MEIM – Instituto Superior de Engenharia de Lisboa

ADEETC – Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

MEIM
Mestrado em Engenharia Informática e Multimédia
Curricular Unit of Master Thesis

Pet adoption platform with a recommendation system based on user preferences and animal recognition in images

Duarte Domingues (45140)
(Licenciado)

Trabalho de Projeto

Orientador

[Doutor] Rui Jesus

Júri

Presidente [Doutor] Pedro Fazenda
Vogal [Doutor] Gonçalo Marques
Vogal [Doutor] Rui Jesus

Outubro, 2024

Resumo

Todos os anos, milhões de gatos, cães e outros animais são colocados em abrigos e organizações de salvamento. Apesar do seu papel crucial, estas instituições enfrentam dificuldades em lidar com a elevada procura de animais que precisam de novos lares. Este projeto aborda estes desafios através do desenvolvimento de '*Petto*', uma aplicação móvel que simplifica a adição de animais de estimação para adoção e facilita a comunicação entre adotantes e potenciais donos de animais.

A aplicação inclui técnicas de aprendizagem automática para construir o sistema de recomendação e a funcionalidade de pesquisa por imagem. O sistema de recomendação é baseado nas preferências do utilizador, obtidas através de um questionário sobre os animais de estimação desejados. O sistema de recomendação é baseado em *clustering k-means*, combinado com *sentence embeddings* derivados de um modelo *sentence transformer* pré-treinado. Isto permite agrupar animais de estimação com base no conteúdo semântico das suas características. A aplicação também fornece uma funcionalidade de pesquisa por imagem que permite aos utilizadores carregarem uma imagem de um animal e visualizarem animais de estimação semelhantes disponíveis para adoção. Esta funcionalidade baseia-se na extração de vetores de características de um modelo de redes neurais convolucionais *Keras*, pré-treinado, para efetuar cálculos de semelhança.

Para treinar e testar as funcionalidades de aprendizagem automática foi construido um conjunto de dados. Está dividido em ficheiros CSV que contêm informações sobre os atributos dos animais e em imagens. No total, o conjunto de dados contém dados de 10.000 cães, 9.000 gatos, 500 roedores, 200 aves e 200 coelhos.

A experiência do utilizador na aplicação foi avaliada através de um questionário. A aplicação obteve uma pontuação de **88** no questionário *System Usability Scale (SUS)*, que mede a usabilidade geral da aplicação. O sistema

de recomendação foi avaliado utilizando a métrica *Precision*, com 5 exemplos de teste, cada um com 10 recomendações, obtendo um valor de **0.94**. A pesquisa baseada em imagens foi avaliada utilizando *Precision* com 5 exemplos de imagens diferentes, com 9 imagens semelhantes para cada uma, obtendo um valor de **0.93**.

Palavras-chave: Aplicação Móvel, Sistema de Recomendação, Pesquisa por imagens, adoção de animais.

Abstract

Each year, millions of cats, dogs, and other animals are placed in shelters and rescue organizations, with many ultimately being euthanized. Despite their crucial role, these institutions struggle to handle the high demand for animals needing new homes. This project addresses these challenges by developing 'Petto,' a mobile application that simplifies adding pets for adoption and facilitates communication between adopters and potential pet owners.

The application incorporates machine learning techniques. It utilises a content-based recommendation system, based on user preferences derived from a questionnaire about desired pets. The recommendation system is based on k-means clustering, combined with sentence embeddings derived from a pre-trained sentence transformer model. This allows for grouping pets based on the semantic content of their characteristics. The application also provides an image search feature, allowing users to upload an image of an animal and view similar pets available for adoption. This functionality relies on extracting feature vectors from a pre-trained Keras Convolutional Neural Network model to perform similarity calculations.

A dataset was built to train and test the machine-learning functionalities. It is divided into CSV files containing information for pet attributes and images saved as JPG files. Overall, the dataset contains data on 10,000 dogs, 9,000 cats, 500 rodents, 200 birds, and 200 rabbits.

The user experience on the application was evaluated through a questionnaire. The application achieved a score of **88** for the System Usability Scale questionnaire (SUS), measuring the overall usability of the application. The recommendation system was evaluated using the Precision metric with 5 different examples, each with 10 recommendations, obtaining a value of **0.94**. The image-based search was measured using Precision with 5 different image examples, each with 9 similar animals, achieving a score of **0.93**.

Keywords: Mobile Application, Recommendation System, Image Search, Animal Adoption.

Acknowledgments

I want to express my gratitude to everyone who supported me throughout my academic journey, helping me overcome challenges and grow as a person. A heartfelt thank you to the professors, staff, and my colleagues of Instituto Superior de Engenharia de Lisboa.

A special thanks goes to Professor Rui Jesus, whose guidance and support were invaluable in the development of this project.

I also want to extend my sincere thanks to my family and friends for always supporting and believing in me.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Animal adoption is a relevant issue, primarily managed by animal shelters and rescue organizations. Despite the importance of these institutions, they can not deal with the high demand for animals requiring a new home. Therefore, only a small number of these animals end up being adopted. Most existing pet adoption applications struggle with common issues like outdated and cluttered screens, and do not allow users to add their own pets for adoption or efficiently interact with potential adopters.

This project consists of developing a mobile application named 'Petto' that integrates machine learning and user experience design principles to facilitate the pet adoption process. The application includes five different species of animals: dogs, cats, rabbits, birds and rodents, each characterized by a set of attributes like color, age and size. It utilizes machine-learning techniques in two ways: (1) Implementing a recommendation system based on user's pet preferences and (2) constructing a search mechanism based on similar images.

The recommendation system is essential to guarantee accurate matches between pets and users. It uses content-based filtering, based on the user's preferred pet characteristics, gathered through a straight-forward questionnaire, to recommend pets to the users. The system uses the K-means clustering algorithm to group pets with similar attributes into clusters within the dataset. The clustering model predicts the pet cluster that most closely resembles the user's pet preferences. Within the predicted cluster, the system

calculates the pets most similar to the user's pet preferences using a similarity measure and recommends them to the user. K-means is combined with sentence embeddings derived from a pre-trained sentence transformer model. This enables the k-means model to group the animals based on numerical features derived from text, capturing the semantic content of the pets' characteristics.

The search mechanism based on similar images identifies pets from the application dataset that closely resemble an image uploaded by the user. It utilizes a pre-trained ResNet50 model to extract feature vectors from the pet images in the dataset. These vectors contain numerical representations of key features such as shapes, colors, and textures of the images. By calculating similarity between these vectors and the vector of the image uploaded by the user, the system finds the most similar pets from the dataset.

The project's GitHub repository is available here: `https://github.com/DuarteDomingues/petto_mobile_application`

1.1 Scope and Objectives

The primary outcome of this project is the implementation of a mobile application centered on pet listings with detailed information about the animals and the users. The application offers intuitive navigation and an easy-to-use interface. It supports search through multiple filters, including geo-location and image similarity, and provides real-time messaging. The mobile application uses React Native for the frontend, and Google Cloud Platform Firebase for backend functionalities like database management, authentication, and storage. Another significant outcome is the creation of a pet recommendation and image search API using Flask, Python, and TensorFlow, which interacts with the application. This API is deployed on the Google Cloud Platform using Cloud Run.

The results of this project are evaluated based on user experience. This is assessed using a feedback-based questionnaire, including the System Usability Scale (SUS), to collect user insights. The image-based search and

recommendation system effectiveness is measured through various tests and evaluation metrics.

1.2 Contributions

The contributions of this project include the development of the mobile application, the recommendation system, image-based search, and the user experience evaluation.

- **Mobile application**: A key contribution is the development of a mobile application to facilitate the pet adoption process, focusing on user interaction and personalized recommendations. This application is built using React Native, allowing availability on multiple platforms.
- **Recommendation system**: Another important contribution is the implementation of a content-based recommendation system. This involves creating and testing a k-means clustering model integrated with sentence embeddings derived from a pre-trained sentence transformer model.
- **Image-based search**: An additional contribution is the implementation and evaluation of a search mechanism based on similar images. This incorporates the use of feature vectors extracted from a trained convolutional neural network for similarity calculation.
- **User experience evaluation**: This contribution involves using user experience design principles for the creation of an application. It encompasses creating wireframe prototypes, developing the visual aspects of the application, and conducting user evaluation through a questionnaire.

1.3 Document Organization

The remainder of this thesis is divided into six chapters, as follows:

2. **Related work and Context**: This chapter reviews similar works and literature on various concepts related to the project.

3. **System Analysis**: The third chapter describes the system architecture and the methodology used throughout the project.

4. **Recommendation System and Image Search**: This chapter focuses on the development of the recommendation system based on user's pet preferences and the search mechanism based on similar images.

5. **Implementation**: Chapter five describes the implementation of the application.

6. **Results and Evaluation**: The sixth chapter evaluates the results obtained with the tests performed.

7. **Conclusions and Future Work**: This chapter concludes the project with a discussion of the effectiveness of the application and suggestions for future work.

Chapter 2

Related work and Context

This chapter reviews similar works and literature on various concepts central to the project. It focuses on the integration of artificial intelligence in pet adoption, the development of recommendation systems, and the application of user experience design principles.

2.1 Artificial intelligence in pet adoption apps

According to research on animal adoption [1], many millions of cats and dogs are installed annually in animal shelters, with most ending up being euthanized. One proposed solution to the pet adoption problem is the use of technology and artificial intelligence.

A pet adoption system [2] implemented and analyzed different artificial intelligence methods, respectively:

- **Question-based Recommendation System**: This system works by asking users simple questions, like their activity level and living space. Based on the user's answers, the system recommends the most suitable animals, predominantly based on the animal breed. This type of approach solves the new user problem that conventional content-based recommendation systems face. This facilitates the process for new time users to choose a desired pet.

  A similar approach was used in our project to obtain the user pet

preferences. A questionnaire was developed with different questions related to the type of pet the user is interested in. Based on these responses, a user preference profile is constructed.

- **Pet Detection System**: It is also developed a pet detection system used to classify pets as either dogs or cats. The model behind this system involves the use of convolutional neural networks. This process of using animal images to obtain meaningful information is relevant in a pet adoption application.

  In our project, pet detection and image processing was used as a search mechanism based on an image inserted by the user. This allows the user to access different pet listings based on an image of an animal they are interested in.

2.2 Recommendation systems

In a pet adoption application, the recommended system is an integral part to guarantee correct matches between adopters and pets. These systems allow enterprises to make individual targeting strategies, while fulfilling the customer needs by offering relevant products [3]. Recommendation systems are broadly categorized into two main groups [3]: content-based filtering and collaborative-based filtering.

2.2.1 Content-based filtering

In content-based filtering, recommendations are made based on item information and the user's profile. A user profile is built based on the data obtained from the user while it interacts with the system, directly or indirectly (Figure 2.1). This profile is established initially after the user creates an account. This type of system needs limited information because only the user data is required to perform recommendations.

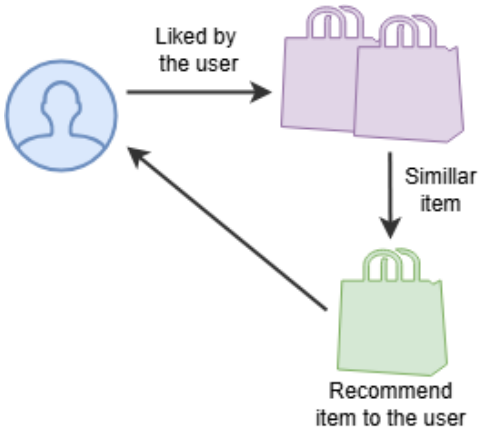


Figure 2.1: Content-based filtering

These systems work by comparing the user profile with available item profiles to recommend similar items. One of the most common techniques associated with content-based filtering is TF-IDF (Term Frequency-Inverse Document Frequency) [4].

TF-IDF is a numerical statistic used to reflect the importance of a word (term) inside a document in a collection of documents. It uses a measure called Term Frequency (TF), which measures how often a term appears in a document. The more frequently a term t is present in a document d, the higher its TF value, as shown in the following formula:

1. $$\text{TF}(t, d) = \frac{\text{number of times } t \text{ appears in } d}{\text{total number of terms in } d}$$

Inverse Document Frequency (IDF) is then used to measure the importance of a term across the entire document collection. It weighs down the importance of the most frequent terms, while highlighting the less frequent terms, indicating that these are more unique and informative. N represents the total number of documents across the collection, d refers to the individual documents in the collection, and t is the term being measured, as presented in the following formula:

2. $$\text{IDF}(t) = log\frac{N}{1 + df}$$

Finally, a TF-IDF score is calculated that indicates the term's importance within a document.

A study demonstrated the application of TF-IDF in content-based filtering for product recommendations using textual descriptions [5]. By analyzing 258 product codes spanned across eight categories with 33 keywords, the system obtained a relevant product recommendation accuracy of 96.5%, showcasing the effectiveness of content-based recommendations with TF-IDF for textual product descriptions. Although this study focused on pen products in an online setting, this procedure can also be applied to animal adoption. It can be used in a pet adoption application to suggest pets based on key attributes of interest to the user, using pet descriptions.

2.2.2 Collaborative filtering

Collaborative filtering differs from the previous approach by focusing on items that other users have liked rather than just matching items to a user profile. This type of filtering considers the similarity between users, instead of the similarity to an item. These systems work by collecting direct feedback from users on items and calculating similarities between users to recommend new items (Figure 2.2) . There are different approaches to collaborative recommendations [3]: user-user collaborative filtering and item-item collaborative filtering.

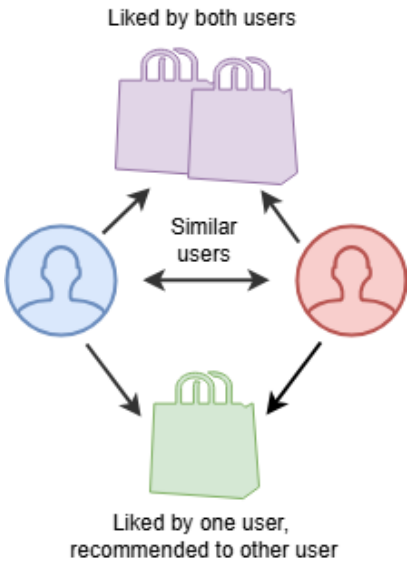


Figure 2.2: Collaborative filtering

2.2.3 Advantages and disadvantages

Content-based filtering has two major shortcomings [3]: the new user problem and overspecialization.

The new user problem is a product of lack of data for new users. Initially, it is hard to perform meaningful recommendations due to the user not having a strong preference for any type of item.

The overspecialization problem is a result of the system being able to only recommend items based on the user profile, resulting in no new or diverse items being suggested. A way to attenuate this problem is through the addition of randomness and genetic algorithms [6].

Another limitation of content-based filtering is that it often includes only the textual data of the items being considered, making it difficult to include multimedia information like images in these systems. This is relevant to our project, because it is important to recommend pets to users based on images as well.

While collaborative filtering allows the usage of collaborative user preferences to create new recommendations, content-based filtering was used for our system. This is because content-based filtering allows us to focus more on specific needs and characteristics of the animals rather than the majority preference of users.

2.3 Clustering in recommendation systems

Clustering [7] is an unsupervised learning technique that consists of grouping a set of data objects based on their attributes, enabling the discovery of hidden patterns within a dataset. In [8], it is discussed how the use of clustering techniques can be used as a preliminary step in different recommendation system designs. In clustering, each item is assigned into a cluster, with items in the same group being more likely to be similar. The paper suggests clustering is frequently used to group users into clusters, based on their similarity. This approach is more efficient than using traditional similarity-

based measures to compare users.

One of the most used clustering algorithms in recommendation systems is k-means [9]. This is a centroid-based clustering algorithm, which means that clusters are represented by a central vector. In this algorithm it is necessary to define the k value, which represents the initial number of clusters.

In our project, k-means clustering was used to group pets with similar attributes in clusters. Afterward, it is also used to predict the pet cluster most related to the user pet preferences vector. Inside the predicted cluster, the most similar pets to the user vector are calculated using a similarity measure and then recommended to the user.

2.3.1 Attribute-based item clustering algorithms

In item clustering, in an application with products, algorithms operate mostly based on the descriptions or the attributes of items. In our project, we chose to base clustering on attributes. This approach was deemed suitable for pet item clustering because attributes such as color, age and breed provide enough distinct and meaningful information to group pets based on similarity. The use of attributes instead of descriptions allows us to base the clustering algorithm on more standardized and consistent information.

2.4 Sentence Transformers

In Natural Language Processing (NLP) [10], sentence transformers [11] are an efficient way for encoding sentences into high-dimensional vectors with fixed size. These vectors capture the semantic meaning behind sentences. This is useful for tasks such as text clustering, translation, and sentence similarity.

The evolution of sentence transformers progressed from Recurrent Neural Networks (RNNs) [12], through increasingly advanced models, leading to their development.

2.4.1 Recurrent Neural Networks

Initially RNNs and their variant, Long Short-Term Memory networks (LSTMs) [13] were the most used architectures for natural language processing tasks, such as translation. These models work in two phases:

- **Encoding phase**: The encoder processes the entire input sentence sequentially, one word at a time. At each step, it updates a hidden state vector, that captures information from the current word and the previous hidden state. After processing the last word, the final hidden state also referred to as the context vector is generated, representing the entire input sequence. Additionally, a sequence of hidden state vectors is also generated, representing the input sequence at different stages.
- **Decoding phase**: The decoder generates the output sequence one word at a time. It starts with the context vector as its initial hidden state. With this context vector, the decoder generates the first word of the output sequence. For subsequent words, the decoder uses the previously generated word and the current hidden state to predict the next word in the sequence. This process continues until the entire output sequence is generated.

The challenge [12] with this approach is that condensing all textual information into a single context vector can lead to information loss, especially in long sequences.

2.4.2 Attention mechanism

To address the challenges associated with RNNs, Bahdanau et al. introduced the attention mechanism [14]. Attention allows the model to place its focus on specific portions of the input sequence when it is generating each word in the output. The encoder in the attention mechanism, produces a sequence of hidden states, one for each word contained in the input sentence. The decoder now instead of relying on a single context vector, calculates attention weights for each hidden state, allowing it to determine which parts of the input to focus on for each generated word. This brought significant improvements in performance, specifically in longer sentences.

2.4.3 Transformer model

In 2017, Vaswani et al. introduced the Transformer model in their paper *Attention Is All You Need* [15]. It eliminated the need for the traditional sequential processing of RNNs by relying entirely on the attention mechanism. It focuses on different portions of the input all at once through multiple layers of attention. This allows the model to analyze various aspects and relationships within the input data. In addition, after each attention layer, feed-forward networks are applied to refine the processed information. This model allowed improvements in performance and efficiency, becoming the basis for models like BERT [16] and GPT [17].

Building on the transformer model, sentence transformers were developed to improve the ability to calculate similarities in texts and generate new content. Sentence transformers built on pre-tained models like BERT and GPT offer significant versatility and performance for tasks such as text clustering and translation. For these reasons, they are well-suited for the clustering model based on pet information developed in this project.

2.5 Convolutional Neural Networks (CNNs)

Convolutional neural networks [18] are a form of neural networks commonly used in image processing due to their ability to detect patterns in images. They consist of an input layer, multiple hidden layers, and an output layer. The hidden layers include convolutional layers, pooling layers, and fully connected layers. Each layer builds on the output of the previous layer to extract features and patterns.

The convolutional layers apply a filter to the input image, creating feature maps that detect essential features and patterns in the image. These layers use a kernel that moves across the image, calculating a dot product between the kernel's weights and the image's pixel values under the kernel region. This results in feature maps that indicate the presence and intensity of specific features, which are then input to the subsequent layers. Initial filters detect basic features, such as lines and edges. As the network progresses, subsequent filters combine the previous basic features to find more complex

patterns like shapes.

Between these layers, pooling layers are used to reduce the dimensionality of the input data, reducing the overall number of parameters and complexity. This enhances the ability of the model to generalize, which helps prevent overfitting. Finally, fully connected layers aggregate the features from previous layers and map them to specific classes. In these layers, every neuron is connected to each neuron in the next layer, providing the information for the network to perform the final classification (Figure 2.3)

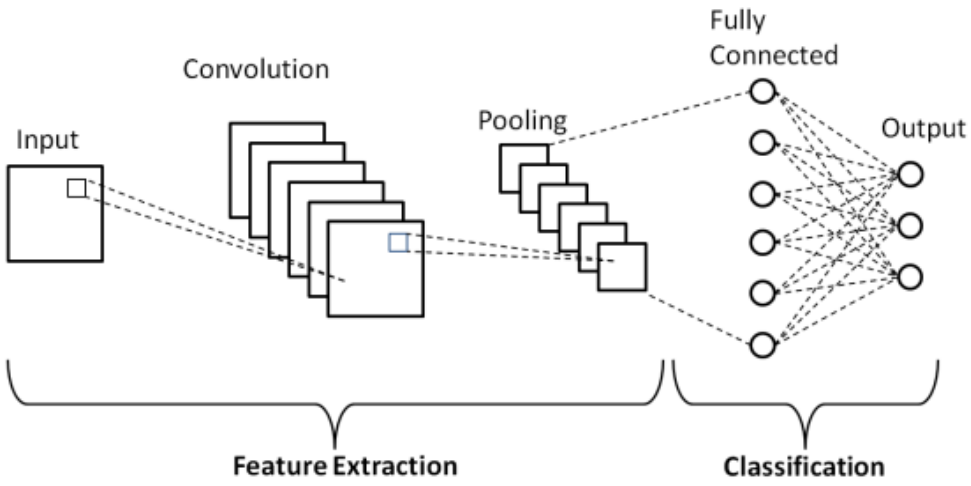


Figure 2.3: CNN Model (source: `https://medium.com/@brtracy1984/pneumonia-diagnosis-using-machine-learning-558aaf416acc`)

2.6 Similar image recommendation

A similar image recommendation system for online shopping [19], utilized the power of convolutional neural networks to calculate similar images based on an input image. The model is divided into two phases: classification and recommendation. Initially, it classifies the image in one of 20 possible product classes. Then, the input vector of the last fully connected layer is used as a feature vector to perform similarity calculation. This feature vector is compared with the feature vector of images of the same class from the dataset to find the closest products.

In our project, we adopt a similar approach for image search, employing neural networks to extract feature vectors from pet images and using similarity calculation to obtain similar images. In contrast, we use a pre-trained ResNet50 model [20] to extract the feature vectors of the images, instead of training a model for classification from scratch. ResNet50 is a fined-tuned model pre-trained on a huge dataset called ImageNet. The usage of a pre-trained model is useful due to limitations on data size and computational resources. Although it has the disadvantage of not classifying the images, it allows us to use the pre-trained model solely to obtain the feature vectors. This way it is necessary to compare the input feature vector with the feature vector of all images in the dataset, possibly slowing down the system.

