної дії - безпосереднього листування, де процес досягає кінцевого стану.

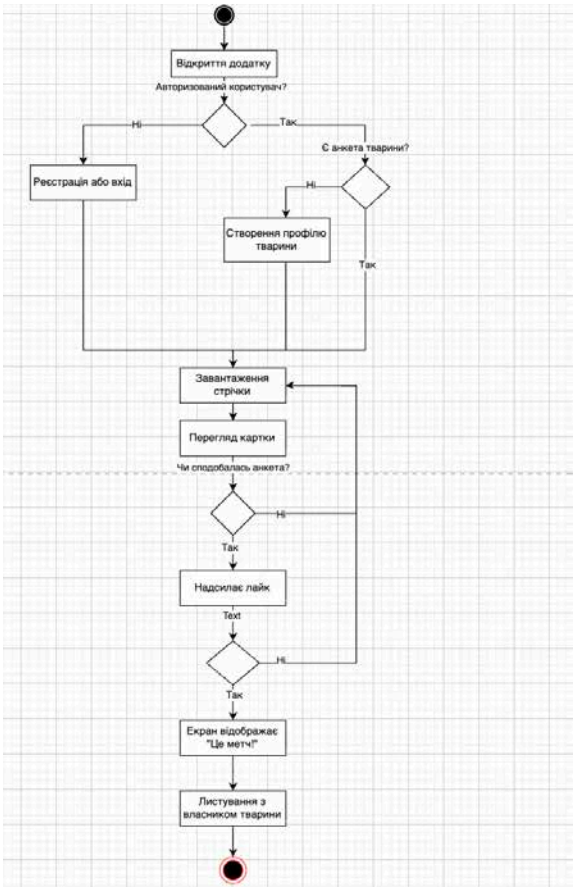

Рис. 1.4 – Діаграма активності

Для представлення інформаційних потоків використано контекстну діаграму (рис. 1.5), яка відображає взаємодію системи з зовнішніми сутностями.

Рис. 1.5 – Діаграма контекстів

Зовнішніми сутностями є ключові суб'єкти або інтегровані компоненти зовнішнього середовища, які взаємодіють із центральним процесом інформаційної системи «pet.io», виступаючи або першоджерелами генерації інформаційних потоків, або кінцевими споживачами результатів обробки даних. У межах побудованої контекстної моделі чітко розмежовано три типи таких сутностей, кожна з яких має унікальний набір прав доступу, функціональних обов'язків та специфікованих інтерфейсів взаємодії.

Першою та найбільш активною зовнішньою сутністю виступає безпосередній користувач, який ініціює більшість транзакцій у реальному часі. Цей суб'єкт постачає до системи вхідні потоки даних у вигляді географічних координат широти й довготи, параметрів фільтрації стрічки виявлення та бінарних реакцій на переглянуті профілі тварин, отримуючи натомість структуровані JSON-відповіді з релевантними анкетами та миттєвими сповіщеннями про утворення комунікаційних сесій у чаті.

Наступною важливою зовнішньою сутністю визначено власника улюбленця, яким може виступати як приватна особа, так і офіційний представник волонтерського притулку. Ця сутність забезпечує наповнення реляційного сховища PostgreSQL та хмарного бакета Supabase Storage метаданими й медіафайлами тварин, а також формує запити на отримання

засобів аналітичного моніторингу, поглинаючи вихідні потоки у вигляді динамічних дашбордів та офіційних звітів у форматі PDF.

Фінальною керуючою зовнішньою сутністю є адміністратор системи, чия взаємодія з процесом нульового рівня зведена до передачі директивних команд модерації контенту й блокування облікових записів порушників при одночасному споживанні логів безпеки та системних звітів про інциденти. Такий підхід до класифікації зовнішніх сутностей дозволяє забезпечити ізоляцію бізнес-логіки та оптимізувати маршрутизацію даних між клієнтською та серверною частинами додатка.

1.5 Функціональні вимоги для додатку

На основі проведеного аналізу предметної області та недоліків існуючих аналогів, можна сформувати перелік ключових вимог до функціоналу розроблюваної системи. Додаток повинен вирішувати завдання соціальної взаємодії, безпеки та інформаційної підтримки.

Основні функціональні блоки системи:

1) Модуль "Профілі та Соціалізація"

Блок, відповідальний за створення цифрової екосистеми взаємодії власників тварин.

- Реєстрація та автентифікація користувачів: безпечний доступ до системи за допомогою хмарних сервісів (Supabase Auth).
- Цифрова анкета тварини: створення детального профілю, що включає тип (кіт, собака тощо), стать, породу, вік, особливості поведінки, рівень дресирування та ветеринарні дані.
- Пошук партнерів для прогулянок: функціонал гіперлокальної соціалізації (Co-walking) на основі параметрів сумісності тварин для безпечного вигулу.

2) Модуль "Прилаштування та Допомога"

Універсальний інструмент для цифровізації процесів передачі тварин новим власникам та підтримки притулків.

- Структурована база оголошень: інтерактивний каталог тварин, які шукають нові домівки, із розширеною системою фільтрації за характеристиками.
- Верифікація та модерація: багаторівнева перевірка волонтерів та притулків для побудови середовища довіри та запобігання шахрайству.
- Центр комунікації: вбудована система миттєвих повідомлень (чат) для зв'язку між куратором притулку та потенційним власником.

3) Модуль "Інтелектуальний підбір та Аналітика"

Технічне ядро системи, що забезпечує автоматизацію вибору та візуалізацію результатів діяльності.

- Механіка Matching: алгоритм персоналізованого підбору тварин на основі вподобань користувача (наприклад, фільтрація за конкретною породою - Jack Russell Terrier).
- Візуалізація даних: модуль аналітики у реальному часі, що відображає статистику бази: кількість підопічних, розподіл за статтю (Male/Female) та видами.
- Цифрова звітність: функція експорту поточної статистики профілю у формат PDF (Export PDF Report) для офіційного документообігу волонтерських організацій.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи здійснено системний аналіз предметної області та досліджено процеси адопції й соціалізації домашніх тварин. З'ясовано, що внаслідок наслідків повномасштабного вторгнення проблема безпритульних тварин в Україні набула критичної гостроти, що зумовило потребу у створенні ефективних цифрових інструментів координації. Порівняльний аналіз ринку показав, що наявні рішення (соціальні мережі, месенджери та GPS-трекери) вирішують завдання лише частково. Головними деструктивними факторами існуючих платформ є висока хаотичність інформаційних потоків та відсутність автоматичної географічної прив’язки, через що критично важливі оголошення швидко втрачаються у стрічці повідомлень.

Обґрунтовано, що комплексним вирішенням виявлених проблем є розробка спеціалізованого мобільного додатка «pet.io», який поєднує інструменти гіперлокальної взаємодії та розумного пошуку. На відміну від стихійних комунікаційних каналів, проектована система забезпечує відображення цільових сповіщень виключно у радіусі фактичного перебування користувача та пропонує гнучку фільтрацію за специфічними ознаками тварин (тип, порода, стать, вік, наявність чипа). Це дозволяє автоматизувати наповнення бази даних, мінімізувати часові витрати на пошук релевантних анкет та суттєво підвищити шанси на успішне прилаштування вихованців.

Формулювання функціональних і технічних вимог у підрозділі 1.4 дозволило чітко визначити бізнес-логіку взаємодії акторів, вимоги до швидкодії серверної частини, кросплатформеності інтерфейсу та безпеки даних. Результати першого розділу сформували повноцінне теоретичне й аналітичне підґрунтя, необхідне для подальшого проектування архітектури, розробки структури реляційної бази даних та безпосередньої програмної реалізації клієнт-серверних компонентів системи у наступних розділах бакалаврської роботи.

2. ПРОЄКТУВАННЯ СПЕЦІАЛІЗОВАНОГО ДОДАТКУ ДЛЯ ВЛАСНИКІВ ТВАРИН

2.1. Обґрунтування архітектурних рішень

Для реалізації соціального додатку, що поєднує функції месенджера, інтерактивної карти та системи сповіщень, доцільно обрати клієнт-серверну архітектуру.

Враховуючи необхідність швидкої розробки та надійної роботи з даними в реальному часі, архітектура будується на базі сучасних хмарних рішень (BaaS - Backend as a Service).

Загальна схема системи складається з трьох рівнів:

1. Клієнтська частина (Mobile Frontend): Кросплатформний мобільний додаток, що відповідає за UI/UX та взаємодію з апаратними функціями телефону (GPS, камера).
2. Серверна частина (Backend API): Відповідає за складну бізнес-логіку, яка не може бути реалізована на стороні клієнта або бази даних (наприклад, специфічні алгоритми метчингу).
3. Рівень даних та інфраструктури (Database & Infrastructure): Забезпечує зберігання даних, автентифікацію та Realtime-синхронізацію.

Вибір технологічного стеку:

- Мобільна розробка: React Native. Цей фреймворк дозволяє використовувати єдину кодову базу (JavaScript/TypeScript) для iOS та Android, забезпечуючи при цьому високу продуктивність, близьку до нативної.
- Backend: FastAPI (Python). Обрано через високу швидкість обробки запитів (асинхронність), автоматичну генерацію документації (Swagger UI).
- База даних та сервіси: Supabase. Це open-source альтернатива Firebase, яка надає:

1) Потужну реляційну базу даних PostgreSQL.
2) Вбудовану систему автентифікації (Auth).
3) Realtime subscriptions (для миттєвого оновлення чатів та геолокації).

- Object Storage (для зберігання фото тварин).

2.2. Проєктування бази даних

Розроблена схема бази даних базується на реляційній моделі та враховує специфіку додатку для знайомств ("Tinder-like mechanics"). Ключовою особливістю є розділення технічних даних автентифікації та публічного профілю, а також наявність проміжної сутності для фіксації взаємодій користувачів (свайпів) (Рис. 2.1).

