

## ЗМІСТ

ВСТУП .....	6
РОЗДІЛ 1.....	8
Аналітичний огляд літературних та інших джерел.....	8
1.1. Методологічні засади дослідження.....	8
1.2. Соціальна та психологічна складові людини, як ключовий фактор створення системи.....	13
1.3. Аналіз відомих методів та рішень.....	14
Висновок до першого розділу.....	22
РОЗДІЛ 2.....	23
Системний аналіз об'єкта дослідження.....	23
2.1. Дерево цілей .....	23
2.2. Конкретизація функціонування системи .....	31
2.3. Побудова ієрархії задач .....	40
Висновок до другого розділу.....	42
РОЗДІЛ 3.....	43
Програмні засоби розв'язання задачі.....	43
3.1. Вибір та обґрунтування засобів розв'язання задачі .....	43
3.2. Технічні характеристики обраних програмних засобів розроблення.....	51
3.3. Машинне навчання та його методи.....	55
3.4. Вибір засобу реалізації машинного навчання.....	59
Висновок до третього розділу .....	63
РОЗДІЛ 4.....	64
Практична реалізація .....	64
4.1. Опис створеного програмного засобу.....	64
4.1.1. Загальні відомості про програму .....	64
4.1.2. Структура бази даних .....	64
4.1.3. Функції, що виконуються програмою .....	66
4.1.4. Опис логічної структури веб-сайту .....	68

4.1.5. Використовувані технічні засоби .....	68
4.1.6. Виклик та завантаження .....	68
4.1.7. Вхідні та вихідні дані .....	69
4.2. Інструкція користувача .....	69
4.2.1. Вступ.....	69
4.2.2. Загальні відомості про програму .....	69
4.2.3. Класи вирішуваних завдань .....	70
4.2.4. Опис основних характеристик і особливостей програми.....	71
4.2.5. Відомості про функціональні обмеження на застосування .....	71
4.3. Аналіз контрольного прикладу.....	71
Висновок до четвертого розділу.....	85
РОЗДІЛ 5.....	87
Економічне обґрунтування доцільності роботи .....	87
5.1. Економічна характеристика програмного продукту .....	87
5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення програмного продукту .....	87
5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ .....	88
5.4. Формування стратегічних альтернатив .....	90
5.5. Бюджетування .....	92
5.6. Вибір стратегії.....	95
Висновок до п'ятого розділу .....	96
ВИСНОВКИ .....	97
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ.....	99
ABSTRACT.....	104
ДОДАТОК А .....	106

## ВСТУП

У сучасному світі існує безліч проблем, які потребують вирішення. Більше того, з початку 21-ого століття люди шукають вирішення усіх своїх турбот саме в інтернеті. Однією з таких проблем є те, що власники домашніх тварин у випадку раптового відрядження чи бажання поїхати відпочивати не знають, де і з ким залишити свого улюбленаця. Особливо, це стосується самотніх людей, що переїхали в нове місто, або тих, хто не має знайомих, які б мали змогу та бажання надати тимчасовий притулок їхній тварині. Саме тому було вирішено створити web-систему каучсерфінгу для домашніх тварин на основі машинного навчання.

*Актуальність теми.* Данна система надає можливість знайти тимчасовий притулок домашній тваринці безоплатно. Варто додати, що на сьогодні існує інше вирішення цієї проблеми – це так звані «готелі для тварин». Переважно, це установи при ветеринарних клініках, де за певну ціну можна залишити свого домашнього улюбленаця на перетримку. Проте таких «готелів для тварин» є досить мало, а також ціна перетримки коштує досить дорого. Більше того, такі умови для тварин можуть бути надзвичайним стресом. Люди завжди більше довіряють людям, а усім світом керує принцип взаємодопомоги. Саме тому ідея створення системи, яка дозволить знайти тимчасового хоста домашній тваринці і при цьому не тільки не переживати за умови її проживання, але й познайомитись з однодумцем, є неабияк актуальною в наш час.

*Мета і задачі дослідження.* Метою створення даної системи є надання можливості власникам тварин у разі необхідності знайти тимчасовий притулок для свого домашнього улюбленаця. Також, люди, що не мають в кого залишити свою тваринку – це, зазвичай, самотні люди. Система також допоможе їм знайти людей з спільним інтересами. Іншою задачею дослідження є те, щоб люди, які часто подорожують не боялись заводити домашніх улюбленців, адже у них появиться сервіс, що допоможе їм знайти тимчасовий притулок для тваринки у разі необхідності.

*Для реалізації даної мети необхідно розв'язати такі задачі:*

- здійснити повноцінний аналіз предметної області. Це означає, що потрібно визначити засади дослідження, а також проаналізувати літературні та інтернет-джерела. Окрім того, варто розглянути схожі реалізовані рішення;
- потрібно побудувати архітектуру системи, побудувати дерево цілей, за критеріями визначити тип системи, а також провести системний аналіз, побудувавши діаграми процесів та ієархію задач;
- проаналізувати засоби, за допомогою, яких можна реалізувати систему. На основі аналізу та технічних характеристик обрати інструменти, що найкраще підійдуть для створення систем;
- реалізувати спроектовану систему та протестувати її на коректність роботи.

*Об'єкт дослідження:* процес знаходження та надання тимчасового притулку для домашніх тварин.

*Предмет дослідження:* засоби та методи надання тимчасового притулку для домашніх тварин.

*Практичне значення одержаних результатів.* Спроектована система реалізована у вигляді веб-сайту, задеплоїна на хостинг, а отже доступна цілодобово при наявності інтернету та необхідного веб-переглядача. Схожих реалізованих рішень не існує, а тому створення такої системи є досить цінним у наш час.

## РОЗДІЛ 1

### Аналітичний огляд літературних та інших джерел

#### **1.1. Методологічні засади дослідження**

Щодня кожна середньостатистична людина стикається з десятками, а інколи сотнями проблем. І в цьому швидкоплинному та «цифровому» світі, де все більшої популяризації набираються поняття такі, які «самотність», «депресія», «тривожність» їй стає важко покладатись лише самій на себе. Саме тому весь світ технологій, побудований на принципах – «допомогти», «вирішити проблему» та «полегшити життя» людині. Цією ідеєю також пронизана інформаційна web-система каучсерфінгу для домашніх тварин на основі машинного навчання.

Розглянемо основні тенденції створення такої системи. На сьогоднішній день майже усі збереження і опрацювання інформації в інтернеті відбувається за допомогою так званих «веб-орієнтованих» інформаційних систем, які можуть використовуватися в локальній мережі.

Вони виникають за допомогою цифрових додатків Web-Application, що виконують задані дії на веб-сервері в автоматичному порядку [1]. Веб-додатки, у цей час, користуються функціоналом браузерів в якості інтерфейсу. Серверні технології та орієнтування на клієнтів, у такому випадку, виконують роль засобів створення веб-простору.

Веб-орієнтовані інформаційні системи мають мало відмінностей (у процесі створення) від розробки програмних забезпечень: сюди, за методом аналогії, зараховуємо вимоги, реалізацію, аналіз, детальне структурування та тестинг. При цьому, використані технології не мають вагомої ролі під час створення кінцевого продукту, результат буде ідентичним: спочатку користувач робить запит для інформації за допомогою введення даних, безпосередньо працює з отриманим матеріалом, меню, а також має змогу зберегти корисну інформацію.

Головною відмінністю у створенні веб-орієнтованих продуктів та програм є їхня локалізація. Програмне забезпечення функціонує на віддалених серверах, з яких можна отримати інформацію, якщо поруч є доступ до інтернет-мережі [2].

Веб-орієнтовані системи мають низку загальних особливостей:

- можливість роботи багатьох користувачів одночасно;
- надійне використання;
- швидкість дії виконання;
- повна автономність стосовно операційних систем;
- незмінна локалізація;
- відсутність необхідності у використанні програм;
- користувач не має важливих функцій і не може виконувати роль адміністратора (за це відповідає розробник);
- невисока габаритність;
- витривалість;
- легкість у використанні;
- прихована внутрішня структура.

Такі системи можна назвати надійними через відсутність несправностей і помилок під час користування, адже, завдяки віддаленому доступу, повністю зникає необхідність щоденного перезавантаження серверу.

Другою тенденцією є подорожі. У 21-ому столітті люди почали все частіше і частіше подорожувати. Подорожі перестали бути лише частиною відпустки чи просто напрямком відрядження. Сьогодні відкриття нових обріїв – це хобі, відпочинок, частина світогляду, а інколи й спосіб життя.

Людина завжди прагнула знаходити щось нове, досліджувати та відкривати. За останніх 20 років подорожі стали набагато більш доступнішими. З'явилося значно більше рейсів у низькоціновій категорії від лоукостних авіакомпаній, завдяки яким можна добрatisь з одного кінця материка на інший за доступною ціною. Усього на сьогодні близько 130 авіакомпаній, які допоможуть зробити подорож бюджетною, а п'ятірку найпопулярніших очолюють Raynair, WizzAir, Jet, «Победа», Norwegian Air [3].

Одним з головних факторів, який повпливав на бюджетні подорожі стала популярність гостинних мереж, що побудовані на волонтерському принципі. Саме завдяки, найпопулярнішій гостинній мережі у світі CouchSurfing.org з'явилось і саме поняття «каучсерфінг», що стало невід'ємною частиною подорожей багатьох людей. Розглянемо детальніше, в чому його суть [4].

Гостинний туризм – це вдала заміна заїждженого відпочинку у класичних готелях та, часом не дуже присмінних, тісних хостелах. Під час такого часопроведення жителі іншої країни чи міста можуть на певний проміжок часу заселитись до будинку місцевих та жити за їхніми правилами. Деякі з гостинних людей окрім ліжка чи власної кімнати може запропонувати безкоштовну екскурсію, свою компанію, незвичні місця, про які знають лише корінні жителі. Саме завдяки цим можливостям гостинний туризм став одним із найпоширеніших методів для пізнання культурного простору певного куточка світу [5].

Каучсеферів можна умовно поділити на три групи, залежно від їхніх пропозицій:

1. Хости – люди, які пропонують всі умови проживання з власними правилами чи їх відсутністю.

2. Серфери – дружелюбні особи, які шукають компаньйонів для домашньої вечірки.

3. Ті, хто з радістю погодяться познайомити вас із містом, але не мають змоги надати вам місце для проживання.

CouchSurfing на даному етапі свого розвитку вже охопив близько двох мільйонів користувачів з усього світу [6]. З 2011 року гостинна мережа розвинулась настільки, що зараз її сміливо можна назвати найбільшою спільнотою з особистими правилами, світоглядною позицією та найвищою кількістю щасливих людей, які мають можливість піznати світ і познайомитись з цікавими людьми майже безкоштовно.

Проте так само, як зростає тенденція на подорожі через гостинні мережі, так само невпинно зростає тенденція бути у безпеці. Саме тому на проекті CouchSurfing багато уваги приділяється забезпеченням безпеки учасників. Для цього існує низка інструментів, що дають можливість дізнатися думку інших учасників про певного

користувача, система верифікації імені та адреси користувача. Крім цього, періодично адміністрація сайту розсилає всім учасникам інформацію про правила безпеки або про виявлених шахраїв, а також на одній з титульних сторінок докладно пояснюю, як залишатись в безпеці та правильно обирати хоста, використовуючи сайт. Крім того зібрано багато відгуків та статей від користувачів гостинних мереж, які діляться порадами щодо безпеки та як знайти правильного хоста для перебування в тому чи іншому місті [7].

Третію та не менш важливою тенденцією – є бажання людини дбати про когось. Скільки ваших знайомих має хоча б одну домашню тваринку? А скільки хотіли б мати, але переживають, що через неї будуть «приковані» до місця проживання?

Як згадувалось на початку, то у сучасному світі, де життя перейшло в цифровий режим, дуже поширилою стала самотність. Тому переважно саме самотні люди заводять собі домашніх тварин[8], адже вони часто допомагають закрити такі соціальні потреби, як дбати про когось, відчувати зв'язок з кимось та нести відповідальність за когось[9]. В даному випадку «кимось» постає домашня тваринка. Люди, які переїхали в інше місто чи країну і водночас мають домашню тваринку, яку привезли з собою, або ж придбали на новому місці, часто стикаються з проблемою, де ж її залишити, якщо тебе відправляють у термінове відрядження, чи у родичів раптова хвороба, або ж з'явилася необхідність погуляти на торжестві близьких друзів [10] - та хіба мало може знайтися приводів терміново зірватися з місця та полетіти чи поїхати кудись на декілька днів чи навіть тижнів?

Варто додати, що проблема «де залишати домашню тваринку» неабияк актуальна у століття фрілансерів. Адже багато людей, які можуть працювати з будь-якої точки світу і хочуть мати домашнього улюбленаця, або вже його мають, думають, що це певний «якір», який тримає їх на одному місці. А як відомо, то однією з найбільших переваг роботи на фрілансі є можливість подорожувати у будь-яку точку світу в будь-який час. І ось тут гостро постає питання: а куди ж дівати свого домашнього вихованця, за яким потрібно доглядати? І чи не буде він тим самим «якорем»?[11]

Розвинені клініки, які надають ветеринарні послуги, останнім часом почали вводити нову тенденцію у лікуванні тварин. Домашніх улюблениців, які перенесли лікування чи готуються до нього можна залишити у вет-готелі на певний час. Це поняття не варто поєднувати з умовами притулку, адже тимчасове житло для тварин у клініці має інший характер. Пацієнтам у таких умовах простіше надавати лікувальні послуги (особливо, якщо вони потребують цього цілодобово) з використанням професійних засобів, медикаментів та апаратної техніки. Це забезпечить надійність лікування, комфортні умови для проведення процедур і можливість швидкого одуження домашніх улюблениців.

Проте варто розуміти, що це не є звичним середовищем перебування для тварини і часто можна стикнутись з тим, що характеристики такого готелю зовсім не відповідають дійсності, а головне не варті ціни[12]. На даний час, такі готелі ще не набули достатньої популяризації, то ж можна буде стикнутись з проблемою того, що їх просто не буде у вашому місті. Також варто зазначити, що умови в готелі для тварин, підійдуть не усім, адже у домашніх улюблениців теж є своя психологія і почуття. Відомий зоопсихолог Дарія Вишеньська, яка активно користується просторами інтернету, у своїй роботі[13] вживає поняття "психологічні відмінності тварин". Все-таки коти та собаки мають різну психіку та відмінні залежності. Для перших великим стресом можна назвати постійну зміну оточення, у той час як для пса немає нічого страшнішого, ніж самотність. Саме з урахуванням цих критерій варто обирати притулок для свого улюбленаця.

Отже, можна зробити висновок, що інформаційна web-система каучсерфінгу для домашніх тварин є неабияк актуальною у час постійних подорожей та самотності спричиненої цифровою революцією у наших життях. Web-система допоможе людям знайти тимчасового хазяїну для його/її домашньої тваринки і зробити це безпечно завдяки системі відгуків та процесу верифікації хостів. Таким чином, можна буде вільно подорожувати на короткі терміни та не переживати за здоров'я та життя своїх вихованців.

## **1.2. Соціальна та психологічна складові людини, як ключовий фактор створення системи**

У загальній системі світового туризму існують так звані мережі гостинності. Так звикли називати корисні спільноти та угрупування мандрівників з різних куточків Землі, які дають змогу знайти місце проживання та компаньйонів для активного часопроведення. Кожен із членів таких угрупувань може запропонувати свої "послуги": кімнату для ночівлі, власного гіда, сніданки у великій компанії, вечірки щоночі тощо[14].

Каучсерфінг (див. детальний опис вище) на сьогодні є найбільшою гостинною системою у світі. Цей термін використовують на позначення спільноти в інтернет-просторі та соціальної практики у світовому туризмі, що з кожним днем набуває все більшої популярності та розгалужень у системі.

Окрім каучсерфінгу існує так звана соціальна практика (зміна суспільного становища та власний розвиток за допомогою світових організацій, закладів освіти, впливових установ тощо)[15]. Вагому роль у розвитку такого методу має життєдіяльність людей та способи її реалізації. У такому випадку не важливі біологічні характеристики людства, варто лише звернути увагу на соціальні угрупування, інтереси спільнот, життєву позицію тощо. Ці показники впливають на зміну суспільства, а отже, мають змогу змінити і регулювати систематично налаштовану діяльність людини [16].

Теорія соціальних практик досліджувалась в роботах багатьох авторів, таких, наприклад, як П. Бергер і Т. Лукман[17], К. Гирц[18], А. Шютц[19], П. Бурдье[20], Г. Гарфінкель[21], Е. Гіddenс[22], Дж. Тернер[23], К. Кнорр-Цетіна[24] і ін.

Таким чином, можна зробити висновок, що система для домашніх тваринок в основі матиме ті самі соціальні практики, адже, щоб залишити свого улюбленаця людині треба буде поспілкуватись та налагодити дружні зв'язки з потенційним тимчасовим хазяїном.

Одним з ключових факторів будь-якої мережі гостинності є взаємодопомога. Людина, як соціальна істота, на психологічному рівні прагне нести щось у соціум та

бути його невід'ємною частиною. Допомагаючи іншим живим істотам, ми починаємо відчувати свою важливість і потребність у цьому світі. Але ще більш важливою є взаємодопомога у певній спільноті адже переважно саме взаємодопомога стає ключовим фактором виживання будь-якої спільноти, не зважаючи на її кількісні розміри. Відомий професіонал у цій сфері П. Кропоткін обґрунтував конструктивну взаємодопомогу і протиставив її агресії [25]. На його думку, допомога "своїм" дає змогу виживати у цьому світі та буде надійну опору у вигляді майбутньої підтримки та взаємного ставлення.

Окрім того, багато відомих та успішних людей у своїх блогах часто зачіпають тему того, як важливо робити щось для близького. Допомагати іншим – ось справжній ключ до щастя[26]. Неважливо, чи ти Біл Гейтс чи простий офісний працівник, чи у тебе мільйон чи ти маєш стільки, скільки й інші, неважливо, що ти їж, звідки ти, але ти завжди маєш можливість допомагати іншим. У своєму блозі Джон Хол ділиться порадами, щодо того як помагати іншим і зазначає, що це відповідальність кожного, особливо, лідерів, адже вони здатні надихати до вчинків людей, що йдуть за ними [27].

Саме на принципах волонтерської мережі гостинності, соціальних практик, таких як взаємодопомога, обмін досвідом та знаннями, соціальна відповідальність буде базуватись система каучсерфінгу для домашніх тварин.

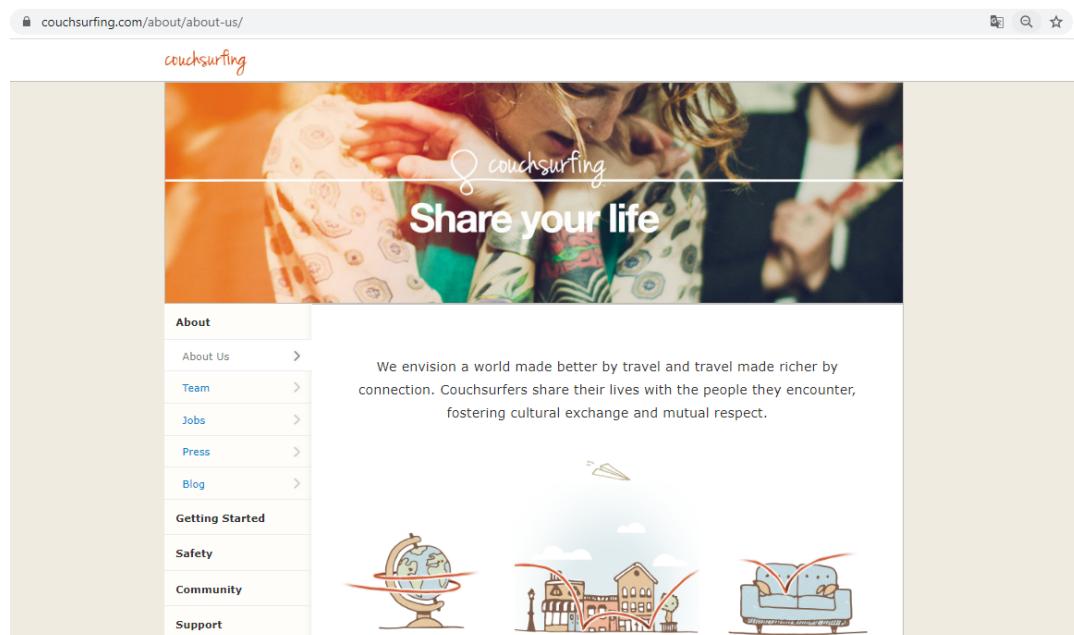
Згідно з проаналізованими джерелами в електронному та літературному форматі можна зробити висновок про те, що подібна соціальна практика знаходиться на стадії раннього розвитку та має кілька конструктивних матеріалів, загальна характеристика яких не може дати чіткої інформації. Досі недослідженим залишається питання стосовно каучсерфінгу: невідомий вектор його розвитку, внутрішні зміни, глибокий функціонал та інші особливості.

### **1.3. Аналіз відомих методів та рішень**

Оскільки, ідея створення мережі гостинності для домашніх тварин на основі практик, що популяризується через каучсерфінг, є новою, а подібних її рішень наразі

не існує, було обрано проаналізувати схожі системи, у яких в основі лежать соціальні практики та подорожі як спосіб життя. Найпопулярнішим таким сайтами є CouchSurfing, Servas, BeWelcome.

«CouchSurfing» – найбільша світова гостинна спільнота. Завдяки її структурі та можливостям, можна швидко знайти помічників у галузі туризму, зекономити на місці тимчасового проживання, познайомитись з цікавими особистостями тощо. "Серфінг по диванах" (дослівний переклад) влучно демонструє атмосферу спільноти [28].



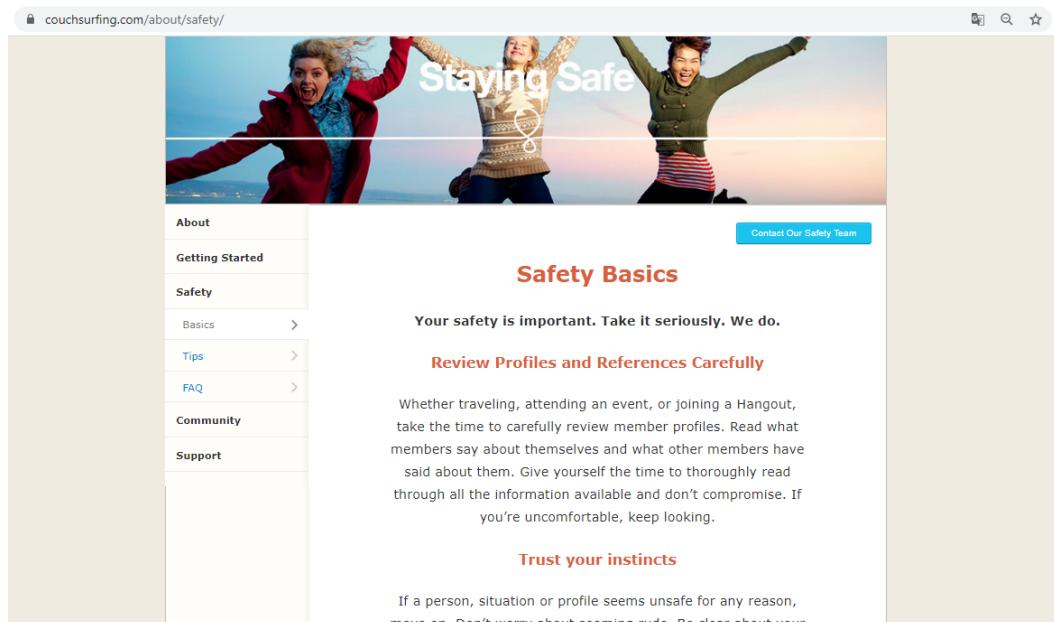
*Rис.1.1. Сторінка “About us” в мережі CouchSurfing*

У 2000 каучсерфінг можна було назвати лише допоміжним веб-сервісом, тоді як на сьогодні, у зв'язку із різким розвитком туризму, мережа набула всесвітнього рівня. Суть цього напрямку полягає не лише у пошуку місця для ночівлі чи бюджетного житла, вона набагато глибша. Люди, які вирішили стати членом світової спільноти, автоматично обрали шлях великого культурного розвитку, вивчення іноземних мов разом із носіями, географічну освіченість тощо. Його також називають соціальним феноменом, адже його популярність свідчить про перевагу соціальних практик, серед яких ключовою є взаємодопомога, над фінансовою вигодою. На даний момент, у системі зареєстровано більше 2 мільйонів користувачів та їхні можливості лише

розширяються. Зараз система пропонує також піти на подію з іншими каучсерферами чи навіть стати її організатором.

Перевагами використання є зручний та інтуїтивно зрозумілий дизайн, відмінна структуризація системи, наявність інструкцій та порад щодо використання, адаптивність на різних пристроях, наявність мобільного додатку для різних операційних систем, безпечність системи завдяки верифікації. Чудовим доповненням до системи є окремий сайт-блог[29], де розписані відповіді на усі можливі питання та поради від досвідчених каучсерферів.

Недоліком не було виявлено, адже система існує досить давно і постійно вдосконалюється, закриваючи усі потреби та побажання користувача.



*Рис. 1.2. Сторінка "Safety" в мережі CouchSurfing*

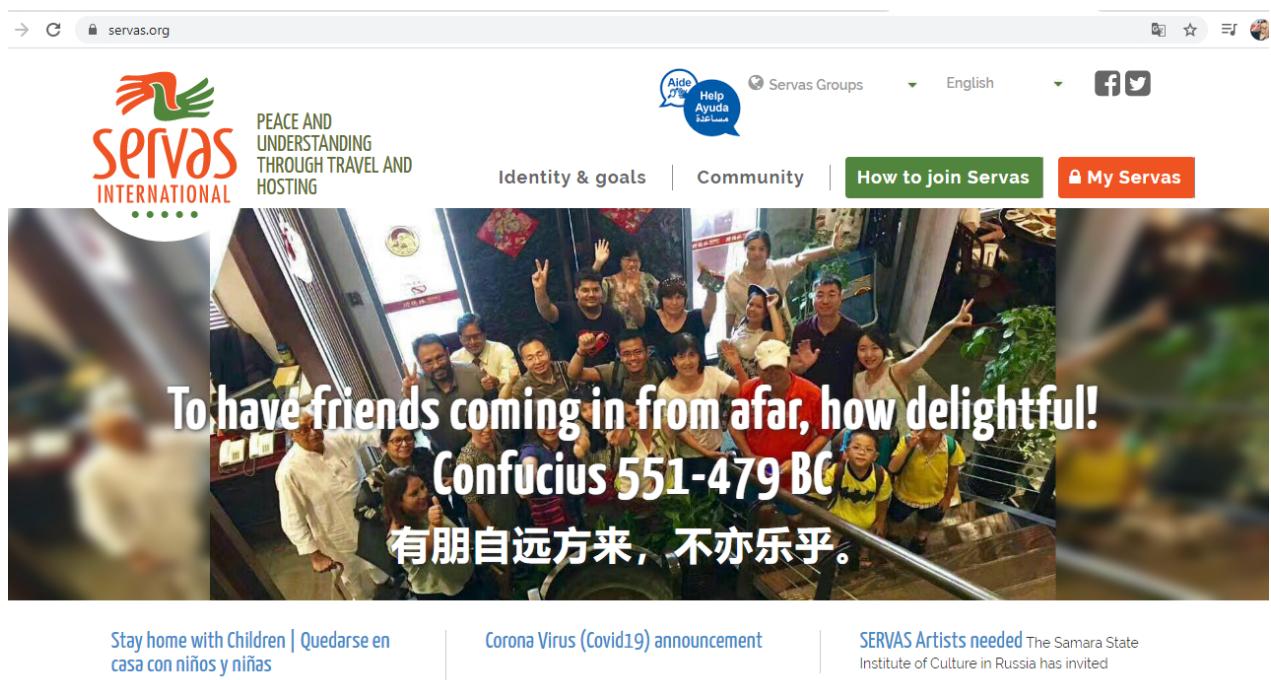
«Servas» (переклад з мови есперанто: "ми служимо") – це організація міжнародного рівня, яка не має ніякого прибутку, але позиціонує себе як активний провідник у світ мирного життя[30].

У 1949 році в Данії група людей-однодумців створила онлайн-спільноту "будівельників мирного світу". Причиною цього стало спустошення більшої частини населення у результаті Другої світової війни. Спільно з іншими миротворцями члени групи вирішили створити систему подорожей (подібну до каучсерфінгу), за

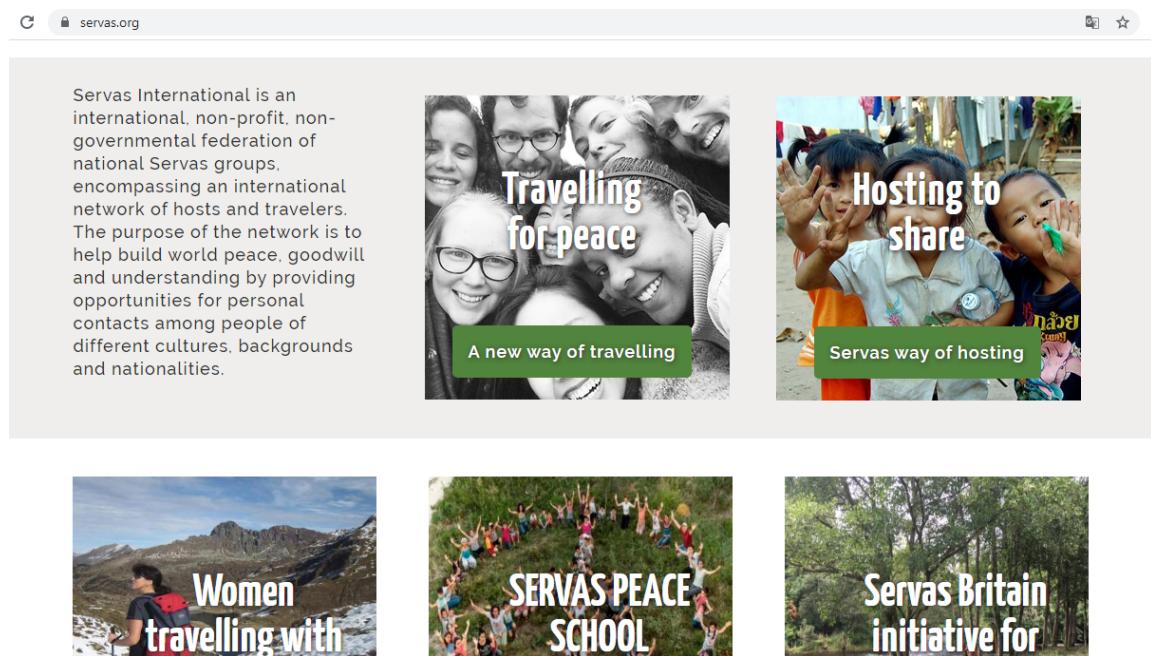
допомогою якої люди можуть вільно пересуватись світом, бути толерантними та знайомитись з новими особистостями, незалежно від країни, в якій вони мешкають.

