Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА

## Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Мобільний застосунок для аналізу і контролю за станом здоров'я собак

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-1
Дмитро ІЛЬІН
(власне ім’я, прізвище)

Спеціальність
123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: ас. Юлія АНДРУСЕНКО
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

# ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ільіну Дмитру Максимовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Мобільний застосунок для аналізу і контролю за станом здоров’я собак

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи 1. План кваліфікаційної роботи
2. Інтегроване середовище розробки PyCharm та DataSpell
3. Мова програмування Python
4. Згорткова нейронна мережа ResNet18

4. Перелік питань, що потрібно опрацювати у роботі
1. Планування реалізації поставлених задач кваліфікаційної роботи
2. Аналіз та вибір програмних засобів реалізації кваліфікаційної роботи
3. Реалізація кваліфікаційної роботи
4. Тестування результатів кваліфікаційної роботи
5. Інструкція користувача

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 14 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1 )

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |
| | | | |

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Строк / терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Дослідження галузі | 27.05.2025-28.05.2025 | |
| 2 | Аналіз існуючих методів реалізації задачі | 29.05.2025-30.05.2025 | |
| | кваліфікаційної роботи | | |
| 3 | Створення та навчання нейронної мережі | 1.06.2025-2.06.2025 | |
| 4 | Створення мобільного застосунку | 3.06.2025-6.05.2025 | |
| 5 | Тестування мобільного застосунку | 7.05.2025-8.06.2025 | |
| 6 | Аналіз отриманих результатів | 9.06.2025-10.06.2025 | |
| 7 | Оформлення матеріалів кваліфікаційної роботи | 11.06.2025-12.06.2025 | |
| 8 | Подання кваліфікаційної роботи керівникові та | 13.06.2025-14.06.2025 | |
| | її попередній захист | | |
| 9 | Подання кваліфікаційної роботи на рецензу- | 15.06.2025-16.06.2025 | |
| | вання | | |

Дата видачі завдання " 26 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Юлія АНДРУСЕНКО
(підпис) (посада, власне ім'я, прізвище)

# РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 67 с., 21 рис., 1 дод., 29 джерел.

GOOGLE CLOUD PLATFORM, ІНТЕРНЕТ, НЕЙРОННА МЕРЕЖА, МОБІЛЬНИЙ ЗАСТОСУНОК, НАБІР ДАНИХ, МАШИННЕ НАВЧАННЯ, PYTHON, JUPYTER

Метою кваліфікаційної роботи є розробка мобільного застосунку для контролю та аналізу стану здоров'я собак. Застосунок має бути з простим інтерфейсом, підтримкою декількох домашніх тварин, функціями аналізу характеристик тварини, порад з догляду та системи сповіщень та нагадувань.

У ході виконання кваліфікаційної роботи має бути створено мобільний застосунок на основі мови програмування Python. Використовуючи відповідний набір даних навчено модель нейронної мережі розпізнавати собаку та ідентифікувати її. Використовуючи різноманітні бібліотеки Conda та Python реалізувати повноцінний програмний застосунок для виконання поставлених задач.

Мобільний застосунок має створювати декілька профілів тварин, з можливістю їх редагування. Створювати нотатки та сповіщення в календарі. Шукати ветеринара за адресою. Пропонувати поради за доглядом за собакою, щоб вона залишалась здорова.

# ABSTRACT

Bachelor’s thesis: 67 pages, 21 figures, 1 appendices, 29 sources.

GOOGLE CLOUD PLATFORM, INTERNET, NEURAL NETWORK, MOBILE APPLICATION, DATASET, MACHINE LEARNING, PYTHON, JUPYTER.

The major goal of this thesis is to develop a mobile application for monitoring and analyzing the health status of dogs. The application should have a simple interface, support for multiple pets, functions for analyzing animal characteristics, care tips, and a system of notifications and reminders.

In order to this thesis, a mobile application based on the Python programming language was created. Using the appropriate dataset, a neural network model was trained to recognize and identify a dog. Using a variety of Conda and Python libraries, a full-fledged software application was implemented to accomplish the tasks.

The mobile application can create multiple animal profiles, with the ability to edit them. Create notes and calendar notifications. Search for a veterinarian by address. Offer advice on dog care to keep your dog healthy.

ЗМІСТ

# СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Бекенд – це взаємодія користувача з внутрішніми даними, які потім відображає фронтенд

Токенізація – це процес, під час якого замість використання реальних даних, використовується унікальний цифровий ідентифікатор

Фронтенд – це створення видимої частини веб-сайту або програми, з якою взаємодіє користувач

API – інтерфейс програмування програм, що дозволяє програмам взаємодіяти між собою (англ., Application Programming Interface)

CSV – формат файлів для зберігання табличних даних, де значення розділені комами (англ., Comma-Separated Values)

GPS – глобальна система позиціонування, використовувана визначення місцезнаходження Землі (англ., Global Positioning System)

IDE – інтегроване середовище розробки, що містить інструменти для написання, компіляції та налагодження коду (англ., Integrated Development Environment)

XML – розширювана мова розмітки, яка використовується для структурованого зберігання та передачі даних (англ., Extensible Markup Language)

# ВСТУП

Домашні улюбленці – це невід’ємна частина багатьох сімей по всьому світу. За найбільш актуальними даними Euromonitor за 2024 рік [1], на території України проживають близько 4.55 мільйона домашніх собак, недарма кажуть, що собака надійний друг людини. Цих хатніх тварин обходять за популярністю лише коти, за даними того самого джерела їх кількість становить 7.14 мільйона. Разом вони складають третину усіх домашніх улюбленців по всій країні.

Метою кваліфікаціної роботи є розробка мобільного застосунку для контролю та аналізу стану здоров'я собак. Застосунок має бути з простим інтерфейсом, підтримкою декількох домашніх тварин, функціями аналізу характеристик тварини, порад з догляду та системи сповіщень та нагадувань.

Завдання кваліфікаційної роботи полягає у створенні мобільного застосунку на основі мови програмування Python. Використовуючи відповідний набір даних навчити модель нейронної мережі розпізнавати собаку та ідентифікувати її. Використовуючи різноманітні бібліотеки Conda та Python реалізувати повноцінний програмний застосунок.

Використані середовища розробки: PyCharm, Google Cloud Platform, DataSpell і Excel.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Актуальність застосунків для собак

У сучасному цифровому середовищі програми для догляду за собаками стали важливим інструментом для відповідального утримання домашніх улюбленців. Вони охоплюють широкий спектр функцій: планування щоденних рутин (годування, вигул, ігри), ведення медичних записів (вакцинації, прийом ліків, візити до ветеринара), відстеження фізичної активності та самопочуття, а також навчання та дресирування. Деякі застосунки інтегруються з фітнес-трекерами або розумними нашийниками, що дає змогу отримувати ще більше точних даних про стан здоров’я тварини в режимі реального часу, так само як і в людей. Таким чином, програмні засоби стають невід'ємною частиною системного підходу до утримання собак, полегшуючи повсякденні обов’язки власника та покращуючи якість життя тварини.

Багато важливої та корисної інформації просто зберігається в інтернеті. У соціальних мережах, популярних веб-ресурсах формату запитання-відповідь, популярні короткі відео типу TikTok, Shorts. Досвідчені кінологи та власники собак можуть дати багато корисних порад до актуальних запитань новачків у цій сфері доступно та швидко.

Разом із цим, цифрові платформи створюють новий рівень спільноти для власників собак. Через мобільні застосунки користувачі можуть обмінюватися порадами, шукати локальних ветеринарів, записуватись на грумінг або тренування, а також організовувати зустрічі з іншими власниками. Така інтеграція зміцнює соціальну відповідальність та сприяє формуванню культури свідомого утримання тварин.

Крім того, використання застосунків дозволяє оперативно реагувати на позаштатні ситуації. Наприклад, деякі програми мають функції пошуку

зниклих тварин, бази даних з фото собак у регіоні або контакти притулків. Це значно підвищує шанси знайти улюбленця у разі його зникнення, а також допомагає у вирішенні проблем безпритульності тварин у масштабах громади.

## 1.2 Аналіз існуючих програмних засобів

Серед великої кількості програм для догляду за собаками спостерігається значна різноманітність як за функціональністю, так і за спрямованістю. Наприклад, програми PetDesk чи DogLog орієнтовані на комплексний щоденний догляд і медичний моніторинг, тоді як Dogo спеціалізується на дресируванні, а Whistle – на GPS-відстеженні та аналізі активності. Це дозволяє користувачам обирати рішення, яке найкраще відповідає потребам саме їхнього улюбленця.

Основними перевагами цих програм є інтуїтивний інтерфейс, доступ до аналітики в реальному часі, можливість налаштування персональних нагадувань, інтеграція з пристроями для спостереження і хмарними сервісами. Також існують і певні недоліки: більшість програм мають обмежений функціонал у безкоштовних версіях, потребують стабільного підключення до інтернету, а також не завжди підтримують локалізацію українською мовою.

Окрему увагу заслуговує сегментація програмних засобів за цільовими групами користувачів. Зокрема, професійно орієнтовані застосунки пропонують розширені можливості для ведення бази клієнтів, документування тренувальних сесій та аналітики поведінки тварин, тоді як продукти, орієнтовані на власників-початківців, здебільшого забезпечують доступ до базових інструкцій, порад та відеоматеріалів. Така адаптація інтерфейсів і функціоналу під різні рівні досвіду користувача позитивно впливає на зручність використання та ефективність програм.

Перспективним напрямом розвитку зазначених програм є інтеграція

багатофункціональних рішень, що поєднують можливості моніторингу здоров’я, навчання, соціальної взаємодії та поведінкової корекції. Актуальними також залишаються питання впровадження технологій штучного інтелекту та машинного навчання, що дозволяють підвищити точність прогнозування стану тварини, автоматизувати рутинні процеси та забезпечити більш гнучку адаптацію програмного забезпечення до змін у поведінці або фізичному стані собаки.

### 1.2.1 PetDesk

PetDesk – це мобільний застосунок, створений для власників домашніх тварин, які хочуть зручно керувати усіма аспектами догляду за своїми улюбленцями [2]. Його головна мета – це спрощення комунікації з ветеринарами, сповіщувати про важливі події та забезпечити повний контроль над здоров’ям тварини.

Застосунок дозволяє зручно планувати візити до ветеринара, грумінг, вакцинації або інші процедури. Користувач отримує нагадування про заплановані події. Через PetDesk можна напряму зв’язатися з клінікою, надіслати повідомлення, записатись на прийом або замовити повторний рецепт. Усі медичні записи тварини – вакцинації, процедури і призначення ліків зберігаються в одному місці, доступному у будь-який момент.

Кожна тварина має окремий профіль, де зберігаються дані про її породу, вік, вагу, хронічні захворювання та фото. Це особливо зручно, якщо у вас кілька тварин. Мобільний застосунок також дозволяє налаштувати графік прийому ліків, вказати дозування та отримувати сповіщення, коли настав час їх дати.

Для користувачів, чия клініка підтримує цю функцію, доступна програма лояльності з бонусами та знижками за візити або використання послуг. Сама ж програма функціонує на основі підписки.

### 1.2.2 DogLog

DogLog – це мобільний застосунок [3], який допомагає власникам собак вести облік повсякденного догляду та координувати дії всієї родини або доглядачів. Головна ідея уникнути плутанини у щоденних завданнях і не забути нічого важливого.

У програмі можна фіксувати всі дії, пов’язані з доглядом за собакою: вигул, годування, тренування, прийом ліків, купання, чищення вух, візити до ветеринара. Кожна дія зберігається з точною датою і часом, тому завжди видно, що вже зроблено, а що ще ні.

Особливо корисна функція – це спільний доступ. Усі члени родини або доглядачі можуть бачити, коли останній раз вигулювали чи годували собаку. Це зручно й для власників, які часто залишають улюбленця з друзями або вигулювачами собак, усі мають одну спільну систему.

Також можна додавати нотатки, фото, і ставити персоналізовані нагадування, наприклад, про щеплення, візит до грумера чи зміну корму. Вся історія активностей собаки зберігається в одному місці, тому легко простежити звички, виявити зміни в поведінці або підготуватись до консультації з ветеринаром. Гарний вибір для тих, хто може залишити собаку на інших людей.

### 1.2.3 Dogo

Dogo – це мобільний тренер для собак [4], який допомагає власникам навчати своїх улюбленців слухняності, трюкам та хорошій поведінці. Програма орієнтована на позитивне підкріплення й підходить як для цуценят, так і для дорослих собак.

У застосунку доступно понад 100 тренувальних вправ і команд, які розбиті за рівнями складності. Інструкції подаються у форматі відео з чіткими покроковими поясненнями. Користувач вибирає програму,

наприклад «Навчання цуценяти», «Вихованість на прогулянці» чи «Веселі трюки», і проходить її разом із собакою день за днем.

Інтерактивний клікер одна з основних функцій програми. Його можна використовувати як інструмент для закріплення правильних дій. Застосунок також дозволяє фіксувати успіхи, ставити цілі, отримувати нагадування про тренування й навіть надсилати відео виконання вправ сертифікованим тренерам для зворотного зв'язку.