The work by Chen and Mall [19] analyzed the results of different similarity measures for recommendation, respectively L2 distance score and cosine distance score. The cosine similarity score outperformed the L2 similarity score.

2.7 Evaluation metrics in recommendation systems

Various characteristics and prediction techniques used in recommendation systems were examined in [21]. It is mentioned that the type of evaluation metrics depends on the filtering technique used. These metrics are classified as statistical or decision support accuracy metrics. Statistical accuracy metrics evaluate the accuracy of filtering calculation by comparing the predicted ratings with the user ratings.

In our recommendation system there are no ratings available, so these techniques cannot be directly applied. However, decision support accuracy metrics deal with prediction in a binary faction, distinguishing correctly recommended items from incorrectly recommended items. Two of these metrics are Precision and Recall. Precision measures the proportion of items that are relevant to the users, while Recall measures the proportion of relevant items that are present in the set of recommended items. ROC curves are also

useful when analyzing the performance of recommendation algorithms. Precision was used to evaluate the effectiveness of our recommendation system and image search. The definitions of Precision and Recall are as follows:

1. $$\text{Precision} = \frac{\text{Correctly Recommended items}}{\text{Total Recommended items}}$$

2. $$\text{Recall} = \frac{\text{Correctly Recommended items}}{\text{Total relevant Recommended items}}$$

2.7.1 Similarity metrics

To compare the data objects in our project, it is necessary to use similarity metrics [22]. These metrics are useful for comparing the user pet preferences vector to the most similar pets in the dataset. These metrics are also used to calculate the most similar pet images to an input image.

Euclidean distance: To calculate the Euclidean distance, each attribute value of one data object is subtracted from the corresponding value of the other data object, and the differences are squared and summed. This similarity metric requires the objects to have numerical attributes, and for it to work well, the values need to be normalized. In the formula x_i and y_i represent the values of two data objects at position i, and n is the total number of data objects, as presented in the following equation:

1. $$\text{Euclidean Distance} = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

Cosine similarity: To calculate cosine similarity between two data objects, both their attributes must be represented as vectors. The similarity is measured by the cosine value of the angle between these vectors, represented in the following formula as A and B:

1. $$\text{Cosine Similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||\,||\mathbf{B}||}$$

2.8 Simillar applications and case studies

2.8.1 UX case study - Pet adoption app

A case study by Rhea Pherwani's [23] focuses on the creation of a user-friendly pet adoption application designed to connect people with pets that match their lifestyles. This case study addresses different problems and solutions in pet adoption.

- **Challenge identification**: The case study analyzes pet adoption challenges, particularly the overcrowding of animal shelters and the urgent need for animals to be adopted.
- **Objectives**: Rhea Pherwani's objectives include matching potential adopters with pets based on lifestyle behaviors, designing an informative application that displays nearby shelters, and ensuring the user experience is as simple and comfortable as possible. In our project, we also have the objective of displaying pets based on location and ensuring a simple and straightforward user experience.
- **User Experience and Design**: The study emphasizes the importance of user-friendly design and providing easy access to user information. This was also fundamental in the development of our application.
- **Research**: Rhea Pherwani's research pointed out common issues in existing pet adoption applications, such as outdated and cluttered screens.
- **User Surveys**: The case study performed use studies and interviews to gather feedback on people's preferences on the pet adoption process. It was gathered that 50% of the users would prefer to adopt through a mobile application and that 80 % of people looking to adopt were previous pet owners. This was important information when choosing the platform for our application and target audience.
- **Wireframes**: The use of wireframes in the study was crucial to help solidify design concepts and to iterate constantly based on user's needs. Wireframes serve a similar purpose in our project.

- **Features and functionalities**: The case study highlights the importance of functionalities such as search filters, chat support, questionnaires to find the most suitable pet, and the inclusion of multimedia elements. Our application integrates these elements including a recommendation system and other functionalities.

Overall, this case study served as an inspiration for the design of our application, helping to gather information on the possible problems and solutions of a pet adoption application. It was also helpful to define certain features and functionalities for our application.

2.8.2 Petfinder – Pet adoption platform

The primary online platform for pet adoption is Petfinder [24], it launched in 1996 and has a website and a mobile application. Petfinder hosts many animals from shelters and rescue organizations in North America. It has comprehensive pet profiles, allowing robust search and filtering based on different attributes like gender and breed. It also has location-based search which allows users to find pets near their location.

Our project offers similar functionalities, such as detailed profiles, different search options, filtering capabilities, and geo-location searches. We drew inspiration from Petfinder to define pet characteristics and species. Petfinder uses a pet quiz to suggest pets to the user, which was also an inspiration for our project.

However, unlike Petfinder, our focus is on user interaction. Allowing users to not only adopt pets, but also list their pets for adoption, while offering simple communication tools to facilitate this process, such as a real time chat. Additionally, our platform provides a user-friendly interface, designed with UX principles.

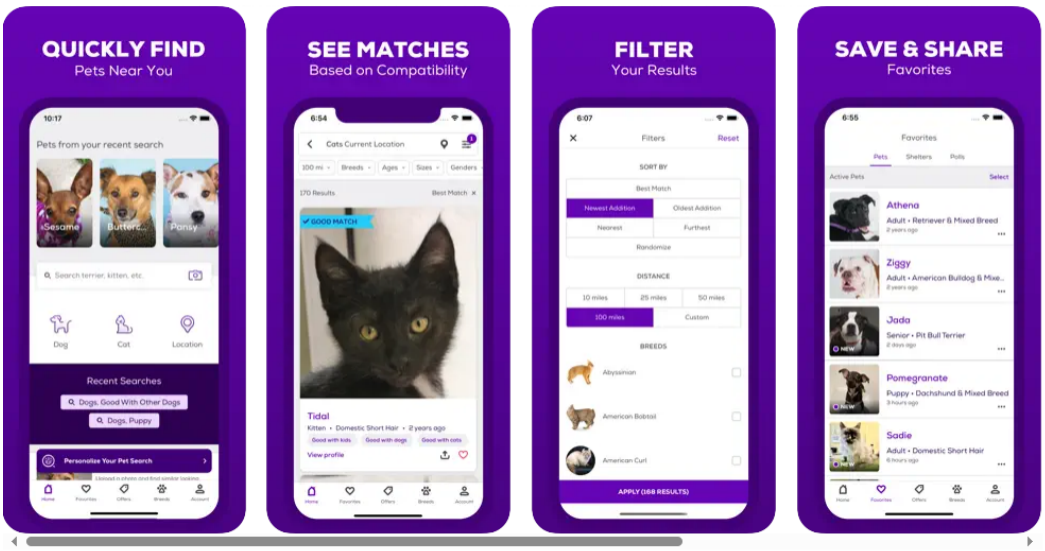


Figure 2.4: PetFinder interface (source: `https://www.petfinder.com/`)

2.8.3 PetMatch

PetMatch [25] is a mobile application that enables users to find similar cats or dogs for adoption by selecting an image. It also offers a wide variety of pet listings, helping users find their ideal pet. This application inspired us to add a search feature based on animal images to our application. While PetMatch focuses on pet's images, our project also emphasizes recommendation based on animal attributes.

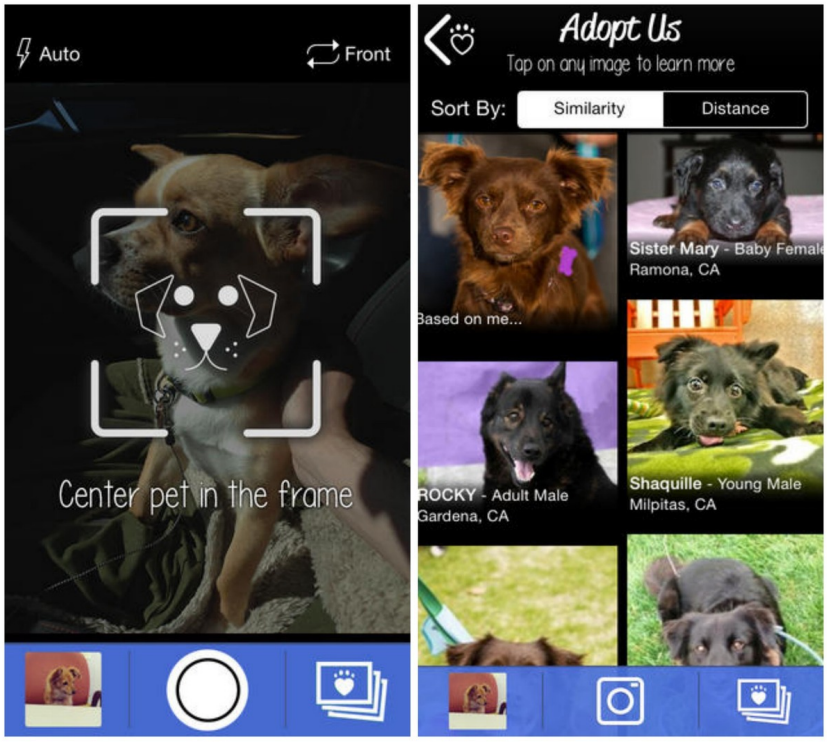


Figure 2.5: PetMatch interface (source: `https://mashable.com/archive/petmatch-app`)

Chapter 3

System Analysis

This chapter covers the system analysis, detailing the project requirements, usage environment, system architecture, technologies utilized, and the design of the application prototype.

3.1 Product placement

This system aims to modernize the pet adoption process. It stands out by targeting potential pet owners looking to re-home their pets, and potential adopters looking for an efficient way to adopt. This application targets users that value convenience and have some level of technological familiarity. To ensure broad accessibility, the application is designed to be intuitive and enjoyable to use.

3.2 Usage Environment

To reach the widest audience, the basis of the system is a hybrid mobile application compatible with multiple operating systems, including Android and IOS. The application is intended to be available in both the Apple Store and Google Play.

3.3 System requirements

The system requirements are presented in Table 3.1. These include specifications for the application, the recommendation system, and administrator

management, including both functional and non-functional aspects.

Table 3.1: System requirements

System Requirements	**Description**
Interactive application	User friendly interface, intuitive navigation
Performance	Fast load time, efficient handling of data
Security	Secure data access, protection against unauthorized access
Analytics	Reports on user engagement and adoption statistics
Pet recommendations	Meaningful pet recommendations, based on user preferences
Detailed pet profiles	Inclusion of textual and multimedia content
Detailed search	Multiple filtering options (age, breed, etc.)
Geo-coordinates	Integration with geo-location services
User management	User authentication and profile editing
Pet management	Add, update, and remove listed pets
Real time messaging	Real time chat functionality between users
Availability	Compatibility with IOS and Android
Admin dashboard	Admin dashboard to manage users and monitor system

3.4 User interaction

The application allows users to perform a variety of activities. There are two types of users: registered users and guest users. Both types can perform similar activities like browsing through the available pets and perform searches, but guest users do not have a user profile and cannot add pets to the system.

An overview of activities the users can perform in the application is presented in Figure 3.1.

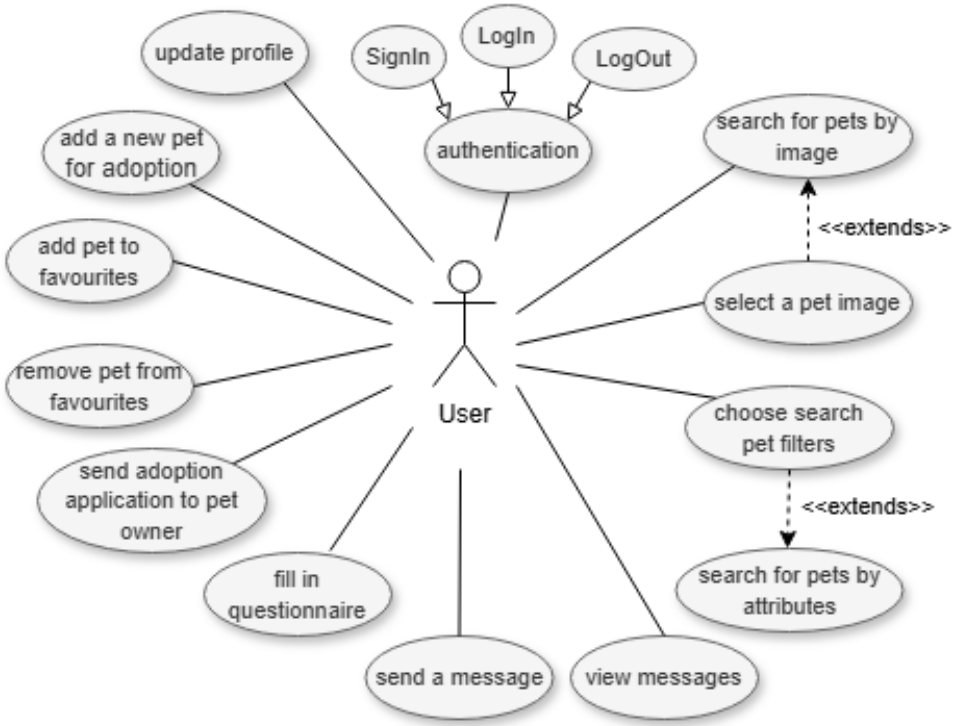


Figure 3.1: User system interaction

3.5 Animal characteristics

An important step of the system analysis is defining the animal characteristics. This step is crucial to ensure that the pets in the application are correctly represented and provide useful information for search and recommendation purposes. Notably, some attributes vary between animal type, such as fur type. To gather this information, we reviewed similar applications and publicly available animal shelters datasets, as referenced in [26] and [27]. This research provided insight into common attributes used in the pet adoption process.

Table 3.2 represents the defined animal characteristics, divided into four categories.

Table 3.2: Animal characteristics

Category	Attribute	Description
General Information	Species	Type of animal
	Breed	Specific animal breed
	Age	Animal age(baby,young,adult,senior)
	Gender	Gender of the animal
Physical Information	Size	Animal size(small, medium, large)
	Color	Color of the animal fur or skin
	Weight	Animal weight
Health Information	Health status	Whether animal needs special needs
	Sterilized	Whether animal is sterilized
Behaviour Information	Activity level	Activity level(low, medium, high)
	Living situation	Type of home appropriate

3.6 System architecture

The system architecture is divided into three main components, the mobile application, the machine-learning algorithms, and backend services.

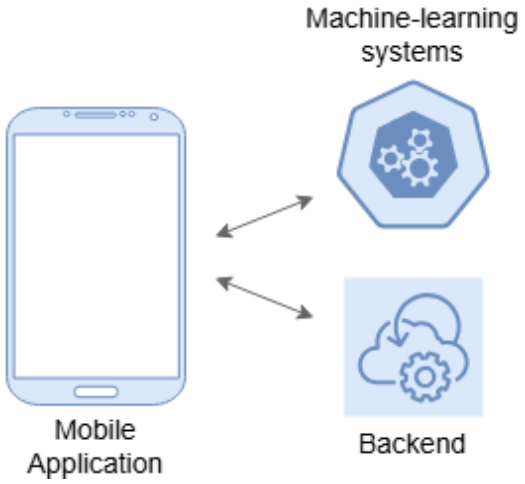


Figure 3.2: System components

3.6.1 Mobile application

The first component is the mobile application itself. It includes all the visual aspects, including the layout, design elements, and interface components.

3.6.2 Backend

The second component is the backend services of the application. In our project, it is necessary to have a backend that provides the following functionalities:

- Database management
- Authentication and authorization
- Synchronization between multiple devices
- Ability to integrate with other systems and services
- Analytics and Monitoring
- Security concerns
- Geo-referencing support

3.6.3 Machine-learning systems

The third component is the machine-learning systems, which hosts the recommendation and image search algorithms on a server and uses an API that enables communication between the client (mobile application) and the server. The developed recommendation and image search API has two endpoints:

- **Recommendations Endpoint**: This endpoint provides recommendations based on user pet preferences.
- **Image Search Endpoint**: The second endpoint is for finding similar pets based on an image submitted by the user.

3.7 Technologies

The technology stack for this system is divided into three parts. Firebase services for user management, database, and file storage in the cloud. Secondly, the front-end that was built with React Native using JavaScript. Finally, the recommendation and image search RESTful API [28] developed in Flask

using Python and TensorFlow. The technology stack for this system is presented in the image below.

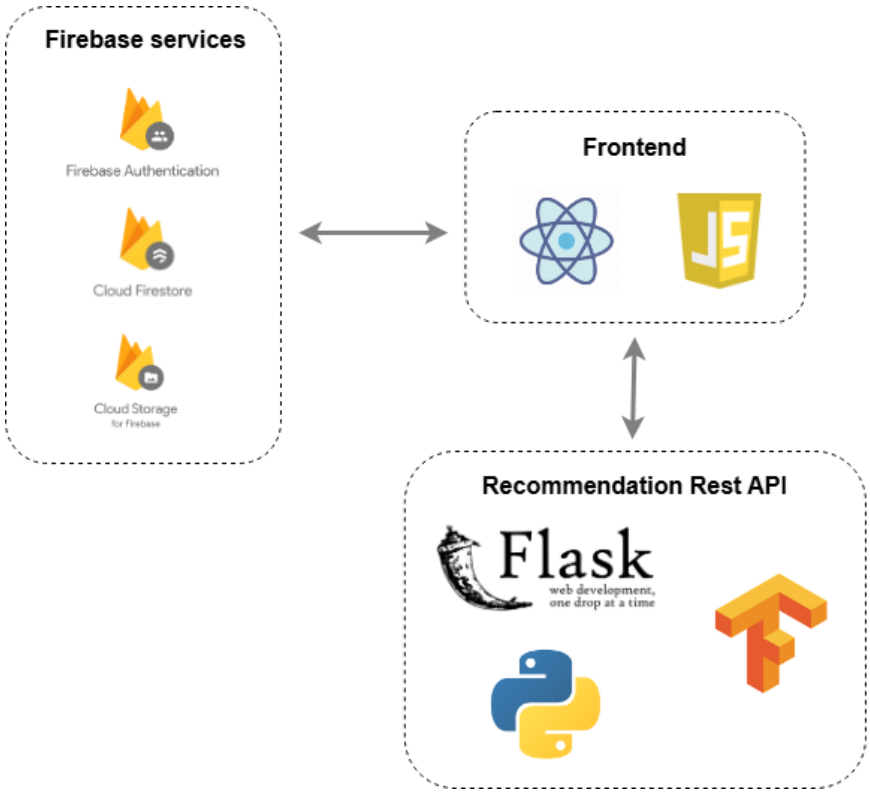


Figure 3.3: Technology stack

3.7.1 React Native

React Native, developed in 2015 by Facebook, is a framework to build mobile applications utilizing JavaScript and React. It allows the creation of native rendered apps for iOS and Android, using the same codebase. This framework uses React components as the core blocks for creating user interfaces. Each component is a reusable piece of code that represents a segment of the application UI, defining its own logic and style. React Native components map to native mobile elements, allowing JavaScript code to interact directly with platform-specific features.

One of the reasons for React Native quick success is the fact that it was built based on React, an already highly popular JavaScript library. Another

reason is its focus on mobile development, allowing integration with native features. This is crucial because it allows access to device-specific functionalities like camera access, GPS and local storage.

The primary reason for choosing React Native for this project was its capability for cross-platform development, which enables quicker development time compared to native app development. Another popular cross-platform framework that could have been used instead is Flutter.

3.7.2 Flutter

Flutter, released in 2018 by Google, is an open-source cross-platform framework, which uses Dart, a programming language developed by Google. Flutter offers a wide set of pre-designed widgets that define the appearance of their view based on the application's current state. It has an unique rendering engine that allows for consistent performance across multiple platforms.

Although Flutter has a rapidly growing community with increasing support, it still stands behind React Native in terms of third-party libraries and tools. React Native was selected for this project based on its well-established ecosystem and JavaScript foundation. While Flutter could also have been a viable choice due to its high performance and consistent UI rendering.

3.7.3 Backend

There are three primary types of backend [29], respectfully:

- **Software as a Service (SaaS)**: SaaS is a software delivery model where the application is hosted by a third-party provider and delivers software through the internet. This type of service usually requires a monthly subscription fee. The problem with using SaaS in a mobile application is its lack of flexibility, as it will usually not fully satisfy all specific system needs.
- **Backend as a Service (BaaS)**: BaaS is a cloud computing model that provides an infrastructure that automates the backend part of a mobile application. It simplifies the development of applications allowing users

to focus on building the frontend. This model abstracts the complexity of backend development, providing a set of backend features through the cloud.

- **Custom Backend**: A custom backend offers total control over the backend side of the application, allowing the implementation of features that cannot be implemented with BaaS. It provides full control over the backend performance, security and deployment. The drawback of this type of solution is the cost and the time to build.

Considering the advantages and disadvantages of all these backend solutions, Mobile Backend as a Service (BaaS) is the most suitable choice for this project due to its functionality and time efficiency.

3.7.4 Google Cloud Platform Firebase

Firebase is Google platform for mobile and web application development. It shares a common infrastructure with Google Platform allowing easy integration with different Google Cloud Services. It provides a set of features like authentication, Realtime databases and cloud messaging. Overall, Firebase goal is to simplify the process of developing applications, cutting the time required to create a backend from scratch, offering tools for managing app infrastructure.

3.7.5 Flask

Flask is a lightweight framework for Python used to create web servers and handle HTTP requests. One of its primary uses is to create RESTful APIs to integrate web services. A similar tool is Django [30], a high-level web framework, which includes many built-in features, such as ORM (Object-Relational Mapping), user authentication and an admin interface. Although Django is excellent for complex projects, its extensive feature set can be excessive for smaller applications or APIs. In this project, since it was only necessary to implement a simple API, Flask was selected due to its lightweight and flexibility. This allowed for a faster and simpler development.

3.7.6 Technology stack

The system architecture, along with the defined technologies, can be divided into three parts:

- **Mobile Application** : Application developed using React Native.
- **Backend services**: Google Cloud Platform backend services that interact with the mobile application through the Firebase JavaScript SDK.
- **Flask server**: Server that hosts the recommendation system and image search mechanism, enabling communication between the server and client through HTTP requests, through a RESTful API. The API is deployed on the Google Cloud Platform using Cloud Run, which exposes the API to the outside and enables automatic scalling.

The system architecture, with the technology stack is presented in Figure 3.4.