Вже за кілька років після створення мережі, країни почали проявляти власну ініціативу та пропонувати у Сервасі свої послуги у наданні безкоштовного місця для ночівлі, довготривалого та короткочасного проживання тощо. Сервіс активно розповсюджувався і менше ніж за 5 років став одним із найбільших туристичних помічників у світі.

У 1972 році на міжнародній конференції мережі дали друге ім'я "Servas International" і документально зареєстрували її у Швейцарії. За цей час було створено офіцерський виконавчий комітет міжнародного рівня стосовно питань "вільного" туризму. І вже наступного року з офіційним реєстром у ЮНЕСКО, спільнота почала активізуватись[31].



*Рис. 1.3. Головна сторінка Servas International*



*Рис. 1.4. Головна сторінка Servas International*

Сервас офіційно поділено на три категорії:

- Servas Host (людина з іншої країни, у якої можна зупинитись з метою тимчасового проживання);
- Servas Day Host (людина, у якої не можна зупинятись, але вона з радістю проведе разом з вами час);
- Servas Traveller (людина, яка зупиняється).

Про кожного з них завжди вказана інформація: ПІБ, вік, адреса, на якій зареєстровано особистість, адреса проживання, номер телефону, розуміння мов, рівень спілкування на кожній з них, інтереси, додаткова інформація (шкідливі звички, корисні звички, хобі, спосіб життя, домашні тварини тощо).

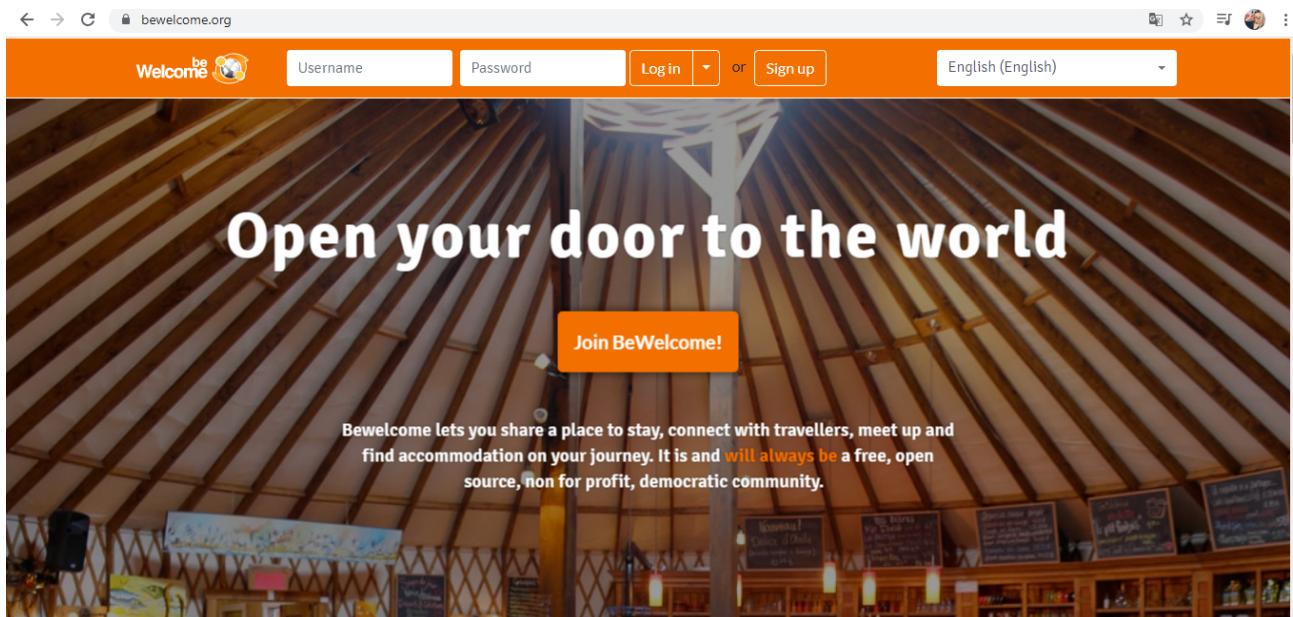
Існують певні неофіційні правила користування мережею, які виникли на основі багаторічного досвіду:

1. Зупинятись варто на 1-2 ночі (власник може сам запропонувати жити у нього більше);
2. Спілкування з господарем дому відбувається за його ініціативи, наявності вільного часу та бажання.

Якщо мешканець України вирішив стати Host/Day Host, його автоматично буде зараховано до списку та відображеного для населення інших країн.

Беззаперечною перевагою цього веб-ресурсу є його наповненість. Дуже багато різної інформації, що може стати у нагоді чи зробити подорож ще більш незабутньою та створити автентичний досвід.

Недоліками є те, що веб-система платна (ціна залежить від країни, у якій ти реєструєшся) і те, що за попередньою домовленістю мандрівника можуть прийняти не більше, ніж на 2 дні. Чи розраховувати на більше він не може знати одразу. Також варто зазначити, що процес реєстрації є досить складним, що може відштовхнути потенційного користувача.



*Рис. 1.5. Головна сторінка BeWelcome.org*

BeWelcome – подібний до інших веб-сайт, метою якого є пошук місця ночівлі та гостинного часопроведення[32].

Платформа побудована на чотирьох принципах:

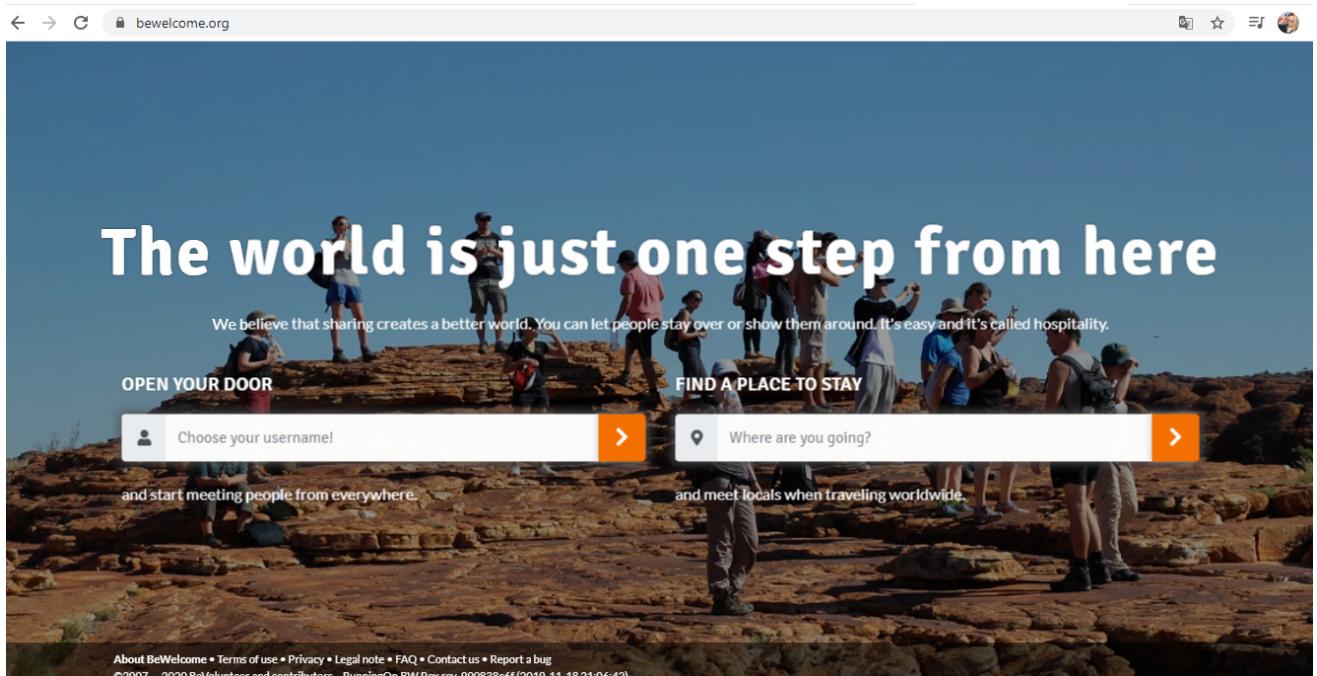
Зустрічай. Спрямований на онлайн-знакомство з подальшою можливістю спілкування наживо;

Подорожуй. Мотивує до організації подорожі з поки що незнайомими людьми для вдалого часопроведення, нових емоцій і вражень;

Приймай. Спільнота пропонує прості засоби для зв'язку із зацікавленими особами, які мають аналогічні життєві пріоритети;

Ділись. Компанія підкреслює важливість життєвого досвіду та обміну ним з іншими особистостями.

Мережа BeWelcome має у своєму керівництві організацію BeVolunteer, яка функціонує на волонтерських засадах некомерційного характеру. Асоціація добровільного напрямку зареєстрована у трьох країнах (Франція, Бретань, Ренні) та має веб-сайт з вільним програмним забезпеченням, оскільки сервер, яким користується компанія (BW-rox) є абсолютно безкоштовним[33].



*Рис. 1.6. Головна сторінка BeWelcome.org.*

Перевагами ресурсу є його безкоштовність, легкість у використанні, адаптивність та зрозумілість інтерфейсу. Ресурс був оновлений у 2019 році, тому працює правильно завдяки новим технологіям.

До недоліків можна віднести не впізнаваність та низьку конкурентну спроможність за рахунок того, що сайт волонтерський. Сервіс нічим не виділяється та не закриває усіх потреб користувача. Через невпізнаваність в системі зареєстровано не так багато користувачів, а тому буде важко знайти хоста у не дуже популярних серед туристів містах.

Усі переваги та недоліки представлено у таблиці 1.1 для того, щоб порівняти сервіси між собою.

*Таблиця 1.1.*

**Таблиця порівняння ресурсів**

Ресурси	Переваги	Недоліки
CouchSurfing	<ol style="list-style-type: none"> <li>1. Зручний та інтуїтивно зрозумілий для користувача.</li> <li>2. Відмінна структуризація системи.</li> <li>3. Наявність порад та інструкцій щодо користування сервісом.</li> <li>4. Адаптивність на різних пристроях.</li> <li>5. Наявність мобільного додатку на різних ОС.</li> <li>6. Продумана безпечність системи.</li> <li>7. Верифікація користувачів.</li> <li>8. Блог, у якому можна знайти відповіді майже на усі питання.</li> <li>9. Понад 2 млн користувачів.</li> </ol>	Система існує досить давно та постійно підтримується спільнотою розробників, тому недоліків не було виявлено.
Servas	<ol style="list-style-type: none"> <li>1. Безперечною перевагою є підтримування сайту ЮНЕСКО.</li> <li>2. Зрозумілий інтерфейс.</li> <li>3. Адаптивність веб-сайтів.</li> <li>4. Безпечність системи.</li> <li>5. Наповненість. Можна знайти різну інформацію, що може стати у нагоді або зробити подорож більш незабутньою.</li> <li>6. Наявність філій в Україні.</li> <li>7. Велика база користувачів.</li> </ol>	<ol style="list-style-type: none"> <li>1. Процес реєстрації є досить складним – це може, взагалі, відштовхнути від користування сервісом.</li> <li>2. Веб-система платна – ціна залежить від країни, у якій ти реєструєшся.</li> <li>3. За попередньою домовленістю гостя можуть прийняти лише на 2 ночі.</li> </ol>
BeWelcome	<ol style="list-style-type: none"> <li>1. Безкоштовне користування сервісом.</li> <li>2. Простий дизайн, що є досить зручним в користуванні.</li> <li>3. Адаптивний веб-сайт, написаний, використовуючи нові технології.</li> </ol>	<ol style="list-style-type: none"> <li>1. Сервіс волонтерський, а тому розвивається дуже повільно.</li> <li>2. Відсутність мобільного додатку.</li> <li>3. Вузький функціонал, що не закриває усіх потреб користувача.</li> </ol>

Дослідивши функціонал та можливості ресурсів стає очевидно, що при розробці системи для домашніх тварин варто посилатись на одну з найбільших мереж гостинності для людей - CouchSurfing, ще більше удосконаливши її. Саме тому розроблювана інформаційна веб-система, характеризуватиметься такими особливостями:

- веб-система буде охоплювати якнайбільшу кількість країн та міст, у яких можуть зареєструватись потенційні користувачі;
- доступний функціонал веб-системи залежатиме від ступені верифікації користувача у цілях безпеки;
- основною валютою веб-системи буде відгук, а зацікавленість у користуванні веб-системою буде підтримувати гейміфікований підхід;
- веб-система буде побудована з використанням машинного навчання, що зекономить час юзера та полегшить користування сервісом.

### **Висновок до первого розділу**

Опрацювавши літературні та електронні джерела здійснено висновки про те, що сервіси, які побудовані на використанні соціальних практик, користуються великою популярністю серед людей з різних країн. У 21-ому столітті подорожі щороку стають невід'ємною частиною життям мільйонів людей, тому питання: «Де ж залишити свого домашнього улюблена на декілька днів?» - залишається відкритим. Реалізована у майбутньому система каучсерфінгу для домашніх тварин не тільки допоможе вирішити це питання, але й принесе нові знайомства самотнім людям. Проведений аналіз підтверджує те, що система швидко стане популярною, адже соціальна складова людина прагне до взаємодопомоги більше, аніж до фінансової вигоди.

## РОЗДІЛ 2

### Системний аналіз об'єкта дослідження

#### **2.1. Дерево цілей**

Системний аналіз – це метод вирішення проблем, який передбачає розгляд ширшої системи, розбиття її на частини та деталізацію процесів, щоб з'ясувати чи досягне систему мету свого функціонування [34].

Для початку розглянемо, що таке система. Система – це загальний набір компонентів, частин або ступенів, які з'єднані між собою, щоб утворити більш складне ціле.

Мета застосування системного аналізу, являє собою можливість підвищити ступінь аргументованого конкретного рішення, що затверджується. За допомогою даного аналізу у архітектора системи є можливість проаналізувати різні можливі типи функціонування систему, після чого обрати найкращий та найоптимальніший.

Під час проведення системного аналізу однією з головних задач є визначення чинників, які потенційно можуть впливати на систему позитивно або негативно .

Основу системного аналізу являє собою системний підхід. Його суть полягає в дослідженні об'єкта як множини елементів, що зв'язані між собою. Він дозволяє розглядати систему одночасно і як підсистему вищестоячих рівнів, і як єдине ціле.

Існує декілька принципів системного підходу, за допомогою яких стає можливо правильно дослідити систему. Основні такі принципи: принцип остаточної мети, зв'язності, модульності, функціональності, децентралізації, невизначеності та розвитку.

Проаналізувавши систему каучсерфінгу для домашніх тварин можна стверджувати:

- принцип остаточної мети передбачає собою створення системи каучсерфінгу для домашніх тварин, за допомогою якої користувач зможе найти тимчасовий притулок для свого домашнього улюблена;
- принцип зв'язності надає можливість знайти внутрішні зв'язки і залежності між різними елементами досліджуваної системи, а також зовнішні

зв'язки з навколоишнім середовищем, в якому вона буде функціонувати та вплив на нього. Також дозволяє правильно розставити пріоритети при побудові системи;

- принцип модульності розглядає систему як сукупність модулів. Таким чином надає можливість працювати з кожним модулем окремо.
- принцип функціональності передбачає собою детальний розгляд усіх функцій даної системи та проектування її структури. При цьому під час аналізу спершу необхідно надати пріоритет функціям і лише після цього приступати до побудови структури;
- принцип децентралізації полягає у розподіленні прав між адміністратором, розробником та користувачам. В системі каучсерфінгу для домашніх тварин, користувачі мають змогу писати відгуки, але не мають змоги їх редагувати чи видаляти, а також змінювати інформацію в будь-якому іншому профілі, окрім свого.
- принцип невизначеності дозволяє проаналізувати та вирахувати можливі невизначені точки поведінки системи та мінімізувати ризик їхньої появи;
- принцип розвитку полягає в доречності постійно розширювати та покращувати функціонал системи, в залежності від потреб користувачів та змін у світі. Система каучсерфінгу для домашніх тварин має здатність до оновлення, додавання нових функцій та можливостей для того, щоб бути зручною для користувача.

Побудовано дерево цілей для проведення системного аналізу системи каучсерфінгу для домашніх тварин на основі машинного навчання. Древо цілей – це певна ієархія підцілей, що застосовується для визначення головного напряму дій.

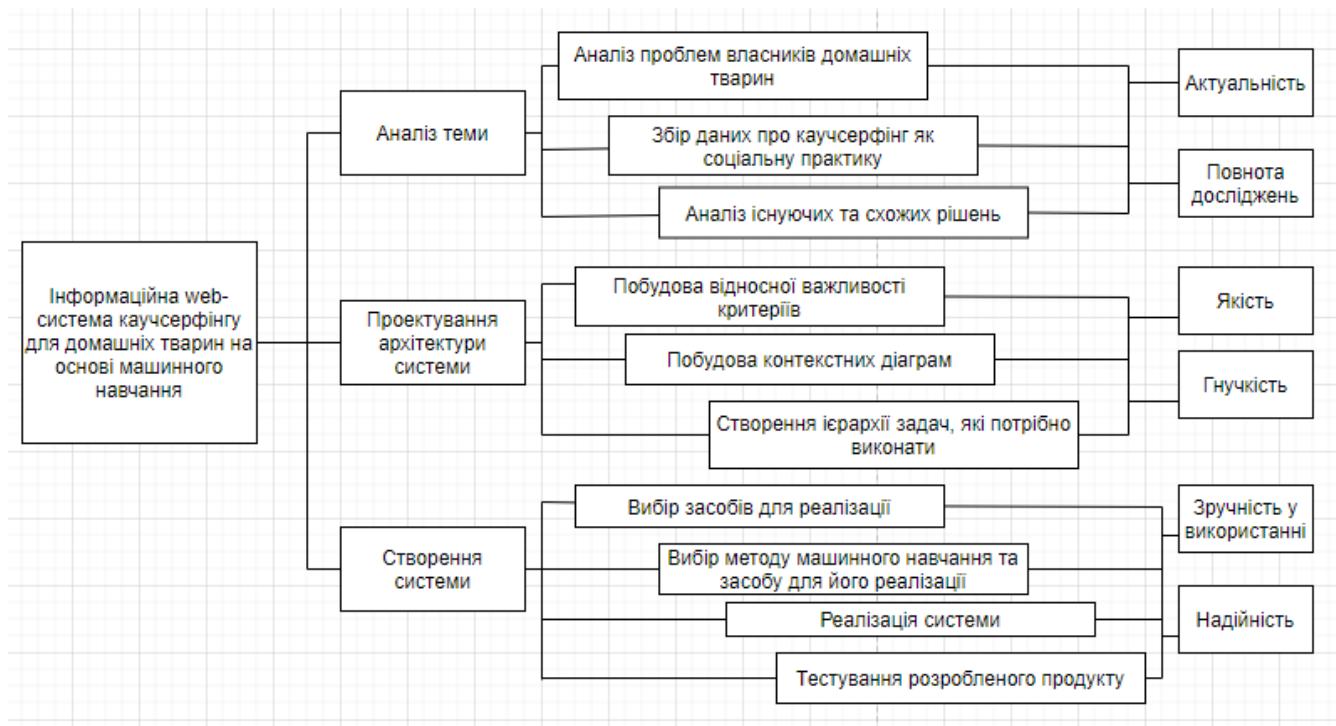


Рис. 2.1. Дерево цілей

Побудоване дерево цілей має три рівні: головна мета, її аспекти та підаспекти, а також критерії, що визначають якість функціонування системи.

Головні аспекти досягнення мети:

1. аналіз теми;
2. проектування архітектури системи;
3. створення системи.

Розглянемо детальніше кожен аспект спрямований на досягнення мети. Варто зазначити, що перший аспект є одним із найважливіших при системному аналізі, адже саме правильний аналіз системної області є міцним фундаментом для проектування та побудови самої системи і від його якості залежать інші два аспекти.

Підаспекти, що входять до аспекту «Аналіз теми»:

- аналіз проблем власників домашніх тварин. Для того, щоб зрозуміти чи буде актуальною система, варто проаналізувати, з якими саме проблемами стикаються власники домашніх тварин та де шукають їх вирішення;
- збір даних про каучсерфінг як соціальну практику допоможе зрозуміти психологічну і соціальну складові людини, які будуть ключовими факторами при

використанні системи. За допомогою цих даних буде зрозуміло чи стане система популярною серед користувачів.

- аналіз існуючих та схожих рішень. Неабияк важливим завданням є проаналізувати існуючі та схожі рішення, а саме виявити їх недоліки, щоб система було кращою за аналоги та закривала усі потреби користувачів.

Критеріями першого аспекту обрано актуальність та повноту досліджень. Одним з ключових критеріїв будь-якої системи є саме актуальність, а ще більш важливо при мінливості сучасного світу, де все дуже швидко розвивається. Повнота дослідження передбачає більш поглиблений пошук інформації та аналіз дотичних сфер, які краще розкриють бачення системи.

Другим аспектом є проектування системи, яке передбачає визначення та деталізацію основних задач, функцій та процесів системи. Критеріями цього аспекту визначено якість та гнучкість, адже система має бути спроектована так, щоб можна було вносити зміни та розширювати функціонал. Для коректного проектування варто виконати такі підаспекти:

- створення ієрархії задач, які потрібно виконати. На цьому етапі необхідно прописати усі задачі та підзадачі, послідовність їх виконання, щоб у результаті їх виконання була реалізована система.
- побудова контекстної діаграми. Необхідно побудувати IDEF0 діаграму, в якій буде розглянуто логічні зв'язки між роботами у вигляді певної ієрархії. Це допоможе краще зрозуміти процес побудови системи.
- побудова відносної важливості критеріїв. Потрібно провести аналіз усіх критеріїв, надавши їм оцінки пріоритетності, щоб у майбутньому реалізована система працювала правильно.

Наступним і останнім аспектом є реалізація самої системи. Необхідно забезпечити швидкість системи, оптимальну роботу навіть при великій відвідуваності, безпеку для усіх користувачів, а також зрозумілий інтерфейс і розумну систему, що допоможе користувачу у виборі. Саме тому для реалізованої

системи було обрано критерії – зручність у використанні та надійність. Для виконання третього аспекту потрібно реалізувати такі цілі:

- вибір засобів для реалізації. Для початку, необхідно дослідити та проаналізувати усі існуючі технології та інструменти, за допомогою яких створюють веб-сайти. Після цього, обрати ті, які допоможуть реалізувати бажаний функціонал. При виборі варто також опиратись на те, чи технологія підтримується розробниками, оновлюється та розширює свій функціонал, щоб система була безпечною та гнучкою;
- реалізація системи. Одним з ключових етапів є поступова та коректна реалізація системи з урахуванням усіх критеріїв, необхідних функцій та можливих ризиків;
- тестування розробленого продукту. Останнім і заключним етапом є саме тестування розробленого продукту. Воно необхідне для того, щоб виявити усі можливі помилки при користуванні системою та виправити їх. При цьому тестування варто повторити декілька разів.

Розглянемо більш детальніше типи систем. За рівнем автоматизації їх поділяють на інформаційно-пошукові, інформаційно-довідкові, інформаційно-управляючі, інтелектуальні та системи прийняття рішень.

Інформаційно-пошукові системи призначені для пошуку в базах даних текстових документів, їх фрагментів тощо, в яких завершальна обробка даних не передбачена.

Інформаційно-довідкові системи забезпечують швидкий пошук та доступ до необхідних даних.

Інформаційно-управляючі системи створюються для автоматизованого вирішення задач управління різного типу.

Інтелектуальні системи – це системи, що побудовані на знаннях та зазвичай містять у собі запрограмований штучний інтелект.

Системи підтримки прийняття рішень – це система, зазвичай інтерактивна, яка спрямована на допомогу прийняття того чи іншого рішення.

Серед останніх двох систем більш детально розрізняють:

- розрахунково-логічні системи, що надають можливість користувачам-непрограмістам в діалозі з ЕОМ розв'язувати задачі з використанням різних складних програм та методів;
- інтелектуальні інформаційні-пошукові системи – це комплекс програмних, лінгвістичних та логіко-математичних засобів, реалізація яких має основне завдання: створення підтримки діяльності людини та пошуку інформації у режимі досить розширеного діалогу природною мовою;
- експертні системи, що передбачають можливість впровадження різних областей комп’ютеризації. В даних системах майже неможливо використання математичних моделей.

Для визначення типу системи каучсерфінгу для домашніх тварин обрано метод аналізу ієархій. Для цього виділено чотири типи альтернатив: інформаційно-пошукову (A1), інформаційно-довідкові (A2), систему прийняття рішень (A3), інтелектуальна інформаційно-пошукова система (A4).

Наступний крок - необхідно обрати критерії якості для вибору типу системи: актуальність (К1), повнота дослідження (К2), якість (К3), гнучкість (К4), зручність у використанні (К5) та надійність (К6).

Далі потрібно побудувати матриці попарних порівнянь критеріїв якості, матрицю попарних порівнянь альтернатив. Після цього розрахувати власні числа (ВЧ) та вектори (ВВ). Для визначення експертної оцінки використано шкалу відносної важливості пріоритетів. Дану шкалу подано в таблиці 2.1.

*Таблиця 2.1.*

### **Шкала відносної важливості пріоритетів**

<b>Значення</b>	<b>Якісна характеристика</b>
1	Рівноцінні елементи
2	Несуттєвий пріоритет
3	Слабкий пріоритет
4	Помірний пріоритет
5	Значний пріоритет
6	Істотний пріоритет

Продовження табл. 2.1.

7	Сильний пріоритет
8	Дуже сильний пріоритет
9	Безумовний пріоритет

На поданій матриці (матриця критеріїв) зображенено відносну важливість критеріїв у порівнянні з іншими.

	K1	K2	K3	K4	K5	K6	<b>VЧ</b>	<b>BB</b>
K1	1,00	0,50	0,16	0,20	0,20	0,14	0,28	0,03
K2	2,00	1,00	0,16	0,20	0,16	0,14	0,34	0,04
K3	6,00	6,00	1,00	0,50	3,00	2,00	2,18	0,26
K4	5,00	5,00	2,00	1,00	4,00	0,50	2,15	0,25
K5	5,00	6,00	0,33	0,25	1,00	0,25	0,92	0,11
K6	7,00	7,00	0,50	2,00	4,00	1,00	2,41	0,29

Для розрахунку власних чисел (ВЧ) застосовують наступну формулу:

$$V\mathcal{C} = \sqrt[n]{\prod_{j=1}^n a_{ij}}. \quad (2.1)$$

Для розрахунку власних векторів (BB) – формулу 2.2:

$$BB = \frac{w_i}{\sum_{i=1}^n w_i}, \quad (2.2)$$

Наступним кроком є побудова матриці парних порівнянь альтернатив окремо для кожного критерія, а також відносно головної цілі:

- для критерія «актуальність»:

	A1	A2	A3	A4	<b>VЧ</b>	<b>BB</b>
A1	1,00	0,50	0,33	0,33	0,62	0,14
A2	2,00	1,00	0,50	0,25	0,79	0,18
A3	3,00	2,00	1,00	0,50	1,2	0,28
A4	3,00	4,00	2,00	1,00	1,7	0,4

- для критерія «повнота дослідження»:

	A1	A2	A3	A4	<b>VЧ</b>	<b>BB</b>
A1	1,00	0,50	0,33	0,50	0,65	0,15
A2	2,00	1,00	3,00	0,25	1,06	0,25
A3	3,00	0,33	1,00	0,33	0,82	0,19
A4	2,00	4,00	3,00	1,00	1,69	0,4

- для критерія «якість»:

	A1	A2	A3	A4	<b>ВЧ</b>	<b>ВВ</b>
A1	1,00	0,50	0,33	0,33	0,62	0,14
A2	2,00	1,00	3,00	0,20	1,03	0,3
A3	3,00	0,33	1,00	0,50	0,89	0,21
A4	3,00	5,00	2,00	1,00	1,76	0,41

- для критерія «гнучкість»:

	A1	A2	A3	A4	<b>ВЧ</b>	<b>ВВ</b>
A1	1,00	2,00	0,33	0,25	0,74	0,17
A2	0,50	1,00	0,33	0,25	0,58	0,13
A3	3,00	3,00	1,00	0,33	1,2	0,27
A4	4,00	4,00	3,00	1,00	1,91	0,43

- для критерія «зручність у використанні»:

	A1	A2	A3	A4	<b>ВЧ</b>	<b>ВВ</b>
A1	1,00	0,25	0,50	0,33	0,59	0,14
A2	4,00	1,00	0,50	0,33	0,93	0,22
A3	2,00	2,00	1,00	4,00	1,59	0,37
A4	3,00	3,00	0,25	1,00	1,15	0,27

- для критерія «надійність»:

	A1	A2	A3	A4	<b>ВЧ</b>	<b>ВВ</b>
A1	1,00	0,33	0,50	0,25	0,59	0,14
A2	3,00	1,00	0,33	0,25	0,79	0,18
A3	2,00	3,00	1,00	2,00	1,51	0,35
A4	4,00	4,00	0,50	1,00	1,41	0,33

- відносно головної цілі:

	A1	A2	A3	A4	<b>ВЧ</b>	<b>ВВ</b>
A1	1,00	0,20	0,50	0,33	0,57	0,13
A2	5,00	1,00	0,25	0,25	0,82	0,11
A3	2,00	4,00	1,00	0,33	1,18	0,26
A4	4,00	4,00	3,00	1,00	1,91	0,43

Заключним етапом слугує побудова матриці порівнянь альтернатив. В залежності від її результатів необхідно визначити, який саме тип інформаційної системи для реалізації каучсерфінгу для домашніх тварин.

В таблиці 2.2 показано матрицю порівнянь альтернатив.