Схема реалізована на базі СУБД PostgreSQL (платформа Supabase) і включає шість основних сутностей.

2.2.1 Специфікація сутності «Користувач» (User)

Базовим елементом інформаційного забезпечення системи «pet.io» є сутність «Користувач» (User), яка призначена для зберігання облікових і персональних даних зареєстрованих осіб, приватних власників тварин та представників притулків. Ця таблиця інтегрована з модулем автентифікації Supabase Auth та містить географічні координати користувачів, необхідні для роботи серверного алгоритму підбору. Повну специфікацію фізичної структури таблиці users із зазначенням типів даних СУБД PostgreSQL наведено в табл. 2.1.

Табл. 2.1

Специфікація фізичної структури таблиці “Users”

Назва поля	**Тип даних**	**Ключ**	**Опис та обмеження поля**

user_id	uuid	PK	Унікальний ідентифікатор користувача, що генерується системою Supabase Auth автоматично.
name	varchar	-	Ім'я або нікнейм користувача для відображення в інтерфейсі додатка.
email	varchar	-	Електронна пошта користувача, яка використовується як логін для авторизації.
latitude	float8	-	Географічна широта поточного або зафіксованого місця перебування користувача.
logntitude	float8	-	Географічна довгота поточного або зафіксованого місця перебування користувача.
image_url	text	-	Абсолютний URL-шлях до файлу аватарки користувача, що зберігається в Supabase Storage.
created_at	timestamp	-	Системна дата та час створення облікового

			запису (значення за замовчуванням NOW()).

2.2.2 Специфікація сутності «Власник тварини» (Pet_owner)

Сутність, що розширює профіль користувача, надаючи йому статус власника або представника притулку. Повну специфікацію фізичної структури таблиці users із зазначенням типів даних СУБД PostgreSQL наведено в табл. 2.2.

Таблиця 2.2

Специфікація фізичної структури таблиці Pet_owner

Назва поля	**Тип даних**	**Ключ**	**Опис та обмеження поля**
owner_id	uuid	PK	Унікальний ідентифікатор власника, що генерується системою Supabase Auth автоматично.
user_id	uuid	FK	Зовнішній ключ, що посилається на user_id в таблиці User
bio	text	-	Біографія або опис власника тварини
created_at	timestamp	-	Системна дата та час створення профілю власника тварини (значення за замовчуванням NOW()).

2.2.3 Специфікація сутності «Тварина» (Pet)

Сутність, що використовується для збереження детальних метаданих про вихованців, що підлягають процедурі геопросторового підбору або адопції. Ця таблиця пов'язана з таблицею користувачів відношенням «один-до-багатьох» через зовнішній ключ власника. Повний набір фізичних атрибутів профілю вихованця, що оптимізований для швидкої фільтрації бекендом на FastAPI та містить посилання на медіафайли у хмарному сховищі Supabase Storage, деталізовано в табл. 2.3.

Таблиця 2.3

Специфікація фізичної структури таблиці Pet

Назва поля	**Тип даних**	**Ключ**	**Опис та обмеження поля**
pet_id	uuid	PK	Унікальний ідентифікатор анкети тварини, що генерується автоматично.
owner_id	uuid	FK	Зовнішній ключ, що посилається на owner_id у таблиці Pet_owner (зв'язок один-до-багатьох).
type	varchar	-	Тип тварини
name	varchar	-	Ім’я тварини
gender	varchar	-	Стать тварини
created_at	timestamp	-	Системна дата та час створення профілю анкети тварини (значення за замовчуванням NOW()).
pet_url	text	-	Абсолютний URL-шлях до файлу фото анкети
species	varchar	-	Порода тварини

behaviour	text	-	Тип поведінки тварини
training	text	-	Тип тренування тварини
age	numeric	-	Вік тварини

2.2.4 Специфікація сутності Swipe (Свайп/Взаємодія)

Сутність, що призначена для фіксації транзакційних дій користувачів під час ітераційного перегляду карток у стрічці виявлення. Вона акумулює дані про вибір користувача, пов'язуючи його ідентифікатор з анкетою тварини, та зберігає тип реакції, що необхідно для роботи алгоритму геопросторового скорингу та виключення вже переглянутих профілів із повторних пошукових запитів. Фізичну структуру таблиці swipes у базі даних PostgreSQL наведено в табл. 2.4.

Таблиця 2.4

Специфікація фізичної структури таблиці Swipe

Назва поля	**Тип даних**	**Ключ**	**Опис та обмеження поля**
swipe_id	uuid	PK	Унікальний системний ідентифікатор запису реакції.
user_id	uuid	FK	Зовнішній ключ, що посилається на user_id у таблиці User (хто свайпає).
pet_id	uuid	FK	Зовнішній ключ, що посилається на pet_id у таблиці Pet (кого свайпають).

is_liked	boolean	-	Логічний прапорець типу реакції: TRUE - позитивна (лайк), FALSE - негативна (дизлайк).
gender	varchar	-	Стать тварини
created_at	timestamp	-	Системна дата та час здійснення операції (за замовчуванням NOW()).

2.2.5. Match (Метч/Співпадіння)

Сутність «Метч» (Match) фіксує факт виникнення взаємного інтересу між користувачем, який шукає вихованця, та власником або представником притулку. Запис у цій таблиці генерується автоматично сервером на FastAPI в момент перевірки умов транзакції, коли поточний лайк збігається із зустрічним лайком. Поява нового запису у цій таблиці є архітектурним тригером для ініціалізації унікальної комунікаційної сесії. Фізичну структуру таблиці matches наведено в табл. 2.5.

Таблиця 2.5

Специфікація фізичної структури таблиці Match

Назва поля (Column Name)	**Тип даних (Data Type)**	**Ключ (Key)**	**Опис та обмеження поля**
match_id	uuid	PK	Унікальний системний ідентифікатор зафіксованого збігу.

user_id	uuid	FK	Зовнішній ключ користувача, чий лайк став взаємним (посилання на User).
pet_id	uuid	FK	Зовнішній ключ анкети тварини, щодо якої зафіксовано інтерес (посилання на Pet).
created_at	timestamp	-	Дата та час автоматичного формування збігу системою.

2.2.6 Chat (Чат)

Сутність, що слугує для агрегації повідомлень у межах окремої комунікаційної сесії, що ініціюється між користувачами після успішної фіксації взаємного інтересу. Ця таблиця пов'язує унікальний ідентифікатор діалогу із відповідним тригером у таблиці збігів, що забезпечує архітектурну цілісність системи комунікації та дозволяє бекенду на FastAPI коректно авторизувати доступ сторін до спільного каналу листування. Фізичну структуру таблиці chats деталізовано в табл. 2.6.

Таблиця 2.6

Специфікація фізичної структури таблиці Chat

Назва поля (Column Name)	**Тип даних (Data Type)**	**Ключ (Key)**	**Опис та обмеження поля**
chat_id	uuid	PK	Унікальний системний ідентифікатор сесії чату.
match_id	uuid	FK	Зовнішній ключ, що посилається на match_id у таблиці Match (основа для відкриття чату).

is_active	boolean	-	Чи активний чат між користувачами
last_message_at	timestamp	-	Дата та час створення останнього повідомлення.

2.2.7 Message (Повідомлення)

Сутність, що призначена для збереження вмісту та метаданих окремих текстових реплік, якими обмінюються користувачі всередині активного діалогу. Вона містить зовнішні ключі для зв'язку з батьківським чатом та конкретним відправником, а також логічний індикатор для моніторингу статусу перегляду повідомлення, що необхідно для коректного відображення маркерів прочитання в UI-компонентах мобільного додатка. Фізичну структуру таблиці messages наведено в табл. 2.7.

Таблиця 2.7

Специфікація фізичної структури таблиці Message

Назва поля (Column Name)	**Тип даних (Data Type)**	**Ключ (Key)**	**Опис та обмеження поля**
message_id	uuid	PK	Унікальний системний ідентифікатор повідомлення.
user_id	uuid	FK	Зовнішній ключ, що посилається на user_id у таблиці User.
chat_id	uuid	FK	Зовнішній ключ, що посилається на chat_id у таблиці Chat.
content	text	-	Контент повідомлення

sent _at	timestamp	-	Дата та час надіслання останнього повідомлення.

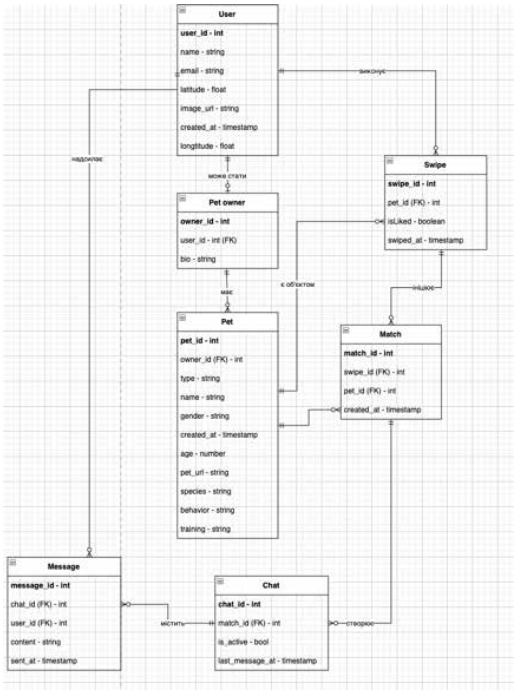

Рис. 2.1 – Діаграма класів

2.3 Створення інформаційної бази

На основі розробленої логічної моделі даних (ER-діаграми) проведено етап фізичного проектування у середовищі Supabase (PostgreSQL). Фізична модель відображає конкретну реалізацію таблиць із визначенням типів даних, обмежень цілісності та зв'язків між сутностями (Рис. 2.2).