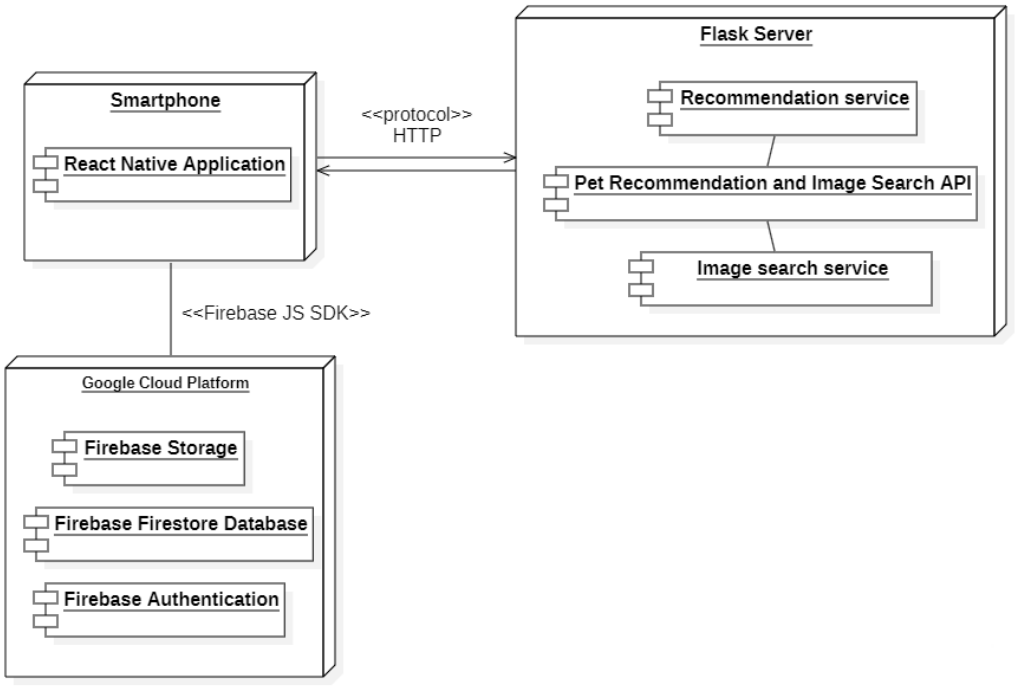


Figure 3.4: System architecture with technology stack

3.8 Application prototype

In the design phase of the system, a prototype of the application user interface was created using wireframes [31]. Wireframes focus on the structural side of the application rather than visual aspects such as colors or styles. These prototypes allows us to visualize and test the user interface, facilitating constant redefining of the application's design and functionality. The wireframes were implemented using Figma, a collaborative interface design tool that offers countless features for creation of interactive prototypes.

In figure 3.5, wireframes for the homepage, search page, and the 'add pet' page are presented. The primary focus of these prototypes was to predominantly display the available pets, ensuring their most relevant information is clearly visible.

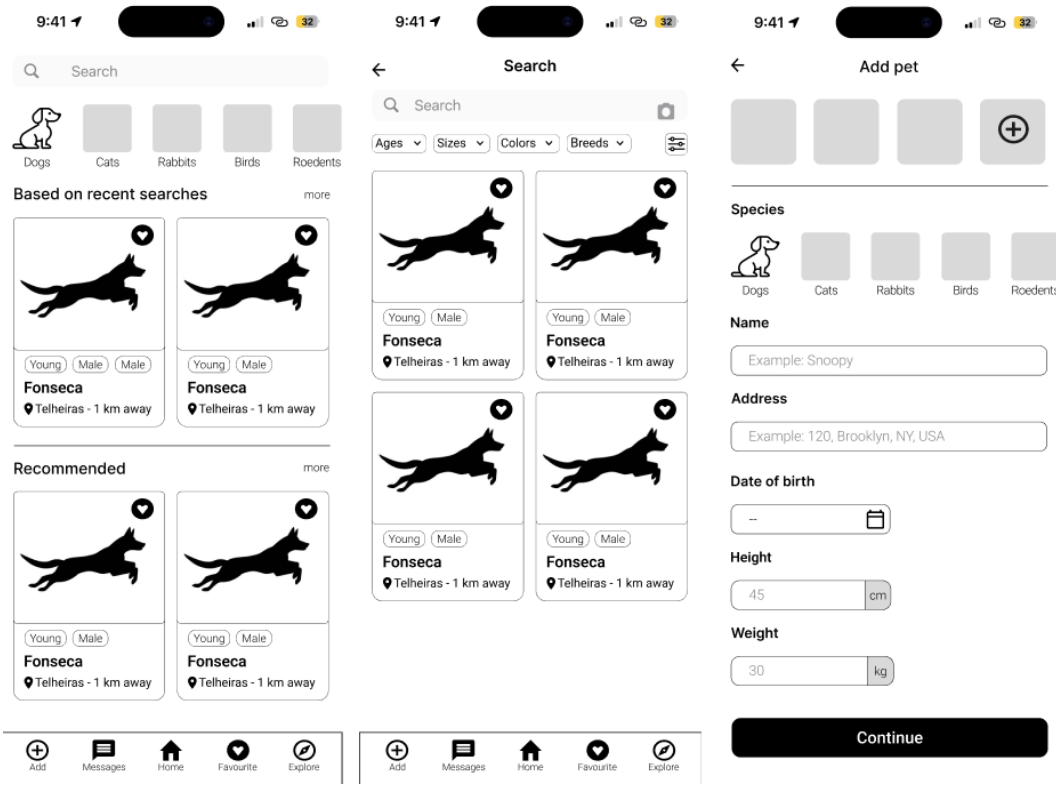


Figure 3.5: Application wireframe

In figure 3.6, wireframes for the animal page are presented. The primary goal of these prototypes was to present all the animal information clearly, highlighting the key details. There is a great focus on the animal's location.

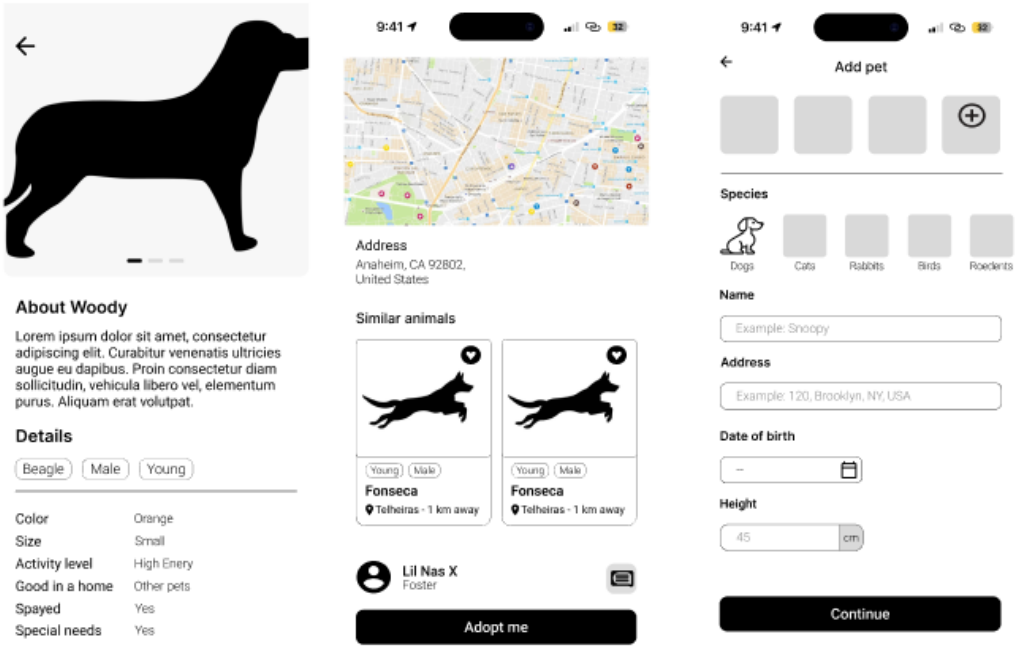


Figure 3.6: Application animal screen wireframes

The remaining wireframers can be viewed through the Figma link in the Appendix section of this paper.

Chapter 4

Recommendation System and Image Search

This chapter describes the recommendation system and the image search mechanism. It starts by the construction of the dataset.

4.1 Data collection

To train and test the machine-learning algorithms, it was necessary to collect a dataset with relevant information about the animals available in the application. The recommendation system is based on animal attributes, while the image search relies on images; therefore, it was necessary to obtain data with the relevant pet attributes and images.

As mentioned in chapter three, the pet attributes defined for pet recommendation include species, color, breed, health, and more. Therefore, it was crucial to gather data and find data that contained these attributes and had associated images.

4.1.1 Dataset

PetFinder Adoption Prediction dataset [32], originally created for a competition focused on pet adoption speed prediction, was used. This dataset provides the necessary data fields for the recommendation system, and image data for pets listed on the PetFinder system. The dataset is divided into train and test sets, containing information for pet attributes in Excel CSV

files and images saved as JPG files. Overall, the dataset contains data on 10,000 dogs and 9,000 cats.

4.1.2 Data preparation

Before the dataset is available to use to train and test the models in our recommendation system it is necessary to perform data preparation. This includes consolidating and cleaning the dataset prior to analyzing it. To process the data in the Excel CSV files of the dataset, the Pandas Python library was utilized. This is a fast and flexible data manipulation and analysis tool created for Python.

Figure 4.1: Data preparation

4.1.3 Data cleaning

The first step in data preparation was cleaning, which involves deleting unnecessary columns, removing null rows, and deleting duplicate rows. Additionally, certain columns names were changed to more appropriate names to improve clarity and cohesion with the application. This step included standardizing the data format and deleting inconsistencies to ensure a clean dataset for the following stages.

4.1.4 Data selection and transformation

The second step in preparing the dataset was selection and transformation. This involves converting column values to standardized and clear values to ensure cohesion within the application. Initially, it was necessary to convert the age fields from numerical values to standardized categories such as “Young” or “Adult”. Similar transformations were applied to the size, color, and species fields. Afterward, it was necessary to process the breed data fields.

With over more than 60 breeds for both cats and dogs, it was important to maintain consistency with the application. Therefore, only the top 35 breeds were selected, while other breeds were either tagged under the "Other" category or combined with one of the top 35 breeds.

The results of the data collection were 10,000 data points of dogs and 9,000 cat data points. Each containing pet attributes and an image.

Table 4.1 presents an example of dataset rows, demonstrating the different attributes and IDs for each animal. These IDs allow the animal rows to be associated to their corresponding image file. The example does not contain all the attributes due to size constraints, it is missing the 'Health' and 'Sterilized' attributes.

Table 4.1: Example of data saved in dataset (CSV) file

Species	**Age**	**Gender**	**Breed**	**Color**	**Size**	**PetID**
CAT	BABY	MALE	Domestic	YELLOW	MEDIUM	a00001
CAT	BABY	MALE	Abyssinia	BLACK	MEDIUM	a00002
CAT	ADULT	MALE	Domestic	ORANGE	SMALL	a00003
CAT	BABY	FEMALE	BENGAL	BLACK	MEDIUM	a00004
CAT	BABY	MALE	Abyssinia	BLACK	MEDIUM	a00005

4.2 Image Search

The basis of the image search is to classify an input image from the user using a Convolutional Neural Network (CNN), then extract the feature vectors from the network's prediction. These feature vectors are used to calculate similarities between the input image and the feature vectors of pets in the dataset, identifying images that are similar to the input image. The classification in the CNN is used only to obtain the feature vectors. The similarity calculation is then performed with all images in the dataset, not limited to any specific class.

Instead of building and training a CNN from scratch to extract the feature vectors, an existing pre-trained Keras model was utilized.

4.2.1 Pre-trained Keras model

Keras applications [33] are deep learning models, each accompanied by pre-trained weights. These models can be applied to a variety of different tasks, such as prediction, feature extraction, and transfer learning.

Models like ResNet50, Xception [34], InceptionV3 [35], were trained on the ImageNet dataset, which contains over 14 million images across 1,000 categories. This rigorous training allows these models to recognize a wide range of features and patterns in different images. Based on these reasons, these models are ideal to perform feature extraction in images.

4.2.2 Pre-trained model selection

Each pre-trained model has 5 performance metrics associated, Top-1 Accuracy, Top-5 Accuracy, Model Size, Parameters, Depth, Inference Time (CPU), and Inference Time (GPU). This metrics are important to choose the correct model for a specific task. Three pre-trained models that were chosen for consideration in the project were: MobileNetV2, Xception, ResNet50.

Ultimately, we used and tested ResNet50 and Xception due to their high accuracy and number of parameters.

Table 4.2: Pre-trained Keras model metrics

Model	Size(MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88	79.0%	94.5%	22.9M	81
ResNet50	98	74.9%	92.1%	25.6M	107
MobileNetV2	14	71.3%	90.1%	3.5M	105

4.2.3 Image search method

The initial step in implementing the search mechanism based on similar images is to extract feature vectors from the animal images in the dataset. These feature vectors are then saved so that they can be used to perform similarity calculation with images inserted in the application by the users. This process is represented in figure 4.2.

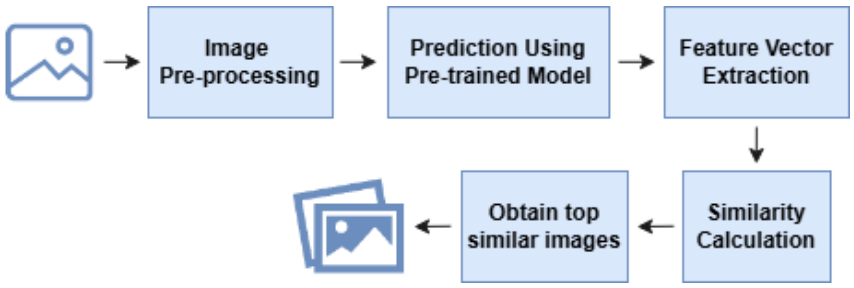


Figure 4.2: Image search process

4.2.4 Image pre-processing

Initially, each image file in the dataset is loaded and pre-processed. This process involves the following steps:

1. **Loading the images**: Each pet image file is read and loaded in the system.

2. **Resize**: The loaded images are resized to a fixed dimension of (244, 244) to standardize the input size for the Neural Network. This specific size is required for the ResNet50 model, which ensures the images must be 244x244 pixels with 3 color channels, to ensure compatibility with its architecture.

3. **Array Transformation**: The images are converted to a NumPy array.

4. **Pre-processing with Keras ResNet50**: To prepare the images for the input ResNet50 model, a pre-processing Keras Apllication function is called. This will convert the input images from RGB to BGR. Next, each color channel will be zero-centered with respect to the ImageNet dataset, without scaling.

4.2.5 Feature vectors extraction

After processing the images, it is necessary to pass them to the ResNet50 model to obtain the feature vectors. This process has the following steps:

1. **Load the ResNet model**: Initially, it is necessary to load the pre-trained ResNet50 model with weights that have been trained on the

ImageNet dataset. This allows the generation of feature vectors based on its extensive training.

2. **Predict the images with the loaded Model**: The input images are fed into the pre-trained model, resulting in a multi-dimensional array, which contains the CNN's feature maps.

3. **Flatten the feature vectors**: The outputs from the previous step are flattened to create one-dimensional feature vector. This simplifies the feature representation for the following steps.

4. **Save the feature vectors on a pickle file**: The feature vectors are saved in a Pickle file for further use.

4.2.6 Image similarities

To find the most similar images to an image inserted by the user, it is necessary to first obtain the feature vector for that input image. The user image comes in binary format, so it is first necessary to decode the image with three image channels, then perform the same steps that were done for the images in the dataset to obtain their feature vectors.

After obtaining the feature vector of the user image, it can be compared with the feature vectors of the images in the dataset. The similarities between the input image and the images in the dataset are calculated using the cosine similarity metric.

An example of the results obtained by the image search system can be observed in Figure 4.3.

Figure 4.3: Example of an image search result

4.3 Recommendation system

The recommendation system involves training a clustering model that groups pets in the dataset according to their attributes. Recommendations are then made by predicting the cluster that best matches the user's preferred pet attributes and performing similarity calculations within this cluster.

The process of generating a new recommendation involves the following steps:

1. The user specifies its pet preferences within the application through a questionnaire.
2. The user pet preferences are sent to the recommendation server through an HTTP request using the recommendation API.
3. The user pet preferences are pre-processed, creating a sentence.
4. Using a sentence tranformer, the sentence is converted into a vector.
5. The vector is inputted into the trained pet clustering model.
6. The model predicts the cluster that aligns best with the user's preferences vector.

7. Inside this predicted cluster, the system recommends pets that closely match the user's preferences, through similarity calculation between the user pet preferences vector and vectors of pets in the cluster.

The clustering model developed is based on sentence embeddings derived from a pre-trained sentence transformer model to process the pet attributes.

4.3.1 K-Means clustering

The initial step in this recommendation system is to train a clustering model with the pet data in the dataset. The K-means model training with sentence embeddings follows the process depicted in figure 4.4

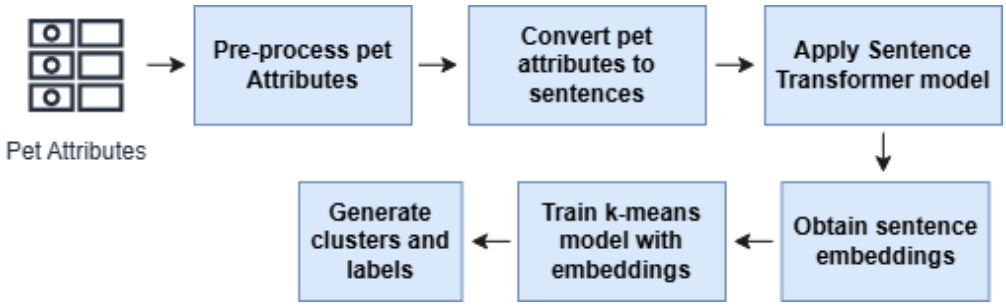


Figure 4.4: K-means clustering model training

4.3.2 Data pre-processing

The data of the pets saved in the database must be first pre-processed so that it can be used to train a clustering model. This process involves eliminating outliers, and null values. A KNN Imputer [36], a method that imputes missing data based on nearest data points in the feature space, is used to complete missing values using k-Nearest Neighbors. It works by replacing missing values in a sample with the average value of the nearest neighbors from the training set.

4.3.3 Data transforming

Sentence transformers are models that convert sentences into vectors of a fixed size, capturing the semantic meaning of the input. To be able to pass

data into the sentence transformer model it is necessary to convert the textual information of the attributes into sentences.

Figure 4.5 illustrates an example of transforming pet attributes in the dataset into a sentence.

Figure 4.5: Example of data transforming

4.3.4 Generate sentence embeddings

The sentences are then passed to the sentence transformer model. The sentence transformer model used was **sentence-transformers/paraphrase-MiniLM-L6-v2** [37], from the Hugging Face Model platform. A platform that contains a large amount of datasets, machine learning models, and other AI-tools. The model maps sentences into a 384 dimensional dense vector space, which are compact yet effective in representing semantic content.

These embeddings are then passed as input to the k-means model, which groups similar embeddings together based on their proximity in this vector space. This process allows the k-means model to group the pets meaningfully, based on semantic relationships.

4.3.5 Calculate the number of clusters (k)

To create the k-means clustering model it is first necessary to choose the number of clusters (k value). This value is chosen using the elbow method, a technique used to determine the optimal number of clusters for a clustering model.

This technique involves running the k-means algorithm in a range of K values, in the project we tested from 2 to 15, and plotting the Within-Cluster

Sum of Squares (WCSS) against the number of clusters. This metric calculates the sum of squared distances between each point and the centroid of its cluster, measuring how compact the clusters are.

The plot of the number of clusters (k) on the x-axis and the WCSS on the y-axis forms an "elbow" point. This point indicates where the rate of decrease in WCSS slows down sharply. Beyond this elbow point, adding more clusters does not significantly improve the compactness of the clusters. Therefore, the elbow point helps identify the optimal number of clusters.

Figure 4.6 shows an example of an elbow point for the k-means model at k=8.

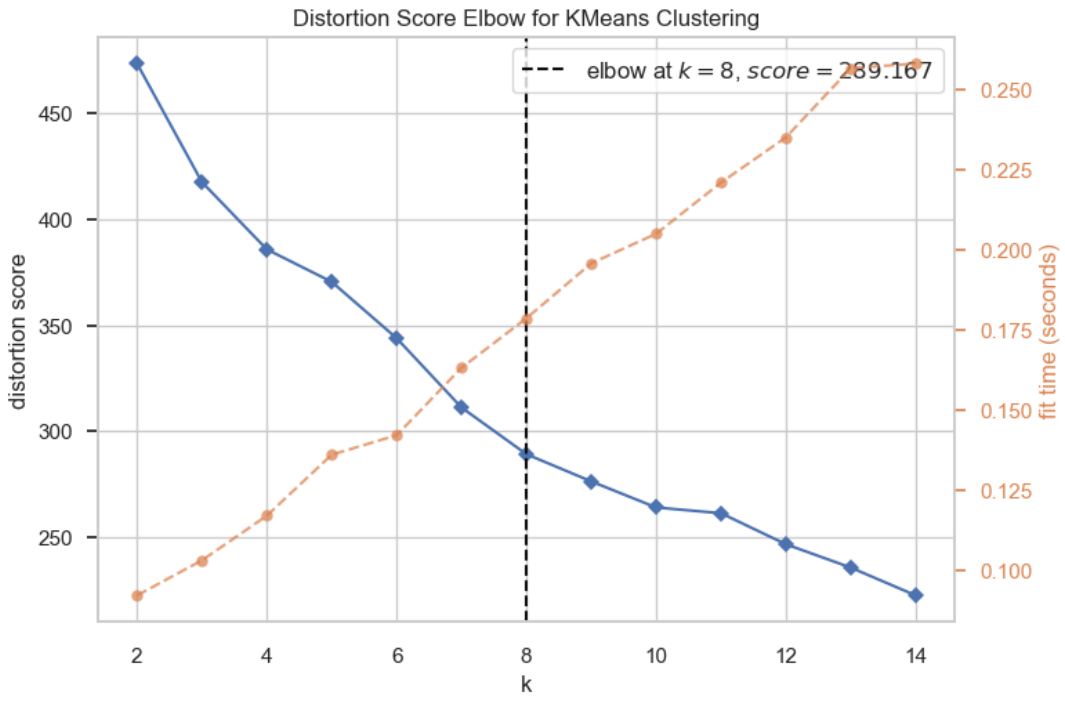


Figure 4.6: Elbow method

4.3.6 Model training

With the calculated k value, the k-means model is trained with the sentence embeddings, resulting in clusters and labels. Afterward, we can predict in

which cluster new data points should be classified to perform recommendations.