Dogo підтримує мотивацію як у собаки, так і у власника: кожен успішний крок приносить віртуальні досягнення, а чітка структура програми допомагає не зійти з дистанції. Це гарний варіант для тих, хто хоче дресирувати собаку самостійно, але з професійною підтримкою кінологів.

### 1.2.4 Wristle

Whistle – це не просто GPS-трекер, а повноцінна система моніторингу здоров'я і активності вашого собаки [5]. Пристрій кріпиться до нашийника і працює разом із мобільним застосунком, де ви бачите все: від місцезнаходження у реальному часі, до змін у поведінці, які можуть сигналізувати про проблеми зі здоров'ям.

Головна функція мобільного застосунку - це відстеження активності. Whistle фіксує, скільки часу собака був активним, скільки відпочивав, скільки калорій спалив. Це допомагає власнику зрозуміти, чи отримує тварина достатньо навантаження і чи все з нею гаразд. Також можна встановлювати цілі по активності й отримувати щоденні звіти.

Другий важливий аспект здоров'я і поведінка. Whistle аналізує зміну режиму сну, частоту пиття, активність. Потім система повідомляє, якщо щось виходить за межі норми. Це дозволяє виявити потенційні проблеми ще до появи явних симптомів.

Функція GPS-відстеження дозволяє у будь-який момент побачити, де знаходиться собака. Якщо улюбленець раптом вибіг за межі дозволеної зони,

застосунок одразу надсилає сповіщення. Власник може прокласти маршрут до місцезнаходження собаки прямо з телефону.

У застосунку зберігається вся історія: активність, поведінка, місцезнаходження, цілі та звіти – все в одному місці. Whistle також дозволяє ділитися даними з ветеринаром для точнішої діагностики. Користувач знає, де собака, що з нею відбувається, і можете реагувати швидко, якщо щось іде не так. Цей застосунок є гарним вибором для власників дуже активних собак.

## 1.3 Постановка задачі

Завданням кваліфікаційної роботи є розробка мобільного застосунку для догляду за собаками. Мобільний застосунок має вирішувати за допомогою нейронної мережі такі задачі, як аналіз фізичних характеристик собаки, аналіз стану здоров'я собаки, аналіз породи собаки та віку, аналіз норми харчування та навантажень для собаки. Мобільний застосунок має шукати лікаря поблизу та пропонувати можливості зв'язатися з ним, має створювати замітки користувача та нагадування, щоб нормалізувати медичне життя собаки. Мобільний застосунок має надавати якісні поради з виховання, тренування та комунікації з собакою.

Для виконання задачі необхідно створити та протестувати модель нейронної мережі для аналізу характеристик собаки, створити мобільний застосунок і інтегрувати до нього створену нейронну мережу.

Тестування є завершальним етапом розробки і дозволяє переконатися у його працездатності та стабільності функціонування. Тестові набори мають покривати різноманітні ситуації на кожному етапі роботи з різноманітними тваринами.

Результатом кваліфікаційної роботи має стати мобільний застосунок, який поєднує сучасні технології і зручний інтерфейс для щоденного використання власниками собак.

# 2 ОГЛЯД ТЕХНОЛОГІЙ РОЗРОБКИ

## 2.1 Опис проєкту

Мобільний застосунок розроблено за допомогою мови програмування python у середовищі PyCharm від розробника JetBrains [6]. Це вже знайомий варіант для розробки з приємним графічним інтерфейсом.

Модель для розпізнавання собаки навчена через середовище DataSpell, від того ж розробника. Особливістю середовища є можливість одночасно запускати код на Jupyter та Python [7], використовуючи інтерпретатори Conda та Python [8].

Набори даних було створено на основі вже існуючих в вільному доступі на платформі Kaggle [9]. Так як задача специфічна, необхідно змінювати частину готових наборів даних і адаптувати їх до власних потреб.

## 2.2 Опис програмного забезпечення

Розробка мобільного застосунку складається з декількох кроків. Першим кроком стало створення тестової версії архітектури програми за допомогою мови програмування Python у середовищі Pycharm на основі пакету Kyvi.

Другим кроком стала розробка моделі згорткової нейронної мережі за допомогою мови програмування Jupyter та Python, завдяки сучасному та зручному середовищу DataSpell, де можна вести один проєкт використовуючи 2 формати файлів pyhon file та jupyter notebook.

Третім кроком стала інтеграція нейронної мережі в середовищі DataSpell, інтеграція пошуку Google Maps і тестування мобільного застосунку.

### 2.2.1 Мова програмування Python

Python – це високорівнева мова програмування загального призначення, створена Ґвідо ван Россумом і вперше представлена у 1991 році. Вона вирізняється лаконічним, читабельним синтаксисом, що наближений до природної мови. Така особливість знижує поріг входу для початківців і водночас забезпечує гнучкість для професійних розробників. Python підтримує кілька парадигм програмування, включаючи об’єктно-орієнтовану, процедурну та функціональну [28].

Однією з ключових переваг Python є його широка сфера застосування. Мова активно використовується в розробці вебзастосунків зокрема, з використанням фреймворків Django та Flask. У галузях машинного навчання та штучного інтелекту завдяки бібліотекам TensorFlow та PyTorch. При аналізі та обробці даних за допомогою Pandas та Numpy, а також у наукових дослідженнях та автоматизації процесів. Така універсальність дозволяє Python залишатися затребуваним інструментом у різних галузях промисловості й науки.

Серед технічних характеристик мови варто відзначити динамічну типізацію, автоматичне керування пам’яттю, високу інтеграцію з іншими мовами, наприклад, C/C++. Також наявність розвиненої стандартної бібліотеки робить Python ефективним для швидкої розробки прототипів, створення модульних систем і реалізації складних обчислювальних задач.

Python має одну з найбільших і найактивніших спільнот розробників. Вона забезпечує постійне оновлення бібліотек, підтримку документації, а також сприяє формуванню відкритого для бажаючих середовища програмного забезпечення. Широка підтримка з боку академічних установ і великих IT-компаній як Google, NASA, Netflix та Meta, свідчить про популярність цієї мови навчання та її невід’ємність в сучасному технологічному світі.

### 2.2.2 Середовище програмування Jupyter

Jupyter – це інтерактивне середовище програмування, орієнтоване на обчислення, аналіз даних і візуалізацію. Проєкт Jupyter виник у 2014 році як еволюція проєкту IPython і швидко набув популярності серед дослідників, аналітиків і викладачів. Назва Jupyter є похідною від назв трьох мов програмування – Julia, Python та R – але підтримка мов значно ширша. Середовище сумісне з понад 40 мовами, що забезпечується за допомогою так званих «kernels».

Основною складовою Jupyter є Jupyter Notebook – інтерактивний документ, у якому можна комбінувати код, текст у форматі Markdown [10], математичні формули, графіку та таблиці в єдиному робочому просторі. Це робить Jupyter надзвичайно зручним інструментом для експериментальної розробки, документування процесу аналізу даних і створення відтворюваних наукових досліджень. Вбудовані засоби візуалізації дозволяють швидко оцінювати проміжні результати та приймати обґрунтовані рішення.

Jupyter не є мовою програмування в класичному розумінні, проте він виступає як високорівневе середовище, яке організовує та спрощує взаємодію з мовами, такими як Python, R [11], Julia [12], Scala [13] та інші. Його модульна архітектура дозволяє розширення функціональності через плагіни, інтеграцію з хмарними сервісами та використання в поєднанні з інструментами штучного інтелекту, обчислювальних кластерів і контейнеризації.

У науковому середовищі Jupyter отримав широке визнання завдяки своїй ролі у відкритій науці та репродуктивних дослідженнях. Такі організації, як CERN, NASA, MIT та Google активно використовують Jupyter у своїх аналітичних та дослідницьких проєктах. Крім того, платформи типу Google Colab надають віддалений доступ до Jupyter-сумісного середовища, що знижує вимоги до локальних ресурсів користувача.

### 2.2.3 Інтерпретатор Python

Інтерпретатор – це програма, яка виконує код, написаний мовою програмування. Відповідно, він читає й аналізує вихідний код на Python та інтерпретує його, построково виконуючи інструкції [29].

Інтерпретатор Python – це виконання коду Python і забезпечення правильної роботи програми. Він перетворює вихідний код Python на машинні інструкції, які процесор може зрозуміти і виконати. Під час запуску скрипту або програми на Python, інтерпретатор зчитує код, перевіряє його синтаксис на помилки і построково виконує інструкції [14].

Одна з ключових особливостей інтерпретатора Python – інтерактивність. Це означає, що відбувається взаємодіяти з інтерпретатором безпосередньо, вводячи команди в інтерактивному командному рядку Python. Це корисно для тестування невеликих кодових фрагментів, налагодження та швидкого експериментування. Інтерпретатор Python працює поетапно, виконуючи вихідний код Python построково.

Інтерпретатор зчитує вихідний код Python і розбиває його на лексеми або токени. Лексеми являють собою найменші логічні елементи мови, такі як ключові слова, ідентифікатори, оператори та числа. Цей процес називається лексичним аналізом або токенізацією.

Після лексичного аналізу інтерпретатор аналізує лексичні токени і будує синтаксичне дерево або абстрактне синтаксичне дерево. Синтаксичне дерево являє собою ієрархічну структуру, яка показує, які операції та вирази пов’язані одна з одною згідно із синтаксисом Python.

Після успішного синтаксичного аналізу інтерпретатор починає виконання коду, використовуючи синтаксичне дерево. Він построково переходить по дереву і виконує кожну інструкцію, починаючи з верхнього рівня і рухаючись вниз по дереву. Під час виконання коду інтерпретатор також керує пам’яттю, змінними, областями видимості та іншими аспектами виконання програми.

Python вважається інтерпретованою мовою програмування, а не компільованою. Різниця між інтерпретатором і компілятором полягає в тому, як код програми виконується і перетворюється на машинний код [15].

Інтерпретатор читає й аналізує вихідний код програми безпосередньо під час виконання. Він виконує код построково і перетворює його на машинний код при отриманні. Кожна інструкція інтерпретується під час кожного запуску програми. Інтерпретатори забезпечують більшу гнучкість, оскільки дають змогу негайно бачити результати змін у коді та навіть взаємодіяти з кодом в інтерактивному режимі.

### 2.2.4 Система Conda

Conda це кросплатформенна система керування пакетами та ізольованими середовищами, розроблена компанією Anaconda [16]. Вона спеціально оптимізована для роботи з мовами програмування Python, R та іншими, що використовуються в наукових і аналітичних обчисленнях. Conda дозволяє легко встановлювати, оновлювати та видаляти пакети, разом із їхніми залежностями, а також створювати незалежні середовища для кожного проєкту, запобігаючи конфліктам між бібліотеками.

На відміну від стандартного менеджера pip, Conda працює з бінарними пакетами, що дозволяє уникнути компіляції з вихідного коду, пришвидшуючи інсталяцію. Крім того, вона підтримує не лише Python-бібліотеки, а й системні залежності, включно з компіляторами, бібліотеками C/C++ та іншими інструментами [17]. Conda активно використовується в наукових проєктах, де важлива стабільність обчислювального середовища та повторюваність результатів.

Завдяки своїй гнучкості та стабільності, Conda стала стандартним інструментом в екосистемі Anaconda – дистрибутиву Python для наукових обчислень, машинного навчання та візуалізації даних.

### 2.2.5 Середовище розробки Pycharm

PyCharm – інтегроване середовище розробки, яке використовується для програмування на Python. Вперше випущене у 2010 році, воно швидко стало одним із найпопулярніших інструментів серед Python-розробників завдяки поєднанню інтелектуальних функцій автодоповнення, статичного аналізу коду, глибокої інтеграції з фреймворками та зручного інтерфейсу. Існує дві основні версії PyCharm: Community – безкоштовна, з базовими функціями та Professional – комерційна, з розширеною підтримкою веброзробки, баз даних та наукових інструментів.

Серед ключових можливостей PyCharm – підтримка автоматичного завершення коду, рефакторингу, пошуку використань, інтегрованого тестування, наприклад, з unittest або pytest. PyCharm також підтримує інтеграцію з Docker, Conda, Jupyter Notebook і віддаленими середовищами, що робить його зручним інструментом не лише для розробки, а й для розгортання застосунків.

Інтелектуальний аналіз коду – одна з найбільш вагомих переваг PyCharm. Середовище виявляє синтаксичні помилки ще до запуску програми, пропонує оптимізації, підказує варіанти імпорту, виявляє «мертвий» код і навіть допомагає дотримуватися стилістичних стандартів, наприклад, PEP8). Такі функції значно підвищують якість і підтримуваність коду в довготривалих проєктах.

Для веброзробки PyCharm Professional пропонує підтримку HTML, CSS, JavaScript, фреймворків Django, Flask і інших технологій фронтенду та бекенду. Це дозволяє створювати повноцінні вебзастосунки у межах єдиного середовища без необхідності перемикатися між окремими редакторами. Крім того, інтегроване середовище розробки підтримує роботу з базами даних, надаючи зручні інструменти для SQL-запитів і візуалізації структури даних.