- Перехід до фізичної схеми: Для кожної сутності логічної моделі створено відповідні таблиці, де ідентифікатори об’єктів (user_id, pet_id, match_id) реалізовано за допомогою типу даних UUID для забезпечення унікальності в розподіленій системі.
- Забезпечення цілісності: На рівні бази даних встановлено обмеження FOREIGN KEY та NOT NULL, що гарантує валідність даних (наприклад,

неможливість існування анкети тварини без прив'язки до власника в таблиці Pet_owner).

- Схема бази даних: Результатом етапу є схема, що ілюструє структуру інформаційної бази, включаючи таблиці для збереження профілів, результатів взаємодії (свайпів) та повідомлень чату.

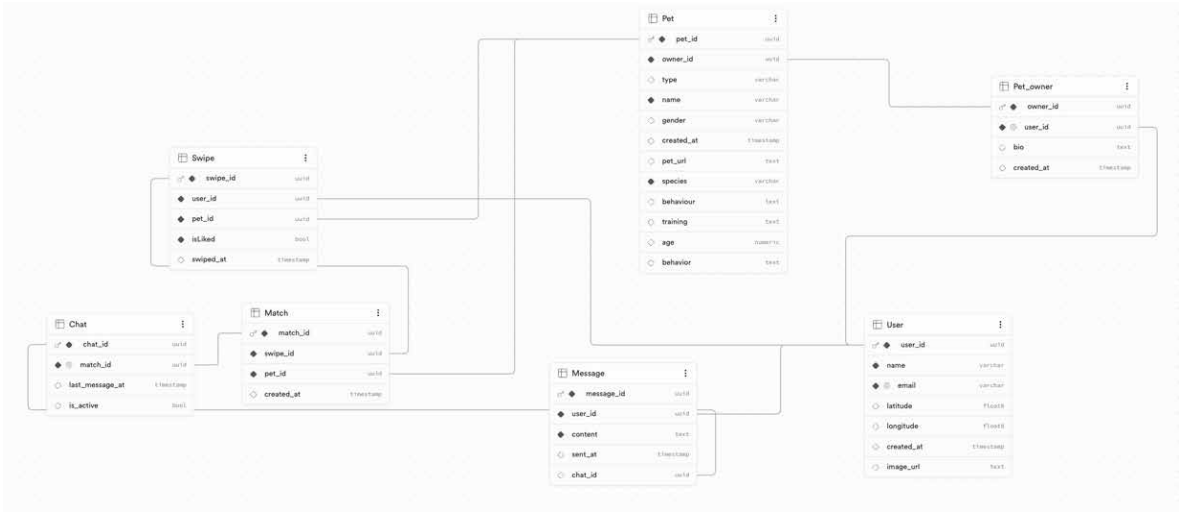

Рис. 2.2 – Спроєктована БД в середовищі Supabase

2.4 Логічне обґрунтування нормалізації бази даних

Для запобігання дублюванню інформації та виникненню аномалій під час виконання операцій додавання, оновлення чи видалення записів, схему бази даних додатка «pet.io» було нормалізовано до третьої нормальної форми (3НФ).

Усі таблиці розробленої бази даних відповідають вимогам першої нормальної форми (1НФ), оскільки кожне поле містить лише одне атомарне значеннння. У структурі повністю відсутні масиви або повторювані групи атрибутів у межах одного рядка. Наприклад, у таблиці pets такі характеристики вихованця, як кличка, стать чи вид, є неподільними, а посилання на фотокартки винесені в окреме текстове поле.

Вимоги другої нормальної форми (2НФ) передбачають, що база даних уже перебуває в 1НФ, а всі неключові поля повністю залежать від первинного ключа.

Оскільки для ідентифікації записів у таблицах users, pets, swipes, matches, chats та messages використовуються унікальні сурогатні первинні ключі типу UUID, часткова залежність атрибутів від частин ключа архітектурно виключена. Кожне неключове поле однозначно визначається унікальним ідентифікатором свого рядка.

На фінальному етапі перевірено відповідність схеми критеріям третьої нормальної форми (3НФ), яка вимагає відсутності транзитивних залежностей між неключовими атрибутами. У системі «pet.io» цей критерій реалізовано шляхом розподілу сутностей за окремими таблицями. Інформація про власника тварини не зберігається всередині профілю вихованця, а винесена у таблицю users, зв’язок із якою забезпечується через зовнішній ключ owner_id. Повідомлення також містять лише ідентифікатор діалогу chat_id та автора sender_id, без дублювання метаданих самого чату.

Оскільки всі неключові атрибути кожної з таблиць залежать виключно від первинного ключа і не пов'язані між собою, схема бази даних повністю відповідає вимогам 3НФ.

2.5 Опис механізмів безпеки та розмежування доступу в Supabase

Захист персональних даних користувачів та розмежування прав доступу в додатку «pet.io» реалізовано за допомогою вбудованих інструментів безпеки платформи Supabase та СУБД PostgreSQL.

Для контролю доступу до даних на рівні окремих рядків таблиць застосовано механізм Row Level Security (RLS), інтегрований у PostgreSQL. На відміну від обмеження доступу до таблиці в цілому, RLS дозволяє динамічно перевіряти права авторизованого користувача для кожного рядка під час виконання операцій вибірки, додавання або оновлення записів. У базі даних «pet.io» налаштовані правила, які дозволяють власнику тварини або представнику притулку редагувати чи видаляти лише ті профілі, де його ідентифікатор збігається зі значенням у полі owner_id. Перегляд та надсилання

повідомлень у таблиці messages також обмежені лише для тих користувачів, які є безпосередніми учасниками конкретної сесії чату (chat_id).

Автентифікацію клієнтів та керування сесіями забезпечує модуль Supabase Auth. Паролі користувачів не зберігаються у відкритому вигляді. Під час реєстрації система виконує криптографічне хешування паролів за допомогою алгоритму bcrypt у захищеній внутрішній схемі бази даних, що унеможливлює їхнє відновлення чи дешифрування.

Для взаємодії мобільного додатка на React Native із серверною частиною на FastAPI використовуються токени JSON Web Tokens (JWT). Після успішного входу користувача Supabase Auth генерує підписаний JWT-токен із обмеженим термін дії. Мобільний клієнт автоматично додає цей токен до заголовка кожного HTTP-запиту, а бекенд-сервер на FastAPI перевіряє його дійсність за допомогою секретного ключа. Це забезпечує безпечний обмін даними в мережі без необхідності постійного повторного надсилання логіна та пароля.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи було виконано безпосередню програмну реалізацію клієнтської та серверної частин інформаційної системи «pet.io». На основі висунутих технічних вимог обґрунтовано вибір технологічного стеку, що включає фреймворк React Native для розробки кросплатформеного мобільного інтерфейсу та асинхронний фреймворк FastAPI для побудови високопродуктивного бекенд-сервера на мові Python. Інтеграція з хмарною платформою Supabase дозволила ефективно вирішити завдання автентифікації користувачів, збереження медіафайлів у сховищі Storage та організації реляційної бази даних на базі СУБД PostgreSQL.

Головну увагу в розділі приділено розробці та опису ключових алгоритмів функціонування системи. Програмно втілено логіку фіксації взаємних лайків для автоматичного створення збігів, а також модуль Real-time чатів для забезпечення обміну повідомленнями між користувачами. Наведені фрагменти програмного

коду та блок-схеми алгоритмів підтверджують логічну цілісність додатка й коректність обробки керуючих потоків даних.

Спроектована клієнт-серверна архітектура забезпечує низьку затримку при обробці запитів за рахунок асинхронного виконання операцій та обміну даними у форматі JSON. Розподіл обчислювального навантаження між мобільним клієнтом та сервером дозволив оптимізувати використання апаратних ресурсів пристроїв і забезпечити швидке оновлення інтерфейсу. Таким чином, результати третього розділу сформували готову програмну основу інформаційної системи, що дозволяє перейти до етапу комплексного тестування та розгортання компонентів у наступній частині роботи.

3. ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННННЯ

3.1 Організаційна структура програмного забезпечення

Для візуалізації фізичної структури розробленої системи та визначення залежностей між її програмними модулями розроблено діаграму пакетів. Вона дозволяє структурувати програмне забезпечення за функціональним призначенням, розділяючи клієнтську та серверну частини.

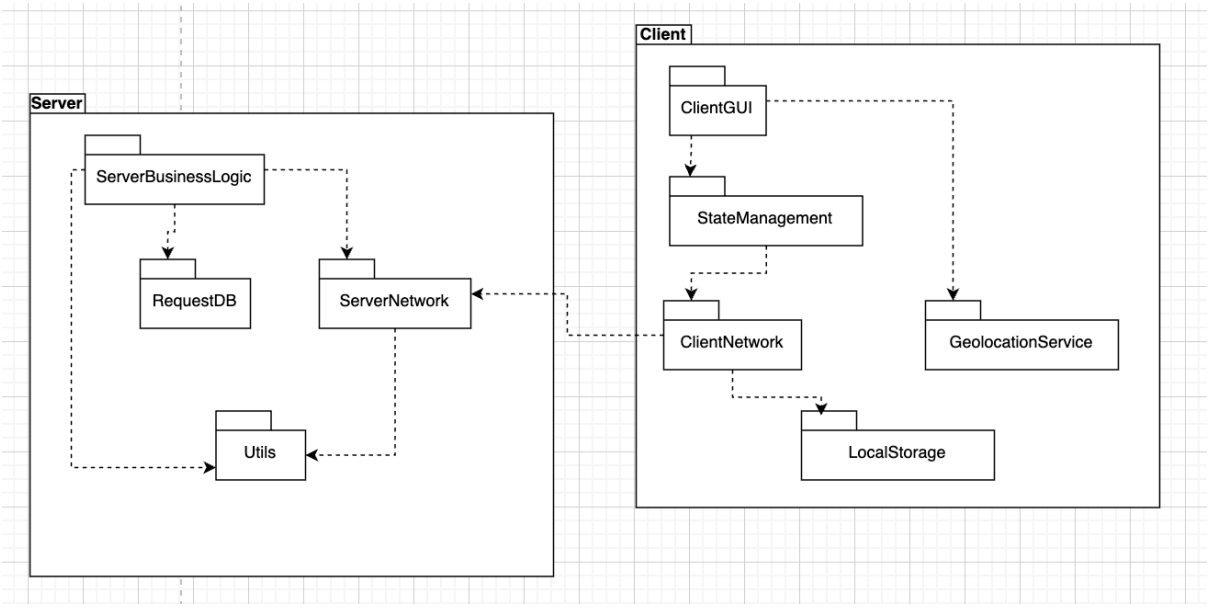

Рис. 3.1 – Діаграма пакетів

Основними пакетами діаграми є:

1) Серверна частина додатку (Server):

Цей блок відповідає за обробку бізнес-логіки, безпечну роботу з даними та взаємодію з базою даних. Він побудований на базі FastAPI та Supabase.