4.3.7 Pet recommendation

Once the clustering model is trained, we can predict in which cluster new data points should be classified to perform recommendations.

The process for recommending pets, based on a new user pet preference request is presented in Figure 4.7.

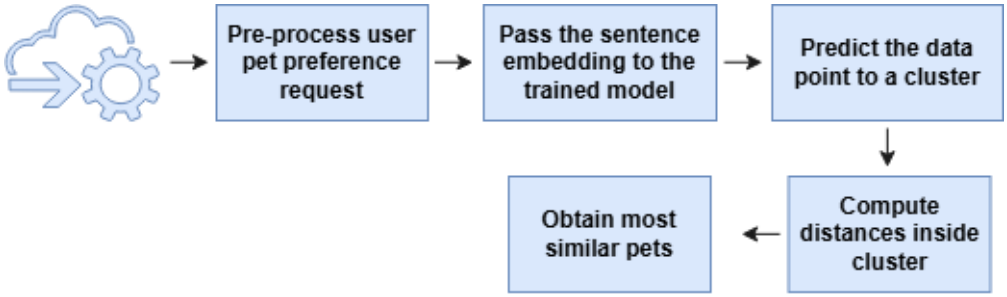


Figure 4.7: Pet recommendation diagram

1. **Pre-process the user pet preference request**: A request with the user pet attribute preferences is sent to the Flask recommendation server. The request is then processed and transformed in the same fashion as the dataset data used to train the model. This generates a sentence embedding ready to be predicted in the clustering model.

2. **Pass the sentence embedding to the trained model**: The processed user pet preferences sentence embedding is passed to the trained clustering model.

3. **Predict the data point to a cluster**: The data point is predicted to the cluster whose centroid is closest by the model.

4. **Calculate similarity inside cluster**: The distances between the user preferences data point and other points in the cluster are measured using cosine similarity. The similarities are sorted, obtaining the most similar pets.

Chapter 5

Implementation

In this chapter, we will go through the steps involved in the implementation phase of the application. This includes detailing the database model, file storage solutions, mobile application development, and API implementation.

5.1 Data model

In this project, Firebase Cloud Firestore was used to create the database. It is a NoSQL database where data is organized into collections, each collection containing multiple documents. Firebase offers real-time synchronization, allowing us to deal with live updates within the application. Its main advantages include flexibility and scalable data storage. However, one drawback is that complex queries can become costly and may require careful indexing to optimize performance. Additionally, since it is not a SQL database, it lacks support for traditional SQL querying and relational data management.

Since Firebase is a NoSQL database, it does not utilize entity–relationship and class diagrams, nor does it contain constraints such as primary keys and foreign keys, due to its schema-less nature. To deal with this problem, a more basic data model was built to highlight the database's entities, attributes, and relationships between entities.

The developed data model is composed of six entities, represented in Firebase NoSQL as collections: 'User', 'Pet', 'Favorite', 'AdoptionRequest', 'Message', and 'Chat'. Each collection is composed of documents that store

the collection data. The data model representing the different collections and their relationships is illustrated in Figure 5.1.

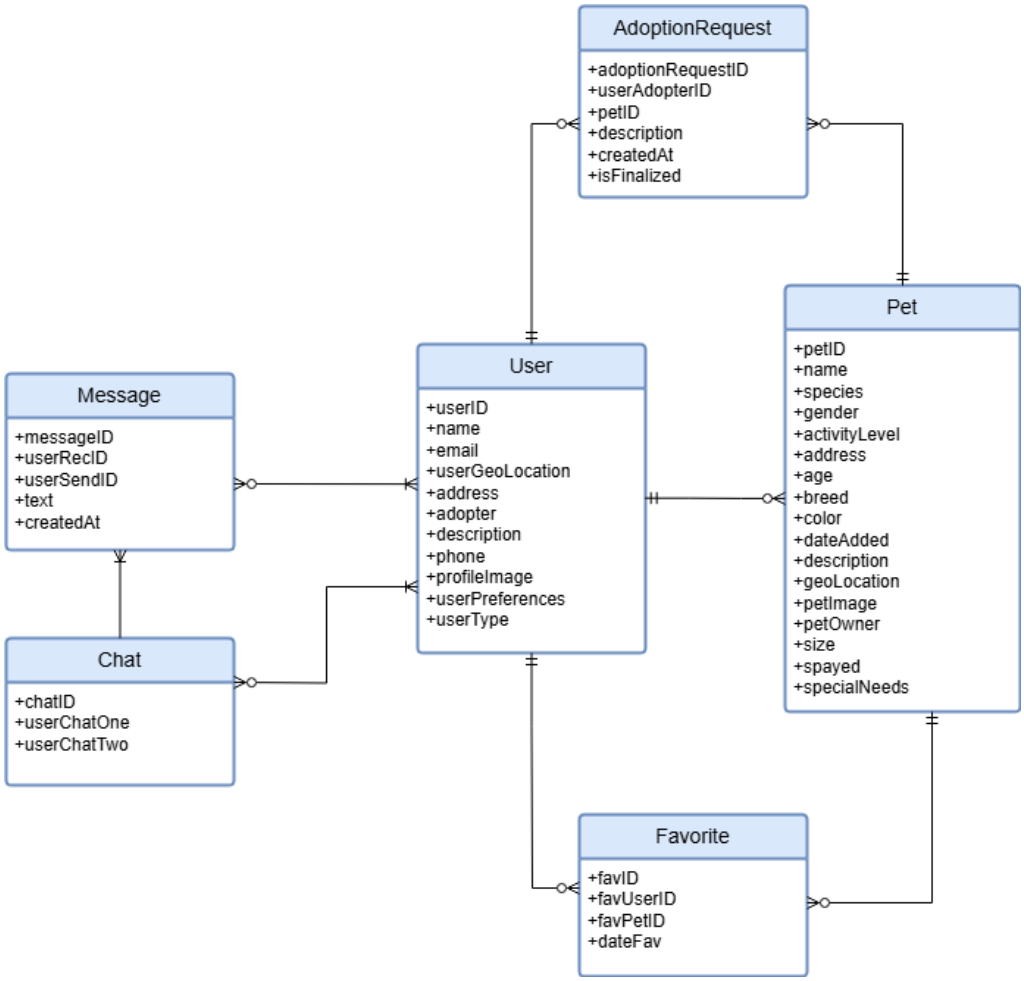


Figure 5.1: Data model

5.1.1 User collection

The user collection contains the user information and its respective attributes. Notable attributes include ‘userPreferences‘, an array that stores answers to the user preferences questionnaire; ‘geo-location‘, which includes latitude and longitude coordinates; and ‘profileImage‘, which stores a link to the image location in cloud storage.

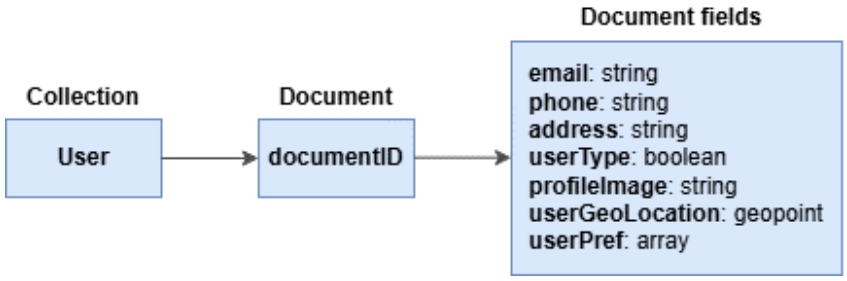


Figure 5.2: User collection

5.1.2 Pet collection

The pet collection holds the pet data. The pet collection contains a reference to its owner through the ‘petOwner‘ field, which contains its owner’s user identifier. Notable attributes include ‘geo-location‘, which includes latitude and longitude coordinates; and ‘petImages‘, an array that stores links to the pet images stored in cloud storage.

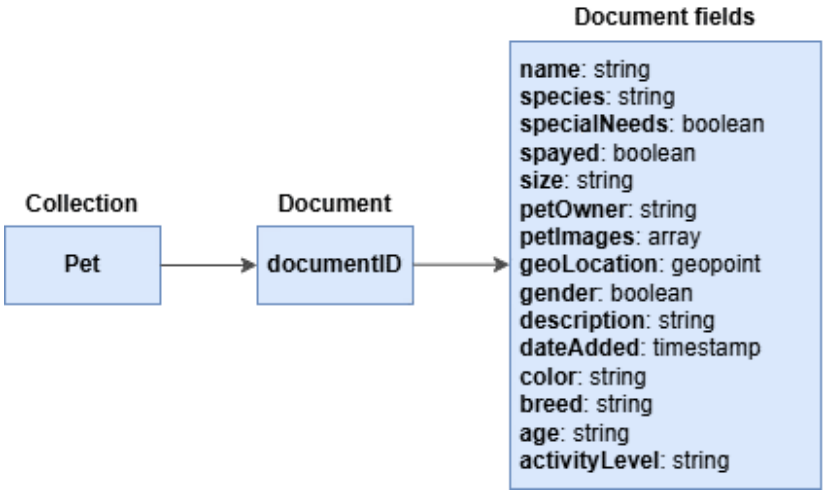


Figure 5.3: Pet collection

5.1.3 Favorite collection

The favorite collection establishes a relationship between a user and their favorited pets. Additionally, it records the date the pet was added to favorites for ordering purposes.

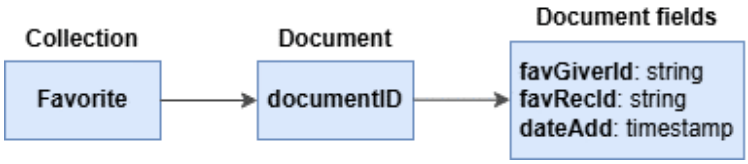


Figure 5.4: Favorites collection

5.1.4 Adoption request collection

The adoption request collection represents the relationship between a user seeking to adopt a pet and the owner. The user submits an adoption request to the pet owner, with a small description explaining the reasons for wanting to adopt the pet. Additionally, the date of the request and its status are stored.

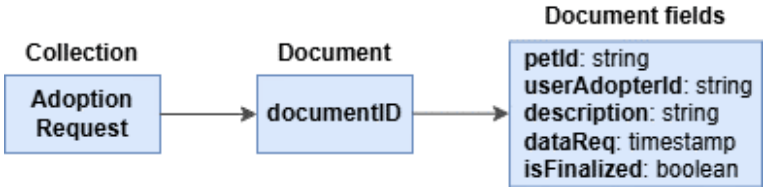


Figure 5.5: Adoption request collection

5.1.5 Chat collection

The chat collection represents a conversation between two users. It includes references to both participants and comprises multiple messages. Each chat document is composed of multiple message documents.

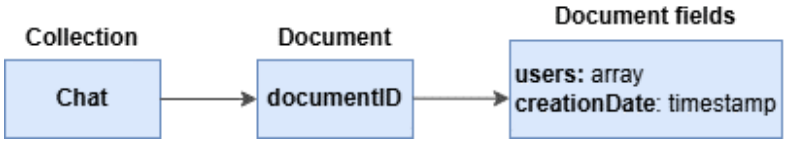


Figure 5.6: Chat collection

5.1.6 Message collection

The message collection records communication between two users, a sender and a recipient, storing the text and the date the message was sent.

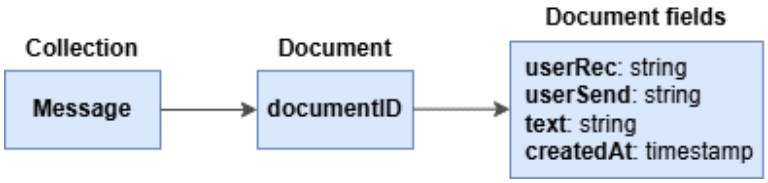


Figure 5.7: Message collection

5.1.7 File storage

The user and pet images are stored in Google Cloud storage, a cloud service incorporated natively into the Firebase architecture. It is used to upload and download files securely in Google Cloud. Each file is stored in a bucket, which are containers for storing objects, at a specified path. After uploading, a download URL for the image is generated and stored in the user's or pet's document in Firestore, allowing for easy access via this URL.

5.2 Petto application

The implementation of the React Native application "Petto" was divided into four layers: Views, Components, Services, and Entities. This modular approach allowed us to keep the code organized and maintainable.

- **Services**: The services establish the communication between the application and Firebase or with the external recommendations API.
- **Components**: Components are reusable UI elements that encapsulate specific elements of the user interface. Different components were created for recurring elements such as the search bar, footer and animal card item.

5.3 Application screens

5.3.1 User Authentication

User authentication in the application is implemented using Firebase Authentication, it provides server-side tools for managing user authentication on the Firebase platform. Validations were added into the Login and sign-up

process, including email format checking, password complexity requirements, and password confirmation matching.

Initially, a user attempts to sign in the application, with their credentials being collected. These credentials are then passed to the Firebase Authentication SDK, which verifies the information and returns a response to the client.

When the user is created in the Firebase project with success, Firebase assigns a set of properties like unique identifier, name, email address, and encrypted password. After the Firebase user is generated, a new user document is created in the user collection with the relevant information. To handle the authentication process, a service called 'AuthenticationService' was created using AngularFireAuth for user authentication with Firebase Authentication. Figure 5.8 presents the screens developed for the authentication process.

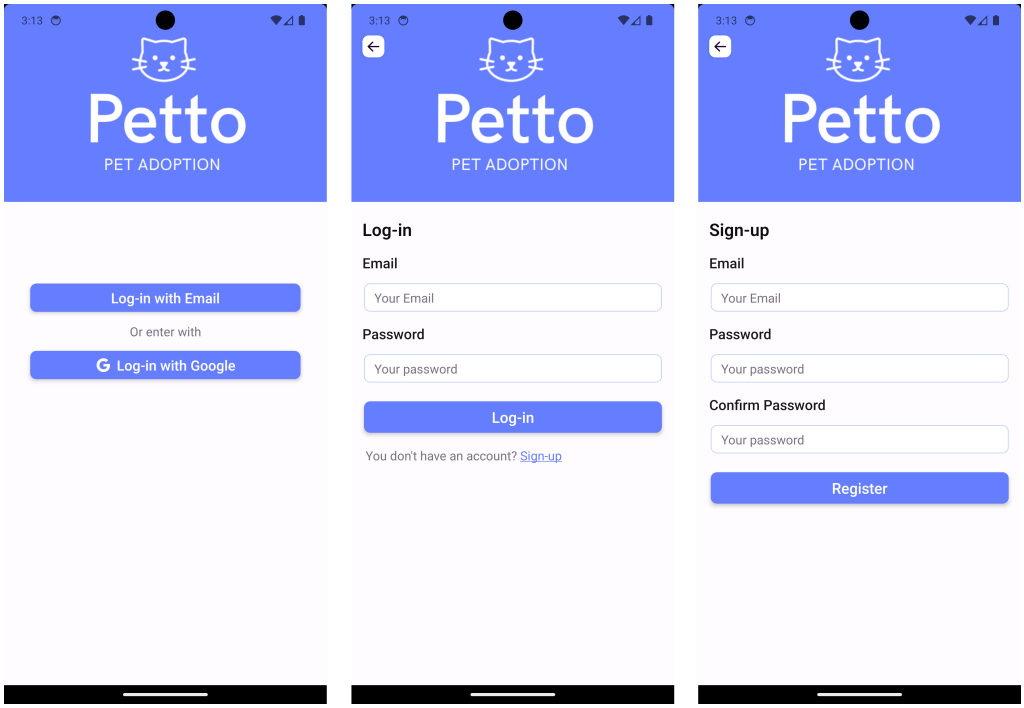


Figure 5.8: Authentication screens

5.3.2 Search

The search screens allow users to search pets in the application based on specific attributes, pet name or similar image, a search option where users can select an image and similar pets are presented.

The pets are displayed using a Flatlist component in React Native, which is designed for rendering large lists efficiently. Initially, the Flatlist only loads a small, predetermined number of pets. As the user scrolls down through the list, infinite scrolling is implemented to load more pets by triggering queries to the database in real-time. This approach helps reduce the load and the database strain associated with fetching the pet's data.

The screens implemented for pet search, filtering, and homepage are presented in Figure 5.9.

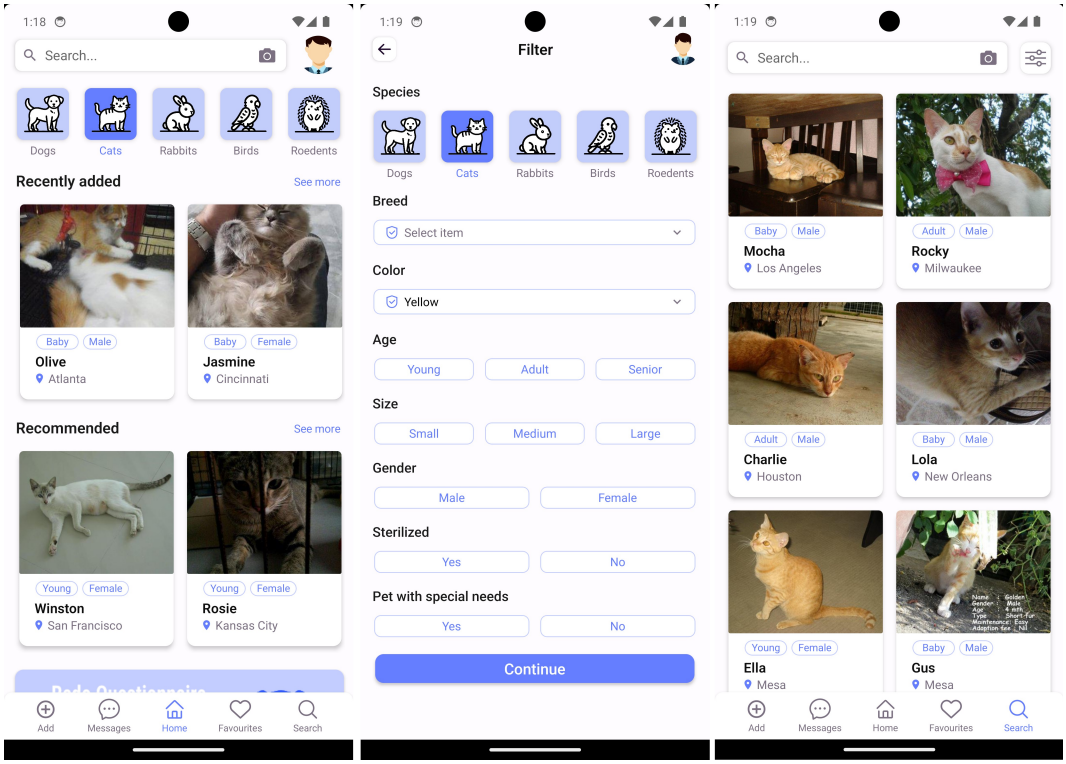


Figure 5.9: Search screens

5.3.3 Pet screen

The pet screen provides detailed information about a pet, including its owner and geographic location. Users have the option to add the pet to their favorites and send an adoption request to the pet's owner. Additionally, similar animals available for adoption are displayed. The pet pages developed are presented in Figure 5.10.

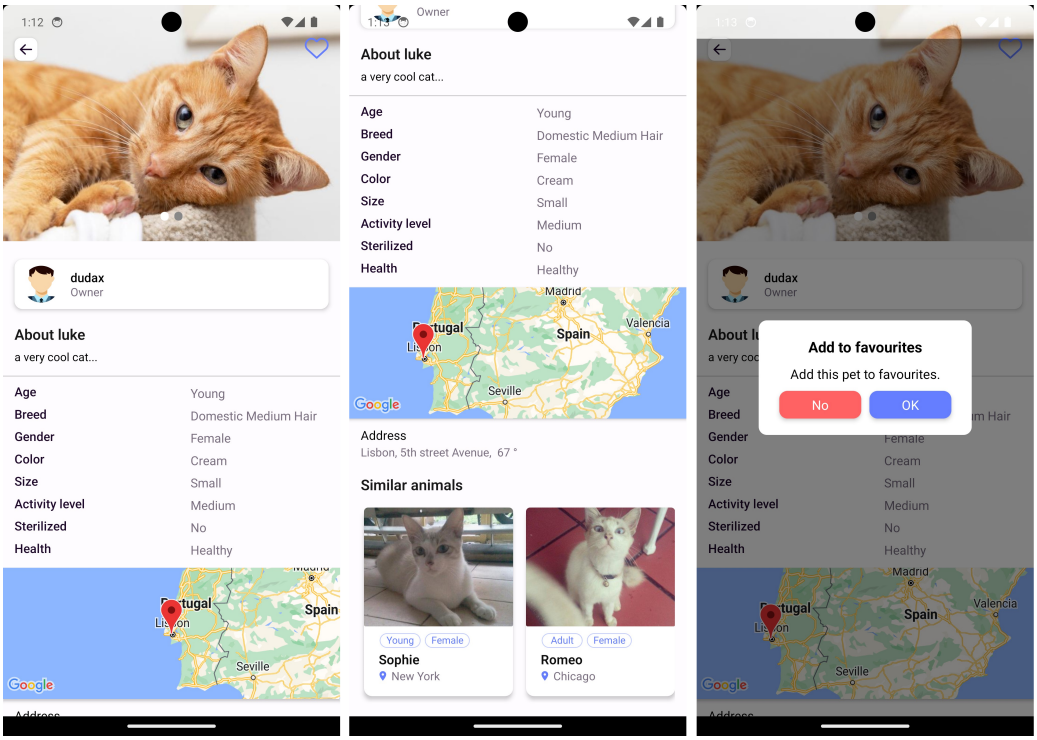


Figure 5.10: Pet screens

5.3.4 User screens

The user screen displays the user's information alongside the pets they have available for adoption. Additionally, there is a screen to edit the user profile. The user screens are presented in Figure 5.11.