*Таблиця 2.2*

### **Матриця порівняння альтернатив**

<b>Критерій</b>	<b>K1</b>	<b>K2</b>	<b>K3</b>	<b>K4</b>	<b>K5</b>	<b>K6</b>	<b>Узагальнені пріоритети</b>
	0,03	0,04	0,26	0,25	0,11	0,29	
<b>A1</b>	0,14	0,15	0,14	0,17	0,14	0,14	0,15
<b>A2</b>	0,18	0,25	0,3	0,13	0,22	0,18	0,21
<b>A3</b>	0,28	0,19	0,21	0,27	0,37	0,35	0,29
<b>A4</b>	0,4	0,4	0,41	0,43	0,27	0,33	0,37

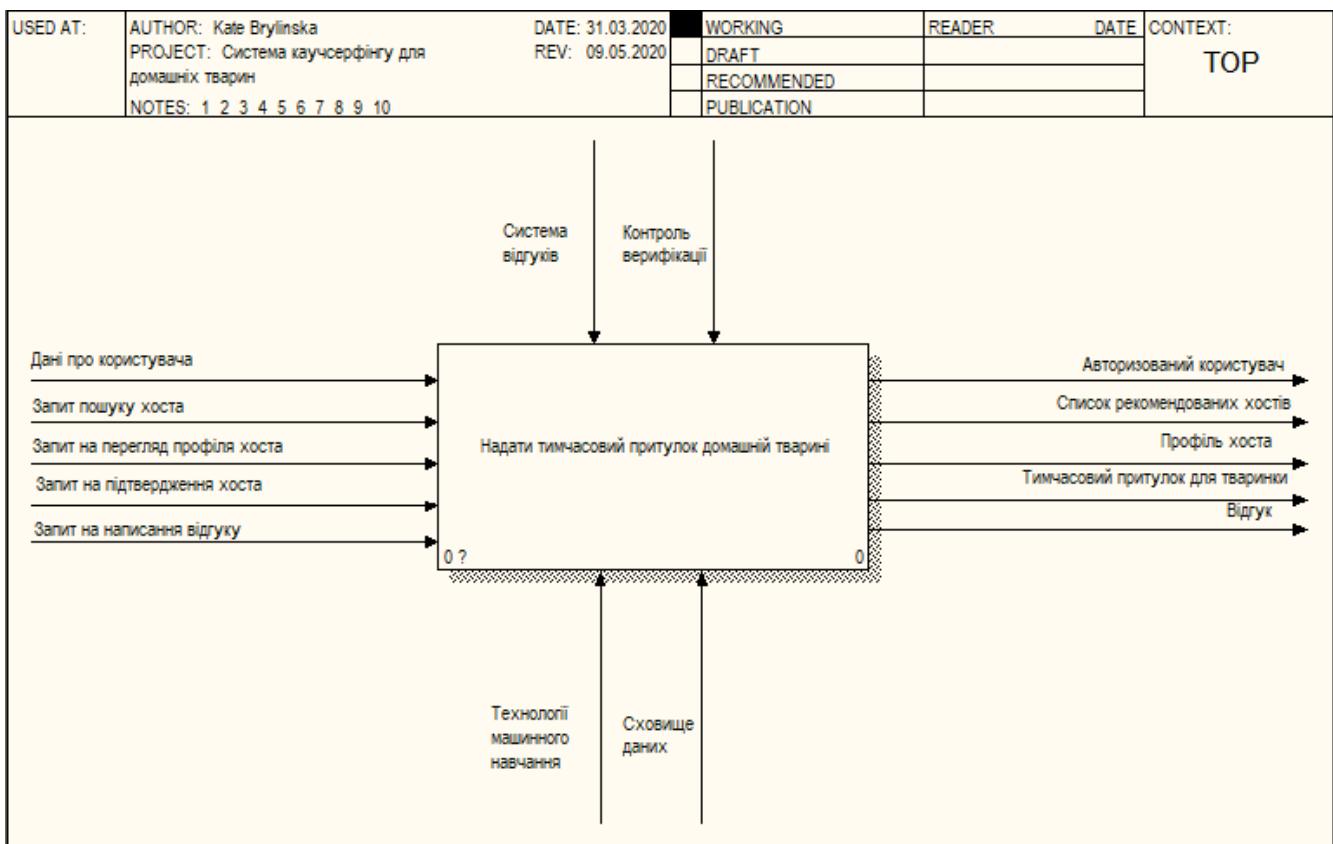
Після проведення аналізу ієрархій було отримано такі результахи:

1. пріоритет 0,15 – інформаційно-пошукова система,
2. пріоритет 0,21 – інформаційно-пошукова система,
3. пріоритет 0,29 – система прийняття рішень,
4. пріоритет 0,37 – інтелектуальна інформаційно-пошукова система.

З цього випливає, що найбільший пріоритет (0,37) має інтелектуальна інформаційно-пошукова система.

## **2.2. Конкретизація функціонування системи**

Обрано IDEF0 діаграму для деталізації функціонування системи каучсерфінгу для домашніх тварин на основі машинного навчання. Діаграму представлено на рис. 2.2.



*Rис. 2.2. IDEF0 діаграма*

Основним процесом, а також метою створення системи є *Надати тимчасовий притулок домашній тварині*. Зліва стрілками зображені вхідні дані, які необхідні для досягнення головної цілі системи. Такими вхідними даними є:

- дані про користувача;
- запит пошуку хоста;
- запит на перегляд профілю хоста;
- запит на підтвердження хоста;
- запит на написання відгуку.

Усі вони в заданій послідовності спрямовані на те, щоб надати тимчасовий притулок домашній тваринці.

На стрілках справа від головного процесу зображені, яку інформацію буде отримано на виході, при обробці вхідних даних. Отже, на виході ми отримуємо:

- авторизованого користувача;
- список рекомендованих хостів;

- профіль хоста;
- тимчасовий притулок для тварини
- та відгук.

Варто зазначити, що при цьому ключовими вихідними даними, які характеризуються успішною роботою системи, є саме тимчасовий притулок для тварини та відгук.

При цьому, на сам процес Надати тимчасовий притулок тварині впливають такі 2 фактори:

- система відгуків;
- контроль верифікації.

Система відгуків допомагає користувачу при виборі потенційного хоста для своєї тваринки та робить цей вибір по можливості безпечним. Відгуки, написані як і хостом, так і власником тваринки, не піддаються редагуванню та видаленню, що робить систему більш надійною.

Також присутній контроль верифікації. Кожен користувач при реєстрації повинен верифікувати свої дані за допомогою пошти. В майбутньому верифікація буде більш розширеною та вміщатиме в себе верифікацію документів, верифікацію адреми, а також грошову верифікацію, що підтверджуватиме спроможність хоста подбати про тваринку у разі чого, а також ці гроші будуть частиною прибутку від системи.

Знизу, під головним процесом зображені стрілки, які беруть участь у досягненні мети системи:

- сховище даних;
- технології машинного навчання.

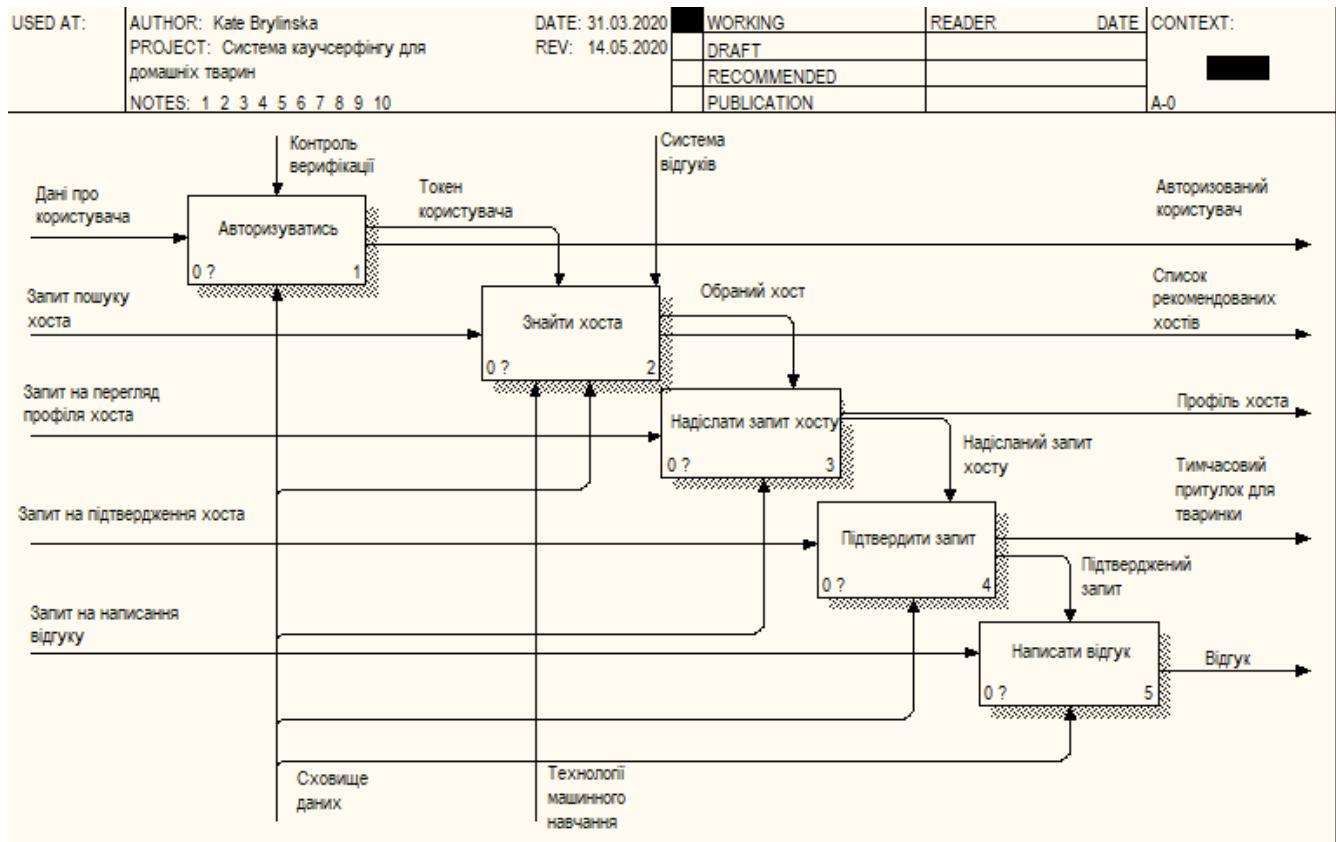
Всі дані, що є та які будуть створюватись зберігаються у сховищі даних. Там буде розміщена усі інформація про користувачів, зареєстрованих в системі, про їх тваринок, а також про запити та їхні стани.

Технології машинного навчання зможуть полегшити користування сайтом для того, хто шукатиме тимчасового хоста своїй тваринці. Завдяки їм, буде відображатись

спісок рекомендованих хостів, якій базуватиметься на даних про попередні створені запити хостам користувачем.

Варто зазначити, що найбільшою особливістю системи є те, що користувачі не будуть розподілені на категорії: хостів і тих, хто шукає хостів. Кожен користувач може як і надати тимчасовий притулок чужій тваринці, так і шукати його для своєї.

Наступним етапом є декомпозиція контекстної діаграми. Деталізована діаграма процесів зображена на рис. 2.3.



*Рис.2.3. Деталізована (декомпозиція I рівня) діаграма процесів*

Головний процес системи Надати тимчасовий притулок тварині складається із п'яти таких підпроцесів:

- Авторизуватись;
- Знайти хоста;
- Надіслати запит хосту;
- Підтвердити запит хоста;
- Написати відгук.

За авторизацією користувача в системі відповідає процес «Авторизуватись». На вході він отримує дані користувача, які потрібно буде опрацювати. Вихідними даними є токен користувача, що створюється при успішній авторизації та відповідає за сесію користувача, а отже і авторизований користувач. На даний процес впливає контроль верифікації, а уся інформація зберігається у сховищі даних.

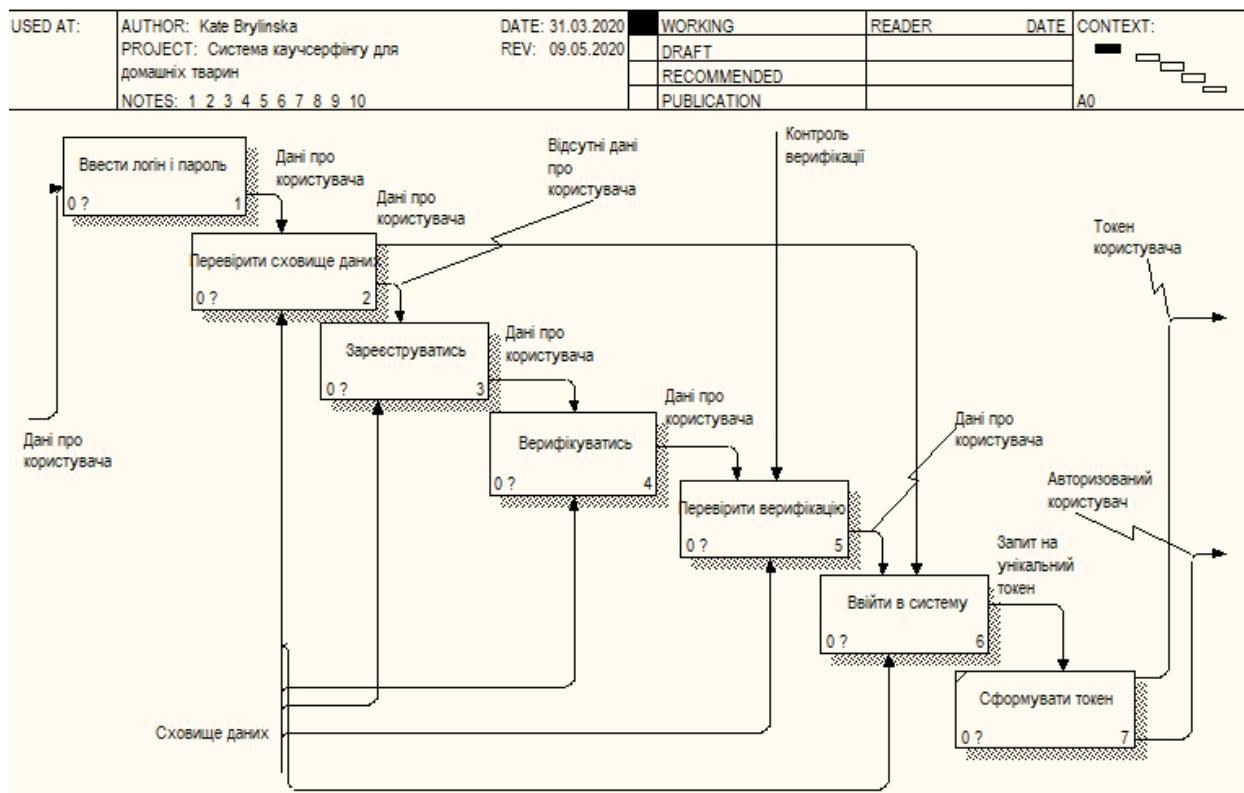
Наступний процес «Знайти користувача» відповідає за надання можливості пошуку бажаного хоста. У якості вхідних даних є токен користувача та запит на пошук хоста. Вихідними даними є список рекомендованих хостів, який згенерований за допомогою технології машинного навчання, а також системи відгуків, та обраний хост. Усі дані зберігаються та оброблюються завдяки сховищу даних.

Процес «Надіслати запит хосту» реалізує можливість перегляду профілю хоста та безпосереднього надсилання йому запиту. Вхідними даними при цьому є запит на перегляд профілю хоста та обраний хост. У якості вихідних даних є профіль хоста та надісланий запит хосту. Запит зберігається у сховищі даних.

Для підтвердження запиту опрацьованого хостом слугує процес «Підтвердити запит хоста». Уся інформація зберігається і опрацьовується у сховищі даних. Вхідними даними цього процесу є надісланий запит хосту та запит на підтвердження хоста, а вихідними – тимчасовий притулок для тваринки, а також підтверджений запит.

Останнім процесом, що забезпечує створення відгуків є процес «Написати відгук». У якості вхідних даних взято підтверджений запит та запит на написання відгуку. Вихідними даними є сам відгук. Усі відгуки та інформація про запит зберігається у сховищі даних.

Після цього деталізуємо усі процеси окремо. Деталізована діаграма для процесу «Авторизуватись» зображена на рис. 2.4.



*Рис.2.4. Деталізована діаграма для процесу «Авторизуватись»*

Користувач вводить свої дані – логін і пароль. Далі перевіряється сховище даних на наявність такого користувача. Якщо дані відсутні, то користувачу потрібно зареєструватись ввівши усі необхідні дані. Після цього користувач верифікується, вводячи свою електронну пошту. І наступним кроком після цього, або ж коли дані про користувача присутні в сховищі даних, буде вход в систему, при якому відправляється запит на формування унікального токена, що буде відповідати за сесію користувача. Даний процес закінчується тим, що на виході ми отримуємо авторизованого користувача та токен, що відповідає за його сесію.

На рис. 2.5 продемонстровано деталізацію другого підпроцесу – «Знайти хоста».

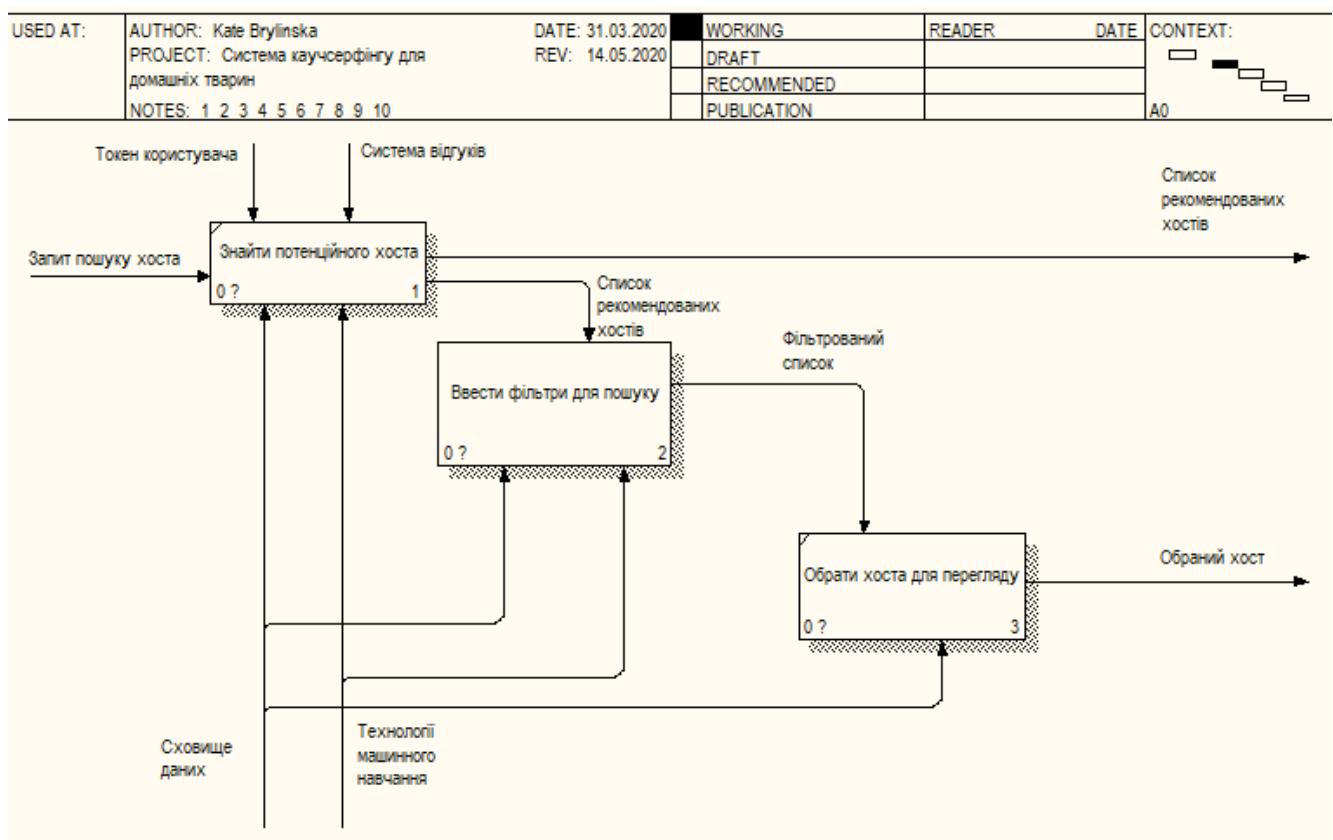


Рис.2.5. Деталізована діаграма для процесу «Знайти хоста»

Виконання даного процесу розпочинається з того, що користувач надсилає запит пошуку хоста, далі виконується сам пошук. Завдяки технологіям машинного навчання користувач отримує список рекомендованих хостів зі сховища даних. Також на формування цього списку впливає система відгуків. Далі користувач може ввести фільтри для пошуку: стать, наявність верифікації, тварин та відгуків. Після цього він отримує уже фільтрований список та має можливість обрати хоста для перегляду профіля. На виході даного процесу є обраний хост (дані про нього).

Деталізовану діаграму процесів процес «Надіслати запит хосту» зображено на рис. 2.6.

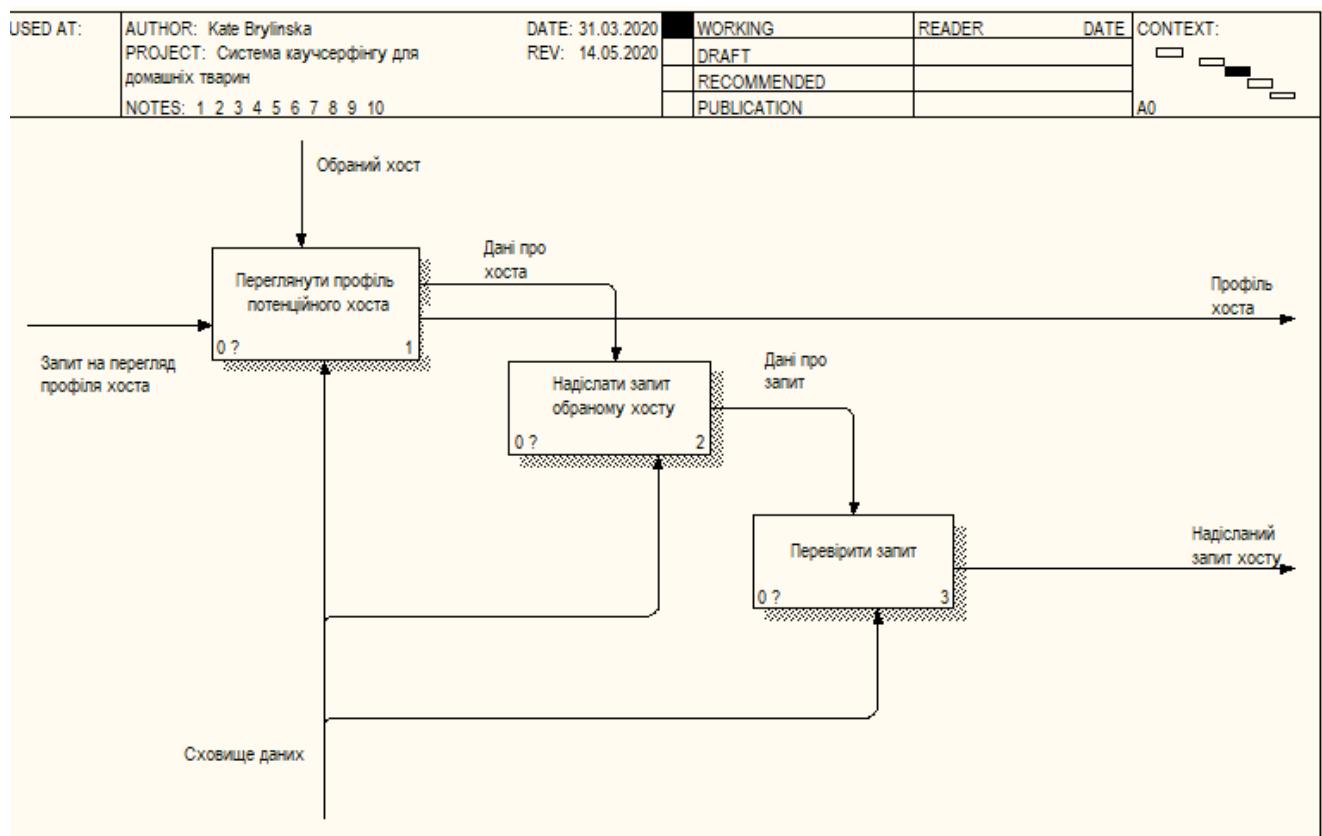


Рис.2.6. Деталізована діаграма для процесу «Надіслати запит хосту»

Запит на перегляд профіля хоста тригерує запуск процесу «Надіслати запит хосту». Передаються дані обраного хоста і користувачу відображається профіль бажаного потенційного хоста. На виході з цього процесу буде профіль хоста. Після того як користувач переглянув даний профіль він надсилає запит обраному хосту, де пише коротке повідомлення та обирає, яких саме тваринок хоче віддати на перетримку. Після цього запит перевіряється в сховизі даних. На виході ми отримуємо надісланий запит хосту, який передається на вхід наступному процесу.

На рис. 2.7 продемонстрована деталізована діаграма процесу «Підтвердити запит».

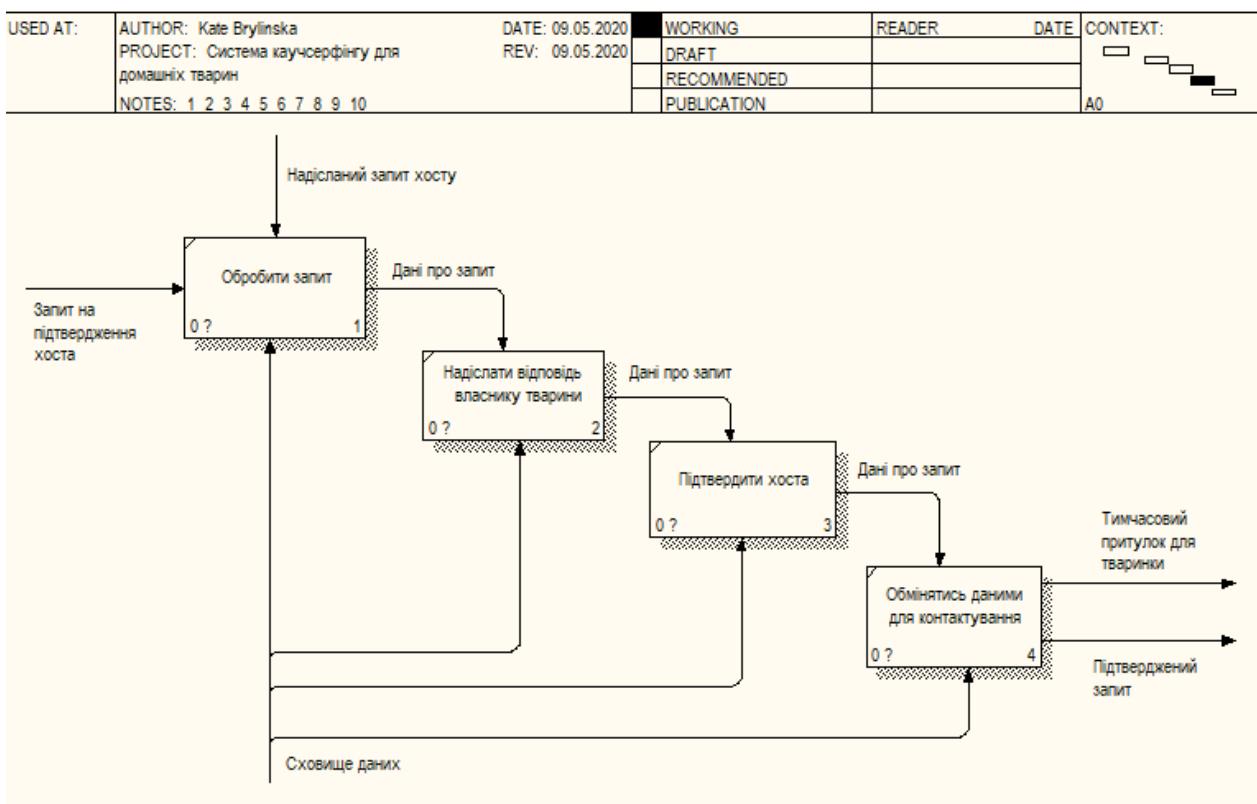


Рис.2.7. Деталізована діаграма для процесу «Підтвердити запит»

На вході даного процесу є запит на підтвердження хоста та уже надісланий запит хосту. Система оброблює запит і при наявності підтвежденого запита хостом надсилає відповідь власнику тваринки. Далі користувач також повинен підтвердити хоста, опираючись на його відповідь. Після цього вони обмінюються контактними даними, щоб домовитись про інші деталі та познайомитись краще. На вихіді даного процесу досягається головна ціль системи – знайдений тимчасовий притулок для тварини. А також на вихіді отримуємо підтверджений запит з двох сторін.

Наступним кроком є деталізація процесу «Написати відгук». Деталізована діаграма цього процесу зображена на рис. 2.8.