– ServerBusinessLogic: Центральний пакет бекенду, де реалізовані основні алгоритми системи, такі як інтелектуальний підбір тварин (Matching) та підготовка даних для аналітики. Він координує роботу інших пакетів.

- RequestDB: Модуль, відповідальний за безпосередню взаємодію з базою даних PostgreSQL у Supabase. Він виконує SQL-запити для читання/запису інформації про користувачів, тварин та чати.
- ServerNetwork: Реалізує REST API інтерфейс сервера. Цей пакет приймає вхідні HTTP-запити від мобільного додатка та повертає відповіді у форматі JSON.
- Utils: Допоміжний пакет, що містить спільні інструменти, наприклад, модуль для генерації PDF-звітів (Export PDF Report) або валідатори даних.

2) Клієнтська частина додатку (Mobile):

Цей блок представляє кросплатформенний мобільний додаток, розроблений на React Native (Expo).

- ClientGUI: Відповідає за візуальний інтерфейс користувача. Включає екрани свайпінгу, профілі тварин та аналітичні дашборди.
- StateManagement: Пакет для керування глобальним станом додатка (наприклад, дані поточного користувача, список метчів). Забезпечує синхронізацію даних між різними екранами.
- ClientNetwork: Модуль зв'язку, що ініціює запити до серверного API (ServerNetwork) для отримання свіжих анкет або відправки результатів свайпів.
- GeolocationService: Критично важливий модуль для реалізації функцій гіперлокальної соціалізації. Він працює з GPS-даними телефону для пошуку партнерів для вигулу поблизу.
- LocalStorage: Використовується для кешування даних та зберігання локальних налаштувань користувача на пристрої для забезпечення швидкодії.

Взаємозв'язки на діаграмі

- Залежність між Client та Server: Клієнтська частина через пакет ClientNetwork залежить від ServerNetwork. Це відображає типову архітектуру, де фронтенд отримує дані через API.
- Внутрішні зв'язки: На бекенді ServerBusinessLogic звертається до RequestDB для отримання даних та до Utils для виконання специфічних завдань. На фронтенді ClientGUI залежить від стану (StateManagement) та мережевого модуля.

3.2 Налаштування середовища розробки та конфігурації проєкту

Процес програмної реалізації інформаційної системи розпочався з підготовки середовища розробки та налаштування взаємодії між клієнтською частиною (React Native), серверною частиною (FastAPI) та хмарною інфраструктурою (Supabase). Опис програмної реалізації описано в діаграмі вузлів (Рис. 3.2)

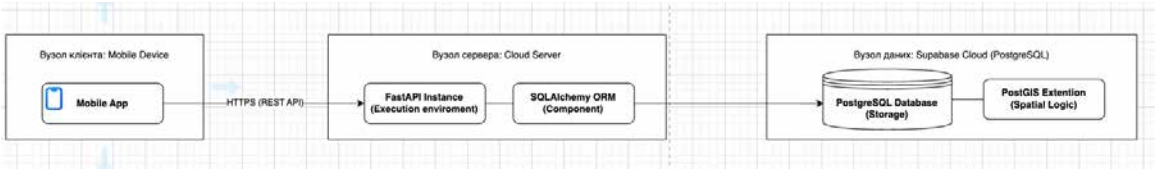

Рис. 3.2 – Діаграма вузлів

3.2.1 Підготовка робочого місця розробника

Для розробки компонентів системи було встановлено та налаштовано наступне програмне забезпечення:

- Node.js (LTS версія) - середовище виконання JavaScript для роботи з пакетним менеджером npm та запуску Expo CLI.
- Python 3.10+ - інтерпретатор для розробки серверної частини.

- Visual Studio Code - основне інтегроване середовище розробки (IDE) з інстальованими розширеннями для підтримки TypeScript, Python (Pylance) та Tailwind CSS IntelliSense.
- Git - система контролю версій для управління кодовою базою проєкту.

3.2.2 Конфігурація клієнтської частини (Frontend)

Створення мобільного додатка ініційовано за допомогою інструментарію Expo. Основні етапи конфігурації включали:

1. Ініціалізація проєкту: виконання команди npx create-expo-app з використанням шаблону TypeScript для забезпечення суворої типізації.
2. Налаштування NativeWind: інтеграція Tailwind CSS для React Native шляхом конфігурації файлу tailwind.config.js та налаштування Babel-плагіна, що дозволило використовувати утилітарні класи для верстки інтерфейсу.
3. Інсталяція залежностей: додавання бібліотек supabase-js для зв'язку з базою даних, react-navigation для побудови маршрутизації між екранами та компонентів для візуалізації графіків аналітики.

3.2.3 Конфігурація серверної частини (Backend)

Розгортання бекенду на базі FastAPI передбачало створення ізольованого середовища:

1. Створення Virtual Environment: використання модуля venv для керування залежностями проєкту.
2. Налаштування середовища: інсталяція пакетів fastapi, uvicorn (ASGI-сервер) та supabase-py.
3. Управління секретами: створення файлу конфігурації .env для збереження конфіденційних даних, таких як SUPABASE_URL та

SUPABASE_ANON_KEY, що забезпечує безпечне підключення до хмарної бази даних.

3.2.4 Інтеграція з інфраструктурою Supabase

Фізичне налаштування бази даних відбувалося в консолі керування Supabase:

- SQL Editor: виконання розробленого SQL-скрипта для створення таблиць (User, Pet, Match, Chat, Message) та встановлення зв'язків між ними за допомогою UUID.
- Storage Configuration: створення публічного кошика (Bucket) у Supabase Storage для зберігання медіафайлів - фотографій тварин та аватарів користувачів. Налаштування політик доступу (RLS) для забезпечення безпеки завантаження файлів.

3.3 Алгоритмізація та програмування програмних модулів

Процес розробки програмних модулів соціального додатку для власників тварин базується на розподілі логіки між клієнтським мобільним додатком та серверною частиною. Ключові компоненти системи реалізовані у вигляді ізольованих, але взаємопов'язаних модулів, які забезпечують автентифікацію, динамічний підбір анкет вихованців за геопозицією, фіксацію взаємних реакцій користувачів та ведення листування у реальному часі. Усі алгоритми побудовані з урахуванням асинхронної моделі обробки даних, що дозволяє мінімізувати час відгуку клієнтського інтерфейсу при високій щільності запитів до реляційного сховища.

Першим елементом архітектури є модуль реєстрації та валідації облікових записів користувачів через Supabase Auth, який відповідає за первинну перевірку вхідних даних на стороні сервера FastAPI. Процес розпочинається з отримання

від клієнтського додатка текстових полів, що містять електронну пошту, ім'я та пароль. Алгоритм виконує перевірку на наявність порожніх значень та відповідність адреси емейла стандартному регулярному виразу, після чого здійснюється верифікація довжини та складності пароля. Після успішної локальної валідації сервер надсилає асинхронний запит до хмарної служби Supabase Auth для перевірки унікальності адреси у системі. У разі відсутності дублікатів генерується унікальний ідентифікатор користувача типу UUID, створюється новий рядок у реляційній таблиці користувачів та повертається підписаний токен сесії. Графічну інтерпретацію цієї логіки представлено на блок-схемі алгоритму реєстрації (рис. 3.3).