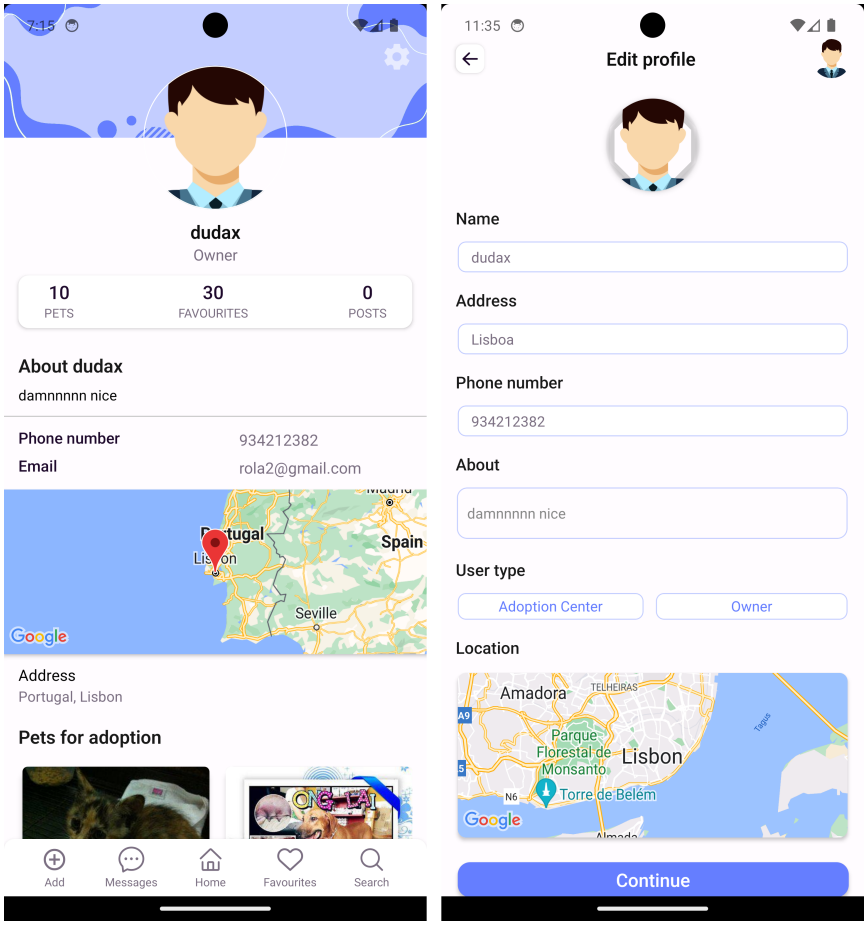


Figure 5.11: User screens

5.3.5 Add pet screens

The 'Add pet' screens allow users to add a new pet for adoption to the application, with each pet having up to four images and various attributes describing it. Additionally, there is an interactive map where the user can pinpoint the pet's location.

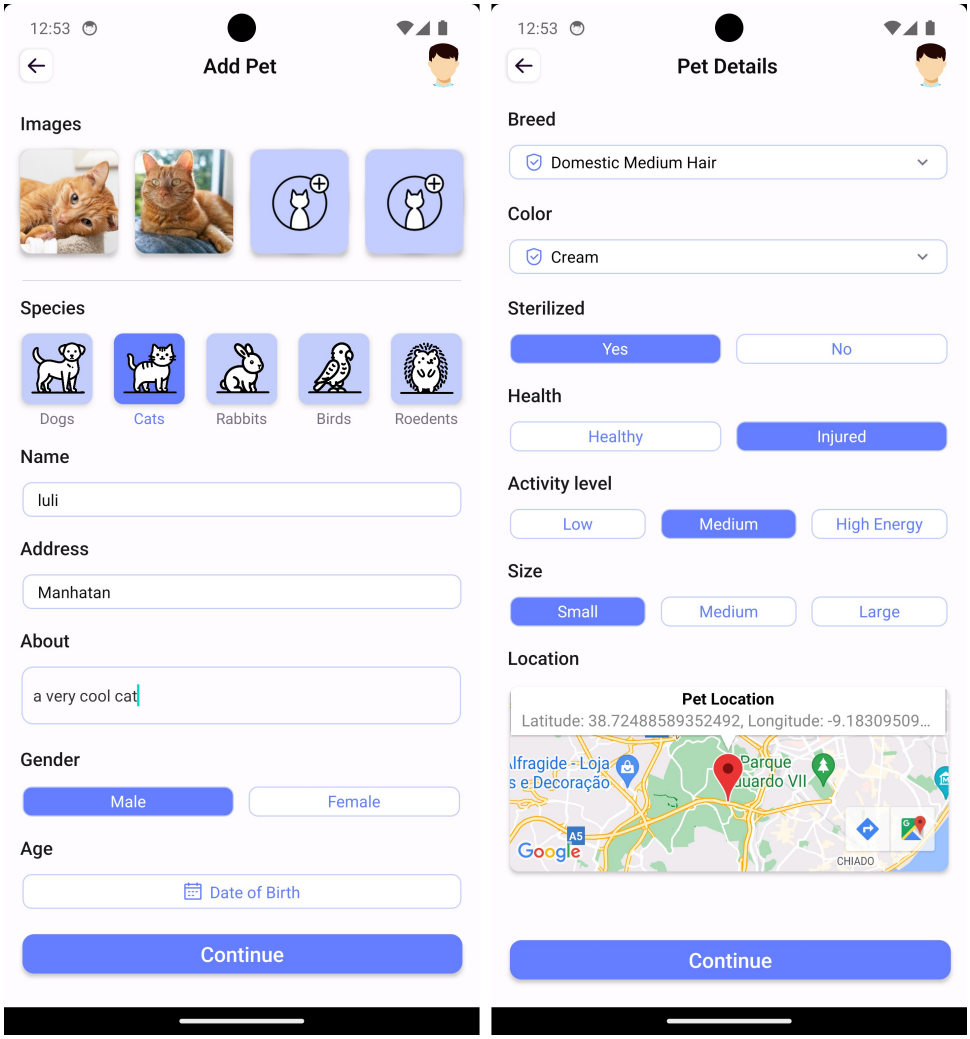


Figure 5.12: Add pet screens

5.3.6 Questionnaire

The questionnaire is a set of questions to gauge user preferences, used for recommendations. It is separated into three groups: user information, pet physical information, and pet health. The questionnaire can be redone anytime the user deems necessary, with the answers being saved in the user document in an array. The questionnaire screens are presented in Figure 5.13.

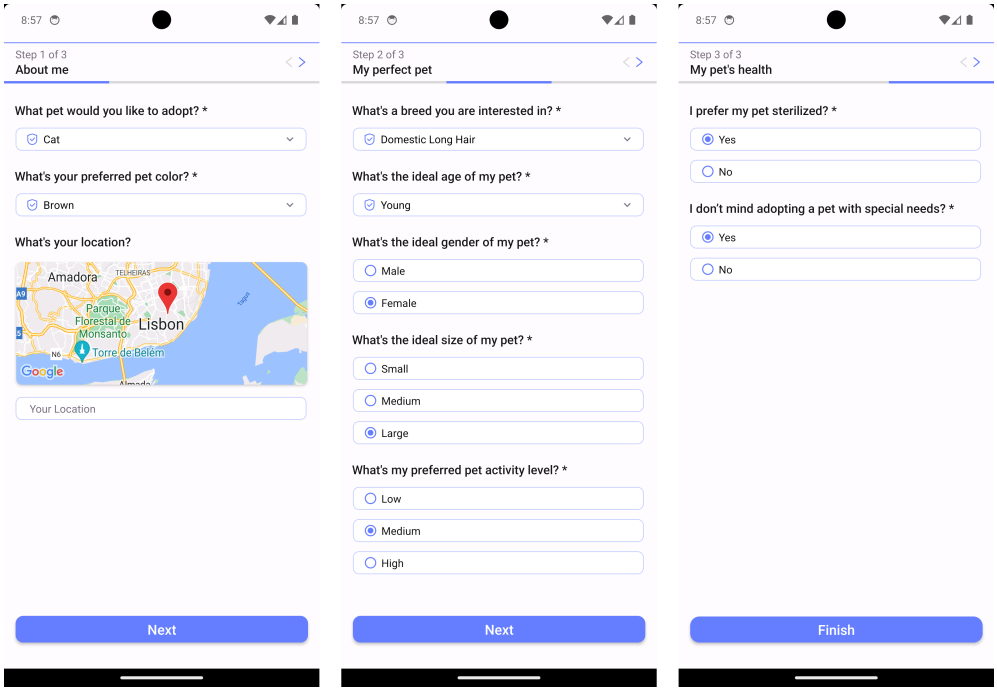


Figure 5.13: Questionnaire screens

5.3.7 Messaging chat

A user can send messages to another user after submitting an adoption request for a specific pet. Users can access their active chats through a messages page.

In Firestore, each 'Chat' document consists of multiple 'Message' documents. For real-time messaging between two users on the chat page, there is a subscriber listening to message updates on the 'Chat' document. The new messages are instantly fetched and presented to the user. Additionally, users can view a list of their favorite pets on a favorites page.

The screens developed for user interaction can be seen in Figure 5.14.

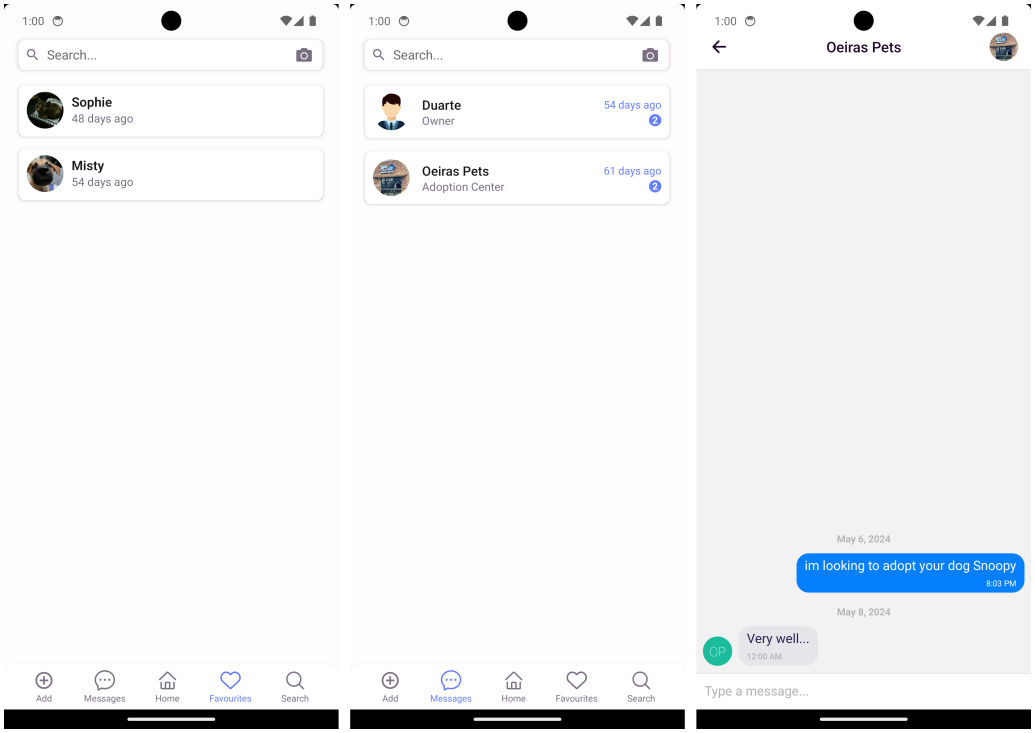


Figure 5.14: User interaction screens

5.4 Recommendation API

The Flask API was developed to enable the application to interact with the machine learning algorithms. In React Native the 'fetch' API is combined with the 'await' keyword to handle asynchronous requests. The API operates on a specific port, receiving and processing HTTP requests from the client application. There are two endpoints: one for recommendation system based on user's pet preferences and one for the search mechanism based on similar images.

The following section presents documentation for the two API endpoints.

5.4.1 Image search endpoint

The definition of the image search endpoint, including the method and its description, is provided in Table 5.1.

Table 5.1: Endpoint definition

Endpoint	/api/recommendations/image
Method	POST
Description	Provides pet recommendations based on an input image

Table 5.2 presents the endpoint request parameters, including type and description.

Table 5.2: Request parameters

Parameter	**Type**	**Description**
'image'	File	The image file to base recommendations on, uploaded as binary content

Table 5.3 presents the endpoint response parameters, including type and description.

Table 5.3: Response parameters

Field	**Type**	**Description**
'status'	int	HTTP status code indicating if the request was successful
'pet_ids'	JSON	An array of similar pets ID based on the input image

5.4.2 Recommendation Endpoint

The definition of the recommendation endpoint, including the method and its description, is provided in Table 5.4.

Table 5.4: Endpoint definition

Endpoint	/api/recommendations/attributes
Method	POST
Description	Provides pet recommendations based on user preferences (pet attributes)

Table 5.5 presents the endpoint request parameters, including type and description.

Table 5.5: Request parameters

Parameter	Type	Description
'attributes'	JSON	a JSON object containing user preference pet attributes

Table 5.6 presents the endpoint response parameters, including type and description.

Table 5.6: Response parameters

Field	Type	Description
'status'	int	HTTP status code indicating if the request was successful
'recommendations'	JSON	An array of recommended pets based on user preferences

5.4.3 API Deployment

The API application is deployed on the Google Cloud Platform. Initially, a Dockerfile is written to build the Python environment and expose the correct port (8080) for deployment. A Docker image of the API is then built. Afterward, the image is pushed to Google Container Registry (GCR) and deployed via Google Cloud Run, which automatically handles scaling and exposes the API.

Chapter 6

Results and Evaluation

In this chapter, the recommendation system, the image search functionality and the user experience of the application were tested and evaluated. This involved assessing the effectiveness of the recommendations through different metrics. Additionally, feedback was collected on the application's usability and user experience.

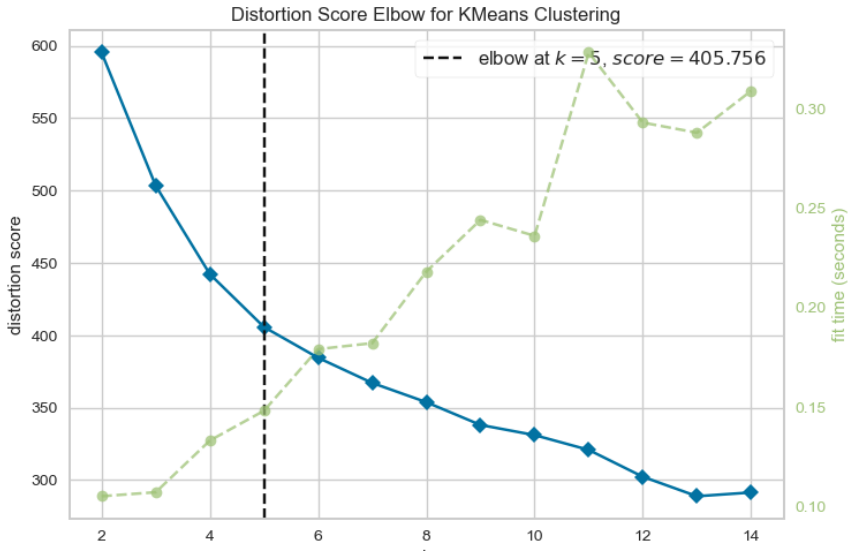
6.1 Recommendation system

For the pet attribute-based recommendation, we trained and evaluated clustering models separately for cats and dogs.

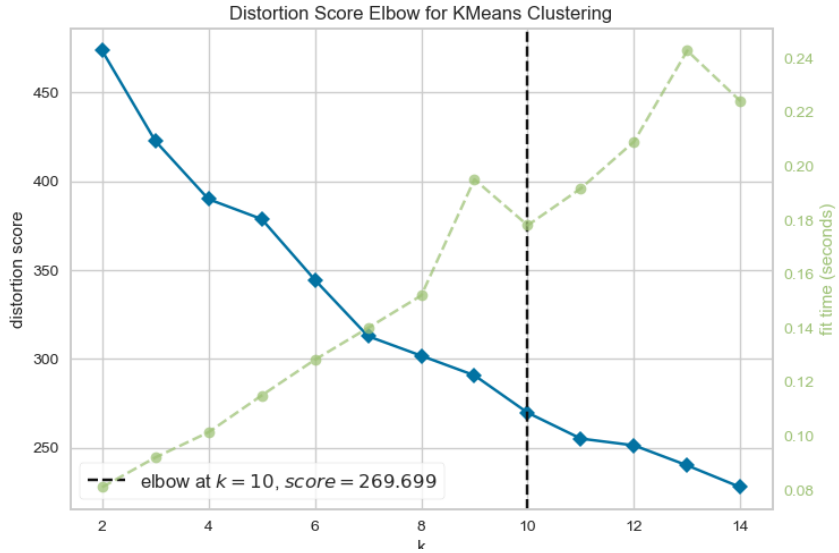
6.1.1 Clustering model evaluation

Separate models were developed for dogs and cats to obtain better clustering results. This approach ensures that the clusters formed for each type of pet are more meaningful and specific to their respective characteristics.

Initially, the elbow method was used to determine the optimal number of clusters (k) for the model, with k values ranging from 2 to 15. The dog model identified an elbow point at k=5, while the cat model identified it at k=10. Based on these results, these k values were chosen, and the impact of using both smaller and larger k values was also evaluated. The elbow points obtained for both models can be observed in Figure 6.1.



(a) Elbow method dog clustering model



(b) Elbow method cat clustering model

Figure 6.1: Elbow method results

Principal Component Analysis (PCA) [38] was used to evaluate and visualize the results of the cluster models. PCA reduces the dimension of data, enabling us to project the high-dimensional feature space onto two principal components. This allows us to visualize distribution and separation of clusters within the dataset, measuring how well the clustering model is separating

points between clusters.

In Figure 6.2, the PCA graph for the cat model is displayed. The clusters exhibit a decent level of separation among the 10 distinct clusters. This separation suggests that the clustering model captured meaningful differences between pet categories, although there is still some overlap present.

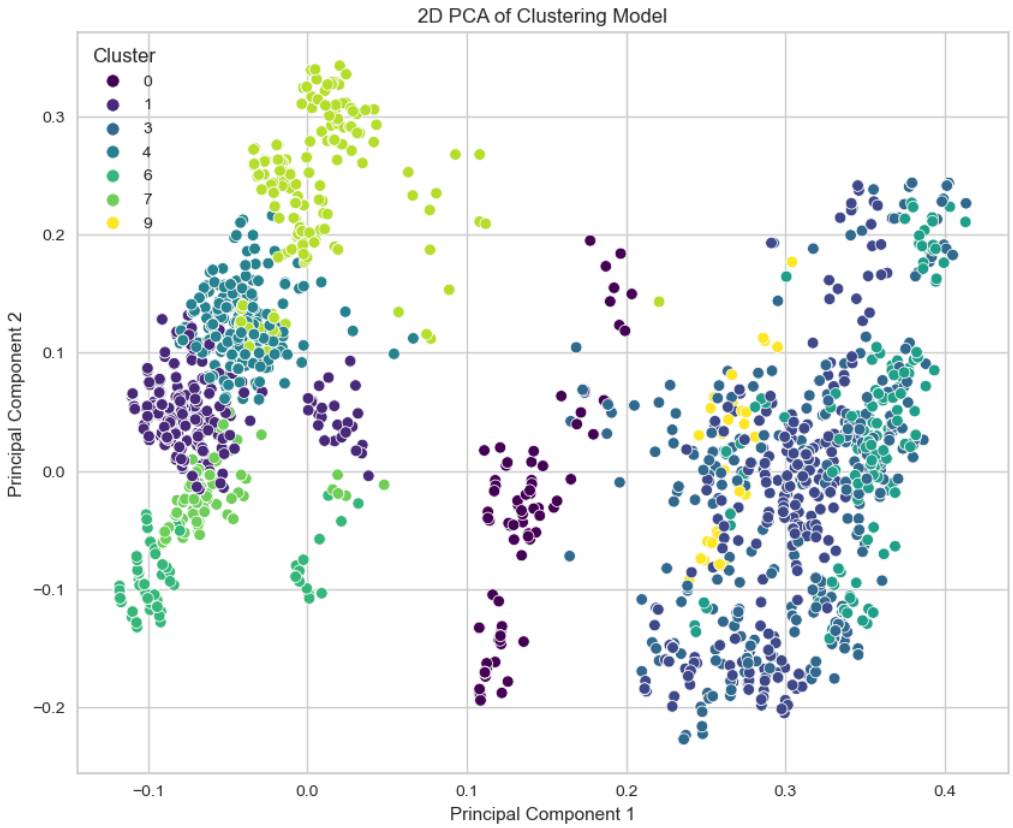


Figure 6.2: Cat model PCA

To be able to evaluate the significance of each variable in our clustering model we constructed a classification model based on the cluster labels. In this model, the input features are the variables used in the K-means clustering process, and the target variable is composed of the cluster labels predicted by the K-means algorithm.

The classifier chosen for this model was the LGBMClassifier [39], a classifier able to handle both categorical and numerical data.

To understand the classification results we used the SHAP TreeExplainer [40] on the classification model. This is a tool that interprets the results of tree models by measuring the contribution of each feature to the final outcome using SHAP values. These values work by assigning an importance value for each feature. The SHAP TreeExplainer helps to understand the impact of each feature on a trained model, allowing us to know how much each feature influences the final prediction. This allowed us to identify the most important attributes in the clustering model.

The results obtained from the SHAP TreeExplainer analysis are presented in Figure 6.3. It is evident that the most significant attributes in the clustering model are breed, color, and age. Possible reasons for this attributes importance are the following:

1. **Breed**: Defines distinct animal characteristics that differentiate pet types, therefore making it a key factor in clustering.
2. **Color**: Provides additional information to distinguish between animals, even in similar breeds.
3. **Age**: Impacts many aspects of a pet's behavior and health, which are important for accurate clustering based on pet attributes.