### 2.2.6 Середовище розробки Dataspell

DataSpell – це інтегроване середовище розробки, розроблене компанією JetBrains для аналітиків даних, науковців та інженерів з машинного навчання. На відміну від універсального PyCharm, DataSpell фокусується на задачах, пов'язаних із аналізом даних, інтерактивним програмуванням і візуалізацією результатів. Основна мета DataSpell - це об'єднання всіх необхідних інструментів для роботи з даними в одному робочому просторі.

Однією з ключових особливостей DataSpell є глибока інтеграція з Jupyter Notebook. Користувачі можуть працювати з Jupyter-ноутбуками безпосередньо у вікні IDE, редагуючи як код, так і текстові блоки Markdown у зручному середовищі. На відміну від браузерної версії Jupyter, DataSpell забезпечує покращену навігацію, підсвітку синтаксису, автодоповнення та зручне управління середовищами виконання.

DataSpell також підтримує стандартні Python-скрипти, інтеграцію з системами керування версіями як Git, візуалізацію даних у вигляді графіків і таблиць, роботу з базами даних, Docker-контейнерами та Conda-середовищами. Крім того, IDE дозволяє швидко перемикатися між нотбуками, Python-файлами та консолями, що оптимізує робочий процес при виконанні експериментальних обчислень.

Особливо корисною функцією є інтерактивне попереднє переглядання результатів обчислень: графіки, набори даних та інші структури виводяться в окремому вікні з можливістю фільтрації, сортування та масштабування. Це робить DataSpell надзвичайно ефективним для робіт із великими обсягами даних, зокрема у сферах фінансового аналізу чи машинного навчання.

### 2.2.7 Табличний процесор Excel

Microsoft Excel або Microsoft Office Excel – табличний процесор, програма для роботи з електронними таблицями, створена корпорацією

Microsoft для Microsoft Windows, Windows NT і macOS. Програма входить до складу офісного пакета Microsoft Office.

Завдяки тому, що лист Excel являє собою готову таблицю, Excel часто використовують для створення документів без усіляких розрахунків, що просто мають табличне представлення. У Excel легко можна створювати різні види графіків і діаграм, які беруть дані для побудови з комірок таблиць. Excel можуть використовувати звичайні користувачі для елементарних розрахунків. Excel також містить багато математичних і статистичних функцій, завдяки чому його можуть використовувати школярі і студенти для розрахунків курсових, лабораторних робіт. Excel часто використовується в бухгалтерії, у багатьох фірмах це основний інструмент для оформлення документів, розрахунків і створення діаграм.

Microsoft Excel широко використовується для первинного опрацювання та аналізу наборів даних, особливо на ранніх етапах роботи з даними або у випадках, коли обсяг інформації є відносно невеликим. Excel дозволяє зручно імпортувати дані з різних джерел (CSV, TXT, бази даних), виконувати фільтрацію, сортування, обчислення за допомогою формул, а також візуалізувати дані за допомогою графіків і діаграм.

Одна з головних переваг Excel – його доступність і простота у використанні, що робить його популярним серед аналітиків, менеджерів та дослідників без глибокої технічної підготовки. Проте при роботі з великими обсягами даних або складною аналітикою Excel має обмеження щодо продуктивності, автоматизації та масштабованості, тому на більш просунутому рівні часто застосовуються спеціалізовані інструменти на кшталт Python, R або SQL.

### 2.2.8 Google Cloud Platform

Google Cloud Platform – це набір послуг хмарних обчислень від Google, який надає низку модульних хмарних сервісів, включаючи обчислення,

зберігання даних, аналітику даних і машинне навчання, а також набір інструментів управління. Він працює на тій самій інфраструктурі, яку Google використовує для своїх продуктів для кінцевих користувачів, таких як Google Search, Gmail і Google Docs.

Як і інші хмарні рішення, програми, розміщені на Google Cloud Platform, схильні до помилок розподілених обчислень – низки хибних уявлень, які можуть призвести до значних проблем при розробці та розгортанні програмного забезпечення.

### 2.2.9 Google Places API

За допомогою Google Places API можна створювати функції, що враховують місцезнаходження, щоб зробити докладні дані про місцезнаходження доступними для ваших користувачів. Дані, доступні через Places API, побудовані на одній з найточніших, найсучасніших і найповніших моделей місць у реальному світі.

## 2.3 Опис моделі нейронної мережі

Для реалізації поставленої задачі необхідно розробити модель глибокого навчання для автоматичної класифікації зображень, що містять зображення собак різних порід. Необхідно обрати найкращу модель для вирішення завдань комп'ютерного зору.

Загальний процес побудови моделі включає кілька етапів: попередня обробка даних, зокрема перебір XML-анотацій [18], виявлення об’єктів за координатами обмежувальних прямокутників зображення, обрізання зображень, нормалізація та масштабування. Ці кроки допоможуть також оптимізувати модель. Після цього формується навчальний та валідаційний набори даних.

Модель використовує попередньо натреновану нейронну мережу, що була модифікована для вирішення конкретного завдання класифікації порід. Це дозволяє ефективно використовувати вже наявні знання моделі для специфічної задачі, що значно пришвидшує процес навчання та покращує якість результатів.

Розроблена модель дозволить автоматично ідентифікувати породу собаки на зображенні з високою точністю. Такий підхід може бути ефективно використаний у практичних застосуваннях, пов’язаних із розпізнаванням тварин, ветеринарною діагностикою, зоологією та мобільними застосунками.

## 2.4 Типи нейронних мереж

Існує багато типів штучних нейронних мереж.

Штучні нейронні мережі – це обчислювальні моделі, натхнені біологічними нейронними мережами. Їх поведінка прагне наблизитись до поведінки нейронів тварин та електричних сигналів в біологічному мозку, які вони передають між «входом», наприклад, від очей або нервових закінчень у руці, обробкою, та «виходом» із мозку, наприклад, реакцією на світло, дотик або тепло.

Спосіб забезпечення нейронами семантичного зв'язку є областю поточних досліджень [19]. Більшість штучних нейронних мереж лише дещо схожі на свої складніші біологічні аналоги, але вони дуже ефективні у виконанні поставлених завдань, наприклад, класифікування чи сегментування.

Деякі штучні нейронні мережі є адаптивними системами, і їх використовують, наприклад, для моделювання популяцій та середовищ, які постійно змінюються.

Нейронні мережі можуть бути апаратними, нейрони подано фізичними складовими та програмними, й можуть використовувати різноманітні топології та алгоритми навчання.

### 2.4.1 Нейронна мережа прямого поширення

Нейронна мережа прямого поширення – вид нейронної мережі [20], в якій сигнали поширюються в одному напрямку, починаючи від вхідного шару нейронів, через приховані шари до вихідного шару і на вихідних нейронах отримується результат опрацювання сигналу. В мережах такого виду немає зворотніх зв’язків. Протилежним видом нейронних мереж із зворотніми зв’язками є рекурентні нейронні мережі.

Прикладом нейронної мережі прямого поширення є перцептрон Розенблатта [21], від якого і беруть свій початок нейромережі прямого розповсюдження. В літературі часто термін перцептрон, багатошаровий перцептрон та нейромережа прямого поширення застосовуються синонімічно. Власне, між різними видами перцептронів спільне одне – вони усі є нейромережами з прямим поширенням сигналу, різняться в основному кількістю шарів, функцією активації та методом навчання.

### 2.4.2 Метод групового урахування елементів

Метод групового урахування аргументів – сімейство індуктивних алгоритмів для математичного моделювання багатопараметрових даних [22]. Метод заснований на селективному відборі моделей, на основі яких будуються складніші моделі. Точність моделювання на кожному наступному кроці збільшується за рахунок ускладнення моделі.

Метод групового урахування аргументів  має повністю автоматичну структурну та параметричну оптимізацію моделі. Передавальними функціями вузлів є поліноми Колмогорова – Габора, які допускають додавання та множення. Він використовує глибокий багатошаровий перцептрон із вісьмома шарами. Це мережа керованого навчання, яка зростає шар за шаром, де кожен шар тренується за допомогою регресійного аналізу. Непотрібні елементи виявляються за допомогою затверджувального набору й

відсікаються за допомогою регуляризації. Розмір та глибина отримуваної мережі залежать від поставленого завдання.

### 2.4.3 Автокодувальник

Автокодувальник – це один із типів штучних нейронних мереж, який використовують для навчання ефективних кодувань немічених даних, некерованого навчання. Це кодування перевіряється та вдосконалюється шляхом намагання відтворювати вхід із цього кодування.

Автокодувальник навчається поданню для набору даних, зазвичай для зниження розмірності, шляхом тренування цієї мережі ігнорувати незначущі дані.

Автокодувальник має дві основні частини: кодувальник, що відображує вхід до коду, та декодувальник, що відображує код до відбудови входу.

Існують варіанти, спрямовані на змушування навчених подань набувати корисних властивостей. Прикладами є регуляризовані автокодувальники – розріджені, знешумлювальні та стягальні відповідно, що є дієвими в навчанні подань для пізніших задач класифікування, та варіаційні автокодувальники, із застосуванням як породжувальних моделей. Автокодувальники застосовують у багатьох задачах, від розпізнавання обличчя до оволодівання семантичним значенням слів.

Найпростішим способом виконувати задачу копіювання ідеальним чином було б просто дублювати сигнал. Натомість автокодувальники зазвичай змушені відбудовувати вхід приблизно, залишаючи в копії лише найдоречніші аспекти даних.

### 2.4.4 Ймовірніста нейронна мережа

Ймовірнісна нейронна мережа – вид штучних нейронних мереж, який використовує баєсову статистику для виконання певних завдань [23].

Ймовірнісна нейронна мережа була розроблена Дональдом Спехтом. Виходи мережі можна інтерпретувати як оцінки ймовірності належності елементу певному класу. Ймовірнісна мережа вчиться оцінювати функцію густини ймовірності, її вихід розглядається як очікуване значення моделі в даній точці простору входів. Це значення пов'язане з густиною ймовірності спільного розподілу вхідних і вихідних даних.

Задача оцінки густини ймовірності відноситься до області баєсової статистики. Звичайна статистика по заданій моделі показує, яка ймовірність того або іншого виходу. Баєсова статистика інтерпретує по-іншому: правильність моделі оцінюється по наявних достовірних даних, тобто надає можливість оцінювати густину ймовірності розподілу параметрів моделі по наявних даних.

При рішенні задач класифікації можна оцінити густину ймовірності для кожного класу, порівняти між собою ймовірності приналежності до різних класів і обрати модель з параметрами, при яких густина ймовірності буде найбільшою.

Оцінка густини ймовірності в мережі заснована на ядерних оцінках. Якщо приклад розташований в даній точці простору, тоді в цій точці є певна густина ймовірності. Кластери з близько розташованих точок свідчать, що в цьому місці густина ймовірності велика. Поблизу спостереження є більша довіра до рівня густини, а по мірі віддалення від нього довіра зменшується і плине до нуля. В методі ядерних оцінок в точку, що відповідає кожному прикладу, поміщається деяка проста функція, потім вони всі додаються і в результаті утворюється оцінка для загальної густини ймовірності. Найчастіше як ядерні функції беруть дзвоноподібні функції. Якщо є достатня кількість навчальних прикладів, такий метод дає добрі наближення до істинної густини ймовірності.

Ймовірнісна мережа має три прошарки: вхідний, радіальний та вихідний. Радіальні елементи беруться по одному на кожний приклад. Кожний з них містить гаусову функцію з центром в цьому прикладі.

Кожному класу відповідає один вихідний елемент. Вихідний елемент з'єднаний лише з радіальними елементами, що відносяться до його класу і підсумовує виходи всіх елементів, що належать до його класу. Значення вихідних сигналів утворюються пропорційно ядерних оцінок ймовірності приналежності відповідним класам.

### 2.4.5 Нейронна мережа з часовою затримкою

Нейронна мережа з часовою затримкою – це архітектура штучної нейронної мережі [24], основною метою якої є класифікація паттернів незалежно від зсуву, тобто не вимагає явного попереднього визначення початкової та кінцевої точок схеми.

Такі мережі вперше запропонували використовувати для класифікації фонеми в мовних сигналах для автоматичного розпізнавання мови, де автоматичне визначення точних сегментів або меж є складним чи неможливим. Нейронна мережа з часовою затримкою розпізнає фонеми та їхні основні акустичні або фонетичні особливості, незалежно від часових зрушень, тобто положення в часі.

Вхідний сигнал доповнюється затриманими копіями як іншими входами, нейронна мережа є інваріантною для часової зміни, оскільки вона не має внутрішнього стану. Нейронні мережі з затримкою часу, як і інші нейронні мережі, працюють з декількома взаємопов'язаними шарами, що складаються з кластерів. Ці кластери призначені для представлення нейронів у мозку, і, як і мозок, кожен кластер повинен зосередити увагу тільки на невеликих регіонах вхідних даних. Прототип мережі має три шари кластерів, один для введення, один для виведення, і середній шар, який обробляє вхід через фільтри. Через їх послідовність ці мережі реалізуються як Нейронна мережа прямого поширення, а не рекурентна нейронна мережа.