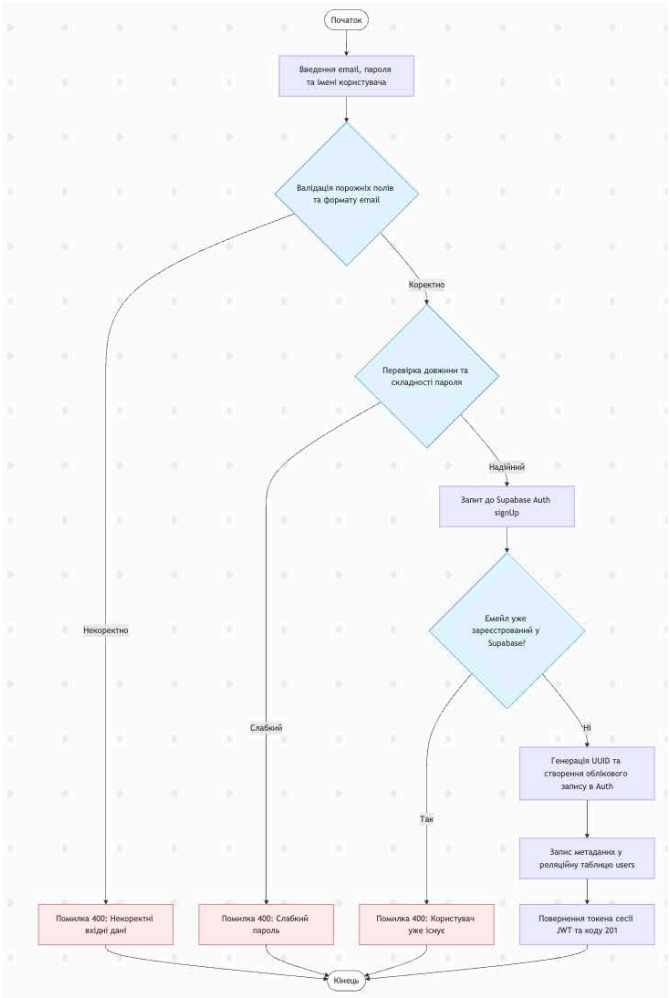

Рис. 3.3 – Алгоритм реєстрації

Основним компонентом бізнес-логіки додатка є модуль геопросторового підбору та фільтрації анкет тварин, що доступний за ендпоінтом /pets/nearby. Функціонування цього модуля починається з отримання поточних координат широти й довготи мобільного пристрою користувача та нормалізації вхідних параметрів пошуку. Для зменшення навантаження на обчислювальні ресурси сервера алгоритм виконує послідовну багатокрокову фільтрацію масиву даних. Спочатку з вибірки повністю виключаються анкети тварин, які належать самому автору запиту, а потім відсікаються профілі, які користувач уже оцінив раніше, що визначається шляхом перевірки ідентифікаторів у таблиці реакцій Swipe. На наступному етапі для решти анкет обчислюється відстань на основі координат власників тварин, і якщо вона перевищує заданий у фільтрах радіус, анкета відхиляється. Для профілів, які пройшли всі етапи фільтрації, запускається функція розрахунку релевантності, яка присвоює числовий бал за збіг породи, статі чи виду вихованця із вподобаннями шукача. У разі наявності історії взаємодій результат сортується за спаданням балів, а для нових користувачів застосовується механізм випадкового перемішування для подолання проблеми «холодного старту». Рух потоків керування цього процесу відображено на блок-схемі алгоритму підбору анкет (рис. 3.4).

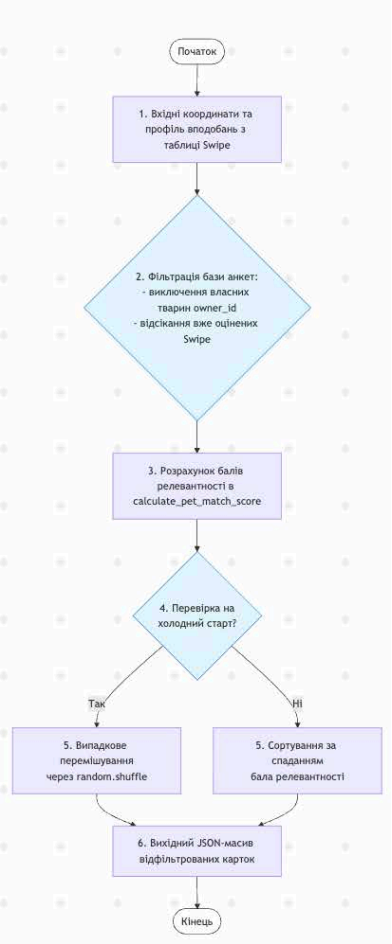

Рис. 3.4 - Алгоритм підбору анкет

Модуль фіксації реакцій та створення збігів опрацьовує дії користувача під час перегляду карток у стрічці виявлення. Коли користувач здійснює свайп, мобільний додаток відправляє на сервер ідентифікатори діючих суб'єктів та логічний прапорець типу реакції, а серверна частина реєструє цей запис у таблиці реакцій. Якщо зафіксовано позитивну відповідь (лайк), алгоритм негайно ініціює перевірку бази даних на наявність зустрічного інтересу. Для цього виконується пошук у таблиці реакцій, де автором виступає власник оціненої тварини, а об'єктом - анкети поточного користувача. У разі виявлення взаємного збігу система автоматично генерує новий запис у таблиці збігів Match, що слугує архітектурним тригером для відкриття унікальної кімнати комунікації в таблиці Chat та відправки миттєвого сповіщення обом сторонам через сокети Supabase Realtime. Логічну послідовність створення збігу деталізовано на схемі алгоритму

метчингу (рис. 3.5). Заключним етапом програмування є реалізація модуля повідомлень, який забезпечує збереження вмісту діалогів у таблиці Message та маркування статусів прочитання з хронологічним сортуванням записів за часом створення.

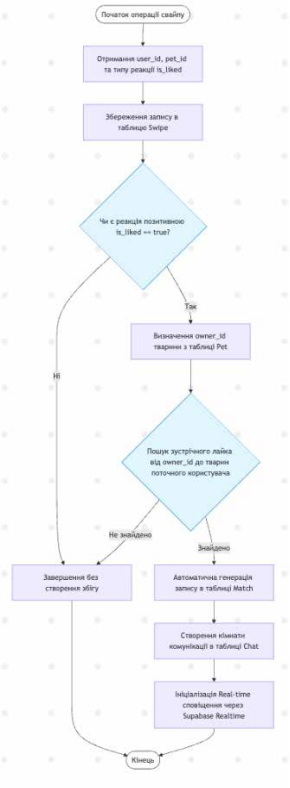

Рис. 3.5 - Алгоритм підбору анкет

У лістингу, який знаходиться в Додатку А, продемонстровано реалізацію логіки фільтрації та підготовки даних до ранжування:

Наведений фрагмент коду реалізує логіку роботи пошукового ендпоінту: від вибірки даних із бази до формування фінального списку анкет. Роботу цього коду можна розділити на три чіткі етапи.

На першому етапі за допомогою інструкції await db.execute виконується асинхронний запит до бази даних. Оператор await дозволяє серверу FastAPI не блокувати робочий потік під час очікування відповіді від PostgreSQL, завдяки чому сервер може одночасно обробляти запити інших користувачів. Сам SQL-

запит використовує два методи join: спочатку таблиця тварин Pet об'єднується з таблицею зв'язків PetOwner, а потім - з таблицею користувачів User. Це дозволяє одним запитом отримати саму анкету тварини та географічні координати її власника, що виключає повторні звернення до бази даних.

Другий етап - це перебір отриманих результатів у циклі for та їх послідовна фільтрація за допомогою умовних операторів if. Перевірка owner_user.user_id == parsed_user_id відсікає власні оголошення поточного користувача, щоб він не бачив у стрічці своїх тварин. Умова pet.pet_id in swiped_pet_ids перевіряє, чи оцінював користувач цю анкету раніше, і якщо так - пропускає її. Наступні перевірки порівнюють вид і тип тварини з налаштуваннями пошуку. Якщо анкета пройшла ці фільтри, функція haversine обчислює відстань у кілометрах між координатами користувача та власника тварини. Якщо відстань менша або дорівнює значенню radius, викликається функція calculate_pet_match_score. Вона розраховує числовий бал релевантності анкети на основі вподобань користувача, після чого дані додаються до масиву nearby.

На третьому етапі виконується фінальне ранжування списку. Якщо у користувача є історія лайків/дизлайків або задані чіткі параметри пошуку, масив сортується за спаданням балів за допомогою методу sort. Для цього використовується лямбда-вираз lambda item: item[0], який вказує сортувати список саме за нульовим елементом кортежу, тобто за обчисленим балом скорингу. Якщо користувач новий і історії взаємодій ще немає (ситуація «холодного старту»), викликається функція random.shuffle. Вона випадково перемішує доступні анкети, що забезпечує рівні шанси для показу всіх оголошень у базі даних і робить стрічку унікальною для кожного оновлення.

3.4 Інструкція користувачу

Даний розділ містить опис послідовності дій для кінцевого користувача під час роботи з інформаційною системою прилаштування тварин. Всі скріни з цього пункту знаходяться в Додатку Б.

3.4.1 Технічні вимоги та встановлення додатку

Для коректної роботи мобільного додатку пристрій користувача повинен відповідати наступним мінімальним вимогам:

- Операційна система: Android 8.0 або iOS 12.0 і вище.
- Апаратні вимоги: наявність модуля GPS та доступ до мережі Інтернет.
- Встановлення: оскільки додаток розроблено на базі Expo, для тестування необхідно встановити середовище Expo Go або скористатися згенерованим інсталяційним файлом (.apk/.ipa).

3.4.2 Етап ознайомлення та входу в систему

При першому запуску мобільного додатка користувачеві демонструється екран ознайомлення з можливостями системи (рис. Б.1), який виконує роль візуальної презентації сервісу.

- Інформаційний блок: На екрані розміщено коротке гасло «Find your pet's new best friend», що окреслює основну мету додатка - пошук тварин для прилаштування або спільного вигулу.
- Елементи керування: * Кнопка «Get Started» ініціює процес реєстрації нового користувача, де необхідно буде вказати базові дані та налаштувати профіль власника або тварини.
- Кнопка «Log In» призначена для авторизації вже зареєстрованих користувачів через сервіс Supabase Auth.

Після обрання опції входу відкривається екран авторизації (рис. Б.2), реалізований на базі сервісу Supabase Auth.

- Введення даних: Користувач має вказати свою електронну адресу та пароль.
- Альтернативний вхід: Передбачена можливість авторизації через сторонні сервіси Google та Apple.

- Завершення входу: Натискання кнопки «Sign in» ініціює перевірку облікових даних на стороні сервера FastAPI та перехід до основного функціоналу.

3.4.3 Робота з модулями додавання, підбору та аналітики

Після входу в систему користувач отримує доступ до інтерактивної карти та системи свайпінгу.