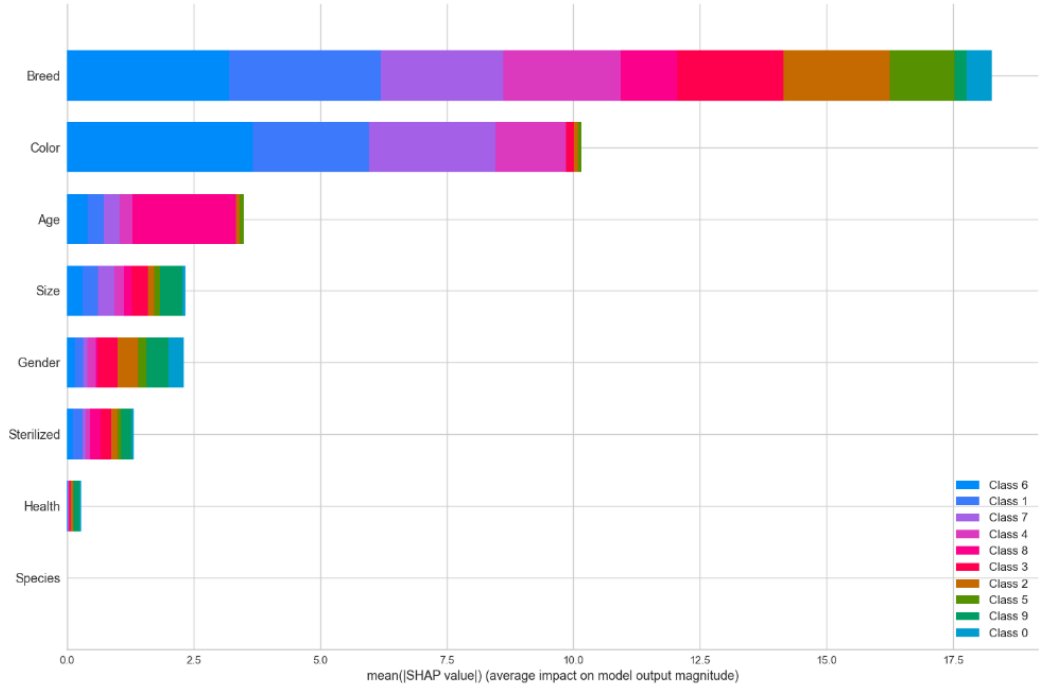


Figure 6.3: Attribute significance in model

Figure 6.4 shows how pets in the dataset are distributed across different clusters in the clustering model. There is a large concentration of data points in both cluster 1 and 6, while there are two clusters with very little data points.

This distribution suggests that there are two clusters with more prominent characteristics in the dataset, possibly containing the most common traits in the pet dataset. In contrast, the sparsity in the other clusters could indicate less frequent attributes, this could also point that the number of clusters used could have been different to achieve a more balanced distribution.

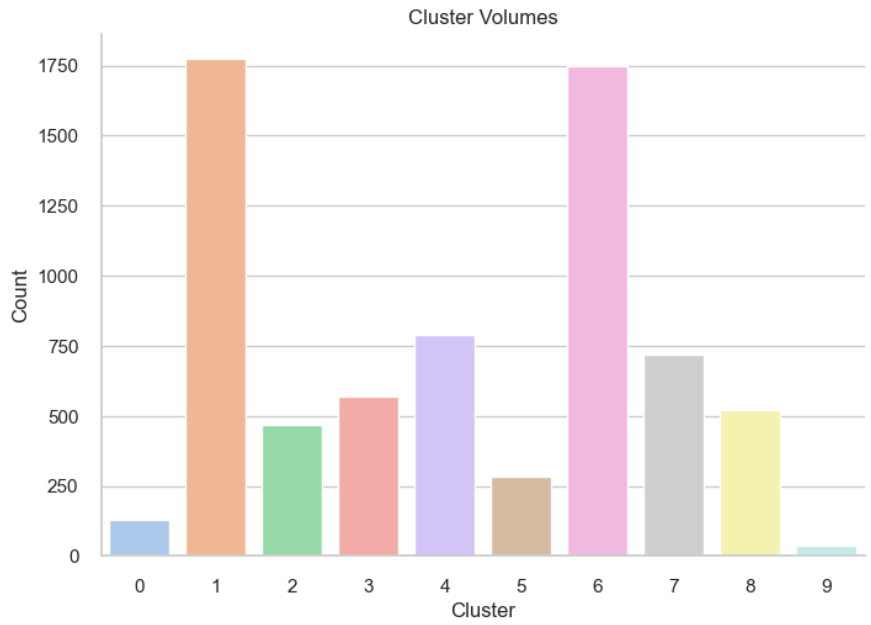


Figure 6.4: Cluster distribution

Figure 6.5 shows the most common attributes for each cluster. This is important to understand the characteristics that define each group. Certain attributes, such as "BABY" and "FEMALE," appear frequently due to their prevalence in the dataset. In contrast, the "Breed" attribute exhibits more variability, indicating a wider range of breeds within the clusters.

clusters	Species	Age	Gender	Breed	Color	Size	Health	Sterilized
0	CAT	BABY	FEMALE	American Shorthair	BLACK	MEDIUM	HEALTHY	NO
1	CAT	BABY	MALE	Domestic Short Hair	BROWN	MEDIUM	HEALTHY	NO
2	CAT	BABY	FEMALE	Siamese	BLACK	MEDIUM	HEALTHY	NO
3	CAT	BABY	FEMALE	Taby	BLACK	MEDIUM	HEALTHY	NO
4	CAT	BABY	FEMALE	Domestic Medium Hair	BROWN	MEDIUM	HEALTHY	NO
5	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO
6	CAT	BABY	FEMALE	Domestic Short Hair	BLACK	MEDIUM	HEALTHY	NO
7	CAT	BABY	FEMALE	Domestic Medium Hair	BLACK	MEDIUM	HEALTHY	NO
8	CAT	SENIOR	FEMALE	Domestic Short Hair	BLACK	MEDIUM	HEALTHY	YES
9	CAT	BABY	FEMALE	Maine Coon	BLACK	LARGE	HEALTHY	NO

Figure 6.5: Most common attributes by cluster

Three metrics were used to evaluate the clustering models [41], respectively:

- **Davies Bouldin (DB) index**: Measures the average similarity of each cluster with its most similar cluster using the ratio of within-cluster to between-cluster distances. The minimum value of the DB index is 0, a lower DB Index value, closer to 0, indicates a better clustering model.
- **Calinski Score**: Measures the ratio of between-cluster dispersion to within-cluster dispersion. A higher index indicates more separable clusters.
- **Silhouette Score**: Measures the quality of the fit for a clustering algorithm, it's useful to determine the optimal value of k. It ranges from -1 to 1, with 0 indicating overlapping clusters, and 1 indicating well-separated, dense clusters.

Table 6.1 presents the results for this three metrics for both the cat and dog model.

Table 6.1: Clustering evaluation metrics

Metric	**Cat model**	**Dog model**
Davies Bouldin index	1.59	1.79
Calinski Score	1146	1714
Silhouette Score	0.29	0.32

The cat model has a slightly better Davies-Bouldin index (1.59) compared to the dog model (1.79). This suggests that the cat clusters are more compact and better separated from each other than the dog clusters.

The dog model achieved a higher Calinski-Harabasz score (1714) than the cat model (1146). This implies that the dog clusters have a higher ratio of between-cluster dispersion to within-cluster dispersion, indicating better-defined and more distinct clusters.

Both models have low Silhouette scores, with the dog model slightly higher (0.32) than the cat model (0.29). This indicates some overlap be-

tween clusters, with the dog model having slightly better cluster separation.

6.1.2 Cluster model recommendations

To evaluate if the cluster model accurately predicts new data points, we conducted tests using input from the application. The Flask server receives a request with a new user's pet attributes, and the predicted cluster based on these attributes is identified.

After predicting the cluster, pairwise distances between the new data point and the points within the cluster are computed using cosine similarity.

In Figure 6.6 an example of a JSON message containing the user's pet preference attributes is sent to the Flask server to obtain recommendations.


```
{
'Species: CAT, Age: BABY, Gender: FEMALE, Breed: Per-
sian, Color: BLACK, Size: MEDIUM, Health: HEALTHY,
Sterilized: NO'
}
```


Figure 6.6: Example of pet preference request

The request is then processed, converting the JSON message to a sentence embedding. Subsequently, the cluster is predicted. The predicted cluster for this example was 5. Next, using cosine similarity, the most similar points in the dataset within this cluster are calculated. The most similar pets are displayed in Figure 6.7.

	Species	Age	Gender	Breed	Color	Size	Health	Sterilized	PetID
1530	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	37c989ac5
2733	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	64cf73826
2668	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	6258b0d76
3998	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	940b0ad69
2449	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	59047a5a1

Figure 6.7: Pet recommendations obtained

6.1.3 Precision

To evaluate the effectiveness of the recommendation system we used the Precision evaluation metric. This metric was calculated as the ratio of correct animal recommendations to the total number of animals recommended. Precision was calculated for 5 examples, each with 10 animals recommended. The average of these values was obtained to get the overall precision.

A recommendation is considered correct if the recommended animal attributes closely match the original pet attributes.

All the results obtained were correct, achieving a precision of **0.94**. The results obtained to calculate precision can be viewed in the Recommendation Results chapter on the appendix section. Overall, the system recommends pet effectively based on the pet characteristics.

6.2 Image search

For the search mechanism based on similar images, we tested the results using two different pre-trained Keras models: ResNet50 and Xception.

We also evaluated the effects of using the layer before the final classification layer to extract the feature vectors of the images.

Finally, we tested the impact of different similarity metrics for calculating image similarity based on feature vectors.

6.2.1 ResNet50 model

Initially, we tested the results of the recommendation system using the ResNet50 model, comparing the outcomes with and without utilizing the layer before the final classification layer to extract the images' feature vectors.

The summary of the last three layers of the ResNet50 pre-trained model is presented in Figure 6.8.

```
Model: "resnet50"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv5_block3_out (Activati  (None, 7, 7, 2048)        0
 on)

 avg_pool (GlobalAveragePoo  (None, 2048)              0
 ling2D)

 predictions (Dense)         (None, 1000)              2049000
```

Figure 6.8: Summary of the last three layers of the ResNet50 model

- **'avg_pool (GlobalAveragePooling2D)'**: Layer before the final classification layer, it is a pooling layer that reduces the spatial dimensions of the output from the convolutional layers. This results in a 1D tensor of size (2048,) that is fed into the fully connected (dense) layer.
- **'predictions (Dense)'**: Fully connected (dense) layer which is the actual last layer in the model used in the classification task. This results in a 1D tensor of size (1000,) this is the number of classes in the classification task that ResNet50 was originally trained on.

The output of the 'avg_pool (GlobalAveragePooling2D)' layer is (2048,) while the 'predictions' layer is (1000,). The 2048-dimensional feature vector produced by the 'avg_pool' layer is rich in visual information and not specific to any particular classification task. Meanwhile, the 1000-dimensional vector from the 'predictions' layer is specific to the classification task, and it is less useful for tasks that require general feature representations like similarity measurement.

Considering the reasons mentioned above, it was determined that using the output of the 'avg_pool' layer was the right choice to obtain better similar image results.

Based on an input image, it was evaluated how accurate the most similar images results were. This was evaluated using the feature vectors of images with the output of the final 'avg_pool (GlobalAveragePooling2D)' layer and

the output of the 'predictions' layer.

In both cases, Cosine similarity was used as the metric to compare feature vectors, ensuring fair results. The results obtained are illustrated in Figure 6.9 and Figure 6.10. As it can be seen in the images, using the 'GlobalAveragePooling2D' layer produces better results, calculating more similar images. This can be observed by the color, background, and the position of the animals of the most similar images to the input image.

Figure 6.9: Image similarity using the output of 'predictions' layer

Figure 6.10: Image similarity using 'GlobalAveragePooling2D' layer

6.2.2 Similarity metrics

After validating that the results of the recommendations utilizing the layer before the final classification layer to extract the images' feature vectors are better, we tested the effect of different similarity metrics. The metrics tested were Cosine similarity, Euclidean distance and Manhattan distance.

Examples of the results obtained for these three metrics based on an input image is presented in Figures 6.11, 6.12, and 6.13.

Figure 6.11: Image similarity example with Cosine similarity

Figure 6.12: Image similarity example with Euclidean distance

Figure 6.13: Image similarity example with Manhattan distance

As seen in the images above, all three metrics produced satisfactory results, with very similar outcomes and processing speeds. Ultimately, Cosine similarity was chosen for the application because it provided slightly better results in terms of similarity to the input image. Additionally, this metric is appropriate for high-dimensional data, such as image feature vectors, because it focuses on the orientation of the vectors rather than their absolute values.

6.2.3 Xception model

Another pre-trained Keras model was also used, the Xception model. This model is often used in image classification tasks, being very effective for handling large image datasets.

In order to compare the results of this model with the ResNet50 model, we also extracted the feature vectors utilizing the layer before the final classification layer, and used Cosine similarity.

The ResNet50 model was trained using the default image size of 224x224 pixels. In contrast, the Xception model utilized a larger default size of 299x299 pixels, enabling it to capture more intricate details in the images, but at the expense of increased computational demands.

Both models produced relatively similar results, making the process of choosing one of the models challenging. However, the ResNet50 model was ultimately chosen due to its faster training and inference times, as well as its slightly superior results in the tested examples.

Figure 6.14: Image similarity example with Xception model

Figure 6.15: Image similarity example with ResNet50 model

6.2.4 Precision

To evaluate the effectiveness of the image search mechanism we used the Precision evaluation metric. This metric was calculated as the ratio of correct similar animals to the total number of animals retrieved in the image search. Precision was calculated for 5 examples, each with 9 animals retrieved. The

average of these values was obtained to get the overall precision.

An animal retrieved in the image search is considered correct if its color and type closely matches those of the animal in the original image.

The precision value obtained was **0.93**. The results obtained to calculate the Precision can be viewed in the Image Similarity Results chapter on the appendix section.

Overall, the system effectively retrieves similar animals, though it encounters some challenges when the original image has issues such as blurriness, poor quality, or when the animal is in an uncommon position, like being curled up.

6.3 User Evaluation

6.3.1 Objective

In the user evaluation phase, the objective is to measure the satisfaction of users as they complete tasks in the application. The user evaluation was conducted through a questionnaire. The application prototype was assessed in terms of usability, functionality, and user experience.

6.3.2 Methodology

The methodology used in developing the questionnaire aimed to obtain the maximum amount of information about the application. The application was evaluated with 10 participants. The questionnaire consists of four parts:

1. Obtaining demographic data and information about the users, such as gender, age, and daily time spent on the smartphone. Additionally, data about the user pet preferences, opinions on pet adoption, and preferred animals was also collected. This data is useful for characterizing the typical users of the application.

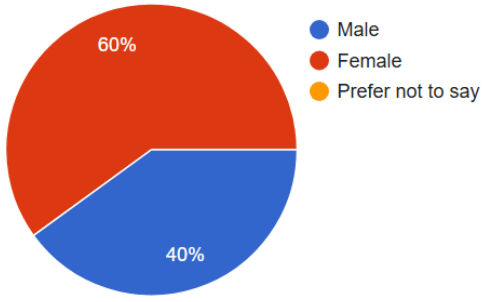
2. Measuring the usability of users in performing tasks. In this phase of the questionnaire, users have to complete a set of tasks and answer questions related to the usability of these tasks. This part helps identify the areas of the application where users experience the most difficulties.

3. Evaluating the overall usability of the application. This was achieved using the System Usability Scale questionnaire (SUS). This robust and widely used questionnaire includes simple questions with a 5-point Likert scale, ranging from Strongly Disagree to Strongly Agree. The resulting score ranges from 0 to 100, with a minimum desired score of 68.

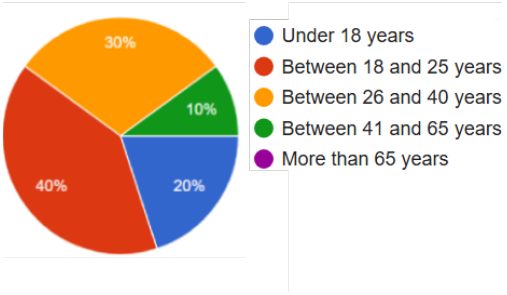
4. Evaluating the application in terms of user experience using the User Experience Questionnaire (UEQ). This quick and viable questionnaire measures the user experience of interactive products, assessing metrics such as attractiveness and originality. The questionnaire contains 26 questions.

6.3.3 Participants

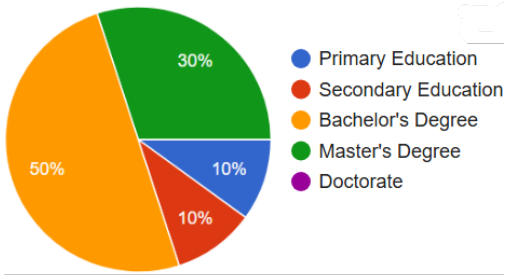
The participants who responded to the questionnaire are predominantly young people aged between 18 and 25 years, with strong knowledge of mobile applications, as it can be observed in Figure 6.16.



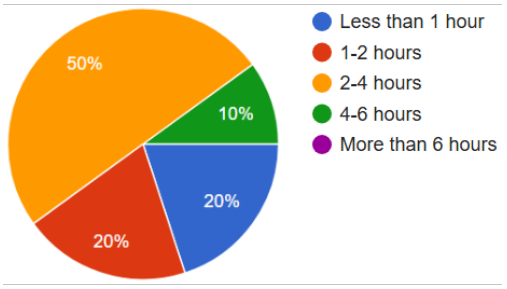
(a) Participants gender



(b) Participants age



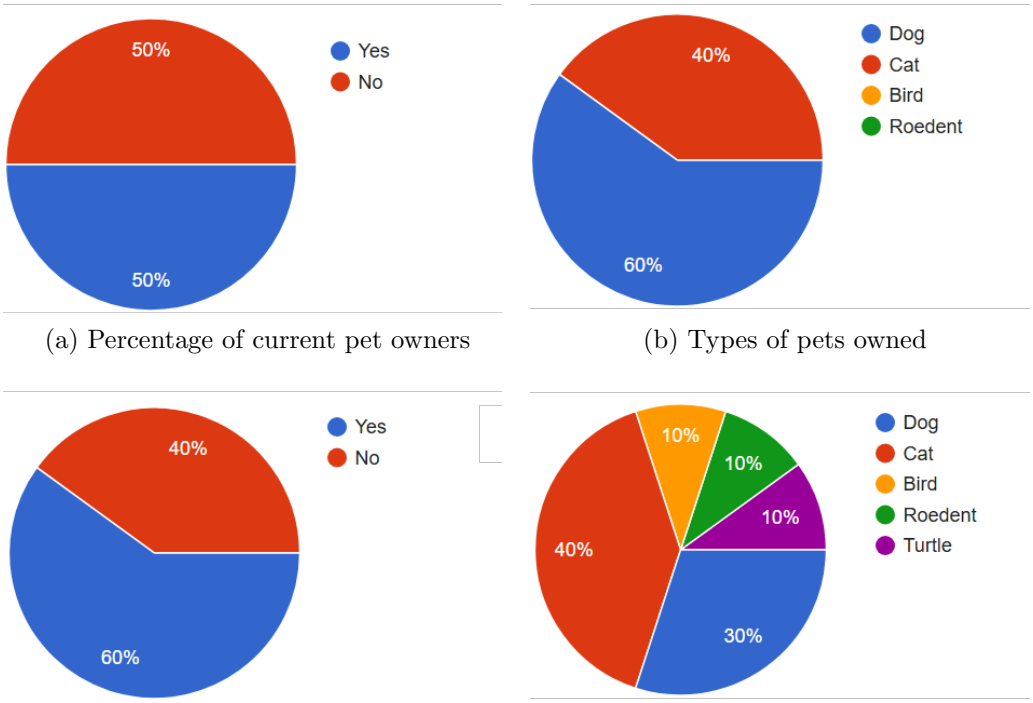
(c) Participants education level



(d) Participants daily phone time

Figure 6.16: Participants characterization

Most of the participants selected have an interest in adopting a pet or have already adopted a pet, making them the most likely to use the application. The participants were mostly interested in adopting cats and dogs, as illustrated in Figure 6.17.



(a) Percentage of current pet owners

(b) Types of pets owned

(c) Percentage of pet adopters

(d) Preferred type of pet for adoption

Figure 6.17: Participants pet interest

The participants showed greater interest in specific adoption requirements, including specific age range and if the pet is house trained, as illustrated in Figure 6.18.

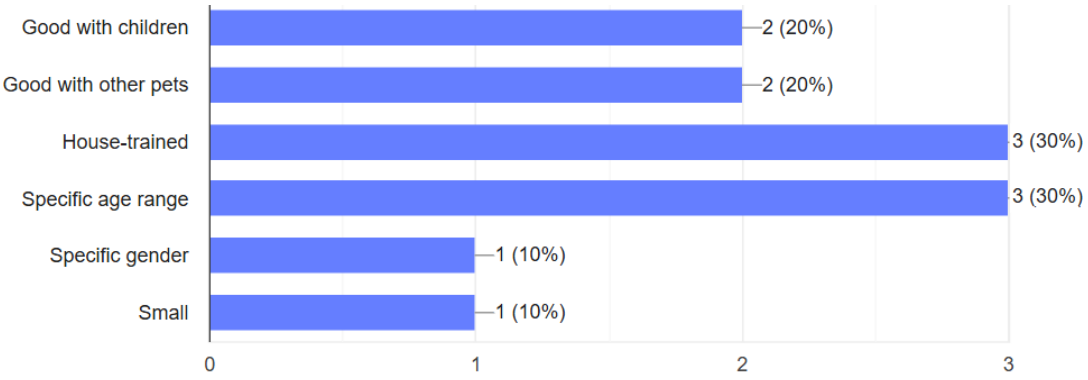
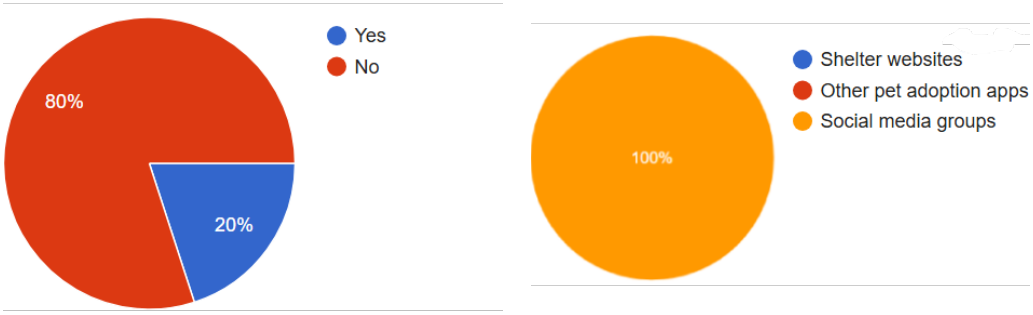


Figure 6.18: Specific adoption requirements

Most participants have never used a pet adoption platform before. The

ones who have, the platforms used were social media groups on applications like Facebook and Instagram, as illustrated in Figure 6.19.