Для досягнення інваріантності зсуву часу, до входу додаються набір затримок, наприклад, аудіофайл, зображення тощо, так що дані представлені

в різні моменти часу. Ці затримки є довільними та специфічними, що, як правило, означає, що вхідні дані налаштовані на певний шаблон затримки. Це і є вся виконана робота по створенню адаптивної часової затримки мереж, де цю ручну настройку викорінено. Затримки є спробою додати часовий вимір до мережі, який відсутній в рекурентних нейронних мережах або багатошарових персептронах зі змінним вікном. Поєднання попередніх входів із поточними вкладами робить підхід нейронних мереж з часовою затримкою унікальним.

Ключовою особливістю нейронної мережі з часовою затримкою є здатність виражати зв'язок між входами у часі. Це співвідношення може бути результатом детектора особливостей і використовується в рамках нейронних мереж з часовою затримкою для розпізнавання шаблонів між затриманими входами.

Одним з основних переваг нейронних мереж є відсутність залежності від попередніх знань для встановлення банків фільтрів на кожному шарі. Однак це тягне за собою те, що мережа повинна вивчати оптимальне значення для цих фільтрів шляхом обробки численних входів для навчання. Навчання під керівництвом, як правило, є алгоритмом навчання, пов'язаним з цими мережами, завдяки його силі в розпізнаванні образів та наближенні функцій. Кероване навчання зазвичай здійснюється за допомогою алгоритму зворотного поширення помилки.

### 2.4.6 Згорткова нейронна мережа

Згорткові нейронні мережі є одним із найефективніших різновидів штучних нейронних мереж [25], які призначені для роботи з упорядкованими даними, передусім з візуальною інформацією – зображеннями, відео або сигналами. Вони широко застосовуються в задачах комп’ютерного зору, розпізнавання образів, медичної діагностики, робототехніки, біометрії та в інших прикладних сферах, де критично важлива здатність виявляти складні

структури у великих масивах даних. Архітектура згорткових мереж базується на принципах, які імітують роботу зорової кори людини, що реагує на окремі просторові особливості сприйнятого середовища.

На відміну від класичних багатошарових персептронів, згорткові нейронні мережі використовують локальні області вхідних даних та фільтри, які виявляють ключові ознаки. У таких мережах нейрони не пов’язані з усіма входами одночасно, а працюють із невеликими локальними фрагментами, що дозволяє моделі зосередитись на просторових залежностях і зменшити загальну кількість параметрів. Завдяки цьому мережа навчається розпізнавати важливі фрагменти, наприклад краї, контури або текстури, які пізніше комбінуються у складніші ознаки.

Типова структура згорткової нейронної мережі складається з послідовності згорткових шарів, шарів зменшення розмірності та завершальних повнозв’язних шарів. Згорткові шари відповідають за виявлення ознак, шари зменшення розмірності – за зниження кількості обчислень і уникнення перенавчання, а кінцеві шари здійснюють остаточну інтерпретацію ознак для розв’язання задачі класифікації, виявлення об’єктів або інших цілей. Такий ієрархічний підхід дає змогу моделі формувати глибоке представлення інформації.

Під час навчання фільтри згорткових шарів автоматично налаштовуються шляхом мінімізації функції помилки, що обчислюється між фактичним і очікуваним результатом. Використовуються методи градієнтної оптимізації, які забезпечують поступове вдосконалення внутрішніх параметрів мережі. На початкових шарах мережа зазвичай виявляє базові геометричні елементи, тоді як на глибших рівнях формуються високорівневі ознаки, здатні ідентифікувати складні об’єкти чи явища. Це дозволяє мережі працювати навіть у ситуаціях, коли вхідні дані частково пошкоджені або спотворені.

Згорткові мережі демонструють здатність до інваріантності щодо зсувів, обертання або масштабування об’єктів на вхідних зображеннях. Це

досягається завдяки локальності обробки і повторному використанню фільтрів, що значно підвищує загальну ефективність моделі. Згорткові мережі добре масштабуються для обробки великих наборів даних і активно використовуються на високопродуктивному обчислювальному обладнанні, зокрема на графічних процесорах.

Серед найвідоміших представників згорткових нейронних мереж – архітектури LeNet, AlexNet, VGG, GoogLeNet, ResNet та інші. Вони стали основою для сучасних систем комп'ютерного зору, здатних автоматично класифікувати мільйони зображень із високою точністю. Сьогодні згорткові нейронні мережі активно впроваджуються в медицину для діагностики за візуальними даними, у транспортну галузь для розпізнавання дорожніх об'єктів, у промисловість для контролю якості, а також у сферу безпеки, аграрні технології та цифрову культуру. Актуальними залишаються дослідження, спрямовані на оптимізацію архітектур, пояснюваність моделей і підвищення їх енергоефективності.

### 2.4.7 Глибока складальна мережа

Глибока складальна мережа є багаторівневою генеративною моделлю, що складається з кількох шарів латентних змінних [26]. Кожен шар цієї мережі побудований на основі обмежених машин Больцмана, які є стохастичними нейронними мережами, здатними моделювати розподіли ймовірностей. Глибокі складальні мережі були запропоновані як спосіб ефективного попереднього навчання глибоких моделей у ситуаціях, коли використання стандартного зворотного поширення помилки не дає бажаного результату через проблеми з локальними мінімумами або втратою градієнта на глибоких рівнях.

Ключовою особливістю глибоких складальних мереж є можливість поетапного навчання кожного шару окремо в ненаглядовий спосіб. На першому етапі навчання кожна пара сусідніх шарів тренується як окрема

обмежена машина Больцмана. Після навчання одного шару його параметри фіксуються, і наступний шар отримує вхідні дані, які є виводом попереднього. Такий підхід дозволяє мережі поступово формувати багаторівневе представлення даних, де кожен наступний шар витягує все складніші та абстрактніші ознаки.

Після завершення попереднього навчання може бути застосовано додаткове коригування всієї мережі у режимі з учителем. Для цього до вищого шару додається вихідний класифікатор, наприклад, логістична регресія або персептрон, після чого всі вагові коефіцієнти уточнюються за допомогою зворотного поширення помилки. Комбінування ненаглядового попереднього навчання з наглядовим донавчанням забезпечує кращу ініціалізацію параметрів, що значно покращує якість класифікації порівняно з прямим навчанням мережі з нуля.

Глибокі складальні мережі є ефективними в задачах, де спостережувані дані мають приховану структуру або високий рівень шуму. Завдяки своїй здатності до побудови щільних ймовірнісних моделей, ці мережі добре працюють у розпізнаванні образів, обробці мови, відновленні сигналів, виявленні аномалій та моделюванні складних залежностей між ознаками. Їх застосовують у біоінформатиці, рекомендаційних системах, фінансовому аналізі та інших галузях, де потрібно навчатися з неповних або неструктурованих даних.

З технічного погляду, навчання глибоких складальних мереж базується на алгоритмах, які враховують стохастичну природу прихованих змінних. Найпоширенішим є контрастивне дивергентне навчання, яке дозволяє швидко наближати градієнт логарифмічної правдоподібності, що є обчислювально ефективнішим за класичні методи Монте-Карло. При цьому мережа не лише навчається відтворювати вхідні дані, але й здатна генерувати нові приклади на основі засвоєних закономірностей.

Попри те, що згодом глибокі складальні мережі частково поступилися місцем згортковим і рекурентним архітектурам у ряді прикладних задач,

вони залишаються важливими з погляду розвитку глибокого навчання як фундаментальної дисципліни. Їх використання стало ключовим етапом у перехідному періоді між традиційними методами машинного навчання та сучасними глибокими моделями. Більш того, досвід побудови глибоких складальних мереж ліг в основу нових класів нейронних мереж, зокрема варіаційних автокодувальників і генеративних моделей, що сьогодні активно досліджуються в контексті пояснюваного штучного інтелекту.

## 2.5 Архітектура нейронної мережі

У рамках даного дослідження було обрано модель глибокого навчання для автоматичної класифікації зображень, що містять зображення собак різних порід. Модель базуватимется на згортковій нейронній мережі, що є одним із найефективніших підходів у задачах комп'ютерного зору. Архітектура моделі буде адаптована для багатокласової класифікації, де кожен клас відповідає окремій породі собаки.

Загальний процес побудови моделі включає кілька етапів: попередня обробка даних, парсинг XML-анотацій, виявлення об'єктів за координатами обмежувальних прямокутників, обрізання зображень, нормалізація та масштабування. Після цього формується навчальний та валідаційний набори даних. Модель використовує попередньо натреновану нейронну мережу, що була модифікована для вирішення конкретного завдання класифікації порід. Це дозволяє ефективно використовувати вже наявні знання моделі для специфічної задачі, що значно пришвидшує процес навчання та покращує якість результатів.

У якості алгоритму оптимізації використовуються сучасні методи стохастичної градієнтної оптимізації. Для оцінки точності моделі застосовується стандартна метрика класифікації – точність результату, яка визначає відсоток правильно класифікованих зображень у валідаційному наборі.

## 2.6 Бібліотеки та фреймворки

Pandas – бібліотека для обробки табличних даних, яка використовувалася для зберігання, фільтрації та аналізу метаданих, що стосуються зображень та відповідних анотацій.

Pillow (PIL) – інструмент для роботи з растровими зображеннями. Забезпечував відкриття, обрізання та збереження фрагментів зображень на основі координат bounding box.

os – стандартний модуль Python, який використовувався для роботи з файловою системою, зокрема для побудови шляхів до зображень та перевірки їхньої наявності.

xml.etree.ElementTree – стандартна бібліотека Python для парсингу XML-документів. Застосовувалась для обробки анотацій у форматі Pascal VOC, що містять інформацію про об'єкти на зображеннях.

scikit-learn (sklearn) – використовувалася для поділу даних на навчальні та валідаційні набори із збереженням збалансованості класів.

PyTorch – основний фреймворк для побудови та навчання глибокої нейронної мережі. Забезпечував реалізацію моделі, функції втрат, оптимізатора та тренувального циклу.

torchvision – доповнення до PyTorch, що надає доступ до попередньо навчених моделей та засоби трансформації зображень. У проєкті була використана модель ResNet18, попередньо навчена на ImageNet, як основа для класифікації зображень за породами.

Onnx – це відкрита екосистема, яка дозволяє розробникам обирати правильні інструменти в міру розвитку проєкту. Вона надає формат з відкритим вихідним кодом для моделей штучного інтелекту, як для глибокого навчання, так і для традиційного машинного навчання.

Json – можна використовувати для роботи з даними формату Json.

Kivy –це безкоштовний фреймворк Python з відкритим вихідним кодом для розробки мобільних застосунків та іншого програмного забезпечення з

простим інтерфейсом користувача.

threading – модуль багатопотоковості надає можливість запускати декілька потоків одночасно в рамках одного процесу. Він дозволяє створювати потоки та керувати ними, що дає можливість виконувати завдання паралельно, спільно використовуючи простір пам'яті.

time – цей модуль надає різні функції, пов'язані з часом.

Requests – це проста бібліотека HTTP.

Webbrowser – модуль надає високорівневий інтерфейс для відображення веб-документів користувачам.

# 3 МОДЕЛЬ НЕЙРОННОЇ МЕРЕЖІ

## 3.1 Створення моделі нейронної мережі

Проєкт передбачає розробку моделі глибокого навчання, здатної класифікувати породи собак за зображенням. Основна задача моделі – на основі фото розпізнати породу собаки, щоб використати ці дані в створенні характеристики тварини.

## 3.2 Підготовка даних

Початковий етап полягав у створенні структури даних: CSV-файли з мітками та шляхами до зображень, а також каталог з обрізаними зображеннями собак. Для покращення результатів було важливо збалансувати кількість прикладів для кожної породи та підготувати їх у форматі, зручному для моделі, чим менше зображення, тим краще.

Для покращення здатності моделі розпізнавати зображення та підвищення її точності та універсальності було використано аугментацію даних. Зміна яскравості, контрасту, насиченості, випадкові повороти та горизонтальні віддзеркалення та масштабування допомогло підвищити точність моделі до 85%.

Це дозволило моделі краще працювати на зображеннях з реального світу, які можуть відрізнятися від тренувального набору.

## 3.3 Архітектура моделі

В якості базової архітектури було використано ResNet18 – популярну згорткову нейронну мережу з хорошим балансом між точністю та швидкістю (рисунок 3.1).