- Головний екран: Користувач переглядає картки тварин, де алгоритм на базі FastAPI пропонує найбільш релевантні варіанти згідно з геопозицією (Рис. Б.3).

Окрім автоматичного підбору, реалізовано гнучку систему фільтрації, що дозволяє користувачу самостійно коригувати параметри пошуку в реальному часі. Використовуючи відповідне меню налаштувань (рис. Б.4), можна встановити максимальний радіус виявлення (Maximum Distance) за допомогою інтерактивного слайдера, обрати конкретний тип тварини або відфільтрувати анкети за статтю та віковим діапазоном. Кожна зміна фільтрів ініціює повторний запит до серверного API, який миттєво оновлює стек карток, забезпечуючи точну відповідність видачі індивідуальним запитам користувача.

Профіль користувача: Центральним модулем персоналізації системи є профіль користувача (рис. Б.5), який дозволяє керувати особистими даними та власними оголошеннями про тварин. На цьому екрані відображається основна інформація про власника: ім’я, контактна адреса та коротка біографія, що дозволяє іншим учасникам спільноти отримати первинне уявлення про користувача.

Основними функціональними можливостями даного модуля є:

- Керування власними анкетами: У розділі «Your pets» (рис. Б.5) власник може переглядати список усіх зареєстрованих ним тварин. Кожна картка містить стислу інформацію (ім’я, порода, стать). Для підтримки актуальності даних передбачені функції редагування та видалення анкет.

Натискання на іконку «олівця» відкриває екран редагування (рис. Б.6), де можна оновити фото тварини, змінити опис її поведінки або вік.

- Додавання нових оголошень: Через модальне вікно «Add pet» (рис. Б.7) користувач ініціює створення нового профілю тварини. Система вимагає заповнення ключових полів: кличка, тип, порода, стать та вік. Особливу увагу приділено полю «Behaviour» (Поведінка), дані з якого використовуються алгоритмом підбору для пошуку найбільш сумісних партнерів. Усі завантажені фотографії автоматично синхронізуються з хмарним сховищем Supabase Storage.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи виконано програмну реалізацію клієнтської та серверної частин інформаційної системи «pet.io». На основі технічних вимог обґрунтовано вибір технологічного стеку, що включає фреймворк React Native для мобільного інтерфейсу, асинхронний фреймворк FastAPI для бекенду на мові Python та хмарну платформу Supabase для автентифікації користувачів, збереження медіафайлів та керування базою даних PostgreSQL. Описано процес підготовки робочого місця розробника, включаючи конфігурацію локального середовища, підключення API-ключів та налаштування інструментів тестування.

Головну увагу в розділі приділено алгоритмізації бізнес-логіки додатка.

Розроблено та представлено у вигляді блок-схем три ключові алгоритми системи: реєстрацію та валідацію користувачів через Supabase Auth, роботу пошукового ендпоінту з багаторівневою геопросторовою фільтрацією та розв'язанням проблеми «холодного старту», а також логіку автоматичного створення збігів (метчів) і кімнат чату при виявленні взаємного інтересу сторін.

Наведений фрагмент програмного коду бекенду підтверджує практичну реалізацію розроблених алгоритмів. Аналіз коду показав, що використання асинхронних запитів з оператором await дозволяє серверу не блокувати потоки

очікуванням відповіді від бази даних, що економить апаратні ресурси при високих навантаженнях. Застосування реляційного об'єднання join та лямбда-виразів дозволило оптимізувати швидкість вибірки й трансформації даних. Таким чином, результати третього розділу сформували готову програмну основу системи для її подальшого тестування та розгортання.

4. ТЕСТУВАННЯ ТА РОЗГОРТАННЯ ПРОГРАМНОГО ПРОДУКТУ

4.1 Стратегія та види тестування

Стратегія тестування спеціального додатка для власників тварин передбачає перевірку працездатності клієнтської та серверної частин, а також коректності їхньої взаємодії. Контроль якості програмного продукту виконувався за принципом «чорної скриньки» для перевірки користувацького інтерфейсу та за допомогою спеціалізованих інструментів для валідації API. Головна увага приділялася надійності обміну даними між мобільним клієнтом на React Native, асинхронним сервером на FastAPI та хмарними сервісами Supabase.

Функціональне тестування проводилося ручним способом для перевірки відповідності реалізованих функцій вимогам до системи. Усі сценарії поведінки користувача тестувалися в середовищі розробки на iOS-емуляторі (симуляторі Xcode). Під час цього етапу перевірялися процеси реєстрації та авторизації, створення й редагування анкет домашніх улюбленців, правильність відображення карток у стрічці пошуку, а також відправка та отримання повідомлень. Окремо перевірялася реакція мобільного додатка на введення некоректних даних, наприклад, порожніх полів або неправильного формату електронної пошти. Це дозволило переконатися, що система виводить користувачеві відповідні попередження, а сервер повертає стандартні коди помилок без збоїв у роботі.

Інтеграційне тестування виконувалося для перевірки взаємодії між мобільним клієнтом, бекендом та хмарною базою даних. Оскільки архітектура системи побудована на асинхронних HTTP-запитах та авторизації за допомогою JWT-токенів, необхідно було підтвердити правильність передачі заголовків безпеки та обробки відповідей сервера.

Для ізольованої перевірки серверних ендпоінтів до їх підключення до інтерфейсу мобільного додатка використовувалися інструменти Postman та

Swagger UI. Це дозволило окремо протестувати роботу SQL-запитів з об'єднанням таблиць у PostgreSQL, точність роботи алгоритму геопросторової фільтрації анкет за заданим радіусом, а також стабільність каналів зв'язку для миттєвого оновлення списку чатів при появі взаємного лайку.

4.2 Сценарії тестування (Тест-кейси)

Для систематичної перевірки функціональності спеціального додатка для власників тварин було розроблено серію тест-кейсів. Вони охоплюють три основні логічні блоки системи: реєстрацію користувачів, алгоритм підбору анкет за геопозицією та логіку створення збігів при взаємному інтересі сторін. Усі перевірки клієнтської частини виконувалися на iOS-емуляторі.

Низькорівнева перевірка полів при створенні облікового запису через модуль Supabase Auth наведена в табл. 4.1.

Таблиця 4.1

Тест-кейс №1: Реєстрація та валідація користувачів

Крок	**Дія користувача / Системи**	**Очікуваний результат**	**Фактичний результат**	**Статус**
1	Надіслати форму реєстрації з порожніми полями	Помилка валідації: поля не можуть бути порожніми	Запит відхилено, виведено попередження	Відповідає
2	Ввести email у неправильному форматі (без знаку @)	Помилка: некоректний формат електронної пошти	Відображено повідомлення про помилку	Відповідає

4	Ввести email, який вже існує в базі даних Supabase	Помилка 400: користувач із такою поштою вже зареєстрований	Отримано код 400 від сервера, виведено повідомлення	Відповідає
5	Ввести коректні та унікальні дані користувача	Створення запису в Auth, генерація UUID, повернення JWT-токена	Акаунт створено, успішний вхід до додатка	Відповідає

Перевірка коректності створення профілю вихованця, валідації введених параметрів та інтеграції з хмарним сховищем медіафайлів наведена в табл. 4.2.

Таблиця 4.2

Тест-кейс №2: Додавання анкетної картки домашнього улюбленця

Крок	**Дія користувача / Системи**	**Очікуваний результат**	**Фактичний результат**	**Статус**
1	Надіслати форму додавання тварини з порожніми текстовими полями (кличка, вид)	Помилка валідації: обов'язкові поля мають бути заповнені	Клієнтська частина заблокувала відправку, виведено підказку	Відповідає
2	Ввести некоректне значення у	Перевірка типу даних: вік повинен	Форма видає помилку, кнопка	Відповідає

	поле віку тварини (наприклад, від'ємне число)	бути додатним числовим значенням	збереження неактивна	
3	Завантажити фотокартку тварини з галереї iOS-емулятора	Файл завантажується у бакет Supabase Storage, система повертає публічний URL посилання	Медіафайл успішно збережено у хмарі, згенеровано текстовий URL	Відповідає
4	Надіслати форму з повністю заповненими та валідними даними улюбленця	Сервер FastAPI повертає код 201 Created та записує об'єкт у таблицю pets	Дані додано в базу, анкету прив'язано до owner_id поточного користувача	Відповідає

Перевірка автоматичного створення сесії зв'язку між двома користувачами при взаємному лайку наведена в табл. 4.3.

Таблиця 4.3

Тест-кейс №3: Логіка створення збігу (метчу) та чату

Крок	**Дія користувача / Системи**	**Очікуваний результат**	**Фактичний результат**	**Статус**
1	Користувач А ставить лайк	Запис фіксується в таблиці swipes зі	Рядок додано в БД, збіг не створюється	Відповідає

	тварині Користувача Б	значенням is_liked = true		
2	Користувач Б ставить взаємний лайк тварині Користувача А	Сервер FastAPI виявляє зустрічний лайк у таблиці swipes	Система фіксує взаємний інтерес	Відповідає
3	Перевірка автоматичних тригерів бази даних після збігу	Автоматичне створення рядка в таблиці matches та нової кімнати в chats	Записи згенеровано, користувачі отримали доступ до чату	Відповідає

4.3 Результати тестування

Практична верифікація розроблених модулів підтвердила коректність роботи всієї системи та стабільність обміну даними між клієнтом і сервером. Під час тестування серверної частини за допомогою вбудованої документації Swagger UI було виконано серію запитів до ендпоінтів авторизації та керування анкетними даними. Усі запити повернули очікувані коди відповідей (зокрема, 200 OK для успішних операцій пошуку та 201 Created при внесенні нових записів до бази даних), а вихідні структури даних у форматі JSON чітко відповідали спроектованим схемам валідації. На рисунку 4.1 представлено приклад успішної генерації JSON-відповіді сервером FastAPI під час виклику ендпоінту підбору анкет.