USED AT:	AUTHOR: Kate Brylinska PROJECT: Система каучсерфінгу для домашніх тварин	DATE: 31.03.2020 REV: 09.05.2020	WORKING	READER	DATE	CONTEXT:
			DRAFT			
			RECOMMENDED			
NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION			A0

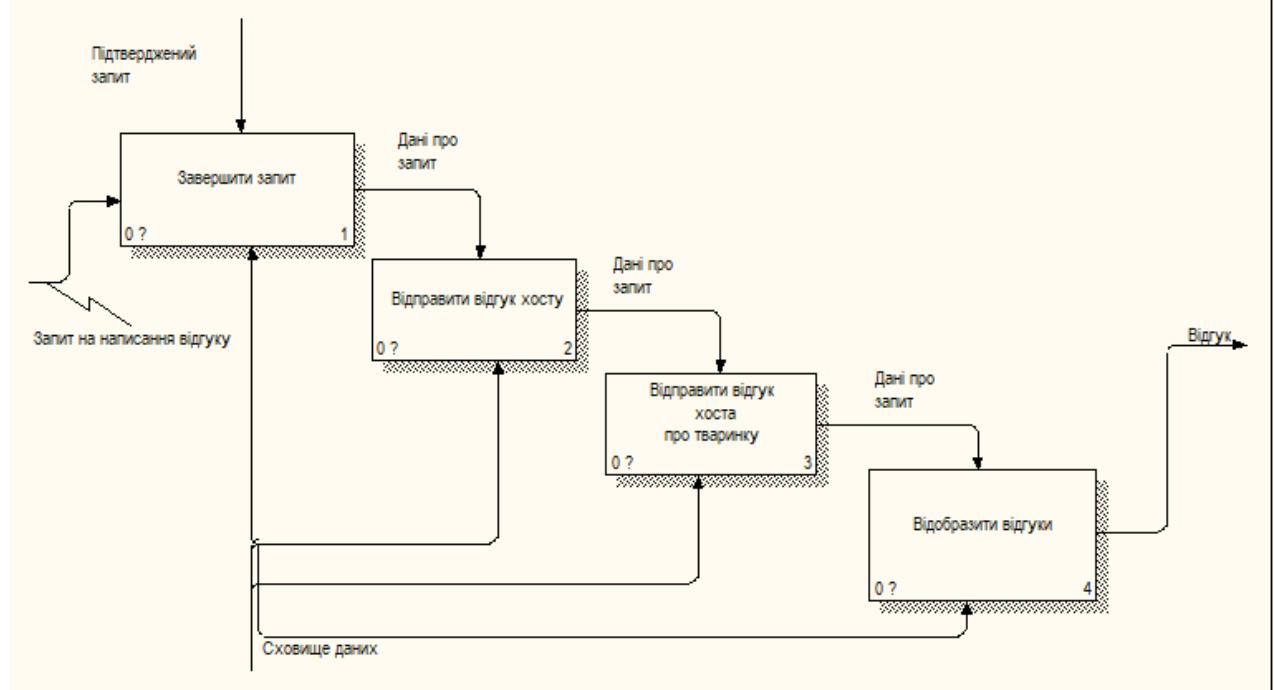


Рис.2.8. Деталізована діаграма для процесу «Написати відгук»

На вхід процесу «Написати відгук» подається запит про написання відгуку та підтверджений запит про надання тимчасового притулку. Для того, щоб виконати цей процес, спочатку, після того, як повернеться тваринка для власника, необхідно завершити запит. Після цього у обох сторін запиту з'явиться можливість написати відгук. Спершу, власник тваринки відправляє відгук хосту, після чого хост відправляє вігук про тваринку, якій він тимчасово надавав притулок. У цих відгуках вони також можуть вибрати чи рекомендують вони хоста або тварнику. Після цього система відображає відгуки. На виході заключного процесу отримуємо відгуки.

## **2.3. Побудова ієрархії задач**

Наступним важливим етапом при системному аналізі є побудова ієрархії задач. Ієрархія задач, що допомагає структурувати процеси та демонструє їх зображення на рис. 2.6.

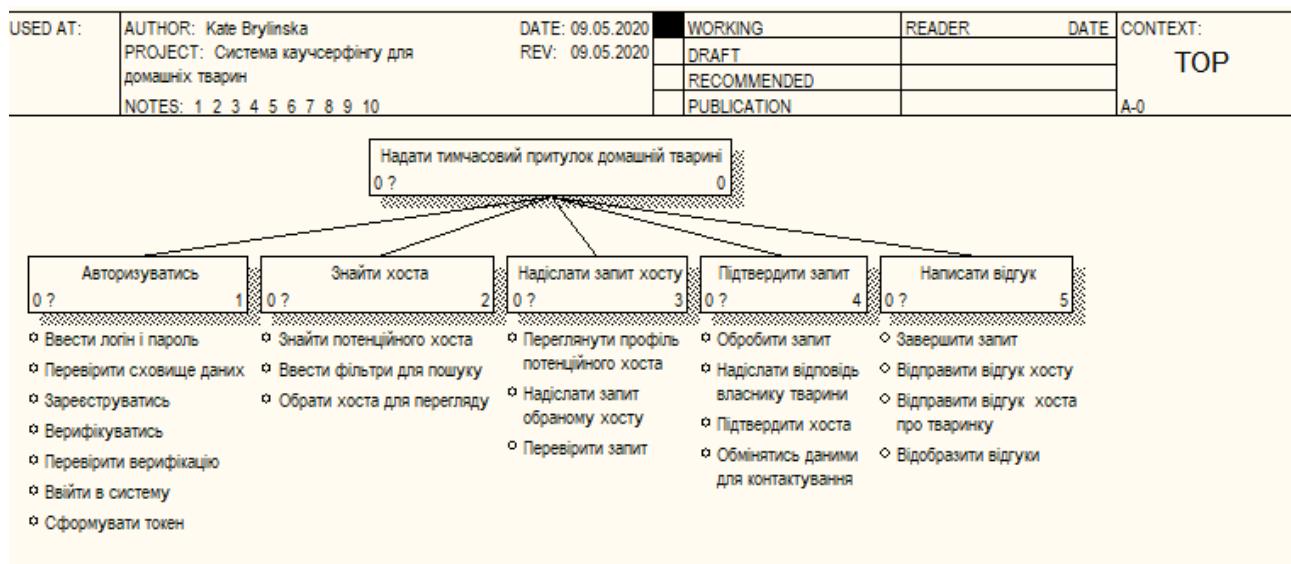


Рис.2.9. Ієрархія задач

«Надати тимчасовий притулок домашній тварині» – основна ціль (задача) системи. Дано задача розгалужується на такі п'ять підзадач: «Авторизуватись», «Знайти хоста», «Надіслати запит хосту», «Підтвердити запит» та «Написати відгук». Для того, щоб зрозуміти кожну, розглянемо їх детальніше.

Задача «Авторизуватись» складається з таких підзадач:

- ввести логін і пароль;
- перевірити сховище даних;
- зареєструватись;
- верифікуватись;
- перевірити верифікацію;
- ввійти в систему;
- сформувати токен.

Задача «Знайти хоста» містить такі підзадачі:

- знайти потенційного хоста;
- ввести фільтри для пошуку;
- обрати хоста для перегляду.

Задача «Надіслати запит хосту»:

- переглянути профіль потенційного хоста;

- надіслати запит обраному хосту;
- перевірити запит.

Задача «Підтвердити запит»:

- обробити запит;
- надіслати відповідь власнику тварини;
- підтвердити хоста;
- обмінятись даними для контактування.

Задача «Написати відгук»:

- завершити запит;
- відправити відгук хосту;
- відправити відгук хоста про тварину;
- відобразити відгуки.

## **Висновок до другого розділу**

Під час виконання другого розділу було здійснено проектування інформаційної web-системи каучсерфінгу для домашніх тварин на основі машинного навчання. Було побудовано дерево цілей системи та визначено основні її критерії. За допомогою методу ієрархій було обрано тип досліджуваної системи. Побудовано контекстну діаграму, яка описує процеси в системі. Проведено деталізацію усіх процесів. Також було сформовано ієрархію задач, яка описує основні методи та функції системи.

## РОЗДІЛ 3

### Програмні засоби розв'язання задачі

#### **3.1. Вибір та обґрунтування засобів розв'язання задачі**

При аналізі існуючих та схожих рішень було зроблено висновок, що систему каучсерфінгу для домашніх тварин варто розробляти у вигляді веб-сайту. Таким чином даний сервіс буде не тільки зручний для користування, але й гнучкий до розширення та пізнішої популяризації його по усьому світу. Щоб розпочати роботу, спершу, необхідно вибрати технології для розробки системи і в залежності від них обрати середовище розробки.

Для розробки front-end частини було вибрано такі технології:

1. HTML;
2. CSS;
3. JavaScript;
4. React;
5. Redux;
6. TypeScript;
7. React Styled Components;
8. React Hook Form;
9. Material-UI;
10. Axios;
11. YUP.

Технології, які були обрані для розробки back-end:

1. Node.js;
2. Express;
3. PostgreSQL;
4. Sequelize;
5. JWT;
6. Joi.

Розглянемо кожну з технологій окремо.

HTML — це мова гіпертекстової розмітки, що описує структуру документу та форматує звичайний текст в заголовки, абзаци, списки та інші структури, а також здатна створювати посилання й на інші сторінки[35]. Вона являє собою інструкції з форматування, які називаються тегами. Теги вбудовані в розділи документа та містять конкретну інформацію. Окрім цього, саме теги повідомляють браузерам, яким чином форматувати та відображати інформацію на екрані. В загальному теги HTML-документів є зрозумілими і простими для використання, адже вони створені за допомогою англійських загальновживаних слів, зрозумілих скорочень та позначень.

У 1989 році Тім Бернерс-Лі запропонував новий компонент технології для розробки гіпертекстової розподіленої системи WWW (World Wide Web) – мову гіпертекстової розмітки HTML. Її ідея полягала в тому, що користувач зможе переглядати інформацію у зручному для нього форматі, наприклад, як при читанні книги. Було виявлено, що цього можна досягти за допомогою гіпертекстових посилань.

Другим, та не менш важливим, фактором, що спричинив неабиякий вплив на розвиток мови HTML, було те, що її обрали як основу. Це означало, що звичайний текстовий файл з використанням HTML можна було створювати в будь-якому текстовому редакторі та у будь-якому середовищі, що зробило її доступною для усіх розробників.

Для того, щоб стилізувати сторінки, які були створені за допомогою HTML, Консоріум Всесвітньої мережі створив специфікації CSS (Cascading Style Sheets)[36]. Варто зазначити, що даний Консоріум розвиває їх і до сьогодні. CSS є одною з основних технологій інтернету поруч із HTML та JavaScript.

Не тільки автори, але й інколи відвідувачі веб-сайтів використовують CSS для того, щоб змінити або обрати шрифти, кольори, верстку та інші критерії того, як виглядає сайт. Однією з головних причин для застосування та перевагою қаскадних стилів є можливість розділити контент, що покращить сприйняття та зрозумільність контенту, в залежності від вигляду документу, в якому будуть описані стилі. Іншими словами, HTML чи XML документ, який має однакове наповнення можна

відображати по-різному, використовуючи CSS. Для відображення сторінки можуть бути:

- Стилі прописані автором (зовнішні або внутрішні таблиці стилів, стилі для окремого елементу (тегу, класу, id);
- Стилі користувача (файл з розширенням .css, що розташований локально і вказаний в налаштуваннях браузера користувачем для використання на сторінках);
- Стилі браузера (дефолтний стиль переглядача, який використовуються у разі неповної чи відсутньої інформації про стиль певного елемента).

JavaScript (JS) — це динамічна мова програмування, яку класифікують як прототипну скриптову. ECMAScript — це стандарт, що відповідає за реалізацію JS. Зазвичай, JS використовують для того, щоб надати можливість користувачу взаємодіяти з сайтом[37]. Також JS реалізує асинхронний обмін даними з сервером і надає можливість змінювати зовнішній вигляд та структуру сторінок веб-сайту.

Отже, JavaScript застосовують для:

- написання скриптів, що роблять з статичної веб-сторінки – динамічну;
- створення Single-Page Application;
- програмування серверної частини;
- мобільних додатків;
- стаціонарних застосунків;
- сценаріїв в прикладному програмному забезпеченні тощо.

React (попередні назви: React.js та ReactJS) — це відкрита JavaScript бібліотека JavaScript, що є у відкритому доступі. Її використовують при створенні інтерфейсу користувача для того, щоб пришвидшити загрузку веб-сторінок[38].

Саме за допомогою React розробники здатні створювати великі програмні застосунки з змінними даними і при цьому не призводити до постійного перезавантаження веб-сторінки. У цієї бібліотеки є три мети: швидкість, простота і

масштабованість. На даний момент React-ом користуються такі великі корпорації, як Netflix, Airbnb, Facebook, Yahoo та багато інших.

Існує два варіанти написання компонентів React – JSX та `.createElement()`. Зазвичай, використовують JSX через його вигляд, що подібний до HTML, а отже легкий для прочитання. Оскільки, браузери не здатні розпізнати код написаний JSX, то його компілює React.

Найчастіше разом з React використовують іншу відкриту бібліотеку - Redux для створення інтерфейсу користувача. Це відкрита JavaScript бібліотека, метою якої є управління станом програми.

У 2015 році під впливом функціональної мови програмування Elm та розробленого Facebook Flux, Ден Абрамов і Ендрю створили Redux, який до сьогодні допомагає розробникам з усього світу оптимізувати код програми. Redux можна використовувати з різними бібліотеками, проте стала вона популярна саме завдяки React. Зазвичай, розмір файлу Redux не перевищує 2kB.

Redux умовно складається з екшенів і редьюсерів. Редьюсери – це функції, що здатні змінювати стан та ділити його на менші, модульні та керовані, частини. Екшени – це функції, що в залежності від свого виду, можуть або запускати редьюсери та змінювати стани, або запускати асинхронні дії.

TypeScript — мова програмування, розроблена Андерсом Гейлсбергом у 2012 році, що попередньо створив такі мови, як C#, Delphi та Turbo. TypeScript містить у собі розширені можливості функціоналу JavaScript.

Під лінцезією Apache розповсюджується код компілятора, який трансляє Typescript через JavaScript. За допомогою сервісу CodePlex розробка TypeScript проводиться в публічному репозиторії. Окрім того, у відкритому доступі знаходяться специфікації мови. Усе це означає, що до розвитку мови може долучитись будь-який розробник, що має відповідні знання та бажання.

Однією з особливостей TypeScript є його залежність із мовою JavaScript. Після того, як програма скомпільована на JavaScript тоді будь-який сучасний браузер може

виконувати TypeScript. Також його можна використовувати з серверною частиною Node.js.

Вже існуючі застосунки з часом поступово будуть перероблюватись та адаптуватись від TypeScript. Увесь код JavaScript повністю сумісний з TypeScript – це основний принцип даної мови. Це означає, що в TypeScript програмах можна застосовувати раніше створені напрацювана, а також стандартні JS-бібліотеки. Okрім того, не варто переписувати увесь JavaScript код, можна просто додати анотації, що стосуються типізації та підключити їх, як файли.

На даний момент, є три способи стилізувати сторінки, використовуючи React:

1. Стрічкова стилізація
2. Styled Components
3. CSS модулі.

Перший метод - це стрічкова або, як прийнято її називати, інлайнова стилізація. Ними досить легко користуватись, якщо володіти мовою HTML та CSS, в яких стилі можна добавляти інлайново. За схожим принципом, можна робити і в React. Суть інлайнової стилізації полягає в тому, щоб добавляти інлайнові стилі окремим компонентам, які розробник бажає відрендерити. Дані стилі записуються як атрибути і передаються елементу.

Другий метод - Styled Components, за допомогою якого можна писати звичний CSS у JavaScript файлі. Цей метод надає можливість використовувати увесь можливий функціонал CSS, навіть включаючи медіа-запити, вкладення тощо.

Також Styled Components - це шаблонні рядки, тобто при визначені стилів, насправді, створюється простий React компонент, до якого вже прикріплена ці стилі. Такі стилізовані компоненти можна перевикористовувати багато разів.

Третій метод - використання CSS модулей. CSS модуль - це CSS файл, який скомпільований. Проте в CSS модулях всі анимації, а також всі імені класів по замовчуванню мають локальну область видимості.

Для роботи було обрано другий метод - React Styled Components, адже він зручний у написанні коду, а також його функції зменшують кількість можливих конфліктів у коді та сам його об'єм.

React Hook Form - це бібліотека для роботи з формами, що спрощує процес валідації, а також робить їх гнучкими, розширюваними та продуктивними. Її можна використовувати з React та React Native.

Axios – це HTTP-клієнт, що застосовують для зв'язку між інтерфейсом та серверною частиною сайту. Він надає можливість передавати та отримувати дані з серверу, а також відслідковувати дані процеси. Більше того, даний клієнт автоматично трансформує JSON дані та не є об'ємний при написанні, що робить код більш читабельним[39].

В 2014 році Google вперше представив мову дизайну, яка називається Material Design. Це свого роду віртуальна мова, що використовує сіткові макети, переходи, анімації та певні ефекти, що стосуються глибини кольору, а саме - тіні і освітлення. Уніфікація, створення і налаштування - це три основні цілі Material Design.

Material-UI — це частина даної мови дизайну, яка представлена у вигляді набору React компонентів, що здатні працювати ізольовано. Це означає, що вони можуть самі себе підтримувати і залучати лише ті стилі, які вони повинні відображати.

Для передачі та перевірки значень було обрано YUP (конструктор схем). Схеми YUP надають можливість будувати складні та взаємозалежні перевірки, а також можливість перетворювати значення. YUP використовується для валідації та перевіряє, чи ввідні дані коректні[40]. API YUP було створено на мотивах JOI, але з більшою швидкодією.

Node.js — це платформа, що використовується для створення серверної частини великих мережевих додатків, які написані за допомогою JavaScript. Засновником даної платформи вважається Раян Дал. Завдяки, Node.js на сервері з'явилася можливість виконувати JavaScript-скрипти та відправляти результат їх виконання користувачеві[41]. Проте сам цикл, що відповідає за обробку подій не є видимий для

розробника. Також Node.js має відкритий код, а отже і велику спільноту розробників, що постійно її розвивають.

Express або Express.js - це найпопулярніший веб-фреймворк для node.js, який також є у відкритому доступі, а тому підтримується і розробляється багатьма розробниками по всьому світу. У ньому є дуже широкий набір різних функцій, що допомагають при створенні мобільних та веб-додатків. Сам по собі Express є досить мінімалістичним, та що не менш важливо - гнучким. Завдяки Express можна легко і швидко створити надійний та безпечний API, адже він має в своєму розпорядженні багато службових методів HTTP[42].

PostgreSQL - це одна з найпотужніших об'єктно-реляційних баз даних, що розвивається вже більше 30 років в рамках POSTGRES. Ця база даних є у відкритому доступі і постійно підтримується великою спільнотою розробників, які періодично пропонують інноваційні рішення щодо її розвитку. PostgreSQL забезпечує цілісність даних, широкий набір функцій, надійність та перевірену архітектуру. Варто також додати, що ця база даних працює на всіх операційних системах, а також має власні потужні застосунки. Одним із них є PostGIS - відомий розширювач геопросторових баз даних.

JWT (JSON Web Token) – це відкритий, а також галузевий стандарт, за допомогою якого формуються токени. Токен створюється на серверній частині при кожному вході та відповідає за сесію користувача. Також JSON токени мають свій визначений час «існування», після якого, зазвичай, в дію приходить Refresh токен, який його оновлює[43].

Sequelize - це бібліотека в Javascript, що робить процес управління базою даних SQL набагато легшим. По своїй суті, Sequelize - це свого роду об'єктно-реляційний мапер, а отже він має змогу відображати синтаксис об'єкта на різні схеми БД. Для відображення Sequelize потрібно використовувати Node.JS, а також синтаксис об'єктів Javascript.

Зважаючи, на технології, які були обратні для реалізації системи каучсерфінг для домашніх тварин, у ролі середовища розробки доцільно буде використовувати Visual

Studio Code [44]. У даного середовища дуже зручний та зрозумілий інтерфейс, що пришвидшує розробку, адже у ньому реалізовано автодоповнення коду різного типу (функції, модулі тощо), а помилки в коді зразу ж виділяються, також код більшості програмних мов підсвічується та легко форматується. Чітко відображенна структуризація файлів та легкий перехід між ними. Деякі частини інтерфейсу продемонстровані на рис. 3.1-3.2.

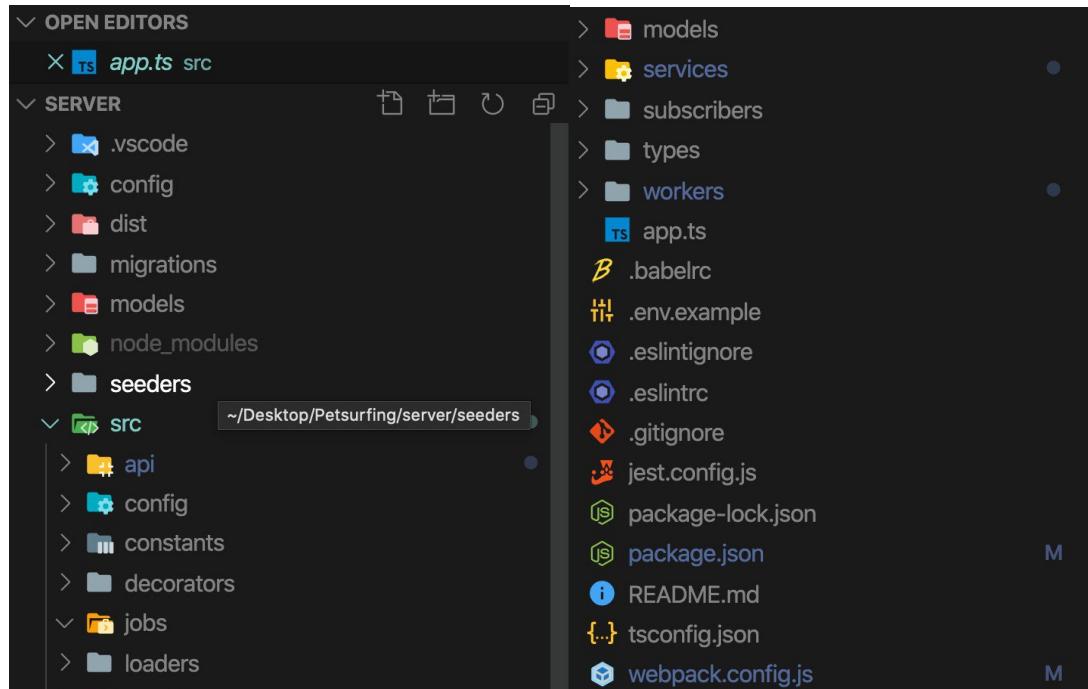


Рис. 3.1. Інтерфейс VS Code. Структура проекту.

The screenshot shows the code editor with the following content:

```

app.ts
src > app.ts > ...
1 import "reflect-metadata";
2
3 import config from "./config";
4 import express from "express";
5 import io from "socket.io";
6 import Logger from "./loaders/logger";
7 import "./types/global";
8
9 async function startServer() {
10   const app = express();
11   await require("./loaders").default({ expressApp: app });
12   const server = app.listen(config.port, () =>
13     Logger.info(` Server listening on port: ${config.port} `)
14   );
15   io.listen(server);
16 }
17
18 startServer();
19

```

Рис. 3.2. Інтерфейс VS Code. Підсвічування коду.

Досить легко підключати потрібні бібліотеки та пакети за допомогою даного застосунка. Це значно спрощує та пришвидшує роботу. Можна стилізувати майже

будь-що, що захочеться та встановлювати будь-яке число другорядних розширень, що потрібні при розробці. VS code є одним з найпопулярніших застосунків серед девелоперів.

### **3.2. Технічні характеристики обраних програмних засобів розроблення**

Усі обрані технології є одними з найпотужніших та найпопулярніших програмних засобів для розробки веб-сайту, а також взаємодоповнюють один одного.

Одним з найважливіших інструментів про розробці є правильно обрана основна front-end бібліотека. Розглянемо детальніше, чому React вважається одним з найзручніших та найкращих інструментів для створення сайту.

Розпочнемо з того, що дана технологія розробляється однією з найбільших корпорацій Facebook, що забезпечує її стабільний швидкий інноваційний розвиток. Серед ключових переваг React є його потужність, адже він здатен витримати велику кількість навантаження. Оскільки, React містить в собі концепцію багаторазових компонентів, це означає, що він дуже гнучкий при розробці, адже деякі компоненти можна використовувати повторно. Також завдяки компонентній структурі стає легше стилізувати сторінку. Перевагою над іншими бібліотеками є те, що він підтримує віртуальний DOM, замість того, щоб покладатись лише на браузер при роботі, що робить розробку системи більш надійною. Окрім того, сама бібліотека є відкритому доступі, а тому постійно розвивається та підтримується спільнотою розробників з усього світу. Оскільки, React не є новою бібліотекою і на даний момент існує уже багато його версій, варто також додати, що у ньому закладена легка міграція між цими версіями.

Серед недоліків як інструменту для використання, варто зазначити, що дана технологія є досить важкою для вивчення, адже потребує знань багатьох суміжних засобів.

За допомогою React можна вбудовувати різні методи життєвого циклу, які дозволяють опрацьовувати дані у будь-яких точках цього циклу:

- `componentWillMount` – це метод життєвого циклу, що перед монтуванням самих даних говорить JavaScript, як налаштовувати ці дані;
- `shouldComponentUpdate`. Даний метод життєвого циклу робить так, щоб Javascript з використанням логічних змін оновив компонент;
- `render` - це найбільш важливий метод життєвого циклу і є необхідним у кожному компоненті. Саме він з'єднується з JSX і здатний його відобразити;
- `componentDidMount` є дуже схожим до компонента `WillMount`, проте він може працювати лише після методу `render`. Його застосовують для того, щоб визначати стани і властивості, додавати JSON-дані.

Усі інші технології для розробки інтерфейсу обирались у відповідності до обраної бібліотеки React, що доповнюють та розширяють її функції. Саме таким чином, було обрано:

- основну динамічну мову JavaScript;
- строго типізований TypeScript, що оптимізує код JS та розширює його функціонал;
- Redux, який вирішує проблему з передачею даних між компонентами та створює глобальне сховище;
- найзручніший метод стилізації компонентів React Styled Components;
- React Hook Form, що спрощує валідацію форм;
- Axios для HTTP-запитів між сервером і клієнтом;
- Yup для валідації інпутів.

Для реалізації серверної частини було обрано технологію, що використовується для сайтів, написаних на JavaScript – Node.js. Написання і front-end частини, і back-end частини на одній ж і тій самій мові програмування сприяє швидкості розробки та кращому розумінню коду. Так само, як і в React, код даної платформи є відкритий, а тому підтримується багатьма розробниками та компаніями. Node.js добре працює в тандемі з найпопулярнішими JS бібліотеками, в тому числі й з React.

Переваги Node.js:

- легка масштабованість. Девелопери можуть легко масштабувати веб-сайт як вертикально, так і горизонтально;
- легка для вивчення. Оскільки, JavaScript є однією з найпопулярніших мов серед розробників, то зрозуміти дану платформу буде достаньо просто для них;
- використовується в якості єдиною мовою програмуванняж
- велика спільнота, що підтримує і розробляє платформу;
- забезпечує кешування окремих модулей;
- всі запити виконуються асинхронно, в одному потоці, не блокуючи системи вводу та виводу. Переважно, це пришвидшує роботу веб-застосунків, адже у всіх запитів є свій пріоритет, і поки відправляється один запит, він може опрацьовувати інші задачі;
- постійно розширюється. Це означає, що можна додатково розширити чи налаштувати Node.js відповідно до вимог.
- вбудовані API для розробки серверів TCP, HTTP та DNS тощо.

Серед недоліків варто виділити такі:

- нестабільний API;
- відсутність сильної підтримки бібліотечної системи JavaScript порівняно з іншими мовами програмування. Це сповільнює реалізацію навіть достатньо простих завдань;
- якщо застосунок буде ставати більш масштабованим, то код, що відповідає за асинхронні виклики, як правило, стає не читабельним та громістким.

Не зважаючи на деякі недоліки, дана платформа ідеально підходить для розробки серверної частини системи каучсерфінгу для домашніх тварин на основі машинного навчання.

Найпопулярніший фреймворк, що є на даний момент для Node.js – Express. Для застосування потрібно імпортувати Express.js бібліотеку у сам файл ноди. Завдяки цьому з'являється можливість використовувати багато готових функцій, які потрібні

для написання маршрутизації веб-застосунку. Без Express прийдеться писати багато складного коду, щоб розробити API. Також багато інших популярних фреймворків базуються саме на Express.

PostgreSQL – це потужна база даних, яка містить в собі багато функцій, що допомагають розробникам при створенні високопродуктивних застосунків, а також адміністраторам, щоб цілісність даних була захищена. PostgreSQL спрощує роботу керування як і малими, так і великими наборами даних. Okрім того, дана система надає можливість створювати свої функції, визначати власні типи даних та писати код на інших мовах програмування і при цьому не буде необхідною перекомпіляція бази даних.

PostgreSQL є дуже обширною. Вона містить багато різних типів даних, зберігає цілісність даних, є надійною та безпечною, підтримує міжнародні символи та вмішає пошук цілого тексту. Більше функцій та особливостей даної платформи можна прочитати у її документації.

На даний момент, є активні кластери у спільноті розробників, що підтримують і розробляють PostgreSQL.

У ролі токенів використовується JSON Web Token, завдяки якому можливий автономний, компактний та безпечний спосіб передачі даних між клієнтом і сервером та є об'єктом JSON. Такі інформацію можна підписати цифровим шляхом за допомогою відкритих ключів (RSA/ECDSA) або секретного, використовуючи про цьому алгоритм HMAC.

У порівнянні з іншими токенами такими, як SWT (Simple Web Tokens) та SAML (Security Assertion Markup Language Tokens), JWT має декілька переваг.

По-перше, код JSON містить менше слів, ніж XML. Так само, коли він зашифрований його розмір теж є набагато меншим, тому JWT вважається більш компактним за SAML.

Другою перевагою є порівняна спрощеність цифрового підписання JSON. Натомість підписання XML з використанням XML Digital Sign зазвичай призводить до появи багатьох незрозумілих отворів у безпеці. SWT може бути підписаний лише

симетрично та секретно за допомогою HMAC алгоритму, проте JWT так само, як і SAML мають у своєму розпорядженні декілька відкритих ключів (сертифікатів).

### **3.3. Машинне навчання та його методи**

Однієї з технологій, що покращить роботу інформаційної web-системи каучсерфінгу для домашніх тварин було обрано машинне навчання.

Розглянемо детальніше цю технологію та засоби, що дозволять її реалізувати.

Машинне навчання (Machine Learning або ML) – це одна з підгалузей Штучного інтелекту, за допомогою якого система може навчатись на даних або ж отриманому досвіді, використовуючи різні алгоритми[45].

У 1952 році Артур Самуель створив програму Samuel Checkers-playing, і вона стала однією з перших у світі, в основі якої були алгоритми, що самостійно навчаються. Її призначення полягало у грі в шашки, однак вже цього було достатньо для значного прогресу Штучного інтелекту.

Найперша модель нейронної мережі, яка успішно реалізувала подібні на сьогоднішні алгоритми ML, була запущена через 5 років – у 1957-му.

Зараз розробляються найрізноманітніші системи машинного навчання, які планують застосовуватись для майбутніх технологій: безпілотний транспорт, робот-помічник, цифрове розумне місто, Інтернет Речей тощо.

Існують беззаперечні факти, що вказують на перспективи ML. Ось деякі з них:

- Компанія Google заявляє, що незабаром її продукти з наслідків звичного для всіх програмування зміняться результатом машинного навчання;
- Світові гіганти Google, Amazon Facebook, Microsoft та Apple давно ведуть активні пошуки здібних спеціалістів у царині штучного інтелекту;
- Не таємниця, що сам Марк Цукерберг бере участь у переманюванні найспроможніших, на його думку, випускників у свою компанію;
- Академічні конференції з машинного навчання відвідують тепер у чотири рази більше, ніж у попередні роки;

- Нові продукти, які зараз користуються шаленою популярністю серед людей усього світу, були створені за допомогою ML.