(a) Percentage of participants that have used adoption platforms

(b) Pet adoption platforms used

Figure 6.19: Participants pet adoption platform usage

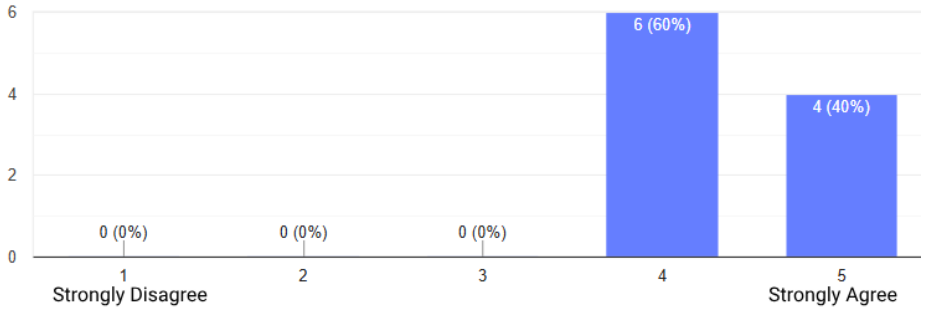
6.3.4 Usability assessment and task execution

This section evaluates the results of the tasks performed by the participants using the application.

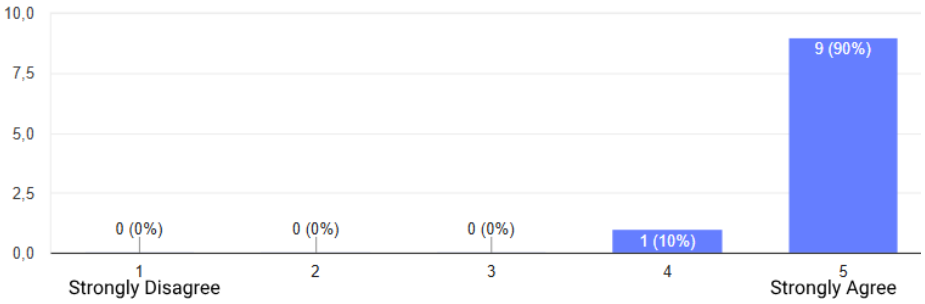
Task 1: Complete the Pet Recommendation Questionnaire

In this task, the participants filled out a questionnaire in the application designed to capture their pet adoption preferences. Most participants were satisfied with the questions in the questionnaire and found it relatively easy, as shown in Figure 6.20, responses for question (a) averaged **4.4**, while question (b) averaged **4.9** on a 0 to 5 scale.

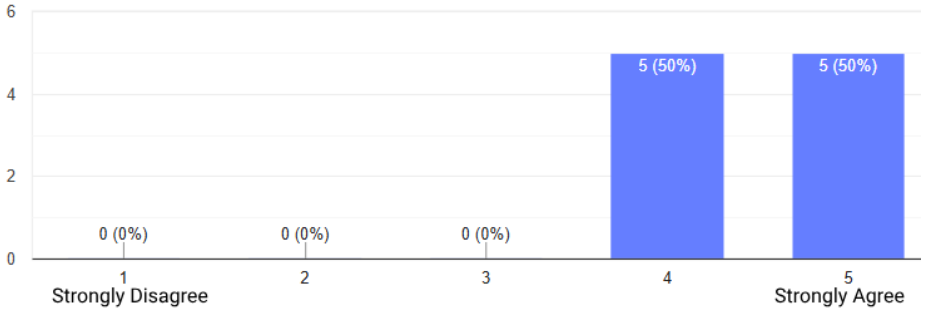
Overall the participants were satisfied with the recommended pets after completing the questionnaire, as illustrated in Figure 6.20, with question (c) responses averaging **4.5**. One of the participants mentioned that the questionnaire could be more interactive and fun, which is something that could be improved upon.



(a) Was the pet recommendation questionnaire easy to understand?



(b) Did the questions in the questionnaire feel relevant to your pet preferences?



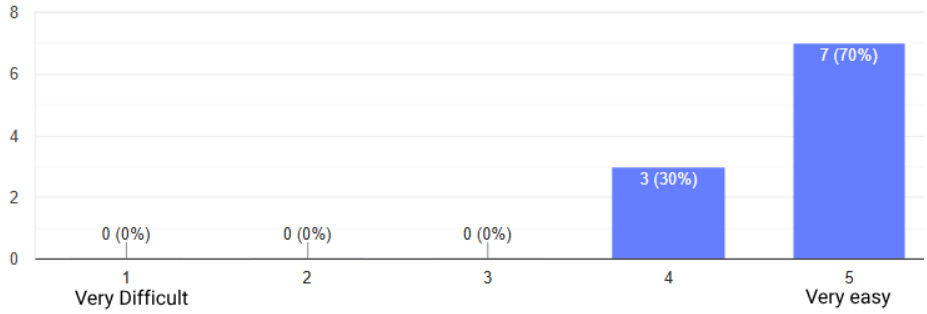
(c) After completing the questionnaire, were you satisfied with the recommended pets?

Figure 6.20: Task 1 results

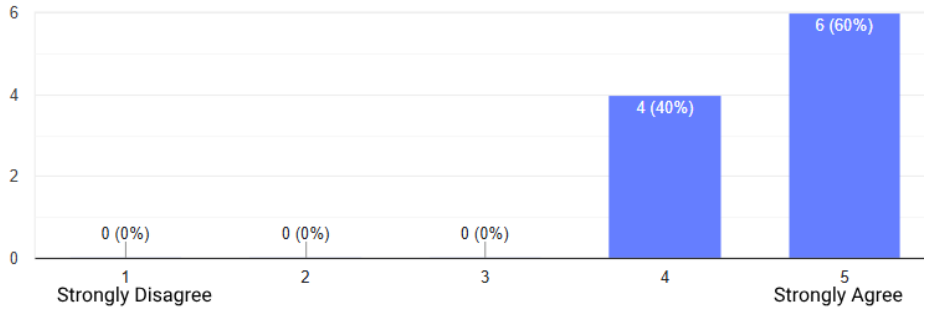
Task 2: Search for a Cat on the Search Page

In this task, the participants were asked to navigate to the search page and use the search filters to specifically look for cats. Most participants found the search page easy to locate and the search filters intuitive and easy to use, as shown in Figure 6.21, responses for question (a) averaged **4.7**, while question (b) averaged **4.6** on a 0 to 5 scale.

Three participants indicated that the page should have some sort of information while waiting for the pets to load, with one suggesting that there should be an indication to use the filters when the page is blank.



(a) On a scale of 1 to 5, how easy was it to locate the search page?



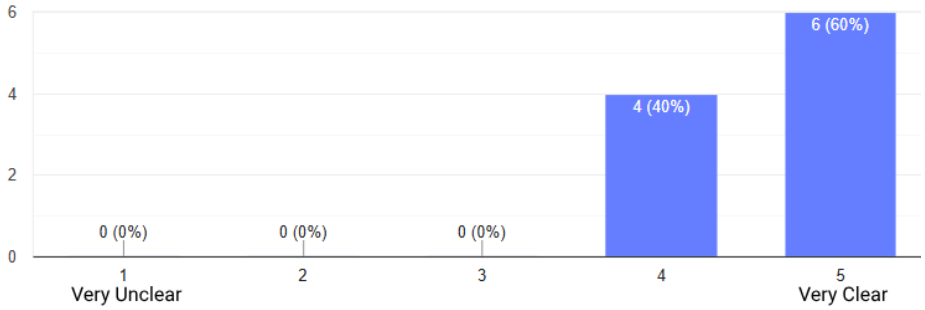
(b) Did you find the search filters intuitive and easy to use?

Figure 6.21: Task 2 results

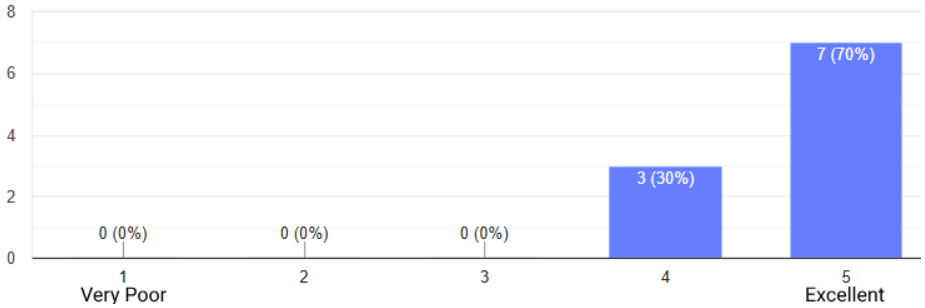
Task 3: View a Pet Profile

In this task, the participants were asked to view a pet profile and review the information provided about the pet, including its breed, age, color, and other characteristics.

The participants found the pet profile information clear and informative and overall had a good experience viewing the profile. As shown in Figure 6.22, responses for question (a) averaged **4.6**, while question (b) averaged **4.7** on a 0 to 5 scale. One participant suggested that the page should include if the pet is vaccinated.



(a) How clear and informative did you find the information presented on the pet profile?

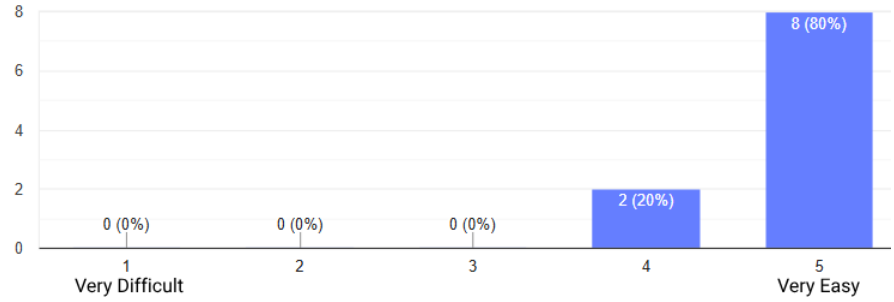


(b) How would you rate your overall experience with viewing a pet profile?

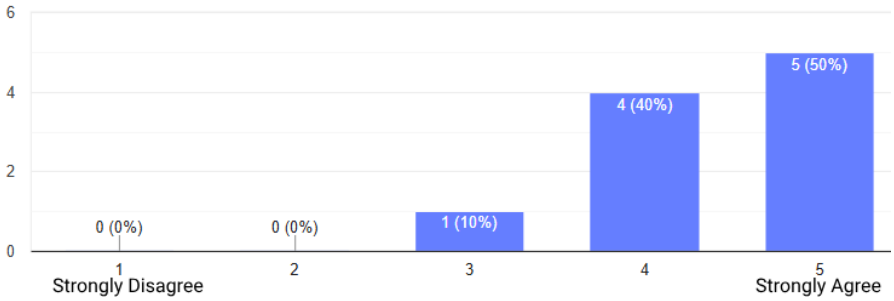
Figure 6.22: Task 3 results

Task 4: Add a Pet to Your Favorites

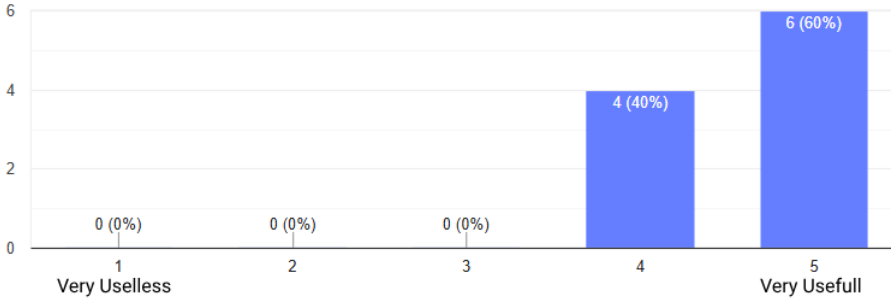
In this task, the participants were asked to add a pet to their favorites. Participants considered the task easy to perform, and found the favorites feature useful. As shown in Figure 6.23, responses for question (a) averaged **4.8**, question(b) averaged **4.2**, while question (c) averaged **4.6** on a 0 to 5 scale. Participants had some difficulty locating the favorite button and suggested it should be more visible, possibly by using a color like red.



(a) On a scale of 1 to 5, how easy was it to add a pet to your favorites?



(b) Did you find the favorite button easily on the pet profile page?



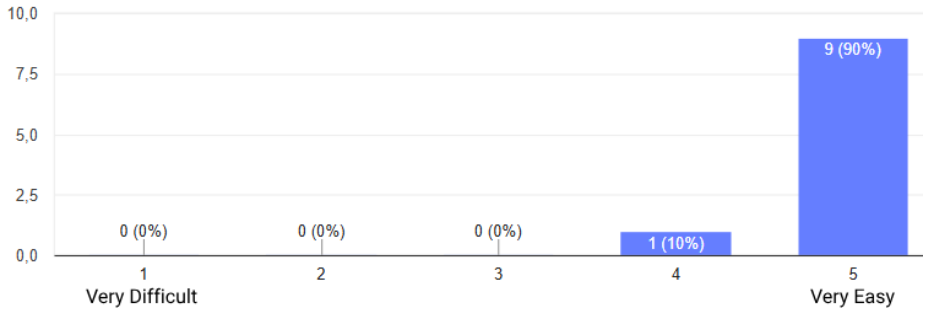
(c) How useful do you find the favorites feature in helping you decide which pet to adopt?

Figure 6.23: Task 4 results

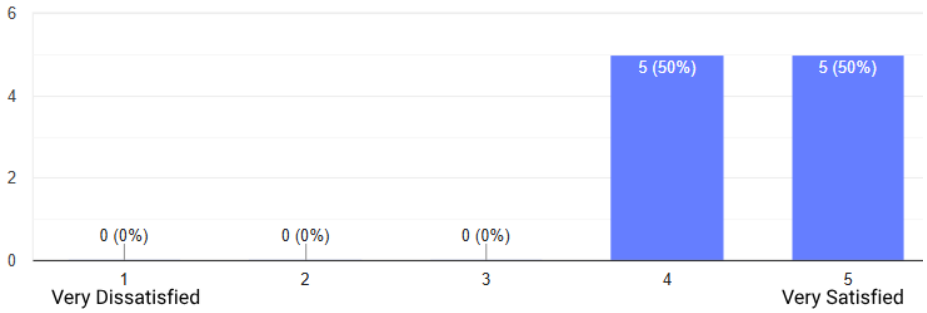
Task 5: Submit an Adoption Request for a Pet

In this task, the participants sent an adoption request to an owner of a pet they were interested in. Participants found the adoption request process easy to complete and were generally satisfied with it. As illustrated in Figure 6.24, responses for question (a) averaged **4.9**, while question (b) averaged **4.5** on a 0 to 5 scale.

Two participants suggested that the adoption request process should indicate more clearly that the request is being sent to the pet owner, which is something to be improved upon.



(a) How easy was it to locate the option to submit an adoption request?

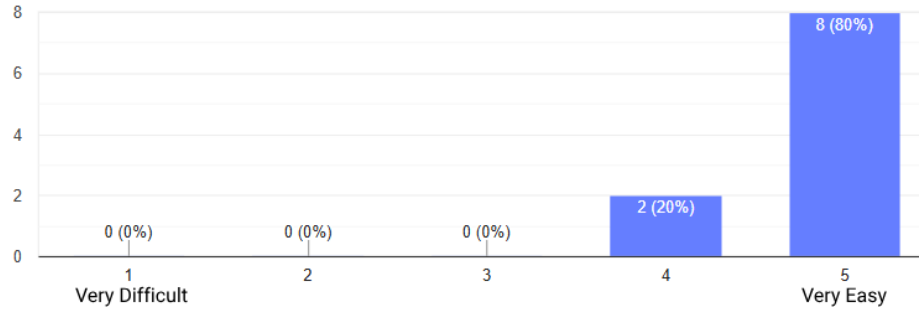


(b) How satisfied are you with the overall process of submitting an adoption request?

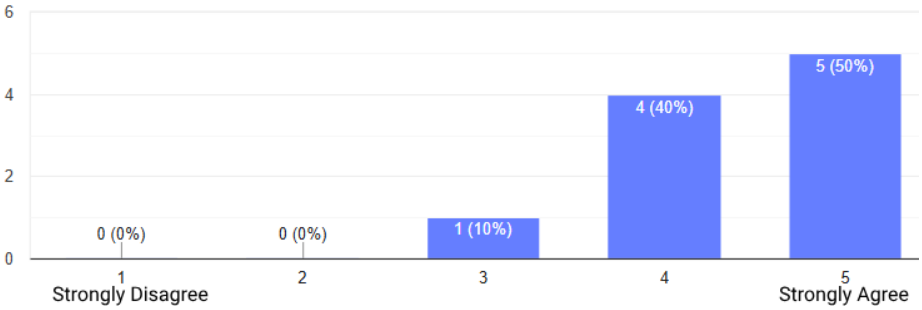
Figure 6.24: Task 5 results

Task 6: Add a New Pet to the Application

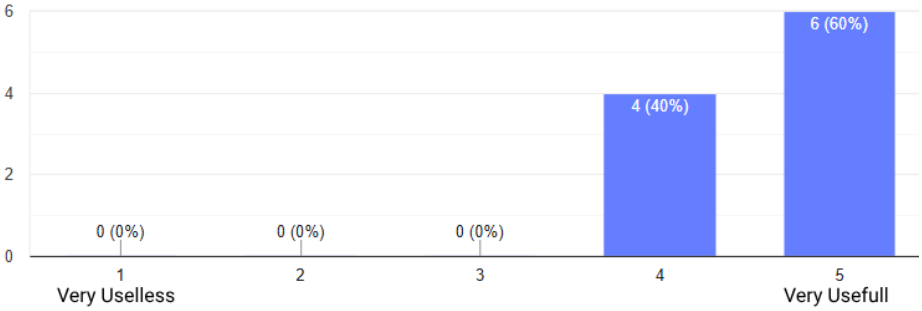
In this task, participants were asked to add a new pet to the application, filling in the required information about the pet. Participants found this task easy to perform and were satisfied with the overall process of adding a new pet. As shown in Figure 6.25, responses for question (a) averaged **4.8**, question(b) averaged **4.2**, while question (c) averaged **4.6** on a 0 to 5 scale.



(a) How easy was it to find the option to add a new pet?



(b) How clear were the instructions and fields required to add a new pet?

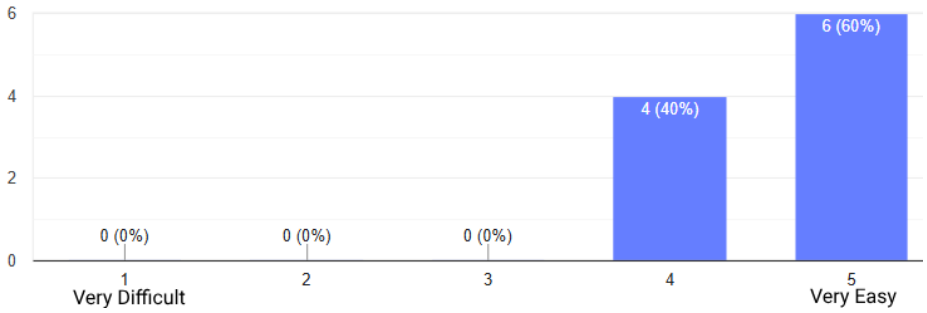


(c) How satisfied are you with the overall process of adding a new pet?

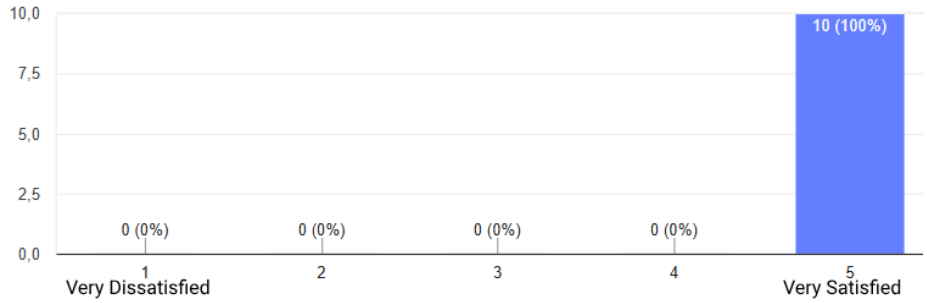
Figure 6.25: Task 6 results

Task 7: Search for a pet by image

In this task, participants searched for pets by inserting an image from their image gallery. Most participants found the image search feature easy to locate. However, one participant suggested that the search bar could include a reference to the image search feature. Overall the participants were satisfied with the image results. As illustrated in Figure 6.26, responses for question (a) averaged **4.6**, while question (b) averaged **5** on a 0 to 5 scale.



(a) How easy was it to find the option to perform image search?



(b) How satisfied are you with the image search results?

Figure 6.26: Task 7 results

6.4 System Usability Scale (SUS) Results

To obtain the System Usability Scale results, it is necessary to calculate the average of the participants' responses for each question. There are 10 questions divided into positive (odd-numbered) and negative (even-numbered) questions.

To calculate the SUS score, the following criteria was followed:

- For odd-numbered questions, subtract 1 from the participant's score.
- For even-numbered questions, subtract the participant's score from 5.
- Sum all the scores from each question.
- Multiply the total by 2.5 to obtain the final SUS score.

The final SUS score was **88**. This result is graded as "A", placing it in the range of promoter, acceptable, and excellent scores, as illustrated in Figure 6.27, indicating a strong usability evaluation.

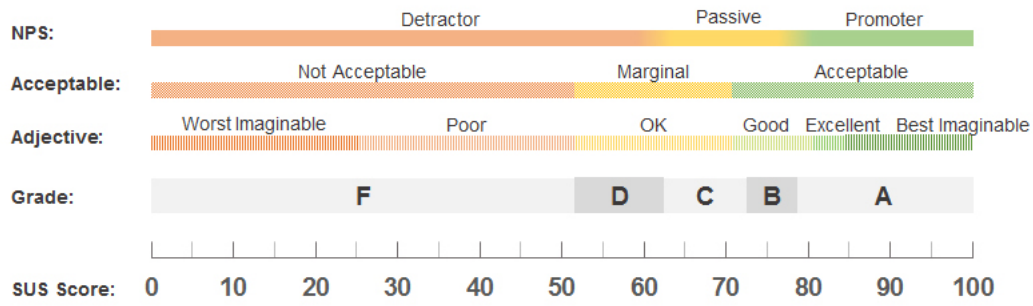


Figure 6.27: SUS scale score (source: `https://measuringu.com/interpret-sus-score/`)

6.5 User Experience Questionnaire (UEQ)

The participants answered the User Experience Questionnaire (UEQ), it is used to measure the user experience and consists of 26 questions that cover the following dimensions:

- **Attractiveness**: The overall impression of the product, whether users liked it or not.
- **Perspicuity**: How easy it is to become familiar with the product and learn how to use it.
- **Efficiency**: Whether users can complete tasks without unnecessary effort.
- **Dependability**: Whether users feel in control of the interaction, and if it is safe and predictable.
- **Stimulation**: Whether using the product is exciting and motivating, and if it's enjoyable to use.
- **Novelty**: Whether the product's design is creative and captures users' interest.