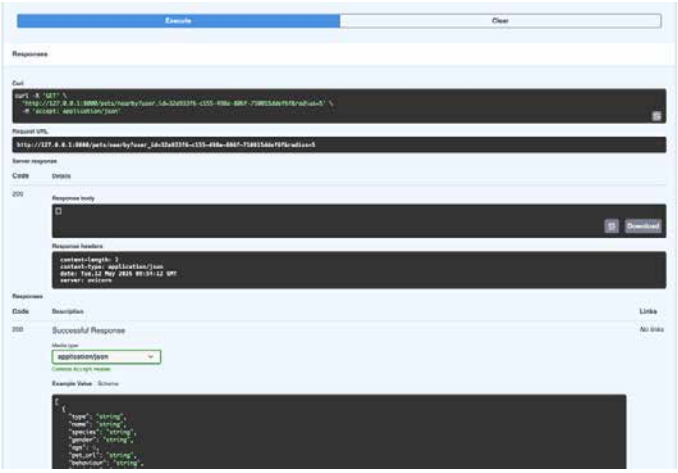

Рис. 4.1 - Результат ендпоїнту */pets/nearby,* коли користувач свайпнув усі анкети

Тестування користувацького інтерфейсу на iOS-емуляторі підтвердило правильність рендерингу графічних компонентів та швидку реакцію додатка на дії користувача. При переході між екранами не виявлено критичних затимок, асинхронне завантаження зображень тварин із хмарного сховища Supabase Storage відбувалося без помилок. Екран відображення карток улюбленців стабільно опрацьовує жести свайпів, оновлюючи стрічку виявлення та передаючи ідентифікатори реакцій на сервер без втрати пакетів даних. Перевірка інтерфейсу екрана створення профілю вихованця (рис. 4.2) показала, що додаток коректно блокує невалідні форми та відображає вибрані користувачем фотографії.

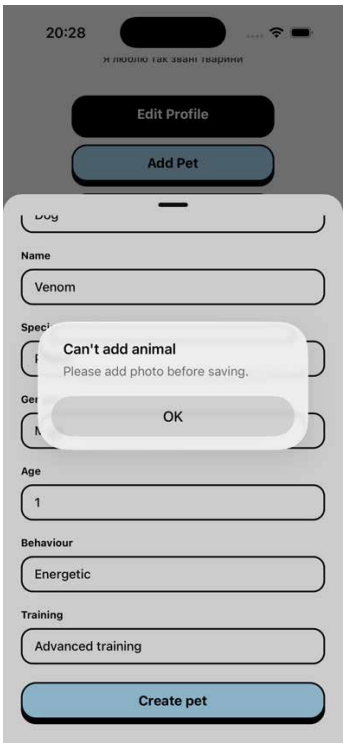

Рис. 4.2 - Інтерфейс додатка на iOS-емуляторі під час виконання сценарію додавання анкети вихованця

4.4 Вимоги до технічного та програмного забезпечення

Для забезпечення стабільної експлуатації та розгортання спеціального додатка для власників тварин було сформульовано вимоги до програмного та апаратного оточення як для клієнтської, так і для серверної частин системи.

Клієнтська частина додатка, реалізована на фреймворку React Native, розрахована на запуск на мобільних пристроях під керуванням операційних систем iOS або Android. Для мінімально стабільної роботи інтерфейсу, коректного відпрацювання анімацій карткових свайпів та асинхронного рендерингу зображень мобільний пристрій повинен мати процесор з архітектурою ARM, щонайменше 2 ГБ оперативної пам'яті та 100 МБ вільного простору на внутрішньому накопичувачі для кешування медіафайлів. Програмні вимоги обмежуються наявністю операційної системи iOS версії 13.0 і вище або Android версії 8.0 і вище. Крім того, обов'язальною умовою є наявність активного

підключення до мережі Інтернет через канали Wi-Fi або мобільний зв'язок для забезпечення взаємодії з бекендом, а також увімкнений модуль геолокації для роботи алгоритму пошуку тварин.

Серверна частина системи, що включає додаток на FastAPI та веб-сервер Uvicorn, оптимізована для розгортання в ізольованих Docker-контейнерах або на віртуальних приватних серверах (VPS). Мінімальні апаратні вимоги до сервера для обслуговування базового пулу користувачів включають 1 ядро центрального процесора (vCPU), 1 ГБ оперативної пам'яті та 10 ГБ дискового простору типу SSD для зберігання логів системи та конфігурацій. Рекомендована конфігурація для підвищення стійкості до навантажень передбачає наявність 2 ядер vCPU та 2 ГБ оперативної пам'яті. Програмне середовище сервера потребує встановленої операційної системи на базі ядра Linux (наприклад, Ubuntu Server 22.04 LTS), інтерпретатора Python версії 3.10 або вище, а також доступу до хмарної інфраструктури Supabase для роботи з базою даних PostgreSQL.

Висновки по розділу 4

У четвертому розділі проведено функціональне та інтеграційне тестування розробленого спеціального додатка для власників тварин, а також визначено технічні умови для його розгортання. За допомогою сформованих тест-кейсів було перевірено роботу ключових компонентів системи: створення облікових записів користувачів, додавання анкет улюбленців із завантаженням фотографій у хмарне сховище, а також роботу логіки створення збігів і кімнат чату при взаємному лайку. Перевірка інтерфейсу мобільного додатка на iOS-емуляторі підтвердила правильність відображення графічних елементів, а тестування серверної частини через Swagger UI та Postman довело коректність обробки HTTP-запитів та валідації вихідних даних у форматі JSON.

Також у розділі визначено мінімальні та рекомендовані системні вимоги для роботи додатка на мобільних телефонах користувачів та для серверної платформи. Побудована UML-діаграма розгортання зафіксувала фізичну

топологію системи та схему взаємодії між клієнтом на React Native, асинхронним бекендом на FastAPI та хмарними сервісами Supabase. Результати виконаних перевірок підтвердили відсутність критичних помилок, стабільність синхронізації даних та готовність програмного продукту до практичної експлуатації.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було розроблено спеціалізовану інформаційну систему для прилаштування тварин та соціалізації власників.

Проведено аналіз предметної області та існуючих рішень для пошуку домашніх улюбленців. Виявлено потребу в інтелектуальному інструменті, який би поєднував функції гіперлокального пошуку, соціальної взаємодії та аналітичного моніторингу для притулків. Це дозволило сформулювати технічне завдання на розробку системи, орієнтованої на кросплатформеність та Real-time взаємодію.

Обґрунтовано вибір технологічного стеку, що базується на поєднанні сучасних та ефективних інструментів: фреймворку React Native (з використанням Expo) для клієнтської частини, FastAPI для серверного API та хмарної платформи Supabase (PostgreSQL) для збереження даних.

Спроектовано архітектуру системи, яка включає детальну модель бази даних, що складається з взаємопов’язаних сутностей для керування користувачами, анкетами тварин, історією взаємодій (свайпів) та чатами. Розроблена діаграма пакетів та компонентів UML підтвердила модульність системи, що спрощує її подальше масштабування та підтримку.

Реалізовано інтелектуальний алгоритм підбору «Nearby Pets», який є ключовою науково-практичною складовою роботи. Алгоритм, розроблений на базі FastAPI, успішно поєднує геопросторову фільтрацію за формулою Haversine з механізмом предиктивного ранжування (Match Score) на основі вподобань користувача. Це забезпечує високу релевантність видачі та покращує користувацький досвід.

Розроблено функціонал аналітики та звітності, що має практичну цінність для адміністраторів притулків. Система дозволяє візуалізувати статистику прилаштування через інтерактивні дашборди та автоматично генерувати

офіційні звіти у форматі PDF, що значно скорочує час на ведення паперової документації.

Проведено тестування системи, результати якого підтвердили її працездатність та стабільність. Перевірка методом «чорної скриньки», а також інструментальне тестування API через Swagger UI підтвердили відсутність критичних помилок, коректність обробки геоданих та надійність синхронізації інформації з хмарною базою даних.

Особливу увагу приділено захисту інформації та розмежуванню прав доступу користувачів. Впровадження політик Row Level Security (RLS) на рівні СУБД PostgreSQL та безсесійного механізму автентифікації на базі токенів JSON Web Tokens (JWT) дозволило забезпечити надійну ізоляцію персональних даних і безпеку комунікації в Real-time чатах. Розроблена
схема захисту унеможливлює несанкціонований доступ або модифікацію анкет сторонніми особами, повністю задовольняючи сучасні вимоги до безпеки мобільних додатків.

Спроектована триланкова архітектура продемонструвала високу ефективність та низьку затримку при обробці просторових даних. Завдяки асинхронному виконанню запитів на сервері FastAPI та оптимізації реляційних об'єднань за допомогою інструментів SQLAlchemy, система раціонально використовує апаратні ресурси. Розподіл обчислювального навантаження між мобільним клієнтом на React Native та сервером дозволяє додатку стабільно функціонувати навіть в умовах нестабільного мережевого з'єднання, зберігаючи плавність роботи інтерфейсу.

Практична значущість отриманих результатів полягає у створенні готового до розгортання цифрового інструменту, що вирішує важливу проблему автоматизації процесів прилаштування безпритульних тварин і координації волонтерських зусиль. Автоматизація рутинних операцій, таких як геопросторовий підбір потенційних власників за критеріями сумісності та швидке формування офіційної PDF-звітності, дозволяє адміністраторам

притулків оптимізувати менеджмент і зосередити ресурси на безпосередній допомозі тваринам.