Як правило, ML поділяють на три групи: навчання з наглядом (контрольоване), без нагляду (неконтрольоване) та з підкріпленням[46].

### **Контрольоване навчання**

Цей тип навчає машин на прикладі. Під час такого навчання алгоритми систем піддаються значній кількості міченых даних: зображень, рукописних фігур із зазначеними цифрами, якому номеру вони відповідають. Опрацювавши достатню кількість прикладів, система вчиться розпізнавати скучення пікселів та форм і врешті може розрізнати рукописні числа, такі як 9 і 4 чи 8 і 6. Те саме з розпізнаванням тварин: спершу в систему закладають алгоритм, який за допомогою певних маркерів дозволяє читувати ознаки конкретного виду, а потім система сама оброблятиме результати й зможе застосовувати алгоритм для схожих завдань.

Однак для підготовки цих систем зазвичай потрібна величезна кількість міченых даних, а деяким системам необхідно освоїти мільйони прикладів, щоб впоратись із завданням.

### **Неконтрольоване навчання**

У цьому випадку алгоритми без попереднього нагляду та навчальних завдань із виявленням шаблонів у даних, намагаються виявити схожість, яка розділяє дані на категорії. Алгоритм не розроблений для виділення конкретних типів даних, він просто шукає ті дані, які можуть бути згруповани за схожістю, або для аномалій, які виділяються.

Прикладом може бути сервіс Airbnb, що об'єднує будинки, доступні для оренди по сусіству, новини Google, які щодня групують історії з подібних тем, стрічка YouTube з пропозиціями переглянути відео на схожі теми, що й раніше тощо.

### **Навчання з підкріпленням**

Цей вид застосовується для складніших задач. Суть у тому, що система має змогу діяти в певному середовищі та збирати дані про сигнали, які середовище надає у відповідь на її дії.

Прикладом може слугувати програма Google DeepMind's Deep Q-network: не маючи доступу до коду ігор, вона змогла перемогти досвідчених гравців. Річ в тому, що система подавала пікселі відожної гри та визначала різну інформацію про перебіг подій, наприклад – відстань між об'єктами на екрані. Потім алгоритми співвідносять стан гри, дії, які система виконує в грі, з оцінкою, якої врешті досягає. Таким чином протягом багатьох циклів гри система нарешті будує модель дій, яка буде максимально оцінена результатом.

Таке навчання часто використовується при розробці навігацій для роботів, які завдяки аналізу власного досвіду, уникають можливих загроз.

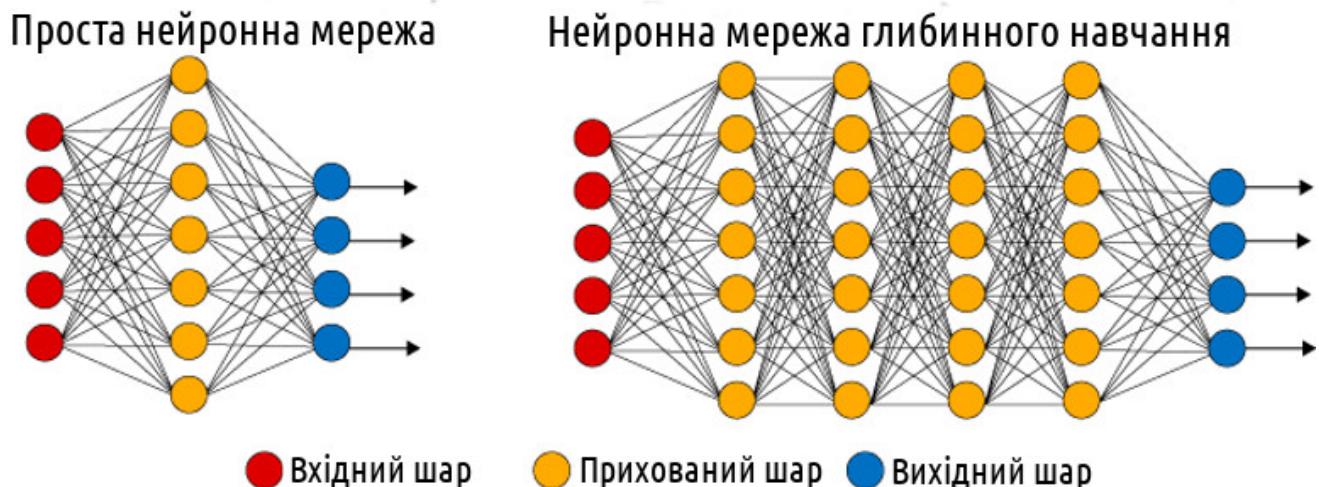
### **Нейронні мережі та глибоке навчання**

Нейронні мережі – така парадигма програмування, яка дає змогу комп'ютеру вчитися з даних спостережень.

Нейронні мережі та глибоке навчання зараз пропонують найкращі рішення багатьох проблем у розпізнаванні зображень, мови та природній обробці мови.

Однак ще до 2006 року вчені не знали, як навчити нейронні мережі перевершувати більш традиційні підходи, за винятком кількох спеціалізованих проблем. Саме у 2006 році було відкриття методик навчання у так званих глибоких нейронних мережах. Ці методи зараз відомі як глибоке (глибинне) навчання. Вони були розвинені далі, і сьогодні широко розгортаються такими компаніями, як Google, Microsoft та Facebook.

Методи глибокого навчання засновані на стохастичному градієнтному спуску та зворотному поширенні, але також вводять нові ідеї. Вони дали можливість набагато глибшим і більшим мережам пройти навчання – люди зараз регулярно тренують мережі з 5 до 10 прихованими шарами. І, виявляється, що вони працюють набагато краще у багатьох проблемах, ніж дрібні нейронні мережі, тобто мережі лише з одним прихованим шаром. Причиною цього є здатність глибоких мереж будувати складну ієрархію понять. Це дещо схоже на те, як звичайні мови програмування використовують модульну конструкцію та ідеї абстракції для створення складних комп'ютерних програм.



*Рис.3.3.Проста нейронна мережа та нейрона мережа для глибинного навчання*

Глибинне навчання – це одна з найгарячіших технологій. Великі корпорації та молоді стартапи вкладають великі бюджети у цю сферу. Видання The Economist пише, що дані – це нова нафта у 21 столітті. Якщо дані – це сира нафта, бази даних та сховища даних – це бурові установки, які викопують та перекачують дані в Інтернеті, тоді слід подумати про глибоке навчання як про переробку нафти, яка нарешті перетворює її в усі корисні кінцеві продукти.

Глибоке навчання важливе, адже без правильних інструментів для переробки цієї так званої нафти, нічого не вийде.

Завдяки попиту з боку геймерів у всьому світі, графічні процесори (одиниці графічної обробки) дозволяють використовувати алгоритми глибокого навчання для створення та навчання моделей із дивовижними результатами в максимально короткі терміни.

### **Машинне навчання для бізнесу**

Починаючи із 2012 року кількість венчурних інвестицій у бізнеси та стартапи, які використовують нові технології, зростає постійно. На сьогодні споживачі очікують персоналізації та «розумних» функцій[47]. Тож є сенс вбудовувати їх, навчатись у них і переконуватися, що механізм зворотного зв'язку працює.

Застосовувати машинне навчання ефективно й доречно у таких випадках, якщо:

- немає програм для обробки даних, кількість яких зростає в геометричній прогресії постійно;

- реальні дані мають погрішності: спотворення, відсутність систематичності або належної повноти;
- неможливо з'ясувати закономірності між даними через їхню суттєву відмінність між собою;
- неможливо прийняти рішення чи спрогнозувати результат без допомоги ML.

Саме тому для системи було обрано використувати нейронну мережу глибинного навчання.

### **3.4. Вибір засобу реалізації машинного навчання**

Наступним кроком є аналіз технологій, завдяки яким можна реалізувати машинне навчання, використовуючи мову JavaScript.

У 2019 році завдяки глибинному навчанню та ядру побудованому на лінійній алгебрі *Tensorflow.js* став невід'ємною частиною для багатьох проектів JavaScript, що використовують машинне навчання. Не зважаючи на те, що досить довгий час найпопулярнішою мовою для реалізації машинного навчання був Python, *Tensorflow.js* досить швидко здобув таку ж кількість підтримуваних API. Також використовуючи його можна вирішити майже буде-яку поставлену задачу.

*Tensorflow.js* не потребує додаткових компіляторів, а тоже може застосовуватись безпосередньо в браузерах, при цьому варто підключати WebGL для пришвидшення його роботи. Переваги та недоліки описано в таблиці 3.1.

*Таблиця 3.1.*

**Таблиця переваг та недоліків *Tensorflow.js***

<b>Переваги</b>	<b>Недоліки</b>
Відмінні обчислювальні графічні візуалізації.	Немає десктопної підтримки для Windows.
Безперебійна продуктивність, швидко оновлюється та часто випускають нові версії з розширеними можливостями.	Відсутні символльні петлі.

Можливість задеплоїти на різних апаратних машинах (від мобільних пристройів до складних комп'ютерних систем).	Немає підтримки GPU, окрім Nvidia та лише на одній мові.
Дана технологія є високопаралельною і спроектована для використання на різних програмних засобах (ASIC, GPU тощо).	

Другою за популярністю є *Brain.js*. Дані бібліотека Javascript використовується для нейронних мереж та добре спрівпрацбє з Node.js, а також у браузері (надає змогу обчислювати примітки). *Brain.js* поміщає в собі різні типи нейронних мереж, в залежності від задачі, яку потрібно вирішити. Переваги та недоліки *Brain.js* описано в таблиці 3.2.

Таблиця 3.2.

### Таблиця переваг та недоліків *Brain.js*

Переваги	Недоліки
У доступі є багато бібліотек з відкритим кодом.	Для шарів softmax та деяких інших структур не існує багато можливостей.
Ідеально підходить для швидкого створення найпростіших нейронних мереж, застосовуючи мови високого рівня	Обмежує мережеву архітектуру до того моменту, коли потрібно використовувати складніші алгоритми та програми.
Не потрібно писати багато коду та мати великий набір даних для того, щоб створити цікавий функціонал	
Є можливість на стороні клієнта працювати на JavaScript.	

Іншою JS бібліотекою загального призначення для реалізації машинного навчання є *Ml.js*, яка призначена для Node.js та браузерів. Вона забезпечує розробника

утилітами, що допоможуть вирішити контролювані та неконтрольовані проблеми машинного навчання, а також критичні для задачі та прості моделі. Опираючись на всебічне машинне навчання та простоту, як один з головних принципів, Ml.js забезпечує TypeScript та JavaScript розробників декомпозицією, лінійними моделями, кластеризацією та іншим. Переваги та недоліки Ml.js описано в таблиці 3.3.

*Таблиця 3.3.*

**Таблиця переваг та недоліків Ml.js**

Переваги	Недоліки
Дана технологія робить можливими бітові операції над масивами та хеш-таблицями, а також сортуванням та генерацією випадкових чисел.	Обмежена підтримка самого апаратного прискорення.
Для маніпуляцій над масивами, лінійної алгебри та оптимізації є свої прописані процедури.	Не передбачає дефолтного доступу до файлової системи у хост-середовищі веб-переглядача.
Підтримує перехресну перевірку.	

Одним з найпростіших та водночас найшвидших способів побудувати API, які будуть нескінченно масштабовані і при цьому «само-лікуватись» є *Stidlib*. Популяризована завдяки AWS Lambda, базою даної стандартної бібліотеки є Функція у ролі Сервісної («безсерверної») архітектури. Вона надає можливість за лічені хвилини створювати для себе або ж інших розробників модульні та масштабовані API. При цьому не буде ніякої потреби створювати SDK, керувати доменами, серверами або шлюзами, чи писати документацію.

Отже, ця бібліотаке спрошує усю роботу розробника. Все, що йому потрібно робити – це писати код, а обробить вона його замість нього. Переваги та недоліки даної бібліотеки для впровадження в систему машинного навчання, *Stidlib*, описано в таблиці 3.4.

Таблиця 3.4.

**Таблиця переваг та недоліків Stidlis**

Переваги	Недоліки
Забезпечений легкий обмін API, а також контроль доступу до них.	Відсутня підтримка обчислень інверсного гіпербологічного секанту.
Документація, що створена автоматично.	Створені проекти, що не містять тверджень про запуск та виконання також не підтримуються.
Дуже висока продуктивність.	
Надійні статистичні та математичні функції.	
Утиліти для розробки бібліотек і додатків є широко досліджені і перевірені.	

Іншою відкритою бібліотекою для створення нейронних мереж, а також застосування глибинного навчання є *ConvNetJS*. Вона забезпечує навчання нейронних мереж у браузерах. Вирішує проблеми регресії та класифікації. Більше того, у ньому присутній на разі експериментальний модуль, що навчає з підкріпленим. Також бібліотека відома тим, що забезпечує підтримку згорткових нейронних мереж, що використовуються при розпізнаванні зображень.

Таблиця 3.5.

**Таблиця переваг та недоліків ConvNetJS**

Переваги	Недоліки
Допомагає формувати та вирішувати проблему створення нейронних мереж з використанням JavaScript.	Досить складний в управлінні, новачкам буде важко розібратись.
Тренує нейронні мережі глибинного навчання повністю у браузері.	Обробка інформації часто відбувається набагато довше, ніж у інших аналогах даної технології.
Не потрібно ніяких додаткових компліторів, чи встановлених засобів.	

Таким чином, проаналізувавши п'ять найбільш популярних відкритих бібліотек для побуди нейронних мереж на JavaScript, було обрано використовувати TensorFlow.js. Данна бібліотека її досить простою для вивчення, а також ідеально підходить для вирішення задач у нескладних системах. Також вона забезпечує багато типів нейронних мереж та не потребує глибоко їх розуміння для впровадження машинного навчання у систему.

### **Висновок до третього розділу**

В третьому розділі проаналізовано технології для створення системи каучсерфінгу для домашніх тварин на основі машинного навчання. Досліджено технологія для написання клієнт та серверної частин веб-сайту. Обрано усі необхідні засоби для відповідної реалізації систем, серед яких ключовими є JavaScript, React, Redux, Node.js, Express. Також обрано середовище розробки в залежності від технологій Visual Studio Code. Окремо проаналізовано методи машинного навчання. Обрано метод, що найбільше підходить для вирішення задач у системі каучсерфінгу для домашніх тварин, а саме нейронну мережу глибинного навчання. Проаналізовано інструменти реалізації машинного навчання та обрано найбільш релевантний для системи – TensorFlow.js.

## РОЗДІЛ 4

### Практична реалізація

#### **4.1. Опис створеного програмного засобу**

##### *4.1.1. Загальні відомості про програму*

Система реалізована у вигляді веб-сайту. «PetShelter» призначений для можливості надання тимчасового притулку домашнім тваринам, шляхом пошуку хостів, що згідні прийняти домашнього улюбленаця іншої людини. Додаток реалізовано за допомогою багатьох технологій, основною з яких є мова JavaScript, а у відповідності до неї бібліотека React, Redux, на серверній частині платформа Node.js та фреймворк Express. Для зберігання усіх необхідних даних використовується сховище даних PostgreSQL.

##### *4.1.2. Структура бази даних*

Для реалізованого веб-сайту «PetShelter» використовується база даних PostgreSQL. Вона складається з п'яти таблиць: Users, userAbout, userHome, userPhotos, Pets, Requests, RequestPets, Feedbacks. Розглянемо детальніше, яку інформацію зберігає кожна з таблиць та які поля містить в собі.

Таблиця Users зreibігає дані про усіх зареєстрованих користувачів системи. Варто зазначити, що користувачі не поділяються на тих, хто може тільки шукати притулок своїй тваринці, і на тих, хто тільки його надає. Користувач може як і шукати хоста своїй тваринці, так і бути хостом для іншої. За бажання хостити чиось тваринку відповідає поле availability. Також таблиця містить інші поля: id, name, birthDate, birthdate, userImg, email, city, gender, phone, password, lastOnline, verified.

Таблиці userAbout, userHome та userPhotos створені для того, щоб не переповнювати таблицю Users інформацією. Логічнішим рішенням було цю інформацію структурувати та розподілити.

У таблиці userAbout міститься інформація про користувача та пояснюється чому він користується «PetShelter». У ній є такі поля: id, description, whyOnPetShelter, a

також вторинний ключ userId, що реалізує зв'язок «один-до-одного» з таблицею Users.

У таблиці userHome зберігаються дані про домівку користувача, що буде корисною для користувачів, які захочуть залишити у нього свою тваринку. У ній присутні наступні поля: id, maxPets, whichPets, lastMinReq, hasPets та вторинний ключ userId, що реалізує зв'язок «один-до-одного» з таблицею Users.

У таблиці userPhotos зберігаються фотографії користувача, а саме є такі поля: id, imgURL, description та вторинний ключ userId, що реалізає зв'язок «один-до-багатьох» з таблицею Users. Тобто в одного користувача може бути багато фото.

Таблиця Pets містить інформацію про тваринок. Данна таблиця має такі поля: id, name, birthdate, petImg, kind, vaccination, passport, а також вторинний ключ userId, що реалізує зв'язок «один-до-багатьох» з таблицею Users. Це означає, що один користувач може мати багато домашніх тваринок.

Таблиця Requests містить інформацію про створений користувачем запит, щоб залишити тваринку в бажаного хоста. У ній є наступні поля: id, requesterUserId, responderUserId, message, status.

Таблиця RequestPets зберігає інформацію про тваринок, які мають бути у запиті, а також реалізує зв'язок «багато-до-багатьох» між таблицею Pets та таблицею Requests. У ній містяться такі поля: id, вторинний ключ petId та вторинний ключ requestId.

Таблиця Feedbacks зберігає усі необхідні дані про відгуки. Завдяки полю requestId реалізується зв'язок з таблицею Requests, з якої є можливість дістати дані про обидві сторони запиту. Також дана таблиця містить поля feedbackMessage та rate.

У всіх таблицях поле id створюються автоматично.

Структуру бази даних продемонстровано на рис.4.1. Схему створено, використовуючи сервіс [ondras.zarovi.cz](http://ondras.zarovi.cz).

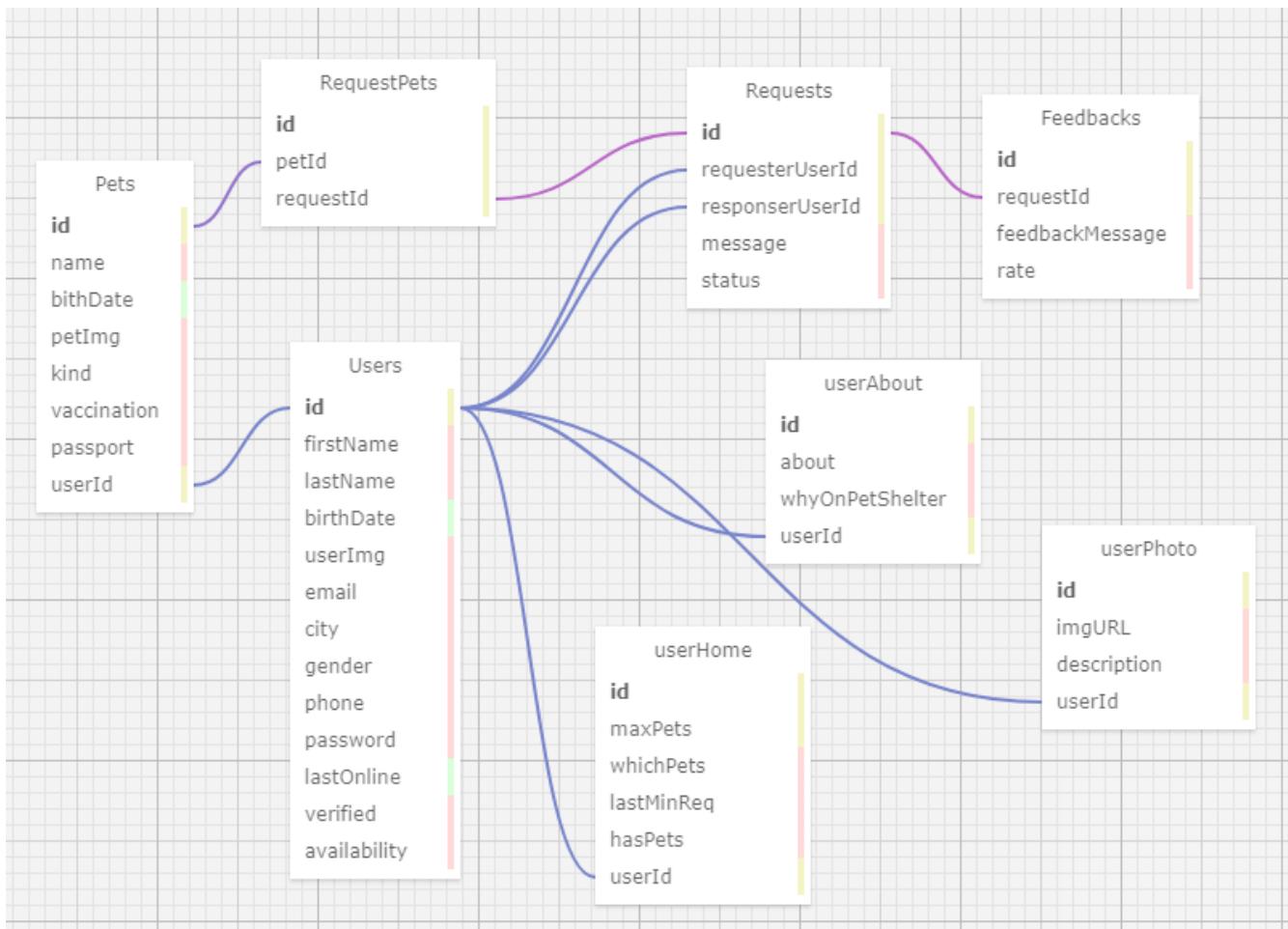


Рис. 4.1. Схема бази даних

#### 4.1.3. Функції, що виконуються програмою

Розроблений програмний продукт містить два типи ролей: користувач та адміністратор.

Програмою забезпечена функція реєстрації нового користувача, а також логінація на веб-сайт у випадку, коли користувач вже зареєстрований. Перед тим, як реєструватись користувач може почитати на окремих сторінках веб-сайту, як саме працює сервіс та що робити, щоб користування сервісом було якомога безпечніше. Після реєстрації користувач верифікується та заповнює профіль усією необхідною інформацією. Системою надати можливість редагувати цю інформацію.

Також завдяки створеному програмного продукту можна знаходити потенційних хостів для тваринки, вибравши в пошуку місто, у якому має проживати хост. Однією з основних функцій є підбір декількох рекомендованих хостів, який базується на

попередньому досвіді користувача. Дану функцію реалізовано за допомогою нейронної мережі глибинного навчання, на вхід якій подається інформація перетворена на цифри про минулий досвід користувача. Досвід береться з минулих запитів, що створювали користувач. При цьому, виділяються такі дані: стать та наявність тварин. Чоловік – «1», жінка «2», інша стать – «3», а також наявність тварин – «1» є тварини, «2» – тварин немає. Також на пошук впливає місто, в якому проживає сам власник тваринки. Якщо попереднього досвіду не існує, то рекомендовані хости підбираються по рейтингу.

Якщо ж рекомендовані хости не підійшли користувача або коло пошуку потрібно більш конкретизувати, система надає можливість фільтрувати користувачів. При цьому користувач може вибрати такі фільтри: місто, стать, статус, наявність тварин, наявність референсів. Після цього користувач може переглянути профіль усіх хостів, а також надіслати запит бажаному хосту. Також програма надає можливість вибрати тваринок для запиту, адже у випадку, якщо у власника багато тваринок різних видів, то важко буде знайти хоста, який зможе прийняти їх усіх.

Усі запити відображаються на окремій сторінці веб-сайту та поділяються на активні та завершені.

Після того, як запит оброблений та погоджений обома сторонами, система відкриває їх контактні дані, щоб вони могли сконтактуватись та домовитись про деталях трансферу тваринки.

Користувач має завершити запит, після чого системою реалізована можливість написати відгуки один одному протягом 14 днів. Варто зазначити, що відгуки будуть відображатись на профілях користувачів, лише у випадку, якщо їх написали обидві сторони. Програмний продукт не надає можливість редагувати чи видаляти відгуки, тим самим робить систему більш надійнішою.

Адміністратор – це безпосередньо розробник веб-сайту, а це означає, що у нього є доступ до усіх даних. На даний момент, саме з його електронної пошти відправляються листи для верифікації на електронні пошти користувачів.

#### *4.1.4. Опис логічної структури веб-сайту*

Структура веб-сайту передбачає різні способи переходу між сторінками всередині файлу і тому є павутинноподібною. На головній сторінці веб-сайту є такі елементи: навігаційне меню, основна частина та банер. Навігаційне меню основній сторінки містить називу сервісу та можливість переходу на сторінки «Як це працює» та «Безпека», а також «Ввійти в систему». На головній сторінці є форма реєстрація та речення, що коротко описує, чим може бути корисна система. Також в них є банер, що повідомляє про використання cookies. Після входу в систему міняється лише навігаційне меню та наповнення основної частини, в залежності від того, на якій вкладці навігаційного меню знаходиться користувач. Вкладки є наступними: поле пошуку, «Головна», «Профіль», «Запити», «Верифікація», випадаюче меню з вкладкою «Налаштування» та кнопкою виходу.

#### *4.1.5. Використовувані технічні засоби*

Для коректної роботи веб-сайта необхідно мати наступне програмне забезпечення:

- Chrome 23+;
- Opera 15+;
- Edge 12+;
- Safari 6+;
- Internet Explorer 10+;
- Firefox 21;+
- Arduin Browser 4.4 +.

Усі ці браузери підтримують ECMAScript6.

#### *4.1.6. Виклик та завантаження*

Для того, щоб у будь-якого користувача інтернетом з'явилася можливість протестувати програмний продукт було вирішено задеплоїти веб-сайт на безплатний хостинг Heroku. На даному хостингу встановлений Node.js, який необхідний для

коректної роботи сайту, а також завантажений менеджер пакетів прт. Окрім цього, потрібно встановити зв'язок з базою даних для того, щоб усі дані мали де зберігатись, а також, щоб виконувати над ними прописані у функціоналі системи операції. Після цього варто буде купити платний хостинг, який буде більш безпечний та потужний для великої кількості користувачів.

#### *4.1.7. Вхідні та вихідні дані*

Вхідними даними, що отримує система є:

- дані про користувачів;
- дані про тваринок;
- запити зроблені користувачем.

Вихідними даними є:

- тимчасовий притулок для тваринки;
- відгуки.

## **4.2. Інструкція користувача**

### *4.2.1. Вступ*

Веб-сайт створений для того, щоб допомогти користувачеві знайти тимчасовий притулок для своєї домашньої тваринки. При цьому, залишаючи її в іншої людини, зовсім безплатно. Власнику тваринки необхідно буде лише забезпечити харчування тваринки та ліки, якщо є така необхідність. Веб-сайт побудований на волонтерському принципі, головним рушієм якого є 2 чинники. Першим, є бажання допомогати один одному, а другим – любов до тварин.

Веб-сайт також буде приносити нові знайомства самотнім людям, адже саме такі люди, зазвичай, не мають знайомих у місті, де проживають, щоб залишити тваринку.

### *4.2.2. Загальні відомості про програму*

Програмний продукт названо «PetShelter». Для зручності у використанні його реалізовано у вигляді веб-сайту. Для написання інтерфейсу використано такі

технології технології: html, css, javascript, react, redux, typescript, react styled components, react hook forms, material-ui, axios, yup. Для написання серверної частини було використано node.js, express, posgreSQL, sequelize, brain.js, jwt, joi. Веб-сайт підтримується усіма основними сучасними веб-переглядачами на будь-якій платформі.

#### *4.2.3. Класи вирішуваних завдань*

Веб-система «PetShelter» створена для вирішення трьох основних задач: знайти тимчасовий притулок для домашньої тварини, надати тимчасовий притулок домашній тварині та познайомитись з новою людиною. Користувач при реєстрації у системі не зобов'язаний обирати чи хоче він бути лише хостом, чи хоче лише шукати потенційних хостів. У певні періоди свого життя він може, як і надати тимчасовий притулок у себе вдома іншій тваринці, так і в разі, наприклад, поїздки, знайти хоста своїй тваринці. Користувачі, які не можуть або ж не хочуть приймати інших тварин, можуть обрати відповідний статус у своєму профілі. Також, передача тваринки на певний час іншій людині передбачає знайомство з цією людиною та будування відносин, що базуються на довірі.

Функції, представлені програмним продуктом:

- авторизація (реєстрація та вход і вихід з системи);
- заповнення профіля (додавання короткого опису, інформації про домівку та завантаження фото) та можливість його редагувати;
- додавання та видалення тваринок;
- можливість обрати хоста, шляхом пошуку за містом;
- можливість надати тваринці тимчасовий притулок;
- можливість фільтрування хостів;
- можливість перегляду профілів інших користувачів;
- надсилання запитів бажаним хостам;
- підтвердження/відхилення запиту, зробленого власником тваринки;

- підтвердження/відхилення хоста;
- обмін контактними даними;
- написання відгуку.

#### *4.2.4. Опис основних характеристик і особливостей програми*

Для користування веб-сайтом необхідний постійний налагоджений зв'язок з інтернетом. Від швидкості інтернету буде залежати швидкість загрузки сервісу.

Оскільки, програмний продукт задеплоїний на хостинг, то доступ до нього можна отримати у будь-який час. Проте, варто розуміти, що при виникненні проблем або збоїв на хостингу, користування системою буде неможлив, доки проблема не вирішиться.

Також, однією з особливостей веб-сайту є те, що його варто відкривати у веб-переглядачах, що підтримують ECMAScript6.

#### *4.2.5. Відомості про функціональні обмеження на застосування*

Як уже було згадано вище, для повноцінної та правильної роботи веб-сайту необхідне підключення до інтернету (бажано швидкісного), а також браузер, що підтримує специфікацію ES6.

Також, для можливості безпоседньої інтеракції з системою необхідні засоби, що забезпечать ввід і вивід даних.

Що стосується основних вимог до розробленого програмного продукту «PetShelter», то необхідно мати Windows 7+, або для мобільних пристроїв IOS 7+, Android 4.1 +.