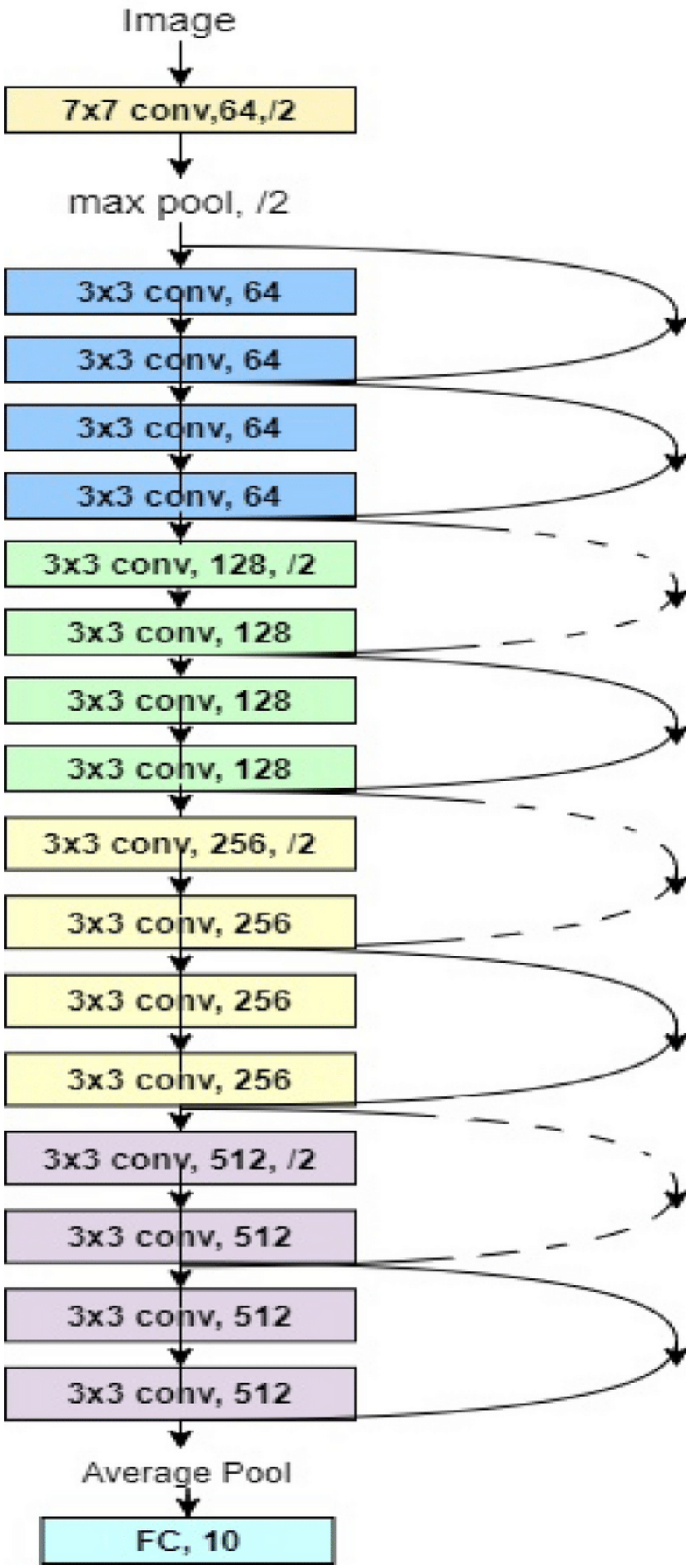


Рисунок 3.1 – Архітектурна діаграма моделі ResNet18

Процес навчання включав: визначення функції втрат (крос-ентропія), використання оптимізатора алгоритму, цикл навчання з 10-ма епохами.

Для оцінки прогресу використовувались метрики точності на тестовій вибірці. Кожна епоха давала можливість моделі покращити свої передбачення, порівнюючи їх із правильними мітками, результати моделі вимірювалися на кожній епосі (лістинги 3.1, 3.2, 3.3).

Лістинг 3.1 – Функція підготовки зображень

```
def crop_objects(df, image_root_folder, save_folder):
    os.makedirs(save_folder, exist_ok=True)
    new_filenames = []
    for idx, row in df.iterrows():
        img_path = os.path.join(image_root_folder,
row["folder"], row["filename"])
        if not os.path.exists(img_path):
            print(f" Пропущено: {img_path}")
            new_filenames.append("")
            continue
        image = Image.open(img_path)
        cropped = image.crop((row["xmin"], row["ymin"],
row["xmax"], row["ymax"]))
        save_name = f"{idx}_{row['label']}.jpg"
        save_path = os.path.join(save_folder, save_name)
        cropped.save(save_path)
        new_filenames.append(save_name)

    df = df.copy()
    df["cropped_filename"] = new_filenames
    return df[df["cropped_filename"] != ""]
```

Лістинг 3.2 – Блок розподілу зображень на тренувальні та тренуємі

```
df_train, df_val = train_test_split(df, test_size=0.2,
stratify=df["label"], random_state=42)
df_train.to_csv(TRAIN_CSV, index=False)
df_val.to_csv(VAL_CSV, index=False)
print(" Розділено на train і val:")
print(f"Train: {len(df_train)} зразків")
print(f"Val: {len(df_val)} зразків")
```

Лістинг 3.3 – Блок для аугментація  вхідних даних

```
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
```

```
    transforms.ColorJitter(brightness=0.2, contrast=0.2,
saturation=0.2),
    transforms.ToTensor()
])
train_dataset = DogDataset(TRAIN_CSV, CROPPED_DIR,
transform=transform)
val_dataset = DogDataset(VAL_CSV, CROPPED_DIR,
transform=transform)
train_loader = DataLoader(train_dataset, batch_size=32,
shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=32)
```

Важливо було також перевірити, чи не відбувається перенавчання на тренувальних даних, перенавчені моделі не ефективні в умовах роботи із новими даними, хоча можуть показувати надзвичайно високі показники при тестуванні. Для цього було додано контроль точності на тестовому наборі (лістинг 3.4).

Лістинг 3.4 – Блок завантаження моделі

```
from torchvision.models import ResNet18_Weights

num_classes = len(train_dataset.label2idx)
weights = ResNet18_Weights.DEFAULT
model = models.resnet18(weights=weights)
model.fc = nn.Linear(model.fc.in_features, num_classes)
```

Було отримано точність в 85.12% для моделі яка навчалась на сотнях порід собак і тисячах фотографій (лістинг 3.5).

Лістинг 3.5 – Блок перевірки точності моделі

```
model.eval()
correct = 0
total = 0
with torch.no_grad():
    for images, labels in val_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

accuracy = 100 * correct / (total)
print(f" Точність на валідації: {accuracy:.2f}%")
```

# 4 СТВОРЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

## 4.1 Загальна архітектура проєкту

Проєкт використовує Kivy для створення інтерфейсу користувача та надає функціонал управління профілями домашніх тварин, додавання зображень, обробки фотографій, ведення календаря подій та пошуку ветеринарних клінік.

Основні модулі:

а) головна програма (PetApp) – реалізує менеджер екранів (ScreenManager) і визначає основну логіку навігації.

б) екран старту (StartScreen) – початкова точка взаємодії користувача з вибором зареєстрованого вихованця або додаванням нового.

в) екран реєстрації (RegisterScreen) – дозволяє додати нового вихованця, завантажити його фото з галереї або зробити знімок камерою.

г) екран профілю (ProfileScreen) – персоналізована сторінка вихованця з полями імені, віку, породи, ваги, статі.

д) головний екран (MainScreen) – містить TabbedPanel з розділами профілю з основними даними собаки, календаря з подіями та порадами для догляду за здоров’ям, через календар відбувається доступ до екрану пошуку ветеринара, через запити Google Places API для пошуку найближчих ветеринарних клінік.

Мобільний застосунок використовує profiles.json для зберігання інформації про вихованців та calendar_data.json для збереження подій.

Робота із зображеннями виконується через пакет PIL (Pillow) для обробки фотографій вихованців, а ONNX Runtime для розпізнавання порід собак. Сповіщення реалізовано через бібліотеку plyer.notification, сповіщення використовується для надсилання нагадувань про заплановані події.

Проєкт завантажено на гугл диск [27].

## 4.2 Інтеграція з Google Places Api

Інтеграція з Google Places API у проєкті дозволяє користувачеві шукати найближчі ветеринарні клініки. Реалізація на основі роботи API-ключа, він завантажується зі змінних оточення або файлу config.json. Якщо ключ не знайдено, пошук, як і доступ до сервісів, неможливий (лістинг 4.1).

Лістинг 4.1 – Функція доступу до ключа

```
def get_google_places_api_key():
    # First try to get from environment variable
    api_key = os.getenv('GOOGLE_PLACES_API_KEY')
    if api_key:
        return api_key

    # Then try to load from config file
    config_file = 'config.json'
    if os.path.exists(config_file):
        try:
            with open(config_file, 'r') as f:
                config = json.load(f)
                return config.get('GOOGLE_PLACES_API_KEY')
        except:
            pass

    return None
```

Застосунок запитує адресу в користувача, в діапазоні 15 км відбувається пошук. Для перетворення адресу в географічні координати використовується Google Geocoding API. Потім все зайве прибирається, та користувачем відбувається пошук за назвою ветеринарної клініки (лістинг4.2).

Лістинг 4.2 – Функція пошуку адреси

```
geocode_url =
f"https://maps.googleapis.com/maps/api/geocode/json?address={quo
te(location)}&key={GOOGLE_PLACES_API_KEY}"
response = requests.get(geocode_url)
data = response.json()

if data['status'] != 'OK':
    loading_popup.dismiss()
```

```
    self.show_error("Не вдалося знайти вказану адресу")
    return

location = data['results'][0]['geometry']['location']
lat, lng = location['lat'], location['lng']

places_url =
f"https://maps.googleapis.com/maps/api/place/nearbysearch/json?l
ocation={lat},{lng}&radius={VET_SEARCH_RADIUS}&type=veterinary_c
are&key={GOOGLE_PLACES_API_KEY}"
response = requests.get(places_url)
data = response.json()

maps_url =
f"https://www.google.com/maps/search/{quote(place['name'])}"
webbrowser.open(maps_url)
```

Після визначення координат виконується запит до Google Places API з параметрами:

- location: координати користувача;
- radius: радіус пошуку (у коді встановлено 15 км);
- type: veterinary_care – категорії ветеринарних клінік.

Якщо клініки знайдені, інформація про них (назва, рейтинг, адреса) відображається користувачеві. При натисканні на клініку створюється посилання на Google Maps для отримання маршруту.

## 4.3 Інтеграція моделі Onnx

Інтеграція ONNX Runtime у проєкт використовується для класифікації порід собак із завантаженого зображення вихованця. Програма завантажує додаткову модель dog_model.onnx, використовуючи ONNX Runtime. Ця модель приймає зображення собаки і передбачає її породу, ґрунтуючись на навчених даних (лістинг 4.3).

Лістинг 4.3 – Блок інтеграції моделі onnx

```
import onnxruntime as ort

MODEL_PATH = "models/dog_model.onnx"
```

```
session = ort.InferenceSession(MODEL_PATH)

input_name = session.get_inputs()[0].name
```

Перш ніж передати фото тварини в нейромережеву модель, воно масштабується, нормалізується та перетворюється на тензор (лістинг 4.4).

Лістинг 4.4 – Блок перетворення та масштабування зображень

```
IMG_SIZE = (224, 224)

def preprocess_image(img_path):
    image = PILImage.open(img_path).convert('RGB')
    transform = transforms.Compose([
        transforms.Resize(IMG_SIZE),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229,
0.224, 0.225])
    ])
    tensor =
transform(image).unsqueeze(0).numpy().astype(np.float32)
    return tensor


input_tensor = preprocess_image(img_path)
output = session.run(None, {input_name: input_tensor})
prediction = np.argmax(output[0])
```

Після обробки зображення воно передається у модель для класифікації. Результатом є індекс породи собаки, який потім зіставляється з назвою породи із заздалегідь підготовленого словника (рисунок 4.1).

```
33: "Ірландський тер'єр",
34: "Норфолк-тер'єр",
35: "Норвіч-тер'єр",
36: "Йоркширський тер'єр",
37: "Жорсткошерстий фокстер'єр",
38: "Лейкленд-тер'єр",
39: "Сіліхем-тер'єр",
40: "Ердельтер'єр",
41: "Керн-тер'єр",
42: "Австралійський тер'єр",
43: "Денді-дінмонт-тер'єр",
44: "Бостон-тер'єр",
45: "Мініатюрний шнауцер",
46: "Ризеншнауцер",
47: "Міттельшнауцер",
48: "Шотландський тер'єр",
49: "Тибетський тер'єр",
50: "Австралійський шовковистий тер'єр",
```

Рисунок 4.1 – Список порід собак

## 4.4 Створення профілю тварини

Створення профілю тварини відбувається через екран реєстрації вихованця і далі використовується в екрані профілю та на головному екрані.

Під час створення нового профілю користувач заповнює анкету з ім'ям, завантажує фото, а потім отримує певну породу за допомогою нейромережевого аналізу (лістинг 4.7).

Основні кроки реєстрації: введення імені вихованця, вибір фото з галереї або зйомка камери, обробка зображення та визначення породи, перехід до профілю тварини з можливістю редагування (лістинг 4.6). Фото обробляється нейромережевою моделлю, і програма відображає передбачувану породу. Можна використати вже наявний профіль тварини або створити новий (лістинг 4.5).