Перспективами подальшого розвитку проекту є інтеграція нейромережевих моделей для автоматичного розпізнавання порід та віку тварин за завантаженими фотокартками, впровадження розширеної системи пуш-сповіщень для мобільних пристроїв, а також розширення аналітичного модуля алгоритмами машинного навчання для прогнозування термінів прилаштування вихованців.

Таким чином, завдання бакалаврської кваліфікаційної роботи виконано в повному обсязі, а розроблена інформаційна система підтвердила свою працездатність і готовність до практичного впровадження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методичні вказівки до виконання бакалаврської кваліфікаційної роботи для студентів спеціальності 122 «Комп'ютерні науки». - К.: НУБіП України, 2024. - 45 с.
2. FastAPI Documentation. [Електронний ресурс]. - Режим доступу: https://fastapi.tiangolo.com/ (дата звернення: 12.05.2026).
3. React Native · A framework for building native apps using React. [Електронний ресурс]. - Режим доступу: https://reactnative.dev/ (дата звернення: 12.05.2026).
4. Supabase Documentation – The Open Source Firebase Alternative. [Електронний ресурс]. - Режим доступу: https://supabase.com/docs (дата звернення: 12.05.2026).
5. PostgreSQL 16.0 Documentation. [Електронний ресурс]. - Режим доступу: https://www.postgresql.org/docs/16/index.html (дата звернення: 12.05.2026).
6. Sinnott, R. W. Virtues of the Haversine // Sky and Telescope. - 1984. - Vol. 68, no. 2. - P. 159.
7. Python 3.10 Documentation. [Електронний ресурс]. - Режим доступу: https://docs.python.org/3.10/ (дата звернення: 12.05.2026).
8. Expo Documentation. [Електронний ресурс]. - Режим доступу: https://docs.expo.dev/ (дата звернення: 12.05.2026).
9. Tailwind CSS Documentation. [Електронний ресурс]. - Режим доступу: https://tailwindcss.com/docs (дата звернення: 12.05.2026).
10. NativeWind – Tailwind CSS for React Native. [Електронний ресурс]. - Режим доступу: https://www.nativewind.dev/ (дата звернення: 12.05.2026).
11. Pydantic – Data validation and settings management using Python type hints. [Електронний ресурс]. - Режим доступу: https://docs.pydantic.dev/ (дата звернення: 12.05.2026).

12. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. - К.: Держспоживстандарт України, 2016. - 16 с.
13. SQLAlchemy 2.0 Documentation. [Електронний ресурс]. - Режим доступу: https://docs.sqlalchemy.org/en/20/ (дата звернення: 12.05.2026).
14. Uvicorn – An ASGI web server, implementation for Python. [Електронний ресурс]. - Режим доступу: https://www.uvicorn.org/ (дата звернення: 12.05.2026).
15. PostGIS Spatial HTML Documentation. [Електронний ресурс]. - Режим доступу: https://postgis.net/documentation/ (дата звернення: 12.05.2026).
16. Introduction to JSON Web Tokens (JWT). [Електронний ресурс]. - Режим доступу: https://jwt.io/introduction/ (дата звернення: 12.05.2026).
17. ReportLab PDF Library User Guide. [Електронний ресурс]. - Режим доступу: https://www.reportlab.com/docs/reportlab-userguide.pdf (дата звернення: 12.05.2026).
18. MDN Web Docs: Working with JSON data. [Електронний ресурс]. - Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON (дата звернення: 12.05.2026).
19. Ткачук М. В. Архітектура та проектування програмного забезпечення. - Харків: НТУ «ХПІ», 2018. - 240 с.
20. Пасічник В. В., Резніченко В. А. Організація баз даних та знань. - К.: Видавнича група BHV, 2013. - 448 с.
21. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. - Prentice Hall, 2017. - 432 p.
22. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. - Addison-Wesley, 2003. - 208 p.
23. Ramalho L. Fluent Python: Clear, Concise, and Effective Programming. - O'Reilly Media, 2022. - 1100 p.
24. Obe R. O., Triessl L. S. PostGIS in Action. - Manning Publications, 2021. - 600 p.

25. Banks A., Porcello E. Learning React: Modern Patterns for Developing Real-World Applications. - O'Reilly Media, 2020. - 360 p.
26. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. - Addison-Wesley, 1994. - 432 p.
27. Lutz M. Learning Python. - O'Reilly Media, 2013. - 1648 p.
28. Richardson L., Ruby S. RESTful Web Services. - O'Reilly Media, 2007. - 448 p.
29. Голуб С. В. Моделювання інформаційних систем. - Черкаси: ЧНУ, 2019. - 212 с.
30. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. - Pearson, 2014. - 1440 p.

ДОДАТОК А

```
# Фрагмент коду ендпоінту Discovery
result = await db.execute(
        select(models.Pet, models.User)
        .join(models.PetOwner, models.Pet.owner_id == models.PetOwner.owner_id)
        .join(models.User, models.PetOwner.user_id == models.User.user_id)
    )

# Розрахунок дистанції та скорингу для кожної тварини
for pet, owner_user in result.all():
        # Do not show the current user's own pets.
        if owner_user.user_id == parsed_user_id:
            continue

        # Do not show pets the user has already acted on.
        if pet.pet_id in swiped_pet_ids:
            continue

        if (
            animal_type_normalized
            and animal_type_normalized != "all"
            and animal_type_normalized != _normalize_match_text(pet.type)
        ):
            continue

        if (
            interested_in_normalized
            and not _matches_interested_in(pet.species, interested_in_normalized)
        ):
            continue

        distance = haversine(lat1, lon1, owner_user.latitude, owner_user.longitude)
        if distance <= radius:
            score = calculate_pet_match_score(
                pet,
                interested_in=interested_in_normalized,
                liked_species=liked_species,
                liked_types=liked_types,
                liked_genders=liked_genders,
                disliked_species=disliked_species,
                disliked_types=disliked_types,
                disliked_genders=disliked_genders,
                distance=distance,
```

```
                radius=radius,
            )
            nearby.append(
                (
                    score,
                    {
                        "pet_id": pet.pet_id,
                        "owner_id": pet.owner_id,
                        "type": pet.type,
                        "name": pet.name,
                        "species": pet.species,
                        "gender": pet.gender,
                        "age": pet.age,
                        "pet_url": pet.pet_url,
                        "behaviour": pet.behaviour,
                        "training": pet.training,
                        "latitude": owner_user.latitude,
                        "longitude": owner_user.longitude,
                        "created_at": pet.created_at,
                    },
                )
            )

    if liked_pets or disliked_pets or interested_in_normalized:
        nearby.sort(key=lambda item: item[0], reverse=True)
    else:
        random.shuffle(nearby)
```

ДОДАТОК Б

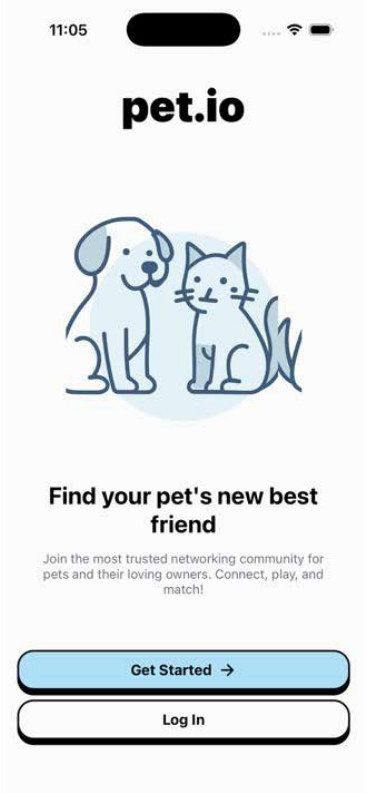

Рис. Б.1 – Анбордінг екран

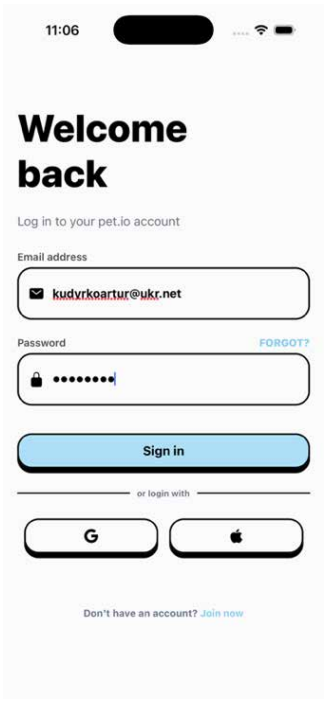

Рис. Б.2 – Екран логіну

Рис. Б.3 – Головний екран

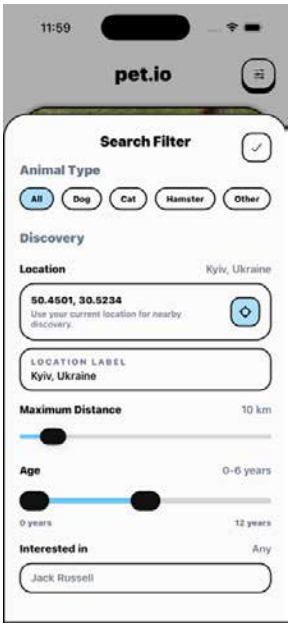

Рис. Б.4 – Фільтр пошуку

Рис. Б.5 - Профіль користувача

Рис. Б.6 - Редагування анкети

Рис. Б.7 – Додавання нової анкети

ДОДАТОК В

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Прилуцький Артур Валентинович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему « Спеціалізований додаток для власників тварин» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - [ ] інструменти генеративного ШІ не використовувалися.
    - [+] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
        - [+] перекладу іншомовних джерел;
        - [+] стилістичного редагування та перевірки граматики;
        - [+] допомоги у написанні/відлагодженні програмного коду;
        - [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 р. _____ **Артур ПРИЛУЦЬКИЙ**

(підпис)