### **4.3. Аналіз контрольного прикладу**

Реалізовано web-систему каучсерфінгу для домашніх тварин на основі машинного навчання - PetShelter. При аналізі схожих систем було виявлено:

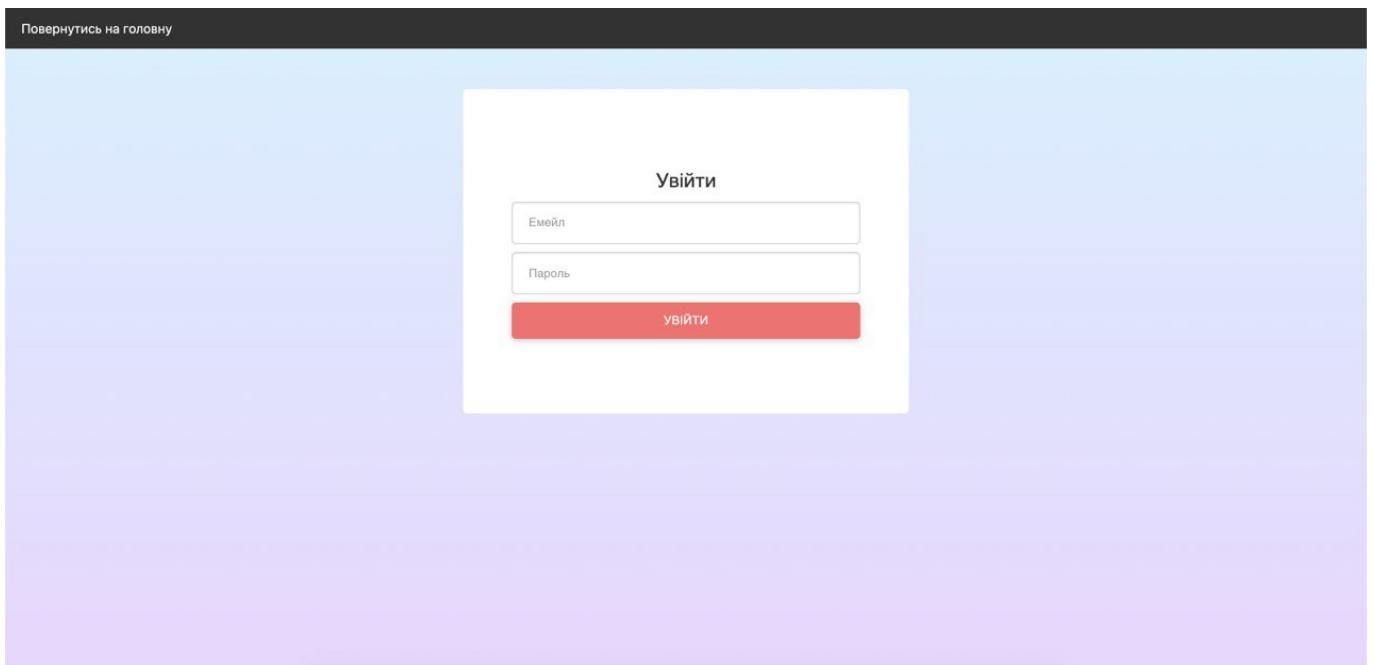
- для кращого сприйняття варто використовувати мінімістичний дизайн, адже він найкраще сприймається користувачами та полегшує роботу з системою;

- як додатковим атрибутом у дизайні використовувати емоджі, адже за останніх декілька років вони стали заміною слів, і викликають емоції;
- полегшити роботу з системою, за допомогою машинного навчання, яке підбираємо користувачам потрібних їм хостів.

Отже, розглянемо як функціонує сайт та які кроки поступово виконує користувач. Спершу, потенційний користувач потрапляє на сторінку реєстрації, де є короткий опис для чого наш сайт та форма реєстрації, де потрібно заповнити основні дані про користувача. Також є можливість перейти на сторінки «Як це працює», де описується як працює сайт, «Безпека», де надані поради, як зробити користування системою безпечної та знайти хоста, який справді подбає про вашу тваринку, та «Увійти», якщо користувач вже зареєстрований в системі. Також внизу сторінки висвітлюється повідомлення про використання файлів cookie.

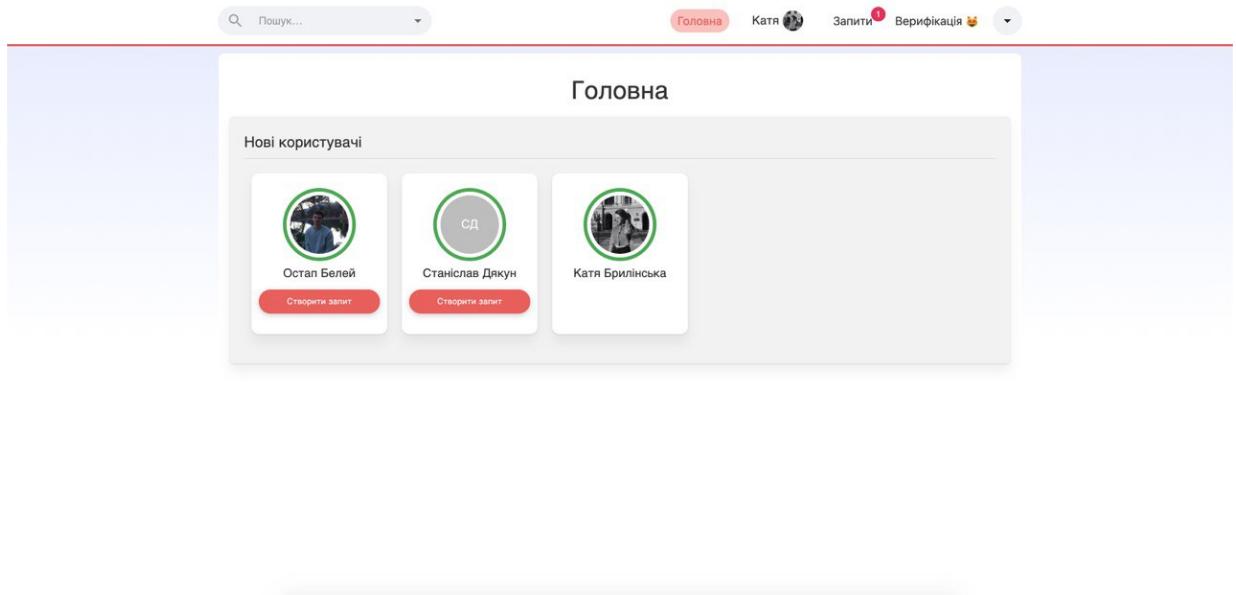
*Рис. 4.1. Сторінка реєстрації.*

У випадку, якщо користувач вже зареєстрований в системі, він переходить на сторінку «Увійти» та вводить свої дані, які відправляються в базу даних і при успішному запиті формується токен, що відповідає за сесію користувача. Нижче зображеній сторінку входу.



*Рис. 4.2. Сторінка входу.*

Після того як користувач ввійшов в систему, він опиняється на головній сторінці, де продемонстровано користувачів в системі.



*Рис. 4.3. Головна сторінка.*

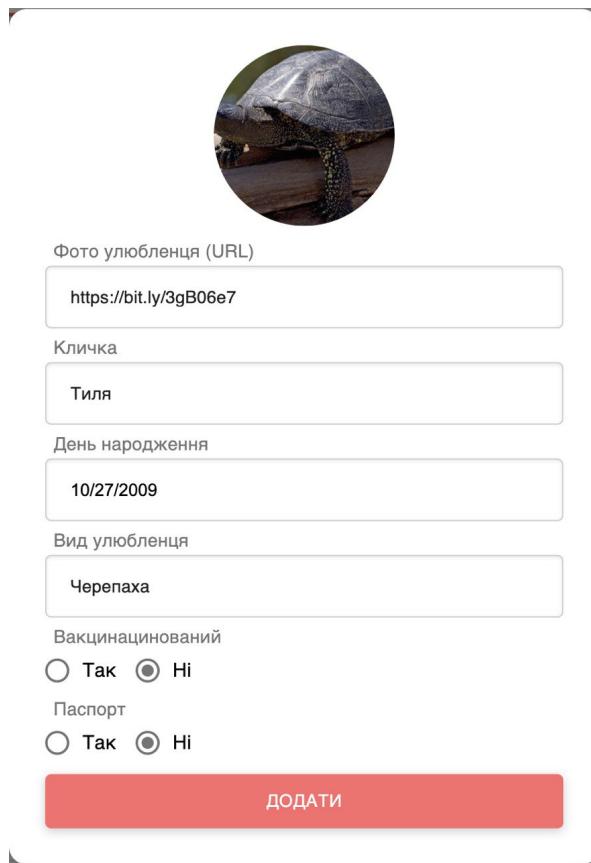
Далі можемо розглянути профіль користувача. Зліва ми бачимо профільне фото, ім'я та прізвище, місто проживання, чи верифікований користувач та дату останнього активного перебування в системі, або ж статус онлайн, якщо юзер зараз в системі. Також зображене, чи користувач приймає тваринок чи ні. Правіше, розташоване навігаційне меню для профілю користувача, в якому присутні такі вкладки: «Про

мене», «Мої тваринки», «Мій дім», «Фото» та «Відгуки». Кожна з вкладок розповідає про користувача, його тваринок, його місцю на сайті та містить відгуки, які допоможуть іншим людям краще зрозуміти чи варто довіряти даному користувачу.

*Рис. 4.4. Профіль користувача. Вкладка «Про мене».*

Далі розглянемо інші вкладки профіля користувача. Вкладка «Мої тваринки» містить інформацію про тваринок користувача (кличка, вид, вік, наявність вакцинацій та паспорту). Тваринок можна добавляти за допомогою відповідної кнопки та видаляти. Можливість видалити тваринку з'являється при наведенні мишкою на блок тваринки.

*Рис. 4.5. Тваринки користувача.*



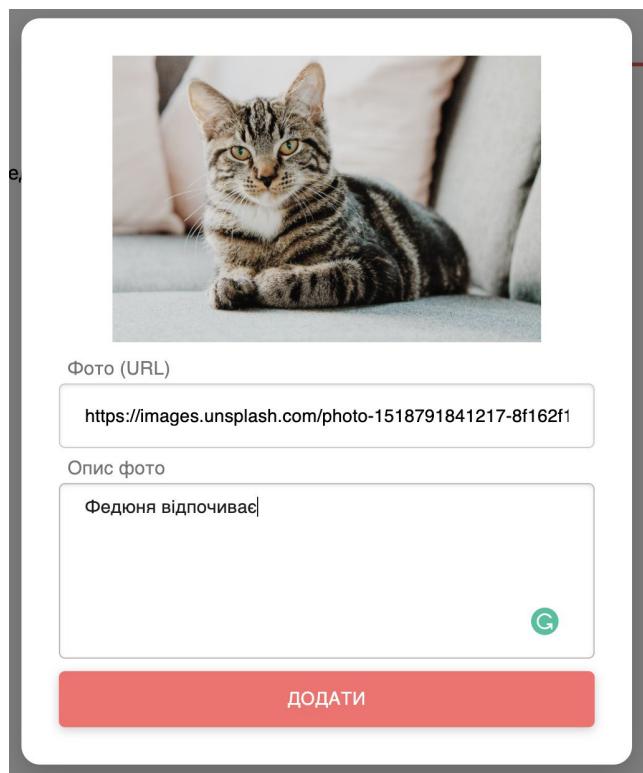
*Рис. 4.6. Функція додавання тваринки.*

На рис. 4.7 зображена вкладка «Мій дім», що містить основну інформацію про дім користувача.

*Рис. 4.7. Вкладка «Miy dim».*

На рис. 4.9 продемонстровано вкладку «Фото». Туди можна добавляти як власні фото, так і фото тваринок, чи куточку для тваринок. Будь-яку інформацію, яка буде

корисна для інших користувачів. Щоб додати фото, потрібно натиснути відповідну кнопку.



*Рис. 4.8. Функція «Додати фото».*

*Рис. 4.9. Вкладка «Фото».*

Також, якщо користувач вже приймав участь у якомусь запиті і виступав або хостом, або власником тваринки, то відгуки про нього та його тваринку міститимуться у відповідній вкладці «Відгуки». Ключовим у відгуку є позначка «Рекомендую» або ж «Не рекомендую». Якщо у користувача немає відгуків, то там буде відповідне повідомлення.

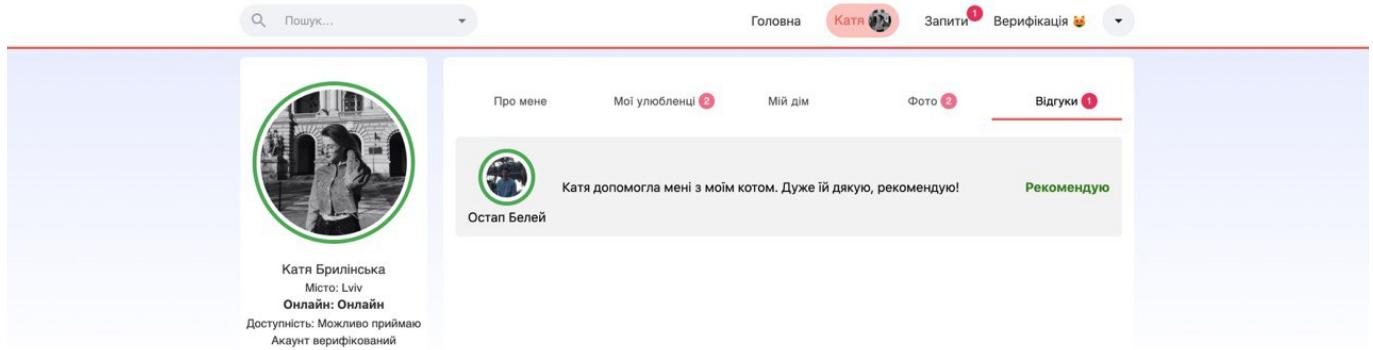


Рис. 4.10. Вкладка «Відгуки».

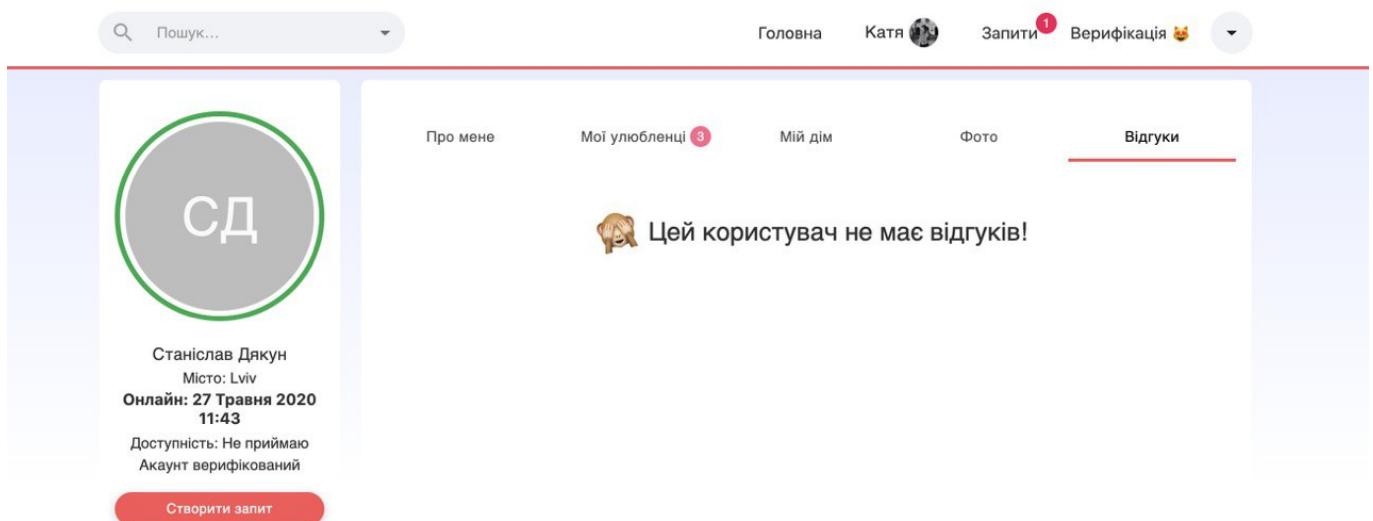
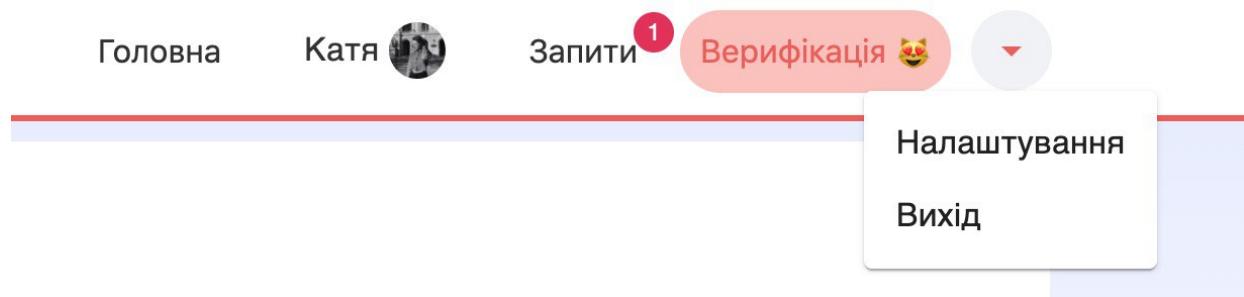


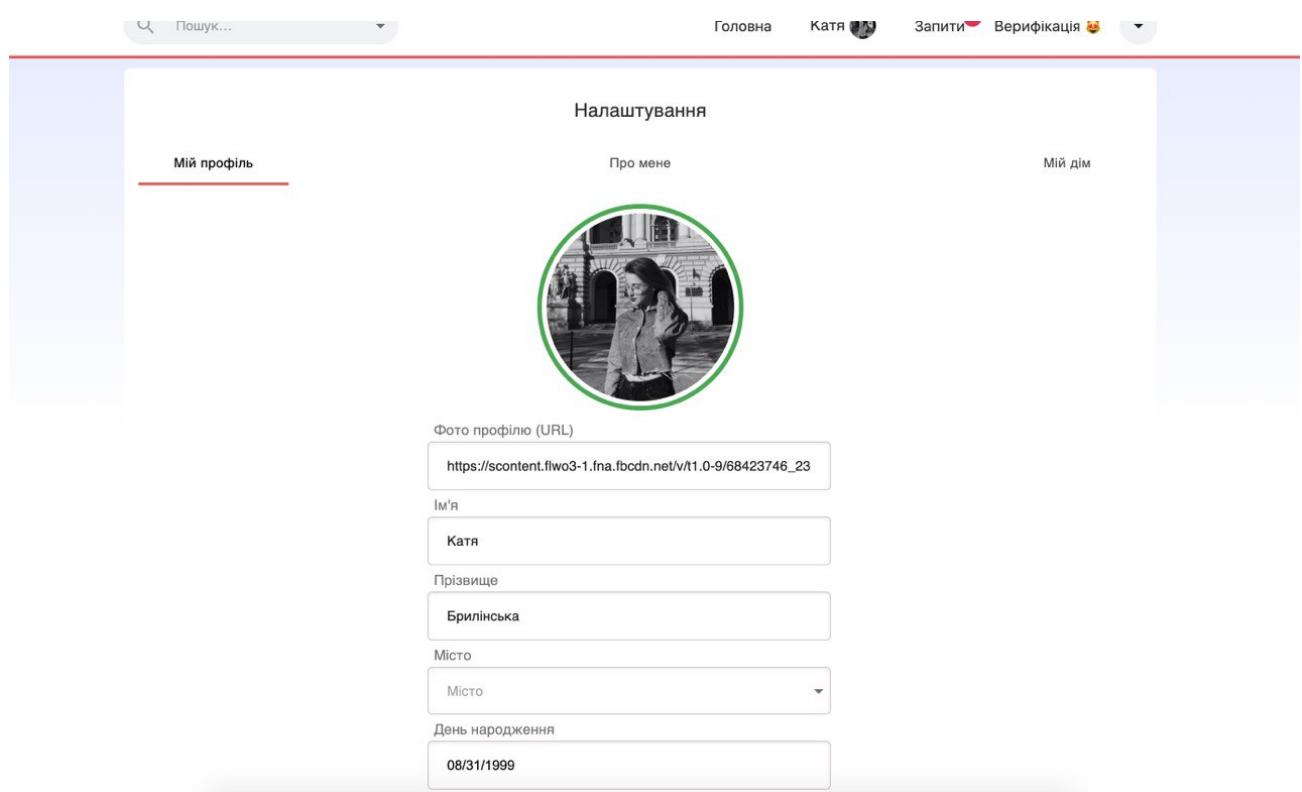
Рис. 4.11. Вкладка «Відгуки» при відсутності відгуків.

Інформацію про себе можна редагувати, перейшовши на випадаюче меню у головному навігаційному меню, та натиснувши на кнопку «Налаштування». Після цього можна буде редагувати основну інформацію, інформацію про себе та про свій дім. Щоб зберегти нові дані, потрібно натиснути на кнопку «Оновити дані». Також

дане випадаюче меню реалізує вихід з системи. Потрібно просто натиснути на саму кнопку «Вихід».



*Рис. 4.12. Випадаюче меню з кнопками «Налаштування» та «Вихід».*



*Рис. 4.13. Редагування основної інформації.*

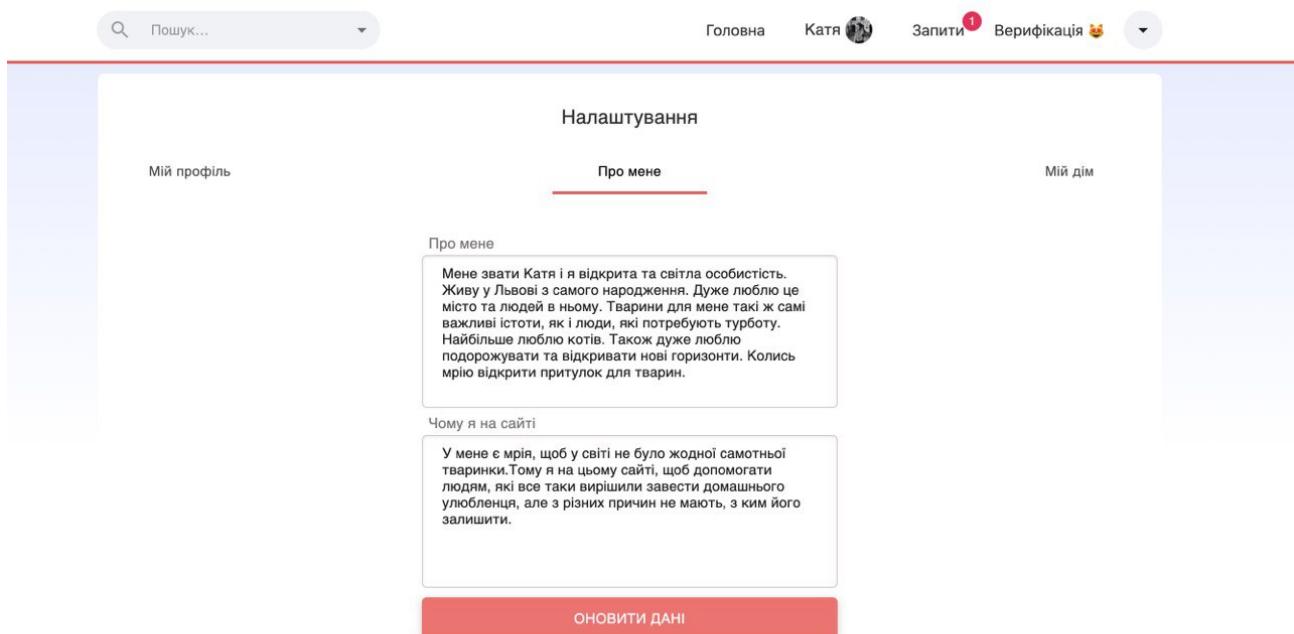


Рис. 4.14. Редагування інформації про себе.

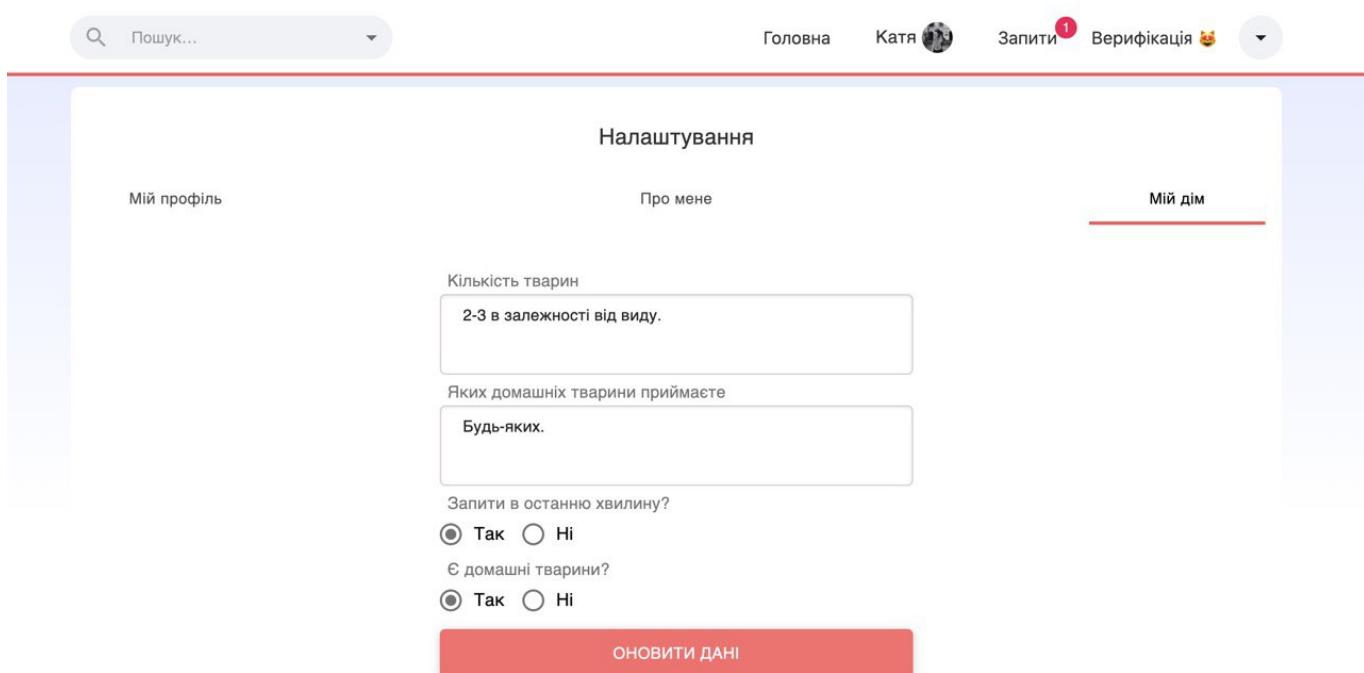
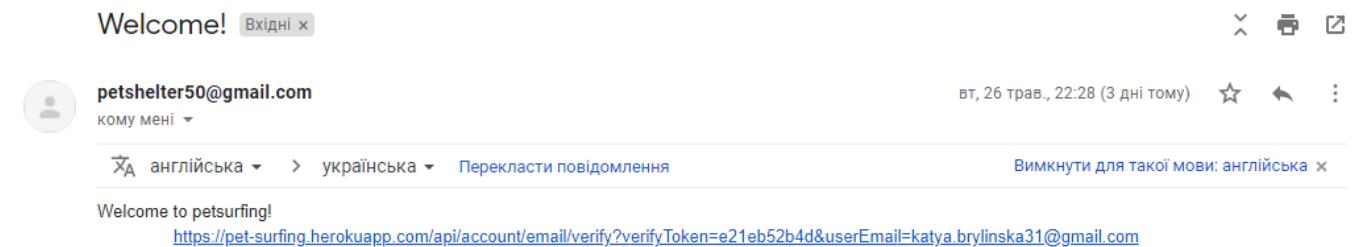


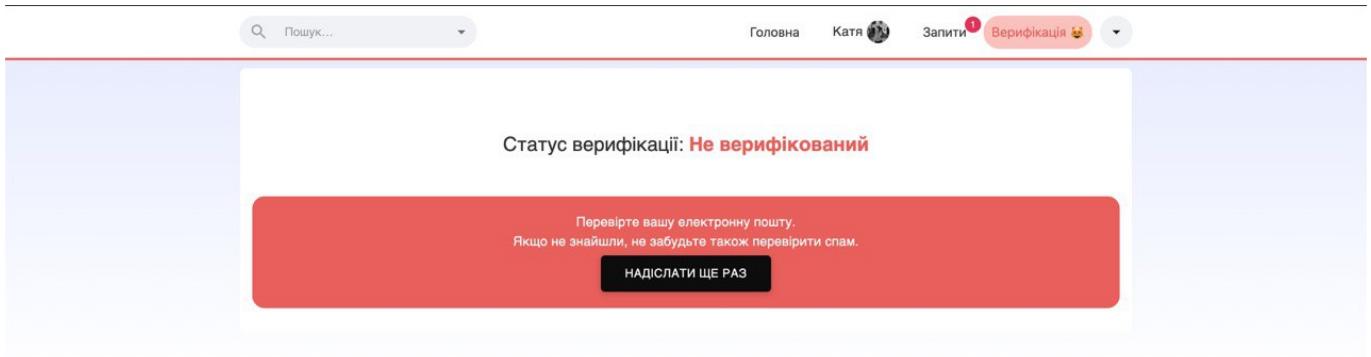
Рис. 4.15. Редагування інформації про свій дім.

Кожен профіль користувача має бути верифікований. Щоб верифікувати акаунт, користувач повинен перейти по посиланню, яке буде надіслане йому системою на вказану електронну адресу. Приклад листа зображенний на рис. 4.16.



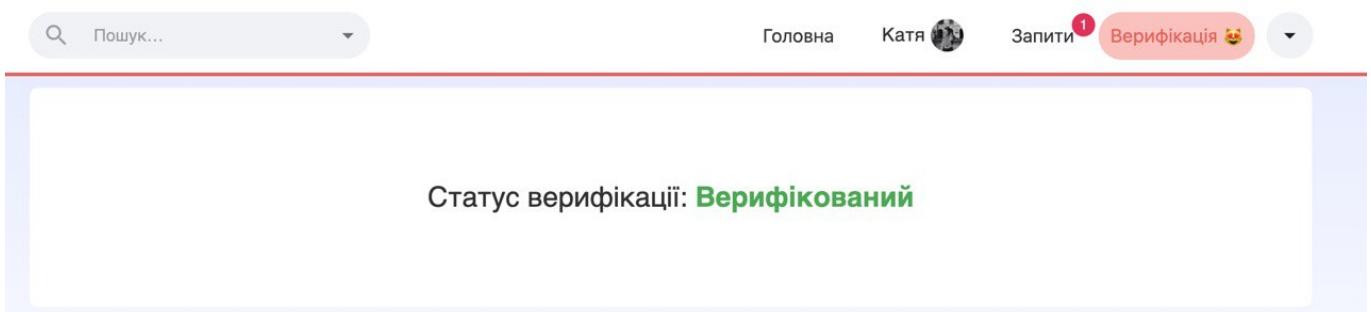
*Рис. 4.16. Електронний лист з посиланням для підтвердження акаунту.*

Доки акаунт не верифікований у вкладці «Верифікація» користувачу буде висвітлене відповідне повідомлення з вказівками.