Each question is rated on a 7-point Likert scale. Responses are transformed to a scale from -3 to +3, with higher scores indicating a more positive user experience. The average scores for each dimension are then calculated, assessing the overall user satisfaction.

In Figure 6.28, the chart displays the mean value of each dimension along with its confidence level. The strongest dimensions are attractiveness and perspicuity, while novelty lags significantly behind the others. A possible reason is that this application follows user interface and navigation styles similar to those of popular mobile apps such as Instagram.

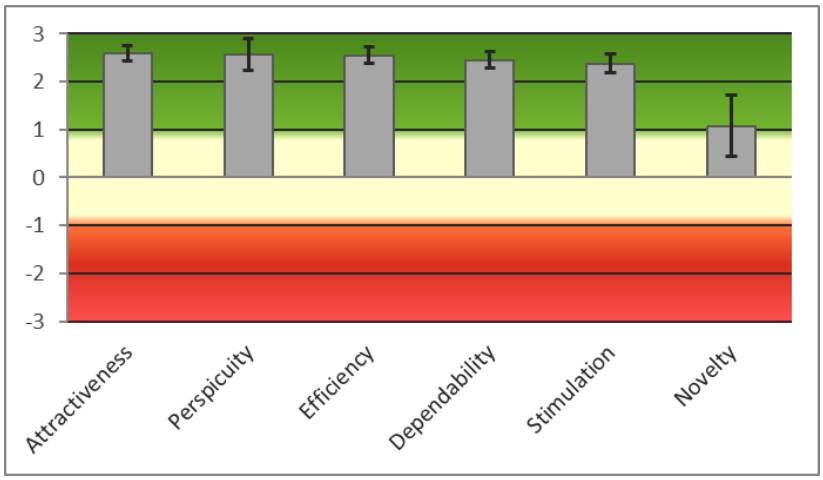


Figure 6.28: Graph of mean values and confidence levels for each parameter

The results for hedonic and pragmatic quality were also analyzed. Pragmatic quality describes task related quality aspects, it refers to the utility, efficiency, and performance of the application. Hedonic quality describes non-task related quality aspects, it refers to the subjective experience of pleasure associated with the application, emphasizing the emotional dimension of the user experience.

Figure 6.29 presents the average of the three pragmatic and hedonic quality aspects. As observed, the application demonstrates very strong pragmatic quality, while hedonic quality lags slightly behind but still delivers a result that inspires confidence.

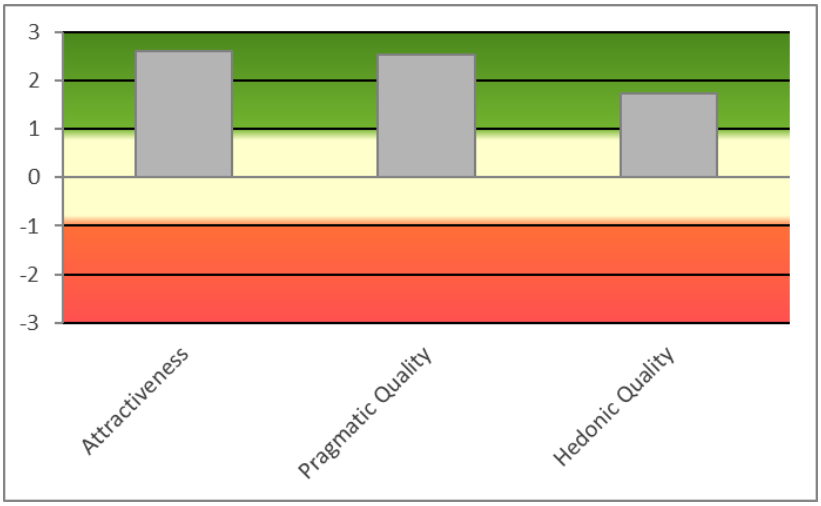


Figure 6.29: Graph of hedonic and pragmatic quality

Finally, the measured scaled mean values for the six UEQ parameters are compared to values from a benchmark data set. This data set contains responses from 21,175 participants across 468 studies involving various products such as business software, websites, web shops, and social networks. The comparison of our application results with the data in the benchmark allows us to draw conclusions about certain aspects of the application and identify areas for improvement.

As it can be observed in Figure 6.30, except for novelty, all the parameters have an excellent result compared to the benchmark. Novelty, however, only shows an above-average result relative to the benchmark.

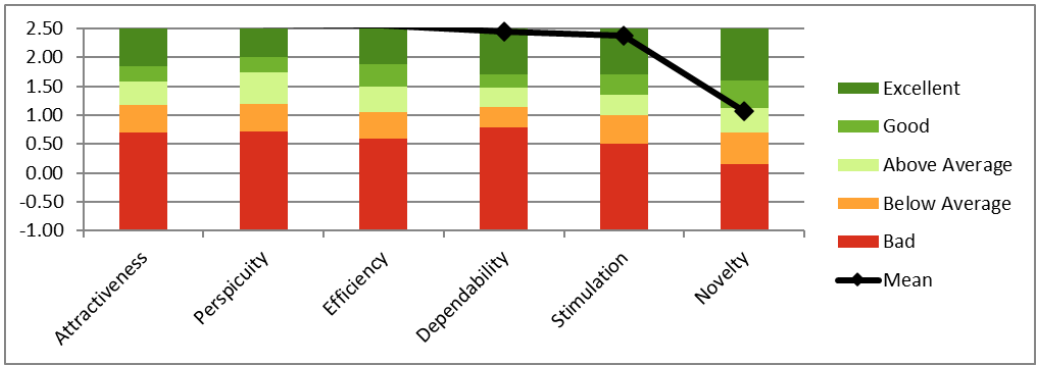


Figure 6.30: Comparison chart of application results and benchmark data

Chapter 7

Conclusions and Future Work

This project provided valuable insights and contributions for the animal adoption problem. It demonstrated that a pet adoption application focused on user experience and interaction can be effective, as evidenced by a positive user evaluation.

The project showcased the potential of sentence embeddings derived from a pre-trained sentence transformer model, combined with k-means clustering to group information based on semantic content. This approach was applied to create a recommendation system based on user pet preferences.

The image search feature was particularly valued by users, offering an alternative method for finding pets beyond traditional filtering or text queries. This feature was based on convolutional neural networks, illustrating the efficiency of using pre-trained Keras models for feature extraction. When combined with similarity metrics like cosine similarity, this approach proved effective in identifying similar images.

This project demonstrated the usefulness of React Native for building cross-platform mobile applications, using the power of components and multiple libraries to construct the application quickly and effectively.

Using Flask to develop the recommendation and image search API in Python facilitated rapid development and offered significant flexibility.

It was observed that the use of Google Cloud Firebase services significantly reduced the necessary time to build the application. This efficiency was achieved through tools for managing the application, such as user authentication, storage, and a NoSQL database. However, there were some limitations, including vendor lock-in and the lack of traditional SQL querying and relational data management capabilities.

One of the limitations of this project was the lack of computational power. Due to insufficient resources, extracting the feature vectors from the trained CNN took longer than expected, making it very hard to conduct a more in-depth study using different types of pre-trained models. Another limitation was the relatively small dataset, consisting of only about 20,000 animals. This limited the ability to assess the speed of the recommendation and image search results with a much larger dataset. A larger dataset would also have provided better insight into how the trained clustering model handles more extensive data.

A potential enhancement for future work includes adding new functionalities to the application. This could involve an administrative panel for managing users and pets, a page for users to provide feedback on successful adoptions, a section offering advice and guidelines on adopting different types of pets, and support for multiple languages.

Currently, the recommendation system derives user pet preferences solely from responses to the pet recommendation questionnaire. An improved approach would be to also gather user preferences through the user's interactions with the system, such as adding a pet to favorites or visiting a pet's page.

For future improvement, fine-tuning the pre-trained Keras model on a dataset with animal images of the species in the application could enhance performance for image similarity. This way the model could learn domain-specific features better and provide more accurate feature vectors for image similarity results. Another potential improvement would be to use the Neural Network's classification to limit the similarity calculation to a specific class,

which could significantly speed up the process by reducing the number of images for comparison.

Bibliography

[1] Zawistowski, Stephen, Julie Morris, M. D. Salman, and Rebecca Ruch-Gallie. "Population Dynamics, Overpopulation, and the Welfare of Companion Animals: New Insights on Old and New Data." Journal of Applied Animal Welfare Science, July 1, 1998.

[2] Kajbaje, Sumit, Rohit Sawant, Ronit Loke, and Vishwajeet Patil. "AI-Based Pet Adoption System" 09, no. 04 (2022).

[3] Shah, Lipi, Hetal Gaudani, and Prem Balani. "Survey on Recommendation System." International Journal of Computer Applications 137 (March 17, 2016).

[4] Aizawa, Akiko. "An Information-Theoretic Perspective of Tf–Idf Measures." Information Processing Management 39, no. 1 (January 1, 2003): 45–65.

[5] Putri, Mariani Widia, Achmad Muchayan, and Made Kamisutara. "Sistem Rekomendasi Produk Pena Eksklusif Menggunakan Metode Content-Based Filtering dan TF-IDF." JOINTECS (Journal of Information Technology and Computer Science) 5, no. 3 (September 30, 2020): 229.

[6] Mirjalili, Seyedali. "Genetic Algorithm." In Evolutionary Algorithms and Neural Networks: Theory and Applications, edited by Seyedali Mirjalili,

43–55. Cham: Springer International Publishing, 2019.

[7] Xu, Rui, and Don Wunsch. Clustering. John Wiley Sons, 2008.

[8] Isinkaye, F. O., Y. O. Folajimi, and B. A. Ojokoh. “Recommendation Systems: Principles, Methods and Evaluation.” Egyptian Informatics Journal 16, no. 3 (November 1, 2015): 261–73.

[9] Ahmed, Mohiuddin, Raihan Seraj, and Syed Mohammed Shamsul Islam. “The K-Means Algorithm: A Comprehensive Survey and Performance Evaluation.” Electronics 9, no. 8 (August 2020): 1295.

[10] Chowdhary, K. R. “Natural Language Processing.” In Fundamentals of Artificial Intelligence, edited by K.R. Chowdhary, 603–49. New Delhi: Springer India, 2020.

[11] “Sentence Transformers: Meanings in Disguise — Pinecone.” Accessed August 18, 2024.

[12] “Bidirectional Recurrent Neural Networks — IEEE Journals Magazine — IEEE Xplore.” Accessed August 18, 2024.

[13] Cheng, Jianpeng, Li Dong, and Mirella Lapata. “Long Short-Term Memory-Networks for Machine Reading.” arXiv, September 20, 2016.

[14] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate.” arXiv, May 19, 2016.

[15] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need," n.d.

[16] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." arXiv, May 24, 2019.

[17] "Bidirectional Recurrent Neural Networks — IEEE Journals Magazine — IEEE Xplore." Accessed August 18, 2024.

[18] "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects — IEEE Journals Magazine — IEEE Xplore." Accessed August 10, 2024.
https://ieeexplore.ieee.org/abstract/document/9451544/

[19] Chen, Luyang, and Serra Mall. "Image-Based Product Recommendation System with Convolutional Neural Networks," n.d.

[20] Koonce, Brett. "ResNet 50." In Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization, edited by Brett Koonce, 63–72. Berkeley, CA: Apress, 2021.

[21] Isinkaye, F. O., Y. O. Folajimi, and B. A. Ojokoh. "Recommendation Systems: Principles, Methods and Evaluation." Egyptian Informatics Journal 16, no. 3 (November 1, 2015): 261–73.

[22] "Similarity Metrics." Accessed August 10, 2024.
http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/mvoget/similarity/similarity.html

[23] Pherwani, Rhea. “UX Case Study — A Pet Adoption App.” Medium (blog), November 27, 2022. https://medium.com/@rheapherwani72/ux-case-study-a-pet-adoption-app-35fe4f8582ca

[24] “Petfinder” Accessed August 8, 2024. https://www.petfinder.com/

[25] ”PetMatch pet adoption platform” Accessed August 8, 2024. https://petmatch.ie/

[26] “Animal Shelter Data.” Accessed August 10, 2024. https://data.longbeach.gov/explore/dataset/animal-shelter-intakes-and-outcomes/table/?disjunctive.animal_type&disjunctive.primary_color&disjunctive.sex&disjunctive.intake_cond&disjunctive.intake_type&disjunctive.reason&disjunctive.outcome_type&disjunctive.outcome_subtype&disjunctive.intake_is_dead&disjunctive.outcome_is_dead

[27] Sonoma County. “Animal Shelter Intake and Outcome — Open Data.” Accessed August 10, 2024. https://data.sonomacounty.ca.gov/Government/Animal-Shelter-Intake-and-Outcome/924a-vesw

[28] Richardson, Leonard, and Sam Ruby. RESTful Web Services. O’Reilly Media, Inc., 2008.

[29] “All You Need to Know About Building a Mobile App Backend,” December 14, 2022. https://blog.back4app.com/how-to-build-a-backend-for-a-mobile-app/

[30] Holovaty, Adrian, and Jacob Kaplan-Moss. The Definitive Guide to Django: Web Development Done Right. Apress, 2009.

[31] Hamm, Matthew J. Wireframing Essentials. Packt Publishing Ltd, 2014.

[32] "PetFinder.My Adoption Prediction." Accessed August 11, 2024. https://kaggle.com/competitions/petfinder-adoption-prediction

[33] Team, Keras. "Keras Documentation: Keras Applications." Accessed August 10, 2024. https://keras.io/api/applications/

[34] Chollet, Francois. "Xception: Deep Learning With Depthwise Separable Convolutions," 1251–58, 2017. https://openaccess.thecvf.com/content_cvpr_2017/html/Chollet_Xception_Deep_Learning_CVPR_2017_paper.html

[35] Team, Keras. "Keras Documentation: InceptionV3." Accessed August 10, 2024. https://keras.io/api/applications/inceptionv3/

[36] scikit-learn. "KNNImputer." Accessed August 10, 2024. https://scikit-learn/stable/modules/generated/sklearn.impute.KNNImputer.html

[37] "Sentence-Transformers/Paraphrase-MiniLM-L6-v2 · Hugging Face." Accessed August 12, 2024. https://huggingface.co/sentence-transformers/paraphrase-MiniLM-L6-v2

[38] Kurita, Takio. "Principal Component Analysis (PCA)." In Computer Vision: A Reference Guide, 1–4. Cham: Springer International Publishing, 2019.

[39] KDnuggets. "LGBMClassifier: A Getting Started Guide." Accessed August 10, 2024. https://www.kdnuggets.com/

lgbmclassifier-a-getting-started-guide

[40] "Shap.TreeExplainer — SHAP Latest Documentation." Accessed August 10, 2024. https://shap-lrjball.readthedocs.io/en/latest/generated/shap.TreeExplainer.html

[41] CR, Aravind. "Exploring Clustering Algorithms: Explanation and Use Cases." neptune.ai, July 22, 2022. https://neptune.ai/blog/clustering-algorithms

Link

GitHub repository: https://github.com/DuarteDomingues/petto_mobile_application

Wireframes link: https://www.figma.com/design/I6SAH9cxGp7QpieyHqOMi8/Untitled?node-id=1-3&t=omNlOzem41ScL3Ls-1

Image Similarity Results

Figure 1: Image similarity example with ResNet50 model

Figure 2: Image similarity example with ResNet50 model

Figure 3: Image similarity example with ResNet50 model

Figure 4: Image similarity example with ResNet50 model

Figure 5: Image similarity example with ResNet50 model

Recommendation Results

Input: 'Species: CAT, Age: BABY, Gender: FEMALE, Breed: Persian, Color: BLACK, Size: MEDIUM, Health: HEALTHY, Sterilized: NO'

Recommended pets

	Species	Age	Gender	Breed	Color	Size	Health	Sterilized	PetID
1530	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	37c989ac5
2733	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	64cf73826
2668	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	6258b0d76
3998	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	940b0ad69
2449	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	59047a5a1
4319	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	9efd6ee7c
1792	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	4158fbb67
4891	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	b2e138d7a
1414	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	33dc6b53a
1260	CAT	BABY	FEMALE	Persian	BLACK	MEDIUM	HEALTHY	NO	2e65d72c2

Figure 6: Pet recommendation example

Input: 'Species: CAT, Age: ADULT, Gender: FEMALE, Breed: Taby, Color: BLACK, Size: MEDIUM, Health: HEALTHY, Sterilized: NO'

Recommended pets

	Species	Age	Gender	Breed	Color	Size	Health	Sterilized	PetID
1739	CAT	ADULT	FEMALE	Taby	BLACK	MEDIUM	HEALTHY	YES	3fc0ccb82
1869	CAT	ADULT	FEMALE	Taby	BLACK	MEDIUM	HEALTHY	YES	4448f7d57
6061	CAT	YOUNG	FEMALE	Taby	BLACK	MEDIUM	HEALTHY	NO	dde3750a1
2904	CAT	BABY	FEMALE	Taby	BLACK	MEDIUM	HEALTHY	NO	6b6cb317b
1905	CAT	BABY	FEMALE	Taby	BLACK	MEDIUM	HEALTHY	NO	4555f9273
1730	CAT	BABY	FEMALE	Taby	BLACK	MEDIUM	HEALTHY	NO	3f50cc4fd
6459	CAT	BABY	FEMALE	Taby	BLACK	MEDIUM	HEALTHY	NO	ec7eecc57
5271	CAT	BABY	FEMALE	Taby	BLACK	MEDIUM	HEALTHY	NO	c0a0a48a0
6047	CAT	BABY	FEMALE	Taby	BLACK	MEDIUM	HEALTHY	NO	dd5275a60
5611	CAT	BABY	FEMALE	Taby	BLACK	MEDIUM	HEALTHY	NO	cce8fa6c2

Figure 7: Pet recommendation example

Input: 'Species: CAT, Age: ADULT, Gender: FEMALE, Breed: Persian, Color: WHITE, Size: MEDIUM, Health: HEALTHY, Sterilized: YES'

Recommended pets

	Species	Age	Gender	Breed	Color	Size	Health	Sterilized	PetID
363	CAT	ADULT	FEMALE	Persian	WHITE	MEDIUM	HEALTHY	YES	0d1dd21e5
4598	CAT	ADULT	FEMALE	Persian	WHITE	MEDIUM	HEALTHY	NO	a8ba21133
6984	CAT	ADULT	FEMALE	Persian	WHITE	SMALL	HEALTHY	YES	fe467b074
7004	CAT	YOUNG	MALE	Persian	WHITE	LARGE	HEALTHY	YES	ff14c3dbe
679	CAT	BABY	MALE	Persian	WHITE	MEDIUM	HEALTHY	NO	1940d9eb3
3029	CAT	BABY	MALE	Persian	WHITE	MEDIUM	HEALTHY	NO	6fc17b073
1783	CAT	BABY	MALE	Persian	WHITE	MEDIUM	HEALTHY	NO	40fa64194
4699	CAT	BABY	MALE	Persian	WHITE	MEDIUM	HEALTHY	NO	abe8790ba
5992	CAT	BABY	MALE	Persian	WHITE	MEDIUM	HEALTHY	NO	dafa3b086
1768	CAT	BABY	MALE	Persian	WHITE	MEDIUM	HEALTHY	NO	409ad5a23

Figure 8: Pet recommendation example

Input: 'Species: CAT, Age: ADULT, Gender: FEMALE, Breed: Domestic Short Hair, Color: BLACK, Size: SMALL, Health: HEALTHY, Sterilized: YES'

↓ Recommended pets

	Species	Age	Gender	Breed	Color	Size	Health	Sterilized	PetID
4593	CAT	ADULT	FEMALE	Domestic Short Hair	BLACK	SMALL	HEALTHY	YES	a89d4e8f3
992	CAT	ADULT	FEMALE	Domestic Short Hair	BLACK	SMALL	HEALTHY	YES	249af4201
4559	CAT	ADULT	FEMALE	Domestic Short Hair	BLACK	SMALL	HEALTHY	YES	a73cb1747
4321	CAT	ADULT	FEMALE	Domestic Short Hair	BLACK	SMALL	HEALTHY	YES	9f01cfff1
3895	CAT	ADULT	FEMALE	Domestic Short Hair	BLACK	LARGE	HEALTHY	YES	906ee1877
6327	CAT	ADULT	FEMALE	Domestic Short Hair	BLACK	LARGE	HEALTHY	YES	e84771c5d
2294	CAT	ADULT	FEMALE	Domestic Short Hair	BLACK	LARGE	HEALTHY	YES	5328a3ad8
3487	CAT	ADULT	FEMALE	Domestic Short Hair	BLACK	LARGE	HEALTHY	YES	818faec45
2227	CAT	ADULT	FEMALE	Domestic Short Hair	BLACK	LARGE	HEALTHY	YES	50811e391
1051	CAT	ADULT	FEMALE	Domestic Short Hair	BLACK	MEDIUM	HEALTHY	YES	2679a20fe

Figure 9: Pet recommendation example

Input: 'Species: CAT, Age: ADULT, Gender: Male, Breed: Maine Coon, Color: WHITE, Size: MEDIUM, Health: HEALTHY, Sterilized: NO'

↓ Recommended pets

	Species	Age	Gender	Breed	Color	Size	Health	Sterilized	PetID
3532	CAT	BABY	FEMALE	Maine Coon	WHITE	MEDIUM	HEALTHY	NO	8368a0a44
5041	CAT	BABY	MALE	Maine Coon	GOLDEN	MEDIUM	HEALTHY	NO	b85cdc8a5
1482	CAT	YOUNG	MALE	Maine Coon	CREAM	LARGE	HEALTHY	NO	36681852f
976	CAT	BABY	MALE	Maine Coon	GRAY	SMALL	HEALTHY	NO	2425e8c69
4126	CAT	BABY	FEMALE	Maine Coon	GOLDEN	MEDIUM	HEALTHY	NO	982ffd67f
696	CAT	BABY	FEMALE	Maine Coon	BROWN	MEDIUM	HEALTHY	NO	19b80fb4d
2411	CAT	BABY	FEMALE	Maine Coon	BROWN	MEDIUM	HEALTHY	NO	5781e566b
1246	CAT	BABY	FEMALE	Maine Coon	BROWN	MEDIUM	HEALTHY	NO	2def67952
6440	CAT	YOUNG	FEMALE	Maine Coon	BROWN	LARGE	HEALTHY	NO	ec250ea11
4900	CAT	ADULT	MALE	Maine Coon	BLACK	LARGE	HEALTHY	YES	b354aa023

Figure 10: Pet recommendation example