Лістинг 4.5 – Блок  для вибору типу взаємодії з профілем тварини

```
select_btn = Button(
    text=" Вибрати зареєстровану тварину",
    on_press=self.go_to_main,
    size_hint=(0.6, None),
    height=dp(40),
    pos_hint={'center_x': 0.5}
)
add_btn = Button(
    text=" Додати тварину",
    on_press=self.go_to_register,
    size_hint=(0.6, None),
    height=dp(40),
    pos_hint={'center_x': 0.5}
)
def go_to_main(self, *args):
    self.manager.current = 'main'

def go_to_register(self, *args):
    self.manager.current = 'register'
```

Лістинг 4.6 – Блок реєстрації тварини

```
class RegisterScreen(Screen):
    def __init__(self, **kwargs):
        super(RegisterScreen, self).__init__(**kwargs)
        self.layout = BoxLayout(orientation='vertical')
```

```
        self.name_input = TextInput(hint_text="Введіть ім'я
тварини")
        self.layout.add_widget(self.name_input)
        self.layout.add_widget(Button(text=" Зробити фото",
on_press=self.capture_photo))
        self.layout.add_widget(Button(text=" Вибрати з галереї",
on_press=self.choose_image))
        self.layout.add_widget(Button(text=" Назад",
on_press=self.back))
        self.add_widget(self.layout)
```

Лістинг 4.7 – Блок заповнення анкети профілю

```
class ProfileScreen(Screen):
    def __init__(self, **kwargs):
        super(ProfileScreen, self).__init__(**kwargs)
        self.layout = BoxLayout(orientation='vertical')

        self.profile_box = BoxLayout(size_hint_y=None,
height=dp(50))
        self.profile_label = Label(text="Профіль:")
        self.profile_spinner = Spinner(
            text='Виберіть профіль',
            values=('Новий профіль',),
            size_hint_x=0.7
        )
        self.profile_box.add_widget(self.profile_label)
        self.profile_box.add_widget(self.profile_spinner)
        self.layout.add_widget(self.profile_box)

        self.image = Image(size_hint=(1, 0.4))
        self.layout.add_widget(self.image)

        self.name_input = TextInput(hint_text="Ім'я")
        self.breed_input = TextInput(hint_text='Порода')

        self.gender_box = BoxLayout(size_hint_y=None,
height=dp(50))
        self.gender_label = Label(text="Стать:")
        self.gender_spinner = Spinner(
            text='Виберіть стать',
            values=('Хлопчик', 'Дівчинка'),
            size_hint_x=0.7
        )
        self.gender_box.add_widget(self.gender_label)
        self.gender_box.add_widget(self.gender_spinner)

        self.age_input = TextInput(hint_text='Вік')
        self.weight_input = TextInput(hint_text='Вага')
```

## 4.5 Розділ «Календар»

У розділі Календар користувачі можуть переглядати місячний календар з виділеними датами, що містять записи, створювати події (наприклад, вакцинація, прийом у ветеринара), додавати текстові нотатки з нагадуванням налаштовувати нагадування на певний час, отримувати повідомлення про майбутні події. Існує 2 типи створюваних подій, зелений та червоний. Одна попереджується при досягнені часу події, а друга завчасно.

Календар оновлюється автоматично та зберігає дані у файлі calendar_data.json (рисунок 4.2).


```
{
  "2025-06-03": [
    {
      "type": "event",
      "title": "на море",
      "time": "21:33",
      "reminder": "6 годин"
    },
    {
      "type": "event",
      "title": "cgtw",
      "time": "12:56",
      "reminder": "2 години"
    }
  ],
  "2025-06-04": [
    {
      "type": "note",
      "text": "Ветеринар моська на 17",
      "time": "16:45"
    }
  ],
```


Рисунок 4.2 – Файл для збереження подій та нотатків

## 4.6 Розділ «Поради»

Розділ порад надає користувачам корисну інформацію щодо догляду за собакою. Правильно доглянутий собака буде здоровим. Він структурований за категоріями, кожна з яких містить рекомендації, описи та посилання на додаткові матеріали.

Представлені такі категорії порад:

а)«Здоров'я» – профілактика, вакцинація, візити до ветеринара;

б)«Догляд за шерстю» – купання, розчісування, догляд за шкірою;

в)«Харчування» – харчування, порції, рекомендації з корму;

г)«Ігри та активність» – фізичні та розумові тренування;

д)«Виховання» – поради з виховання тварини;

е)«Безпека» – поради для безпечного сумісного життя;

Кожна категорія включає кілька корисних порад, які допомагають власникам дбати про вихованця. У розділі передбачена навігація між категоріями та швидкий перехід до порад (лістинг 4.8).

Лістинг 4.8 – Блок з порадами про виховання

```
'Виховання': {
    'tips': [
        {
            'title': 'Базові команди',
            'content': 'Навчіть собаку базовим командам для 
безпеки та кращого контролю. Починайте з простого і поступово 
ускладнюйте.',
            'link': 'https://www.akc.org/expert-
advice/training/teach-your-puppy-these-5-basic-commands/'
        },
        {
            'title': 'Соціалізація',
            'content': 'Знайомте собаку з різними людьми, 
тваринами та ситуаціями з раннього віку.',
            'link': 
'https://www.animalhumanesociety.org/resource/socializing-your-
dog'
        }
    ]
},
```

# 5 ІНСТРУКЦІЯ КОРИСТУВАЧА

## 5.1 Інструкція з використання застосунку

При запуску застосунку, може обрати один з 2 варіантів початку роботи у програмі. Натиснути на кнопку «Вибрати зареєстровану тварину», або натиснути на кнопку «Додати тварину» (рисунок 5.1).

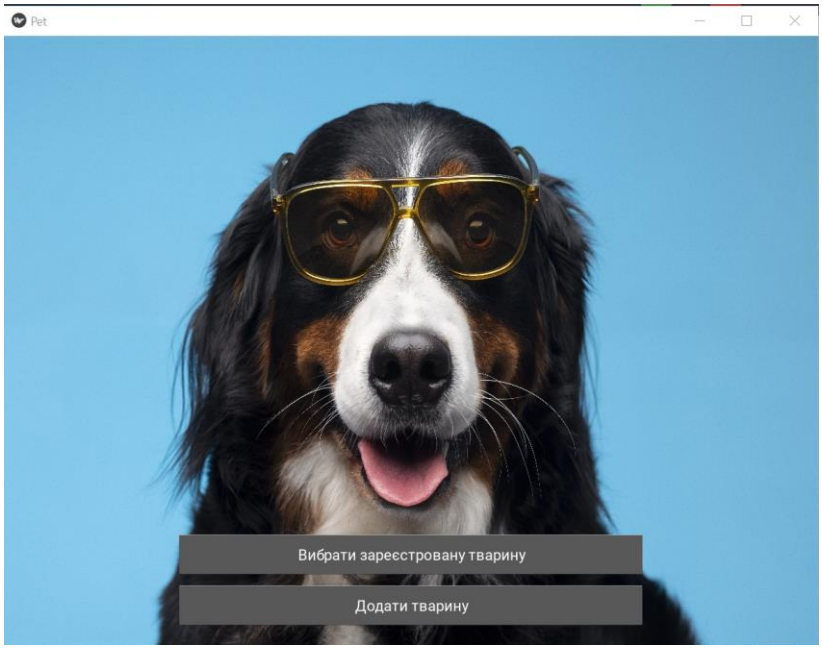


Рисунок 5.1 – Головний екран

У разі вибору першого варіанту, ми переходимо в основну частину програми та можемо обрати один з наявних профілів собак (рисунок 5.2).

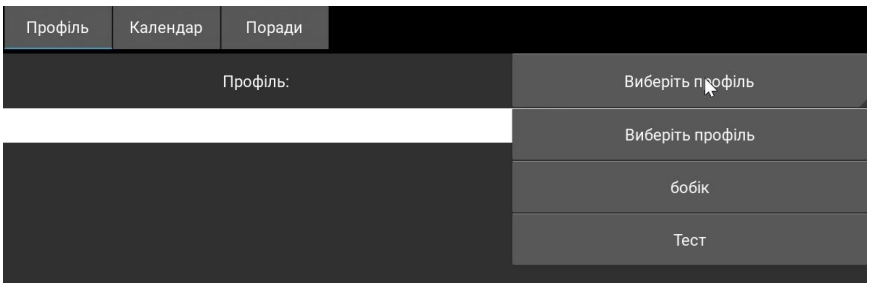


Рисунок 5.2 – Вибір профілю собаки

Після вибору профілю, його можна переглянути. Приклад профілю собаки «Тест» (рисунок 5.3).

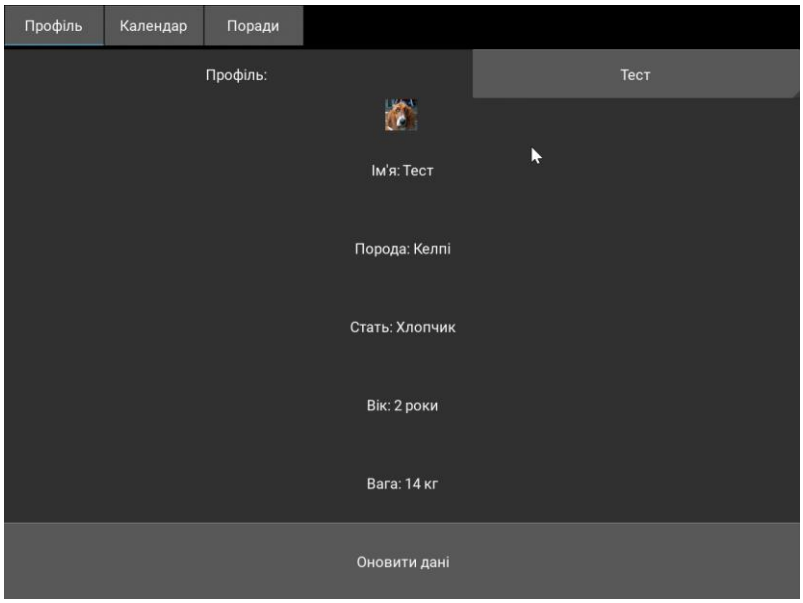


Рисунок 5.3 – Профіль «Тест»

Дані профілю можна оновити, тоді запускається та сама ж функція, як при виборі варіанту «Додати тварину». Тут можна написати ім’я, або потім, зробити фото собаки або обрати з галереї (рисунок 5.4).

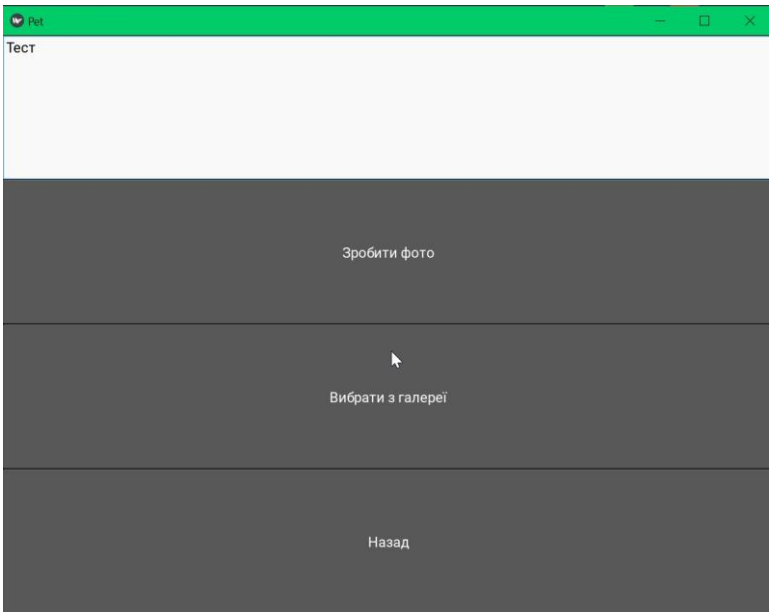


Рисунок 5.4 – Створення нового або редагування профілю

Наступне зображення приклад меню, яке з’являється при виборі пункту «Вибрати з галереї», якщо обрати варіант, то відкриється «камера» та доведеться зробити фото, фотографія буде збережена локально та використана програмою для заповнення профілю (рисунок 5.5).

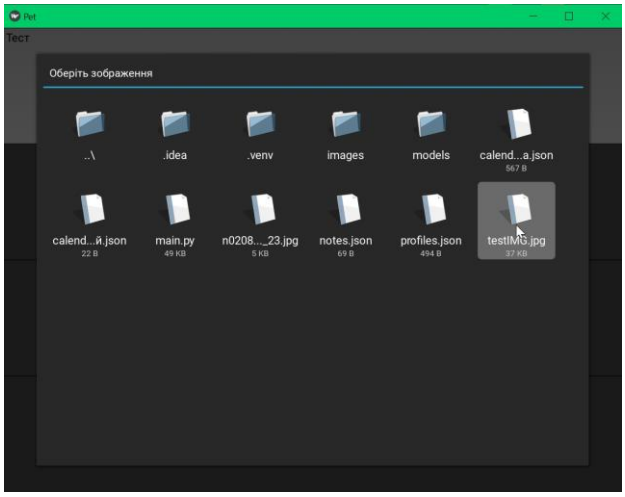


Рисунок 5.5 – Вибір фотографії зі сховища пристрою

Для заповнення «анкети» профілю необхідно заповнити поле імені, якщо це не зроблено раніше, породи, якщо порода не в списку підтримуваних моделю, вказати вік та вагу, обрати стать тварини. Після завершення необхідно натиснути кнопку «Зберегти» або «Назад», щоб відмінити зміни (рисунок 5.6).