*Рис. 4.17. Сторінка «Верифікація», якищо акаунт не верифікований.*

У випадку, якщо акаунт верифікований, то дана вкладка буде виглядати по-іншому. Зображене на рис. 4.18.

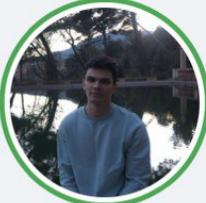


*Рис. 4.18. Сторінка «Верифікація», якищо акаунт верифікований.*

Для того, щоб знайти бажаного хоста потрібно натиснути на іконку «Лупа», щоб побачити користувачів або ж вибрати місто, в якому шукаєте хоста, з випадаючого списку. Варто зазначити, що у випадку, якщо користувач уже має хоча б 1 завершений запит, то система буде рекомендувати йому користувачів за допомогою машинного

навчання. Якщо ж користувач немає створених запитів, а лише приймав тваринок, або ж не приймав взагалі, то система показуватиме список користувачів у його місті.

**Користувачі**



**Остап Белей**  
Вік: 22  
Місто: Lviv  
Онлайн: 29 Травня 2020 14:37  
Улюблениці: Має 1 улюблениці.  
Чому я на сайті: Щоб допомагати іншим людям, коли ім потрібна допомога з іхніми тваринами, і тако...



**Станіслав Дякун**  
Вік: 20  
Місто: Lviv  
Онлайн: 27 Травня 2020 11:43  
Улюблениці: Має 3 улюблениці.  
Чому я на сайті: Я тут, тому що маю багато місця дома!

Рис. 4.19. Сторінка «Пошуку» для користувача, що не створював жодного запиту.

**Користувачі**

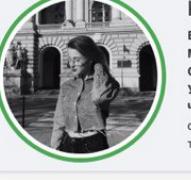
Рекомендовані користувачі



**Катя Брилінська**



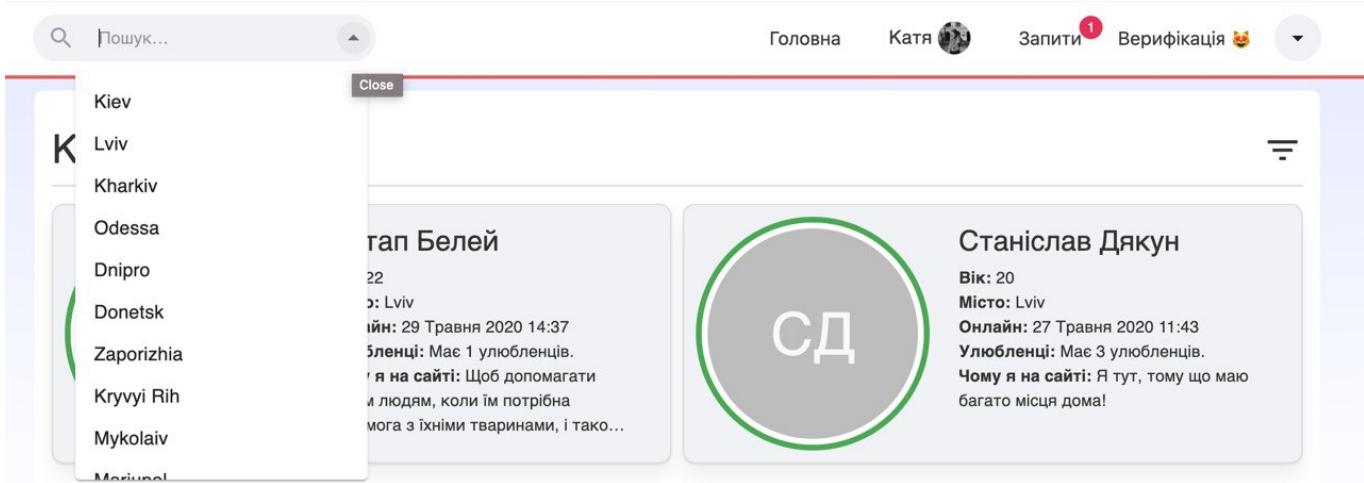
**Станіслав Дякун**  
Вік: 20  
Місто: Lviv  
Онлайн: 27 Травня 2020 11:43  
Улюблениці: Має 3 улюблениці.  
Чому я на сайті: Я тут, тому що маю багато місця дома!



**Катя Брилінська**  
Вік: 20  
Місто: Lviv  
Онлайн: Онлайн  
Улюблениці: Має 2 улюблениці.  
Чому я на сайті: У мене є мрія, щоб у світі не було жодної самотньої тваринки. Тому я на цьому сайті, що...

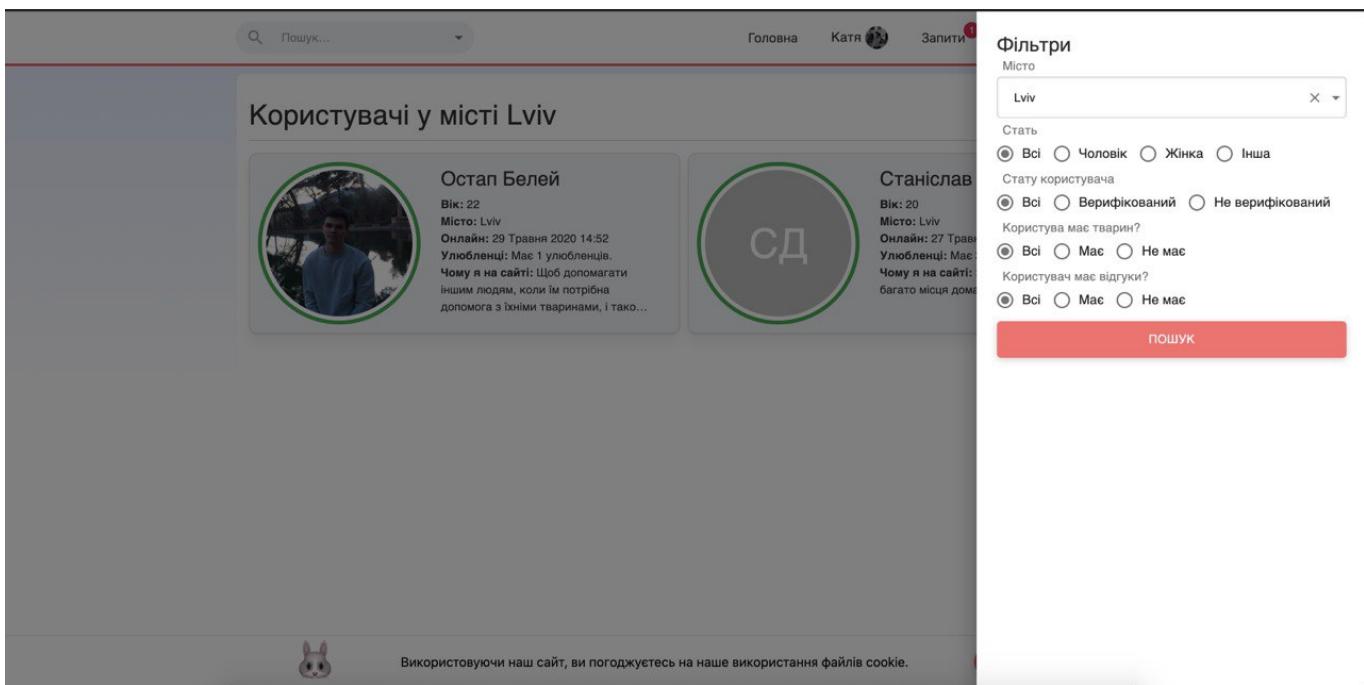
Рис. 4.20. Сторінка «Пошуку» для користувача, що створював хоча б один запит.

Як можна побачити з попередніх скріншотів, для користувача Остапа, що має вже завершений створений ним запит, система рекомендує користувача Катю. Для користувача Каті ж система ж не може порекомендувати жодного користувача, адже Катя не створювала запитів, а тому не відомо критеріїв її вибору.



*Рис. 4.21. Випадаючий список найбільших міст України.*

Окрім того, якщо користувач має певні критерії вибору хоста він може задати їх, використовуючи фільтри. За допомогою фільтрів користувач може обрати місто, стать, статус, наявність тварин та відгуків.



*Рис. 4.22. Фільтри для пошуку.*

Після того, як користувач задав усі критерії для пошуку, він має можливість переглянути профіль бажаного хоста. Під фотографією хоста та ключовою інформацією про нього знаходиться червона кнопка «Створити запит», натиснувши яку користувач може надіслати запит обраному хосту.

Пошук...

Головна Катя Запити 1 Верифікація 1

Про мене Мої улюбленці 1 Мій дім Фото 1 Відгуки 1

**Overview**

Вік: 22 Стать: Чоловік  
Користувач з 2020 Місто: Lviv

**Про мене**

Обожнюю подорожі, книги, знайомства з новими людьми і тварин 🐾

**Чому я на сайті**

Щоб допомагати іншим людям, коли їм потрібна допомога з їхніми тваринами, і також знайти хороших людей, яким я зможу залишити своїх.

Створити запит

Рис. 4.23. Профіль іншого користувача.

При створенні запиту власник тваринки повинен написати коротке повідомлення хосту та обрати, які саме улюбленці потребують тимчасового притулку.

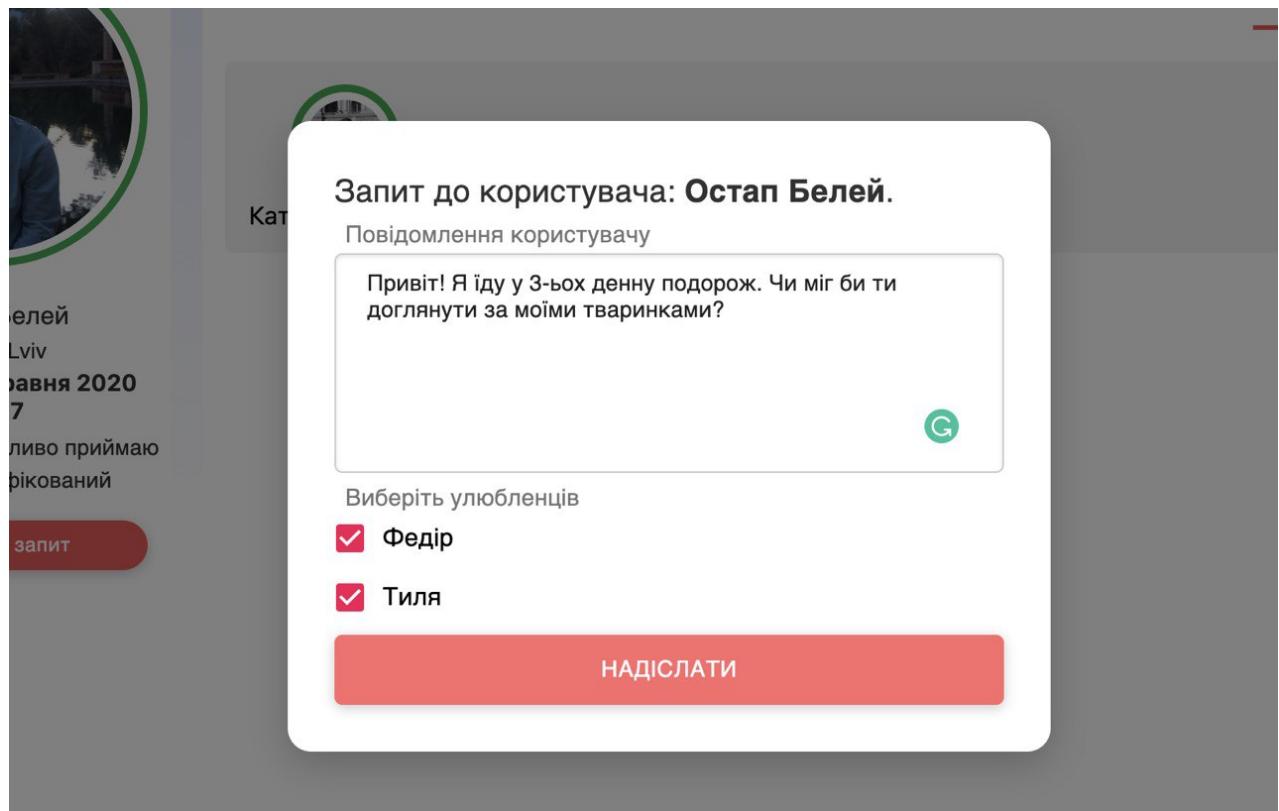
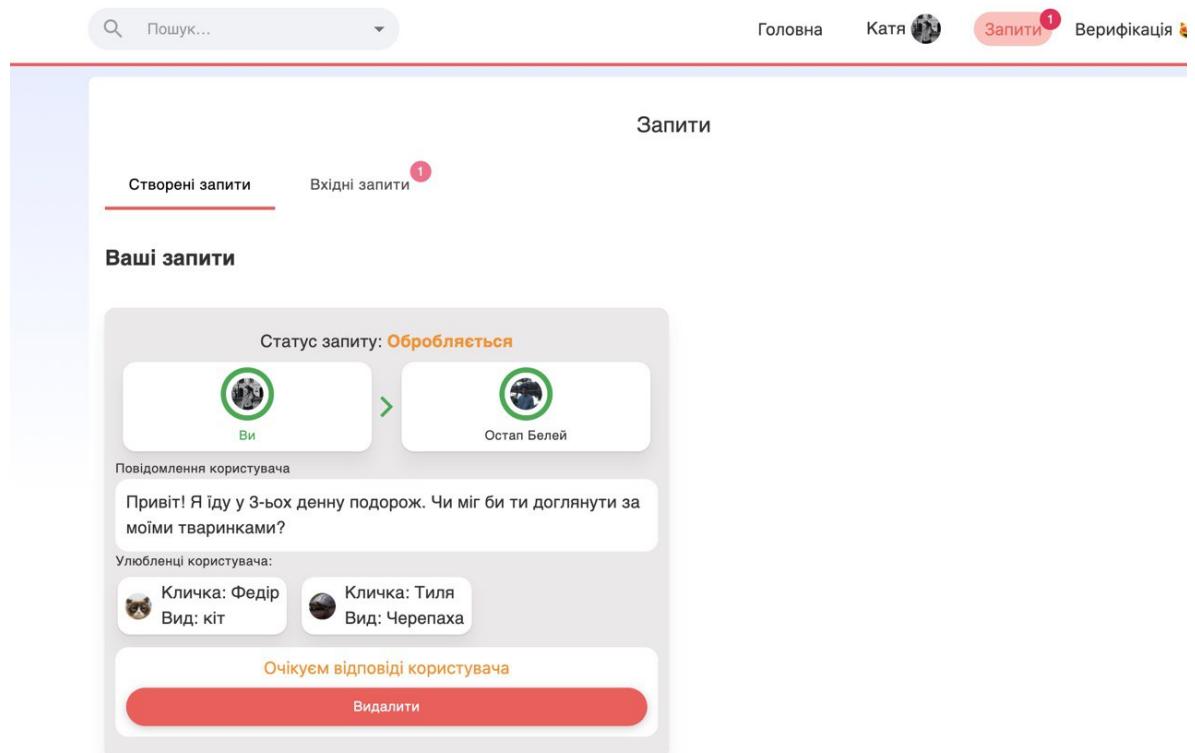


Рис. 4.24. Створення запиту обраному хосту.

Наступною у головному навігаційному меню є кнопка «Запити». Перейшовши на дану сторінку на екрані появиться сторінка з запитами – Створеними та Вхідними запитами.

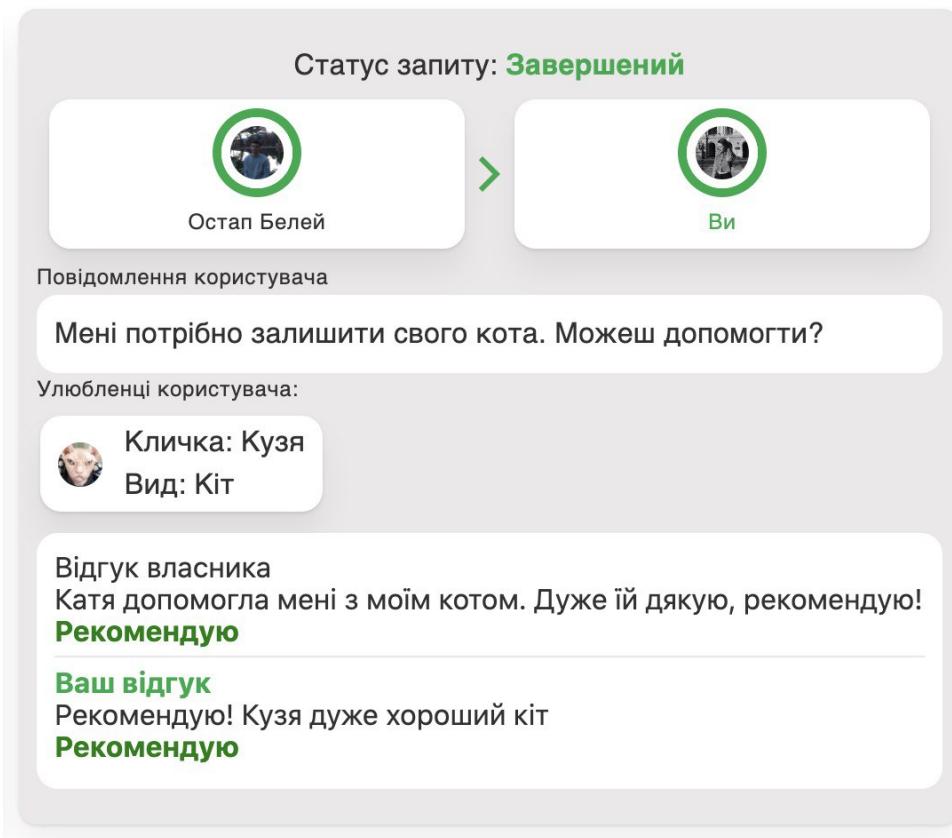
На рис. 4.25 зображено, як виглядає сам запит, поки він знаходить в обробці. Варто відзначити, що для кращого результату користувачу варто надіслати запит декільком потенційним хостам та заради безпеки поспілкуватись з ними на особистій зустрічі для того, щоб впевнитись чи буде його тварині добре з обраною людиною. Система в свою чергу спроектована з акцентом на безпеку, для цього створена система відгуків всередині сайту, які не можна редагувати та видаляти. Також варто додати, що при надсиланні запиту потенційному хосту варто вказувати, в яких умовах любить перебувати тваринка, щоб вияснити усі важливі моменти наперед.



*Рис. 4.25. Вигляд створеного запиту, що чекає на відповідь хоста.*

Після того, як хазяїн забрав свою тваринку, сайт рекомендує залишити відгук про хоста. При цьому потрібно вибрати головний критерій – Рекомендую або ж Не рекомендую та написати відгук не менше 100 символів, який допоможе іншим користувачам зрозуміти якомога більше про людину, як про хоста. В свою сергу хост

має залишити відгук про тваринку та хоста, чи співпадала онлайн-інформація з реальністю та чи все було добре, або ж навпаки. Особливість відгуків полягає у тому, що їх не можна видаляти та редагувати ні відправником, ні отримувачем. У користувача є 14 днів, щоб залишити відгук. Згодом ці відгуки з'являються у вкладці Відгуки у профілі користувача.



*Рис. 4.26. Завершений запит з відгуками.*

## **Висновок до четвертого розділу**

Під час виконання даного розділу описано програмний продукт у вигляді веб-сайту відповідно до заданого стандарту. Також узагальнено описано відомості про реалізований програмний продукт, функціональне призначення даного продукту, описано структуру бази даних, логічну структуру і усі використовувані технічні засоби, вхідні та вихідні дані, а також виклик та завантаження. Було створено інструкцію користувача, в якій описані усі загальні відомості, а також класи вирішуваних завдань. У розділі описано основні характеристики та особливості

програми і надано інформацію про можливі функціональні обмеження. Проаналізовано контрольний приклад і визначено, що система досягає поставленої цілі та працює у відповідності до вимог.

## РОЗДІЛ 5

### Економічне обґрунтування доцільності роботи

#### **5.1. Економічна характеристика програмного продукту**

Тема дипломної роботи – інформаційна web-система каучсерфінгу для домашніх тварин на основі машинного навчання. Вона розрахована для власного користування – вона надає можливість власникам тварин знайти безплатний тимчасовий притулок для свого домашнього улюблена. Також дана система допоможе людям, які самотні, зустріти та познайомитись з новими людьми зі спільними інтересами та стати частиною спільноти.

Проаналізувавши існуючі варіанти рішень для власників тварин, було вирішено, що створювати таку систему неабияк доцільно в наш час.

Аналогів даній системі наразі не існує, а проблема з тим, де залишити домашнього улюблена, коли їдеш у термінове відрядження чи відпустку – досі залишається невирішеною. Існують так звані «готелі для тварин», проте у них є багато недоліків та вони розташовані лише в великих містах.

Система побудована на принципі каучсерфінгу, що призначений соціальним феноменом. Після проведеного дослідження, зроблено висновок, що система керуватиметься попитом, адже є багато людей, що живуть не в рідному місті, або ж багато подорожують та не мають в кого залишити свою тваринку. Люди завжди більше довіряли людям, тому дана система швидко стане необхідною та популярною серед користувачів.

#### **5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення програмного продукту**

У сучасному світі все більшого розповсюдження набуває «цифрове» життя. Щодня створюються тисячі нових програмних продуктів, які намагаються вирішити проблеми, з якими стикаються люди. Так само і самі люди вже давним давно шукають вирішення цих проблем чи відповіді на усі свої запитання саме в інтернеті. Саме тому

з таким стрімким розвитком сфери технологій – розробники стають одними з найзатребуваніших ресурсів у будь-якій компанії.

Окрім цього, варто зазначити, що програмні продукти вплинули на життя людей з усіх сторін. Як і з технічної, так і з соціальної. Спілкування між людьми стало набагато легшим з появою різних соціальних мереж. Тепер, щоб знайти людину з спільними інтересами, не потрібно виходити на вулицю і шукати, достатньо просто клікнути декілька клавіш, написавши у пошуку потрібні слова. Також досить великої популярності набирають системи побудовані на волонтерському принципі, де люди готові допомогти один одному просто так. Однією з таким мереж є CouchSurfing, що вийшла за рамки поняття «соціальна мережа» та стала спільнотою з більше, ніж 2-ома мільйонами учасників та користується популярністю по усьому світу.

Саме за таким принципом, реалізована система каучсерфінгу для домашніх тварин на основі машинного навчання. Вона допоможе людям, що мають домашніх тваринок, при необхідності знайти їм тимчасовий притулок у інших людей. Однією з особливостей системи є те, що користувачі не розподіляються чітко на окремі категорії тих, хто віddaє тваринку на деякий час, та тих, хто її приймає. У різні проміжки свого життя користувач може як і шукати допомогу, так і надавати її. Таким чином, користувачі зможуть, справді, формувати спільноту однодумців.

Оскільки, аналогів даній системі немає, а проблема досі існує, то розроблена система з великою ймовірністю знайде свою цільову аудиторію користувачів.

### **5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ**

Даний пункт передбачає оцінювання та аналіз факторів зовнішнього та внутрішнього середовища певною групою експертів.

Для оцінювання впливу зовнішніх факторів використовується шкала [-5;5]. Значення -5 – це максимальний негативний вплив, 0 – нейтральний вплив, 5 – максимальний позитивний вплив. Для оцінювання впливу внутрішніх факторів застосовується шкала [0;5], де 0 – показує нерозвинутість, катастрофічний стан

фактора або його відсутність, 5 показує високий рівень розвитку фактора. Варто зазначити, що сума вагомостей усіх факторів становить одиницю, рівень вагомості для кожного фактора визначається за допомогою коефіцієнтів. Зважений рівень впливу факторів розраховується як добуток впливу фактора у балах та рівня вагомості.

Таблиця 5.1

**Результати експертного оцінювання впливу факторів зовнішнього та внутрішнього середовищ**

Фактори	Середня експертна оцінка, бали	Середня вагомість факторів	Зважений рівень впливу, бали
<i>Фактори зовнішнього середовища</i>			
Споживачі	5	0,65	0,65
Постачальники	0	0,00	0,00
Конкуренти	-1	-0,15	-0,6
Державні органи влади	0	0,00	0,00
Інфраструктура	0	0,00	0,00
Законодавчі акти	0	0,00	0,00
Профспілки, партії та інші громадські організації	0	0,00	0,00
Система економічних відносин в державі	0	0,00	0,00
Організації-сусіди	0	0,00	0,00
Міжнародні події	0	0,00	0,00
Міжнародне оточення	3	0,09	0,00
Науково-технічний прогрес	5	0,25	0,15
Політичні обставини	2	0,06	0,00
Соціально-культурні обставини	5	0,50	0,40
Рівень техніки та технологій	4	0,20	0,25
Особливості міжнародних економічних відносин	2	0,02	0,00
Стан економіки	-4	-0,24	-0,18
<b>Загальна сума</b>		<b>1</b>	<b>1,38</b>
<i>Фактори внутрішнього середовища</i>			
Цілі	5	0,75	0,75
Структура	3	0,45	0,45
Завдання	5	0,50	0,40

Продовження табл. 5.1

Технологія	4	0,80	0,80
Працівники	4	0,60	0,45
Ресурси	2	0,50	0,50
<b>Загальна сума</b>		<b>1</b>	<b>3,60</b>

Зважений рівень впливу зовнішнього середовища є позитивним. Він становить 1,38. Позитивні фактори – споживачі, міжнародне оточення, науково-технічний процес, політичні обставини, соціально-культурні обставини, особливості міжнародних економічних відносин та рівень техніки і технологій. Негативними факторами виступають: конкуренти та стан економіки. Незабаром після реалізації системи відразу ж появляться конкуренти та намагатимуться скопіювати продукт. Варто бути до цього готовим про розробці. Стан економіки напряму впливає на платоспроможність людей, їх можливість виїжджати за кордон, тобто бути в ситуації, де їм потрібно покинути на деякий час домівку та залишити з кимось тваринку. Зважений рівень впливу факторів внутрішнього середовища становить 3,6.

Аналізуючи отримані дані варто зазначити, що сформувалась потреба у продукті, пов'язаного із надання тимчасового притулку для домашніх тварин.

#### 5.4. Формування стратегічних альтернатив

Необхідно обрати стратегічні альтернативи із першої та другої груп.

*Перша група стратегічних альтернатив* (рис.5.1).



Рис.5.1. Стратегічні альтернативи першої групи

**Стратегія розроблення нового продукту** характеризується створенням нового програмного забезпечення, яке допоможе вирішити актуальні потреби людини, суспільства, економіки тощо.

**Стратегія розвитку існуючого продукту** означає модифікацію програмного забезпечення, а також його якісних характеристик.

**Стратегія розвитку існуючого продукту з супутніми послугами** означає вихід на ринок модифікованого програмного забезпечення із додатковими послугами (встановлення, супроводження, адаптування до специфіки конкретного підприємства тощо).

**Стратегія нового продукту з супутніми послугами** означає розроблення нового програмного забезпечення та пропонування при його експлуатації додаткових послуг.

Серед даних стратегій обрано стратегію розроблення нового продукту, оскільки розроблено нову систему, аналогів якій немає на даний момент.

*Друга група стратегічних альтернатив* (рис.5.2). Критеріями поділу альтернативних стратегій розвитку є існуючий ринок та продукт, новий ринок та продукт.



*Рис.5.2. Стратегічні альтернативи другої групи*

**Глибоке проникнення на ринок** передбачає використання існуючого продукту для збільшення частки на існуючому ринку.

**Стратегія розвитку ринку** полягає в використанні існуючого продукту або незначній його модифікації для виходу на новий сегмент ринку, весь ринок або іноземний ринок.

**Стратегія розвитку продукту** полягає у створенні нового продукту для існуючого сегменту ринку.

**Стратегія диверсифікації** реалізується шляхом виходу на нові сфери бізнесу. Тобто розширення номенклатури товарів, послуг тощо.

З даної групи альтернатив обрано стратегію розвитку продукту, тому що створено новий продукт в існуючому сегменті ринку.

## 5.5. Бюджетування

На даному етапі необхідно визначити собівартість продукту, який розробляється та економічно обґрунтувати доцільність вибору однієї із стратегій. Для цього необхідно визначити, якими будуть витрати на що вони ітимуть та провести розрахунки.

Результати розрахунків подано в наступних таблицях.

*Таблиця 5.2*

### Бюджет витрат матеріалів та комплектуючих виробів

Назва матеріалів та комплектуючих	Марка, тип, модель	Фактична кількість, шт.	Ціна за одиницю, грн.	Разом, грн.
Периферійний пристрій (клавіатура)	Crown CMK-11 USB	1	200	200
Маршрутизатор	Xiaomi Mi WiFi Router 4A R4AC	1	600	600
Монітор	DELL Professional P2214Hb	1	2450	2450
<b>Разом:</b>				3250

Систему розроблятимуть двоє осіб: архітектор системи та програміст. Загальний час розробки системи складає 35 днів.

Таблиця 5.3

**Бюджет витрат на оплату праці**

<b>Посада, спеціальність</b>	<b>Кількість працівників, осіб</b>	<b>Час роботи, дні</b>	<b>Денна заробітна плата працівників, грн.</b>	<b>Сума витрат на оплату праці, грн.</b>
<i>Основна заробітна плата</i>				
Архітектор системи	1	6	600	3600
Програміст	1	29	1500	43500
<b>Разом:</b>				<b>47100</b>

На наступному етапі необхідно розрахувати єдиний соціальний внесок (ЕСВ), який складає 22% від заробітної плати.

$$\text{ЕСВ (архітектор системи)} = 3600 * 0,22 = 792 \text{ грн.}$$

$$\text{ЕСВ (програміст)} = 43500 * 0,22 = 9570 \text{ грн.}$$

В таблиці 5.4 показано розрахунок бюджету загальновиробничих витрат. Вони поділяються на змінні та постійні витрати.

Таблиця 5.4

**Бюджет загальновиробничих витрат**

<b>Статті витрат</b>	<b>Сума, грн.</b>
<i>Змінні загальновиробничі витрати, у т.ч.:</i>	
- заробітна плата допоміжного персоналу;	950
- витрати на МШП;	350
- витрати на електроенергію та технологічні цілі	900
<b>Разом змінних витрат:</b>	<b>2100</b>
<i>Постійні загальновиробничі витрати, у т.ч.:</i>	
- комунальні послуги;	2850
- витрати на оренду;	7000
- інші постійні витрати;	700
<b>Разом постійних витрат:</b>	<b>10650</b>
<b>Разом загальновиробничих витрат:</b>	<b>12750</b>

Наступним етапом є розрахунок бюджету адміністративних витрат та витрат на збут. Дані показано в таблиці 5.5.

Таблиця 5.5

**Бюджет адміністративних витрат та витрат на збут**

<b>Статті витрат</b>	<b>Сума, грн.</b>
<i>Адміністративні витрати, у т.ч.:</i>	
- заробітна плата адміністративного персоналу;	2000

*Продовження табл. 5.5*

- витрати на МШП;	400
- витрати на сплату податків і зборів;	800
Разом адміністративних витрат:	3200
<i>Витрати на збут, у т.ч.:</i>	
- витрати на рекламу;	1000
Разом витрат на збут:	1000

В таблиці 5.6 показано зведений кошторис витрат на розробку проектного рішення.

*Таблиця 5.6***Зведений кошторис витрат на розробку проектного рішення (продукту)**

Статті витрат	Одиниці виміру	Фактична кількість, шт.	Ціна одиниці, грн.	Разом, грн.
Матеріали та комплектуючі вироби	грн	-	-	3250
Основна заробітна плата	грн	-	-	47100
Додаткова заробітна плата	грн	-	-	-
Відрахування на соціальне страхування	грн	-	-	10 362
Загальновиробничі витрати, у т.ч.:				
- змінні;	грн	-	-	2100
- постійні;	грн	-	-	10650
<b>Разом виробничих витрат:</b>	грн	-	-	<b>73462</b>
Адміністративні витрати	грн	-	-	3200
Витрати на збут	грн	-	-	1000
Інші операційні витрати	грн	-	-	-
<b>Разом виробничих і операційних витрат:</b>	грн	-	-	<b>77662</b>

Наступним етапом є розрахунок фінансових результатів. Для цього необхідно розрахувати вартість продукту.

$$\Pi = СБ + СБ * P, \quad (5.1)$$

де  $\Pi$  – ціна проектного рішення (програмного продукту), грн.

$СБ$  – собівартість проектного рішення (програмного продукту), грн.

$P$  – рентабельність, 49%

$$\Pi = 73462 + 73462 * 0,49 = 109458,38 \text{ (грн)}.$$

Податок на додану вартість:

$$109458,38 * 0,2 = 21891,7 \text{ (грн)}.$$

Чистий дохід від реалізації продукції:

$$109458,38 - 21891,7 = 87566,7 \text{ (грн).}$$

Валовий прибуток:

$$87566,7 - 73462 = 14104,7 \text{ (грн).}$$

Фінансовий результат від операційної діяльності:

$$14104,7 - 4200 = 9904,7 \text{ (грн).}$$

Податок на прибуток:

$$9904,7 * 0,18 = 1782,85 \text{ (грн).}$$

Чистий прибуток (збиток):

$$9904,7 - 1782,85 = 8121,85 \text{ (грн).}$$

*Таблиця 5.7*

#### **Бюджет фінансових результатів**

<b>Показники</b>	<b>Сума, грн.</b>
Дохід від реалізації продукції	109458
Податок на додану вартість***	21892
Чистий дохід від реалізації продукції	87567
Собівартість реалізованої продукції	73462
Валовий прибуток	14105
Операційні витрати:	
- адміністративні витрати:	3200
- витрати на збут;	1000
- інші операційні витрати;	-
Фінансовий результат від операційної діяльності	9905
Податок на прибуток ****	1783
Чистий прибуток (збиток)	8122

\*\*\*ставка податку на додану вартість відповідає діючій на момент виконання випускної роботи – 20%;

\*\*\*\*ставка податку на прибуток відповідає діючій на момент виконання випускної роботи – 18%;

#### **5.6. Вибір стратегії**

Обрано стратегію розроблення нового продукту, оскільки даний програмний продукт є новий та вирішує низку проблем. Аналогів даній системі не було знайдено, а рішення, які існують не онлайн, у вигляді готелів для тварин не є популярними та

поширеними. Окрім того, готелі для тварин є досить дорогими та спричиняють стрес тваринам, особливо тим, яким потрібні поруч люди.

Проаналізувавши системи, які побудовані на схожому принципі взаємодопомоги, можна виділи, що система каучсерфінгу для домашніх тварин буде зручною, оскільки всредеині неї закладене машинне навчання, що допомогатиме користувачу зробити вигляд. Також інтерфейс буде інтуїтивно зрозумілим та легким для сприйняття. Що більш важливо – система буде безпечною. Кожен користувач має верифікуватись, а також після кожного надання тваринці житла, користувачі повинні залишити відгуки один одному, які не підлягатимуть редагуванню чи видаленню. Система також допоможе користувачам знайти людей зі спільними інтересами – домашніми улюбленацями.

### **Висновок до п'ятого розділу**

В процесі виконання даного розділу наведено економічну характеристику програмного продукту, проведено збір та обробку інформації щодо аналогічного продукту на ринку, оцінено та проаналізовано фактори внутрішнього та зовнішнього середовищ, обрано стратегічні альтернативи, проведено бюджетування та обрано стратегію. З першої групи стратегічних альтернатив обрано стратегію розроблення нового продукту, оскільки розроблено нову систему, аналогів який немає. З другої групи - стратегію розвитку продукту, тому що створено новий продукт в існуючому сегменті ринку.

При проведенні бюджетування визначено, що дохід від реалізації складе 109458 грн. Рентабельність враховувалася на рівні 49%. Враховуючи податки, чистий дохід від реалізації продукції складе 87567 грн. Беручи до уваги всі видатки, чистий прибуток складе 8122 грн.

## ВИСНОВКИ

Результатом виконання дипломної роботи є реалізована web-система каучсерфінгу для домашніх тварин на основі машинного навчання. Здійснено аналіз предметної області, системний аналіз, а також аналіз технологій необхідних для створення системи. Реалізовано систему у вигляді веб-сайту, задеплоєно її на безкоштовний хостинг та протестовано програмний продукт на коректність роботи. Також проведено економічні підрахунки для підтвердження рентабельності створення системи.

В процесі виконання первого етапу проведено поглиблений аналіз літературних та інтернет-джерел, спираючись не лише на доцільність створювати систему, але й на психологічну та соціальну складову людини, що керуватимуть нею при користуванні системою. Також проаналізовано схожі рішення, адже немає існуючих рішень даної проблеми. Виділено основні переваги та недоліки схожих рішень та на їх основі виділено аспекти, необхідні для власної системи. Серед таких аспектів є: створення зручного та зрозумілого інтерфейсу для користувача, реєстрація в системі безоплатна, додавання машинного навчання, що полегшить користувачеві роботу з сайтом, стійка система відгуків для безпеки тваринок.

На другому етапі спроектовано архітектуру інформаційної web-системи. Проведено системний аналіз, який включав в себе побудову дерева цілей та виділення необхідних критеріїв системи. За допомогою аналізу ієархій визначено тип системи – інтелектуально-пошукова система. Також побудовано контекстну діаграму та її декомпозицію на процеси. Після чого продемонстровано ієархію задач, що знадобиться для правильної реалізації системи.

На третьому етапі виконання дипломної роботи проаналізовано технічні засоби, які необхідні при реалізації системи. Для написання інтерфейсу обрано такі технології: html, css, javascript, react, redux, typescript, react styled components, react hook forms, material-ui, axios, yup. Для написання серверної частини було використано node.js, express, posgreSQL, sequelize, jwt, joi. Також було обрано середовище розробки Visual Studio Code. Окрім цього, проаналізовано методи машинного

навчання. Обрано метод, який найкраще підходить для спроектованої системи, а саме метод з використанням нейронних мереж і глибинного навчання. Проаналізовано бібліотеки машинного навчання. Визначено їх основні переваги та недоліки та обрано tensorflow.js для реалізації машинного навчання в системі.

В процесі виконання четвертого етапу проаналізовано реалізований програмний продукт та визначено, що він виконує усі поставлені задачі та досягає поставленої мети, а саме надання тимчасового притулку домашнім тваринам. Описано загальні відомості про програмний продукт, візуалізовано та описано структуру бази даних, функціонал системи, виклик та завантаження, вхідні та вихідні даних. Також детально описано інструкцію для користувача, класи вирішуваних задач та необхідні технічні пристрой для коректної роботи веб-сайту.

В економічній частині детально обґрунтовано доцільність розробки даного програмного продукту, а також розраховано економічні характеристики. В процесі виконання даного розділу наведено економічну характеристику програмного продукту, проведено збір та обробку інформації щодо аналогічного продукту на ринку, оцінено та проаналізовано фактори внутрішнього та зовнішнього середовищ, обрано стратегічні альтернативи, проведено бюджетування та обрано стратегію. Також обрано рівень рентабельності та розраховано чистий прибуток в результаті реалізації, який складає 8122 грн.

## СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Особливості веб-орієнтованих інформаційних систем [Електронний ресурс]. – Режим доступу до ресурсу: <https://studfile.net/preview/5535715/page:3/> (дата звернення 10.02.2020).
2. Бойко Н.І. Моделювання Web-орієнтованих систем та напрямки розвитку Web-ресурсів / Н.І. Бойко // Вісник Національного університету "Львівська політехніка". – Львів, 2012. – № 743 : Інформаційні системи та мережі. – С. 16–25. (дата звернення 10.02.2020).
3. Актуальныи список лоукост-авиакомпаний [Електронний ресурс]. – Режим доступу до ресурсу: <https://samokatus.ru/lowcost-airlines/> (дата звернення 10.02.2020).
4. Komarova A.A. Social network for tourism «Couchsurfing» as a social phenomenon / Komarova A.A., Kryshtanovskaya O.V.// Vestnik Universiteta. -2019.-Vol. (3).-P.173-176. (дата звернення 11.02.2020).
5. Що таке каучсерфінг та як користуватись сервісом: 7 порад мандрівнику [Електронний ресурс]. – Режим доступу до ресурсу: <http://1001idea.info/scho-take-kauchserfinh-7-vidpovidej-ta-porad-mandrivnyku/> (дата звернення 11.02.2020).
6. CouchSurfing [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.life-in-travels.ru/couchsurfing/> (дата звернення 12.02.2020).
7. Безопасность в CouchSurfing [Електронний ресурс]. – Режим доступу до ресурсу: <http://litetrip.ru/bezopasnost-v-couchsurfing-kouchserfinge-lichnye-rekomendacii.html> (дата звернення 13.02.2020).
8. Психология владельца домашнего животного [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.svoboda.org/a/377148.html> (дата звернення 13.02.2020).
9. Чому люди заводять домашніх тварин? [Електронний ресурс]. – Режим доступу до ресурсу: <http://animalcity.com.ua/> (дата звернення 14.02.2020).
10. Господарю, а як же я?», або куди подіти улюблена на час відпустки [Електронний ресурс]. – Режим доступу до ресурсу:

[https://www.bbc.com/ukrainian/entertainment/2013/06/130614\\_pets\\_vacation\\_ag](https://www.bbc.com/ukrainian/entertainment/2013/06/130614_pets_vacation_ag) (дата звернення 14.02.2020).

11. Фриланс и домашние животные: друзья или якорь к дому? [Електронний ресурс]. – Режим доступу до ресурсу: <https://freelancehunt.com/blog/frilans-i-domashniie-zhivotnyie-druzia-ili-iakor-k-domu> (дата звернення 15.02.2020).

12. Гостиница для животных: плюсы и минусы [Електронний ресурс]. – Режим доступу до ресурсу: <https://wiki.dog/gostinica-dlya-zhivotnyx-plyusy-i-minusy> (дата звернення 16.02.2020).

13. Беседы о поведении животных [Електронний ресурс]. – Режим доступу до ресурсу: <https://zoopsychologist.livejournal.com/98280.html#comments> (дата звернення 17.02.2020).

14. Акулич М.М. Каучёрфинг как социальная практика / Акулич М.М., Батырева М.В., Голованова Ю.И. // Вестник Российского университета дружбы народов. – Москва, 2017. – Серия: Социология. – №4. – С. 568-577. (дата звернення 18.02.2020).

15. What is social practice [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.igi-global.com/dictionary/boundaryless-work-role-mobile-ict/27461> (дата звернення 18.02.2020).

16. Балич Н.Л., Социальные практики и их роль в современном обществе / Балич Н.Л. // Социология: теория, методы, маркетинг. — 2013. — № 4. — С. 69–78. — Бібліогр.: 8 назв. — рос. (дата звернення 18.02.2020).

17. Питер Бергер Социальное конструирование реальности. Трактат по социологии знания / Питер Бергер, Томас Лукман — М., 1995. — 289 с. (дата звернення 19.02.2020).

18. Гирц К. Интерпретация культур / Гирц К. — М., 2004. — 560 с. (дата звернення 20.02.2020).

19. Шюц А Мир, светящийся смыслом/ Шюц А. — М., 2004. — 1056 с. (дата звернення 20.02.2020).

20. Бурдье П. Социология социального пространства / Бурдье П. — СПб., 2007. — 288 с. (дата звернення 21.02.2020).
21. Garfinkel H. Studies in ethnomethodology / Garfinkel H. — NJ: Prentice-Hall, 1967. — P. 285. (дата звернення 21.02.2020).
22. Гіденс Е. Соціологія / Гіденс Е. — К., 1999. — 302 с. (дата звернення 22.02.2020).
23. Тернер Дж. Структура социологической теории / Тернер Дж. . — М., 1985. — 433 с. (дата звернення 22.02.2020).
24. Кнорр-Цетина К. Теоритические проблемы социологии / Кнорр-Цетина К. // Журнал социологии и социальной антропологии. — 1997. — Vol. (14). — №4. — Р. 130. (дата звернення 23.02.2020).
25. Кропоткин П.А. Взаимопомощь как фактор эволюции / Кропоткин П.А. — М.: Самообразование, 2007. — 256 с. (дата звернення 24.02.2020).
26. Do things for others [Електронний ресурс]. — Режим доступу до ресурсу: <https://www.actionforhappiness.org/10-keys-to-happier-living/do-things-for-others/details> (дата звернення 24.02.2020);
27. 10 Ways To Help Others That Will Lead You To Success [Електронний ресурс]. — Режим доступу до ресурсу: <https://www.forbes.com/sites/johnhall/2013/05/26/10-ways-to-help-others-that-will-lead-you-to-success/#4404da022bce> (дата звернення 24.02.2020).
28. Couchsurfing.com [Електронний ресурс]. — Режим доступу до ресурсу: <https://www.couchsurfing.com/> (дата звернення 25.02.2020).
29. Community Post: How to be a Good Couchsurfing Guest and Host. [Електронний ресурс]. — Режим доступу до ресурсу: <https://blog.couchsurfing.com/community-post-how-to-be-a-good-couchsurfing-guest-and-host/> (дата звернення 25.02.2020);
30. Servas.org [Електронний ресурс]. — Режим доступу до ресурсу: <https://www.servas.org/> (дата звернення 25.02.2020).

31. Servas [Електронний ресурс]. – Режим доступу до ресурсу: [https://www.huffpost.com/entry/servas-an-open-door-to-travel\\_b\\_6713902](https://www.huffpost.com/entry/servas-an-open-door-to-travel_b_6713902) (дата звернення 25.02.2020).
32. BeWelcome.org [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.bewelcome.org/> (дата звернення 25.02.2020).
33. CouchSurfing vs BeWelcome [Електронний ресурс]. – Режим доступу до ресурсу: [https://www.reddit.com/r/couchsurfing/comments/1hwvk7/couchsurfing\\_vs\\_bewelcome/](https://www.reddit.com/r/couchsurfing/comments/1hwvk7/couchsurfing_vs_bewelcome/) (дата звернення 25.02.2020);
34. Системний аналіз як метод ухвалення і обґрунтування рішень [Електронний ресурс]. – Режим доступу до ресурсу: [https://pidruchniki.com/1678102440718/buhgalterskiy\\_oblik\\_ta\\_audit/sistemniy\\_analiz\\_metod\\_uhvalenya\\_obgruntuvannya\\_rishen](https://pidruchniki.com/1678102440718/buhgalterskiy_oblik_ta_audit/sistemniy_analiz_metod_uhvalenya_obgruntuvannya_rishen) (дата звернення 29.02.2020).
35. Веб-документація MDN HTML [Електронний ресурс]. Режим доступу: <https://developer.mozilla.org/uk/docs/Web/HTML> (дата звернення 05.03.2020).
36. Веб-документація MDN CSS [Електронний ресурс]. Режим доступу: <https://developer.mozilla.org/uk/docs/Web/CSS> (дата звернення 05.03.2020).
37. Веб-документація MDN JavaScript [Електронний ресурс]. Режим доступу: <https://developer.mozilla.org/uk/docs/Web/JavaScript> (дата звернення 06.03.2020).
38. axios/axios [Електронний ресурс]. Режим доступу: <https://github.com/axios/axios> (дата звернення 08.03.2020).
39. React – JavaScript-бібліотека для створення користувачьких інтерфейсів [Електронний ресурс]. Режим доступу: <https://uk.reactjs.org/> (дата звернення 11.03.2020).
40. YUP [Електронний ресурс]. Режим доступу: <https://www.npmjs.com/package/yup> (дата звернення 11.03.2020).
41. Introduction to Node.js [Електронний ресурс]. Режим доступу: <https://nodejs.dev/> (дата звернення 11.03.2020).

42. Express - фреймворк веб-приложений Node.js [Електронний ресурс].

Режим доступу: <https://expressjs.com/ru/> (дата звернення 11.03.2020).

43. JWT.IO - JSON Web Tokens Introduction [Електронний ресурс]. Режим доступу: <https://jwt.io/introduction/> (дата звернення 12.03.2020).

44. Visual Studio Code - Code Editing. Redefined [Електронний ресурс]. Режим доступу: <https://code.visualstudio.com/> (дата звернення 12.03.2020).

45. What is machine learning? Everything you need to know [Електронний ресурс]. Режим доступу: <https://www.zdnet.com/article/what-is-machine-learning-everything-you-need-to-know/> (дата звернення 12.03.2020).

46. Introducing Deep Learning and Neural Networks — Deep Learning for Rookies [Електронний ресурс]. Режим доступу: <https://towardsdatascience.com/introducing-deep-learning-and-neural-networks-deep-learning-for-rookies-1-bd68f9cf5883/> (дата звернення 12.03.2020).

47. How to Apply Machine Learning to Business Problems [Електронний ресурс]. Режим доступу: <https://emerj.com/ai-executive-guides/how-to-apply-machine-learning-to-business-problems/> (дата звернення 12.03.2020).

48. Методичні вказівки до виконання дипломних робіт для студентів напряму підготовки 6.050101 “Комп’ютерні науки” / В. В. Литвин, Є. В. Буров, Т. М. Басюк, О. М. Верес, А. В. Катренко, П. О. Кравець. – Львів: Видавництво Львівської політехніки, 2017. – 32 с.

49. Методичні вказівки до виконання бакалаврської роботи з курсу «Технології створення програмних продуктів» /Басюк Т. М. – Львів: НУ «ЛП», 2017. – 41 с.

## ABSTRACT

In today's world there are many problems that need to be considered. Moreover, since the 21st century, people have been trying to achieve a reduction in their worries on the Internet. One such problem is that pet owners do not know where and with whom to leave their pets in the event of a sudden business trip or a desire to go on vacation. That is why it was decided to create a web-system of couchsurfing for pets based on machine learning.

This system allows you to find temporary shelter for your pet for free. It should be added that today there is another solution to this problem - the so-called "hotels for animals". Mostly, these are institutions at veterinary clinics, where for a certain price you can leave your pet for retention. However, there are very few such "hotels for animals", and the price of overexposure is quite expensive. Moreover, such conditions for animals can be extremely stressful.

The purpose of this system is to help to find temporary shelter for pet if necessary. Also, people who have no one to leave their pet with are usually single people. The system will also help them find people with common interests. Another goal of the study is to ensure that people who travel frequently are not afraid to have pets, as they will have a service that will help them find temporary shelter for the pet if necessary.

This work consists of five sections. The first section provides an in-depth analysis of literary and Internet sources, based not only on the feasibility of creating a system, but also on the psychological and social component of the person who will manage it when using the system. Similar solutions are also analyzed, as there are no existing solutions to this problem. In the second section, the architecture of the information web-system is designed. A systematic analysis was performed, namely a tree of goals, business process diagrams and task hierarchy was designed. At the third stage of the thesis, the technical means necessary for the implementation of the system are analyzed. In the process of performing the fourth stage, the implemented software product was analyzed and it was determined that it fulfills all the tasks and achieves the goal, namely the provision of temporary shelter for pets. In the fifth part the expediency of development of the given software product is substantiated in detail, and also economic characteristics are calculated.

The result of this work is a implemented system of couchsurfing for pets based on machine learning, which will help people if necessary to leave their pet with someone for a temporary period.

## ДОДАТОК А

### **app.ts**

```
import "reflect-metadata";

import config from "./config";
import express from "express";
import io from "socket.io";
import Logger from "./loaders/logger";
import "./types/global";

async function startServer() {
  const app = express();
  await require("./loaders").default({ expressApp: app });
  const server = app.listen(config.port, () =>
    Logger.info(`🛡️ Server listening on port: ${config.port} 🛡️`)
  );
  io.listen(server);
}

startServer();
```

### **HomeGuest.tsx:**

```
import React from 'react';
import { useSelector } from 'react-redux';

import { EmojiSpan, Text, FloatBlockDiv } from '../../components';
import { GuestLayoutWrapper } from '../../components/Layout';
import { SButton } from '../../components/FormControls';
import { RegistrationForm } from './RegistrationForm';
import { LeftMessageDiv, RightMessageDiv } from './Styled';

const HomeGuest = () => {
  const registrationSuccess: any = useSelector<any>(
    (state) => state.system.registrationSuccess
  );

  return (
    <GuestLayoutWrapper>
      <LeftMessageDiv>
        <FloatBlockDiv>
          <EmojiSpan label="Cat" emoji="🐱" rotate="-15deg" fontSize="85px" />
        </FloatBlockDiv>
        <div>
          <Text weight="bold" variant="h6">
            Збираєшся у подорож і не знаєш з ким залишити домашнього улюбленаця?
          </Text>
          <Text variant="subtitle1" separator padding="0 10px 0 0">
            Тут знайдеться багато хороших людей, які зможуть тобі допомогти.
          </Text>
    </LeftMessageDiv>
  );
}
```

```

<SButton padding="5px 40px">Як це працює?</SButton>
</div>
</LeftMessageDiv>
<RightMessageDiv>
{registrationSuccess ? (
<Text weight="bold" variant="h6">
    Реєстрація успішна
</Text>
) : (
<React.Fragment>
<Text weight="bold" variant="h6">
    Створіть акаунт
</Text>
<RegistrationForm />
</React.Fragment>
)}
</RightMessageDiv>
</GuestLayoutWrapper>
);
};
};

export default HomeGuest;

```

### **RegistrationForm.tsx:**

```

import React from 'react';
import { useForm, Controller } from 'react-hook-form';
import { useDispatch, useSelector } from 'react-redux';
import * as yup from 'yup';

import { StyledForm } from '../../components';
import {
    FormInput,
    DefaultButton,
    FormRadio,
    FormDatePicker,
    AutocompleteInput,
} from '../../components/FormControls';
import { validationConst } from '../../constants';
import { registration as registerAction } from '../../store/system';
import { userMaleConst } from '../../constants';

const registrationValidationSchema = yup.object().shape({
    firstName: yup.string().required(validationConst.ERRORS.firstName).min(2),
    lastName: yup.string().required(validationConst.ERRORS.lastName).min(2),
    birthDate: yup.date().required(validationConst.ERRORS.required),
    city: yup.string().required(validationConst.ERRORS.required),
    gender: yup.string().required(validationConst.ERRORS.required),
    email: yup.string().email().required(validationConst.ERRORS.required),
    password: yup.string().required(validationConst.ERRORS.required).min(5),
});

```

```

export function RegistrationForm() {
  const dispatch = useDispatch();

  const { handleSubmit, errors, control, register } = useForm({
    validationSchema: registrationValidationSchema,
  });

  const onSubmit = (data: any): any => dispatch(registerAction(data));
  const cities = useSelector<any>((state) => state.system.cities);

  return (
    <StyledForm onSubmit={handleSubmit(onSubmit)}>
      <FormInput
        name="firstName"
        label="Ім'я"
        inputRef={register}
        placeholder="Ім'я"
        errors={errors}
      />
      <FormInput
        name="lastName"
        label="Прізвище"
        inputRef={register}
        placeholder="Прізвище"
        errors={errors}
      />
      <Controller
        as={FormDatePicker}
        label="День народження"
        name="birthDate"
        placeholder="День народження"
        maxDate={new Date()}
        control={control}
        defaultValue={null}
      />
      <AutocompleteInput
        name="city"
        label="Місто"
        inputRef={register}
        placeholder="Місто"
        options={cities}
        errors={errors}
      />
      <Controller
        as={FormRadio}
        values={userMaleConst}
        label="Стать"
        control={control}
        name="gender"
      />
    
```

```

defaultValue=""
direction="row"
errors={errors}
/>
<FormInput
name="email"
label="Емейл"
inputRef={register}
placeholder="Емейл"
errors={errors}
/>
<FormInput
name="password"
label="Пароль"
type="password"
inputRef={register}
placeholder="Пароль"
errors={errors}
/>
<DefaultButton type="submit">Зареєструватися</DefaultButton>
</StyledForm>
);
}

```

### **UserProfilePage.tsx:**

```

import React, { useState, useEffect } from 'react';
import { RouteComponentProps } from 'react-router-dom';
import { useSelector, useDispatch } from 'react-redux';

import { UserLayoutWrapper } from '../../components/Layout';
import {
  Text,
  StyledTabs,
  StyledTab,
  Avatar,
  StyledBadge,
} from '../../components';
import { SButton } from '../../components/FormControls';
import {
  getUserProfile,
  clearUserProfile,
} from '../../store/users/users.actions';
import { getFeedbacks } from '../../store/requests/requests.action';
import {
  BlockWrapper,
  UserBlockDiv,
  ActionButtons,
  DetailsBlockDiv,
  UserOnline,
  UserSkeleton,
}

```

```

} from './Styled';
import { isCurrentUser, getUserName, lastLogin } from '../..../helpers';
import { MakeRequestModal } from '../../containers/Modals';

import {
  MyPetsTab,
  AboutMeTab,
  MyHomeTab,
  ReferenceTab,
  PhotosTab,
} from './tabs';

const UserProfilePage: React.FC<RouteComponentProps<{ id: string }>> = ({ match, }) => {
  const dispatch = useDispatch();
  const [tab, setTab] = useState('About');
  const [requestModal, setRequestModal] = useState(false);

  const userId = Number(match.params.id);

  useEffect(() => {
    if (userId) {
      dispatch(getUserProfile(userId));
      dispatch(getFeedbacks(userId));
      setTab('About');
    }
  }, []);

  return () => {
    dispatch(clearUserProfile());
  };
}, [userId, dispatch]);

const data: any = useSelector<any>((state) => state.system.user);
const user: any = useSelector<any>((state) => state.users.data);

const isCurrent = userId && data && isCurrentUser(Number(userId), data.id);

const tabs = [
  { tab: 'About', label: 'Про мене', count: null },
  {
    tab: 'My Pets',
    label: 'Мої улюбленці',
    count: useSelector<any>((state) => state.users.pets.length),
  },
  { tab: 'My Home', label: 'Мій дім', count: null },
  {
    tab: 'Photos',
    label: 'Фото',
    count: useSelector<any>((state) => state.users.photos.length),
  }
];

```

```

    },
    {
      tab: 'References',
      label: 'Відгуки',
      count: useSelector<any>((state) => state.requests.feedbacks.count),
    },
  ];
}

const openRequestModal = () => setRequestModal(true);
const closeRequestModal = () => setRequestModal(false);

const userOnline = user && lastLogin(user.lastOnline);

return (
  <UserLayoutWrapper>
    <BlockWrapper>
      <UserBlockDiv>
        {!user ? (
          <UserSkeleton />
        ) : (
          <>
            {' '}
            <Avatar
              data={user}
              size="200px"
              fontSize="50px"
              userStatus={true}
            />
            <Text padding="20px 0 0 0">{getUserName(user)}</Text>
            <Text variant="subtitle2">Місто: {user.city}</Text>
            <UserOnline>
              {user.lastOnline && userOnline === 'Online' ? (
                <>
                  Статус: <span>Онлайн</span>
                </>
              ) : (
                Онлайн: ${userOnline}
              )}
            </UserOnline>
            <Text variant="subtitle2">Доступність: {user.availability}</Text>
            <Text variant="subtitle2">
              {user.verified
                ? 'Акаунт верифікований'
                : 'Акаунт НЕ верифікований'}
            </Text>
            {isCurrent && (
              <ActionButtons>
                <SButton
                  width="100%"
                  margin="5px 0 0 0"
                >

```

```

    onClick={openRequestModal}
  >
  Створити запит
</SButton>
</ActionButtons>
)}
</>
)}
</UserBlockDiv>
</BlockWrapper>

<BlockWrapper width="100%">
<DetailsBlockDiv>
<StyledTabs value={tab} onChange={(e, value) => setTab(value)}>
  {tabs.map(({ tab, count, label }) => (
    <StyledTab
      key={tab}
      value={tab}
      label={<StyledBadge badgeContent={count} right="-14px" top="12px">
        {label}
      </StyledBadge>
    }
    />
  )))
</StyledTabs>
{tab === 'About' && <AboutMeTab user={user} />}
{tab === 'My Pets' && <MyPetsTab isCurrent={isCurrent} />}
{tab === 'My Home' && <MyHomeTab />}
{tab === 'Photos' && <PhotosTab isCurrent={isCurrent} />}
{tab === 'References' && <ReferenceTab userId={userId} />}
</DetailsBlockDiv>
</BlockWrapper>
{user && user.id && (
  <MakeRequestModal
    open={requestModal}
    currentUser={data}
    responder={user}
    onClose={closeRequestModal}
  />
)
})
</UserLayoutWrapper>
);
};

export default UserProfilePage;

```