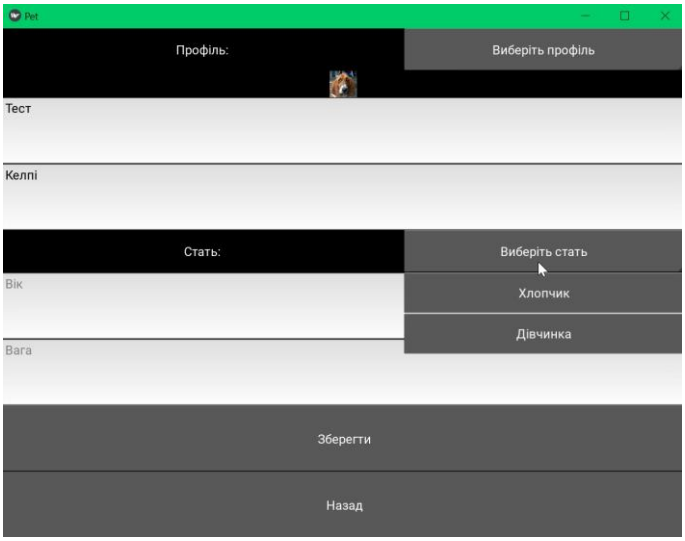


Рисунок 5.6 – Заповнення анкети профілю

Після успішного завершення профілю можна отримати доступ до інших елементів програми. Наприклад наступне поле «Календар». «Календар» дозволяю створювати «Нотатки» та планувати «Події», а також виконувати пошук ветеринарів, якщо натиснути кнопку «Знайти ветеринара поблизу». Для створення «Події» червоного кольору або «Нотатки» зеленого кольору необхідно натиснути на обраний день в календарі, місяць можна обрати за допомогою кнопок-стрілок над календарем зліва та справа (рисунок 5.7).

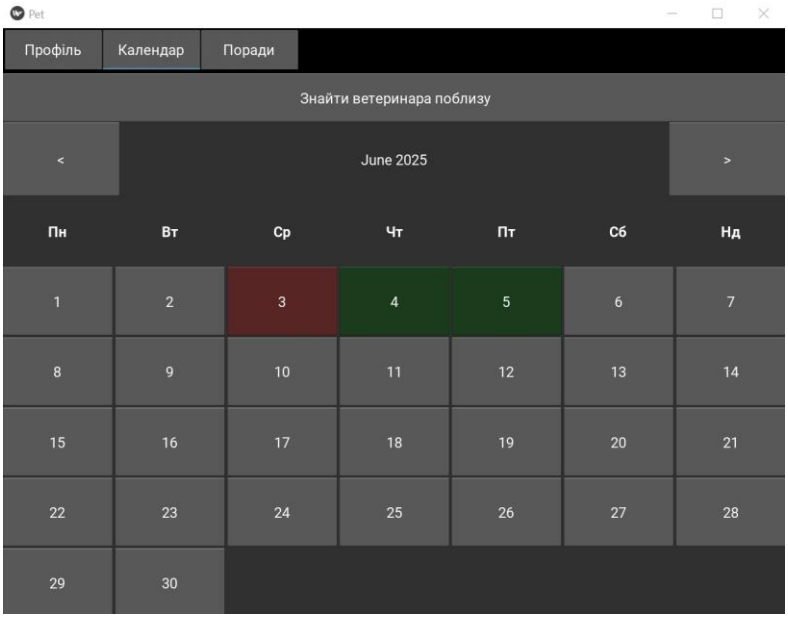


Рисунок 5.7 – Меню «Календар»

При натисканні на обраний день, можна натиснути «Створити подію» або «Створити нотатку». «Подія» – це подія, яка має бути запланована заздалегідь, та має подвійне нагадування, нагадування в час події, а також заздалегідь, максимум за день (рисунок 5.10). Рекомендується маркувати таким чином важливі події для яких необхідна додаткова підготовка, наприклад, візит до лікаря можна попередити за 10 хвилин до часу виходу та за 2 години, щоб підготувати тварину та її документи. «Нотатка» – це подія, яка є менш важливою, тому подія має одне нагадування в конкретний час (рисунок 5.11). Рекомендується маркувати таким чином менш значущі події,

наприклад, регулярний прийом ліків від шкідників або запланована прогулянка з тренуванням (рисунок 5.8).

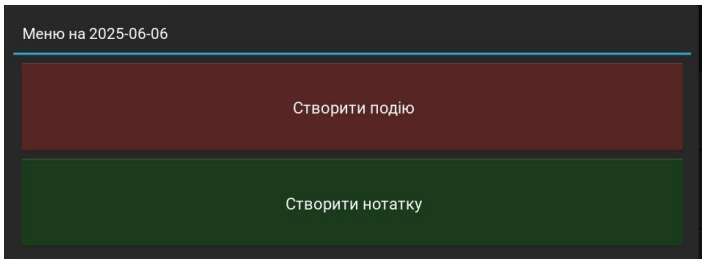


Рисунок 5.8 – Створення «Нотатки» або «Події»

Для виходу з меню створення «Нотатки» або «Події» достатньо натиснути на будь-який простір поза цим меню. Для продовження роботи необхідно натиснути на один з запропонованих варіантів, після цього необхідно заповнити анкету, та натиснути кнопку «Зберегти», а також зробити вибір в меню додаткового нагадування, натиснувши кнопку «Нагадування за» (рисунок 5.9).

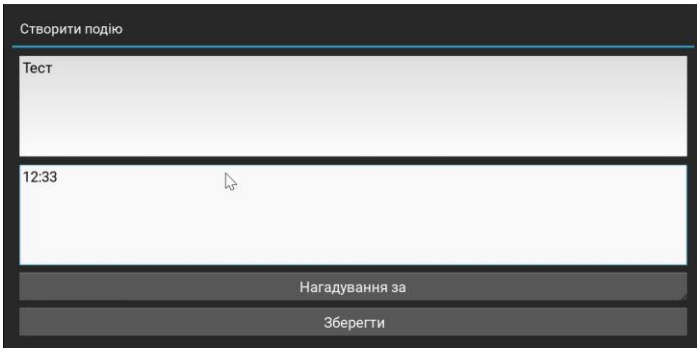


Рисунок 5.9 – Анкета для заповнення в меню створення «Події»

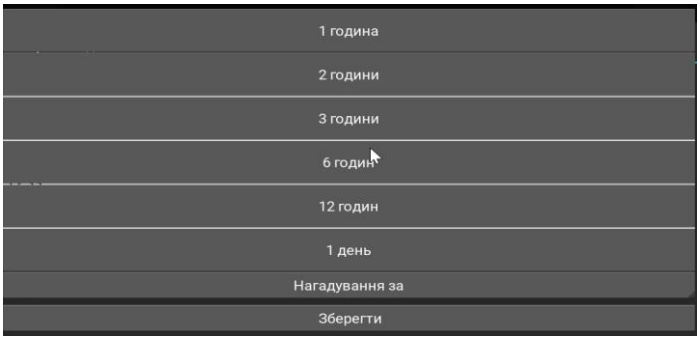


Рисунок 5.10 – Меню вибору часу для додаткового нагадування

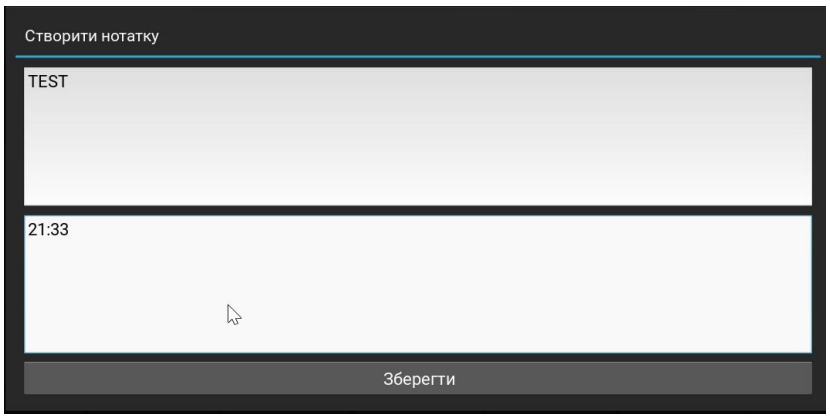


Рисунок 5.11 – Анкета для заповнення в меню створення «Нотатки»

В цьому ж меню «Календаря» можна знайти ветеринара, якщо натиснути на кнопку «Знайти ветеринара поблизу». При натисканні з'явиться можливість пошуку за допомогою адреси (рисунок 5.12). Необхідно ввести бажану адресу, тоді відбудеться пошук в радіусі 15 кілометрів від введеної адреси (рисунок 5.13).


Рисунок 5.12– Поле для введення адреси


Рисунок 5.13 – Приклад заповненого поля адреси

В цьому прикладі відбувається пошук біля вулиці Льотної, будинку 35 в місті Чернігів. Далі з'являться пропозиції ветеринарних клінік з показником рейтингу та кнопкою «Показати на карті», яку можна натиснути та відкрити Google maps для продовження (рисунок 5.14).

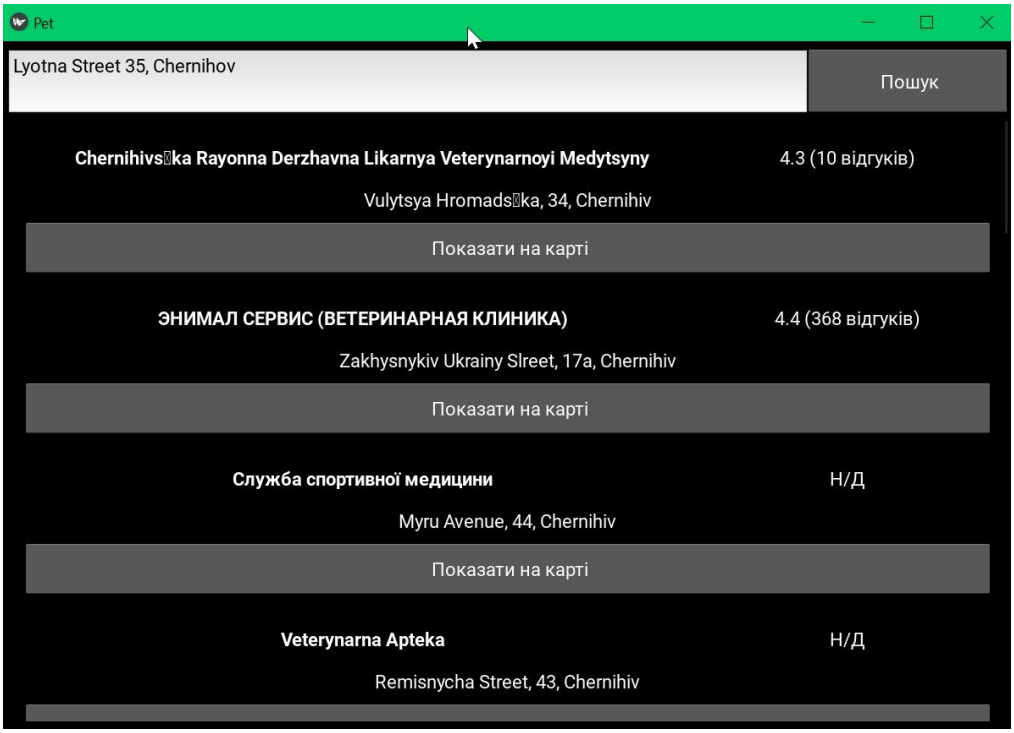


Рисунок 5.14 – Приклад результатів пошуку

Наприклад, обравши перший варіант зі списку, можна подивитись детальну інформацію про це місце. Після цього можна зателефонувати, щоб отримати запис на прийом або прийти разом із твариною особисто (рисунок 5.15). Результат пошуку знаходится поруч із заданою адресою (рисунок 5.16).

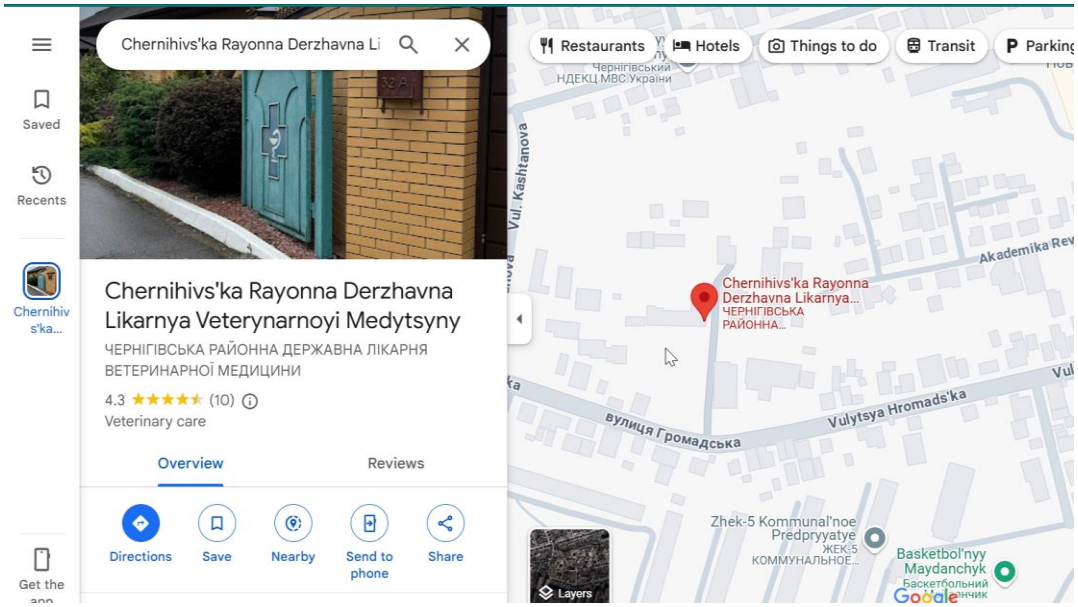


Рисунок 5.15 – Перший результат пошуку

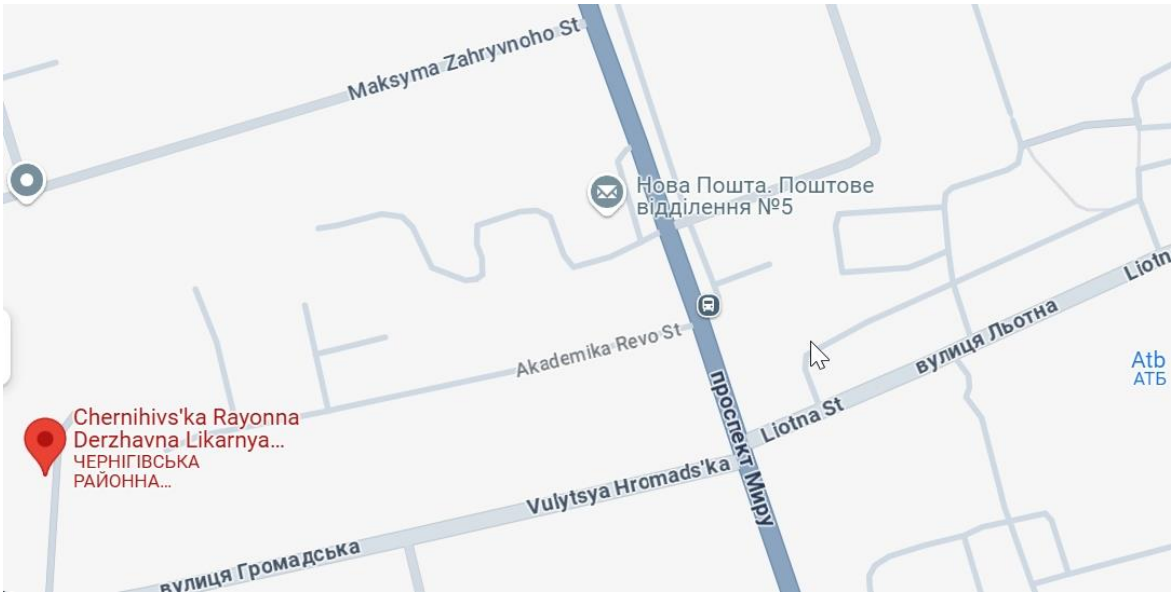


Рисунок 5.16 – Порівняння розташування вулиці Льотної та адреси лікарні

Наступний розділ це «Поради», в цьому меню можна обрати один із видів «Порад» у списку зліва. Наприклад, «Здоров'я», яке відкривається автоматично. При перемиканні між різними видами порад у списку зліва, фон програми може змінюватися відповідно до теми порад. Справа видно різні типо порад, наприклад, порада «Профілактичні огляди» та «Вакцинація» (рисунок 5.17).

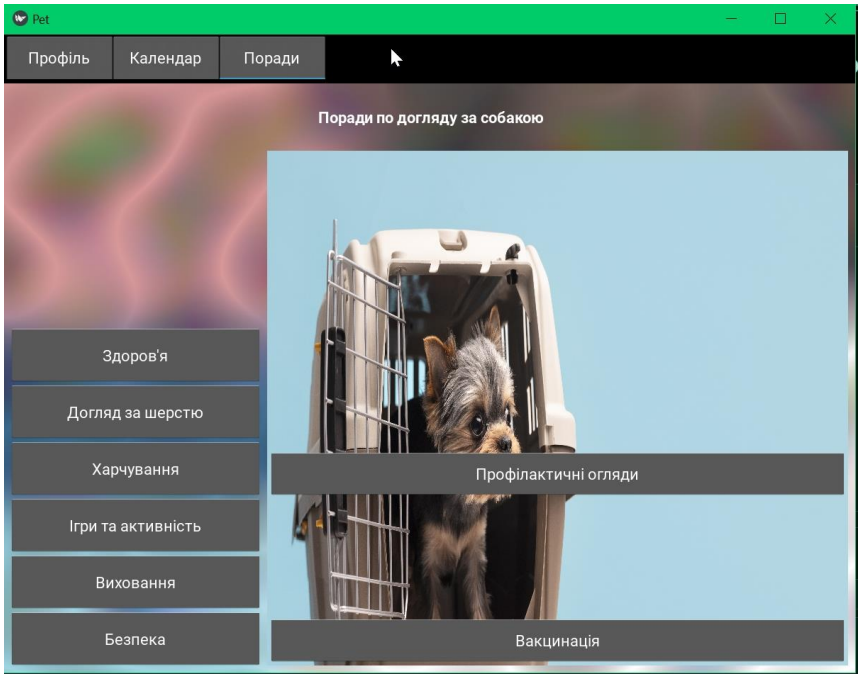


Рисунок 5.17– Поради меню «Здоров’я»

Якщо натиснути на пораду, наприклад на «Профілактичні огляди», то можна побачити зміст поради. Серед змісту можна знайти 2 кнопки «Дізнатися більше» та «Закрити». При натисканні на кнопку «Закрити», можна повернутися до списку «Порад». При натисканні на кнопку «Дізнатися більше» можна відкрити перевірений веб-ресурс із необхідною інформацією, щодо теми (рисунок 5.18).

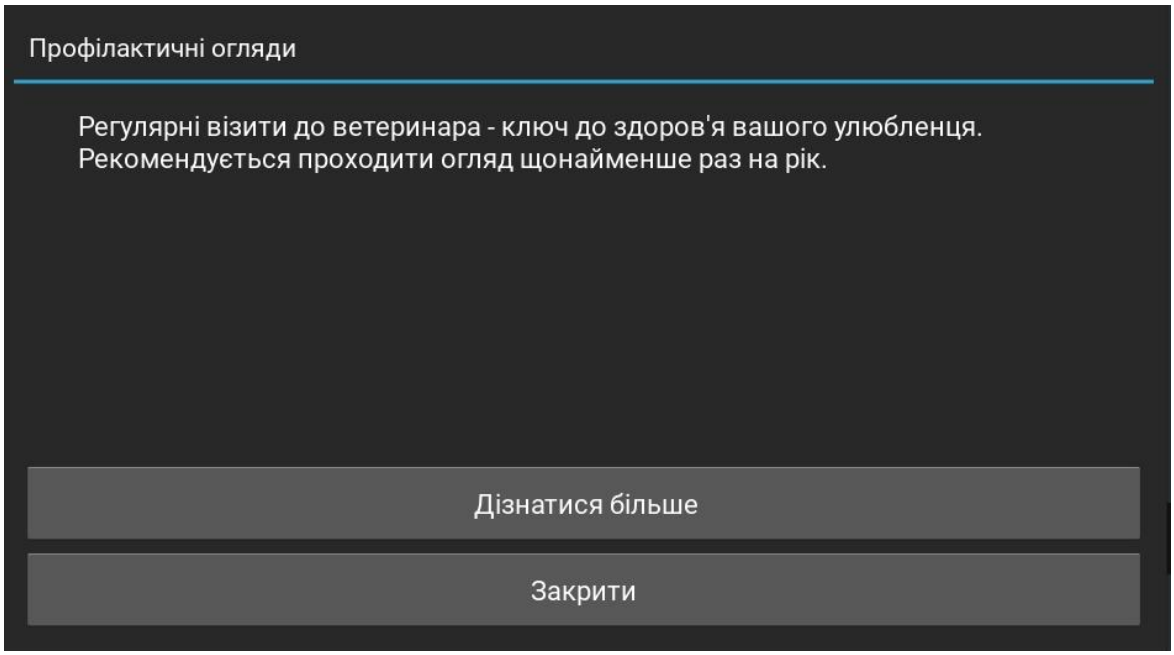


Рисунок 5.18– Зміст поради «Профілактичні огляди»

# ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено та протестовано першу версію мобільного застосунку для аналізу і контролю за станом здоров'я собак. Програмні компоненти реалізовано на сучасних мовах програмування Python та Jupiter у середовищі розробки DataSpell та PyCharm. Для повноцінного функціонування програми використано модель нейронної мережі для аналізу собак за допомогою камери. В мобільному застосунку присутній простий та мінімалістичний інтерфейс користувача на основі пакету Kyvi.

Розроблений мобільний застосунок має підтримку декількох домашніх тварин. Також мобільний застосунок може виконувати наступні функції: спостереження за фізичним станом тварини, пропонувати основні та неосновні рішення догляду за твариною, призначати нагадування важливих дат для тварини, наприклад, прийом ліків, знайти ветеринара, і інші малі приємні функції для спрощення життя разом із собакою для досвідчених заводчиків та власників-новачків.

Проєкт є робочим та виконує всі поставлені завдання, а також має перспективу подальшого розвитку та підтримки, наприклад, введення нового функціоналу або нового змісту вже існуючих розділів.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Першоджерело дослідження кількості тварин в Україні. URL: https://www.euromonitor.com/pet-care-in-ukraine/report#.

2. Petdesk (мобільний застосунок). URL: https://petdesk.com/

3. Doglog (мобільний застосунок). URL: https://www.doglogapp.com/

4. Dogo (мобільний застосунок). URL: https://dogo.app/de/

5. Whistle (мобільний застосунок). URL: https://www.whistle.com/

6. JetBrains (Поставник середовищ для розробки DataSpell та PyCharm). URL: https://www.jetbrains.com/#.

7. Посібник користувача з використання Jupyter та Python.

8. Посібник користувача з використання Conda та Python. URL: https://docs.conda.io/projects/conda/en/latest/user-guide/index.html.

9. Stanford Dog Dataset – основа набору даних для розробки. URL: https://www.kaggle.com/datasets/jessicali9530/stanford-dogs-dataset/data.

10. Cone M. The Markdown Guide. Fastly. San-Francisco: No Starch Press. 2018. 85 с.

11. R (мова програмування) – Clarke E. R Tutorial for Beginners: Learn R Programming Language. URL: https://www.guru99.com/r-tutorial.html

12. Julia (мова програмування). URL: https://julialang.org/

13. Scala (мова програмування). URL: https://www.scala-lang.org/

14. Barry P. Head First Python: A Learner's Guide to the Fundamentals of Python Programming, a Brain-Friendly Guide. 3rd Edition. Sebastopol: O'Reilly Media. 2023. 624 с.

15. Sweigart A. Practical Programming for Total Beginners. 2nd Edition. San-Francisco: No Starch Press. 2016. 500 с.

16. Anaconda (видавник програмного забезпечення).URL: https://www.anaconda.com/press/introducing-the-anaconda-ai-platform

17. Stroustrup B. C++ PROGRAMMING LANGUAGE: Special Edition. Boston: Addison Wesley. 2000. 1040 с.

18. XML-аннотації. URL: https://www.w3schools.com/xml/

19. Freeman L. C. A set of measures of centrality based on betweenness. Sociometry.Washington: American Sociological Association 1977. 35 с.

20. Sabry F. Feedforward Neural Networks - Fundamentals and Applications for The Architecture of Thinking Machines and Neural Webs. London: One billion knowledgeable. 2023. 130 с.

21. Rosenfeld E., Anderson J. A. Cambridge: MIT Press. 2000. 448с.

22. Krimpmann A. Principles of Group Accounting Under IFRS. New York: Willey. 2015. 864с.

23. Dieu Tien Bui, Pijush Samui, Ravinesh Deo, Subrata Chakraborty. Handbook of Probabilistic Models. Oxford: Butterworth-Heinemann. 2019. 590с.

24. Ziye Zhang, Zhen Wang, Jian Chen, Chong Lin. Complex-Valued Neural Networks Systems with Time Delay - Stability Analysis and (Anti-) Synchronization Control. Singapore: Springer Nature Singapore. 2022. 229с.

25. Salman Khan, Hossein Rahmani, Syed Afaq Ali Shah, Mohammed Bennamoun, Gerard Medioni, Sven Dickinson. A Guide to Convolutional Neural Networks for Computer Vision. 2nd Edition. San Rafael: Morgan & Claypool Publishers. 2018. 207с.

26. Aboul-Ella Hassanien, Mohamed Hamed N. Taha, Nour Eldeen M. Khalifa. Enabling AI Applications in Data Science. Basel: Springer International Publishing. 2020. 650с.

27. Проєкт. URL: https://drive.google.com/drive/folders/1Ul7aiw-MiaM9OPx4MJz1Pbf63EXZxI5D?usp=drive_link

28. Васильєв О. М. Програмування мовою Python. Тернопіль: Bohdan books. 2022. 504с.

29. Інтерпретатор Python. URL: https://foxminded.ua/interpretator-python/